Computers & Fluids 53 (2012) 133–144

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

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

An implicit discontinuous Galerkin method for the unsteady compressible Navier–Stokes equations

Hong Luo [a,*], Hidehiro Segawa [a], Miguel R. Visbal [b]

[a] Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, USA
[b] Computational Sciences Branch, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA

ARTICLE INFO

Article history:
Received 1 June 2010
Received in revised form 17 May 2011
Accepted 7 October 2011
Available online 18 October 2011

Keywords:
Discontinuous Galerkin methods
Implicit methods
Compressible Navier–Stokes equations

ABSTRACT

A high-order implicit discontinuous Galerkin method is developed for the time-accurate solutions to the compressible Navier–Stokes equations. The spatial discretization is carried out using a high order discontinuous Galerkin method, where polynomial solutions are represented using a Taylor basis. A second order implicit method is applied for temporal discretization to the resulting ordinary differential equations. The resulting non-linear system of equations is solved at each time step using a pseudo-time marching approach. A newly developed fast, *p*-multigrid is then used to obtain the steady state solution to the pseudo-time system. The developed method is applied to compute a variety of unsteady subsonic viscous flow problems. The numerical results obtained indicate that the use of this implicit method leads to significant improvements in performance over its explicit counterpart, while without significant increase in memory requirements.

1. Introduction

While DG was originally introduced by Reed and Hill [1] for solving the neutron transport equation back in 1973, major interest did not focus on it until the nineties [2–5]. Nowadays, it is widely used in the computational fluid dynamics, computational aeroacoustics, and computational electromagnetics, to name just a few [6–21]. What is known so far about this method offers a tantalizing glimpse of its full potential. Indeed, what sets this method apart from the crowd is many attractive features it possesses: (1) It has 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 method is well suited for complex geometries since it can be applied on unstructured grids. In addition, the method can also handle non-conforming elements, where the grids are allowed to have hanging nodes. (4) The method is highly parallelizable, as it is 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 method. (5) It 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 method allows easy implementation of *hp*-refinement, for example, the order of accuracy, or shape, can vary from element to element. (6) It has the ability to compute low Mach number flow problems without any special treatment.

In recent years, significant progress has been made in developing numerical algorithms for solving the compressible Euler and Navier–Stokes equations using discontinuous Galerkin methods. Most of these numerical methods are based on the semi-discrete approach: discontinuous Galerkin finite element methods are used for the spatial discretization, rendering the original partial differential equations (PDE) into a system of ordinary differential equations (ODE) in time. After constructing the ODE system, a time-stepping strategy is used to advance the solution in time. Usually, explicit temporal discretizations such as multi-stage Runge–Kutta schemes are used to integrate the semi-discrete system in time. In general, explicit schemes and their boundary conditions are easy to implement, vectorize and parallelize, and require only limited memory storage. However, for large-scale problems and especially for the higher-order DG solutions, the rate of convergence slows down dramatically, resulting in inefficient solution techniques. In order to speed up convergence, a multi-grid strategy or an implicit temporal discretization is required.

On the other hand, numerical algorithms for computing unsteady flows have lagged behind in the context of the discontinuous Galerkin methods. Explicit methods, deemed to be too slow for obtaining steady solutions, may be the only choice for certain unsteady applications such as shock wave and transition simulations, when the time scales of interest are small, or more precisely, when they are comparable to the spatial scales. However, when dealing with many

* Corresponding author.
E-mail address: hong_luo@ncsu.edu (H. Luo).

0045-7930/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2011.10.009

low reduced frequency phenomena with disparate temporal and spatial scales, explicit methods are notoriously time-consuming, since the allowable time step is much more restrictive than that needed for an acceptable level of time accuracy. Therefore, it is desirable to develop a fully implicit method, where the time step is solely determined by the temporal accuracy consideration for the flow physics and is not limited by the numerical stability consideration.

When an implicit scheme is used to compute unsteady flows, one has to drive the unsteady residual to zero (or at least to truncation error) at each time step. In the context of factored implicit schemes or linearized implicit schemes, this is usually done by employing inner iterations. It is the role of these inner iterations to eliminate errors, if any, due to factorization and linearization, and sometimes also errors arising from employing a lower order approximation on the implicit side. Alternatively, a pseudo-time marching can be used to drive the unsteady residual to zero. In this context, multigrid techniques or implicit methods are typically used to accelerate this pseudo-time stepping evolution to reduce the computational cost of each implicit time-step.

The objective of the effort discussed in this paper is to develop a high-order accurate and fast implicit discontinuous Galerkin method for the time accurate solutions of the compressible Navier–Stokes equations on arbitrary grids. The spatial discretization is carried out using a discontinuous Galerkin method, where the polynomial solutions are constructed using a Taylor basis [13–15]. A non-linear system of equations arising from a fully implicit temporal discretization of the unsteady Euler and Navier–Stokes equations is solved at each time step using a pseudo-time marching approach. A newly developed fast, p-multigrid [17,18] is then used to obtain the steady state solution to the pseudo-time system. The developed method is applied to compute a variety of unsteady subsonic viscous flow problems. The numerical results obtained indicate that the use of the present implicit method leads to significant improvements in performance over its explicit counterpart, while without significant increase in memory requirements. The remainder of this paper is structured as follows. The governing equations are described in Section 2. The implicit 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 Reynolds-averaged 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 $\mathbf{U}$, inviscid flux vector $\mathbf{F}$, and viscous flux vector $\mathbf{G}$, 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

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

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

$$T=\frac{p}{\rho R} \tag{2.6}$$

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

3. Numerical method

The governing Eq. (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,

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

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. 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

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

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 such as

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

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,

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

where N is the dimension of the polynomial space. 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.

Since the numerical solution $\mathbf{U}_h$ is discontinuous between element interfaces, the interface fluxes are not uniquely defined. 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. In this respect, discontinuous Galerkin formulations are very similar to finite volume methods, 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. Many upwind schemes have been implemented for computing the inviscid fluxes, although HLLC scheme is exclusively used for the approximate solution of the Riemann problem at the interfaces in this work.

DG methods are indeed a natural choice for the discretization of the inviscid fluxes. However, the DG formulation is far less certain and advantageous for the discretization of the viscous and heat 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 developed in the literature, such as those by Bassi and Rebay [19,20], Cockburn and Shu [21], Baumann and Oden [22], Luo et al. [23,24], and many others. Although many schemes have been implemented for computing the viscous and heat fluxes, the second Bassi–Rebay scheme (referred as BR2 [20] in the literature) is exclusively chosen for the discretization of the viscous fluxes in the present work.

In the traditional DG methods either standard Lagrange or hierarchical node-based finite element basis functions are used to represent numerical polynomial solutions in each element. As a result, the unknowns to be solved are the variables at the nodes and the polynomial solutions are dependent on the shape of elements. For example, for a linear polynomial approximation in 2D, a linear polynomial approximation is used for triangular elements and the unknowns to be solved are the variables at the three vertices and a bi-linear polynomial approximation is used for quadrilateral elements and the unknowns to be solved are the variables at the four vertices. In the present work, the numerical polynomial solutions are represented using a Taylor series expansion at the cell centroid [14], which can be further expressed as a combination of cell-averaged values and their derivatives at the cell centroid. 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 a result, this formulation is able to provide a unified framework, where both cell-centered and vertex-centered finite volume schemes can be viewed as special cases of this discontinuous Galerkin method by choosing reconstruction schemes to compute the derivatives, offer the insight why the DG methods are a better approach than the finite volume methods based on either TVD/MUSCL reconstruction or ENO/WENO reconstruction, and possesses a number of distinct, desirable, and attractive features and advantages. First, the same numerical polynomial solutions are used for any shapes of elements, which can be triangle, quadrilateral, and polygon in 2D, and tetrahedron, pyramid, prism, and hexahedron in 3D. Using this formulation, DG method can be easily implemented on arbitrary meshes. The numerical method based on this formulation has the ability to compute 1D, 2D, and 3D problems using the very same code, which greatly alleviates the need and pain for code maintenance and upgrade. Secondly, cell-averaged variables and their derivatives are handily available in this formulation. This makes implementation of WENO limiter straightforward and efficient that is required to eliminate non-physical oscillations in the vicinity of discontinuities. Thirdly, the basis functions are hierarchic. This greatly facilitates implementation of p-multigrid methods and p-refinement. Last, cell-averaged variable equations are decoupled from their derivatives equations in this formulation. This makes development of fast, low-storage implicit methods possible. The details of this DG formulation can be found in Ref. [14].

