AIAA AVIATION Forum
13-17 June 2016, Washington, D.C.
46th AIAA Fluid Dynamics Conference

10.2514/6.2016-3331

Progress Towards the Application of the Recovery-Based Discontinuous Galerkin Method to Practical Flow Physics Problems

Philip E. Johnson* & Eric Johnsen†

Mechanical Engineering Department, University of Michigan, Ann Arbor, MI 48109

Recent progress in the development of the Recovery-Based Discontinuous Galerkin method of Van Leer and Lo is reported. The method is shown to be competent for boundary value problems that involve second order derivatives, including shear terms, such as the viscous terms of the Navier-Stokes equations. As expected, the method's high order of accuracy yields superior results in terms of error versus computational time.

I. Introduction

SINCE being paired with the explicit Runge-Kutta time integration method by Cockburn & Shu,[1] the Discontinuous Galerkin (DG) method has proven to be well-suited for time-dependent hyperbolic systems of equations, such as the Euler equations for compressible inviscid flow. The method exbihits excellent parallel efficiency, can be naturally extended to arbitrarily high order of accuracy, and is compatible with complicated mesh geometry. However, the method is not naturally compatible with parabolic/elliptic PDE, and this deficiency has been a barrier to its widespread application for general convection-diffusion systems, such as the Navier-Stokes equations of fluid dynamics. Many modifications to the general DG method have been proposed to remedy this deficiency, but these methods typically exhibit certain undesireable features, such as very tight stability constraints and ambiguously defined numerical parameters.

A unique scheme among these modifications is the Recovery-based DG (RDG) method of Van Leer & Nomura,[2] explored in detail by Lo.[3] Compared to other DG modifications for handling parabolic/elliptic PDE behavior, RDG is characterized by extremely high order of accuracy, high computational cost per iteration, and a specialized boundary treatment to maintain high order accuracy. The general concept of Recovery has assisted in the development of similar schemes, for example the reconstruction-based DG method of Luo et al.[4] Additionally, Recovery DG has been shown to competent in Direct Numerical Simulation (DNS) of compressible turbulence with periodic boundary conditions by Johnsen et al.[5,6]. However, the Recovery-based DG method as developed by Lo and Van Leer, despite its promise, has been sparsely applied to real flow physics problems.

Our particular interest is using Recovery DG for DNS of unsteady turbulent shock-wave/boundary-layer interaction (SWBLI). This problem is well-described in a review by Clemens.[7] There are many challenges associated with this particular flow physics problem, and a Recovery-based DG approach may work well. However, the method is untested with regard to the prescription of realistic boundary conditions. Thus, boundary procedures for the RDG method have recently been the focus of our research efforts.

This paper has multiple objectives. The first is to explain the RDG1x+ and RDG1x++CO schemes, which are the most versatile versions of the various Recovery-based schemes described by Lo.[3] Next, the

*Graduate Student, University of Michigan, 1231 Beal Ave. 2043 Walter E. Lay Automotive Laboratory, Ann Arbor, MI 48109-2133. AIAA Student Member

†Assistant Professor, University of Michigan, 1231 Beal Ave. 2043 Walter E. Lay Automotive Laboratory, Ann Arbor, MI 48109-2133. AIAA Senior Member

full boundary recovery procedure is reviewed, with emphasis on our findings regarding the difficulties of shear-diffusion systems as opposed to pure (Laplacian) diffusion systems. Then, the scheme is compared with the Local DG (LDG) method of Cockburn and Shu,[8] which is representative of the mixed-formulation approaches typically used for diffusion systems.

II. Standard DG for Transient Hyperbolic PDE

For the sake of clarity, we refer to the established DG method for handling hyperbolic PDE, outlined by Cockburn & Shu,[1] as standard DG. It is briefly described here in semi-discrete form for a time-dependent problem in two spatial dimensions to provide context for the description of RDG. We restrict ourselves to quadrilateral elements, using a tensor product solution basis. Consider the scalar convection-diffusion equation, shown below as a conservation law.

$$\frac{\partial u}{\partial t} + \nabla\cdot\vec{\mathbf{F}} - \nabla\cdot\vec{\mathbf{G}} = 0 \tag{1a}$$

$$\vec{\mathbf{F}} = \begin{bmatrix} v_x u \\ v_y u \end{bmatrix} \qquad \vec{\mathbf{G}} = \underline{\mathbf{D}}\nabla u \tag{1b}$$

$$\underline{\mathbf{D}} = \begin{bmatrix} \mu & \lambda\mu \\ \lambda\mu & \mu \end{bmatrix} \qquad \nabla u = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} \tag{1c}$$

In this form, μ is the diffusion coefficient, λ is a shear diffusion factor, v_x is the advection velocity in the x direction, and v_y is the advection velocity in the y direction. We refer to $\underline{\mathbf{D}}$ as the diffusion tensor.

Let the physical domain Ω of the problem be partitioned by a set of finite elements, each denoted by Ω_e, such that $\Omega = \cup\Omega_e$. Additionally, let each edge of each element $\partial\Omega_e$ be either shared with another element, as an interface, or coincident with a boundary of the physical domain. In the case of the DG method, it is convenient to map every element in physical space to some reference element. In our particular case, a unit square is used with reference coordinates ξ and η, as shown below.

