Annu. Rev. Fluid. Mech. 1997. 29:473–514

UNSTRUCTURED GRID TECHNIQUES

D. J. Mavriplis
Institute for Computer Applications in Science and Engineering NASA Langley Research Center, Hampton, VA 23681-0001

KEY WORDS: computational, fluid, Navier-Stokes, aerodynamic, mesh, multigrid

ABSTRACT

An overview of the current state of the art in unstructured mesh techniques for computational fluid dynamics is given. The topics of mesh generation and adaptation, spatial discretization, and solution techniques for steady flows are covered. Remaining difficulties in these areas are highlighted, and directions for future work are outlined.

1. INTRODUCTION

The field of computational fluid dynamics has been evolving rapidly over the past several decades. Aerospace applications, particularly in the field of aerodynamics, have provided the major impetus for the advances in computational fluid dynamics technology. Great strides in available computational power, coupled with rapid advances in algorithmic efficiency, have enabled computational fluid dynamics to substantially reduce the amount of wind-tunnel time required for airframe and propulsion system design.

At the same time, computational methods are continuously being required to deliver more accurate solutions for more complex realistic configurations at lower computational cost. Traditionally, structured or block-structured methods that rely on regular arrays of quadrilateral or hexahedral cells in two or three dimensions, respectively, have been employed to discretize the computational domain. Unstructured grid methods originally emerged as a viable alternative to the structured or block-structured grid techniques for discretizing complex geometries. These methods make use either of collections of simplicial elements (triangles in two dimensions, tetrahedra in three dimensions) or

0066-4189/97/0115-0473$08.00

of elements of mixed type with irregular connectivity. This not only provides greater flexibility for discretizing complex domains but also enables straightforward implementation of adaptive meshing techniques where mesh points may be added, deleted, or moved about, while mesh connectivity is updated locally, in order to enhance solution accuracy.

The advent of powerful desktop computers has greatly expanded the applicability of computational fluid dynamics outside the realm of pure research and aeronautics applications, into all areas of fluid mechanics, as well as into multidisciplinary fields. The generality of unstructured grid methods and their ability to enhance solution accuracy through adaptive procedures have proved to be such a great advantage for these diverse applications that many if not most commercial computational fluid dynamics codes currently rely on the use of unstructured meshes.

Despite their recent success, unstructured mesh techniques still incur substantially more overhead than structured grid methods, a fact that has limited their use particularly for large three-dimensional viscous flow cases. The development of efficient and robust unstructured mesh algorithms for grid generation and flow solution represents a considerable ongoing challenge. This paper provides an overview of the current state-of-the-art of such techniques; it points out the advantages and deficiencies of various approaches as well as promising avenues for future improvements.

In section 2 an overview of the most common grid generation and adaptivity techniques is given. Section 3 describes the various approaches to spatial discretization for the Euler and Navier-Stokes equations, and section 4 describes techniques for integrating the discretized equations to steady-state. Owing to space limitations, the additional but equally important topics of unsteady solution techniques and parallel implementations are not covered. Survey papers on these items can be found in the literature (see Special course 1992; Venkatakrishnan 1995 for example).

2. MESH GENERATION

The initial phase of any numerical simulation begins with the generation of a suitable mesh. The basic premise is that, owing to the random way elements may be assembled to fill computational space, unstructured mesh generation techniques are inherently more automatic and amenable to complex geometries than traditional structured or block-structured grids. Therefore, the problem of unstructured mesh generation is largely one of designing algorithms that are automatic, robust, and yield suitable element shapes and distributions for the flow solver.

In the following section, we examine the most prevalent techniques for generating unstructured meshes: the advancing-front method, and Delaunay-based approaches. While the advancing-front method is somewhat heuristic in nature, Delaunay-based methods are firmly rooted in computational geometry principles. In practice, both methods have resulted in successful three-dimensional grid generation tools for arbitrary geometries. All of these techniques have also demonstrated robustness problems and grid quality issues that have led to the development of hybrid methods, which attempt to capitalize on the strengths of the various methods.

2.1 *Advancing-Front Techniques*

In the advancing-front technique (Lo 1985, Peraire et al 1987, Gumbert et al 1989) an unstructured mesh is generated by adding individual elements one at a time to an existing front of generated elements. Generation of a two-dimensional grid begins with a discretization of the geometry boundaries as a set of edges. These edges form the initial front that is to be advanced out into the field. A particular edge of this front is selected, and a new triangle is formed with this edge as its base by joining the two ends of the current edge either to a newly created point or to an existing point on the front. The current edge is then removed from the front, since it is now obscured by the new triangle. Similarly, the remaining two edges of the new triangle are either assigned to the front or removed from the front, depending on their visibility, as shown in Figure 1.

The front thus constitutes a stack (or priority queue), and edges are continuously added to or removed from the stack. The process terminates when

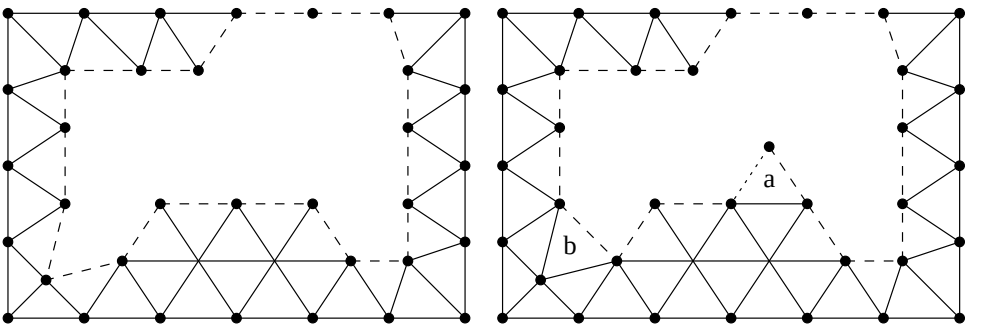

Figure 1 Generation of a new triangle using new point (*a*), or existing front point (*b*), in the advancing-front method. Dashed line edges denote current front.

the stack is empty, i.e. when all fronts have merged upon each other and the domain is entirely covered. One of the critical features of such methods is the placement of new points. Upon generating a new triangle, a new point is first placed at a position that is determined to result in a triangle of optimal size and shape. The parameters that define this optimum triangle as a function of field position are obtained by a prescribed field function. (This field function is often user defined and is evaluated by interpolating from values stored on a coarse background grid.) The triangle generated with this new point may result in a cross-over with other front edges, and may therefore be rejected. This is determined by computing possible intersections with "nearby" front edges. Alternately, an existing point on the front may coincidentally be located very close to the new point, and thus should be employed as the forming point for the new triangle, to avoid the appearance of a triangle with a very small edge at some later stage. Existing candidate points are thus also searched by locating all "nearby" front points.

For three-dimensional grid generation, a surface grid is first constructed by generating a two-dimensional triangular mesh on the surface boundaries of the domain. This triangular mesh forms the initial front, which is then advanced into the flow-field by placing new points ahead of the front and forming tetrahedral elements. The required intersection checking now involves triangular front faces rather than edges as in the two-dimensional case.

Advancing-front methods generally result in smooth high-quality triangulations in most regions of the flow-field. However, difficulties may be encountered in regions where the fronts must be merged. The algorithm is somewhat heuristic in nature, and the need to perform intersection checking operations on the moving front requires the use of dynamic quad/octree data structures for efficient implementations.

2.2 *Delaunay Triangulation Methods*

Given a set of points in the plane, there exist many possible triangulations of these points. A Delaunay construction represents a unique triangulation of these points that exhibits a large class of well-defined properties (Preparata & Shamos 1985). The empty circumcircle property forms the basis of the Bowyer-Watson algorithm (Bowyer 1981, Watson 1981). This property states that no triangle in a Delaunay triangulation can contain a point other than its three forming vertices within its circumcircle. Thus, given an initial triangulation, a new point may be inserted into the triangulation by first locating and deleting all existing triangles whose circumcircles contain the newly inserted point. A new triangulation is then formed by joining the new point to all boundary vertices of the cavity created by the previous removal of intersected triangles, as shown in Figure 2. Point-insertion algorithms can be employed as the basis for a mesh generation

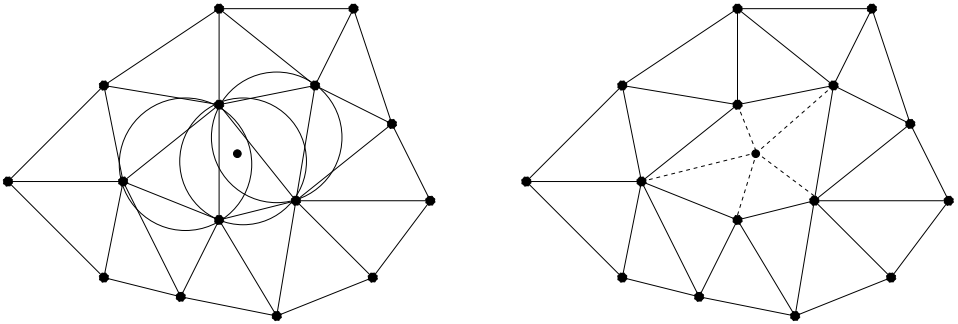

Figure 2 Illustration of the Bowyer/Watson algorithm for constructing Delaunay triangulations.

strategy where the mesh points have been predetermined. The mesh points are put in a list and an initial triangulation is artificially constructed (with auxiliary points) that completely covers the domain to be gridded. The mesh points in the list are then inserted sequentially into the existing triangulation using the Bowyer-Watson algorithm. The final mesh is obtained when all points from the list have been inserted. The mesh point distribution can in principle be random, but most often it is either pregenerated using Cartesian mesh stencils (Baker 1987) or multiple overlapping structured meshes (Weatherill 1985; Mavriplis 1990a, 1991b) or may be generated on the fly using automatic point placement strategies.

A simple automatic point placement technique has been proposed in Holmes & Snyder (1988). Starting with an initial coarse triangulation that covers the entire domain, a priority queue is constructed based on some parameter of the individual triangles (circumradius, for example). A field value is assumed to exist that determines the local maximum permissible value for the circumradius of the triangles (or any other parameter). The first triangle in the queue is examined, and a point is added at its circumcenter if the triangle circumradius is larger than the locally prescribed maximum. This new point is inserted into the triangulation using Bowyer-Watson's algorithm, and the newly formed triangles are inserted into the queue if their circumcircles are too large; otherwise they are labeled as acceptable and do not appear in the queue. The final grid is obtained when the priority queue empties out.

Delaunay triangulation techniques based on point insertion extend naturally to three dimensions by considering the circumsphere associated with a tetrahedron. These methods are more efficient than advancing-front techniques owing to the simplicity of the algorithm. However, they are only valid on convex domains. Therefore, Delaunay triangulation methods typically employ an artificially constructed convex domain (such as a quadrilateral in two dimensions or

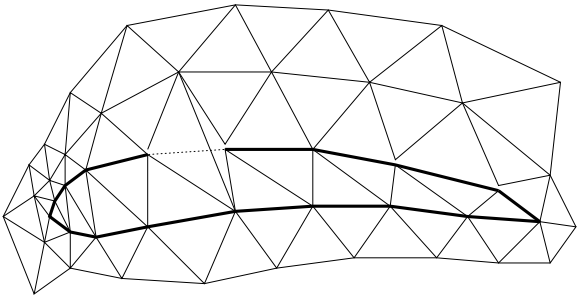

Figure 3 Illustration of non-boundary-conforming triangulation for a simple airfoil geometry.

a cube in three dimensions) that covers the entire flow-field. The triangulation is initially performed in this domain, and the triangles or tetrahedra that lie in the flow-field are then extracted from the original convex domain to form the computational mesh. The main disadvantages of Delaunay triangulation techniques relate to their inability to guarantee boundary integrity. In two dimensions, the set of edges that define the domain boundaries must form a subset of the edges of the original convex domain triangulation. If this is not the case, edges of the triangulation exist that penetrate the boundary, as shown in Figure 3.

In three dimensions, the equivalent problem is that of constructing a tetrahedralization of the domain that contains, as a subset of its faces, a given surface boundary triangulation. The use of local transformations as described in the following section (edge and face-edge swapping, in two and three dimensions, respectively), has been developed for locally modifying Delaunay meshes to conform to specified boundary discretizations (George et al 1991, Weatherill et al 1993).

In two dimensions, the existence of constrained Delaunay triangulations (Chew 1989) guarantees the possibility of constructing triangulations that contain a prescribed subset of edges and thus respect a given boundary discretization. In three dimensions, the recovery of prescribed boundary triangulations may not always be achievable through swapping techniques. A somewhat less restrictive problem is that of constructing a tetrahedralization of the boundary and interior points for which no edges or faces intersect the boundary surfaces. One way of achieving this is by modifying the boundary point resolution (by judiciously adding surface points) in regions where breakthroughs occur (Baker 1991).

