Journal of Computational Physics **172,** 718–738 (2001)
doi:10.1006/jcph.2001.6853, available online at http://www.idealibrary.com on IDEAL®

An Efficient Implicit Discontinuous Spectral Galerkin Method

Patrick Rasetarinera and M. Y. Hussaini

School of Computational Science and Information Technology, The Florida State University, Tallahassee, Florida 32306-4120
E-mail: rasetari@csit.fsu.edu; myh@csit.fsu.edu

Received August 14, 2000; revised May 25, 2001

The present paper discusses an implicit discontinuous spectral Galerkin method for the solution of the compressible Euler equations. A matrix-free Newton–Krylov–Schwarz algorithm with one-level and two-level nonoverlapping Schwarz preconditioners is used to solve the implicit systems. The study shows that this method is a factor of 50 faster than an explicit method that employs local time-stepping to accelerate convergence to steady-state solution. Procedures using LU-SGS preconditioner appear to provide the best performance. The two-level procedure is found necessary for relatively fast convergence in the case of large numbers of mesh elements.

Key Words: discontinuous Galerkin method; implicit method; matrix-free; Newton–Krylov–Schwarz.

1. INTRODUCTION

The discontinuous Galerkin method has recently become popular for the solution of systems of conservation laws. In its original formulation for the discretization of the neutron transport equation resulting from Reed and Hill [19], the solution is computed element by element. This is not obviously possible for nonlinear problems, and these problems can be treated by a discontinuous space-time discretization (leading to a global system of nonlinear algebraic equations); see Bar–Yoseph [1] and Bar–Yoseph and Elata [2]. For computational ease, a frequently adopted approach employs explicit time discretization. The so-called Runge–Kutta discontinuous Galerkin (RKDG) method introduced by Cockburn and Shu [6] uses a total variation diminishing (TVD) Runge–Kutta scheme developed by Shu and Osher [21]. The time step is then limited by a (linear) stability condition that depends on the order of the Runge–Kutta method and on the order of the spatial discretization. The RKDG method is easy to implement and has been successfully applied to a wide range of unsteady problems [8, 11, 17, 18]. For steady-state computation, a common procedure uses a local

0021-9991/01 $35.00

time-stepping technique to accelerate convergence. However, the maximum time step is still limited by a local stability condition. Thus, the convergence may become dramatically slow for large-scale simulations.

Implicit solvers, which do allow large time steps, are widely used in the computational fluid dynamics community for the steady solution of nonlinear conservation laws [10]. Using finite-volume or finite-element discretizations, these solvers generally rely on some linearization of a nonlinear operator, which leads to a Newton-like method. Iterative methods and approximate factorization methods are then used to solve the linear system.

The Newton–Krylov–Schwarz method has recently emerged as a promising technique for the parallel implicit solution of large-scale aerodynamics problems [13]. It combines the Newton–Krylov method as the nonlinear solver and the Krylov–Schwarz iterative method for the solution of the linear system arising from the Newton linearization. The Krylov–Schwarz method has become popular especially in parallel computing because of its locality. It is specially well suited for the discontinuous spectral Galerkin method, since each subdomain can be treated separately.

In [3] Bassi and Rebay show the efficiency of the generalized minimum residual iterative method (GMRES) [20] using a simple block Jacobi preconditioner for the implicit solution of the compressible Navier–Stokes equations. In this work, we propose a "matrix-free" Newton–Krylov–Schwarz algorithm for the implicit discretization of the discontinuous Galerkin method. In the "matrix-free" approach, the Jacobian-vector product within the Krylov algorithm is approximated by the Frechet derivative [4]. This permits considerable saving in storage. One-level and two-level, additive and multiplicative nonoverlapping Schwarz preconditioners are implemented and compared.

2. DISCONTINUOUS GALERKIN METHOD

Consider a conservation equation for a quantity $\mathbf{u}$ in a two-dimensional region $\mathcal{D}$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{u}) = 0, \tag{1}$$

where $\mathbf{F} = (F, G)^t$ is a flux vector. Let the domain $\mathcal{D}$ be partitioned into $\mathcal{N}_e$ nonoverlapping subdomains, or elements, $\mathcal{D}_i$. The discontinuous Galerkin method is a finite-element method in which the approximation space, V_h, may be discontinuous across element interfaces. In the semi-discrete formulation, V_h contains only spatial functions

$$V_h = \{v \in L^1(\mathcal{D}) : v\,|\,\mathcal{D}_i \in \mathcal{P}(\mathcal{D}_i),\ i = 1, \ldots, \mathcal{N}_e\},$$

where $\mathcal{P}(\mathcal{D}_i)$ is a polynomial space defined on $\mathcal{D}_i$. The degrees of freedom of the solution are then obtained by solving a weak formulation of (1).

Let $\mathcal{B}_i = \{v_\ell^i\}_{\ell=0,\ldots,N-1}$ be a local basis set such that

$$Span(\mathcal{B}_i) = \mathcal{P}(\mathcal{D}_i), \quad Supp\big(v_\ell^i\big) = \mathcal{D}_i, \quad \ell = 0, \ldots, N-1.$$

Then the approximate solution $\mathbf{u}_h$ satisfies

$$\int_{\mathcal{D}_i} \left(\frac{\partial \mathbf{u}_h}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{u}_h) \right) \mathbf{v}\, d\mathbf{x} = 0, \quad \forall \mathbf{v} \in \mathcal{B}_i \tag{2}$$

in each element $\mathcal{D}_i$. Using Green's formula, Eq. (2) is recast as

$$\int_{\mathcal{D}_i} \frac{\partial \mathbf{u}_h}{\partial t}\mathbf{v} - \mathbf{F}(\mathbf{u}_h)\cdot\nabla\mathbf{v}\, d\mathbf{x} + \sum_j \int_{\partial\mathcal{D}_{ij}} \mathbf{F}(\mathbf{u}_h)\cdot\mathbf{n}\,\mathbf{v}\, ds = 0, \quad \forall\mathbf{v}\in\mathcal{B}_i\,, \tag{3}$$

where $\partial\mathcal{D}_i$ is the boundary of $\mathcal{D}_i$ and $\mathbf{n}$ denotes the unit outward normal vector. Since the data is discontinuous across the interface of contiguous domains, two values of $\mathbf{u}_h$ ($\mathbf{u}_h^i$ inside $\mathcal{D}_i$ and $\mathbf{u}_h^i$ outside $\mathcal{D}_i$) are available at the interface. A numerical flux $\mathbf{F}_{num}$ is then used to evaluate the interface flux in the last integral of the Eq. (3)

$$\mathbf{F}(\mathbf{u}_h)\cdot\mathbf{n}|_{\partial\mathcal{D}_i} = \mathbf{F}_{num}\left(\mathbf{u}_h^i, \mathbf{u}_h^j, \mathbf{n}\right). \tag{4}$$

3. IMPLICIT TIME DISCRETIZATION

We discuss here an implicit algorithm based on the backward Euler time integration scheme. Although it is unconditionally stable, it is time-accurate only if its time step resolves the temporal scales of the problem. The temporal accuracy requirements in most aerodynamic flows are far less stringent than the stability limit of most popular time discretization schemes. In such cases, the present algorithm becomes highly competitive in that it permits relatively large time steps although constrained by the characteristic time scale of the problem. This advantage is lost in the case of direct numerical simulation of transitional and turbulent flows where the constraints of time scales that need to be resolved are more stringent than the stability limit. This advantage can be exploited to the utmost in the case of steady flows, which is the focus of the present work. A naive way to find a steady-state solution is to compute a time-accurate process to the point where all transient effects have disppeared, assuming of course they do. An expedient way is to compute a time-like process wherein the time path is not necessarily physical. Iterative or relaxation techniques to solve the set of coupled equations resulting from the time-independent terms and boundary conditions may be identified with this fictitious time scheme [14].

