Commun. Comput. Phys.
doi: 10.4208/cicp.070210.020610a

Vol. 9, No. 2, pp. 363-389
February 2011

A Parallel, Reconstructed Discontinuous Galerkin Method for the Compressible Flows on Arbitrary Grids

Hong Luo[1,*], Luqing Luo[1], Amjad Ali[1], Robert Nourgaliev[2] and Chunpei Cai[3]

[1] *Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC, 27695, USA.*
[2] *Reactor Safety Simulation Group, Idaho National Laboratory, Idaho Falls, ID, 83415, USA.*
[3] *Department of Mechanical and Aerospace Engineering, New Mexico State University, Las Cruces, NM, 88001, USA.*

Received 7 February 2010; Accepted (in revised version) 2 June 2010

Communicated by Kun Xu

Available online 27 August 2010

Abstract. A reconstruction-based discontinuous Galerkin method is presented for the solution of the compressible Navier-Stokes equations on arbitrary grids. In this method, an in-cell reconstruction is used to obtain a higher-order polynomial representation of the underlying discontinuous Galerkin polynomial solution and an inter-cell reconstruction is used to obtain a continuous polynomial solution on the union of two neighboring, interface-sharing cells. The in-cell reconstruction is designed to enhance the accuracy of the discontinuous Galerkin method by increasing the order of the underlying polynomial solution. The inter-cell reconstruction is devised to remove an interface discontinuity of the solution and its derivatives and thus to provide a simple, accurate, consistent, and robust approximation to the viscous and heat fluxes in the Navier-Stokes equations. A parallel strategy is also devised for the resulting reconstruction discontinuous Galerkin method, which is based on domain partitioning and Single Program Multiple Data (SPMD) parallel programming model. The RDG method is used to compute a variety of compressible flow problems on arbitrary meshes to demonstrate its accuracy, efficiency, robustness, and versatility. The numerical results demonstrate that this RDG method is third-order accurate at a cost slightly higher than its underlying second-order DG method, at the same time providing a better performance than the third order DG method, in terms of both computing costs and storage requirements.

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

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

*Corresponding author. *Email addresses:* hong_luo@ncsu.edu (H. Luo), lluo2@ncsu.edu (L. Luo), aali3@ncsu.edu (A. Ali), robert.nourgaliev@inl.gov (R. Nourgaliev), ccai@nmsu.edu (C. Cai)

1 Introduction

The discontinuous Galerkin methods [1–25] (DGM) have recently become popular for the solution of systems of conservation laws. Nowadays, they are widely used in computational fluid dynamics, computational acoustics, and computational electromagnetics. The discontinuous Galerkin methods combine two advantageous features commonly associated to 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 methods are therefore 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 method 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 to 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, because 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, that recovers a smooth continuous solution that in the weak sense is indistinguishable from the discontinuous discrete solution.

Dumbser et al. [18–20] have originally introduced a new family of reconstructed 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 ($m \geq n$) that is used to compute the fluxes. 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., 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 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 principal, 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 [37], linear rDG(P1P5) is 6th order accurate, quadratic rDG(P2P8) is 9th-order accurate, and cubic rDG(P3P11) 12th-order accurate, versus the 2nd, 3rd, and 4th-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 reduced polynomial solution 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, that 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.

The objective of the effort discussed in this paper is to present a reconstructed discontinuous Galerkin method, termed RDG(P1P2) in short, using a Taylor basis [13] for computing the compressible flow problems on arbitrary grids, where the upper case *R* denotes Reconstruction, being different from *r* for Recovery, an in-cell reconstruction scheme [35] is used obtain a quadratic polynomial representation of the underlying linear DG solution, and an inter-cell reconstruction [36] on top of the in-cell reconstruction is introduced to obtain a continuous quadratic polynomial solution on the union of two neighboring, interface-sharing cells. The in-cell reconstruction is designed to enhance the accuracy of the discontinuous Galerkin method by increasing the order of the underlying polynomial solution. The inter-cell reconstruction is devised to remove an interface discontinuity of the solution and its derivatives and thus to provide a simple, accurate, consistent, and robust approximation to the diffusive fluxes. A parallel strategy is devised for the resulting reconstruction discontinuous Galerkin method, which is based on domain partitioning and Single Program Multiple Data (SPMD) parallel programming model using Message-Passing-Interface (MPI) programming paradigm for distributed memory parallel computing architectures. The developed RDG(P1P2) 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(P1P2) method is able to obtain a third-order accurate solution at a slightly higher cost than its second-order DG method and significantly increases its performance over the third order DG method in terms of computing costs and storage requirements, and the inter-cell reconstruction DG method is a simple alternative for an accurate, stable, consistent, and efficient discretization of the viscous fluxes, and is capable of delivering the same accuracy as BR2 scheme at a half of its computing costs. The good parallelization characteristics of the RDG(P1P2) method are demonstrated, achieved by hiding communication latency behind computation. The remainder of this paper is structured 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 Navier-Stokes equations governing unsteady compressible viscous flows can be expressed as

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

where the summation convention has been used. The conservative variable vector **U**, advective (inviscid) flux vector **F**, and viscous flux vector **G** are defined by

$$\mathbf{U}=\begin{pmatrix}\rho\\ \rho u_i\\ \rho e\end{pmatrix},\qquad \mathbf{F}_j=\begin{pmatrix}\rho u_j\\ \rho u_i u_j+p\delta_{ij}\\ u_j(\rho e+p)\end{pmatrix},\qquad \mathbf{G}_j=\begin{pmatrix}0\\ \sigma_{ij}\\ u_l\sigma_{lj}+q_j\end{pmatrix}. \tag{2.2}$$

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. The components of the viscous stress tensor σ_{ij} and the heat flux vector are given by

$$\sigma_{ij}=\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij},\qquad q_j=\frac{1}{\gamma-1}\frac{\mu}{\mathrm{Pr}}\frac{\partial T}{\partial x_j}. \tag{2.4}$$

In the above equations, T is the temperature of the fluid, Pr the laminar Prandtl number, which is taken as 0.7 for air. μ represents the molecular viscosity, which can be determined through Sutherland's law

$$\frac{\mu}{\mu_0}=\left(\frac{T}{T_0}\right)^{\frac{3}{2}}\frac{T_0+S}{T+S}. \tag{2.5}$$

μ_0 denotes the viscosity at the reference temperature T_0, and S is a constant which for are assumes the value $S=110^\circ$K. The temperature of the fluid T is determined by

$$T=\gamma\frac{p}{\rho}. \tag{2.6}$$

Neglecting viscous effects, the left-hand side of Eq. (2.1) represents the Euler equations governing unsteady compressible inviscid flows.

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,

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

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_h^p

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

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

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

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 a piecewise polynomial function 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,

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

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 choice of these fluxes is crucial for the DG formulation. Like in the finite volume methods, the inviscid flux function $\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k$ appearing in the boundary integral can be 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. The computation of the viscous fluxes in the boundary integral has to properly resolve the discontinuities at the interfaces. 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(P_0) method, i.e., to the discontinuous Galerkin method using a piecewise constant polynomial. Consequently, the DG(P_k) 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

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

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

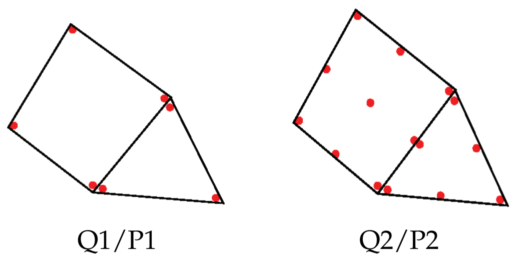

Figure 1: Representation of polynomial solutions using finite element shape functions.

