Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 211 (2006) 767–783

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids

Hong Luo [a,*], Joseph D. Baum [a], Rainald Löhner [b]

[a] *Center for Applied Computational Sciences, Science Applications International Corporation, 1710 SAIC Drive, MS 2-6-9, McLean, VA 22102, United States*

[b] *Institute for Computational Sciences and Informatics, George Mason University, Fairfax, VA 22030, United States*

Received 21 December 2004; received in revised form 27 June 2005; accepted 29 June 2005
Available online 25 August 2005

Abstract

A p-multigrid (p = polynomial degree) discontinuous Galerkin method is presented for the solution of the compressible Euler equations on unstructured grids. The method operates on a sequence of solution approximations of different polynomial orders. A distinct feature of this p-multigrid method is to use different time integration schemes on different approximation levels, resulting in an accurate, fast, and low memory method that can be used to accelerate the convergence of the Euler equations to a steady state for discontinuous Galerkin methods. The developed method is used to compute the compressible flows for a variety of test problems on unstructured grids. The numerical results obtained strongly indicate the order independent property of this p-multigrid method. An overall speed-up factor more than one order of magnitude for both second- and third-order solutions of all test cases in comparison with the explicit method is demonstrated.

1. Introduction

The use of unstructured meshes for computational fluid dynamics problems has become widespread due to their ability to discretize arbitrarily complex geometries and due to the ease of adaptation in enhancing the solution accuracy and efficiency through the use of adaptive refinement techniques. In recent years, significant progress has been made in developing numerical algorithms for the solution of the compressible Euler and Navier–Stokes equations on unstructured grids.

The discontinuous Galerkin methods [1–13] have recently become popular for the solution of systems of conservation laws. Nowadays, they are widely used in the computational fluid dynamics, computational

[*] Corresponding author. Tel.: +1 703 676 5957; fax: +1 703 676 5323.
E-mail address: hong.luo@saic.com (H. Luo).

acoustics, and computational electromagnetics. The discontinuous Galerkin methods combine two advantageous features commonly associated to finite element and finite volume methods. As in classical finite element methods, accuracy is obtained by means of high-order polynomial approximation within an element rather than by wide stencils as in the case of finite volume methods. The physics of wave propagation is, however, accounted for by solving the Riemann problems that arise from the discontinuous representation of the solution at element interfaces. In this respect, the methods are therefore similar to finite volume methods. The discontinuous Galerkin methods have many features: (1) The methods are well suited for complex geometries since they can be applied on unstructured grids. In addition, the methods can also handle nonconforming elements, where the grids are allowed to have hanging nodes. (2) The methods are highly parallelizable, as they are compact and each element is independent. Since the elements are discontinuous, and the inter-element communications are minimal, domain decomposition can be efficiently employed. The compactness also allows for structured and simplified coding for the methods. (3) They can easily handle adaptive strategies, since refining or coarsening a grid can be achieved without considering the continuity restriction commonly associated with the conforming elements. The methods allow easy implementation of *hp*-refinement, for example, the order of accuracy, or shape, can vary from element to element. (4) They have several useful mathematical properties with respect to conservation, stability, and convergence. However, these methods have their own weaknesses. Compared to the finite element methods and finite volume methods, the discontinuous Galerkin methods require the solutions of systems of equations with more unknowns for the same grids. Consequently, these methods have been recognized as expensive in terms of both computational cost and storage requirement.

Most efforts in the development of the discontinuous Galerkin methods are primarily focused on the spatial discretization. The temporal discretization methods have lagged far behind. Usually, explicit temporal discretizations such as multi-stage TVD (total variation diminishing) Runge–Kutta schemes [1,7–11] are used to advance the solution 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 simulations and especially for high-order solutions, the rate of convergence slows down dramatically, resulting in inefficient solution techniques to steady state solution. In order to speed up convergence, a multigrid strategy or an implicit temporal discretization is required.

In general, implicit methods require the solution of a linear system of equations arising from the linearization of a fully implicit scheme at each time step or iteration. Recently, significant efforts have been made to develop efficient implicit solution methods for discontinuous Galerkin methods [12,13]. Unfortunately, the drawback is that they require a considerable amount of memory to store the Jacobian matrix, which may be prohibitive for large-scale problems and high-order solutions. This is especially troublesome for discontinuous Galerkin method, which is recognized as expensive in terms of both computational operation count and storage requirement. Even in the implementation of so called matrix-free implicit methods [13], where only a block diagonal matrix is required to store, the memory requirements can still be extremely demanding. The block diagonal matrix requires a storage of (neqns $\times$ ndegr) $\times$ (neqns $\times$ ndegr) $\times$ nelem, where neqns is the number of unknown variables (4 for 2D, and 5 for 3D Euler equations), ndegr is the degrees of freedom for the polynomial (3 for P_1, 6 for P_2, and 10 for P_3 for triangle element in 2D; 4 for P_1, 10 for P_2, and 20 for P_3 for tetrahedral element in 3D), and nelem is the number of elements for the grid. For example, for a fourth-order (cubic polynomial finite element approximation P_3) discontinuous Galerkin method in 3D, the storage of this block diagonal matrix alone requires 10,000 words per element!

p-Multigrid method is an iterative scheme in which systems of equations arising from compact, high-order finite element discretization, such as spectral-*hp* or discontinuous Galerkin formulation, are solved by recursively iterating on solution approximations of different polynomial order. For example, to solve equations derived using a polynomial approximation order of 2, the solution can be iterated on at an approximation order of $p = 2$, 1, and 0. The p component of this algorithm was introduced by Rönquist and

Patera [14], and analyzed by Maday and Munoz [15] for a one-dimensional, Galerkin spectral element discretization of the Laplace equation. Bassi and Rebay presented their work on the *p*-multigrid method for the Euler equations in [16]. Helenfrook et al. [17] examined the performance of *p*-multigrid for Laplace equation and the convection equation in two dimensions.

The objective of the effort discussed in this paper is to develop a fast, low storage *p*-multigrid method for discontinuous Galerkin methods to solve the compressible Euler equations on unstructured grids. Explicit multi-stage Runge–Kutta method [1,7], matrix-free symmetric Gauss–Seidel (SGS) method [18], and matrix-free lower–upper symmetric Gauss–Seidel (LU-SGS) [18,19,23] method are implemented, used, and discussed as iterative smoothers. Unlike the traditional *p*-multigrid methods where the same time integration scheme is used on all approximation levels, the present *p*-multigrid method uses multi-stage Runge–Kutta scheme as the iterative smoother on the higher level approximations, and matrix-free SGS method as the iterative smoother on the lowest level approximation in an attempt to significantly reduce the storage requirements and fully capitalize on the mature matrix-free implicit methods developed for the finite volume methods [18,19]. The developed method has been used to compute compressible flows for a variety of test problems on unstructured grids. The numerical results obtained strongly imply the order independent property of this *p*-multigrid method, and indicate that this novel *p*-multigrid method provides a very efficient way for accelerating the convergence of the compressible Euler equations to a steady state without significant increase in memory requirement. Efficiency increases of one order of magnitude for both second- and third-order solutions of all test cases in comparison with the explicit method are demonstrated by the use of this *p*-multigrid method.

2. Governing equations

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

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

where, the conservative state vector **U** and the inviscid flux vectors **F** are defined by

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

where the summation convention has been used and 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. This set of equations is completed by the addition of the equation of state

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

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

3. Discontinuous Galerkin method

To formulate the discontinuous Galerkin method, we first introduce the following weak formulation of (2.1), which is obtained by multiplying Eq. (2.1) by a test function **W**, integrating over the domain Ω, and performing an integration by parts:

$$\int_{\Omega} \frac{\partial \mathbf{U}}{\partial t} \mathbf{W} \, \mathrm{d}\Omega + \int_{\Gamma} \mathbf{F}_j \mathbf{n}_j \mathbf{W} \, \mathrm{d}\Gamma - \int_{\Omega} \mathbf{F}_j \frac{\partial \mathbf{W}}{\partial x_j} \mathrm{d}\Omega = 0 \quad \forall \mathbf{W}, \tag{3.1}$$

where Γ $(= \partial\Omega)$ denotes the boundary of Ω, and $\mathbf{n}_j$ the unit outward normal vector to the boundary.

