Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes

D. J. Mavriplis*
NASA Langley Research Center, Hampton, Virginia

A multigrid algorithm has been developed to accelerate the convergence of the Euler equations to a steady state on unstructured triangular meshes. The method operates on a sequence of coarse and fine unstructured meshes and assumes no relation exists between the various meshes. A tree-search algorithm is used to efficiently identify regions of overlap between coarse and fine grid cells. The multigrid method is applied to two different discretizations of the Euler equations, and results about multi-element airfoils are presented. For both discretization schemes, convergence is accelerated by an order of magnitude, resulting in convergence rates comparable to those obtained with structured multigrid algorithms.

Introduction

STEADY-STATE solutions of the Euler equations about complex geometries have generally been impeded by the ability to generate adequate meshes about such configurations and/or the efficiency with which the Euler equations may be solved on these meshes. Two basic approaches to the problem have emerged in the literature. The first, and perhaps more traditional, involves the use of structured meshes of quadrilateral elements. The difficulties involved in generating such meshes rapidly increase with the geometric complexity of the configuration. This problem can be relieved somewhat by resorting to hybrid or overlapping meshes[1] and to a certain extent, adaptive meshes.[2] However, these techniques introduce additional complexities in the flow solution algorithm, which must be capable of transferring information back and forth between the various local meshes or refined regions.

The alternative approach is to employ completely unstructured meshes. These can most easily be constructed using triangular elements in two dimensions, and the mesh generation problem for complex configurations can be simplified considerably. However, unstructured mesh solvers suffer from inherent efficiency limitations. For example, megaflop rates attainable on present-day supercomputers are lower than for structured mesh solvers, due to the gather and scatter operations that are needed to form contiguous vectors of variables. One can at best hope to minimize these effects by the use of efficient algorithms, coding, and computer architecture.

On the other hand, it is also evident that the development of unstructured mesh flow solvers has not kept pace with advances in structured mesh solvers. Many of the ideas developed for structured mesh solvers, such as approximate factorization and nested multigrid methods, cannot be applied to unstructured meshes. They must either be modified or abandoned in favor of more general algorithms.

It is the objective of this paper to demonstrate how the efficiency of an unstructured mesh flow solver can be made competitive with existing structured mesh solvers. This has been achieved with the use of a novel multigrid algorithm that operates on a sequence of unstructured, unrelated meshes. The algorithm has been applied to two different discretizations of the Euler equations, which will be described briefly.

Received March 23, 1987; revision received Nov. 7, 1987.

*Staff Scientist, Institute for Computer Applications in Science and Engineering. Member AIAA.

Mesh Generation

The generation of unstructured triangular meshes about multielement airfoils is accomplished in three essentially independent steps. First, a set of mesh points in the flowfield is generated. This is accomplished by generating a regular 0-mesh about each airfoil element, resulting in a set of overlapping structured meshes. If the mesh cells are ignored, and the points that lie inside neighboring airfoil elements are omitted, a distribution of points in the flowfield is obtained. These points are then joined together by line segments to form a set of triangular elements using the Delaunay Triangulation algorithm.[3] There exist many ways of triangulating a given set of points. The Delaunay algorithm represents a unique construct of this type. It also has the desirable property of minimizing the aspect ratios of the triangular cells. Further details on Delaunay triangulation can be found in Refs. 3–5. The resulting mesh is then postprocessed by a smoothing filter, which slightly repositions the mesh points to ensure a distribution of smoothly varying elements. The new position of a mesh point is calculated as

$$x_i^{\text{new}} = x_i^{\text{old}} + \frac{\omega}{n} \sum_{k=1}^{n} (x_k - x_i) \tag{1}$$

with a similar expression for the y-coordinate; ω is a relaxation factor, and the sum is over all edges meeting at point i.

Discretization of the Equations

The values to be determined are the pressure, density, Cartesian velocity components, total energy, and total enthalpy, denoted by p, ρ, u, E, and H, respectively. Since for a perfect gas we have the relations

$$E = \frac{p}{(\gamma - 1)\rho} + \frac{1}{2}(u^2 + v^2), \qquad H = E + \frac{p}{\rho} \tag{2}$$

where γ is the ratio of specific heats, we need only solve for the four variables ρ, ρu, ρv, and ρE. These values are determined by solving the Euler equations, which in integral form read

$$\frac{\partial}{\partial t} \iint_{\Omega} w \, dx \, dy + \int_{\partial\Omega} (f \, dy - g \, dx) = 0 \tag{3}$$

where x and y denote Cartesian coordinates, and the solution and flux vectors are

$$w = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \qquad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix} \qquad g = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{bmatrix} \tag{4}$$

The letter Ω represents an arbitrary control volume with fixed boundary $\partial\Omega$.

These equations may be discretized by dividing the flowfield into a large number of small control volumes (i.e., generating a mesh) and then applying Eq. (3) over each individual control volume. In this work, two such finite-volume discretization schemes have been adopted.

Cell-Centered Scheme

This represents the extension to unstructured meshes of the scheme originally proposed in Ref. 6. The control volumes are taken as the triangular elements of the mesh (see Fig. 1). The flow variables are stored at the center of each cell and assumed to represent an average value over the entire control volume. In order to evaluate the boundary integral in Eq. (3), estimates of the flow variables along the edges of the triangular cell are needed. These are taken as the average of the values in both cells on either side of that edge. This is the equivalent of central differencing on a Cartesian grid and is second-order accurate for a smooth mesh.

