INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **33**: 657–680

A fast nested multi-grid viscous flow solver for adaptive Cartesian/Quad grids

Z. J. Wang*[,1]

CFD Research Corporation, 215 Wynn Drive, Huntsville, AL, U.S.A.

SUMMARY

A nested multi-grid solution algorithm has been developed for an adaptive Cartesian/Quad grid viscous flow solver. Body-fitted adaptive Quad (quadrilateral) grids are generated around solid bodies through 'surface extrusion'. The Quad grids are then overlapped with an adaptive Cartesian grid. Quadtree data structures are employed to record both the Quad and Cartesian grids. The Cartesian grid is generated through recursive sub-division of a single root, whereas the Quad grids start from multiple roots—a forest of Quadtrees, representing the coarsest possible Quad grids. Cell-cutting is performed at the Cartesian/Quad grid interface to merge the Cartesian and Quad grids into a single unstructured grid with arbitrary cell topologies (i.e., arbitrary polygons). Because of the hierarchical nature of the data structure, many levels of coarse grids have already been built in. The coarsening of the unstructured grid is based on the Quadtree data structure through reverse tree traversal. Issues arising from grid coarsening are discussed and solutions are developed. The flow solver is based on a cell-centered finite volume discretization, Roe's flux splitting, a least-squares linear reconstruction, and a differentiable limiter developed by Venkatakrishnan in a modified form. A local time stepping scheme is used to handle very small cut cells produced in cell-cutting. Several cycling strategies, such as the saw-tooth, W- and V-cycles, have been studies. The V-cycle has been found to be the most efficient. In general, the multi-grid solution algorithm has been shown to greatly speed up convergence to steady state—by one to two orders. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: adaptive grid; multi-grid solver; viscous flow

1. INTRODUCTION

It is generally recognized that unstructured grid-based computational fluid dynamics (CFD) algorithms offer the best promise for automation in fluid flow simulations. The last decade has seen a tremendous progress in unstructured grid methods. Types of unstructured grids include classical triangular or tetrahedral grids [1–5], quadrilateral or hexahedral grids [6], prismatic grids [7], or mixed grids [8,9]. The most appealing properties of unstructured grids are the

* Correspondence to: CFD Research Corporation, 215 Wynn Drive, Huntsville, AL 35805, U.S.A.
[1] E-mail: zjw@cfdrc.com

Received 1 October 1998
Revised 4 August 1999

geometric flexibility and the ease in which the grid can be adapted according to flow features. Tetrahedral grids are the easiest to generate. Many well-known grid generation algorithms, such as the advancing front [10] and the Delauney triangulation method [11], have been developed to generate tetrahedral grids for complex geometries. However, experience has indicated that tetrahedral grids are not as efficient and/or accurate as hexahedral or prismatic grids for viscous boundary layers. On the other hand, prismatic grids and hexahedral grids can resolve boundary layers more efficiently, however, they are more difficult to generate than tetrahedral grids. Many CFD researchers have come to the conclusion that mixed grids (or hybrid grids) are the way to go.

Recently, there has been a renewed interest in using Cartesian grids for complex geometries [12–17]. Coupled with a tree-based data structure and grid adaptation, with respect to both the geometry and the flow field, these methods have been demonstrated to be very viable tools for inviscid flows, with very complex geometry. The main advantages of the Cartesian grid methods are the following: (1) automatic grid generation, (2) automatic grid adaptation, and (3) simplified data structure. With the adaptive Cartesian grid approach, grid-independent solutions have been obtained [13]. To achieve the same solutions with a non-adaptive mesh would be prohibitively expensive.

One obvious drawback of the adaptive Cartesian grid method is its inability to support directional grid adaptation required in viscous boundary layer-type flow problems. Conventional grid adaptations in a boundary layer are not only too expensive but inefficient as well [14]. Furthermore, the irregular cut cells near the solid wall boundaries have been shown to produce non-positive numerical scheme for the Navier–Stokes equations and may cause convergence problems. To eliminate this problem, Karman [18] introduced the adaptive Cartesian/prism method, in which an adaptive Cartesian grid and a fixed prism grid are mixed to tackle viscous flows. Very complex flow problems were tackled successfully. One defect of this method is the use of a fixed prism grid, which partly negates the effectiveness of grid adaptation performed in the Cartesian grid. This defect has motivated the present author to develop a hybrid adaptive Cartesian/adaptive prism grid methodology. This method has been successfully demonstrated in the two-dimensional case in an early study [19]. Grid-independent inviscid and viscous solutions have been obtained through automatic grid adaptation.

One distinctive advantage of a structured grid over an unstructured grid is that multiple levels of coarse grids can be easily generated through coarsening in each co-ordinate direction. These coarse grids are nicely nested (i.e., fine grid cells are embedded in the cells of the coarse grid). Very efficient multi-grid solution algorithms can then be implemented. For an unstructured grid (not the tree-type), coarse grids need to be either generated independently [20] or conglomerated [21] through some very sophisticated algorithms. The tree-based adaptive Cartesian grid has the same advantage as the structured grid in that many levels of coarse grids can be easily generated through tree traversals from leaves to roots simply because the computational grid is generated through recursive sub-divisions of a base grid. In the case of an adaptive Cartesian grid, the computational grid is produced through sub-divisions of a single cell.

In this paper, an efficient multi-grid algorithm for the adaptive Cartesian/Quad (quadrilateral) grid viscous flow solver is developed. For the sake of completeness, the issues concerning the adaptive Cartesian/Quad grid methodology are discussed first. Then, coarse grid generation through reverse tree-traversal is described. Next, the full approximation storage (FAS) multi-grid algorithm for the adaptive Cartesian/Quad grid solver is developed, and grid adaptation criteria are discussed. After that, several test cases will be shown to demonstrate the performance of the algorithm. Finally, conclusions from the study are summarized.

2. ISSUES ON ADAPTIVE CARTESIAN/QUAD GRID GENERATION

2.1. Data structure

A hierarchical cell-based Quadtree data structure [13] is employed for both the Cartesian and the Quad grids, i.e., parent cells can be refined by division into four children. Each node of the Quadtree represents a Cartesian or Quad cell. To enable recursive tree traversals in both directions (from parent to children or otherwise), each cell has a pointer to its parent cell (if one exists) and to its four children (if they exist). The cells farthest down the hierarchy, i.e., the finest grid cells with no children, are the cells on which the calculation takes place. Since the Cartesian grid is generated from a single cell through recursive sub-divisions, a single Quadtree is employed for the Cartesian cells. Multiple Quadtrees are used for the Quad cells with the roots of these Quadtrees corresponding to the coarsest Quad cells. Mutual relations of the coarsest Quad grids are stored in a connectivity array.

Each tree node also stores the pointers to the four corner vertices. With these vertices, the geometry of the cell represented by the tree node is well defined. These pointers are automatically updated during recursive grid sub-divisions.

2.2. Adaptive Quad grid generation

A unique feature of the current approach is that adaptive Quad grids are generated around solid bodies, such that viscous boundary layers can be properly resolved through non-isotropic grid adaptations. In this study, two-dimensional geometries are represented by segments of cubic B-Spline curves and straight lines. The geometry is preserved at all levels of grid adaptations. The generation of the Quad grids is fulfilled in the following steps:

2.2.1. Boundary discretization. The boundary curves are discretized based on three user-specified parameters, $\mathrm{d}s_{\min}$, $\mathrm{d}s_{\max}$, α. The lengths of the boundary segments are always bounded by $\mathrm{d}s_{\min}$ and $\mathrm{d}s_{\max}$. An angular criterion is used to cluster more grid points near high curvature parts of the curve. This criterion dictates that the angle between two neighboring segments should be less than α.

