Journal of Computational Physics 229 (2010) 6961–6978

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A reconstructed discontinuous Galerkin method for the compressible Navier–Stokes equations on arbitrary grids

Hong Luo [a,*], Luqing Luo [a], Robert Nourgaliev [b], Vincent A. Mousseau [b], Nam Dinh [b]

[a] Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, United States
[b] Thermal Science and Safety Analysis Department, Idaho National Laboratory, Idaho Falls, ID 83415, United States

ARTICLE INFO

Article history:
Received 6 February 2010
Received in revised form 13 May 2010
Accepted 24 May 2010
Available online 31 May 2010

Keywords:
Reconstruction schemes
Discontinuous Galerkin methods
Compressible Navier–Stokes equations

ABSTRACT

A reconstruction-based discontinuous Galerkin (RDG) method is presented for the solution of the compressible Navier–Stokes equations on arbitrary grids. The RDG method, originally developed for the compressible Euler equations, is extended to discretize viscous and heat fluxes in the Navier–Stokes equations using a so-called inter-cell reconstruction, where a smooth solution is locally reconstructed using a least-squares method from the underlying discontinuous DG solution. Similar to the recovery-based DG (rDG) methods, this reconstructed DG method eliminates the introduction of ad hoc penalty or coupling terms commonly found in traditional DG methods. Unlike rDG methods, this RDG method does not need to judiciously choose a proper form of a recovered polynomial, thus is simple, flexible, and robust, and can be used on arbitrary grids. 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 method is able to deliver the same accuracy as the well-known Bassi–Rebay II scheme, at a half of its computing costs for the discretization of the viscous fluxes in the Navier–Stokes equations, clearly demonstrating its superior performance over the existing DG methods for solving the compressible Navier–Stokes equations.

1. Introduction

The discontinuous Galerkin methods [1–31] (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 magneto-hydrodynamics. 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 (>2nd) 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

* Corresponding author. Tel.: +1 919 513 3898; fax: +1 919 515 7968.
E-mail address: hong_luo@ncsu.edu (H. Luo).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.05.033

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. However, DGM have a number of weaknesses that have yet to be addressed, before they can become a viable, attractive, competitive, and ultimately superior numerical method over more mature and well-established second-order finite volume methods 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, because the arithmetic mean of the solution derivatives does not take into 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 and Xu [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. The rDG method is further developed by Nourgaliev et al. [33] to solve the compressible Navier–Stokes equations on structured grids. The most attractive feature of rDG is the simplicity and generality of the recovery principle, which immediately allows one to create and further develop DG diffusion schemes. These schemes are of higher accuracy than any other DG diffusion schemes currently in use, while boasting the smallest eigenvalues, hence the largest stability range for explicit time-marching schemes, as shown by Huynh [34]. However, one significant weakness of rDG methods is a lack of flexibility. For instance in the case of DG(P1), due to the embedded 1-D interpolation problem, a cubic basis at an interface is required in the direction ζ connecting two centroids of the two cells adjacent to that interface. In 2D, two more basis (out of six degrees of freedom) is needed in the direction η normal to ζ. In 3D, four more basis (out of eight degrees of freedom) is required in two directions normal to ζ. This makes the choice of the recovery basis nontrivial. In addition, the basis of the recovered solution is twice as large as the basis of the underlying DG solution. This higher-order representation of the recovered solution indeed yields higher-order accuracy for the diffusion equations on regular grids, and however is unnecessary for the Navier–Stokes equations, as the overall order of accuracy is determined not only by diffusive fluxes but also advective fluxes. The fundamental issue is how to judiciously choose a proper form of a recovered polynomial in such a way that the resulting recovered linear system is well conditioned, and thus can be inverted. Clearly, this lack of flexibility makes rDG methods less appealing for the multi-dimensional problems. Fortunately, recovery is not the only way to obtain a locally smooth polynomial solution at the interface 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.

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 ($m \geqslant n$) 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 ($\mathrm{m} \geqslant 1$) 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. 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. Dumbser [20] has applied this *PnPm* method to solve the compressible Navier–Stokes equations on unstructured grids, where the discretization of the viscous fluxes is based on the exact solution of the diffusive generalized Riemann problem developed by Gassner et al. [27].

The objective of the effort discussed in this paper is to develop a reconstructed discontinuous Galerkin method, originally introduced for the compressible Euler equations using an in-cell reconstruction [18,35], for the solution of the compressible Navier–Stokes equations using an inter-cell reconstruction. Starting from an underlying discontinuous DG solution, a smooth solution is locally reconstructed on the union of two cells adjacent to an interface to compute the viscous and heat fluxes in the Navier–Stokes equations at the interface. The reconstructed continuous solution is required to be conservative and to match the point values of the underlying DG solution and its derivatives at the two neighboring cells. The resultant over-determined system is then solved using a least-squares method. Similar to the recovery-based DG (rDG) methods, this reconstructed DG method automatically generates penalty or coupling terms found in traditional DG methods, and thus is stable, and compact. Unlike rDG methods, this RDG does not need to judiciously choose a proper form of a recovered polynomial, thus is simple, flexible, and robust, and can be implemented on arbitrary grids. 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 method is able to deliver the same accuracy, convergence, and stability as the well-known Bassi–Rebay II scheme, at a half of its computing costs for the discretization of the viscous terms in the Navier–Stokes equations, leading us to believe its potential to replace the existing DG methods for the discretization of the Navier–Stokes equations. 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. 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

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

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

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

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 $S = 110$ 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 W, integrating over the domain Ω, and then performing an integration by parts,

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

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 $\boldsymbol{V}_h^p$

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

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

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

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

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

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,

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

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 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 can be 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 3, 6, and 13 points for triangular elements and four, nine, and 16 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 follows:

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

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

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

Fig. 1. Representation of polynomial solutions using finite element shape functions.

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

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

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

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

where $\widetilde{\mathbf{U}}$ 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 are

$$\begin{aligned} &B_1 = 1, \quad B_2 = x - x_c, \quad B_3 = y - y_c, \quad B_4 = \frac{(x-x_c)^2}{2} - \frac{1}{\Omega_e}\int_{\Omega_e} \frac{(x-x_c)^2}{2} d\Omega, \\ &B_5 = \frac{(y-y_c)^2}{2} - \frac{1}{\Omega_e}\int_{\Omega_e} \frac{(y-y_c)^2}{2}\, d\Omega, \quad B_6 = (x-x_c)(y-y_c) - \frac{1}{\Omega_e}\int_{\Omega_e} (x-x_c)(y-y_c) d\Omega. \end{aligned} \tag{3.9}$$

The discontinuous Galerkin formulation then leads to the following six equations

$$\begin{aligned} &\frac{d}{dt}\int_{\Omega_e} \widetilde{\mathbf{U}}\, d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k\, d\Gamma = \int_{\Gamma_e} \mathbf{G}_k(\mathbf{U}_h)\mathbf{n}_k\, d\Gamma, \quad i = 1, \\ &M_{5\times 5}\frac{d}{dt}\left(\left.\frac{\partial \mathbf{U}}{\partial x}\right|_c \quad \left.\frac{\partial \mathbf{U}}{\partial y}\right|_c \quad \left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_c \quad \left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_c \quad \left.\frac{\partial^2 \mathbf{U}}{\partial x \partial y}\right|_c\right)^T + R_{5\times 1} = 0, \end{aligned} \tag{3.10}$$

where the element of the mass matrix $m_{ij} = \int_{\Omega_e} B_{i+1} B_{j+1}\, d\Omega$. 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

$$\int_{\Omega_e} B_1 B_i\, d\Omega = 0, \quad 2 \leqslant i \leqslant 6. \tag{3.11}$$

This set of basis functions has a considerable disadvantage, especially when higher-order polynomial solutions are considered. Occasionally an inverse of the mass matrix M may not exist. 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:

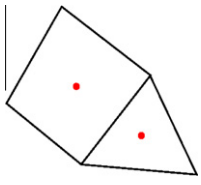

Fig. 2. Representation of polynomial solutions using a Taylor series expansion.

$$
\begin{aligned}
&B_1 = 1, \quad B_2 = \frac{x - x_c}{\Delta x}, \quad B_3 = \frac{y - y_c}{\Delta y}, \quad B_4 = \frac{(x - x_c)^2}{2\Delta x^2} - \frac{1}{\Omega_e}\int_{\Omega_e} \frac{(x - x_c)^2}{2\Delta x^2}\, d\Omega, \\
&B_5 = \frac{(y - y_c)^2}{2\Delta y^2} - \frac{1}{\Omega_e}\int_{\Omega_e} \frac{(y - y_c)^2}{2\Delta y^2}\, d\Omega, \quad B_6 = \frac{(x - x_c)(y - y_c)}{\Delta x \Delta y} - \frac{1}{\Omega_e}\int_{\Omega_e} \frac{(x - x_c)(y - y_c)}{\Delta x \Delta y}\, d\Omega,
\end{aligned} \tag{3.12}
$$

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 as

$$
\mathbf{U}_h = \widetilde{\mathbf{U}} + \left.\frac{\partial \mathbf{U}}{\partial x}\right|_c \Delta x B_2 + \left.\frac{\partial \mathbf{U}}{\partial y}\right|_c \Delta y B_3 + \left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_c \Delta x^2 B_4 + \left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_c \Delta y^2 B_5 + \left.\frac{\partial^2 \mathbf{U}}{\partial x \partial y}\right|_c \Delta x \Delta y B_6. \tag{3.13}
$$

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, 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. In addition, the Taylor-basis DG method makes the implementation of both in-cell and inter-cell reconstruction schemes straightforward and simple [35].