Nodal Scheme

This is the two-dimensional equivalent of the scheme proposed in Ref. 4. The flow variables are stored at the vertices of the triangles. The control volume for a particular node i is taken to be the union of all triangles with a vertex at i, as shown in Fig. 2. The boundary integral in Eq. (3) is approximated by the trapezoidal integration rule: for each edge delimiting the control volume boundary, the contribution to the boundary integral is obtained by evaluating the fluxes at the nodes on both ends of the edge and then taking the scalar product of their averaged value with the directed length of the edge. This discretization can be shown to be equivalent to a finite-element Galerkin approximation,[4] which is known to be second-order accurate in space for a smooth mesh with a lumped mass matrix.

Artificial dissipation is added to both schemes as a finite-volume approximation to an undivided Laplacian operator in the vicinity of a shock, and as an approximation to an undivided biharmonic operator in regions of smooth flow. This formulation is analogous to the blended second and fourth differences used by Jameson for structured quadrilateral meshes.[7] For both schemes, on smoothly varying meshes, this construction of the dissipation results in second-order accuracy everywhere except near a shock, where the schemes become locally first-order accurate. Further details can be found in Refs. 6 and 8 for the cell-centered scheme and in Refs. 4 and 8 for the nodal scheme.

Integration to a Steady State

Discretization of the Euler equations in space transforms the governing equations into a set of coupled ordinary-differential equations that must be integrated in time to obtain the steady-state solution. Thus, Eq. (3) becomes the set

$$S_i \frac{dw_i}{dt} + [Q(w_i) - D(w_i)] = 0, \qquad i = 1, 2, \ldots \tag{5}$$

where S_i is the area of the control volume i and is independent of time. These equations are integrated in time using a fully explicit five-stage hybrid time-stepping scheme, where the convective operator $Q(w)$ is evaluated at each stage in the time step, and the dissipative operator $D(w)$ is only evaluated in the first two stages. Thus, we advance in time as

$$w^{(0)} = w^n \tag{6a}$$

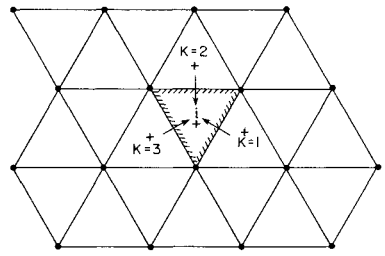

Fig. 1. Cell-centered scheme.

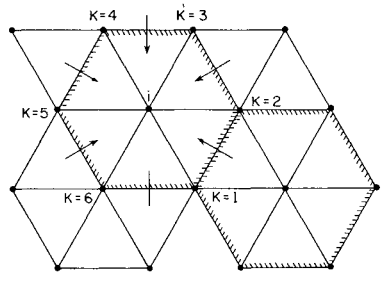

Fig. 2. Nodal scheme.

$$w^{(1)} = w^{(0)} - \alpha_1 \Delta t/S\,[Q(w^{(0)}) - D(w^{(0)})] \tag{6b}$$

$$w^{(2)} = w^{(0)} - \alpha_2 \Delta t/S\,[Q(w^{(1)}) - D(w^{(1)})] \tag{6c}$$

$$\vdots$$

$$w^{(5)} = w^{(0)} - \alpha_5 \Delta t/S\,[Q(w^{(4)}) - D(w^{(1)})] \tag{6d}$$

$$w^{n+1} = w^{(5)} \tag{6e}$$

where w^n and w^{n+1} are the values at the beginning and the end of the nth time step. The standard values of the coefficients are

$$\alpha_1 = \frac{1}{4}, \qquad \alpha_2 = \frac{1}{6}, \qquad \alpha_3 = \frac{3}{8}, \qquad \alpha_4 = \frac{1}{2}, \qquad \alpha_5 = 1$$

This scheme represents a particular case of a large class of hybrid time-stepping schemes, which has been specifically designed to produce strong damping characteristics of high-frequency error modes. It is, thus, well suited to drive the multigrid algorithm.

Convergence to a steady state is also accelerated by using the maximum permissible time step at each point in the flowfield as determined by local stability analysis, by the use of enthalpy damping,[7] and implicit residual averaging.[6,8]

Data Structure and Vectorization

For structured grids, mesh coordinate directions can be identified and used to number the cells and nodes. Thus, a cell may be identified by its two indices I and J in the mesh coordinate system, and its neighbors may be located by incrementing one of these indices. However, for unstructured meshes this is no longer possible, since in principle the cells and nodes are ordered randomly. Thus, the use of unstructured meshes requires the storage of connectivity information along with the use of an indirect addressing system.

Since fluxes are to be calculated across each edge of the mesh for both the cell-centered and the nodal scheme, a data structure based on the mesh edges can be employed. To define an edge-based structure, four integer addresses must be stored for each edge. Thus, an array must be dimensioned $ND(NEDGE,4)$ where $NEDGE$ is the total number of edges in the mesh. For each edge, the first two values in ND, $N1$, and $N2$, correspond

to the address of the nodes on either end of that edge, as shown in Fig. 3. For the cell-centered scheme, the two remaining values $I1$ and $I2$ denote the address of the triangular cells on either side of that edge. In the case of the nodal scheme, these two values are taken as $N3$ and $N4$, the addresses of the remaining vertex of each triangle on either side of that edge, which are also the addresses of the nodes associated with the two control volumes delimited by that edge.