2.2.2. Surface extrusion. The technique to generate the Quad grid is similar to the hyperbolic or advancing layer grid generation approaches [22], i.e., to extrude the surface grid along the surface normal directions. For concave surfaces, however, it is possible that surface normals cross into each other, resulting in folding Quad cells with negative volumes. This problem is eliminated by first smoothing the surface normals in the neighborhood of the crossings. If the problem of folding Quad cells still exists the normal distances of Quad cells are reduced locally until no crossings occur. By connecting each point on the body surface with the corresponding point of the ‘extruded’ surface, one obtains one layer of quad cells. The initial height of the

Quad is prescribed by the user based on the thickness of the boundary layers depending on the Reynolds number. This layer of Quads is further divided into several layers with grid clustering near the body surface. The resultant Quad mesh is then the root mesh that further tree-based grid refinements are based on.

Ideally the outer most layer of the Quad grid should be isotropic to match the adaptive Cartesian cells. To achieve this, one can use different local viscous layer heights depending on the local boundary curvature and boundary discretization. This approach may be difficult to implement when multiple bodies are close to each other. In this paper, a fixed viscous layer height is used everywhere except in regions where boundary normals cross in each other, or multiple bodies are close to each other. In those regions, the viscous layer height is locally reduced to enable the generation of a valid mesh. This approach can always produce a valid mesh. However, the cells near the outer boundary of the Quad mesh are usually anisotropic. The strategy in this paper is to reduce the effects of grid non-smoothness through solution based grid adaptations. It will be shown later that smooth solutions are obtained near the Cartesian/Quad interfaces.

2.2.3. Recursive sub-divisions. The root mesh is usually very coarse and several levels of automatic grid refinement are performed first. After that, the grid is adapted according to local curvatures until a more strict angular criterion is satisfied. Because the grid will also be adapted according to the flow field, the quality of the initial grid is not as important as that of a fixed grid. As long as the initial grid adequately resolves the geometric features, the flow features will be resolved through flow-based grid adaptations.

2.2.4. Grid smoothing. The resultant grid is then further smoothed such that some unwanted properties are eliminated. For example, the cell level difference between any neighbors should be at most one.

2.3. Adaptive Cartesian grid generation

The Cartesian grid is generated from one large root cell, which covers the complete flow field through recursive sub-divisions. The roots Cartesian cell is sub-divided recursively until a user specified minimum grid resolution is obtained. Then the Cartesian cells, which are intersected by the outer boundaries of the Quad grids, are further divided until these Cartesian grid cells have as comparable grid resolutions as the Quad cells.

2.4. Cell cutting

Due to the appearance of the Quad grids, holes are cut out of the Cartesian grid using the outer boundaries of the Quad grids. The basic algorithm for cell cutting is line–line intersection. The arbitrarily shaped polygon cells resulting from cell cutting are called cut cells. Cells that lie completely inside the Quad grids are excluded from the computational domain. An example of cell cutting is shown in Figure 1. The final computational grid consists of cells with an arbitrary number of edges (e.g., an edge with a hanging node is actually treated as two separate edges). All cells (Cartesian, Quad, and cut cells) are treated in the same manner. It will been shown that this uniform treatment is highly efficient when reconstruction coefficients are stored.

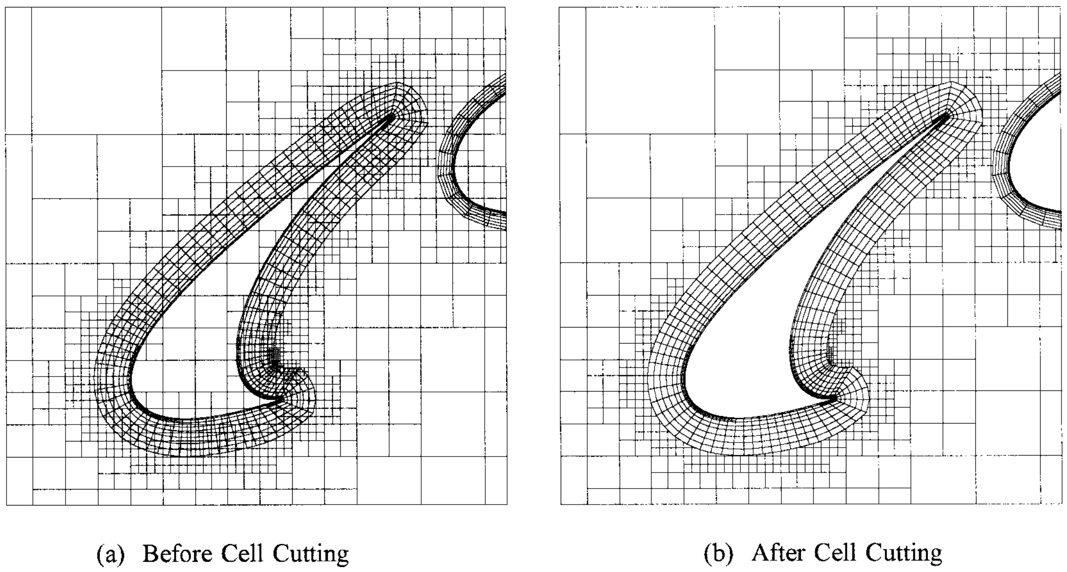


Figure 1. Illustration of cell cutting at Cartesian/Quad interface.

3. COARSE GRID GENERATION

Since the Quadtree data structure has all levels of grid cells built in (from root to leaf) it is easy to construct several levels [3–5] of nested coarse grids for the multi-grid solution procedure. A cell's level in the tree is defined as the number of successive parents to the root cell. For M multi-grid levels, the cells on multi-grid level M correspond to the finest level, containing all the computational cells without children, i.e., the leaves of the Quadtrees. Therefore, it is obvious that cells on the same multi-grid level can be different cell levels in the tree. In constructing multi-grid level $M-1$ from level M, cells that are located in the same Quad and that all belong to level M are allowed to coarsen. As a consequence, some cells are not coarsened and are included in more than one multi-grid level. To illustrate this situation, an example of grid coarsening is shown in Figure 2.

It is seen that cell A belongs to one multi-grid level, cell B is included in two multi-grid levels, and cell C occupies three multi-grid levels.

Cell cutting at the Cartesian/Quad interface can cause a slight complication, which needs special treatment. A situation is shown in Figure 3. When the fine grid is coarsened, the coarse grid cell is intersected by the interface twice. In this case, two separate coarse grid cut cells (denoted by the shadowed cells) are produced for the Quad. This situation is allowed because flow variables are associated with cells in the cell list rather than the Quads in the Quadtree, since the flow solver is directly operated on the computational grid with cell, face, and node lists generated from the Quad trees rather than the Quad trees themselves.

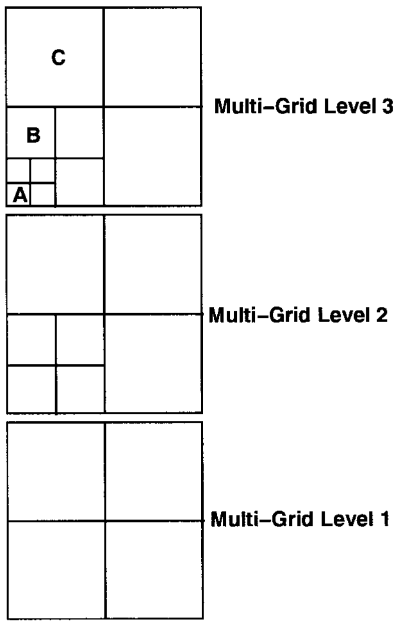


Figure 2. Illustration of grid coarsening for multi-grid.

