Commun. Comput. Phys.
doi: 10.4208/cicp.250911.030212a

Vol. **12**, No. 5, pp. 1495-1519
November 2012

A Reconstructed Discontinuous Galerkin Method for the Euler Equations on Arbitrary Grids

Hong Luo[1,*], Luqing Luo[1] and Robert Nourgaliev[2]

[1] *Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC, 27695, USA.*
[2] *Thermal Science and Safety Analysis Department, Idaho National Laboratory, Idaho Falls, ID, 83415, USA.*

Received 25 September 2011; Accepted (in revised version) 3 February 2012

Communicated by Kun Xu

Available online 22 May 2012

Abstract. A reconstruction-based discontinuous Galerkin (RDG(P1P2)) method, a variant of P1P2 method, is presented for the solution of the compressible Euler equations on arbitrary grids. In this method, an in-cell reconstruction, designed to enhance the accuracy of the discontinuous Galerkin method, is used to obtain a quadratic polynomial solution (P2) from the underlying linear polynomial (P1) discontinuous Galerkin solution using a least-squares method. The stencils used in the reconstruction involve only the von Neumann neighborhood (face-neighboring cells) and are compact and consistent with the underlying DG method. The developed RDG method is used to compute a variety of flow problems on arbitrary meshes to demonstrate its accuracy, efficiency, robustness, and versatility. The numerical results indicate that this RDG(P1P2) method is third-order accurate, and outperforms the third-order DG method (DG(P2)) in terms of both computing costs and storage requirements.

AMS subject classifications: 65M60, 65M99, 76M25, 76M10

Key words: Discontinuous Galerkin methods, least-squares reconstruction, compressible Euler equations.

1 Introduction

The discontinuous Galerkin methods [1–28] (DGM) have recently become popular for the solution of systems of conservation laws. Originally introduced for the solution of neutron transport equations [1], nowadays they are widely used in computational fluid dynamics, computational acoustics, and computational magneto-hydrodynamics. The discontinuous Galerkin methods combine two advantageous features commonly associated

*Corresponding author. *Email addresses:* hong_luo@ncsu.edu (H. Luo), lluo2@ncsu.edu (L. Luo), Robert.Nourgaliev@inl.gov (R. Nourgaliev)

with finite element and finite volume methods. As in classical finite element methods, accuracy is obtained by means of high-order polynomial approximation within an element rather than by wide stencils as in the case of finite volume methods. The physics of wave propagation is, however, accounted for by solving the Riemann problems that arise from the discontinuous representation of the solution at element interfaces. In this respect, the DG methods are similar to finite volume methods. The discontinuous Galerkin methods have many attractive features: 1) They have several useful mathematical properties with respect to conservation, stability, and convergence; 2) The methods can be easily extended to higher-order ($>2^{\text{nd}}$) approximation; 3) The methods are well suited for complex geometries since they can be applied on unstructured grids. In addition, the methods can also handle non-conforming elements, where the grids are allowed to have hanging nodes; 4) The methods are highly parallelizable, as they are compact and each element is independent. Since the elements are discontinuous, and the inter-element communications are minimal, domain decomposition can be efficiently employed. The compactness also allows for structured and simplified coding for the methods; 5) They can easily handle adaptive strategies, since refining or coarsening a grid can be achieved without considering the continuity restriction commonly associated with the conforming elements. The methods allow easy implementation of *hp*-refinement, for example, the order of accuracy, or shape, can vary from element to element; 6) They have the ability to compute low Mach number flow problems without recourse to the time-preconditioning techniques normally required for the finite volume methods. In contrast to the enormous advances in the theoretical and numerical analysis of the DGM, the development of a viable, attractive, competitive, and ultimately superior DG method over the more mature and well-established second order methods is relatively an untouched area. This is mainly due to the fact that the DGM have a number of weaknesses that have yet to be addressed, before they can be robustly used for flow problems of practical interest in a complex configuration environment. In particular, there are three most challenging and unresolved issues in the DGM: a) how to efficiently discretize diffusion terms required for the Navier-Stokes equations, b) how to effectively control spurious oscillations in the presence of strong discontinuities, and c) how to develop efficient time integration schemes for time accurate and steady-state solutions. Indeed, compared to the finite element methods and finite volume methods, the DG methods require solutions of systems of equations with more unknowns for the same grids. Consequently, these methods have been recognized as expensive in terms of both computational costs and storage requirements.

DG methods are indeed a natural choice for the solution of the hyperbolic equations, such as the compressible Euler equations. However, the DG formulation is far less certain and advantageous for the compressible Navier-Stokes equations, where viscous and heat fluxes exist. A severe difficulty raised by the application of the DG methods to the Navier-Stokes equations is the approximation of the numerical fluxes for the viscous fluxes, that has to properly resolve the discontinuities at the interfaces. Taking a simple arithmetic mean of the solution derivatives from the left and right is inconsistent, be-

cause the arithmetic mean of the solution derivatives does not take in account a possible jump of the solutions. A number of numerical methods have been proposed in the literature, such as those by Bassi and Rebay [21, 22], Cockburn and Shu [23], Baumann and Oden [24], Peraire and Persson [25], and many others. Arnold et al. [26] have analyzed a large class of discontinuous Galerkin methods for second-order elliptic problems in a unified formulation. All these methods have introduced in some way the influence of the discontinuities in order to define correct and consistent diffusive fluxes. Lately, Gassner et al. [27] introduced a numerical scheme based on the exact solution of the diffusive generalized Riemann problem for the discontinuous Galerkin methods. Liu et al. [28], and Luo et al. [29] used a BGK-based DG method to compute numerical fluxes at the interface for the Navier-Stokes equations, which has the ability to include both convection and dissipation effects. Unfortunately, all these methods seem to require substantially more computational effort than the classical continuous finite element methods, which are naturally suited for the discretization of elliptic problems. More recently, van Leer et al. [30–32] proposed a recovery-based DG (rDG) method for the diffusion equation using the recovery principle, which recovers a smooth continuous solution that in the weak sense is indistinguishable from the discontinuous discrete solution.

Dumbser et al. [18–20] have introduced a new family of in-cell recovery DG methods, termed PnPm schemes, where Pn indicates that a piecewise polynomial of degree of n is used to represent a DG solution, and Pm represents a reconstructed polynomial solution of degree of m ($m > n$) that is used to compute the fluxes. The PnPm schemes are designed to enhance the accuracy of the discontinuous Galerkin method by increasing the order of the underlying polynomial solution. The beauty of PnPm schemes is that they provide a unified formulation for both finite volume and DG methods, and contain both classical finite volume and standard DG methods as two special cases of PnPm schemes, and thus allow for a direct efficiency comparison. When $n = 0$, i.e. a piecewise constant polynomial is used to represent a numerical solution, P0Pm is nothing but classical high order finite volume schemes, where a polynomial solution of degree m ($m \geq 1$) is reconstructed from a piecewise constant solution. When $m = n$, the reconstruction reduces to the identity operator, and PnPn scheme yields a standard DG method. Obviously, the construction of an accurate and efficient reconstruction operator is crucial to the success of the PnPm schemes. Normally, this is achieved using a so-called in-cell recovery similar to the inter-cell recovery originally proposed by Van Leer et al. [30–32], where recovered equations are obtained using a L_2 projection, i.e., the recovered polynomial solution is uniquely determined by making it indistinguishable from the underlying DG solutions in the contributing cells in the weak sense. This recovery-based discontinuous Galerkin PnPm schemes are termed rDG(PnPm) in this paper, where the lower case r indicates that a higher order polynomial solution of degree m is obtained using a recovery principle, i.e., a weak interpolation. Nourgaliev et al. [33] have shown that in 1D, the resulting recovery-based DG method using piecewise-constant approximation rDG(P0P2) is nothing but FV-PPM method [34], linear rDG(P1P5) is 6^{th} order accurate, quadratic rDG(P2P8) is 9^{th}-order accurate, and cubic rDG(P3P11) 12^{th}-order accurate, versus the 2^{nd}, 3^{rd}, and

4^{th}-order accuracy of the underlying DG method, while keeping the same number of the degrees of freedom and being compact. This recovery-based DG method has been successfully extended to 2D problems on quadrilateral grids. However, the resulting rDG methods are not completely satisfactory, since the stencils in the recovery have to involve the vertex-neighboring cells and thus destroy the compactness of the underlying DG method. For instance, in the case of rDG(P0Pm) recovery, a quadratic polynomial solution ($m=2$) in a cell can be fully recovered using piecewise constant solutions at that cell and its two neighbors in 1D. However, a fully quadratic polynomial has six degrees of freedom, and thus requires six cells in order to recover a quadratic solution in 2D. Unfortunately, there are only five cells available on quadrilateral grids and four cells on triangular grids, when only face-neighboring cells are used in the recovery. Clearly, most of appealing features possessed by the rDG method are lost for the multidimensional problems, and especially on unstructured arbitrary grids. The key issue is how to judiciously choose a proper form of a recovered polynomial and a set of contributing cells in such a way that the resulting recovered linear system is well conditioned, and thus can be inverted. Instead of attempting to recover a full polynomial solution that has the same number of degree of freedom as the number of recovered equations, Dumbser et al. only recover a polynomial solution of a reduced order that has less number of the degrees of freedom than the number of the recovered equations. The resultant over-determined system is then solved using a constraint least-squares method, which guarantees exact conservation, not only of the cell averages but also of all higher order moments in the reconstructed cell itself, such as slopes and curvatures. However, this conservative least-squares recovery approach is computationally expensive, as it involves both recovery of a polynomial solution of higher order and least-squares solution of the resulting over-determined system. Fortunately, recovery is not the only way to obtain a polynomial solution of higher order from the underlying discontinuous Galerkin solutions. Rather, reconstruction widely used in the finite volume methods provides an alternative, probably a better choice to obtain a higher-order polynomial representation. More recently, Zhang et al. [43] presented a class of hybrid DG/FV methods for the conservation laws, where the second derivatives in a cell are obtained from the first derivatives in the cell itself and its neighboring cells using a Green-Gauss reconstruction widely used in the finite volume methods. This provides a fast, simple, and robust way to obtain a higher-order polynomial solutions.

