A Space-Time Discontinuous Galerkin Method for Navier-Stokes with Recovery

by

Kwok Ho Marcus Lo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2011

Doctoral Committee:

Professor Bram van Leer, Chair
Professor Nikolaos D. Katopodes
Professor Philip L. Roe
Assistant Professor Krzysztof J. Fidkowski
Hung T. Huynh, NASA Glenn

ACKNOWLEDGEMENTS

I appreciate my loving mother, Katherine, for teaching me to rely on myself and to stand tall. I thank my dad, Alan, for allowing me to pursue my love for science. Thank you my brother, Michael, for taking care of me when I was almost dead on the hospital bed. I am thankful for my wonderful pets, Miumiu and Momo, for accompanying me through day and night, and through ups and downs. I am grateful for Hung Huynh's enthusiasm and advice in CFD. Cheers to my research comrade, Loc, for enduring my eccentricity. Lastly, I am infinitely happy for this wonderful life-learning experience with my friendly neighborhood professor, Bram van Leer and his wife, Lia.

PREFACE

Computational fluid dynamics (CFD) is the branch of fluid mechanics that uses numerical methods and algorithms to solve fluid problems. While there are currently a myriad of numerical methods being used and developed, this dissertation focuses on the Discontinuous Galerkin (DG) method because this method possesses superior properties in regard to adapting to problem geometry, to parallelizing on today's computer architecture, and to achieving super-convergence. Although DG works well in hyperbolic problems where discontinuities frequently arise, the development in parabolic problems, such as diffusion, is lagging. The purpose of this dissertation is to present a completely new concept, interface-centered recovery-based discontinuous Galerkin (RDG) for diffusion. Our goal is to develop a new diffusion scheme that is a quantum level better than existing methods; this includes higher-order of accuracy, weaker time-stepping restriction, and easier understanding and implementation. We begin by providing a historical overview of developments in DG for time-marching and diffusion schemes. The second chapter illustrates the generalized concept of recovery in a mathematical framework. In next chapter, we present analysis and numerical results in one dimension to facilitate understanding of the recovery concept. The fourth chapter expands the recovery concept to two dimensions and unstructured grids. We also include a new application of the recovery concept beyond diffusion. Recovery is used for enhancing the solution polynomial to a higher order; as a result the recovery procedure uses more isotropic information resulting in a higher order of accuracy. The choice of information is subject to optimization. The fifth chapter provides numerical results for solving the Navier-Stokes viscous terms. The final chapter is the development of a new time-marching scheme for DG, moment method, not restricted to diffusion. The moment method was first developed by Dr. Huynh (NASA Glenn) for piecewise-linear solution representation. We present the implementation and numerical results of higher-order moment schemes.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

CHAPTER I

Introduction

We introduce two new methods into the discontinuous Galerkin (DG) framework. The first is the recovery-based discontinuous Galerkin (RDG) method for diffusion, and the second is the Hancock-Huynh discontinuous Galerkin (HH-DG) method as a space-time method. This dissertation details the progress and challenges we have encountered, and is intended to be a guide to further research in these areas. The methods demonstrate exceptional performance for scalar partial differential equations (PDE) by outperforming existing methods. Our goal is to extend these methods to multi-dimensional nonlinear systems of equations. A good starting point to understanding these new methods is to introduce the DG framework and the person who invented it.

1.1 Boris Grigoryevich Galerkin

Who was Boris Grigoryevich Galerkin? He was a Russian/Soviet mathematician and engineer born in Polozk, Belarus in 1871 [1]. Galerkin was heavily involved in the Russian Social-Democratic Party in his early life and was arrested in 1906 for organizing strikes. Galerkin then abandoned his revolutionary activities and focused on science and engineering; he designed a boiler power plant while he was in prison. After being released in 1908, Galerkin became a teacher at the Saint Petersburg Polytechnical Institute and published numerous scientific articles on structures/frames and pivot systems. In 1915, Galerkin penned a landmark paper on the idea of an approximate method for boundary-value problems. Galerkin developed the weak formulation of the partial differential equation (PDE) completely independent of the variational method. The variational method finds an approximate solution to a system based on the optimization of a specially designed function which depends on carefully chosen

variables; the idea to derive exact or approximate equations from the condition that the function value must be stationary under perturbations is called the variational principle. However, not all problems entail a variational principle, and most of the classical problems suited for optimization have been solved already. The Galerkin method is much more versatile in handling a broad range of problems because it is not limited by the variational principle; although it is known that one can recover a variation method from a Galerkin method [11]. Today Galerkin's weak formulation is the foundation of many numerical algorithms spanning mechanics, thermodynamics, electromagnetism, hydrodynamics and many other disciplines. In the years leading to the end of his life in 1945, numerous types of Galerkin methods were developed; these include the Ritz-Galerkin, Bubnov-Galerkin and Petrov-Galerkin methods. These methods are now collectively called finite-element methods, and are very popular within the field of structural mechanics.

Perhaps the most interesting Galerkin method is the discontinuous Galerkin (DG) method developed in the 1970's. Unlike its predecessors for physical structures, where continuity of the solution is natural, DG allows for a discontinuous representation of the solution. This added freedom is helpful in solving differential equations for which the solution involves strongly varying gradients or even discontinuities as in compressible-flow. We begin by looking into the list of improvements which DG brought into computational fluid dynamics (CFD) over traditional finite-difference and finite-volume methods.

Figure 1.1: Boris Grigoryevich Galerkin

1.2 The discontinuous Galerkin method

The DG method is rapidly becoming the preferred method for CFD, in particular, because of the ease of achieving a high order of accuracy on irregular, adaptively refined grids. Low-order methods are still frequently employed in practical CFD applications where efficiency, stability and robustness are priorities, but they tend to have inadequate accuracy and fail to provide detailed information in complex CFD

problems. Therefore, the CFD community is now focusing on high-order methods where very accurate results obtained at a reasonable cost yield computational efficiency. DG achieves high order by increasing the number of subcell data, hence the increase in the order of accuracy is realized within a single cell instead of by interpolation between cells, as in traditional finite-volume and finite-difference methods. The trouble with high-order finite-volume and finite-difference methods is twofold. In the first place, constructing a highly accurate interpolant based on data on an unstructured, adaptively refined grid calls for great code complexity. In the second place, it requires a dramatic increase in stencil size, which in turn causes complications near the boundary of the computational domain and also at internal boundaries between subdomains handled by different processors. The problem with high-order interpolation near the domain boundary is often resolved with the introduction of many layers of ad-hoc ghost elements, or with a highly skewed stencil which may imperil stability. With regard to domain decomposition, long-range element interconnectivity heavily burdens the data-link communication between computer nodes in a parallelized computer architecture. As a result, the traditional high-order methods based on wide interpolation stencils are no longer desirable on highly unstructured grids. The compactness of DG is arguably better suited for today's computer architectures and for complex CFD problems.

We start with outlining the mathematical principles of the DG method; the corresponding numerical discretization will be discussed later. The method of primary interest for fluid dynamics is the weighted-residual method. The general concept is to project the entire problem from an infinite solution space onto a finite solution space. The terms "solution space" and "projection" are central to understanding Galerkin methods. Ideally, one would work with an infinitely large solution space to obtain the complete information about the solution, but that would take infinite computing time. By working with a smaller solution space, we are accepting a certain degree of error in our solution. The mathematical notion for transforming the solution from one solution space to a smaller one is called projection (see Appendix A for more information). The projection operator, a result of the inner product, integrates the product of the solution and the basis functions of the smaller solution space.

In order to facilitate our discussion, the simple 1-D linear diffusion equation is provided as an example,

$$U_t - (DU_x)_x = 0, \tag{1.1}$$

or

$$U_t = DU_{xx}, \tag{1.2}$$

where D is a constant. The weighted-residual formulation is obtained by multiplying the PDE with a test function (or weighting function) v and integrating over the entire physical domain $\mathcal{D}$,

$$\int_{\mathcal{D}} vU_t \, dx = D \int_{\mathcal{D}} vU_{xx} \, dx. \tag{1.3}$$

This operation is also known as the inner product of v and U_t. The intent is to do this with a whole space of test functions. We introduce the residual R, of the governing equation as

$$\int_{\mathcal{D}} vR \, dx = \int_{\mathcal{D}} vU_t \, dx - D \int_{\mathcal{D}} vU_{xx} \, dx = 0. \tag{1.4}$$

By enforcing the residual to be equal to zero, this weighted-residual formulation provides the necessary equations to solve for the unknowns. Let us now step down from the full solution U to an approximate solution u. First we focus on the definition of the solution space of v and u. The solution space is described by the order of the polynomial p, and a solution space capable of representing polynomials up to degree p is mathematically denoted as $\mathcal{P}^p$, where $\mathcal{P}$ is polynomial space. The solution space is spanned by basis functions. One can view basis functions as building blocks of a solution space, where any polynomial of order P can be constructed by a linear combination of basis functions. If the test function v belongs to the same solution space as u, then we have the Galerkin method.

The concept of test-function space and solution space may seem daunting at first, but the concept is elegantly simple and powerful. Let u be a function of x. If we integrate the product of u with a zeroth-degree test function, $v(x) = 1$, over the domain $\mathcal{D}$, we will extract the average of u. Similarly, if we take the inner product of u with a first-degree test function, $v(x) = x$, we will extract the average first derivative. If we repeat this process for higher and higher-order test functions, we will be able to extract higher-degree average derivatives from u. Repeating this process infinitely many times will allow us to know everything about u and the PDE. However, with limited computational resources, we are forced to work in a finite solution space. Thanks to Galerkin's weak formulation, we can solve the PDE within the solution space and ignore the high-order information that lies in the complement space. The complement space is strictly orthogonal to the solution space, and hence conveniently it contains the error in the solution. The official weak statement reads: "Let u be a solution in W, and v be a test function in V, then u must satisfy the partial differential equation tested with v." Again, if $W = V$, we have the Galerkin method. We are now ready to introduce the discontinuous Galerkin discretization.

Discretization in CFD means dividing up the physical domain D into smaller non-overlapping elements Ω_j, such that $D = \sum_j \Omega_j$. Each Ω_j contains a solution space valid within the element boundary only, or simply called a solution with compact support. In this example, let u_j be the discretized solution of degree p within Ω_j, and $u_j = 0$ outside of Ω_j. In the continuous Galerkin (CG) method, the solution must be continuous across an element interface, while in DG, the solution allows for a discontinuity across the element interface. We emphasize the solution within each element of DG is continuous, it only jumps across the interfaces. Figure 1.2 shows two types of piecewise-linear solution representation in 1-D.

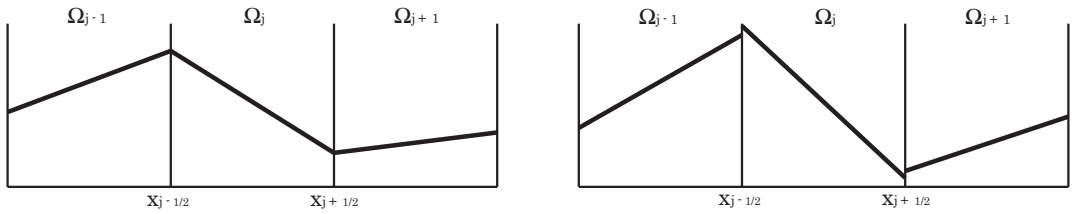

Figure 1.2: Continuous Galerkin (left) and discontinuous Galerkin (right) solution representations showing the key difference is the requirement of continuity across element interface

The concepts of solution space and weak formulation are now applied to each individual element,

$$\int_D v u_t \, dx = \sum_j \int_{\Omega_j} v_j u_{j,t} \, dx = \sum_j D \int_{\Omega_j} v_j u_{j,xx} \, dx. \tag{1.5}$$

We obtain the global solution over $\mathcal{D}$ by locally solving for u_j in every element. Notice the test function, v_j, can now be locally defined inside each cell,

$$v_j = \begin{cases} v & x \in \Omega_j \\ 0 & x \notin \Omega_j \end{cases} . \tag{1.6}$$

We can also say v_j has compact support. Note that this doesn't mean v_j is undefined outside the cell; it is just strictly zero beyond the cell boundary. DG is a truly compact method where each element interacts only with its immediate neighbors. This interaction occurs through the flux, when conserved quantities flow across the element interface. The flux between two elements provides the necessary element coupling, and is central to tying the local solutions together to form one global solution. In the PDE 1.1 the flux equals $-DU_x$. Let us focus on the weak equation of one cell only, and apply integration by parts once to introduce element coupling at the element

interface,

$$\int_{\Omega_j} v_j u_{j,t}\, dx = D\left[v_j u_{j,x}\right]_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} - D\int_{\Omega_j} v_{j,x} u_{j,x}\, dx, \tag{1.7}$$

where the notation $[\cdot]_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}}$ represents a difference across the element from $x_{j+\frac{1}{2}}$ to $x_{j-\frac{1}{2}}$ (see Figure 1.2). However, in the DG method neither the solution nor its derivatives are well-defined at the element interface. If one takes the derivative value within the element, there will be no element coupling. Hence we need to replace the non-unique solution at the element interface with a unique solution, $\hat{u}_{(j,j+1)} = \hat{u}\left(u_j, u_{j+1}\right)$. The flux value at the element interface is always a function of the two solutions sharing that interface, hence the final discretized equation reads,

$$\int_{\Omega_j} v u_{j,t}\, dx = D\left(v_j\left(x_{j+\frac{1}{2}}\right)\hat{u}_{(j,j+1),x} - v_j\left(x_{j-\frac{1}{2}}\right)\hat{u}_{(j-1,j),x}\right) - D\int_{\Omega_j} v_{j,x} u_{j,x}\, dx, \tag{1.8}$$

where $v_j\left(x_{j\pm\frac{1}{2}}\right)$ is evaluated inside Ω_j. Deriving a good numerical-flux algorithm is not an easy task, and is the subject of extensive research. In fact, the first major subject of this thesis, RDG, is a very recent (2005) addition to the family of DG diffusion fluxes. Furthermore, Eqn 1.8 is only a semi-discretization; it requires a matching time-marching method. The second large subject of this thesis, HH-DG, deals with this aspect of DG. We continue with a short history of DG for advection and diffusion, highlighting both spatial discretizations and marching in time.

1.3 History of advection and explicit time-marching methods for DG

The DG approach was originally developed for the steady neutron-transport equations by Reed and Hill (1973) [32]; their formulation was impressively general, being based on a structured triangular grid and going up to $p = 6$. Immediately, LeSaint and Raviart [22] proved that the order of accuracy of steady 1-D DG advection solutions is $2p + 1$; this order is found even for unsteady Euler solutions (see Section 6.2.2). Later, Johnson and Pitkaranta [21] showed the order of accuracy on general triangular elements is $p + \frac{1}{2}$. Independently, Van Leer (1977) [37] introduced a time-accurate DG method for scalar advection, replicating the exact shift operator. Since then, few attempts were made to improve upon the time marching aspect of the DG machinery.

The procedure to extend the order of spatial discretization in a DG method is simple and elegant, however, a matching temporal discretization for arbitrarily high order proved to be far more difficult, as evident by the lack of progress in the following decade.

Eventually, Cockburn and Shu (1989) [7] extended the spatial DG discretization method to fluid dynamic equations by coupling it with Runge-Kutta time-marching, yielding the RK-DG scheme. Cockburn and Shu's paper sparked a renewed interest in using DG for CFD. Around this time, analysis on DG stability and accuracy started to blossom and DG quickly matured. It is not uncommon to see newer DG methods achieve an order of accuracy beyond $2p + 2$, e.g. the RDG and cRDG methods.

While RK-DG dominated the stage regarding time-marching for 15 years, the limitation of RK time marching to 5th-order of accuracy [16] was felt as the aerospace industry expressed its desire for high-order codes. A new class of time-marching schemes based on Taylor-expansions in time was developed; these include the "arbitrary order using derivatives" DG method (ADER-DG) [10], space-time-expansion DG (STE-DG) [15], the upwind moment scheme (UMS) [17] and the Hancock-Huynh discontinuous Galerkin (HH-DG) method. Extension to high order for these methods is automatic and requires less memory than traditional RK-DG methods; however, an adverse effect on the CFL number for ADER-DG and STE-DG may render them slower than RK-DG. Only UMS and HH-DG improve the CFL-number range over RK-DG, which renders them potential candidates for replacing these popular methods. We describe the works of the authors mentioned above in chronological order.

1.3.1 The original DG: neutron transport

In 1973 Reed and Hill [32] demonstrated the first successful use of a DG spatial discretization for steady-state neutron transport problems on 2-D orthogonal triangular meshes. In their landmark paper they argued the superiority of discontinuous Galerkin over the commonly used continuous Galerkin (CG) method. The difference between the two methods lies in the continuity requirement across element interfaces. The CG method enforces the C^0 condition, meaning that the function value must be continuous while the derivatives are allowed to jump across the interface.

Reed and Hill showed a p-th order DG method achieves the same error level (or lower) than a $(p + 1)$-st order CG method. The p-th order DG method required slightly more CPU time than its p-th order continuous counterpart on a 200-element grid; however, the DG method appeared faster on the 800-element grid. The issue

of superiority of DG over CG was not clear until Reed and Hill tested problems containing optically thick regions. Schemes based on continuous solutions were well known to produce flux oscillations, however, DG quickly damped the oscillations toward the finite medium solution and exhibited remarkable stability in comparison to CG. The phenomenal spatial resolution of DG was clearly demonstrated by Reed and Hill; however, it would be 16 years before DG was fully ready for time-accurate problems.

1.3.2 Van Leer's schemes III & VI

Van Leer pioneered the concept of an exact shift operator in 1977 [37] for DG. He replaced the initial-value solution per mesh with an approximate solution and convected the resulting distribution exactly for the scalar advection equation (named "convection" at that time). He constructed various schemes that allow for discontinuity of the solution; in particular, his scheme III and VI are really piecewise-linear and piecewise-quadratic DG schemes. Independent of Reed and Hill's development, Van Leer recognized the need for separate update equations for the higher moments at the cost of extra computational resources. The resulting advection method is fully explicit and time-accurate. A Fourier-analysis showed the order of accuracy of scheme III to be three. Van Leer did not analyze scheme VI due to its sheer complexity and also thought it unattractive because of its high storage requirement, especially in multi-dimensions, an issue that seems trivial for today's computer, but forbidding to the computers of the 1970's. Although the paper's title mentions difference schemes, schemes III and VI were the first time-accurate DG methods and could be easily extended to arbitrarily high order. When trying to extend his schemes to a nonlinear system of conservation laws, Van Leer ran into problems with the time integration and he abandoned the project. Over the next one-and-a-half decade, various research efforts focused on extending RK-DG's time accuracy to orders of five and beyond, and increasing the allowable CFL number. Interestingly, Van Leer's III & VI schemes were reinvented in the atmospheric sciences and used for species transport [31, 34]. But it took 27 years before a correct extension of scheme III to the Euler equations was presented (UMS).

1.3.3 Runge-Kutta methods

Runge-Kutta methods are an important family of explicit and implicit multi-stage methods for approximating the solution of ordinary differential equations. The method

was developed by German mathematicians C. Runge and M.W. Kutta around 1900 [47]. Jameson, Schmidt and Turkel [20] incorporated the 4-stage Runge-Kutta (RK) method in their finite-volume code for the Euler equations in 1981. The method was a huge improvement over existing multi-stage and other finite-difference Euler methods due to the sharpness of captured shocks, and the explicit convergence-acceleration techniques included. RK is an extremely user-friendly time discretization; anybody can pick up an ordinary-differential-equations textbook and implement RK methods without the need to know the details. The same spatial operator is applied at each stage. Stages can be added or modified to increase order of accuracy in time, or to enlarge the stability domain for achieving a higher CFL number.

The RK method was ported into the DG framework by Cockburn and Shu [7] for advection in 1989. The RK temporal discretization and the DG spatial discretization work very well together, since both discretization are compact and can be formally of high order. Although it appears that an n-stage RK method only uses information from immediate neighbors, the actual stencil is much larger since each immediate neighbor also borrows information from neighboring elements. The end result is a pyramid structure in space and time involving $2n + 1$ elements in 1-D. Nevertheless, the coupling of RK and DG results in a stable and sharp shock-capturing method called RK-DG. The authors prove RK-DG to be total-variation-bounded in the means (TVBM), which implies the method allows new local extrema to occur within the element, but not in the mean.

1.3.4 ADER and STE-DG methods

A group of German scientists focused exclusively on developing low-storage time-marching schemes for DG. The storage requirement for RK-DG becomes severe as the order of the solution polynomial increases, and as the number of stages of the RK method increases. Besides the memory issue with RK-DG, the extension to very high order [37] is extremely cumbersome. Followers of RK-DG must refer to lengthy tables to implement each stage of an RK-DG method. Dumbser and Munz [10] ported the ADER-FV (Arbitrary order using derivatives for finite volume) method into DG and knighted the scheme as ADER-DG. The key component of ADER is to Taylor-expand the solution in both space and time, and then convert the time and cross-derivatives into spatial derivatives with the Cauchy-Kovalevskaya or Lax-Wendroff procedure. With the solution expressed in terms of both space and time variables, the governing equation is then integrated in both space and time analytically. The Taylor expansion

in space and time to any order is fully automated, hence ADER-DG fits nicely into the DG machinery.

In 2007, Lörcher, Gassner and Munz [24, 15] introduced space-time expansion DG (STE-DG) as a modification of ADER-DG to account for grids with a large variation in element size. The analytic integration in time is now replaced with Gaussian quadrature to allow for flexibility in time-stepping. The flux values are obtained from the Taylor expansion of the solutions on both sides of the element interface, and a Riemann solver is used at the designated temporal Gaussian points. Both methods are suited for time-step adaptation, i.e, using multiple small time steps in small cells to catch up with the marching in large cells. Logic gates must be implemented to keep track of the order in which elements are updated. Despite the slight overhead in logic, the reduction in CPU time on multi-scale grids is significant. Such adaptive time-stepping does not take away local conservation from the scheme, while STE-DG is time-accurate to the order $p+1$. The memory requirement to store all the flux values at the Gaussian points in time is significantly less than for RK-DG. Despite all the improvements over RK-DG, ADER-DG and STE-DG fail to address the fundamental issue of stability, that is, the pursuit of the maximum time step. In fact, both ADER-DG and STE-DG have a much tighter CFL limit than RK-DG, rendering these space-time methods highly inefficient on structured grids.

1.3.5 Huynh's upwind moment method

In the same year STE-DG was introduced, Huynh introduced the Upwind Moment scheme (UMS) for conservation laws [17]. Huynh's scheme is based on Van Leer's Scheme III [37] for advection, which applies the exact shift operator to the initial values. Huynh extended Van Leer's Scheme III to the Euler equations for piecewise-linear solutions by making Hancock's predictor-corrector method, previously used only with interpolated subcell gradients, suited for DG; the key lies in treating the space-time volume integral accurately. The solution is marched to the half-time level by a subcell Taylor-series expansion, after which the time-centered flux is calculated with upwind data or, in general, a Riemann solver. UMS is extremely simple, 3rd-order accurate, and realizes exact one-mesh-translation for linear advection at a CFL number of unity. Suzuki [36] successfully adapted UMS to hyperbolic-relaxation systems in 2008.

1.3.6 Hancock discontinuous Galerkin method

In 2009, we extended Huynh's UMS method to arbitrary order and also incorporated the recovery procedure (RDG) for diffusion. The method is named Hancock-Huynh discontinuous Galerkin (HH-DG); UMS is included in it. The design of HH-DG is guided by Hancock's observation (1980): the solution can be advanced within a cell with any complete set of flow equations; it is the interface flux that must describe the interaction of adjacent cells. This rule is also observed in ADER-DG and STE-DG. Huynh added to this that in DG the space-time volume integral must also be affected by the cell interactions. This influence is missing in ADER and STE-DG. HH-DG has demonstrated remarkable reliability in dealing with the Euler equations and with the scalar diffusion equation; the order of accuracy is $2p + 1$ for time-accurate Euler in 1-D and $2p + 2$ for time-accurate 1-D diffusion. Like UMS, HH-DG reduces to the exact shift operator for the scalar advection equation. Even in the nonlinear advection case, HH-DG remains stable with a CFL number of unity. This is by far the fastest explicit time-marching DG scheme in development; our current research is focused on increasing the speed of HH-DG for diffusion.

1.4 History of diffusion methods for DG

DG combines the local discontinuous solution representation of a finite-volume scheme with the compactness and versatility of a finite-element scheme. DG captures discontinuous hyperbolic phenomena such as contact discontinuities, slip surfaces and shock waves naturally with its discontinuous basis functions; however, discontinuous basis functions are not the natural way to handle the diffusion operator. Development in DG methods for the diffusion operator became successful starting with Arnold (1982) [2], who introduced a penalty term to penalize discontinuities at the cell interfaces. Much later Baumann (1998) [28] designed a stable DG method without that penalty term, but the enthusiasm for this method died out soon. Bassi and Rebay (1997)[4] and Cockburn and Shu (1998) [8] introduced the highly parallelizable BR2 and local discontinuous Galerkin (LDG) methods, respectively, for the advection-diffusion equation, and were able to reach high orders of accuracy. Arnold [3] showed that all diffusion schemes for DG (as of 2002) could be expressed in terms of a sequence of bilinear terms with scheme-specific coefficients.

In 2005, Van Leer and Nomura [44] introduced a discontinuous Galerkin method based on the recovery concept (RDG). RDG was completely different from any pre-

existing methods, conceptually simpler and extremely high-order. Gassner, Lörcher and Munz [14] (2006) developed a diffusion flux that derives from the exact solution of the generalized diffusive Riemann problem; this approach loses consistency when reducing the time-step without refining spatially. Recently, Huynh [18, 19] developed a family of diffusion schemes which incorporate LDG, BR2 and RDG. His technique of using different correction functions allows him to develop and experiment with new diffusion schemes. Owing to his detailed analysis and numerical experiments, we are able to compare different diffusion methods in a head-to-head manner. The following sections detail in chronological order the major developments in diffusion for DG.

1.4.1 The (σ, μ)-family

In the early days, DG had ridden on the success of adopting Riemann solvers for the flux computation from finite-volume methods for hyperbolic equations. However, when it came to parabolic and elliptic equations, the DG community was at a loss. One of the first attempts to solve parabolic and elliptic equations with DG is credited to a venerable class of schemes called the (σ, μ)-family:

$$\begin{aligned}\int_{\Omega_j} v u_t \, dx &= -D\left(< u_x > [v] \mid_{j+\frac{1}{2}} + < u_x > [v] \mid_{j-\frac{1}{2}}\right) - D\int_{\Omega_j} v_x u_x \, dx \\ &+ \sigma D\left(< v_x > [u] \mid_{j+\frac{1}{2}} + < v_x > [u] \mid_{j-\frac{1}{2}}\right) \\ &- \frac{\mu D}{\Delta x}\left([v][u] \mid_{j+\frac{1}{2}} - [v][u] \mid_{j-\frac{1}{2}}\right). \end{aligned} \tag{1.9}$$

Van Leer et al. [44, 42] provided a simple interpretation of the (σ, μ)-family for scalar diffusion; the notation is explained in Chapter 3. The first line of the equation is the original DG equation for the time-accurate diffusion equation. In 1979, Delves and Hall [9] added the σ bilinear term in the second line of the equation and set $\sigma = -1$ (later known as the symmetric scheme). The symmetric scheme is not stable for parabolic problems, hence it is only used to solve elliptic steady-state problems with an implicit solver.

Arnold [2] introduced the internal-penalty method (IPM) in 1982 by adding the bilinear μ-term in the third line of the equation. The IPM sets $(\sigma, \mu) = (-1, 1)$ and successfully solves time-accurate diffusion problems. The reason for adding the μ-term is to penalize the discontinuity of the DG solution. This is a rather ironic name because it is the same discontinuity that brings fame and honor to DG for advection

problems. These bilinear terms are added to the diffusion equation to increase the chances of coming up with a stable method, while consistency with the governing equation seems to be secondary. Van Leer and Nomura pointed out that for $p = 0$ the (σ, μ)-family is only consistent with the diffusion equation owing to Arnold's penalty term and only if $\mu = 1$. In retrospect, Arnold did accurately determine the need for an additional bilinear term. In 1997, Baumann and Oden [28] presented the choice of $(\sigma, \mu) = (1, 0)$, representing a complete sign change from the symmetric scheme and abandonment of Arnold's term. The freedom to switch to totally different values of (σ, μ) over the course of two decades clearly indicates there existed no good guiding principle for choosing the right set of coefficients. In the early stages of our research in RDG, we were able to find an inkling of a relationship between RDG and the (σ, μ)-family for $p = 1$; this is further extended in Section 3.3.5. The progress in diffusion for DG was painfully slow until the beginning of the 21st century. The next set of methods marks the first departure from the (σ, μ)-family.

1.4.2 Approaches based on a system of first-order equations

Bassi and Rebay [4] (1997) as well as Cockburn and Shu [8] (1998) introduced a different approach to diffusion for DG by writing the 2nd-order diffusion equation as a system of 1st-order equations. For scalar diffusion, the equations are written as,

$$u_t - Dq_x = 0, \tag{1.10}$$

$$q - u_x = 0, \tag{1.11}$$

where D is the diffusion coefficient, and q is the additional variable. The DG formulation is obtained by testing both of these equations and integrating over an element. The first equation requires the flux q, which is provided by the next equation. However, the flux of the second equation is determined from the derivative of the discontinuous solution u. Once again the discontinuity presents a problem of non-uniqueness at element interfaces, and numerical techniques must be applied to approximate u_x in the 2nd equation.

The papers listed above can be described as a mathematical festival of finite-element terminologies and notations for multi-dimensional Navier-Stokes equations. If you prefer to understand these schemes over a cup of tea on a good Sunday afternoon, you will appreciate that Huynh [18, 19] analyzed and presented these schemes in a simple unifying framework for scalar 1-D linear diffusion. In order to fit all ex-

isting schemes into a unifying framework, Huynh used integration by parts twice on the scalar diffusion equation and applied different correction functions to calculate the common function value and derivative value across the element interface. His interpretation of the Bassi-Rebay schemes for diffusion, also known as BR1 and BR2, is shown in Figure 1.3.

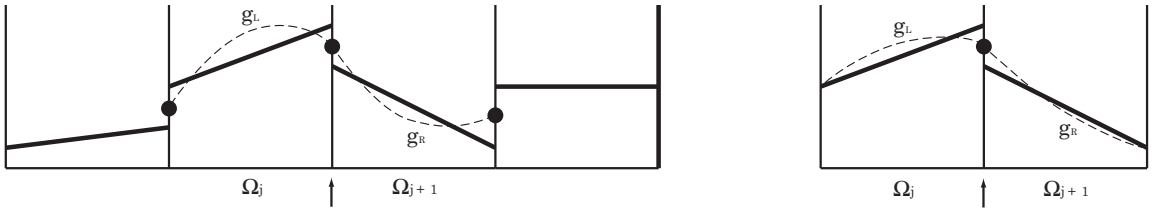

Figure 1.3: Interpretation of BR1 (left) and BR2 (right) for scalar 1-D diffusion. Two new local solutions are reconstructed on the left and right of the indicated interface as g_L and g_R respectively. The large solid dot indicates the average of the left and right values at the interface. The schemes also adopt the average of the derivative values as the interface value of u_x. Notice BR1 utilizes a 4-cell stencil for an interface flux, while the newer BR2 utilizes a 2-cell stencil.

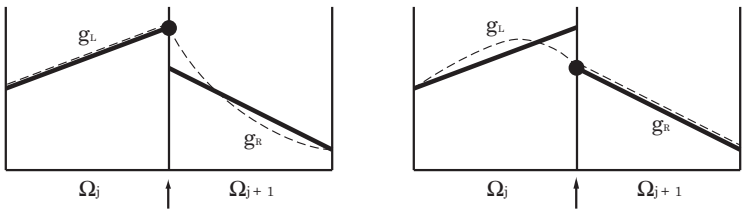

Figure 1.4: Interpretation of two versions of LDG for scalar 1-D diffusion. Two new local solutions are reconstructed on the left and right of the indicated interface as g_L and g_R respectively. The large solid dot indicates the common function value. The LDG is a non-symmetric scheme using the function value of g_L for u, but the derivative value of g_R for u_x, (left) or vice versa (right).

For each interface, two functions g_L and g_R of order $p+1$ are reconstructed by sharing most moments with the original solution in their respective elements. In BR1, the functions g in each element must satisfy Dirichlet conditions at both left and right interfaces. The Dirichlet condition is simply to satisfy the average solution value across the interface, which leads to a clumsy and costly 4-cell stencil. Bassi and Rebay introduced the simpler BR2 version in which each g only needs to satisfy one cell-coupling Dirichlet boundary condition and one local Dirichlet boundary condition. The g functions in both BR1 and BR2 take the same function value at the interface,

but not the same derivative value. Bassi and Rebay then used the arithmetic mean of $\frac{\partial g_L}{\partial x}$ and $\frac{\partial g_R}{\partial x}$ to approximate u_x. Cockburn and Shu's Local DG (LDG) scheme for scalar diffusion is shown in Figure 1.4. The pair of g functions share most moments with the solution in their respective elements. One of the intervals provides the solution value at the interface; the g function in the other interval then provides the desired solution derivative. Thus, there are two variants of the 1-D LDG scheme. As Huynh indicated, Bassi-Rebay and Cockburn-Shu have different interpretations of the "common" function value and function derivative value. Huynh studied several other choices to be detailed later.

In 2008, Peraire and Persson [30] reduced the storage requirement and stencil size of LDG and called their new scheme Compact DG (CDG). The paper compared BR2, LDG and CDG and showed LDG and CDG have roughly the same spectral radius, while BR2's spectral radius is about 1.5 times greater, meaning a narrower stability range for the time step. Although CDG is exactly the same as LDG in 1-D, the storage requirement is dramatically reduced for higher spatial dimensions.

1.4.3 Recovery-based discontinuous Galerkin method

RDG was introduced in 2005 by Van Leer and Nomura [44], and subsequently expanded in [43, 41]. Central to RDG is the concept of recovery between two elements where the solution is discontinuous. RDG recovers a smooth function in the union of the two elements that is indistinguishable from the original solutions in the weak sense. RDG demonstrated a remarkable order of accuracy for scalar diffusion equation on structured grids. Our current research effort focuses on adapting RDG to nonlinear diffusion problems on unstructured grids. A full treatment to RDG begins in Chapter 2.

1.4.4 Flux-based diffusion approach

In 2006, Gassner, Lörcher and Munz [14] introduced a numerical diffusion flux based on the exact solution of the diffusion equation at the discontinuity of a piecewise continuous solution. The concept is similar to that of Godunov-type finite-volume methods for hyperbolic equations. The diffusive generalized Riemann problem (dGRP) can be solved for any order of the solution space. The flux is obtained from the general bounded solution of the discontinuous initial-value problem for the diffusion equation and depends on both space and time. The resulting dGRP-DG is a space-time method for diffusion. However, the dGRP flux is only consistent if $\Delta x \propto \Delta t$. The

scheme uses the time integral of the diffusion flux for the update; the time average of the flux always contains a term proportional to $1/\sqrt{\Delta t}$. For $\Delta t \to 0$, while keeping Δx fixed, the flux blows up, rendering the scheme sensitive to small time steps; the flux is unsuited for combination with multi-stage time integration. The dGRP-DG scheme performs relatively fast in steady-steady problems, as the stability limit on Δt is proportional to Δx only instead of the usual $(\Delta x)^2$ for explicit methods. In addition, Gassner et al. were able to express the dGRP-DG scheme in bilinear terms, and discovered dGRP-DG shared many terms with the (σ, μ)-family. The scheme has a μ coefficient different from the one in the symmetric interior-penalty method, and contains an additional bilinear term of the form

$$< v_x > [u_x] \mid_{j+\frac{1}{2}} - < v_x > [u_x] \mid_{j-\frac{1}{2}} . \tag{1.12}$$

Gassner et al. observed that the presence of the new higher-order bilinear term increases the stability limit, however, the increase in stability limit decreases for higher p. This is an important observation pertinent to the recovery-based discontinuous Galerkin (RDG) method, as we shall show in a later chapter that the RDG method automatically generates these higher-order bilinear terms.

1.4.5 Huynh's reconstruction approach

Huynh studied a family of diffusion schemes using his reconstruction approach [18, 19]. The concept is to reconstruct correction functions to correct for the discontinuity of the solution across element interface. One of Huynh's primary motivations is to come up with a scheme that replicates the success in accuracy and stability of RDG, while reducing the computational cost. To this end, Huynh limited his correction functions to one element only, and coupled the correction functions at the interface, whereas in RDG, the single recovery function spans two elements and naturally couples the elements. Huynh studied 16 different ways to couple the correction functions sharing the same interface. One of the most promising new methods is, "I-continuous - g_{Le}/ SP - g_{DG}", or aptly called "Poor-man's recovery." Instead of defining a common function value and common derivative, the Poor-man's method naturally obtains these values by solving a small system that couples the left and right correction functions. The method is computationally cheap and achieves a higher order of accuracy than existing BR2 and LDG methods. While comparison with RDG still demonstrates RDG's superiority in accuracy and stability, it is worthwhile to further study how these schemes compare to each other in terms of CPU time in a real time-accurate

calculation.

1.4.6 Combining DG advection with DG diffusion

If one has satisfactory DG methods for discretizing the diffusion and advection operators, it is a rather trivial task to combine the two in a DG method for advection-diffusion. Because of the fundamental differences between these two operators and their discretizations, such a blended advection-diffusion method does not look attractive. The advection and diffusion discretizations are not consistent in their interpretation of the numerical solution. For example, combining the upwind-biased discretization, which fully accepts the discontinuous nature of the solution, with a (σ, μ)-type discretization of the diffusion term, which penalizes the solution for being discontinuous, seems to reveal an inner conflict.

A more ambitious approach would attempt to unify the solution representations used for the different operators. Traditionally, a pure diffusion operator is preferably discretized with a continuous DG method, especially a nodal-point-based one where it is trivial to guarantee that the solution representation is continuously differentiable. However, if one insists on a $C^{(1)}$ representation of the solution for the sake of diffusion, while linked to a primary discontinuous basis for advection, the relation between the two ceases to be local. It leads to a global system of equations where the smooth solution representation is based on the interface value and derivative of the solution; the equation system will globally link these to the coefficients of the interior basis functions.

A recent example of a method that blends a $C^{(1)}$ interface-based solution to the discontinuous cell-based solution is the HDG method of Cockburn et al. [6, 27]. Here the "H" stands for hybridizable, which appears to refer to the capability of the method to be used along-side a continuous Galerkin method, used perhaps for other equations of a system. This method combines the usual upwind DG discretization for advection with a discretization of the diffusion equation written as a system of first-order equations, similar to LDG (see Section 1.4.2). As explained before, this method requires the solution of a global system; the authors point out, though, that the unknowns are just the interface values, regardless of the order of the method.

In recovery-based DG, we also attempt the unification of the two solution representations; however, it is a local unification. The smooth solutions recovered at the interfaces of one cell overlap each other in the interior of the cell, but are not equal; therefore, no global smooth recovered function exists. Given the high order of

accuracy achieved by RDG methods, it is doubtful that recovering a globally smooth unique solution will bring any advantage. Moreover, it has been shown earlier by Van Raalte and Van Leer [45] that the solution representation of the locally recovered solution in the union of two cells is identical to the discontinuous solution once we switch from the discontinuous basis to the smooth recovery basis; see also our paper [41] on "Unification of discontinuous Galerkin methods for advection and diffusion".

1.5 Motivation

Our motivation directs us towards the ultimate discontinuous Galerkin scheme for conservation laws. As of the year 2010, high-order compact methods such as DG are poised to take over the finite-volume method which still counts as the industrial standard. However, several issues still prevent the industry from moving away from its safe in-house codes; DG is relatively new and has yet to establish a good record of reliability. DG was dogged by its high storage requirement, the lack of a good viscous discretization, lack of error estimation, lack of an efficient high-order time-marching scheme, and lack of a multi-dimensional high-order-compatible limiter. We are pleased to say that these issues are nearing resolution with a high level of confidence. This thesis focuses on resolving the issues of the viscous solver and efficient time-marching for DG.

The design and presentation of current diffusion schemes for DG is overly complex and lacks physical interpretation. Numerical schemes were designed from the standpoint of facilitating mathematical proofs instead of observing physical clarity. Frustration is what drove us to design a new method that is a quantum level better than existing methods. Van Leer and Nomura first introduced the recovery-based DG (RDG) method for diffusion. The recovery concept is elegantly simple and, to their surprise, achieved an order of accuracy beyond traditional superconvergence within a generous stability range. Our first goal is to further pursue the concept of recovery and demonstrate RDG's ability to perform in multi-dimensional nonlinear diffusion problems.

In the field of time-marching schemes for DG, nothing dramatic has risen above the horizon since RK-DG. In fact, DG's high storage requirement drove researchers toward implicit time-marching methods. Implicit time-marching methods naturally damp eigenmodes and may not be time-accurate for all transient phenomena of interest. Our goal is to design a fast, explicit, and high-order time marching scheme for DG. To this end, we present the arbitrarily-high-order Hancock-Huynh discontinuous Galerkin

(HH-DG) method. HH-DG is a true space-time DG method in which spatial elements are strongly coupled to temporal elements via a Cauchy-Kovalevskaya procedure. HH-DG's true beauty comes from being the exact shift operator for linear advection, which is a good starting point for a new scheme; it allows CFL numbers up to unity.

The first half of the thesis is dedicated to introducing the recovery-based discontinuous Galerkin (RDG) method for diffusion. In Chapter 2, we introduce the recovery concept, and the weak formulation of the diffusion equation with recovery, then present a stability proof. The next two chapters detail RDG for 1-D and 2-D, respectively, along with Fourier analyses and comparisons to other existing DG diffusion schemes. Chapter 5 applies RDG to the Navier-Stokes equations and provides numerical results to popular test cases. We introduce the new space-time DG method called Hancock-Huynh discontinuous Galerkin (HH-DG) in Chapter 6, formulated to accommodate both advection and diffusion. In Chapter 7, we summarize our findings and point to a perspective of developments to be expected in the future.

1.6 Outline of this thesis: the River of Recovery

As an outline of this thesis, we offer Figure 1.5, the "River of Recovery." It depicts the main stream of our research as a river flowing from the upper left to the lower right, while broadening and with two cascades representing major obstacles in the progress of the research. The river starts from the humble RDG-2x method for the 1-D linear diffusion equation and ends at the successful extension to the viscous terms of the 2-D Navier-Stokes equations. Along the way we present much Fourier analysis and many failed schemes including those of the RDG-1x type. Numbers in parentheses refer to thesis chapters.

Three side-streams are visible. One represents purely theoretical developments, which do impact the main research as indicated by the merging of the side-stream with the mainstream. Another stream represents the influence of our work on others, in particular the group at Idaho National Laboratory, which developed cell-centered RDG (cRDG). This work eventually fed back into our own research by helping us cross the nonlinearity cascades, as well as the cascades of cross-derivatives. Finally there is the space-time DG stream which has led to arbitrary-order space-time DG methods for advection and diffusion separately, but not yet for advection-diffusion in a satisfactory way.

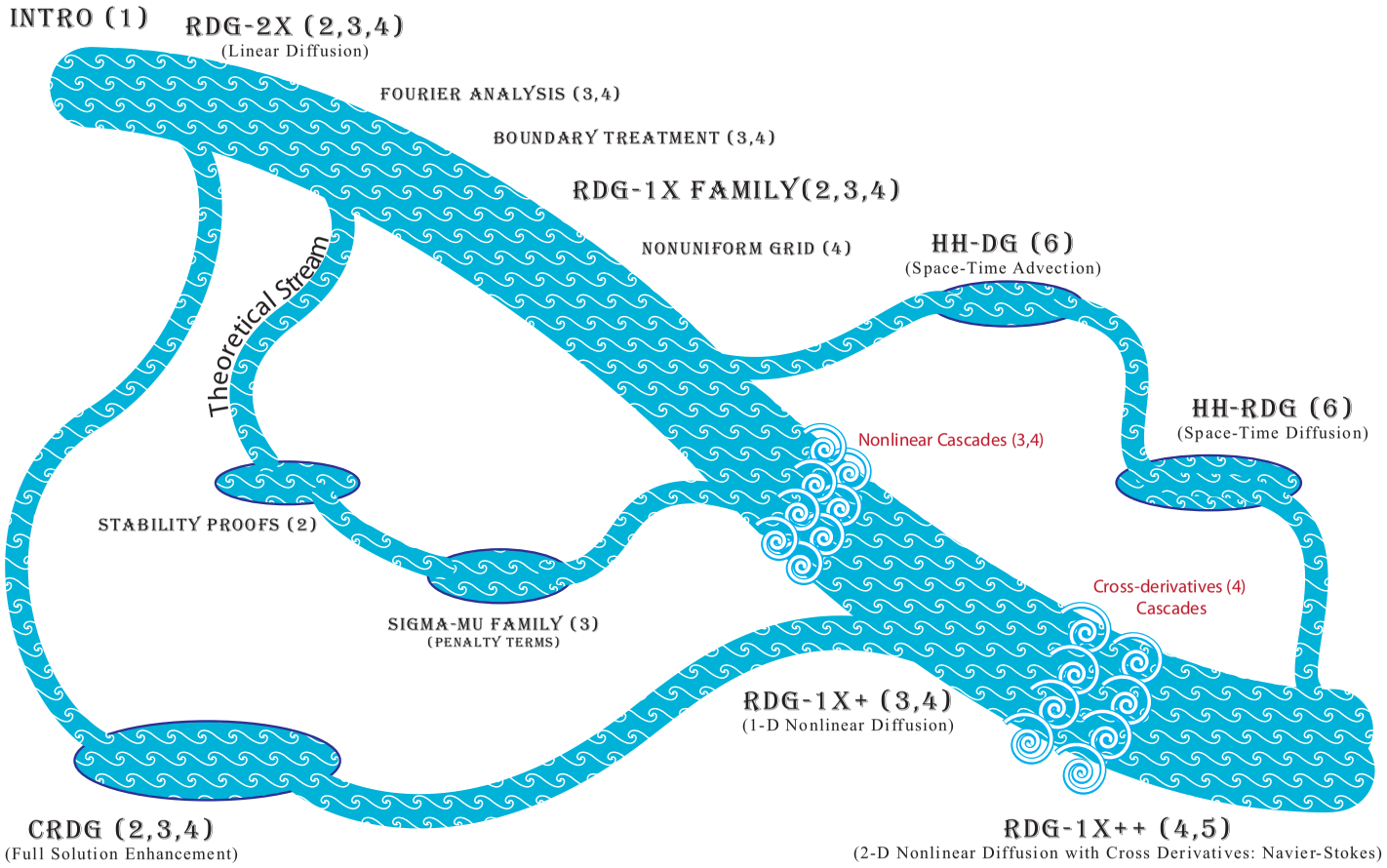

Figure 1.5: The outline of this thesis is represented by the River of Recovery.

CHAPTER II

Interface-centered Recovery-based Method for Diffusion

Recovery-based discontinuous Galerkin (RDG) is due to Van Leer and appeared on the DG stage in 2005, marking the beginning of a new perspective on the diffusion operator for DG. The name "recovery" is attributed to K.W. Morton [26], who used it in the context of "shock recovery," which is the inverse process: recovering a discontinuity from a smoothed profile. The further development of RDG was documented in a train of papers [43, 41, 23, 45]. Unlike previous methods based on facilitating mathematical proofs or choosing the correct values for the stability parameters, RDG is based on a physical-numerical interpretation of the solution. The end product is a universal concept of "recovery" that transcends its original purpose and can be applied to different areas of DG and finite-volume methods. For this reason we sometimes attach the term "interface-based" in front of RDG, iRDG, to indicate we are talking about recovery for diffusion; for most of this thesis, we will simply use RDG and drop the "i". This chapter first describes the motivation behind RDG starting with an interesting quote. We then cover various aspects of recovery concept and present RDG for diffusion. Next, we present stability proofs for various RDG methods for scalar nonlinear diffusion. The chapter ends with a brief talk on other uses of the recovery concept.

2.1 The beginning of RDG

RDG literally began with Van Leer's observation, "What's sauce to the goose is not sauce to the gander." The original English idiom actually implies both genders should be held to the same standards, but the word "not" was added to emphasize the funda-

mental difference between advection and diffusion operators. What works numerically for advection does not work for diffusion, and vice versa.

The DG community garnered much success in numerical advection by borrowing from finite-volume discretizations, notably, the concept of Riemann solvers; however, there was no "Riemann solver" (now there is [14]) or equally successful finite-volume algorithm for diffusion. The DG community was then forced to invent its own diffusion discretizations, resulting in the contrived, sometimes unstable or even inconsistent schemes mentioned in the previous chapter. Van Leer summarized the situation in the cartoon below.

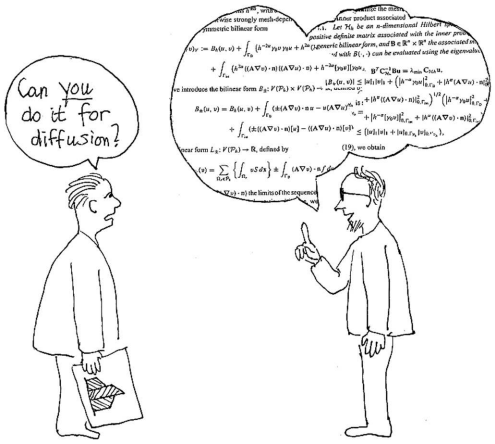

Figure 2.1: Finite-element analyst answering finite-volume analyst regarding DG for diffusion. Courtesy of Van Leer, 2005.

Figure 2.1 alludes to the complexity of the first generation of DG diffusion schemes mentioned in the previous chapter; the answer to the question was cut from the article [5] defining the Baumann-Oden scheme. It became clear that DG needed a fairer way to deal with diffusion than by penalties.

While looking at the diffusion problem for DG, Van Leer noticed that a smooth initial solution projected on to DG basis functions becomes discontinuous. Hence in reality the discontinuous solution is only trying to represent a smooth solution. The solution becomes embarrassingly simple: one only needs to "recover" the smooth solution from the discontinuous solution. In general the recovery of the original is not exact, but makes the most out of the available data. The coming sections illustrate the recovery concept in detail.

2.2 Cartoon illustration of recovery in 1-D

The concept of interface recovery will be played out by a three-member cast: U as the continuous smooth solution, u as the discontinuous solution, and f as the recovered solution. The stage is set in Figure 2.2. The story begins with the exact solution U in the left picture. Smooth and untamed, the exact solution is represented by dotted line spanning from $x = -1$ to 1.

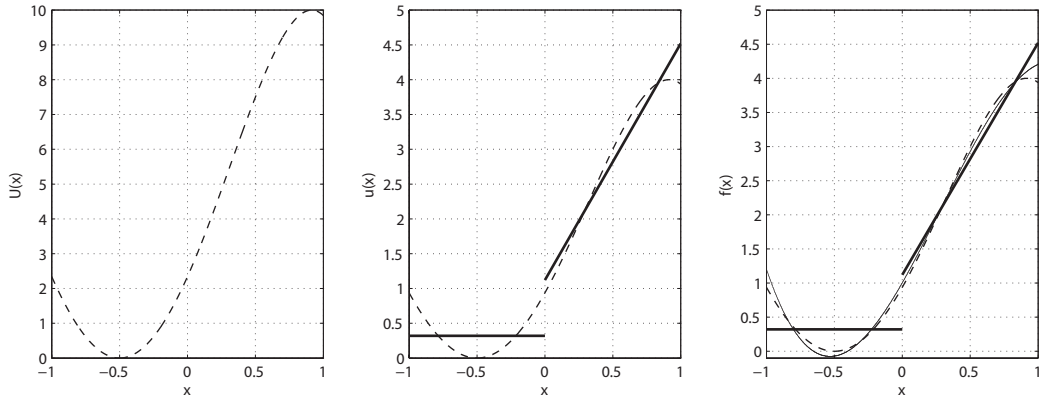

Figure 2.2: The essence of recovery. Start with the exact solution U on the left (dashed line). The discretized solution u in the middle is piecewise linear (thick solid line). We recover a smooth function f (right, thin solid line).

However, one must face the cruel reality that resources are finite; we split the domain at $x = 0$ into two elements and then approximate the original function with u, which is a polynomial projection of U in each element, and discontinuous across the element interface. The loss of innocence occurs when we strip away most of the derivatives of U. The resulting piecewise-linear functions in both elements share the cell average and (least-squares) average gradient with U (see middle figure). We say that u is weakly equivalent to U in both elements. The original solution, which may be non-polynomial, is suddenly reduced to a mere four pieces of data. Furthermore, the discontinuity becomes a source of trouble for the diffusion operator since neither the solution nor its derivatives are uniquely defined at the element interface. Fortunately, we can *recover* a cubic function from the preserved data; the end result is a smooth function f represented by the thin solid line in the right figure. The recovered solution shares the cell average and first derivative of u in both elements; hence we say f is weakly equal to u in both elements. By a transitive relationship, all three members are indistinguishable from each other in the weak sense. The new f spans the union of two elements and provides unique solution and derivative values at the element interface.

This story has a happy ending. The recovery concept presents a fresh way to look at discontinuous solution in the union of two elements; and the example above makes clear why this method is called "recovery". Anyone understanding the recovery principle may now stop reading and develop RDG entirely by himself or herself. Regardless, we shall lay up next the mathematical foundation of recovery.

2.3 Interface recovery equations

We consider the most general framework inside an n-dimensional euclidean space, $\mathbb{R}^n$. Let $\mathcal{D}$, the physical domain of interest, be a subspace in $\mathbb{R}^n$. We introduce two vector spaces: the solution space W and the test space V in $\mathcal{D}$. Let ϕ be a set of smooth functions with compact support satisfying $W = \{\phi \in W : \phi \in P^k(\mathcal{D})\}$ where $\mathcal{P}^p(\mathcal{D})$ is the polynomial space of $\mathcal{D}$ containing polynomials of at most degree p. Similarly let $V = \{v \in V : v \in P^k(\mathcal{D})\}$. The domain is discretized into non-overlapping cells such that $\mathcal{D} = \sum_j \Omega_j$. If the solution space of V is the same as W, then this is a Galerkin method; furthermore, if the basis functions are discontinuous across the element interface, then this is specifically the DG method. Let Ω_j and Ω_{j+1} be two elements sharing the element interface $S_{j,j+1}$, and their solutions be expressed respectively as,

$$u_j = \sum_{i=1}^{N_j} a_{j,i}\,\phi_i, \tag{2.1}$$

$$u_{j+1} = \sum_{i=1}^{N_{j+1}} a_{j+1,i}\,\phi_i, \tag{2.2}$$

where $a_{j,i}$ and $a_{j+1,i}$ are the degrees of freedom and the integers N_j and N_{j+1} are the number of degrees of freedom in Ω_j and Ω_{j+1}, respectively. We introduce the recovery function, $f_{j,j+1}$ centered on $S_{j,j+1}$ as,

$$f_{j,j+1} = \sum_{i=1}^{N_j+N_{j+1}} b_i\,\psi_i, \tag{2.3}$$

where b_i are the unknown coefficients and ψ_i are the recovery basis functions spanning the union of elements Ω_j and Ω_{j+1}. The proper choice of ψ_i is essential for a well-conditioned recovery as will become clear soon. The recovery equations are of the

form,

$$\begin{aligned} \int_{\Omega_j} v_j u_j \, d\Omega &= \int_{\Omega_j} v_j f_{j,j+1} \, d\Omega, \\ \int_{\Omega_{j+1}} v_{j+1} u_{j+1} \, d\Omega &= \int_{\Omega_{j+1}} v_{j+1} f_{j,j+1} \, d\Omega. \end{aligned} \tag{2.4}$$

where v_j and v_{j+1} belong to Ω_j and Ω_{j+1}, respectively, and cycle through all ϕ_i. Notice the recovery equations always form a pair with one equation dedicated to each side of the interface. Inserting Eqns (2.1-2.3) enables us to put the recovery equations into matrix-vector form,

$$\boldsymbol{M}_{j,j+1} \, \vec{a}_{j,j+1} = \boldsymbol{R}_{j,j+1} \, \vec{b}_{j,j+1}, \tag{2.5}$$

where $\boldsymbol{M}_{j,j+1}$ is the block diagonal mass matrix, $\vec{a}_{j,j+1}$ is a vector containing the coefficients $a_{j,i}$ and $a_{j+1,i}$, $\boldsymbol{R}_{j,j+1}$ is simply the matrix on the right hand side, and $\vec{b}_{j,j+1}$ is the vector of unknown to solve for. The detailed expressions for $\boldsymbol{M}_{j,j+1}$ and $\boldsymbol{R}_{j,j+1}$ are

$$\boldsymbol{M}_{j,j+1} = \begin{pmatrix} \begin{pmatrix} < v_0, \phi_0 > & < v_0, \phi_1 > & \cdots \\ < v_1, \phi_0 > & < v_1, \phi_1 > & \\ \vdots & & \ddots \end{pmatrix}_j & \\ & \begin{pmatrix} < v_0, \phi_0 > & < v_0, \phi_1 > & \cdots \\ < v_1, \phi_0 > & < v_1, \phi_1 > & \\ \vdots & & \ddots \end{pmatrix}_{j+1} \end{pmatrix},$$

$$\boldsymbol{R}_{j,j+1} = \begin{pmatrix} \begin{pmatrix} < v_0, \psi_0 > & < v_0, \psi_1 > & < v_0, \psi_2 > & < v_0, \psi_3 > & < v_0, \psi_4 > & \cdots \\ < v_1, \psi_0 > & < v_1, \psi_1 > & < v_1, \psi_2 > & < v_1, \psi_3 > & < v_1, \psi_4 > & \\ \vdots & & & & & \ddots \end{pmatrix}_j \\ \begin{pmatrix} < v_0, \psi_0 > & < v_0, \psi_1 > & < v_0, \psi_2 > & < v_0, \psi_3 > & < v_0, \psi_4 > & \cdots \\ < v_1, \psi_0 > & < v_1, \psi_1 > & < v_1, \psi_2 > & < v_1, \psi_3 > & < v_1, \psi_4 > & \\ \vdots & & & & & \ddots \end{pmatrix}_{j+1} \end{pmatrix},$$

where $< \cdot, \cdot >_j$ denotes the inner product of two functions on element Ω_j. Solving for $\vec{b}$ yields,

$$\vec{b}_{j,j+1} = \boldsymbol{Q}_{j,j+1} \, \vec{a}_{j,j+1}, \tag{2.6}$$

$$\boldsymbol{Q}_{j,j+1} = \boldsymbol{R}^{-1}_{j,j+1} \boldsymbol{M}_{j,j+1}; \tag{2.7}$$

$\boldsymbol{Q}_{j,j+1}$ is called the recovery matrix. The following sections detail how to choose ψ_i to ensure that $\boldsymbol{R}_{j,j+1}$ is fully ranked, and ways to reduce the condition number of $\boldsymbol{R}_{j,j+1}$.

2.3.1 Full-rank $\boldsymbol{R}_{j,j+1}$

Choosing the correct recovery basis functions requires extra attention; perhaps it is useful to think of this as a Tetris game. Tetris is a game where uniquely shaped blocks fall downward due to gravity, and the blocks stack upon each other depending on where the player chooses to drop them. Our "recovery game" follows the same rule. The recovery function is composed of basis functions in the face-normal direction, and in the face-parallel directions. The maximum order of basis functions in each direction is dictated by the number of available data in each direction. Consider the 1-D examples for $p = 0$ and $p = 1$ in the figure below.

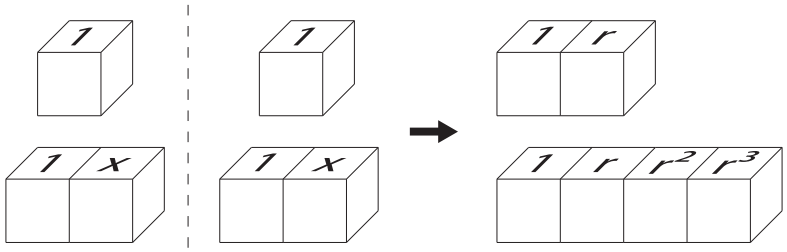

Figure 2.3: A little block game for determining the recovery basis in 1-D. The blocks on the left and right of the dashed line indicate basis functions of the solution in Ω_j and Ω_{j+1} respectively. Now imagine gravity pulls to the left; the blocks from Ω_{j+1} fall on "top" of the blocks in Ω_j to form the recovery basis.

The recovery basis functions are in the face-normal direction, or r-coordinate, and the dashed line indicates the orientation of the element interface. For $p = 0$, only the cell averages are available on each side of the interface; they are represented by the "1" blocks. If gravity pulls leftward, the two single blocks will stack up on each other to form a two-block as shown in the figure to the right. The new two-block represents two available data in the face-normal direction, hence recovery between two $p = 0$ elements results in a linear recovery function, indicated by the basis functions 1 , r. Similarly, for $p = 1$, the two blocks with "1" and "x" represent the cell average and gradient within each element. The recovery between two $p = 1$ elements results in a four-block, or a cubic recovery function. In 2-D, the same strategy is followed, only we now introduce the s-coordinate to account for solution variations in the face-parallel

direction. Consider the 2-D examples below for $p = 1$ and $p = 2$.

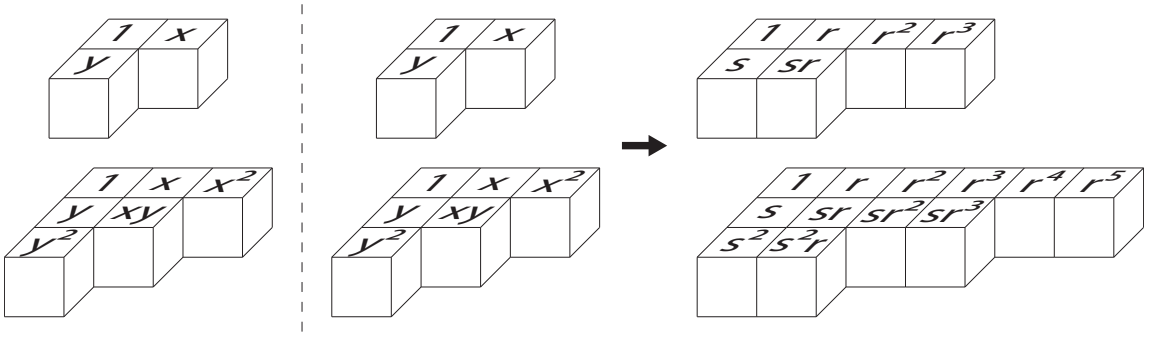

Figure 2.4: A little block game for determining the recovery basis in 2-D. The blocks on the left and right of the dashed line indicate basis function of the solution in Ω_j and Ω_{j+1} respectively. Now imagine gravity pulls to the left; the blocks from Ω_{j+1} fall on "top" of the blocks in Ω_j to form the recovery basis. Notice there are more blocks in the r-coordinate than in the s-coordinate.

Notice the recovery function includes twice as many basis functions in the r-coordinate than in the s-coordinate in 2-D. This raises the question whether the lower-order s-components of the recovery function will contaminate the high-order r-components. The 2-D numerical results in Chapter 4 will provide a nuanced answer; the details for triangular structured and unstructured grids will be spelled out later.

The concept of this block game may be childish, but plays an important role in determining the rank of $\boldsymbol{R}_{j,j+1}$. In the 2-D example for $p = 1$, if we naively attempt to recover a complete quadratic recovery function from the six blocks in Figure 2.4, $\boldsymbol{R}_{j,j+1}$ will be rank-insufficient, and the recovery process fails. The choice of recovery basis is not ad-hoc, but follows an "information conservation" principle, where each block in this game represents a piece of information the solution has in a certain direction. If one can play with blocks, then one can perform recovery; the 3-D game goes exactly in the same way. Achieving a full-rank matrix is the first step; the next step is to reduce the condition number of $\boldsymbol{R}_{j,j+1}$.

2.3.2 Reducing the condition number of $\boldsymbol{R}_{j,j+1}$

The reduction of the condition number of $\boldsymbol{R}_{j,j+1}$ is central to reducing the magnitude of error and danger of instability of RDG. The conditioning number is the ratio between the largest and smallest eigenvalues of the system; consequently, the lowest achievable condition number is unity. There are two major ways to reduce the condition number. The first is the universal method of using a preconditioner to solve a

system of equations. Instead of solving the system in Eqn 2.5, we solve

$$\boldsymbol{P}_{j,j+1}\boldsymbol{M}_{j,j+1}\,\vec{a}_{j,j+1} = \boldsymbol{P}_{j,j+1}\boldsymbol{R}_{j,j+1}\,\vec{b}_{j,j+1}, \tag{2.8}$$

where $\boldsymbol{P}_{j,j+1}$ is the preconditioning matrix. We enforce the following condition,

$$\mathrm{Cond}\left(\boldsymbol{R}_{j,j+1}\right) \geq \mathrm{Cond}\left(\boldsymbol{P}_{j,j+1}\boldsymbol{R}_{j,j+1}\right), \tag{2.9}$$

where Cond$(\cdot)$ stands for condition number, and invert $\boldsymbol{P}_{j,j+1}\boldsymbol{R}_{j,j+1}$ instead of just $\boldsymbol{R}_{j,j+1}$. The error in largest eigenvalue will swamp the smaller eigenvalues and mar the results. The higher the condition number, the more contamination occurs. It is an important task for the preconditioning matrix to bring all the eigenvalues of the system to the same relative order. A simple Jacobi-preconditioner matrix, which normalizes the L_2-norm of each column of $\boldsymbol{R}_{j,j+1}$ to unity, works well for RDG:

$$\boldsymbol{P}_{j,j+1} = \begin{pmatrix} \frac{1}{\|\boldsymbol{R}_{j,j+1}(:,1)\|_2} & & & & \\ & \frac{1}{\|\boldsymbol{R}_{j,j+1}(:,2)\|_2} & & & \\ & & \frac{1}{\|\boldsymbol{R}_{j,j+1}(:,3)\|_2} & & \\ & & & \ddots & \\ & & & & \ddots \end{pmatrix}, \tag{2.10}$$

$$\left\|\boldsymbol{R}_{j,j+1}\left(:,k\right)\right\|_2 = \frac{1}{N_j + N_{j+1}}\sqrt{\sum_{i}^{i\leq N_j+N_{j+1}} \left(\boldsymbol{R}_{j,j+1}\left(i,k\right)\right)^2}. \tag{2.11}$$

$\boldsymbol{R}_{j,j+1}\left(:,k\right)$ is the Matlab notation for the k-column of $\boldsymbol{R}_{j,j+1}$. There exist other types of preconditioners that are more effective; however, those are beyond the scope of this thesis.

The second method is the use of orthogonal basis functions for both ϕ and ψ in Eqns (2.1-2.4). RDG works with any kind of basis functions, however, much reduction in the condition number comes from the smart choice of basis functions. In analytical mathematics, all basis functions spanning the polynomial space $\mathcal{P}$ are equal, but in numerical mathematics, some basis functions are more equal than the others. So far you have seen the monomial basis functions, r^k, where r is a variable and k is an integer greater or equal to zero. The choice of monomials for recovery results in the Vandermonde matrix which is extremely ill-conditioned. Although the monomial basis functions are easy to program and fast to evaluate, the negative effect on the condition number can be prohibitive for high-order schemes.

Orthogonal basis functions are a very special class of basis with unique numerical properties. The basis functions of an orthogonal set are completely independent of each other; therefore, the mass matrix in Eqn 2.5 can be diagonalized for fast evaluation. Furthermore, the elements of a diagonalized matrix are the eigenvalues; hence a correct normalization of the orthogonal basis functions easily reduces the condition number to unity. In 2-D we use a special orthogonal basis functions called the Legendre polynomials for quadrilateral grids (see Section 4.1.3), and our own orthogonal basis functions for triangular grids (see Section 4.1.3). In short, if ϕ is a set of orthogonal basis functions on domain $\mathcal{D}$, then

$$\int_{\mathcal{D}} \phi_i \phi_k \, d\mathcal{D} = 0, \quad i \neq k, \tag{2.12}$$

$$\int_{\mathcal{D}} \phi_i \phi_k \, d\mathcal{D} \neq 0, \quad i = k. \tag{2.13}$$

Notice if we change the domain $\mathcal{D}$, the relationship above no longer holds and orthogonality is lost. In our example below, the quadrilateral Legendre polynomials (abbreviated as Q. Legendre) are orthogonal on the unit square domain; however, the quadrilateral Legendre polynomials are not orthogonal on a unit triangle domain. For recovery between two quadrilateral elements, using orthogonal Legendre basis functions for the solution and monomials for the recovery basis functions generally results in acceptable condition numbers. The best condition number achieved for quadrilateral recovery is when both the solution and recovery basis functions are orthogonal quadrilateral Legendre polynomials.

An elementary numerical study of recovery between triangular elements was performed with various choices of solution basis functions, and with or without Jacobi-preconditioner. We use the quadrilateral Legendre polynomials to illustrate the effect of a poor basis for recovery between triangles, and compared that to a real orthogonal basis for triangles. The results are detailed in the Table 2.1.

It is seen that the largest reduction in condition number is due to the correct choice of solution basis functions and the preconditioner. We have effectively reduced the conditon number by an order of magnitude when comparing the worst case to the best case. Note that simple orthogonal recovery basis functions rarely exist in the union of two elements except on Cartesian grids. Now that we have all the tools to acquire and condition $\boldsymbol{R}_{j,j+1}$, let us find out the physical meaning embedded within $\boldsymbol{R}_{j,j+1}$.

Max Condition #	ϕ	ψ	Jacobi Preconditioner
25	Orthogonal Triangular	Monomial	On
40	Q. Legendre	Monomial	On
116	Q. Legendre	Monomial	Off
286	Orthogonal Triangular	Monomial	Off
910	Q. Legendre	Q. Legendre	Off

Table 2.1: Condition number of $\boldsymbol{R}_{j,j+1}$ for recovery between two right triangular elements with $p = 1$ solutions. Notice quadrilateral Legendre polynomials (Q. Legendre) are no longer orthogonal on a triangle domain. Each face of the triangle has a different condition number; the maximum condition number of the three is reported.

2.4 Smooth recovery polynomial basis

Van Raalte and Van Leer [45] discovered the concept of a "local smooth recovery polynomial basis," which we call smooth recovery basis (SRB) for short. The term "local" refers to the union of two elements and the term "smooth" differentiates SRB from the discontinuous solution basis. There exist an infinite number of bases as potential candidates for ψ in Eqn 2.3; however, there is only one unique basis such that $\vec{b} = \vec{a}$ in Eqn 2.5. We called this unique basis the SRB, $\hat{\psi}$.

The existence of SRB allows us to interpret the discontinuous solution as a continuous solution. The two solutions are equivalent in the weak sense, but provide a completely different picture at the element interface. SRB is always defined in the union of the two elements; the discontinuous solutions are expressed in Eqn 2.1, and the smooth solution is expressed as,

$$u_{j,j+1} = \sum_{i=1}^{N_j+N_{j+1}} (\vec{a}_i)_{j,j+1} \, \hat{\psi}_i. \tag{2.14}$$

The smooth solution inherits the coefficients from the two neighboring discontinuous solutions; we call this the Principal Recovery Theorem. Depending on the nature of the problem, we may choose to use the discontinuous solution or the smooth solution. SRB provides a unified treatment of the advection and diffusion operator. The quest to find the members of SRB is not difficult; in fact they are hidden in $\boldsymbol{Q}_{j,j+1}$.

