AIAA JOURNAL
Vol. 38, No. 12, December 2000

Fast, Block Lower-Upper Symmetric Gauss–Seidel Scheme for Arbitrary Grids

R. F. Chen* and Z. J. Wang†
CFD Research Corporation, Huntsville, Alabama 35805

A block implicit lower-upper symmetric Gauss–Seidel (LU-SGS) approximate factorization scheme is developed and implemented for unstructured grids of arbitrary topology including viscous adaptive Cartesian grids. CPU times and memory requirements for the block LU-SGS (BLU-SGS), the original LU-SGS, and a fully implicit scheme (FIS) with a preconditioned conjugate gradient squared solver for several representative test examples are compared. Computational results showed that the block LU-SGS scheme requires about 20–30% more memory than the original LU-SGS but converges many times faster. The BLU-SGS scheme has a convergence rate competitive to and in many cases faster than the FIS while requiring much less memory.

Introduction

THE difficulty in generating structured grids and the desire to compute flows over complex geometries spawned a surge of activities in the area of unstructured grids during the last decade. Unstructured grids provide considerable flexibility in tackling complex geometries and for adapting the computational grids according to flow features. Types of unstructured grids include classical triangular or tetrahedral grids,[1–5] quadrilateral or hexahedral grids,[6] prismatic grids,[7] and mixed grids.[8] Tetrahedral grids are the easiest to generate. However, experiences have indicated that they are not as efficient and accurate as prismatic or hexahedral grids for viscous boundary layers.[9] On the other hand, prismatic and hexahedral grids are more difficult to generate than tetrahedral grids. Many computational fluid dynamics (CFD) researchers have come to the conclusion that mixed grids (or hybrid grids) are the way to go. For example, a hybrid tetrahedral/prismatic grid approach[10,11] was successfully demonstrated for complex geometries. One disadvantage of tetrahedral grids is that tetrahedra are not as efficient as Cartesian cells in filling three-dimensional space given a certain grid resolution. This can be easily understood with the fact that at least five tetrahedra are required to fill a cube without adding any new grid points. Therefore, it seems that the most appealing grid topology is a hybrid of Cartesian and viscous layer grids. One example is the viscous Cartesian grid method developed by Wang[12] and Wang et al.[13] Coupled with anisotropic grid adaptations in both the Cartesian and viscous-layer grid regions, the viscous Cartesian grid method is capable of achieving grid-independent turbulent flow solutions through solution-based grid adaptations.[14] This grid-generation methodology thus has the potential of fully automating the grid generation task for complex geometries.

To handle a viscous Cartesian grid, a flow solver capable of handling unstructured grids of arbitrary topology needs to be developed. Furthermore, an implicit time-integration scheme is highly desired for improved efficiency. Many implicit schemes have been developed and applied successfully to unstructured grids to accelerate convergence to steady state.[15–19] Among them, the most commonly used technique is to linearize the implicit operator and form a linearized system of equations whose left-hand side corresponds to a lower-order approximation of the spatial discretization of the right-hand side. The solution of the flowfield is then advanced one time step by approximately solving the resulting large sparse linear system. Lower-order approximations are used on the left-hand side because of storage considerations, computational complexity, and the fact that the resulting lower-order linear system is better conditioned than the higher-order counterpart. The mismatch between the left-hand-side matrices and the right-hand-side residual, however, results in a suboptimum convergence rate to steady state. Iterative methods such as generalized minimum residual (GMRES) and conjugate gradient squared (CGS) with an appropriate preconditioner are often used to approximately solve the sparse linear system because of the enormous computational cost and the large memory requirement of direct methods. The memory requirements per grid cell for such implicit technique depend on the discretization stencil. For unstructured grids with arbitrary polyhedral cells, the required memory can be enormous as a result of large stencils used compared to structured grids. Recently, a matrix-free Newton–Krylov method is attracting attentions because it does not need to form the matrices explicitly. Large matrices, however, are still needed for preconditioning purposes.[20,21]

Another very attractive implicit scheme, the implicit lower-upper symmetric Gauss–Seidel (LU-SGS) approximate factorization scheme, which was originally developed for structured grids by Jameson and Yoon[22] has been extended and applied to hybrid structured/unstructured grids and tetrahedra/prism unstructured grids.[23,24] Unstructured-grid-based LU-SGS schemes have demonstrated performance similar to that on structured grids. With LU-SGS a special first-order approximation is employed in linearizing the left-hand side resulting in the reduction of the block diagonal matrices to diagonal matrices. As a result, LU-SGS does not require any extra memory compared to explicit methods and is free from any matrix inversion. All of the off-diagonal matrices still contribute to the implicit operator through one forward and one backward sweep of a Gauss–Seidel iteration, thus drastically improving efficiency over an explicit scheme. The special first-order approximation used in deriving LU-SGS does degrade convergence rate, especially after several orders of convergence. As is indicated in the numerical examples in this study, LU-SGS is not competitive to a fully implicit scheme (FIS) in term of CPU time.

In this paper an improved block LU-SGS (BLU-SGS) scheme is developed. BLU-SGS is capable of achieving comparable convergence rate with FIS, but requires much less memory than FIS. The idea is to retain the block diagonal matrices but employ LU-SGS-like backward and forward Gauss–Seidel iterations. A similar approach for structured grids was reported in Ref. 25. The present paper seems to represent the first such extension to unstructured arbitrary grids. There are also important differences in how the off-diagonal matrices contribute to the implicit operator between the method presented in Ref. 25 and the present paper. In the following sections the implicit finite volume discretization of the Navier–Stokes equations is given first. Then the LU-SGS scheme on a fully unstructured grids is presented, followed by the derivation of the BLU-SGS scheme. For the sake of completeness, FIS is also briefly described. Several

Presented as Paper 99-0935 at the AIAA 37th Aerospace Sciences Meeting, Reno, NV, 11–14 January 1999; received 12 Feburary 1999; revision received 17 April 2000; accepted for publication 28 April 2000.

*Senior Engineer, 215 Wynn Drive. Member AIAA.
†Group Leader; currently at Department of Mechanical Engineering, 2555 Engineering Building, Michigan State University, East Lansing, MI 48824. Member AIAA.

demonstration cases including both inviscid and viscous flow problems using viscous Cartesian grids are presented to illustrate the performance of BLU-SGS. Finally conclusions from the study are summarized.

