AIAA JOURNAL
Vol. 37, No. 10, October 1999

Directional Agglomeration Multigrid Techniques for High-Reynolds-Number Viscous Flows

D. J. Mavriplis*
NASA Langley Research Center, Hampton, Virginia 23681-0001

A preconditioned directional-implicit agglomeration multigrid algorithm is developed for solving two- and three-dimensional viscous flows on highly anisotropic unstructured meshes of mixed-element types. The multigrid smoother consists of a preconditioned point- or line-implicit solver that operates on lines constructed in the unstructured mesh using a weighted graph algorithm. Directional coarsening or agglomeration is achieved using a similar weighted graph algorithm. A tight coupling of the line construction and directional agglomeration algorithms enables the use of aggressive coarsening ratios in the multigrid algorithm, which in turn reduces the cost of a multigrid cycle. Convergence rates that are independent of the degree of grid stretching are demonstrated in both two and three dimensions. Further improvement of the three-dimensional convergence rates through a Generalized Minimum Residual (GMRES) technique is also demonstrated.

I. Introduction

THE goal of this work is the development of an efficient solver for compressible steady-state high-Reynolds-number Navier–Stokes flows on unstructured meshes. The overall strategy is based on a multigrid approach. Multigrid methods form the basis of some of the most efficient available solvers for such problems, both on structured and unstructured grids. For inviscid transonic flow problems, multigrid methods can deliver converged solutions in under 100 cycles.[1] However, for high-Reynolds-number Navier–Stokes problems and for flows involving large regions of low velocity fluid, multigrid convergence rates degrade seriously. This degradation is caused partly to the stiffness induced by the highly stretched grids that are required to resolve efficiently the thin boundary layers and wakes that occur at high Reynolds numbers. Additional stiffness is induced in regions of low Mach number flow because of the disparity in eigenvalues corresponding to the acoustic and convective wave speeds, as the Mach number tends to zero.

The construction of an efficient solver requires simultaneous treatment of these effects. Semicoarsening multigrid techniques as well as implicit line solvers can be used effectively on structured grids to relieve the stiffness associated with highly stretched meshes.[2,3] The basic semicoarsening strategy consists of constructing coarser multigrid levels by coarsening the original grid in the coordinate direction normal to the grid stretching, rather than in all directions simultaneously. When conflicting stretching directions exist, multiple coarse grids must be constructed, each generated by a coarsening in a particular coordinate direction.[4] However, when a single stretching direction can be identified, only one family of directionally coarsened grids is required.[5]

Semicoarsening techniques can be generalized to unstructured meshes as directional coarsening methods.[6–9] Graph algorithms can be constructed to remove mesh vertices based on the local degree and direction of anisotropy in either the grid or the discretized equations. This is achieved by basing point-removal decisions on the values of the discrete stencil coefficients, which is the basis for algebraic multigrid methods[9] that operate on sparse matrices directly, rather than on geometric meshes. These techniques are more general than those available for structured meshes because they can deal with multiple regions of anisotropies in conflicting directions.

One of the drawbacks of semi- or directional-coarsening techniques is that they result in coarse grids of higher complexity. Although a full-coarsening approach reduces grid complexity between successively coarser levels by a factor of 4 in two dimensions, and 8 in three dimensions, semicoarsening techniques only achieve a grid complexity reduction of 2, in both two and three dimensions. This increases the cost of a multigrid V-cycle and makes the use of W-cycles impractical. Perhaps more importantly for unstructured mesh calculations, the amount of memory required to store the coarse levels is dramatically increased, particularly in three dimensions.

An alternative to semicoarsening is to use an implicit line solver in the direction normal to the grid stretching coupled with a regular full coarsening multigrid algorithm. Although predetermined grid lines do not exist in an unstructured mesh, such lines can be constructed by identifying and grouping together neighboring mesh edges using a graph algorithm.[10,11] By using a weighted graph algorithm with edge weights that reflect the degree of coupling in the discretization between neighboring grid points, sets of lines that propagate in the direction of strong grid coupling can be constructed.[12]

The solution strategy described in this paper addresses the anisotropy-induced stiffness problem through a combination of implicit line solvers coupled with directional coarsening multigrid. This coupled algorithm permits faster coarsening rates, which result in more optimal coarse grid complexities. The low Mach-number stiffness problem is addressed using preconditioning techniques,[13–16] which are integrated into the overall directional implicit multigrid algorithm. The combination of these three techniques into a single solver has previously been demonstrated in the context of geometric multigrid for two-dimensional problems.[12] The current work represents an extension of this strategy to the more practical agglomeration or algebraic multigrid approach for unstructured meshes, as well as the extension to three dimensions.

II. Discretization

The governing equations are discretized using a finite volume approach. Flow variables are stored at the vertices of the mesh, and control volumes are formed by the median-dual graph of the original mesh, as shown in Fig. 1. A control-volume flux balance is computed by summing fluxes evaluated along the control volume faces, using the average values of the flow variables on either side of the face in the flux computation. This construction of the convective terms corresponds to a central difference scheme, which requires additional dissipation terms for stability. These may either be constructed explicitly as a blend of a Laplacian and biharmonic operator, or may be obtained by writing the residual of a standard upwind scheme as the sum of a convective term and dissipation term:

$$\sum_{k=1}^{\text{neighbors}} \frac{1}{2}[\boldsymbol{F}(w_i) + \boldsymbol{F}(w_k)] \cdot \boldsymbol{n}_{ik} - \frac{1}{2}|\boldsymbol{A}_{ik}|(w_L - w_R) \qquad (1)$$

where the convective fluxes are denoted by $\boldsymbol{F}(w)$, $\boldsymbol{n}_{ik}$ represents the normal vector of the control volume face separating the neighboring

Presented as Paper 98-0612 at the AIAA 36th Aerospace Sciences Meeting, Reno, NV, 12–15 January 1998; received 24 January 1998; revision received 2 February 1999; accepted for publication 4 March 1999.

*Research Fellow, Institute for Computer Applications in Science and Engineering, MS 132C.

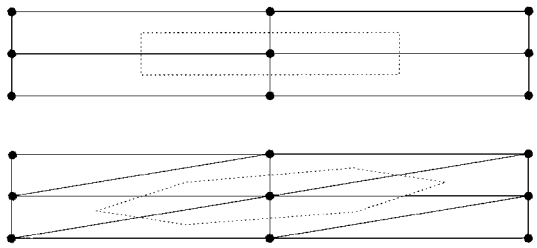

Fig. 1 Median control volumes for stretched quadrilateral and triangular elements.

vertices i and k, and $\boldsymbol{A}_{ik}$ is the flux Jacobian evaluated in the direction normal to this face. w_L and w_R represent extrapolated flow values at the left- and right-hand sides of the control volume face, respectively. For a first-order scheme these are taken as the values at the vertices to the left and right of the control volume interface, whereas for a second-order scheme these are extrapolated from the corresponding vertex values using solution gradients precomputed at these vertices. In this work a matrix artificial dissipation is employed. The matrix-based artificial dissipation scheme is obtained by utilizing the same transformation matrix $|\boldsymbol{A}_{ik}|$ as the upwind scheme, but using this to multiply a difference of blended first and second differences (i.e., blended Laplacian and biharmonic operator) rather than a difference of reconstructed states at control-volume boundaries. The traditional scalar artificial dissipation scheme[17–19] is obtained by replacing the four eigenvalues $u,\ u,\ u+c,\ u-c$ in the $|\boldsymbol{A}_{ik}|$ matrix of the matrix dissipation model by the maximum eigenvalue $|u|+c$, where u and c denote local fluid velocity and speed of sound, respectively. This matrix dissipation construction has been found to deliver accuracy comparable to an upwind scheme, while eliminating the need to compute and store flow gradients at mesh vertices.

