A BLOCK LU-SGS IMPLICIT DUAL TIME-STEPPING ALGORITHM FOR HYBRID DYNAMIC MESHES

L.P. Zhang[†] and Z.J. Wang[‡]
Department of Mechanical Engineering
Michigan State University, East Lansing, MI 48824

ABSTRACT

A block lower-upper symmetric Gauss-Seidel (BLU-SGS) implicit dual time-stepping method is developed for moving body problems with hybrid dynamic grids. To simulate flows over complex configurations, a hybrid grid method is adopted in this paper. Body-fitted quadrilateral (quad) grids are generated first near solid bodies. An adaptive Cartesian mesh is then generated to cover the entire computational domain. Cartesian cells which overlap the quad grids are removed from the computational domain, and a gap is produced between the quad grids and the adaptive Cartesian grid. Finally triangular grids are used to fill this gap. With the motion of moving bodies, the quad grids move with the bodies, while the adaptive Cartesian grid remains stationary. Meanwhile, the triangular grids are deformed according to the motion of solid bodies with a 'spring' analogy approach. If the triangular grids become too skewed, or the adaptive Cartesian grid crosses into the quad grids, the triangular grids are regenerated. Then the flow solution is interpolated from the old to the new grid. The fully implicit equation is solved using a dual time-stepping solver. A Godunov-type scheme with Roe's flux splitting is used to compute the inviscid flux. Several sub-iteration schemes are investigated in this study. Both supersonic and transonic unsteady cases are tested to demonstrate the accuracy and efficiency of the method.

1. INTRODUCTION

The use of unstructured grids in computational fluid dynamics (CFD) has become widespread during the last two decades due to their ability to discretize arbitrarily complex geometries and the flexibility in supporting solution-based grid adaptations to enhance the solution accuracy and efficiency [1-6]. In the early days of unstructured grid development, triangular/ tetrahedral grids were employed primarily in dealing with complex geometries. Recently, mixed or hybrid grids including many different cell types have gained popularity because of the improved efficiency and accuracy over pure tetrahedral grids. For example, hybrid prism/tetrahedral grids [7], mixed grids including tets/prism/pyramid/hex cells [8], and adaptive Cartesian grid methods [9-15] have been used in many applications with complex configurations. In addition, solution algorithms for computing steady flows on unstructured and hybrid grids have evolved to a high degree of sophistication. The state-of-the-art spatial discretization algorithm is probably the second-order Godunov-type finite volume method [12]. For time integration, explicit algorithms such as multi-stage Runge-Kutta schemes are the easiest to implement. Convergence acceleration techniques such as local time-stepping and implicit residual smoothing [3] have also been employed in this context. However, for large-scale problems and especially for the solution of viscous turbulent flows, implicit schemes are required to speed up the convergence rate. In the last decade, significant progress has been made in the development of implicit numerical algorithms for steady flow simulations on unstructured grids [16-19]. For example, a block lower-upper symmetric Gauss-Seidel (BLU-SGS) scheme [19] was shown to dramatically speed up the convergence of complex turbulent flow computations with arbitrary grids, while its memory requirement is comparable to a point-implicit scheme.

The success demonstrated by unstructured grids for steady flow problems has prompted their applications to unsteady moving boundary flow problems. For a moving boundary flow problem, the computational grids must move with the moving boundaries. The most straightforward approach is to deform the computational grid locally using a spring-analogy type algorithm to follow the motion of the moving boundaries [20]. The approach is very efficient because it does not require solution interpolation. A disadvantage of the approach is that the grid integrity can be destroyed by large motions or shear-type of boundary motions. To remedy this drawback, local remeshing can be applied whenever the grid becomes too skewed. With local remeshing, solution interpolations from the old to the new grid become necessary. The hybrid approach of combining grid deformation with grid local remeshing seems to be the

[†] Visiting Scholar, Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, P. O. Box 211, Mianyang, Sichuan, 621000, China. zhangla@egr.msu.edu; or zhanglp@my-public.sc.cninfo.net
[‡] Associate Professor, AIAA Senior Member. zjw@egr.msu.edu

state-of-the-art in handling moving boundary problems, and has been used successfully for a variety of applications [21]. Most of the simulations for moving boundary problems are performed with unstructured triangular or tetrahedral grids only [20-23]. In this paper, we advocate a hybrid adaptive Cartesian/quad/ triangular grid approach for dynamic moving boundary flow problems in 2D (In 3D, the approach becomes adaptive Cartesian/prism/tetrahedral grid based). Body-fitted quad grids are generated first near solid bodies to resolve viscous boundary layers. An adaptive Cartesian grid is then generated to cover the outer domain, while triangular grids are used to fill the gap between the quad and Cartesian grids. If the bodies move, the quad grids move with the bodies, while the Cartesian grid remains stationary. Meanwhile, the unstructured grids are deformed according to the motions of the bodies with a 'spring' analogy approach. If the triangular grids become too skewed because of the deformation, the triangular grids are then regenerated, and the solutions are also interpolated from the old to the new grids. The advantages of above hybrid grid method include: 1) more efficient than the fully unstructured triangular grids, especially for viscous flow simulations; 2) more accurate for viscous flow computations because of the high-quality grids in the boundary layer; 3) Only a small local region needs to be remeshed resulting in less data interpolation and less errors caused by data interpolations.

The solution efficiency of the time-integration algorithm for a moving boundary flow problem becomes more important than for a steady problem because the unsteady residual should be driven close to zero at each time step. For viscous flow problems with highly clustered computational grids, an explicit time-marching method is clearly out of the question because of the time-step limit imposed by an explicit scheme. Therefore, only implicit schemes are considered in this study, while an explicit scheme may be used for comparison purposes. As we mentioned earlier, one has to drive the unsteady residual to zero (or at least to truncation error) at each time step when an implicit scheme is used to compute unsteady flows. Many of the successful implicit solvers for steady flows problems can be used to converge the unsteady residual, and there are successful examples in the literature [21]. The convergence rate of this "inner" solver determines the overall efficiency of the solution algorithm. In this study, we will extend and test the successful BLU-SGS scheme [19] to hybrid dynamic grids for moving boundary problems. Comparisons will also be made between explicit, point implicit and BLU-SGS inner iteration solvers.

The paper is organized as follows. In the next section, the hybrid adaptive Cartesian/quad/triangular grid generation approach will be presented, together with illustration examples. After that, the finite volume, Godunov-type second-order dual time-stepping method for dynamic grids is described. Details of the BLU-SGS inner iteration solver are presented. In Section 4, several unsteady moving boundary problems are computed. Temporal and spatial refinement studies are performed to ensure the computational solutions are time-step and grid independent. Computational results are compared with experimental data whenever possible. Finally conclusions from this study are summarized in Section 5.