Assuming that Ω_h is a classical triangulation of Ω where the domain Ω is subdivided into a collection of nonoverlapping elements Ω_e, the following semi-discrete form of (3.1) is obtained by applying (3.1) on each element Ω_e

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_e} \mathbf{U}_h \mathbf{W}_h \, \mathrm{d}\Omega + \int_{\Gamma_e} \mathbf{F}_j(\mathbf{U}_h) \mathbf{n}_j \mathbf{W}_h \, \mathrm{d}\Gamma - \int_{\Omega_e} \mathbf{F}_j(\mathbf{U}_h) \frac{\partial \mathbf{W}_h}{\partial x_j} \mathrm{d}\Omega = 0 \quad \forall \mathbf{W}_h, \tag{3.2}$$

where $\mathbf{U}_h$ and $\mathbf{W}_h$ represent the finite element approximations to the analytical solution $\mathbf{U}$ and test function $\mathbf{W}$, respectively. Assume the approximate solution and test function be piece-wise polynomials in each element and let the polynomial basis function be $B_m(\mathbf{x})$, then $\mathbf{U}_h$ and $\mathbf{W}_h$ can be expressed as

$$\mathbf{U}_h(\mathbf{x}, t) = \sum_{m=1}^{N} \mathbf{U}_m(t) B_m(\mathbf{x}), \qquad \mathbf{W}_h(\mathbf{x}) = \sum_{m=1}^{N} \mathbf{W}_m B_m(\mathbf{x}), \tag{3.3}$$

where $B_m(\mathbf{x})$, $1 \leqslant m \leqslant N$ is the basis function of the polynomials of degree p. The dimension of the polynomial space, $N = N(p, d)$ depends on the degree of the polynomials of the expansion p, and the number of spatial dimensions d, as

$$N = \frac{(p+1)(p+2)\cdots(p+d)}{d!} \quad \text{for } d = 1, 2, 3. \tag{3.4}$$

Eq. (3.2) must be satisfied for any test function $\mathbf{W}_h$. Since B_n is the basis for $\mathbf{W}_h$, (3.2) is, therefore, equivalent to the following system of N equations:

$$\frac{\mathrm{d}\mathbf{U}_m}{\mathrm{d}t} \int_{\Omega_e} B_m B_n \, \mathrm{d}\Omega + \int_{\Gamma_e} \mathbf{F}_j(\mathbf{U}_h) \mathbf{n}_j B_n \, \mathrm{d}\Gamma - \int_{\Omega_e} \mathbf{F}_j(\mathbf{U}_h) \frac{\partial B_n}{\partial x_j} \mathrm{d}\Omega = 0, \quad 1 \leqslant n \leqslant N, \tag{3.5}$$

where $\mathbf{U}_h$ is replaced with Eq. (3.3). Since the numerical solution $\mathbf{U}_h$ is discontinuous between element interfaces, the interface fluxes are not uniquely defined. The flux function $\mathbf{F}_j(\mathbf{U}_h)\mathbf{n}_j$ appearing in the second term of Eq. (3.5) is replaced by a numerical Riemann flux function $\mathbf{H}\left(\mathbf{U}_h^{\mathrm{L}}, \mathbf{U}_h^{\mathrm{R}}, \mathbf{n}\right)$, where $\mathbf{U}_h^{\mathrm{L}}$ and $\mathbf{U}_h^{\mathrm{R}}$ are the conservative state vector at the left and right side of the element boundary. In order to guarantee consistency and conservation, $\mathbf{H}(\mathbf{U}^{\mathrm{L}}, \mathbf{U}^{\mathrm{R}}, \mathbf{n})$ is required to satisfy

$$\mathbf{H}(\mathbf{U}, \mathbf{U}, \mathbf{n}) = \mathbf{F}_j(\mathbf{U})\mathbf{n}_j, \qquad \mathbf{H}(\mathbf{U}, \mathbf{V}, \mathbf{n}) = -\mathbf{H}(\mathbf{V}, \mathbf{U}, \mathbf{n}). \tag{3.6}$$

This scheme is called discontinuous Galerkin method of degree p, or in short notation "DG(p) method". Note that discontinuous Galerkin formulations are very similar to finite volume schemes, especially in their use of numerical fluxes. Indeed, the classical first-order cell-centered finite volume scheme exactly corresponds to the DG(0) method, i.e., to the discontinuous Galerkin method using piece-wise constant polynomial. Consequently, the DG(p) methods with $p > 0$ can be regarded as a "natural" generalization of finite volume methods to higher order methods. By simply increasing the degree p of the polynomials, DG methods of corresponding higher orders are obtained.

In the present work, the Riemann flux function is approximated using the HLLC approximate Riemann solver [20], which has been successfully used to compute compressible viscous and turbulent flows on both structured grids [21] and unstructured grids [22]. This HLLC scheme is found to have the following properties: (1) exact preservation of isolated contact and shear waves, (2) positivity-preserving of scalar quantity, (3) enforcement of entropy condition. In addition, the implementation of HLLC Riemann solver is easier and the computational cost is lower compared with other available Riemann solvers.

The domain and boundary integrals in Eq. (3.5) are calculated using Gauss quadrature formulas. The number of quadrature points used is chosen to integrate exactly polynomials of order of $2p$ on the reference element. In the case of linear, quadratic, and cubic shape function, the domain integrals are evaluated using three, six, and twelve points, respectively, and the boundary integrals are evaluated using two, three, and four points, respectively.

By assembling together all the elemental contributions, a system of ordinary differential equations governing the evolution in time of the discrete solution can be written as

$$M\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathbf{R}(\mathbf{U}), \tag{3.7}$$

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. Since the shape functions $\mathbf{B}|_{\Omega_e}$ are nonzero within element Ω_e only, the mass matrix M has a block diagonal structure that couples the N degrees of freedom of each component of the unknown vector only within Ω_e. As a result, the inverse of the mass matrix M can be easily computed by hand considering one element at a time in advance.

The semi-discrete system can be integrated in time using explicit methods. For example, the following explicit three-stage third-order TVD Runge–Kutta scheme [1,7]

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

is widely used to advance the solution in time. This method is linearly stable for a Courant number less than or equal to $1/(2p+1)$. The inefficiency of the explicit method due to this rather restrictive CFL condition motivates us to develop the p-multigrid method to accelerate the convergence of the Euler equations to a steady state.

4. *p*-Multigrid method

Nowadays, geometric multigrid methods are widely and 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 multigrid acceleration can drastically reduce the computational costs. p-Multigrid method is a natural extension of geometric 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 4, 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. In this study, we are only interested in DG(2) and DG(1) methods. For both methods, a two level V-cycle p-multigrid method has been used to drive the iterations. More specifically, this two level p-multigrid method consists of the following steps at each p-multigrid cycle:

(1) Perform a time-step at the highest approximation order, be piece-wise linear P_1 or piece-wise quadratic P_2, which yields the initial solution $\mathbf{U}^{n+1}_{\mathrm{P}_{1,2}}$.
(2) Transfer the flow solution and residual to the lowest approximation level: piece-wise constant P_0. This can be readily obtained using the shape function as

$$\mathbf{U}_{\mathrm{P}_0}(\Omega_e) = \sum_{i=1}^{N} \mathbf{U}_{i\mathrm{P}_{1,2}}^{n+1} B_i(\mathbf{x}_c), \qquad \mathbf{R}_{\mathrm{P}_0}(\Omega_e) = \sum_{i=1}^{N} \mathbf{R}_i\left(\mathbf{U}_{\mathrm{P}_{1,2}}^{n+1}\right) B_i(\mathbf{x}_c), \tag{4.1}$$

where $\mathbf{x}_c$ is the coordinates of the center of element Ω_e.

