49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2011, Orlando, Florida

AIAA 2011-685

A Diagonally Implicit Runge-Kutta Method for the Discontinuous Galerkin Solutions of the Navier-Stokes Equations

Hidehiro Segawa[1] and Hong Luo[2]
North Carolina State University, Raleigh, NC 27695, USA

Robert Nourgaliev[3]
Idaho National Laboratory, Idaho Falls, ID 83415, USA

A high-order implicit discontinuous Galerkin method is developed for the time-accurate solutions to the compressible Navier-Stokes equations on arbitrary grids. The spatial discretization is carried out using a high order discontinuous Galerkin method, where polynomial solutions are represented using a Taylor basis. A diagonally implicit Runge-Kutta method is applied for temporal discretization to the resulting ordinary differential equations. The resulting nonlinear system of equations is solved at each stage 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 viscous flow problems. The numerical results obtained indicate that the use of this implicit method leads to orders of improvements in performance over its explicit counterpart, while without significant increase in memory requirements.

I. 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

[1] Ph.D. student, Department of Mechanical and Aerospace Engineering, Student Member AIAA.
[2] Associate Professor, Department of Mechanical and Aerospace Engineering, Senior Member.
[3] Senior Scientist, Multiphysics Methods Group, Member AIAA.

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 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. Implicit methods, such as the first-order accurate backward Euler scheme, the Crank-Nicholson method, and the second-order backward differentiation formula, can be used for these types of problems. These time integration schemes are relatively efficient because they solve only one implicit set of equations per time step. However, they require a fixed time step, thus rendering them less efficient. They are not A-stable, thus rendering them less robust. They only provide a second-order temporal accuracy, thus rendering them less accurate. The development of accurate and fast arbitrary high-order implicit methods is needed in order to keep the overall higher-accuracy and higher-efficiency of the higher-order discontinuous Galerkin methods. Recently, a diagonally implicit Runge-Kutta (IRK) method, originally developed by Bijl et al[23] for the finite volume solutions of the Navier-Stokes equations is extended by Wang et al.[24] to solve the compressible Euler equations using higher-order discontinuous Galerkin methods. They conclude that the diagonally implicit Runge-Kutta method is more efficient than the second-order time integration schemes.

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. Clearly, how to efficiently solve the nonlinear system at each time step is crucial to the success of any implicit time integration methods.

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 work is strongly motivated by the need to develop an accurate, and fast arbitrary high-order implicit method for solving time-accurate flow problems in order to keep the overall higher-accuracy of the higher-order discontinuous Galerkin methods, as the use of higher order methods in space alone does not ensure a more accurate solution in that the error deduced by the time-stepping methods can be dominant. The spatial discretization is carried out using a discontinuous Galerkin method, where the polynomial solutions are constructed using a Taylor basis[13-15]. A system of nonlinear equations arising from a diagonally implicit Runge-Kutta 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 viscous flow problems. The numerical results obtained indicate that the use of the present implicit method leads to orders of improvements in performance over its explicit counterpart, while without significant increase in memory requirements.

II. 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 **U**, inviscid flux vector **F**, and viscous flux vector **G**, are defined by

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

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

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

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

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

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.

III. Numerical Method

The governing equation (2.1) is discretized using a discontinuous Galerkin finite element formulation. In addition to using a Local Discontinuous Galerkin (LDG) formulation (Rebay-Bassi scheme)[19] to discretize the viscous fluxes in the Navier-Stokes equations, a Bhatnagar-Gross-Krook **(**BGK) scheme is also developed to compute the fluxes[13] which not only couples the convective and dissipative terms together, but also includes both discontinuous and continuous representation in the flux evaluation at a cell interface through a simple hybrid gas distribution function. 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 Reference 14.

The discontinuous Galerkin finite element approximation to the governing equations (2.1) leads to the following semi-discrete system of nonlinear equations:

$$M\frac{d\mathbf{U}}{dt} + R(\mathbf{U}) = 0 \tag{3.1}$$

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. (3.1) 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. In this context, the second-order implicit backwards differencing and the second order Crank-Nicholson schemes are widely used to integrate the resulting spatially discretized Eq. (3.1) in time[22,24]. These schemes are relatively efficient because they solve only one implicit set of equations per time step. However, they are difficult to use with variable time-steps, thus rendering them less efficient. Although they can be generalized to a higher order accuracy by increasing the number of time levels involved, they require more memory to store the solutions at the previous time steps, are not A-stable beyond a second-order temporal accuracy, and are much more difficult to satisfy the geometric conservation laws, that usually required for the Arbitrary Lagrangian-Eulerian formulation. Recently, a fourth-order implicit Runge-Kutta scheme is developed for the time-accurate of the Euler equations in Reference 24 by Wang and Mavriplis. It is found that the higher-order time integration schemes such as fourth-order implicit Runge-Kutta to be more efficient in terms of computational cost for a given accuracy level as compared to lower order implicit schemes. In this work, a m-stage diagonally implicit Runge-Kutta (IRK) scheme, termed IRKm in this paper, originally developed for a finite volume method by Bijl et al is used to integrate the governing equation (3.1) in time as follows

(i) $\mathbf{U}^{(0)} = \mathbf{U}^{n}$

(ii) For k = 1, ..., m

$$\mathbf{U}^{(k)} = \mathbf{U}^{n} - \Delta t\sum_{j=1}^{k} a_{kj} M^{-1}\mathbf{R}(\mathbf{U}^{(j)}) \tag{3.2}$$

(iii) $\mathbf{U}^{n+1} = \mathbf{U}^{(m)}$

where a_{kj} are the Butcher coefficients of the scheme and n denotes the time level. The set of coefficients, a_{kj} , defines the implicit RK schemes. This The first stage is explicit due to $a_{11} = 0$. The solution in the last stage is the solution for the next time step. This class of Runge-Kutta schemes is termed the ESDIRK in the literature, which stands for Explicit first stage, Single Diagonal coefficient, diagonally Implicit Runge-Kutta. They can be easily used in the presence of variable time steps and constructed to be A- and L-stable for any temporal order. However, they require to solve a multiple nonlinear system of equations per time step. Clearly, how to devise a fast method for the solution of the nonlinear system of the equations is crucial to the success of the IRK schemes. In this work, the resulting non linear system of coupled equations (3.2) is solved using a pseudo-time marching approach. The solution at each implicit stage is obtained by treating it as a steady state problem by introducing a pseudo-time variable t_*

$$M\frac{d\mathbf{U}}{dt_*} + \mathbf{R}_*(\mathbf{U}) = 0 \tag{3.3}$$

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

$$\mathbf{R}_*(\mathbf{U}^{(k)}) = \frac{M}{\Delta t}\mathbf{U}^{(k)} + a_{ss}\mathbf{R}(\mathbf{U}^{(k)}) + S(\mathbf{U}^n) \tag{3.4}$$

with the source term

$$S(\mathbf{U}^n) = -\left[\frac{M}{\Delta t}\mathbf{U}^n - \sum_{j=1}^{k-1} a_{kj}\mathbf{R}(\mathbf{U}^{(j)})\right] \tag{3.5}$$

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:

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

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,

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

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 $\mathbf{U}_{P1}^{n+1}$

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

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

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

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 $\tilde{\mathbf{I}}$ 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

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

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 the prolongation and restriction operators between orders are local to the element. As a result, they can be easily computed by hand considering one element at a time in advance.

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, this *p*-multigrid method uses an explicit multi-stage Runge-Kutta scheme[2,3] as the iterative smoother on the higher level approximations ($p>0$), and a matrix-free implicit GMRES+LU-SGS method[27,28] as the iterative smoother on the lowest level approximation ($p=0$). 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 results obtained 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.