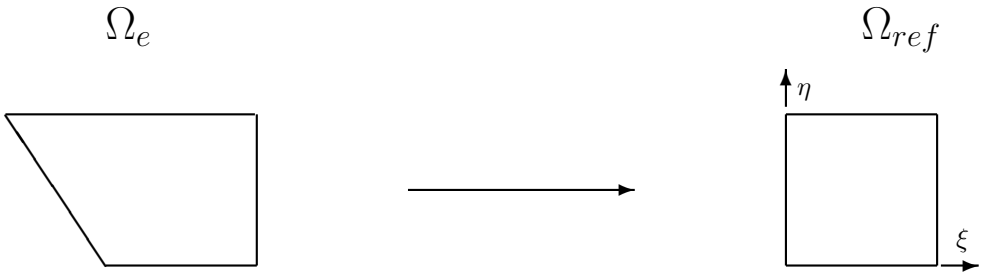

Figure 1. The reference element.

We use a polynomial expansion inside each element to represent the numerical solution. Let V^h be a finite solution space of polynomials; the natural choice on quadrilateral elements is a tensor product basis built from one-dimensional polynomials. The approximate solution u^h within each element is represented as a linear combination of local degrees of freedom (DOF), denoted $\hat{u}$, and the shape functions in the solution space. In this work, we assume that the order of each element is the same, and that each element is mapped to the reference element, specifically the unit square. The formal representation is shown below, with p representing the degree of the one-dimensional shape functions used to form the tensor product V^h and K being the degrees of freedom per element.

$$V^h = \{\phi^m(\xi,\eta), \quad (\xi,\eta) \in [0,1]\times[0,1]\} \tag{2a}$$

$$u_e^h(\xi,\eta,t) = \sum_{m=1}^{K} \hat{u}_e^m(t)\phi^m(\xi,\eta) \tag{2b}$$

In the system shown above, $K = (p+1)^2$ is the dimension of the solution space V^h, and ξ and η are reference coordinates within the local finite element Ω_e. Each finite element has its own vector of time-dependent solution weights $\hat{\mathbf{u}}_\mathbf{e}$, and this element-specific vector contains K entries. From now on, the time argument is dropped from the cell DOF, and the spatial arguments are dropped from the shape functions for cleanliness. In order to satisfy the conservation law, we satisfy the weak form, shown below, for K test functions ϕ^k in each element.

$$\begin{aligned}\int_{\Omega_e} \frac{\partial u_e^h}{\partial t}\phi^k dA = &-\int_{\partial\Omega_e} \phi^k(\hat{\mathbf{H}}_{conv}(u^-,u^+)\cdot\vec{n}^-)ds + \int_{\Omega_e}\vec{\mathbf{F}}(u_e^h)\cdot\nabla\phi^k dA \\ &+\int_{\partial\Omega_e}\phi^k(\hat{\mathbf{H}}_{diff}(u^-,u^+,\nabla u^-,\nabla u^+)\cdot\vec{n}^-)ds - \int_{\Omega_e}\vec{\mathbf{G}}(u_e^h,\nabla u_e^h)\cdot\nabla\phi^k dA\end{aligned} \tag{3}$$

Along the edges of the local element, Ω_e, the approximate solution is multi-valued because either another element shares the edge, or the edge is a member of the boundary of the physical domain. The job of the $\hat{\mathbf{H}}$ operators is to return a single flux vector, either $\vec{\mathbf{F}}$ in the case of $\hat{\mathbf{H}}_{conv}$ or $\vec{\mathbf{G}}$ in the case of $\hat{\mathbf{H}}_{diff}$, given the two approximations u^+ from the neighboring cell (or boundary) and u^- from the local cell, Ω_e. The unit normal vector $\vec{n}^-$ points outward from the local element Ω_e.

For purely hyperbolic PDE, such as the time-dependent Euler equations, the standard method achieves $2p+1$ order of accuracy in the L_2 norm of the cell-averaged error when the solution is smooth. For the convection-diffusion system shown above, if the diffusion coefficient μ is nonzero, the method underperforms compared to the purely hyperbolic case.

DG performs poorly for parabolic PDE because of two issues. First, information is lost when taking a gradient of the polynomial approximation u^h. Thus, the approximate gradient ∇u^h is of lower quality than the approximation u^h itself, and the accuracy of the method suffers. The second difficulty is specifying a value for $\hat{\mathbf{H}}_{diff}$ along the element boundaries. For conservation, this value needs to be the same along the edge for both of the neighboring elements. The most natural choice, taking $\vec{\mathbf{G}}(avg(u^-,u^+), avg(\nabla u^-,\nabla u^+))$, performs very poorly, and an alternate approach in necessary.

Multiple methods have been proposed for addressing the two issues discussed above, two popular candidates being the Local DG (LDG) method of Cockburn & Shu[8] and the BR2 scheme of Bassi & Rebay.[9] These schemes fit under the umbrella of mixed formulation methods,[10] which are built from the motivation of substituting a high-quality approximation σ^h, constructed directly in the space V^h, for the gradient ∇u^h. A thorough review of mixed formulation methods and related penalty methods can be found in the 2001 analysis of Douglas et al.[10]

Van Leer's Recovery philosophy takes a different approach than the mixed formulation methods, instead building a high-order continuous and differentiable solution f, the recovered solution, across each interface in order to populate the viscous interface term $\hat{\mathbf{H}}_{diff}$. The details of the method, further developed by Lo,[3] are given in the next section. The basic philosophy of the method has been applied in the formation of two new algorithms that we are aware of, namely Borrel & Ryan's Elastoplast DG method[11] and the Enhanced-Stability Recovery (ESR) scheme of Ferrero et al.,[12] but this paper will focus on the evolved forms of RDG developed by Lo, which preserve extremely high order of accuracy.

