ON THE COMPUTATION OF COMPRESSIBLE TURBULENT FLOWS ON UNSTRUCTURED GRIDS

Hong Luo, Dmitri Sharov, and Joseph D. Baum
Science Applications International Corporation
1710 Goodridge Drive, MS 2-6-9
McLean, VA 22102, USA

and

Rainald Löhner
Institute for Computational Sciences and Informatics
George Mason University, Fairfax, VA 22030, USA

ABSTRACT

An accurate, fast, matrix-free implicit method has been developed to solve compressible turbulent flow problems using the Spalart and Allmaras one equation turbulence model on unstructured meshes. The mean-flow and turbulence-model equations are decoupled in the time integration in order to facilitate the incorporation of different turbulence models and reduce memory requirements. Both mean flow and turbulent equations are integrated in time using a linearized implicit scheme. A recently developed, fast, matrix-free implicit method, GMRES+LU-SGS, is then applied to solve the resultant system of linear equations. The spatial discretization is carried out using a hybrid finite volume and finite element method, where the finite volume approximation based on a containment dual control volume rather than the more popular median-dual control volume is used to discretize the inviscid fluxes, and the finite element approximation is used to evaluate the viscous flux terms. The developed method is used to compute a variety of turbulent flow problems in both 2D and 3D. The results obtained are in good agreement with theoretical and experimental data and indicate that the present method provides an accurate, fast, and robust algorithm for computing compressible turbulent flows on unstructured meshes.

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 ease of adaption in enhancing the solution accuracy and efficiency through the use of adaptive refinement.

While unstructured grids composed of geometric simplices, i.e., triangle in two dimensions and tetrahedra in three dimensions are widely and routinely used for producing high-quality solutions for inviscid flow simulations, the development and application of simplicial unstructured grids to viscous flow simulations are far less popular, acceptable, and successful. For high Reynolds number viscous flows, the velocity variations in the normal direction are much larger than those in the tangential direction in the boundary layer and wake regions. This requires the use of the highly stretched meshes to efficiently resolve such viscous regions. For simplicial elements, this stretching invariably leads to the use of long and thin triangular cells. Numerical evidence, as reported by Aftosmis et al.[1], Haselbacher et al.[2], and Viozat et al.[3], indicates that for the discretization of the inviscid fluxes, the classic 1D-based upwind schemes using median-dual finite volume approximation suffer from excessive numerical diffusion due to such skewing. This is mainly due to the presence of diagonal connections, where the right and left state data required for the upwind schemes are not well aligned with the normal to the interface. Two general options exist for solving this problem: scheme-based methods and mesh-based methods. The former approach attempts to develop multidimensional upwind methods that are based more deeply on physical aspects of the Euler equations. Several efforts have been made in this direction[4−9]. Among them are rotated upwind methodologies, multidimensional wave decomposition methods, and characteristic theory-based advection schemes. Despite much progress, numerical investigations have shown that none of these methods have yet reached a satisfactory level with respect to their accuracy and robustness. The latter approach is based on improving of the grids. The typical example of this approach is to use hybrid grids[10−11], where the cell shapes, which do not become skewed with stretching (e.g., hexahedra and prisms), are used in the viscous regions and tetrahedral cells away from viscous regions. However, this is not completely satisfactory from the application point of view, as the existence of several cell types in a hybrid grid undoubtedly makes mesh generation and adaptation, flow solver, and post-processing more complex. Alternatively, one could use a containment dual finite volume approximation[3,12], which significantly reduces the influence of the diagonal edges in the grid. In

fact, one can show that the resulting control volume on an ideal triangulated quadrilateral grid using such containment dual is essentially identical to the control volume on the quadrilateral grid using a median control volume. Although the concept of the containment dual-based finite volume method is not very new, its accuracy, efficiency, and capability to treat real engineering problems is relatively unexplored. One of the major objectives of the efforts presented in this paper is to demonstrate that using containment dual-based finite volume approximation, simplicial unstructured grids can be applied to accurately compute high Reynolds number viscous flows.

Development of computationally efficient algorithms for high Reynolds number viscous flow simulations on highly stretched unstructured grids remains one of the unresolved issues in computational fluid dynamics. Due to the large differences between the convective and diffusive time scales, high Reynolds number flows produce very stiff systems of equations. This presents a severe challenge to the time advancement procedure. By itself, explicit, multi-stage time advancement is not a computationally viable alternative. A multigrid strategy or an implicit temporal discretization is required in order to speed up convergence. In general, implicit methods can outperform their explicit counterparts by a factor of 10 or even more. Unfortunately, this performance is usually achieved at the expense of memory increase, as some efficient implicit methods may need up to an order of magnitude more storage than the explicit methods. This is mainly due to the fact that implicit methods usually require the storage of the left-hand-side Jacobian matrix. Any implicit methods requiring the storage of the Jacobian matrix would be impractical, if not impossible, to use to perform turbulent simulations involving complex, realistic aerodynamic configurations, where hundreds of thousands of mesh points are necessary to represent such engineering-type configurations accurately. The authors have developed a fast, matrix-free implicit method, GMRES+LU-SGS[13], for solving compressible Euler and Navier-Stokes equations on unstructured grids. The developed GMRES+LU-SGS method has proven to be very effective for accelerating the convergence to both steady[13] and unsteady[14] flow problems. Another objective of this paper is to demonstrate that the matrix-free GMRES+LU-SGS method can also been extended and applied to efficiently compute high Reynolds number viscous flows on highly stretched unstructured grids.

The outline of this paper is as follows. In section 2, the Reynolds-averaged Navier-Stokes equations and the Spalart-Allmaras one-equation turbulence model[15] are described. The numerical method for solving the governing equations are presented in section 3. The computational results for a variety of turbulent flow problems and comparison of numerical solutions with available analytical solution and experimental data are presented in section 4. Finally, conclusions are summarized in section 5.

2. GOVERNING EQUATIONS

The Reynolds-averaged 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 flow variable vector $\mathbf{U}$, inviscid flux vector $\mathbf{F}$, and viscous flux vector $\mathbf{G}$, are defined by

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

$$\mathbf{G}^j = \begin{pmatrix} 0 \\ \sigma_{ij} \\ u_l \sigma_{lj} + q_j \end{pmatrix}. \tag{2.2}$$

Here ρ, p, and e denote the density, pressure, and specific total energy of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. The pressure can be computed from the equation of state

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

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

$$\sigma_{ij} = (\mu + \mu_t)(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) - \frac{2}{3}(\mu + \mu_t)\frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{2.4}$$

$$q_j = \frac{1}{\gamma - 1}(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t})\frac{\partial T}{\partial x_j}. \tag{2.5}$$

In the above equations, T is the temperature of the fluid, Pr the laminar Prandtl number, which is taken as 0.7 for air, and Pr_t the turbulent Prandtl number, which is taken as 0.9. μ represents the molecular viscosity, which can be determined through Sutherland's law

$$\frac{\mu}{\mu_0} = (\frac{T}{T_0})^{\frac{3}{2}}\frac{T_0 + S}{T + S}. \tag{2.6}$$

μ_0 denotes the viscosity at the reference temperature T_0, and S is a constant which for air assumes the value S=110°K. The temperature of the fluid T is determined by

$$T = \gamma\frac{p}{\rho}, \tag{2.7}$$

and μ_t denotes the turbulence eddy viscosity, which is computed using Spalart and Allmaras one equation turbulence model

$$\mu_t = \rho\tilde{\nu} f_{v1}. \tag{2.8}$$

The system is closed by including the governing equation for the working variable $\tilde{\nu}$, which can be written as

$$\frac{\mathrm{D}\tilde{\nu}}{\mathrm{D}t} = \frac{1}{\sigma}\nabla\cdot((\nu+(1+c_{b2})\tilde{\nu})\nabla\tilde{\nu}) - \frac{c_{b2}}{\sigma}\tilde{\nu}\nabla\tilde{\nu}$$

$$+c_{b1}(1-f_{t2})\tilde{S}\tilde{\nu} - (c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2})(\frac{\tilde{\nu}}{d})^2. \qquad (2.9)$$

The constants appearing in the above equations take on the standard values recommended in Ref. 15. In the sequel, we assume that Ω is the flow domain, Γ its boundary, and $\mathbf{n}_j$ the unit outward normal vector to the boundary.