The discontinuous Galerkin finite element approximation (3.1) leads to the following semi-discrete system of non-linear equations:

$$M\frac{d\mathbf{U}}{dt} + R(\mathbf{U}) = 0 \tag{3.6}$$

where M denotes the mass matrix, $\mathbf{U}$ is the global vector of the degrees of freedom, and $\mathbf{R}(\mathbf{U})$ is the residual vector. An explicit time-accurate advance of Eq. (3.6) in time can be accomplished using a multi-stage TVD Runge–Kutta scheme [2,3]. However, when the maximum allowable time step imposed by an explicit stability requirement is much smaller than that imposed by the acceptable level of time accuracy, implicit schemes are to be preferred. The implicit time integration used in the present work is based on a two-parameter family scheme. Eq. (3.6) is integrated in time as follows

$$\frac{M}{\Delta t}\left[(1+\varphi)\mathbf{U}^{n+1} - \frac{\mathbf{U}^n}{1-\varphi} + \varphi\mathbf{U}^{n-1}\right] + \vartheta\mathbf{R}(\mathbf{U}^{n+1}) + (1-\vartheta)\mathbf{R}(\mathbf{U}^n) = 0 \tag{3.7}$$

Here n denotes the time level. If $\varphi = 0$, and $\vartheta = 1$, the scheme is the first-order backward Euler method. If $\varphi = 1/2$, and $\vartheta = 1$, the popular three-point backward differencing method is obtained, which is a second-order accurate discretization in time. If $\varphi = 0$, and $\vartheta = 1/2$, the resulting scheme known as Crank–Nicholson method is second-order accurate in time. Eq. (3.7) represents a non-linear system of coupled equations, that has to be solved at each time step. It can be solved by treating it as a steady state problem by introducing a pseudo-time variable t_*

$$M(1+\phi)\frac{d\mathbf{U}}{dt_*} + \mathbf{R}_*(\mathbf{U}) = 0 \tag{3.8}$$

and 'time-marching' the solution using local pseudo-timesteps Δt_*, until $\mathbf{U}$ converges to $\mathbf{U}^{n+1}$. Here $\mathbf{U}$ is the approximation to $\mathbf{U}^{n+1}$ and the unsteady residual $\mathbf{R}_*(\mathbf{U})$ is defined as

$$\mathbf{R}_*(\mathbf{U}) = \frac{M}{\Delta t}(1+\varphi)\mathbf{U} + \vartheta\mathbf{R}(\mathbf{U}) + S(\mathbf{U}^n, \mathbf{U}^{n-1}) \tag{3.9}$$

with the source term

$$S(\mathbf{U}^n, \mathbf{U}^{n-1}) = \frac{M}{\Delta t}\left[-\frac{\mathbf{U}^n}{1-\varphi} + \varphi\mathbf{U}^{n-1}\right] + (1-\vartheta)\mathbf{R}(\mathbf{U}^n) \tag{3.10}$$

which remains fixed during the pseudo-time marching procedure. In the present work, a p-multigrid method [17,18] is used to accelerate this pseudo-time stepping evolution to reduce the computational cost of each implicit time step. The main contribution of this work is to show that this p-multigrid method, which has proven to be computationally efficient and robust for steady flows around complex geometries, can be extended and used to accelerate convergence to steady state solutions of this pseudo-time unsteady problem.

Nowadays, geometric multigrid methods (termed as *h*-multigrid from now on) are routinely used to accelerate the convergence of the Euler and Navier–Stokes equations to a steady state on unstructured grids. It is well established that *h*-multigrid acceleration can drastically reduce the computational costs. In standard *h*-multigrid methods, solutions on spatially coarse grids are used to correct solutions on the fine grid. *p*-multigrid method is a natural extension of *h*-multigrid methods to high-order finite element formulation, such as spectral-*hp* or discontinuous Galerkin methods, where systems of equations are solved by recursively iterating on solution approximations of different polynomial order. For example, to solve equations derived using a polynomial approximation order of 3, the solution can be iterated on at an approximation order of $p = 2$, 1, and 0. The basic idea of a *p*-multigrid method is to perform time steps on the lower order approximation levels to calculate corrections to a solution on a higher order approximation level. For example, the *p*-multigrid method for DG (P2) method consists of the following steps at each *p*-multigrid cycle:

(1) Perform a time-step at the highest approximation order P2, which yields the initial solution $\mathbf{U}_{P2}^{n+1}$.
(2) Restrict the solution and residual vectors from P2 to one lower level approximation P1:

$$\mathbf{U}_{P1} = \mathbf{I}_{P2}^{P1}\mathbf{U}_{P2}^{n+1}, \quad \mathbf{R}_{P1} = \widetilde{I}_{P2}^{P1}\mathbf{R}(\mathbf{U}_{P2}^{n+1}).$$

(3) Compute the force terms on the lower approximation level P1,

$$\mathbf{F}_{P1} = \mathbf{R}_{P1} - \mathbf{R}(\mathbf{U}_{P1}).$$

(4) Perform a time-step at the lower approximation level P1 where the residual is given by

$$\mathbf{R} = \mathbf{R}(\mathbf{U}_{P1}) + \mathbf{F}_{P1},$$

which yields the solution at the lower level $\mathbf{U}_{P1}^{n+1}$.
(5) Restrict the solution and residual vectors from P1 to one lower level approximation P0,

$$\mathbf{U}_{P0} = \mathbf{I}_{P1}^{P0}\mathbf{U}_{P1}^{n+1}, \quad \mathbf{R}_{P0} = \widetilde{I}_{P1}^{P0}(\mathbf{R}(\mathbf{U}_{P1}^{n+1}) + \mathbf{F}_{P1}).$$

(6) Compute the force terms on the lower approximation level P0,

$$\mathbf{F}_{P0} = \mathbf{R}_{P0} - \mathbf{R}(\mathbf{U}_{P0}).$$

(7) Perform a time-step at the lower approximation level P0 where the residual is given by

$$\mathbf{R} = \mathbf{R}(\mathbf{U}_{P0}) + \mathbf{F}_{P0},$$

which yields the solution at the lower level $\mathbf{U}_{P0}^{n+1}$.
(8) Prolongate the correction $\mathbf{C}_{P0}$ from the lowest level P0 to update the higher level solution $\mathbf{U}_{P1}^{n+1}$

$$\mathbf{C}_{P0} = \mathbf{U}_{P0}^{n+1} - \mathbf{U}_{P0}, \quad \widetilde{\mathbf{U}}_{P1}^{n+1} = \mathbf{U}_{P1}^{n+1} + \mathbf{J}_{P0}^{P1}\mathbf{C}_{P0}.$$

(9) Prolongate the correction $\mathbf{C}_{P1}$ back from the level P1 to update the higher level $\mathbf{U}_{P2}^{n+1}$

$$\mathbf{C}_{P1} = \widetilde{\mathbf{U}}_{P1}^{n+1} - \mathbf{U}_{P1}, \quad \widetilde{\mathbf{U}}_{P2}^{n+1} = \mathbf{U}_{P2}^{n+1} + \mathbf{J}_{P1}^{P2}\mathbf{C}_{P1}.$$

The above single *p*-multigrid cycle will produce a more accurate solution at the higher level, starting from an initial solution at the same level, where $\mathbf{I}$ is the state restriction operator, $\mathbf{J}$ is the state prolongation operator, and $\widetilde{I}$ is the residual restriction operator and is not necessary the same as the state restriction operator. The definition of these operators can be introduced in a standard manner using the basis of the finite element approximation spaces. Specifically,

$$\mathbf{I}_p^q = (M^q)^{-1}M^{qp},$$
$$\mathbf{J}_q^p = (M^p)^{-1}M^{pq},$$

and

$$\widetilde{I}_p^q = M^{qp}(M^p)^{-1},$$

where

$$M_{i,j}^p = \int_{\Omega e} B_i^p B_j^p d\Omega,$$
$$M_{i,j}^{pq} = \int_{\Omega e} B_i^p B_j^q d\Omega.$$

Note that our *p*-multigrid method does not iterate at the various levels, i.e., the number of pre-post smoothing iterations is set one at all levels.

In general, the same time integration scheme is applied to advance the solution on all levels P2, P1, and P0. Implicit time integration schemes such as element Jacobian and element line Jacobian methods have been used as smoothers for *p*-multigrid for solving the compressible Navier–Stokes equations [12]. Unfortunately, they require prohibitively large memory and computing cost for Jacobian matrix, rendering them impractical, if not impossible, for large scale problems, and especially for high-order solutions. Furthermore, the implementation of slope limiters for DG methods in any implicit schemes is problematic and difficult, limiting them to only smooth flows without shocks or discontinuities. On the other hand, when the multi-stage TVD Runge–Kutta explicit scheme is used as an iterative smoother, the performance of the resulting *p*-multigrid method is quite disappointing [11], in contrast to the success that *h*-multigrid methods enjoyed to accelerate the convergence of the Euler and Navier–Stokes equations using the multi-stage Runge–Kutta explicit scheme as an iterative smoother. This is mainly due to the fact that explicit schemes are inefficient to reduce lower frequency errors on the lowest level, though they are fairly efficient at eliminating high frequency error modes in the solution (i.e., local error). By transferring the discrete equations to a coarse approximation level, once the high frequency error modes on the fine approximation level have been eliminated, the lower frequency modes from the fine approximation level now appear as higher frequency modes on the coarse approximation level in the *p*-multigrid scheme, and are effectively handled by the explicit scheme on this approximation level. This observation motivates us to use an explicit smoother on the higher approximation levels P1, and P2, and an implicit smoother on the coarsest level P0. Implicit smoothers have better convergence properties, and are far more effective in eliminating the lowest frequency errors, and yet the storage requirements and computational costs for Jacobian matrix on the coarsest level P0 are relatively small. Note that DG (P0) method, corresponding to zero order basis functions, degenerates to the classical first-order cell-centered finite volume scheme, so that the fairly mature implicit methods developed over the last decades can be readily used as the implicit iterative smoother.

