AIAA'87

AIAA-87-0452

Improvements to the Aircraft Euler Method

A. Jameson and T. J. Baker
Princeton University, Princeton, N. J.

AIAA 25th Aerospace Sciences Meeting

January 12-15, 1987/Reno, Nevada

IMPROVEMENTS TO THE AIRCRAFT EULER METHOD

A. Jameson and T. J. Baker
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, New Jersey

Abstract

An unstructured mesh of tetrahedra offers an attractive approach to mesh generation for complex three dimensional shapes. The combination of a method for constructing tetrahedral meshes and a finite element technique for solving the Euler equations has resulted in a powerful new approach to the problem of calculating flows over complex geometries. Recent progress and improvements to the authors' aircraft Euler method are described and results presented to demonstrate its capability.

1. Introduction

During the last two decades aerodynamics has been transformed by the widespread introduction of computational methods to treat previously intractable problems. Improvements in computer technology have made it feasible to attempt numerical calculations of progressively more complicated mathematical models of fluid flow, and to apply these methods to increasingly elaborate geometric configurations. Following the introduction of panel methods for subsonic flow in the sixties[1,2], and major advances in the simulation of transonic flow by the potential flow approximation in the seventies[3-6], the eighties have seen rapid developments in methods for solving the Euler and Navier Stokes equations[7-13].

A major pacing item has been the development of a suitable method for mesh generation. For simple wing/body combinations it is possible to generate rectilinear meshes without too much difficulty[5]; for more complicated configurations consisting, for example, of wing/body/tail/fin, it becomes increasingly difficult to produce a structured mesh that is aligned with all solid surfaces[14]. A further increase in geometric complexity, for example, a complete aircraft with pylon mounted engines, poses extremely severe problems for any method that attempts to generate a structured rectilinear mesh. An alternative is to use tetrahedral cells in a unstructured mesh which can be adapted to conform to the complex surface of an aircraft. This approach was adopted by Bristeau, Glowinski, Periaux, Perrier, Pironneau and Poirer[6], who achieved a striking success in solving the potential flow equation to predict the flow around a Falcon 50 by a finite element method.

Finite element methods have also been developed for solving the Euler equations on triangular meshes[15,16]. The main motivation for using unstructured triangular meshes is the relative ease with which complex geometries can be treated. However, the generation of a tetrahedral mesh around complicated three dimensional configurations remains a formidable problem. During the last two years the present authors have been engaged in the development of a new finite element method to solve the three dimensional Euler equations. The method links a novel approximation procedure to a new method of generating tetrahedral meshes around arbritrary shapes, and we have applied it to calculate the flow past a complete aircraft. Preliminary results were presented at the AIAA 24th Aerospace Sciences Meeting[17]. In reference 17 we indicated the need for various improvements in order to bring the method to the point where it would be a useful tool for analysis and design. In particular it was necessary to reduce the computational costs associated with both the mesh generation and the flow solution, and to improve the accuracy by better mesh control. Also the method might be applied to a broader range of applications if it were extended to treat viscous flows. In this paper we review the salient features of the method, and report on our progress in realizing some of the needed improvements.

The finite element approximation can be obtained by directly approximating the integral equations for conservation of mass, momentum and energy in

polyhedral control volumes. Steady state solutions are obtained by integrating the time dependent equations with a multistage time stepping scheme. Convergence is accelerated by the use of locally varying time steps, residual averaging and enthalpy damping[9]. It is shown in section 2 that the scheme can also be derived from a Galerkin method in which the test function space is the set of piecewise linear tetrahedral elements. The resulting equations can be shown to be equivalent to a flux balance based on polyhedral control volumes formed by the union of tetrahedra meeting at a common vertex. It turns out that each face is associated with precisely two such control volumes. It is therefore possible to reformulate the calculation in a particularly elegant way, in which the fluxes are evaluated in a single main loop over the faces. This novel decomposition leads to a substantial reduction in computational complexity.

While this scheme can equally well be interpreted as a finite element or a finite volume method when it is applied to the Euler equations, the interpretation as a Galerkin method yields a natural extension to the Navier Stokes equations. It turns out that the contributions of the viscous stresses can also be accumulated very neatly by loops over the faces, like the calculation of the Euler terms. A scheme of this type has already been included in a new program which we are developing for viscous flow.

Efficient realization of the scheme on computers with vector registers, such as the Cray machines, poses some difficulties. The references by table look up to indirectly addressed nodes in the loops over the faces can lead to a vector dependency. Section 5 indicates how vectorization can be accomplished by separating the faces into groups such that no dependency occurs.

Another way to improve the computational efficiency is to reduce the number of cycles needed to reach a steady state. We believe that multigrid procedures offer an attractive opportunity to achieve this, although their realization on unstructured meshes will require complex programming. Alternative multigrid schemes based on cell centered and nodal algorithms for unstructured two dimensional meshes are described in a companion paper[18]. Significant gains in efficiency were demonstrated with both schemes.

