A Recovery-Assisted Discontinuous Galerkin Method for Direct Numerical Simulation of Compressible Turbulence

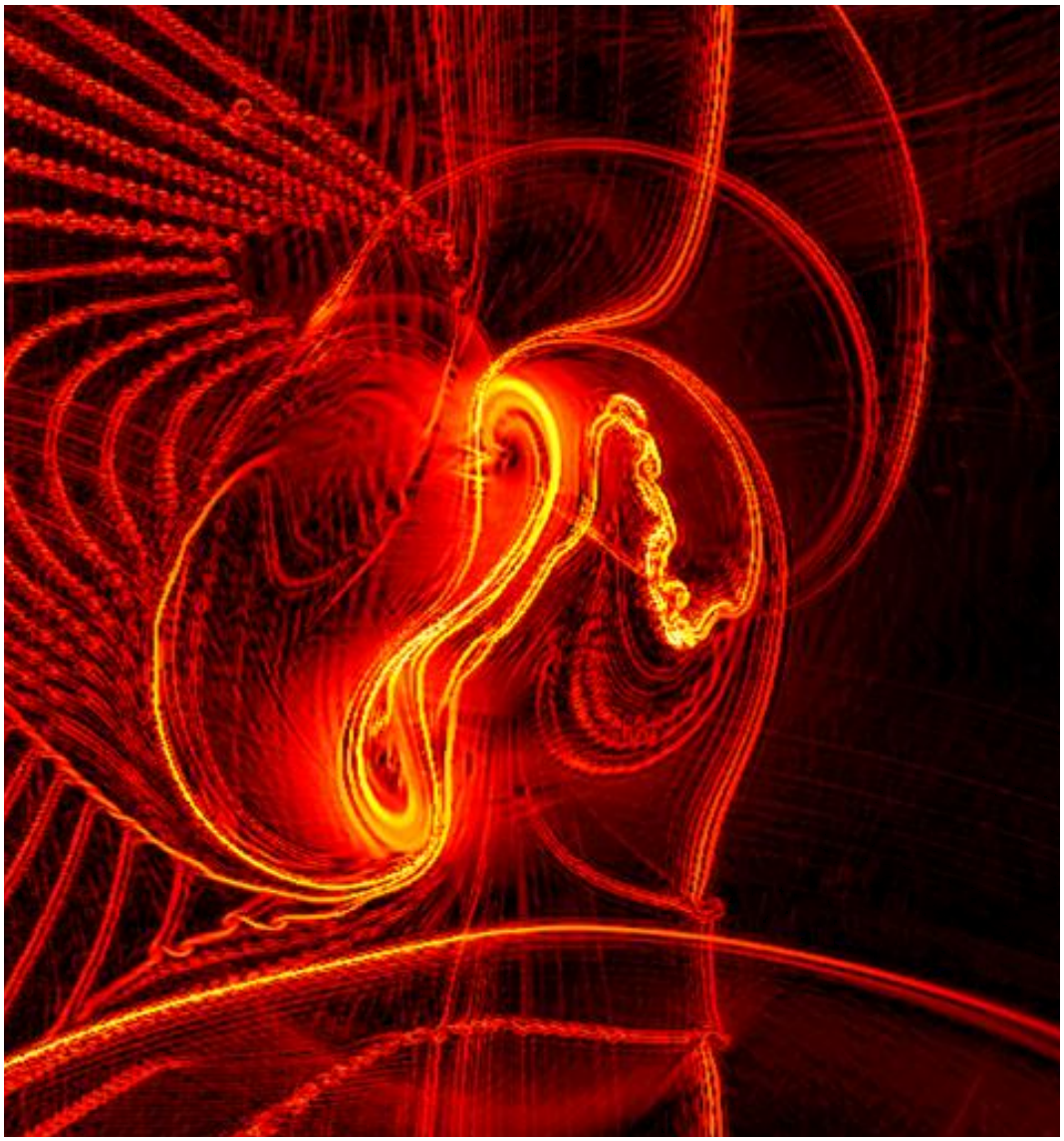
by

Philip E. Johnson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2019

Doctoral Committee:

Associate Professor Eric Johnsen, Chair
Associate Professor Krysztof Fidkowski
Professor Krishnakumar Garikipati
Dr. H.T. Huynh, NASA Glenn Research Center
Professor Philip L. Roe



Philip E. Johnson
phedjohn@umich.edu
ORCID iD: 0000-0002-2164-3705

ACKNOWLEDGMENTS

I thank Marc Henry de Frahan and the rest of the Scientific Computing & Flow Physics Laboratory for fruitful conversations regarding numerical methods for compressible flow. I would also like to thank Professor Philip L. Roe, Professor Eric Johnsen, Professor Krishnakumar Garikipati, Professor Krzysztof Fidkowski, and Dr. H.T. Huynh for taking the time to serve on the dissertation committee, especially the committee chair (and my advisor), Eric Johnsen. He has consistently served not only as a patient mentor during my time at Michigan, but also as an advocate in my career development, and for that I am grateful.

Dr. Huynh deserves special recognition for taking the time to serve as my mentor during my short internship at NASA Glenn Research Center; he was especially helpful in his determination to make sure the information technology folks at Glenn promptly resolved my software problems. I am also thankful to various members of the Inlets & Nozzles branch of NASA Glenn for offering their own candid anecdotes when asked about the intricacies of working for NASA.

A significant portion of the funding for this work has been provided by a Rackham Merit Fellowship from the University of Michigan. The larger parallel calculations required for this study have been performed on the Conflux cluster at the University of Michigan, provided by the NSF via grant 1531752 MRI: Acquisition of Conflux, A Novel Platform for Data-Driven Computational Physics (Tech. Monitor: Ed Walker). Todd Raeker's guidance proved indispensable when porting our DG code to the Conflux cluster. In addition to the Conflux cluster, this work made use of the Extreme Science and Engineering Discovery Environment (XSEDE) Comet cluster at San Diego State University. XSEDE is supported by National Science Foundation grant number ACI-1548562.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Computational Fluid Dynamics (CFD) serves as a valuable complement to analytical and experimental methods in the study of fluid mechanics. However, the engineering and fundamental research communities are continually plagued by the conflict between solution accuracy and computational resource availability. Increasingly accurate simulations demand more computational memory and more wall time, so many flows of interest, particularly turbulent flows at moderate to high Reynolds numbers, cannot be simulated with high fidelity. This issue motivates further development of high-order CFD methods, which promise to deliver better return on investment with respect to accuracy versus computational resource consumption compared to second-order methods (the standard in applied CFD). In the high-order community, the discontinuous Galerkin (DG) method is a popular approach due to its capability for arbitrarily high orders of accuracy and near-trivial extension to unstructured meshes and distributed-memory architectures. In this work, the state-of-the-art DG method is paired with a relatively new tool known as the recovery operator. The objective is to improve the resolution properties of the DG method in the context of advection-diffusion systems (such as the compressible Navier-Stokes equations), thereby facilitating more accurate simulations of turbulent flows for a given gridpoint count and solution order p.

To discretize the advective fluxes of the governing PDE(s), a biased version of the recovery operator is combined with the traditional upwind DG formulation. This simple modification improves the DG method's order of accuracy while retaining its well-regarded tendency to damp away spurious nonphysical oscillations. Where the traditional upwind DG formulation approximates the exact translation operator with order $2p + 1$ accuracy, the new scheme achieves order $2p + 2$ accuracy. Optimal convergence is achieved in the global L_2 norm. For the diffusive flux terms, the recovery operator is combined with the traditional mixed formulation to leverage the extraordinary

accuracy of the recovery operator while maintaining stability on a compact computational stencil. The result of our efforts is a new advection-diffusion discretization, now known as the *Recovery-assisted DG* method. Detailed analysis and a comprehensive suite of test problems are presented to demonstrate that for flows containing a broad range of length scales, the new approach is consistently more accurate than the state-of-the-art DG method for the compressible Navier-Stokes equations. As a bonus topic, we show how the familiar Fourier analysis technique can be extended to predict the performance of Flux Reconstruction methods (including a novel Recovery-assisted formulation) on unstructured meshes.

Overall, this study indicates that with regard to error minimization at fixed computational cost, manipulation of interface flux terms in the DG weak form is a reliable path to scheme improvement; in our case, the performance gain is achieved via the recovery operator. Fourier analysis proves the Recovery-assisted method superior to the conventional approach for linear problems. The exceptional performance of the new method extends to 3D turbulence simulations, validating the general scheme development philosophy of performing detailed analysis with simple model PDEs. By design, the Recovery-assisted method is relatively easy to implement in an existing DG code and avoids a prohibitive increase in computational cost; these traits, combined with the method's superior resolution properties, make it a valuable new tool for CFD analysis.

CHAPTER 1

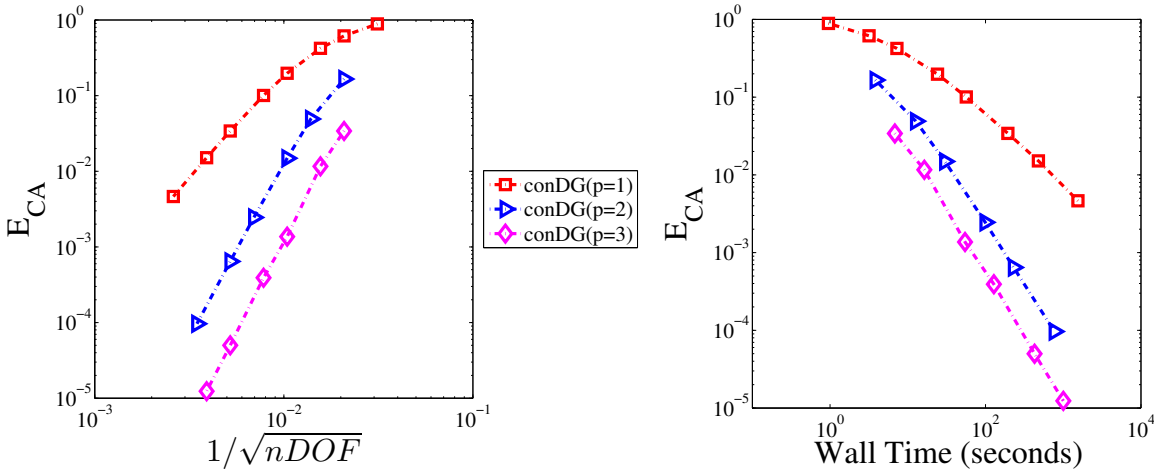
Introduction

The engineering design process frequently mandates understanding and prediction of both low and high speed flows. While there are accepted sets of partial differential equations (PDEs) for prediction of flow features around and within engineered devices, these equations are typically too complicated (specifically, coupled and nonlinear) to be solved by hand, and engineers have historically relied on two methods for predicting flow behavior. The more convenient method is to work with highly simplified versions (low-fidelity models) of the governing PDEs, allowing decent prediction of flow patterns either by hand or with inexpensive computer programs. Examples include potential flow analysis and panel methods for lift calculations. On the other hand, it is typical to design a scale model of the device (such as a submarine), place it in a wind/water tunnel, and observe the flow. This approach is costly not only because the scale model must be constructed, but also because high-quality flow measurement devices are expensive. Moving past the usage of inexpensive models and vehicle-specific experiments, engineers are also aided by the collective knowledge of the general fluid dynamics research community. However, to build a sufficient understanding of fundamental flow physics processes (such as cavitation erosion or fluid mixing), fluid dynamicists are also faced with a difficult choice: either struggle with the analysis of complicated PDEs or conduct physical experiments to gain insight.

Nowadays, analysis and physical experimentation are routinely complemented by approximate solutions of the governing PDEs, computed via specialized numerical algorithms, thereby streamlining the research and design processes. This approach is known as Computational Fluid Dynam-

ics (CFD). It is a constantly evolving field of study because present methods are far from being adequate to accurately predict flow physics around all possible vehicle designs in all flow regimes; see van Leer & Powell [103] for a brief history of the field. In the academic research community, where the priority is to better understand fundamental fluid dynamics processes rather than device-specific flow patterns, CFD practitioners frequently rely on the general class of *high-order methods* to facilitate efficient usage of computational resources when pursuing high-fidelity computational results. Generally, an approximate solution obtained via CFD contains some amount of error; if the error in the solution is judged to be too large, the computational setup is adjusted (via mesh refinement) to devote more computational resources to the simulation. If the CFD method works as designed, the increased use of resources yields a decrease in the simulation error (in technical terms, this property is known as *consistency*). However, different methods achieve different amounts of error reduction. The general idea of high-order methods is to maximize the amount of error reduction as more computational resources are devoted to the simulation. High-order methods have yet to be wholheartedly embraced by industry (where robustness is prioritized over computational efficiency), but they are an indispensable component of the academic research process.

We provide a concrete example to show the value of high-order methods. The transport of an isentropic vortex in uniform inviscid flow (test problem C1.6 in [111]) is simulated through one translational period using a conventional DG spatial discretization (to be fully described in Chapter 2) in a variety of configurations. Three particular versions are included in this study: conDG($p = 1$), conDG($p = 2$), and conDG($p = 3$). Each discretization is applied on a variety of mesh resolutions. The conDG method includes a special parameter p; when this parameter is increased, the method's order of accuracy is increased. Figure 1.1 plots the error of each simulation versus the computational cost. In Figure 1.1a, computational cost is quantified in terms of the number of "degrees of freedom", which are directly correlated (but not linearly correlated) with the memory usage of the code. In Figure 1.1b, cost is instead quantified by computational wall time. Each point on each plot corresponds to a single simulation. In the case of a relatively low-



(a) Error vs. *DOF* count.

(b) Error vs. wall time.

Figure 1.1: Sample refinement study. On the left, the cell-average error E_{CA} in the velocity is plotted against the inverse of the square root of the degree-of-freedom count, $nDOF$. On the right, the error is plotted against computational wall time. The highest order scheme, conDG($p = 3$), provides the lowest error for a fixed amount of computational resources, making it the best scheme in this test.

order method (the $p = 1$ case), the decrease in error is relatively shallow as the computational cost increases. As the order of the method improves, the slope of the error-vs.-cost curve becomes steeper, indicating more substantial decreases in error relative to the computational cost. The slope of the error-vs.-cost curve on the log-log plot for a particular method is the order of convergence (also commonly referred to as the order of accuracy) in the particular error norm employed. The ideal method would be one which produces an error-free simulation with zero wall time, but that is impossible, so high-order schemes are pursued to minimize error at a given computational cost. In practice, the term "high-order" usually refers to methods whose order of convergence is greater than two. However, there are some who see 2^{nd} order methods as high-order as well, so it is not a well-defined criterion; the exact definition is not important. What is important is that increasing the numerical scheme's order of accuracy is an avenue towards simultaneous minimization of both error and computational cost. Note that in Figure 1.1, for a given computational cost (in terms of either wall time or DOF count), the error in the numerical solution is minimized by maximizing the order of accuracy. This advantage motivates the study of high-order CFD methods. We point out that the lowest-order method in Figure 1.1 (specifically the $p = 1$ configuration), achieves 3^{rd}

order convergence in the particular norm employed; presently, the industry standard is 2^{nd} order methods while the research community tends to show greater interest in high-order methods.

The Navier-Stokes equations are viewed by most as the governing differential equations for fluids and are consequently closely linked with the development of CFD. These equations are part of a general class of equations known as advection-diffusion (or convection-diffusion) systems. Advection refers to the transport of certain quantities; for example, a pollutant in a river is carried downstream by the velocity field of the water. In another example, oceanic currents serve to re-distribute heat on a global scale; in fact, global climate simulation/modelling is one of the primary drivers for the development of sophisticated CFD methods [73]. In contrast, diffusion refers to the spreading of a certain quantity independent of a directional transport mechanism. Returning to the example of a pollutant in a river, the pollutant can be expected to spread from regions of high concentration to regions of low concentration (like dye in a cup of water), and this spread of the pollutant is diffusion. In reality, the advection and diffusion processes tend to be present together: as the pollutant is transported downstream by the water velocity, it can spread from one bank of the river to the other (transverse to the water's velocity vector) by diffusion. In this work, while the ultimate target of the proposed discretizations is the compressible Navier-Stokes equations, the methods proposed are thought of more generally in terms of advection-diffusion systems to aid in analysis and simplify the potential extension past the compressible Navier-Stokes equations.

This thesis is devoted to the study of a particular high-order method, known as the discontinuous Galerkin (DG) method, in the context of the compressible Navier-Stokes equations and other advection-diffusion systems. The DG method was originally proposed [88] as a special variant of the finite element approach and is popular for a variety of reasons, the main one being that is has the capability of arbitrarily high orders of accuracy on nontrivial geometries. Compared to the commonly applied continuous finite element approach, the DG method has the advantage of a block-diagonal global mass matrix; see [108, 1, 61, 105] for discussion of the continuous finite element approach alongside the DG method. However, it is our opinion that the conventional DG discretization for advection-diffusion problems does not make the best use of the information avail-

able in the computational stencil, and there is ample room for method improvement (with regard to accuracy versus cost) by thoughtful manipulation of the base discretization. The overall goal of this thesis is to provide a more accurate version of the conventional DG discretization while avoiding a prohibitive increase in computational cost. We now discuss the motivation for working with numerical methods and the DG method in particular; then, Chapter 1 will be closed with an outline of the rest of this document.

1.1 Turbulence: Our Motivation for Method Development

Considering the difficulty involved in the development and testing of numerical methods for PDEs, it is quite reasonable to ask why anyone should be working on numerical methods at all. The answer to this question varies across disciplines; our immediate motivation for working on numerical methods is the general field of fluid mechanics. One of the fundamental issues that make fluids difficult to predict is the presence of turbulence. For the layperson, turbulence is best described as chaotic, unsteady fluid motion on a variety of length scales that are smaller than the characteristic length of the overall flow field. For example, in the case of a hurricane, the characteristic length scale is the hurricane diameter (a few hundred miles). At a much smaller length scale, a human observer on the beach where the hurricane makes landfall will experience unsteady gusts in the air velocity; while the sustained wind speed may be around 70 mph, the wind could momentarily gust to higher speeds or lull to lower speeds. These unsteady, seemingly chaotic wind velocity fluctuations occur over a length scale much smaller than the hurricane diameter, thus the overall flowfield is turbulent.

Empirical evidence demonstrates that the presence of turbulence, despite the fact that it is characterized by velocity fluctuations over relatively small length scales, affects the behavior of the large-scale mean flow. An outstanding example of this phenomena (typically presented to undergraduate engineering students) is pipe flow. For a horizontal, constant-area pipe, fluid must be pushed through the pipe by a pressure difference (denoted Δp) across the full length of the pipe. The necessary pressure difference overcomes the viscous drag exerted by the pipe wall on the

moving fluid. The drag itself is known to scale with the empirical Darcy friction factor [87]. This friction factor itelf depends on the Reynolds number, $Re = \frac{VD}{\nu}$; V is the mean fluid velocity in the pipe, D is the pipe diameter, and ν is the dynamic viscosity. Larger Reynolds numbers correspond to larger levels of turbulence in the flow field. The fact that the friction factor depends on the Reynolds number indicates that the level of drag depends on the level of turbulence inside the pipe, despite the fact that turbulent flow structures tend to be smaller than the diameter of the pipe. In the case of pipe flow, numerous physical experiments have made it possible for engineers to predict pipe drag (and the necessary pressure difference Δp) with sufficient accuracy for the design process across a broad range of Reynolds numbers. The quantitative influence of turbulence on other physical systems, such as a large airplane descending to land or the combustion of fuel-air mixture in a scramjet, is more difficult to predict.

The presence of turbulence can be explained by analysis of the Navier-Stokes equations. Inspection of the incompressible version of the Navier-Stokes equations in wavenumber space [85] indicates that when two flow features of distinct length scales interact, they produce a third flow feature of a different length scale. Thus, in a turbulent flowfield, coherent structures of different length scales are constantly being created, evolving, and getting dissipated away. As the Reynolds number increases (meaning the fluid momentum becomes an increasingly dominant player against the fluid's dissipation properties), the fluid structures exist across a greater span of length scales. For an example of this phenomena, see Figure 1.2, taken from [77]. As the Reynolds number of the pipe flow increases, the coherent flow structures become smaller and smaller. To capture the entire system by numerical simulation (and be sure that full-scale system properties such as viscous drag and mixing rates are being properly predicted), the spatial domain must encompass the entire pipe diameter while also being able to properly resolve the formation and evolution of flow structures that are orders of magnitude smaller than the pipe diameter. This issue leads to astronomically high DOF counts ($nDOF \geq 100,000,000$ is not rare) in high-fidelity numerical simulations of turbulent flows. Thus, it is necessary to work towards numerical methods that minimize the necessary DOF count for a given level of accuracy. High-order CFD methods are naturally suited for this task.

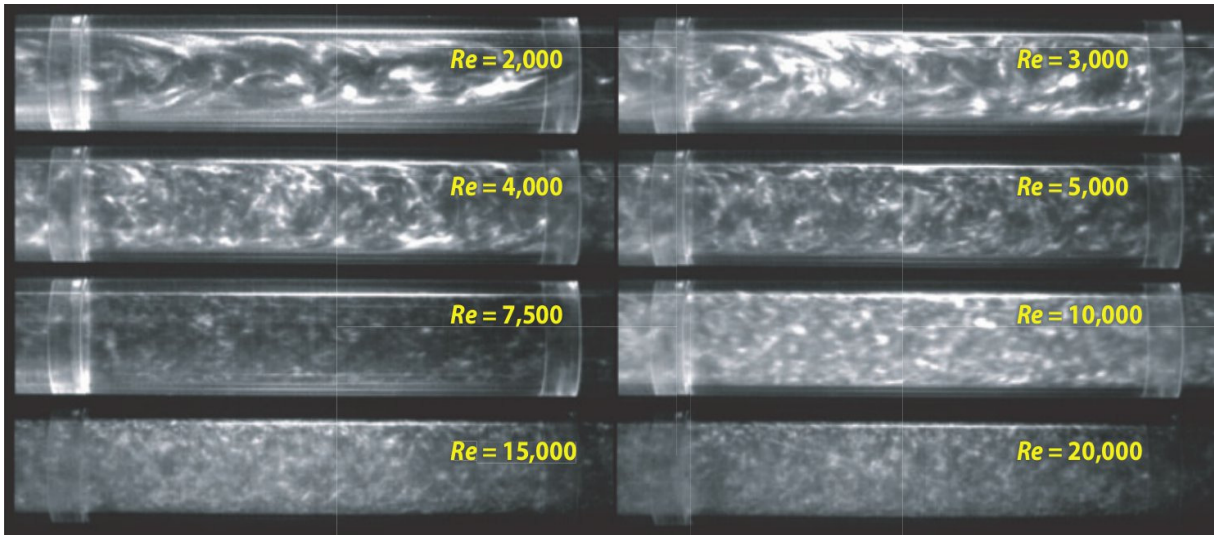


Figure 1.2: Illustration of pipe flow, taken with permission from Mullin [77]. As the pipe Reynolds number is increased, turbulent flow structures appear at smaller length scales.

1.2 The DG Method and Turbulence

Computational studies of turbulent flows on simple geometries ("turbulence in a box") are common in the academic research community as a means to better understand turbulent flow dynamics across various regimes (compressible vs. incompressible, singlefluid vs. multifluid, etc.). We are particularly interested in simulations of the compressible case, where the fluid's ability to expand and compress influences the exchange of energy across different length scales [109]. On the more applied side, the engineering design process frequently mandates consideration of turbulent flow dynamics on nontrivial geometries such as aircraft wings, turbine blades, and vortex generators that operate in the compressible regime. A particulary fascinating intersection of turbulent flow dyanamics and compressible phenomena is turbulent shock-wave/boundary-layer interaction [25, 78], which can hinder aircraft performance by facilitating oscillatory flow features on lifting surfaces and within high-speed engine inlets.

In the industrial setting, the Reynolds-averaged Navier-Stokes (RANS) approach tends to be favored for moderate solution quality at tolerable computational cost, as it models the effects of small-scale turbulent flow features without requiring the spatial discretization to properly resolve

these features. However, thorough understanding of the underlying turbulent interactions requires physical experimentation alongside high-fidelity computational results obtained via unsteady large eddy simulation (LES) and/or direct numerical simulation (DNS) of the compressible Navier-Stokes equations. In the case of LES, some of the turbulent length scales are properly resolved by the grid while the smallest scales are modelled. In the case of DNS, all turbulent length scales are properly resolved in order to provide the most authoritative description of the flow. Unfortunately, as the Reynolds number of the flow is raised, the necessary gridpoint count grows with it. Thus, high-fidelity simulations are restricted by computational cost to relatively low Reynolds numbers, and this limitation hinders our understanding of turbulent flows. In the interest of benefitting the general fluid dynamics community, it is worthwhile to continue developing and building improved CFD codes for high-fidelity compressible turbulence simulations. Such codes should ideally meet the following requirements:

- Efficient scaling on large distributed-memory architectures, to ensure efficient resource utilization in massively parallel simulations.
- High-order accuracy, allowing a more accurate simulation than a low-order scheme at a given gridpoint count.
- Applicability to unstructured meshes, preferably allowing high-order accuracy on nontrivial geometries.

The DG method meets these requirements, making it a promising candidate for the job.

However, the DG method is not the only candidate for the discretization of these types of problems. In fact, limited evidence indicates that the conventional DG discretization offers little to no advantage over high-order finite volume schemes for compressible turbulence simulations [72]. One would expect that this shortcoming could be overcome by the DG method's theoretical capability for an arbitrarily high order of accuracy: for a sufficiently smooth problem and a given level of error, the DOF count necessary to achieve that error is expected to decrease as the solution polynomial order p is raised and the mesh is coarsened [111, 110], as demonstrated in

Figure 1.1. Unfortunately, this advantage is diminished by practical considerations. First, a high-p discretization is effective only if the coarsened mesh is of high quality, but the generation of high-quality meshes for high-p discretizations has proven to be a difficult task and remains an active research area [41, 111]. Additionally, for highly compressible flows [72] (where shocklets may be present), evidence shows that h-refinement, while keeping p relatively small, is more efficient than p-refinement for minimizing error at a given DOF count. The work of Chapelier et al. [21] in the incompressible regime suggests that h-refinement is more effective than p-refinement for minimizing error in simulations of wall-bounded viscous flows at a fixed DOF count, at least for the p values typically chosen in the DG method. For nonlinear PDE problems, the reduced numerical dissipation of high-p discretizations tends to worsen the aliasing issue, necessitating stabilization through a variety of methods [11, 36, 114] that tend to complicate the spatial discretization and/or significantly raise the computational cost. There is also the widely studied issue of suppressing Gibbs oscillations in regions of steep gradients [33, 118, 23, 6, 81], which tends to become more inconvenient as the solution order is raised. Furthermore, in the case of unsteady flows, the order of the temporal discretization scheme is rarely greater than 4^{th} order. Due to the relatively high truncation error in time, a CFD code with better than 4^{th} order accuracy in space acts with reduced effectiveness. With these issues in mind, while exploiting the extraordinary performance gains of p-refinement where possible, it is also worthwhile to develop alternative DG discretizations that reduce the error at a given p compared to the conventional DG discretization (as opposed to relying exclusively on p-refinement) while maintaining a compact computational stencil and avoiding a severe increase in computational cost.

1.3 Goal, Document Outline, and List of Schemes

The goal of this thesis is to improve upon the conventional DG discretization for advection-diffusion problems with regard to accuracy versus computational cost at a given polynomial order p. The proposed DG variants are designated *Recovery-assisted* DG schemes owing to their utilization of a relatively new idea known as Recovery [102] and the associated recovery opera-

tor. Our motivation for including the recovery operator is simple: the solution representation in the DG method is multivalued at certain locations (known as interfaces), and as will be shown in Chapter 2, solution quantities must be approximated at the interfaces. The recovery operator intelligently interprets the discontinuous DG approximation near an interface to build an exceedingly accurate estimate for the solution along the interface. We hypothesized that by careful application of the recovery operator, the truncation error of the full DG spatial discretization at a given p could be substantially reduced for both advection and advection-diffusion problems. The core of this document methodically tells the story of the methods built to test this hypothesis.

Before moving on, we address the caveat of unstructured versus structured meshes. The majority of the analysis and testing in this document is aimed at high-fidelity simulations on uniform structured meshes, where one could certainly employ the global spectral/pseudospectral method (within parallel efficiency constraints) or a high-order finite difference scheme to beat the performance of the DG method in terms of accuracy versus simulation cost. However, for maximum versatility, a CFD code should be able to handle structured and unstructured meshes. Towards this end, we focus on improving the DG method's accuracy on structured meshes while maintaining its applicability to unstructured meshes. The Recovery-assisted DG schemes are built to achieve this goal (as will be made apparent in this document), but their robustness on unstructured meshes has not been explored in detail and remains an open issue.

The remaining layout is as follows. In Chapter 2, the DG spatial discretization is summarized and a pair of simple test problems is presented. The tests indicate that while the DG method performs well for the linear advection equation, a naive implementation fails to properly discretize the linear diffusion equation. A discussion on Fourier analysis (sometimes called von Neumann wavenumber analysis) is presented. Then, the Recovery concept [102] is introduced; the original implementation of the recovery operator is described and an alternative derivative-based implementation is proposed.

The Recovery-based DG schemes of Lo [67] (descended from the original Recovery DG method of van Leer & Nomura [102]) are built specifically to maximize accuracy on scalar dif-

fusion problems; they represent the most direct application of the Recovery concept in the DG framework. They achieve impressive convergence rates for parabolic PDE problems. However, the Recovery-based DG approach requires a non-compact stencil to achieve stable and consistent treatment of shear diffusion terms, which appear in the compressible Navier-Stokes equations; additionally, implementation in an existing DG code is a painfully invasive procedure. In response, Chapter 3 presents our Interface Gradient Recovery family of schemes for parabolic PDEs. We combine the Recovery concept with the commonly applied mixed formulation to build a family of schemes that compare favorably to common DG methods for parabolic PDEs (with respect to accuracy) but are less intrusive to implement than the Recovery-based DG schemes of Lo [67].

Moving away from parabolic PDEs, Chapter 4 reviews the Interface-Centered-Binary (ICB) reconstruction schemes of Khieu & Johnsen [56] for advection problems (hyperbolic PDEs). The ICB schemes employ a biased version of the recovery operator to facilitate stable discretization of the linear advection equation. We analyze and test a trio of schemes within the ICB reconstruction framework; these schemes tend to achieve an advantage in accuracy compared to the conventional upwind DG discretization for advection.

Our favorite advection and diffusion schemes from Chapter 3 and Chapter 4 are combined in Chapter 5 to form the family of Recovery-assisted DG schemes for advection-diffusion problems, such as the compressible Navier-Stokes equations. These Recovery-assisted DG schemes are the centerpiece of the thesis. We employ a recently proposed form of Fourier analysis to predict scheme performance over a broad range of Peclet numbers, showing that the new advection-diffusion schemes compare favorably to a conventional DG approach in terms of accuracy. The new schemes are evaluated with a set of test problems. These demonstrations are necessary in order to verify that the Recovery-assisted DG schemes' advantages, predicted in Fourier analysis, carry over to the nonlinear compressible Navier-Stokes equations.

Overall, Chapter 3, Chapter 4, and Chapter 5 form the core of our thesis work. Chapter 6 and Chapter 7 present exciting miscellaneous topics. Chapter 6 examines the Recovery-based DG methods of van Leer, Nomura, and Lo [102, 67, 69, 57, 101, 68] for parabolic PDEs. A pair of

boundary schemes is proposed for the application of Dirichlet and Neumann boundary conditions; a comparison of different boundary schemes is carried out through a simple test problem. Leaving the DG framework behind, Chapter 7 shows how the Recovery concept can be applied in the relatively new Flux Reconstruction (FR) framework [46, 47] to obtain an attractive FR scheme for parabolic PDEs. Additionally, the Fourier analysis technique is extended to irregular mesh layouts, as one would expect to encounter on the unstructured meshes that are typical in design-related CFD analysis scenarios. Chapter 8 closes the thesis with some concluding remarks and ideas for future research.

To orient the reader, Figure 1.3 lists our proposed schemes next to established approaches for advection problems, diffusion problems, and advection-diffusion problems. For advection problems, the ICB approaches upgrade the upwind DG approach for advection with a biased version of the recovery operator. On the other end of the diagram, the Interface Gradient Recovery family represents a compromise between the versatilty of the mixed formulation and the accuracy of Lo's [67] Recovery-based DG schemes. A conventional DG discretization for advection-diffusion problems is obtained by combining the upwind DG discretization for advection with the BR2 scheme [8] (or a similar scheme such as interior penalty [3, 40]) for diffusion. The Recovery-assisted DG schemes for advection-diffusion are formed by combining the ICB schemes (for the advective fluxes) with a compact member of the Interface Gradient Recovery family (for the diffusive fluxes).

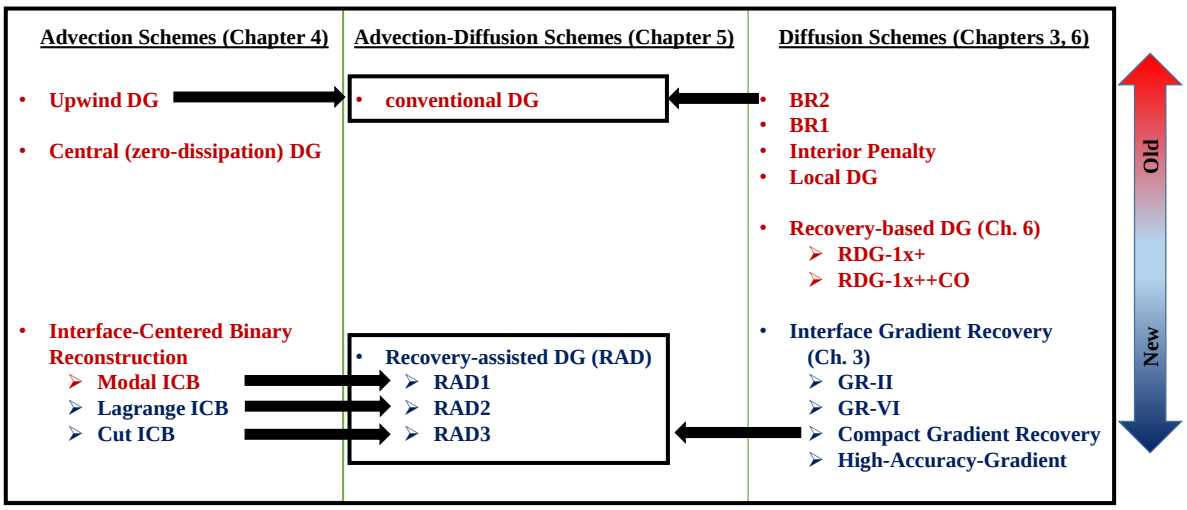


Figure 1.3: Summary of proposed schemes. Novel schemes are listed in blue text while previously existing approaches are listed in red text. A DG scheme for advection-diffusion problems is formed by combining an advection scheme (from the left side of the diagram) with a diffusion scheme (from the right side of the diagram).

CHAPTER 2

Fundamental Concepts

2.1 Chapter Overview

This chapter has four purposes. The first is to introduce the governing partial differential equations (PDEs) that make an appearance in this work. The second purpose is to introduce the discontinuous Galerkin (DG) method. We will demonstrate the method's excellent performance for the linear advection equation in 1D, then show that a naive application of the method fails to properly discretize the 1D heat equation. Third, the Fourier analysis technique, which has been crucial to the analysis of the DG variants proposed in this work, is reviewed. Fourth, the recovery operator [102] is described. The recovery operator is a tool to be applied within the DG method. It is described in detail to allow for a lucid explanation of DG variants in later chapters. Additionally, the Recovery section of this chapter introduces the novel derivative-based implementation of the recovery operator. Those already familiar with the compressible Navier-Stokes equations and the DG spatial discretization need only read Section 2.4 and Section 2.6.

2.1.1 Novelty and Articles

Each chapter begins with a statement on novelty to summarize our contributions and list relevant conference proceedings and/or journal articles. This chapter contains two novel concepts. The first novelty is the Lagrange ICB reconstruction technique in Section 2.6. The Lagrange ICB reconstruction is a modification of Khieu & Johnsen's [56] icbp[0] technique. The second novelty

is the derivative-based implementation of the Recovery operator, which was inspired by the bilinear form of the RDG-2x scheme [104].

The derivative-based recovery implementation is proposed in the following journal article, which is presently in preparation:

- P. E. Johnson, L. H. Khieu, & E. Johnsen, *A Compact Recovery-Assisted Discontinuous Galerkin Method for the Compressible Navier-Stokes Equations*, in preparation.

2.1.2 Usage of Recovery

In addition to a statement on novelty, each chapter begins with a statement on how the Recovery concept (to be explained in Section 2.6) is utilized. It will be explicitly stated whether the recovery operations are applied in the derivative-based or inner-product based implementations.

2.2 Governing Equations

This section describes all of the partial differential equations (PDEs) that make an appearance in this work. The state variable is denoted U and is a function of space $\boldsymbol{x}$ and time t such that $U = U(\boldsymbol{x}, t)$. For our purposes, the PDEs are always cast in the following form:

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{Q}\left(U, \nabla U\right) = \mathcal{S}(\boldsymbol{x}, t), \tag{2.1}$$

where $\boldsymbol{Q}$ is referred to as the *flux*. In general, it depends on both the solution U and its gradient ∇U. The boldface notation has been chosen to indicate that the flux $\boldsymbol{Q}$ is typically a vector, even when the state variable U is a scalar. Certain test cases also include a nonzero source term $\mathcal{S}$, which is independent of U for all cases considered in this work.

In this section, the letter p makes an appearance as the variable for pressure of an ideal gas. This p is completely unrelated to the polynomial order p that will appear later in the description of the discontinuous Galerkin method.

2.2.1 1D Diffusion Equation

The 1D diffusion equation features a single variable U constrained as follows:

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}\left(\mu \frac{\partial U}{\partial x}\right), \tag{2.2}$$

where μ is the diffusivity. Making use of a diffusive flux $\mathcal{G}$, an equivalent statement is

$$\frac{\partial U}{\partial t} - \frac{\partial \mathcal{G}}{\partial x} = 0 \quad \text{where} \quad \mathcal{G} = \mu \frac{\partial U}{\partial x}. \tag{2.3}$$

If μ is constant, then in the absence of a source (forcing) term, $U(x,t)$ tends towards a straight line as t approaches infinity, effectively smoothing away any oscillations. The equation is more difficult to solve when μ is not constant. The 1D diffusion equation is parabolic and serves as a highly simplified model of the diffusive terms in the compressible Navier-Stokes equations.

2.2.2 1D Linear Advection Equation

The 1D advection equation features a single variable U constrained as follows:

$$\frac{\partial U}{\partial t} + \frac{\partial \mathcal{F}}{\partial x} = 0 \quad \text{where} \quad \mathcal{F} = aU, \tag{2.4}$$

with a being the advection speed. In this work, a is kept constant, so we are working with the 1D *linear* advection equation. In the case of periodic boundary conditions, the solution of this equation is $U(x,t) = U(x-at,0)$ where $U(x,0)$ is the initial condition. The 1D advection equation is hyperbolic and serves as a highly simplified model of the inviscid terms in the compressible Navier-Stokes equations.

2.2.3 1D Linear Advection-Diffusion Equation

Combining the linear advection equation and the 1D diffusion equation yields the linear advection-diffusion equation:

$$\frac{\partial U}{\partial t} + \frac{\partial(\mathcal{F} - \mathcal{G})}{\partial x} = 0 \quad \text{where} \quad \mathcal{F} = aU \quad \text{and} \quad \mathcal{G} = \mu \frac{\partial U}{\partial x}. \tag{2.5}$$

This equation is a highly simplified model of the compressible Navier-Stokes equations. It is the prototypical advection-diffusion system, exhibitting mixed hyperbolic/parabolic behavior due to the transport mechanism of the advection equation and the smoothing behavior of the diffusion equation.

2.2.4 Multidimensional Linear Advection Equation

The general multidimensional linear advection equation features a single variable U constrained as follows, where N_D is the number of spatial dimensions:

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0 \quad \text{where} \quad \boldsymbol{\mathcal{F}} = \left\{ \begin{array}{ll} v_1 U & \text{for } N_D = 1 \\ (v_1 U\ ,\ v_2 U) & \text{for } N_D = 2 \\ (v_1 U\ ,\ v_2 U\ ,\ v_3 U) & \text{for } N_D = 3 \end{array} \right\}. \tag{2.6}$$

In this case, $\boldsymbol{V} = (v_1, v_2, v_3)$ the the velocity vector; it governs the transport of the scalar variable U. As with the 1D version, this equation is hyperbolic.

2.2.5 Multidimensional Diffusion Equation

Typically, studies of diffusion in 2D and 3D focus on the scalar Laplacian equation. To model the difficulties of capturing the viscous stress tensor in the Navier-Stokes equations, we allow for a shear term in the diffusion equation. Again, N_D is the number of spatial dimensions and the single

variable $U = U(\boldsymbol{x}, t)$ is constrained according to a generalized Fourier law:

$$\frac{\partial U}{\partial t} - \nabla \cdot \boldsymbol{\mathcal{G}} = \mathcal{S} \quad \text{where} \quad \boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{D}} \nabla U)^T , \tag{2.7}$$

with $\boldsymbol{\mathcal{D}}$ being the diffusivity matrix. The diffusivity matrix scales with the diffusivity μ and is constructed as follows:

$$\boldsymbol{\mathcal{D}} = \left\{ \begin{array}{ll} \mu, & \text{for } N_D = 1 \\ \begin{bmatrix} \mu & \mu\theta \\ \mu\theta & \mu \end{bmatrix}, & \text{for } N_D = 2 \\ \begin{bmatrix} \mu & \mu\theta & \mu\theta \\ \mu\theta & \mu & \mu\theta \\ \mu\theta & \mu\theta & \mu \end{bmatrix}, & \text{for } N_D = 3 \end{array} \right\}, \tag{2.8}$$

where θ is known as the shear factor in this work. When $\theta = 0$, this equation is the scalar Laplacian. Many of the schemes explored in this work exhibit different stability properties depending on the value of θ.

2.2.6 The Euler Equations

The Euler equations govern ideal gas flow in the limit of vanishing viscosity. The conservative form of the equation is employed here. The conserved variables are stored in a vector U that contains the density, momentum, and energy of the gas. In the 1D case,

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0 \quad \text{where} \quad U = \begin{bmatrix} \rho \\ \rho v_1 \\ E_g \end{bmatrix} \quad \text{and} \quad \boldsymbol{\mathcal{F}} = \begin{bmatrix} \rho v_1 \\ \rho v_1^2 + p \\ v_1 \left(E_g + p \right) \end{bmatrix}. \tag{2.9}$$

In this equation, ρ is the density (mass per volume) of the fluid, v_1 is its velocity in the x direction (such that ρv_1 is the momentum), and E_g is the energy density (energy per volume), typically just referred to as the energy. The pressure p is linked to the conserved variables via an equation of

state. The letter p here has nothing to do with the polynomial order in the DG spatial discretization. The equations can be written in a single, compact form for the general multidimensional case, but in the interest of clarity, the 2D and 3D versions are reported separately in this section. In the 2D case,

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0 \text{ where } U = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ E_g \end{bmatrix}, \boldsymbol{\mathcal{F}} = \left\{ \begin{bmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ v_1 \left(E_g + p \right) \end{bmatrix}, \begin{bmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ v_2 \left(E_g + p \right) \end{bmatrix} \right\}, \tag{2.10}$$

with v_2 being the velocity component in the y direction and ρv_2 being the corresponding momentum term. In the 3D case,

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0 \text{ where } U = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ E_g \end{bmatrix}, \boldsymbol{\mathcal{F}} = \left\{ \begin{bmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ v_1 \left(E_g + p \right) \end{bmatrix}, \begin{bmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ \rho v_2 v_3 \\ v_2 \left(E_g + p \right) \end{bmatrix}, \begin{bmatrix} \rho v_3 \\ \rho v_1 v_3 \\ \rho v_2 v_3 \\ \rho v_3^2 + p \\ v_3 \left(E_g + p \right) \end{bmatrix} \right\}, \tag{2.11}$$

with v_3 being the velocity component in the z direction and ρv_3 being the corresponding momentum term. The equation of state for the pressure p is as follows:

$$p = \left\{ \begin{array}{ll} (\gamma - 1) \left(E_g - \frac{\rho}{2} v_1^2 \right) & \text{for } N_D = 1 \\ (\gamma - 1) \left(E_g - \frac{\rho}{2} \left(v_1^2 + v_2^2 \right) \right) & \text{for } N_D = 2 \\ (\gamma - 1) \left(E_g - \frac{\rho}{2} \left(v_1^2 + v_2^2 + v_3^2 \right) \right) & \text{for } N_D = 3 \end{array} \right\}, \tag{2.12}$$

which can be written in compact form as $p = (\gamma - 1) \left(E_g - \frac{\rho}{2} v_i v_i \right)$ where the repeated index indicates summation. The parameter γ is the ratio of specific heats; for all work presented in this document, $\gamma = 1.4$.

2.2.7 The Compressible Navier-Stokes Equations

The compressible Navier-Stokes equations are obtained by adding physical diffusive flux terms (sometimes called the viscous flux terms) to the Euler equations. The vector of conserved variables, U, is the same as that used for the Euler equations. In addition to the pressure p, these equations involve the fluid temperature T which is governed by the ideal gas law, $p = \rho R_g T$, where R_g is the gas constant. Temperature diffuses according to Fourier's law with thermal conductivity $\Lambda = \frac{\mu C_p}{Pr}$. The parameters C_p and Pr are the specific heat (at constant pressure) and the Prandtl number, respectively. The specific heat is related to the gas constant (R_g) and the ratio of specific heats (γ) as $C_p = \frac{\gamma R_g}{\gamma - 1}$. The conserved variables are governed as follows:

$$\frac{\partial U}{\partial t} + \nabla \cdot (\mathcal{F} - \mathcal{G}) = 0, \tag{2.13}$$

where the inviscid flux $\mathcal{F}$ is taken from the Euler equations. It remains to define the diffusive flux, $\mathcal{G}$. In compact form, the diffusive flux takes the following form:

$$\mathcal{G} = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \boldsymbol{V} + \Lambda \left(\nabla T\right)^T \end{bmatrix} \quad \text{where} \quad \tau_{ij} = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right). \tag{2.14}$$

The term $\boldsymbol{\tau}$ is the viscous stress tensor; it bears some resemblance to the flux for the multidimensional diffusion equation (Eq. 2.7). In the definition of $\boldsymbol{\tau}$, we have taken the usual approach of assuming the bulk viscosity to be zero (i.e., Stokes' assumption); see Buresti [18] for a concise, modern discussion of this assumption. The term $(\nabla T)^T$ is the transpose of the temperature gradient (not the temperature gradient raised to a power T). The diffusive flux in the 2D case is explicitly

stated here for the reader's convenience:

$$\mathcal{G} = [\mathcal{G}^x \quad , \quad \mathcal{G}^y] \quad \text{where}$$

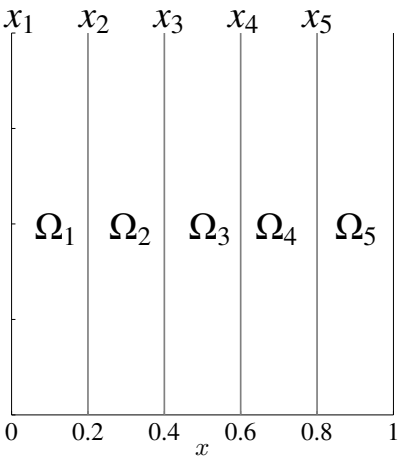
$$\mathcal{G}^x = \begin{bmatrix} 0 \\ \frac{2\mu}{3}\left(2\frac{\partial}{\partial x}v_1 - \frac{\partial}{\partial y}v_2\right) \\ \mu\left(\frac{\partial}{\partial y}v_1 + \frac{\partial}{\partial x}v_2\right) \\ v_1\left(\frac{2\mu}{3}\left(2\frac{\partial}{\partial x}v_1 - \frac{\partial}{\partial y}v_2\right)\right) + v_2\left(\mu\left(\frac{\partial}{\partial y}v_1 + \frac{\partial}{\partial x}v_2\right)\right) + \Lambda\frac{\partial T}{\partial x} \end{bmatrix}, \tag{2.15}$$

$$\mathcal{G}^y = \begin{bmatrix} 0 \\ \mu\left(\frac{\partial}{\partial y}v_1 + \frac{\partial}{\partial x}v_2\right) \\ \frac{2\mu}{3}\left(2\frac{\partial}{\partial y}v_2 - \frac{\partial}{\partial x}v_1\right) \\ v_2\left(\frac{2\mu}{3}\left(2\frac{\partial}{\partial y}v_2 - \frac{\partial}{\partial x}v_1\right)\right) + v_1\left(\mu\left(\frac{\partial}{\partial y}v_1 + \frac{\partial}{\partial x}v_2\right)\right) + \Lambda\frac{\partial T}{\partial y} \end{bmatrix}.$$

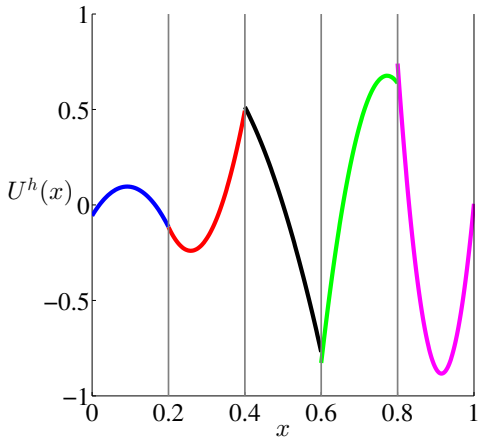
The compressible Navier-Stokes equations are coupled and nonlinear, which makes them difficult to solve by hand; a general closed-form analytical solution has thus far evaded the human race.

2.3 The Discontinuous Galerkin (DG) Method in 1D

This section gives an explanation of the discontinuous Galerkin (DG) method in the context of the 1D linear advection equation (Eq. 2.4). The method was originally proposed by Reed & Hill [88] as a special version of the finite element approach; the 1D version for the linear advection equation can also be found in van Leer's high-order generalizations [100] of the original Godunov scheme. The semi-discrete approach is taken, meaning that the DG method functions as a tool with which to transform a partial differential equation (PDE) into a system of ordinary differential equations (ODEs) in time, which are then solved by the user's method of choice. Assume spatially periodic boundary conditions and take the solution to be set to some initial condition U_{IC} at $t = 0$: $U(x, 0) = U_{IC}(x)$. The solution U is to be approximated by $U^h(x, t)$, which is defined as a piece-



(a) The neighboring elements; each contains $p + 1$ DOFs.



(b) The discontinuous polynomial $U^h(x)$ with $p = 2$.

Figure 2.1: Domain decomposition demonstration. The domain $\mathbf{\Omega} = \{x \mid x \in [0, 1]\}$ is partitioned into five equally spaced, distinct elements with $h = \frac{1}{5}$. Vertical gray lines indicate element interfaces. In the right figure, the discontinuous polynomial U^h approximates the function $U(x) = x\sin(6\pi x)$, and each element's polynomial is granted a distinct color.

wise continuous polynomial; the technique for forming this polynomial requires the use of domain subdivisions known as *elements.*

Given the 1D spatial domain, $\mathbf{\Omega} = \{x \mid x \in [0, L]\}$, partition it into M non-overlapping subintervals, known as elements and labelled Ω_m, such that $\mathbf{\Omega} = \cup_{m=1}^{M} \overline{\Omega}_m$ as shown in Figure 2.1a. Each element Ω_m has a left boundary and a right boundary. For example, the left boundary of Ω_2 in Figure 2.1a sits at $x_2 = 0.2$ and the right boundary sits at $x_3 = 0.4$. At both of these boundaries, also known as interfaces, Ω_2 interfaces with other elements in the domain.

Take each element Ω_m to be mapped to a reference element, Ω_{ref}, which exists in the bi-unit interval and is spanned by reference coordinate ξ such that $\Omega_{ref} = \{\xi \mid \xi \in [-1, 1]\}$. Let $\boldsymbol{\phi_{ref}}$ be a degree-p polynomial basis (with $p + 1$ members) in the reference element. For example, if $p = 2$, then $\boldsymbol{\phi_{ref}} = \{1, \xi, \xi^2\}$ is an acceptable choice. Given the physical left endpoint of Ω_m, labelled x_m, and the element width, labelled h_m, the mapping between the reference element Ω_{ref} and the

physical element Ω_m takes the following form:

$$\xi(x) = -1 + \frac{2}{h_m}(x - x_m), \tag{2.16a}$$

$$x(\xi) = x_m + \frac{h_m}{2}(\xi + 1). \tag{2.16b}$$

Let each Ω_m have its own set of basis functions ϕ_m^k which are nonzero on Ω_m and equal to zero outside Ω_m:

$$\begin{aligned} \phi_m^k(x \in \Omega_m) &= \phi_{ref}^k(\xi(x)) \quad \forall k \in \{0, 1, ..., p\}, \\ \phi_m^k(x \notin \Omega_m) &= 0 \quad \forall k \in \{0, 1, ..., p\}, \end{aligned} \tag{2.17}$$

where the subscript m indicates that the basis function is native to Ω_m. The element's collection of basis functions is denoted with boldface: $\boldsymbol{\phi_m} = \{\phi_m^0, \phi_m^1, ..., \phi_m^p\}$. Now, the necessary pieces are in place to desribe the discontinuous polynomial U^h:

$$U^h(x \in \Omega_m, t) = U_m^h(x, t) = \sum_{k=0}^{p} \hat{U}_m^k(t)\phi_m^k(x). \tag{2.18}$$

The global approximation U^h is described at any time t by the collection of coefficients $\hat{\mathbf{U}}(t)$, which are known as the degrees of freedom (DOFs), and the basis functions $\boldsymbol{\phi}$, which are independent of time. An example approximation U^h is given in Figure 2.1b; note that U^h is multi-valued along interfaces, as there are no continuity constraints in the basis functions $\boldsymbol{\phi}$ or the approximation U^h.

In general, it is impossible for U^h to satisfy the governing differential equation (Eq. 2.4 in the present example) for all x and all t, so we attempt to satisfy it in the integral sense instead:

$$\int_{\Omega} \phi_m^k \left(\frac{\partial U}{\partial t} + a\frac{\partial U}{\partial x} \right) dx = 0 \quad \forall m \leq M, \quad \forall k \leq p. \tag{2.19}$$

Given the fact that each ϕ_m^k is nonzero over exactly one Ω_m, the constraints are addressed in an

element-by-element fashion:

$$\int_{\Omega_m} \phi_m^k \left(\frac{\partial U}{\partial t} + a \frac{\partial U}{\partial x} \right) dx = 0 \quad \forall \Omega_m \in \mathbf{\Omega}, \quad \forall k \leq p. \tag{2.20}$$

We substitute the polynomial approximation U^h for U and perform an integration by parts to obtain the DG weak form:

$$\int_{\Omega_m} \phi_m^k \frac{\partial U^h}{\partial t} dx + \left[\phi_m^{k-} \tilde{aU} \right] |_{x_m}^{x_m + h_m} - \int_{\Omega_m} \frac{\partial \phi_m^k}{\partial x} a U^h dx = 0 \quad \forall \Omega_m \in \mathbf{\Omega}, \quad \forall k \leq p. \tag{2.21}$$

The term ϕ_m^{k-} is the limit of ϕ_m^k as the element boundary is approached from inside Ω_m; recall that since ϕ_m^k goes to zero outside the element, it is in fact multivalued at $x = x_m$ and $x = x_m + h_m$. The DG weak form requires that an approximation for the aU term be populated along each interface; this task is not trivial because the DG approximation is, in general, multivalued along interfaces, hence aU^h is also multivalued. The tilde mark above the interface aU term is the notation used in this work to indicate that a common value must be specified for aU along the interface. Both elements sharing this interface must make use of this single aU approximation when populating the DG weak form (Eq. 2.21). In the DG community, the task of calculating the interface flux value is generally handled by a Riemann solver [99, 90]; a more detailed discussion of this task is given in Chapter 4.

The terms in Eq. (2.21) are grouped into a residual term and a mass matrix term. For notational ease, we take the indexing of rows and columns in matrices and vectors to start at zero, i.e. the first row is designated row 0, the second row is designated row 1, etc. Define the residual vector $\boldsymbol{R}_m$ for element Ω_m as

$$R_m^{row} = \left[\phi_m^{row-} \tilde{aU} \right] |_{x_m}^{x_m + h_m} - \int_{\Omega_m} \frac{\partial \phi_m^{row}}{\partial x} a U^h dx \tag{2.22}$$

and the mass matrix $\boldsymbol{\mathcal{M}_m}$ for element Ω_m as

$$\mathcal{M}_m^{row,col} = \int_{\Omega_m} \phi_m^{row} \phi_m^{col} dx. \tag{2.23}$$

Assuming the DOFs $\hat{\mathbf{U}}(t)$ to be known at time t, the terms in the residual vector $\boldsymbol{R_m}$ for each Ω_m are immediately available, assuming one has a rule for determining $a\tilde{U}$ along interfaces. Making use of the mass matrix, Eq. (2.21) is written in matrix-vector form for each Ω_m:

$$\boldsymbol{\mathcal{M}_m}\frac{d}{dt}\begin{bmatrix}\hat{U}_m^0\\ \hat{U}_m^1\\ \vdots\\ \hat{U}_m^p\end{bmatrix} = -\begin{bmatrix}R_m^0\\ R_m^1\\ \vdots\\ R_m^p\end{bmatrix}. \tag{2.24}$$

Eq. (2.24) provides the means to establish the time derivatives of the DOF vector $\hat{\mathbf{U}}_\mathbf{m}$ given the residual vector $\boldsymbol{R_m}$. Now, instead of a partial differential equation (Eq. 2.4), the DG spatial discretization has been used to formulate a system of ODEs for the DOFs $\hat{\mathbf{U}}$.

The discretization scheme for the governing PDE is not complete until a method is chosen to solve the system of ODEs for the DOFs $\hat{\mathbf{U}}$. As an example, the forward Euler method is demonstrated here:

$$\frac{\hat{\mathbf{U}}_m(t+\Delta t) - \hat{\mathbf{U}}_m(t)}{\Delta t} = -\boldsymbol{\mathcal{M}}_\boldsymbol{m}^{-1}\boldsymbol{R_m}(\hat{\mathbf{U}}(t)), \tag{2.25}$$

where Δt is known as the timestep size. We emphasize that the complete residual vector $\boldsymbol{R}$ depends on the known DOF vector $\hat{\mathbf{U}}(t)$. The practice of evolving the system from time t to time $t + \Delta t$ is known as time integration, and the forward Euler method is the simplest possible scheme.

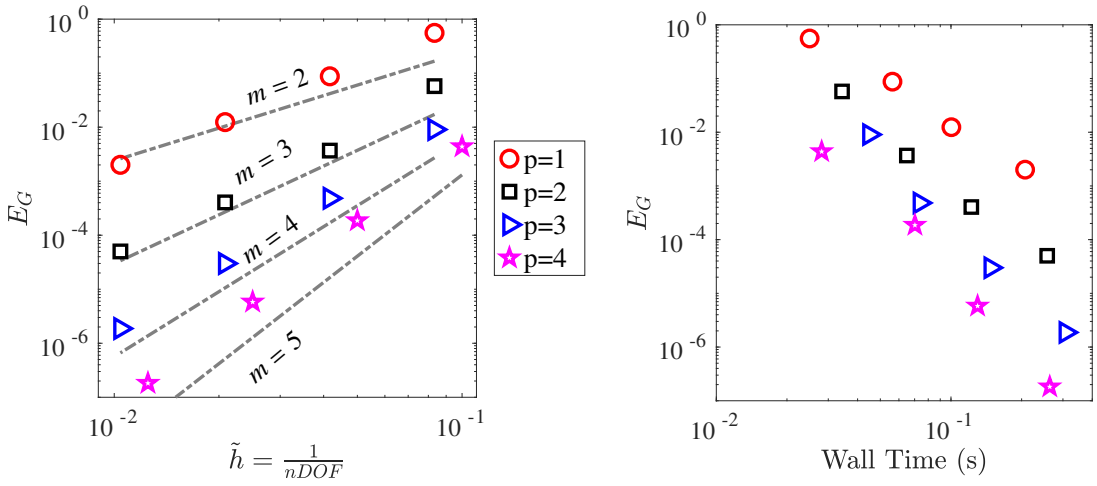
The main reason for the popularity of the DG method is that the order of accuracy scales with the solution order p. In the case of the linear advection equation, the DG spatial discretization approximates the exact translation operator with order $2p + 1$ accuracy. To achieve this performance, the interface term $a\tilde{U}$, known as the interface flux, must be properly approximated. By taking $a\tilde{U}$ from the "upwind" side of the interface (meaning the left element if $a > 0$ and the right element if $a < 0$), order $2p + 1$ accuracy is achieved. The practice of upwinding can be justified either by physical intiuition or mathematical properties. From a physics perspective, if two solution states, U_L and U_R, are present at the same location at a given time, then the expected solution state at

the specific location after some infinitessimal amount of time has passed depends on which way the advection velocity a is pointing. From a numerical methods perspective, the practice of upwinding introduces some numerical dissipation that proves useful for damping away nonphysical oscillations in underresolved simulations; see Brooks & Hughes [16] or Brezzi et al. [14] for elegant explanations of this connection. It is also possible to populate $a\tilde{U}$ by taking the average of U^h from the two sides of the interface; this technique is known as a *central* scheme. The central scheme is generally frowned upon in the DG community. With regard to implementation, the typical approach for populating the integrals that appear in the DG weak form (Eq. 2.21) is Gaussian quadrature, with the number of Gaussian quadrature nodes being chosen appropriately high for the solution order p. It is also possible to solve the integrals analytically in terms of the DOFs $\hat{\mathbf{U}}$, but that approach is cumbersome and memory-intensize for nonlinear flux laws.

A demonstration of the method's performance for the 1D linear advection equation is presented. The spatial domain is $x \in [0, 2\pi]$, the advection speed is $a = \pi$, the initial condition is $U(x, 0) = \sin(x)$, and the equation is simulated from $t = 0$ to $t = 8$ (four translational periods). Time integration is handled by the 8^{th} order explicit Runge-Kutta scheme of Prince & Dormand [86]. In Figure 2.2, we present the error in the global L_2 norm, labelled E_G and calculated as follows:

$$E_G = \sqrt{\sum_{m=1}^{M} \int_{\Omega_m} (U^h - U)^2 \, dx}. \tag{2.26}$$

The horizontal axis in the the first plot is the characteristic mesh width, $\tilde{h} = \frac{1}{nDOF}$, where $nDOF = (M)(p + 1)$ is the number of DOFs present in a given simulation with M being the number of elements. At a fixed DOF count, the DG method becomes more accurate as the solution order p is increased. The expected behavior is for the DG scheme to achieve order $p + 1$ convergence, and that behavior is observed in Figure 2.2a. Error is plotted against computational wall time in Figure 2.2b, demonstrating that the higher p simulations give smaller error for a given allocation of wall time.



(a) Error vs. Mesh Width.

(b) Error vs. Wall Time.

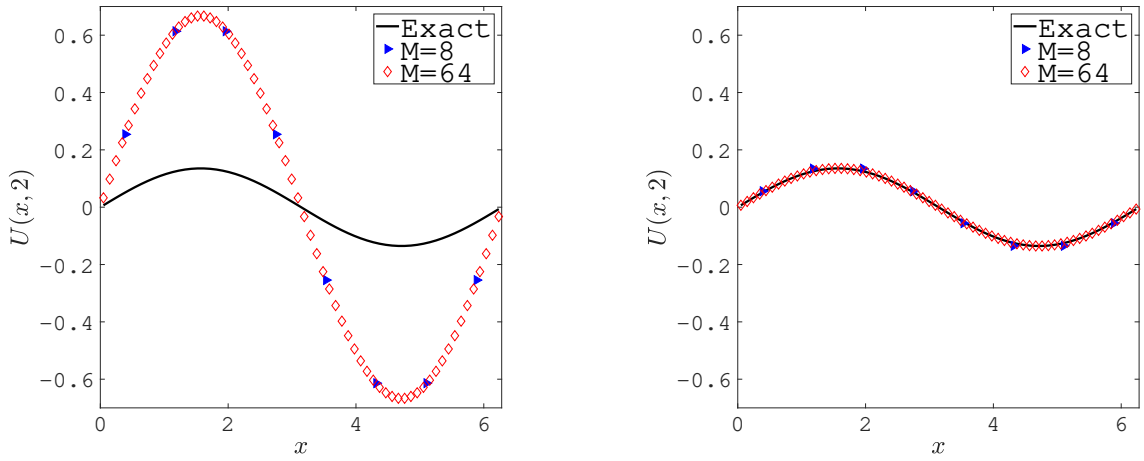
Figure 2.2: Mesh Refinement study for 1D scalar advection equation under spatially periodic boundary conditions. The numerical scheme is the conventional DG scheme with an upwind flux at interfaces. Approximate convergence rates are denoted with m and a dotted gray line; the expected behavior is $m = p + 1$ with respect to characteristic mesh width $\tilde{h}$.

2.3.1 Naive Extension to Diffusion Problems

Moving away from the linear advection case, suppose instead that the governing differential equation is the 1D heat equation (Eq. 2.2). The flux $\mathcal{G}$ depends on the solution gradient, unlike the linear advection case where the flux depends on U. Given the DG weak form for some element Ω_m,

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t}\left(\sum_{n=0}^{p} \hat{U}_m^n \phi_m^n\right) dx = \left[\phi_m^{k-} \tilde{\mathcal{G}}\right]|_{x_m}^{x_m+h_m} - \int_{\Omega_m} \left(\frac{\partial}{\partial x}\phi_m^k\right)(\mathcal{G}) dx, \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \tag{2.27}$$

the most natural approach is to directly differentiate the DG polynomial U^h over the interior of Ω_m to calculate the gradient (for the flux $\mathcal{G}$), then take the average gradient at each interface to calculate the interface flux, denoted $\{\{\nabla U^h\}\}$. The task is simple because the DOFs can be multiplied against



(a) Naive DG, $p = 1$.

(b) 2^{nd} order FV.

Figure 2.3: Results from unsteady heat equation test, spatially periodic domain. The naive DG scheme converges to an incorrect solution as the mesh is refined; the 2^{nd} order finite volume approach gives a satisfactory solution.

the derivatives of the basis functions to approximate the gradient:

$$\frac{\partial U_m^h}{\partial x} = \sum_{n=0}^{p} \hat{U}_m^n \frac{\partial \phi_m^n}{\partial x}. \tag{2.28}$$

We refer to this approach as the *naive* scheme. The resulting residual vector for each Ω_m is

$$R_m^{row} = -\left[\phi_m^{row-} \mu \{\{\nabla U^h\}\}\right]|_{x_m}^{x_m+h_m} + \int_{\Omega_m} \frac{\partial \phi_m^{(row)}}{\partial x} \mu \left(\sum_{n=0}^{p} \hat{U}_m^n \frac{\partial \phi_m^n}{\partial x}\right) dx. \tag{2.29}$$

Unfortunately, the naive approach is unstable and inconsistent regardless of p [117], meaning that as the mesh is refined, the numerical approximation U^h tends towards an erroneous solution as opposed to approaching the exact solution. In Figure 2.3, the naive DG approach with $p = 1$ is compared to a second-order finite volume (FV) scheme. The 1D diffusion equation (Eq. 2.2) is simulated with periodic boundary conditions, diffusivity $\mu = 1$, and initial condition $U(x, 0) = \sin(x)$ from $t = 0$ to $t_{final} = 2.0$ over the domain $x \in [0, 2\pi]$. As the grid is refined (from $M = 8$ to $M = 64$ elements), the DG approximation approaches an erroneous solution while the FV scheme closely matches the exact solution for both mesh sizes. Due to the failure of the naive approach, many modifications to the DG method have been proposed to handle diffusive PDE

behavior [102, 35, 40, 3, 116, 9, 8, 82, 13, 26] (see Arnold et al. [4] for a thorough summary of legacy approaches), and the Interface Gradient Recovery schemes of Chapter 3 are our entry to this set of modifications.

2.4 Fourier Analysis

Fourier analysis (sometimes called von Neumann wavenumber analysis) is a tool used to predict the performance of a spatial discretization. This section describes a basic implementation of the technique for the model equations in 1D to set the stage for more sophisticated techniques in later chapters. In this work, Fourier analysis is applied exclusively to the spatial discretization, ignoring the implications of the time integration scheme in the fully discrete form. The overall goal is to express the DG spatial discretization as a matrix whose eigenvalues give an indication of scheme performance. We describe Fourier analysis in the context of DG, but it is applicable to other spatial discretizations as well and can be used to compare different spatial discretizations [2]. Fourier analysis of the DG method (and the closely related Flux Reconstruction method) is quite common. The most simple approach is to judge scheme accuracy by judging only the consistent eigenvalue and whether or not the scheme is stable by the real components of the full eigenvalue spectrum [56, 55, 117]. More involved analyses discuss the implications of the full eigensystem of the update matrix on solution accuracy [39]. While scheme accuracy is usually expressed in terms of the consistent eigenvalue's order of accuracy, the accuracy can instead be quantified by inspecting error growth rates across a collection of wavenumbers [112, 76]. In rare scenarios, analysis extends to the influence of boundary conditions [75] and the role of the time integration scheme in the fully discrete form [106]. In this section, we review the simplest possible Fourier analysis approach for discontinuous finite element methods; the more complicated resolving efficiency technique [112] will be described and employed in Chapter 4.

Let the governing PDE be either the 1D linear advection equation (Eq. 2.4) or the 1D linear diffusion equation (Eq 2.2) under spatially periodic boundary conditions. Additionally, assume each element to be of uniform width h. Taking Q to be the flux function (either $Q = aU$ or

$Q = -\mu \frac{\partial}{\partial x} U$), consider the DG weak form for element Ω_m:

$$\int_{\Omega_m} \phi_m^k \frac{\partial U^h}{\partial t} dx + \left[\phi_m^{k-} \tilde{Q} \right] |_{x_m}^{x_m+h} - \int_{\Omega_m} \frac{\partial \phi_m^k}{\partial x} Q dx = 0 \quad \forall \phi_m^k \in \boldsymbol{\phi_m}. \tag{2.30}$$

Due the the common flux term ($\tilde{Q}$) along the interfaces of Ω_m, the DG weak form for Ω_m generally involves information from Ω_{m-1} (the element left of Ω_m) and Ω_{m+1} (the element right of Ω_m). Thus, with the flux law being linear, the DG weak form can be rewritten in matrix-vector form using $\hat{\mathbf{U}}_{\mathbf{m-1}}$, $\hat{\mathbf{U}}_{\mathbf{m}}$, and $\hat{\mathbf{U}}_{\mathbf{m+1}}$:

$$\frac{d}{dt} \hat{\mathbf{U}}_{\mathbf{m}} = \boldsymbol{D_L} \hat{\mathbf{U}}_{\mathbf{m-1}} + \boldsymbol{D_C} \hat{\mathbf{U}}_{\mathbf{m}} + \boldsymbol{D_R} \hat{\mathbf{U}}_{\mathbf{m+1}}, \tag{2.31}$$

where the matrices $\boldsymbol{D_L}$, $\boldsymbol{D_C}$, and $\boldsymbol{D_R}$ are specific to the DG spatial discretization of order p. We refer to these matrices as the *differentiation* matrices. Next, the initial condition is taken to be a Fourier mode of dimensional wavenumber ω':

$$U(x, 0) = \exp(i\omega' x), \tag{2.32}$$

where $i = \sqrt{-1}$. The DOFs in the three-cell stencil Ω_{m-1}, Ω_m, Ω_{m+1} are assumed to adhere to the following form in accordance with the initial condition:

$$\hat{\mathbf{U}}_{\mathbf{m+J}} = \exp(iJ\omega) \hat{\mathbf{U}}_{\mathbf{m}}, \tag{2.33}$$

where $\omega = \omega' h$ is the nondimensionalized wavenumber. Large ω corresponds to a relatively large mesh spacing h (a coarse mesh, with few DOFs) and small ω corresponds to a relatively small mesh spacing h (a fine mesh, with many DOFs). Eq. (2.33) is substituted into Eq. (2.31):

$$\frac{d}{dt} \hat{\mathbf{U}}_{\mathbf{m}} = (\exp(-i\omega) \boldsymbol{D_L} + \boldsymbol{D_C} + \exp(i\omega) \boldsymbol{D_R}) \hat{\mathbf{U}}_{\mathbf{m}}, \tag{2.34}$$

forming a system of $p + 1$ ordinary differential equations (ODEs) governing the evolution of the DOFs $\hat{\mathbf{U}}_{\mathbf{m}}$. The differentiation matrices are combined into a single update matrix, denoted $\mathcal{A}$,

which is itself a function of the nondimensionalized wavenumber ω. Additionally, based on the governing PDE under consideration, certain scalar factors are extracted from the differentiation matrices and put in front of the update matrix:

$$\text{Linear advection:} \quad \frac{a}{h}\boldsymbol{\mathcal{A}}(\omega) = \exp(-i\omega)\,\boldsymbol{D_L} + \boldsymbol{D_C} + \exp(i\omega)\,\boldsymbol{D_R}, \tag{2.35a}$$

$$\text{Linear diffusion:} \quad \frac{\mu}{h^2}\boldsymbol{\mathcal{A}}(\omega) = \exp(-i\omega)\,\boldsymbol{D_L} + \boldsymbol{D_C} + \exp(i\omega)\,\boldsymbol{D_R}. \tag{2.35b}$$

The system of ODEs (Eq. 2.34) is rewritten in simpler form:

$$\text{Linear advection:} \quad \frac{d}{dt}\hat{\mathbf{U}}_\mathbf{m} = \frac{a}{h}\boldsymbol{\mathcal{A}}(\omega)\,\hat{\mathbf{U}}_\mathbf{m}, \tag{2.36a}$$

$$\text{Linear diffusion:} \quad \frac{d}{dt}\hat{\mathbf{U}}_\mathbf{m} = \frac{\mu}{h^2}\boldsymbol{\mathcal{A}}(\omega)\,\hat{\mathbf{U}}_\mathbf{m}. \tag{2.36b}$$

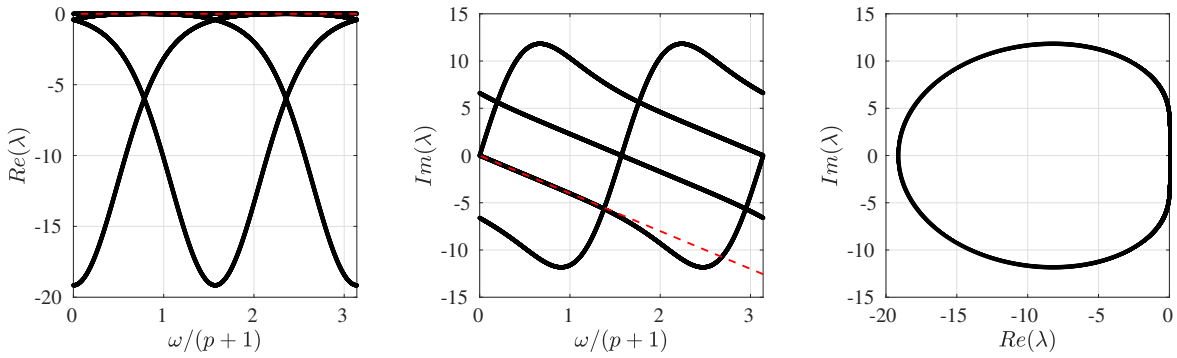
The update matrix $\boldsymbol{\mathcal{A}}(\omega)$ governs the evolution of the DOFs $\hat{\mathbf{U}}_\mathbf{m}$; it depends on the spatial discretization but not the time integration scheme. The eigenvalues of the update matrix are themselves functions of the wavenumber ω and yield three crucial pieces of information:

- If the real component of any eigenvalue of $\boldsymbol{\mathcal{A}}$ is positive for any ω, then the spatial discretization is unstable, meaning that the numerical approximation U^h will grow in an unbounded fashion. Such behavior is undesirable.

- Among the eigenvalues of $\boldsymbol{\mathcal{A}}(\omega)$, one will match the exact eigenvalue, either $\lambda^{ex}(\omega) = 0 - i\omega$ for linear advection or $\lambda^{ex}(\omega) = -\omega^2 + 0i$ for linear diffusion, as ω approaches zero. This one eigenvalue is known as the *consistent* eigenvalue (also referred to as the *principal* eigenvalue) and labelled $\lambda^{con}(\omega)$. The better the consistent eigenvalue matches the exact eigenvalue, the more accurate the scheme is.

- In the case of explicit time integration, the maximum stable timestep size scales as $\frac{h}{\rho_s}$ for linear advection or $\frac{h^2}{\rho_s}$ for linear diffusion, where ρ_s, known as the spectral radius, is the maximum magnitude of any of the $p+1$ eigenvalues of $\boldsymbol{\mathcal{A}}(\omega)$ over all ω.

The order of accuracy and the spectral radius are now discussed in detail.



(a) Real component. (b) Imaginary component. (c) Full spectrum.

Figure 2.4: Eigenvalues of $\mathcal{A}(\omega)$ for the conventional upwind DG scheme with $p = 3$. For each ω, the exact eigenvalue is plotted as the dotted red line, and the $p+1 = 4$ eigenvalues of $\mathcal{A}(\omega)$ are plotted as solid lines. In addition to plotting the real and imaginary components versus wavenumber, we plot the eigenvalues on the complex plane.

2.4.1 Order of Accuracy

Under the chosen setup, with $U(x, 0) = \exp\left(i\frac{\omega}{h}x\right)$, the exact solution is as follows:

$$\begin{aligned} \text{Linear advection:} \quad & U(x,t) = \exp\left(-\frac{i\omega a t}{h}\right)\exp\left(\frac{i\omega x}{h}\right) \\ \text{Linear diffusion:} \quad & U(x,t) = \exp\left(-\left(\frac{\omega}{h}\right)^2 t\right)\exp\left(\frac{i\omega x}{h}\right). \end{aligned} \tag{2.37}$$

These exact solutions correspond to either $\lambda^{ex}(\omega) = -i\omega$ (for linear advection) or $\lambda^{ex}(\omega) = -\omega^2$ (for linear diffusion) as eigenvalues of $\mathcal{A}$. In the case of the DG method, this exact eigenvalue is not present in $\mathcal{A}$. Instead, if the method is implemented properly, there is one eigenvalue, namely the consistent eigenvalue (designated λ^{con}), that closely approximates $\lambda^{ex}(\omega)$ as $\omega \to 0$. This occurrence is illustrated in Figure 2.4, which shows the eigenvalues of the conventional upwind DG discretization with $p = 3$ for the linear advection equation. Note that while the consistent eigenvalue closely approximates λ^{ex} near $\omega = 0$, the other eigenvalues do not. The eigenvalues are in general complex, hence the need to plot both the imaginary component (Figure 2.4b) and the real component (Figure 2.4a). Also, note that instead of using ω on the horizontal axis, the plots use $\frac{\omega}{p+1}$ instead. Plotting the eigenvalues against $\frac{\omega}{p+1}$ allows comparison of DG schemes of different p from a per-gridpoint perspective.

To evaluate scheme accuracy, the consistent eigenvalue is cast as a Taylor expansion involving coefficients $\hat{c}$:

$$\lambda^{con}(\omega) = \sum_{n=0}^{\infty} \hat{c}_n \omega^n, \tag{2.38}$$

such that the error in the consistent eigenvalue is also a Taylor expansion:

$$d(\omega) = \frac{\lambda^{con}(\omega) - \lambda^{ex}(\omega)}{\lambda^{ex}(\omega)} = \sum_{n=0}^{\infty} \hat{d}_n \omega^n. \tag{2.39}$$

If the spatial discretization is *consistent*, then $\hat{d}_0 = 0$. Additionally, the coefficients $\hat{\boldsymbol{d}}$ are zero up to some particular index, designated L:

$$d(\omega) = \frac{\lambda^{con}(\omega) - \lambda^{ex}(\omega)}{\lambda^{ex}(\omega)} = \sum_{n=L}^{\infty} \hat{d}_n \omega^n. \tag{2.40}$$

If $\hat{d}_L \neq 0$ is the first nonzero coefficient, then the scheme's order of accuracy is L. The higher the order of accuracy is, the smaller the error in λ^{con} is near $\omega = 0$. We follow the approach of Huynh [46, 47] to obtain the order of accuracy L: the difference $d(\omega)$ is measured at discrete values of ω, and the rate with which $d(\omega)$ approaches zero as $\omega \to 0$ yields the order of accuracy L. Using an upwind flux at interfaces, the conventional DG method achieves order $2p+1$ accuracy (this result can also be obtained outside of Fourier analysis, see the explanation by Roe [91]). In the case of the linear diffusion equation, most DG methods achieve order $2p$ accuracy. It is the goal of the DG variants proposed in this work to push the discretized advection/diffusion operator closer to the exact eigenvalue for a given solution order p, so Fourier analysis immediately follows the proposal of any new scheme.