There exists one unique SRB for each choice of ϕ in Eqn 2.1. Let ψ in Eqn 2.3 be anything as long as it satisfies the Tetris game mentioned in the previous section and solve for $\boldsymbol{Q}_{j,j+1}$ in Eqn 2.6. The column vectors of $\boldsymbol{Q}_{j,j+1}$ span the vector space of $\vec{b}$ and may therefore serve as a basis, or the SRB. The SRB is now expressed in terms

of the elements of $\boldsymbol{Q}_{j,j+1}$ as,

$$\hat{\psi}_i = \sum_{k}^{N_j+N_{j+1}} \left(\boldsymbol{Q}_{k,i}\right)_{j,j+1} \psi_k. \tag{2.15}$$

We consider two 1-D examples, for $p = 0$ (Figure 2.5) and $p = 1$ (Figure 2.6), using the Legendre polynomials for ϕ. The domain of interest runs from 0 to 2, with Ω_j spanning from 0 to 1, and Ω_{j+1} spanning from 1 to 2.

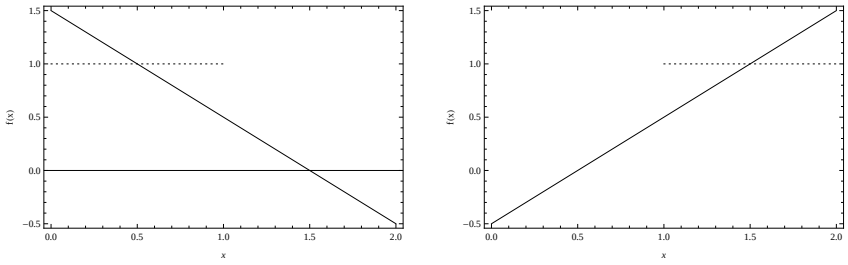

Figure 2.5: In the $p = 0$ case, there is a total of two SRB functions. The smooth recovery basis functions (solid lines) replacing $v_j = 1$ (dashed lines) in Ω_j (left), and $v_{j+1} = 1$ in Ω_{j+1} (right).

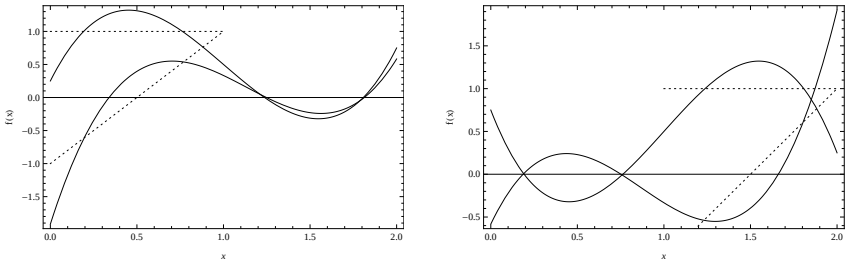

Figure 2.6: In the $p = 1$ case, there is a total of four SRB functions. The smooth recovery basis functions (solid lines) replacing $v_j = 1$ and $v_j = 2\xi - 1$ (dashed lines) in Ω_j (left), and $v_{j+1} = 1$ and $v_{j+1} = 2\xi - 1$ in Ω_{j+1} (right).

It is seen the SRB functions, though indistinguishable in a weak sense from the discontinuous basis functions, provide a completely different interpretation of the solution at the interface. Take the highest line in the left plot of Figure 2.6 as an example, this line shares the same average as $v_j = 1$ on Ω_j, but has an zero average gradient in Ω_j. In addition, this line has a zero average and zero averaged first gradient in Ω_{j+1}. The SRB version can be used to compute diffusive fluxes in a DG scheme; the discontinuous functions are preferred for advection if combined with a

Riemann solver. The physical meaning of matrix $\boldsymbol{Q}_{j,j+1}$ has been revealed; we are now ready to apply recovery to the diffusion equation.

2.5 Interface-centered recovery-based discontinuous Galerkin (iRDG) Method

The previous section was dedicated to explaining the mathematical properties of the recovery function. We now make the connection with the physical phenomenon of diffusion and detail the interface-centered recovery-based discontinuous Galerkin (iRDG) method for diffusion (we will drop the "i"). Diffusion is a physical process in which certain conserved properties of the fluid are transported due to molecular collisions. This quantities may be species concentration (mass), momentum, and energy of the fluid. The process occurs in the presence of spatial varying gradients; the diffusion process tends to reduce the magnitude of all derivatives. Consider the scalar nonlinear diffusion equation,

$$u_t = \nabla \cdot \left(D\left(u\right) \nabla u \right), \tag{2.16}$$

where $D\left(u\right)$ is the diffusion coefficient. We test the governing equation in element Ω and apply integration by parts once to the right hand side to arrive at the weak form,

$$\int_{\Omega} v\, u_t \, d\Omega = \oint_{\partial\Omega} \left(v D\left(u\right) \nabla u \right) \cdot \hat{n} \, d\partial\Omega - \int_{\Omega} \nabla v \cdot \left(D\left(u\right) \nabla u \right) \, d\Omega. \tag{2.17}$$

For simplicity, we have dropped the cell index j from v_j, u_j, Ω_j. In this form, we see the original 2nd-order derivative is reduced to a 1st-order derivative. It is useful to define $\mathfrak{D}$ as the primitive function of $D\left(u\right)$, i.e. $\nabla \mathfrak{D}\left(u\right) = D\left(u\right) \nabla u$. Eqn 2.17 thus may be rewritten as,

$$\int_{\Omega} v\, u_t \, d\Omega = \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(u\right) \right) \cdot \hat{n} \, d\partial\Omega - \int_{\Omega} \nabla v \cdot \nabla \mathfrak{D}\left(u\right) \, d\Omega. \tag{2.18}$$

We can apply integration by parts once more to arrive at a form sometimes called the strong form,

$$\int_{\Omega} v\, u_t \, d\Omega = \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(u\right) - \mathfrak{D}\left(u\right) \nabla v \right) \cdot \hat{n} \, d\partial\Omega + \int_{\Omega} \mathfrak{D}\left(u\right) \, \nabla \cdot \nabla v \, d\Omega. \tag{2.19}$$

The surface integral along $\partial\Omega$ is not well-defined due to the discontinuity at the element interface. The strong form brings more weight to the boundary as compared

to the weak form due to the extra boundary term. As argued before, the solution discontinuities at the interfaces are numerical artifacts; the only exception is if there actually is physical discontinuity in the initial values. In that case use of the diffusive Riemann flux discussed in Section 1.4.4 is appropriate; otherwise, recovery is indicated.

RDG first appeared in 2005 based on Eqn 2.19. The strong form works extremely well for linear diffusion because of the added coupling between elements with the extra boundary term in the surfae integral; in addition, the volume integral in the strong form can be evaluated exactly. The form also extends to scalar nonlinear diffusion but does not appear to allow extension to a nonlinear system of diffusion equations. Hence we have to seriously consider the weak form for nonlinear diffusion. We will consider three different ways to incorporate the recovered solution f into the weak and strong forms of the linear diffusion equation. For the weak form, we have both RDG-1x-Naive,

$$\int_{\Omega} v\, u_t \, d\Omega \;=\; \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(f\right)\right) \cdot \hat{n}\, d\partial\Omega - \int_{\Omega} \nabla v \cdot \nabla \mathfrak{D}\left(u\right)\, d\Omega. \tag{2.20}$$

and RDG-1x-Smart,

$$\int_{\Omega} v\, u_t \, d\Omega \;=\; \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(f\right)\right) \cdot \hat{n}\, d\partial\Omega - \int_{\Omega} \nabla v \cdot \nabla \mathfrak{D}\left(\tilde{u}\right)\, d\Omega, \tag{2.21}$$

where $\tilde{u}$ is some function that shares the same original moments as u, but contains higher order moments, inside Ω. The choice of names and various ways to construct $\tilde{u}$ will be explained in later chapters. For now it suffices to know RDG-1x-Naive performs poorly in comparison to RDG-1x-Smart. The term "1x" simply means integration by parts once, and "2x" means integration by parts twice. For the strong form of the diffusion equation, RDG-2x reads

$$\int_{\Omega} v\, u_t \, d\Omega \;=\; \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(f\right) - \mathfrak{D}\left(f\right) \nabla v\right) \cdot \hat{n}\, d\partial\Omega + \int_{\Omega} \mathfrak{D}\left(u\right)\, \nabla \cdot \nabla v \, d\Omega, \tag{2.22}$$

The following sections and chapters are dedicated to studying the stability and accuracy of RDG; we begin with the scalar linear diffusion equation in 1-D and progress towards the nonlinear Navier-Stokes equations in 2-D. The question of the stability range relates to the efficiency of the numerical method. Since diffusion results from molecular collisions in small time scales, the time step of a numerical method is severely restricted by the physical diffusion process itself. A numerical method

should not impose a further unnecessary penalty on the time step. The study of the accuracy of RDG includes demonstrating the improvement over existing methods, and capturing complicated diffusion-dominated phenomena such as those found in a boundary layer. We begin our analysis with stability proofs of various RDG schemes.

2.6 Evolution of the recovery-based discontinuous Galerkin methods

Since its inception in 2005, RDG has weathered numerous obstacles and grown into a mature method. The first RDG method was called RDG-2x and achieved an unsurpassed order of accuracy for the linear diffusion equation. The use of twice integrating by parts works well on the weighted residual of the linear diffusion equation because the resulting volume integral is evaluated exactly. However, twice integrating by parts for nonlinear diffusion equations is usually implausible because the primitive in Eqn. 2.17 does not exist.

A small cottage industry slowly grew from the seeds of the recovery concept. Marshall C. Galbraith applied a backward integration by parts on the volume integral to obtain the RDG-3x form,

$$\int_{\Omega} v\, u_t \, d\Omega \;=\; \oint_{\partial\Omega} \left(v \nabla \mathfrak{D}\left(f\right) + \mathfrak{D}\left(u - f\right) \nabla v\right) \cdot \hat{n} \, d\partial\Omega - \int_{\Omega} \nabla v \cdot \nabla \mathfrak{D}\left(u\right) \, d\Omega. \quad (2.23)$$

French, Galbraith and Osorio [12] worked further with a modified RDG method using a symmetric recovery procedure to facilitate math proofs. RDG-3x introduces additional terms and complexity into the boundary integral; hence, RDG-3x is usually used in proofs instead of numerical calculations. Park et al. [29] were the first to apply the recovery concept as an solution-enhancement technique (see Section 2.8). Solution enhancement increases the order of accuracy of a scheme without introducing additional solution coefficients; this has a significant positive impact on the performance of DG schemes by reducing the memory burden.

Till the end of 2009, our research was focused on Fourier analysis, stability proofs and extension to 2-D unstructured grids. RDG-1x was of direct interest due to its shorter form; however, the accuracy of the volume integral has to be improved. In this respect two new schemes in the RDG-1x-Smart family were introduced, RDG-1x$\bar{f}$ and RDG-1x$\tilde{f}$. These schemes are limited in success due to instability for higher p and added complexity on unstructured grids. In 2010 we recognized the need to improve on the solution and the recovery function inside the volume and surface

integral, respectively, to accommodate the nonlinear diffusion equations. The newest scheme, RDG-1x+ and RDG-1x++, are shown to work with Navier-Stokes viscous terms, and numerical Fourier analysis shows the two schemes are stable and extremely accurate. The "+" symbol indicates the number of times that the recovery information is reused. In RDG-1x+, the recovered function is used for the the diffusion flux, and then it is reused to enhance the solution inside the volume integral. RDG-1x++ takes it one step further by reusing the enhanced solution to obtain a new enhanced recovered function. An experimental RDG-0x+ scheme is also introduced briefly to demonstrate the ability to discretize the PDE without any integration by parts. RDG-0x+ achieves this by absorbing the surface integral into the volume integral, allowing us to produce very efficient schemes. Currently the RDG-1x-Smart family is too complicated for a general mathematical proof; as a result, we only provide numerical evidence of stability in Chapters 3 and 4. Meanwhile, we continue with the mathematical proofs of stability for both RDG-2x and RDG-1x-Naive.

2.7 Stability proofs for interface recovery-based discontinuous Galerkin methods

A popular method of studying stability of DG methods is to look at the time rate of change of the total energy of the solution. The strategy is to show the energy is decaying in time, or equivalently showing the RHS's of Eqns 2.20, 2.21, and 2.23 are negative if $v = u$. We first begin with the original RDG-2x method; the proof of RDG-1x-Naive is a simplified version of RDG-2x.

2.7.1 Proof for RDG-2x

The proof first focuses on the $p \geq 1$ case, and then deals with the case $p = 0$, which is different since the volume integral vanishes. In Eqn 2.23 we replace the test function v by the numerical solution u, thus testing u with itself; then we integrate over the complete computational domain $\mathcal{V}$ with boundary $\mathcal{S}$,

$$\begin{aligned} \int_{\mathcal{V}} \frac{\partial}{\partial t}\left(\frac{u^2}{2}\right) d\mathcal{V} &= \oint_{\mathcal{S}} \left(u\nabla\mathfrak{D}\left(f\right) - \mathfrak{D}\left(f\right)\nabla u\right)\cdot\hat{n}\, d\mathcal{S} \\ &+ \oint_{\mathcal{E}} \left[u\nabla\mathfrak{D}\left(f\right) - \mathfrak{D}\left(f\right)\nabla u\right]\cdot\hat{n}\, d\mathcal{E} \\ &+ \int_{\mathcal{V}} \mathfrak{D}\left(u\right)\,\nabla\cdot\nabla u\, d\mathcal{V}. \end{aligned} \tag{2.24}$$

The LHS is the time rate of change of the energy of the solution, and the RHS consists of a surface integral over the computational domain boundary $\mathcal{S}$, a surface integral over all element interfaces $\mathcal{E}$, and a volume integral of the whole domain $\mathcal{V}$. Note that if the normal $\hat{n}$ is defined as pointing from element Ω_j to element Ω_{j+1}, the jump across an interface must be defined as, $[\cdot] = (\cdot)_j - (\cdot)_{j+1}$. We undo the second integration by parts by performing a backward integration by parts on the the volume integral,

$$\begin{aligned}
\int_{\mathcal{V}} \frac{\partial}{\partial t}\left(\frac{u^2}{2}\right) d\Omega &= \oint_{\mathcal{S}} \left(u\nabla\mathfrak{D}(f) - \mathfrak{D}(f)\nabla u\right)\cdot\hat{n}\, d\mathcal{S} \\
&+ \oint_{\mathcal{E}} \left[u\nabla\mathfrak{D}(f) + \left(\mathfrak{D}(u) - \mathfrak{D}(f)\right)\nabla u\right]\cdot\hat{n}\, d\mathcal{E} \\
&- \int_{\mathcal{V}} \nabla u\cdot\nabla\mathfrak{D}(u)\, d\mathcal{V}. \qquad (2.25)
\end{aligned}$$

The diffusion coefficient is not a function of mesh size, hence we assume $D(u) \sim O(1)$. There are three terms on the RHS with the third term being a volume integral that is negative definite of $O(1)$. The strategy is to show the first surface integral terms, the domain-boundary surface integral and the element-interface surface integral, are much smaller than the volume integral. Consider a normalized physical domain where $\mathcal{V}$ and $\mathcal{S}$ are of $O(1)$, then $\mathcal{E}$ is $O(h^{-1})$ where h is a typical mesh width and much smaller than unity. It is easy to visualize the total area of the element-interfaces is much larger than the area of the domain boundary, therefore we conclude the element-interface surface integral is potentially much larger than the domain-boundary surface integral. Focusing on the element-interface surface integral, we rewrite the integrand in terms of average $< \cdot >$ and jump $[\cdot]$ notations:

$$\begin{aligned}
\left[u\nabla\mathfrak{D}(f) + \left(\mathfrak{D}(u) - \mathfrak{D}(f)\right)\nabla u\right]\cdot\hat{n} &= \left\{[u]\nabla\mathfrak{D}(f) + [\mathfrak{D}(u) - \mathfrak{D}(f)] < \nabla u >\right\}\cdot\hat{n} + \\
&\quad + \left\{< \mathfrak{D}(u) - \mathfrak{D}(f) > [\nabla u]\right\}\cdot\hat{n}, \\
&= \left\{([u]\nabla\mathfrak{D}(f) + [\mathfrak{D}(u)] < \nabla u >)\right\}\cdot\hat{n} + \\
&\quad + \left\{(< \mathfrak{D}(u) > -\mathfrak{D}(f))[\nabla u]\right\}\cdot\hat{n}. \qquad (2.26)
\end{aligned}$$

Assume the initial values, of which u is the projection, are sufficiently smooth, say, $p+1$ times continuously differentiable, then both $[u]$ and $< \mathfrak{D}(u) > -\mathfrak{D}(f)$ are of $O(h^{p+1})$, while $[\nabla u]$ is $O(h^p)$ for $p \geq 1$ and 0 for $p = 0$. Hence the two terms in parentheses on the RHS of Eqn 2.26 are of $O(h^{p+1})$ and $O(h^{2p+1})$ respectively for $p \geq 1$. The first term with $O(h^{p+1})$ dominates the integrand; as a result the second term in Eqn 2.25 is of $O(h^p)$. For a sufficiently fine grid, the boundary integrals

becomes negligibly small compared to the negative definite volume integral.

For $p = 0$, $\nabla u = 0$, so the negative-definite integral reduces to zero, and the surface integral $\mathcal{E}$ assumes its role. We need to compare the magnitude of the surface integral over $\mathcal{S}$ to the surface integral over $\mathcal{E}$. We first look at the magnitude of the element-interface surface integral term. The integrand in Eqn 2.26 simplifies to $[u]\,\nabla\mathfrak{D}\,(f)\cdot\hat{n}$. In this case $[u] = O\,(h)$ and the recovery procedure yields

$$\nabla\mathfrak{D}\,(f)\cdot\hat{n} = -\frac{[\mathfrak{D}\,(u)]}{h_{cn}}, \tag{2.27}$$

where h_{cn} is the distance between the centroids of neighboring elements measured along their interface-normal. The integrand becomes,

$$[u]\,\nabla\mathfrak{D}\,(f)\cdot\hat{n} = -\frac{\overline{\mathrm{D}}\,[u]^2}{h_{cn}} = -|O\,(h)\,| \rightarrow \oint_{\mathcal{E}} [u]\,\nabla f\cdot\hat{n}\,d\mathbf{E} = -\mid O\,(1)\mid; \tag{2.28}$$

here $\overline{\mathrm{D}} = \frac{1}{[u]}\int_{jump}\mathfrak{D}\,(u)\;du > 0$. The boundary conditions are applied on the domain-boundary surface integral. We show for both Dirichlet and Neumann boundary conditions that the surface integral over $\mathcal{S}$ for $p = 0$ is still negligibly small. In the RDG method, the boundary condition is applied on the recovery function f, not the interior solution u. For a Dirichlet boundary condition with $f = 0$ on $\mathcal{S}$, the integrand for $p \geq 0$ reduces to

$$u\nabla\mathfrak{D}\,(f) + (\mathfrak{D}\,(u) - \mathfrak{D}\,(f))\,\nabla u = (u-f)\,\nabla\mathfrak{D}\,(f) + (\mathfrak{D}\,(u) - \mathfrak{D}\,(f))\,\nabla u = O\left(h^{p+1}\right), \tag{2.29}$$

which again is small compared to terms of $O\,(1)$. For an adiabatic Neumann boundary condition with $\nabla\mathfrak{D}\,(f)\cdot\hat{n} = 0$ on $\mathcal{S}$, the integral is simplified to

$$\oint_S \nabla\frac{(u-f)^2}{2}\cdot\hat{n}\,d\mathcal{S} = O\left(h^{2p+1}\right),\; p \geq 0. \tag{2.30}$$

Once again this term is small compared to $O\,(1)$. This implies for that $p \geq 0$ the negative-definite element-interface surface integral dominates the domain-boundary surface integral. This completes the stability proof for all p. The proof of RDG-1x mimics the one for RDG-2x.

2.7.2 Proof for RDG-1x-Naive

We first replace the test functions with the numerical solution u in Eqn 2.20 and integrate over the complete computational domain $\mathcal{V}$ with boundary $\mathcal{S}$,

$$\begin{aligned}\int_{\mathcal{V}} \frac{\partial}{\partial t}\left(\frac{u^2}{2}\right) d\mathcal{V} &= \oint_{\mathcal{S}} \left(u\nabla\mathfrak{D}\left(f\right)\right)\cdot\hat{n}\, d\mathcal{S} \\ &+ \oint_{\mathcal{E}} \left[u\nabla\mathfrak{D}\left(f\right)\right]\cdot\hat{n}\, d\mathcal{E} \\ &- \int_{\mathcal{V}} \nabla u\cdot\left(D\left(u\right)\nabla u\right)\, d\mathcal{V}. \end{aligned} \tag{2.31}$$

The volume integral is negative definite of magnitude $O\left(1\right)$ for $p > 0$. Again, we consider a normalized physical domain where $\mathcal{V}$ and $\mathcal{S}$ are of $O\left(1\right)$, then $\mathcal{E}$ is $O\left(h^{-1}\right)$ where h is a number much smaller than unity. Using the same argument as for RDG-2x, we conclude the element-interface surface integral is the largest of the two surface integrals. We may rewrite its integrand as

$$\left[u\nabla\mathfrak{D}\left(f\right)\right]\cdot\hat{n} = \left\{\left[u\right]\nabla\mathfrak{D}\left(f\right)\right\}\cdot\hat{n}, \tag{2.32}$$

here $[u]$ is of magnitude $O\left(h^{p+1}\right)$, making the internal surface integral $O\left(h^{p}\right)$ for $p \geq 1$. For a sufficiently fine grid, both boundary integrals become negligibly small compared to the negative definite volume integral.

For $p = 0$, RDG-2x is identical to RDG-1x-Naive. The proof from the previous sections applies here; we conclude RDG-1x-Naive is energy stable for all p.

2.7.3 Note on RDG-1x-Smart

RDG-1x-Smart is a family of schemes designed to improve on the surface and volume integral in the weak form of the diffusion operator. There is no stability proof of RDG-1x-Smart due to the large number of choices for $\tilde{f}$ and $\tilde{u}$; currently the members include RDG-1x$\bar{f}$, RDG-1x$\tilde{f}$, RDG-1x+, RDG-1x++, and RDG-1x++CO where "CO" stands for Cartesian optimization. In later chapters, we use numerical experiment and Fourier analysis to show that certain members of the smart family are stable for various p. We believe the proof is in the pudding; the extremely high order of accuracy is not coincidental. Research on improving RDG-1x-Smart is ongoing; a stability proof for RDG-1x-Smart may appear in a future publication.

2.8 Recovery concept for solution enhancement

In this thesis, solution enhancement is defined as a process to add higher order components to the solution. A full solution enhancement adds enough components to raise the solution polynomial by a full order. Park et al. [29] experimented with full solution enhancement using the recovery concept over a large stencil in cell-centered RDG (cRDG). Since the main expense of their numerical experiments comes from solving a large implicit system, their objective is to increase the order of the scheme without increasing the number of solution coefficients. Full solution enhancement is extremely costly; hence, we explore cheaper ways of enhancing the solution.

We differentiate ourselves by focusing on recovery between two cells only (binary recovery). Binary recovery involves far fewer cells, which significantly reduces the size of matrix operations and allows for effective parallelization. Although binary recovery does not provide a full solution enhancement, it does allow us to selectively choose the correct components to add to the solution for a more accurate evaluation of the diffusion operator. As we will show later, one cannot blindly pursue a full solution enhancement due to the rising cost in higher physical dimension; one must carefully choose the components to enhance based on the nature of the governing equations.

CHAPTER III

RDG in One-Dimension for Scalar Equation

Our research principle is to design a user-friendly DG diffusion scheme which everybody can understand without a master's degree in applied mathematics. This chapter is intended to explore and reveal the most fundamental properties of RDG in a simple 1-D setting. Focusing on the linear diffusion equation, we analyze the scheme for linear stability and accuracy. This simple setting also provides a common framework to compare RDG with various members of the (σ, μ)-family. Most importantly, when we rewrite RDG schemes in terms of jumps and averages of quantities across cell interfaces, RDG schemes contain penalty-like terms previously unseen in the (σ, μ)-family. Finally, we couple RDG with Runge-Kutta (RK) time-marching and solve the advection-diffusion and nonlinear diffusion equations.

Before the show begins, we provide a list of the RDG schemes in the order of appearance. The first scheme is the original RDG-2x which uses twice integration by parts on the weighted-residual formulation of the linear diffusion equation and the recovery concept for the diffusion flux. We then compare RDG with the (σ, μ)-family, which usually follow from the form once integrated by parts. On this platform we introduce RDG-1x$\bar{f}$ which attempts to improve upon RDG-1x-Naive. Finally, we present RDG-1x+ for nonlinear diffusion. Out of the many schemes introduced in this chapter, we focus on analysis of RDG-2x only for the following reasons. First, numerical experiment and Fourier analysis show RDG-1x$\bar{f}$ is unstable for $p \geq 3$. Secondly, the RDG-1x+ scheme is equivalent to RDG-2x for 1-D linear diffusion equation; the analysis of RDG-2x applies directly to RDG-1x+. Lastly, RDG-1x-Naive is not of particular interest because of its lack of accuracy.

3.1 1-D recovery equations for binary recovery

The basic ingredient of any RDG scheme is the recovered function based on binary recovery (BR). The recovered function provides an unique solution value and derivatives at a cell interface where the solution was originally undefined due to the discontinuity. As the name of BR implies, only the two cells sharing one common interface are involved. Consider the solutions in two adjacent cells of the form presented in Eqn 2.1, we look for a recovered function centered on the shared interface in the form of Eqn 2.3. The 1-D recovery equations are

$$\begin{aligned} \int_{\Omega_j} v_j u_j \, dx &= \int_{\Omega_j} v_j f_{j,j+1} \, dx, \\ \int_{\Omega_{j+1}} v_{j+1} u_{j+1} \, dx &= \int_{\Omega_{j+1}} v_{j+1} f_{j,j+1} \, dx. \end{aligned} \tag{3.1}$$

The recovered function is in a coordinate system different from that of the solutions of each cell. We introduce the r-coordinate for the recovered function and the ξ-coordinate for the local solution. The transformation between r and ξ coordinates is strictly geometry-dependent; the details are explained below. We consider three examples with Legendre polynomials for piecewise-constant ($p = 0$), piecewise-linear ($p = 1$) and piecewise-quadratic ($p = 2$) solutions on an uniform grid.

3.1.1 Piecewise-constant ($p = 0$) recovery

A DG scheme with $p = 0$ is equivalent to a finite-volume scheme. Let the test space V be equal to the solution space U, and be spanned by the following basis function,

$$v_0 = 1, \quad \xi \in [0,\, 1]\,, \tag{3.2}$$

where ξ is the local coordinate, and the solution in Ω_j be expressed as

$$u_j = a_{0,j} v_{0,j} = a_{0,j}. \tag{3.3}$$

Here, a_0 can be viewed as the cell average. We look for a recovered function between Ω_j and Ω_{j+1} of the form,

$$f_{j,j+1}\,(r) = b_0 + b_1 r, \quad r \in [-1,\, 1], \tag{3.4}$$

such that Ω_j belongs to $r \in [-1, 0]$ and Ω_{j+1} belongs to $r \in [0, 1]$ (see Figure 3.1). The transformation of variable is given by

$$r = \xi - 1, \quad \text{for } \Omega_j, \tag{3.5}$$

$$r = \xi - 0, \quad \text{for } \Omega_{j+1}. \tag{3.6}$$

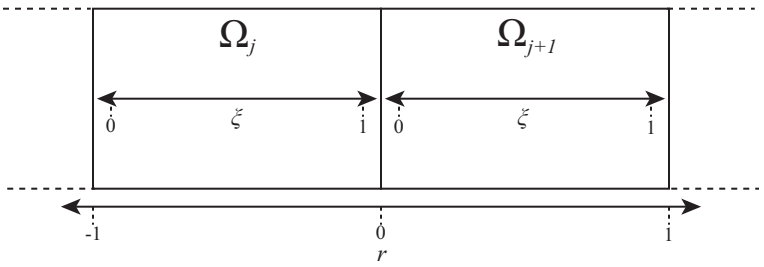

Figure 3.1: The recovery coordinate system spans the union of two neighboring cells, while each cell has a local coordinate ξ.

Note that $r = 0$ is centered on the interface shared by Ω_j and Ω_{j+1}. Eqn 3.1 is rewritten into matrix-vector form after performing the necessary integrations,

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_{0,j} \\ a_{0,j+1} \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} \\ 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}_{j,j+1}. \tag{3.7}$$

Solving for b_i gives,

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix}_{j,j+1} = \begin{pmatrix} \frac{a_{0,j}+a_{0,j+1}}{2} \\ -a_{0,j} + a_{0,j+1} \end{pmatrix}. \tag{3.8}$$

3.1.2 Piecewise-linear $(p = 1)$ recovery

Let the test space V be equal to the solution space U, and be spanned by the following basis functions,

$$v_0 = 1,\ v_1 = 2\xi - 1, \quad \xi \in [0, 1]\,, \tag{3.9}$$

where ξ is the local coordinate, and the solution in Ω_j be expressed as,

$$u_j = a_{0,j} v_{0,j} + a_{1,j} v_{1,j}. \tag{3.10}$$

We look for a recovered function between Ω_j and Ω_{j+1} of the form,

$$f_{j,j+1}(r) = b_0 + b_1 r + b_2 r^2 + b_3 r^3, \quad r \in [-1, 1], \tag{3.11}$$

such that Ω_j belongs to $r \in [-1,\, 0]$ and Ω_{j+1} belongs to $r \in [0,\, 1]$. Using the same variable transformation as the previous section, Eqn 3.1 is rewritten into matrix-vector form after performing the necessary integrations,

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} a_{0,j} \\ a_{1,j} \\ a_{0,j+1} \\ a_{1,j+1} \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & \frac{1}{3} & -\frac{1}{4} \\ 0 & \frac{1}{6} & -\frac{1}{6} & \frac{3}{20} \\ 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ 0 & \frac{1}{6} & \frac{1}{6} & \frac{3}{20} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}_{j,j+1} . \tag{3.12}$$

Solving for b_i gives,

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}_{j,j+1} = \begin{pmatrix} \frac{a_{0,j}+a_{0,j+1}}{2} + \frac{a_{1,j}-a_{1,j+1}}{3} \\ \frac{9(a_{0,j+1}-a_{0,j})}{4} - \frac{5(a_{1,j}+a_{1,j+1})}{4} \\ a_{1,j+1} - a_{1,j} \\ \frac{5(a_{0,j+1}-a_{0,j})}{2} + \frac{5(a_{1,j}+a_{1,j+1})}{2} \end{pmatrix} . \tag{3.13}$$

3.1.3 Piecewise-quadratic $(p = 2)$ recovery

Let the test space V be equal to the solution space U, and be spanned by the following basis functions,

$$v_0 = 1,\, v_1 = 2\xi - 1,\, v_2 = 6\xi^2 - 6\xi + 1, \quad \xi \in [0,\, 1]\,, \tag{3.14}$$

where ξ is the local coordinate, and the solution in Ω_j be expressed as,

$$u_j = a_{0,j} v_{0,j} + a_{1,j} v_{1,j} + a_{2,j} v_{2,j}. \tag{3.15}$$

We look for a recovered function between Ω_j and Ω_{j+1} of the form,

$$f_{j,j+1}\,(r) = b_0 + b_1 r + b_2 r^2 + b_3 r^3 + b_4 r^4 + b_5 r^5, \quad r \in [-1,\, 1]. \tag{3.16}$$

The resulting system in matrix-vector form looks like

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{5} \end{pmatrix} \begin{pmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{0,j+1} \\ a_{1,j+1} \\ a_{2,j+1} \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & \frac{1}{3} & -\frac{1}{4} & \frac{1}{5} & -\frac{1}{6} \\ 0 & \frac{1}{6} & -\frac{1}{6} & \frac{3}{20} & -\frac{2}{15} & \frac{5}{42} \\ 0 & 0 & \frac{1}{30} & -\frac{1}{20} & \frac{2}{35} & -\frac{5}{84} \\ 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ 0 & \frac{1}{6} & \frac{1}{6} & \frac{3}{20} & \frac{2}{15} & \frac{5}{42} \\ 0 & 0 & \frac{1}{30} & \frac{1}{20} & \frac{2}{35} & \frac{5}{84} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}_{j,j+1}, \tag{3.17}$$

with solution

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}_{j,j+1} = \begin{pmatrix} \frac{a_{0,j}+a_{0,j+1}}{2} + \frac{13(a_{1,j}-a_{1,j+1})}{32} + \frac{7(a_{2,j}+a_{2,j+1})}{32} \\ \frac{15(-a_{0,j}+a_{0,j+1})}{4} - \frac{11(a_{1,j}+a_{1,j+1})}{4} + \frac{6(-a_{2,j}+a_{2,j+1})}{32} \\ \frac{15(-a_{1,j}+a_{1,j+1})}{8} - \frac{21(a_{2,j}+a_{2,j+1})}{8} \\ \frac{25(a_{0,j}-a_{0,j+1})}{2} + \frac{25(a_{1,j}+a_{1,j+1})}{2} + 8\left(a_{2,j} - a_{2,j+1}\right) \\ \frac{35(a_{1,j}-a_{1,j+1})}{32} + \frac{105(a_{2,j}+a_{2,j+1})}{32} \\ \frac{21(-a_{0,j}+a_{0,j+1})}{2} + \frac{21(a_{1,j}+a_{1,j+1})}{2} + \frac{42(-a_{2,j}+a_{2,j+1})}{5} \end{pmatrix}. \tag{3.18}$$

3.2 RDG-2x for linear diffusion

The original RDG-2x is applied to the spatial derivatives on the RHS of the linear diffusion equation, while the temporal derivative is handled by an explicit time-marching Runge-Kutta (RK) scheme. This section demonstrates the discretization of the RHS of the linear diffusion equation with RDG-2x for $p = 0$, 1 and 2. The recovered function is used as the flux component in the RDG-2x scheme,

$$\int_{\Omega_j} v u_t \, dx = D \left[v f_x - v_x f \right]_{j-\frac{1}{2}}^{j+\frac{1}{2}} + D \int_{\Omega_j} v_{xx} u \, dx, \tag{3.19}$$

where the solution u is replaced with the smooth recovered function f in the surface integral. In 1-D the surface integrals reduce to point values of f and f_x at the interface. However, the update equation is given in terms the global spatial coordinate x; it is necessary to convert the global coordinate into local coordinate for a DG scheme. The transformation between ξ and x is given by $\frac{d\xi}{dx} = \frac{1}{\Delta x}$; the update equation in terms of ξ is then

$$\Delta x \int_{\Omega_j} v u_t \, d\xi = \frac{D}{\Delta x} \left[v f_\xi - v_\xi f \right]_{j-\frac{1}{2}}^{j+\frac{1}{2}} + \frac{D}{\Delta x} \int_{\Omega_j} v_{\xi\xi} u \, d\xi. \tag{3.20}$$

Recall that f uses a different coordinate system, where the transformation from r to ξ is conveniently given by $\frac{dr}{d\xi} = 1$, hence $f_\xi = f_r$. Upon inserting the recovered functions from the previous section, we obtain the update matrix for each cell Ω_j.

Update scheme for RDG-2x ($p = 0$)

With only one basis function, $v_0 = 1$, the update equation reduces to

$$\frac{d}{dt}\left(a_{0_j}\right)\Delta x = \frac{D}{\Delta x}\left[f_\xi\right]_{j-\frac{1}{2}}^{j+\frac{1}{2}} = \frac{D}{\Delta x}\left[\left(f_\xi\right)_{j,j+1} - \left(f_\xi\right)_{j-1,j}\right], \tag{3.21}$$

where the first derivative of recovered function evaluates to

$$\left(f_r\left(0\right)\right)_{j,j+1} = \left(b_1\right)_{j,j+1} = -a_{0,j} + a_{0,j+1}, \tag{3.22}$$

$$\left(f_r\left(0\right)\right)_{j-1,j} = \left(b_1\right)_{j,j+1} = -a_{0,j-1} + a_{0,j}. \tag{3.23}$$

The final update scheme becomes,

$$\begin{aligned}\frac{d}{dt}\left(a_{0_j}\right) &= \frac{D}{\Delta x^2}\left(a_{0,j+1} - 2a_{0,j} + a_{0,j-1}\right). \\ &= \frac{D}{\Delta x^2}\left([1]\,[-2]\,[1]\right)\begin{pmatrix} a_{0,j-1} \\ a_{0,j} \\ a_{0,j+1}\end{pmatrix}.\end{aligned} \tag{3.24}$$

Indeed the RDG-2x scheme for $p = 0$ reduces to the central difference scheme which is 2nd-order accurate in space, and the only consistent scheme for $p = 0$. Although the $p = 0$ case is not of interest in the DG framework, it is still satisfying to show the RDG-2x scheme comes from a very good pedigree. Note that the update equation is rewritten into matrix-vector form which will come in handy in a later section.

Update scheme for RDG-2x ($p = 1$)

Using the linear basis functions $v_0 = 1$ and $v_1 = 2\xi - 1$, the update equations become

$$\frac{d}{dt}\begin{pmatrix} a_{0_j} \\ \frac{a_{1_j}}{3}\end{pmatrix} = \frac{D}{\Delta x^2}\begin{pmatrix} \left(f_\xi\right)_{j,j+1} - \left(f_\xi\right)_{j-1,j} \\ \left(\left(f_\xi\right)_{j,j+1} + \left(f_\xi\right)_{j-1,j}\right) - 2\left(f_{j,j+1} - f_{j-1,j}\right)\end{pmatrix}. \tag{3.25}$$

Here $f(0) = b_0$ and $f_r(0) = b_1$ from Eqn 3.13; the final update scheme becomes

$$\frac{d}{dt}\begin{pmatrix} a_{0_j} \\ \frac{a_{1_j}}{3} \end{pmatrix} = \frac{D}{\Delta x^2}\left(\begin{bmatrix} \frac{9}{4} & \frac{5}{4} \\ -\frac{5}{4} & -\frac{7}{12} \end{bmatrix}\begin{bmatrix} -\frac{9}{2} & 0 \\ 0 & -\frac{23}{6} \end{bmatrix}\begin{bmatrix} \frac{9}{4} & -\frac{5}{4} \\ \frac{5}{4} & -\frac{7}{12} \end{bmatrix}\right)\begin{pmatrix} \begin{pmatrix} a_{0,j-1} \\ a_{1,j-1} \end{pmatrix} \\ \begin{pmatrix} a_{0,j} \\ a_{1,j} \end{pmatrix} \\ \begin{pmatrix} a_{0,j+1} \\ a_{1,j+1} \end{pmatrix} \end{pmatrix}. \tag{3.26}$$

Update scheme for RDG-2x ($p = 2$)

Using the quadratic basis functions $v_0 = 1$, $v_1 = 2\xi - 1$, and $v_2 = 6\xi^2 - 6\xi + 1$, the update equations become

$$\frac{d}{dt}\begin{pmatrix} a_{0,j} \\ \frac{a_{1,j}}{3} \\ \frac{a_{2,j}}{5} \end{pmatrix} = \frac{D}{\Delta x^2}\begin{pmatrix} (f_\xi)_{j,j+1} - (f_\xi)_{j-1,j} \\ \left((f_\xi)_{j,j+1} + (f_\xi)_{j-1,j}\right) - 2\left(f_{j,j+1} - f_{j-1,j}\right) \\ \left((f_\xi)_{j,j+1} - (f_\xi)_{j-1,j}\right) - 6\left(f_{j,j+1} + f_{j-1,j}\right) + 12\int_{\Omega_j} u\, d\xi \end{pmatrix}. \tag{3.27}$$

Here $f(0) = b_0$ and $f_r(0) = b_1$ from Eqn 3.13; the final update scheme becomes,

$$\frac{d}{dt}\begin{pmatrix} a_{0,j} \\ \frac{a_{1,j}}{3} \\ \frac{a_{2,j}}{5} \end{pmatrix} =$$

$$\frac{D}{\Delta x^2}\left(\begin{bmatrix} \frac{15}{4} & \frac{11}{4} & \frac{6}{5} \\ -\frac{11}{4} & -\frac{31}{16} & -\frac{61}{80} \\ \frac{3}{4} & \frac{5}{16} & -\frac{9}{80} \end{bmatrix}\begin{bmatrix} -\frac{15}{2} & 0 & -\frac{12}{5} \\ 0 & -\frac{57}{8} & 0 \\ -\frac{3}{2} & 0 & -\frac{201}{40} \end{bmatrix}\begin{bmatrix} \frac{15}{4} & -\frac{11}{4} & \frac{6}{5} \\ \frac{11}{4} & -\frac{31}{16} & \frac{61}{80} \\ \frac{3}{4} & -\frac{5}{16} & -\frac{9}{80} \end{bmatrix}\right)\begin{pmatrix} \begin{pmatrix} a_{0,j-1} \\ a_{1,j-1} \\ a_{2,j-1} \end{pmatrix} \\ \begin{pmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \end{pmatrix} \\ \begin{pmatrix} a_{0,j+1} \\ a_{1,j+1} \\ a_{2,j+1} \end{pmatrix} \end{pmatrix}. \tag{3.28}$$

3.3 Fourier analysis and eigenvectors

As Professor David Gottlieb famously noted, "we can always analyze a 1-D system to death," which in scientific language means a very complete analysis can be performed on a simple system of equations. We use Fourier analysis to reveal the order of accuracy and stability limit of RDG schemes for the linear diffusion equation. The key components in our Fourier analysis is to identify the eigenvalues and eigenvectors. We begin by giving an overview of the steps involved in the Fourier analysis, and explaining the physical meaning behind the eigenvalues and eigenvectors.

Our Fourier analysis is based on the scalar linear diffusion equation,

$$u_t = D u_{xx}, \tag{3.29}$$

where D is a positive constant. The weighted-residual formulation after once integration by parts becomes,

$$\int_\Omega v u_t \, dx = D \oint_{\partial\Omega} v u_x \, d\partial\Omega - D \int_\Omega v_x u_x \, dx, \tag{3.30}$$

or twice integrated by parts,

$$\int_\Omega v u_t \, dx = D \oint_{\partial\Omega} \left(v u_x - v_x u \right) \, d\partial\Omega + D \int_\Omega v_{xx} u \, dx. \tag{3.31}$$

Note we dropped the cell index from v and u for simplicity. We obtain RDG-2x by replacing the solution u in the surface integral with the recovered function f,

$$\int_\Omega v u_t \, dx = D \oint_{\partial\Omega} \left(v f_x - v_x f \right) \, d\partial\Omega + D \int_\Omega v_{xx} u \, dx. \tag{3.32}$$

Our next step is to rewrite the RHS of Eqn 3.32 as a discrete Fourier operator and compare the result with the exact Fourier operator. Fourier operators are expressed in terms of the Fourier mode, β. The solution in cells adjacent to Ω_j is related through the Fourier mode,

$$\hat{u}_{j+k} = e^{i\beta k} \hat{u}_j = T^k \hat{u}_j, \tag{3.33}$$

where T^k is the translation operator and $\hat{u}_j$ is a vector of solution coefficients. The RHS of Eqn 3.32 is simply expressed as,

$$\frac{1}{D} \frac{\partial}{\partial t} \hat{u}_j = M\left(\beta\right) \hat{u}_j, \tag{3.34}$$

where $M(\beta)$ is a matrix representing the discrete Fourier operator. The exact Fourier symbols for 1st and 2nd-order spatial differentiations are

$$\frac{\hat{\partial}}{\partial x} = \frac{i\beta}{\Delta x}, \tag{3.35}$$

$$\frac{\hat{\partial}^2}{\partial x^2} = -\frac{\beta^2}{\Delta x^2}, \tag{3.36}$$

respectively; hence, the exact Fourier operator for the 2nd-order diffusion equation is

$$\hat{M}(\beta) = -D\beta^2 I, \tag{3.37}$$

where I is the identity matrix. A comparison of the eigenvalues of the discrete Fourier operator with the exact Fourier operator reveals the level of accuracy at which the numerical scheme approximates the diffusion operator. The signs of the eigenvalues also dictate the stability of a scheme; in this case, a positive eigenvalue indicates instability. However, the eigenvalues only allow us to determine the stability range and the evolution error. The other issue of crucial concern is the initial projection error, which is a measure of the scheme's ability to preserve a smooth solution. The projection error is determined from the eigenvectors of $M(\beta)$. We are now ready for some Fourier analysis.

3.3.1 RDG-2x Fourier analysis: piecewise-linear ($p = 1$)

Consider a numerical solution expanded in terms of a solution vector $\vec{u}_j$ and the piecewise-linear Legendre basis functions in the local spatial coordinate,

$$\vec{u}_j = \begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \end{pmatrix}, \tag{3.38}$$

$$u_j = \bar{u}_j + \overline{\Delta u}_j (2\xi - 1), \quad \xi \in [0, 1], \tag{3.39}$$

The complete update scheme with RDG for solving Eqn 3.32 is given by,

$$\frac{\partial}{\partial t}\begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \end{pmatrix} = \frac{D}{\Delta x^2}\left(\begin{bmatrix} \frac{9}{4} & \frac{5}{4} \\ -\frac{15}{4} & -\frac{7}{4} \end{bmatrix} \begin{bmatrix} -\frac{9}{2} & 0 \\ 0 & -\frac{23}{2} \end{bmatrix} \begin{bmatrix} \frac{9}{4} & -\frac{5}{4} \\ \frac{15}{4} & -\frac{7}{4} \end{bmatrix} \right) \begin{pmatrix} T^{-1}(\vec{u}_j) \\ I(\vec{u}_j) \\ T^{+1}(\vec{u}_j) \end{pmatrix}. \tag{3.40}$$

Note this is very similar to Eqn. 3.26, except T^k is the translation operator such that

$$T^{+1}\left(\vec{u}_j\right) = \vec{u}_{j+1}, \tag{3.41}$$

$$I\left(\vec{u}_j\right) = \vec{u}_j, \tag{3.42}$$

$$T^{-1}\left(\vec{u}_j\right) = \vec{u}_{j-1}, \tag{3.43}$$

where j is a spatial index, and I is the identity operator. We convert the translation operators into Fourier modes via

$$T^{\pm j} = e^{\pm i\,\beta\,j} = \cos\left(j\beta\right) \pm i \sin\left(j\beta\right), \tag{3.44}$$

and arrive at the following form,

$$\frac{\partial}{\partial x}\begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \end{pmatrix} = \frac{D}{\Delta x^2}\begin{pmatrix} -\frac{9}{2}\left(1-\cos\beta\right) & -\frac{5}{2} i \sin\beta \\ \frac{15}{2} i \sin\beta & -\frac{1}{2}\left(23+7\cos\beta\right) \end{pmatrix}\begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \end{pmatrix} \tag{3.45}$$

$$= M_{\text{RDG-2x},p1}\left(\beta\right)\begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \end{pmatrix}. \tag{3.46}$$

The two eigenvalues of $M_{\text{RDG-2x},p1}$ are given by,

$$\lambda_{1,2} = -8 + \frac{\cos\beta}{2} \pm \sqrt{118 + 112\cos\beta + \left(\cos\beta\right)^2 - 6\cos 2\beta}, \tag{3.47}$$

and then Taylor-expanding the eigenvalues for small β reveals,

$$\lambda_1 = -\beta^2 + \frac{\beta^6}{360} + O\left(\beta^8\right), \tag{3.48}$$

$$\lambda_2 = -15 + \frac{\beta^2}{2} + \frac{\beta^4}{24} - \frac{\beta^6}{240} + O\left(\beta^8\right). \tag{3.49}$$

The left side of Figure 3.2 shows the two eigenvalues attempting to represent the diffusion operator with the curve $\lambda = -\beta^2$. By convention, the order of accuracy of a scheme usually refers to the eigenvalues for small β, or low frequency data. The contribution of high frequency data to the diffusion operator is less significant (they are also known as "noise"). The evolution error is determined from the good eigenvalue λ_1, such that $\lambda_{error} = |\lambda_1 + \beta^2|$. We define the order of accuracy of a scheme to be, O.O.A. $= \frac{\lambda_{error}}{\beta^2}$. In this case, the order of accuracy RDG-2x ($p = 1$) is four. Notice the first eigenvalue approximates the curve $\lambda = -\beta^2$ until $\beta = \pi$, and then the second eigenvalue takes over. This observation leads to the discovery of important passive

elements of DG schemes. DG schemes inherently contain "switches" which turn on and off eigenvalues to maximize the accuracy in representing the solution. The switches are not engineered, but possibly nature's way of dealing with sub-grid information in an element. The study of these switches is done by analyzing the contribution from each of the eigenvectors.

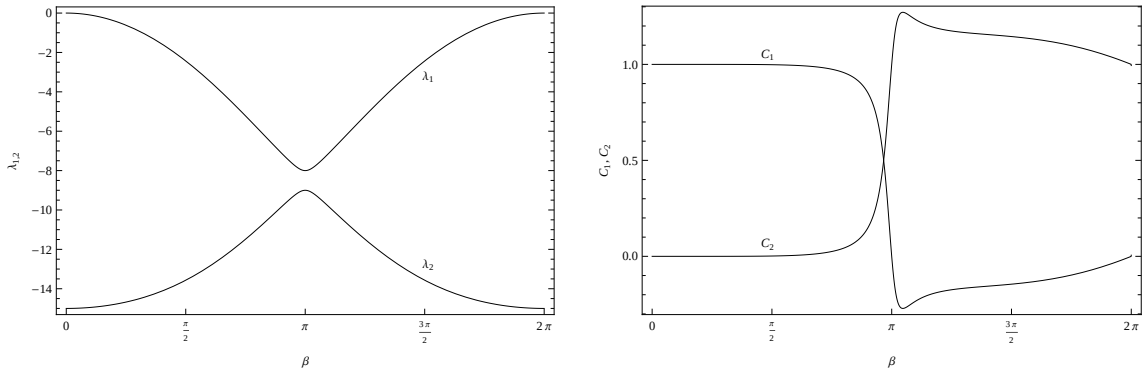

Figure 3.2: Eigenvalues of RDG-2x ($p = 1$) are shown on the left. The switch functions are shown on the right demonstrating the contribution of each eigenfunction for various β.

Consider the initial value of the solution to be consisted of Fourier modes,

$$U(x) = U_0 e^{\frac{i\beta x}{\Delta x}}, \tag{3.50}$$

then the initial discretization in each element Ω_j is given by

$$\overline{u}_j = \frac{\int_{\Omega_j} v_0 U \, dx}{\int_{\Omega_j} v_0^2 \, dx} = \frac{2\sin\left(\frac{\beta}{2}\right)}{\beta} U_0, \tag{3.51}$$

$$\overline{\Delta u}_j = \frac{\int_{\Omega_j} v_1 U \, dx}{\int_{\Omega_j} v_1^2 \, dx} = -\frac{6i\left(\beta\cos\left(\frac{\beta}{2}\right) - 2\sin\left(\frac{\beta}{2}\right)\right)}{\beta^2} U_0. \tag{3.52}$$

The general strategy of this analysis is to feed waves of different Fourier modes β into the RDG diffusion operator, and observe how well the RDG operator preserves the

initial waves. We conveniently normalized the eigenvectors by $\bar{u}_j$,

$$\boldsymbol{r}_1 = \begin{pmatrix} \frac{2\sin\left(\frac{\beta}{2}\right)}{\beta} \\ \frac{30i\sin\left(\frac{\beta}{2}\right)\sin(\beta)}{\beta\left(8\cos(\beta)-\sqrt{\cos^2(\beta)+112\cos(\beta)-6\cos(2\beta)+118}+7\right)} \end{pmatrix}, \tag{3.53}$$

$$\boldsymbol{r}_2 = \begin{pmatrix} \frac{2\sin\left(\frac{\beta}{2}\right)}{\beta} \\ \frac{30i\sin\left(\frac{\beta}{2}\right)\sin(\beta)}{\beta\left(8\cos(\beta)+\sqrt{\cos^2(\beta)+112\cos(\beta)-6\cos(2\beta)+118}+7\right)} \end{pmatrix}. \tag{3.54}$$

We define the corresponding local eigenfunctions of the eigenvectors as,

$$\begin{aligned} g_1(\xi) &= \boldsymbol{r}_1(1) + \boldsymbol{r}_1(2)(2\xi-1)\Delta x \\ g_2(\xi) &= \boldsymbol{r}_2(1) + \boldsymbol{r}_2(2)(2\xi-1)\Delta x \end{aligned} \qquad \xi \in \Omega_j, \tag{3.55}$$

where the initial solution is composed of a linear combination of the eigenfunctions,

$$U(x) = C_1 g_1 + C_2 g_2. \tag{3.56}$$

here the coefficients $C_{1,2}$ are the switches, or weighting coefficients. If the weighting coefficient is zero, it acts like a switch turning off the eigenvector, whereas if the weighting coefficient is unity, the switch turns on the eigenvector. Expanding the initial solution in terms of Eqns 3.51 and 3.52,

$$U(x) = U_0\left(\frac{2\sin\left(\frac{\beta}{2}\right)}{\beta}(1) - \frac{6i\left(\beta\cos\left(\frac{\beta}{2}\right) - 2\sin\left(\frac{\beta}{2}\right)\right)}{\beta^2}(2\xi-1)\right), \tag{3.57}$$

we solve for $C_{1,2}$ numerically for a wide range of β. The solutions of $C_{1,2}$ are plotted on the right side of Figure 3.2. The switches clearly show the first eigenvector is the key player for low frequency waves ($\beta < \pi$) , while the second eigenvector takes care of the higher frequency waves. This is also observed by focusing on specific wavelengths of the eigenfunctions as shown in Figure 3.3-3.4 .

In order to correctly determine the magnitude of the initial projection error, we focus on the switches for small β. The figures show g_1 to be the good eigenvector and g_2 to be the bad eigenvector; hence the contribution from g_2 should be minimized. Taylor expanding the switch coefficients for small β reveals,

$$C_1 = 1 - \frac{\beta^6}{9450} + O\left(\beta^8\right), \tag{3.58}$$

$$C_2 = \frac{\beta^6}{9450} + O\left(\beta^8\right). \tag{3.59}$$

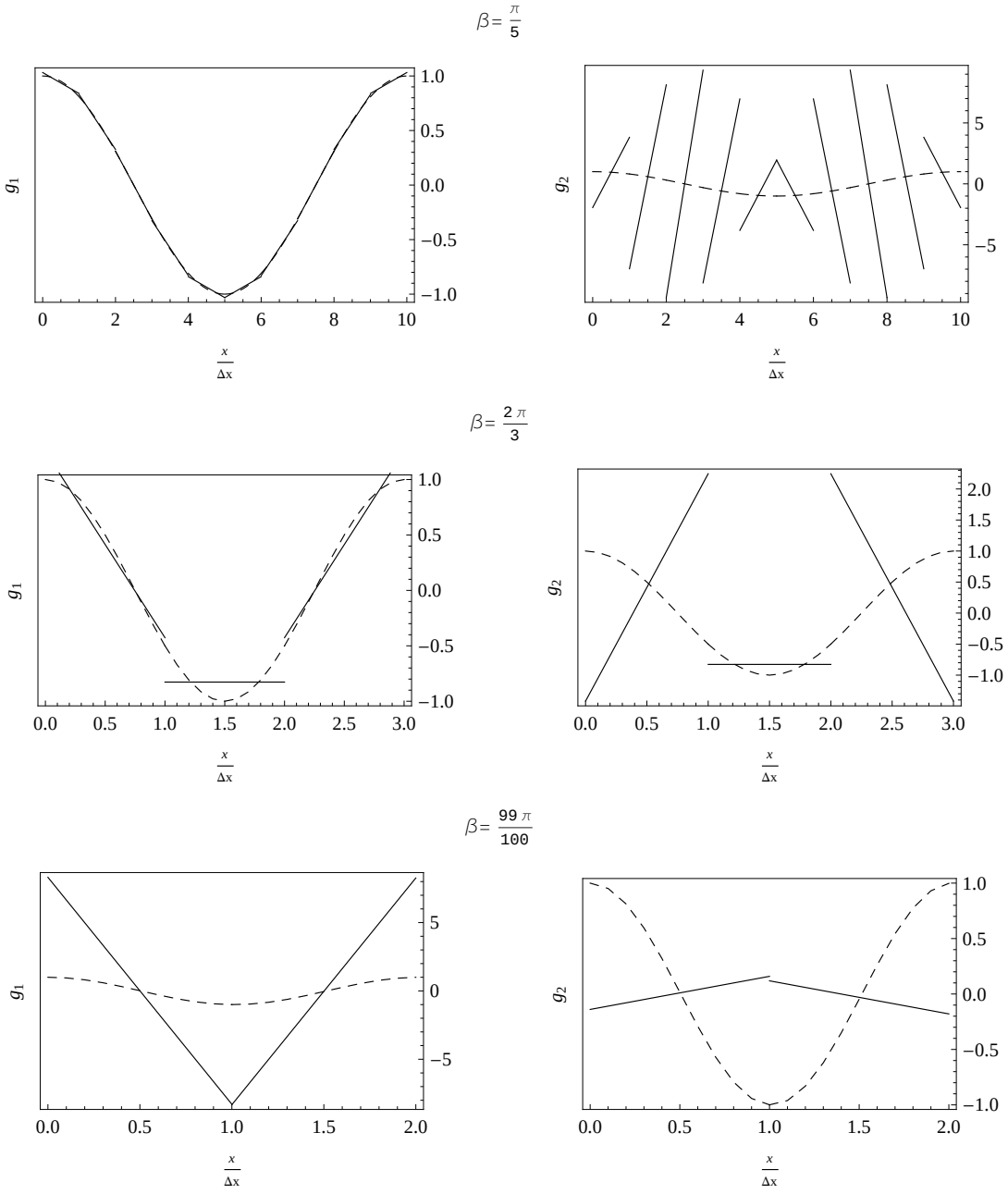

Figure 3.3: Eigenfunctions of RDG-2x ($p = 1$) for β between $\frac{\pi}{5}$ and $\frac{99\pi}{100}$. Notice the change in scaling of both x-axis and g-axis. The dashed line represents the analytical wave.

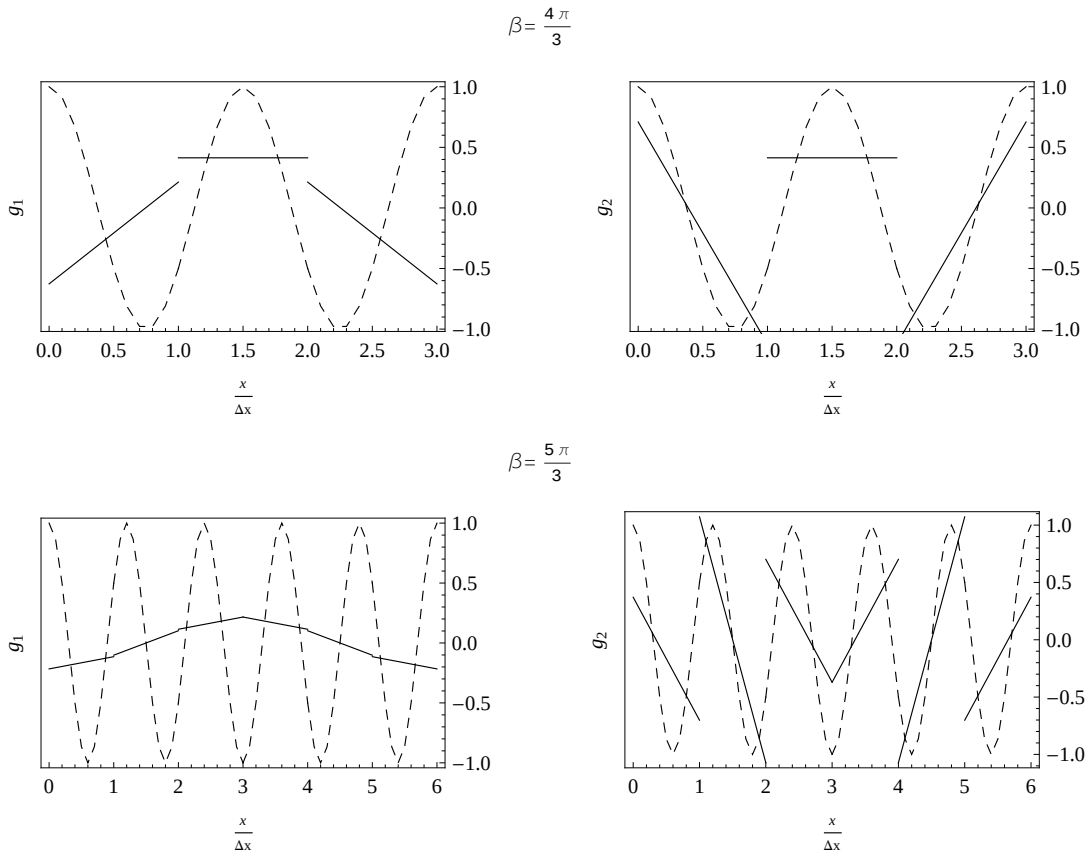

Figure 3.4: Eigenfunctions of RDG-2x ($p = 1$) for β for $\frac{4\pi}{3}$ and $\frac{5\pi}{3}$. Notice the change in scaling of both x-axis and g-axis. The dashed line represents the analytical wave.

Due to the way the eigenvectors are normalized, g_2 contains the exact cell average, but an incorrect gradient. The incorrect gradient of order $\frac{1}{\beta}$ is then multiplied with C_2 of order β^6, resulting in a initial error of β^5. For RDG-2x ($p = 1$), the evolution error of $O(\beta^4)$ dominates the initial projection error; the end result is a 4th-order scheme.

3.3.2 RDG-2x Fourier analysis: piecewise-quadratic ($p = 2$)

Consider a numerical solution expanded in terms of a solution vector $\vec{u}_j$ and the piecewise-quadratic Legendre basis functions in local spatial coordinate,

$$\vec{u}_j = \begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \\ \overline{\Delta^2 u}_j \end{pmatrix}, \tag{3.60}$$

$$u_j = \bar{u}_j + \overline{\Delta u}_j (2\xi - 1) + \overline{\Delta^2 u}_j \left(6\xi^2 - 6\xi + 1\right), \quad \xi \in [0,\, 1], \tag{3.61}$$

Without repeating the details, the complete update scheme for solving Eqn. 3.32 is given by,

$$\begin{aligned} \frac{\partial}{\partial t}(\vec{u}_j) &= \frac{D}{\Delta x^2} M_{\text{RDG-2x},p2} \begin{pmatrix} T^{-1}(\vec{u}_j) \\ I(\vec{u}_j) \\ T^{+1}(\vec{u}_j) \end{pmatrix}, \\ M_{\text{RDG-2x},p2} &= \begin{bmatrix} \frac{15}{4} & \frac{11}{4} & \frac{6}{5} \\ -\frac{33}{4} & -\frac{93}{16} & -\frac{183}{80} \\ \frac{15}{4} & \frac{25}{16} & -\frac{9}{16} \end{bmatrix} \begin{bmatrix} -\frac{15}{2} & 0 & -\frac{12}{5} \\ 0 & -\frac{171}{8} & 0 \\ -\frac{15}{2} & 0 & -\frac{201}{8} \end{bmatrix} \begin{bmatrix} \frac{15}{4} & -\frac{11}{4} & \frac{6}{5} \\ \frac{33}{4} & -\frac{93}{16} & \frac{183}{80} \\ \frac{15}{4} & -\frac{25}{16} & -\frac{9}{16} \end{bmatrix} \end{aligned} \tag{3.62}$$

or in Fourier mode,

$$\frac{\partial}{\partial t}(\vec{u}_j) = \frac{D}{\Delta x^2} \begin{pmatrix} \frac{15\cos(\beta)}{2} - \frac{15}{2} & -\frac{11}{2} i \sin(\beta) & \frac{12\cos(\beta)}{5} - \frac{12}{5} \\ \frac{33}{2} i \sin(\beta) & -\frac{93\cos(\beta)}{8} - \frac{171}{8} & \frac{183}{40} i \sin(\beta) \\ \frac{15\cos(\beta)}{2} - \frac{15}{2} & -\frac{25}{8} i \sin(\beta) & -\frac{9\cos(\beta)}{8} - \frac{201}{8} \end{pmatrix} (\vec{u}_j). \tag{3.63}$$

The three eigenvalues of $M_{\text{RDG-2x},p2}$ are shown on the left plot of Figure 3.5. Notice λ_1 and λ_2 do not cross each other, but instead act in a manner of a relay race. λ_1 represents the diffusion operator until $\beta = \pi$ and passes it onto λ_2.

A closer look at the good eigenvalue λ_1 for small β in Figure 3.6 reveals that the order of the scheme is eight. Again, the eigenvalue only tells us the evolution error, we now look into the projection error.

Consider the same initial value distribution as $p = 1$ case with the addition of a

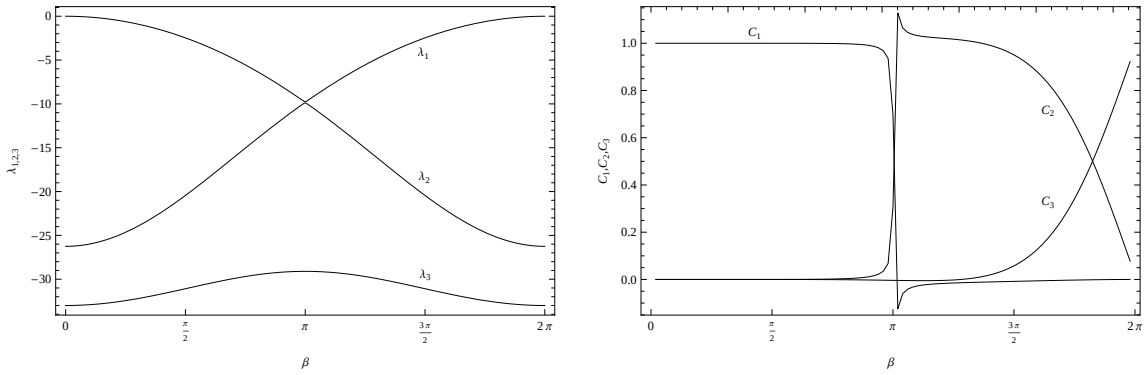

Figure 3.5: Eigenvalues of RDG-2x ($p = 2$), are shown on the left. The switch functions are shown on the right demonstrating the contribution of each eigenfunction for various β.

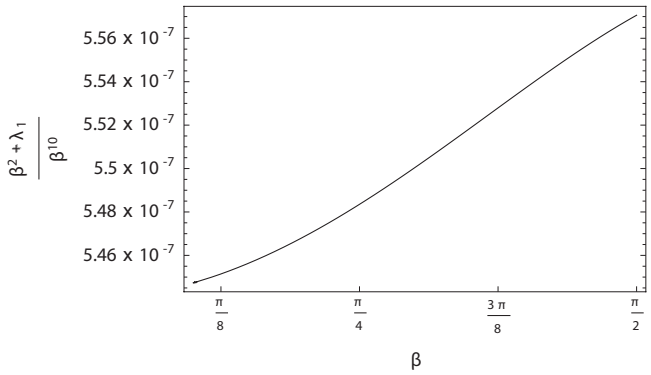

Figure 3.6: A closer look at the good eigenvalue of RDG-2x ($p = 2$), for small β. The difference between the good eigenvalue and the exact diffusion operator is of order β^{10}, implying the evolution error of the scheme is 8th-order.

second gradient,

$$\overline{\Delta^2 u_j} = \frac{\int_{\Omega_j} v_2 U \, dx}{\int_{\Omega_j} v_2^2 \, dx} = \frac{10\left((\beta^2 - 12)\sin\left(\frac{\beta}{2}\right) + 6\beta\cos\left(\frac{\beta}{2}\right)\right)}{\beta^3} U_0. \tag{3.64}$$

The eigenvectors $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$ are too complicated to be presented in analytic form. We define the corresponding local eigenfunctions of the eigenvectors as

$$\begin{aligned} g_1(\xi) &= \boldsymbol{r}_1(1)\, v_0 + \boldsymbol{r}_1(2)\, v_1 \Delta x + + \boldsymbol{r}_1(2)\, v_2 \Delta x^2 \\ g_2(\xi) &= \boldsymbol{r}_2(1)\, v_0 + \boldsymbol{r}_2(2)\, v_1 \Delta x + + \boldsymbol{r}_2(2)\, v_2 \Delta x^2 \quad \xi \in \Omega_j, \\ g_2(\xi) &= \boldsymbol{r}_3(1)\, v_0 + \boldsymbol{r}_3(2)\, v_1 \Delta x + + \boldsymbol{r}_3(2)\, v_2 \Delta x^2 \end{aligned} \tag{3.65}$$

where the initial solution is composed of a linear combination of the eigenfunctions,

$$U(x) = C_1 g_1 + C_2 g_2 + C_3 g_3. \tag{3.66}$$

We first acquire $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$ numerically for fixed β, and then solve for the switches $C_{1,2,3}$. The right plot of Figure 3.5 shows the switches in action for various values of β. Figure 3.7 shows the structure of the three eigenfunctions for different Fourier modes.

The figures show that g_1 is the good eigenvector for small β. We look for the contribution of the bad eigenvectors by determining the magnitude of C_2 and C_3. Figure 3.8 shows that both C_2 and C_3 are of order β^8. This is already puzzling: we would not expect the scheme to be 8th-order accurate with such projection errors, but numerical results suggest it is. In order to determine the full magnitude of the projection error, we also need to know the error in the average first and second gradients of g_1 as a function of β. We have not pursued this; the question of the projection error of RDG-2x for $p \geq 2$ is a subject for furture research.

Fourier analysis paints an interesting picture of the inner workings of RDG-2x. As we can observe already, the complexity of the analysis for piecewise-quadratic is quite cumbersome; therefore, we proceed no further with higher p. Next, we perform Fourier analysis for various penalty methods and compare them with the results for the RDG-2x scheme.

3.3.3 The venerable (σ, μ)-family Fourier analysis for $p = 1$

This section takes us all the way back to 1982 when Arnold introduced the first successful discretization of the DG diffusion operator with the interior penalty method (IPM). Van Raalte (2005) put the penalty schemes into a two-parameter family, called the (σ, μ)-family, but did not go beyond the traditional pairs of values. Van Leer [42] was the first to explore the entire family for $p = 1$. Our immediate goal is to highlight the difference between RDG-2x and the (σ, μ)-family through Fourier analysis. We then explore the (σ, μ)-plane for worthy schemes which were previously undiscovered.