On each cell, a system of $N\times N$ has to be solved, where 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 are the variables at the three vertices and a bi-linear polynomial is used for quadrilateral elements and the unknowns to be solved are the variables at the four vertices. However, numerical polynomial solutions **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 cell centroid, the quadratic polynomial solutions can be expressed as follows

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

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

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

where $\tilde{\mathbf{U}}$ is the mean value of **U** in this cell. The unknowns to be solved 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.

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

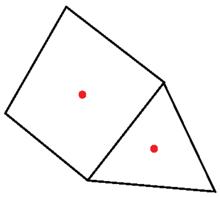

Figure 2: Representation of polynomial solutions using a Taylor series expansion for a cell-centered scheme.

are

$$B_1=1,\quad B_2=x-x_c,\quad B_3=y-y_c,\quad B_4=\frac{(x-x_c)^2}{2}-\frac{1}{\Omega_e}\int\limits_{\Omega_e}\frac{(x-x_c)^2}{2}d\Omega,$$

$$B_5=\frac{(y-y_c)^2}{2}-\frac{1}{\Omega_e}\int\limits_{\Omega_e}\frac{(y-y_c)^2}{2}d\Omega,\quad B_6=(x-x_c)(y-y_c)-\frac{1}{\Omega_e}\int\limits_{\Omega_e}(x-x_c)(y-y_c)d\Omega. \quad (3.9)$$

The discontinuous Galerkin formulation then leads to the following six equations

$$\frac{d}{dt}\int\limits_{\Omega_e}\tilde{\mathbf{U}}d\Omega+\int\limits_{\Gamma_e}\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k d\Gamma=\int\limits_{\Gamma_e}\mathbf{G}_k(\mathbf{U}_h)\mathbf{n}_k d\Gamma,\qquad i=1,$$

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

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

$$\int\limits_{\Omega_e}B_1B_i d\Omega=0,\qquad 2\leqslant i\leqslant 6. \quad (3.11)$$

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:

$$B_1=1,\quad B_2=\frac{x-x_c}{\Delta x},\quad B_3=\frac{y-y_c}{\Delta y},\quad B_4=\frac{(x-x_c)^2}{2\Delta x^2}-\frac{1}{\Omega_e}\int\limits_{\Omega_e}\frac{(x-x_c)^2}{2\Delta x^2}d\Omega,$$

$$B_5=\frac{(y-y_c)^2}{2\Delta y^2}-\frac{1}{\Omega_e}\int\limits_{\Omega_e}\frac{(y-y_c)^2}{2\Delta y^2}d\Omega,\quad B_6=\frac{(x-x_c)(y-y_c)}{\Delta x\Delta y}-\frac{1}{\Omega_e}\int\limits_{\Omega_e}\frac{(x-x_c)(y-y_c)}{\Delta x\Delta y}d\Omega,$$

(3.12)

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

$$\mathbf{U}_h=\tilde{\mathbf{U}}+\left.\frac{\partial\mathbf{U}}{\partial x}\right|_c\Delta xB_2+\left.\frac{\partial\mathbf{U}}{\partial y}\right|_c\Delta yB_3+\left.\frac{\partial^2\mathbf{U}}{\partial x^2}\right|_c\Delta x^2B_4+\left.\frac{\partial^2\mathbf{U}}{\partial y^2}\right|_c\Delta y^2B_5+\left.\frac{\partial^2\mathbf{U}}{\partial x\partial y}\right|_c\Delta x\Delta yB_6. \quad (3.13)$$

The above normalization is especially important to alleviate the stiffness of the system matrix for higher-order DG approximations.

This Taylor-basis DG method has a number of attractive features. Theoretically, this formulation allows us to clearly see the similarity and difference between DG and FV methods. In fact, the discretized governing equations for the cell-averaged variables and the assumption of polynomial solutions on each cell are exactly the same for both finite volume and DG methods. The only difference between them is the way how they obtain high-order (>1) polynomial solutions. In the finite volume methods, the polynomial solution of degrees p are reconstructed using the mean values of the neighboring cells, which can be obtained using either TVD/MUSCL or ENO/WENO reconstruction schemes. Unlike the FV methods, the DG methods compute the derivatives in a manner similar to the mean variables. This is compact, rigorous, and elegant mathematically in contrast with arbitrariness characterizing the reconstruction schemes with respect how to compute the derivatives and how to choose the stencils in the FV methods. Furthermore, 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. In addition, the Taylor-basis DG method makes the implementation of both in-cell and inter-cell reconstruction schemes straightforward and simple [35,36].

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. One 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. It is appealing to develop a numerical method that combines the efficiency of the reconstruction methods and the accuracy of the DG methods. The "reconstructed DG" methods, originally proposed by Dumbser et al. [18–20], and termed PnPm schemes represent a first step in this direction. The key issue is how to construct an accurate and efficient reconstruction operator, i.e., how to obtain a higher order polynomial solution from an underlying DG solution. Normally, this is achieved using a so-called in-cell recovery where recovered equations are obtained using a L_2 projection in the weak sense. However, recovery is not the only way to obtain a higher-order polynomial representation of an underlying DG solution. Rather, reconstruction widely used in the finite volume methods provides an alternative, probably a better choice to obtain a higher-order polynomial representation. Although our discussion in this work is mainly focused on the RDG(P1P2) method in 2D, its extension to higher-order and 3D DG methods is straightforward. In the case of RDG(P1P2) method, a linear polynomial solution $\mathbf{U}_i$ in any cell i is

$$\mathbf{U}_i = \tilde{\mathbf{U}}_i + \mathbf{U}_{\mathbf{x}i} B_2 + \mathbf{U}_{\mathbf{yi}} B_3. \tag{3.14}$$

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

$$\mathbf{U}_i^R = \tilde{\mathbf{U}}_i^R + \mathbf{U}_{xi}^R B_2 + \mathbf{U}_{yi}^R B_3 + \mathbf{U}_{xxi}^R B_4 + \mathbf{U}_{yyi}^R B_5 + \mathbf{U}_{xyi}^R B_6. \tag{3.15}$$

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.,

$$\tilde{\mathbf{U}}_i^R = \tilde{\mathbf{U}}_i, \qquad \mathbf{U}_{xi}^R = \mathbf{U}_{xi}, \qquad \mathbf{U}_{yi}^R = \mathbf{U}_{yi}. \tag{3.16}$$

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

$$\begin{aligned}
&\mathbf{U}_j = \tilde{\mathbf{U}}_i + \mathbf{U}_{xi} B_2 + \mathbf{U}_{yi} B_3 + \mathbf{U}_{xxi}^R B_4 + \mathbf{U}_{yyi}^R B_5 + \mathbf{U}_{xyi}^R B_6,\\
&\left.\frac{\partial \mathbf{U}}{\partial x}\right|_j = \mathbf{U}_{xi}\frac{1}{\Delta x_i} + \mathbf{U}_{xxi}^R \frac{B_2}{\Delta x_i} + \mathbf{U}_{xyi}^R \frac{B_3}{\Delta x_i},\\
&\left.\frac{\partial \mathbf{U}}{\partial y}\right|_j = \mathbf{U}_{yi}\frac{1}{\Delta y_i} + \mathbf{U}_{yyi}^R \frac{B_3}{\Delta y_i} + \mathbf{U}_{xyi}^R \frac{B_2}{\Delta y_i},
\end{aligned} \tag{3.17}$$

