AIAA JOURNAL
Vol. 34, No. 2, February 1996

Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Navier–Stokes Equations

Mark E. Braaten* and Stuart D. Connell†
General Electric Research and Development Center, Schenectady, New York 12301

A solution adaptive multigrid scheme for solving the three-dimensional Navier–Stokes equations on unstructured meshes is presented. The algorithm solves the equations on a fully unstructured mesh of tetrahedral elements, using a multigrid time-marching scheme. The initial unstructured mesh is successively refined based on gradients in the flow solution, with the multigrid levels being determined by the refinement procedure. Important issues related to solving viscous flows on unstructured meshes are discussed. These include calculation of the viscous stress terms in a manner that prevents wiggles, calculation of the time steps to include both inviscid and viscous effects, numerical smoothing considerations, minimization of computer storage requirements, and implementation of the k–ε turbulence model. Solutions are presented for several examples of industrial importance that illustrate the potential of the method, including transonic flow about an aircraft engine nacelle, and in both rotating and nonrotating turbomachinery passages.

I. Introduction

OVER the past decade, computational fluid dynamics has made a significant impact on the design of turbomachinery, combustors, and other flow devices (see, for example, Refs. 1 and 2). As successes have mounted, the demands on computational fluid dynamics algorithms have continued to grow. The geometry of turbomachinery blade rows has become complicated by high blade turning, significant blade twist, the presence of part-span shrouds or splitters, and the presence of tip clearance. Such problems cannot be easily computed with single block structured grid codes. Multiblock structured grid codes have been developed, but generation of multiblock grids remains very difficult and time consuming.

Unstructured grid algorithms have demonstrated the potential for handling arbitrary geometries while readily allowing grid adaptation to efficiently resolve small scale flow features.[3–8] Recent developments in unstructured mesh generation allow an initial unstructured mesh to be automatically generated from a solid model surface representation of the geometry.[4,9,10] As the solution progresses, the mesh is successively refined to accurately resolve flow features, such as trailing edges and shocks, with a minimum number of grid points.[5,7,9,11] The reduction in the time required to mesh the geometry, and the elimination of the need to decide where to place the mesh points a priori, makes the code more user friendly to the designer and can lead to a significant reduction in design cycle time.

The goal of this work is to develop a flow solver based on an unstructured tetrahedral mesh capable of fast and accurate analysis of transonic flows in complicated three-dimensional geometries. The viscous algorithm described here is an extension of an earlier inviscid algorithm described in Ref. 11. The inviscid algorithm is briefly reviewed before describing the extension of the method to viscous flows. Some example calculations are then shown.

II. Governing Equations

The three-dimensional Navier–Stokes equations may be written in conservation law form as

$$\frac{\partial \boldsymbol{U}}{\partial t}+\frac{\partial \boldsymbol{F}}{\partial x}+\frac{\partial \boldsymbol{G}}{\partial y}+\frac{\partial \boldsymbol{H}}{\partial z}=\boldsymbol{S} \tag{1}$$

where $\boldsymbol{U}$ represents the state vector, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ represent the flux vectors in the three coordinate directions, and $\boldsymbol{S}$ represents the source terms. To facilitate the solution of flows in rotating turbomachinery blade rows, the equations are cast in terms of the velocities relative to a reference frame rotating with an angular velocity Ω about the z axis. The flux vectors can be decomposed into inviscid flux vectors $\boldsymbol{F}_i$, $\boldsymbol{G}_i$, and $\boldsymbol{H}_i$, and viscous flux vectors $\boldsymbol{F}_v$, $\boldsymbol{G}_v$, and $\boldsymbol{H}_v$. These vectors are then defined as

$$\boldsymbol{U}=\begin{pmatrix}\varrho\\ \varrho w_x\\ \varrho w_y\\ \varrho w_z\\ \varrho E\\ \varrho k\\ \varrho \varepsilon\end{pmatrix},\qquad \boldsymbol{F}_i=\begin{pmatrix}\varrho w_x\\ \varrho w_x^2+p\\ \varrho w_x w_y\\ \varrho w_x w_z\\ w_x(\varrho E+p)\\ \varrho w_x k\\ \varrho w_x \varepsilon\end{pmatrix} \tag{2}$$

$$\boldsymbol{G}_i=\begin{pmatrix}\varrho w_y\\ \varrho w_x w_y\\ \varrho w_y^2+p\\ \varrho w_y w_z\\ w_y(\varrho E+p)\\ \varrho w_y k\\ \varrho w_y \varepsilon\end{pmatrix},\qquad \boldsymbol{H}_i=\begin{pmatrix}\varrho w_z\\ \varrho w_x w_z\\ \varrho w_y w_z\\ \varrho w_z^2+p\\ w_z(\varrho E+p)\\ \varrho w_z k\\ \varrho w_z \varepsilon\end{pmatrix} \tag{3}$$

$$\boldsymbol{F}_v=\begin{pmatrix}0\\ -\tau_{xx}\\ -\tau_{xy}\\ -\tau_{xz}\\ -w_x\tau_{xx}-w_y\tau_{xy}-w_z\tau_{xz}+q_x\\ -\tau_{xx}^k\\ -\tau_{xx}^\varepsilon\end{pmatrix} \tag{4}$$

$$\boldsymbol{G}_v=\begin{pmatrix}0\\ -\tau_{xy}\\ -\tau_{yy}\\ -\tau_{yz}\\ -w_x\tau_{xy}-w_y\tau_{yy}-w_z\tau_{yz}+q_y\\ -\tau_{yy}^k\\ -\tau_{yy}^\varepsilon\end{pmatrix} \tag{5}$$

Received July 28, 1995; revision received Aug. 1, 1995; accepted for publication Aug. 1, 1995.

*Mechanical Engineer, Fluid Mechanics Program, P.O. Box 8.
†Mechanical Engineer, Fluid Mechanics Program, P.O. Box 8. Member AIAA.

$$\boldsymbol{H}_v = \begin{pmatrix} 0 \\ -\tau_{xz} \\ -\tau_{yz} \\ -\tau_{zz} \\ -w_x\tau_{xz} - w_y\tau_{yz} - w_z\tau_{zz} + q_z \\ -\tau^k_{zz} \\ -\tau^\varepsilon_{zz} \end{pmatrix} \tag{6}$$

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ \varrho\left(x\Omega^2 - 2\Omega w_y\right) \\ \varrho\left(y\Omega^2 - 2\Omega w_x\right) \\ 0 \\ 0 \\ \mu_t G - \varrho\varepsilon \\ c_1\mu_t G\varepsilon/k - c_2\varrho\varepsilon^2/k \end{pmatrix} \tag{7}$$

where w_x, w_y, and w_z are the relative velocities in the x, y, and z coordinate directions. Also, ϱ is the density, p the pressure, Ω the rotational speed, k the turbulent kinetic energy, ε the turbulent energy dissipation, μ_t the turbulent viscosity, τ_{ij} the viscous shear stresses, and q_i the viscous heat fluxes. The relative velocities w_x, w_y, and w_z are related to the absolute velocities v_x, v_y, and v_z by

$$w_x = v_x + \Omega y \tag{8}$$

$$w_y = v_y - \Omega x \tag{9}$$

$$w_z = v_z \tag{10}$$

The turbulent diffusion fluxes are given by relations of the form[12]

$$\tau^k_{xx} = \frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x}, \qquad \tau^\varepsilon_{xx} = \frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x}, \ldots \tag{11}$$

The quantity ϱE is given by

$$\varrho E = [p/(\gamma - 1)] + \tfrac{1}{2}\varrho\left[w_x^2 + w_y^2 + w_z^2 - (\omega r)^2\right] \tag{12}$$

III. Summary of Inviscid Algorithm

The viscous flow algorithm described here is an extension of the inviscid algorithm described by Connell and Holmes.[11] The inviscid algorithm will be summarized briefly here; the reader is referred to the original reference for the details.