Implicit Finite Volume Discretization on Unstructured Grids

The governing equations for compressible viscous flow can be written in integral form over a control volume V as

$$\int_V \frac{\partial \boldsymbol{Q}}{\partial t}\,\mathrm{d}V + \int_{\partial V} (\boldsymbol{F} - \boldsymbol{F}_v)\,\mathrm{d}S = 0 \tag{1}$$

where $\boldsymbol{Q}$ is the vector of conserved variables and $\boldsymbol{F}$ and $\boldsymbol{F}_v$ comprise the inviscid and viscous flux vectors, respectively. Spatial discretization of Eq. (1) in grid cell i gives

$$V_i \frac{\partial \boldsymbol{Q}_i}{\partial t} + \sum_{j \in N(i)} (\tilde{\boldsymbol{F}}_{ij} - \tilde{\boldsymbol{F}}_{v,ij}) S_{ij} = 0 \tag{2}$$

where V_i is the volume of cell i, $\boldsymbol{Q}_i$ is the cell averaged vector of conserved variables, $N(i)$ is the set of face neighbor cells of cell i, S_{ij} is the area of the cell face shared by cell i and cell j, and $\tilde{\boldsymbol{F}}$ and $\tilde{\boldsymbol{F}}_v$ are the second-order-accurate numerical inviscid and viscous flux vectors at the face, respectively. Equation (2) is still true for $\boldsymbol{Q}_i$ evaluated at the cell center of cell i by neglecting a second-order temporal term. As a matter of fact, time accuracy is not important because only steady-state problems are considered in this study. Hereafter let $\boldsymbol{Q}_i$ be the vector of conserved variables evaluated at the cell center of cell i. The numerical inviscid flux vector $\tilde{\boldsymbol{F}}_{ij}$ is computed using Roe's approximate Riemann solver[26] with reconstructed state variables at both sides of the face. A least-squares linear reconstruction scheme of the primitive variables is used. Venkatakrishnan's limiter[27] is employed to make the scheme monotone. Other details of the flow solver are contained in Ref. 28. Applying the backward Euler scheme for time integration in Eq. (2), we obtain

$$\frac{V_i}{\Delta t_i}\left(\boldsymbol{Q}_i^{n+1} - \boldsymbol{Q}_i^n\right) + \sum_{j \in N(i)} \left(\tilde{\boldsymbol{F}}_{ij}^{n+1} - \tilde{\boldsymbol{F}}_{v,ij}^{n+1}\right) S_{ij} = 0 \tag{3}$$

where Δt_i is the local time step at cell i. Equation (3) can be rewritten in the following delta form:

$$\frac{V_i}{\Delta t_i}\Delta \boldsymbol{Q}_i^n + \sum_{j \in N(i)} \left(\Delta \tilde{\boldsymbol{F}}_{ij}^n - \Delta \tilde{\boldsymbol{F}}_{v,ij}^n\right) S_{ij} = \mathrm{Res}_i^n \tag{4}$$

where Δ is the forward difference in time, for example,

$$\Delta \boldsymbol{Q}_i^n = \boldsymbol{Q}_i^{n+1} - \boldsymbol{Q}_i^n \tag{5}$$

and the right-hand side residual can be written as

$$\mathrm{Res}_i^n = -\sum_{j \in N(i)} \left(\tilde{\boldsymbol{F}}_{ij}^n - \tilde{\boldsymbol{F}}_{v,ij}^n\right) S_{ij} \tag{6}$$

In practice, $\Delta\tilde{\boldsymbol{F}}$ and $\Delta\tilde{\boldsymbol{F}}_v$ on the left-hand side of Eq. (4) are usually approximated by their first-order counterparts $\Delta\bar{\boldsymbol{F}}$ and $\Delta\bar{\boldsymbol{F}}_v$, respectively, and Eq. (4) becomes

$$\frac{V_i}{\Delta t_i}\Delta \boldsymbol{Q}_i^n + \sum_{j \in N(i)} \left(\Delta \bar{\boldsymbol{F}}_{ij}^n - \Delta \bar{\boldsymbol{F}}_{v,ij}^n\right) S_{ij} = \mathrm{Res}_i^n \tag{7}$$

Note that

$$\begin{aligned}\Delta \tilde{\boldsymbol{F}}_{ij}^n - \Delta \tilde{\boldsymbol{F}}_{v,ij}^n &= \left[\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^{n+1}, \boldsymbol{Q}_j^{n+1}\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right)\right] \\ &+ \left[\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\right] - \left[\tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^{n+1}, \boldsymbol{Q}_j^{n+1}\right)\right. \\ &\left. - \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right)\right] - \left[\tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right) - \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\right]\end{aligned} \tag{8}$$

Linearizing the first and the third terms on the right-hand side of Eq. (8), we have

$$\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^{n+1}, \boldsymbol{Q}_j^{n+1}\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right) \approx \frac{\partial \bar{\boldsymbol{F}}_{ij}}{\partial \boldsymbol{Q}_i}\Delta \boldsymbol{Q}_i^n \tag{9}$$

$$\tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^{n+1}, \boldsymbol{Q}_j^{n+1}\right) - \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^{n+1}\right) \approx \frac{\partial \bar{\boldsymbol{F}}_{v,ij}}{\partial \boldsymbol{Q}_i}\Delta \boldsymbol{Q}_i^n \tag{10}$$

Substituting Eqs. (8–10) back to Eq. (7), we obtain

$$\begin{aligned}&\boldsymbol{D}\Delta \boldsymbol{Q}_i^n + \sum_{j \in N(i)} \left[\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta \boldsymbol{Q}_j^n\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\right] S_{ij} \\ &\quad - \sum_{j \in N(i)} \left[\tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta \boldsymbol{Q}_j^n\right) - \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\right] S_{ij} = \mathrm{Res}_i^n\end{aligned} \tag{11}$$

where matrix $\boldsymbol{D}$ is given by

$$\boldsymbol{D} = \frac{V_i}{\Delta t_i}\boldsymbol{I} + \sum_{j \in N(i)} \left(\frac{\partial \tilde{\boldsymbol{F}}_{ij}}{\partial \boldsymbol{Q}_i} - \frac{\partial \tilde{\boldsymbol{F}}_{v,ij}}{\partial \boldsymbol{Q}_i}\right) S_{ij} \tag{12}$$

and $\boldsymbol{I}$ is the identity matrix.

Original LU-SGS

In the original LU-SGS approach the first-order numerical flux vectors in the left-hand side of Eq. (11) are chosen as

$$\tilde{\boldsymbol{F}}_{ij} - \tilde{\boldsymbol{F}}_{v,ij} = \tfrac{1}{2}[\boldsymbol{F}_i + \boldsymbol{F}_j - \lambda_{ij}(\boldsymbol{Q}_j - \boldsymbol{Q}_i)] \tag{13}$$

where λ_{ij} is the spectral radius of the flux Jacobian matrix at the cell face:

$$\lambda_{ij} = |\boldsymbol{V} \cdot \boldsymbol{n}_{ij}| + a + \frac{2(\mu + \mu_t)}{\rho\,|\boldsymbol{n}_{ij} \cdot (\boldsymbol{r}_j - \boldsymbol{r}_i)|} \tag{14}$$

where $\boldsymbol{n}_{ij}$ is the normal of the cell face pointing from cell i to cell j; $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are the position vectors of cell centers of cell i and cell j, respectively; $\boldsymbol{V}$ is the velocity vector; ρ is the density; a is the speed of sound; μ and μ_t are kinematic and turbulent viscosities, respectively. Because each control volume is closed, for cell i we have

$$\sum_{j \in N(i)} \frac{\partial \boldsymbol{F}_i(n_{ij})}{\partial \boldsymbol{Q}_i} S_{ij} = 0 \tag{15}$$

Substituting Eqs. (13) into Eq. (12) and using Eqs. (14) and (15), we obtain

$$\boldsymbol{D} = \left(\frac{V_i}{\Delta t} + \frac{1}{2}\sum_{j \in N(i)} \lambda_{ij} S_{ij}\right)\boldsymbol{I} \tag{16}$$

which is reduced to the identity matrix with a scale factor. Equation (11) becomes

$$\begin{aligned}&\left(\frac{V_i}{\Delta t} + \frac{1}{2}\sum_{j \in N(i)} \lambda_{ij} S_{ij}\right)\Delta \boldsymbol{Q}_i^n + \frac{1}{2}\sum_{j \in N(i)} \left\{\left[F\left(\boldsymbol{Q}_j^n + \Delta \boldsymbol{Q}_j^n\right)\right.\right. \\ &\quad \left.\left. - F\left(\boldsymbol{Q}_j^n\right)\right] - \lambda_{ij}\Delta \boldsymbol{Q}_j^n\right\} S_{ij} = \mathrm{Res}_i^n\end{aligned} \tag{17}$$

which is then solved using one sweep of symmetric Gauss–Seidel iteration as shown in the following.