In this section, the backward Euler time integration is applied to (3) in the element $\mathcal{D}_i$ wherein the solution is represented by $\mathbf{u}_i$,

$$\begin{aligned}\int_{\mathcal{D}_i} \frac{\delta\mathbf{u}_i^n}{\delta t}\mathbf{v}_l^i\, dx - \int_{\mathcal{D}_i} \mathbf{F}\left(\mathbf{u}_i^n + \delta\mathbf{u}_i^n\right)\cdot\nabla\mathbf{v}_l^i\, dx \\ + \sum_j \int_{\partial\mathcal{D}_{ij}} \mathbf{F}_{num}\left(\mathbf{u}_i^n + \delta\mathbf{u}_i^n, \mathbf{u}_j^n + \delta\mathbf{u}_j^n, \mathbf{n}\right)\mathbf{v}_l^i\, d\sigma = 0, \quad l = 0,\ldots,N-1,\end{aligned} \tag{5}$$

where the superscript n indicates the time

$$\delta t = t^{n+1} - t^n \quad \text{and} \quad \delta\mathbf{u}_i^n = \mathbf{u}_i^{n+1} - \mathbf{u}_i^n. \tag{6}$$

Setting

$$\mathbf{F}(\mathbf{u}_i + \delta\mathbf{u}_i) = \mathbf{F}(\mathbf{u}_i) + \mathbf{A}(\mathbf{u}_i)\delta\mathbf{u}_i + \mathcal{O}\left(\delta\mathbf{u}_i^2\right)$$

and

$$\mathbf{F}_{num}(\mathbf{u}_i + \delta\mathbf{u}_i, \mathbf{u}_j + \delta\mathbf{u}_j, \mathbf{n}) = \mathbf{F}_{num}(\mathbf{u}_i, \mathbf{u}_j, \mathbf{n}) + \mathbf{A}_{ij}^1\delta\mathbf{u}_i + \mathbf{A}_{ij}^2\delta\mathbf{u}_j + \mathcal{O}(\delta^2)$$

with

$$\mathbf{A} = \frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}}, \quad \mathbf{A}^1 = \frac{\partial \mathbf{F}_{num}(\mathbf{u}, \mathbf{v}, \mathbf{n})}{\partial \mathbf{u}}, \quad \mathbf{A}^2 = \frac{\partial \mathbf{F}_{num}(\mathbf{u}, \mathbf{v}, \mathbf{n})}{\partial \mathbf{v}},$$
$$\mathbf{A}^1_{ij} = \mathbf{A}^1(\mathbf{u}_i, \mathbf{u}_j, \mathbf{n}), \quad \mathbf{A}^2_{ij} = \mathbf{A}^2(\mathbf{u}_i, \mathbf{u}_j, \mathbf{n}),$$

then dropping terms of second and higher order, (5) becomes

$$\int_{\mathcal{D}_i} \frac{\delta \mathbf{u}_i^n}{\delta t} \mathbf{v}_l^i \, dx - \int_{\mathcal{D}_i} \mathbf{A}\left(\mathbf{u}_i^n\right) \cdot \delta \mathbf{u}_i^n \cdot \nabla \mathbf{v}_l^i \, dx + \sum_j \int_{\partial \mathcal{D}_{ij}} \left(\mathbf{A}^1_{ij} \delta \mathbf{u}_i^n + \mathbf{A}^2_{ij} \delta \mathbf{u}_j^n\right) \mathbf{v}_l^i \, d\sigma = R_{i,l}(\mathbf{u}^n), \quad l = 0, \ldots, N-1, \tag{7}$$

where $R_{i,l}$ is the residual

$$R_{i,l}(\mathbf{u}^n) = \int_{\mathcal{D}_i} \mathbf{F}\left(\mathbf{u}_i^n\right) \cdot \nabla \mathbf{v}_l^i \, dx - \sum_j \int_{\partial \mathcal{D}_{ij}} \mathbf{F}_{num}\left(\mathbf{u}_i^n, \mathbf{u}_j^n, \mathbf{n}\right) \mathbf{v}_l^i \, d\sigma.$$

Now, let $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_{\mathcal{N}_e})$ with $\hat{\mathbf{u}}_i = (\hat{\mathbf{u}}_{i,0}, \ldots, \hat{\mathbf{u}}_{i,N-1})$ be the global vector of expansion coefficients for $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_{\mathcal{N}_e})$ such that

$$\mathbf{u}_i(\mathbf{x}) = \sum_{l=0}^{N-1} \hat{\mathbf{u}}_{i,l} \mathbf{v}_l^i(\mathbf{x}), \quad i = 1, \ldots, \mathcal{N}_e.$$

Then the scheme (7) can be written in the equivalent matrix form

$$M(\mathbf{u}^n) \delta \hat{\mathbf{u}}^n = R(\mathbf{u}^n) \tag{8}$$

with

$$M(\mathbf{u}^n) = \frac{\gamma}{\delta t} - \frac{\partial R(\mathbf{u}^n)}{\partial \hat{\mathbf{u}}},$$

where the matrix $\gamma = diag(\gamma_1, \ldots, \gamma_{\mathcal{N}_e})$ is the block diagonal mass matrix with

$$[\gamma_i]_{kl} = \int_{\mathcal{D}_i} \mathbf{v}_k^i \mathbf{v}_l^i \, dx$$

and $R(\mathbf{u})$ is the global vector of the residual

$$R(\mathbf{u}) = (R_1(\mathbf{u}), \ldots, R_{\mathcal{N}_e}(\mathbf{u})), \quad R_i(\mathbf{u}) = (R_{i,0}(\mathbf{u}), \ldots, R_{i,N-1}(\mathbf{u})).$$

In steady-state computations, the left-hand side of (8) vanishes as $\delta \hat{\mathbf{u}}$ tends to zero. The spatial accuracy of the solution depends solely on the discretization of the residual R. In the limit $\delta t \to \infty$ the scheme represented by (6), (8) reduces to a Newton iteration. To use a large time step and attain quadratic convergence, the spatial discretization of the left-hand side must be consistent with the discretization of the right-hand side.

4. KRYLOV–SCHWARZ

We now address the solution of the linear system (8) that results from the Newton-like algorithm. The nonsymmetric nature of the large sparse matrix M suggests the use of a Krylov subspace based algorithm to solve (8). The key advantage of this method is that information about the Jacobian matrix needs to be accessed only in the form of matrix–vector products. As these products can be approximated using finite differences, the algorithm can be implemented without forming the Jacobian matrix explicitly [4]:

$$\frac{\partial R(\mathbf{u})}{\partial \mathbf{u}}\hat{v} = \frac{R(\mathbf{u}+hv) - R(\mathbf{u})}{h}. \tag{9}$$

This implementation, termed "matrix-free," yields a considerable saving in storage compared to the standard implementation.

It has been implicitly assumed in the previous section that the numerical flux F_{num} is continuously differentiable. However, one can use the formula (9) to handle a continuous but nondifferentiable flux such as the Roe flux, in which case it is found that the quadratic convergence of the Newton method is lost.

The Krylov method employed in this work is the generalized minimum residual method (GMRES) [20]. The parameter h in (9) is computed following Brown and Saad [4] via

$$h = \frac{\sqrt{\epsilon}}{\|v\|_2^2}\, max\{\mathbf{u}_{typ}\|v\|_1, |\mathbf{u}^T v|\} sign(\mathbf{u}^T v),$$

where ϵ is the machine epsilon and $\mathbf{u}_{typ} > 0$ is a typical size of the components of $\mathbf{u}$ provided by the user. By default we set $\mathbf{u}_{typ} = 10^{-8}$ in our computations.

4.1. Preconditioning

The matrix M is generally ill conditioned when δt is large. When a polynomial basis of degree at most n_p is used in V_h, then the condition number of the advection operator on a single element grows like ${n_p}^2$ [5]. The use of a preconditioner P is thus necessary. Instead of (8), we solve the system

$$P^{-1}M(\mathbf{u})\delta\hat{\mathbf{u}} = P^{-1}R(\mathbf{u}), \tag{10}$$

where P is a nonsingular matrix.

Single-Level Preconditioner

We have implemented two preconditioners for the discontinuous spectral Galerkin method. They are single-level preconditioners. The first corresponds to a block-diagonal preconditioner and the second is related to LU-symmetric Gauss–Seidel (LU-SGS) preconditioner.

Let us write the matrix M as $M = L + D + U$ where L is a block strictly lower triangular matrix, U is a block strictly upper triangular matrix, and D is a block diagonal matrix, each with $N \times N$ square blocks. Each block i of D represents the contributions from the element $\mathcal{D}_i$. The matrix vector product $U\delta\hat{\mathbf{u}} = ([U\delta\hat{\mathbf{u}}]_1, \ldots, [U\delta\hat{\mathbf{u}}]_{\mathcal{N}_e})$,

$L\delta\hat{\mathbf{u}} = ([L\delta\hat{\mathbf{u}}]_1, \ldots, [L\delta\hat{\mathbf{u}}]_{\mathcal{N}_e})$ and $D\delta\hat{\mathbf{u}} = ([D\delta\hat{\mathbf{u}}]_1, \ldots, [D\delta\hat{\mathbf{u}}]_{\mathcal{N}_e})$ can be computed as

$$[U\delta\hat{\mathbf{u}}]_i = \left\{ \sum_{j>i} \int_{\partial\mathcal{D}_{ij}} \mathbf{v}_l^i \mathbf{A}_{ij}^2 \delta\mathbf{u}_j \, d\sigma \right\}_{l=0,\ldots,N-1} \tag{11}$$

$$[L\delta\hat{\mathbf{u}}]_i = \left\{ \sum_{j<i} \int_{\partial\mathcal{D}_{ij}} \mathbf{v}_l^i \mathbf{A}_{ij}^2 \delta\mathbf{u}_j \, d\sigma \right\}_{l=0,\ldots,N-1} \tag{12}$$

$$[D\delta\hat{\mathbf{u}}]_i = \left\{ \int_{\mathcal{D}_i} \frac{\delta\mathbf{u}_i}{\delta t} \mathbf{v}_l^i \, dx - \int_{\mathcal{D}_i} \mathbf{A}\left(\mathbf{u}_i^n\right) \cdot \delta\mathbf{u}_i \cdot \nabla\mathbf{v}_l^i \, dx + \sum_j \int_{\partial\mathcal{D}_{ij}} \mathbf{v}_l^i \mathbf{A}_{ij}^1 \delta\mathbf{u}_i \, d\sigma \right\}_{l=0,\ldots,N-1.} \tag{13}$$

A simple choice is

$$P = D. \tag{14}$$

It corresponds to a block-diagonal preconditioner or a block-Jacobi preconditoner.