The objective of the effort discussed in this paper is to develop a Reconstructed Discontinuous Galerkin method, termed RDG(P1P2) in short, using a Taylor basis [13] for the solution of the compressible Euler equations on arbitrary grids, where the upper case R denotes Reconstruction, different from lower case r for Recovery. Similar to the rDG methods, higher order accuracy can be expected due to a higher order representation of the reconstructed polynomial solution. Unlike the rDG methods which use a recovery principle (weak interpolation) to obtain a higher order polynomial solution, the RDG methods use a strong interpolation, requiring point values and derivatives to be interpolated, to obtain a higher order polynomial solution. The resulting over-determined linear

system of equations is then solved in the least-squares sense. This reconstruction scheme only involves the von Neumann neighborhood, and thus is compact, simple, robust, and flexible. Furthermore, the reconstruction scheme guarantees exact conservation, not only of the cell averages but also of their slopes due to a judicious choice of our Taylor basis. As the underlying DG method is second-order, and the basis functions are at most linear functions, fewer quadrature points are then required for both domain and face integrals, and the number of unknowns (the number of degrees of freedom) remains the same as for the DG(P1). Consequently, this RDG method is expected to be more efficient than the third order DG method. The developed RDG method is used to compute a variety of flow problems on arbitrary meshes to demonstrate its accuracy, efficiency, and robustness. The numerical results indicate that this RDG method is able to obtain a third-order accurate solution at a slightly higher cost than its second-order DG method and provide a significant increase in performance over the third order DG method in terms of computing costs and storage requirements. The developed RDG method has also successfully been extended to discretize the viscous and heat fluxes in the Navier-Stokes equations using the so-called "inter-cell reconstruction" [35,38]. The remainder of this paper is organized as follows. The governing equations are listed in Section 2. The underlying reconstructed discontinuous Galerkin method is presented in Section 3. Extensive numerical experiments are reported in Section 4. Concluding remarks are given in Section 5.

2 Governing equations

The Euler equations governing unsteady compressible inviscid flows can be expressed as

$$\frac{\partial \mathbf{U}(x,t)}{\partial t}+\frac{\partial \mathbf{F}_k(\mathbf{U}(x,t))}{\partial x_k}=0, \tag{2.1}$$

where the summation convention has been used. The conservative variable vector $\mathbf{U}$, and inviscid flux vector $\mathbf{F}$ are defined by

$$\mathbf{U}=\begin{pmatrix}\rho\\ \rho u_i\\ \rho e\end{pmatrix}, \qquad \mathbf{F}_j=\begin{pmatrix}\rho u_j\\ \rho u_i u_j+p\delta_{ij}\\ u_j(\rho e+p)\end{pmatrix}. \tag{2.2}$$

Here ρ, p, and e denote the density, pressure, and specific total energy of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. The pressure can be computed from the equation of state

$$p=(\gamma-1)\rho\left(e-\frac{1}{2}u_j u_j\right) \tag{2.3}$$

which is valid for perfect gas, where γ is the ratio of the specific heats.

3 Reconstructed discontinuous Galerkin method

The governing equation (2.1) is discretized using a discontinuous Galerkin finite element formulation. To formulate the discontinuous Galerkin method, we first introduce the following weak formulation, which is obtained by multiplying the above conservation law by a test function $\mathbf{W}$, integrating over the domain Ω, and then performing an integration by parts,

$$\int_\Omega \frac{\partial \mathbf{U}}{\partial t}\mathbf{W}d\Omega+\int_\Gamma \mathbf{F}_k\mathbf{n}_k d\Gamma-\int_\Omega \mathbf{F}_k\frac{\partial \mathbf{W}}{\partial x_k}d\Omega=0,\qquad \forall \mathbf{W}\in V, \tag{3.1}$$

where $\Gamma(=\partial\Omega)$ denotes the boundary of Ω, and $\mathbf{n}_j$ the unit outward normal vector to the boundary. We assume that the domain Ω is subdivided into a collection of non-overlapping elements Ω_e, which can be triangles, quadrilaterals, polygons, or their combinations in 2D and tetrahedra, prisms, pyramids, and hexahedra or their combinations in 3D. We introduce the following broken Sobolev space V_p^h

$$V_h^p=\left\{v_h\in[L_2(\Omega)]^m:\ v_h|_{\Omega_e}\in\left[V_p^m\right]\forall\Omega_e\in\Omega\right\}, \tag{3.2}$$

which consists of discontinuous vector-values polynomial functions of degree p, and where m is the dimension of the unknown vector and

$$V_p^m=\text{span}\ \left\{\prod x_i^{\alpha_i}:\ 0\leqslant\alpha_i\leqslant p,\ 0\leqslant i\leqslant d\right\}, \tag{3.3}$$

where α denotes a multi-index and d is the dimension of space. Then, we can obtain the following semi-discrete form by applying weak formulation on each element Ω_e

Find $\mathbf{U}_h\in V_h^p$ such as

$$\frac{d}{dt}\int_{\Omega_e}\mathbf{U}_h\mathbf{W}_h d\Omega+\int_{\Gamma_e}\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k\mathbf{W}_h d\Gamma-\int_{\Omega_e}\mathbf{F}_k(\mathbf{U}_h)\frac{\partial \mathbf{W}_h}{\partial x_k}d\Omega=0,\qquad \forall\mathbf{W}_h\in V_h^p, \tag{3.4}$$

where $\mathbf{U}_h$ and $\mathbf{W}_h$ represent the finite element approximations to the analytical solution $\mathbf{U}$ and the test function $\mathbf{W}$ respectively, and they are approximated by piecewise-polynomial functions of degrees p, which are discontinuous between the cell interfaces. Assume that B is the basis of polynomial function of degrees p, this is then equivalent to the following system of N equations,

$$\frac{d}{dt}\int_{\Omega_e}\mathbf{U}_h B_i d\Omega+\int_{\Gamma_e}\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k B_i d\Gamma-\int_{\Omega_e}\mathbf{F}_k(\mathbf{U}_h)\frac{\partial B_i}{\partial x_k}d\Omega=0,\qquad 1\leqslant i\leqslant N, \tag{3.5}$$

where N is the dimension of the polynomial space. Since the numerical solution $\mathbf{U}_h$ is discontinuous between element interfaces, the interface fluxes are not uniquely defined. The flux function $\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k$ appearing in the second terms of Eq. (3.5) is replaced by a

numerical Riemann flux function $\mathbf{H}_k(\mathbf{U}_h^L,\mathbf{U}_h^R,\mathbf{n}_k)$ where $\mathbf{U}_h^L$ and $\mathbf{U}_h^R$ are the conservative state vector at the left and right side of the element boundary. This scheme is called discontinuous Galerkin method of degree p, or in short notation DG(P) method. Note that discontinuous Galerkin formulations are very similar to finite volume schemes, especially in their use of numerical fluxes. Indeed, the classical first-order cell-centered finite volume scheme exactly corresponds to the DG(P0) method, i.e., to the discontinuous Galerkin method using a piecewise-constant polynomial. Consequently, the DG(Pk) methods with $k>0$ can be regarded as a natural generalization of finite volume methods to higher order methods. By simply increasing the degree P of the polynomials, the DG methods of corresponding higher order are obtained.

The domain and boundary integrals in Eq. (3.5) are calculated using Gauss quadrature formulas. The number of quadrature points used is chosen to integrate exactly polynomials of order of $2p$ on the reference element. In 2D, two, three, and four points are used for linear, quadratic, and cubic basis function in the boundary integrals. The domain integrals are evaluated using three, six, and thirteen points for triangular elements and four, nine, and sixteen points for quadrilateral elements, respectively.

In the traditional DGM, numerical polynomial solutions $\mathbf{U}_h$ in each element are expressed using either standard Lagrange finite element or hierarchical node-based basis as following

$$\mathbf{U}_h=\sum_{i=1}^{N}\mathbf{U}_i(t)B_i(x), \tag{3.6}$$

where B_i are the finite element basis functions. As a result, the unknowns to be solved for are the variables at the nodes $\mathbf{U}_i$, as illustrated in Fig. 1 for linear and quadratic polynomial approximations.

Figure 1: Representation of polynomial solutions using finite element shape functions.

On each cell, a system of $N\times N$ has to be solved, where the polynomial solutions are dependent on the shape of elements. For example, for a linear polynomial approximation in 2D as shown in Fig. 1, a linear polynomial is used for triangular elements and the unknowns to be solved for are the variables at three vertices. For quadrilateral elements, a bi-linear polynomial is used, and the unknowns to be solved for are the variables at four vertices. However, numerical polynomial solutions $\mathbf{U}$ can be expressed in other forms as well. In the present work, the numerical polynomial solutions are represented using a Taylor series expansion at the center of the cell. For example, if we do a Taylor series

expansion at the centroid of the cell, the quadratic polynomial solutions can be expressed as follows

$$\begin{aligned}\mathbf{U}_h=&\mathbf{U}_c+\left.\frac{\partial \mathbf{U}}{\partial x}\right|_c(x-x_c)+\left.\frac{\partial \mathbf{U}}{\partial y}\right|_c(y-y_c)+\left.\frac{\partial^2}{\mathbf{U}}\partial x^2\right|_c\frac{(x-x_c)^2}{2}\\&+\left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_c\frac{(y-y_c)^2}{2}+\left.\frac{\partial^2 \mathbf{U}}{\partial x\partial y}\right|_c(x-x_c)(y-y_c),\end{aligned}\tag{3.7}$$

which can be further expressed as cell-averaged values and their derivatives at the center of the cell:

$$\begin{aligned}\mathbf{U}_h=&\tilde{\mathbf{U}}+\left.\frac{\partial \mathbf{U}}{\partial x}\right|_c(x-x_c)+\left.\frac{\partial \mathbf{U}}{\partial y}\right|_c(y-y_c)+\left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_c\left(\frac{(x-x_c)^2}{2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(x-x_c)^2}{2}d\Omega\right)\\&+\left.\frac{\partial^2}{\mathbf{U}}\partial y^2\right|_c\left(\frac{(y-y_c)^2}{2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(y-y_c)^2}{2}d\Omega\right)\\&+\left.\frac{\partial^2}{\mathbf{U}}\partial x\partial y\right|_c\left((x-x_c)(y-y_c)-\frac{1}{\Omega_e}\int_{\Omega_e}(x-x_c)(y-y_c)d\Omega\right),\end{aligned}\tag{3.8}$$

where $\tilde{\mathbf{U}}$ is the mean value of $\mathbf{U}$ in this cell. The unknowns to be solved for in this formulation are the cell-averaged variables and their derivatives at the center of the cells, regardless of element shapes, as shown in Fig. 2.