Forward sweep:

$$\begin{aligned}&\boldsymbol{D}\Delta \boldsymbol{Q}_i^* = \mathrm{Res}_i^n - 0.5 \\ &\quad \times \sum_{j \in L(i)} \left\{\left[\boldsymbol{F}\left(\boldsymbol{Q}_j^n + \Delta \boldsymbol{Q}_j^*\right) - \boldsymbol{F}\left(\boldsymbol{Q}_j^n\right)\right] - \lambda_{ij}\Delta \boldsymbol{Q}_j^*\right\} S_{ij}\end{aligned} \tag{18}$$

Backward sweep:

$$\begin{aligned}&\Delta \boldsymbol{Q}_i = \Delta \boldsymbol{Q}_i^* - 0.5\boldsymbol{D}^{-1} \\ &\quad \times \sum_{j \in U(i)} \left\{\left[\boldsymbol{F}\left(\boldsymbol{Q}_j^n + \Delta \boldsymbol{Q}_j\right) - \boldsymbol{F}\left(\boldsymbol{Q}_j^n\right)\right] - \lambda_{ij}\Delta \boldsymbol{Q}_j\right\} S_{ij}\end{aligned} \tag{19}$$

where $L(i)$ and $U(i)$ represent the lower and upper neighbor cells of cell i according to the cell ordering.

Block LU-SGS

To improve the convergence rate of the original LU-SGS, we keep the diagonal block of the implicit system and then use forward and backward sweeps to include the implicit contributions from the off-diagonal blocks. The improved BLU-SGS scheme is designed to achieve faster convergence rate with a minimum increase in memory requirement. Therefore, the first-order numerical flux vectors, which are consistent with the second order flux vectors in the spatial discretization, are used on the left-hand side for BLU-SGS. In this study, for example, Roe's approximate Riemann solver with state variables at the cell centers on both sides of the face is used for the numerical inviscid flux vector on the left-hand side of Eq. (11). Equation (11) is then solved using symmetric Gauss–Seidel iterations with a multiple number of inner iterations to further boost convergence speed. The inner iteration depends on a prescribed convergence tolerance and a maximum number of sweeps.

To be more specific, given the solutions $\Delta \boldsymbol{Q}^{(k-1)}$ at sweep level $k-1$, we compute the solution at the kth sweep using the following algorithm.

Forward sweep:

$$\begin{aligned}\boldsymbol{D}\Delta\boldsymbol{Q}_i^* &+ \sum_{j\in L(i)} \Big[\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right) \\ &- \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*\right) + \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\Big] S_{ij} \\ &+ \sum_{j\in U(i)} \Big\{\tilde{\boldsymbol{F}}\left[\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^{(k-1)}\right] - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right) \\ &- \tilde{\boldsymbol{F}}_v\left[\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^{(k-1)}\right] + \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\Big\} S_{ij} = \mathrm{Res}_i^n \end{aligned} \tag{20}$$

Backward sweep:

$$\begin{aligned}\boldsymbol{D}\Delta\boldsymbol{Q}_i^{(k)} &+ \sum_{j\in L(i)} \Big[\tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*\right) - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right) \\ &- \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*\right) + \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\Big] S_{ij} \\ &+ \sum_{j\in U(i)} \Big\{\tilde{\boldsymbol{F}}\left[\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^{(k)}\right] - \tilde{\boldsymbol{F}}\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right) \\ &- \tilde{\boldsymbol{F}}_v\left[\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^{(k)}\right] + \tilde{\boldsymbol{F}}_v\left(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n\right)\Big\} S_{ij} = \mathrm{Res}_i^n \end{aligned} \tag{21}$$

where $\tilde{\boldsymbol{F}}(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n)$ is the first-order Roe's flux difference splitting and $\bar{\boldsymbol{F}}(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*) - \bar{\boldsymbol{F}}(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n)$ is reformulated so that it can be efficiently evaluated. In a similar approach presented in Ref. 25 for structured grids, these flux differences were further linearized. The viscous flux difference $\tilde{\boldsymbol{F}}_v(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n + \Delta\boldsymbol{Q}_j^*) - \tilde{\boldsymbol{F}}_v(\boldsymbol{Q}_i^n, \boldsymbol{Q}_j^n)$ is approximated by $J_v \Delta\boldsymbol{Q}_j^*$ with the following approximate viscous Jacobian matrix:

$$J_v = \frac{(\mu + \mu_t)}{|\boldsymbol{n}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)|} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3}n_x^2 + 1 & \frac{1}{3}n_x n_y & \frac{1}{3}n_x n_z & 0 \\ 0 & \frac{1}{3}n_x n_y & \frac{1}{3}n_y^2 + 1 & \frac{1}{3}n_y n_z & 0 \\ 0 & \frac{1}{3}n_x n_z & \frac{1}{3}n_y n_z & \frac{1}{3}n_z^2 + 1 & 0 \\ -\frac{pC_p}{\rho^2 PrR(\mu+\mu_t)} & \frac{1}{3}n_x V_n + u & \frac{1}{3}n_y V_n + v & \frac{1}{3}n_z V_n + w & \frac{C_p}{\rho PrR(\mu+\mu_t)} \end{bmatrix}$$

where C_p is the specific heat, Pr the turbulent Prandtl number, and R the gas constant. The initial state variables for the inner sweep are set to be same as those at the last time step, i.e., $\Delta\boldsymbol{Q}^{(0)} = 0$. The number of inner sweeps is controlled by the following conditions:

$$\frac{\left\|\Delta\boldsymbol{Q}^{(k)} - \Delta\boldsymbol{Q}^{(k-1)}\right\|}{\left\|\Delta\boldsymbol{Q}^{(1)}\right\|} \le \varepsilon \tag{22}$$

$$k \le K_{\max} \tag{23}$$

where ε is the convergence tolerance and $K_{\max}$ a prescribed maximum number of inner sweeps.

The flow variables at time level $n+1$ are then updated by

$$\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^n + \Delta\boldsymbol{Q}^{(k)} \tag{24}$$

In this approach matrix $\boldsymbol{D}$ is no longer an identity matrix with a scale factor. Instead it is a 5×5 matrix in three dimensions and a 4×4 matrix in two dimensions. Therefore extra memory is required to store the block diagonal matrix in the BLU-SGS approach. Equations (20) and (21) are solved efficiently with an exact LU decomposition method.