III. The Recovery and Enhancement Schemes for Nonlinear Shear-Diffusion

For the purely diffusive case of Equation (1), that is, setting the advection speeds to zero, RDG1x++CO on a Cartesian grid achieves $3p+1$ order of accuracy for odd p and $3p+2$ order of accuracy for even p, with

the error taken in the cell-averaged L_2 norm. The stated order of accuracy is based both on Fourier analysis and numerical experimenation. The cheaper RDG1x+ scheme achieves this advertised order of accuracy only for the specific case of a diagonal diffusivity tensor on Cartesian elements. Both of these schemes use Recovery to remove ambiguity from the numerical solution at each element interface, as explained in this section. This section is exclusively a review of methods designed by Lo.[3] It exists to orient the reader before the boundary procedures are described in a later section; when Neumann or Dirichlet boundary conditions are involved, as opposed to the case of periodic boundaries, special care must be taken to maintain the method's high order of accuracy.

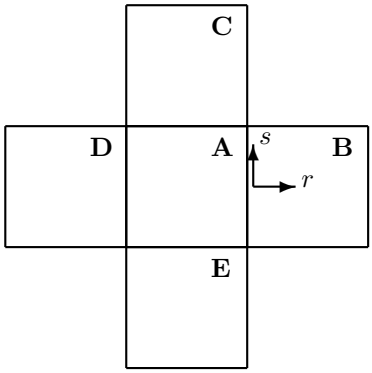

Figure 2. Neighboring elements.

To describe Recovery, we continue with the assumption of quadrilateral elements. Consider elements A,B,C,D,and E, as shown above, on the interior of the computational grid. The philosophy of Recovery is to define a unique, continuous, and differentiable polynomial solution across each element edge. For example, the interface that separates element A and element B will be populated by some recovered solution f_{AB}. On each interface, we make use of a set of interface coordinates, with r running in the face-normal direction and s in the face-tangential direction. Similarly, each of the other three edges that border element A will have an associated recovered solution and unique coordinate system.

To perform recovery, we construct a polynomial approximation f for u^h over the union of two adjacent elements, such that f gives a high-quality approximation over the interface shared by the elements. Over the union of two elements Ω_A and Ω_B, we have K solution weights defining u_A^h in Ω_A and K solution weights defining u_B^h in Ω_B. This information is combined to form a polynomial expansion, f, which approximates the solution over $\Omega_A \cup \Omega_B$ as a polynomial expansion using $2K$ solution weights. Let f be written as a polynomial expansion of functions in the space W. Like V^h, W is a tensor product basis, but while limited to order p in the face-tangential direction, s, it posesses very high polynomial order in the face-normal direction, r. The interface reference coordinates r and s can be normalized according to user preference.

$$W = \{\psi^n(r,s)\} \tag{4a}$$

$$f(r,s) = \sum_{n=1}^{2K} \hat{f}^n \psi^n(r,s) \tag{4b}$$

As an example of the space W, in the $p1$ case, each element contains 4 DOF. Thus, the recovered solution contains 8 degrees of freedom. However, while the DG solution basis V^h is bilinear in each element, the recovered function is built on a tensor product basis that is quartic in the face-normal direction but only linear in the face-tangential direction.

The recovered function, f, is weakly equivalent to u^h over each of the two elements that share the interface. Its DOF $\hat{\mathbf{f}}$ must be set based on the approximate solutions u_A^h and u_B^h. There are $2K$ degrees of freedom for every function f, and they are constrained as shown below.

$$\int_{\Omega_A} f\phi^k dxdy = \int_{\Omega_A} u_A^h \phi^k dxdy \quad \forall k \in \{1..K\} \tag{5a}$$

$$\int_{\Omega_B} f\phi^k dxdy = \int_{\Omega_B} u_B^h \phi^k dxdy \quad \forall k \in \{1..K\} \tag{5b}$$

By rewriting Equation (5) in terms of the vector of recovery weights $\hat{\mathbf{f}}$ and the two adjacent vectors of solution weights $\hat{\mathbf{u}}_\mathbf{A}$ and $\hat{\mathbf{u}}_\mathbf{B}$, we obtain a matrix-vector system that yields $\hat{\mathbf{f}}$ once $\hat{\mathbf{u}}_\mathbf{A}$ and $\hat{\mathbf{u}}_\mathbf{B}$ are known.

We emphasize here that the recovered function is obtained via the DG solution coefficients $\hat{\mathbf{u}}$. Obtaining the recovery weights $\hat{\mathbf{f}}$ across each interface is the first step in populating Equation (3), given the DG solution $\hat{\mathbf{u}}$ at some time t. A schematic of the process is given in Figure 3. For this particular example, a $p2$ discretization in two dimensions is used, with 9 DOF per element. The DG solution shown is weakly equivalent to the exact solution over each element, but introduces a discontinity at the interface. The recovered function, f_{AB}, uses the DG solution in each element to create a smooth, high-order solution approximation over $\Omega_A \cup \Omega_B$.