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:

$$\begin{pmatrix} B_4^j & B_5^j & B_6^j \\ B_2^j & 0 & B_3^j \\ 0 & B_3^j & B_2^j \end{pmatrix} \begin{pmatrix} \mathbf{U}_{xxi}^R \\ \mathbf{U}_{yyi}^R \\ \mathbf{U}_{xyi}^R \end{pmatrix} = \begin{pmatrix} \mathbf{U}_j - (\mathbf{U}_i B_1^j + \mathbf{U}_{xi} B_2^j + \mathbf{U}_{yi} B_3^j) \\ \frac{\Delta x_i}{\Delta x_j}\mathbf{U}_{xj} - \mathbf{U}_{xi} \\ \frac{\Delta y_i}{\Delta y_j}\mathbf{U}_{yj} - \mathbf{U}_{yi} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1^j \\ \mathbf{R}_2^j \\ \mathbf{R}_3^j \end{pmatrix}, \tag{3.18}$$

where $\mathbf{R}$ 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

$$\begin{pmatrix} \sum\limits_j (B_4^j B_4^j + B_4^j B_4^j) & \sum\limits_j B_4^j B_5^j & \sum\limits_j (B_4^j B_6^j + B_2^j B_3^j) \\ \sum\limits_j B_4^j B_5^j & \sum\limits_j (B_5^j B_5^j + B_3^j B_3^j) & \sum\limits_j (B_5^j B_6^j + B_2^j B_3^j) \\ \sum\limits_j (B_4^j B_6^j + B_2^j B_3^j) & \sum\limits_j (B_5^j B_6^j + B_2^j B_3^j) & \sum\limits_j (B_6^j B_6^j + B_2^j B_2^j + B_3^j B_3^j) \end{pmatrix} \begin{pmatrix} \mathbf{U}_{xxi}^R \\ \mathbf{U}_{yyi}^R \\ \mathbf{U}_{xyi}^R \end{pmatrix}$$
$$= \begin{pmatrix} \sum\limits_j (B_4^j \mathbf{R}_1^j + B_2^j \mathbf{R}_2^j) \\ \sum\limits_j (B_5^j \mathbf{R}_1^j + B_3^j \mathbf{R}_3^j) \\ \sum\limits_j (B_6^j \mathbf{R}_1^j + B_3^j \mathbf{R}_2^j + B_2^j \mathbf{R}_3^j) \end{pmatrix}. \tag{3.19}$$

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 RDG(P1P2) 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.

The discretization of the Navier-Stokes equations requires the evaluation of the viscous fluxes at a cell interface, which has to properly resolve the discontinuities at the interfaces. Taking a simple arithmetic mean of the viscous fluxes from the left and right cells is inconsistent, because the arithmetic mean of the solution derivatives does not take into account a possible jump of the solutions. In the reconstructed RDG(P1P2) method, a continuous quadratic polynomial solution $\mathbf{U}^R$ is reconstructed on the union of two cells $\Omega_{ij}(=\Omega_i \cup \Omega_j)$ adjacent to the interface based on the reconstructed in-cell discontinuous Galerkin solution in the two abutting elements. This reconstructed smooth solution is then used to compute the viscous fluxes at the interface. Without lose of generality, let us

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

consider the case of RDG(P1P2) method, where the reconstructed solution $\mathbf{U}^R$, similar to the underlying DG solution on Ω_i, can be expressed in Ω_{ij} using a Taylor basis as follows:

$$\mathbf{U}^R = \tilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_2(\mathbf{x}) + \mathbf{U}_y^{ij} B_3(\mathbf{x}) + \mathbf{U}_{xx}^{ij} B_4(\mathbf{x}) + \mathbf{U}_{yy}^{ij} B_5(\mathbf{x}) + \mathbf{U}_{xy}^{ij} B_6(\mathbf{x}), \tag{3.20}$$

where $\tilde{\mathbf{U}}_{ij}$ is the mean value of $\mathbf{U}^R$ on Ω_{ij}, and the derivatives are the point-wise value at the center of Ω_{ij}. There are six degrees of freedom, and therefore six unknowns to be determined. However, the cell-average value $\tilde{\mathbf{U}}_{ij}$ can be trivially obtained, by requiring the reconstruction scheme to be conservative, a fundamental requirement. Due to the judicious choice of Taylor basis in our DG formulation, this leads to

$$\tilde{\mathbf{U}}_{ij} = \frac{\tilde{\mathbf{U}}_i \Omega_i + \tilde{\mathbf{U}}_j \Omega_j}{\Omega_i + \Omega_j}. \tag{3.21}$$

The remaining five degrees of freedom can be determined by requiring that the reconstructed solution and its derivatives are equal to the underlying and reconstructed DG solution and its derivatives at cells i and j. Consider cell i, one obtains

$$\begin{aligned}
&\mathbf{U}_i = \tilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_{2i} + \mathbf{U}_y^{ij} B_{3i} + \mathbf{U}_{xx}^{ij} B_{4i} + \mathbf{U}_{yy}^{ij} B_{5i} + \mathbf{U}_{xy}^{ij} B_{6i},\\
&\left.\frac{\partial \mathbf{U}}{\partial x}\right|_i \Delta x_i = \left(\mathbf{U}_x^{ij} \frac{\partial B_{2i}}{\partial x} + \mathbf{U}_{xx}^{ij} \frac{\partial B_{4i}}{\partial x} + \mathbf{U}_{xy}^{ij} \frac{\partial B_{6i}}{\partial x} \right) \Delta x_i,\\
&\left.\frac{\partial \mathbf{U}}{\partial y}\right|_i \Delta y_i = \left(\mathbf{U}_y^{ij} \frac{\partial B_{3i}}{\partial y} + \mathbf{U}_{yy}^{ij} \frac{\partial B_{5i}}{\partial y} + \mathbf{U}_{xy}^{ij} \frac{\partial B_{6i}}{\partial y} \right) \Delta y_i,\\
&\mathbf{U}_{xxi}^R = \mathbf{U}_{xx}^{ij} \frac{\partial^2 B_{4i}}{\partial x^2} \Delta x_i^2 \quad \mathbf{U}_{yyi}^R = \mathbf{U}_{yy}^{ij} \frac{\partial^2 B_{5i}}{\partial y^2} \Delta y_i^2,\\
&\mathbf{U}_{xyi}^R = \mathbf{U}_{xy}^{ij} \frac{\partial^2 B_{6i}}{\partial x \partial y} \Delta x_i \Delta y_i,
\end{aligned} \tag{3.22}$$

where the basis function B is evaluated at the center of cell i, i.e., $B_i = B(x_j, y_j)$. This can be written in a matrix form as follows:

$$\begin{pmatrix} B_{2i} & B_{3i} & B_{4i} & B_{5i} & B_{6i} \\ \frac{\partial B_{2i}}{\partial x}\Delta x_i & 0 & \frac{\partial B_{4i}}{\partial x}\Delta x_i & 0 & \frac{\partial B_{6i}}{\partial x}\Delta x_i \\ 0 & \frac{\partial B_{3i}}{\partial y}\Delta y_i & 0 & \frac{\partial B_{5i}}{\partial y}\Delta y_i & \frac{\partial B_{6i}}{\partial y}\Delta y_i \\ 0 & 0 & \frac{\partial^2 B_{4i}}{\partial x^2}\Delta x_i^2 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial^2 B_{5i}}{\partial y^2}\Delta y_i^2 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial B_{6i}}{\partial x \partial y}\Delta x_i \Delta y_i \end{pmatrix} \begin{pmatrix} \mathbf{U}_x^{ij} \\ \mathbf{U}_y^{ij} \\ \mathbf{U}_{xx}^{ij} \\ \mathbf{U}_{yy}^{ij} \\ \mathbf{U}_{xy}^{ij} \end{pmatrix} = \begin{pmatrix} \mathbf{U}_i = \tilde{\mathbf{U}}_{ij} \\ \frac{\partial \mathbf{U}}{\partial x}\Big|_i \Delta x_i \\ \frac{\partial \mathbf{U}}{\partial y}\Big|_i \Delta y_i \\ \mathbf{U}_{xxi}^R \\ \mathbf{U}_{yyi}^R \\ \mathbf{U}_{xyi}^R \end{pmatrix}. \tag{3.23}$$

Similar equations could be derived for the cell j, which leads to a non-square matrix. The size of the resulting non-square matrix is 12×5. This over-determined linear system of 12 equations for 5 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 matrix transpose, yields a symmetric linear system of equations 5×5. This linear system of equations can be then trivially solved to obtain the five derivatives of the reconstructed continuous quadratic polynomial solution. This reconstructed smooth quadratic polynomial solution is then used to compute the viscous and heat fluxes in the Navier-Stokes equations at the interfaces. Similar to the recovered DG methods, the inter-cell reconstruction is compact, as it only involves two cells adjacent to the interfaces. Unlike the recovery-based DG methods, the reconstructed DG method only reconstructs a smooth polynomial solution of the same order as the underlying DG solution, thus there is no need to judiciously choose a proper form of a recovered polynomial and make sure that the recovered system is well conditioned and can be inverted. As the computation of the viscous and heat fluxes requires the differentiation of the solution in the direction normal to the interfaces and the reconstruction is anisotropic due to the embedded 1D interpolation problem in the direction connecting the centers of two cells i and j, it is natural to increase the accuracy of the reconstructed polynomial solution in that direction. This can be done by adding a cubic term in that direction to the reconstructed polynomial solution (3.20), which reads

$$\mathbf{U}^R = \tilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_2(\mathbf{x}) + \mathbf{U}_y^{ij} B_3(\mathbf{x}) + \mathbf{U}_{xx}^{ij} B_4(\mathbf{x}) + \mathbf{U}_{yy}^{ij} B_5(\mathbf{x}) + \mathbf{U}_{xy}^{ij} B_6(\mathbf{x}) + \mathbf{U}_{\xi\xi\xi}^{ij} B_7(\xi), \tag{3.24}$$

where

$$\begin{aligned} B_7(\xi) &= \frac{\xi^3}{6\Delta\xi^3} - \frac{1}{\Omega_{ij}} \int_{\Omega_{ij}} \frac{\xi^3}{6\Delta\xi^3} d\Omega, \\ \xi &= (\mathbf{x} - \mathbf{x}_{ij}) \bullet \mathbf{n} = (x - x_{ij}) n_x + (y - y_{ij}) n_y. \end{aligned} \tag{3.25}$$

This inter-cell reconstruction leads to an over-determined system of 12 equations with 6 unknowns. The numerical experiments indicate that the use of this inter-cell reconstruction significantly increases the accuracy of RDG method for the discretization of the diffusive fluxes, at a moderate increase of the computing costs and storage requirements. This scheme for the discretization of the viscous and heat fluxes will be referred to as RDG(P1P2+) method from now on, where + indicates that the reconstructed polynomial solutions contain a higher order term in the normal direction to the interface.

It is worth to note that the application of this inter-cell reconstruction to DG(P0) method where the first derivative is approximated using a second-order central differencing method demonstrates that this reconstruction DG method automatically provides the coupling terms required for the stability and leads to a 5-point second-order scheme for the diffusive operator (second derivative) in 1D on a uniform grid, contrary to most of discretization methods that lead to a 3-point stencil second-order method. This analysis indicates the potential of this reconstruction method for the accurate and robust discretization of the viscous fluxes on highly non-uniform, highly stretched, and highly distorted grids, as it is practically impossible to obtain a second-order accurate and compact cell-centered finite volume method for multi-dimensional problems on such grids.

This reconstructed DG method has been implemented in a well-tested 2D DG code [13–17, 29]. In this code, a fast, low-storage p-multigrid method [16, 17] is 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. The RDG methods, like DG methods, are highly parallelizable, as they are data-wise compact and each element is independent. Since the elements are discontinuous, and the inter-element communications are minimal, domain decomposition approach can be efficiently employed for parallelization. Therefore in the parallel implementation of the discontinuous Galerkin method on distributed memory parallel computing architectures, the computational domain is partitioned among the available processors and the information at the partition boundary faces, i.e., faces having its *left* and *right* cells on different processors, only needs to be exchanged between the corresponding neighboring processors. Moreover, it also naturally fits to Single Program Multiple Data (SPMD) parallel programming model. For this, Message-Passing-Interface (MPI) programming paradigm for distributed memory systems has been employed. Particular attention is paid on hiding communication latency behind computation in the flux computation part.

4 Numerical examples

Unless stated otherwise, all computations are conducted 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.

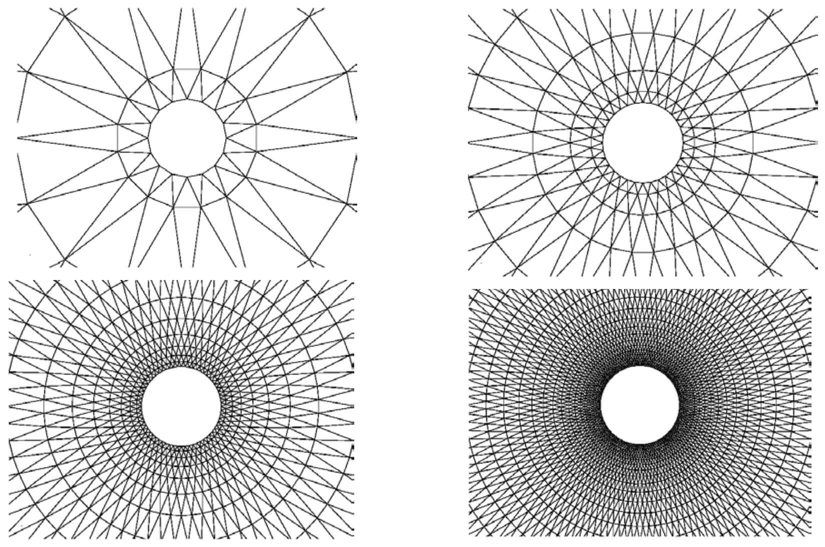

Figure 3: Sequences of four successively globally refined triangular meshes 16×5, 32×9, 64×17, 128×33 for computing subsonic flow past a circular cylinder.