(3) Compute the force terms on the lowest approximation level,

$$\mathbf{F}_{\mathrm{P}_0} = \mathbf{R}_{\mathrm{P}_0} - \mathbf{R}(\mathbf{U}_{\mathrm{P}_0}). \tag{4.2}$$

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

$$\mathbf{R} = \mathbf{R}(\mathbf{U}_{\mathrm{P}_0}) + \mathbf{F}_{\mathrm{P}_0}, \tag{4.3}$$

which yields the solution at the lowest level $\mathbf{U}_{\mathrm{P}_0}^{n+1}$.

(5) Interpolate the correction $\mathbf{C}_{\mathrm{P}_0}$ back from the lowest level to update the highest level solution

$$\mathbf{C}_{\mathrm{P}_0} = \mathbf{U}_{\mathrm{P}_0}^{n+1} - \mathbf{U}_{\mathrm{P}_0}, \qquad \widetilde{\mathbf{U}}_{\mathrm{P}_{1,2}}^{n+1} = \mathbf{U}_{\mathrm{P}_{1,2}}^{n+1} + \mathbf{C}_{\mathrm{P}_0}. \tag{4.4}$$

The above one p-multigrid cycle will produce a better solution at the higher level from an initial solution at the same level.

In general, the same time integration scheme is applied to advance the solution on both levels $\mathrm{P}_{1,2}$ and P_0. However, 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, in contrast to the success that geometric multigrid methods enjoyed to accelerate the convergence of the Euler equations using the multi-stage TVD Runge–Kutta explicit scheme as an iterative smoother. This is maybe due to the severely anisotropic (hyperbolic) nature of the Euler equations and the isotropic (elliptic) nature of p-multigrid iterations. Implicit schemes have better convergence properties. Unfortunately, this is achieved at the expense of dramatic storage increase, making the implicit iterative smoother impractical, if not impossible to use for large-scale problems and especially with high-order discontinuous Galerkin methods. The approach proposed here is to use different time integration schemes at different approximation levels: the multi-stage TVD Runge–Kutta explicit scheme is used on the highest level $\mathrm{P}_{1,2}$, where one cannot afford to use implicit iterative smoother because of storage consideration, and the implicit scheme is used in the lowest level P_0, where the storage requirement is not as demanding as in the higher level. Note that on level P_0, the discontinuous Galerkin method, corresponding to zeroth-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. Using Euler implicit time-integration, the spatially discretized Euler equations on P_0 level can be linearized in time and written as

$$\left(\frac{V}{\Delta t}\mathbf{I} - \frac{\partial \mathbf{R}_{\mathrm{P}_0}}{\partial \mathbf{U}_{\mathrm{P}_0}}\right)\Delta \mathbf{U}_{\mathrm{P}_0} = \mathbf{R}_{\mathrm{P}_0}, \tag{4.5}$$

where V is the element volume, and $\mathbf{R}_{\mathrm{P}_0}$ is the right-hand-side residual. Note that the subscript P_0 will be omitted from here on, if there is no confusion. Eq. (4.5) represents a large system of linear simultaneous algebraic equations that needs to be solved. In this study, symmetric Gauss–Seidel (SGS) method [17] is used to solve the linear system (4.5). The SGS method with k iterations, called SGS(k) here after, can be written as

0. Initialization:

$$\Delta \mathbf{U}^0 = 0, \tag{4.6}$$

1. Forward Gauss–Seidel iteration:

$$(D+L)\Delta\mathbf{U}^{k+\frac{1}{2}} + U\Delta\mathbf{U}^{k} = \mathbf{R}, \tag{4.7}$$

2. Backward Gauss–Seidel iteration:

$$(D+U)\Delta\mathbf{U}^{k+1} + L\Delta\mathbf{U}^{k+\frac{1}{2}} = \mathbf{R}, \tag{4.8}$$

where U, L, and D represent strict upper, strict lower, and diagonal matrices, respectively, and k is the number of iterations for SGS method. The main advantage of SGS method is that it does not require any additional storage beyond that of the matrix itself. Note that if only one iteration is used in the SGS method and the initial guess is set to zero, the resulting method is nothing but so called LU-SGS method [18,22], which can be written as
Lower (forward) sweep:

$$(D+L)\Delta\mathbf{U}^{\star} = \mathbf{R}, \tag{4.9}$$

Upper (backward) sweep:

$$(D+U)\Delta\mathbf{U} = D\Delta\mathbf{U}^{\star}. \tag{4.10}$$

It is clear that the above algorithms involve primarily the Jacobian matrix-solution incremental vector product. Such operation can be approximately replaced by computing increments of the flux vector product. Such Jacobian matrix-solution incremental vector can then be approximately replaced by computing increments of the flux vector. This is achieved by using scalar dissipation to derive the left-hand-side matrix. The detailed matrix-free approach can be found in [17,18]. The most remarkable achievement of this approximation is that there is no need to store the upper- and lower-matrices U and L, which substantially reduces the memory requirements. The only matrix needed to store is the diagonal matrix D, which requires a memory of neqns $\times$ neqns $\times$ nelem, where neqns is the number of solution vector variables (4 for 2D, and 5 for 3D Euler equations), and nelem is the number of elements for the grid. The storage of diagonal matrix only requires 16 words per element in 2D and 25 words per element in 3D. Compare to the memory requirement of about 100 words per element for the explicit DG(1) method and 250 words per element for the explicit DG(2) method in 2D, the additional storage requirement for the present p-multigrid method is not significant at all.

5. Numerical results

Only shockless (smooth subsonic and supersonic) flows are considered here in an effort to ensure that computational efficiency is not affected by a limiting processing, which is required to eliminate spurious oscillations in the vicinity of discontinuities. Note that construction of an accurate, efficient, and robust limiter remains one of the issues and challenges for the DG methods. All of the computations are performed on an Dell Inspiron 8200 laptop computer (1.6 GHz CPU with 1 GB memory) running the Suse 9.1 Linux operating system. The relative L_2 norm of the density residual is taken as a criterion to test convergence history. In most of the numerical tests performed, the p-multigrid method is compared to the explicit method in order to emphasize the efficiency. The explicit method uses the explicit three-stage third-order TVD Runge–Kutta time-stepping scheme described in Section 3 with local time stepping technique. All computations are started with uniform flow and carried out using a CFL number of 2 for the explicit method and a CFL number of 5000 for the implicit method used as iterative smoother in the p-multigrid method, unless stated otherwise.

5.1. Supersonic vortex flow

The problem under consideration is an inviscid supersonic vortex flow. This test case is chosen to verify the implementation of the developed computer code and assess the order of accuracy of the discontinuous Galerkin method. For nonlinear hyperbolic systems like the compressible Euler equations, there are practically no a priori error estimates available. Therefore, quantitative measurement of the order of accuracy and discretization error associated with the discontinuous Galerkin method can only be obtained through numerical test cases, for which exact, closed form, analytical solutions exist. For this purpose, we consider the solution of the 2D compressible Euler equations to the supersonic vortex flow problem, that is one of the few nontrivial problems of the 2D Euler equations for which a smooth analytical solution is known. The inviscid, isentropic, supersonic flow of a compressible fluid between concentric circular arcs presents a flow where the velocity varies inversely with radius. The expression for density ρ as a function of radius r is given by

$$\rho(r) = \rho_i \left\{ 1 + \frac{\gamma - 1}{2} M_i^2 \left[1 - \left(\frac{r_i}{r} \right)^2 \right] \right\}^{\frac{1}{\gamma - 1}},$$

where M_i and r_i are the Mach number and radius at the inner arc. In the present calculation, the Mach number, density, and pressure at the inner radius r_i are specified 2.25, 1, and $1/\gamma$, respectively. The inner and outer radius are 1 and 1.384. In the following, we compute the numerical solutions to this problem using DG(0), DG(1), and DG(2) methods on four successively refined grids and evaluate the L^2-error of the solutions. All the computations are initialized with the exact solution. Fig. 1(a) and (b) shows four meshes used in the computation and the density contours in the flow fields obtained by the DG(1) method, respectively. The detailed results of this test case are presented in Table 1a–c. They show the number of elements, the number of degrees of freedom, the L^2-error of the solutions, and the order of convergence. These tables clearly indicate that the discontinuous Galerkin method applied to the steady compressible Euler equations exhibits a full $O(h^{p+1})$ order of convergence on smooth solutions. Fig. 1(c) provides the details of the spatial accuracy of each DG method for this numerical experiment. Finally, Fig. 1(d) shows the L^2-error of the DG(p), $0 \leqslant p \leqslant 2$ methods plotted against the number of degrees of freedom, where one can clearly see that a higher order DG method requires much less number of degrees of freedom than a lower order DG method to achieve the same accuracy. Note that although the discontinuous Galerkin method is based on ideas of upwinding, it is not limited to convergence of first-order but allows arbitrarily high orders of convergence for smooth solutions depending on the order of finite element approximation.