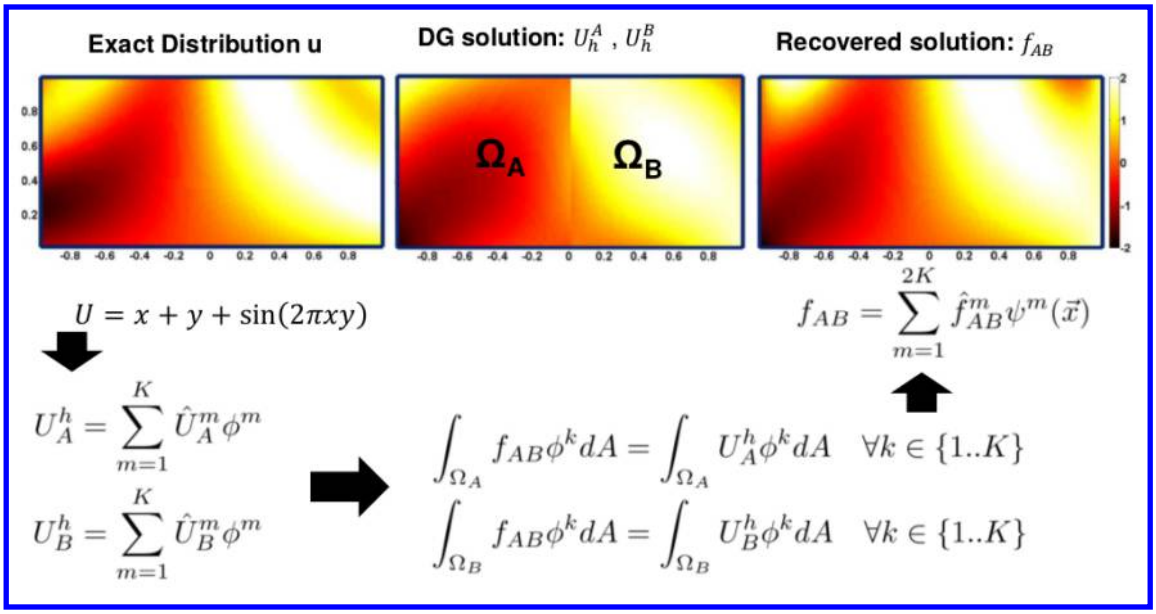

Figure 3. The Recovery process in 2D, $p2$ discretization.

Once the primary recovery weights throughout the physical domain are known, they are used to populate the primary recovered solution along every interface. In the case of RDG1x+, $\vec{\mathbf{G}}(f, \nabla f)$ is substituted for $\hat{\mathbf{H}}_{diff}$ in Equation (3) to populate the surface terms. In the case of the more versatile (and more expensive) scheme RDG-1x++CO scheme, there are more operations to be completed before populating $\hat{\mathbf{H}}_{diff}$.

For both of the schemes, the primary recovered solution is used to assist in the process deemed solution enhancement by Van Leer and Lo. A thorough summary is given by Lo.[3] Whereas recovery is used to populate a solution approximation along element interfaces, solution enhancement is used to replace the DG solution u_e^h over each element with an enhanced solution, u_e^{en}. Specifically, where the DG solution u^h consists of K DOF per element, the enhanced solution u^{en} makes use of $K + 4(p+1)$ DOF per element, where p is again the one-dimensional order of V^h. For example, considering again the $p1$ case, $K = 4$, and the enhanced solution contains 12 DOF. Like the DG solution, the enhanced solution is a polynomial expansion, as shown below.

$$V^{en} = \{\phi^m_{en}(\xi,\eta) \quad , \quad (\xi,\eta) \in [0,1]\times[0,1]\} \tag{6a}$$

$$u^{en}_e(\xi,\eta) = \sum_{m=1}^{K^{en}} \hat{u}^{en,m}_e \phi^m_{en}(\xi,\eta) \quad , \quad K^{en} = (p+1)^2 + 4(p+1) \tag{6b}$$

We provide an illustration for clarity here, again assuming a $p1$ discretization. Suppose that the basis functions $\phi \in V^h$ are constructed as a tensor product of 1D monomials. The enhanced basis, V^{en}, will inherit all members of V^h, and it will contain extra shape functions that are used to improve the accuracy of the approximate solution. Regardless of p, the enhanced solution basis will contain 2 extra columns of shape functions in the ξ direction and 2 extra rows in the η direction, as shown below.

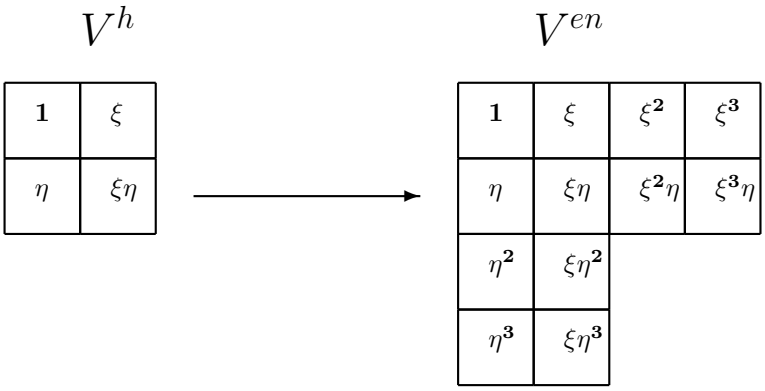

Figure 4. Basis construction for u^h and u^{en}.

The organization of the enhanced basis is based on how we wish to constrain the enhanced solution over each element. For the case of a quadrilateral, the K^{en} DOFs in each element's enhanced solution u^{en} are constrained as follows. For the quadrilateral element, let $\{\tau_0,\tau_1,\tau_2,\tau_3\}$ be the set of the four edges that border Ω_e. Each edge is populated by a primary recovered solution f. We take $L^k(s_q)$ to be some appropriately chosen 1-dimensional shape function taken at coordinate s_q along a given interface τ_q. An example of an appropriate shape function would be a simple monomial. These constraints force u^{en}_e to be weakly equivalent to u^h_e over the element interior and weakly equivalent to the appropriate recovered function f over each of the four bordering edges.