A distinctive feature of our mesh generation procedure is the separation of the processes of generating the mesh points and connecting them to form tetrahedra. Separate meshes are first generated around the individual aircraft components to create a cluster of points surrounding the whole aircraft. We do not require any regularity in this initial point distribution, only that a reasonable point density is created corresponding to the anticipated variation in the flowfield. The swarm of mesh points is then connected together to form tetrahedral cells which provide the basis for a single finite element approximation for the entire domain. This use of triangulation to unify separately generated meshes bypasses the need to devise interpolation procedures for transferring information between overlapping grids. The triangulation of a set of points to form disjoint tetrahedra is in general nonunique; our procedure is to generate the Delaunay triangulation[19-24]. This is dual to the Voronoi diagram[20] that results from a division of the domain into polyhedral neighborhoods, each consisting of the subdomain of points nearer to a given mesh point that any other mesh point. The Voronoi diagram has been exploited by others as a natural subdivision for calculations involving irregularly spaced points[25,26]. In our work it is used solely as a device for generating a triangulation, and is entirely independent of the solution algorithm, which might also be linked to any other triangulation scheme capable of producing a mesh with a distribution of points adequate to resolve the flow. The implementation of a procedure for Delauany triangulation which is also capable of maintaining the integrity of solid interior surfaces presents a number of interesting problems. These are discussed in Sections 6 and 7, which also outline improvements in the procedure which have led to a dramatic reduction in the computing time required for the triangulation.

2. Finite Element Approximation

Let p, ρ, u, v, w, E and H denote the pressure, density, Cartesian velocity components, total energy and total enthalpy. For a perfect gas

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

where γ is the ratio of specific heats. The Euler equations for flow of a compressible inviscid fluid can be written in integral form as

$$\frac{\partial}{\partial t}\iiint_{\Omega} w\,d\Omega + \iint_{\partial\Omega} \underline{F}\cdot d\underline{S} = 0 \qquad (1)$$

for a domain Ω with boundary $\partial\Omega$ and directed surface element $\underline{dS}$. Here w represents the conserved quantity and $\underline{F}$ is the corresponding flux. For mass conservation

$$w = \rho\ ,\ \underline{F} = (\rho u,\ \rho v,\ \rho w)$$

For conservation of momentum in the x direction

$$w = \rho u,\ \underline{F} = (\rho u^2 + p,\ \rho uv,\ \rho uw)$$

with y and z momentum quantities similarly defined, and for energy conservation

$$w = \rho E,\ \underline{F} = (\rho Hu,\ \rho Hv,\ \rho Hw)$$

In order to derive a Galerkin approximation, consider the differential form of equation (1)

$$\frac{\partial w}{\partial t} + \nabla\cdot\underline{F} = 0$$

Multiplying by a test function ϕ and integrating by parts over space leads to

$$\frac{\partial}{\partial t}\iiint_{\Omega}\phi w\,d\Omega = \iiint_{\Omega}\underline{F}\cdot\nabla\phi\,d\Omega - \iint_{\partial\Omega}\phi\underline{F}\cdot d\underline{S} \qquad (2)$$

Suppose now that we take ϕ to be the piecewise linear function with the value unity at one node (denoted by 0 in Figure 1), and zero at all other nodes. Then the last term vanishes except in the case when 0 is adjacent to the boundary. Also, $\nabla\phi$ is constant in every tetrahedron, and differs from zero only in the tetrahedra with a common vertex at node 0. Since ϕ_x is constant in a tetrahedron it may be evaluated as

$$\phi_x = \frac{1}{V}\iiint \phi_x\,dxdydz = \frac{1}{V}\sum_k S_{x_k}\bar{\phi}_k$$

where V is the cell volume, S_{x_k} and $\bar{\phi}_k$ are projected area of the kth face in the x direction and the average value of ϕ on the kth face, and the sum is taken over the faces of the tetrahedron. For the given test function $\bar{\phi} = 1/3$ on the faces 012, 023 and 031, and zero on the face 123. Also, the projected area S_x on face 123 is equal and opposite to the sum of the projected face areas of the other three faces. Using the same procedure to evaluate ϕ_y and ϕ_z, it follows that

$$\nabla\phi = -\underline{S}/3V \qquad (3)$$

where $\underline{S}$ is the directed area of the face opposite vertex 0. Now treat $\underline{F}$ as piecewise linear and use equation (3) to evaluate the volume integral on the right side of equation (2). Then each tetrahedron meeting at node 0 introduces a contribution $(\underline{F}\cdot\underline{S})/3$ where $\underline{F}$ is the average value of $\underline{F}$ in the cell. For the cell illustrated in Figure 1, for example,

$$\underline{F} = \tfrac{1}{4}(\underline{F}_0 + \underline{F}_1 + \underline{F}_2 + \underline{F}_3)$$

Summing over all cells meeting at node 0 leads to the total contribution

$$\frac{1}{3}\sum_k \underline{F}_k\cdot\underline{S}_k$$

Since the control volume is closed, however,

$$\sum_k \underline{S}_k = 0$$

Therefore, the contribution of $\underline{F}_0$ to $\underline{F}_k$ can be discarded, leading to a sum over the faces multiplied by a constant. Thus, if we write

$$\tilde{\underline{F}} = \tfrac{1}{3}(\underline{F}_1 + \underline{F}_2 + \underline{F}_3)$$

for the average value of $\underline{F}$ on the face opposite vertex 0 we find that the righthand side of equation (2) can be replaced by

$$-\frac{1}{4}\sum_k \tilde{\underline{F}}_k\cdot\underline{S}_k$$

On the left hand side of equation (2) we take w to be constant inside the control volume. This corresponds to the use of a lumped mass matrix in the Galerkin approximation. Since ϕ is piecewise linear, the volume average value is $\bar{\phi} = 1/4$. The factor 1/4 cancels on each side and the approximation to equation (2) can therefore be written as

$$\frac{d}{dt}\Big(\sum_k V_k\Big)w + \sum_k \tilde{\underline{F}}_k\cdot\underline{S}_k = 0 \qquad (4)$$

Equation (4) can also be interpreted directly as a finite volume approximation to the conservation laws

in integral form. The introduction of a test function Φ and the use of the weak form, equation (2), has the advantage, however, that it is easily extended to treat the Navier Stokes equations. Since the flux vector F is not differentiated in equation (2), this formulation eliminates the need to introduce explicit formulas to approximate second derivatives.