4.1 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(P1P2) method can be achieved for the compressible Euler equations on unstructured grids. Fig. 3 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 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

$$r_i = r_1\left(1+\frac{2\pi}{123}\sum_{j=0}^{i-1}\alpha^j\right), \quad i=2,\cdots,33,$$

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

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

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

Mesh	No. DOFs	L_2-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	L_2-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	L_2-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	4,9152	4.16294E-06	3.434

Table 5: Subsonic circular cylinder test case: RDG(P1P2) is order of $\mathcal{O}(h^3)$.

Mesh	No. DOFs	L_2-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

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. 4 shows the computed Mach number contours in the flow field obtained by FV(P1) on the128$\times$33 mesh, DG(P1) on the 64$\times$17 mesh, and DG(P2) and RDG(P1P2) on the 32$\times$8 mesh, respectively. One can see that the results obtained by DG(P2) on the 32$\times$8 mesh are more accurate than the ones obtained by DG(P1) on the 64$\times$17 mesh, which in turn are more accurate than the ones obtained by FV(P1) on the 128$\times$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. 5, 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 reconstructed DG(P1) 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. 6 illustrates that higher order DG

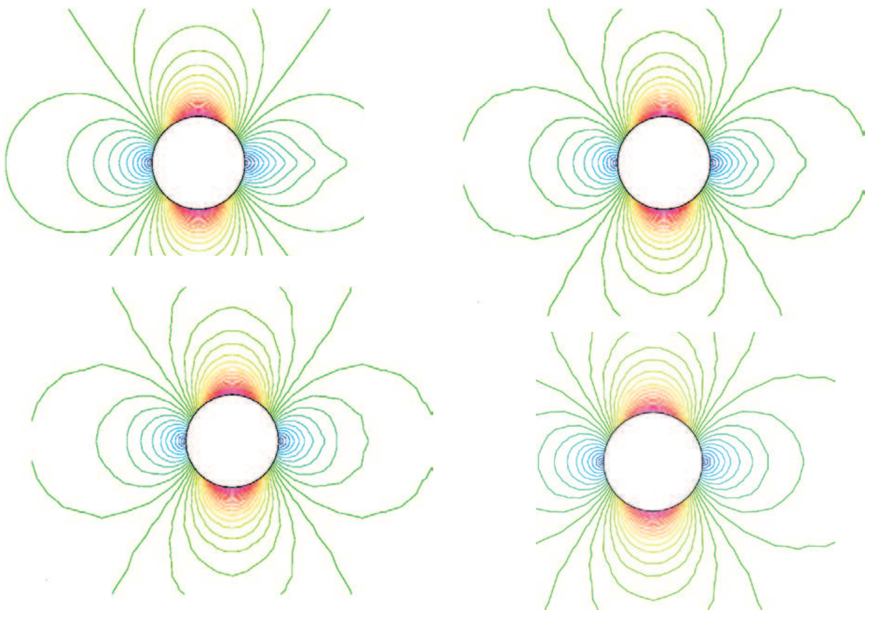

Figure 4: Computed Mach number contours in the flow field obtained by the FV(P1) method on 128×33 mesh (top left), DG(P1) method on 64×17 mesh (top right), DG(P2) method on 32×9 mesh (bottom left), and RDG(P1P2) on 32×9 mesh (bottom right) for subsonic flow past a circular cylinder at $M_\infty=0.38$.

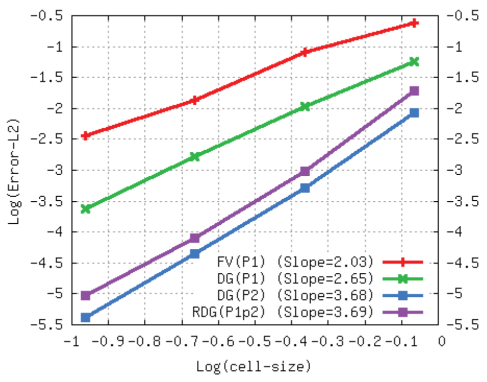

Figure 5: Convergence history for subsonic flow past a circular cylinder for FV(P1), DG(P1), DG(P2), and RDG(P1P2) methods.

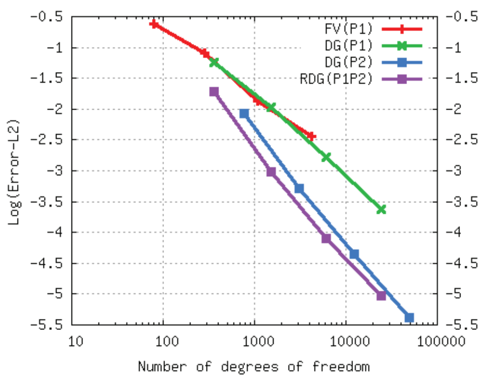

Figure 6: L_2-errors of numerical solutions vs. the number of degrees of freedom for subsonic flow past a circular cylinder by FV(P1), DG(P1), DG(P2), and RDG(P1P2) methods.

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.

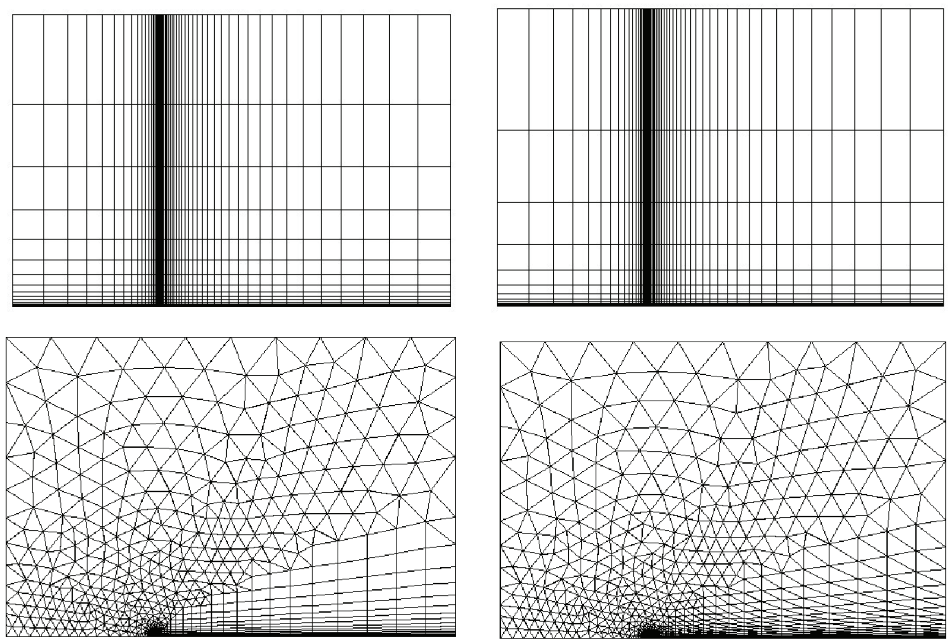
Figure 7: The quadrilateral grid with a stretching ratio of 1.2 (top left), the quadrilateral grid with a stretching ratio of 1.3 (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