2. GRID GENERATION STRATEGY

In this study, a hybrid adaptive Cartesian/quad/ triangular grid approach is adopted to discretize a complex computational domain. Body-fitted quad grids are generated first near solid bodies with an advancing layer method [6,7] by marching in the surface normal direction. When the aspect ratio of the quad cell reaches a pre-defined scale (for example, 0.6) or the width of remaining gap between solid bodies is only one or two times of the local grid size, the local advancing layer procedure stops. Then an adaptive Cartesian grid is generated to cover the outer domain by a modified quadtree method [24]. This adaptive Cartesian grid is generated by recursively subdividing a large root cell covering the entire computational domain. The grid resolution of the adaptive Cartesian grid automatically matches that of the quad grids near the outer layer of the quad grids. Of course, Cartesian cells close and inside the last advancing layer of the quad grids are removed from the computational domain. Usually the gap width between the quad grid and the adaptive Cartesian grid is set to several times of the local grid size (usually 6 times or less). Then triangular grids are used to fill the gap using an advancing front method (AFM) [25-26]. To control the grid distribution smoothly, a structured background grid [27] is employed, and some controlling point or line sources are specified in the field according to the configuration of interest. Finally, a 'spring' analogy approach is employed to smooth the initial grids.

Due to the motion of the moving boundaries, the grids at time level *n* is different from that at time level *n+1*. The grid at time *n+1* is usually generated by deforming the grid at time *n* with the grid connectivity or topology remaining the same. With our hybrid adaptive Cartesian/quad/triangular grid approach, we adopt the following strategy. The quad grids around moving bodies move (translate or rotate) with the moving bodies, while the adaptive Cartesian grid remains stationary. Meanwhile, the unstructured triangular grids are deformed according to the motions of the moving bodies with a 'spring' analogy approach

[20]. Because the quad grids are not deformed, the quality of the quad grids remains the same, which is obviously a benefit for computing the viscous boundary layer. In addition, the outer adaptive Cartesian grid always keeps the original topology. When large motions occur, the quad grids may cross into the original adaptive Cartesian grid, or the triangular grids become too skewed. In this case, a remeshing step is undertaken. The holes in the adaptive Cartesian are regenerated based on the outer boundary of the quad grids. New triangular grids are generated to fill the gap between the adaptive Cartesian and quad grids. The flow solutions are also interpolated from the old to the new grids at the time when the remeshing step is taken. Note that this remeshing step is always local, and occurs near the triangular grids. Therefore, this approach is expected to be very efficient, and accurate.

Several examples are used here to illustrate the basic grid generation algorithm. The first example is a fighter aircraft with a store. The store separates from the fighter with a given trajectory. Fig. 1a shows the initial hybrid grid over the configuration, while Fig.1b displays the close-up view near the body. Note that the grids are very smooth in the entire computational domain, and the grid quality is very good even in the narrow gap between the fighter and the store. Fig. 1c and Fig. 1d show the grids when the store is far away from the fighter. During the separation, the body-fitted quad grid around the store moves with the store. Note that the grid quality is still satisfactory even after very large motions.

The second example is a double airfoil configuration, which is made up by the authors to test the ability of present hybrid grid generation method in handling narrow gaps and shear-type of grid motions. The main airfoil is a RAE2822 airfoil, and the small one is a NACA0012, which is scaled to one-third of the main chord. Fig. 2a shows the hybrid grid over the combined configuration. Figs. 2b displays the close-up view of the same grid, while Fig. 2c and Fig. 2d present the hybrid grids when the small airfoil separates away from the cavity. Once again, the present hybrid grid generator produces high-quality grids for this quite complex multi-body geometry.

3. NUMERICAL METHOD

3.1. Finite Volume Method for Dynamic Grids

The time-dependent Reynolds-averaged Navier-Stokes equations for dynamic grids can be expressed in the integral form as

$$\frac{\partial}{\partial t}\int_V Q dV + \oint_S \left(F^i(Q) - Q\mathbf{v}_g \cdot \mathbf{n}\right) dS = \oint_S F^v(Q) dS \quad (3.1)$$

where S is the surface surrounding the control volume V, $\boldsymbol{n}$ is the out-going unit normal of S, $\boldsymbol{v_g}$ is the velocity of S, and $\boldsymbol{Q}$ is the vector of conservative variables, $\boldsymbol{F^i}$ is the inviscid and $\boldsymbol{F^v}$ the viscous flux vectors. The eddy viscosity for turbulent flow is calculated by the standard κ-ε turbulence model with a wall function [13].

If we integrate Equation (3.1) in a polygonal control volume V_i, we obtain

$$\frac{\partial}{\partial t}(QV_i) + \sum_f \left(F^i(Q) - Qv_{gn}\right)_f dS_f = \sum_f F_f^v(Q) dS_f$$

(3.2)

where the summation index f represents all the faces surrounding control volume V_i, and $v_{gn} = \mathbf{v}_g \cdot \mathbf{n}$. The inviscid flux is calculated using Roe's approximate Riemann solver with reconstructed state variables at both sides of a face. A least square linear reconstruction scheme of the primitive variables is used. Venkatakrishnan's limiter [28] is employed to make the scheme monotone. For the viscous term, a second-order centered scheme is used here. Other details of the flow solver are contained in [12-13].

The conservation of a constant flow is a necessary condition for any viable numerical scheme. Otherwise mass, momentum or energy would be produced unphysically by the numerical simulation. If we examine Equation (3.2), in order to preserve a uniform free stream, we must have:

$$\frac{\partial V_i}{\partial t} = \sum_f v_{gn} dS_f \quad (3.3)$$

This is the so-called Geometric Conservative Law (GCL) [29-30] in its semi-discretized form. Assume that the grid velocity is computed at time level n+1/2. Then we can us the following time discretization to achieve second-order accuracy

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} = \sum_f v_{gn} dS_f \ . \quad (3.4)$$

Instead of having the grid velocity v_{gn} satisfy Equation (3.4), we utilize the equation to calculate v_{gn}. In this case, we are absolutely sure that GCL is guaranteed. To this end, we employ a simple fact: the volume that a cell sweeps over is equal to the total of the volumes swept by its faces, i.e.,

$$V_i^{n+1} - V_i^n = \sum_f \Delta V_f \quad (3.5)$$

where ΔV_f represents the volume swept by face f. Comparing Equations (3.4) and (3.5), we arrive at the following equation:

$$\Delta V_f = \Delta t v_{gn} dS_f \quad or \quad v_{gn} = \frac{\Delta V_f}{\Delta t dS_f} \quad (3.6)$$

3.2. Time Integration Algorithm

Once the fluxes are evaluated for each cell face using the preceding finite volume scheme, the semi-discrete form of the governing equations is then integrated in time. For convenience, we rewrite Equation (3.2) as the following nonlinear system:

$$\frac{\partial (QV)_i}{\partial t} + R_i(Q) = 0 \tag{3.7}$$

where R_i is the residual given by

$$R_i(Q) = \sum_f \left(F^i(Q) - Qv_{gn} - F^v(Q)\right)_f dS_f \tag{3.8}$$

Then a family of implicit schemes for Equation (3.7) can be constructed as:

$$\frac{Q_i^{n+1}V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1-\theta)R_i(Q^{n+1}) + \theta R_i(Q^n) = 0 \tag{3.9}$$

If $\theta = 0$, the scheme is the backward Euler method. If $\theta = 1/2$, the resulting scheme known as the Crank-Nicholson method is second-order accurate in time.