2.4.2 Order of Accuracy vs. Rate of Convergence

In this document, since more than one error norm is employed for scheme evaluation, the term *order of convergence* or *rate of convergence* is used to describe the behavior of particular error norms in mesh refinement studies. The term "order of accuracy" is employed exclusively to refer

to the behavior of the principal eigenvalue in Fourier analysis. This behavior goes against the typical practice in the finite element community, which is to use the term "order of accuracy" to describe convergence rates in the global L_2 norm.

The distinction is necessary because Fourier analysis only measures how well a spatial discretization replicates the exact eigenvalue of the governing PDE. In contrast, when a simulation's error is measured by the global L_2 norm or a similar quantity, the error depends not only on how well the spatial discretization replicates the governing PDE, but also how well the finite-dimensional approximation U^h can replicate a globally smooth exact solution. Consider the difference between the exact solution U and the discontinuous polynomial approximation U^h:

$$E(x) = U(x) - U^h(x), \tag{2.41}$$

where $E(x)$ denotes the error at a particular location x. Now, let $\mathcal{P}_{\boldsymbol{\phi}}(U)$ be the projection of the exact solution U onto the polynomial basis $\boldsymbol{\phi}$, such that

$$\int_{\boldsymbol{\Omega}} \phi_m^k \left(\mathcal{P}_{\boldsymbol{\phi}}(U) - U\right) dx = 0, \quad \forall \phi_m^k \in \boldsymbol{\phi}. \tag{2.42}$$

The error is thus rewritten:

$$E(x) = \left(\mathcal{P}_{\boldsymbol{\phi}}(U) + U - \mathcal{P}_{\boldsymbol{\phi}}(U)\right) - U^h, \tag{2.43a}$$

$$E(x) = \left(\mathcal{P}_{\boldsymbol{\phi}}(U) - U^h\right) + \left(U - \mathcal{P}_{\boldsymbol{\phi}}(U)\right). \tag{2.43b}$$

If the update scheme in the DG discretization were perfect, we would have $\left(\mathcal{P}_{\boldsymbol{\phi}}(U) - U^h\right) = 0$. However, $E(x)$ would still, in general, be nonzero because the basis $\boldsymbol{\phi}$ cannot capture all possible functions $U(x)$; in other words, $U - \mathcal{P}_{\boldsymbol{\phi}}(U) \neq 0$. The consequence is that the global L_2 norm is typically limited to rate of convergence $m = p + 1$ even though the order of accuracy for the linear advection equation is $2p + 1$ according to Fourier analysis.

2.4.3 Spectral Radius

The spectral radius ρ_s for a given DG spatial discretization is defined as follows:

$$\rho_s = \max\left(|\lambda_n(\omega)|\right) \quad \text{for} \quad n \in \{0, 1, ..., p\} \quad \text{and} \quad \omega \in [0, \infty), \tag{2.44}$$

where $\lambda_n(\omega)$ is the n^{th} eigenvalue of $\boldsymbol{\mathcal{A}}$ at a specific ω. It is obtained by calculating all $p+1$ eigenvalues of $\boldsymbol{\mathcal{A}}$ for some discrete set of wavenumbers ω, then identifying the maximum eigenvalue magnitude. The behavior of the eigenvalues is periodic in ω, so the spectral radius can be identified by discretizing the wavenumbers in the range $\omega \in [0, 2\pi(p+1)]$ instead of $\omega \in [0, \infty)$. In practice, when the DG spatial discretization is paired with explicit time integration, the maximum stable timestep size depends on both the eigenvalue spectrum of $\boldsymbol{\mathcal{A}}$ and the chosen time integration scheme (i.e., forward Euler, Runge-Kutta, Adams-Bashforth). However, regardless of the explicit time integration technique, the maximum allowable timestep size scales as $\frac{h}{\rho_s}$ for linear advection and $\frac{h^2}{\rho_s}$ for linear diffusion. The review of Fourier analysis is now complete.

2.5 The Discontinuous Galerkin (DG) Method in 2D and 3D

The explanation of DG is now extended to the multidimensional case for a general hyperbolic PDE system with flux $\boldsymbol{\mathcal{F}}$. As described in detail by Cockburn & Shu [27], the Discontinuous Galerkin (DG) method in its semi-discrete form is appropriate for either steady-state or time resolved simulations of hyberbolic PDE systems on arbitrary element geometries and can be extended to arbitrarily high orders of accuracy. Some of the information here is redundant with Section 2.3 to clarify the transition from the 1D case to the multidimensional case. The full extension to an advection-diffusion system is made in Chapter 5.

2.5.1 Geometry Considerations

Given a polygonal spatial domain $\boldsymbol{\Omega}$, the domain is partitioned in to a set of non-overlapping subdomains (elements) such that $\cap_{m=1}^{M} \Omega_e = \emptyset$ and $\cup_{m=1}^{M} \overline{\Omega}_e = \overline{\boldsymbol{\Omega}}$, where $\overline{\Omega}_e$ denotes the closure, i.e., $\overline{\Omega}_m = \Omega_m \cup \partial\Omega_m$. The boundary of the Ω_m is denoted $\partial\Omega_m$. With regard to notation, the boldface $\boldsymbol{x}$ denotes a location in N_D-dimensional space, where N_D is the number of spatial dimensions (either 1, 2, or 3 in this work). Each element's set of basis functions $\boldsymbol{\phi_m}$ is a polynomial basis of at most degree p in a given direction and contains K members.

In this work, the DG method is always applied with uniform polynomial order for all elements. This constraint discards the possible benefits of an hp-adaptive mesh refinement procedure. For quadrilateral elements in 2D and hexahedral elements in 3D, a full tensor product basis of degree p in each direction is employed, such that $K = (p+1)^{N_D}$, where N_D is the number of spatial dimensions. For simplex (triangular) elements in 2D, $K = \frac{1}{2}(p+1)(p+2)$. As with the 1D case, each element in the physical domain is mapped to the single reference element, Ω_{ref}, which is spanned by an appropriate choice of reference coordinates. We consider only straight-edged elements, such that the geometry of a given Ω_m is completely defined by the N_V vertices of the element. The reference element should have the same vertex count as the corresponding physical element. In the case of 2D quadrilateral elements, given a point $\boldsymbol{\xi} \in [-1,1]^2$ on the reference square Ω_{ref}, the corresponding physical location $\boldsymbol{x}(\boldsymbol{\xi})$ on a quadrilateral element Ω_m is defined as follows:

$$\begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} x_{v1} & x_{v2} & x_{v3} & x_{v4} \\ y_{v1} & y_{v2} & y_{v3} & y_{v4} \end{bmatrix} \begin{bmatrix} (1-\xi)(1-\eta) \\ (1+\xi)(1-\eta) \\ (1+\xi)(1+\eta) \\ (1-\xi)(1+\eta) \end{bmatrix}, \tag{2.45}$$

where $\boldsymbol{x}_{v1}$, $\boldsymbol{x}_{v2}$, $\boldsymbol{x}_{v3}$, and $\boldsymbol{x}_{v4}$ are the four vertices of Ω_m. If instead the domain is partitioned with triangular (simplex) elements, and an equilateral triangle with vertices $\{\left(-1, -\frac{1}{\sqrt{3}}\right), \left(1, -\frac{1}{\sqrt{3}}\right), \left(0, \frac{2}{\sqrt{3}}\right)\}$

is used as the reference element, the transformation takes the following form:

$$\begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \begin{bmatrix} x_{v1} & x_{v2} & x_{v3} \\ y_{v1} & y_{v2} & y_{v3} \end{bmatrix} \begin{bmatrix} \frac{1}{3} - \frac{\xi}{2} - \frac{\eta}{2\sqrt{3}} \\ \frac{1}{3} + \frac{\xi}{2} - \frac{\eta}{2\sqrt{3}} \\ \frac{1}{3} + \frac{\eta}{\sqrt{3}} \end{bmatrix}. \tag{2.46}$$

For both cases, the mapping between the physical and reference coordinates is differentiated to produce the Jacobian matrix for each element:

$$\boldsymbol{J_m}(\xi,\eta) = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}, \tag{2.47}$$

which in general varies over the element. The Jacobian matrix is necessary to calculate the gradients of the basis functions, $\boldsymbol{\phi}$, on the physical elements. The basis functions are defined as polynomials in the reference coordinates on Ω_{ref}, so their derivatives with respect to the reference coordinates are known. The Jacobian matrix and its inverse provide the means to transform derivatives between the reference coordinates and the physocal coordinates. In the 2D case, we have

$$\begin{bmatrix} \frac{\partial \phi_m^k}{\partial \xi} & \frac{\partial \phi_m^k}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi_m^k}{\partial x} & \frac{\partial \phi_m^k}{\partial y} \end{bmatrix} \boldsymbol{J_m}, \quad \therefore \quad \begin{bmatrix} \frac{\partial \phi_m^k}{\partial x} & \frac{\partial \phi_m^k}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi_m^k}{\partial \xi} & \frac{\partial \phi_m^k}{\partial \eta} \end{bmatrix} \boldsymbol{J_m}^{-1} \tag{2.48}$$

for some basis function ϕ_m^k of element Ω_m.

Integrals in physical space are computed using the reference element and the Jacobian matrix. Given a particular physical element Ω_m and the corresponding Jacobian matrix $\boldsymbol{J_m}$,

$$\int_{\Omega_m} f d\boldsymbol{x} = \int_{\Omega_{ref}} f |\boldsymbol{J_m}| d\boldsymbol{\xi}, \tag{2.49}$$

where $|\boldsymbol{J_m}|$ is the determinant of the element's Jacobian matrix.

2.5.2 The Spatial Discretization

Consider an initial-value problem of the form

$$U(\boldsymbol{x}, 0) = U_{IC}(\boldsymbol{x}), \tag{2.50a}$$

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}}(U) = 0, \tag{2.50b}$$

subject to appropriate boundary conditions, where $\boldsymbol{\mathcal{F}}(U)$ is the flux function. Instead of each element having $p+1$ basis functions, each element is now said to have K basis functions and K DOFs, where K depends on the element geometry and the solution order p. The approximation U^h is again a piecewise continuous polynomial defined by time-dependent DOFs and space-dependent basis functions:

$$U^h\left(\boldsymbol{x} \in \Omega_m, t\right) = U_m^h(\boldsymbol{x}, t) = \sum_{k=0}^{K-1} \phi_m^k(\boldsymbol{\xi}(\boldsymbol{x})) \hat{U}_m^k(t). \tag{2.51}$$

As with the 1D case, each ϕ_m^k is nonzero only over Ω_m. The goal of the polynomial approximation U^h is to satisfy the governing PDE and the initial condition in the integral sense over the spatial domain $\boldsymbol{\Omega}$:

$$\int_{\boldsymbol{\Omega}} v \left(\frac{\partial}{\partial t} U + \nabla \cdot \boldsymbol{\mathcal{F}}\left(U\right) \right) d\boldsymbol{x} = 0, \quad \forall v \in \boldsymbol{\phi}, \tag{2.52a}$$

$$\int_{\boldsymbol{\Omega}} v \left(U\left(\boldsymbol{x}, 0\right) - U_{IC}\left(\boldsymbol{x}\right) \right) d\boldsymbol{x} = 0, \quad \forall v \in \boldsymbol{\phi}, \tag{2.52b}$$

where $\boldsymbol{\phi}$ denotes the collection of all of the basis functions for all elements. As with the 1D case, we substitute in U^h for U, split the integrals into individual element integrals, and perform integration by parts to obtain the DG weak form:

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t} \left(\sum_{n=0}^{K-1} \hat{U}_m^n \phi_m^n \right) d\boldsymbol{x} = - \int_{\partial \Omega_m} \phi_m^{k-} \left(\tilde{\boldsymbol{\mathcal{F}}} \cdot \mathbf{n}_{\mathbf{m}}^- \right) ds + \int_{\Omega_m} \nabla \phi_m^k \cdot \boldsymbol{\mathcal{F}} d\boldsymbol{x}, \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \boldsymbol{\Omega}, \tag{2.53}$$

which yields a total of $K \times M$ constraints per field variable for the vector of DOF time derivatives, $\frac{d}{dt}\hat{\mathbf{U}}$. The $\mathbf{n}_\mathbf{m}^-$ term is the outward normal vector from Ω_m along $\partial\Omega_m$.

The interface flux $\tilde{\boldsymbol{\mathcal{F}}}$ must be single-valued along each interface to enforce flux continuity; it is calculated from the competing DG approximations along each interface, as with the 1D case. The presence of two solution states at a given location along an interface is known as a Riemann problem. Given two approximations (call them U_L and U_R) at a given location on an interface, the DG weak form requires that a single value for the flux be chosen. In the linear advection case, one may simply take the U value from the upwind side of the interface and use it to calculate the flux. For a general hyperbolic system of equations, population of $\tilde{\boldsymbol{\mathcal{F}}}$ requires the use of an exact or approximate Riemann solver. As its name suggests, the Riemann solver's purpose is to calculate the interface flux $\tilde{\boldsymbol{\mathcal{F}}}$ based on the Riemann problem introduced by the two competing solution states. In this work, we write

$$\tilde{\boldsymbol{\mathcal{F}}} = Rie\left(U_L, U_R, \mathbf{n}\right) \tag{2.54}$$

to denote the application of the Riemann solver at a given point along a given interface. The necessary arguments to the Riemann solver are the solution state on the left side of the interface (denoted U_L), the solution state on the right side of the interface (denoted U_R), and the interface's normal vector (denoted $\mathbf{n}$), which points outwards from the left region. For a lucid explanation of approximate Riemann solvers (and the exact Riemann solver) for ideal compressible flow (i.e., the Euler equations), see the textbook of Toro [99]. Let $\boldsymbol{x}_0$ be a given point along the interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$ shared by Ω_A and Ω_B. In the conventional DG discretization, the interface flux is calculated as follows:

$$\tilde{\boldsymbol{\mathcal{F}}}|_{\boldsymbol{x}_0} = Rie\left(U_A^h|_{\boldsymbol{x}_0}, U_B^h|_{\boldsymbol{x}_0}, \mathbf{n}_\mathbf{A}^-\right), \tag{2.55}$$

where $\mathbf{n}_\mathbf{A}^-$ is the outward normal from Ω_A at $\boldsymbol{x}_0$. The solution states $U_A^h|_{\boldsymbol{x}_0}$ and $U_B^h|_{\boldsymbol{x}_0}$ are the limits of U_A^h and U_B^h as $\boldsymbol{x}_0$ is approached from inside Ω_A and inside Ω_B, respectively. In Chapter 4, it will be shown how this procedure can be modified to improve the accuracy of the DG method.

Over the interior of each Ω_m, $\boldsymbol{\mathcal{F}}$ is directly calculated using the DG approximation, U_m^h. To

discretize the left-hand side of Eq. (2.53), each element's $K \times K$ mass matrix,

$$\mathcal{M}_m^{row,col} = \int_{\Omega_m} \phi_m^{(row)} \phi_m^{(col)} d\boldsymbol{x}, \tag{2.56}$$

is employed to isolate the time derivatives of the DOFs. The right-hand side of Eq. (2.53) is assembled in the K-row residual vector $\boldsymbol{R_m}$ for each Ω_m and combined with the mass matrix to form a system of ODEs:

$$R_m^{row}\left(U^h\right) = \int_{\partial\Omega_m} \phi_m^{(row-)} \left(\tilde{\boldsymbol{\mathcal{F}}} \cdot \mathbf{n}_\mathbf{m}^-\right) ds - \int_{\Omega_m} \nabla\phi_m^{(row)} \cdot \boldsymbol{\mathcal{F}}\left(\boldsymbol{U_m^h}\right) d\boldsymbol{x}, \tag{2.57a}$$

$$\frac{d}{dt}\hat{\mathbf{U}}_\mathbf{m} = \boldsymbol{\mathcal{M}_m}^{-1}\left(-\boldsymbol{R_m}\left(U^h\right)\right), \qquad \forall\Omega_m \in \boldsymbol{\Omega}. \tag{2.57b}$$

The resulting system of ODEs in time (Eq. 2.57b) is discretized with an explicit Runge-Kutta method (a family that includes the forward Euler method) to integrate the system forward in time. As with the 1D case, the integrals appearing in the weak form are populated by a Gaussian quadrature rule with a sufficient number of points to complement the polynomial order p. We use the typical approach of populating the flux at the set of quadrature points, then using the resulting distribution of the flux to calculate the integrals.

In this description of the DG method, a few outstanding issues have been overlooked. These include the destabilization of the method in the presence of physical discontinuities (such as shock waves), the issue of polynomial aliasing when the flux law is nonlinear, and the need for preconditioners when the spatial discretization is paired with an implicit time integration scheme. As the goal of this thesis is to improve the accuracy of the DG spatial discretization itself, which serves as the foundation of a DG-based CFD solver, the presented explanation is sufficient.

2.5.3 Compactness

An important feature of the conventional DG discretization is that the residual for the DOFs in a given element depend only on the DOFs in that element and its face-connected neighbors, as

Figure 2.5: Stencil illustration. To form the residual in the center element (Ω_0), a scheme with a compact stencil (i.e., conventional upwind DG for advection) requires information only from the face-connected (nearest) neighbors (X) of Ω_0. In the Cartesian case, certain non-compact schemes (such as the RDG-1x++CO scheme that we work with in Chapter 6) can be optimized to use only the vertex-connected neighbors of Ω_0 (X, Y). Schemes with non-compact stencils generally require information from an extra layer of elements (X, Y, Z) compared to the compact case.

illustrated in Figure 2.5. In other words, the stencil is compact. In the case of hyperbolic PDEs, there exist reconstruction schemes, such as the $P_N P_M$ approach of Dumbser [31], the cell-centered reconstruction schemes of Khieu & Johnsen [56], and the reconstructed DG scheme of Luo et al. [70] that extend the stencil past the face-connected neighbors in the interest of improving the spatial discretization's accuracy. In the case of parabolic PDEs, since the mixed formulation (detailed in Chapter 4) involves solving a pair of first-order systems during each residual evaluation, it is not rare to encounter non-compact DG schemes. A compact scheme is advantageous for computational efficiency, especially in the parallelized case where element DOFs must be exchanged across different processing cores. The three Recovery-assisted DG schemes revealed in Chapter 5 for advection-diffusion problems all maintain a compact computational stencil.

2.6 The Recovery operator

This section describes the recovery operator [102] within the DG framework. Application of the recovery operator is detailed in Chapters 3, 4, 5, 6, and 7; in this section, our only interest is a thorough description of the operator itself. In the construction of our Recovery-assisted schemes for advection-diffusion problems, the recovery operator is employed exclusively as a tool to approximate the solution U along element-element interfaces. The Recovery concept originates from an important observation [102]: the discontinuous polynomial form of U^h is an attempt to replicate some globally smooth, underlying exact solution U. The recovery operator is an intelligent

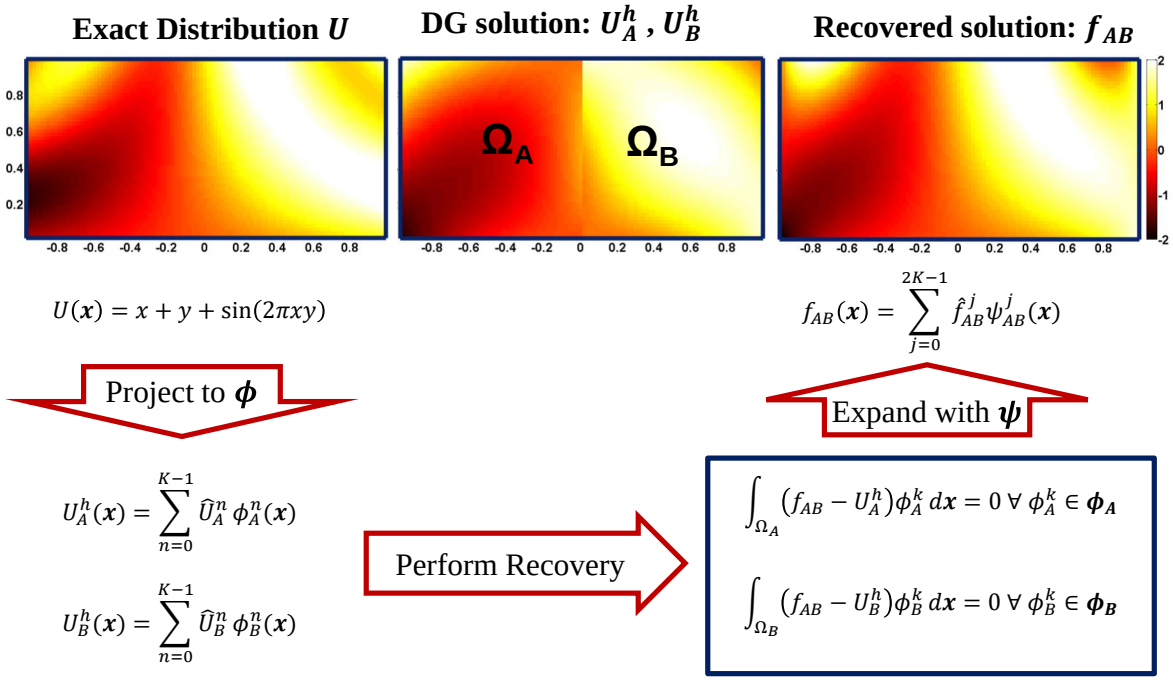


Figure 2.6: The Recovery process in 2D. The recovery operator projects the discontinuous DG polynomial U^h into a smooth polynomial basis $\boldsymbol{\psi}$ to "recover" the smooth exact solution U. Plotted result uses a $p = 2$ ($K = 9$) discretization on Cartesian elements. While the DG approximation U^h is discontinuous along the interface between the elements, the recovered solution is smooth across the entire union.

attempt to "recover" this underlying exact solution over a subdomain of the global spatial domain $\boldsymbol{\Omega}$. For reasons of practicality, this subdomain is always chosen to be the union of just two adjacent elements. An alternative perspective is cast as a question: given some DG approximation U^h, what would the solution look like if two adjacent elements (Ω_A and Ω_B) were combined into a single, larger element, without discarding any of the information contained in $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$? The answer to this question is the recovered solution, which preserves all the information of $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$ in the formation of a smooth polynomial over $\Omega_A \cup \Omega_B$. The idea is illustrated in Figure 2.6, adapted from Johnson & Johnsen [49]. The discontinuous polynomial U^h approximates the underlying smooth exact solution, U. When the DG approximation U^h is fed through the recovery operator, it produces a smooth polynomial f that shows excellent agreement with the exact solution U over the union $\Omega_A \cup \Omega_B$. With the basic idea now understood, the recovery procedure is described from

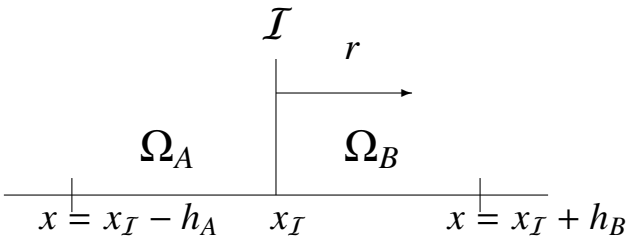


Figure 2.7: Setup of two neighboring elements in 1D, linked by $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$.

a technical perspective, beginning with the 1D case.

2.6.1 Recovery in 1D

Let Ω_A and Ω_B be a pair of neighboring elements of widths h_A and h_B, respectively, sharing an interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$, as shown in Figure 2.7. The DG approximation in each element is a degree p polynomial with $K = p + 1$ DOFs. In addition to the DG bases $\boldsymbol{\phi_A}$ and $\boldsymbol{\phi_B}$ in the two elements, we introduce the Recovery basis:

$$\boldsymbol{\psi} = \{\psi^m(r) = \mathcal{L}^m(r) \mid m \in \{0, 1, ...K_R - 1\}\}, \tag{2.58}$$

where $\mathcal{L}^m(r)$ is the degree m Legendre polynomial and r is a function of x. Specifically,

$$r(x) = \frac{(x - x_{\mathcal{I}}) \cdot n_A^-}{\min(h_A, h_B)}. \tag{2.59}$$

As demonstrated in Figure 2.7, r originates at the interface and is known as the *recovery coordinate.* The dimension of the recovery basis, denoted K_R, depends on the type of recovery. Regardless of the type of recovery, the basis $\boldsymbol{\psi}$ is supported over $\Omega_A \cup \Omega_B$ and is smooth across the entire union. Let $W(r)$ be a smooth polynomial in the recovery basis:

$$W(r) = \sum_{m=0}^{K_R-1} \psi^m(r)\hat{W}^m, \tag{2.60}$$

where the coefficients $\hat{\boldsymbol{W}}$ are constrained according to the following two sets of equations:

$$\int_{\Omega_A} \left(W - U_A^h\right) \Theta_A^k dx = 0, \quad \forall k \in \{0, 1, ..., N_A - 1\}, \tag{2.61a}$$

$$\int_{\Omega_B} \left(W - U_B^h\right) \Theta_B^k dx = 0, \quad \forall k \in \{0, 1, ..., N_B - 1\}. \tag{2.61b}$$

These equations constrain the approximation W to be indistinguishable from U^h, in the weak sense, with respect to the testing functions $\boldsymbol{\Theta_A}$ and $\boldsymbol{\Theta_B}$. The polynomial W is subject to N_A constraints in Ω_A and N_B constraints in Ω_B. The test functions $\boldsymbol{\Theta}$ depend on the particular type of recovery. To match the dimension of $\boldsymbol{\psi}$ to the number of constraints, $K_R = N_A + N_B$. The constraints of Eq. (2.61) yield the following linear system for a given field variable:

$$\underbrace{\begin{bmatrix} \int_{\Omega_A} \psi^{col} \Theta_A^0 dx \\ \int_{\Omega_A} \psi^{col} \Theta_A^1 dx \\ \vdots \\ \int_{\Omega_A} \psi^{col} \Theta_A^{N_A-1} dx \\ - - - - - - \\ \int_{\Omega_B} \psi^{col} \Theta_B^0 dx \\ \int_{\Omega_B} \psi^{col} \Theta_B^1 dx \\ \vdots \\ \int_{\Omega_B} \psi^{col} \Theta_B^{N_B-1} dx \end{bmatrix}}_{K_R \times K_R} \begin{bmatrix} \hat{W}^0 \\ \hat{W}^1 \\ \vdots \\ \hat{W}^{K_R-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \int_{\Omega_A} \phi_A^{col} \Theta_A^0 dx & [0]_{1\times K} \\ \int_{\Omega_A} \phi_A^{col} \Theta_A^1 dx & [0]_{1\times K} \\ \vdots & \vdots \\ \int_{\Omega_A} \phi_A^{col} \Theta_A^{N_A-1} dx & [0]_{1\times K} \\ - - - & - - - \\ [0]_{1\times K} & \int_{\Omega_B} \phi_B^{col-K} \Theta_B^0 dx \\ [0]_{1\times K} & \int_{\Omega_B} \phi_B^{col-K} \Theta_B^1 dx \\ \vdots & \vdots \\ [0]_{1\times K} & \int_{\Omega_B} \phi_B^{col-K} \Theta_B^{N_B-1} dx \end{bmatrix}}_{K_R \times 2K} \begin{bmatrix} \hat{\mathbf{U}}_\mathbf{A} \\ \hat{\mathbf{U}}_\mathbf{B} \end{bmatrix}, \tag{2.62}$$

which is inverted to solve for the coefficient vector $\hat{\boldsymbol{W}}$. Then, the coefficients are multiplied by the recovery basis functions (Eq. 2.60) to yield the solution along the interface. In practice, this entire process is stored in the discrete recovery operator, $\boldsymbol{\mathcal{R}}$, for a given interface:

$$W(r = 0) = \boldsymbol{\mathcal{R}} \begin{bmatrix} \hat{\boldsymbol{U}}_A \\ \hat{\boldsymbol{U}}_B \end{bmatrix}, \tag{2.63}$$

where $\mathcal{R}$ has only one row but $2K$ columns. The formation of the discrete recovery operator $\mathcal{R}$ is described in Appendix A for the interested reader. With the implementation understood, the classical (full-order) and biased recovery types are described in the context of Eq. (2.61).

2.6.2 Classical (Full-Order) Recovery

In this case, the testing functions are taken from the DG solution basis: $\Theta_A^k(x) = \phi_A^k(x)$ for all $k < K$ in Ω_A, and in Ω_B, $\Theta_B^k(x) = \phi_B^k(x)$ for all $k < K$. Weak equivalence is enforced with respect to all functions in the DG solution space: $N_A = N_B = K$. The dimension of the recovery basis is $K_R = 2K$. In this case, W is the *recovered solution*, denoted f (in keeping with the convention of van Leer & Nomura [102]). This approach is known as *full-order* recovery because the total constraint count matches the DOF count in the union of the two elements. It is the original form of the recovery procedure and has been used [67] to build many attractive DG schemes for the diffusion equation (Eq. 2.7).

2.6.3 Biased Recovery

Khieu & Johnsen [56] note that either N_A or N_B can be taken to be less than K in Eq. (2.61) and the approximation W is usually more accurate (closer to the underlying exact solution) than either U_A^h or U_B^h at the interface. The biased recovery operator is applied to form their interface-centered binary (ICB) reconstruction scheme, specifically the approach labelled icbp[0]. The biased recovery approach builds two reconstructions over $\mathcal{U} = \Omega_A \cup \Omega_B$. The A-biased reconstruction $W = U_A^{ICB}$ closely resembles U_A^h but contains some information from Ω_B; in contrast, the B-biased reconstruction $W = U_B^{ICB}$ closely resembles U_B^h but contains some information from Ω_A. This biasing is achieved as follows:

- For the A-biased solution, $N_A = K$, $N_B = 1$, $\Theta_A^k = \phi_A^k \ \forall k < K$, and $\Theta_B^0 = 1$. Consequently, over Ω_B, U_A^{ICB} and U^h have the same average. Over Ω_A, U_A^{ICB} and U^h are weakly equivalent with respect to $\boldsymbol{\phi_A}$.

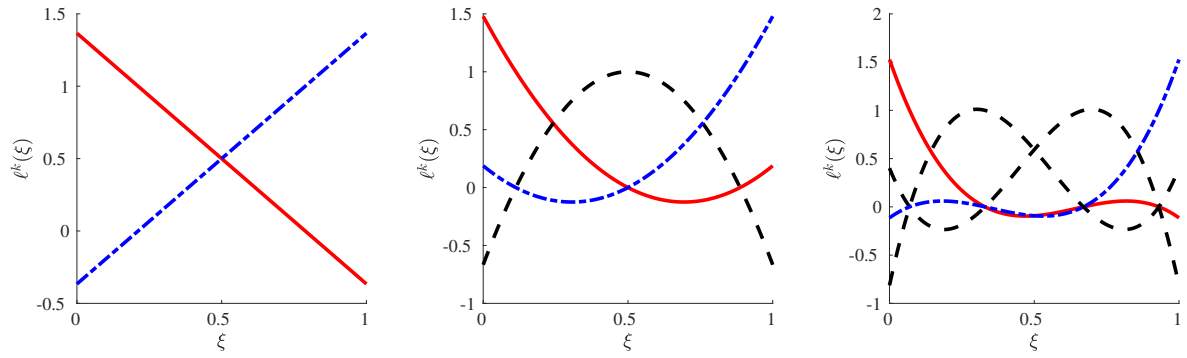
- In contrast, for the B-biased solution, $N_A = 1$, $N_B = K$, $\Theta_B^k = \phi_B^k \ \forall k < K$, and $\Theta_A^0 = 1$. Consequently, over Ω_A, U_B^{ICB} and U^h have the same average. Over Ω_B, U_B^{ICB} and U^h are weakly equivalent with respect to $\boldsymbol{\phi_B}$.

This approach [56], where $\Theta = 1$ in the non-dominant element, is labelled *Modal ICB*, abbreviated ICBM. We coined the term Modal ICB because $\Theta = 1$ is the zeroth mode of a modal polynomial expansion. The reason that the biased recovery approach is restricted to $N_B = 1$ (for the A-biased reconstruction) and $N_A = 1$ (for the B-biased reconstruction) is that employing a higher-order approximation W produces an unstable numerical scheme for the linear advection equation, as discovered by Khieu & Johnsen [56] and verified by Frahan & Johnsen [43] through $p = 5$. The numerical scheme resulting from the use of Modal ICB inside the DG framework will be discussed in Chapter 4.

In [51], we introduced a small alteration to the biased recovery operation. Consider the A-biased reconstruction, $W = U_A^{ICB}$. Again taking $N_A = K$ and $N_B = 1$, let $\Theta_A^k(x) = \phi_A^k(x) \ \forall k < K$. Now, in contrast to the Modal ICB approach, define $\boldsymbol{\Theta_B} = \boldsymbol{\ell_B}$ to be the degree-p Lagrange basis whose interpolation points are the set of $p + 1$ Gauss-Legendre points in Ω_B; such a basis is illustrated in Figure 2.8. To apply the single constraint in Ω_B, take $\Theta_B^0 = \ell_B^0$ to be the Lagrange polynomial that is unity at the Gauss-Legendre point closest to the interface. Similarly, the B-biased reconstruction is obtained by taking $\Theta_B^k(x) = \phi_B^k(x) \ \forall k < K$ and setting the single weighting function in Ω_A to be the Lagrange polynomial that is unity at the Gauss-Legendre point closest to the interface. Due to the choice of a Lagrange testing basis for $\boldsymbol{\Theta}$ (regardless of the solution basis $\boldsymbol{\phi}$), this reconstruction approach is labelled *Lagrange ICB*. The benefits of this particular biased recovery technique are explored in Chapter 4.

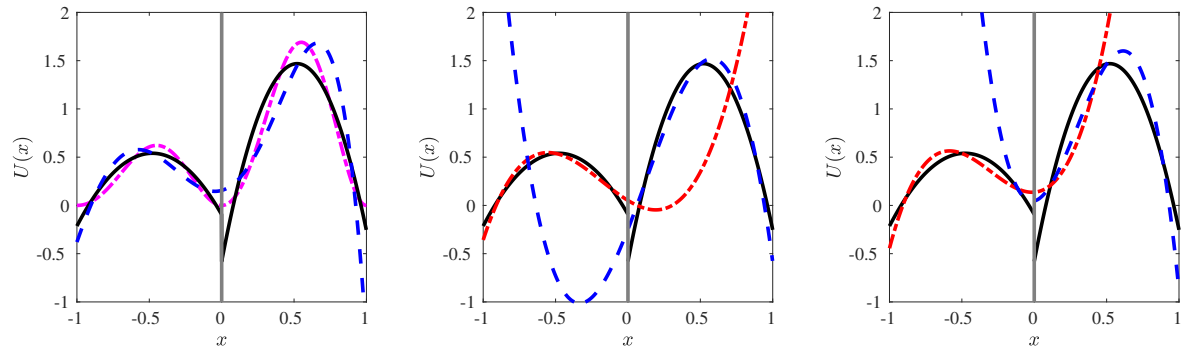
2.6.4 Comparison of Recovery Types in 1D

The three flavors of recovery (full-order, Modal ICB, and Lagrange ICB) are summarized in Table 2.1. Figure 2.9 shows the behavior of the three types of recovery when applied to the same discontinuous approximation U^h. Away from the interface, the ICB reconstructions diverge severely



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 2.8: Lagrange testing functions $\boldsymbol{\ell}$ for the Lagrange ICB discretization, plotted on the unit interval. The solid red line (——) is taken as ℓ_B^0 when performing A-biased recovery. The dot-dash blue line (— - — - —) is taken as ℓ_A^0 when performing B-biased recovery. The remaining Lagrange polynomials (dotted black lines, — — — —) are unused.



(a) Classical recovery. Key: Exact: — - — - ; f: — — — .

(b) Modal ICB recovery. Key: U_A^{ICBM}: — - — ; U_B^{ICBM}: — — — .

(c) Lagrange ICB recovery. Key: U_A^{ICBL}: — - — ; U_B^{ICBL}: — — — .

Figure 2.9: Sample recovered solutions for $p = 2$ with $\Omega_A = \{x \in (-1, 0)\}$, $\Omega_B = \{x \in (0, 1)\}$, and $x_{\mathcal{I}} = 0$. The DG approximation U^h (solid black lines,——) is initialized via Galerkin equivalence (Eq. 5.2c) to an initial condition, $U_{IC}(x) = e^x \sin^2(\pi x)$. Then, the various recovery operators are applied to the DG data to reconstruct the underlying smooth solution.

from the exact solution; this property is tolerable because ultimately, the only purpose of the ICB procedure (to be elaborated on in Chapter 4) is to produce an accurate approximation along the interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$.

2.6.5 Extension to Multiple Spatial Dimensions

The description of the full-order recovery operator is now extended to the 2D case, from which the reader can properly extrapolate the 3D implementation on Cartesian elements. The necessary modifications for the biased recovery procedure (Modal ICB) in 2D are discussed afterwards. Suppose U^h to be a degree-p polynomial in each element. For a given union of face-connected elements Ω_A and Ω_B linked by interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$, the recovered solution $f_\mathcal{I}$ (particular to the interface) is a polynomial expansion in the basis $\boldsymbol{\psi}$:

$$f_\mathcal{I}(r,s) = \sum_{n=0}^{K_R-1} \hat{f}_\mathcal{I}^n \psi^n(r,s), \quad \boldsymbol{\psi} = \{\psi^n(r,s) = L^a(r)L^b(s), \quad (a,b) \in \{0,...,p_r\} \times \{0,...,p_s\}\}, \quad (2.64)$$

where $L^j(\zeta)$ represents the Legendre polynomial of degree j taken at location ζ. Any other degree $2p+1$ hierarchical polynomial basis set is also acceptable. In our implementation, to maximize robustness on non-Cartesian meshes, the recovery coordinates r and s originate at the average location ($\overline{\mathbf{x}}_0$) of the two element centroids, with r running along the axis connecting the centroids and s being the tangential coordinate, as shown in Figs. 2.10a and 2.10c for Cartesian and simplex elements, respectively. The recovery coordinates are scaled such that the maximum magnitudes of both r and s are unity over $\Omega_A \cup \Omega_B$; the two-component vector containing r and s is denoted $\boldsymbol{r} = (r,s)$. The recovery process requires that the recovery coordinates $\boldsymbol{r} = (r,s)$ be obtainable as a function of the physical coordinates $\boldsymbol{x}$. This transformation, $\boldsymbol{r} = \boldsymbol{C}(\boldsymbol{x} - \overline{\mathbf{x}}_0)$, requires knowledge of the vertices of the two elements and is described in detail by Lo [67]. Given a physical location $\boldsymbol{x}$,

Table 2.1: Forms of Recovery, constrained according to Eq. (2.61).

Scheme	Solution	$\boldsymbol{\Theta_A}$	$\boldsymbol{\Theta_B}$	N_A	N_B	K_R
Classical (full-order)	$W = f$	$\boldsymbol{\phi_A}$	$\boldsymbol{\phi_B}$	K	K	$2K$
Modal ICB	$W = U_A^{ICBM}$	$\boldsymbol{\phi_A}$	1	K	1	$K+1$
Modal ICB	$W = U_B^{ICBM}$	1	$\boldsymbol{\phi_B}$	1	K	$K+1$
Lagrange ICB	$W = U_A^{ICBL}$	$\boldsymbol{\phi_A}$	$\boldsymbol{\ell_B}$	K	1	$K+1$
Lagrange ICB	$W = U_B^{ICBL}$	$\boldsymbol{\ell_A}$	$\boldsymbol{\phi_B}$	1	K	$K+1$

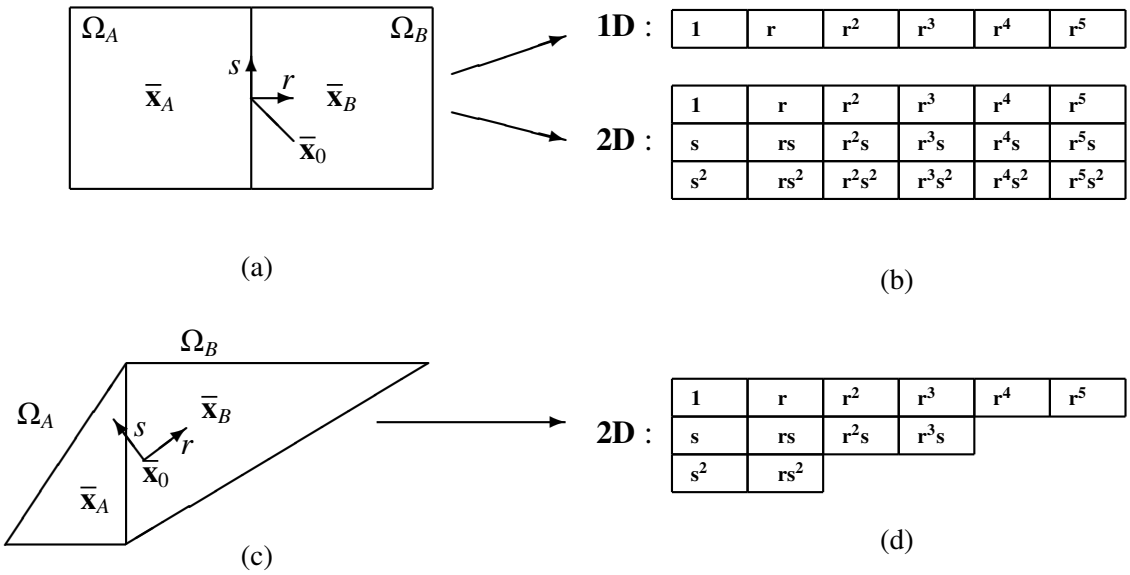


Figure 2.10: (a) A pair of neighboring Cartesian elements Ω_A and Ω_B. (b) Sample $p = 2$ recovery bases $\boldsymbol{\psi}$ for the 1D case and the 2D Cartesian case. (c) A pair of neighboring simplex elements Ω_A and Ω_B. (d) Sample $p = 2$ recovery basis $\boldsymbol{\psi}$ for the simplex case. Here, $\bar{\mathbf{x}}$ marks the centroid of each element, and $\bar{\mathbf{x}}_0$ is the average position of the two centroids.

we first calculate the path between $\bar{\mathbf{x}}_0$ and $\boldsymbol{x}$. Then, the N_D-component path vector is rotated and scaled through the transformation matrix $\boldsymbol{C}$ to yield the corresponding recovery coordinates $\boldsymbol{r}$.

The arrangement of $\boldsymbol{\psi}$, specifically the values of the index limits p_r and p_s in Eq. (2.64), depends on the element geometry. For 1D elements, $(p_r, p_s) = (2p + 1, 0)$. For 2D quadrilateral elements, $(p_r, p_s) = (2p + 1, p)$, and $\boldsymbol{\psi}$ is formed as an anisotropic tensor product basis; the two arrangements are illustrated in Fig. 2.10b. For illustrative purposes, the recovery basis is constructed via a tensor product basis from the 1D Taylor basis. In the case of 2D simplex elements, p_r depends on the index b (see Eq. 2.64 for the role of b); specifically, $p_r = 2(p - b) + 1$ with $p_s = p$. This setup is illustrated in Fig. 2.10d. So far, the 3D extension of the method has only addressed hexahedral elements. In this case, the tensor product recovery basis is degree $2p + 1$ in the face-normal direction and degree p in the two face-tangential directions, such that the dimension of the recovery basis is $K_R = 2(p + 1)^3$.

Given the recovery basis $\boldsymbol{\psi}$, the coefficients $\hat{\boldsymbol{f}}_I$ of the recovered solution are constrained by requiring equality between U^h and f_I in the weak sense over Ω_A and Ω_B. Specifically, with respect to test functions $\phi_A^k(\boldsymbol{x})$ and $\phi_B^k(\boldsymbol{x})$, the interface's recovered function f_I satisfies K equivalence

relations over each element:

$$\int_{\Omega_A} \left(f_I - U_A^h\right) \phi_A^k d\boldsymbol{x} = 0 \quad \text{and} \quad \int_{\Omega_B} \left(f_I - U_B^h\right) \phi_B^k d\boldsymbol{x} = 0, \quad \forall k \in \{0, ..., K-1\}. \tag{2.65}$$

In practice, the recovery constraints (Eq. 2.65) are recast in matrix-vector form such that the recovery coefficients $\hat{\boldsymbol{f}}_I$ are obtainable directly from the DG DOFs $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$ of the two neighboring elements through a matrix-vector multiplication. Once the coefficients $\hat{\boldsymbol{f}}_I$ are known for the shared interface $\mathcal{I}$, they are combined with the basis $\boldsymbol{\psi}$ according to Eq. (2.64) to form the recovered polynomial across the interface. Specifically, given the interface's quadrature points $\boldsymbol{r}_\mathbf{1}$ through $\boldsymbol{r}_{\boldsymbol{Q}_S}$, where Q_S is the number of quadrature points on the interface, the discrete recovery operator $\mathcal{R}$ yields the distribution of the recovered solution given the neighboring solution DOFs:

$$\begin{bmatrix} f_I\left(\boldsymbol{r}_\mathbf{1}\right) \\ f_I\left(\boldsymbol{r}_\mathbf{2}\right) \\ \vdots \\ f_I\left(\boldsymbol{r}_{\boldsymbol{Q}_S}\right) \end{bmatrix} = \mathcal{R} \begin{bmatrix} \hat{\mathbf{U}}_\mathbf{A} \\ \hat{\mathbf{U}}_\mathbf{B} \end{bmatrix}. \tag{2.66}$$

The elements of $\mathcal{R}$ do not depend on $\hat{\mathbf{U}}_\mathbf{A}$ or $\hat{\mathbf{U}}_\mathbf{B}$. We write $f_I = \mathcal{R}(U_A^h, U_B^h)$ to denote the application of the full-order recovery procedure along interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$. The interested reader is directed to Appendix A for additional details regarding the formation of the discrete recovery operator.

The application of the biased recovery schemes in 2D and 3D requires an abbreviated set of testing functions compared to the classical recovery case. The 2D version of the Modal ICB recovery operator is utilized for the novel GR-VI scheme in Chapter 3, so it is covered here. The biased recovery approach is not useful on simplex elements (in our experience, it always yields unstable schemes), so we only cover the quadrilateral case. As with full-order recovery, the recovery basis is a smooth polynomial basis over $\mathcal{U} = \Omega_A \cup \Omega_B$. The basis $\boldsymbol{\psi}$ is formed from a

hierarchical modal basis set, such as the Legendre polynomials:

$$U^{ICB}_{\{A/B\}}(r,s) = \sum_{n=0}^{K_R-1} \hat{W}^n_{\{A/B\}} \psi^n(r,s), \quad \boldsymbol{\psi} = \{\psi^n(r,s) = L^a(r)L^b(s), \quad (a,b) \in \{0,...,p_r\} \times \{0,...,p_s\}\}. \tag{2.67}$$

For the Modal ICB operator in 2D, $p_r = p+1$ and $p_s = p$. Thus, $K_R = (p+2)(p+1)$. The A-biased ICB reconstruction for an interface on a 2D quadrilateral mesh is constrained as follows:

$$\int_{\Omega_A} \left(U^{ICB}_A - U^h_A\right) \phi^k_A d\boldsymbol{x} = 0, \quad \forall \phi^k_m \in \boldsymbol{\phi_m}, \tag{2.68a}$$

$$\int_{\Omega_B} \left(U^{ICB}_A - U^h_B\right) L^k(s) d\boldsymbol{x} = 0, \quad \forall k \in \{0,1,...,p\}, \tag{2.68b}$$

where s is the tangential recovery coordinate and $L^k(s)$ is the degree-k Legendre polynomial evaluated at s. A similar approach holds for the B-biased reconstruction along a particular interface.

2.6.6 Derivative-Based Recovery

The recovery approach detailed so far (for both the full-order and biased recovery types) shall henceforth be referred to as the *inner-product based* implementation of recovery because of the appearance of the integrals in the linear system of recovery constraints (Eq. 2.62). It is the original implementation by van Leer & Nomura [102]. The inner-product based implementation requires (*i*) the formation of the recovery coordinate, (*ii*) the formation of the recovery basis, and (*iii*) the formation and inversion of a linear system for each element-element interface in the domain. While these operations are simple in the 1D case, they become cumbersome in the multidimensional case, and personal experience showed that implementation in an existing DG code is a painfully invasive procedure. In addition to being cumbersome to implement [52], this procedure can lead to an ill-conditioned discrete recovery operator $\boldsymbol{\mathcal{R}}$ on non-Cartesian meshes in the multidimensional case, which threatens scheme stability.

In response to the difficulties of the inner-product approach to recovery, we searched for an alternate definition of the discrete recovery operator, subject to two requirements. First, the alternate

definition must be free of the recovery basis and the inversion of the associated linear system in Eq. (2.62). Second, the new approach must replicate the accuracy of the typical recovery system on 1D meshes and 2D Cartesian meshes while being easy to generalize to multiple dimensions. These goals are achieved by defining the discrete recovery operator $\mathcal{R}$ based on derivative jumps along interfaces instead of Eq. (2.61). In this section, the reasoning behind the new approach is presented and the new, *derivative-based* recovery implementation is cast in a form that is applicable in any number of spatial dimensions. This new implementation is applicable to all types of recovery (full-order and biased).

Consider the full-order recovery operator in 1D with $p = 1$ ($K = 2$, $K_R = 4$) on a uniform mesh with element width h. Let Ω_B be the element bordering Ω_A to the right. Take the DG solution basis to be the Lagrange basis with nodes at the element endpoints, with $\hat{U}_e^0$ being the left endpoint solution and $\hat{U}_e^1$ being the right endpoint solution for a given Ω_e. The discrete recovery operator (formed via the inner-product implementation, Eq. 2.62) takes the following form:

$$\mathcal{R} = \frac{1}{12}\begin{bmatrix} 1 & 5 & 5 & 1 \end{bmatrix}. \tag{2.69}$$

Thus, the value of the recovered solution at the interface ($r = 0$) is:

$$f(0) = \frac{1}{12}(\hat{U}_A^0 + 5\hat{U}_A^1 + 5\hat{U}_B^0 + \hat{U}_B^1). \tag{2.70}$$

Consider that both elements contribute approximations for the solution and its first derivative at the interface, $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$:

$$\begin{bmatrix} U_A^h \\ \frac{\partial}{\partial x} U_A^h \end{bmatrix}_{x=x_\mathcal{I}} = \begin{bmatrix} \hat{U}_A^1 \\ \frac{1}{h}(\hat{U}_A^1 - \hat{U}_A^0) \end{bmatrix} \quad , \quad \begin{bmatrix} U_B^h \\ \frac{\partial}{\partial x} U_B^h \end{bmatrix}_{x=x_\mathcal{I}} = \begin{bmatrix} \hat{U}_B^0 \\ \frac{1}{h}(\hat{U}_B^1 - \hat{U}_B^0) \end{bmatrix}. \tag{2.71}$$

The recovered solution (Eq. 2.70) is recast in terms of the interface quantities of Eq. (2.71):

$$f(0) = \frac{1}{2}(U_A^h - \frac{h}{6}\frac{\partial}{\partial x} U_A^h)|_{x=x_\mathcal{I}} + \frac{1}{2}(U_B^h + \frac{h}{6}\frac{\partial}{\partial x} U_B^h)|_{x=x_\mathcal{I}}, \tag{2.72}$$

demonstrating that the recovered solution is set according to (*i*) the average of the solution traces $U^h_{\{A,B\}}$ at the interface and (*ii*) the jump in the first derivative. Thus, it is possible to form the interface approximation $f(0)$ by adding a derivative-based correction to the solution average at the interface. Eq. (2.72) holds regardless of the choice of basis functions, but it is specific to the case where $h_A = h_B = h$ and $p = 1$. In general, $h_A \neq h_B$.

Maintaining the assumption of a 1D mesh, suppose that Ω_B is either left or right of Ω_A. For both the full-order and biased recovery schemes (taking Ω_A to be the dominant element), the derivative-based recovery operator depends on *recovery weights* $\boldsymbol{C}$ and takes the following form for arbitrary solution order p:

$$W(0) = \left(C_0 U^h_A + (1 - C_0) U^h_B\right)|_{x=x_I} + \sum_{j=1}^{p} C_j \left(\frac{\partial^j}{\partial r^j} U^h_A - \frac{\partial^j}{\partial r^j} U^h_B\right)|_{x=x_I}, \tag{2.73}$$

where the derivatives are evaluated by direct differentiation of the DG basis functions and all quantities are evaluated at the interface. The derivatives are written in terms of the recovery coordinate r (Eq. 2.59), which points out of Ω_A into Ω_B. As the difference in the derivatives tends to zero, the interface approximation tends towards a linear combination of U^h_A and U^h_B. In cases where the derivative differences are nonzero, the recovery operator uses the jumps in the derivatives to form a correction to the interface approximation. This behavior is not surprising; the inner-product approach to recovery, by enforcing weak equivalence relations involving all DOFs of the neighboring pair of elements, considers the full modal behavior of U^h over $\Omega_A \cup \Omega_B$. The derivative-based approach achieves the same goal by considering not just the values of U^h_A and U^h_B at the interface, but also all of the nonzero spatial derivatives contained in the finite-dimensional solution space.

The recovery weights $\boldsymbol{C}$ are themselves functions of both the type of recovery (full-order or biased) and a mesh uniformity index, $Q = \frac{h_A - h_B}{h_A + h_B}$. The index Q goes to zero as $h_A \to h_B$ and approaches $Q = \pm 1$ as the difference $h_A - h_B$ grows. For all types of recovery, we built the inner-product based discrete recovery operator (using the linear system of Eq. 2.62) for a variety of mesh

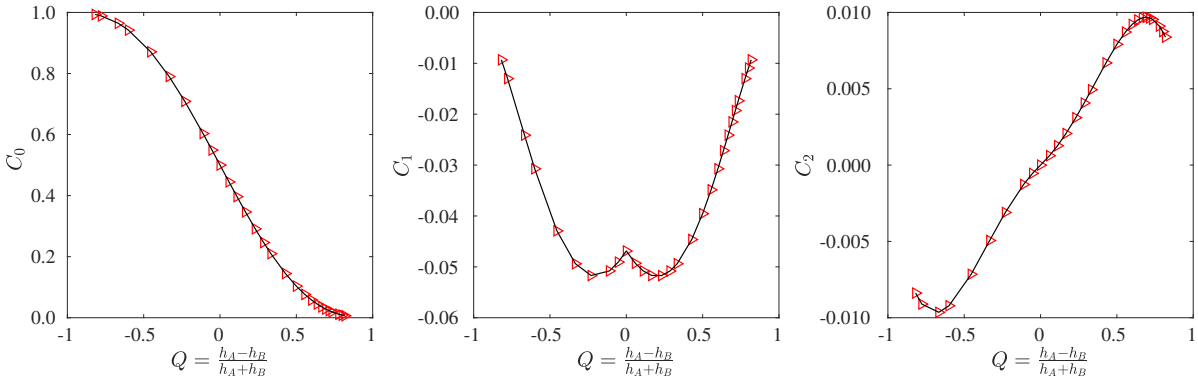


Figure 2.11: Recovery weights for classical recovery with $p = 2$. Coefficients extracted from the inner-product recovery operator are denoted by red triangles (▷) while the black line is the five-point interpolation used to build the derivative-based recovery operator.

uniformity indices, inspected the weights $\boldsymbol{C}$, and built interpolating polynomials,

$$C_j(Q) = \sum_{n=0}^{N-1} \hat{C}_j^n Q^n, \tag{2.74}$$

where the interpolation coefficients $\hat{C}_j^n$ depend on p and the particular type (full-order, Modal ICB, or Lagrange ICB) of recovery, with N being an appropriately high interpolation order. While an analytical form may exist for the weights, the resulting equations would likely be exceedingly complicated, so we chose the interpolation approach instead. The interpolations are described and tabulated in the appendices for implementation. Alternatively, Section 2.6.6.1 describes a way to circumvent the interpolation when implementing derivative-based recovery. Fig. 2.11 illustrates the weights C_0, C_1, and C_2 for the full-order recovery approach with $p = 2$. The recovery weights are predicted using a six-point interpolation; see Appendix B for the details.

By inspection of the derivative-based recovery implementation, we discovered that full-order recovery makes no use of even-order solution derivatives on evenly-spaced meshes. For example, observe that in Figure 2.11, $C_2 = 0$ when $Q = 0$. Similarly, for the $p = 4$ and $p = 5$ cases, $C_2 = C_4 = 0$ when $Q = 0$. This analysis unveiled another previously unreported feature of recovery: as $Q \to -1$, $C_0 \to 1$; consequently, as h_A becomes much smaller than h_B, the recovered function tends towards U_A^h instead of $\frac{1}{2}(U_A^h + U_B^h)$.

The derivative-based implementation can accommodate the biased recovery types (Modal and

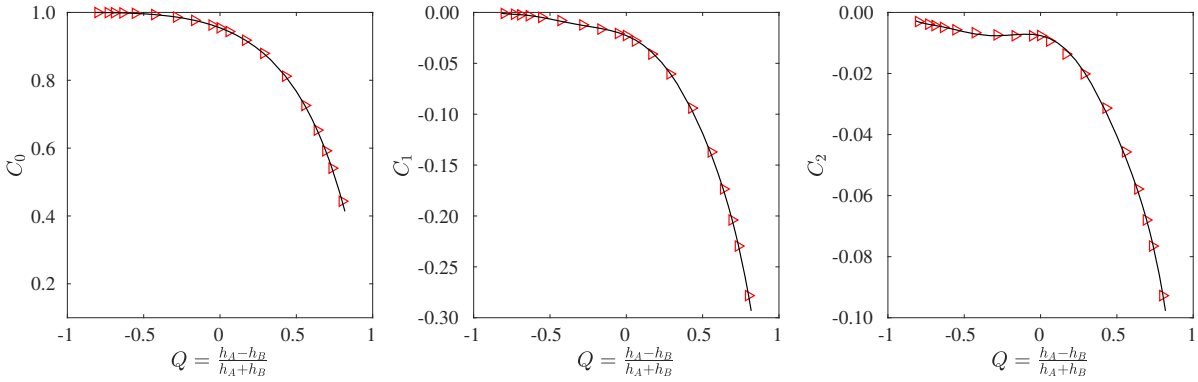


Figure 2.12: Recovery weights for A-biased Modal ICB reconstruction with $p = 2$. Weights extracted from the inner-product recovery operator are denoted by red triangles (▷) while the black line is the eight-point interpolation used to build the derivative-based recovery operator.

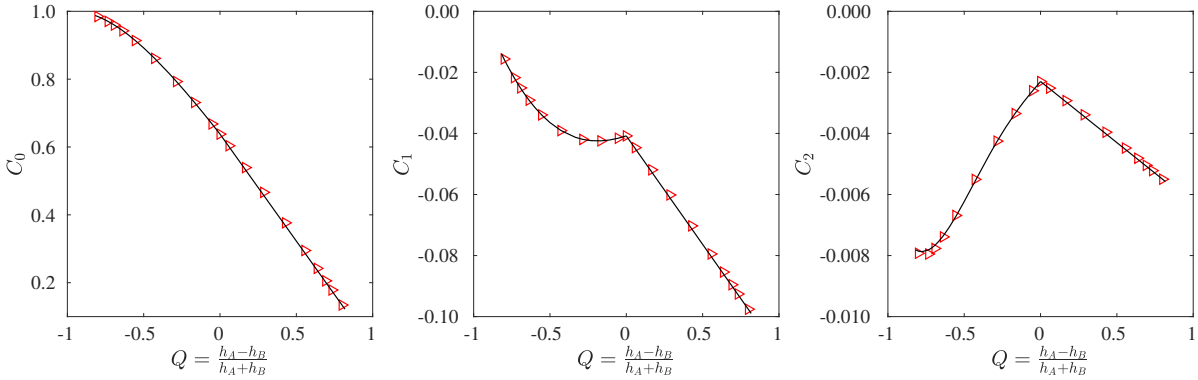


Figure 2.13: Recovery weights for A-biased Lagrange ICB reconstruction with $p = 2$. Weights extracted from the inner-product recovery operator are denoted by red triangles (▷) while the black line is the piecewise interpolation (five-point for $Q < 0$, two-point for $Q > 0$) used to build the derivative-based recovery operator.

Lagrange ICB) just as easily as the full-order recovery operator; the type of recovery governs the recovery weights $\boldsymbol{C}$. Figure 2.12 illustrates the weights of the Modal ICB operator for $p = 2$. In contrast to the full-order recovery operator, the weights are no longer symmetric about $Q = 0$, indicating a bias towards the dominant element in the ICB reconstruction. An eight-point interpolation is required to achieve a satisfactorily accurate $C_j(Q)$ polynomial for all cases, and the coefficients are listed in Appendix C. Figure 2.13 illustrates the weights of the Lagrange ICB operator for $p = 2$. Note the sharp corner at $Q = 0$, requiring a piecewise interpolation approach; the interpolation coefficients are given in Appendix D.

While the derivative-based implementation sheds some insight on the recovery concept, it is of

little practical use in the 1D case. Given a particular type of recovery, the inner-product form is easy to implement in 1D, and our experience suggests that the associated linear system (Eq. 2.62) is sufficiently well-conditioned to maintain stability even on nonuniform meshes. However, in the multidimensional case, the inner-product approach, in addition to being difficult to implement, can affect scheme stability via poor conditioning of the associated linear system. Thus, the derivative-based approach becomes useful in 2D and 3D as a way to exploit the accuracy of the recovery operation without the difficulties of the inner-product approach.

To extend the derivative-based implementation to the multidimensional case, the face-normal recovery coordinate r is needed. Consider a point $\boldsymbol{x}_{\mathcal{I}}$ along the shared interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$. Taking $\boldsymbol{x} = (x_1, x_2, x_3)$ to denote the traditional Cartesian coordinates of a point in space, we denote the face's normal vector (pointing from Ω_A to Ω_B) as $\mathbf{n} = (n_1, n_2, n_3)$ and the interface point as $\boldsymbol{x}_{\mathcal{I}} = (x_{\mathcal{I},1}, x_{\mathcal{I},2}, x_{\mathcal{I},3})$. Given some mesh width $\tilde{h}$ for the two elements, the face-normal recovery coordinate is defined as

$$r\left(x_1, x_2, x_3\right) = \frac{1}{\tilde{h}} \sum_{d=1}^{N_D} \left(x_d - x_{\mathcal{I},d}\right)\left(n_d\right). \tag{2.75}$$

With the face-normal recovery coordinate defined, the interface approximation takes the same form as the 1D case:

$$W|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}} = \left(C_0 U_A^h + (1 - C_0) U_B^h\right)|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}} + \sum_{j=1}^{p} C_j \left(\frac{\partial^j}{\partial r^j} U_A^h - \frac{\partial^j}{\partial r^j} U_B^h\right)|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}}. \tag{2.76}$$

In practice, the derivatives w.r.t. r in Eq. (2.76) are populated via the derivatives of the DG basis functions:

$$\frac{\partial^j}{\partial r^j} U_e^h = \sum_{k=0}^{K-1} \hat{U}_e^k \frac{\partial^j}{\partial r^j} \phi_e^k, \tag{2.77}$$

requiring the derivatives of the DG basis functions w.r.t r. Since the transformation from $\boldsymbol{x}$ to r is linear, each $\frac{\partial^j}{\partial r^j}\phi$ is obtained as follows:

$$\frac{\partial \phi}{\partial r} = \tilde{h} \sum_{a=1}^{N_D} n_a \frac{\partial \phi}{\partial x_a} \quad , \quad \frac{\partial^2 \phi}{\partial r^2} = \tilde{h}^2 \sum_{a=1}^{N_D} \sum_{b=1}^{N_D} n_a n_b \frac{\partial^2 \phi}{\partial x_a \partial x_b} \quad , \quad \frac{\partial^3 \phi}{\partial r^3} = \tilde{h}^3 \sum_{a=1}^{N_D} \sum_{b=1}^{N_D} \sum_{c=1}^{N_D} n_a n_b n_c \frac{\partial^3 \phi}{\partial x_a \partial x_b \partial x_c}, etc. \tag{2.78}$$

The mesh width $\tilde{h}$ in Eq. (2.75) and Eq. (2.78) should account for both the orientation of the interface and the geometry of each of the surrounding elements Ω_A and Ω_B. As an element's geometric Jacobian matrix,

$$\boldsymbol{J_e} = \left[\frac{\partial x_{row}}{\partial \xi_{col}}\right], \tag{2.79}$$

provides a measure of the mesh width in each direction, it should be included in the calculation of h_e for each element. Let L be the length of the reference element; for example, if the reference element is the bi-unit square, $L = 2$. For each element, the mesh width h_e (specific to the interface point $\boldsymbol{x}_{\mathcal{I}}$) is defined as

$$h_e = L\sqrt{l_1^2 + l_2^2}\,, \quad \text{where} \quad \begin{bmatrix} l_1 & l_2 \end{bmatrix} = \begin{bmatrix} n_1 & n_2 \end{bmatrix} \boldsymbol{J_e} \quad \text{if} \quad N_D = 2, \tag{2.80a}$$

$$h_e = L\sqrt{l_1^2 + l_2^2 + l_3^2}\,, \quad \text{where} \quad \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix} = \begin{bmatrix} n_1 & n_2 & n_3 \end{bmatrix} \boldsymbol{J_e} \quad \text{if} \quad N_D = 3, \tag{2.80b}$$

with the element's Jacobian matrix $\boldsymbol{J_e}$ evaluated at $\boldsymbol{x}_{\mathcal{I}}$. For a given point along the interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$, the associated mesh width is $\tilde{h} = \min(h_A, h_B)$. Other methods of measuring h may be acceptable but have not been tested.

2.6.6.1 Summary and Limitations of Derivative-Based Recovery

We summarize the application of the derivative-based form of recovery; it is applied whenever a solution approximation is needed at an interface quadrature point. For a given quadrature point on a given interface, the mesh widths h_A and h_B are calculated according to Eq. (2.80). Then, the mesh uniformity index $Q = \frac{h_A - h_B}{h_A + h_B}$ is plugged into the $\boldsymbol{C}$ interpolation (see Appendices A, B, and C) to yield the weights $\boldsymbol{C}$. Next, given the common mesh width $\tilde{h} = \min(h_A, h_B)$ and the weights $\boldsymbol{C}$, the derivatives of the basis functions are used to cast the interface approximation (Eq. 2.76) in matrix-vector form:

$$W|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}} = \left(C_0 U_A^h + (1 - C_0) U_B^h\right)|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}} + \sum_{j=1}^{p} C_j \left(\frac{\partial^j}{\partial r^j} U_A^h - \frac{\partial^j}{\partial r^j} U_B^h\right)|_{\boldsymbol{x}=\boldsymbol{x}_{\mathcal{I}}} = \boldsymbol{\mathcal{R}}^{\mathcal{DB}} \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{A}} \\ \hat{\mathbf{U}}_{\mathbf{B}} \end{bmatrix}, \tag{2.81}$$

such that the interface approximation W is immediately available at the interface through a matrix-vector multiplication. In practice, the derivative-based discrete recovery operator $\boldsymbol{\mathcal{R}}^{\mathcal{DB}}$ is calculated for each interface quadrature point and stored during initialization. Then, whenever a recovered solution is required in the formation of the DG residual, it is immediately available from a matrix-vector multiplication. The computational cost of this operation is discussed in Chapter 4.

Before moving on, we note that there is a way to bypass interpolation of the weights $\boldsymbol{C}$. Given h_A and h_B for an interface quadrature point in the multidimensional case, the weights $\boldsymbol{C}$ can be obtained as follows. First, set up a pair of 1D elements Ω_A^{1D} and Ω_B^{1D} with lengths h_A and h_B. Then, use the inner-product approach to build the corresponding discrete recovery operator $\boldsymbol{\mathcal{R}}^{\mathbf{1D}}$. With knowledge of the basis functions on Ω_A^{1D} and Ω_B^{1D}, the coefficients $\boldsymbol{C}$ are then extracted from $\boldsymbol{\mathcal{R}}^{\mathbf{1D}}$ and plugged into the multidimensional derivative-based recovery operator $\boldsymbol{\mathcal{R}}^{\mathcal{DB}}$ (Eq. 2.81).

So far, the derivative-based approach has only proven useful on 1D elements, 2D quadrilateral elements, and 3D hexahedral elements. On 2D simplex elements, we have been unable to replicate the performance of the inner-product based approach with the derivative-based implementation. This shortcoming may be due to how the element width $\tilde{h}$ is calculated. It may also be related to the lack of the full tensor product basis on the simplex element. Due to this limitation in the derivative-based implementation, the Recovery-assisted DG schemes of Chapter 5 are restricted to quadrilateral meshes in 2D and hexahedral meshes in 3D.

CHAPTER 3

The Interface Gradient Recovery Family

3.1 Chapter Overview

In response to the failure of the naive diffusion scheme described in Section 2.3.1, this chapter describes our attempts to construct attractive DG schemes for parabolic PDE (diffusion) problems. Outside of our work, there are established DG methods for handling diffusion problems in a stable manner; however, established DG methods for parabolic PDEs suffer from a trio of ailments. First, where the standard DG method for hyperbolic PDEs achieves order $2p+1$ convergence in either cell-average or functional error measurements, the modifications for parabolic PDEs generally achieve only order $2p$ convergence [40]. Second, when explicit time integration is used, as is common in unsteady flow physics problems, small timestep sizes are required for numerical stability, thus increasing the necessary number of timesteps to solve an unsteady problem. Third, the need to accurately approximate the solution gradient along element interfaces often results in schemes with non-compact stencils, i.e., stencils extending beyond nearest neighbors (see Figure 2.5). These ailments are not always simultaneously present; for instance, where the BR2 [8] and IP [3] methods are plagued by the first two ailments but maintain a compact stencil, the Local DG [26] (LDG) and Recovery DG [69] methods introduce non-compact stencils in the multidimensional case, thus increasing the computational costs and parallel communication requirements. Certain versions of Recovery DG can be implemented with a compact stencil for the Laplacian diffusion problem and achieve unrivaled orders of accuracy alongside relatively small spectral radii. Unfortunately, for

shear-diffusion problems, which model the viscous stress tensor of the compressible Navier-Stokes equations, the analysis of Lo [67] shows that Recovery DG requires a non-compact stencil for consistency. Additionally, our experimentation with the Recovery DG family showed that the compact schemes are unstable for $p > 2$ in the shear-diffusion case.

We formed the Interface Gradient Recovery (IGR) family of schemes to attempt to improve on previous DG schemes for discretization of diffusion problems with respect to the issues listed above. From conception, the goal of the family was to take advantage of the accuracy of the recovery operator without suffering the disadvantages of Lo and van Leer's [67] Recovery-based DG (RDG) methods, which represent the most direct application of the recovery concept in the DG framework. These disadvantages include a non-compact stencil for general 2D diffusion laws and difficulty maintaining high orders of accuracy under Dirichlet/Neumann boundary conditions. The IGR approach works around these shortcomings by applying the Recovery concept within the mixed formulation. This chapter will demonstrate that the usage of the mixed formulation allows the numerical scheme to benefit from the accuracy of the recovery operator while bypassing the differentiation of the recovered solution, which is actually the cause of the Recovery DG family's shortcomings for shear diffusion [67]. The new schemes are described and analyzed alongside the established BR2 [8] and Local DG [26] methods, which are the state-of-the-art in DG methods for diffusion problems.

3.1.1 Novelty and Articles

All of the proposed schemes in the Interface Gradient Recovery (IGR) family are new. The related High-Accuracy-Gradient (HAG) scheme introduced in this chapter is also new. The idea of combining the recovery operator with the mixed formulation had previously not been explored. We note that the term "Gradient Recovery" has been used outside our research to refer to certain post-processing strategies [28, 95] that have nothing to do with van Leer, Nomura, and Lo's [102, 67] recovery operator, which is the operator employed to form the IGR schemes.

The material of this chapter appears in one AIAA conference manuscript, one submitted JCP

article, and one in-preparation article:

- P. E. Johnson & E. Johnsen, *A New Family of Discontinuous Galerkin Schemes for Diffusion Problems*, AIAA Paper 2017-3444.
- P. E. Johnson & E. Johnsen, *The Compact Gradient Recovery Discontinuous Galerkin Method for Diffusion Problems*, JCP Manuscript Number JCOMP-D-19-00070, under review.
- P. E. Johnson & E. Johnsen, *A Simple, Optimally Convergent Replacement for the BR1 Discontinuous Galerkin Method with 2D Fourier Analysis*, in preparation.

3.1.2 Usage of Recovery

The schemes of this chapter employ the full-order and biased forms of the recovery operator. These two types of recovery are applied exclusively in the inner-product based implementation as opposed to the derivative-based approach of Section 2.6.6. For the 2D test cases, the recovery operators are implemented as detailed in Section 2.6.5.

3.2 The Mixed Formulation

The IGR schemes are built in the DG framework, so the overall setup is the same as described in Section 2.5; the complete spatial domain $\mathbf{\Omega}$ is partitioned into M non-overlapping elements, linked by interfaces. The numerical approximation U^h is a smooth polynomial over the interior of each element Ω_e described by K DOFs $\hat{\mathbf{U}}_\mathbf{e}$ and K basis functions $\boldsymbol{\phi_e}$. Consider an unsteady PDE system with a diffusive flux function $\mathcal{G}$,

$$\frac{\partial}{\partial t} U = \nabla \cdot \boldsymbol{\mathcal{G}}(U, \nabla U), \tag{3.1}$$

to be satisfied in the weak sense over each element Ω_m. The DG weak form for the general multi-dimensional case (3.2) is repeated here with the addition of an auxiliary variable, $\boldsymbol{\sigma}$:

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t} U^h d\boldsymbol{x} = \int_{\partial\Omega_m} \phi_m^{k-} (\tilde{\boldsymbol{\mathcal{G}}} \cdot \mathbf{n}_m^-) ds - \int_{\Omega_m} (\nabla \phi_m^k) \cdot (\boldsymbol{\mathcal{G}}(U_m^h, \boldsymbol{\sigma_m})) d\boldsymbol{x} \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \mathbf{\Omega}. \tag{3.2}$$

The term $\mathbf{n}_\mathbf{m}^-$ is the outward normal along the boundary of an element, and the integration coordinate s traverses the perimeter of the element. In Eq. (3.2), the *auxiliary variable* $\boldsymbol{\sigma}$ has been introduced as an approximation for the solution gradient. The population of the flux $\boldsymbol{\mathcal{G}}$ at the quadrature points requires a sufficiently accurate approximation for the solution gradient, hence the introduction of the auxiliary variable. The strategy of introducing the auxiliary variable is known as the mixed formulation; see Arnold et al. [4], Cockburn & Shu [26], or Bassi & Rebay [10] for additional in-depth explanation.

Like the approximate solution U^h, the auxiliary variable is constructed by multiplying coefficients against the DG basis functions:

$$\nabla U^h(\boldsymbol{x} \in \Omega_m) \approx \boldsymbol{\sigma}_m(\boldsymbol{x}) = \sum_{k=0}^{K-1} \phi_m^k(\boldsymbol{x})\, \hat{\boldsymbol{\sigma}}_m^k. \tag{3.3}$$

The auxiliary variable is an approximation for the gradient, which has one component for each spatial dimension, so while ϕ_m^k is a scalar, each coefficient $\hat{\boldsymbol{\sigma}}_m^k$ has one component per spatial dimension, per field variable. The auxiliary variable is used to define the gradient for the diffusive flux function $\boldsymbol{\mathcal{G}}(U, \nabla U)$; it is a finite-dimensional projection of ∇U^h, constrained to be indistinguishable from the gradient of the global DG solution U^h (i.e., $\nabla U^h - \boldsymbol{\sigma} = 0$) in the weak sense. As with the DG weak form, the enforcement of this condition takes advantage of an integration by parts:

$$\int_{\Omega_m} \phi_m^k \boldsymbol{\sigma}_m d\boldsymbol{x} = \int_{\partial\Omega_m} \phi_m^{k-}(\tilde{U} \cdot \mathbf{n}_m^-) ds - \int_{\Omega_m} U_m^h \nabla \phi_e^k d\boldsymbol{x} \quad \forall \phi_m^k \in \boldsymbol{\phi_m}. \tag{3.4}$$

Within the mixed formulation, the particular discretization scheme for the diffusive conservation law is completely described once the common interface solution $\tilde{U}$ (in Eq. 3.4) and common interface flux $\tilde{\boldsymbol{\mathcal{G}}}$ (in Eq. 3.2) are defined. Similar to the common interface flux, the common interface solution $\tilde{U}$ must take the same value for both elements sharing an interface (as opposed to U^h, which generally is discontinuous across interfaces). For a list of $\tilde{U}$ and $\tilde{\boldsymbol{\mathcal{G}}}$ choices corresponding to various DG schemes for diffusion, see Arnold et al.[4].

It is typical, though not necessary, to perform a second integration by parts step on Eq. (3.4).

Define the broken gradient, $\nabla^h U^h$, as follows:

$$\nabla^h U^h(\boldsymbol{x} \in \Omega_m) = \nabla^h U_m^h(\boldsymbol{x}) = \sum_{k=0}^{K-1} \hat{U}_m^k \nabla \phi_m^k, \tag{3.5}$$

where the ∇ operator refers to the gradient in physical space. The broken gradient $\nabla^h U$ is in general multivalued (discontinuous) along interfaces. Consider the following identity, exploited by Bassi et al. [8] in the formation of the BR2 scheme:

$$\int_{\Omega_e} U_m^h \nabla \phi_m^k d\boldsymbol{x} = \int_{\partial\Omega_m} \phi_m^{k-} U_m^{h-} \cdot \mathbf{n}_{\mathbf{m}}^- ds - \int_{\Omega_m} \phi_m^k \nabla^h U_m^h d\boldsymbol{x} \quad \forall \phi_m^k \in \boldsymbol{\phi_m}. \tag{3.6}$$

By substituting Eq. (3.6) into Eq. (3.4), we obtain the constraining equation for $\boldsymbol{\sigma_m}$ in terms of the broken gradient and an interface-based correction:

$$\int_{\Omega_e} \boldsymbol{\sigma_m} \phi_m^k d\boldsymbol{x} = \sum_{F=1}^{N_F} \int_{\mathcal{I}_{m,F}} \phi_m^{k-} (\tilde{U} - U_m^{h-}) \mathbf{n}_F^- ds + \int_{\Omega_m} \phi_m^k \nabla^h (U_m^h) d\boldsymbol{x} \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \tag{3.7}$$

where each $\mathcal{I}_{m,F}$ is one of the N_F interfaces constituting $\partial\Omega_m$ and $\mathbf{n}_F^-$ is the outward normal from Ω_m along the interface. Eq. (3.7) demonstrates that the difference $\boldsymbol{\sigma_m} - \nabla^h(U_m^h)$ depends on a penalty term along each interface that goes to zero as $\tilde{U}$ approaches U_m^h. This correction remedies the loss of information inherent in the differentiation of U_m^h by allowing exchange of information across interfaces. With regard to implementation, both Eq. (3.4) and Eq. (3.7) are acceptable.

The common interface flux is obtained as follows: a common interface gradient $\tilde{\boldsymbol{\sigma}}$ is determined for each quadrature point along a given interface, and that gradient is plugged into the diffusive flux law to yield the interface gradient: $\tilde{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}(\tilde{U}, \tilde{\boldsymbol{\sigma}})$. There are additional pieces of the mixed formulation that will be necessary before introducing the compact members of the IGR family. However, for the non-compact members of the IGR family, all necessary components of the mixed formulation are ready, so we will now demonstrate how the Recovery concept can be combined with the mixed formulation to form the majority of the IGR family.

3.3 The Non-Compact IGR Schemes

Two special operators are used to define the common interface solution $\tilde{U}$ and common interface gradient $\tilde{\boldsymbol{\sigma}}$; these operators are the full-order recovery operator $\mathcal{R}(U_A^h, U_B^h)$ and the biased recovery operator, denoted $\mathcal{B}_{A/B}(U_A^h, U_B^h)$, as defined in Section 2.6. In this chapter, we exclusively make use of the Modal ICB version of the biased recovery operator, excluding the Lagrange ICB reconstruction. The notation $\mathcal{B}_{[]}$ indicates the dominant element in the biased recovery operator: $\mathcal{B}_A$ indicates the A-biased reconstruction and $\mathcal{B}_B$ indicates the B-biased reconstruction. These operators are listed in Table 3.1. Table 3.2 lists the non-compact members of the IGR family and distinguishes them according to the strategies employed for calculating $\tilde{U}$ and $\tilde{\boldsymbol{\sigma}}$. Along an interface of an element Ω_e, y^+ indicates that the interface value for some quantity y is set by the interface trace from the neighbor element, while y^- indicates that the interface value for y is set from Ω_e. For a given interface, y_A and y_B denote the interface traces of the discontinuous polynomial approximations for the quantity y from the left and right elements, respectively. Following previous convention in DG methods, $\{\{y\}\} = \frac{1}{2}(y_A + y_B)$ indicates the average of the traces from the neighboring elements. The Local DG (LDG) method of Cockburn & Shu [26] (with one-sided fluxes, denoted LDG-OS) and the first scheme of Bassi & Rebay [10] (BR1) are also shown for comparison. The first seven IGR schemes, labelled GR-I through GR-VII, have non-compact stencils. Note that the recovery operators are employed for both $\tilde{U}$ and $\tilde{\boldsymbol{\sigma}}$; since $\boldsymbol{\sigma_m}$ for a given Ω_m uses the same polynomial form as U_m^h, the recovery operator built to approximate $\tilde{U}$ from U_A^h and U_B^h can be directly applied to approximate $\tilde{\boldsymbol{\sigma}}$ from $\boldsymbol{\sigma_A}$ and $\boldsymbol{\sigma_B}$. The family designation "Interface Gradient Recovery" is based on the fact that most of these schemes make some use of the Recovery concept when calculating

Table 3.1: Interface recovery operators in terms of input, output, and constraint count.