The inviscid algorithm solves the Euler equations on a three-dimensional fully unstructured mesh of tetrahedral elements, using an adaptive multigrid time-marching scheme. The initial unstructured mesh is obtained by destructuring of a structured mesh or by use of the advancing front method.[4] The solution procedure begins with this initial unstructured mesh. The evolving solution is examined, and the mesh is enriched in regions of high flow gradients by H refinement. New boundary nodes are placed at their proper location on the boundary via a node snapping procedure that refers back to a parametric representation of the true boundary shape. This adaptation procedure is used to generate the multigrid levels used by the solver. This multigrid scheme has several advantages over other multigrid schemes based on adaptive remeshing. The multigrid levels are formed naturally by the refinement procedure, the construction of the multigrid transfer operators is greatly simplified, and most importantly, the H-refinement procedure is orders of magnitude faster and more robust than remeshing.

A hierarchical data structure is adopted to meet the different needs of the refiner, flow solver, and postprocessor. With this hierarchical data structure, the connectivity between any two grid entities (cells, faces, edges, and nodes) can be efficiently derived by a single pass through the data structures. After refinement, the data structure contains all of the original grid information, plus all of the new entities. Pointers are used to identify the data for each grid level; no duplication of information between the different grid levels is needed.

The multigrid flow solver uses a vertex-based finite volume scheme with nonoverlapping control volumes, and a Runge–Kutta time marching procedure. The convective terms are computed using central differences, along with an adaptive blend of second- and fourth-order smoothing based on that of Jameson.[13] Flux balances, smoothing residuals, and time steps are computed using efficient edge-based loops. Local time steps are used to accelerate convergence to the steady state. Implicit residual averaging is not currently employed, as we have not been successful with it on unstructured grids to date. A simple V cycle is used for the multigrid. The storage requirements for all of the node-based, edge-based, and boundary data for the inviscid algorithm works out to be equivalent to about 120 words/node.

IV. Viscous Algorithm Development

A. Calculation of Viscous Stresses

Evaluation of the shear stress terms in the Navier–Stokes equations requires the calculation of the corresponding velocity derivatives. Various alternative means for computing these derivatives exist. Important considerations in choosing from these alternatives are the compactness of the resulting difference stencil and the desire that the resulting viscous terms sense oscillations in the solution and help to smooth them. In Ref. 12, it was noted that averaging the cell derivatives on either side of an edge to get the edge derivatives did not allow the derivatives computed on quadrilateral cells to smooth wiggles. In this work, we have chosen to directly compute the derivative components at the centers of the edges, which correspond to the centers of the control volume faces. The use of edge derivatives provides a compact derivative stencil and allows the viscous terms to smooth wiggles on tetrahedral meshes. Computing nodal derivatives and averaging them to the edge centers was rejected because such a procedure is incapable of removing these wiggles.

The derivative at the center of the edge e is computed by applying Green's theorem over the volume composed of all of the tetrahedra sharing the common edge e, that is,

$$\left.\frac{\partial F}{\partial x}\right|_e = \frac{1}{V}\iint_A F\,\mathrm{d}A_x \tag{13}$$

The situation is shown in Fig. 1. The derivative calculations for the derivatives of a function F in all three coordinate directions are accomplished by a single loop over cells. The three flux components on each cell face and the cell volume are accumulated to the edges with only one node on that face. The derivatives are then obtained by an edge loop that divides the accumulated flux sums by the accumulated edge volumes.

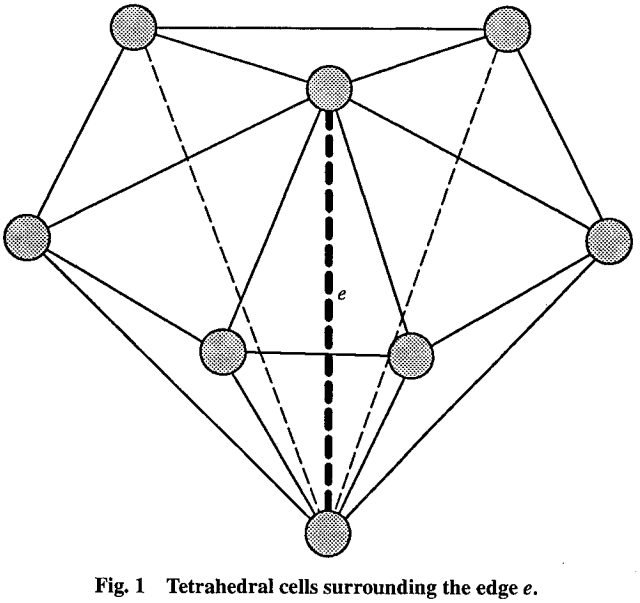


Fig. 1 Tetrahedral cells surrounding the edge e.

Since an edge array typically requires about seven times as much storage as a nodal array, the storage required for computing edge-based derivatives can become substantial. To minimize this storage, we allocate only nine edge arrays for viscous derivatives. These nine arrays are sufficient to store all of the velocity derivatives in each of the three coordinate directions. After the velocity derivatives are calculated, all other terms that require them are then computed. This allows the nine arrays to then be reused to store the edge derivatives of the internal energy and the turbulence quantities for subsequent use. Nodal derivatives can be computed in a similar manner to the edge derivatives, i.e., by summing fluxes over faces of cells surrounding the node and dividing by the sum of the surrounding cell volumes.

B. Viscous Time Step

The solution procedure adopted is a multistep Runge–Kutta scheme with local time steps to accelerate convergence, but without residual averaging. A three-step scheme is used for viscous flows, with the viscous and smoothing terms computed only on the first partial step. For viscous flows, we have found that it is important to properly account for both inviscid and viscous contributions to the local time steps to achieve rapid and robust convergence.

The approach adopted computes the local time step on the basis of both inviscid and viscous considerations. It allows the use of Courant–Friedrichs–Lewy (CFL) multipliers comparable to those used for inviscid flows (0.5–0.75) and leads to stable and rapid convergence. The inviscid time step Δt_{inv} is computed from the formula

$$\Delta t_{\text{inv}} = \frac{V(\text{CFL}M)(\text{CFL})_0}{\frac{1}{2}\sum_f(\lambda_i + \lambda_j + \lambda_k)} \tag{14}$$

where V is the cell volume, CFLM the CFL multiplier, CFL_0 the CFL number for the particular multistage scheme, and λ_i, λ_j, λ_k the eigenvalues that are computed from

$$\lambda_i + \lambda_j + \lambda_k = \sum_f (cA + A_x w_x + A_y w_y + A_z w_z) \tag{15}$$

where c is the sound speed, A the face area, and w the flow velocity.

The viscous time step t_{vis} at each cell is computed from

$$\Delta t_{\text{vis}} = \beta(h^2/6\nu) \tag{16}$$

where β is a multiplying factor less than one, ν the kinematic viscosity, and h an estimate of the minimum cell height that is computed from

$$h = 3(V/b_{\max}) \tag{17}$$

where V is the cell volume and $b_{\max}$ the largest face area of the cell. The use of this estimate of the cell height was found to be far more effective than using the shortest edge length of the cell. A cell with nearly equal edge lengths can be squashed down to form a very flat cell, which is not detected by the latter practice. The local viscous time step at each node is taken as the minimum value from among all of the cells that have that node as a vertex. The local time step at each node is then computed as the minimum of t_{inv} and t_{vis}.