Referring to Figure 2, which illustrates a two dimensional mesh, it may be seen that with a triangular or tetrahedral mesh, each face is a common external boundary to exactly two control volumes. Therefore, each internal face can be associated with a set of 5 mesh points consisting of its three corners 1, 2 and 3, and the vertices 4 and 5 of the two tetrahedra based on the face, as illustrated in Figure 3. Vertices 4 and 5 are the centers of the two control volumes influenced by the face. It is now possible to generate the approximation (4) by presetting the flux balance at each mesh point to zero, and then performing a single loop over the faces. For each face one first calculates the fluxes of mass, momentum and energy across the face, and then one assigns these contributions to the vertices 4 and 5 with positive and negative signs respectively. Since every contribution is transferred from one control volume into another, all quantities are perfectly conserved. Mesh points on the inner and outer boundaries lie on the surface of their own control volumes, and the accumulation of the flux balance in these volumes has to be correspondingly modified. At a solid surface it is also necessary to enforce the boundary condition that there is no convective flux through the faces contained in the surface.

3. Dissipation

Equation (4) represents a nondissipative approximation to the Euler equations. Dissipative terms may be needed for two reasons; to eliminate the occurrence of undamped or lightly damped modes, and to prevent oscillations near shock waves.

The simplest form of dissipation is to add a term generated from the difference between the value at a given node and its nearest neighbors. That is, at node 0, we add a term

$$D_0 = \sum_k \epsilon_{k0}^{(1)} (w_k - w_0) \qquad (5)$$

where the sum is over the nearest neighbors, as illustrated in Figure 4. The contribution $\epsilon_{k0}^{(1)}(w_k - w_0)$ is balanced by a corresponding contribution $\epsilon_{k0}^{(1)}(w_0 - w_k)$ at node k, with the result that the scheme remains conservative. The coefficients $\epsilon_{k0}^{(1)}$ may incorporate metric information depending on local cell volumes and face areas, and can also be adapted to gradients of the solution. It is shown in reference 17 that the addition of properly controlled differences along edges can be used to assure a positivity condition on the coefficients of the semi-discrete scheme, which will prevent growth in the maximum norm and inhibit oscillations in the solution.

Formula (5) is no better than first order accurate unless the coefficients are proportional to the mesh spacing. A more accurate scheme is obtained by recycling the edge differencing procedure. After first setting

$$E_0 = \sum_k (w_k - w_0) \qquad (6)$$

at every mesh point, one then sets

$$D_0 = -\sum_k \epsilon_{0k}^{(2)} (E_k - E_0) \qquad (7)$$

An effective scheme is produced by blending formulas (5) and (7), and adapting $\epsilon_{0k}^{(1)}$ to the local pressure gradient. This is accomplished by calculating

$$P_0 = \sum_k \left| \frac{P_k - P_0}{P_k + P_0} \right|$$

at every mesh point, and then taking $\epsilon_{0k}^{(1)}$ proportional to max (P_0, P_k). Formulas of this type have been found to have good shock capturing properties, and the required sums can be efficiently assembled by loops over the edges.

4. Integration to a Steady State

The discretization procedures of Sections 2 and 3 lead to a set of coupled ordinary differential equations, which can be written in the form

$$\frac{dw}{dt} + R(w) = 0 \qquad (8)$$

where w is the vector of the flow variables at the mesh points, and R(w)

is the vector of the residuals, consisting of the flux balances defined by equation (4), together with the added dissipative terms. These are to be integrated until they reach a steady state.

For this purpose we use a multistage time stepping scheme of the same type which has proved effective in calculations on rectilinear meshes. Let w^n be the result after n steps. To advance one step Δt with an m stage scheme we set

$$w^{(0)} = w^n$$

$$w^{(1)} = w^{(0)} - \alpha_1 \Delta t \, R^{(0)}$$

$$\cdots$$

$$w^{(m-1)} = w^{(0)} - \alpha_{m-1} \Delta t \, R^{(m-2)}$$

$$w^{(m)} = w^{(0)} - \Delta t \, R^{(m-1)}$$

$$w^{n+1} = w^{(m)}$$

The residual in the q+1st stage is evaluated as

$$R^{(q)} = \frac{1}{V} \sum_{r=0}^{q} \{\beta_{qr} Q(w^{(r)}) - \gamma_{qr} D(w^{(r)})\}$$

where Q(w) is the approximation to the Euler equations and D(w) represents the dissipative terms, and the coefficients β_{qr} and γ_{qr} satisfy the consistency condition that

$$\sum_{r=0}^{q} \beta_{qr} = \sum_{r=0}^{q} \gamma_{qr} = 1$$

In practice a three stage scheme in which the dissipative terms are evaluated only once has proved effective. For this scheme