$$\int_{\Omega_e} u^{en}_e \phi^k dxdy = \int_{\Omega_e} u^h_e \phi^k dxdy \quad \forall \phi^k \in V^h \tag{7a}$$

$$\int_{\tau_q} u^{en}_e L^k_s(s_q) ds_q = \int_{\tau_q} f_q L^k_s(s_q) ds_q \quad \forall k \in \{1..p+1\} \quad \forall q \in \{1..4\} \tag{7b}$$

The system of Equation (7) gives K constraints based on the approximate solution u^h_e of the local element, and each of the four interfaces contribute $p+1$ unique constraints. As with the primary recovery constraints of Equation (5), this system can be written as a linear matrix-vector equation that yields the K^{en} DOF $\hat{\mathbf{u}}^{\mathbf{en}}_{\mathbf{e}}$ given the K DOF $\hat{\mathbf{u}}^{\mathbf{h}}_{\mathbf{e}}$ and the $2K$ DOF $\hat{\mathbf{f}}_{\mathbf{q}}$ along each of the four relevant interfaces. The enhanced solution is then used to populate $\vec{G}(u^{en}, \nabla u^{en})$ over the interior of the local element.

The difference between the RDG1x+ and RDG1x++CO schemes is that for the latter, recovery is actually performed twice per interface in the effort to populate Equation (3). The second iteration of recovery is necessary because the primary recovered solution across each interface is a polynomial of only order p in the interface-tangential direction, which means that RDG1x+ cannot accurately calculate $\hat{\mathbf{H}}_{diff}$ unless $\underline{\mathbf{D}}$ is purely diagonal and the elements are Cartesian. RDG1x++CO remedies this shortcoming.

We now describe the final piece of the more versatile RDG1x++CO scheme. Consider the space V^{en}; the interface traces of u^{en} produce a high-quality gradient approximation in all directions, thanks to the extra shape functions in V^{en} compared to V^h. After primary recovery and solution enhancement have been performed, RDG1x++CO performs recovery on the enhanced solution u^{en} in order to produce the secondary recovered function, f^{en}, across each interface. When a second iteration of recovery is performed using the enhanced solutions u^{en}, the secondary recovered solution f^{en} inherits the high order of u^{en} in the face-tangential direction, enabling the method to maintain the greater than $3p$ order of accuracy, even in the case of a non-diagonal diffusion tensor. The interface flux $\hat{\mathbf{H}}_{diff}$ is then populated using $\vec{\mathbf{G}}(f^{en}, \nabla f^{en})$.

There is an important peculiarity to the secondary recovery. The *CO* in the scheme is short for Cartesian Optimization. Without Cartesian Optimization, the secondary recovery would consist of setting $2K^{en}$ degrees of freedom for the solution f^{en} across each interface, using K^{en} DOF from each of elements that share the interface.

Instead, each secondary recovered solution f^{en} contains $2(p+1)(p+3)$ DOF; we ignore some of the DOF in the enhanced solution u^{en} of each element. For example, in the $p1$ case, where u^{en} has 12 DOF per element, we could define a 24 DOF enhanced recovery function f^{en} across each interface. Instead, Cartesian Optimization retains only 16 DOF in the secondary recovered function on each interface. The details of this procedure, which relies heavily on careful designation of the test functions for the secondary recovery, are given by Lo.[3] Compared to perfoming a recovery iteration with the full spectrum of the $2K^{en}$ available DOF, which we would designate RDG1x++, the Cartesian Optimization procedure yields less computational work per interface and a larger Von Neumann number for numerical stability, while maintaining greater than $3p$ order of accuracy on quadrilateral elements.

IV. Observations

The main advantage of recovery is that where conventional DG schemes for diffusion achieve $2p$ order of accuracy in the cell-averaged L_2 error norm, Recovery achieves greater than $3p$ order of accuracy. However, this increased accuracy comes at a cost, the sources of which include the following three chracteristics. First, the recovery and enhancmcement procedures rely on more matrix-vector multiplications than are present in mixed formulation methods. Second, for a given approximation order p, due to the higher degree of the recovered and enhanced solutions, RDG requires higher-resolution numerical quadrature than conventional DG schemes in order to achieve the advertised order of accuracy. Third, the second recovery iteration of RDG1x++CO results in a non-compact stencil, which could be a hindrance in massively parallel simulations.

Our preliminary work in moving the scheme to non-Cartesian quadrilateral elements has uncovered another issue. For non-uniform elements, in addition to populating a mass matrix for every element, RDG requires that each element have access to a unique matrix for the enhancement procedure, in addition to the individual matrices necessary to solve for the weights of the recovered solutions across the interfaces. The memory usage of our code for Recovery on skewed quadrilaterals is consequently severe compared to mixed formulation approaches. Additionally, the numerical stability of the scheme is sensitive to the matrix conditions numbers of the enhancement and recovery matrices, so a prudent choice of basis functions is crucial. Our preferred method is to use Legendre polynomials to build the recovery and enhancement solution spaces. These challenges must be kept in mind as Recovery DG is applied to large scientific computing problems.

V. Boundary Procedure

Consider an element, Ω_A, that shares at least one interface with a Dirichlet boundary, $\partial\Omega_D$. We refer to Ω_A as the boundary element. The intersection $\partial\Omega_A \cup \partial\Omega_D$ is the boundary interface, and will be labeled τ_D. Also, let Ω_B be the element that touches Ω_A opposite the boundary interface. For example, if the physical domain is a square with Dirichlet boundaries and Ω_A touches the left-side boundary of Ω, then Ω_B is the element immediately to the right of Ω_A. The element Ω_B is known as the interior element. The situation is shown below, with the recovery reference coordinates r and s originating on τ_D.