5.2. Subsonic flow in a channel with a circular bump on the lower wall

The second example is the well-known Ni's test case: a subsonic flow in a channel with a 10% thick circular bump on the bottom. The length of the channel is 3, its height 1, and its width 0.5. Inlet Mach number is 0.5. The mesh, which contains 839 grid points, 1559 elements, and 117 boundary points, is depicted in Fig. 2(a). The computed pressure contours obtained with P_1 (second-order accurate) method in the flow field are shown in Fig. 2(b). Fig. 2(c) and (d) displays a comparison of convergence histories versus time steps and CPU, respectively, for the P_0 (first-order accurate) and P_1 RKDG methods. One can clearly see that the convergence of the TVD RKDG method deteriorates drastically for P_1 finite element approximation, as a result of a larger number of degrees of freedom and smaller time-steps for P_1 element. Fig. 2(e) and (f) displays a comparison of convergence histories versus time steps and CPU, respectively, for the P_0 element approximation among 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. It is clear that

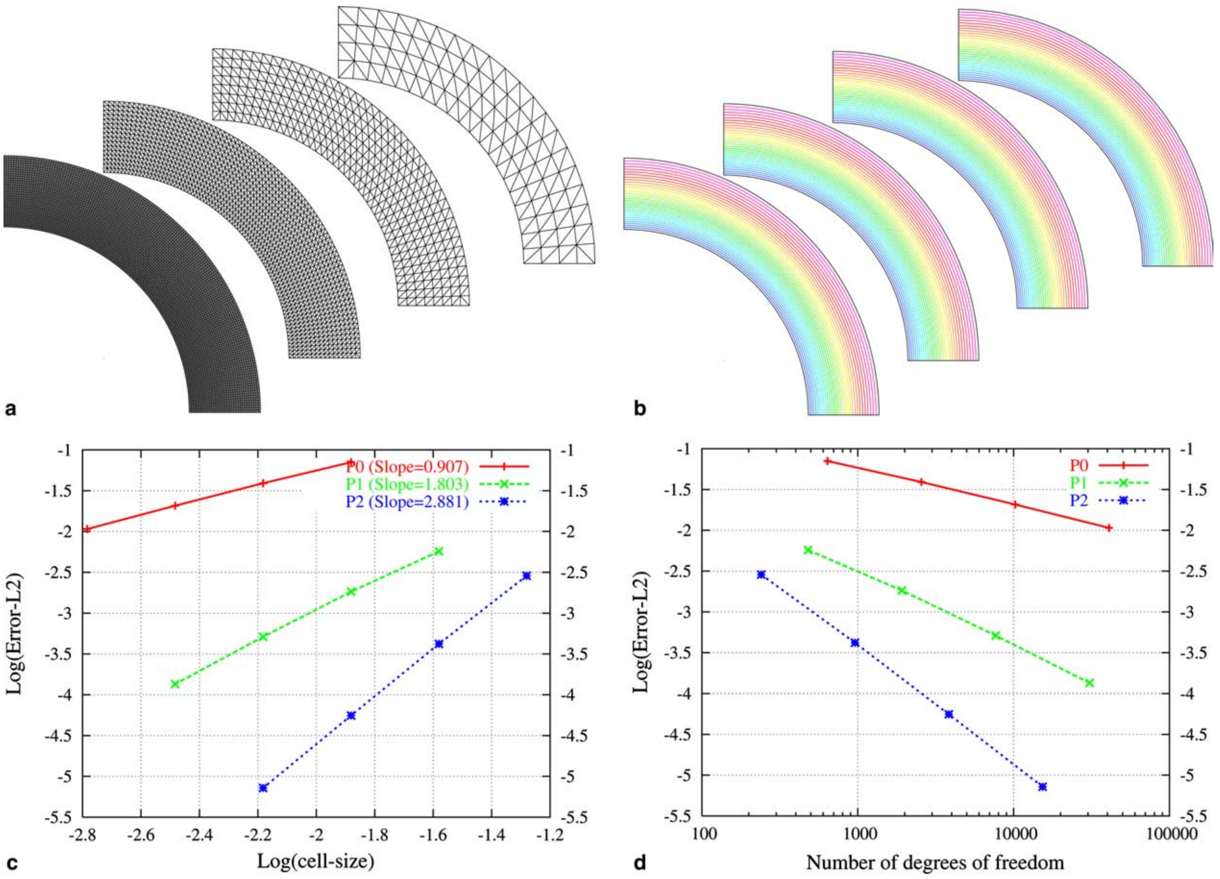


Fig. 1. (a) Sequences of four successively globally refined meshes used in DG(1) method for computing supersonic vortex flow problem. (b) Computed density contours for supersonic vortex flow problem using DG(1) method. (c) Accuracy summary for supersonic vortex flow for discontinuous Galerkin method using piecewise constant, piecewise linear, and piecewise quadratic element approximations. (d) L^2-error of numerical solutions against the number of degrees of freedom for supersonic vortex flow by DG(p) method, $0 \leqslant p \leqslant 2$.

both LU-SGS and SGS(10) methods are superior to their explicit counterpart, and SGS(10) gives the best convergence history. It is worth noting that per time step, the present LU-SGS method costs about the same as the three-stage Runge–Kutta explicit method. Fig. 2(g) and (h) displays a comparison of convergence histories versus time steps and CPU, respectively, for the P_1 element approximation among the three-stage TVD Runge–Kutta explicit method, the p-multigrid method with three-stage TVDRK smoother, p-multigrid method with the matrix-free LU-SGS smoother, p-multigrid method with the matrix-free SGS(5) smoother, and p-multigrid method with the matrix-free SGS(10) smoother. One can clearly see that both SGS and LU-SGS methods are far more efficient than TVDRK method as a smoother, and the SGS method with five iterations seems to give the best performance. Fig. 2(i) and (j) shows a comparison of convergence histories versus time steps and CPU time, respectively, for the P_1 and P_2 element approximation between the three-stage TVD Runge–Kutta explicit method, and p-multigrid method with the matrix-free SGS(5) smoother. The mesh used for P_2 computations is chosen such as that the number of degrees of freedom for P_1 and P_2 computation is about the same.