C. Numerical Smoothing for Viscous Flows

Viscous flow solutions place even greater demands on the numerical smoothing than inviscid flows. The numerical smoothing must be reduced in regions where the viscous stresses are important to avoid corrupting the results. The viscous solution algorithm must also cope with the very high aspect ratio cells that occur in viscous meshes in the boundary layer. The fourth-order smoothing fluxes cause problems near the viscous walls, since the large velocity gradients cause the fourth-order terms to become large and because it is difficult to compute these terms accurately in this region because of the need to resort to one-sided differencing at the wall.

Although a number of alternative means of reducing the smoothing in the boundary layer have been proposed to alleviate these problems, here we have only considered variations of the Jameson adaptive blend of second- and fourth-order smoothing, and simple first-order upwinding approaches. We plan to consider more sophisticated upwinding schemes at a later date.

A common practice for viscous flows is to devise some means of reducing the smoothing in the boundary layers. Often the smoothing fluxes are turned off entirely at both the wall and the near-wall edges. In addition, in structured grid algorithms, the smoothing on grid lines normal to the walls is often reduced by means of the ratio of the local Mach number to a freestream Mach number.[14] This practice is of questionable utility in some of the flows of interest here. An example is a highly turning turbine stator, where the passage Mach number can vary from as low as 0.05 at the inlet to as high as 1.3 at the exit. A practice that has been adopted on unstructured grids is to reduce the fourth-order dissipation as a function of the local cell Reynolds number.[12] A drawback of this approach is that the smoothing in all directions is reduced, which can lead to oscillations in the streamwise direction.

Martinelli[15] proposed a directional smoothing algorithm that is claimed to stabilize the numerical scheme for grids with aspect ratios as high as 500. The directional scheme acts to amplify the smoothing on the short edges, relative to that obtained from a standard isotropic scheme.

Calculations made with various smoothing schemes led to several basic conclusions. Turning the smoothing off on certain edges near the wall proved to be essential. Solutions obtained without turning off the smoothing at any near-wall edges were very diffusive. Potentially desirable edges on which to reduce the smoothing are the near-wall edges with one wall node and one interior node, and the edges with one near-wall node and one interior node. Removing the smoothing on these latter edges as well as the near-wall edges proved to be beneficial, particularly in terms of improving the turbulence profiles in the boundary layer. The boundary-layer profiles of the turbulence variables proved to be far more sensitive to the details of the smoothing scheme than did the velocity and pressure profiles. Increasing the amount of second-order smoothing, or resorting to upwinding, reduced oscillations in the velocity and pressure profiles in regions of strong gradients, but also led to unphysical behavior in the profiles of turbulent kinetic energy and dissipation.

Figure 2 shows the turbulent kinetic energy profiles in the near-wall region, for a developing duct flow, calculated with four different smoothing schemes. The first scheme is a simple vanilla implementation of an adaptive blend of second- and fourth-order differences, with an unweighted pseudo-Laplacian used to compute the smoothing fluxes. The second scheme attempts to improve the accuracy of the smoothing fluxes through the use of improved estimates of the third differences at the center of the edges, as done in Ref. 12. The third scheme uses Martinelli's[15] directional smoothing in conjuction with this improved scheme. The fourth scheme uses the standard adaptive second- and fourth-order smoothing on edges away from the walls, and simple first-order upwinding on edges near the walls.

The results for the first two schemes show the proper physical behavior, with the peak value of the turbulence energy occuring at the wall and decaying into the freestream. The results for the latter two schemes show large unphysical peaks in the profile away from the wall. This same unphysical behavior occurs with the simple scheme when the coefficient of the second-order smoothing term is excessively large. This observed behavior shows that excessive second-order smoothing can lead to unphysical behavior of the turbulence profiles. Overall, the simple nondirectional scheme gives the best performance in terms of high accuracy and low computer time.

D. Memory Requirements

Unstructured grid algorithms typically require substantially more storage per node than their structured grid counterparts, because of the need to store all sorts of grid connectivity information. Numerous steps have been taken to reduce the memory requirements of this algorithm. The full hierarchical data structure required by the refinement algorithm contains much more data than the solver requires. A separate preprocessing program was developed to input the file containing the full hierarchical data structure and output a streamlined file containing just the data required by the solver. The

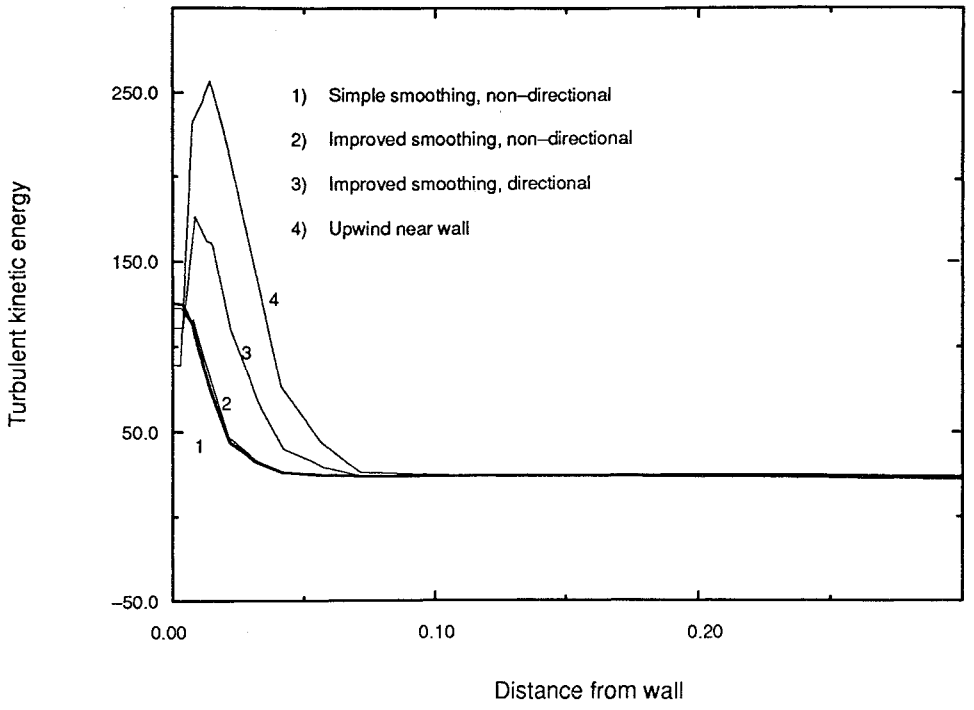


Fig. 2 Effect of smoothing on turbulent kinetic energy profile.

use of edge-based loops rather than face-based loops to compute the flux balances reduces the memory requirements. Another useful tactic is the reuse of the edge arrays required by the derivative calculations as described earlier.

The addition of the viscous terms and the turbulence variables does cause a significant increase in the memory requirements of the algorithm. Additional nodal arrays are required for the viscous accumulators (since the viscous terms are computed only at the first partial step) and for the turbulence quantities. Additional edge arrays are required to store the edge derivatives and the edge volumes. Additional cell arrays are required to store cell-to-node pointers, and cell-to-edge pointers required by the derivative calculations and the viscous terms.

The additional memory requirements can be expressed in terms of equivalent nodal arrays. Each edge array is equivalent to about seven nodal arrays, and each cell array is equivalent to about six nodal arrays. With this conversion, the additional memory requirement is about 180 additional words/node, with most of this stemming directly from the derivative calculations. Added to the 120 words/node storage requirement of the inviscid algorithm, this brings the total storage requirement for the viscous algorithm to about 300 words/node. This requirement compares favorably with that of other three-dimensional unstructured schemes: a value of 250 words/node reported in Ref. 5 for a nonmultigrid scheme, and values of 280–350 words/node reported in Ref. 8 for another multigrid scheme. Currently, the refinement code has an even larger memory requirement of about 450 words/node, which limits how large a mesh can be run on our current workstations. With 350 Mbytes of available memory (including virtual memory), the maximum grid size is limited to about 150,000 nodes.