Figure 2: Representation of the polynomial solutions using a Taylor series expansion for a cell-centered scheme (left) and vertex-centered scheme (right).

In this case, the dimension of the polynomial space is six and the six basis functions are

$$B_1=1,\qquad B_2=x-x_c,\qquad B_3=y-y_c,\tag{3.9a}$$

$$B_4=\frac{(x-x_c)^2}{2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(x-x_c)^2}{2}d\Omega,\qquad B_5=\frac{(y-y_c)^2}{2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(y-y_c)^2}{2}d\Omega,\tag{3.9b}$$

$$B_6=(x-x_c)(y-y_c)-\frac{1}{\Omega_e}\int_{\Omega_e}(x-x_c)(y-y_c)d\Omega.\tag{3.9c}$$

The discontinuous Galerkin formulation then leads to the following six equations

$$\frac{d}{dt}\int_{\Omega_e}\tilde{\mathbf{U}}d\Omega+\int_{\Gamma_e}\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k d\Gamma=0,\qquad i=1, \tag{3.10a}$$

$$M_{5\times5}\frac{d}{dt}\left(\left.\frac{\partial\mathbf{U}}{\partial x}\right|_c\quad\left.\frac{\partial\mathbf{U}}{\partial y}\right|_c\quad\left.\frac{\partial^2}{\mathbf{U}}\partial x^2\right|_c\quad\left.\frac{\partial^2}{\mathbf{U}}\partial y^2\right|_c\quad\left.\frac{\partial^2}{\mathbf{U}}\partial x\partial y\right|_c\right)^T+R_{5\times1}=0. \tag{3.10b}$$

Note that in this formulation, equations for the cell-averaged variables are decoupled from equations for their derivatives due to the judicial choice of the basis functions and the fact that

$$\int_{\Omega_e}B_1B_i d\Omega=0,\qquad 2\leqslant i\leqslant 6. \tag{3.11}$$

In the implementation of this DG method, the basis functions are actually normalized in order to improve the conditioning of the system matrix (3.5) as follows:

$$B_1=1,\qquad B_2=\frac{x-x_c}{\Delta x},\qquad B_3=\frac{y-y_c}{\Delta y}, \tag{3.12a}$$

$$B_4=\frac{(x-x_c)^2}{2\Delta x^2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(x-x_c)^2}{2\Delta \mathrm{x}^2}d\Omega,\qquad B_5=\frac{(y-y_c)^2}{2\Delta \mathrm{y}^2}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(y-y_c)^2}{2\Delta \mathrm{y}^2}d\Omega, \tag{3.12b}$$

$$B_6=\frac{(x-x_c)(y-y_c)}{\Delta x\Delta y}-\frac{1}{\Omega_e}\int_{\Omega_e}\frac{(x-x_c)(y-y_c)}{\Delta x\Delta y}d\Omega, \tag{3.12c}$$

where $\Delta x=0.5(x_{\max}-x_{\min})$, and $\Delta y=0.5(y_{\max}-y_{\min})$, and $x_{\max}$, $x_{\min}$, $y_{\max}$, and $y_{\min}$ are the maximum and minimum coordinates in the cell Ω_e in x-, and y-directions, respectively. A quadratic polynomial solution can then be rewritten

$$\mathbf{U}_h=\tilde{\mathbf{U}}B_1+\mathbf{U}_xB_2+\mathbf{U}_yB_3+\mathbf{U}_{xx}B_4+\mathbf{U}_{yy}B_5+\mathbf{U}_{xy}B_6, \tag{3.13}$$

where

$$\mathbf{U}_\mathbf{x}=\frac{\partial\mathbf{U}}{\partial x}|_c\Delta x,\qquad \mathbf{U}_y=\frac{\partial\mathbf{U}}{\partial y}|_c\Delta y, \tag{3.14a}$$

$$\mathbf{U}_{xx}=\frac{\partial^2\mathbf{U}}{\partial x^2}|_c\Delta x^2,\qquad \mathbf{U}_{yy}=\frac{\partial^2\mathbf{U}}{\partial y^2}|_c\Delta y^2,\qquad \mathbf{U}_{xy}=\frac{\partial^2\mathbf{U}}{\partial x\partial y}|_c\Delta x\Delta y. \tag{3.14b}$$

This formulation allows us to clearly see the similarities and differences between the DG and FV methods. In fact, the discretized governing equations for the cell-averaged variables and the assumption of the polynomial solutions on each cell are exactly the same for both methods. The only difference between them is the way how they obtain high-order (>1) polynomial solutions. In the finite volume methods, the polynomial solutions of degree p are reconstructed using information from the cell-averaged values of the flow

variables, which can be obtained using either TVD/MUSCL or ENO/WENO reconstruction schemes. Unfortunately, the multi-dimensional MUSCL approach suffers from two shortcomings in the context of unstructured grids: 1) Uncertainty and arbitrariness in choosing the stencils and methods to compute the gradients in the case of linear reconstruction; This explains why a nominally second-order finite volume scheme is hardly able to deliver a formal solution of the second order accuracy in practice for unstructured grids. The situation becomes even more evident, severe, and profound, when a highly stretched tetrahedral grid is used in the boundary layers. Many studies, as reported by many researchers [36–38] have demonstrated that it is difficult to obtain a second-order accurate flux reconstruction on highly stretched tetrahedral grids and that for the discretization of inviscid fluxes, the classic 1D-based upwind schemes using median-dual finite volume approximation suffer from an excessive numerical diffusion due to such skewing. 2) Extended stencils required for the reconstruction of higher-order ($>1^{\text{st}}$) polynomial solutions. This is exactly the reason why the current finite-volume methods using the TVD/MUSCL reconstruction are not practical at higher order and have remained second-order on unstructured grids. When the ENO/WENO reconstruction schemes are used for the construction of a polynomial of degree p on unstructured grids, the dimension of the polynomial space $N=N(p,d)$ depends on the degree of the polynomials of the expansion p, and the number of spatial dimensions d. One must have three, six, and ten cells in 2D and four, ten, and twenty cells in 3D for the construction of a linear, quadratic, cubic Lagrange polynomial, respectively. Undoubtedly, it is an overwhelmingly challenging task to judiciously choose a set of admissible and proper stencils that have such a large number of cells on unstructured grids especially for higher order polynomials and higher dimensions. Nevertheless, recently, higher-order ENO/WENO methods have been successfully extended on unstructured grids [46–52]. Unlike the FV methods, where the derivatives are reconstructed using cell average values of the neighboring cells, the DG method computes the derivatives in a manner similar to the mean variables. This is compact, rigorous, and elegant mathematically in contrast to the arbitrariness characterizing the reconstruction schemes with respect how to compute the derivatives and how to choose the stencils used in the FV methods. It is our believe that this is one of the main reasons why the second order DG methods are more accurate than the FV methods using either TVD/MUSCL or ENO/WENO reconstruction schemes and are less dependent on the mesh regularity, which has been demonstrated numerically [13]. Furthermore, the higher order DG methods can be easily constructed by simply increasing the degree p of the polynomials locally, in contrast to the finite volume methods which use the extended stencils to achieve higher order of accuracy.

However, in comparison to the reconstructed FV methods, the DG methods have a significant drawback in that they require more degrees of freedom, an additional domain integration, and more Gauss quadrature points for the boundary integration, and therefore more computational costs and storage requirements. On one hand, the reconstruction methods that FV methods use to achieve higher-order accuracy are relatively inexpensive but less accurate and robust. On the other hand, the DG methods that can

be viewed as a different way to extend a FV method to higher orders are accurate and robust but costly. The "reconstructed DG" method to be described below, which can be regarded as a variant of more general PnPm schemes originally introduced by Dumbser et al. [18–20], is our attempt to combine the efficiency of the reconstruction methods and the accuracy of the DG methods. Although our discussion in this work is mainly focused on the linear DG method in 2D, its extension to higher-order and 3D DG methods is straightforward. In the case of DG(P1) method, a linear polynomial solution $\mathbf{U}_i$ in any cell i is

$$\mathbf{U}_i = \tilde{\mathbf{U}}_i + \mathbf{U}_{\mathbf{x}i} B_2 + \mathbf{U}_{\mathbf{y}i} B_3. \tag{3.15}$$

Using this underlying linear polynomial DG solution in the neighboring cells, one can reconstruct a quadratic polynomial solution $\mathbf{U}_i^R$ as follows:

$$\mathbf{U}_i^R = \tilde{\mathbf{U}}_i^R + \mathbf{U}_{xi}^R B_2 + \mathbf{U}_{yi}^R B_3 + \mathbf{U}_{xxi}^R B_4 + \mathbf{U}_{yyi}^R B_5 + \mathbf{U}_{xyi}^R B_6. \tag{3.16}$$