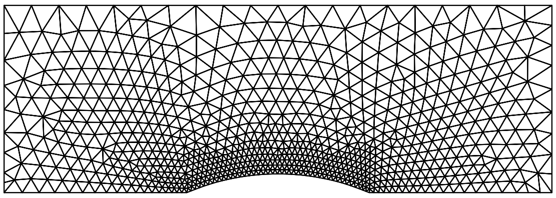
Table 1
Supersonic vortex flow problem

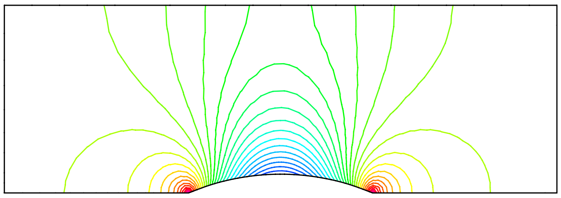
No. elements	No. DOFs	L^2-error	Order
(a) DG(0) is of order O(h)			
640	640	7.05667E−03	–
2560	2560	3.90032E−03	0.855
10,240	10,240	2.06871E−03	0.915
40,960	40,960	1.06955E−03	0.952
(b) DG(1) is of order O(h^2)			
160	480	5.72664E−03	–
640	1920	1.83358E−03	1.644
2560	7680	5.12743E−04	1.839
10,240	30,720	1.34865E−04	1.927
(c) DG(2) is of order O(h^3)			
40	240	2.86088E−03	–
160	960	4.19567E−04	2.775
640	3840	5.56673E−05	2.916
2560	15,360	7.20112E−06	2.951

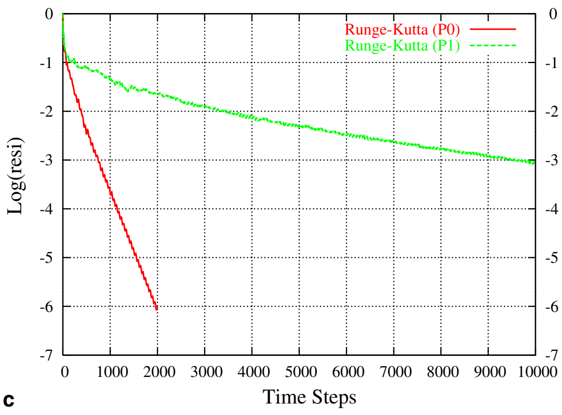
The coarse mesh used for P_2 computation has 778 elements, 431 grid points, and 82 boundary points. The present p-multigrid method can obtain the convergence in nearly the same number of cycles at $p = 2$ as $p = 1$, and P_2 computation actually requires less cpu time than the P_1 computation. The reason that P_1 and P_2 computations converge at about the same rate for the same number of unknowns is partially due to the fact that P_2–P_0 V-cycle provides more appropriate p-multigrid than P_1–P_0 V-cycle. Considering that P_2 approximation is much more accurate than P_1 approximation for the same number of degrees of freedom (as demonstrated in test case 1), this indicates that the present p-multigrid method is able to obtain more accurate solution using less computing time, which makes the present p-multigrid method especially attractive for DG(2) method. For this case, the p-multigrid method is about 33 times faster for P_1 computation and 40 times faster for P_2 computation than its 3 stage TVDRK counterpart.

5.3. Subsonic flow past a NACA0012 airfoil

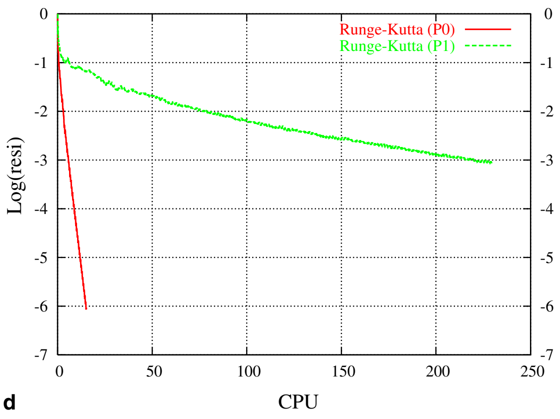
The third example is the subsonic flow past a NACA0012 airfoil at a Mach number of 0.63, and an angle of attack 2°. The mesh, which contains 3537 grid points, 6891 elements, and 183 boundary points, is depicted in Fig. 3(a). The computed pressure contours obtained using P_1-element approximation in the flow field are shown in Fig. 3(b). Fig. 3(c) and (d) displays a comparison of convergence histories versus time steps and CPU, respectively, for the 3-stage TVD Runge–Kutta explicit scheme between P_0 element and P_1 element approximation. The trends are almost exactly the same as for the last test case with drastically deteriorating convergence when from P_0 finite element approximation to P_1 finite element approximation. Fig. 3(e) and (f) illustrates a comparison of convergence histories versus time steps and CPU, respectively, for the P_0 element approximation using 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. Again, one can observe that both LU-SGS and SGS(10) methods are superior to their explicit counterpart, and SGS(10) exhibits the fastest convergence. Fig. 3(g) and (h) displays a comparison of convergence histories versus time steps and CPU, respectively, for the P_1 element approximation among the three-stage TVD Runge–Kutta explicit method, the p-multigrid method with three-stage TVDRK smoother, p-multigrid method with the matrix-free LU-SGS smoother, p-multigrid method with the matrix-free SGS(5) smoother, and p-multigrid method with the matrix-free SGS(10) smoother. One can clearly see that both SGS and LU-SGS methods are far more efficient than TVDRK method as a smoother, and the SGS method with 10 iterations seems to give the best performance. Fig. 3(i) and (j) shows a comparison of convergence histories versus time steps and CPU time, respectively, for the P_1 and P_2 element approximation between the three-stage TVD Runge–Kutta explicit method, and p-multigrid method with the matrix-free SGS(5) smoother. The mesh used for P_2 computations was chosen such as that the number of degrees of freedom for P_1 and P_2 computation is about the same. The coarse mesh used for P_2 computation has 3447 elements, 1786 grid points, and 125 boundary points. Again, the present p-multigrid method can obtain the convergence in nearly the same number


a


b

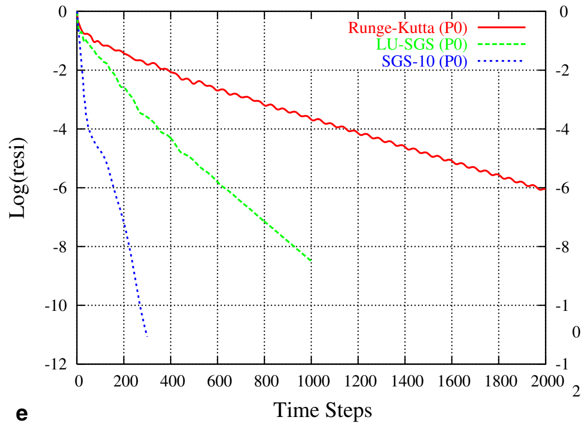


c

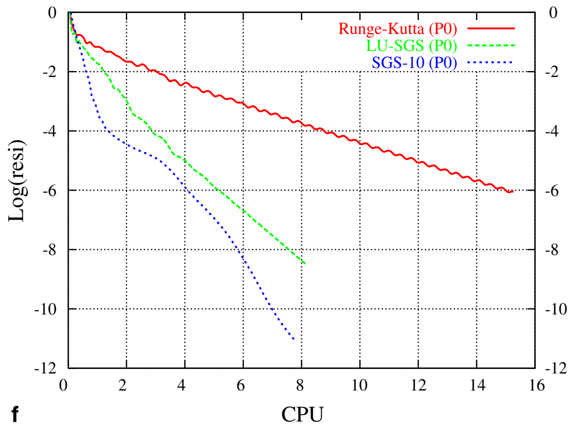


d

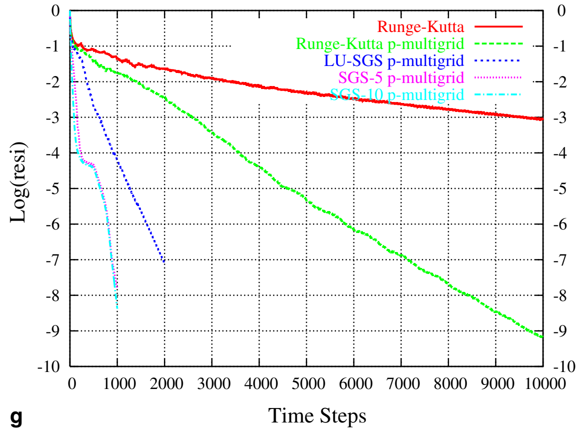
Fig. 2. (a) Unstructured mesh used for computing subsonic flow past Ni's bump configuration (nelem = 1559, npoin = 839, nboun = 117). (b) Computed pressure contours in the channel at $M_\infty = 0.5$, $\alpha = 0°$. (c) Convergence history versus time steps for subsonic channel flow using 3-stage TVD Runge–Kutta explicit method between P_0-element approximation and P_1-element approximation. (d) Convergence history versus CPU time for subsonic channel flow using 3-stage TVD Runge–Kutta explicit method between P_0-element approximation and P_1-element approximation. (e) Convergence history versus time steps for subsonic channel flow for P_0-element approximation among 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. (f) Convergence history versus CPU time for subsonic channel flow for P_0-element approximation among 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. (g) Convergence history versus time steps for subsonic channel flow for P_1-element approximation among 3-stage TVD Runge–Kutta explicit method, 3-stage TVD Runge–Kutta p-multigrid method, matrix-free LU-SGS p-multigrid method, and matrix-free SGS-10 p-multigrid method. (h) Convergence history versus CPU time for subsonic channel flow for P_1-element approximation among 3-stage TVD Runge–Kutta explicit method, 3-stage TVD Runge–Kutta p-multigrid method, matrix-free LU-SGS p-multigrid method, and matrix-free SGS-10 p-multigrid method. (i) Convergence history versus time steps for subsonic channel flow using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 p-multigrid method. j: Convergence history versus CPU time for subsonic channel flow using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 p-multigrid method.