Equation (3.9) represents a nonlinear system of coupled equations, which has to be solved at each time step. It can be solved by introducing a pseudo-time variable τ,

$$\frac{\partial (QV)_i}{\partial \tau} + R_i^*(Q) = 0 \tag{3.10}$$

and 'time-marching' the solution using local pseudo-time $\Delta\tau$, until Q converges to Q^{n+1}. Here Q is the approximation of Q^{n+1} and the unsteady residual $R_i^*(Q)$ is defined as

$$R_i^*(Q) = \frac{Q_i V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1-\theta)R_i(Q) + \theta R_i(Q^n) \tag{3.11}$$

Obviously, Equation (3.10) can be solved by using a variety of numerical schemes including the explicit multi-stage Runge-Kutta method. Of course the best efficiency is expected to be achieved by an implicit inner iteration schemes.

Implicit block LU-SGS inner iteration scheme

Chen and Wang developed a block LU-SGS (BLU-SGS) method for steady flows on arbitrary grids [19], which showed superior convergence property for steady flow simulations. Here we extend this method to unsteady flow computations on moving grids. For simplicity of presentation, we choose $\theta = 1$. Discretizing the pseudo-time in (3.10) with a backward Euler scheme, we obtain the following equation

$$\frac{V_i^{n+1}\left(Q_i^{(m+1)} - Q_i^{(m)}\right)}{\Delta\tau} + \frac{V_i^{n+1}Q_i^{(m+1)} - V_i^n Q_i^n}{\Delta t} + \sum_f \left[\tilde{F}_f^{\,i}\left(Q^{(m+1)}\right) - F_f^{\,v}\left(Q^{(m+1)}\right)\right] dS_f = 0 \tag{3.12}$$

where indices n, m indicates the real time and the pseudo-time levels. Here we let $Q^{(0)} = Q^n$ and the converged solution is then Q^{n+1}, and

$$\tilde{F}^i\left(Q^{(m)}\right) = F^i\left(Q^{(m)}\right) - Q^{(m)}v_{gn}\,. \tag{3.13}$$

Equation (3.12) can be rewritten further in the following delta form:

$$\frac{V_i^{n+1}\Delta Q_i^{(m)}}{\Delta\tau} + \frac{V_i^{n+1}\Delta Q_i^{(m)}}{\Delta t} + \sum_f \left[\Delta\tilde{F}^{i(m)} - \Delta F^{v(m)}\right]_f dS_f = -R_i^*\left(Q^{(m)}\right) \tag{3.14}$$

where

$$\begin{aligned} \Delta\tilde{F}^{i(m)} &= \tilde{F}^i\left(Q^{(m+1)}\right) - \tilde{F}^i\left(Q^{(m)}\right), \\ \Delta F^{v(m)} &= F^v\left(Q^{(m+1)}\right) - F^v\left(Q^{(m)}\right). \end{aligned} \tag{3.15}$$

If we assume that the inviscid and viscous fluxes depend only on the state variables at the two cells sharing the face, the flux differences can be approximated with the following formula

$$\begin{aligned} \Delta\tilde{F}^{i(m)} &= \left[\tilde{F}^i\left(Q_i^{(m+1)}, Q_j^{(m+1)}\right) - \tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m+1)}\right)\right] \\ &+ \left[\tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m+1)}\right) - \tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m)}\right)\right] \end{aligned} \tag{3.16}$$

and

$$\begin{aligned} \Delta F^{v(m)} &= \left[F^v\left(Q_i^{(m+1)}, Q_j^{(m+1)}\right) - F^v\left(Q_i^{(m)}, Q_j^{(m+1)}\right)\right] \\ &+ \left[F^v\left(Q_i^{(m)}, Q_j^{(m+1)}\right) - F^v\left(Q_i^{(m)}, Q_j^{(m)}\right)\right] \end{aligned} \tag{3.17}$$

where the subscripts i and j denote the current cell under consideration and its neighbor cell that shares face f, respectively. Linearizing the first terms on the right hand sides of Equations (3.16) and (3.17), we have

$$\tilde{F}^i\left(Q_i^{(m+1)}, Q_j^{(m+1)}\right) - \tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m+1)}\right) \approx \frac{\partial\tilde{F}^i}{\partial Q_i}\Delta Q_i^{(m)}$$

(3.18)

$$F^v\left(Q_i^{(m+1)}, Q_j^{(m+1)}\right) - F^v\left(Q_i^{(m)}, Q_j^{(m+1)}\right) \approx \frac{\partial F^v}{\partial Q_i}\Delta Q_i^{(m)}$$

(3.19)

Substituting Equations (3.16), (3.17), (3.18) and (3.19) back to Equation (3.14), we obtain:

$$\begin{aligned} &D\Delta Q_i^{(m)} \\ &+ \sum_f \left[\tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m)} + \Delta Q_j^{(m)}\right) - \tilde{F}^i\left(Q_i^{(m)}, Q_j^{(m)}\right)\right]_f dS_f \\ &- \sum_f \left[F^v\left(Q_i^{(m)}, Q_j^{(m)} + \Delta Q_j^{(m)}\right) - F^v\left(Q_i^{(m)}, Q_j^{(m)}\right)\right]_f dS_f \\ &= -R_i^*\left(Q^{(m)}\right) \end{aligned} \tag{3.20}$$

where

$$D = \left(\frac{V_i^{n+1}}{\Delta t} + \frac{V_i^{n+1}}{\Delta\tau}\right)I + \sum_f \left[\frac{\partial\tilde{F}_f^{\,i(m)}}{\partial Q_i} - \frac{\partial F_f^{\,v(m)}}{\partial Q_i}\right] dS_f \tag{3.21}$$

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

In the original LU-SGS approach, the first order numerical flux vectors in the left hand side of equation (3.20) are chosen as

$$\tilde{F}^i - F^v = \frac{1}{2}\left(\tilde{F}_i + \tilde{F}_j - \lambda_{ij}\left(Q_j - Q_i\right)\right) \tag{3.22}$$

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

$$\lambda_{ij} = \left|\mathbf{v} \cdot n\right| + a + \frac{2(\mu + \mu_t)}{\rho\left|n \cdot (r_j - r_i)\right|} \tag{3.23}$$

where $\boldsymbol{n}$ 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, $\mathbf{v}$ is the velocity vector, ρ is the density, a is the speed of sound, μ and μ_t are kinematic and turbulent viscosities, respectively. Then matrix D changes into a diagonal matrix and equation (3.20) degenerates into a set of scalar equations.