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 DG method, a continuous solution $\mathbf{U}^R$ is locally reconstructed on the union of two cells $\Omega_{ij}(=\Omega_i \cup \Omega_j)$ adjacent to the interface based on the underlying 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 consider the case of DG(P2) 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:

$$
\mathbf{U}^R = \widetilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_2(\mathbf{x}) + \mathbf{U}_y^{ij} B_3(\mathbf{x}) + \mathbf{U}_{xx}^{ij} B_4(\mathbf{x}) + \mathbf{U}_{yy}^{ij} B_5(\mathbf{x}) + \mathbf{U}_{xy}^{ij} B_6(\mathbf{x})_{xy}^{ij} B_6(\mathbf{x}), \tag{3.14}
$$

where $\widetilde{\mathbf{U}}_{ij}$ 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 $\widetilde{\mathbf{U}}_{ij}$ 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

$$
\widetilde{\mathbf{U}}_{ij} = \frac{\widetilde{\mathbf{U}}_i \Omega_i + \widetilde{\mathbf{U}}_j \Omega_j}{\Omega_i + \Omega_j}. \tag{3.15}
$$

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

$$
\begin{aligned}
&\mathbf{U}_i = \widetilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_{2i} + \mathbf{U}_y^{ij} B_{3i} + \mathbf{U}_{xx}^{ij} B_{4i} + \mathbf{U}_{yy}^{ij} B_{5i} + \mathbf{U}_{xy}^{ij} B_{6i}, \\
&\left.\frac{\partial \mathbf{U}}{\partial x}\right|_i \Delta x_i = \left(\mathbf{U}_x^{ij} \frac{\partial B_{2i}}{\partial x} + \mathbf{U}_{xx}^{ij} \frac{\partial B_{4i}}{\partial x} + \mathbf{U}_{xy}^{ij} \frac{\partial B_{6i}}{\partial x}\right) \Delta x_i, \\
&\left.\frac{\partial \mathbf{U}}{\partial y}\right|_i \Delta y_i = \left(\mathbf{U}_y^{ij} \frac{\partial B_{3i}}{\partial y} + \mathbf{U}_{yy}^{ij} \frac{\partial B_{5i}}{\partial y} + \mathbf{U}_{xy}^{ij} \frac{\partial B_{6i}}{\partial y}\right) \Delta y_i, \\
&\left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_i \Delta x_i^2 = \mathbf{U}_{xx}^{ij} \frac{\partial^2 B_{4i}}{\partial x^2} \Delta x_i^2, \\
&\left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_i \Delta y_i^2 = \mathbf{U}_{yy}^{ij} \frac{\partial^2 B_{5i}}{\partial y^2} \Delta y_i^2, \\
&\left.\frac{\partial^2 \mathbf{U}}{\partial x \partial y}\right|_i \Delta x_i \Delta y_i = \mathbf{U}_{xy}^{ij} \frac{\partial^2 B_{6i}}{\partial x \partial y} \Delta x_i \Delta y_i,
\end{aligned} \tag{3.16}
$$

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:

$$\begin{pmatrix} B_{2i} & B_{3i} & B_{4i} & B_{5i} & B_{6i} \\ \frac{\partial B_{2i}}{\partial x}\Delta x_i & 0 & \frac{\partial B_{4i}}{\partial x}\Delta x_i & 0 & \frac{\partial B_{6i}}{\partial x}\Delta x_i \\ 0 & \frac{\partial B_{3i}}{\partial y}\Delta y_i & 0 & \frac{\partial B_{5i}}{\partial y}\Delta y_i & \frac{\partial B_{6i}}{\partial y}\Delta y_i \\ 0 & 0 & \frac{\partial^2 B_{4i}}{\partial x^2}\Delta x_i^2 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial^2 B_{5i}}{\partial y^2}\Delta y_i^2 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial B_{6i}}{\partial x\,\partial y}\Delta x_i\Delta y_i \end{pmatrix} \begin{pmatrix} \mathbf{U}_x^{ij} \\ \mathbf{U}_y^{ij} \\ \mathbf{U}_{xx}^{ij} \\ \mathbf{U}_{yy}^{ij} \\ \mathbf{U}_{xy}^{ij} \end{pmatrix} = \begin{pmatrix} \mathbf{U}_i = \widetilde{\mathbf{U}}_{ij} \\ \frac{\partial \mathbf{U}}{\partial x}\Big|_i \Delta x_i \\ \frac{\partial \mathbf{U}}{\partial y}\Big|_i \Delta y_i \\ \frac{\partial^2 \mathbf{U}}{\partial x^2}\Big|_i \Delta x_i^2 \\ \frac{\partial^2 \mathbf{U}}{\partial y^2}\Big|_i \Delta y_i^2 \\ \frac{\partial^2 \mathbf{U}}{\partial x \partial y}\Big|_i \Delta x_i \Delta y_i \end{pmatrix}. \tag{3.17}$$

Similar equations can be derived for the cell j. As a result, a non-square matrix of 12×5 is obtained. This over-determined linear system of 12 equations for five unknowns can then 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 ensure that the recovered system is well conditioned and can be inverted. The resulting DG method for the discretization of the viscous and heat fluxes is termed a reconstructed DG method (RDG(P2) in short notation). 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 (3.14), which reads

$$\mathbf{U}^R = \widetilde{\mathbf{U}}_{ij} + \mathbf{U}_x^{ij} B_2(\mathbf{x}) + \mathbf{U}_y^{ij} B_3(\mathbf{x}) + \mathbf{U}_{xx}^{ij} B_4(\mathbf{x}) + \mathbf{U}_{yy}^{ij} B_5(\mathbf{x}) + \mathbf{U}_{xy}^{ij} B_6(\mathbf{x}) + \mathbf{U}_{\xi\xi\xi}^{ij} B_7(\xi), \tag{3.18}$$

where

$$\begin{aligned} B_7(\xi) &= \frac{\xi^3}{6\Delta\xi^3} - \frac{1}{\Omega_{ij}} \int_{\Omega_{ij}} \frac{\xi^3}{6\Delta\xi^3}\, d\Omega, \\ \xi &= (\mathbf{x} - \mathbf{x}_{ij}) \cdot \mathbf{n} = (x - x_{ij}) n_x + (y - y_{ij}) n_y. \end{aligned} \tag{3.19}$$

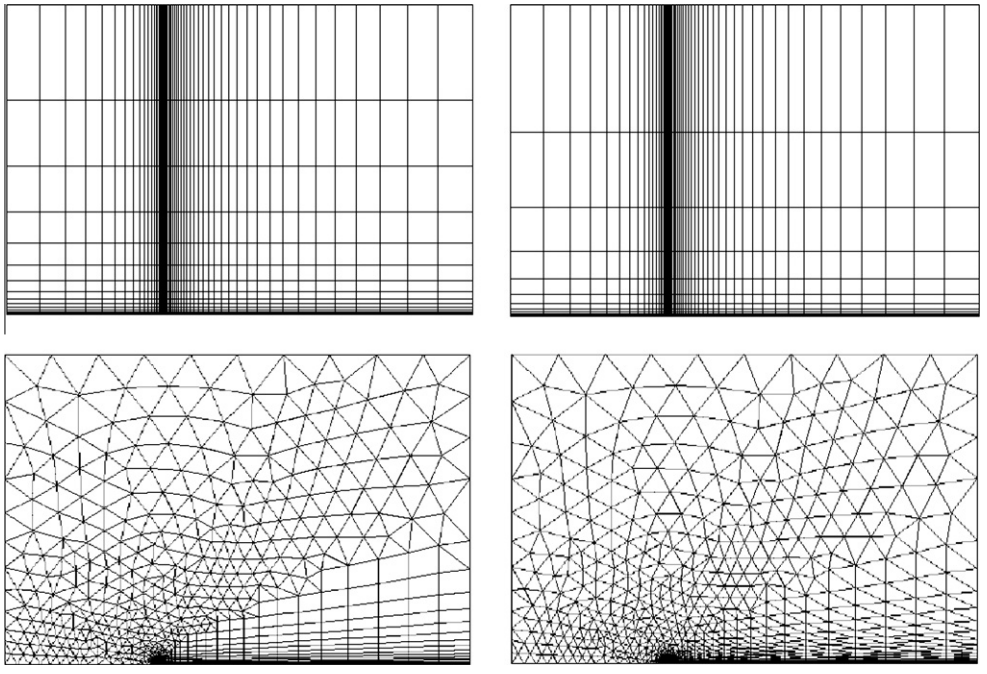

Fig. 3. Logarithmic plot of the computed skin friction coefficient distribution along the flat plate obtained by the RDG(P1), RDG(P1+), and LDG(P1) solutions on the quadrilateral grid with a stretching ratio of 1.2 (top left), the quadrilateral grid with a stretching ratio of 1.3 (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

This inter-cell reconstruction leads to an over-determined system of 12 equations with six 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(P2+) method from now on, where + indicates that the reconstructed polynomial solutions contain a higher-order term in the normal direction to the interface.