Another preconditioner [15]

$$P = (D + L)D^{-1}(D + U) \tag{15}$$

is the so called LU-symmetric Gauss–Seidel (LU-SGS) introduced by Jameson and Yoon [12] for the implicit resolution of the Euler equation on structured meshes and extended to unstructured meshes in [15]. The implementation of this preconditioner in the case of the discontinuous Galerkin method is straightforward using Eqs. (11)–(13). Indeed, the GMRES algorithm applied to (10) involves the solution of systems of the type

$$P\hat{\mathbf{y}} = \hat{\mathbf{x}},$$

which can be decomposed in two steps:

Forward sweep:

$$(D + L)\hat{\mathbf{y}}^* = \hat{\mathbf{x}}$$

Backward sweep:

$$(D + U)\hat{\mathbf{y}} = D\hat{\mathbf{y}}^*.$$

Note that there is no need to store U and L since the forward and backward sweep can be expressed as

$$\hat{\mathbf{y}}_i^* = D_i^{-1}(\hat{\mathbf{x}}_i - [L\hat{\mathbf{y}}^*]_i), \quad i = 1, \ldots, \mathcal{N}_e$$

$$\hat{\mathbf{y}}_i = \hat{\mathbf{y}}_i^* - D_i^{-1}[U\hat{\mathbf{y}}]_i, \quad i = \mathcal{N}_e, \ldots, 1.$$

Only the block diagonal matrix D^{-1} needs then to be stored. For both preconditioners, an exact LU solver is used to compute D^{-1}. In our implementation, we have also stored the face values of $\mathbf{u}$ to facilitate the efficient computation of $\mathbf{A}^2_{ij}$ in (11) and (12).

In general, the numerical flux $\mathbf{F}_{num}$ is nonlinear and the Jacobians $\mathbf{A}^1$ and $\mathbf{A}^2$ are difficult to compute. Since explicit forms of these Jacobians are only needed in the evaluation of the preconditioner P, approximate Jacobians which are easier to calculate are used. However, one should be careful in the choice of the Jacobian $\mathbf{A}^1$ and $\mathbf{A}^2$ to avoid having a singular preconditioner P when $\delta t \to \infty$. One possibility is to take

$$\mathbf{A}^1_{ij} = \frac{\mathbf{A}(\mathbf{u}_i, \mathbf{n}) + |\mathbf{A}(\mathbf{u}_i, \mathbf{n})|}{2}, \quad \mathbf{A}^2_{ij} = \frac{\mathbf{A}(\mathbf{u}_j, \mathbf{n}) - |\mathbf{A}(\mathbf{u}_j, \mathbf{n})|}{2}, \tag{16}$$

which guarantees a nonsingular matrix D if $\mathbf{A}$ is nonsingular. Furthermore, the nonnegativity of the eigenvalues of $\mathbf{A}^1$ and the nonpositivity of the eigenvalues of $\mathbf{A}^2$ are important for the efficiency of the preconditioner (15) as mentioned in [12].

Two-Level Preconditioner

The convergence rate of the single-level preconditioned method may deteriorate when $\mathcal{N}_e$ becomes large, especially for the block Jacobi preconditioning, since information is exchanged only between neighboring elements. This deterioration may be overcome by introducing a coarse grid solver, which has a global communication among all elements.

Let V_H denote the coarse grid space

$$V_H = \left\{ v \in L^1(\mathcal{D}) : v \mid \mathcal{D}_k \equiv \textit{constant}, \quad v \mid \mathcal{D}_k^c \equiv 0, \quad k = 1, \ldots, \mathcal{N}_e \right\},$$

with I_H the interpolation operator from the coarse grid to the fine grid

$$\begin{aligned} I_H &: V_H \to V_h \\ (I_H \mathbf{u}_H)|_{\mathcal{D}_i} &= \mathbf{u}_H|_{\mathcal{D}_i} \sum_{l=0}^{N-1} \frac{\mathbf{v}_l^i}{\gamma_l} \int_{\mathcal{D}_i} \mathbf{v}_l^i \, dx, \quad \forall \mathbf{u}_H \in V_H, \end{aligned}$$

and J_H the restriction operator form the fine grid to the coarse grid

$$\begin{aligned} J_H &: V_h \to V_H \\ (J_H \mathbf{u}_h)|_{\mathcal{D}_i} &= \frac{1}{|\mathcal{D}_i|} \int_{\mathcal{D}_i} \mathbf{u}_h \, dx, \quad \forall \mathbf{u}_h \in V_h, \end{aligned}$$

with $\gamma_l = \int_{\mathcal{D}_i} (\mathbf{v}_l^i)^2 \, dx$. The coarse grid operator M_H associated with V_H is obtained by setting $\mathbf{v} = 1$ in Eq (11)–(13). In fact, M_H corresponds to the Jacobian form of a first-order finite-volume discretization of (1). Keeping the same notation, a two-level preconditioner P can be constructed as

$$P^{-1} = I_H M_H^{-1} J_H + P_1,$$

where P_1 is the single-level preconditioner (14) or (15).

5. SPACE DISCRETIZATION

Although the choice of basis functions does not affect the accuracy of the discontinuous Galerkin method, it may greatly impact the implementation and the efficiency of the algorithms described above. In particular, for a nonlinear flux, the formula (9) for the matrix vector product involves several forward transforms from the Galerkin space to the physical space at each iteration. This must be done efficiently in order to have an efficient method. In this section, we describe a collocation form of the discontinuous Galerkin method on quadrilateral elements, for which the transformation from the Galerkin space to the physical space is the identity.

To evaluate Eq. (3), it is convenient to map the element $\mathcal{D}_i$ into the reference square $\hat{\Omega} = [-1, 1] \times [-1, 1]$. Under such a mapping, Eq. (3) becomes

$$\forall \mathbf{v} \in \mathcal{B}_i, \quad \int_{-1}^{1}\int_{-1}^{1} \mathbf{v} J \frac{\partial \mathbf{u}_i}{\partial t} d\xi\, d\eta - \int_{-1}^{1}\int_{-1}^{1} \tilde{\mathbf{F}}(\mathbf{u}_i) \cdot \nabla_{\xi\eta} \mathbf{v}\, d\xi\, d\eta + \sum_{j=1}^{4} \int_{-1}^{1} \tilde{\mathbf{F}}_{num}(\mathbf{u}_i, \mathbf{u}_j, \hat{\mathbf{n}}_j)\mathbf{v}\, ds = 0, \tag{17}$$

where (ξ, η) are the local coordinates in $\hat{\Omega}$, $\hat{\mathbf{n}}_k$ is the outward normal vector of the *kth* face of $\hat{\Omega}$ and

$$\tilde{\mathbf{F}} = (\tilde{F}, \tilde{G})^t, \quad \tilde{F} = y_\eta F - x_\eta G, \quad \tilde{G} = -y_\xi F + x_\xi G, \quad J(\xi, \eta) = x_\xi y_\eta - x_\eta y_\xi.$$

To compute the integral arising in (17) the Gauss quadrature rule is used,

$$\sum_{k=0}^{n-1} f(x_k)\, \omega_k = \int_{-1}^{1} f(\xi)\, d\xi, \tag{18}$$

to replace the integral. Here, the x_k are the Legendre–Gauss collocation nodes associated with the weights ω_k. The quadrature formula (18) is exact for polynomial function f of order up to $2n - 1$. We refer to [5] for further details on the properties of the Legendre polynomials.