3. NUMERICAL METHOD

The numerical algorithms for solving the governing equations (2.1) and (2.9) are based on a loosely coupled approach. In this approach, the flow and turbulence variables are updated separately. At each time step, the flow equations (2.1) are solved first using the previously updated turbulent eddy viscosity. The turbulent equation (2.9) is then solved using the newly updated flow variables to advance the turbulence variable in time. Such a loosely coupled approach allows for the easy interchange of new turbulence models. Another advantage of this approach is that all the working arrays used to solve the flow equations can be used to solve the turbulence equation. As a result, no additional storage is needed for the present implementation.

3.1 HYBRID DISCRETE FORMULATION

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

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

where $\mathcal{T}_h$ is a discrete approximation space of suitably continuous functions. Inviscid fluxes are discretized using a cell-vertex finite volume formulation, where the control volumes are nonoverlapping dual cells. Due to the easy construction, the median dual control volume, which is constructed by connecting the centroid of the triangle to its midpoint sides as shown in Figure 1, is widely used for the discretization of the inviscid fluxes. It can also be shown that the finite volume approximation using such median dual control volume is equivalent to the finite element approximation using linear elements. Numerical evidence, as reported by Aftosmis et al.[1], Haselbacher et al.[2], and Viozat et al.[3], indicates that finite volume schemes based on the median dual control volume suffer from excessive numerical diffusion on highly stretched triangular grids, resulting in inaccurate discretization methods. This is mainly due to the diagonal connections, where the right and left state data introduced into the upwind schemes to compute the convective terms are not well aligned with the normal to the interface. However, it would be somewhat naive to use the median dual geometry on highly stretched triangular grids, as there are other common dual meshes. One of them, containment dual, is especially helpful to design accurate methods for discretizing the convective terms on highly stretched triangulations. The containment dual control volume is constructed by connecting the circumcenter of the triangle to its midpoints sides as shown in Figure 2. For an obtuse triangle where its circumcenter is outside the triangle, the circumcenter point is moved to the midpoint of the longest side of this triangle. As a result, the control volume on an ideal triangulated quadrilateral grid using such containment dual is essentially identical to the control volume on the quadrilateral grid using a median control volume as shown in Figure 3. In general, the construction of containment duality is more sophisticated than its median counterpart, but the resulting improvement in accuracy can be dramatic. Furthermore, the use of containment duality also improves computational efficiency, as the fluxes computations are not required for the diagonal edges of grids.

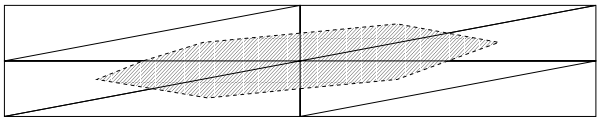

Figure 1. Median dual control-volume for stretched triangular grid.

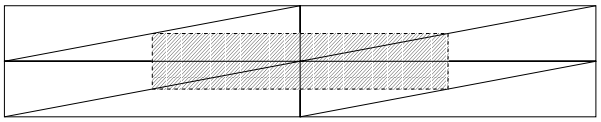

Figure 2. Containment dual control-volume for stretched triangular grid.

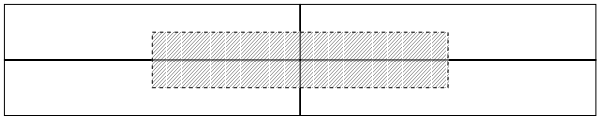

Figure 3. Median dual control-volume for stretched quadrilateral grid.

In the present study, the numerical flux functions for inviscid fluxes at the dual mesh cell interface are computed using AUSM+ (Advection Upwind Splitting Method) scheme[16]. A MUSCL[17] 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. 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, and $\mathbf{R}_i$ is the right-hand-side residual and equals to zero for a steady state solution.

The discretization for the Spalart-Allmaras turbulence model is very similar to the flow equations, where the advective terms are discretized using a first-order upwind method for robustness purposes and the diffusive and anti-diffusive terms are discretized using a Galerkin finite element method with piecewise linear elements.

3.2 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, equation (3.2) can be written in discrete form as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = -\mathbf{R}_i^{n+1}, \tag{3.3}$$

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

$$\Delta \mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n \ . \tag{3.4}$$

Equation (3.3) can be linearized in time as

$$V_i \frac{\Delta \mathbf{U}_i^n}{\Delta t} = -(\mathbf{R}_i^n + \frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}} \Delta \mathbf{U}_i), \tag{3.5}$$

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

$$A \Delta \mathbf{U} = -\mathbf{R}, \tag{3.6}$$

where

$$A = \frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}}. \tag{3.7}$$

Note that as Δt tends to infinity, the scheme reduces to standard Newton's method for solving a system of nonlinear equations. Newton's method is known to have a quadratic convergence property. The term $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$ 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 non-zero 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 $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$ 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

$$\mathbf{R}_i = \sum_j \frac{1}{2}[\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij}) - \mid \lambda_{ij} \mid (\mathbf{U}_j - \mathbf{U}_i)] \mid \mathbf{s}_{ij} \mid , \tag{3.8}$$

where

$$\mid \lambda_{ij} \mid = \mid \mathbf{V}_{ij} \cdot \mathbf{n}_{ij} \mid + C_{ij} + \frac{\mu_{ij} + \mu_{t_{ij}}}{\rho_{ij} \mid \mathbf{x}_j - \mathbf{x}_i \mid}, \tag{3.9}$$

where $\mathbf{s}_{ij}$ is the area vector normal to the control-volume interface associated with the edge ij, $\mathbf{n}_{ij} = \mathbf{s}_{ij} / \mid \mathbf{s}_{ij} \mid$ its unit vector in the direction $\mathbf{s}_{ij}$, C_{ij} the speed of sound, and the summation is over all neighboring vertices j of vertex i. 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[18]

$$\mathbf{R}_i^{\text{inv}} = \sum_j \frac{1}{2}[\mathbf{F}(\mathbf{U}_i, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j, \mathbf{n}_{ij}) - \mid J(\tilde{U}) \mid (\mathbf{U}_j - \mathbf{U}_i)] \mid \mathbf{s}_{ij} \mid , \tag{3.10}$$

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 (3.8) yields

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} = \sum_j \frac{1}{2}(J(\mathbf{U}_i, \mathbf{n}_{ij}) + \mid \lambda_{ij} \mid \mathbf{I}) \mid \mathbf{s}_{ij} \mid \tag{3.11}$$

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j} = \sum_j \frac{1}{2}(J(\mathbf{U}_j, \mathbf{n}_{ij}) - \mid \lambda_{ij} \mid \mathbf{I}) \mid \mathbf{s}_{ij} \mid , \tag{3.12}$$

where $J = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ represents the Jacobian of the 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-side in equation (3.6). 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 i-th column and j-th row and similarly the j-th column and i-th 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

$$U_{ij} = \frac{1}{2}(J(\mathbf{U}_j, \mathbf{n}_{ij}) - \mid \lambda_{ij} \mid \mathbf{I}) \mid s_{ij} \mid, \tag{3.13}$$

$$L_{ij} = \frac{1}{2}(-J(\mathbf{U}_i, \mathbf{n}_{ij}) - \mid \lambda_{ij} \mid \mathbf{I}) \mid s_{ij} \mid, \tag{3.14}$$

$$D_{ii} = \frac{\mathrm{V}}{\Delta t}\mathbf{I} + \sum_j \frac{1}{2}(J(\mathbf{U}_i, \mathbf{n}_{ij}) + \mid \lambda_{ij} \mid \mathbf{I}) \mid s_{ij} \mid . \tag{3.15}$$

Note that U, L, and D represent strict upper matrix, strict lower matrix, and 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) the number of unknown variables, and $nedge$ the number of edges. Note that in 3D $nedge \approx 7npoin$. Clearly, the upper and lower matrix consume substantial amounts of memory, taking 93% storage required for left-hand-side Jacobian matrix.