V. Turbulence Model

Turbulent flows are modeled here using the standard k–ε turbulence model with wall functions. This model was chosen in preference to a mixing length model because the unstructured nature of the grid in the boundary layer makes it difficult to apply the mixing length model, as discussed in Ref. 12. The solution of the two additional conservation equations required by the k–ε model was performed using the same Runge–Kutta procedure as for the other conservation equations. The convective fluxes in the k and ε equations were evaluated using first-order upwinding, to prevent the possibility of overshoots that could lead to negative values or unrealistic levels of turbulent viscosity, as described in Ref. 7.

Implementation of the wall functions on an unstructured boundary-layer grid requires the building of a list of the wall nodes, a list of the near-wall nodes, a list of the near-wall edges (i.e., those edges with one wall node and one interior node), and a list of pointers from each wall node to the closest near-wall point. It is important to note that for unstructured grids, there is not in general a one-to-one correspondence between the complete list of near-wall nodes and the near-wall nodes that are pointed to by the wall nodes. The list of wall nodes, along with the list of pointers from the wall nodes to the closest near-wall nodes, is used to enforce the zero gradient boundary conditions for k and ε at the wall points. The complete list of near-wall nodes is used to enforce the usual ε boundary condition, namely, a prescribed value of ε at all points one-off the wall. The list of near-wall edges is used to apply the wall functions to calculate the shear stresses from the universal velocity profile and apply them to the control volume faces normal to these edges. The distance between each near-wall node and the wall is determined by first finding the closest wall point and then computing the normal distance to this point using the wall normals.

An additional refinement criterion based on the local value of the wall variable y^+ was added to the grid refinement procedure to accommodate the turbulent wall functions. Refinement was triggered if $y^+ > 500$, and derefinement was triggered for $y^+ < 10$, to ensure the validity of the wall functions at the near-wall nodes. This refinement criterion also proved useful in preventing derefinement on the endwalls near the inlet of highly turning turbine blade rows, where the velocity and pressure gradients are quite small.

The k–ε equations are notoriously difficult to solve by time-marching methods because the turbulent source terms are large and very strongly coupled. However, standard pressure correction codes routinely solve these equations on a structured grid using implicit solution techniques, such as relaxation. We have utilized the idea of source term linearization common in these implicit relaxation algorithms to improve the time-marching solution of the k–ε equations.

Consider a discretized convection/diffusion equation, written in the following form:

$$\left(\sum a_{nb} + a_{p_0} - S_p\right)\phi_p = \sum a_{nb}\phi_{nb} + S_u \qquad (18)$$

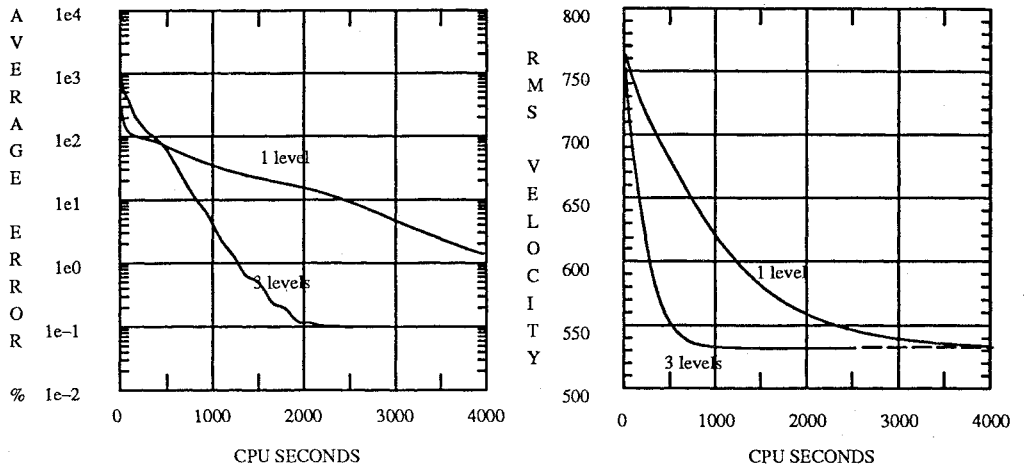


Fig. 3 Effect of multigrid on convergence for viscous problem.

where a_{nb} represents the contributions from the convection and diffusion terms, a_{p_0} the contribution from the unsteady term, S_p the linear contribution from the source term, and S_u the constant part of the source term.

Retaining the linearized influence of the source term in the time-marching procedure results in the following equation for the solution update:

$$\phi_p = \frac{\left[\sum a_{nb}(\phi_{nb} - \phi_p) + S_u\right]}{\left(a_{p_0} - S_p\right)} \tag{19}$$

Note that as the source term becomes large, Eq. (19) has the correct limiting behavior, giving $\phi_p = -S_u/S_p$, and it also reverts back to a standard time-marching scheme as the source term vanishes.

The addition of this source-term linearization to the time marching was found to significantly improve the solution of the k–ε equations. Convergence is faster and more stable, and the need to clip the turbulent viscosity during the evolution of the solution to prevent unphysical values from occurring is eliminated.

VI. Multigrid

The multigrid scheme utilizes a simple V-cycle and the pushdown procedure described in Ref. 11 to generate the multigrid levels. For inviscid flow over a cascade of biconvex airfoils, six levels of multigrid was found to reduce the CPU time by a factor of 13.7 over the same case without multigrid. Figure 3 shows the effect of multigrid for a typical laminar viscous calculation. With three levels of multigrid, the CPU time is reduced by almost a factor of four over the same case without multigrid. In viscous flows, it is difficult to utilize more than three or four multigrid levels with the current procedure, without either having initial grids that are so coarse that the geometry cannot be adequately represented, or having the number of grid points in the boundary layer grow so rapidly that the grid size becomes larger than can be handled with the available computing resources.

For turbulent flows, three levels of multigrid typically gives a twofold reduction in CPU time over the same case without multigrid. For many problems, the use of the multigrid for the k–ε equations was found to provoke numerical instability. The likely cause of this problem is that the turbulent source terms are not updated at each of the coarse grid levels, but are merely prolongated from the finest level. Turning off the multigrid for the k and ε equations, while retaining it for the remaining equations, was found to remedy this problem. Speedups obtained with this partial multigrid were found to be comparable to that obtained with the full multigrid, for cases where the full multigrid would work. Improving the effectiveness of the multigrid scheme for turbulent flows is a high priority for future work.

VII. Boundary Conditions

The boundary conditions adopted here are the standard ones typically used for turbomachinery calculations. At the inlet, total pressure, total temperature, and either flow angle or tangential velocity are prescribed. For rotating blade rows, the total conditions and flow angles can be specified in either absolute or relative terms. At the exit plane, a fixed static pressure distribution is prescribed. For rotating flows, a radial equilibrium pressure distribution can alternatively be enforced. This radial equilibrium boundary condition requires special interpolations to be performed to compute the circumferential averages on the unstructured grid on the exit plane. The nonslip condition is imposed for viscous wall boundaries, and the wall surfaces are assumed adiabatic. Wall functions are used for the turbulence variables, as described in Sec. V.

VIII. Viscous Mesh Generation

Three means of generating the initial unstructured grids used for the calculations described in this paper were used. These methods include: 1) the destructuring of structured hexahedral grids into tetrahedra, 2) the extrusion of two-dimensional unstructured triangular grids into three-dimensional prismatic grids, and their subsequent decomposition into tetrahedra, and 3) the generation of semistructured meshes of prisms and tetrahedra by the combined use of surface inflation and an advancing front mesh generator.