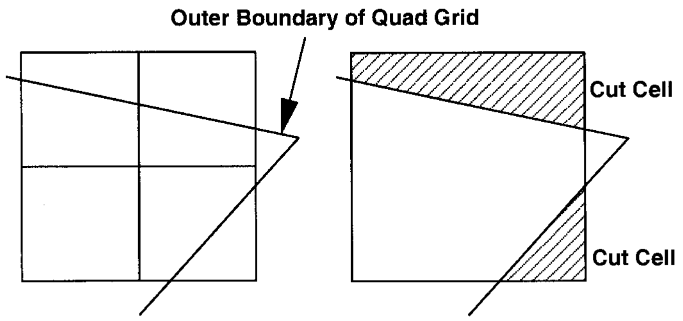


Figure 3. Illustration of grid coarsening for a cut cell.

4. MULTI-GRID FLOW SOLVER

The Navier–Stokes equations in integral form can be written as

$$\int_V \frac{\mathrm{d}Q}{\mathrm{d}t}\,\mathrm{d}V + \int_S (F - F_\mathrm{v})\,\mathrm{d}S = 0 \tag{1}$$

where Q is the vector of conserved variables, and F and F_v are the inviscid and viscous flux vector respectively. In all the calculations to be shown the perfect gas law is assumed. The integration of Equation (1) in a control volume, V, gives

$$\frac{d\bar{Q}}{dt} V + \sum_f F_f S_f = \sum_f F_{vf} S_f \tag{2}$$

where $\bar{Q}$ is the cell averaged conserved variable, and F_f and $F_{v,f}$ are the numerical inviscid and viscous fluxes at face f. The overbar of $\bar{Q}$ will be dropped from here on. Several ingredients of the flow solver—reconstruction, Riemann solver, viscous flux calculation and time integration—will be briefly described in the following sub-sections.

4.1. Reconstruction

In a cell centered finite volume procedure, flow variables are known in a cell average sense. No indication is given as to the distribution of the solution over the control volume. In order to evaluate the flux through a face, flow variables are required at both sides of the face. This task is fulfilled by reconstruction. A least-squares reconstruction method [23] is selected in this study. This reconstruction is capable of preserving a linear function on an arbitrary grid. To be more specific, we seek to reconstruct the gradient (q_x, q_y) of current cell c (q indicates primitive variables) from data at first face neighbor cells, i.e.,

$$q(x, y) = q_c + q_x(x - x_c) + q_y(y - y_c) \tag{3}$$

with

$$q_x = \frac{1}{\Delta}\left[I_{yy} \sum_n (q_n - q_c)(x_n - x_c) - I_{xy} \sum_n (q_n - q_c)(y_n - y_c) \right]$$

$$q_y = \frac{1}{\Delta}\left[-I_{xy} \sum_n (q_n - q_c)(x_n - x_c) + I_{xx} \sum_n (q_n - q_c)(y_n - y_c) \right]$$

$$I_{xx} = \sum_n (x_n - x_c)^2$$

$$I_{xy} = \sum_n (x_n - x_c)(y_n - y_c)$$

$$I_{yy} = \sum_n (y_n - y_c)^2$$

$$\Delta = I_{xx} I_{yy} - I_{xy}^2$$

where n indicates the support neighbor cells. It can be seen that the reconstruction can be performed efficiently with one loop over neighboring cells once I_{xx}, I_{xy}, and I_{yy} are stored for each cell. A limiting procedure is employed to avoid numerical oscillations at steep gradients. A limiter ϕ is applied to Equation (3) to give

$$q(x, y) = q_c + \phi \nabla q \cdot (\mathbf{r} - \mathbf{r}_c) \tag{4}$$

A single limiter ϕ is used for all variables. A limiter shown to have a good convergence rate was chosen in this study [3]. It is well known that limiters hinder convergence to steady state. Venkatakrishnan found that limiting in near constant flow regions caused the convergence to stall, and suggested the following limiter:

$$\phi = \frac{1}{\Delta_-} \left[\frac{(\Delta_+^2 + \epsilon^2)\Delta_- + 2\Delta_-^2 \Delta_+}{\Delta_+^2 + 2\Delta_-^2 + \Delta_+ \Delta_- + \epsilon^2} \right] \tag{5}$$

with

$$\Delta_- = q - q_c$$

$$\Delta_+ = \begin{cases} q_{\max} - q_c & \text{if } \Delta_- > 0 \\ q_{\min} - q_c & \text{if } \Delta_- < 0 \end{cases}$$

$$\epsilon^2 = (K\Omega)^3 \tag{6}$$

where K is a constant and Ω is the mesh size, $q_{\max}$ and $q_{\min}$ are the maximum and minimum values over the current cell and its neighbors. Note that ϵ has the dimension of q itself. It is obvious that K is very difficult to determine since it is dependent not only on the mesh size but also on the flow variable. With an adaptive grid, the mesh sizes can vary drastically with the ratio between the biggest and the smaller cell size exceeding six orders. The parameter ϵ^2 as defined in (6) can vary by 18 orders or more, causing not enough limiting at some large cells, and too severe limiting at some small cells. Numerical tests showed that it was nearly impossible to select an appropriate constant K that yields smooth and stable solutions. The difficulty in selecting a proper constant K may also be attributed to the fact that the adaptive Cartesian mesh is intrinsically non-smooth. The sizes of many neighboring cells differ by a factor of 2. As a result, parameter ϵ^2 can differ by a factor of 8 in neighboring cells causing convergence difficulties. In order to relieve this difficulty, ϵ is redefined in this paper as

$$\epsilon = \epsilon'(q^{\max} - q^{\min}) \tag{7}$$

with $\epsilon' \in (001, 0.20)$ and $q^{\max}$, $q^{\min}$ are the maximum and minimum of q over the entire computational domain. With this definition, oscillations exceeding a certain percentage of the maximum range of the flow field are limited, and ϵ' is a non-dimensional quantity. Note that parameter ϵ is independent of the local cell size. Numerical tests indicated that the limiters only fired near discontinuities, while remaining nearly 1 at smooth flow region with the new definition. Therefore, the globally accuracy of the flow solver is not compromised. One can, however, relate the parameter ϵ to the minimum grid size in the flow domain, i.e., ϵ can be reduced when the grid is refined. For most cases, it has been found an ϵ' value of 0.05 yields good results. This modification has worked very satisfactorily for all the cases.

4.2. Riemann solver

The Riemann solver selected is Roe's approximate Riemann solver for its simplicity and accuracy for viscous flow. It is used to calculate the inviscid flux through a face with normal $\mathbf{n}$, left and right flow variables q_L and q_R, i.e.

$$F = F(q_L, q_R, \mathbf{n}) \tag{8}$$

Details of this Riemann Solver are contained in Reference [24].

4.3. Viscous flux

The viscous flux at a face can be expressed as a function of flow variables and of their gradients, i.e.

$$F_{v,f} = f_v(q_f, \nabla q_f) \tag{9}$$

where q_f is obtained through simple averages of q_{fL} and q_{fR}. If simple averages of ∇q_L and ∇q_R are used at the face, decoupling of the first face neighbor cells results. Hence, the following viscous reconstruction is developed for both efficiency and accuracy.

Let $\mathbf{m}$ be the unit normal in the face tangential direction and $\mathbf{l}$ be the unit vector connecting the left cell and the right cell of a face. The derivative of variable in the $\mathbf{m}$-direction is obtained from the cellwise inviscid reconstruction, i.e.

$$\frac{dq}{dm} = \frac{1}{2}[\nabla q_L \cdot \mathbf{m} + \nabla q_R \cdot \mathbf{m}] \tag{10}$$

where ∇q_L and ∇q_R are the gradients in the left and right cells of the face from the inviscid reconstruction. Then, dq/dl is calculated from definition, i.e.

$$\frac{dq}{dl} = \frac{q_R - q_L}{|\mathbf{r}_R - \mathbf{r}_L|} \tag{11}$$

Finally, $\nabla q = \{q_x, q_y\}$ is solved from the two equations

$$q_x \cdot m_x + q_y \cdot m_y = \frac{\mathrm{d}q}{\mathrm{d}m}$$

$$q_x \cdot l_x + q_y \cdot l_y = \frac{\mathrm{d}q}{\mathrm{d}l} \tag{12}$$

This reconstruction has the advantage of compact support (using data at the two cells sharing the face) but avoids an expensive separate reconstruction for viscous fluxes. The k–ϵ model [25] with wall functions was implemented to handle turbulent flows at high Reynolds number. The effective viscosity is used instead of the dynamic viscosity in the governing Navier–Stokes equations for turbulent flows.