$$\alpha_1 = .6 \, , \; \alpha_2 = .6$$

$$\beta_{qq} = 1 \, , \; \beta_{qr} = 0 \, , \; q > r$$

$$\gamma_{q0} = 1 \, , \; \gamma_{qr} = 0 \, , \; r > 0$$

Convergence to a steady state is accelerated by using a variable time step close to the stability limit at each mesh point. The scheme is accelerated further by the introduction of residual averaging[9]. At the mesh point 0 the residual R_0 is replaced by $\tilde{R}_0$ where $\tilde{R}_0$ is an approximation to the solution $\bar{R}_0$ of the equation

$$\bar{R}_0 + \sum_k \epsilon(\bar{R}_0 - \bar{R}_k) = R_0 \qquad (9)$$

in which the sum is over the nearest neighbors. This is similar to the weighted average appearing in the Galerkin method, but with the opposite sign for the coefficient ϵ, leading to an increase in the permissible time step instead of a reduction. In practice it has been found effective to obtain $\bar{R}$ by using two steps of the Jacobi iteration

$$\tilde{R}_0^{(m)} + \sum_k \epsilon(\tilde{R}_0^{(m)} - \tilde{R}_k^{(m-1)}) = R_0 \qquad (10)$$

5. Vectorization

The loop over the faces which is used to calculate the residuals has the following schematic form:

```
      DO 10  I = 1, NFACE
      N1  =  NDF(I,1)
      N2  =  NDF(I,2)
      N3  =  NDF(I,3)
      N4  =  NDF(I,4)
      N5  =  NDF(I,5)
      FLUX = FCTN(VARIABLES AT N1, N2, N3)
      RESIDUAL(N4) = RESIDUAL(N4) + FLUX
      RESIDUAL(N5) = RESIDUAL(N5) - FLUX
   10 CONTINUE
```

Here NDF(I,J), J=1,5 are the five nodes associated with the Ith face as illustrated in figure 3. The flux through each face is calculated once, and added to and subtracted from the control volumes enclosing nodes N4 and N5, thus ensuring exact conservation of mass, momentum and energy.

The last two instructions, which accumulate the residuals, contain recurrences which lead to a vector dependency in the event that the same node is referred to as N4 or N5 more than once during the execution of the loop. If all the faces are included in a single loop, this must occur because every node is influenced by all the faces of the polyhedron containing that node. In a machine using vector registers separate copies of the same residual may be placed at different locations in the vector registers, resulting in improper accumulation. Consequently vectorization is suppressed by an automatic vectorizing compiler. This difficulty can be overcome by

separating the faces into groups such that within each group no node is referenced as N4 or N5 more than once. Then separate loops are performed over each group, and the compiler can be instructed by a compiler directive to allow vectorization.

The same difficulty arises with the accumulation of dissipative terms by loops over the edges. This may be overcome in a similar manner by separating the edges into groups. The faces and edges can be separated into suitable groups by sorting procedures which need be performed only once after the mesh has been generated. These sorting procedures pose some interesting problems, presently unresolved, such as how to obtain the smallest possible number of groups, or a set of groups such that the smallest group of the set has the largest possible size.

6. Triangulation

The connection of a cloud of points to produce a tessellation by tetrahedra is clearly non-unique. However, it is always possible to join the points so that the sphere through the four points forming any tetrahedron contains no other points. This unique tesselation is known as the Delaunay triangulation[19] and has found application in diverse fields.[25,26]

The geometric dual of the Delaunay triangulation is the Voronoi diagram[20] which marks off the region of space closer to each point than any other. Thus, the Voronoi neighborhood of a point P_i is defined as the region of space

$$V_i = \{x \mid d(x,P_i) < d(x,P_j), \quad \forall\, i \neq j\}$$

Here x is a point in space and $d(x,P_i)$ is the distance from x to P_i. The boundary of the Voronoi neighborhood is formed from segments of the bisectors of the lines joining P_i to each of the surrounding points. This is illustrated for a two dimensional triangulation in Figure 5. The aggregate of these boundaries make up the Voronoi diagram and, in general, a vertex of the Voronoi diagram is formed where three edges meet in the two dimensional case. In three dimensions, the Voronoi diagram is composed of planar facets and, in general, four edges meet at each vertex of the Voronoi diagram. Since each facet of the Voronoi diagram is equidistant from two points, a vertex is equidistant from four points in three dimensions and is therefore the circumcenter of the sphere passing through the four points. The Delaunay triangulation is precisely the triangulation formed by joining each point to its neighbor across each facet of the Voronoi diagram.

Several algorithms have been proposed for creating the Delaunay triangulation of a set of points in the plane.[21-24] In order to achieve an efficient algorithm some ingenious techniques have been adopted. These include divide-and-conquer strategies[23] and, more recently, a sweepline approach.[24] None of these algorithms, however, extends easily to three dimensions. The algorithm we have used is therefore based on the method of Bowyer[22], which can be readily applied to any number of dimensions. The process is incremental; each new point is introduced into the existing structure which is broken and then reconnected to form a new Delaunay triangulation. This is illustrated in Figure 6(a) where the new point P is found to lie inside the circles through triangles ABD and BCD. These two triangles are therefore removed to leave the cavity ABCD. The point P is then joined to points A, B, C and D on the boundary of the cavity (figure 6(b)). The legitimacy of this procedure is based on the fact the cavity of broken triangles will always be star shaped and that the new triangles formed by the reconnections also satisfy the Delaunay criterion.