In order to maintain the compactness of the DG methods, the reconstruction is required to involve only von Neumann neighborhood, i.e., the adjacent cells that share a face with the cell i under consideration. There are six degrees of freedom, and therefore 6 unknowns must be determined. The first three unknowns can be trivially obtained, by requiring the consistency of the RDG with the underlying DG: 1) The reconstruction scheme must be conservative, and 2) The values of the reconstructed first derivatives are equal to the ones of the first derivatives of the underlying DG solution at the centroid i. Due to the judicious choice of Taylor basis in our DG formulation, these three degrees of freedom simply coincide with the ones from the underlying DG solution, i.e.,

$$\tilde{\mathbf{U}}_i^R = \tilde{\mathbf{U}}_i, \qquad \mathbf{U}_{xi}^R = \mathbf{U}_{xi}, \qquad \mathbf{U}_{yi}^R = \mathbf{U}_{yi}. \tag{3.17}$$

As a result, only three second derivatives need to be determined. This can be accomplished by requiring that the point-wise values and first derivatives of the reconstructed solution and of the underlying DG solution are equal at the cell centers for all the adjacent face neighboring cells. Consider a neighboring cell j, one requires

$$\mathbf{U}_j = \tilde{\mathbf{U}}_i + \mathbf{U}_{xi} B_2 + \mathbf{U}_{yi} B_3 + \mathbf{U}_{xxi}^R B_4 + \mathbf{U}_{yyi}^R B_5 + \mathbf{U}_{xyi}^R B_6, \tag{3.18a}$$

$$\left.\frac{\partial \mathbf{U}}{\partial x}\right|_j = \mathbf{U}_{xi}\frac{1}{\Delta x_i} + \mathbf{U}_{xxi}^R \frac{B_2}{\Delta x_i} + \mathbf{U}_{xyi}^R \frac{B_3}{\Delta x_i}, \tag{3.18b}$$

$$\left.\frac{\partial \mathbf{U}}{\partial y}\right|_j = \mathbf{U}_{yi}\frac{1}{\Delta y_i} + \mathbf{U}_{yyi}^R \frac{B_3}{\Delta y_i} + \mathbf{U}_{xyi}^R \frac{B_2}{\Delta y_i}, \tag{3.18c}$$

where the basis functions B are evaluated at the center of cell j, i.e., $B = \mathbf{B}(x_j, y_j)$. This can be written in a matrix form as follows:

$$\begin{pmatrix} B_4^j & B_5^j & B_6^j \\ B_2^j & 0 & B_3^j \\ 0 & B_3^j & B_2^j \end{pmatrix} \begin{pmatrix} \mathbf{U}_{xxi}^R \\ \mathbf{U}_{yyi}^R \\ \mathbf{U}_{xyi}^R \end{pmatrix} = \begin{pmatrix} \mathbf{U}_j - (\mathbf{U}_i B_1^j + \mathbf{U}_{xi} B_2^j + \mathbf{U}_{yi} B_3^j) \\ \frac{\Delta x_i}{\Delta x_j}\mathbf{U}_{xj} - \mathbf{U}_{xi} \\ \frac{\Delta y_i}{\Delta y_j}\mathbf{U}_{yj} - \mathbf{U}_{yi} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1^j \\ \mathbf{R}_2^j \\ \mathbf{R}_3^j \end{pmatrix}, \tag{3.19}$$

where **R** is used to represent the right-hand-side for simplicity. Similar equations can be written for all cells connected to the cell i with a common face, which leads to a non-square matrix. The number of face-neighboring cells for a triangular and a quadrilateral cell is three and four, respectively. Correspondingly, the size of the resulting non-square matrix is 9×3 and 12×3, respectively. This over-determined linear system of 9 or 12 equations for 3 unknowns can be solved in the least-squares sense. In the present work, it is solved using a normal equation approach, which, by pre-multiplying through by the matrix transpose, yields a symmetric linear 3×3 system of equations as follows

$$\begin{pmatrix} \sum_j (B_4^j B_4^j + B_2^j B_2^j) & \sum_j B_4^j B_5^j & \sum_j (B_4^j B_6^j + B_2^j B_3^j) \\ \sum_j B_4^j B_5^j & \sum_j (B_5^j B_5^j + B_3^j B_3^j) & \sum_j (B_5^j B_6^j + B_2^j B_3^j) \\ \sum_j (B_4^j B_6^j + B_2^j B_3^j) & \sum_j (B_5^j B_6^j + B_2^j B_3^j) & \sum_j (B_6^j B_6^j + B_2^j B_2^j + B_3^j B_3^j) \end{pmatrix} \begin{pmatrix} \mathbf{U}_{xxi}^R \\ \mathbf{U}_{yyi}^R \\ \mathbf{U}_{xyi}^R \end{pmatrix}$$
$$= \begin{pmatrix} \sum_j (B_4^j \mathbf{R}_1^j + B_2^j \mathbf{R}_2^j) \\ \sum_j (B_5^j \mathbf{R}_1^j + B_3^j \mathbf{R}_3^j) \\ \sum_j (B_6^j \mathbf{R}_1^j + B_3^j \mathbf{R}_2^j + B_2^j \mathbf{R}_3^j) \end{pmatrix}. \tag{3.20}$$

This linear system of 3×3 can be then trivially solved to obtain the second derivatives of the reconstructed quadratic polynomial solution.

This reconstructed quadratic polynomial solution is then used to compute the domain and boundary integrals of the underlying DG(P1) method in Eq. (3.5). The resulting DG method, termed a "reconstructed DG" method (RDG(P1P2) in short notation), is expected to have the third order of accuracy at a moderate increase of computing costs in comparison to the underlying DG(P1) method. The extra costs are mainly due to the least-squares reconstruction, which is relatively cheap in comparison to the evaluation of fluxes, and an extra Gauss quadrature point, which is required to calculate the domain integrals for the triangular element (four quadrature points). Like in the DG(P1), two quadrature points are used to calculate the boundary integrals, and four points are used to calculate the domain integrals for quadrilateral elements. In comparison to DG(P2), this represents a significant saving in terms of flux evaluations. Furthermore, the number of degrees of freedom is considerably reduced, which leads to a significant reduction in memory requirements, and from which implicit methods will benefit tremendously. The cost analysis for the FV(P1), DG(P1), RDG(P1P2) and DG(P2) is summarized in Table 1, where the memory requirement for storing only the implicit diagonal matrix is given as well, and which grows quadratically with the order of the DG methods. We would like to emphasize that the storage requirements for the implicit DG methods are extremely demanding, especially for higher-order DG methods.

This reconstructed DG method has been implemented in a well-tested, fully-verified 2D DG code [13–17, 29]. In this code, a fast, low-storage p-multigrid method [16, 17] is

Table 1: Cost analysis for different numerical methods in 2D.

	FV(P1)	DG(P1)	RDG(P1P2)	DG(P2)
Number of quadrature points for boundary integrals	1	2	2	3
Number of quadrature points for domain integrals	0	3(triangle) 4(quadrilateral)	4 4	6 9
Reconstruction	Yes	No	Yes	No
Order of Accuracy	$\mathcal{O}(h^2)$	$\mathcal{O}(h^2)$	$\mathcal{O}(h^3)$	$\mathcal{O}(h^3)$
Storage for Implicit Diagonal Matrix	25 words Per element	225	225	900

developed to obtain steady state solutions, and an explicit three-stage third-order TVD Runge-Kutta scheme is used to advance solution in time for the unsteady flow problems. Many upwind schemes have been implemented for the discretization of the inviscid fluxes, although HLLC scheme is exclusively used for the approximate solution of the Riemann problem in this work.

4 Numerical examples

All of the computations are performed on a Dell XPS M1210 laptop computer (2.33 GHz Intel(R) Core(TM) 2 CPU T7600 with 4GBytes memory) using a Suse 11.0 Linux operating system. A well-tested finite volume code [39–44] is used as a reference to compare the accuracy and performance of the DG method for some test cases. All the unstructured hybrid grids are generated using the method described in References [44] and [45]. All the test cases are chosen to demonstrate that the developed RDG(P1P2) method is able to maintain the accuracy and robustness of the underlying DG method.

4.1 Sod shock tube problem

The shock tube problem constitutes a particularly interesting and difficult test case, since it represents an exact solution to the full system of one-dimensional Euler equations containing simultaneously a shock wave, a contact discontinuity, and an expansion fan. This test case is chosen to demonstrate that the reconstructed DG method is able to maintain the robustness of the underlying DG methods, as the reconstruction scheme used in the finite volume methods will unavoidably generate negative values for density and energy in this case, leading to a breakdown of the finite volume solution process. The initial conditions in the present computation are the following: $\rho=1$, $u=0$, $p=1$ for $0\leq x\leq 0.5$, and $\rho=0.125$, $u=0$, $p=0.1$ for $0.5\leq x\leq 1$. Fig. 3 shows a comparison of the density profiles obtained by the unlimited DG(P0), DG(P1), DG(P2), and RDG(P1P2) and the exact solutions, respectively. This is a 2D simulation of a 1D problem. The mesh consists of 50 cells in the X-direction and 1 cell in the Y-direction. As expected, unlimited higher or-