The inter-cell reconstruction described above can be applied to the boundary cells in a straightforward manner with the aid of a so-called ghost cell approach, where the computational domain is extended to a set of ghost cells. The values of the flow variables in the ghost cells are determined by the desired boundary conditions. The compactness of the DG methods makes the imposition of boundary conditions simple and easy, which only requires one layer of ghost cells, regardless of the order of the DG methods.

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

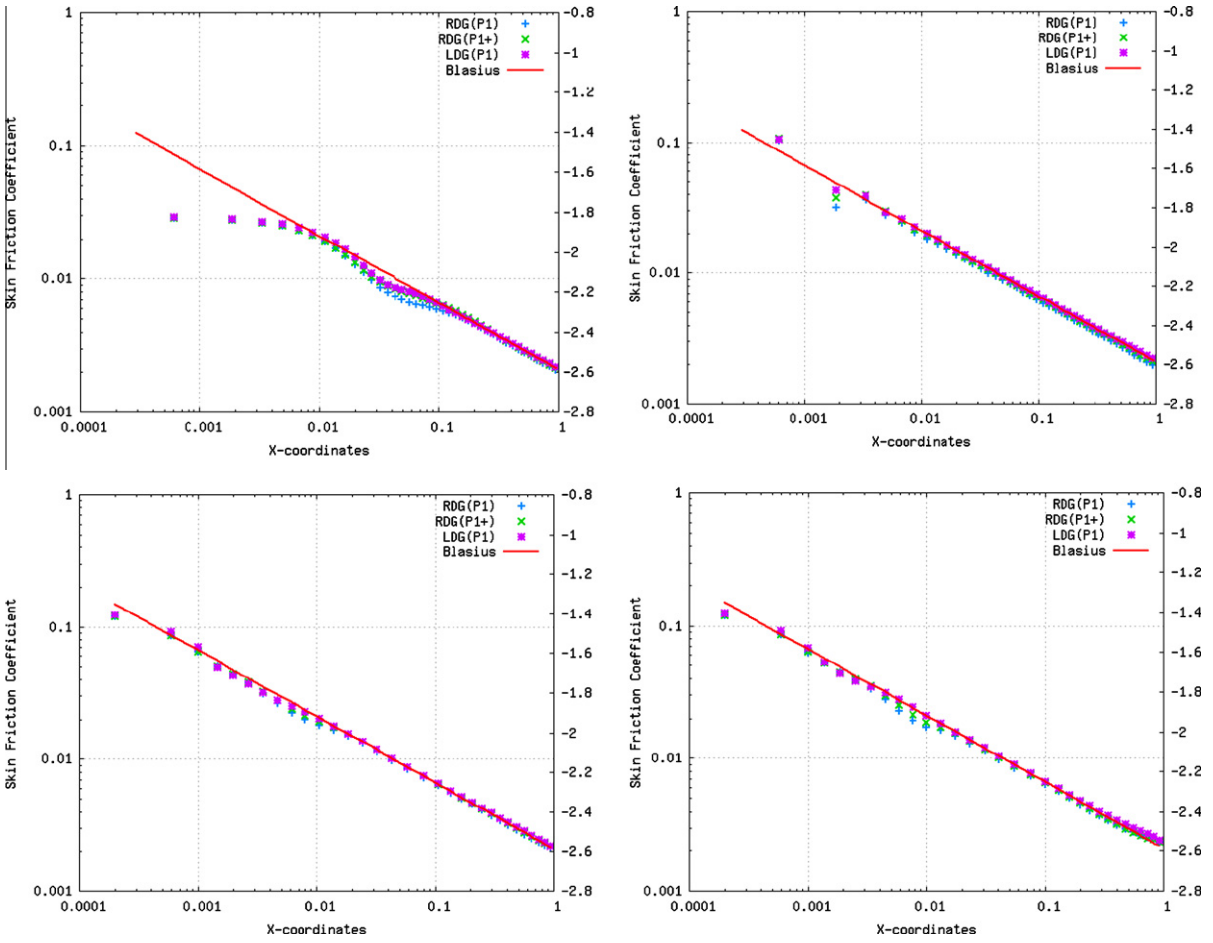

Fig. 4. Logarithmic plot of the computed skin friction coefficient distribution along the flat plate obtained by the RDG(P1), RDG(P1+), and BR2(P1) solutions on the quadrilateral grid with a stretching ratio of 1.2 (top left), the quadrilateral grid with a stretching ratio of 1.3 (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

solution of the Riemann problem in this work. Among many possible schemes that developed for the discretization of the viscous fluxes, we chose to implement the second Bassi–Rebay scheme (BR2) [22] for the discretization of the viscous fluxes, as BR2 scheme is the only one proposed in the literature to achieve optimal order of accuracy and compact. This allows us to use BR2 scheme as a reference to compare the accuracy and performance of the RDG method.

Our numerical results have shown that the computing costs required by this RDG method is about half of the computing costs by BR2 scheme for the discreitization of the viscous fluxes in the Navier–Stokes equations. This is mainly due the fact that the BR2 scheme requires to solve a linear system. The formation of this linear system involves the boundary integrals using Gauss quadrature formulas, and thus is computationally intensive. The RDG method only involves the solution of a least-squares problem at an inter-cell interface. The resulting reconstruction matrix can be pre-computed, inverted, and stored at the start of the computation, thus leading to a very efficient discretization method for the viscous fluxes. Note that this is achieved at the expense of an increasing storage requirement. In the present work, the reconstruction matrix is not pre-computed and stored in order to avoid the significant increase in memory requirements.

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.

4.1. 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 free-stream 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

