JOURNAL OF COMPUTATIONAL PHYSICS **146,** 664–690 (1998)
ARTICLE NO. CP986076

A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids

Hong Luo,*,[1] Joseph D. Baum,* and Rainald Löhner†

**Applied Physics Operations, Science Applications International Corporation, McLean, Virginia 22102; †Institute for Computational Sciences and Informatics, George Mason University, Fairfax, Virginia 22030*
E-mail: luo@mclapo.saic.com

Received October 21, 1997; revised July 31, 1998

A fast, matrix-free implicit method has been developed to solve the three-dimensional compressible Euler and Navier–Stokes equations on unstructured meshes. An approximate system of linear equations arising from the Newton linearization is solved by the GMRES (generalized minimum residual) algorithm with a LU-SGS (lower–upper symmetric Gauss–Seidel) preconditioner. A remarkable feature of the present GMRES+LU-SGS method is that the storage of the Jacobian matrix can be completely eliminated by approximating the Jacobian with numerical fluxes, resulting in a matrix-free implicit method. The method developed has been used to compute the compressible flows around 3D complex aerodynamic configurations for a wide range of flow conditions, from subsonic to supersonic. The numerical results obtained indicate that the use of the GMRES+LU-SGS method leads to a significant increase in performance over the best current implicit methods, GMRES+ILU and LU-SGS, while maintaining memory requirements similar to its explicit counterpart. An overall speedup factor from eight to more than one order of magnitude for all test cases in comparison with the explicit method is demonstrated. © 1998 Academic Press

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

[1] Corresponding author.

developing numerical algorithms for the solution of the compressible Euler and Navier–Stokes equations on unstructured grids.

Early efforts in the development of temporal discretization methods using unstructured grids focused on explicit schemes. Usually, explicit temporal discretizations such as multistage Runge–Kutta schemes are used to drive the solution to steady state. Acceleration techniques such as local time-stepping and implicit residual smoothing have also been combined in this context. In general, explicit schemes and their boundary conditions are easy to implement, vectorize and parallelize, and require only limited memory storage. However, for large-scale problems and especially for the solution of the Navier–Stokes equations, the rate of convergence slows down dramatically, resulting in inefficient solution techniques. 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. The most widely used methods to solve a linear system on unstructured grids are iterative solution methods and approximate factorization methods. Significant efforts have been made to develop efficient iterative solution methods. These range from Gauss–Seidel to Krylov subspace methods that use a wide variety of preconditioners (see, e.g., Stoufflet [1], Batina [2], Venkatakrishnan *et al*. [3], Knight [4], Whitaker [5], Luo *et al*. [6], and Barth *et al*. [7]). The most successful and effective iterative method is to use the Krylov subspace methods [8, 9] such as GMRES and BICGSTAB with an ILU (incomplete lower–upper) factorization preconditioner. The drawback is that they require a considerableœ amount of memory to store the Jacobian matrix, which may be prohibitive for large problems. Recently, the lower–upper symmetric Gauss-Seidel method developed first by Jameson and Yoon [10] on structured grids has been successfully generalized and extended to unstructured meshes by several authors [11–13]. The most attractive feature of this approximate factorization method is that the evaluation and storage of the Jacobian matrix inherent in the original formulation of the LU-SGS method can be completely eliminated by making some approximations to the implicit operator. The resulting LU-SGS method can be made even cheaper than the explicit method per time step. However, this method is less effective than the most efficient iterative methods such as GMRES+ILU, because of slow convergence, requiring thousands of time steps to achieve a steady state.

The objective of the effort discussed in this paper is to develop a fast implicit method for solving compressible flow problems around 3D complex, realistic aerodynamic configurations on unstructured grids. Typically, hundreds of thousands of mesh points are necessary to represent such engineering-type configurations accurately. Any implicit methods requiring the storage of the Jacobian matrix would be impractical, if not impossible to use to solve such large-scale problems, where the storage requirement can easily exceed the memory limitation of present computers. In the present work a system of linear equations, arising from an approximate linearization of a fully implicit temporal discretization at each time step, is solved iteratively by a GMRES algorithm with an LU-SGS preconditioner. The idea behind this is to combine the efficiency of the iterative methods and low memory requirement of approximate factorization methods in an effort to develop a fast, low storage implicit method. An apparent advantage of the LU-SGS preconditioner is that it uses the Jacobian matrix of the linearized scheme as a preconditioner matrix, as compared with ILU preconditioner and, consequently, does not require any additional memory storage and computational effort to store and compute the preconditioner matrix. Furthermore, the storage of the approximate Jacobian matrix can be completely eliminated by approximating the Jacobian with numerical fluxes, which will lead to a fast, low-storage implicit algorithm. The matrix-free implicit method developed has been used to compute a wide range of test problems and has been compared with a well-known GMRES+ILU algorithm and an approximately factored implicit algorithm LU-SGS. The new algorithm is found to offer substantial CPU time savings over the best current implicit methods, while maintaining a memory requirement competitive with the explicit method.

2. GOVERNING EQUATIONS

The Navier–Stokes equations governing unsteady compressible viscous flows can be expressed in the conservative form as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^j}{\partial \mathbf{x}_j} = \frac{\partial \mathbf{G}^j}{\partial \mathbf{x}_j}, \tag{2.1}$$

where the summation convention has been employed. The unknown vector **U**, inviscid flux vector **F**, and viscous flux vector **G** are defined by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, \quad \mathbf{F}^j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix}, \quad \mathbf{G}^j = \begin{pmatrix} 0 \\ \sigma_{ij} \\ u_l \sigma_{lj} + k\frac{\partial T}{\partial x_j} \end{pmatrix}. \tag{2.2}$$

Here ρ, p, e, T, and k denote the density, pressure, specific total energy, temperature, and thermal conductivity 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,

$$p = (\gamma - 1)\rho\left(e - \frac{1}{2}u_j u_j\right), \quad T = \left(e - \frac{1}{2}u_j u_j\right)\Big/ C_v, \tag{2.3}$$

which is valid for perfect gas, where γ is the ratio of the specific heats, and C_v is the specific heat at constant volume. The components of the viscous stress tensor σ_{ij} are given by

$$\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \lambda\frac{\partial u_k}{\partial x_k}\delta_{ij}. \tag{2.4}$$

The thermal conductivity k and viscosity coefficient μ are assumed to be a function of the temperature and are determined using Sutherland's empirical relation. It is assumed that λ and μ are related by Stokes' hypothesis

$$\lambda = -\frac{2\mu}{3}. \tag{2.5}$$

The left-hand side of Eq. (2.1) constitutes the Euler equations governing unsteady compressible inviscid flows.

In the sequel, we assume that Ω is the flow domain, Γ is its boundary, and n_j is the unit outward normal vector to the boundary.

3. HYBRID DISCRETE FORMULATION

Assuming Ω_h is a classical triangulation of Ω, N_I is a standard linear finite element shape function associated with a node I, and C_I is a dual mesh cell associated with the node, the hybrid finite volume and finite element formulation used for discretization of the Navier–Stokes equations is

$$\text{find } \mathbf{U}_h \in \mathcal{T}_h \text{ such that for each } N_I\ (1 \le I \le n)$$
$$\int_{C_I} \frac{\partial \mathbf{U}_h}{\partial t}\, d\Omega + \int_{\partial C_I} \mathbf{F}^j(\mathbf{U}_h)\cdot \mathbf{n}_j\, d\Gamma = \int_{\Omega_h} \frac{\partial \mathbf{G}^j(\mathbf{U}_h)}{\partial \mathbf{x}_j} N_I\, d\Omega, \tag{3.1}$$

where $\mathcal{T}_h$ is a discrete approximation space of suitable continuous functions. Inviscid fluxes are discretized using a cell-vertex finite volume formulation, where the control volumes are nonoverlapping dual cells constructed by the median planes of the tetrahedra. In the present study the numerical flux functions for inviscid fluxes at the dual mesh cell interface are computed using the AUSM+ [14] (advection upwind splitting method) scheme. A MUSCL [15] approach is used to achieve high-order accuracy. The Van Albada limiter based on primitive variables is used to suppress the spurious oscillation in the vicinity of the discontinuities. The implementation of the precise MUSCL strategy used in the present work can be found in Ref. [21]. Viscous flux terms are evaluated using a linear finite element approximation, which is equivalent to a second-order accurate central difference.

Equation (3.1) can then be rewritten in a semi-discrete form as

$$V_i \frac{\partial \mathbf{U}_i}{\partial t} = \mathbf{R}_i, \tag{3.2}$$

where V_i is the volume of the dual mesh cell (equivalent to the lumped mass matrix in the finite element), and $\mathbf{R}_i$ is the right-hand side residual and equals zero for a steady-state solution.

4. IMPLICIT TIME INTEGRATION

In order to obtain a steady-state solution, the spatially discretized Navier–Stokes equations must be integrated in time. Using Euler implicit time-integration, Eq. (3.2) can be written in discrete form as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = \mathbf{R}_i^{n+1}, \tag{4.1}$$

where Δt is the time increment and $\Delta \mathbf{U}^n$ is the difference of an unknown vector between time levels n and $n+1$; i.e.,

$$\Delta \mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n. \tag{4.2}$$

Equation (4.1) can be linearized in time as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = \mathbf{R}_i^n + \frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}} \Delta \mathbf{U}_i, \tag{4.3}$$

where $\mathbf{R}_i$ is the right-hand side residual and equals zero for a steady-state solution. Writing the equation for all nodes leads to the delta form of the backward Euler scheme

$$A\Delta\mathbf{U} = \mathbf{R}, \tag{4.4}$$

where

$$A = \frac{V}{\Delta t}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}}. \tag{4.5}$$