IV. 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 4GBytes memory) using a Suse 11.0 Linux operating system. For all the computations, the accuracy level of 0.01, I.e., two orders of magnitude drop in residual, is used to derive 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. Convection of an isentropic vortex

The convection of a 2D inviscid isentropic given for example by Dumbser et al[29] is considered in this test case to conduct a temporal convergence study of IRK3. The analytical solution to this problem at any time t is simply the passive advection of the initial solution at t=0, which provides a valuable reference for measuring the accuracy of a numerical solution. The initial condition is a linear superposition of a mean uniform flow with some perturbations δ. The free stream flow conditions are $(\rho_\infty, u_\infty, v_\infty, p_\infty) = (1,1,1,1)$. The perturbations of the velocity components *u* and *v*, entropy S, and temperature *T* for the vortex are given by

$$\begin{pmatrix}\delta u\\ \delta v\end{pmatrix} = \frac{\varepsilon}{2\pi} e^{\frac{1-r^2}{2}} \begin{pmatrix}-(y-y_0)\\ (x-x_0)\end{pmatrix}, \quad \delta S = 0, \qquad \delta T = -\frac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-r^2},$$

where $r^2 = (x-x_0)^2 + (y-y_0)^2$, (x_0,y_0) is the coordinate of the vortex center, and ε is the vortex strength. From $\rho=\rho_\infty+\delta\rho$, $u=u_\infty+\delta u$, $v=v_\infty+\delta v$, $T=T_\infty+\delta T$, and the isentropic relation, other physical variables can be determined as follows

$$\rho = T^{\frac{1}{\gamma-1}} = (T_\infty + \delta T)^{\frac{1}{\gamma-1}} = \left[T_\infty - \frac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-r^2} \right]^{\frac{1}{\gamma-1}},$$

$$\rho u = \rho(u_\infty + \delta u) = \rho\left[u_\infty - \frac{\varepsilon}{2\pi} e^{\frac{1-r^2}{2}} (y-y_0) \right],$$

$$\rho v = \rho(v_\infty + \delta v) = \rho\left[v_\infty + \frac{\varepsilon}{2\pi} e^{\frac{1-r^2}{2}} (x - x_0) \right],$$

$$p = \rho^\gamma,$$

$$e = \frac{p}{\rho(\gamma - 1)} + \frac{1}{2}(u^2 + v^2).$$

In this test case, the vortex strength ε=5, and the coordinate of the vortex center (x_0,y_0) is (5,5). The computational domain Ω is [0,10]x[0,10] and the periodic boundary conditions are imposed. The numerical solutions are obtained after ten periods of time t=100, and compared with the exact solution simply given by the initial condition. The follow L_2-norm

$$\left\| \rho_h - \rho^e \right\|_{L^2(\Omega)} = \left(\int_\Omega \left| \rho_h - \rho^e \right|^2 d\Omega \right)^{\frac{1}{2}},$$

is used to measure the error between the numerical and analytical solutions, where ρ_h is the numerical solution for the density. Figure 1 shows the triangular grid used in the computation, which consists of 8,864 triangles and has 16, 32, and 64 faces in each dimension. The spatial discretization is carried out using a DG(P2) method to ensure the overall error will be dominated by the temporal discretization. Figure 2 provides the details of the temporal convergence of the IRK3 method for this numerical experiment. As expected, the IRK3 scheme exhibits a slope of 2.98, indicating the IRK3 method indeed offer the design $O(\Delta t^3)$ order of the convergence

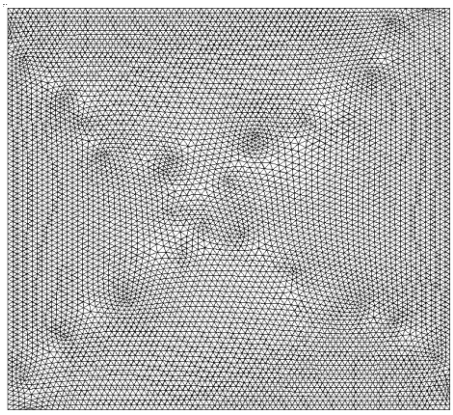

Figure 1. Computational mesh used for computing the advection of an isentropic vortex test case.

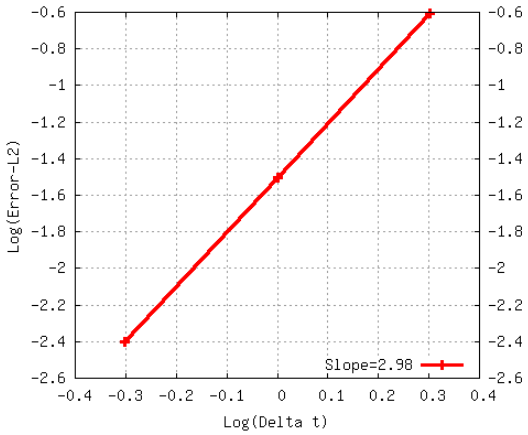

Figure 2. Results of grid-refinement study for the stationary vortex.

B.Shedding flow past a triangular wedge

This test case taken from reference 24 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 Figure 3, which contains 12,307 triangular elements, 6,248 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. Figures 4, 5, and 6 show the computed density contours in the flow fields at t=100 obtained by the first order implicit Backwards Differencing Scheme (BF1), the second order implicit Backwards Differencing Scheme (BF2), and the third order Implicit Runge-Kutta scheme (IRK3), respectively, where the spatial discretization is carried out using a third order DG method (DG(P2)). A fixed time-step size of $\Delta t = 0.05$, corresponding to a maximal CFL number of about 500, is used for all computations. Unlike Reference 23 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 performed 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 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 BDF2 displays a substantially better shape retaining property for the vortex. However, the IRK3 scheme is clearly seen to be more accurate than the BDF2 scheme, keeping and retaining the initial vortex shape almost intact.

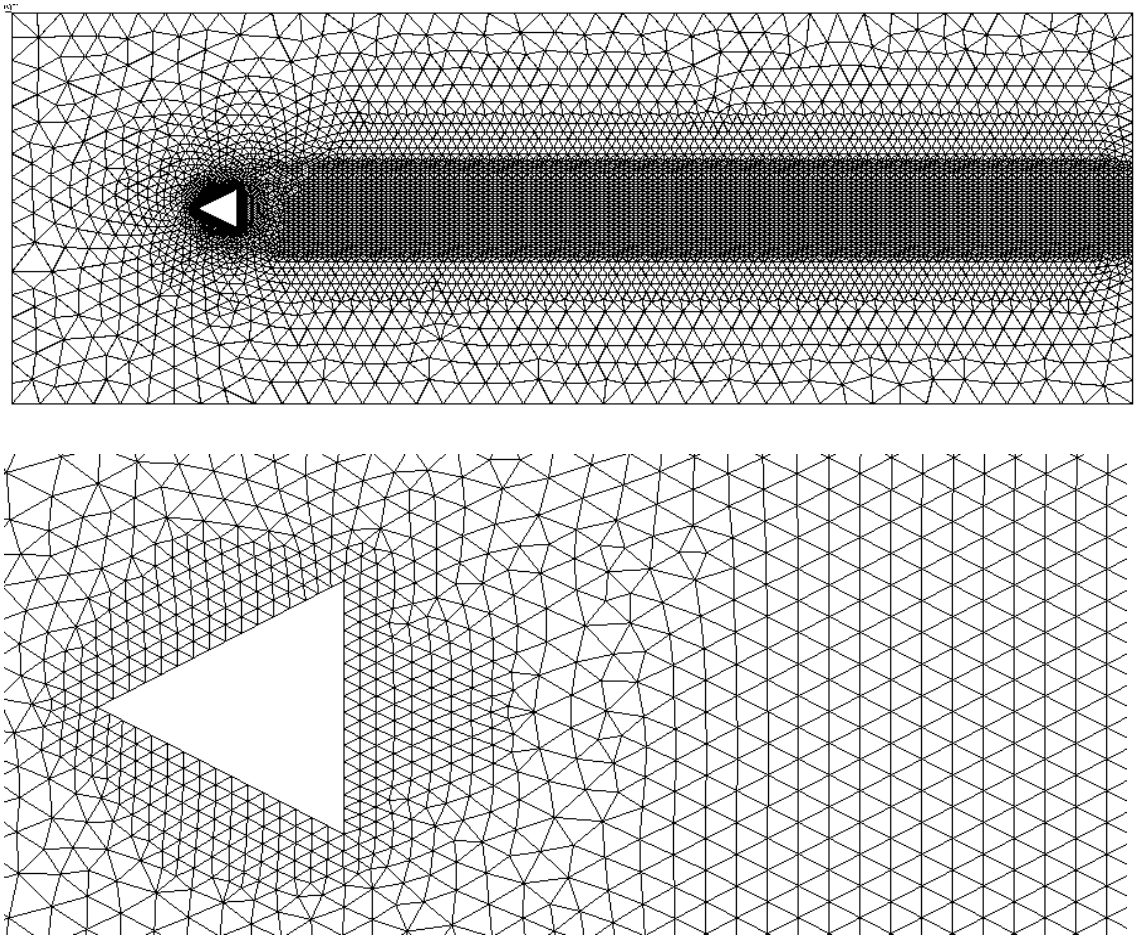

Figure 3. Mesh used for computing an inviscid flow past a wedge (nelem=12,307, npoin=6,248, nboun=189).