Suppose first that the cavity of broken triangles were not star shaped. In particular, we consider the situation, shown in Figure 7(a), in which the line joining the new point P to the cavity boundary point D passes through part of the unbroken structure. The broken triangle, whose base was CD, would have had either E or F as its third forming point. However, the circle circumscribing that triangle could not contain the point P, since P lies on the opposite side of the edge CD from points E and F. Thus the triangle formed by the edge CD and either point E or point F could not have been broken by the new point P. It follows that D must be an interior point of the unbroken region and not a boundary point of the cavity. In other words, the cavity formed precisely from those triangles broken by the new point, must necessarily be star shaped.

We now consider the triangles formed

by the reconnections. In particular, consider the triangle ABC in Figure 7(b) that is in the unbroken region with edge AC as part of the cavity boundary. The new point P lies strictly outside the circle ABC and hence circle PAC does not contain point B. Since this condition holds for all unbroken triangles, which have edges on the cavity boundary, it follows that circles circumscribing the new triangles do not contain any other points.

To implement Bowyer's algorithm it is necessary to maintain a list of the four forming points for each extant tetrahedron and a list of the four tetrahedra which have a face in common with each extant tetrahedron. When a new point is introduced, a search is made to find the first tetrahedron that is broken. This occurs if the distance from the new point to the circumcenter is less than the radius of the circumscribing sphere. When a first broken tetrahedron has been found, the whole cavity can be obtained by a tree search through neighboring tetrahedra.

The number of triangles that are broken by the introduction of a new point will depend on the order in which the points are introduced. Likewise, the time taken to find the first broken triangle will depend critically on the order in which the triangles are stored and searched. For example in Figure 8, we illustrate a situation which results in the creation of very large broken triangle cavities. In this example, a new point introduced into the hatched region will lie inside all circles and hence cause all existing triangles to be broken. The worst case cavity could result in O(n) triangles being broken, and likewise, a worst case search could take O(n) time. If either situation occurs for each new point we obtain a worst case time complexity of $O(n^2)$, which quickly leads to prohibitively large computation times.

It is apparent from Figure 8 that any point introduction procedure, which inserts successive points close together, is likely to be inefficient. It is therefore necessary to introduce a coarse sprinkling of points followed by a finer distribution. More precisely, we employ a strategy of successive mesh refinement in much the same manner that one defines a sequence of grids for a multigrid algorithm. Consider, for example, a regular lattice of points that have already been triangulated (shown in Figure 9 by the circles). We can now introduce the next level of refinement, in which new points (shown by crosses in Figure 9) are spaced at half the mesh interval of the old mesh level. If the new point falls in the middle of one of the old lattice squares, then only two old triangles are broken; if it falls on the mid point of an old lattice side, then two squares or four old triangles will be affected. It is therefore apparent that the cavity of broken triangles will always remain O(1) and will never approach a worst case O(n).

It is still possible that the time taken to find the first broken triangle will be O(n) unless further measures are taken. We therfore introduce the new points in a forward and backward line by line sweep across the lattice, as indicated by the arrows in Figure 9. This particular sweep direction together with the fact that each new point is only half an old lattice width from the previously introduced point, ensures that at least one of the newly created triangles is among those broken by the latest point. If we start the search for a broken triangle by examining those triangles that were created by filling the previous cavity, we can also ensure that the time taken to find the first broken triangle is O(1). Thus, we anticipate an expected overall time complexity of O(n) to carry out the triangulation of n points. In general, there will not be a constant spacing between the points in the mesh. However, the strategy described above still works extremely well.

The combination of the above point introduction procedure and a restructuring of some of the coding has led to a dramatic improvement in triangulation efficiency. This is well illustrated by the triangulation that was carried out for the airplane calculation presented in reference 17. At that time the triangulation algorithm did not incorporate the efficient procedure for introducing points, and the triangulation of a mesh containing 12,000 points around an aircraft took eight hours on a CRAY X-MP. The application of the above point introduction strategy, together with the use of flags to eliminate the need for some redundant searches, has contributed to a reduction of the triangulation time for this case from eight hours to eight minutes.

7. Integrity of Solid Surfaces

The triangulation algorithm joins all points to cover completely the space inside the covex hull of the outermost points. It is necessary to identify and remove all tetrahedra which form the object around which the flow is being

calculated. For a simple convex object, such as a symmetric wing, it is merely necessary to identify those triangles, all of whose forming points are on the object. In geometrically more complicated cases, the identification procedure is more difficult.

Let the set of points {P} be denoted by V and suppose that the triangulation procedure has generated n tetrahedra or simplices $\{S_i\}_{i=1}^{n}$. Each S_i contains four distinct points and

$$V = \{P | P \in S_i \quad \text{for } i=1,\ldots,n\}$$

Let $B \subset V$ be the set of points which make up the object (these can be either surface points or points lying inside the object). Now consider a division of B into m subsets $\{C_k\}_{k=1}^{m}$ where C_k corresponds to the kth component (wing, fuselage, tail, etc.), and a further collection of subsets $\{I_{jk}\}_{j,k=1,j\neq k}^{m}$ where I_{jk} contains points on the interface between component j and component k. We wish to select those simplices which make up each component. Specifically, for each component we form a set containing those simplices all of whose forming points belong either to that component or lie on an interface with an adjacent component. Thus for each k=1 to m we form the set

$$F_k = \{S_i | P \in C_k \text{ or } P \in \cup_j I_{jk}, \quad \forall P \in S_i\}$$

The union $\cup_{k=1}^{m} F_k$ contains tetrahedra which make up the object. To ensure that this collection of tetrahedra exhausts all possible tetrahedra which make up the object, it is necessary to prevent triangulations that create connections between points lying on two different components (for example the connection AC in figure 10(a)). It is therefore necessary to introduce a sufficient density of internal points on each interface (for example, point P in figure 10(b)) to achieve a valid triangulation.