Destructuring of structured single-block hexahedral meshes is attractive for geometries that can be easily meshed with a structured grid generator. The power of the unstructured grid algorithm in adaptively refining the solution can then be enjoyed. Many geometries, however, cause difficulties for single-block structured meshes, including turbomachinery blades with high turning, and geometries such as transition ducts, where the cross sectional shape changes dramatically.[16]

Extrusion of two-dimensional unstructured triangular grids into three-dimensional prismatic meshes, which are then destructured into tetrahedra, works well for geometries that do not have large variations in shape in the extruded direction. This procedure is also a convenient means of generating grids to facilitate testing and comparison of the new three-dimensional code with our established two-dimensional code. The extrusion procedure proved useful for generating blade row meshes for blades with little spanwise variation, but high turning.

The generation of semistructured grids through a combination of surface inflation and the advancing front method is described fully in a companion paper.[16] The structured prisms generated by the surface inflation ensure a high-quality mesh in the boundary layer, and the advancing front procedure, in conjunction with an edge swapping procedure that improves the cell aspect ratios, results in a high-quality tetrahedral mesh in the interior. This approach to grid generation has the capability of handling very complicated geometries, and can be directly coupled to a CAD system, to automatically generate the starting mesh directly from the CAD surface representation.

IX. Results

Results for some realistic geometries of interest to aircraft engine gas turbines and power generation equipment are presented

in this section. Comparisons of the code results have been made with the results of an earlier structured grid code,[17,18] widely used for design within General Electric. This structured grid code has been applied to transonic fans, compressor blade rows, and turbine blade rows, and has been extensively validated against experimental data. The intent of this section is to demonstrate the basic capabilities of the new unstructured algorithm, not to exhaustively present detailed solutions for any of these geometries. Detailed code validations are currently underway for these and other similar geometries.

A. Case 1: Aircraft Engine Nacelle

The first test problem involves turbulent flow about an isolated aircraft engine nacelle. The angle of attack is 4 deg and the inlet Mach number is 0.82, corresponding to cruise conditions. This case was previously calculated for inviscid flow, and the results compared very well with test data. Detailed comparisons of the surface Mach numbers with experimental data are given in Ref. 11. The initial viscous grid for this case was generated using the surface inflation/advancing front technique described in Ref. 16. Figure 4 shows the initial mesh; the high quality of the grid at the leading edge of the nacelle and in the boundary layer along the nacelle surface is obvious. The solution was obtained on a twice-refined mesh with 131,123 nodes and 724,116 cells. Figure 5 shows the convergence path for the complete nacelle calculation. As can be seen, the calculation on the finest grid takes more than 75% of the total CPU time. The number of time steps required to reduce the initial error by the same amount remains roughly constant for each grid, indicating the usefulness of the multigrid scheme.

Figure 6 shows the refined mesh, and Fig. 7 shows the calculated Mach number contours on the symmetry plane. The stagnation region on the nose of the centerbody and the acceleration of the flow around the leading edge of the nacelle can be seen from the solution.

B. Case 2: Turbine Vane

The second problem presented here concerns the flow in a highly turning transonic turbine vane. The flow enters at a low Mach number of about 0.1, and exits with a peak Mach number of about 1.4. The initial mesh for this problem, illustrated in Fig. 8, contains 1550 grid points and was generated by the extrusion of a two-dimensional unstructured mesh, as described in Sec. VIII. The blade lies wholly within the passage, and periodic sufaces bound the solution domain.

Figure 9 shows the mesh on the hub surface, for the initial mesh and for the mesh after two levels of refinement. The mesh refinement on the viscous wall surfaces is seen to be nearly uniform. This is a problem with purely tetrahedral meshes, since tetrahedra cannot

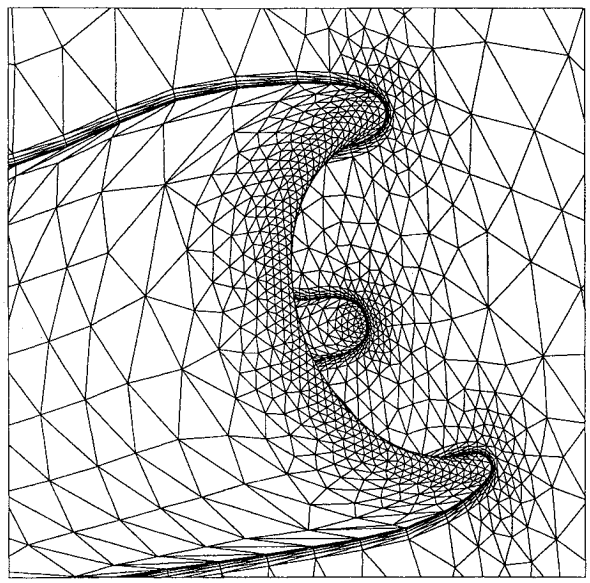


Fig. 4 Initial viscous mesh for nacelle case.

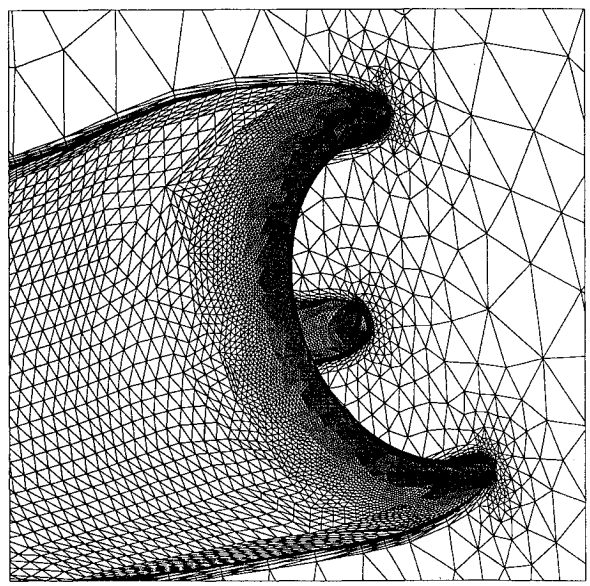


Fig. 6 Refined nacelle mesh.

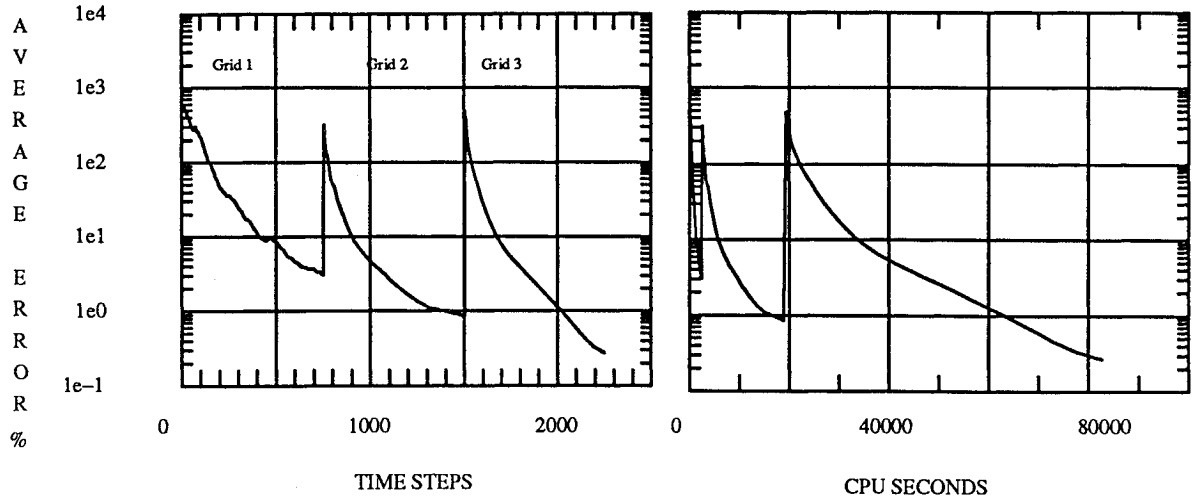


Fig. 5 Convergence path for nacelle calculation.

