JOURNAL OF COMPUTATIONAL PHYSICS 90, 271–291 (1990)

Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation

DIMITRI J. MAVRIPLIS

Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665

Received January 9, 1989; revised June 26, 1989

A method for generating an unstructured triangular mesh in two dimensions, suitable for computing high Reynolds number flows over arbitrary configurations is presented. The method is based on a Delaunay triangulation, which is perfored in a locally stretched space, in order to obtain very high-aspect-ratio tiangles in the boundary layer and wake regions. It is shown how the method can be coupled with an unstructured Navier–Stokes solver to produce a solution-adaptive mesh generation procedure for viscous flows. © 1990 Academic Press, Inc.

1. INTRODUCTION

In recent years, the use of unstructured triangular and tetrahedral meshes in two and three dimensions has become more widespread for computational fluid dynamics problems. The advantages of unstructured meshes lie in their ability to deal with arbitrarily complex geometries, while providing a natural setting for the use of adaptive mesh enrichment techniques. On the other hand, the accuracy of unstructured mesh discretizations and the efficiency of unstructured mesh solvers have generally fallen short of their structured mesh counterparts. The appearance of more efficient and accurate Euler solvers for unstructured meshes [1–3], combined with the benefits of adaptive meshing and a general drive to problems of higher geometric complexity have combined to make unstructured meshes the preferred choice for many inviscid flow problems [4, 5]. However, few attempts at solving high Reynolds number viscous flows about complex configurations with unstructured meshes are known. The efficient solution of such flows requires the generation of highly stretched elements in the thin boundary-layer regions, where the resolution required in the direction normal to the layer can be several orders of magnitude greater than that in the streamwise direction. Present efforts at computing such flows have concentrated on the use of composite structured–unstructured meshes [6–8], where a thin structured triangular or quadrilateral mesh is placed in the boundary-layer regions, and an unstructured mesh is used to fill the remainder of the domain. The presence of a structured mesh in the regions of viscous flow can be advantageous, for it enables the use of proven and efficient

0021-9991/90 $3.00

structured-mesh flow solvers in these regions, where the behavior of the governing equations can become considerably stiff. This approach also enables a straightforward implementation of the often-used algebraic turbulence models for compressible turbulent flow calculations. However, it has the disadvantage of resulting in a nonautomatic grid generation procedure. The extent of the structured region and the definition of the structured–unstructured interface are somewhat arbitrary and must be prescribed interactively. This procedure quickly runs into difficulty as more complex geometries are involved, such as, for example, geometries consisting of multiple closely positioned bodies. In such cases, the structured meshes from two neighboring bodies may tend to overlap unless they are kept extremely thin, in which case it may prove difficult to obtain a smooth variation of elements between the stretched structured mesh and the isotropic unstructured mesh. Furthermore, the use of adaptive meshing in the regions of viscous flow tends to destroy the structure of the these thin local boundary meshes, thus complicating the implementation of the flow solver and turbulence model in these regions. In this work, it is proposed to employ an unstructured mesh of triangles throughout the entire domain. This approach requires extreme stretching of the unstructured mesh in the boundary-layer regions. However, it has the advantage of providing a completely automatic grid generation tool for arbitrary configurations, obviating the need for any human interaction, such as that required to define the structured–unstructured interface in the former approach. It also offers the possibility of obtaining a fully adaptive smoothly varying mesh throughout the viscous regions as well as in regions where the distance between neighboring boundaries may be smaller than the boundary-layer thickness. The recent success of fully explicit multigrid Navier–Stokes solvers [9, 10], which have also been applied to unstructured meshes [11], demonstrate the possibility of obtaining efficient solutions employing unstructured meshes throughout the viscous regions. While the use of unstructured meshes considerably complicates the implementation of algebraic turbulence models [12], it is felt that, for the complicated geometries for which these meshes are intended, a more general higher level turbulence model, such as a two-equation model, will be required due to the more complicated flow patterns which will prevail. Such models require the solution of two additional field equations which can be discretized in a straightforward manner on unstructured meshes. In fact, two-equation turbulence models have also been advocated for use on composite structured–unstructured meshes [8].

Of the various algorithms for generating triangular meshes in two dimensions, the advancing-front method [13] and the Delaunay triangulation method [14, 15] have been successfully applied to generate solution-adaptive evolving meshes. Sophisticated implementations of the advancing-front method incorporate directional stretching and refinement by successive remeshing according to stretching factors and directions obtained from a flow solution on a previous mesh. While this technique has proven valuable for inviscid flow calculations, the stretchings obtained are still several orders of magnitude smaller than that required for resolving viscous boundary-layer flows.

The first application of Delaunay triangulation to aerodynamic problems was performed by Weatherill [16]. The extension of this procedure to three dimensions has been pursued by Baker [17]. Delaunay triangulation is particularly well suited for adaptive meshing techniques, since it may be formulated as a sequential and local process. New points may be added and triangulated locally without the need for remeshing the domain in whole or in part. For a given set of data points, a Delaunay triangulation will produce the most equiangular triangles possible and thus is not well suited for the generation of directionally refined meshes. In this paper, it will be shown how a Delaunay triangulation can be modified to accommodate directional stretching of any desired magnitude.

2. The Delaunay Triangulation

Given a set of points in two dimensions, there exist many ways of joining them together to form a set of non-overlapping triangles. A Delaunay triangulation represents a unique construction of this type, which obeys certain specific properties. The geometric dual of the Delaunay triangulation is known as the Dirichlet tessalation. It is constructed by associating with each data point the area of the plane which is closer to that point, in terms of Euclidean distances, than to any other point in the plane. These regions have polygonal shapes and the tessalation of a closed domain results in a set of non-overlapping convex polygons covering the entire domain. If all point pairs whose Dirichlet regions have a face in common are joined by straight line segments, the Delaunay triangulation of these points is obtained. Figure 1 depicts the Dirichlet regions and associated Delaunay triangulation for a small set of points.

A Delaunay triangulation obeys the circumcircle property, which states that no

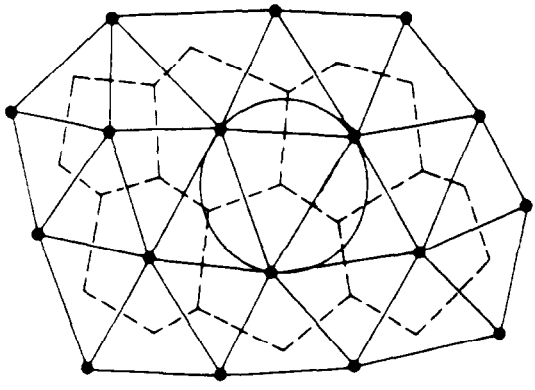


FIG. 1. Dirichlet tessalation and Delaunay triangulation of a set of points showing the circumcircle of one of the triangles.

vertex from any triangle may lie within the circumcircle of any other triangle. This can also be shown [18] to be equivalent to the equiangular property, which states that a Delaunay triangulation is that which maximizes the minimum of the six angles in any pair of triangles of the mesh which make up a convex quadrilateral. Each of these properties may be used as the basis for a method of constructing a Delaunay triangulation.