Note that as Δt tends to infinity, the scheme reduces to the standard Newton's method for solving a system of nonlinear equations. Newton's method is known to have a quadratic convergence property. The term $\partial\mathbf{R}/\partial\mathbf{U}$ represents symbolically the Jacobian matrix. It involves the linearization of both inviscid and viscous flux vectors. In order to obtain the quadratic convergence of Newton's method, the linearization of the numerical flux function must be virtually exact. Unfortunately, explicit formation of the Jacobian matrix resulting from the exact linearization of any second-order numerical flux functions for inviscid fluxes can require excessive storage and is extremely expensive, if not impossible to evaluate. In order to reduce the number of nonzero entries in the matrix and to simplify the linearization, only a first-order representation of the numerical fluxes is linearized. This results in the graph of the sparse matrix $\partial\mathbf{R}/\partial\mathbf{U}$ being identical to the graph of the supporting unstructured mesh. In addition, the following simplified flux function is used to obtain the left-hand side Jacobian matrix,

$$\mathbf{R}_i(\mathbf{U}_i, \mathbf{U}_j, \mathbf{n}_{ij}) = \frac{1}{2}(\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij})) - \frac{1}{2}|\lambda_{ij}|(\mathbf{U}_j - \mathbf{U}_i), \tag{4.6}$$

where

$$|\lambda_{ij}| = |\mathbf{V}_{ij}\cdot\mathbf{n}_{ij}| + C_{ij} + \frac{\mu_{ij}}{\rho_{ij}|x_j - x_i|}, \tag{4.7}$$

where $\mathbf{n}_{ij}$ is the unit vector normal to the cell interface, $\mathbf{V}_{ij}$ is the velocity vector, and C_{ij} is the speed of sound. Note that this flux function is derived by replacing the Roe's matrix by its spectral radius in the well-known Roe's Flux function [16],

$$\mathbf{R}_i^{\text{inv}} = \frac{1}{2}(\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij})) - \frac{1}{2}|J(\tilde{U})|(\mathbf{U}_j - \mathbf{U}_i) \tag{4.8}$$

for the inviscid flux vector, and the viscous Jacobian matrix is simply approximated by its spectral radius in the above linearization process. The linearization of flux function (4.6) yields

$$\frac{\partial\mathbf{R}_i}{\partial\mathbf{U}_i} = \frac{1}{2}(J(\mathbf{U}_i) + |\lambda_{ij}|\mathbf{I}) \tag{4.9}$$

$$\frac{\partial\mathbf{R}_i}{\partial\mathbf{U}_j} = \frac{1}{2}(J(\mathbf{U}_j) - |\lambda_{ij}|\mathbf{I}), \tag{4.10}$$

where $J = \partial\mathbf{F}/\partial\mathbf{U}$ represents the Jacobian of the inviscid flux vector. The penalty for making these approximations in the linearization process is that the quadratic convergence of

Newton's method can no longer be achieved because of the mismatch and inconsistency between the right- and left-hand sides in Eq. (4.4). Although the number of time steps (Newton iterations, if Δt tends to infinity) may increase, the cost per each time step is significantly reduced: it takes less CPU time to compute the Jacobian matrix and the conditioning of the simplified Jacobian matrix is improved, thus reducing computational cost to solve the resulting linear system.

As only a first-order representation of the numerical fluxes is considered, the number of nonzero entries in each row of the matrix is related to the number of edges incident to the node associated with that row. In other words, each edge ij will guarantee nonzero entries in the ith column and jth row and, similarly, the jth column and ith row. In addition, nonzero entries will be placed on the diagonal of the matrix. Using an edge-based data structure, the left-hand side Jacobian matrix is stored in upper, lower, and diagonal forms, which can be expressed as

$$U = \frac{1}{2}(J(\mathbf{U}_j, \mathbf{n}_{ij}) - |\lambda_{ij}|\mathbf{I}), \tag{4.11}$$

$$L = \frac{1}{2}(-J(\mathbf{U}_i, \mathbf{n}_{ij}) - |\lambda_{ij}|\mathbf{I}), \tag{4.12}$$

$$D = \frac{V}{\Delta t}\mathbf{I} + \sum_j \frac{1}{2}(J(\mathbf{U}_i, \mathbf{n}_{ij}) + |\lambda_{ij}|\mathbf{I}). \tag{4.13}$$

Note that U, L, and D represent the strict upper matrix, the strict lower matrix, and the diagonal matrix, respectively. Both upper and lower matrices require a storage of *nedge* $\times$ *neqns* $\times$ *neqns* and the diagonal matrix needs a storage of *npoin* $\times$ *neqns* $\times$ *neqns*, where *npoin* is the number of grid points; *neqns* (=5 in 3D) is the number of unknown variables and *nedge* is the number of edges. Note that in 3D *nedge* $\approx$ 7*npoin*. Clearly, the upper and lower matrix consume substantial amounts of memory, taking 93% of the storage required for left-hand side Jacobian matrix.

Equation (4.4) represents a system of linear simultaneous algebraic equations and needs to be solved at each time step. The most widely used methods to solve this linear system are iterative solution methods and approximate factorization methods. Recently, the lower-upper symmetric Gauss–Seidel method developed first by Jameson and Yoon [10] on structured grids has been successfully generalized and extended to unstructured meshes by several authors [11–13]. The LU-SGS method is attractive because of its good stability properties and competitive computational cost in comparison to explicit methods. In this method, the matrix A is split in three matrices, a strict lower matrix L, a diagonal matrix D, and a strict upper matrix U. This system is approximately factored by neglecting the last term on the right-hand side of Eq. (4.14). The resulting equation can be solved in the two steps shown in Eqs. (4.15) and (4.16), each of them involving only simple block matrix inversions:

$$(D + L)D^{-1}(D + U)\Delta\mathbf{U} = \mathbf{R} + (LD^{-1}U)\Delta\mathbf{U}. \tag{4.14}$$

Lower (forward) sweep:

$$(D + L)\Delta\mathbf{U}^{\star} = \mathbf{R}. \tag{4.15}$$

Upper (backward) sweep:

$$(D+U)\Delta\mathbf{U} = D\Delta\mathbf{U}^{\star}. \tag{4.16}$$

Both lower and upper sweeps can be vectorized by appropriately reordering the grid points [13], resulting in a very efficient algorithm. It is found that the CPU cost of one LU-SGS step is approximately 50% of one three-stage Runge–Kutta explicit step.

It is clear that the above algorithm involves primarily the Jacobian matrix-solution incremental vector product. Such operation can be approximately replaced by computing increments of the flux vector $\Delta\mathbf{F}$:

$$J\Delta\mathbf{U} \approx \Delta\mathbf{F} = \mathbf{F}(\mathbf{U}+\Delta\mathbf{U}) - \mathbf{F}(\mathbf{U}). \tag{4.17}$$

This idea of the matrix-free approach, in which the product of Jacobian matrix and incremental vector is approximated by the increment of the flux vector, was first introduced in the work of Sharov and Nakahashi [13]. The forward sweep and backward sweep steps can then be expressed as

$$\Delta\mathbf{U}_i^{\star} = D^{-1}\left[\mathbf{R}_i - \sum_{j:j<i}\frac{1}{2}(\Delta\mathbf{F}_j^{\star} - |\lambda_{ij}|\Delta\mathbf{U}_j^{\star})s_{ij}\right], \tag{4.18}$$

$$\Delta\mathbf{U}_i = \Delta\mathbf{U}_i^{\star} - D^{-1}\sum_{j:j>i}\frac{1}{2}(\Delta\mathbf{F}_j - |\lambda_{ij}|\Delta\mathbf{U}_j)s_{ij}. \tag{4.19}$$

The most remarkable achievement of this approximation is that there is no need to store the upper and lower matrics U and L, which substantially reduces the memory requirements. It is found that this approximation does not compromise any numerical accuracy, and the extra computational cost is negligible.

Although the LU-SGS method is more efficient than its explicit counterpart, a significant number of time steps are still required to achieve the steady-state solution, due to the nature of the approximation factorization schemes. One way to speed up the convergence is to use iterative methods. In this work, the system of linear equations is solved by the generalized minimal residual (GMRES) method of Saad and Schultz [8]. This is a generalization of the conjugate gradient method for solving a linear system where the coefficient matrix is not symmetric and/or positive definite. The use of GMRES combined with different preconditioning techniques is becoming widespread in the CFD community for the solution of the Euler and Navier–Stokes equations [3, 5–7]. GMRES minimizes the norm of the computed residual vector over the subspace spanned by a certain number of orthogonal search directions. It is well known that the speed of convergence of an iterative algorithm for a linear system depends on the condition number of the matrix A. GMRES works best when the eigenvalues of matrix A are clustered. The easiest and the most common way to improve the efficiency and robustness of GMRES is to use preconditioning to attempt to cluster the eigenvalues at a single value. The preconditioning technique involves solving an equivalent preconditioned linear system,

$$\tilde{A}\Delta\tilde{\mathbf{U}} = \tilde{\mathbf{R}}, \tag{4.20}$$

instead of the original system (4.4), in the hope that $\tilde{A}$ is well conditioned. Left preconditioning involves premultiplying the linear system with a matrix as

$$P^{-1}A\Delta\mathbf{U} = P^{-1}\mathbf{R}, \tag{4.21}$$

where P is the preconditioning matrix. The best preconditioning matrix for A would cluster as many eigenvalues as possible at unity. Obviously, the optimal choice of P is A, in which case the underlying matrix problem for GMRES is trivially solved with one Krylov vector. The motivation for preconditioning is twofold: (a) reduce the computational effort required to solve the linearized system of equations at each time-step; (b) reduce the total number of time steps required to obtain a steady state solution. Preconditioning will be cost-effective only if the additional computational work incurred for each subiteration is compensated for by a reduction in the total number of iterations to convergence. In this way, the total cost of solving the overall nonlinear system is reduced. In the present work, the LU-SGS presented above is used as a preconditioner, i.e.,

$$P = (D + L)D^{-1}(D + U). \tag{4.22}$$

A clear advantage of the LU-SGS preconditioner is that it uses the Jacobian matrix of the linearized scheme as a preconditioner matrix, as compared with ILU preconditioner. Consequently it does not require any additional memory storage and computational effort to store and compute the preconditioner matrix. The preconditioned restarted GMRES algorithm is described below.

ALGORITHM. Restarted preconditioned GMRES(k).

```
For $l = 1, m$ do                                    $m$ restart iterations
    $\mathbf{v}_0 = \mathbf{R} - A\Delta\mathbf{U}_0$                       initial residual
    $\mathbf{r}_0 := P^{-1}\mathbf{v}_0$                             preconditioning step
    $\beta := \|\mathbf{r}_0\|_2$                                initial residual norm
    $\mathbf{v}_1 := \mathbf{r}_0/\beta$                               define initial Krylov
    for $j = 1, k$ do                                 inner iterations
        $\mathbf{y}_j := A\mathbf{v}_j$                           matrix–vector product
        $\mathbf{w}_j := P^{-1}\mathbf{y}_j$                        preconditioning step
        For i $= 1$, j do                              Gram–Schmidt step
            $h_{i,j} := (\mathbf{w}_j, \mathbf{v}_i)$                  .
            $\mathbf{w}_j = \mathbf{w}_j - h_{i,j}\mathbf{v}_i$              .
        EndDo                                         .
        $h_{j+1,j} := \|\mathbf{w}_j\|_2$
        $\mathbf{v}_{j+1} := \mathbf{w}_j/h_{j+1,j}$                  define Krylov vector
    EndDo
    $\mathbf{z} := \min_{\hat{\mathbf{z}}} \|\beta\mathbf{e}_1 - H\hat{\mathbf{z}}\|_2$            least squares solve
    $\Delta\mathbf{U} := \Delta\mathbf{U}_0 + \sum_{i=1}^{m} \mathbf{v}_i\mathbf{z}_i$         approximate solution
    if $\|\beta\mathbf{e}_1 - H\mathbf{z}\|_2 \le \epsilon$ exit          convergence check
    $\Delta\mathbf{U}_0 := \Delta\mathbf{U}$                          restart
EndDo
```

Note that the above GMRES algorithm only requires matrix–vector products, the same technique used in the LU-SGS method can be applied to eliminate the storage of the upper and lower matrices.