4.4. Time integration with multi-grid

A FAS multi-grid algorithm has been implemented in this study [26]. To be more specific, we solve

$$L_h(Q_h) = r_h \tag{13}$$

In Equation (13), h denotes the finest mesh and $r_h = 0$. Given a sequence of grids symbolically represented by (h, $2h$, $4h$, ...) and the initial solutions on the finest mesh Q_h, the following steps are used to update the solution on the fine mesh in one cycle.

1. Improve Q_h by application of n_1 times the smoother to

$$L_h(Q_h) = r_h \tag{14}$$

2. Find an approximate solution Q_{2h} on the next coarser mesh

$$Q_{2h} = I_h^{2h} Q_h \tag{15}$$

3. Compute the defect

$$d_h = r_h - L_h(Q_h) \tag{16}$$

4. Compute

$$r_{2h} = L_{2h}(Q_{2h}) + \bar{I}_h^{2h} d_h \tag{17}$$

5. Do n_2 times. Solve

$$L_{2h}(Q_{2h}) = r_{2h} \tag{18}$$

to obtain $\bar{Q}_{2h}$

6. Correct the current solution on the fine mesh by

$$\bar{Q}_h = Q_h + I^h_{2h}(\bar{Q}_{2h} - Q_{2h}) \tag{19}$$

7. Improve $\bar{Q}_h$ by applications of n_3 times the smoother to

$$L_h(Q_h) = r_h \tag{20}$$

When solving Equation (18), the same procedure can be used on grid $2h$ and grid $4h$. This procedure is carried out until the coarsest grid, where an equation like (18) is solved up to a given degree of accuracy. The restriction operator, I^{2h}_h use in Equation (15) is a volume weighted averaging of conserved variables. The restriction operator, $\bar{I}^{2h}_h$, used in Equation (17) is a summation of all defects of the finer grid cells. The prolongation operator, I^h_{2h}, used in Equation (19) is a piecewise linear distribution. In this study, three to five levels of coarse grids were employed. It was found no further speed up was obtained with more than five levels of grids.

An explicit three-stage scheme is used as the smoother for Equation (14), i.e.

$$\begin{aligned} &Q^{(0)} = Q^n \\ &Q^{(k)} = Q^{(0)} + \alpha_k \frac{\Delta t}{V}[L(Q^{(k-1)}) - r]; \quad k = 1, 2, 3 \\ &Q^{n+1} = Q^{(3)} \end{aligned} \tag{21}$$

with $\alpha_{1,2,3} = 0.18$, 0.5, 1.0 for optimal damping for high frequency errors [27].

All the popular cycling strategies can be obtained by adjusting n_1, n_2, and n_3, i.e.

Saw-tooth cycle:	$n_1 = 1,\ n_2 = 1,\ n_3 = 0$
V-cycle:	$n_1 = 1,\ n_2 = 1,\ n_3 = 1$
W-cycle:	$n_1 = 1,\ n_2 = 2,\ n_3 = 1$

5. SOLUTION-BASED GRID ADAPTATION

To achieve automation in flow simulation, grid adaptation is essential. There are a variety of adaptation criteria in the literature. A comprehensive study was performed in Reference [13]. It was identified that a criterion based on compressibility and rotationality worked best, and it is selected in this study. Briefly, grid adaptation is based on a statistical description of two parameters

$$\tau_{ci} = |\nabla \times \mathbf{v}| d_i^{3/2} \tag{22a}$$

$$\tau_{di} = |\nabla \cdot \mathbf{v}| d_i^{3/2} \tag{22b}$$

where $\mathbf{v}$ is the velocity vector, $i = 1, 2, \ldots, N$ (N being the total number of cells), and $d = V^{1/2}$ (V is the volume of the cell). The standard deviation of both parameters is computed as

$$\sigma_c = \sqrt{\frac{\sum_{i=1}^{N} \tau_{ci}^2}{N}}, \qquad \sigma_d = \sqrt{\frac{\sum_{i=1}^{N} \tau_{di}^2}{N}} \tag{23}$$

Then the following conditions are used for grid adaptation:

1. if either $\tau_{ci} > \sigma_c$ or $\tau_{di} > \sigma_d$, cell i is flagged for refinement;
2. if both $\tau_{ci} > 1/10\sigma_c$ and $\tau_{di} > 1/10\sigma_d$, cell i is flagged for coarsening.

The parameters in Equation (22) need to be modified for stretched cells in an arbitrary direction to take into account of the cell aspect ratio [19].

Apart from grid adaptation based on divergence and curl, the first cells away from solid bodies are also adapted based on the local cell Reynolds number, which is defined by

$$Re_{\text{cell}} = \frac{\rho |v| \Delta n}{\mu} \tag{24}$$

where ρ and μ are the density and dynamic viscosity respectively, Δn is the distance from the cell center to the solid body. Cells with $Re_{\text{cell}} > 1$ are refined.

6. TEST CASES

6.1. Case 1. Transonic flow around NACA0012 airfoil ($M_\infty = 0.85$, $\alpha = 1°$)

This case was chosen as a validation case and to study the different cycling strategies in the multi-grid algorithm. This flow is characterized by two shock waves on both the upper and lower surfaces and is a good test case of shock capturing and grid adaptation capabilities of the flow solver. A coarse initial grid was first generated with the described technique. The geometry was discretized with 101 points, which were then splined by the code. The initial grid has a total of 2184 cells and is displayed in Figure 4. The outer boundary was placed at 250 chords away from the airfoil. Three extra levels of nested coarse grids are then produced for the multi-grid algorithm. Converged solutions were obtained on each level of adaptive grid before the grid was further adapted. Five levels of grid adaptation (both refinement and coarsening) were carried out. The adapted grid at level 5 has 17136 cells and is displayed in Figure 5(a). The corresponding coarse grids for this grid is shown in Figure 5(b)–(d). It is obvious that fine grid cells were automatically generated for both the strong and the week shocks and the shear layer. Furthermore, grid cells were also refined in smooth flow regions for expansion and compression waves, which is critical to resolve global flow features. The Mach number contours on the finest grid are shown in Figure 6. This picture confirms that

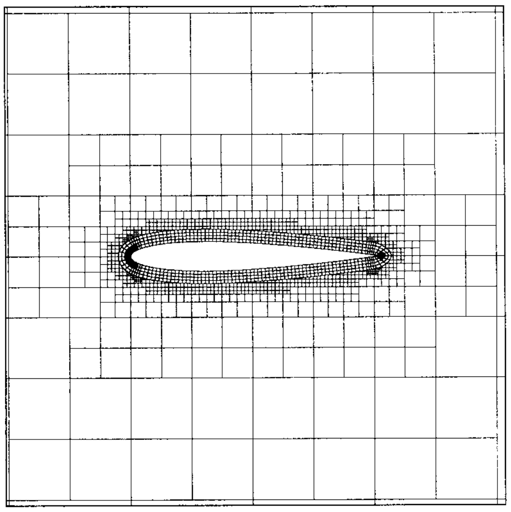


Figure 4. Initial grid for transonic flow around NACA0012 airfoil.

high gradient flow features (shocks, shear layer) were resolved very well. It is also observed that contour lines pass very smoothly from the Quad grids to the Cartesian cells even in the presence of highly irregular cut cells.