Fully Implicit Scheme

In a fully implicit approach the off-diagonal matrices are also included on the left-hand side of Eq. (11). After linearization Eq. (11) reduces to the following large sparse linear system:

$$\boldsymbol{D}\Delta\boldsymbol{Q}_i^n + \sum_{j\in N(i)} \left(\frac{\partial\tilde{\boldsymbol{F}}_{ij}}{\partial\boldsymbol{Q}_j} - \frac{\partial\tilde{\boldsymbol{F}}_{v,ij}}{\partial\boldsymbol{Q}_j}\right) S_{ij}\Delta\boldsymbol{Q}_j^n = \mathrm{Res}_i^n \tag{25}$$

The linear system is usually solved with an iterative method such as GMRES or CGS with proper preconditioning. The number of inner sweeps in solving Eq. (25) is also controlled by a prescribed convergence tolerance and a maximum number of sweeps. Equation (25) requires the storage of not only the diagonal block matrices, but also off-diagonal block matrices, which depend on the number of neighboring cells. Therefore the memory required by FIS is much more than that of the original LU-SGS and BLU-SGS depending on the local discretization stencil. In this study Eq. (25) is solved with a CGS solver with a block incomplete LU preconditioner.

Numerical Results

Several representative test cases are selected to compare the CPU times and memory requirements of LU-SGS, BLU-SGS, and FIS. CPU seconds in the test cases represent the actual CPU time on a Pentium III 450 machine running the Linux operating system. Details are presented next.

Transonic Flow over a NACA 0012 Airfoil

This simple inviscid transonic flow over the NACA 0012 airfoil with a freestream Mach number of 0.85 and an angle of attack 1 deg was chosen as the first test case to study the effects of the Courant–Friedrichs–Lewy (CFL) number and inner iteration control parameters on convergence characteristics and to compare the performance of LU-SGS, BLU-SGS, and FIS. Figure 1 shows an automatically generated adaptive Cartesian/quad unstructured grid around the NACA 0012 airfoil. The outer boundary is placed 250 chord lengths away. This mesh consists of 2703 nodes, 4887 faces, and 2184 cells. Several different CFL numbers were tested for LU-SGS, and the convergence histories are compared in Fig. 2. LU-SGS is insensitive to CFL once it is over a 100, and a CFL of infinity can be used right from the beginning, indicating that the diagonal dominance is very strong in LU-SGS. For BLU-SGS a maximum CFL of about 400 can be used for this case. A CFL of 800 drove the computation unstable. It is also necessary to ramp the CFL gradually from about five to the maximum CFL starting from the freestream condition. The convergence histories in terms of CPU seconds for

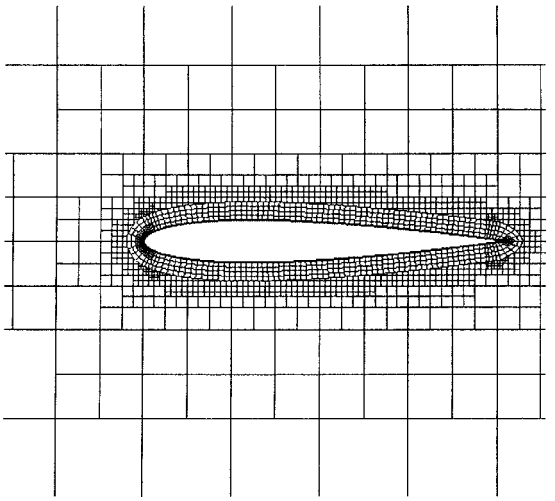


Fig. 1 Adaptive Cartesian/quad unstructured grid around the NACA 0012 airfoil.

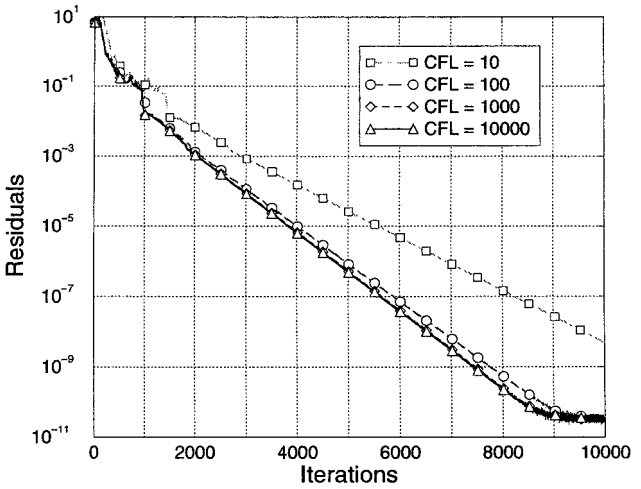


Fig. 2 Effect of CFL number on the convergence rate of LU-SGS.

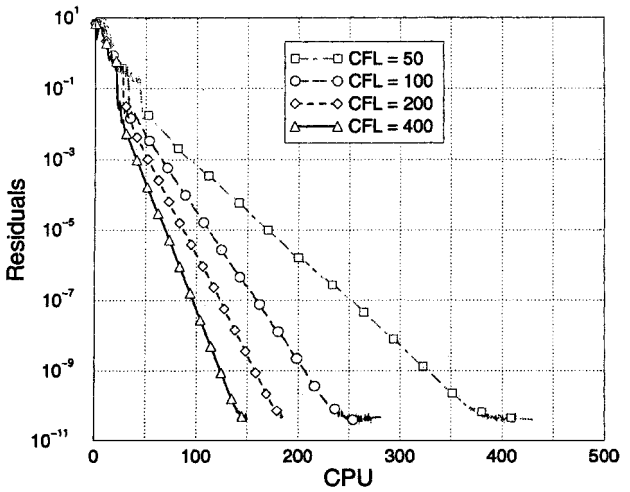


Fig. 3 Effect of CFL number on the convergence rate of BLU-SGS.

BLU-SGS with different CFL numbers are compared in Fig. 3. The inner iteration convergence tolerance ε and the maximum number of inner sweeps were set at 0.1 and 10. It is not surprising to see that the convergence rate is strongly dependent on the CFL number. The effect of the inner iteration control parameters on the convergence rate was also tested for BLU-SGS. In this test the CFL was increased linearly from 5 to 400 over 80 iterations with ε fixed at 0.01 and the maximum number of inner sweeps from 3 to 12. Inner sweeps of fewer than 3 diverged the computation. The convergence

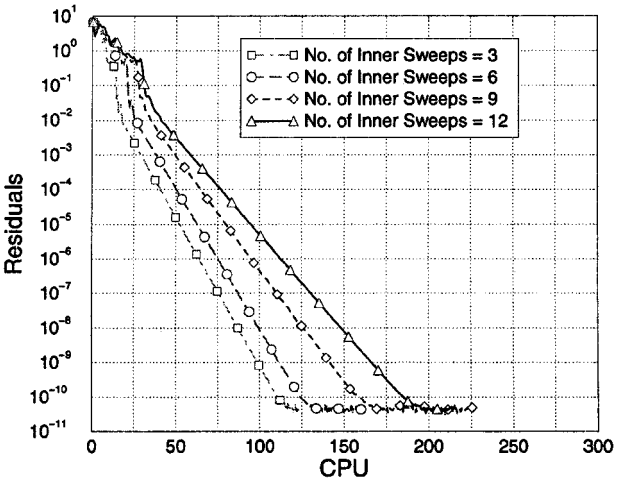


Fig. 4 Effect of number of inner iterations on the convergence rate of BLU-SGS.