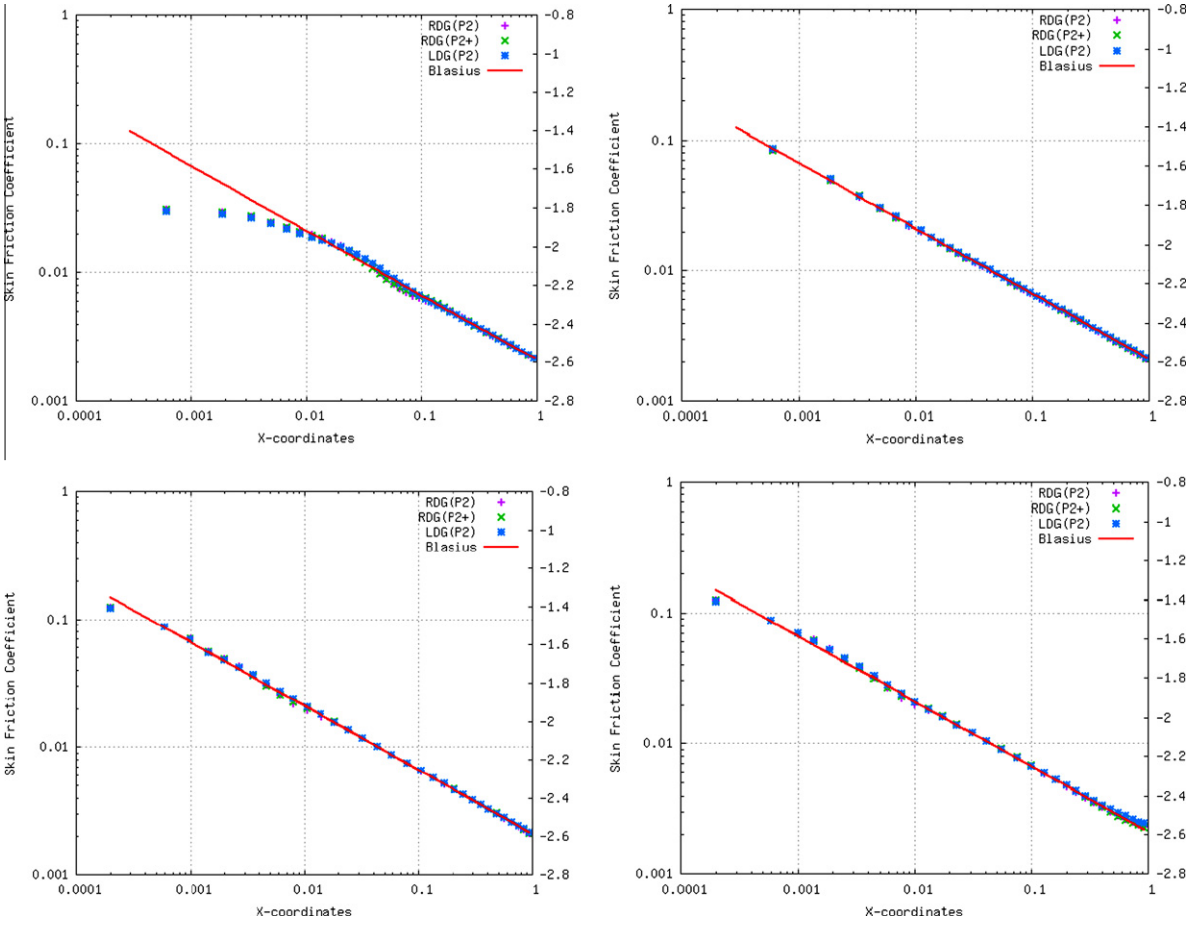

Fig. 5. Logarithmic plot of the computed skin friction coefficient distribution along the flat plate obtained by the RDG(P2), RDG(P2+), and BR2(P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

(0,0) and extends to (1,0). This problem is chosen to illustrate the accuracy of the inter-cell reconstruction scheme for the discretization of the viscous and heat fluxes in 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. 3, to assess the accuracy and consistence of the reconstructed discontinuous Galerkin 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. 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 on 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(P1), RDG(P1+), BR(P1), RDG(P2), RDG(P2+), and BR(P2) on these four grids are presented, and compared with the theoretical one given by the well-known Blasius solution. Figs. 4 and 5 show the logarithmic plot of the computed skin friction coefficient obtained by the linear and quadratic DG solutions, respectively. Figs. 6 and 7 compare the profiles of velocity component in the x-direction at $x = 0.2$ in the boundary layer obtained by DG(P1) and DG(P2) solutions with Blasius solution, respectively, while the velocity profiles in the y-direction obtained by DG(P1) and DG(P2) solutions are compared with Blasius solution in Figs. 8 and 9, respectively. Comparing the numerical solutions on the two quadrilateral grids, one can observe a consistent convergence of both reconstructed DG and BR2 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

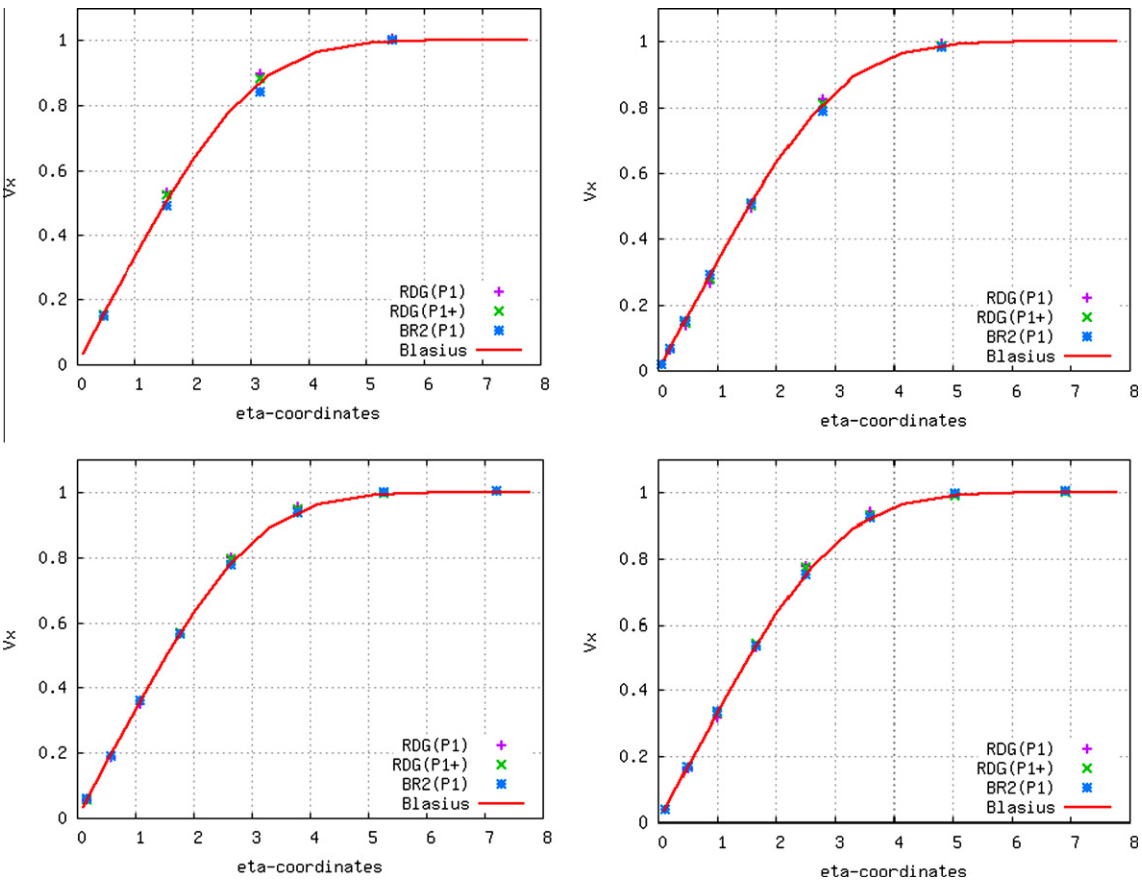

Fig. 6. Comparison of the velocity profiles in the x-direction at $x = 0.2$ in the boundary layer obtained by the RDG(P1), RDG(P1+), and BR2(P1) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

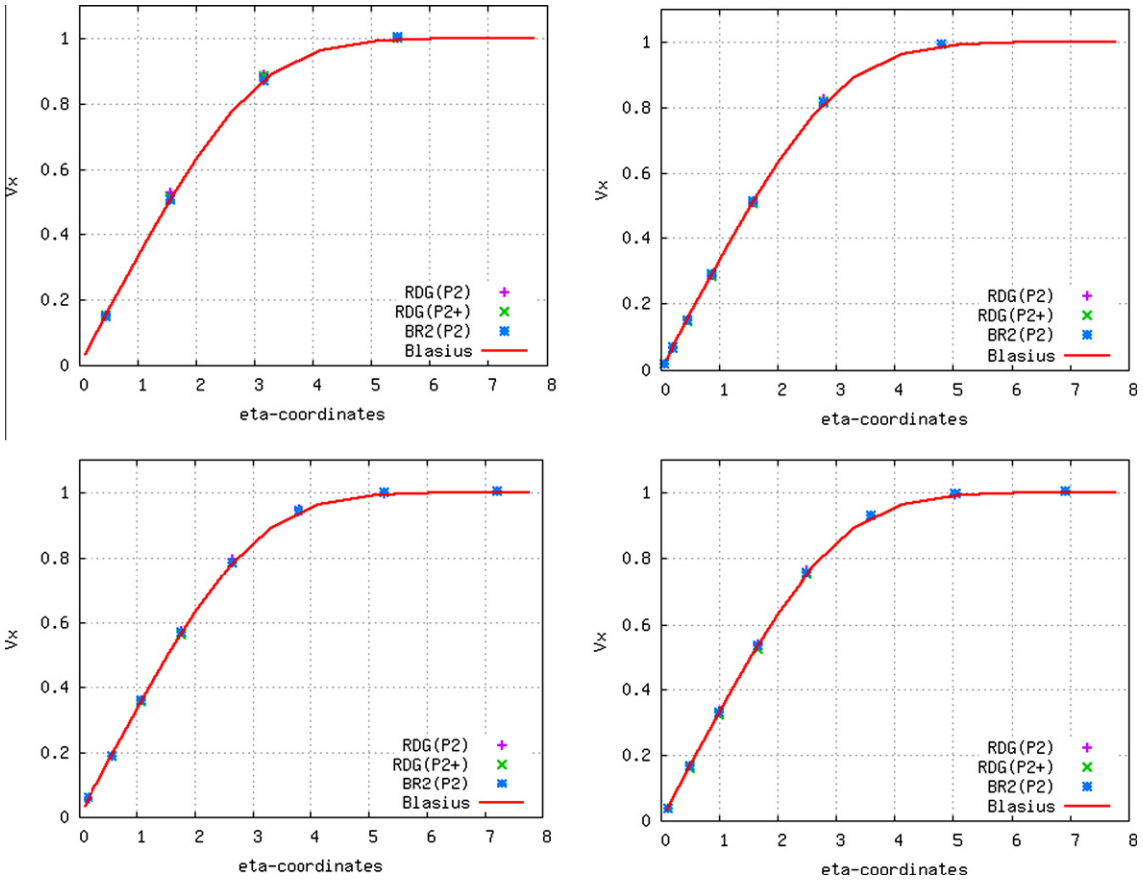

Fig. 7. Comparison of the velocity profiles in the x-direction at $x = 0.2$ in the boundary layer obtained by the RDG(P2), RDG(P2+), and BR2(P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