In order to improve the convergence rate of the original LU-SGS, BLU-SGS keeps 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 sweep procedures are solved efficiently with an exact LU decomposition method. More details can be found in [19].

3.3. Boundary Conditions

In order to treat the boundary cells as transparent as possible, a ghost cell is generated for each boundary cell. Then the solution variables at the ghost cell are computed from the boundary cell according to the physical boundary condition. For a steady inviscid flow, the velocity components at the ghost cell for a solid wall boundary are computed as:

$$u_{ghost} = u - 2n_x v_n, \qquad v_{ghost} = v - 2n_y v_n \tag{3.24}$$

where v_n is the normal velocity given by $v_n = un_x + vn_y$. Meanwhile, the density and pressure of the ghost cell are set to be the same as those of the boundary cell. For unsteady moving boundary problems, the condition must be adjusted since the boundary face now is moving. Then the normal velocity should be modified $v_n = un_x + vn_y - v_{gn}$. Similarly for an unsteady viscous surface boundary, the velocity components at the ghost cell are computed using the following equation,

$$u_{ghost} = -u + 2n_x v_{gn} \qquad v_{ghost} = -v + 2n_y v_{gn}\,. \tag{3.25}$$

In the far field, a characteristic analysis based on Riemann invariants is used to determine the values of the flow variables on the outer ghost cells. This analysis correctly accounts for wave propagation in the far field, which is important for rapid convergence to steady state and serves as a 'non-reflecting' boundary condition for unsteady applications.

4. NUMERICAL RESULTS

4.1. Moving Cylinder with Mach 4

This case was selected as a validation case to test the dynamic grid implementation. A steady state problem of supersonic flow around a cylinder (with respect to a reference frame fixed on the cylinder) was simulated as a moving boundary problem of a cylinder traveling at Mach 4 through stationary air. After the initial transients, the flow field around the moving cylinder should settle down, and should become "steady" with respect to the cylinder. A major feature of this flow problem is a bow shock ahead of the moving cylinder. For comparison purposes, this problem was also run in the steady mode. For the moving body simulation, a sequence of hybrid computational grids at different times are generated, and displayed in Fig. 3. These grids have around a total of 11K cells, with about 2.3K quad cells, 3.7K triangular cells and 5.0K Cartesian cells. The pressure contours at the corresponding times are also shown in Fig. 3. Note that a bow shock is generated from the wall when the cylinder starts to move. Later the shock moves ahead of the moving cylinder. Finally the bow shock remains at a fixed location relative to the cylinder. The pressure distributions along the cylinder surface from both the steady state and moving body simulations are compared in Fig. 4. It is obvious that the agreement is very good, indicating that the implementation of the dynamic grid solver is successful.

4.2. Inviscid Flow over an Oscillating NACA0012 Airfoil

This unsteady test case has been used extensively in the literature for validation and demonstration studies because of the availability of experimental data [31]. The geometry is the well-known NACA0012 airfoil, which undergoes harmonic pitching motion about the quarter chord with the following time-dependent angle of attack $\alpha = \alpha_m + \Delta a \sin \omega t$, where α_m is the mean angle of attack, and $\Delta\alpha$ the oscillation amplitude. The reduced frequency, which is an important similarity parameter for this unsteady problem, is defined as $\kappa = \omega c / 2U_\infty$, where U_∞ is the free stream velocity and c the chord length of the airfoil.

Since the flow is attached, it is assumed inviscid. The following flow parameters are chosen: $\alpha_m = 0.016°$, $\Delta\alpha$ = 2.51°, κ = 0.0814, and $M_\infty = 0.755$. The unsteady calculation was started from a steady-state solution at the same Mach number with the mean angle of attack. Several different views of the initial computational grid is shown in Fig. 5. The hybrid grid has a total of 8,373 cells, with 1,384 quad cells, 2,877 triangular cells, and 4,122 adaptive Cartesian cells. Even though the flow is assumed inviscid flow, the use of the quad cells near the moving body ensures that the computational grid has high quality throughout the computational domain.

The time history of the lift coefficient vs. the time dependent angle of attack using 16, 32 and 64 time steps per cycle is displayed in Fig. 6. It is clear from this figure that the computational solution is acceptable with only 16 time steps per cycle. The solutions are almost on top of each other when 32 and 64 steps are taken during each cycle. Also shown is a comparison with the experimental data of Landon [31]. Note that the difference between the computed and the experimental lift histories is quite small. In addition, the present computational results agree very well with those in [17] and [21], although they are not shown here. A grid refinement study was also performed to make sure that the computed solution is grid-independent. In Fig. 7, the lift coefficient histories computed on two different grids are shown. The coarse grid has about 8.5K cells, while the fine grid has about 16.8K cells. Note that the solutions on the coarse and fine grids are on top of each other.

To demonstrate the convergence property of the BLU-SGS inner iteration solver, we compare the convergence histories of BLU-SGS with those of the point implicit method and explicit Runge-Kutta method. Fig. 8 shows the convergence in terms of the number of inner-iterations at the first time step (starting from the initial steady-state solution with 16 time steps per cycle), which demonstrates the fast convergence rate by the BLU-SGS solver.

4.3. Turbulent Flow over Oscillating NACA0012 Airfoil

The last test case we considered was a viscous turbulent flow problem. The geometry is the same, but the flow parameters are different than those in the last case. The new flow parameters are: α_m=4.86°, $\Delta\alpha$=2.44°, κ=0.0810, $M_\infty = 0.6$, Re=4.8x10^6. To capture the viscous boundary layer accurately, the quad grid is clustered near the airfoil surface. Since a wall function is employed for the κ-ε turbulence model, the grid resolution in the boundary layer is deemed adequate. For the solution of the initial steady-state flow field, the hybrid grid has a total of 14,999 grid cells, with 7,937 quad, 3,254 triangular and 3,808 Cartesian cells. The unsteady moving body simulation started from the steady-state solution. Then 50 time steps are used for each cycle to capture the unsteady features. Fig. 9a displays the computed Mach number contours at several different angles of attack. For comparison purposes, an inviscid simulation was also carried out with the same flow conditions. The computed Mach contours from this inviscid simulation are shown in Fig. 9b for the same angles of attack. Note that the flow fields are quite different near the airfoil, especially at the trailing edge. The turbulent boundary layer and the wake are visible in the viscous solutions. The plot of the lift coefficient vs. the angle of attack is shown in Fig. 10. Results from both the inviscid and viscous simulations are plotted in the figure. It is obvious that the viscous simulation agrees much better with the experimental data [31] than the inviscid calculation, especially at lower angle of attack. At higher angle of attack, both the inviscid and viscous simulations miss the experimental data, with the inviscid simulation over-predicting and the viscous simulation under-predicting the lift. A possible cause of this discrepancy may be due to the inability of the turbulence model in resolving the separated region. Fig. 11 shows the time history of the lift coefficient. This figure demonstrates that the present dual time-stepping approach has a fast convergence rate, and the solution reaches a periodic steady state within the first two cycles. In Fig. 12, the pressure coefficient distributions along the airfoil surface are displayed. It is clear that the viscous results are slightly better on the lower surface than the inviscid results, and are much better near the trailing edge. Generally speaking, the viscous simulation produces better *Cp* distribution than its inviscid counterpart. Finally the convergence histories are plotted for both the inviscid and viscous simulations in Figs. 13. As expected, the BLU-SGS solver delivers much faster convergence rate than the point implicit or the explicit Runge-Kutta solvers.