With this information, a complete flux balance over the entire flowfield can be accomplished with a simple loop over the edges. For example, for the nodal scheme, a simplified loop is constructed as

$DO\ 10\ I = 1,\ NEDGE$

$N1 = ND(I,1)$

$N2 = ND(I,2)$

$N3 = ND(I,3)$

$N4 = ND(I,4)$

$\text{Flux} = FCTN(\text{variables at } N1 \text{ and } N2)$

$\text{Residual}(N3) = \text{Residual}(N3) + \text{Flux}$

$10\ \text{Residual}(N4) = \text{Residual}(N4) - \text{Flux}$

The flux at each edge is calculated only once and then added to and subtracted from the two control volumes on either side of that edge, thus ensuring conservation. The above loop will not readily vectorize due to recurrences in the last two statements. This arises from the fact that the individual control volumes (i.e., $N3$ and $N4$ in this case) receive contributions from more than one edge in the mesh. This problem can be circumvented by reordering the edges and grouping them into several distinct groups such that, within each group, no control volume of the mesh is accessed more than once. This, however, results in a reduction of the vector lengths that can be used in a flux balance calculation.

Unstructured Multigrid Algorithm

The basic idea of a multigrid strategy is to perform time steps on coarser meshes to calculate corrections to a solution on a finer mesh. The advantages of time stepping on coarse meshes are twofold: first, the permissible time step is much larger since it is proportional to the mesh width, and secondly, the work involved is much less because of the smaller number of grid points. The use of multigrid methods with unstructured meshes presents an additional challenge. Coarser grids can no longer be generated by amalgamating groups of fine grid cells. An alternative that has been suggested[9] is to generate fine meshes by repeatedly subdividing cells of a coarse unstructured mesh in some manner. However, generally poor topological control of the fine mesh results from such a procedure. It is also improbable that the extremely coarse meshes (with only two or three cells between inner and outer boundaries), which are instrumental in achieving rapid convergence rates, can be used as the starting point of the subdivision.

It was thus decided to pursue an unstructured multigrid method in which a sequence of completely unrelated coarse and fine grids are employed. In fact, the underlying theory of multigrid does not assume any relation between the various meshes, merely that variables can be transferred back and forth between them. This approach provides great flexibility in determining the configuration of both the finest and the coarsest grid. To our knowledge, the only previous attempt of this type has been by Lohner and Morgan.[10] One of the key elements for the success of such a method is the development of efficient transfer mechanisms between grids, as well as a rapid-search algorithm to enable identification of overlapping cells between two grids.

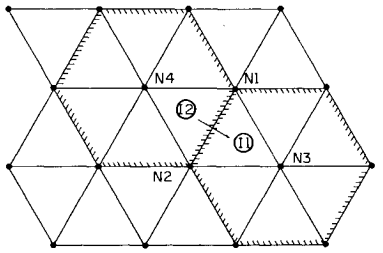

Fig. 3 Edge data structure.

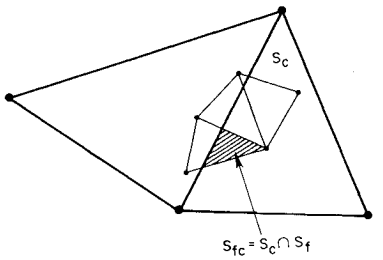

Fig. 4. Intersecting area between coarse and fine grid cells.

Grid Transfers for the Cell-Centered Scheme

An area weighting transfer rule is used to transfer flow variables and residuals from the fine grid to the coarse grid in the cell-centered scheme. If S_{fc} represents the area of intersection between a coarse grid cell and a fine grid cell, when both meshes are superimposed as shown in Fig. 4, the flow variables on the coarse grid are determined by

$$w_c = \sum_{\text{fine grid cells}} \frac{w_f S_{fc}}{S_c} \tag{7}$$

where w_c is the flow variable of a particular coarse grid cell with area S_c, and w_f is the flow variable associated with each fine grid cell. The summation is taken over all fine grid cells. Similarly, the residuals are transferred to the coarse grid as

$$R_c = \sum_{\text{fine grid cells}} \frac{R_f S_{fc}}{S_f} \tag{8}$$

where the subscripts refer to coarse and fine values. This procedure is conservative, and for a sequence of structured, nested meshes reduces to the transfer rules used by Jameson.[6,11] The transfer of corrections from the coarse grid back to the fine grid follows the rule

$$c_f = \sum_{\text{coarse grid cells}} \frac{c_c S_{fc}}{S_f} \tag{9}$$

where c_f is the transferred correction at a particular fine grid cell of area S_f, c_c is the coarse grid correction, and the summation is taken over all coarse grid cells. Furthermore, the transferred corrections c_f are smoothed by averaging them with their neighbors. This constitutes an approximation to the bilinear interpolation formula used by Jameson.[11]

In the above transfer formulae, the summations are to be carried out over all cells of the coarse or fine grid. In practice, these summations need only cover those cells that have a nonzero intersecting area S_{fc} with the coarse grid cell of Eqs. (7) and (8), or the fine grid cell of Eq. (9). Thus, the key to achieving these grid transfers is the determination of the intersecting areas. This can be reduced through geometrical reasoning to the problem of locating the coarse grid cell that encloses a particular fine grid node.

Grid Transfers for the Nodal Scheme

For the nodal scheme, the flow variables, residuals, and corrections are transferred in different manners. Flow variables at a coarse grid node P are taken as the linear interpolation of the corresponding values at nodes 1, 2, and 3, as shown in Fig. 5, which are the vertices of the fine grid triangle enclosing P. These three nodes include the fine grid node that is closest to P, thus ensuring an accurate representation of the flowfield on the coarse grid.