A further difficulty arises in preserving the integrity of the bounding surface. For example, in Figure 11(a) we illustrate an airfoil together with triangles and accompanying circumcircles. The hatched area indicates a region of space outside the airfoil that lies within these Delaunay circles (spheres in 3-D). If a new point is now introduced inside the hatched region, one or more of the triangles will be broken. This situation is shown in Figure 11(b), where a new point P, which lies above the upper surface, has connected to the point E on the lower surface. The segment AB of the airfoil surface has thus been removed and the triangles APE and BPE will be treated as belonging to the flow field. Likewise a new point Q will connect from outside the lower surface to the upper surface point B. The opportunities for surface breakthrough are much more plentiful in three dimensions where elongated tetrahedra that form around wing leading edges can create large Delaunay spheres. A wing leading edge breakthrough is illustrated in Figure 12.

In order to prevent this problem, the object (i.e. surface or internal) points and farfield points are introduced first, and at this stage the tetrahedra, which compose the object, are identified. The remaining points are then introduced by the successive refinement procedure described above. If the introduction of a flow field point would break an object tetrahedron and thereby cause a surface breakthrough, the new point is rejected. It is apparent (see for example Figure 11) that this strategy, while preventing surface breakthrough, will not admit points close to the surface unless all Delaunay spheres of the object tetrahedra are kept small enough. In two dimensions this can usually be achieved fairly easily by taking a dense enough distribution of surface points. In three dimensions this naive approach does not always lead to smaller Delaunay spheres, and a more systematic procedure must be found.

The triangulation starts by first introducing the farfield points and then the aircraft surface points, component by component. After each component has been introduced, the tetrahedra are scanned to discover which of them belong to the most recent component. The list of tetrahedra which form the component is further examined to determine which tetrahedra have Delaunay spheres exceeding a specified threshold. When a tetrahedron has been flagged as exceeding the threshold, an extra point is introduced, the new triangulation computed and the set of component tetrahedra re-determined. This procedure can be iterated until all Delaunay spheres have been reduced to an acceptable size. After all points on the surface and inside of every aircraft component have been triangulated, the flow field points are introduced.

The deleted region, associated with a new point, is now examined and if the deleted region contains tetrahedra belonging to an aircraft component, the point is rejected. Provided the threshold on the allowable size of the component Delaunay spheres is kept sufficiently small, then almost all points which genuinely lie outside the aircraft will be accepted. Of course, any points that fall inside the aircraft structure will be detected and rejected by this process.

8. Results

Our method has now matured to the point where we believe that it is capable of predicting realistic flow patterns, provided that adequate computational resources are deployed. In figures 13 and 14 we present the result of a transonic flow calculation for a Boeing 747-200 flying at Mach 0.84 and an angle of attack of 2.73 degrees. Flow is allowed through the engine nacelles and no attempt has been made to simulate powered engine effects. The flow has been calculated over one half of the aircraft; the mesh shown in figure 13 and the surface pressure contours displayed in figure 14 have been reflected about the aircraft plane of symmetry.

One set of mesh points was generated for the combination of wing/body/tail/fin and another set of mesh points was generated for each nacelle. Details of the sequential mapping technique used to generate the initial mesh can be found in references 14 and 17. Finally a set of points was defined on the surface of each pylon, including points lying along the wing/pylon and nacelle/pylon intersections. The union of mesh points was connected to form an unstructured mesh of tetrahedra by the triangulation module. The total number of points, including extra points added to improve the surface triangulation, is 35370. The mesh is comprised of 181952 tetrahedra and a total of 369854 faces, leading to a memory requirement for this case of just under twelve megawords.

This example has been run on a CRAY XMP 2/16 using only one of the processors. The mesh generation and triangulation took 1100 seconds. The flow solution was calculated with 400 time steps in a further 3300 seconds, giving a total computation time of about $1\frac{1}{4}$ hours. The flow solution was obtained using a three stage scheme with implicit residual averaging and enthalpy damping. The nominal Courant number was five and the average residual reduction rate over 400 cycles was 0.97. Details of the convergence history are presented in the table below.

Computed pressure contours on the aircraft surface are presented in figure 14. All the significant flow features are predicted, including the supersonic regions on the wing upper surface and fuselage, the interference effects of the wing and tail on the fuselage and interference between the wing, nacelles and pylons.

The development of error free software for complex problems is itself a major task which needs to be addressed with great care. Our approach has been to develop a modular program, in which the principal functional modules are almost entirely independent of each other. In its present form the program has four main modules:

1) Mesh point generator
2) Triangulator
3) Mesh post processor
4) Flow solver

The mesh point generator generates the cloud of mesh points around the entire configuration by superimposing separately generated meshes for the principal components. The triangulator connects the points to form tetrahedra while preserving the integrity of the surface of the configuration. The mesh post processor identifies and labels the faces and edges. It also sorts them into groups to allow vectorization of the solution algorithm. The flow solver generates the solution by integrating the equations until they reach a steady state. It should be noted that modules 3 and 4 can be used to calculate solutions on triangular meshes generated by other methods. (They have actually been used to calculate a solution for a Falcon 20 on a mesh provided by Avions Marcel Dassault).