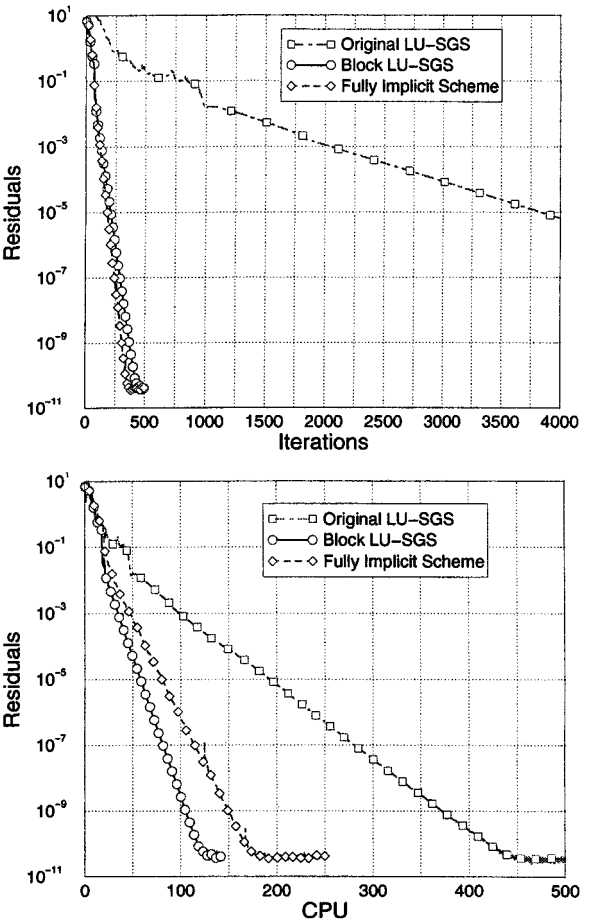


Fig. 5 Comparison of convergence characteristics between LU-SGS, BLU-SGS, and FIS in terms of number of iterations and CPU seconds on a Pentium III 450.

histories vs CPU seconds with different inner sweeps are shown in Fig. 4. The minimum number of inner sweeps gave the best CPU performance. However, it is common sense that a fewer number of inner sweeps usually results in less stable computation. Therefore an inner iteration number of 10 is usually chosen to yield robust computation with good CPU performance.

To compare LU-SGS, BLU-SGS, and FIS, we used the largest possible CFL numbers. The CFL number for BLU-SGS and FIS increases linearly from 5 to 400 over 80 iterations, with $\varepsilon = 0.1$ and $K_{\max} = 10$. The CFL for LU-SGS is essentially infinity (10^{40}). Figure 5 shows the convergence histories in terms of both the number

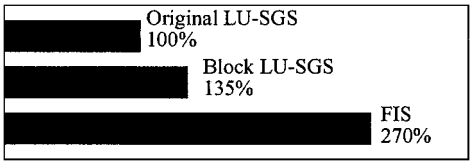


Fig. 6 Percentage comparison of memory requirements for the simulations of transonic flow over the NACA 0012 airfoil.

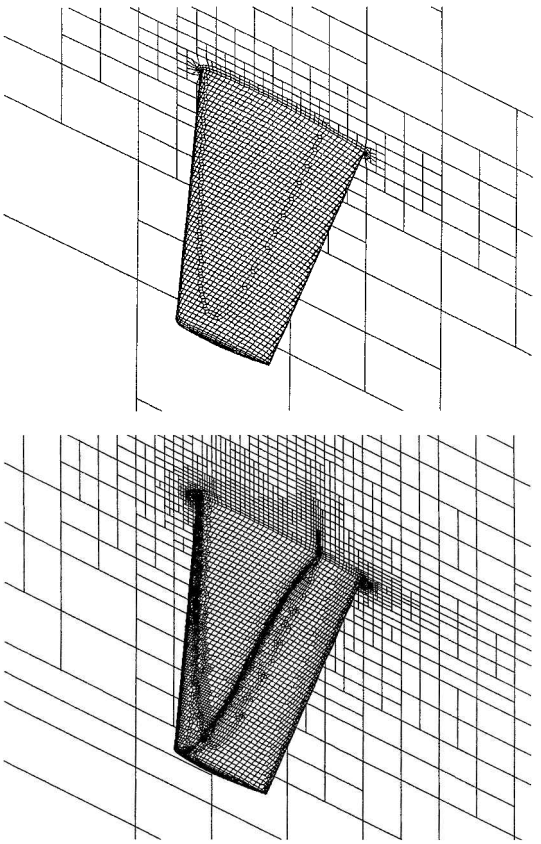


Fig. 7 Initial and level 3 anisotropic adaptive Cartesian grid for the ONERA M6 wing.

of iterations and CPU seconds. It is shown that BLU-SGS converges about 4.5 times as fast as LU-SGS in terms of CPU time. Although FIS converges to machine zero in fewer iterations than BLU-SGS, BLU-SGS is faster than FIS in terms of CPU time. Figure 6 shows the percentage comparison of total memory requirements. BLU-SGS requires about 35% more total memory than LU-SGS, but only takes half of the memory used by FIS for this two-dimensional case.

Transonic Flow over an ONERA M6 Wing

Transonic flow over an ONERA M6 wing configuration was selected to represent three-dimensional problems.[29] The M6 wing has a leading-edge sweep angle of 30 deg, an aspect ratio 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 was computed at a Mach number of 0.84, an angle of attack of 3.06 deg, and with inviscid flow assumption. A coarse viscous Cartesian grid was first generated and is shown in Fig. 7. The mesh consists of 14,141 cells and 47,904 faces. Three levels of solution-based grid adaptations were then performed after converged solutions were obtained on each grid. The adaptation criteria are pressure and Mach-number gradients. The level 3 viscous Cartesian grid is shown in the same figure. The grid has 88,296 cells and 300,573 faces. Anisotropic Cartesian cells were used to capture the flow features very efficiently. Convergence characteristics of LU-SGS, BLU-SGS, and FIS were then studied using this fine mesh. All of the simulations started from the freestream condition.

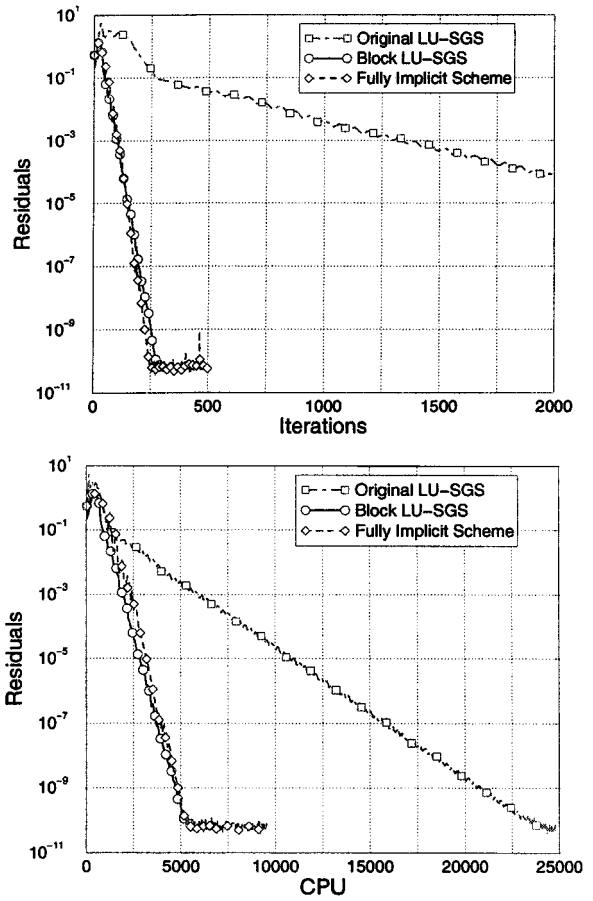


Fig. 8 Comparison of convergence characteristics between LU-SGS, BLU-SGS, and FIS in terms of number of iterations and CPU seconds for transonic flow over ONERA M6 wing.