2.1. *Bowyer's Algorithm*

Bowyer's algorithm [19] makes use of the circumcircle property to generate a Delaunay triangulation in a sequential manner. The mesh points are introduced one at a time into an existing triangulation. The triangles whose circumcircles are intersected by the new mesh point are flagged. These may be quickly determined by first locating the triangle which encloses the new point. The circumcircle of this triangle must be intersected by the new point, and so it is flagged. The neighbors of this triangle are then searched, and then their neighbors, thus proceeding outwards in a tree-search pattern, each leg of which terminates when a non-intersected triangle has been located. The union of the flagged triangles forms a convex polygonal region, and a new structure is defined in this region by joining the new point to all the vertices of the polygon. Proofs that the polygon is convex and that the resulting triangulation is indeed a Delaunay structure can be found in the literature [17]. If an efficient search strategy is employed, Bowyer's algorithm exhibits linear computational complexity with the number of mesh points.

2.2. *Diagonal Swapping Algorithm*

This algorithm, originally proposed in [20], and reviewed in [18], makes use of the equiangular property. Assuming we have an arbitrary triangulation of a given set of points, we may proceed to transform it into a Delaunay triangulation by repeatedly swapping the positions of the edges in the mesh in accordance with the equiangular property. Hence, each pair of triangles which constitute a convex quadrilateral are examined. The two possible configurations for the diagonal interior to the quadrilateral are examined, as shown in Fig. 2, and the one which

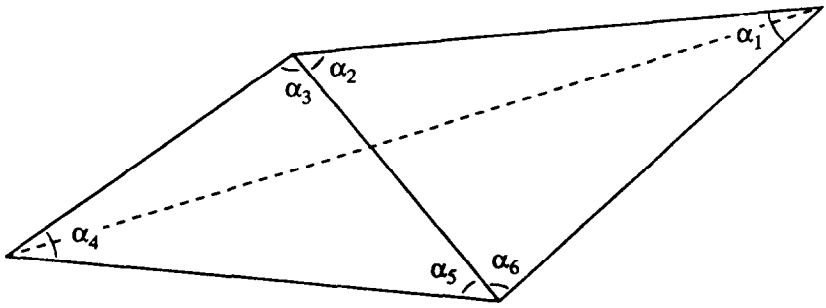


FIG. 2. The two possible configurations for the diagonal in a convex quadrilateral and the six angles associated with the most equiangular configuration (solid line diagonal).

maximizes the minimum of the six interior angles of the quadrilateral is chosen. Each time an edge swap is performed, the triangulation becomes more equiangular. Multiple edge swapping passes through the entire mesh are then effected, until the most equiangular (Delaunay) triangulation is obtained. Although this algorithm is guaranteed to converge, it has a much higher complexity than Bowyer's algorithm and is only useful for constructing an equiangular triangulation either when the initial mesh is coarse, or when it represents a small deviation from a Delaunay triangulation.

3. Stretching Factors

Equiangular triangulations, which have been termed "best fit or optimal triangulations" may be desirable if one wishes to tessalate or subdivide a domain in a uniform manner. However, by its very nature, such triangulations are ill-suited for the generation of meshes with directional stretchings. In fact, even if the mesh points are distributed sparsely in one direction and compactly in the perpendicular direction, the Delaunay construction will generally produce low aspect-ratio triangles of widely varying size. Thus, the standard Delaunay construction must be modified to accommodate directional mesh stretching.

We proceed by defining a stretching vector, i.e., a direction and magnitude, at each point of the mesh. It is important to note that this stretching is a local property, and that it must vary smoothly throughout the domain. Because a Delaunay triangulation is a local construction, we may thus map a local region of the mesh onto the stretched space defined by the stretching vector in that region, perform the triangulation in this space, and then project the triangulation back into physical space. A stretching vector at a point is given by its magnitude s and its direction θ, where

$$s \geqslant 1 \qquad \text{and} \qquad -\frac{\pi}{2} < \theta < \frac{\pi}{2}. \tag{1}$$

When $s = 1$, no stretching occurs. If a stretching smaller than unity is encountered, it is replaced by the inverse stretching in the perpendicular direction. The permissible values of θ in Eq. (1) reflect the fact that two stretchings of equal magnitudes in opposite directions are equivalent.

The mapped space is defined uniquely by the stretching vectors in the flow field. It is obtained by considering a two-dimensional control surface in three-dimensional space as described in [21]. For a one-dimensional problem, the analogous control surface consists of a line in two-dimensional space. If a grid with non-uniform spacing is employed, this "control line" is constructed such that the mesh points are equidistant from one another in terms of arc-length along the line, as shown in Fig. 3. For the two-dimensional problem, a control surface is constructed such that, when the mesh point distribution from the physical space is projected

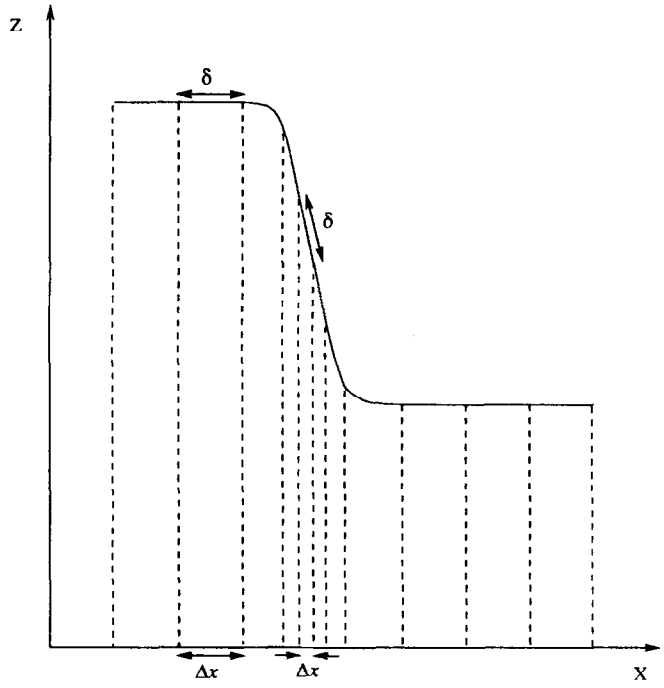


FIG. 3. Two-dimensional control line for one-dimensional grid. Variable grid spacing Δx is transformed to constant spacing δ along control line.

onto the control surface, these points are approximately equidistant from each other in all space directions (isotropically) in terms of arc-length along the control surface, as shown in Fig. 4. Thus, a Delaunay triangulation can be constructed on the control surface which, when projected back down onto the physical space will result in a triangulation of elements stretched in the desired direction and amount, as originally defined by the local stretching. If d represents a Euclidean distance in physical space:

$$d^2 = \Delta x^2 + \Delta y^2 \tag{2}$$

then, in the mapped space, it is replaced by

$$\delta^2 = \Delta x^2 + \Delta y^2 + \Delta z^2, \tag{3}$$

where

$$\Delta z = [\Delta x \sin\theta - \Delta y \cos\theta](s-1). \tag{4}$$

For a stretching of unity, Δz vanishes and the mapped space and the physical space become identical. If for example, a stretching of s is applied in the x-coordinate