In order to verify that the solution is grid-dependent, a grid refinement study was carried also out. A finer grid with about 25K cells is generated, and used in the same simulation. Fig. 14a shows the plots of lift coefficient vs. angle of attack computed from the viscous simulations using two different grids. Fig. 14b shows corresponding plots from the inviscid simulations using two different grids with 8.2K and 16.4K cells respectively. Once again, grid independence is demonstrated for both the inviscid and viscous cases.

5. CONCLUSION

A fast block LU-SGS implicit method has been extended to solve two-dimensional compressible unsteady flows on hybrid dynamic grids. The hybrid grid approach has been shown to be accurate, efficient and capable of handling complex geometries with moving boundaries. The use of the hybrid adaptive Cartesian/quad/triangular grids enables high resolution of viscous boundary layers, and allows high quality meshes to be generated throughout the computational domain. In addition, only the triangular grids are regenerated when remeshing occurs, making the approach accurate and efficient. Both inviscid and viscous calculations have been presented for the flow over moving bodies, including a cylinder, and an oscillating NACA0012 airfoil. The computational results have shown good agreement with other simulations or experimental data. The block LU-SGS solver demonstrated much higher convergence rate than the point implicit or explicit solvers.

ACKNOWLEDGEMENTS

This work is partially supported by AFOSR grant F49620-03-1-0202. The authors acknowledge the startup funding from the Department of Mechanical Engineering, College of Engineering of Michigan State University (MSU). The first author would like to thank China's Ministry of Education and the Delia Koo Grant in MSU for sponsoring his visit to MSU.

REFERENCES

1. Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J. Comput. Phys. 1987; 72:449-466.
2. Lohner R. and Parikh P. Generation of three-dimensional unstructured grids by the advancing front method. International J. Numerical Methods Fluids 1988; 8:1135-1149.
3. Jameson A, Baker TJ and Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103, 1986.
4. Venkatakrishnan V. A perspective on unstructured grid flow solvers. AIAA Paper 95-0667, Jan. 1995.
5. Weatherill NP. Unstructured grids: procedures and applications. Handbook of grid generation, Edited by Thompson JF, Soni BK and Weatherill NP, CRC Press, 1998: Chapter 26.
6. Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method, *AIAA Journal*, 1996, Vol.34, No.1: 43-49.
7. Kallinderis Y, Khawaja A, and McMorris H. Hybrid prismatic/tetrahedral grid generation for complex geometries. AIAA Journal 1996; 34:291-298.
8. Coirier WJ and Jorgenson PCE. A mixed volume grid approach for the Euler and Navier-Stokes equations. AIAA Paper 96-0762, Jan. 1996.
9. Coirier WJ and Powell KG. Solution–adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA J*. 1996; 34:938–945.
10. Aftosmis MJ, Berger MJ and Melton JE. Robust and efficient Cartesian mesh generation for component–based geometry. AIAA Paper No. 97–0196, 1997.
11. Karman SL. SPLITFLOW: a 3D unstructured Cartesian/ prismatic grid CFD code for complete geometries. AIAA–95–0343, 1995.
12. Wang ZJ. A Quadtree-based adaptive Cartesian/Quad grid flow solver for Navier-Stokes equations, *Computers & Fluids*, 1998, vol.27, no.4: 529-549.
13. Z.J. Wang, and R.F. Chen, "Anisotropic Solution-Adaptive Viscous Cartesian Grid Method for Turbulent Flow Simulation," *AIAA Journal*, Vol. 40, pp. 1969-1978, 2002.
14. Zhang LP, Zhang HX and Gao SC, A Cartesian/unstructured hybrid grid solver and its applications to 2D/3D complex inviscid flow fields, Proceedings of the 7th International Symposium on CFD, September 1997, Beijing, China, pp347-352.
15. Zhang LP, Yang YJ and Zhang HX, Numerical simulations of 3D inviscid/viscous flow fields on Cartesian/ unstructured/prismatic hybrid grids, Proceedings of the 4th Asian CFD Conference, September 2000, Mianyang, Sichuan, China.
16. Vassberg JC. A fast, implicit unstructured-mesh Euler method. AIAA Paper 92-2693, 1992.
17. Venkatakrishnan V and Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. AIAA Paper 95-1705, 1995.
18. Luo H, Baum JD and Lohner R. A fast, matrix-free implicit method for compressible flows on unstructured grid. *Journal of Computational Physics*, 1998, vol.146: 664-690.
19. Chen RF and Wang ZJ, Fast, Block Lower-Upper Symmetric Gauss Seidel Scheme for Arbitrary Grids, *AIAA Journal*, 2000, vol. 38, no. 12: 2238-2245.
20. Batina JT. Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis. *AIAA Journal*, 1991, vol.29, no.3: 327-333.
21. Luo H, Baum JD and Lohner R. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grid. *Computers & Fluids*, 2001, vol.30: 137-159.
22. Singh KP, Newman JC and Baysal O. Dynamic unstructured method for flows past multiple objects in relative motion. *AIAA Journal,* 1995, vol.33, no.4: 641-649.
23. Singh KP, Newman JC and Baysal O. Dynamic unstructured method for flows past multiple objects in relative motion. *AIAA Journal,* 1995, vol.33, no.4: 641-649.
24. Merry MA and Shephard MS. Automatic three-dimensional mesh generation by the modified-octree technique, *Int. J. Num. Meth. Eng.*, 1984, Vol.20: 1965-1990.
25. Parikh P, Pirzadeh S and Lohner R. A package for 3D unstructured grid generation, finite element flow solution and flow field visualization, NACA CR-182090, 1990.
26. Zhang LP, Guo C, Zhang HX and Gao SC. An unstructured grid generator and its applications for three-dimensional complex geometries. *Chinese Journal of Computational Physics*, 1999, Vol.16

(5): 552-558.
27. Pirzadeh S. Structured background grids for generation of unstructured grids by advancing front method, *AIAA Journal*, 1993, Vol.3, no.2.
28. Venkatakrishnan V. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 1995, vol.118, no.1: 120-130.
29. Thomas PD. And Lombard CK. Geometric conservation law and it's application to flow computations on moving grids. *AIAA Journal,* 1979: 1030-1037.
30. Wang ZJ and Yang HQ. Unstready flow simulation using a zonal multi-grid approach with moving boundaries. AIAA Paper 94-0057.
31. Landon H. Compendium of unsteady aerodynamic measurements. AGARD Report No. 702, 1983.