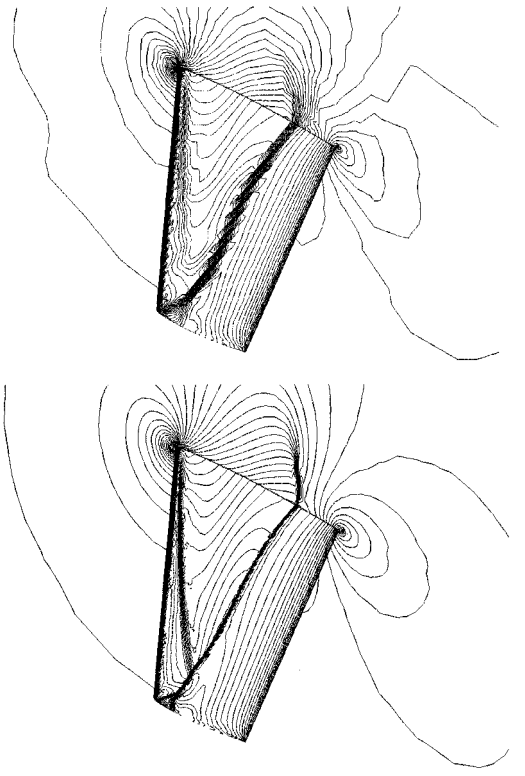


Fig. 9 Pressure contours on the initial and level 3 adaptive grid for ONERA M6 wing ($M = 0.84$ and $\alpha = 3.06$).

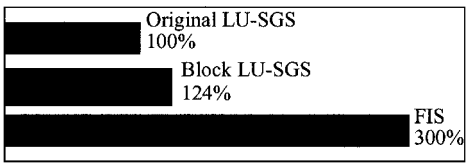


Fig. 10 Percentage comparison of memory requirements for transonic flow over ONERA M6 wing.

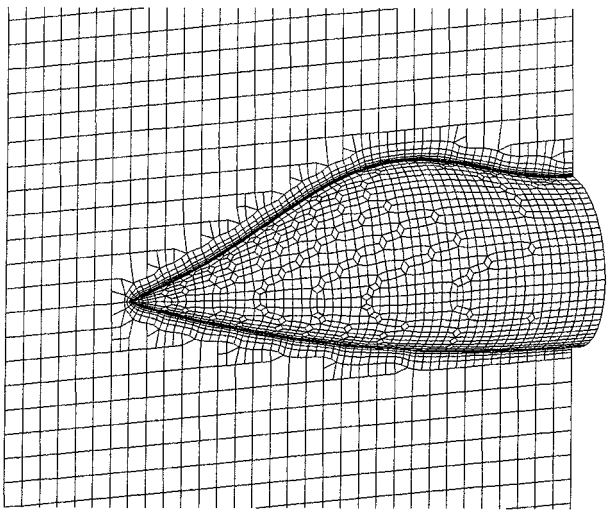


Fig. 11 Viscous Cartesian computational grid around the analytical forebody.

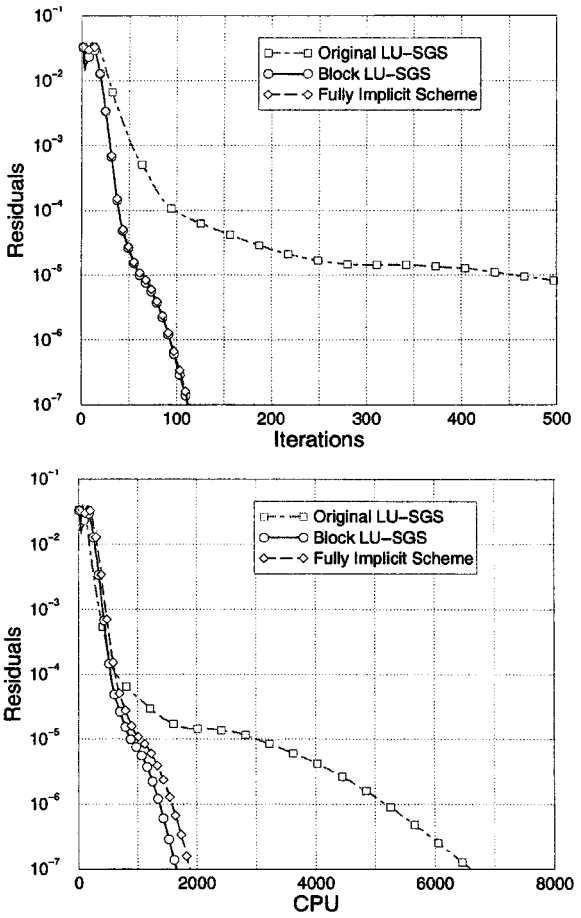


Fig. 12 Convergence histories in terms of iterations and CPU seconds for supersonic turbulent flow over the analytical forebody.

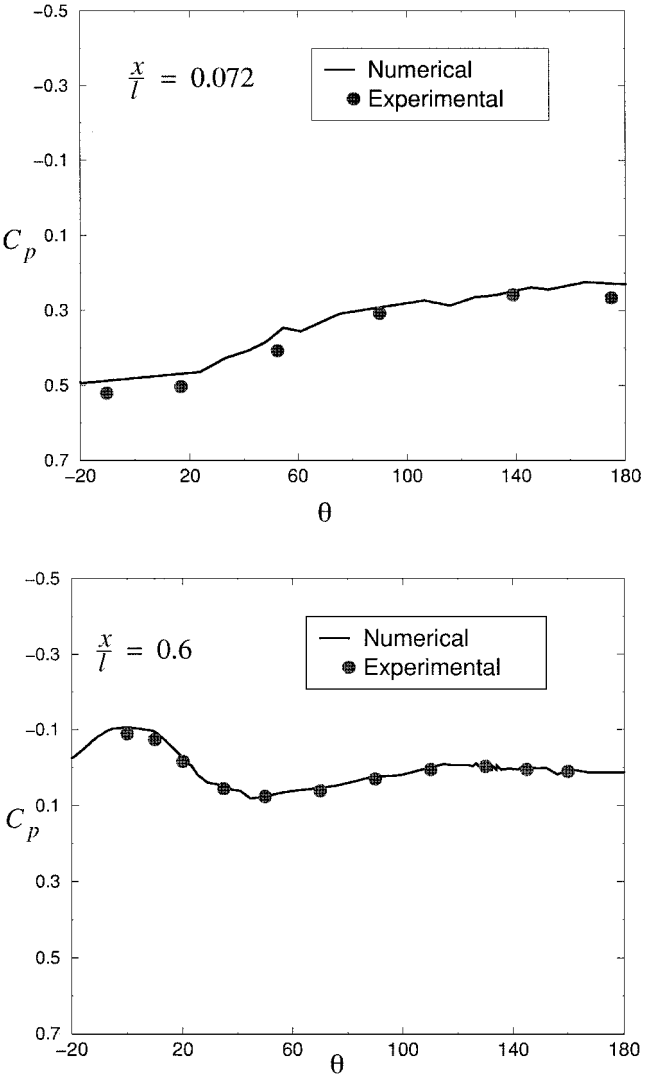
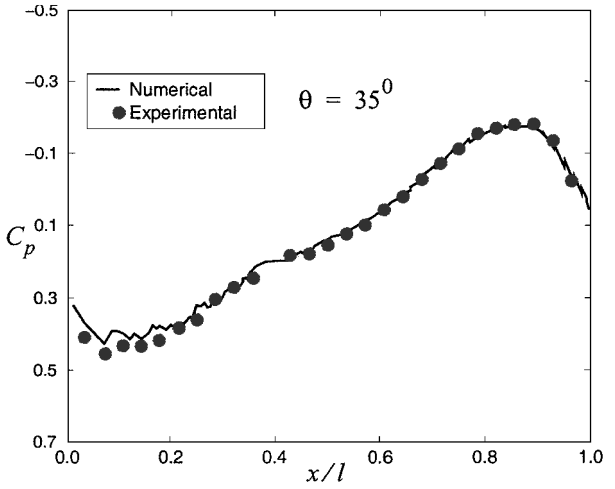