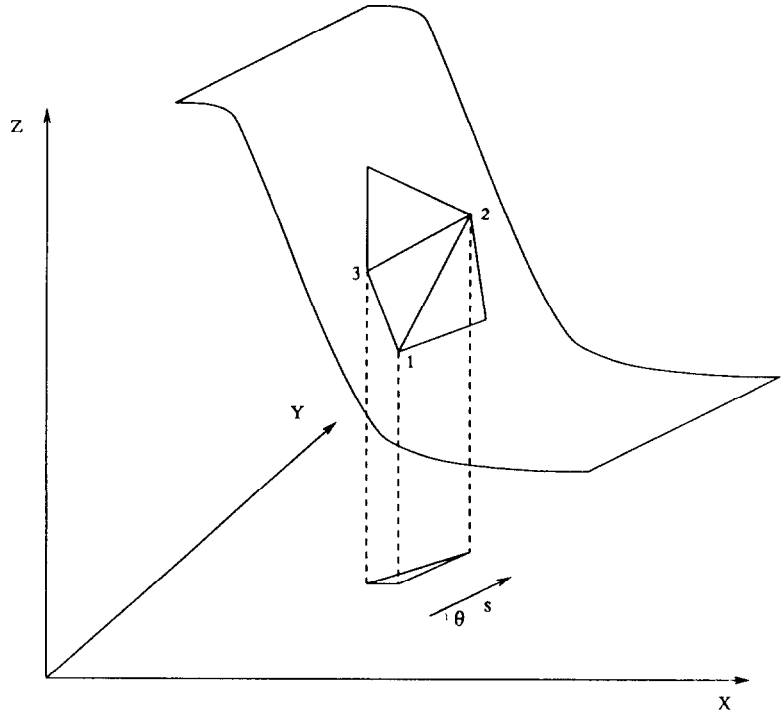


FIG. 4. Three-dimensional control surface for two-dimensional grid. Delaunay triangulation on the control surface is transformed to stretched triangulation in physical space as determined by the direction and magnitude of the stretching vector s.

direction, as would be required to resolve a boundary layer aligned with this direction, then, taking $\theta = 0$, an increment in the x direction maps to

$$\delta_x = \Delta x \tag{5}$$

and an increment in the y direction maps to

$$\delta_y = \Delta y[1 + (s-1)^2]^{1/2} \tag{6}$$

which, for large values of s, approaches

$$\delta_y = s\,\Delta y. \tag{7}$$

Thus, if we have a distribution of mesh points in the physical space which is closely packed in the y-direction and sparse in the x-direction, then, in the mapped space, this mesh point distribution becomes more uniform in both directions. By triangulating this mapped mesh point distribution, we obtain an equiangular triangulation in the stretched space and a directionally stretched mesh in the physical space. If the diagonal swapping algorithm is to be used for constructing the

triangulation, then the six interior angles of each convex quadrilateral in the locally stretched space must be considered. On the other hand, when Bowyer's algorithm is employed, the procedure can be thought of as the construction of a modified Delaunay triangulation, where the circumcircles of the triangles in the stretched space correspond to ellipses in the physical space.

In the implementation of this technique, a simplification is effected by requiring that the mapping apply only locally and that, in this region, the stretching vectors can be assumed to be constant, taken as the average of all stretching vectors in the region in which the mapping applies. Thus locally, the control surface becomes a control plane, and the calculation of arc lengths is no longer required: they may simply be replaced by straight line segments along the plane which can be computed as distances in three dimensions. Furthermore, the definition of a control surface $z = f(x, y)$ covering the entire domain, the topology of which can be extremely complicated, is no longer necessary. We are merely required to calculate increments in z between neighboring points, according to the direction of the control surface or plane, which applies only locally. The necessary condition for this stretched triangulation to succeed requires that the local variation of the stretching vectors in space be small compared with the average local cell size. Thus, in regions were the stretching values vary rapidly, a fine mesh resolution is needed so that, on the scale of the local mesh cells, the assumption of a constant stretching can be made and the stretched space appears locally planar or Euclidean.

4. Initial Mesh Generation

In the above discussion, it has been assumed that an initial triangulation with adequate resolution in highly stretched regions exists and that all further modifications or refinements of the mesh are of a purely local nature. However, the construction of an initial mesh is a global procedure, and thus local stretching values cannot be directly accommodated at this initial stage. To circumvent this difficulty, a regular Delaunay triangulation is first generated in physical space using Bowyer's algorithm, while disregarding the stretching values. Although this step involves non-local procedures, it results in a discretization of the physical space upon which purely local procedures may now be performed. The edge swapping algorithm is then employed to transform the mesh from an equiangular triangulation in physical space, to an equiangular triangulation in the stretched space. Because the initial mesh need only be coarse and, since no edge swapping is required in regions where the stretching values are small, the edge swapping algorithm can be expected to converge rapidly. Once this initial coarse stretched mesh is obtained, it can be adaptively refined by adding points and retriangulating locally following Bowyer's algorithm in the locally stretched space.

5. General Procedure for Multi-element Airfoils

To generate a stretched Delaunay mesh around an arbitrary geometry, a set of mesh points and their associated stretching vectors must first be defined. For the case of a multi-element airfoil configuration, this is achieved by first generating a structured quadrilateral C-mesh about each airfoil element, using a hyperbolic grid generator developed specifically for single airfoil geometries [22], thus resulting in a set of overlapping structured meshes as shown in Fig. 5. Stretching vectors are then defined at each mesh point, by taking their magnitude s as

$$s = \max\left(\frac{\Delta\xi}{\Delta\eta}, \frac{\Delta\eta}{\Delta\xi}\right) \tag{8}$$

and their direction equal to the direction of the ξ or η structured mesh lines, depending on whether the first or second values are chosen for s in Eq. (8). Here,

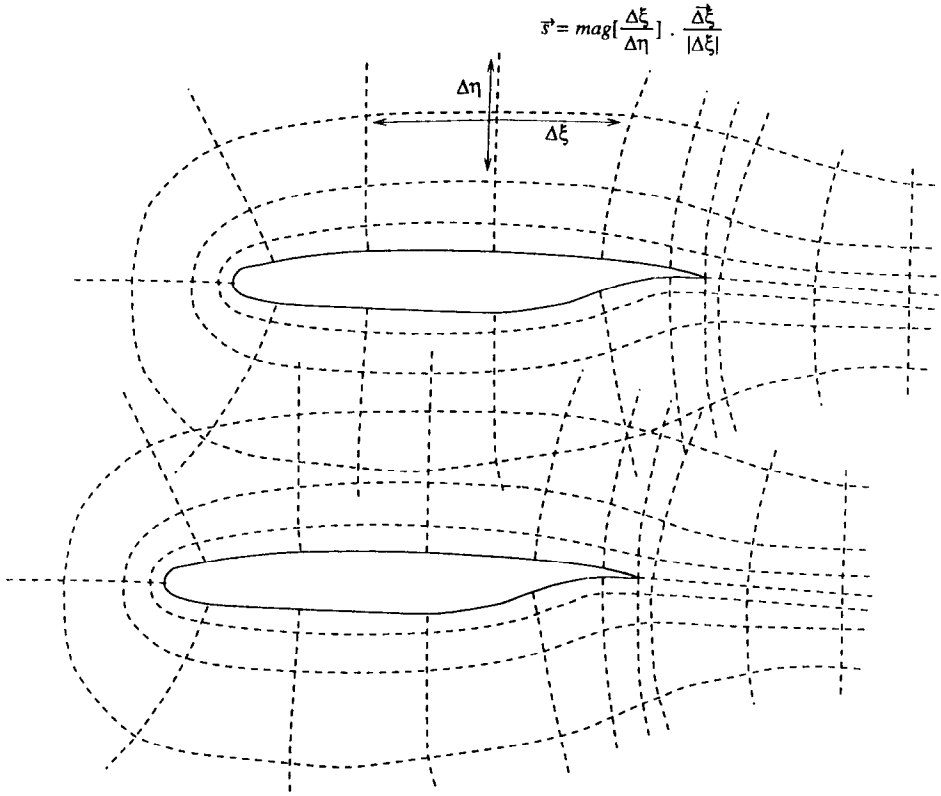


Fig. 5. Overlapping structured C-meshes for multi-element airfoil geometry providing initial definition of mesh-point distribution and local stretching factors.