We anticipate that the four existing modules could eventually form the core of a software ensemble for universal flow prediction. Initially the current modules would be embedded in a package of the type illustrated in Figure 15. This would contain a preprocessor to interface with standard data bases such as those used for CADAM, CATIA and NASTRAN, and a post processor to generate appropriate graphical displays of the results (currently we are using software available at Cray Research such as Movie BYU). Ultimately a more complex software package would include options to treat viscous effects, either by coupling the inviscid method to a boundary layer analysis, or by treatment of the Reynolds averaged Navier Stokes

equations with appropriate turbulence models. Such a package might also include a "zoom" capability for more detailed analysis of the flow in particular regions of interest, or quick analysis of isolated components.

While the method is capable of computing transonic flows over aircraft, it is not inherently restricted in any way to aeronautical applications. In particular, we anticipate that with the viscous terms included it might be used to advantage for the predicition of flows past automobiles and trucks. A preliminary Navier Stokes solver is already under development.

Ackowledgments

Development of the airplane computer program has been carried out on computers belonging to the Cray Research Corporation. We are extremely grateful to Cray Research for providing us with substantial access to their computers and, in particular, we should like to thank Kent Misegades for his help and support. Considerable financial support for our work has been provided by the IBM Corporation, the office of Naval Research and NASA Langley Research Center.

References

1. Hess, J. L. and Smith, A. M. O., "Calculation of Non-Lifting Potential Flow About Arbitrary Three Dimensional Bodies", Douglas Aircraft Report, ES 40622, 1962.

2. Rubbert, P. E. and Saaris, G. R., "A General Three Dimensional Potential Flow Method Applied to V/STOL Aerodynamics", SAE Paper 680304, 1968.

3. Murman, E. M. and Cole, J. D., "Calculation of Plane Steady Transonic Flows", AIAA Journal, Vol. 9, 1971, pp. 114-121.

4. Jameson, A., "Iterative Solution of Transonic Flows Over Airfoils and Wings, Including Flows at Mach 1", Comm. Pure. Appl. Math, Vol. 27, 1974, pp. 283-309.

5. Jameson, A. and Caughey, D. A., "A Finite Volume Method for Transonic Potential Flow Calculations", Proc. AIAA 3rd Computational Fluid Dynamics Conference, Albuquerque, 1977, pp. 35-54.

6. Bristeau, M. O., Pironneau, O., Glowinski, R., Periaux, J., Perrier, P. and Poirier, G., "On the Numerical Solution of Nonlinear Problems in Fluid Dynamics by Least Squares and Finite Element Methods (II). Application to Transonic Flow Simulations", Proc. 3rd International Conference on Finite Elements in Nonlinear Mechanics, FENOMECH 84, Stuttgart, 1984, edited by J. St. Doltsinis, North Holland, 1985, pp. 363-394.

7. Jameson, A., Schmidt, W. and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes", AIAA Paper 81-1259, AIAA 14th Fluid Dynamics and Plasma Dynamics Conference, Palo Alto, 1981.

8. Ni, R. H., "A Multiple Grid Scheme for Solving the Euler Equations", Proc. AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, 1981, pp. 257-264.

9. Jameson, A. and Baker, T. J., "Solution of the Euler Equations for Complex Configurations", Proc. AIAA 6th Computational Fluid Dynamics Conference, Danvers, 1983, pp. 293-302.

10. Jameson, A. and Baker, T. J., "Multigrid Solution of the Euler Equations for Aircraft Configurations", AIAA 22nd Aerospace Sciences Meeting, Reno, Nevada, AIAA 84-0093, January 1984.

11. Jameson, A., "Multigrid Algorithms for Compressible Flow Calculations", Proceedings of 2nd European Conference on Multigrid Methods, in Lecture Notes in Mathematics, Vol. 1228, edited by Hackbusch and Trottenberg, Springer-Verlag, 1986.

12. Pulliam, T. H. and Steger, J. L., "Recent Improvements in Efficiency, Accuracy and Convergence for Implicit Approximate Factorization Algorithms", AIAA Paper 85-0360, AIAA 23rd Aerospace Sciences Meeting, Reno, January 1985.

13. MacCormack, R. W., "Current Status of Numerical Solutions of the Navier-Stokes Equations", AIAA Paper 85-0032, AIAA 23rd Aerospace Sciences Meeting, Reno, January 1985.

14. Baker, T. J., "Mesh Generation by a Sequence of Transformations", to appear in Applied Numerical Mathematics, December, 1986, Princeton University Report MAE 1739.

15. Löhner, R., Morgan, K., Peraire, J. and Zienkiewicz, O. C., "Finite Element Methods for High Speed Flows", AIAA Paper 85-1531, AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, Ohio, July 1985.

16. Jameson, A. and Mavriplis, D., "Finite Volume Solution of the Two Dimensional Euler Equations on a Regular Triangular Mesh, AIAA Paper 83-0435, AIAA 23rd Aerospace Sciences Meeting, Reno, January 1985.

17. Jameson, A., Baker, T. J. and Weatherill, N. P., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft", AIAA Paper 86-0103, AIAA 24th Aerospace Sciences Meeting, Reno, January, 1986.

18. Mavriplis, D. and Jameson, A., "Multigird Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes", AIAA Paper 87-0353, AIAA 25th Aerospace Sciences Meeting, Reno, January 1987.