The fine grid residual R_a at "a" in Fig. 5 is linearly distributed to the coarse grid nodes A, B, and C, which are the vertices of the coarse grid triangle enclosing "a". This linear distribution is accomplished by the use of shape functions that have the value 1 at one of the coarse grid triangle vertices and vanish at the other two vertices. This implies that the sum of the residual contribution to A, B, and C equals the residual at "a", and the weighting is such that if "a" and A coincide, then the contribution at A is equal to R_a, and the contributions at B and C vanish. This type of transfer is conservative.

When transferring the corrections from the coarse grid back to the fine grid, a simple linear interpolation formula is used. Thus, the correction at the fine grid node "a" is taken as the linear interpolation of the three corrections at nodes A, B, and C that enclose "a" on the coarse grid.

The remaining difficulty lies in the determination of the nodes A, B, and C, to be associated with each fine grid node "a". This is equivalent to the problem of locating the address of the coarse grid cell that encloses a particular fine grid node.

Search Algorithm

It turns out that the grid transfer problem for both schemes can be reduced to that of locating the coarse grid cell IC that encloses a particular fine grid node NF. A naive search over all the coarse grid cells would require $0(N^2)$ operations, where N is the number of grid points, and thus would be prohibitively expensive, requiring more time than the flow solution itself. Hence, an efficient search algorithm is needed. In this work, a tree-search algorithm has been adopted. It requires information about the neighbors of each node or cell to be stored for both the coarse and the fine grid. It is initiated by providing an initial guess IC_1 for the coarse grid cell, and then testing IC_1 to see if it encloses the fine grid node NF. Since we are free to begin the search with any NF of the fine grid and any IC_1 of the coarse grid, we choose points whose locations are known (such as the trailing-edge values). If the test is negative, then the neighbors of IC_1 are tested. If these tests also fail, then the neighbors of these neighbors are tested. This process is continued until the address IC_n of the cell enclosing NF is located. This entire procedure must be repeated for every node of the fine grid. The next fine grid node NF_2 is thus chosen as a neighbor of NF, and the initial guess for the enclosing cell is taken as IC_n, the coarse grid cell which is now known to enclose the previous NF. In this manner we are assured of a good initial guess, since IC_n and NF_2 must be located in the same region of the computational domain.

In practice, it is found that an average of 10 searches are required to locate an enclosing cell. Furthermore, this value does not depend on the mesh size, but rather on the relative difference in mesh densities between the fine and coarse grids.

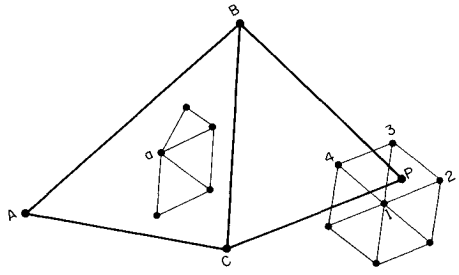

Fig. 5. Grid transfers for nodal multigrid scheme.

Multigrid Algorithm

The first step of the unstructured multigrid procedure involves the construction of a complete sequence of unstructured meshes generated independently by the method described previously, and designed such that each coarser mesh has roughly one-quarter the number of mesh points as the previous mesh. Next, the grid transfer coefficients are calculated and stored using the search algorithm described above, in a preprocessing stage prior to the flow solution phase. The first time step is taken on the finest grid, and the flow variables and residuals are then transferred to the next coarser grid. If R' represents the transferred residuals and w' the transferred flow variables, a forcing function on the coarse grid may be defined as

$$P = R' - R(w') \tag{10}$$

Now, on the coarse grid, time stepping proceeds as

$$w^{(q)} = w^{(0)} - \alpha_q \Delta t[R(w^{(q-1)}) + P] \tag{11}$$

for the qth stage. In the first stage, $w^{(q-1)}$ reduces to the transferred flow variable w'. Thus, the calculated residuals on the coarse grid are cancelled by the second term in the forcing function P, leaving only the R' term, indicating that the coarse grid solution is driven by the fine grid residuals. This procedure is repeated on successively coarser grids, performing one time step on each grid level. When the coarsest grid has been reached, the corrections are transferred back to the finer grids without any intermediate time stepping.

Results

The first test case involves a two-element airfoil system in supercritical flow for which experimental data is available. The basic configuration, which consists of a main airfoil fitted with a leading-edge slat, has been the subject of extensive wind-tunnel investigations to determine the effectiveness of slats as a transonic maneuvering aid for fighter configurations.[12] The particular case chosen was found by Volpe[12] to exhibit minimal viscous effects. The Mach number is 0.5 and the angle of incidence is 7.5 deg. The complete sequence of fine and coarse meshes used in the multigrid algorithm is shown in Fig. 6. The finest mesh contains 5629 mesh points, with 128 points on the main airfoil and 80 points on the slat. The coarsest mesh contains a total of merely 114 mesh points. The calculated pressure distributions for both schemes are plotted vs the experimental data in Fig. 7. Agreement is generally very good and all features of the flow are reproduced by the numerical schemes. The slat is supercritical with a shock appearing on its upper surface and the main airfoil is subcritical. Both the position and strength of the shock are well predicted by the two schemes. Agreement degrades only in regions of strong viscous interactions such as at the trailing edges of both airfoils, and at the foot of the shock, where rapid boundary-layer growth occurs. The convergence of both schemes is depicted in Fig. 8 as measured by the L_2 norm of the density residual, and also the buildup of supersonic points in the flowfield. For the cell-centered scheme, the residuals are reduced by almost 8 orders of magnitude in 200 multigrid cycles, corresponding to an average convergence rate of 0.9162 per multigrid cycle. Convergence of the nodal scheme is even more rapid, yielding an average rate of 0.8854, with a decrease of the residuals by over 10 orders of magnitude in 200 cycles. For both schemes, the total number of supersonic points as well as the lift and drag coefficients are frozen in less than 100 cycles. Engineering calculations may be performed for this case using 50–75 multigrid cycles. This represents an order of magnitude increase in convergence rate as compared to the single grid solution. Furthermore, these convergence rates are comparable to those obtained with structured multigrid Euler solvers.[6,11]