$\Delta\xi$ and $\Delta\eta$ represent the local spacing of the structured C-mesh in the two mesh coordinate directions.

The mesh point distribution resulting from the set of overlapping C-meshes can now be used as the basis for a Delaunay triangulation. An initial triangulation is set up by joining the trailing-edge point of the main airfoil to all the outer boundary points. The points on the surface of the airfoils are then introduced and triangulated into the existing structure using Bowyer's algorithm. Because neighboring surface points on a given airfoil are much closer to each other than they are to any points on other airfoils, or to the far-field boundary points, the resulting triangulation is body conforming. That is, the triangulation contains elements inside the regions defined by the airfoils, as well as in the exterior of these regions, and the interior triangles all have a face aligned with the airfoil surfaces. These interior triangles are then identified and protected, thus preventing their structure from being broken when new mesh points are introduced, and hence preserving the integrity of the boundaries. The remaining mesh points are then introduced and triangulated into the existing structure. Because the mesh points are introduced in a sequential manner, in the initial stages of this construction, an extremely coarse grid containing a small subset of the total number of mesh points, and consisting of a small number of very large triangles will cover the entire domain. Thus, when introducing and retriangulating new mesh points, large regions of the domain will be affected, as shown in Fig. 6, and thus Bowyer's algorithm can no longer be considered to be a purely local process. Hence local stretching values cannot be taken into account at this stage, and a Delaunay triangulation in the physical space must be generated. However, once this triangulation has been constructed, the space is sufficiently discretized so that all further operations may be effected in a purely local manner. The next step consists of applying a Laplacian-type averaging procedure to the local stretching vectors over the existing Delaunay triangulation, thus ensuring a smoothly varying distribution of the stretching throughout the domain. The edge swapping routine is then applied in conjunction with the smoothed local stretching values to obtain a Delaunay triangulation in the stretched space. This mesh may finally be smoothed by slightly repositioning the points according to a Laplacian filtering operation described previously [1].

Once this initial stretched triangulation has been generated, it may be combined with a flow solver to produce an adaptive mesh refinement procedure, where the mesh point distribution is defined by the evolving solution of the flow field. For steady state calculations, this corresponds to converging the solution on the initial coarse mesh, adding points in regions where the computed flow gradients are large, and retriangulating these points into the existing structure using Bowyer's algorithm in the locally stretched space. When new points are introduced, they are assigned stretching values taken as the average of the stretchings of the neigboring points, thus maintaining a smooth distribution of stretching throughout the domain. When points are added on the surface of an airfoil, they must be repositioned onto the original surface definition of the airfoil, which, for curved surfaces, will not coincide with the position determined by linear interpolation between the

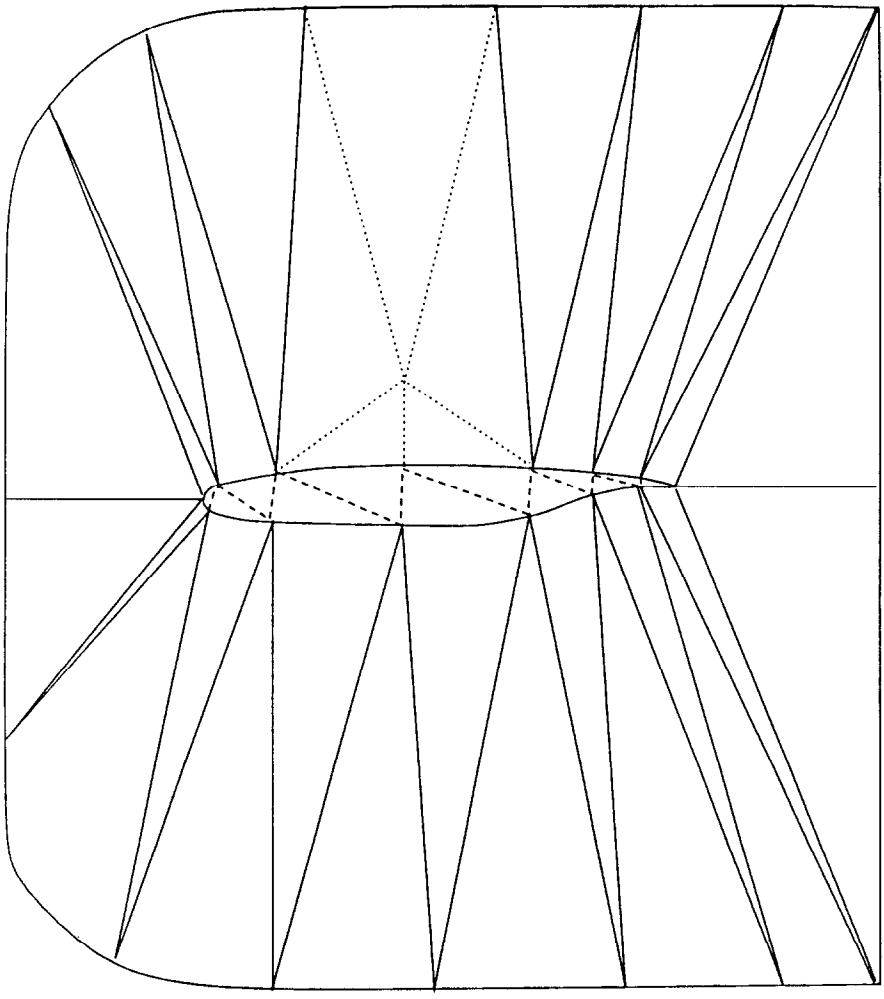


FIG. 6. Triangulation ofsingle airfoil geometry obtained in initial stages of Bowyer's algorithm when only a small subset of total grid points have been introduced. Dashed lines depict protected triangles interior to airfoil; dotted lines depect triangulation of newly inserted point.

two adjacent surface points. The new surface points are then triangulated by joining them to the two neighboring surface points, and to the vertices of all triangles exterior to the airfoil whose circumcircles are intersected. The flow solution is then interpolated onto the new finer mesh, and the entire solution-adaptation process is repeated. In this procedure, the total number of mesh points increases at each adaptation stage, since new mesh points are added in regions of large flow gradients, and no provisions are made for removing mesh points in regions of small gradients. For transient flow problems, where flow features may propagate across the domain,

this may prove to be unacceptably inefficient. However, for steady-state flow calculations, such as those considered in this work, the salient flow features remain approximately stationary throughout the convergence process, and mesh adaptation by enrichment alone yields a suitable mesh point distribution.