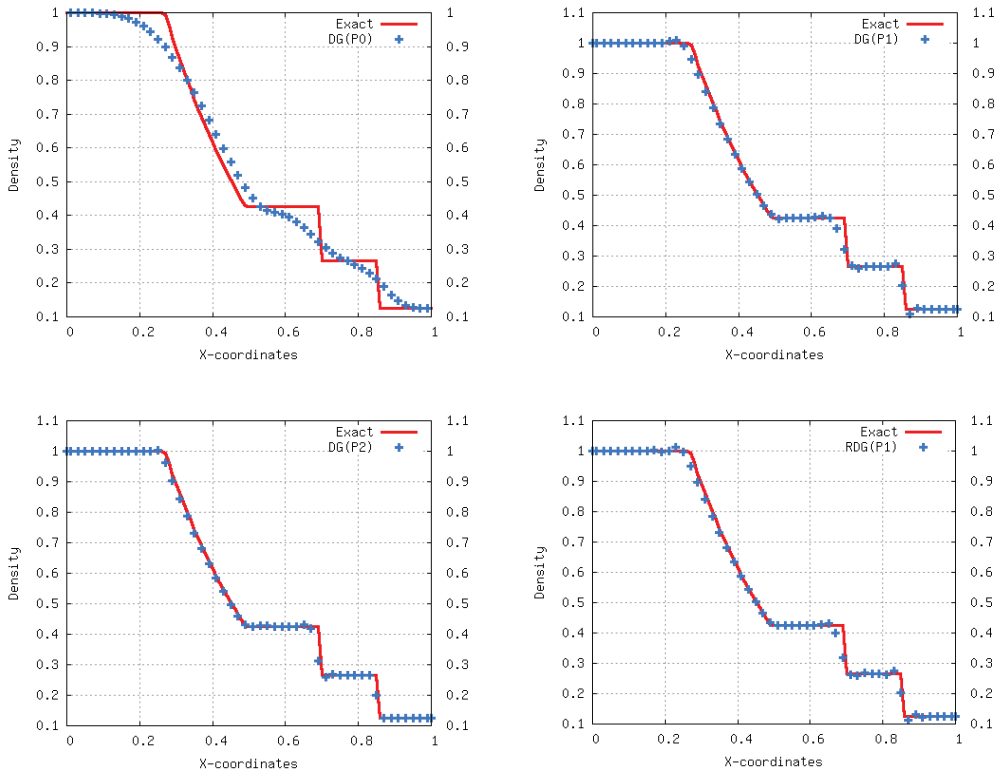

Figure 3: Comparison of the computed density profile for Sod shock tube problem obtained by unlimited DG(P0), DG(P1), DG(P2), and RDG(P1P2) solutions with the analytical solution.

der DG(P1), DG(P2), and RDG(P1P2) solutions exhibit small oscillations in the vicinity of discontinuities and higher order DG solution yields a sharper resolution for both contact discontinuity and shock wave than a lower order DG solution. The RDG(P1P2) solution is more accurate than the DG(P1) solution, although not as accurate as DG(P2) solution. The reconstruction scheme used in the DG method does not create negative density, and thus does not lead to the breakdown of the solution process. This can be attributed to the robustness of the underlying DG(P1) method upon which a reconstruction is based.

4.2 Subsonic flows past a circular cylinder

This is a well-known test case: subsonic flow past a circular cylinder at a Mach number of $M_\infty = 0.38$. This test case is chosen to verify if a formal order of the convergence rate of the RDG method can be achieved for the compressible Euler equations on unstructured grids. Fig. 4 shows four successively refined o-type grids having 16×5, 32×9, 64×17, and 128×33 points, respectively. The first number is the number of points in the angular direction, and the second number is the number of points in the radial direction. The

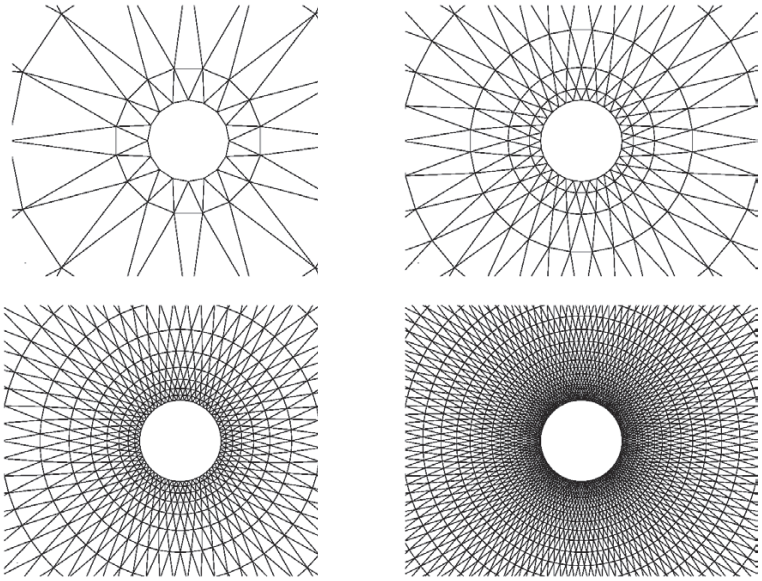

Figure 4: Sequences of four successively globally refined triangular meshes 16×5, 32×9, 64×17, 128×33 for computing subsonic flow past a circular cylinder.

radius of the cylinder is $r_1=0.5$, the domain is bounded by $r_{33}=20$, and the radii of concentric circles for 128×33 mesh are set up as

$$r_i=r_1\left(1+\frac{2\pi}{128}\sum_{j=0}^{i-1}\alpha_j\right),\qquad i=2,\cdots,33,$$

where $\alpha=1.1580372$. The coarser grids are generated by successively coarsing the finest mesh. Numerical solutions to this problem are computed using FV(P1), DG(P1), DG(P2), and RDG(P1P2) methods on these four grids to obtain quantitative measurement of the order of accuracy and discretization errors. The detailed results for this test case are presented in Tables 2-5. They show the mesh size, the number of degrees of freedom, the L_2-error of the solutions, and the order of convergence. In this case, the following entropy production ε defined as

$$\varepsilon=\frac{S-S_\infty}{S_\infty}=\frac{p}{p_\infty}\left(\frac{\rho_\infty}{\rho}\right)^\gamma-1$$

is served as the error measurement, where S is the entropy. Note that the entropy production is a very good criterion to measure accuracy of the numerical solutions, since the flow under consideration is isentropic. Fig. 5 shows the computed Mach number contours in the flow field obtained by FV(P1) on the 128×33 mesh, DG(P1) on the 64×17 mesh, and DG(P2) and RDG(P1P2) on the 32×8 mesh, respectively. One can see that the

Table 2: Subsonic circular cylinder test case: reconstructed FV(P1) is order of $\mathcal{O}(h^2)$.

Mesh	No. DOFs	L2-error	Order
16×5	80	2.37148E-01	-
32×9	288	7.76551E-01	1.595
64×17	1,088	1.36962E-02	2.551
128×33	4,224	3.54568E-03	1.951

Table 3: Subsonic circular cylinder test case: DG(P1) is order of $\mathcal{O}(h^2)$.

Mesh	No. DOFs	L2-error	Order
16×5	360	5.68722E-02	-
32×9	1,536	1.07103E-02	2.443
64×17	6,144	1.67302E-03	2.688
128×33	24,576	2.34369E-04	2.838

Table 4: Subsonic circular cylinder test case: DG(P2) is order of $\mathcal{O}(h^3)$.

Mesh	No. DOFs	L2-error	Order
16×5	768	8.40814E-03	-
32×9	3,072	5.26017E-04	4.055
64×17	12,288	4.48952E-05	3.536
128×33	49,152	4.16294E-06	3.434

Table 5: Subsonic circular cylinder test case: RDG(P1) is order of $\mathcal{O}(h^3)$.

Mesh	No. DOFs	L2-error	Order
16×5	360	1.91161E-02	-
32×9	1,536	9.72523E-04	4.358
64×17	6,144	8.00571E-05	3.615
128×33	24,576	9.33899E-06	3.102

results obtained by DG(P2) on the 32×8 mesh are more accurate than the ones obtained by DG(P1) on the 64×17 mesh, which in turn are more accurate than the ones obtained by FV(P1) on the 128×33 mesh. Both RDG(P1P2) and DG(P2) solutions are virtually identical for this case. However, the DG(P2) does yield a slightly more accurate solution than the RDG(P1P2) at the same grid resolution. This can be seen in Fig. 6, providing the details of the spatial convergence of each method for this numerical experiment. As expected, the DG method exhibits a full $\mathcal{O}(h^{p+1})$ order of convergence. The RDG(P1P2) method does offer a full $\mathcal{O}(h^{p+2})$ order of the convergence, adding one order of accuracy to the underlying DG(P1) method. Fig. 7 illustrates that higher order DG methods require significantly less degrees of freedom than lower order ones for the same accuracy. Moreover, the RDG(P1P2) outperforms DG(P2), by measuring the number of the degrees of freedom required to achieve the same accuracy.

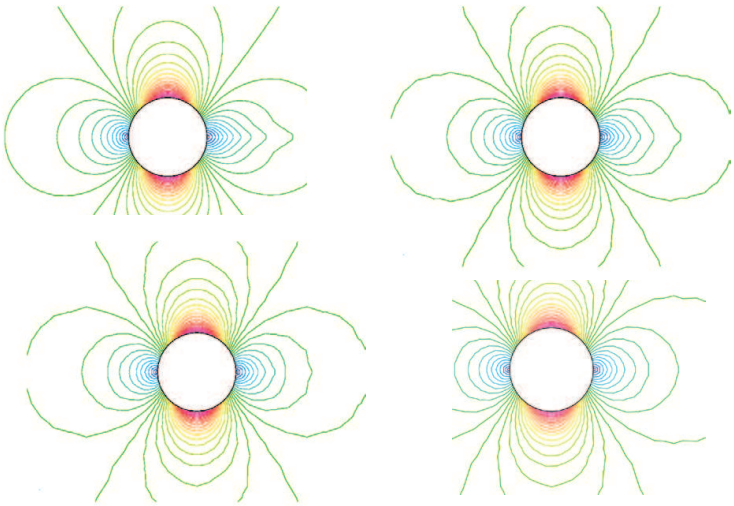

Figure 5: Computed Mach number contours in the flow field obtained by the FV(P1) method on 128×33 mesh (top left), DG(P1) method on 64×17 mesh (top right), DG(P2) method on 32×9 mesh (bottom left), and RDG(P1P2) on 32×9 mesh (bottom right) for subsonic flow past a circular cylinder at $M_\infty=0.38$.