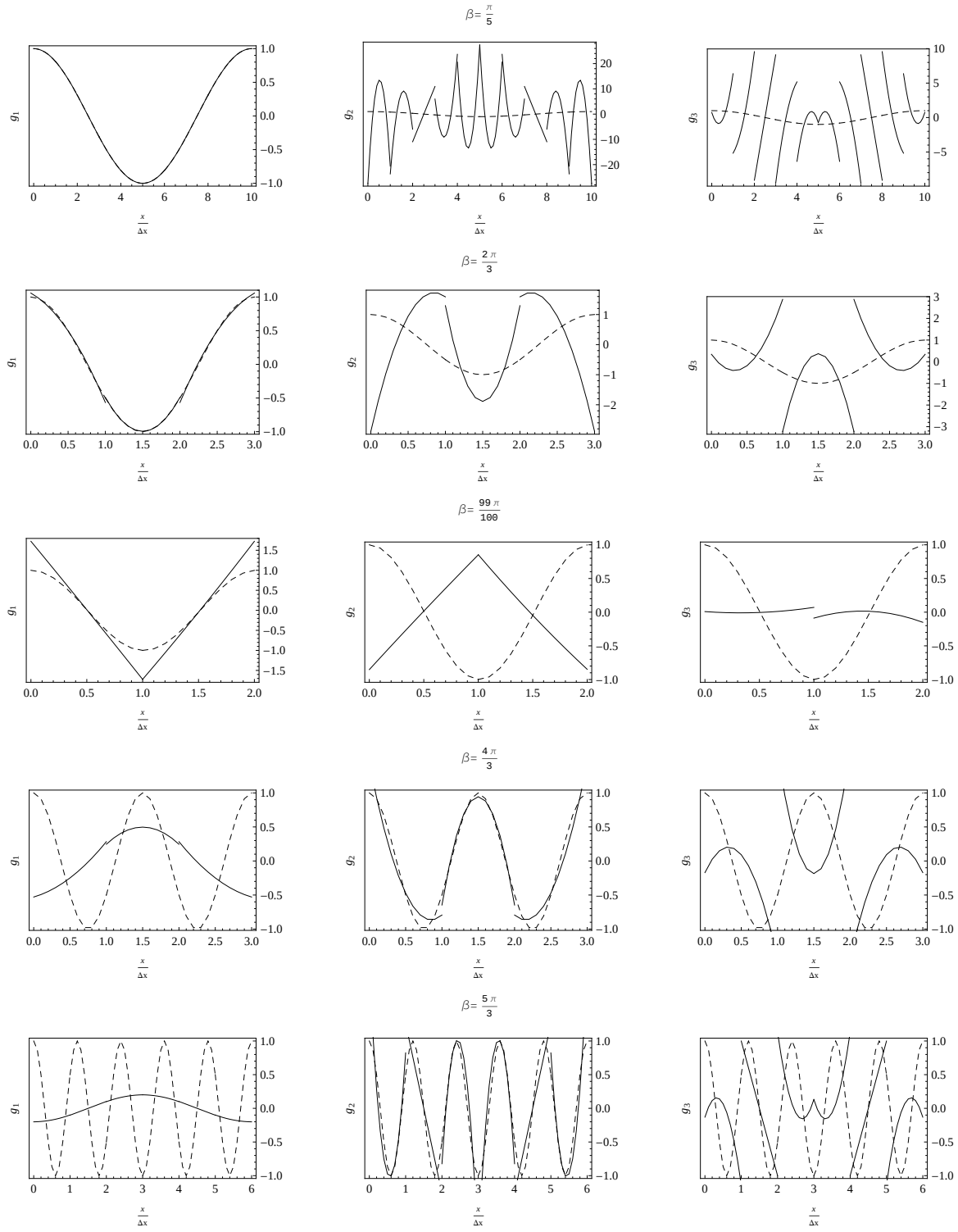

Figure 3.7: Eigenfunctions of RDG-2x ($p = 2$) for β between $\frac{\pi}{5}$ and $\frac{5\pi}{3}$. Notice the change in scaling of both x-axis and g-axis. The dashed line represents the analytical wave.

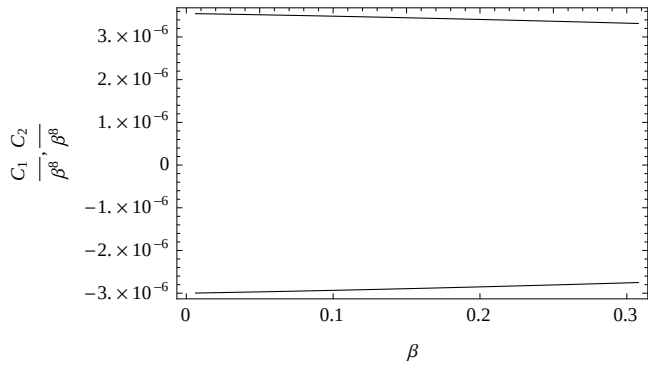

Figure 3.8: For small β, C_1 and C_2 are the weighting coefficients of the bad eigenvectors. They scale with β^8. However, no solid conclusion can be drawn from this analysis due to the lack of analytical formulas.

The (σ, μ)-family for the linear diffusion equation is given by,

$$\begin{aligned}\int_{\Omega_j} v u_t \, dx &= -D\left(\langle u_x\rangle [v] \mid_{j+\frac{1}{2}} + \langle u_x\rangle [v] \mid_{j-\frac{1}{2}}\right) - D\int_{\Omega_j} v_x u_x \, dx \\ &+ \sigma D\left(\langle v_x\rangle [u] \mid_{j+\frac{1}{2}} + \langle v_x\rangle [u] \mid_{j-\frac{1}{2}}\right) \\ &- \frac{\mu D}{\Delta x}\left([v][u] \mid_{j+\frac{1}{2}} - [v][u] \mid_{j-\frac{1}{2}}\right), \end{aligned} \tag{3.67}$$

where $< \ . \ >$ is the average operator across an interface, and $[.]$ is the difference operator across an interface from the right to the left. There is a distinct difference between applying these operators to u and v. When applied to the solution, u is taken globally such that the value on each side of the interface comes from the two cells sharing the interface, providing the necessary coupling between cells. When applied to the test function, v is taken locally such that the value outside the cell is always zero. Term 1 on the RHS is needed to achieve consistency for $p > 0$. For example, if $p = 1$, term 1 becomes,

$$D\left(< u_x >|_{j+\frac{1}{2}} - < u_x >|_{j-\frac{1}{2}}\right). \tag{3.68}$$

For smooth problems, this may appear to be a good approximation for the diffusion flux, but it ignores the jump in the solution. Picture two abutting elements with different piecewise-constant solution; term 1 always evaluate to zero because the average gradient is zero. But the physics of diffusion should induce a flux from high to low! Term 1 and term 2 together yield an inconsistent scheme. Term 3 is a penalty term with coefficient σ. With the choice of $\sigma = -1$, terms 1 to 3 form a symmetric operator; however, stability is not guaranteed. Term 4 is the interior penalty term

introduced by Arnold, which seems necessary for both stability, and for $p = 0$ even for consistency. In terms of $(\sigma,\, \mu)$, the four venerable members are the inconsistent $(0, 0)$, symmetric $(-1, 0)$, stabilized symmetric/Arnold $(-1, 1)$, and Baumman $(1, 0)$ schemes.

Consider a numerical solution expanded in terms of a solution vector $\vec{u}_j$ and the piecewise-linear Legendre basis functions in global spatial coordinate,

$$\vec{u}_j = \begin{pmatrix} \overline{u}_j \\ \overline{\Delta u_j} \end{pmatrix}, \tag{3.69}$$

$$u_j = \bar{u}_j + \overline{\Delta u}_j \frac{(x - x_j)}{\Delta x}, \quad x \in \left[x_{j-\frac{1}{2}},\, x_{j+\frac{1}{2}} \right], \tag{3.70}$$

The update equation for $(\sigma,\, \mu)$-family $(p = 1)$ in matrix form is

$$\frac{\partial}{\partial t}\left(\vec{u}_j\right) = \frac{D}{\Delta x^2} \begin{pmatrix} [M_{L,\sigma-\mu}] & [M_{C,\sigma-\mu}] & [M_{R,\sigma-\mu}] \end{pmatrix} \begin{pmatrix} T^{-1}\left(\vec{u}_j\right) \\ I\left(\vec{u}_j\right) \\ T^{+1}\left(\vec{u}_j\right) \end{pmatrix},$$

$$M_{L,\sigma-\mu} = \begin{pmatrix} \mu & -\frac{(1-\mu)}{2} \\ -6\left(\sigma+\mu\right) & 3 - 3\left(\sigma+\mu\right) \end{pmatrix},$$

$$M_{C,\sigma-\mu} = \begin{pmatrix} -2\mu & 0 \\ 0 & -6 - 6\left(\sigma+\mu\right) \end{pmatrix},$$

$$M_{R,\sigma-\mu} = \begin{pmatrix} \mu & \frac{(1-\mu)}{2} \\ 6\left(\sigma+\mu\right) & 3 - 3\left(\sigma+\mu\right) \end{pmatrix}. \tag{3.71}$$

which is written into Fourier mode,

$$\begin{aligned} \frac{\partial}{\partial t}\left(\vec{u}_j\right) &= \frac{D}{\Delta x^2} \begin{pmatrix} -2\mu\left(1-\cos\beta\right) & i\left(1-\mu\right)\sin\beta \\ 12i\left(\sigma+\mu\right)\sin\beta & -6\left(1+\sigma+\mu\right) + 6\left(1-\sigma-\mu\right)\cos\beta \end{pmatrix} \left(\vec{u}_j\right) \\ &= \frac{D}{\Delta x^2} M_{(\sigma-\mu)}\left(\beta\right)\left(\vec{u}_j\right). \end{aligned} \tag{3.72}$$

Again we want the eigenvalues of $M_{(\sigma-\mu)}$ to approximate the exact diffusion operator

in Eqn 3.37. The eigenvalues are found in the following form,

$$\lambda_1 = C_0^{\lambda_1} + C_2^{\lambda_1}\beta^2 + C_4^{\lambda_1}\beta^4 + C_6^{\lambda_1}\beta^6 + O\left(\beta^8\right), \tag{3.73}$$

$$\lambda_2 = C_0^{\lambda_2} + C_2^{\lambda_2}\beta^2 + C_4^{\lambda_2}\beta^4 + C_6^{\lambda_2}\beta^6 + O\left(\beta^8\right). \tag{3.74}$$

where all the coefficients, $C_j^{\lambda} = C_j^{\lambda}(\sigma, \mu)$, are dependent on σ and μ. Notice there are no odd-order exponents in the coefficients of the eigenvalues. The coefficients C_j^{λ} are found with Mathematica, and they are too complicated to be revealed. Table 3.2 provides the order of accuracy of various (σ, μ)-schemes based on the first eigenvalue. Recall from the previous section that the two eigenvalues play different roles depending on the frequency; λ_1 approximates the exact spatial operator in the low frequency range, while λ_2 approximates of the spatial operator in the high frequency range.

The (σ, μ) playing field

Our analysis of the (σ, μ)-family is summarized on the map of the $\sigma - \mu$ plane in Figure 3.9. Six lines of importance are drawn, their equations are:

$$\sigma + \mu = 0, \tag{3.75}$$

$$\sigma + \mu = \frac{5}{2}, \tag{3.76}$$

$$\sigma - \mu = -2, \tag{3.77}$$

$$\sigma + \mu = 1, \tag{3.78}$$

$$\mu = 1, \tag{3.79}$$

$$\mu = 0. \tag{3.80}$$

The first line, $\sigma + \mu = 0$, is the line of separation between stable and unstable domains. The domain above the line is stable, while the region below is unstable. On the line specifically, $\mu \geq 0$ is stable and $\mu < 0$ is unstable. Stability comes from both eigenvalues in Eqn 3.74 being negative, e.g. $C_0^{\lambda_{1,2}} \leq 0$ only, and the other coefficients are ignored because they are multiplied with higher even powers of small β. Assuming $\sigma + \mu \neq 0$, the eigenvalues of $M_{(\sigma-\mu)}$ are

$$\lambda_1 = -\beta^2 + O\left(\beta^4\right), \tag{3.81}$$

$$\lambda_2 = -12(\sigma + \mu) + O\left(\beta^2\right). \tag{3.82}$$

λ_1 is always negative; on the other hand, we must enforce the following condition to

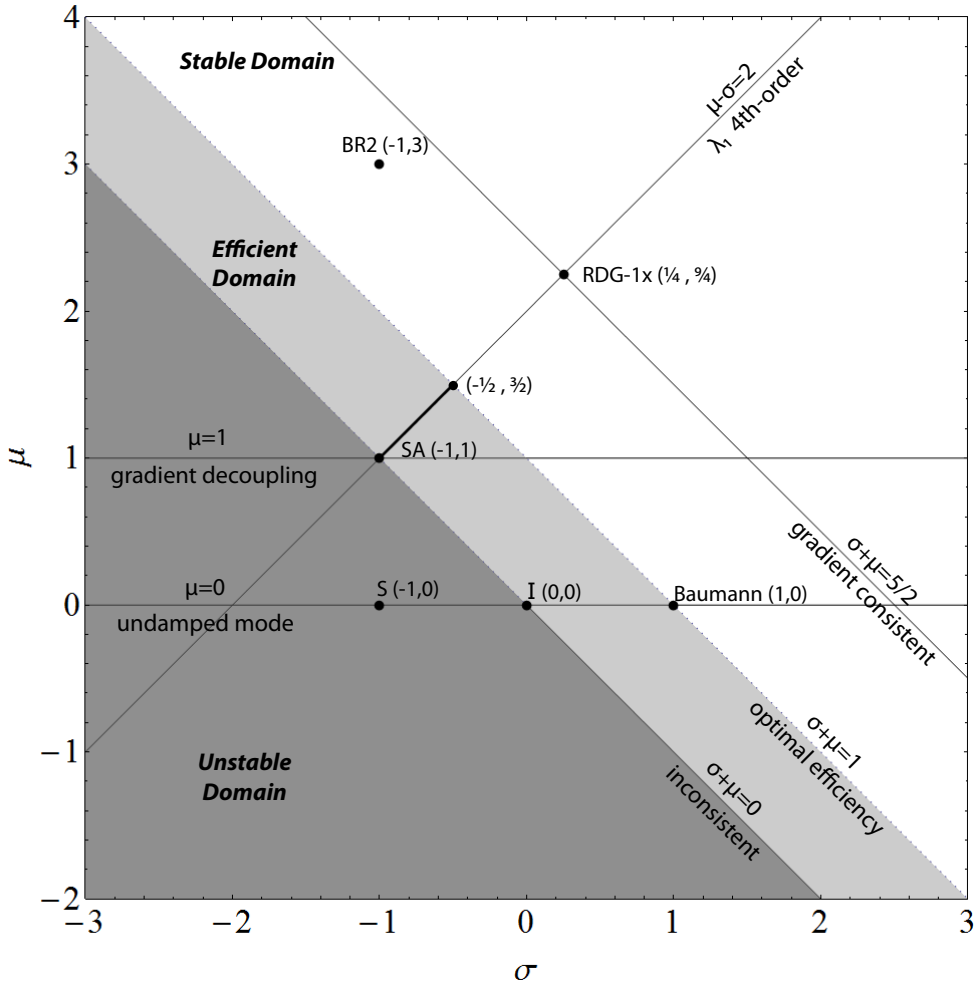

Figure 3.9: A intuitive map of the (σ, μ)-family. The symbols are defined as follow: "S" is the symmetric scheme, "SA" is the symmetric/Arnold scheme, "I" is the inconsistent scheme, "BR2" is Bassi-Rebay 2. The dark region indicates instability, the light gray region represents efficient and stable schemes with the largest Von Neumann number (VNN), and the white region designates the stable domain.

ensure λ_2 is negative,

$$\sigma + \mu > 0. \tag{3.83}$$

In the case where $\sigma + \mu = 0$, the eigenvalues of $M_{(\sigma-\mu)}$ become

$$\lambda_1 = -2\mu\,(1 - \cos\beta)\,, \tag{3.84}$$

$$\lambda_2 = -6\,(1 - \cos\beta)\,. \tag{3.85}$$

It is clear from λ_1 that the scheme will be stable if $\mu \geq 0$, and this is the reason why schemes below $\mu = 0$ on the line $\sigma + \mu = 0$ are unstable.

The second line, $\sigma - \mu = -2$, is locus of 4th-order schemes determined from looking at the error of λ_1, $\lambda_{error} = |\lambda_1 + \beta^2|$. We define the order of accuracy of a scheme to be O.O.A. $= \frac{\lambda_{error}}{\beta^2}$. First, we look for 2nd-order schemes on the $\sigma - \mu$ plane by enforcing the following conditions,

$$C_0^{\lambda_1} = 0,\, C_2^{\lambda_1} = 1. \;\; \text{(O.O.A.=2)} \tag{3.86}$$

The solution turns out to be the entire $\sigma - \mu$ plane. Next, we look for 4th-order schemes by imposing one additional condition,

$$C_0^{\lambda_1} = 0,\, C_2^{\lambda_1} = 1,\, C_4^{\lambda_1} = 0, \quad \text{(O.O.A.=4)} \tag{3.87}$$

The solution is the locus of points along the line, $\mu - \sigma = 2$. If we delve even further and look for a 6th-order scheme with an additional condition,

$$C_0^{\lambda_1} = 0,\, C_2^{\lambda_1} = 1,\, C_4^{\lambda_1} = 0,\, C_6^{\lambda_1} = 0, \quad \text{(O.O.A.=6)} \tag{3.88}$$

we obtain a unique solution, $(\sigma,\, \mu) = \left(\frac{1}{9},\, \frac{19}{9}\right)$. Imposing any further conditions on $C_j^{\lambda_1}$ beyond 6th-order will generate an ill-conditioned system of equations. Although it seems fantastic to have an infinite number of potential 4th-order schemes (with one being 6th-order), looking at just one eigenvalue does not paint the whole picture. The conditions listed here are merely necessary but not sufficient conditions for 4th-order or higher schemes. Very often the second eigenvalue introduces an initial projection error of the solution that cannot be immediately damped, hence contaminating the high-order evolution owing to the first eigenvalue. We perform a numerical study of the projection error in a later section.

The third line, $\sigma + \mu = \frac{5}{2}$, is the gradient-consistent line. We have shown that

all schemes on the (σ, μ)-plane are at least 2nd-order accurate in the cell average; however, that does not guarantee the accuracy of the first gradient. One would expect the gradient update to approximate the third-order diffusion equation,

$$\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right) = D\frac{\partial^3 u}{\partial x^3}; \tag{3.89}$$

therefore, if $\overline{\Delta u}$ is to remain an accurate approximation of the gradient of the solution, it must satisfy

$$\frac{1}{\Delta x}\frac{\partial \overline{\Delta u}_j}{\partial t} = D\left(\frac{\partial^3 u}{\partial x^3}\right)_j + O\left(\Delta x^2\right). \tag{3.90}$$

The actual update equation, as seen from Eqn 3.71, is

$$\begin{aligned}\frac{1}{\Delta x}\frac{\partial \overline{\Delta u}_j}{\partial t} &= D\left\{6\left(\sigma+\mu\right)\left(T^1 - T^{-1}\right)\bar{u}_j\right\} \\ &+ D\left\{3\left(T^1 - 2I + T^{-1}\right) - 3\left(\sigma+\mu\right)\left(T^1 + 2I + T^{-1}\right)\right\}\overline{\Delta u}_j \end{aligned} \tag{3.91}$$

Using the following identities,

$$T^1 - T^{-1} = \left(T^1 - I\right) + \left(I - T^{-1}\right), \tag{3.92}$$

$$T^1 + 2I + T^{-1} = \left(T^1 + I\right) + \left(I + T^{-1}\right), \tag{3.93}$$

and together with

$$[u]_{j+\frac{1}{2}} = \left(T^1 - I\right)\bar{u}_j + \frac{1}{2}\left(T^1 + I\right)\overline{\Delta u}_j, \tag{3.94}$$

$$[u]_{j-\frac{1}{2}} = \left(I - T^{-1}\right)\bar{u}_j + \frac{1}{2}\left(I + T^{-1}\right)\overline{\Delta u}_j. \tag{3.95}$$

The gradient update equation simplifies to

$$\frac{1}{\Delta x}\frac{\partial \overline{\Delta u}_j}{\partial t} = D\left[6\left(\sigma+\mu\right)\left([u]_{j-\frac{1}{2}} + [u]_{j+\frac{1}{2}}\right) + 3\left(T^1 - 2I + T^{-1}\right)\overline{\Delta u}_j\right]. \tag{3.96}$$

Here we borrow from the theory of recovery to approximate the jump terms. From the results of recovery in the piecewise-linear section in Eqn 3.8, we know the interface jump of the solution scales with the third derivative. Let $b_3 \approx \frac{\partial^3 u}{\partial x^3}$, $a_0 = \bar{u}$, and $a_1 = \frac{\overline{\Delta u}}{2}$, then the fourth component in Eqn 3.8 becomes,

$$\frac{1}{\Delta x^3}[u] = -\frac{1}{15}\frac{\partial^3 u}{\partial x^3} + O\left(\Delta x^2\right). \tag{3.97}$$

The central-difference operator applied to the first gradient is also an approximation to the third derivative,

$$\frac{1}{\Delta x^3}\left(T^1 - 2I + T^{-1}\right)\overline{\Delta u}_j = \frac{\partial^3 u}{\partial x^3} + O\left(\Delta x^2\right). \tag{3.98}$$

Inserting the two results above into the update equation yields

$$\frac{1}{\Delta x}\frac{\partial \overline{\Delta u}_j}{\partial t} = D\left[-\frac{4}{5}(\sigma+\mu)+3\right]\left(\frac{\partial^3 u}{\partial x^3}\right)_j + O\left(\Delta x^2\right). \tag{3.99}$$

Upon comparing this with Eqn 3.90 we conclude that gradient consistency requires $\sigma + \mu = \frac{5}{2}$.

The fourth line, $\sigma + \mu = 1$, along with $\sigma + \mu = 0$, bounds the region of maximum speed. This region is stable and the maximum eigenvalue is always equal to -12. We show this numerically later.

The fifth line, $\mu = 1$, is the gradient-decoupling line. Inserting $\mu = 1$ into Eqn 3.71 causes the upper right element of $M_{(\sigma-\mu)}$ to become zero, which implies the update of the cell average does not involve the first gradient (hence the name of this line). Although the update of the gradient involves the cell average, eigenvector analysis reveals a complete decoupling in the update of the two solution coefficients.

The sixth line, $\mu = 0$, is the line of undamped mode for $\beta = \pi$. Specifically, for this mode, the update matrix in Eqn. 3.71 becomes

$$M_{(\sigma-\mu)} = \begin{pmatrix} -4\mu & 0 \\ 0 & -12 \end{pmatrix}, \tag{3.100}$$

where the first eigenvalue becomes zero for $\mu = 0$. In the numerical-experiment section later, we show this undamped mode causes convergence to a steady state to slow down dramatically.

This completes our analysis of the first eigenvalue. Next, we look at the initial projection error based on the analysis of eigenvectors. The eigenvector analysis is similar to that of the previous section; the magnitude of the C_2-switch is presented in Table 3.2 in conjunction with the numerical results. We defer the discussion of the results to a later section. Before we dive into the numerical details, let us first put existing schemes and new schemes into perspective.

Interesting points on the (σ, μ)-map

We are now in a position to identify characteristics of some of the existing DG diffusion schemes based on their position on the (σ, μ)-map in Figure 3.9. First, the inconsistent (I) scheme, with coordinates (0, 0), is stable but inconsistent. The symmetric (S) scheme at $(-1, 0)$ is unstable without Arnold's penalty term. The symmetric/Arnold (SA) scheme at $(-1, 1)$ is stabilized by the penalty term. Although SA lies on the 4th-order evolution line, it is only a 2nd-order scheme. The newer Baumann scheme at (1, 0) is stable without a penalty term and lies on the undamped-mode line. Notice all four schemes are only 2nd-order accurate. These are the results of the first two decades of research into diffusion for DG; at this point, it is not even clear when the inclusion of the penalty term is necessary for stability.

We look at two other schemes that are derived from Eqn 3.30, and then cast into penalty form as in Eqn 3.71. First, the modified Bassi-Rebay 2 scheme based on Legendre polynomials (see Chapter 1) is the point $(-1, 3)$, which is 2nd-order accurate and stable. In this version of BR2, the correction function shares the average value at the interface, and both moments of the corresponding cell. Next, we introduce a new recovery method, RDG-1x$\bar{f}$. In Eqn 3.30, the solution in the surface integral is replaced with the recovered function f, and the solution in the volume integral is replaced with an average recovered function $\bar{f}$,

$$\int_\Omega v u_t \, dx = D \oint_{\partial\Omega} v f_x \, d\partial\Omega - D \int_\Omega v_x \bar{f}_x \, dx. \tag{3.101}$$

At the left and right interfaces of Ω_j lie the recovered functions $f_{j-1,j}$ and $f_{j,j+1}$, respectively. The average recovered function is defined as,

$$\bar{f}_j = \frac{f_{j-1,j} + f_{j,j+1}}{2}. \tag{3.102}$$

RDG-1x$\bar{f}$ is identified by the coordinates $\left(\frac{1}{4}, \frac{9}{4}\right)$ in the(σ, μ)-family, which lies on both the 4th-order accurate and gradient-consistent lines. Table 3.1 shows the results of the numerical Fourier analysis of RDG-1x$\bar{f}$ based on the linear diffusion equation. Unfortunately the scheme is unstable for $p \geq 3$. As the order of the recovered function increases, the oscillation far away from the center of the interface also increases. This leads to undesirable swings in the derivative. RDG-1x$\bar{f}$ remains an experimental scheme intended for comparison with existing penalty methods.

p	Min $(Re\ (\lambda))$	Max $(Re\ (\lambda))$	Max $(Im\ (\lambda))$	O.O.A.
1	-30.0	0.0	0.0	4
2	-125.2	0.0	0.0	6
3	-423.4	26.1	0.1	N/A
4	-1280.2	404.8	0.0	N/A

Table 3.1: Numerical Fourier analysis of RDG-1x$\bar{f}$ show the scheme to be unstable for $p \geq 3$ due to positive eigenvalues. RDG-1x$\bar{f}$ is an experimental scheme.

Numerical results for various members of the (σ, μ)-family

This section introduce our famous guinea pig test case for determining the order of accuracy of the diffusion operator, and this test case is used frequently throughout this thesis. We introduce the linear diffusion equation with a source term,

$$u_t = u_{xx} + 4\pi^2 \sin(2\pi x), \tag{3.103}$$

with the following boundary conditions on the domain $x \in [0, 1]$,

$$u(0) = 1, \tag{3.104}$$

$$u(1) = 0. \tag{3.105}$$

The steady state solution is

$$u(x, \infty) = 1 - x + \sin(2\pi x). \tag{3.106}$$

We coupled the (σ, μ)-spatial discretization with a Runge-Kutta (RK) time-marching scheme. Note the order of the RK scheme is irrelevant in this steady-state problem. The maximum allowable time-step for RK schemes is determined by the Von Neumann number $\frac{D\Delta t}{\Delta x^2}$ (VNN). The initial solution is the same as the exact solution. We let the scheme march and slowly deviate from the exact solution; the resulting difference between the initial and final solutions is considered the error. The complete numerical results for all the test cases are found in Appendix B. Table 3.2 provides a summary of the order of accuracy of each scheme and the total CPU time required for the numerical solution to converge with convergence criterion of $\varepsilon < 1e-16$ on the 320-cell grid. We define ε in the L_∞-norm,

$$\varepsilon = \text{Max} \left| u_j^{n+1} - u_j^n \right|. \tag{3.107}$$

Numerical results show that only the recovery scheme on the 4th-order-evolution line is indeed 4th-order accurate in the L_2-error norm. The rest of the schemes on the line are only 3rd-order accurate. Another exception on the 4th-order line is the symmetric/Arnold scheme which is only 2nd-order. Other schemes lying elsewhere are 2nd-order. The schemes $\left(-\frac{99}{100}, \frac{101}{100}\right)$ and $\left(-\frac{1}{2}, \frac{3}{2}\right)$ are on the thick-solid line on the (σ, μ)-map, indicating the highest possible VNN number. Numerical experiment confirms these schemes are of 3rd-order and exhibit good convergence speed. They are able to utilize the maximum VNN of 0.20, making them among the fastest schemes for time-accurate problems. This maximum VNN is determined by fitting the largest eigenvalue of -12 into the stability domain of a 3rd-order Runge-Kutta method. The recovery method converges at least twice as fast in comparison to other members of the (σ, μ)-family. We conclude that the Fourier analysis of just one eigenvalue provides limited insight to the overall order of accuracy of a scheme. We also determine the initial projection error by looking at the magnitude of C_2 (the weighting coefficient of the bad eigenvector).

The initial projection error really comes from both the good and bad eigenvectors. The cell average of the good eigenvector is exact due to our choice of normalization; however, the average first gradient contains an error. We discovered the error in the gradient of the good eigenvector matches the error in the bad eigenvector as regards to magnitude. Hence it suffices to just show the magnitude of contribution C_2 of the bad eigenvector in Table 3.2. It turns out the gradient in the bad eigenvector of all (σ, μ)-schemes contains an $O\left(\frac{1}{\beta}\right)$ error; this means the order of the projection error is really equal to the order of C_2 subtracted by one (since C_2 is multiplied by the bad gradient). The numerical experiments confirm that the order of accuracy of a scheme is determined by the minimal of the order of λ_1 and order of C_2 minus one.

Type	(σ, μ, VNN)	Minimum λ	λ_1 Order	C_2 Order	$\overline{u}$ Rate	$\overline{\Delta u}$ Rate	Time (s)
RDG-1x$\bar{f}$	$\frac{1}{4}$, $\frac{9}{4}$, 0.08	-30.0	4	6	4	5	40.4
λ_1 6th-order	$\frac{1}{9}$, $\frac{19}{9}$, 0.09	-26.6	6	4	3	3	105.9
BR2	-1, 3, 0.10	-24.0	2	4	2	3	213.0
Baumann	1, 0, 0.20	-12.0	2	4	2	3	1659.2
λ_1 4th-order *	$-\frac{1}{2}$, $\frac{3}{2}$, 0.19	-12.6	4	4	3	3	73.0
λ_1 4th-order *	$-\frac{99}{100}$, $\frac{101}{100}$, 0.20	-12.0	4	4	3	3	89.9
λ_1 4th-order	0, 2, 0.10	-24.0	4	4	3	3	107.5
λ_1 4th-order	$\frac{1}{6}$, $\frac{13}{6}$, 0.08	-28.0	4	4	3	3	108.4
λ_1 4th-order	1, 3, 0.05	-48.0	4	4	3	3	213.1
Symmetric/Arnold	-1, 1, 0.20	-12.0	2	N/A	2	1	106.8
$\mu = 1$	0, 1, 0.20	-12.0	2	N/A	2	3	108.8
$\mu = 1$	1, 1, 0.10	-24.0	2	N/A	2	3	207.3
$\mu = 1$	2, 1, 0.06	-36.0	2	N/A	2	3	338.8
Gradient consistent	$-\frac{3}{7}$, $\frac{41}{14}$, 0.09	-30.0	2	8	2	3	249.3
Gradient consistent	$-\frac{1}{2}$, 3, 0.08	-30.0	2	6	2	3	253.9
Gradient consistent	1, $\frac{3}{2}$, 0.08	-30.0	2	6	2	3	250.0
Random	1, 2, 0.06	-36.0	2	4	2	3	310.2
Random	0, 3, 0.06	-36.0	2	4	2	3	310.9
Random	$\frac{3}{2}$, 1, 0.08	-30.0	2	4	2	3	270.2
Random	-1, 2, 0.18	-13.3	2	4	2	3	125.7

Table 3.2: Stability range and order of error convergence of various schemes of the (σ, μ)-family. The maximum stable Von Neumann number (VNN) is found numerically to the nearest one-hundredths, and the CPU time for numerical convergence is given in seconds. The * symbol indicates schemes lying on the thick solid line of the (σ, μ)-map.

3.3.4 The new $(\sigma,\, \mu,\, \omega)$-family for $p = 1$

In the previous section, we tacitly showed that any DG diffusion scheme can be written in terms of jumps and averages at the interface. For the penalty method in Eqn 3.71, the terms in the RHS only involve the jump in the solution $[u]$, and the average of the solution derivative $< u_x >$. It turns out RDG-2x contains terms not found in the traditional $(\sigma,\, \mu)$-family; for $p = 1$ we see a new term involving $[u_x]$. In order to accommodate the new term, we introduce a new three-parameter family of DG diffusion schemes called the $(\sigma,\, \mu,\, \omega)$-family. We add a new term with coefficient ω to the $(\sigma,\, \mu)$-family,

$$\begin{aligned}
\int_{\Omega_j} v u_t \, dx &= -D\left(< u_x > [v] \mid_{j+\frac{1}{2}} + < u_x > [v] \mid_{j-\frac{1}{2}} \right) - D \int_{\Omega_j} v_x u_x \, dx \\
&+ \sigma D \left(< v_x > [u] \mid_{j+\frac{1}{2}} + < v_x > [u] \mid_{j-\frac{1}{2}} \right) \\
&- \frac{\mu D}{\Delta x} \left([v][u] \mid_{j+\frac{1}{2}} - [v][u] \mid_{j-\frac{1}{2}} \right), \\
&+ \omega D \Delta x \left([v_x]\,[u_x] \mid_{j+\frac{1}{2}} + [v_x]\,[u_x] \mid_{j-\frac{1}{2}} \right). \qquad (3.108)
\end{aligned}$$

The update scheme in matrix form is given by,

$$\frac{\partial}{\partial t} \begin{pmatrix} \bar{u}_j \\ \overline{\Delta u}_j \end{pmatrix} = \frac{D}{\Delta x^2} M\left(T, \sigma, \mu, \omega\right) \begin{pmatrix} \bar{u}_j \\ \overline{\Delta u}_j \end{pmatrix}, \qquad (3.109)$$

$$M\left(T, \sigma, \mu, \omega\right) =$$

$$\begin{pmatrix} \mu\left(T^1 - 2T^0 + T^{-1}\right) & \frac{1-\mu}{2}\left(T^1 - T^{-1}\right) \\ 6\left(\sigma + \mu\right)\left(T^1 - T^{-1}\right) & \left(3 - 12\omega\right)\left(T^1 - 2T^0 + T^{-1}\right) - 3\left(\sigma + \mu\right)\left(T^1 + 2T^0 + T^{-1}\right) \end{pmatrix}.$$

The choice of $(\sigma,\, \mu,\, \omega) = \left(-1,\, \frac{9}{4},\, \frac{1}{12}\right)$ reproduces the RDG-2x, $p = 1$ scheme which is 4th-order accurate. Clearly RDG-2x is a new scheme that is very different from the venerable $(\sigma,\, \mu)$-family. One may play with the idea of exploring the whole $(\sigma,\, \mu,\, \omega)$-space for interesting schemes, but it is clear from the previous section that all but one scheme is of higher order. The beauty of RDG-2x is that it automatically generates the correct coefficients and bilinear terms for the best accuracy. In the next section, we show RDG-2x introduces more and more bilinear terms as p increases.

3.3.5 Bilinear form of RDG

We have succeeded in writing the 1-D form of the RDG-2x scheme of arbitrary order as an expansion in penalty-like terms. On a uniform grid it reads,

$$\begin{aligned}
\frac{1}{D}\int_{\Omega_j} v u_t\, dx \;=\; & -\Big(< u_x > [v]\,|_{j+\frac{1}{2}} + < u_x > [v]\,|_{j-\frac{1}{2}}\Big) \\
& -\Big(< u > [v_x]\,|_{j+\frac{1}{2}} + < u > [v_x]\,|_{j-\frac{1}{2}}\Big) - \int_{\Omega_j} v_x u_x\, dx \\
& + \frac{R_0}{\Delta x}\Big([u]\,[v]\,|_{j+\frac{1}{2}} + [u]\,[v]\,|_{j-\frac{1}{2}}\Big) \\
& + R_1 \Delta x \Big([u_x]\,[v_x]\,|_{j+\frac{1}{2}} + [u_x]\,[v_x]\,|_{j-\frac{1}{2}}\Big) \\
& + R_2 \Delta x \Big([u_{xx}]\,[v]\,|_{j+\frac{1}{2}} + [u_{xx}]\,[v]\,|_{j-\frac{1}{2}}\Big) \\
& + R_3 \Delta x^3 \Big([u_{xxx}]\,[v_x]\,|_{j+\frac{1}{2}} + [u_{xxx}]\,[v_x]\,|_{j-\frac{1}{2}}\Big) \\
& + R_4 \Delta x^3 \Big([u_{xxxx}]\,[v]\,|_{j+\frac{1}{2}} + [u_{xxxx}]\,[v]\,|_{j-\frac{1}{2}}\Big) \\
& + R_5 \Delta x^5 \Big([u_{xxxxx}]\,[v_x]\,|_{j+\frac{1}{2}} + [u_{xxxxx}]\,[v_x]\,|_{j-\frac{1}{2}}\Big) \\
& + R_6 \Delta x^5 \Big([u_{xxxxxx}]\,[v]\,|_{j+\frac{1}{2}} + [u_{xxxxxx}]\,[v]\,|_{j-\frac{1}{2}}\Big) + \ldots \quad (3.110)
\end{aligned}$$

It is seen that the i-th bilinear "penalty" term, with coefficient R_i, contains the jump in the i-th derivative of u and the jump in either v or v_x, depending on whether the index i is even or odd, respectively. For an RDG scheme based on a polynomial space of degree p, the index running from 0 to p, the coefficients R_i are given in Table 3.3 for schemes up to $p = 5$.

p	$\mu = -R_0$	R_1	R_2	R_3	R_4	R_5
0	1					
1	$\frac{9}{4}$	$\frac{1}{12}$				
2	$\frac{15}{4}$	$\frac{3}{64}$	$-\frac{3}{80}$			
3	$\frac{175}{32}$	$\frac{1}{32}$	$-\frac{5}{192}$	$-\frac{1}{6720}$		
4	$\frac{945}{128}$	$\frac{35}{1536}$	$-\frac{5}{256}$	$-\frac{5}{86016}$	$\frac{5}{129024}$	
5	$\frac{4851}{512}$	$\frac{9}{512}$	$-\frac{63}{4096}$	$-\frac{1}{36864}$	$\frac{7}{369640}$	$\frac{1}{9461760}$

Table 3.3: Coefficients of the penalty-like terms in 1-D RDG-2x for $p \leq 5$.

3.4 Numerical results for the "original" RDG-2x

This section is intended to be a guide to implementing RDG-2x with explicit time-marching; one can jump over the implementation details and jump directly to the numerical results section without loss of continuity. The implementation describes everything from the interior scheme to recovery at the domain boundary. Next, we show numerical results for linear advection, linear diffusion, and linear advection-diffusion test cases. The advection operator is the simple upwind scheme, while the diffusion operator is where RDG-2x is applied. The numerical tests are designed to show the difference in order of accuracy between the advection and diffusion operators, and finally show how the two work together. For the linear diffusion test case, we run both time-accurate and steady-state problems to reveal an interesting "infinite-accuracy" property of the RDG-2x scheme.

3.4.1 Recovery at the domain boundary

The concepts described here apply strictly to structured grids only. This section is meant to illustrate the basic principle of enforcing the boundary-recovered function to satisfy the given boundary conditions. Recovery at the boundary is slightly different from interior recovery; however, the concept remains the same in which the boundary-recovered function must satisfy certain moments of the solution near the boundary. For 1-D problems the boundary-recovered function must satisfy a function value or a derivative at the boundary. We introduce two types of boundary-recovered functions: the full boundary-recovered function f_F, and the compact boundary-recovered function f_C.

Full boundary-recovered function

The full boundary-recovered function covers a larger domain, which includes the domain boundary and the first two cells adjacent to the boundary (see left diagram of Figure 3.10). The polynomial order of f_F and interior recovered function are the same. We enforce f_F to satisfy the $p+1$ moments of the first adjacent cell, and the p moments of the next adjacent cell. The last condition is the Dirichlet or Neumann condition. Let Ω_1 be the first cell adjacent to the domain boundary, and Ω_2 be the

next adjacent cell. The recovery equations for f_F are

$$\int_{\Omega_1} v_i \, f_F \, dx = \int_{\Omega_1} v_i \, u_1 \, dx \quad \text{for } i = 0, .., p, \tag{3.111}$$

$$\int_{\Omega_2} v_i \, f_F \, dx = \int_{\Omega_2} v_i \, u_2 \, dx \quad \text{for } i = 0, .., p-1, \tag{3.112}$$

$$f_F = C_{\text{Dir}} \quad \text{or} \quad \frac{df_F}{dx} = C_{\text{Neu}}, \tag{3.113}$$

where C_{Dir} and C_{Neu} are prescribed values. f_F is expanded in terms of r-monomials

$$f_F(r) = \sum_{i=0}^{2p+1} b_i r^i, \tag{3.114}$$

where the b_i coefficients are the unknown, and $r = 0$ is the location of the domain boundary. For the left boundary domain, we seek a f_F that spans the union of Ω_1 and Ω_2 such that Ω_1 belongs to $r \in [0, 1]$ and Ω_2 belongs to $r \in [1, 2]$. For the right boundary domain, we seek a f_F that spans the union of Ω_1 and Ω_2 such that Ω_1 belongs to $r \in [-1, 0]$ and Ω_2 belongs to $r \in [-2, -1]$. With this specific coordinate system, $\frac{d}{d\xi}$ is equal to $\frac{d}{dr}$.

The solution gradients based on the full boundary-recovered functions with Dirichlet condition on the left domain boundary are given by

$$\frac{df_F}{dr}(0) = \frac{99}{14} a_{0,\Omega_1} - \frac{40}{7} a_{1,\Omega_1} + \frac{3}{14} a_{0,\Omega_2} - \frac{51}{7} C_{\text{Dir}}, \quad p = 1, \tag{3.115}$$

$$\frac{df_F}{dr}(0) = \frac{951}{58} a_{0,\Omega_1} - \frac{363}{29} a_{1,\Omega_1} + \frac{336}{29} a_{2,\Omega_1} - \frac{45}{58} a_{0,\Omega_2} + \frac{13}{29} a_{1,\Omega_2} - \frac{453}{29} C_{\text{Dir}}, \quad p = 2, \tag{3.116}$$

$$\begin{aligned} \frac{df_F}{dr}(0) &= \frac{10237}{412} a_{0,\Omega_1} - \frac{22735}{824} a_{1,\Omega_1} + \frac{14197}{824} a_{2,\Omega_1} - \frac{2112}{103} a_{3,\Omega_1} \\ &+ \frac{875}{412} a_{0,\Omega_2} - \frac{1341}{824} a_{1,\Omega_2} + \frac{629}{824} a_{2,\Omega_2} - \frac{2778}{103} C_{\text{Dir}}, \quad p = 3. \end{aligned} \tag{3.117}$$

The function values based on the full boundary-recovered functions with Neumann condition on the right domain boundary are given by,

$$f_F(0) = \frac{33}{34} a_{0,\Omega_1} + \frac{40}{51} a_{1,\Omega_1} + \frac{1}{34} a_{0,\Omega_2} + \frac{7}{51} C_{\text{Neu}}, \quad p = 1, \tag{3.118}$$

$$f_F(0) = \frac{317}{302}a_{0,\Omega_1} + \frac{121}{151}a_{1,\Omega_1} + \frac{112}{151}a_{2,\Omega_1} - \frac{15}{302}a_{0,\Omega_2} - \frac{13}{453}a_{1,\Omega_2} + \frac{29}{453}C_{\text{Neu}}, \quad p = 2, \tag{3.119}$$

$$\begin{aligned} f_F(0) &= \frac{10237}{11112}a_{0,\Omega_1} + \frac{22735}{22224}a_{1,\Omega_1} + \frac{14197}{22224}a_{2,\Omega_1} + \frac{352}{463}a_{3,\Omega_1} \\ &+ \frac{875}{1112}a_{0,\Omega_2} + \frac{447}{7408}a_{1,\Omega_2} + \frac{629}{22224}a_{2,\Omega_2} + \frac{103}{2778}C_{\text{Neu}}, \quad p = 3. \end{aligned} \tag{3.120}$$

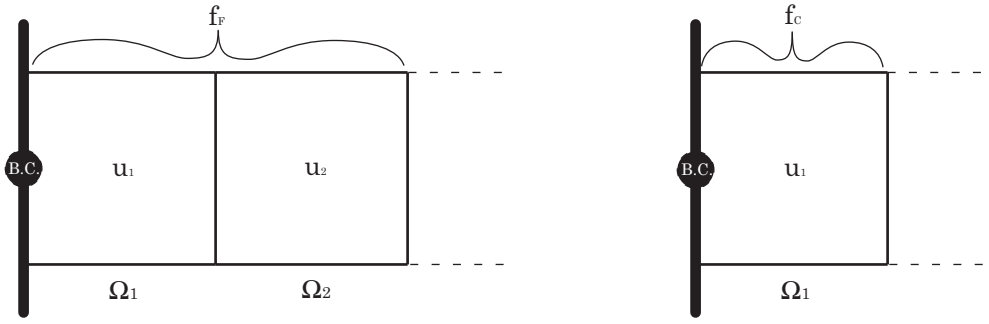

Figure 3.10: Stencils of full boundary-recovered function (left) and compact boundary-recovered function (right). The thick solid line indicates the domain boundary, and B.C. stands for boundary condition, which can be Dirichlet or Neumann.

Compact boundary-recovered function

As the name implies the compact boundary-recovered function has a smaller domain than its cousin. f_C only involves the domain boundary and the first cell adjacent to the boundary (see right diagram of Figure 3.10). The polynomial order of f_C is always p lower than the interior recovered function f,

$$f_C(r) = \sum_{i=0}^{p+1} b_i r^i, \tag{3.121}$$

where the b_i coefficients are the unknown, and $r = 0$ is the location of the domain boundary. We use the same coordinate system as above. Let Ω_1 be the first cell adjacent to the domain; the recovery equations for f_C are

$$\int_{\Omega_1} v_i \, f_F \, dx = \int_{\Omega_1} v_i \, u_1 \, dx \quad \text{for } i = 0, .., p, \tag{3.122}$$

$$f_C = C_{\text{Dir}} \quad \text{or} \quad \frac{df_C}{dx} = C_{\text{Neu}}, \tag{3.123}$$

The solution derivatives based on the compact boundary-recovered functions with Dirichlet condition on the left domain boundary are given by

$$\frac{df_C}{dr}(0) = 6a_{0,\Omega_1} - 4a_{1,\Omega_1} - 6C_{\text{Dir}}, \quad p = 1, \tag{3.124}$$

$$\frac{df_C}{dr}(0) = 12a_{0,\Omega_1} - 10a_{1,\Omega_1} + 6a_{2,\Omega_1} - 12C_{\text{Dir}}, \quad p = 2, \tag{3.125}$$

$$\frac{df_C}{dr}(0) = 20a_{0,\Omega_1} - 18a_{1,\Omega_1} + 14a_{2,\Omega_1} - 8a_{3,\Omega_1} - 20C_{\text{Dir}}, \quad p = 3. \tag{3.126}$$

The function values based on the compact boundary-recovered functions with Neumann condition on the right domain boundary are given by

$$f_C(0) = a_{0,\Omega_1} + \frac{2}{3}a_{1,\Omega_1} + \frac{1}{6}C_{\text{Neu}}, \quad p = 1, \tag{3.127}$$

$$f_C(0) = a_{0,\Omega_1} + \frac{5}{6}a_{1,\Omega_1} + \frac{1}{2}a_{2,\Omega_1} + \frac{1}{12}C_{\text{Neu}}, \quad p = 2, \tag{3.128}$$

$$f_C(0) = a_{0,\Omega_1} + \frac{9}{10}a_{1,\Omega_1} + \frac{7}{10}a_{2,\Omega_1} + \frac{2}{5}a_{3,\Omega_1} + \frac{1}{20}C_{\text{Neu}}, \quad p = 3. \tag{3.129}$$

Although we demonstrate later that the compact version is vastly inferior on a Cartesian grid, the compact version is definitely more practical on an unstructured grid; this is left as a future research topic.

3.4.2 Linear diffusion

The numerical results for linear diffusion equation is divided into three sections: steady-state problem with periodic boundary condition, time-accurate problem with periodic boundary condition, and steady-state problem with mixed boundary conditions. The steady-state problem with periodic boundary condition is designed to illustrate the spatial accuracy of the RDG-2x interior scheme. The time-accurate problem with periodic boundary condition reveals the combined accuracy of RDG-2x with Runge-Kutta time-marching. Finally, the steady-state problem with mixed

boundary conditions shows the results of the implementation of boundary-recovered functions. We like to make a note that the total CPU time for each scheme is not released because the time scale is too short for any meaningful comparison.

Steady-state problem with periodic boundary condition

We solve the linear diffusion equation with a source term,

$$u_t = u_{xx} + 4\pi^2 \sin(2\pi x), \tag{3.130}$$

on the domain $x \in [0, 1]$. The initial solution and final solution are given by

$$u(x,t) = \sin(2\pi x). \tag{3.131}$$

The solution is marched forward with a Runge-Kutta scheme until the change in the solution, measured in the L_∞-norm, is less than $\epsilon \leq 1e-15$. Note that the order of the Runge-Kutta scheme is irrelevant in a steady-state problem. Table 3.4 provides the results for $p = 1, 2$ and 3. The results for $p = 1$ confirm the findings from our earlier Fourier analysis. RDG-2x ($p = 1$) is 4th-order accurate; however, the gradient is 5th-order accurate! For $p \geq 2$, we observe a trend in which the lower $p-1$ moments are of infinite accuracy. We reserve this terminology to describe solution coefficients that are solved with near computer-zero error. During our earliest investigations we repeated the calculations with a quadruple-precision code with computer zero of 1×10^{-32}, and the results confirm the infinite-accuracy behavior of the RDG-2x scheme for $p \geq 2$. This special property, encountered only for steady 1-D solutions of the linear diffusion equation, is explained in [43]. The order of accuracy is better revealed in the following time-accurate test case.

Time-accurate problem with periodic boundary conditions

We solve the linear diffusion equation,

$$u_t = u_{xx}, \tag{3.132}$$

on the domain $x \in [0, 1]$. The solution for any $t \geq 0$ is given by

$$u(x,t) = \sin(2\pi x)\, e^{-4\pi^2 t}. \tag{3.133}$$

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	2.88×10^{-4}	1.35×10^{-4}						
	20	1.88×10^{-5}	4.37×10^{-6}			3.9	4.9		
	40	1.19×10^{-6}	1.38×10^{-7}			3.9	4.9		
	80	7.47×10^{-8}	4.31×10^{-9}			3.9	5.0		
	160	4.64×10^{-9}	1.34×10^{-10}			4.0	5.0		
	320	1.70×10^{-10}	3.01×10^{-12}			4.7	5.4		
2	4	9.28×10^{-10}	1.13×10^{-4}	3.21×10^{-4}					
	8	7.32×10^{-13}	9.12×10^{-7}	5.97×10^{-6}		*	6.9	5.7	
	16	5.16×10^{-13}	7.16×10^{-9}	9.69×10^{-8}		*	6.9	5.9	
	32	6.09×10^{-13}	5.60×10^{-11}	1.53×10^{-9}		*	7.0	5.9	
	64	6.06×10^{-15}	4.37×10^{-13}	2.39×10^{-11}		*	7.0	6.0	
	128	2.69×10^{-15}	3.43×10^{-15}	3.74×10^{-13}		*	6.9	6.0	
3	4	8.58×10^{-15}	7.11×10^{-15}	1.37×10^{-6}	3.17×10^{-6}				
	8	2.86×10^{-14}	1.13×10^{-14}	7.36×10^{-9}	7.42×10^{-9}	*	*	7.5	8.7
	12	6.51×10^{-14}	1.71×10^{-14}	3.05×10^{-10}	2.00×10^{-10}	*	*	7.8	8.9
	16	4.25×10^{-14}	8.34×10^{-15}	3.12×10^{-11}	1.52×10^{-11}	*	*	7.9	8.9
	20	6.11×10^{-15}	9.83×10^{-16}	5.28×10^{-12}	2.06×10^{-12}	*	*	7.9	8.9
	24	1.15×10^{-15}	2.28×10^{-16}	1.24×10^{-12}	4.00×10^{-13}	*	*	7.9	8.9

Table 3.4: L_2-error of RDG-2x scheme for steady-state problem with periodic boundary condition. * stands for undetermined order of accuracy. The extremely high accuracy of the cell average of the $p = 2$ scheme, and both cell average and first gradient of the $p = 3$ scheme are referred to as "infinite accuracy".

The final time of the simulation is $t = 0.01$[1]. The solution is marched forward in time with a 3rd, 4th and 5th-order Runge-Kutta method (in simple notation: RK3, RK4, and RK5) for RDG-2x schemes with $p = 1$, 2, and 3, respectively. The numerical time step is determined from linear stability, or the Von Neumann number (VNN),

$$\text{VNN} = \frac{\Delta t}{\Delta x^2}. \tag{3.134}$$

Since Δt scales with Δx^2, an n-th order of accuracy in time scheme will achieve $2n$-th order of accuracy in space. For example, if we naively use a 3rd-order RK with RDG-2x ($p = 2$), the combined scheme only achieves the 6th-order of accuracy. A 4th-order RK must be used to unleash the full potential of RDG-2x ($p = 2$). The VNN used for RK3, RK4, and RK5 are 0.15, 0.08, and 0.04, respectively. Table 3.5 provides the results for the time-accurate problem. Again, RDG-2x ($p = 1$) obtains the 4th-order of accuracy; however this time around, RDG-2x ($p = 2$) and ($p = 3$) achieve the 8th and 10th-order of accuracy, respectively. For odd p we observe that the first gradient is of higher order of accuracy than the cell average. Now that the interior RDG-2x scheme is validated; we look at the boundary schemes next.

Steady-state problem with mixed boundary condition

We solve the same problem as in the $\sigma - \mu$ section. Consider the linear diffusion equation with a source term,

$$u_t = u_{xx} + 4\pi^2 \sin(2\pi x), \tag{3.135}$$

on the domain $x \in [0, 1]$. The initial solution and final solution are given by

$$u(x, t) = \sin(2\pi x) - x + 1. \tag{3.136}$$

We enforce a Dirichlet boundary condition on the left, and a Neumann boundary condition on the right,

$$u(0) = 1, \tag{3.137}$$

$$u_x(1) = 2\pi - 1. \tag{3.138}$$

[1]For accurate numerical results, the final simulation time must end exactly at $t = 0.01$. Even if the final time is off by $1e-8$, the order of accuracy of the scheme reduces to 2nd-order.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	7.32×10^{-5}	5.40×10^{-5}						
	20	4.96×10^{-6}	1.73×10^{-6}			3.8	4.9		
	40	3.16×10^{-7}	5.46×10^{-8}			3.9	4.9		
	80	1.98×10^{-8}	1.71×10^{-9}			4.0	5.0		
	160	1.24×10^{-9}	5.34×10^{-11}			4.0	5.0		
	320	7.75×10^{-11}	1.67×10^{-12}			4.0	5.0		
2	4	1.70×10^{-5}	1.18×10^{-4}	1.79×10^{-4}					
	8	4.05×10^{-8}	8.19×10^{-7}	4.05×10^{-6}		8.7	7.1	5.4	
	16	5.44×10^{-11}	4.87×10^{-9}	6.57×10^{-8}		9.5	7.3	5.9	
	32	2.09×10^{-13}	3.78×10^{-11}	1.03×10^{-9}		8.0	7.0	6.0	
	64	1.49×10^{-14}	2.94×10^{-13}	1.61×10^{-11}		3.8	7.0	6.0	
	128	5.61×10^{-14}	9.87×10^{-16}	2.52×10^{-13}		-1.9	8.2	6.0	
3	4	2.91×10^{-8}	1.94×10^{-8}	9.68×10^{-7}	2.25×10^{-6}				
	8	3.30×10^{-11}	1.13×10^{-11}	5.04×10^{-9}	5.07×10^{-9}	9.7	10.7	7.5	8.7
	12	5.90×10^{-13}	1.38×10^{-13}	2.07×10^{-10}	1.36×10^{-10}	9.9	10.8	7.8	8.9
	16	3.35×10^{-14}	5.89×10^{-15}	2.11×10^{-11}	1.03×10^{-11}	9.9	10.9	7.9	8.9
	20	3.74×10^{-15}	5.85×10^{-16}	3.57×10^{-12}	1.39×10^{-12}	9.8	10.3	7.9	8.9
	24	8.10×10^{-16}	1.84×10^{-16}	8.34×10^{-13}	2.70×10^{-13}	8.3	6.3	7.9	8.9

Table 3.5: L_2-error of RDG-2x scheme for time-accurate problem with periodic boundary condition.

Table 3.6 and 3.7 show the results for RDG-2x with full and compact boundary-recovered function, respectively. The first major difference between the two is the VNN. We use RK3, RK4, and RK5 for $p = 1$, 2, and 3, respectively. RDG-2x using full boundary-recovered function is stable using the maximum VNN of 0.08, 0.02, and 0.01 for $p = 1$, 2, and 3, respectively, while the compact boundary version uses a VNN of 0.12, 0.04, and 0.015 for $p = 1$, 2, and 3, respectively. Although the VNN is not important in a steady-state problem, the results imply that the compact version is faster than the full version for time-accurate problems, albeit at a significant loss in accuracy of half an order or more. Using the full boundary-recovered function brings back the high order of accuracy as found in the previous experiments. It is interesting to see that both compact and full boundary-recovered schemes display the infinite-accuracy phenomenon for both $p = 2$ and 3.

Important observations from numerical experiments

We summarize the important findings from the numerical experiments on the linear diffusion equation. RDG-2x is an extremely accurate spatial discretization of the diffusion operator; the order of accuracy (at least in the cell average) is $3p+1$ for odd p and $3p+2$ for even p. A matching high-order time marching scheme must be used to employ the full potential of RDG-2x. For recovery next to the domain boundary, we demonstrated two methods, full boundary-recovered function and compact boundary-recovered function. The full version is more expensive, but compatible with the interior scheme. The compact version is cheaper, but fairly inaccurate. High-order schemes are extremely delicate; every component from the interior scheme, boundary scheme, to the time-marching scheme must be working at the same level of precision to achieve high order of accuracy.

3.4.3 Linear advection-diffusion

The linear advection-diffusion equation contains two types of physics which require completely different numerical approaches. Our main goal is to use RDG-2x for the diffusion component and see how well it works with the advection counterpart in terms of accuracy and stability. The linear advection component is discretized with an upwind scheme; hence, we call this scheme upwind-RDG-2x. We solve the linear advection of the form

$$u_t + au_x = \mu u_{xx}, \tag{3.139}$$

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	2.83×10^{-4}	1.67×10^{-4}						
	20	1.88×10^{-5}	4.93×10^{-6}			3.9	5.0		
	40	1.19×10^{-6}	1.47×10^{-7}			3.9	5.0		
	80	7.46×10^{-8}	4.46×10^{-9}			4.0	5.0		
	160	4.45×10^{-9}	1.37×10^{-10}			4.0	5.0		
	320	6.42×10^{-11}	2.43×10^{-12}			6.1	5.8		
2	4	9.97×10^{-10}	9.09×10^{-5}	3.71×10^{-4}					
	8	8.82×10^{-12}	7.87×10^{-7}	6.10×10^{-6}		6.9	6.8	5.9	
	16	7.76×10^{-12}	6.65×10^{-9}	9.71×10^{-8}		0.5	6.8	5.9	
	32	8.22×10^{-13}	5.39×10^{-11}	1.53×10^{-9}		2.9	6.9	5.9	
	64	4.39×10^{-15}	4.29×10^{-13}	2.39×10^{-11}		7.6	6.9	6.0	
	128	1.25×10^{-16}	3.46×10^{-15}	3.74×10^{-13}		6.0	6.9	6.0	
3	4	5.01×10^{-14}	9.87×10^{-15}	1.09×10^{-6}	3.93×10^{-6}				
	8	1.50×10^{-13}	1.50×10^{-14}	7.20×10^{-9}	8.45×10^{-9}	*	*	7.2	8.8
	12	4.87×10^{-14}	9.66×10^{-15}	3.03×10^{-10}	2.19×10^{-10}	*	*	7.8	9.0
	16	4.64×10^{-14}	8.04×10^{-15}	3.11×10^{-11}	1.63×10^{-11}	*	*	7.9	9.0
	20	8.21×10^{-15}	1.53×10^{-15}	5.28×10^{-12}	2.18×10^{-12}	*	*	7.9	9.0
	24	1.20×10^{-15}	3.22×10^{-16}	1.23×10^{-12}	4.20×10^{-13}	*	*	7.9	9.0

Table 3.6: L_2-error of RDG-2x scheme for steady-state problem with mixed boundary condition using full boundary-recovered function.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	4.62×10^{-4}	7.98×10^{-4}						
	20	4.17×10^{-5}	6.58×10^{-5}			3.4	3.6		
	40	3.61×10^{-6}	5.71×10^{-6}			3.5	3.5		
	80	3.13×10^{-7}	5.02×10^{-7}			3.5	3.5		
	160	2.73×10^{-8}	4.43×10^{-8}			3.5	3.5		
	320	2.39×10^{-9}	3.92×10^{-9}			3.5	3.5		
2	4	9.98×10^{-10}	6.06×10^{-4}	8.88×10^{-4}					
	8	9.72×10^{-13}	1.24×10^{-5}	2.37×10^{-5}		*	5.6	5.2	
	16	2.88×10^{-13}	2.67×10^{-7}	5.44×10^{-7}		*	5.5	5.4	
	32	1.26×10^{-12}	5.85×10^{-9}	1.21×10^{-8}		*	5.5	5.4	
	64	2.63×10^{-13}	1.29×10^{-10}	2.67×10^{-10}		*	5.5	5.5	
	128	1.53×10^{-14}	2.85×10^{-12}	5.88×10^{-12}		*	5.5	5.5	
3	4	4.12×10^{-14}	8.02×10^{-15}	5.39×10^{-5}	1.26×10^{-4}				
	8	1.30×10^{-13}	1.29×10^{-14}	1.40×10^{-6}	3.39×10^{-6}	*	*	5.2	5.2
	12	2.92×10^{-13}	1.93×10^{-14}	1.55×10^{-7}	3.77×10^{-7}	*	*	5.4	5.4
	16	4.98×10^{-13}	2.44×10^{-14}	3.23×10^{-8}	7.83×10^{-8}	*	*	5.4	5.4
	20	1.44×10^{-13}	1.17×10^{-14}	9.50×10^{-9}	2.31×10^{-8}	*	*	5.4	5.4
	24	1.13×10^{-13}	1.08×10^{-14}	3.49×10^{-9}	8.49×10^{-9}	*	*	5.4	5.4

Table 3.7: L_2-error of RDG-2x scheme for steady-state problem with mixed boundary condition using compact boundary-recovered function.

where a is the constant advection speed and μ is the constant diffusion coefficient. The update equation is obtained by testing the equation in both space and time, after suitable integration by parts,

$$\begin{aligned}\iint vu_t\,dxdt &= -a\iint vu_x\,dxdt + \mu\iint vu_{xx}\,dxdt \\ &= -a\int\left(\oint vu\,dx - \int v_x u\,dx\right)dt \\ &\quad +\mu\int\left(\oint (vu_x - v_x u)\,dx + \int v_{xx}u\,dx\right)dt. \end{aligned} \tag{3.140}$$

We discretize by applying the equation to each element Ω_j and replace the global coordinate with a local coordinate,

$$\begin{aligned}\Delta x\int_{\Omega_j} vu_t d\xi dt &= \underbrace{-a\Delta t\int\left([vu]\,|_0^1 - \int_{\Omega_j} v_\xi u\,d\xi\right)d\tau}_{\text{advection}} \\ &\quad + \underbrace{\frac{\mu\Delta t}{\Delta x}\int\left([vu_\xi - v_\xi u]\,|_0^1 + \int_{\Omega_j} v_{\xi\xi}u\,d\xi\right)d\tau}_{\text{diffusion}}. \end{aligned} \tag{3.141}$$

For the advection component, the solution in the surface integral is replaced with the upwind flux $\hat{u}$. The upwind flux for $a > 0$ is given by

$$[vu]\,|_0^1 \approx v\,(1)\,\hat{u}_{j+\frac{1}{2}} - v\,(0)\,\hat{u}_{j-\frac{1}{2}}, \tag{3.142}$$

$$\hat{u}_{j+\frac{1}{2}} = u_j\,(\xi = 1)\,, \tag{3.143}$$

$$\hat{u}_{j-\frac{1}{2}} = u_{j-1}\,(\xi = 1)\,. \tag{3.144}$$

Without going through the details, the final update matrices for $p = 1$ is given by,

$$(\vec{u}_j)_t = \begin{pmatrix} \overline{u}_j \\ \overline{\Delta u_j} \end{pmatrix}_t = \begin{pmatrix} \mathbf{M}_{L,p1} & \mathbf{M}_{C,p1} & \mathbf{M}_{R,p1} \end{pmatrix} \begin{pmatrix} \vec{u}_{j-1} \\ \vec{u}_j \\ \vec{u}_{j+1} \end{pmatrix}, \tag{3.145}$$

$$\mathbf{M}_{L,p1} = \begin{bmatrix} \frac{9r}{4} + \nu & \frac{5r}{4} + \nu \\ -\frac{15r}{4} - 3\nu & -\frac{7r}{4} - 3\nu \end{bmatrix}, \tag{3.146}$$

$$\mathbf{M}_{C,p1} = \begin{bmatrix} -\frac{9r}{2} - \nu & -\nu \\ 3\nu & -\frac{23r}{2} - 3\nu \end{bmatrix}, \tag{3.147}$$

$$\mathbf{M}_{R,p1} = \begin{bmatrix} \frac{9r}{4} & -\frac{5r}{4} \\ \frac{15r}{4} & -\frac{7r}{4} \end{bmatrix}, \tag{3.148}$$

and for $p = 2$,

$$(\vec{u}_j)_t = \begin{pmatrix} \overline{u}_j \\ \overline{\Delta u}_j \\ \overline{\Delta^2 u}_j \end{pmatrix}_t = \begin{pmatrix} \mathbf{M}_{L,p2} & \mathbf{M}_{C,p2} & \mathbf{M}_{R,p2} \end{pmatrix} \begin{pmatrix} \vec{u}_{j-1} \\ \vec{u}_j \\ \vec{u}_{j+1} \end{pmatrix}, \tag{3.149}$$

$$\mathbf{M}_{L,p2} = \begin{bmatrix} \frac{15r}{4} + \nu & \frac{11r}{4} + \nu & \frac{6r}{5} + \nu \\ 3\left(-\frac{11r}{4} - \nu\right) & 3\left(-\frac{31r}{16} - \nu\right) & 3\left(-\frac{61r}{80} - \nu\right) \\ 5\left(\frac{3r}{4} + \nu\right) & 5\left(\frac{5r}{16} + \nu\right) & 5\left(-\frac{9r}{80} + \nu\right) \end{bmatrix}, \tag{3.150}$$

$$\mathbf{M}_{C,p2} = \begin{bmatrix} -\frac{15r}{2} - \nu & -\nu & -\frac{12r}{5} - \nu \\ 3\nu & 3\left(-\frac{57r}{8} - \nu\right) & -3\nu \\ 5\left(-\frac{3r}{2} - \nu\right) & 5\nu & 5\left(-\frac{201r}{40} - \nu\right) \end{bmatrix}, \tag{3.151}$$

$$\mathbf{M}_{R,p2} = \begin{bmatrix} \frac{15r}{4} & -\frac{11r}{4} & \frac{6r}{5} \\ \frac{33r}{4} & -\frac{93r}{16} & \frac{183r}{80} \\ \frac{15r}{4} & -\frac{25r}{16} & -\frac{9r}{16} \end{bmatrix}, \tag{3.152}$$

where $r = \frac{\mu \Delta t}{\Delta x^2}$, and $\nu = \frac{a \Delta t}{\Delta x}$. Here we use r as a short hand notation for VNN, and ν for CFL. The stability condition is given by

$$\Delta t \leq \frac{1}{\frac{1}{\nu_{max}} \frac{a}{\Delta x} + \frac{1}{r_{max}} \frac{\mu}{\Delta x^2}}, \tag{3.153}$$

where the maximum VNN and maximum CFL for $p = 1$ using 3rd-order Runge-Kutta are

$$\nu_{max} = 0.4, \quad r_{max} = 1/6, \tag{3.154}$$

and for $p = 2$ using 4th-order Runge-Kutta are

$$\nu_{max} = 0.27, \quad r_{max} = 1/10. \tag{3.155}$$

These maximum values are obtained numerically via Fourier analysis of the update matrices presented above.

Numerical results for linear advection

Before we delve into the results of the combine advection and diffusion operators, we quickly review the numerical results of the DG upwind operator. We simply obtain the DG upwind scheme by setting $\mu = 0$. Consider the following equation to be solved,

$$u_t = -au_x, \tag{3.156}$$

where $a = 1$ on the domain $x \in [0,\, 1]$. We use a periodic boundary condition and the following solution,

$$u\,(x, t) = \sin\left(2\pi\,(x - at)\right). \tag{3.157}$$

The final time[2] of the simulation is $t = 100$. The CFL numbers for $p = 1$ and 2 follow from the previous section, while the CFL number for $p = 3$ is 0.16 with RK5. The results are presented in Appendix B. The order of error convergence in the L_2-norm is (3, 2) for $\left(\bar{u},\, \overline{\Delta u}\right)$ with $p = 1$, (5, 4, 3) for $\left(\bar{u},\, \overline{\Delta u},\, \overline{\Delta^2 u}\right)$ with $p = 2$, and (7, 8, 5, 4) for $\left(\bar{u},\, \overline{\Delta u},\, \overline{\Delta^2 u},\, \overline{\Delta^3 u}\right)$ with $p = 3$.

Numerical results for linear advection-diffusion

We solve a simple time-dependent linear advection-diffusion problem with the following solution,

$$u\,(t) = e^{-k^2\mu t}\sin k\,(x - at)\,; \quad x \in [0,\, 2\pi]\,. \tag{3.158}$$

Observe the solution becomes zero as $t \to \infty$. Let $t = 0$ and $k = 1$ be the initial condition, and let $t = 100$ be the final time. For a fixed $\mu = 0.01$, we varied the global Péclet number,

$$Pe_G = \frac{al}{\mu}, \tag{3.159}$$

[2]The time must be large enough for the error to develop fully; this is essential to determine the correct order of accuracy of the scheme.

where l is the length of the domain. The numerical results for $p = 1$ and $p = 2$ are given in the two info-graphics below. The numbers represent the order of accuracy of the scheme based on cell refinement measured in the horizontal direction. The diagonal lines represent lines of constant local Péclet number,

$$Pe_L = \frac{a\Delta x}{\mu}. \tag{3.160}$$

The info-graphic for $p = 1$ (see Figure 3.11) shows that the order of accuracy scales with the local Péclet number instead of the global Péclet number. For diffusion-dominated problems, the order of accuracy approaches 4, while for advection-dominated problems, the order of accuracy approaches 3. A similar pattern is not observed for $p = 2$ (see Figure 3.12) since the error levels on the finer grids have reached computer zero.

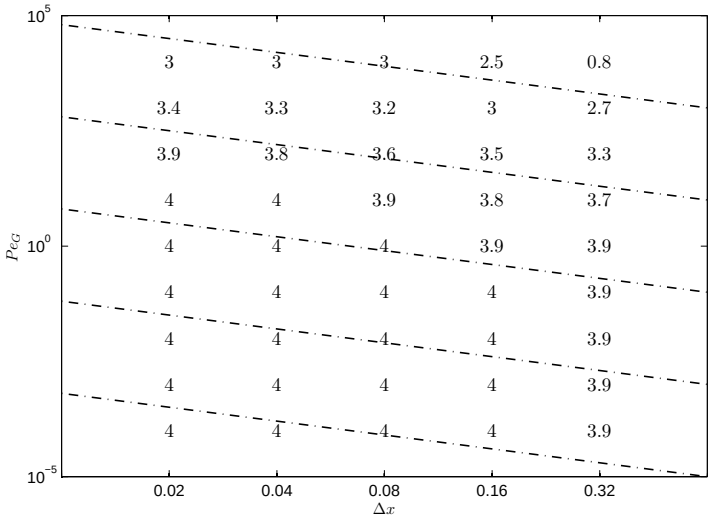

Figure 3.11: Upwind-RDG-2x ($p = 1$) Péclet number study. Numbers indicate the order of accuracy in the L_2-norm. The left axis indicates the value of Pe_G, while the dotted lines indicate contours of constant Pe_L. We observe the order of accuracy of upwind-RDG-2x decreases in the upper right corner where the advection physics dominate, reflecting the order of accuracy of the DG upwind scheme only.