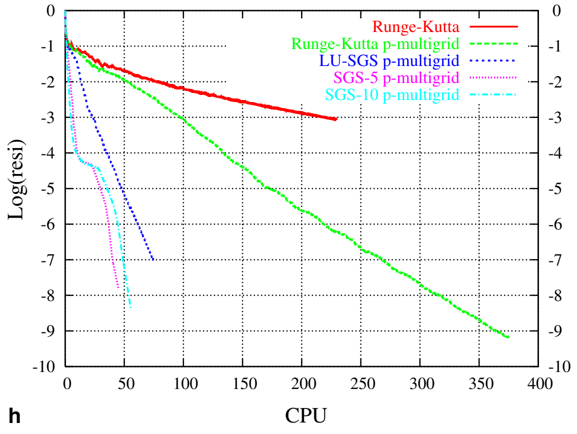
e



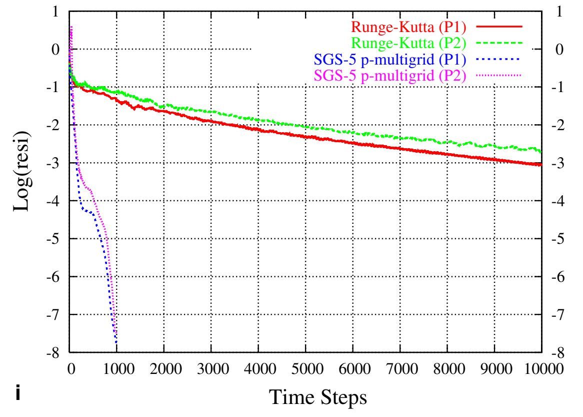
f



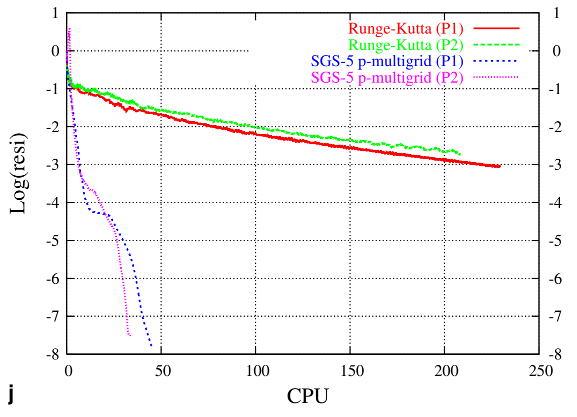
g



h



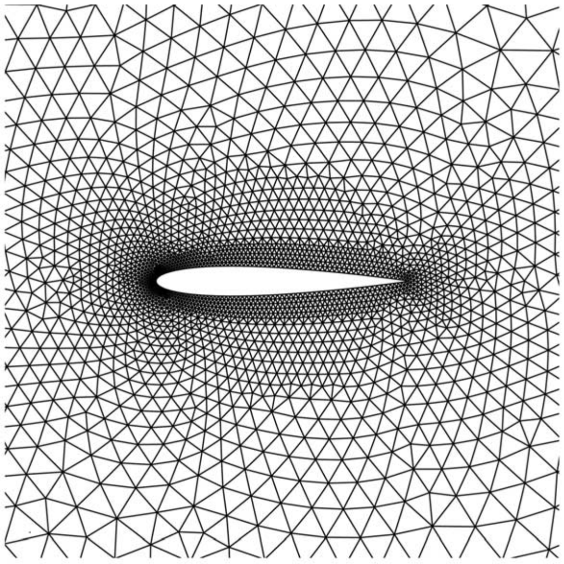
i

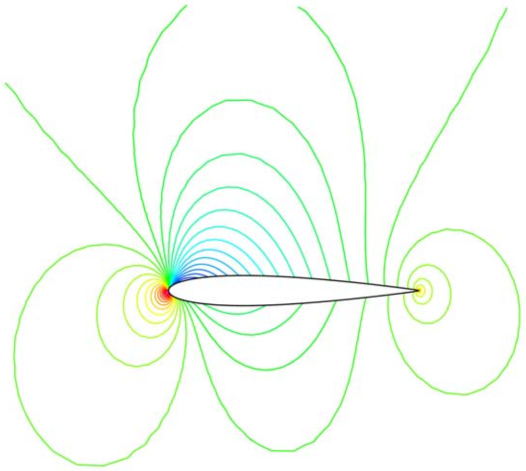


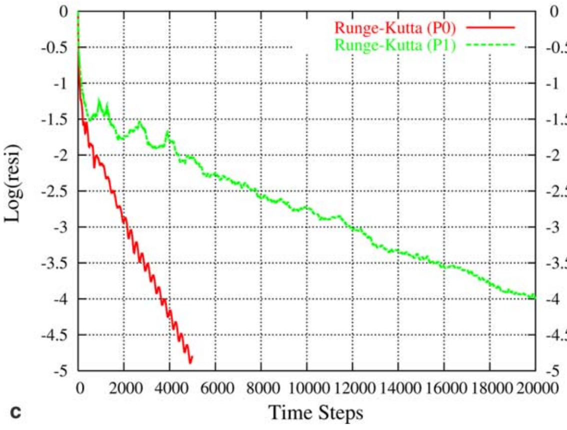
j

Fig. 2 (*continued*)

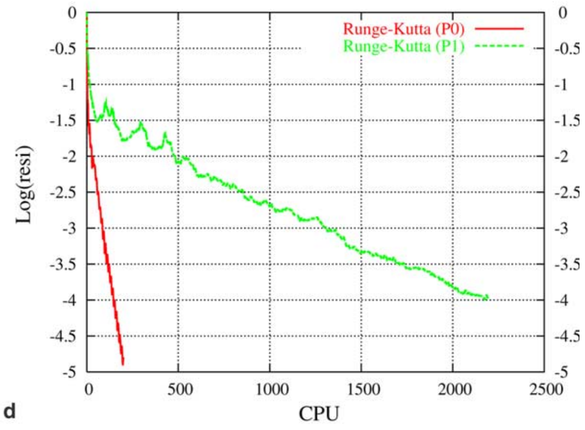
of cycles at $p = 2$ as $p = 1$, and P_2 computation actually requires less cpu time than the P_1 computation. For this case, the *p*-multigrid method is about 10 times faster for P_1 computation and 18 times faster for P_2 computation than its 3 stage TVDRK counterpart.