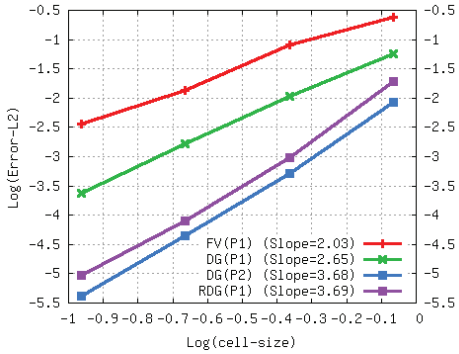

Figure 6: Convergence history for subsonic flow past a circular cylinder for FV(P1), DG(P1), DG(P2), and RDG(P1P2) methods.

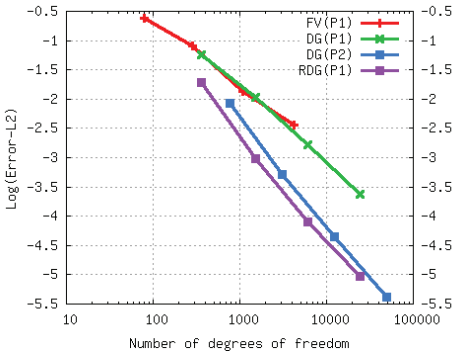

Figure 7: L_2-errors of numerical solutions vs. the number of degrees of freedom for subsonic flow past a circular cylinder by FV(P1), DG(P1), DG(P2), and RDG(P1P2) methods.

4.3 Low Mach number flow past a circular cylinder

This test case is chosen to demonstrate that the RDG method is able to maintain the accuracy of the underlying DG method for solving low Mach number flow problems. The computation performed for low speed flows past a circular cylinder at a Mach number of $M_\infty=0.01$ using both DG(P2) and RDG(P1P2) methods on a hybrid mesh is shown in Fig. 8, where the comparison of the velocity distributions on the surface of the circular cylinder obtained by these two methods, and those of the FV(P1) solution and the analytic solution for incompressible flows are presented as well. One can see clearly dramatic

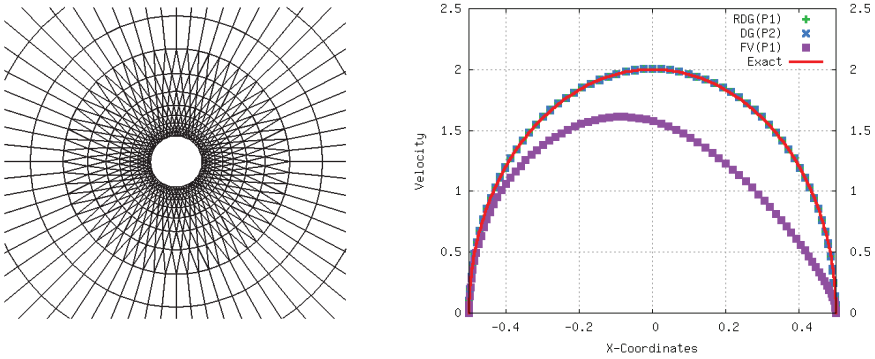

Figure 8: Left: Unstructured hybrid grid (1,024 triangles, 512 quadrilaterals, 1,536 grid points, 128 boundary faces) used for computing low Mach number flow past a circular cylinder. Right: Comparison of the computed velocity distributions on the surface of the circular cylinder obtained by the RDG(P1P2) and DG(P2) methods with the incompressible solution for low Mach number flow past a circular cylinder at $M_\infty = 0.01$.

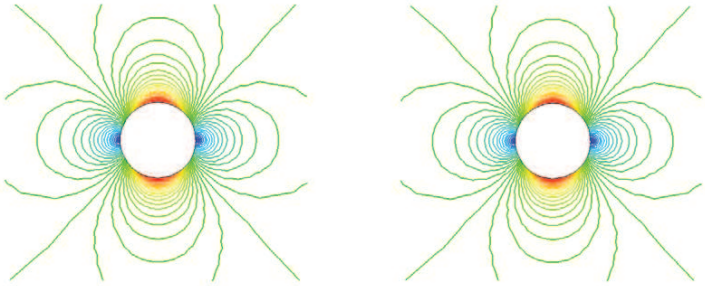

Figure 9: Computed Mach number contours obtained by the RDG(P1P2) (left) and DG(P2) (right) methods for low Mach number flow past a circular cylinder at $M_\infty = 0.01$.

deterioration of the FV(P1) solution. The mesh consists of 1,024 triangular elements, 512 quadrilateral elements, 1,536 grid points, and 128 boundary faces. The computed Mach number contours in the flow field obtained by these two methods are shown in Fig. 9, respectively. The results illustrate that the RDG(P1P2) and the DG(P2) solutions are virtually identical, demonstrating that the RDG(P1P2) method is able to yield an accurate solution, and thus maintain the ability of the underlying DG method for the low Mach number flows. A grid convergence study is also conducted for this test case to verify if a formal order of convergence rate for the RDG(P1P2) method can be achieved for the low Mach number flows on the hybrid triangular and quadrilateral grids. Fig. 10 shows four successively refined hybrid grids having 16×5, 32×9, 64×17, and 128×33 points, respectively. The grid convergence history versus the cell size and the number of degrees of freedom is displayed in Fig. 11, where the convergence history obtained by RDG(P1P2) for the test case 2 is also provided to see the effect of the Mach number on convergence. Again, one observes that the RDG(P1P2) does provide a full 3^{rd} order of convergence for low Mach number flows and on unstructured hybrid grids. What is surprising is that

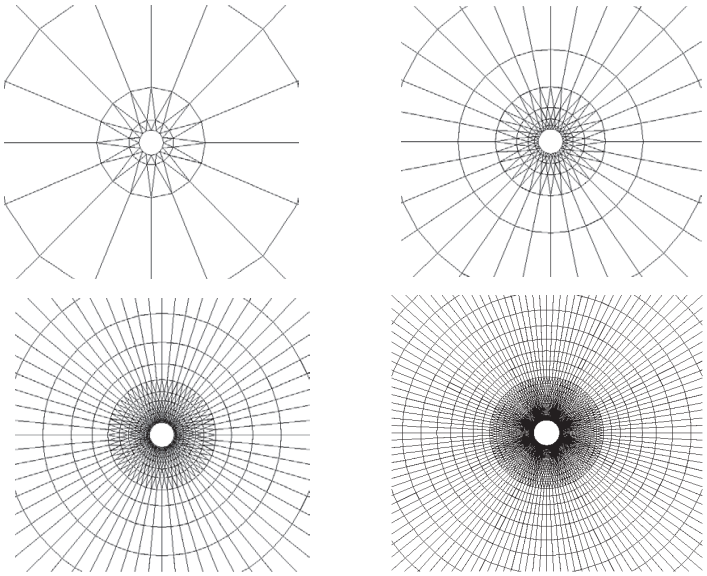

Figure 10: Sequences of four successively globally refined hybrid meshes 16×5, 32×9, 64×17, 128×33 for computing subsonic flow past a circular cylinder.

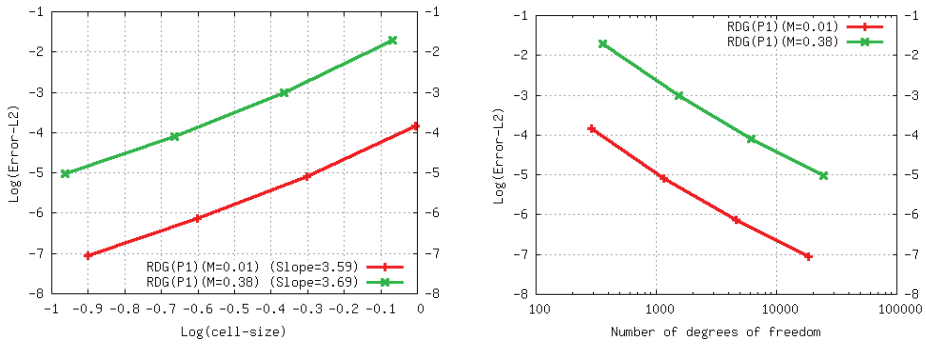

Figure 11: L_2-error of numerical solutions against the cell size (left) and the number of degrees of freedom (right) for low Mach number and subsonic flow past a circular cylinder by the RDG(P1P2) method.

RDG(P1P2) converges faster for low Mach number flows ($M_\infty = 0.01$) than for higher Mach number flows ($M_\infty = 0.38$).