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. As expected, both RDG(P1+) and RDG(P2+) methods are more accurate than RDG(P1) and RDG(P2), respectively, due to the additional higher-order term in the reconstructed in-cell polynomial solutions. This is especially evident, when one compares the y-component velocity profiles obtained by RDG(P2) and RDG(P2+) methods. The numerical solutions obtained by the RDG(+) method are as accurate as, if not more than, the ones produced by the RB2 scheme, demonstrating that the developed reconstructed DG method is able to deliver the same accuracy, convergence, and stability as the well-known Bassi–Rebay II 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. In order to compare computing costs required by the RDG and Bassi–Rebay II methods for the discretization of the viscous terms, a profile analysis is performed on the hybrid grid. Tables 1 and 2 list the execution profiles obtained for both RDG and Bassi–Rebay II method, where three-stage third-order TVD Runge–Kutta scheme is used to advance solution in time. The name, rhsbounp1_hllc_br denotes the subroutine to obtain the right-hand-side for the boundary integral where HLLC and Bassi–Rebay II schemes are used to compute the inviscid, and viscous fluxes, respectively. The rhsdomn_br_p and rhsdomn_br_q are the subroutine names to compute the right-hand-side from the domain integral for the triangular and quadrilateral cells, respectively. The exupwi2d is the subroutine that initializes the RHS, calls these two subroutines, and obtains the solution at the new stages for the Runge–Kutta method. Note that the CPU time used to compute the numerical fluxes at the interfaces includes the one spent on computation of the inviscid fluxes. Excluding that computing costs, the CPU time required by the RDG method is about half of the one required by BR2 scheme for the discretization of the viscous fluxes in the Navier–Stokes equations.

4.2. 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 free-stream velocity and the chord length of the airfoil. An adiabatic wall is

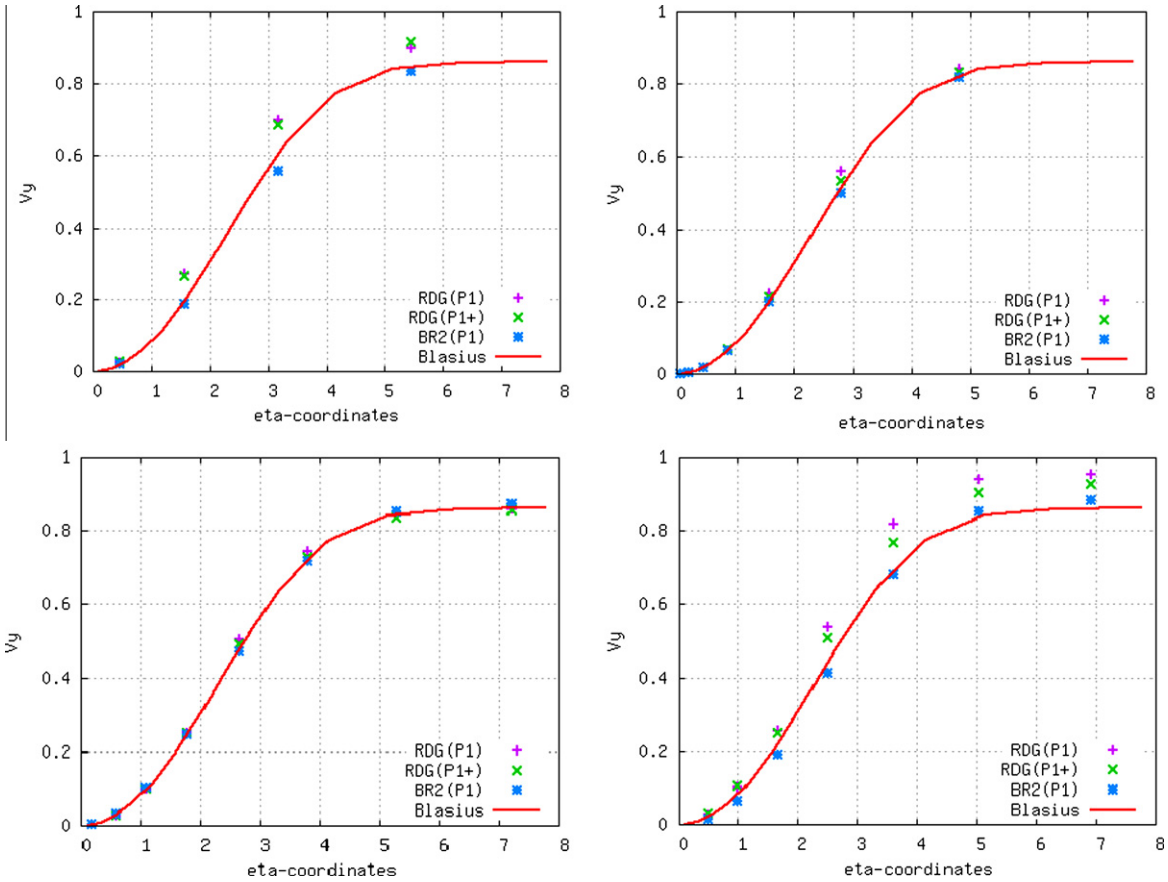

Fig. 8. Comparison of the velocity profiles in the y-direction at $x = 0.2$ in the boundary layer obtained by the RDG(P1), RDG(P1+), and BR2(P1) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

Table 1
Execution Profile for the second-order DG method on hybrid grid.

% Cumulative		Self		Self	Total	Name
Time	Seconds	Seconds	Calls	Ks/call	Ks/call	
Flat profile: (DG(P1) (BR2))						
68.91	27.08	27.08	12000	0.00	0.00	rhsbounp1_hllc_br2_
9.92	30.98	3.90	12000	0.00	0.00	rhsdomnp1_br2_p_
3.16	32.22	1.24	12000	0.00	0.00	rhsdomnp1_br2_q_
1.40	32.77	0.55	12000	0.00	0.00	rhsdomnp1_br2_
Flat profile: (RDG(P1+))						
66.77	20.84	20.84	12000	0.00	0.00	rhsbounp1_hllc_rdg_
12.30	24.68	3.84	12000	0.00	0.00	rhsdomnp1_rdg_p_
3.75	25.85	1.17	12000	0.00	0.00	rhsdomnp1_rdg_q_
0.00	25.85	0.00	12000	0.00	0.00	rhsdomnp1_rdg_

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 DG(BP2), and RDG(P2+) methods. Fig. 10 shows the computational grid used in this test case, consisting of 2495 triangular elements, 549 quadrilateral elements, 1856 grid points, and 121 boundary faces, and the computed Mach number contours in the flow field obtained by RDG(P2+) 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. 11. The computed skin friction coefficients and pressure coefficients obtained by RDG(P2+), and BR2(P2) are compared in Fig. 12, where the two solutions are virtually identical in this test case, again demonstrating that the developed reconstructed DG method is able to deliver the same accuracy as the well-established Bassi–Rebay II scheme.

Table 2
Execution profile for the third-order DG method on hybrid grid.