Specifically, the *p*-multigrid method uses the following explicit three-stage third-order TVD Runge–Kutta scheme [2,3]

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t M^{-1}\mathbf{R}(\mathbf{U}^n),$$
$$\mathbf{U}^{(2)} = \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\left[\mathbf{U}^{(1)} + \Delta t M^{-1}\mathbf{R}(\mathbf{U}^{(1)})\right]$$
$$\mathbf{U}^{n+1} = \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\left[\mathbf{U}^{(2)} + \Delta t M^{-1}\mathbf{R}(\mathbf{U}^{(2)})\right]$$

as the iterative smoother on the higher level approximations ($p > 0$). This method is linearly stable for a Courant number less than or equal to $1/(2p+1)$. On the lowest level approximation ($p = 0$), the spatially discretized Navier–Stokes equations is integrated in time

using the backward Euler implicit method, which can be linearized and written as

$$\left(\frac{V}{\Delta t}\mathbf{I}-\frac{\partial \mathbf{R}_{P_0}}{\partial \mathbf{U}_{P_0}}\right)\Delta \mathbf{U}_{P_0}=\mathbf{R}_{P_0}$$

where V is the element volume. This linear system of equations are solved using a matrix-free implicit GMRES+LU-SGS method [25–27].

As a result, this p-multigrid method has two remarkable features: (1) Low memory requirements. The implicit smoothing is only used on the lowest level P0, where the storage requirement is not as demanding as on the higher level. (2) Natural extension to flows with discontinuities such as shock waves and contact discontinuities. A monotonic limiting procedure required to eliminate spurious oscillations of high-order approximations in the vicinity of discontinuities can be easily implemented as a post-processing filter (smoothing) in an explicit method, but not in an implicit method. The numerical experiments for a variety of steady-state applications [17,18]strongly indicate the order independent property of this p-multigrid method and demonstrate that this method is orders of magnitude faster than its explicit counterpart.

4. Examples

A few examples are presented here to illustrate the high accuracy and efficiency of this implicit DG method for a variety of unsteady flow problems. All the computations are performed on a Dell XPS M1210 laptop computer (Intel 2.33 GHz T7600 CPU with 4 GB memory) using a Suse 11.0 Linux operating system. For all the unsteady flow problems, the accuracy level of 0.01, i.e., two orders of magnitude drop in residual, is used to drive the unsteady residual at each time step for the pseudo-time system. The relative L_2 norm of the density residual is taken as a criterion to test convergence history. A CFL number of $1/(2p+1)$ is used for the explicit Runge–Kutta method and a CFL number of 100 is used for the implicit GMRES+LU-SGS method. The solution tolerance for GMRES is set to 0.1 with 10 search directions and 20 iterations.

4.1. A stationary isentropic vortex

The convection of an inviscid isentropic vortex is a well-known test case, which is widely used in the literature to assess the accuracy of the numerical methods. The exact solution for this case at any time t is the initial solution translated over a distance $u_\infty t$ for a horizontally convecting vortex, providing a valuable reference for measuring the accuracy of the numerical solution. The stationary isentropic vortex test case is considered here. The problem setup is taken from Ref. [28]. For this test case, a grid convergence study is conducted to numerically assess the order of accuracy of the implicit DG method. Fig. 1 shows three successively refined grids used in the study, having 834, 3336, 13,344 elements respectively. Numerical solutions to this problem at $t = 10$ are computed using DG (P1) method on these three grids to obtain quantitative

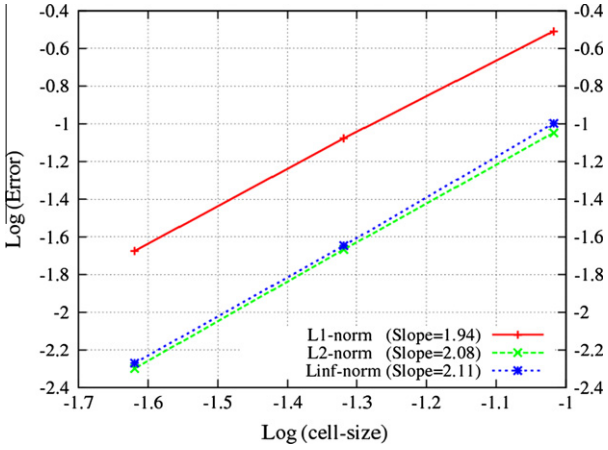

Fig. 2. Results of grid-refinement study for the stationary vortex.

measurement of the order of accuracy and discretization errors, as shown in Fig. 2. As expected, a formal second order of accuracy is obtained for all L_1, L_2 and L_∞ norm, implicitely demonstrating the second order of accuracy of the implicit DG method in time. Fig. 3 shows the computed density contours in the flow field at $t = 0$ and $t = 100$ using the fine grid, respectively, where the second order DG (P1) solution exhibits a very good shape retaining property for the vortex. Fig. 4 displays the comparison of the density profiles along the horizontal centerline at $t = 0$, 50, and 100, where almost no visable deviation can be observed between the computed and exact solutions, and the vortex core is well conserved, clearly demonstrating the high accuracy of the DG method. As a comparison, the results for the same test case from Ref. [28] are presented in Fig. 5, where one can see that all of the finite difference methods presented there, regardless of the order of the schemes are more diffusive, especially around the vortex core than our second order DG (P1) method.

4.2. Shedding flow past a triangular wedge

This test case taken from reference 29 is used to simply illustrate the importance of the temporal discretization on the accuracy of the numerical solutions. An inviscid flow over a triangular wedge placed on the centerline of the domain is considered in this test case. The numerical solution is presented at a Mach number of 0.2. The mesh used in the computation is shown in Fig. 6, which contains 12,307 triangular elements, 6248 grid points, and 189 boundary faces. The grid has a size of 0.0625 in the vicinity of the triangular wedge, and 0.08 in the wake region. Figs. 7 and 8 show the computed density contours in the flow fields at $t = 100$ obtained using the first and second order temporal discretization, respectively, where the second order DG method is used for the spatial discretization. A fixed time-step size of $\Delta t = 0.05$, corresponding to a maximal CFL number of about 500, is used for both

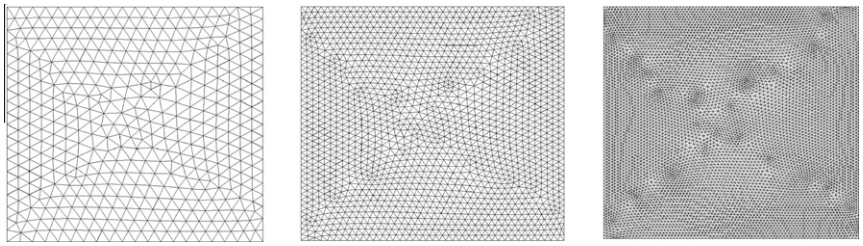

Fig. 1. Sequences of three successively globally refined meshes used for computing the stationary vortex test case.

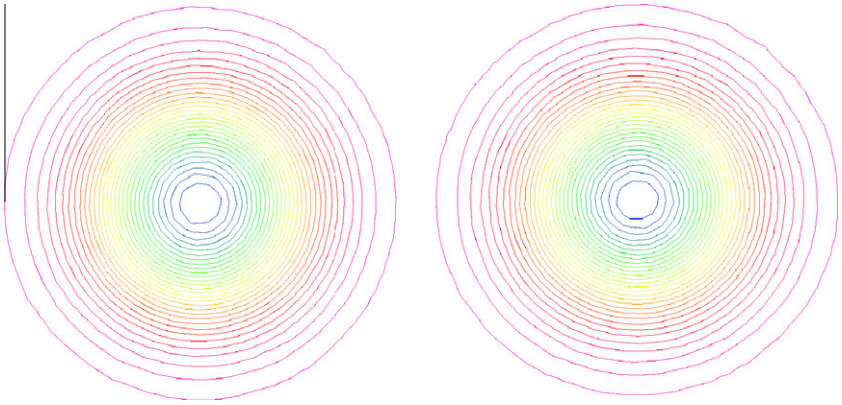

Fig. 3. Density contours for the stationary vortex flow at $t = 0$ (left) and $t = 100$ (right), respectively.