Three multi-grid cycling strategies (V, W and saw-tooth) were compared in this case. The parameter ϵ' was set at 0.025 for the modified Venkatakrishnan limiter. Convergence histories in terms of the number of multi-grid cycles with and without multi-grid on the level 2 adaptive grid are shown in Figure 7. Note that the W-cycle took the least number of multi-grid cycles to reach the steady state, followed by the V-cycle. The saw-tooth cycle failed to reach a steady state in this case. Because the W-cycle took more CPU time than the V-cycle, it is not necessarily true that the W-cycle is more efficient than the V-cycle. As a matter of fact, the contrary is true. The convergence histories in terms of CPU seconds on an SGI R4400 machine are presented in Figure 8. The V-cycle is clearly the most CPU-efficient. As a result, all the rest of the cases were simulated with the V-cycle.

6.2. Case 2. Multi-element airfoil case

This configuration was produced by McDonnell Douglas Aerospace (MDA) and has been investigated experimentally as part of a cooperative effort between MDA and NASA Langley [28]. Extensive experimental data is available for validating CFD codes. The test flow conditions are: $M_\infty = 0.2$, $\alpha = 16°$ and $Re = 9 \times 10^6$ based on the chord length. The flow was simulated as turbulent with the k–ϵ model and wall functions. The initial grid for this case is shown in Figure 9. The grid has a total of 9410 cells. The coarse grids used in the multi-grid algorithm are shown in Figure 10. Two levels of grid adaptation were carried out beyond the base mesh. The final grid has 31 730 cells and is displayed in Figure 11. The computed Mach

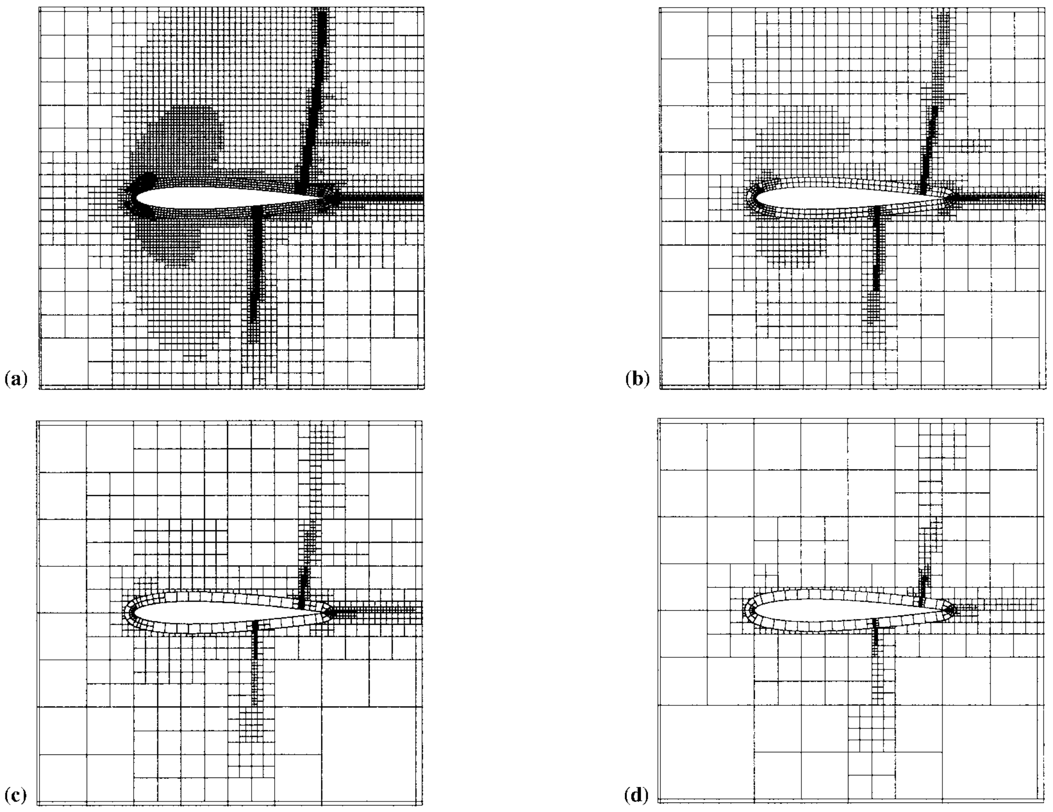


Figure 5. Final grid for NACA0012 airfoil. (a) Base grid (multi-grid level 4); (b) multi-grid level 3; (c) multi-grid level 2; (d) multi-grid level 1.

contours on the final adapted grid are shown in Figure 12, and the Cp profile is compared with experiment data in Figure 13. Note that after the first grid adaptation step, the computed Cp profile is nearly identical to the profile after the second grid adaptation step, indicating a clear trend toward a grid-independent solution. The converged Cp profile also showed an excellent agreement with the experimental data. In an earlier simulation assuming inviscid flow, the computed Cp profile shown in Figure 14 displays a visible difference between the simulated and experimental data, demonstrating the importance of capturing flow turbulence in high-Reynolds number flows. The convergence histories on all three levels of adaptive grids are shown in Figure 15. Note the convergence rate was quite good.

6.3. Case 3. Viscous flow over a backward-facing step

The case of backward-facing steps flow is widely used for benchmark validation of CFD method because detailed experimental data is available for this case [29]. The flow conditions

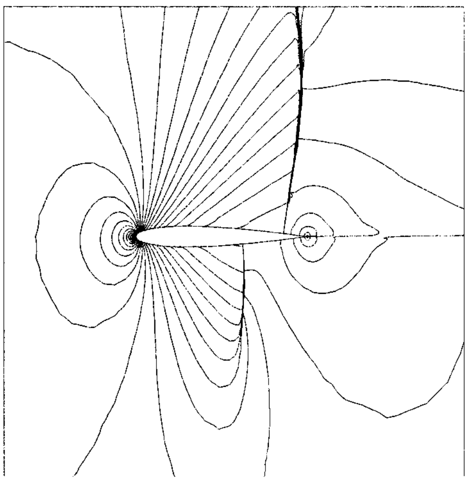


Figure 6. Mach number contours for transonic flow around NACA0012 airfoil.

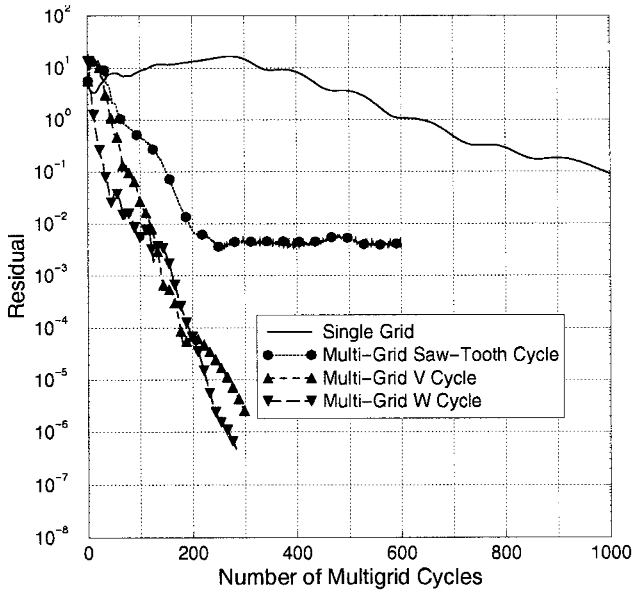


Figure 7. Convergence histories with and without multi-grid in terms of multi-grid cycles.