% Cumulative		Self		Self	Total	Name
Time	Seconds	Seconds	Calls	Ks/call	Ks/call	
Flat profile: (DG(P2) (BR2))						
66.58	64.24	64.24	12000	0.01	0.01	rhsbounp2_hllc_br2_
11.58	75.41	11.17	12000	0.00	0.00	rhsdomnp2_br2_p_
5.83	81.03	5.62	12000	0.00	0.00	rhsdomnp2_br2_q_
3.82	84.72	3.69	4000	0.00	0.02	exupwi2d_
3.46	88.06	3.34	12000	0.00	0.00	rhsdomnp2_br2_
Flat profile: (RDG(P2+))						
61.44	43.41	43.41	12000	0.00	0.00	rhsbounp2_hllc_rdg_
13.74	53.12	9.71	12000	0.00	0.00	rhsdomnp2_rdg_p_
7.59	58.48	5.36	12000	0.00	0.00	rhsdomnp2_rdg_q_
6.03	62.74	4.26	4000	0.00	0.02	exupwi2d_

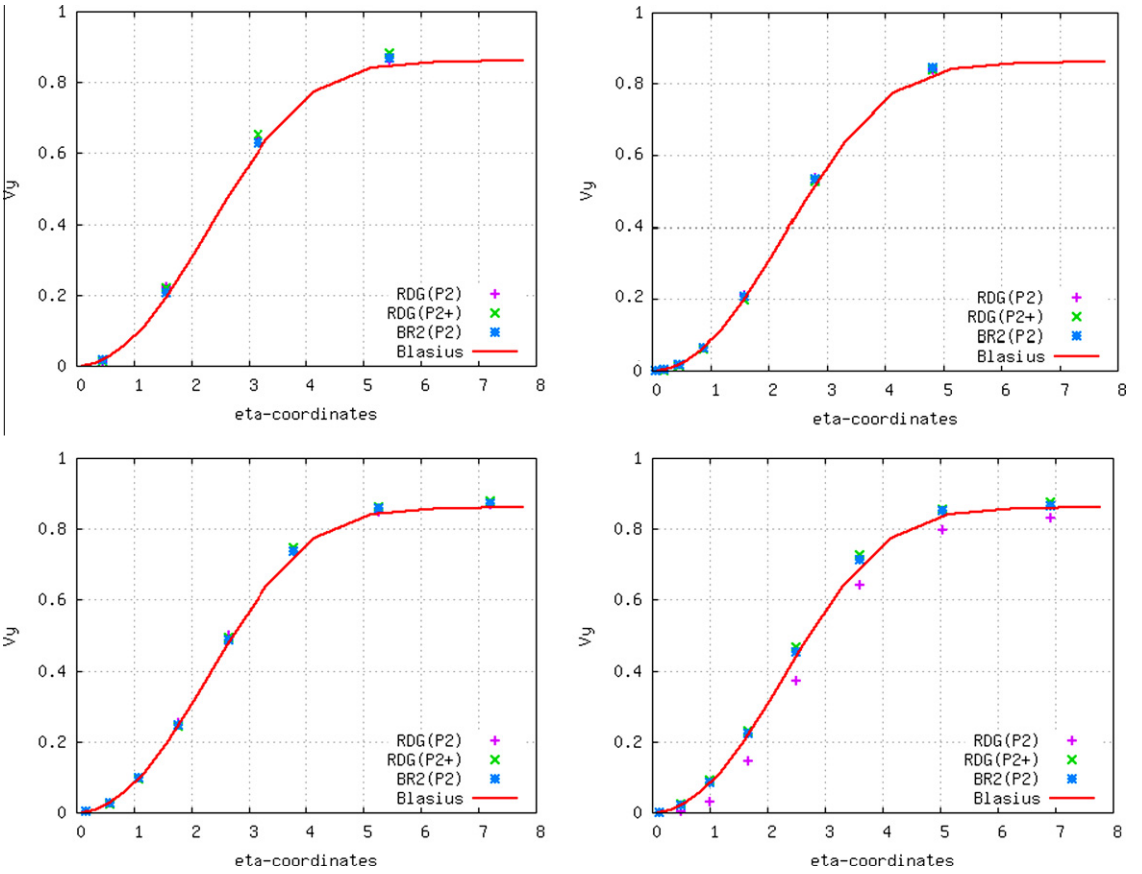

Fig. 9. Comparison of the velocity profiles in the y-direction at $x = 0.2$ in the boundary layer obtained by the RDG(P2), RDG(P2+), and BR2(P2) solutions on the quadrilateral grid with a stretching ratio of 1.2 in the y-direction (top left), the quadrilateral grid with a stretching ratio of 1.3 in the y-direction (top right), the hybrid grid (bottom left), and the triangular grid (bottom right).

4.3. Subsonic flow past a circular cylinder

The subsonic flow past a circular cylinder is considered in this test case. The initial condition is a uniform free-stream with non-slip boundary conditions on the solid wall. The mesh used in the computations consists of 21,809 triangular elements, 11,004 grid points, and 199 boundary faces with 105 points on the surface of the cylinder, as shown in Fig. 13. The cell size is $0.03d$ at the cylinder, where d is the diameter of the cylinder. Three computations are preformed using RDG(P1), RDG(P2+), and RB2(P1+) methods, respectively at a Reynolds number of 40 based on the diameter of the cylinder and at

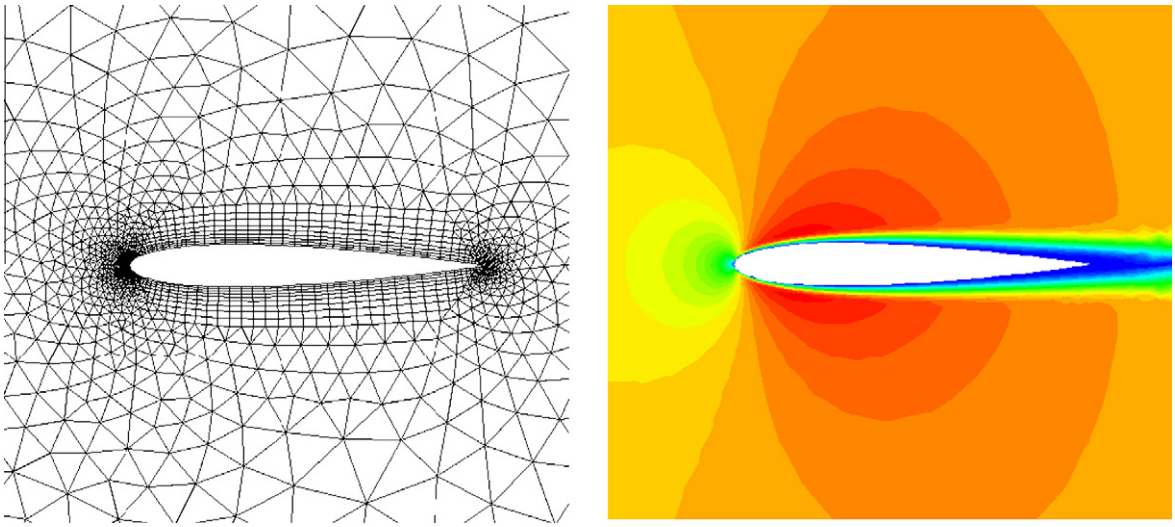

Fig. 10. Unstructured hybrid mesh (top left) (ntria = 3469, npoin = 3346, nbfac = 157) and computed Mach number contours by the RDG(P2+) (top right) for subsonic flow past a NACA0012 airfoil at $M_\infty = 0.5$, Re = 5,000, $\alpha = 0°$.

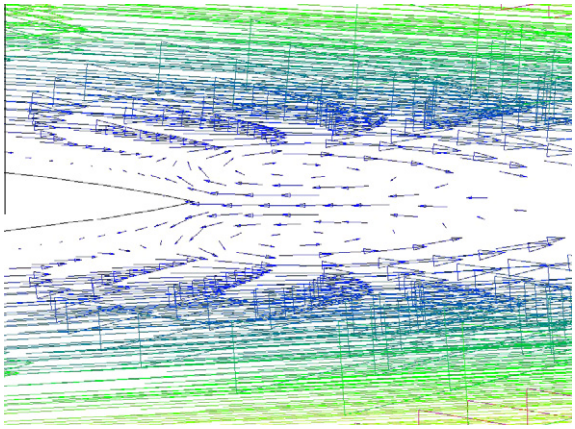

Fig. 11. Velocity vector plot in the vicinity of the trailing edge for subsonic flow past a NACA0012 airfoil at $M_\infty = 0.5$, Re = 5000, $\alpha = 0°$.

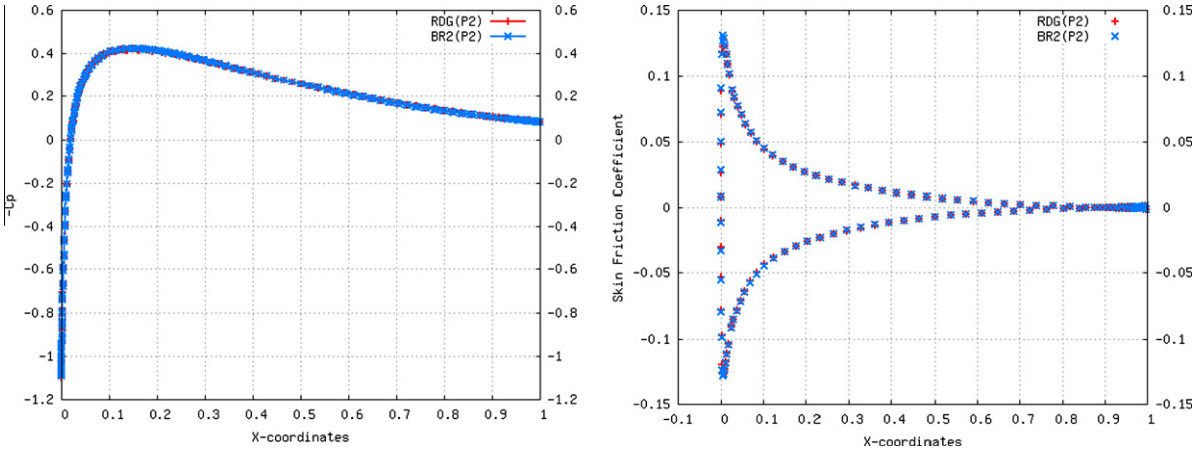

Fig. 12. Computed skin friction coefficient distributions (left) and pressure coefficients (right) on the airfoil obtained by the RDG(P2+), and BR2(P2) methods.