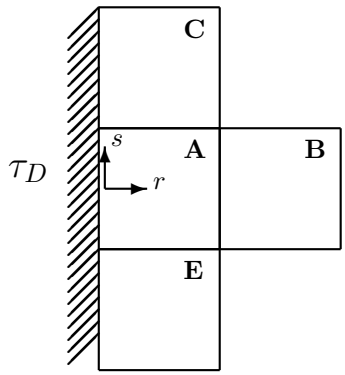

Figure 5. The Boundary Environment

We summarize the full boundary procedure of Lo[3] before presenting our findings. For the boundary interface of Ω_A, in addition to weak equivalence relations over Ω_A and Ω_B, the recovered function f_{AD} must be weakly equivalent to the prescribed Dirichlet condition $C_D(s)$ along the boundary. As with Recovery over an interior interface, the recovered function f along the boundary interface will be a polynomial expansion with $2K$ DOF. These degrees of freedom are constrained by weak equivalence relations as shown below.

$$\int_{\Omega_A} f\phi^k dxdy = \int_{\Omega_A} u_A^h \phi^k dxdy \quad \forall k \in \{1..K\} \tag{8a}$$

$$\int_{\Omega_B} f\beta^k dxdy = \int_{\Omega_B} u_B^h \beta^k dxdy \quad \forall k \in \{1..K-(p+1)\} \tag{8b}$$

$$\int_{\tau_D} fL^k(s)ds = \int_{\tau_D} C_D(s)L^k(s)ds \quad \forall k \in \{1..p+1\} \tag{8c}$$

In this case, the functions β^k are carefully chosen from the DG basis V^h such that the resulting matrix-vector system is invertible. As with the solution enhancement section, L^k represents some 1D test function along the boundary interface. This procedure maintains high order of accuracy for the solution derivatives normal to the boundary, but not their derivatives in the tangential direction; if the tangential derivative along a boundary is needed, which could be the case for the Navier-Stokes equations, then the method underperforms. We have worked to remedy this shortcoming.

Let the preceding boundary procedure be known as the **F∅** procedure. The F stands for full boundary procedure, and the ∅ is used to show that no special treatment is made for tangential derivatives along the boundary. It is possible to perform secondary recovery along a boundary using a similar approach to that shown in Equation (8), forcing equivalence with the enhanced cell solutions u^{en} rather than u^h; this method is abbreviated **FF**, standing for full primary recovery, then full secondary recovery along the boundary. It yields an enhanced recovery solution, f_{AD}^{en}, that is of high polynomial degree in both the boundary-normal and boundary-tangential directions.

The **FF** procedure was our first attempt to remedy the shortcoming of the **F∅** procedure. Our conclusion is that **FF** is difficult to implement and not worth the effort. Experimentation has shown two outstanding issues with this approach. First, it lowers the maximimum allowable timestep for stable explicit time integration. Second, it does not, in general, maintain the greater than $3p$ order of accuracy of RDG.

Our newly-developed scheme to combat this issue is the **FM** scheme, for full primary recovery along the boundary, and a mixed approach for populating the gradient information. It consists of three steps:

1 : Perform full primary boundary recovery procedure to obtain f_{AD}.

2 : Perform cell solution enhancement to obtain u_A^{en}.

3 : Along τ_D, use f_{AD} for the boundary-normal derivative and u_A^{en} for the boundary-tangential derivative.

The performance of the three schemes is now demonstrated with a test. Again, we make use of Equation (1), but set advection speeds to zero. The remaining parameters, and the manufactured solution, are shown in Equation (9). The spatial domain is the unit square, partitioned by uniform square elements. Each simulation runs from $t = 0$ to $t = 2k_w^{-2}$. Based on the manufactured solution, space-time dependent Dirichlet conditions are enforced along boundaries. For each of the three boundary schemes discussed, the test problem is solved on a series of succesively finer meshes, each characterized by the uniform element edge length Δx. Our measurement of error is the L_2 norm of the cell-averaged error in u.

$$\mu = 1 + \frac{u^2}{10} \quad , \quad \lambda = 0.25 \quad , \quad k_w = 2\pi \quad , \quad v_x = v_y = 0 \tag{9a}$$

$$u = e^{-{k_w}^2 t}(sin(k_w(x-y)) + sin(k_w x)sin(k_w y)) \tag{9b}$$

The requirement for a scheme to pass this test is that it achieves Recovery's advertised $3p+1$ order of accuracy for even p and $3p+2$ order of accuracy for odd p. In addition to reporting the grid resolution and simulation error for each case, we report the base 10 logarithms of these two quantities. Then, the order of accuracy (abbreviated **OOA**) is calculated and reported between every two data points.

As shown in Table 1, Table 2, and Table 3, where the other two schemes fail, the **FM** scheme achieves 8^{th} order accuracy in the $p2$ case and 10^{th} order accuracy in the $p3$ case. Thus, our DG scheme is working just as well near the boundaries as it is over the domain interior, and the **FM** scheme passes this test. We have no authoritative explanation for the failure of the **FF** scheme.

Table 1. Performance of F∅ scheme.

Δx	**p**	**Error**	log **Δx**	log **Error**	**OOA**
1.047	2	$6.08E-6$	+0.02	−5.22	
0.785	2	$2.20E-6$	−0.11	−5.66	3.54
0.524	2	$5.07E-7$	−0.28	−6.30	3.61
0.393	2	$1.74E-7$	−0.41	−6.76	3.73
0.262	2	$3.70E-8$	−0.58	−7.43	3.81
2.094	3	$9.67E-6$	+0.32	−5.02	
1.571	3	$2.73E-6$	+0.20	−5.56	4.40
1.047	3	$3.19E-7$	+0.02	−6.50	5.29
0.785	3	$6.34E-8$	−0.11	−7.20	5.62
0.524	3	$6.22E-9$	−0.28	−8.21	5.73
0.393	3	$1.19E-9$	−0.41	−8.92	5.74

Table 2. Performance of FF scheme.