a


b



c



d

Fig. 3. (a) Unstructured mesh used for computing subsonic flow past a NACA0012 airfoil (nelem = 6891, npoin = 3537, nboun = 183). (b) Computed pressure contours for subsonic flow past a NACA0012 airfoil at $M_\infty = 0.675$, $\alpha = 2°$. (c) Convergence history versus time steps for subsonic flow past NACA0012 airfoil using 3-stage TVD Runge–Kutta explicit method between P_0-element approximation and P_1-element approximation. (d) Convergence history versus CPU time for subsonic flow past NACA0012 airfoil using 3-stage TVD Runge–Kutta explicit method between P_0-element approximation and P_1-element approximation. (e) Convergence history versus time steps for subsonic flow past NACA0012 airfoil for P_0-element approximation among 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. (f) Convergence history versus CPU time for subsonic flow past NACA0012 airfoil for P_0-element approximation among 3-stage TVD Runge–Kutta explicit method, matrix-free LU-SGS method, and matrix-free SGS method with 10 iterations. (g) Convergence history versus time steps for subsonic flow past NACA0012 airfoil for P_1-element approximation among 3-stage TVD Runge–Kutta explicit method, 3-stage TVD Runge–Kutta *p*-multigrid method, matrix-free LU-SGS *p*-multigrid method, and matrix-free SGS-10 *p*-multigrid method. (h) Convergence history versus CPU time for subsonic flow past NACA0012 airfoil for P_1-element approximation among 3-stage TVD Runge–Kutta explicit method, 3-stage TVD Runge–Kutta *p*-multigrid method, matrix-free LU-SGS *p*-multigrid method, and matrix-free SGS-10 *p*-multigrid method. (i) Convergence history versus time steps for subsonic flow past NACA0012 airfoil using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 *p*-multigrid method. (j) Convergence history versus CPU time for subsonic flow past NACA0012 airfoil using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 *p*-multigrid method.

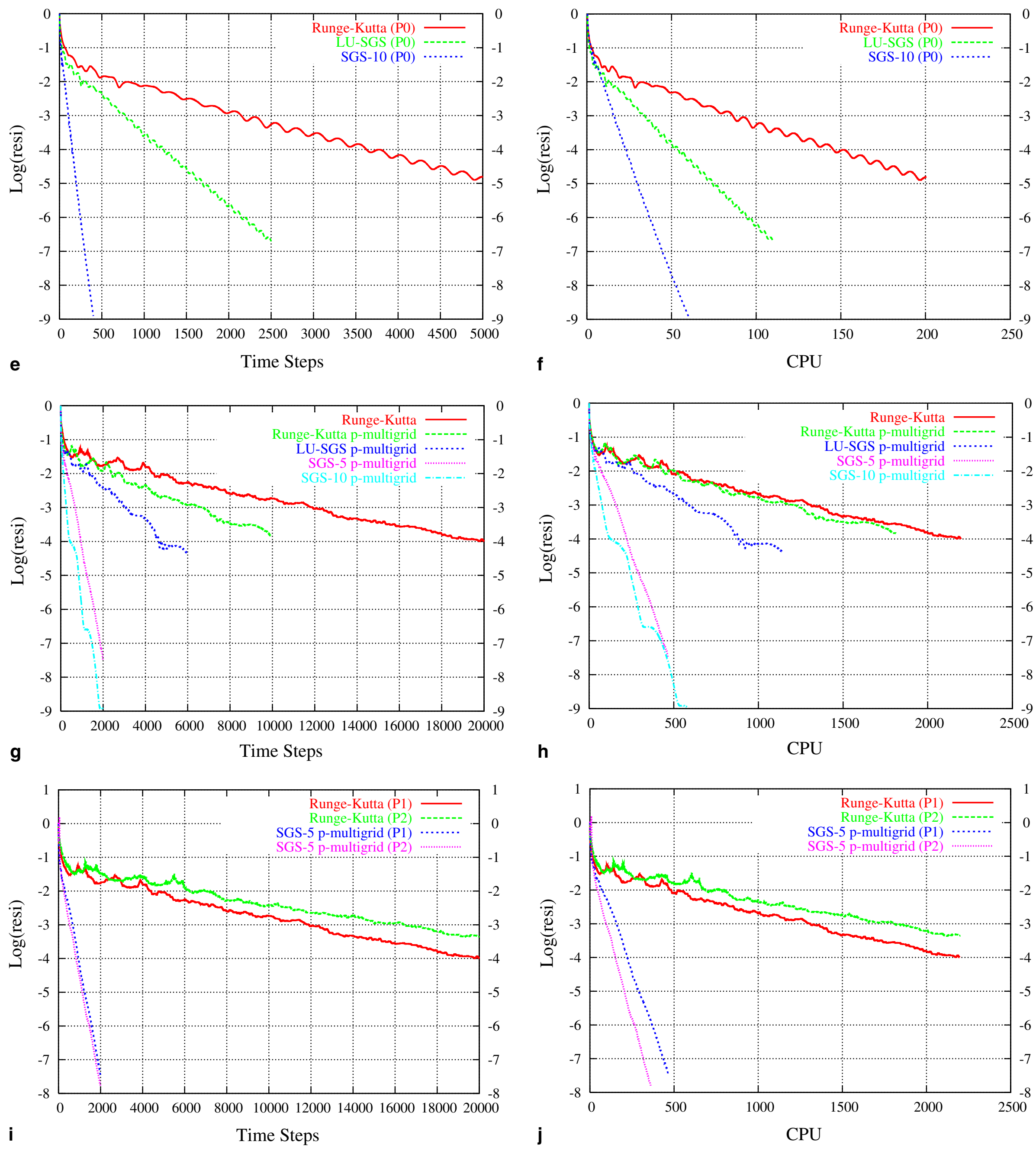


Fig. 3 (*continued*)

5.4. Subsonic flow past a RAE2822 airfoil

The last example is the subsonic flow past a RAE2822 airfoil at a Mach number of 0.63, and an angle of attack 0°. The fine mesh used for P_1 computation, which contains 2681 grid points, 5163 elements, and 199