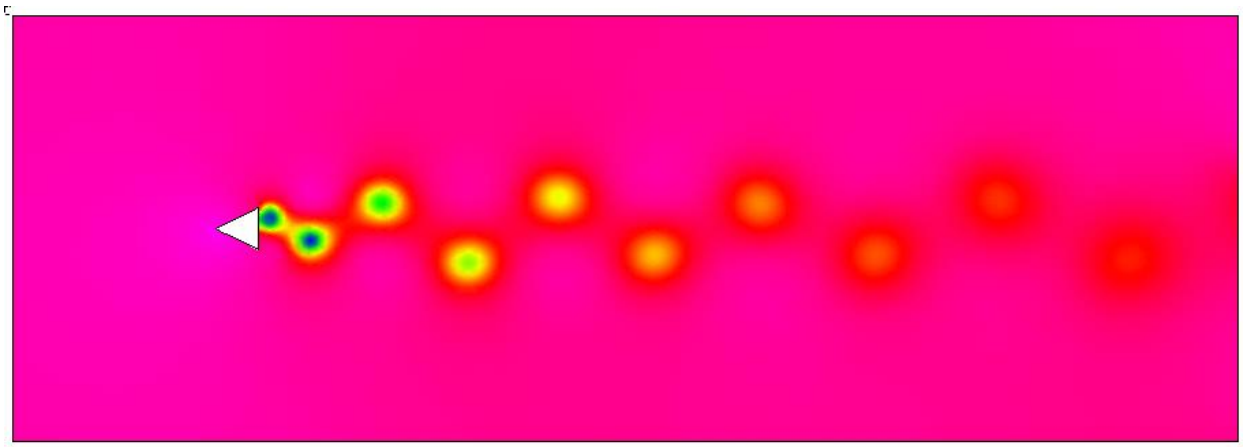

Figure 4. Computed density contours obtained using BDF1 and DG(P2) at t=100 for flow past a wedge at a Mach number of 0.2.

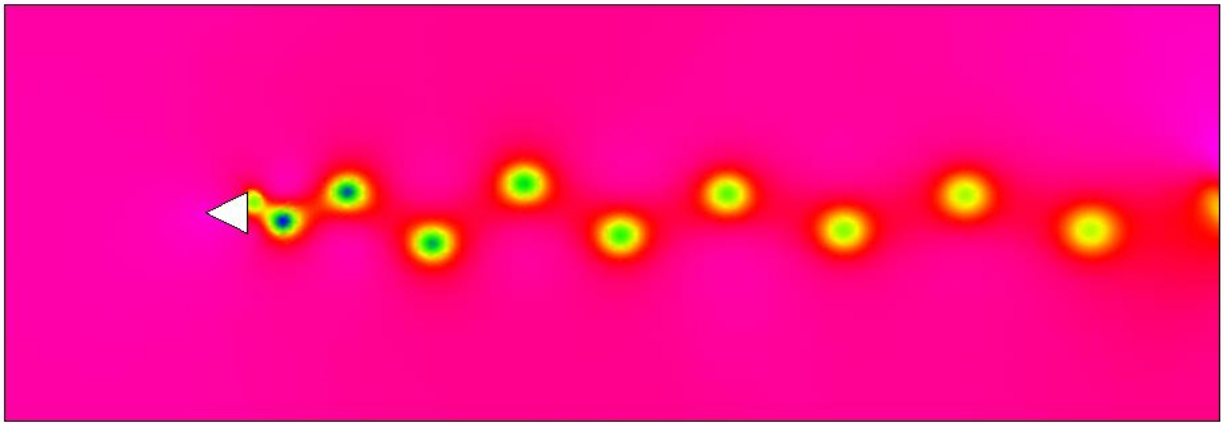

Figure 5. Computed density contours obtained using BDF2 and DG(P2) at t=100 for flow past a wedge at a Mach number of 0.2.

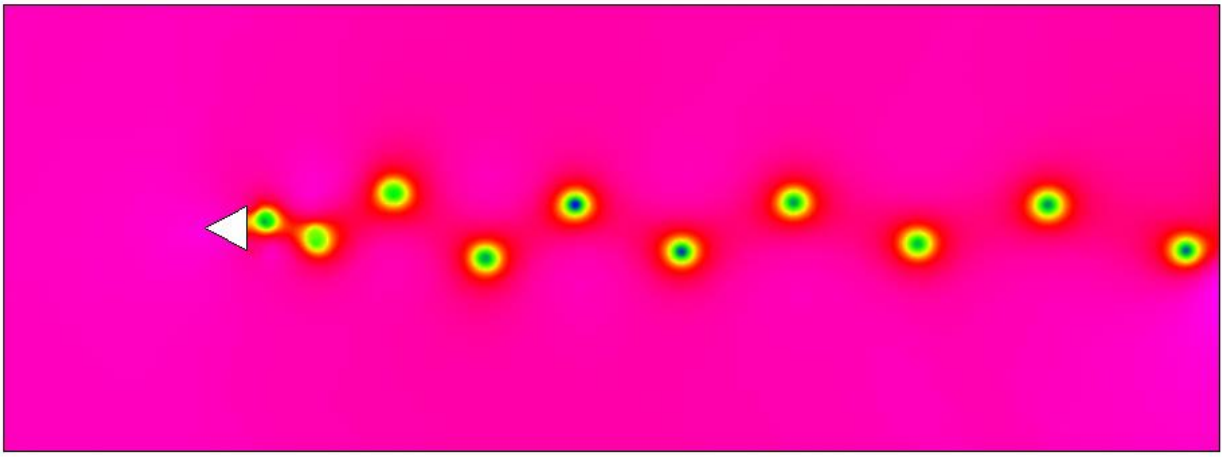

Figure 6. Computed density contours obtained using IRK3 and DG(P2) at t=100 for flow past a wedge at a Mach number of 0.2.

C. 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 Figure 7. Three computations are preformed using the IRK3 for time integration and the DG(P2) method for spatial discretization 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 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. An explicit multi-stage Runge-Kutta method would have to use a much smaller time step of 0.0002 due to the CFL condition for this case. As CPU time depends primarily on the number of time steps required in the computation, the implicit method offers more than one-order-of-magnitude improvement over its explicit counterpart, clearly demonstrating the efficiency of the present implicit method. The computed entropy contours in the flow fields at t=100 for these three cases are shown in Figure 8. The computed lift and drag coefficients are shown in Figure 9, 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 Figure 10, while those of the computed viscous lift and drag coefficients are displayed in Figure 11.

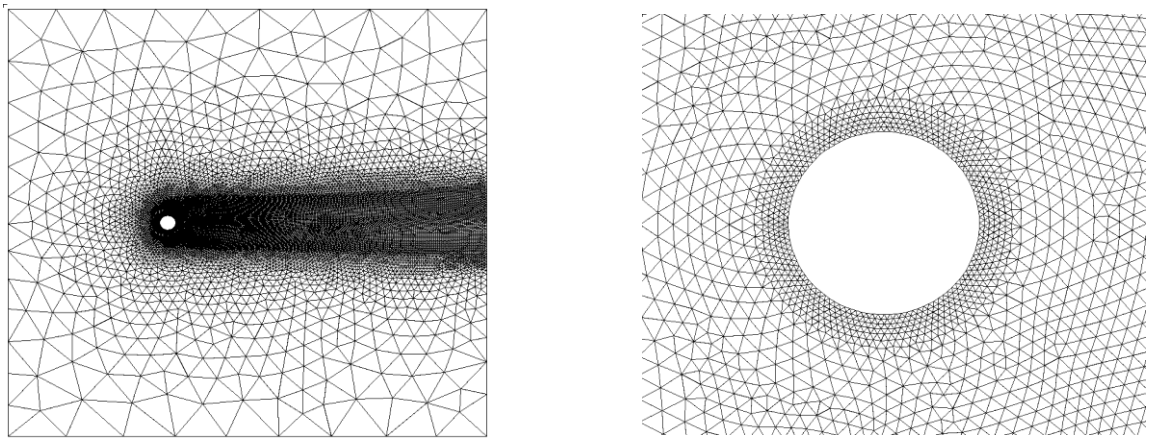

Figure 7. Mesh used for computing a viscous flow past a cylinder (nelem=21,809, npoin=11,004, nboun=199).