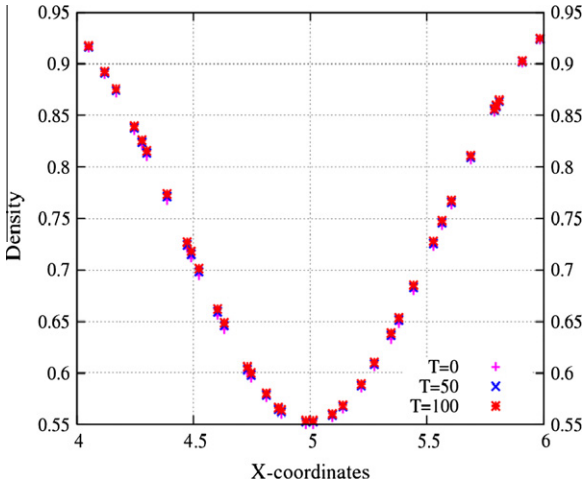

Fig. 4. Comparison of density profiles at different times.

computations. Unlike Ref. [29] where the first order space DG (P0) solution is used as the initial condition for the higher-order DG solutions, all of our computations are carried out using a uniform flow as the initial condition to demonstrate the robustness of our DG method. As the flow passes the wedge, the flow separates after some time due to the artificial viscosity and vortices are originated around the two sharp corners and then convected downstream with shedding. What we are interested in here is the ability of the implicit time-stepping schemes for retaining the shape of the vortices as they are convected downstream of the body, which provides a good measure for the accuracy of the computed solutions. One can clearly see that the vortex is significantly diffused using the first order implicit backwards differencing scheme (BDF1), as the error deduced by the first order time integration method becomes overwhelmingly dominant due to the use of the large time step. As expected, the second-order implicit Backwards Differencing Scheme (BDF2) time discontinuous Galerkin method DG (P1) displays a substantially better shape retaining property for the vortex, clearly indicating the superior accuracy of a higher-order time

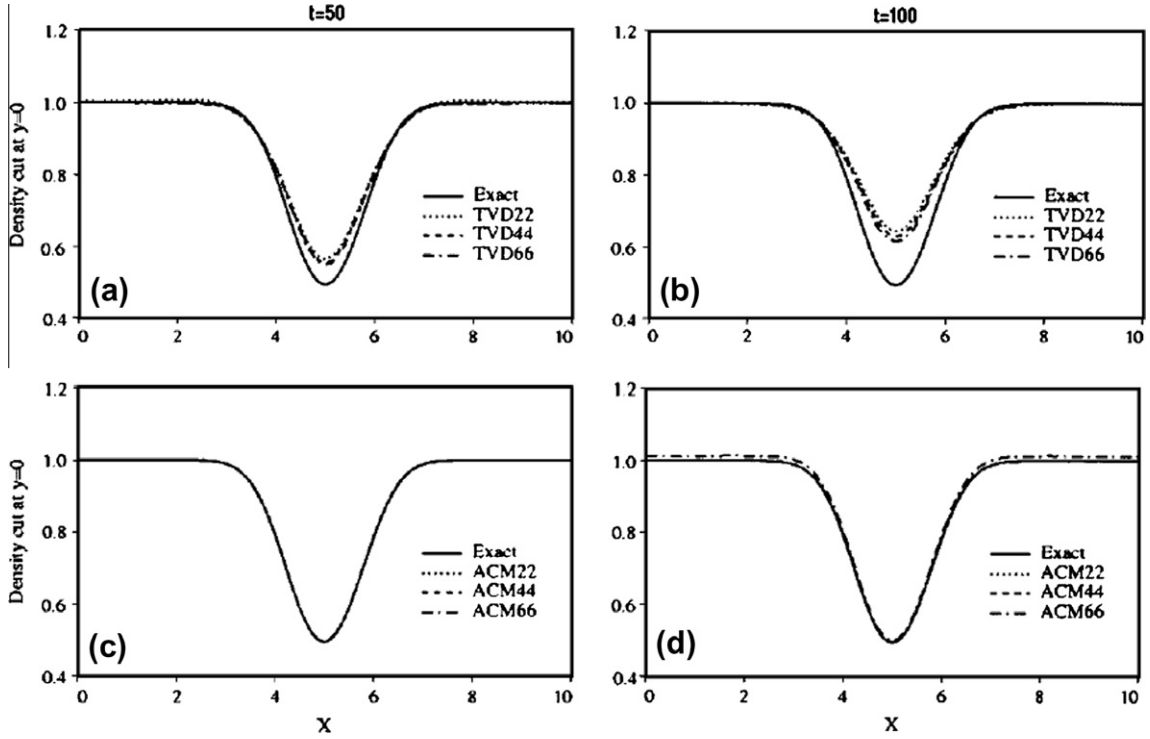

Fig. 5. Computational results from reference 28.

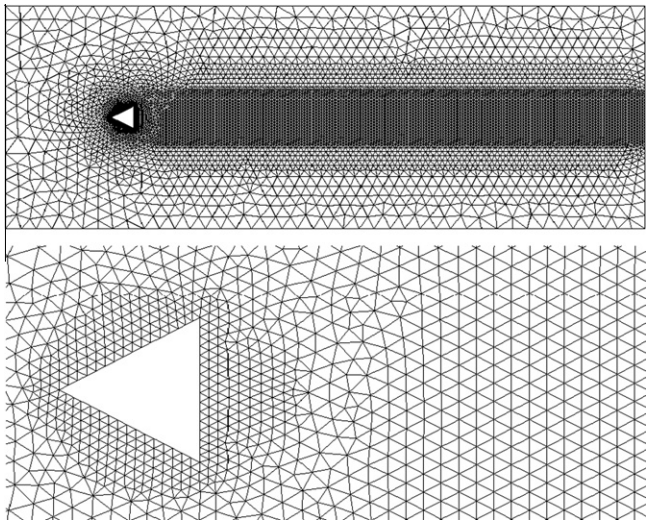

Fig. 6. Mesh used for computing an inviscid flow past a wedge (nelem = 12,307, npoin = 6248, nboun = 189).

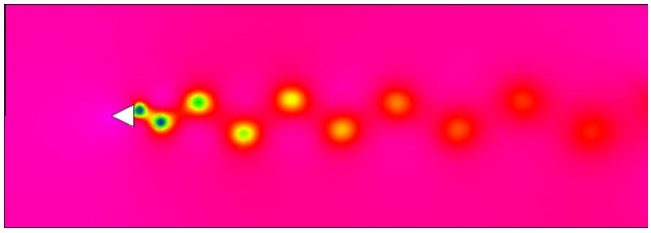

Fig. 7. Computed density contours obtained using BDF1 and DG (P1) at $t = 100$ for flow past a wedge at a Mach number of 0.2.

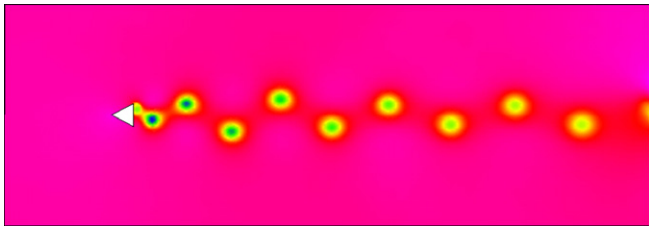

Fig. 8. Computed density contours obtained using BDF2 and DG (P1) at $t = 100$ for flow past a wedge at a Mach number of 0.2.

discontinuous Galerkin method and necessity of using higher order temporal discretization schemes for the accurate and efficient solutions of unsteady flow problems. Fig. 9 shows the computed density contours in the flow field obtained using the third order DG (P2) spatial discretization, where the high accuracy of the numerical solution is evident, due to both spatial and temporal high order discretizations.

To illustrate the high efficiency of the present implicit method, the computational efforts for both implicit and explicit DG (P2) solutions are listed in Table 1, which indicates that the implicit method requires about five times less CPU time than its explicit counterpart in this case. This is mainly due the fact that the explicit three-stage Runge–Kutta method has to use a much smaller time step of 0.00025, which is determined by the stability consideration rather than the accuracy consideration. Although the implicit method requires relatively fewer time steps in comparison with its explicit counterpart, the CPU time required by the explicit method is much lower than the one required by the implicit method. At each

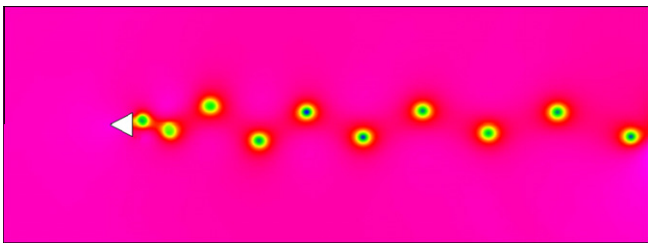

Fig. 9. Computed density contours obtained using BDF2 and DG (P2) at $t = 100$ for flow past a wedge at a Mach number of 0.2.

Table 1
Comparison of the CPU time between explicit and implicit methods for the shedding vortex flow.