The present GMRES+LU-SGS method only requires the storage of the diagonal matrix. In addition, a storage corresponding to $2 \times$ *nedge* is required for the two index arrays, which are necessary to achieve the vectorization of LU-SGS method. The need for additional storage associated with the GMRES algorithm is an array of size $(k + 2) \times$ *neqns* $\times$ *npoin*, where k is the number of search directions. Since the GMRES+LU-SGS is completely separated from the flux computation procedure, memory, which is used to compute fluxes can be used by the GMRES+LU-SGS. Overall, the extra storage of the GMRES+LU-SGS method is approximately 10% of the total memory requirements.

5. NUMERICAL RESULTS

The present implicit method has been used to compute a variety of compressible flow problems for a wide range of flow conditions, from subsonic to supersonic, for both inviscid and viscous flows, in both 2D and 3D. Only a few typical examples in 3D are presented here to demonstrate the effectiveness and robustness of the present implicit method over the

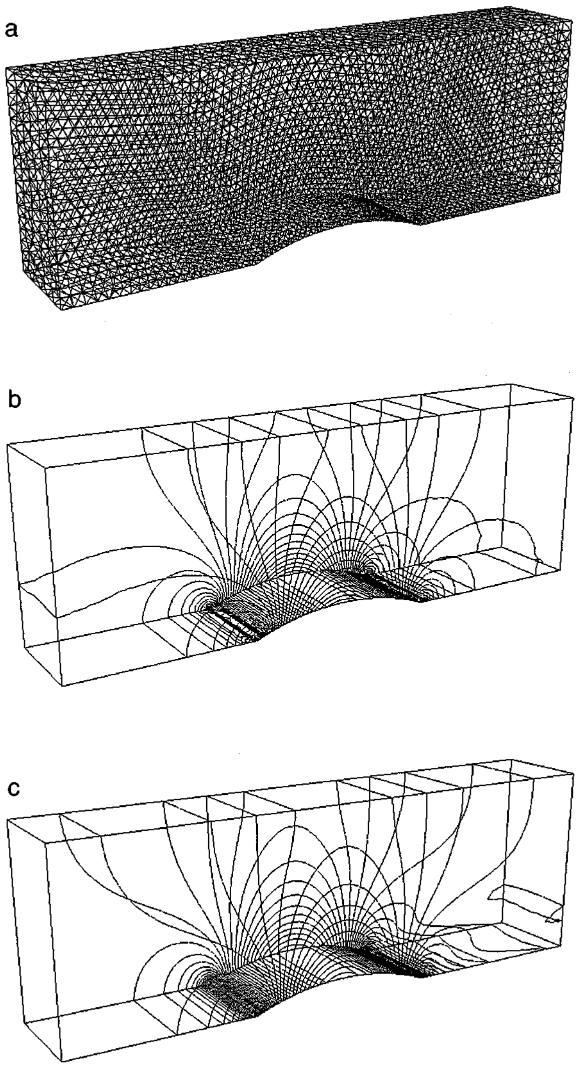
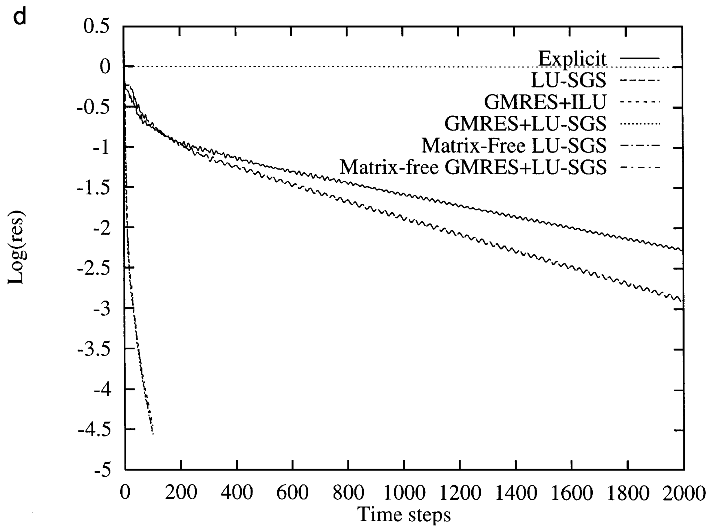


FIG. 1. (a) Surface mesh used for computing channel flow (nelem = 68,097, npoin = 13,091, nboun = 4,442). (b) Computed pressure contours on the channel surface at $M_{in} = 0.5$. (c) Computed Mach number contours on the channel surface at $M_{in} = 0.5$.



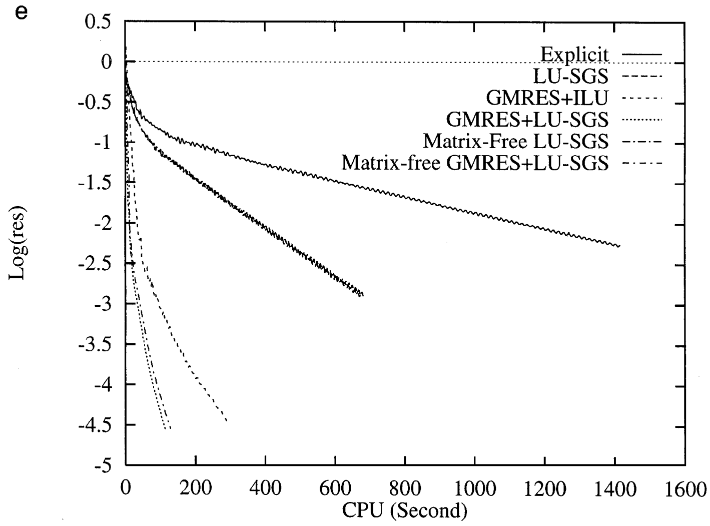
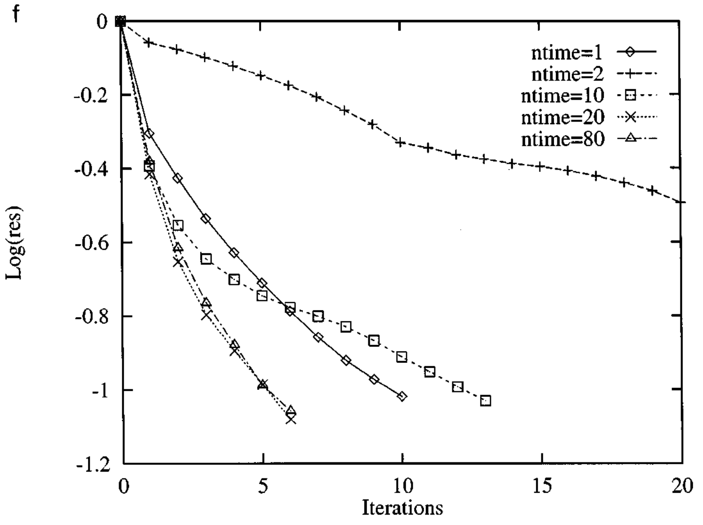


FIG. 1. (d) Convergence history versus time steps for subsonic channel flow using different schemes: explicit, GMRES+ILU, LU-SGS, GMRES+LU-SGS, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (e) Convergence history versus CPU time for subsonic channel flow using different schemes: explicit, GMRES+ILU, LU-SGS, GMRES+LU-SGS, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

existing implicit methods. No attempt has been made to use our GMRES+ILU method [6] to solve the Navier–Stokes equations and large-scale problems, as its storage requirement exceeds the memory limitation of the computer available to us.

All of the grids used here were generated by the advancing front technique [17]. All computations were started with uniform flow. The relative L_2 norm of the density residual is taken as a criterion to test convergence history. The solution tolerance for GMRES is set to 0.1 with 10 search directions and 20 iterations. We observed that during the first few time steps, more iterations are spent to solve the system of the linear equations: even 20 iterations cannot guarantee that the stopping criterion will be satisfied for some problems. However, it only takes four or five iterations to solve the linear equations at a later time, f



g

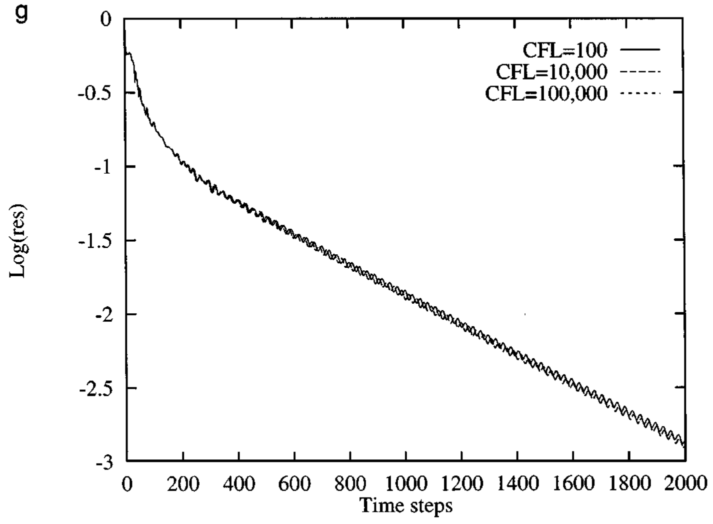
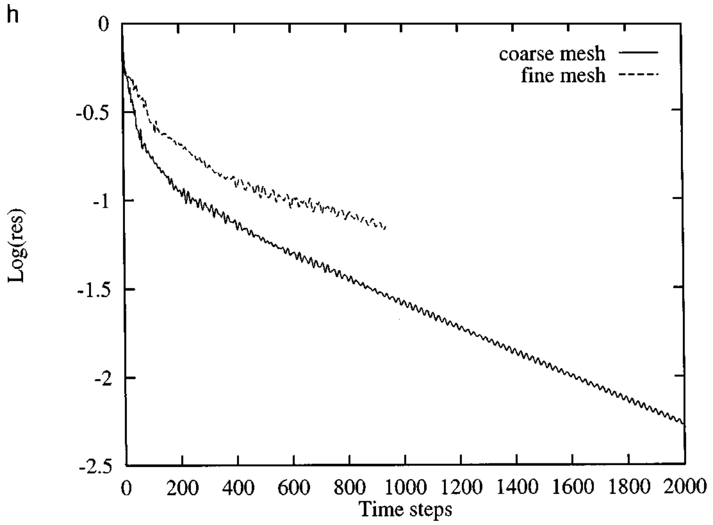


FIG. 1. (f) Convergence history of linear system at different time-steps for subsonic flow by matrix-free GMRES+LU-SGS method. (g) Effect of CFL number on convergence history by matrix-free LU-SGS method for subsonic flow.

and global convergence is not affected by a lack of linear system convergence during the first few time steps.