For elements with straight sides, the Jacobian J is linear and the outward normal vectors $\hat{\mathbf{n}}_k$ are constant along a face. The integrals in (17) are then evaluated exactly when the basis functions are polynomials of degree at most $n - 1$. According to Cockburn and Shu [7], the formal order of accuracy of the approximation scheme is then n. If the sides are curved, however, the Jacobian J is a polynomial of order greater than 2 and an additional quadrature error is incurred as the quadrature rule (18) is no longer exact.

For quadrilateral elements, a natural choice is to use tensor product basis functions. Using Lagrange interpolating polynomials $\{L_k\}_{k=0,\ldots,n-1}$ of order less than n in the ξ direction and Lagrange polynomials $\{L_l\}_{l=0,\ldots,m-1}$ of order less than m in the η direction as basis functions, the approximate solution in $\mathcal{D}_i$ reads

$$\mathbf{u}_i(\xi, \eta) = \sum_{k=0}^{n-1}\sum_{l=0}^{m-1} u_{kl} L_k(\xi) L_l(\eta),$$

where the Lagrange polynomials have nodes at the Legendre–Gauss quadrature points and

$$u_{kl} = \mathbf{u}_i(x_k, y_l).$$

With this choice of basis functions, the transformation operator from the Galerkin space to physical space is the identity and the mass matrix is diagonal

$$\int_{-1}^{1}\int_{-1}^{1} L_i(\xi)L_k(\xi)L_j(\eta)L_l(\eta)\,d\xi\,d\eta = \delta_{ik}\delta_{jl}\omega_k\omega_l,$$

where $i, k = 0, \ldots, n-1$, $j, l = 0, \ldots, m-1$.

In order to implement the method described in Section 4, it is convenient to write (17) in a matrix form. Let $\mathbf{W}$ be the mass matrix, $\mathbf{D}_\xi$ and $\mathbf{D}_\eta$ be the differentiation matrix in the ξ and η directions, respectively, $\mathbf{J}$ be the diagonal matrix whose entries are the values of the Jacobian J at the collocation points, $\mathbf{I}_{f_j}$ be the interpolation matrix on the face j, and $\mathbf{W}_j$ be the diagonal matrix whose entries are the integration weights on the face j

$$\begin{aligned}
\{\mathbf{W}\}_{i+nj,k+nl} &= \delta_{ik}\delta_{jl}\omega_k\omega_l \\
\{\mathbf{D}_\xi\}_{i+nj,k+nl} &= \delta_{jl}L_k'(\xi_i) \\
\{\mathbf{D}_\eta\}_{i+nj,k+nl} &= \delta_{ik}L_l'(\eta_j). \\
\{\mathbf{I}_{f_\alpha}\}_{i,k+nl} &= \delta_{ik}L_l\left(-1^{\alpha(\alpha-1)/2+1}\right); \quad \alpha = 1, 3 \\
\{\mathbf{I}_{f_\beta}\}_{j,k+nl} &= \delta_{jl}L_k\left(-1^{\beta(\beta-1)/2+1}\right); \quad \beta = 2, 4 \\
\{\mathbf{W}_\alpha\}_{i,k} &= \delta_{ik}\omega_k; \quad \alpha = 1, 3 \\
\{\mathbf{W}_\beta\}_{j,l} &= \delta_{jl}\omega_l; \quad \beta = 2, 4
\end{aligned}$$

with $i, k = 0, \ldots, n-1$; $j, l = 0, \ldots, m-1$. Then we can write the Eq. (17) in the matrix form

$$\mathbf{W}\mathbf{J}\frac{\partial \mathbf{u}_i}{\partial t} - \left(\mathbf{D}_\xi^t\tilde{F} + \mathbf{D}_\eta^t\tilde{G}\right)\mathbf{W}\mathbf{u}_i + \sum_{j=1}^{4}\mathbf{I}_{f_j}^t\mathbf{W}_j\tilde{\mathbf{F}}_{num_j} = 0,$$

where $\tilde{\mathbf{F}}_{num_j}$ is the vector of the numerical flux at the face j. Using the same notation, the *ith* diagonal block of the matrix D in the preconditioners (14) and (15) reads

$$D_i = \mathbf{W}\mathbf{J} - \left(\mathbf{D}_\xi^t\tilde{A} + \mathbf{D}_\eta^t\tilde{B}\right)\mathbf{W} + \sum_{j=1}^{4}\mathbf{I}_{f_j}^t\tilde{\mathbf{A}}_{ij}^1\mathbf{W}_j\mathbf{I}_{f_j}$$

with

$$\tilde{A} = \frac{\partial\tilde{F}(\mathbf{u})}{\partial\mathbf{u}}; \quad \tilde{B} = \frac{\partial\tilde{G}(\mathbf{u})}{\partial\mathbf{u}}; \quad \tilde{\mathbf{A}}_{ij}^1 = \frac{\partial\tilde{F}_{num_j}}{\partial\mathbf{u}}.$$

The obvious advantage of using Lagrange basis functions is that the expansion coefficients of the approximate solution coincide with their nodal values at the quadrature points. Therefore, the evaluation of nonlinear functions at the quadrature points is straightforward.

6. NUMERICAL RESULTS

We now present numerical results from the computation of the two-dimensional steady-state compressible inviscid flows governed by the Euler equations (in conservative form)

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0,$$

with

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad F(\mathbf{u}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \quad G(\mathbf{u}) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}.$$

In most of the numerical tests performed, we compare the implicit method to the explicit method in order to emphasize the efficiency. The explicit method uses a second-order Runge–Kutta time discretization with a local time-stepping technique. We use the Osher approximate Riemman solver as the numerical flux $\mathbf{F}_{num}$. The simple Steger–Warming splitting [22], which coincides with (16), is used as the approximate Jacobian when constructing the preconditioner P.

All computations are started with uniform flow and CFL number unity unless specified otherwise. In order to advance the solution to steady-state, the time step is updated adaptively using the "switched evolution relaxation" (SER) method [16], where the CFL number is increased in inverse proportion to the residual reduction:

$$CFL^n = Min(CFL^{n-1}, 10^6) \frac{\| R(\mathbf{u}^{n-1}) \|}{\| R(\mathbf{u}^n) \|}.$$

Since the storage requirements of GMRES increase linearly with the number of search directions in the Krylov subspace, the GMRES is terminated when the size of the Krylov subspace is equal to a parameter m. GMRES is then restarted using the most recent solution as the initial guess. This is known as the restarted GMRES algorithm. In general, m is chosen between 5 and 30 [20].

The choice of an optimal parameter m to restart GMRES depends on the problem and on the order of the solution technique. For the discontinuous spectral Galerkin method, the condition number of the preconditioned matrix degrades as the order of the method increases. In order to use a fixed parameter m that accommodates different orders of approximation in the numerical tests, we find $m = 30$ in our computations to be optimal in a sense.

The GMRES iterations are also terminated when the residual norm reaches an exit tolerance β that depends on the residual $R(\mathbf{u}^n)$ at each Newton step:

$$\beta = 0.01 \| R(\mathbf{u}^n) \|. \tag{19}$$

The Newton correction is then solved only approximately leading to the so-called "inexact Newton methods" [9].

The computations were performed on an SGI origin 200 (180 Mhz R10000 CPU with 32 kbytes of primary cache and 2Mbytes of secondary cache).

6.1. Subsonic Flow in a Channel with a Circular Bump

The first test is that of a Mach 0.2 subsonic flow over a circular bump in a channel. The computations are performed on a coarse grid containing 70 elements and on a fine grid consisting of 264 elements. The coarse grid geometry and collocation points for fifth-order spatial discretization are shown in Fig. 1, along with the contours of computed Mach number.

Figure 2 displays comparisons of the convergence histories among the implicit schemes with fifth-, sixth-, and seventh-order spatial discretization on the coarse grid. We note that the convergence behavior of the implicit method can be divided in two phases. In the first phase, the solution is far from the steady state, the time steps are small yielding a diagonally dominant matrix M. Many Newton steps are then performed during this stage but only few GMRES inner iterations are needed to satisfy the convergence criterion (19). During the second phase, the convergence is steep and the steady solution is reached within only a few Newton iterations. Unlike in the first phase, the matrix M is no longer diagonally dominant in the second phase and many inner GMRES iterations are performed to reach convergence. The performance of the implicit method depends then on the efficiency the preconditioner P. This second phase starts when the solution is near the steady state in our case when the residual is approximately 10^{-3}.

We observe in Fig. 2 that increasing the order of spatial discretizations increases the number of nonlinear Newton steps in the first phase since more iterations are needed to damp high frequencies. On the other hand, we note in Table I that the average number of the inner GMRES iterations remains approximately the same for different orders of spatial approximation on the same mesh.