For solution at $t = 100$	Time-step size	Time steps	CPU time (s)
Implicit (BDF2)	$\Delta t = 0.05$	2000	7323
Explicit (3stage RK)	$\Delta t = 0.00025$	400,000	35,600

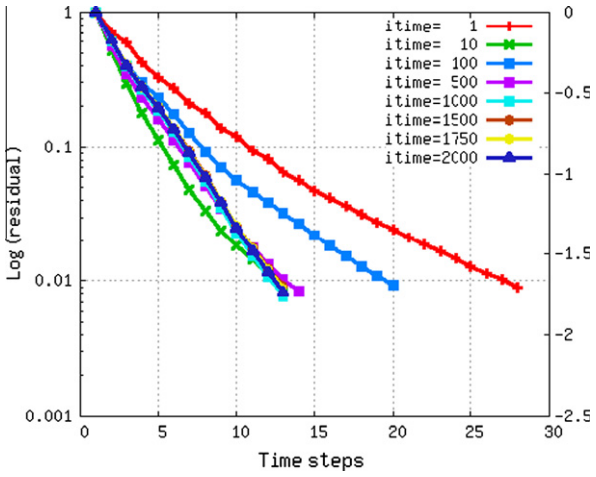

Fig. 10. Convergence history of the pseudo-time system at different time-steps using the *p*-multigrid method.

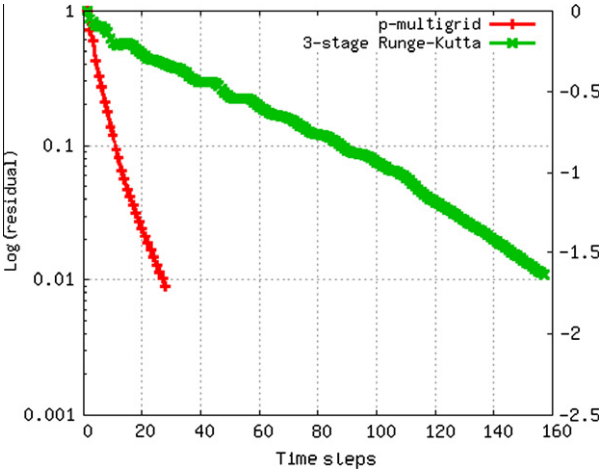

Fig. 11. Convergence history of the pseudo-time system at the first time-step using 3-stage Runge–Kutta method and the *p*-multigrid method.

time step, the explicit three-stage Runge–Kutta method only requires three residual evaluations, while the implicit method

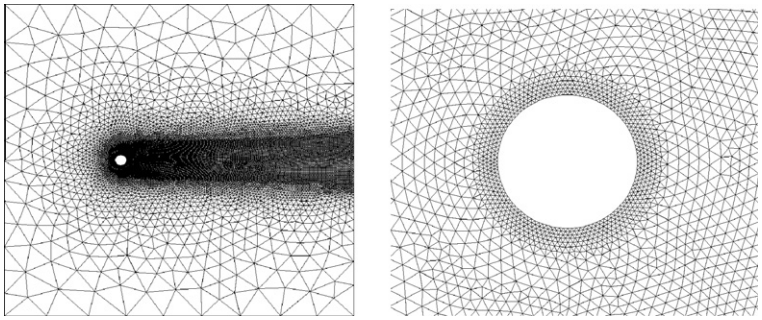

Fig. 12. Mesh used for computing a viscous flow past a cylinder (nelem = 21,809, npoin = 11,004, nboun = 199).

requires the solution of a large non-linear system of equations. Needless to say, an efficient solver for this task is crucial for the success of the implicit method. Fig. 10 shows the convergence history of the pseudo-system at different time steps. Typically, more time steps are required to obtain a steady-state solution to the pseudo-system at earlier time steps, and fewer iterations are needed to meet the stop criterion at latter time-steps. Fig. 11 compares the convergence histories of the explicit three-stage Runge–Kutta and the *p*-multigrid methods for the steady state solution to the pseudo-time system at the first time step. The 3-stage TVD Runge–Kutta method needs 158 time-steps to drive the unsteady residual down 2 orders of magnitude, while the use of the *p*-multigrid method can significantly reduce the time-steps required to meet the same stop criterion, clearly demonstrating that the *p*-multigrid method can be used to accelerate convergence to steady-state solutions of the pseudo-time unsteady problems.

4.3. Von-Karman vortex street

The von-Karman vortex street is probably one of the most extensively studied cases both experimentally and numerically in fluid dynamics. The initial condition is a uniform free stream with non-slip boundary conditions on the solid wall. The numerical solution is presented at a Mach number of 0.1. 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. 12. Three computations are preformed using DG (P1) and BF2 method at a Reynolds number of 100, 200, and 400 based on the diameter of the cylinder using a fixed time-step size of $\Delta t = 0.05$ respectively. The computed entropy contours in the flow fields at $t = 100$ for these three cases are shown in Fig. 13. The computed lift and drag coefficients are shown in Figs. 14 and 15, respectively, where Strouhal number for these three cases is given as well. These results compare well with experimental measurements and other numerical results in the literature. Time variations of the computed pressure lift and drag coefficients are presented in Fig. 16, while those of the computed viscous lift and drag coefficients are displayed in Fig. 17.

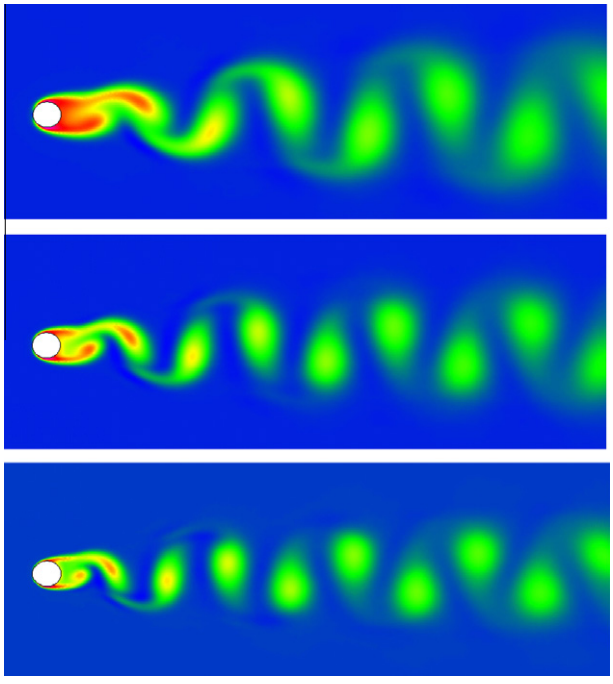

Fig. 13. Computed entropy contours at $t = 100$ for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100 (top), 200 (middle), and 400 (bottom), respectively.

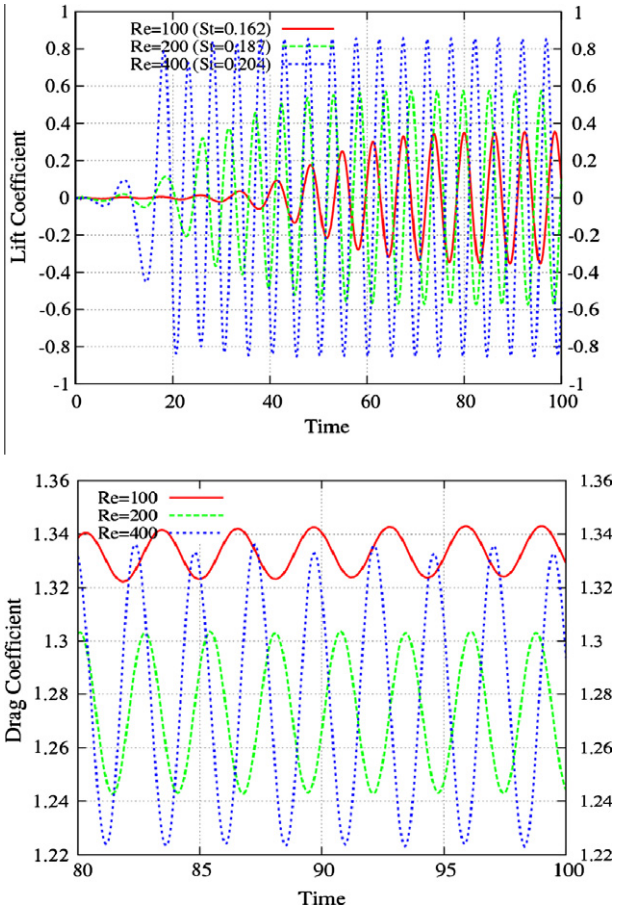

Fig. 14. Time history of the computed lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