A. *Inviscid Subsonic Flow in a Channel with a Circular Bump on the Lower Wall*

The first 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 is 1, and its width is 0.5. The inlet Mach number is 0.5. This is a three-dimensional simulation of a two-dimensional flow. Since no shock waves are present in the flow fields, all solutions were obtained using a second-order scheme without any limiters. The mesh, which contains 13,891 grid points, 68,097 elements, and 4442 boundary points is depicted in Fig. 1a. The h



i

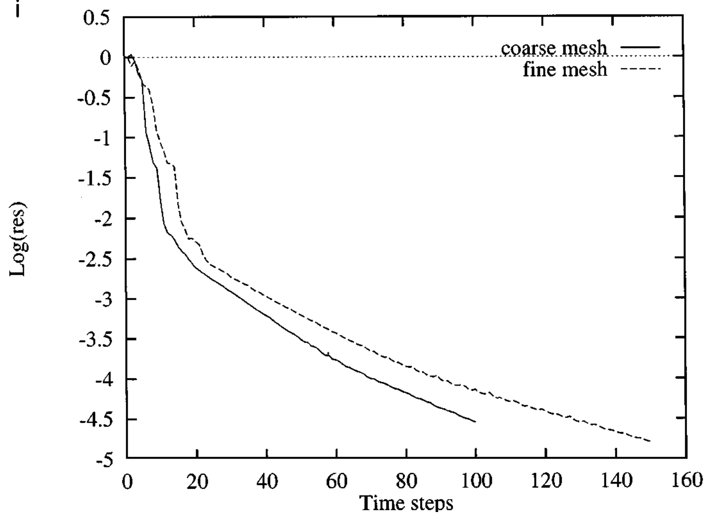


FIG. 1. (h) Convergence history of the residual for coarse and fine meshes for subsonic flow by explicit method. (i) Convergence history of the residual for coarse and fine meshes for subsonic flow by GMRES+LU-SGS implicit method.

computed Mach number and pressure contours in the flow field are shown in Figs. 1b and 1c, respectively. Figures 1d and 1e display a comparison of convergence histories among the explicit scheme, the GMRES+ILU scheme, the LU-SGS scheme, the GMRES+LU-SGS scheme, the matrix-free LU-SGS scheme, and the matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. The explicit method used a three-stage Runge–Kutta time-stepping scheme with local time stepping and implicit residual smoothing. The computation was advanced with a CFL number of 4. A CFL number of 10,000 was used by all implicit methods in the computation. It is clear that GMRES+LU-SGS methods are superior to both GMRES+ILU and LU-SGS methods. The present GMRES+LU-SGS method is over 100 times faster than its explicit counterpart for this particular case. This is due to the fact that the convergence of the explicit method deteriorates dramatically for a low-speed flow problem. From Fig. 1e we observe that both the matrix-free LU-SGS scheme and the matrix-free GMRES+LU-SGS scheme yield a convergence history that is identical to their respective matrix counterparts. This indicates that both matrix-free schemes yield solutions that are identical to their matrix counterparts. However, the matrix-free GMRES+LU-SGS scheme is slightly more expensive than its matrix counterpart, as we can see from Fig. 1f. This is due the fact that for each GMRES iteration, the former involves the numerical flux computation, while the latter involves the computation of a matrix vector product, which is apparently cheaper to calculate. It is worth noting that per time step the present LU-SGS method costs approximately half of a three-stage Runge–Kutta explicit method with a residual smoothing. However, the cost of the GMRES+LU-SGS method per time step is not fixed, as more iterations and, therefore, more CPU time, are required to solve the linear system at earlier time steps. This is shown in Fig. 1f, where the convergence history of the linear system at different time steps using the matrix-free GMRES+LU-SGS method is displayed. Typically, only a few iterations are sufficient to meet the stop criterion at latter time-steps. Finally, Fig. 1g illustrates the convergence history of matrix-free LU-SGS method for different CFL number. Although, the LU-SGS method is stable for a

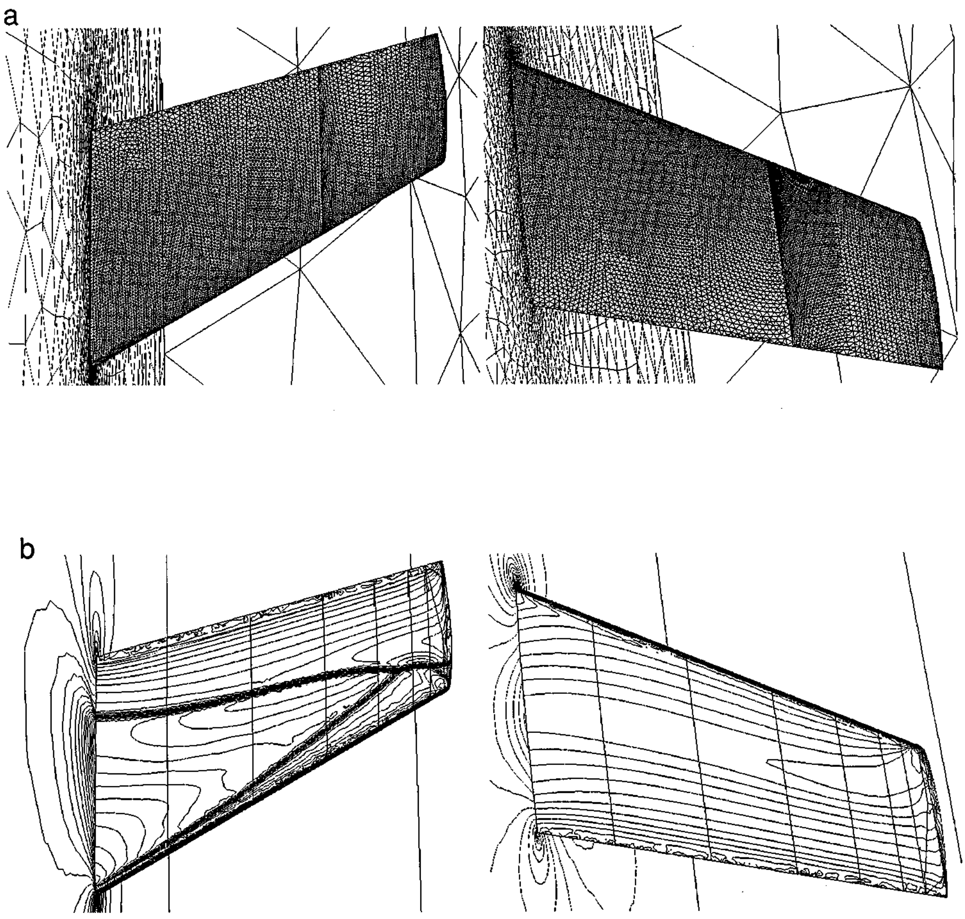


FIG. 2. (a) Upper and lower surface mesh used for M6wing configuration (nelem = 741,098, npoin = 136,051, nboun = 20,762). (b) Computed pressure contours on the upper and lower surface at $M_\infty = 0.84$ and $\alpha = 3.06^\circ$.

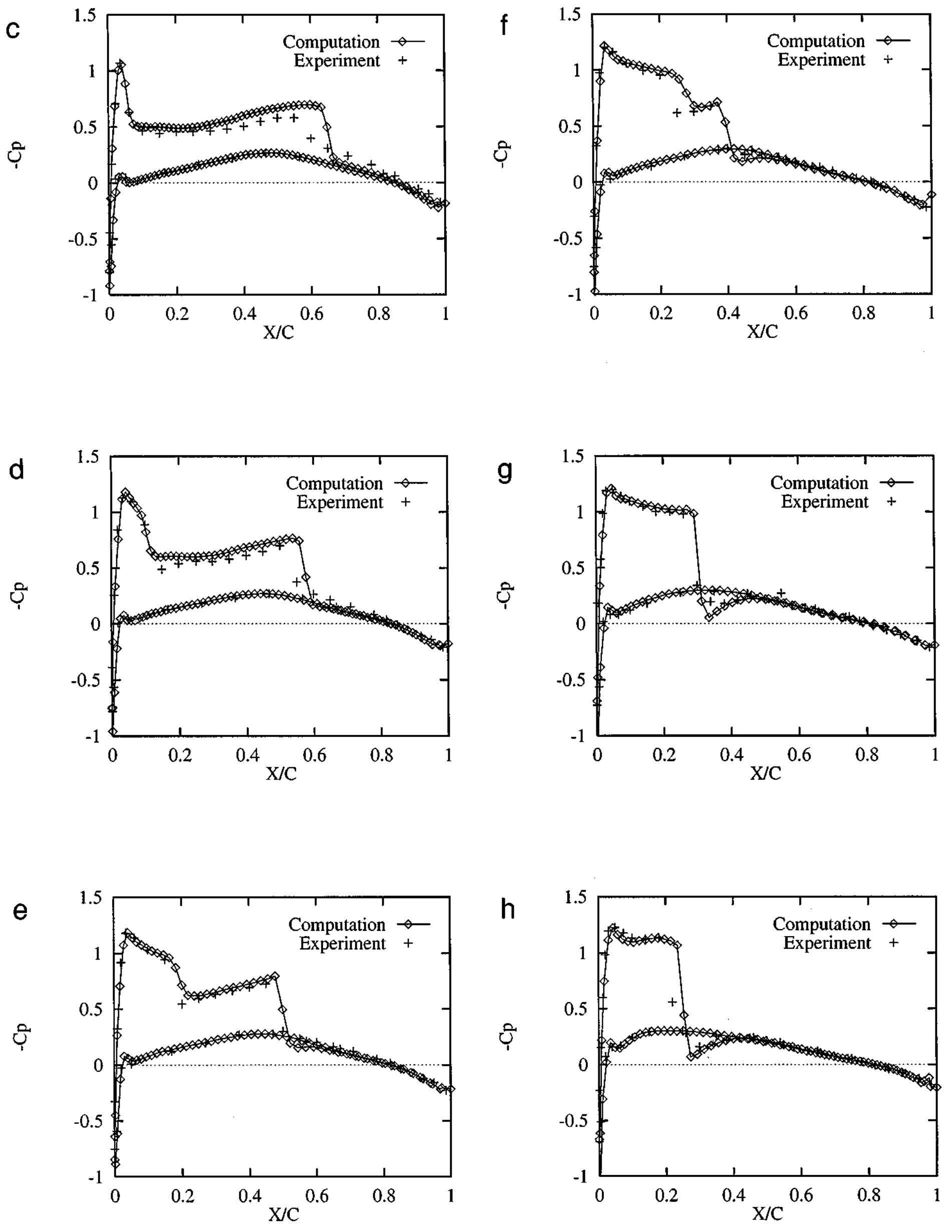
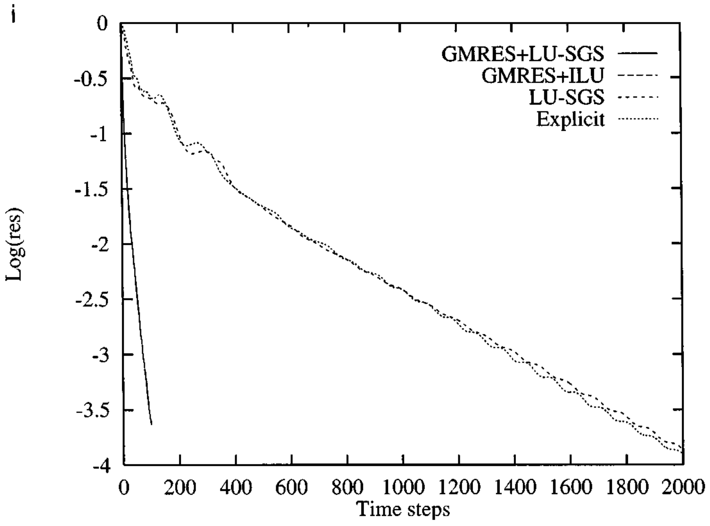