Equation (3.6) 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[19] on structured grids has been successfully generalized and extended to unstructured meshes by several authors.[20–22] 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 equation (3.16). The resulting equation can be solved in the two steps shown in equations (3.17) and (3.18), each of them involving only simple block matrix inversions.

$$(D+L)D^{-1}(D+U)\Delta\mathbf{U} = -\mathbf{R} + (LD^{-1}U)\Delta\mathbf{U} \tag{3.16}$$

Lower (forward) sweep:

$$(D+L)\Delta\mathbf{U}^\star = -\mathbf{R} \tag{3.17}$$

Upper (backward) sweep:

$$(D+U)\Delta\mathbf{U} = D\Delta\mathbf{U}^\star. \tag{3.18}$$

Both lower and upper sweep can be vectorized by appropriately reordering the grid points,[22] resulting 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 $\Delta\mathbf{F}$:

$$J\Delta\mathbf{U} \approx \Delta\mathbf{F} = \mathbf{F}(\mathbf{U}+\Delta\mathbf{U}) - \mathbf{F}(\mathbf{U}). \tag{3.19}$$

The forward sweep and backward sweep steps can then be expressed as

$$\Delta\mathbf{U}_i^\star = D^{-1}[-\mathbf{R}_i - \sum_{j:j<i} \frac{1}{2}(\Delta\mathbf{F}(\mathbf{u}_j^\star, \mathbf{n}_{ij}) - \mid \lambda_{ij} \mid \Delta\mathbf{U}_j^\star) \mid s_{ij} \mid], \tag{3.20}$$

$$\Delta\mathbf{U}_i = \Delta\mathbf{U}_i^\star - D^{-1} \sum_{j:j>i} \frac{1}{2}(\Delta\mathbf{F}(\mathbf{U}_j\mathbf{n}_{ij}) - \mid \lambda_{ij} \mid \Delta\mathbf{U}_j) \mid s_{ij} \mid . \tag{3.21}$$

The most remarkable achievement of this approximation is that there is no need to store the upper- and lower- matrices U and L, which substantially reduces the memory requirements. 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.[23] 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.[12,13,24–27] 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

$$\tilde{A}\Delta\tilde{\mathbf{U}} = -\tilde{\mathbf{R}} \tag{3.22}$$

instead of the original system (3.6), in the hope that $\tilde{A}$ is well conditioned. Left preconditioning involves premultiplying the linear system with a matrix as

$$P^{-1}A\Delta\mathbf{U} = -P^{-1}\mathbf{R}\ , \tag{3.23}$$

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, and 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 non-linear system is reduced. In the present work, the LU-SGS presented above is used as a preconditioner, i.e.,

$$P = (D + L)D^{-1}(D + U). \tag{3.24}$$

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. As GMRES 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.

The turbulence model equation is integrated in time using an implicit method similar to that of the mean flow equations. The same GMRES+LU-SGS procedure is then used to solve the resulting linear system. No matrix-free approach is attempted to solve the turbulence equation, as the storage of the left-hand-side Jacobian for the single turbulence equation is not very demanding. Since the turbulence equation is solved separately from the mean flow equations, all the working arrays used to solve the flow equations can be used to solve the turbulence equation without any additional storage.

4. NUMERICAL RESULTS

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 required to solve the system of the linear equations: even 20 iterations can not guarantee that the stopping criterion will be satisfied for some problems. However, it takes less than 20 iterations to solve the linear equations at a later time, and global convergence is not affected by a lack of linear system convergence during the first few time steps. All 2D computations were performed on an Intel Pentium II 400 MHz PC using Linux operating system. All 3D computations were performed on a single processor CRAY C916/161024 supercomputer at the HPCSM ERODEC center.

Test Case 1. Transonic flow past RAE2822 airfoil

The first test case was a simulation of transonic flow over a RAE2822 airfoil at a Mach number of 0.73, a chord Reynolds number of 6.5 million, and an incidence of 2.8°. The flow condition is denoted as test case 9 in the experimental report by Cook et al.[28] The mesh used in the computation shown in Fig. 4a. contains 25,172 elements, 12,741 points, and 310 boundary points. The mesh in the vicinity of the trailing edge was severely distorted, yielding elements with very large angles (nearly 180°). The computed Mach number and pressure contours in the flow field are displayed in Figs. 4b and 4c, respectively. A shock wave forms at about 55 per cent on the upper surface. The computed eddy viscosity is depicted in Fig. 4d. A smooth distribution of eddy viscosity throughout the boundary-layer, and vanishingly small values in the inviscid regions of flow are observed. The computed surface pressure and skin friction distributions are compared with experimental data[28] in Figs. 4e and 4f, respectively, indicating an overall good agree-

ment. The computed lift coefficient of 0.801 and drag coefficient of 0.0160 are compared favorably with the experimental values of 0.803 and 0.0168, respectively. Figures 4g and 4h display a comparison of convergence histories among the explicit scheme, matrix-free LU-SGS scheme, and 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 2. A CFL number of 500 was used by all implicit methods in the computation. It is clear that the GMRES+LU-SGS methods are superior to both the explicit method and the LU-SGS method. For this particular problem, the GMRES+LU-SGS method is about two orders of magnitude faster than its explicit counterpart, demonstrating the effectiveness of GMRES+LU-SGS method for accelerating the convergence for compressible turbulent flow problems.

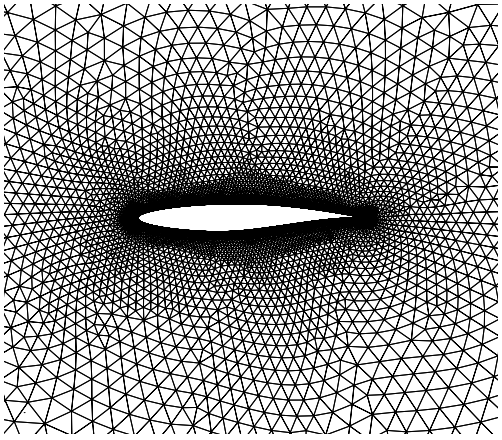

Fig. 4a: Unstructured mesh used for computing turbulent flow over a RAE2822 airfoil (nelem=25,172, npoin=12,741, nboun=310).

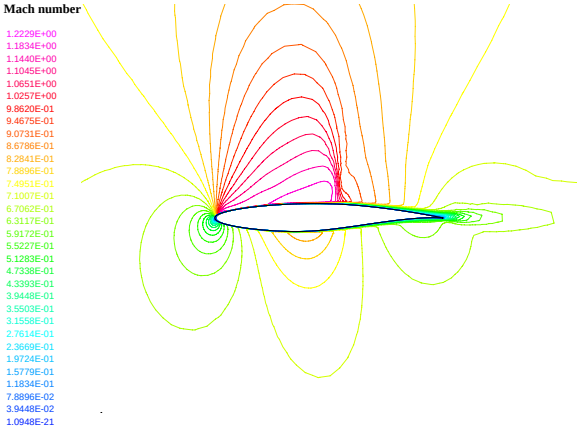

Fig. 4b: Computed Mach number contours for turbulent flow over a RAE2822 airfoil (M_∞=0.73, Re=6.5 million, α=2.8°).

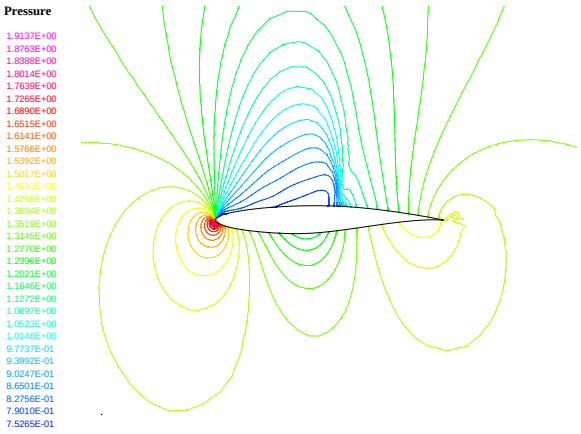

Fig. 4c: Computed pressure contours for turbulent flow over a RAE2822 airfoil (M_∞=0.73, Re=6.5 million, α=2.8°).