The thin-layer form of the Navier–Stokes equations is employed in all cases, and the viscous terms are discretized to second-order accuracy by finite difference approximation. For multigrid calculations a first-order discretization is employed for the convective terms on the coarse grid levels.

The single-equation turbulence model of Spalart and Allmaras[20] is used to account for turbulence effects. This equation is discretized and solved in a manner completely analogous to the flow equations, with the exception that the convective terms are only discretized to first-order accuracy.

This particular discretization is designed to enable the use of mixed element meshes in two dimensions (quadrilaterals and triangles) and three dimensions (tetrahedra, prisms, pyramids, hexahedra). Meshes of differing element types are handled by employing a single edge-based data structure to assemble the fluxes across all element types.[21] In two dimensions quadrilateral elements are employed in the regions of high mesh stretching, whereas triangular elements are employed in isotropic regions of the mesh. In three dimensions hexahedra or prisms are employed in regions near the wall, whereas tetrahedra are generally employed elsewhere. The use of different element types in regions of high mesh stretching enables a more complete decoupling of the discretization in the stretching and normal directions, as discussed in Sec. IV.

III. Preconditioned Smoothing

Once the governing equations are discretized, they must be integrated in time to obtain the steady-state solution. This is achieved using a preconditioned multistage time-stepping scheme. An explicit k-stage scheme can be written as

$$\begin{aligned}
\boldsymbol{w}_i^{(0)} &= \boldsymbol{w}_i^{(n)} \\
&\ \ \vdots \qquad\quad \vdots \\
\boldsymbol{w}_i^{(q)} &= \boldsymbol{w}_i^{(0)} + \Delta t_i \times \boldsymbol{R}_i\left[\boldsymbol{w}^{(q-1)}\right] \\
\boldsymbol{w}_i^{(q+1)} &= \boldsymbol{w}_i^{(0)} + \Delta t_i \times \boldsymbol{R}_i\left[\boldsymbol{w}^{(q)}\right] \\
&\ \ \vdots \qquad\quad \vdots \\
\boldsymbol{w}_i^{(n+1)} &= \boldsymbol{w}_i^{(q=k)}
\end{aligned} \tag{2}$$

where Δt represents the scalar time step estimate and $\boldsymbol{R}_i$ is the discrete residual at vertex i. Whereas such a scheme is commonly used for scalar artificial dissipation discretizations, for upwind or matrix dissipation discretizations substantial increases in efficiency can be obtained by using a Jacobi preconditioning approach in conjunction with the multistage scheme.[22–25] The $(q+1)$th stage of a Jacobi preconditioned multistage scheme can be written as

$$\boldsymbol{w}_i^{(q+1)} = \boldsymbol{w}_i^{(0)} + [\boldsymbol{D}_i]^{-1} \times \boldsymbol{R}_i\left[\boldsymbol{w}^{(q)}\right] \tag{3}$$

where the scalar time step Δt from Eq. (2) is replaced by the matrix time step given by the inverse of the matrix

$$[\boldsymbol{D}_i] = -\frac{\partial \boldsymbol{R}_i(\boldsymbol{w})}{\partial \boldsymbol{w}_i} = \left(\sum_{k=1}^{\text{neighbors}} \frac{1}{2}|\boldsymbol{A}_{ik}|\right) \tag{4}$$

which is a 5×5 matrix (4×4 in two dimensions) corresponding to the pointwise Jacobian of the residual. Note that for a scalar dissipation scheme, this matrix becomes diagonal, and the scalar time-step estimate is recovered, thus reducing the scheme to the standard explicit multistage scheme (in the case of inviscid flow).

Additional preconditioning of the type described in Refs. 13–16 must be implemented to address the stiffness problems induced by regions of low-Mach-number flows. Traditionally, such preconditioners are described as a matrix multiplying an explicit updating scheme and a similar matrix-based modification to the dissipation terms, which improves the accuracy at low Mach numbers. Thus, the $(q+1)$th stage of the standard multistage scheme [cf. Eq. (2)] is rewritten using the preconditioning matrix $\boldsymbol{P}$ as

$$\boldsymbol{w}_i^{(q+1)} = \boldsymbol{w}_i^{(0)} + \boldsymbol{P}\Delta t_i \times \left\{ \sum_{k=1}^{\text{neighbors}} \frac{1}{2}\left[\boldsymbol{F}\left(w_i^q\right) + \boldsymbol{F}\left(w_k^q\right)\right] \cdot \boldsymbol{n}_{ik} - \frac{1}{2}\boldsymbol{P}^{-1}|\boldsymbol{P}\boldsymbol{A}_{ik}|\left(w_L^q - w_R^q\right) \right\} \tag{5}$$

In the present work we wish to implement this type of preconditioner in the context of a point-implicit (Jacobi-preconditioned) or line-implicit scheme. Because the low-Mach-number preconditioning matrix is a point-wise matrix, its implementation for point-implicit schemes is similar as for line-implicit, or any implicit scheme. The approach taken, which was originally described in Refs. 21 and 26, is to modify the dissipation terms in the discretization, as per Eq. (5), and then simply take this modification into account in the point-wise linearization that is required for the point-implicit Jacobi scheme. Thus, the $(q+1)$th stage of the low-Mach-number preconditioned Jacobi multistage scheme becomes

$$\boldsymbol{w}_i^{(q+1)} = \boldsymbol{w}_i^{(0)} + \left[\sum_{k=1}^{\text{neighbors}} \frac{1}{2}\boldsymbol{P}^{-1}|\boldsymbol{P}\boldsymbol{A}_{ik}|\right]^{-1} \times \left\{ \sum_{k=1}^{\text{neighbors}} \frac{1}{2}\left[\boldsymbol{F}\left(w_i^q\right) + \boldsymbol{F}\left(w_k^q\right)\right] \cdot \boldsymbol{n}_{ik} - \frac{1}{2}\boldsymbol{P}^{-1}|\boldsymbol{P}\boldsymbol{A}_{ik}|\left(w_L^q - w_R^q\right) \right\} \tag{6}$$

In regions where the Mach number is relatively large, the low-Mach-number preconditioning matrix $\boldsymbol{P}$ becomes the identity matrix, and the effect of the preconditioner vanishes. In this case the preceding scheme reverts to the Jacobi-preconditioned scheme of Eq. (3). Likewise, for scalar dissipation discretizations (i.e., when $|\boldsymbol{P}\boldsymbol{A}_{ik}|$ is approximated as a diagonal matrix), this scheme reverts to the low Mach-number preconditioned schemes characterized by Eq. (5) and described in Refs. 13, 14, and 16. The particular form of the preconditioning matrix $\boldsymbol{P}$ employed is that described in Ref. 27. The implementation described therein is attractive because it can be achieved without any change of variables in the original discretization.

For simplicity, the inviscid flow equations have been assumed in the preceding discussion. For viscous flows the residual contains additional physical viscous terms that must be taken into account in the linearization. Thus, the matrix to be inverted in Eq. (6) must also be modified to include the linearization of the viscous terms

appearing in the residual, as per Eq. (3) and the first equality of Eq. (4).

Equation (6) represents the scheme used in isotropic regions of the mesh. In regions of large mesh stretching, this pointwise scheme is replaced by a line implicit scheme, operating on grid lines that are preconstructed in the grid. The implicit system generated by the set of lines can be viewed as a simplification of the general Jacobian obtained from a linearization of a backwards Euler-time discretization, where the Jacobian is that obtained from a first-order discretization, assuming a constant Roe matrix in the linearization. For block-diagonal preconditioning all off-diagonal block entries are deleted, whereas in the line-implicit method the block entries corresponding to the edges that constitute the lines are preserved. The line-implicit solver is introduced into the current solution strategy as an extension of the Jacobi preconditioner. At each stage in the multistage scheme, the corrections previously obtained by multiplying the residual vector by the inverted block-diagonal matrix are replaced by corrections obtained by solving the implicit system of block-tridiagonal matrices generated from the set of lines. This implementation has the desirable feature that it reduces exactly to the block-diagonal preconditioned multistage scheme when the line length becomes one (i.e., one vertex and zero edges), as is the case in isotropic regions of the mesh.