Operator	Input	Output	Constraints, Ω_A	Constraints, Ω_B
$\mathcal{R}(U_A, U_B)$	$\hat{\mathbf{U}}_A, \hat{\mathbf{U}}_B$	$\tilde{U}$	K	K
$\mathcal{B}_A(U_A, U_B)$	$\hat{\mathbf{U}}_A, \hat{\mathbf{U}}_B$	$\tilde{U}$	K	$(p+1)^{N_D-1}$
$\mathcal{B}_B(U_A, U_B)$	$\hat{\mathbf{U}}_A, \hat{\mathbf{U}}_B$	$\tilde{U}$	$(p+1)^{N_D-1}$	K

the common interface gradient.

We now explain our reasoning for the formation of these schemes. The goal of Eq. (3.7) is to form a gradient approximation, $\boldsymbol{\sigma}$, that is closer to the gradient of the underlying exact solution U than the broken gradient $\nabla^h U^h$. We hypothesized that by pushing the interface term $\tilde{U}$ towards the exact solution, we could optimize the accuracy of the $\boldsymbol{\sigma}$ variable. Of course, one does not, in general, have access to the exact solution U in the middle of a simulation. By intelligently combining information from the two elements neighboring an interface, the recovery operator represents the best attempt to recover the underlying, smooth, exact solution U along the interface. Thus, we hypothesized that by using the recovery operator to calculate the common interface term $\tilde{U}$ in the auxiliary weak form (Eq. 3.7), we could obtain DG schemes that are more accurate than the state-of-the-art for diffusion problems. By similar reasoning, the recovery operator has also been empoyed to calculate the interface gradient $\tilde{\boldsymbol{\sigma}}$ along each interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$ from the neighboring gradient approximations $\boldsymbol{\sigma_A}$ and $\boldsymbol{\sigma_B}$. As the combination of the Recovery concept with the mixed formulation had never before been explored, we decided to form many different IGR schemes, all using the full-order and biased recovery operators in slightly different manners, to identify the best way to combine the Recovery concept with the mixed formulation.

Table 3.2: Configurations of non-compact DG schemes for diffusion. Schemes are described in terms of how common quantities $\tilde{U}$ and $\tilde{\boldsymbol{\sigma}}$ are calculated along interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$. GR-I through GR-VII are the non-compact members of the IGR family.

Method	$\tilde{U}$	$\tilde{\sigma}$
GR-I	$\{\{U\}\}$	$\mathcal{R}(\sigma_A, \sigma_B)$
GR-II	$\mathcal{R}(U_A, U_B)$	$\mathcal{R}(\sigma_A, \sigma_B)$
GR-III	$\mathcal{R}(U_A, U_B)$	$\{\{\sigma\}\}$
GR-IV	U_A	$\mathcal{R}(\sigma_A, \sigma_B)$
GR-V	U^+	$\mathcal{R}(\sigma_A, \sigma_B)$
GR-VI	$\mathcal{B}_A(U_A, U_B)$	$\mathcal{B}_B(\sigma_A, \sigma_B)$
GR-VII	U_A	$\mathcal{B}_A(\sigma_A, \sigma_B)$
LDG-OS	U_A	σ_B
BR1	$\{\{U\}\}$	$\{\{\sigma\}\}$

Note that the GR-V scheme uses the exterior solution trace, U^+, to set the common solution along a given interface. Thus, two elements sharing an interface end up using different values of $\tilde{U}$ along the shared interface when solving for $\boldsymbol{\sigma}$. The scheme represents a test on the necessity of using a consistent value of $\tilde{U}$ along a given interface; it is an excellent example of what not to do. Two of the IGR schemes (GR-VI and GR-VII) make use of the biased version of the recovery operator; this practice reduces the flow of information compared to the full-order recovery operator. As the flow of information is reduced, the element Ω_e gains more authority over its update vector $\frac{d}{dt}\hat{\mathbf{U}}_\mathbf{e}$.

3.4 The Compact IGR Schemes

There are three compact schemes in the IGR family, bearing the general label *Compact Gradient Recovery*, abbreviated CGR. These schemes are the standard CGR scheme, the CGR-Light variant, and the CGR-Heavy variant. This section covers the CGR schemes and the High-Accuracy-Gradient (HAG) scheme, which is a largely experimental relative of the IGR family but not a member, strictly speaking.

The reason that the non-compact members of IGR (and the BR1 scheme, as noted by Bassi et al. [8]) are non-compact is the common interface gradient calculation. The general trend of these schemes is that the auxiliary variable $\boldsymbol{\sigma}$ is employed when calculating the common interface gradient. Consider five adjacent elements on a 1D geometry: Ω_{m-2}, Ω_{m-1}, Ω_m, Ω_{m+1}, and Ω_{m+2}. The auxiliary variable $\boldsymbol{\sigma}$ in a given element makes use of the DOFs in the immediate three-cell stencil: $\boldsymbol{\sigma}_{\boldsymbol{m-1}} = \boldsymbol{\mathcal{H}}(\hat{\mathbf{U}}_{\mathbf{m-2}}, \hat{\mathbf{U}}_{\mathbf{m-1}}, \hat{\mathbf{U}}_{\mathbf{m}})$, $\boldsymbol{\sigma}_{\boldsymbol{m}} = \boldsymbol{\mathcal{H}}(\hat{\mathbf{U}}_{\mathbf{m-1}}, \hat{\mathbf{U}}_{\mathbf{m}}, \hat{\mathbf{U}}_{\mathbf{m+1}})$, and $\boldsymbol{\sigma}_{\boldsymbol{m+1}} = \boldsymbol{\mathcal{H}}(\hat{\mathbf{U}}_{\mathbf{m}}, \hat{\mathbf{U}}_{\mathbf{m+1}}, \hat{\mathbf{U}}_{\mathbf{m+2}})$ where $\boldsymbol{\mathcal{H}}$ is the operation that obtains the auxiliary variable from the available DOFs. The interface flux $\tilde{\boldsymbol{\mathcal{G}}}$ on the left interface of Ω_m depends on $\{\boldsymbol{\sigma}_{\boldsymbol{m-1}}, \boldsymbol{\sigma}_{\boldsymbol{m}}\}$ while the flux on the right interface of Ω_m depends on $\{\boldsymbol{\sigma}_{\boldsymbol{m}}, \boldsymbol{\sigma}_{\boldsymbol{m+1}}\}$. Thus, because of the information required to form the auxiliary variable in each element, the update scheme for Ω_m depends on $\hat{\mathbf{U}}_{\mathbf{m-2}}$, $\hat{\mathbf{U}}_{\mathbf{m-1}}$, $\hat{\mathbf{U}}_{\mathbf{m}}$, $\hat{\mathbf{U}}_{\mathbf{m-1}}$, and $\hat{\mathbf{U}}_{\mathbf{m+2}}$. In other words, the update scheme is non-compact. For computational efficiency and minimal invasiveness in the implementation process, it is preferable that the diffusion scheme make use of a compact

stencil.

In the construction of the CGR and HAG schemes, an extra variable known as the *semi-connected gradient* is introduced to prevent the scheme from becoming non-compact. Consider an element Ω_e with N_F sides, such that $\partial\Omega_e = \cup_{F=1}^{N_F} \mathcal{I}_{e,F}$, where $\mathcal{I}_{e,F}$ is the F^{th} edge/face of Ω_e. There is one semi-connected gradient for each interface of each element; ultimately, these semi-connected gradients are not stored in memory, but they serve as crucial building blocks for a compact version of the mixed formulation. The semi-connected gradient, like the auxiliary polynomial, is built in the discontinuous polynomial space $\boldsymbol{\phi}$, such that for a given interface $\mathcal{I}_{e,F}$ of a given element Ω_e,

$$\boldsymbol{g_{e,F}}(\boldsymbol{x}) = \sum_{n=0}^{K-1} \phi_e^n(\boldsymbol{x}) \hat{\boldsymbol{g}}^{\boldsymbol{n}}{}_{e,F}. \tag{3.8}$$

The semi-connected gradient is constrained by a modified version of Eq. (3.7). Instead of allowing a correction $\tilde{U} - U_e^{h-}$ to be enforced along all N_F interfaces, the semi-connected gradient $\boldsymbol{g}_{e,F}$ only makes use of the jump term along $\mathcal{I}_{e,F}$:

$$\int_{\Omega_e} \boldsymbol{g_{e,F}} \phi_e^k d\boldsymbol{x} = \chi \int_{\mathcal{I}_{e,F}} \phi_e^{k-} (\tilde{U} - U_e^{h-}) \mathbf{n}_e^- ds + \int_{\Omega_e} \phi_e^k \nabla^h (U_e^h) d\boldsymbol{x} \quad \forall \phi_e^k \in \boldsymbol{\phi_e}, \tag{3.9}$$

where $\tilde{U}$ is the distribution of the common interface solution along $\mathcal{I}_{e,F}$. The variable $\boldsymbol{g_{e,F}}$ is called the semi-connected gradient because it includes information only from Ω_e and the element sharing interface $\mathcal{I}_{e,F}$ as opposed to including information from all neighbors of Ω_e (in which case it would be fully connected). The interface-based correction in Eq. (3.9) is often cast as a local lifting operator (see [8, 15, 4, 13, 82]).

The scalar χ is the jump parameter. Usually, it is taken larger than unity to compensate for the fact that Eq (3.9) includes an interface correction from only one of the N_F interfaces of Ω_e. Increasing the χ value amplifies the influence that the jump $\tilde{U} - U_e^{h-}$ exerts on the semi-connected gradient. In cases where this jump is large, the semi-connected gradient becomes large, which leads to large values of the diffusive interface flux $\tilde{\boldsymbol{\mathcal{G}}}$. The artificial inflation of the diffusive flux can be helpful for numerical stability, so we keep χ in the discretization as a tunable parameter.

We favor $\chi = 2$ for reasons that will be apparent after Fourier analysis.

For each non-boundary interface $\mathcal{I}$, there are two elements sharing the interface. Consider a non-boundary interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$. Each of the two elements (Ω_A and Ω_B) has one semi-connected gradient associated with the interface. In the CGR scheme, the recovery operator is applied to these two semi-connected gradient polynomials to calculate the common gradient, hence the name "Compact Gradient Recovery". If interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$ influences $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{F}_A}$ of Ω_A and $\boldsymbol{g}_{\boldsymbol{B},\boldsymbol{F}_B}$ of Ω_B, then $\tilde{\boldsymbol{\sigma}}_{\mathcal{I}} = \mathcal{R}(\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{F}_A}, \boldsymbol{g}_{\boldsymbol{B},\boldsymbol{F}_B})$ in the standard CGR scheme. The benefit of the semi-connected gradients is that since $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{F}_A}$ and $\boldsymbol{g}_{\boldsymbol{B},\boldsymbol{F}_B}$ depend only on $\hat{\mathbf{U}}_{\mathbf{A}}$ and $\hat{\mathbf{U}}_{\mathbf{B}}$, the stencil is compact. Thus, the extra complication of the semi-connected gradient is justified. Appendix F presents a sample implementation of the interface gradient calculation.

The precise designs of the different CGR schemes and the related HAG scheme are now described; the conventional BR2 scheme [8] also fits nicely into this framework. Within the mixed formulation, a compact scheme is distinguished by three features. The first is the common interface solution $\tilde{U}$ for the auxiliary weak form (Eq. 3.4). The second is the interface correction term $\mathrm{COR}(\chi, U^h)$ in the following constraint equation, generalized from Eq. (3.9) for the semi-connected gradients:

$$\int_{\Omega_e} \boldsymbol{g}_{\boldsymbol{e},\boldsymbol{F}} \phi_e^k d\boldsymbol{x} = \int_{\mathcal{I}_{e,F}} \phi_e^{k-} \mathrm{COR}(\chi, U^h) \mathbf{n}_F^- ds + \int_{\Omega_e} \phi_e^k \nabla^h (U_e^h) d\boldsymbol{x} \quad \forall \phi_e^k \in \boldsymbol{\phi}_{\boldsymbol{e}}. \tag{3.10}$$

The third feature is the calculation of the interface gradient from the semi-connected gradients along each interface. The variants of the CGR scheme, the new HAG scheme, and the conventional BR2 scheme [8] are described in Table 3.3. In addition to the standard CGR method, there exist two variants: CGR-Heavy and CGR-Light. The Heavy formulation makes more use of the recovery procedure than the original CGR method. The Light formulation minimizes the use of Recovery to simplify the implementation process. The HAG scheme is a counterpoint to CGR-Light: both schemes make minimal use of the recovery operator, but HAG employs Recovery for the common interface solution while CGR-Light employs Recovery for the common interface gradient. These different schemes allowed us to explore the effect of the recovery operator on different terms in the

mixed formulation. The naive scheme covered in Section 2.3.1 is also reported in Table 3.3; the notation U^- in the $\tilde{U}$ column indicates that each element uses its own interface solution trace for $\tilde{U}$ in Eq. 3.4, thereby ignoring the information of neighboring elements.

We remark that the description of BR2 given here is slightly different than the primal form described by Bassi et al. [8]. Where our description calculates an average gradient at the interface and feeds that average gradient $\tilde{\boldsymbol{\sigma}}$ to the flux law, the primal form of the BR2 scheme [8] calculates $\mathcal{G}$ along both sides of the interface using the semi-connected gradients, then takes the average of the two fluxes. The two BR2 implementations are equivalent in the case of a linear flux law.

3.5 Boundary conditions

For any IGR scheme (and the HAG scheme as well), the common interface solution along a Dirichlet boundary interface is set by the Dirichlet condition, thus $\tilde{U} = C_D$ is immediately available; however, the interface gradient $\tilde{\boldsymbol{\sigma}}$ must be approximated. We apply the approach of Bassi et al. [8]. Consider a boundary interface, $\mathcal{I}_D$, on which the Dirichlet condition is C_D. For the element Ω_B whose perimeter $\partial\Omega_B$ includes $\mathcal{I}_D$, we solve for $\boldsymbol{g}_{B,\mathcal{I}_D}$ with the correction term in Eq. (3.10) set to $COR(\chi, U^h) = \chi(C_D - U_B^{h-})$, where $\boldsymbol{g}_{B,\mathcal{I}_D}$ is the semi-connected gradient associated with the Dirichlet interface. Then, $\boldsymbol{g}_{B,\mathcal{I}_D}$ is applied to calculate $\mathcal{G}$ along $\mathcal{I}_D$. The relatively simple Neu-

Table 3.3: Common interface solution $\tilde{U}$, correction term COR (Eq. 3.10), and common interface gradient $\tilde{\boldsymbol{\sigma}}$ along face F_A of Ω_A, where $\mathcal{I}_{A,F_A} = \partial\Omega_A \cap \partial\Omega_B$, for different schemes. The interface is face F_B of Ω_B. $\{\{U\}\} = \frac{1}{2}(U_A^h + U_B^h)$ along the interface.

Scheme	$\tilde{U}(\mathcal{I}_{A,F_A})$	$\mathrm{COR}(\chi, U^h)$	$\tilde{\boldsymbol{\sigma}}(\mathcal{I}_{A,F_A})$
CGR	$\mathcal{R}(U_A^h, U_B^h)$	$\frac{\chi}{2}(U^+ - U^-)$	$\mathcal{R}(\boldsymbol{g}_{A,F_A}, \boldsymbol{g}_{B,F_B})$
CGR-Heavy	$\mathcal{R}(U_A^h, U_B^h)$	$\chi(\tilde{U} - U^-)$	$\mathcal{R}(\boldsymbol{g}_{A,F_A}, \boldsymbol{g}_{B,F_B})$
CGR-Light	$\{\{U^h\}\}$	$\frac{\chi}{2}(U^+ - U^-)$	$\mathcal{R}(\boldsymbol{g}_{A,F_A}, \boldsymbol{g}_{B,F_B})$
HAG	$\mathcal{R}(U_A^h, U_B^h)$	$\frac{\chi}{2}(U^+ - U^-)$	$\frac{1}{2}(\boldsymbol{g}_{A,F_A} + \boldsymbol{g}_{B,F_B})$
BR2	$\{\{U^h\}\}$	$\frac{\chi}{2}(U^+ - U^-)$	$\frac{1}{2}(\boldsymbol{g}_{A,F_A} + \boldsymbol{g}_{B,F_B})$
Naive	U^-	0	$\frac{1}{2}(\nabla^h U_A^h + \nabla^h U_B^h)$

mann case (where $\nabla U \cdot \mathbf{n}$ is given) is ignored in this work. The described boundary procedure is applicable for both the compact and non-compact versions of the IGR family.

3.6 Fourier Analysis, 1D Diffusion Equation

For Fourier analysis of the IGR family, we maintain the assumption of a p-uniform grid with constant mesh spacing h. The governing conservation law is the heat equation with diffusivity of unity:

$$\frac{\partial U}{\partial t} = \mu \frac{\partial^2 U}{\partial x^2}. \tag{3.11}$$

As with Section 2.4, the solution is assumed to be a Fourier mode of nondimensionalized wavenumber ω, and we restrict ourselves to the 1D case such that $K = p + 1$. However, the stencil for a non-compact scheme, in general, includes the middle element and the two elements on each side of it (for a total of five elements in the stencil). Thus, the update matrix is built as follows:

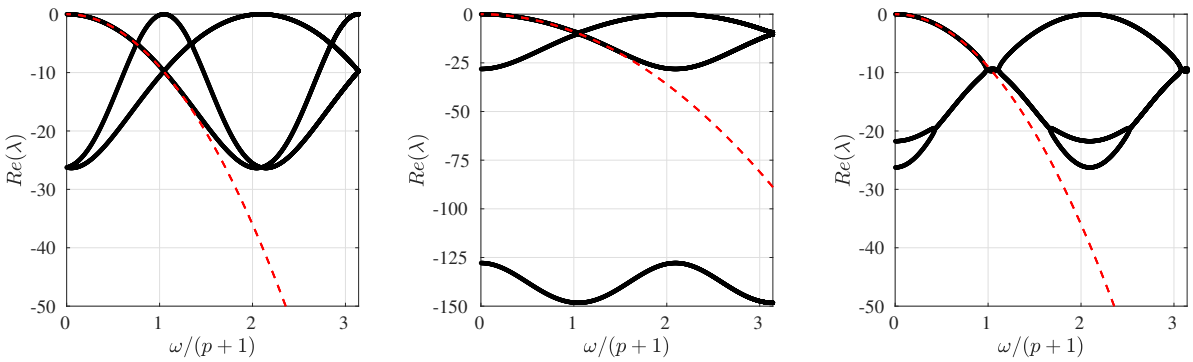
$$\frac{d}{dt}\hat{\mathbf{U}}_\mathbf{m} = \boldsymbol{D_{LL}}\hat{\mathbf{U}}_{\mathbf{m-2}} + \boldsymbol{D_L}\hat{\mathbf{U}}_{\mathbf{m-1}} + \boldsymbol{D_C}\hat{\mathbf{U}}_\mathbf{m} + \boldsymbol{D_R}\hat{\mathbf{U}}_{\mathbf{m+1}} + \boldsymbol{D_{RR}}\hat{\mathbf{U}}_{\mathbf{m+2}}, \tag{3.12a}$$

$$\frac{\mu}{h^2}\mathcal{A}(\omega) = \exp(-2i\omega)\boldsymbol{D_{LL}} + \exp(-i\omega)\boldsymbol{D_L} + \boldsymbol{D_C} + \exp(i\omega)\boldsymbol{D_R} + \exp(2i\omega)\boldsymbol{D_{RR}}. \tag{3.12b}$$

The DG update scheme is rewritten in terms of a $K \times K$ update matrix $\mathcal{A}(\omega)$:

$$\frac{d}{dt}\hat{\mathbf{U}}_m = \frac{\mu}{h^2}\mathcal{A}(\omega)\hat{\mathbf{U}}_m. \tag{3.13}$$

As described in Section 2.4, the scheme's order of accuracy, stability property, and spectral radius (at a given p) are calculated based on the eigenvalue spectrum of $\mathcal{A}(\omega)$. Figure 3.1 shows three example eigenvalue plots; these particular cases are the GR-II, LDG-OS, and CGR($\chi = 1$) schemes with $p = 2$. The real components of each of the $K = 3$ eigenvalues are plotted versus ω. For GR-II and CGR, the maximum magnitude of any of the eigenvalues of $\mathcal{A}$ is approximately $\rho_s = 26.5$; the spectral radius of the LDG-OS scheme is much higher. Fourier analysis has been performed for all members of the IGR family; the spectral radii and orders of accuracy are shown



(a) GR-II. (b) Local DG, One-Sided. (c) CGR ($\chi = 1$).

Figure 3.1: Eigenvalues of the update matrix $\mathcal{A}(\omega)$ for various diffusion schemes in the $p = 2$ case. The dotted red line is the exact eigenvalue for heat equation, $\lambda^{ex} = -\omega^2$.

in Tables 3.4 and 3.5 for $p \in \{1, 2, 3, 4, 5\}$. The properties of established DG methods are also listed for comparison.

Out of all the schemes analyzed, the GR-V scheme is the only one to exhibit inconsistency ($p = 1$) or instability ($p = 5$). This undesirable behavior confirms that it is poor form to let different elements use different $\tilde{U}$ values along a shared interface. The GR-IV and GR-VII methods show order $2p + 1$ accuracy; for both schemes, the consistent eigenvalue has a nonzero imaginary component that converges to zero at rate $2p+1$, so while the real component of λ^{con} is order $2p+2$

Table 3.4: Order of accuracy from Fourier analysis. **X** indicates an unstable scheme. For each CGR scheme (and the HAG scheme), order of accuracy is independent of χ.

Scheme	**p = 1**	**p = 2**	**p = 3**	**p = 4**	**p = 5**
GR-I	2	6	6	10	10
GR-II	4	8	10	14	16
GR-III	2	6	6	10	10
GR-IV	3	5	7	9	11
GR-V	0	2	2	4	**X**
GR-VI	4	6	8	10	12
GR-VII	3	5	7	9	11
CGR-Heavy	4	4	8	8	12
CGR	4	4	8	8	12
CGR-Light	2	4	6	8	10
HAG	4	4	8	8	12
LDG-OS	4	6	8	10	12
BR2	2	4	6	8	10
BR1	2	6	6	10	10

accurate, the imaginary component reduces the schemes to order $2p + 1$ accuracy.

There are certain scheme similarities and peculiarities worth commenting on. The GR-II scheme is by far the most accurate in the 1D case; it achieves order $3p + 1$ accuracy for odd p and order $3p + 2$ accuracy for even p. The reduced accuracy of the GR-I and GR-III schemes indicates that for a non-compact scheme, the recovery operator must be employed for both the common interface solution and the common interface gradient to achieve a performance advantage over BR1. The GR-VI method, by employing the biased recovery operator instead of the full-order recovery operator, is reduced to order $2p + 2$ accuracy. The BR1, HAG, CGR, and CGR-Heavy schemes all show irregular behavior in the order of accuracy, achieving either order $2p$ or order $2p + 2$ accuracy depending on whether p is even or odd. With $\chi = 1$, the BR2 and CGR-Light schemes perform nearly identically (the single point where they differ is the $p = 1$ case). The CGR-Heavy scheme and the standard CGR scheme perform identically in the 1D case and are especially attractive in the case of odd p; they offer order $2p + 2$ accuracy while having the smallest spectral radii of all schemes analyzed. The HAG scheme matches the CGR scheme in order of accuracy even though it does not make use of the recovery operator in the calculation of the common interface gradient. This performance suggests that for compact diffusion schemes, the

Table 3.5: Spectral radius, $\rho_s = |(\lambda)|_{max}$, rounded up to the nearest integer. **X** indicates an unstable scheme. HAG, BR2, and the CGR schemes are implemented with $\chi = 1$.

Scheme	**p = 1**	**p = 2**	**p = 3**	**p = 4**	**p = 5**
GR-I	13	60	171	381	739
GR-II	9	27	54	90	135
GR-III	7	60	171	381	739
GR-IV	7	60	171	381	739
GR-V	19	60	171	381	**X**
GR-VI	7	98	336	859	1832
GR-VII	24	114	377	940	1974
CGR-Heavy($\chi = 1$)	9	27	50	86	132
CGR($\chi = 1$)	9	27	50	86	132
CGR-Light($\chi = 1$)	12	60	171	381	739
HAG($\chi = 1$)	12	42	123	282	564
LDG-OS	36	149	439	1046	2143
BR2($\chi = 1$)	14	60	171	381	739
BR1	16	64	177	388	747

strategy used to calculate the interface gradient has little effect on scheme accuracy; instead, it is the strategy for calculating $\tilde{U}$ in the auxiliary weak form that governs scheme accuracy.

The peculiar stairstepping behavior in the orders of accuracy of CGR and HAG is connected to the traits of the recovery operator. In the analysis of the derivative-based implementation of the recovery operator (see Section 2.6.6 and Appendix B), it was discovered that on uniform meshes, the even-order recovery wieghts (C_2, C_4) are always zero. Thus, the recovery operator is ignorant of the 2nd and 4th spatial derivatives in the interface-normal direction; consequently, it is not shocking that the order of accuracy stays constant when moving from odd p to even p.

3.7 Numerical Test: 1D Heat Equation

The family of IGR schemes is evaluated by simulating the 1D diffusion equation (Eq. 2.2) with unity diffusivity and periodic boudary conditions. While the 1D diffusion equation is not of much practical interest, it serves as an effective filter to identify which IGR schemes are worth pursuing in more complicated implementations. For this particular test, the initial condition is $U(x, 0) = \sin(x)$ and the spatial domain is $\mathbf{\Omega} = [0, 2\pi]$ with periodic boundary conditions. A uniform grid is used in all cases, such that $\Delta x = \frac{2\pi}{M}$, where M is the number of elements. Each method is applied to simulate the system from $t = 0$ to $t_f = 2$ using the explicit 8^{th} order Runge-Kutta scheme of Prince and Dormand [86]. Two error metrics are used for the investigation: the L_2 error in cell averages, denoted E_{CA}, and the global L_2 error norm, denoted E_G:

$$E_{CA} = \sqrt{\frac{1}{M}\sum_{e=1}^{M}(\overline{U}_e^h - \overline{U}_e)^2} \qquad , \qquad E_G = \sqrt{\sum_{e=1}^{M}\int_{\Omega_e}(U_e^h - U)^2 dx}. \tag{3.14}$$

The cell averages of the DG and exact solutions are denoted $\overline{U}_e^h$ and $\overline{U}_e$, respectively, in each element Ω_e. All error measurements are obtained by comparing the DG solution at the final time, $U^h(x, 2)$, to the exact solution at the final time, $U(x, 2) = \exp(-2)\sin(x)$. Every scheme is run on a set of successively refined meshes for $p \in \{1, 2, 3\}$, and the error's order of convergence is

measured in both norms.

This test revealed that some of the IGR methods are sub-optimal in the global error norm, meaning that the order of convergence in E_G is less than $p+1$ for at least one choice of $p \in \{1, 2, 3\}$. These schemes are GR-I, GR-III, and GR-V. Additionally, the BR1 scheme is sub-optimal; this fact is known within the DG communuty and was confirmed by the test. Given the sub-optimal nature of BR1, it is not surprising that GR-I and GR-III, which bear close resemblance to BR1, were also sub-optimal. All of the remaining schemes from Table 3.4 are optimal in the global error norm, meaning that order $p + 1$ convergence is achieved. We refer to these remaining schemes as E_G-optimal to denote their optimal behavior in the global L_2 norm.

The convergence behavior in E_{CA} is now explored. Table 3.6 lists the number of elements M, the cell-average error E_{CA}, and the rate of convergence in E_{CA} for the grid refinement study of the GR-II scheme; the rate of convergence is measured between each successive pair of meshes for each discretization order p. From the tabulated data, it is evident that GR-II converges at rates four, eight, and ten for $p = 1$, $p = 2$, and $p = 3$, respectively. The convergence rates of all of the other E_G-optimal schemes were determined in the same manner and are listed in Table 3.7. For many schemes, the convergence rates in E_{CA} match the orders of accuracy given by Fourier analysis; in other cases, the two values are mismatched. For example, in the case of the GR-VII scheme, E_{CA} converges with order $2p + 1$ for $p \in \{1, 2\}$, but order $2p + 2$ for $p = 3$; alternatively, LDG-OS is order $2p + 2$ accurate from Fourier analysis, but sometimes only converges at rate $2p + 1$ in E_{CA}. Based on the results of this test, the only IGR schemes implemented in the multidimensional case are GR-II, GR-VI, the HAG scheme, and the CGR schemes. The rest of the schemes, while interesting, are not worth pursuing in the multidimensional case.

Table 3.6: GR-II Performance, 1D heat equation.

p	**M**	$\mathbf{E_{CA}}$	**Rate**	**p**	**M**	$\mathbf{E_{CA}}$	**Rate**	**p**	**M**	$\mathbf{E_{CA}}$	**Rate**
1	10	$4.58E-05$	–	2	10	$3.05E-09$	–	3	3	$6.46E-08$	–
1	20	$2.87E-06$	3.99	2	20	$1.16E-11$	8.04	3	4	$3.38E-09$	10.26
1	30	$5.68E-07$	4.00	2	30	$4.50E-13$	8.01	3	6	$5.92E-11$	9.97
1	40	$1.80E-07$	4.00	2	40	$5.00E-14$	7.64	3	8	$3.35E-12$	9.98
1	50	$7.37E-08$	4.00	2	50	$1.00E-14$	7.21	3	10	$3.60E-13$	10.00

3.7.1 Effect of the Jump Parameter (χ)

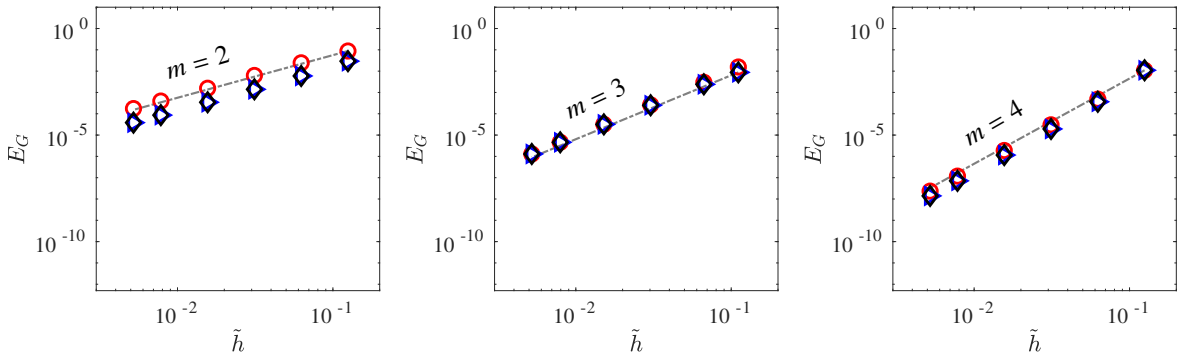
Supplementary analysis and testing was performed on the HAG, BR2, and CGR schemes with the jump parameter set to $\chi = 100$. Fourier analysis (data not shown) indicates that with $\chi = 100$, the BR2 scheme is always order $2p$ accurate. The HAG and CGR schemes are order $2p + 2$ accurate for odd p and order $2p$ accurate for even p. This behavior is the same as what was observed with $\chi = 1$ and $\chi = 2$. The heat equation test was repeated with the HAG, BR2, and CGR schemes, all implemented with $\chi = 100$. The error is reported in the global L_2 error and the cell-average error in Figure 3.2 and Figure 3.3, respectively. The results indicate that the convergence rates are unaffected by the substantial ampification of the jump parameter.

3.8 Fourier Analysis: 2D Shear Diffusion Equation

The IGR schemes were proposed with the ultimate purpose of discretizing the viscous terms in the compressible Navier-Stokes equations (Eq. 2.14); the viscous stress tensor $\boldsymbol{\tau}$ contains shear terms that distinguish it from the prototypical Laplacian diffusion equation. Thus, to predict scheme performance for the viscous terms of the compressible Navier-Stokes equations, we performed

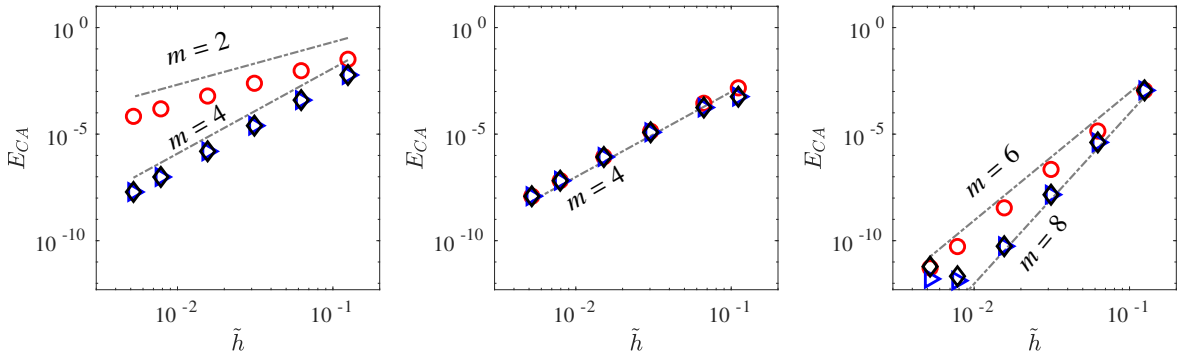
Table 3.7: Convergence rates in E_{CA} for 1D heat equation test. Schemes dependent on χ achieve the same rates of convergence regardless of χ, assuming $\chi \geq 1$. Only E_G-optimal schemes are included.

Method	**p**	**Rate**	**Method**	**p**	**Rate**	**Method**	**p**	**Rate**
GR-II	1	4	GR-II	2	8	GR-II	3	10
GR-IV	1	3	GR-IV	2	5	GR-IV	3	8
GR-VI	1	4	GR-VI	2	6	GR-VI	3	8
GR-VII	1	3	GR-VII	2	5	GR-VII	3	8
CGR-Heavy($\chi = 1$)	1	4	CGR-Heavy($\chi = 1$)	2	4	CGR-Heavy($\chi = 1$)	3	8
CGR-Heavy($\chi = 2$)	1	4	CGR-Heavy($\chi = 2$)	2	4	CGR-Heavy($\chi = 2$)	3	8
CGR($\chi = 1$)	1	4	CGR($\chi = 1$)	2	4	CGR($\chi = 1$)	3	8
CGR($\chi = 2$)	1	4	CGR($\chi = 2$)	2	4	CGR($\chi = 2$)	3	8
CGR-Light($\chi = 1$)	1	2	CGR-Light($\chi = 1$)	2	4	CGR-Light($\chi = 1$)	3	6
HAG($\chi = 1$)	1	4	HAG($\chi = 1$)	2	4	HAG($\chi = 1$)	3	8
BR2($\chi = 2$)	1	2	BR2($\chi = 2$)	2	4	BR2($\chi = 2$)	3	6
BR2($\chi = 1$)	1	2	BR2($\chi = 1$)	2	4	BR2($\chi = 1$)	3	6
LDG-OS	1	3	LDG-OS	2	6	LDG-OS	3	7



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.2: Convergence study in global L_2 error for 1D heat equation test with $\chi = 100$. Dashed gray lines: convergence rates m. Symbol Key: **BR2**:○, **CGR-Heavy**:▷, **HAG**:◇.



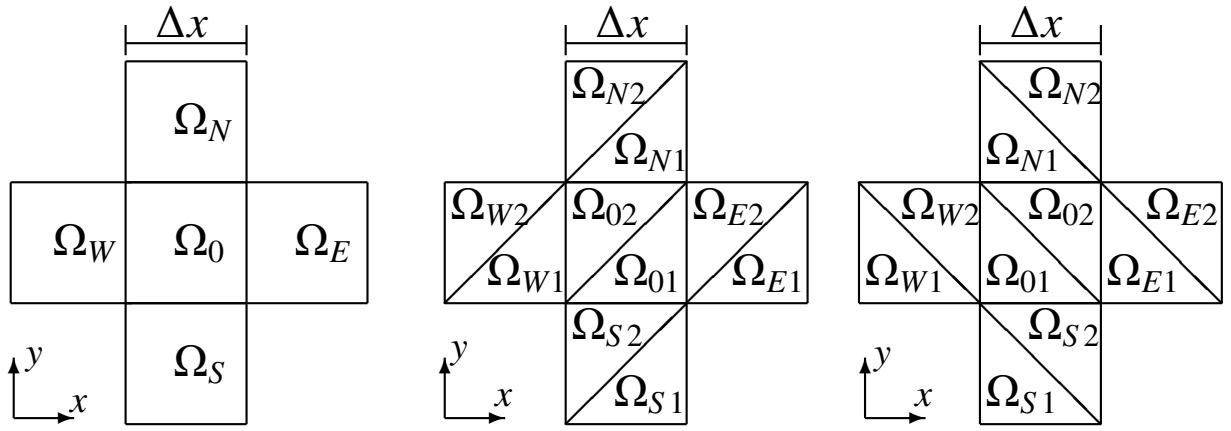
(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.3: Convergence study in cell-average error for 1D heat equation test with $\chi = 100$. Dashed gray lines: convergence rates m. Symbol Key: **BR2**:○, **CGR-Heavy**:▷, **HAG**:◇.

Fourier analysis with the 2D shear diffusion equation (Eq. 2.7) taken as the governing differential equation:

$$\frac{\partial U}{\partial t} = \nabla \cdot \left(\begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix} \right)^T = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + 2\theta \frac{\partial^2 U}{\partial x \partial y}, \tag{3.15}$$

where the diffusivity is set to unity and the shear factor θ is left as a parameter. Previous analysis of the RDG-1x++CO version of Recovery-based DG method [69] for the shear diffusion equation (Eq. 3.15) used $\theta = \frac{1}{2}$. However, we believe $\theta = \frac{1}{6}$ to be a more reasonable choice for modeling the behavior of the compressible Navier-Stokes equations. Consider the governing equation for the x component of momentum in the 2D compressible Navier-Stokes equations (Eq. 2.13), with



(a) Grid 1: Cartesian case. (b) Grid 2: Simplex case. (c) Grid 3: Alt. Simplex case.

Figure 3.4: Element orientations employed for Fourier analysis. In the Cartesian case (a), there are five elements in the stencil of Ω_0, each a square of side length Δx. In the structured simplex case (b, c), each element is part of a two-element square block of side length Δx for both the lower-left to upper-right diagonal (b) and the lower-right to upper-left diagonal (c) cases.

the derivatives of the viscous stress tensor expanded as velocity derivatives under the assumption of constant viscosity:

$$\frac{\partial}{\partial t}(\rho v_1) + \frac{\partial}{\partial x}(\rho v_1^2 + p) + \frac{\partial}{\partial y}(\rho v_1 v_2) = \mu(\frac{\partial^2 v_1}{\partial x^2} + \frac{\partial^2 v_1}{\partial y^2} + \frac{1}{3}\frac{\partial^2 v_1}{\partial x^2} + \frac{1}{3}\frac{\partial^2 v_2}{\partial x \partial y}). \tag{3.16}$$

This equation contains the scalar Laplacian of v_1, an extra $\frac{\partial^2 v_1}{\partial x^2}$ term, and the cross-derivative of v_2 (the y component of velocity). Assuming the gradients of v_1 and v_2 to be of similar magnitudes, the $\frac{1}{3}$ in front of the cross-derivative of v_2 corresponds to a choice of $\theta = \frac{1}{6}$ in the scalar shear diffusion equation (Eq. 3.15), hence our decision to apply $\theta = \frac{1}{6}$ in this analysis. We include results with $\theta = \frac{1}{2}$ as well for comparison to the RDG class of schemes [67].

Fourier analysis is performed on multiple geometries: first a Cartesian mesh, then two similar meshes of simplex (triangular) elements, as illustrated in Figure 3.4. The initial condition is taken as $U(x, y, 0) = \exp(i\omega' x + i\omega' y)$, where ω' is the dimensional wavenumber; a more thorough analysis would involve taking the wavenumbers in x and y to be independent parameters. In our experience, a scheme that is stable under the given initial condition is also stable for different choices of the x and y wavenumbers. The analysis procedures and results are reported separately for the Cartesian and simplex cases.

3.8.1 Cartesian Elements

The description of the Fourier analysis procedure is restricted to the compact case; the analysis of the non-compact schemes is a tedious but straightforward extension. We focus on a specific element, Ω_0, with the stencil shown in Fig. 3.4a. Defining a new variable $\beta = \omega' \Delta x$, where Δx is the element side length and β is the nondimensional wavenumber per element, the degrees of freedom in the stencil (Figure 3.4a) adhere to the form

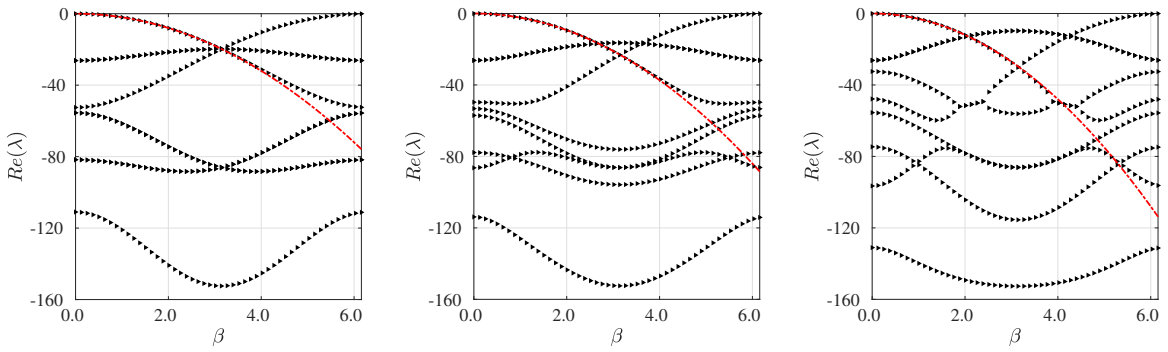
$$\begin{bmatrix} \hat{\mathbf{U}}_\mathbf{N} \\ \hat{\mathbf{U}}_\mathbf{S} \\ \hat{\mathbf{U}}_\mathbf{E} \\ \hat{\mathbf{U}}_\mathbf{W} \\ \hat{\mathbf{U}}_\mathbf{0} \end{bmatrix} = \begin{bmatrix} \exp(i\beta)\hat{\mathbf{U}}_\mathbf{0} \\ \exp(-i\beta)\hat{\mathbf{U}}_\mathbf{0} \\ \exp(i\beta)\hat{\mathbf{U}}_\mathbf{0} \\ \exp(-i\beta)\hat{\mathbf{U}}_\mathbf{0} \\ \hat{\mathbf{U}}_\mathbf{0} \end{bmatrix}. \tag{3.17}$$

The update scheme for Ω_0 is rewritten as a single matrix-vector multiplication,

$$\frac{d}{dt}\hat{\mathbf{U}}_\mathbf{0} = \frac{1}{(\Delta x)^2}\mathcal{A}(\theta, \beta)\hat{\mathbf{U}}_\mathbf{0}, \tag{3.18}$$

where the update matrix $\mathcal{A}$, in addition to depending on the particular DG method employed, depends on the nondimensional wavenumber β and the shear factor θ.

As with the 1D case, the eigenvalues of $\mathcal{A}$ inform us of the stability of the scheme, the spectral radius ρ_s, and the leading order of the error. For a given wavenumber, the update matrix still contains a single consistent eigenvalue. Sample eigenvalue spectra for the standard CGR scheme are displayed in Fig. 3.5. As β approaches zero, the consistent eigenvalue approximates the exact eigenvalue, $\lambda^{ex} = -2\beta^2(1+\theta)$, corresponding to the exact solution of Eq. (3.15), $U(x, y, t) = \exp(-2(1+\theta)(\omega')^2 t)\exp(i\omega' x + i\omega' y)$. Given the consistent eigenvalue, the strategy for determining the order of accuracy is the same as the 1D case (Section 3.6). Stability information is obtained for $p \leq 7$, and the order of accuracy is calculated for $p \leq 5$; for $p > 5$, our approach is unable to authoritatively identify the order of accuracy because the difference between the exact



(a) Eigenvalue spectrum, $\theta = 0$. (b) Eigenvalue spectrum, $\theta = \frac{1}{6}$. (c) Eigenvalue spectrum, $\theta = \frac{1}{2}$.

Figure 3.5: Eigenvalues of the CGR update scheme with $p = 2, \chi = 2$, and $\theta \in \{0, \frac{1}{6}, \frac{1}{2}\}$ on Grid 1 (Cartesian). The dashed red line in shows the exact eigenvalue, $\lambda^{ex}(\beta) = -2\beta^2(1+\theta)$. For all three cases, the spectral radius is $\rho_s = 153$ and the order of accuracy is 4.

and consistent eigenvalues drops below machine precision in the neighborhood of $\beta = 0$.

The results of Fourier analysis on the Cartesian grid are reported in three tables. Table 3.8 reports the spectral radii of the CGR and BR2 schemes; all of these schemes have compact stencils. Table 3.9 includes the spectral radii of Lo's [69] RDG-1x++CO scheme, copied from his thesis [67], alongside the spectral radii of the GR-II, GR-VI, BR1, and HAG schemes; the HAG scheme is the only compact approach on this table. For shear diffusion problems, RDG-1x++CO scheme is the best version of Recovery-based DG [69]. Note that the non-compact schemes reported in Table 3.9 do not make use of the χ parameter; recall that χ is part of the semi-connected gradient calculation, so it plays no role in the non-compact schemes. Table 3.10 reports the orders of accuracy of all schemes analyzed. The order of accuracy was found to be independent of θ and χ, assuming stability.

The spectral radius has been normalized by p^2; the tabulated quantity is thus $\frac{\rho_s}{p^2}$. The standard CGR scheme and the CGR-Heavy variant perform identically in terms of spectral radius, stability, and order of accuracy on Cartesian elements, so the Heavy scheme is omitted from Table 3.8. Our analysis reveals that the stability and the spectral radii of a scheme depend on the shear factor θ and the size of the jump parameter χ. The shear term θ destabilizes certain CGR configurations when $\chi = 1$; by increasing χ, stability is restored. However, the spectral radius grows with χ, so if explicit time integration is employed, it is advantageous to keep χ near the minimum stable value.

This relationship is illustrated in Fig. 3.6 for the standard CGR scheme with $p = 3$ and $p = 7$. We generally recommend the choice of $\chi = 2$ when implementing the CGR and HAG schemes in 2D and 3D. The commonly applied BR2 scheme is similar to the CGR scheme in that the spectral radius grows with both θ and χ; this effect is known within the DG community [82], so it is not a surprising trait in the CGR schemes. For a given choice of θ and χ, CGR systematically exhibits a smaller spectral radius than BR2, allowing larger timestep sizes in the case of explicit time integration; this advantage in spectral radius grows with increasing polynomial order p. The spectral radius of CGR-Light varies between that of standard CGR and BR2; this behavior is expected because CGR-Light is more similar to the BR2 discretization than the other CGR schemes. The HAG scheme (reported in Table 3.9) exhibits similar spectrall radii to the BR2 scheme.

Concerning accuracy, the CGR-Light variant and BR2 are always order $2p$ accurate. Standard

Table 3.8: **Cartesian Analysis, Grid 1:** Spectral radii of various schemes normalized by p^2 for the shear diffusion equation. The dash symbol indicates an unstable scheme. CGR-Light is abbreviated CGR-L. CGR-Heavy and standard CGR perform identically.

		$\chi = 1$			$\chi = 2$			$\chi = 4$		
p	θ	CGR	CGR-L	BR2	CGR	CGR-L	BR2	CGR	CGR-L	BR2
1	0	16.3	24.0	26.6	48.0	48.0	72.0	120.0	120.0	168.0
2	0	13.1	30.0	30.0	38.1	38.1	73.0	105.6	105.6	181.0
3	0	11.0	37.8	37.8	39.8	39.8	93.3	117.8	117.8	235.2
4	0	10.6	47.5	47.5	43.3	47.5	121.9	136.4	136.4	308.8
5	0	10.5	59.1	59.1	46.7	59.1	157.1	158.1	158.1	398.0
6	0	10.4	72.4	72.4	50.6	72.4	198.2	181.7	181.7	501.6
7	0	10.0	87.5	87.5	56.3	87.5	244.9	206.7	206.7	619.2
1	1/6	18.6	24.9	29.5	48.0	48.0	72.0	120.0	120.0	168.0
2	1/6	15.4	30.5	32.2	38.1	38.6	73.0	105.6	105.6	181.0
3	1/6	–	38.2	40.7	39.8	42.0	93.3	117.8	117.8	235.2
4	1/6	–	–	51.3	43.5	49.4	122.0	136.5	136.4	308.8
5	1/6	–	–	64.1	47.4	59.8	157.2	158.2	158.1	398.0
6	1/6	–	–	79.0	51.6	72.8	198.3	181.8	181.7	501.6
7	1/6	–	–	95.8	57.5	87.6	245.1	206.9	206.6	619.2
1	1/2	24.1	30.0	36.1	32.3	37.3	50.7	120.0	120.0	168.0
2	1/2	–	–	41.3	26.6	36.0	51.3	105.6	105.6	181.0
3	1/2	–	–	52.2	28.1	42.7	64.0	118.0	117.9	235.3
4	1/2	–	–	65.9	30.9	51.3	82.6	136.8	136.4	309.0
5	1/2	–	–	–	34.6	61.9	106.2	158.8	158.0	398.2
6	1/2	–	–	–	38.7	74.3	133.9	182.9	181.3	502.0
7	1/2	–	–	–	–	88.4	165.5	208.4	205.8	619.7

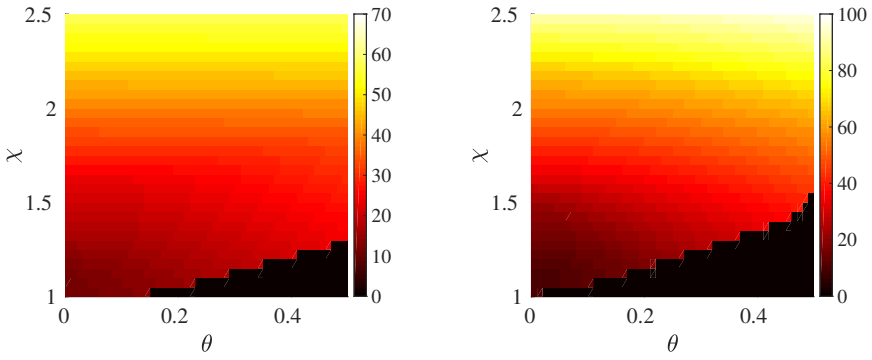
CGR and the Heavy variant are order $2p + 2$ accurate for odd p and order $2p$ accurate for even p. Thus, while maintaining a nearest-neighbors stencil, standard CGR and CGR-Heavy achieve equal to or greater order of accuracy than the BR2 scheme. We expected this behavior based on the 1D analysis, but an analysis in the 2D case, with the shear diffusion equation, was necessary to be sure. The HAG scheme also achieves order $2p + 2$ accuracy for odd p.

Table 3.9: **Cartesian Analysis, Grid 1:** Spectral radii of various schemes normalized by p^2 for the shear diffusion equation. The dash symbol indicates an unstable scheme.

		Noncompact				Compact		
p	θ	RDG-1x++CO	GR-II	GR-VI	BR1	HAG($\chi = 1$)	HAG($\chi = 2$)	HAG($\chi = 4$)
1	0	30.0	9.0	32.0	16.0	24.0	72.0	168.0
2	0	16.5	13.1	69.0	31.3	21.0	73.1	181.0
3	0	15.0	10.1	97.5	26.3	27.2	93.3	235.3
4	0	13.6	10.0	128.9	48.0	35.1	122.0	308.9
5	0	12.1	10.6	171.4	45.1	45.0	157.2	398.0
6	0	unknown	8.6	218.1	72.7	56.8	198.2	501.6
7	0	unknown	10.0	273.7	70.6	70.2	245.0	619.2
1	1/2	33.7	9.0	48.0	16.0	30.5	72.0	168.0
2	1/2	21.9	19.7	56.4	47.0	32.6	73.1	181.0
3	1/2	18.8	15.0	96.4	37.7	–	95.6	235.4
4	1/2	17.4	15.0	158.7	71.9	–	122.6	309.1
5	1/2	16.5	15.9	234.6	66.4	–	158.4	398.5
6	1/2	unknown	13.2	270.0	109.1	–	–	502.3
7	1/2	unknown	15.1	365.3	104.8	–	–	620.2

Table 3.10: **Cartesian Analysis, Grid 1:** Orders of accuracy, independent of θ.

$p =$	**1**	**2**	**3**	**4**	**5**
GR-II	4	8	10	14	16
GR-VI	4	6	8	10	12
CGR-Heavy	4	4	8	8	12
CGR	4	4	8	8	12
CGR-Light	2	4	6	8	10
HAG	4	4	8	8	12
BR1	2	6	6	10	10
BR2	2	4	6	8	10
RDG-1x++CO	4	8	10	14	16



(a) $p = 3$. (b) $p = 7$.

Figure 3.6: **Cartesian analysis, Grid 1:** Spectral radius normalized by p^2 for the CGR scheme. The region corresponding to unstable configurations is colored black. The stairstep pattern is a numerical artifact corresponding to limited resolution in (θ, χ) space.

3.8.2 Simplex Elements

Fourier analysis has also been conducted on simplex elements. Numerical testing on simplex elements showed the GR-II scheme to be unstable. Additionally, the GR-VI scheme's biased recovery operation cannot be applied on simplex elements, where a tensor product basis is unavailable. The Recovery DG method has been extended to simplex elements in a stable manner for shear diffusion [57], but the resulting scheme involves an abundance of extra constraints in the solution enhacement step of the scheme, which we prefer to avoid. Thus, the Fourier analysis on simplex elements is limited to the CGR schemes, the HAG scheme, and the BR2 scheme. The approach taken here is similar to that of Castonguay et al. [20] to analyze their energy-stable flux reconstruction scheme. Grid 2 (Fig. 3.4b) and Grid 3 (Fig. 3.4c) are used for the analysis. We consider two grids because, for $\theta \neq 0$, scheme performance depends on the orientation of the elements. We first focus on Grid 2, where the diagonal runs up to the right. Each element in Grid 2 belongs to a square pair (or block) of elements; and element block's DOF collection is denoted with two hats instead of one:

$$\hat{\hat{\mathbf{U}}}_{\mathbf{N}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{N1}} \\ \hat{\mathbf{U}}_{\mathbf{N2}} \end{bmatrix}, \hat{\hat{\mathbf{U}}}_{\mathbf{W}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{W1}} \\ \hat{\mathbf{U}}_{\mathbf{W2}} \end{bmatrix}, \hat{\hat{\mathbf{U}}}_{\mathbf{S}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{S1}} \\ \hat{\mathbf{U}}_{\mathbf{S2}} \end{bmatrix}, \hat{\hat{\mathbf{U}}}_{\mathbf{E}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{E1}} \\ \hat{\mathbf{U}}_{\mathbf{E2}} \end{bmatrix}, \hat{\hat{\mathbf{U}}}_{\mathbf{0}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{01}} \\ \hat{\mathbf{U}}_{\mathbf{02}} \end{bmatrix}. \tag{3.19}$$

As with the analysis on Cartesian elements, the initial condition is assumed to be a Fourier mode so that the DOFs in the five blocks obey the following form:

$$\begin{bmatrix} \hat{\hat{\mathbf{U}}}_{\mathbf{N}} \\ \hat{\hat{\mathbf{U}}}_{\mathbf{S}} \\ \hat{\hat{\mathbf{U}}}_{\mathbf{E}} \\ \hat{\hat{\mathbf{U}}}_{\mathbf{W}} \\ \hat{\hat{\mathbf{U}}}_{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \exp(i\beta)\hat{\hat{\mathbf{U}}}_{\mathbf{0}} \\ \exp(-i\beta)\hat{\hat{\mathbf{U}}}_{\mathbf{0}} \\ \exp(i\beta)\hat{\hat{\mathbf{U}}}_{\mathbf{0}} \\ \exp(-i\beta)\hat{\hat{\mathbf{U}}}_{\mathbf{0}} \\ \hat{\hat{\mathbf{U}}}_{\mathbf{0}} \end{bmatrix}. \tag{3.20}$$

Next, the update scheme for each element is expressed in matrix form. For example, the update scheme for element Ω_{01} depends on $\hat{\mathbf{U}}_{01}$, $\hat{\mathbf{U}}_{02}$, $\hat{\mathbf{U}}_{S2}$, and $\hat{\mathbf{U}}_{E2}$ as follows:

$$\frac{d}{dt}\hat{\mathbf{U}}_{01} = \mathcal{D}^{\mathbf{01}}_{\mathbf{01}}\hat{\mathbf{U}}_{01} + \mathcal{D}^{\mathbf{01}}_{\mathbf{02}}\hat{\mathbf{U}}_{02} + \mathcal{D}^{\mathbf{01}}_{\boldsymbol{S2}}\hat{\mathbf{U}}_{S2} + \mathcal{D}^{\mathbf{01}}_{\boldsymbol{E2}}\hat{\mathbf{U}}_{E2}, \tag{3.21}$$

where $\mathcal{D}^{\boldsymbol{m}}_{\boldsymbol{n}}$ contains the contribution of Ω_n to the residual of Ω_m. Fourier analysis requires formation of the following matrices: $\mathcal{D}^{\mathbf{01}}_{\mathbf{01}}$, $\mathcal{D}^{\mathbf{01}}_{\mathbf{02}}$, $\mathcal{D}^{\mathbf{01}}_{\boldsymbol{S2}}$, $\mathcal{D}^{\mathbf{01}}_{\boldsymbol{E1}}$, $\mathcal{D}^{\mathbf{02}}_{\mathbf{02}}$, $\mathcal{D}^{\mathbf{02}}_{\mathbf{01}}$, $\mathcal{D}^{\mathbf{02}}_{\boldsymbol{N1}}$, and $\mathcal{D}^{\mathbf{02}}_{\boldsymbol{W1}}$. Once these matrices are formed, they are combined with Eq. (3.20) and Eq. (3.19) to form the amplification matrix:

$$\frac{d}{dt}\begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{01}} \\ \hat{\mathbf{U}}_{\mathbf{02}} \end{bmatrix} = \frac{d}{dt}\hat{\hat{\mathbf{U}}}_0 = \mathcal{A}(\theta,\beta)\hat{\hat{\mathbf{U}}}_0. \tag{3.22}$$

The dimension of the amplification matrix is $2K \times 2K$, where $K = \frac{1}{2}(p+1)(p+2)$ is the DOF count per element, because it accounts for the update schemes of two elements. As with the Cartesian case, the eigenvalue spectrum of $\mathcal{A}$ is inspected to determine (*i*) whether or not the scheme is linearly stable, (*ii*) the scheme's order of accuracy, and (*iii*) the spectral radius. In the case of Grid 3 (Fig. 3.4c), the setup is similar but the amplification matrix $\mathcal{A}$ takes a different form due to the altered grid orientation.

Table 3.11 reports the calculated orders of accuracy. On simplex elements, the CGR schemes and the HAG scheme are reduced to order $2p$ accuracy, like the BR2 scheme. Assuming scheme

stability, the orders of accuracy are unaffected by the jump parameter χ and the shear factor θ.

Table 3.12 and Table 3.13 present the spectral radii for the CGR, HAG, and BR2 schemes on the two grid configurations tested. As with the Cartesian case, the stability of a scheme depends on the combination of p, θ, and χ. However, unlike the Cartesian case, the shear factor θ actually has a stabilizing effect on Grid 2. We suspected this trend to be related to the grid orientation, so Fourier analysis on Grid 3 was performed as well. For the scalar Laplacian case ($\theta = 0$), scheme performance is independent of the grid orientation, but in the shear diffusion cases ($\theta = \frac{1}{6}$ and $\theta = \frac{1}{2}$), the spectral radii are substantially increased on Grid 3 compared to Grid 2. The cause of this behavior is the $\tilde{\mathcal{G}} \cdot \mathbf{n}$ term in the DG weak form; when the shear term is present, the magnitude of $\tilde{\mathcal{G}} \cdot \mathbf{n}$ depends heavily on the orientation of an interface. On Grid 3, where the diagonal runs up to the left, the magnitude of the interface flux term is maximized, leading to larger overall spectral radii compared to Grid 2.

For sufficiently high p, some CGR schemes are unstable regardless of the size of the χ parameter, as verified by repeating the analysis up to $\chi = 6$ (data not shown). The standard CGR scheme remains stable through $p = 6$ with sufficiently large χ; for $p > 6$, the only stable CGR scheme is the CGR-Light variant. For a given value of χ and solution order p, the CGR schemes consistently yield smaller spectral radii than the BR2 scheme. Overall, the main thing that stands out is the lack of stable CGR schemes when χ is set to 1; higher χ values are necessary to stabilize the approach. Our conclusion is that on simplex elements, the standard CGR method should be applied with $\chi = 2$ and $p < 4$ to achieve stability while minimizing the spectral radius when solving the compressible Navier-Stokes equations. The CGR-Heavy scheme should be avoided on simplex meshes. Where robustness is a priority, the commonly applied BR2 scheme remains the

Table 3.11: **Simplex Analysis, Grids 2 and 3:** Orders of accuracy, regardless of θ.

$p =$	**1**	**2**	**3**	**4**	**5**
CGR-Heavy	2	4	6	8	10
CGR	2	4	6	8	10
CGR-Light	2	4	6	8	10
HAG	2	4	6	8	10
BR2	2	4	6	8	10

best choice.

Table 3.12: **Simplex Analysis, Grids 2 and 3, $\chi \leq 2$**: Spectral radii normalized by p^2 for the shear diffusion equation. The dash symbol indicates an unstable scheme. CGR-Heavy and CGR-Light are abbreviated CGR-H and CGR-L, respectively.

			$\chi = 1$					$\chi = 2$				
p	θ	Grid	CGR-H	CGR	CGR-L	HAG	BR2	CGR-H	CGR	CGR-L	HAG	BR2
1	0	2, 3	47.7	49.3	54.9	65.5	66.7	116.9	115.9	117.1	154.8	156.0
2	0	2, 3	–	35.8	43.0	50.8	50.9	84.6	88.1	87.2	133.2	131.8
3	0	2, 3	–	–	–	57.9	57.9	84.5	89.5	88.9	148.9	147.9
4	0	2, 3	–	–	–	70.9	70.3	95.4	103.9	103.0	182.9	180.8
5	0	2, 3	–	–	–	–	86.4	–	119.7	119.0	221.5	219.3
6	0	2, 3	–	–	–	–	105.1	–	141.2	140.0	270.9	267.6
7	0	2, 3	–	–	–	–	126.6	–	–	162.3	–	319.8
1	1/6	2	42.6	44.6	50.6	58.0	59.1	104.5	104.1	104.5	135.6	136.6
2	1/6	2	29.2	31.5	37.6	45.0	44.9	75.9	79.8	79.3	117.5	116.4
3	1/6	2	–	–	–	51.2	51.4	74.9	80.5	80.2	129.6	128.7
4	1/6	2	–	–	–	62.4	62.2	84.4	93.7	93.3	159.9	158.1
5	1/6	2	–	–	–	76.9	76.4	–	107.8	107.7	192.4	190.5
6	1/6	2	–	–	–	–	92.8	–	127.4	127.0	236.0	233.1
7	1/6	2	–	–	–	–	111.7	–	147.5	147.4	–	277.4
1	1/6	3	53.7	54.6	60.0	73.0	74.4	129.5	128.5	129.9	174.1	175.5
2	1/6	3	–	–	–	57.0	57.5	93.5	96.7	95.4	149.7	148.3
3	1/6	3	–	–	–	65.0	64.8	93.6	97.7	97.5	169.0	167.9
4	1/6	3	–	–	–	79.8	78.6	106.8	114.7	113.2	207.3	205.0
5	1/6	3	–	–	–	–	97.0	–	131.6	130.5	–	249.8
6	1/6	3	–	–	–	–	117.7	–	155.9	153.8	308.0	304.4
7	1/6	3	–	–	–	–	142.2	–	–	177.8	–	364.7
1	1/2	2	34.8	36.5	44.5	43.5	44.2	81.9	84.3	84.7	102.0	102.0
2	1/2	2	22.0	23.9	30.6	34.9	35.6	60.2	64.9	65.3	91.0	91.0
3	1/2	2	21.8	25.0	33.4	39.7	39.9	58.1	65.0	66.2	97.8	97.8
4	1/2	2	–	–	–	47.9	48.4	65.0	76.2	77.2	122.2	122.2
5	1/2	2	–	–	–	59.2	59.3	72.4	87.7	89.4	144.9	144.8
6	1/2	2	–	–	–	71.7	72.1	82.2	104.2	106.1	179.1	179.1
7	1/2	2	–	–	–	–	86.8	–	121.2	123.4	–	210.8
1	1/2	3	–	–	–	88.2	90.0	155.1	154.1	155.8	213.1	214.8
2	1/2	3	–	–	–	71.2	71.1	112.0	114.7	112.9	184.0	182.7
3	1/2	3	–	–	–	79.8	79.1	–	118.8	118.5	210.5	209.3
4	1/2	3	–	–	–	–	96.6	130.2	137.3	135.0	258.2	255.7
5	1/2	3	–	–	–	–	118.9	–	–	155.8	–	313.1
6	1/2	3	–	–	–	–	–	–	–	183.3	–	381.4
7	1/2	3	–	–	–	–	–	–	–	211.7	–	458.1

3.8.3 Remarks

On Cartesian meshes, multiple members of the IGR family provide superior performance compared to the BR1 and BR2 schemes, which serve as representatives of conventional DG approaches

Table 3.13: **Simplex Analysis, Grids 2 and 3, $\chi > 2$**: Spectral radii normalized by p^2 for the shear diffusion equation. The dash symbol indicates an unstable scheme. CGR-Heavy and CGR-Light are abbreviated CGR-H and CGR-L, respectively.

			$\chi = 3$					$\chi = 4$				
p	θ	Grid	CGR-H	CGR	CGR-L	HAG	BR2	CGR-H	CGR	CGR-L	HAG	BR2
1	0	2, 3	189.3	187.8	189.0	244.3	245.5	261.8	259.9	261.0	333.9	335.0
2	0	2, 3	140.9	145.8	144.6	216.2	214.9	197.6	203.9	202.5	299.4	298.1
3	0	2, 3	140.6	147.6	146.4	244.2	243.3	197.5	206.5	205.1	340.3	339.3
4	0	2, 3	159.6	172.3	170.6	300.9	298.9	224.8	241.3	239.4	420.1	418.2
5	0	2, 3	–	197.5	195.5	365.6	363.6	–	276.1	273.7	511.8	509.9
6	0	2, 3	–	234.1	231.6	446.9	443.9	–	327.5	324.5	625.8	623.1
7	0	2, 3	–	–	266.9	–	531.1	–	–	373.4	–	746.5
1	1/6	2	168.2	168.2	168.1	213.4	214.3	232.0	232.4	232.2	291.2	292.1
2	1/6	2	126.1	131.8	130.9	190.2	189.1	176.4	183.8	182.8	263.0	261.8
3	1/6	2	124.5	132.7	131.8	211.1	210.2	174.6	185.2	184.1	293.3	292.3
4	1/6	2	141.4	155.5	154.4	261.3	259.4	198.9	217.4	216.1	363.5	361.7
5	1/6	2	–	177.9	176.6	314.6	312.7	–	248.3	246.6	438.5	436.6
6	1/6	2	–	211.5	210.0	385.8	383.0	–	295.6	293.6	537.8	535.1
7	1/6	2	–	–	242.0	–	455.6	–	–	337.9	–	637.1
1	1/6	3	210.7	209.2	210.6	275.5	276.8	291.9	290.0	291.3	376.8	378.1
2	1/6	3	156.4	160.6	159.1	243.7	242.4	219.7	225.0	223.5	337.9	336.6
3	1/6	3	156.7	162.3	161.6	278.5	277.5	221.2	228.1	227.3	388.8	387.8
4	1/6	3	178.5	190.1	187.9	342.9	340.8	251.7	266.5	264.1	479.9	478.0
5	1/6	3	–	217.3	215.4	418.9	416.8	–	304.5	302.3	587.9	586.0
6	1/6	3	–	258.1	254.7	511.6	508.6	–	361.3	357.5	718.6	716.0
7	1/6	3	–	–	293.4	–	610.4	–	–	411.3	–	860.3
1	1/2	2	131.7	135.5	135.7	162.0	162.0	181.6	186.7	186.8	222.0	222.0
2	1/2	2	99.3	106.7	107.0	147.2	147.2	138.3	148.4	148.6	203.3	203.3
3	1/2	2	95.9	107.0	107.6	158.1	158.1	134.0	149.0	149.4	218.7	218.6
4	1/2	2	108.9	126.3	127.0	198.3	198.3	152.8	176.3	176.8	274.7	274.7
5	1/2	2	120.4	144.7	145.6	234.0	233.9	168.7	201.5	202.2	323.7	323.7
6	1/2	2	139.3	173.1	174.2	290.1	290.1	196.0	241.5	242.3	401.9	401.9
7	1/2	2	–	–	201.5	–	339.8	–	–	279.7	–	469.8
1	1/2	3	253.9	252.5	254.2	338.2	339.8	352.8	350.9	352.6	463.3	464.9
2	1/2	3	188.4	191.6	189.8	300.9	299.7	265.6	269.6	267.8	418.0	416.9
3	1/2	3	191.8	195.7	194.5	348.8	347.8	270.4	276.2	275.0	488.0	487.0
4	1/2	3	217.4	227.4	224.5	430.0	428.0	307.2	319.7	316.6	603.5	601.6
5	1/2	3	–	261.1	258.4	–	526.3	–	366.8	363.9	–	741.7
6	1/2	3	–	–	303.9	–	642.7	–	–	427.7	–	907.2
7	1/2	3	–	–	350.3	–	773.5	–	–	492.6	–	1093.2

for 2D diffusion problems. The GR-II scheme provides extraordinarily high orders of accuracy and small spectral radii. Among the compact schemes, the standard and heavy CGR schemes, as well as the related HAG scheme, achieve order $2p + 2$ accuracy for odd p. This behavior compares favorably to the order $2p$ accuracy exhibited by the BR2 scheme, which serves as a representative for established compact DG schemes for diffusion. Analysis across various shear factors θ and jump parameters χ demonstrated that it is sometimes necessary to increase χ past unity in order to maintain stability.

On simplex meshes, the CGR and HAG schemes lose much of their appeal. For both of the grid configurations tested, it is common to encounter unstable CGR schemes, even when the shear factor is set to $\theta = 0$. Additionally, the order of accuracy is fixed at $2p$. For a given jump parameter χ, the HAG and CGR schemes usually provide smaller spectral radii than the BR2 scheme. However, since BR2 is typically stable for $\chi = 1$ where the CGR schemes are not, it can be applied with smaller spectral radii. Based solely on spectral radius and order of accuracy, the CGR schemes do not appear worth implementing on simplex elements; we provide a set of numerical test problems in the next section to allow the reader to decide whether or not implementing these schemes is worth the trouble.

3.9 Numerical Tests: Linear 2D Diffusion

The IGR family's performance is now evaluated on a set of 2D diffusion problems. Moving forward from analysis, we decided to focus on the CGR scheme rather than the non-compact members of IGR, so the non-compact IGR schemes are omitted from some test cases. In distributed-memory architectures, a non-compact scheme requires more cross-processor communication than a compact scheme, which interferes with computational efficiency. Additionally, the non-compact schemes for diffusion are more invasive to code than the compact HAG and CGR schemes, as the interface gradient $\tilde{\boldsymbol{\sigma}}$ on a given element interface must account for information from all of the other interfaces on that element.

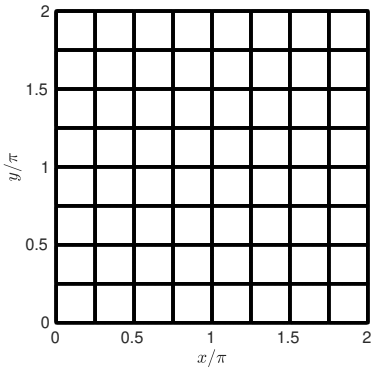
The test cases are summarized in Table 5.3 and involve both periodic and Dirichlet boundary

conditions; the new schemes are compared to the established BR1, BR2, and Local DG (LDG) schemes. All cases are unsteady, and the error is measured at the end of the simulation, $t = t_{final}$. For all test cases, the standard CGR method is compared to BR2, the gold standard for diffusion on a nearest-neighbor stencil; comparisons between the standard CGR approach and its Heavy/Simple variants are discussed in Section 3.10. The GR-II, GR-VI, and HAG schemes are also tested on select problems. The most important test cases in this section are cases 2 and 4, so the non-compact IGR members are tested alongside the CGR and BR2 schemes for those cases. The polynomial orders employed are $p = \{1, 2, 3\}$ and unless otherwise indicated, the jump parameter is $\chi = 2$ regardless of mesh geometry and polynomial order. With the exception of case 3B, the spatial domain for all cases is a $2\pi \times 2\pi$ square.

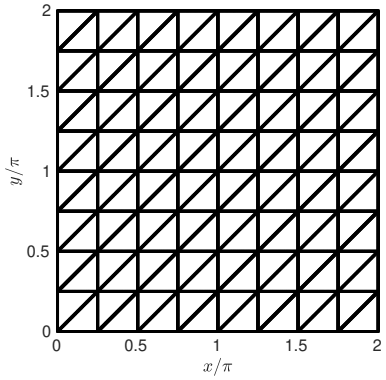
Each test case involves a mesh refinement study with $p \in \{1, 2, 3\}$ across a set of six mesh resolutions. Each mesh is characterized by directional resolution R, which is the number of elements in each direction. A Cartesian mesh has $M = R^2$ elements and a simplex mesh has $M = 2R^2$ elements. The mesh resolutions applied for both simplex and Cartesian meshes are listed in Table 3.15. Sample meshes with $R = 8$ elements per direction are shown in Fig. 3.7. In addition to Cartesian meshes, we demonstrate scheme performance on structured uniform and nonuniform simplex meshes as well as randomly perturbed quadrilateral elements. The uniform meshes partition the 2π square while the nonuniform simplex meshes (for case 3B) partition the quadrilateral defined by the following four vertices: $(x, y) = \{(0, 0), (2\pi, 0), (\frac{3}{2}\pi, \frac{3}{2}\pi), (\frac{1}{2}\pi, 2\pi)\}$.

Table 3.14: Summary of the test cases.

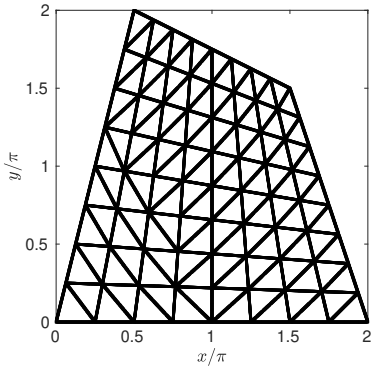
Test Case	**Governing Equation**	**Boundary Conditions**	**Mesh** (See Fig. 3.7)	**Schemes Tested**
1	scalar Laplacian diffusion	periodic	Cartesian	CGR, BR2
2A	scalar shear diffusion	periodic	Cartesian	CGR, BR2, LDG, BR1, GR-II, GR-VI
2B	scalar shear diffusion	Dirichlet	Cartesian	CGR, BR2
3A	scalar shear diffusion	periodic	uniform simplex	CGR, BR2
3B	scalar shear diffusion	Dirichlet	nonuniform simplex	CGR, BR2
4	scalar shear diffusion	periodic	r.p. quad	CGR, BR2, LDG, BR1, GR-II, GR-VI, HAG



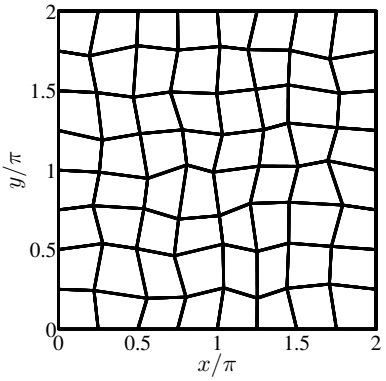
(a) $M = 8 \times 8$ Cartesian mesh with $\Delta x = 2\pi/8$ for cases $\{1, 2\}$.



(b) $M = (2\times8)\times8$ uniform simplex mesh for case $3A$.



(c) $M = (2 \times 8) \times 8$ nonuniform simplex mesh for case $3B$.



(d) $M = 8 \times 8$ randomly perturbed quad mesh for case 4.

Figure 3.7: Sample meshes with $R = 8$ elements per direction.

The governing PDE is the scalar diffusion equation:

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + 2\theta \frac{\partial^2 U}{\partial x \partial y} + \mathcal{S}(\boldsymbol{x}, t), \tag{3.23}$$

Table 3.15: Mesh resolutions in terms of directional resolution R and element count M.

p	Element Type	R	M
1	Cartesian	$\{8, 12, 16, 24, 32, 48\}$	$\{64, 144, 256, 576, 1024, 2304\}$
2	Cartesian	$\{5, 8, 10, 16, 21, 32\}$	$\{25, 64, 100, 256, 441, 1024\}$
3	Cartesian	$\{4, 6, 8, 12, 16, 24\}$	$\{16, 36, 64, 144, 256, 576\}$
1	Simplex	$\{8, 12, 16, 24, 32, 48\}$	$\{128, 288, 512, 1152, 2048, 4608\}$
2	Simplex	$\{5, 8, 10, 16, 21, 32\}$	$\{50, 128, 200, 512, 882, 2048\}$
3	Simplex	$\{4, 6, 8, 12, 16, 24\}$	$\{32, 72, 128, 288, 512, 1152\}$

where the shear factor θ is zero for case 1 (Laplacian diffusion) and $\theta = \frac{1}{6}$ for cases 2, 3 and 4 (shear diffusion). For the initial condition $U(x, y, 0) = 1 + \sin(x)\sin(y)$, the manufactured source term $\mathcal{S}(\boldsymbol{x}, t)$, determined exactly from Eq. (3.23), forces the solution to the exact, time-accurate solution, $U(x, y, t) = 1 + \exp(-2t)\sin(x)\sin(y)$. The final time is $t_{final} = 0.5$.

Once initialized, the solution is advanced in time using explicit Runge-Kutta schemes. The standard fourth-order, four-stage method (RK4) is used for $p < 3$ and the fifth-order, six-stage approach of Cash & Karp [19] for $p = 3$. The timestep is set to satisfy the stability constraints from Section 3.8,

$$\Delta t = \nu \frac{h_m^2}{\mu}, \qquad \nu = 0.9 \frac{C_{RK}}{\rho_s}, \tag{3.24}$$

where for stability $C_{RK} = 2.8$ (RK4) and $C_{RK} = 3.2$ (RK5). The number of timesteps and thus the wall time are directly proportional to the spectral radius.