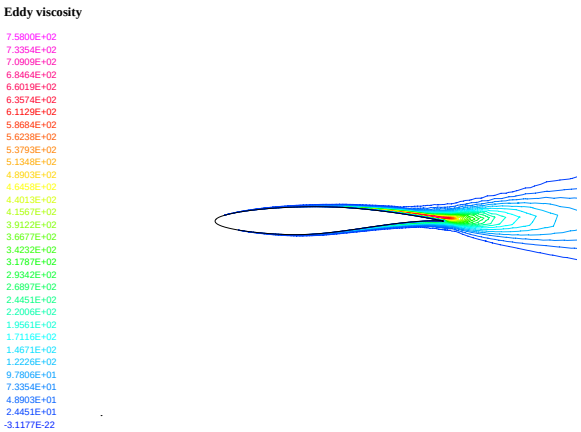

Fig. 4d: Computed eddy viscosity contours for turbulent flow over a RAE2822 airfoil (M_∞=0.73, Re=6.5 million, α=2.8°).

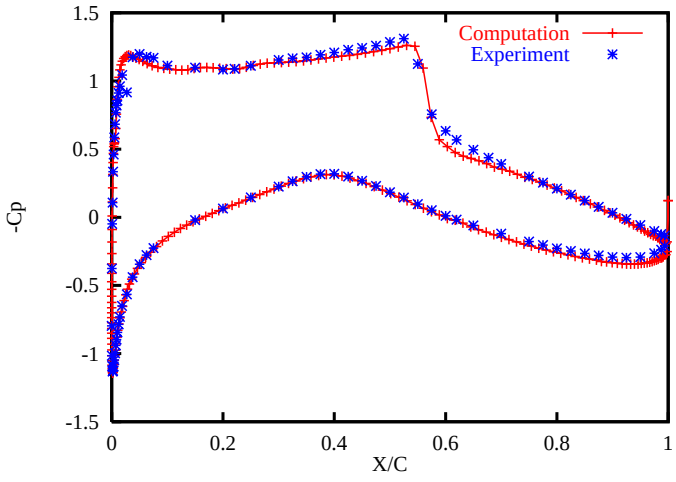

Fig. 4e: Comparison of computed surface pressure coefficient with experimental data for turbulent flow over a RAE2822 airfoil.

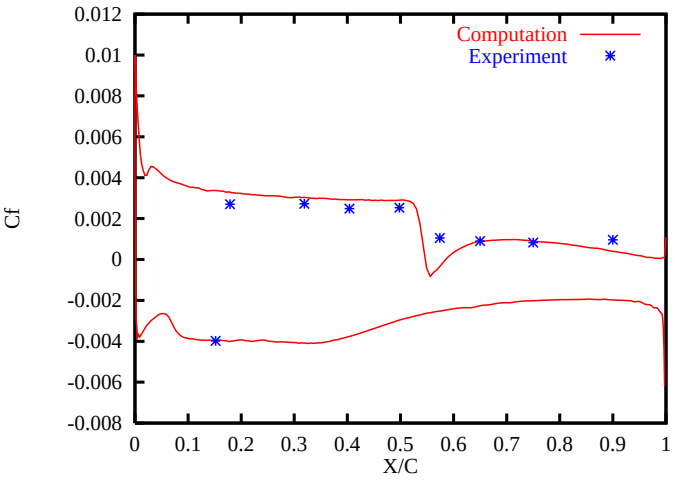

Fig. 4f: Comparison of computed skin friction coefficient with experimental data for turbulent flow over a RAE2822 airfoil.

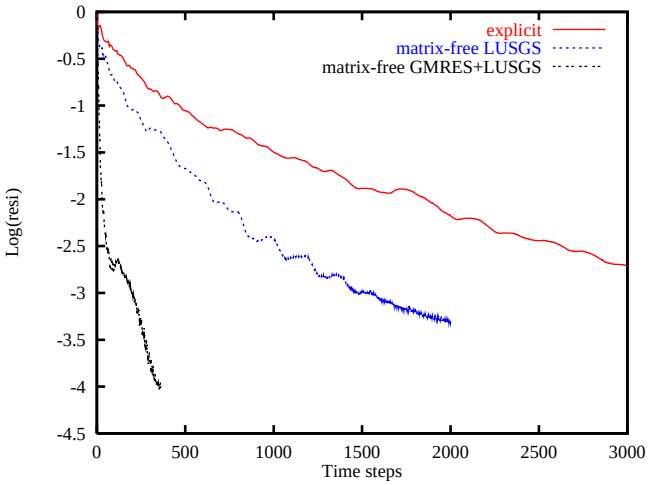

Fig. 4g: Residual convergence history versus time steps for turbulent flow over a RAE2822 airfoil using explicit, matrix-free LU-SGS and matrix-free GMRES+LU-SGS methods.

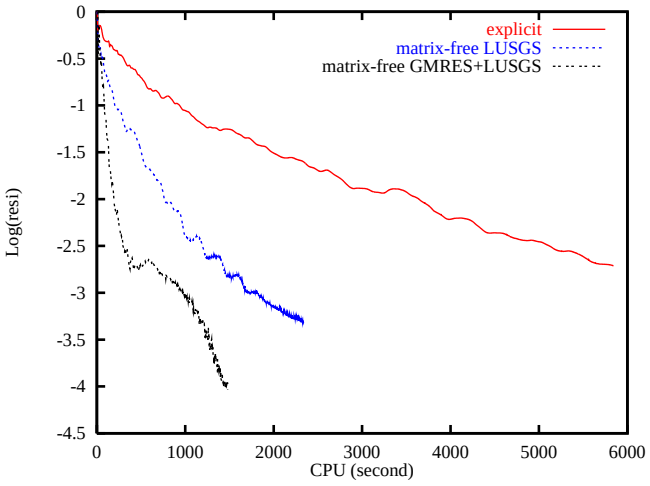

Fig. 4h: Residual convergence history versus CPU time for turbulent flow over a RAE2822 airfoil using explicit, matrix-free LU-SGS and matrix-free GMRES+LU-SGS methods.

Test Case 2. Subsonic flow past Douglas three-element airfoil

The second test case involves the modeling of subsonic flow past a Douglas three-element airfoil in a landing configuration at a Mach number of 0.2, an angle of attack of 16°, and a Reynolds number of 9 million. This test case was chosen to demonstrate that the complex geometries and resulting flowfields can be easily handled by the present method. The computational mesh shown in Fig. 5a consists of 66,530 elements, 33,618 points, and 710 boundary points. The computed Mach number and pressure contours in the flow field are displayed in Figs. 5b and 5c, respectively. The computed eddy viscosity is depicted in Fig. 5d. The computed surface pressure distribution in Fig. 5e is seen to compare favorably with experimental data[29]. Figures 5f and 5g display a comparison of convergence histories among the explicit 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 four-stage Runge-Kutta time-stepping scheme with local time stepping and implicit residual smoothing. The computation was advanced with a CFL number of 0.6. A CFL number of 200 was used by all implicit methods in the computation. The effectiveness of the GMRES+LU-SGS method, compared to both LU-SGS and explicit methods, is especially apparent for this test case. Note that the explicit method required unrealistically CPU time to reach a steady state solution, indicating the necessity of using implicit methods for computing turbulent flows on a highly stretched unstructured grid.

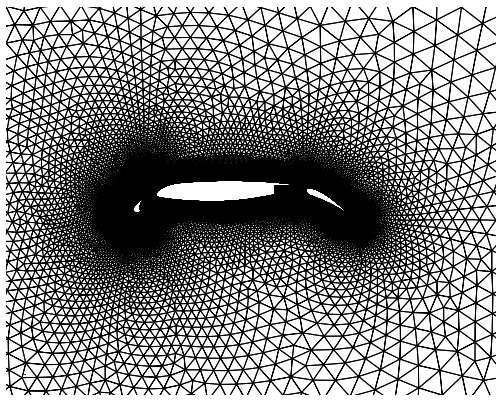

Fig. 5a: Unstructured mesh used for computing turbulent flow over a Douglas 3-element airfoil (nelem=42,059, npoin=21,343, nboun=633).