3.5 RDG schemes for variable diffusion coefficient

We move away from the simple constant diffusion coefficient to study a more complex diffusion phenomenon simulated with a variable diffusion coefficient. The variable

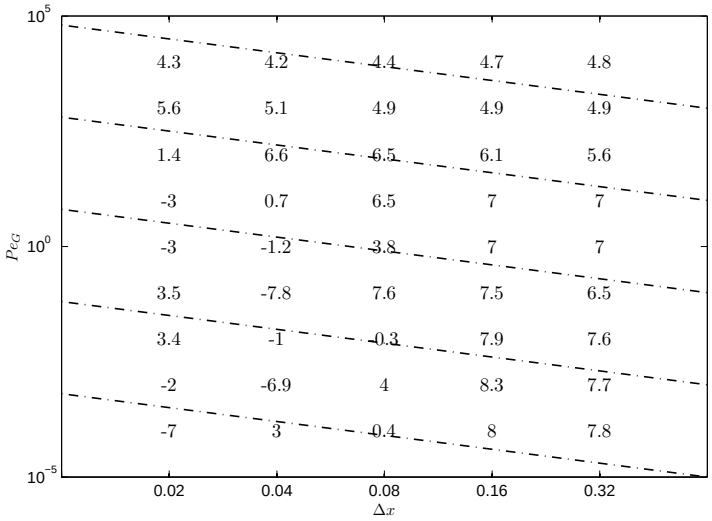

Figure 3.12: Upwind-RDG-2x ($p = 2$) Péclet number study. Numbers indicate the order of accuracy in the L_2-norm. Unfortunately, the rates are not reliable because the diffusion scheme is too accurate and the errors are at the computer-zero level. We are still able to observe that near the top right region where the advection physics dominate, the rate is dominated by the upwind scheme.

diffusion coefficient introduces new complexity which requires a change of game play. This complexity is split into two classes: linear variable and nonlinear variable diffusion coefficients. The difference in the two classes is meant to show that while some simple schemes may work for a linearly varying coefficient (the simpler case), they may not work for a nonlinearly varying coefficient. For readers interested in the ultimate 1-D diffusion scheme that works for both linear and nonlinear diffusion coefficients, please jump directly to the RDG-1x+ section.

The invention of RDG-2x by Van Leer in 2005 came as a blessing in which a simple and elegant DG diffusion method finally became available. As we have shown already, RDG-2x performance in the linear diffusion equation is unparalleled; however, as we deal with more complicated diffusion coefficients (as found in Navier-Stokes equations), the method must also evolve to handle the increasing complexity of the physical system. In 2008, we departed from the twice-integrated-by-parts formulation and focused on the once-integrated form. We already knew that RDG-1x-Naive performs horribly and hence must be improved. In this regard, a new class of schemes called RDG-1x-Smart arrived at the scene. The smart family recreates a more accurate solution within the cell. One of the first members was the RDG-1x$\bar{f}$, which

worked well for $p = 1$ and 2; unfortunately, the scheme becomes unstable for $p \geq 3$. RDG-1x$\bar{f}$ suffers from high-order oscillations at points far away from the center of the Taylor expansion. As a result the interior solution derivative is completely ruined by the oscillations. From the ashes of RDG-1x$\bar{f}$, RDG-1x+ is born in which a new solution enhancement technique is used to minimize oscillations within the cell.

The following section presents the schemes RDG-2x, RDG-1x-Naive, and RDG-1x$\bar{f}$ for a general variable diffusion equation. The concept of applying integration by parts twice to the diffusion operator is not commonly used by the rest of the industry. Admittedly, integration by part twice cannot be applied if the primitive of the diffusion coefficient (see Section 2.5) does not exist in terms of elementry functions (see Section 3.5.4), or not at all (see Section 5.4) as in the system case. The variable scalar diffusion equation is given as,

$$u_t = \left(\mu\left(u\right) u_x\right)_x, \tag{3.161}$$

where $\mu = \mu(u)$ is the variable diffusion coefficient given as a function of the solution u. We get the weak formulation by testing the equation with a test function v in element Ω_j,

$$\int_{\Omega_j} v u_t \, dt = \int_{\Omega_j} v \left(\mu\left(u\right) u_x\right)_x \, dx = \int_{\Omega_j} v \left(M_x\right)_x \, dx, \tag{3.162}$$

where $M_x = M_x\left(u\right)$ may or may not be an analytically integrable function of x. The integrability of M_x dictates the number of times integration by parts can be taken. Applying integration by parts once,

$$\int_{\Omega_j} v u_t \, dt = \left[v M_x\right] - \int_{\Omega_j} v_x M_x \, dx, \tag{3.163}$$

and if M_x is integrable (i.e. $\int M_x \, dx = M$), integration by parts one more time results in

$$\int_{\Omega_j} v u_t \, dt = \left[v M_x\right] - \left[v_x M\right] + \int_{\Omega_j} v_{xx} M \, dx. \tag{3.164}$$

Eqn 3.163 and 3.164 are discretized by various RDG contenders. Two schemes in the once-integrated by parts form are called

$$\int_{\Omega_j} v u_t \, dt = \left[v \tilde{M}_x\right] - \int_{\Omega_j} v_x M_x \, dx \qquad \left[\text{RDG-1x-Naive}\right], \tag{3.165}$$

$$\int_{\Omega_j} v u_t \, dt = \left[v \tilde{M}_x\right] - \int_{\Omega_j} v_x \hat{M}_x \, dx \qquad \left[\text{RDG-1x-Smart}\right], \tag{3.166}$$

where the surface integral terms are now replaced with the recovered function,

$$M_x(u) \approx \tilde{M}_x(f), \tag{3.167}$$

and $\hat{M}_x$ is a special function designed to give the volume integral a better accuracy. The juxtaposition of the terms "naive" and "smart" is meant to emphasize the improvement that the family of RDG-1x-Smart schemes brings. The difference between members of the RDG-1x-Smart family lies in the construction of $\hat{M}_x$; currently there are two ways to construct $\hat{M}_x$: the aforementioned RDG-1x$\bar{f}$ and the new RDG-1x+. RDG-1x+ uses the recovery concept for both diffusion flux and solution enhancement (see section below). As for now we briefly review the RDG-1x$\bar{f}$ scheme. The special term $\hat{M}_x$ appearing in the volume integral of RDG-1x$\bar{f}$ is defined to be the average of the recovered functions on the left and right interfaces of Ω_j,

$$\hat{M_{x,j}} = \frac{1}{2}\left(\tilde{M}_x(f_{j,j+1}) + \tilde{M}_x(f_{j-1,j})\right). \tag{3.168}$$

Notice $f_{j,j+1}$ and $f_{j-1,j}$ have different coordinate system; hence, once must be careful when converting the recovery coordinates into the local coordinate of Ω_j.

The classic twice integrated by parts scheme from the previous sections is given by

$$\int_{\Omega_j} v u_t \, dt = \left[v\tilde{M}_x\right] - \left[v_x\tilde{M}\right] + \int_{\Omega_j} v_{xx} M \, dx \text{ [RDG-2x]}, \tag{3.169}$$

and we apply a backward integration by parts on the integral in the RHS of RDG-2x to arrive at the RDG-3x form,

$$\int_{\Omega_j} v u_t \, dt = \left[v\tilde{M}_x\right] - \left[v_x\tilde{M} - v_x M\right] - \int_{\Omega_j} v_x M_x \, dx \text{ [RDG-3x]}. \tag{3.170}$$

Notice RDG-2x and RDG-3x are mathematically equivalent; however, RDG-3x provides more creative space to deal with a non-integrable M_x that will be explained later.

3.5.1 RDG-1x+ with recovery concept for solution enhancement

Recovery was first conceptualized for DG diffusion; however, there is a much broader application of the recovered function. Solution enhancement is the operation to increase the polynomial order of a solution based on information from the surrounding

cells. Very often we need to enhance the polynomial order of the solution to compensate for the reduction in polynomial order when a derivative of the solution is taken. The reduction in polynomial order has a negative effect on accuracy and stability as seen in RDG-1x-Naive. Another reason for solution enhancement is the presence of nonlinearity in the equations.

Park at el. [29] first experimented with solution enhancement over a large stencil in the cell-centered RDG (cRDG) scheme. When dealing with a large multi-scale system, the most resource-consuming part of the scheme is the implicit time-marching algorithm. The number of equations to solve is directly proportional to the polynomial order of the scheme; for this reason, cRDG seeks to increase the order of the scheme without increasing the number of solution coefficients. cRDG recovers a new interior solution that satisfies all moments of neighboring cells. This is very similar to the reconstruction process in a finite-volume code, except cRDG works for $p \geq 0$.

The new RDG schemes make use of binary recovery for solution enhancement. Our approach results in a smaller system of equations and efficiently recycles the recovered function: the recovered function is used for both diffusion flux and solution enhancement. Consider a rough comparison of the numerical work required for solution enhancement between RDG and cRDG on a Cartesian grid with a tensor-product basis. cRDG solves a system of $N_{cRDG,d}\,(p+1)^d$ equations, where N_{cRDG} is the number of cells in the cRDG stencil and d is the physical dimension. RDG solves $2\,(p+1)^d$ equations per binary recovery for $N_{RDG,d}$ interfaces per cell, followed by a solution enhancement step with $(p+1)^2 + N_{RDG,d}(p+1)$ equations. Assuming $\frac{n+n^2}{2}$ floating point operations (flops) to invert an $(n \times n)$ matrix via LU-decomposition, Table 1 shows the approximate amount of computational work required for solution enhancement by RDG and cRDG. Clearly, cRDG becomes prohibitively expensive for higher polynomial order and physical dimension.

1-D		
p	RDG	cRDG
1	30	21
2	57	45
3	93	78
4	138	120

2-D		
p	RDG	cRDG
1	72	666
2	231	3321
3	528	10440
4	1584	25425

3-D		
p	RDG	cRDG
1	800	23436
2	6291	266085
3	29392	1.49×10^6
4	100700	5.69×10^6

Table 3.8: Flops comparison between RDG and cRDG for solution enhancement on a Cartesian grid, where (N_{RDG}, N_{cRDG}) is $(2, 3)$, $(5, 9)$, and $(6, 27)$ for 1-D, 2-D, and 3-D, respectively. RDG in 2-D and 3-D is more than an order of magnitude cheaper than cRDG.

Although the difference in computational work required between RDG and cRDG is significant, cRDG's enhanced solution contains the complete set of cross-derivatives which are useful if the PDE requires them. RDG's enhanced solution does not span a complete higher-order polynomial space (see Figure 3.13); however, the Navier-Stokes equations after integration by parts once no longer require the knowledge of the cross-derivatives in the discretized solution. Ideally, the level of solution enhancement should be optimized for the PDE being solved.

The recovery function proves to be a valuable tool; it is used for both diffusion flux and solution enhancement. Interestingly, the recovery concept represents a new green movement in CFD, in which both subroutines and results are reused to obtain a high order of accuracy. Starting off with the discretized solution u in each cell, a binary recovery code is first applied to produce all recovered functions f along the cell interfaces. A unifying code is then applied to acquire an enhanced solution $\hat{u}$ in each cell (see Figure 3.13). We can apply the same binary recovery code again to get an enhanced recovery function $\hat{f}$, and follow it up with the same solution enhancement code to get a doubly enhanced solution, $\hat{\hat{u}}$. Furthermore, $\hat{f}$ can be used as an enhanced diffusion flux. Solution enhancement comes at the expense of a growing stencil in which each binary interaction brings in information from cells further away; therefore, the optimal level of enhancement must be chosen for each PDE to ensure the allowable time step is maximized. The process can be repeated until the desired level of enhancement dictated by the PDE's is achieved. The following sections detail two types of solution enhancement necessary for approximating the Navier-Stokes viscous terms: the enhancement of the discretized solution and then the enhancement of the recovered function.

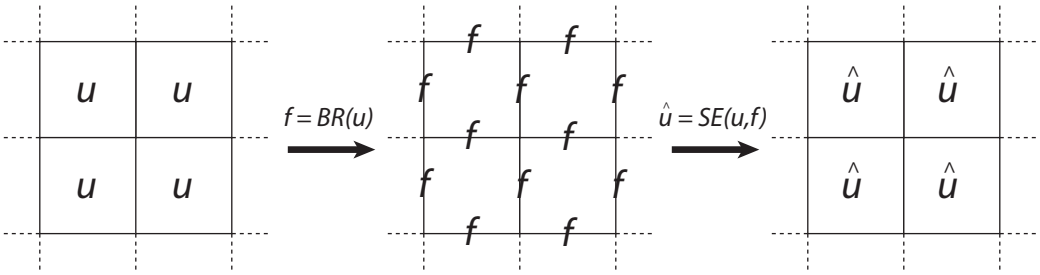

Figure 3.13: Recovery concept is used for both diffusion flux and solution enhancement. BR stands for binary recovery; SE stands for solution enhancement. The cycle of BR followed by SE can be repeated over and over again until the desired level of enhancement is achieved.

Interior-solution enhancement

We will not repeat the procedure to acquire f via binary recovery and focus on the solution-enhancement step. The enhanced solution $\hat{u}$ replaces the original solution u in the volume integral,

$$\begin{aligned}\int_{\Omega_i} (v_k)_i \; u_{i,t} \, d\Omega &= \oint_{\partial\Omega_i} (v_k)_i \; F\left(f_j, \nabla f_j\right) \cdot \hat{n} \, d\partial\Omega \\ &- \int_{\Omega_i} \nabla \left(v_k\right)_i \cdot F\left(\hat{u}_i, \nabla \hat{u}_i\right) \, d\Omega, \end{aligned} \tag{3.171}$$

where F is a vector representing the nonlinear diffusion terms. We require $\hat{u}$ to share all original moments with u, and in addition, to share all moments with f along the cell boundaries. The equations for solution enhancement in a general case are as follow:

$$\int_{\Omega_i} (v_k)_i \; u_i \, d\Omega = \int_{\Omega_i} (v_k)_i \; \hat{u}_i \, d\Omega, \tag{3.172}$$

$$\oint_{\partial\Omega_i} (v_m)_i \; \hat{u}_{i,t} \, d\partial\Omega = \oint_{\partial\Omega_i} (v_m)_i \; f_j \, d\partial\Omega, \quad m = 1, .., (p+1)^{d-1}, \tag{3.173}$$

where m is the index to the $(d-1)$-dimensional tensor-product test functions. The second equation uses the test functions from a reduced space because certain coordinate variables are fixed at the cell boundary. The only task left is to define the basis functions of $\hat{u}$, which lies in a larger function space $\hat{V}$. The following sections show 1-D examples on a Cartesian grid.

1-D solution-enhancement basis functions

In 1-D, Eqn 3.173 becomes a strong interpretation; $\hat{u}$ is required to be equal to f at the two points to the left and right of the cell. Let $P\left(p\right)$ be the function space with polynomials of degree of at most $p \geq 0$. If $u \in P\left(K\right)$, then $\hat{u} \in P\left(p+2\right)$. Take $p = 1$ for example, if u is spanned by $\{1, \, 2\xi - 1\}$, then $\hat{u}$ of Ω_j is spanned by $\{1, \, 2\xi - 1, \, 6\xi^2 - 6\xi + 1, \, 20\xi^3 - 30\xi^2 + 12\xi - 1\}$. The resulting system in matrix-vector form is given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ f_{j-1,j}\left(0\right) \\ f_{j,j+1}\left(0\right) \end{pmatrix}, \tag{3.174}$$

where the top two rows come from satisfying the moments of the original solution, and the last two rows arrive from satisfying the recovered functions on the two cell interfaces ($r = 0$). Here $\hat{a}_i$ are the solution coefficients to $\hat{u}$. In fact SE is so easy with our choice of basis functions for u and $\hat{u}$, we provide the system for $p = 2$ free of charge,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \hat{a}_4 \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ f_{j-1,j}(0) \\ f_{j,j+1}(0) \end{pmatrix}, \tag{3.175}$$

where $\hat{a}_4$ is the solution coefficient to the 4th-order Legendre polynomial. The inversion of the matrix on the LHS is trivial, making solution enhancement for 1-D RDG fast and cost efficient.

RDG-0x+

We briefly introduce an experimental scheme called RDG-0x+. As the name implies, the scheme acts upon the original weak form of the PDE's without any integration by parts, and replaces u with a special enhanced solution, $\tilde{u}$,

$$\int_{\Omega_i} (v_k)_i \; u_{i,t} \, d\Omega \;\; = \;\; \int_{\Omega_i} (v_k)_i \, \nabla \cdot F\left(\tilde{u}_i, \nabla \tilde{u}_i\right) \, d\Omega. \tag{3.176}$$

The construction of $\tilde{u}$ is similar to that of $\hat{u}$; however, $\tilde{u}$ also shares the normal derivative, f_n, of f,

$$\int_{\partial\Omega_i} (v_m)_i \; \tilde{u}_{n,i} \, d\partial\Omega = \int_{\partial\Omega_i} (v_m)_i \; f_{n,j} \, d\partial\Omega, \quad m = 1, .., (p+1)^{d-1}. \tag{3.177}$$

A comparison between RDG-1x+ and RDG-0x+ reveals many interesting differences. The number of terms RDG-0x+ deals with is significantly less in 2-D and 3-D, which translates to greater computational speed. However, this is offset by a more complicated solution-enhancement procedure. Users of RDG may choose between dealing with complicated terms in the 1x form, or spend more computational work to acquire $\tilde{u}$. Currently, this experimental scheme only works with PDE without cross-derivatives.

An interesting note: Huynh [19] analyzed various DG schemes for scalar diffusion

and found RDG-2x to be the most accurate while having the smallest spectral radius (albeit expensive). Smaller eigenvalues in the spatial discretization allow an explicit time-marching scheme to take larger time steps. For the scalar diffusion operator, both RDG-0x+ and RDG-1x+ expand into the original RDG-2x scheme; thus, both RDG-0x+ and RDG-1x+ come from the same fine pedigree, with the additional ability to solve nonlinear equations.

3.5.2 Fourier analysis of various RDG schemes

The behavior of all these new RDG schemes is best predicted with Fourier analysis. Unlike the previous section where we scrutinized the eigensystem, we only look at the order of accuracy and the largest eigenvalue. Unfortunately, Fourier analysis can only be performed on linear systems, hence we quickly return to the linear diffusion equation. The analysis is done numerically and the results are presented in Table 3.9. The mysterious RDG-1x++ is really a 2-D scheme whose secret profile will be revealed in Chapter 4; obviously it is not the best 1-D scheme. The closer the largest eigenvalue is to zero, the faster the scheme is. RDG-1x+ is equivalent to RDG-2x for the linear diffusion equation, which is a good property to have. RDG-1x$\bar{f}$ is unstable for $p \geq 3$. RDG-1x-Naive is the worst scheme with an order of accuracy of at best, $p+1$, and also with the largest eigenvalue.

$(\max(\mathrm{Re}(|\lambda|)), \max(\mathrm{Im}(\lambda)), \text{O.O.A.})$

p		RDG-1x-Naive	RDG-2x	RDG-1x$\bar{f}$	RDG-1x+	RDG-1x++
0		(4/0/2)	(4/0/2)	(4/0/2)	(4/0/2)	(10/0/2)
1		(27/0/2)	(15/0/4)	(30/0/4)	(15/0/4)	(41/0/4)
2		(89/0/2)	(33/0/8)	(125/0/6)	(33/0/8)	(92/0/6)
3		(221/0/4)	(67/5.5/10)	N/A	(67/5.5/10)	(159/0/10)
4		(465/0/4)	(109/9.2/14)	N/A	(109/9.2/14)	(239/0/10)
5		(870/0/6)	(152/10.7/16)	N/A	(152/10.7/16)	(325/10.7/14)

Table 3.9: Rate of L_2-error convergence. The VNN number are given for RK3, RK4 and RK5 for $p = 1$, 2, and 3, respectively.

3.5.3 Linear variation of diffusion coefficient

We first develop a test case where M_x is integrable. This case is considerably easier than the nonlinear version because the resulting volume integral is still evaluated

exactly. Consider a linear variation in μ,

$$\mu(u) = \mu_0 + \frac{u}{\mu_1}, \tag{3.178}$$

where $\mu_{0,1}$ are constants. The integral of M_x reads,

$$\int \left(\mu_0 u_x + \frac{u u_x}{\mu_1} \right) dx = \mu_0 u + \frac{u^2}{2\mu_1}. \tag{3.179}$$

We subject RDG-1x-Smart, RDG-1x-Naive and RDG-2x (RDG-3x is equivalent to RDG-2x in this linear case) to compute the steady-state solution of the following manufactured problem with a source term, $S(x)$,

$$u_t = (\mu(u) u_x)_x + S(x), \tag{3.180}$$

$$S(x) = -\pi^2 \cos(2\pi x), \tag{3.181}$$

$$\mu(u) = u. \tag{3.182}$$

Together with the initial and final solution of

$$u(x) = \sin(\pi x), \tag{3.183}$$

the diffusion coefficient will always be positive. We use Neumann boundary conditions on the left and right of the physical domain $x \in [0, 1]$:

$$u_x(0) = \pi, \tag{3.184}$$

$$u_x(1) = -\pi. \tag{3.185}$$

The solution starts with the exact solution, and then marches forward in time with a RK3, RK4, and RK5 scheme for $p = 1$, 2, and 3, respectively. The order of the RK schemes is not of importance in this steady-state problem; however, high-order RK does provide faster convergence rates. The simulation ends when the change in solution between each time step is less than $\varepsilon < 1 \times 10^{-15}$. A summary of the results is presented in Table 3.10, while the full results is found in Appendix B. The VNN number shows the potential difference in speed between the schemes for time-accurate problems. Clearly, RDG-2x and RDG-1x+ are the fastest schemes while RDG-1x-Naive is the slowest. In terms of order of accuracy, RDG-1x-Naive lives up to its name. RDG-1x$\bar{f}$ suffers from instability for $p \geq 3$ (as shown by Fourier analysis)

even with VNN as low as 0.0001. Although the full results in the Appendix B show that RDG-1x$\bar{f}$ ($p = 3$) is potentially 10th-order for certain grids, we can no longer use this scheme due to the lack of robustness. Our classic RDG-2x scheme performs fairly well, but not as its former glory. RDG-2x seems to suffer at the boundary (hence the n-th and a half order results)[3]. Our winner for this test case is RDG-1x+ with both the highest VNN and rate of convergence in L_2-error.

Scheme	p	VNN		Rate $\bar{u}$	Rate $\overline{\Delta u}$	Rate $\overline{\Delta^2 u}$	Rate $\overline{\Delta^3 u}$
RDG-1x-Naive	1	0.07		2	3		
	2	0.02		2	3	4	
	3	0.01		4	5.5	4	5
RDG-2x	1	0.15		4	5		
	2	0.08		5.5	5.5	5.5	
	3	0.04		7.5	7.2	7.4	7
RDG-1x$\bar{f}$	1	0.08		4	5		
	2	0.02		6	7	6	
	3	N/A		N/A	N/A	N/A	N/A
RDG-1x+	1	0.15		4	5		
	2	0.08		8	7	6	
	3	0.04		10	9	8	9

Table 3.10: Convergence rate of L_2-error. The VNN number are given for RK3, RK4 and RK5 for $p = 1$, 2, and 3, respectively.

3.5.4 Nonlinear variation of diffusion coefficient

The nonlinear variation in the diffusion coefficient will very likely generate non-integrable diffusion flux terms, especially in a system of equations. A non-integrable diffusion flux makes implementation of RDG-2x impossible. Consider the following nonlinear diffusion coefficient,

$$\mu(u) = e^{-u^2}, \tag{3.186}$$

then the diffusion flux,

[3]An issue with the boundary scheme is usually revealed by looking at the rate of L_1, L_2 and L_∞-error together. If the problem occurs only at the domain boundary, the rate of L_1-error will be half-an-order larger than the rate of L_2-error, and the rate of L_2-error will also be half-an-order larger than the rate of the L_∞- error.

$$M_x = u_x e^{-u^2}, \tag{3.187}$$

is not integrable. We briefly sidetrack and talk about an attempt to implement recovery for RDG-2x with an additional correction term in the next section.

RDG-3x

The key experimental concept within the RDG-3x scheme is to approximate M_x with a monomial which is integrable and to estimate the magnitude of the correction term. We have demonstrated the difference in the treatment of the volume integral between RDG-1x-Naive and RDG-1x+ results in orders of difference in solution accuracy. Let us define the difference in the volume integrals of the two classes of RDG-1x as the correction term. If we rewrite the RDG-2x in RDG-3x form, we observe that RDG-1x-Naive is very similar to RDG-3x, except for the component $[v_x\tilde{M} - v_x M]$, which is called the correction term,

$$\left[v_x\tilde{M} - v_x M\right] \approx \int v_x \hat{\tilde{M}}_x \, dx - \int v_x M_x \, dx. \tag{3.188}$$

Our numerical results for RDG-1x shows the correction term changes the scheme from 2nd-order to 4th-order for $p = 1$, and from 2nd-order to 6th-order for $p = 2$, clearly revealing the correction term is $O\left(\Delta x^2\right)$. With the magnitude of the correction term in mind, we proceed to approximate local M_x and recovered $\tilde{M}_{x(j,j+1)}$ with monomials,

$$M_x \cong N_x = \Sigma_{i=1}^{p} a_i x^{i-1}, \tag{3.189}$$

$$\tilde{M}_{x(j,j+1)} \cong \tilde{N}_{x(j,j+1)} = \Sigma_{i=1}^{2p+1} b_i x^{i-1}, \tag{3.190}$$

where a_i and b_i are unknown coefficients to solve for via the following projections,

$$\int_{\Omega_j \cup \Omega_{j+1}} x^i \tilde{M}_x \, dx = \int_{\Omega_j \cup \Omega_{j+1}} x^i \tilde{N}_x \, dx, \tag{3.191}$$

$$\int_{\Omega_j} x^i M_x \, dx = \int_{\Omega_j} x^i N_x \, dx. \tag{3.192}$$

The monomials are easily integrable,

$$\int N_x \, dx = \Sigma_{i=1}^{p} a_i \int x^{i-1} \, dx + c, \tag{3.193}$$

$$\int \tilde{N}_x \, dx = \Sigma_{i=1}^{2p+1} b_i \int x^{i-1} \, dx + \tilde{c}. \tag{3.194}$$

with unknown constants, c and $\tilde{c}$. The constants are calibrated with the knowledge that $[v_x\tilde{M} - v_x M] \approx O(\Delta x^2)$. In order to make the correction term small, we enforced the following in the weak sense,

$$\int_{\Omega_j} N_x \, dx = \int_{\Omega_j} \tilde{N}_x \, dx. \tag{3.195}$$

Note these constants are not unique, there exist different constants for each interface and element. This scheme ultimately proved to be unstable, and the complexity makes it difficult to implement in multi-dimensional cases. This concludes our attempt to use RDG-2x for nonlinear fluxes.

Numerical results for nonlinear diffusion coefficient

We present the following manufactured solution for the nonlinear diffusion equation,

$$u_t = \left(u_x e^{-u^2}\right)_x + S(x,t), \tag{3.196}$$

with a source term as a function of both space and time,

$$S(x) = \sin(\pi x)\, e^{-3t - e^{-2t}\sin(\pi x)^2} \left(-e^{2t + e^{-2t}\sin(\pi x)^2} + \pi^2 \left(e^{2t} + 1 + \cos(2\pi x)\right)\right). \tag{3.197}$$

The initial and final solutions are given by

$$u(x,t) = \sin(\pi x)\, e^{-t}, \tag{3.198}$$

on the domain $x \in [0,\, 1]$. We advance in time with RK3, RK4, and RK5 for $p = 1$, 2, and 3, respectively, until $t = 1$ is reached. The numerical results for RDG-1x+ are included in Table 3.11. We do not include RDG-1x-Naive because it is of extremely low accuracy and RDG-1x$\bar{f}$ because it is not stable for all p. The results are very similar to the linear-variation-diffusion-coefficient case. In the case of $p = 3$ with a coarse grid of two cells only, the odd derivatives are zero, hence they exhibit computer-zero error.

3.6 Chapter summary

We hope we have convinced you that RDG is a very simple diffusion method for DG. The recovered function, obtained by the binary recovery operation, is used for the dif-

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	1.60×10^{-5}	2.03×10^{-4}						
	15	3.26×10^{-6}	7.21×10^{-6}			3.9	4.9		
	20	1.04×10^{-6}	9.72×10^{-7}			3.9	4.9		
	30	2.08×10^{-7}	2.32×10^{-7}			3.9	4.9		
	40	6.60×10^{-8}	3.08×10^{-8}			3.9	4.9		
	60	1.31×10^{-8}	7.32×10^{-9}			3.9	4.9		
2	4	6.54×10^{-8}	5.78×10^{-6}	1.61×10^{-5}					
	5	1.15×10^{-8}	1.22×10^{-6}	4.61×10^{-6}		7.7	6.9	5.6	
	8	2.99×10^{-10}	4.57×10^{-8}	2.98×10^{-7}		7.7	6.9	5.8	
	10	5.14×10^{-11}	9.59×10^{-9}	7.95×10^{-8}		7.8	7.0	5.9	
	12	1.12×10^{-11}	2.68×10^{-9}	2.69×10^{-8}		7.9	6.9	5.9	
	16	1.22×10^{-12}	3.57×10^{-10}	4.83×10^{-9}		7.9	7.0	5.9	
3	2	8.55×10^{-8}	7.45×10^{-18}	4.42×10^{-6}	2.80×10^{-17}				
	3	4.06×10^{-9}	5.28×10^{-9}	4.89×10^{-7}	1.80×10^{-6}	7.5	N/A	5.4	N/A
	4	2.98×10^{-10}	2.07×10^{-10}	6.85×10^{-8}	1.58×10^{-7}	9.0	11.2	6.8	8.4
	5	3.73×10^{-11}	1.88×10^{-11}	1.34×10^{-8}	2.31×10^{-8}	9.3	10.7	7.3	8.6
	8	3.51×10^{-13}	9.93×10^{-14}	3.67×10^{-10}	3.69×10^{-10}	9.9	11.1	7.6	8.8
	10	2.72×10^{-14}	1.60×10^{-14}	6.40×10^{-11}	5.08×10^{-11}	11.4	8.1	7.8	8.8

Table 3.11: L_2-error of RDG-1x+ scheme for steady-state nonlinear diffusion problem with periodic boundary condition.

fusion flux at the cell interface. Both Fourier analysis and numerical experiment show the good stability and accuracy of RDG schemes. When compared to existing methods such as the (σ, μ)- family, RDG contains new bilinear terms unseen before. The original RDG-2x is an excellent choice for linear problems. For nonlinear problems we must extend the recovery concept to include solution enhancement. The RDG-1x+ uses enhanced solution in the volume integral to achieve high order of accuracy for nonlinear diffusion problems. We next extend these 1-D concepts to 2-D.

CHAPTER IV

RDG in Two-Dimensions for a Scalar Equation

The extension of any numerical method from 1-D to multiple dimensions is not a simple task, especially when going from 1-D to 2-D. Once the framework is finalized from 1-D to 2-D, going to 3-D is just a matter of exercising existing concepts. One major drawback of RDG is that the accuracy of the recovered function is dependent on the direction; RDG only improves the accuracy in the face-normal direction, while the accuracy in the face-parallel direction is left unchanged. This initial drawback makes RDG-2x unsuited to deal with equations with cross-derivatives; however, this issue is alleviated with the arrival of RDG-1x++. We first introduce the recovery concept, then the various variable transformations involved, and finally the concept of solution enhancement to overcome difficulty with approximating the cross-derivatives in nonlinear diffusion equations.

4.1 2-D recovery equations for binary recovery

The 2-D recovery equations for binary recovery (BR) are almost the same as the 1-D version, with the addition of the y-coordinate. Consider the solutions in two adjacent cells of the form presented in Eqn 2.1, where the basis functions are expressed in terms of local coordinates (ξ, η). We look for a recovered function centered on the

shared interface in the form of Eqn 2.3. The 2-D recovery equations are

$$\begin{aligned} \iint_{\Omega_j} v_j u_j \, dxdy &= \iint_{\Omega_j} v_j f_{j,j+1} \, dxdy, \\ \iint_{\Omega_{j+1}} v_{j+1} u_{j+1} \, dxdy &= \iint_{\Omega_{j+1}} v_{j+1} f_{j,j+1} \, dxdy. \end{aligned} \tag{4.1}$$

As we have indicated in Chapter 2, this set of equations applies to all $v \in V$, unless stated otherwise. The face-normal direction is by default the r-coordinate, while the face-parallel direction is the s-coordinate. Since Eqn 4.1 contains variables from three different coordinate systems, we choose to express everything in terms of the local variables for simplicity. Our first step is to convert the integrals over the global variables into integrals over the local variables. Secondly, we express the recovered function in terms of local variables, i.e. $f(r,s) \rightarrow f(\xi,\eta)$.

4.1.1 Transformation from global to local coordinate

Consider the mapping from an unstructured triangle T to a standard triangle T_S, and the mapping from an unstructured quadrilateral Q to a standard square Q_S in Figure 4.1 on the left and right, respectively. The mapping from local to global coordinates for a triangle is,

$$x = (x_B - x_A)\,\xi + (x_C - x_A)\,\eta + x_A, \tag{4.2}$$

$$y = (y_B - y_A)\,\xi + (y_C - y_A)\,\eta + y_A, \tag{4.3}$$

and for a quadrilateral is,

$$x = (1-\xi)(1-\eta)\,x_A + (\xi)(1-\eta)\,x_B + (\xi\eta)\,x_C + (1-\xi)(\eta)\,x_D, \tag{4.4}$$

$$y = (1-\xi)(1-\eta)\,y_A + (\xi)(1-\eta)\,y_B + (\xi\eta)\,y_C + (1-\xi)(\eta)\,y_D. \tag{4.5}$$

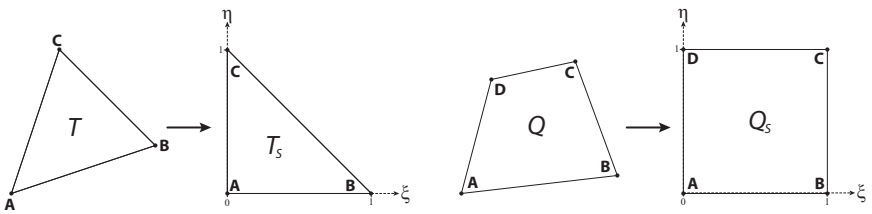

Figure 4.1: Mapping from the global coordinates to the local coordinates. The arbitrary triangle T is transformed into a standard triangle T_S. The arbitrary quadrilateral Q is transformed into a standard square Q_S.

If the quadrilateral is a square with $x_A = x_D$, $x_B = x_C$, $y_A = y_B$, and $y_C = y_D$, then the bilinear expression becomes linear. Linear mapping is preferable because the inverse mapping is easily obtained; however, that is not true for the bilinear case. We are also interested in the the transformation of derivatives between coordinate systems. Applying the chain rule, the local derivatives are expressed in terms of global derivatives,

$$\frac{\partial}{\partial \xi} = \frac{\partial}{\partial x} \cdot \frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y} \cdot \frac{\partial y}{\partial \xi}, \tag{4.6}$$

$$\frac{\partial}{\partial \eta} = \frac{\partial}{\partial x} \cdot \frac{\partial x}{\partial \eta} + \frac{\partial}{\partial y} \cdot \frac{\partial y}{\partial \eta}, \tag{4.7}$$

or in matrix form,

$$\begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = J \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}, \tag{4.8}$$

where J is the Jacobian matrix. The integration domain over global coordinates is now transformed into local coordinates via

$$dxdy = |J|\, d\xi d\eta, \tag{4.9}$$

where $|J|$ is the determinant of J,

$$|J| = \frac{\partial x}{\partial \xi} \cdot \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta}. \tag{4.10}$$

In our case where the mapping is one-to-one, $|J|$ is strictly non-zero within the domain of the element. For a triangle, $|J|$ is a constant that is equal to twice the area of the triangle. Notice all vertices are labeled in a counter-clockwise manner; this is to ensure $|J|$ remains positive. The inverse transformation of the derivatives is given by,

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix}. \tag{4.11}$$

Note that J^{-1} must be calculated numerically for quadrilaterals because the analytical expression for the inverse transformation does not exist. The real gain from converting an integration over the global coordinates into local coordinates is the simplification of the integration limits. For example, the integration of any function g over an

arbitrary triangle T becomes,

$$\iint_T g\,dxdy = \int_0^1 \int_0^{1-\eta} g\,d\xi d\eta, \tag{4.12}$$

and for an arbitrary quadrilateral Q,

$$\iint_Q g\,dxdy = \int_0^1 \int_0^1 g\,d\xi d\eta. \tag{4.13}$$

With these convenient integration limits, one can quickly apply one's favorite Gaussian integration techniques. We conclude our review of finite-element coordinate systems and look at the recovery coordinate system.

4.1.2 Tranformation from global to recovery coordinates

The recovery coordinate system, $(r,\, s)$, is a bit more complicated and requires a closer look. There are two major issues at hand, the configuration of the coordinate system and the domain of integration. The configuration of $(r,\, s)$ is based on three simple steps: translation, rotation, scaling. We first translate the coordinate system to the midpoint, (x_m, y_m), of the interface as shown in Figure 4.2. Next, we rotate the frame by α degrees to align the r-coordinate to the face-normal direction, and the s-direction to the face-parallel direction. Finally, we scale the s-coordinate axis down to unity with the $r-$coordinate conforming to the same scaling factor, C. The unity-scaling provides a better condition number for solving the recovery equations. The combined collection of linear transformations results in the following expression in matrix form,

$$\begin{bmatrix} r \\ s \end{bmatrix} = C \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x - x_M \\ y - y_M \end{bmatrix}, \tag{4.14}$$

where the scaling constant is determined from the length of the shared interface e,

$$C = \frac{2}{\|e\|}. \tag{4.15}$$

This mapping from $(x,\, y)$ to $(r,\, s)$, along with the mapping from the previous section from $(\xi,\, \eta)$ to $(x,\, y)$, allows us to relate the three coordinate systems and ultimately express everything in terms of the local coordinates. Our next step is to define the basis functions of the solution u and the recovery function f.

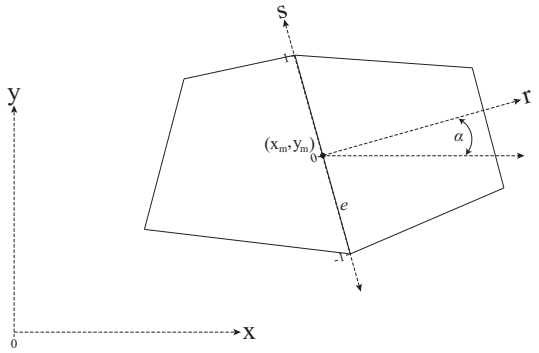

Figure 4.2: Mapping from global to recovery coordinates.

4.1.3 Orthogonal basis functions

All basis functions are mathematically equivalent as long as they span the same function space; however, they are not numerically equivalent in regard to the condition number of an invertible system of equations, the simplification of the update equations, and the ease of Gaussian integration. Basis functions are generally split into the two classes, nodal and modal bases. A nodal basis is characterized by the physical shape of the function. The first to come to mind is the famous hat-function of the finite-element community. For an arbitrarily high-order basis, the finite-element community uses Lagrangian interpolation functions. On the other hand, modal basis functions are simply mathematical functions. We have previously introduced monomials and Legendre polynomials

The difference between modal and nodal bases lies in their ability to adapt to multiple dimensions and to the computational grid. The nodal basis functions are commonly used due to their ease to adapt to computational grids of any dimension and shape. In addition, the nodal basis functions can be chosen to coincide with the Gaussian points to speed up numerical runs. A modal basis is harder to generate in higher dimensions, but possesses mathematical elegance. As seen from the previous chapters, each component of a modal basis represents a physically meaningful quantity: the cell average, the first gradient, etc. In particular, the orthogonal basis functions exhibit excellent numerical properties in which the condition number of an invertible system is significantly reduced, minimizing numerical error. The update equations become decoupled on the LHS. We will see this orthogonal property playing an important role in Chapter 6 for the Hancock-Huynh DG scheme.

We focus our attention on the orthogonal basis functions for quadrilaterals and triangles. We first cover the quadrilateral because the 1-D basis function is easily

extended to 2-D via tensor product. Imagine a multiplication table with the Legendre basis functions expanded in terms of ξ on the horizontal axis, and the Legendre basis functions expanded in terms of η on the vertical axis as shown in Table 4.1. The tensor product automatically guarantees the resulting cross-multiplied basis functions of ξ and η to be orthogonal on the standard square.

1	$2\xi - 1$	$6\xi^2 - 6\xi + 1$	$\dots$
$2\eta - 1$	$(2\xi - 1)(2\eta - 1)$	$(6\xi^2 - 6\xi + 1)(2\eta - 1)$	$\dots$
$6\eta^2 - 6\eta + 1$	$(2\xi - 1)(6\eta^2 - 6\eta + 1)$	$(6\xi^2 - 6\xi + 1)(6\eta^2 - 6\eta + 1)$	$\dots$
$20\eta^3 - 30\eta^2 + 12\eta - 1$	$(2\xi - 1)(20\eta^3 - 30\eta^2 + 12\eta - 1)$	$(6\xi^2 - 6\xi + 1)(20\eta^3 - 30\eta^2 + 12\eta - 1)$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 4.1: Tensor-product basis for Legendre polynomials.

Unfortunately there exist no standard orthogonal basis for the standard triangle; hence, we provide our own homebrewed method (a.k.a trial and error) for your viewing pleasure. Based on our hard-earned experience, it is best to start with very simple functions. Our goal is to create an orthonormal basis whose mass matrix M,

$$M = \begin{bmatrix} < v_0, \phi_0 > & < v_0, \phi_1 > & \dots \\ < v_1, \phi_0 > & < v_1, \phi_1 > & \\ \vdots & & \ddots \end{bmatrix}, \tag{4.16}$$

is the identity matrix. We begin by assuming our first basis function is of the form, $\phi_0 = C_{0,0}$, where C is used to indicate an unknown constant. With one equation and one unknown constant, this equation is easily solved with $\phi_0 = \sqrt{2}$. We then include the next basis function which will contain two unknown constants. We assume a basis function of the form, $\phi_1 = C_{1,0}\xi + C_{1,1}\eta$. The mass matrix becomes a two by two matrix providing two conditions for the two unknown coefficients. The result is $\phi_1 = 2\sqrt{3}(\xi - \eta)$. Our next basis function contains three unknowns, $\phi_2 = C_{2,0} + C_{2,1}\xi + C_{2,2}\eta$. The pattern is to add one additional unknown constant to every higher-order basis, together with a corresponding function in terms of ξ and η. The methodology is not unique; we provide our sample orthogonal basis on the standard triangle in Table 4.2.

The number of moments per cell, N_{mom}, is used frequently in expresssions. This constant is dependent on the p-th order of the scheme and is different for standard triangle and square. The N_{mom} for a standard triangle is given by

$$N_{\text{mom},T} = \left(\frac{p}{2} + 1\right)(p + 1), \tag{4.17}$$

$\sqrt{2}$
$2\sqrt{3}\,(\xi-\eta)$
$6\xi+6\eta-4$
$\sqrt{14}\left(\frac{12}{7}\,(\xi+\eta)-\frac{60}{7}\xi\eta-\frac{3}{7}\right)$
$\sqrt{70}\left(\frac{24}{7}\,(\xi+\eta)+\frac{6}{7}\,(\xi\eta-1)-3\,(\xi+\eta)^2\right)$

Table 4.2: A sample orthonormal basis ($p=4$) for the standard triangle.

and for a standard square,

$$N_{\text{mom},S}=(p+1)^2\,. \tag{4.18}$$

4.1.4 Recovery basis derived from tensor product basis

The Tetris game idea from Chapter 2 also applies to the 2-D tensor-product basis. Figure 4.3 shows examples for $p=1$ and 2 where the recovery basis is more accurate in the face-normal direction (r-coordinate). For a p-th order scheme with $N_{\text{mom},S}$ moments per cell, the recovered function f is defined to be,

$$f=\sum_{i=0}^{2p+1}\sum_{j=0}^{p} b_{i,j}\,r^i s^j, \tag{4.19}$$

where $b_{i,j}$ are the unknown coefficients to solve for.

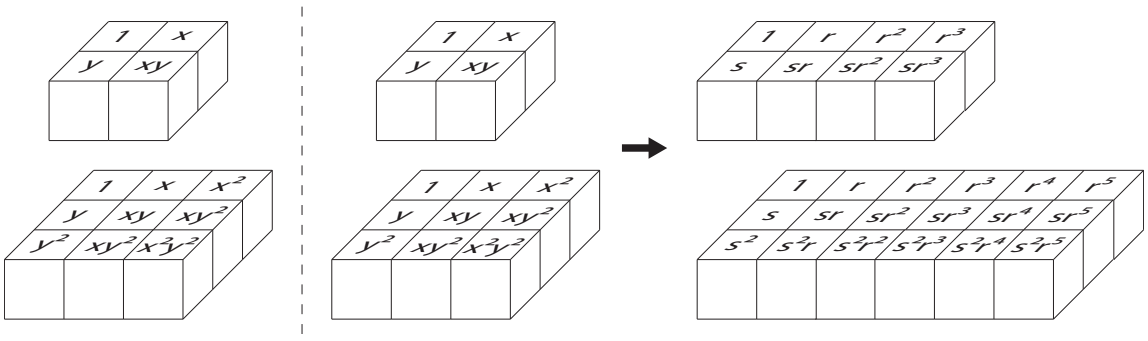

Figure 4.3: A little block game for determining the recovery basis in 2-D. The blocks on the left and right of the dotted line indicate basis functions of the solution in Ω_j and Ω_{j+1} respectively. Now imagine gravity pulls to the left; the blocks from Ω_{j+1} fall on "top" of the blocks in Ω_j to form the recovery basis. Notice there are more blocks in the r-coordinate than in the s-coordinate.

4.2 RDG Schemes for 2-D

This section on RDG schemes is loosely split into structured/unstructured and linear/nonlinear numerical experiments. The combination of a linear problem with an unstructured grid allows us to focus primarily on recovery at the domain-boundary, while the combination of a nonlinear problem with a structured grid allow us to focus on dealing with cross-derivative terms. Common to all cases is the need to recover at the domain boundary. Fortunately, recovery at the domain boundary is very similar to the material presented in the previous chapter; hence, we will not go into such detail again. For linear equations we introduce the RDG-2x, RDG-1x$\bar{f}$, RDG-1x$\tilde{f}$, and RDG-1x-Naive. As for nonlinear equations, we present RDG-1x+, RDG-1x++, and RDG-1x++CO ("CO" stands for Cartesian optimization).

4.2.1 Recovery at the domain boundary

Recovery at the domain boundary is a tricky business on 2-D grids; it's hard to obtain a high-order recovered function on unstructured grids. Recall the full boundary-recovered function, f_F, requires two cells adjacent to the domain boundary. This procedure is straightforward for structured Cartesian grid, but is complicated for triangular grids. The compact boundary-recovered function f_C, which only requires one cell adjacent to the domain boundary, is prefered for unstructured grids.

Full boundary-recovered function for structured Cartesian grid

The full boundary-recovered function is recommended for structured Cartesian grids due to the convenient alignment of cell centers. Figure 4.4 shows the stencil for the full boundary-recovered function for $p = 1$ on the left and $p = 1$ on the right. Note that we are using tensor-product basis in this example. This is very similar to the 1-D case, with the exception that the domain boundary provides $p + 1$ pieces of information instead of one. Our goal is to ensure the polynomial orders of f_F and the interior recovered function are the same. We force f_F to have the $(p+1)^2$ moments of the first adjacent cell, and the $(p+1)^2 - (p+1)$ moments of the next adjacent cell. The last set of conditions come from the Dirichlet boundary condition, $g(x, y)$, on the domain boundary, $\partial\Omega_e$. Let Ω_1 be the first cell adjacent to the domain boundary,

and Ω_2 be the next adjacent cell. The recovery equations for f_F are

$$\iint_{\Omega_1} v_i \, f_F \, dxdy = \iint_{\Omega_1} v_i \, u_1 \, dxdy \quad \text{for } i = 0, .., (N_{\text{mom},S} - 1), \tag{4.20}$$

$$\iint_{\Omega_2} v_i \, f_F \, dxdy = \iint_{\Omega_2} v_i \, u_2 \, dxdy \quad \text{for } i = 0, .., (N_{\text{mom},S} - p - 2), \tag{4.21}$$

$$\int_{\partial\Omega_e} v_i \, f_F \, d\partial\Omega_e = \int_{\partial\Omega_e} v_i \, g\,(x, y) \, d\partial\Omega_e \quad \text{for } i = 0, .., p. \tag{4.22}$$

The choice of moments to be used from Ω_2 is not arbitrary. Figure 4.4 shows if the Tetris-game idea is applied again, we do not need the higher order information in the r-coordinate in Ω_2. The form of the boundary f_F is identical to that of the interior f,

$$f_F\,(r) = \sum_{i=0}^{2p+1} \sum_{j=0}^{p} b_{i,j} \, r^i s^j, \tag{4.23}$$

where $b_{i,j}$ coefficients are the unknowns, and $r = 0$ is the location of the domain boundary. For the left boundary domain, we seek a f_F that spans the union of Ω_1 and Ω_2 such that Ω_1 belongs to $(r, s) \in [0,\, 2] \times [-1,\, 1]$ and Ω_2 belongs to $(r, s) \in [2,\, 4] \times [-1,\, 1]$.

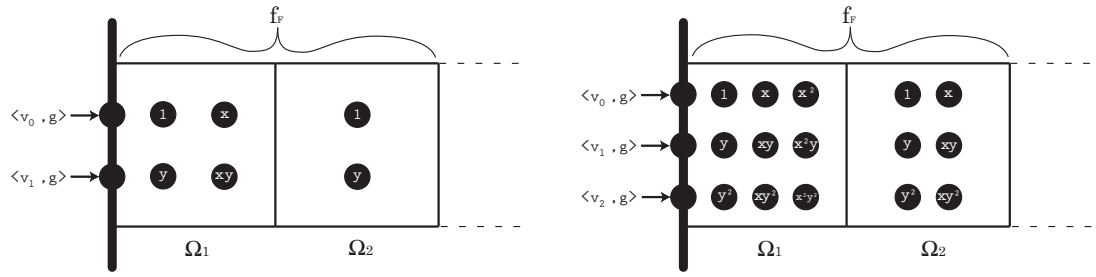

Figure 4.4: Full boundary-recovered functions for 2-D Cartesian grid for $p = 1$ on the left and $p = 2$ on the right. The domain boundary provides $p + 1$ conditions based on the Dirichlet function, g. Notice the choice of moments to satisfy in Ω_2 is not arbitrary.

Compact boundary-recovered function for unstructured triangular grid

The compact boundary-recovered function is recommended for unstructured triangular grid due to the inconvenient alignment of cell centers (see left frame of Figure 4.5). In a full boundary-recovery procedure, we use information from a faraway cell in the face-normal direction. In this example, the immediate adjacent cell of Ω_1 is either Ω_2 or Ω_3. Their cell centers show that they do not provide strong enough additional

information in the face-normal direction. If we forcefully attempt a full boundary-recovery based on Ω_2 or Ω_3, the resulting system of equation will be ill-conditioned. The right of Figure 4.5 shows the stencil for a compact boundary-recovered function for $p = 1$. We enforce f_C to satisfy the $N_{\text{mom},T}$ moments of the first adjacent cell, and the $(p+1)$ moments from the Dirichlet boundary condition, $g(x, y)$, on the domain boundary, $\partial\Omega_e$. Let Ω_1 be the first cell adjacent to the domain boundary. The recovery equations for f_C are

$$\iint_{\Omega_1} v_i \, f_C \, dxdy = \iint_{\Omega_1} v_i \, u_1 \, dxdy \qquad \text{for } i = 0, .., (N_{\text{mom},S} - 1), \tag{4.24}$$

$$\int_{\partial\Omega_e} v_i \, f_C \, d\partial\Omega_e = \int_{\partial\Omega_e} v_i \, g(x, y) \, d\partial\Omega_e \quad \text{for } i = 0, .., p. \tag{4.25}$$

Regrettably, the function space of f_C is much smaller than the function space of the interior recovered function f. This will suffice if we are only expecting the scheme to be of the $(p+1)$-th order. We express f_C in terms of monomials,

$$f_C(r) = \sum_{i=0}^{p} \sum_{j=0}^{p-i} b_{i,j} \, r^i s^j + \sum_{i=0}^{p} b_{i+1,p-i} \, r^{i+1} s^{p-i}. \tag{4.26}$$

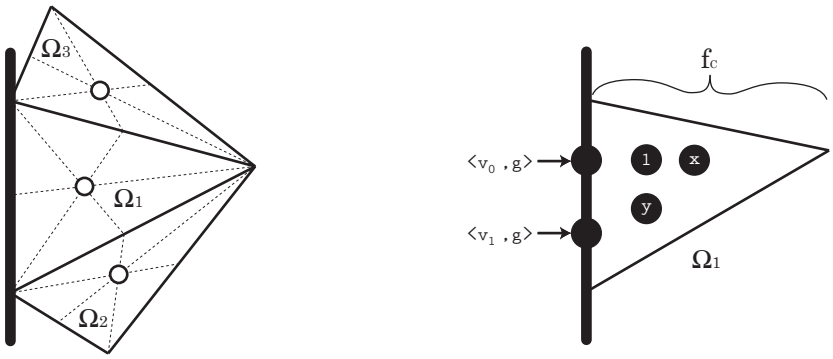

Figure 4.5: (Left) The hollow circles indicate the centroids of the triangles. It is difficult to obtain a full boundary-recovered function due to the alignment of cell centers that is more biased in the face-parallel direction. (Right) Compact boundary-recovered function for 2-D Cartesian grid for $p = 1$. The domain boundary provides $p + 1$ conditions based on the Dirichlet boundary condition, $g(x, y)$.

4.2.2 Linear RDG schemes

We approach the linear diffusion equation again as our first test case in 2-D. This section is divided into parts regarding structured and unstructured grids. The linear diffusion equation is given by,

$$u_t = \nabla \cdot \nabla u = u_{xx} + u_{yy}. \tag{4.27}$$

The weighted-residual form for Ω_j is given by,

$$\iint_{\Omega_j} v u_t \, dt = \iint_{\Omega_j} v \nabla \cdot \nabla u \, dxdy, \tag{4.28}$$

The once and twice partly integrated forms are given by

$$\begin{aligned} \iint_{\Omega_j} v u_t \, dt &= \oint_{\partial\Omega_j} (v \nabla u) \cdot \hat{n} \, d\partial\Omega - \iint_{\Omega_j} \nabla v \cdot \nabla u \, dxdy, & (4.29) \\ &= \oint_{\partial\Omega_j} (v \nabla u - u \nabla v) \cdot \hat{n} \, d\partial\Omega + \iint_{\Omega_j} u \nabla \cdot \nabla v \, dxdy. & (4.30) \end{aligned}$$

RDG-2x

The first scheme to be introduced is the original RDG-2x scheme based on twice integrating by parts. We replace the solution in the surface integral with the recovered function f,

$$\iint_{\Omega_j} v u_t \, dt = \oint_{\partial\Omega_j} (v \nabla f - f \nabla v) \cdot \hat{n} \, d\partial\Omega + \iint_{\Omega_j} u \nabla \cdot \nabla v \, dxdy. \tag{4.31}$$

The equation decouples nicely into x- and y-components on a Cartesian grid, resulting in two 1-D RDG-2x schemes being applied in each direction.

RDG-1x-Naive

The underdog from the previous chapter is presented again with the sole purpose of illustrating what not to do. RDG-1x-Naive replaces the solution in the surface integral of the once partly integrated formulation with the recovered function,

$$\iint_{\Omega_j} v u_t \, dt = \oint_{\partial\Omega_j} (v \nabla f) \cdot \hat{n} \, d\partial\Omega - \iint_{\Omega_j} \nabla v \cdot \nabla u \, dxdy. \tag{4.32}$$

Since the 2-D linear diffusion problem decouples nicely into two 1-D problems in the x- and y-directions, we expect the same poor results as in one dimension.

RDG-1x$\bar{f}$

This scheme (see previous chapter) is one of the first members of the RDG-1x-Smart family. RDG-1x$\bar{f}$ replaces the solution in the volume integral with the average of all recovered functions $\bar{f}$ available in Ω_j,

$$\iint_{\Omega_j} v u_t \, dt = \oint_{\partial\Omega_j} (v \nabla f) \cdot \hat{n} \, d\partial\Omega - \iint_{\Omega_j} \nabla v \cdot \nabla \bar{f} \, dxdy. \tag{4.33}$$

Fourier analyses in both 1-D and in 2-D (end of this chapter) reveal the scheme is unstable for $p \geq 3$. Though the scheme had some success in 1-D, its performance in 2-D is abysmal. The averaging of recovered functions, which mixes the accurate information in the face-normal direction with the less accurate information in the face-parallel direction, causes the scheme to perform poorly (on the same level as RDG-1x-Naive). This observation leads us to our next generation scheme, RDG-1x$\tilde{f}$.

RDG-1x$\tilde{f}$

We design a scheme to take full advantage of the highly accurate information in the face-normal direction while omitting the inaccurate face-parallel information. Our design goal is to obtain a better ∇u for the volume integral. Since we have many ∇f candidates from each cell interface, recovery of the solution gradient involves a least-squares procedure. Let N_j be the matrix whose rows are the vectors $\hat{n}_{j,l}$, where index l cycles through the neighbors Ω_l of Ω_j, $l = l_1, \ldots, l_L$, and let $g_j(\vec{x})$ be the vector of recovered normal derivatives $\partial f_{j,l}(\vec{x}) / \partial n_{j,l}$, evaluated in a point $\vec{x} \in \Omega_j$; that is

$$N_j = \begin{pmatrix} (n_{l_1})_x & (n_{l_1})_y & (n_{l_1})_z \\ \vdots & \vdots & \vdots \\ (n_{l_L})_x & (n_{l_L})_y & (n_{l_L})_z \end{pmatrix}, \quad g_j(\vec{x}) = \begin{pmatrix} \frac{\partial f_{j,l_1}(\vec{x})}{\partial n_{j,l_1}} \\ \vdots \\ \frac{\partial f_{j,l_L}(\vec{x})}{\partial n_{j,l_L}} \end{pmatrix}. \tag{4.34}$$

The recovered gradient $\nabla \tilde{f}_j(\vec{x})$ then follows from

$$N_j^T N_j \nabla \tilde{f}_j(\vec{x}) = N_j^T g_j(\vec{x}). \tag{4.35}$$

The resulting RDG-1x$\tilde{f}$ scheme reads

$$\iint_{\Omega_j} v u_t \, dt = \oint_{\partial\Omega_j} (v \nabla f) \cdot \hat{n} \, d\partial\Omega - \iint_{\Omega_j} \nabla v \cdot \nabla \tilde{f} \, dxdy. \tag{4.36}$$

This system will have to be solved in all quadrature points used in evaluating the volume integral. In 1-D the RDG-1x$\tilde{f}$ scheme reduces to the RDG-1x$\bar{f}$ scheme, meaning they share the same stability issue.

In the next section we show a series of numerical experiements to demonstrate the performance of RDG schemes for linear problems on structured and unstructured grids. Experiment 1 demonstrates the accuracy of the RDG-2x interior scheme. Experiment 2 shows the performance of RDG schemes with the full boundary-recovered function. Experiment 3 shows the degenerate performance of RDG schemes on unstructured grid with the compact boundary-recovered function.

Experiment 1: Linear diffusion (unsteady, periodic BC, Cartesian grid)

Our first numerical experiment shows the performance of the interior RDG-2x scheme on a uniform 2-D Cartesian grid. This simple problem eliminates any possible complication coming from irregular grids, nonlinear diffusion fluxes, and boundary treatment. The solution is a time-accurate decaying wave,

$$u(x, y, t) = \sin(2\pi x) \sin(2\pi y) \, e^{-t}, \tag{4.37}$$

where $(x, y) \in [0, 1] \times [0, 1]$ and the final time is $t = 0.5$. The RDG-2x spatial discretization is coupled with a Runge-Kutta temporal discretization. Figure 4.6, 4.7 and 4.8 shows the L_2-error for $p = 1$, 2, and 3, respectively, for the cell average and the average first gradient. Since the problem is symmetric, the average first gradients in x and y directions are equal: $\overline{\Delta_x u} = \overline{\Delta_y u}$. The results are exactly the same as the 1-D case due to the nice decoupling of the diffusion fluxes in the x and y direction. The rate of error convergence for RDG-2x ($p = 1$) is 4th-order for the cell average and 5th-order for the average first gradient, for RDG-2x ($p = 2$) is 8th-order for the cell average and 7th-order for the first gradient, and for RDG-2x ($p = 3$) is 10th-order for the cell average and 11th-order for the first gradient. Note the weird dip in the error for the average first gradient for $p = 3$; this is the result of averaging the first gradient of a sine wave on a 2-by-2 grid which results in a zero averaged gradient. It is interesting to observe that odd-order RDG schemes have gradients that are outperforming the cell average as found in one dimension; see Section 3.4.2.

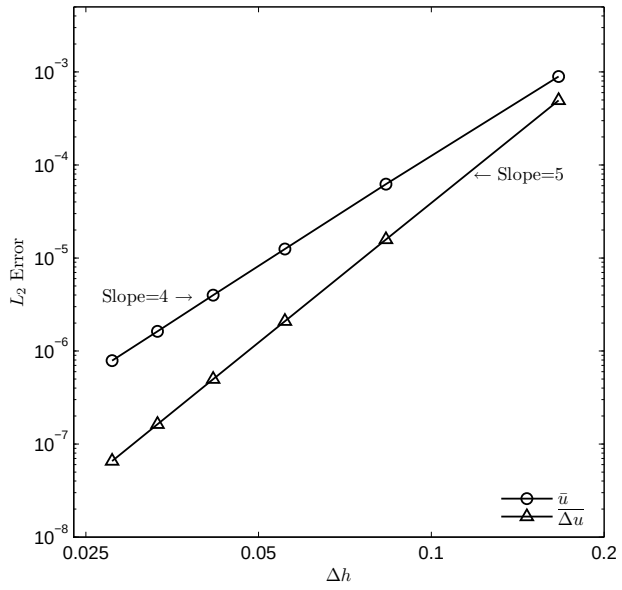

Figure 4.6: Experiment 1, RDG-2x ($p = 1$) for time-accurate problem with periodic boundary conditions. The order of error convergence is 4th-order for the cell average, and 5th-order for the averaged first gradient.

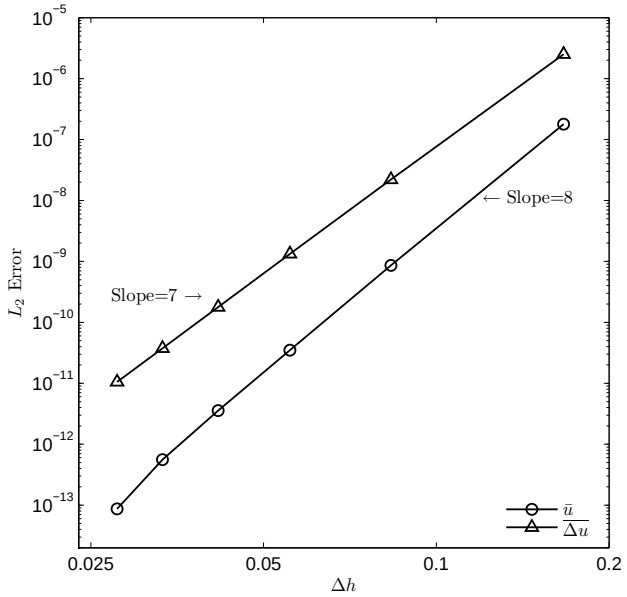

Figure 4.7: Experiment 1, RDG-2x ($p = 2$) for time-accurate problem with periodic boundary conditions. The order of error convergence is 8th-order for the cell average, and 7th-order for the averaged first gradient.

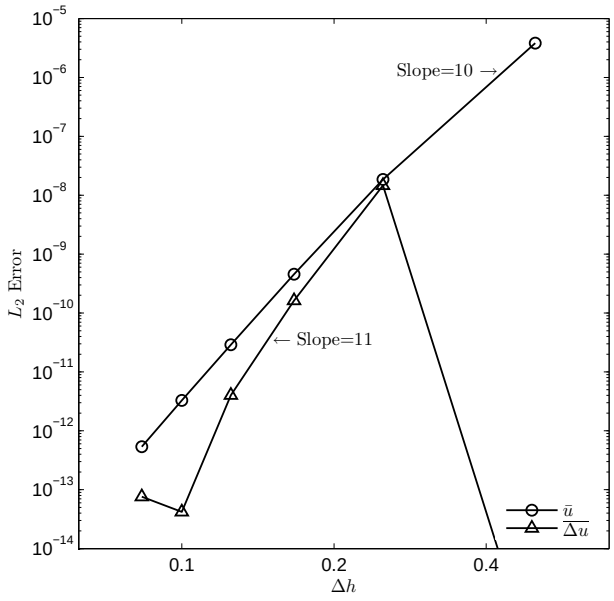

Figure 4.8: Experiment 1, RDG-2x ($p = 3$) for time-accurate problem with periodic boundary conditions. The order of error convergence is 10th-order for the cell average, and 11th-order for the averaged first gradient. The dip for the course grid is due to the average first gradient being zero on a 2 by 2 grid.

Experiment 2: Poisson problem (steady-state, full Dirchlet b.c, Cartesian grid, $p = 1$)

Figure 4.9 shows properties of the solution to the following 2-D Poisson problem on $[0,1] \times [0,1]$:

$$\begin{aligned} u(x,0) &= 0.5\cos(2\pi x), && (4.38)\\ u(x,1) &= 0.5\cos(2\pi x), && (4.39)\\ u(0,y) &= 0.5\cos(2\pi y), && (4.40)\\ u(1,y) &= 0.5\cos(2\pi y), && (4.41)\\ S(x,y) &= 2\pi^2\{\cos(2\pi x) + \cos(2\pi x)\}, && (4.42) \end{aligned}$$

where S is the source term. These Dirchlet boundary conditions are satisfied with the full boundary-recovered functions. The exact solution of the problem is

$$U(x,y) = 0.5\{\cos(2\pi x) + \cos(2\pi y) - 1\}. \tag{4.43}$$

The left graph in Figure 4.9 shows an intriguing aspect of RDG-2x in which the solution gradient is calculated with higher precision than the solution average. The results clearly show the full boundary-recovered function achieving the same level of accuracy as the interior RDG-2x scheme from the previous experiment. The right graph in Figure 4.9 shows a comparison of the L_2-error of the cell average for various RDG schemes with $p = 1$. RDG-1x-Naive and RDG-1x$\bar{f}$ are the under-performing schemes. The better schemes are classic RDG-2x and RDG-1x$\tilde{f}$, where 4th-order of accuracy is obtained. In terms of absolute error, RDG-2x is still the king of the RDG family for linear diffusion problems.

Experiment 3: Poisson problem (steady-state, compact Dirchlet b.c, structured/unstructured triangular grid)

We consider an almost structured irregular triangular grid and a structured orthogonal grid. Unfortunately, we must use the compact recovered function at the domain boundary for triangular grid. We solve a Dirichlet problem on the square $[0,1] \times [0,1]$ where the domain is discretized by randomly perturbing the nodes of a regular grid by 30% (see left frame of Figure 4.10). The careful choice of an exact steady state solution,

$$u(x,y) = x^4 + x^3y - 6x^2y^2 - xy^3 + y^4, \tag{4.44}$$

results in a zero source term. The right of Figure 4.10 shows the order of RDG-2x reduces to $p+1$ for unstructured grid. The result is expected due to the use of compact boundary-recovered functions; however, the L_1-error indicates that the same order of error is also committed by the interior scheme too.

Next, we solve a different Dirichlet problem taken from [30] on the same unit square with the following steady-state solution,

$$u(x,y) = e^{-0.1\sin(-5.1x+6.2y)+0.3\cos(4.3x+3.4y)}. \tag{4.45}$$

The orthogonal triangular grid and the results are shown in Figure 4.11. The results clearly show the order of accuracy of both RDG-1x-Naive and RDG-2x to be $p+1$. When compared to LDG, CDG, and BR2 in [30], RDG schemes typically perform better in terms of accuracy for higher p. For example, RDG is a factor four more accurate than CDG on the finest grid for $p=2$, and a factor of seven more accurate than CDG on the finest grid for $p=3$.

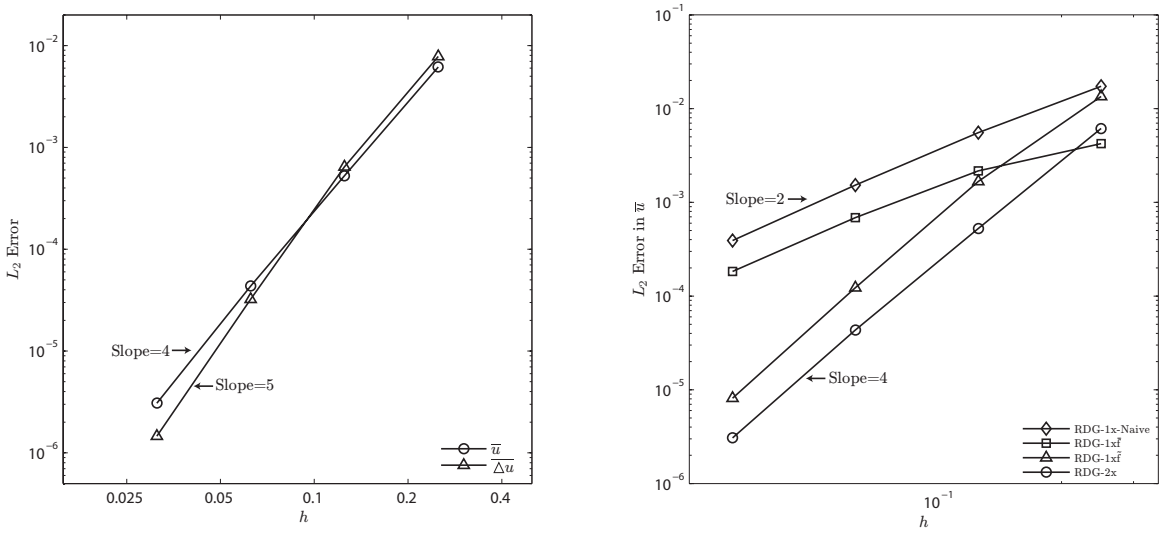

Figure 4.9: Experiment 2. Left: RDG-2x ($p=1$) for steady-state problem using full boundary-recovered function at the Dirichlet boundaries. Right: Comparison of various RDG ($p=1$) schemes with full boundary-recovered function.

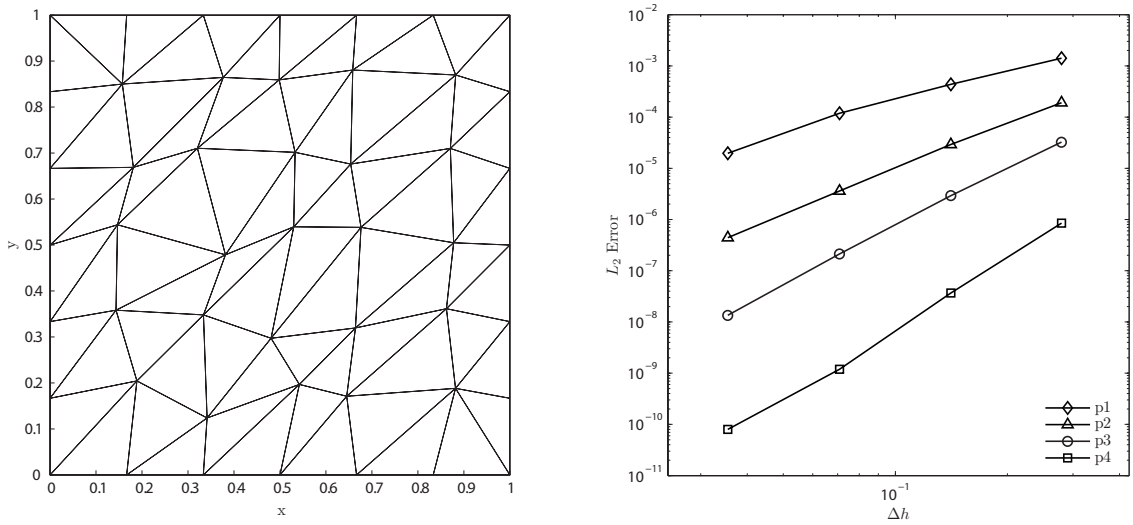

Figure 4.10: Experiment 3, RDG-2x ($p = 1, 2, 3, 4$) for steady-state problem using compact boundary-recovered function at the Dirichlet boundaries. A sample perturbed grid is shown on the left, and the graph on the right shows the order of accuracy of the scheme to be $p + 1$ on the irregular triangular grid.