2.3 *Edge and Face Swapping Techniques*

The Delaunay triangulation represents but one of many possible triangulations of a given point set. An algorithm for transforming one triangulation to another

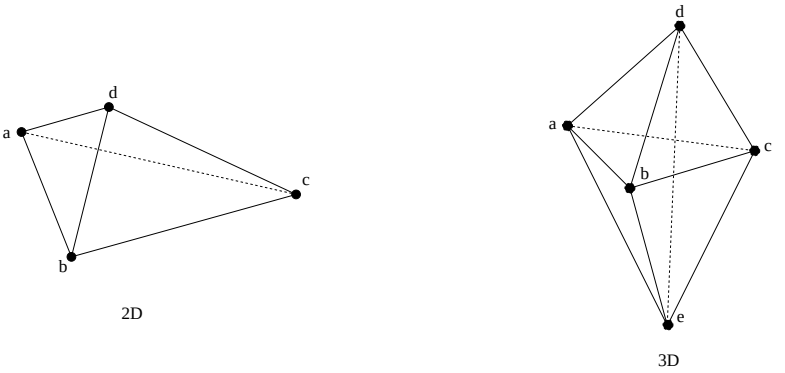

Figure 4 Two possible configurations for the diagonal of a convex pair of triangles in the edge-swapping algorithm, and for two tetrahedra in the three-dimensional face-edge swapping algorithm.

triangulation of the same point set is given by Lawson (1972). Since all two-dimensional planar graphs obey Euler's formula (Mortenson 1985), all possible triangulations of a given set of points contain the same number of edges and triangles. Thus, any one triangulation may be obtained by simply rearranging the edges of another triangulation of the same set of points. The edge-swapping algorithm of Lawson (1972, 1977) consists of successively examining pairs of neighboring triangles in the mesh. For each pair of triangles there exist two possible configurations of the diagonal edge, as shown in Figure 4.

The algorithm chooses the diagonal configuration that optimizes some given criterion. The algorithm terminates when no further optimizations are possible. The edge-swapping algorithm can be used to transform an arbitrary triangulation into a Delaunay triangulation. In this case, the edge-swapping criterion to be used is the edge configuration that maximizes the minimum angle within the two neighboring triangles. (Delaunay triangulations are also known as max-min triangulations: They maximize the smallest angles in the triangulation.)

Edge-swapping can be used to construct triangulations other than the Delaunay triangulation, such as the minimum maximum angle (min-max) triangulation, or the minimum total edge length triangulation (Preparata & Shamos 1985, Barth 1991a). However, the edge-swapping procedure is not guaranteed to result in the global optimum triangulation for these cases: The algorithm may terminate within local optima. A more sophisticated and complex algorithm, $(O(N^2 \log N))$, known as the edge-insertion algorithm (Edelsbrunner et al 1992), has been developed for transforming a given triangulation into the global optimum min-max triangulation.

In three dimensions, analogous local transformations can be achieved through face- and edge-swapping. By considering pairs of neighboring tetrahedra, the

common triangular face may be removed and the edge joining the two opposing end points of the tetrahedra can be drawn, as shown in Figure 4. Note that the original configuration that contained two tetrahedra is transformed into a configuration containing three tetrahedra. Joe (1989) has shown that, when a new point is introduced into an existing Delaunay tetrahedralization and connected in an arbitrary manner, the global Delaunay tetrahedralization of the new point set can be recovered through a sequence of face-edge swappings. However, the more general case of transforming an arbitrary tetrahedral mesh into the Delaunay tetrahedral mesh through face-edge swapping often terminates prematurely in a local optimum.

Face-edge swapping procedures are nevertheless useful in improving the quality of three-dimensional meshes. In fact, Delaunay tetrahedralizations are not necessarily the best constructions for three-dimensional meshes, since sliver tetrahedra may easily be formed when four approximately co-circular points are encountered. Therefore, face-edge swapping techniques based on the min-max property are often employed to avoid or repair such ill-shaped elements, even though the global min-max optimum for the entire grid is most often unattainable.

2.4 *Other Triangulation Methods*

Quad/octree-based methods (Yerry & Shephard 1984) perform mesh generation through a recursive subdivision of space down to a prescribed (spatially varying) resolution. The vertices of the resulting quad or octree structure are used as mesh points, and the tree quads or octants are divided up into triangular or tetrahedral elements, in two or three dimensions, respectively. At boundaries, the quad/octree cells intersect the boundary surfaces, and the vertices must be displaced or warped to coincide with the boundary. The method is relatively simple, inexpensive, and produces good quality meshes in interior regions of the domain. The technique's main drawback is an irregular cell distribution near boundaries.

Various techniques have been developed that combine the advantages of both advancing-front methods with the theoretical rigor of Delaunay triangulations. Rebay (1993) and Mueller & Roe (1992) have developed similar algorithms, where the triangulation method is based on Bowyer-Watson's Delaunay triangulation algorithm, but the point-insertion process is designed to mimic that achieved by the advancing-front method. A background field function that determines the desired distribution of element sizes is defined. A constrained Delaunay triangulation covering the entire domain and conforming to the boundaries is initially constructed. Mesh vertices are then inserted along the boundary edges, at locations determined by the field function. This procedure results in near-boundary elements that conform to the element size distributions specified

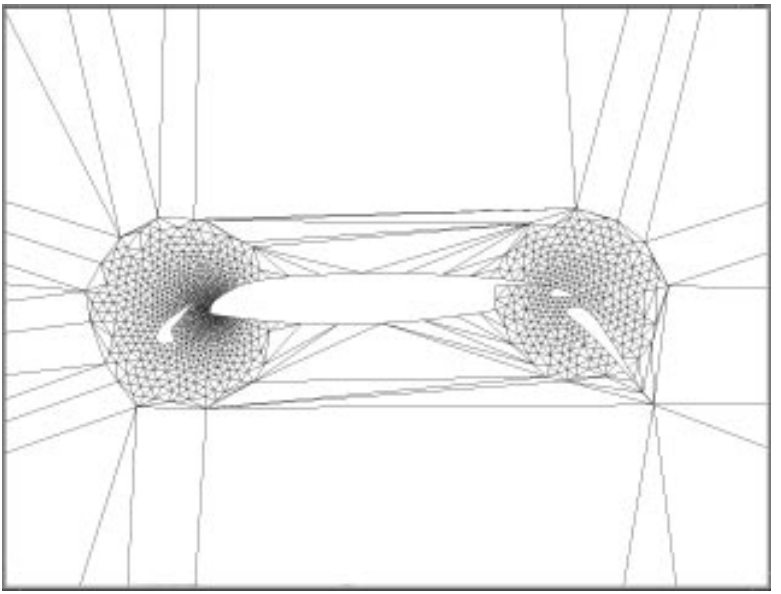

Figure 5 Partially completed advancing-front Delaunay point-insertion mesh.

by the background function. The boundary between these accepted elements and other elements much larger than the required field function values constitutes the front, and new mesh points are continuously added ahead of this front.

Figure 5 illustrates this process, where the front is clearly visible and is advanced based on the minimum edge length in the front. The result is the very smooth point distribution and high-quality mesh usually associated with advancing-front techniques, coupled with the efficiency of the Delaunay point-insertion algorithm, although the boundary integrity difficulties of the latter method are inherited.

Marcum and Weatherill (1994) have developed a three-dimensional advancing-front point-insertion algorithm that makes use of face-edge swapping. An initial boundary-conforming mesh is constructed, and new points are inserted as described above. However, the Bowyer-Watson algorithm is not employed; instead, the newly inserted vertex is simply connected to the four corners of the existing element that contains it. A face-edge swapping operation is then employed to improve the connectivity pattern of the elements in the vicinity of the new vertex. One of the advantages of this approach is that the swapping algorithm can be based on criteria other than the Delaunay triangulation. In fact, the min-max criterion is most often employed in the reconnection strategy

because it avoids sliver tetrahedra. However, a straightforward application of the min-max criterion quickly gets caught in local optima and yields poor-quality meshes. A key feature that enables a closer approximation of the global optimum and yields higher-quality meshes is the use of an intermediate Delaunay-based swapping criterion. In this approach, each time a new point is inserted into the mesh, the surrounding faces and edges are first swapped according to the Delaunay triangulation criterion and then reswapped according to the min-max criterion. The use of this intermediate construction serves to broaden the range of influence of the new point, providing a more global effect and avoiding the immediate local optima that typically plague the straight application of the min-max criterion.

2.5 *Stretched Mesh Generation*

The drive towards full Navier-Stokes solvers has necessitated the development of stretched grid generation techniques in order to resolve the thin boundary-layers, wakes, and other viscous regions characteristic of high-Reynolds-number viscous flows. Proper boundary-layer resolution usually requires mesh spacing several orders of magnitude smaller in the direction normal to the boundaries than in the streamwise direction, resulting in large cell aspect-ratios in these regions.

In Babushka & Aziz (1976) it is shown how the accuracy of a two-dimensional finite-element approximation on triangular elements degrades as the maximum angle of the element increases. Therefore, stretched obtuse triangles that contain one large angle and two small angles are to be avoided, while stretched right-angle triangles, with one small and two nearly right angles, are preferred. Delaunay triangulations, which maximize the minimum angles of any triangulation, tend to produce equiangular triangulations and are thus ill suited for the construction of highly stretched triangular elements. One of the earliest approaches for generating highly stretched triangulations for viscous flows makes use of a Delaunay triangulation performed in a locally mapped space (Mavriplis 1990b, 1991b; Vallet et al 1991; Castro-Diaz et al 1995). By defining a mapped space based on the desired amount and direction of stretching, an isotropic Delaunay triangulation can be generated in this mapped space that, when mapped back to physical space, provides the desired stretched triangulation. Difficulties with such methods involve defining the stretching transformations and determining suitable point distributions for avoiding obtuse triangular elements.

The fact that obtuse triangles should be avoided has led to the development of methods based on minimum-maximum angle triangulations in both two and three dimensions (Barth 1991a, Marcum 1995). These can be implemented either as a transformation from a Delaunay or other triangulation to a more optimal min-max triangulation or by generating the min-max connectivity on the fly,

as points are inserted, using local reconnection techniques. Although the same reconnection strategy can be employed in stretched and unstretched regions of the flow-field, different point placement strategies must be employed in these regions, in order to obtain the required anisotropic mesh-point resolution in the stretched regions, and to avoid the formation of obtuse triangular elements, which cannot be eliminated by reconnection strategies alone. Point-placement strategies for the highly stretched regions most often mimic structured hyperbolic mesh generation techniques, where the points are placed along normal lines emanating from each boundary vertex at intervals determined by the specified degree of mesh stretching. One of the drawbacks of min-max triangulation approaches is that the resulting connectivity may be very sensitive to the distribution of vertices, with very small differences in mesh-point distributions leading to substantially different connectivities.

An alternative to the above approaches is to generate a locally structured or semistructured mesh in the regions where high stretching is required. One approach (Nakahashi 1987, Ward & Kallinderis 1993) attempts to preserve the mesh structure in the direction normal to the boundary up to a specified distance away from the boundary, after which fully unstructured isotropic meshing techniques are employed. Special care must be taken in this case to avoid mesh cross-overs in regions of concave curvature and to ensure a smooth transition between the structured and unstructured region of the mesh. Another strategy Löhner 1993; Pirzadeh 1994a,b; Connell & Braaten 1995) consists of generating a semistructured mesh, where the "stack" of mesh cells emanating from each individual boundary face may terminate independently from those at other boundary faces, as shown in Figure 6. Termination of these "advancing-layers" (Pirzadeh 1994a) is triggered when the local cell aspect-ratio approaches unity, or when cross-over with other cells is detected, such as in concave corners. The remaining region is then gridded with a conventional isotropic unstructured mesh generation approach. The resulting structured or semistructured meshes can either be conserved as local structured entities of quadrilaterals in two dimensions and prisms in three dimensions (since the surface grid is generally assumed to be triangular), or the different element types may be divided into triangles or tetrahedra in two or three dimensions, respectively.

2.6 *Mixed-Element Meshes*

Delaunay constructions and face-edge swapping procedures are only relevant for simplicial meshes (triangles in two dimensions, tetrahedra in three dimensions). While simplicial elements provide the most flexibility in discretizing complex geometries, the use of other types of elements may be beneficial. The most common alternate element types are quadrilaterals in two dimensions, and prisms, pyramids, and hexahedra in three dimensions.

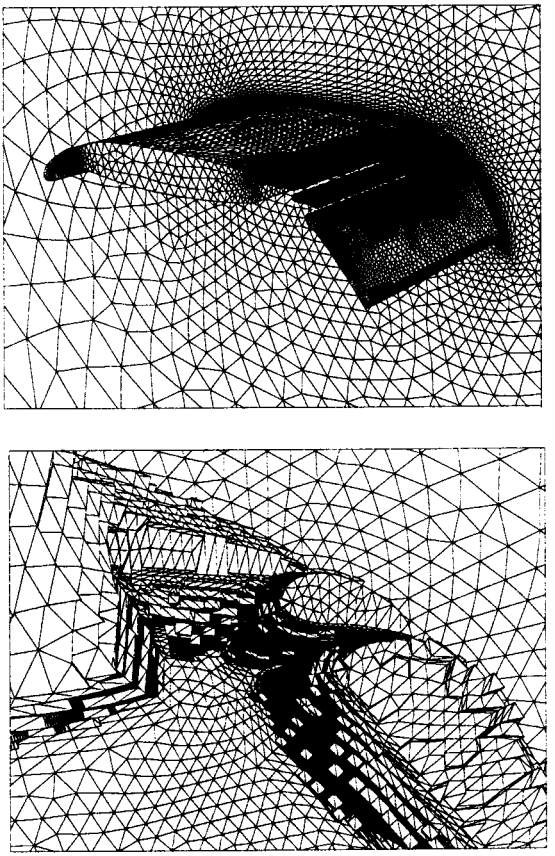

Figure 6 Illustration of the advancing-layers method for three-dimensional unstructured mesh generation about segmented wing geometry. [Reproduced from Pirzadeh 1994b with permission.]