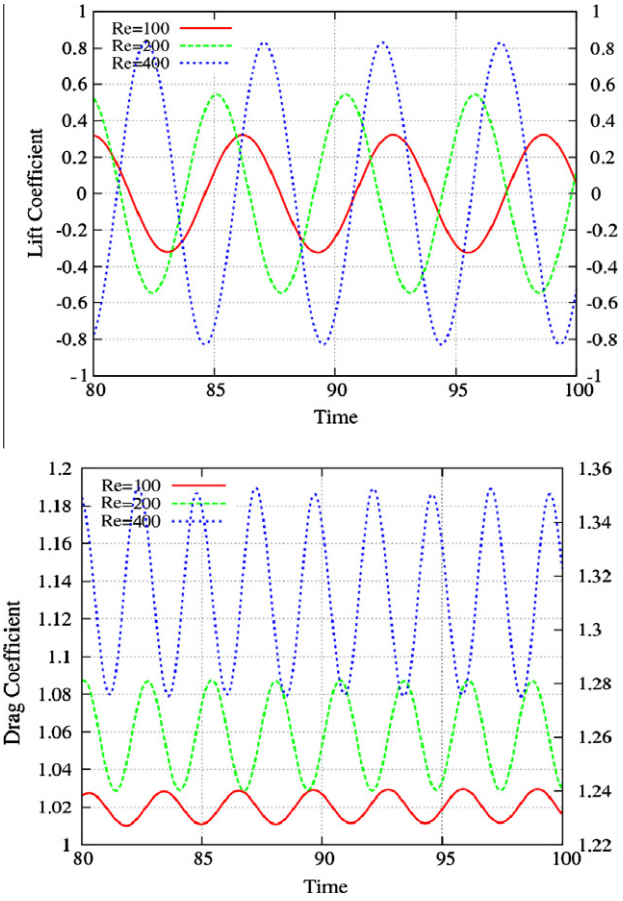

Fig. 15. Time evolution of the computed pressure lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

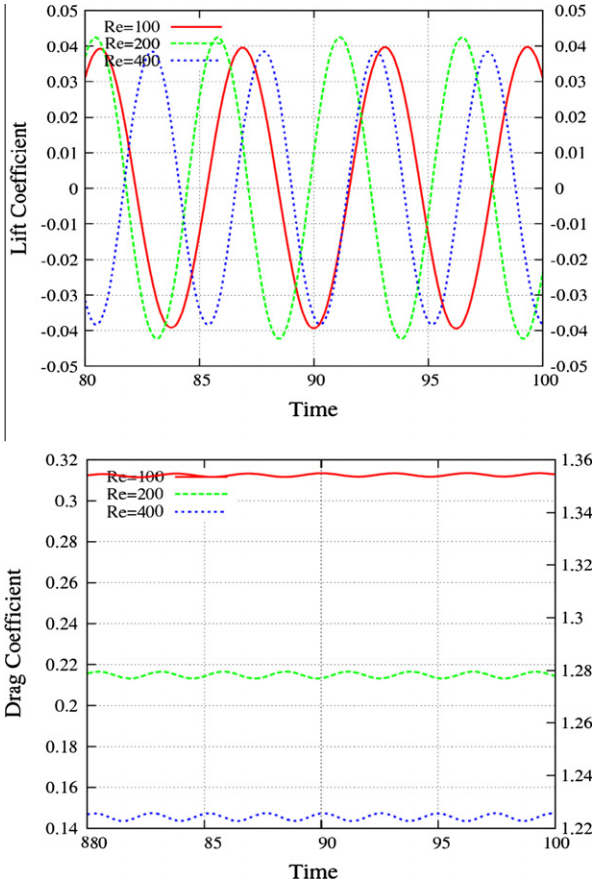

Fig. 16. Time evolution of the computed viscous lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

As an illustrative example to demonstrate the effectiveness of the current implicit method, Table 2 compares the characteristics of the explicit and implicit methods to this problem at a Reynolds number of 400. The explicit three-stage Runge–Kutta method has to use a much smaller time step of 0.00033 due to the CFL condition. As CPU time depends primarily on the number of time steps required in the computation, the implicit method offers almost one-order-of-magnitude improvement over its explicit counterpart for this test case, clearly demonstrating the efficiency of the present implicit method. Although the implicit method BF2 entails the solution of a non-linear problem at each time step, the present *p*-multigrid method is very efficient and effective to solve the time-implicit problems via the pseudo-time approach. Fig. 17 shows the convergence history of the pseudo-system at different time steps. Due to the use of the *p*-multigrid method, the number of time steps to reach a steady-state solution to the pseudo-time system at each time step is typically less than ten for the unsteady residual to drop two orders of magnitude. To assess the efficiency of the overall *p*-multigrid method for time-implicit problems, Fig. 18 compares the convergence histories of the explicit three-stage Runge–Kutta and the *p*-multigrid methods for the steady state solution to the pseudo-time system at the first time step. The 3-stage TVD Runge–Kutta method needs 197 time-steps to drive the unsteady residual down 2 orders of magnitude for the pseudo-time system, while the use of the *p*-multigrid method can significantly reduce the time-steps required to meet the same stop criterion, indicating the superior performance of the *p*-multigrid method.

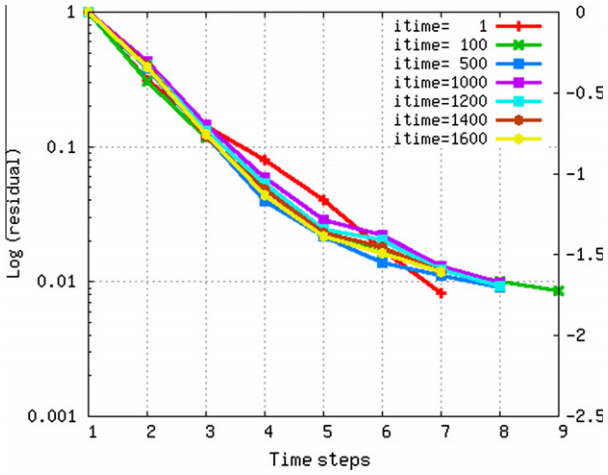

Fig. 17. Convergence history of the pseudo-time system at different time-steps using the *p*-multigrid method for the Karman vortex street problem at a Reynolds number of 400.

Table 2
Comparison of the CPU time between explicit and implicit methods for Karman-Vortex street problem at a Reynolds number of 400.

For solution at $t = 80$	Time-step size	Time steps	CPU time (s)
Implicit (BDF2)	$\Delta t = 0.05$	1600	5669
Explicit (3stage RK)	$\Delta t = 0.000333$	240,240	51,446

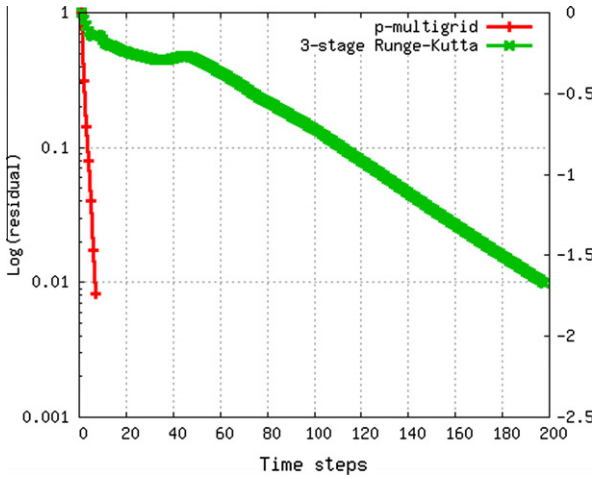

Fig. 18. Convergence history of the pseudo-time system at the first time-step using 3-stage Runge–Kutta method and the *p*-multigrid method for the Karman vortex street problem at a Reynolds number of 400.

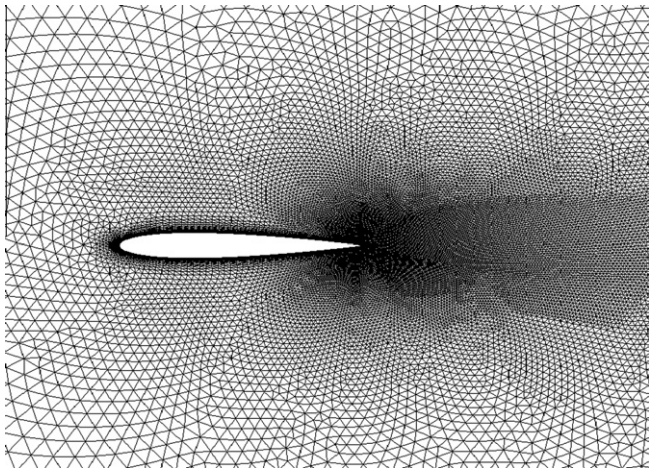

Fig. 19. Mesh used for computing an inviscid flow past a wedge (nelem = 60,126, npoin = 30,199, and nboun = 272).

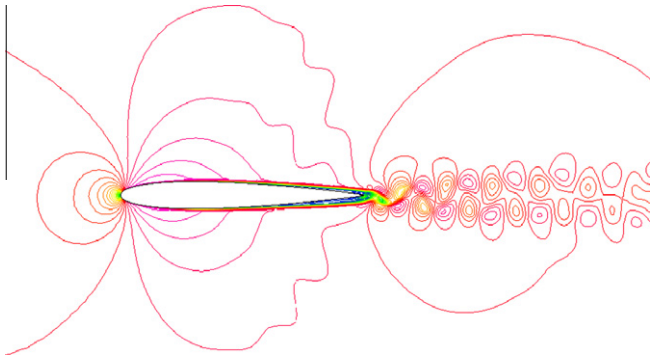

Fig. 20. Computed velocity contours in the flow field for flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0, and a Reynolds number of 10,000.