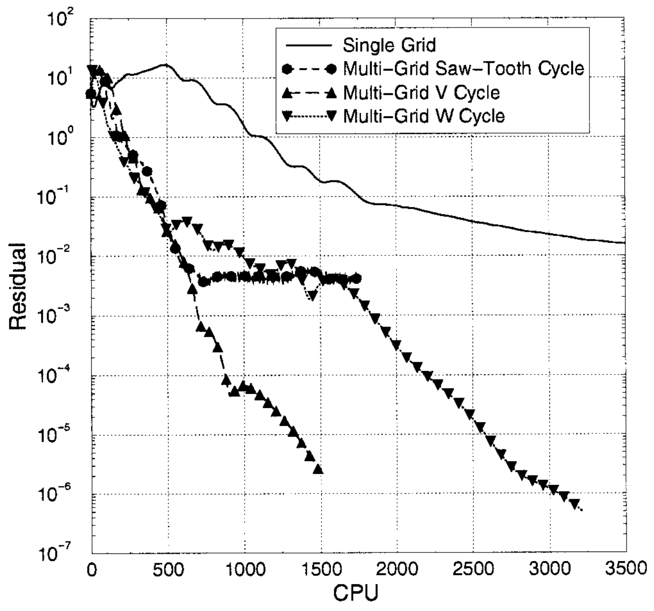


Figure 8. Convergence histories with and without multi-grid in terms of CPU seconds.

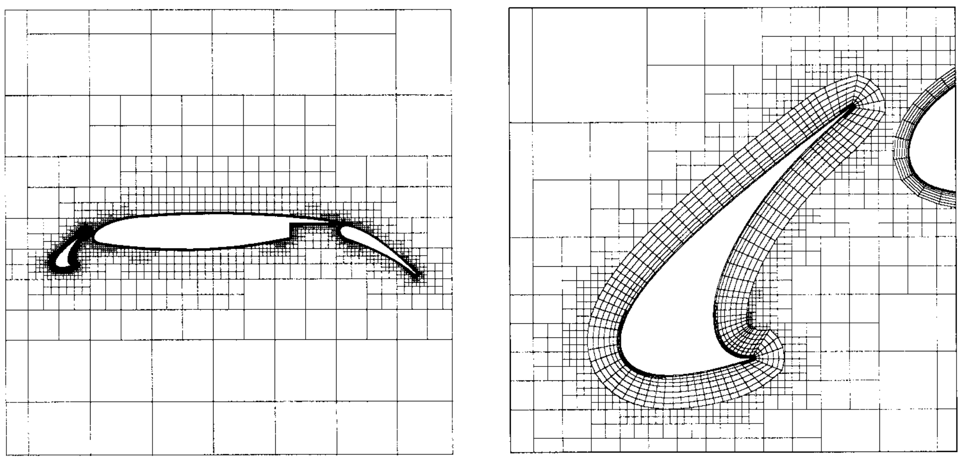


Figure 9. Initial computational grid for subsonic flow around a multi-element airfoil.

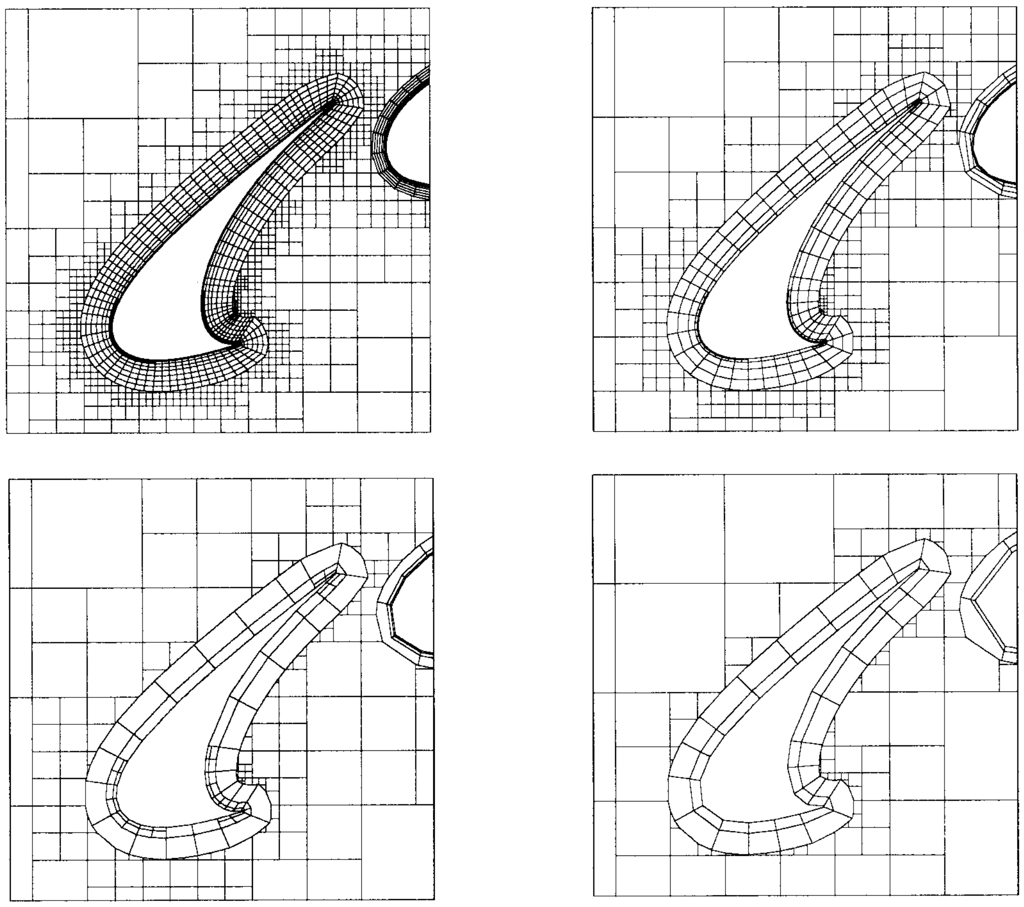


Figure 10. Different levels of coarse grids for the initial computational grid.

used are $M_\infty = 0.2$ and $Re = 389$. The initial grid has 2070 cells and is shown in Figure 16(a). Five levels of grid adaptation were then performed. The final grid has 24654 cells and is presented in Figure 16(b). At each grid, four multi-grid levels were used in the multi-grid solution procedure. Convergence history in terms of multi-grid V-cycles is displayed in Figure 17. It is a pleasant surprise that a near constant convergence rate was obtained on all levels of grids, which is achievable with multi-grid but rarely realized in practical flow simulations. Velocity profiles at $x/s = 4.18$ (s being the height of the step) from the present simulation are compared with experimental data in Figure 18. It is observed that a grid-independent velocity profile was obtained after three levels of grid adaptation. The convergence of the main

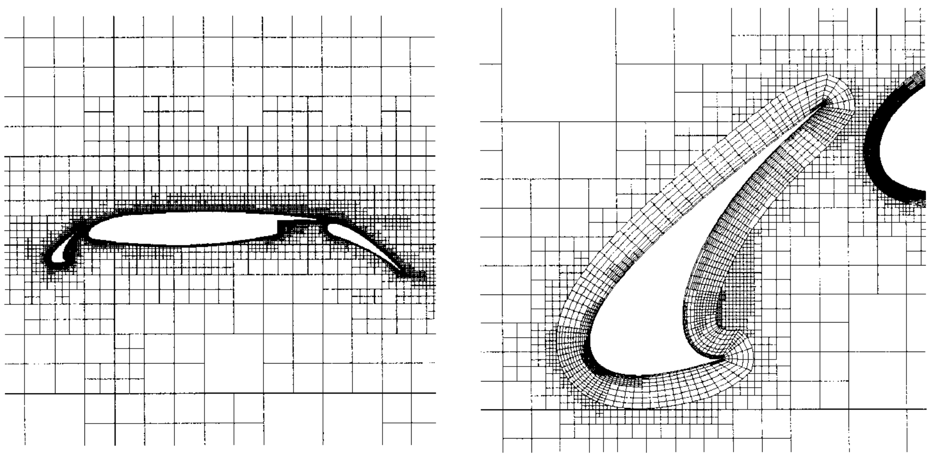


Figure 11. Computational grid for multi-element airfoil after two levels of grid adaptation.