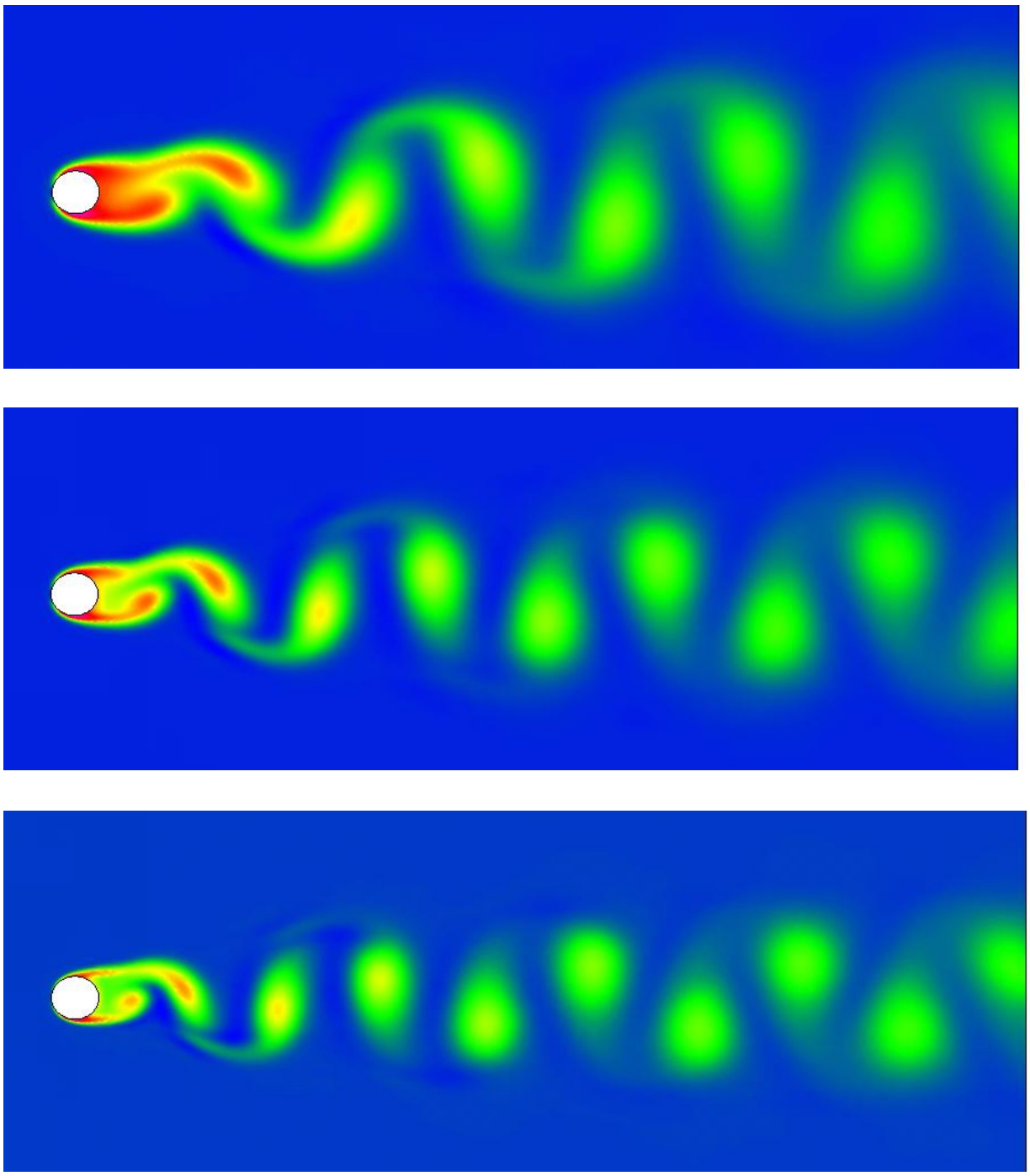

Figure 8. Computed entropy contours at t=100 for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100 (top), 200 (middle), and 400 (bottom), respectively.

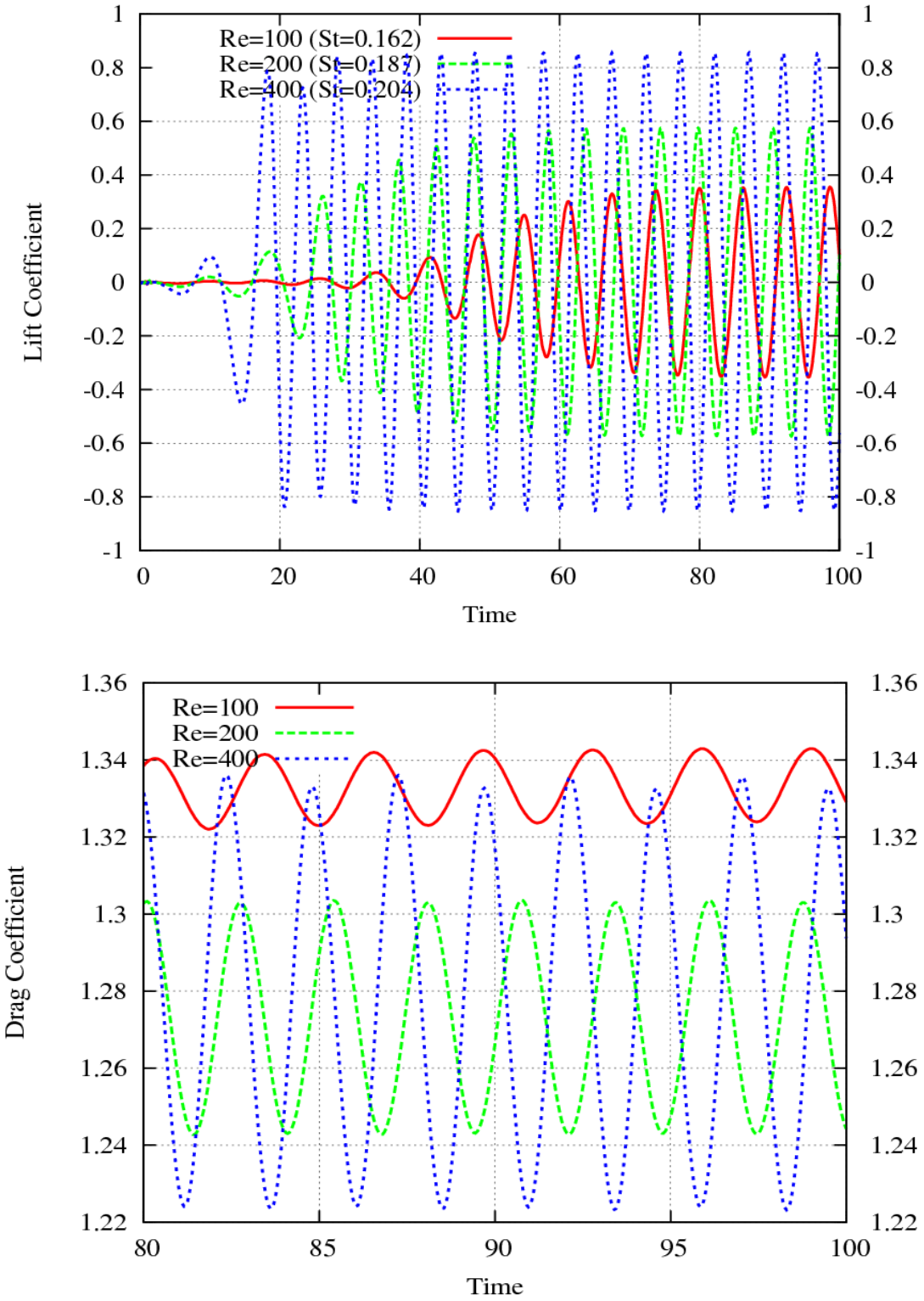

Figure 9. Time history of the computed lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