In summary, the final scheme, which is used as a smoother for multigrid on all levels, results in a point-implicit low Mach-number preconditioned multistage scheme in isotropic regions of the mesh and a line-implicit low Mach-number preconditioned multistage scheme in regions of high mesh stretching. A three-stage multistage scheme with stage coefficients optimized for high-frequency damping properties,[28] and a Courant–Friedrichs–Lewy number of 1.8 is used in all computations.

IV. Directional Agglomeration and Line Construction

The stiffness caused by grid anisotropy is addressed by a directional agglomeration multigrid strategy coupled with a line-implicit smoother. The combination of these two strategies into a single algorithm has been found to result in a more robust and efficient solution method than the use of either strategy alone.[12,29]

In regions of high grid stretching, standard directional agglomeration (i.e., coarsening) results in the removal of one grid point for every retained coarse grid point. This produces a sequence of coarse grid levels for which the complexity between successive levels decreases by a factor of 2. Isotropic agglomeration, on the other hand, produces a coarse grid complexity reduction of 4:1 in two dimensions and 8:1 in three dimensions. The higher complexity of the directionally coarsened levels greatly increases memory overheads, particularly in three dimensions, and makes the use of the multigrid W-cycle impractical because the operation count of the W-cycle becomes unbounded in such cases as the number of grid levels is increased.

The implicit line solver achieves superior smoothing of error components along the direction of the implicit lines, as compared to a regular explicit scheme. This in turn permits the use of an accelerated coarsening schedule by the agglomeration multigrid algorithm. However, because the implicit line solver is only effective at smoothing error components along the implicit lines, multigrid coarsening must proceed precisely along the direction of these lines. This requires a close coupling between the directional agglomeration algorithm and the line construction algorithm. Both techniques are based on weighted-graph algorithms and must employ the same definition of the graph weights.

Agglomeration multigrid may be viewed as a simplified algebraic multigrid strategy. Coarse-level grids are constructed by fusing together or agglomerating neighboring control volumes to form a coarser set of larger but more complex control volumes. In the algebraic interpretation of agglomeration multigrid, the coarse levels are no longer geometric grids, but represent groupings of fine grid equations that are summed together to form the coarse grid equations sets.[6,30] Therefore, it is important to base the directional agglomeration and line-construction graph weights on algebraic quantities such as stencil coefficients, rather than geometric quantities such as edge lengths, which may be ill-defined on the coarse levels. However, a one-to-one correspondence between stencil coefficients and grid edges only exists for scalar equations and is not possible for systems of equations. For this reason the edge weights for the line-construction algorithm and the directional agglomeration algorithm are taken as the stencil coefficients of a scalar convection equation discretized on the fine grid using the finite volume approach. On the fine level these correspond to the area-weighted normals of the control volume faces delimiting two neighboring vertices. On the coarser levels these are constructed by summing the constituent fine-level face normals.

For highly stretched quadrilateral cells this definition results in large weights being associated with grid edges normal to the direction of stretching and small edge weights in the direction parallel to the stretching, as can be inferred from the relative sizes of the control volume faces in Fig. 1. However, for stretched triangular cells the diagonal grid edges result in weights that may be comparable in the two directions. This weaker decoupling of the normal and stretching directions for triangular elements in two dimensions can produce undesirable results in the line and agglomeration algorithms. Therefore, we employ quadrilateral elements in two dimensions in regions of high mesh stretching and prismatic (or hexahedral) elements in highly stretched regions for three-dimensional meshes. An alternate approach would be to employ a different control volume definition, such as a containment dual-based control volume[31] and retain simple elements in these regions, although this has not been attempted to date.

The line-construction algorithm begins by precomputing the ratio of maximum-to-average adjacent edge weight for each vertex. The vertices are then sorted according to this ratio. The first vertex in this ordered list is then picked as the starting point for a line. The line is built by adding to the original vertex the neighboring vertex, which is most strongly connected to the current vertex, provided this vertex does not already belong to a line and provided the ratio of maximum to minimum edge weights for the current vertex is greater than α (using $\alpha = 4$ in all cases). The line terminates when no additional vertex can be found. If the originating vertex is not a boundary point, then the procedure must be repeated beginning at the original vertex and proceeding with the second strongest connection to this point. When the entire line is completed, a new line is initiated by proceeding to the next available vertex in the ordered list. Ordering of the initial vertex list in this manner ensures that lines originate in regions of maximum anisotropy and terminate in isotropic regions of the mesh. The algorithm results in a set of lines of variable length. In isotropic regions, lines containing only one point are obtained, and the point-implicit or Jacobi preconditioned scheme is recovered.

The agglomeration algorithm consists of choosing a seed point (i.e., a control volume), which initiates a local agglomeration, and then agglomerating the neighboring control volumes to the seed point. The isotropic version of this algorithm[18,32,33] constitutes an unweighted-graph algorithm. In this version of the algorithm, each time a seed point is chosen, all neighboring points are agglomerated to this point. The directional agglomeration algorithm is based on a weighted-graph technique. The edge weights are defined in the same manner as for the line-construction algorithm. Once a seed point is chosen, only those neighboring points that are connected to the seed point through an edge of weight greater than $\beta \times \max_{\text{weight}}$ are agglomerated, where $\max_{\text{weight}}$ denotes the maximum edge weight incident to the seed point. Taking $\beta = 0.5$ reproduces the isotropic agglomeration algorithm in regions were all edge weights are close in size. However, in regions where one edge weight is much larger than the others, a directional coarsening is achieved. This results in a 2:1 coarsening ratio in such regions. To obtain a 4:1 coarsening ratio, the process must be repeated. This will result in the agglomeration of points or control volumes, which were not originally neighbors of the initial seed point. This type of aggressive coarsening can only be tolerated in regions where the implicit line solver is used as a smoother. Therefore, the coarsening process is repeated only if the agglomerated control volume is joined to the current seed point by an edge that is part of an implicit line. The process is repeated until four control volumes are agglomerated together, or until no line edges can be found.

From the preceding description one can see that the line construction and coarsening process are closely coupled and must be carried out simultaneously. The edge weights, once defined on the finest

level, are computed on the fly for each coarser level as they are created. The whole process is performed in a preprocessing phase, and the output, consisting of sets of lines for each level and coarse grid groupings, is passed to the flow solver. As an example, the directional implicit agglomeration multigrid algorithm has been applied to the grid of Fig. 2. The lines created on the finest grid level are depicted in Fig. 3. The first coarse agglomerated level is illustrated in Fig. 4, depicting the agglomerated cells in the boundary-layer region near the leading edge, where a 4:1 directional coarsening is observed. Table 1 documents the complexity of the coarse grid levels using the isotropic agglomeration algorithm of Ref. 18, as well as the coarse grid complexity achieved using the current directional agglomeration multigrid algorithm. The resulting complexity for a multigrid W-cycle is just 15% larger for the directionally agglomerated grids than for the isotropically agglomerated grids.