4.2 Blasius boundary Layer

The laminar boundary layer over an adiabatic flat plate at a free-stream Mach number of 0.2 and a Reynolds number of 100,000 based on the freestream velocity and the length of the flat plate is considered in this test case, where the computational domain is bounded from -0.5 to 1 in the x-direction and 0 to 1 in the y-direction, and the flat plate starts at point (0,0) and extends to (1,0). This problem is chosen to illustrate that RDG(P1P2) method is able to maintain the same level of the accuracy as RDG(P2) method for the numerical solution of the Navier-Stokes equations, as the Blasius solution can be used to measure accuracy of the numerical solutions. Computations are performed on four grids: two quadrilateral grids, one hybrid grid, and one triangular grid, shown in Fig. 7, to assess the accuracy and consistence of the RDG(P1P2) method on different types of grids. The first two grids used in this test case have the same number of grid points (61×17), with 20 cells ahead of the flat plate and 40 cells for the flat plate, the same distribution of the grid points in the x-direction, but a different distribution of grid points in the y-direction. In order to cluster points near the wall, the point distribution in the y-direction follows a geometric stretching. The stretching ratio is the ratio of the heights of the two successive elements. A stretching ratio of 1.2 and 1.3 is used for the two meshes in the computation, respectively. For the grid with a stretching ratio of 1.2, the height of the first element is 0.1291E-02, and the cell sizes in the x-direction for the first element at the leading and trailing edges of the flat plate are 0.12086e-02 and 0.110386, respectively.

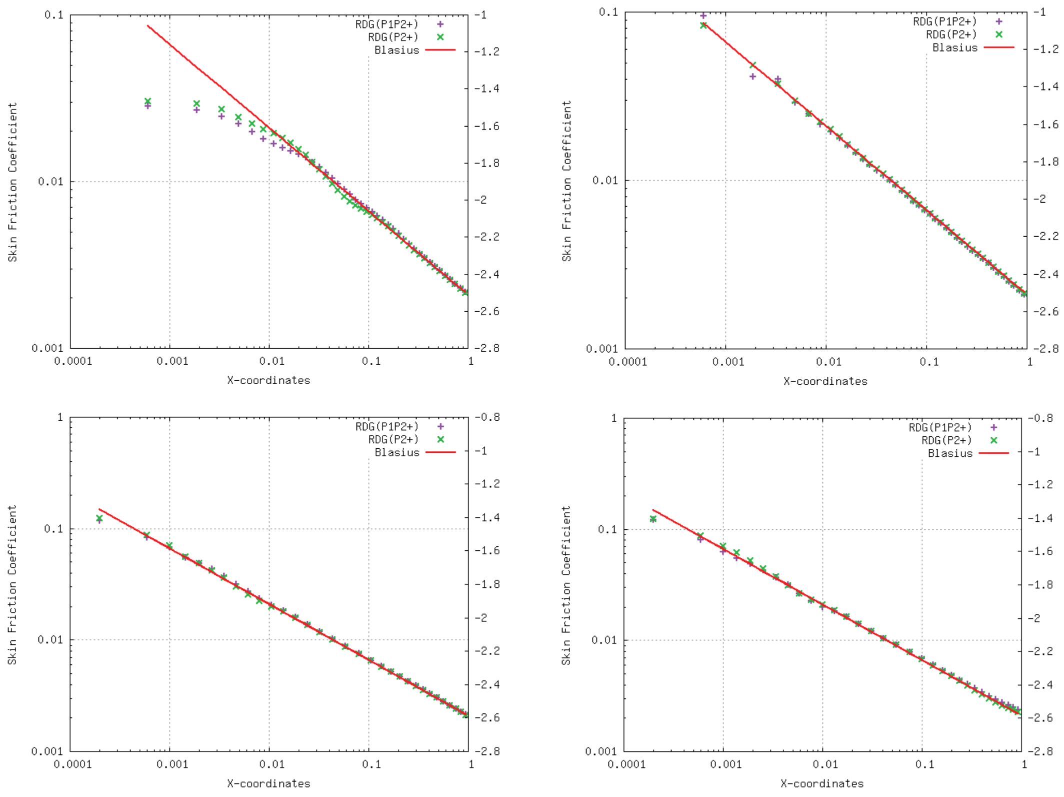

Figure 8: Logarithmic plot of the computed skin friction coefficient distribution along the flat plate obtained by the RDG(P2), and RDG(P1P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

When a stretching ratio is set to 1.3, the first grid-spacing off the wall is 0.155869E-03. The last two grids consist of 900 grid points, and 105 boundary points, with 31 grid points on the flat plate. The height of the first element is 0.3464E-03 and 0.82649E-03 at the leading and trailing edge of the flat plate respectively. As a result, the quadrilateral grid with a stretching ratio of 1.3 provides the best grid resolution for the boundary layers, and the quadrilateral grid with a stretching ratio of 1.2 has the least grid points in the boundary layers. The numerical results obtained by RDG(P2), and RDG(P1P2) on these four grids are presented, and compared with the theoretical one given by the well-known Blasius solution. Fig. 8 shows the logarithmic plot of the computed skin friction coefficient obtained by RDG(P2) and RDG(P1P2) solutions, respectively. Figs. 9 and 10 compare the profiles of velocity component in the x-, and y-direction at $x=0.2$ obtained by RDG(P2) and RDG(P1P2) solutions with Blasius solution, respectively. One can observe that both RDG(P2) and RDG(P1P2) solutions agree very well with the theoretical Blasius solution, even with as few as four grid cells in the boundary layer. Comparing the numerical

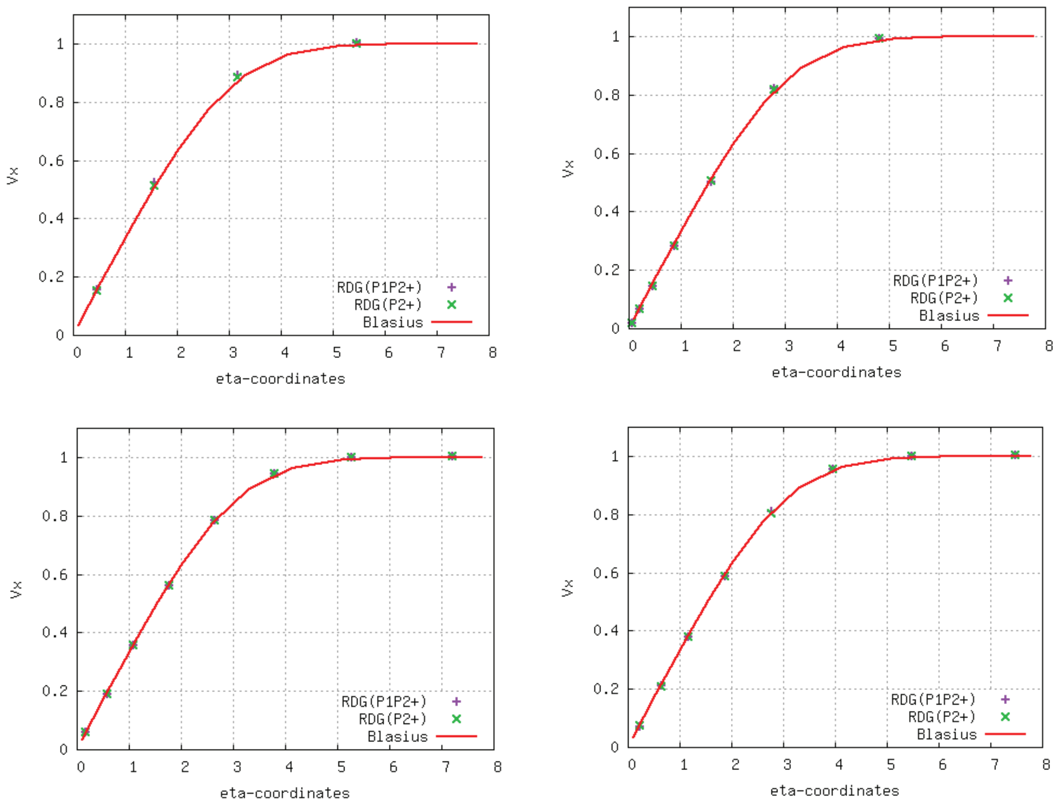

Figure 9: Comparison of the velocity profiles in the x-direction at $x=0.2$ obtained by the RDG(P2), and RDG(P1P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