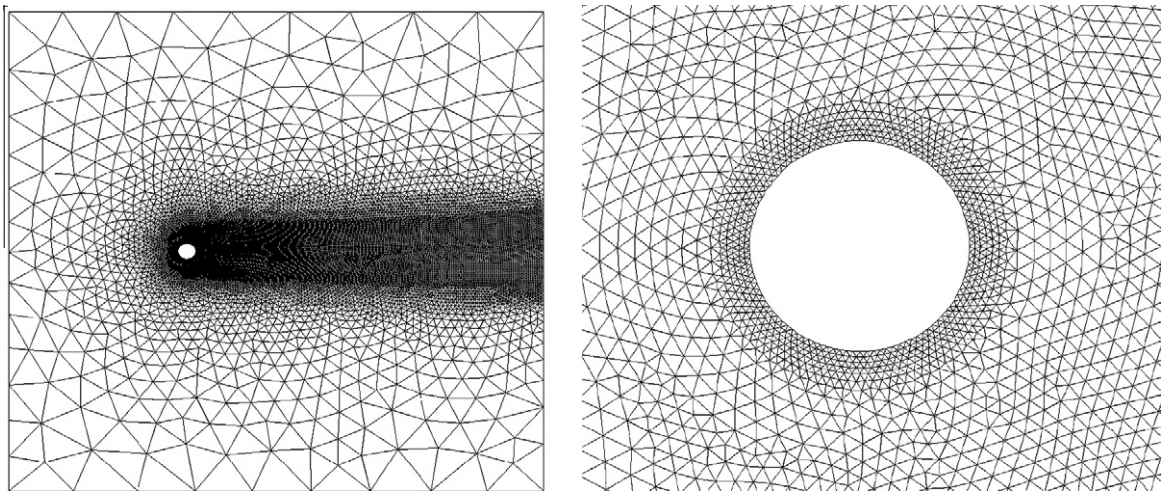

Fig. 13. Mesh used for computing a viscous flow past a cylinder (nelem = 21,809, npoin = 11,004, nboun = 199).

a Mach number of 0.1. The computed Mach number contours in the flow field and the velocity vector plot behind the cylinder obtained by RDG(P2+) method are displayed in Figs. 14 and 15 respectively. The length of the recirculation bubble, and the pressure and viscous drag coefficients for these three simulations are summarized in Table 3. One can observe that the difference among these three solutions is relatively small, indicating that the obtained solution is order-independent, i.e., the solution is convergent. Note that these numerical results are in good agreement with those reported in [36].

4.4. Subsonic flow past a SD7003 airfoil

A viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 0 degree, and a Reynolds number of 10,000 is considered in this case to illustrate that the developed RDG method can be used to accurately solve unsteady flow problems. The computation is initialized with constant free-stream values in the entire domain with non-slip boundary conditions on the solid wall. Fig. 16 shows the hybrid mesh used in the computation, which consists of 23,172 triangular elements, 2225 quadrilateral elements, 25,397 grid points, and 279 boundary faces with 200 grid points on the surface of the airfoil. The computation is performed using RDG(P2+) method. Typical computed pressure and vorticity contours in the flow

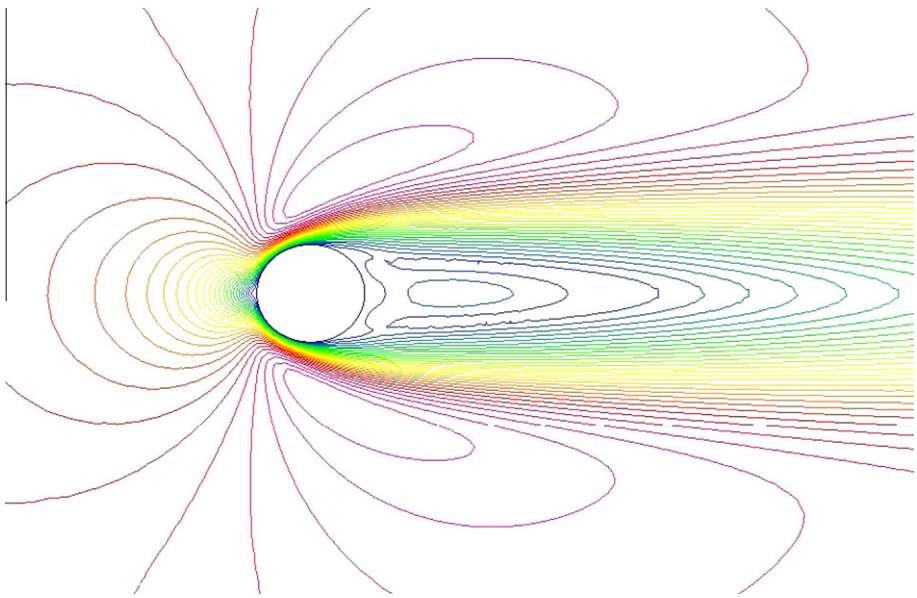

Fig. 14. Computed Mach number contours for flow past a cylinder at a Mach number of 0.1, and a Reynolds number of 40.

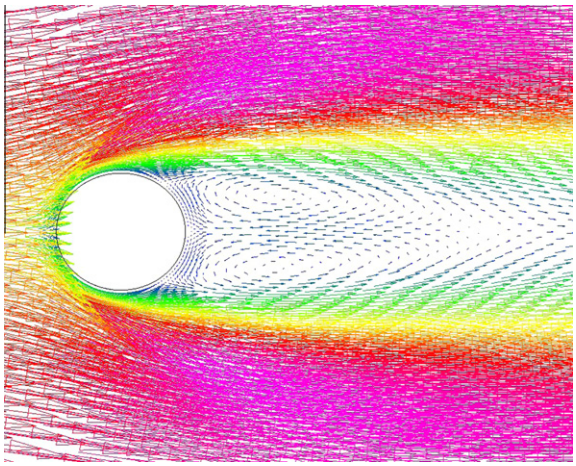

Fig. 15. Velocity vector plot behind the cylinder for subsonic flow past a circular cylinder at a Mach number of 0.1, and a Reynolds number of 40.

Table 3
Comparison of the recirculation bubble, and the pressure and viscous drag coefficients.

	Pressure drag coefficients	Viscous drag coefficients	Length of recirculation
BRII(P1)	1.0264	0.49925	2.276
RDG(P1+)	1.0235	0.51216	2.276
RDG(P2+)	1.0163	0.52102	2.276

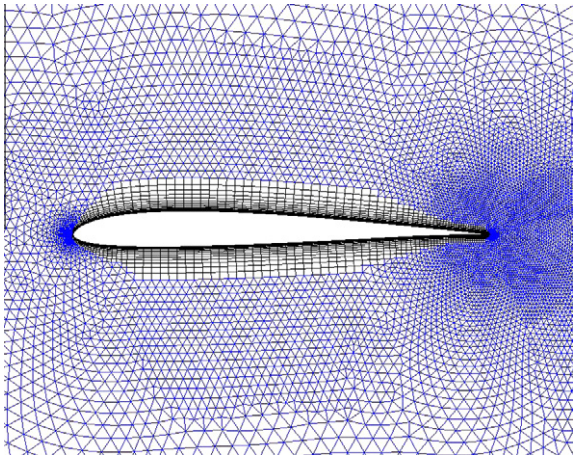

Fig. 16. Grid used for computing the unsteady viscous flow past a SD7003 airfoil (ntria = 23,172, nquad = 2225, npoin = 25,397, nboun = 279).

field are shown in Figs. 17 and 18, respectively, where one can see that the flow features, such as separation of the flow on the upper surface of the airfoil and shedding of the tailing vorticies are well captured by the present RDG(P2+) solution.