Table 1 Comparison of coarse grid complexity and resulting W-cycle complexity for regular isotropic agglomeration and directional agglomeration multigrid (AMG)

	Regular AMG		Directional AMG	
Mesh level	Node	Ratio	Node	Ratio
1	16,167	1.0	16,167	1.0
2	4,074	3.99	4,476	3.61
3	1,038	3.92	1,383	3.24
4	268	3.87	585	2.36
	W-cycle complexity			
	1.89		2.18	

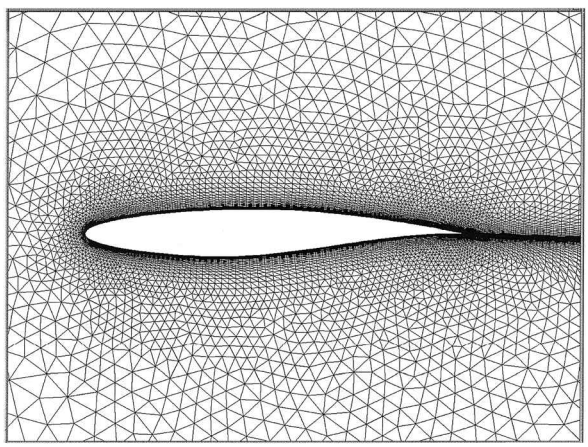

Fig. 2 Unstructured grid used for computation of transonic flow over RAE 2822 airfoil (number of points = 16167, wall resolution = 10^{-6} chords).

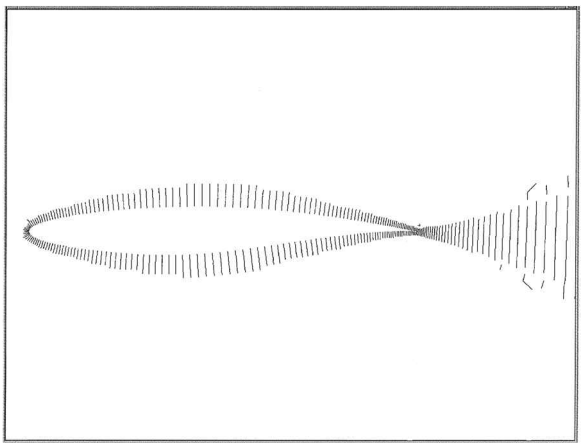

Fig. 3 Directional-implicit lines constructed on grid of Fig. 2 by weighted-graph algorithm.

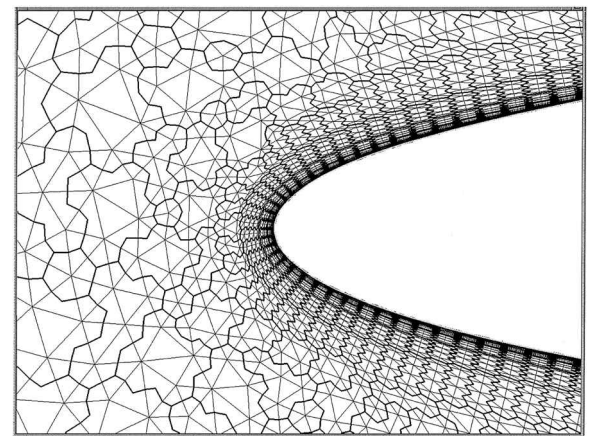

Fig. 4 First agglomerated multigrid level constructed on grid of Fig. 2 illustrating 4:1 directional coarsening in boundary-layer region.

V. Two-Dimensional Results

The combined directional-implicit agglomeration multigrid algorithm produces convergence rates independent of the degree of grid anisotropy, which is demonstrated in two dimensions by solving the transonic flow over a Royal Aircraft Establishment (RAE) 2822 airfoil on three different grids. All three grids contain the same distribution of boundary points, but different resolutions in the direction normal to the boundary and wake regions. The first grid contains a normal wall spacing of 10^{-5} chords and a total of 12,568 points; the second grid contains a normal wall spacing of 10^{-6} chords and 16,167 points; and the third grid contains a normal wall spacing of 10^{-7} chords and 19,784 points. The cells in the boundary layer and wake regions are generated using a geometric progression of 1.2 for all three grids. The second grid, depicted in Fig. 2, contains what is generally regarded as a suitable normal and streamwise resolution for accurate computation of this type of problem, whereas the first and third grids are most likely underresolved and overresolved in the direction normal to the boundary layer, respectively. The Mach number for this case is 0.73, the incidence is 2.31 deg, and the Reynolds number is 6.5×10^6. To isolate the effects of the turbulence model, the turbulence values have been frozen after 150 to 200 cycles in these computations. The flow is transonic, and the low-Mach-number preconditioning matrix reverts to the identity matrix for this case. With the effect of this preconditioning removed, a more direct comparison between the directional implicit multigrid and the previously developed unpreconditioned full-coarsening multigrid method[18,19] is possible. The convergence rates of both methods on all three grids are shown in Fig. 5. The explicit full-coarsening multigrid solver produces convergence rates that decay substantially as the grid stretching is increased. In fact, the asymptotic rate of this scheme for the most highly stretched grid is almost two orders of magnitude slower than that achieved on the least stretched grid. On the other hand, the directional-implicit agglomeration scheme produces convergence to machine zero in under 600 cycles and is essentially unaffected by the degree of grid anisotropy. This comparison represents the best possible performance for each scheme. The explicit full-coarsening multigrid algorithm employs a five-stage time-stepping scheme that is augmented with implicit residual smoothing and is used to solve a scalar dissipation discretization. The use of more accurate matrix dissipation with the explicit full-coarsening multigrid algorithm produces slower and less robust convergence rates. The directional-implicit agglomeration algorithm operates on the matrix dissipation discretization and uses a three-stage time-stepping scheme with no residual smoothing but with point or line preconditioning where the Jacobians are evaluated and inverted only at the first stage of the scheme and then frozen for the remaining stages. Although Fig. 5 compares the two schemes in terms of multigrid cycles, the cost per cycle of both schemes is relatively close, the directional implicit agglomeration scheme being about 15% more expensive per cycle, which is mainly because of the added work for the evaluation of the matrix dissipation.

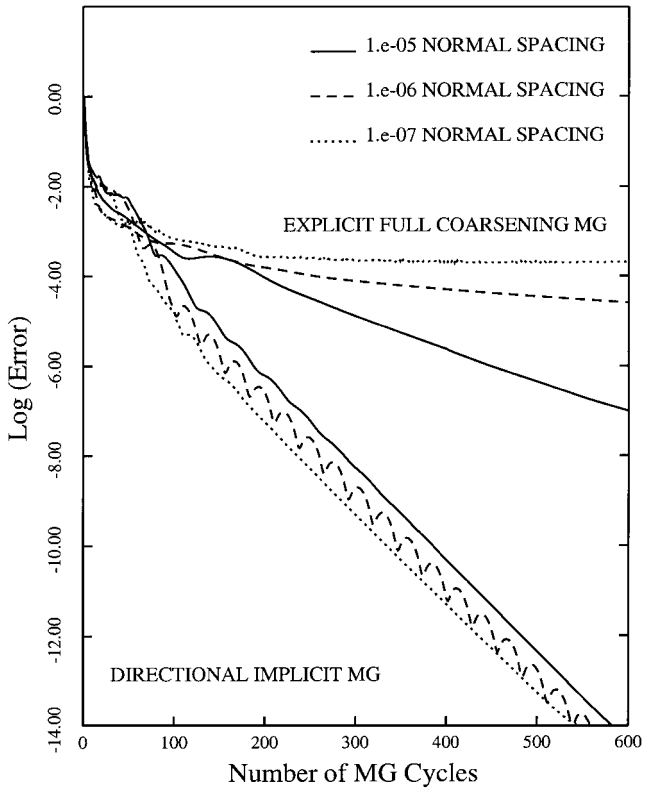

Fig. 5 Comparison of explicit isotropic and directional-implicit agglomeration multigrid algorithm convergence rates on three grids of varying normal resolution.