One of the advantages of nonsimplicial meshes is their lower connectivity. For example, while tetrahedral meshes contain approximately six times as many cells as vertices, and seven times as many edges as vertices, hexahedral meshes contain the same number of cells as vertices, and only three times the number of edges, while prismatic meshes contain twice as many cells and four times as many edges (asymptotically, neglecting boundary effects). Since the cost of a vertex-based discretization is directly proportional to the number of edges in the mesh, a substantial efficiency gain can be realized by operating on nonsimplicial meshes for a given number of unknowns or grid points. Furthermore, while regular arrays of hexahedra or prisms can be employed to discretize three-dimensional space, it is impossible to pack space with arrays of self-similar tetrahedra. Therefore, regular arrays of nonsimplicial elements may also enhance accuracy, owing to a local cancellation of truncation errors that may not occur on groups of nonsimilar tetrahedra. On the other hand, much of the accuracy argument is either qualitative or speculative, and a controlled quantitative study has yet to be performed. For highly stretched two-dimensional grids, Aftosmis et al (1994) observed similar accuracy on equivalent grids of quadrilaterals and triangles in boundary layer regions and advocate the use of quadrilateral elements owing to their lower connectivity.

Hybrid or mixed-element meshes may be constructed in a variety of manners. For example, advancing-front techniques may be modified to generate quadrilateral or hexahedral elements (Blacker & Stephenson 1991, Zhu et al 1991), although simplicial elements are often required to mend the mesh in regions of geometric complexity. In regions of high mesh stretching, hexahedral or prismatic meshes are in some sense natural (Nakahasi 1987, Ward & Kallinderis 1993, Connell & Braaten 1995), depending on whether the surface discretization contains triangular or quadrilateral elements, because the definition of a stretching direction implies a certain degree of structure in the mesh, which can be exploited using locally structured or semistructured mesh generation techniques.

Another approach is to attempt to reduce the complexity of an existing mesh by combining simplicial elements together to form more complex elements with lower connectivity (Merriam 1991, Tembulkar & Hanks 1992, Mavriplis & Venkatakrishnan 1995b). Mesh merging techniques of this sort are most often based on detecting groups of triangular or tetrahedral elements whose forming vertices are nearly co-circular and which constitute a well-shaped quadrilateral or prismatic element when merged. While merging algorithms have been relatively successful in two dimensions, robust three-dimensional merging criteria have yet to be demonstrated. A major advantage of the mesh-merging approach is that the hybrid representation of the mesh remains internal to the flow solver, which greatly simplifies other operations such as mesh adaptation and visualization, which may operate on a fully simplicial representation of the mesh.

2.7 *Adaptive Meshing*

Aside from the treatment of complex geometries, the second main advantage of unstructured meshes is the ease with which solution-adaptive meshing may be implemented. Since no inherent structure is assumed in the representation of the mesh, mesh points may be added, deleted, or displaced, and the mesh connectivity may be locally reconfigured in the affected regions. The goal of mesh adaptation is the determination of the optimum mesh-point distribution that results in equipartition of the error for each individual simulation.

The character of the problem to be solved dictates the requirements of the mesh adaptation strategy. For example, steady-state problems usually involve a small number of adaptation phases as part of a lengthy solution process. Therefore, relatively sophisticated adaptation strategies can be employed, such as, in the extreme case, complete mesh regeneration. Mesh refinement procedures are most important here, while de-refinement has only a minor effect and can often be omitted for steady-state cases. For transient problems, mesh adaptation must be performed every several time steps, and thus efficiency is much more important than optimality. Mesh refinement and de-refinement are both essential for transient cases, as well as mesh movement for cases with moving boundaries. Furthermore, the accuracy of interpolation from the original mesh to the refined mesh affects the solution accuracy (unlike the steady-state case), and thus accurate transfer schemes are required.

Delaunay-based mesh generation techniques can easily be extended to incorporate adaptive refinement capabilities (Mavriplis 1990a, 1995b; Weatherill et al 1993; Barth 1992). Once a solution has been obtained on an initial mesh, new points can be added in regions where high errors are detected. These new points can be triangulated into the mesh using the Bowyer-Watson point-insertion algorithm. Alternatively, if a non-Delaunay mesh is employed, new points may be inserted through element subdivision, and the connectivity of the resulting mesh may be optimized through several face-edge swapping passes based on any appropriate criterion.

Rule-based hierarchical element subdivision is a very effective adaptive technique, particularly for unsteady flows, where efficiency and accuracy of interpolation between successive grids are important considerations (Stoufflet et al 1987, Löhner & Baum 1992, Rausch et al 1992, Braaten & Connell 1996). The essential approach consists in recursively subdividing mesh elements where large solution errors are detected, but conforming to a set of well-defined subdivision rules, which are necessary to prevent the formation of degenerate element shapes and connectivities. Storing the hierarchy of the recursive process enables de-refinement to be implemented by simply retracing the subdivision history. Hierarchical subdivision is also applicable to nonsimplicial hybrid meshes by constructing a library of subdivision rules for each element type.

Mesh movement or r-refinement has applications in both steady and unsteady flows. For example, r-refinement may be useful for optimizing the normal resolution of the mesh in a developing boundary layer or for clustering points around shock waves (Palmerio 1994, Castro-Diaz et al 1995). For transient problems with moving boundaries, r-refinement is indispensable. Difficulties center around conserving a minimum degree of grid quality under severe deformations. In areas where grid deformation becomes unacceptable, local reconnection through swapping techniques can easily be performed, but only for simplicial meshes (Kennon et al, Baum et al 1994, Venkatakrishnan & Mavriplis 1995b). In practice, combinations of h- and r-refinement are often employed (Baum et al 1994, Castro-Diaz et al 1995) for such cases.

Central to the implementation of any solution-adaptive scheme is the ability to detect and assess solution error. The construction of a suitable refinement criterion represents the weakest point of most adaptive strategies. The main problem is that an exact characterization of the error requires a knowledge of the solution itself, which is obviously impractical. Most error estimates are based on the assumption that the solution is smooth and asymptotically close to the exact solution. This is often not the case for fluid dynamics problems, which are governed by nonlinear hyperbolic partial-differential equations. Solutions may contain discontinuities, and downstream flow features often depend on adequate resolution of upstream flow features. Most refinement criteria are heuristically based on (undivided) gradients and/or second differences of the various flow variables. Conservative criteria (i.e. over-refining) are often employed to compensate for the inability to accurately characterize the true solution error. Because adaptive meshing results in different mesh topologies for each simulation, even when the geometry of the problem is unchanged, parametric studies (typically used in design processes) are complicated by the requirement to distinguish between grid-induced and physical solution variations. Nevertheless, for problems with disparate length scales, adaptive meshing is often indispensable for resolving small flow features, and their full potential awaits the development of more well-founded adaptive criteria.

3. SPATIAL DISCRETIZATION

The previous section describes techniques for generating suitable meshes about arbitrary geometric configurations. Once such a mesh has been generated, it serves as the basis for the spatial discretization of the governing fluid dynamic equations. Unstructured mesh discretization techniques can generally be classified as finite-volume or finite-element strategies, depending on whether a discrete integral or variational viewpoint is adopted, although many common discretizations may be simultaneously interpreted from both viewpoints.

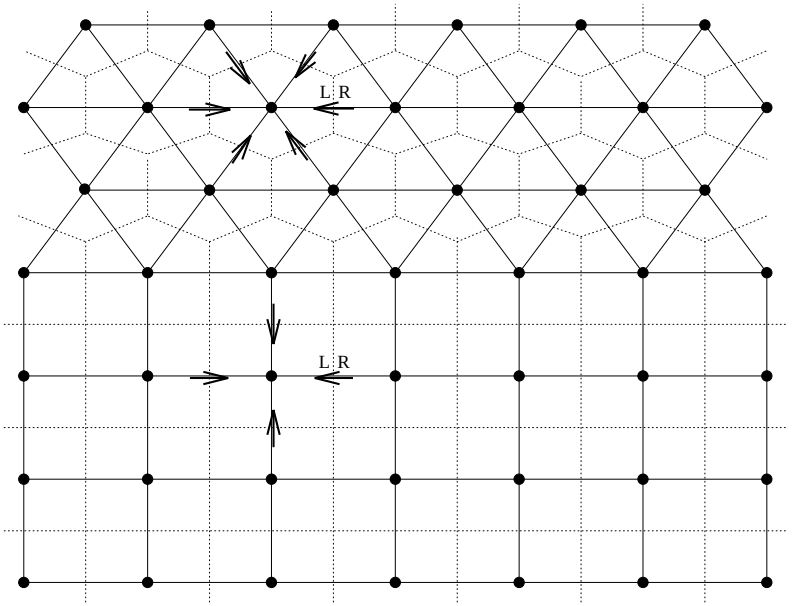

Figure 7 Illustration of dual mesh for mixed triangular-quadrilateral unstructured mesh and associated control-volumes with edge-based fluxes for a vertex-based scheme.

For finite-volume formulations, a distinction between vertex-based and cell-centered schemes can be made. In a vertex-based scheme, the flow variables are stored at the vertices of the mesh. The discrete equations usually involve stencils that are defined by the nearest or next-to-nearest neighbors of each mesh point. In a cell-centered scheme, the flow variables are stored at the centers of the cells or elements of the mesh. The stencils of the discrete equations usually involve the nearest or next-to-nearest neighboring cells. The equivalence between vertex-based and cell-centered schemes can be demonstrated with the concept of a dual mesh. If a dual mesh point is created at each cell center and dual mesh edges are drawn by joining neighboring cell centers, the cell-centered scheme can be seen to be equivalent to a vertex-based scheme operating on the dual mesh.

Figure 7 illustrates the dual mesh for a mixed quadrilateral and triangular mesh in two dimensions. For a purely quadrilateral or hexahedral mesh, the dual mesh also contains quadrilateral or hexahedral elements, and the number of vertices and edges in the original and dual meshes is identical. For triangular or tetrahedral meshes, the number of dual mesh vertices is larger than the number of original mesh vertices. This is due to the fact that a triangular mesh contains twice as many triangles as vertices (neglecting boundary effects), and

a tetrahedral mesh five to six times more elements than vertices. On the dual mesh, the degree of a vertex (number of incoming edges) is fixed and equal to three for triangular elements, or four for tetrahedral elements, whereas on the original mesh, the degree of each vertex is variable. Similar relationships hold for other elements such as prisms and pyramids.

Thus, cell-centered and vertex-based schemes operating on the same grid result in vastly different discretizations when nonquadrilateral or nonhexahedral elements are present. In particular, a cell-centered scheme can be expected to result in a much larger number of unknowns while generating relatively simple stencils of fixed size. The vertex-based scheme, on the other hand, will result in a smaller number of unknowns with larger variable-size stencils. Because of the larger number of unknowns, cell-centered schemes generally incur larger overheads than vertex-based schemes on equivalent grids. However, there is evidence to suggest that they also provide more accurate solutions on equivalent grids. The question of whether the additional overheads are offset by the increase in accuracy is still an open one, especially since vertex-based schemes operate on more complex stencils (more fluxes per unknown) than cell-centered schemes.

3.1 *Central Differencing and Riemann-Based Upwind Methods*

A simple finite-volume discretization can be obtained by defining a control volume about each unknown (mesh-element for cell-centered schemes, dual-mesh cell for vertex-based schemes) and integrating the fluxes over the boundary of the control volume using a simple midpoint integration rule, where the value of the flux at each control volume face, as shown by the arrows in Figure 7, is computed by averaging the flow variables in the two control volumes on either side of the face. For vertex-based schemes on simplicial meshes, this particular scheme can also be derived using Galerkin finite elements, where the fluxes are assumed to vary linearly over the triangular or tetrahedral elements of the mesh (Jameson et al 1986, Rostand & Stoufflet 1988, Barth 1991a). Such a scheme corresponds to a central difference discretization on structured meshes and requires the addition of artificial diffusive terms to ensure stability. By analogy with the blended second and fourth differences employed on structured meshes (Jameson et al 1981), artificial dissipation is constructed as a combination of an undivided Laplacian and biharmonic operator. First-order shock-capturing diffusive terms are provided by the Laplacian operator, which is turned off away from shocks in regions of smooth flow where the biharmonic dissipation ensures stability with second-order accuracy. This approach has been used extensively to compute inviscid and viscous two- and three-dimensional flows (Jameson et al 1986; Mavriplis 1987, 1992; Smith 1990; Mavriplis & Venkatakrishnan

1995a; Peraire et al 1992). The discretization is simple to implement and relatively inexpensive to evaluate. However, the construction of the dissipative terms is nonoptimal in that it does not differentiate between the various wave components of the governing fluid dynamic equations.