It is apparent from Table I that the algorithms using block Jacobi preconditioners perform twice as many inner GMRES iterations as those using the block LU-SGS preconditioners. Thus, the LU-SGS preconditioner is more efficient than the Jacobi preconditioner. We note also in Table I that only a small improvement is obtained in the average number of GMRES iterations when applying the two-level preconditioners. As a result the performance of the Newton–Krylov–Schwarz algorithm on the coarse mesh with the one-level preconditioners is better than with the two-level preconditioners.

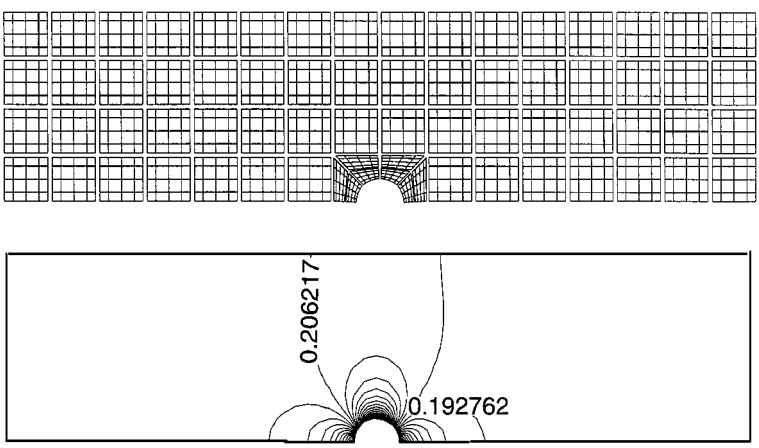

FIG. 1. 70-element mesh with fifth-order spatial discretization (top) and Mach number contours (bottom).

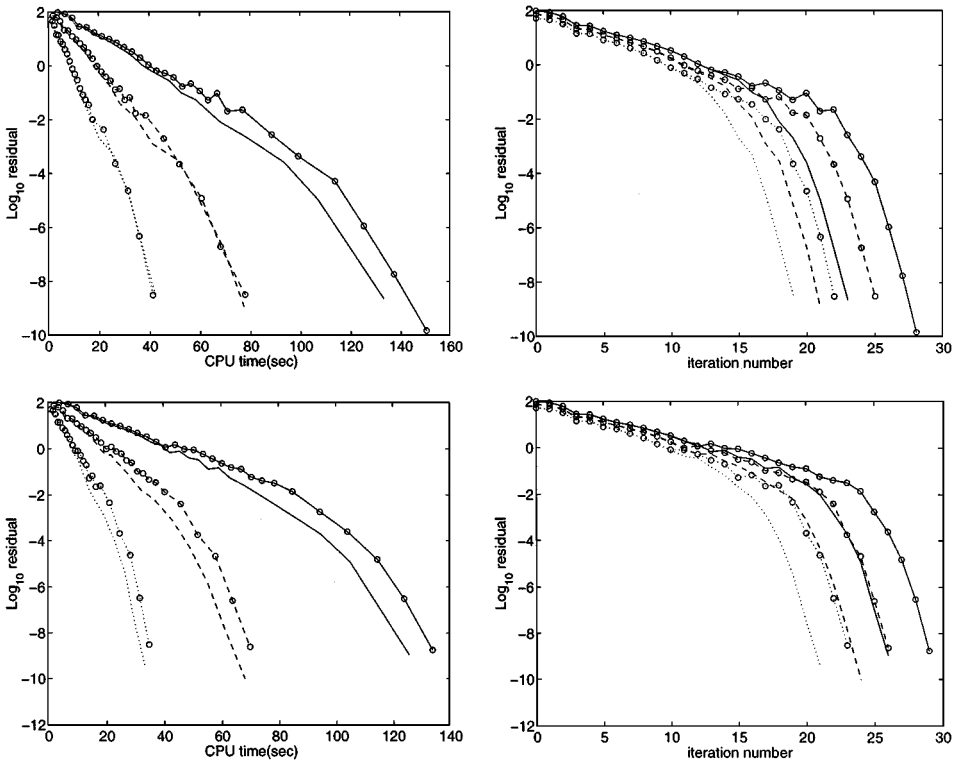

FIG. 2. Convergence history on the coarse mesh (70 elements) with the one-level and two-level block Jacobi preconditioners (top) and the one-level and two-level block LU-SGS preconditioners (bottom). Seventh-order (——— one-level, —○— two-level), sixth-order (----- one-level, --○-- two-level), fifth-order (····· one-level, ···○··· two-level). Log residual vs. CPU time (left), Log residual vs. Outer Newton iteration number (right).

The computation on the fine mesh has been performed with third- and fourth-order spatial discretizations. Table I shows over 37% improvements in the average number of inner GMRES iterations when using the two-level block Jacobi preconditioner. This led to a faster convergence in CPU time for the algorithm using the two-level method as shown in Fig. 3. For the block LU-SGS preconditioner, the improvement in the average number of GMRES iterations with the two-level method is modest.

TABLE I

Average Number of Inner GMRES Iterations for the Subsonic Channel Flow Using One-level and Two-level Block Jacobi and Block LU-SGS Preconditioners on the Coarse (70 Elements) and the Fine (264 Elements) Grids

	BJ1	BJ2	GS1	GS2
Order 3 (264 elements)	44.7000	26.5417	15.8571	11.9167
Order 4 (264 elements)	39.6818	24.7407	15.9583	11.7778
Order 5 (70 elements)	23.7000	17.3043	9.1364	7.8750
Order 6 (70 elements)	22.4545	17.4615	8.6800	7.6667
Order 7 (70 elements)	20.6250	17.3103	7.6667	7.2333

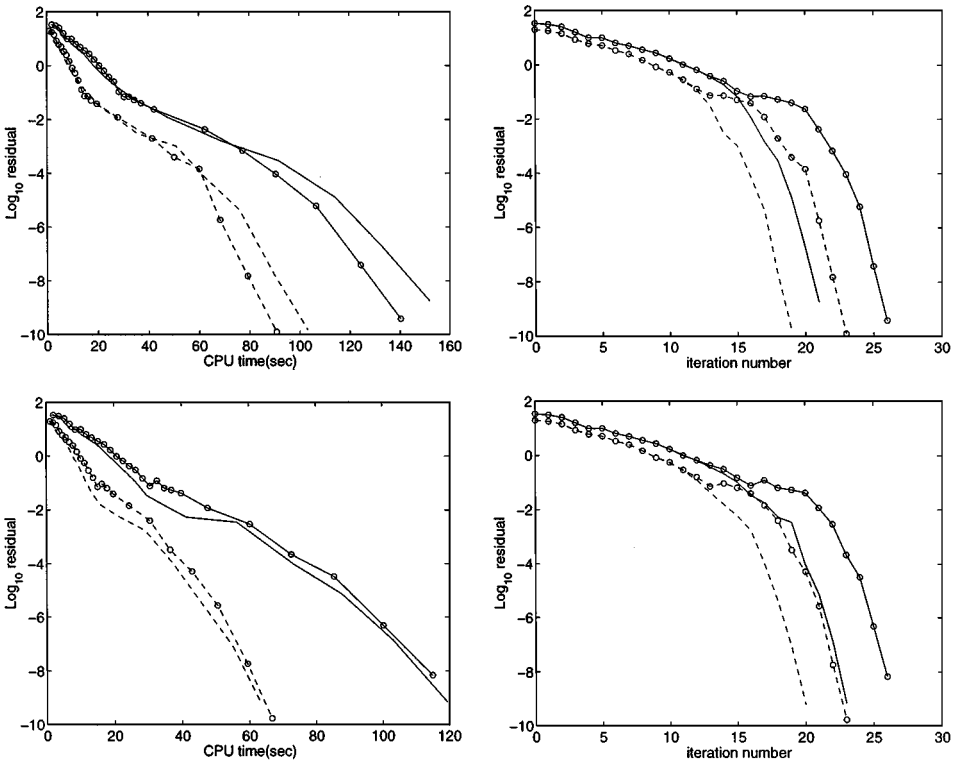

FIG. 3. Convergence history on the fine mesh (264 elements) with the one-level and two-level block Jacobi preconditioners (top) and the one-level and two-level block LU-SGS preconditioners (bottom). Fourth-order (— one-level, —o— two-level), third-order (- - - one-level, - - o - - two-level). Log residual vs. CPU time (left), Log residual vs. outer Newton iteration number (right).

Finally, we present in Fig. 4 a comparison of the convergence histories between the implicit and the explicit algorithms on the fine and the coarse meshes. The implicit solvers converge about 50 times faster than the explicit method.

6.2. Transonic Flow in a Convergent-Divergent Nozzle

The isentropic flow in a convergent-divergent nozzle is computed as a transonic test case. The geometry and the mesh are presented in Fig. 5. The nozzle consists of a converging section with a half angle of 45° and a diverging section with a half angle of 15°. The corresponding quasi-one-dimensional nozzle solution is used as initial condition.