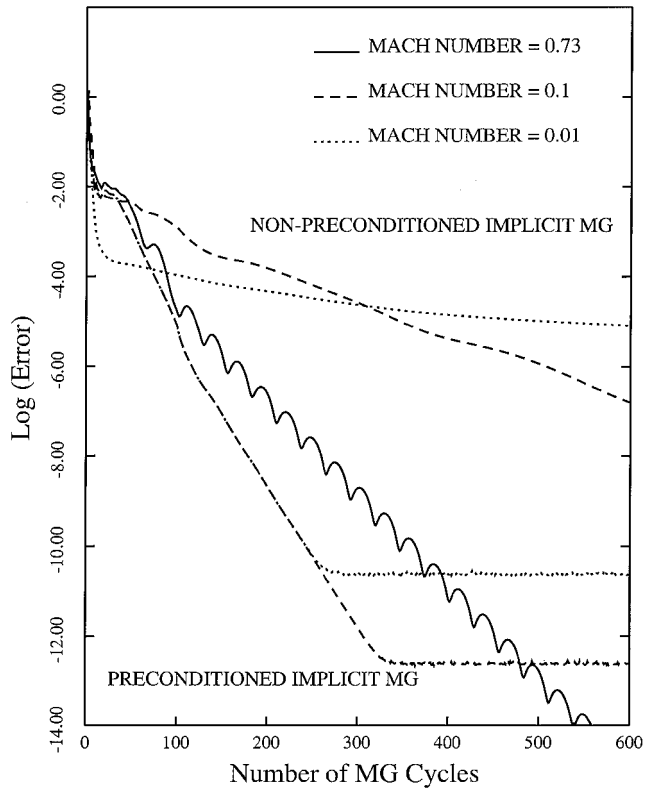

Fig. 6 Comparison of low mach-number preconditioned and unpreconditioned directional-implicit agglomeration multigrid algorithm convergence rates for various freestream Mach numbers.

The benefits of low-Mach-number preconditioning are demonstrated in Fig. 6, where the flow over an RAE 2822 airfoil at varying Mach numbers has been computed on the grid of Fig. 2 using the directional-implicit agglomeration algorithm with and without the low-Mach-number preconditioner. For the transonic case the preconditioner is not active, and both cases give identical convergence. However, as the Mach number is lowered, the convergence rate degrades substantially for the cases run with no preconditioning, whereas the preconditioned cases all converge to machine zero in approximately 300 cycles. This example demonstrates the importance of employing both techniques simultaneously (low-Mach-number preconditioning and directional-implicit agglomeration) in order to obtain rapid convergence rates for subsonic Navier–Stokes flows.

The computation of high-lift flows simultaneously involves regions of low velocity fluid and high-grid anisotropy, therefore providing a good demonstration of the current algorithm. Figure 7 depicts an unstructured grid about a three-element airfoil high-lift configuration. The grid contains a total of 61,104 points and a normal spacing of 10^{-6} chords at the surface of each airfoil element. The implicit lines generated by the graph algorithm for this case are depicted in Fig. 8, and a qualitative view of the solution as a set of Mach contours is given in Fig. 9. The freestream Mach number for this case is 0.2, the incidence is 16 deg, and the Reynolds number is 9×10^6. The convergence rates of the directional-implicit agglomeration scheme and the explicit full-coarsening agglomeration scheme are compared on the basis of CPU time in Fig. 10. The explicit full-coarsening scheme employs a five-stage time-stepping scheme and residual smoothing and solves the scalar dissipation discretization, whereas the directional-implicit agglomeration scheme employs the preconditioned three-stage time-stepping scheme with Jacobians frozen at the second and third stages and solves the matrix dissipation discretization. To isolate the effects of the turbulence model, the turbulence values have been frozen after 150 to 200 cycles. As can be inferred from Fig. 10, the directional-implicit agglomeration scheme achieves a six-order-of-magnitude residual reduction in approximately one-quarter of the time required by the explicit full-coarsening approach, which permits engineering solutions to be obtained in approximately 1.5 h on an inexpensive workstation.

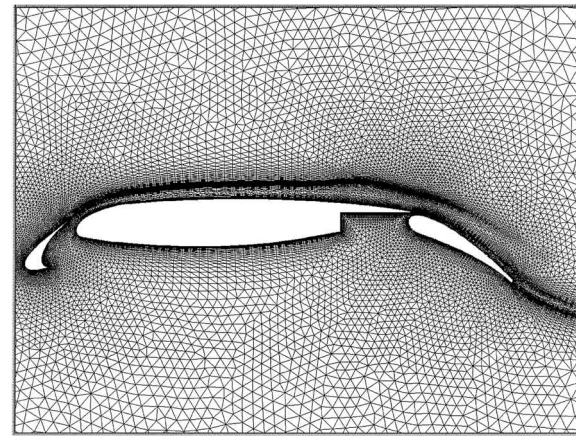

Fig. 7 Unstructured grid used for computation of subsonic flow over three-element airfoil geometry (number of points = 61,104, wall resolution = 10^{-6} chords).

VI. Three-Dimensional Results

In three dimensions, a directional coarsening ratio of 8:1 is required to match the coarse grid complexities achieved by an isotropic full-coarsening algorithm. Unfortunately, robustness problems associated with 8:1 coarsening ratios have been encountered. Therefore, at present a 4:1 coarsening ratio is employed in three dimensions, although faster coarsening ratios are still under investigation. This results in a 50% increase in storage and CPU time per multigrid W-cycle as compared to that achieved by an 8:1 coarsening algorithm, but nevertheless results in an efficient solution procedure for three-dimensional problems. To isolate the effects of the turbulence model, the turbulence values have been frozen after 200 cyles for all three-dimensional computations. The first three-dimensional test case involves the computation of a two-dimensional flow using

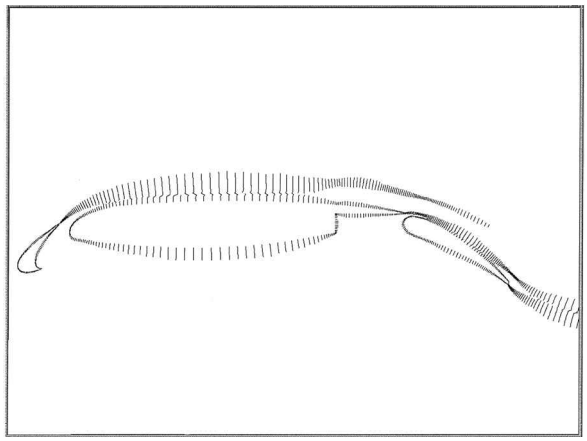

Fig. 8 Implicit lines generated by weighted-graph algorithm on grid of Fig. 7.

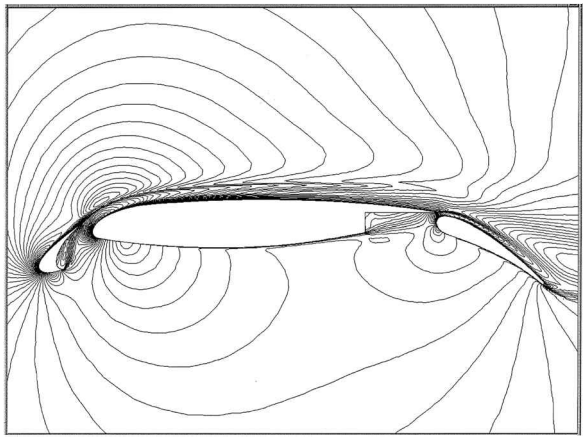

Fig. 9 Computed Mach contours for flow over three-element airfoil (Mach = 0.2, incidence = 16 deg, Reynolds number = 9×10^6).