Three different error norms are evaluated at $t = t_{final}$ in each simulation to illustrate different convergence behaviors. The first norm is the global L_2 error in the solution (E_G) and the second is the error in the cell-averages (E_{CA}):

$$E_G = \sqrt{\sum_{m=1}^{M} \int_{\Omega_m} (U_m^h(\boldsymbol{x}) - U(\boldsymbol{x}))^2 d\boldsymbol{x}} \quad , \quad E_{CA} = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (\overline{U}_m^h - \overline{U}_m)^2}, \tag{3.25}$$

where M is the number of elements. $\overline{U}_m^h$ and $\overline{U}_m$ are the averages of the numerical and exact solutions, respectively, over element Ω_m. The cell-average error is naturally suited to comparing the performance of DG to that of finite volume methods. The third norm is the functional error norm, which compares global integral quantities:

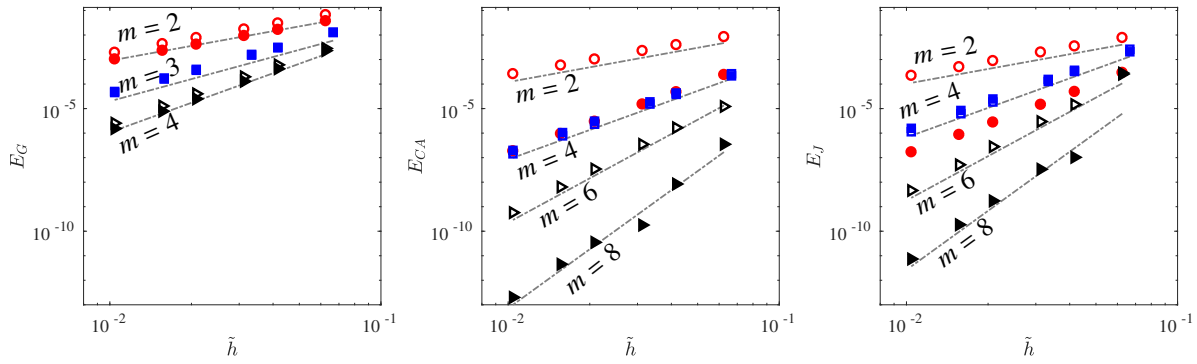
$$E_J = \left| \sum_{m=1}^{M} \int_{\Omega_m} \kappa(\boldsymbol{x}) U(\boldsymbol{x}) d\boldsymbol{x} - \sum_{m=1}^{M} \int_{\Omega_m} \kappa(\boldsymbol{x}) U^h(\boldsymbol{x}) d\boldsymbol{x} \right|, \tag{3.26}$$

where $\kappa(\boldsymbol{x})$ is some kernel function over $\boldsymbol{\Omega}$.

These error norms complement each other's downsides. The functional error norm E_J has the advantage of being easy to calculate when an exact solution is unavailable; instead, a resolved

calculation (on a fine mesh) can be substituted into the first integral in Eq. (3.26) to establish a reference value for $\int \kappa U d\boldsymbol{x}$. However, the functional error can give a deceptively favorable evaluation of a numerical solution depending on the kernel function, κ; for example, if $\kappa = 1$, the functional error depends only on the average of the global solution, and $E_J = 0$ becomes a likely event in this case, regardless of the upper bound of $U^h - U$. In this section, the kernel function is $\kappa(x, y) = \sin(x/4)\sin(3y/4) + \sin(xy/2)$ for all test cases. Assuming a kernel κ with a sufficient number of nonzero derivatives, we experimentally verified that the convergence behavior is nearly independent of κ. However, any given test problem has different requirements on κ to prevent deceptively high convergence rates. On the other hand, the global L_2 error norm consistently yields an authoritative quantification of how accurate the numerical solution is throughout the domain. However, its convergence rate is always limited to order $p + 1$ because of the finite-dimensional nature of U^h, regardless of the quality of the update scheme. Hence, while it is handy for showing the improvement of a DG scheme as p increases, the global L_2 norm is not an ideal tool for comparing numerical schemes at a given p. In contrast, the cell-average error E_{CA} exclusively inspects the behavior of a single mode in each element (the average), and this particular mode is included in the DG basis regardless of the DG solution order p. Hence, the cell-average error measures how effective the DG update scheme is without being affected by the finite-dimensional nature of U^h.

For all error norms, $Q_V = 10 \times 10$ quadrature points over each element regardless of p to calculate the errors. Convergence is measured with respect to the characteristic mesh width, $\tilde{h} = 1/\sqrt{nDOF}$, where $nDOF = M \times K$ is the number of degrees of freedom. In many applications, the quantity of interest depends on the solution gradient ∇U rather than the solution U itself, e.g., skin friction or heat flux. Errors in the solution gradient are discussed in Section 3.10.



(a) Global error. (b) Cell-average error. (c) Functional error.

Figure 3.8: Convergence study for test case 1 (scalar Laplacian diffusion with periodic boundary conditions on a Cartesian mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2 ; ($p1$:○, $p2$:□,$p3$:▷).

3.9.1 Test 1: Laplacian diffusion on a uniform Cartesian mesh with periodic boundary conditions

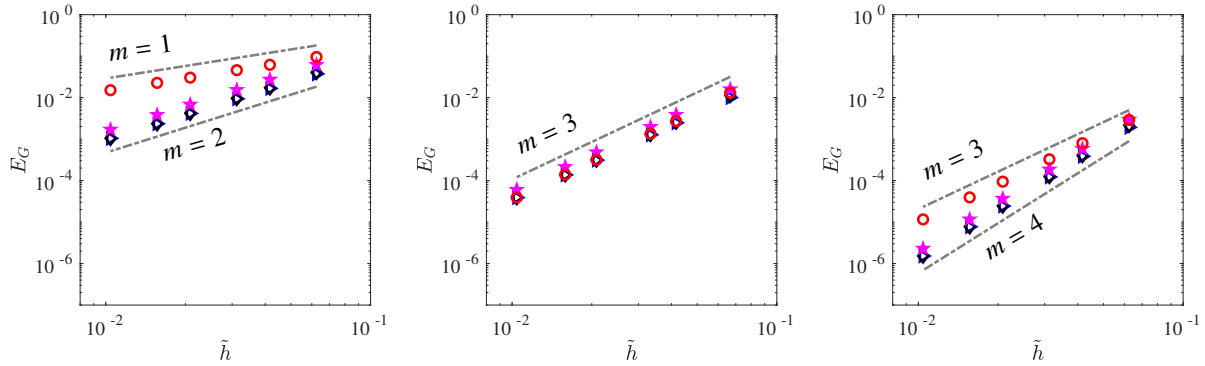
We first consider linear Laplacian diffusion ($\theta = 0$) on a uniform Cartesian mesh with periodic boundary conditions. Fig. 3.8 shows the global L_2 error, the cell-average error, and the functional error across all meshes for $p \in \{1, 2, 3\}$. In the global L_2 error, both CGR and BR2 achieve the optimal rate of convergence, $p + 1$. In the functional and cell-average error norms, where BR2 converges with order $2p$, CGR converges with order $2p + 2$ for odd p and order $2p$ for even p. In the $p = 2$ case, the errors of BR2 and CGR are nearly identical in all norms. For odd polynomial order p, the CGR scheme is superior in all three error norms. The $p = 1$ CGR discretization is particularly impressive, as it matches the $p = 2$ discretizations in the cell-average error for a given characteristic mesh width. The convergence rates in the cell-average and functional errors match the orders of accuracy obtained via Fourier analysis.

3.9.2 Test 2: Shear diffusion on a uniform Cartesian mesh with different boundary conditions

We consider shear diffusion with $\theta = \frac{1}{6}$ on a uniform Cartesian mesh with periodic (case 2A) and Dirichlet (case 2B) boundary conditions. The CGR scheme was engineered precisely for the purpose of maintaining stability and consistency on a compact stencil for shear diffusion problems while exploiting the recovery operator, so this test case serves as a crucial scheme benchmark. This test case, in addition to showing a demonstraton between the CGR and BR2 schemes, is employed to evaluate the performance of the non-compact GR-II and GR-VI schemes.

We discuss the non-compact schemes first. Figure 3.9 shows the errors of various non-compact DG schemes in the global L_2 error norm for this particular problem. The LDG, GR-II, and GR-VI schemes maintain optimal convergence in the global L_2 norm. In contrast, the BR1 scheme is sub-optimal for odd p; this deficiency is well known in the DG community. In the functional error norm, shown in Figure 3.10, the two IGR schemes (GR-II and GR-VI) consistently achieve order $2p + 2$ convergence. The BR1 scheme achieves order $2p$ convergence for odd p and order $2p + 2$ convergence for $p = 2$. The LDG scheme is typically order $2p + 2$ convergent, but it achieves 6^{th} order convergence in the functional error for $p = 1$. This exceptionally high convergence rate occurred for various choices of the kernel function. Figure 3.11 shows the cell-average errors of the non-compact schemes. The GR-II scheme consistently achieves order $2p + 2$ convergence; the other three schemes are less accurate, with GR-VI performing better than the established LDG and BR1 schemes. Overall, the GR-II and GR-VI schemes tend to produce more accurate results across the different error norms than the established BR1 and LDG methods. Thus, we see these non-compact approaches as viable improvements over the BR1 and LDG schemes on Cartesian meshes for developers who are willing to tolerate a non-compact stencil.

Figure 3.12 and Figure 3.13 show the global, cell-average, and functional error norms achieved with the CGR and BR2 approaches across all mesh resolutions for the scalar shear diffusion problem with periodic (case 2A) and Dirichlet (case 2B) boundary conditions; the latter are constant



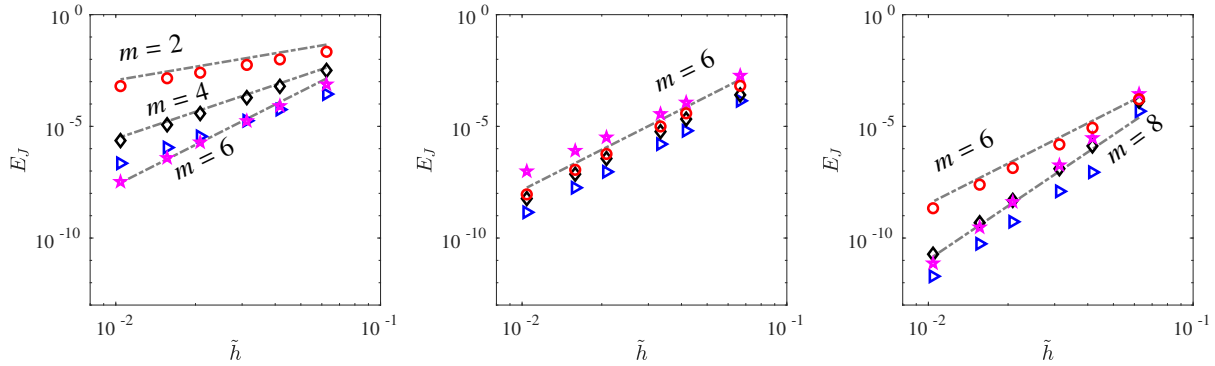
(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.9: Convergence study in global L_2 error for test case 2A (shear diffusion with periodic boundary conditions on a Cartesian mesh) using various non-compact schemes. Dashed gray lines: convergence rates m. Symbol Key: **BR1**:○, **GR-II**:▷, **LDG**:★, **GR-VI**:◇.

in space and time ($C_D = 1$) and are implemented based on the approach of Section 3.5. With both types of boundary conditions, the results for the global and functional error norms are very similar to those obtained in test 1 for Laplacian diffusion. However, there is a significant difference in the behavior of the cell-average error. Where the CGR scheme achieves order $2p + 2$ convergence in the cell-average error norm for $p = 3$ in case 1, it achieves only order $2p$ convergence in the present test. This change indicates that at least with respect to the cell averages the CGR update scheme can become less accurate in the presence of shear diffusion. Nevertheless, even though the convergence rates are the same, the error magnitude with CGR remains smaller than that with BR2 for $p = 3$. The presented results indicate that on Cartesian meshes CGR achieves order $2p + 2$ convergence in the functional error for odd p while maintaining a compact stencil. Additionally, no degradation in convergence rates is observed when using Dirichlet boundary conditions.

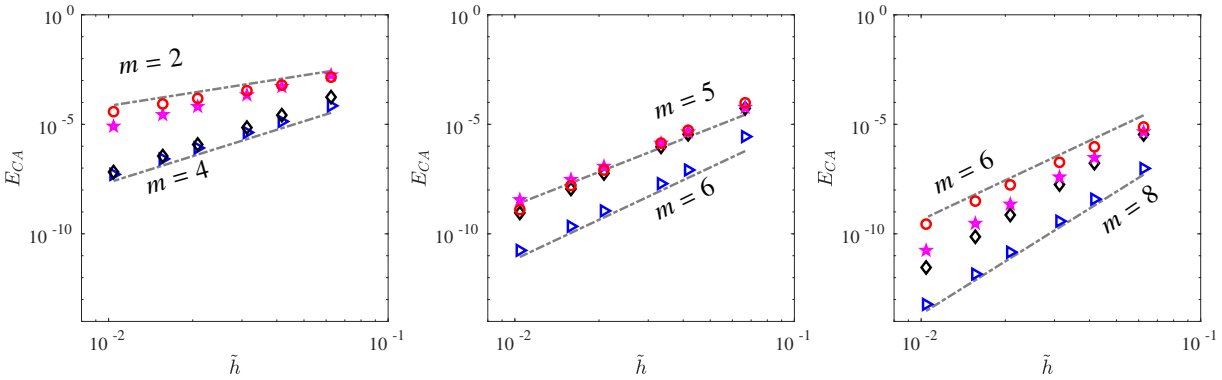
3.9.3 Test 3: Shear diffusion on a structured simplex mesh with different boundary conditions

To assess our approach on different meshes, we consider shear diffusion with $\theta = \frac{1}{6}$ on a structured simplex mesh with periodic (case 3A) and Dirichlet (case 3B) boundary conditions. For a given p, since there are fewer degrees of freedom (and fewer nonzero testing functions) per element,



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.10: Convergence study in functional error for test case 2A (shear diffusion with periodic boundary conditions on a Cartesian mesh) using various non-compact schemes. Dashed gray lines: convergence rates m. Symbol Key: **BR1**:○, **GR-II**:▷, **LDG**:★, **GR-VI**:◇.

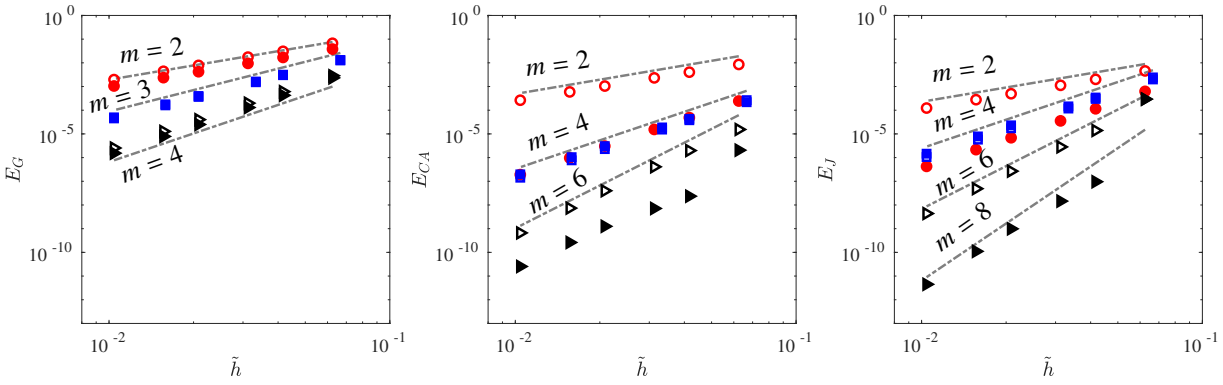


(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.11: Convergence study in cell-average error for test case 2A (shear diffusion with periodic boundary conditions on a Cartesian mesh) using various non-compact schemes. Dashed gray lines: convergence rates m. Symbol Key: **BR1**:○, **GR-II**:▷, **LDG**:★, **GR-VI**:◇.

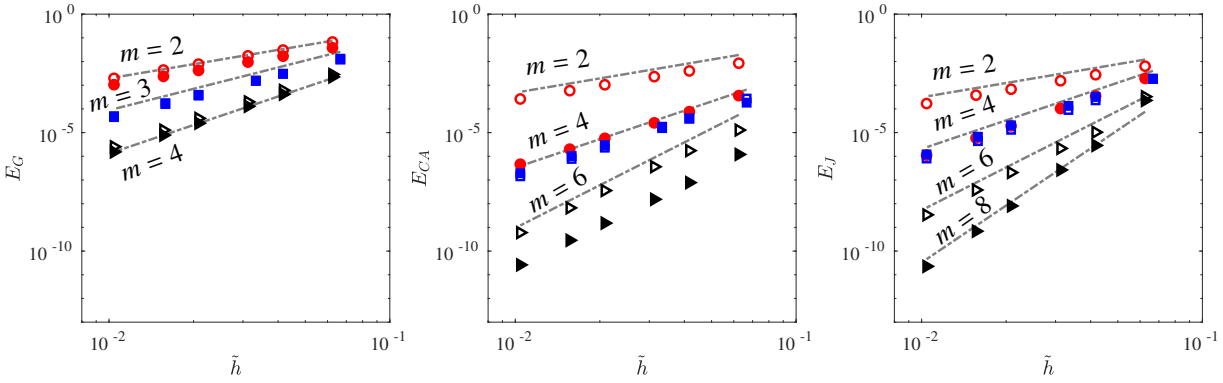
we expect some degradation in convergence rates when going from a Cartesian implementation to simplex elements. As with the tests on Cartesian elements, the BR2 and CGR schemes are implemented with $\chi = 2$. Though this value goes against the general recommendation of setting χ equal to the number of sides per element when implementing BR2, we witness no change in convergence behavior when the test is repeated with $\chi > 2$. The quadrature is still performed with $Q_V = (p+1)^2$ points on each element by taking the quadrature grid from an order p Cartesian element and collapsing the reference square to a right triangle [74].

We first consider the periodic problem (case 3A) on the simplex mesh shown in Fig. 3.7b;



(a) Global error. (b) Cell-average error. (c) Functional error.

Figure 3.12: Convergence study for test case 2A (scalar shear diffusion with periodic boundary conditions on a Cartesian mesh) with compact schemes. Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2; ($p1$:○, $p2$:□,$p3$:▷).



(a) Global error. (b) Cell-average error. (c) Functional error.

Figure 3.13: Convergence study for test case 2B (scalar shear diffusion with Dirichlet boundary conditions on a Cartesian mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2 ; ($p1$:○, $p2$:□,$p3$:▷).

the mesh refinement study for all p is presented in Fig. 3.14. As with the Cartesian case, both schemes achieve optimal convergence rates in the global L_2 error. The rate of convergence is $2p$ in the functional and cell-average errors regardless of the polynomial order p; while the convergence rates between CGR and BR2 are the same in the functional error, CGR consistently exhibits a smaller error. This experiment has been repeated with a mesh where the diagonals run up and to the left (instead of up and to the right), with similar results.

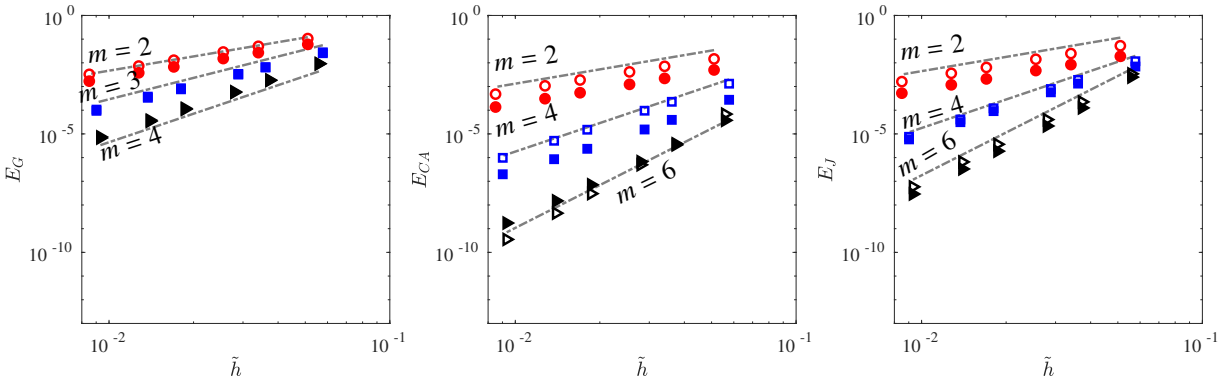


Figure 3.14: Convergence study for test case 3A (scalar shear diffusion with periodic boundary conditions on a uniform simplex mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2; ($p1$:○, $p2$:□,$p3$:▷).

For Dirichlet conditions (case 3B), we consider a more challenging mesh, the non-square polygon shown in Fig. 3.7c. For a given mesh resolution R, the domain is split into $M = R \times R$ quadrilateral elements. Then, each quadrilateral is split along the shorter of its two diagonals to produce the simplex mesh. The Dirichlet condition varies in time and space along the boundary, where the exact solution is prescribed. Thus, in addition to illustrating that the CGR scheme can handle the application of a Dirichlet boundary condition on a nonuniform mesh, this problem evaluates the scheme's ability to represent a non-constant Dirichlet distribution. As with the Cartesian case, the boundary procedure described in Section 3.5 is applied. The mesh refinement study is presented in Fig. 3.15. Both schemes experience degradation in the convergence rate of the cell-average error, but scheme performance is otherwise similar to the spatially periodic, uniform-mesh case. This result is encouraging, as it shows that the benefits of the Recovery concept in its full form can extend beyond uniform meshes.

3.9.4 Test 4: Shear diffusion on randomly perturbed quadrilateral mesh with periodic boundary conditions

We repeat test 2A (scalar shear diffusion with $\theta = \frac{1}{6}$, periodic boundary conditions) but alter the mesh. For a given mesh resolution ($M = R^2$), each interior element vertex in the Cartesian mesh is

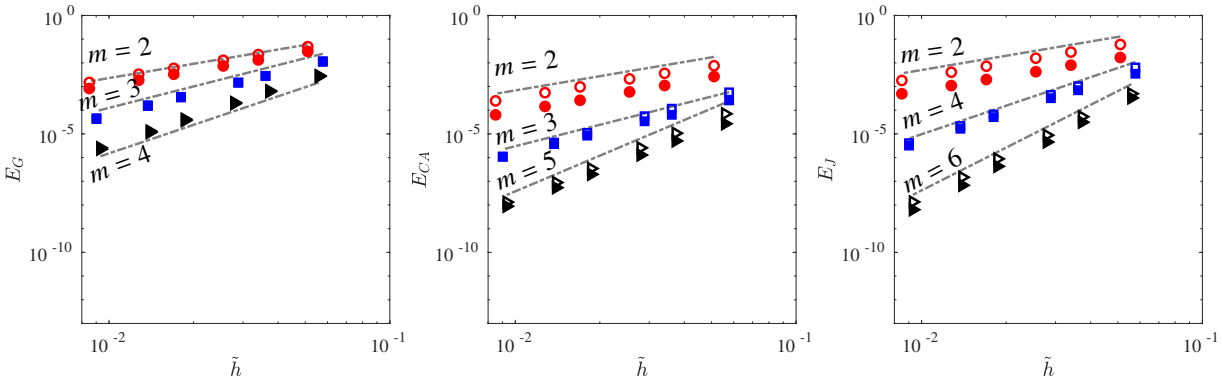
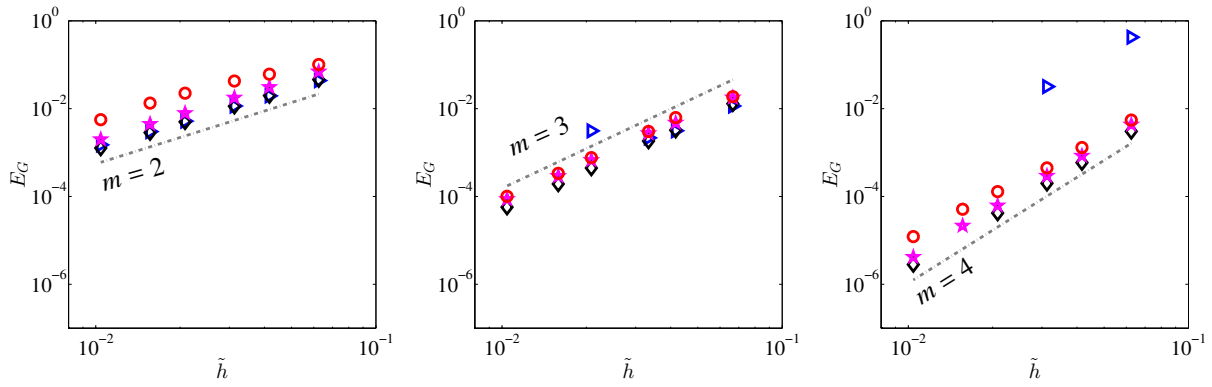


Figure 3.15: Convergence study for test case 3B (scalar shear diffusion with Dirichlet boundary conditions on a nonuniform simplex mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2; ($p1$:○, $p2$:□,$p3$:▷).

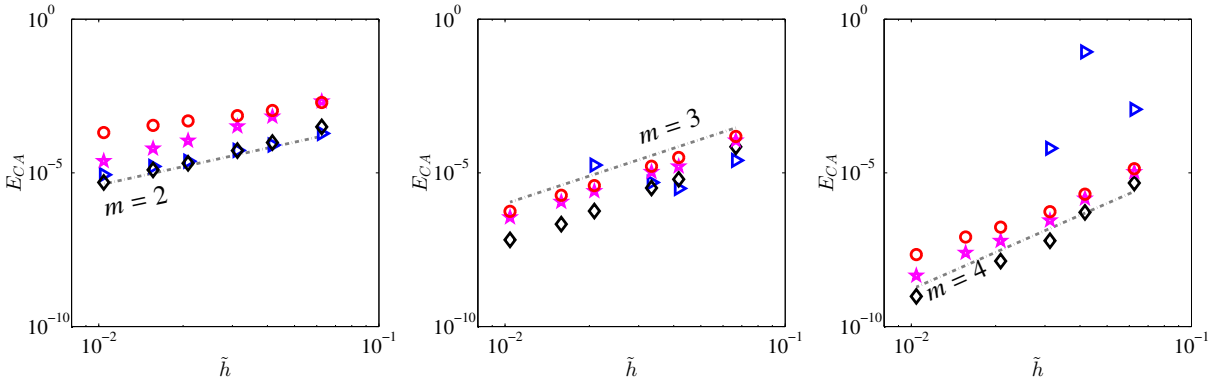
randomly perturbed up to $\frac{h}{4}$ in the x direction and $\frac{h}{4}$ in the y direction, where $h = \frac{2\pi}{R}$ is the element width. To account for the spatially varying Jacobian matrix, the quadrature rule is increased to $Q_V = (p+2)^2$ points per element volume and $Q_S = p+2$ points per interface. The goal of this test is to predict scheme performance on unstructured meshes, which are typical in design-related analysis. The schemes tested are the CGR-Heavy, standard CGR, HAG, GR-II, GR-VI, BR2, BR1, and LDG schemes. The compact HAG scheme achieved similar error to the CGR schemes in the 1D test case and has been applied here as a test of robustness; the CGR-Heavy scheme is included to determine whether it performs the same as the standard CGR approach on a non-Cartesian quadrilateral mesh.

We first discuss the non-compact schemes. Figure 3.16 and Figure 3.17 plot the global L_2 and cell-average errors, respectively. Note that some data points are missing, for example the GR-VI error on the second-finest mesh in the $p = 3$ case. These missing entries correspond to astronomically large errors in cases where the schemes were revealed to be unstable. The GR-II and GR-VI schemes cannot be trusted to maintain stability on a non-Cartesian mesh. For our particular mesh setup, GR-II is unstable for $p > 1$ and GR-VI is unstable for $p > 2$, while the established LDG and BR1 schemes maintained stability. Clearly, stability on the Cartesian mesh (guaranteed by Fourier analysis) does not guarantee stability on a mesh of perturbed quadrilateral meshes. When stable,



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.16: Convergence study in global L_2 error for test case 4 (shear diffusion with periodic boundary conditions on a perturbed quadrilateral mesh) using various non-compact schemes. Missing data points correspond to unstable configurations. Dashed gray lines: convergence rates m. Symbol Key: **BR1**:○, **GR-II**:▷, **LDG**:★, **GR-VI**:◇.

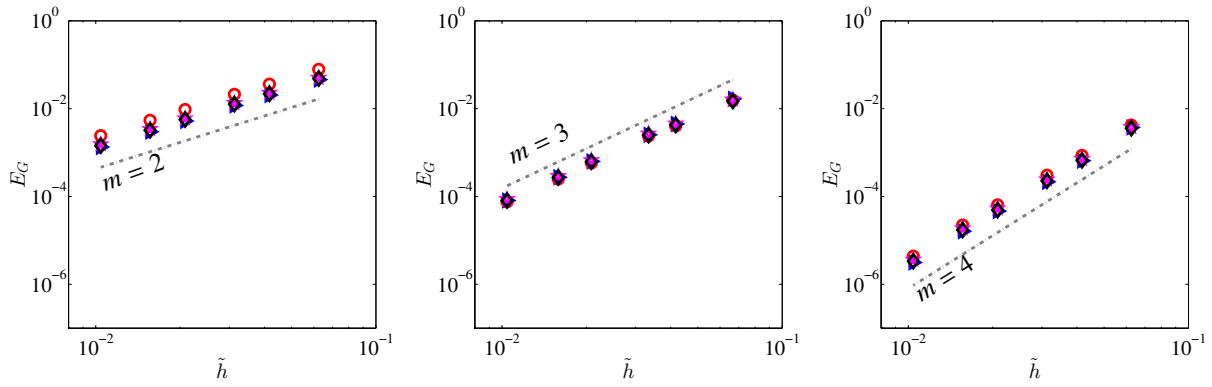


(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.17: Convergence study in cell-average error for test case 4 (shear diffusion with periodic boundary conditions on a perturbed quadrilateral mesh) using various non-compact schemes. Missing data points correspond to unstable configurations. Dashed gray lines: convergence rates m. Symbol Key: **BR1**:○, **GR-II**:▷, **LDG**:★, **GR-VI**:◇.

all schemes achieve optimal convergence in the global L_2 norm; order $p + 1$ convergence is also observed in the cell-average error. When the GR-VI and GR-II schemes maintain stability, they are more accurate than the LDG and BR1 schemes. However, their tendency towards instability is a liability, so we recommend GR-VI and GR-II on Cartesian meshes only. It is our hope that future studies uncover a way to stabilize these promising schemes.

We now discuss the compact schemes. With the default choice of $\chi = 2$, the CGR-Heavy and



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.18: Convergence study in global L_2 error for test case 4 (shear diffusion with periodic boundary conditions on a perturbed quadrilateral mesh) using various compact schemes. Dashed gray lines: convergence rates m. Symbol Key: **BR2**:○, **CGR-Heavy**:▷, **standard CGR**:★, **HAG**:◇.

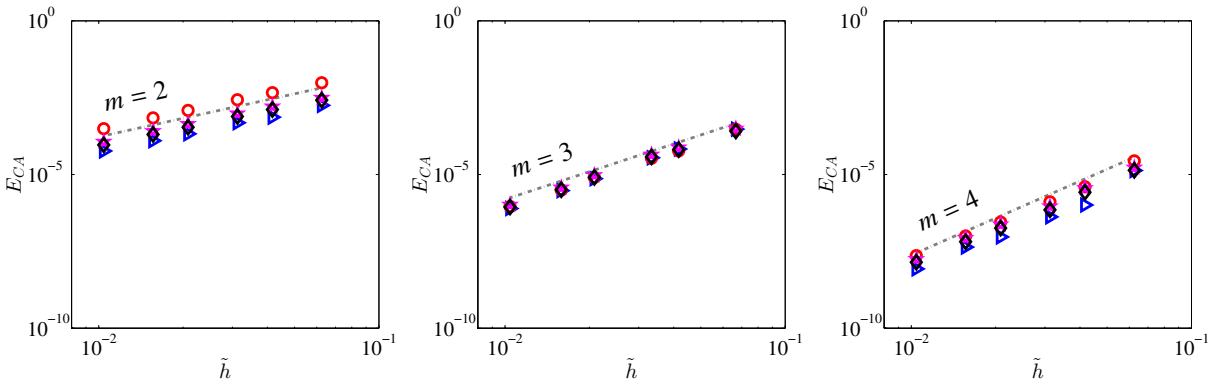
standard CGR schemes were unstable for $p > 2$. Thus, they are instead applied with $\chi = 4$ for this particular test case. In contrast, the HAG scheme and the established BR2 scheme maintained stability with $\chi = 2$. Figure 3.18 shows that all four of these compact schemes maintain optimal convergence in the global L_2 norm. In the cell-average error (Figure 3.19), the convergence rate is again $m = p + 1$. In general, the differences in accuracy are small, but where differences are apparent, the BR2 scheme is the least accurate scheme and CGR-Heavy is the most accurate scheme.

3.10 Further Evaluation and Discussion of CGR Schemes

This section provides further inspection of the CGR scheme and its light/heavy variants. In addition to discussing computational cost and comparing the three CGR schemes, we inspect the standard CGR scheme's error in the solution gradient.

3.10.1 Floating Point Operations

It is common to employ a nodal basis in DG, with the interpolation nodes being the Lobatto points (in 1D) or a 2D/3D tensor product grid built from the Lobatto points. Here, we estimate the



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 3.19: Convergence study in cell-average error for test case 4 (shear diffusion with periodic boundary conditions on a perturbed quadrilateral mesh) using various compact schemes. Dashed gray lines: convergence rates m. Symbol Key: **BR2**:○, **CGR-Heavy**:▷, **standard CGR**:★, **HAG**:◇.

flop count for a residual calculation, including multiplication by the inverse mass matrix, on a 2D quadrilateral element, assuming a nodal basis, with scalar Laplacian diffusion (no advection) as the flux law. In this case, $K = (p + 1)^2$. Table 3.16 lists the flop count per residual calculation substep for both the BR2 scheme and the standard CGR scheme; the quadrature node counts for the volume integrals (Q_V per element) and surface integrals (Q_S per face) are included as parameters. Costwise, the only difference between the two schemes is the calculation of $\tilde{U}$, where CGR uses the discrete recovery operator (see Appendix A); BR2 has a cost advantage here, as an element's solution on an interface quadrature point involves only the DOFs lying along the interface. Note that for interface-related calculations, the overall workload is typically shared between a pair of elements, hence the occasional division by 2 in the flop counts of individual processes. Table 3.17 gives the estimated flop count per DOF per residual evaluation (using the formulae of Table 3.16) for $p = 1$ through $p = 5$ with $Q_S = p + 1$ and $Q_V = K$. The $\tilde{U}$ calculation is a relatively small part of the overall cost to compute the residual, so the computational cost penalty incurred by CGR is less than ten percent.

3.10.2 Error in Solution Gradient

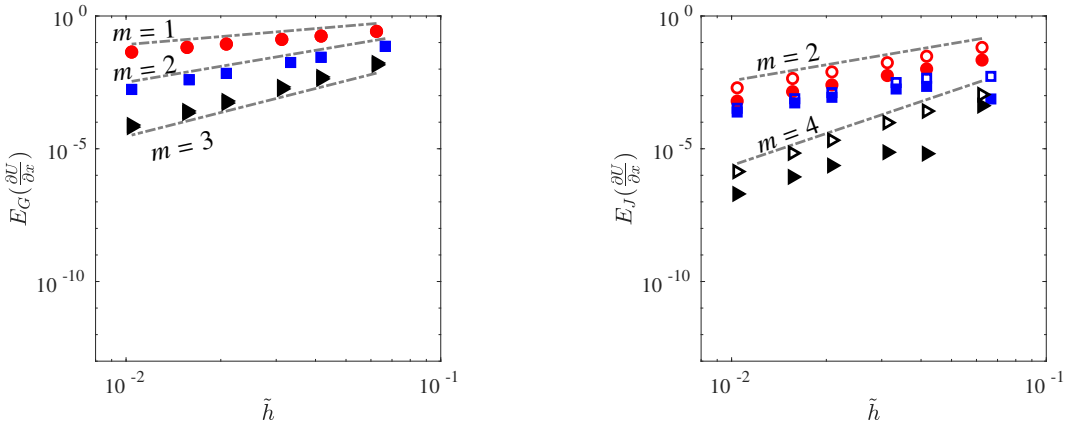
The BR2 and CGR schemes are compared based on the global and functional errors in the x and y derivatives. The broken gradient $\nabla^h U$ (see Eq. 3.5) is applied to measure the gradient of the numerical solution U^h; then, either the x component or y component of $\nabla^h U$ is compared to the derivative of the exact solution. The error is measured using the broken gradient instead of the auxiliary polynomial because the procedure for forming the broken gradient from $\hat{\mathbf{U}}$ is identical for BR2 and CGR, while the procedure for the formation of $\boldsymbol{\sigma}$ differs between the methods. The error in each derivative is measured using both the global and functional errors; the functional error uses the same kernel κ used to measure the error in U. Fig. 3.20 shows these errors in $\frac{\partial U}{\partial x}$ from test case 2B; because of the symmetry of this problem, we omit the error in the y derivative. In the global error norm, both BR2 and CGR produce a convergence rate $m = p$; this behavior is expected, as the broken gradient $\nabla^h U$ is a degree $p-1$ polynomial. In the functional error norm, for

Table 3.16: Floating point operations per element (2D quadrilateral) per procedure. $K = (p+1)^2$, Q_V is number of volume quadrature points, Q_S is number of quadrature points on each interface, $N_F = 4$ is number of sides, $N_D = 2$ is number of spatial dimensions.

Step	Calculation	Flops (BR2)	Flops (CGR)
1	$\tilde{U}$, interface quadr. points	$N_F \times Q_S \times 4(p+1)/2$	$N_F \times Q_S \times 4K/2$
2	$\boldsymbol{\sigma}$, interior quadr. points	$N_D \times Q_V \times (2K + 2N_F \times Q_S)$	$N_D \times Q_V \times (2K + 2N_F \times Q_S)$
3	$\boldsymbol{\mathcal{G}}(\boldsymbol{\sigma})$, interior quadr. points	$N_D \times Q_V$	$N_D \times Q_V$
4-5	$\tilde{\boldsymbol{\sigma}}$, interface quadr. points	$N_F \times N_D \times Q_S \times 4K/2$	$N_F \times N_D \times Q_S \times 4K/2$
6	$\boldsymbol{\mathcal{G}}(\tilde{\boldsymbol{\sigma}})$, interface quadr. points	$N_D \times N_F \times Q_S$	$N_D \times N_F \times Q_S$
7	$\frac{d}{dt}\hat{\mathbf{U}}$, given $\boldsymbol{\mathcal{G}}$ distribution	$K \times (2N_D \times Q_V + N_F \times 2Q_S + 3K)$	$K \times (2N_D \times Q_V + N_F \times 2Q_S + 3K)$

Table 3.17: Total flop count per DOF to populate DG residual given $\hat{\mathbf{U}}$.
$K = (p+1)^{N_D}$, $Q_V = K$, $Q_S = p+1$, $N_F = 4$, $N_D = 2$.

p	Flops (BR2)	Flops (CGR)	$\frac{\text{CGR Flops}}{\text{BR2 Flops}}$
1	114	122	1.070
2	195.7	211.7	1.082
3	300	324	1.080
4	426.6	458.6	1.075
5	575.3	615.3	1.070



(a) Global error in $\frac{\partial U}{\partial x}$.

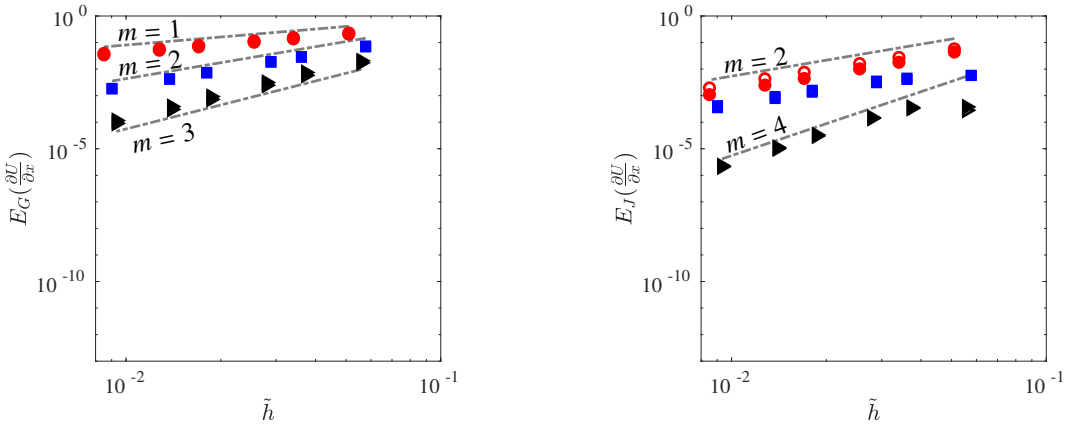
(b) Functional error in $\frac{\partial U}{\partial x}$.

Figure 3.20: Convergence study in $\frac{\partial U}{\partial x}$ for test case 2B (scalar shear diffusion with Dirichlet boundary conditions on a Cartesian mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2 ($p1$:○, $p2$:□, $p3$:▷).

both BR2 and CGR, the x derivative converges at rate $m = 2$ for $p = 1$ and $p = 2$, then jumps to rate $m = 4$ for the $p = 3$ case. However, while the convergence rates are similar, the gradient error of the CGR scheme is consistently smaller than that of the BR2 scheme. Figs. 3.21 and 3.22 show the global and functional error norms for the x and y derivatives in test case 3B; due to the asymmetry of the domain in this test case (see Fig. 3.7c), we show errors in both derivative components. BR2 and CGR perform similarly in this test case. These results suggest that with regard to gradient accuracy, CGR is more accurate than BR2 when the mesh is Cartesian and behaves similarly to BR2 when mesh uniformity is lost.

3.10.3 CGR variants

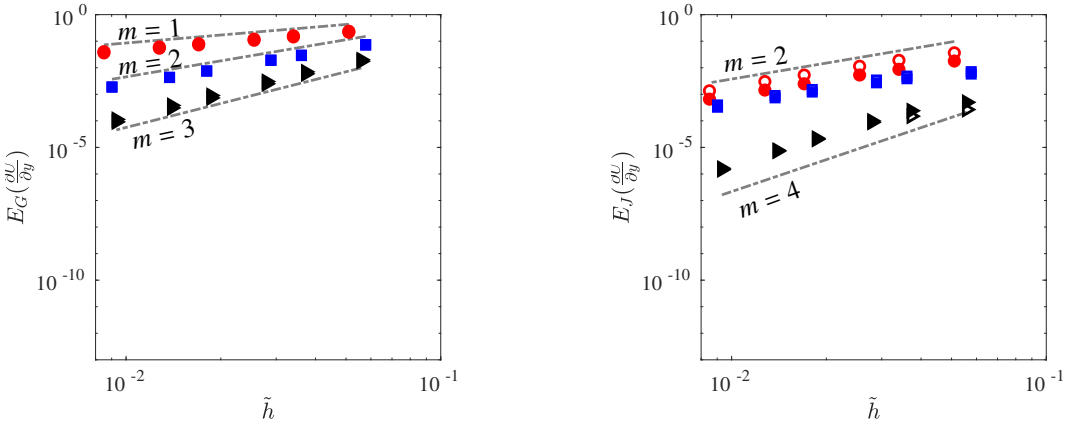
Figs. 3.23 and 3.24 show mesh refinement studies using the three CGR variants (standard CGR, CGR-Heavy, and CGR-Light) to solve the scalar shear diffusion problem on Cartesian (case 2A) and uniform simplex (case 3A) meshes, respectively, with periodic boundary conditions for $p = 1$ and $p = 3$. On Cartesian elements, the CGR-Heavy variant gives identical results to the standard CGR scheme while CGR-Light is less accurate. On simplex elements, the Heavy variant is more



(a) Global error in $\frac{\partial U}{\partial x}$.

(b) Functional error in $\frac{\partial U}{\partial x}$.

Figure 3.21: Convergence study in $\frac{\partial U}{\partial x}$ for test case 3B (scalar shear diffusion with Dirichlet boundary conditions on a nonuniform simplex mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2 ($p1$:○, $p2$:□, $p3$:▷).

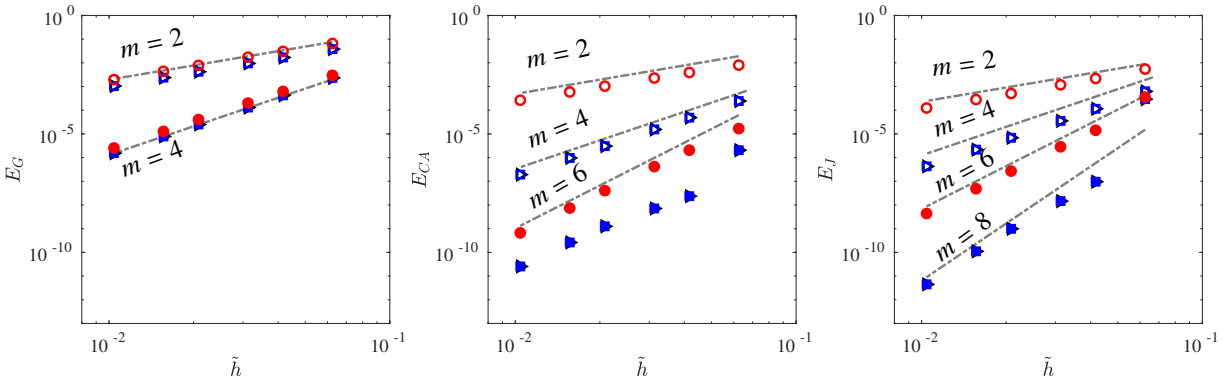


(a) Global error in $\frac{\partial U}{\partial y}$.

(b) Functional error in $\frac{\partial U}{\partial y}$.

Figure 3.22: Convergence study in $\frac{\partial U}{\partial y}$ for test case 3B (scalar shear diffusion with Dirichlet boundary conditions on a nonuniform simplex mesh). Dashed gray lines: convergence rates m; closed symbols: CGR; open symbols: BR2 ($p1$:○, $p2$:□, $p3$:▷).

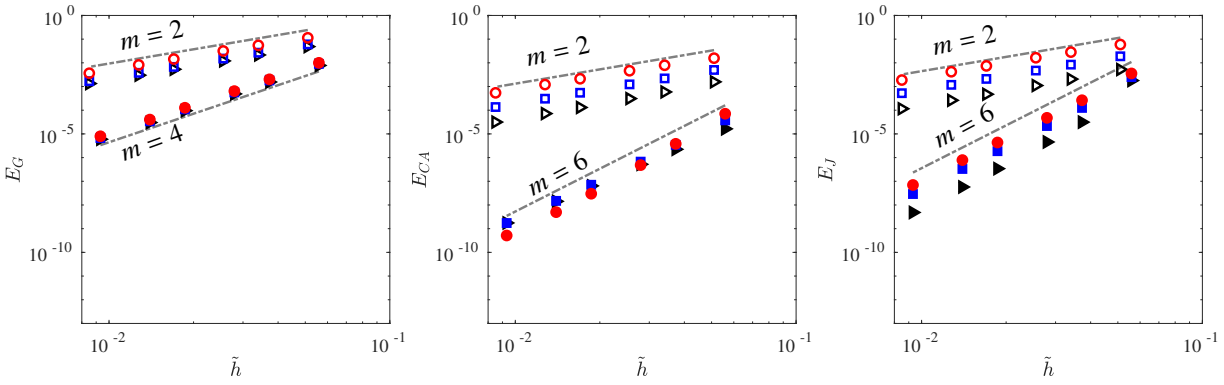
accurate than the standard CGR scheme but does not offer improved convergence rates. However, the standard CGR method is more robust under nonuniform mesh geometry, such as that used for case 3B.



(a) Global error. (b) Cell-average error. (c) Functional error.

Figure 3.23: Convergence study in U for test case 2A (scalar shear diffusion with periodic boundary conditions on a Cartesian mesh) with standard CGR and the Heavy/Light variants. Dashed gray lines: convergence rates m; open symbols: $p = 1$; closed symbols: $p = 3$ (CGR-Light:○, CGR-Standard:□, CGR-Heavy:▷).

Implemented in the mixed formulation, the three CGR schemes exhibit different levels of accuracy depending on the application of the recovery operation; BR2 does not employ Recovery but rather uses an arithmetic average to calculate interface terms. CGR-Light makes minimal use of the recovery operation, leveraging it only to calculate the interface gradient from the semi-connected gradients of neighboring elements. Fourier analysis and computational results indicate that this sparse application of the recovery operation provides no benefit in accuracy compared to the BR2 approach. In contrast to CGR-Light, the CGR-Heavy variant and the standard CGR scheme make use of the recovery operation when populating the auxiliary variable by setting $\tilde{U} = \mathcal{R}(U_A^h, U_B^h)$. Consequently, the gradient approximation used to calculate the fluxes for the element interior integrals is far more accurate than with CGR-Light and BR2. The improved accuracy of the fluxes over the element interiors is the feature that allows CGR-Heavy, standard CGR, and the HAG scheme to achieve greater accuracy (order $2p + 2$ for odd p).



(a) Global error. (b) Cell-average error. (c) Functional error.

Figure 3.24: Convergence study in U for test case 3A (scalar shear diffusion with periodic boundary conditions on a uniform simplex mesh) with standard CGR and the Heavy/Light variants. Dashed gray lines: convergence rates m; open symbols: $p = 1$; closed symbols: $p = 3$ (CGR-Light:○, CGR-Standard:□, CGR-Heavy:▷).

3.11 Chapter Conclusion

The Interface Gradient Recovery (IGR) discontinuous Galerkin family for discretizing diffusion problems, obtained by combining the Recovery concept with the mixed formulation, contains multiple attractive numerical schemes. In particular, the GR-II scheme stands out as having small spectral radii and high orders of accuracy relative to the established BR1 and LDG schemes. The GR-II, GR-VI, and CGR methods of the family have been shown to generalize to 2D problems and perform well on Cartesian elements. Fourier analysis and computional test cases revealed that as we suspected, the use of the recovery operator in the calculation of ambiguous interface flux terms yields relatively accurate mixed formulation schemes compared to the established LDG, BR1, and BR2 approaches. However, the GR-II and GR-VI methods are unsuitable for non-Cartesian elements.

We favor the standard CGR method on account of many properties: it has smaller spectral radii than established DG schemes across all values of p analyzed, it is order $2p + 2$ accurate for odd p, and it appears robust on both simplex and Cartesian elements. Perhaps most importantly, it

maintains the compact computational stencil of the conventional DG advection scheme. The HAG scheme offers the same orders of accuracy as the standard CGR scheme while being more robust, but it has larger spectral radii, which becomes disadvantageous when explicit time integration is employed. The CGR-Heavy scheme provides superior performance compared to HAG and the standard CGR scheme on perturbed quadrilateral elements while being equivalent to standard CGR on Cartesian elements, so the CGR-Heavy variant was chosen as our designated diffusion scheme in the formation of the Recovery-assisted advection-diffusion schemes of Chapter 5.

CHAPTER 4

Recovery-assisted Advection Schemes

4.1 Chapter Overview

In the previous chapter, we showed how the Recovery concept can be applied to form DG schemes that achieve an accuracy advantage over conventional DG approaches for diffusion problems. However, in the discretization of an advection-diffusion problem, the scheme used to capture the diffusive flux terms is only a piece of the complete discretization; one must also employ a scheme to properly capture the advective terms. This chapter is devoted to the study of advection schemes that can achieve an accuracy advantage over the conventional DG method.

Fourier analysis indicates that the conventional DG method (with an upwind flux at interfaces) is order $2p+1$ accurate for the linear advection equation. The interface-centered binary (ICB) reconstruction schemes of Khieu & Johnsen [56] improve this order of accuracy to $2p+2$ in a stable fashion with a biased version of the recovery operator. Cell-centered reconstruction schemes were also proposed by Khieu & Johnsen [56], with extraordinarily high orders of accuracy, but we did not pursue further development of those schemes because they exhibit non-compact computational stencils. Instead, we devoted our efforts to further analysis and development of the ICB reconstruction approach for advection problems. We refer to the original ICB scheme as Modal ICB to distinguish it from our newer approach, known as Lagrange ICB. After describing these schemes, we inspect them via Fourier analysis and linear advection test problems. We also demonstrate scheme performance for the 1D Euler equations to show that when paired with an appropriate lim-

iting scheme, the ICB discretizations maintain stability in the presence of physical discontinuities.

We note that the practice of applying reconstructions to improve the accuracy of the basic DG method for advection problems has been applied in the reconstructed DG method [70, 79] and the $P_N P_M$ method of Dumbser [31]. Henry de Frahan also experimented with variations of the interface-centered binary reconstruction schemes in his thesis [42]. All three of these listed approaches lead to non-compact computational stencils, hence the decision to pursue the analysis and alteration of the original ICB approach instead.

4.1.1 Novelty and Articles

The Lagrange ICB scheme is a novel ICB variant, inspired by the older approach (namely Modal ICB) of Khieu & Johnsen [56]. As the Lagrange ICB approach was only recently proposed [51], we are the first to perform Fourier analysis on the scheme. Additionally, this chapter contains the first formal evaluation of the Modal and Lagrange ICB schemes under the influence of a limiting scheme (for shock-capturing).

The material of this chapter appears in one AIAA conference manuscript and one in-preparation article:

- P. E. Johnson & E. Johnsen, *A Compact Discontinuous Galerkin Method for Advection-Diffusion Problems*, AIAA Paper 2018-1091.
- P. E. Johnson, L. H. Khieu, & E. Johnsen, *A Compact Recovery-Assisted Discontinuous Galerkin Method for the Compressible Navier-Stokes Equations*, in preparation.

4.1.2 Usage of Recovery

The novel schemes in this chapter employ the biased form of the recovery operator in both the Modal ICB and Lagrange ICB configurations (Section 2.6.3). Without exception, the recovery operators are applied in the derivative-based implementation of Section 2.6.6.

4.2 Scheme Design

The ICB schemes for advection upgrade the conventional upwind DG scheme for hyperbolic PDEs with the biased recovery operation (introduced in Section 2.6). The goal of the biased recovery operators is to retain the upwinding capability of the conventional DG scheme while improving accuracy. Since the biased reconstruction process involves two reconstructions (one biased towards each element) for each interface, a solution jump is retained at the interface, allowing sufficient numerical dissipation to be introduced via the practice of upwinding. Usage of the full-order recovery operator is not prudent for advection problems because if the recovered solution is applied to calculate the common interface flux, then the upwinding capability of the DG discretization is removed due to the lack of any jumps in the smooth recovered solution.

Since the Modal and Lagrange varieties of the biased recovery operators were already described (Section 2.6), the description of the new advection schemes is a simple matter. Consider a hyperbolic PDE problem expressed in conservation law form,

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = 0, \tag{4.1}$$

to be satisfied by the DG polynomial approximation U^h. The DG weak form constraining the evolution of the DOFs $\hat{\mathbf{U}}$ is repeated here:

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t}(\sum_{n=0}^{K-1} \hat{U}_m^n \phi_m^n) d\boldsymbol{x} = -\int_{\partial\Omega_m} \phi_m^{k-}(\tilde{\boldsymbol{\mathcal{F}}} \cdot \mathbf{n}_{\mathbf{m}}^-) ds + \int_{\Omega_m} \nabla\phi_m^k \cdot \boldsymbol{\mathcal{F}} d\boldsymbol{x}, \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \boldsymbol{\Omega}. \tag{4.2}$$

The only ambiguity is the definition of the common flux $\tilde{\boldsymbol{\mathcal{F}}}$ along interfaces. The strategy for calculating this flux is what distinguishes the ICB schemes from the conventional DG scheme. Consider a quadrature point $\boldsymbol{x}_g$ along an interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$ shared by Ω_A and Ω_B. Let U_L

and U_R be the competing limits of U^h, from inside Ω_A and Ω_B, respectively, at $\boldsymbol{x}_g$:

$$U_L = \lim_{\boldsymbol{x}\to\boldsymbol{x}_g} U_A^h \quad , \quad U_R = \lim_{\boldsymbol{x}\to\boldsymbol{x}_g} U_B^h. \tag{4.3}$$

In the conventional "upwind DG" approach, the flux $\tilde{\boldsymbol{\mathcal{F}}}$ is calculated using an exact or approximate Riemann solver (recall Section 2.5), effectively introducing some dissipation via upwinding: $\tilde{\boldsymbol{\mathcal{F}}} = Rie(U_L, U_R, \mathbf{n}_\mathbf{A}^-)$. A central DG scheme, which is dissipation-free, is formed by instead taking the interface flux to be the average of the individual fluxes determined from the left and right solution states: $\tilde{\boldsymbol{\mathcal{F}}} = \frac{1}{2}(\boldsymbol{\mathcal{F}}(U_L) + \boldsymbol{\mathcal{F}}(U_R))$.

The Modal and Lagrange ICB schemes for advection also make use of the Riemann solver, and as suggested by the scheme names, it is at this point that we make use of the biased reconstruction operations detailed in Section 2.6. Considering the same interface quadrature point $\boldsymbol{x}_g$, the left and right solution states are taken from the two ICB reconstructions over the union $\Omega_A \cup \Omega_B$:

$$U_L = \lim_{\boldsymbol{x}\to\boldsymbol{x}_g} U_A^{ICB} \quad , \quad U_R = \lim_{\boldsymbol{x}\to\boldsymbol{x}_g} U_B^{ICB}, \tag{4.4}$$

where U_A^{ICB} is the A-biased reconstruction and U_B^{ICB} is the B-biased reconstruction. Again, the interface flux is calculated using the Riemann solver: $\tilde{\boldsymbol{\mathcal{F}}} = Rie(U_L, U_R, \mathbf{n}_\mathbf{A}^-)$. The Modal and Lagrange ICB schemes are distinguished by the type of biased recovery used to form U^{ICB}, either Modal or Lagrange. The change in the interface flux calculation is the *only difference* between the conventional upwind DG approach and the ICB schemes. The various DG schemes for advection are summarized in Table 4.1. For element edges coincident with the Dirichlet boundary $\boldsymbol{\partial\Omega_D}$, the advective flux is set based on the boundary trace of U^h and the specified boundary condition.

With regard to compactness, the ICB reconstructions particular to an interface use information only from the neighboring elements Ω_A and Ω_B; thus, the interface flux $\tilde{\boldsymbol{\mathcal{F}}}$ depends only on $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$. The ICB schemes therefore maintain the compact, nearest-neighbors stencil of the conventional upwind DG method.

In addition to the two main Modal and Lagrange ICB schemes, we also introduce the "Cut" ICB

scheme, abbreviated ICBC. The ICBC scheme is a modification of the Modal ICB approach. To obtain ICBC from Modal ICB, set $C_p = 0$ in the derivative-based recovery operator (Eq. 2.76). This procedure removes the highest available derivative term from the interface solution approximation, hence the designation "Cut" ICB. Experimentation demonstrated that on unstructured meshes, the removal of the top derivative term by setting $C_p = 0$ can be helpful for numerical stability. Indeed, there may be an abundance of attractive DG schemes available by various combinations of recovery weights $\boldsymbol{C}$ in the derivative-based interface approximation (Eq. 2.76).

4.2.1 Relationships among ICB, central DG, and upwind DG

In the case of the 1D linear advection equation, the usage of the derivative-based recovery approach in the DG weak form (Eq. 4.2) produces the following scheme, assuming $a > 0$ and the use of the upwind flux at each interface:

$$\begin{aligned}\int_{\Omega_m} \phi_m^k \frac{\partial U}{\partial t} dx - \int_{\Omega_m} a U_m^h \frac{d\phi_m^k}{dx} dx + \left[a\phi_m^{k-} C_0 U_L \right]_{x_m}^{x_m+h} \\ + \left[a\phi_m^{k-} (1 - C_0) U_R \right]_{x_m}^{x_m+h} \\ + \sum_{j=1}^{p} C_j h^j \left[a\phi_m^{k-} \left[\left[\frac{\partial^j U}{\partial x^j} \right] \right] \right]_{x_m}^{x_m+h} = 0 \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \boldsymbol{\Omega}.\end{aligned} \tag{4.5}$$

Table 4.1: Summary of advection schemes. The distinguishing procedure is the technique for the advective interface flux $\tilde{\boldsymbol{\mathcal{F}}}$ along each interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$.

Scheme	Abbreviation	$\tilde{\boldsymbol{\mathcal{F}}}$	$\boldsymbol{\mathcal{F}}\|_{x\in\Omega_m}$
conventional (upwind) DG	conDG	$Rie(U_A^h, U_B^h, \mathbf{n}_\mathbf{A}^-)$	$\boldsymbol{\mathcal{F}}(U_m^h)$
Modal ICB	ICBM	$Rie(U_A^{ICBM}, U_B^{ICBM}, \mathbf{n}_\mathbf{A}^-)$	$\boldsymbol{\mathcal{F}}(U_m^h)$
Lagrange ICB	ICBL	$Rie(U_A^{ICBL}, U_B^{ICBL}, \mathbf{n}_\mathbf{A}^-)$	$\boldsymbol{\mathcal{F}}(U_m^h)$
Cut ICB	ICBC	$Rie(U_A^{ICBC}, U_B^{ICBC}, \mathbf{n}_\mathbf{A}^-)$	$\boldsymbol{\mathcal{F}}(U_m^h)$
central DG	cenDG	$\frac{1}{2}(\boldsymbol{\mathcal{F}}(\boldsymbol{U_A^h}) + \boldsymbol{\mathcal{F}}(\boldsymbol{U_B^h}))$	$\boldsymbol{\mathcal{F}}(U_m^h)$

Recall that x_m and $x_m + h$ are the left and right endpoints, respectively, of Ω_m. For a given interface (either the left or right boundary of Ω_m), U_L is the limit of U^h from the left side of the interface and U_R is the limit of U^h from the right side of the interface. The weights C_0, C_1, etc. are the recovery weights in the derivative-based interface approximation (Eq. 2.76); recall that these weights depends on the particular type of biased recovery employed. In the $a > 0$ case, it is the left element's ICB reconstruction that is employed to calculate $\tilde{\mathcal{F}}$. Following convention, given some quantity v that is multivalued at an interface, $[[v]] = v_L - v_R$. Further manipulation yields

$$\int_{\Omega_m} \phi_m^k \frac{\partial U}{\partial t} dx - \int_{\Omega_m} a U_m^h \frac{d\phi_m^k}{dx} dx + \left[a\phi_m^{k-} \left(\{\{U\}\} + [[U]] \left(C_0 - \frac{1}{2} \right) \right) \right]_{x_m}^{x_m+h} + \sum_{j=1}^{p} C_j h^j \left[a\phi_m^{k-} \left[\left[\frac{\partial^j U}{\partial x^j} \right] \right] \right]_{x_m}^{x_m+h} = 0 \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \boldsymbol{\Omega}, \tag{4.6}$$

where $\{\{U\}\} = \frac{1}{2}(U_L + U_R)$ denotes the arithmetic average at an interface. It is evident from Eq. (4.6) that by using the derivative-based recovery form to populate the interface solution traces, we are adding interface derivative terms to the DG weak form. The conventional upwind DG scheme and the central DG scheme correspond to $C_j = 0 \; \forall \; j > 0$. The choice $C_0 = 1$ corresponds to the upwind scheme and $C_0 = \frac{1}{2}$ corresponds to the central scheme. When the biased recovery approach is employed (as in the Modal or Lagrange ICB schemes), the recovery weights C_j become nonzero for $j > 0$. Clearly, the ICB approaches cannot be replicated simply by mixing the upwind and central DG schemes, as the ICB approach adds terms not seen in the DG weak form for either the upwind or central DG schemes. We note that the idea of adding derivative terms to the DG weak form was mentioned by Gassner et al. [37] in the analysis of the diffusive generalized Riemann problem and by Zhang et al. [116] in the formation of the direct discontinuous Galerkin method for diffusive flux terms, suggesting a connection to the many DG variants proposed in this document. The higher-order spatial derivative terms are also present in the ADER-DG schemes [32] for advection problems; however, the ADER-DG approach employs the spatial derivatives as a means to close the fully discrete space-time discontinuous Galerkin method, so a direct connection to the ICB

advection schemes is unlikely.

4.3 Fourier Analysis

The schemes listed in Table 4.1 were inspected in the context of the 1D linear advection equation with the standard Fourier analysis approach detailed in Section 2.4. For a thorough comparison of advection schemes, the standard Fourier analysis approach is insufficient, so in addition to identifying stability, spectral radius, and order of accuracy, we employed the wavenumber resolution analysis of Watkins et al. [112]. Where the order of accuracy calculation from the standard Fourier analysis technique characterizes the consistent eigenvalue's behavior in the limit of $\omega \to 0$, the wavenumber resolution analysis includes all of the eigenvalues and can provide a measure of scheme performance over moderate wavenumbers.

4.3.1 Stability and Spectral Radius

All schemes analyzed are linearly stable, meaning that the eigenvalues of the update matrix $\mathcal{A}$ remain nonpositive. Table 4.2 lists the spectral radii and orders of accuracy of the five DG schemes described in Table 4.1. With respect to the spectral radius, the Lagrange ICB method offers dramatic improvement over the other schemes analyzed, which is an attractive property for those interested in explicit time integration. We note that spectral radius is not the sole determining factor in the timestep size; the timestep must be sufficiently small that the eigenvalues of the spatial discretization lie completely within the Fourier footprint of the explicit time integration scheme.

Table 4.2: Spectral Radii (ρ_s) and orders of accuracy (o.o.a.) for the DG advection schemes.

Scheme:	**conDG**		**Modal ICB**		**Lagrange ICB**		**cenDG**		**Cut ICB**	
p	ρ_s	o.o.a.	ρ_s	o.o.a.	ρ_s	o.o.a.	ρ_s	o.o.a.	ρ_s	o.o.a.
1	6.0	3	4.0	4	3.0	3^+	4.0	2	4.5	3
2	11.8	5	9.0	6	5.5	6	8.0	6	10.8	5
3	19.2	7	15.5	8	8.4	7^+	13.3	6	15.5	7
4	27.8	9	23.2	10	11.8	10	19.7	10	24.7	9
5	37.8	11	32.1	12	15.6	11^+	27.3	10	32.1	11

Thus, while the timestep size scales inversely with the spectral radius, the precise limit on the timestep size also depends on the time integration scheme.

The astute reader may wonder why the biased recovery approach in 1D always uses a degree $p+1$ polynomial for each reconstruction; presumably, the accuracy of the advection scheme could be improved by raising the degree of the biased recovered solution. Khieu & Johnsen [56] explored this issue; their results show that regardless of the solution order p, the Modal ICB advection scheme becomes unstable if the degree of the reconstructed polynomial is greater than $p+1$. Our analysis with the Lagrange ICB scheme yielded the same conclusion: if the reconstructed polynomial is greater than degree $p+1$, the ICB advection scheme is unstable.

4.3.2 Order of Accuracy

The orders of accuracy are listed in Table 4.2. Using the strategy outlined in Section 2.4, we encountered difficulty determining the order of accuracy of the Lagrange ICB scheme. For the $p=1$, $p=3$, and $p=5$ cases, the method's order of accuracy seems to sit between order $2p+1$ and order $2p+2$, hence the tabulated orders of accuracy 3^+, 7^+, and 11^+. This difficulty was part of our motivation to instead pursue the wavenumber resolution technique. The Modal ICB scheme is always order $2p+2$ accurate. The conventional upwind DG scheme is always order $2p+1$ accurate. Thus, the ICB schemes provide an accuracy advantage.

4.3.3 Wavenumber Resolution

We briefly describe the wavenumber resolution analysis technique of Watkins et al. [112] and apply it to compare the advection schemes. To begin, diagonalize the update matrix:

$$\mathcal{A}(\omega) = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}, \tag{4.7}$$

where $\boldsymbol{V}$ is the $K \times K$ collection of eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues, where $K = p+1$ is the number of DOFs per element. Now, take Ω_m to be the element whose left endpoint

is $x = 0$. Using a nodal basis with the interpolation points being the K Gauss-Legendre points in each element, the DG degrees of freedom of the initial condition's projection are

$$\hat{\mathbf{U}}_{\mathbf{m}}(\omega, 0) = \begin{bmatrix} \exp(i\omega\xi_0) \\ \exp(i\omega\xi_1) \\ \vdots \\ \exp(i\omega\xi_p) \end{bmatrix}, \tag{4.8}$$

where ξ_k is the k^{th} Gauss-Legendre point on the unit reference element. With the initial condition $\hat{\mathbf{U}}_{\mathbf{m}}(\omega, 0)$ defined, the first-order system of ordinary differential equations (Eq. 2.36a) has the solution:

$$\hat{\mathbf{U}}_{\mathbf{m}}(\omega, t) = \boldsymbol{V} \exp(\boldsymbol{\Lambda} t) \boldsymbol{V}^{-1} \hat{\mathbf{U}}_{\mathbf{m}}(\omega, 0). \tag{4.9}$$

Now, let $\boldsymbol{\beta}$ be the $K \times 1$ vector of expansion coefficients necessary to write the initial condition in terms of the eigenvectors:

$$\boldsymbol{V}\boldsymbol{\beta} = \hat{\mathbf{U}}_{\mathbf{m}}(\omega, 0). \tag{4.10}$$

Given the eigenvector expansion coefficients $\boldsymbol{\beta}$, Watkins et al. [112] derive an upper bound on the initial growth rate of the error, labelled $\mathcal{E}$:

$$\mathcal{E}(\omega) = \frac{1}{\sqrt{p+1}} \sum_{n=1}^{p+1} |\beta_n||\lambda_n - \lambda^{ex}|. \tag{4.11}$$

In this equation, λ_n is the n^{th} eigenvalue of $\mathcal{A}$ and β_n is the n^{th} entry of $\boldsymbol{\beta}$ for a given ω. In the immediate case of 1D linear advection, the exact eigenvalue is $\lambda^{ex}(\omega) = -i\omega$.

Given some error tolerance ϵ, there exists a cutoff wavenumber ω_f such that $\mathcal{E}(\omega) \leq \epsilon$ for all $\omega \in [0, \omega_f]$. The resolving efficiency [112] is then defined as

$$\eta = \frac{\omega_f}{(p+1)\pi}. \tag{4.12}$$

The resolving efficiency calculation involves the full eigenvalue spectrum of $\mathcal{A}$. The vector $\boldsymbol{\beta}(\omega)$

determines which eigenvalues are active at a given wavenumber ω, so there is no need to designate the principal eigenvalue in the analysis process. The resolving efficiency corresponds to the maximum wavenumber that can be properly advected within the given error tolerance. In other words, for a fixed number of gridpoints, a scheme with relatively high resolving efficiency will do a better job tracking small-scale flow features. As the goal of this thesis is the development of numerical schemes for LES and DNS of turbulent flows, broadband wavenumber resolution is of great interest to us, so the wavenumber analysis approach described is a valuable tool.

Table 4.3 lists the resolving efficiencies of various DG spatial discretizations for a tolerance $\epsilon = \frac{1}{10}$. Table 4.4 repeats the test with $\epsilon = 2$. For a given tolerance ϵ, a resolving efficiency of $\eta = 1$ indicates that all wavenumbers that can be properly projected onto the grid (as opposed to excessively high wavenumbers, which are aliased to lower wavenumbers) are transmitted with error growth rate less than ϵ. Note that Modal ICB and Cut ICB yield similar resolving efficiencies, indicating that while Modal ICB has a higher order of accuracy, the removal of the C_p term does not substantially affect the scheme's resolving efficiency. The central DG scheme achieves better resolving efficiency than the conventional upwind DG scheme because the central scheme has zero dissipation error; however, as is shown in Section 4.4, this lack of numerical dissipation is ultimately unfavorable. The Lagrange ICB scheme generally achieves superior resolving efficiencies compared to the other schemes tested.

4.3.4 Dispersion/Dissipation

Figure 4.1 shows the imaginary components of the principal eigenvalues of the advection schemes for $p \leq 3$. The real (dissipative) components of the eigenvalues are reported in Fig. 4.2. Over-

Table 4.3: Resolving Efficiencies of Advection Schemes with $\epsilon = \frac{1}{10}$.

Scheme:	**conDG**	**Modal ICB**	**Lagrange ICB**	**cenDG**	**Cut ICB**
$p = 1$:	0.094	0.187	0.274	0.094	0.094
$p = 2$:	0.136	0.194	0.278	0.167	0.143
$p = 3$:	0.167	0.211	0.312	0.203	0.195
$p = 4$:	0.190	0.224	0.318	0.191	0.245
$p = 5$:	0.213	0.240	0.332	0.249	0.253

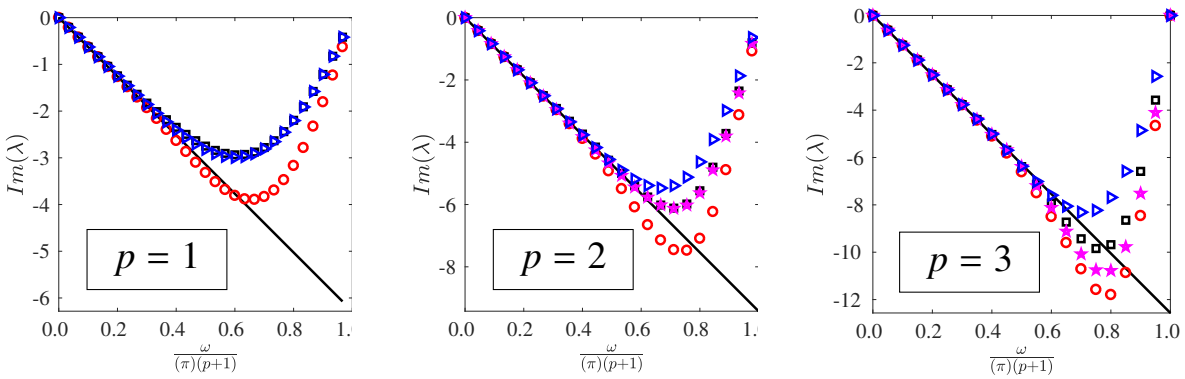


Figure 4.1: Imaginary component of the principal eigenvalue versus effective wavenumber. The exact eigenvalue, $\lambda = 0 - i\omega$, represents perfect translation of the initial condition. Symbol Key: Exact Eigenvalue: —— ; **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★.

all, the ICB schemes achieve superior accuracy compared to the conventional approach thanks not only to reduced dissipation, but also smaller dispersion error over small to moderate wavenumbers. Among the ICB schemes, Modal ICB is substantially more dissipative than Lagrange ICB, but both introduce some dissipation due to the upwinding behavior facilitated by the biased reconstruction process. These results show that the ICB schemes maintain an important property of the conventional upwind DG approach: when the dispersion error becomes large, numerical dissipation is also present to damp away spurious waves. The application of numerical dissipation in CFD is a topic of debate, but it is generally viewed as beneficial when applied in small doses.

4.4 Numerical Tests

Four numerical tests are employed to evaluate the various DG schemes for advection. These test cases are relatively simple; the more advanced test cases are reserved for the full advection-

Table 4.4: Resolving Efficiencies of Advection Schemes with $\epsilon = 2$.

Scheme:	**conDG**	**Modal ICB**	**Lagrange ICB**	**cenDG**	**Cut ICB**
$p = 1$:	0.461	0.554	0.675	0.751	0.544
$p = 2$:	0.371	0.434	0.598	0.354	0.464
$p = 3$:	0.364	0.406	0.535	0.420	0.425
$p = 4$:	0.362	0.397	0.507	0.505	0.397
$p = 5$:	0.361	0.390	0.509	0.360	0.385

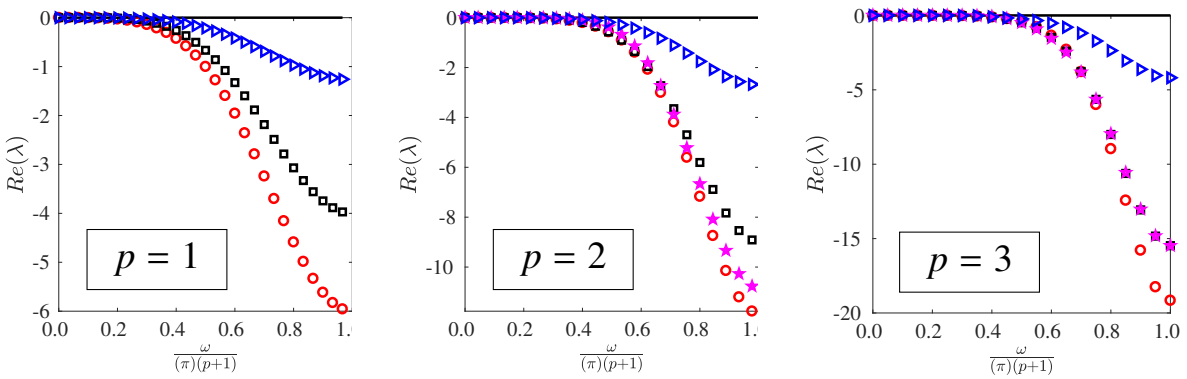


Figure 4.2: Real component of the principal eigenvalue versus effective wavenumber. The exact eigenvalue, $\lambda = 0 - i\omega$, represents perfect translation of the initial condition. Symbol Key: Exact Eigenvalue: — ; **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★.

diffusion schemes discussed in Chapter 5.

4.4.1 Test 1: Linear Advection of a Sine Wave

The spatial domain for this test is $\mathbf{\Omega} = x \in [0, 4\pi)$ with spatially periodic boundary conditions. The initial condition is $U(x, 0) = \sin(\frac{x}{2})$. Regardless of spatial discretization, the time integration scheme is the explicit 8^{th} order scheme of Prince & Dormand [86]. The advection speed is $a = \pi$ and the problem is simulated from $t = 0$ to $t_{final} = 80$, corresponding to 20 translational periods. The ICBC, cenDG, conDG, ICBM, and ICBL schemes are included. The mesh refinement study is presented in Figure 4.3 and Figure 4.4 for $p \in \{1, 3, 5\}$; both the global L_2 and cell-average error norms are plotted against the characteristic mesh width, $\tilde{h} = \frac{1}{nDOF}$. In this case, most configurations achieve order $p + 1$ convergence in the global L_2 norm. Note that the conDG scheme with $p = 1$ maintains third order convergence in the global L_2 norm. This behavior occurs because the solution is sufficiently underresolved that the dominant error in U^h comes from the evolution scheme itself, not the finite-dimensional solution representation. In contrast, for the more accurate ICBL scheme, the finite-dimensional solution representation is the dominant source of error, so the proper asymptotic rate of convergence ($m = p + 1 = 2$) is observed on the finer meshes. The convergence rates in the cell-average error tend towards the orders of accuracy from Fourier analysis, which is typical. The cell-average error in the $p = 5$ cases behaves irregularly because for the

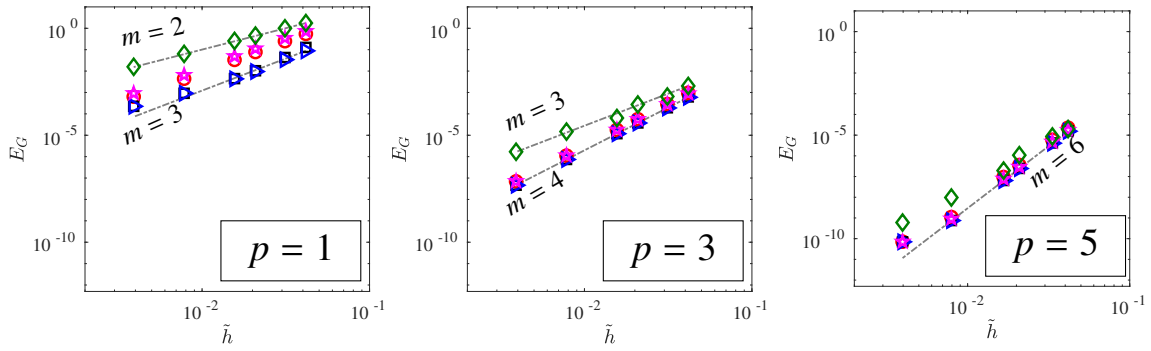


Figure 4.3: **Test 1** (Linear advection of a sine wave): Mesh refinement study in global L_2 error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★; **cenDG:** ◊.