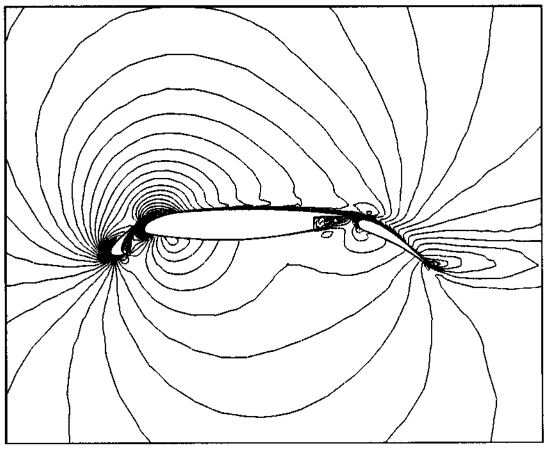


Figure 12. Computed Mach contours on the final grid for multi-element airfoil case.

reattachment point after the step is illustrated in Figure 19, which plots the skin friction coefficients at the lower channel wall. The trend of grid convergence is clearly conveyed by the figure. It is obvious that C_f is a bit more difficult to converge than the velocity profile. In this case, four levels of grid adaptation were required. The converged reattachment point is located

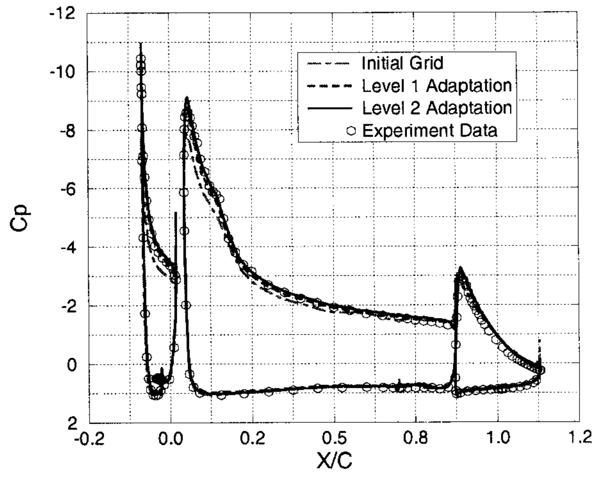


Figure 13. Comparison of Cp distributions on airfoil surfaces between computations and experiment assuming turbulent flow.

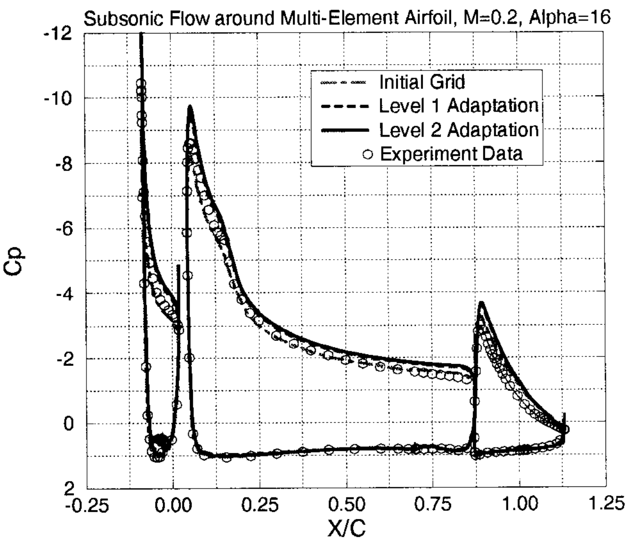


Figure 14. Comparison of Cp distributions on airfoil surfaces between computations and experiment assuming inviscid flow.

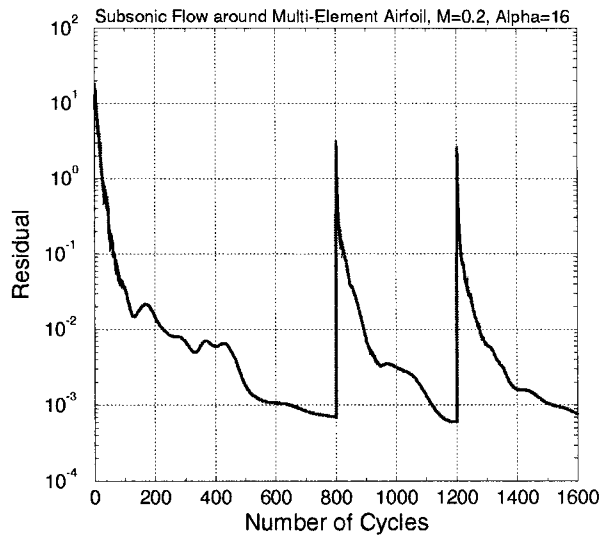
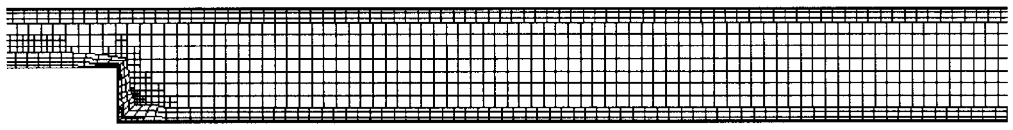
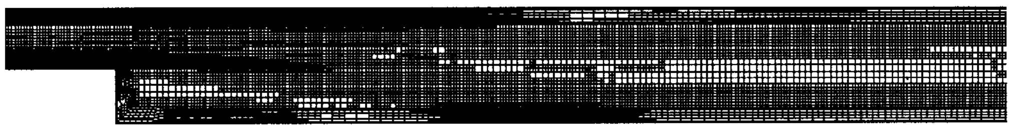


Figure 15. Convergence histories on different levels of grids.



(a) Initial Grid



(b) Level 5 Grid

Figure 16. Adaptive computational grids for backward-facing step problem.

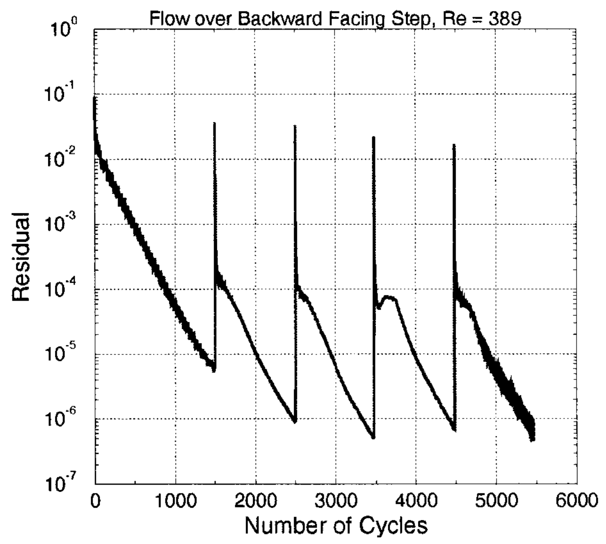


Figure 17. Convergence history on different levels of grids for backward-facing step problem.

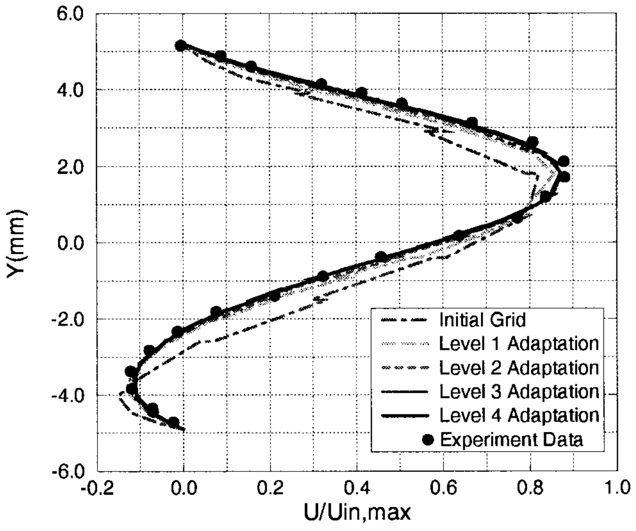


Figure 18. Comparisons of velocity profile between experiment and computations at $x/s = 4.18$.