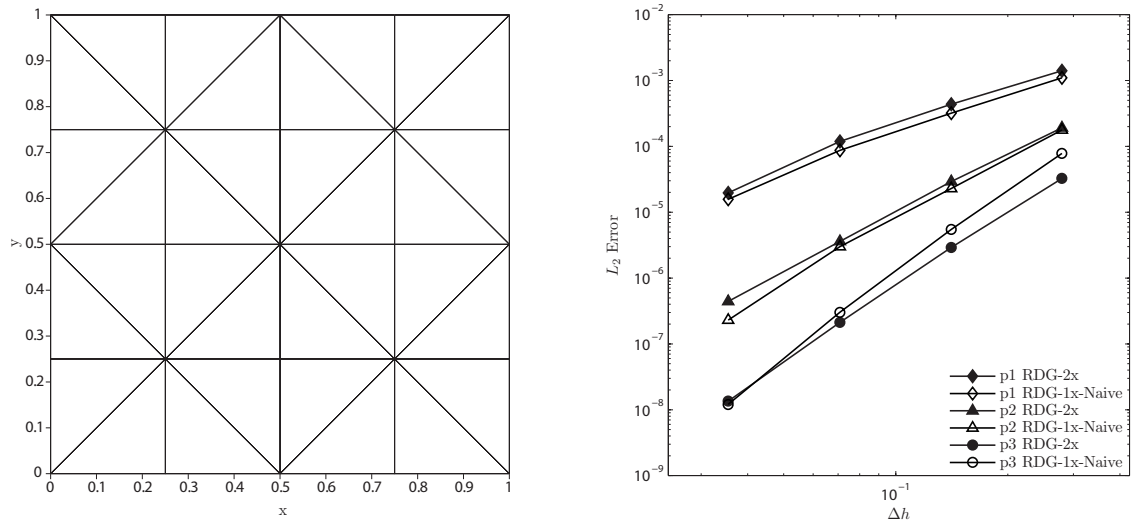

Figure 4.11: Experiment 3, RDG-2x and RDG-1x-Naive ($p = 1, 2, 3$) for steady-state problem using a compact boundary-recovered function at the Dirichlet boundaries. It appears that RDG-2x is only slightly better than RDG-1x on a triangular grid.

4.2.3 Nonlinearity and cross-derivatives in RDG schemes

In addition to nonlinearity, 2-D RDG schemes require new techniques to handle cross-derivatives properly. More precisely, the recovered function in the surface integral needs to be more accurate in the face-parallel direction; the steps to handle the volume integral remains the same. We present RDG-1x++ and RDG-1x++CO as improvement over RDG-1x+. Note that RDG-1x++CO is just an optimized version of RDG-1x++ for Cartesian grids.

RDG-1x+

We reconstruct a higher-order solution, $\hat{u}$, to replace the interior solution in the volume integral in the once partly integrated form of the PDE. We require $\hat{u}$ to share all original moments with u, and, in addition, to share all moments with f along the cell boundaries. The equations for solution enhancement in 2-D are as follows:

$$\int_{\Omega_i} (v_k)_i \; u_i \, d\Omega = \int_{\Omega_i} (v_k)_i \; \hat{u}_i \, d\Omega, \tag{4.46}$$

$$\oint_{\partial\Omega_i} (v_m)_i \; \hat{u}_{i,t} \, d\partial\Omega = \oint_{\partial\Omega_i} (v_m)_i \; f_j \, d\partial\Omega, \quad m = 1, .., (p+1), \tag{4.47}$$

In 2-D, $\hat{u}$ is required to satisfy $p+1$ moments of f along a line. The choice of basis functions for $\hat{u}$ is best illustrated by Figure 4.12. The positive x-axis represents increasing polynomial order in x, and the negative y-axis represents increasing polynomial order in y. Consider the solid-line square to represent the original basis functions of u. The data on the left and right boundaries of a cell add two $(p+1)$-element columns to the right of the square, while the top and bottom boundaries add two $(p+1)$-element rows to the bottom of the square. While v and u belong to the solution space V, $\hat{u}$ belongs to a larger solution space $\hat{V}$ such that $V \subset \hat{V}$.

RDG-0x+

This experimental scheme acts upon the original weak form of the PDE's without any integration by parts, and replaces u with a special enhanced solution, $\tilde{u}$,

$$\int_{\Omega_i} (v_k)_i \; u_{i,t} \, d\Omega \;\; = \;\; \int_{\Omega_i} (v_k)_i \, \nabla \cdot F\left(\tilde{u}_i, \nabla \tilde{u}_i\right) \, d\Omega. \tag{4.48}$$

The construction of $\tilde{u}$ is similar to that of $\hat{u}$; however, $\tilde{u}$ also shares the normal derivative, f_n, of f,

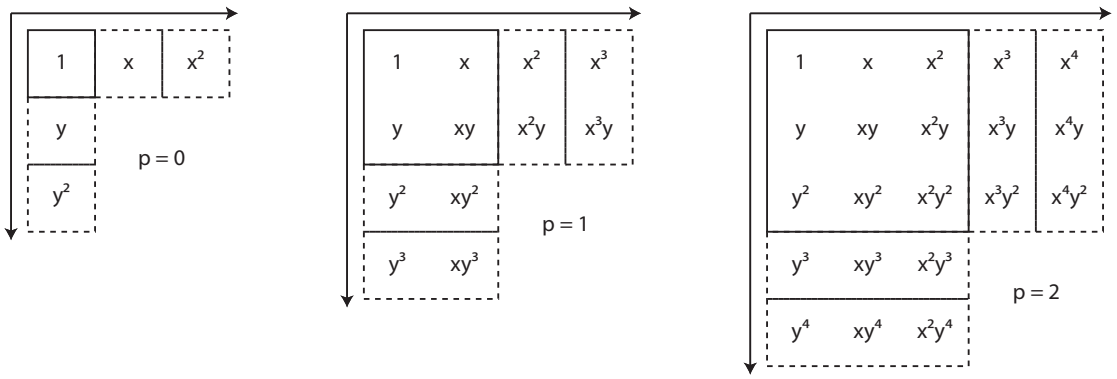

Figure 4.12: Basis functions for $\hat{u}$ in 2-D Cartesian grid for $p = 0, 1$, and 2 from left to right.

$$\int_{\partial\Omega_i} (v_m)_i \; \tilde{u}_{n,i} \, d\partial\Omega = \int_{\partial\Omega_i} (v_m)_i \; f_{n,j} \, d\partial\Omega, \quad m = 1, .., (p+1). \tag{4.49}$$

The main purpose of this scheme is to show that further enhancing of the solution in the face-normal direction does not help approximating a PDE with cross-derivatives. It is the solution in the surface integral that must be improved. In addition, the appearance of a method not requiring integration by parts is new in finite-element practice.

Experiment 4: nonlinear diffusion without cross-derivative (time-accurate, periodic b.c, Cartesian grid)

The following numerical experiment is designed to isolate the need for recovery-function enhancement, and focuses on solution enhancement only. We consider the scalar 2-D nonlinear diffusion equation with a source term,

$$u_t = e^{-u^2} \left(u_{xx} + u_{yy} \right) + S\left(t \right), \tag{4.50}$$

on the domain $x \in [0, 1]$, where $S\left(t \right)$ is a time-dependent source term determined by Mathematica software for the manufactured solution

$$u\left(x, t \right) = \sin\left(2\pi x \right) \sin\left(2\pi y \right) e^{-t}. \tag{4.51}$$

Table 4.3 shows the L_2-error of the cell average at $t = 1$ for RDG-1x+ and RDG-0x+. The RDG spatial discretizations are coupled with the 3rd, 4th, and 5th-order explicit Runge-Kutta temporal schemes for $p = 1, 2$, and 3, respectively. RDG-0x+ is slightly more accurate for $p = 2$ and 3, while both schemes obtain the same order of

accuracy as the original RDG-2x scheme for scalar diffusion. In this problem where the nonlinear diffusion equation does not contain any cross-derivatives, both schemes performed superbly; we next develop new techniques to handle cross-derivatives.

RDG-1x+			
p	Cells	L_2-error	O.O.A.
1	18×18	$1.24e-05$	
	24×24	$3.96e-06$	4.0
	30×30	$1.63e-06$	4.0
	36×36	$7.89e-07$	4.0
2	18×18	$1.50e-10$	
	24×24	$1.65e-11$	7.7
	30×30	$2.87e-12$	7.8
	36×36	$6.70e-13$	8.0
3	6×6	$8.66e-08$	
	8×8	$6.35e-09$	9.1
	10×10	$7.14e-10$	9.8
	12×12	$1.08e-10$	10.3

RDG-0x+			
p	Cells	L_2-error	O.O.A.
1	18×18	$1.23e-05$	
	24×24	$3.96e-06$	4.0
	30×30	$1.63e-06$	4.0
	36×36	$7.91e-07$	4.0
2	6×6	$5.01e-11$	
	12×12	$5.16e-12$	7.9
	18×18	$9.00e-13$	7.8
	24×24	$2.60e-13$	6.8
3	6×6	$2.46e-09$	
	8×8	$7.07e-11$	12.3
	10×10	$6.02e-12$	11.0
	12×12	$9.27e-13$	10.3

Table 4.3: Experiment 4, L_2-error of the cell average for the RDG-1x+ and RDG-0x+ schemes.

RDG-1x++ (Recovered-function enhancement)

Using the recovered function for the viscous fluxes in 1-D problems is sufficient; however, in multi-dimensional problems, the recovered function is not accurate in the face-tangential direction of the cell boundary. The need for an accurate representation of the solution's face-tangential derivatives, such as the ones appearing in the Navier-Stokes shear terms, is imperative for achieving overall high-order accuracy. Our newest scheme performs binary recovery on top of the new enhanced solution $\hat{u}$ from the previous section to obtain an enhanced recovered function $\hat{f}$ as shown in Figure 4.13.

As mentioned before, the subroutine for recovery is the same for any level of enhancement, with the equations for the enhanced recovered equation similiar to those of f:

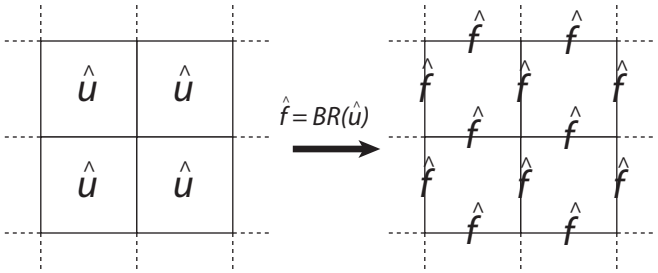

Figure 4.13: The recovered function is inaccurate in the face-tangential direction. We apply binary recovery on top of $\hat{u}$ to get an enhanced recovered function $\hat{f}$ to improve on the accuracy of f in the face-tangential directions.

$$\begin{aligned}
\int_{\Omega_L} (\hat{v}_k)_L \, \hat{u}_L d\Omega &= \int_{\Omega_L} (\hat{v}_k)_L \, \hat{f}_{(L,R)} d\Omega, \quad \forall k \text{ s.t. } \hat{v}_k \in \hat{V}, \\
\int_{\Omega_R} (\hat{v}_k)_R \, \hat{u}_R d\Omega &= \int_{\Omega_R} (\hat{v}_k)_R \, \hat{f}_{(L,R)} d\Omega, \quad \forall k \text{ s.t. } \hat{v}_k \in \hat{V}.
\end{aligned} \tag{4.52}$$

Here, $\hat{v}$ is a basis function of $\hat{V}$ (see Figure 4.12) and $\hat{f}$ belongs to a different function space $\hat{W}$,

$$\hat{f}_{(L,R)} = \sum_k \hat{b}_k \hat{\omega}_k, \quad , \hat{\omega} \in \hat{W}, \tag{4.53}$$

where $\hat{\omega}$ denotes the basis function for $\hat{f}$ (see [45] and Chapter 2 for more information about recovery bases). In Chapter 3, we have shown the Fourier-analysis results in Chapter 3 for the 1-D linear-diffusion schemes. At the end of this chapter, we show the 2-D Fourier-analysis results along for the other RDG schemes.

This extra layer of binary recovery comes at a hefty cost due to the increased stencil size, which effectively decreases the maximum allowable time-step in an explicit scheme, or increases connectivity cost in an implicit scheme. Figure 4.14 compares the stencil size of various RDG schemes. It is worth noting that both Compact DG (CDG) [30] and BR2 share the compact stencil of RDG-1x, but these schemes cannot handle PDE's with cross-derivatives at the $p = 0$ level. Clearly, cRDG is the most expensive of all the schemes present, while RDG-1x++CO is the optimal scheme to handle PDE's with cross-derivatives for all $p \geq 0$, if the grid is Cartesian.

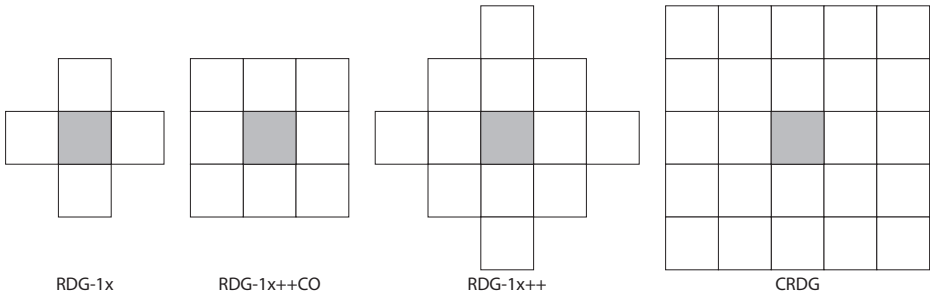

Figure 4.14: The 2-D stencils for various RDG schemes. Stencil size has direct influence on the time-step of explicit time-marching schemes, and also on the matrix density of implicit time-marching schemes.

RDG-1x++CO (Cartesian Optimization)

We draw our inspiration to optimize the RDG-1x++ scheme from dimension-splitting techniques found in [40, 39]. We recognize the solution-enhancement step for 2-D Cartesian grids can be factorized into 1-D steps. The convenience of such a factorization is twofold. Firstly, the same operator developed for 1-D is reusable for 2-D case; secondly, multi-dimensional codes reduce to a sequence of 1-D sweeps.

On a Cartesian grid the enhanced recovered function is too accurate in face-normal direction; hence we eliminate the extraneous information in the face-normal direction by using fewer moments in Eqn 4.52. Consider the stencils in Figure 4.15 used to obtain the enhanced recovered function for RDG-1x++ and RDG-1x++CO. The thick solid line indicates the interface of interest. Since we do not need more information in the face-normal direction, we can optimize RDG-1x++ by taking away the two cells furthest away from the interface in the face-normal direction. This optimization techniques require different sets of enhanced solutions $\hat{u}^x$ and $\hat{u}^y$, which are solution-enhanced in the x-direction and y-direction, respectively.

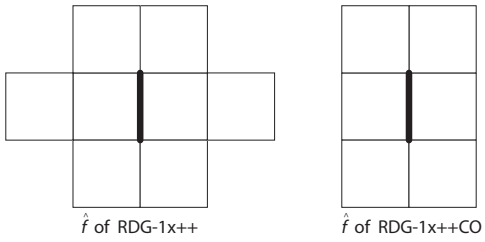

Figure 4.15: The stencils of the enhanced recovered function for RDG-1x++ and RDG-1x++CO on the left and right, respectively.

This section describes the steps to acquire an enhanced recovered function on a vertical interface from $\hat{u}^y$, and because this is a factorization technique, the same steps

can be applied to get an enhanced recovered function on a horizontal interface from $\hat{u}^x$. We first cycle through all cells to get a new enhanced solution $\hat{u}^y$. Figure 4.2.3 shows a $p = 1$ example beginning from the left. The subscripts M, L, and R, stand for middle, left and right, respectively. We require $\hat{u}^y_M$ to share all moments of u_M, and in addition, share the moments (without the y basis) of the recovered functions on the top and bottom faces of the cell. The resulting $\hat{u}^y_M$ will be enhanced in the y-direction only, as shown in the middle of Figure 4.2.3, where the solution within the cell now contains quadratic and cubic information in the y-direction. Our next step is to recover the enhanced recovered function, $\hat{f}$, from the vertically enhanced solutions on the left and right of an interface. Using the standard binary recovery technique with $\hat{u}^y_L$ and $\hat{u}^y_R$ as inputs, the resulting $\hat{f}$ is more accurate in both r and s directions (see right-most frame of Figure 4.2.3). For higher p, recovery in the face-normal direction will still be more accurate than in the face-parallel direction; nevertheless, the slight improvement in the s-direction is sufficient for high-order accuracy.

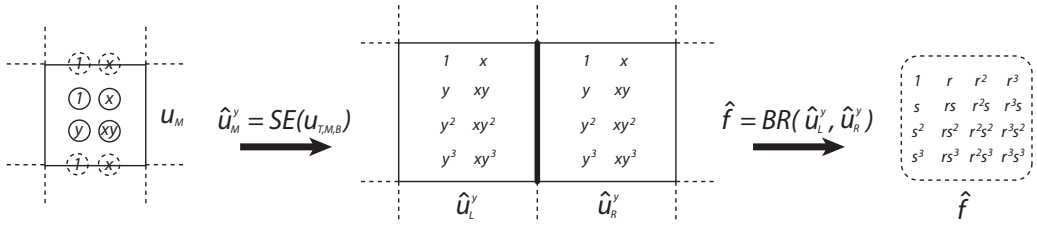

Figure 4.16: Reduced-accuracy y-recovery, followed by standard x-recovery, to create an enhanced recovered function $\hat{f}$ for use at an interface along the y-direction.

Experiment 5: diffusion-shear equation (time-accurate, periodic b.c, Cartesian grid)

This simple experiment adds a cross-derivative term to the linear diffusion equation[1],

$$u_t = u_{xx} + u_{yy} + u_{xy}. \tag{4.54}$$

Despite the simplicity of this equation, RDG-2x, RDG-1x-Naive, RDG-1x+, and RDG-0x+ completely fail to handle the cross-derivative term, resulting in an inconsistent (but stable) scheme. We consider the following time-accurate problem on the unit square with periodic boundary condition and the exact solution of

[1]When a shear term appears, one can always find a rotated frame in which the shear term vanishes. This theoretical insight does not help to solve the numerical problem of including influence of corner elements into the DG formulation.

$$u(x,y,t) = e^{-t}\left(\sin\left(2\pi\left(x-y\right)\right) + \sin\left(2\pi x\right)\sin\left(2\pi y\right)\right), \tag{4.55}$$

where the appropriate source term must be added to the RHS of Eqn 4.54, obtained with Mathematica. The L_2-error in the first two moments of $p = 1$, 2, and 3 are presented in Figure 4.17-4.19, respectively. The order of accuracy of RDG-1x++CO mimics the results of RDG-2x for the linear diffusion equation. RDG-1x++ underperforms in comparison to RDG-1x++CO for $p = 2$. This strange behavior is further explored in the following Fourier analysis.

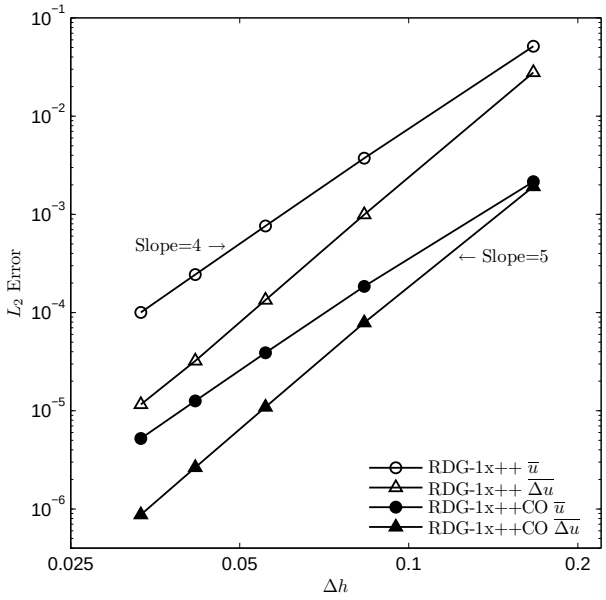

Figure 4.17: Experiment 5, RDG-1x++ and RDG-1x++CO ($p = 1$) for time-accurate problem with periodic boundary conditions. The order of error convergence is 4 for the cell average, and 5 for the averaged first gradient.

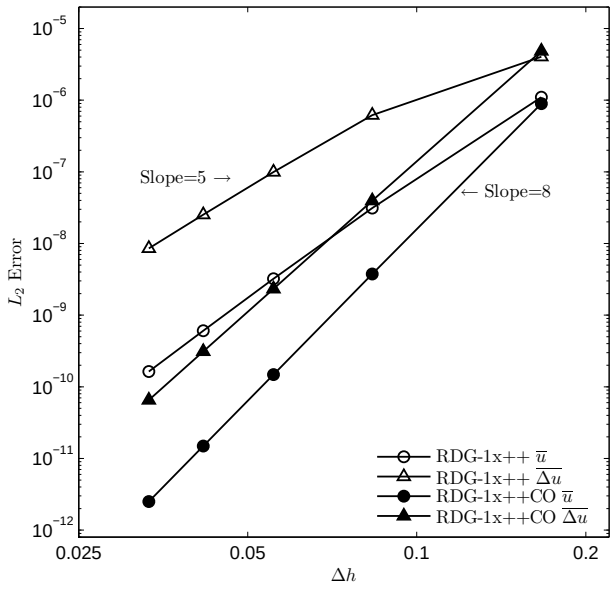

Figure 4.18: Experiment 5, RDG-1x++ and RDG-1x++CO ($p = 2$) for time-accurate problem with periodic boundary conditions. Notice the Cartesian optimized version performs extremely well.

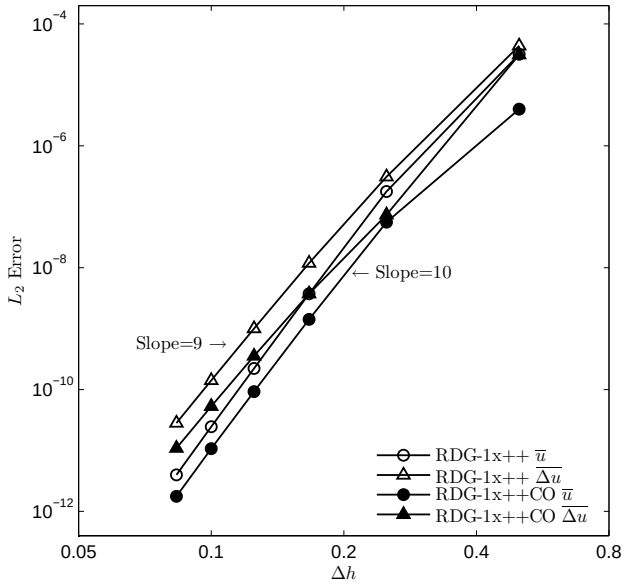

Figure 4.19: Experiment 5, RDG-1x++ and RDG-1x++CO ($p = 3$) for time-accurate problem with periodic boundary conditions. The order of error convergence is 10 for the cell average, and 9 for the averaged first gradient.

$(\max(\mathrm{Re}(\lambda)), \max(\mathrm{Im}(\lambda)), \text{O.O.A.})$

p	RDG-2x	RDG-1x+, RDG-1x++CO	RDG-1x++
0	(7.9/0/2)	(7.9/0/2)	(20.2/0/2)
1	(30.0/0/4)	(30.0/0/4)	(82.5/0/4)
2	(66.0/0/8)	(66.0/0/8)	(183.3/0/6)
3	(134.9/11.1/10)	(134.9/11.1/10)	(318.8/0/10)
4	(217.2/18.4/14)	(217.2/18.4/14)	(477.1/0/14)
5	(302.3/21.4/16)	(302.3/21.4/16)	(649.9/20.5/16)

Table 4.4: Fourier-analysis results for $\alpha = 0$ (pure Laplacian). Note that RDG-1x+, RDG-1x++CO, and RDG-2x are identical.

Numerical Fourier Analysis of 2-D RDG Schemes

Now that all the new RDG schemes have been revealed, we present Fourier-analysis results for all RDG schemes on the 2-D Cartesian grid. The scalar equation of interest is the Laplacian with a cross-derivative term,

$$u_t = u_{xx} + u_{yy} + \alpha u_{xy}, \tag{4.56}$$

where α is a constant, with the requirement $-2 \leq \alpha \leq 2$ for the PDE to be stable. The cross-derivative term is included to mimic the behavior of certain Navier-Stokes viscous terms. The numerical schemes should approximate the Fourier operator,

$$u_{xx} + u_{yy} + \alpha u_{xy} \simeq \frac{2\beta^2}{h^2} + \frac{\alpha\beta^2}{h^2}, \tag{4.57}$$

where β is the frequency of the Fourier mode assumed equal in the x- and y-directions, and h is the cell width. Table 4.4 shows the results for $\alpha = 0$ (pure Laplacian), and Table 4.5 shows the results for $\alpha = 1$. The three numbers shown are the largest real part of an eigenvalue $\max(\mathrm{Re}(\lambda))$, the largest imaginary part of an eigenvalue $\max(\mathrm{Im}(\lambda))$, and the order of accuracy. In designing diffusion schemes, we want the maximum real eigenvalue to be as small as possible to allow for a larger time-step and the imaginary component to be as close to zero as possible.

RDG-2x was first presented in 2005 and represents our best scheme for the scalar Laplacian ($\alpha = 0$), with the smallest real eigenvalue parts and the highest order of accuracy. For the scalar Laplacian, both RDG-1x+ and RDG-1x++CO reduce to the RDG-2x scheme. RDG-1x++ has higher real eigenvalues due to its larger stencil.

In the case with cross-derivative ($\alpha = 1$), the older generation RDG schemes, RDG-2x, RDG-1x, and RDG-1x+, fail to approximate the cross-derivative term for

$(\max(\mathrm{Re}(\lambda)), \max(\mathrm{Im}(\lambda)), \text{O.O.A.})$

p	RDG-1x++	RDG-1x++CO
0	(20.2/0/2)	(7.9/0/2)
1	(82.5/0/4)	(33.7/0.4/4)
2	(183.6/0.76/6)	(87.1/3.0/8)
3	(319.2/7.59/10)	(169.6/10.7/10)
4	(495.3/30.43/10)	(278.6/23.8/14)
5	(803.6/105.5/14)	(412.8/43.4/16)

Table 4.5: Fourier analysis results for $\alpha = 1$ (with cross-derivative). The maximum real eigenvalue of RDG-1x++CO is about half of that of RDG-1x++.

all p; they result in zeroth-order schemes. The only exception is for RDG-1x+ with $p = 1$, which achieves 4th-order of accuracy. Of the two remaining schemes, RDG-1x++CO allows for twice as large a time-step and obtains a higher order of accuracy for certain p.

4.3 Chapter summary

The appearance of cross derivatives in 2-D gives rise to new numerical challenges. In order to overcome this new problem, we present RDG-1x++ and RDG-1x++CO as candidates. The order of accuracy achieved by these schemes is equal to or higher than $2p + 2$. In particular, the RDG-1x++CO version achieves the same order of accuracy on a fully nonlinear problem with cross derivatives as the classic RDG-2x on a linear diffusion problem! Although the accuracy of RDG schemes is extremely high, the true beauty of RDG, as we stress once again, is in the simplicity of concept and implementation. Unlike other numerical methods with cookbook instructions, we have clearly identified the obstacles stemming from a nonlinear multidimensional problem: the nonlinear volume integral and the cross derivative in the flux integral. In this regard, we presented solution enhancement and recovery-function enhancement to remedy the respective issues.

We have independently studied the effect of irregular grids on RDG, and also the performance of RDG schemes for nonlinear diffusion problems. Superconvergence is extremely difficult to achieve on an irregular grid, and currently we have yet to devise a high-order boundary scheme for triangular grids. Our next step is to study the performance of these new schemes on irregular grids for nonlinear diffusion problems. The $2p + 2$ order of accuracy is expected to reduce to $p + 1$ on unstructured grids; therefore, the future RDG schemes for unstructured grids will likely utilize a more

compact stencil that sacrifices accuracy for efficiency.

CHAPTER V

Navier-Stokes Equations with RDG

The original equations for viscous flow were developed more than a century ago, however, these equations are too complicated to solve even with the most advanced computers today. There are actually many flow equations to decribe a viscous flow; the three main equations are the conservation of mass, momentum and energy. The other equations decribe physical phenomena such as chemical reactions and electromagnetic effects. These equations go through a rigorous and scientific process of simplification that results in the Navier-Stokes equations. The Navier-Stokes equations are only valid for a Newtonian viscous fluid in therodynamic equibrium; nevertheless, these equations are very practical and describes a wide range of flow phenonomon. For a detailed discussion of the Navier-Stokes equations we refer to White [46] .

5.1 1-D Navier-Stokes Equations with RDG

We present the Navier-Stokes equations in one dimension with advection component $\mathbf{F}$, and diffusion component $\mathbf{G}$,

$$\mathbf{U}_t + \mathbf{F}\left(\mathbf{U}\right)_x = \mathbf{G}\left(\mathbf{U}\right)_x, \tag{5.1}$$

where the conserved quantities, advection flux, and diffusion flux are,

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \tag{5.2}$$

$$\mathbf{F} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix}, \tag{5.3}$$

$$\mathbf{G} = \begin{pmatrix} 0 \\ \tau_{xx} \\ u\tau_{xx} - q_x \end{pmatrix}. \tag{5.4}$$

The advection component simply consists of the flux terms from the Euler equations, while the diffusion component includes both the viscous stress and the heat flux. Our numerical strategy is to apply an upwind flux to the advection component, and RDG to the diffusion component. It is easier to work with conserved variables for the sake of conservation; however, the quantities appearing in the diffusion flux are frequently given in terms of primitive variables. For the shear stress component, τ_{xx}, we assume a Newtonian fluid and apply Stokes' hypothesis ($\lambda = -\frac{2}{3}\mu$) to get an approximation,

$$\tau_{xx} = (2\mu + \lambda)\, u_x = \frac{4}{3}\mu u_x. \tag{5.5}$$

The heat flux q_x is given by Fourier's law,

$$q_x = -\kappa\mu \frac{\partial}{\partial x}\left(\frac{p}{\rho}\right), \tag{5.6}$$

$$\kappa = \frac{\gamma}{Pr\,(\gamma - 1)}, \tag{5.7}$$

$$Pr = \frac{4\gamma}{9\gamma - 5}, \tag{5.8}$$

where Pr is the approximate Prandtl number for a general case. Both shear-stress and heat-flux terms are functions of the viscosity coefficient $\mu\,(T)$. There are many available viscous models for Prandtl number and viscosity coefficient valid for different fluids and temperature ranges. For example, Sutherland's law is valid for atmospheric air with temperatures between $200K$ and $1000K$,

$$\mu\,(T) = \mu_0 \frac{T_0 + C}{T + C}\left(\frac{T}{T_0}\right)^{\frac{3}{2}}, \tag{5.9}$$

with the following physical constants,

$$\gamma = 1.4, \tag{5.10}$$

$$\mu_0 = 1.716 \times 10^{-5}[Pa \cdot s], \tag{5.11}$$

$$C = 110.5[K], \tag{5.12}$$

$$T_0 = 273.1[K], \tag{5.13}$$

$$R = 287[N \cdot m/kg]. \tag{5.14}$$

The other thermodynamic quantities of importance are given by,

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u^2\right), \tag{5.15}$$

$$H = E + \frac{p}{\rho}, \tag{5.16}$$

$$a = \gamma\frac{p}{\rho} = \gamma RT, \tag{5.17}$$

$$T = \frac{1}{R}\frac{p}{\rho}, \tag{5.18}$$

where p is the pressure, H is the total enthalpy, a is the speed of sound, and T is the temperature.

5.1.1 Discretization of 1-D Navier-Stokes viscous terms

We proceed to obtain the weak formulation of the 1-D Navier-Stokes equations by testing Eqn 5.1 with a test function v in space, apply integration by parts once on

both the advection flux and the diffusion flux,

$$\int v\mathbf{U}_t\,dx \;=\; -\int v\mathbf{F}_x dx + \int v\mathbf{G}_x\,dx, \tag{5.19}$$

$$\int v\mathbf{F}_x\,dx \;=\; \left[v\begin{pmatrix}\rho u\\ p+\rho u^2\\ \rho uH\end{pmatrix}\right] - \begin{pmatrix}0\\ \int v_x\left(p+\rho u^2\right)\,dx\\ \int v_x\left(\rho uH\right)\,dx\end{pmatrix},$$

$$\int v\mathbf{G}_x\,dx \;=\; \left[v\begin{pmatrix}0\\ \frac{4}{3}\mu u_x\\ \frac{4}{3}\mu u u_x + \kappa\mu\frac{\partial}{\partial x}\left(\frac{p}{\rho}\right)\end{pmatrix}\right] - \begin{pmatrix}0\\ \int \frac{4}{3}v_x\mu u_x\,dx\\ \int\left(\frac{4}{3}v_x\mu u u_x + v_x\kappa\mu\left(\frac{p}{\rho}\right)_x\right)\,dx\end{pmatrix}.$$

We use the following shorthand notation for the conservative variables $U_0=\rho$, $U_1=\rho u$, $U_2=\rho E$, and express quantities in the RHS of the weak formulation in terms of the conservative variables,

$$u \;=\; \frac{U_1}{U_0}, \tag{5.20}$$

$$p \;=\; (\gamma-1)\left(U_2-\frac{U_1^2}{2U_0}\right), \tag{5.21}$$

$$H \;=\; \frac{U_2+p}{U_0}. \tag{5.22}$$

Fortunately, these are the only terms we need to express in terms of the conservative variables since all other thermodynamic variables are related back to p and ρ. We now discretize the weak equations on a local element Ω_j with $x\in\left[x_{j-\frac{1}{2}},\,x_{j+\frac{1}{2}}\right]$; the transformation to local coordinates in space reads,

$$\xi \;=\; \frac{x-x_{j-\frac{1}{2}}}{\Delta x},\;\frac{\partial\xi}{\partial x}=\frac{1}{\Delta x}\equiv\frac{1}{h}.$$

The goal is to transform all global derivatives into local derivatives, then the governing equations reads,

$$\Delta x \int_{\Omega_j} v\mathbf{U}_t \, d\xi \;=\; -\int_{\Omega_j} v\mathbf{F}_\xi d\xi + \int_{\Omega_j} v\mathbf{G}_\xi \, d\xi, \tag{5.23}$$

$$\int_{\Omega_j} v\mathbf{F}_\xi \, d\xi \;=\; \left[v \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix} \right]_{\xi=0}^{\xi=1} - \int_{\Omega_j} v_\xi \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix} d\xi, \tag{5.24}$$

$$\begin{aligned} \int_{\Omega_j} v\mathbf{G}_\xi \, d\xi \;=\; & \frac{1}{h} \left[v \begin{pmatrix} 0 \\ \frac{4}{3}\mu u_\xi \\ \frac{4}{3}\mu u u_\xi + \kappa\mu \frac{\partial}{\partial \xi}\left(\frac{p}{\rho}\right) \end{pmatrix} \right]_{\xi=0}^{\xi=1} \\ & -\frac{1}{h} \begin{pmatrix} 0 \\ \int_{\Omega_j} \frac{4}{3} v_\xi \mu u_\xi \, d\xi \\ \int_{\Omega_j} \frac{4}{3} v_\xi \mu u u_\xi + \kappa \left(v_\xi \mu\right)_\xi \frac{p}{\rho} \, d\xi \end{pmatrix}, \end{aligned} \tag{5.25}$$

We may handle the computation of the $\mathbf{F}$ flux with an approximate Riemann solver; here, we focus our attention on the $\mathbf{G}$ flux. The recovery procedure generates three smooth functions, one for each conservative variable across the interface. However, acquiring the exact derivatives for terms like u_ξ, μ_ξ and $\left(\frac{p}{\rho}\right)_\xi$ is rather complicated, hence a numerical approach is preferred. We refer to Appendix A for a discussion on derivatives.

Numerical results for 1-D Navier-Stokes viscous terms

In order to bring to light the quality of RDG, this section focuses on just the diffusion terms in the Navier-Stokes equations. All of the numerical test cases are manufactured solutions with no physical meaning, but are intended to capture the mathematical properties of the Navier-Stokes equations and exemplify the ability of the numerical diffusion operators to solve the system.

This test case throws out the advection terms in the Navier-Stokes equation in order to focus on the diffusion terms. We keep the density variable constant since the

diffusion flux for density is zero. Our time-accurate solution is given by,

$$\begin{aligned} \rho(t) &= 2, && (5.26)\\ u(t) &= \sin(\pi x)\, e^{-t}, && (5.27)\\ P(t) &= 101000 + 10000 \sin(\pi x)\, e^{-t}, && (5.28) \end{aligned}$$

with Sutherland's law as the viscosity model with the following physical constants,

$$\gamma = 1.4, \tag{5.29}$$

$$\mu_0 = 1[Pa \cdot s], \tag{5.30}$$

$$C = 110.5[K], \tag{5.31}$$

$$T_0 = 273.1[K], \tag{5.32}$$

$$R = 287[N \cdot m/kg]. \tag{5.33}$$

Notice these constants are the same as those for atmospheric air with the exception of μ_0. The real value of μ_0 is too low; a higher value guarantees a larger change in the solution in a shorter amount of time. We solve the following system,

$$\begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}_t = \begin{pmatrix} 0 \\ \frac{4}{3}\mu u_x \\ \frac{4}{3}\mu u u_x + \kappa\mu \frac{\partial}{\partial x}\left(\frac{p}{\rho}\right) \end{pmatrix} + \begin{pmatrix} 0 \\ \mathrm{S1}(t) \\ \mathrm{S2}(t) \end{pmatrix}, \tag{5.34}$$

where S1 and S2 are source terms determined by inserting the time-accurate solution into the system of equations above, using Mathematica. In order the ascertain the quality of the manufactured solution, we numerically compared the values of the flux terms with the values of the source terms to ensure the source terms are not dominating the RHS. The conserved variables are recovered across the interface and then used to determine the nonlinear flux at the interface. The solution is marched forward in time using a 3rd-order Runge-Kutta scheme until $t = 3$. The numerical results for RDG-1x-Naive, RDG-1x$\bar{f}$, and RDG-1x+ for $p = 1$ and $p = 2$ are presented in Table 5.1 and 5.2. Note that only the L_2 error of the total-energy variable is shown;

the momentum variable behaves similarly.

For both $p = 1$, RDG-1x+ is clearly the fastest scheme; however, it is slightly less accurate than RDG-1x$\bar{f}$ in terms of absolute error. This may be due to the fact that RDG-1x$\bar{f}$ is taking much smaller time steps. The story is drastically different for $p = 2$, the order of accuracy of RDG-1x+ is two higher than that of RDG-1x$\bar{f}$. A numerical run time is not available because the experiments were done on a different set of grids.

Name/VNN	N_{ele}	$L_2\,\bar{U}_{2error}$	$L_2\,\overline{\triangle U}_{2error}$	Rate		Time(s)
RDG-1x-Naive	10	3.57	1.98			4.3
VNN = 0.08	20	0.936	0.25	1.9	3.0	33.0
	40	0.237	3.13×10^{-2}	2.0	3.0	259.6
	80	5.94×10^{-2}	3.91×10^{-3}	2.0	3.0	2073.4
RDG-1x$\bar{f}$	10	0.104	9.91×10^{-2}			5.3
VNN = 0.08	20	6.24×10^{-3}	3.13×10^{-3}	3.9	5.0	41.2
	40	4.38×10^{-4}	9.79×10^{-5}	4.0	5.0	324.0
	80	2.75×10^{-5}	3.06×10^{-6}	4.0	5.0	2591.0
RDG-1x+	10	0.425	0.188			2.8
VNN = 0.20	20	2.77×10^{-2}	6.07×10^{-3}	3.9	4.9	22.2
	40	1.75×10^{-3}	1.91×10^{-4}	3.8	4.9	173.0
	80	1.10×10^{-4}	5.98×10^{-6}	3.9	5.0	1332.2

Table 5.1: $p = 1$ results for RDG-1x-Naive, 1x$\bar{f}$, and 1x+. RDG-1x+ is clearly the fastest scheme, while its accuracy is on par with RDG-1x$\bar{f}$.

Name/VNN	N_{ele}	$L_2\,\bar{U}_{2error}$	$L_2\,\overline{\triangle U}_{2error}$	$L_2\,\overline{\triangle^2 U}_{2error}$	Rate		
RDG-1x-Naive	10	4.82	2.04	0.106			
VNN = 0.02	20	1.24	0.257	6.67×10^{-3}	1.9	2.9	3.9
	40	0.313	3.22×10^{-2}	4.18×10^{-4}	1.9	3.0	4.0
	80	7.84×10^{-2}	4.03×10^{-3}	2.16×10^{-5}	2.0	3.0	4.0
RDG-1x$\bar{f}$	10	7.13×10^{-3}	2.61×10^{-3}	2.63×10^{-3}			
VNN = 0.02	20	1.22×10^{-4}	2.20×10^{-5}	4.38×10^{-5}	5.9	6.9	5.9
	40	2.15×10^{-6}	1.76×10^{-7}	6.96×10^{-7}	5.8	7.0	6.0
	80	3.54×10^{-6}	2.02×10^{-9}	1.09×10^{-8}	-0.7	6.5	6.0
RDG-1x+	4	2.73×10^{-3}	1.47×10^{-1}	4.05×10^{-1}			
VNN = 0.12	8	1.32×10^{-5}	1.15×10^{-3}	7.46×10^{-3}	7.7	7.0	5.7
	12	5.33×10^{-7}	6.70×10^{-5}	6.72×10^{-4}	7.9	7.0	5.9
	16	5.88×10^{-8}	8.94×10^{-6}	1.21×10^{-4}	7.6	7.0	5.9

Table 5.2: $p = 2$ results for RDG-1x-Naive, 1x$\bar{f}$, and 1x+. RDG-1x+ is clearly the fastest and most accurate scheme.

We remind the reader that RDG-1x$\bar{f}$ is no longer stable for $p \geq 3$. RDG-1x-Naive remains a flat second-order scheme for both $p = 1$ and 2, making it no better or even worse than existing methods such as LDG and BR2. RDG-1x+ is superior in terms of accuracy and speed for nonlinear diffusion.

5.2 2-D Navier-Stokes

In this section we lay out the 2-D Navier-Stokes equations and the set of approximations of the shear and viscous stresses. The Navier-Stokes equations in 2-D require an additional momentum equation in the y-direction. The two different discretizations of advection and diffusion terms are sufficiently described in the 1-D Navier-Stokes section, hence we will not split them in this section; instead we will split the fluxes according to the x and y directions, denoting them by $\mathbf{F}$ and $\mathbf{G}$, respectively. The 2-D Navier-Stokes equations in conservative form are

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = 0, \tag{5.35}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \tag{5.36}$$

$$\mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uH - u\tau_{xx} - v\tau_{xy} + q_x \end{pmatrix}, \tag{5.37}$$

$$\mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv - \tau_{yx} \\ \rho v^2 + p - \tau_{yy} \\ \rho uvH - u\tau_{yx} - v\tau_{yy} + q_y \end{pmatrix}. \tag{5.38}$$

The viscous stresses in 2-D for a Newtonian fluid are given by,

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right), \tag{5.39}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \tag{5.40}$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \tag{5.41}$$

Applying Stokes's hypothesis ($\lambda = -\frac{2}{3}\mu$), the viscous stresses simplify to

$$\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right), \tag{5.42}$$

$$\tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right). \tag{5.43}$$

The heat fluxes are given by

$$q_x = -\frac{\gamma\mu}{Pr\,(\gamma-1)}\frac{\partial}{\partial x}\left(\frac{p}{\rho}\right), \tag{5.44}$$

$$q_y = -\frac{\gamma\mu}{Pr\,(\gamma-1)}\frac{\partial}{\partial y}\left(\frac{p}{\rho}\right). \tag{5.45}$$

The viscousity coefficient μ is determined from Sutherland's Law given in the previous sections. The primitive variable p, pressure, is given by,

$$p = (\gamma - 1)\left(\rho E - \frac{\rho u^2 + \rho v^2}{2}\right). \tag{5.46}$$

Next we test the accuracy of RDG schemes by ignoring the advection terms and only discretizing the viscous terms of the 2-D Navier-Stokes equations. In the last section we solve the complete 2-D Navier-Stokes equations.

5.2.1 2-D Navier-Stokes Viscous Terms

This section follows the same strategy as before where we use manufactured solutions to determine the order of accuracy of RDG schemes. We first cast the 2-D Navier-Stokes equations with an arbitrary source term into the weak formulation, and then discretize the system with RDG. We begin by testing the system with a test function v,

$$\iint_{\Omega_j} v \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}_t dxdy =$$

$$\iint_{\Omega_j} v \left(\begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}_x + \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} - q_y \end{pmatrix}_y + \mathbf{S} \right) dxdy; \tag{5.47}$$

after switching to local coordinates for Cartesian grids with $h = \Delta x = \Delta y$,

$$\begin{aligned} h^2 \iint_{\Omega_j} v \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}_t d\xi d\eta &= h \iint_{\Omega_j} v \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}_\xi d\xi d\eta \\ &\quad + h \iint_{\Omega_j} v \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}_\xi d\xi d\eta \end{aligned}$$

$$+ h^2 \iint_{\Omega_j} v\mathbf{S}\, d\xi d\eta. \tag{5.48}$$

The equations above are ready for the RDG-0x+ discretization. The quantities inside the volume integral on the RHS are calculated from $\tilde{\mathbf{U}}$ as described in Section 4.2.3. To obtain the form for RDG-1x(+), we apply integration by parts once and arrive at

$$\begin{aligned}
h^2 \iint_{\Omega_j} v \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}_t d\xi d\eta &= h \oint_{\partial\Omega_j} v \begin{pmatrix} 0 \\ \begin{pmatrix} \tau_{xx} \\ \tau_{xy} \end{pmatrix} \cdot \hat{n} \\ \begin{pmatrix} \tau_{yx} \\ \tau_{yy} \end{pmatrix} \cdot \hat{n} \\ \begin{pmatrix} u\tau_{xx} + v\tau_{xy} - q_x \\ u\tau_{yx} + v\tau_{yy} - q_y \end{pmatrix} \cdot \hat{n} \end{pmatrix}_\xi d\partial\Omega_j \\
&- h \iint_{\Omega_j} \begin{pmatrix} 0 \\ \begin{pmatrix} v_\xi \\ v_\eta \end{pmatrix} \cdot \begin{pmatrix} \tau_{xx} \\ \tau_{xy} \end{pmatrix} \\ \begin{pmatrix} v_\xi \\ v_\eta \end{pmatrix} \cdot \begin{pmatrix} \tau_{yx} \\ \tau_{yy} \end{pmatrix} \\ \begin{pmatrix} v_\xi \\ v_\eta \end{pmatrix} \cdot \begin{pmatrix} u\tau_{xx} + v\tau_{xy} - q_x \\ u\tau_{yx} + v\tau_{yy} - q_y \end{pmatrix} \end{pmatrix}_\eta d\eta d\xi \\
&+ h^2 \iint_{\Omega_j} v\mathbf{S}\, d\xi d\eta.
\end{aligned} \tag{5.49}$$

where $\hat{n} = \begin{pmatrix} \xi_n \\ \eta_n \end{pmatrix}$. The equations above are ready for RDG-1x+ discretization. The quantities inside the volume integral on the RHS are calculated from $\hat{\mathbf{U}}$ as described in Section 4.2.3, and the quantities inside the surface integral on the RHS are calculated from the interface recovery solution, $\mathbf{U}_f$.

Numerical Results for 2-D Navier-Stokes Viscous Terms

In order to isolate the numerical scheme for diffusion, we remove the Euler terms from the Navier-Stokes equations and use RDG for the viscous fluxes. As a result the

density equation drops out resulting in the following system:

$$\begin{pmatrix} \rho u \\ \rho v \\ \rho E \end{pmatrix}_t = \begin{pmatrix} \tau_{xx} \\ \tau_{yx} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}_x + \begin{pmatrix} \tau_{xy} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} - q_y \end{pmatrix}_y , \tag{5.50}$$

Our numerical test case is for a manufactured solution with approximate physics; therefore, the values of the physical constants (such as Prandtl number, gas constant, and specific-heat ratio) are fixed. The simplified equations are as follows. The shear stresses with Stokes' hypothesis are

$$\tau_{xx} = \frac{2}{3}\mu\left(2u_x - v_y\right), \tag{5.51}$$

$$\tau_{yy} = \frac{2}{3}\mu\left(2v_y - v_x\right), \tag{5.52}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(u_y + v_x\right), \tag{5.53}$$

and the heat fluxes are,

$$q_x = \frac{19}{4}\mu\left(\frac{P}{\rho}\right)_x , \tag{5.54}$$

$$q_y = \frac{19}{4}\mu\left(\frac{P}{\rho}\right)_y . \tag{5.55}$$

We use a viscousity coefficient similar to that of Sutherland's law,

$$\mu = \left(\frac{2}{T+1}\right) T^{\frac{3}{2}}, \tag{5.56}$$

where T is the temperature from the ideal gas law, $p = \rho RT$, with $R = 1$. We consider the following manufactured solutions,

$$\rho = U_0\left(x, y, t\right) = 2, \tag{5.57}$$

$$\rho u = U_1\left(x, y, t\right) = \sin\left(2\pi x\right)\sin\left(2\pi y\right)e^{-t}, \tag{5.58}$$

$$\rho v = U_2\left(x, y, t\right) = \sin\left(2\pi x\right)\sin\left(2\pi y\right)e^{-t}, \tag{5.59}$$

$$\rho E = U_3\left(x, y, t\right) = 5+\sin\left(2\pi x\right)\sin\left(2\pi y\right)e^{-t}, \tag{5.60}$$

and generate the appropriate source terms with Mathematica. We consider a periodic-boundary unit-square domain with $x \in [0, 1]$ and $y \in [0, 1]$. The RDG spatial discretizations are coupled with the 3rd-, 4th-, and 5th-order explicit Runge-Kutta tem-

poral schemes for $p = 1$, 2, and 3, respectively. Table 5.3 shows the L_2-error of the cell average of total energy for both RDG-1x++ and RDG-1x++CO at $t = 1$. These discretizations both achieve the same high orders of accuracy as RDG-2x for linear scalar diffusion, just as we wished to achieve. It is worth noting that RDG-1x++CO is roughly three to four times faster than RDG-1x++. In addition, RDG-1x++CO is of much higher order than RDG-1x++ for $p = 2$; the results for RDG-1x++CO is showing signs of saturation on the finer grids.

RDG-1x++			
p	Cells	L_2-error	O.O.A.
1	6×6	0.000544	
	12×12	$3.79e-05$	3.8
	18×18	$7.61e-06$	4.0
	24×24	$2.42e-06$	4.0
2	12×12	$1.91e-08$	
	16×16	$3.89e-09$	5.5
	20×20	$1.08e-09$	5.7
	24×24	$3.76e-10$	5.8
3	6×6	$1.32e-09$	
	8×8	$8.61e-11$	9.5
	10×10	$9.80e-12$	9.7
	12×12	$1.62e-12$	9.9

RDG-1x++CO			
p	Cells	L_2-error	O.O.A.
1	6×6	0.000487	
	12×12	$3.66e-05$	3.7
	18×18	$7.49e-06$	3.9
	24×24	$2.40e-06$	4.0
2	12×12	$1.17e-09$	
	16×16	$7.52e-11$	9.5
	20×20	$1.27e-11$	8.0
	24×24	$4.68e-12$	5.5
3	6×6	$3.27e-10$	
	8×8	$2.39e-11$	9.1
	10×10	$2.92e-12$	9.4
	12×12	$4.93e-13$	9.8

Table 5.3: L_2-error of the cell average of total energy for the RDG-1x++ and RDG-1x++CO schemes.

5.3 Chapter summary

We extended the concepts from the chapter on two-dimensional scalar equations to a system of equations. Our newest schemes, RDG-1x++ and RDG-1x++CO, were able to maintain the same high level of performance for the viscous terms of the Navier-Stokes equations. We recommend the use of RDG-1x++CO on a Cartesian grid due to its high order of accuracy and weaker time-step restriction. Our future research includes adding the advection terms to see the combined effect of the advection and diffusion operators.

CHAPTER VI

Hancock-Huynh Discontinuous Galerkin Method

The discontinuous Galerkin method (DG) is well known for its ability to handle complicated geometries, to achieve high efficiency on parallel machines, and to obtain an arbitrarily high order of accuracy. The method was introduced as a spatial discretization that increases the order of spatial accuracy by adding more information per cell. In 1973 Reed and Hill [32] demonstrated the spatial accuracy of DG schemes for steady-state problems; however, the way to implement an equally high-order temporal discretization for time-accurate problems remained an enigma until much later. In 1989 Shu and Cockburn[7] successfully coupled the then popular Runge-Kutta method with DG, thus introducing the first successful high-order time-accurate DG scheme, the Runge-Kutta discontinus Galerkin method (RK-DG). RK-DG is a semi-discrete method where the temporal and spatial discretizations are completely decoupled, therefore allowing the coding process to be modularized and CFD-algorithm developers to focus on just the spatial discretization. In simple language, RK advances in time by making a number of estimates of the solution (or stages) in the course of one time step, and then combining all of the estimates to take one full time step forward. Despite the elegance of the seperation of spatial and temporal discretization, RK-DG is not efficient due to the rapidly increasing number of stages needed for orders greater than 4, and the reduction in stability domain caused by the hidden increase in stencil size for one stage within the method.

Ruuth[35] introduced a special class of optimized RK schemes called Strong-Stability-Preserving Runge-Kutta (SSP-RK), and Gottlieb[16] showcased a fifth order SSP-RK scheme consisting of 9 stages with Courant-Frierich-Lewy number (CFL) of 2.69. Ruuth and Gottlieb showed that increasing the number of stages also increases

the CFL number. The extension to a high order SS-PRK is not automatic; implementation requires strict adherence to the massive tables of coefficients provided by the papers mentioned above. Perhaps the most disturbing aspect of this special class of RK methods is the lack of consideration for real engineering problems; the flux evaluation in each stage can be so expensive that it offsets the gain in CFL number. We learn from this case that it is both the CFL and the number of flux evaluations that determine the quality of a temporal scheme.

Lörcher, Gassner, and Munz[24] introduced an arbitrary high-order space-time expansion (STE-DG) method. The process of going to a higher order is fully automatic and the scheme allows for local time stepping. In a global time-stepping scheme like RK-DG, the time step is restricted by the smallest cell in the computational grid. A local time-stepping scheme allows each cell to take multiple timesteps of different size, hence significantly increasing the computational efficiency. Unlike RK-DG, STE-DG suffers a CFL reduction as the order of the solution polynomial increases, making it worse than RK-DG on uniform grids. Recently Huynh[18] revisited Van Leer's[37] scheme III and developed the "moment scheme." The moment scheme is strictly for $p = 1$ and it is for hyperbolic equations only. To avoid confusion with other uses of the word "moment," the name Hancock-Huynh DG (HH-DG) will be used.

A new generation of space-time methods is evolving from Hancock's observation and will be discussed in detail in the next section. Our research goal is to apply Hancock's observation and extend Huynh's moment scheme beyond $p = 1$. In addition, we want to extend these concepts to diffusion!

The key component of the new generation of DG schemes is to integrate the governing equation in both space and time. Since the integration is over both time and space, the temporal and spatial discretizations are no longer seperate! In this regard, the original solution must be modified to contain the time variable. Both HH-DG and STE-DG acquire a space-time expanded solution from the purely spatial initial discretization of the solution. The space-time expanded solution contains both temporal and spatial derivatives, which provides a complete estimate of the solution at later times. Traditionally, the space-time expansion is done with Cauchy-Kovalevasakya (CK) procedure, which is used in STE-DG. CK is an exact analytical procedure for determining temporal derivatives from the spatial derivatives based on the governing equations. However, the overhead for determining a space-time expanded solution might cost more than a few flux evaluations, making it computationally expensive. Huynh proposed a cost-efficient way to acquire the space-time expanded solution based on a local Runge-Kutta method (LRK), which can be viewed as a numerical

version of the CK procedure.

This chapter begins by reviewing Hancock's observation, which is then followed by three main sections: HH-DG for the advection operator, HH-DG for the diffusion operator, and HH-DG for the combined advection and diffusion operators. The advection section is presented first due to its simplicity; we want the HH-DG advection operator to mimic the exact shift operator. Within this section we cover the numerical techniques for acquiring the space-time expanded solution via the LRK procedure, and most importantly, the accurate evaluation of space-time integrals.

6.1 Hancock's Observation

In 1971 Steve Hancock[1] made an important observation regarding upwind schemes for hyperbolic equations, leading to the most efficient implementation of the second-order, Godunov-type MUSCL scheme[38]. The observation is illustrated in Figure 6.1, an (x,t)-diagram for the Euler equations showing cell Ω_j and neighbors $\Omega_{j\pm1}$. In an upwind finite-volume scheme the upwind bias comes from computing the interface fluxes with the Backward Method of Characteristics. The diagram shows the characteristics drawn backward in time from each interface starting at time $t^n+\tau$. The flow in this region is assumed to be subsonic in the positive direction, so there are two forward characteristics and a backward one. It is seen that at interface $x_{j-\frac{1}{2}}$ the forward characteristics bring information from cell Ω_{j-1} to the interface, while the backward characteristic brings information from cell Ω_j. Similarly, at interface $x_{j+\frac{1}{2}}$ the forward characteristics bring information from Ω_j, while the backward characteristic brings information from Ω_{j+1}.

This description stems from an interface-centered mindset. Hancock noticed that, if we switch to a cell-centered point of view, we see that inside cell Ω_j all three characteristic equations are used to update the solution to time-level $t^n+\tau$. These form a complete set of flow equations; thus, we may as well use any other complete set of flow equations to describe the evolution within this cell, for instance, conservative or primitive rather than characteristic variables. Staying in the cell-centered mindset, we may update the solution inside cell Ω_j to time $t^n+\tau$ using information from this cell only, as if it had no neighbors, as if the (polynomial) initial-value distributions of flow variables stretched beyond the cell. This is the core of Hancock's Observation.

Advancing the solution internally in all cells, in particular, in Ω_{j-1} and Ω_{j+1}, creates a discontinuity at each interface, in particular, at $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$. At time

[1] Then at Physics International, Hayward, CA.

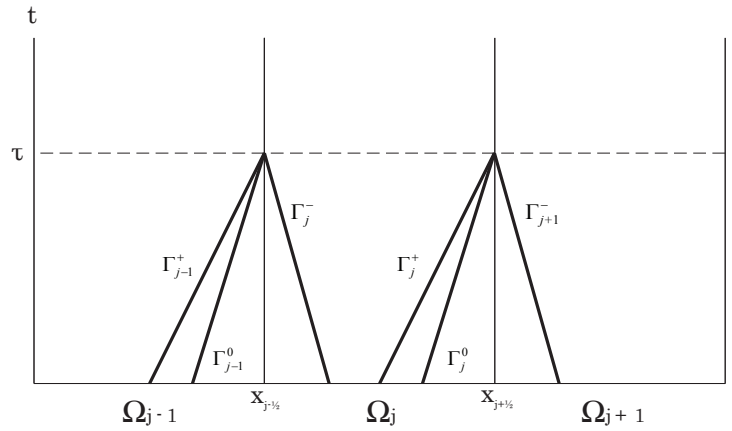

Figure 6.1: Hancock observes that the waves generated from the local evolution of two elements, Ω_j and Ω_{j+1}, result in the correct waves arriving at the element interface centered on $x_{j+\frac{1}{2}}$.

$t^n + \tau$ a unique flux vector may obtained at each interface by applying a Riemann solver to each discontinuity; this makes the flux upwind-biased. Hancock made his observation for a scheme with linear subcell data obtained by interpolation (a finite-volume method with $p = 1$); in this case an upwind flux computed at $t^n + \frac{1}{2}\Delta t$ suffices to update the conserved quantities to time $t^n + \Delta t$ with second-order accuracy. When applying this scheme to linear advection the upwind-biased Fromm scheme[13], the finite-volume version of Van Leer's Scheme III, appears.

Hancock's formulation of the $p = 1$ scheme was the first STE scheme *avant la lettre*. When making Hancock's Observation for an upwind-biased scheme of a higher order, it follows that Riemann fluxes are needed at various times with $\tau \leq \Delta t$, in order to compute a more accurate time-integral of the flux; as we shall see below, this is precisely the STE method, whether a finite-volume or a DG discretization is considered. It appears that Lörcher et al. were not aware of Hancock's version of MUSCL, or did not recognize its significance.

It is the merit of Huynh to have realized that for a space-time DG method to achieve superior accuracy and stability, the space-time volume integral must also be subjected to the influence of neighboring cells, rather than being computed solely from the interior STE solution.

We are now ready to discuss in detail the discretization of the space-time weak equations.

6.2 Space-time discontinuous Galerkin discretization for advection

We begin with a hyperbolic conservation law and progress towards the space-time weak formulation,

$$\boldsymbol{U}_t + \nabla \cdot \mathbf{F}(\boldsymbol{U}) = 0; \quad \boldsymbol{x} \in \mathbb{R},\, t > 0, \tag{6.1}$$

where $\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{x}, t)$ and $\boldsymbol{x} \in \mathbb{R}^m$. Since this is a Galerkin method, we let the solution $\boldsymbol{U}$ and the test function v be in the same solution space V that spans a polynomial space of degree p. We take the inner product of the hyperbolic conservation law with any test function $v = v(\boldsymbol{x})$ over both space and time, and then integrate by parts once,

$$\begin{aligned} \iint v\boldsymbol{U}_t \, dt d\boldsymbol{x} &= -\iint v \nabla \cdot \mathbf{F}(\boldsymbol{U}) \, d\boldsymbol{x} dt, \\ \int \left(\oint v\boldsymbol{U} \, dt - \int v_t \boldsymbol{U} \, dt \right) d\boldsymbol{x} &= -\int \left(\oint v\mathbf{F}(\boldsymbol{U}) \cdot \hat{n} \, d\boldsymbol{x} \right) dt. \\ &\quad + \int \left(\int \nabla \cdot v\mathbf{F}(\boldsymbol{U}) \, d\boldsymbol{x} \right) dt. \end{aligned} \tag{6.2}$$

Note that the test function is in space only ($v_t = 0$), which means our method is not a complete space-time DG method since the solution space does not involve time. Nevertheless, the integration in time marks a step closer to a more coupled space-time method. We cannot yet integrate the RHS of the equation because none of the terms has a clearly defined time dependence. Two major substitutions distinguish HH-DG from any other methods. First, we introduce the time variable into the RHS by replacing the generic $\boldsymbol{U}(\boldsymbol{x}, t)$ in the surface integral with the space-time expanded solution, $\boldsymbol{U}^{st}(\boldsymbol{x}, t)$. Secondly, we replace $\boldsymbol{U}(\boldsymbol{x}, t)$ in the volume integral with the updated solution, $\boldsymbol{U}^{n+\tau}(\boldsymbol{x}, t)$ and arrive at the final weak space-time formulation,

$$\begin{aligned} \int \left(\oint v\boldsymbol{U} \, dt - \int v_t \boldsymbol{U} \, dt \right) d\boldsymbol{x} &= -\int \oint v\mathbf{F}\left(\boldsymbol{U}^{st}(\boldsymbol{x}, t)\right) \cdot \hat{n} \, d\boldsymbol{x} dt \\ &\quad + \iint \nabla \cdot v\mathbf{F}\left(\boldsymbol{U}^{n+\tau}(\boldsymbol{x}, t)\right) \, d\boldsymbol{x} dt. \end{aligned} \tag{6.3}$$

Note we have not yet begun our discretization. The discussion of the space-time expanded solution begins in the next section, but perhaps the immediate question on the reader's mind now is the definition of $\boldsymbol{U}^{n+\tau}(\boldsymbol{x}, t)$. In short, if we let V be spanned by an orthogonal basis (i.e. Legendre polynomials), the resulting volume

integral in the RHS of Eqn 6.3 becomes explicit. The update equation for the zeroth moment does not contain a volume integral, hence the zeroth moment is first updated. The update equation for the first gradient will use the newest result from the zeroth moment. The choice of an orthogonal basis creates a hierarchical structure in which the update equation of the n-th moment depends on the solution of the $(n-1)$-th or lower moments. This will also be explained in detail in the next section.

6.2.1 One-dimension linear advection

We begin with the simplest equation to demonstrate the discretization of the RHS of Eqn 6.3. Consider the scalar linear advection equation,

$$u_t + a u_x = 0; \;\; x \in \mathbb{R},\, t > 0. \tag{6.4}$$

For element Ω_j with $x \in \left[x_{j-\frac{1}{2}},\, x_{j+\frac{1}{2}}\right]$ and at time $t = t_n$, the transformation to local coordinates in both space and time reads,

$$\begin{aligned} \xi &= \frac{x - x_{j-\frac{1}{2}}}{\Delta x}, \\ \tau &= \frac{t - t_n}{\Delta t}, \end{aligned} \tag{6.5}$$

with both $\xi, \tau \in [0,1]$. We acquire the space-time DG formulation by taking the inner product of Eqn(6.4) with a test function $v(\xi)$ and integrate by parts over element Ω_j,

$$\begin{aligned} \Delta x \int \left(\oint vu \, d\tau \right) d\xi &= -a\Delta t \int \left(\oint vu \cdot \hat{n} \, d\xi - \int v_\xi u \, d\xi \right) d\tau, \\ \Delta x \int v \left(u^{n+1} - u^n \right) d\xi &= -a\Delta t \int \left(vu \mid_{\xi=1} - vu \mid_{\xi=0} - \int v_\xi u \, d\xi \right) d\tau, \end{aligned}$$

We isolate the unknown u^{n+1} on the LHS,

$$\int v u^{n+1} d\xi = \int v u^n d\xi - \frac{a\Delta t}{\Delta x} \int \left(vu \mid_{\xi=1} - vu \mid_{\xi=0} - \int v_\xi u \, d\xi \right) d\tau. \tag{6.6}$$

In the RHS, u^n is simply the old solution, while the solutions in the surface and volume integrals are yet to be defined. We now introduce a space-time expanded solution u^{st}, as a replacement for u in the surface integral and the updated solution

$u^{n+\tau}$, as a replacement of u in the volume integral,

$$\int v u^{n+1} d\xi = \int v u^n d\xi - \frac{a\Delta t}{\Delta x} \int \left(v u^{st} \mid_{\xi=1} - v u^{st} \mid_{\xi=0} - \int v_\xi u^{n+\tau} \, d\xi \right) d\tau. \quad (6.7)$$

Our goal for the advection operator is to mimic the exact shift operator, which is a highly desirable property when solving for advection-dominated problems. This minimizes dissipation and dispersive errors, ensuring accurate propagation of waves over long distances. We shall discuss the exact shift operator for advection, and then design u^{st} and $u^{n+\tau}$ to mimic its behavior.

Exact shift operator for DG in 1-D

The term "shift" refers to the solution being advected over a finite distance; the term "exact" remains ambiguous at this moment. In order to clarify this ambiguity, the section is partitioned in two: the exact shift operator and the exact projected shift operator. The Courant number is universally defined as

$$\nu = a \frac{dt}{dx}. \quad (6.8)$$

The exact shift operator can only be numerically achieved with $\nu = 1$; while for $\nu < 1$, an extra projection step must be applied to obtain the exact projected shift operator.

Exact shift operator

A numerical method that results in the exact shift operator when $\nu = 1$ is certainly desirable because such a method exhibits zero dissipation and phase error while advecting waves over long distance and time. For linear advection with $a > 0$, the solution at a new time level is shifted to the right as shown in Figure 6.2. As a result, the update equation is simply

$$u_j^{n+1} = u_{j-1}^n, \quad a > 0. \quad (6.9)$$

Exact Projected Shift Operator

When shifting the solution with $\nu < 1$, the discontinuity between meshes Ω_j and Ω_{j-1} rests inside of Ω_j. As a result, an extra projection step must be taken to get the solution back into the space V. In this piecewise-linear ($p = 1$) example shown

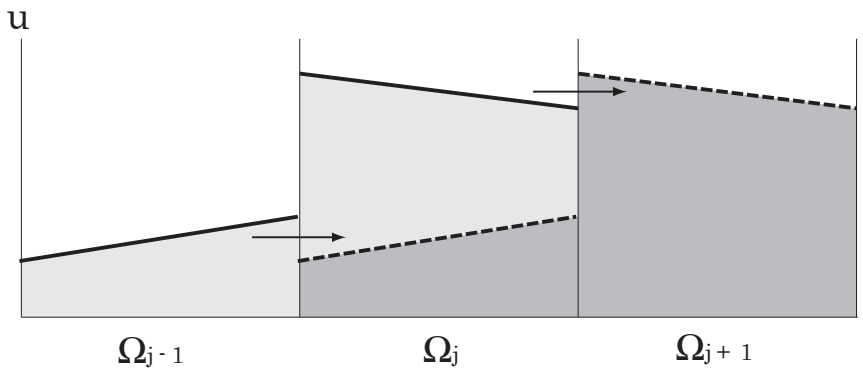

Figure 6.2: The exact shift operator occurs when $\nu = 1$. The solution of Ω_j at $t = t_0 + \Delta t$ is equal to the solution of Ω_{j-1} at $t = t_0$.

in Figure 6.3, the dashed line indicates the location of the shift solution, $u_j^{shifted}$. We find a new projected solution by enforcing for all v in V,

$$\int_{\Omega_j} v u_j^{n+1} dx = \frac{1}{\Delta x} \int_{\Omega_A} v u_j^{shifted} d\xi + \frac{1}{\Delta x} \int_{\Omega_B} v u_j^{shifted} dx, \tag{6.10}$$

where the intervals are defined in terms of local coordinate to be $\Omega_A = [0\ \nu]$, and $\Omega_B = [\nu\ 1]$. Note that the result of this type of projection is no longer exact, but this is the best one can achieved with a shift less than one mesh.

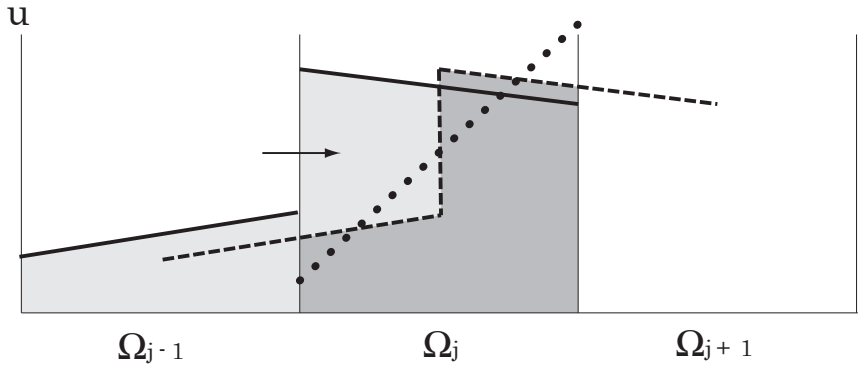

Figure 6.3: For $\nu < 1$, the subcell shift causes the original discontinuity at the interface to be shifted to the interior of Ω_j. The new solution of Ω_j (dotted line) is now acquired by projecting the discontinuous solution $u_j^{shifted}$ into the solution space.