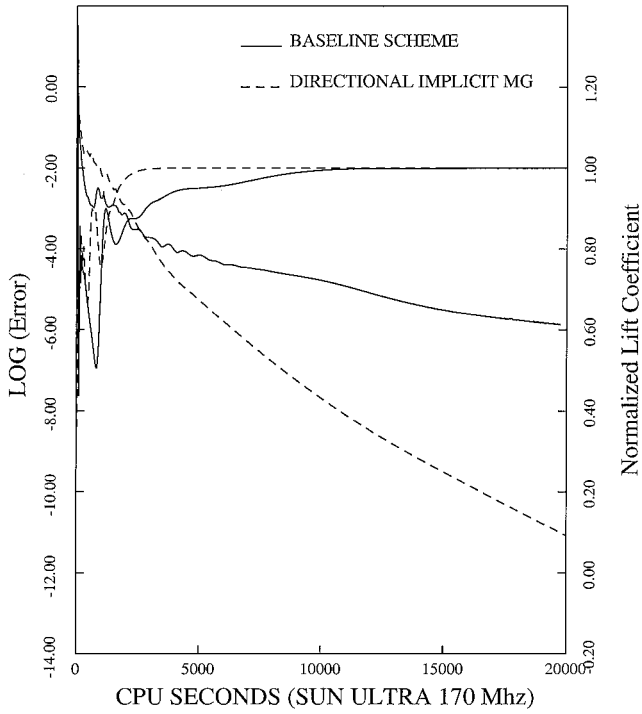

Fig. 10 Comparison of convergence rates obtained for flow over three-element airfoil in terms of CPU time on workstation.

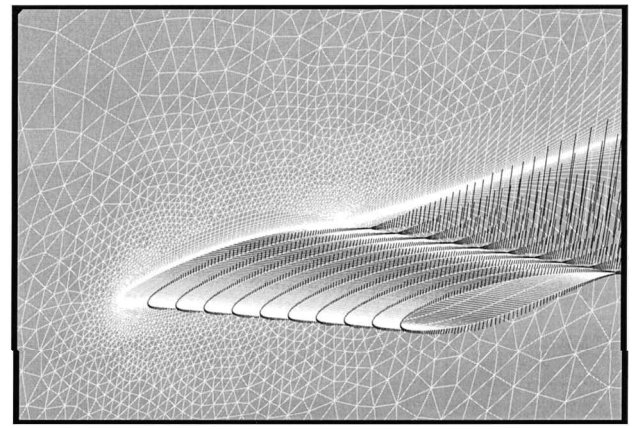

Fig. 11 Illustration of mixed element grid and implicit lines used for computation of two-dimensional flow over three-dimensional wing geometry (number of grid points = 177,837).

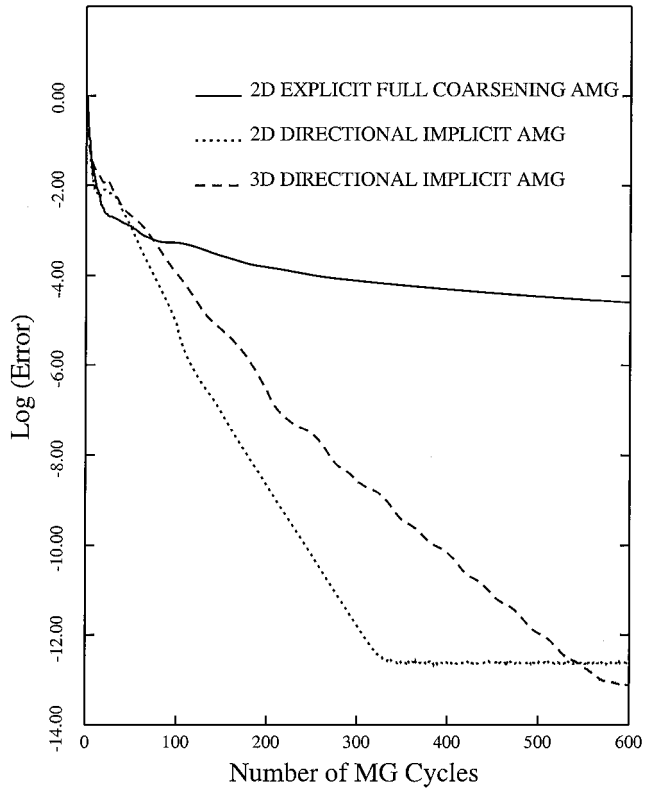

Fig. 12 Comparison of convergence rate obtained by three-dimensional directional-implicit multigrid algorithm with rate obtained on equivalent two-dimensional problem.

a three-dimensional grid to compare the performance of the three-dimensional code with that of the two-dimensional code. The grid of Fig. 2 has been extruded in the spanwise direction, resulting in a three-dimensional grid of 177,837 vertices. Whereas the original two-dimensional grid contained quadrilateral elements in the boundary and wake regions and triangular elements elsewhere, the three-dimensional grid contains hexahedral elements in the viscous regions and prismatic elements in the inviscid regions. The surface grid and the implicit lines generated by the three-dimensional graph algorithm are depicted in Fig. 11. The prescribed coarsening ratio of 4:1 results in coarse levels that are very similar to those produced by the two-dimensional algorithm, at least near the wing surface. The convergence rates of the two- and three-dimensional codes are compared in Fig. 12. The Mach number is 0.1, the incidence 2.31 deg, and the Reynolds number is 6.5×10^6. The three-dimensional convergence curve is much faster than the convergence of the isotropic

algorithm, reaching machine zero in just under 600 multigrid cycles. However, it is somewhat slower than the equivalent two-dimensional algorithm, which reaches machine zero in just 300 iterations.

The next example demonstrates the insensitivity of the current three-dimensional algorithm to grid stretching. Three unstructured tetrahedral grids have been constructed about an ONERA M6 wing using the VGRID grid-generation package.[34] These grids all contain the same surface resolution, but different normal resolutions near the wing surface. The first grid contains a normal wall spacing of 10^{-5} chords and a total of 1.2×10^6 points; the second grid contains a normal wall spacing of 10^{-6} chords and 1.6×10^6 points; and the third grid contains a normal wall spacing of 10^{-7} chords and 2×10^6 points. The cells in the boundary layer and wake regions are generated using a geometric progression of 1.2 for all three grids. As already explained, prismatic elements are required in the boundary-layer regions for the directional-implicit agglomeration algorithm. Because the VGRID grid-generation package produces fully tetrahedral meshes, a mesh-merging technique is employed to merge tetrahedra triplets into prisms in regions near the wall.[21] At this initial stage of development, a uniform height of prism layers is employed over the entire wing surface. This avoids the use of hanging edges or transitional elements such as pyramids when a variable number of prismatic layers are allowed in the grid. On the other hand, this restriction may result in the incomplete merging of some stretched tetrahedral elements. The convergence rates of the directional-implicit agglomeration algorithm on all three grids are depicted in Fig. 13. A four level W-cycle was used for these computations. The coarsening ratios achieved between the first and second, second and third, and third and fourth levels were 3.69:1, 3.16:1, and 2.2:1, respectively, for the 10^{-6} spacing grid, with similar coarsening ratios obtained on the other grids. The Mach number is 0.1, the incidence is 2.0 deg, and the Reynolds number is 3×10^6.