Δx	p	Error	$\log \Delta x$	$\log$ Error	OOA
1.047	2	$1.22E-7$	$+0.02$	-6.92	
0.785	2	$1.97E-8$	-0.11	-7.71	6.33
0.524	2	$1.87E-9$	-0.28	-8.73	5.80
0.393	2	$3.76E-10$	-0.41	-9.43	5.59
0.262	2	$4.02E-11$	-0.58	-10.4	5.51
2.094	3	$5.92E-7$	$+0.32$	-6.23	
1.571	3	$5.20E-8$	$+0.20$	-7.28	8.46
1.047	3	$1.39E-9$	$+0.02$	-8.86	8.93
0.785	3	$2.35E-10$	-0.11	-9.63	6.19
0.524	3	$1.93E-11$	-0.28	-10.7	6.16
0.393	3	$3.31E-12$	-0.41	-11.5	6.13

Table 3. Performance of FM scheme.

Δx	p	Error	$\log \Delta x$	$\log$ Error	OOA
1.047	2	$1.10E-7$	$+0.02$	-6.96	
0.785	2	$1.23E-8$	-0.11	-7.91	7.64
0.524	2	$5.37E-10$	-0.28	-9.27	7.72
0.393	2	$5.81E-11$	-0.41	-10.2	7.73
0.262	2	$2.59E-12$	-0.58	-11.6	7.67
2.094	3	$5.88E-7$	$+0.32$	-6.23	
1.571	3	$5.31E-8$	$+0.20$	-7.28	8.35
1.047	3	$3.04E-10$	$+0.02$	-9.52	12.7
0.785	3	$1.24E-11$	-0.11	-10.9	11.1
0.524	3	$2.30E-13$	-0.28	-12.6	9.84

VI. Performance Comparison with LDG

We now present a quantitative comparison between RDG-1x++CO and the well-known LDG scheme. This test will compare the two schemes in terms of both numerical error and computation time, accounting for the increased cost and the increased order of accuracy of Recovery. The governing equation is again Equation (1), with the advection speeds set to zero to obtain a purely parabolic PDE. The remaining parameters are shown below in Equation (10). The domain for this simulation is a square, $(x, y) \in [0, 2\pi] \times [0, 2\pi]$. We use a uniform mesh of square elements, each with edge length Δx. The solution is marched from the initial condition at $t = 0$ to $t = 2k_w^{-2}$. The familiar heat equation, free of shear terms, is solved with time-dependent Dirichlet boundary conditions, in accordance with the solution shown below.

$$\mu = 1 \quad , \quad \lambda = 0 \quad , \quad k_w = 1.2 \quad , \quad v_x = v_y = 0 \tag{10a}$$

$$u = e^{-{k_w}^2 \mu t}(sin(k_w x) + sin(k_w y)) \tag{10b}$$

The Recovery result here makes use of the **FM** scheme developed in the previous section. For this

particular case, the $\mathbf{F}\emptyset$ scheme would be appropriate. However, we make use of the more costly **FM** scheme to present a comparison between two schemes that would be appropriate for the viscous terms of the Navier-Stokes equations, not just the simple heat quation.

Table 4 gives some scheme configuration parameters. Explicit Runge-Kutta (RK) time integration is used for all schemes. We use different order RK schemes based on the method and the solution order p, in order to both minimize computation time and ensure that the time integration scheme is of sufficient order to preserve each scheme's spatial order of accuracy. The exception to this rule is the $p2$ case with Recovery. For this particular test, Recovery behaves erratically when paired with the RK4 scheme, but achieves the desired order of accuracy when we switch to RK5. The cause for this behavior is presently under investigation. Using the cell-averaged L_2 error norm, the order of accuracy of LDG is $2p$, such that it is 2^{nd} order accurate in the $p1$ case and 4^{th} order accurate in the $p2$ case. RDG is 4^{th} order accurate in the $p1$ case and 8^{th} order accurate in the $p2$ case.

For each configuration, the maximum Von Neumann number for numerical stability is determined experimentally and reported in Table 4. The increment Δt is recalculated every time step based on the maximum diffusion coefficient μ found in the time-dependent approximate solution u^h. For square elements, we define the VNN as shown below.

$$VNN = \frac{2\mu_{max}\Delta t}{\Delta x^2} \tag{11}$$

Table 4. Configuration

Method	p	RK order	VNN
LDG	1	2	$\frac{1}{19}$
LDG	2	2	$\frac{1}{74}$
RDG	1	2	$\frac{1}{8}$
RDG	2	5	$\frac{1}{10}$

The Von Neumann numbers must not be viewed here as a direct comparison of the stability of the schemes; for the $p2$ case, Recovery is paired with RK5 instead of RK2, which influences the maximum allowable Δt for numerical stability. If Recovery were paired with RK2 instead of RK5, the maximum allowable VNN would be expected to drop below its present value, based on the stability regions of the RK5 and RK2 time integration routines.

The measure of error is the L_2 norm of the cell-averaged error in u. Figure 6 shows the error in the element-averaged L_2 norm vs. the work load for each simulation. All computations are performed on a single processor. To calculate work load, we take the total run time of a simulation and subtract the post-processing and initialization times. The run times are then non-dimensionalized by the time taken for the $p2$ Recovery case to run on the coarsest of the meshes, which is approximately 4.1 seconds. This nondimensionalized run time is reported as the work load. Several runs are presented in each configuration, with different mesh resolutions Δx; our coarsest mesh uses 144 total elements (12 in each direction), and the finest mesh uses $16,384$ elements (128 in each direction).