First, the solution is computed on a mesh containing 132 elements (see Fig. 5) with third-, fourth-, and fifth-order spatial discretizations. Figure 2 illustrates the convergence history of the implicit method. As in the previous test case, the one-level block LU-SGS preconditioner is more efficient than the two-level one. The improvement in the average number of GMRES iterations is negligible when using the two-level block LU-SGS preconditioner. For the block Jacobi preconditioners, the use of the two-level preconditioner led to 30% improvement for the fifth-order scheme and to more than 40% improvement for the third and fourth in the average number of inner GMRES iterations. This results in a faster convergence rate in CPU time for the two-level method. Again, the results indicate that increasing the order of spatial discretization increases the number of outer Newton iterations

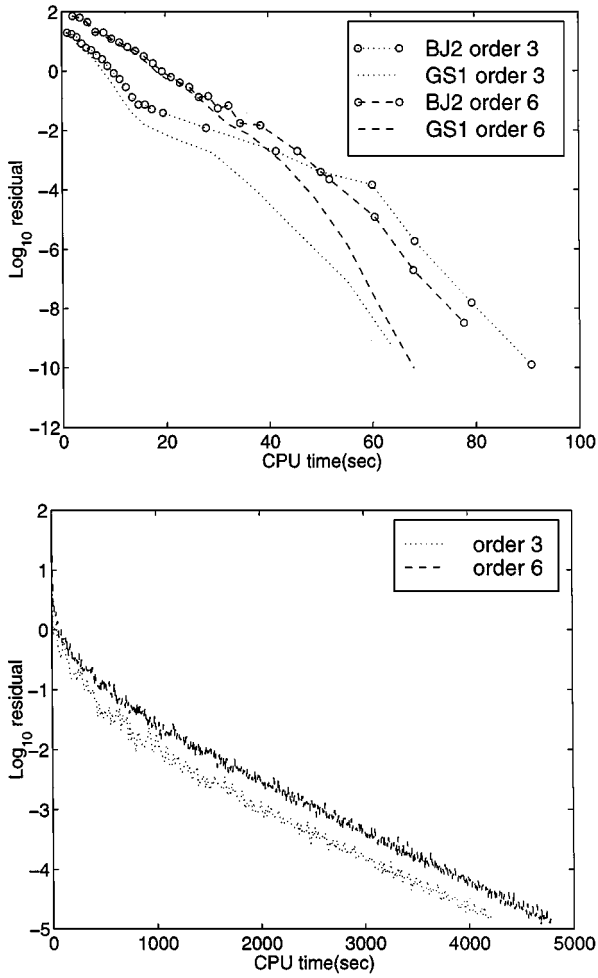

FIG. 4. Convergence history of the implicit method using the two-level block Jacobi (BJ2) and the one-level block LU-SGS (GS1) preconditioners (top), and the explicit method (bottom). (· · · · and · · o · ·) third-order method on the fine mesh, (- - - - and - - o - -) sixth-order method on the coarse mesh.

but the average number of inner GMRES iterations remains approximately the same (see Table II).

Next, the solution is computed on three different meshes with the same number of degrees of freedom. The fine mesh is obtained by dividing each element of the medium mesh (132 elements) into four cells and the coarse mesh is obtained by gathering each four neighboring elements of the medium mesh into one cell. The fine mesh contains 528 elements, the medium mesh 132 elements, and the coarse mesh 33 elements. They are respectively discretized with second-, fourth-, and eighth-order methods. The number of degrees of freedom in each mesh is then 8448. As expected we can see from Fig. 7 that the two-level preconditioners are more efficient as the mesh is refined. The improvement is dramatic for the block Jacobi method with a speedup of more than a factor 2 on the fine mesh.

The comparison between the implicit and the explicit method is presented in Fig. 8. The best performance is obtained with the Newton–Krylov–Schwarz method using the

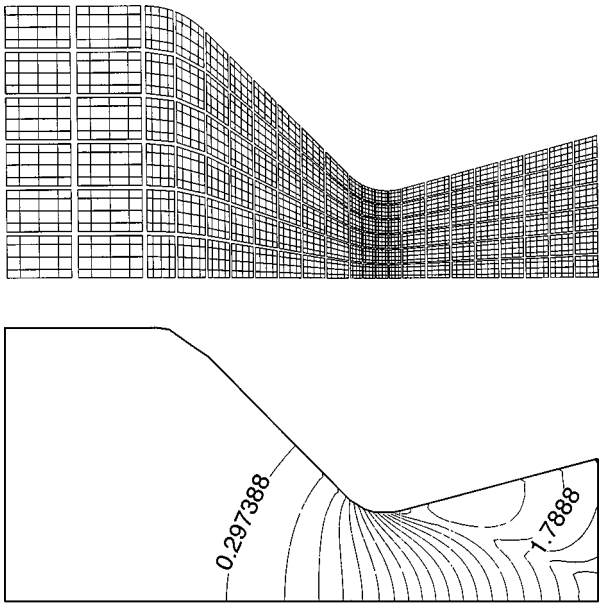

FIG. 5. Geometry and grid with fifth-order spatial discretization (top) and Mach number contours (bottom).

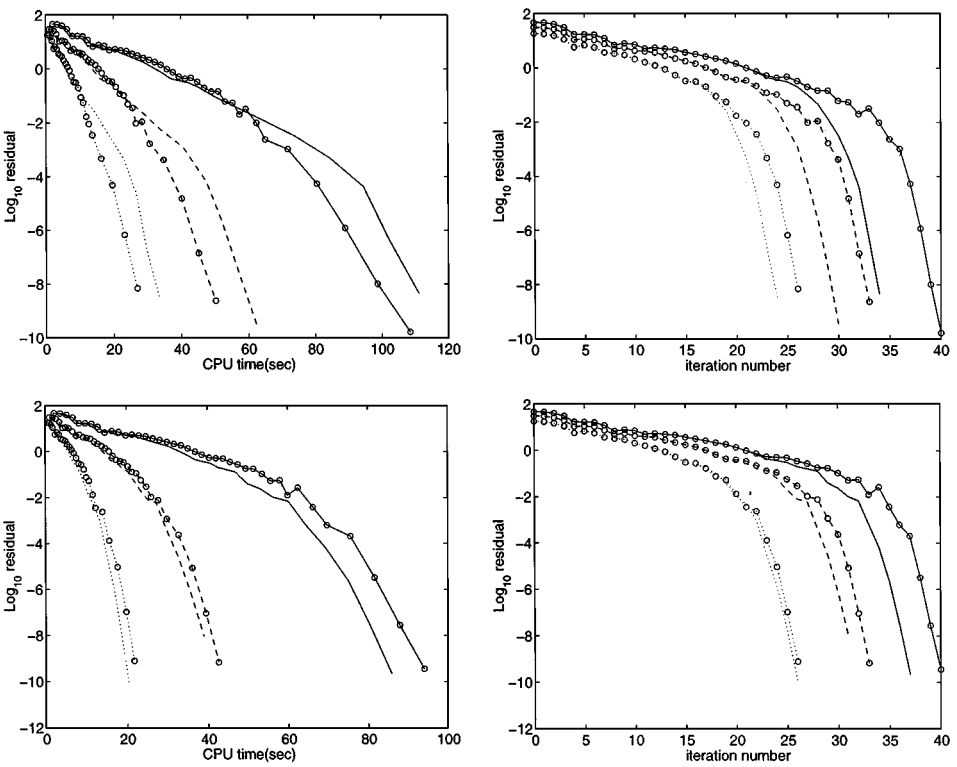

FIG. 6. Convergence history on the medium mesh (132 elements) with the one-level and two-level block Jacobi preconditioners (top) and the one-level and two-level block LU-SGS preconditioners (bottom). Fifth-order (— one-level, —o— two-level), fourth-order (- - - one-level, - - o - - two-level), third-order (· · · one-level, · · o · · two-level). Log residual vs. CPU time (left), Log residual vs. outer Newton iteration number (right).