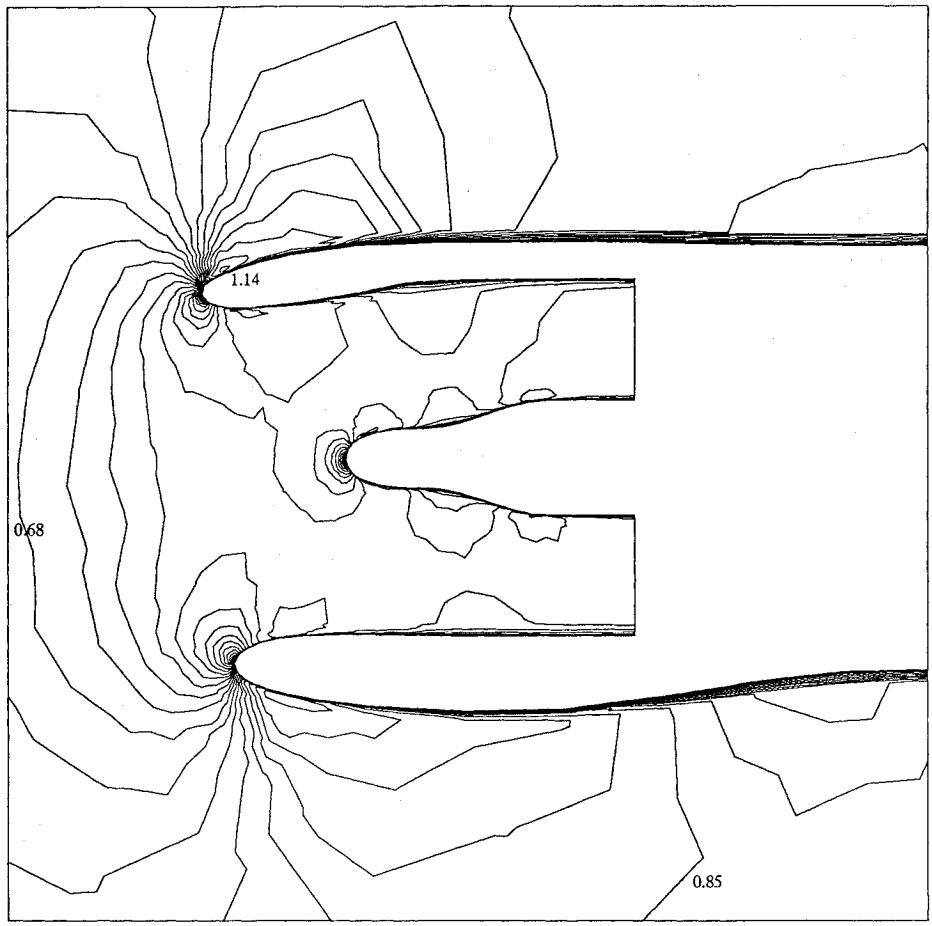


Fig. 7 Mach number contours for nacelle case.

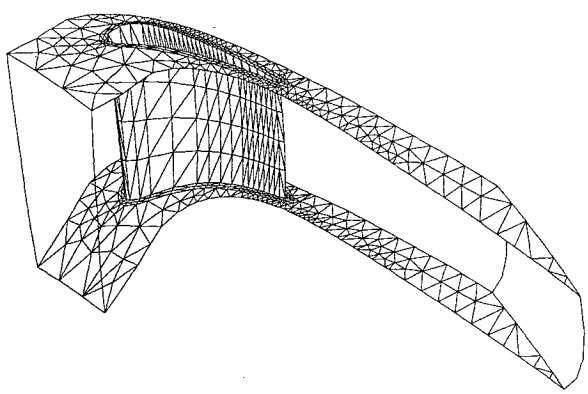


Fig. 8 Initial mesh for turbine vane.

be efficiently refined in a one-dimensional manner. The refinement triggered by the high normal flow gradients in the boundary layer causes an accompanying refinement along the wall surface. As more levels of grid refinement are used, the number of grid points tends to mushroom, and a larger and larger percentage of the grid points tend to cluster in the boundary layers.

The solution was obtained using four levels of multigrid, resulting in a final mesh with 95,829 nodes and 477,185 cells. Figure 10 shows the calculated Mach numbers and pressure contours on the pitchline. Two blade passages are shown to illustrate more clearly the periodicity of the solution. The shock at the trailing edge and the wake region behind the trailing edge are clearly visible. The passage shock is somewhat stronger and the trailing shock is somewhat weaker that computed with the structured grid code for this same case. Because of the coarseness of the initial unstructured mesh,

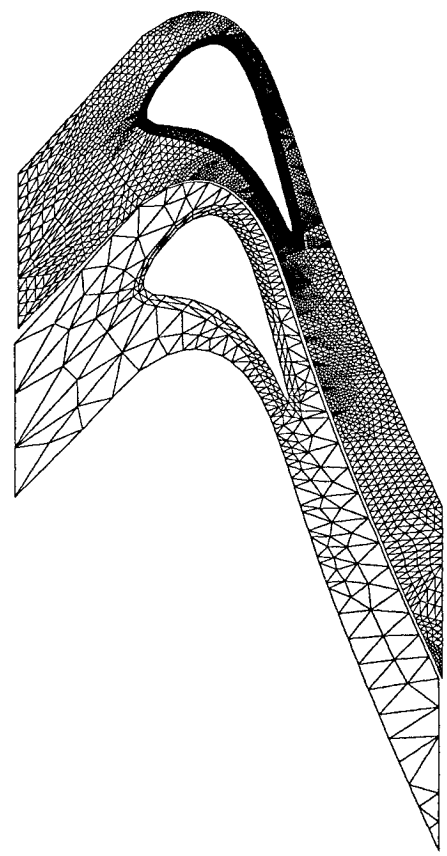
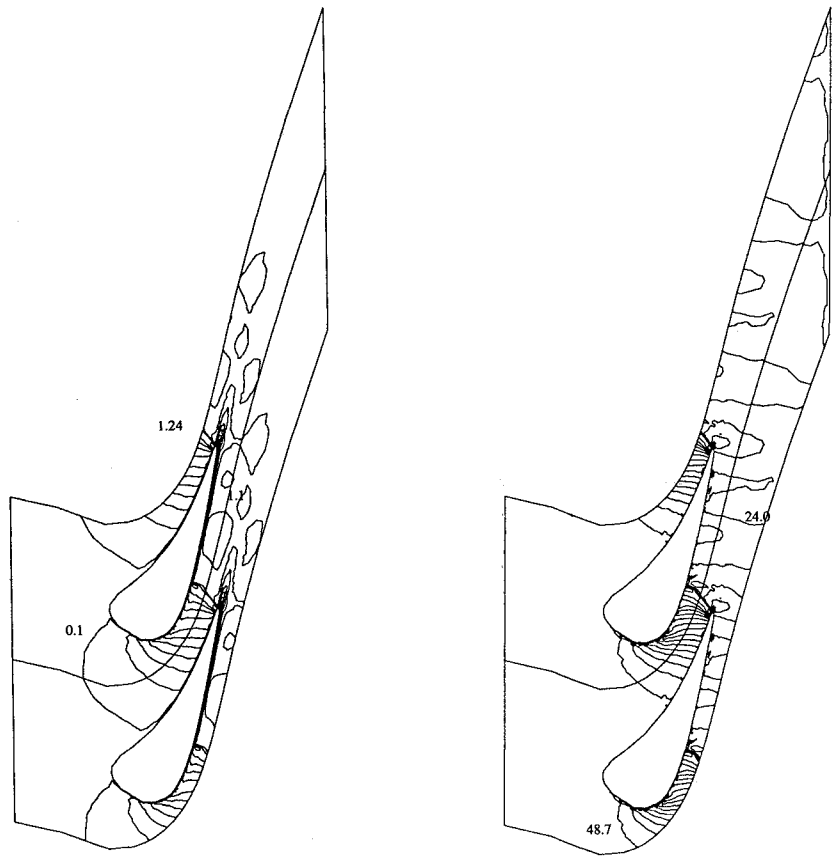


Fig. 9 Initial and refined meshes on hub surface.



a) Mach number b) Pressure

Fig. 10 Calculated contours on pitchline.