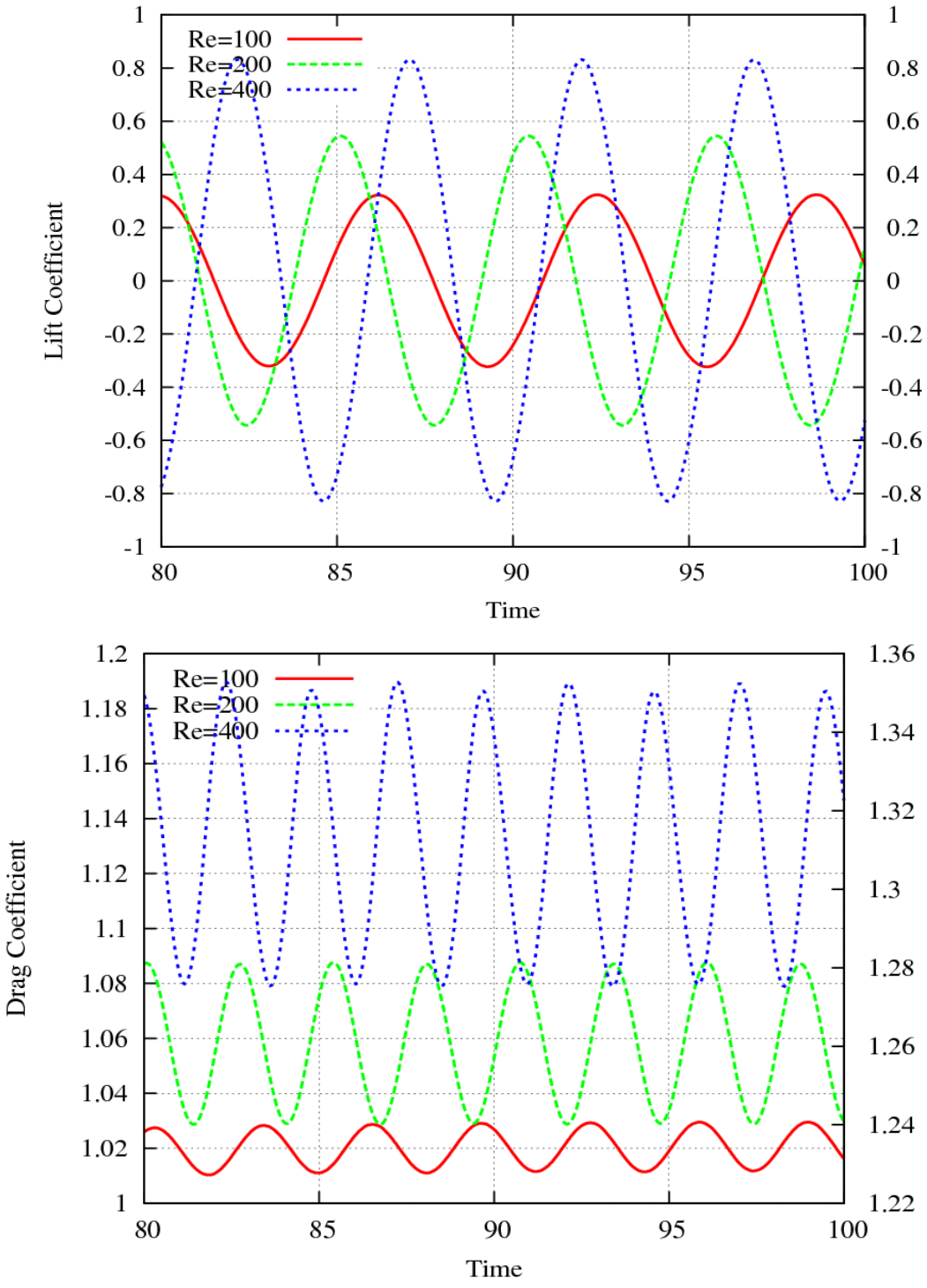

Figure 10. Time evolution of the computed pressure lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

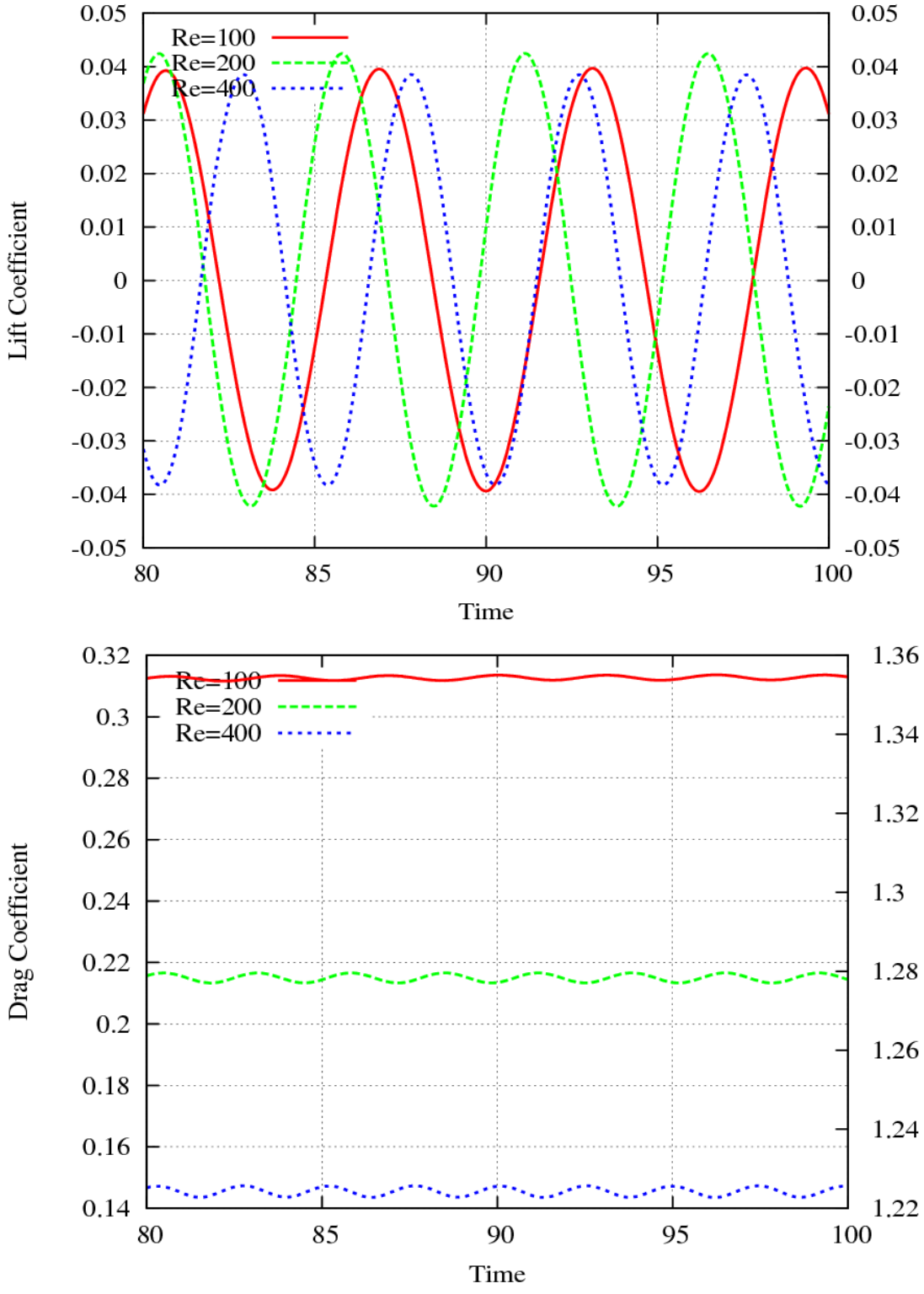

Figure 11. Time evolution of the computed viscous lift and drag coefficients for flow past a cylinder at a Mach number of 0.1, and Reynolds number of 100, 200, and 400, respectively.