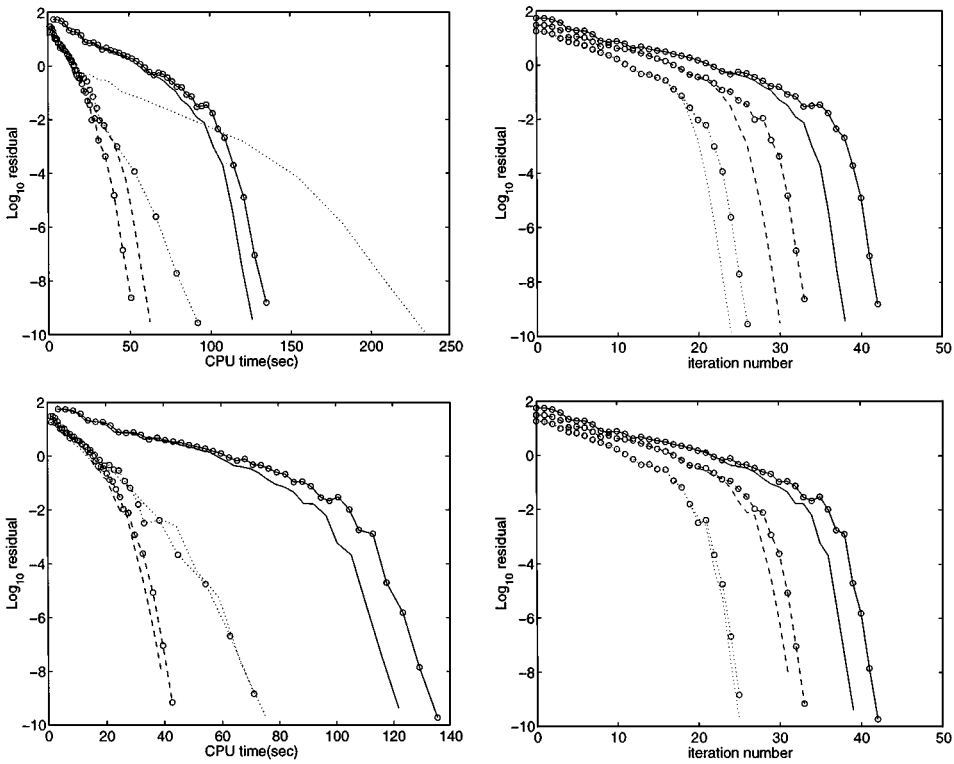

FIG. 7. Convergence history with the one-level and two-level block Jacobi preconditioners (top) and the one-level and two-level block LU-SGS preconditioners (bottom). Eighth-order on the coarse mesh (— one-level, —o— two-level), fourth-order on the medium mesh (- - - one-level, - - o - - two-level), second-order on the fine mesh (· · · · one-level, · · o · · two-level). Log residual vs. CPU time (left), Log residual vs. outer Newton iteration number (right).

one-level LU-SGS preconditioner, which is over 50 times faster than the explicit method.

Figures 7 and 8 show that the scheme with order 4 takes less CPU time to converge than the scheme of order 2. This is due to the fact that the average number of inner GMRES iterations increases as the number of elements increases while it remains approximately the same for different orders of spatial approximation on the same mesh (see Tables I and II).

TABLE II

Average Number of Inner GMRES Iterations for the Transonic Nozzle Flow Using One-Level and Two-Level Block Jacobi and Block LU-SGS Preconditioners on Coarse (33 Elements), Medium (132 Elements), and Fine (528 Elements) Grids

	BJ1	BJ2	GS1	GS2
Order 2 (528 elements)	60.0000	15.8889	10.4615	8.0000
Order 3 (132 elements)	22.0400	13.1481	6.6296	6.4074
Order 4 (132 elements)	20.5161	11.9706	5.7812	5.7647
Order 5 (132 elements)	17.7143	12.4634	5.7895	5.6585
Order 8 (33 elements)	10.3590	9.0465	4.4250	4.7907

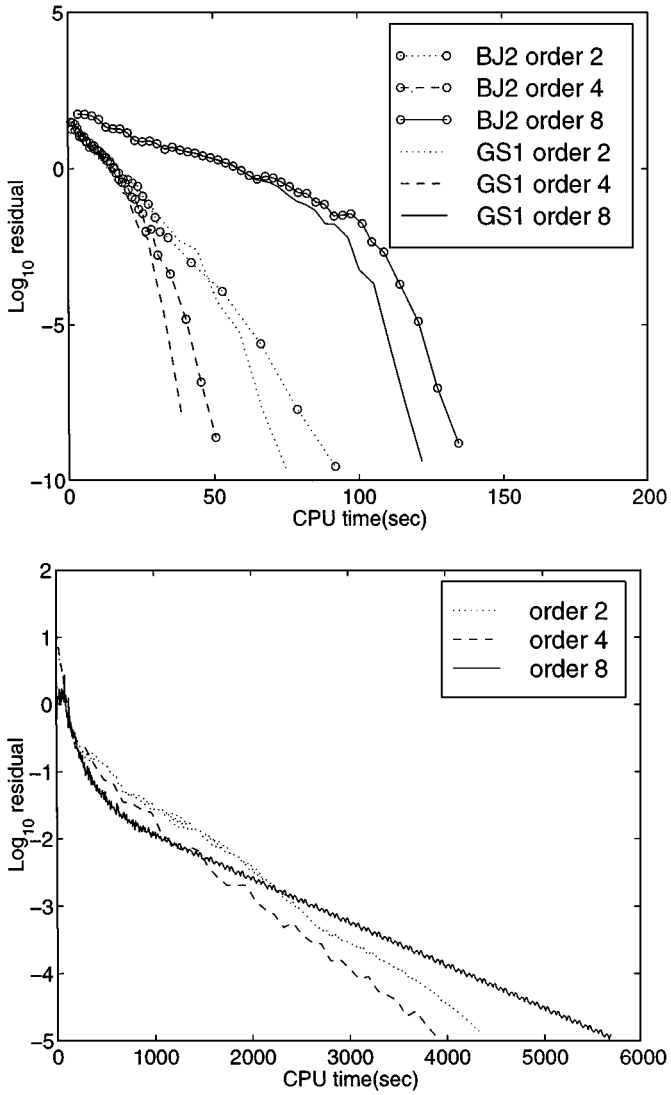

FIG. 8. Convergence history of the implicit method using the two-level block Jacobi (BJ2) and the one-level block LU-SGS (GS1) preconditioners (top), and the explicit method (bottom). (· · · · and · · o · ·) second-order on the fine mesh, (- - - - and - - o - -) fourth-order on the medium mesh, (— and –o–) eighth-order on the coarse mesh.

The scheme with order 4 (132 elements) and the scheme with order 2 (528 elements) have the same convergence rate in the first phase where many Newton steps are performed with only few inner GMRES iterations. In the second phase, where superconvergence occurs, Table II shows that the scheme with order 2 requires more inner GMRES iterations than the scheme with order 4 which results in a slower convergence for the lower order scheme.

6.3. Subsonic Flow Over a Three-Element Airfoil

The third example computes the flow over a three-element airfoil. The grid and geometry for a fifth-order spatial discretization scheme are shown in Fig. 9. The mesh is highly

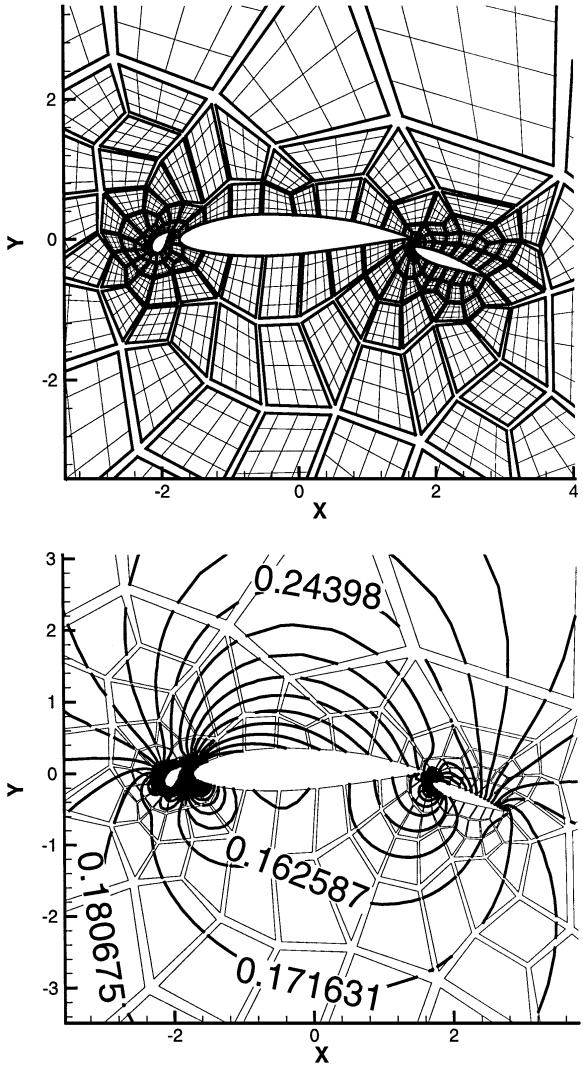

FIG. 9. Geometry and grid with fifth-order spatial discretization (top). Mach number contours (bottom).

unstructured and contains 295 quadrilateral elements. The number of degrees of freedom from the spatial discretization with a fifth-order method is 29500. The freestream condition is set to Mach 0.2 at zero degree angle of attack.