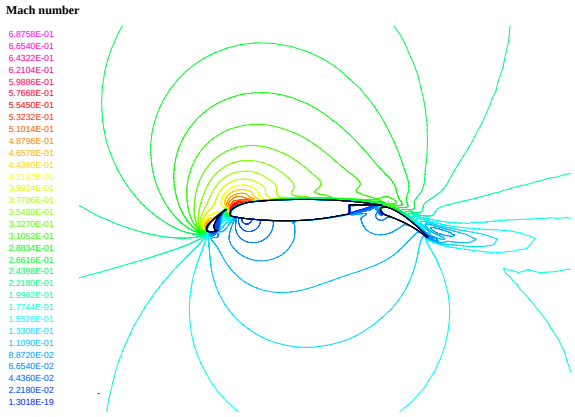

Fig. 5b: Computed Mach number contours for turbulent flow over a four-element airfoil (M_∞=0.1, Re=10 million, α=10°).

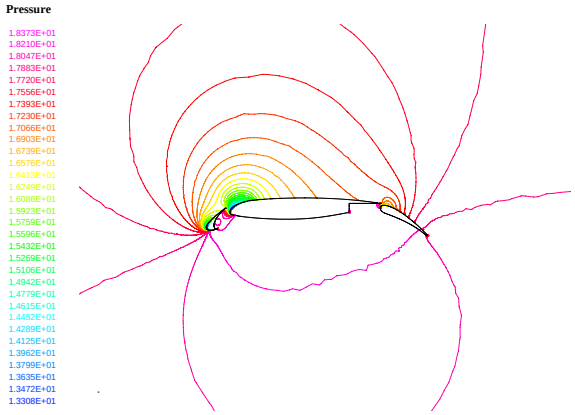

Fig. 5c: Computed pressure contours for turbulent flow over a Douglas 3-element airfoil (M_∞=0.1, Re=10 million, α=10°).

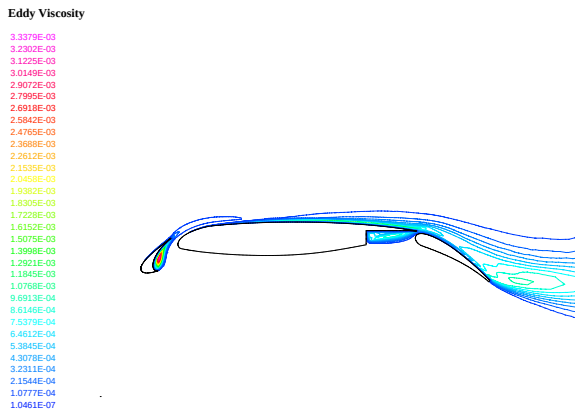

Fig. 5d: Computed eddy viscosity contours for turbulent flow over a Douglas 3-element airfoil (M_∞=0.1, Re=10 million, α=10°).

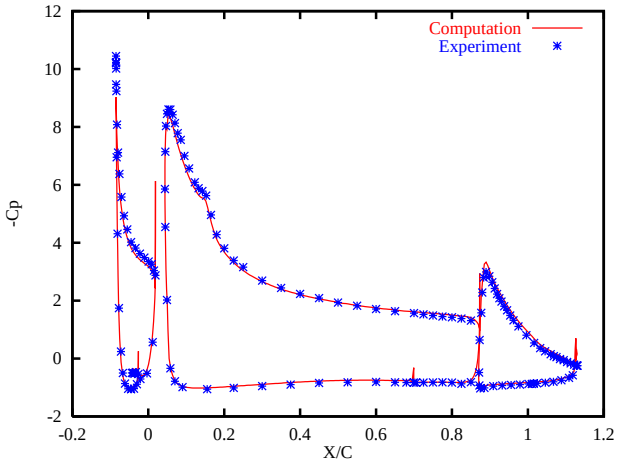

Fig. 5e: Comparison of computed surface pressure coefficient with experimental data for turbulent flow over a Douglas 3-element airfoil.

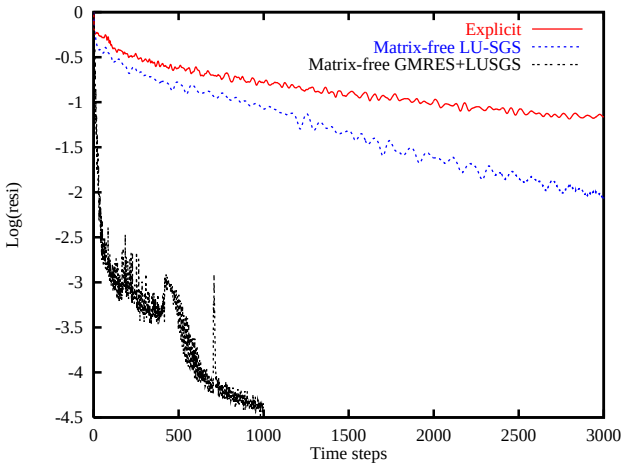

Fig. 5f: Residual convergence history versus time steps for turbulent flow over a Douglas 3-element airfoil using different methods: explicit, matrix-free LU-SGS and matrix-free GMRES+LU-SGS.

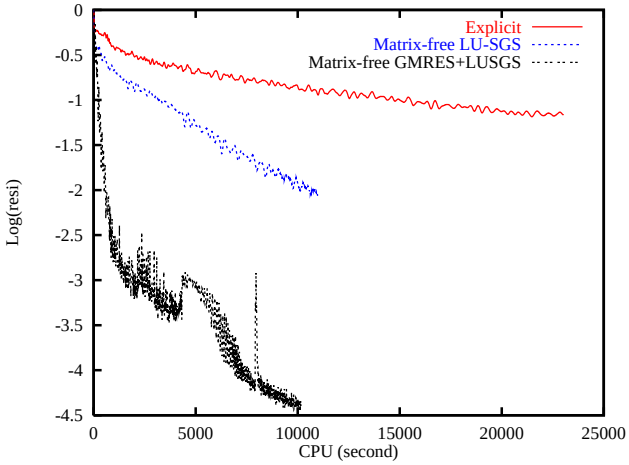

Fig. 5g: Residual convergence history versus CPU time for turbulent flow over a Douglas 3-element airfoil using different methods: explicit, matrix-free LU-SGS and matrix-free GMRES+LU-SGS.

Test Case 3. Turbulent flat plate

This test case involves modeling the turbulent flow over a flat plate. This simple problem is chosen to validate the implementation of the Spalart and Allmaras turbulence model and assess the accuracy of the numerical solution for 3D computations. The mesh used to compute the flat plate boundary layer is depicted in Fig. 6a. It contains 212,295 elements, 39,338 grid points, and 7,576 boundary points. The first point normal to the plate is located at a distance of 0.82E-06. The free stream Mach number is 0.4, and the Reynolds number of flow based on the length of the plate is 1 million. The resulting velocity profiles at x=0.5 using both median dual and containment dual control volumes, plotted in Fig. 6b, are compared with the well known one seventh power law distribution. It can be seen that the solution using containment dual control volumes is more accurate than the one using median dual control volumes. Note that the median dual control volume solution looks fairly good, as the mesh does not contain very high stretching ratios.

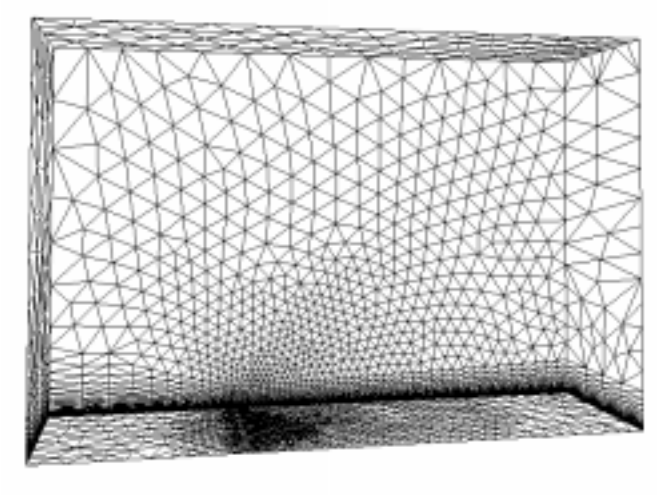

Fig. 6a: Surface mesh used for computing turbulent flat plate boundary layer problem (nelem=212,295, npoin=39,338, nboun=7,576).