D. 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 degree, 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 IRK3. The mesh used in the computation consists of 60,126 triangular elements, 30,199 grid points, and 272 boundary faces, as shown in Figure 12. The computed velocity and entropy contours are shown in Figures 13 and 14, respectively, where tail vortices and back flows are clearly visible.

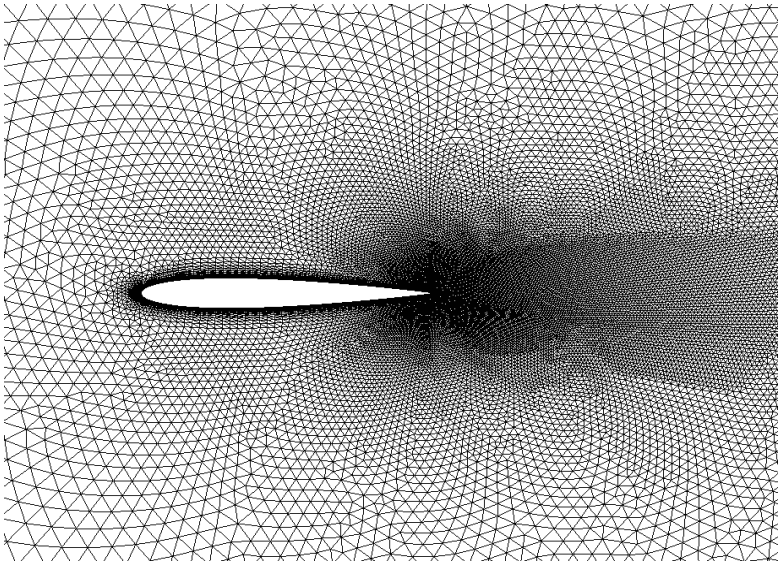

Figure 12. Mesh used for computing an inviscid flow past a wedge (nelem=60,126, npoin=30,199, and nboun=272).

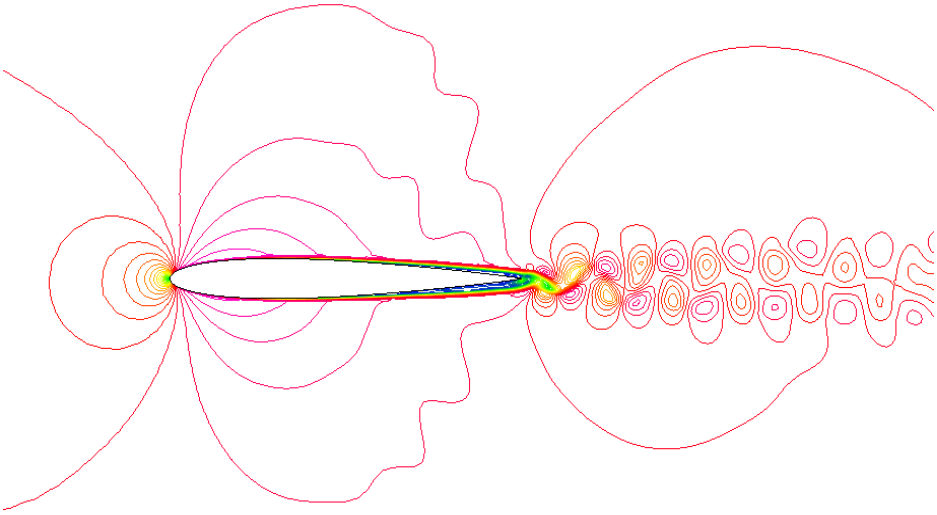

Figure 13. Computed velocity contours in the flow field for flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0, and a Reynolds number of 10,000.

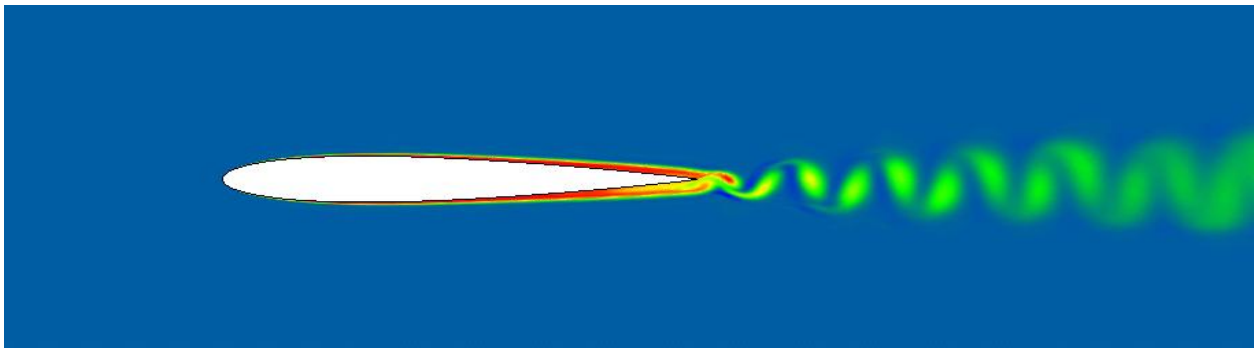

Figure 14. Computed entropy contours in the flow field for flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0, and a Reynolds number of 10,000.

E. 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. The computation is initialized with constant free-stream values in the entire domain with non-slip boundary conditions on the solid wall. Figure 15 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 IRK3 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[24-26] in Figure 16, while the comparison for the vorticity contours between the DG method and the compact method is shown in Figure 17. 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 Figure 18 to illustrate that accurate and smooth solutions are obtained using the DG(P2) method in spite of the highly stretched grid used in the boundary layer. Figure 19 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.