4.4 Subsonic flow past a NACA0012 airfoil

Our next test case involves an inviscid flow past a NACA0012 airfoil at a Mach number of 0.63, and an angle of attack 2°. This numerical experiment is designed to test the ability of the RDG method to obtain highly accurate solutions to the Euler equations on viscous hybrid grids. Being able to produce an accurate inviscid solution on a highly

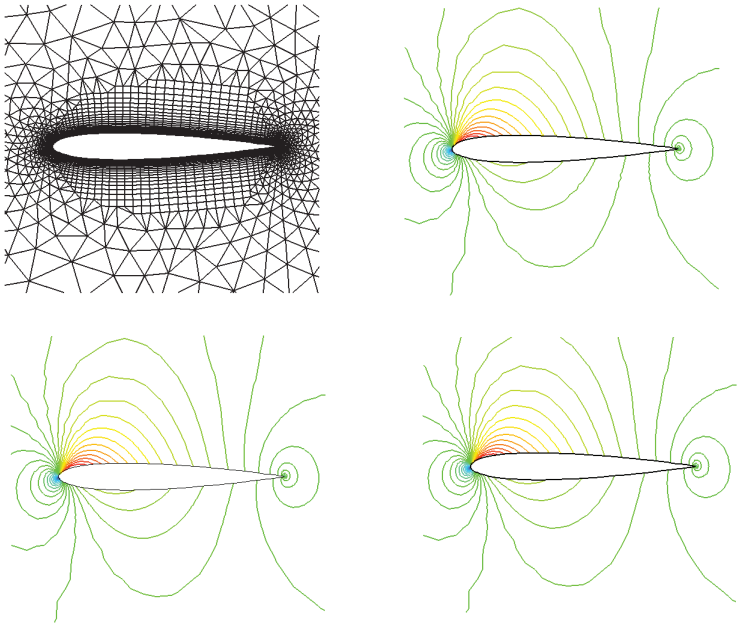

Figure 12: Unstructured hybrid mesh (top left) (1,533 quadrilaterals, 3,469 triangles, 3,346 grid points, 157 boundary faces) and computed Mach number contours by the DG(P1) (top right), the RDG(P1P2) (bottom left), and the DG(P2) (bottom right) methods, respectively for subsonic flow past a NACA0012 airfoil at $M_\infty=0.63$, $\alpha=2^\circ$.

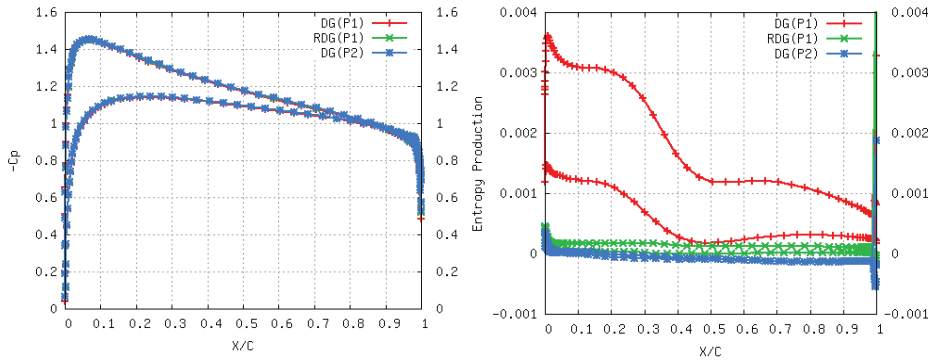

Figure 13: Comparison of the computed pressure coefficient (left) and entropy production (right) distributions for subsonic flow past a NACA0012 airfoil at $M_\infty=0.63$, $\alpha=2^\circ$.

stretched Navier-Stokes grid is extremely difficult and challenging, and yet of utmost importance for the accurate solution to the Navier-Stokes equations, thus serving as a good criterion to measure accuracy and robustness of a numerical method. Many finite volume methods are unable to obtain the same quality of an inviscid solution on an anisotropic Navier-Stokes grid as on an isotropic Euler grid, suffering from either excessive numerical dissipation or spurious oscillations due to a combination of the mesh

irregularity and deficiencies of the reconstruction schemes. Fig. 12 shows the mesh used in this numerical experiment and the computed Mach number contours in the flow field obtained by DG(P1), RDG(P1P2), and DG(P2) methods, respectively. The mesh consists of 1,533 quadrilateral cells, 3,469 triangular cells, 3,346 grid points, and 157 boundary faces. The computed Mach number contours in the flow field obtained using DG(P1), RDG(P1P2), and DG(P2) methods are also shown in Fig. 12, where accurate and smooth solutions are observed in spite of the highly stretched grid used in the boundary layer. The computed pressure coefficient and entropy production distribution on the surface of the airfoil obtained by these three methods are compared in Fig. 13. All three solutions are virtually identical by judging the Mach number contours in the flow field and the pressure coefficient distributions on the surface of the airfoil, indicating that the numerical solution is order-independent, (i.e., a convergence on these flow quantities is reached). The RDG(P1P2) solution however is significantly improved compared with the DG(P1) solution by judging the entropy production distribution on the surface of the airfoil. Note that the entropy production is directly related to the error of the numerical methods, as it should be zero everywhere for subsonic flows. The DG(P2) solution provides a further improvement over the RDG(P1) solution, although the difference is very small. This numerical experiment demonstrates that the RDG(P1P2) method, unlike some of its finite volume counterparts, has the ability to accurately solve the compressible Euler equation on anisotropic grids designed for solving the Navier-Stokes equations.

4.5 Subsonic flow past a three-element airfoil

Finally, an illustrative example is presented to demonstrate that the developed RDG method can be applied to problems of scientific and industrial interests. The computation is performed on a three-element airfoil at a free stream of Mach number of 0.2 and an angle of attack of 10°. The hybrid unstructured Cartesian and triangle grid is shown in Fig. 14, which consists of 13,892 Cartesian cells and 3,981 triangular elements. The computed Mach number and pressure contours in the flow field are shown in Fig. 15, demonstrating the accuracy and robustness of the developed RDG method for computing complicated flows of practical importance.

5 Conclusions

A reconstructed discontinuous Galerkin method using a Taylor basis has been developed for the solution of the compressible Euler equations on arbitrary grids. Similar to the recovery-based rDG(P1P2) method, The developed reconstruction-based RDG(P1P2) method is able to achieve a third-order accuracy: one order accuracy higher than the underlying DG method. In comparison with the recovery-based DG methods, the reconstruction DG methods are relatively simple to implement, numerically robust and flexible, and computationally efficient. A number of numerical experiments have been conducted to demonstrate the superior performance of the RDG(P1P2) method over the

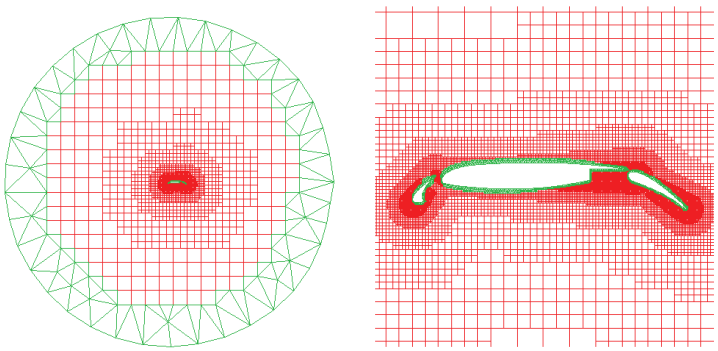

Figure 14: Hybrid unstructured Cartesian and triangle mesh (13,892 Cartesian cells, and 3,981 triangular cells) used for computing subsonic flow past a three-element airfoil at $M_\infty = 0.2$, $\alpha = 10^\circ$.

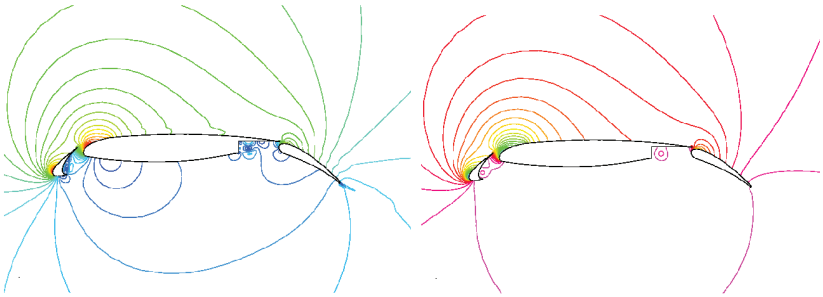

Figure 15: The computed Mach number (left) and pressure (right) contours obtained by the RDG(P1P2) method for subsonic flow past a three-element airfoil at $M_\infty = 0.2$, $\alpha = 10^\circ$.

underlying DG(P1) and DG(P2) methods. The developed RDG method has also successfully been extended to discretize the viscous and heat fluxes in the Navier-Stokes equations using the so-called "inter-cell reconstruction". The current development is focused on the extension of this reconstructed DG method for 3D problems, and the development of the RDG(P2P4) method is also under consideration.

Acknowledgments

This manuscript has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/CON-09-16528) with the U.S. Department of Energy. The United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The first author would like to acknowledge the partial support for this work provided by the INL staff-faculty exchange program, while he was in residence at Reactor Safety Simulation Group, Idaho National Laboratory, Idaho Falls, ID.

References

[1] W.H. Reed and T.R. Hill, Triangular Mesh Methods for the Neutron Transport Equation, Los Alamos Scientific Laboratory Report, LA-UR-73-479, 1973.

[2] B. Cockburn, S. Hou, and C.W. Shu, TVD Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for conservation laws IV: The Multidimensional Case, Mathematics of Computation, Vol. 55, pp. 545-581, 1990.

[3] B. Cockburn and C.W. Shu, The Runge-Kutta Discontinuous Galerkin Method for conservation laws V: Multidimensional System, Journal of Computational Physics, Vol. 141, pp. 199-224, 1998.

[4] B. Cockburn, G. Karniadakis, and C.W. Shu, The Development of Discontinuous Galerkin Method, in Discontinuous Galerkin Methods, Theory, Computation, and Applications, edited by B. Cockburn, G.E. Karniadakis, and C.W. Shu, Lecture Notes in Computational Science and Engineering, Springer-Verlag, New York, 2000, Vol. 11 pp. 5-50, 2000.

[5] F. Bassi and S. Rebay, High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations, Journal of Computational Physics, Vol. 138, pp. 251-285, 1997.

[6] H.L. Atkins and C.W. Shu, Quadrature Free Implementation of Discontinuous Galerkin Method for Hyperbolic Equations, AIAA Journal, Vol. 36, No. 5, 1998.