FIG. 2. (c) Comparison between computed and experimental surface pressure coefficient for wing section at 20% semispan. (d) Comparison between computed and experimental surface pressure coefficient for wing section at 44% semispan. (e) Comparison between computed and experimental surface pressure coefficient for wing section at 65% semispan. (f) Comparison between computed and experimental surface pressure coefficient for wing section at 80% semispan. (g) Comparison between computed and experimental surface pressure coefficient for wing section at 90% semispan. (h) Comparison between computed and experimental surface pressure coefficient for wing section at 95% semispan.



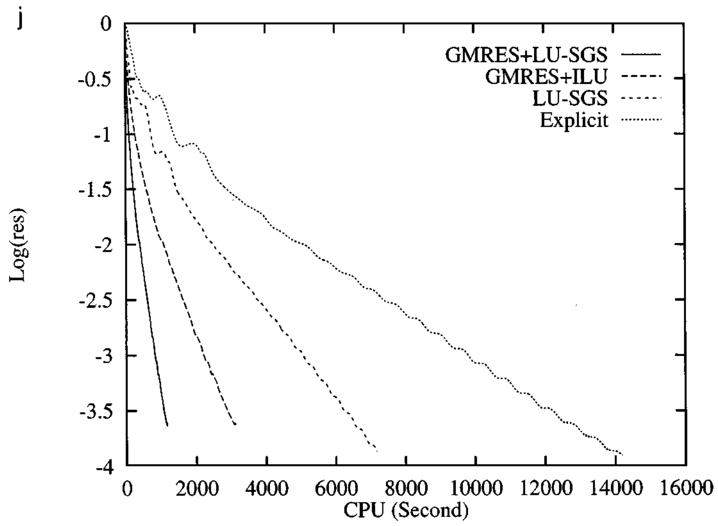


FIG. 2. (i) Residual convergence history versus time steps for M6wing using different schemes: explicit, matrix-free LU-SGS, GMRES+ILU, and matrix-free GMRES+LU-SGS. (j) Residual convergence history versus CPU time for M6wing using different schemes: explicit, matrix-free LU-SGS, GMRES+ILU, and matrix-free GMRES+LU-SGS.

very large CFL number, the convergence history is almost indistinguishable for all the three CFL numbers used.

Finally, the same computation has been performed on a finer mesh to study the behavior of convergence history as the grid is refined. The refined mesh contains 99,683 grid points, 533,060 elements, and 17,391 boundary points. Figures 1h and 1i show the convergence histories of the residual for coarse and finer grids using the explicit method and the matrix-free GMRES+LU-SGS method, respectively. As expected, the explicit method for the refined mesh requires approximately twice the number of time steps to achieve the same convergence; the behavior of convergence history for the implicit method is quite similar as on a coarse mesh, although the rate of convergence does slow down, demonstrating that the advantage of the present implicit method is not diminished for finer grids.

B. *ONERA M6 Wing Configuration*

The second, well-documented case is the inviscid transonic flow over a ONERA M6 wing configuration. The M6 wing has a leading-edge sweep angle of 30°, an aspect of 3.8, and a taper ratio of 0.562. The airfoil section of the wing is the ONERA "D" airfoil, which is a 10% maximum thickness-to-chord ratio conventional section. The flow solutions are presented at a Mach number of 0.84 and an angle of attack of 3.06. The mesh used in the computation

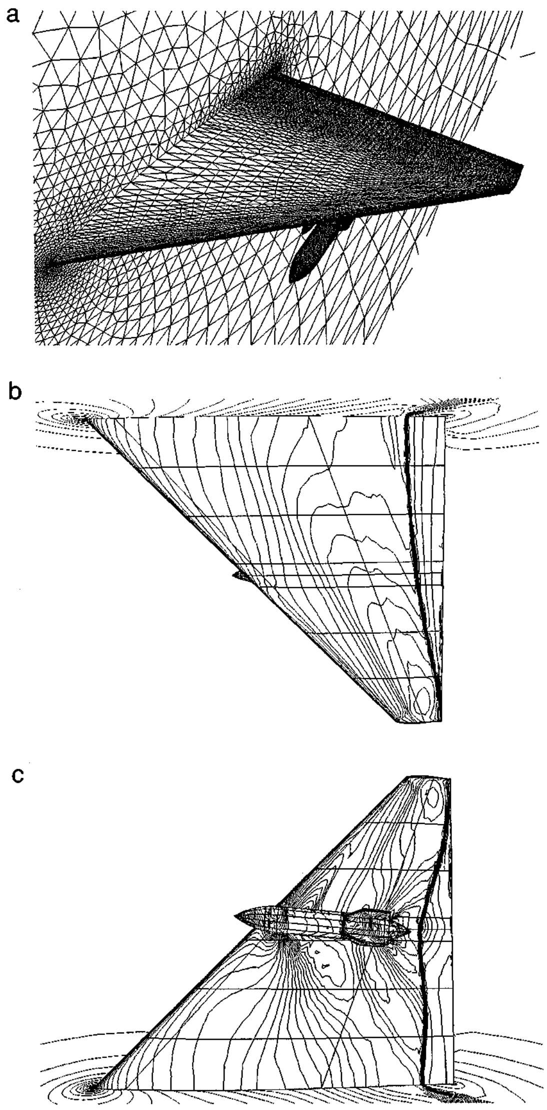
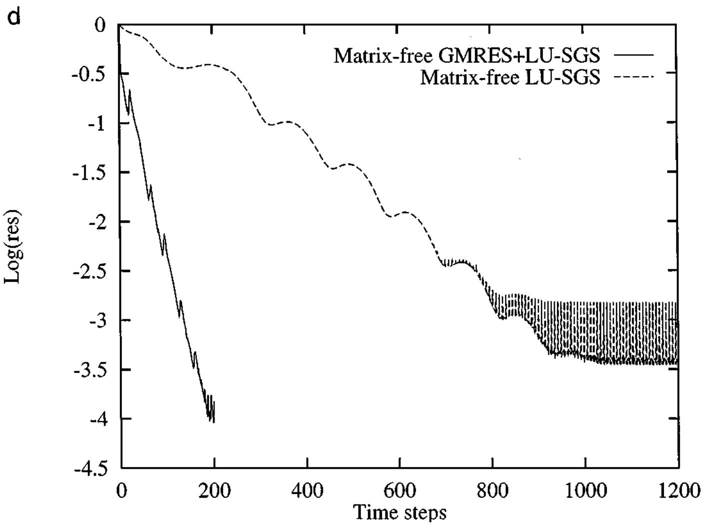


FIG. 3. (a) Surface mesh used for Wing/Pylon/Finned-Store configuration (nelem = 1,329,694, npoin = 239,547, nboun = 27,359). (b) Computed pressure contours on the upper surafce at $M_\infty = 0.95$ and $\alpha = 0°$. (c) Computed pressure contours on the lower surafce at $M_\infty = 0.95$ and $\alpha = 0°$.



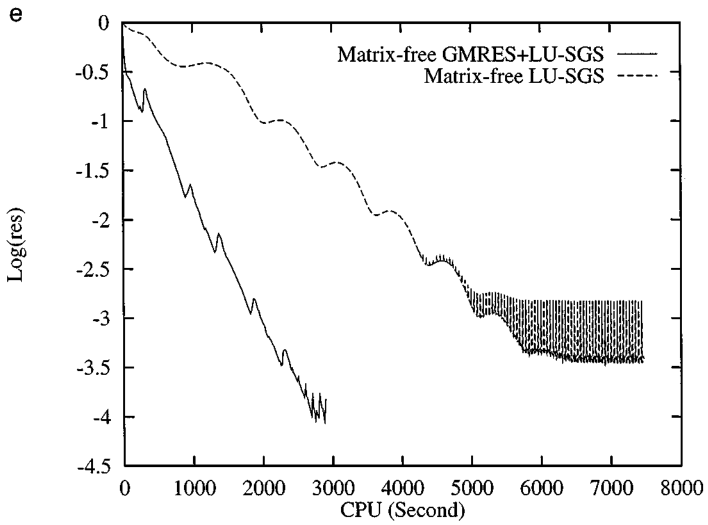
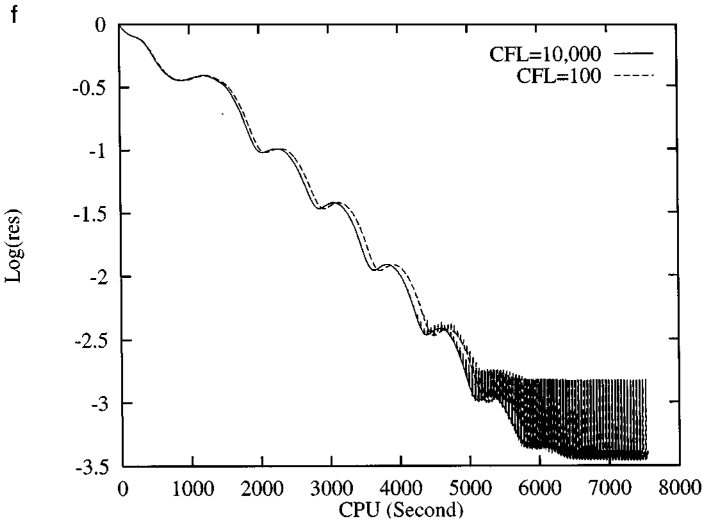