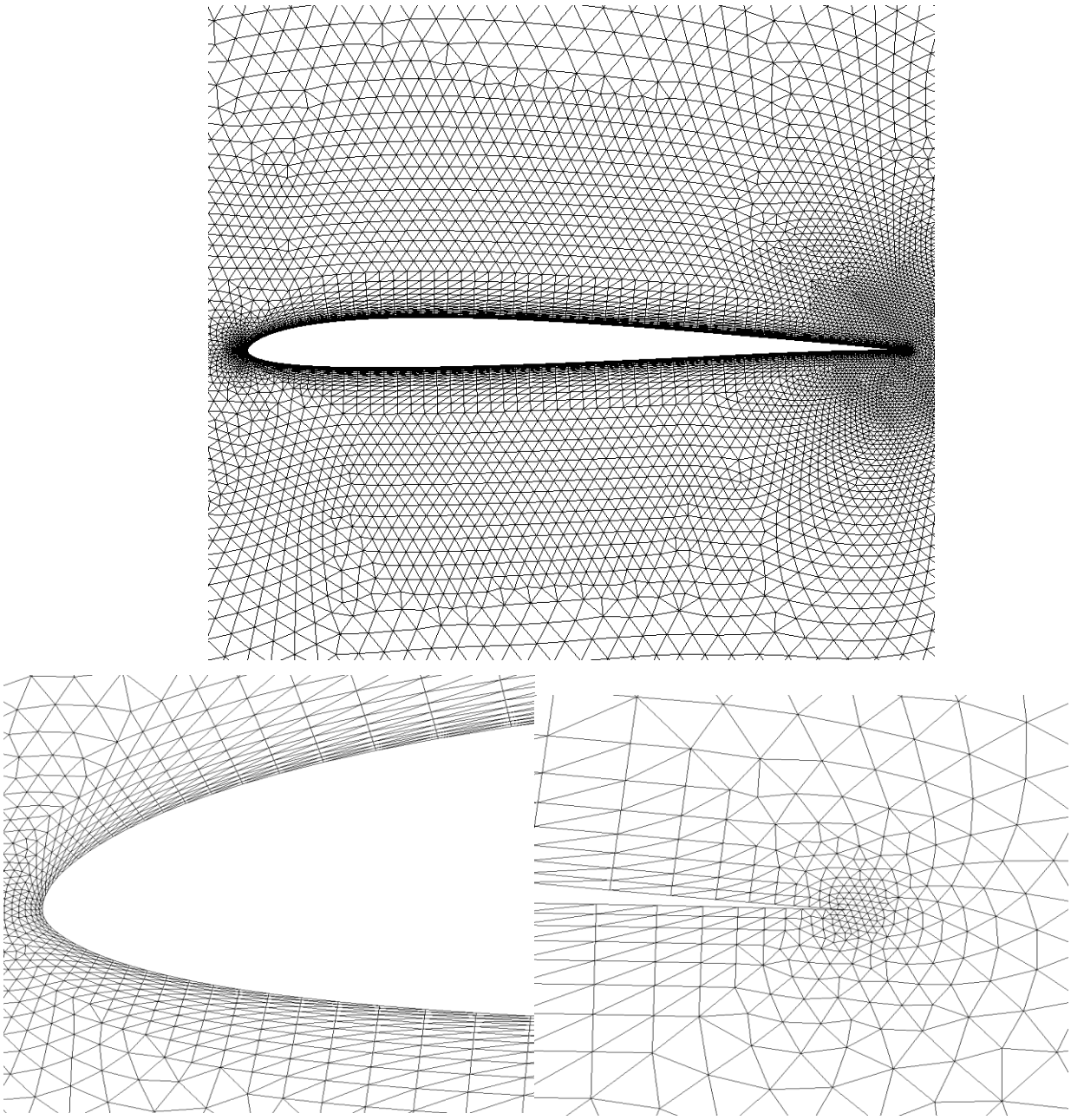

Figure 15. Grid used for computing the unsteady viscous flow past a SD7003 airfoil (nelem=50,781, npoin=25,530, nboun=279).

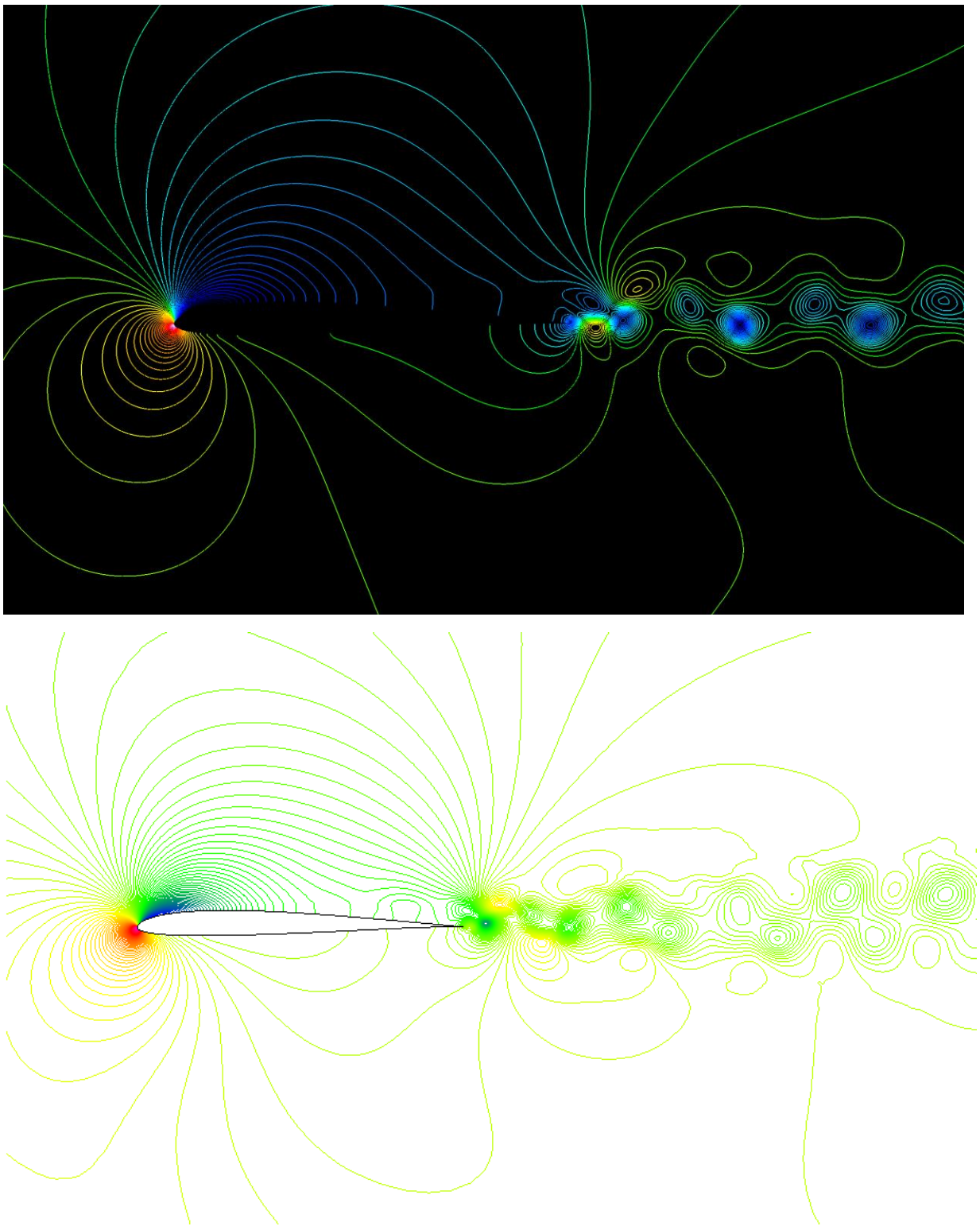

Figure 16. Computed pressure contours in the flow field obtained by the compact method (upper) and DG(P2) method (lower) for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4^{o}, and a Reynolds number of 10,000.