Example of shift operators in matrix form

In this simple example, we show the exact projected shift operator for $a > 0$ up to $p = 2$. The new solution of Ω_j at $t = t_0 + \Delta t$ is a linear combination of the old solution of Ω_{j-1} and Ω_j at $t = t_0$,

$$\mathbf{u}_j^{t_0+\Delta t} = \mathbf{M}_L \mathbf{u}_{j-1}^{t_0} + \mathbf{M}_C \mathbf{u}_j^{t_0}, \tag{6.11}$$

where $\mathbf{u}$ is a column vector of coefficients of u starting with the lowest moment from top to bottom. The solution u is spanned by the orthogonal Legendre polynomials,

$$u = \overline{u} + \overline{\Delta u}\,(2\xi - 1) + \overline{\Delta^2 u}\left(6\xi^2 - 6\xi + 1\right). \tag{6.12}$$

The exact projected-shift matrices $\mathbf{M}_L$ and $\mathbf{M}_C$ are expressed below. For $p = 0$,

$$\mathbf{M}_L = [\nu]\,,\ \mathbf{M}_C = [1 - \nu]\,. \tag{6.13}$$

For $p = 1$,

$$\mathbf{M}_L = \begin{bmatrix} \nu & \nu - \nu^2 \\ -3\nu + 3\nu^2 & -3\nu + 6\nu^2 - 2\nu^3 \end{bmatrix},\ \mathbf{M}_C = \begin{bmatrix} 1 - \nu & -v + v^2 \\ 3\nu - 3\nu^2 & 1 - 3\nu + 2\nu^3 \end{bmatrix}. \tag{6.14}$$

For $p = 2$,

$$\begin{aligned} \mathbf{M}_L &= \begin{bmatrix} \nu & \nu - \nu^2 & \nu - 3\nu + 2\nu^3 \\ -3\nu + 3\nu^2 & -3\nu + 6\nu^2 - 2\nu^3 & -3\nu + 12\nu^2 - 12\nu^3 + 3\nu^4 \\ 5\nu - 15\nu^2 + 10\nu^3 & 5\nu - 20\nu^2 + 20\nu^3 - 5\nu^4 & 5\nu - 30\nu^2 + 50\nu^3 - 30\nu^4 + 6\nu^5 \end{bmatrix}, \\ \mathbf{M}_C &= \begin{bmatrix} 1 - \nu & -\nu + \nu^2 & -\nu + 3\nu - 2\nu^3 \\ 3\nu - 3\nu^2 & 1 - 3\nu + 2\nu^3 & -3\nu + 6\nu^2 - 3\nu^4 \\ -5\nu + 15\nu^2 - 10\nu^3 & 5\nu - 10\nu^2 + 5\nu^4 & 1 - 5\nu + 10\nu^3 - 6\nu^5 \end{bmatrix}. \end{aligned} \tag{6.15}$$

Note that for $\nu = 1$, $\mathbf{M}_C$ becomes the zero matrix, and $\mathbf{M}_L$ becomes the exact shift operator. To the knowledge of the author, no time-marching method coupled with DG spatial discretization for $p \geq 1$ is stable with $\nu = 1$, hence the exact shift operator is rarely achieved in practice. We will soon show HH-DG is the exact projected shift operator for $\nu < 1$ and the exact shift operator for $\nu = 1$. But before we get excited, we first define the space-time expanded solution.

Space-time expanded solution

HH-DG treats the surface integral on the RHS with special care; the spatial solution $u_j(\xi)$ is replaced with a space-time expanded solution $u_j^{st}(\xi,\tau)$. For the scalar linear-advection equation, determining u_j^{st} via the Cauchy-Kovalevskaya (CK) procedure is straightforward. First the solution is Taylor-expanded in both space and time (to 2nd-order in this example),

$$\begin{aligned} u(x,t) &= u_j + u_{x,j}(x-x_j) + u_{t,j}(t-t_n) + \frac{u_{xx,j}}{2}(x-x_j)^2 \\ &\quad + u_{xt,j}(x-x_j)(t-t_n) + \frac{u_{tt,j}}{2}(t-t_n)^2, \end{aligned} \tag{6.16}$$

and then the CK prodecure automatically express all temporal derivatives in terms of spatial derivatives based on the governing equation. The higher order derivatives in t are determined using a hierarchical procedure starting with the lowest order derivative,

$$\begin{aligned} u_t &= -au_x, \\ u_{tt} &= (-au_x)_t = a^2 u_{xx}, \\ u_{xt} &= (u_x)_t = -au_{xx}. \end{aligned}$$

The CK procedure becomes extremely complicated for nonlinear systems of equations as experienced by Lörcher[24] while experimenting with STE-DG. The need to solve for these space-time derivatives efficiently means a numerical estimation is preferred over an analytical one.

We introduce Huynh's approximation to u_j^{st} using a local Runge Kutta (LRK) procedure. The term local implies Eqn (6.6) is solved by only considering the local value of u_j. Figure (6.4) reveals the definition of a local flux. The solution in Ω_j is assumed to extend beyond the element boundary in both directions, hence the characteristic lines are continuous across the element interfaces. The solution on the interface is now uniquely defined; therefore there is no need for an approximate Riemann solver. LRK advances the solution to Radau points (see Appendix A) in a sequential manner, and the solution at these Radau points are stored. Figure(6.4) shows the location three Radau points of a HH-DG ($p=2$) scheme.

Notice we only need to know the values of u_j^{st} at the element interfaces. We briefly discuss the technical details of the LRK procedure.

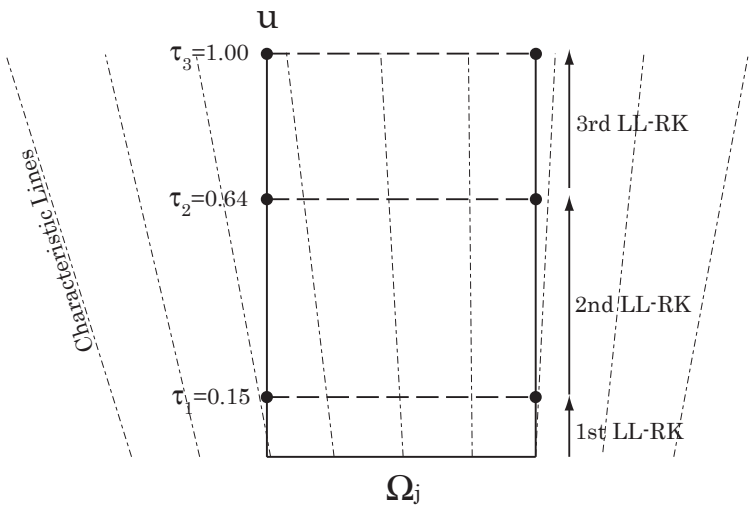

Figure 6.4: HH-DG ($p = 2$) using local Runge-Kutta to obtain u^{st}. Dashed lines indicates location of stored space-time solution values. The lightly dotted lines are characteristics from Ω_j and they illustrate an important property of "locality." For LRK, these characteristics are assumed to be valid outside of Ω_j, hence the function values of u on the boundaries are uniquely defined.

Local linear Runge-Kutta

From our numerical experiments we discovered a linear LRK (LL-RK) procedure is sufficient for the determination of space-time derivatives for both linear and nonlinear governing equations. A linear RK scheme is very low on storage requirement, and the use of local fluxes makes it extremely economical. The K-th order linear RK scheme for a system of variables comes from the nested expansion of $e^{\lambda t}$, where λ is the eigenvalue of the spatial operator,

$$\begin{aligned}
U^{(1)} &= U^n + \frac{\Delta t}{K}\mathrm{Res}\left(U^n\right), \\
U^{(2)} &= U^n + \frac{\Delta t}{K-1}\mathrm{Res}\left(U^{(1)}\right), \\
&\vdots \\
U^{(i)} &= U^n + \frac{\Delta t}{K-i+1}\mathrm{Res}\left(U^{(i-1)}\right), \\
&\vdots \\
U^{(K)} &= U^n + \frac{\Delta t}{1}\mathrm{Res}\left(U^{(K-1)}\right),
\end{aligned} \tag{6.17}$$

where Res() is the residual. For the linear advection equation, the residual is

$$\text{Res}\left(u_j\right) = -\frac{a}{\Delta x}\left(vu_j\mid_{\xi=1} - vu_j\mid_{\xi=0} - \int v_\xi u\, d\xi\right). \tag{6.18}$$

We emphasize here again that the residual of u_j is only a function of u_j itself. Referring back to our HH-DG ($p = 2$) example, we use a 5th-order LL-RK to advance from $\tau = 0$ to $\tau = 0.15$, then from $\tau = 0.15$ to $\tau = 0.64$, and finally from $\tau = 0.64$ to $\tau = 1$. This ensures the space-time solution is globally 5th-order at the three Radau points. We repeat this procedure for all cells in the computational domain and stored their space-time expanded solution. Now the solutions at all Radau points are known, we proceed to evaluate the surface integral.

Surface integral of the RHS with u^{st}

The space-time expanded solution is used exclusively to evaluate the surface integral of the flux over time. This surface integral requires a unique function value of $u^{st}_{j+\frac{1}{2}}$ at the element interface; however, the value of $u^{st}_{j+\frac{1}{2}}$ is multi-valued due to the discontinuity of the solution. The DG community borrows the approximate Riemann solver from finite-volume methods by replacing $u^{st}_{j+\frac{1}{2}}$ with a numerical flux, $\hat{u}^{st}_{j+\frac{1}{2}} = \hat{u}_{j+\frac{1}{2}}\left(u^{st}_j, u^{st}_{j+1}\right)$. Returning to our $p = 2$ example, consider the space-time expanded solution in elements Ω_j and Ω_{j+1} in Figure 6.5. We use a flux solver at each of the Radau points to acquire an unique numerical flux $\hat{u}^{st}_{j+\frac{1}{2}}$. The number of flux evaluations is equal to the number of Radau points in a HH-DG scheme.

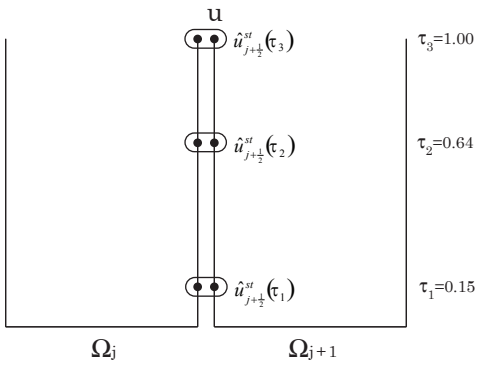

Figure 6.5: At the interface between two space-time expanded solutions (represented by solid dots), an approximate Riemann solver (indicated by ellipses) is applied at the Radau points to acquire unique flux values at each time level.

We use the upwind flux for linear advection with $a > 0$,

$$\hat{u}_{j+\frac{1}{2}} = u_j^{st}\left(\xi = 1, \tau\right). \tag{6.19}$$

Applying this to Eqn.(6.6) results in

$$\begin{aligned}\int v u^{n+1} d\xi &= \int v u^n d\xi - \frac{a\Delta t}{\Delta x}\left(\underbrace{\int \left(v\left(1\right) u_j^{st}\left(\xi = 1, \tau\right) - v\left(0\right) u_{j-1}^{st}\left(\xi = 1, \tau\right)\right) d\tau}_{\text{surface integral}}\right) \\ &\quad + \frac{a\Delta t}{\Delta x}\left(\underbrace{\iint v_\xi u_j^{n+\tau}\, d\xi d\tau}_{\text{volume integral}}\right). \end{aligned} \tag{6.20}$$

The last component to be discussed is the volume integral. In addition, we shall discuss a numerical technique to integrate in time based on the Radau points.

Volume integral of RHS based on $u_j^{n+\tau}$

The correct evaluation of the volume integral in Eqn (6.20) is essential for high-order accuracy. In the volume integral of STE-DG, $u_j^{n+\tau}$ is simply taken as u_j^{st}; this results in a lower-order scheme with severely restricted CFL number. The use of u_j^{st} in the volume integral implies no information with neighboring elements penetrates into the interior of Ω_j; the omission of wave interaction from neighboring elements results in inaccurate physics. The HH-DG scheme takes advantage of the orthogonality of the Legendre polynomials and incorporates wave interaction into the volume integral by using the newest solution available. Let us illustrate this through a $p = 3$ example'; consider u at $t = t_n$ to be constructed out of Legendre polynomials up to the cubic component,

$$v_0 = 1,$$

$$v_1 = 2\xi - 1,$$

$$v_2 = 6\xi^2 - 6\xi + 1,$$

$$v_3 = 20\xi^3 - 30\xi^2 + 12\xi + 1,$$

$$u_j^n(\xi) = \overline{u}\, v_0 + \overline{\Delta u}\, v_1 + \overline{\Delta^2 u}\, v_2 + \overline{\Delta^3 u} v_3, \tag{6.21}$$

where $\overline{u}$, $\overline{\Delta u}$, $\overline{\Delta^2 u}$, and $\overline{\Delta^3 u}$ are the cell average, first average gradient, second average gradient and third average gradient, respectively. The final update equations are

$$\overline{u}^{n+\tau} = \overline{u}^n - \frac{a\Delta t}{\Delta x}\int_0^\tau \left(u_j^{st}(1,\tau) - u_j^{st}(0,\tau)\right) d\tau, \tag{6.22}$$

$$\begin{aligned}\overline{\Delta u}^{n+\tau} &= \overline{\Delta u}^n - \frac{3a\Delta t}{\Delta x}\int_0^\tau \left(u_j^{st}(1,\tau) + u_j^{st}(0,\tau)\right) d\tau \\ &\quad + \frac{3a\Delta t}{\Delta x}\int_0^\tau 2\overline{u}^{n+\tau} d\tau,\end{aligned} \tag{6.23}$$

$$\begin{aligned}\overline{\Delta^2 u}^{n+\tau} &= \overline{\Delta^2 u}^n - \frac{5a\Delta t}{\Delta x}\int_0^\tau \left(u_j^{st}(1,\tau) - u_j^{st}(0,\tau)\right) d\tau \\ &\quad + \frac{5a\Delta t}{\Delta x}\int_0^\tau 2\overline{\Delta u}^{n+\tau} d\tau,\end{aligned} \tag{6.24}$$

$$\begin{aligned}\overline{\Delta^3 u}^{n+\tau} &= \overline{\Delta^3 u}^n - \frac{7a\Delta t}{\Delta x}\int_0^\tau \left(u_j^{st}(1,\tau) + u_j^{st}(0,\tau)\right) d\tau \\ &\quad + \frac{7a\Delta t}{\Delta x}\int_0^\tau \left(2\overline{u}^{n+\tau} + 2\overline{\Delta^2 u}^{n+\tau}\right) d\tau.\end{aligned} \tag{6.25}$$

Our obvious goal is to find the solution at $\tau = 1$, but that is not so straightforward for the gradients. With our specific choice of orthogonal basis, the update equation for $\overline{u}$ is a stand-alone equation that can be solved first. Next, we consider the update equation for $\overline{\Delta u}$ where the volume integral on the RHS requires us to know $\overline{u}^{n+\tau}$ at the Radau points. In other words, we need to solve the update equation of $\overline{u}$ at various Radau points to update $\overline{\Delta u}$. Sequentially, we need to solve the update equation of $\overline{\Delta u}$ at various Radau points to update $\overline{\Delta^2 u}$. This hierachical structure for updating variables is the strategy employed in HH-DG to include wave interaction into the volume integral. The integration of the surface and volume integrals in time may seem expensive; in this regard, we introduce a numerical trick called implicit

Radau integration in Appendix A to reduce the computational cost.

The current HH-DG scheme with Legendre polynomial is explicit in time. If we are to use a non-orthogonal basis, the volume integral will include other solution coefficients, turning HH-DG into an implicit scheme. This would imply an extra nonlinear solver must be implemented; the study on the cost and accuracy of using non-orthogonal basis is left for the future.

Numerical results for linear advection

The following numerical example takes $a = 1$ on the domain $x \in [0, 1]$, with periodic boundary condition. The initial values form a sine wave,

$$U_0 = \sin(2\pi x). \tag{6.26}$$

The ending time is a positive integer, hence the exact final solution is the same as the initial values. For both HH-DG and STE-DG with p-th order solution polynomial, the code uses a $(p+1)$-point Radau-Gaussian integration in time, and a $(2p+1)$-point Gaussian integration in space. A comparison between STE-DG and HH-DG for $p = 1$ is provided in Table 6.1. HH-DG is clearly faster due to the larger time step; in this case the CFL number achieved by HH-DG is more than 3 times larger. In reality, HH-DG is able to achieve a CFL of unity, which results in the exact solution due to the exact-shift property; however, we refrained from doing that in order to get a measurement of the order of accuracy.

We then consider advecting waves over a long period of time, $t_{max} = 300$. Table 6.2 shows the order of accuracy is clearly $2p+1$ in the cell average, whereas the order of accuracy of the first average gradient is quite unpredictable. Again the CFL is set to 0.9375 because there is no numerical error with a CFL of unity.

Brief remark on stability analysis

The Fourier analysis for stability is omitted for the HH-DG advection operator for one good reason. It turns out the final update matrix for the scalar linear advection equation is simply given by the exact shift and exact projected shift operators presented in the previous section. As long as $\nu \leq 1$, stability is ensured.

a) STE-DG $p = 1$, CFL $= 0.3$

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	Time [s]
10	9.07×10^{-2}		2.65×10^{-2}		1.0
20	2.30×10^{-2}	1.9	3.03×10^{-3}	3.1	3.6
40	5.75×10^{-3}	2.0	3.07×10^{-4}	3.3	14.2
80	1.44×10^{-4}	2.0	2.02×10^{-5}	3.9	57.1
160	3.60×10^{-4}	2.0	2.03×10^{-6}	3.3	225.5
320	8.99×10^{-5}	2.0	1.39×10^{-6}	0.5	896.2

b) HH-DG $p = 1$, CFL $= 0.9375$

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	Time(s)
10	3.10×10^{-3}		1.88×10^{-2}		0.4
20	4.14×10^{-4}	3.0	4.49×10^{-3}	2.0	1.1
40	5.30×10^{-5}	3.0	1.27×10^{-3}	2.0	4.4
80	6.68×10^{-6}	3.0	3.18×10^{-4}	2.0	17.9
160	8.38×10^{-7}	3.0	7.95×10^{-5}	2.0	71.4
320	1.05×10^{-7}	3.0	1.99×10^{-5}	2.0	283.2

Table 6.1: A comparison of order of accuracy between STE-DG and HH-DG for $p = 1$, $t_{final} = 3$. HH-DG is roughly 3 times faster than STE-DG, and is one order higher in terms of accuracy of the cell average.

6.2.2 One-dimensional Euler equations

The Euler equations are our next testing ground for our HH-DG advection operator. We demonstrate that, regardless of the nonlinearity, a LL-RK procedure and a hierarchical update of the conservative variables still work flawlessly. The 1-D Euler equations are given by,

$$\begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix}_x = 0, \tag{6.27}$$

where ρ is density, u is the velocity, p is the pressure, E is the total energy, and H is the total enthalpy. Pressure and total enthalpy are expressed in terms of ρ, u, and E,

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u^2\right), \tag{6.28}$$

a) HH-DG $p = 1$

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	4.01×10^{-2}		1.80×10^{-2}	
20	5.28×10^{-3}	2.9	1.88×10^{-3}	3.3
40	6.71×10^{-4}	3.0	6.25×10^{-4}	1.6
80	8.40×10^{-5}	3.0	2.66×10^{-4}	1.2
160	1.05×10^{-5}	3.0	2.09×10^{-5}	3.7
320	1.31×10^{-6}	3.0	1.37×10^{-5}	0.6

b) HH-DG $p = 2$

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	$L_2\,\overline{\triangle^2 u}_{error}$	Rate
10	9.61×10^{-4}		2.96×10^{-4}		1.20×10^{-3}	
20	3.08×10^{-5}	5.0	5.15×10^{-6}	5.8	1.50×10^{-4}	3.0
40	9.73×10^{-7}	5.0	3.21×10^{-7}	4.0	1.87×10^{-5}	3.0
80	3.07×10^{-8}	5.0	3.45×10^{-8}	3.2	2.34×10^{-6}	3.0
160	9.71×10^{-10}	5.0	4.03×10^{-9}	3.1	2.93×10^{-7}	3.0
320	3.12×10^{-11}	5.0	4.92×10^{-10}	3.0	3.66×10^{-8}	3.0

b) HH-DG $p = 3$

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	5.58×10^{-6}		1.75×10^{-6}	
20	4.50×10^{-8}	7.0	5.86×10^{-8}	4.9
40	3.69×10^{-10}	6.9	3.64×10^{-9}	4.0
80	4.13×10^{-12}	6.5	2.27×10^{-10}	4.0
160	1.23×10^{-13}	5.1	1.42×10^{-11}	4.0

N_{ele}	$L_2\,\overline{\triangle^2 u}_{error}$	Rate	$L_2\,\overline{\triangle^3 u}_{error}$	Rate
10	2.75×10^{-6}		4.92×10^{-5}	
20	8.68×10^{-8}	4.9	3.08×10^{-6}	4.0
40	2.72×10^{-9}	5.0	1.92×10^{-7}	4.0
80	8.51×10^{-11}	5.0	1.20×10^{-8}	4.0
160	2.66×10^{-12}	5.0	7.51×10^{-10}	4.0

Table 6.2: Order of accuracy for $p = 1, 2, 3$, CFL $= 0.9375$, $t_{final} = 300$.

$$H = E + \frac{p}{\rho}. \tag{6.29}$$

The primitive variables are replaced with the conservative variables,

$$\boldsymbol{U}_t + \nabla \cdot \mathbf{F}(\boldsymbol{U}) = \begin{pmatrix} U_0 \\ U_1 \\ U_2 \end{pmatrix}_t + \begin{pmatrix} U_1 \\ p + \frac{U_1^2}{U_0} \\ U_1 H \end{pmatrix}_x = 0, \tag{6.30}$$

where p and H is now,

$$p = (\gamma - 1)\left(U_2 - \frac{U_1^2}{2U_0}\right), \tag{6.31}$$

$$H = \frac{U_1}{U_0}\left(U_2 + (\gamma - 1)\left(U_2 - \frac{U_1^2}{2U_0}\right)\right). \tag{6.32}$$

We transform the conservative form of the Euler equations into weak form as with Eqn 6.3,

$$\begin{aligned} \int v\boldsymbol{U}^{n+1} d\xi = & \int v\boldsymbol{U}^n d\xi + \frac{\Delta t}{\Delta x}\int v_\xi \mathbf{F}\left(\boldsymbol{U}_j^{n+\tau}\right) d\xi d\tau \\ & - \frac{\Delta t}{\Delta x}\left(\int \left(v(1)\,\mathbf{F}\left(\boldsymbol{U}_j^{st}(1,\tau)\right) - v(0)\,\mathbf{F}\left(\boldsymbol{U}_{j-1}^{st}(1,\tau)\right)\right) d\tau\right) \end{aligned} \tag{6.33}$$

We acquire $\boldsymbol{U}_{j-1}^{st}$ by using a suitable order of LL-RK, and then solve the Riemann problem at the various Radau points using an approximate Roe solver[33]. The real enigma arises in the the volume integral of the RHS where the volume integrals for momentum and total energy contain highly nonlinear terms that make it impossible to express $\boldsymbol{U}_j^{n+\tau}$ solely in terms of the updated variables. Our solution is to express $F\left(\boldsymbol{U}_j^{n+\tau}\right)$ in terms of both updated variables $\boldsymbol{U}_j^{n+\tau}$, and the space-time expanded solution, $\boldsymbol{U}_{j-1}^{st}$. The following examples for $p = 1$ and 2 shall kindly supplement your understanding.

1-D Euler HH-DG $p = 1$ discretization

The piecewise-linear orthogonal Legendre basis is given by

$$\begin{aligned} v_0 &= 1, \\ v_1 &= 2\xi - 1. \;\; \xi \in [0,1] \end{aligned}$$

The three conservative variables are expressed as

$$\begin{aligned} U_0 &= \overline{U_0} + \overline{\Delta U_0}\,(2\xi - 1)\,, \\ U_1 &= \overline{U_1} + \overline{\Delta U_1}\,(2\xi - 1)\,, \\ U_2 &= \overline{U_2} + \overline{\Delta U_2}\,(2\xi - 1)\,, \end{aligned}$$

summing up to a total of six degrees of freedom per element. We churn out the update equations by inserting these expressions into Eqn 6.33,

$$\overline{\boldsymbol{U}}^{n+\tau} = \overline{\boldsymbol{U}}^{n} - \frac{\Delta t}{\Delta x}\left(\int_0^{\tau}\left(\hat{\mathbf{F}}_{j+\frac{1}{2}} - \hat{\mathbf{F}}_{j-\frac{1}{2}}\right)d\tau\right), \tag{6.34}$$

$$\begin{aligned} \overline{\Delta \boldsymbol{U}}^{n+\tau} &= \overline{\Delta \boldsymbol{U}}^{n} - \frac{3\Delta t}{\Delta x}\left(\int_0^{\tau}\left(\hat{\mathbf{F}}_{j+\frac{1}{2}} + \hat{\mathbf{F}}_{j-\frac{1}{2}}\right)d\tau\right) \\ &\quad + \frac{3\Delta t}{\Delta x}\left(\iint_0^{\tau} v_1(\xi)\,\mathbf{F}\left(\boldsymbol{U}_{p1}^{HDG}\right)d\tau d\xi\right). \end{aligned} \tag{6.35}$$

Notice the same $\int_0^{\tau}\hat{\mathbf{F}}_{j+\frac{1}{2}}d\tau$ is used in both update equations. We first use a LL-RK procedure to acquire $\boldsymbol{U}_j^{st}(\xi,\tau_k)$ at all Radau points τ_k,

$$\boldsymbol{U}^{st}(\xi,\tau_k) = \begin{pmatrix} \overline{U_0}^{st,\tau_k} + \overline{\Delta U_0}^{st,\tau_k}(2\xi - 1) \\ \overline{U_1}^{st,\tau_k} + \overline{\Delta U_1}^{st,\tau_k}(2\xi - 1) \\ \overline{U_2}^{st,\tau_k} + \overline{\Delta U_2}^{st,\tau_k}(2\xi - 1) \end{pmatrix}. \tag{6.36}$$

The space-time expanded solutions are then fed into a numerical flux solver,

$$\hat{\mathbf{F}}_{j+\frac{1}{2}}^{\tau_k} = \hat{\mathbf{F}}_{j+\frac{1}{2}}^{\tau_k}\left(\boldsymbol{U}_j^{st}(1,\tau_k), \boldsymbol{U}_{j+1}^{st}(0,\tau_k)\right). \tag{6.37}$$

We solve for the flux at all Radau points to integrate $\int_0^{\tau}\hat{\mathbf{F}}_{j+\frac{1}{2}}d\tau$ numerically. Eqn 6.34 is easily solved to provide the new cell averages at the Radau points. As for updating

the first average gradient, we introduce a new place-holder character $\boldsymbol{U}_{p1}^{HDG}$, which is a mixture of updated information and old information,

$$\boldsymbol{U}_{p1}^{HDG}(\xi,\tau_k)=\begin{pmatrix}\overline{U_0}^{n+\tau_k}+\overline{\Delta U_0}^{st,\tau_k}(2\xi-1)\\ \overline{U_1}^{n+\tau_k}+\overline{\Delta U_1}^{st,\tau_k}(2\xi-1)\\ \overline{U_2}^{n+\tau_k}+\overline{\Delta U_2}^{st,\tau_k}(2\xi-1)\end{pmatrix}. \tag{6.38}$$

With $\boldsymbol{U}_{p1}^{HDG}$ defined at all Radau points, we complete the update for the first averaged gradient for $\tau=1$.

1-D Euler HH-DG $p=2$ discretization

The section briefly goes through the discretization of $p=2$; the local Legendre polynomials are given by

$$\begin{aligned}v_0 &= 1,\\ v_1 &= 2\xi-1,\\ v_2 &= 6\xi^2-6\xi+1,\quad \xi\in[0,1]\end{aligned}$$

The three conservative variables are

$$U_0 = \overline{U_0}+\overline{\Delta U_0}(2\xi-1)+\overline{\Delta^2 U_0}\left(6\xi^2-6\xi+1\right),$$

$$U_1 = \overline{U_1}+\overline{\Delta U_1}(2\xi-1)+\overline{\Delta^2 U_0}\left(6\xi^2-6\xi+1\right),$$

$$U_2 = \overline{U_2}+\overline{\Delta U_2}(2\xi-1)+\overline{\Delta^2 U_0}\left(6\xi^2-6\xi+1\right),$$

with a total of nine degrees of freedom per element. The update equations are given by

$$\overline{\boldsymbol{U}}^{n+\tau} = \overline{\boldsymbol{U}}^{n}-\frac{\Delta t}{\Delta x}\left(\int_0^{\tau}\left(\hat{\mathbf{F}}_{j+\frac{1}{2}}-\hat{\mathbf{F}}_{j-\frac{1}{2}}\right)d\tau\right), \tag{6.39}$$

$$\begin{aligned}\overline{\Delta\boldsymbol{U}}^{n+\tau} &= \overline{\Delta\boldsymbol{U}}^{n}-\frac{3\Delta t}{\Delta x}\left(\int_0^{\tau}\left(\hat{\mathbf{F}}_{j+\frac{1}{2}}+\hat{\mathbf{F}}_{j-\frac{1}{2}}\right)d\tau\right)\\ &\quad+\frac{3\Delta t}{\Delta x}\left(\iint_0^{\tau}v_1(\xi)\,\mathbf{F}\left(\boldsymbol{U}_{p1}^{HDG}\right)d\tau d\xi\right),\end{aligned} \tag{6.40}$$

$$\begin{aligned}\overline{\Delta^2 \boldsymbol{U}}^{n+\tau} &= \overline{\Delta^2 \boldsymbol{U}}^{n} - \frac{5\Delta t}{\Delta x}\left(\int_0^{\tau}\left(\hat{\mathbf{F}}_{j+\frac{1}{2}} - \hat{\mathbf{F}}_{j-\frac{1}{2}}\right)d\tau\right) \\ &\quad + \frac{5\Delta t}{\Delta x}\left(\iint_0^{\tau} v_2(\xi)\,\mathbf{F}\left(\boldsymbol{U}_{p2}^{HDG}\right)d\tau d\xi\right). \end{aligned} \tag{6.41}$$

We first solve Eqn 6.39 to obtain the new $\overline{\boldsymbol{U}}^{n+\tau}$ at all Radau points, and insert them into the place-holder variable,

$$\boldsymbol{U}_{p1}^{HDG}(\xi,\tau_k) = \begin{pmatrix} \overline{U_0}^{n+\tau_k} + \overline{\Delta U_0}^{st,\tau_k}(2\xi-1) + \overline{\Delta^2 U_0}^{st,\tau_k}(6\xi^2-6\xi+1) \\ \overline{U_1}^{n+\tau_k} + \overline{\Delta U_1}^{st,\tau_k}(2\xi-1) + \overline{\Delta^2 U_1}^{st,\tau_k}(6\xi^2-6\xi+1) \\ \overline{U_2}^{n+\tau_k} + \overline{\Delta U_2}^{st,\tau_k}(2\xi-1) + \overline{\Delta^2 U_2}^{st,\tau_k}(6\xi^2-6\xi+1) \end{pmatrix}. \tag{6.42}$$

Again, $\boldsymbol{U}_{p1}^{HDG}$ is a mixture of the newest solution and the space-time expanded solution. We then solve for $\overline{\Delta \boldsymbol{U}}^{n+\tau}$ at all Radau points, and insert them into the next place-holder variable,

$$\boldsymbol{U}_{p2}^{HDG}(\xi,\tau_k) = \begin{pmatrix} \overline{U_0}^{n+\tau_k} + \overline{\Delta U_0}^{n+\tau_k}(2\xi-1) + \overline{\Delta^2 U_0}^{st,\tau_k}(6\xi^2-6\xi+1) \\ \overline{U_1}^{n+\tau_k} + \overline{\Delta U_1}^{n+\tau_k}(2\xi-1) + \overline{\Delta^2 U_1}^{st,\tau_k}(6\xi^2-6\xi+1) \\ \overline{U_2}^{n+\tau_k} + \overline{\Delta U_2}^{n+\tau_k}(2\xi-1) + \overline{\Delta^2 U_2}^{st,\tau_k}(6\xi^2-6\xi+1) \end{pmatrix}. \tag{6.43}$$

This allows us to update $\overline{\Delta^2 \boldsymbol{U}}^{n+\tau}$ and completes the update for all the degrees of freedom.

Numerical experiment with entropy wave

We test our HH-DG $p = 1$ and 2 discretizations for the Euler equations on the advection of an entropy wave[25]. The exact solution for the compressible Euler equation is

$$\begin{aligned} \rho &= \rho_\infty + A\sin\left[\pi\left(x - U_\infty t\right)\right], \\ u &= U_\infty, \\ p &= P_\infty, \end{aligned}$$

where A is some constant. We solve this on domain $x \in [-3, 3]$ with periodic boundary condition. The errors of the density variable are presented in Table 6.3. For nonlinear equations, we took CFL = 1 and achieved 3rd-order and 5th-order accuracy in the cell average for $p = 1$ and $p = 2$, respectively. It is extremely pleasing to demonstrate both accuracy and stability at a CFL number of unity, while RK-DG is only stable

with CFL = 0.4 and 0.27 for $p = 1$ and 2, respectively (see Section 3.4.1).

N_{ele}	DOF	$L_2\,\bar{\rho}_{error}$	Rate	$L_2\,\overline{\triangle\rho}_{error}$	Rate
12	24	3.14×10^{-1}		2.57×10^{-1}	
24	48	1.54×10^{-1}	1.0	6.18×10^{-2}	2.0
48	96	2.60×10^{-2}	2.6	5.41×10^{-3}	3.5
96	192	3.42×10^{-3}	3.0	5.91×10^{-4}	3.2
192	384	4.31×10^{-4}	3.0	5.05×10^{-5}	3.6
384	768	5.39×10^{-5}	3.0	2.19×10^{-5}	1.2

N_{ele}	DOF	$L_2\,\bar{\rho}_{error}$	Rate	$L_2\,\overline{\triangle\rho}_{error}$	Rate	$L_2\,\overline{\triangle^2\rho}_{error}$	Rate
12	24	8.53×10^{-3}		8.49×10^{-3}		3.32×10^{-3}	
24	48	3.35×10^{-4}	4.7	4.79×10^{-5}	7.5	7.08×10^{-5}	5.5
48	96	1.14×10^{-5}	4.9	1.99×10^{-5}	1.3	8.56×10^{-6}	3.1
96	192	4.09×10^{-7}	4.8	2.88×10^{-6}	2.8	5.29×10^{-7}	4.0
192	384	1.60×10^{-8}	4.7	3.77×10^{-7}	3.0	1.03×10^{-7}	2.4
384	768	6.00×10^{-10}	4.7	3.58×10^{-8}	3.4	4.98×10^{-8}	1.1

Table 6.3: Entropy wave case: order of accuracy for HH-DG $p = 1$ and 2, CFL = 1.0, $t_{final} = 50$. A density sine-wave is advected over 50 periods, while velocity and total energy remain constant.

Numerical experiment with a single expansion wave

The previous experiment is really just a single advection equation for density; in this experiment we take on a harder problem where all the conserved variables are coupled. Consider a single expansion wave[25] centered in $x \in [0, 1]$ with the following solutions,

$$V(\xi) = (0.02)\tanh\left(10\left(x - 0.5 - Vt\right)\right), \tag{6.44}$$

$$\frac{u(x,t)}{a_\infty} = M_\infty + \frac{2}{\gamma+1}\frac{V(\xi)}{a_\infty}, \tag{6.45}$$

$$\frac{\rho(x,t)}{\rho_\infty} = \left[1 + \frac{\gamma-1}{2}\left(\frac{u(\xi)}{a_\infty} - M_\infty\right)\right]^{\frac{2}{\gamma-1}}, \tag{6.46}$$

$$\frac{p(x,t)}{p_\infty} = \left[1 + \frac{\gamma-1}{2}\left(\frac{u(\xi)}{a_\infty} - M_\infty\right)\right]^{\frac{2\gamma}{\gamma-1}}, \tag{6.47}$$

where $\xi = x - Vt$ and $V = u + a$. For our specific problem, the freestream conditions are as listed,

$$\rho_\infty = \gamma, \tag{6.48}$$
$$p_\infty = 1, \tag{6.49}$$
$$a_\infty = 1, \tag{6.50}$$
$$M_\infty = -1. \tag{6.51}$$

All characteristics are linear and entropy is constant in space and time as in the Riemann invariant $u - \frac{2a}{\gamma-1}$. The flow exits supersonically to the left and enters subsonically from the right. A simple outflow boundary condition is used on the left; we apply the same condition on the right with caution. We extend the computational domain to $x \in [0, 2]$ and ensure the forward wave of the expansion fan has not reached the right boundary. The numerical results are presented in Table 6.4. The scheme is clearly 3rd-order accurate for $p = 1$. Let us try something harder for our next test drive.

N_{ele}	DOF	$L_2\,\bar{\rho}_{error}$	Rate	$L_2\,\overline{\triangle\rho}_{error}$	Rate
12	24	2.99×10^{-5}		8.61×10^{-5}	
24	48	5.16×10^{-6}	2.5	1.94×10^{-5}	2.1
48	96	6.89×10^{-7}	2.9	4.65×10^{-6}	2.0
96	192	8.76×10^{-8}	2.9	1.15×10^{-6}	2.0

Table 6.4: A single expansion fan case: order of accuracy for HH-DG ($p = 1$), CFL = 1.0, $t_{final} = 10$.

Numerical experiment for double expansion waves

The following description of the double expansion test case is a shortened version from Van Leer's CFD II course. This problem is beautifully designed where the flow exits supersonically at the domain boundaries and it can be easily extended to two and three dimensions. Ghost cells are used at the boundary of this 1-D problem. The two expansion waves will eventually meet each other at time $t = t_m$ and the interaction between them becomes nonlinear. A quick way to measure the error of the numerical scheme is to look at the change of entropy. Entropy is initialized to be constant in space and should remain constant through time since the entropy is constant along characteristics of an expansion fan. This is true even in interacting expansion waves.

Consider the following problem initialization on the the domain $x \in [-3, 3]$,

$$u = \begin{cases} u_{\infty,L} & x < -1.5 \\ a_\infty \left(M_{\infty,L} - \frac{u_{\infty,L}}{2a_\infty} \tanh\left(\frac{x+1}{\frac{1}{4}-(x+1)^2} + 1 \right) \right) & -1.5 \le x \le -0.5 \\ 0 & -0.5 < x < 0.5 \\ a_\infty \left(M_{\infty,R} + \frac{u_{\infty,R}}{2a_\infty} \tanh\left(\frac{x-1}{\frac{1}{4}-(x+1)^2} - 1 \right) \right) & 0.5 \le x \le 1.5 \\ u_{\infty,R} & x > 1.5 \end{cases}, \tag{6.52}$$

with the following conditions in the freestreams on the left and right side,

$$u_{\infty,L} = 2/\sqrt{\gamma}, \tag{6.53}$$

$$u_{\infty,R} = -2/\sqrt{\gamma}, \tag{6.54}$$

$$M_{\infty,L} = -2, \tag{6.55}$$

$$M_{\infty,R} = 2, \tag{6.56}$$

$$a_\infty = 1/\sqrt{\gamma}, \tag{6.57}$$

with $\gamma = 1.4$. The equations for density and pressure come from the previous test case. The change in entropy is also defined to be the error of the numerical scheme,

$$S = \Delta s = \log\left(\left(\frac{p_{t_{final}}}{\rho^\gamma_{t_{final}}} \right) / \left(\frac{p_{t_0}}{\rho^\gamma_{t_0}} \right) \right), \tag{6.58}$$

where $t_0 = 0$ and $t_{final} = 2$. The results for $p = 1$ and 2 are shown in Table 6.5. It appears for this highly nonlinear problem that the results on the coarsest grids are not fine enough to capture the complex interaction of the two expansion waves. The order of accuracy is better revealed on the finer grids where the $p = 1$ scheme is 3rd-order accurate, and roughly 4th-order accurate for $p = 2$. The latter accuracy is disappointing as we expected 5th-order acuracy $(2p + 1)$. Moreover, we have notice that the L_∞ error norm (not shown) is one order lower than L_2; this is indeed puzzling and will be left for future investigation.

6.2.3 Summary of HH-DG for advection

HH-DG is a space-time DG scheme that mimics the exact shift operator. The scheme is blessed with the good spatial accuracy of a typical DG scheme; it is also the first scheme to achieve CFL = 1 for all p. The secret comes in the careful evaluation of both surface and volume integrals. We demonstrated the volume integral must

N_{ele}	DOF	$L_2\,\overline{S}_{error}$	Rate	$L_2\,\overline{\Delta S}_{error}$	Rate
12	24	2.41×10^{-2}		8.53×10^{-2}	
24	48	4.25×10^{-3}	2.5	1.44×10^{-2}	2.5
48	96	3.02×10^{-4}	3.8	1.52×10^{-3}	3.2
96	192	3.85×10^{-5}	2.9	1.99×10^{-4}	2.9
192	384	5.00×10^{-6}	2.9	2.92×10^{-5}	2.7
384	768	6.96×10^{-7}	2.8	4.91×10^{-6}	2.5

N_{ele}	DOF	$L_2\,\overline{S}_{error}$	Rate	$L_2\,\overline{\Delta S}_{error}$	Rate	$L_2\,\overline{\Delta^2 S}_{error}$	Rate
12	24	2.46×10^{-2}		3.46×10^{-2}		5.96×10^{-2}	
24	48	1.61×10^{-3}	3.9	1.85×10^{-3}	4.2	5.05×10^{-3}	3.5
48	96	4.01×10^{-4}	2.0	3.14×10^{-4}	2.5	3.21×10^{-4}	3.9
96	192	3.36×10^{-5}	3.5	2.68×10^{-5}	3.5	2.07×10^{-5}	3.9
192	384	1.61×10^{-6}	4.3	1.55×10^{-6}	4.1	1.29×10^{-6}	4.0
384	768	8.36×10^{-8}	4.2	1.15×10^{-7}	3.7	8.33×10^{-8}	3.9

Table 6.5: Double expansion fans case: order of accuracy for HH-DG $p = 1$ and 2, CFL $= 1.0$, $t_{final} = 2$. Two expansion waves expand and interact at the center of the domain. The error is given by the change of entropy which is supposed to be zero.

include wave interaction in order to achieve good accuracy and stability. Numerical results for linear advection confirm the excellent stability and accuracy of HH-DG and the use of a local linear RK procedure to replace the CK procedure. As for the results for Euler equations, HH-DG works very well for $p = 1$ (which is expected since it is identical to Huynh's moment scheme); however, much work is ahead to ensure the $p = 2$ version works as well as its linear counterpart.

6.3 Space-time discontinuous Galerkin for diffusion

The HH-DG advection operator demonstrates remarkable robustness and high-order accuracy. We wish to repeat that success in the HH-DG diffusion operator; precisely, we want to combine HH-DG with RDG to form the Hancock-Huynh recovery-based DG (HH-RDG) scheme. Unlike the previous section where we have an "exact" operator to replicate, our design objective is to obtain a diffusion scheme with the highest possible Von Neumann number (VNN). We establish a few high-order diffusion schemes based on the classical finite-difference framework, and a version of finite difference with subgrid information. These schemes will provide a basis for comparison and insight into designing space-time diffusion schemes.

Consider the scalar linear diffusion equation with a source term,

$$u_t = Du_{xx} + S(x), \tag{6.59}$$

where D is the constant diffusion coefficient. The governing equation is tested with $v(x)$ in both space and time,

$$\iint v u_t \, dxdt = \iint v D u_{xx} \, dxdt + \iint v S(x) \, dxdt, \tag{6.60}$$

We express everything in terms of local variables,

$$\Delta x \iint v u_\tau \, d\xi d\tau = \frac{D\Delta t \iint v u_{\xi\xi} \, d\xi d\tau}{\Delta x} + \iint v S(x) \, dxdt, \tag{6.61}$$

$$= \frac{D\Delta t \int \left([v u_\xi - v_\xi u] \, |_{\xi=0}^{\xi=1} + \int v_{\xi\xi} u \, d\xi \right) d\tau}{\Delta x} + \iint v S(x) \, dxdt. \tag{6.62}$$

Note the RHS is currently expressed in spatial variables only. We implement the following two major substitutions to incorporate HH-DG concepts into the diffusion operator. First we introduce the time variable by replacing $u(\xi)$ in the surface integral with the space-time expanded solution $u^{st}(\xi, \tau)$. Secondly, we replace $u(\xi)$ in the volume integral with the updated solution, $u^{n+\tau}$:

$$\Delta x \iint v u_\tau \, d\xi d\tau = \frac{D\Delta t}{\Delta x} \int \left(\left[v u_\xi^{st}(\xi, \tau) - v_\xi u^{st}(\xi, \tau) \right] \, |_{\xi=0}^{\xi=1} \right) d\tau + \frac{D\Delta t}{\Delta x} \iint v_{\xi\xi} u^{n+\tau} \, d\xi \, d\tau + \frac{\Delta t}{\Delta x} \iint v S \, d\xi \, d\tau. \tag{6.63}$$

The discretization of this space-time diffusion equation for DG is a completely unexplored territory; we must look at other numerical frameworks for guidance. Our readers can jump directly to Section 6.3.6 for the HH-RDG discretization without loss in continuity; however, if our readers wish to follow our journey, the following sections provide space-time discretizations based on finite-difference frameworks.

6.3.1 Deriving space-time diffusion schemes

We derive different space-time diffusion schemes based on the classic finite-difference framework, finite difference with subgrid information, and DG. The part on DG details our naive attempts to generate a diffusion scheme based on the advection scheme from

the previous section; the result is disappointing and once again shows what is good for advection is not good for diffusion. Near the end of this section, we summarize our findings and compare all the diffusion schemes.

As with all great ideas, we begin with the simplest 1-D linear diffusion equation (no source term),

$$u_t = Du_{xx}. \tag{6.64}$$

The classical finite-difference scheme stores the function value at each point. The finite-difference scheme with subgrid information stores the solution and its derivatives at each point. Finally, our DG with modal basis stores moments within each cell. We compare these schemes based on their spatial order of accuracy and range of Von Neumann number (VNN). Notice a space-time scheme contains errors measured in both space and time, which are proportional to Δx and Δt, respectively. Depending on the nature of the PDE, we can always convert an error in Δt into Δx: i.e., $\Delta t \propto \Delta x$ in an advection-dominated problem, and $\Delta t \propto \Delta x^2$ in a diffusion-dominated problem. From here on, the term "order of accuracy" of a space-time scheme implies an error in terms of Δx.

Classical finite-difference scheme based on Taylor expansion in time

The finite-difference (FD) method is good for a preliminary study because of the ease in formulation. We will show later that FD schemes make excellent space-time diffusion schemes, however, at the cost of a large stencil. We proceed to derive 4th- and 6th-order schemes.

Fourth-order FD diffusion scheme

To acquire the approximate solution at $t = t_o + \Delta t$, we work with the Taylor expansion of the solution in time,

$$u\left(t_0 + \Delta t\right) = u\left(t_o\right) + u_t\,\Delta t + u_{tt}\,\frac{\Delta t^2}{2} + u_{ttt}\,\frac{\Delta t^3}{6} + u_{tttt}\,\frac{\Delta t^4}{24} + \ldots \tag{6.65}$$

We truncate the infinite series and keep the 0th, 1st, and 2nd-order terms. The truncated series is discretized in a FD framework by assuming equally spaced points and centering on a point located at x_j,

$$u_j\left(t_0+\Delta t\right)=u_j\left(t_o\right)+u_{t,j}\,\Delta t+u_{tt,j}\,\frac{\Delta t^2}{2}. \tag{6.66}$$

The two temporal derivatives are transformed into spatial derivatives via the Cauchy-Kovalevskaya (CK) procedure and then approximated with central finite-difference operators; our first pass at this gives

$$u_{t,j}=Du_{xx,j}\approx D\frac{\Delta^2 u_j}{\Delta x^2}=D\frac{u_{j+1}-2u_j+u_{j-1}}{\Delta x^2}, \tag{6.67}$$

$$u_{tt,j}=D^2u_{xxxx,j}\approx D^2\frac{\Delta^4 u_j}{\Delta x^4}=D^2\frac{\Delta^2\left(\Delta^2 u_j\right)}{\Delta x^4}=D^2\frac{u_{j+2}-4u_{j+1}+6u_j-4u_{j-1}+u_{j-2}}{\Delta x^4}. \tag{6.68}$$

Notice the approximately equal signs are used since the finite-difference operators used here contain numerical errors. The "crudeness" of the approximation is quantified by Taylor expanding the solution in space,

$$u_{j\pm l}=u\left(x_j\pm l\Delta x\right)=u_j\pm u_{x,j}\,l\Delta x+u_{xx,j}\,\frac{\left(l\Delta x\right)^2}{2}\pm u_{xxx,j}\,\frac{\left(l\Delta x\right)^3}{6}+u_{xxxx,j}\,\frac{\left(l\Delta x\right)^4}{24}+\ldots, \tag{6.69}$$

where l is an integer. We first look at the error in the second derivative. Inserting the Taylor expansion into Eqn 6.67 yields

$$\frac{\Delta^2 u_j}{\Delta x^2}=u_{xx,j}+u_{xxxx,j}\frac{\Delta x^2}{12}+u_{xxxxxx,j}\frac{\Delta x^4}{360}+O\left(\Delta x^6\right). \tag{6.70}$$

This 2nd-order difference operator approximates the real second derivative u_{xx} with an error of $O\left(\Delta x^2\right)$. Since this term gets multiplied by Δt in Eqn 6.66, the final error becomes $O\left(\Delta x^4\right)$ for a diffusion-dominated problem. We seek to improve upon the approximation of u_{xx} by adding a correction term to Eqn 6.70 to remove the 2nd-order error,

$$u_{xx,j}\approx\frac{\Delta^2 u_j}{\Delta x^2}-u_{xxxx,j}\frac{\Delta x^2}{12}, \tag{6.71}$$

hence our slightly refined approximation of u_{xx} contains an error of $O\left(\Delta x^4\right)$. We look at the error in the fourth derivative by inserting the Taylor expansion into Eqn 6.68,

$$\frac{\Delta^4 u_j}{\Delta x^4}=u_{xxxx,j}+u_{(6),j}\frac{\Delta x^2}{6}+u_{(8),j}\frac{\Delta x^4}{80}+O\left(\Delta x^6\right), \tag{6.72}$$

The $O\left(\Delta x^2\right)$ error is acceptable since this term is multiplied with $\Delta t^2\propto\Delta x^4$, resulting in a final error of $O\left(\Delta x^6\right)$. Inserting Eqns 6.68 and 6.71 into Eqn 6.90 results in

a scheme that is 4th-order in space,

$$\begin{aligned} u_j\left(t_0+\Delta t\right) &= u_j\left(t_o\right)+D\Delta t\left(\frac{\Delta^2 u_j}{\Delta x^2}-\frac{1}{12}\frac{\Delta^4 u_j}{\Delta x^2}\right)+\frac{D^2\Delta t^2}{2\Delta x^4}\Delta^4 u_j, \\ &= u_j\left(t_o\right)+r\Delta^2 u_j+\left(\frac{r^2}{2}-\frac{r}{12}\right)\Delta^4 u_j, \end{aligned} \tag{6.73}$$

where r is the shorthand notation of the Von Neumann number,

$$r=\frac{D\Delta t}{\Delta x^2}. \tag{6.74}$$

We replace central-difference operators with Fourier mode

$$u_j=e^{i\beta j}, \tag{6.75}$$

and acquire an amplification factor of

$$g\left(r,\beta\right)=1+2r\left(\cos\beta-1\right)+4\left(\frac{r^2}{2}-\frac{r}{12}\right)\left(\cos\beta-1\right)^2. \tag{6.76}$$

Figure 6.6 shows the amplification factor for different values of r. The line with $r=\frac{2}{3}$ represents the highest possible VNN without inducing instability.

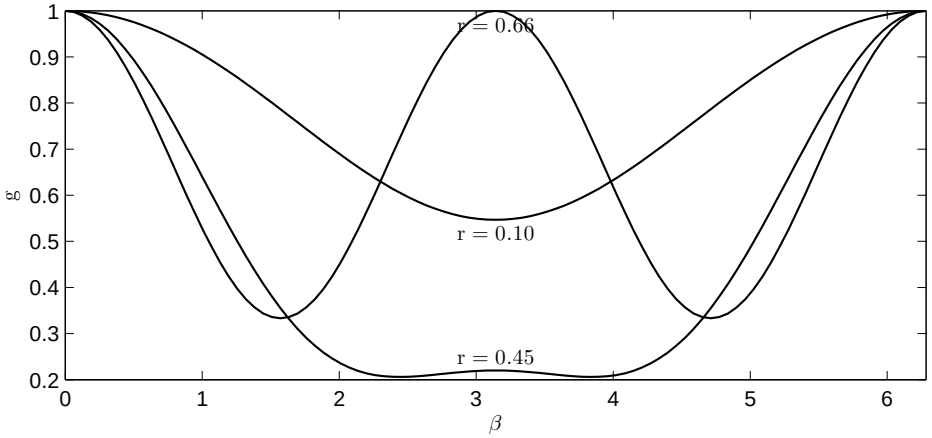

Figure 6.6: Amplification factor of the 4th-order diffusion scheme using central differencing. A maximum of $r=0.66$ is achieved with regard to stability.

6th-order FD diffusion scheme

We quickly generate a 6th-order FD diffusion scheme without detailing too much of the mathematics. The 6th-order central-difference operator is

$$\frac{\Delta^6 u_j}{\Delta x^6} = \frac{u_{j+3} - 6u_{j+2} + 15u_{j+1} - 20u_j + 15u_{j-1} - 6u_{j-2} + u_{j-3}}{\Delta x^6}. \tag{6.77}$$

Inserting the Taylor expansion (Eqn 6.69) results in

$$\frac{\Delta^6 u_j}{\Delta x^6} = u_{(6),j} + u_{(8),j}\frac{\Delta x^2}{8} + O\left(\Delta x^4\right); \tag{6.78}$$

the 2nd-order error is fine since it is multiplied by $\Delta t^3 \propto \Delta x^6$, resulting in an error of $O\left(\Delta x^8\right)$. We use 6th-order central-difference operator to improve upon the 4th-order operator in Eqn 6.68,

$$u_{xxxx,j} \approx \frac{\Delta^4 u_j}{\Delta x^4} - u_{(6),j}\frac{\Delta x^2}{6}. \tag{6.79}$$

We then use the 6th derivative together with the improved 4th derivative to correct the errors in the 2nd-order central difference operator,

$$u_{xx,j} \approx \frac{\Delta^2 u_j}{\Delta x^2} - u_{xxxx,j}\frac{\Delta x^2}{12} + u_{(6),j}\frac{\Delta x^4}{90}. \tag{6.80}$$

Inserting Eqns 6.78-6.80 into Eqn 6.65 results in the following 6th-order FD scheme,

$$\begin{aligned} u_j\left(t_0 + \Delta t\right) &= u_j\left(t_o\right) + D\Delta t\left(\frac{\Delta^2 u_j}{\Delta x^2} - \frac{1}{12}\frac{\Delta^4 u_j}{\Delta x^2} + \frac{1}{90}\frac{\Delta^6 u_j}{\Delta x^2}\right) \\ &+ \frac{D^2\Delta t^2}{2}\left(\frac{\Delta^4 u_j}{\Delta x^4} - \frac{1}{6}\frac{\Delta^6 u_j}{\Delta x^4}\right) + \frac{D^3\Delta t^3}{6}\left(\frac{\Delta^6 u_j}{\Delta x^6}\right), \end{aligned} \tag{6.81}$$

with the following amplification factor,

$$\begin{aligned} g\left(r,\beta\right) &= 1 + 2r\left(\cos\beta - 1\right) + 4\left(\frac{r^2}{2} - \frac{r}{12}\right)\left(\cos\beta - 1\right)^2 \\ &+ 8\left(\frac{r^3}{6} - \frac{r^2}{12} + \frac{r}{90}\right)\left(\cos\beta - 1\right)^3. \end{aligned} \tag{6.82}$$

We present a plot of the amplification factor in Figure 6.7. The scheme is stable for VNN up to 0.8413.

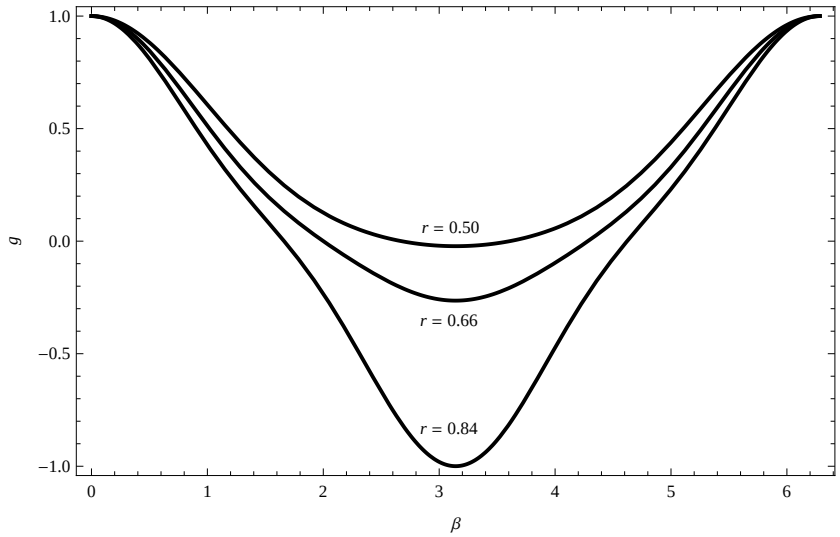

Figure 6.7: Amplification factor of the 6th-order diffusion scheme using central differencing. A maximum of $r = 0.84$ is achieved.

Summary of FD findings

The classical FD framework generates space-time diffusion schemes with very high VNN's. The 4th-order scheme with a 5-point stencil obtains a maximum VNN of 0.66, while the 6th-order scheme with a 7-point stencil obtains a maximum VNN of 0.84. Perhaps the most surprising finding in this exercise is that a growing stencil for diffusion increases the VNN; this is contrary to advection where a growing stencil centered around the update point (as in RK) decreases the CFL number. We believe a growing stencil mimics the infinite propagation speed of the characteristics in a diffusion equation; in contrast, a growing stencil for advection brings in additional information from downwind, which is destabilizing. However, a growing stencil greatly increases the complexity of the code for unstructured grids and makes boundary treatment a nightmare. We proceed to look at FD schemes with subgrid information to see what we can achieve with a compact stencil.

Finite-difference scheme with subgrid information based on Taylor expansion in time

We delve into two FD schemes with subgrid information. Considering equally spaced points, we consider a $p = 1$ scheme with the solution value and its first derivative stored at each point, and then a $p = 2$ scheme with the solution value and its first and second derivatives stored at each point.

FD with $p = 1$ subgrid information

We consider an FD scheme with $p = 1$ subgrid information; this allows us to store the solution value u and its first derivative u_x at each point. The manner of analysis is the same as the previous section applied to a system instead. Consider the 2nd-order Taylor expansion of the solution and its derivative in time,

$$\begin{aligned} u_j\left(t_0+\Delta t\right) &= u_j\left(t_o\right)+u_{t,j}\,\Delta t+u_{tt,j}\,\frac{\Delta t^2}{2}, \\ u_{x,j}\left(t_0+\Delta t\right) &= u_{x,j}\left(t_o\right)+u_{xt,j}\,\Delta t+u_{xtt,j}\,\frac{\Delta t^2}{2}. \end{aligned} \tag{6.83}$$

Our strategy is to fit a 5th-order polynomial through the three points and obtain its derivatives via a CK procedure,

$$\begin{aligned} u_t &= Du_{xx} \approx \frac{2D\left(u_{j+1}-2u_j+u_{j-1}\right)}{\Delta x^2}-\frac{D\left(u_{x,j+1}-u_{x,j-1}\right)}{2\Delta x}, \\ u_{xt} &= Du_{xxx} \approx \frac{15D\left(u_{j+1}-u_{j-1}\right)}{2\Delta x^3}-\frac{3D\left(u_{x,j+1}+8u_{x,j}+u_{x,j-1}\right)}{2\Delta x^2}, \\ u_{tt} &= D^2u_{xxxx} \approx -\frac{12D^2\left(u_{j+1}-2u_j+u_{j-1}\right)}{\Delta x^4}+\frac{6D^2\left(u_{x,j+1}-u_{x,j-1}\right)}{\Delta x^3}, \\ u_{xtt} &= D^2u_{xxxxx} \approx -\frac{90D^2\left(u_{j+1}-u_{j-1}\right)}{\Delta x^5} \\ &\quad -\frac{30D^2\left(u_{x,j+1}+4u_{x,j}+u_{x,j-1}\right)}{\Delta x^4}. \end{aligned} \tag{6.84}$$

Inserting Eqn 6.69 and the Taylor expansion of $u_{x,j}$ below,

$$u_{x,j\pm1}=u_x\left(x_j\pm\Delta x\right)=u_{x,j}\pm u_{xx,j}\,\Delta x+u_{xxx,j}\,\frac{\Delta x^2}{2}\pm u_{xxxx,j}\,\frac{\Delta x^3}{6}+u_{xxxxx,j}\,\frac{\Delta x^4}{24}+\ldots, \tag{6.85}$$

gives us the error associated with each approximation of the derivatives,

$$\begin{aligned} E\left(u_{xx,j}\right) &= -\frac{\Delta x^4}{360}u_{xxxxxx,j}+O\left(\Delta x^6\right), \\ E\left(u_{xxx,j}\right) &= -\frac{\Delta x^4}{240}u_{xxxxxxx,j}+O\left(\Delta x^6\right), \\ E\left(u_{xxxx,j}\right) &= +\frac{\Delta x^2}{15}u_{xxxxxx,j}+O\left(\Delta x^4\right), \\ E\left(u_{xxxxx,j}\right) &= +\frac{\Delta x^4}{12}u_{xxxxxxx,j}+O\left(\Delta x^4\right). \end{aligned} \tag{6.86}$$

This means the RHS of Eqn 6.83 contains spatial errors of $O\left(\Delta x^6\right)$; however, since Eqn 6.83 is only 2nd-order in time, the resulting scheme is 4th-order accurate for

time-dependent problems, and 6th-order accurate for steady-state problems. The system can be expressed in terms of Fourier mode β,

$$\begin{pmatrix} u^{n+1} \\ u_x^{n+1} \end{pmatrix}_j = \begin{pmatrix} u^n \\ u_x^n \end{pmatrix}_j + M(r,\beta) \begin{pmatrix} u^n \\ u_x^n \end{pmatrix}_j , \tag{6.87}$$

where M is a 2×2 matrix,

$$M = \begin{bmatrix} -4r + 12r^2 + 4r\,cos(\beta) - 12r^2\,cos(\beta) & -i\,r\,\Delta x\,sin(\beta) + 6\,i\,r^2\,\Delta x\,sin(\beta) \\ \frac{15\,i\,r\,sin(\beta)}{\Delta x} - \frac{90\,i\,r^2\,sin(\beta)}{\Delta x} & -12r + 60r^2 - 3r\,cos(\beta) + 30r^2\,cos(\beta) \end{bmatrix}. \tag{6.88}$$

Let $\lambda_{1,2}$ be the two eigenvalues of M, then the amplification factors are

$$g_{1,2} = 1 + \lambda_{1,2}(r,\beta). \tag{6.89}$$

Figure 6.8 shows the two amplification factors for various values of r. Notice this 4th-order scheme using a compact stencil with subgrid information has a maximum r of $\frac{1}{6}$.

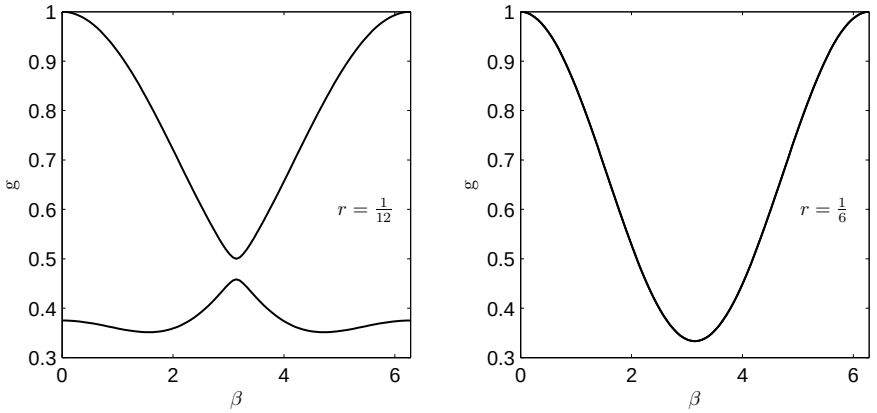

Figure 6.8: The two amplification factors associated with a finite-difference scheme with $p = 1$ subgrid information. The resulting scheme is 4th-order accurate in time. Notice for $r = \frac{1}{12}$, the two amplification factors are distinct, while for $r = \frac{1}{6}$, the two amplification factors coincide with each other.

FD with $p = 2$ subgrid information

We derive a 6th-order FD scheme for diffusion with piecewise-quadratic ($p = 2$) subgrid information; the variables u, u_x, and u_{xx} stored at each point. Consider the

3rd-order Taylor expansion of the three variables in time,

$$\begin{aligned} u_j\left(t_0+\Delta t\right) &= u_j\left(t_o\right)+u_{t,j}\,\Delta t+u_{tt,j}\,\frac{\Delta t^2}{2}+u_{ttt,j}\,\frac{\Delta t^3}{6}, \\ u_{x,j}\left(t_0+\Delta t\right) &= u_{x,j}\left(t_o\right)+u_{xt,j}\,\Delta t+u_{xtt,j}\,\frac{\Delta t^2}{2}+u_{xttt,j}\,\frac{\Delta t^3}{6}, \\ u_{xx,j}\left(t_0+\Delta t\right) &= u_{xx,j}\left(t_o\right)+u_{xxt,j}\,\Delta t+u_{xxtt,j}\,\frac{\Delta t^2}{2}+u_{xxttt,j}\,\frac{\Delta t^3}{6}. \end{aligned} \tag{6.90}$$

This time we fit a 8th-order polynomial through the three points and obtain its derivatives via a CK procedure,

$$\begin{aligned} u_{(3),j} \approx & -\frac{3}{8\Delta x^3}\left(35\left(u_{j-1}-u_{j+1}\right)+11\Delta x\left(u_{x,j+1}+\frac{48}{11}u_{x,j}+u_{x,j-1}\right)\right) \\ & -\frac{3}{8\Delta x^3}\left(\Delta x^2\left(u_{xx,j-1}-u_{xx,j+1}\right)\right), \end{aligned} \tag{6.91}$$

$$\begin{aligned} u_{(4),j} \approx & -\frac{3}{2\Delta x^4}\left(48\left(-u_{j-1}+2u_j-u_{j+1}\right)+13\Delta x\left(u_{x,j+1}-u_{x,j-1}\right)\right) \\ & -\frac{3}{2\Delta x^4}\left(\Delta x^2\left(-u_{xx,j-1}+24u_{xx,j}-u_{xx,j+1}\right)\right), \end{aligned} \tag{6.92}$$

$$\begin{aligned} u_{(5),j} \approx & -\frac{15}{\Delta x^5}\left(-21\left(u_{j-1}+u_{j+1}\right)+9\Delta x\left(-u_{x,j+1}-\frac{24}{9}u_{x,j}-u_{x,j-1}\right)\right) \\ & -\frac{15}{\Delta x^5}\left(\Delta x^2\left(u_{xx,j-1}-u_{xx,j+1}\right)\right), \end{aligned} \tag{6.93}$$

$$\begin{aligned} u_{(6),j} \approx & -\frac{90}{\Delta x^6}\left(32\left(u_{j-1}-2u_j+u_{j+1}\right)+11\Delta x\left(-u_{x,j+1}+u_{x,j-1}\right)\right) \\ & -\frac{90}{\Delta x^6}\left(\Delta x^2\left(u_{xx,j-1}-12u_{xx,j}+u_{xx,j+1}\right)\right), \end{aligned} \tag{6.94}$$

$$\begin{aligned} u_{(7),j} \approx & -\frac{315}{\Delta x^7}\left(15\left(u_{j-1}-u_{j+1}\right)+\Delta x\left(7u_{x,j+1}+16u_{x,j}+7u_{x,j-1}\right)\right) \\ & -\frac{315}{\Delta x^7}\left(\Delta x^2\left(-u_{xx,j-1}+u_{xx,j+1}\right)\right), \end{aligned} \tag{6.95}$$

$$\begin{aligned} u_{(8),j} \approx & -\frac{2520}{\Delta x^8}\left(24\left(-u_{j-1}+2u_j-u_{j+1}\right)+\Delta x\left(9u_{x,j+1}-9u_{x,j-1}\right)\right) \\ & -\frac{2520}{\Delta x^8}\left(\Delta x^2\left(-u_{xx,j-1}+8u_{xx,j}-u_{xx,j+1}\right)\right), \end{aligned} \tag{6.96}$$

Inserting the Taylor expansion of u_j into Eqn 6.69, $u_{x,j}$ into Eqn 6.85, together with the Taylor expansion of $u_{xx,j}$ below,

$$u_{xx,j\pm1} = u_{xx}\left(x_j \pm \Delta x\right) = u_{xx,j} \pm u_{xxx,j}\,\Delta x + u_{xxxx,j}\,\frac{\Delta x^2}{2} \pm u_{xxxxx,j}\,\frac{\Delta x^3}{6} + u_{xxxxxx,j}\,\frac{\Delta x^4}{24} + \ldots, \tag{6.97}$$

gives us the error associated with each approximation of the derivative,

$$\begin{aligned}
E\left(u_{xxx,j}\right) &= -\frac{\Delta x^6}{17920} u_{x(5),j} + O\left(\Delta x^8\right), \\
E\left(u_{xxxx,j}\right) &= +\frac{\Delta x^6}{13440} u_{x(6),j} + O\left(\Delta x^8\right), \\
E\left(u_{x(5),j}\right) &= +\frac{\Delta x^4}{1344} u_{x(7),j} + O\left(\Delta x^6\right), \\
E\left(u_{x(6),j}\right) &= -\frac{\Delta x^4}{224} u_{x(8),j} + O\left(\Delta x^6\right), \\
E\left(u_{x(7),j}\right) &= +\frac{\Delta x^2}{64} u_{x(9),j} + O\left(\Delta x^4\right), \\
E\left(u_{x(8),j}\right) &= +\frac{\Delta x^2}{8} u_{x(10),j} + O\left(\Delta x^4\right).
\end{aligned} \tag{6.98}$$

The resulting method is 8th-order accurate in space, and 6th-order accurate in time. We rewrite the update equations in matrix form,

$$\begin{pmatrix} u^{n+1} \\ u_x^{n+1} \\ u_{xx}^{n+1} \end{pmatrix}_j = \begin{pmatrix} u^n \\ u_x^n \\ u_{xx}^n \end{pmatrix}_j + M\left(r, \beta\right) \begin{pmatrix} u^n \\ u_x^n \\ u_{xx}^n \end{pmatrix}_j, \tag{6.99}$$

where M is a 3×3 matrices. The matrix and its eigenvalues are omitted here; only the amplification factors are shown in Figure 6.9. This scheme is stable up to $r = 0.124$; however, the eigenvalues only remain real up to $r = 0.0529$.