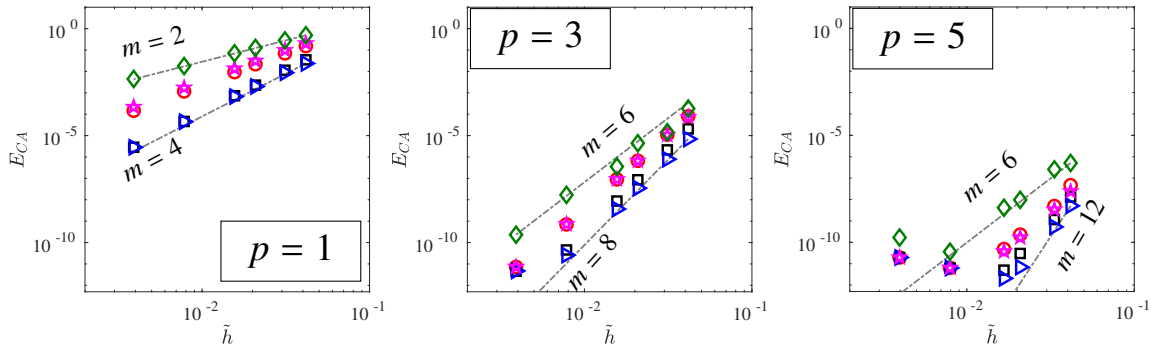
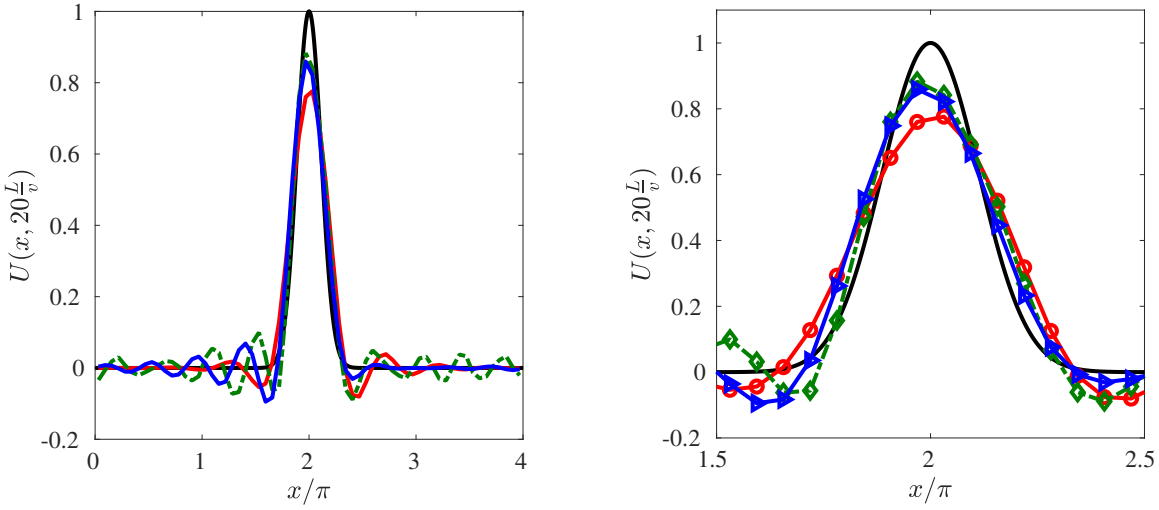


Figure 4.4: **Test 1** (Linear advection of a sine wave): Mesh refinement study in cell-average error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★; **cenDG:** ◊.

exceptionally long simulation time employed, the buildup of roundoff error becomes significant relative to the truncation error.

4.4.2 Test 2: Linear Advection of a Gaussian Pulse

This test case is identical to the first except that the initial condition is $U(x,0) = \exp(-4(x-2\pi)^2)$ and the mesh resolutions are finer. All other test parameters are held the same. Sample results are presented in Figure 4.5 to demonstrate scheme behavior. For a given mesh resolution, the lower wavenumber components are advected properly while the higher wavenumbers cannot be resolved properly; they are either dissipated away or remain in the system forever and manifest as spurious oscillations, as with the dissipation-free central DG scheme. A mesh refinement study



(a) Solution over full domain.

(b) Solution for $x \in \{1.5\pi, 2.5\pi\}$.

Figure 4.5: **Test 2** (Linear advection of a Gaussian pulse): Numerical solutions for Gaussian pulse problem with $p = 3$, $M = 16$ elements. The Lagrange ICB result shows less dissipation error than the conventional DG result and less dispersion error than the central scheme.
Symbol/Color Key: **Exact:** —— ; **conDG:**—○— ; **cenDG:** -.◊-. ; **ICBL:**—▷—.

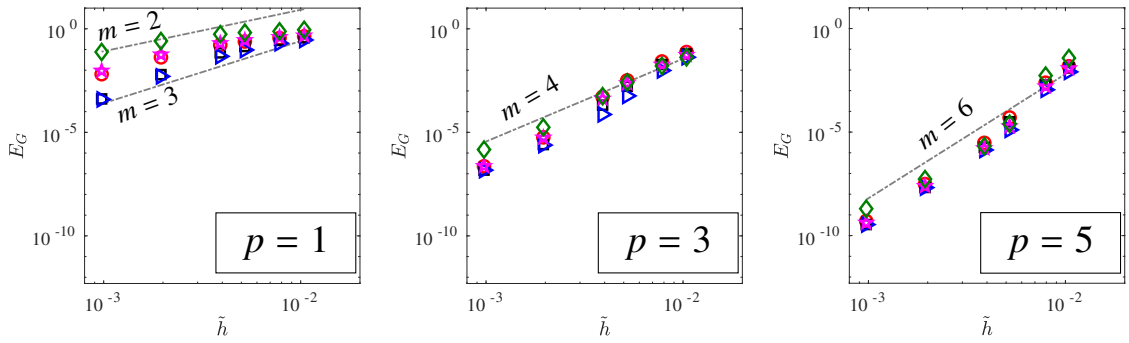


Figure 4.6: **Test 2** (Linear advection of a Gaussian pulse): Mesh refinement study in global L_2 error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★; **cenDG:** ◊.

is presented in Figure 4.6 and Figure 4.7 for $p \in \{1, 3, 5\}$. All three of the ICB schemes achieve optimal convergence (order $p + 1$) in the global L_2 norm. Overall, the Modal and Lagrange ICB schemes offer the best performance; the central DG scheme exhibits the worst performance.

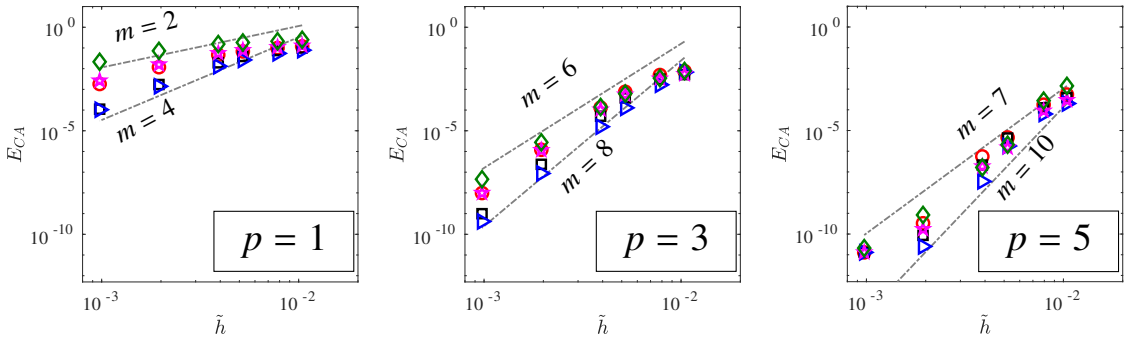


Figure 4.7: **Test 2** (Linear advection of a Gaussian pulse): Mesh refinement study in cell-average error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **ICBC:**★; **cenDG:** ◊.

4.4.3 Test 3: Linear Advection of Sine Wave, Limiter Active

This test case is identical to test 1 with the exception of one alteration. The basic DG advection scheme, whether it be central DG, conventional upwind DG, or an ICB variant, is supplemented with a limiter. The hierarchical limiting scheme of Krivodonova [64] is employed to limit the discontinuous polynomial U^h after each Runge-Kutta substep and at the end of each timestep. While a limiter is not necessary for stable simulation of the linear advection equation, it becomes necessary for the Euler equations. The purpose of this test, where the limiter is paired with the scalar advection equation, is to evaluate the accuracy of the ICB schemes when a limiting scheme is applied. The mesh refinement study is presented in Figure 4.8 and Figure 4.9 for $p \in \{1, 2, 3\}$. All three of the ICB schemes achieve optimal convergence (order $p + 1$) in the global L_2 norm. The central DG scheme performs poorly; the ICB and conventional DG schemes achieve nearly identical performance. The limiter reduces the convergence rate in the cell-average error to $m = p + 1$.

4.4.4 Test 4: Sod Problem

This test case features the 1D Euler equations (Eq. 2.9). It was proposed as a benchmarking problem in Sod's survey of numerical schemes for hyperbolic conservation laws [94]. The spatial

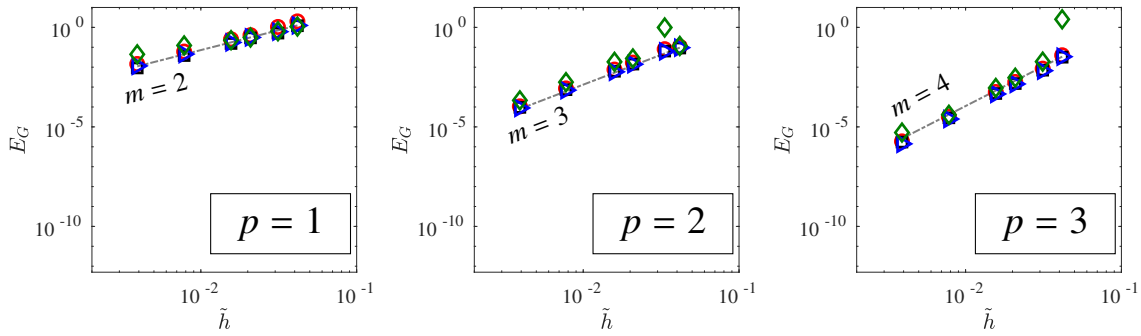


Figure 4.8: **Test 3** (Linear advection of a Sine wave with limiter): Mesh refinement study in global L_2 error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **cenDG:** ◊.

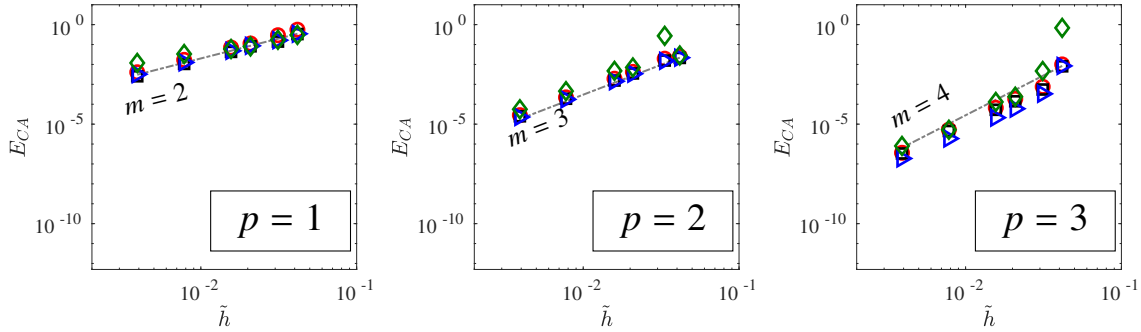
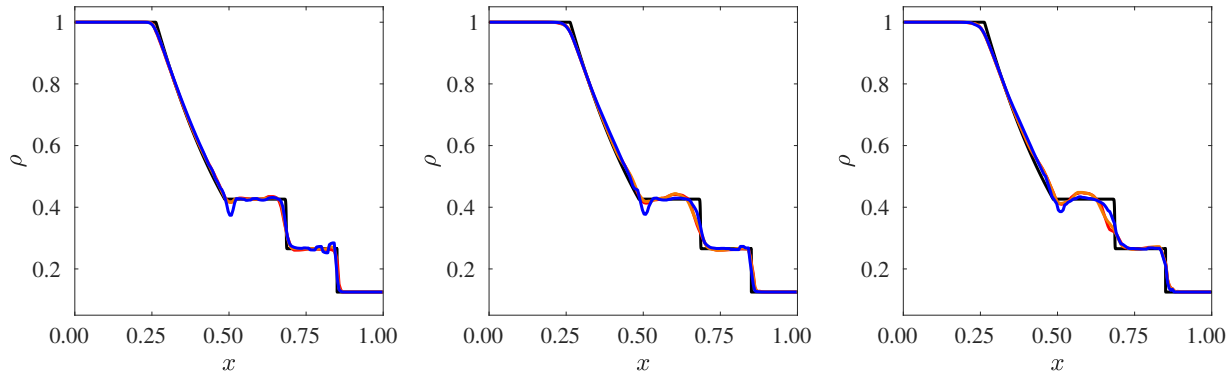


Figure 4.9: **Test 3** (Linear advection of a Sine wave with limiter): Mesh refinement study in cell-average error. Gray lines illustrate convergence rates, denoted m.
Symbol Key: **conDG:**○ ; **ICBM:**□ ; **ICBL:**▷ ; **cenDG:** ◊.

domain is $x \in [0, 1]$ and the initial condition is defined in the primitive variables as follows:

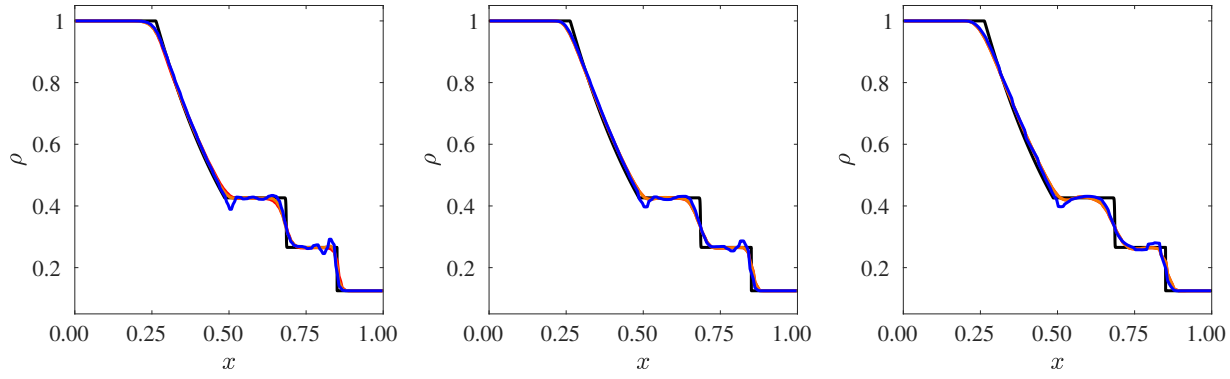
$$\begin{bmatrix} \rho & u & p \end{bmatrix}^T = \left\{ \begin{array}{ll} \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T & \text{for } x < \frac{1}{2} \\ \begin{bmatrix} 0.125 & 0 & 0.1 \end{bmatrix}^T & \text{for } x \geq \frac{1}{2} \end{array} \right\}. \tag{4.13}$$

The SLAU2 flux [58] is employed as the Riemann solver for the interface fluxes. The boundaries are zero-penetration walls. Explicit time integration is employed via the standard four-stage RK4 scheme. Slope limiting is necessary to maintain stability on this problem; the slope limiting scheme affects the quality of the results, so we employ two slope limiting approaches. The first approach is the moment-based limiter of Krivodonova [64], abbreviated MK. The second approach is the hierarchical reconstruction scheme of Henry de Frahan et al. [44], abbreviated HR. No sensor is applied; instead, the limiter is applied to every element after every residual evaluation and at the



(a) $p = 1$, $M = 96$. (b) $p = 2$, $M = 64$. (c) $p = 3$, $M = 48$.

Figure 4.10: **Test 4B** (Sod problem, MK Limiter): Density profile at $t = 0.2$ in Sod problem with MK limiter. Color Key: **Reference:** —— ; **conDG:** —— ; **ICBM:**—— ; **ICBL:** ——.



(a) $p = 1$, $M = 96$. (b) $p = 2$, $M = 64$. (c) $p = 3$, $M = 48$.

Figure 4.11: **Test 4A** (Sod problem, HR Limiter): Density profile at $t = 0.2$ in Sod problem with HR limiter. Color Key: **Reference:** —— ; **conDG:** —— ; **ICBM:**—— ; **ICBL:** ——.

end of each explicit timestep. The flow is simulated to $t_{final} = 0.2$. The exact solution consists of three traveling waves: the left-running rarefaction, the right-running contact discontinuity, and the right-running shock wave. Density profiles at $t = t_{final}$ for $p \in \{1, 2, 3\}$, using 192 DOFs per field variable, are shown in Figure 4.10 (using the moment-based limiter) and Figure 4.11 (using the hierarchical reconstruction limiter). Regardless of the limiting scheme, the Modal ICB and conventional DG results are nearly identical at a given p; the Lagrange ICB scheme exhibits sharper profiles at the physical discontinuities. However, the Lagrange ICB scheme tends to exhibit nonphysical solution features in the wake of the shock wave, so the sharper feature resolution

comes at a cost. Note that for a fixed DOF count, the higher p results are more smeared than the $p = 1$ case. While this effect is not always apparent in slope limiting schemes, it underscores the importance of optimizing scheme performance at a given p, since some cases demand h-refinement over p-refinement.

4.4.5 Additional Testing and Limitation

Additional testing of the Modal and Lagrange ICB schemes on 2D and 3D Cartesian elements (data not shown) demonstrated that they maintain the same accuracy advantages over the conventional upwind approach as in the 1D case. The ICB schemes are also stable on non-Cartesian quadrilateral elements when the derivative-based recovery implementation is employed. However, we were unable to identify stable and sufficiently accurate applications of the biased recovery approach on 2D simplex meshes. Many different configurations of the recovery basis $\boldsymbol{\psi}$ were tested, and in the rare cases where a particular configuration appeared stable, the ICB scheme appeared less accurate than the conventional upwind DG method. To maximize the versatility of the new ICB advection schemes, the deficiency on simplex elements needs to be addressed. However, the difficulty with simplex elements does not render the ICB schemes completely useless on unstructured meshes, as a single simplex element can always be split into three smaller quadrilateral elements.

4.5 Computational Cost

In one space dimension, the DG spatial discretization is typically implemented with Lagrange polynomials as the basis functions and the Lobatto points taken as the solution points, where the DOFs are stored. On quads in 2D and hexes in 3D, a tensor product grid of the 1D Lobatto points is applied, and that organization is employed in our DG code. Thus, the calculation of the numerical solution U^h along element boundaries is a relatively inexpensive process, as most of an element's basis functions are zero along a given element face. In contrast, the ICB schemes require the use of the biased recovery operation along interfaces, making them more computationally expensive

than the conventional upwind DG scheme. The computational cost difference is explored in this section, and a simple 2D test case is submitted as evidence that the improved accuracy of the ICB schemes is worth the extra computational work.

Table 4.5 describes the flop count per element necessary to calculate the spatial residual in the discretization of the scalar advection equation in 1D, 2D (on quads), and 3D (on hexes) for a given solution order p. Both the conventional DG and ICB schemes are included; the costs of all ICB variants are identical. The numerical flop counts per DOF per residual evaluation in the 2D and 3D cases (using the formulae from Table 4.5) are presented in Table 4.6. The severe increase in flop count per DOF as the number of spatial dimensions increases is due to the coupling of a single DOF's update scheme to all of the other DOFs in the same element. This severe increase could be mitigated by the line-based optimization proposed by Persson [83], but such a level of computational optimization is beyond the scope of this thesis.

Regarding the difference between the conventional DG and ICB approaches, the jump in flop

Table 4.5: Floating point operations per element per residual evaluation to solve the scalar advection equation. $K = (p+1)^{N_D}$, Q_V is number of interior quadrature points, Q_S is number of quadrature points on each interface, $N_V = 2N_D$ is number of sides per element, N_D is number of spatial dimensions.

Calculation	**conDG Flops**	**ICB Flops**
U_L or U_R, interface quadr. points	$N_V \times Q_S \times 4(p+1)$	$N_V \times Q_S \times 4K$
U^h, interior quadr. points	$Q_V \times 2K$	$Q_V \times 2K$
$\mathcal{F}(U^h)$, interior quadr. points	$N_D \times Q_V$	$N_D \times Q_V$
$\tilde{\mathcal{F}}(U_L, U_R, \mathbf{n})$, interface quadr. points	$N_D \times N_V \times Q_S$	$N_D \times N_V \times Q_S$
$\frac{d}{dt}\hat{\mathbf{U}}$, given $\mathcal{F}$ distribution	$K \times (2N_D \times Q_V + N_V \times 2Q_S + 3K)$	$K \times (2N_D \times Q_V + N_V \times 2Q_S + 3K)$

Table 4.6: Flops per DOF per residual evaluation to solve the scalar advection equation with $Q_V = K$, $Q_S = (p+1)^{N_D-1}$.

	$N_D = 2$			$N_D = 3$		
p	**Flops (conDG)**	**Flops (ICB)**	$\frac{\textbf{ICB Flops}}{\textbf{conDG Flops}}$	**Flops (conDG)**	**Flops (ICB)**	$\frac{\textbf{ICB Flops}}{\textbf{conDG Flops}}$
1	66	90	1.364	172	244	1.419
2	117.7	157.7	1.340	450	630	1.400
3	188	244	1.298	951.5	1287.5	1.353
4	276.6	348.6	1.260	1741.6	2281.6	1.310
5	383.3	471.3	1.230	2886	3678	1.274

count can exceed 40% in certain scenarios, demanding a quantitative efficiency study. Towards this end, we conduct a pair of test cases (5A and 5B) with the 2D scalar advection equation with $v_x = \pi$ and $v_y = \frac{\pi}{2}$:

$$\frac{\partial}{\partial t} U + \nabla \cdot \mathcal{F} = 0 \quad \text{where} \quad \mathcal{F} = (\pi U, \frac{\pi}{2} U). \tag{4.14}$$

The spatial domain is the 2π square with periodic boundary conditions, the initial condition is $U(x, y, 0) = \sin(x)\sin(y)$, and the problem is simulated to $t_{final} = 4$ so that the wave profile travels two periods in the x direction and one period in the y direction. The mesh resolutions are given in Table 4.7. The maximum allowable CFL number for each of the three advection schemes for each p when using explicit RK4 time integration has been determined experimentally; the results are given in Table 4.8. The constraint for stability was that the numerical solution remain stable for all mesh resolutions listed in Table 4.7. These CFL numbers are problem-dependent, being influenced by the temporal discretization scheme (explicit RK4 in this case) and the eigenvalue spectrum of the spatial discretization scheme under the specific velocity profile.

Two mesh refinement studies are conducted. For case 5A, each simulation uses the CFL number corresponding to the conventional upwind DG scheme. Thus, for a given solution order p and element count M, the three DG schemes use the same timestep size and thus require the same number of timesteps to run to completion. In case 5B, each spatial discretization instead uses the maximum allowable CFL number, taken from Table 4.8, such that the ICB schemes (which have smaller spectral radii) require fewer timesteps than the conventional DG scheme to run to completion. The computational wall times are listed in tables 4.9, 4.10, and 4.11; all jobs were run on a single 1.9 GHz Intel Xeon(R) CPU E5-2609 v3 processor. The global L_2 error for each simulation in case 5A is plotted against the wall time in Figure 4.12. The error's order of convergence with respect to the mesh resolution is limited to 4 because the explicit RK4 scheme is employed for the temporal discretization. The cost grows as h^{-3}, so the order of convergence in the error versus wall time plots is limited to $\frac{4}{3}$. In test case 5A, while the ICB schemes are more expensive than conventional DG for a given mesh resolution, the benefit in accuracy outweighs the wall time penalty. Figure 4.13 shows the error versus the wall time for test case 5B. Note that due to the improvement

in timestep size, the ICB schemes require less wall time than the uDG scheme on a given mesh. In the $p = 3$ case, since the primary source of error is the temporal discretization and the Lagrange ICB scheme allows substantially larger timestep sizes than the other two schemes, it exhibits larger errors, but it maintains the same convergence rate.

In the 2D case, the maximum stable CFL number depends on the angle of the velocity vector. In addition to identifying the maximum allowable CFL number with $\boldsymbol{V} = (\pi, \pi/2)$, we identified the maximum CFL number for two additional cases: $\boldsymbol{V} = (\pi, 0)$ and $\boldsymbol{V} = (\pi, \pi)$. The resulting CFL limits are tabulated in Table 4.12 and Table 4.13.

Table 4.7: Mesh resolutions for test cases 5A, 5B.

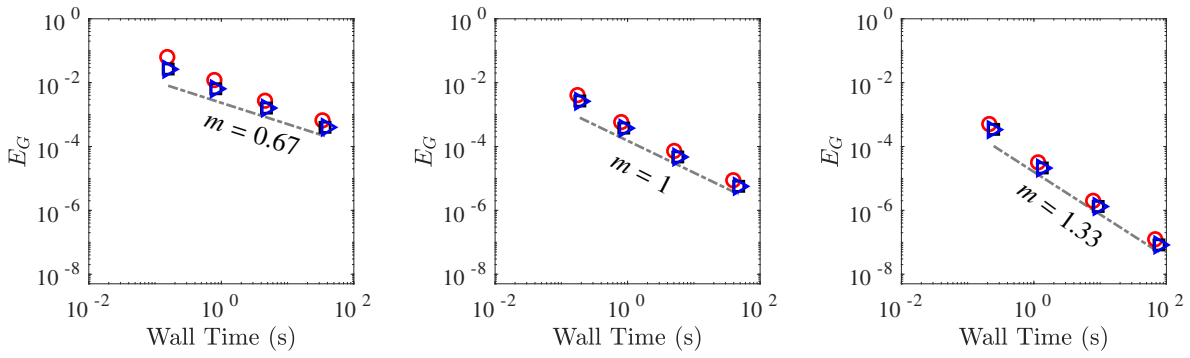
p	**Elements per direction (R)**	**Element Count (M)**	**DOF count ($nDOF$)**
1	{16 , 32 , 64 , 128}	{256 , 1024 , 4096 , 16384}	$\{32^2, 64^2, 128^2, 256^2\}$
2	{11 , 21 , 42 , 85}	{121 , 441 , 1764 , 7225}	$\{33^2, 63^2, 126^2, 255^2\}$
3	{8 , 16 , 32 , 64}	{64 , 256 , 1024 , 4096 }	$\{32^2, 64^2, 128^2, 256^2\}$

Table 4.8: Maximum stable CFL numbers for test case 5B: scalar advection on 2D Cartesian grid. The timestep size is set as $\Delta t = CFL \times \frac{h}{|V|}$, where h is the element width and $\boldsymbol{V} = (v_x, v_y)$ is the velocity vector. The time integration scheme is explicit RK4.

p	CFL_{conDG}	CFL_{ICBM}	CFL_{ICBL}
1	0.34	0.52	0.73
2	0.17	0.23	0.39
3	0.10	0.13	0.26

Table 4.9: Wall time (seconds) for case 5A ($CFL = CFL_{conDG}$) and case 5B ($CFL = CFL_{Max}$) with $p = 1$.

Case:	5A : CFL_{conDG}				5B : CFL_{Max}			
Resolution:	$R = 16$	$R = 32$	$R = 64$	$R = 128$	$R = 16$	$R = 32$	$R = 64$	$R = 128$
conDG wall time:	0.077	0.476	3.70	30.2	0.077	0.476	3.70	30.2
Modal ICB wall time:	0.079	0.509	3.88	34.3	0.056	0.341	2.60	22.2
Lagrange ICB wall time:	0.081	0.508	3.89	33.8	0.043	0.247	1.86	15.9



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 4.12: Error versus computational wall time (in seconds) for test case 5A (2D scalar advection on Cartesian elements). Time integration is explicit RK4; timestep size set by CFL_{conDG} for all p for each scheme. Symbol Key: conDG:○ ; ICBM:□ ; ICBL: ▷.

Table 4.10: Wall time (seconds) for case 5A ($CFL = CFL_{conDG}$) and case 5B ($CFL = CFL_{Max}$) with $p = 2$.

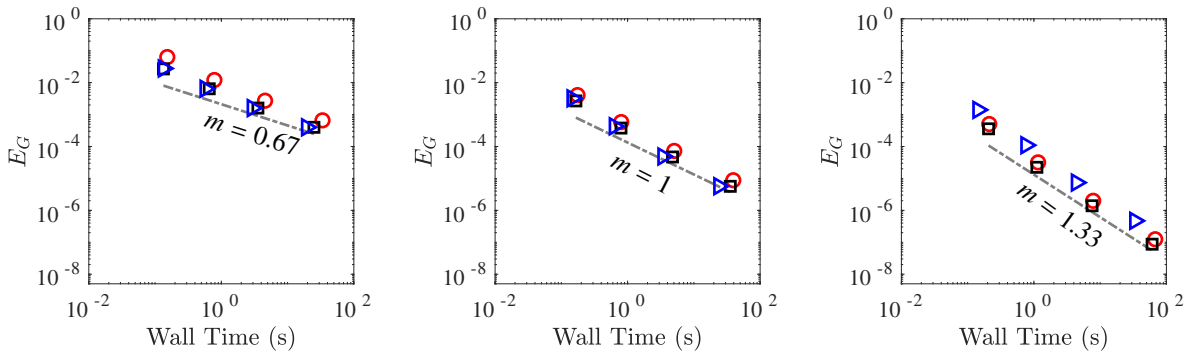
Case:	5A : CFL_{conDG}				5B : CFL_{Max}			
Resolution:	$R = 11$	$R = 21$	$R = 42$	$R = 85$	$R = 11$	$R = 21$	$R = 42$	$R = 85$
conDG wall time:	0.089	0.548	4.23	37.3	0.089	0.548	4.23	37.3
Modal ICB wall time:	0.108	0.629	4.83	45.2	0.086	0.481	3.66	34.2
Lagrange ICB wall time:	0.109	0.629	4.99	45.8	0.051	0.284	2.16	20.0

Table 4.11: Wall time (seconds) for case 5A ($CFL = CFL_{conDG}$) and case 5B ($CFL = CFL_{Max}$) with $p = 3$.

Case:	5A : CFL_{conDG}				5B : CFL_{Max}			
Resolution:	$R = 8$	$R = 16$	$R = 32$	$R = 64$	$R = 8$	$R = 16$	$R = 32$	$R = 64$
conDG wall time:	0.125	0.897	7.02	66.0	0.125	0.897	7.02	66.0
Modal ICB wall time:	0.152	1.10	8.58	77.6	0.123	0.888	6.83	61.8
Lagrange ICB wall time:	0.151	1.10	8.64	77.9	0.067	0.464	3.71	32.6

Table 4.12: Maximum stable CFL numbers for scalar advection on 2D Cartesian grid with $\boldsymbol{V} = (\pi, 0)$. The timestep size is set as $\Delta t = CFL \times \frac{h}{|\boldsymbol{V}|}$, where h is the element width. The time integration scheme is explicit RK4.

$\boldsymbol{p}$	$\boldsymbol{CFL_{conDG}}$	$\boldsymbol{CFL_{ICBM}}$	$\boldsymbol{CFL_{ICBL}}$
1	0.47	0.70	0.99
2	0.24	0.31	0.54
3	0.14	0.18	0.35



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 4.13: Error versus computational wall time (in seconds) for test case 5B (2D scalar advection on Cartesian elements). Time integration is explicit RK4; timestep size set by CFL_{max} for all p for each scheme. Symbol Key: conDG:○ ; ICBM:□ ; ICBL: ▷.

4.6 Chapter Conclusion

Three DG schemes for advection, in addition to the conventional upwind DG method and the less popular central method, are summarized, analyzed, and tested. The three schemes covered are the Modal ICB scheme, which was introduced by Khieu & Johnsen [56], and the new Lagrange ICB and Cut ICB variants. All of the ICB schemes employ a biased recovery operation to build interface solution approximations. In contrast, the conventional and central DG approaches use the limits of the discontinuous polynomial U^h. The use of the biased recovery operation allows the ICB schemes to achieve an accuracy advantage over the conventional DG approach while maintaining a compact computational stencil and retaining the dissipation mechanism of the conventional upwind discretization. The Lagrange ICB scheme is less dissipative than Modal ICB and

Table 4.13: Maximum stable CFL numbers for scalar advection on 2D Cartesian grid with $\boldsymbol{V} = (\pi, \pi)$. The timestep size is set as $\Delta t = CFL \times \frac{h}{|\boldsymbol{V}|}$, where h is the element width. The time integration scheme is explicit RK4.

p	CFL_{conDG}	CFL_{ICBM}	CFL_{ICBL}
1	0.33	0.49	0.70
2	0.16	0.22	0.38
3	0.10	0.12	0.24

achieves significantly better resolving efficiencies. The Cut ICB scheme is obtained by cutting the top derivative correction term from the Modal ICB scheme and is less accurate than either Modal ICB or Lagrange ICB. All three of the ICB schemes achieve optimal convergence in the global L_2 norm. The Modal and Lagrange ICB schemes consistently achieve better performance than the conventional DG scheme for linear advection; additionally, when paired with an appropriate slope limiting scheme, they maintained stability in the classical Sod shock tube problem. The next chapter reveals the exceptional advection-diffusion schemes that result from combining the ICB approaches of this chapter with the Compact Gradient Recovery scheme of the previous chapter.

CHAPTER 5

Recovery-assisted DG methods for Advection-Diffusion

5.1 Chapter Overview

The goal of this dissertation is the construction of a DG scheme for advection-diffusion problems that improves upon the conventional DG approach with regard to accuracy. Up to this point, the advection and diffusion discretizations were analyzed separately. In this chapter, the full advection-diffusion discretizations resulting from the combination of the ICB schemes of Chapter 4 (which use the biased recovery operation to improve the accuracy of the advective interface fluxes) and the Compact Gradient Recovery (CGR) scheme of Chapter 3 (which uses the full-order recovery operator to improve interface approximations in the mixed formulation) are analyzed in detail. There are three schemes of interest, all carrying the designation *Recovery-assisted DG* (abbreviated RAD), reflecting the pivotal role of the recovery operator in the discretization. One would expect that compared to a conventional DG discretization, the combination of ICB (for advective terms) and CGR (for diffusive terms) would yield an exceptional advection-diffusion scheme; we are now prepared to explore this claim. Fourier analysis is performed on the new RAD schemes to evaluate performance over a broad range of Peclet numbers. Then, the schemes are applied to solve a set of advection-diffusion test problems, starting with the linear case and working up to the nonlinear compressible Navier-Stokes equations in 2D and 3D.

5.1.1 Novelty and Articles

Previous to our doctoral studies, the interface-centered binary (ICB) reconstruction schemes had not been applied within an advection-diffusion discretization. This chapter represents the first pairing of any ICB approach with an appropriate diffusion scheme (namely our CGR scheme) alongside appropriate analysis of the resulting advection-diffusion discretization. Additionally, to our knowledge this is the only study where the wavenumber resolution analysis of Watkins et al. [112] is applied across a spectrum of Peclet numbers rather than just one or two Peclet numbers.

The material of this chapter appears in one AIAA conference manuscript and one in-preparation article:

- P. E. Johnson & E. Johnsen, *A Compact Discontinuous Galerkin Method for Advection-Diffusion Problems*, AIAA Paper 2018-1091.
- P. E. Johnson, L. H. Khieu, & E. Johnsen, *A Compact Recovery-Assisted Discontinuous Galerkin Method for the Compressible Navier-Stokes Equations*, in preparation.

5.1.2 Usage of Recovery

The RAD schemes make use of the full-order recovery operator to handle the diffusive terms and the biased recovery operator to handle the advective terms. These recovery operators are applied exclusively in the derivative-based implementation of Section 2.6.6. The derivative-based recovery implementation is not effective on simplex elements, so with regard to unstructured meshes, this chapter is restricted to unstructured quadrilateral elements as opposed to unstructured simplex elements.

5.2 The Discontinuous Galerkin Method for Advection-Diffusion

We re-state the DG weak form to allow lucid explanation of how the advection and diffusion schemes work together. The spatial domain $\mathbf{\Omega}$ is partitioned into a set of M non-overlapping

elements, such that $\cap_{e=1}^{M}\Omega_e = \emptyset$ and $\cup_{e=1}^{M}\overline{\Omega}_e = \overline{\boldsymbol{\Omega}}$, where $\overline{\Omega}_e$ denotes the closure, i.e., $\overline{\Omega}_e = \Omega_e \cup \partial\Omega_e$. Over each element, the solution is approximated as a degree-p polynomial U^h, cast as a linear combination of K basis functions (denoted ϕ_e^k) and K DOFs (denoted $\hat{U}_e^k$). Additionally, the auxiliary variable $\boldsymbol{\sigma}$ is built in the DG solution space:

$$U^h(\boldsymbol{x} \in \Omega_e, t) = U_e^h(\boldsymbol{x}, t) = \sum_{k=0}^{K-1} \phi_e^k(\boldsymbol{\xi}(\boldsymbol{x}))\hat{U}_e^k(t) \qquad , \qquad \boldsymbol{\sigma}(\boldsymbol{x} \in \Omega_e) = \boldsymbol{\sigma_e}(\boldsymbol{x}) = \sum_{k=0}^{K-1} \phi_e^k(\boldsymbol{\xi}(\boldsymbol{x}))\hat{\boldsymbol{\sigma}}_e^k. \tag{5.1}$$

Taking the Bubnov-Galerkin approach, the solution basis $\boldsymbol{\phi}$ (containing $K \times M$ members) forms the testing basis with which to build the auxiliary weak form (Eq. 5.2a), the DG weak form (Eq 5.2b), and the initial condition constraint (Eq 5.2c), each of which must be satisfied for every basis function $\phi_e^k \in \boldsymbol{\phi}$ and over every element $\Omega_e \in \boldsymbol{\Omega}$:

$$\underbrace{\int_{\Omega_e} \phi_e^k \boldsymbol{\sigma_e} d\boldsymbol{x}}_{\text{A0}} = \underbrace{\int_{\partial\Omega_e} \phi_e^{k-} \tilde{U} \mathbf{n}_\mathbf{e}^- d\boldsymbol{s}}_{\text{A1}} - \underbrace{\int_{\Omega_e} U_e^h \nabla\phi_e^k d\boldsymbol{x}}_{\text{A2}}, \tag{5.2a}$$

$$\underbrace{\int_{\Omega_e} \phi_e^k \frac{\partial}{\partial t} U_e^h d\boldsymbol{x}}_{\text{B0}} = -\underbrace{\int_{\partial\Omega_e} \phi_e^{k-} (\tilde{\boldsymbol{\mathcal{F}}} \cdot \mathbf{n}_\mathbf{e}^-) d\boldsymbol{s}}_{\text{B1}} + \underbrace{\int_{\Omega_e} \nabla\phi_e^k \cdot \boldsymbol{\mathcal{F}} d\boldsymbol{x}}_{\text{B2}} + \underbrace{\int_{\partial\Omega_e} \phi_e^{k-} (\tilde{\boldsymbol{\mathcal{G}}} \cdot \mathbf{n}_\mathbf{e}^-) d\boldsymbol{s}}_{\text{B3}} - \underbrace{\int_{\Omega_e} \nabla\phi_e^k \cdot \boldsymbol{\mathcal{G}} d\boldsymbol{x}}_{\text{B4}}, \tag{5.2b}$$

$$\int_{\Omega_e} \phi_e^k U_e^h(\boldsymbol{x}, 0) d\boldsymbol{x} = \int_{\Omega_e} \phi_e^k U_{IC}(\boldsymbol{x}) d\boldsymbol{x}. \tag{5.2c}$$

Along any shared interface, both of the elements sharing the interface must use the same values of U (in term A1), $\boldsymbol{\mathcal{F}}$ (in term B1), and $\boldsymbol{\mathcal{G}}$ (in term B3); the tilde mark above an interface quantity (for example, $\tilde{U}$ instead of U) indicates that a common interface value is chosen from the multi-valued DG approximation. As demonstrated in previous chapters, the strategies for calculating these interface quantities distinguish DG schemes from eachother; the advection scheme (either conventional upwind DG or ICB) determines how $\tilde{\boldsymbol{\mathcal{F}}}$ is calculated and the diffusion scheme determines how $\tilde{\boldsymbol{\mathcal{G}}}$ and $\tilde{U}$ are calculated. Along $\boldsymbol{\partial\Omega}$, the interface quantities ($\tilde{\boldsymbol{\mathcal{F}}}$, $\tilde{\boldsymbol{\mathcal{G}}}$, $\tilde{U}$) provide the

means to enforce boundary conditions. The update scheme consistent with Eq. (5.2) for the DOFs $\hat{\mathbf{U}}$ is given in Table 5.1; we take the semi-discrete approach, where the DG spatial discretization is used to calculate $\frac{d}{dt}\hat{\mathbf{U}}$ and the resulting system of ODEs is integrated forward in time.

5.3 Design of DG Schemes

A typical DG discretization for advection-diffusion is obtained by combining the conventional upwind DG approach for advection with the BR2 scheme [8] (which performs similarly to an optimal interior penalty approach [40]) for diffusion. Using the separate advection and diffusion schemes proposed in previous chapters, we form three additional advection-diffusion schemes: Recovery-assisted DG scheme 1, abbreviated RAD1, is formed by using the Modal ICB reconstruction for the advective fluxes and the Compact Gradient Recovery (CGR) scheme [50, 51] for diffusive terms.

Table 5.1: Procedure for updating the DG DOFs $\hat{\mathbf{U}}$, governed by Eq. 5.2, in a time-accurate simulation.

Step	**Action**
1	Along each interface in the domain, calculate $\tilde{U}$ based on the pair of elements that share the interface (for interior interfaces) or take $\tilde{U}$ from the Dirichlet condition (for boundary interfaces).
2	$\forall\Omega_e$, solve the auxiliary weak form (Eq. 5.2a) to obtain $\boldsymbol{\sigma_e}$. Then, use U_e^h and $\boldsymbol{\sigma_e}$ to calculate the viscous flux, $\boldsymbol{\mathcal{G}}$, at all volume quadrature points (term B4).
3	$\forall\Omega_e$, populate $\boldsymbol{\mathcal{F}}$ at all volume quadrature points (term B2) using the local DG polynomial U_e^h.
4	Calculate $\tilde{\boldsymbol{\mathcal{F}}}$ and $\tilde{\boldsymbol{\mathcal{G}}}$ in terms B1 and B3.
5	Solve for the time derivatives $\frac{d}{dt}\hat{\mathbf{U}}$ using each element's inverse mass matrix, with Gaussian quadrature applied to populate the integrals of terms A1, A2, B1, B2, B3, and B4.

Recovery-assisted DG scheme 2, abbreviated RAD2, is formed by instead using the Lagrange ICB reconstruction for the advective fluxes and the CGR scheme for the diffusive terms. Also listed is a fourth scheme, RAD3, which is like RAD1 except that it uses the Cut ICB modification of the Modal ICB reconstruction. These schemes are listed and summarized in Table 5.2 for the reader's convenience. The specific CGR configuration is the CGR-Heavy variant with $\chi = 2$; while CGR-Heavy is identical to the standard CGR scheme on Cartesian meshes, CGR-Heavy performs slightly better on non-uniform quadrilateral meshes (see test case 4 in Chapter 3).

Regarding the diffusive flux $\tilde{\mathcal{G}}$ along each interface, both the BR2 and CGR schemes use the semi-connected gradient polynomials to build an approximation for the gradient along the interface. Then, the common interface gradient, $\tilde{\boldsymbol{\sigma}}$, is applied to calculate the diffusive flux along the interface. See Table 3.3 in Section 3.4 for further information.

5.4 Fourier Analysis for Linear Advection-Diffusion

Fourier analysis is performed on the set of DG schemes for advection-diffusion (Table 5.2), focusing on the conventional DG scheme (conDG) and the first two Recovery-assisted schemes (RAD1, RAD2). The wavenumber resolution analysis employed in Section 4.3 is repeated here, except that the governing differential equation is the 1D advection-diffusion equation as opposed to the advection equation. The analysis process is almost identical to that of linear advection; we describe the necessary alterations here..

Table 5.2: Summary of advection-diffusion schemes in terms of individual advection (see Chapter 4) and diffusion (see Chapter 3) components.

Scheme	Abbreviation	Advection	Diffusion	$\tilde{\mathcal{F}}$	$\tilde{U}$
conventional DG	conDG	upwind DG	BR2	$Rie(U_A^h, U_B^h, \mathbf{n}_\mathbf{A}^-)$	$\frac{1}{2}(U_A^h + U_B^h)$
Recovery-assisted DG 1	RAD1	Modal ICB	CGR	$Rie(U_A^{ICBM}, U_B^{ICBM}, \mathbf{n}_\mathbf{A}^-)$	$\mathcal{R}(U_A^h, U_B^h)$
Recovery-assisted DG 2	RAD2	Lagrange ICB	CGR	$Rie(U_A^{ICBL}, U_B^{ICBL}, \mathbf{n}_\mathbf{A}^-)$	$\mathcal{R}(U_A^h, U_B^h)$
Recovery-assisted DG 3	RAD3	Cut ICB	CGR	$Rie(U_A^{ICBC}, U_B^{ICBC}, \mathbf{n}_\mathbf{A}^-)$	$\mathcal{R}(U_A^h, U_B^h)$

Take the governing equation to be the 1D linear advection-diffusion equation,

$$\frac{\partial U}{\partial t} = -\frac{\partial}{\partial x}(\mathcal{F} - \mathcal{G}) \quad \text{with} \quad \mathcal{F} = aU \quad \text{and} \quad \mathcal{G} = \mu \frac{\partial}{\partial x} U, \tag{5.3}$$

where the advection speed a and diffusivity μ are constants. Let the domain be spatially periodic (such that boundary conditions can be neglected) and partitioned into elements of uniform width h. The DOFs are assumed to adhere to the following relation:

$$\hat{\mathbf{U}}_{\mathbf{e+j}} = \exp(i\omega j)\hat{\mathbf{U}}_{\mathbf{e}}. \tag{5.4}$$

The analysis must allow for different Peclet numbers; define the element-wise Peclet number as

$$Pe_h = \frac{ah}{\mu}. \tag{5.5}$$

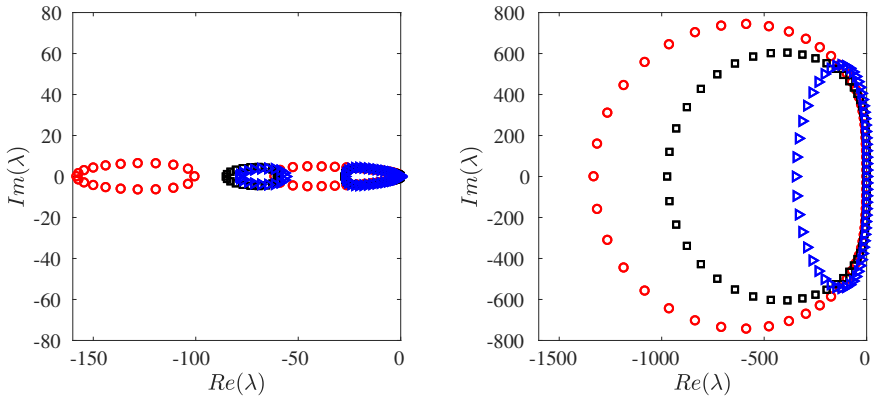
The advection-dominated regime is $Pe_h \to \infty$ and the diffusion-dominated regime is $Pe_h \to 0$. For a given choice of advection and diffusion schemes, let $\mathcal{D}^a$ denote the differentiation matrices of the advection scheme and let $\mathcal{D}^d$ denote the differentiation matrices of the diffusion scheme. For a given Peclet number (Pe_h) and wavenumber (ω), the differentiation matrices are combined to form the single update matrix:

$$\mathcal{A}(Pe_h, \omega) = \sum_{j=-1}^{j=1} \exp(i\omega j)(-Pe_h \mathcal{D}^a_{e+j} + \mathcal{D}^d_{e+j}), \tag{5.6}$$

such that by taking Eq. (5.4) into account, the update scheme is recast as a system of $p+1$ ordinary differential equations:

$$\frac{d}{dt}\hat{\mathbf{U}}_{\mathbf{e}} = \frac{\mu}{h^2}\mathcal{A}(Pe_h, \omega)\hat{\mathbf{U}}_{\mathbf{e}}. \tag{5.7}$$

The 1D advection-diffusion discretization is linearly stable if all eigenvalues of $\mathcal{A}$ are non-positive; this property holds for all schemes explored in this chapter. The spectral radius ρ_s is defined as the maximum magnitude of any eigenvalue of $\mathcal{A}$ for a fixed Pe_h and any possible ω. Keeping the



(a) $Pe_h = 1$. (b) $Pe_h = 100$.

Figure 5.1: Eigenvalue spectrums ($\omega \in [0, 4\pi]$) of various DG schemes with $p = 2$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

diffusivity fixed at $\mu = 1$ and varying the Peclet number, the allowable stable timestep scales as $\frac{h^2}{\rho_s}$. Sample eigenvalue spectrums are illustrated in Fig. 5.1, showing that the Recovery-assisted schemes (RAD1 and RAD2) yield smaller spectral radii than the conventional approach (conDG). Note that in the chosen setup, spectral radius ρ_s grows with Peclet number.

Similar to the advection analysis in Section 4.3, for a given Peclet number and discrete wavenumber ω, $\boldsymbol{\mathcal{A}}$ is diagonalized:

$$\boldsymbol{\mathcal{A}}(Pe_h, \omega) = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}. \tag{5.8}$$

For a given wavenumber ω, the initial condition in Ω_e, denoted $\hat{\mathbf{U}}(\omega, 0)$, is recast in terms of the eigenvectors:

$$\boldsymbol{\beta}(\omega) = \boldsymbol{V}^{-1}\hat{\mathbf{U}}(\omega, 0). \tag{5.9}$$

The exact eigenvalue corresponding to the linear advection-diffusion equation is $\lambda^{ex}(\omega, Pe_h) = -iPe_h\omega - \omega^2$. The error growth rate is now a function of both the wavemumber ω and the Peclet number:

$$\mathcal{E}(Pe_h, \omega) = \frac{1}{\sqrt{p+1}} \sum_{n=1}^{p+1} |\lambda_n(\omega) - \lambda^{ex}(\omega, Pe_h)|\, |\beta_n(\omega)|, \tag{5.10}$$

where $\beta_n(\omega)$ is the n^{th} entry of $\boldsymbol{\beta}(\omega)$ and $\lambda_n(\omega)$ is the corresponding n^{th} eigenvalue of $\boldsymbol{\mathcal{A}}(Pe_h, \omega)$.

As before, given some tolerance ϵ, define the maximum resolved wavenumber ω_f as a function

of Pe_h according to :

$$\mathcal{E}(Pe_h, \omega) \leq \epsilon \quad \text{whenever} \quad 0 \leq \omega \leq \omega_f(Pe_h). \tag{5.11}$$

Then, the resolving efficiency η is defined as follows:

$$\eta(Pe_h) = \frac{\omega_f(Pe_h)}{\pi(p+1)}. \tag{5.12}$$

For scheme comparison, the resolving efficiencies of conDG, RAD1, and RAD2 have been calculated over a discrete set of Peclet numbers spanning the domain $[10^{-2}, 10^3]$. The tolerance in Eq. (5.11) is scaled with the Peclet number:

$$\epsilon(Pe_h) = \begin{cases} \frac{1}{10} & \text{for } Pe_h < 1, \\ \frac{Pe_h}{10} & \text{for } Pe_h \geq 1. \end{cases} \tag{5.13}$$

This scaling is applied so that in the advective limit ($Pe_h \to \infty$), the resolving efficiency approaches a finite limit point matching the resolving efficiencies of Section 4.3. The resolving efficiencies of the conDG, RAD1, and RAD2 schemes with $p \in \{1, 2, 3, 4, 5\}$ are plotted in Figures 5.2 through 5.6. Additionally, Figure 5.7 shows the resolving efficiencies for $p = 4$ and $p = 5$ when the error tolerance is multiplied by 2, i.e., $\epsilon = 0.2$ for $Pe_h < 0$ and $\epsilon = \frac{Pe_h}{5}$ for $Pe_h \geq 0$. This extra plot is included to show that while the resolving eficiency grows with the error tolerance, the qualitative behavior of the schemes does not. The spectral radius, normalized by $\max(1, Pe_h)$, is plotted alongside the resolving efficiencies in Figures 5.2 through 5.6. This normalization is applied to force the plotted quantity to a finite limit point (as opposed to going to infinity) as $Pe_h \to \infty$.

The presented analysis shows that the Recovery-assisted DG schemes achieve superior wavenumber resolution compared to the conventional DG scheme. Between the Recovery-assisted schemes, RAD2 offers better performance than the RAD1 method; this behavior was expected based on the analysis of the ICBM and ICBL advection schemes in Section 4.3. For those interested in explicit time integration, the RAD2 scheme offers the smallest spectral radii (largest allowable timestep) while the conventional DG scheme has the largest spectral radii (smallest allowable timestep).

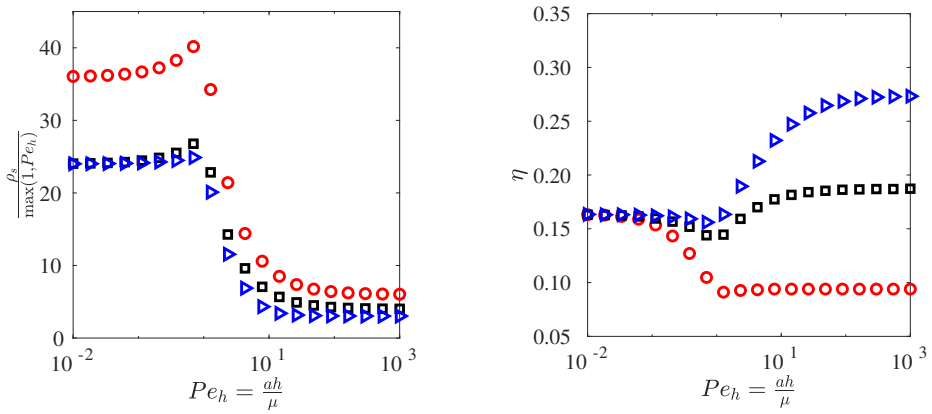


Figure 5.2: Normalized spectral radii (left) and resolving efficiencies (right) with $\epsilon = \frac{1}{10}\max(1, Pe_h)$ for $p = 1$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

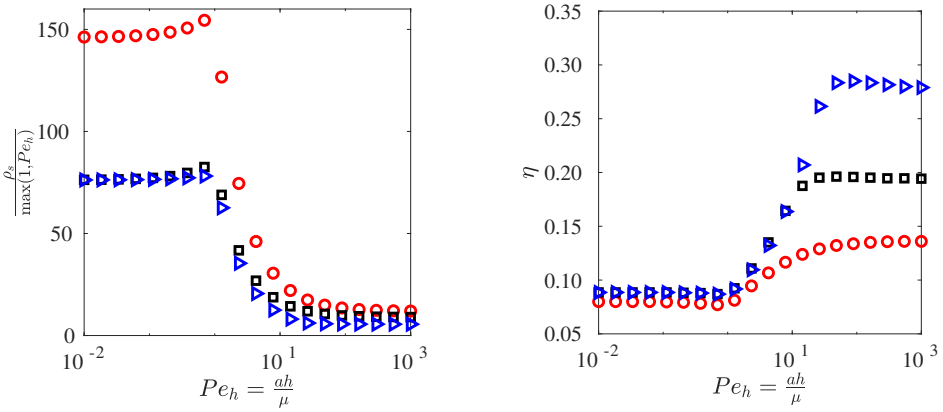


Figure 5.3: Normalized spectral radii (left) and resolving efficiencies (right) with $\epsilon = \frac{1}{10}\max(1, Pe_h)$ for $p = 2$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

The results of the analysis are unsurprising. Chapter 3 demonstrated that CGR is superior to BR2 for pure diffusion problems. Chapter 4 demonstrated that the ICB schemes are superior to the typical upwind DG approach for pure advection problems. Thus, one would expect that for advection-diffusion problems, superior performance could be obtained by pairing the ICB schemes with the CGR scheme. The presented analysis confirms this suspicion across a broad spectrum of Peclet numbers, from the diffusion-dominated regime to the advection-dominated regime.

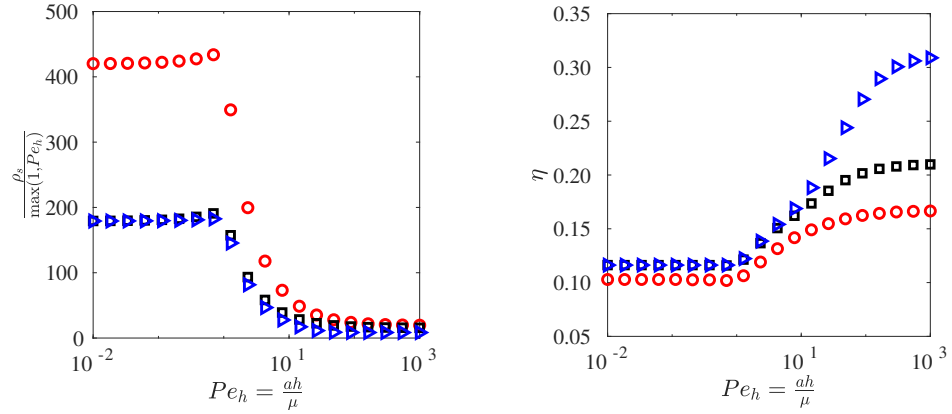


Figure 5.4: Normalized spectral radii (left) and resolving efficiencies (right) with $\epsilon = \frac{1}{10}\max(1, Pe_h)$ for $p = 3$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

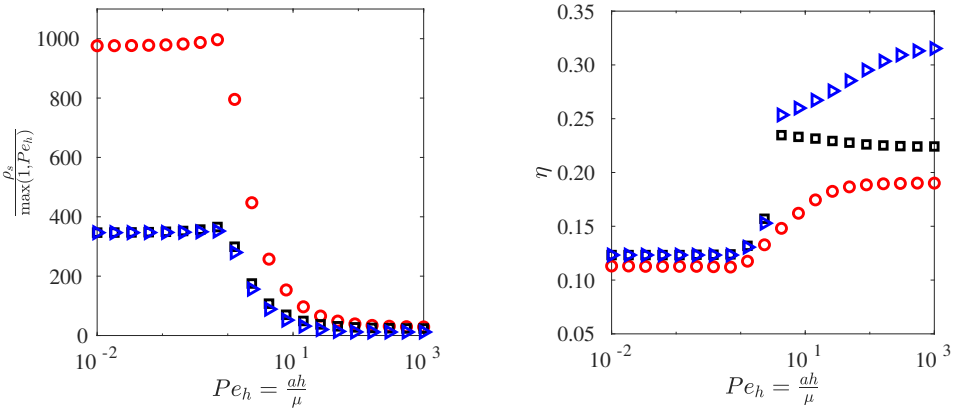


Figure 5.5: Normalized spectral radii (left) and resolving efficiencies (right) with $\epsilon = \frac{1}{10}\max(1, Pe_h)$ for $p = 4$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

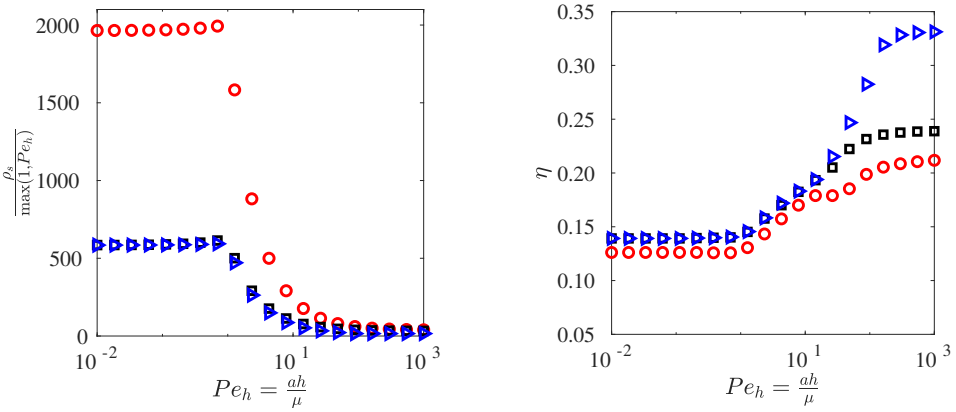


Figure 5.6: Normalized spectral radii (left) and resolving efficiencies (right) with $\epsilon = \frac{1}{10}\max(1, Pe_h)$ for $p = 5$. Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

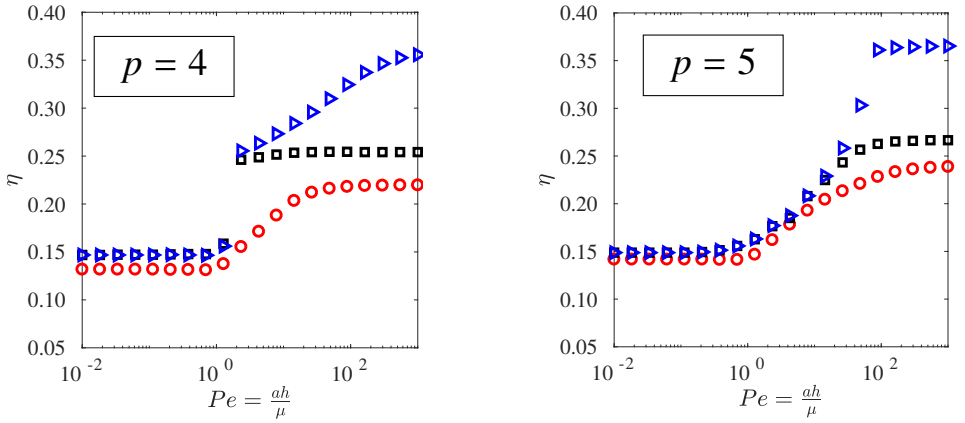


Figure 5.7: Resolving efficiencies with $\epsilon = \frac{2}{10}\max(1, Pe_h)$.
Symbol Key: **conDG:**○ ; **RAD1:**□ ; **RAD2:**▷.

5.5 Test Problems

5.5.1 Preliminaries

The RAD1 and RAD2 methods are evaluated through four test problems and compared to the conventional DG approach. The first problem is scalar advection-diffusion in 1D. Then, the dipole-wall collision problem studied by Orlandi [80] is simulated by discretization of the 2D compressible Navier-Stokes equations; the dipole-wall interaction, in addition to demonstrating our method's ability to handle a realistic boundary condition, will be used to gauge scheme performance on an unstructured quadrilateral mesh. The third problem is the Taylor-Green vortex on a uniform mesh, simulated through discretization of the 3D compressible Navier-Stokes equations. This flow tests a method's ability to capture the transition to turbulence under periodic boundary conditions. Then, compressible homogeneous isotropic turbulence (HIT) is simulated, again testing each method's ability to properly capture turbulent flow dynamics.

5.5.1.1 Time marching

Once initialized, the solution is advanced in time using explicit Runge-Kutta schemes. For the scalar problem (case 1), the eighth-order, thirteen-stage explicit RK algorithm of Prince & Dor-

mand [86] is chosen regardless of p to minimize temporal errors. For all other test cases, the standard fourth-order, four-stage method (RK4) is applied. This approach sacrifices temporal accuracy compared to the eighth-order approach but has been chosen to give an indication of how the Recovery-assisted discretizations will perform in practical flow physics simulations, where excessively high-order explicit RK schemes become impractical due to memory requirements. The time step is determined based on CFL and VNN constraints as described in Appendix E. In all test cases, regardless of the spatial discretization, the CFL and Von Neumann numbers are set according to the upwind DG and BR2 approaches, respectively. By applying the same timestep sizes to the conventional and RAD discretizations, we ensure honest comparison of the spatial discretizations themselves.

5.5.1.2 Implementation

The RAD spatial discretizations have been implemented alongside the conventional DG approach in a DG code developed at the University of Michigan, now known as the Recovery-assisted Michigan DG code (RAMdG). Quadrature for both the mass matrix and the residuals in the DG weak form (Eq. 5.2) is performed with $(p+1)^{N_D}$ Gaussian quadrature points over each element volume and $(p+1)^{N_D-1}$ Gaussian quadrature points over each interface. The basis functions (used for both the solution and test spaces) are the Lagrange polynomials; in the 1D case, the nodes for the Lagrange basis are the $p+1$ Lobatto points, while in the 2D and 3D cases, the nodes are distributed on a tensor product grid built from the 1D Lobatto points. The classical and biased reovery operators

Table 5.3: Summary of the test cases.

Test Case	Governing Equation(s)	Boundary Conditions	Mesh
1A	1D scalar advection-diffusion	periodic	uniform
1B	1D scalar advection-diffusion	periodic	non-uniform
2A	2D compressible Navier-Stokes	periodic & no-slip	stretched Cartesian
2B	2D compressible Navier-Stokes	periodic & no-slip	uniform Cartesian
2C	2D compressible Navier-Stokes	periodic & no-slip	unstructured quadrilateral
3	3D compressible Navier-Stokes	periodic	uniform Cartesian
4	3D compressible Navier-Stokes	periodic	uniform Cartesian

necessary for the CGR and ICB approaches, respectively, are implemented in the derivative-based form detailed in Section 2.6.6. Advective interface fluxes are handled via the upwind flux for case 1 and the SLAU2 Riemann solver [58] for cases 2, 3 and 4. Parallelization is achieved via MPI; the larger simulations in test cases 2 and 3 were executed on the Conflux cluster at the University of Michigan and the Comet cluster of the XSEDE program. The software Gmsh [38] is used to generate all meshes.

5.5.2 Test case 1: Scalar advection-diffusion in 1D

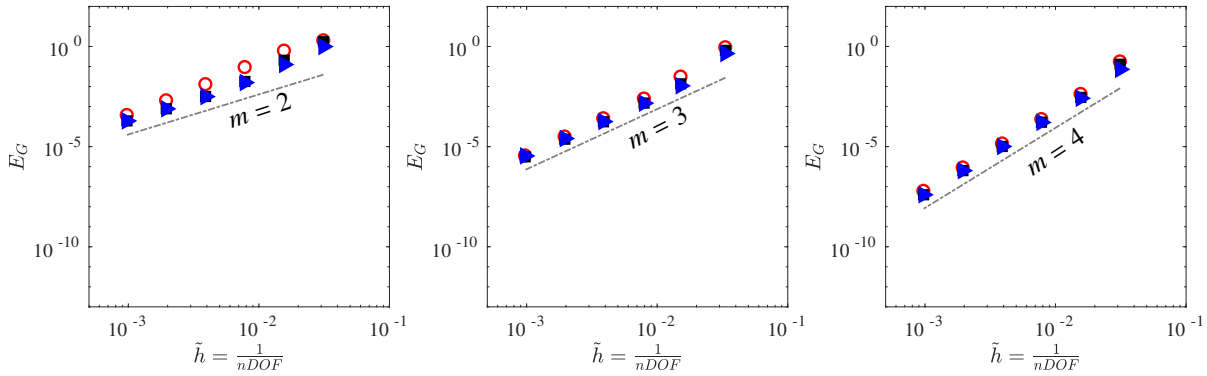
We discretize the 1D scalar advection-diffusion equation (Eq. 2.5) with $a = \pi$ and $\mu = \pi/100$. While this equation is not particularly exciting, it serves as an important check for whether or not the RAD schemes achieve optimal convergence in the global L_2 norm. The spatial domain is $x \in [0, 8\pi]$; periodicity is enforced along the left and right boundaries. The initial condition is $U(x, 0) = \sin(x)$ and the exact time-accurate solution is $U(x, t) = \sin(x - at)e^{-\mu t}$, such that four wavelengths are contained in the spatial domain. Once initialized in accordance with Eq. (5.2c), the solution is marched forward in time to $t_{final} = 16$ (two translational periods).

The conDG, RAD1, and RAD2 discretizations are tested on both uniform grids (case 1A) and a non-uniform grid (case 1B). We first describe the uniform grid test. Each scheme is applied with $p \in \{1, 2, 3\}$ on a series of grids to perform a mesh refinement study. The grid resolutions are given in Table 5.4 in terms of element count M and the total count of degrees of freedom, $nDOF = (p+1)(M)$. For each simulation, the error is measured at $t = t_{final}$ and quantified in terms of the global L_2 error E_G and the cell-average error E_{CA} (see Eq. 3.14).

The error in the global L_2 norm is plotted in Figure 5.8 against the characteristic mesh width ($\tilde{h}$) for all simulations. All three methods studied achieve the optimal convergence rate ($m = p+1$).

Table 5.4: Mesh resolutions for test case 1A.

p	Element Count (M)	Degrees of Freedom (nDOF)
1	{16 , 32 , 64 , 128 , 256 , 512}	{32 , 64 , 128 , 256 , 512 , 1024}
2	{10 , 22 , 42 , 86 , 170 , 342}	{30 , 66 , 126 , 258 , 510 , 1026}
3	{8 , 16 , 32 , 64 , 128 , 256}	{32 , 64 , 128 , 256 , 512 , 1024}

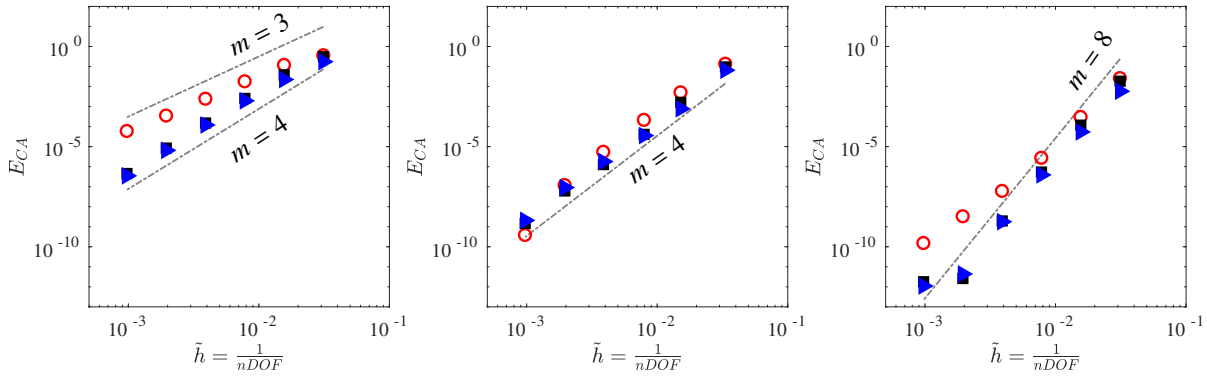


(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.8: Convergence study in global L2 error for test case 1A (scalar advection-diffusion on uniform grid). Symbol Key: conDG:○ ; RAD1:■ ; RAD2:►. Dotted gray lines denote approximate convergence rates m.

In the $p = 1$ case, our RAD schemes are clearly more accurate than the conDG approach, but for $p > 1$, the differences in the global error norm are negligible. Instead, Figure 5.9 plots the error in the cell-average error. The RAD schemes achieve 4^{th} order convergence for $p = 1$ while the conventional approach is limited to 3^{rd} order convergence. For $p = 2$, all methods give similar results in the cell-average error; this behavior is not unexpected, as previous analysis of the CGR method (Section 3.6) shows that it performs nearly identically to BR2 for $p = 2$. In the $p = 3$ case, while all discretizations exhibit 8^{th} order convergence on coarse meshes, the conventional approach is limited to 5^{th} order convergence on fine meshes while the new RAD discretizations maintain 8^{th} order convergence until machine precision becomes relevant on the finest mesh.

For test case 1B, the scalar advection-diffusion equation is instead solved on a non-uniform mesh, illustrated in Figure 5.10. The mesh is split into four zones. For $x \in [0, 2\pi]$ (zone 1), the mesh contains eight elements with stretching ratio $s = 1.2$; specifically, given a left element and a right element in this zone, we have $\frac{h_{right}}{h_{left}} = 1.2$. The next zone, $x \in [2\pi, 4\pi]$, contains sixteen uniform elements. The third zone, $x \in [4\pi, 6\pi]$ contains eight elements with stretching ratio $s = 1.25$ and the remaining fourth zone contains eight elements with $s = 0.8$. The smallest element is of width $h_{min} = 0.3808$ while the largest element is of width $h_{max} = 1.51$, corresponding to element Peclet numbers of $Pe_h = 38.08$ and $Pe_h = 151$, respectively. In addition to predicting



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.9: Convergence study in cell-average error for test case 1A (scalar advection-diffusion on uniform grid). Symbol Key: conDG:○ ; RAD1:■ ; RAD2:►. Dotted gray lines denote approximate convergence rates m.

the performance of the RAD discretizations on stretched mesh segments, this test features severe jumps in element width. We perform a p-refinement study; with the element count fixed, each spatial discretization is applied with $p \in \{1, 2, 3, 4, 5\}$. In addition to evaluating the schemes in terms of the global and cell-average error, we employ a functional error,

$$E_J = |\sum_{m=1}^{M} \int_{\Omega_m} \kappa(x)U(x)dx - \sum_{m=1}^{M} \int_{\Omega_m} \kappa(x)U^h(x)dx|, \tag{5.14}$$

where $\kappa(x) = \sin(1.1x)$ is the chosen kernel function. The errors from the p-refinement study are presented on semilog plots in Figure 5.11. Spectral convergence is observed in all three error norms as p is increased. While all methods perform similarly in the global L_2 error, the RAD schemes give superior performance in the cell-average and functional error norms. Figure 5.10 shows the results of all three discretizations for $p = 1$. The RAD approaches are minimally dissipative (with RAD1 being slightly more dissipative than RAD2) but stable on the non-uniform mesh, yielding dramatic improvement over the conventional DG approach.

We also applied a central scheme, where the BR2 scheme for diffusion is paired with the central DG scheme for advection. The central scheme produced unstable simulations on the non-uniform mesh, indicating a clear divide between the simple central scheme and the minimally dissipative

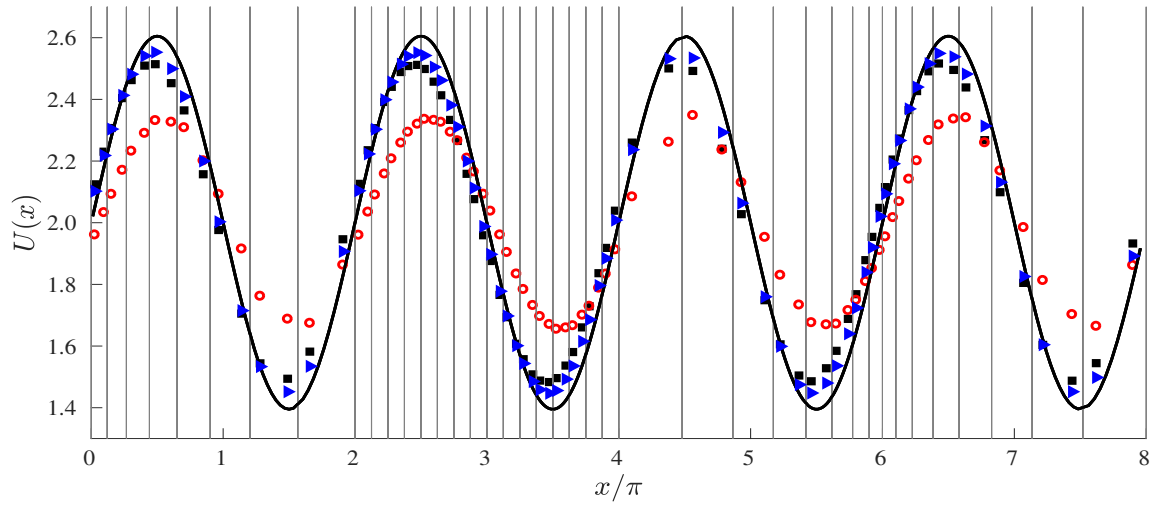


Figure 5.10: Results of test case 1B (scalar advection-diffusion on non-uniform grid) with $p = 1$ after two translational periods. Vertical lines indicate element-element interfaces. Symbol Key: Exact Solution: —— ; conDG:○ ; RAD1:■ ; RAD2:▶. The RAD schemes minimize numerical dissipation while maintaining stability on the non-uniform grid.

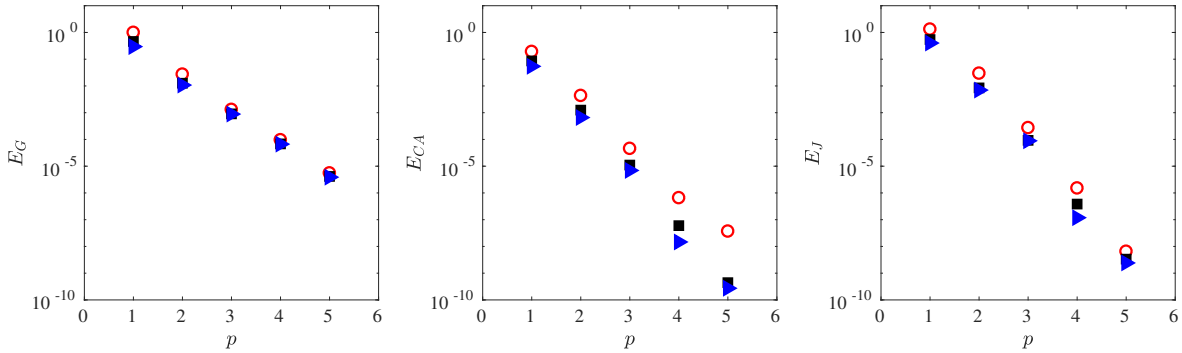


Figure 5.11: p-refinement study for test case 1B (scalar advection-diffusion on non-uniform grid); the polynomial order p is enriched while the grid (See Figure 5.10) is unchanged. Symbol Key: conDG:○ ; RAD1:■ ; RAD2:▶. The RAD schemes are superior for all p in all three error norms.

RAD approaches.

5.5.3 Test case 2: Dipole-Wall Interaction in 2D

The 2D dipole-wall interaction problem has previously been studied by Keetels et al. [54] to evaluate the performance of an adaptive wavelet method for the incompressible Navier-Stokes equations and by Chapelier et al. [21] to evaluate a conventional DG solver for the compressible Navier-

Stokes equations. The spatial domain is the bi-unit square: $(x, y) \in [-1, 1]^2$. The governing equations are the compressible Navier-Stokes equations (Eq. 2.13). The boundaries at $x = \pm 1$ are isothermal no-slip walls; spatial periodicity is enforced along the $y = \pm 1$ boundaries. The wall temperature is set to the freestream temperature according to the ideal gas law: $T_{wall} = T_\infty = \frac{p_\infty}{R_g \rho_\infty}$. The fluid parameters are as follows: $\gamma = 1.4$, $R_g = 287.15$, $Pr = 0.71$, and $\mu = 0.001$. The initial condition is a quiescent freestream with a pair of vortices (the dipole) near the center of the domain. The pair of vortices is initialized at $(x_1, y_1) = (0, -0.1)$ and $(x_2, y_2) = (0, 0.1)$. Each vortex is characterized by the vorticity magnitude ω_e, which is set to $\omega_e = 299.528385375226$ following the example of Keetels et al. [54]. Given the vortex radius, $r_0 = 0.1$, the initial velocity distribution is given by:

$$v_1(x, y, 0) = -\frac{\omega_e}{2}(y - y_1)e^{-(r_1/r_0)^2} + \frac{\omega_e}{2}(y - y_2)e^{-(r_2/r_0)^2}, \tag{5.15a}$$

$$v_2(x, y, 0) = \frac{\omega_e}{2}(x - x_1)e^{-(r_1/r_0)^2} - \frac{\omega_e}{2}(x - x_2)e^{-(r_2/r_0)^2}, \tag{5.15b}$$

where $r_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2}$ is the distance to the first vortex and $r_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2}$ is the distance to the second vortex. The initial density is set constant at $\rho_0 = \rho_\infty = 1$. Following the example of Chapelier et al. [21], the freestream pressure is set according to a Mach number constraint. The kinetic energy in the domain at $t = 0$ is approximately $KE(0) = 2$, which yields a characteristic velocity of $|\boldsymbol{V}| = \sqrt{KE(0)/2} = 1$. The initial pressure is set such that the Mach number of the characteristic velocity is $\frac{1}{100}$: specifically, $M_\infty = |\boldsymbol{V}|/\sqrt{\gamma p_\infty/\rho_\infty} = 0.01$. This constraint yields the freestream pressure, p_∞. Then, the initial pressure field is set to be compatible with the vortex dipole:

$$p_0(x, y) = p_\infty - (\frac{\omega_e r_0}{4})^2(\exp[-2(r_1/r_0)^2] + \exp[-2(r_2/r_0)^2]). \tag{5.16}$$

Following Keetels et al. [54], the Reynolds number is set according to the characteristic velocity and the half-width ($W = 1$) of the domain:

$$Re = \frac{\rho_\infty |\boldsymbol{V}| W}{\mu}. \tag{5.17}$$

The value of $\mu = 0.001$ has been chosen to achieve $Re = 1000$, matching the simulations performed by Keetels et al. [54] and Chapelier et al. [21]. With density, velocity, and pressure known over the entire domain at $t = 0$, the vector of conserved variables, $U = [\rho, \rho v_1, \rho v_2, E]^T$ is known. The DG DOFs are initialized according to Eq. (5.2c); this detail is important because if the DOFs are instead initialized based on nodal equivalence, there may be severe errors in the initial values for enstrophy and kinetic energy on coarse meshes.

Error quantification is performed by comparison to a reference simulation, run on a uniform mesh with $M = 384^2$ elements using the conventional DG discretization with $p = 3$. The development of the flow is illustrated in Figure 5.12; we plot vorticity contours from our reference simulation in the subdomain $(x, y) \in [-1, 0] \times [-\frac{1}{2}, \frac{1}{2}]$. Due to the cumulative velocity field created by the two vortices, the dipole is propelled towards the isothermal no-slip wall at $x = -1$. As the dipole approaches the wall, a thin boundary layer forms along the wall. During the dipole's collision with the wall, two new vortices are formed; the resulting four-vortex system is shown in Figure 5.12c at $t = 0.5$.