4.4. Viscous flow past a NACA0012 airfoil

A viscous flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0°, and a Reynolds number of 10,000 is considered in this case. The computation is initialized with constant free-stream values in the entire domain with non-slip boundary conditions on the solid wall. This computation is performed using DG (P2) and BDF2. The mesh used in the computation consists of

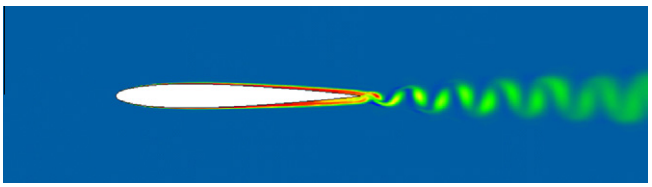

Fig. 21. Computed entropy contours in the flow field for flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0, and a Reynolds number of 10,000.

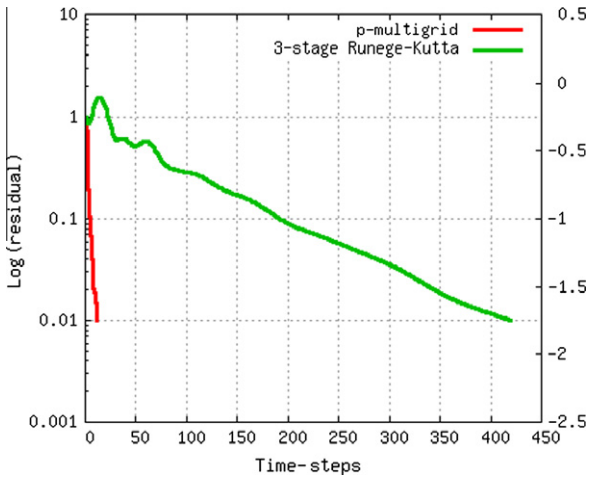

Fig. 22. Convergence history of the pseudo-time system at the first time-step using 3-stage Runge–Kutta method and the *p*-multigrid method for flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0, and a Reynolds number of 10,000.

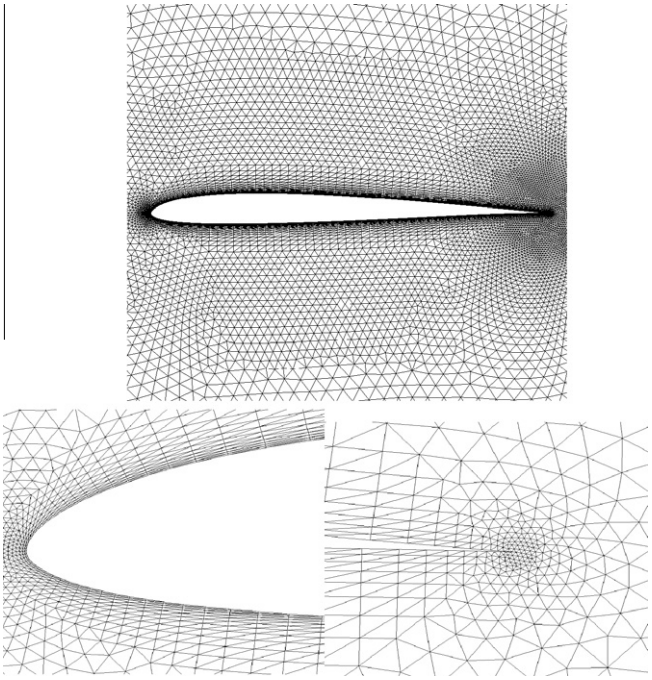

Fig. 23. Grid used for computing the unsteady viscous flow past a SD7003 airfoil (nelem = 50,781, npoin = 25,530, nboun = 279).

60,126 triangular elements, 30,199 grid points, and 272 boundary faces, as shown in Fig. 19, where the highly stretching elements are used close to the airfoil. The computed velocity and entropy contours are shown in Figs. 20 and 21, respectively, where tail vortices and back flows are clearly visible. To assess the performance of the overall *p*-multigrid method on highly stretching grids, the convergence histories for the steady state solution to the pseudo-time system at the first time step between the explicit three-stage Runge–Kutta and the *p*-multigrid methods are compared in Fig. 22. In this case, the 3-stage TVD Runge–Kutta and the *p*-multigrid methods require 420 and 12 time steps to drive the unsteady residual down 2 orders of magnitude for the pseudo-time system, respectively. Although the performance of the *p*-multigrid method is degraded somewhat, the convergence provided by the *p*-multigrid method is even more impressive in comparison with its explicit counterpart on highly stretching grids, demonstrating the superior performance of *p*-multigrid method on highly stretching grids.

4.5. 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°, and a Reynolds number of 10,000 is considered in this case. The computation is initialized with constant free-stream values in the entire domain with non-slip boundary conditions on the solid wall. Fig. 23 shows the mesh used in the computation, which consists of 50,781 triangular elements, 25,530 grid points, and 279 boundary faces with 200 grid points on the surface of the airfoil. The computation is performed using BDF2 temporal and DG (P2) spatial discretizations and a fixed time-step size of $\Delta t = 0.001$, corresponding to a maximal CFL number of about 500. Typical computed pressure contours in the flow field are compared with those obtained using a compact scheme [30–32] in Fig. 24, while the comparison for the vorticity contours between the DG method and the compact method is shown in Fig. 25. Qualitatively, both solutions look very similar, capturing the same flow features: separation of the flow on the upper surface of the airfoil and shedding of the tailing vorticies, in spite of the fact that a round trailing edge is used in the compact difference solution and a sharp trailing edge is used in the DG solution. The computed pressure contours in the flow filed are presented along with the mesh in Fig. 26 to illustrate that accurate and smooth solutions are obtained using the DG (P2) method in

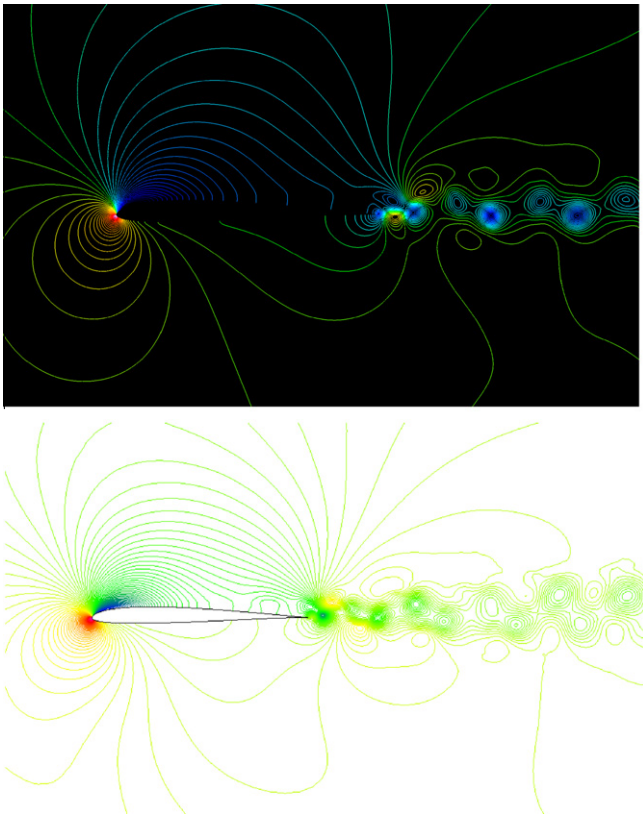

Fig. 24. Computed pressure contours in the flow field obtained by the compact method (upper) and DG (P2) method (lower) for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4°, and a Reynolds number of 10,000.

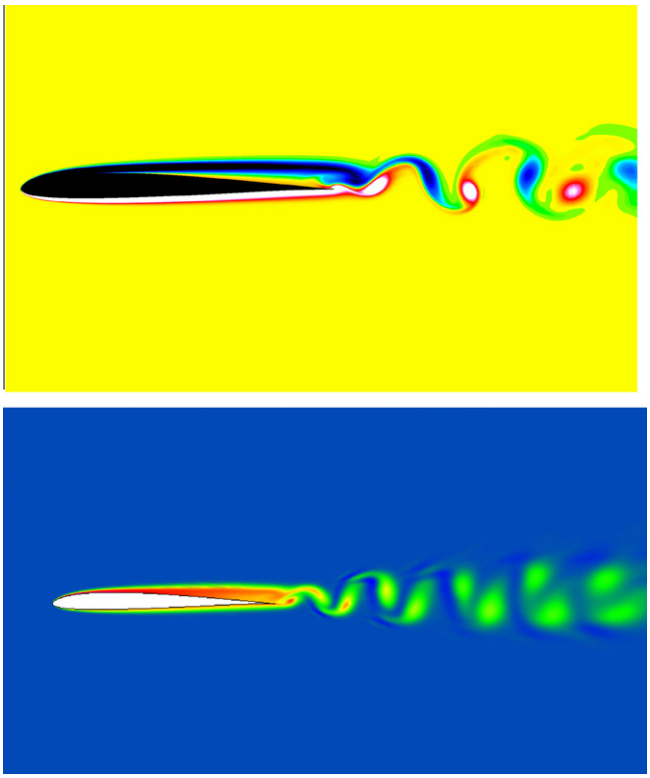

Fig. 25. Computed vorticity contours (upper) obtained by the compact method and computed entropy contours obtained by the DG (P2) method in the flow field for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4°, and a Reynolds number of 10,000.