Upwind schemes attempt to remedy this deficiency by applying the appropriate amount of dissipation to each wave component. This is achieved by diagonalizing the system of governing equations, using a local one-dimensional approximation normal to each control-volume interface, and constructing the dissipation terms in the diagonalized or "split" form of the equations. In practice, this is done by solving a Riemann problem at the interfaces between neighboring control volumes. For first-order schemes, the flow variables are assumed to be piecewise constant over the control volumes. The values on either side of the interface, often referred to as "left" and "right" values (denoted by **L** and **R** in Figures 7 and 8), are then simply taken as the values in the neighboring control volumes. For second-order approximations, the flow variables are assumed to vary linearly over the control volumes, and interface values must be extrapolated from centroidal values to the control-volume interfaces using estimated gradient information. Higher-order methods may similarly be

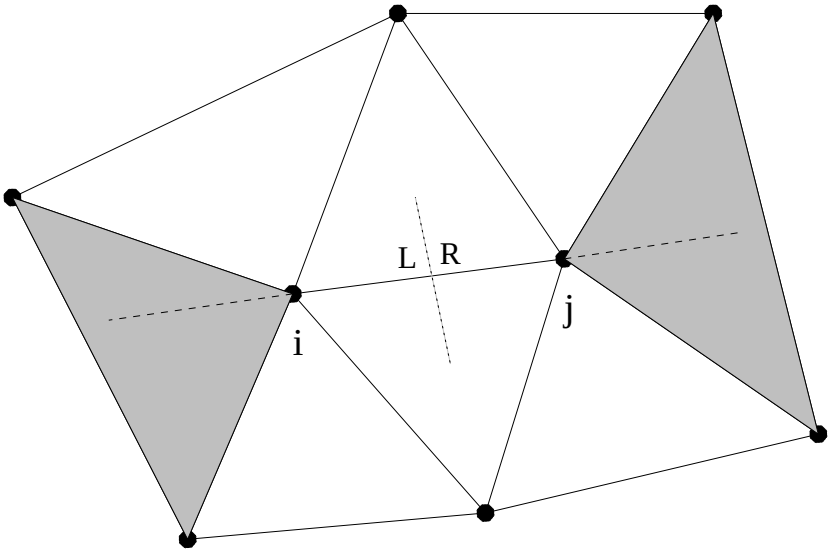

Figure 8 Illustration of neighboring triangles employed to compute gradients in direction of edge ij for schemes in Desideri & Dervieux 1988 and Jameson 1994. L and R refer to the left and right states about the control-volume interface associated with edge ij for the scheme of Desideri & Dervieux (1988).

formed by reconstructing approximations to higher-order derivatives in the solution variables, which are then used to extrapolate more accurate values to either side of the interface. The resulting interface values are discontinuous, although in the absence of shock waves the magnitude of the jumps decreases with increasing order of accuracy of the scheme. A Riemann solver is then required to calculate a flux at the interface given the discontinuous left and right states. Approximate Riemann solvers are typically employed, the most common being those of Osher (1984) and Roe (1981). In order to prevent oscillations near discontinuities such as shocks, the extrapolated interface values must be limited, such that the creation of new extrema is never permitted.

Some of the earliest applications of upwind schemes to unstructured meshes were reported by Desideri & Dervieux (1988). They developed a vertex-based scheme that estimates gradients along an edge from the projection of gradients from neighboring triangles, as depicted in Figure 8, and employs an Osher Riemann solver. Limiting was applied along cell interfaces in a quasi-one-dimensional manner. Barth & Jespersen (1989) proposed a second-order scheme, using a Roe Riemann solver, where both reconstruction and limiting are done in a truly multidimensional fashion. Interface values are extrapolated from the control-volume centroids by computing a gradient at each control volume using a Green-Gauss integration around the control volume boundary. These gradients are then limited to avoid the formation of new extrema, based on neighboring values. The method was later extended to include a least-squares technique for estimating gradients (Barth 1991a), which was shown to produce better approximations on distorted meshes. Extensions using high-order reconstructions have also been developed (Barth 1991b). A cell-centered upwind scheme for three dimensional tetrahedral meshes has been developed by Frink (1992) which achieves linear reconstruction by computing cell-based gradients using vertex values obtained by a weighted average of surrounding cell-based quantities. One of the drawbacks of upwind schemes is the required use of a nonlinear limiter to capture discontinuities such as shock waves without numerical oscillations. Many limiter formulations have been developed, but most are sensitive to small variations in the solution variables, which in turn adversely affect convergence of the overall scheme. Venkatakrishnan (1995b) has investigated this matter and has proposed a weaker form of limiter which permits small oscillations which vanish as the grid is refined, but provides superior convergence characteristics.

The development of upwind schemes for unstructured meshes has enabled the re-evaluation of central difference schemes with added dissipation as a scalar counterpart to upwind schemes. This has resulted in alternate formulations based on gradients rather than Laplacian and biharmonic operators for the artificial dissipation terms (Peraire et al 1992).

3.2 *Other Approaches*

A discretization scheme is said to be positive if, in the scalar case, it can be expressed in the form

$$\frac{dw_i}{dt} = \sum_j C_{ij}(w_j - w_i) \tag{1}$$

where the C_{ij} coefficients are all positive. The summation is over the neighbors j of i. If w_i is a maximum, then $w_j - w_i$ is negative and $\frac{dw_i}{dt} < 0$, from the positivity of the C_{ij}. Thus maxima must be decreasing, and conversely minima must be increasing. This Local Extremum Diminishing (LED) principle (Jameson 1994) represents a multidimensional generalization of the Total Variation Diminishing (TVD) concept (Harten 1984). This has important implications for the construction of robust and accurate schemes. The LED principle can be used to guarantee the capture of shock waves without any oscillations. Furthermore, the maximum principle inherent in these schemes can provide a robust solution strategy—for example, by guaranteeing the positivity of all flow quantities simply if they are initialized as positive quantities.

Unfortunately, if the C_{ij}'s are constant, the scheme is linear and can be at best first-order accurate. To achieve second-order accuracy, a nonlinear scheme must be constructed, where the C_{ij} coefficients depend on the solution variables. These usually include the use of limiters based on ratios of successive gradients on neighboring mesh intervals. These methods have their roots in the early work of Boris & Book (1973) on flux-corrected transport schemes. The original idea is to construct a first-order scheme, the accuracy of which is then refined by adding specific amounts of antidiffusive fluxes, under the constraints of monotonicity. This scheme was extended to unstructured meshes by Löhner in the FEM-FCT schemes (Löhner et al 1987). Jameson has derived a Symmetric Limited Positive (SLIP) scheme that is both positive and second-order accurate in smooth regions for unstructured meshes (Jameson et al 1986), where one-dimensional solution variations along stencil edges are obtained by projecting adjacent triangle gradients into the direction of the edge, in a manner similar to that of Desideri & Dervieux (1988) (cf Figure 8). These schemes can be applied to the unsplit form of the governing flow equations, in a scalar dissipation mode, or to the split equations, characteristically or otherwise, to achieve more accurate resolution of shock waves and boundary layers. However, for systems of equations, monotonicity is not strictly guaranteed, as it is in the scalar case. Additionally, these schemes rely on the use of a nonlinear limiter in all regions of the flow-field (not only in the vicinity of shock waves), which may adversely impact convergence, depending on the form of limiter employed.

The weak link in the upwind schemes described in the previous section involves the use of a dimensionally split Riemann solver, which is applied in

a quasi-one-dimensional fashion normal to control-volume interfaces. While this approach is employed routinely for structured and unstructured grids, it may misinterpret flow features not aligned with the control-volume interfaces. Fluctuation splitting schemes provide an alternative to Riemann solvers for introducing upwinding. In this approach, variables are stored at the mesh vertices, and intermediate residuals are computed at cell centers. The values at the vertices are then updated by redistributing the cell-based residuals to the vertices. If these residuals are redistributed in an unbiased manner, then the well-known Ni scheme (Ni 1982) is recovered, which corresponds to a central difference scheme and requires additional dissipative terms. Upwind schemes are obtained by introducing downwind biasing into the redistribution formulae by considering the effect of linear wave solutions evolving the piecewise linear data over a mesh cell. Several upwind fluctuation-splitting schemes for the scalar advection equation have been developed and are compared in Paillere et al (1993). Extensions of fluctuation-splitting schemes to systems of equations rely on the decomposition of the flux divergence into scalar waves. The decomposition is not unique, and several strategies such as characteristic decomposition (Deconinck et al 1986), simple wave models (Parpia & Michalak 1993, Roe 1986), and algebraic approaches (Sidilkover 1994) have been investigated. The advantages of fluctuation-splitting schemes are that they do not require the use of dimensionally split Riemann solvers and they result in a compact nearest-neighbor stencil for simplicial meshes. However, they are second-order accurate only at steady state (Venkaktkrishnan 1996).

In the finite-element approach, the solution is expanded in a set of basis functions, and the discrete equations are obtained by requiring the residuals to be orthogonal to a set of trial functions. For second-order approximations, the basis functions are generally taken as linear functions of compact support, which have the value 1 at a given vertex and vanish at all other vertices. The solution expansion is obtained by multiplying these basis functions by vertex-based-solution variables and summing over the entire grid. When the test functions and the basis functions are the same, the method is called Galerkin, otherwise it is called Petrov-Galerkin. The correspondence between (*a*) a Galerkin finite-element strategy using linear elements and (*b*) finite-volume based central-differencing has already been discussed. For the convective terms, additional artificial dissipation is required for stability. The Galerkin finite-element method can also be employed to discretize the viscous terms for the Navier-Stokes equations, as discussed in the next section.

The Streamwise Upwind Petrov Galerkin (SUPG) (Hughes & Brooks 1979) and Galerkin least-squares methods (Hughes et al 1989, Hughes 1987) introduce dissipation to a regular Galerkin method by adding terms proportional to the residual. In contrast to traditional artificial-dissipation formulations,

these approaches do not compromise the accuracy of the scheme, since at steady-state convergence, the dissipation vanishes along with the residual. The global error in the L_2 norm can be shown to vary as $h^{p+\frac{1}{2}}$ for the advection dominated case and as h^{p+1} for the diffusion dominated case, where p is the degree of the piecewise-polynomial basis function (Hughes 1987, Venkatakrishnan 1996). However, these methods are linear and not suitable for capturing discontinuities. A shock-capturing operator that locally reduces the order of accuracy near discontinuities has been proposed by Hughes & Mallet (1986).

3.3 *Considerations for the Navier-Stokes Equations*

The above discussion on discretizations is principally concerned with the purely convective terms that arise in the context of inviscid flows. With the exception of the finite-element methods described above, where convection and diffusion terms can be treated in a unified manner, additional viscous terms must be discretized for the Navier-Stokes equations. These terms are diffusive in nature and generally take the form of second differences. Stable second-order discretizations can therefore be constructed using simple central differences.

For cell-centered schemes, one approach consists of first computing the gradients at the mesh vertices and then averaging these to the cell faces in order to compute second derivatives at cell centers (Frink 1994). For vertex-based schemes, a similar two-pass procedure based on finite-volume central-difference arguments can be used to construct the viscous term discretization. For simplicial meshes, gradients can first be computed on the mesh elements, which can then be interpolated to the control-volume faces in order to form second differences at the mesh vertices. This type of discretization can be derived more formally using a Galerkin finite-element procedure and assumes linear variation of the flow variables on the mesh elements. The resulting discretization produces a nearest-neighbor stencil and may be implemented as a single loop over the edges of the mesh, rather than as a two-pass procedure that computes intermediate cell-based gradients (Barth 1992, Mavriplis 1995c). For nonsimplicial meshes, Galerkin finite-element formulations using bilinear or trilinear variations on quadrilateral or hexahedral elements can be constructed. The resulting stencils, however, are no longer compact, as they involve vertices within mesh elements that are not connected by a mesh edge, such as diagonally opposed vertices in hexahedral elements (Braaten & Connell 1996).

An alternative strategy for discretizing viscous terms for vertex-based schemes is to employ the vertex-based gradients already computed in the context of second-order upwind schemes (using a Green-Gauss integration around the

vertex-based control volumes, for example), instead of the element-based gradients described above. A vertex-based second difference can then be computed by integrating these gradients themselves around the control-volume boundaries (Luo et al 1993). This approach enables the viscous term discretizations to be assembled on meshes of arbitrary element types using the same data structures as required for the upwind convection terms. The principal drawback of this method is that it results in a large stencil that involves neighbors and next-to-neighbors, which on a structured mesh reduces to a stencil of size $2h$, where h represents the mesh spacing. This not only reduces overall accuracy but is also ineffective at damping odd-even oscillations in the numerical scheme.

For high-Reynolds-number flows of practical interest, the Reynolds-averaged form of the Navier-Stokes equations is generally employed, which requires the use of additional turbulence-modeling equation(s). Although algebraic models can be implemented on unstructured grids (Mavriplis 1991a), they were conceived for simple wall-bounded flows and are thus ill suited for flows over complex geometries. The current practice is to employ two-equation models of the $k-\epsilon$ or $k-\omega$ type, or simpler one-equation eddy viscosity models (Baldwin & Barth 1992, Spalart & Allmaras 1992). These equations contain convective, diffusive, and source terms that can be discretized in a manner analogous to the discretization of the flow equations. Turbulence modeling equations often result in stiff numerical systems, and care must be taken to devise schemes that preserve positivity of the turbulence quantities at all stages of the solution process. A common practice is to discretize the convective terms using a first-order upwind strategy, from which a positive scheme that obeys a maximum principle can be obtained.

3.4 *Data Structures*

A brief discussion on data structures is useful because the success of most discretization methods ultimately depends on how efficiently they may be implemented. Traditionally, finite-element methods have relied on element-based data structures, where for each element of the mesh a list of the forming vertex addresses is stored (i.e. four vertices for tetrahedra, eight vertices for hexahedra, etc). For many fluid dynamics problems, the discretizations, which are typically thought of as summations of fluxes, can be implemented more effectively using an edge-based data structure (Barth 1992; Mavriplis 1992, 1995c; Peraire et al 1992; Luo et al 1993). For a vertex-based scheme, this corresponds to storing, for each edge of the mesh, the addresses of the two vertices on either end of the edge. For a cell-centered scheme, the relevant entity is the dual edge that joins two neighboring cell centroids and pierces the face common to these two cells. The discretization may be evaluated by computing a flux on each edge, which is then added to and subtracted from the respective control volumes on

each end of the edge. In order to compute this flux, a face area must be stored for each edge, which corresponds to the area of the dual control-volume face associated with the mesh edge in the vertex-based scheme, and to the area of the cell face pierced by the dual edge in the cell-centered scheme. The use of edge-based data structures results in lower memory overheads and increased computational throughputs because redundant computations are eliminated and the gather-scatter required for vectorization in supercomputers is minimized. Furthermore, because sets of edges can be used as building blocks for arbitrarily shaped elements, hybrid meshes with mixed element types may be handled by a single edge-based data structure, at least for inviscid flows.