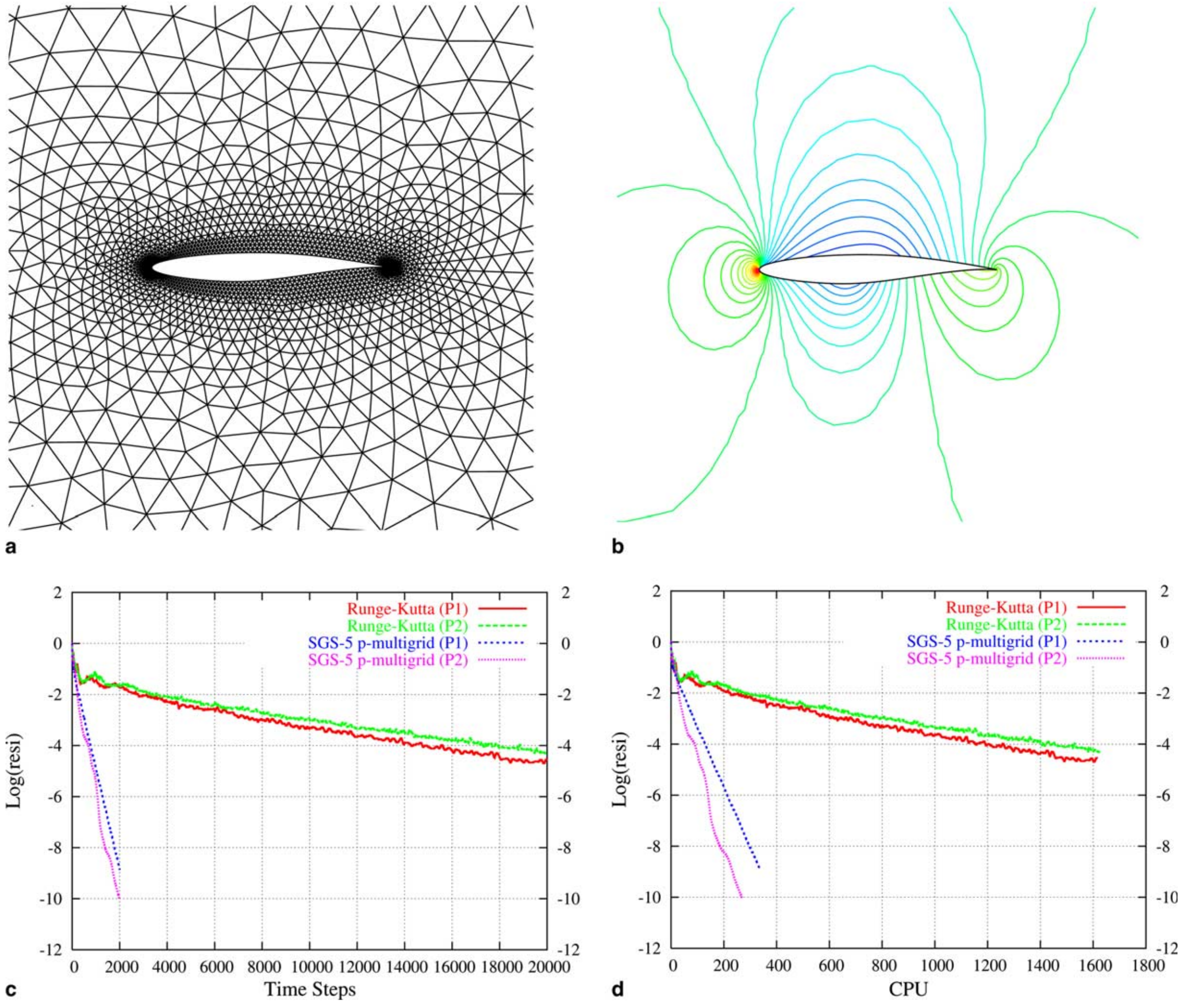


Fig. 4. (a) Unstructured mesh used for computing subsonic flow past a RAE2822 airfoil (nelem = 5163, npoin = 2681, nboun = 199). (b) Computed pressure contours for subsonic flow past a RAE2822 airfoil at $M_\infty = 0.675$, $\alpha = 0°$. (c) Convergence history versus time steps for subsonic flow past a RAE2822 airfoil using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 *p*-multigrid method. (d) Convergence history versus CPU time for subsonic flow past a RAE2822 airfoil using the same number of degrees of freedom for P_1- and P_2-element approximations between 3-stage TVD Runge–Kutta explicit method and matrix-free SGS-5 *p*-multigrid method.

boundary points, is depicted in Fig. 4(a). The coarse mesh used for P_2 computations, chosen to have about the same number of degrees of freedom for P_1 and P_2 computation, consists of 2582 elements, 1360 grid points, and 138 boundary points. The computed pressure contours obtained using P_1-element approximation in the flow field are shown in Fig. 4(b). Fig. 4(c) and (d), shows a comparison of convergence histories versus time steps and CPU time, respectively, for the P_1 and P_2 computation between the three-stage TVD Runge–Kutta explicit method and *p*-multigrid method with the matrix-free SGS(5) smoother. The trends are almost exactly the same as for the last two test cases: the convergence rate for the *p*-multigrid method is insensitive to the polynomial order, and P_2 computation actually requires less cpu time than the P_1 computation. For this case, the *p*-multigrid method converges about 11 times faster for P_1 computation and 16 times faster for P_2 computation than the explicit method.

6. Conclusions

A *p*-multigrid discontinuous Galerkin finite element method has been developed for solving the compressible Euler equations on unstructured grids. It is found that 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 due to the severely anisotropic nature of the Euler equations, and the isotropic nature of *p*-multigrid iterations. Implicit schemes as an iterative smoother provide much better convergence properties at the expense of significant increase in storage requirements. This observation has led us to use different iterative smoothers at different approximation levels in an attempt to develop an efficient method to accelerate the convergence of the Euler equations to a steady state without significant increase in memory requirement. The developed *p*-multigrid has been used to compute compressible flows for a variety of test problems on unstructured grids. The numerical results obtained strongly indicate the order independent property of this novel *p*-multigrid method. This makes our *p*-multigrid method especially significant and attractive for DG(2) method, as a third-order solution actually requires less CPU time than a second-order solution for the same number of degrees of freedom. The overall performance of this novel *p*-multigrid is one order of magnitude better than the explicit method without significant increase in memory. Using this acceleration method, the discontinuous Galerkin methods provide a viable and competitive alternative to the traditional finite-volume, finite-element, and finite-difference methods. The extension of the present *p*-multigrid method to other types of elements and three-dimensional problems appears to be straightforward. Future work will also explore application of this method for the solution of the Navier–Stokes equations.

References

[1] B. Cockburn, S. Hou, C.W. Shu, TVD Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, Mathematics of Computation 55 (1990) 545–581.

[2] S.Y. Lin, Y.S. Chin, Discontinuous Galerkin finite element method for Euler and Navier–Stokes equations, AIAA Journal 31 (1993) 2016–2023.

[3] R. Biswas, K.D. Devine, J. Flaherty, Parallel, adaptive finite element methods for conservation laws, Applied Numerical Mathematics 14 (1994) 255–284.

[4] K.S. Bey, J.T. Oden, A. Patra, A parallel *hp*-adaptive discontinuous Galerkin method for hyperbolic conservation laws, Applied Numerical Mathematics 20 (1996) 321–336.

[5] H.L. Atkins, C.W. Shu, Quadrature free implementation of discontinuous Galerkin method for hyperbolic equations, AIAA Journal 36 (5) (1998) 775–782.

[6] D.P. Lockard, H.L. Atkins, Efficient implementations of the quadrature free discontinuous Galerkin method, AIAA Paper, AIAA-99-3309, 1999.

[7] B. Cockburn, C.W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional system, Journal of Computational Physics 141 (1998) 199–224.

[8] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D euler equations, Journal of Computational Physics 138 (1997) 251–285.

[9] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, Journal of Computational Physics 131 (1997) 267–279.

[10] J.J.W. van der Vegt, H. van der Ven, Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows, Journal of Computational Physics 141 (1998) 46–77.

[11] M. Remaki, W.G. Habashi, D. Ait-Ali-Yahia, A. Jay, A 3D discontinuous Galerkin method for multiple pure tone noise problems, AIAA Paper, AIAA-2002-0229, 2002.

[12] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations, in: B. Cockburn, G.E. Karniadakis, C.W. Shu (Eds.), Discontinuous Galerkin Methods, Theory, Computation, and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer-Verlag, New York, 2000, pp. 197–208.

[13] P. Rasetarinera, M.Y. Hussaini, An efficient implicit discontinuous spectral Galerkin method, Journal of Computational Physics 172 (2001) 718–738.

[14] E.M. Rönquist, A.T. Patera, Spectral element multigrid, I. Formulation and numerical results, Journal of Scientific Computing 2 (4) (1987) 389–406.

[15] Y. Maday, R. Munoz, Spectral element multigrid, Part 2: Theoretical justification, Tech. Rep. 88-73, ICASE, 1988.

[16] F. Bassi, S. Rebay, Numerical solution of the euler equations with a multiorder discontinuous finite element method, in: Proceedings of the Second International Conference on Computational Fluid Dynamics, Sydney, Australia, 15–19 July 2002.

[17] B.T. Helenbrook, D. Mavriplis, H.L. Atkins, Analysis of *p*-multigrid for continuous and discontinuous finite element discretizations, AIAA Paper 2003-3989, 2003.

[18] H. Luo, D. Sharov, J.D. Baum, R. Löhner, A class of matrix-free implicit methods for compressible flows on unstructured grids, in: Proceedings of the First International Conference on Computational Fluid Dynamics, Kyoto, Japan, 10–14 July 2000.

[19] H. Luo, J.D. Baum, R. Löhner, A fast, matrix-free implicit method for compressible flows on unstructured grids, Journal of Computational Physics 146 (2) (1998) 664–690.

[20] E.F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1994) 25–34.

[21] P. Batten, M.A. Leschziner, U.C. Goldberg, Average-state Jacobians and implicit methods for compressible viscous and turbulent flows, Journal of Computational Physics 137 (1997) 38–78.

[22] H. Luo, J.D. Baum, R. Löhner, High-Reynolds number viscous flow computations using an unstructured-grid method, Journal of Aircraft 42 (2) (2005) 483–492.

[23] A. Jameson, S. Yoon, Lower–upper implicit schemes with multiples grids for the euler equations, AIAA Journal 25 (7) (1987) 929–935.