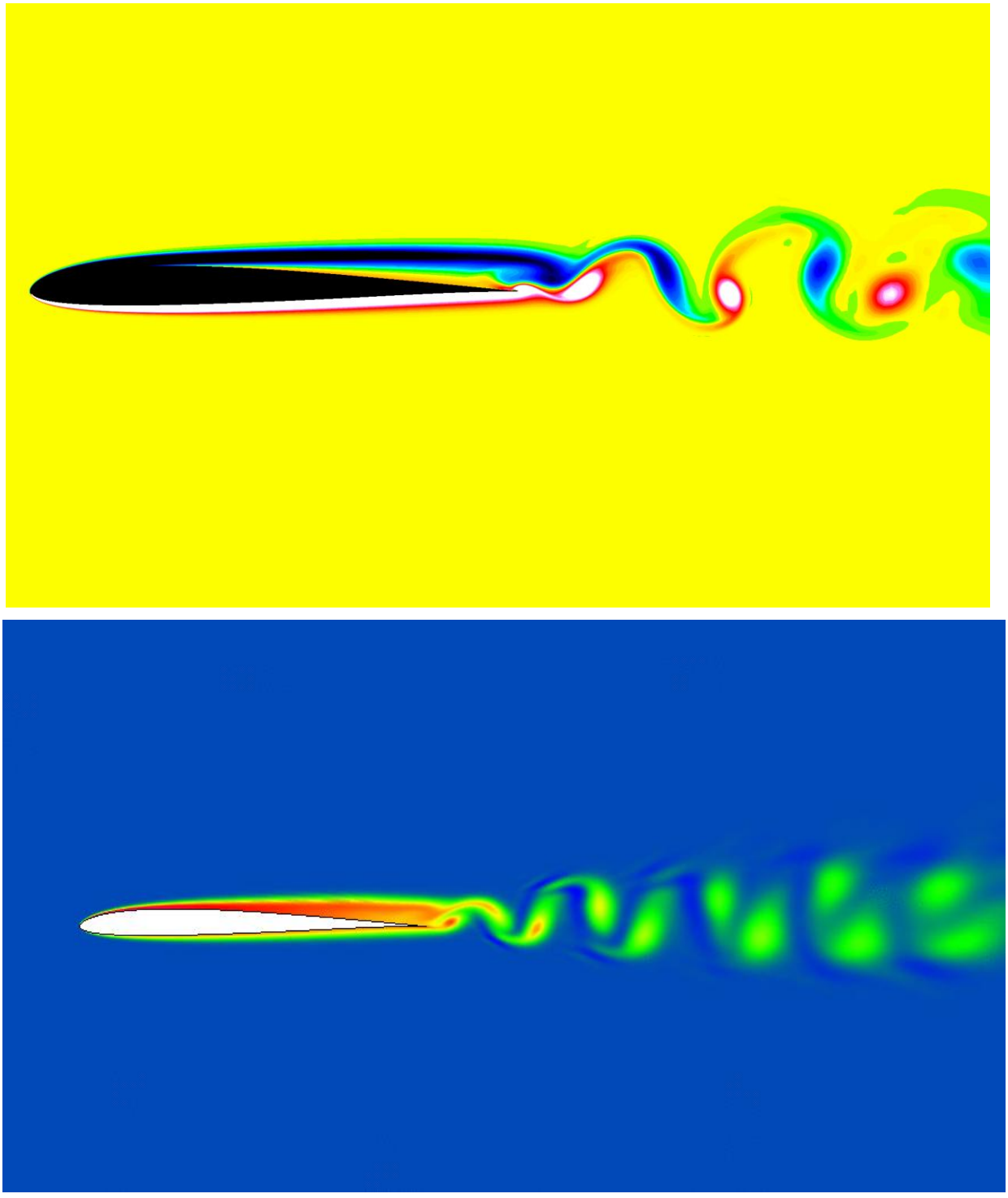

Figure 17. Computed vorticity contours (upper) obtained by the compact method and computed entropy contours obtained by the IRK3 and DG(P2) method in the flow field for the viscous flow past a SD7003 airfoil at a Mach number of 0.1, an angle of attack of 4^o, and a Reynolds number of 10,000.

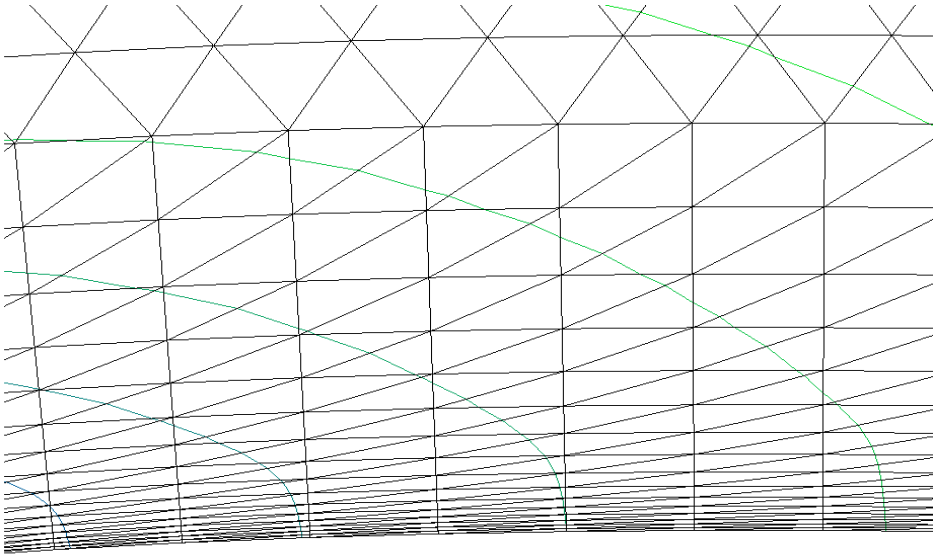

Figure 18. Computed pressure contours on the upper surface of the airfoil for the viscous flow past a SD7003 airfoil.

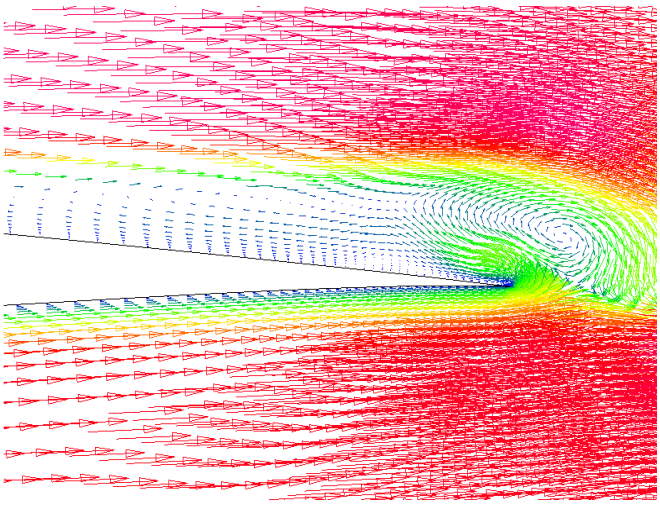

Figure 19. Computed velocity vectors near the airfoil for the viscous flow past a SD7003 airfoil.

V. 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 flow problems on arbitrary grids. The numerical results demonstrated the superior accuracy of this discontinuous Galerkin method and indicated that the use of the implicit Runge-Kutta method leads to a significant increase in performance over its explicit counterpart, while maintaining competitive memory requirements.

Acknowledgments

This manuscript has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/CON-09-16255) with the U.S. Department of Energy. The United States Government retains and the published, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this

manuscript, or allow others to do so, for United States Government purposes. The authors would like to acknowledge the partial support for this work provided by DOE under Nuclear Engineering University Program.

References

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

[2]B. Cockburn, S. Hou, and C. W. Shu, ``TVD Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for conservation laws IV: the Multidimensional Case," **Mathematics of Computation**, Vol. 55, pp. 545-581, 1990.eorge, P. L., Automatic Mesh Generation, J. Wiley & Sons, 1991.

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

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

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

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

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

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

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

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

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

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

[13]H. Luo and K. Xu, ``A BGK-based Discontinuous Galerkin Method for the Navier-Stokes Equations on Arbitrary Grids", AIAA-2008-0748, 2008.

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

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

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

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

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

[19]F. Bassi and S. Rebay, ``A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations," **Journal of Computational Physics**, Vol. 131, pp. 267-279, 1997.

[20]B. Cockburn and C.W. Shu, ``The Local Discontinuous Galerkin Method for Time-dependent Convection-Diffusion System, **SIAM, Journal of Numerical** Analysis, Vo. 16, 2001.

[21]C. E. Baumann and J. T. Oden, ``A Discontinuous hp Finite Element Method for the Euler and Navier-Stokes Equations," **International Journal for Numerical Methods in Fluids**, Vol. 31, 1999.

[22]H. Luo, Segawa, H., and Visbal, M.R, `` An Implicit Discontinuous Galerkin Method for the Unsteady Compressible Navier-Stokes Equations", AIAA-2009-0951, 2009.

[23]H. Bijl, M.H. Carpenter, V.N. Vasta, and C. A. Kennedy, ``Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow", **Journal of Computational Physics**, Vol. 179, pp. 313-329, 2002.

[24]L. Wang and D. Mavriplis, ``Implicit Solution of the Unsteady Euler Equations for High-Order Accurate Discontinuous Galerkin Discretization," **Journal of Computational Physics**, Vol. 255, No. 2, pp. 1994-2015, 2007.

[24] R. E. Gordnier and M. R. Visbal, ``Compact Difference Scheme Applied to Simulation of Low-Sweep Delta Wing Flow", **AIAA Journal**, Vol. 43, No. 8, 00.1744-1752, 2005.

[25]M. R. Visbal and D. V. Gaitonde, ``On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes", **Journal of Computational Physics**, Vol. 181, No. 1, pp. 155-185, 2002.

[26]M. R. Visbal, Private Communication.
[27]H. Luo, J. D. Baum, and R. Löhner, ``A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids," **Journal of Computational Physics**, Vol. 146, No. 2, pp. 664-690, 1998.
[28]H. Luo, D. Sharov, J. D. Baum, and R. Löhner, ``A Class of Matrix-Free Implicit Methods for Compressible Flows on Unstructured Grids," **Proceedings of the First International Conference on Computational Fluid Dynamics**, Kyoto, Japan, 10-14, July 2000.
[29]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**, 227:8209-8253, 2008.