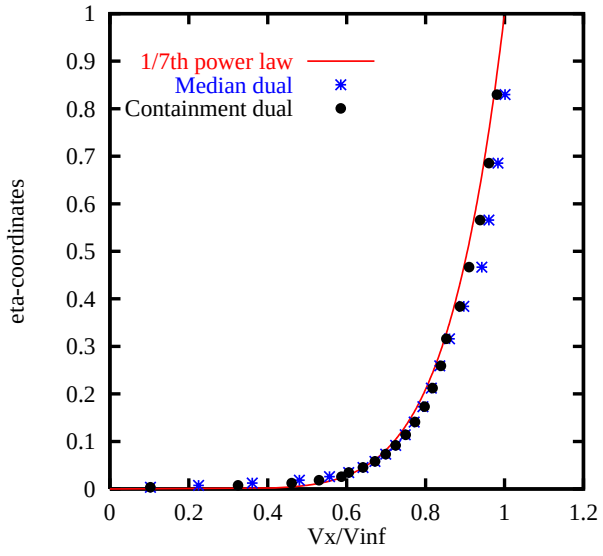

Fig. 6b: Computed boundary layer velocity profile at x=0.5 over a flat plate at $M_\infty = 0.001$, $\alpha = 0.0$, and Re=1,000,000.

Test Case 4. Turbulent flow past NACA0012 airfoil

This test case computes turbulent flow over a NACA0012 airfoil at a Mach number of 0.3, a chord Reynolds number of 1.86 million, and an incidence of 3.59 degrees. This is a 3D simulation of a 2D problem. The mesh used in the computation, shown in Fig. 7a, contains 2,014,616 elements, 359,860 grid points, and 45,865 boundary points. The minimum normal spacing at the wall is about 0.94E-05. The computed

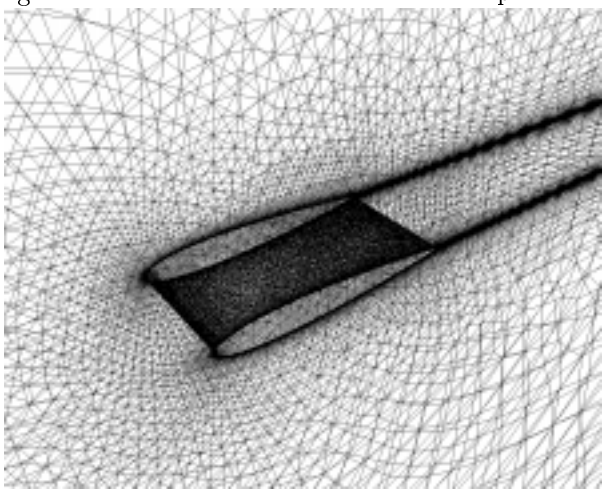

Fig. 7a: Mesh used for computing turbulent flow past a naca0012 airfoil (nelem=2,014,616, npoin=359,860, nboun=45,865).

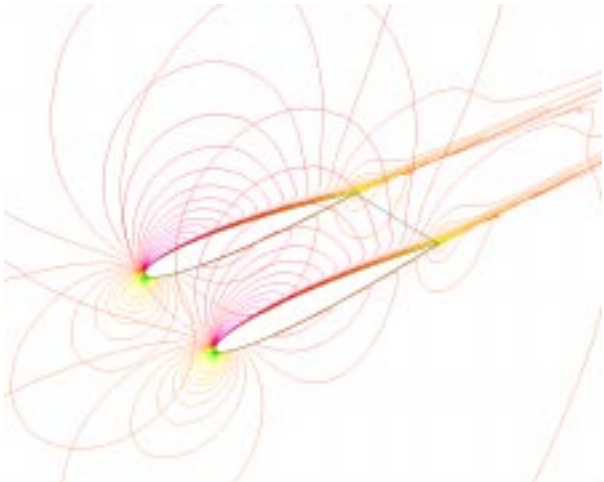

Fig. 7b: Computed Mach number contours on the surface of the airfoil at $M_\infty = 0.3$, $\alpha = 3.59°$, and Re=1,860,000.

Mach number and eddy viscosity contours in the flow field are displayed in Figs. 7b-c, respectively. A smooth distribution of eddy viscosity throughout the boundary-layer, and vanishingly small values in the

inviscid regions of flow are observed. The computed surface pressure distributions using both median dual and containment dual control volumes are compared with the experimental measurement[30] in Fig. 7d. Although both yield fairly good results, the solution using containment dual control volumes is more accurate than the one using median dual control volumes.

Fig. 7c: Computed eddy viscosity contours in the flow field at $M_\infty = 0.3$, $\alpha = 3.59°$, and Re=1,860,000.

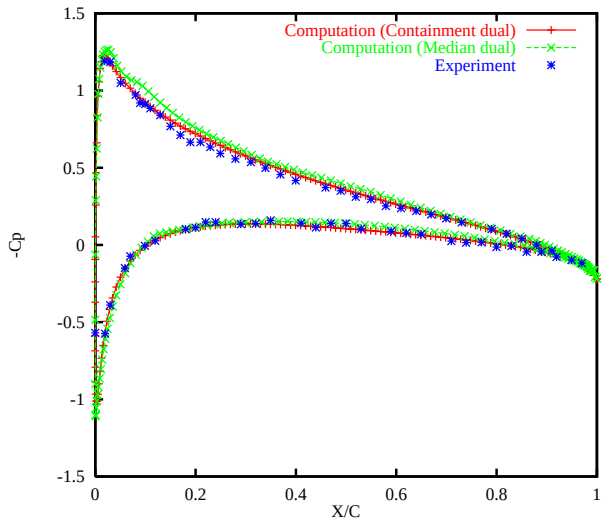

Fig. 7d: Comparison of pressure coefficient distribution between the computational result and experimental data at $M_\infty = 0.3$, $\alpha = 3.59°$, and Re=1,860,000.

Test Case 5. Transonic turbulent flow past a M6wing

The fifth test is the well-documented case of transonic flow over a ONERA M6 wing configuration. The flow solutions are presented at a Mach number of 0.84, a Reynolds number of 18.2 millions, and an angle of attack of 3.06°. The mesh used in the computation consists of 2,953,333 elements, 504,790 grid points, and 24,166 boundary points. The minimum normal spacing at the wall is about 2.0E-06 and the biggest stretching ratio of the tetrahedra is 11,961. The upper surface meshes are shown in Fig. 8a. The computed pressure contours on the upper surfaces, displayed in Fig. 8b, clearly show the sharply captured lambda-type shock structure formed by the two inboard shock waves, which merge together near 90% semispan to form the single strong shock wave in the outboard region of the wing. Figs. 8c-8h show the comparison of experimental data[31] and computed pressure coefficient distributions obtained using hybrid grid, tetrahedral grid with median dual control volume, and tetrahedral grid with containment dual control volume at six spanwise stations, respectively. It can be seen that the solutions on the hybrid and tetrahedral grid with containment dual control volumes are virtually identical while the median dual control volume solution exhibits more smearing in the region of the shock waves and leading edges. This example clearly indicates that on tetrahedral grids, the containment dual control volume can attain the same accuracy as the corresponding hybrid grid solution.

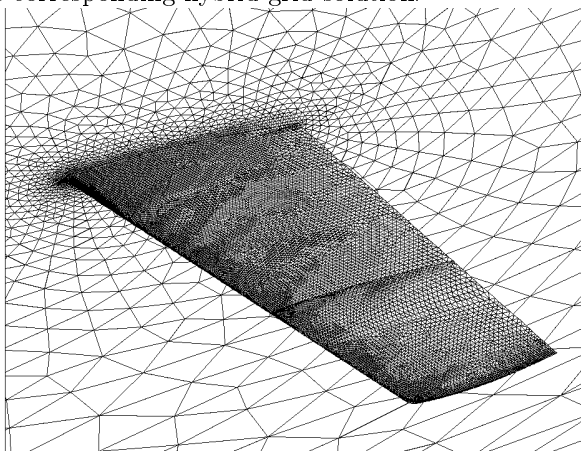

Fig. 8a: Mesh used for computing turbulent flow past M6wing (nelem=2,953,333, npoin=504,790, nboun=24,166).