solutions on the two quadrilateral grids, one can observe a consistent convergence of both reconstructed RDG(P2) and RDG(P1P2) methods. The more grid points are in the boundary layer, the more accurate the numerical solutions are, regardless of the highly non-uniformity of the grids. Note that most of the cell-centered finite volume methods are unable to obtain a consistent convergence on highly non-uniform grids and will produce a more accurate solution on the less stretching ratio grid. The numerical solutions obtained by the RDG(P1P2) method are very close to the ones produced by the RDG(P2) scheme, demonstrating that the in-cell reconstruction RDG(P1P2) method is able to deliver the same accuracy, convergence, and stability as the original DG(P2) scheme. Finally, by comparing the computed results between the hybrid and triangular grids, one can clearly understand the justification of using the hybrid grids for the computation of the viscous flows. A triangular grid has twice many grid cells than a quadrilateral grid, and yet yields much less accurate solutions than its hybrid counterpart.

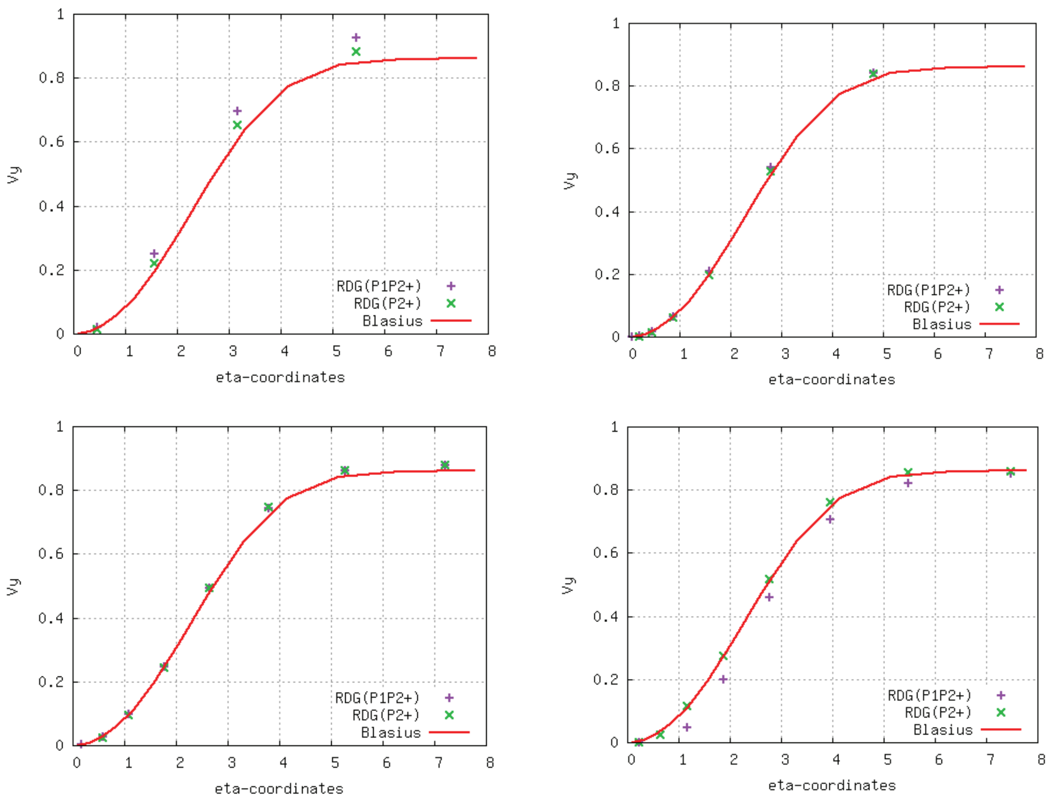

Figure 10: Comparison of the velocity profiles in the y-direction at $x=0.2$ obtained by the RDG(P2), and RDG(P1P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

4.3 Subsonic flows past a NACA0012 airfoil

The second test case involves a subsonic flow past a NACA0012 airfoil at a Mach number of 0.5, and an angle of attack 0°, and a Reynolds number of 5000 based on the freestream velocity and the chord length of the airfoil. An adiabatic wall is assumed in this test case. The Reynolds number is close to the upper limit of a steady flow. This computation is performed on a hybrid grid using RDG(P2+), and RDG(P1P2+) methods. Fig. 11 shows the computational grid used in this test case, consisting of 2,495 triangular elements, 549 quadrilateral elements, 1,856 grid points, and 121 boundary faces, and the computed Mach number contours in the flow field obtained by RDG(P1P2+) method. A distinguishing feature of this test case is the separation of the flow occurring near the trailing edge, which causes the formation of two small recirculation bubbles in the wake region. This can be clearly seen from the velocity vector plot in the vicinity of the trailing edge as shown in Fig. 12. The computed skin friction coefficients and pressure coeffi-

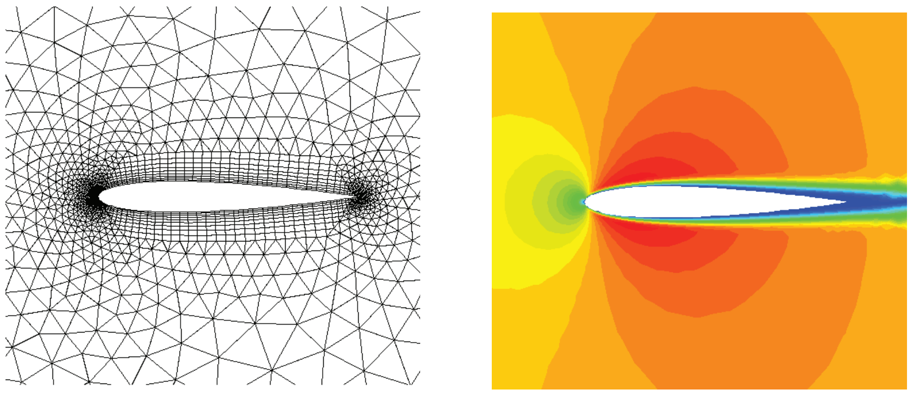

Figure 11: Unstructured mesh (left) (ntria=3,469, npoin=3,346, nbfac=157) and computed Mach number contours by the RDG(P2+) (right) for subsonic flow past a NACA0012 airfoil at $M_\infty=0.5$, $\mathrm{Re}=5{,}000$, $\alpha=0°$.