6. Flow Solving Algorithm

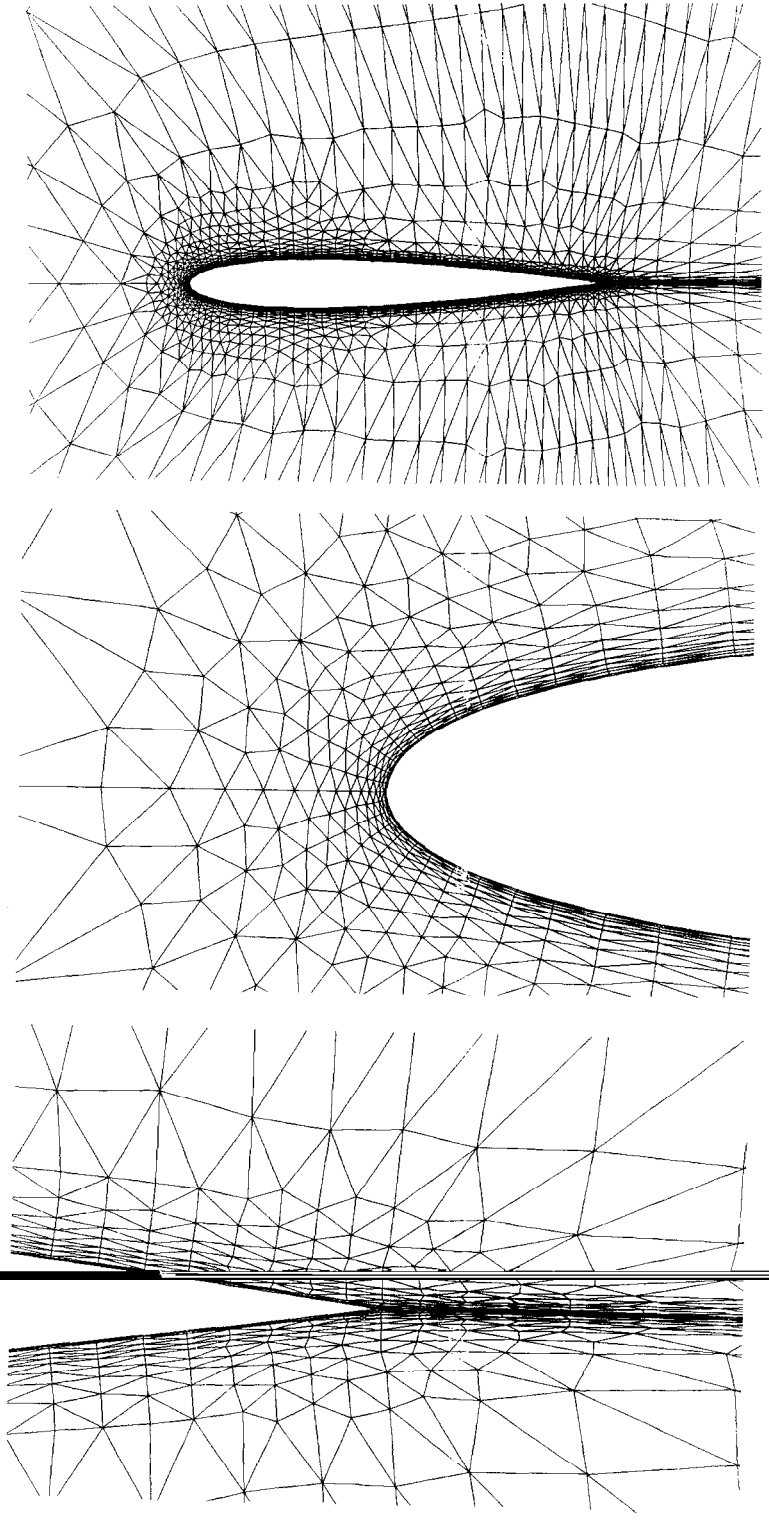
Adaptive meshing strategies necessarily involve interaction between the mesh-generation procedure and the flow-solution algorithm. The steady-state solution of the full Navier–Stokes equations on unstrucutred meshes is obtained following the method described in [11]. The flow variables are stored at the vertices of the triangles, and the gradients necessary for computing the viscous stresses in the equations are computed at the center of each triangular element by integrating the appropriate flow variables around the boundary of each triangle.

The discretization in space of the governing equations is then obtained through a Galerkin finite-element technique operating on these quantities and employing a lumped mass matrix. Additional artificial dissipation terms are required for stability and they are constructed as a blend of a Laplacian and biharmonic operator in the flow variables. Once the equations have been discretized in space, they are integrated in time until the steady-state solution is obtained using a multi-stage Runge–Kutta time-stepping scheme. Implicit smoothing of the residuals is employed to accelerate convergence. In future applications, a sequence of coarser unstructured meshes may be generated, and the unstructured multigrid algorithm of [11] may be employed to further improve the efficiency of the solver. At present, only laminar flows have been considered. For turbulent flow calculations, a field-equation turbulence model may be discretized and solved on the unstructured mesh.

7. Results

As a first example, an adaptive Navier–Stokes triangular mesh has been generated about a single NACA 0012 airfoil. An initial mesh, containing 1360 points, is first generated by constructing a structured C-mesh around the airfoil with a hyperbolic grid generator, triangulating this point distribution, and then swapping the diagonals. The full Navier–Stokes equations are then discretized and solved for on this mesh. The mesh is then adaptively refined according to the local gradient of density. The difference in density along each edge of the mesh is

FIG. 7. Illustration of the adaptively generated stretched triangulation about a NACA 0012 airfoil including details at the leading and trailing edges (number of points = 2316).



examined. When this value is larger than the average difference of the density across all the mesh edges, a new point is added midway along that edge. These new points are then triangulated into the existing mesh using Bowyer's algorithm in the locally stretched space. The refined mesh is then smoothed out by slightly repositioning the mesh points according to a Laplacian filtering operation [1] to ensure a smooth distribution of elements. The refined mesh, which contains a total of 2316 points, is depicted in Fig. 7, where the refinement occurring in the boundary layer regions and at the leading and trailing edges is apparent. The figure illustrates the topology of the mesh in the vicinity of the leading and trailing edges, where a smooth