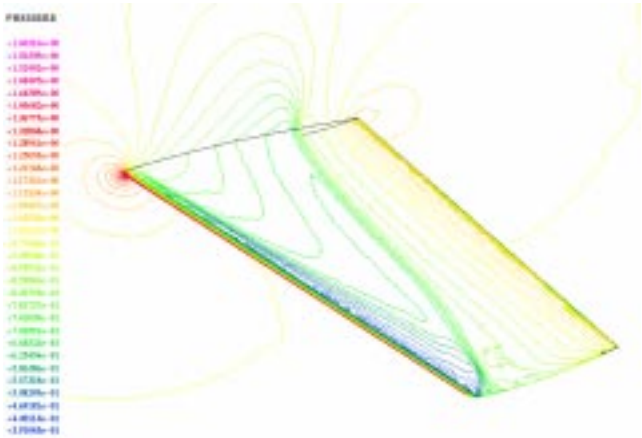

Fig. 8b: Computed pressure contours on the surface of the ONERA M6wing at $M_\infty = 0.84$, $\alpha = 3.06°$, and Re=18,200,000.

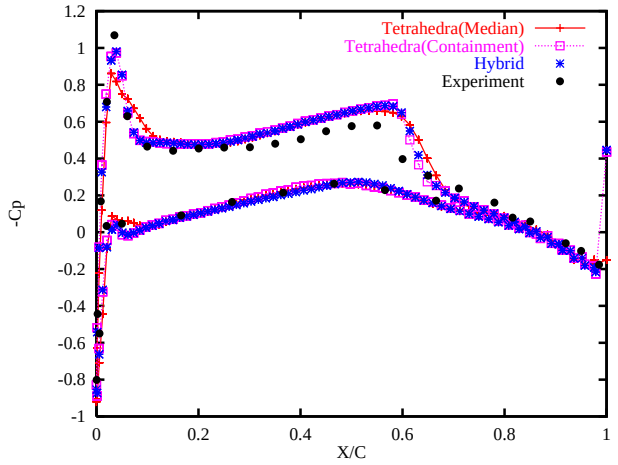

Fig. 8c: Comparison between experimental and computed pressure coefficient distributions for wing section at 20% semispan.

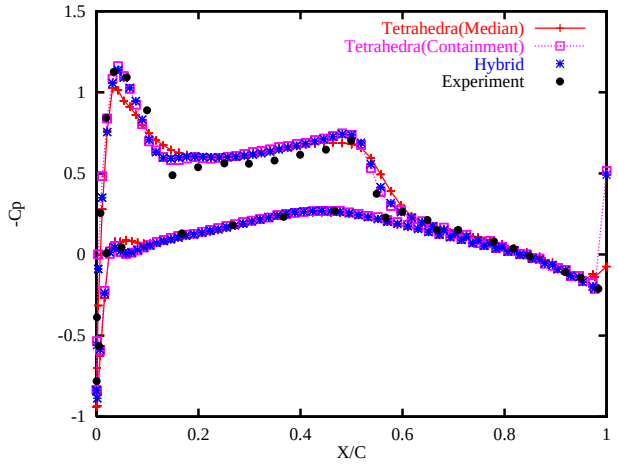

Fig. 8d: Comparison between experimental and computed pressure coefficient distributions for wing section at 44% semispan.

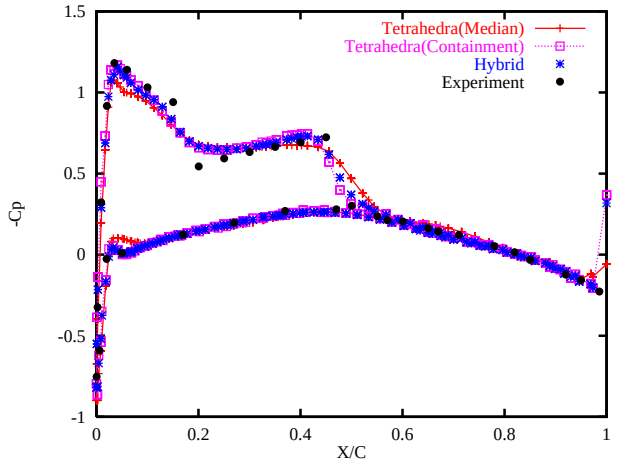

Fig. 8e: Comparison between experimental and computed pressure coefficient distributions for wing section at 65% semispan.

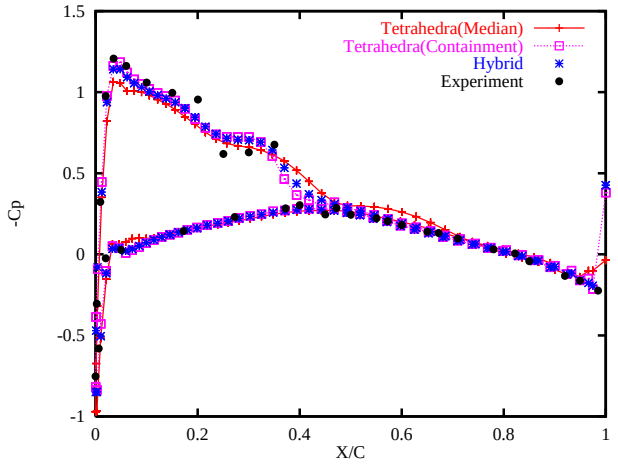

Fig. 8f: Comparison between experimental and computed pressure coefficient distributions for wing section at 80% semispan.

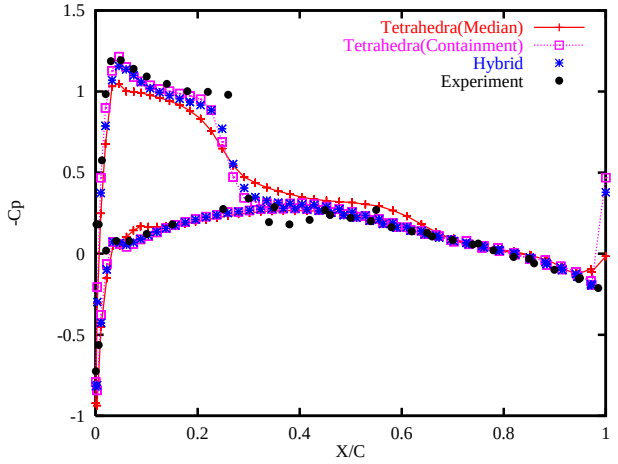

Fig. 8g: Comparison between experimental and computed pressure coefficient distributions for wing section at 90% semispan.

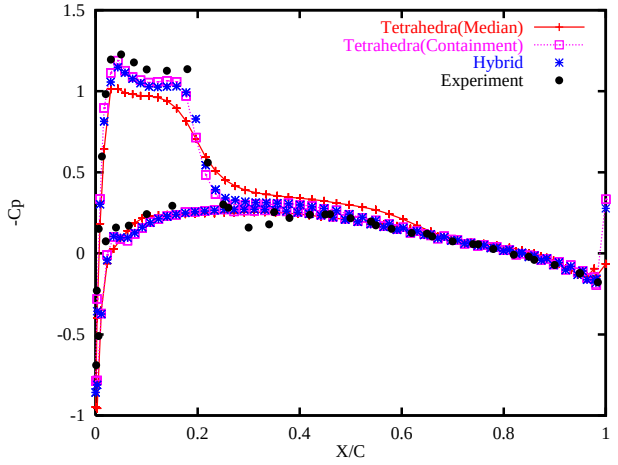

Fig. 8h: Comparison between experimental and computed pressure coefficient distributions for wing section at 95% semispan.

Test Case 6. Transonic turbulent flow past a M5 airplane

Finally, as an example of application, the developed method has been used to compute the transonic flow around the ONERA M5 configuration. The mesh used in the computation, which contains 2,953,333 tetrahedra, 504,790 grid points, and 24,166 boundary points for the half-span airplane, is shown in Fig. 9a. The computation is performed at a Mach number of 0.84, a Reynolds number of 1 million, and an angle of attack of -1°. The computed pressure contours on the surface of the airplane are displayed in Fig. 9b. Some of the features occurring in this flow regime, such as the canopy and wing shocks, are well captured.