The convergence rate of a properly formulated multigrid algorithm should be independent of the size and complexity of the finest mesh, and should uniquely depend on the coarsest

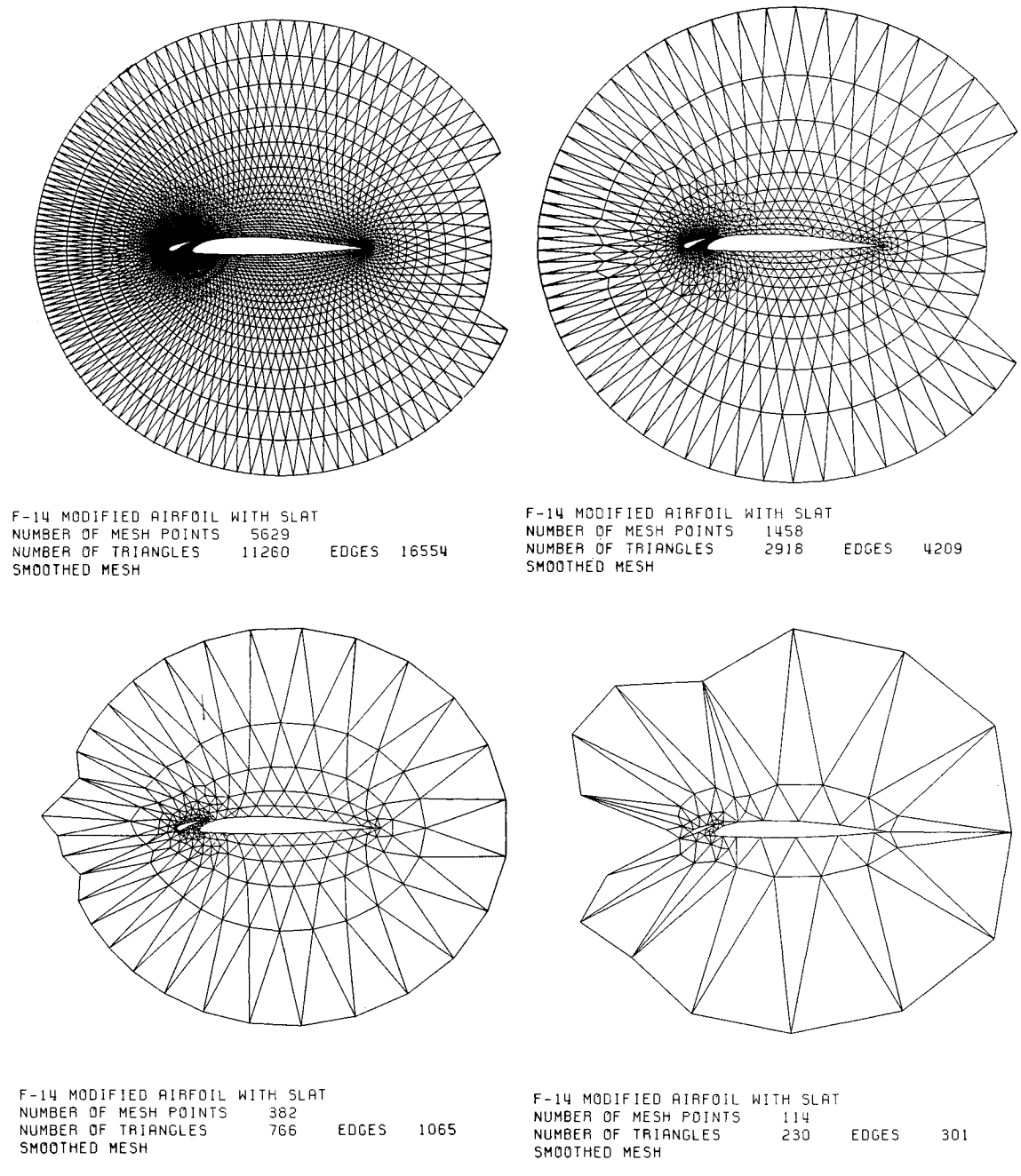

Fig. 6 Sequence of unstructured meshes about the airfoil-slat system used for the unstructured multigrid algorithm (partial view of meshes only).

mesh of the sequence. This type of behavior is demonstrated in Table 1 for the present unstructured multigrid algorithm by comparing the convergence rate of the above problem on each grid of the sequence depicted in Fig. 6, when solving on that grid alone, with that obtained by using a multigrid sequence with the maximum possible number of grids. The degradation of the multigrid convergence rate as finer grids are added is seen to be minimal, especially when compared to the single grid rates.

The next test case is that of a four-element airfoil system. The geometry represents a section of a wing of a commercial aircraft in its landing configuration. The Mach number is taken as 0.3 and the angle of incidence is 5 deg. Although no data are available for comparison, this case is intended to serve as a demonstration of the flexibility and efficiency of the present Euler solvers in dealing with very complex geometries. The finest mesh, shown in Fig. 9, contains 6322 points, including 70 points on the leading-edge slat, 100 points on the main airfoil, and 70 and 80 points on the intermediate and trailing-edge flaps, respectively. The coarsest mesh contains a total of 149 points. The surface pressure distributions for each element of this configuration calculated by the cell-centered scheme are given in Fig. 10. The pressure distributions calculated by the nodal scheme are omitted since they are essentially identical.