Fig. 13 Comparison of surface pressure coefficient in the azimuthal direction between computational and experimental data at different axial locations.

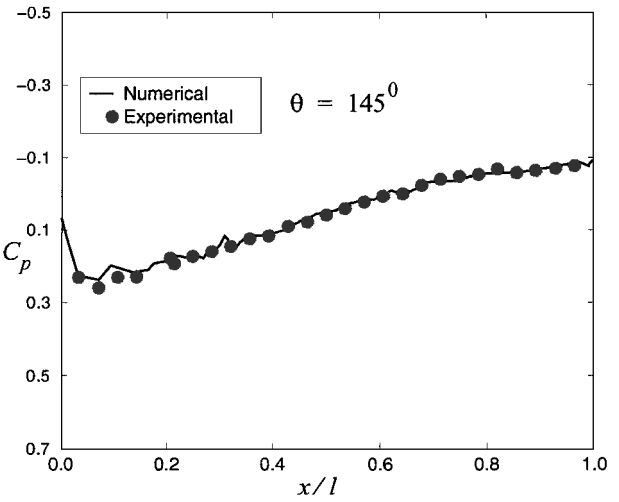
For LU-SGS the CFL was essential infinity (1.e40). For BLU-SGS and FIS the CFL number was increased from 10 to 100 over 50 iterations, with $\varepsilon = 0.1$ and $K_{\max} = 10$. The convergence histories in terms of both the number of iterations and CPU seconds are shown in Fig. 8. It is shown that BLU-SGS converges about five times as fast as LU-SGS in terms of CPU time. Although FIS converges to machine zero in fewer iterations than BLU-SGS, BLU-SGS is as fast as FIS in terms of CPU time. The pressure contours on the initial and fine grids are presented in Fig. 9. It is obvious that much better resolution was obtained through solution-based grid adaptations. The characteristic lamda shock wave was captured sharply on the fine mesh. Figure 10 shows the percentage comparison of total memory requirements. BLU-SGS requires about 24% more total memory than LU-SGS but takes less than half the memory used by FIS for this three-dimensional case.

Supersonic Flow over a Three-Dimensional Forebody

The case of supersonic flow over an analytical forebody was chosen to test BLU-SGS for viscous flow problems. The geometry and experimental data are described in Ref. 30 for an extensive set of flow conditions. The flow presented here was computed for the following conditions: freestream Mach number $M_\infty = 1.7$, angle of attack $\alpha = -5$ deg, angle of sideslip $\beta = -0.04$ deg, and the Reynolds number based on the freestream flow properties and body characteristic length $Re = 2.33 \times 10^6$. The flow was assumed fully turbulent in the simulation. The viscous Cartesian computational grid for the



θ = 35 deg



θ = 145 deg

Fig. 14 Comparison of surface pressure coefficient in the longitudinal direction between the computational and experimental data at different azimuthal angles.

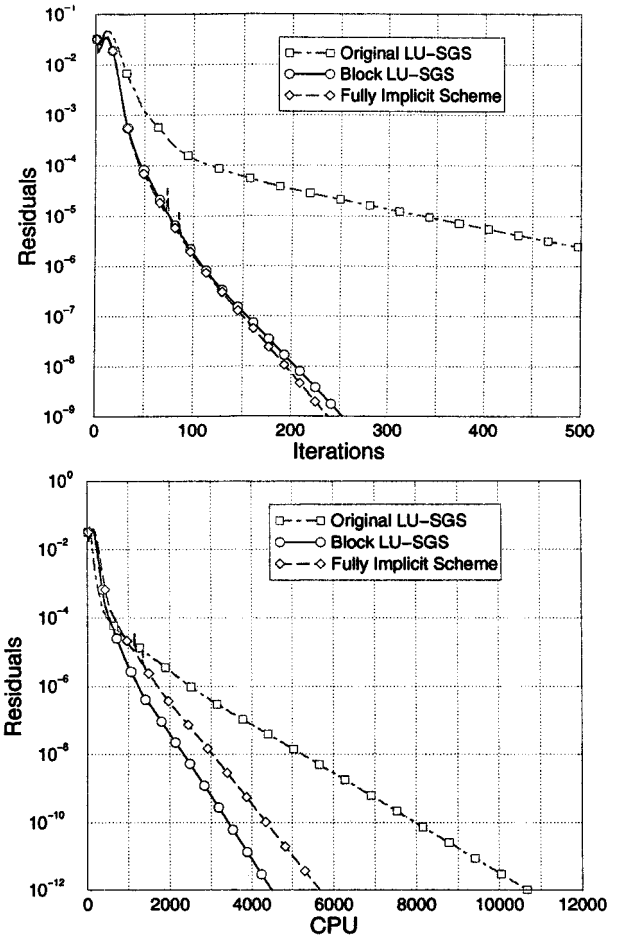


Fig. 15 Convergence histories in terms of iterations and CPU seconds for supersonic laminar flow over the analytical forebody.

forebody is shown in Fig. 11. Eight viscous layers were generated in the grid to capture the turbulent boundary layer. In the computation the k–ε turbulence model with a wall function is employed. Hyperbolic tangent grid clustering was used to achieve an average y^+ value of 27, which is appropriate for turbulence models with wall functions. The mesh has 63,454 nodes; 177,968 faces; and 57,307 cells.