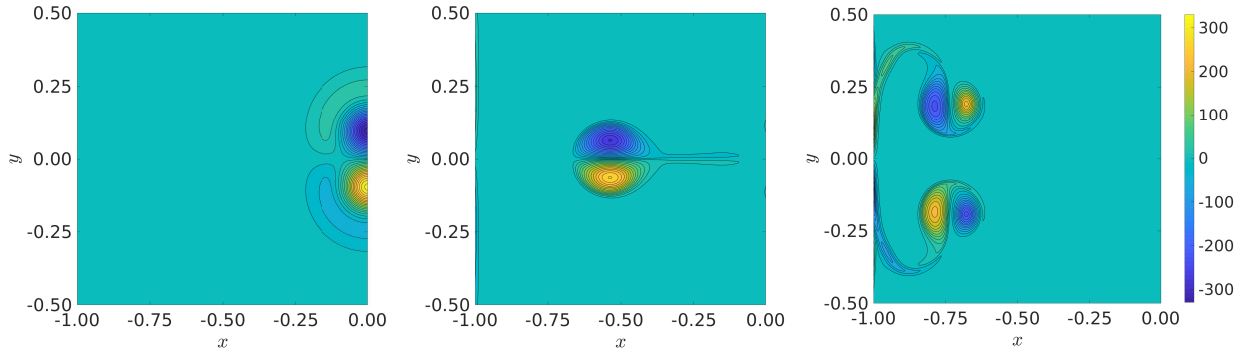
For test cases 2A, 2B, and 2C, the quality of each solution is judged according to the enstrophy history:

$$\xi(t) = \frac{1}{2} \int_{-1}^{1} \int_{-1}^{1} \omega_3^2 d\boldsymbol{x}, \tag{5.18}$$

where $\omega_3(\boldsymbol{x}, t) = \frac{\partial}{\partial x} v_2 - \frac{\partial}{\partial y} v_1$ is the z component of vorticity.

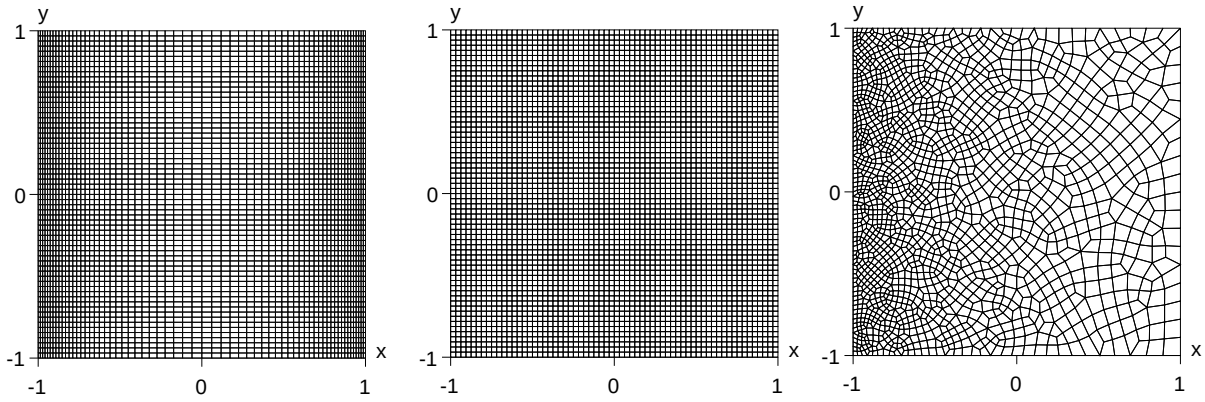
5.5.3.1 Case 2A: Stretched mesh

The three DG discretizations are compared on the structured, stretched mesh shown in Figure 5.13a. The mesh has 64 elements in each direction for a total of $M = 64^2 = 4096$ elements. While the



(a) $t = 0.0$. (b) $t = 0.2$. (c) $t = 0.5$.

Figure 5.12: Progression of dipole-wall interaction from the reference simulation ($p = 3$, conDG, $M = 384^2$); the plotted quantity is the z component of vorticity. The vortex pair is initially located along $x = 0$. The cumulative velocity of the dipole forces both vortices towards the no-slip wall at $x = -1$, and a small boundary layer develops. Once the dipole encounters the no-slip wall, a pair of secondary vortices are formed. All three plots use the same colormap; each contour represents $\Delta\omega = 20$.



(a) Stretched, $M = 64^2$. (b) Uniform, $M = 64^2$. (c) Unstructured, $M = 1697$.

Figure 5.13: Sample meshes for test cases 2A, 2B, and 2C. Test case 2A uses the stretched $M = 64^2$ mesh (element width ratio 1.05) while test case 2B uses a progression of uniform meshes. For test case 2C, all simulations take place on the unstructured $M = 1697$ mesh, which has 72 elements along the $x = -1$ edge, 18 elements along the $x = 1$ edge, and 34 elements along the $y = \pm 1$ edges.

element height ($\Delta y = \frac{2}{64}$) is uniform, the element width Δx is varied to achieve clustering near the no-slip walls at $x = \pm 1$. The element widths for $x > 0$ and $x < 0$ are set according to a width ratio of 1.05; for each pair of adjacent elements along a horizontal line, the larger element is 1.05 times the width of the smaller element.

The enstrophy histories for $p \in \{1, 2, 3\}$ are plotted in Figure 5.14; the enstrophy history from

the reference calculation is included for comparison. For $p > 1$, all discretizations achieve excellent resolution of the enstrophy peak at $t = 0.35$; in contrast, for $t > 0.5$, only the $p = 3$ simulations properly predict the enstrophy. The advantage of the RAD schemes over the conventional DG approach in the $p = 1$ case is extraordinary. While the conventional approach never returns to the initial enstrophy value of $\xi = 800$, the RAD schemes preserve the qualitative character of the dipole-wall interaction.

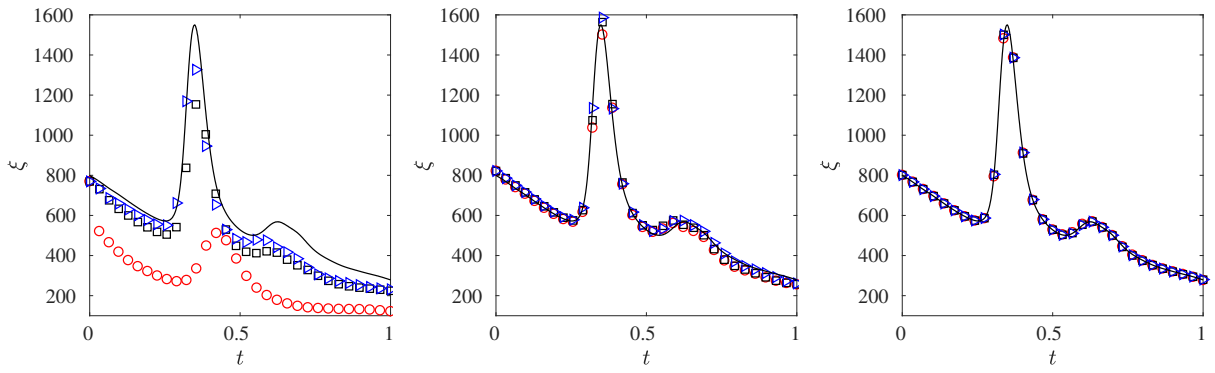
The maximum enstrophy achieved by each simulation is plotted in Figure 5.15a against the polynomial order p. The black line plotted at $\xi_{ref} = 1550.2$ denotes the maximum enstrophy achieved by the reference simulation. Note that for $p = 3$, all three discretizations overpredict the maximum enstrophy; similar behavior has been observed in test cases 2B and 2C. When the simulation is severely underresolved, the maximum enstrophy is underpredicted, but as the resolution improves, the enstrophy is eventually overpredicted, then descends towards the reference value as the resolution continues to improve. Figure 5.15b plots the relative error in the maximum enstrophy,

$$\frac{Error(\xi_{max})}{\xi_{ref}} = \frac{\xi_{max} - \xi_{ref}}{\xi_{ref}}, \tag{5.19}$$

versus the polynomial order p. Overall, the error decreases as the polynomial order p is increased. The RAD1 method is the exception; for $p = 2$, it happens to be more accurate than all other simulations in predicting the maximum enstrophy, but this configuration is a crossover point. When the polynomial order is increased to $p = 3$, the RAD1 approach overshoots the maximum enstrophy by the same amount as the RAD2 approach.

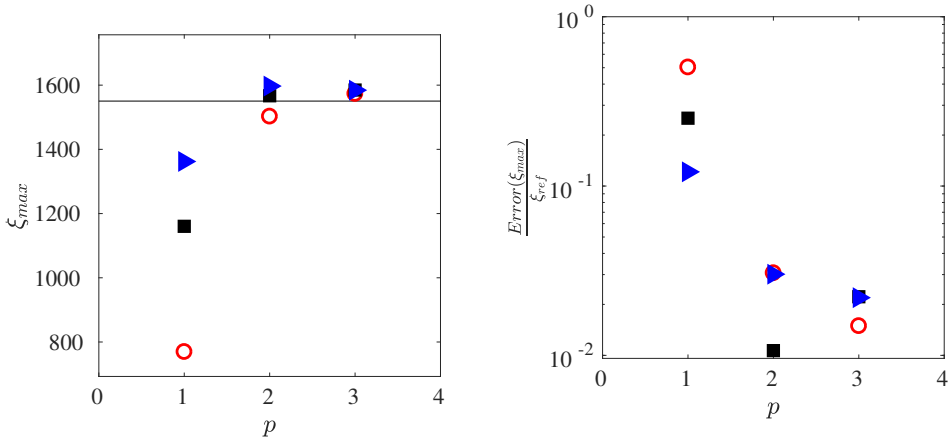
5.5.3.2 Case 2B: Uniform Mesh

We now perform a mesh refinement study using a series of uniform meshes. For $p \in \{1, 2, 3\}$, each DG spatial discretization is applied on a series of four uniform meshes such as that shown in Figure 5.13b. Each mesh consists of $M = R^2$ square elements, where R is the number of elements in each direction and $\Delta x = \Delta y = 2/R$. The mesh resolutions used for each p, along with the DOF counts, are listed in Table 5.5. The accuracy of each simulation is measured via the maximum



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.14: Enstrophy versus time for test case 2A (dipole-wall collision on stretched mesh) with $M = 64^2$ elements. Symbol Key: Reference Simulation: —— ; conDG:○ ; RAD1:□ ; RAD2: ▷.



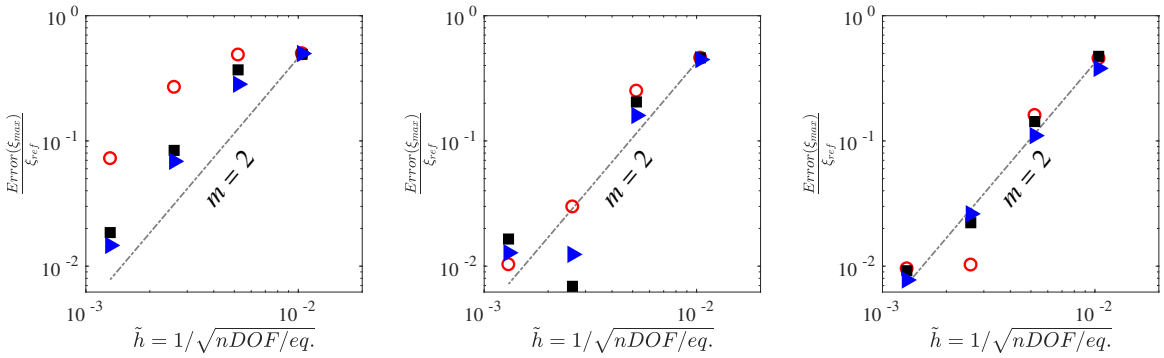
(a) Maximum enstrophy.

(b) Error in maximum enstrophy.

Figure 5.15: Maximum enstrophy and error in maximum enstrophy versus polynomial order p for test case 2A (dipole-wall collision on stretched mesh) with $M = 64^2$. The black line denotes the maximum enstrophy level, $\xi_{ref} = 1550.2$, from the reference simulation.
Symbol Key: conDG: ○ ; RAD1: ■ ; RAD2:▶.

enstrophy achieved, as with test case 2A.

Figure 5.16 plots the enstrophy error versus the characteristic mesh width for $p \in \{1, 2, 3\}$. In all cases, the relative enstrophy error converges to zero at rate $m = 2$. The RAD discretizations are superior in the $p = 1$ case. Note that for a given characteristic mesh width $\tilde{h}$, the $p = 2$ and $p = 3$ simulations achieve similar levels of error for all three DG discretizations. We attribute this behavior to the thin boundary layer along $x = -1$; as noted by Chapelier et al. [21], h-refinement



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

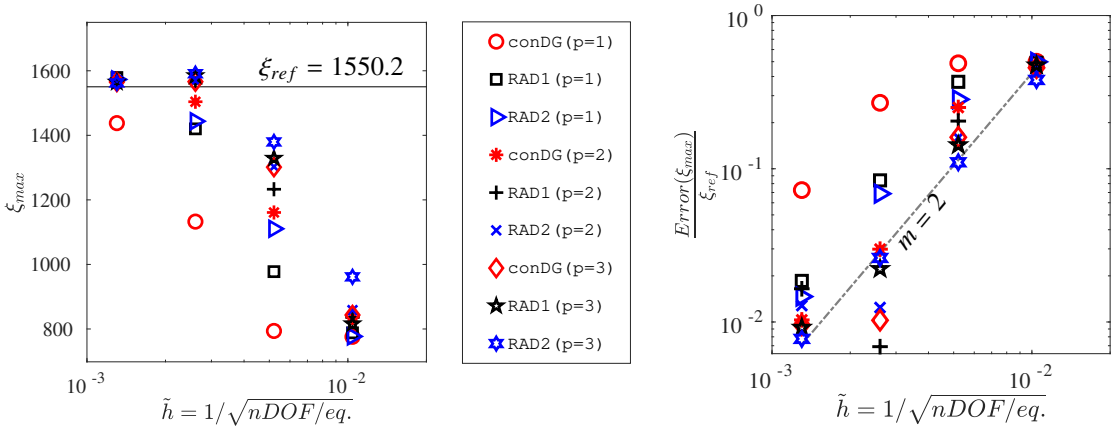
Figure 5.16: Mesh refinement study for test case 2B (dipole-wall collision on uniform mesh). Symbol Key: conDG:○ ; RAD1: ■ ; RAD2:▶. The gray line shows the path of 2^{nd} order convergence.

is often as effective or more effective than p-refinement for reducing error in regions of relatively steep gradients.

In Figure 5.17a, the maximum enstrophy from each simulation is plotted against the characteristic mesh width; all three DG discretizations for $p \in \{1, 2, 3\}$ are presented on the same plot. On coarse meshes ($nDOF/eq. = \{9216, 36864\}$), the RAD2 approach with $p = 3$ exhibits the largest ξ_{max} value. On the second-finest mesh ($nDOF/eq. = 147456$), five out of the nine total configurations overshoot the reference enstrophy value, $\xi_{ref} = 1550.2$. Unsurprisingly, the four configurations that do not overshoot are the three $p = 1$ configurations and the $p = 2$ conDG configuration, which are the most dissipative of the nine configurations. Figure 5.17b collects the relative enstrophy error from all nine DG configurations.

Table 5.5: Mesh resolutions for test case 2B.

p	**Elements per direction (R)**	**Element Count (M)**	**DOF per equation ($nDOF/eq.$)**
1	{48 , 96 , 192 , 384}	{2304 , 9216 , 36864 , 147456}	{9216 , 36864 , 147456 , 589824}
2	{32 , 64 , 128 , 256}	{1024 , 4096 , 16384 , 65536}	{9216 , 36864 , 147456 , 589824}
3	{24 , 48 , 96 , 192}	{576 , 2304 , 9216 , 36864}	{9216 , 36864 , 147456 , 589824}



(a) Maximum enstrophy.

(b) Error in maximum enstrophy.

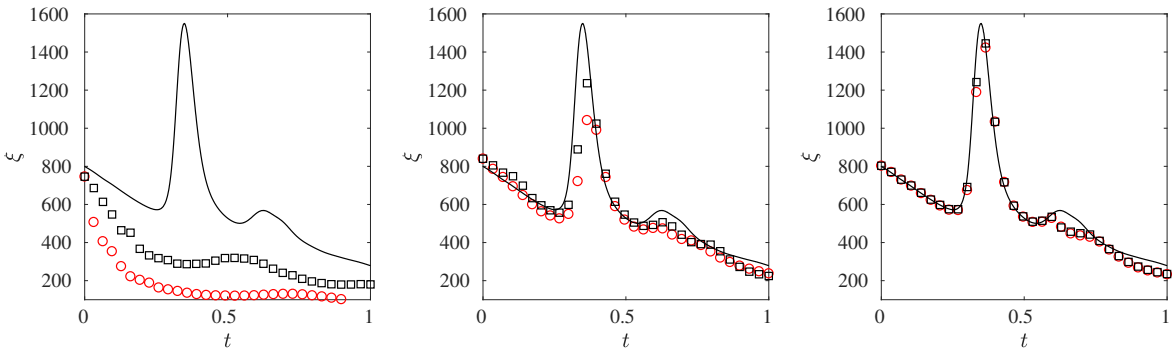
Figure 5.17: Maximum enstrophy (a) and error in maximum enstrophy (b) for all simulations in test case 2B (dipole-wall collision on uniform mesh). The black line denotes the maximum enstrophy level, $\xi_{ref} = 1550.2$, from the reference simulation.

5.5.3.3 Case 2C: Unstructured mesh

We now perform a p-refinement study on a fixed unstructured mesh. The mesh is shown in Figure 5.13c. The mesh has 72 elements along the $x = -1$ wall, 18 elements along the $x = 1$ wall, and 34 elements along each of the spatially periodic boundaries at $y = \pm 1$. The mesh consists exclusively of quadrilateral elements. The unstructured quadrilateral mesh has a total of $M = 1697$ elements, putting it on the relatively coarse end of the mesh resolutions used for case 2B.

The RAD2 approach is absent from this test because it failed to maintain stability for any polynomial order p while using the SLAU2 flux. Experimentation showed that stability could be achieved by instead applying the dissipative Rusanov [92] flux. However, the other two methods performed well with the SLAU2 flux, so instead of using the Rusanov flux with all three approaches, we excluded the less robust RAD2 scheme.

The conDG scheme is compared to the RAD1 scheme for $p < 3$. For the $p = 3$ case, the RAD1 scheme became unstable, so the RAD3 approach was applied instead. The enstrophy histories with $p \in \{1, 2, 3\}$ are given in Figure 5.18. In the $p = 1$ case, neither discretization exceeds the initial enstrophy value of $\xi(0) = 800$. In Figure 5.19a, we plot the max enstrophy of each simula-



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

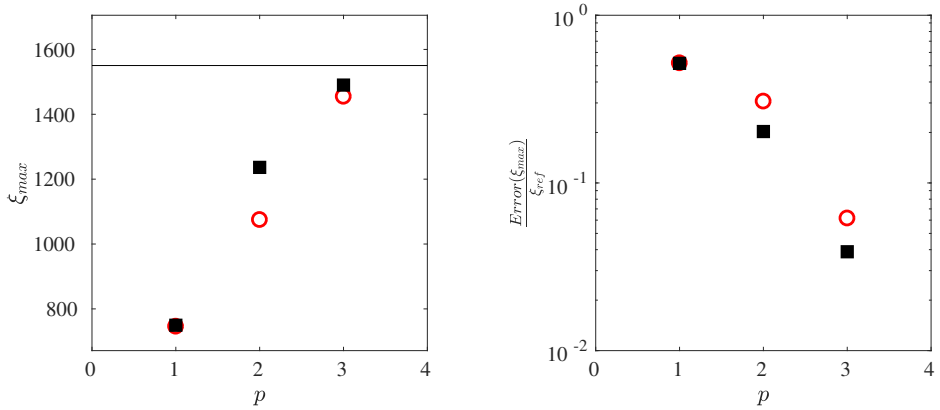
Figure 5.18: Enstrophy versus time for test case 2C (dipole-wall collision on unstructured mesh) with $M = 1697$ elements. Symbol Key: Reference Simulation: —— ; conDG: ○ ; RAD: □.

tion versus the polynomial order p. As p is increased on the fixed mesh, the maximum enstrophy prediction improves, as expected. The relative enstrophy error is plotted versus polynomial order in Figure 5.19b. For both $p = 2$ and $p = 3$, the chosen RAD method is more accurate than the conventional DG approach. This result is encouraging, as it shows that the improved accuracy of the Recovery-assisted philosophy extends not just to stretched meshes but to unstructured quadrilateral meshes as well. However, the need to switch from RAD1 to RAD3 when the polynomial order exceeds 2 is troubling and provides an opening for future research.

Vorticity contours at $t = 0.5$ are plotted in Figure 5.20 and Figure 5.21 for $p \in \{1, 2, 3\}$. At $t = 0.5$, there should be four vortices present in each of these plots: the two initial vortices of the dipole and the two secondary vortices formed from the dipole-wall collision (see Figure 5.12c). The $p = 1$ results suffer from significant asymmetry about the $y = 0$ line, and the conventional discretization completely misses the formation of the secondary vortices; the RAD1 discretization, while not properly prediciting the locations of the secondary vortices, does facilitate their formation. As the polynomial order increases, both schemes tend towards a symmetric dipole-wall collision.

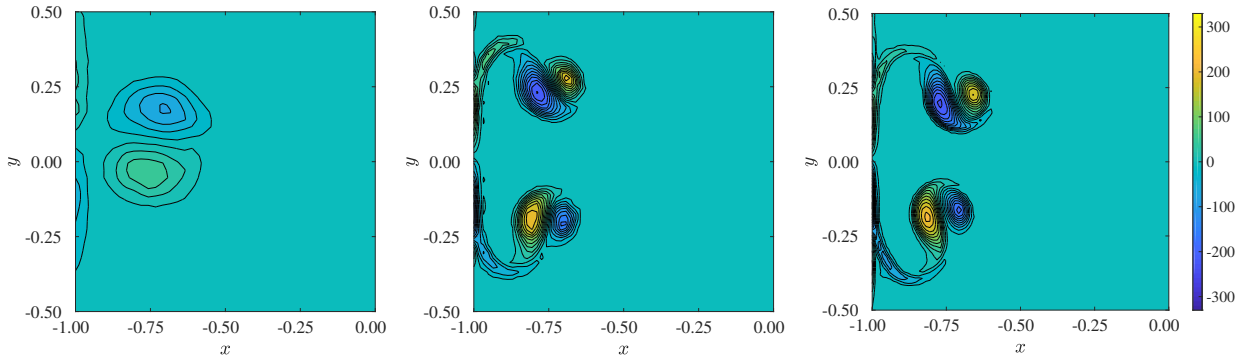
5.5.4 Test 3: Taylor-Green Vortex

The Taylor-Green vortex (TGV) is a popular benchmarking test for research-oriented CFD codes, as the flow exhibits transition from a deterministic initial condition to anisotropic turbulence. The



(a) (b)

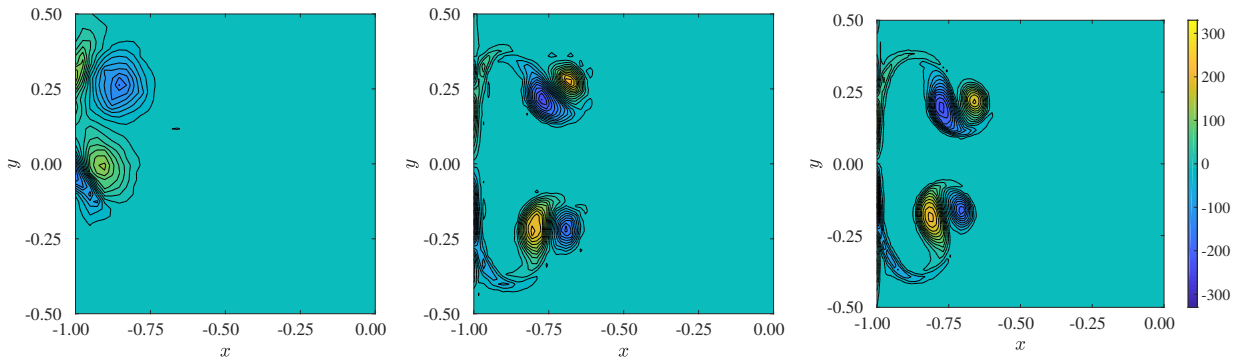
Figure 5.19: Maximum enstrophy (left) and error in maximum enstrophy (right) versus polynomial order p for tet case 2C (dipole-wall interaction on unstructured mesh) with $M = 1697$ elements. The black line denotes the maximum enstrophy level, $\xi_{ref} = 1550.2$, from the reference simulation. Symbol Key: conDG:○ ; RAD:■.



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.20: Vorticity contours at $t = 0.5$ from the conDG simulations of the dipole-wall collision on the unstructured $M = 1697$ mesh (test case 2C). The colormap and contour levels are the same as Figure 5.12; each contour represents $\Delta\omega = 20$.

governing equations are the 3D compressible Navier-Stokes equations (Eq. 2.13), the spatial domain is $(x, y, z) \in [-\pi L, \pi L]^3$, and spatial periodicity is enforced on all boundaries. The initial



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.21: Vorticity contours at $t = 0.5$ from the RAD simulations of the dipole-wall collision on the unstructured $M = 1697$ mesh (test case 2C). The colormap and contour levels are the same as Figure 5.12; each contour represents $\Delta\omega = 20$.

condition is characterized by velocity V_0, pressure p_0, and density ρ_0:

$$\rho = \rho_0, \tag{5.20a}$$

$$v_1 = V_0 \sin(x/L) \cos(y/L) \cos(z/L), \tag{5.20b}$$

$$v_2 = -V_0 \cos(x/L) \sin(y/L) \cos(z/L), \tag{5.20c}$$

$$v_3 = 0, \tag{5.20d}$$

$$p = p_0 + \frac{\rho_0 V_0^2}{16} (\cos(2x/L) + \cos(2y/L)) (\cos(2z/L) + 2). \tag{5.20e}$$

The specific flow parameters in this study are $L = 1$, $V_0 = 1$, and $\rho_0 = 1$; the constant viscosity is set to $\mu = 0.000625$ such that the Reynolds number with respect to the characteristic length L is $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$. The remaining fluid parameters are $\gamma = 1.4$, $R_g = 273.15$, and $Pr = 0.71$. This setup is common for high-order CFD solvers [30, 96, 21]. The reference pressure p_0 is set such that V_0 corresponds to a Mach number of $\frac{1}{10}$; thus, $p_0 = \frac{\rho_0}{\gamma}(10V_0)^2$. We present a mesh refinement study for $p \in \{1, 2, 3\}$ on a set of uniform Cartesian meshes with $\Delta x = \Delta y = \Delta z$. The mesh resolutions are given in Table 5.6 and described as coarse, medium, or fine.

Each simulation is run to $t = 10t_c$ where $t_c = L/V_0$ is the convective time unit. The simulations are compared according to the entrophy-based kinetic energy dissipation rate (KEDR), denoted ϵ

and defined as follows:

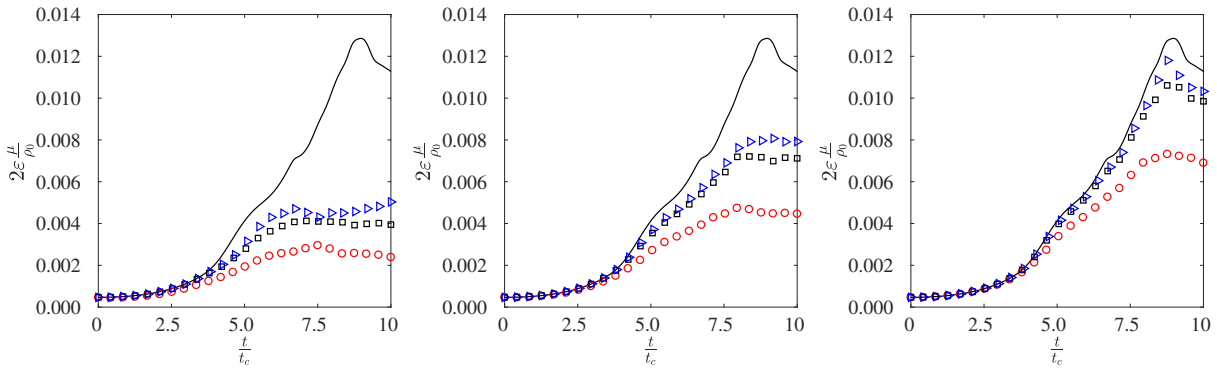
$$\epsilon = 2\mathcal{E}\frac{\mu}{\rho_0} \quad \text{where} \quad \mathcal{E} = \frac{1}{\rho_0 \mathcal{V}} \int_{\Omega} \frac{\rho}{2}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}) d\boldsymbol{x}. \tag{5.21}$$

This quantity is calculated from the velocity field, but it matches the kinetic energy dissipation rate in the incompressible limit, hence the designation "enstrophy-based kinetic energy dissipation rate." The enstrophy-based KEDR is plotted in Figure 5.22 for the $p = 1$ case and compared to the reference solution, which is calculated with a pseudospectral code [30] using 512^3 DOF per equation. The conventional DG discretization with $p = 1$ is widely regarded as being overly dissipative, and that sentiment is reflected here; even on the fine grid ($M = 128^3$), the enstrophy profile from the conventional DG simulation bears little resemblance to that of the reference solution. For $p = 1$, the less dissipative RAD discretizations offer dramatic improvement. Figure 5.23 shows the enstrophy-based KEDR versus time for the $p = 2$ case. In this case, the RAD1 discretization performs slightly better than the conventional approach while the RAD2 approach offers dramatically improved performance.

Figure 5.24 shows the enstrophy-based KEDR versus time for the $p = 3$ case. In this case, due to polynomial aliasing errors associated with the nonlinearity of the compressible Navier-Stokes equations, simulations gave unreasonable results (sometimes even becoming unstable) when the SLAU2 flux was employed. To stabilize the simulations, we switched to the highly dissipative Rusanov flux [92]. Overintegration [63, 11] could also be employed to stabilize the simulations, but with higher computational cost; in fact, to increase the quadrature precision by one point in each direction, the number of volume quadrature points per element would have increased from $(p+1)^3 = 64$ to $(p+2)^3 = 125$. Figure 5.24 shows the conDG discretization behaving as expected:

Table 5.6: Mesh resolutions for test case 3.

p	Elements per direction (R)	Element Count (M)	DOF per equation ($nDOF/eq.$)
1	{32 , 64 , 128}	{32768 , 262144 , 2097152}	{64^3 , 128^3 , 256^3}
2	{21 , 42 , 85}	{9261 , 74088 , 614125}	{63^3 , 126^3 , 255^3}
3	{16 , 32 , 64}	{4096 , 32768 , 262144}	{64^3 , 128^3 , 256^3}

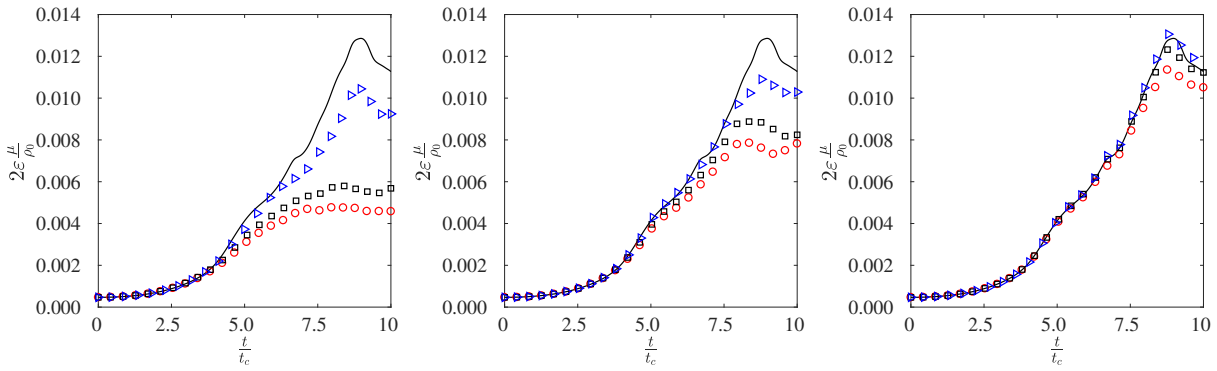


(a) $p = 1$, $M = 32^3$. (b) $p = 1$, $M = 64^3$. (c) $p = 1$, $M = 128^3$.

Figure 5.22: Enstrophy-based KEDR versus time for test case 3 with $p = 1$.
Symbol Key: Reference Simulation: —— ; conDG: ○ ; RAD1: □ ; RAD2: ▷.

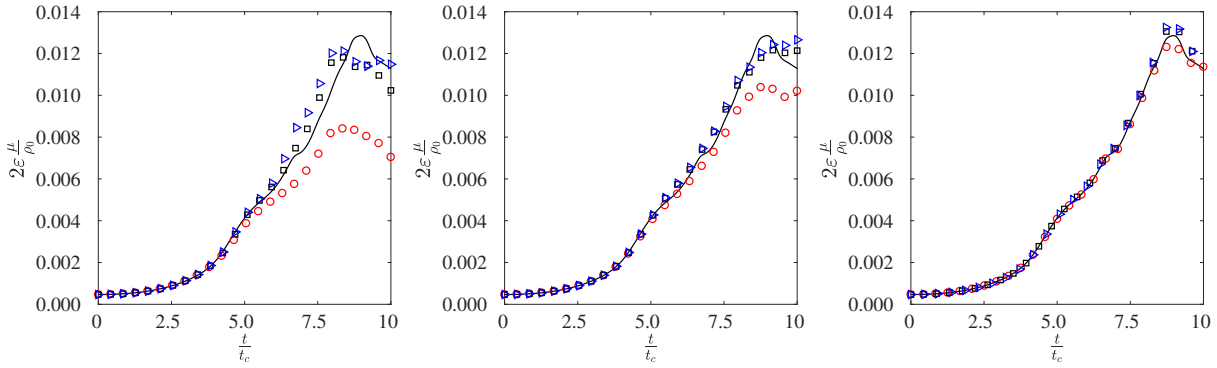
for a given mesh width (fine, medium, or coarse), the $p = 3$ simulation is superior to the $p = 1$ and $p = 2$ simulations. The only anomaly in the conDG results is that in the medium resolution case ($M = 32^3$), the enstrophy-based KEDR is increasing at $t = 10t_c$.

The relative error in the maximum enstrophy is plotted in Figure 5.25. Given the maximum enstrophy, ϵ_{max}, from a given simulation, the error is calculated as $Error(\epsilon_{max}) = |\epsilon_{max} - \epsilon_{ref}|$, where $\epsilon_{ref} = (0.129)\frac{\rho_0}{2\mu}$ is the maximum enstrophy from the reference simulation. Note that this error norm can be deceptive; for the $p = 3$ case with $M = 32^3$, the RAD2 discretization has the highest maximum enstrophy, but that maximum enstrophy occurs at $t = 10t_c$, where the enstrophy should be decreasing. Thus, the error norms presented in Figure 5.25 must be taken in context with the profiles provided in Figures 5.22, 5.23, and 5.24. For $p < 3$, the RAD2 scheme is clearly superior in capturing the enstrophy profile of the Taylor-Green vortex flow. Of the nine mesh configurations used for the simulation (three polynomial orders by three mesh resolutions), eight show an identical trend: the conDG discretization is the least accurate while the RAD2 scheme is the most accurate. In contrast, for the finest mesh in the $p = 3$ progression, the conventional DG simulation is the most accurate because the RAD simulations significantly overshoot the maximum enstrophy from the reference simulation.



(a) $p = 2$, $M = 21^3$. (b) $p = 2$, $M = 42^3$. (c) $p = 2$, $M = 85^3$.

Figure 5.23: Enstrophy-based KEDR versus time for test case 3 with $p = 2$.
Symbol Key: Reference Simulation: —— ; conDG: ○ ; RAD1: □ ; RAD2: ▷.

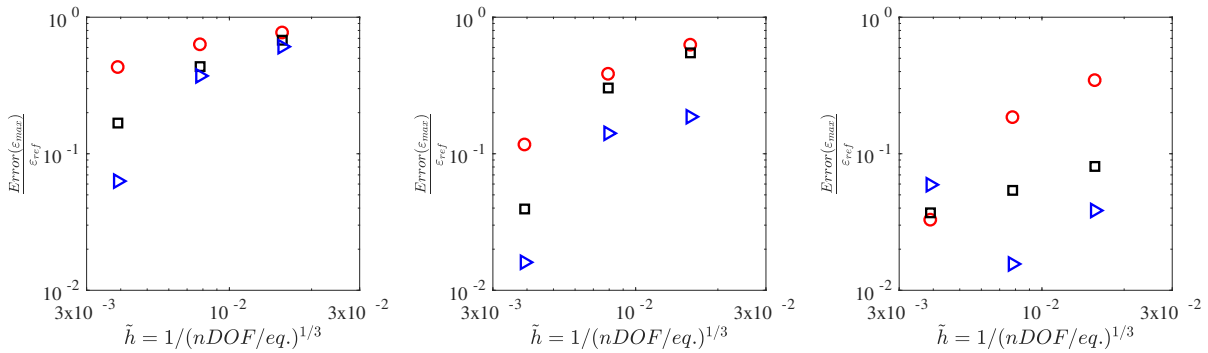


(a) $p = 3$, $M = 16^3$. (b) $p = 3$, $M = 32^3$. (c) $p = 3$, $M = 64^3$.

Figure 5.24: Enstrophy-based KEDR versus time for test case 3 with $p = 3$. These calculations use a Rusanov flux instead of SLAU2.
Symbol Key: Reference Simulation: —— ; conDG: ○ ; RAD1: □ ; RAD2: ▷.

5.5.5 Test 4: Decaying Compressible HIT

The 3D compressible Navier-Stokes equations are discretized to simulate the decay of compressible homogeneous isotropic turbulence (HIT). This problem has been simulated by several authors. Lee et al. [65] demonstrated that eddy shocklets occur at sufficiently high turbulent Mach number. Johnsen et al. [48] employed the test case to compare different high-order discretizations for the compressible Navier-Stokes equations; Honein & Moin [45] used it to study the effect of nonlinear aliasing errors in finite-difference schemes. Here, we follow the setup of Lv and Ihme [72], who employed the problem to compare an entropy-bounded DG method [71] to a finite-volume solver.



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 5.25: Relative error in max enstrophy versus characteristic mesh width for test case 3. Symbol Key: conDG: ○ ; RAD1: □ ; RAD2: ▷.

The descriptions of the initial condition and the simulation comparison procedure are relatively verbose to allow for replication of our results.

As with the Taylor-Green vortex, the initial condition is specified in terms of the primitive variables. The fluid's ratio of specific heats is $\gamma = 1.4$, the gas constant is $R_g = 1$, and the Prandtl number is $Pr = 0.7$. The spatial domain is $\boldsymbol{x} \in [0, 2\pi]^3$. The pressure and density are taken to be spatially uniform at $t = 0$, i.e. $p(\boldsymbol{x}, 0) = p_0$ and $\rho(\boldsymbol{x}, 0) = \rho_0$. The initial velocity field is solenoidal with zero mean flow and is set according to two parameters: the root-mean-square velocity u_{rms} and the most energetic wavenumber k_0. Given those two parameters, the velocity field is a sum of Fourier modes adhering to the following energy spectrum:

$$E_u(k) = 16\sqrt{\frac{2}{\pi}} u_{rms}^2 \frac{k^4}{k_0^5} \exp(-2\frac{k^2}{k_0^2}). \tag{5.22}$$

This model spectrum is the typical setup for compressible HIT studies. We import the velocity field employed by the study of Lv & Ihme [72] and available online; the dominant wavenumber is $k_0 = 4$ and the rms velocity in each direction is unity. The flow is characterized by the turbulent Mach number,

$$Ma_t = \frac{\sqrt{3}u_{rms}}{c}, \tag{5.23}$$

where c is the intial speed of sound in the domain, and the Reynolds number,

$$Re_\lambda = \frac{\rho_0 u_{rms} \lambda}{\mu_0}, \tag{5.24}$$

where ρ_0 is the intial density, $\lambda = \frac{2}{k_0}$ is the Taylor microscale, and μ_0 is the initial viscosity. The initial density is $\rho_0 = 1$. With the speed of sound being defined as $c = \sqrt{\gamma p_0/\rho_0}$, the initial pressure is set to

$$p_0 = \frac{\rho_0}{\gamma}(\frac{\sqrt{3}u_{rms}}{Ma_t})^2 \tag{5.25}$$

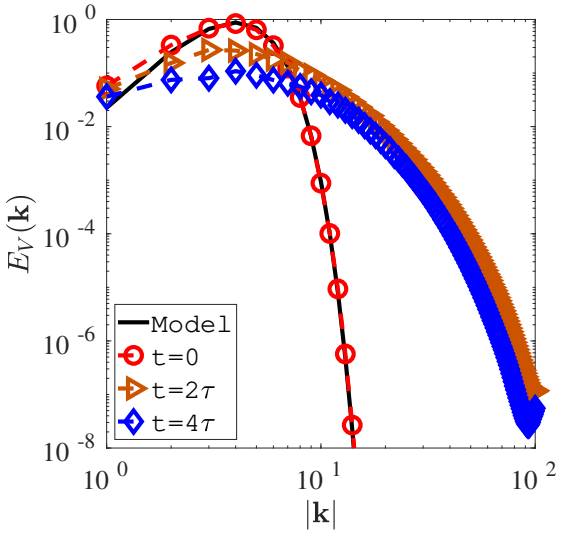
to satisfy the Mach number constraint. The viscosity μ is a function of temperature: $\mu = \mu_{ref}(\frac{T}{T_{ref}})^{0.76}$, where $T = \frac{p}{\rho R_g}$. Taking the reference viscosity to be $\mu_{ref} = 0.0125$, the reference temperature T_{ref} is set so that the initial viscosity μ_0 satisfies the chosen Reynolds number. The chosen Mach and Reynolds numbers for this study are $Ma_t = 0.3$ and $Re_\lambda = 100$; in this setup, the flow is free of shock waves [45]. This particular setup was chosen to yield a direct comparison between the Recovery-assisted and conventional DG discretizations without the complication of a shock-capturing scheme. The eddy turnover time is $\tau = \lambda_0/u_{rms} = 0.5$. Each simulation is run to four turnover times: $t_{final} = 4\tau$.

The initialization of the velocity field is now described in detail for the sake of replication; it relied heavily on manipulating an imported velocity dataset. The imported dataset of Lv & Ihme [72] consists of purely real velocity values on an evenly spaced grid of 257 points in each direction; the field is spatially periodic, and the first and last entries along each 1D string of points were duplicates. We removed the duplicate data to obtain the velocity field on a 256^3 grid of evenly spaced points, which we call the "reference" dataset. The discrete Fourier transform of the reference dataset was calculated to determine the reference Fourier coefficients in 256^3 wavenumber space; these Fourier coefficients display certain symmetry properties due to the physical data being purely real, but this symmetry was not exploited. The set of reference Fourier coefficients can be filtered down to a smaller set of Fourier coefficients, with the highest-wavenumber components being cut off. The 256^3 set of Fourier coefficients was filtered to four specific resolutions: a 24^3 set of coeffi-

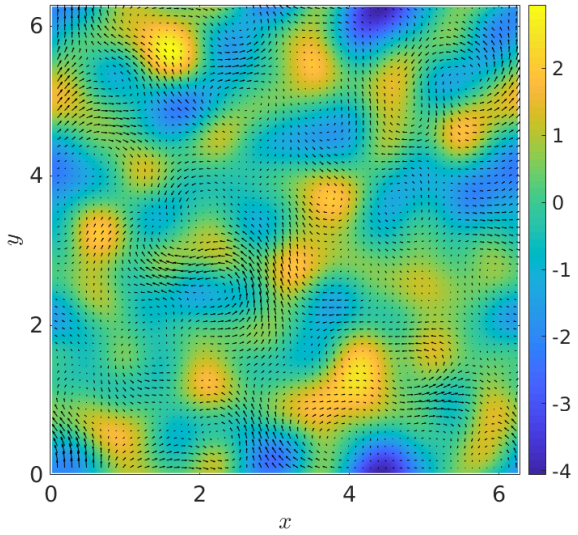
cients, a 48^3 set of coefficients, a 96^3 set of coefficients, and a 192^3 set of coefficients. Each of these four sets of Fourier coefficients were sent through an inverse discrete Fourier transform to yield purely real nodal velocity values on an evenly spaced grid. When the code runs an HIT simulation with R gridpoints per direction on the 2π cube, the initial velocity field is initialized based on nodal equivalence to the corresponding initial velocity distribution at R evenly spaced points across the 2π cube in each direction. One could instead initialize the flow field based on Galerkin equivalence to the reference velocity dataset, but that approach would alias high-wavenumber components to low-wavenumber components in Fourier space, potentially causing a severe mismatch between the refererence velocity dataset and the initial DG DOFs on a coarse grid. Our approach is built specifically to avoid this problem, as it enforces equivalence in the wavenumber domain up to a certain cutoff wavenumber for each grid. Additionally, the chosen initialization strategy can be matched to a finite-difference or spectral/pseudospectral code in a nodally exact manner.

The initial energy spectrum is presented in Figure 5.26a alongside the model energy spectrum (Eq. 5.22). Figure 5.26b shows the initial velocity field on the $z = 0$ face. As the flow evolves, relatively high-wavenumber velocity fluctuations arise, as illustrated in Figure 5.26c and Figure 5.26d at $t = 2\tau$ and $t = 4\tau$, respectively. The flow progression is also observable in Figure 5.26a, where the energy spectrums at $t = 2\tau$ and $t = 4\tau$ are plotted alongside the initial energy spectrum. While the overall kinetic energy of the flow decays in time, the energy is transferred from the large scales towards the smaller scales (high wavenumbers). All results in Figure 5.26 are taken from the reference simulation. The reference simulation uses the RAD1 discretization with $p = 2$ on a Cartesian mesh with $nDOF/eq. = 258^3$ (86 elements per direction). At this resolution, the conDG, RAD1, and RAD2 schemes yield identical results.

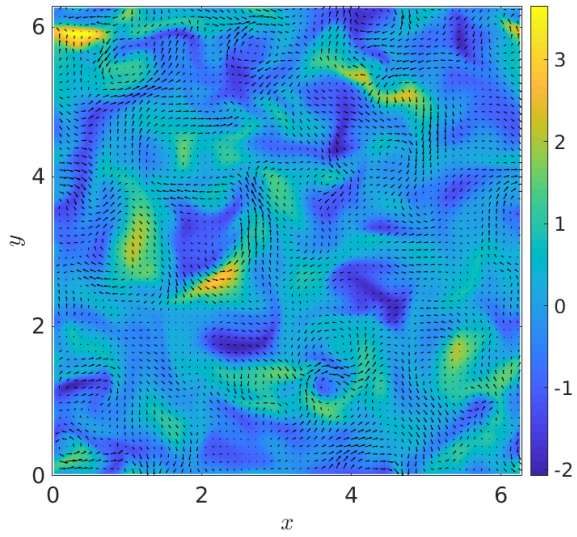
To compare the conventional DG approach to the Recovery-assisted schemes, the flow is simulated on a set of Cartesian meshes using $nDOF/eq. = 48^3$ (coarse mesh) and $nDOF/eq. = 96^3$ (fine mesh) with $p \in \{1, 2, 3\}$. The discretizations are compared in terms of the mass-averaged enstrophy history and the kinetic energy dissipation rate. The mass-averaged enstrophy is computed



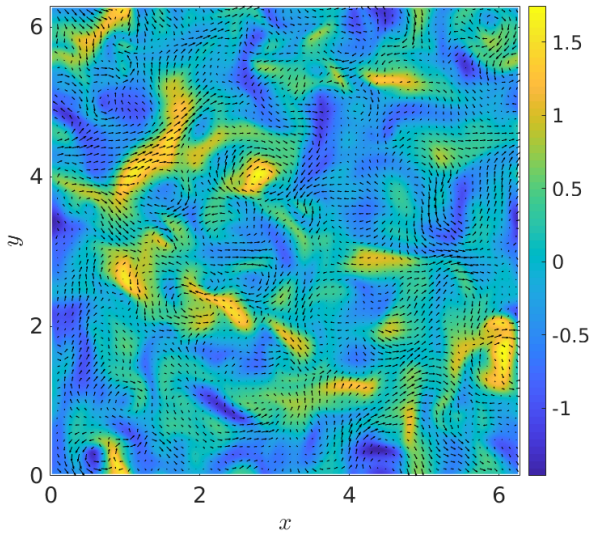
(a) Velocity energy spectrum.



(b) Velocity field on $z = 0$ face, $t = 0$.



(c) Velocity field on $z = 0$ face, $t = 2\tau$.

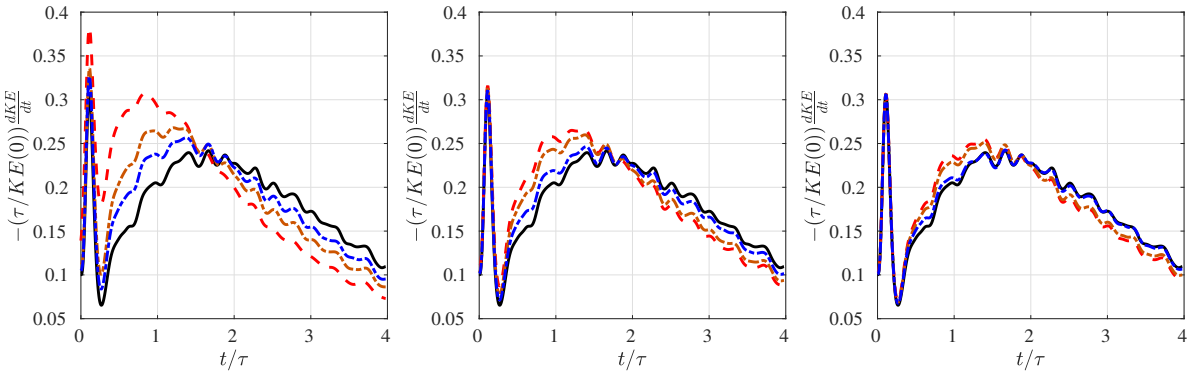


(d) Velocity field on $z = 0$ face, $t = 4\tau$.

Figure 5.26: **Test 4:** HIT flow overview. In subfigures b, c, and d, the plotted color indicates the velocity component in the z direction. The transverse velocity components are displayed as small arrows. As the flow progresses in time, kinetic energy is transferred to smaller length scales and the overall kinetic energy decreases.

at the end of each timestep as follows:

$$< \frac{\rho}{2}\omega_i\omega_i >= \frac{\int_{\mathbf{\Omega}} \frac{\rho}{2}\omega_i\omega_i d\boldsymbol{x}}{\rho_0 \int_{\mathbf{\Omega}} d\boldsymbol{x}}, \tag{5.26}$$

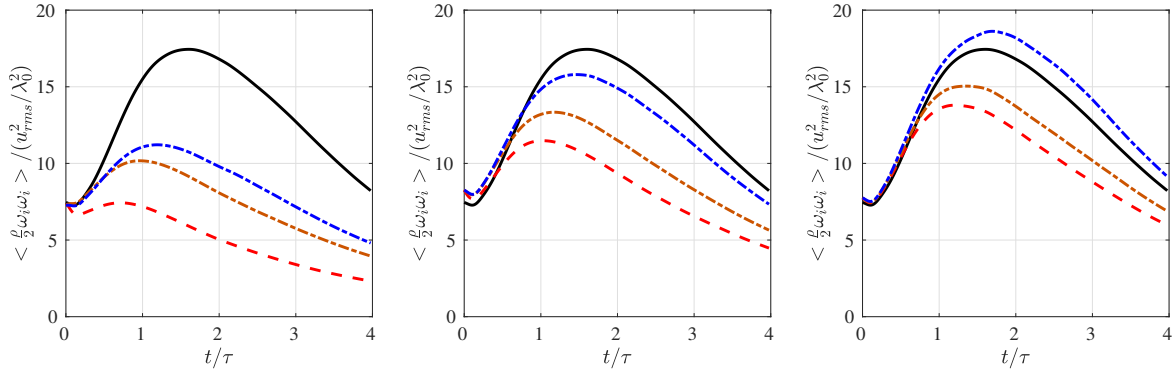


(a) $p = 1$, $M = 24^3$. (b) $p = 2$, $M = 16^3$. (c) $p = 3$, $M = 12^3$.

Figure 5.27: **Test 4:** Kinetic energy dissipation rate vs. time with $nDOF/eq. = 48^3$.
Key: **Reference:** —— ; **conDG:** − − − ; **RAD1:**−. − .− ; **RAD2:**−. − .− .

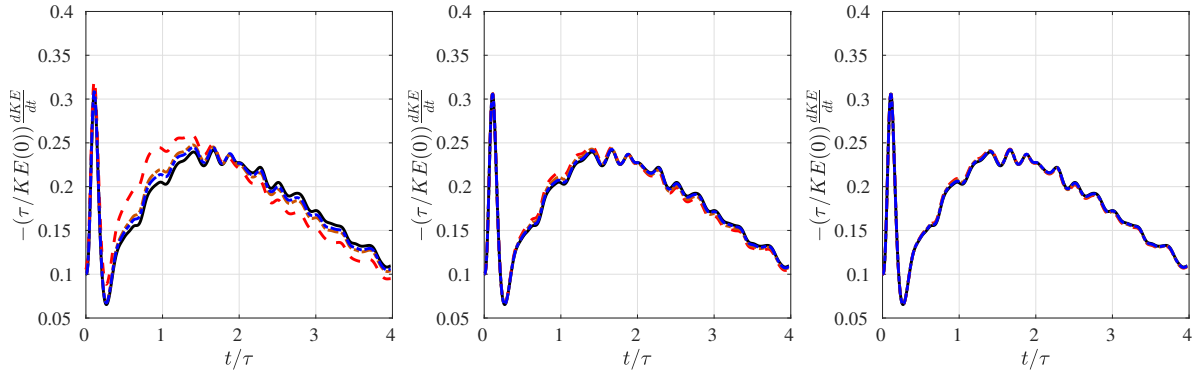
where repeated indices indicate summation over the three spatial dimensions and $\boldsymbol{\omega}$ is the vorticity vector. The kinetic energy dissipation rate (KEDR) is computed in the post-processing step. After each timestep, the code outputs the kinetic energy, $KE(t) = \int_{\boldsymbol{\Omega}} \frac{\rho}{2} v_i v_i d\boldsymbol{x}$. In the post-processing step, the kinetic energy history is differentiated in time with a four-point finite difference to calculate the KEDR. The integration for kinetic energy and enstrophy is performed in an element-by-element fashion using $Q_V = (p+1)^2$ quadrature points per element.

Figure 5.27 and Figure 5.28 report the KEDR and enstrophy, respectively, from the simulations on the coarse mesh ($nDOF/eq. = 48^3$). The KEDR is normalized by $\frac{KE(0)}{\tau}$ and the mass-averaged enstrophy is normalized by $\frac{u_{rms}^2}{\lambda_0^2}$. The RAD2 scheme provides superior results compared to the RAD1 and conDG schemes for all tested solution orders p. Results from the simulations on the fine mesh ($nDOF/eq. = 96^3$) are reported in Figure 5.29 and Figure 5.30. The Recovery-assisted schemes provide superior performance compared to the conventional DG approach. Between the two Recovery-assisted schemes, RAD2 consistently exhibits larger enstrophy levels, which is not necessarily a positive attribute because it overshoots the reference simulation's enstrophy level in the $p = 2$ and $p = 3$ cases. Regardless of the spatial discretization, the underresolved simulations in this study tend to overpredict the KEDR in early time and underpredict it in late time. For both the coarse and fine mesh simulations, the simulation results improve with the polynomial order p. The $p = 3$ simulations on the fine mesh exhibit excellent agreement with the reference solution.



(a) $p = 1$, $M = 24^3$. (b) $p = 2$, $M = 16^3$. (c) $p = 3$, $M = 12^3$.

Figure 5.28: **Test 4:** Mass-averaged enstrophy vs. time with $nDOF/eq. = 48^3$.
Key: **Reference:** —— ; **conDG:** – – – ; **RAD1:**–. – .– ; **RAD2:**–. – .– .

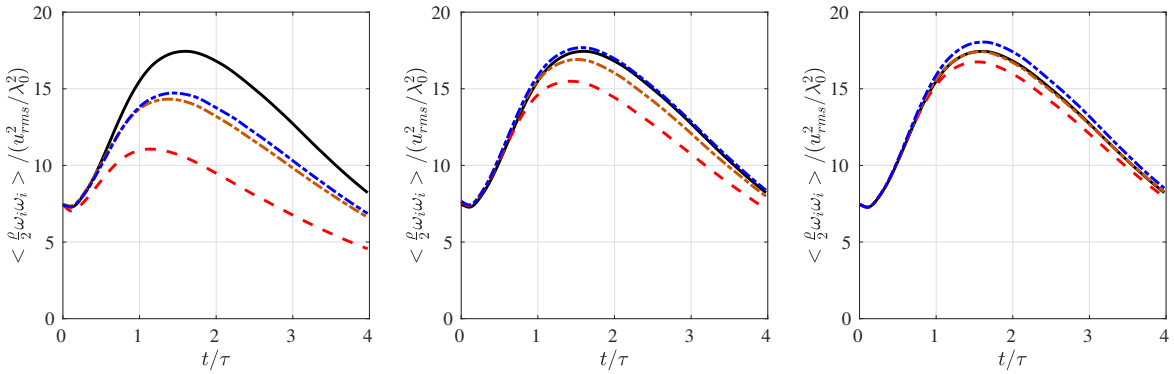


(a) $p = 1$, $M = 48^3$. (b) $p = 2$, $M = 32^3$. (c) $p = 3$, $M = 24^3$.

Figure 5.29: **Test 4:** Kinetic energy dissipation rate vs. time with $nDOF/eq. = 96^3$.
Key: **Reference:** —— ; **conDG:** – – – ; **RAD1:**–. – .– ; **RAD2:**–. – .– .

5.6 Chapter Conclusion

The ICB approach for advection and the CGR approach for diffusion were combined to form a set of Recovery-assisted DG (RAD) schemes for advection-diffusion problems. The ICB approach uses a biased version of the recovery operator to improve the accuracy of the advective flux terms while the CGR approach augments the mixed formulation for the diffusive flux terms with the classical full-order recovery operator. The results were encouraging. Fourier analysis showed that the new Recovery-assisted advection-diffusion discretizations achieve superior wavenumber resolution compared to the conventional DG approach over a broad spectrum of element Peclet numbers.



(a) $p = 1$, $M = 48^3$. (b) $p = 2$, $M = 32^3$. (c) $p = 3$, $M = 24^3$.

Figure 5.30: **Test 4:** Mass-averaged enstrophy vs. time with $nDOF/eq. = 96^3$.
Key: **Reference:** —— ; **conDG:** − − − ; **RAD1:**−. − .− ; **RAD2:**−. − .− .

A set of linear and nonlinear test problems demonstrated that the new RAD schemes achieve superior performance compared to the conventional approach, assuming a sufficiently regular mesh. The superior accuracy of the Recovery-assisted schemes stems from the usage of the biased and full-order recovery operators to calculate the ambiguous interface terms in Eq. (5.2).

There is room for improvement on unstructured meshes. Both the RAD1 and RAD2 schemes are less robust than the conventional DG approach on an unstructured quadrilateral mesh, and this deficiency hampers their applicability on general flow physics problems. Additionally, testing on simplex elements is noticeably absent from this chapter because we have so far been unable to formulate a stable and satisfactorily accurate implementation of the biased recovery procedure on simplex elements. Further analysis could yield a fix for this behavior; extensive study of the primal form of the spatial discretization could yield an appropriate tuning of the recovery weights in the derivative-based recovery operators to maximize robustness while maintaining an appreciable accuracy advantage over the conventional DG approach. The ideal scheme would optimize the dispersion relation while applying a suitable amount of dissipation to underresolved modes.

CHAPTER 6

Boundary Procedures for van Leer & Lo's Recovery-based DG Method

6.1 Chapter Overview

DG schemes for handling the 1D diffusion equation are usually order $2p$ accurate; preferably, the order of accuracy would be at least $2p+1$ so that the accuracy of the DG method was not degraded when moving from a pure advection problem to an advection-diffusion problem. Additionally, DG schemes for the diffusion equation tend to exhibit relatively large spectral radii, which becomes a liability when explicit time integration is employed. Third, they typically contain tunable stabilization parameters and novel mathematical constructs (lifting operators) that are difficult to understand for anyone who is not an expert in the DG method. In response to this list of issues, van Leer & Nomura [102] proposed the Recovery concept and a Recovery-based DG scheme for the 1D diffusion equation, laying the foundation for Lo [67] to propose a variety of Recovery-based DG methods. In this chapter, we focus on one of Lo's methods, namely the RDG-1x++CO version of Recovery-based DG. Our contribution to the method was to minimize the degradation in convergence rates when solving a shear diffusion equation ($\theta > 0$ in Eq. 2.7) under Dirichlet or Neumann boundary conditions. In the case of Dirichlet boundary conditions, we constructed a boundary scheme that can preserve order $3p$ convergence in the cell-average error; this convergence rate is unrivaled by other DG schemes for diffusion, making the Recovery-based DG scheme an excellent choice for the discretization of pure diffusion problems on Cartesian meshes.

In the pursuit of an appropriate diffusion scheme for a combined advection-diffusion scheme, we ultimately abandoned the Recovery-based DG family in favor of the Interface Gradient Recovery (IGR) family developed in Chapter 3. The main issue of the Recovery-based DG schemes is that when a shear term is present in the diffusion equation, no compact Recovery-based DG scheme can maintain stability and consistency for $p > 1$. This shortcoming inspired the formation of the Compact Gradient Recovery (CGR) and High-Accuracy-Gradient (HAG) approaches in Chapter 3, which are built to exploit the accuracy of the recovery operator without the shortcomings of Recovery-based DG.

In this chapter, we review the origins of the Recovery-based DG family of schemes. Then, the specific RDG-1x++CO discretization is described, and we propose two new approaches for the accommodation of Dirichlet and Neumann boundary conditions; in the interest of keeping this thesis manageably brief, some details on the RDG-1x++CO approach are omitted; the interested reader is directed to Lo's thesis [67] for a more thorough description. Finally, the boundary schemes are evaluated with a simple test problem.

6.1.1 Novelty and Articles

The novelty of this chapter is a pair of new schemes for handling Dirichlet and Neumann boundary conditions within the RDG-1x++CO spatial discretization. Previously, Dirichlet and Neumann boundary conditions in the Recovery-based DG family had only been addressed for the simpler RDG-1x+ scheme [67].

The material of this chapter appears in one AIAA conference manuscript:

- P. E. Johnson & E. Johnsen, *Progress Towards the Application of the Recovery-Based Discontinuous Galerkin Method to Practical Flow Physics Problems*, AIAA Paper 2016-3331.

6.1.2 Usage of Recovery

The recovery operator is applied in the inner-product based implementation. Only the full-order recovery type (Section 2.6.2) is employed, and it is implemented exclusively on 2D Cartesian

elements. The biased recovery operation makes no appearance in this chapter.

6.2 A Brief History of Recovery-based DG

The naive approach for the diffusion equation, detailed in Section 2.3.1, performs poorly for two reasons. First, the practice of taking the average gradient along each interface does not account for the jump in the approximation U^h. Large solution jumps at interfaces correspond to large gradients, and the naive scheme does not properly relate the two quantities. Second, since U^h is a polynomial, information is lost when evaluating the gradient in each element. The first issue is the more important of the two. The Recovery-based DG schemes of Lo & van Leer [69, 67, 68, 101] address this shortcoming with the use of the recovered solution (Section 2.6). The recovered solution, in addition to providing a solution approximation at the interface, can be differentiated to provide an approximation for the solution gradient. When the recovered solution is differentiated, the information contained in the interface solution jump is accounted for. In the case of the 1D diffusion equation (Eq. 2.2), this procedure results in stable DG schemes (RDG-2x and RDG-1x+) without the complications of global lifting operators, local lifting operators, and penalization parameters [4, 15] that are required for other DG schemes. Additionally, the RDG-1x+ and RDG-2x schemes proposed by van Leer & Lo achieve order $3p+1$ convergence or greater in the cell-average error, which is frequently used to compare DG methods to finite volume methods; other DG methods are typically limited to order $2p$ convergence. These two schemes (RDG-2x and RDG-1x+) maintain a compact stencil in the 1D case. However, in the 2D case, RDG-2x and RDG-1x+ become unstable and/or inconsistent in the presence of a shear diffusion flux law ($\theta > 0$ in Eq. 2.7), necessitating the development of non-compact Recovery-based DG schemes instead. While the loss of compactness is undesirable, the resulting RDG-1x++CO scheme performs exceptionally well for 2D shear diffusion problems; the orders of accuracy and spectral radii for $p \leq 5$ are listed in Table 3.9 (as part of the IGR Fourier analysis) and compare favorably to other DG schemes for diffusion. The schemes developed in this chapter minimize performance degradation when moving from periodic boundary conditions to Dirichlet or Neumann boundary conditions.

6.3 The Recovery-based DG Method

From now on, the RDG-1x++CO scheme [69] is simply referred to as RDG. This section is exclusively a review of a method. It exists to orient the reader before the boundary procedures are described in the next section.

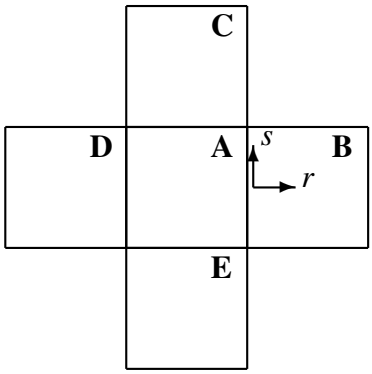


Figure 6.1: Neighboring elements.

The RDG scheme is for Cartesian elements in 2D. In addition to assuming a Cartesian mesh, let $\boldsymbol{\phi_e}$ for each Ω_e be a modal tensor product basis with $K = (p+1)^2$ members for each element; the use of a modal basis is important for the solution enhancement step of RDG. Given some diffusive conservation law with flux $\boldsymbol{\mathcal{G}}$, the DG weak form is:

$$\int_{\Omega_m} \phi_m^k \frac{\partial}{\partial t}(\sum_{n=0}^{K-1} \hat{U}_m^n \phi_m^n) d\boldsymbol{x} = \int_{\partial\Omega_m} \phi_m^{k-}(\tilde{\boldsymbol{\mathcal{G}}} \cdot \mathbf{n}_\mathbf{m}^-) ds - \int_{\Omega_m} \nabla\phi_m^k \cdot \boldsymbol{\mathcal{G}} d\boldsymbol{x}, \quad \forall \phi_m^k \in \boldsymbol{\phi_m}, \quad \forall \Omega_m \in \boldsymbol{\Omega}. \tag{6.1}$$

Consider elements A,B,C,D,and E, as shown in Figure 6.1, on the interior of the computational grid. On each interface, we make use of a set of recovery coordinates, with r running in the face-normal direction and s in the face-tangential direction. Recall that the recovered solution is built in a recovery basis, $\boldsymbol{\psi}$, that contains higher polynomial modes than the DG basis $\boldsymbol{\phi}$. For example, in the $p = 1$ case, each element contains 4 degrees of freedom. Thus, the recovered solution contains 8 degrees of freedom. While the DG solution basis $\boldsymbol{\phi}$ is bilinear in each element, the recovered function is built in a tensor product basis that is cubic in the face-normal direction but only linear in the face-tangential direction.

Given the DOFs $\hat{\mathbf{U}}$ at time t, the first step in the RDG scheme is to calculate the primary

recovered solution $f(r, s)$ along each interface using the discrete recovery operator described in Section 2.6. It is called the *primary* recovered solution because in the process of populating the DG weak form (Eq. 6.1), a secondary recovery step will be necessary.

Next, the primary recovered solution f is used to assist in the process deemed *solution enhancement* by van Leer & Lo [69]. A thorough summary is given in the thesis of Lo [67]. Whereas Recovery is used to populate a solution approximation along element interfaces, solution enhancement is used to replace the DG approximation U_e^h over each Ω_e with an enhanced solution, U_e^{en}. Where the DG approximation U^h consists of K DOFs per element, the enhanced solution U^{en} makes use of $K^{en} = K + 4(p + 1)$ coefficients per element. For example, considering again the $p = 1$ case, $K = 4$, and the enhanced solution contains $K^{en} = 12$ coefficients. The enhanced solution is a polynomial expansion, as shown below:

$$U_e^{en}(x, y) = \sum_{n=0}^{K^{en}-1} \hat{U}_e^{en,m} \phi_e^{en,m}(x, y) \quad , \quad K^{en} = (p + 1)^2 + 4(p + 1). \tag{6.2}$$

The members of the enhanced basis $\boldsymbol{\phi_e^{en}}$ are formed by starting from the modal tensor product basis $\boldsymbol{\phi_e}$ of degree p and adding $2(p + 1)$ extra basis functions in the reference element's ξ direction and $2(p + 1)$ extra basis functions in the reference element's η direction. The result is an incomplete degree $p + 2$ tensor product basis; the missing members are the four basis functions corresponding to degree greater than p in both the ξ and η directions. The enhanced basis, $\boldsymbol{\phi_e^{en}}$, inherits all members of $\boldsymbol{\phi_e}$; additionally, it contains extra basis functions that are used to improve the accuracy of the approximate solution. Regardless of p, the enhanced solution basis contains 2 extra columns of shape functions in the ξ direction of the reference element and 2 extra rows in the η direction of the reference element. The monomial example with $p = 1$ is illustrated in Figure 6.2. The K^{en} coefficients in each element's enhanced solution U^{en} are constrained as follows. Let $\{\tau_0, \tau_1, \tau_2, \tau_3\}$ be the four edges that border a given element Ω_e. Each edge is populated by a primary recovered solution f. We take $L^k(s_q)$ to be the k^{th} Legendre polynomial evaluated at coordinate $s_q \in [-1, 1]$ along a given interface τ_q. The enhanced solution is weakly equivalent to U_e^h over the element interior and weakly equivalent to the appropriate recovered function f over each of the four bordering

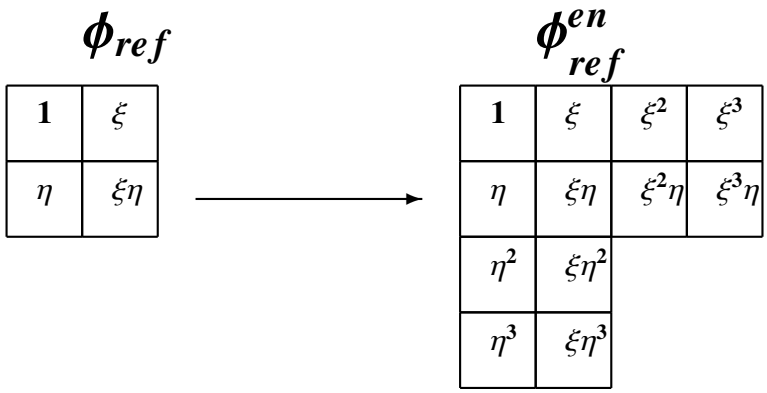


Figure 6.2: Basis construction for U^h and U^{en}; the reference element's basis is shown.

edges:

$$\int_{\Omega_e} U_e^{en}\phi_e^k d\boldsymbol{x} = \int_{\Omega_e} U_e^h\phi_e^k d\boldsymbol{x}, \quad \forall\phi_e^k \in \boldsymbol{\phi_e}, \tag{6.3a}$$

$$\int_{\tau_q} U_e^{en}L_s^k(s_q)ds_q = \int_{\tau_q} f_q L_s^k(s_q)ds_q, \quad \forall k \in \{0,1,...,p\} \quad , \forall q \in \{1,2,3,4\}. \tag{6.3b}$$

The system of Eq. (6.3) gives K constraints based on the DG approximation U_e^h of the local element; additionally, each of the four interfaces contribute $p+1$ unique constraints. As with the recovered solution constraints (Section 2.6), this system can be written as a linear matrix-vector equation that yields the K^{en} coefficients $\hat{\mathbf{U}}_\mathbf{e}^\mathbf{en}$ given the K DOFs $\hat{\mathbf{U}}_\mathbf{e}$ and the distribution of f along τ_1, τ_2, τ_3, and τ_4. The enhanced solution is then used to populate the viscous flux function over the interior of the local element: $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}(U_e^{en}, \nabla^h U_e^{en})$ throughout Ω_e.

To finish populating the DG weak form (Eq. 6.1), the flux along element-element interfaces must be populated. Early versions of the Recovery DG family used the gradient of the primary recovered function to populate the viscous flux along interfaces, but that approach is inconsistent for $p > 1$, $\theta > 0$, as discussed by Lo [67]. Instead, the RDG scheme uses a second iteration of Recovery to build another recovered solution over the interface. After primary recovery and solution enhancement have been performed, RDG performs recovery on the enhanced solution U^{en} in order to produce the secondary recovered function, f^{en}, across each interface. The interface flux $\tilde{\boldsymbol{\mathcal{G}}}$ is then populated as $\tilde{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}(f^{en}, \nabla f^{en})$.

There is an important peculiarity to the secondary recovery. The *CO* in the full scheme name

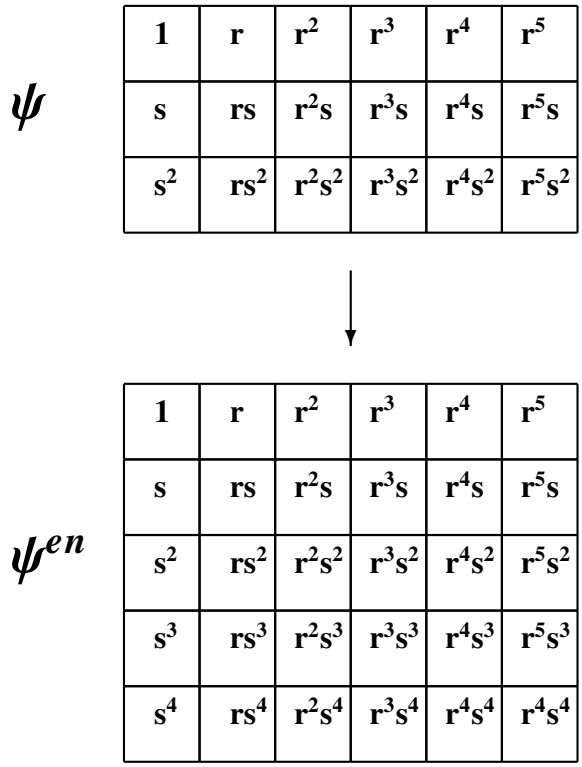


Figure 6.3: Basis construction for f and f^{en} in the Cartesian-optimized approach. The face-normal recovery coordinate is r and the face-tangential recovery coordinate is s.

(RDG-1x++CO) is short for *Cartesian Optimization.* Without Cartesian Optimization, the secondary recovery would consist of setting $2K^{en}$ degrees of freedom for the solution f^{en} across each interface, using K^{en} DOFs from each of elements that share the interface. Instead, each secondary recovered solution f^{en} contains $2(p+1)(p+3)$ coefficients; the enhanced recovery basis is degree $2p+1$ in the face-normal direction and degree $p+2$ in the face-tangential direction. This difference between the primary recovery basis $\boldsymbol{\psi}$ and the enhanced recovery basis $\boldsymbol{\psi^{en}}$ is illustrated in Figure 6.3 for the $p = 2$ case. The procedure for setting the coefficients in the enhanced recovered solution for a given interface depends on the orientation of the interface. Let $(\overline{x_e}, \overline{y_e})$ be the centroid of Ω_e and Δx_e, Δy_e to be the element widths in the x and y directions, respectively. For this element, define

$$X_e(x) = \frac{2}{\Delta x_e}(x - \overline{x_e}) \quad , \quad Y_e(y) = \frac{2}{\Delta y_e}(y - \overline{y_e}) \tag{6.4}$$

as transformations from the physical coordinates (x, y) to local element coordinates $(X, Y) \in [-1, 1]^2$. Let $L^k(\zeta)$ to be the 1D Legendre polynomial of degree k and assume the interface's normal to be

in the x direction. The recovery constraints for the enhanced recovered solution are as follows for a given pair of neighboring elements Ω_A and Ω_B:

$$\int_{\Omega_A} L^{kx}(X_A(x))\, L^{ky}(Y_A(y))\, (U_A^{en} - f_{AB}^{en})\, d\boldsymbol{x} = 0, \quad \forall (kx, ky) \in \{\{0, 1, ..., p\} \times \{0, 1, ..., p+2\}\}, \quad (6.5a)$$

$$\int_{\Omega_B} L^{kx}(X_B(x))\, L^{ky}(Y_B(y))\, (U_B^{en} - f_{AB}^{en})\, d\boldsymbol{x} = 0, \quad \forall (kx, ky) \in \{\{0, 1, ..., p\} \times \{0, 1, ..., p+2\}\}, \quad (6.5b)$$

where f_{AB}^{en} is the secondary recovered solution over $\Omega_A \cup \Omega_B$. If instead the interface normal is in the y direction, the constraints for f^{en} take the following form:

$$\int_{\Omega_A} L^{kx}(X_A(x))\, L^{ky}(Y_A(y))\, (U_A^{en} - f_{AB}^{en})\, d\boldsymbol{x} = 0, \quad \forall (kx, ky) \in \{\{0, 1, ..., p+2\} \times \{0, 1, ..., p\}\}, \quad (6.6a)$$

$$\int_{\Omega_B} L^{kx}(X_B(x))\, L^{ky}(Y_B(y))\, (U_B^{en} - f_{AB}^{en})\, d\boldsymbol{x} = 0, \quad \forall (kx, ky) \in \{\{0, 1, ..., p+2\} \times \{0, 1, ..., p\}\}. \quad (6.6b)$$

Once the secondary recovered solution has been calculated, it is differentiated to populate the viscous flux on each interface. The RDG scheme maintains greater than $3p$ order of accuracy (from Fourier analysis, see Table 3.9) on Cartesian elements. However, due to the secondary (enhanced) recovery step, the overall update scheme for a single element is non-compact (see Figure 2.5).

6.4 Boundary Procedures

Consider an initial boundary value problem (IBVP) involving either a Dirichlet or Neumann boundary condition along the entirety of $\boldsymbol{\partial\Omega}$ with the 2D shear diffusion equation being the governing equation:

$$U(x, y, 0) = U_{IC}(x, y) \quad \forall (x, y) \in \boldsymbol{\Omega}, \quad (6.7a)$$

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}(\mu \frac{\partial U}{\partial x} + \mu\theta \frac{\partial U}{\partial y}) + \frac{\partial}{\partial y}(\mu\theta \frac{\partial U}{\partial x} + \mu \frac{\partial U}{\partial y}) + \mathcal{S}(x, y, t), \quad \forall (x, y, t) \in \boldsymbol{\Omega} \times (0, t_f], \quad (6.7b)$$

$$\left\{ \begin{array}{ll} U = C_D & , \quad \text{for Dirichlet} \\ \nabla U \cdot \mathbf{n} = C_N, & \quad \text{for Neumann} \end{array} \right\}, \quad \forall (x, y, t) \in \boldsymbol{\partial\Omega} \times (0, t_f], \quad (6.7c)$$

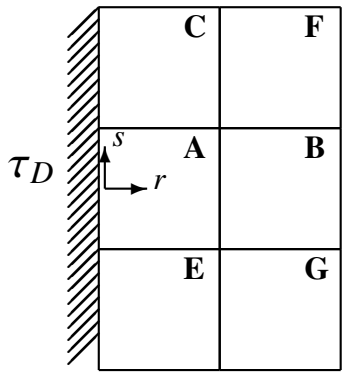


Figure 6.4: The Boundary Environment.

where C_D is a space-time dependent Dirichlet condition, C_N is a space-time dependent Neumann condition, and the problem is simulated to time t_f. We constructed and tested a set of schemes for handling either Dirichlet or Neumann boundary conditions.

The set of boundary schemes begins with Lo's *full boundary recovery* approach [67]. We label this approach as the **F∅** procedure; the F stands for full boundary recovery, and the ∅ is used to show that no special treatment is made for tangential derivatives along the boundary. Consider the situation illustrated in Figure 6.4. The element Ω_A, referred to as the boundary element, shares at least one interface with a Dirichlet or Neumann boundary, $\boldsymbol{\partial\Omega}$. Let $\tau_D = \partial\Omega_A \cap \boldsymbol{\partial\Omega}$ be known as the *boundary interface*. The face-tangential recovery coordinate s spans the domain $s \in [-1, 1]$ over τ_D. Let Ω_B be the element that touches Ω_A opposite the boundary interface. For example, if the physical domain is a square with Dirichlet boundaries and Ω_A touches the left-side boundary of $\boldsymbol{\Omega}$, as shown in Figure 6.4, then Ω_B is the element immediately to the right of Ω_A. The element Ω_B is known as the *interior element*.