For viscous flows, the Galerkin finite-element discretization of the diffusion terms on simplicial meshes results in a nearest-neighbor stencil and thus may be implemented using an edge-based data structure (Mavriplis 1995c). However, this is not the case for nonsimplicial meshes, since the resulting stencils involve vertices that are not connected to the center vertex by a mesh edge (such as diagonally opposed vertices in hexahedral elements). In these situations, the element-based data structure must be retained (Braaten & Connell 1996). An alternative is to resort to the thin-layer approximation of the viscous terms on nonsimplicial meshes, which can be implemented exclusively along edges (Mavriplis & Venkatakrishnan 1995b). The limitations of this approach are obvious, although it is justifiable for highly stretched prismatic or hexahedral meshes, where streamwise resolution has been sacrificed for efficiency.

An interesting property of the edge-based data structure is that it can provide an interpretation of the discrete operator as a sparse matrix. For nearest-neighbor stencil discretizations, all points in the stencil are joined to the center point by a mesh edge. The discretization operator can be written as a sparse matrix, where each nonzero entry in the matrix corresponds to a stencil coefficient or edge of the mesh. For systems of equations, the edges correspond to nonzero block matrix entries in the large sparse matrix. This interpretation has implications for the implementation of implicit and algebraic multigrid solution schemes (Venkatakrishnan & Mavriplis 1993, Mavriplis & Venkatakrishnan 1995b).

One of the disadvantages of the edge-based data structure is that it requires a preprocessing operation to extract a unique list of edges from the list of mesh elements and to compute the associated edge coefficients. For unsteady flows with dynamic meshes, this preprocessing must be performed every time the mesh is altered, although this may be done locally. Additionally, for dynamic grid cases, the element data structures are generally required for performing mesh motion or adaptation, since edge lists represent a lower-level description of the mesh.

4. SOLUTION TECHNIQUES

The discretization of the spatial terms in the governing equations results in a large coupled set of ordinary differential equations of the form

$$\frac{d(VM\mathrm{w})}{dt} + \mathbf{R}(\mathrm{w}) = 0 \tag{2}$$

where V represents the local control volume, M the mass matrix, and $\mathbf{R}(\mathrm{w})$ the spatially discretized terms. For a vertex-based scheme, the mass matrix relates the average value of a control volume to the values at the vertices of the mesh. Both V and M are constants for static meshes, and can therefore be taken outside of the time derivative. For steady-state cases, these equations must be integrated in time towards $t \to \infty$, where the time derivatives become vanishingly small. Since time accuracy is not a concern in such cases, the mass matrix may be lumped—i.e. replaced by the identity matrix—and each equation may be advanced with the maximum permissible time step, as determined from local stability considerations.

4.1 *Explicit Schemes*

A simple time integration scheme is obtained by replacing the time derivative by a simple forward difference and evaluating the residual at the current time level:

$$V\frac{\mathrm{w}^{n+1} - \mathrm{w}^{n}}{\Delta}t + \mathbf{R}(\mathrm{w}^{n}) = 0 \tag{3}$$

This corresponds to a single-stage explicit scheme, since updates are obtained from one evaluation of currently available quantities. Like Runge-Kutta schemes for ordinary differential equations, multistage time-stepping schemes (which are required with higher-order discretizations for stability reasons) involve the combination of updates obtained at multiple explicitly evaluated intermediate states.

The stage coefficients of these schemes can be optimized either to provide large time steps at the expense of time accuracy or to enhance high-frequency damping properties of the scheme, which is required in the context of a multigrid algorithm (Jameson et al 1981, van Leer et al 1989).

These schemes are extremely simple to implement and require little additional computer storage. However, stability considerations restrict the maximum permissible time step to values proportional to the local cell size. Thus, finer meshes lead to smaller time steps, which in turn lead to larger solution times. For very fine meshes, the convergence of explicit schemes becomes unacceptably slow, and more sophisticated solution strategies must be adopted.

4.2 *Implicit Schemes*

An implicit scheme is obtained by evaluating the spatial residual terms at the new time level $n+1$. Since these quantities are not known explicitly, a linearization must be performed about the current time level:

$$\left(\frac{V}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathrm{w}}\right)\Delta \mathrm{w} = -\mathbf{R}(\mathrm{w}^{\mathrm{n}}) \tag{4}$$

$\frac{\partial \mathbf{R}}{\partial \mathrm{w}}$ represents the Jacobian and constitutes a large sparse matrix. At each time step, the above linear system must be solved for the corrections $\Delta \mathrm{w} = (\mathrm{w}^{n+1} - \mathrm{w}^{n})$ from which the flow variables can be updated.

Implicit schemes may be classified by the degree to which the true Jacobian matrix is approximated. If an exact linearization is employed, the method is unconditionally stable and reduces to Newton's method for $\Delta t \to \infty$. Although the quadratic convergence properties of Newton's method generally produce solutions in a very small number of time steps (often of the order of 10) (Venkatakrishnan & Barth 1989, Barth & Linton 1995, Nielsen et al 1995), each time step requires the inversion of a large sparse matrix, which becomes prohibitively expensive in terms of storage and CPU-time for fine meshes.

A common simplification is to replace the exact linearization with one based on a first-order discretization (Struijs et al 1989, Fezoui & Stoufflet 1989, Venkatakrishnan & Mavriplis 1993, Anderson et al 1995). While this considerably reduces the storage requirements of the linear system and improves its condition number, it precludes attaining quadratic convergence rates, owing to the use of an inexact linearization. This in turn favors the use of iterative methods to solve the linear system, which can be used to converge the system only partially, because exact inversion of the Jacobian is no longer beneficial, owing to the mismatch between Jacobian and residual. However, rigorous criteria for determining the optimal level of convergence of the linear system at each time step have yet to be determined. Simple iterative schemes such as Jacobi and Gauss-Seidel schemes have been utilized successfully in the context of implicit schemes with first-order linearizations (Struijs et al 1989, Fezoui & Stoufflet 1989, Batina 1993, Anderson et al 1995). However, since these are still local techniques, their convergence degrades with grid size. More sophisticated techniques such as preconditioned GMRES methods provide enhanced convergence rates, particularly when applied with powerful preconditioners such as Incomplete Lower Upper (ILU) factorization methods, which provide more global information for the solution of the linear system (Venkatakrishnan & Mavriplis 1993, Barth & Linton 1995, Anderson et al 1995).

Although first-order linearization methods require substantially less memory overheads than exact linearization methods, their memory requirements are still nontrivial. For example, on a three-dimensional tetrahedral grid, the first-order

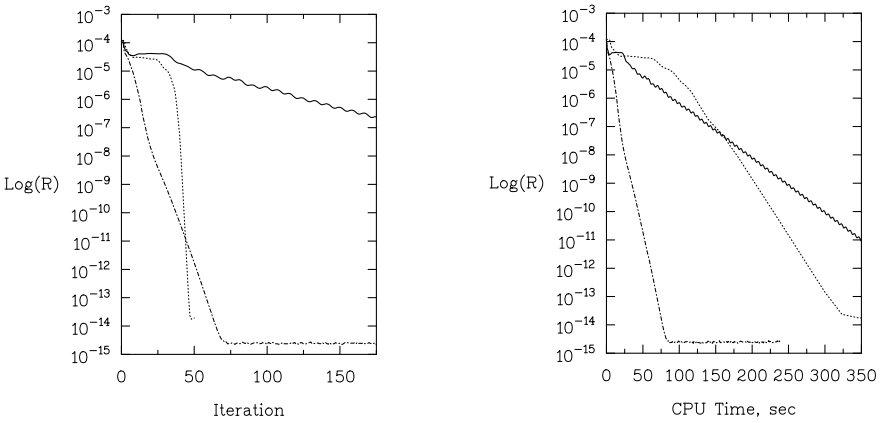

Figure 9 Convergence rates of the ILU-preconditioned Newton-Krylov method, versus Gauss-Siedel implicit method and multigrid method using implicit Gauss-Seidel as a multilevel smoother in terms of iteration count, and CPU-time, for two-dimensional inviscid subsonic flow over NACA 0012 airfoil. *Solid line:* Gauss Siedel; *dotted line*: ILU preconditioned Newton-Krylov; *dashed line:* multigrid method (using Gauss Siedel as smoother). (Reproduced from Nielsen et al 1995 with permission.)

Jacobian of a vertex-based scheme requires on the order of 350 storage locations per vertex, over three times the number required for the implementation of an explicit scheme. An ILU preconditioning strategy can require an equivalent additional amount of memory. Since GMRES methods only require the evaluation of matrix vector products of the form $\frac{\partial \mathbf{R}}{\partial \mathrm{w}} \cdot \Delta \mathrm{w}$, it is possible to forgo the storage of the Jacobian, and evaluate the Jacobian vector product directly by finite-difference techniques:

$$\frac{\partial \mathbf{R}}{\partial \mathrm{w}} \cdot \Delta w = \frac{\mathbf{R}(\mathrm{w} + \epsilon \Delta \mathrm{w}) - \mathbf{R}(\mathrm{w})}{\epsilon} \tag{5}$$

where ϵ represents a small parameter. This requires multiple residual evaluations (one for each matrix vector product) and represents a classic tradeoff between storage and computation, particularly for expensive nonlinear residual constructions. However, it also permits the use of the more accurate second-order linearization in the implicit scheme, by using the same second-order residual in the finite difference evaluation as in the right-hand side of equation (4), as is done in the so-called Newton-Krylov methods (Cai et al 1994, Barth & Linton 1995, Nielsen et al 1995). Figure 9 compares the convergence rates of a Newton-Krylov method that employs ILU preconditioning with convergence of a Gauss-Seidel iterative method applied to a first-order linearization, as well as with the convergence obtained by a multigrid scheme using the same

Gauss-Seidel solver as a multilevel smoother, as described in section 4.3. The depicted case involves the solution of two-dimensional subsonic inviscid flow over a NACA 0012 airfoil on a mesh of 8578 points, using a vertex-based second-order upwind scheme (Nielsen et al 1995). The quadratic convergence property of the Newton-Krylov approach is evident, which achieves a very rapid asymptotic convergence rate in terms of number of iterations. When compared in terms of overall CPU-time, the Newton-Krylov scheme, although still superior to the Gauss-Seidel implicit method, is overtaken by the multigrid scheme, due to the expense of each Newton-Krylov iteration.

Additional improvements to Newton-Krylov methods can be achieved through continuation techniques, which provide better initial guesses (for example by coarse to fine mesh sequencing), in order to reduce the initial phase of poor convergence of these schemes. However, one of the drawbacks of these methods is that they still require powerful preconditioners in order to be effective, and the best known preconditioners are generally matrix based (such as ILU), which entail storage requirements similar to those of the first-order linearization. A compromise, with obvious limitations, is to use a weaker but low-storage preconditioner, such as block-diagonal with a Newton-Krylov method (Shakib et al 1989). For upwind discretizations based on reconstruction techniques, higher-order Jacobian vector products can be formed by multiplying the first-order Jacobian with a higher-order reconstructed vector, as pointed out by Barth (Barth & Linton 1995). Although the method requires the storage of the first-order Jacobian, it is exact and, unlike finite-difference techniques, can also be applied directly to the solution of the adjoint equations, which is required in particular formulations of the design optimization problem.

Additional approximations to the exact Jacobian can be employed to further reduce the memory requirements of implicit schemes. For example, one may choose to group subregions of the mesh together and only retain the terms of the Jacobian that pertain to coupling within each region (Chan & Mathew 1994). These regions may be determined by a variety of methods and may exhibit varying degrees of overlap. These approximations may be applied to the Jacobian itself but are most often applied to the matrix-based preconditioners operating on the true Jacobian (Venkatakrishnan 1994, Barth & Linton 1995, Chan & Mathew 1994). Alternatively, the regions may be reduced to lines through the mesh joining neighboring vertices, as in the method of linelets (Hassan et al 1990, Martin & Löhner 1992), which attempts to approximate the Alternate Direction Implicit (ADI) scheme of structured grids. On sequential machines, the regions or lines can be processed individually, thus reducing the maximum memory requirements to that of the largest region or line (although on parallel machines no savings are realized if each region is assigned to an

individual processor). The numerical coupling between regions or lines is lost in these approaches, a loss that in turn degrades overall convergence, as the number of regions increases. However, this degradation can be minimized by judicious choices of the regions based on the character of the problem, a technique that has not yet been fully exploited.

If these regions are reduced to individual grid points (i.e. all implicit coupling between grid points is discarded), only the diagonal block matrices of the original Jacobian are retained, and the point-implicit method is obtained (Hassan et al 1990, Thareja et al 1988). These terms represent the coupling between the various fluid dynamic equations at a grid point. While point-implicit methods can offer increases in efficiency over regular explicit schemes, they are also plagued by slow convergence for fine grids. Point-implicit methods are sometimes viewed as preconditioning techniques for explicit schemes, where the point-implicit matrix is the preconditioner (Riemslagh & Dick 1993, Morano & Dervieux 1993, Ollivier-Gooch 1995, Pierce & Giles 1996).