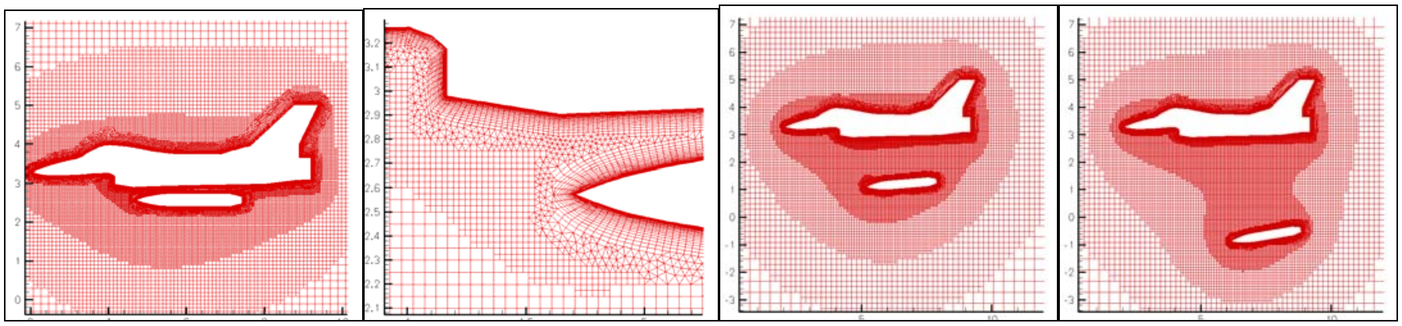


Fig. 1. Hybrid grids for a store-separation problem at different times

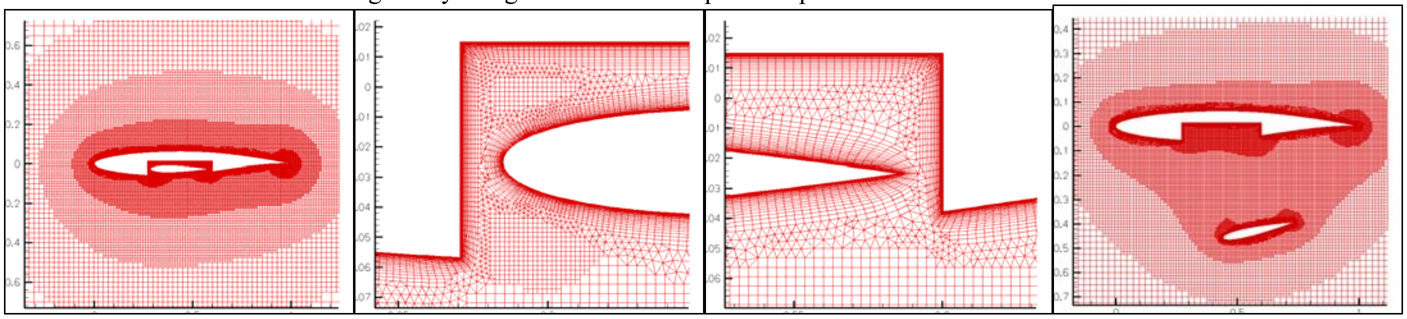
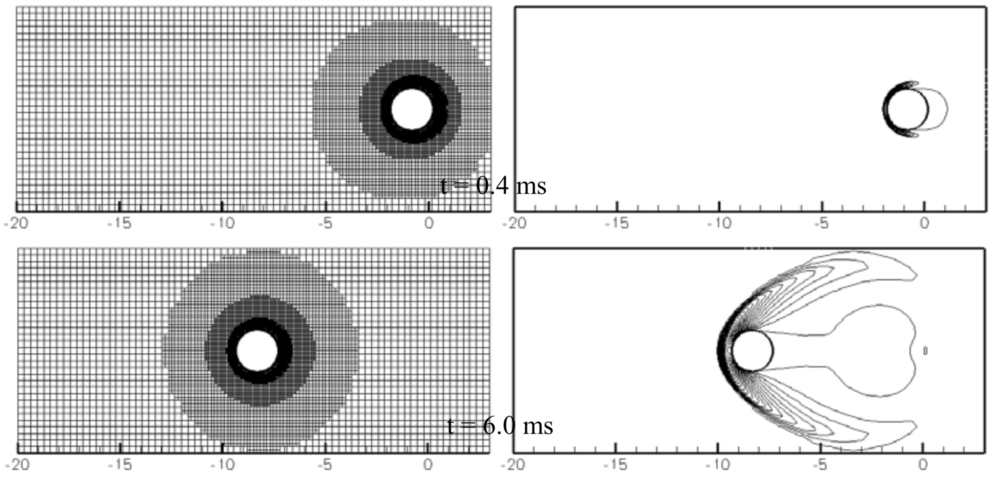