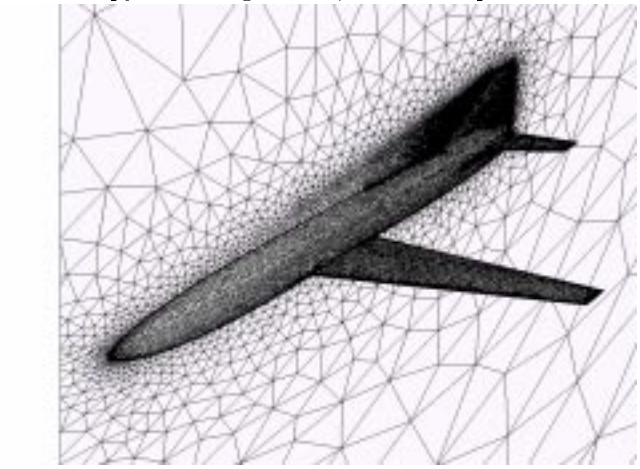

Fig. 9a. Surface mesh used for computing turbulent flow past M5 airplane (nelem=2,104,803, npoin=368,122, nboun=33,234).

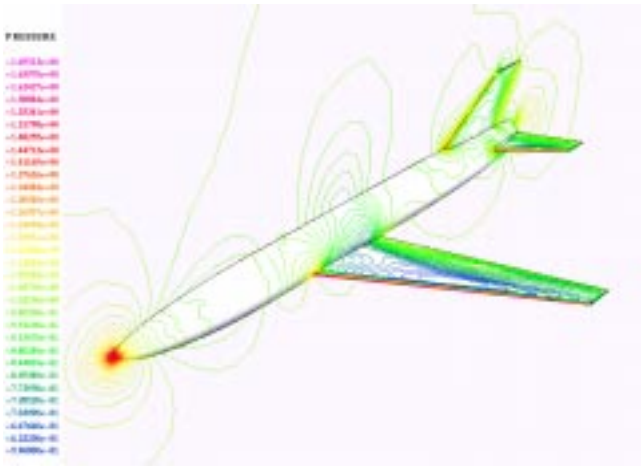

Fig. 9b. Computed pressure contours on the surface of the M5 airplane at M_∞=0.84, $\alpha = -1.0°$, and Re=1,000,000.

5. CONCLUSIONS

An accurate, fast, matrix-free implicit method has been presented to solve compressible turbulent flow problems using Spalart and Allmaras one equation turbulence model on unstructured meshes. It is found that the finite volume approximation based on the classic median dual control volume for convective fluxes suffers from excessive numerical diffusion on highly stretched triangular and tetrahedral grids. The use of containment dual based control volume approximation results in dramatic improvement in accuracy on highly stretched unstructured grids, and allows unstructured grid to produce same quality solutions as its hybrid counterpart. The matrix-free implicit method, GMRES+LU-SGS, can be used efficiently to compute compressible turbulent flow problems. The numerical results indicate that the present method provides an accurate, viable, fast, and robust algorithm for computing compressible turbulent flows on unstructured meshes.

References

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

[2]A. Haselbacher, J. J. McGuirk, and G. J. Page, "Finite-Volume Discretization Aspects for Viscous Flows for Mixed Unstructured Meshes," *AIAA Journal*, 37(2), pp. 177-184, 1999.

[3]C. Viozat, C. Held, K. Mer, and A. Dervieux, "On Vertex-Centered Unstructured Finite-Volume Methods for Stretched Anisotropic Triangulations," INRIA Repart de Recherche No. 3464, 1998.

[4]C. P. Rumsey, B. van Leer, and P. L. Roe, "A Multidimensional Flux Function with Applications to the Euler and Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 105, No. 2, pp. 306-323, 1993.

[5]P. L. Roe, "Discrete Models for the Numerical Analysis of Time-Dependent Multidimensional Gas Dynamics," *Journal of Computational Physics*, Vol. 63, No. 2, pp. 458-476, 1986.

[6]H. Deconinck, C. Hirsch, and J. Peuteman, "Characteristic Decomposition Methods for the Euler Equations," *Lecture Notes in Physics*, Vol. 6, pp. 216-221, 1986.

[7]P. Van Ransbeeck, and C. Hirsch, "New Upwind Dissipation Models with a Multidimensional Approach," AIAA Paper 93-3304, 1993.

[8]I. H. Parpia, and D. J. Michalek, "A Nearly-Monotone Genuinely Multidimensional Scheme for the Euler Equations," AIAA Paper 92-0325, 1995.

[9]X. D. Zhang, J. Y. Tŕepanier, M. Reggio, A. Benmeddour, and R. Camaero, "Grid Influence on Upwind Schemes for the Euler and Navier-Stokes Equations," *AIAA Journal*, Vol. 34, No. 4, pp. 717-727, 1996.

[10]K. Nakahashi, D. Sharov, S. Kano, and M. Kodera, "Applications of Unstructured Hybrid Grid Method to High-Reynolds Number Viscous Flows," *Int. J. Numer. Meth. Fluids*, Vol. 31, 1999.

[11]A. Haselbacher, and J. Blazek, "On the Accurate and Efficient Discretization of the Navier-Stokes Equations on Mix Grids," AIAA Paper 99-3363, 1999.
[12]T. J. Barth, and S. W. Linton, "An unstructured Mesh Newton Solver for Compressible Turbulent Flows and Its Parallel Implementation," AIAA Paper 95-0221, 1995.
[13]H. Luo, J.D. Baum, and R. Löhner, "A Fast, Matrix-free Implicit Method for Compressible Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 146, 1998.
[14]H. Luo, J.D. Baum, and R. Löhner, "An Accurate, Fast, Matrix-free Implicit Method for Computing Unsteady Flows on Unstructured Grids," AIAA Paper 99-0937, 1999.
[15]P.R. Spalart, and S.R. Allmaras, "A One-equations Turbulence Model for Aerodynamic Flows", AIAA Paper 92-0439, 1992.
[16]M. S. Liou, "Progress towards an Improved CFD Method: AUSM+," *Journal of Computational Physics*, Vol. 129, 1996.
[17]B. van Leer, "Towards the Ultimate Conservative Difference Scheme, II. Monotonicity and Conservation Combined in a Second Order Scheme," *Journal of Computational Physics*, Vol. 14, 1974.
[18]P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981.
[19]A. Jameson, and S. Yoon, "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, 1987.
[20]M. Soetrisno, S. T. Imlay, and D. W. Roberts, "A Zonal Implicit Procedure for Hybrid Structured-Unstructured Grids," AIAA Paper 94-0617, 1994.
[21]I. Men'shov, and Y. Nakamura, "An Implicit Advection Upwind Splitting Scheme for Hypersonic Air Flows in Thermochemical Nonequilibrium," *6th Int. Symp. on CFD*, 1995.
[22]D. Sharov, and K. Nakahashi, "Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier-Stokes Computations," AIAA Paper 97-2102, 1997.
[23]Y. Saad, and M. H. Schultz, "GMRES: a Generalized Minimal Residual Algorithm for Solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comp.*, Vol. 7, No 3 (1988), pp 89-105.
[24]V. Venkatakrishnan and D. J. Mavriplis, "Implicit Solvers for Unstructured Meshes," *Journal of Computational Physics* Vol. 105, pp. 83-91, 1993.
[25]D. D. Knight, "A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting," AIAA Paper 93-0875, 1993.
[26]D. L. Whitaker, "Three-dimensional Unstructured Grid Euler Computations Using a Fully-Implicit, Upwind Method," AIAA Paper 93-3337, 1993.
[27]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.
[28]P. H. Cook, M. A. McDonald, and M. C. P. Firmin, "Aerofoil RAE 2822 pressure distributions, and boundary layer and wake measurements," AGARD Advisory Report No. 138, May 1979.
[29]W. O. Valarezo, C. J. Dominik, R. J. McGhee, and W. L. Goodman, "High Reynolds Number Configuration Development of a High-Lift Airfoil," in *AGARD Conference on High-Lift Aerodynamics*, Banff, Canada, October, 1992.
[30]"Experimental Data Base for Computer Program Assessment", AGARD Advisory Report, No. 138, May 1979.
[31]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.