4.3 *Multigrid Methods*

Multigrid methods are effective techniques that can deliver fast convergence rates with minimal memory overheads. The basic idea of a multigrid method is to accelerate the solution of a set of fine grid equations by time stepping on a sequence of fine and coarse grids using a simple explicit scheme. Multigrid methods capitalize on the effectiveness of explicit methods in reducing high-frequency error modes. A high-frequency error can be thought of as a solution component that oscillates between neighboring grid points. While explicit methods are effective at eliminating such errors, they are relatively ineffective at reducing low-frequency or global error components. This is to be expected, because explicit schemes rely exclusively on local information supplied through the residual stencil.

The multigrid strategy consists in explicitly time stepping the fine grid equations until the high-frequency errors have been annihilated. The solution is then transferred to a coarser grid, where the remaining low-frequency errors now manifest themselves as high-frequency errors and can be effectively damped by the same explicit method. The process is repeated recursively through a complete sequence of grids, where each level is responsible for a particular bandwidth of errors. When the coarsest grid of the sequence is reached, the corrections are interpolated back to each successively finer grid. This entire multigrid cycle is then repeated until overall convergence is achieved.

In the context of unstructured meshes, the main issue involves the generation of the coarse levels for a multigrid scheme. Given a fine unstructured mesh, there is no simple procedure for constructing uniformly coarser levels, as in the structured-grid case.

If the fine mesh is the product of an adaptive meshing algorithm using element subdivision, then the nested parent meshes are natural candidates for coarse multigrid levels (Parthasarathy & Kallinderis 1994, Braaten & Connell 1996). Although this approach can be employed effectively when multiple adaptive levels are present, the nested multigrid strategy places conflicting demands on the multilevel construction. On the one hand, a very coarse mesh is desired for rapid convergence, while a high-quality fine mesh is required for solution accuracy. However, repeated subdivision of a very coarse initial mesh often results in poor quality fine meshes, thus limiting the number of levels that may be employed. The lack of flexibility in handling problems on grids of arbitrary origin represents an additional shortcoming of this approach.

An alternative approach to unstructured multigrid methods is to generate a sequence of completely independent coarse and fine meshes and use linear interpolation to transfer variables back and forth between the various meshes of the sequence, within a multigrid cycle (Mavriplis 1992, 1995c; Peráire et al 1992; Leclercq 1990; Morano & Dervieux 1993; Riemslagh & Dick 1993). The meshes may be generated using any feasible grid generation technique and will generally be nonnested, and may even not contain any common points. The only requirement is that they conform to the same domain boundaries. This technique is more flexible than the nested subdivision approach, since the fine and coarse meshes are not constrained, and may be optimized independently for accuracy and speed of convergence respectively. Furthermore, this approach can be applied to a problem with a prespecified fine mesh. The intergrid transfer operators can be determined in a preprocessing operation but require smart search techniques to determine the patterns between coarse and fine grid elements, which in principle may overlap randomly. Once these have been determined and stored, grid transfers are simply implemented as a weighted gather and scatter of data between coarse and fine grid arrays. Application of this technique to a three-dimensional inviscid flow is depicted in Figures 10 and 11. The fine grid contains 804,000 points, or 4.5 million elements, and the overset multigrid solver achieves a six-order reduction in the residual in approximately 100 multigrid cycles, which represents close to an order of magnitude savings over the single-grid scheme.

One of the drawbacks of this approach is that it still requires the user to generate multiple grids for a single solution, which is at best tedious. With this in mind, several efforts have been pursued to automate the process. Methods that remove selected mesh points, using graph-based algorithms, and retriangulate the remaining subset of vertices have been demonstrated as effective tools for generating coarse-level multigrid meshes (Guillard 1993, Chan & Smith 1993, Matheson & Tarjan 1994). However, for very complex geometries, the

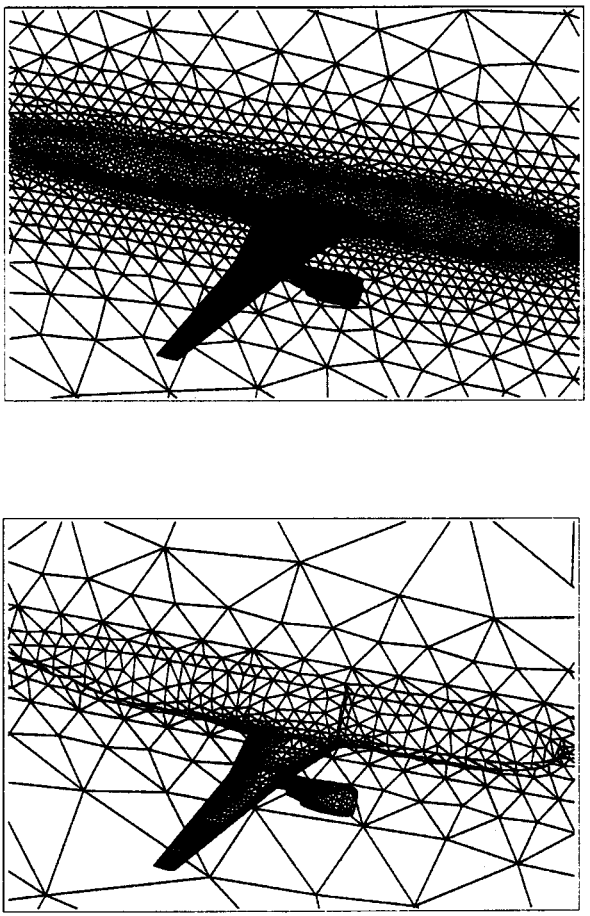

Figure 10 Two grids from the sequence of grids employed in the overset mesh multigrid scheme for the computation of transonic flow over an aircraft configuration.

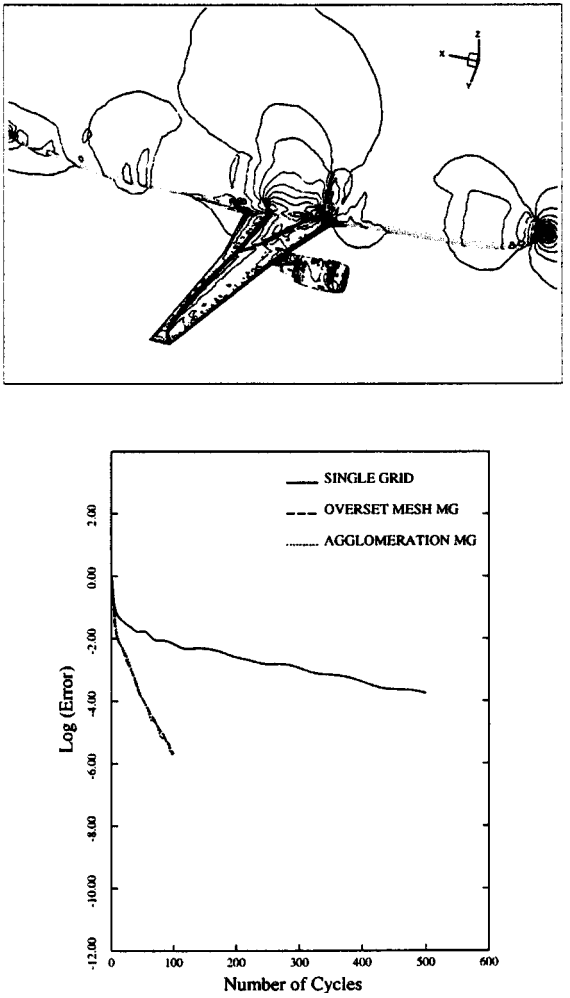

Figure 11 Computed solution displayed as Mach contours and observed convergence rates for the single-grid, overset-multigrid, and agglomeration-multigrid algorithms for the computation of transonic flow over an aircraft configuration.

generation of coarse meshes that conform to the original geometry either manually or automatically, becomes a difficult task.

An alternative is provided by agglomeration multigrid methods. Based on the control-volume formulation, these methods form coarser-level meshes by fusing or agglomerating fine-level control volumes together with their neighbors (Lallemand et al 1992, Smith 1990, Venkatakrishnan & Mavriplis 1995a). The resulting coarse mesh contains a smaller number of larger and more complex control volumes. The precise manner in which control volumes are agglomerated can be controlled through a graph-based algorithm similar to the vertex removal procedure described above. The flow solver must be modified to run on arbitrarily shaped control volumes on the coarse levels. For inviscid-flow control-volume formulations, this presents little difficulty, since the equations are generally discretized as fluxes over individual control-volume faces, which can be used to build contour integrals about arbitrarily shaped control volumes. Figure 12 depicts two coarse levels generated by the agglomeration algorithm for the same case discussed above in the context of the overset-mesh multigrid approach. The observed convergence rate of the agglomeration multigrid method is almost identical to that obtained with the overset-mesh method, as depicted in Figure 11.

For viscous flows, the discretization of diffusion terms on arbitrarily shaped control volumes is no longer straightforward. An algebraic interpretation of agglomeration multigrid provides a mechanism for dealing with equations sets that contain diffusion terms. Borrowing from the algebraic multigrid literature (Ruge & Stüben 1987), a coarse-grid operator may be constructed by projecting the fine-grid operator onto the space spanned by the coarse-grid basis functions. If I_h^H represents the restriction or fine- to coarse-grid transfer operator, I_H^h the prolongation or coarse- to fine-grid operator, and $\mathbf{A}$ the fine-grid discretization operator, then the so-called Galerkin coarse-grid operator is given by the sequence of operators $I_h^H \mathbf{A} I_H^h$. If $\mathbf{A}$ represents a finite-volume approximation for inviscid flow on the fine grid, and restriction and prolongation are taken as piecewise constant operators, then the Galerkin construction recovers the traditional coarse-grid finite-volume discretization discussed above. However, this strategy can be extended in a straightforward manner for other types of governing equations, such as the Navier-Stokes equations and field-equation turbulent models. This approach may be interpreted algebraically as summing the constituent fine-grid equations within an agglomerated coarse-grid cell to form a smaller number of coarse-grid equations. This algebraic interpretation allows one to decouple the geometry from the solution process, thus discarding any notion of coarse-level spatial discretizations, which enhances the robustness of the solution process for complicated geometries. A solution of viscous turbulent flow over a three-dimensional wing configuration using the agglomeration

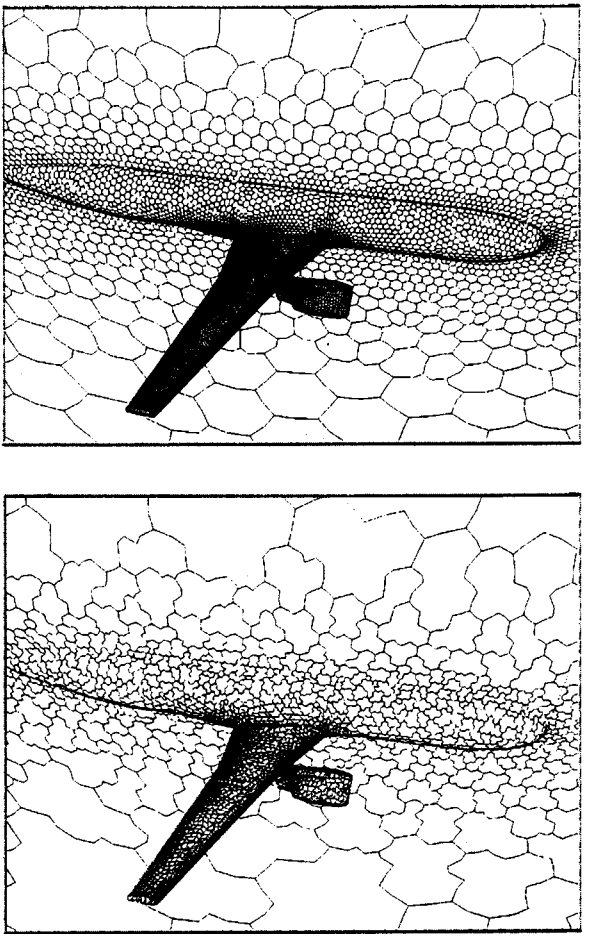

Figure 12 Two coarse agglomerated levels from the sequence of levels employed by the agglomeration-multigrid method for the computation of transonic flow over an aircraft configuration.

multigrid approach is depicted in Figures 13 and 14. The convergence rate is shown to be relatively independent of grid size by comparing the convergence history on a coarse grid of 300,000 points with the the convergence obtained on a finer grid of 2.4 million points (Mavriplis & Venkatakrishnan 1995a).

In general, the multigrid convergence rates achieved for viscous flow cases are substantially slower than those achieved for inviscid cases. This can be surmised to some degree by comparing the convergence plots of Figures 11 and 14. The slower convergence in the viscous flow cases is principally due to the anisotropic stretching of the mesh in the boundary-layer and wake regions. Unstructured multigrid methods offer the possibility of designing schemes that are insensitive to anisotropic effects. The graph-based agglomeration or vertex-coarsening algorithms described above can be modified to merge only those neighboring control volumes or delete only those neighboring vertices that are the most strongly coupled to the current point. By coarsening or agglomerating only in the direction of strongest coupling, as determined by the magnitude of the coefficients in the fine-grid discrete equations, rather than eliminating all possible neighbors, coarse levels that are optimal for the problem at hand can be generated. In the case of a stretched-boundary-layer grid, this reduces to coarsening across the boundary layer, or to semicoarsening, which has been used effectively in the structured-grid context to remove anisotropy-induced stiffness (Mulder 1989, Pierce & Giles 1996). Although these techniques are in their infancy (Morano et al 1995, Raw 1996), they offer the possibility of automatically accounting for grid-induced as well as algebraically induced stiffness, since they operate at the equation level.