FIG. 3. (d) Residual convergence history versus time steps for Wing/Pylon/Finned-Store configuration using matrix-free LU-SGS and matrix-free GMRES+LU-SGS. (e) Residual convergence history versus CPU time for Wing/Pylon/Finned-Store configuration using different schemes: matrix-free LU-SGS and matrix-free GMRES+LU-SGS.

consists of 741,095 elements, 136,051 grid points, and 20,762 boundary points. The upper and surface meshes are shown in Fig. 2a. The computed pressure contours on the upper and lower surfaces are displayed in Fig. 2b. The upper surface contours clearly show the sharply captured lambda-type shock structure formed by the two inboard shock waves, which merge together near 80% semispan to form the single strong shock wave in the outboard region of the wing. The computed pressure coefficient distributions are compared with experimental data [18] at six spanwise stations in Figs. 2c–2h. We can observe that there is only one grid point within the shock structure; this demonstrates the sharp shock-capturing ability of AUSM+ scheme. The results obtained compare closely with experimental data, except at the root stations, due to lack of viscous effects. Figures 2i and 2j display a comparison f



g

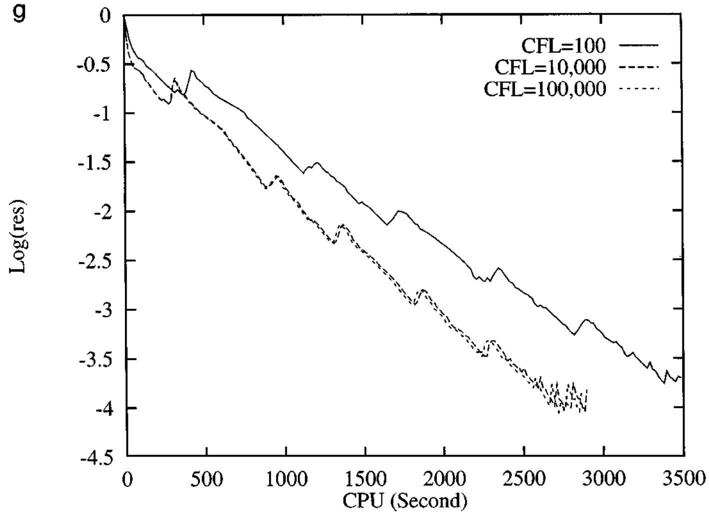


FIG. 3. (f) Effect of CFL number on convergence history for for Wing/Pylon/Finned-Store configuration using matrix-free LU-SGS. (g) Effect of CFL number on convergence history for for Wing/Pylon/Finned-Store configuration using matrix-free GMRES+LU-SGS.

of convergence histories among the explicit scheme, matrix-free LU-SGS scheme, GMRES+ILU scheme, and matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. GMRES+LU-SGS methods provide the best convergence performance. The present GMRES+LU-SGS method is about three times faster than the GMRES+ILU method, about six times faster than the LU-SGS methods and 14 times faster than its explicit method.

C. *Wing/Pylon/Finned-Store Configuration*

The third test case is conducted for a wing/pylon/finned-store configuration reported in Ref. [19]. The configuration consists of a clipped delta wing with a 45° sweep composed

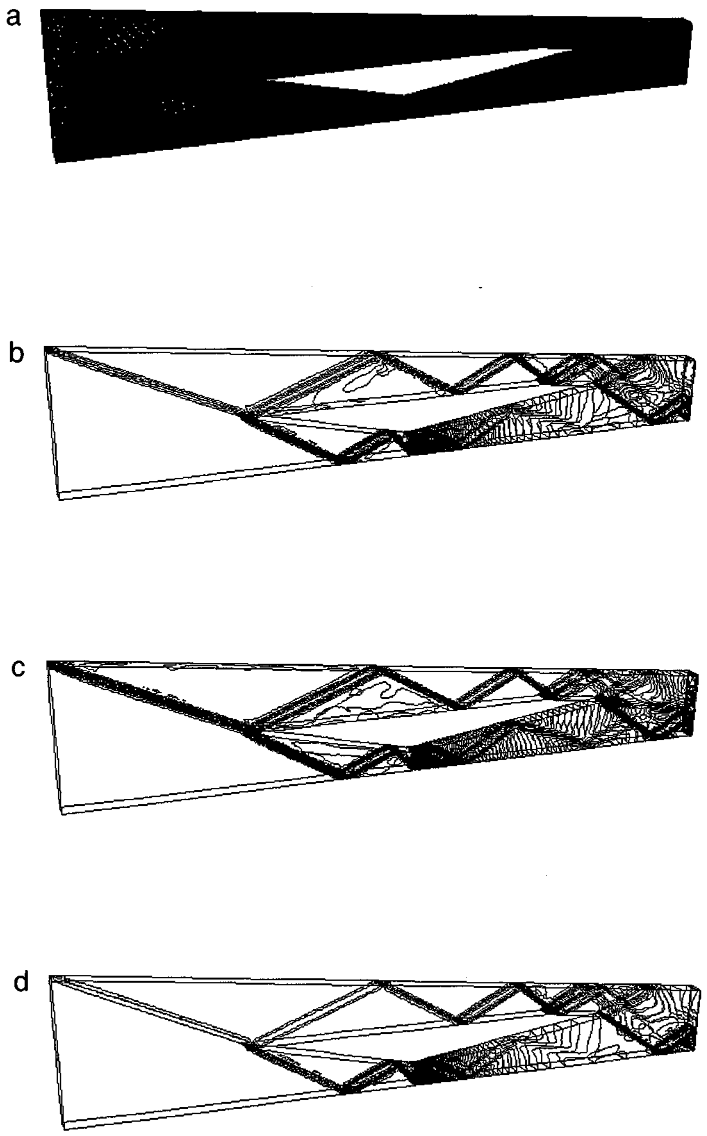
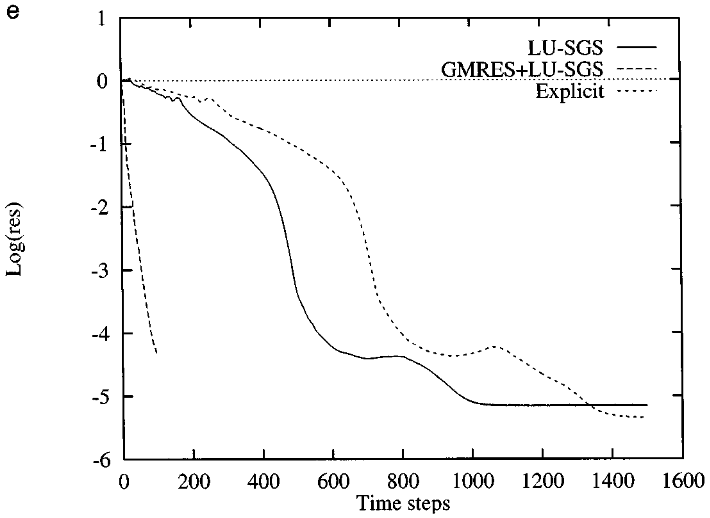


FIG. 4. (a) Surface mesh used for computing internal supersonic flow (nelem = 542,895, npoin = 106,285, nboun = 30,094). (b) Computed density contours for supersonic inlet flow ($M_{in} = 3$). (c) Computed Mach number contours for supersonic inlet flow ($M_{\mathrm{in}} = 3$). (d) Computed pressure contours for supersonic inlet flow ($M_{\mathrm{in}} = 3$).

of a constant NACA64010 symmetric airfoil section. The wing has a root chord of 16 in., a semispan of 13 in., and a taper ratio of 0.134. The pylon is located at the midspan station and has a cross section characterized by a flat plate closed at the leading and trailing edges by a symmetrical ogive shape. The width of the pylon is 0.294 in. The four fins on the store are defined by a constant NACA0008 airfoil section with a leading-edge sweep of 45° and a truncated tip. The mesh used in the computation is shown in Fig. 3a. It contains 1,329,694 elements, 239,547 grid points, and 27,359 boundary points. The flow solutions are presented at a Mach number of 0.95 and an angle of attack of 0°. Figures 3b and 3c show the pressure contours on the upper and lower wing surface, respectively. Because of large size of mesh, no attempt has been made using matrix LU-SGS and GMRES+LU-SGS methods to compute e



f

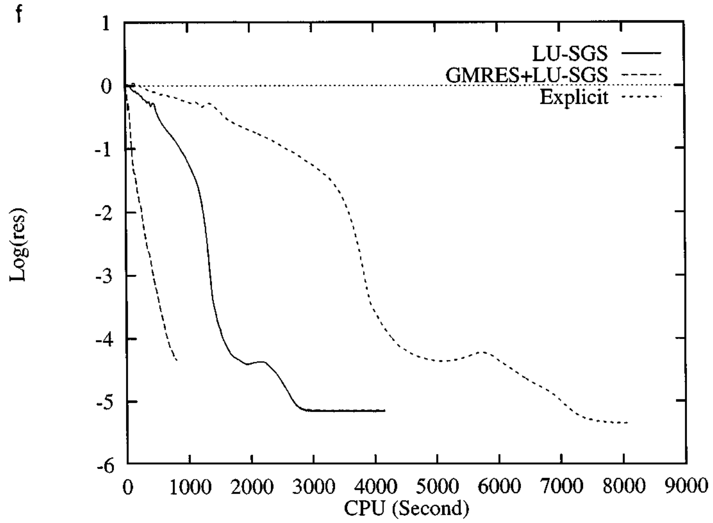


FIG. 4. (e) Residual convergence history versus time steps for supersonic inlet configuration using different schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit method. (f) Residual convergence history versus CPU time for supersonic inlet configuration using different schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit method.

this problem. Figures 3d and 3e display a comparison of convergence histories between a matrix-free LU-SGS scheme and a matrix-free GMRES+LU-SGS scheme versus time steps and CPU time, respectively. Again, the GMRES+LU-SGS method provides faster convergence than the LU-SGS method. Figures 3f and 3g illustrate the convergence history using different CFL numbers of GMRES+LU-SGS and LU-SGS methods, respectively. Again, we can see that, although the LU-SGS method is stable for a very large CFL number, the convergence history is almost indistinguishable for all the three CFL numbers used. However, a substantial gain can be achieved by using a larger CFL number for GMRES+ LU-SGS method, although no significant difference can be observed once the CFL number is large enough.