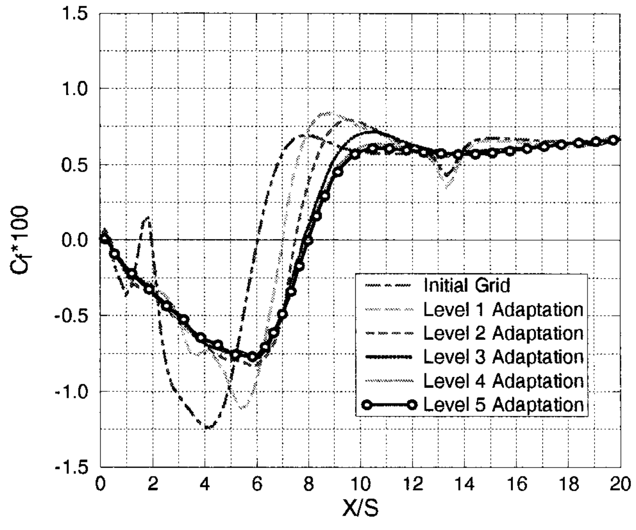


Figure 19. Skin friction distributions at lower channel wall.

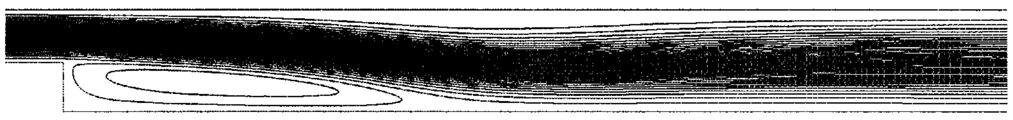


Figure 20. Computed streamlines on the final grid.

at $x/s = 8.0$, which is in good agreement with experimentally measured position of $x/s = 8.25$. The streamlines computed with the final grid are shown in Figure 20, which are very smooth.

7. CONCLUSIONS

A Quadtree-based adaptive Cartesian/Quad grid generator, grid adaptor, and multi-grid flow solver have been developed and successfully demonstrated in this study. The hybrid Cartesian/Quad grid generation procedure was shown to be robust and general. Complex geometries were tackled successfully. In addition, the Quadtree data structures enable nested coarse grids to be easily generated for an efficient multi-grid solution procedure. Different cycling strategies for multi-grid (i.e., saw-tooth, V and W) were tested on a variety of flow problems. The V-cycle was identified to be the most CPU efficient. Multi-grid was shown to speed up convergence to steady state by at least an order of magnitude in CPU. The grid adaptation criteria were demonstrated to perform well. Grid-independent inviscid and viscous solutions have been achieved through *automatic* grid adaptation.

ACKNOWLEDGMENTS

The author gratefully acknowledges the financial support by the NSF SBIR Phase I award No. DMI-9460372. He thanks Dr Marsha Berger of the Courant Institute for her valuable insights. Thanks are also due to Drs A.J. Przekwas and A.K. Singhal for their support.

REFERENCES

1. Barth TJ, Frederickson PO. Higher-order solution of the Euler equations on unstructured grids using quadratic reconstruction. AIAA Paper No. 90-0013, 1990.
2. Batina JT. Unsteady Euler algorithm with unstructured dynamic mesh for complex aircraft aerodynamic analysis. *AIAA Journal* 1991; **29**: 327–333.
3. Venkatakrishnan V. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics* 1995; **118**: 120–130.
4. Anderson WK, Bonhaus DL. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers and Fluids* 1994; **23**: 1–21.
5. Connell SD, Holmes DG. A 3D unstructured adaptive multigrid scheme for the Euler equations. AIAA Paper No. 93-3339-CP, 1993.
6. Schneiders R. Automatic generation of hexahedral finite element meshes. In *Proceedings of 4th International Meshing Roundtable '95*. Albuquerque, NM, October, 1995; 103–114.
7. Nakahashi K. Adaptive-prismatic-grid method for external viscous flow computations. In *Proceedings of 11th AIAA Computational Fluid Dynamics Conference*. Orlando, FL, AIAA-93-3314-CP, July, 1993; 195–203.
8. Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for complex geometries. AIAA Paper No. 95-0210, 1995.
9. Coirier WJ, Jorgenson PCE. A mixed volume grid approach for the Euler and Navier–Stokes equations. AIAA Paper No. 96-0762, 1996.
10. Lohner R, Parikh P. Generation of three-dimensional unstructured grids by advancing front method. *International Journal for Numerical Method in Fluids* 1988; **8**: 1135–1144.
11. Baker TJ. Three-dimensional mesh generation by triangulation of arbitrary point sets. AIAA Paper No. 87-1124, 1987.
12. Berger MJ, LeVeque RJ. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA Paper No. 89-1930-CP, 1989.
13. DeZeeuw D, Powell K. An adaptively refined Cartesian mesh solver for the Euler equations. AIAA Paper No. 91-1542-CP, 1991.
14. Coirier WJ, Powell KG. Solution-adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA Journal* 1996; **34**(5): 938–945.
15. Bayyuk AA, Powell KG, van Leer B. A simulation technique for 2D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrarily geometry. AIAA Paper No. 93-3391-CP, 1993.
16. Melton JE, Enomoto FY, Berger MJ. 3D automatic Cartesian grid generation for Euler flows. AIAA Paper No. 93-3386-CP, 1993.
17. Aftosmis MJ, Berger MJ, Melton JE. Robust and efficient Cartesian mesh generation for component-based geometry. AIAA Paper No. 97-0916, 1997.
18. Karman SL. SPLITFLOW: a 3D unstructured Cartesian/prismatic grid CFD code for complete geometries. AIAA Paper No. 95-0343, 1995.
19. Wang ZJ. Proof of concept—an automatic CFD computing environment with a Cartesian/prism grid generator, grid adaptor and flow solver. In *Proceedings of 4th International Meshing Roundtable*. Albuquerque, NM, October, 1995; 87–99.
20. Mavriplis DJ, Jameson A. Multigrid solution of the Navier–Stokes equations on triangular meshes. *AIAA Journal* 1990; **28**(8): 1415–1425.
21. Venkatakrishnan V, Mavriplis DJ. Agglomeration multigrid for the three-dimensional Euler equations. *AIAA Journal* 1995; **33**(4): 633–640.
22. Pirzadeh S. Viscous unstructured three-dimensional grids by the advancing layers methods. AIAA Paper No. 94-0417, 1994.
23. Aftosmis M, Gaitonde D, Tavares TS. On the accuracy, stability and monotonicity of various reconstruction algorithms for unstructured meshes. AIAA Paper No. 94-0415, 1994.
24. Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* 1983; **43**: 357.

25. Launder BE, Spalding DB. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering* 1974; **3**: 269–289.
26. Hemker PW, Koren B. Defect correction and non-linear multi-grid for the steady Euler equations. Lecture Notes, 1988-05 von Karman Institute for Fluid Dynamics, 1988.
27. van Leer B, Tai CH, Powell KG. Design of optimally smoothing multi-stage schemes or the Euler equations. AIAA Paper No. 89-1993-CP, 1989.
28. Chin VD, Perters DW, Spaid FW, McGhee RJ. Flow field measurements about a multi-element airfoil at high Reynolds number. AIAA Paper No. 93-3137, 1993.
29. Armaly BF, Pereira JCF, Schonung B. Experimental theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics* 1983; **127**: 473–496.