An alternate strategy for improving the convergence of multigrid methods is to employ a stronger smoother as the base grid solver. Any of the implicit methods described above may be substituted for an explicit method. Of course, most implicit methods incur additional overheads, but the benefits can justify these additional requirements. Point-implicit methods (Morano & Dervieux 1993, Riemslagh & Dick 1993, Ollivier-Gooch 1995), Gauss-Seidel methods (Anderson 1994), and subregion methods (Raw 1996, Chan & Mathew 1994) have all been demonstrated as effective multigrid solvers. There is an obvious duality between the construction of subregions for implicit methods and coarsening strategies for multigrid methods that should be exploited in the future to better couple locally implicit strategies with multigrid methods.

The above discussion on multigrid methods centers around the Full Approximation Storage (FAS) technique, where multigrid is applied directly to the nonlinear form of the governing equations. Multigrid may also be employed as a solver for the linear system arising at each time step of an implicit scheme, or even as a preconditioner for an iterative solution strategy such as GMRES applied to the linearized implicit system (Chan & Smith 1993, Raw 1996).

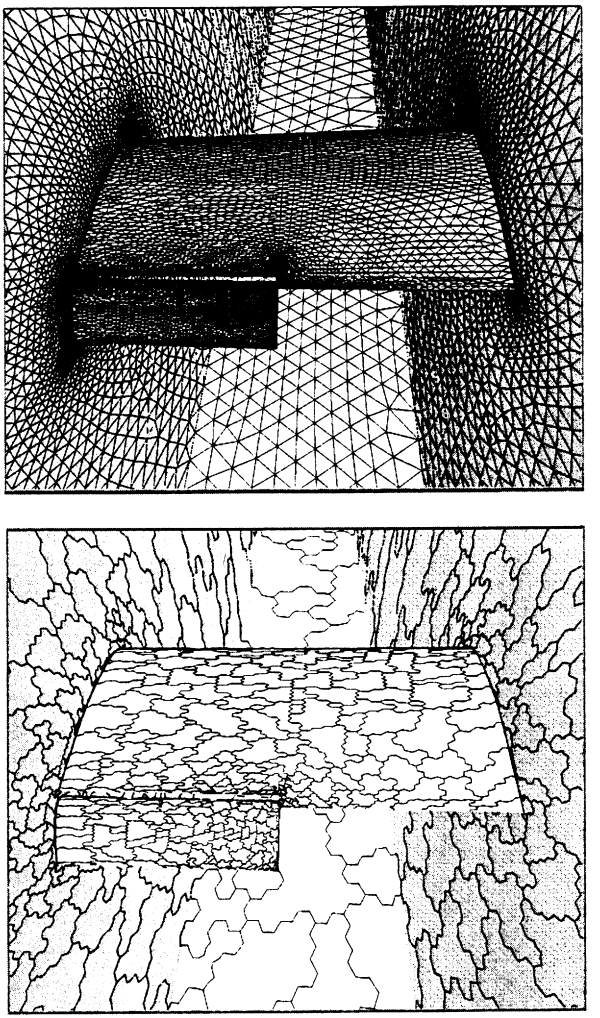

Figure 13 Initial 300,000-point grid and coarse agglomerated level employed in the agglomeration-multigrid scheme for the computation of viscous turbulent flow over a wing configuration.

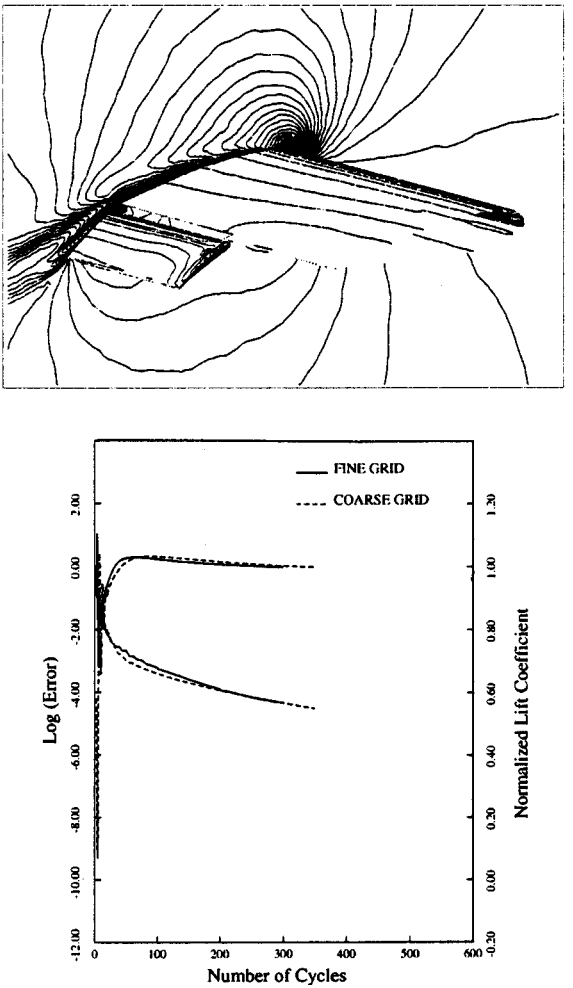

Figure 14 Computed solution displayed as Mach contours on the wall and pressure contours on the wing surface for fine 2.4 million–point (13.6 million–tetrahedra) grid and observed convergence rates of agglomeration-multigrid algorithm on 300,000-point grid and 2.4 million–point grid, demonstrating grid-independent convergence rates.

While these strategies may yield desirable robustness characteristics, they are plagued by the high memory requirements of the implicit linearization, which are avoided in the FAS approach.

5. CONCLUSIONS

An overview of the state-of-the-art in the areas of mesh generation, adaptivity, discretizations, and solution strategies for unstructured meshes has been given. An attempt has been made to outline the merits and disadvantages of the various approaches in these areas. Other areas such as solution techniques for unsteady flows and parallel implementations have not been addressed owing to space limitations. Excellent survey papers in these areas, as well as more in-depth reviews of some of the items covered in this paper, can be found in the literature (Thompson & Weatherill 1993; Baker 1996; Venkatakrishnan 1995, 1996; Special course 1992; Mavriplis 1995a,b).

Although unstructured mesh techniques have undergone rapid development in the last several years, much improvement is still needed. Three-dimensional inviscid analyses about complex geometries using unstructured meshes can be performed routinely with present technology. However, viscous flow analyses using the Reynolds-averaged Navier-Stokes equations are severely hindered for large cases by seemingly insatiable CPU-time and memory requirements, even for steady-state calculations. In particular, most cases are limited by hardware memory costs, which have not decreased as rapidly as the cost of CPU-cycles over the last five years. Further advances must be sought in the areas of higher-order schemes, efficient and robust solution algorithms, improved mesh generation strategies (particularly for highly stretched meshes and meshes of mixed-element types), and better adaptive criteria to enable routine use of adaptive meshing and the generation of solutions with guaranteed error bounds.

ACKNOWLEDGMENTS

Dr. V. Venkatakrishnan is acknowledged for his helpful comments in preparing this paper. The author is also grateful to Dr. K. W. Anderson, Dr. S. Pirzadeh, and E. Nielsen for providing figures for this paper.

Literature Cited

Aftosmis M, Gaitonde D, Tavares TS. 1994. On the accuracy, stability and monotonicity of various reconstruction algorithms for unstructured meshes. *AIAA Pap. 94-0415*

Anderson WK. 1994. A grid generation and flow solution method for the Euler equa-

tions on unstructured grids. *J. Comput. Phys.* 110(1):23–38

Anderson WK, Rausch R, Bonhaus D. 1995. Implicit multigrid algorithms for incompressible turbulent flows on unstructured grids. *Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1740-CP*

Babushka I, Aziz AK. 1976. On the angle condition in the finite-element method. *SIAM J. Numer. Anal.* 13(6)

Baker TJ. 1987. Three dimensional mesh generation by triangulation of arbitrary point sets. *AIAA Pap. 87-1124*

Baker TJ. 1991. Unstructured meshes and surface fidelity for complex shapes. *Proc. AIAA CFD Conf., 10th, Honolulu*, pp. 714–25. *AIAA Pap. 91-1591-CP*

Baker TJ. 1996. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elem. Anal. Des.* In press

Baldwin BS, Barth TJ. 1991. A one-equation turbulence transport model for high Reynolds number wall-bounded flows. *AIAA Pap. 91-0610*

Barth TJ. 1991a. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. *AIAA Pap. 91-0721*

Barth TJ. 1991b. A three-dimensional upwind Euler solver for unstructured meshes. *AIAA Pap. 91-1548-CP*

Barth TJ. 1992. Aspects of unstructured grids and finite-element volume solvers for the Euler and Navier-Stokes equations. *von Karman Inst. Lect. Ser., AGARD Publ. R-787*

Barth TJ, Jespersen DC. 1989. The design and application of upwind schemes on unstructured meshes. *AIAA Pap. 89-0366*

Barth TJ, Linton SW. 1995. An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. *AIAA Pap. 95-0221*

Batina JT. 1993. Implicit upwind solution algorithms for three-dimensional unstructured meshes. *AIAA J.* 31(5):801–5

Baum JD, Luo H, Löhner R. 1994. A new ALE adaptive unstructured methodology for the simulation of moving bodies. *AIAA Pap. 94-0414*

Blacker TD, Stephenson MB. 1991. Paving: a new approach to automated quadrilateral mesh generation. *Int. J. Numer. Methods Eng.* 32:811–47

Boris JP, Book DL. 1973. Flux corrected transport, 1 SHASTA, a fluid transport algorithm that works. *J. Comput. Phys.* 11:38–69

Bowyer A. 1981. Computing Dirichlet tessalations. *Comput. J.* 24(2):162–66

Braaten ME, Connell SD. 1996. Three dimensional unstructured adaptive multigrid scheme for the Navier-Stokes equations. *AIAA J.* 34(2):281–90

Cai XC, Gropp WD, Keyes DE, Tidriri MD. 1994. Newton-Krylov-Schwarz methods in CFD. *Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations. Notes on Numerical Fluid Mechanics*, Vol. 47:17–30. Braunschweig/Wiesbaden: Vieweg

Castro-Diaz MJ, Hecht F, Mohammadi B. 1995. Anisotropic unstructured mesh adaptation for flow simulations. *Int. Mesh Roundtable, 4th, Albuquerque, NM*, pp. 73–85

Chan TF, Mathew TP. 1994. Domain decomposition algorithms. *Acta Numer.* pp. 61–143

Chan TF, Smith B. 1993. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. *UCLA CAM Rep. 93-42*, Dep. Math., Univ. Calif., Los Angeles

Chew LP. 1989. Constrained Delaunay triangulations. *Algorithmica* 4:97–108

Connell SD, Braaten ME. 1995. Semi-structured mesh generation for three-dimensional Navier-Stokes calculations. *AIAA J.* 33(6):1017–24

Deconinck H, Hirsch Ch, Peuteman J. 1986. Characteristic decomposition methods for the multidimensional Euler equations. *Lect. Notes Phys.* 264:216–22

Desideri JA, Dervieux A. 1988. Compressible flow solvers using unstructured grids. *VKI Lect. Ser.* 1988-05:1–115

Edelsbrunner H, Tan TS, Waupotitsch R. 1992. An $0(N^2 \log N)$ time algorithm for the min-max angle triangulation. *SIAM J. Sci. Stat. Comput.* 13:994–1008

Fezoui L, Stoufflet B. 1989. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J. Comput. Phys.* 84:174–206

Frink NT. 1992. Upwind scheme for solving the Euler equations on unstructured tetrahedral meshes. *AIAA J.* 30(1):70–77

Frink NT. 1994. Recent progress toward a three-dimensional unstructured Navier-Stokes flow solver. *AIAA Pap. 94-0061*

George PL, Hecht F, Saltel E. 1991. Automatic mesh generator with specified boundary. *Comput. Methods Appl. Mech. Eng.* 33:975–95

Guillard H. 1993. Node nested multigrid with Delaunay coarsening. *INRIA Rep. No. 1898*

Gumbert C, Lohner R, Parikh P, Pirzadeh S. 1989. A package for unstructured grid generation and finite element flow solvers. *AIAA Pap. 89-2175*

Harten A. 1984. On a class of high-resolution total-variation-stable finite difference schemes. *SIAM J. Numer. Anal.* 21(1):1–23

Hassan O, Morgan K, Peraire J. 1989. An adaptive implicit/explicit finite element scheme

for compressible high speed flows. *AIAA Pap. 89-0363*

Hassan O, Morgan K, Peraire J. 1990. An implicit finite element method for high speed flows. *AIAA Pap. 90-0402*

Holmes DG, Snyder DD. 1988. The generation of unstructured meshes using Delaunay triangulation. *Numer. Grid Gener. Comput. Fluid Mech. Proc. Int. Conf. Numer. Grid Gener. Comput. Fluid Dyn., 2nd, Miami,* ed. S Sengupta, J Hauser, PR Eisman, JF Thompson Swansea, UK: Pineridge

Hughes TJR. 1987. Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations. *Int. J. Numer. Methods Fluids* 7:1261–75

Hughes TJR, Brooks A. 1979. A multidimensional upwind scheme with no crosswind diffusion. In *Finite Element Methods for Convection Dominated Flows.* New York: ASME

Hughes TJR, Franca LP, Hulbert GM. 1989. A new finite element formulation for computational fluid dynamics. VIII. Galerkin/least-squares method for advective-diffusive systems. *Comput. Methods Appl. Mech. Eng.* 73:173–89