Fig. 2. Hybrid grids for a store-separation problem at different times



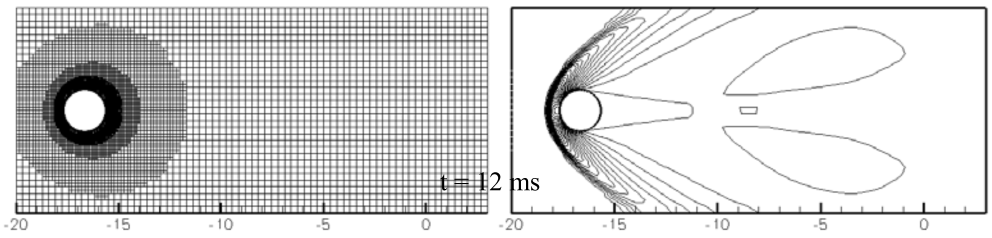


Fig. 3. Computational grids and computed pressure contours at different times

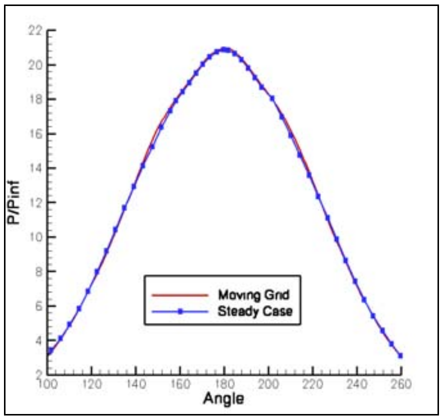


Fig. 4. Surface pressure distribution

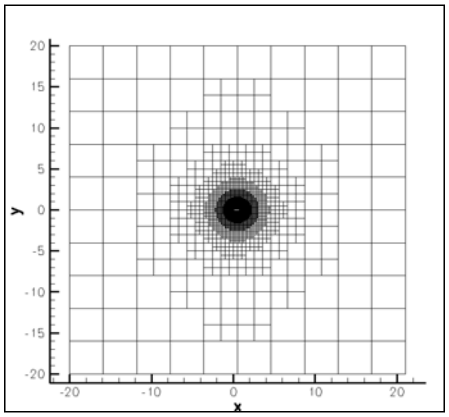


Fig.5a Hybrid grid around NACA0012 airfoil

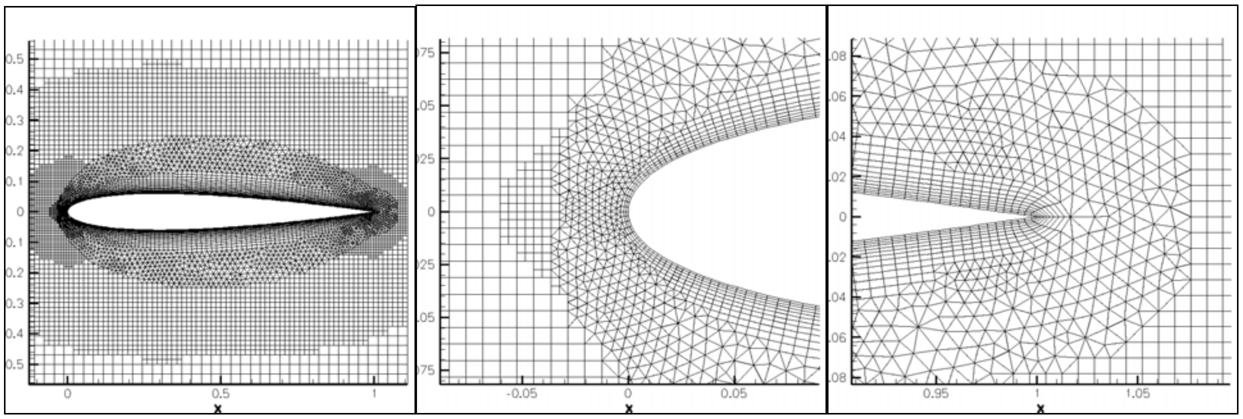


Fig. 5. Hybrid adaptive Cartesian/quad/triangular grids around the NACA0012 airfoil

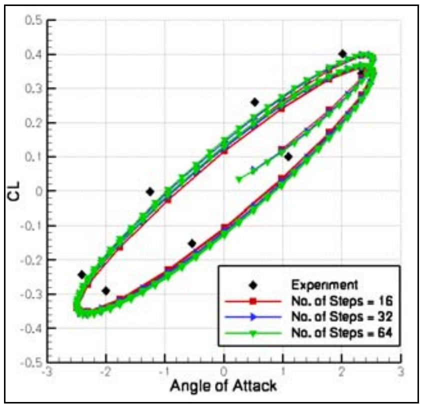


Fig. 6. Lift coefficient vs. angle of attack

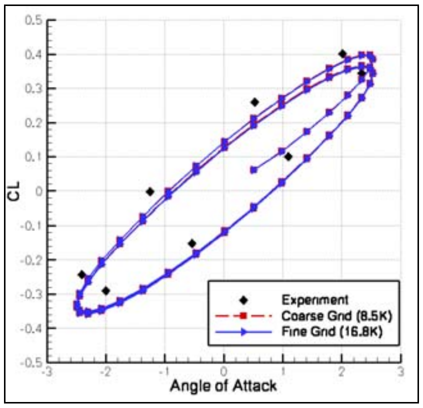


Fig. 7. Lift coefficient vs. angle of attack

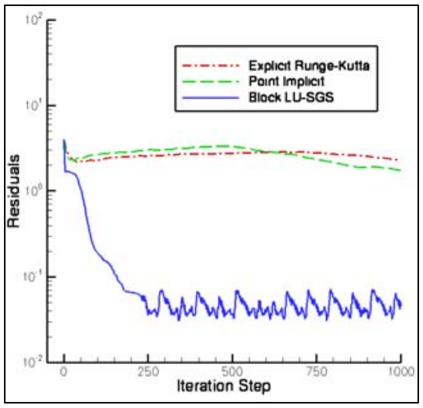


Fig. 8. Convergence histories

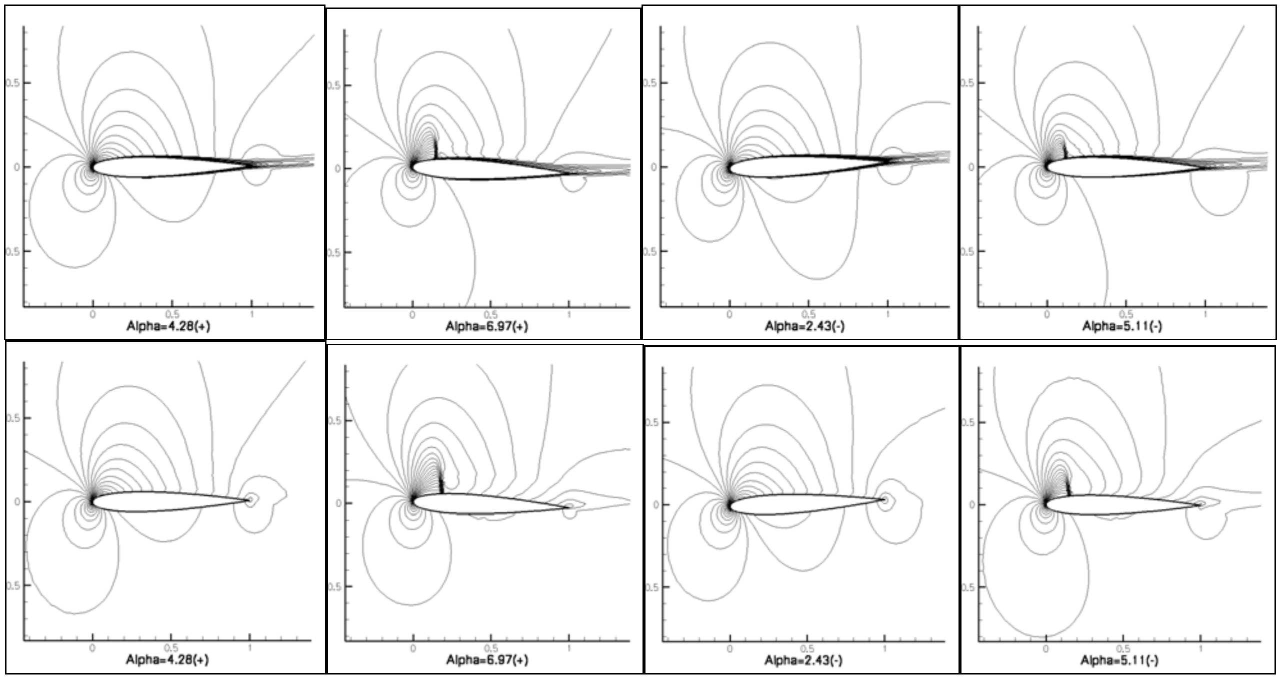


Fig. 9. Computed Mach number contours at different times (Up: Viscous; Down: Inviscid)