D. *Supersonic Duct Flow*

This internal inviscid supersonic flow case, taken from Nakahashi and Saito [20], represents part of a scramjet intake. The inlet Mach number is 3. The total length of the device is $l = 8.0$, and the element size was set uniformly throughout the domain to $\delta = 0.03$. The mesh shown in Fig. 4a consists of 542,895 elements, 106,285 grid points, and 30,094 boundary points. The computed density, Mach numbers, and pressure contours are shown in Figs. 4b,

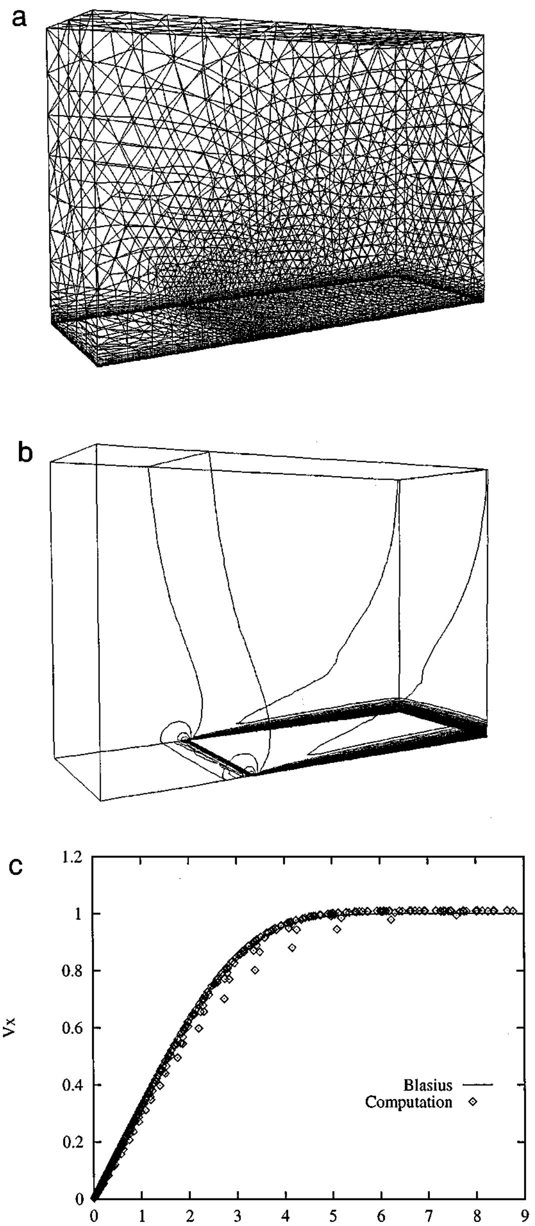
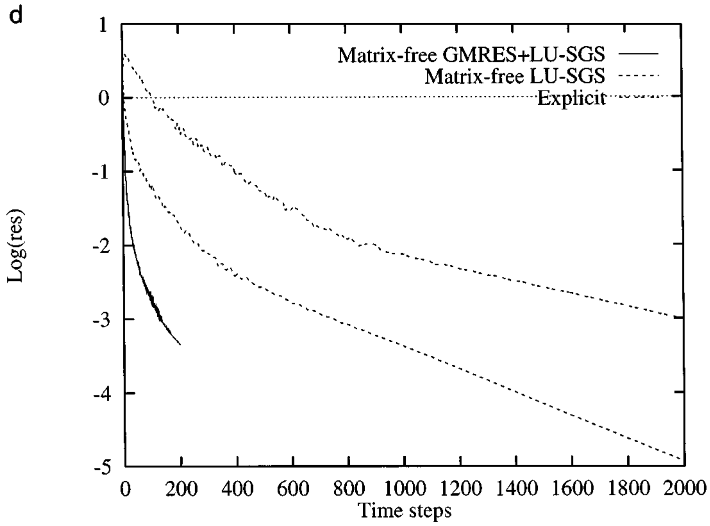


FIG. 5. (a) Surface mesh used for flat plate configuration (nelem = 81,885, npoin = 15,694, nboun = 3,774). (b) Computed Mach number contours for flat plate at $M_\infty = 0.4$, $\alpha = 0.0$, and Re = 10,000. (c) Computed boundary layer velocity profile over a flat plate at $M_\infty = 0.4$, $\alpha = 0.0$, and Re = 10,000.



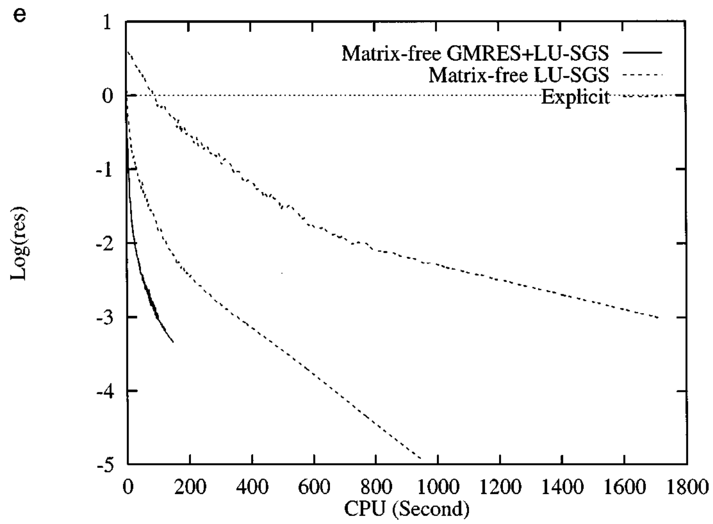


FIG. 5. (d) Residual convergence history versus time steps for flat plate using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (e) Residual convergence history versus cpu time for flat plate using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

4c, and 4d, respectively. Figures 4e and 4f illustrate the convergence history among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is about eight time faster than the explicit method for this particular problem.

E. *Laminar Flow Past a Flat Plate*

In this test case, Blasius boundary layers are computed for a flat plate at a Mach number of 0.4 and a chord Reynolds number of 10,000. The computational domain is considered from $x = -0.5$ to $x = 1$, $y = 0$ to $y = 1$, and $z = 0$ to $z = 0.5$, where the plate starts at $x = 0$.

The mesh used in the computation is shown in Fig. 5a. It contains 81,885 elements, 15,694 points, and 3774 boundary points. The computed Mach number contours in the flow field are depicted in Fig. 5b, where the development of a boundary layer can be clearly observed. Figure 5c shows the comparison of the Blasius velocity profile and the computed velocity profiles as scaled by the Blasius similarity law for all boundary layer points. The Blasius velocity profile is almost identically matched by all data points with the exception of a

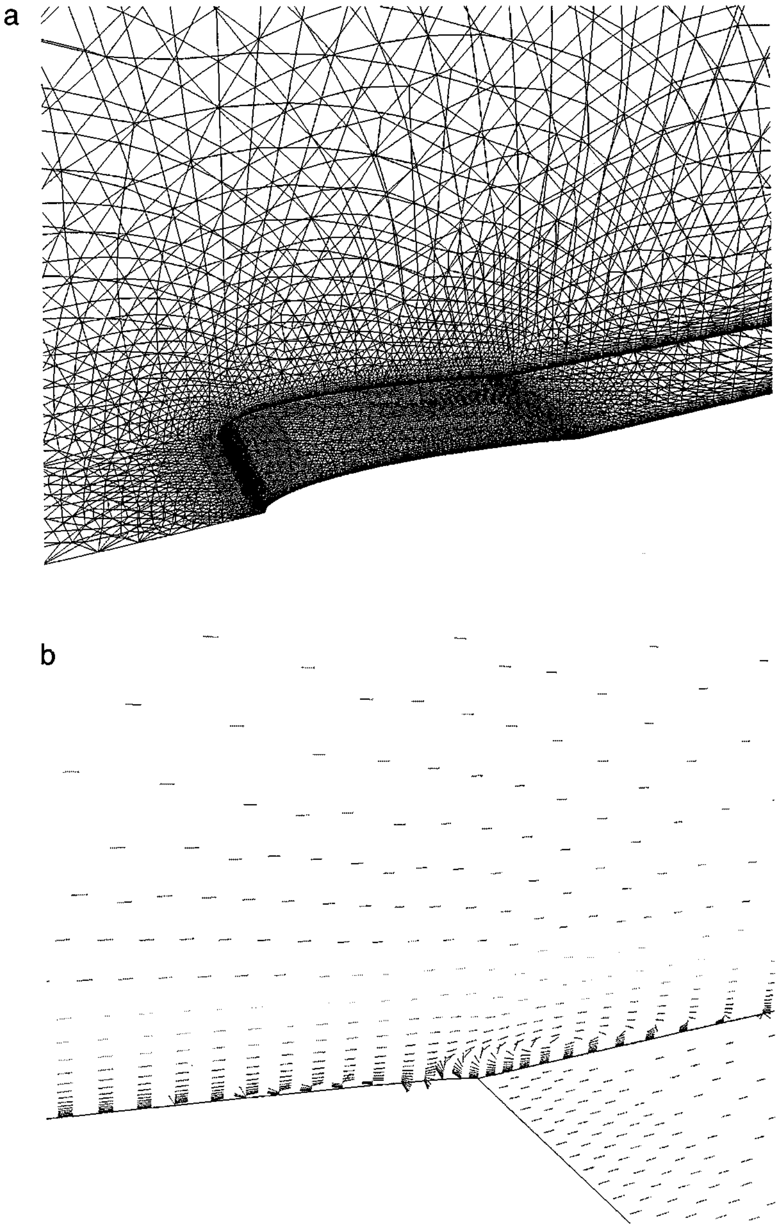


FIG. 6. (a) Surface mesh used for NACA0012 airfoil configuration (nelem = 697,655, npoin = 128,448, nboun = 22,925). (b) Computed velocity vector distribution near leading edge of airfoil at $M_\infty = 0.5$, $\alpha = 0.0$, and Re = 5,000.