Summary of FD with subgrid information findings

FD schemes with subgrid information produce high-resolution spatial discretizations; e.g., $p = 1$ is 6th-order accurate in space and $p = 2$ is 8th-order accurate in space. However, these schemes are only 4th- and 6th-order accurate in time. In addition, their VNN is much lower than that of the classical FD schemes. We clearly see that using a compact stencil has an adverse effect on the VNN; the above schemes represent the best one can do with a compact stencil using subgrid information.

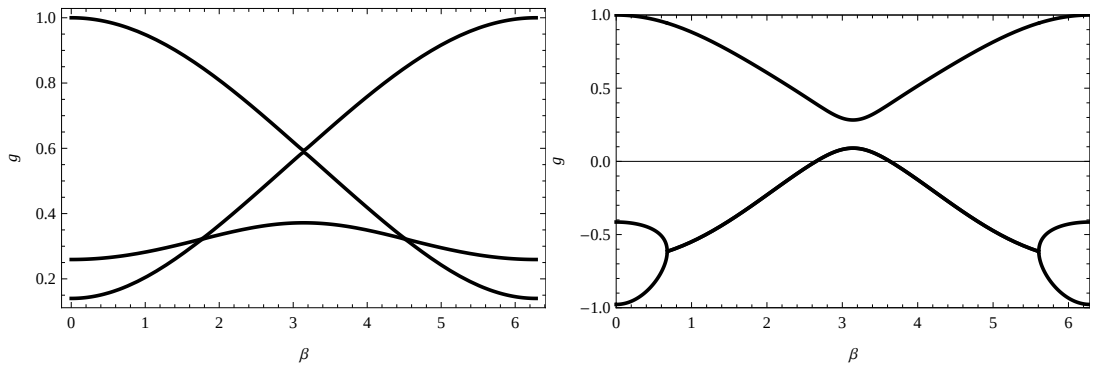

Figure 6.9: The three amplification factors associated with the FD scheme with $p = 2$ subgrid information is shown for $r = 0.0529$ (left) and $r = 0.124$ (right). The eigenvalues are real up to $r = 0.0529$. The eigenvalues become complex for $r \geq 0.0529$ and the amplification factors remain under unity up till $r = 0.124$.

DG schemes based on Hancock's observation: The rise of HH-RDG

The concepts in the FD scheme with subgrid information can be easily ported into the DG framework via a change of basis from nodal (strong interpolation) to modal (weak interpolation). However, this time around we shall derive a DG diffusion scheme using Hancock's observation instead of Taylor expansions in time.

We wish to replicate the HH-DG advection operator for diffusion; unfortunately, what works for the gander does not work for the goose. If we naively replicate our procedure for advection, the resulting diffusion scheme will be poor in accuracy and low in VNN; we will show this in three of our naive attempts below. In addition, the diffusion equation exhibits a different scaling of time and space, which ultimately affects the conversion of temporal to spatial derivatives.

Let us go through a little thought experiment and imagine the Cauchy-Kovalevaskaya (CK) procedure as a black box that puts in spatial derivatives and puts out temporal derivatives. For advection of the form $u_t = u_x$, the conversion rate is one-to-one. In mathematical terminology, it means each spatial derivative of order k results in a temporal derivative of the same order. Consider the piecewise-linear solution in space with u and u_x as variables. The CK procedure relates u_t to u_x, hence the set of $\{u, u_x, u_t\}$ forms a complete $p = 1$ space-time basis.

When dealing with diffusion, the CK procedure does not work in our favor. With diffusion of the form, $u_t = u_{xx}$, the conversion rate becomes two-to-one. This means it takes two spatial derivatives to get one temporal derivative. With the same piecewise-

linear solution in space with u and u_x as variables, the CK procedure fails to yield u_t because u_{xx} is not available. If one naively use the LL-RK (numerical version of CK), the order of accuracy of the scheme is at most 2nd-order. This simple thought experiment demonstrates that the CK procedures for advection and diffusion require completely different treatments. The fix is simple; the CK procedure for the diffusion operator simply requires more information in the spatial direction than its advection counterpart. Next, we demonstrate numerically the three naive and one smart attempts to incorporate Hancock's observation into DG diffusion.

Naive scheme I: LL-RK applied on $p = 1$ solution, RDG ($p = 1$) at the Radau points

In the first of the series of naive attempts, we apply the LL-RK procedure on the piecewise-linear solution, and then apply spatial RDG to recover cubic polynomials at the two Radau time levels . We solve the steady-state linear diffusion equation with a source term,

$$u_t = u_{xx} + 4\pi^2 sin\left(2\pi x\right), \quad x \in [0\ 1]\,. \tag{6.100}$$

Though one might question the use of a steady-state problem for a space-time scheme, it turns out this simple problem is good enough to reveal stability limit and spatial accuracy of the scheme. If the scheme cannot even achieve a good order of accuracy in a steady-state problem, then there is no further need to test the scheme on a time-accurate problem. The procedure is as follows: we sweep through all cells with LL-RK to obtain u and u_x at all Radau points. We then sweep through all interfaces with RDG to obtain the recovered function at all Radau points. The steady-state numerical results are provided in Table 6.6. This method fails to achieve 3rd-order accuracy and obtains a maximum VNN of $\frac{1}{8}$. Our first attempt falls short of the desired 4th-order of accuracy, and also fails to reach a modest goal of $r = \frac{1}{6}$. With a $p = 1$ solution, there is simply not enough spatial information per cell to obtain a 1st-order time derivative. This observation leads us directly to our next naive attempt.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	1.14×10^{-2}		3.82×10^{-3}	
20	40	3.62×10^{-3}	1.7	5.94×10^{-4}	2.7
40	80	1.00×10^{-3}	1.9	8.04×10^{-5}	2.9
80	160	2.61×10^{-4}	1.9	1.04×10^{-5}	2.9

Table 6.6: Naive scheme 1 with a maximum VNN of $\frac{1}{8}$.

Naive scheme II: CRDG from $p = 1$ to $p = 5$, LL-RK applied on $p = 5$, RDG ($p = 5$) at the Radau points

Our second attempt uses a technique called CRDG (see Section 3.5.1). CRDG raises the polynomial order of the piecewise-linear ($p = 1$) solution to piecewise-quintic ($p = 5$) using information from all neighboring cells. We then apply the LL-RK procedure to the piecewise-quintic solution to each cell and performed RDG to recover 11th-order polynomials at the two Radau time levels at each interface. The end result is a DG scheme with a 5-cell stencil. The steady-state numerical results are given in Table 6.7. The finest grid requires a numerical environment that supports more than 16 decimal places, and hence intentionally left blank. Notice this scheme achieves 5th-order (approaching 6th-order) accuracy and obtains a maximum of $r = 0.03$. The order of accuracy is high but the time step is too small. Contrary to our finding in the FD section, a wider stencil in DG leads to a lower VNN. We believe the major culprit to be the imbalance between the amounts of spatial and temporal information used, leading to inefficiency in the scheme. Our next naive attempt will decrease the amount of spatial information.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	2.68×10^{-6}		1.14×10^{-6}	
20	40	6.66×10^{-8}	5.3	6.04×10^{-9}	7.6
40	80	1.21×10^{-9}	5.8	4.36×10^{-11}	7.1
80	160	-	-	-	-

Table 6.7: Naive scheme 2 with a maximum VNN of 0.03.

Naive scheme III: CRDG from $p = 1$ to $p = 3$, LL-RK applied on $p = 3$, RDG ($p = 3$) at the Radau points

In an effort to tone down our previous attempt, the CRDG procedure raises the piecewise-linear ($p = 1$) solution to piecewise-cubic ($p = 3$) only; instead of using all the moments from the neighboring cells, we only use the cell averages. The LL-RK

procedure is applied to the piecewise-cubic solution and then RDG recovers a 7th-order polynomials at the two Radau points. The numerical results are presented in Table 6.8. The scheme approaches the 4th order of accuracy and the maximum VNN is 0.04, representing a mere 30% improvement over the previous case. At this point, it becomes apparent that we need to do something drastically different to increase the stability limit.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	2.28×10^{-4}		6.78×10^{-5}	
20	40	2.74×10^{-5}	3.1	1.63×10^{-6}	5.4
40	80	2.14×10^{-6}	3.7	5.03×10^{-8}	5.0
80	160	1.47×10^{-7}	3.9	1.62×10^{-9}	5.0

Table 6.8: Naive scheme 3 with maximum VNN of 0.04.

Summary of the three naive attempts

We make two important observations from the three naive attempts. First, a CK (or LL-RK) procedure applied to a piecewise-linear solution is insufficient for obtaining a first-order temporal derivative. Second, naive schemes II and III attempt to remedy the first problem by converting the original $p = 1$ solution into a high-order solution, but the result is even worse than for the first naive scheme in terms of VNN. It is clear using the same approach as for the HH-DG advection operator is leading nowhere, and a drastically new strategy is needed. The next section humbly presents a scheme resolving these two issues.

Smart attempt IV (HH-RDG): LL-RK applied on RDG ($p = 1$)

We seek a completely different approach by applying the CK (or LL-RK) procedure on the recovered function. Instead of acquiring a local evolution of the solution, we acquire the local evolution of the recovered function! This scheme only applies the recovery procedure once at $\tau = 0$ instead of multiple times at different Radau points. Since only the function value and the first derivative are required at the interface, the CK procedure is applied on the recovered function f in the differential form. This is much cheaper compared to the integral form because Gaussian integration is avoided. The end result is a space-time recovered function, f^{st}, that is a function of time.

Starting with a piecewise-linear solution at $\tau = 0$ for all elements Ω_j, we recover an unique cubic recovery function $f_{j+\frac{1}{2}}$ between elements Ω_j and Ω_{j+1}. Note we are using the same Legendre polynomials from Chapter 3. The recovered function $f_{j+\frac{1}{2}}$

contains 4 unique coefficients $b_{k,j+\frac{1}{2}}$,

$$f_{j+\frac{1}{2}} = b_{0,j+\frac{1}{2}} + b_{1,j+\frac{1}{2}} r + b_{2,j+\frac{1}{2}} r^2 + b_{3,j+\frac{1}{2}} r^3, \quad r \in [-\frac{1}{2}\ \frac{1}{2}], \tag{6.101}$$

where r now represents the coordinate normal to the interface; as in Section 3.1. The interface lies on $r = 0$. The local coordinate r is confined to $[-\frac{1}{2}\ \frac{1}{2}]$, and is related to $x \in [-\Delta x\ \Delta x]$ through $r = \frac{x}{2\Delta x}$, hence $\frac{\partial r}{\partial x} = \frac{1}{2\Delta x}$. We enforce the recovered function $f_{j+\frac{1}{2}}$ to satisfy the original PDE, and 3 derivatives of the PDE,

$$\begin{aligned}
f_t &= D \quad f_{xx} + S(x), \\
f_{xt} &= D \quad f_{xxx} + \frac{\partial}{\partial x} S(x), \\
f_{xxt} &= D \quad f_{xxxx} + \frac{\partial^2}{\partial x^2} S(x), \\
f_{xxxt} &= D \quad f_{xxxxx} + \frac{\partial^3}{\partial x^3} S(x).
\end{aligned} \tag{6.102}$$

Notice the subscript $j+\frac{1}{2}$ is dropped. We obtain the residual update form by inserting Eqn 6.101 with $r = 0$ into Eqn 6.102:

$$\begin{pmatrix} f \\ f_x \\ f_{xx} \\ f_{xxxx} \end{pmatrix}_t = \begin{pmatrix} b_0 \\ b_1 \left(\frac{\partial r}{\partial x}\right) \\ 2b_2 \left(\frac{\partial r}{\partial x}\right)^2 \\ 6b_3 \left(\frac{\partial r}{\partial x}\right)^3 \end{pmatrix}_t = \begin{pmatrix} Df_{rr}/(2\Delta x)^2 + (2\pi)^2 sin(2\pi x) \\ Df_{rrr}/(2\Delta x)^3 + (2\pi)^3 cos(2\pi x) \\ Df_{rrrr}/(2\Delta x)^4 - (2\pi)^4 sin(2\pi x) \\ Df_{rrrrr}/(2\Delta x)^5 - (2\pi)^5 cos(2\pi x) \end{pmatrix},$$

$$= \begin{pmatrix} D(2b_2)/(2\Delta x)^2 + (2\pi)^2 sin(2\pi x) \\ D(6b_3)/(2\Delta x)^3 + (2\pi)^3 cos(2\pi x) \\ -(2\pi)^4 sin(2\pi x) \\ -(2\pi)^5 cos(2\pi x) \end{pmatrix}.$$

The RHS expressions are input as "Res" to the LL-RK update in Eqn 6.17. We advance this system with LL-RK to the Radau points; we have obtained $f^{st}(\tau)$ at

the Radau points only. We replace u^{st} in the surface integral of Eqn 6.63 with f^{st},

$$\begin{aligned}\Delta x \iint v u_\tau \, d\xi d\tau &= \frac{D\Delta t}{\Delta x} \int \left(\left[v f_\xi^{st}(\tau) - v_\xi f^{st}(\tau) \right] |_{\xi=0}^{\xi=1} \right) d\tau \\ &+ \frac{D\Delta t}{\Delta x} \iint v_{\xi\xi} u^{n+\tau} \, d\xi \, d\tau + \Delta x \Delta t \iint v S \, d\xi \, d\tau. \quad (6.103)\end{aligned}$$

Note the volume integral for $p = 1$ is zero. The numerical results for smart scheme IV are presented in Table 6.9. The scheme clearly demonstrates 4th-order of accuracy and obtains a maximum VNN of $\frac{1}{6}$, representing a healthy combination of accuracy and speed. The next logical step is to repeat the same feat for piecewise-quadratic DG ($p = 2$).

N_{ele}	DOF	$L_2\, \bar{u}_{error}$	Rate	$L_2\, \overline{\triangle u}_{error}$	Rate
10	20	1.67×10^{-4}		1.11×10^{-4}	
20	40	1.15×10^{-5}	3.9	3.43×10^{-6}	5.0
40	80	7.58×10^{-7}	3.9	1.04×10^{-7}	5.0
80	160	4.86×10^{-8}	4.0	3.04×10^{-9}	5.1
160	320	3.05×10^{-9}	4.0	7.49×10^{-11}	5.3
320	640	1.84×10^{-10}	4.1	1.76×10^{-12}	5.4

Table 6.9: Smart scheme IV with a maximum VNN of $\frac{1}{6}$. The smart scheme IV will soon be named the HH-RDG scheme.

Smart attempt IV (HH-RDG): LL-RK applied on RDG ($p = 2$) with correct volume integral

For $p \geq 2$, the volume integral enters into the update equation of the 2nd gradient and hence must be treated carefully. We follow the same technique as applied to the volume integral in the advection operator, which is to infiltrate the volume integral with information from neighboring elements. Starting with a piecewise-quadratic solution at $\tau = 0$ for all elements Ω_j, we recover an unique quintic recovered function $f_{j+\frac{1}{2}}$ between elements Ω_j and Ω_{j+1}. The recovered function $f_{j+\frac{1}{2}}$ contains 6 unique coefficients $b_{k,j+\frac{1}{2}}$,

$$f_{j+\frac{1}{2}} = b_{0,j+\frac{1}{2}} + b_{1,j+\frac{1}{2}} r + b_{2,j+\frac{1}{2}} r^2 + b_{3,j+\frac{1}{2}} r^3 + b_{4,j+\frac{1}{2}} r^4 + b_{5,j+\frac{1}{2}} r^5, \quad r \in [-\frac{1}{2}\ \frac{1}{2}], \quad (6.104)$$

where the interface lies on $r = 0$. We enforce $f_{j+\frac{1}{2}}$ to satisfy 6 derivatives (including the zeroth derivative) of the governing equation in differential form,

$$\frac{\partial^l}{\partial x^l} f_t \;\; = D \frac{\partial^l}{\partial x^l} \quad \left(f_{xx} + S\left(x\right)\right), \quad l = 0, 1, 2, 3, 4, 5. \tag{6.105}$$

Again the subscript $j + \frac{1}{2}$ is dropped. The equations above are rewritten in residual form for $r = 0$,

$$\begin{pmatrix} f \\ f_x \\ f_{xx} \\ f_{x(3)} \\ f_{x(4)} \\ f_{x(5)} \end{pmatrix}_t = \begin{pmatrix} 0!\, b_0 \\ 1!\, b_1 \frac{\partial r}{\partial x} \\ 2!\, b_2 \left(\frac{\partial r}{\partial x}\right)^2 \\ 3!\, b_3 \left(\frac{\partial r}{\partial x}\right)^3 \\ 4!\, b_4 \left(\frac{\partial r}{\partial x}\right)^4 \\ 5!\, b_5 \left(\frac{\partial r}{\partial x}\right)^5 \end{pmatrix}_t = \begin{pmatrix} D\left(2b_2\right)/\left(2\Delta x\right)^2 + \left(2\pi\right)^2 sin\left(2\pi x\right) \\ D\left(6b_3\right)/\left(2\Delta x\right)^3 + \left(2\pi\right)^3 cos\left(2\pi x\right) \\ D\left(24b_4\right)/\left(2\Delta x\right)^4 - \left(2\pi\right)^4 sin\left(2\pi x\right) \\ D\left(120b_5\right)/\left(2\Delta x\right)^5 - \left(2\pi\right)^5 cos\left(2\pi x\right) \\ \left(2\pi\right)^6 sin\left(2\pi x\right) \\ \left(2\pi\right)^7 cos\left(2\pi x\right) \end{pmatrix}.$$

We apply LL-RK procedure to this system to obtain $f^{st}\left(\tau\right)$ at all Radau points. We are almost done with the discretization except for the new volume integral appearing in the update equation of $\overline{\Delta^2 u}$,

$$\begin{aligned} \overline{\Delta^2 u}^{n+1} \;\; = \;\; & \overline{\Delta^2 u}^{n} - \frac{D\Delta t}{\Delta x^2} \int \left(\left[v_2 f_\xi^{st}\left(\tau\right) - v_{2,\xi} f^{st}\left(\tau\right) \right] |_{\xi=0}^{\xi=1} \right) d\tau \\ & + \frac{D\Delta t}{\Delta x^2} \iint 12 u^{n+\tau}\, d\xi\, d\tau + \Delta t \iint v S\, d\xi\, d\tau. \end{aligned} \tag{6.106}$$

We simplify the interior volume integral of u to

$$\iint 12 u\, d\xi\, d\tau = \int 12 \overline{u}\left(\tau\right) d\tau, \tag{6.107}$$

where $\overline{u}\left(\tau\right)$ is obtained from the update equation of $\overline{u}$. Hence this diffusion scheme follows the same hierarchical updating structure as in the advection case. The numerical results are presented in Table 6.10. This $p = 2$ scheme achieves the 7th order of accuracy and obtained a maximum VNN of $\frac{1}{10}$.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	$L_2\,\overline{\triangle^2 u}_{error}$	Rate
10	30	7.84×10^{-8}		5.67×10^{-8}		1.31×10^{-6}	
20	60	6.81×10^{-10}	6.8	4.80×10^{-10}	6.9	2.06×10^{-8}	6.0
40	120	5.49×10^{-12}	6.9	3.87×10^{-12}	6.9	3.21×10^{-10}	6.0
80	240	4.40×10^{-14}	6.9	3.07×10^{-14}	7.0	5.02×10^{-12}	6.0

Table 6.10: Smart scheme IV ($p = 2$) with a maximum VNN of $\frac{1}{10}$. The scheme demonstrates 7th-order accuracy.

Not-so-smart scheme V: LL-RK applied on RDG ($p = 2$) with incorrect volume integral based on STE-DG

We interrupt with a brief section on the incorrect treatment of the space-time volume integral for $p = 2$. Consider the STE-DG method where the space-time volume integral only uses information from the local element. STE-DG uses u^{st} in the volume integral,

$$\begin{aligned}\Delta x \iint v u_\tau \, d\xi d\tau &= \frac{D\Delta t}{\Delta x} \int \left(\left[v f_\xi^{st}(\tau) - v_\xi f^{st}(\tau) \right] |_{\xi=0}^{\xi=1} \right) d\tau \\ &\quad + \frac{D\Delta t}{\Delta x} \iint v_{\xi\xi} u^{st} \, d\xi \, d\tau + \Delta x \Delta t \iint v S \, d\xi \, d\tau. \quad (6.108)\end{aligned}$$

The numerical results are presented in Table 6.11. This hybrid scheme obtains a maximum VNN of $\frac{1}{30}$, and 4th-order of accuracy. These results are consistent with the observations made in the advection section. Clearly, we see the correct treatment of the volume integral is essential for high-order accuracy and high VNN.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	$L_2\,\overline{\triangle^2 u}_{error}$	Rate
10	30	1.97×10^{-4}		2.22×10^{-5}		7.65×10^{-4}	
20	60	1.23×10^{-5}	4.0	7.69×10^{-7}	4.9	4.47×10^{-5}	4.1
40	120	7.81×10^{-7}	4.0	2.51×10^{-8}	4.9	2.75×10^{-6}	4.0
80	240	4.92×10^{-8}	4.0	7.99×10^{-10}	5.0	1.71×10^{-7}	4.0

Table 6.11: Not-so-smart scheme V with a maximum VNN of $\frac{1}{30}$. This hybrid scheme ($p = 2$) is only 4th-order accurate.

Summary of the DG diffusion schemes

We have demonstrated one successful space-time DG diffusion scheme, the smart scheme IV, with high-order of accuracy and a high VNN. The scheme is different from HH-DG for advection in that the recovered function locally evolves instead of the solution. The only similarity is the hierachical update structure. This z has been

a long one, but the results are fruitful. We are now ready for a rigorous presentation of HH-RDG and tackle time-accurate problems.

6.3.2 Hancock-Huynh interface-centered recovery-based discontinuous Galerkin method (HH-RDG)

In this section we bring together the important discoveries made in the previous attempts to generate a fast diffusion operator based on Hancock's observation. We present the complete Hancock-Huynh interface-centered recovery-based Galerkin Method (HH-RDG) for diffusion that is of high order in both space and time. HH-RDG achieves a high VNN with the minimal number of flux evaluations. We shall detail the discretization of the update equation, local evolution of the recovered function, implementation of boundary conditions, and Fourier analysis of the method.

We solve the linear diffusion equation,

$$u_t = Du_{xx}, \quad x \in [0\ 2\pi]\,, \tag{6.109}$$

where D is the constant diffusion coefficient. The space-time HH-RDG equation is obtained by testing the diffusion equation with a test function v and then replacing all interface values of u with the space-time recovered function f^{st}, and replacing the interior solution in the volume integral with the updated solution $u^{n+\tau}$,

$$\int_{\Omega_j} v\left(u^{n+1}-u^n\right)d\xi = \frac{D\Delta t}{\Delta x^2}\int\left(\left[vf^{st}_{\xi}-v_{\xi}f^{st}\right]\,|^1_0+\int_{\Omega_j} v_{\xi\xi}u^{n+\tau}\,d\xi\right)d\tau. \tag{6.110}$$

At the interface between Ω_j and Ω_{j+1} for $\tau = 0$, there exists an unique recovered function $f_{j+\frac{1}{2}}$ (see Section 3.1). We now force $f_{j+\frac{1}{2}}$ to satisfy the inner product of the local governing equation with a test function ψ,

$$\int_{\Omega_j \bigcup \Omega_{j+1}} \psi \frac{\partial f_{j+\frac{1}{2}}}{\partial t}\,dx = D\int_{\Omega_j \bigcup \Omega_{j+1}} \psi \frac{\partial^2 f_{j+\frac{1}{2}}}{\partial x^2}\,dx. \tag{6.111}$$

Note that ψ is the basis function for the recovery function and this equation applies to all ψ spanning the solution space of f. This is slightly different from the procedure listed for the smart scheme IV; it turns out a weak formulation is needed for time-accurate problems. The space-time solution $f^{st}(\tau)$ is obtained from a local evolution of Eqn 6.111 via the LL-RK procedure. The following examples illustrate how to

obtain $f^{st}(\tau)$ for $p = 1$ and 2.

Time-accurate linear diffusion, $p = 1$, periodic boundary condition

For a piecewise-linear solution, RDG recovers an unique f at $\tau = 0$,

$$f = b_0\psi_0 + b_1\psi_1 + b_2\psi_2 + b_3\psi_3, \tag{6.112}$$

where the recovery basis functions are the monomials

$$\psi_i = r^i, \quad r \in \left[-\frac{1}{2}\ \frac{1}{2}\right]. \tag{6.113}$$

Inserting these test functions into Eqn 6.111 and rewriting in residual form yields

$$\begin{pmatrix} b_0 + \frac{b_2}{12} \\ \frac{b_1}{12} + \frac{b_3}{80} \\ \frac{b_0}{12} + \frac{b_2}{80} \\ \frac{b_1}{80} + \frac{b_3}{448} \end{pmatrix}_t = \frac{D}{\Delta x^2} \begin{pmatrix} \frac{b_2}{2} \\ \frac{b_3}{8} \\ \frac{b_2}{24} \\ \frac{3b_3}{160} \end{pmatrix}. \tag{6.114}$$

We use LL-RK to solve for the coefficients b_i at the two Radau points ($\tau = \frac{1}{3}, 1$). With f^{st} defined at the Radau points, we proceed to numerically integrate Eqn 6.110 in time.

We test HH-RDG ($p = 1$) with a solution depicting a decaying sine wave,

$$u_0 = \sin(x), \quad x \in [0\ 2\pi], \tag{6.115}$$

$$u(t) = e^{-\mu t}\sin(x). \tag{6.116}$$

The numerical results are presented in Table 6.12. The HH-RDG ($p = 1$) scheme achieves 4th-order accuracy with just one flux solver at $\tau = 0$ for each time step. This is impressive since it achieves the same order of accuracy as RK-RDG ($p = 1$) which requires three flux evaluations.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	5.57×10^{-5}		2.12×10^{-5}	
20	40	3.46×10^{-6}	4.0	6.60×10^{-7}	5.0
40	80	2.16×10^{-7}	4.0	2.03×10^{-8}	5.0
80	160	1.35×10^{-8}	4.0	6.38×10^{-10}	5.0
160	320	8.43×10^{-10}	4.0	2.02×10^{-11}	5.0
320	640	5.27×10^{-11}	4.0	5.60×10^{-13}	5.2

Table 6.12: HH-RDG ($p = 1$) allows a maximum VNN of $\frac{1}{6}$ for the decaying sine-wave problem with $\mu = 1$ and $t_{final} = 2$. A periodic boundary condition is applied. The scheme achieves the same order of accuracy as RK-RDG ($p = 1$).

Time-accurate linear diffusion, $p = 2$, periodic boundary condition

With a piecewise-quadratic solution, RDG recovers an unique f at $\tau = 0$,

$$f = b_0\psi_0 + b_1\psi_1 + b_2\psi_2 + b_3\psi_3 + b_4\psi_4 + b_5\psi_5. \tag{6.117}$$

This time we use an orthogonal recovery basis for fun,

$$\begin{matrix} \psi_1 = 1 \\ \psi_2 = 2r \\ \psi_3 = 6r^2 - \frac{1}{2} \\ \psi_4 = 20r^3 - 3r \\ \psi_5 = 70r^4 - 15r^2 + \frac{3}{8} \\ \psi_6 = 252r^5 - 70r^3 + \frac{15}{4}r \end{matrix} \quad , \quad r \in \left[-\frac{1}{2}\ \frac{1}{2} \right]. \tag{6.118}$$

Inserting these test functions into Eqn 6.111 results in

$$\begin{pmatrix} b_0 \\ \frac{b_1}{3} \\ \frac{b_2}{5} \\ \frac{b_3}{7} \\ \frac{b_4}{9} \\ \frac{b_5}{11} \end{pmatrix}_t = \frac{D}{\Delta x^2} \begin{pmatrix} 3b_2 + 10b_4 \\ 5b_3 + 14b_5 \\ 7b_4 \\ 9b_5 \\ 0 \\ 0 \end{pmatrix}. \tag{6.119}$$

We use LL-RK to solve for the six coefficients b_i at the three Radau points ($\tau =$ 0.15, 0.64, 1.00). We solve the same periodic decaying sine-wave problem and print the results in Table 6.13. Notice the scheme is hitting computer zero on the two finest

grids. The order of accuracy appears to be 8th-order in cell average, 7th-order in 1st averaged gradient and finally 6th-order in 2nd-averaged gradient. HH-RDG ($p = 2$) is able to achieve a maximum VNN of $\frac{1}{10}$, which suggests it is still faster than the finite-difference schemes with subgrid information presented in the previous section.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	$L_2\,\overline{\triangle^2 u}_{error}$	Rate
10	30	2.35×10^{-10}		3.12×10^{-9}		1.06×10^{-8}	
20	80	1.05×10^{-12}	7.9	2.15×10^{-11}	7.2	2.46×10^{-10}	5.4
40	120	4.22×10^{-15}	7.9	1.83×10^{-13}	6.9	4.96×10^{-12}	5.6
80	240	8.94×10^{-16}	2.2	3.14×10^{-15}	5.9	5.58×10^{-14}	6.5
160	480	8.60×10^{-16}	0.1	5.47×10^{-17}	5.8	5.04×10^{-15}	3.5

Table 6.13: HH-RDG ($p = 2$) allows a maximum VNN of $\frac{1}{10}$ for the decaying sine-wave problem with $\mu = 1$ and $t_{final} = 2$. A periodic boundary condition is applied.

Time-accurate linear diffusion, $p = 1$, Dirichlet boundary condition

We implement Dirichlet boundary conditions on the left ($x = 0$) and on the right ($x = 2\pi$),

$$u(0) = u_L^C = 0, \tag{6.120}$$

$$u(2\pi) = u_R^C = 0, \tag{6.121}$$

where the superscript C stands for constant. When first applying recovery at the boundaries for $\tau = 0$, we require the recovered function to satisfy the Dirchlet condition (see Section 3.4.1 for the full boundary-recovered function). For HH-RDG, we require the time-evolution of the recovered function to satisfy three of the highest moments of the PDE, and one strong condition,

$$\begin{aligned} \int_{\Omega_j \bigcup \Omega_{j+1}} \psi_i \frac{\partial f_{j+\frac{1}{2}}}{\partial t}\, dx &= \int_{\Omega_j \bigcup \Omega_{j+1}} \psi_i \frac{\partial^2 f_{j+\frac{1}{2}}}{\partial x^2}\, dx. \quad i = 1, 2, 3 \\ \frac{\partial}{\partial t} f_{j+\frac{1}{2}}(\tau)\,|_{r=0} &= 0. \end{aligned}$$

Notice the second equation is a strong statement for the boundary, while the first equation is the usual weak statement for the interior scheme. The last equation simply states that the Dirichlet boundary condition remains steady in time. Inserting

$f = \sum_{i=1}^{4} b_i r^i$ for the left Dirichlet boundary results in

$$\begin{pmatrix} b_0 \\ \frac{b_0}{2} + \frac{b_1}{3} + \frac{b_2}{4} + \frac{b_3}{5} \\ \frac{b_0}{3} + \frac{b_1}{4} + \frac{b_2}{5} + \frac{b_3}{6} \\ \frac{b_0}{4} + \frac{b_1}{5} + \frac{b_2}{6} + \frac{b_3}{7} \end{pmatrix}_t = \begin{pmatrix} u_L^C \\ \frac{D}{\Delta x^2}\,(b_2 + 2b_3) \\ \frac{D}{\Delta x^2}\left(\frac{2b_2}{3} + \frac{3b_3}{2}\right) \\ \frac{D}{\Delta x^2}\left(\frac{b_2}{2} + \frac{6b_3}{5}\right) \end{pmatrix}. \tag{6.122}$$

Similarly, for the right Dirichlet boundary condition,

$$\begin{pmatrix} b_0 \\ -\frac{b_0}{2} + \frac{b_1}{3} - \frac{b_2}{4} + \frac{b_3}{5} \\ \frac{b_0}{3} - \frac{b_1}{4} + \frac{b_2}{5} - \frac{b_3}{6} \\ -\frac{b_0}{4} + \frac{b_1}{5} - \frac{b_2}{6} + \frac{b_3}{7} \end{pmatrix}_t = \begin{pmatrix} u_R^C \\ \frac{D}{\Delta x^2}\,(-b_2 + 2b_3) \\ \frac{D}{\Delta x^2}\left(\frac{2b_2}{3} - \frac{3b_3}{2}\right) \\ \frac{D}{\Delta x^2}\left(-\frac{b_2}{2} + \frac{6b_3}{5}\right) \end{pmatrix}. \tag{6.123}$$

The results of HH-RDG ($p = 1$) with full boundary-recovered function are presented in Table 6.14.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	5.57×10^{-5}		3.37×10^{-5}	
20	40	3.45×10^{-6}	4.0	1.06×10^{-6}	5.0
40	80	2.16×10^{-7}	4.0	3.53×10^{-8}	4.9
80	160	1.35×10^{-8}	4.0	1.08×10^{-9}	5.0
160	320	8.43×10^{-10}	4.0	3.18×10^{-11}	5.1
320	640	5.27×10^{-11}	4.0	1.52×10^{-12}	4.4

Table 6.14: HH-RDG ($p = 1$) obtains a maximum VNN of $\frac{1}{6}$ for the decaying sine-wave problem with $\mu = 1$ and $t_{final} = 2$. Dirichlet boundary conditions on both sides are satisfied with full boundary-recovered function.

We now demonstrate the downside of doing things wrongly at the boundary. Let us naively have f evolve via LL-RK at the boundary element without satisfying the boundary condition, hence the local evolution equation is the same as the one for the interior scheme. For f on the left side of the numerical domain,

$$\begin{pmatrix} b_0 + \frac{b_1}{2} + \frac{b_2}{3} + \frac{b_3}{4} \\ \frac{b_0}{2} + \frac{b_1}{3} + \frac{b_2}{4} + \frac{b_3}{5} \\ \frac{b_0}{3} + \frac{b_1}{4} + \frac{b_2}{5} + \frac{b_3}{6} \\ \frac{b_0}{4} + \frac{b_1}{5} + \frac{b_2}{6} + \frac{b_3}{7} \end{pmatrix}_t = \frac{D}{\Delta x^2} \begin{pmatrix} 2b_2 + 3b_3 \\ b_2 + 2b_3 \\ \frac{2b_2}{3} + \frac{3b_3}{2} \\ \frac{b_2}{2} + \frac{6b_3}{5} \end{pmatrix}, \tag{6.124}$$

and for f on the right side of the numerical domain,

$$\begin{pmatrix} b_0 - \frac{b_1}{2} + \frac{b_2}{3} - \frac{b_3}{4} \\ -\frac{b_0}{2} + \frac{b_1}{3} - \frac{b_2}{4} + \frac{b_3}{5} \\ \frac{b_0}{3} - \frac{b_1}{4} + \frac{b_2}{5} - \frac{b_3}{6} \\ -\frac{b_0}{4} + \frac{b_1}{5} - \frac{b_2}{6} + \frac{b_3}{7} \end{pmatrix}_t = \frac{D}{\Delta x^2} \begin{pmatrix} 2b_2 - 3b_3 \\ -b_2 + 2b_3 \\ \frac{2b_2}{3} - \frac{3b_3}{2} \\ -\frac{b_2}{2} + \frac{6b_3}{5} \end{pmatrix}. \tag{6.125}$$

One might immediately question whether f will satisfy the boundary condition during a local evolution. Table 6.15 shows this naive scheme does satisfy the boundary condition, however the scheme suffers a 50% time-step restriction.

N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10	20	2.13×10^{-5}		2.09×10^{-5}	
20	40	1.73×10^{-6}	3.6	5.53×10^{-7}	5.2
40	80	1.22×10^{-7}	3.8	1.69×10^{-8}	5.0
80	160	8.01×10^{-8}	3.9	5.23×10^{-10}	5.0
160	320	5.14×10^{-10}	4.0	1.63×10^{-11}	5.0
320	640	3.26×10^{-11}	4.0	5.07×10^{-13}	5.0

Table 6.15: Naive Dirichlet boundary scheme. HH-RDG ($p = 1$) with a maximum VNN of $\frac{1}{12}$ for a decaying sine-wave problem with $\mu = 1$ and $t_{final} = 2$. The naive method suffers reduction in both accuracy and VNN.

Fourier Analysis of HH-RDG for linear diffusion, $p = 1, 2$

In order to rid critics of all skeptism, we performed Fourier analysis on the the HH-RDG ($p = 1$) scheme to determine the stability of the scheme. We rewrite the final update equations for the two variables in matrix form,

$$\frac{\partial}{\partial t}\left(\vec{u}_j\right) = \left(\boldsymbol{M}_L \boldsymbol{M}_C \boldsymbol{M}_R\right) \begin{pmatrix} T^{-1}\left(\vec{u}_j\right) \\ I\left(\vec{u}_j\right) \\ T^{+1}\left(\vec{u}_j\right) \end{pmatrix}, \tag{6.126}$$

$$\boldsymbol{M}_L = \begin{pmatrix} \frac{9r}{4} - \frac{15r^2}{2} & \frac{5r}{4} - \frac{15r^2}{2} \\ \frac{45r^2}{2} - \frac{15r}{4} & \frac{33r^2}{2} - \frac{7r}{4} \end{pmatrix}, \tag{6.127}$$

$$\boldsymbol{M}_C = \begin{pmatrix} 15r^2 - \frac{9r}{2} + 1 & 0 \\ 0 & 57r^2 - \frac{23r}{2} + 1 \end{pmatrix}, \tag{6.128}$$

$$\boldsymbol{M}_R = \begin{pmatrix} \frac{9r}{4} - \frac{15r^2}{2} & \frac{15r^2}{2} - \frac{5r}{4} \\ \frac{15r}{4} - \frac{45r^2}{2} & \frac{33r^2}{2} - \frac{7r}{4} \end{pmatrix}, \tag{6.129}$$

where r is the Von Neumann number. We translate the shift operators into Fourier modes,

$$\frac{\partial}{\partial t}\left(\vec{u}_j\right) = \begin{pmatrix} 15r^2 - \frac{9r}{2} + 2e^{-i\beta}\left(\frac{9r}{4} - \frac{15r^2}{2}\right) + 1 & e^{-i\beta}\left(\frac{5r}{4} - \frac{15r^2}{2}\right) + e^{-i\beta}\left(\frac{15r^2}{2} - \frac{5r}{4}\right) \\ e^{-i\beta}\left(\frac{15r}{4} - \frac{45r^2}{2}\right) + e^{-i\beta}\left(\frac{45r^2}{2} - \frac{15r}{4}\right) & 57r^2 - \frac{23r}{2} + 2e^{-i\beta}\left(\frac{33r^2}{2} - \frac{7r}{4}\right) + 1 \end{pmatrix}\left(\vec{u}_j\right).$$

The two eigenvalues associated with the matrix above read,

$$\lambda_1 = \frac{1}{2}\left(-30r^2e^{-i\beta} + 30r^2 + 9re^{-i\beta} - 9r + 2\right), \tag{6.130}$$

$$\lambda_2 = \frac{1}{2}e^{-i\beta}\left(114r^2e^{i\beta} + 66r^2 - 23re^{i\beta} - 7r + 2e^{i\beta}\right). \tag{6.131}$$

Stability requires the magnitude of the eigenvalues as a function of r must not exceed unity for any β. Figure 6.10 shows the polar plot of the eigenvalues for $r = \frac{1}{6}$ and $r = \frac{1}{5}$. The two eigenvalues overlap each other for $r = \frac{1}{6}$, and remains bounded within the unit circle. However, for r larger than $\frac{1}{6}$, one of the eigenvalue lies outside of the unit circle implying instability. We go through the same analysis procedure for $p = 2$ and only present the final results in Figure 6.11.

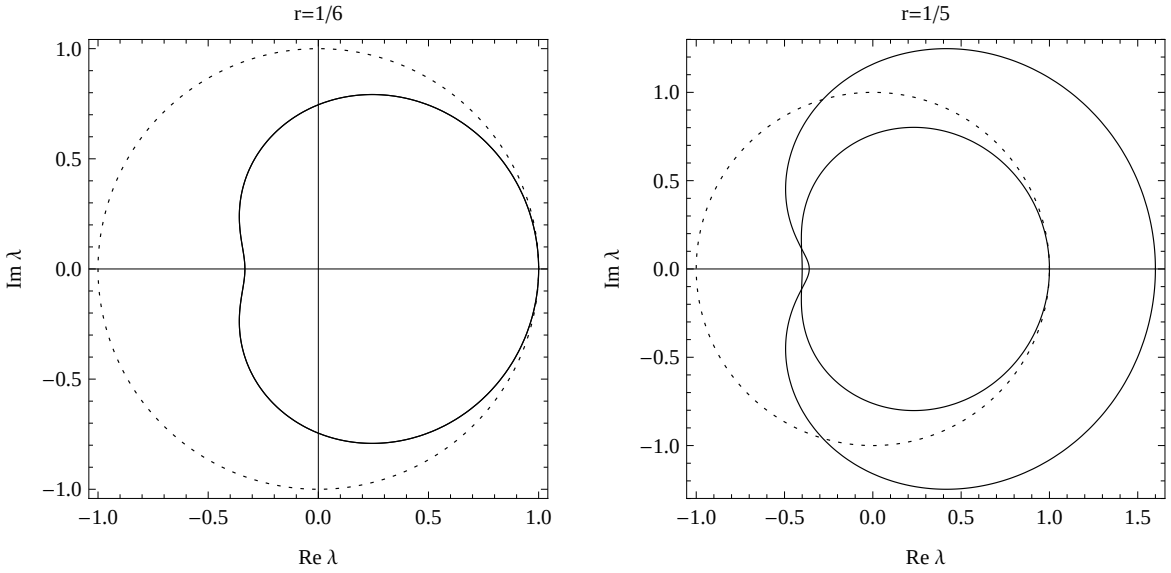

Figure 6.10: Polar plots in the complex plane of the two eigenvalues of the update matrix of HH-RDG ($p = 1$) scheme. The dashed line indicates the stability boundary. For $r = \frac{1}{6}$, the two eigenvalues coincide. For anything larger than $r = \frac{1}{6}$, one eigenvalue lies outside of the stability domain.

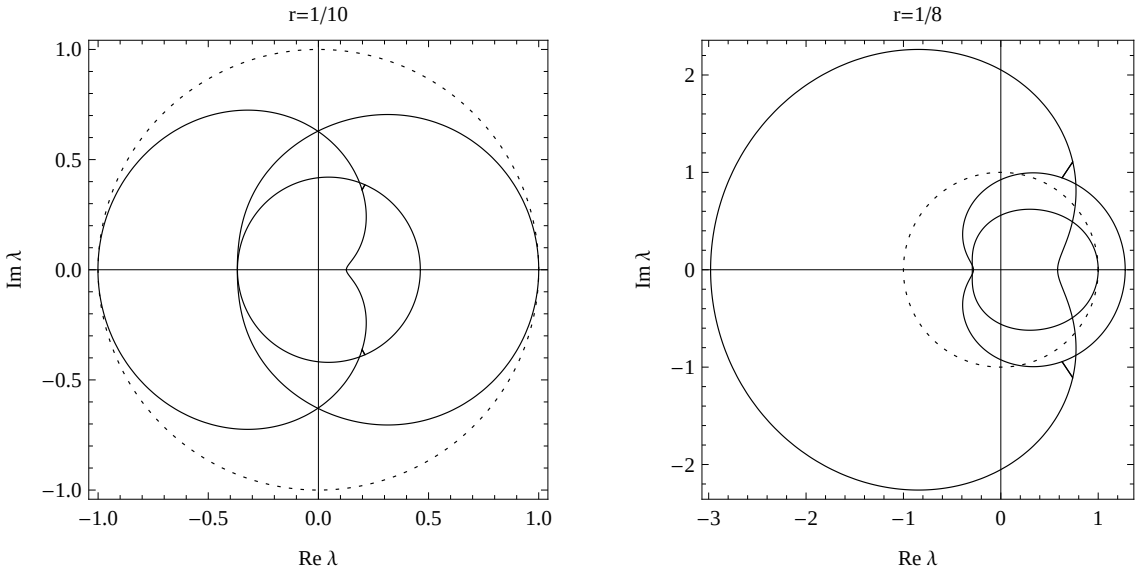

Figure 6.11: Polar plots in the complex plane of the two eigenvalues of the update matrix of HH-RDG ($p = 2$) scheme. The dashed line indicates the stability boundary. For $r = \frac{1}{10}$, the three eigenvalues remain bounded by the stability circle. For r larger than $\frac{1}{10}$, one eigenvalue lies outside of the stability domain.

Summary of HH-RDG for diffusion

We have formulated the HH-RDG scheme for time-accurate diffusion problems. Numerical experiments show that the accuracy obtained by HH-RDG is on par with RK-RDG. Although a direct comparison in total CPU time is not available, HH-RDG requires many fewer binary-recovery operations than RD-RDG. We also demonstrated how to incorporate boundary conditions into the HH-RDG. Fourier analysis for $p = 1$ and 2 confirms the maximum VNN found in numerical experiments. The journey to develop the HH-RDG scheme has been a long one due to the diffence in advection and diffusion operators. We are now ready to observe the behavior of the combined HH-DG advection and diffusion operators.

6.4 HH-DG linear advection-diffusion in 1-D

Most literature in CFD lumps together advection and diffusion terms and then magically approximates them differently (as with LDG in Section 1.4.2). Throughout the entire span of this thesis, we emphasize advection and diffusion are completely different physical phenomena and require completely different numerical approaches.

This section utilizes the HH-DG upwind operator for advection and the HH-RDG operator for diffusion to solve for different parts of the governing equation. We accentuate the fact that our operators are completely independent of each other, and require no ad-hoc parameters for stability or error tuning.

We solve the linear advection of the form,

$$u_t + a u_x = D u_{xx}, \tag{6.132}$$

where a is the constant advection speed and D is the constant diffusion coefficient. The update equation is obtained by testing the equation in both space and time, after suitable integrations by parts (once for advection term and twice for diffusion term),

$$\begin{aligned} \iint v u_t \, dxdt &= -a \iint v u_x \, dxdt + \mu \iint v u_{xx} \, dxdt, \\ &= -a \int \left(\oint v u \, dx - \int v_x u \, dx \right) dt \\ &\quad + D \int \left(\oint (v u_x - v_x u) \, dx + \int v_{xx} u \, dx \right) dt. \end{aligned} \tag{6.133}$$

We discretize by applying the equation to each element Ω_j and replacing global coordinates with local coordinates in Eqn 6.5,

$$\begin{aligned} \int_{\Omega_j} v \left(u^{n+1} - u^n \right) d\xi &= \underbrace{-\frac{a\Delta t}{\Delta x} \int \left([vu] \, |_0^1 - \int_{\Omega_j} v_\xi u \, d\xi \right) d\tau}_{\text{advection}} \\ &\quad \underbrace{+\frac{D\Delta t}{\Delta x^2} \int \left([v u_\xi - v_\xi u] \, |_0^1 + \int_{\Omega_j} v_{\xi\xi} u \, d\xi \right) d\tau}_{\text{diffusion}}. \end{aligned} \tag{6.134}$$

Notice all the terms on the RHS appear to be only a function of space because we have yet to implement our new ideas. For the advection terms, we apply Hancock's observation and replace the solution in the surface integral with $u^{st}(\xi, \tau)$; following Huynh we compute the volume integral with $u^{n+\tau}$. For the diffusion terms, we replace the solution in the surface integral with $f^{st}(\tau)$ and in the volume integral with $u^{n+\tau}$. The weak space-time expansion for u is obtained by using LL-RK (see Eqn 6.17) on

the following system,

$$\Delta x \int v u_t \, d\xi = \left(-a \left([vu] \, |_0^1 - \int_{\Omega_j} v_\xi u \, d\xi \right) + \frac{D}{\Delta x} \left([v u_\xi - v_\xi u] \, |_0^1 + \int_{\Omega_j} v_{\xi\xi} u \, d\xi \right) \right). \tag{6.135}$$

In contrast, the weak space-time expansion for f is acquired by sharing moments with the original governing equation,

$$\int_{\Omega_j \bigcup \Omega_{j+1}} \psi \, f_{t,j+\frac{1}{2}} \, dx \;\; = \int_{\Omega_j \bigcup \Omega_{j+1}} \psi \;\; \left(-a f_{x,j+\frac{1}{2}} + D f_{xx,j+\frac{1}{2}} \right) dx. \tag{6.136}$$

The space-time solutions of both u and f are advanced to the Radau points in time and then stored. The true purpose of this short study is to analyze the relationship between the advection and diffusion operators. Physically speaking, when the advection phenomenon dominates the diffusion phenomenon, we expect the order of accuracy and time-step restriction to approach those of the advection operator. Likewise, we expect the opposite when diffusion dominates the flow. The balance between advection and diffusion is described by the global Péclet number,

$$Pe_G = \frac{aL}{D}, \tag{6.137}$$

where L is the characteristic length scale of the physical domain of interest. The global Péclet number approaches zero ($Pe_G \to 0$) for diffusion-dominated problems, and infinity ($Pe_G \to \infty$) for advection-dominated problems. We also define a local Péclet number, which is more relevant to our numerical studies,

$$Pe_L = \frac{a\Delta x}{D}, \tag{6.138}$$

where the characteristic length scale is now the width of the element. A grid-refinement study is always performed with constant global Péclet number.

The local Péclet number is also the ratio between the CFL number and VNN,

$$Pe_L = \left(\frac{a\Delta t}{\Delta x} \right) / \left(\frac{D\Delta t}{\Delta x^2} \right) = \frac{a\Delta x}{D}, \tag{6.139}$$

and appears in the stability condition of explicit advection-diffusion schemes. Take, e.g, the simplest upwind-advection/central-diffusion difference scheme; a Fourier anal-

ysis yields the stability condition

$$\mathrm{CFL} + 2\mathrm{VNN} \leq 1, \tag{6.140}$$

or

$$\frac{a\Delta t}{\Delta x} + \frac{2D\Delta t}{\Delta x^2} \leq 1. \tag{6.141}$$

This can be reduced to:

$$\Delta t \leq \frac{\Delta x}{a} \frac{Pe_L}{2 + Pe_L}. \tag{6.142}$$

Let us vary Pe_L by varying Δx at constant a and D. For large Pe_L (large Δx) Eqn 6.142 approaches the CFL condition, and $\Delta t \sim \Delta x$. For small Pe_L (small Δx), Δt becomes proportional to Δx^2 and the condition reduces to $\mathrm{VNN} < \frac{1}{2}$. For the HH-DG schemes we shall assume the following form of the stability condition:

$$\frac{1}{\mathrm{VNN}} \cdot \frac{D\Delta t}{\Delta x^2} + \frac{1}{\mathrm{CFL}} \cdot \frac{a\Delta t}{\Delta x} \leq 1 \cdot C_{\text{Adv-Diff}}, \tag{6.143}$$

where $\mathrm{CFL} = 1$ for all HH-DG advection operator using upwind flux, and VNN is the highest stable Von Neumann number for pure diffusion determined from the previous sections. The coefficient $C_{\text{Adv-Diff}}$ is a safety factor. Solving for Δt gives

$$\Delta t \leq \frac{C_{\text{Adv-Diff}}}{\frac{1}{\mathrm{VNN}} \cdot \frac{D}{\Delta x^2} + \frac{1}{\mathrm{CFL}} \frac{a}{\Delta x}}. \tag{6.144}$$

In pure diffusion or advection cases, the constant coefficient $C_{\text{Adv-Diff}}$ equals 1, however it is numerically found that for this value accuracy is reduced for mixed advection-diffusion problems. Simply setting $C_{\text{Adv-Diff}} \leq 1$ recovers the desired order of accuracy. We will next look at detailed derivations for piecewise-linear ($p = 1$) and piecewise-quadratic ($p = 2$).

6.4.1 Piecewise-linear & piecewise-quadratic HH-DG for linear advection-diffusion equation

Perhaps one of the greatest advantages of using orthogonal basis functions is that the update equations of the moments become independent of each other (on the LHS). For example, this section hits two birds with one stone by deriving the update equations for a piecewise-quadratic scheme: one easily gets the piecewise-linear scheme by setting all appearances of $\overline{\Delta^2 u}$ to zero. Let the P^2 polynomial space be spanned by

the orthogonal Legendre polynomials,

$$\begin{aligned} v_0 &= 1, \\ v_1 &= 2\xi - 1, \\ v_2 &= 6\xi^2 - 6\xi + 1, \end{aligned}$$

and the solution be

$$u_j(\xi) = \overline{u}_j v_0 + \overline{\Delta u}_j v_1 + \overline{\Delta^2 u}_j v_2, \quad \xi \in [0\ 1], \tag{6.145}$$

where the variables $\overline{u}$, $\overline{\Delta u}$, and $\overline{\Delta^2 u}$ represents the cell average, first average gradient and second average gradient, respectively. We first write out the final update equations in full by inserting the Legendre basis functions into Eqn 6.134, and then determine the missing pieces of the puzzle,

$$\begin{aligned} \overline{u}_j^{n+\tau} &= \overline{u}_j^n - \frac{a\Delta t}{\Delta x}\int_0^\tau \left(u^{st}(1,\tau) - u^{st}(0,\tau)\right) d\tau \\ &\quad + \frac{D\Delta t}{\Delta x^2}\int_0^\tau \left(f^{st}_{\xi,j+\frac{1}{2}}(\tau) - f^{st}_{\xi,j-\frac{1}{2}}(\tau)\right) d\tau, \end{aligned}$$

$$\begin{aligned} \overline{\Delta u}_j^{n+\tau} &= \overline{\Delta u}_j^n - \frac{a\Delta t}{\Delta x}\int_0^\tau \left(u^{st}(1,\tau) + u^{st}(0,\tau)\right) d\tau + \frac{a\Delta t}{\Delta x}\int_0^\tau 2\overline{u}_j(\tau)\ d\tau \\ &\quad + \frac{D\Delta t}{\Delta x^2}\int_0^\tau \left(f^{st}_{\xi,j+\frac{1}{2}}(\tau) + f^{st}_{\xi,j-\frac{1}{2}}(\tau) - 2f^{st}_{j+\frac{1}{2}}(\tau) + 2f^{st}_{j-\frac{1}{2}}(\tau)\right) d\tau, \end{aligned}$$

$$\begin{aligned} \overline{\Delta^2 u}_j^{n+\tau} &= \overline{\Delta^2 u}_j^n - \frac{a\Delta t}{\Delta x}\int_0^\tau \left(u^{st}(1,\tau) - u^{st}(0,\tau)\right) d\tau \\ &\quad + \frac{a\Delta t}{\Delta x}\int_0^\tau 2\overline{\Delta u}_j(\tau)\ d\tau \\ &\quad + \frac{D\Delta t}{\Delta x^2}\int_0^\tau \left(f^{st}_{\xi,j+\frac{1}{2}}(\tau) - f^{st}_{\xi,j-\frac{1}{2}}(\tau) - 6f^{st}_{j+\frac{1}{2}}(\tau) - 6f^{st}_{j-\frac{1}{2}}(\tau)\right) d\tau \\ &\quad + \frac{D\Delta t}{\Delta x^2}\int_0^\tau 12\overline{u}_j(\tau)\ d\tau. \end{aligned} \tag{6.146}$$

In practice, we never solve for explicit expressions for $u^{st}(\tau)$ and $f^{st}(\tau)$. Instead, only their values at the Radau points in time are needed for the Gaussian integration in time. We feed u and f at $\tau = 0$ into LL-RK and solve for their values at the 2 Radau points for $p = 1$, and at the 3 Radau points for $p = 2$. The LL-RK equations

for u^{st} are very similar to the ones above, except there is no integration in time, and all fluxes are obtained from inner element values,

$$\overline{u}_{j,t} = -\frac{a}{\Delta x}\left(u\left(1\right)-u\left(0\right)\right)+\frac{D}{\Delta x^2}\left(u_\xi\left(1\right)-u_\xi\left(0\right)\right),$$

$$\begin{aligned}\left(\overline{\Delta u}_j\right)_t &= -\frac{a}{\Delta x}\left(u\left(1\right)+u\left(0\right)\right)\\ &\quad +\frac{a}{\Delta x}\left(2\overline{u}_j\right)+\frac{D}{\Delta x^2}\left(u_\xi\left(1\right)+u_\xi\left(0\right)-2u\left(1\right)+2u\left(0\right)\right),\end{aligned}$$

$$\begin{aligned}\left(\overline{\Delta^2 u}_j\right)_t &= -\frac{a}{\Delta x}\left(u\left(1\right)-u\left(0\right)\right)\\ &\quad +\frac{a}{\Delta x}\left(2\overline{\Delta u}_j\right)+\frac{D}{\Delta x^2}\left(u_\xi\left(1\right)-u_\xi\left(0\right)-6u\left(1\right)-6u\left(0\right)\right)\\ &\quad +\frac{D}{\Delta x^2}\left(12\overline{u}_j\right).\end{aligned} \tag{6.147}$$

We get the $p=1$ update equations by setting $\overline{\Delta^2 u}$ to zeros wherever it appears. The same procedure follows for f^{st}. Interface recovery generates a smooth quintic polynomial at the interface with 6 unique coefficients,

$$f_{j+\frac{1}{2}} = b_{0,j+\frac{1}{2}}+b_{1,j+\frac{1}{2}}r+b_{2,j+\frac{1}{2}}r^2+b_{3,j+\frac{1}{2}}r^3+b_{4,j+\frac{1}{2}}r^4+b_{5,j+\frac{1}{2}}r^5,\quad r\in[-\frac{1}{2}\ \frac{1}{2}], \tag{6.148}$$

where $r=0$ is the interface; it must satisfy 6 moments of the differential form of the governing equation. The result for $r=0$ reads,

$$f_t = 0!\,b_{0,t} = -a\left(b_1\right)+\frac{D}{\Delta x}\left(2b_2\right), \tag{6.149}$$

$$f_{tt} = 1!\,b_{1,t} = -a\left(2b_2\right)+\frac{D}{\Delta x}\left(3b_3\right), \tag{6.150}$$

$$f_{t(3)} = 2!\,b_{2,t} = -a\left(6b_3\right)+\frac{D}{\Delta x}\left(12b_4\right), \tag{6.151}$$

$$f_{t(4)} = 3!\,b_{3,t} = -a\left(24b_4\right)+\frac{D}{\Delta x}\left(60b_5\right), \tag{6.152}$$

$$f_{t(5)} = 4!\, b_{4,t} = -a\,(120 b_5)\,, \tag{6.153}$$

$$f_{t(6)} = 5!\, b_{5,t} = 0. \tag{6.154}$$

These equations can easily be modified for $p = 1$ by setting b_4 and b_5 to zero. With all pieces of the puzzles in hand, we proceed to test our HH-DG advection-diffusion operators for time-dependent problems.

6.4.2 Numerical results for linear advection-diffusion with HH-DG

We solve the simple time-dependent problem of a decaying sine wave,

$$u\,(t) = e^{-k^2 Dt} sin\, k\,(x - at)\,. \tag{6.155}$$

Observe the solution becomes zero as $t \to \infty$. Let $t = 0$ and $k = 1$ be the initial condition, and let $t = 1$ be the final time. Note this is the same test case used by Cockburn and Shu[8] to test LDG. We solve this equation for various values of the Péclet number; the numerical results for $p = 1$ are presented in Figure 6.12. The lower left corner is diffusion dominated, while the upper right corner is advection dominated; the order of accuracy transitions from 4 to slightly below 3 between these corners. We observe the order of accuracy is roughly constant along the dashed lines representing constant local Péclet number.

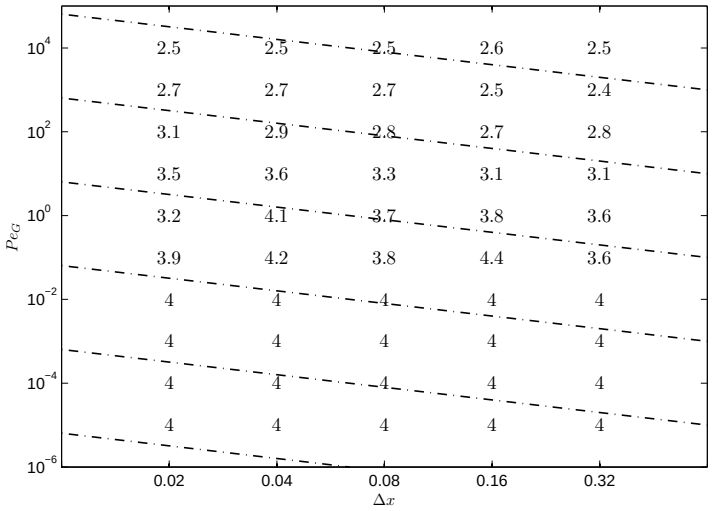

Figure 6.12: HH-DG for linear advection-diffusion problem, $p = 1$. Dashed line represents constant cell Péclet number. The order of accuracy gradually transitions from 4 at the bottom left corner to roughly 3 at the top right corner.

6.4.3 Fourier analysis of HH-DG for linear advection diffusion

This section briefly covers the Fourier analysis of HH-DG for linear advection diffusion for $p = 1$ and $p = 2$. The exact update matrices for $p = 1$ based on 3-stage LL-RK and two Radau points, and for $p = 2$ based on 5-stage LL-RK and three Radau points, are too extensive to be presented because the number of terms in the matrix elements increases with the product of the number of LL-RK stages and the number of Radau points. The following figures present the polar plots of the eigenvalues of the update matrices for a range of Péclet numbers. The time step is chosen in accordance with Eqn 6.144, with VNN $= \frac{1}{6}$ for $p = 1$, VNN $= \frac{1}{10}$ for $p = 2$ and (nominally) $C_{\text{Adv-Diff}} = 1$. It is seen that for $Pe \approx 1$ the stability of the method is reduced, requiring a reduction of $C_{\text{Adv-Diff}}$ (see Figures 6.16-6.17). For $p = 1$ this reduction is mild ($C_{\text{Adv-Diff}} = 0.9375$); for $p = 2$ it is severe ($C_{\text{Adv-Diff}} = 0.20$). It is not clear what causes this reduction in the stable time-step range when advection and diffusion are equally important; this remains a subject of future investigation.

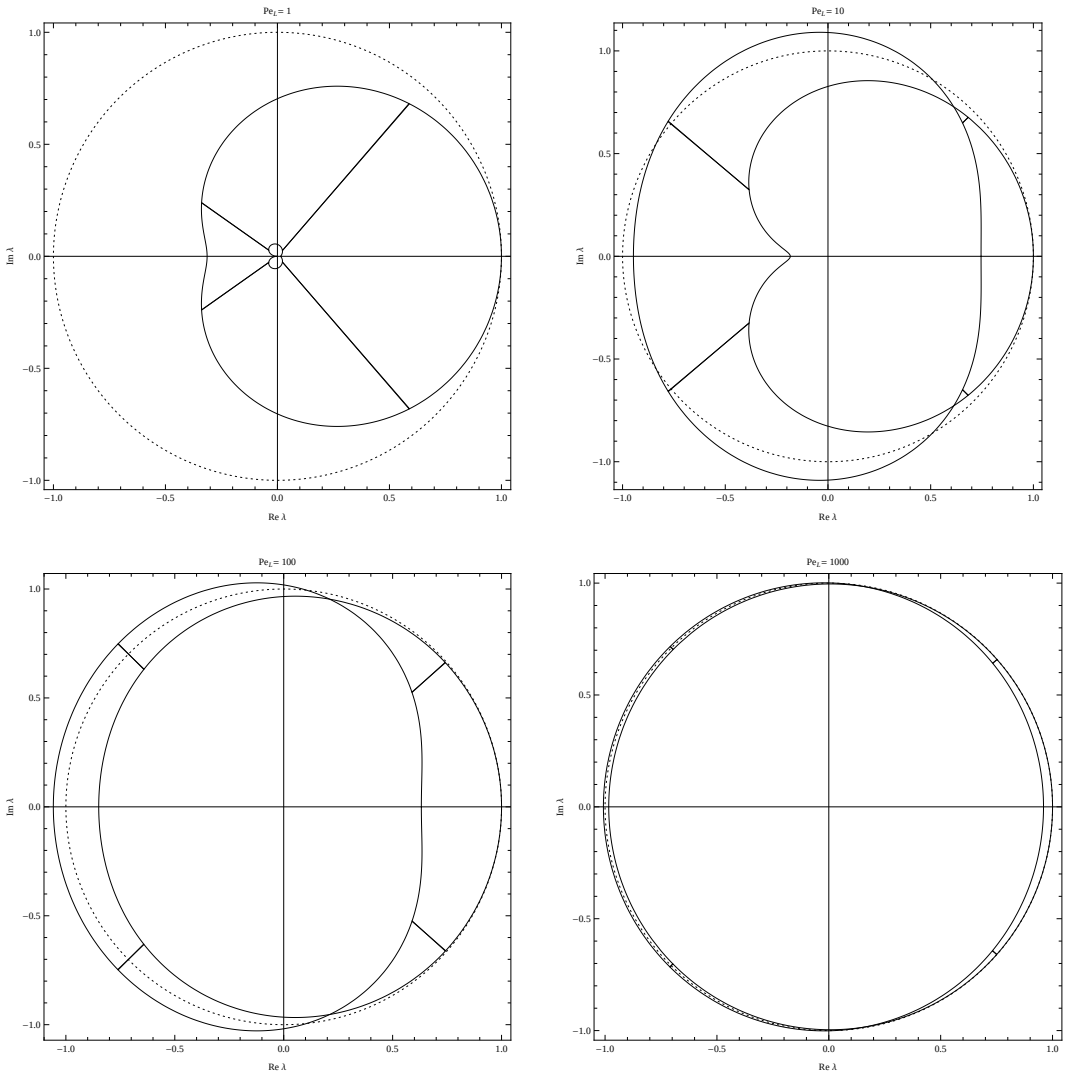

Figure 6.13: Polar plots in the complex plane of the two eigenvalues associated with HH-DG ($p = 1$) linear advection-diffusion scheme using $C_{\text{Adv-Diff}} = 1$. The unit circle is also drawn (dotted line). Notice the eigenvalues go beyond stability domain for $1 < Pe_L < 1000$. A safety factor of $C_{\text{Adv-Diff}} = 0.9375$ stabilizes the scheme for all Péclet numbers.

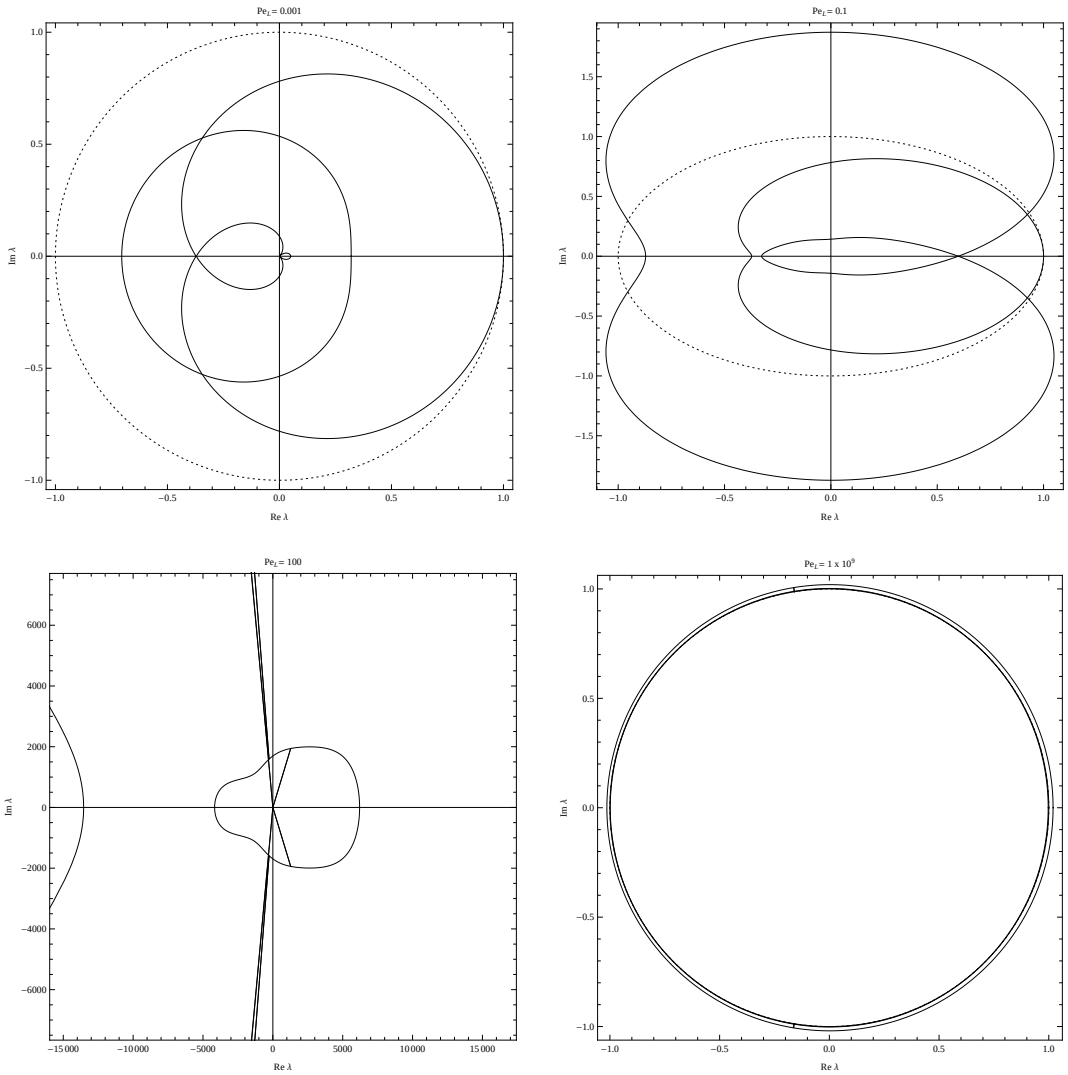

Figure 6.14: Polar plots in the complex plane of the three eigenvalues associated with HH-DG ($p = 2$) linear advection-diffusion scheme using $C_{\text{Adv-Diff}} = 1$. The unit circle is also drawn (dotted line). Notice the eigenvalues go beyond the stability domain for $10^{-3} < Pe_L < 10^8$. A safety factor of $C_{Adv-Diff} = 0.2$ stabilizes the scheme for all Péclet numbers.

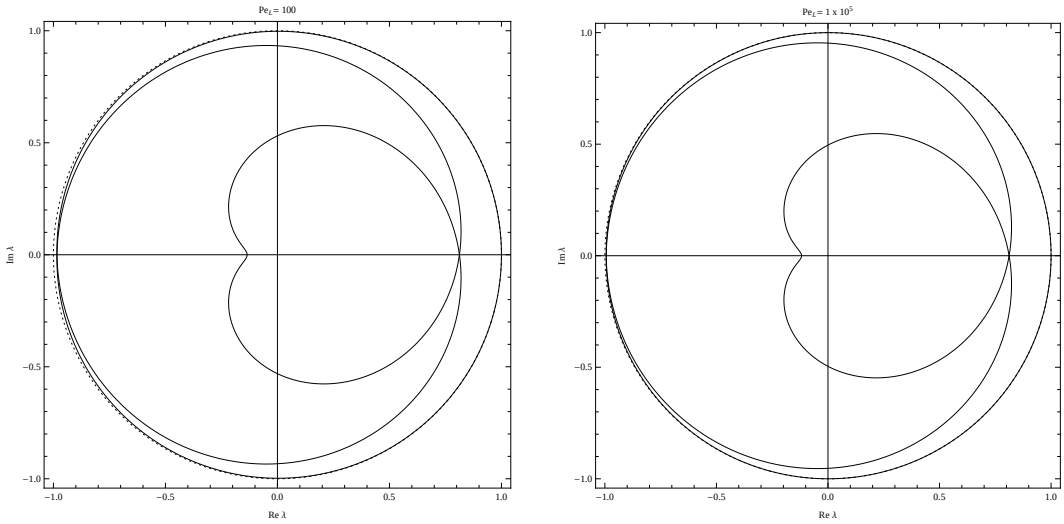

Figure 6.15: After applying a safety factor of $C_{Adv-Diff} = 0.2$, the three eigenvalues associated with HH-DG ($p = 2$) linear advection-diffusion scheme lie within the stability domain for all range of Péclet number.