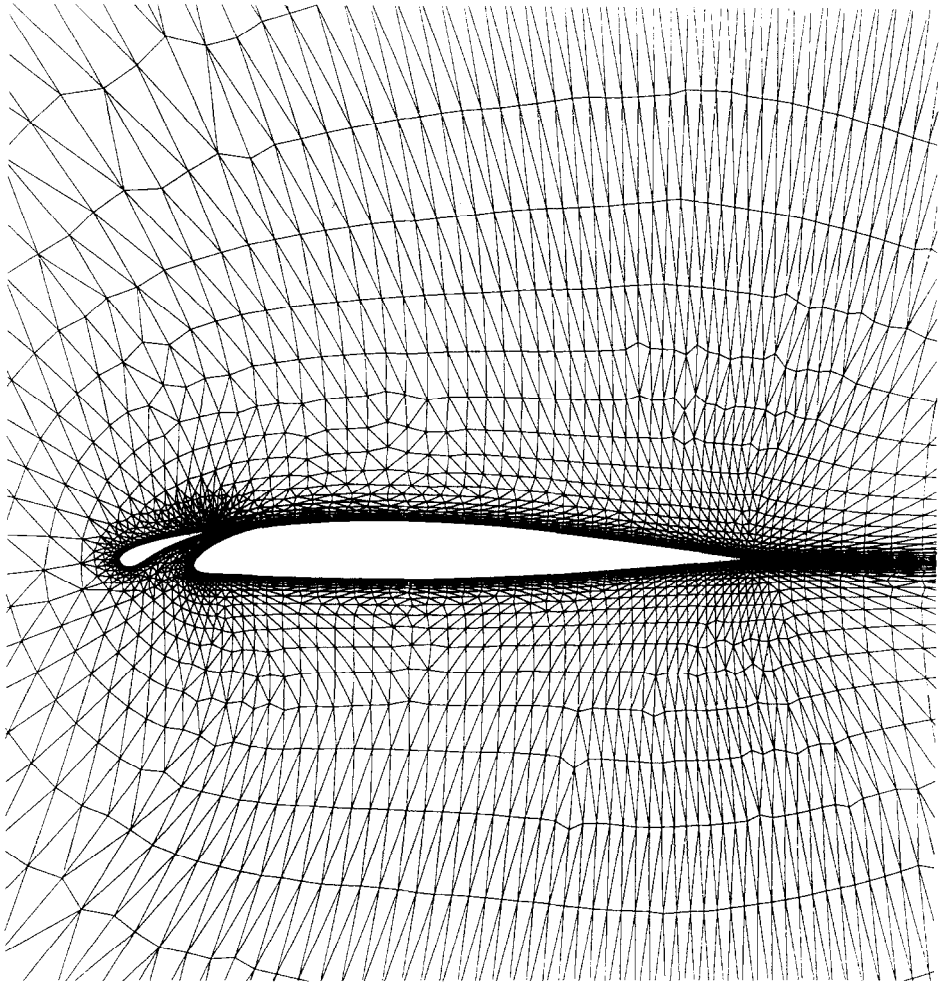


FIG. 8. Illustration of the initial coarse mesh for a two-element airfoil configuration including details of the mesh in the gap region (number of points = 5856).

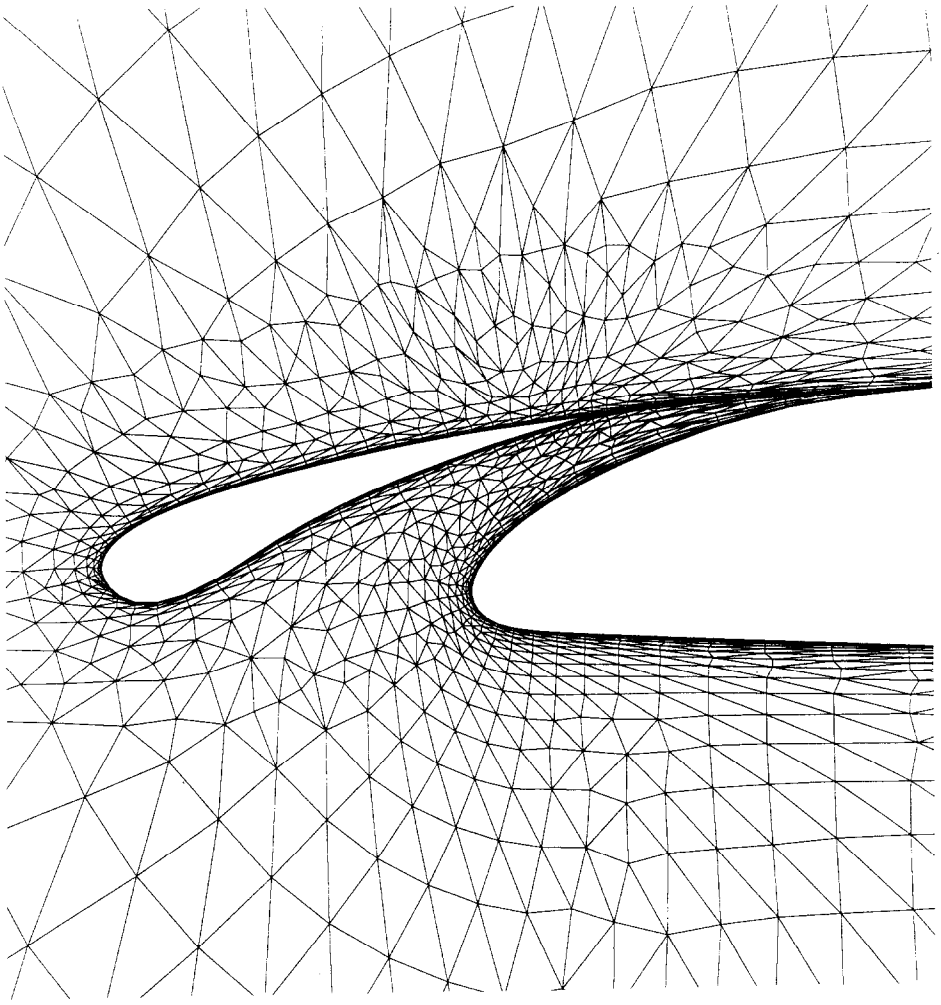


FIG. 8—*Continued.*

transition from an essentially regular, highly stretched triangulation near the body, to a random unstructured triangulation further out in the flow-field is observed.

The second configuration consists of a main airfoil with a leading edge slat. The initial mesh point distribution is obtained by constructing a structured C-mesh around each airfoil. The C-mesh about the main airfoil extends out to the far-field boundary, while the C-mesh about the slat is truncated less than one chord out from the surface of the slat, and also beyond the region where the wake lines from the slat impinge upon the main airfoil. This point distribution is then triangulated, the stretching values are smoothed, the edges swapped, and finally the point distribution is smoothed. The resulting mesh, which contains 5856 points is depicted in Fig. 8. The stretching of the mesh in the boundary layer and wake regions of

FIG. 9. Mach contours in the flow-field obtained with the Navier–Stokes solution on the initial coarse mesh for the two-element airfoil configuration, Mach = 0.5, incidence = 3°, Re = 5000.

both airfoils is apparent, and a smooth transition of the elements is observed in the gap region, between the main airfoil and the leading edge slat. The steady-state solution of the Navier–Stokes equations was obtained on this mesh, for a Mach number of 0.5, an incidence of 3°, and a Reynolds number of 5000. A plot of the Mach contours in the flow-field is given in Fig. 9, where the boundary layer regions are evident, and a recirculation region is observed near the trailing edge on the upper surface of the main airfoil. A region of low velocity fluid is also seen to occur in the gap region between the main airfoil and the slat. A plot of the velocity vectors in this region, as given in Fig. 10, clearly shows the boundary layer profiles on