[7] F. Bassi and S. Rebay, GMRES discontinuous Galerkin solution of the Compressible Navier-Stokes Equations, Discontinuous Galerkin Methods, Theory, Computation, and Applications, edited by B. Cockburn, G.E. Karniadakis, and C.W. Shu, Lecture Notes in Computational Science and Engineering, Springer-Verlag, New York, 2000, Vol. 11, pp. 197-208, 2000.

[8] T.C. Warburton and G. E. Karniadakis, A Discontinuous Galerkin Method for the Viscous MHD Equations, Journal of Computational Physics, Vol. 152, pp. 608-641, 1999.

[9] J.S. Hesthaven and T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Texts in Applied Mathematics, Vol. 56, 2008.

[10] P. Rasetarinera and M.Y. Hussaini, An Efficient Implicit Discontinuous Spectral Galerkin Method, Journal of Computational Physics, Vol. 172, pp. 718-738, 2001.

[11] B.T. Helenbrook, D. Mavriplis, and H.L. Atkins, Analysis of p-Multigrid for Continuous and Discontinuous Finite Element Discretizations, AIAA Paper 2003-3989, 2003.

[12] K.J. Fidkowski, T.A. Oliver, J. Lu, and D.L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, Journal of Computational Physics, Vol. 207, No. 1, pp. 92-113, 2005.

[13] H. Luo, J.D. Baum, and R. Löhner, A Discontinuous Galerkin Method Using Taylor Basis for Compressible Flows on Arbitrary Grids, Journal of Computational Physics, Vol. 227, No 20, pp. 8875-8893, October 2008.

[14] H. Luo, J.D. Baum, and R. Löhner, On the Computation of Steady-State Compressible Flows Using a Discontinuous Galerkin Method, International Journal for Numerical Methods in Engineering, Vol. 73, No. 5, pp. 597-623, 2008.

[15] H. Luo, J. D. Baum, and R. Löhner, A Hermite WENO-based Limiter for Discontinuous Galerkin Method on Unstructured Grids, Journal of Computational Physics, Vol. 225, No. 1, pp. 686-713, 2007.

[16] H. Luo, J.D. Baum, and R. Löhner, A p-Multigrid Discontinuous Galerkin Method for the Euler Equations on Unstructured Grids, Journal of Computational Physics, Vol. 211, No. 2, pp. 767-783, 2006.

[17] H. Luo, J.D. Baum, and R. Löhner, Fast, p-Multigrid Discontinuous Galerkin Method for Compressible Flows at All Speeds, AIAA Journal, Vol. 46, No. 3, pp.635-652, 2008.

[18] M. Dumbser, D.S. Balsara, E.F. Toro, and C.D. Munz, A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes, Journal of Computational Physics, Vol 227, pp. 8209-8253, 2008.
[19] M. Dumbser and O. Zanotti, Very high order PNPM schemes on unstructured meshes for the resistive relativistic MHD equations, Journal of Computational Physics, Vol. 228, pp. 6991-7006, 2009.
[20] M. Dumbser, Arbitrary High Order PNPM Schemes on Unstructured Meshes for the Compressible Navier-Stokes Equations, Computers & Fluids, Vol. 39, pp. 60-76. 2010.
[21] 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, Vol. 131, pp. 267-279, 1997.
[22] F. Bassi and S. Rebay, Discontinuous Galerkin Solution of the Reynolds-Averaged Navier-Stokes and k-ω Turbulence Model Equations, Journal of Computational Physics, Vol. 34, pp. 507-540, 2005.
[23] B. Cockburn and C.W. Shu, The Local Discontinuous Galerkin Method for Time-dependent Convection-Diffusion System, SIAM Journal of Numerical Analysis, Vol. 35, pp. 2440-2463, 1998.
[24] C.E. Baumann and J.T. Oden, A Discontinuous hp Finite Element Method for the Euler and Navier-Stokes Equations, International Journal for Numerical Methods in Fluids, Vol. 31, pp. 79-95, 1999.
[25] J. Peraire and P.O. Persson, The Compact Discontinuous Galerkin Method for Elliptic Problems. SIAM Journal on Scientific Computing, Vol. 30, pp. 1806-1824, 2008.
[26] 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. Vol. 39, No. 5, pp. 1749-1779, 2002.
[27] 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 Computational Physics, Vol. 224, No. 2, pp. 1049-1063, 2007.
[28] H. Liu and K. Xu, A Runge-Kutta Discontinuous Galerkin Method for Viscous Flow Equations, Journal of Computational Physics, Vol. 224, No. 2, pp. 1223-1242, 2007.
[29] H. Luo, L. Luo, and K. Xu, A Discontinuous Galerkin Method Based on a BGK Scheme for the Navier-Stokes Equations on Arbitrary Grids, Advances in Applied Mathematics and Mechanics, Vol. 1, No. 3, pp. 301-318, 2009.
[30] B. van Leer and S. Nomura, Discontinuous Galerkin Method for Diffusion, AIAA Paper 2005-5108,2005.
[31] B. van Leer and M. Lo, A Discontinuous Galerkin Method for Diffusion Based on Recovery, AIAA Paper 2007-4083, 2007.
[32] M. Raalte and B. van Leer, Bilinear Forms for the Recovery-Based Discontinuous Galerkin Method for Diffusion, Communication in Computational Physics, Vol. 5, No. 2-4, pp. 683-693, 2009.
[33] R. Nourgaliev, H. Park, and V. Mousseau, Recovery Discontinuous Galerkin Jacobian-Free Newton-Krylov Method for Multiphysics Problems, Computational Fluid Dynamics Review 2010, 2010.
[34] P. Colella and P.R. Woodward, The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations, Journal of Computational Physics, Vol. 54, No. 1, pp. 115-173, 1984.
[35] H. Luo, L. Luo, R. Nourgaliev, V. Mousseau, and N. Dinh, A Reconstructed Discontinuous Galerkin Method for the Compressible Navier-Stokes Equations on Arbitrary Grids, Journal

of Computational Physics, Vol. 229, pp. 6961-6978, 2010.
[36] M. Aftosmis, D. Gaitonde, and T.S. Tavares, On the Accuracy, Stability, and Monotonicity of Various Reconstruction Algorithms for Unstructured Meshes, AIAA Paper 94-0415, 1994.
[37] A. Haselbacher, J.J. McGuirk, and G.J. Page, Finite-Volume Discretization Aspects for Viscous Flows for Mixed Unstructured Meshes, AIAA Journal, 37(2), pp. 177-184, 1999.
[38] H. Luo, L. Luo, A. Ali, R. Nourgaliev, and C. Cai, A Parallel, Reconstructed Discontinuous Galerkin Method for the Compressible Flows on Arbitrary Grids, Communication in Computational Physics, Vol. 9, pp. 363-389, 2010.
[39] H. Luo, D. Sharov, J.D. Baum, and R. Löhner, On the Computation of Compressible Turbulent Flows on Unstructured Grids, International Journal of Computational Fluid Dynamics, Vol. 14, pp. 253-270, 2001.
[40] H. Luo, J.D. Baum, and R. Löhner, A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids, Journal of Computational Physics, Vol. 146, No. 2, pp. 664-690, 1998.
[41] H. Luo, J.D. Baum, and R. Löhner, High-Reynolds Number Viscous Flow Computations Using an Unstructured-Grid Method, Journal of Aircraft, Vol. 42, No. 2, pp. 483-492, 2005.
[42] H. Luo, J.D. Baum, and R. Löhner, Extension of HLLC Scheme for Flows at All Speeds, AIAA Journal, Vol. 43, No. 6, pp. 1160-1166, 2005.
[43] L.P. Zhang, W. Liu, L.X. He, X.G. Deng, and H.X. Zhang, A Class of Hybrid DG/FV Methods for Conservation Laws II: Two-dimensional Cases, Journal of Computational Physics, Vol. 231, pp. 1104-1120, 2011.
[44] D. Sharov, H. Luo, J.D. Baum, and R. Löhner, Unstructured Navier-Stokes Grid Generation at Corners and Ridges, International Journal for Numerical Methods in Fluids, Vol. 43, pp. 717-728, 2003.
[45] H. Luo, S. Spiegel, and R. Löhner, A Hybrid Unstructured Cartesian and Triangular/Tetrahedral Grid Generation Method for Complex Geometries, AIAA journal. Vol. 48, No., 11 pp. 2639-2647, 2010.
[46] R. Abgrall, On Essentially Non-Oscillatory Schemes on Unstructured Meshes: Analysis and Implementation, Journal of Computational Physics, Vol. 114, pp. 45-58, 1994.
[47] O. Friedrich, Weighted Essentially Non-Oscillatory Schemes for the Interpolation of Mean Values on Unstructured Grid, Journal of Computational Physics, Vol. 144, pp. 194-212, 1998.
[48] M. Dumbser and M. Käser, Arbitrary High Order Non-Oscillatory Finite Volume Schemes on Unstructured Meshes for Linear Hyperbolic Systems, Journal of Computational Physics, Vol. 221, pp. 693-723, 2007.
[49] M. Dumbser, M. Käser, V.A. Titarev, and E.F. Toro, Quadrature-free Non-Oscillatory Finite Volume Schemes on Unstructured Meshes for Nonlinear Hyperbolic Systems, Journal of Computational Physics, Vol. 226, pp. 204-243, 2007.
[50] C. Hu and C.W. Shu, Weighted Essentially Non-Oscillatory Schemes on Triangular Meshes, Journal of Computational Physics, Vol. 150, pp. 97-127, 1999.
[51] Y.T. Zhang and C.W. Shu, Third Order WENO Scheme on Three Dimensional Tetrahedral Meshes, Communication in Computational Physics, Vol. 5, pp. 836-848, 2009.
[52] P. Tsoutsanis, V.A. Titarev, and D. Drikakis, WENO Schemes on Arbitrary Mixed-Element Unstructured Meshes in Three Space Dimensions, Journal of Computational Physics, Vol. 230, pp. 1585-1601, 2011.