6.5 Chapter summary

We have extended Huynh's moment scheme beyond piecewise linear (HH-DG), and developed a new space-time diffusion scheme for DG (HH-RDG). Common to both methods is the use of a numerical LL-RK procedure to approximate the analytical CK procedure, and the use of a hierchical procedure to include the newest information into the volume integral. We have shown once more that what works for advection does not work for diffusion. In HH-DG the solution evolves in time, while in HH-RDG both the solution and the recovered function evolve in time.

The HH-DG scheme for advection is demonstrated to be the exact shift operator with CFL = 1 for linear problems. When applied to the Euler equations, HH-DG appears to be 3rd-order accurate for $p = 1$ (as already demonstrated in [17]); the results for $p = 2$ are currently unsatisfactory and will be the subject of future investigation.

The HH-RDG scheme for diffusion is still in its infant stage of development; we have only demonstrated its ability for scalar linear diffusion problems. The results so far look promising. In comparison to RK-RDG, HH-RDG achieves the same order of accuracy with much fewer flux evaluations and a higher VNN number. Our next goal is to extend HH-RDG to nonlinear diffusion, incorporating the solution enhancement techniques of Section 3.5.1.

The study on the combined HH-DG and HH-RDG schemes for advection-diffusion

problems reveal a stability issue for a broad range of Péclet numbers. The reason for this remains unclear and will be a subject of primary focus in the future.

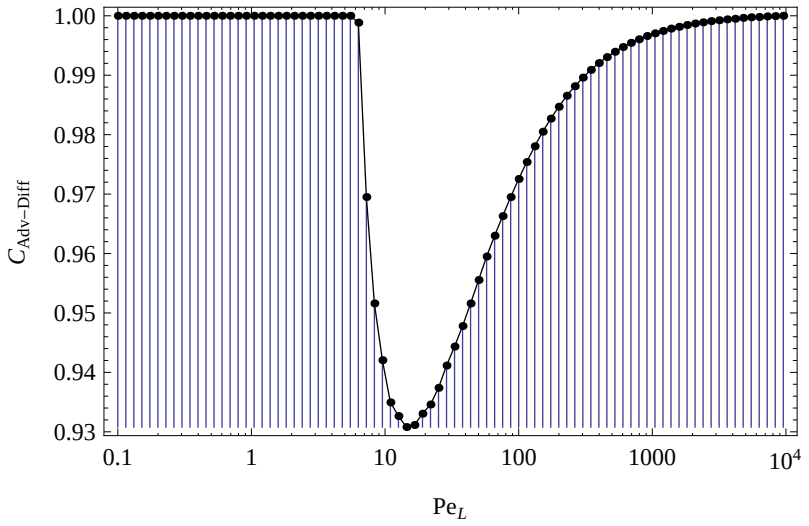

Figure 6.16: Safety factor of HH-DG ($p = 1$) linear advection-diffusion scheme applied to Δt based on Eqn 6.144

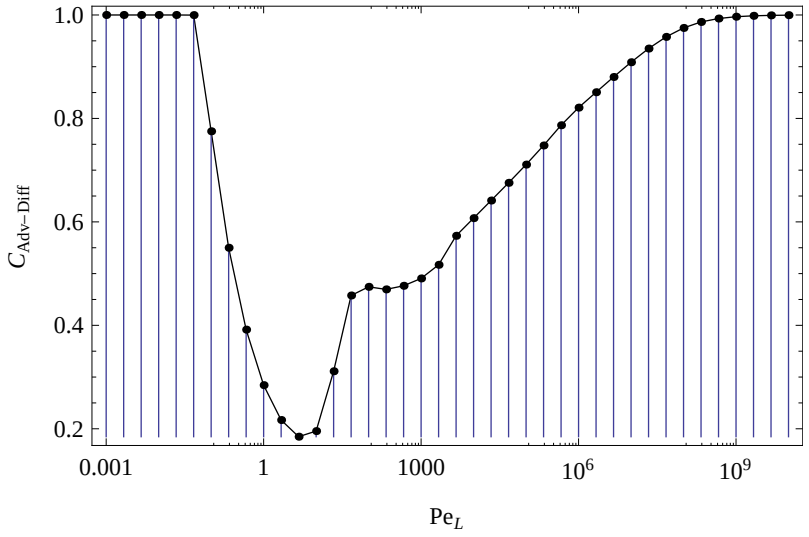

Figure 6.17: Safety factor of HH-DG ($p = 2$) linear advection-diffusion scheme applied to Δt based on Eqn 6.144

CHAPTER VII

Conclusions

Our research objectives have always been to design schemes that are conceptually easy to understand, and a "quantum" level better than existing schemes. We hope we have stayed true to our design principle and entertained our reader thus far. This thesis details the growth of two new and exciting numerical methods: interface-centered recovery-based DG (RDG) and Hancock-Huynh DG (HH-DG). The fundamental ideas behind both schemes arrive from an engineering perspective; none of our schemes are based on facilitating or reducing the number of pages of mathematical proofs. RDG is a diffusion operator for DG where the basis idea is to recover a smooth solution from the discontinuous ones. The order of accuracy and stability region of RDG are unsurpassed by any current DG diffusion schemes. HH-DG is a space-time scheme that sets itself apart from the popular Runge-Kutta DG (RK-DG) with equal or smaller number of flux evaluations, larger time-steps, and lower memory requirement. In fact, HH-DG is the exact shift operator for advection problems.

7.1 Summary

This author would like to honor Van Leer & Nomura for inventing the recovery concept[44] in 2005, and Huynh for successfully porting Hancock's version of Van Leer's scheme III into the DG framework[17] in 2006. Their work lays the foundation for the topics covered in this thesis. The major contributions include:

- Ways to reduce the condition number for binary recovery in a numerical scheme. We showed the condition number for solving the recovery equations is highly dependent on the choice of basis for the solution and the test functions. A combination of an orthogonal basis with a preconditioner will greatly reduce

the condition number.

- The concept of the smooth recovery basis (Section 2.4 and [43, 45]), which relates the discontinuous solution to the smooth solution, provides a mechanism to unify the treatment of advection and diffusion. The smooth recovery basis is designed to have the same solution coefficients as the discontinuous solutions. One can use the discontinuous basis for the advection operator, and the smooth recovery basis for the diffusion operator.

- Stability proofs for the classic RDG-2x and the naive RDG-1x are provided in the end of Chapter 2. We proved the stability of these schemes for nonlinear diffusion problems with both Dirichlet and Neumann problems.

- We further investigate the stability and accuracy of various RDG schemes via Fourier analysis. Chapter 3 provides both analytical and numerical Fourier analyses. Our investigation allowed us to better understand errors in DG schemes; there are projection and evolution errors. One must carefully design a scheme to minimize both! We showed RDG has the smallest evolution error compared with other schemes, i.e., the(σ, μ)-family and LDG.

- In order to see RDG's performance in relation to other diffusion schemes, we analyzed a venerable family of diffusion schemes: the (σ, μ)-family for $p = 1$ (see Section 3.3.3 and [42]). We were able to discover interesting lines and regions in the (σ, μ)-plane, and eventually new (σ, μ)-schemes with fast convergence rates and modest accuracy. We designed an RDG scheme, RDG-1x$\bar{f}$, to fit into the (σ, μ)-framework, and showed this scheme to be the best in the family. This study also shows that RDG-2x is different from the (σ, μ)-family, which leads to our next important discovery.

- We expressed RDG-2x in terms of penalty-like (or bilinear) terms in Section 3.3.5. RDG-2x automatically generates higher-order bilinear terms as p increases, with corresponding coefficients. This is completely different from the traditionally penalty method where the number of bilinear terms is fixed regardless of p and the corresponding coefficients are chosen in a rather ad-hoc manner.

- Our initial research was mostly focused on the theoretical analysis of RDG for the scalar linear diffusion equation. Extending RDG to nonlinear diffusion required us to refocus on algorithm development. In order to handle nonlinear

terms, we introduced a solution-enhancement technique in Section 3.5 that increases the accuracy of the evaluation of the volume integral appearing in the RDG-1x formulation. Solution enhancement is achieved by reusing information from the recovered functions to increase the order of the solution space within a cell.

- We greatly increased the scope of RDG by extending the recovery concept from 1-D to 2-D in Chapter 4. Although the recovery concept remains the same, the recovered functions at the domain boundaries require extra attention. We introduced the full boundary-recovered and compact boundary-recovered functions for Cartesian and unstructured triangular grids, respectively. We also studied the performance of RDG schemes on irregular triangular grids.

- In dealing with nonlinear problems for 2-D, the cross-derivative terms become a new obstacle for RDG (see Section 4.2.3). The binary recovery procedure creates a recovered function that is more accurate in the face-normal direction than the face-parallel direction. Having cross-derivative terms means high-order accuracy is demanded in the face-parallel direction; in this regard we introduce recovered-function enhancement to remedy the problem. The current recovered-function-enhancement technique requires the enhanced solution to create a higher-order recovered function. Unfortunately, this increases the stencil size of RDG schemes as in RDG-1x++; RDG-1x++CO is an optimized version for Cartesian grids with a smaller stencil. The numerical results for RDG-1x++CO are reassuring; it achieves the same order of accuracy for nonlinear problems with cross derivatives as the classic RDG-2x for linear problems.

- We presented numerical results for the Runge-Kutta upwind-DG (RK-uDG) for linear advection, RK-RDG for linear diffusion, and the combined RK-uDG and RK-RDG for linear advection-diffusion equations. The study on the linear advection-diffusion equation is based on the Péclet number. For advection-dominated problems, the order of accuracy approaches to that of RK-uDG, and for diffusion-dominated problems, the order of accuracy approaches that of RK-RDG scheme. There is a smooth transition in the order of accuracy between the two schemes when the Péclet number is of order unity.

- The extension of Huynh's moment scheme[17] to arbitrarily high order (HH-DG in Section 6.2), while maintaining the perfect shift property. The flux evaluation is based on the classic Hancock observation. The incorporation of

wave interaction within a cell is based on Huynh's treatment of the volume integral in the moment scheme. The HH-DG scheme is remarkably fast due to a CFL of unity, and relatively fast compared to RK-DG due to the reduced number of flux evaluations per time step. We demonstrated HH-DG abilities on both the linear advection and Euler equations; these results also confirm the viability of replacing the analytical Cauchy-Kovalevskaya (CK) procedure with a numerical one, the local linear Runge-Kutta (LL-RK) technique (see Section 6.2.1).

- The highly nonlinear extension of HH-DG to recovery (HH-RDG in Section 6.3). What works for the gander does not work for the goose; it turns out diffusion requires a completely different approach as advection. In HH-DG, we apply LL-RK to the solution to get a space-time expanded solution, but for HH-RDG, we apply LL-RK to the *recovered function* to get a space-time expanded recovered function. This has a major benefit because only one binary-recovery operation is needed at the beginning of a time step. We showed HH-RDG achieves the same order of accuracy as RK-RDG with much fewer binary-recovery operations while having the same Von Neumann numbers.

- We analyzed and tested the combined HH-DG and HH-RDG schemes for the linear advection-diffusion equation in Section 6.4. Unlike the combined RK-uDG and RK-RDG schemes, the combined HH-DG/HH-RDG scheme suffers a time-step penalty for Péclet numbers around unity. The problem appears to be growing worse for increasing p.

- The making of the poster "History of CFD: Part II" for Van Leer's AIAA Fluid Dynamics Award lecture in 2010. The people and papers discussed in this award lecture are directly related to the development of high-order advection schemes. Van Leer's scheme III[37] is the groundwork for the current HH-DG scheme.

7.2 Future work

RDG and HH-DG are "new" schemes introduced into the world of DG. These newborns are still going through development and many important questions remain at the end of this thesis. In order to address these issues, a list of future research topics is listed below:

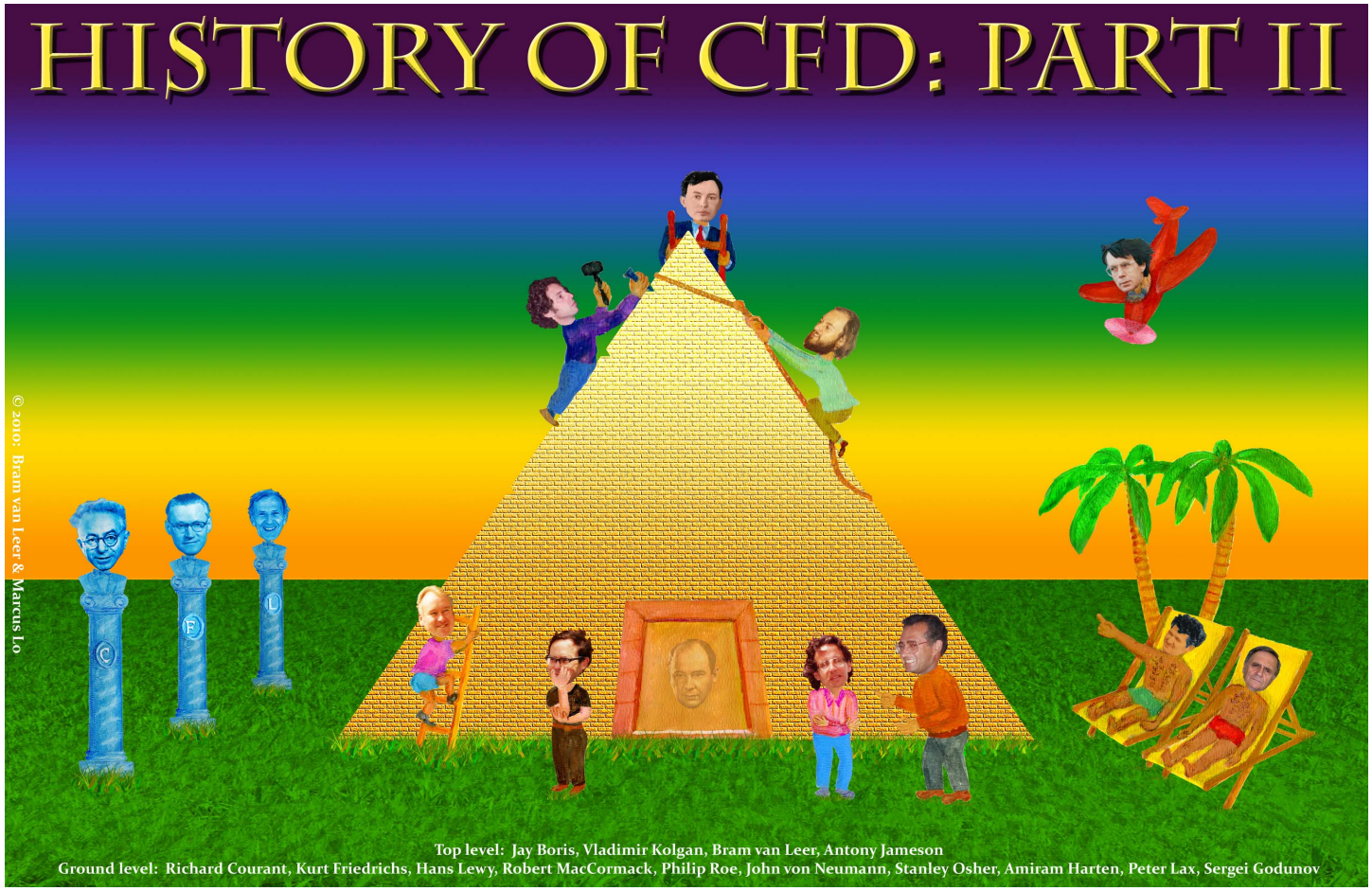

Figure 7.1: History of CFD Part II: Courtesy of Van Leer and Lo.

- The development of a compact RDG scheme for nonlinear problems with cross-derivatives terms. The current nonlinear RDG schemes utilize a large stencil that is not well suited for triangular grids or implicit time-marching schemes. The use of a compact stencil will most likely decrease the accuracy, but increases the speed of recovery since fewer equations are involved.

- We have yet to run numerical experiements on the full Navier-Stokes equations with the combined RDG and uDG schemes. Although the combined scheme worked well for linear advection-diffusion, nothing is for certain in the fully nonlinear case with shear terms.

- The results and analyses for HH-RDG are only for linear diffusion. We wish to incorporate solution- and recovered-function enhancements into HH-RDG to handle nonlinear diffusion problems.

- We wish to demonstrate local time-stepping with HH-DG. A major strength of space-time DG methods is the ability to take local time steps in each cell to reduce computational cost on grids with a large variation in cell size.

- Perhaps the most puzzling problem of HH-DG and HH-RDG is the reduction in time step for Péclet numbers close to unity. Investigating and resolving this issue is essential to turning these space-time DG methods into viable numerical methods.

7.3 The effort: revisiting the River of Recovery

The thesis summary does not show the effort in achieving each of the goals listed. Let us return to the River of Recovery in Section 1.6. The longest time and largest effort was spent between the river source and the first cascade. Although the diffusion equations we studied were linear, the analysis was hard enough and it simply took time to get a feeling for DG methods for diffusion. Crossing the nonlinearity cascade, although a harder problem, took less time owing to the insight we had gathered upstream. Crossing the cascades of the cross-derivatives was a similar challenge and took a comparable amount of research time. The work on space-time DG, as depicted in the upper side stream, started as a diversion during the second half of the research period and was carried out with interruptions.

In retrospect the hardest-won results were three in number. In the first place, the use of solution enhancement, based on recycling recovery information, to overcome

inaccuracy of the volume integral caused by nonlinearity. Second-hardest was applying the ideas of the space-time DG method for advection to diffusion. Thirdly, the further use of binary recovery steps on top of the earlier solution enhancement, in order to accurately represent cross-derivatives in the PDE.

At the end of this long journey down the river we recovered the effort spent in the form of the joy of seeing the original concept of RDG mature into a collection of methods that are suited for practical nonlinear diffusion problems.

APPENDICES

APPENDIX A

Elements of Computational Fluid Dynamics

The purpose of this section is to focus on selected fundamental ideas of computational fluid dynamics (CFD). These ideas include Gaussian integration, solution projection, derivatives of primitive variables and implicit Radau integration.

A.1 Gaussian quadrature

There is not a single bread-and-butter solution when it comes to the evaluation of finite integrals; the choice of an analytical or a numerical integration technique depends on the situation. Analytical integration is preferred when the integrand is simple and the code requires high precision. The integrand arising from a nonlinear system of equations is frequently too complex for analytical integration; in this case, a numerical technique is preferred.

The idea behind Gaussian quadrature is to express the finite integral of a function as a weighted sum of the function at specific points,

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i)\,, \tag{A.1}$$

where x_i is the Gaussian point and w_i is the corresponding weight. In general, a classical n-points Gaussian quadrature rule evaluates a polynomial of degree $2n-1$ exactly (see left diagram of Figure A.1). Note all the Gaussian points are located in the interior of the integration domain. One can think of the number of Gaussian points and their corresponding weights as degrees of freedom to represent a polynomial. If we are to force the location of one Gaussian point to the boundary (as in the

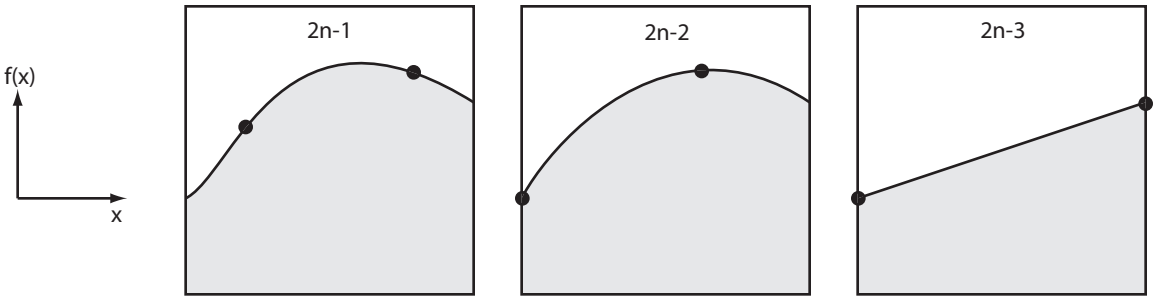

Figure A.1: Three fundamental types of Gaussian quadrature. The difference lies in the location of the endpoints, where enforcing the endpoints to coincide with the interval boundaries results in lower-order polynomial representation.

middle diagram of Figure A.1), i.e., Gauss-Radau quadrature, we will only be able to represent a polynomial of degree $2n-2$. Finally, if we are to impose two Gaussian points on the boundary (see right diagram of Figure A.1), called the Gauss-Lobatto quadrature, we will only be able to represent a polynomial of degree $2n-3$. In this thesis we frequently used the classical Gaussian quadrature for spatial integrations, and the Gauss-Radau quadrature for temporal integrations. We provide sample list of Gaussian points and weights in Table A.1-A.2 for the classical Gaussian quadrature and the Gauss-Radau for $x \in [0, 1]$.

Gaussian quadrature only works on a smooth function. The interval over which Gaussian quadrature is applied must not contain a discontinuity or a jump in the any of the derivatives. One can easily circumvent this problem by breaking up an interval such that the jumps rest on the interval boundary.

$n = 3$	x_i	w_i
	0.1127016653792583115	0.2777777777777777778
	0.5	0.4444444444444444444
	0.8872983346207416885	0.2777777777777777778

$n = 5$	x_i	w_i
	0.0469100770306680036	0.1184634425280945438
	0.230765344947158454	0.2393143352496832340
	0.5	0.2844444444444444444
	0.769234655052841546	0.2393143352496832340
	0.953089922969331994	0.1184634425280945438

Table A.1: Sample classical Gaussian quadrature points and weights for the interval $x \in [0, 1]$.

$n=2$	x_i	w_i
	1	0.25
	0.6666666666666667	0.75

$n=3$	x_i	w_i
	1	0.111111111111111
	0.644948974278318	0.512485826188422
	0.155051025721682	0.376403062700467

$n=4$	x_i	w_i
	1	0.062500000000000
	0.787659461760847	0.328844319980060
	0.409466864440735	0.388193468843172
	0.088587959512704	0.220462211176768

Table A.2: Sample Gauss-Radau quadrature points and weights for the interval $x \in [0, 1]$.

A.2 Projection of a function in real space into a finite polynomial space

In DG the downward projection is often used for initializing the discretized solution or finding the exact projected solution. Let u be an infinitely smooth function, and u_h be the discretized solution. Recall from the introduction that the discretized solution is a linear combination of a finite number of basis functions v with corresponding weights w,

$$u_h = \sum_{i=0}^{p} w_i v_i, \tag{A.2}$$

where p is the degree of the polynomial space containing the solution. To initialize the solution, the weights are calculated based on the following equality,

$$\int vu \, d\Omega = \int vu_h \, d\Omega. \tag{A.3}$$

Applying this equation for all basis functions results in $p+1$ equations for $p+1$ unknowns. A special case occurs when the basis functions are orthogonal; then the weights are determined exactly by

$$w_i = \frac{\int v_i u \, d\Omega}{\int v_i^2 \, d\Omega}. \tag{A.4}$$

The term "error" is perhaps one of the most negligently used terminologies in DG papers. If we compare solutions across different polynomial spaces, we called that the projection error. If we compare solutions in the same polynomial space (as in this thesis), we call it the evolution error. We define the L_2-norm of the projection error to be

$$e_{L_2,proj} = \left(\sum_{j=1}^{N_{ele}} \frac{\int \left(u_{h,j} - u\right)^2 d\Omega_j}{N_{ele}} \right)^{\frac{1}{2}}, \tag{A.5}$$

where N_{ele} is the total number of cells. Here we are taking the difference between the infinitely smooth solution with the discrete solution, which always results in an error of order $p+1$. This leads us to a major observation: all DG schemes contain at least a $(p+1)$-order projection error. This error is committed regardless of the numerical scheme, and hence what differentiates all numerical schemes from each other is the evolution error. The L_2-norm of the evolution error is defined to be

$$e_{L_2,evol} = \left(\sum_{j=1}^{N_{ele}} \frac{\left(\int v \left(u_{h,j} - u\right) d\Omega_j\right)^2}{N_{ele}} \right)^{\frac{1}{2}}. \tag{A.6}$$

Here we are comparing the downward projection of the exact solution with the discrete solution. With our choice of Legendre basis, we obtain the evolution error of the cell average if $v = 1$. Similarily, we obtain the evolution error of the first average gradient if $v = 2\xi - 1$. The following numerical experiment will illuminate the difference between projection and evolution errors.

Evolution error vs projection error

We consider three (σ, μ)-schemes $(p = 1)$ from Section 3.3.3 to solve a steady-state problem with a source term,

$$u_t = u_{xx} + 4\pi^2 \sin\left(2\pi x\right), \tag{A.7}$$

and with Dirichlet boundary conditions,

$$u\left(0\right) = 1, \tag{A.8}$$

$$u\left(1\right) = 0. \tag{A.9}$$

The steady solution is

$$u\left(x, \infty\right) = 1 - x + \sin\left(2\pi x\right). \tag{A.10}$$

on the domain $x \in [0,1]$. The first scheme is the Symmetric/Arnold $(-1,1)$ with 2nd-order-accurate eigenvalue. The second scheme is a new efficient scheme $(-0.99, 1.01)$ with 3rd-order-accurate eigenvalue. The last scheme is the exceptional RDG-1x$\bar{f}$ $(0.25, 2.25)$ with 4th-order-accurate eigenvalue. The problem starts with the exact solution, hence the schemes should preserve the initial solution as well as they can. Table A.3 shows the initial projection errors of three different (σ,μ)-schemes to be equivalent and the order of convergence of $e_{L_2,proj}$ to be 2.

N_{ele}	$(\sigma,\mu)=(-1,1)$	$(\sigma,\mu)=(-0.99,1,01)$	$(\sigma,\mu)=(0.25,2.25)$		Rate		
10	0.0104	0.0104	0.0104				
20	2.60×10^{-3}	2.60×10^{-3}	2.60×10^{-3}		2	2	2
40	6.50×10^{-4}	6.50×10^{-4}	6.50×10^{-4}		2	2	2
80	1.60×10^{-4}	1.60×10^{-4}	1.60×10^{-4}		2	2	2

Table A.3: Initial projection error $e_{L_2,proj}$ of three different (σ,μ)-schemes at $t=0$.

We next look at the evolution error of the cell average after the solution has converged in Table A.4. Here we are clearly able to see the difference between the three schemes. The $(-1,1)$-scheme has an evolution error that is of the same size as its initial projection error. Both $(-0.99,1,01)$- and $(0.25,2.25)$-schemes have evolution errors smaller than the projection error.

N_{ele}	$(\sigma,\mu)=(-1,1)$	$(\sigma,\mu)=(-0.99,1,01)$	$(\sigma,\mu)=(0.25,2.25)$		Rate		
10	1.86×10^{-2}	1.59×10^{-2}	3.36×10^{-5}				
20	5.23×10^{-3}	3.61×10^{-3}	3.45×10^{-6}		1.8	2.1	3.3
40	1.38×10^{-3}	6.20×10^{-4}	2.58×10^{-7}		1.9	2.5	3.7
80	3.54×10^{-4}	8.37×10^{-5}	1.74×10^{-8}		1.9	2.9	3.9

Table A.4: Evolution error $e_{L_2,evol}$ with $v=1$ of three different (σ,μ)-schemes at $t=\infty$.

Lastly, we look at the final projection error in Table A.5. Interesting enough, the final projection error appears to be the sum of both the initial projection error and the evolution error. The bad evolution error of the $(-1,1)$-scheme caused the convergence of the final projection error to degrade down to 1st-order. The $(-0.99,1.01)$-scheme has a slightly larger final projection error than the initial projection error, but has the same order. The most surprising result is that of the $(0.25,2.25)$-scheme; it has the same initial projection error as the final projection error. This means the evolution error is so small that it does not contribute anything towards the final error!

In this experiment we have shown that not all DG schemes are equal. All schemes commit the same initial projection error, and it is the evolution error that differenti-

N_{ele}	$(\sigma,\mu)=(-1,1)$	$(\sigma,\mu)=(-0.99,1,01)$	$(\sigma,\mu)=(0.25,2.25)$		Rate		
10	1.41×10^{-1}	8.68×10^{-2}	1.04×10^{-2}				
20	7.32×10^{-2}	2.07×10^{-2}	2.60×10^{-3}		0.9	2.0	2.0
40	3.69×10^{-2}	4.00×10^{-3}	6.50×10^{-4}		0.9	2.3	1.9
80	1.85×10^{-2}	6.80×10^{-4}	1.60×10^{-4}		1.0	2.5	1.9

Table A.5: Final projection error $e_{L_2,proj}$ of three different (σ,μ)-schemes at $t=\infty$.

ates the schemes. It is imperative to design a numerical scheme where the convergence rate of the evolution error must be as least $p+1$. Yet once again we have shown RDG-1x$\bar{f}$ to be superior.

A.3 Derivatives of primitive variable

In Chapter 5 we often need to express the derivatives of primitive variables in terms of the conservative variables. Here we consider a simple example with conservative variables ρ and $m=\rho u$ spanned by the Legendre basis in terms of ξ up to any polynomial order:

$$\rho(\xi) = \overline{\rho}+\overline{\Delta\rho}(2\xi+1)+\overline{\Delta^2\rho}\left(6\xi^2-6\xi+1\right)+\ldots, \tag{A.11}$$

$$m(\xi) = \overline{m}+\overline{\Delta m}(2\xi+1)+\overline{\Delta^2 m}\left(6\xi^2-6\xi+1\right)+\ldots, \tag{A.12}$$

and there derivatives are

$$\frac{\partial\rho}{\partial\xi}(\xi) = \overline{\Delta\rho}(2)+\overline{\Delta^2\rho}(12\xi-6)+\ldots, \tag{A.13}$$

$$\frac{\partial m}{\partial\xi}(\xi) = \overline{\Delta m}(2)+\overline{\Delta^2 m}(12\xi-6)+\ldots. \tag{A.14}$$

We wish to obtain the derivative of u in terms of ξ. We first express u in terms of conservative variables,

$$u=\frac{m}{\rho}, \tag{A.15}$$

and the derivative of u becomes

$$u_\xi = m\frac{\partial\rho^{-1}}{\partial\xi}+\frac{1}{\rho}\frac{\partial m}{\partial\xi}. \tag{A.16}$$

The term $\frac{\partial \rho^{-1}}{\partial \xi}$ requires a substitution trick. Let $q = \frac{1}{\rho}$, and then we use the chain rule to obtain

$$\frac{\partial q}{\partial \xi} = \frac{\partial q}{\partial \rho}\frac{\partial \rho}{\partial \xi}, \tag{A.17}$$

where $\frac{\partial q}{\partial \rho} = -\frac{1}{\rho^2}$. This is an effective technique to obtain the derivative at point values which are needed for Gaussian integration.

A.4 Implicit Radau Integration

The Radau points listed in Appendix A.1 are for explicit Radau integration. Explicit means we acquire the numerical integral over the whole time interval $\tau \in [0\ 1]$ by a correct linear combination of weights and function values at selected points. However, in HH-DG we frequently need the integrate over a different interval, i.e., $\tau \in [0\ \tau_k]$, where τ_k is any Radau point less than unity. Naively one can normalized another set of Radau points to fit into $\tau \in [0\ \tau_k]$, resulting in more flux evaluations in the new sub-interval. This trick, donated by Dr. Huynh, circumvents this problem is to fit a smooth function through the original Radau points over the interval $\tau \in [0\ 1]$. We then integrate the fitted function over the sub-intervals $\tau \in [0\ \tau_k]$ to acquire the proper function weights based on the function values at the original Radau points. This is important because we reuse the information we already have and avoid extra flux evaluations, making this an efficient numerical technique.

Consider an arbitrary function to be numerical evaluated,

$$\int_0^{\tau_k} f(\tau)\, d\tau. \tag{A.18}$$

If we have n Radau points, we can reconstruct a polynomial of degree $n-1$, $\hat{f}(\tau) = a_0 + a_1\tau + a_2\tau^2 + \cdots + a_{n-1}\tau^{n-1}$, on the interval $\tau = [0\ 1]$, by enforcing the following conditions,

$$\hat{f}(\tau_k) = f(\tau_k), \quad k = 1, 2, \ldots, n-1. \tag{A.19}$$

Solving for the equations above will express the coefficients a_k in terms of $f(\tau_k)$. Consequently, the integration of Eqn A.18 for various values of τ_k is also expressed in terms of the original $f(\tau_k)$. Table A.6 shows the Radau points on the left and the volume integral table on the right for $n = 2$ and 3. Let us take $n = 2$ for example: the numerical value of $\int_0^{\tau_1} f\, d\tau$ is simply $\frac{1}{4}f(\tau_1) + \frac{3}{4}f(\tau_2)$.

$n=2$			$f(\tau_1)$	$f(\tau_2)$
τ_1	1	$\int_0^{\tau_1} f\,d\tau$	$\frac{1}{4}$	$\frac{3}{4}$
τ_2	$\frac{1}{3}$	$\int_0^{\tau_2} f\,d\tau$	$-\frac{1}{12}$	$\frac{5}{12}$

$n=3$			$f(\tau_1)$	$f(\tau_2)$	$f(\tau_3)$
τ_1	1	$\int_0^{\tau_1} f\,d\tau$	0.11111	0.51248	0.37640
τ_2	0.64494	$\int_0^{\tau_2} f\,d\tau$	-0.04154	0.29207	0.39442
τ_3	0.15505	$\int_0^{\tau_3} f\,d\tau$	0.02377	-0.06553	0.19681

Table A.6: Implicit Radau integration weights for $n = 2$ and 3.

APPENDIX B

Graveyard of Numbers

The arduous computer performs an amazing deed,
churning out myriads of numbers at dazzling speed.
Here lies an endless sea of numerical table,
Pages of pages one's mind couldn't fable.

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	Time(s)	Iteration
			$(\sigma, \mu, \text{VNN}) = \left(\frac{1}{4},\ \frac{9}{4},\ 0.08\right)$			
10	$3.36E-05$		$5.72E-05$		0.02	515
20	$3.45E-06$	3.28	$2.09E-06$	4.77	0.11	2034
40	$2.58E-07$	3.74	$6.95E-08$	4.91	0.69	6924
80	$1.74E-08$	3.89	$2.23E-09$	4.96	3.97E+00	20800
160	$1.13E-09$	3.94	$7.06E-11$	4.98	1.93E+01	51700
320	$6.45E-11$	4.13	$2.15E-12$	5.04	55.16	74762
			$(\sigma, \mu, \text{VNN}) = \left(\frac{1}{9},\ \frac{19}{9},\ 0.09\right)$			
10	$2.88E-04$		$5.97E-04$		0.01	499
20	$3.73E-05$	2.95	$8.27E-05$	2.85	0.11	2030
40	$4.73E-06$	2.98	$1.09E-05$	2.92	0.73	7320
80	$5.94E-07$	2.99	$1.39E-06$	2.97	4.66	24355
160	$7.45E-08$	3.00	$1.76E-07$	2.98	27.9	74637
320	$9.32E-09$	3.00	$2.22E-08$	2.99	149.07	202466
			$(\sigma, \mu, \text{VNN}) = (-1,\ 1,\ 0.20)$			
10	0.0186		0.242		0.01	317
20	0.00523	1.83	0.126	0.94	0.07	1245
40	0.00138	1.92	0.0639	0.98	0.46	4673
80	$3.54E-04$	1.96	0.032	1.00	3.27	16951
160	$8.97E-05$	1.98	$1.60E-02$	1.00	22.5	59731
320	$2.26E-05$	1.99	$8.02E-03$	1.00	152.05	204592
			$(\sigma, \mu, \text{VNN}) = (1,\ 0,\ 0.20)$			
10	$2.47E-01$		0.0228		0.16	4894
20	$6.20E-02$	1.99	$2.85E-03$	3.00	1.05	19479
40	$1.56E-02$	1.99	$3.56E-04$	3.00	7.36	73271
80	0.0039	2.00	$4.45E-05$	3.00	51.27	265085
160	0.000977	2.00	$5.56E-06$	3.00	353.6	931621
320	0.000244	2.00	$6.95E-07$	3.00	2372.97	3176655
			$(\sigma, \mu, \text{VNN}) = \left(-\frac{1}{2},\ \frac{3}{2},\ 0.19\right)$			
10	0.00283		0.00598		0.01	277
20	0.000375	2.92	0.000932	2.68	0.07	1126
40	$4.67E-05$	3.01	0.000128	2.86	0.43	4126
80	$5.79E-06$	3.01	$1.66E-05$	2.95	2.84	14212
160	$7.19E-07$	3.01	$2.11E-06$	2.98	18.07	46119
320	$8.95E-08$	3.01	$2.66E-07$	2.99	107.72	138965

Table B.1: Linear diffusion: L_2-error of various $\sigma - \mu$ schemes ($p = 1$).

N_{ele}	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	Time(s)	Iteration
			$(\sigma, \mu, \text{VNN}) = (0,\ 2,\ 0.10)$			
10	0.000582		0.00113		0.02	469
20	$7.52E-05$	2.95	0.000163	2.79	0.1	1904
40	$9.40E-06$	3.00	$2.16E-05$	2.92	0.69	6894
80	$1.17E-06$	3.01	$2.78E-06$	2.96	4.43	23187
160	$1.46E-07$	3.00	$3.52E-07$	2.98	27.03	72322
320	$1.82E-08$	3.00	$4.43E-08$	2.99	149.84	202302
			$(\sigma, \mu, \text{VNN}) = (0.0893,\ 2.5392,\ 0.08)$			
10	0.00402		0.00143		0.02	606
20	0.0013	1.63	0.000218	2.71	0.15	2640
40	0.000361	1.85	$2.93E-05$	2.90	1.03	10276
80	$9.46E-05$	1.93	$3.77E-06$	2.96	7.19	37473
160	$2.42E-05$	1.97	$4.78E-07$	2.98	49.14	130970
320	$6.10E-06$	1.99	$6.01E-08$	2.99	327.31	440358
			$(\sigma, \mu, \text{VNN}) = \left(\frac{1}{6},\ \frac{13}{6},\ 0.08\right)$			
10	0.000157		0.000367		0.02	540
20	$2.05E-05$	2.94	$4.82E-05$	2.93	0.11	2198
40	$2.65E-06$	2.95	$6.25E-06$	2.95	0.79	7875
80	$3.38E-07$	2.97	$7.97E-07$	2.97	4.98	26005
160	$4.27E-08$	2.98	$1.01E-07$	2.98	29.31	78417
320	$5.36E-09$	2.99	$1.27E-08$	2.99	151.56	205253
			$(\sigma, \mu, \text{VNN}) = (-1,\ 3,\ 0.10)$			
10	0.0126		0.00552		0.01	475
20	0.00452	1.48	0.00092	2.58	0.12	2199
40	0.0013	1.80	0.000127	2.86	0.88	8743
80	0.000345	1.91	$1.66E-05$	2.94	6.23	32344
160	$8.86E-05$	1.96	$2.11E-06$	2.98	43.46	115168
320	$2.24E-05$	1.98	$2.66E-07$	2.99	294.62	396262
			$(\sigma, \mu, \text{VNN}) = (0,\ 3,\ 0.06)$			
10	0.00535		0.00107		0.02	795
20	0.00165	1.70	0.000161	2.73	0.19	3491
40	0.000449	1.88	$2.16E-05$	2.90	1.36	13607
80	0.000117	1.94	$2.78E-06$	2.96	9.52	49637
160	$2.98E-05$	1.97	$3.52E-07$	2.98	64.95	173359
320	$7.51E-06$	1.99	$4.43E-08$	2.99	431.54	582031

Table B.2: Linear diffusion: L_2-error of various $\sigma - \mu$ schemes ($p = 1$).

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	10	6.43×10^{-1}	2.03×10^{-1}						
	20	1.93×10^{-1}	3.06×10^{-2}			1.7	2.7		
	40	2.79×10^{-2}	2.59×10^{-3}			2.8	3.5		
	80	3.55×10^{-3}	3.87×10^{-4}			2.9	2.7		
	160	4.45×10^{-4}	9.12×10^{-5}			3.0	2.0		
	320	5.57×10^{-5}	2.27×10^{-5}			3.0	2.0		
2	4	3.38×10^{-1}	2.77×10^{-1}	7.51×10^{-2}					
	8	1.77×10^{-2}	7.07×10^{-3}	3.10×10^{-3}		4.7	4.4	2.9	
	16	6.20×10^{-4}	1.28×10^{-4}	3.60×10^{-4}		4.9	3.9	2.9	
	32	2.36×10^{-5}	2.96×10^{-6}	4.46×10^{-5}		4.8	3.9	3.0	
	64	1.11×10^{-6}	1.06×10^{-7}	5.58×10^{-6}		4.4	4.0	3.0	
	128	6.26×10^{-8}	4.91×10^{-9}	6.97×10^{-7}		4.0	4.0	3.0	
3	4	6.16×10^{-3}	5.05×10^{-3}	1.37×10^{-4}	2.50×10^{-3}				
	8	5.63×10^{-5}	2.24×10^{-5}	9.48×10^{-6}	1.59×10^{-4}	6.7	7.8	7.2	3.9
	12	3.45×10^{-6}	9.08×10^{-7}	1.19×10^{-6}	3.16×10^{-5}	6.9	7.9	5.1	4.0
	16	4.77×10^{-7}	9.44×10^{-8}	2.81×10^{-7}	1.00×10^{-5}	6.8	7.8	5.0	4.0
	20	1.04×10^{-7}	1.66×10^{-8}	9.21×10^{-8}	4.10×10^{-6}	6.8	7.8	5.0	4.0
	24	3.02×10^{-8}	4.06×10^{-9}	3.70×10^{-8}	1.98×10^{-5}	6.8	7.7	5.0	4.0

Table B.3: L_2-error of RK-Upwind-DG scheme for time-accurate problem with periodic boundary condition. A sine wave is advected to the right for 100 cycles.

$P_{e,G}$	N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10^{-4}	10	20	1.03×10^{-2}		4.90×10^{-3}	
	20	40	6.88×10^{-4}	3.9	1.67×10^{-4}	4.9
	40	80	4.37×10^{-5}	4.0	5.07×10^{-6}	5.0
	80	160	2.75×10^{-6}	4.0	1.59×10^{-7}	5.0
	160	320	1.72×10^{-7}	4.0	5.04×10^{-9}	5.0
	320	640	1.07×10^{-8}	4.0	1.87×10^{-10}	4.8
10^{-3}	10	20	1.03×10^{-2}		4.91×10^{-3}	
	20	40	6.88×10^{-4}	3.9	1.68×10^{-4}	4.9
	40	80	4.37×10^{-5}	4.0	5.12×10^{-6}	5.0
	80	160	2.75×10^{-6}	4.0	1.74×10^{-7}	4.9
	160	320	1.72×10^{-7}	4.0	9.78×10^{-9}	4.2
	320	640	1.07×10^{-8}	4.0	1.06×10^{-9}	3.2
10^{-2}	10	20	1.03×10^{-2}		4.99×10^{-3}	
	20	40	6.88×10^{-4}	3.9	1.80×10^{-4}	4.8
	40	80	4.37×10^{-5}	4.0	7.58×10^{-6}	4.6
	80	160	2.75×10^{-6}	4.0	6.91×10^{-7}	3.5
	160	320	1.72×10^{-7}	4.0	8.38×10^{-8}	3.0
	320	640	1.07×10^{-8}	4.0	1.04×10^{-8}	3.0
10^{-1}	10	20	1.02×10^{-2}		6.43×10^{-3}	
	20	40	6.87×10^{-4}	3.9	5.66×10^{-4}	3.5
	40	80	4.37×10^{-5}	4.0	5.38×10^{-5}	3.4
	80	160	2.75×10^{-6}	4.0	6.68×10^{-6}	3.0
	160	320	1.72×10^{-7}	4.0	8.35×10^{-7}	3.0
	320	640	1.07×10^{-8}	4.0	1.04×10^{-7}	3.0
10^{0}	10	20	1.09×10^{-2}		2.49×10^{-2}	
	20	40	7.40×10^{-4}	3.9	4.18×10^{-3}	2.6
	40	80	4.84×10^{-5}	3.9	5.28×10^{-4}	3.0
	80	160	3.09×10^{-6}	4.0	6.65×10^{-5}	3.0
	160	320	1.95×10^{-7}	4.0	8.33×10^{-6}	3.0
	320	640	1.22×10^{-8}	4.0	1.04×10^{-6}	3.0
10^{1}	10	20	5.52×10^{-2}		2.31×10^{-1}	
	20	40	4.26×10^{-3}	3.7	3.52×10^{-2}	2.7
	40	80	2.97×10^{-4}	3.8	4.85×10^{-3}	2.9
	80	160	1.96×10^{-5}	3.9	6.63×10^{-4}	2.9
	160	320	1.26×10^{-6}	4.0	8.15×10^{-5}	3.0
	320	640	8.00×10^{-8}	4.0	1.03×10^{-5}	3.0

Table B.4: Linear advection-diffusion: RK-Upwind-RDG-2x, $p = 1$, $r = \frac{1}{6}$, CFL $= 0.4$, $\mu = 0.01$, $t_{final} = 100$, periodic boundary condition.

$P_{e,G}$	N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate
10^2	10	20	1.22		0.717	
	20	40	0.128	3.3	0.141	2.4
	40	80	1.17×10^{-2}	3.5	2.67×10^{-2}	2.4
	80	160	9.53×10^{-4}	3.6	4.45×10^{-3}	2.6
	160	320	7.06×10^{-5}	3.8	6.68×10^{-4}	2.7
	320	640	4.88×10^{-6}	3.9	9.29×10^{-5}	2.9
10^3	10	20	11.9		3.75	
	20	40	1.9	2.7	0.351	3.4
	40	80	2.31×10^{-1}	3.0	5.13×10^{-2}	2.8
	80	160	2.63×10^{-2}	3.2	1.12×10^{-2}	2.2
	160	320	2.74×10^{-3}	3.3	2.39×10^{-3}	2.3
	320	640	2.57×10^{-4}	3.4	4.64×10^{-4}	2.4
10^4	10	20	25.6		8.08	
	20	40	14.4	0.8	2.26	1.8
	40	80	2.49	2.5	0.201	3.5
	80	160	0.321	3.0	0.0181	3.5
	160	320	3.94×10^{-2}	3.0	3.30×10^{-3}	2.5
	320	640	4.70×10^{-3}	3.0	7.75×10^{-4}	2.1

Table B.5: Linear advection-diffusion: RK-Upwind-RDG-2x, $p = 1$, $r = \frac{1}{6}$, CFL $= 0.4$, $\mu = 0.01$, $t_{final} = 100$, periodic boundary condition.

$P_{e,G}$	N_{ele}	DOF	$L_2\,\bar{u}_{error}$	Rate	$L_2\,\overline{\triangle u}_{error}$	Rate	$L_2\,\overline{\triangle^2 u}_{error}$	Rate
10^{-4}	10	30	2.08×10^{-7}		7.37×10^{-6}		5.96×10^{-5}	
	20	60	9.69×10^{-10}	7.8	4.98×10^{-8}	7.2	9.43×10^{-7}	6.0
	40	120	3.83×10^{-12}	8.0	5.86×10^{-11}	9.7	1.48×10^{-8}	6.0
	80	240	2.89×10^{-12}	0.4	2.06×10^{-11}	1.5	2.32×10^{-10}	6.0
	160	480	3.69×10^{-13}	3.0	1.47×10^{-12}	3.8	3.69×10^{-12}	6.0
	320	960	4.59×10^{-11}	-7.0	5.41×10^{-13}	1.4	8.39×10^{-14}	5.5
10^{-3}	10	30	2.07×10^{-7}		6.56×10^{-6}		6.01×10^{-5}	
	20	60	9.78×10^{-10}	7.7	9.25×10^{-9}	9.5	9.59×10^{-7}	6.0
	40	120	3.20×10^{-12}	8.3	3.38×10^{-9}	1.5	1.53×10^{-8}	6.0
	80	240	1.93×10^{-13}	4.0	2.35×10^{-10}	3.9	2.48×10^{-10}	6.0
	160	480	2.28×10^{-11}	-6.9	1.53×10^{-11}	3.9	4.18×10^{-12}	5.9
	320	960	9.09×10^{-11}	-2.0	1.82×10^{-12}	3.1	1.02×10^{-13}	5.4

Table B.6: Linear advection-diffusion: RK-Upwind-RDG-2x, $p = 2$, $r = \frac{1}{10}$, CFL $= 0.27$, $\mu = 0.01$, $t_{final} = 100$, periodic boundary condition.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	6	5.51×10^{-3}	3.19×10^{-3}						
	12	1.35×10^{-3}	4.26×10^{-4}			2.0	2.9		
	18	5.99×10^{-4}	1.29×10^{-4}			2.0	2.9		
	24	3.37×10^{-4}	5.52×10^{-5}			2.0	2.9		
	30	2.15×10^{-4}	2.85×10^{-5}			2.0	2.9		
	36	1.50×10^{-4}	1.66×10^{-5}			1.9	2.9		
2	4	1.45×10^{-3}	1.38×10^{-3}	1.77×10^{-3}					
	8	5.51×10^{-4}	2.08×10^{-4}	1.76×10^{-4}		1.4	2.7	3.3	
	12	2.69×10^{-4}	6.43×10^{-5}	4.38×10^{-5}		1.7	2.9	3.4	
	16	1.57×10^{-4}	2.76×10^{-5}	1.62×10^{-5}		1.8	2.9	3.4	
	20	1.02×10^{-4}	1.42×10^{-5}	7.50×10^{-6}		1.9	2.9	3.4	
	24	7.19×10^{-5}	8.27×10^{-6}	3.98×10^{-6}		1.9	2.9	3.4	
3	4	5.64×10^{-5}	1.53×10^{-5}	1.43×10^{-4}	5.13×10^{-5}				
	6	1.24×10^{-5}	1.75×10^{-6}	3.02×10^{-5}	6.17×10^{-6}	3.7	5.3	3.8	5.2
	8	4.07×10^{-6}	3.68×10^{-7}	9.75×10^{-6}	1.38×10^{-6}	3.8	5.4	3.9	5.2
	10	1.69×10^{-6}	1.09×10^{-7}	4.03×10^{-6}	4.31×10^{-7}	3.9	5.4	3.9	5.2
	12	8.25×10^{-7}	4.02×10^{-8}	1.95×10^{-6}	1.68×10^{-7}	3.9	5.4	3.9	5.1
	14	4.48×10^{-7}	1.73×10^{-8}	1.06×10^{-6}	7.56×10^{-8}	3.9	5.4	3.9	5.1

Table B.7: RDG-1x-Naive, VNN = 0.07, 0.02, and 0.01 for RK3, RK4, and RK5, respectively: L_2-error of steady linear-variation diffusion problem with two-sided Neumann boundary conditions.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	6	1.38×10^{-4}	7.51×10^{-5}						
	12	9.09×10^{-6}	2.12×10^{-6}			3.9	5.1		
	18	1.81×10^{-6}	2.66×10^{-7}			3.9	5.1		
	24	5.75×10^{-7}	6.13×10^{-8}			3.9	5.1		
	30	2.36×10^{-7}	1.97×10^{-8}			3.9	5.0		
	36	1.14×10^{-7}	7.83×10^{-9}			3.9	5.0		
2	4	1.93×10^{-6}	3.26×10^{-7}	5.38×10^{-6}					
	8	5.14×10^{-8}	1.60×10^{-8}	1.12×10^{-7}		5.2	4.3	5.9	
	12	5.79×10^{-9}	1.94×10^{-9}	1.18×10^{-8}		5.3	5.2	5.5	
	16	1.21×10^{-9}	4.17×10^{-10}	2.42×10^{-9}		5.4	5.3	5.5	
	20	3.59×10^{-10}	1.25×10^{-10}	7.05×10^{-10}		5.4	5.4	5.3	
	24	1.32×10^{-10}	4.62×10^{-11}	2.58×10^{-10}		5.4	5.4	5.5	
3	4	6.16×10^{-8}	1.92×10^{-7}	4.23×10^{-8}	4.11×10^{-7}				
	6	3.24×10^{-9}	1.07×10^{-8}	2.51×10^{-9}	2.66×10^{-8}	3.7	5.3	3.8	5.2
	8	3.89×10^{-10}	1.35×10^{-9}	3.14×10^{-10}	3.72×10^{-9}	3.8	5.4	3.9	5.2
	10	7.42×10^{-11}	2.71×10^{-10}	6.10×10^{-11}	8.02×10^{-10}	3.9	5.4	3.9	5.2
	12	1.90×10^{-11}	7.30×10^{-11}	1.58×10^{-11}	2.28×10^{-10}	3.9	5.4	3.9	5.1
	14	5.98×10^{-12}	2.41×10^{-11}	5.04×10^{-12}	7.83×10^{-11}	3.9	5.4	3.9	5.1

Table B.8: RDG-2x, VNN = 0.15, 0.08, and 0.04 for RK3, RK4, and RK5, respectively: L_2-error of steady linear-variation diffusion problem with two-sided Neumann boundary conditions.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	6	7.59×10^{-5}	3.26×10^{-5}						
	12	6.21×10^{-6}	1.10×10^{-6}			3.6	4.8		
	18	1.30×10^{-6}	1.43×10^{-7}			3.8	5.0		
	24	4.20×10^{-7}	3.35×10^{-8}			3.9	5.0		
	30	1.74×10^{-7}	1.08×10^{-8}			3.9	5.0		
	36	8.43×10^{-8}	4.32×10^{-9}			3.9	5.0		
2	4	3.47×10^{-6}	2.14×10^{-6}	6.87×10^{-6}					
	8	1.08×10^{-7}	2.48×10^{-8}	1.28×10^{-7}		5.0	6.4	5.7	
	12	1.10×10^{-8}	1.59×10^{-9}	1.16×10^{-8}		5.6	6.7	5.9	
	16	2.07×10^{-9}	2.19×10^{-10}	2.10×10^{-9}		5.8	6.8	5.9	
	20	5.58×10^{-10}	4.66×10^{-11}	5.54×10^{-10}		5.8	6.9	5.9	
	24	1.88×10^{-10}	1.31×10^{-11}	1.86×10^{-10}		5.9	6.9	5.9	
3	4	5.32×10^{-8}	2.99×10^{-9}	1.10×10^{-7}	1.33×10^{-8}				
	6			Diverged					
	8			Diverged					
	10			Diverged					
	12	6.05×10^{-14}	4.30×10^{-13}	1.10×10^{-11}	8.56×10^{-13}	12.4	8.0	8.3	8.7
	14	1.16×10^{-14}	1.08×10^{-13}	3.22×10^{-12}	2.14×10^{-13}	10.9	8.9	7.9	9.4

Table B.9: RDG-1x$\bar{f}$, VNN = 0.08, 0.02, and 0.0001 for RK3, RK4, and RK5, respectively: L_2-error of steady linear-variation diffusion problem with two-sided Neumann boundary conditions.

p	N_{ele}	$L_2\,\bar{u}_{error}$	$L_2\,\overline{\triangle u}_{error}$	$L_2\,\overline{\triangle^2 u}_{error}$	$L_2\,\overline{\triangle^3 u}_{error}$	Rate			
1	6	1.38×10^{-4}	7.51×10^{-5}						
	12	9.09×10^{-6}	2.12×10^{-6}			3.9	5.1		
	18	1.81×10^{-6}	2.66×10^{-7}			3.9	5.1		
	24	5.75×10^{-7}	6.13×10^{-8}			3.9	5.1		
	30	2.36×10^{-7}	1.97×10^{-8}			3.9	5.0		
	36	1.14×10^{-7}	7.83×10^{-9}			3.9	5.0		
2	4	1.49×10^{-7}	6.97×10^{-7}	6.27×10^{-6}					
	8	1.11×10^{-9}	6.37×10^{-9}	9.77×10^{-8}		7.0	6.7	6.0	
	12	5.72×10^{-11}	3.89×10^{-10}	8.59×10^{-9}		7.3	6.9	6.0	
	16	6.83×10^{-12}	5.30×10^{-11}	1.53×10^{-9}		7.3	6.9	6.0	
	20	1.27×10^{-12}	1.12×10^{-11}	4.01×10^{-10}		7.5	6.9	6.0	
	24	2.87×10^{-13}	3.15×10^{-12}	1.34×10^{-10}		8.1	6.9	6.0	
3	4	4.85×10^{-10}	1.43×10^{-9}	6.81×10^{-9}	1.24×10^{-8}				
	6	1.27×10^{-11}	4.25×10^{-11}	2.95×10^{-10}	3.48×10^{-10}	8.9	8.6	7.4	8.8
	8	8.70×10^{-13}	3.25×10^{-12}	3.06×10^{-11}	2.69×10^{-11}	9.3	8.9	7.8	8.9
	10	1.01×10^{-13}	4.32×10^{-13}	5.21×10^{-12}	3.66×10^{-12}	9.6	9.0	7.9	8.9
	12	1.66×10^{-14}	8.27×10^{-14}	1.22×10^{-12}	7.15×10^{-13}	9.9	9.0	7.9	8.9
	14	3.31×10^{-15}	2.04×10^{-14}	3.58×10^{-13}	1.79×10^{-13}	10.4	9.0	7.9	8.9

Table B.10: RDG-1x+, VNN = 0.15, 0.08, and 0.04 for RK3, RK4, and RK5, respectively: L_2-error of steady linear-variation diffusion problem with two-sided Neumann boundary conditions.

$P_{e,G}$	N_{ele}	DOF	$L_2\,\bar{u}_{error}$	OOA	$L_2\,\overline{\triangle u}_{error}$	OOA
10^{-5}	10	20	6.59×10^{-5}		3.68×10^{-5}	
	20	40	4.13×10^{-6}	4.0	1.17×10^{-6}	4.9
	40	80	2.59×10^{-7}	4.0	3.66×10^{-8}	5.0
	80	160	1.62×10^{-8}	4.0	1.15×10^{-9}	5.0
	160	320	1.01×10^{-9}	4.0	4.11×10^{-11}	4.8
	320	640	6.34×10^{-11}	4.0	1.81×10^{-12}	4.5
10^{-4}	10	20	6.59×10^{-5}		3.68×10^{-5}	
	20	40	4.13×10^{-6}	4.0	1.18×10^{-6}	5.0
	40	80	2.59×10^{-7}	4.0	3.77×10^{-8}	5.0
	80	160	1.62×10^{-8}	4.0	1.47×10^{-9}	4.7
	160	320	1.01×10^{-9}	4.0	2.02×10^{-10}	2.9
	320	640	6.33×10^{-11}	4.0	1.42×10^{-11}	3.8
10^{-3}	10	20	6.58×10^{-5}		3.81×10^{-5}	
	20	40	4.13×10^{-6}	4.0	1.54×10^{-6}	4.6
	40	80	2.59×10^{-7}	4.0	8.62×10^{-8}	4.2
	80	160	1.62×10^{-8}	4.0	9.10×10^{-9}	3.2
	160	320	1.01×10^{-9}	4.0	1.97×10^{-9}	2.2
	320	640	6.33×10^{-11}	4.0	1.41×10^{-10}	3.8
10^{-2}	10	20	6.53×10^{-5}		7.35×10^{-5}	
	20	40	4.13×10^{-6}	4.0	9.45×10^{-6}	3.0
	40	80	2.59×10^{-7}	4.0	7.60×10^{-7}	3.6
	80	160	1.62×10^{-8}	4.0	9.08×10^{-8}	3.0
	160	320	1.01×10^{-9}	4.0	1.80×10^{-8}	2.3
	320	640	6.35×10^{-11}	4.0	1.42×10^{-9}	3.7
10^{-1}	10	20	8.43×10^{-5}		3.54×10^{-4}	
	20	40	6.74×10^{-6}	3.6	9.98×10^{-5}	2.4
	40	80	3.24×10^{-7}	4.4	7.01×10^{-6}	3.8
	80	160	2.30×10^{-8}	3.8	1.17×10^{-6}	2.6
	160	320	1.29×10^{-9}	4.2	1.13×10^{-7}	3.4
	320	640	8.57×10^{-11}	3.9	1.62×10^{-8}	2.8
10^{0}	10	20	2.94×10^{-4}		2.34×10^{-3}	
	20	40	2.53×10^{-5}	3.6	4.19×10^{-4}	2.5
	40	80	1.78×10^{-6}	3.8	5.88×10^{-5}	2.8
	80	160	1.42×10^{-7}	3.7	9.57×10^{-6}	2.6
	160	320	8.20×10^{-9}	4.1	1.08×10^{-6}	3.2
	320	640	8.72×10^{-10}	3.2	2.46×10^{-7}	2.1

Table B.11: Linear advection-diffusion: HH-DG $p = 1$, $r = \frac{1}{6}$, CFL $= 1$, $\mu = 0.01$, $t_{final} = 100$, $C_{\text{Adv-Diff}} = 0.9375$, periodic boundary condition.

$P_{e,G}$	N_{ele}	DOF	$L_2\,\bar{u}_{error}$	OOA	$L_2\,\overline{\triangle u}_{error}$	OOA
10^{+1}	10	20	1.48×10^{-3}		7.69×10^{-3}	
	20	40	1.70×10^{-4}	3.1	1.34×10^{-3}	2.5
	40	80	1.98×10^{-5}	3.1	3.25×10^{-4}	2.0
	80	160	2.01×10^{-6}	3.3	6.92×10^{-5}	2.2
	160	320	1.64×10^{-7}	3.6	8.02×10^{-6}	3.1
	320	640	1.40×10^{-8}	3.5	2.11×10^{-6}	1.9
10^{+2}	10	20	6.57×10^{-3}		5.57×10^{-3}	
	20	40	9.77×10^{-4}	2.8	1.51×10^{-3}	1.9
	40	80	1.49×10^{-4}	2.7	2.91×10^{-4}	2.4
	80	160	2.21×10^{-5}	2.8	8.07×10^{-5}	1.9
	160	320	3.00×10^{-6}	2.9	1.72×10^{-5}	2.2
	320	640	3.60×10^{-7}	3.1	7.01×10^{-6}	1.3
10^{+3}	10	20	2.27×10^{-2}		7.48×10^{-3}	
	20	40	4.35×10^{-3}	2.4	1.34×10^{-3}	2.5
	40	80	7.53×10^{-4}	2.5	3.05×10^{-4}	2.1
	80	160	1.19×10^{-4}	2.7	8.55×10^{-5}	1.8
	160	320	1.79×10^{-5}	2.7	2.52×10^{-5}	1.8
	320	640	2.71×10^{-6}	2.7	4.39×10^{-6}	2.5
10^{+4}	10	20	8.18×10^{-3}		2.50×10^{-2}	
	20	40	1.48×10^{-3}	2.5	2.43×10^{-3}	3.4
	40	80	2.46×10^{-3}	2.6	3.93×10^{-4}	2.6
	80	160	4.29×10^{-4}	2.5	2.25×10^{-5}	4.3
	160	320	7.68×10^{-5}	2.5	1.44×10^{-5}	0.6
	320	640	1.34×10^{-5}	2.5	5.01×10^{-6}	1.5
10^{+5}	10	20	2.46×10^{-1}		7.79×10^{-2}	
	20	40	8.89×10^{-2}	1.5	1.41×10^{-2}	2.5
	40	80	1.40×10^{-2}	2.7	1.14×10^{-3}	3.6
	80	160	1.95×10^{-3}	2.8	1.20×10^{-4}	3.3
	160	320	2.83×10^{-4}	2.8	1.78×10^{-5}	2.8
	320	640	0.00×10^{-11}	0.0	0.00×10^{-8}	0.0

Table B.12: Linear advection-diffusion: HH-DG $p = 1$, $r = \frac{1}{6}$, CFL $= 1$, $\mu = 0.01$, $t_{final} = 100$, $C_{\text{Adv-Diff}} = 0.9375$, periodic boundary condition.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Saint Petersburg State Polytechnical University. `http://smitu.cef.spbstu.ru/galerkin.htm`.

[2] D.N. Arnold. "An interior penalty finite element method with discontinuous elements". *SIAM Journal on Numerical Analysis*, 19:742–760, 1982.

[3] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. "Unified analysis of discontinuous galerkin methods for elliptic problems". *SIAM Journal on Numerical Analysis*, 39:1749–1779, 2002.

[4] F. Bassi and S. Rebay. "A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier-stokes equations". *Journal of Computational Physics*, 131:267–279, 1997.

[5] C.E. Baumann and J.T. Oden. "A discontinuous hp finite element method for convection-diffusion problems". *Computer Methods in Applied Mechanics and Engineering*, 175:311–341, 1999.

[6] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. "Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems". *Siam, Journal on Numerical Analysis*, 47:1319–1365, 2009.

[7] B. Cockburn and C.W. Shu. "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework". *Mathematics of Computation*, 52:411–435, 1989.

[8] B. Cockburn and C.W. Shu. "The local discontinuous Galerkin method for time-dependent convection-diffusion systems". *Siam, Journal on Numerical Analysis*, 35:2440–2463, 1998.

[9] L.M. Delves and C.A. Hall. "An implicit matching principle for global element calculations". *Institute of Mathematics and its Applications*, 23:223–234, 1979.

[10] M. Dumbser and C-D. Munz. "ADER discontinuous Galerkin schemes for aeroacoustics". *C.R. Mecanique 333*, pages 683–687, 2005.

[11] B. Finlayson. *"The method of weighted residuals and variational principle, with application in fluid mechanics"*. Academic Press, 1972.

[12] D. French, M.C. Galbraith, and M. Osorio. "Error analysis of a modified discontinuous Galerkin recovery scheme for diffusion problems". *48th AIAA Aerospace Siciences Meeting including the New Horizons Forum and Aerospace Exposition, AIAA Paper 2010-1071*, 2010.

[13] J.E. Fromm. "A method for reducing dispersion in convective difference schemes". *Journal of Computational Physics*, 3:176–189, 1968.

[14] G. Gassner, F. Lorcher, and C.D. Munz. "A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes". *Journal of Scientific Computational Physics*, 224:1049–1063, 2007.

[15] G. Gassner, F. Lorcher, and C.D. Munz. "A discontinuous Galkerin scheme based on a space-time expansion. II. viscous flow equations in multi dimensions". *Journal of Scientific Computing*, 34:260–286, 2008.

[16] S. Gottlieb. "On high order strong stability preserving Runge-Kutta and multi step time discretizations". *Journal of Scientific Computing*, 25:105–128, 2005.

[17] H.T. Huynh. "An upwind moment scheme for conservation laws". *Computational Fluid Dynamics 2004, Proceedings of the Third International Conference on Computational Fluid Dynamics, ICCFD3, Toronto*, pages 761–766, 2006.

[18] H.T. Huynh. "A flux reconstruction approach to high-order schemes including discontinuous Galkerin methods". *18th AIAA Computational Fluid Dynamics Conference*, 2007.

[19] H.T. Huynh. "A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion". *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, AIAA Paper 2009-0403*, 2009.

[20] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes". *AIAA Fluid and Plasma Dynamics Conference, 14th, Palo Alto, CA, USA, AIAA Paper 81-1259*, 1981.

[21] C. Johnson and J. Pitkaranta. "An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation". *Mathematics of Computation*, 46:1–26, 1986.

[22] P. LeSaint and P.A. Raviart. "On a finite element method for solving the neutron transport equation". *Mathmetical Aspects of Finite Elements in Partial Differential Equations*, pages 89–123, 1974.

[23] M. Lo and B. van Leer. "Analysis and implementation of Recovery-based Discontinuous Galerkin for diffusion". *19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas, USA, AIAA Paper 2009-3786*, 2009.

[24] F. Lorcher, G. Gassner, and C.D. Munz. "A discontinuous Galkerin scheme based on a space-time expansion. I. inviscid compressible flow in one space dimension". *Journal of Scientific Computating*, 32:175–199, 2007.

[25] K. Masatsuka. *"I do like CFD, Vol. 1"*. Lulu.com, 2009.

[26] K.W. Morton. *"Shock capturing, fitting and recovery"*. Springer Berlin/Heidelberg, 1982.

[27] N.C. Nguyen, J. Peraire, and B. Cockburn. "An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations". *Journal of Computational Physics*, 228:3232–3254, 2009.

[28] J.T. Oden, I. Babuska, and C.E. Baumann. "A discontinuous hp finite element method for diffusion problems". *Journal of Computational Physics*, 146:491–519, 1998.

[29] H. Park, R. Nourgaliev, V. Mousseau, and D. Knoll. "Recovery discontinuous Galerkin - Jacobian-free Newton Krylov (rDG-JFNK) method for all-speed Navier-Stokes equations". *International Conference on Computational Fluid Dynamics (ICCFD), Seoul, Korea*, 2008.

[30] J. Peraire and P.O. Persson. "The compact discontinuous galerkin (CDG) method for elliptic problems". *SIAM Journal of Scientific Computating*, 30:1806–1824, 2008.

[31] M. J. Prather. "Numerical advection by conservation of second-order moments". *JGR*, 91:6671–6681, 1986.

[32] W. Reed and T. Hill. "Triangular mesh methods for the neutron transport equation". *Tech. Rep. LA-UR 73-479, Los Alamos National Laboratory*, 1973.

[33] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.

[34] G.L. Russel and J.A. Lerner. A new finite-differencing scheme for the tracer transport equation. *Journal of Applied Meteorology*, 20:1483–1498, 1981.

[35] S.J. Ruuth. "Global optimization of explicit strong-stability-preserving Runge-Kutta methods". *Mathematics of Computation*, 75:183–207, 1997.

[36] Y. Suzuki. *"Discontinuous Galerkin Methods for Extended Hydrodynamics"*. PhD Thesis, 2008.

[37] B. van Leer. "Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection". *Journal of Computational Physics*, 23:276–299, 1977.

[38] B. van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". *Journal of Computational Physics*, 32:101–136, 1979.

[39] B. van Leer. "Computational methods for ideal compressible flow". *In Von Karman Inst. for Fluid Dynamics Computational Fluid Dyn.*, 1:45, 1984.

[40] B. van Leer. "Multidimensional explicit difference schemes for hyperbolic conservation laws". *Computational Methods in Applied Sciences and Engineering, VI*, pages 493–497, 1984.

[41] B. van Leer and M. Lo. "Unification of Discontinuous Galerkin methods for advection and diffusion". *47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida, USA, AIAA Paper 2009-400*, 2009.

[42] B. van Leer, M. Lo, R. Gitik, and S. Nomura. *"A Venerable Family of Discontinuous Galerkin Schemes for Diffusion Revisited"*. World Scientific Review Volume 9, 2010.

[43] B. van Leer, M. Lo, and M. van Raalte. "A discontinuous Galerkin method for diffusion based on recovery". *18th AIAA Computational Fluid Dynamics Conference, Miami, Florida, USA, AIAA Paper 2007-4083*, June 2007.

[44] B. van Leer and S. Nomura. "Discontinuous Galerkin for diffusion". *17th AIAA Computational Fluid Dynamics Conference, Tornoto, Ontario, Canada, AIAA Paper 2005-5108*, 2005.

[45] M. van Raalte and B. van Leer. "Bilinear forms for the Recovery-based Discontinuous Galerkin method for diffusion". *Communications in Computational Physics*, 5:683–693, 2008.

[46] F.M. White. *"Viscous Fluid Flow, Second Edition"*. McGraw-Hill, 1991.

[47] Wikipedia. `http://en.wikipedia.org/wiki/Runge_Kutta_methods`.