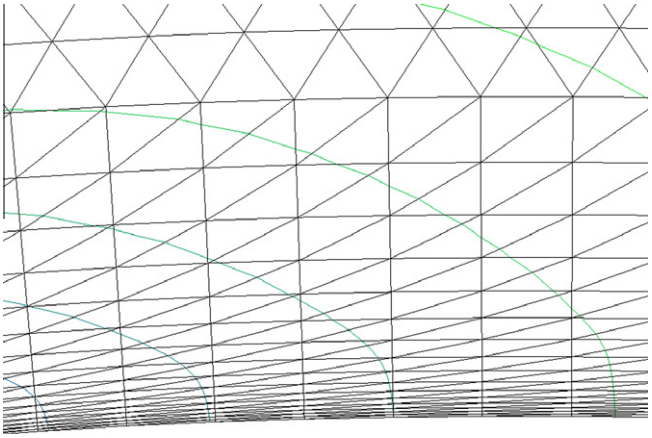

Fig. 26. Computed pressure contours on the upper surface of the airfoil for the viscous flow past a SD7003 airfoil.

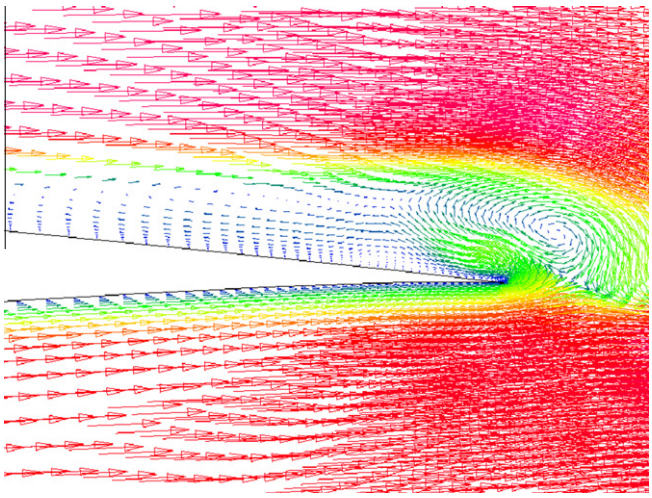

Fig. 27. Computed velocity vectors near the airfoil for the viscous flow past a SD7003 airfoil.

spite of the highly stretched grid used in the boundary layer. Fig. 27 shows the velocity vectors in the flow field, where the development of the boundary layers and flow separation on the upper of airfoil are clearly visible.

5. Conclusions and outlook

An accurate and fast implicit discontinuous Galerkin method has been developed to solve the compressible Euler and Navier–Stokes equations. The developed method has been applied to compute a variety of time-accurate subsonic flow problems on arbitrary grids. The numerical results demonstrated the superior accuracy of this discontinuous Galerkin method and indicated that the use of the present implicit method leads to a significant increase in performance over its explicit counterpart, while maintaining competitive memory requirements.

Acknowledgment

The first author would like to acknowledge the partial support for this work provided by the Air Force Summer faculty Fellowship Program, while he was in residence at Computational Sciences Branch, Air Force Research Laboratory, Dayton, OH.

References

[1] Reed WH, Hill TR. Triangular mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Report, LA-UR-73-479; 1973.
[2] Cockburn B, Hou S, Shu CW. TVD Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. Math Comput 1990;55:545–81;
George PL. Automatic mesh generation. J. Wiley & Sons; 1991.
[3] Cockburn B, Shu CW. The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional system. J Comput Phys 1998;141:199–224.
[4] Cockburn B, Karniadakis G, Shu CW. The development of discontinuous Galerkin method. In: Cockburn B, Karniadakis GE, Shu CW, editors. Discontinuous Galerkin methods, theory, computation, and applications. Lecture notes in computational science and engineering, vol. 11. New York: Springer-Verlag; 2000. p. 5–50.
[5] Bassi F, Rebay S. High-order accurate discontinuous finite element solution of the 2D Euler equations. J Comput Phys 1997;138:251–85.
[6] Atkins HL, Shu CW. Quadrature free implementation of discontinuous Galerkin method for hyperbolic equations. AIAA J 1998;36(5).
[7] Bassi F, Rebay S. GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations. In: Cockburn B, Karniadakis GE, Shu CW, editors. Discontinuous Galerkin methods, theory, computation, and applications. Lecture notes in computational science and engineering, vol. 11. New York: Springer-Verlag; 2000. p. 197–208.
[8] Warburton TC, Karniadakis GE. A discontinuous Galerkin method for the viscous MHD equations. J Comput Phys 1999;152:608–41.
[9] Hesthaven JS, Warburton T. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications. Texts Appl Math 2008;56.
[10] Rasetarinera P, Hussaini MY. An efficient implicit discontinuous spectral Galerkin method. J Comput Phys 2001;172:718–38.
[11] Helenbrook BT, Mavriplis D, Atkins HL. Analysis of *p*-multigrid for continuous and discontinuous finite element discretizations. AIAA Paper 2003-3989; 2003.
[12] Fidkowski KJ, Oliver TA, Lu J, Darmofal DL. *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. J Comput Phys 2005;207(1):92–113.
[13] Luo H, Luo L, Xu K. A BGK-based discontinuous Galerkin method for the Navier–Stokes equations on arbitrary grids. Adv Appl Math Mech 2009;1(3):301–18.
[14] Luo H, Baum JD, Löhner R. A discontinuous Galerkin method using Taylor basis for compressible flows on arbitrary grids. J Comput Phys 2008;227(20):8875–93.
[15] Luo H, Baum JD, Löhner R. On the computation of steady-state compressible flows using a discontinuous Galerkin method. Int J Numer Methods Eng 2008;73(5):597–623.
[16] Luo H, Baum JD, Löhner R. A hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids. J Comput Phys 2007;225(1):686–713.
[17] Luo H, Baum JD, Löhner R. A *p*-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids. J Comput Phys 2006;211(2):767–83.
[18] Luo H, Baum JD, Löhner R. Fast, *p*-multigrid discontinuous Galerkin method for compressible flows at all speeds. AIAA J 2008;46(3):635–52.
[19] Bassi F, Rebay S. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J Comput Phys 1997;131:267–79.
[20] Bassi F, Rebay S. Discontinuous Galerkin solution of the Reynolds-averaged Navier–Stokes and k-ω turbulence model equations. J Comput Phys 2005;34:507–40.
[21] Cockburn B, Shu CW. The local discontinuous Galerkin method for time-dependent convection-diffusion system. SIAM, J Numer Anal 2001;16.
[22] Baumann CE, Oden JT. A discontinuous hp finite element method for the Euler and Navier–Stokes equations. Int J Numer Methods Fluids 1999;31.
[23] Luo H, Luo L, Norgaliev R, Mousseau VA, Dinh N. A reconstructed discontinuous Galerkin method for the compressible Navier–Stokes equations on arbitrary grids. J Comput Phys 2010;229:6961–78.
[24] Luo H, Luo L, Ali A, Norgaliev R, Cai C. A parallel, reconstructed discontinuous Galerkin method for the compressible flows on arbitrary grids. Commun Comput Phys 2011;9(2):363–89.
[25] Luo H, Baum JD, Löhner R, Fast A. Matrix-free implicit method for compressible flows on unstructured grids. J Comput Phys 1998;146(2):664–90.
[26] Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for computing low Mach number flows on unstructured grids. Int J Comput Fluid Dynam 2001;14:133–57.
[27] Luo H, Baum JD, Löhner R. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids. Comput Fluids 2001;30(2):137–59.
[28] Yee HC, Sandham ND, Djomehri MJ. Low-dissipative high-order shock-capturing method using characteristic-based filters. J Comput Phys 1999;150(1):199–238.
[29] Wang L, Mavriplis D. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretization. J Comput Phys 2007;255(2):1994–2015.
[30] Gordnier RE, Visbal MR. Compact difference scheme applied to simulation of low-sweep delta wing flow. AIAA J 2005;43(8). 00.1744-1752.
[31] Visbal MR, Gaitonde DV. On the use of higher-order finite-difference schemes on curvilinear and deforming meshes. J Comput Phys 2002;181(1):155–85.
[32] Visbal MR. Private communication.