Hughes TJR, Mallet M. 1986. A new finite element formulation for CFD. IV. A discontinuity-capturing operator for multidimensional advective-diffusive systems. *Comput. Methods Appl. Mech. Eng.* 58:329–36

Jameson A. 1994. Analysis and design of numerical schemes for gas dynamics. I. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence. *Int. J. Comput. Fluid Dyn.* 4:171–218

Jameson A, Baker TJ, Weatherill NP. 1986. Calculation of inviscid transonic flow over a complete aircraft. *AIAA Pap. 86-0103*

Jameson A, Schmidt W, Turkel E. 1981. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. *AIAA Pap. 81-1259*

Joe B. 1989. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.* 10:718–41

Kennon SR, Meyering JM, Berry CW, Oden JT. 1992. Geometry based Delaunay tetrahedralization and mesh movement strategies for multi-body CFD. *AIAA Pap. 92-4575*

Lallemand M, Steve H, Dervieux A. 1992. Unstructured multigridding by volume agglomeration: current status. *Comput. Fluids* 21(3):397–433

Lawson CL. 1972. Transforming triangulations. *Discrete Math.* 3:365–72

Lawson CL. 1977. Software for C^1 surface interpolation. In *Mathematical Software III,* ed. JD Rice, pp. 161–95. New York: Academic

Leclercq MP. 1990. *Resolution des equations d'Euler par des methods multigrilles conditions aux limites en regime hypersonique.* PhD thesis. Dep. Appl. Math, Univ. de Saint-Etienne

Lo SH. 1985. A new mesh generation scheme for arbitrary planar domains. *Int. J. Numer. Methods Eng.* 21:1403–26

Löhner R. 1993. Matching semi-structured and unstructured grids for Navier-Stokes calculations. *AIAA Pap. 93-3348*

Löhner R, Baum JD. 1992. Adaptive h-refinement on 3D unstructured grids for transient problems. *Int. J. Numer. Methods Fluids* 14:1407–19

Löhner R, Morgan K, Peraire J, Vahdati M. 1987. Finite element flux-corrected transport (FCT) for the Euler and Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.* 7:1093–1109

Luo H, Baum JD, Löhner R, Cabello J. 1993. Adaptive edge-based finite element schemes for the Euler and Navier-Stokes equations on unstructured grids. *AIAA Pap. 93-0336*

Marcum DL. 1995. Generation of unstructured grids for viscous flow applications. *AIAA Pap. 95-0212*

Marcum DL, Weatherill NP. 1994. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Pap. 94-1926*

Martin D, Lohner R. 1992. An implicit linelet-based solver for incompressible flows. *AIAA Pap. 92-0668*

Matheson L, Tarjan R. 1994. Unstructured multigrid strategies on massively parallel computers: a case for integrated design. *Hawaii. Int. Conf. Syst. Sci., 27th,* 2:169–78. New York: IEEE

Mavriplis DJ. 1987. *Solution of the two-dimensional Euler equations on unstructured triangular meshes.* PhD thesis. Dep. Mech. Aerospace Eng., Princeton Univ.

Mavriplis DJ. 1990a. Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA J.* 28(2):213–21

Mavriplis DJ. 1990b. Adaptive mesh generation for viscous flows using Delaunay triangulation. *J. Comput. Phys.* 90(2):271–91

Mavriplis DJ. 1991a. Algebraic turbulence modeling for unstructured and adaptive meshes. *AIAA J.* 29(12):2086–93

Mavriplis DJ. 1991b. Unstructured and adaptive mesh generation for high-Reynolds number viscous flows. *Proceedings of the International Conference on Numerical Grid Generation: Computational Fluid Dynamics and Related Fields, 3rd, Barcelona, Spain,* ed. AS Arcilla, J Hauser, PR Eisman, JF Thompson, pp. 79–92. New York: North-Holland

Mavriplis DJ. 1992. Three-dimensional multigrid for the Euler equations. *AIAA J.* 30(7):1753–61

Mavriplis DJ. 1995a. Multigrid techniques for unstructured meshes. *VKI Lect. Ser. Comput. Fluid Dyn., 26th, VKI-LS 1995-02*

Mavriplis DJ. 1995b. A three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. *AIAA J.* 33(3):445–53

Mavriplis DJ. 1995c. Unstructured mesh generation and adaptivity. *VKI Lect. Ser. Comput. Fluid Dyn., 26th, VKI-LS 1995-02*

Mavriplis DJ, Venkatakrishnan V. 1994. Agglomeration multigrid for two-dimensional viscous flows. *Comput. Fluids* 24(5):553-70

Mavriplis DJ, Venkatakrishnan V. 1995a. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. *AIAA Pap. 95-0345*

Mavriplis DJ, Venkatakrishnan V. 1995b. A unified multigrid solver for the Navier-Stokes equations on mixed element meshes. *AIAA Pap. 95-1666*

Merriam ML. 1991. An efficient advancing-front algorithm for Delaunay triangulation. *AIAA Pap. 91-0792*

Morano E, Dervieux A. 1993. Looking for O(N) Navier-Stokes solutions on non-structured meshes. *Copper Mountain Conf. Multigrid Methods, 6th*, pp. 449–64. *NASA CP-3224*

Morano E, Mavriplis DJ, Venkatakrishnan V. 1995. Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems. *ICASE Rep. No. 95-34.* ICASE, NASA Langley Res. Cent., Hampton, VA. *SIAM J. Sci. Stat. Comput.* Submitted

Mortenson ME. 1985. *Geometric Modeling.* New York: Wiley & Sons

Mueller JD, Roe PL, Deconinck H. 1992. A frontal approach for node generation in Delaunay triangulations. *VKI Lect. Ser., AGARD Publ. R-787*

Mulder WA. 1989. A new multigrid approach to convection problems. *J. Comput. Phys.* 83(2):303–23

Nakahashi K. 1987. FDM-FEM approach for viscous flow computations over multiple bodies. *AIAA Pap. 87-0604*

Ni RH. 1982. A multiple-grid scheme for solving the Euler equations. *AIAA J.* 20(11):1565–71

Nielsen EJ, Anderson WK, Walters RW, Keyes DE. 1995. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. *Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1733-CP*

Ollivier-Gooch CF. 1995. Towards problem-independent multigrid convergence rates for unstructured mesh methods. I. inviscid and laminar flows. *Proc. 6th Int. Symp. CFD, Lake Tahoe, NV*

Osher S. 1984. Riemann solvers, the entropy condition and difference approximations. *SIAM J. Numer. Anal.* 21(2):217–35

Paillere H, Deconinck H, Struijs R, Roe PL, Mesaros LM, Muller JD. 1993. Computations of compressible flows using fluctuation-splitting on triangular meshes. *AIAA Pap. 93-3301-CP*

Palmerio B. 1994. An attraction-repulsion mesh adaption model for flow solution on unstructured grids. *Comput. Fluids* 23(3):487-506

Parpia I, Michalak DJ. 1993. Grid-independent upwind scheme for multidimensional flows. *AIAA J.* 31(4):646–51

Parthasarathy V, Kallinderis Y. 1994. New multigrid approach for three-dimensional unstructured, adaptive grids. *AIAA J.* 32(5):956–63

Peraire J, Peirö J, Morgan K. 1992. A 3D finite-element multigrid solver for the Euler equations. *AIAA Pap. 92-0449*

Peraire J, Vehdati M, Morgan K, Zienkiewicz OC. 1987. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.* 72:449–66

Pierce N, Giles M. 1996. Preconditioning on stretched meshes. *AIAA Pap. 96-0889. J. Comput. Phys.* Submitted

Pirzadeh S. 1994a. Unstructured viscous grid generation by the advancing-layers method. *AIAA J.* 32(8):1735–37

Pirzadeh S. 1994b. Viscous unstructured three-dimensional grids by the advancing-layers method. *AIAA Pap. 94-0417*

Preparata FP, Shamos MI. 1985. *Computational Geometry, An Introduction.* Texts Monogr. Comput. Sci. New York: Springer-Verlag

Rausch RD, Batina JT, Yang HTY. 1992. Spatial adaptation of unstructured meshes for unsteady aerodynamic flow computations. *AIAA J.* 30(5):1243–51

Raw M. 1996. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. *AIAA Pap. 96-0297*

Rebay S. 1993. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer/Watson algorithm. *J. Comput. Phys.* 106(1):125–38

Riemslagh K, Dick E. 1993. A multigrid method for steady Euler equations on unstructured adaptive grids. See Morano & Dervieux 1993, pp. 527–42

Roe PL. 1981. Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.* 43(2):357–72

Roe PL. 1986. Discrete models for the numerical analysis of time-dependent multidi-

mensional gas dynamics. *J. Comput. Phys.* 63:458–76

Rostand P, Stoufflet B. 1988. TVD schemes to compute compressible viscous flows on unstructured meshes. *Notes on Numerical Fluid Mechanics*, Vol. 24:510–520. Braunschweig, Ger: Vieweg

Ruge JW, Stüben K. 1987. Algebraic multigrid. *Multigrid Methods, SIAM Frontiers in Applied Mathematics*, ed. SF McCormick, pp. 73–131. Philadelphia: SIAM

Shakib F, Hughes TJR, Johan Z. 1989. A multi-element group preconditioned GMRES algorithm for nonsymmetric problems arising in finite element analysis. *Comput. Methods Appl. Mech. Eng.* 87:415–56

Sidilkover D. 1994. A genuinely multidimensional upwind scheme and efficient multigrid solver for the compressible Euler equations. *ICASE Rep. No. 94-84*. ICASE, NASA Langley Res. Cent., Hampton, VA

Slack DC, Whitaker DL, Walters RW. 1994. Time integraton algorithms for the two-dimensional Euler equations on unstructured meshes. *AIAA J.* 32(6):1158–66

Smith WA. 1990. Multigrid solution of transonic flow on unstructured grids. *Recent Advances and Applications in Computational Fluid Dynamics*, pp. 93–103. *Proc. ASME Winter Annu. Meet.*, ed. O Baysal

Spalart PR, Allmaras SR. 1992. A one-equation turbulence model for aerodynamic flows. *AIAA Pap. 92-0439*

Special course on unstructured grid methods for advection dominated flows. 1992. *AGARD Publ. R-787*

Stoufflet B, Periaux J, Fezoui F, Dervieux A. 1987. Numerical simulation of 3-D hypersonic Euler flows around space vehicles using adapted finite-elements. *AIAA Pap. 87-0560*

Struijs R, Vankeirsbilck P, Deconinck D. 1989. An adaptive grid polygonal finite-element method for the compressible flow equations. *AIAA Pap. 89-1957-CP*

Tembulkar JM, Hanks BW. 1992. On generating quadrilateral elements from a triangular mesh. *Comput. Struct.* 42(4):665–67

Thareja RR, Stewart JR, Hassan O, Morgan K, Peraire J. 1988. A point implicit unstructured grid solver for the Euler and Navier-Stokes equations. *AIAA Pap. 88-0036*

Thompson JF, Weatherill NP. 1993. Aspects of numerical grid generation: Current science and art. *AIAA Pap. 93-3539-CP*

Vallet MG, Hecht F, Mantel B. 1991. Anisotropic control of mesh generation based upon a Voronoi type method. See Mavriplis 1991b, pp. 93–103

van Leer B, Tai CH, Powell KG. 1989. Design of optimally-smoothing multi-stage schemes for the Euler equations. *AIAA Pap. 89-1933*

Venkatakrishnan V. 1994. Parallel implicit unstructured grid Euler solvers. *AIAA J.* 32(10):1985–91

Venkatakrishnan V. 1995a. Implicit schemes and parallel computing in unstructured grid CFD. *VKI Lect. Ser. VKI-LS 1995-02*

Venkatakrishnan V. 1995b. On the accuracy of limiters and convergence to steady state solutions. *J. Comput. Phys.* 118:120–30

Venkatakrishnan V. 1996. A perspective on unstructured grid flow solvers. *AIAA J.* 34(3):533–47

Venkatkrishnan V, Barth TJ. 1989. Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes. *AIAA Pap. 89-0364*

Venkatakrishnan V, Mavriplis DJ. 1993. Implicit solvers for unstructured meshes. *J. Comput. Phys.* 105(1):83–91

Venkatakrishnan V, Mavriplis DJ. 1995a. Agglomeration multigrid for the three-dimensional Euler equations. *AIAA J.* 33(4):633–40

Venkatakrishnan V, Mavriplis DJ. 1995b. Implicit method for the computation of unsteady flows on unstructured grids. *Proc. AIAA CFD Conf., 12th, San Diego. AIAA Pap. 95-1705-CP*

Ward S, Kallinderis Y. 1993. Hybrid prismatic/tetrahedral grid generation for complex 3D geometries. *AIAA Pap. 93-0669*

Watson DF. 1981. Computing the n-dimensional Delaunay tessalation with application to Voronoi polytopes. *Comput. J.* 24(2):167–71

Weatherill NP. 1985. The generation of unstructured grids using Dirichlet tessalations. *Princeton Univ. Dep. Mech. Aerospace Eng. Rep. MAE 1715*

Weatherill NP, Hassan O, Marcum DL. 1993. Calculation of steady compressible flowfields with the finite-element method. *AIAA Pap. 93-0341*

Yerry MA, Shephard MS. 1984. Automatic three dimensional mesh generation by the modified-octree technique. *Int. J. Numer. Methods Eng.* 20:1965–90

Zhu JZ, Zienkiewicz OC, Hinton E, Wu J. 1991. A new approach to the development of automatic quadrilateral mesh generation. *Int. J. Numer. Methods Eng.* 32:849–66

Annual Review of Fluid Mechanics
Volume 29, 1997

CONTENTS