For LU-SGS a CFL number of infinity is again used to achieve the best performance. For BLU-SGS and FIS a maximum CFL of 50 was found to give near-optimum performance. The inner iteration convergence tolerance ε is set to 0.1, and the maximum number of inner sweeps $K_{\max}$ is 10. BLU-SGS required about 26% more memory than LU-SGS, but FIS took 2.4 times the memory required by BLU-SGS. Figure 12 shows the residual histories vs number of iterations and CPU time. BLU-SGS and FIS converged nearly identically in terms of number of iterations. However BLU-SGS is faster than FIS in terms of CPU time. BLU-SGS is about four times as fast as LU-SGS. To give the reader some idea on the accuracy of the simulation, Fig. 13 and 14 show the comparison of surface pressure coefficients between the computation and experimental data at different locations. The computational prediction agrees very well with experimental data.

As a numerical exercise LU-SGS, BLU-SGS, and FIS were also tested for this geometry assuming a laminar flow condition. The Reynolds number used in this exercise is two orders larger than that used in the experiment, i.e., $Re = 2.33 \times 10^4$ with other conditions remaining the same. Figure 15 presents the residual histories vs number of iterations and CPU time for all of the schemes. Although FIS converges in fewer iterations than BLU-SGS, BLU-SGS is faster in terms of CPU time. BLU-SGS is about 2.5 times as fast as LU-SGS.

Conclusions

An improved BLU-SGS with increased efficiency is developed. BLU-SGS is significantly more efficient than the original LU-SGS, while requiring only 20–30% more memory. The convergence rate of BLU-SGS is comparative to and in many cases faster than the fully implicit scheme while requiring much less memory. The improved efficiency of BLU-SGS for both inviscid and viscous flows including turbulent problems has been demonstrated.

Acknowledgments

The research was supported by the National Science Foundation under award DMI-9460372, DMI-9560782, and DMI-9704186. The authors would like to thank Andrzej Przekwas and Ashok Singhal of CFD Research Corporation for many helpful discussions.

References

[1] Barth, T. J., and Frederickson, P. O., "High-Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," AIAA Paper 90-0013, Jan. 1990.

[2] Rausch, R. D., Batina, J. T., and Yang, H. T., "Spatial Adaptation Procedures on Unstructured Meshes for Accurate Unsteady Aerodynamic Flow Computation," AIAA Paper 91-1106, July 1991.

[3] Aftosmis, M., Gaitonde, D., and Tavares, S., "On the Accuracy, Stability and Monotonicity of Various Reconstruction Algorithms for Unstructured Meshes," AIAA Paper 94-0415, Jan. 1994.

[4] Venkatakrishnan, V., "A Perspective on Unstructured Grid Flow Solvers," AIAA Paper 95-0667, Jan. 1995.

[5]Anderson, W. K., "A Grid Generation and Flow Solution Method for the Euler Equations on Unstructured Grids," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 23–38.

[6]Schneiders, R., "Automatic Generation of Hexahedral Finite Element Meshes," *Proceedings of 4th International Meshing Roundtable '95*, Sandia National Labs., Albuquerque, NM, 1995, pp. 130–114.

[7]Nakahashi, K., "Adaptive Prismatic Grid Method for External Viscous Flow Computations," *Proceedings of 11th AIAA Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1993, pp. 195–203.

[8]Coirier, W. J., and Jorgenson, P. C. E., "A Mixed Volume Grid Approach for the Euler and Navier–Stokes Equations," AIAA Paper 96-0762, Jan. 1996.

[9]Luo, H., Sharov, D., Baum, J. D., and Lohner, R., "On the Computation of Compressible Turbulent Flows on Unstructured Grids," AIAA Paper 2000-0927, Jan. 2000.

[10]Kallinderis, Y., Khawaja, A., and McMorris, H., "Hybrid Prismatic/Tetrahedral Grid Generation for Complex Geometries," *AIAA Journal*, Vol. 34, No. 2, 1996, pp. 291–298.

[11]McMorris, H., and Kallinderis, Y., "Octree-Advancing Front Method for Generation of Unstructured Surface and Volume Meshes," *AIAA Journal*, Vol. 35, No. 6, 1997, pp. 976–984.

[12]Wang, Z. J., "An Automated Viscous Adaptive Cartesian Grid Generation Method for Complex Geometries," *Proceeding of 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, Univ. of Greenwich, Greenwich, England, U.K., 1998, pp. 577–586.

[13]Wang, Z. J., Chen, R. F., Hariharan, N., Przekwas, A. J., and Grove, D., "A 2^N Tree Based Automated Viscous Cartesian Grid Methodology for Feature Capturing," *Proceedings of 14th AIAA Computational Fluid Dynamics Conference*, AIAA, Reston, VA, 1999, pp. 447–457.

[14]Wang, Z. J., and Chen, R. F., "Anisotropic Cartesian Grid Method for Viscous Turbulent Flow," AIAA Paper 2000-0395, Jan. 2000.

[15]Venkatakrishnan, V., and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes," *Journal of Computational Physics*, Vol. 105, No. 1, 1993, pp. 83–91.

[16]Venkatakrishnan, V., and Mavriplis, D. J., "Implicit Method for the Computation of Unsteady Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 127, No. 2, 1996, pp. 380–397.

[17]Soetrisno, M., Imlay, S. T., and Roberts, D. W., "A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids," AIAA Paper 94-0645, Jan. 1994.

[18]Frink, N. T., "Assessment of an Unstructured-Grid Method for Predicting 3-D Turbulent Viscous Flows," AIAA Paper 96-0292, Jan. 1996.

[19]Blanco, M., and Zingg, D. W., "A Fast Solver for the Euler Equations on Unstructured Grids Using a Newton-GMRES Method," AIAA Paper 97-0331, Jan. 1997.

[20]Tidriri, M. D., "Krylov Methods for Compressible Flows," Inst. for Computer Applications in Science and Engineering, ICASE Rept. 95-48, Hampton, VA, June 1995.

[21]Meister, A., "Comparison of Different Krylov Subspace Methods Embedded in an Implicit Finite Volume Scheme for the Computation of Viscous and Inviscid Flow Fields on Unstructured Grids," *Journal of Computational Physics*, Vol. 140, No. 2, 1998, pp. 311–345.

[22]Jameson, A., and Yoon, S., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, 1987, pp. 929–935.

[23]Sharov, D., and Nakahashi, K., "Reordering of 3D Hybrid Unstructured Grids for Vectorized LU-SGS Navier–Stokes Computations," AIAA Paper 97-2102, July 1997.

[24]Sharov, D., and Nakahashi, K., "Low Speed Preconditioning and LU-SGS Scheme for 3D Viscous Flow Computations on Unstructured Grids," AIAA Paper 98-0614, Jan. 1998.

[25]Wright, M. J., Candler, G. V., and Prampolini, M., "Data-Parallel Lower-Upper Relaxation Method for the Navier–Stokes Equations," *AIAA Journal*, Vol. 34, No. 7, 1996, pp. 1371–1377.

[26]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[27]Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol. 118, No. 1, 1995, pp. 120–130.

[28]Wang, Z. J., "A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier–Stokes Equations," *Computers and Fluids*, Vol. 27, No. 4, 1998, pp. 529–549.

[29]Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA M6-Wing at Transonic Mach Numbers," Experiment Data Base for Computer Program Assessment, AR-138, AGARD, 1979.

[30]Townsend, J. C., Howell, D. T., Collins, I. K., and Hayes, C., "Surface Pressure Data on a Series of Analytic Forebodies at Mach Number from 1.70 to 4.50 and Combined Angles of Attack and Sideslip," NASA TM 80062, 1979.

J. Kallinderis
Associate Editor