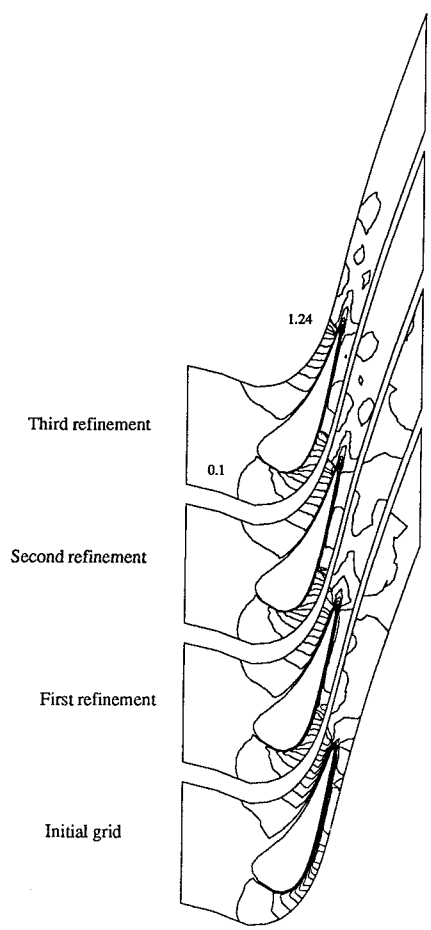


Fig. 11 Effect of mesh refinement on calculated Mach numbers.

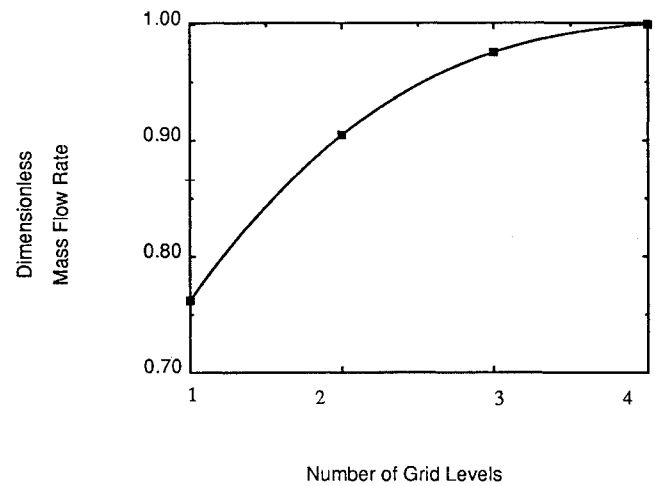


Fig. 12 Effect of grid level on mass flow rate.

the geometry of the curved trailing edge is not particularly well resolved, and this probably accounts for this discrepancy.

Figure 11 illustrates the effect of mesh refinement on the calculated Mach numbers on the pitchline halfway between the hub and the casing. The refiner adds grid points primarily near the trailing edge shock and in the wake region behind the blade. The calculated mass flow increases asymptotically as the grid is refined, as shown in Fig. 12.

C. Case 3: Compressor Rotor

The third problem considered here involves the transonic flow in a compressor rotor. The initial grid for this problem was obtained by the destructuring of a structured grid with 38,073 grid points. This initial destructured grid is shown in Fig. 13. The solution was computed on a grid that was refined once, resulting in a mesh with

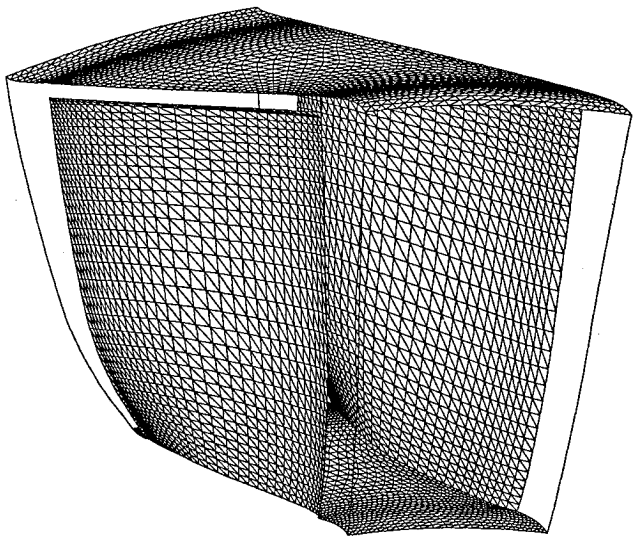


Fig. 13 Destructured grid for compressor rotor.

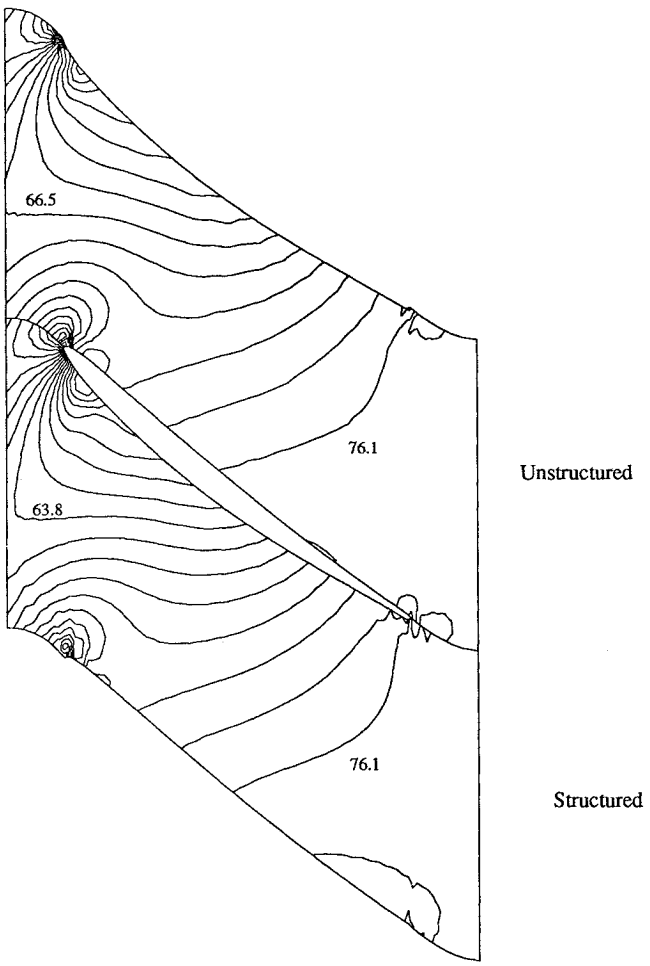


Fig. 14 Calculated pressure contours for compressor rotor.