The boundary interface is associated with a recovered polynomial, denoted f_{AD}. In addition to weak equivalence relations over Ω_A and Ω_B, full boundary recovery demands that f_{AD} be in agreement with the prescribed boundary condition along τ_D. To preserve accuracy, the recovered function f_{AD} is a polynomial expansion with $2K$ coefficients. These coefficients are constrained by the boundary element, the interior element, and the boundary condition (Either Neumenn or

Dirichlet) as follows:

$$\int_{\Omega_A} f\phi_A^k d\boldsymbol{x} = \int_{\Omega_A} U_A^h \phi_A^k d\boldsymbol{x}, \quad \forall k \in \{0, 1, ..., K-1\}, \tag{6.8a}$$

$$\int_{\Omega_B} f\,\beta_B^k d\boldsymbol{x} = \int_{\Omega_B} U_B^h \beta_B^k d\boldsymbol{x}, \quad \forall k \in \{0, 1, ..., K-1-(p+1)\}, \tag{6.8b}$$

$$\left\{ \begin{array}{ll} \int_{\tau_D} f L^k(s) ds = \int_{\tau_D} C_D(s) L^k(s) ds, & \text{for Dirichlet} \\ \int_{\tau_D} (\nabla f \cdot \mathbf{n}) L^k(s) ds = \int_{\tau_D} C_N(s) L^k(s) ds, & \text{for Neumann} \end{array} \right\}, \quad \forall k \in \{0, 1, ..., p\}, \tag{6.8c}$$

where $\mathbf{n}$ is the outward normal along τ_D and L^k is the 1D Legendre polynomial of degree k. There is one set of $p+1$ constraints that is enforced in the case of a Dirichlet problem and a different set of $p+1$ constraints in the case of a Neumann problem. The functions β^k are members of $\boldsymbol{\phi_B}$. If the boundary interface is normal to the x direction, then the functions β^k should exclude the highest basis modes of $\boldsymbol{\phi_B}$ in the x direction, and if the boundary interface is normal to the y direction, then the functions β^k exclude the highest basis modes of $\boldsymbol{\phi_B}$ in the y direction. This procedure performs well if the IBVP does not involve derivatives in the interface-tangential direction. If the tangential derivative along a boundary is needed, which could be the case for the Navier-Stokes equations (where a velocity profile may be prescribed along a boundary), then the **F∅** method performs poorly. Our two proposed methods are attempts to remedy this shortcoming.

It is possible to perform secondary recovery along a boundary using a similar approach to that shown in Equation (6.8), forcing equivalence with the enhanced cell solutions U^{en} rather than U^h; this method is abbreviated **FF**, standing for full primary recovery, then full secondary recovery along the boundary. It yields an enhanced recovery solution, f_{AD}^{en}, that is of high polynomial degree in both the boundary-normal and boundary-tangential directions.

We also propose the **FM** scheme, for full primary recovery along the boundary, and a mixed approach for populating the gradient information. It consists of three steps:

1 : Perform full primary boundary recovery procedure to obtain f_{AD}.

2 : Perform cell solution enhancement to obtain U_A^{en}.

3 : Along τ_D, use ∇f_{AD} for the boundary-normal derivative and ∇U_A^{en} for the boundary-tangential derivative. These derivative choices are used to calculate the flux $\mathcal{G}$ along τ_D.

The **F∅**, **FF**, and **FM** schemes are distinguished from eachother by the amount of data pulled in from neighboring elements. With respect the element addresses in Figure 6.4, the **F∅** scheme uses only the information contained in the boundary condition and the two DOF vectors $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$. By employing the enhanced solution in Ω_A, the **FM** scheme makes use of $\hat{\mathbf{U}}_\mathbf{A}$, $\hat{\mathbf{U}}_\mathbf{B}$, $\hat{\mathbf{U}}_\mathbf{C}$, $\hat{\mathbf{U}}_\mathbf{E}$, and the boundary condition. Finally, by performing recovery over the enhanced solutions in Ω_A and Ω_B, the **FF** scheme makes use of the boundary condition, $\hat{\mathbf{U}}_\mathbf{A}$, $\hat{\mathbf{U}}_\mathbf{B}$, $\hat{\mathbf{U}}_\mathbf{C}$, $\hat{\mathbf{U}}_\mathbf{E}$, $\hat{\mathbf{U}}_\mathbf{F}$, and $\hat{\mathbf{U}}_\mathbf{G}$. We hypothesized that the accuracy of the boundary scheme would improve as more information was added to the stencil. Under this hypothesis, the **F∅** scheme should be the least accurate and the **FF** scheme should be the most accurate.

6.5 Numerical Test: Dirichlet/Neumann Boundary Conditions

The boundary schemes are evaluated with a simple test case. The IBVP (Eq. 6.7) is simulated with a nonzero source term $\mathcal{S}$ implemented to force $U(x, y, t)$ to a manufactured solution. The flux parameters and the manufactured solution are shown in Eq. (6.9).

$$\mu = 1 + \frac{U^2}{10} \quad , \quad \theta = 0.25 \quad , \quad k_w = 2\pi, \tag{6.9a}$$

$$U(x, y, t) = e^{-{k_w}^2 t}(sin(k_w(x - y)) + sin(k_w x)sin(k_w y)). \tag{6.9b}$$

The spatial domain is the unit square, partitioned by uniform square elements. Each simulation runs from $t = 0$ to $t = 2k_w^{-2}$. The problem is simulated under periodic boundary conditions, then under Dirichlet boundary conditions, then under Neumann boundary conditions; the Dirichlet and Neumann boundary conditions (C_D and C_N) are taken from the exact solution for U in Eq (6.9). For each of the three boundary schemes discussed (**F∅**, **FF**, **FM**), the test problem is solved on a series of succesively finer meshes, each characterized by the uniform element edge length $h = \Delta x = \Delta y$.

Under periodic boundary conditions, the RDG scheme achieves order $3p+1$ or greater convergence in the cell-average error, E_{CA}:

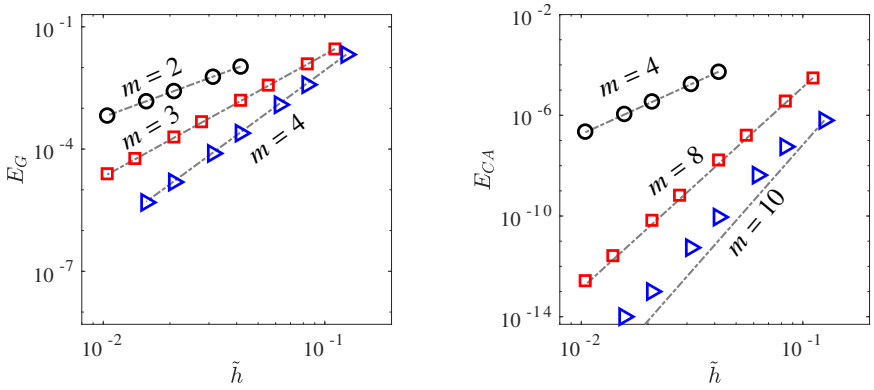
$$E_{CA} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}(\overline{U^h}_m - \overline{U}_m)^2}, \tag{6.10}$$

where $\overline{U^h}_m$ and $\overline{U}_m$ are the averages of the polynomial U^h_m and the exact solution, respectively, over Ω_m. It also achieves the optimal convergence rate in the global L_2 error. These behaviors are demonstrated in Figure 6.5. The goal of each of the boundary schemes presented (**F∅**, **FF**, and **FM**) is to match the orders of convergence observed in the spatially periodic case. If this goal is not achieved, the extraordinary accuracy of the RDG scheme over the domain interior (and the associated computational effort) goes to waste.

Figure 6.6 presents a mesh refinement study of the IBVP under Dirichlet boundary conditions. For the $p = 1$ case, all three boundary schemes are effective, achieving 4^{th} order convergence. For $p > 1$, only the **FM** scheme achieves the desired behavior, replicating the orders of convergence observed in the spatially periodic case. This result disagrees with our initial hypothesis; we expected the **FF** scheme to be the most accurate, but ultimately, the extra information employed by the **FF** scheme pollutes the estimate of the gradient along the boundary. Figure 6.7 presents a mesh refinement study of the IBVP under Neumann boundary conditions. For the $p = 1$ case, the **FF** and **FM** boundary schemes are effective. For $p > 1$, none of the proposed boundary schemes are capable of preserving the desired order of convergence, but the **FM** scheme is the best of the three and maintains greater than order $2p$ convergence in the cell-average error. The simple **F∅** scheme shows poor performance for this problem, achieving only order p convergence.

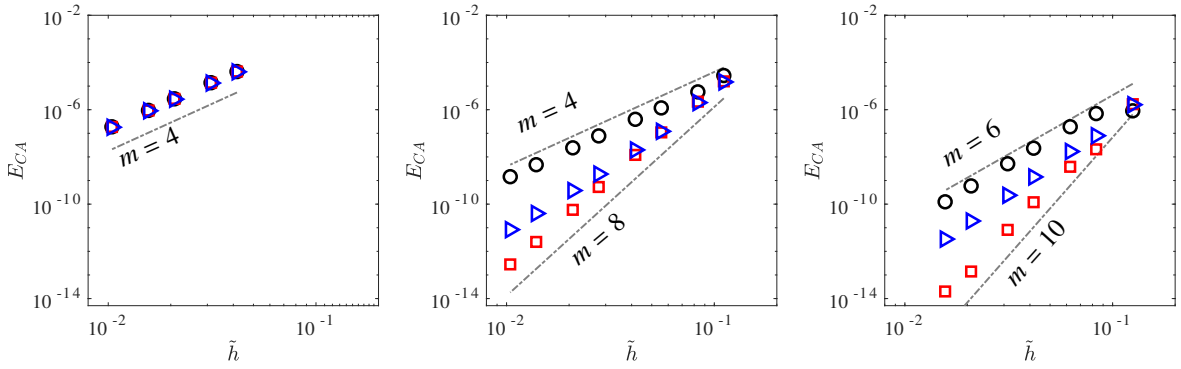
6.6 Chapter Conclusion

We studied three distinct schemes for accommodating Dirichlet and Neumann boundary conditions in the Recovery-based DG framework. The first scheme (**F∅**) is the original work of Lo [67] and the other two schemes (**FF**, **FM**) are our own novel inventions. The numerical test demonstrated that to



(a) Global L_2 error. (b) Cell-average error.

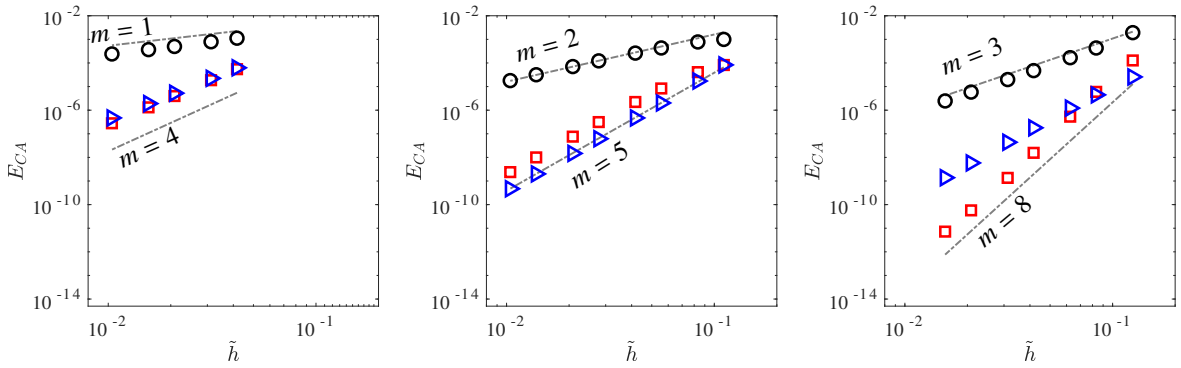
Figure 6.5: Convergence study for the intial value problem with periodic boundary conditions. Dashed gray lines denote approximate convergence rates m.
Symbol Key: $p = 1$:○ ; $p = 2$:▷ ; $p = 3$:□.



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 6.6: Convergence study in cell-average error for Dirichlet IBVP. Dashed gray lines denote approximate convergence rates m. Symbol Key: **F∅**:○ ; **FF**:▷ ; **FM**:□.

minimize the degradation in accuracy when moving from spatially periodic boundary conditions to either Dirichlet or Neumann boundary conditions, our proposed **FM** scheme is the best option. The scheme draws in the ideal amount of information from the elements near a given boundary interface to maximize accuracy. Overall, the Recovery-based DG scheme is an exceptionally accurate tool for the discretization of purely parabolic PDE problems on a Cartesian mesh.



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 6.7: Convergence study in cell-average error for Neumann IBVP. Dashed gray lines denote approximate convergence rates m. Symbol Key: **F∅**:○ ; **FF**:▷ ; **FM**:□.

CHAPTER 7

Recovery in the Flux Reconstruction Method

7.1 Chapter Overview

The Flux Reconstruction (FR) method is an alternative to the discontinuous Galerkin (DG) approach but shares many properties of DG. Similar to the DG method, FR makes use of a set of piecewise polynomials to approximate the solution, allowing high-order accuracy on nontrivial mesh geometries. Where the DG approach is based on satisying the governing differential equation (GDE) in the weak sense, the FR approach is built to satisfy the GDE in differential form at a discrete set of points, known as solution points, over each element. We had the opportunity to work with Dr. H.T. Huynh, the creator of the FR method [46, 47], during the summer of 2018 to analyze various FR approaches for diffusion problems. In this chapter, the FR method is described in the context of a 2D advection-diffusion problem, but we eventually restrict ourselves to pure diffusion problems. Then, it will be shown that the familiar Fourier analysis technique, which saturates the earlier chapters of this work, can be applied to nonuniform mesh geometries. A recovery-assisted FR approach is described and analyzed. As one would expect, the recovery-assisted approach provides superior performance on Cartesian meshes; additionally, it remains stable on an irregular arrangement of simplex elements. The FR method is closely related to the DG method, so many of the operations detailed in this chapter have DG counterparts in Chapter 3 and Chapter 2.

7.1.1 Novelty and Articles

There are three novel components in this chapter. The most significant contribution is the extension of the Fourier analysis technique to nonuniform 2D meshes, which to our knowledge has never before been achieved. Second, the extension of the recovery-assisted philosophy to the FR method yields a new numerical scheme for diffusion problems. Third, this chapter shows how the *I-continuous* approach of Huynh [47], known by some as "poor-man's recovery," performs on simplex elements in 2D.

The material of this chapter appears in one AIAA conference manuscript, to be presented at the 2019 AIAA Aviation conference:

- P. E. Johnson, E. Johnsen, & H. T. Huynh, *A Novel Flux Reconstruction Method for Diffusion Problems*, AIAA 2019 Aviation Forum.

7.1.2 Usage of Recovery

The biased recovery approach makes no appearance in this chapter. Instead, the recovery-assisted FR scheme employs the classical, full-order recovery operator (Section 2.6.2) in the inner-product implementation. The operation is applied on quads and simplices in 2D as detailed in Section 2.6.5.

7.2 The Flux Reconstruction Method

In this section, we describe the FR method in the context of an arbitrary advection-diffusion system,

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{Q} = 0 \quad \text{with} \quad \boldsymbol{Q} = \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{G}}, \tag{7.1}$$

where $U(\boldsymbol{x}, t)$ is the space and time dependent state variable, $\boldsymbol{\mathcal{F}}(U)$ is the advective flux component, and $\boldsymbol{\mathcal{G}}(U, \nabla U)$ is the diffusive flux component. The focus is the 2D implementation of FR, so the notation and methodology in this chapter are particular to the 2D case. A given FR approach is defined by a variety of parameters, including: the polynomial degree p of the solution in each

element, the element shape, the choice of correction functions/fields for gradient calculations, and the strategy for defining ambiguous element-element interface terms. Much of the terminology is the same as in the DG method because the DG and FR approaches are closely related.

7.2.1 Preliminaries

7.2.1.1 Domain Decomposition

Given some spatial domain $\mathbf{\Omega}$ on which a solution to Eq. (7.1) is to be calculated, the domain is split into M non-overlapping elements, each labeled Ω_m (with closure $\overline{\Omega}_m = \Omega_m \cup \partial\Omega_m$, where $\partial\Omega_m$ is the boundary of Ω_m), as with the DG method.

7.2.1.2 Solution Representation

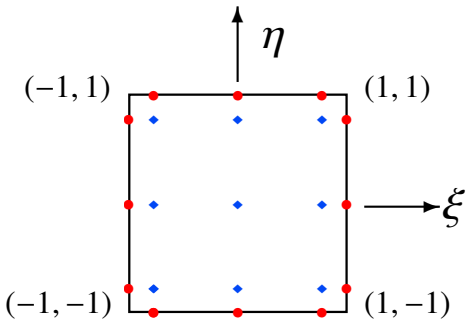
As with the DG method, the approximate numerical solution to Eq. (7.1) is denoted $U^h(\boldsymbol{x}, t)$ and is a piecewise-continuous polynomial of degree at most p in each element. Over each element, U^h is a linear combination of K time-dependent degrees of freedom (DOFs), denoted $\hat{U}_m^k(t)$, and K space-dependent basis functions, denoted $\phi_m^k(\boldsymbol{x})$:

$$U^h(\boldsymbol{x} \in \Omega_m, t) = U_m^h(\boldsymbol{x}, t) = \sum_{k=1}^{K} \hat{U}_m^k(t)\phi_m^k(\boldsymbol{x}). \tag{7.2}$$

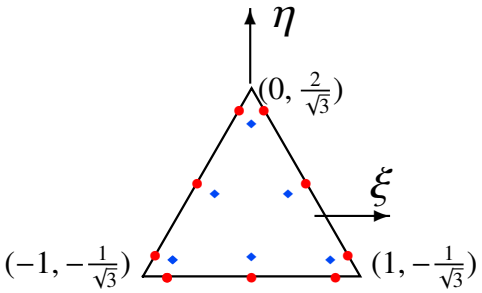
The FR method requires a nodal basis. The corresponding basis functions $\boldsymbol{\phi}$ are fully explained in Section 7.2.1.5.

7.2.1.3 The Reference Element

Given the choice of either quadrilateral or triangular elements to partition the spatial domain, each physical element Ω_m is mapped to the single reference element, Ω_{ref}. The reference element for quadrilaterals is the bi-unit square, and the reference element for triangles is an equilateral triangle with side length 2; these reference elements are shown in Figure 7.1. The reference elements exist in the reference domain, which is spanned by orthogonal coordinates ξ and η instead of the physical



(a) Reference Quadrilateral ($N_V = 4$).



(b) Reference Triangle ($N_V = 3$).

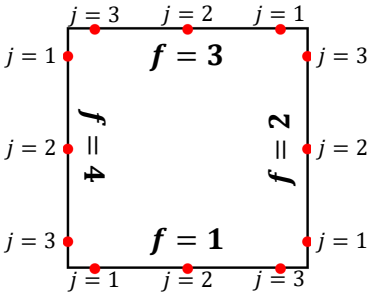
Figure 7.1: Reference elements in the $p = 2$ case. Solution points are denoted by blue diamonds (♦) and interface flux points by red circles (●). Both elements have centroid $(\xi, \eta) = (0, 0)$, and the N_V vertices of each element are listed in reference coordinate space. Each basis function ℓ_k is unity at a particular solution point and zero at all other solution points.

coordinates x and y. Each reference element has its centroid at $\boldsymbol{\xi} = (0, 0)$, as shown in Figure 7.1.

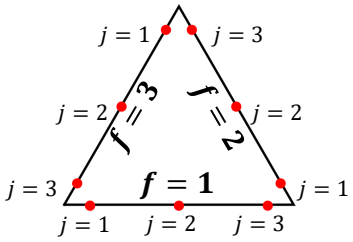
The reference element is spanned by a K-dimensional solution basis $\boldsymbol{\ell}$; each member of this basis is a polynomial that is unity at a particular "solution point" on the reference element and zero at all other solution points. Letting $\boldsymbol{\xi}_k$ denote the k^{th} solution point on Ω_{ref}, the solution basis is defined as follows:

$$\ell^n(\boldsymbol{\xi}_k) = \left\{ \begin{array}{ll} 1 & \text{for } k = n \\ 0 & \text{for } k \neq n \end{array} \right\}, \tag{7.3}$$

with each ℓ^k being a polynomial of at most degree p in each direction on Ω_{ref}. The distribution of the solution points on the reference element is a matter of user preference; in this work, we use a tensor product grid of the $p + 1$ Gauss-Legendre points in each direction for the square reference element and the strong quadrature points tabulated by Taylor et al. [98] for the reference triangle. Figure 7.1 shows the distribution of solution points in the $p = 2$ case. As with the DG method, the geometrical Jacobian matrix (Section 2.5.1) is employed to obtain gradients of the basis functions with respect to physical coordinates.

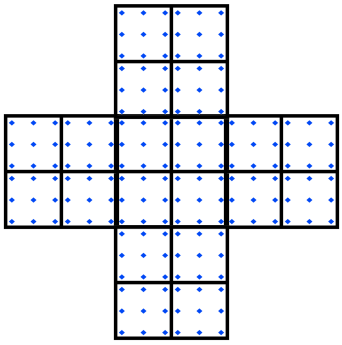


(a) Reference Quadrilateral ($N_V = 4$).

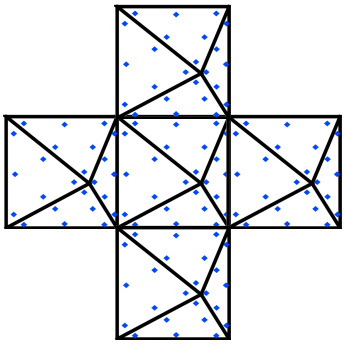


(b) Reference Triangle ($N_V = 3$).

Figure 7.2: Flux point distribution in the $p = 2$ case along the perimeter of the reference element; the flux points are numbered counterclockwise along each face. The face index, f, and the point index, j, correspond to the definition of the correction field ψ_{fj}.



(a) Cartesian Grid.



(b) Perturbed Triangles.

Figure 7.3: The grids used for Fourier analysis. In addition to the element boundaries, we have plotted the solution point locations in the $p = 2$ case; note that a $p2$ triangle has fewer points than a $p2$ quadrilateral.

7.2.1.4 Solution Points and Flux Points

Each physical element is inhabited by K *solution points*. The K solution points on each Ω_m are defined as follows:

$$\boldsymbol{x}_m^k = \boldsymbol{x}(\boldsymbol{\xi}_k), \tag{7.4}$$

for $k \in \{1, 2, ..., K\}$, where $\boldsymbol{\xi}_k$ is the k^{th} solution point on Ω_{ref}, $\boldsymbol{x}(\boldsymbol{\xi})$ is the mapping corresponding to Ω_m, and $\boldsymbol{x}_m^k$ is the k^{th} solution point on Ω_m. The distribution of physical solution points for the two grids employed for Fourier analysis is illustrated in Figure 7.3.

In addition to the solution points, each edge of each element is populated by $p+1$ *flux points*. Thus, each element is associated with $N_V \times (p+1)$ flux points. The distribution of flux points is a matter of user preference; in this work, the flux points are the $p+1$ Gauss-Legendre points on each edge, as illustrated in Figure 7.2. These flux points serve a similar role to the interface quadrature points in the DG method.

7.2.1.5 Solution Basis

The nodal solution basis $\boldsymbol{\phi_m}$ in a given element Ω_m is defined as follows:

$$\phi_m^k(\mathbf{x}(\boldsymbol{\xi})) = \ell^k(\boldsymbol{\xi}) \quad \forall k \in \{1, 2, ..., K\}, \tag{7.5}$$

where the one-to-one mapping $\boldsymbol{x}(\boldsymbol{\xi})$ is available directly from the element geometry. Each element's basis inherits the Kronecker delta property from the reference element:

$$\phi_m^n(\boldsymbol{x_m^k}) = \left\{ \begin{array}{ll} 1 & \text{for } k = n \\ 0 & \text{for } k \neq n \end{array} \right\}. \tag{7.6}$$

Additionally, each ϕ_m^k is equal to zero outside of Ω_m; along $\partial\Omega_m$, each ϕ_m^k is in general multivalued, as it has a nonzero limit from inside Ω_m and the limit is zero from outside Ω_m.

The FR method mandates that the gradient of the solution basis be available as well. At a given point in space, the gradient is calculated as follows:

$$\begin{bmatrix} \phi_{m,x}^k \\ \phi_{m,y}^k \end{bmatrix} (\boldsymbol{x}(\boldsymbol{\xi})) = [\underline{J_m^{-1}}]^T \begin{bmatrix} \ell_{,\xi}^k \\ \ell_{,\eta}^k \end{bmatrix} (\boldsymbol{\xi}), \tag{7.7}$$

where the Jacobian matrix and the right-hand-side gradient w.r.t. the reference coordinates are evaluated at $\boldsymbol{\xi}$ and the gradient w.r.t. the physical coordinates is given at $\mathbf{x}(\boldsymbol{\xi})$. For cases where the Jacobian is non-constant over an element, it is crucial that the variation of $\underline{J}$ be properly accounted for in Eq. (7.7). The transformation to the physical gradient can also be conveniently written in

terms of the individual components of the Jacobian:

$$\begin{bmatrix} \phi^k_{m,x} \\ \phi^k_{m,y} \end{bmatrix} (\mathbf{x}(\boldsymbol{\xi})) = \frac{1}{|J|} \begin{bmatrix} y_{,\eta} \ell^k_{,\xi} - y_{,\xi} \ell^k_{,\eta} \\ x_{,\xi} \ell^k_{,\eta} - x_{,\eta} \ell^k_{,\xi} \end{bmatrix} (\boldsymbol{\xi}), \tag{7.8}$$

where the determinant of the Jacobian, as well as the metric terms on the right-hand side, should be evaluated at $\boldsymbol{\xi}$. This approach can be applied to recast the governing differential equation (Eq. 7.1) in terms of the "transformed flux" rather than the physical flux; see the work of Castonguay et al. [20] as an example. In this work, we do not make explicit use of the transformed flux but instead directly obtain the divergence of the physical flux in the physical frame.

The "broken gradient" and "broken divergence" operators must also be defined. For a given element Ω_m, let $V(\mathbf{x})$ be some scalar quantity built as a linear combination using nodal coefficients $\hat{V}^k_m$ in the FR solution space:

$$V(\boldsymbol{x}) = \sum_{k=1}^{K} \hat{V}^k_m \phi^k_m(\boldsymbol{x}). \tag{7.9}$$

The broken gradient of V is defined as follows:

$$\nabla^h V = \begin{bmatrix} \sum_{k=1}^K \hat{V}^k_m \frac{\partial}{\partial x} \phi^k_m \\ \sum_{k=1}^K \hat{V}^k_m \frac{\partial}{\partial y} \phi^k_m \end{bmatrix}; \tag{7.10}$$

the x and y derivatives of the FR solution basis are used to form a gradient approximation. This broken gradient is the same term employed to form the compact members of the IGR family in Chapter 3.

Suppose instead that $\boldsymbol{V}$ is a vector entity with an x component u and a y component v such that

$$\boldsymbol{V}(\boldsymbol{x}) = \begin{bmatrix} u(\boldsymbol{x}) \\ v(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^K \hat{u}^k_m \phi^k_m(\boldsymbol{x}) \\ \sum_{k=1}^K \hat{v}^k_m \phi^k_m(\boldsymbol{x}) \end{bmatrix}. \tag{7.11}$$

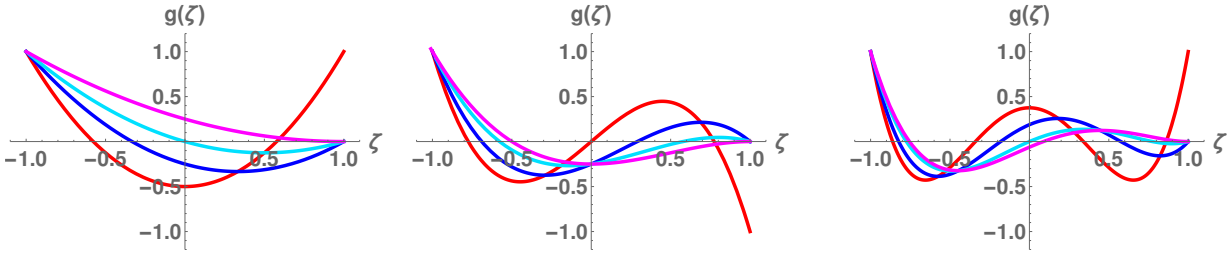
In this case, the broken divergence operator is

$$\nabla^h \cdot \boldsymbol{V} = \sum_{k=1}^{K} \hat{u}_m^k \frac{\partial}{\partial x} \phi_m^k + \sum_{k=1}^{K} \hat{v}_m^k \frac{\partial}{\partial y} \phi_m^k. \tag{7.12}$$

Note that the broken gradient operator maps a scalar function to a vector function and that the broken divergence operator maps a vector function to a scalar function. Also, these functions are defined only for polynomial inputs in the FR solution space; the broken gradient/divergence cannot be applied to an arbitrary function.

7.2.1.6 Correction Polynomials in 1D

The implementation of FR on quadrilateral elements depends on a set of 1D functions known as "correction polynomials" defined on the bi-unit interval, $\zeta \in [-1, 1]$. As described by Huynh [46], each correction polynomial is unity at the left boundary and approximates zero over the element interior; with the exception of the Legendre polynomial, each of these correction polynomials is zero at the right-side edge of the reference element. For an FR scheme of order p, four different 1D correction polynomials are utilized in this work. The first is the degree $p+1$ Radau polynomial, abbreviated $g_{DG}(\zeta)$ because it corresponds to a differential formulation of the DG method. The second is the Lagrange polynomial that is unity at $\zeta = -1$, zero at $\zeta = 1$, and zero at the p Gauss-Legendre points over the element interior, abbreviated $g_{SD}(\zeta)$ because it facilitates a spectral-difference (SD) approach. The third is the special function of Huynh [46] that is unity at $\zeta = -1$ and zero at $\zeta = 1$. Additionally, the first derivative of Huynh's function is zero at the the set of $p + 1$ Lobatto points, excluding the left boundary point. We abbreviate Huynh's function $g_{Hu}(\zeta)$. The remaining correction function, which we refer to as the "zeroth" correction polynomial, is the degree $p + 1$ Legendre polynomial multiplied by $(-1)^{p+1}$ so that it is unity at $\zeta = -1$, abbreviated $g_{Le}(\zeta)$. All four correction polynomials are illustrated in Figure 7.4 for $p = 1$, $p = 2$, and $p = 3$.



(a) $p = 1$. (b) $p = 2$. (c) $p = 3$.

Figure 7.4: Correction polynomials on the 1D bi-unit domain. Plotted functions are g_{Le} **(red)**, g_{DG} **(blue)**, g_{SD} **(turquoise)**, and g_{Hu} **(magenta)**.

7.2.2 Spatial Discretization of the Governing Differential Equation

Given the collection of all DOFs in the spatial domain at timestep s, i.e. $t = t^s$, the FR spatial discretization is used to estimate the flux divergence $(\nabla \cdot \boldsymbol{Q})$ in Eq. (7.1) at each solution point $\boldsymbol{x}_m^k$. Then, since $\frac{\partial U}{\partial t} = -\nabla \cdot \boldsymbol{Q}$, each nodal DOF $\hat{U}_m^k$ is directly updated as follows:

$$\frac{\hat{U}_m^k(t^s + \Delta t) - \hat{U}_m^k(t^s)}{\Delta t} = -(\nabla \cdot \boldsymbol{Q})|_{\boldsymbol{x}_m^k}, \tag{7.13}$$

where Δt is the timestep size; the forward Euler method is shown but a more sophisticated technique (such as high-order Runge-Kutta) is immediately applicable. The fundamental difference between DG and FR is that while the DG method is built on a set of integral equations, the FR approach mandates that the differential form of the conservation law be satisfied at a discrete set of points. A similar split is present between the pseudospectral and classical Galerkin forms of Fourier/Chebyshev spectral methods. In fact, the seeds of the FR method are present in the spectral multidomain method of Kopriva [62], which seeks to improve the parallel efficiency of the global pseudospectral method by breaking the global domain $\boldsymbol{\Omega}$ into a set of subdomains (one might say elements), then applying a Chebyshev pseudospectral method over each subdomain. See De Grazia et al. [29] for a clear explanation of the similarities and differences between the FR and DG methods, highlighting the split between the Galerkin approach and the differential approach. See also Yu et al. [115] for detailed discussion on the computational efficiency of the FR and DG methods. It remains to describe how the flux divergence is approximated.

In this work, we follow the general energy-stable (ESFR) framework proposed by Castonguay et al. [20]. The much simpler approach of Huynh [46, 47] can be employed on quadrilateral elements, but since the objective here is to provide Fourier analysis on both quadrilateral and simplex elements, we begin with the more general approach of Castonguay et al. [20]. The overall philosophy of flux reconstruction is simple: given the nodal (solution point) values of the flux $\boldsymbol{Q}$ in an element, a *flux polynomial* is formed to approximate $\boldsymbol{Q}(\boldsymbol{x})$ over the element. Then, the flux polynomial is differentiated (as with a finite-difference approach) using the element's solution basis. To allow elements to exchange information, special correction terms are added to the calculated flux divergence at each solution point. Given the corrected flux divergence at each solution point, the temporal derivative is immediately available, allowing the DOFs to be updated according to Eq. (7.13).

In the remainder of this section, we first show how to calculate the flux divergence based on the known distribution of U and ∇U. Then, we show the proper methodology for approximating ∇U.

7.2.2.1 Flux Divergence

For a given element Ω_m at a given time t, assume the nodal solution DOFs $\hat{\mathbf{U}}_\mathbf{m}$ to be known, and assume that a gradient approximation of the following form is known over the element:

$$\nabla U(\boldsymbol{x} \in \Omega_m) \approx \boldsymbol{\sigma_m}(\boldsymbol{x}) = \sum_{k=1}^{K} \hat{\boldsymbol{\sigma}}_m^k \phi_m^k(\boldsymbol{x}), \tag{7.14}$$

where $\hat{\boldsymbol{\sigma}}_{\boldsymbol{m}}^{\boldsymbol{k}}$ is an approximation for ∇U at solution point $\boldsymbol{x}_m^k$. Additionally, suppose that for each flux point along $\partial\Omega_m$, a gradient approximation $\tilde{\boldsymbol{\sigma}}$ has been calculated; for consistency, both elements sharing an interface must make use of the same $\tilde{\boldsymbol{\sigma}}$ value at each flux point, so it is known as the "common gradient." As described by Huynh [46] and Castonguay et al. [20], the FR method consists of five steps to calculate $\frac{\partial U}{\partial t}$ at each solution point in Ω_m, enumerated in Table 7.1. With the overall methodology explained, we now make some clarifications regarding the algorithm of Table 7.1.

7.2.2.2 Discontinuous Flux Polynomial

Just as the approximate solution U^h exists in the solution space $\boldsymbol{\phi}$, the spatial distribution of the flux is also defined as a polynomial in $\boldsymbol{\phi}$. Let

$$\hat{\boldsymbol{\mathcal{F}}}^{\boldsymbol{k}}_{\boldsymbol{m}} = \boldsymbol{\mathcal{F}}(U^h_m(\boldsymbol{x}^k_m)) \quad \text{and} \quad \hat{\boldsymbol{\mathcal{G}}}^{\boldsymbol{k}}_{\boldsymbol{m}} = \boldsymbol{\mathcal{G}}(U^h_m(\boldsymbol{x}^k_m), \boldsymbol{\sigma}_m(\boldsymbol{x}^k_m)) \tag{7.17}$$

be the advective flux component and diffusive flux component, respectively, calculated at solution point $\boldsymbol{x}^k_m$ from the approximate solution U^h_m and the gradient approximation $\boldsymbol{\sigma}_m$. Now, using the

Table 7.1: FR update scheme for an element Ω_m, assuming the gradient approximation $\boldsymbol{\sigma}_m \approx \nabla U^h$ and the common interface gradients $\tilde{\boldsymbol{\sigma}}$ are known.

Task	**Details**
1: Interface Limits of U^h.	Using the polynomial expansions U^h_m (Eq. 7.2) and $\boldsymbol{\sigma}_m$ (Eq. 7.14), calculate U^h_m and $\boldsymbol{\sigma}_m$ at each flux point on the perimeter $\partial\Omega_m$.
2: Discontinuous Flux Calculation.	Based on the distribution of U^h_m and $\boldsymbol{\sigma}_m$, calculate $\boldsymbol{\mathcal{F}}(U^h_m)$ and $\boldsymbol{\mathcal{G}}(U^h_m, \sigma_m)$ at each solution point $\mathbf{x}^k_m$ on Ω_m and each flux point on $\partial\Omega_m$.
3: Common Flux Calculation.	Each flux point on each interface is shared by two elements. For each of these shared flux points, calculate a common advective flux component $\tilde{\boldsymbol{\mathcal{F}}}$ and a common diffusive flux component $\tilde{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}(\tilde{U}, \tilde{\boldsymbol{\sigma}})$; set $\tilde{\boldsymbol{\mathcal{Q}}} = \tilde{\boldsymbol{\mathcal{F}}} - \tilde{\boldsymbol{\mathcal{G}}}$. Each flux point is now associated with two calculated discontinuous flux values (one from each element) and a single common flux value.
4: Flux Divergence Calculation.	For each solution point $\mathbf{x}^k_m$, with $k \in \{1, 2, ..., K\}$, approximate the flux divergence as follows: $$(\nabla \cdot \boldsymbol{\mathcal{Q}})|_{\mathbf{x}^k_m} = \nabla^h \cdot \boldsymbol{\mathcal{Q}}^h_m + \sum_{f=1}^{N_V} \sum_{j=1}^{K} (\Delta Q)_{fj} \frac{1}{|J|_{\boldsymbol{\xi}^k}} \psi_{fj}(\boldsymbol{\xi}_k), \tag{7.15}$$ where $\nabla^h \cdot ()$ is the uncorrected divergence operator defined in Section 7.2.1.5. The individual components of Eq. (7.15) are further explained in Section 7.2.2.2, Section 7.2.2.3, and Section 7.2.2.4.
5: Update.	The divergence provides the time derivative of U^h at each solution point, which is used to update the field variables: $$\frac{d}{dt}\hat{U}^k_m = -(\nabla \cdot \boldsymbol{\mathcal{Q}})|_{\mathbf{x}^k_m} \tag{7.16}$$

solution space $\boldsymbol{\phi_m}$ in each element, the flux polynomial,

$$\boldsymbol{Q}^h(\boldsymbol{x} \in \Omega_m) = \boldsymbol{Q}^h_m(\boldsymbol{x}) = \sum_{k=1}^{K} \phi^k_m(\boldsymbol{x}) \hat{\boldsymbol{Q}}^{\boldsymbol{k}}_{\boldsymbol{m}} \quad \text{where} \quad \hat{\boldsymbol{Q}}^{\boldsymbol{k}}_{\boldsymbol{m}} = \hat{\boldsymbol{\mathcal{F}}}^{\boldsymbol{k}}_{\boldsymbol{m}} - \hat{\boldsymbol{\mathcal{G}}}^{\boldsymbol{k}}_{\boldsymbol{m}} \quad \forall k \in \{1, 2, ..., K\}, \tag{7.18}$$

allows for the calculation of $\boldsymbol{Q}$ at any point in a given element. Due to the lack of continuity constraints at element-element interfaces, this polynomial is known as the *discontinuous* flux polynomial of Ω_m. Given the nodal DOFs $\hat{\mathbf{U}}$ at time t, the discontinuous flux polynomial is defined by calculating the advective and diffusive flux components as shown in Eq. (7.17) at each solution point in each element. This calculation of nodal flux values is the first half of step 2 in Table 7.1. Then, the discontinuous flux polynomial is applied to calculate the discontinuous flux of Ω_m at each flux point on $\partial\Omega_m$.

7.2.2.3 Flux Difference

Step 4 in the FR update scheme (Table 7.1) contains a flux jump term with notation $(\Delta Q)_{fj}$. The indices correspond to a particular flux point; the f index is the face of the element that the flux point is on, and the j index is the local index of that flux point on face f. For example, flux point $\boldsymbol{\xi}^{\boldsymbol{F}}_{\mathbf{13}}$ would be the third flux point on the first face of the element (see Figure 7.2). Consider an element Ω_m; the flux jump at each flux point of Ω_m is

$$(\Delta Q)_{fj} = (\tilde{\boldsymbol{Q}} - \boldsymbol{Q}^h_m)|_{fj} \cdot \begin{bmatrix} n_\xi y_{,\eta} - n_\eta y_{,\xi} \\ n_\eta x_{,\xi} - n_\xi x_{,\eta} \end{bmatrix}|_{fj}. \tag{7.19}$$

The term $\tilde{\boldsymbol{Q}}$ is the "common interface flux;" it is calculated during Step 3 of Table 7.1. The flux term $\boldsymbol{Q}^h_m$ is evaluated via the discontinuous flux polynomial of Ω_m. The terms n_ξ and n_η are the ξ and η components, respectively, of the outward normal from Ω_{ref} along face f. Recall that the flux itself, and therefore the difference $\tilde{\boldsymbol{Q}} - \boldsymbol{Q}^h_m$, has two spatial components per field variable, so it is appropriate to dot it against a two-row column vector. The metric terms must be evaluated at flux point $\boldsymbol{\xi}^{\boldsymbol{F}}_{\boldsymbol{fj}}$ on $\partial\Omega_{ref}$.

7.2.2.4 Correction Field

The function $\psi_{fj}(\boldsymbol{\xi})$ is the "correction field" associated with flux point $\boldsymbol{\xi}^{\boldsymbol{F}}_{\boldsymbol{fj}}$, which is the j^{th} flux point on face f of the reference element (see Figure 7.2). Note that the usage of ψ has nothing to do with the recovery basis of Chapter 2. For the case of triangular elements, these fields are defined according to Equation 5.49 in Castonguay et al. [20]. Their approach is summarized here. Let $\boldsymbol{L}$ be a basis of K orthogonal polynomials on the reference triangle, such that

$$\int_{\Omega_{ref}} L_i(\xi,\eta)L_j(\xi,\eta)dA = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j. \end{cases} \tag{7.20}$$

For arbitrary degree p, this set of polynomials is identified using the modified Gram-Schmidt process of Bassi et al. [7]. Each of the $N_v \times (p+1)$ correction fields are constructed via linear combination in the orthogonal basis:

$$\psi_{fj}(\boldsymbol{\xi}) = \sum_{k=1}^{K} \hat{\psi}^k_{fj} L_k(\boldsymbol{\xi}). \tag{7.21}$$

The coefficients $\hat{\boldsymbol{\psi}}$ (there are a total of $N_v \times (p+1) \times K$) are set according to Equation 5.49 in Castonguay et al. [20]. For a given ψ_{fj}, the set of K coefficients $\hat{\boldsymbol{\psi}}_{\boldsymbol{fj}}$ is set as follows:

$$c \sum_{k=1}^{K} \hat{\psi}^k_{fj} \sum_{m=1}^{p+1} \binom{p}{m-1} (D^{(m,p)}L_i)(D^{(m,p)}L_k) = -\hat{\psi}^i_{fj} + \int_{\partial\Omega_{ref}} (\boldsymbol{h}_{fj} \cdot \hat{\boldsymbol{n}}) L_i ds \quad \forall i \in \{1, ..., K\}, \tag{7.22}$$

where the term $\boldsymbol{h}_{fj} \cdot \hat{\boldsymbol{n}}$ is a scalar-valued degree p polynomial along edge f; this polynomial is defined only along the surface of Ω_{ref} and is therefore a 1D polynomial. According to Equation 4.24 of Castonguay et al. [20], the $\boldsymbol{h}_{fj} \cdot \hat{\boldsymbol{n}}$ polynomial is constrained as follows:

$$(\boldsymbol{h}_{fj} \cdot \hat{\boldsymbol{n}})(\boldsymbol{\xi}^{\boldsymbol{F}}_{\boldsymbol{mn}}) = \begin{cases} 1 & \text{if} \quad f = m \quad \& \quad j = n \\ 0 & \text{otherwise.} \end{cases} \tag{7.23}$$

The operator D appearing in Eq. (7.22) is defined by Castonguay et al. [20], Equation 5.1:

$$D^{(m,p)}L_i = \frac{\partial^p L_i}{\partial r^{(p-m+1)} \partial s^{(m-1)}} \tag{7.24}$$

where r is a face-tangential reference coordinate along face f of the reference element and s is the coordinate orthogonal to r; one may consult Castonguay et al. [20] for additional clarification. Note the presence of the scalar parameter c in Eq. (7.22); this number should be nonnegative for numerical stability [20] and can be varied to influence the behavior of the FR method.

In contrast, for the case of quadrilateral elements, each ψ_{fj} is the product of one polynomial in ξ and another polynomial in η. To imitate the work of Huynh [46, 47], the values of each ψ_{fj} are determined by the derivatives of the 1D correction polynomials g. Given a choice of 1D correction polynomial (recall that the choice of g is one of the design options in an FR scheme), the correction field ψ_{fj} is defined as follows:

$$\psi_{fj}(\xi,\eta) = \begin{cases} -\frac{\partial g(\eta)}{\partial \eta} \times \ell_j^{1D}(\xi) & \text{if} \quad f = 1 \\ -\frac{\partial g(-\xi)}{\partial (-\xi)} \times \ell_j^{1D}(\eta) & \text{if} \quad f = 2 \\ -\frac{\partial g(-\eta)}{\partial (-\eta)} \times \ell_j^{1D}(-\xi) & \text{if} \quad f = 3 \\ -\frac{\partial g(\xi)}{\partial \xi} \times \ell_j^{1D}(-\eta) & \text{if} \quad f = 4, \end{cases} \tag{7.25}$$

where ℓ_j^{1D} is the 1D, degree-p Lagrange polynomial that is unity at flux point j on the face and zero at all other flux points. The effect of ℓ_j^{1D} is that each correction field ψ_{fj} is zero along a set of p lines that run either vertically or horizontally across the reference element.

7.2.3 Gradient Approximations

In addition to the discontinuous polynomial approximation U^h, we build a set of gradient approximations inside each element Ω_m using the solution basis $\boldsymbol{\phi_m}$. One of these is the auxiliary polynomial vector $\boldsymbol{\sigma_m}$, previously described in Eq. (7.14). We also consider the "broken gradient"

polynomial, denoted $\mathbf{v}_m$ in each Ω_m:

$$\mathbf{v}_m(\boldsymbol{x}) = \sum_{k=1}^{K} \hat{\mathbf{v}}_m^k \phi_m^k(\boldsymbol{x}) \quad \text{where} \quad \hat{\mathbf{v}}_m^k = \nabla^h U_m^h|_{\boldsymbol{x}=\boldsymbol{x}_m^k}. \tag{7.26}$$

With the broken gradient polynomial defined, the coefficients $\hat{\boldsymbol{\sigma}}_{\boldsymbol{m}}^{\boldsymbol{k}}$ for each solution point $\mathbf{x}_m^k$ in each element Ω_m in the auxiliary polynomial expansion are defined as:

$$\hat{\boldsymbol{\sigma}}_{\boldsymbol{m}}^{\boldsymbol{k}} = \hat{\mathbf{v}}_m^k + \sum_{f=1}^{N_V} \sum_{j=1}^{K} (\Delta U)_{fj} \frac{1}{|J|_{\boldsymbol{\xi}^k}} \psi_{fj}(\boldsymbol{\xi}_k), \tag{7.27}$$

which is similar to the form (Eq. 7.15) used to approximate the flux divergence from the flux distribution. The term $(\Delta U)|_{fj}$ at a point $\boldsymbol{x}_{\boldsymbol{fj}}^{\boldsymbol{F}} = \boldsymbol{x}(\boldsymbol{\xi}_{\boldsymbol{fj}}^{\boldsymbol{F}})$ on $\partial\Omega_m$ is a vector quantity defined as follows:

$$(\Delta U)_{fj} = \begin{bmatrix} (\tilde{U} - U_m^h)(n_\xi y_{,\eta} - n_\eta y_{,\xi}) \\ (\tilde{U} - U_m^h)(n_\eta x_{,\xi} - n_\xi x_{,\eta}) \end{bmatrix}|_{fj}, \tag{7.28}$$

where $\tilde{U}$ is the "common interface solution," defined in Section 7.2.3.1. Through the correction field and the jumps ΔU along $\partial\Omega_m$, the auxiliary polynomial (Eq. 7.14) draws in information from neighboring elements (via the coefficient definition in Eq. 7.27) and is expected to yield a more accurate gradient approximation than $\mathbf{v}$.

Similar to the DG method, careless usage of the auxiliary polynomial leads to a non-compact stencil, which is undesirable. As with DG, we introduce the "semi-connected" gradient polynomial. Each element posesses N_V semi-connected gradient polynomials (SMGP), and each SMGP is associated with a particular face of a particular element. The SMGP for a given face f of a given element Ω_m, denoted $\boldsymbol{\gamma}_{\boldsymbol{mf}}(\boldsymbol{x})$, is crafted as follows:

$$\boldsymbol{\gamma}_{\boldsymbol{mf}}(\boldsymbol{x}) = \sum_{k=1}^{K} \hat{\boldsymbol{\gamma}}_{mf}^k \phi_m^k(\boldsymbol{x}) \quad \text{where} \quad \hat{\boldsymbol{\gamma}}_{mf}^k = \hat{\mathbf{v}}_m^k + \sum_{j=1}^{K} (\Delta U)_{fj} \frac{\chi}{|J|_{\boldsymbol{\xi}^k}} \psi_{fj}(\boldsymbol{\xi}_k). \tag{7.29}$$

Note that the SMGP for face f uses information only from face f, not any of the other faces of Ω_m. The scalar quantity χ is a stabilization parameter that is frequently set to $\chi = 1$ but can be

taken as $\chi = 2$ instead to improve scheme stability; it is the same jump parameter that appeared in Chapter 3 when describing the compact diffusion schemes.

7.2.3.1 Common Interface Solution

Both the auxiliary polynomial and the SMGPs of each element depend on a quantity $\tilde{U}$, known as the "common interface solution." For a given flux point on an interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$, $\tilde{U}$ is an estimate for the true solution along the interface; the estimate makes use of U_A^h and U_B^h. In this work, three different strategies have been applied to calculate the common interface solution.

- The simple option is to take $\tilde{U}|_{\mathbf{x_f}} = \frac{1}{2}(U_A^h + U_B^h)|_{\mathbf{x_f}}$, where $\mathbf{x_f}$ is the location of the flux point. This strategy was originally referred to by Huynh as the "I-centered" approach (see Schemes 1 through 10 in his original work [47]), and we maintain that naming convention here.

- The second option is a special approach described by Huynh and referred to as the "I-continuous" scheme [47]. In this approach, the common interface solution is set in such a manner that the SMGPs associated with the face (one for each element) have the same interface-normal derivative component.

- The third option is to use the recovered solution at the interface, thoroughly described in Chapter 2. This approach is called the "recovery-assisted" FR method, abbreviated REAFR, and it is new.

The FR equivalent of the oldest Recovery-based DG approach, referred to as RDG-2x [68], has previously been studied in the FR framework [47] and is not the same as the REAFR scheme. In the RDG-2x approach, as with the REAFR scheme, the recovered solution is used to calculate the interface solution $\tilde{U}$. The difference between the two schemes is in the calculation of the common interface gradient, described in Section 7.2.3.2.

Once the common interface solution $\tilde{U}$ has been calculated from available data along all interfaces (using one of the approaches just described), Eq. (7.27) is solved to yield the auxiliary

polynomial, which allows calculation of the diffusive portion of the discontinuous flux polynomial.

7.2.3.2 Common Interface Gradient

Calculation of the viscous portion of the common flux $\tilde{\boldsymbol{Q}}$ at each interface flux point requires a common gradient $\tilde{\boldsymbol{\sigma}}$ to be calculated; once the common gradient is known, it is applied to calculate the common viscous flux component $\tilde{\boldsymbol{G}}$.

This study includes two approaches available for calculating $\tilde{\boldsymbol{\sigma}}$. The first option is to take the average of the two SMGPs associated with the face: $\tilde{\boldsymbol{\sigma}}|_{\mathbf{x_f}} = \frac{1}{2}(\boldsymbol{\gamma}_A + \boldsymbol{\gamma}_B)|_{\mathbf{x_f}}$, where $\mathbf{x_f}$ is the location of the flux point and $\boldsymbol{\gamma}_A$ and $\boldsymbol{\gamma}_B$ are the face-associated SMGPs of Ω_A and Ω_B, respectively. This averaged gradient is used for the I-centered scheme, the I-continuous scheme, and the REAFR scheme. Thus, the REAFR scheme is actually the FR equivalent of the HAG scheme introduced in Chapter 3. Similary, the I-centered scheme is the FR equivalent of the BR2 scheme. However, to our knowledge, there is no DG equivalent to Huynh's I-continuous scheme, and the formation of such a DG scheme represents an exciting avenue for future research.

A second option for calculating $\tilde{\boldsymbol{\sigma}}$ is to use the gradient of the interface's recovered solution, as described by Lo & van Leer [67, 68] and applied previously by Huynh [47]. This scheme, known as RDG-2x, provides exceptional performance in the 1D case, but the analysis of Section 7.3 shows that it becomes unstable for 2D problems. Note that the interface gradient is what separates the new REAFR scheme and the RDG-2x scheme; where the REAFR scheme takes the average of the SMGPs as the interface gradient, the RDG-2x scheme differentiates the recovered polynomial.

7.2.3.3 Calculation and Usage of Gradients in Update Scheme

The update procedure given in Table 7.1 assumes the gradient approximations $\tilde{\boldsymbol{\sigma}}_{\mathcal{I}}$ and $\boldsymbol{\sigma}_m$ for each interface $\mathcal{I}$ and element Ω_m to be known. We now fill in some remaining details.

Step 1 in Table 7.1 requires $\boldsymbol{\sigma_m}$ to be known for each Ω_m. Thus, we perform the following two substeps before executing Step 1 in the main update scheme (Table 7.1):

(a): Calculate $\tilde{U}$ for each flux point on each interface.

(b): Use Eq. (7.27) to evaluate auxiliary polynomial coefficients $\hat{\sigma}_m^k$ for all solution points of each element.

In Table 7.1, Step 3 requires a common gradient at each interface flux point to facilitate calculation of $\tilde{\mathcal{G}}$. To address this need, we perform the following substeps between Step 2 and Step 3:

(a): Calculate $\boldsymbol{\gamma_A}$ and $\boldsymbol{\gamma_B}$ at each interface flux point, where Ω_A and Ω_B are the intersecting elements.

(b): Given $\boldsymbol{\gamma_A}$ and $\boldsymbol{\gamma_B}$, calculate the common interface gradient.

Once the common interface gradient is known, one may proceed with Step 3 in Table 7.1.

7.2.4 Implementation Summary

With the calculation and usage of gradient terms now defined, the FR scheme for advection-diffusion problems on 2D elements (either triangles or quadrilaterals) is nearly complete. To complete the description, we would also need to discuss the interface advective flux, $\tilde{\mathcal{F}}$. Typically, this flux is taken as the upwind flux based on the solution states at the interface flux point, which can become a nontrivial matter depending on the problem setup. However, the focus of this work is scalar diffusive phenomena, where $\mathcal{F} = 0$, so no more discussion of advective fluxes is necessary. With the desription of the FR method complete, we move on to Fourier analysis.

7.3 Fourier Analysis

Fourier analysis was performed on a vast collection of possible FR configurations to determine accuracy and stability properties. The Fourier analysis technique employed is very similar to the approach applied on simplex elements in Section 3.8 of Chapter 3. The difference between that section and this section is that where the analysis of CGR schemes involved two elements per mesh block, the analysis here employs four elements per mesh block. We begin by describing the geometry setup. Next, the procedure for performing analysis is summarized. Then, our findings are reported.

7.3.1 Mesh Setup

This analysis features two mesh setups. The first is a Cartesian mesh and the second is a nonuniform mesh of triangular (simplex) elements, obtained by perturbing a uniform simplex mesh. Regardless of the mesh type (Cartesian or simplex), the mesh consists of 20 elements, distributed in 5 blocks with 4 elements in each block. The arrangement of the five blocks (designated **BL1**, **BL2**, etc.) is illustrated in Figure 7.5. The middle block (**BL5**) of elements has its lower-left corner at $(x, y) = (2, 2)$, and each of the remaining four blocks (**BL1** through **BL4**) shares a side with the center block.

For a given mesh configuration, each of the five blocks in the mesh use the same arrangement of elements. These element arrangements are shown in Figure 7.6 and Figure 7.3. In the Cartesian case, each element is a square of side length $h = \frac{\Delta x}{2}$. In the simplex case, the single intersection point of the four elements sharing the block is perturbed by $(\delta x, \delta y) = (\frac{h}{2}, -\frac{h}{5})$ from the block centroid, but each of the five blocks remains square.

Each element is assigned a two-digit index. The first digit corresponds to the block (**BL1** to **BL5**) that the element resides in. The second index corresponds to the element's designation in the repeating 4-element pattern. For example, element 23 (denoted Ω_{23}) is the 3^{rd} element in **BL2**. The elements are numbered counterclockwise in each block. This element indexing must be accounted

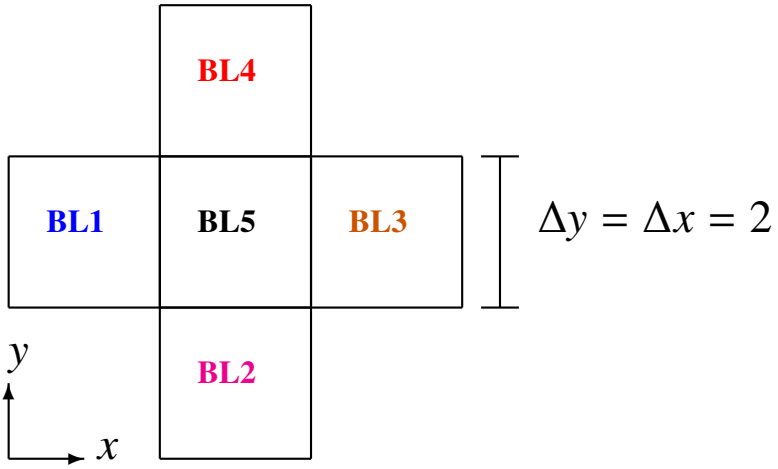
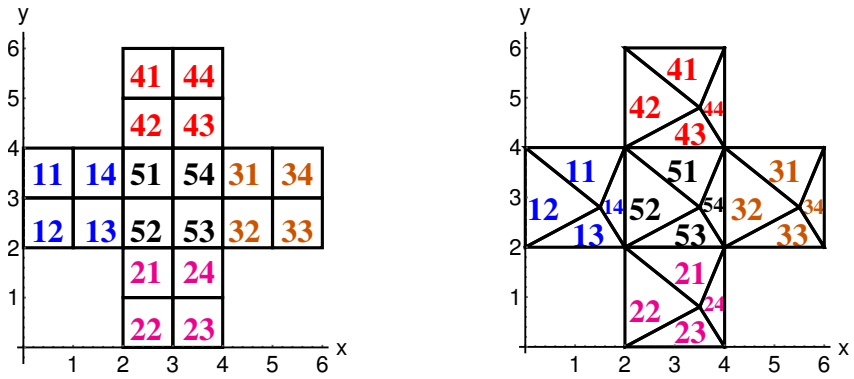


Figure 7.5: The five blocks in the Fourier analysis mesh; each block is a square of side length 2, and each block is partitioned into four elements as shown in Figure 7.6.



(a) Cartesian Grid.

(b) Perturbed Triangles.

Figure 7.6: The grids used for Fourier analysis. The integer inside each element is the element index; the first digit corresponds to the mesh block and the second digit to the element's designation within that block.

for in the formation of the update matrix for Fourier analysis.

7.3.2 Problem Setup

The governing differential equation is taken to be the 2D shear diffusion equation (Eq. 2.7) with unit diffusivity, where the parameter θ is set to zero for Laplacian diffusion or $\theta = \frac{1}{2}$ for the shear

diffusion case. Let the initial condition be

$$U_{IC}(x, y) = \exp[i(\omega_x x + \omega_y y)], \tag{7.30}$$

where ω_x and ω_y are the solution wavenumbers in the x and y directions, respectively, and $i^2 = -1$. The corresponding exact solution, assuming spatial periodicity, is

$$U(x, y, t) = \exp[-t(\omega_x^2 + 2\theta\omega_x\omega_y + \omega_y^2)]U_{IC}(x, y). \tag{7.31}$$

For the purpose of analysis, the wavenumber pair is characterized by wavenumber angle α and magnitude $|\omega|$, where $\omega_x = |\omega|\cos(\alpha)$ and $\omega_y = |\omega|\sin(\alpha)$.

7.3.3 Parameter Space

Fourier analysis has been performed over a broad set of FR discretizations. There are four interface quantity ($\tilde{U}$, $\tilde{\boldsymbol{\sigma}}$) approaches, each of which can make use of any correction field for the flux divergence and either the same or different correction field for the $\boldsymbol{\gamma}$ calculation. These schemes are the I-centered approach, the I-continuous approach, the recovery-assisted approach (abbreviated REAFR), and the RDG-2x approach, all of which are described in Section 7.2.3. For the first three approaches, regardless of how $\tilde{U}$ is calculated, the common interface gradient is $\tilde{\boldsymbol{\sigma}} = Avg(\boldsymbol{\gamma_L}, \boldsymbol{\gamma_R})$. The RDG-2x scheme instead uses the gradient of the recovered solution for $\tilde{\boldsymbol{\sigma}}$; consequently, the behavior of the RDG-2x scheme is unaffected by the χ parameter and the choice of correction field for the $\boldsymbol{\gamma}$ correction.

In the Cartesian case, for each ($\tilde{U}$, $\tilde{\boldsymbol{\sigma}}$) approach, twelve possible correction field combinations are inspected: three choices for the flux divergence versus four choices for the gradient correction. The correction field is characterized by the 1D correction polynomial that is differentiated (see Section 7.2.2.4) to form the correction field. For the flux divergence, the three choices are g_{DG}, g_{SD}, and g_{Hu}. For the $\boldsymbol{\gamma}$ calculation, the four choices are g_{Le}, g_{DG}, g_{SD}, and g_{Hu}.

In the perturbed simplex case, the correction field (for either flux divergence or $\boldsymbol{\gamma}$ calculation)

is defined by a scalar parameter c as mentioned in Section 7.2.2.4. In the $p = 1$ case, the values are $\{c_1, c_2, c_3\} = \{0,\ 1/3,\ 4/3\}$. In the $p = 2$ case, the values are $\{c_1, c_2, c_3\} = \{0,\ 4/135,\ 1/15\}$. In the $p = 3$ case, the values are $\{c_1, c_2, c_3\} = \{0,\ 1/1050,\ 8/4725\}$. These c values are selected as extensions of the g_{DG}, g_{SD}, and g_{Hu} polynomials to the simplex element and are taken from the description of the ESFR method in 1D by Vincent et al. [107]. The correction field for the flux divergence is not required to be the same correction field as that of the $\boldsymbol{\gamma}$ calculation, so there are nine possible combinations for the pair of correction fields.

In addition to the interface solution strategy and the choice of correction fields, another parameter that affects the FR method is the choice of χ in Eq. (7.29). Analysis has been performed with both $\chi = 1$ and $\chi = 2$ based on our experience with the DG method.

The solution order p, interface solution strategy, correction field pair, and χ parameter define an FR discretization; however, a single discretization's properties can depend on the initial condition and the form of the governing differential equation. With regard to the governing differential equation (Eq. 2.7), analysis has been performed for both the $\theta = 0$ and $\theta = \frac{1}{2}$ cases; the first case is the scalar Laplacian case and the second is the shear diffusion case. The initial condition (Eq. 7.30) is characterized by the wavenumber magnitude $|\omega|$ and the wavenumber angle $\alpha = \arctan(\omega_y/\omega_x)$, with $\omega_x = |\omega|\cos(\alpha)$ and $\omega_y = |\omega|\sin(\alpha)$. When performing Fourier analysis, the wavenumber magnitude $|\omega|$ is discretized, and a broad range of $|\omega|$ values is swept over to examine the scheme's properties. Thus, $|\omega|$ itself is not considered an element of the parameter space, but the wavenumber angle is. This analysis is thus far more thorough than the technique presented in Chapter 3.

The seven-dimensional parameter space consists of the solution order p, the interface solution strategy, the correction field used for the flux divergence, the correction field used for the $\boldsymbol{\gamma}$ calculation, the χ parameter, the shear factor θ, and the wavenumber angle α. The resulting update matrix $\mathcal{A}$ (whose construction is described in Section 7.3.4) is a function of $|\omega|$ and is set once the listed parameters have been specified. The complete list of parameter choices explored in the analysis is given in Table 7.2 and Table 7.3. The set of α values merits explanation: in addition to testing for stability over an evenly spaced partition of $\alpha \in [0, \pi]$, we extract the order of accuracy for the

$\alpha = \frac{\pi}{4}$ and $\alpha = \frac{\pi}{7}$ cases. The sweep over a range of α values is necessary to ensure stability for a wide range of initial conditions, while the order of accuracy analysis is conducted at two separate wavenumber angles to explore whether or not a scheme's order of accuracy can be affected by α.

7.3.4 Analysis Technique

Given a set of scheme and problem parameters for the FR method, the divergence of the flux at each of the solution points in the four **BL5** elements is expressed in terms of all of the DOFs in the surrounding stencil. For example, the update scheme for element Ω_{51} in the Cartesian case takes the form:

$$\frac{d}{dt}\hat{\mathbf{U}}_{\mathbf{51}} = \mathcal{D}^{\mathbf{51}}_{\mathbf{14}}\hat{\mathbf{U}}_{\mathbf{14}} + \mathcal{D}^{\mathbf{51}}_{\mathbf{52}}\hat{\mathbf{U}}_{\mathbf{52}} + \mathcal{D}^{\mathbf{51}}_{\mathbf{54}}\hat{\mathbf{U}}_{\mathbf{54}} + \mathcal{D}^{\mathbf{51}}_{\mathbf{42}}\hat{\mathbf{U}}_{\mathbf{42}}, \tag{7.32}$$

Table 7.2: Parameter Space for Fourier Analysis, Cartesian Mesh.

Parameter	**Tested Values**
$\boldsymbol{p}$:	$\{1, 2, 3\}$
Interface Solution Strategy	{I-centered , I-continuous , REAFR , RDG-2x}
Correction Field: $\nabla \cdot \boldsymbol{Q}$	$\{g_{DG}\ ,\ g_{SD}\ ,\ g_{Hu}\ \}$
Correction Field: $\boldsymbol{\gamma}$	$\{g_{Le},\ g_{DG}\ ,\ g_{SD}\ ,\ g_{Hu}\ \}$
$\boldsymbol{\chi}$	$\{1\ ,\ 2\}$
$\boldsymbol{\theta}$	$\{0\ ,\ \frac{1}{2}\}$
$\boldsymbol{\alpha}$	$\{0, \frac{\pi}{10}, \frac{2\pi}{10}, \frac{3\pi}{10}, ..., \pi\} \cup \{\frac{\pi}{4}, \frac{\pi}{7}\}$

Table 7.3: Parameter Space for Fourier Analysis, Simplex Mesh.

Parameter	**Tested Values**
$\boldsymbol{p}$:	$\{1, 2, 3\}$
Interface Solution Strategy	{I-centered , I-continuous , REAFR , RDG-2x}
Correction Field: $\nabla \cdot \boldsymbol{Q}$	$\{c_1\ ,\ c_2\ ,\ c_3\ \}$
Correction Field: $\boldsymbol{\gamma}$	$\{c_1\ ,\ c_2\ ,\ c_3\ \}$
$\boldsymbol{\chi}$	$\{1\ ,\ 2\}$
$\boldsymbol{\theta}$	$\{0\ ,\ \frac{1}{2}\}$
$\boldsymbol{\alpha}$	$\{0, \frac{\pi}{10}, \frac{2\pi}{10}, \frac{3\pi}{10}, ..., \pi\} \cup \{\frac{\pi}{4}, \frac{\pi}{7}\}$

where $\mathcal{D}_{\boldsymbol{B}}^{\boldsymbol{A}}$ is the matrix containing the contribution made by the DOFs of Ω_B to the flux divergence calculation of Ω_A. This procedure is time-consuming but straightforward; a symbolic math toolbox is used to form the update matrix for this study, but a typical programming language can also be used. For a given block of elements, define the "DOF block" as follows:

$$\hat{\mathbf{U}}_{\mathbf{BLb}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{b1}} \\ \hat{\mathbf{U}}_{\mathbf{b2}} \\ \hat{\mathbf{U}}_{\mathbf{b3}} \\ \hat{\mathbf{U}}_{\mathbf{b4}} \end{bmatrix}. \tag{7.33}$$

Based on the assumed form of the initial condition (Eq. 7.30), each DOF block is rewritten in terms of the DOF block of **BL5**:

$$\begin{aligned} \hat{\mathbf{U}}_{\mathbf{BL1}} &= \exp[-i\omega_x \Delta x]\hat{\mathbf{U}}_{\mathbf{BL5}} \quad , \quad \hat{\mathbf{U}}_{\mathbf{BL3}} = \exp[i\omega_x \Delta x]\hat{\mathbf{U}}_{\mathbf{BL5}} \, , \\ \hat{\mathbf{U}}_{\mathbf{BL2}} &= \exp[-i\omega_y \Delta y]\hat{\mathbf{U}}_{\mathbf{BL5}} \quad , \quad \hat{\mathbf{U}}_{\mathbf{BL4}} = \exp[i\omega_y \Delta y]\hat{\mathbf{U}}_{\mathbf{BL5}} \, . \end{aligned} \tag{7.34}$$

Let $\mathcal{D}_{\boldsymbol{BL\alpha}}$ be the matrix containing the contributions of all DOFs in block **BL**$\boldsymbol{\alpha}$ to the flux divergence calculations at all solution points in block **BL5**. For example, in the Cartesian case, since Ω_{51} borders Ω_{14} and Ω_{52} borders Ω_{13}, the matrix $\mathcal{D}_{\boldsymbol{BL1}}$ is a $4K \times 4K$ block matrix with nonzero entries copied in from $\mathcal{D}_{\mathbf{14}}^{\mathbf{51}}$ and $\mathcal{D}_{\mathbf{13}}^{\mathbf{52}}$:

$$\textbf{Cartesian:} \qquad \mathcal{D}_{\boldsymbol{BL1}} = \begin{bmatrix} \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathcal{D}_{\mathbf{14}}^{\mathbf{51}} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathcal{D}_{\mathbf{13}}^{\mathbf{52}} & \mathbf{0}_{K\times K} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} \end{bmatrix}. \tag{7.35}$$

In contrast, for the perturbed simplex mesh, the only interface linking blocks **BL1** and **BL5** is $\partial\Omega_{52} \cap \partial\Omega_{14}$. The corresponding block update matrix is

$$\textbf{Simplex:} \qquad \mathcal{D}_{\boldsymbol{BL1}} = \begin{bmatrix} \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathcal{D}_{\mathbf{14}}^{\mathbf{52}} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} \end{bmatrix}. \tag{7.36}$$

Making use Eq. (7.34) and the block update matrices, the semi-discrete update scheme is written:

$$\begin{aligned} &\frac{d}{dt}\hat{\hat{\mathbf{U}}}_{\mathbf{BL5}} = \\ &(\exp[-i\omega_x\Delta x]\mathcal{D}_{\boldsymbol{BL1}} + \exp[-i\omega_y\Delta y]\mathcal{D}_{\boldsymbol{BL2}} + \exp[i\omega_x\Delta x]\mathcal{D}_{\boldsymbol{BL3}} + \exp[i\omega_y\Delta y]\mathcal{D}_{\boldsymbol{BL4}} + \mathcal{D}_{\boldsymbol{BL5}})\hat{\hat{\mathbf{U}}}_{\mathbf{BL5}}. \end{aligned} \tag{7.37}$$

The FR update scheme is consequently reduced to a system of $4K$ ordinary differential equations:

$$\frac{d}{dt}\hat{\hat{\mathbf{U}}}_{\mathbf{BL5}} = \mathcal{A}\hat{\hat{\mathbf{U}}}_{\mathbf{BL5}}, \tag{7.38}$$

where

$$\begin{aligned} &\mathcal{A} = \mathcal{A}(\omega_x, \omega_y) = \\ &\exp[-i\omega_x\Delta x]\mathcal{D}_{\boldsymbol{BL1}} + \exp[-i\omega_y\Delta y]\mathcal{D}_{\boldsymbol{BL2}} + \exp[i\omega_x\Delta x]\mathcal{D}_{\boldsymbol{BL3}} + \exp[i\omega_y\Delta y]\mathcal{D}_{\boldsymbol{BL4}} + \mathcal{D}_{\boldsymbol{BL5}}. \end{aligned} \tag{7.39}$$

Similar to our analysis of DG schemes, Huynh's [47] approach is applied to the update matrix $\mathcal{A}$ to inspect the stability, spectral radius, and order of accuracy of the update scheme for the $4K$ DOFs contained in block **BL5**. In addition to depending on the wavenumber pair (ω_x, ω_y), the update matrix depends on the particular configuration of the FR method and the shear parameter θ in the governing differential equation (Eq. 2.7). Note that the technique of combining simplex elements for Fourier analysis has previously been applied by Khieu et al. [57] and Castonguay et al. [20];

however, to our knowledge, this study is the first to use the four-element block pattern to apply Fourier analysis to a nonuniform mesh.

With regard to implementation, for a specific solution order p, interface solution strategy, χ value, θ value, and correction field pair, the update matrix is assembled as a symbolic function of α and $|\omega|$. This assembly has been performed in the Mathematica software for this study, but could be performed in another symbolic math toolbox if desired. The wavenumber magnitude $|\omega|$ is discretized over the range $|\omega| \in [0, 2\pi(p+1)]$ with 50 evenly spaced points. Then, over each discrete $(\alpha, |\omega|)$ pair, the eigenvalues of $\mathcal{A}$ are calculated. The maximum eigenvalue magnitude encountered for any $(\alpha, |\omega|)$ pair is reported as the spectral radius. Additionally, we check for positivity of the eigenvalues; if the real component of any eigenvalue for any $(\alpha, |\omega|)$ is greater than zero, then the particular FR configuration is designated unstable.

To extract the order of accuracy, we follow the strategy of Huynh [46, 47]; for a particular α value, the eigenvalues $\mathcal{A}$ are calculated at a set of $|\omega|$ values. At each $|\omega|$, the scheme error is calculated as

$$E(\lambda) = |Re(\lambda^{con}) - Re(\lambda^{ex})|, \tag{7.40}$$

where $\lambda^{ex} = -(\omega_x^2 + 2\theta\omega_x\omega_y + \omega_y^2)$ corresponds to the exact solution of the governing differential equation (Eq. 7.31) and λ^{con} is the eigenvalue of $\mathcal{A}$ that is closest to the exact eigenvalue. The rate at which $E(\lambda)$ converges to zero as $|\omega|$ approaches zero is $m+2$ where m is the order of accuracy. This relationship is used to identify the order of accuracy at a given α.

7.3.5 Findings

For a given FR configuration, the most important properties are the order of accuracy and whether or not the scheme is stable. The spectral radius (defined here as the maximum eigenvalue magnitude in the matrix $\mathcal{A}$ over the full sweep of α and $|\omega|$) is inversely proportional to the maximum allowable timestep size when explicit time integration is applied and is consequently a quantity of interest, but is less important than stability and order of accuracy. All three properties are reported in this section. The reporting of results is broken into two segments: one segment for reporting

and discussion of the Cartesian mesh results, then another for the perturbed simplex mesh.

Analysis has demonstrated that the RDG-2x technique is unstable when $\theta = \frac{1}{2}$ and p is greater than 2. Thus, while this method performs very well for scalar Laplacian diffusion, we have elected not to include it in this section; analysis results on the Cartesian mesh can be found in Appendix G. Future studies may reveal a way to stabilize this promising scheme in the shear diffusion case.

7.3.5.1 Cartesian Mesh

The spectral radii (over a range of α) and orders of accuracy (for $\alpha = \frac{\pi}{7}$) of the FR configurations on the Cartesian mesh are reported in Table 7.4, Table 7.5, and Table 7.6 for $p = 1$, $p = 2$, and $p = 3$, respectively. The chosen analysis technique required a great deal of compute time on our workstation, and the compute time grows significantly with p, so analysis was not performed for $p > 3$. Unstable schemes are marked with an **X**. The I-continuous scheme is slightly less robust than the other two, offering a smaller (but still ample) count of stable sub-configurations. As for the choice of correction functions, stability is guaranteed when using the g_{Le} polynomial for the $\boldsymbol{\gamma}$ correction field. The choice of the correction field for the flux divergence has little influence over stability, which is unsurprising because for the advection case [46], all of these correction field choices yield stable discretizations.

The reported spectral radii apply to the case where each element is of unit side length and would need to be scaled according to Δx^2 to be applied to a mesh of finer or coarser element width. With regard to spectral radius, the I-centered scheme achieves similar, but sometimes slightly larger, spectral radii than the I-continuous and REAFR schemes. The correction fields have a substantial effect on spectral radius; for both the flux divergence and $\boldsymbol{\gamma}$ corrections, the spectral radius is highest when using the g_{DG} or g_{Le} polynomial and smallest when using the g_{Hu} polynomial; this trend is in agreement with previous observations made by Huynh [46, 47]. The spectral radius grows as both χ and θ are increased; from a scheme configuration perspective, raising χ is usually disadvantageous because it raises the spectral radius, but it can have the beneficial effect of stabilizing a scheme that is unstable when $\chi = 1$ and $\theta = \frac{1}{2}$.

When stable, all schemes are at least order $2p$ accurate. Some schemes achieve order $2p + 2$ accuracy; in particular, the I-continuous and REAFR schemes achieve order $2p + 2$ accuracy when $\theta = 0$, the correction polynomial for the flux is g_{DG}, and the correction polynomial for the $\boldsymbol{\gamma}$ calculation is g_{Le}. For $\theta = \frac{1}{2}$, the REAFR approach has the most sub-configurations that achieve order $2p + 2$ accuracy. Note the peculiar effect of the I-centered scheme with g_{DG} for the flux divergence correction and g_{Le} for the $\boldsymbol{\gamma}$ correction; the scheme is 2^{nd} order accurate when $p = 1$, but the order of accuracy jumps to 6 when $p = 2$. Huynh [47] reports similar behavior when analyzing this scheme in the 1D diffusion case, where it is labelled "Scheme 1: I-centered-g_{Le}/SP-g_{DG}."

The orders of accuracy of the various configurations have also been calculated for the $\alpha = \frac{\pi}{4}$ case and are reported in Appendix G. Typically, the order of accuracy observed in the $\alpha = \frac{\pi}{7}$ case is the same order of accuracy observed in the $\alpha = \frac{\pi}{4}$ case. A notable exception is the configuration using the I-centered approach with $p = 1$, $\theta = \frac{1}{2}$, and g_{SD} for the flux divergence correction; this scheme is 2^{nd} order accurate for the $\alpha = \frac{\pi}{7}$ case but 4^{th} order accurate for the $\alpha = \frac{\pi}{4}$ case.

7.3.5.2 Simplex Mesh

The spectral radii and orders of accuracy of the FR configurations on the perturbed simplex mesh are reported in Table 7.7, Table 7.8, and Table 7.9 for $p = 1$, $p = 2$, and $p = 3$, respectively. The REAFR and I-centered approaches are more robust than the I-continuous approach, showing a larger quantity of stable sub-configurations. However, even with the I-continuous approach being the least robust, it offers plenty of stable schemes when using $c = 0$ for the $\boldsymbol{\gamma}$ correction. Regardless of the interface solution strategy, the choice of $c = 0$ for the interface gradient ($\boldsymbol{\gamma}$) correction is more likely to yield stable schemes than other choices for the correction field. With regard to the correction for the flux divergence ($\nabla \cdot \boldsymbol{Q}$), the choice of correction field has little influence over whether or not a scheme is stable. The (θ, χ) pair has an effect on stability; a scheme is less likely to be stable in the $\theta = \frac{1}{2}$ case than in the $\theta = 0$ case. In contrast, increasing χ from 1 to 2 improves the likelihood of a scheme being stable.