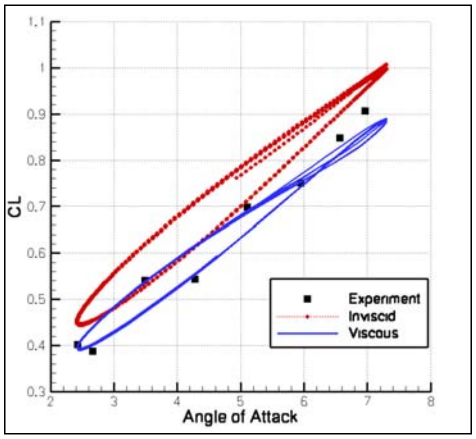


Fig. 10. Lift coefficient vs. angle of attack for both inviscid and viscous simulations

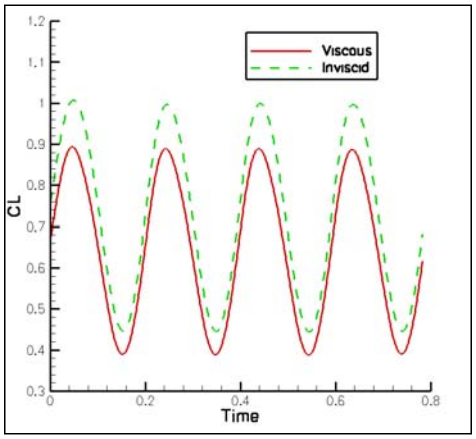
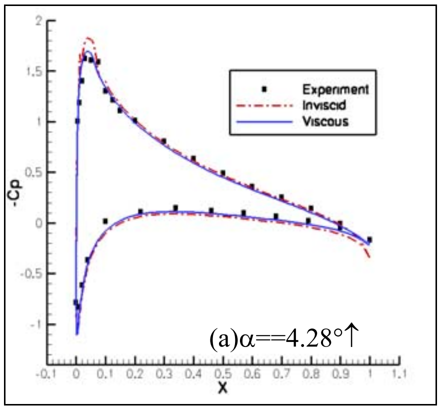
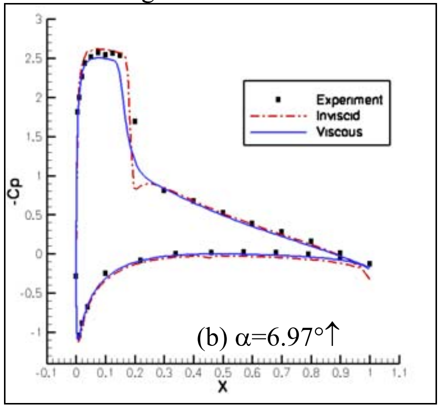


Fig. 11. Computed lift coefficient histories assuming inviscid and viscous flows





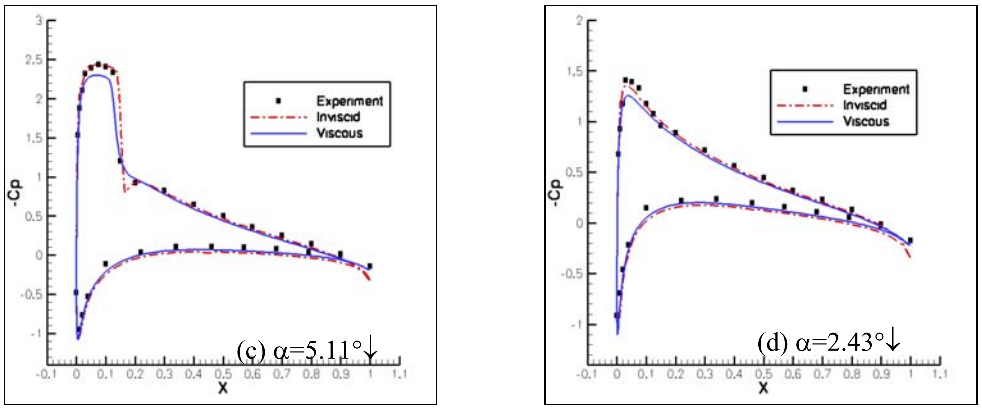


Fig. 12. Pressure coefficient distributions at four typical angles of attack

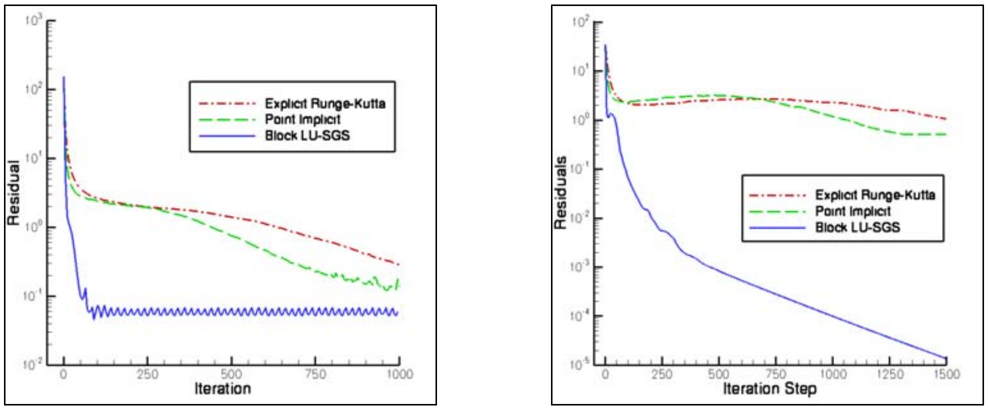


Fig. 13. Convergence histories in terms of number of iterations assuming inviscid and viscous flow

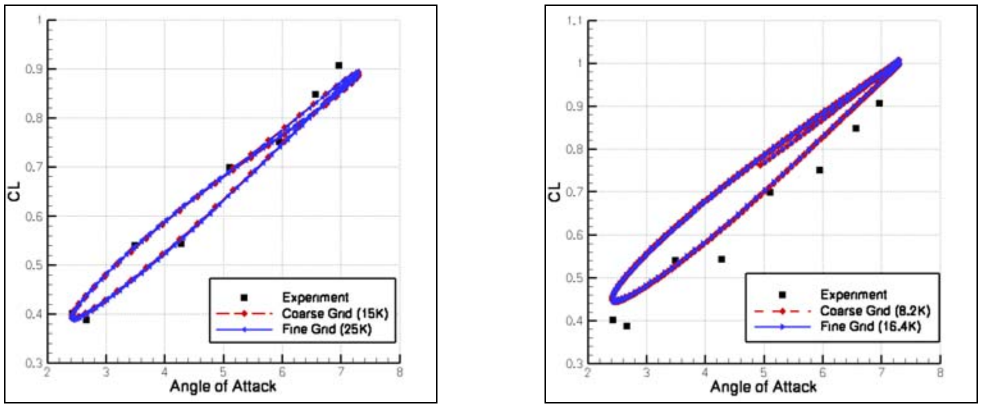


Fig. 14. Lift coefficient vs. angle of attack computed on two different grids for Viscous and Inviscid cases