84,035 grid points and 440,234 cells. The outer shroud remains stationary, and a tip gap is present between the rotor blades and the shroud.

Figure 14 compares the pressure contours calculated with the adaptive unstructured algorithm described here with earlier results computed with the structured grid code described in Ref. 17, and Fig. 15 shows a corresponding Mach number comparison. The results are shown for a radial plane about halfway between the hub and shroud. The results are found to be in good agreement with the earlier structured code results.

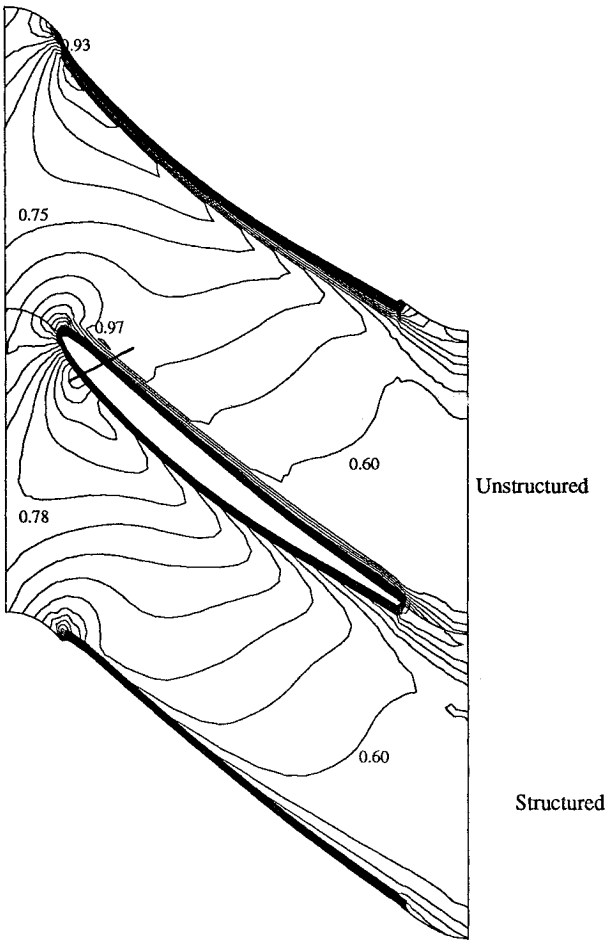


Fig. 15 Calculated relative Mach number contours for compressor rotor.

X. Concluding Remarks

A three-dimensional adaptive multigrid algorithm for solving the Navier–Stokes equations on unstructured tetrahedral meshes has been developed and demonstrated on a variety of important industrial problems. The power and flexibility of the algorithm in treating flows in complicated geometries with accurate resolution of the flow features makes the algorithm very attractive.

The use of purely tetrahedral meshes has some drawbacks. The computed solutions tend to be somewhat noisy in regions of high flow gradients, and an excessive number of points may be inserted by the refiner in the boundary layers. The need for one-dimensional refinement in such regions necessitates the use of prismatic or hexahedral elements in the boundary layer. It is likely that the multigrid will also be more effective for viscous cases when the coarsening of the boundary-layer cells can be done in a one-dimensional fashion. The surface inflation/advancing front grid generation method provides a natural means of generating meshes with prismatic elements in the boundary layers and tetrahedral elements elsewhere. Development of the capability to generate, perform one-dimensional refinement, and solve on such mixed meshes is the focus of our current research.

Solution times for these three-dimensional calculations still remain rather long. The addition of multigrid by agglomeration[19] to the present refinement scheme would allow the use of more multigrid levels. The initial grid could be made fine enough to adequately resolve the geometry, with the coarser grid levels generated by agglomeration and the finer grid levels generated through refinement.

Acknowledgments

The authors would like to acknowledge the General Electric Company for permission to publish this paper. Thanks are also due to our colleagues D. G. Holmes and J. Cofer.

References

[1]Ucer, A. S. (ed.), *Turbomachinery Design Using CFD*, AGARD Lecture Series 195, NATO, 1994.

[2]Burrus, D. L., "Application of Numerical Models for Predictions of Turbine Engine Combustor Performance," ASME Paper 89-GT-251, American Society of Mechanical Engineers, June 1989.

[3]Mavripilis, D. J., "Three Dimensional Unstructured Multigrid for the Euler Equations," *AIAA Journal*, Vol. 30, No. 7, 1992, pp. 1753–1761.

[4]Peraire, J., Morgan, K., and Peiro, J., "Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD," AGARD Specialists Meeting (Loen, Norway), May 1989.

[5]Dawes, W. N., "The Extension of a Solution Adaptive 3D Navier–Stokes Solver Towards Geometries of Arbitrary Complexity," ASME Paper 92-GT-363, American Society of Mechanical Engineers, 1992.

[6]Löhner, R., "Adaptive Remeshing for Transient Problems with Moving Bodies," AIAA Paper 88-3737, 1988.

[7]Connell, S. D., Holmes, D. G., and Braaten, M. E., "Adaptive Unstructured 2D Navier–Stokes Solutions on Mixed Quadrilateral/Triangular Meshes," ASME IGTI, Turbo–Expo, ASME Paper 93-GT-99 (Cincinnati, OH), American Society of Mechanical Engineers, May 1993.

[8]Spragle, G. S., Smith, W. A., and Yadlin, Y., "Application of an Unstructured Flow Solver to Planes, Trains, and Automobiles," AIAA Paper 93-0089, 1993.

[9]Weatheril, N. P., "Adaptive Inviscid Flow Solution for Aerospace Geometries on Efficiently Generated Unstructured Tetrahedral Meshes," AIAA Paper 93-3390, 1993.

[10]Shepperd, M. S., and Georges, M. K., "Automatic Three Dimensional Mesh Generation by the Finite Octree Technique," *International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 709–739.

[11]Connell, S. D., and Holmes, D. G., "A 3D Unstructured Adaptive Multigrid Scheme for the Euler Equations," AIAA Paper 93-3339, Jan. 1993.

[12]Holmes, D. G., and Connell, S. D., "Solution of the 2D Navier–Stokes Equations on Unstructured Adaptive Grids," AIAA Paper 89-1932, June 1989.

[13]Jameson, A., and Baker, T. J., "Improvements to the Aircraft Euler Method," AIAA Paper 87-0452, 1987.

[14]Varma, R. R., and Caughey, D. A., "Evaluation of Navier–Stokes Solutions Using the Integrated Effect of Numerical Dissipation," *AIAA Journal*, Vol. 32, No. 2, 1994, pp. 294–300.

[15]Martinelli, L., "Calculation of Viscous Flows with a Multigrid Method," Ph.D. Thesis, Dept. of Mechanical and Aerospace Engineering, Princeton Univ., Princeton, NJ, Oct. 1987.

[16]Connell, S. D., and Braaten, M. E., "Semi-Structured Mesh Generation for Three-Dimensional Navier–Stokes Calculations," *AIAA Journal*, Vol. 33, No. 6, 1995, pp. 1017–1024.

[17]Turner, M. G., and Jennions, I. K., "An Investigation of Turbulence Modeling in Transonic Fans Including a Novel Implementation of an Implicit k–ε Turbulence Model," *Journal of Turbomachinery*, Vol. 115, 1992, pp. 249–260.

[18]Turner, M. G., Liang, T., Beauchamp, P. P., and Jennions, I. K., "The Use of Orthogonal Grids in Turbine CFD Calculations," ASME Paper 93-GT-38, American Society of Mechanical Engineers, 1993.

[19]Venkatakrishan, V., and Mavriplis, D., "Agglomeration Multigrid for the 3D Euler Equations," AIAA Paper 94-0069, Jan. 1994.