As with the Cartesian analysis, the spectral radius grows with both χ and θ. The choice of the c parameter in the correction fields has a significant effect on the spectral radius; with all other scheme parameters being held constant, an increase in the c parameter (for either the flux divergence correction or the interface gradient correction) tends to decrease the spectral radius. This occurrence is consistent with previous observations of the FR method [113]. The interface

Table 7.4: Analysis on Cartesian Mesh, $p = 1$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes have an **X** listed for the spectral radius and order or accuracy.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{48, 27, 24, 24}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, **X**, **X**}	{51, 37, **X**, **X**}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{120, 72, 48, 27}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{120, 72, 51, 37}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{32, 20, 16, 17}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{36, 26, 23, **X**}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{80, 48, 32, 20}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{81, 49, 36, 26}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{24, 16, 12, 9}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{25, 18, 15, **X**}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{48, 32, 24, 16}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{49, 33, 25, 18}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 2, 2, 2}	{48, 24, 13, 8}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 2, **X**, **X**}	{49, 28, **X**, **X**}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 2, 2, 2}	{120, 73, 48, 25}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 2, 2, **X**}	{120, 73, 49, **X**}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{32, 17, 12, 8}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{33, 21, 18, **X**}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{81, 48, 32, 16}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, **X**}	{81, 48, 33, **X**}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{25, 16, 12, 8}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{25, 18, 14, **X**}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{48, 32, 24, 16}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, **X**}	{49, 33, 25, **X**}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 2}	{48, 24, 17, 16}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{49, 31, **X**, **X**}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{120, 72, 48, 24}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{120, 72, 49, 31}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{32, 17, 13, 11}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{33, 23, 20, **X**}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{80, 48, 32, 17}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{81, 48, 33, 23}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{24, 16, 12, 8}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}	{25, 18, 14, **X**}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{48, 32, 25, 16}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{49, 33, 25, 18}

solution strategy also has an effect on spectral radius. Overall, the REAFR scheme has the smallest spectral radii and the I-centered scheme has the largest spectral radii; the spectral radii of the I-continuous scheme are close to those of the REAFR scheme.

On the perturbed simplex mesh, all stable schemes are order $2p$ accurate. Overall, the error readings ($E(\lambda)$ in Eq. 7.40) from the I-centered scheme are larger than the error readings of the

Table 7.5: Analysis on Cartesian Mesh, $p = 2$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes denoted with an **X**.

p	Interface Sol. Strategy	Correction Field, $\nabla\cdot\boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}	{150, 121, 121, 121}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{186, 165, **X**, **X**}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 6}	{437, 293, 197, 150}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{437, 293, 213, 186}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{121, 84, 73, 72}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{140, 115, **X**, **X**}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{295, 199, 144, 121}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{295, 206, 159, 140}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{121, 84, 61, 52}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, **X**}	{128, 97, 79, **X**}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{264, 192, 144, 121}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{267, 196, 150, 128}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}	{149, 85, 61, 49}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, **X**, **X**, **X**}	{160, **X**, **X**, **X**}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{437, 293, 197, 149}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, **X**, **X**}	{437, 293, **X**, **X**}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{120, 85, 61, 49}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{134, 101, **X**, **X**}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{295, 199, 144, 120}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, **X**}	{295, 200, 154, **X**}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{121, 84, 61, 49}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{126, 92, **X**, **X**}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{264, 193, 145, 120}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, **X**}	{267, 196, 149, **X**}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}	{149, 84, 60, 53}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{157, 131, **X**, **X**}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 6}	{437, 293, 197, 149}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{437, 293, 197, 157}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{121, 85, 61, 49}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}	{133, 103, **X**, **X**}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{295, 199, 145, 120}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{295, 200, 155, 133}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{121, 85, 61, 49}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, **X**}	{126, 93, 72, **X**}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{264, 192, 145, 120}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{267, 196, 149, 126}

other two schemes, but since the order of accuracy is the same, this difference in the magnitude of error readings is not significant.

Table 7.6: Analysis on Cartesian Mesh, $p = 3$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes denoted with an **X**.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{374, 341, 341, 341}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{511, 470, **X**, **X**}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{1159, 841, 602, 523}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}	{1160, 841, 627, 575}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 197, 197}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{391, 322, **X**, **X**}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{841, 629, 485, 437}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}	{843, 650, 515, 472}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 178, 166}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{366, 285, **X**, **X**}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{821, 629, 485, 437}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}	{829, 640, 500, 455}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{8, 6, 6, 6}	{367, 245, 173, 149}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{8, **X**, **X**, **X**}	{440, **X**, **X**, **X**}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{8, 6, 6, 6}	{1159, 841, 602, 523}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{8, 6, **X**, **X**}	{1161, 843, **X**, **X**}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 173, 149}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{375, 289, **X**, **X**}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{841, 629, 485, 437}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, **X**, **X**}	{843, 646, **X**, **X**}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 173, 149}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{361, 271, **X**, **X**}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{821, 629, 485, 437}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, **X**, **X**}	{828, 639, **X**, **X**}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{8, 8, 8, 6}	{367, 245, 173, 149}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{8, **X**, **X**, **X**}	{431, **X**, **X**, **X**}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{8, 8, 8, 8}	{1159, 841, 602, 523}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{8, 8, 8, 8}	{1161, 843, 606, 527}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 173, 149}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{374, 290, **X**, **X**}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{841, 629, 485, 437}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}	{843, 646, 508, 462}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{341, 245, 173, 149}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}	{360, 272, **X**, **X**}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{821, 629, 485, 437}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}	{828, 639, 498, 452}

7.4 Chapter Conclusion

A new FR scheme for solving diffusion problems, namely the recovery-assisted approach (abbreviated REAFR), was presented. This scheme is built within the typical FR approach for diffusion problems but makes use of the Recovery concept to achieve more accurate estimates of the solu-

Table 7.7: Analysis on Perturbed Simplex Mesh, $p = 1$. Each row corresponds to three possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes have an **X** listed for the spectral radius and order or accuracy.

p	Interface Sol. Strategy	Correction Field, $\nabla\cdot\boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
1	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{64, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{64, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{132, 79, 52\}$
1	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{131, 79, 52\}$
1	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{43, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{43, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{88, 55, 38\}$
1	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{88, 55, 38\}$
1	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{38, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{37, \mathbf{X}, \mathbf{X}\}$
1	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{75, 48, 34\}$
1	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{75, 48, 34\}$
1	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{45, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{49, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{111, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{112, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{34, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{36, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{72, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{73, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{31, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{32, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{62, \mathbf{X}, \mathbf{X}\}$
1	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{62, \mathbf{X}, \mathbf{X}\}$
1	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, 2, \mathbf{X}\}$	$\{55, 35, \mathbf{X}\}$
1	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{55, \mathbf{X}, \mathbf{X}\}$
1	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{109, 66, 44\}$
1	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{110, 66, 44\}$
1	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, 2, \mathbf{X}\}$	$\{34, 22, \mathbf{X}\}$
1	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{35, \mathbf{X}, \mathbf{X}\}$
1	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{67, 42, 29\}$
1	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{68, 44, 31\}$
1	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{2, 2, \mathbf{X}\}$	$\{29, 19, \mathbf{X}\}$
1	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{2, \mathbf{X}, \mathbf{X}\}$	$\{30, \mathbf{X}, \mathbf{X}\}$
1	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{2, 2, 2\}$	$\{57, 37, 26\}$
1	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{2, 2, 2\}$	$\{58, 38, 27\}$

tion U along element-element interfaces. Fourier analysis demonstrated that when paired with the proper correction field, the new scheme provides superior performance compared to the commonly applied I-centered scheme. When it is stable, the I-continuous approach of Huynh (Schemes 11, 12, and 13 in [47]) also achieves better performance than the I-centered scheme.

Fourier analysis was performed not only on a Cartesian mesh, but also a perturbed 2D simplex

Table 7.8: Analysis on Perturbed Simplex Mesh, $p = 2$. Each row corresponds to three possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes have an $\mathbf{X}$ listed for the spectral radius and order or accuracy.

p	Interface Sol. Strategy	Correction Field, $\nabla\cdot\boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
2	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{202, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{201, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{451, 260, 221\}$
2	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{449, 259, 221\}$
2	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{148, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{147, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{321, 195, 170\}$
2	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{320, 195, 170\}$
2	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{142, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{141, \mathbf{X}, \mathbf{X}\}$
2	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{303, 186, 163\}$
2	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{301, 186, 162\}$
2	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{160, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{172, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{395, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{399, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{126, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{130, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{277, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{280, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{121, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{124, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{261, \mathbf{X}, \mathbf{X}\}$
2	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{263, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{169, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{178, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{373, 220, 190\}$
2	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{375, 223, 192\}$
2	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{120, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{123, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{242, 153, 135\}$
2	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{246, 156, 137\}$
2	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{115, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{4, \mathbf{X}, \mathbf{X}\}$	$\{117, \mathbf{X}, \mathbf{X}\}$
2	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{4, 4, 4\}$	$\{227, 144, 128\}$
2	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{4, 4, 4\}$	$\{230, 146, 130\}$

mesh. In both mesh setups, the I-continuous and recovery-assisted schemes allow larger timestep sizes than the I-centered scheme, though the difference is far more pronounced on the perturbed simplex mesh. On the Cartesian mesh, the I-continuous and recovery-assisted schemes can also allow greater orders of accuracy when paired with the proper correction fields; the use of g_{DG} is necessary to achieve order $2p + 2$ accuracy in the Cartesian case. The extension of the Fourier

Table 7.9: Analysis on Perturbed Simplex Mesh, $p = 3$. Each row corresponds to three possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes have an **X** listed for the spectral radius and order or accuracy.

p	Interface Sol. Strategy	Correction Field, $\nabla\cdot\boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{7}$	Spectral Radius
3	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{524, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{520, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{1167, 709, 658\}$
3	I-centered	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{1156, 704, 654\}$
3	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{402, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{397, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{890, 567, 531\}$
3	I-centered	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{884, 564, 528\}$
3	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{394, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{389, \mathbf{X}, \mathbf{X}\}$
3	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{867, 556, 521\}$
3	I-centered	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{862, 553, 518\}$
3	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{408, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{1036, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{1046, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{337, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{780, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{786, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{331, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$	$\{\mathbf{X}, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{759, \mathbf{X}, \mathbf{X}\}$
3	I-continuous	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{764, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{416, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{455, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{953, 595, 556\}$
3	REAFR	$c = 0$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{964, 605, 566\}$
3	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{320, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{332, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{667, 440, 416\}$
3	REAFR	$c = c_2$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{678, 448, 422\}$
3	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	0	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{313, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	1	$\frac{1}{2}$	$\{6, \mathbf{X}, \mathbf{X}\}$	$\{323, \mathbf{X}, \mathbf{X}\}$
3	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	0	$\{6, 6, 6\}$	$\{647, 428, 405\}$
3	REAFR	$c = c_3$	$\{c = 0, c = c_2, c = c_3\}$	2	$\frac{1}{2}$	$\{6, 6, 6\}$	$\{656, 435, 410\}$

analysis technique to nonuniform meshes is a significant aid to the scheme design process. Not only can a scheme's performance be thoroughly explored on a uniform mesh, as with the classical Fourier analysis approach, but the extension to non-uniform meshes can now be explored without full-scale implementation.

The excellent performance of the recovery-assisted FR scheme was not a surprise. The DG method can be interpreted as a special case of the FR method [29], so an approach that works well for the DG method should also work well in the Flux Reconstruction framework. The REAFR scheme itself is in fact the FR counterpart to the High-Accuracy-Gradient scheme (from Chapter 3) in the DG framework, which is able to achieve order $2p+2$ accuracy on Cartesian meshes for odd p. Keeping the link to DG in mind, it is reasonable to expect that the recovery-assisted advection schemes of Chapter 4 could yield attractive FR schemes for advection problems, and the analysis of such an approach offers an exciting path for future research in numerical methods.

CHAPTER 8

Conclusion

8.1 Summary

The DG method is a popular approach for the discretization of the compressible Navier-Stokes equations and possesses a handful of properties (specifically, the simple application of upwind-based dissipation, the block-diagonal global mass matrix, and the capability for arbitrarily high orders of accuracy on nontrivial geometries) that make it an attractive choice for high-fidelity simulation of turbulent flows. Additionally, it has the potential to become the future workhorse of industrial CFD applications if engineers begin to demand high-order accuracy in practical calculations. For both of these reasons, it is worthwhile to work towards improving the basic spatial discretization in the sense of accuracy versus cost. In this work, we explored ways to improve the accuracy of the basic discretization at a fixed solution order p, exploiting the accuracy of the recovery operator throughout. A collection of DG variants for advection problems, diffusion problems, and advection-diffusion problems was proposed. Analysis and testing focused on structured meshes, which are typical for fundamental flow physics research, but the resulting advection-diffusion schemes of Chapter 5 are applicable on unstructured quadrilateral meshes as well. Our scheme development process can be summarized by two specific strategies:

- Instead of working with the full compressible Navier-Stokes equations, the schemes were developed in the context of simple model equations. The linear advection equation was chosen as the simplified model of the Euler equations. The linear advection-diffusion equation

was chosen to model the compressible Navier-Stokes equations.

- Advection and diffusion processes were handled separately. One family of schemes (namely the Interface Gradient Recovery schemes of Chapter 3) was built specifically for diffusion problems; another family of schemes (namely the interface-centered binary reconstruction schemes of Chapter 4) is specific to advection problems. In Chapter 5, we paired our favorite compact member of the Interface Gradient Recovery family with the interface-centered binary reconstruction schemes to form three Recovery-assisted (RAD) schemes for advection-diffusion problems.

The overall philosophy in the scheme design process was to use the recovery operator in both its full-order and biased forms to estimate solution quantities along element-element interfaces.

Each proposed scheme, whether it be an advection scheme, a diffusion scheme, or an advection-diffusion scheme, was subjected to Fourier analysis to examine its worth. In addition to the typical 1D Fourier analysis approach, where only the principal eigenvalue is examined, we employed the technique of Watkins et al. [112] to examine the resolving efficiencies of advection and advection-diffusion schemes. In our exploration of diffusion schemes within both the DG framework and the related Flux Reconstruction (FR) framework, we extended the Fourier analysis technique to 2D problems on both Cartesian and non-Cartesian mesh geometries. Fourier analysis demonstrated that the RAD schemes for advection-diffusion provide a substantial accuracy advantage over the conventional DG method. A suite of test cases involving the linear advection equation, the linear advection-diffusion equation, and the compressible Navier-Stokes equations demonstrated that this accuracy advantage carries over to simulation of both linear and nonlinear PDE systems on structured meshes; limited success was achieved on unstructured meshes. Computational cost analyses indicated that the Recovery-assisted schemes increase the flop count compared to the conventional state-of-the-art approach; however, the cost penalty is small compared to the gain in accuracy.

8.2 Key Findings and Contributions

The key findings and contributions of each chapter are summarized independently.

8.2.1 Chapter 2: Fundamental Topics

The Recovery concept was originally developed with the idea of enforcing weak equivalence between the discontinuous DG polynomials (U^h) and the smooth recovered solution over a given two-element union. We discovered that instead of being implemented with an unwieldy matrix-vector system of inner products, the recovered solution's value at a given interface could be recast as a set of derivative-based corrections to a weighted average of the competing limits of U^h. The resulting derivative-based recovery approach has two advantages over the traditional inner-product approach. First, it allows a specific Recovery-assisted DG approach to be recast in terms of just the approximation U^h and its derivatives, which could be advantageous for analysis. Second, the derivative-based form substantially reduces the complexity of the recovery process. On either quadrilateral elements in 2D or hexahedral elements in 3D, the recovered solution at an element-element interface can be implemented as a linear combination of pre-computed recovery weights and face-normal derivative jumps, where the jumps are simple to calculate thanks to the polynomial nature of each element's solution basis.

8.2.2 Chapter 3: The Interface Gradient Recovery Family

An ample supply of stable DG schemes for the diffusion equation were obtained by combining the Recovery concept with the traditional mixed formulation for handling second-order PDE systems. In the case of the non-compact schemes of the Interface Gradient Recovery (IGR) family, accuracy is maximized by using the full-order recovery operator for both the common solution $\tilde{U}$ and the common gradient $\tilde{\boldsymbol{\sigma}}$ along each interface. Among the compact members of the family, we discovered that scheme accuracy is primarily affected by the technique used to calculate $\tilde{U}$ along each interface. The strategy for calculating $\tilde{\boldsymbol{\sigma}}$, which determines the interface flux $\tilde{\boldsymbol{\mathcal{G}}}$, primarily affects

scheme stability, allowable timestep size, and robustness on non-Cartesian meshes. As we hypothesized (and hoped), the strategy of using the recovery operator to estimate solution quantities along element-element interfaces produced superior numerical schemes (with regard to accuracy at fixed p) compared to the state-of-the art.

The relation between the IGR family and the Recovery-based DG schemes of Chapter 6 merits special attention. For the 2D shear diffusion equation on Cartesian elements, both the GR-II scheme of the IGR family and the RDG-1x++CO scheme of Lo & van Leer [69] achieve order $3p+1$ accuracy for odd p and order $3p+2$ accuracy for even p. The GR-II scheme is simpler to implement, but the RDG-1x++CO scheme has a smaller stencil; where the GR-II stencil requires two layers of face-connected neighbor elements, the RDG-1x++CO scheme involves only the vertex-connected neighbors (see Figure 2.5). The Compact Gradient Recovery (CGR) schemes achieve stable and consistent discretization of the shear diffusion equation on both Cartesian and simplex elements in 2D while leveraging the accuracy of the Recovery operator. The Recovery-based DG family has so far been unable to replicate this feat, and this deficiency in the Recovery-based DG family was the motivation for the formation of the Compact Gradient Recovery schemes.

8.2.3 Chapter 4: Recovery-assisted Advection Schemes

In the discretization of advection problems, the Modal ICB reconstruction scheme of Khieu & Johnsen [56] achieves superior resolving efficiency compared to the conventional upwind DG method via prudent application of a biased recovery operation for calculating the advective interface fluxes. The resolving efficiency and allowable explicit timestep size of the original Modal ICB advection scheme can be significantly improved by altering the reconstructed solution's constraint in the non-dominant element. In particular, our proposed Lagrange ICB scheme, which places heavy emphasis on matching the reconstructed solutions to the near-interface behavior of the discontinuous DG polynomials, achieved superior performance compared to the original ICB scheme. With regard to the multidimensional case, the reported results (along with the dipole-wall interaction results in Chapter 5) show that the scheme's accuracy advantage extends to the 2D case

on quadrilateral elements. However, we were unable to design stable ICB schemes for simplex elements in 2D, and this deficiency needs to be addressed in the future. While it was not reported in Chapter 4, we also had success in simulating 2D shocked flows by combining the Lagrange ICB scheme with a generalization of the smooth artificial viscosity approach proposed by Reisner et al. [89]. In fact, the image decorating the second page of this document is from one of our submitted simulations for the shock-vortex interaction test case at the Fifth International Workshop on High-Order CFD Methods. Based on our various results with shocked flows, we conclude that the use of the biased recovery operation in the DG framework does not prevent stable shock-capturing.

8.2.4 Chapter 5: Recovery-assisted Advection-Diffusion Schemes

The combination of the biased recovery approach for advection (the ICB schemes) and the full-order recovery operator inside the mixed formulation (the Compact Gradient Recovery method) produced an exceptional pair of Recovery-assisted DG schemes for advection-diffusion problems. Fourier analysis showed the Recovery-assisted advection-diffusion schemes to be superior to the conventional DG approach. The new schemes consistently provided better accuracy than the conventional DG approach at a given p throughout the suite of test problems, both linear and nonlinear. Our limited experimentation suggests that the extremely low dissipation of the second Recovery-assisted scheme (RAD2) renders it unfit for unstructured meshes, but this issue should be explored further before making a definitive judgment on the worth of the RAD2 scheme.

8.2.5 Chapter 6: Boundary Procedures for van Leer & Lo's Recovery-based DG Method

The outstanding performance of the Recovery-based DG (RDG) method [67] for diffusion problems is degraded when Dirichlet or Neumann boundary conditions are applied instead of spatially periodic boundary conditions. This shortcoming is specific to shear-diffusion problems (not the scalar Laplacian). In the case of Dirichlet boundary conditions, we retained greater than order $3p$ convergence in the cell-average error by pulling in the ideal amount of information from bordering

elements when calculating the solution gradient along boundary interfaces. The new boundary scheme was less effective in the case of a Neumann boundary condition but still retained better than order $2p$ convergence. The accuracy of the boundary scheme in the RDG method is adversely affected if the computational stencil for the boundary interface's gradient calculation becomes too large. This particular finding was the greatest surprise encountered during our studies, as the general trend in spatial discretization schemes is that for smooth solutions, accuracy improves as the stencil is expanded.

8.2.6 Chapter 7: Recovery in the Flux Reconstruction Method

In the context of discontinuous finite element methods, namely DG and the Flux Reconstruction (FR) method, the Fourier analysis technique is now applicable on nonuniform mesh geometries. However, our approach requires a single four-element pattern to repeat across the mesh, so it is not applicable to completely unstructured mesh geometries. The Recovery-assisted Flux Reconstruction scheme has the potential to achieve superior orders of accuracy compared to other FR methods for the diffusion equation. However, the correction polynomials (a unique aspect of the FR method) must be properly chosen to achieve order of accuracy greater than $2p$, regardless of whether or not the recovery operator is employed to calculate interface quantities.

8.3 Future Work

This work has opened the door to additional research questions. We suggest that the following topics be addressed in future studies.

8.3.1 Implicit Time Integration

For steady-state problems and prohibitively stiff unsteady problems, it is typical to pair the DG spatial discretization with an implicit time integration technique, such as the Jacobian-free Newton-Krylov approach [60]. The results reported in this document involve explicit time integration only,

and it remains to be seen whether or not the improved accuracy of the Recovery-assisted DG approach (for either advection, diffusion, or advection-diffusion) extends to simulations involving implicit time integration.

8.3.2 Flux Reconstruction and Biased Recovery

The interface-centered binary (ICB) reconstruction schemes presented for advection problems in Chapter 4 can easily be extended to Huynh's Flux Reconstruction (FR) method, which some see as a generalization of the DG method. The correction polynomials of the FR method can be tuned to optimize certain properties of the FR discretization, for example the allowable explicit timestep size or the dispersion/dissipation relation (see [46, 107] for excellent discussions on this topic). We expect that combining the biased recovery operation with an appropriate choice of correction polynomial would yield an exceptional high-order discretization for advection problems with respect to both resolving efficiency and explicit timestep size.

8.3.3 Dispersion Relation Optimization

Section 2.6.6 showed that the recovery operation can be recast as a weighted average of DG polynomial limits and derivative jumps (multiplied by recovery weights) at element-element interfaces. In Section 4.2.1 it was demonstrated that the recovery weights are present in the primal form of the ICB discretization for the linear advection adquation. We suspect that these weights need not correspond to a specific recovery operator; instead, one could tune the weights at a given p to optimize the dispersion relation. The idea of tuning the discretization to optimize the dispersion relation at a given stencil size has been explored extensively in the finite difference community [97, 24, 93, 66]. Asthana & Jameson [5] showed how correction polynomials could be tuned to optimize the dispersion relation in the FR method, yielding promising results in the simulation of the Taylor-Green vortex flow [17]. We expect that the recovery weights of the derivative-based recovery operation could be tuned in a similar manner to minimize wave propagation error in both the FR and DG frameworks. The need to maintain stability in the spatial discretization could make the optimiza-

tion strategy challenging, so it seems like an ideal project for an ambitious new graduate student.

8.3.4 Analysis of Primal Form of Diffusion Schemes

As with the ICB scheme for advection, it should be possible to write the Compact Gradient Recovery (CGR) scheme of Chapter 3 for diffusion problems in the primal form using the recovery weights from the derivative-based recovery operator. Then, one could attempt to prove energy stability using the primal form itself. Such analysis might uncover the reason for the instability of certain CGR configurations on non-Cartesian meshes. The non-compact GR-II and GR-VI schemes of Chapter 3 might also benefit from such analysis, though writing out the primal form would be considerably more difficult due to the non-compact stencil.

8.3.5 Recovery as a Diagnostic Tool

Throughout this document, the recovery operator was employed as means to obtain highly accurate solution approximations along element-element interfaces while populating the DG weak form. However, its use can be extended beyond this purpose. Frequently, application of the DG method requires some form of "resolution detector." For example, Chapelier & Lodato [22] probe the decay rate of the Legendre coefficients in each element to determine how much sub-grid dissipation is needed in application of the DG method to large-eddy simulation. Similarly, Persson & Peraire [84] use the solution coefficient decay rate in modal space to decide where to apply artificial viscosity in the context of shock-capturing with the DG method. The recovery operator, in either the biased or full-order form, could be employed in a similar fashion. If the DG polynomials and the recovered solution agreed in a pointwise fashion over a two-element union, it would be apparent that the numerical approximation is well-resolved. Outside of this ideal scenario, the disagreement between the DG data and any recovered solution at a given location in the domain could be used as a gauge for how close the simulation is to properly resolving the flow.

8.3.6 Recovery DG in the Incompressible Navier-Stokes Equations

Our primary criticism of the older Recovery-based DG schemes for diffusion problems is that as discussed in Section 6.2, the family requires a non-compact stencil for dealing with shear diffusion terms, as one encounters in the compressible Navier-Stokes (NS) equations. However, the viscous stress terms in the *incompressible* NS equations involve the Laplacian operation on the velocity components, without any cross-derivative terms ultimately appearing in the divergence of the diffusive flux terms. Thus, the compact RDG-2x scheme, which performs exceptionally well for the scalar Laplacian but fails for the shear diffusion problem, might work well for the diffusive terms in the incompressible NS equations. Recently, many authors have explored the application of the DG method to the incompressible NS equations [34, 53, 12, 116, 59], and the use of the RDG-2x scheme might provide a superior spatial discretization compared to other DG approaches.

Appendices

Appendix A

The Recovery Procedure

For clarity, we describe the recovery procedure from a linear algebra perspective for the classical, full-order recovery operator. The setup is the same as in Chapter 2: the goal is to form the recovered polynomial within the dimension-$2K$ basis $\boldsymbol{\psi}$ over the union $\mathcal{U} = \Omega_A \cup \Omega_B$. The basis functions of Ω_A and Ω_B are the sets $\boldsymbol{\phi_A}$ and $\boldsymbol{\phi_B}$, respectively. The goal of the discrete recovery operator, denoted $\mathcal{R}$, is to populate the values of the recovered solution at a discrete set of Q_S quadrature points along $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$. To begin, the recovery constraints (2.65) are rewritten by integrating the recovery basis functions against the DG basis functions,

$$\begin{bmatrix} \int_{\Omega_A} \psi^{col}\phi_A^0 d\boldsymbol{x} \\ \int_{\Omega_A} \psi^{col}\phi_A^1 d\boldsymbol{x} \\ \vdots \\ \int_{\Omega_A} \psi^{col}\phi_A^{K-1} d\boldsymbol{x} \\ ------ \\ \int_{\Omega_B} \psi^{col}\phi_B^0 d\boldsymbol{x} \\ \int_{\Omega_B} \psi^{col}\phi_B^1 d\boldsymbol{x} \\ \vdots \\ \int_{\Omega_B} \psi^{col}\phi_B^{K-1} d\boldsymbol{x} \end{bmatrix} \begin{bmatrix} \hat{f}_{\mathcal{I}}^0 \\ \hat{f}_{\mathcal{I}}^1 \\ \vdots \\ \hat{f}_{\mathcal{I}}^{2K-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M_A} & 0 \\ --- & --- \\ 0 & \boldsymbol{M_B} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{A}} \\ \hat{\mathbf{U}}_{\mathbf{B}} \end{bmatrix}, \tag{A.1}$$

where the element mass matrices have been employed to account for the integration of U^h against $\boldsymbol{\phi}$ in the recovery constraints (2.65). The $2K \times 2K$ matrix on the left-hand side of the recovery

system (2.62) is inverted to yield the recovery coefficients $\hat{\boldsymbol{f}}_{\boldsymbol{I}}$ given the DG DOFs $\hat{\mathbf{U}}_{\mathbf{A}}$ and $\hat{\mathbf{U}}_{\mathbf{B}}$. Then, a matrix containing the recovery basis functions is employed to populate the distribution of f_I along the interface:

$$\begin{bmatrix} f_I(\boldsymbol{r_1}) \\ f_I(\boldsymbol{r_2}) \\ \vdots \\ f_I(\boldsymbol{r_{GQ_s}}) \end{bmatrix} = \begin{bmatrix} \psi^{col}(\boldsymbol{r_1}) \\ \psi^{col}(\boldsymbol{r_2}) \\ \vdots \\ \psi^{col}(\boldsymbol{r_{GQ_s}}) \end{bmatrix} \begin{bmatrix} \int_{\Omega_A} \psi^{col} \phi_A^0 d\boldsymbol{x} \\ \int_{\Omega_A} \psi^{col} \phi_A^1 d\boldsymbol{x} \\ \vdots \\ \int_{\Omega_A} \psi^{col} \phi_A^{K-1} d\boldsymbol{x} \\ ------ \\ \int_{\Omega_B} \psi^{col} \phi_B^0 d\boldsymbol{x} \\ \int_{\Omega_B} \psi^{col} \phi_B^1 d\boldsymbol{x} \\ \vdots \\ \int_{\Omega_B} \psi^{col} \phi_B^{K-1} d\boldsymbol{x} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{M_A} & 0 \\ --- & --- \\ 0 & \boldsymbol{M_B} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{A}} \\ \hat{\mathbf{U}}_{\mathbf{B}} \end{bmatrix} = \boldsymbol{\mathcal{R}} \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{A}} \\ \hat{\mathbf{U}}_{\mathbf{B}} \end{bmatrix}. \tag{A.2}$$

The recovery operator, in discrete form, is the $Q_S \times (2K)$ matrix $\boldsymbol{\mathcal{R}}$, where Q_S is the number of quadrature points (where we need the solution) along the interface. It is formed from the product of the recovery basis matrix, the inverse of the left-hand matrix of the recovery system (A.1), and the mass matrices of the two elements. The discrete recovery operator yields the distribution of the recovered solution along the interface given the DOF $\hat{\mathbf{U}}_{\mathbf{A}}$ and $\hat{\mathbf{U}}_{\mathbf{B}}$ of the neighboring elements.

In practice, the discrete recovery operator $\boldsymbol{\mathcal{R}}$ is computed for every interface in the domain during the code's initialization phase. Then, whenever the DG residual calculation requires the reovered solution to be calculated, the pre-computed discrete recovery operator is combined with the DOFs of the neighboring elements to populate the recovered solution along the interface. The strategy of storing the discrete recovery operator has also been applied by Ferrero et al. [35]. We note that the use of the discrete recovery operator comes at some cost: the pre-computed matrix $\boldsymbol{\mathcal{R}}$ contains $Q_S \times (2K)$ entries. In the general case of an unstructured mesh, a distinct discrete Recovery matrix must be stored for each element-element interface. We believe this storage requirement to be acceptable because it scales with the DOF count (as opposed the the square of $nDOF$ or the

cube of *nDOF*, etc.).

Appendix B

Recovery Weights: Full-Order Recovery

The recovery weights $\boldsymbol{C}$ in the derivative-based implementation of the full-order recovery operator are given here. We first discuss $C_0(Q)$, where $Q = \frac{h_A - h_B}{h_A + h_B}$ is the uniformity index discussed in Section 2.6.6. $C_0(Q)$ is symmetric about $Q = 0$ and takes the following form:

$$C_0(Q) = \begin{cases} L_0(Q) & \text{for} \quad Q \geq 0 \\ 1 - L_0(-Q) & \text{for} \quad Q < 0, \end{cases} \tag{B.1}$$

where $L_0(Q)$ depends on p. For $j > 0$, the weights $C_j(Q)$ are symmetric about $Q = 0$; the symmetry is even when j is odd, and the symmetry is odd when j is even. Thus, for $j > 0$, each C_j is described as follows:

$$C_j(Q) = \begin{cases} L_j(Q) & \text{for} \quad Q \geq 0 \\ L_j(-Q) * (-1)^{j+1} & \text{for} \quad Q < 0. \end{cases} \tag{B.2}$$

For each combination of j and p, the function L_j is a degree 5 polynomial:

$$L_j(Q) = \sum_{n=0}^{5} Q^n \hat{L}_j^n. \tag{B.3}$$

The coefficients $\hat{L}_j^n$ are given in Table B.1. After using these coefficients to form the derivative-based recovery operator, an easy check on the implementation is as follows. First, fit the DG data to the function $U(x) = x^{2p+1}$ as described in Eq. (5.2c). Then, use the derivative-based recovery operator to approximate the interface solution; for the initial condition $U(x) = x^{2p+1}$ or any poly-

nomial of lower degree, the recovery operator should return the exact solution (within machine precision) on a 1D grid. If implementing the recovery operator on a uniform grid, only the $\hat{L}_j^0$ coefficients are needed because $Q = 0$ along all interfaces.

Table B.1: Interpolation coefficients for $L_j(Q)$ in the full-order recovery case. For $j > p$, $L_j(Q) = 0$.

j = 0 :	$\hat{L}_0^0$	$\hat{L}_0^1$	$\hat{L}_0^2$	$\hat{L}_0^3$	$\hat{L}_0^4$	$\hat{L}_0^5$
p = 1	5.000e-01	-7.500e-01	-1.007e-04	2.503e-01	-4.197e-04	2.092e-04
p = 2	5.000e-01	-9.375e-01	-1.251e-04	6.254e-01	-5.032e-04	-1.873e-01
p = 3	5.000e-01	-1.092e-00	-2.587e-02	1.258e-00	-4.360e-01	-2.194e-01
p = 4	5.000e-01	-1.226e-00	-8.376e-02	2.181e-00	-1.478e-00	7.554e-02
p = 5	5.000e-01	-1.345e-00	-1.678e-01	3.359e-00	-3.113e-00	7.316e-01
j = 1 :	$\hat{L}_1^0$	$\hat{L}_1^1$	$\hat{L}_1^2$	$\hat{L}_1^3$	$\hat{L}_1^4$	$\hat{L}_1^5$
p = 1	-8.333e-02	-8.333e-02	8.332e-02	8.336e-02	-3.147e-05	1.309e-05
p = 2	-4.688e-02	-4.687e-02	9.375e-02	9.376e-02	-4.689e-02	-4.687e-02
p = 3	-3.125e-02	-3.084e-02	8.608e-02	1.433e-01	-2.305e-01	5.900e-02
p = 4	-2.279e-02	-2.216e-02	7.908e-02	1.724e-01	-3.778e-01	1.702e-01
p = 5	-1.758e-02	-1.703e-02	7.669e-02	1.696e-01	-4.496e-01	2.417e-01
j = 2 :	$\hat{L}_2^0$	$\hat{L}_2^1$	$\hat{L}_2^2$	$\hat{L}_2^3$	$\hat{L}_2^4$	$\hat{L}_2^5$
p = 2	0	9.375e-03	1.875e-02	-3.895e-06	-1.874e-02	-9.378e-03
p = 3	0	5.298e-03	8.720e-03	5.840e-03	-5.189e-02	3.115e-02
p = 4	0	3.276e-03	6.040e-03	-2.571e-03	-3.513e-02	2.931e-02
p = 5	0	2.137e-03	5.421e-03	-1.187e-02	-1.033e-02	1.588e-02
j = 3 :	$\hat{L}_3^0$	$\hat{L}_3^1$	$\hat{L}_3^2$	$\hat{L}_3^3$	$\hat{L}_3^4$	$\hat{L}_3^5$
p = 3	1.488e-04	4.623e-04	-7.489e-04	-5.510e-04	-7.553e-03	8.091e-03
p = 4	5.813e-05	1.611e-04	-1.094e-04	-2.904e-03	2.341e-03	7.744e-04
p = 5	2.713e-05	6.351e-05	8.746e-05	-3.010e-03	5.035e-03	-2.114e-03
j = 4 :	$\hat{L}_4^0$	$\hat{L}_4^1$	$\hat{L}_4^2$	$\hat{L}_4^3$	$\hat{L}_4^4$	$\hat{L}_4^5$
p = 4	0	-2.357e-05	7.724e-08	-5.329e-04	1.407e-03	-7.828e-04
p = 5	0	-9.694e-06	-2.339e-06	-2.119e-04	7.276e-04	-5.250e-04
j = 5 :	$\hat{L}_5^0$	$\hat{L}_5^1$	$\hat{L}_5^2$	$\hat{L}_5^3$	$\hat{L}_5^4$	$\hat{L}_5^5$
p = 5	-1.057e-07	-4.069e-07	-1.491e-06	1.668e-05	1.267e-07	-2.312e-05

Appendix C

Recovery Weights: Modal ICB Recovery

The recovery weights $\boldsymbol{C}$ in the derivative-based implementation of the original biased recovery operator (Modal ICB reconstruction) are approximated by an eight-point interpolant:

$$C_j(Q) = \sum_{n=0}^{7} Q^n \hat{C}_j^n, \tag{C.1}$$

with the coefficients $\hat{C}_j^n$ given in Table C.1 and Table C.2 for $p \leq 5$. As with the full-order recovery case, if the mesh is uniform ($Q = 0$), then only the $\hat{C}_j^0$ coefficients are needed, and all others can be discarded.

Table C.1: Interpolation coefficients, $n \in \{0, 1, 2, 3\}$, for $C_j(Q)$ in the Modal ICB case. For $j > p$, $C_j(Q) = 0$.

j = 0 :	$\hat{C}_0^0$	$\hat{C}_0^1$	$\hat{C}_0^2$	$\hat{C}_0^3$
p = 1	8.333e-01	-3.889e-01	-2.963e-01	-9.870e-02
p = 2	9.546e-01	-1.697e-01	-2.563e-01	-2.129e-01
p = 3	9.889e-01	-5.808e-02	-1.315e-01	-1.579e-01
p = 4	9.975e-01	-1.868e-02	-5.709e-02	-6.549e-02
p = 5	9.995e-01	-7.147e-03	-2.692e-02	-6.716e-03
j = 1 :	$\hat{C}_1^0$	$\hat{C}_1^1$	$\hat{C}_1^2$	$\hat{C}_1^3$
p = 1	-8.333e-02	-9.415e-02	-3.764e-01	-6.032e-01
p = 2	-2.273e-02	-6.257e-02	-1.699e-01	-2.619e-01
p = 3	-5.556e-03	-2.483e-02	-7.258e-02	-1.120e-01
p = 4	-1.269e-03	-8.624e-03	-2.960e-02	-3.871e-02
p = 5	-2.769e-04	-3.460e-03	-1.362e-02	-4.334e-03
j = 2 :	$\hat{C}_2^0$	$\hat{C}_2^1$	$\hat{C}_2^2$	$\hat{C}_2^3$
p = 2	-7.576e-03	-1.100e-02	-8.042e-02	-1.368e-01
p = 3	-1.852e-03	-6.288e-03	-2.809e-02	-5.067e-02
p = 4	-4.230e-04	-2.519e-03	-1.046e-02	-1.564e-02
p = 5	-9.229e-05	-1.095e-03	-4.626e-03	-1.925e-03
j = 3 :	$\hat{C}_3^0$	$\hat{C}_3^1$	$\hat{C}_3^2$	$\hat{C}_3^3$
p = 3	-4.630e-04	-9.253e-04	-8.684e-03	-1.553e-02
p = 4	-1.058e-04	-5.058e-04	-2.869e-03	-4.702e-03
p = 5	-2.307e-05	-2.523e-04	-1.193e-03	-6.429e-04
j = 4 :	$\hat{C}_4^0$	$\hat{C}_4^1$	$\hat{C}_4^2$	$\hat{C}_4^3$
p = 4	-2.115e-05	-6.964e-05	-6.606e-04	-1.058e-03
p = 5	-4.614e-06	-4.459e-05	-2.514e-04	-1.639e-04
j = 5 :	$\hat{C}_5^0$	$\hat{C}_5^1$	$\hat{C}_5^2$	$\hat{C}_5^3$
p = 5	-7.690e-07	-6.217e-06	-4.550e-05	-3.087e-05

Table C.2: Interpolation coefficients, $n \in \{4, 5, 6, 7\}$, for $C_j(Q)$ in the Modal ICB case. For $j > p$, $C_j(Q) = 0$.

j = 0 :	$\hat{C}_0^4$	$\hat{C}_0^5$	$\hat{C}_0^6$	$\hat{C}_0^7$
p = 1	-3.273e-02	-1.114e-02	-4.204e-03	-1.201e-03
p = 2	-1.242e-01	-8.260e-02	-6.881e-02	-3.043e-02
p = 3	-1.190e-01	-1.457e-01	-2.051e-01	-1.083e-01
p = 4	-2.738e-02	-1.416e-01	-3.203e-01	-1.877e-01
p = 5	5.753e-02	-9.647e-02	-3.582e-01	-2.255e-01
j = 1 :	$\hat{C}_1^4$	$\hat{C}_1^5$	$\hat{C}_1^6$	$\hat{C}_1^7$
p = 1	6.572e-01	1.283e-00	-5.314e-01	-9.836e-01
p = 2	8.049e-02	3.244e-01	-1.555e-01	-2.873e-01
p = 3	-3.464e-02	7.106e-03	-1.245e-01	-1.138e-01
p = 4	-9.736e-03	-5.607e-02	-1.637e-01	-1.049e-01
p = 5	2.937e-02	-4.581e-02	-1.797e-01	-1.146e-01
j = 2 :	$\hat{C}_2^4$	$\hat{C}_2^5$	$\hat{C}_2^6$	$\hat{C}_2^7$
p = 2	9.401e-02	2.226e-01	-1.032e-01	-1.842e-01
p = 3	1.515e-03	3.341e-02	-5.247e-02	-6.101e-02
p = 4	-1.107e-03	-1.211e-02	-5.645e-02	-3.986e-02
p = 5	1.011e-02	-1.409e-02	-6.017e-02	-3.908e-02
j = 3 :	$\hat{C}_3^4$	$\hat{C}_3^5$	$\hat{C}_3^6$	$\hat{C}_3^7$
p = 3	4.836e-03	1.498e-02	-1.642e-02	-2.048e-02
p = 4	5.597e-04	-1.207e-03	-1.481e-02	-1.132e-02
p = 5	2.660e-03	-3.137e-03	-1.516e-02	-1.006e-02
j = 4 :	$\hat{C}_4^4$	$\hat{C}_4^5$	$\hat{C}_4^6$	$\hat{C}_4^7$
p = 4	3.304e-04	3.253e-05	-3.116e-03	-2.487e-03
p = 5	5.730e-04	-5.475e-04	-3.065e-03	-2.071e-03
j = 5 :	$\hat{C}_5^4$	$\hat{C}_5^5$	$\hat{C}_5^6$	$\hat{C}_5^7$
p = 5	1.039e-04	-8.280e-05	-5.165e-04	-3.524e-04

Appendix D

Recovery Weights: Lagrange ICB Recovery

The weights in the derivative-based implementation of the Lagrange ICB recovery operator are given here. Each weight $C_j(Q)$ is approximated in piecewise form:

$$C_j(Q) = \begin{cases} L_j(Q) & \text{for} \quad Q < 0 \\ M_j(Q) & \text{for} \quad Q \geq 0, \end{cases} \tag{D.1}$$

with the functions L and M being five-point and two-point interpolations, respectively, such that:

$$L_j(Q) = \sum_{n=0}^{4} Q^n \hat{L}_j^n \quad , \quad M_j(Q) = \sum_{n=0}^{1} Q^n \hat{M}_j^n. \tag{D.2}$$

The coefficients $\hat{\mathbf{L}}$ are given in Table D.1. The coefficients $\hat{\mathbf{M}}$ are given in Table D.2.

Table D.1: Interpolation coefficients for $L_j(Q)$ in the Lagrange ICB recovery case. For $j > p$, $L_j(Q) = 0$.

j = 0 :	$\hat{L}_0^0$	$\hat{L}_0^1$	$\hat{L}_0^2$	$\hat{L}_0^3$	$\hat{L}_0^4$
p = 1	0.60566	-0.56161	-0.079943	-0.0022809	-0.083903
p = 2	0.63796	-0.57736	-0.10584	0.017914	-0.093691
p = 3	0.65169	-0.5836	-0.11664	0.031731	-0.093622
p = 4	0.65874	-0.5867	-0.12208	0.040351	-0.092343
p = 5	0.66283	-0.58845	-0.12518	0.045877	-0.091167
j = 1 :	$\hat{L}_1^0$	$\hat{L}_1^1$	$\hat{L}_1^2$	$\hat{L}_1^3$	$\hat{L}_1^4$
p = 1	-0.083333	0.048593	0.063078	0.0018463	0.066197
p = 2	-0.040803	0.016413	0.03475	-0.030482	0.00090331
p = 3	-0.024184	0.0076108	0.020982	-0.029446	-0.012233
p = 4	-0.016008	0.0042728	0.013923	-0.023946	-0.013282
p = 5	-0.011385	0.0027125	0.0098938	-0.019002	-0.011782
j = 2 :	$\hat{L}_2^0$	$\hat{L}_2^1$	$\hat{L}_2^2$	$\hat{L}_2^3$	$\hat{L}_2^4$
p = 2	-0.0022993	0.0057778	-0.00061717	0.017183	0.020163
p = 3	-0.00083957	0.0020558	0.00023306	0.0069651	0.0093611
p = 4	-0.00037548	0.0009023	0.00015353	0.0030263	0.004374
p = 5	-0.0001922	0.00045598	8.3507e-05	0.0014799	0.0022473
j = 3 :	$\hat{L}_3^0$	$\hat{L}_3^1$	$\hat{L}_3^2$	$\hat{L}_3^3$	$\hat{L}_3^4$
p = 3	-1.9431e-05	8.332e-05	-0.00017612	8.3703e-05	-0.00046453
p = 4	-5.8712e-06	2.762e-05	-1.0634e-05	0.00019628	7.2132e-05
p = 5	-2.1632e-06	1.0614e-05	3.9351e-06	0.00010369	6.6487e-05
j = 4 :	$\hat{L}_4^0$	$\hat{L}_4^1$	$\hat{L}_4^2$	$\hat{L}_4^3$	$\hat{L}_4^4$
p = 4	-6.8855e-08	-5.3742e-07	-1.6118e-05	-6.0934e-05	-8.0213e-05
p = 5	-1.8261e-08	-7.0658e-08	-3.1778e-06	-1.1634e-05	-1.6193e-05
j = 5 :	$\hat{L}_5^0$	$\hat{L}_5^1$	$\hat{L}_5^2$	$\hat{L}_5^3$	$\hat{L}_5^4$
p = 5	-1.233e-10	-2.6501e-08	-4.2066e-07	-1.7265e-06	-1.9778e-06

Table D.2: Interpolation coefficients for $M_j(Q)$ in the Lagrange ICB recovery case. For $j > p$, $M_j(Q) = 0$.

j = 0 :	$\hat{M}_0^0$	$\hat{M}_0^1$
p = 1	0.60566	-0.60015
p = 2	0.63796	-0.63049
p = 3	0.65169	-0.64336
p = 4	0.65874	-0.64997
p = 5	0.66283	-0.65379
j = 1 :	$\hat{M}_1^0$	$\hat{M}_1^1$
p = 1	-0.083333	-0.12683
p = 2	-0.040803	-0.071057
p = 3	-0.024184	-0.04467
p = 4	-0.016008	-0.03049
p = 5	-0.011385	-0.022076
j = 2 :	$\hat{M}_2^0$	$\hat{M}_2^1$
p = 2	-0.0022993	-0.0040041
p = 3	-0.00083957	-0.0015508
p = 4	-0.00037548	-0.00071515
p = 5	-0.0001922	-0.00037269
j = 3 :	$\hat{M}_3^0$	$\hat{M}_3^1$
p = 3	-1.9431e-05	-3.5891e-05
p = 4	-5.8712e-06	-1.1183e-05
p = 5	-2.1632e-06	-4.1947e-06
j = 4 :	$\hat{M}_4^0$	$\hat{M}_4^1$
p = 4	-6.8855e-08	-1.3114e-07
p = 5	-1.8261e-08	-3.5409e-08
j = 5 :	$\hat{M}_5^0$	$\hat{M}_5^1$
p = 5	-1.233e-10	-2.3907e-10

Appendix E

Timestep Size for Advection-Diffusion Problems

This appendix describes how the timestep size is set for the advection-diffusion problems of Chapter 5. For test case 1,

$$\Delta t = \frac{S}{1/\Delta t_v + 1/\Delta t_a}, \quad \text{with } \Delta t_a = \frac{h_m C_{RK}}{a\rho_s^c} \quad \text{and } \Delta t_v = \frac{h_m^2 C_{RK}}{\mu\rho_s^v}, \tag{E.1}$$

where S is a safety factor, h_m is the minimum element width in the domain, $C_{RK} = 5$ is the multiplier associated with the stability region of the RK8 scheme, ρ_s^c is the spectral radius corresponding to advection, and ρ_s^v is the spectral radius corresponding to diffusion; these spectral radii are taken from Fourier anlaysis. For test cases 2, 3, and 4,

$$\Delta t = \frac{S}{1/\Delta t_v + 1/\Delta t_a}, \quad \text{with } \Delta t_a = \frac{h_m C_{RK}}{\rho_s^a \left(|\boldsymbol{V}| + a\right)_{max}} \quad \text{and } \Delta t_v = \frac{h_m^2 C_{RK}}{2^{N_D-1} \max[(4/3), (\gamma/Pr)]\, \rho_s^v \left(\frac{\mu}{\rho}\right)_{max}}, \tag{E.2}$$

where S is a safety factor, h_m is the minimum element width in the domain, $C_{RK} = 2.8$ is the multiplier associated with the stability region of the RK4 scheme, and $(|\boldsymbol{V}| + a)_{max}$ (with a representing the speed of sound) is the maximum wavespeed in the domain. The denominator of the Δt_v equation accounts for viscous and heat transfer effects in addition to accounting for the number of spatial dimensions.

Appendix F

Calculation of Interface Gradients

For the CGR method in 2D, an element's semi-connected gradient coefficient vector corresponding to an element-element interface has $2K$ entries: K entries $\hat{\boldsymbol{g}}^{\boldsymbol{x}}$ corresponding to the derivative w.r.t. x and K entries $\hat{\boldsymbol{g}}^{\boldsymbol{y}}$ for the derivative w.r.t. y. For interface $\mathcal{I} = \partial\Omega_A \cap \partial\Omega_B$, let $\boldsymbol{M1}_A^x$ be the matrix that obtains $\hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{A,s}$ from $\hat{\mathbf{U}}_A$ and $\hat{\mathbf{U}}_B$. Similarly, let $\boldsymbol{M1}_B^x$ obtain $\hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{B,s+}$ from $\hat{\mathbf{U}}_A \cup \hat{\mathbf{U}}_B$. Let $\boldsymbol{M1}_A^y$ and $\boldsymbol{M1}_B^y$ be configured similarly, but for $\hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{A,s}$ and $\hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{B,s+}$ instead. Given the gradient coefficients, the recovery operator $\mathcal{R}$ obtains the gradient along the interface:

$$\underline{\boldsymbol{\sigma}}_{\mathcal{I}}^{\boldsymbol{x}} = \begin{bmatrix} \frac{\partial U^h}{\partial x}(\boldsymbol{r}_1) \\ \vdots \\ \frac{\partial U^h}{\partial x}(\boldsymbol{r}_{Qs}) \end{bmatrix} = \mathcal{R}\begin{bmatrix} \hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{A,s} \\ \hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{B,s+} \end{bmatrix} \quad , \quad \underline{\boldsymbol{\sigma}}_{\mathcal{I}}^{\boldsymbol{y}} = \begin{bmatrix} \frac{\partial U^h}{\partial y}(\boldsymbol{r}_1) \\ \vdots \\ \frac{\partial U^h}{\partial y}(\boldsymbol{r}_{Qs}) \end{bmatrix} = \mathcal{R}\begin{bmatrix} \hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{A,s} \\ \hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{B,s+} \end{bmatrix}. \tag{F.1}$$

To avoid explicit storage of the semi-connected gradient coefficients, the recovery operator is combined with the interface's four $\boldsymbol{M1}$ matrices to populate the interface gradient approximation di-

rectly from $\hat{\mathbf{U}}_\mathbf{A}$ and $\hat{\mathbf{U}}_\mathbf{B}$:

$$\underline{\boldsymbol{\sigma}}_{\boldsymbol{\mathcal{I}}} = \begin{bmatrix} \frac{\partial U^h}{\partial x}(\boldsymbol{r}_1) \\ \vdots \\ \frac{\partial U^h}{\partial x}(\boldsymbol{r}_{Q_S}) \\ - - - \\ \frac{\partial U^h}{\partial y}(\boldsymbol{r}_1) \\ \vdots \\ \frac{\partial U^h}{\partial y}(\boldsymbol{r}_{Q_S}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{R}} & 0 \\ 0 & \boldsymbol{\mathcal{R}} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{A,s} \\ \hat{\boldsymbol{g}}^{\boldsymbol{x}}{}_{B,s+} \\ \\ \hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{A,s} \\ \hat{\boldsymbol{g}}^{\boldsymbol{y}}{}_{B,s+} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{R}} & 0 \\ 0 & \boldsymbol{\mathcal{R}} \end{bmatrix} \begin{bmatrix} \boldsymbol{M1}^x_A \\ \boldsymbol{M1}^x_B \\ \\ \boldsymbol{M1}^y_A \\ \boldsymbol{M1}^y_B \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}}_A \\ \hat{\mathbf{U}}_B \end{bmatrix}. \tag{F.2}$$

The product of the recovery operator and the $\boldsymbol{M1}$ matrices on the far right-hand side of the interface gradient system (Eq. F.2) is precalculated and stored for the duration of the simulation. Then, given $\hat{\mathbf{U}}_\mathbf{A} \cup \hat{\mathbf{U}}_\mathbf{B}$, the gradient $\tilde{\boldsymbol{\sigma}}$ at all quadrature points along the shared interface is available by a single matrix-vector multiplication. The BR2 scheme requires that an averaging operator be substituted for $\boldsymbol{\mathcal{R}}$.

Appendix G

Additional Fourier Analysis Results, Flux Reconstruction

This appendix contains two collections of data. The first collection is the report of Fourier analysis results for the RDG-2x scheme on Cartesian elements. The second collection is the order of accuracy analysis for the I-centered, I-continuous, and REAFR schemes on the Cartesian grid when $\alpha = \frac{\pi}{4}$.

G.1 RDG-2x Results:

Fourier analysis results of the RDG-2x scheme on the Cartesian mesh is provided in Table G.1. For this scheme, instead of sweeping over α values, we present only the $\alpha = \frac{\pi}{4}$ results. These results were sufficient to prove the scheme unstable for $p > 2$ and $\theta = \frac{1}{2}$. Note that the scheme performance is independent of the correction field chosen for the $\boldsymbol{\gamma}$ calculation and the χ parameter. This independence is a result of the fact that the RDG-2x scheme uses the gradient of the recovered solution to calculate the common interface gradient, so the $\boldsymbol{\gamma}$ calculation is removed from the discretization.

G.2 Cartesian Analysis of I-centered, I-continuous, and REAFR schemes with $\alpha = \frac{\pi}{4}$:

The tables in Section 7.3.5.1 list the calculated orders of accuracy for $\alpha = \frac{\pi}{7}$. In addition, the orders of accuracy in the $\alpha = \frac{\pi}{4}$ case are listed in Table G.2, Table G.3, and Table G.4 for $p = 1$, $p = 2$, and $p = 3$, respectively.

Table G.1: RDG-2x Analysis on Cartesian Mesh, $p \in \{1, 2, 3\}$ with $\alpha = \frac{\pi}{4}$ (no sweep over α values). Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field, such that each row represents a particular choice of p, interface solution strategy, χ, θ, and the correction field for $\nabla \cdot \boldsymbol{Q}$. Unstable schemes have an **X** listed for the spectral radius and order or accuracy.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy	Spectral Radius
1	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{31, 31, 31, 31}
1	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, 4}	{33, 33, 33, 33}
1	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{31, 31, 31, 31}
1	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{33, 33, 33, 33}
1	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{21, 21, 21, 21}
1	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, 2}	{24, 24, 24, 24}
1	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{21, 21, 21, 21}
1	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{24, 24, 24, 24}
1	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}	{19, 19, 19, 19}
1	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, 2}	{19, 19, 19, 19}
1	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}	{19, 19, 19, 19}
1	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}	{19, 19, 19, 19}
2	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{8, 8, 8, 8}	{66, 66, 66, 66}
2	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, 4}	{100, 100, 100, 100}
2	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{8, 8, 8, 8}	{66, 66, 66, 66}
2	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{100, 100, 100, 100}
2	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{67, 67, 67, 67}
2	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, 4}	{79, 79, 79, 79}
2	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{67, 67, 67, 67}
2	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{79, 79, 79, 79}
2	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}	{66, 66, 66, 66}
2	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, 4}	{72, 72, 72, 72}
2	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}	{66, 66, 66, 66}
2	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}	{72, 72, 72, 72}
3	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{12, 12, 12, 12}	{136, 136, 136, 136}
3	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}
3	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{12, 12, 12, 12}	{136, 136, 136, 136}
3	RDG-2x	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}
3	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{136, 136, 136, 136}
3	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}
3	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{136, 136, 136, 136}
3	RDG-2x	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}
3	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}	{136, 136, 136, 136}
3	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}
3	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}	{136, 136, 136, 136}
3	RDG-2x	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{**X**, **X**, **X**, **X**}	{**X**, **X**, **X**, **X**}

Table G.2: Order of accuracy on Cartesian Mesh with $\alpha = \frac{\pi}{4}$ and $p = 1$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes labelled with an **X**.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{4}$
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, **X**, **X**}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, **X**}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 2, 2, 2}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 2, **X**, **X**}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 2, 2, 2}
1	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 2, 2, **X**}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, **X**}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, **X**}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 2}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
1	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{2, 2, 2, 2}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{2, 2, 2, **X**}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{2, 2, 2, 2}
1	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{2, 2, 2, 2}

Table G.3: Order of accuracy on Cartesian Mesh with $\alpha = \frac{\pi}{4}$ and $p = 2$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes labelled with an **X**.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{4}$
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 6}
2	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, **X**}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, **X**, **X**, **X**}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, **X**}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, **X**}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 4, 4, 4}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 6}
2	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, **X**, **X**}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{4, 4, 4, 4}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{4, 4, 4, **X**}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{4, 4, 4, 4}
2	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{4, 4, 4, 4}

Table G.4: Order of accuracy on Cartesian Mesh with $\alpha = \frac{\pi}{4}$ and $p = 3$. Each row corresponds to four possible choices for the $\boldsymbol{\gamma}$ correction field. Unstable schemes labelled with an **X**.

p	Interface Sol. Strategy	Correction Field, $\nabla \cdot \boldsymbol{Q}$	Correction Field, $\boldsymbol{\gamma}$	χ	θ	Order of Accuracy, $\alpha = \frac{\pi}{4}$
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	I-centered	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	I-centered	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	I-centered	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{8, 6, 6, 6}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{8, **X**, **X**, **X**}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{8, 6, 6, 6}
3	I-continuous	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{8, 6, **X**, **X**}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	I-continuous	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	I-continuous	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{8, 8, 8, 6}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{8, **X**, **X**, **X**}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{8, 8, 8, 8}
3	REAFR	g_{DG}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{8, 8, 8, 8}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	REAFR	g_{SD}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	0	{6, 6, 6, 6}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	1	$\frac{1}{2}$	{6, 6, **X**, **X**}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	0	{6, 6, 6, 6}
3	REAFR	g_{Hu}	$\{g_{Le}, g_{DG}, g_{SD}, g_{Hu}\}$	2	$\frac{1}{2}$	{6, 6, 6, 6}

BIBLIOGRAPHY

[1] D. S. Abdi and F. X. Giraldo. Efficient construction of unified continuous and discontinuous Galerkin formulations for the 3D Euler equations. *J. Comput. Phys.*, 320:46–68, sep 2016.

[2] M. Alhawwary and Z. J. Wang. Fourier analysis and evaluation of DG, FD and compact difference methods for conservation laws. *J. Comput. Phys.*, 373:835–862, nov 2018.

[3] D. N. Arnold. An Interior Penalty Finite Element Method with Discontinuous Elements. *SIAM J. Numer. Anal.*, 19(4):742–760, aug 1982.

[4] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2002.

[5] K. Asthana and A. Jameson. High-Order Flux Reconstruction Schemes with Minimal Dispersion and Dissipation. *J. Sci. Comput.*, 62(3):913–944, mar 2015.

[6] G. E. Barter and D. L. Darmofal. Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation. *J. Comput. Phys.*, 229(5):1810–1827, mar 2010.

[7] F. Bassi, L. Botti, A. Colombo, D. A. Di Pietro, and P. Tesini. On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations. *J. Comput. Phys.*, 231(1):45–65, 2012.

[8] F. Bassi, A. Crivellini, S. Rebay, and M. Savini. Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations. *Comput. Fluids*, 34(4-5):507–540, 2005.

[9] F. Bassi and S. Rebay. A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations. *J. Comput. Phys.*, 131(2):267–279, mar 1997.

[10] F. Bassi and S. Rebay. High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations. *J. Comput. Phys.*, 138(2):251–285, dec 1997.

[11] 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. *Int. J. Numer. Meth. Fluids*, 76(8):522–548, nov 2014.

[12] L. Botti, A. Colombo, and F. Bassi. h-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems. *J. Comput. Phys.*, 347:382–415, 2017.

[13] S. Brdar, A. Dedner, and R. K. Ofkorn. Compact and stable discontinuous Galerkin methods for convection-diffusion problems. *SIAM J. Sci. Comput.*, 34(1):263–282, 2012.

[14] F. Brezzi, B. Cockburn, L. D. Marini, and E. Süli. Stabilization mechanisms in discontinuous Galerkin finite element methods. *Comp. Meth. in App. Mech. and Eng.*, 195(25-28):3293–3310, may 2006.

[15] F. Brezzi, G. Manzini, D. Marini, P. Pietra, and A. Russo. Discontinuous Galerkin approximations for elliptic problems. *Numer. Meth. Part. D.E.*, 16(4):365–378, 2000.

[16] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comp. Meth. in App. Mech. and Eng.*, 32(1-3):199–259, sep 1982.

[17] J. R. Bull and A. Jameson. Simulation of the TaylorGreen Vortex Using High-Order Flux Reconstruction Schemes. *AIAA Journal*, 53(9):2750–2761, sep 2015.

[18] G. Buresti. A note on Stokes' hypothesis. *Acta Mechanica*, 226(10):3555–3559, oct 2015.

[19] J. R. Cash and A. H. Karp. A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides. *ACM Trans. Mathematical Software*, 16(3):201–222, 1990.

[20] P. Castonguay, P. E. Vincent, and A. Jameson. A New Class of High-Order Energy Stable Flux Reconstruction Schemes for Triangular Elements. *J. Sci. Comput.*, 51(1):224–256, apr 2012.

[21] J. B. Chapelier, M. de la Llave Plata, F. Renac, and E. Lamballais. Evaluation of a high-order discontinuous Galerkin method for the DNS of turbulent flows. *Comput. Fluids*, 95:210–226, may 2014.

[22] J. B. Chapelier and G. Lodato. A spectral-element dynamic model for the Large-Eddy simulation of turbulent flows. *J. Comput. Phys.*, 321:279–302, sep 2016.

[23] A. Chaudhuri, G. B. Jacobs, W. S. Don, H. Abbassi, and F. Mashayek. Explicit discontinuous spectral element method with entropy generation based artificial viscosity for shocked viscous flows. *J. Comput. Phys.*, 332:99–117, mar 2017.

[24] P. H. Chiu and T. W. H. Sheu. On the development of a dispersion-relation-preserving dual-compact upwind scheme for convectiondiffusion equation. *J. Comput. Phys.*, 228(10):3640–3655, jun 2009.

[25] N. T. Clemens and V. Narayanaswamy. Low-Frequency Unsteadiness of Shock Wave/Turbulent Boundary Layer Interactions. *Annu. Rev. Fluid Mech.*, 46(1):469–492, jan 2014.

[26] B. Cockburn and C. W. Shu. The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463, 1997.

[27] B. Cockburn and C. W. Shu. Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *J. Sci. Comput.*, 16(3):89, 2001.

[28] E. Creuse and S. Nicaise. A posteriori error estimator based on gradient recovery by averaging for convection-diffusion-reaction problems approximated by discontinuous Galerkin methods. *IMA J. Numer. Anal.*, 33(1):212–241, jan 2013.

[29] D. De Grazia, G. Mengaldo, D. Moxey, P. E. Vincent, and S. J. Sherwin. Connections between the discontinuous Galerkin method and high-order flux reconstruction schemes. *Int. J. Numer. Meth. Fluids*, 75(12):860–877, aug 2014.

[30] C. C. De Wiart, K. Hillewaert, M. Duponcheel, and G. Winckelmans. Assessment of a discontinuous Galerkin method for the simulation of vortical flows at high Reynolds number. *Int. J. Numer. Meth. Fluids*, 74(7):469–493, 2014.

[31] M. Dumbser. Arbitrary high order $P_N P_M$ schemes on unstructured meshes for the compressible Navier-Stokes equations. *Comput. Fluids*, 39(1):60–76, jan 2010.

[32] M. Dumbser and C. D. Munz. ADER discontinuous Galerkin schemes for aeroacoustics. *Comptes Rendus Mécanique*, 333(9):683–687, sep 2005.

[33] M. Dumbser, O. Zanotti, R. Loubère, and S. Diot. A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws. *J. Comput. Phys.*, 278:47–75, 2014.

[34] F. Fambri and M. Dumbser. Semi-implicit discontinuous Galerkin methods for the incompressible NavierStokes equations on adaptive staggered Cartesian grids. *Comp. Meth. in App. Mech. and Eng.*, 324:170–203, sep 2017.

[35] A. Ferrero, F. Larocca, and G. Puppo. A robust and adaptive recovery-based discontinuous Galerkin method for the numerical solution of convection-diffusion equations. *Int. J. Numer. Meth. Fluids*, 77(2):63–91, jan 2015.

[36] D. Flad, A. Beck, and C. D. Munz. Simulation of underresolved turbulent flows by adaptive filtering using the high order discontinuous Galerkin spectral element method. *J. Comput. Phys.*, 313:1–12, may 2016.

[37] G. Gassner, F. Lörcher, and C. D. Munz. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *J. Comput. Phys.*, 224(2):1049–1063, jun 2007.

[38] C. Geuzaine and J. F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Meth. Fluids*, 79(11):1309–1331, sep 2009.

[39] W. Guo, X. Zhong, and J. M. Qiu. Superconvergence of discontinuous Galerkin and local discontinuous Galerkin methods: Eigen-structure analysis based on Fourier approach. *J. Comput. Phys.*, 235:458–485, 2013.

[40] R. Hartmann and P. Houston. An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 227(22):9670–9685, 2008.

[41] R. Hartmann and T. Leicht. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *Int. J. Numer. Meth. Fluids*, 82(6):316–333, oct 2016.

[42] M. T. Henry de Frahan. *Numerical simulations of shock and rarefaction waves interacting with interfaces in compressible multiphase flows*. Ph. d. dissertation, University of Michigan, 2016.

[43] M. T. Henry de Frahan and E. Johnsen. High-order Discontinuous Galerkin Methods Applied to Multiphase Flows. *AIAA Paper 2015-3045*, jun 2015.

[44] M. T. Henry de Frahan, S. Varadan, and E. Johnsen. A new limiting procedure for discontinuous Galerkin methods applied to compressible multiphase flows with shocks and interfaces. *J. Comput. Phys.*, 280:489–509, jan 2015.

[45] A. E. Honein and P. Moin. Higher entropy conservation and numerical stability of compressible turbulence simulations. *J. Comput. Phys.*, 201(2):531–545, dec 2004.

[46] H. T. Huynh. A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods. *AIAA Paper 2007-4079*, 2007.

[47] H. T. Huynh. A Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin for Diffusion. *AIAA Paper 2009-403*, jan 2009.

[48] E. Johnsen, J. Larsson, A. V. Bhagatwala, W. H. Cabot, P. Moin, B. J. Olson, P. S. Rawat, S. K. Shankar, B. Sjögreen, H. C. Yee, X. Zhong, and S. K. Lele. Assessment of high-resolution methods for numerical simulations of compressible turbulence with shock waves. *J. Comput. Phys.*, 229(4):1213–1237, 2010.

[49] P. E. Johnson and E. Johnsen. Progress Towards the Application of the Recovery-Based Discontinuous Galerkin Method to Practical Flow Physics Problems. *AIAA Paper 2016-3331*, jun 2016.

[50] P. E. Johnson and E. Johnsen. A New Family of Discontinuous Galerkin Schemes for Diffusion Problems. *AIAA Paper 2017-3444*, 2017.

[51] P. E. Johnson and E. Johnsen. A Compact Discontinuous Galerkin Method for Advection-Diffusion Problems. *AIAA Paper 2018-1091*, 2018.

[52] P. E. Johnson and E. Johnsen. The Compact Gradient Recovery Discontinuous Galerkin Method for Diffusion Problems. *Under Review, J. Comput. Phys.*, 2019.

[53] A. Karakus, N. Chalmers, K. Świrydowicz, and T. Warburton. A GPU accelerated discontinuous Galerkin incompressible flow solver. *J. Comput. Phys.*, 390:380–404, 2019.

[54] G. H. Keetels, U. D'Ortona, W. Kramer, H. J. H. Clercx, K. Schneider, and G. J. F. van Heijst. Fourier spectral and wavelet solvers for the incompressible Navier-Stokes equations with volume-penalization: Convergence of a dipole-wall collision. *J. Comput. Phys.*, 227(2):919–945, dec 2007.

[55] L. H. Khieu, K. Fidkowski, and E. Johnsen. Discontinuous Galerkin for Advection with Interface-Centered Reconstruction. *AIAA Paper 2016-1337*, jan 2016.

[56] L. H. Khieu and E. Johnsen. Analysis of Improved Advection Schemes for Discontinuous Galerkin Methods. *AIAA Paper 2014-3221*, jun 2014.

[57] L. H. Khieu, B. van Leer, and M. Lo. Optimal Accuracy of Discontinuous Galerkin for Diffusion. *AIAA Paper 2013-2691*, jun 2013.

[58] K. Kitamura and E. Shima. Towards shock-stable and accurate hypersonic heating computations: A new pressure flux for AUSM-family schemes. *J. Comput. Phys.*, 245:62–83, jul 2013.

[59] B. Klein, F. Kummer, and M. Oberlack. A SIMPLE based discontinuous Galerkin solver for steady incompressible flows. *J. Comput. Phys.*, 237:235–250, 2013.

[60] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193(2):357–397, jan 2004.

[61] M. A. Kopera and F. X. Giraldo. Mass conservation of the unified continuous and discontinuous element-based Galerkin methods on dynamically adaptive grids with application to atmospheric simulations. *J. Comput. Phys.*, 297:90–103, sep 2015.

[62] D. A. Kopriva. A spectral multidomain method for the solution of hyperbolic systems. *Appl. Numer. Mathematics*, 2(3-5):221–241, oct 1986.

[63] D. A. Kopriva. Stability of Overintegration Methods for Nodal Discontinuous Galerkin Spectral Element Methods. *J. Sci. Comput.*, 76(1):426–442, jul 2018.

[64] L. Krivodonova. Limiters for high-order discontinuous Galerkin methods. *J. Comput. Phys.*, 226(1):879–896, sep 2007.

[65] S. Lee, S. K. Lele, and P. Moin. Eddy shocklets in decaying compressible turbulence. *Physics of Fluids A: Fluid Dynamics Physics of Fluids*, 3(April):657–664, 1991.

[66] V. Linders, M. Kupiainen, and J. Nordström. Summation-by-Parts operators with minimal dispersion error for coarse grid flow calculations. *J. Comput. Phys.*, 340:160–176, jul 2017.

[67] M. Lo. *A Space-Time Discontinuous Galerkin Method for Navier-Stokes with Recovery*. Ph. d. dissertation, University of Michigan, 2011.

[68] M. Lo and B. van Leer. Analysis and Implementation of Recovery-Based Discontinuous Galerkin for Diffusion. *AIAA Paper 2009-3786*, 2009.

[69] M. Lo and B. van Leer. Recovery-Based Discontinuous Galerkin for Navier-Stokes Viscous Terms. *AIAA Paper 2011-3406*, 2011.

[70] H. Luo, L. Luo, R. Nourgaliev, V. A. Mousseau, and N. Dinh. A reconstructed discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids. *J. Comput. Phys.*, 229(19):6961–6978, sep 2010.

[71] Y. Lv and M. Ihme. Entropy-bounded discontinuous Galerkin scheme for Euler equations. *J. Comput. Phys.*, 295:715–739, 2015.

[72] Y. Lv, P. C. Ma, and M. Ihme. On underresolved simulations of compressible turbulence using an entropy-bounded DG method: Solution stabilization, scheme optimization, and benchmark against a finite-volume solver. *Comp. Fluids*, 161:89–106, jan 2018.

[73] S. Marras, J. F. Kelly, M. Moragues, A. Müller, M. A. Kopera, M. Vázquez, F. X. Giraldo, G. Houzeaux, and O. Jorba. A Review of Element-Based Galerkin Methods for Numerical Weather Prediction: Finite Elements, Spectral Elements, and Discontinuous Galerkin. *Arch. Comput. Methods Eng.*, 23(4):673–722, dec 2016.

[74] G. Mengaldo, D. De Grazia, D. Moxey, P. E. Vincent, and S. J. Sherwin. Dealiasing techniques for high-order spectral element methods on regular and irregular grids. *J. Comput. Phys.*, 299:56–81, oct 2015.

[75] G. Mengaldo, R. C. Moura, B. Giralda, J. Peiró, and S. J. Sherwin. Spatial eigensolution analysis of discontinuous Galerkin schemes with practical insights for under-resolved computations and implicit LES. *Comp. Fluids*, 169:349–364, jun 2018.

[76] R. C. Moura, S. J. Sherwin, and J. Peiró. Linear dispersiondiffusion analysis and its application to under-resolved turbulence simulations using discontinuous Galerkin spectral/ hp methods. *J. Comput. Phys.*, 298:695–710, oct 2015.

[77] T. Mullin. Experimental Studies of Transition to Turbulence in a Pipe. *Annu. Rev. Fluid Mech.*, 43(1):1–24, jan 2011.

[78] J. N. Murugan and R. N. Govardhan. Shock wave-boundary layer interaction in supersonic flow over a forward-facing step. *J. Fluid Mech.*, 807:258–302, 2016.

[79] R. Nourgaliev, H. Luo, B. Weston, A. Anderson, S. Schofield, T. Dunn, and J. P. Delplanque. Fully-implicit orthogonal reconstructed Discontinuous Galerkin method for fluid dynamics with phase change. *J. Comput. Phys.*, 305:964–996, 2016.

[80] P. Orlandi. Vortex dipole rebound from a wall. *Phys. Fluids A: Fluid Dyn.*, 2(8):1429–1436, aug 1990.

[81] J. S. Park and C. Kim. Higher-order multi-dimensional limiting strategy for discontinuous Galerkin methods in compressible inviscid and viscous flows. *Comp. Fluids*, 96:377–396, jun 2014.

[82] J. Peraire and P. O. Persson. The Compact Discontinuous Galerkin (CDG) Method For Elliptic Problems. *SIAM J. Sci. Comput.*, 30(4):1806–1824, jan 2008.

[83] P. O. Persson. A sparse and high-order accurate line-based discontinuous Galerkin method for unstructured meshes. *J. Comput. Phys.*, 233:414–429, jan 2013.

[84] P. O. Persson and J. Peraire. Sub-Cell Shock Capturing for Discontinuous Galerkin Methods. *AIAA Paper 2006-112*, jan 2006.

[85] S. B. Pope. *Turbulent flows*. Cambridge University Press, 2000.

[86] P. J. Prince and J. R. Dormand. High order embedded Runge-Kutta formulae. *J. Comput. Appl. Math.*, 7(1):67–75, 1981.

[87] P. J. Pritchard. *Fox and McDonald's Introduction to Fluid Mechanics*. John Wiley & Sons, Hoboken, N.J., 8th ed. edition, 2011.

[88] W. H. Reed and T. R. Hill. Triangular Mesh Methods for the Neutron Transport Equation. Technical report, Los Alamos Scientific Laboratory, 1973.

[89] J. Reisner, J. Serencsa, and S. Shkoller. A Space-time Smooth Artificial Viscosity Method For Nonlinear Conservation Laws. *J. Comput. Phys.*, 235:912–933, apr 2012.

[90] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *J. Comput. Phys.*, 135:250–258, 1997.

[91] P. L. Roe. A Simple Explanation of Superconvergence for Discontinuous Galerkin Solutions to u t +u x = 0. *Commun. Comput. Phys*, 21(4):905–912, 2017.

[92] V. V. Rusanov. Calculation of Interaction of Non-Steady Shock Waves with Obstacles. *J. Comput. Math. Phys. USSR,*, pages 267–279, 1962.

[93] B. Sjögreen and H. C. Yee. Accuracy consideration by DRP schemes for DNS and LES of compressible flow computations. *Comp. Fluids*, 159:123–136, dec 2017.

[94] G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27(1):1–31, apr 1978.

[95] L. Song and Z. Zhang. Superconvergence property of an over-penalized discontinuous Galerkin finite element gradient recovery method. *J. Comput. Phys.*, 299:1004–1020, oct 2015.

[96] S. C. Spiegel, J. R. Debonis, and H. T. Huynh. Overview of the NASA Glenn Flux Reconstruction Based High-Order Unstructured Grid Code. *AIAA Paper 2016-1061*, 2016.

[97] C. K. W. Tam and J. C. Webb. Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics. *J. Comput. Phys.*, 107(2):262–281, aug 1993.

[98] M. Taylor, B. Wingate, and L. P. Bos. *Several new quadrature formulas for polynomial integration in the triangle*. feb 2005.

[99] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics : a practical introduction*. Springer-Verlag, 1999.

[100] B. van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *J. Comput. Phys.*, 23(3):276–299, mar 1977.

[101] B. van Leer, M. Lo, and M. van Raalte. A Discontinuous Galerkin Method for Diffusion Based on Recovery. *AIAA Paper 2007-4083*, 2007.

[102] B. van Leer and S. Nomura. Discontinuous Galerkin for Diffusion. *AIAA Paper 2005-5108*, jun 2005.

[103] B. van Leer and K. G. Powell. Introduction to Computational Fluid Dynamics. In *Encyclopedia of Aerosp. Eng.* John Wiley & Sons, Ltd, Chichester, UK, dec 2010.

[104] M. van Raalte, B. van Leer, and W. M. Keck. Bilinear Forms for the Recovery-Based Discontinuous Galerkin Method for Diffusion. *Commun. Comput. Phys.*, (5):683–693, 2008.

[105] V. Venkatakrishnan, S. Allmaras, D. Kamenetskii, and F. Johnson. Higher Order Schemes for the Compressible Navier-Stokes Equations. *AIAA Paper 2003-3987*, jun 2003.

[106] B. C. Vermeire and P. E. Vincent. On the behaviour of fully-discrete flux reconstruction schemes. *Comp. Meth. in App. Mech. and Eng.*, 315:1053–1079, mar 2017.

[107] P. E. Vincent, P. Castonguay, and A. Jameson. A new class of high-order energy stable flux reconstruction schemes. *J. Sci. Comput.*, 47(1):50–72, apr 2011.

[108] L. Wang, W. K. Anderson, J. T. Erwin, and S. Kapadia. Discontinuous Galerkin and Petrov Galerkin methods for compressible viscous flows. *Comp. Fluids*, 100:13–29, sep 2014.

[109] L. Wang and X. Y. Lu. Flow topology in compressible turbulent boundary layer. *J. Fluid Mech*, 703:255, 2012.

[110] Z. J. Wang. High-order methods for the Euler and Navier-Stokes equations on unstructured grids. *Prog. Aerosp. Sci.*, 43(1-3):1–41, 2007.

[111] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P. O. Persson, B. van Leer, and M. Visbal. High-order CFD methods: current status and perspective. *Int. J. Numer. Meth. Fluids*, 72(8):811–845, jul 2013.

[112] J. Watkins, K. Asthana, and A. Jameson. A numerical analysis of the nodal Discontinuous Galerkin scheme via Flux Reconstruction for the advection-diffusion equation. *Comput. Fluids*, 139:233–247, 2016.

[113] D. M. Williams, P. Castonguay, P. E. Vincent, and A. Jameson. Energy stable flux reconstruction schemes for advection-diffusion problems on triangles. *J. Comput. Phys.*, 250:53–76, oct 2013.

[114] A. R. Winters, R. C. Moura, G. Mengaldo, G. J. Gassner, S. Walch, J. Peiro, and S. J. Sherwin. A comparative study on polynomial dealiasing and split form discontinuous Galerkin schemes for under-resolved turbulence computations. *J. Comput. Phys.*, 372:1–21, nov 2018.

[115] M. Yu, Z. J. Wang, and Y. Liu. On the accuracy and efficiency of discontinuous Galerkin, spectral difference and correction procedure via reconstruction methods. *J. Comput. Phys.*, 259:70–95, feb 2014.

[116] F. Zhang, J. Cheng, and T. Liu. A direct discontinuous Galerkin method for the incompressible Navier-Stokes equations on arbitrary grids. *J. Comput. Phys.*, 380:269–294, 2019.

[117] M. Zhang and C. W. Shu. Fourier analysis for discontinuous Galerkin and related methods. *Science Bulletin*, 54(11):1809–1816, jun 2009.

[118] J. Zhu, X. Zhong, C. W. Shu, and J. Qiu. Runge-Kutta Discontinuous Galerkin Method with a Simple and Compact Hermite WENO Limiter. *Commun. Comput. Phys.*, 19(04):944–969, apr 2016.