Very similar convergence rates are achieved on all three grids. All cases exhibit a slowdown in convergence after approximately 6 or 7 orders of magnitude, but achieve close to an eight-order-of-magnitude reduction in 600 cycles. Considering that these three grids represent a two-order-of-magnitude variation in the degree

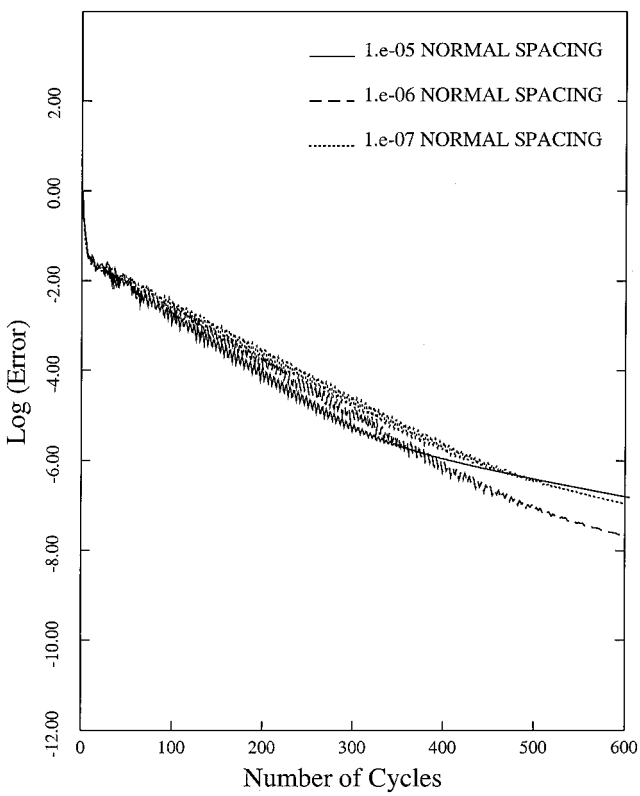

Fig. 13 Comparison of convergence rates achieved by directional-implicit agglomeration multigrid algorithm on three grids of varying normal resolution for ONERA M6 wing geometry at Mach number = 0.1.

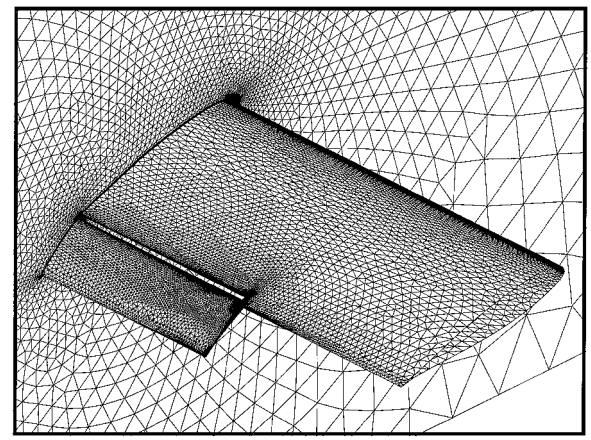

Fig. 14 Illustration of unstructured grid employed for computation of flow over partial-span flap geometry (number of vertices = 549,176).

of grid stretching, the convergence rates can be qualified as independent of the grid stretching. As an example of the computational overheads incurred, the grid containing 1.6×10^6 vertices required a total of 350 Mwords of memory and 53 CPU h to converge 600 cycles. This case was run on 8 CPU of the Cray C90 and achieved a CPU-to-wall-clock time ratio of seven on a lightly loaded machine.

The final case consists of a three-dimensional high-lift application. The geometry involves a partial span flap unswept wing in a wind tunnel. The unstructured grid employed for this case is depicted in Fig. 14. This mesh contains a total of 549,176 points and a normal spacing at the wing surface of 10^{-5} chords. As in the preceding case, a uniform height of prismatic layers was created in the boundary-layer regions using the mesh-merging algorithm of Ref. 21. The Mach number for this case is 0.2, the incidence is 10 deg, and the Reynolds number is 3.7×10^6. A four-level W-cycle was used for this computation. The coarsening ratios achieved between the first and second, second and third, and third and fourth levels were 3.84:1, 3.43:1, and 2.23:1, respectively. The convergence obtained by the directional-implicit agglomeration multigrid algorithm is compared with that achieved by the explicit full-coarsening agglomeration multigrid algorithm in Fig. 15. As in the two-dimensional comparisons, this represents the best possible performance for each algorithm: the directional algorithm employs a three-stage time-stepping scheme and operates on the matrix dissipation discretization, whereas the isotropic algorithm employs a five-stage scheme with residual smoothing and operates on the scalar dissipation discretization. The directional algorithm produces substantially faster convergence than the isotropic algorithm, although a slowdown is observed after 4 to 5 orders of magnitude. Although the asymptotic rate of the directional algorithm is still substantially faster than that of the isotropic algorithm, the rate is much slower than that achieved in two dimensions.

Further increases in the convergence rate can be achieved by resorting to a Krylov acceleration technique such as Generalized Minimum Residual (GMRES).[35] The preconditioned directional-implicit agglomeration algorithm can be employed as a preconditioner to GMRES.[12,24] The current implementation uses a nonlinear GMRES solver[36] that computes Jacobian-vector products by finite differencing the residual. The addition of GMRES incurs little extra CPU time, measured on a multigrid cycle basis, but requires considerable additional storage because a solution vector must be stored for each of the Krylov search directions. In the current implementation 20 search directions are employed, resulting in a memory increase of 100 words per vertex (about 50% increase). The convergence rate using GMRES is depicted in Fig. 15. The solver was run initially 150 multigrid cycles using the directional agglomeration multigrid algorithm alone and then another 462 multigrid cycles using the preconditioned GMRES approach [i.e., 22 GMRES(20) cycles]. The addition of GMRES is largely successful in overcoming the slowdown in the asymptotic convergence rate observed by the simpler directional implicit multigrid algorithm alone, achieving an overall

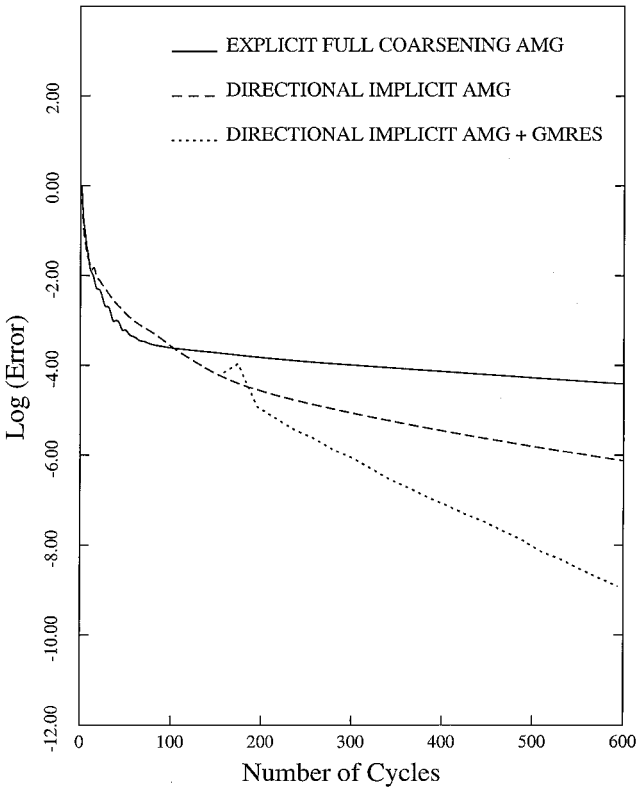

Fig. 15 Comparison of convergence rates achieved by various multigrid schemes for flow over partial span flap geometry.

residual reduction of 9 orders of magnitude over 600 cycles. This case required a total of 230 Mwords of memory and 20 CPU h on the Cray C90 and ran at a CPU-time-to-wall-clock ratio of 7 on eight processors.