This test indicates that for a given amount of computational time, a high-order Recovery scheme gives superior results for a scalar parabolic PDE, whether the discretization is with $p1$ or $p2$ elements. Future work will include attempting to extend this behavior to the full Navier-Stokes equations with realistic boundary conditions. Additionally, the performance of recovery on non-uniform meshes must be explored in detail. We have obtained excellent results when applying Recovery on skewed quadrilateral grids with periodic boundary conditions, but the method's performance has not yet been measured for Dirichlet/Neumann boundary conditions.

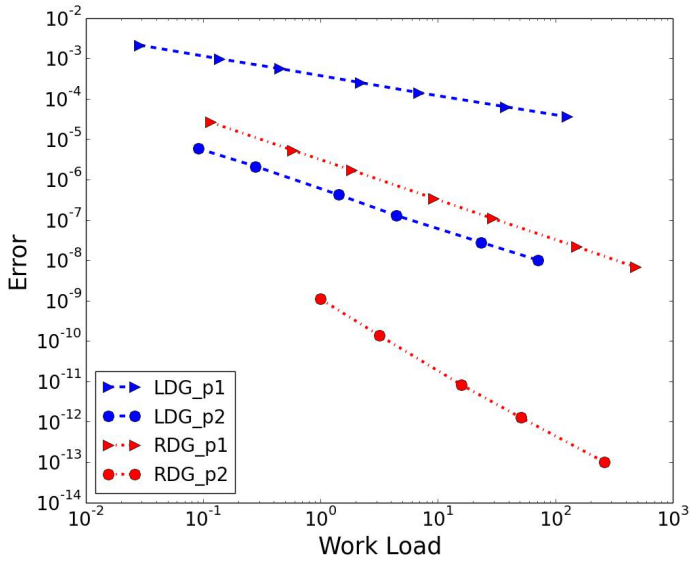

Figure 6. Comparison of Local DG and Recovery DG for heat equation problem.

VII. Conclusion

In this study, we have presented a method to maintain Recovery DG's high-order accuracy for shear-diffusion problems under the application of Dirichlet boundary conditions. We discovered that great care must be taken to avoid compromising the scheme's high order of accuracy. Additionally, we have compared the scheme with the popular Local DG method in terms of accuracy versus computational cost. Recovery DG has been shown to perform comparatively well in this regard, as is to be expected from a high-order method.

Some work remains before Recovery DG can be proficiently applied to general flow physics problems. The main issue is the fact that when paired with standard DG for advection-diffusion systems, the full scheme will be limited to $2p+1$ order of accuracy. This issue can be addressed by a reduced order Recovery scheme, which would lower the cost of Recovery while bringing its order of accuracy near that of the standard advection scheme. Alternatively, one could attempt to improve the order of accuracy with which the advective terms are captured through a reconstruction/enhancement scheme, pushing the full scheme's order of accuracy past $2p + 1$.

Additionally, it is still not clear how to generalize RDG1x+ and RDG1x++CO to non-quadrilateral elements while retaining greater than $3p$ order of accuracy. These issues are to be addressed in future studies.

References

[1]Cockburn, B. and Shu, C.-W., "Runge Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 89.

[2]Van Leer, Bram; Nomura, S., "Discontinuous Galerkin for Diffusion," *17th AIAA Computational Fluid Dynamics Conference, Toronto, Ontario*, 2005.

[3]Lo, K., *A Space-Time Discontinuous Galerkin Method for Navier-Stokes with Recovery*, Dissertation, University of Michigan, 2011.

[4]Luo, H., Luo, L., Nourgaliev, R., Mousseau, V. A., and Dinh, N., "A reconstructed discontinuous Galerkin method for the compressible NavierStokes equations on arbitrary grids," *Journal of Computational Physics*, Vol. 229, No. 19, sep 2010, pp. 6961–6978.

[5]Johnsen, E., Varadan, S., and Leer, B. V., "A Three-Dimensional Recovery-Based Discontinuous Galerkin Method for

Turbulence Simulations," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, 2013.

[6]Johnsen, E., Nair, A., and Varadan, S., "Recovery Discontinuous Galerkin Method for Compressible Turbulence," *21st AIAA Computational Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, jun 2013.

[7]Clemens, N. T. and Narayanaswamy, V., "Low-Frequency Unsteadiness of Shock Wave/Turbulent Boundary Layer Interactions," *Annual Review of Fluid Mechanics*, Vol. 46, No. 1, jan 2014, pp. 469–492.

[8]Cockburn, B. and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1997, pp. 2440–2463.

[9]Bassi, F., Crivellini, A., Rebay, S., and Savini, M., "Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and kω turbulence model equations," *Computers & Fluids*, Vol. 34, No. 4-5, 2005, pp. 507–540.

[10]Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., "Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems," *SIAM J. Numer. Anal.*, Vol. 39, No. 5, 2001, pp. 1749–1779.

[11]Borrel, M. and Ryan, J., "A New Discontinuous Galerkin Method for the NavierStokes Equations," *Spectral and High Order Methods for Partial Differential Equations*, 2011, pp. 373–381.

[12]Ferrero, A., Larocca, F., and Puppo, G., "A robust and adaptive recovery-based discontinuous Galerkin method for the numerical solution of convection-diffusion equations," *International Journal for Numerical Methods in Fluids*, Vol. 77, No. 2, jan 2015, pp. 63–91.