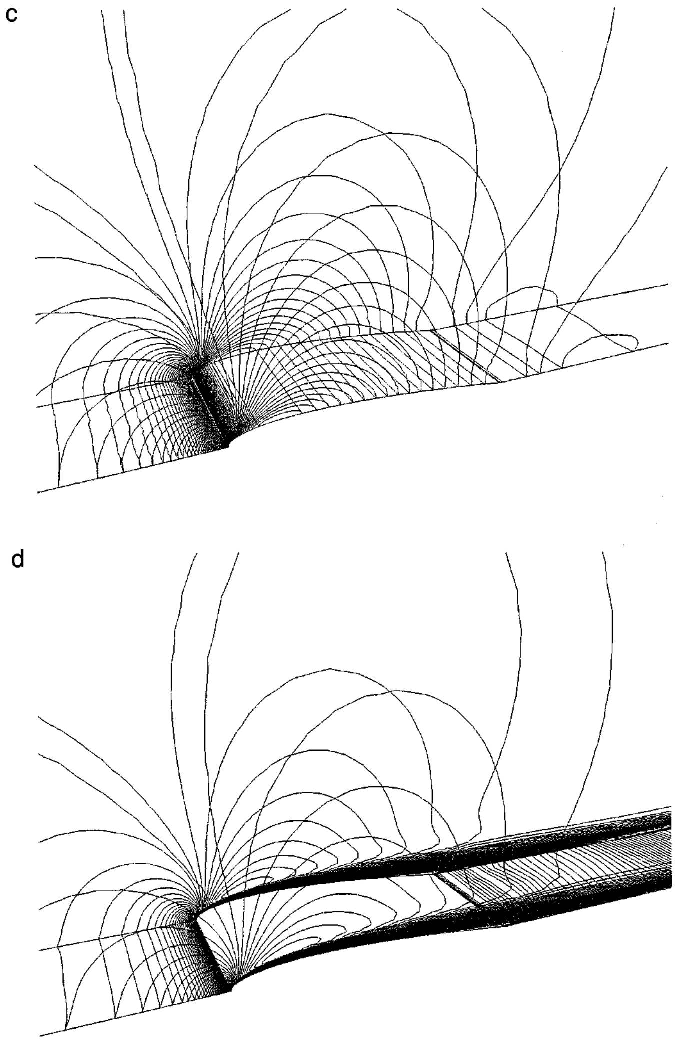


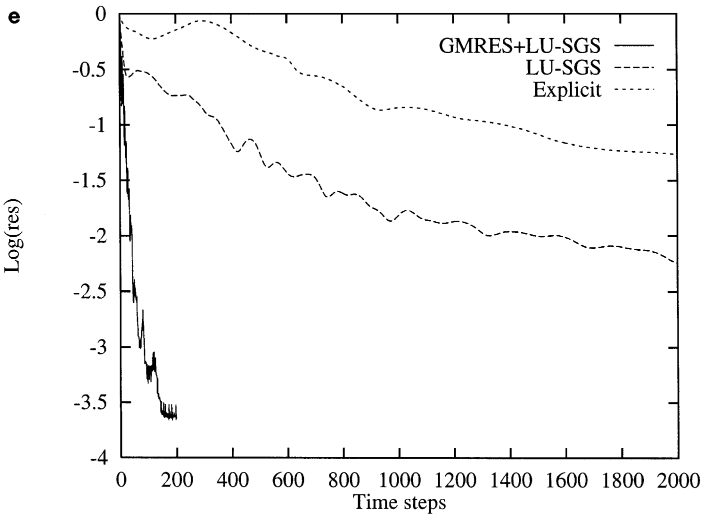
FIG. 6. (c) Computed pressure contours on the surface at $M_\infty = 0.5$, $\alpha = 0.0$, and $Re = 5{,}000$. (d) Computed Mach number contours on the surface at $M_\infty = 0.5$, $\alpha = 0.0$, and $Re = 5{,}000$.

few points near leading edge. The slight discrepancy for these points is attributed to the leading edge singularity. Figures 5d and 5e show a comparison of convergence histories among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is about 10 times faster than the explicit method for this particular problem.

F. *Laminar Flow over a NACA0012 Airfoil*

This test case involves a laminar flow past a NACA0012 airfoil at a Mach number of 0.5, an angle of attack of 0°, and a chord Reynolds number of 5000. This computation was performed to see the effectiveness of the present matrix-free GMRES+LU-SGS method for the solution of the Navier–Stokes equations. The mesh used in the computation is shown in Fig. 6a. It contains 697,655 elements, 128,488 grid points, and 22,925 boundary points.

e



f

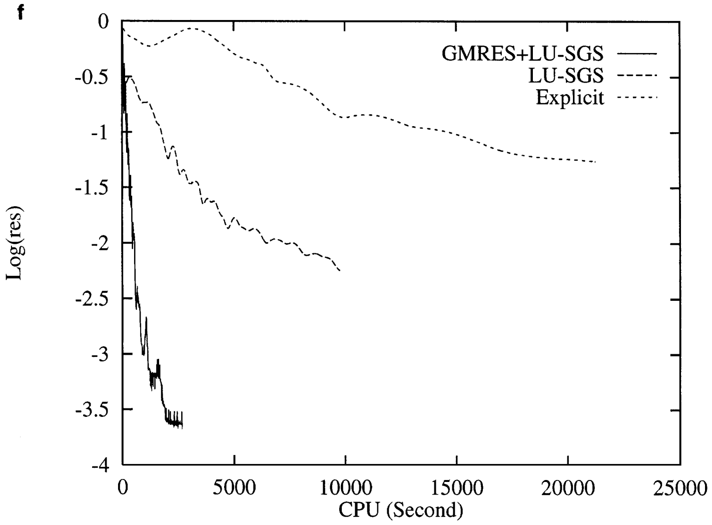


FIG. 6. (e) Residual convergence history versus time steps for NACA0012 airfoil using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS. (f) Residual convergence history versus cpu time for NACA0012 airfoil using different schemes: explicit, matrix-free LU-SGS, and matrix-free GMRES+LU-SGS.

The computed velocity vector distribution in the vicinity of the trailing edge of the airfoil is shown in Fig. 6b, where the separation and a small recirculation bubble can be clearly observed. The computed separation point is at 81.6% chord, which compares well to the one obtained by Mavriplis [22]. The computed pressure and Mach-number contours are shown in Figs. 6c and 6d, respectively. Figures 6e and 6f illustrate the convergence history among different numerical schemes: matrix-free LU-SGS, matrix-free GMRES+LU-SGS, and explicit methods, respectively. It indicates that the GMRES+LU-SGS method is far superior to the LU-SGS method. CPU time comparison shows that the GMRES+LU-SGS method is more than two orders of magnitude faster than the explicit method for this particular problem. The effectiveness of the present matrix-free GMRES+LU-SGS method for the solution of the Navier–Stokes equations is clearly demonstrated in this example.

6. CONCLUSIONS

A matrix-free implicit method has been developed to solve the three-dimensional Navier–Stokes and Euler equations on unstructured meshes. The developed method has been used to compute the compressible flows around 3D complex aerodynamic configurations for a wide range of flow conditions from subsonic to supersonic. The numerical results obtained indicate that the use of the GMRES+LU-SGS method leads to a significant increase in performance over the best current implicit methods, the GMRES+ILU and the LU-SGS methods, while maintaining memory requirements that are competitive with its explicit counterpart. In comparison to the explicit method, we demonstrate an overall speedup factor from eight to more than one order of magnitude for all test cases. The GMRES+LU-SGS method has also been extended and applied successfully to solve the unsteady Euler and Navier–Stokes equations and will be reported in a later paper. The current work is to extend the present GMRES+LU-SGS method for turbulent flow problems.

ACKNOWLEDGMENTS

This research was sponsored by the Defense Special Weapon Agency. Dr. Michael E. Giltrud served as the technical program monitor. Partial funding for the last author was also provided by the Air Force Office of Scientific Research. Dr. Leonidas Sakell served as the technical monitor.

REFERENCES

1. B. Stoufflet, Implicit finite element methods for the Euler equations, in *Numerical Methods for the Euler Equations of Fluid Dynamics*, edited by F. Angrand (SIAM, Philadelphia, 1985).
2. J. T. Batina, Implicit flux-split Euler schemes for unsteady aerodynamic analysis involving unstructured dynamic meshes, *AIAA J.* **29**(11), (1991).
3. V. Venkatakrishnan and D. J. Mavriplis, Implicit solvers for unstructured meshes, *J. Comput. Phys.* **105**, 83 (1993).
4. D. D. Knight, *A Fully Implicit Navier–Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting*, AIAA Paper 93-0875, 1993.
5. D. L. Whitaker, *Three-dimensional Unstructured Grid Euler Computations Using a Fully-Implicit, Upwind Method*, AIAA Paper 93-3337, 1993.
6. H. Luo, J. D. Baum, R. Löhner, and J. Cabello, *Implicit Schemes and Boundary Conditions for Compressible Flows on Unstructured Meshes*, AIAA Paper 94-0816, 1994.
7. T. J. Barth and S. W. Linton, *An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation*, AIAA Paper 95-0221, 1995.
8. Y. Saad, Iterative methods for sparse linear systems, on unstructured meshes, 1996.
9. Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.* **7**(3), 89 (1988).
10. A. Jameson and S. Yoon, Lower-upper implicit schemes with multiple grids for the Euler equations, *AIAA J.* **25**(7), (1987).
11. M. Soetrisno, S. T. Imlay, and D. W. Roberts, *A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids*, AIAA Paper 94-0617, 1994.
12. I. Men'shov and Y. Nakamura, An implicit advection upwind splitting scheme for hypersonic air flows in thermochemical nonequilibrium, in *6th Int. Sympos. on CFD, 1995*.
13. D. Sharov and K. Nakahashi, *Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier–Stokes Computations*, AIAA Paper 97-2102, 1997.
14. M. S. Liou, *Progress towards an Improved CFD Method: AUSM+*, AIAA Paper 95-1701, Jun. 1995.

15. B. Van Leer, Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme, *J. Comput. Phys.* **14**, 361 (1974).

16. P. L. Roe, Approximate Riemann solvers, parameter vectors and difference schemes, *J. Comput. Phys.* **43**, 357 (1981).

17. J. D. Löhner and P. Parikh, Three-dimensional grid generation by the advancing front method, *Int. J. Numer. Methods Fluids* **8**, 1135 (1988).

18. V. Schmitt and F. Charpin, Pressure distributions on the ONERA M6-wing at transonic Mach numbers, *Experiment Data Base for Computer Program Assessment*, AGARD AR-138, 1979.

19. E. R. IIeim, *CFD Wing/Pylon/Finned Store Mutual Interference Wind Tunnel Experiment*, AEDC-TSR-91-P4, Arnold Engineering Development Center, Arnold AFB, TN, Jan. 1991.

20. K. Nakahashi and E. Saitoh, *Space-Marching Method on Unstructured Grid for Supersonic Flows with Embedded Subsonic Regions*, AIAA-96-0418, 1996.

21. H. Luo, J. D. Baum, and R. Löhner, An edge-based upwind finite element scheme for the Euler equations, *AIAA J.* **32**(6), (1994).

22. D. J. Mavriplis and A. Jameson, Multigrid solution of the Navier–Stokes equations on triangular meshes, *AIAA J.* **28**(8), (1990).