The convergence history of the Newton–Krylov–Schwarz algorithm with different preconditioners is displayed in Fig. 10. We can see from the computations that the two-level method is much more effective than the one-level method. From Table III, we note that the average number of inner GMRES iterations decreases by a factor 4 with the two-level block Jacobi preconditioner and by a factor 2 with the two-level block LU-SGS preconditioner. The two-level implicit algorithms with the block Jacobi and the block LU-SGS preconditioner converge 68% and 34% faster in CPU time, respectively, than their single-level counterparts. As in the previous test case, we note that the use of a two-level method is necessary for the block Jacobi preconditioner when the grid contains a large number of elements.

TABLE III
Statistics for the Newton–Krylov–Schwarz Algorithm for the Subsonic Flow Over a Three-Element Airfoil

	BJ1	BJ2	GS1	GS2
Total CPU time (s)	1687.73	1004.39	1279.29	952.96
Average number of GMRES iterations	76.058	18.7040	26.0244	12.6852
Number of outer Newton iterations	68	124	81	107
Total number of iterations	5248	2338	2134	1370

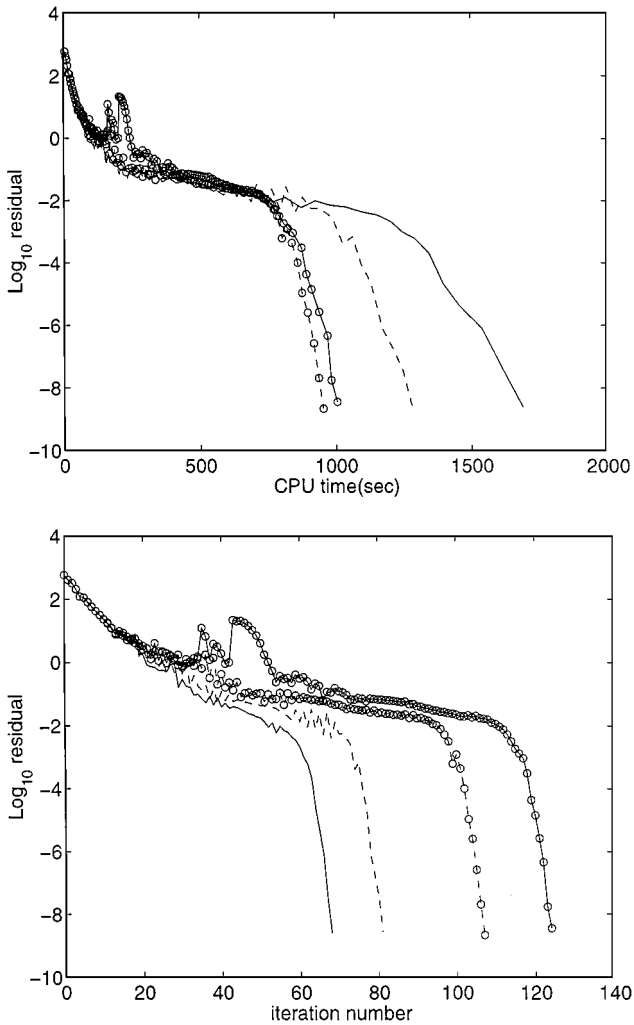

FIG. 10. Convergence history of the Newton–Krylov–Schwarz algorithm using the one-level (—) and the two-level (—o—) block Jacobi and the one-level (- - - -) and the two-level (- - o - -) block LU-SGS preconditioners with a fifth-order spatial discretization. Log residual vs. CPU time (top), Log residual vs. outer Newton iteration number (bottom).

7. CONCLUDING REMARKS

In summary, the implicit discontinuous spectral Galerkin method is applied to three prototypical aerodynamic problems—subsonic flow in a channel with a circular bump, transonic flow in a nozzle, and subsonic flow over a three-element airfoil. Both block Jacobi and LU-symmetric Gauss–Seidel preconditioners are implemented. The latter is found in general to provide a more efficient method. The convergence rate of a discontinuous spectral Galerkin method may become slow considerably with increasing number of elements and a two-level preconditioner is found to ameliorate the convergence rate.

For a small number of elements however, the one-level LU-SGS preconditioner performs better than the two-level preconditioner. The increase in the number of the outer Newton iterations with the two-level method and the overhead of the two-level LU-SGS method in comparison with the one-level LU-SGS method lead to a larger CPU time for convergence when used with small or moderate number of elements. However, the two-level method is essential when the number of elements is large, and the improvement may be dramatic as in the case of three-element airfoil.

The overall performance of the implicit version of the method is orders of magnitude better than an explicit method. Thus, such methods appear to provide a viable alternative to the traditional finite-volume and finite-difference methods for aerodynamic problems. They do require more computation time per node, and this may prove advantageous in the context of a computer's communication speed always lagging behind its computation speed. Furthermore, the algorithms that are required to resolve the flow physics are necessarily of high-order accuracy. In addition to these features, their robustness and easy parallelizability will make these methods more popular in the future.

REFERENCES

1. P. Bar-Yoseph, Space-time discontinuous finite element approximations for multidimensional nonlinear hyperbolic systems, *Comput. Mech*. **5**, 145 (1989).
2. P. Bar-Yoseph and D. Elata, An efficient L^2 Galerkin finite element method for multidimensional nonlinear hyperbolic systems, *Int. J. Numer. Meth. Eng*. **29**, 1229 (1990).
3. F. Bassi and S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations, 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. 197–208.
4. P. N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Stat. Comp*. **11**, 450 (1990).
5. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, Spectral methods in fluid dynamics, Springer Series in Computational Physics (Springer Verlag, New York, 1988).
6. B. Cockburn and C.-W. Shu, TVD Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework, *Math. Comput*. **52**, 411 (1989).
7. B. Cockburn and C.-W. Shu, The Runge–Kutta discontinuous Galerkin finite-element method for conservation laws V: multidimensional systems, *J. Comput. Phys*. **141**, 199 (1998).
8. B. Cockburn, G. E. Karniadakis, and C.-W. Shu, *Discontinuous Galerkin Methods. Theory, Computation and Applications*. Lecture Notes in Computational Science and Engineering, (Springer-Verlag, New York, 2000), Vol. 11.
9. R. Dembo, S. Eisenstat, and T. Steilhaug, Inexact Newton Methods, *SIAM J. Numer. Anal*. **19**, 400 (1982).
10. L. Fezoui and B. Stoufflet, A class of implicit upwind schemes for Euler simulations with unstructured meshes, *J. Comput. Phys*. **84**, 174 (1989).

11. F. Q. Hu, M. Y. Hussaini, and P. Rasetarinera, An analysis of the discontinuous Galerkin method for wave propagation problems, *J. Comput. Phys.* **151**, 921 (1999).
12. A. Jameson and S. Yoon, Lower-upper implicit schemes with multiples grids for the Euler equations, *AIAA J.* **25**, 7 (1987).
13. D. E. Keyes, Aerodynamic applications of Newton–Krylov–Schwarz solvers, in *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, edited by S. M. Deshpande, Lecture Notes in Physics (Springer-Verlag, Berlin, 1995), Vol. 453, pp. 1–20.
14. H. Lomax and J. L. Steger, Relaxation methods in fluid mechanics, *Ann. Rev. Fluid Mech.* **7**, 63 (1975).
15. H. Luo, J. D. Baum, and R. Löhner, A fast, matrix-free implicit method for compressible flows on unstructured grids, *J. Comput. Phys.* **146**, 664 (1998).
16. W. Mulder and B. V. Leer, Experiments with implicit upwind methods for the Euler equations, *J. Comput. Phys.* **59**, 232 (1985).
17. P. Rasetarinera, M. Y. Hussaini, and F. Q. Hu, Some remarks on the accuracy of a 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. 407–412.
18. P. Rasetarinera, D. Kopriva, and M. Y. Hussaini, Discontinuous spectral element solution of acoustic radiation from thin airfoils, *AIAA J.*, in press.
19. W. H. Reed and T. R. Hill, *Triangular Mesh Methods for the Neutron Transport Equation*, Technical Report LA-UR-73-479 (Los Alamos Scientific Laboratory, 1973).
20. Y. Saad and M. H. Schultz, GMRES: A genearlized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.* **7**, 865 (1986).
21. C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.* **77**, 439 (1988).
22. J. Steger and R. F. Warming, Flux vector splitting of the inviscid gas-dynamic equations with applications to the finite difference methods, *J. Comput. Phys.* **40**, 263 (1981).