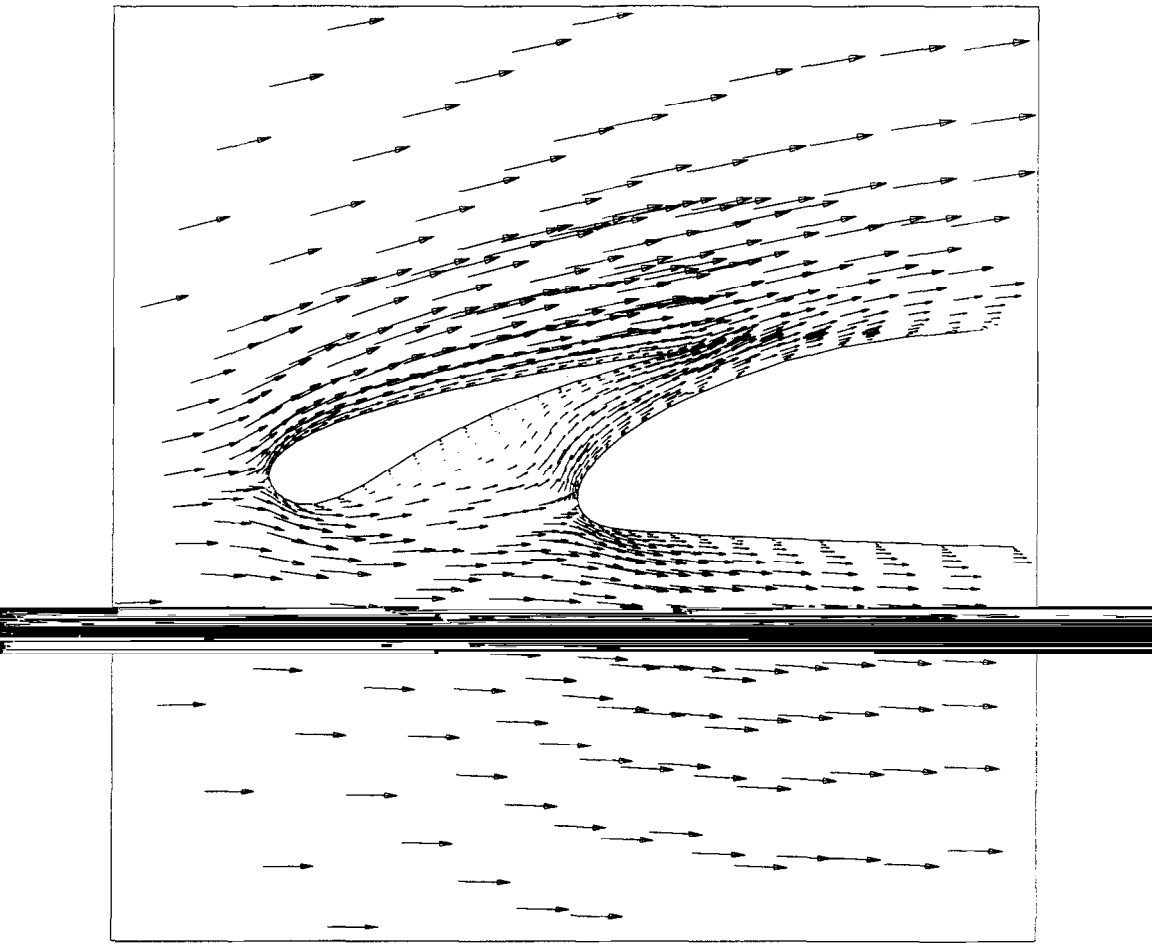


FIG. 10. Vector velocities in the gap region of the two-element airfoil configuration computed on the initial coarse mesh, Mach = 0.5, incidence = 3°, Re = 5000.

both airfoils, as well as the region of recirculating flow on the lower surface of the slat. This solution was then used to adaptively refine the mesh according to the local gradient of the Mach number. The refined mesh, depicted in Fig. 11, contains 11,377 points. Refinement is seen to occur in the boundary layer regions, as well as in a portion of the gap region. However, as expected, the regions of recirculating flow are left unrefined. This mesh contains triangles of aspect ratio of the order of 1000:1 in the wake regions, and up to 100:1 midway along the surface of the main airfoil and on the upper surface of the slat.

From the figures, it can be seen that an essentially regular triangular mesh is

obtained in most of the boundary layer regions. However, in regions where two bodies are within close proximity to one another and in regions where mesh adaptation takes place, the mesh point distribution may become somewhat irregular. This irregularity of the mesh way have adverse effects on the accuracy of the solution in the boundary layer region. In order to compare theoretical and computed boundary layer profiles on such meshes, a profile at a particular streamwise station must be constructed by interpolating the values from the closest mesh points in regions where the mesh is irregular. This procedure introduces additional interpolation errors, thus creating further uncertainties in the comparison. Further work is therefore required to more precisely assess the effect of the mesh structure on the flow solution in these regions.

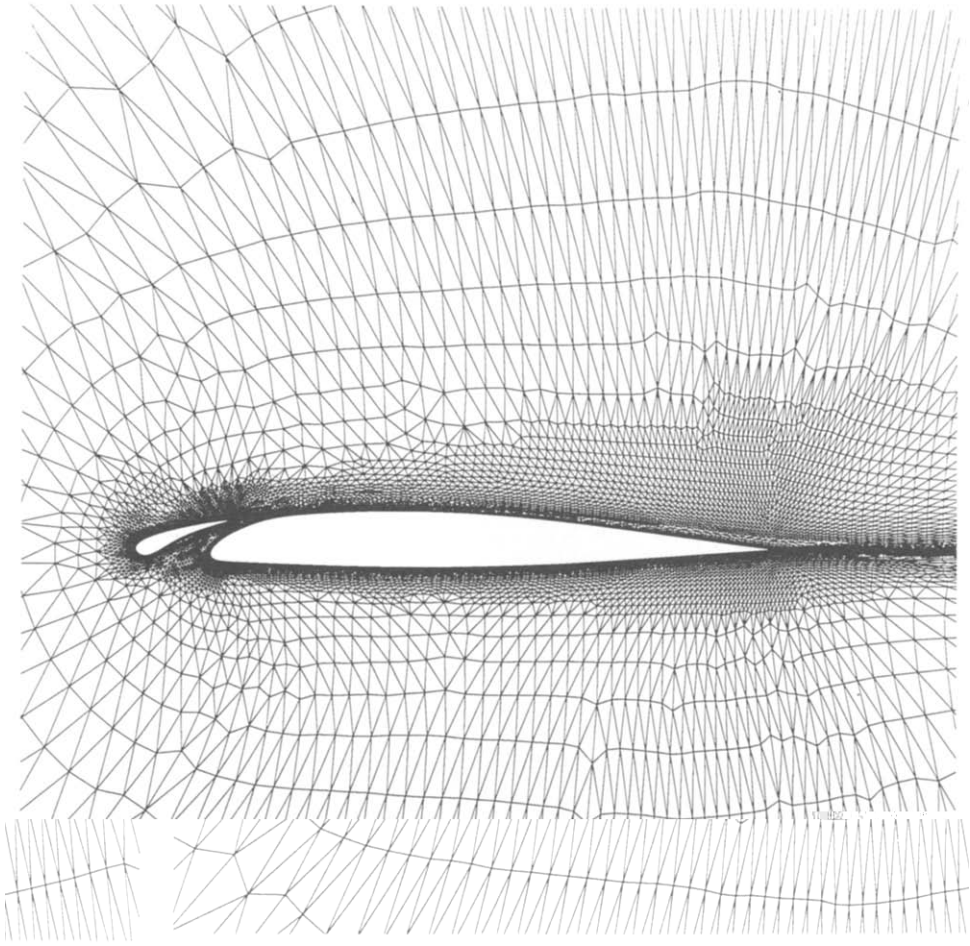


FIG. 11. Illustration of the adaptively refined mesh for the two-element airfoil configuration including details of the mesh in the gap region (number of points = 11,377).