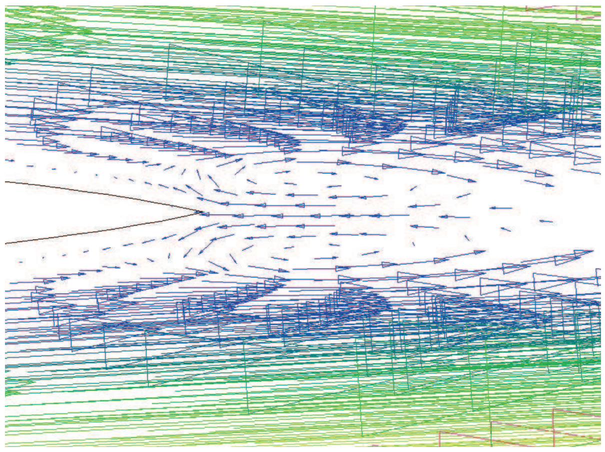

Figure 12: Velocity vector plot in the vicinity of the trailing edge for subsonic flow past a NACA0012 airfoil at $M_\infty=0.5$, $\mathrm{Re}=5{,}000$, $\alpha=0°$.

cients obtained by RDG(P2+), and RDG(P1P2+) are compared in Fig. 13, where the two solutions are virtually identical in this test case, again demonstrating that the developed reconstructed RDG(P1P2+) method is able to deliver the same accuracy as the RDG(P2+) method.

4.4 Parallel performance

The inviscid subsonic flow at a Mach number of 0.38, and an angle of attack of 0° past a circular cylinder is considered again in this test case to assess the performance of the developed parallelization. The computation is performed on a triangular element mesh consisting of 45362 elements, 22,888 grid points, and 414 boundary points. Fig. 14 shows the mesh split into 8 blocks using METIS [38] and for this test case the parallel perfor-

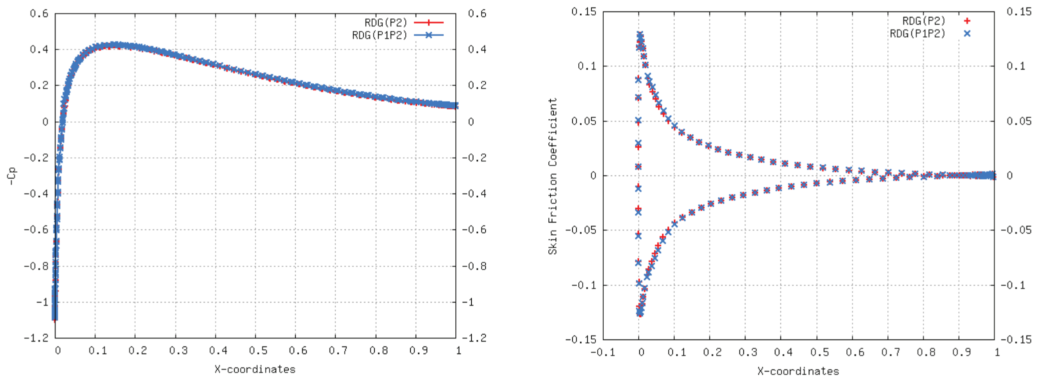

Figure 13: Computed skin friction coefficient distributions (left) and pressure coefficients (right) on the airfoil obtained by the RDG(P2) and RDG(P1P2).

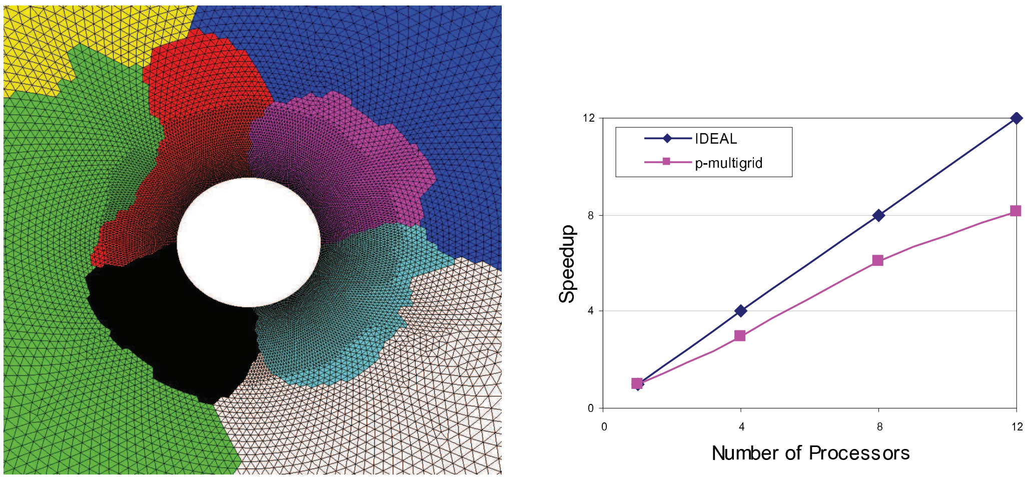

Figure 14: Partitioned mesh (left) and parallel speed-ups for the subsonic flow around a circular cylinder.

mance up to 12 processors on an Ethernet based cluster having dual multicore processors on the nodes, where one can observe the favorable parallelization of the RDG method. For the test case, a maximum of 4 nodes of the cluster has been used, with 3 processes mapped to each node for the 12 processors case.

5 Conclusion

A reconstructed discontinuous Galerkin RDG(P1P2) method has been presented for the solution of the compressible Navier-Stokes equations on arbitrary grids. An in-cell reconstruction is developed to obtain a piecewise quadratic polynomial solution from the underlying piecewise linear DG solution using a conservative least-squares method. The reconstructed quadratic polynomial solution is used for the computation of the inviscid fluxes and for the reconstruction of a continuous quadratic polynomial solution using a so-called inter-cell reconstruction. The reconstructed continuous quadratic polynomial solution on the union of two neighboring is then used for the discretization of the viscous and heat fluxes at the cell interfaces. The developed RDG(P1P2) method has been parallelized using MPI and used to compute a variety of flow problems on arbitrary grids to demonstrate its accuracy, efficiency, robustness, and versatility. The numerical results indicate that the developed RDG(P1P2) method is third-order accurate at a cost slightly higher than its underlying second-order DG method, at the same time providing a better performance than the third order DG method, in terms of both computing costs and storage requirements.

Acknowledgments

This manuscript has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/CON-10-17571) with the U.S. Department of Energy. The United States Government retains and the published, by accepting the article for publication, acknowledges that 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 Idaho National Laboratory, Idaho Falls, ID. The first and last authors would also like to acknowledge the partial support for this work provided by the NSF under project No. NSF-DMS0914706.

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, B. Cockburn, G. E. Karniadakis and C. W. Shu (Eds.), 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, B. Cockburn, G. E. Karniadakis and C. W. Shu (Eds.), 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, 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, A 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 of 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 2009, 2009, to appear.

[34] H.T Huynh, A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion, AIAA Paper, 2009-0403, 2009.

[35] H. Luo, L. Luo, R. Nourgaliev and V. Mousseau, A reconstructed discontinuous Galerkin method for the compressible Euler equations on arbitrary grids, AIAA Paper, 2009-3788, 2009.

[36] H. Luo, L. Luo, R. Norgaliev, V. A. Mousseau and N. Dinh, A reconstructed discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids, Journal of Computational Physics, Vol. 229, No. 19, pp. 6961-6978, 2010.

[37] 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.

[38] G. Karypis and V. Kumar, A fast and high quality scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing, Vol. 20, pp. 359-392, 1999.