5. Conclusion

A reconstructed discontinuous Galerkin method has been developed for the solution of the compressible Navier–Stokes equations on arbitrary grids. A smooth solution is locally reconstructed to discretize viscous and heat fluxes in the Navier–Stokes equations using a least-squares method from the underlying discontinuous discrete solution. 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 method is able to deliver the same accuracy as the well-known Bassi–Rebay II scheme, at a half of its computing costs for the discretization of the viscous fluxes, indicating that this RDG method provides a viable, attractive, competitive, and perhaps superior DG method over existing DG methods for solving the compressible Navier–Stokes equations.

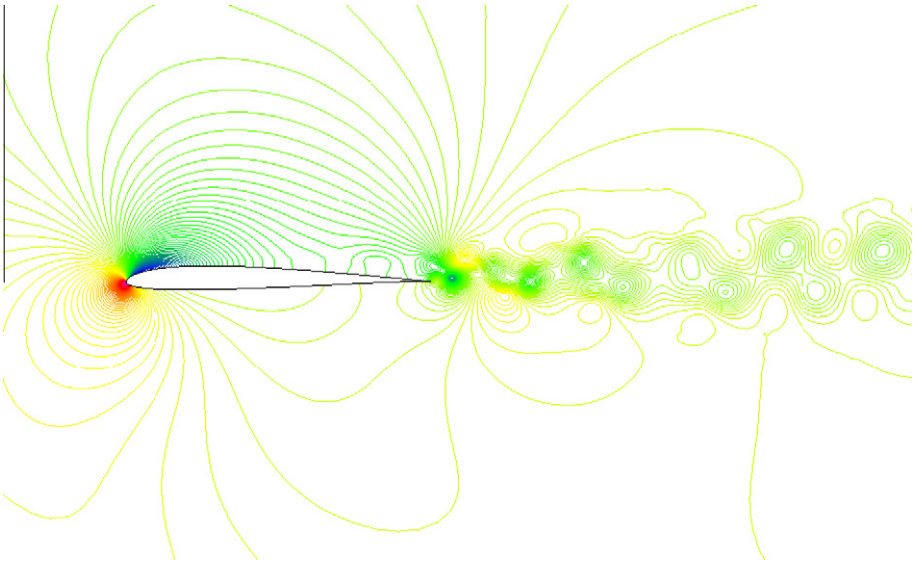

Fig. 17. Computed pressure contours in the flow field obtained by the RDG(P2+) method for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4°, and a Reynolds number of 10,000.

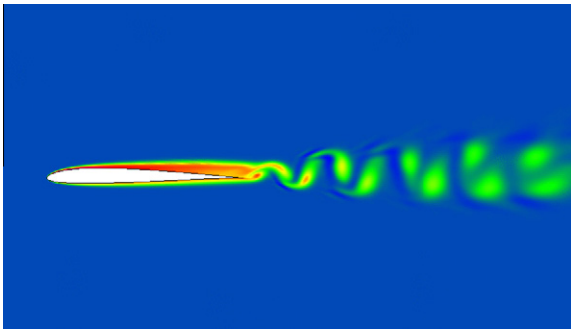

Fig. 18. Computed vorticity contours in the flow field obtained by the RDG(P2) method for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4°, and a Reynolds number of 10,000.

Acknowledgments

This manuscript has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/CON-10-17570) with the US 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 Reactor Safety Simulation Group, Idaho National Laboratory, Idaho Falls, ID.

References

[1] Reed, W.H. Reed, 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, C.W. Shu, TVD Runge–Kutta Local Projection Discontinuous Galerkin Finite Element Method for conservation laws IV: the Multidimensional Case, Mathematics of Computation 55 (1990) 545–581;
P.L. George, Automatic Mesh Generation, J. Wiley & Sons, 1991.
[3] B. Cockburn, C.W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: Multidimensional system, Journal of Computational Physics 141 (1998) 199–224.
[4] B. Cockburn, G. Karniadakis, C.W. Shu, The development of discontinuous Galerkin method, in: B. Cockburn, G.E. Karniadakis, C.W. Shu (Eds.), Discontinuous Galerkin Methods, Theory, Computation, and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer-Verlag, New York, 2000, pp. 5–50.
[5] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, Journal of Computational Physics 138 (1997) 251–285.
[6] H.L. Atkins, C.W. Shu, Quadrature free implementation of discontinuous Galerkin method for hyperbolic equations, AIAA Journal 36 (5) (1998).

[7] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations, in: B. Cockburn, G.E. Karniadakis, C.W. Shu (Eds.), Discontinuous Galerkin Methods, Theory, Computation, and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer-Verlag, New York, 2000, pp. 197–208.
[8] T.C. Warburton, G.E. Karniadakis, A discontinuous Galerkin method for the viscous MHD equations, Journal of Computational Physics 152 (1999) 608–641.
[9] J.S. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Texts in Applied Mathematics 56 (2008).
[10] P. Rasetarinera, M.Y. Hussaini, An efficient implicit discontinuous spectral Galerkin method, Journal of Computational Physics 172 (2001) 718–738.
[11] B.T. Helenbrook, D. Mavriplis, 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, D.L. Darmofal, *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, Journal of Computational Physics 207 (1) (2005) 92–113.
[13] H. Luo, J.D. Baum, R. Löhner, A discontinuous Galerkin method using Taylor basis for compressible flows on arbitrary grids, Journal of Computational Physics 227 (20) (2008) 8875–8893. 210.1016/j.jcp.2008.06.035.
[14] H. Luo, J.D. Baum, R. Löhner, On the computation of steady-state compressible flows using a discontinuous Galerkin method, International Journal for Numerical Methods in Engineering 73 (5) (2008) 597–623.
[15] H. Luo, J.D. Baum, R. Löhner, A hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids, Journal of Computational Physics 225 (1) (2007) 686–713.
[16] H. Luo, J.D. Baum, R. Löhner, A *p*-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, Journal of Computational Physics 211 (2) (2006) 767–783.
[17] H. Luo, J.D. Baum, R. Löhner, A fast *p*-multigrid discontinuous Galerkin method for compressible flows at all speeds, AIAA Journal 46 (3) (2008) 635–652.
[18] M. Dumbser, D.S. Balsara, E.F. Toro, 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 227 (2008) 8209–8253.
[19] M. Dumbser, O. Zanotti, Very high order PNPM schemes on unstructured meshes for the resistive relativistic MHD equations, Journal of Computational Physics 228 (2009) 6991–7006.
[20] M. Dumbser, Arbitrary high order PNPM schemes on unstructured meshes for the compressible Navier–Stokes equations, Computers and Fluids 39 (2010) 60–76.
[21] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, Journal of Computational Physics 131 (1997) 267–279.
[22] F. Bassi, S. Rebay, Discontinuous Galerkin solution of the Reynolds-averaged Navier–Stokes and k–ω turbulence model equations, Journal of Computational Physics 34 (2005) 507–540.
[23] B. Cockburn, C.W. Shu, The local discontinuous Galerkin method for time-dependent convection–diffusion system, SIAM Journal of Numerical Analysis 16 (2001).
[24] C.E. Baumann, J.T. Oden, A discontinuous *hp* finite element method for the Euler and Navier–Stokes equations, International Journal for Numerical Methods in Fluids 31 (1999).
[25] J. Peraire, P.O. Persson, The compact discontinuous Galerkin method for elliptic problems, SIAM Journal on Scientific Computing 30 (2008) 1806–1824.
[26] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM Journal on Numerical Analysis 39 (5) (2002) 1749–1779.
[27] G. Gassner, F. Lorcher, C.D. Munz, A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes, Journal of Computational Physics 224 (2) (2007) 1049–1063.
[28] H. Liu, K. Xu, A Runge–Kutta discontinuous Galerkin method for viscous flow equations, Journal of Computational Physics 224 (2) (2007) 1223–1242.
[29] H. Luo, L. Luo, 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 1 (3) (2009) 301–318.
[30] B. van Leer, S. Nomura, Discontinuous Galerkin Method for Diffusion, AIAA Paper 2005-5108, 2005.
[31] B. van Leer, M. Lo, A Discontinuous Galerkin Method for Diffusion Based on Recovery, AIAA Paper 2007-4083, 2007.
[32] M. Raalte, B. van Leer, Bilinear forms for the recovery-based discontinuous Galerkin method for diffusion, Communication of Computational Physics 5 (2-4) (2009) 683–693.
[33] R. Nourgaliev, H. Park, V. Mousseau, Recovery discontinuous Galerkin Jacobian-free Newton–Krylov method for multiphysics problems, Computational Fluid Dynamics Review, in press.
[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, V. Mousseau, A Reconstructed Discontinuous Galerkin Method for the Compressible Euler Equations on Arbitrary Grids, AIAA-2009-3788, 2009.
[36] Y.H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of Computational Physics 192 (2003) 593–623.