An average residual reduction of 3 orders of magnitude, which is sufficient for engineering calculations, can be achieved with 120 multigrid cycles for the cell-centered scheme and 75 cycles for the nodal scheme. The relative improvements in convergence can be assessed from Figs. 11 and 12, where convergence on a single grid, multiple grids, and multiple grids with residual averaging are shown. The L_2 norm of the density residuals are plotted vs the actual CPU time, normalized by the time required for a single time step on the finest grid alone without residual averaging. The real gains in efficiency due to these techniques can be assessed from the figures. The time spent in the multigrid preprocessing stage, although larger for the cell-centered scheme than for the nodal scheme, is seen to be essentially negligible. For the former, this roughly corresponds to

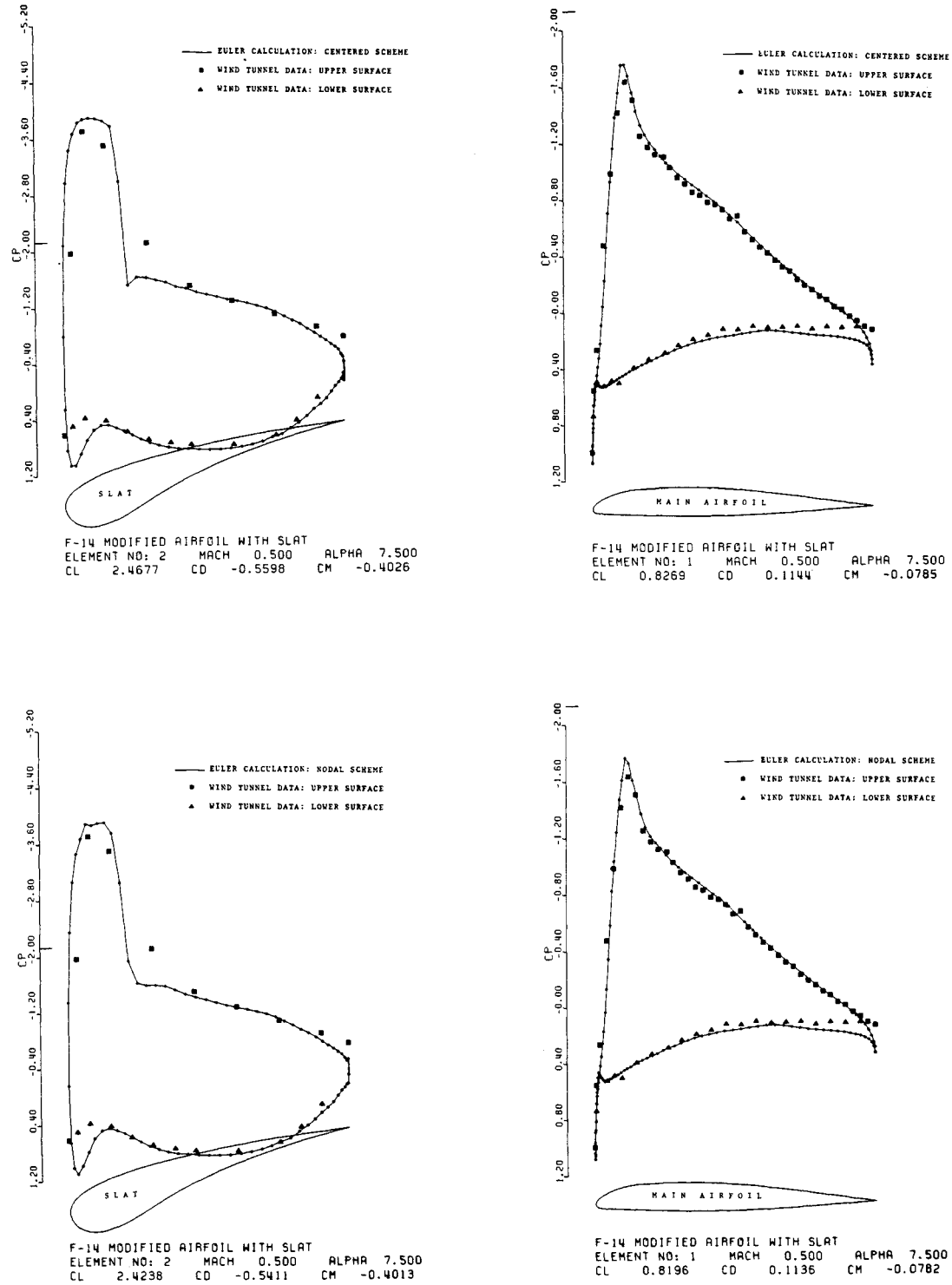

Fig. 7 Comparison of computed surface pressure distribution for the airfoil-slat system with experimental wind-tunnel data, Mach number = 0.5, $\alpha = 7.5$ deg.

the time required by five multigrid cycles. A multigrid cycle using the complete sequence of four meshes requires a 100% increase in CPU time over the single grid cycle for the cell-centered scheme. This increase can be broken down as follows: 27% due to time stepping on coarser grids, 38% for extra residual evaluation prior to the transfer to a coarser grid, and 35% due to the actual transfer of variables between grids. For the nodal scheme, a multigrid cycle requires a 67% increase in CPU time: 24% for time stepping on coarser grids, 35% for extra residual evaluation, and 5% for the transfer of variables between grids. In addition to this, the residual averaging operation requires roughly a 50% increase in CPU time over the base multigrid cycle for both schemes. Thus, for the multigrid cycle with residual averaging, 13% of the total time is spent in transferring variables between meshes for the cell-centered scheme, whereas this figure drops to 2% for the nodal scheme. Clearly, the more complicated area weighting rule used for transfers in the cell-centered scheme introduces a time penalty. For the nodal scheme, this case required 0.75 CPU seconds per multigrid cycle on the CRAY-2 supercomputer. Thus, an engineering calculation of 75 cycles, for example, would require roughly one minute of CPU time.