VII. Conclusion

The preconditioned directional agglomeration multigrid algorithm has been shown to produce grid-aspect-ratio independent convergence rates in both two and three dimensions. In particular, the two-dimensional implementation of the current algorithm has resulted in a very efficient solver for viscous flows. However the convergence rates obtained in three dimensions, although substantially faster than those achieved by the isotropic algorithm, are still slower than those observed in two dimensions, which may be because of the possibility of the existence of multidimensional stretching in three dimensions. In such cases a modified line-construction and coarsening strategy may be required. Furthermore, the coarsening ratios achieved in three dimensions, which are often even lower that the prescribed 4:1 rate, result in additional CPU time per multigrid cycle and increase the overall memory requirements of the solver. Future work will focus on augmenting the three-dimensional coarsening ratios to approximate the 8:1 ratio observed in isotropic cases as closely as possible and on accelerating the three-dimensional convergence rates through improved line construction and preconditioning. The efficient convergence of the combined system of flow and turbulence equations is also under investigation.

Acknowledgments

This work was made possible in large part because of the computational resources provided by the Numerical Aerospace Simulation facility. The author wishes to thank S. Pirzadeh for his time spent in generating the ONERA M6 wing and partial span flap unstructured meshes.

References

[1]Jameson, A., "Artificial Diffusion, Upwind Biasing, Limiters and Their Effect on Accuracy and Multigrid Convergence in Transonic and Hypersonic Flow," AIAA Paper 93-3359, July 1993.

[2]Brandt, A., "Multigrid Techniques with Applications to Fluid Dynamics: 1984 Guide," *VKI Lecture Series*, von Kármán Inst., Rhode Saint Genese, Belgium, 1984, pp. 1–176.

[3]Allmaras, S. R., "Analysis of Semi-Implicit Preconditioners for Multigrid Solution of the 2D Compressible Navier–Stokes Equations," AIAA Paper 95-1651, June 1995.

[4]Mulder, W. A., "A High Resolution Euler Solver Based on Multigrid Semi-Coarsening and Defect Correction," *Journal of Computational Physics*, Vol. 100, No. 1, 1992, pp. 91–104.

[5]Pierce, N., Giles, M., Jameson, A., and Martinelli, L., "Accelerating Three-Dimensional Navier–Stokes Calculations," AIAA Paper 97-1953, June 1997.

[6]Mavriplis, D. J., "Multigrid Techniques for Unstructured Meshes," *VKI Lecture Series*, von Kármán Inst., Rhode Saint Genese, Belgium, 1995.

[7]Raw, M., "Robustness of Coupled Algebraic Multigrid for the Navier–Stokes Equations," AIAA Paper 96-0297, Jan. 1996.

[8]Francescatto, J., "Résolution de l'Équation de Poisson sur des Maillages Étirés par Une Methode Multigrille," Institut National de Recherche en Informatique et Automatique, INRIA Rept. 2712, Sophi-Antipolis, France, Nov. 1995.

[9]Ruge, J. W., and Stüben, K., "Algebraic Multigrid," *Multigrid Methods*, edited by S. F. McCormick, SIAM Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1987, pp. 73–131.

[10]Hassan, O., Morgan, K., and Peraire, J., "An Adaptive Implicit/Explicit Finite Element Scheme for Compressible High Speed Flows," AIAA Paper 89-0363, Jan. 1989.

[11]Martin, D., and R. Löhner, "An Implicit Linelet-Based Solver for Incompressible Flows," AIAA Paper 92-0668, Jan. 1992.

[12]Mavriplis, D. J., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," AIAA Paper 97-1952, June 1997.

[13]Turkel, E., "Preconditioning Methods for Solving the Incompressible and Low Speed Compressible Equations," *Journal of Computational Physics*, Vol. 72, No. 2, 1987, pp. 277–298.

[14]van Leer, B., Lee, W. T., and Roe, P., "Characteristic Time-Stepping or Local Preconditioning of the Euler Equations," AIAA Paper 91-1552, June 1991.

[15]Choi, Y., and Merkle, C., "The Application of Preconditioning to Viscous Flows," *Journal of Computational Physics*, Vol. 105, No. 2, 1993, pp. 207–223.

[16]Weiss, J. M., and Smith, W. A., "Preconditioning Applied to Variable and Constant Density Time-Accurate Flows on Unstructured Meshes," AIAA Paper 94-2209, June 1994.

[17]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.

[18]Mavriplis, D. J., and Venkatakrishnan, V., "Agglomeration Multigrid for Two Dimensional Viscous Flows," *Computers and Fluids*, Vol. 24, No. 5, 1995, pp. 553–570.

[19]Mavriplis, D. J., and Venkatakrishnan, V., "A 3D Agglomeration Multigrid Solver for the Reynolds-Averaged Navier–Stokes Equations on Unstructured Meshes," *International Journal for Numerical Methods in Fluids*, Vol. 23, No. 6, 1996, pp. 527–544.

[20]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aérospatiale*, Vol. 1, 1994, pp. 5–21.

[21]Mavriplis, D. J., and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier–Stokes Equations on Mixed Element Meshes," *International Journal for Computational Fluid Dynamics*, Vol. 8, 1997, pp. 247–263.

[22]Riemslagh, K., and Dick, E., "A Multigrid Method for Steady Euler Equations on Unstructured Adaptive Grids," *6th Copper Mountain Conference on Multigrid Methods*, NASA CP 3224, 1993, pp. 527–542.

[23]Morano, E., and Dervieux, A., "Looking for O(N) Navier–Stokes Solutions on Non-Structured Meshes," *6th Copper Mountain Conference on Multigrid Methods*, NASA CP 3224, 1993, pp. 449–464.

[24]Ollivier-Gooch, C., "Towards Problem-Independent Multigrid Convergence Rates for Unstructured Mesh Methods I: Inviscid and Laminar Viscous Flows," *Proceedings of the 6th International Symposium on CFD*, Lake Tahoe, NV, 1995, pp. 913–930.

[25]Pierce, N., and Giles, M., "Preconditioning on Stretched Meshes," AIAA Paper 96-0889, Jan. 1996.

[26]Turkel, E., "Preconditioning-Squared Methods for Multidimensional Aerodynamics," AIAA Paper 97-2025, June 1997.

[27]Jespersen, D., Pulliam, T., and Buning, P., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.

[28]van Leer, B., Tai, C. H., and Powell, K. G., "Design of Optimally Smoothing Multi-Stage Schemes for the Euler Equations," AIAA Paper 89-1933, June 1989.

[29]Mavriplis, D. J., "Directional Coarsening and Smoothing Multigrid Strategies for Anisotropic Navier–Stokes Problems," *8th Copper Mountain Conference on Multigrid Methods*, Electronic Transactions on Numerical Analysis (ETNA), Vol. 6, April 1997, pp. 182–197 (available at http://etna.mcs.kent.edu).

[30]Hutchinson, B. R., and Raithby, G. D., "A Multigrid Method Based on the Additive Correction Strategy," *Numerical Heat Transfer*, Vol. 9, No. 5, 1986, pp. 511–537.

[31]Barth, T. J., and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," AIAA Paper 95-0221, Jan. 1995.

[32]Lallemand, M., Steve, H., and Dervieux, A., "Unstructured Multigridding by Volume Agglomeration: Current Status," *Computers and Fluids*, Vol. 21, No. 3, 1992, pp. 397–433.

[33]Smith, W. A., "Multigrid Solution of Transonic Flow on Unstructured Grids," *Recent Advances and Applications in Computational Fluid Dynamics, Proceedings of the American Society of Mechanical Engineers Winter Annual Meeting*, edited by O. Baysal, American Society of Mechanical Engineers, New York, 1990.

[34]Pirzadeh, S., "Viscous Unstructured Three-Dimensional Grids by the Advancing-Layers Method," AIAA Paper 94-0417, Jan. 1994.

[35]Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[36]Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamic Codes," AIAA Paper 85-1494, July 1985.

J. Kallinderis
Associate Editor