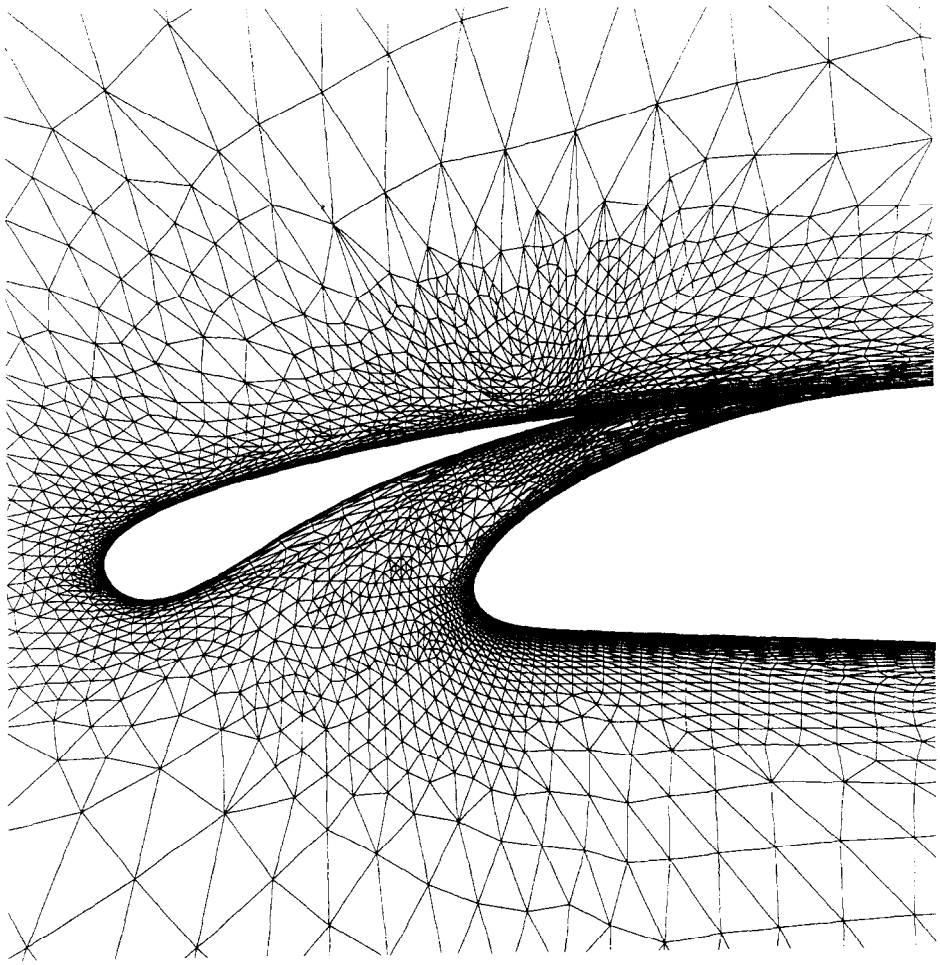


FIG. 11—*Continued.*

8. CONCLUSION

A method for adaptively generating unstructured triangular meshes with highly stretched elements, suitable for Navier–Stokes computations has been demonstrated. The method is efficient, in that once an initial coarse stretched triangulation has been set up, all further refinements are of a local nature and have a near linear computational complexity. For the two-element airfoil mesh of the previous section, the initial coarse mesh required approximately 210 CPUs to generate on a Convex C-1 computer. Roughly one third of this time was required to construct the initial Delaunay triangulation in physical space, and the remainder

was required by the edge-swapping algorithm, which converged in 13 passes through the mesh. The adaptive refinement of this grid, which effectively doubled the number of mesh points, required roughly 80s of CPU time.

In the examples presented in this work, the stretching factors for the mesh were not determined adaptively by the flow solution, but by the initial mesh, and are held constant throughout the adaption process. In future work, it may be possible to modify the stretching vectors according to the evolving flow solution. At each adaptation stage, the mesh can then be reconfigured according to the new values of the stretching vectors using the diagonal swapping algorithm. However, since the mesh point distribution must be closely linked to the distribution of stretching in order to obtain a smooth mesh, it may prove desirable to regenerate the entire mesh starting with new mesh point and stretching distributions, both determined by the flow solution.

The extension of this technique to three dimensions is not entirely straightforward. Although Bowyer's algorithm extends readily to higher dimensions, the equiangular property applies only in two dimensions, and thus no straightforward counterpart to the edge-swapping algorithm exists in three dimensions. However, all the other concepts apply in three dimensions; thus if an initial coarse stretched mesh can be generated in three dimensions, then it may easily be adaptively refined.

ACKNOWLEDGMENT

The help of G. Erlebacher of the Computational Methods Branch at NASA Langley is greatly appreciated for his sharing of knowledge and ideas in differential geometry.

REFERENCES

1. D. J. MAVRIPLIS, *AIAA J.* **26**, 824 (1988).
2. M. GILES, in *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics, Williamsburgh, VA, 1988*, edited by D. Dwoyer, Y. Hussaini, and R. Voigt (Springer-Verlag, New York/Berlin, 1988), p. 273.
3. R. LOHNER, K. MORGAN, J. PERAIRE, AND M. VAHDATI, *Int. J. Num. Methods Fluids* **7**, 1093 (1987).
4. A. JAMESON, T. J. BAKER, AND N. P. WEATHERILL, "Calculation of Inviscid Transonic Flow over a Complete Aircraft," AIAA paper 86-0103, 1986 (unpublished).
5. B. STOUFFLET, J. PERIAUX, F. FEZOUI, AND A. DERVIEUX, "Numerical Simulation of 3-D Hypersonic Euler Flows Around Space Vehicles Using Adapted Finite Elements," AIAA paper 87-0560, 1987 (unpublished).
6. N. P. WEATHERILL, L. J. JOHNSTON, A. J. PEACE, AND J. A. SHAW, Aircraft Research Association Ltd. Report 70, Bedford, UK, December 1986 (unpublished).
7. N. NAKAHASHI, "FDM-FEM Zonal Approach for Viscous Flow Computations over Multiple Bodies," AIAA paper 87-0604, 1987 (unpublished).
8. D. G. HOLMES, AND S. CONNELL, "Solution of the 2-D Navier–Stokes Equations on Unstructured Adaptive Grids," AIAA paper 89-1932, 1989 (unpublished).
9. L. MARTINELLI, Ph. D. thesis, Department of Mechanical and Aerospace Engineering, Princeton University, 1988 (unpublished).

10. R. RADESPIEL AND R. C. SWANSON, "An Investigation of Cell-Centered and Cell-Vertex Multigrid Schemes for the Navier–Stokes Equations," AIAA paper 89-0543, 1989 (unpublished).
11. D. J. MAVRIPLIS, A. JAMESON, AND L. MARTINELLI, "Multigrid Solution of the Navier–Stokes Equations on Triangular Meshes," AIAA Paper 89-0120, 1989 (unpublished).
12. P. ROSTAND, ICASE Rep. 88-63, 1988 (unpublished).
13. J. PERAIRE, M. VAHDATI, K. MORGAN, AND O. C. ZIENKIEWICZ, *J. Comput. Phys.* **72**, 449 (1987).
14. D. J. MAVRIPLIS, "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," AIAA paper 88-3706-CP, 1988 (unpublished).
15. D. G. HOLMES AND S. H. LAMSON, "Adaptive Triangular Meshes for Compressible Flow Solutions," in *Proceedings, International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Karlsruhe, West Germany, 1986*, edited by J. Hauser and C. Taylor (Pineridge, Swansea, UK, 1986), p. 413.
16. N. P. WEATHERILL, Princeton University MAE Report No. 1715, 1985 (unpublished).
17. T. J. BAKER, "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets," AIAA paper 87-1124, 1987 (unpublished).
18. R. SIBSON, *Comput. J.* **21**, 243 (1978).
19. A. BOWYER, *Comput. J.* **24**, 162 (1981).
20. C. L. LAWSON, Cal. Tech. Jet Propulsion Lab. Tech. Memo 299, 1972 (unpublished).
21. P. EISEMAN AND M. BOCKELIE, "Adaptive Grid Solution for Shock-Vortex Interaction," in *Proceedings, 11th International Conference on Numerical Methods in Fluid Dynamics, Williamburgh, VA, 1988*, edited by D. Dwoyer, Y. Hussaini, and R. Voigt (Springer-Verlag, New York/Berlin, 1988), p. 240.
22. L. WIGTON, The Boeing Company, private communication, (1988).