Table 1 Average convergence rates of both schemes on each grid of the sequence with and without the multigrid algorithm for the supercritical airfoil-slat system

	Number of mesh nodes	Average residual reduction per single grid cycle	Average residual reduction per multigrid cycle	Number of grid levels
Cell-centered scheme	114	0.897	—	1
	382	0.950	0.890	2
	1458	0.980	0.907	3
	5629	0.994	0.916	4
Nodal scheme	114	0.841	—	1
	382	0.913	0.847	2
	1458	0.965	0.848	3
	5629	0.981	0.885	4

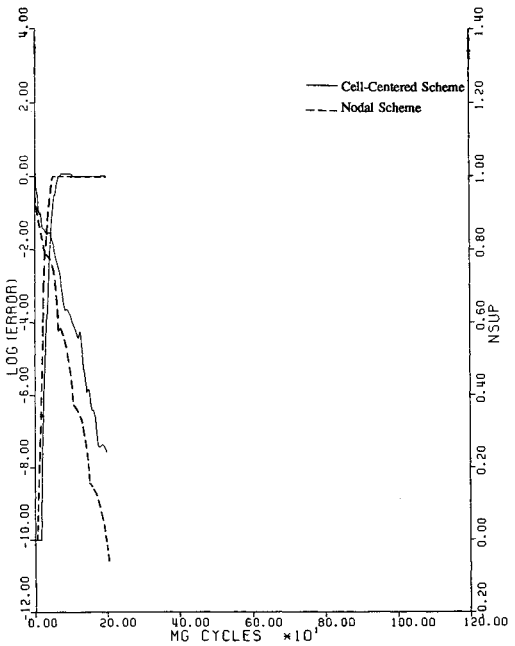

Fig. 8. Convergence rate of both schemes for the supercritical airfoil-slat system as measured by the RMS average of the density residual and the buildup of supersonic points.

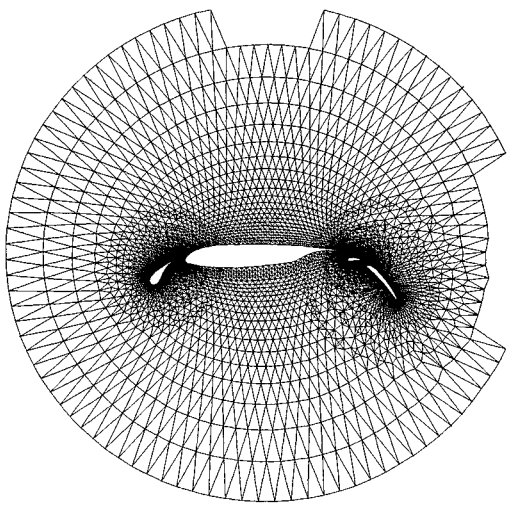

Fig. 9. Generated fine mesh for the four-element airfoil configuration.

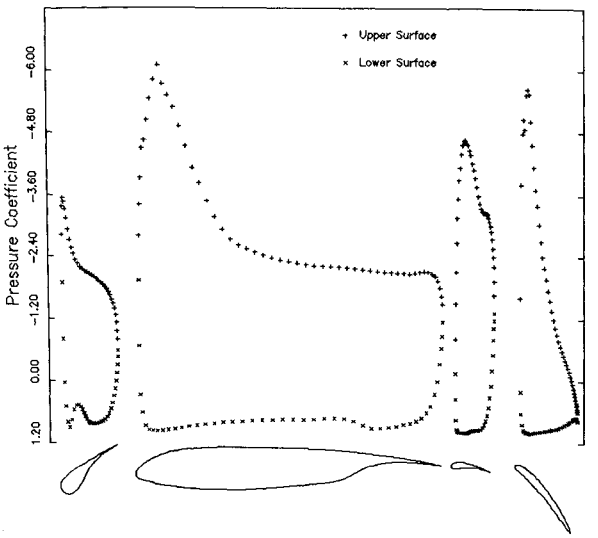

Fig. 10. Surface pressure distribution computed with the cell-centered scheme for the four-element airfoil system, Mach number = 0.3, $\alpha = 5$ deg.

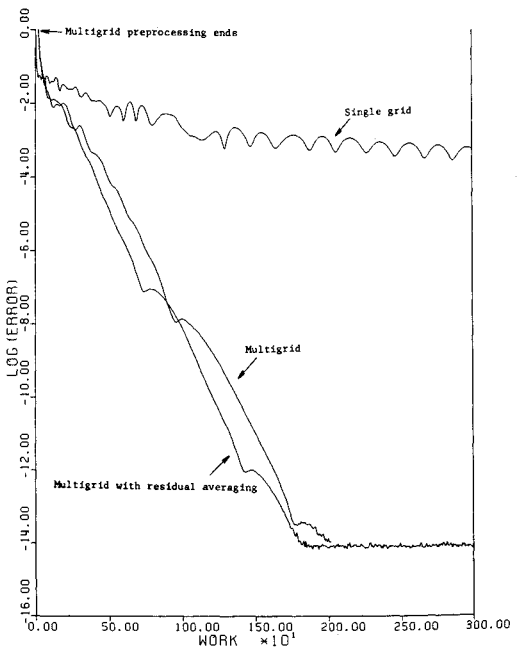

Fig. 11. Convergence history of the cell-centered scheme for the four-element airfoil configuration as measured by the RMS average of the density residual vs CPU time normalized as work units. (A work unit is the time required for a single fine grid time step.)

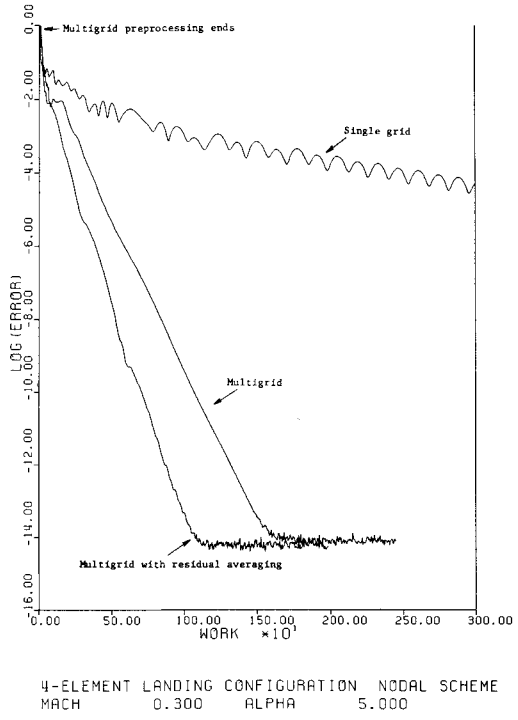

Fig. 12. Convergence history of the nodal scheme for the four-element airfoil configuration as measured by the RMS average of the density residual vs CPU time normalized as work units. (A work unit is the time required for a single fine grid time step.)

Conclusions

Both unstructured triangular mesh solvers have been shown to exhibit convergence rates comparable with those obtained by available structured mesh solvers. Efficiency increases of an order of magnitude have been demonstrated by the use of the unstructured multigrid algorithm. The time requirements for both schemes are roughly equivalent. The nodal scheme, for example, was found to run at a speed of 0.12×10^{-3} s/MG cycle/node on the CRAY-2. By comparison, FL052, a structured mesh Euler solver runs at about 0.03×10^{-3} s/MG cycle/node. However, the CRAY-2 hardware gather-scatter routines achieve only 25% of the speed exhibited by similar routines on the CRAY-XMP. Thus, both the nodal and cell-centered schemes, which depend heavily on these routines, can be expected to run at least twice as fast on a CRAY-XMP. Furthermore, the connectivity between nodes is about 50% higher on triangular meshes, and thus a larger number of fluxes must be calculated at each time step. On the other hand, this may result in higher accuracy for the triangle codes for a given number of mesh points. When hardware gather-scatter routines are employed, there is no need to store intermediate strings of sorted vectors. Thus, in principle, the present methods require additional storage only for the integer array of pointers, i.e., four integers per mesh edge in the time-stepping calculations. The multigrid preprocessing stage does, however, require additional memory for the tree search.

The unstructured multigrid algorithm would appear to extend readily to three dimensions using tetrahedral meshes. For the cell-centered scheme, the concept of overlapping surfaces translates into intersecting volumes. For the nodal scheme, the residual at a fine grid node should be transferred to the four vertices of the tetrahedral cell enclosing it. Future work will explore these aspects as well as the inclusion of viscous effects.

References

[1]Atta, E. H. and Vadyak, J., "A Grid Interfacial Zonal Algorithm for Three-Dimensional Transonic Flows about Aircraft Configurations," AIAA Paper 82-1017, June 1982.

[2]Danenhoffer, J. F. and Baron, J. R., "Robust Grid Adaptation for Complex Transonic Flows," AIAA Paper 86-0495, Jan. 1986.

[3]Weatherill, N. P., "The Generation of Unstructured Grids Using Dirichlet Tessalations," Princeton Univ. Rept. MAE 1715, July 1985.

[4]Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow over a Complete Aircraft," AIAA Paper 86-0103, Jan. 1986.

[5]Mavriplis, D. J., "Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," Ph.D. Thesis, Mechanical and Aerospace Engineering Dept., Princeton Univ., Princeton, NJ, 1987.

[6]Jameson, A. and Mavriplis, D. J., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh," *AIAA Journal*, Vol. 24, April, 1986, pp. 611–618.

[7]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes," AIAA Paper 81-1259, June 1981.

[8]Mavriplis, D. and Jameson, A., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," AIAA Paper 87-0353, Jan. 1987.

[9]Perez, E., "Finite Element and Multigrid Solution of the Two-Dimensional Euler Equations on a Non-Structured Mesh," Institut National de Recherches en Informatique et Automatique, Rept. No. 442, Sept. 1985.

[10]Lohner, R. and Morgan, K., "Unstructured Multigrid Methods," 2nd European Conf. on Multigrid Methods, Cologne, FRG, Oct. 1985.

[11]Jameson, A., "Solution of the Euler Equations by a Multigrid Method," *Applied Mathematics and Computations*, Vol. 13, No. 3, Nov. 1983, pp. 327–356.

[12]Volpe, G., "A Multigrid Method for Computing the Transonic Flow Over Two Closely-Coupled Airfoil Components," 14th International Conf. of Aeronautical Sciences (ICAS) Congress, Toulouse, France, Sept. 1984.