19. Delaunay, B., "Sur la Sphere vide", Bull. Acad. Science USSR VII: Class. Sci., Mat. Nat. pp. 793-800, 1934.

20. Voronoi, G., "Nouvelles Applications des Parametres Continus a la Theorie des Formes Quadratiques. Deuxieme Memoire: Recherches Sur les Parallelloedres Primitifs", J. Reine Angew. Math., Vol. 134, pp 198-287, 1908.

21. Watson, D. F., "Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes", The Computer Journal, Vol. 24, No. 2, pp. 167-172, 1981.

22. Bowyer, A., "Computing Dirichlet Tessellations", The Computer Journal, Vol. 24, No. 2, pp. 162-166, 1981.

23. Shamos, M. I., "Geometric Complexity", Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, pp. 224-233, 1975.

24. Fortune, S., "A Sweepline Algorithm for Voronoi Diagrams", Proc. of the 2nd Computational Geometry Conference, Yorktown Heights, NY, June 1986.

25. Augenbaum, J. M., "A Langrangian Method for the Shallow Water Equations Based on a Voronoi Mesh-One Dimensional Results", J. Comp. Physics, Vol. 53, No. 2, Feburary 1984.

26. McCartin, B., "Discretization of the Semiconductor Device Equations", in New Problems and New Solutions for Device and Process Modeling, pp. 72-82, Bode Press, 1985.

CYCLE	AVERAGE RESIDUAL	AVERAGE ENTHALPY	NUMBER OF SUPERSONIC POINTS
1	2.42×10^{2}	4.84×10^{-2}	0
50	8.65×10^{0}	1.20×10^{-2}	1765
100	1.40×10^{0}	3.86×10^{-3}	2278
150	6.32×10^{-1}	2.48×10^{-3}	2372
200	1.93×10^{-1}	1.31×10^{-3}	2408
250	7.80×10^{-2}	4.45×10^{-4}	2418
300	3.76×10^{-2}	1.66×10^{-4}	2425
350	1.67×10^{-2}	2.93×10^{-5}	2425
400	5.13×10^{-3}	1.68×10^{-5}	2423

Table showing convergence history for complete aircraft calculation

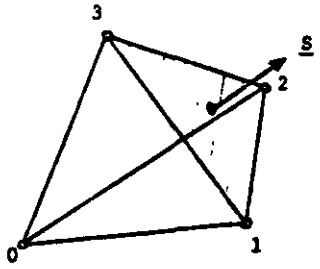


Figure 1

One tetrahedron of the control volume centered at node 0.

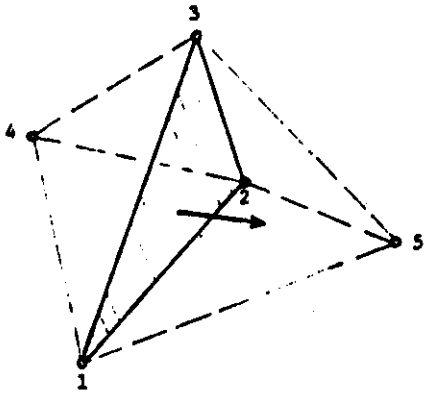


Figure 3

Flux through face defined by nodes 1, 2 and 3 is out of the control volume centered at node 4 and into the control volume centered at node 5.

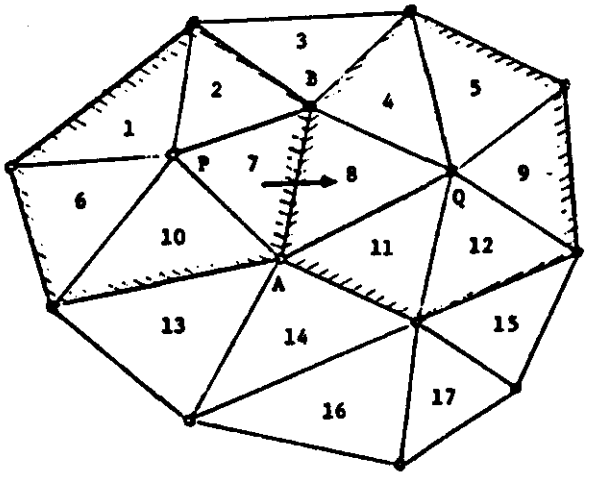


Figure 2

A triangular mesh in 2 dimensions: The control volume at P is the union of triangles 1, 6, 10, 7 and 2, while that at Q is the union of triangles 4, 8, 11, 12, and 9. The flux across the edge AB is from the control volume at P to the control volume at Q.

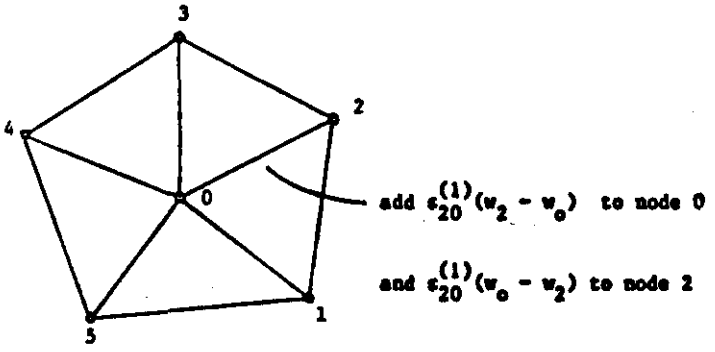


Figure 4

Construction of dissipation from differences along edges in a two dimensional mesh.

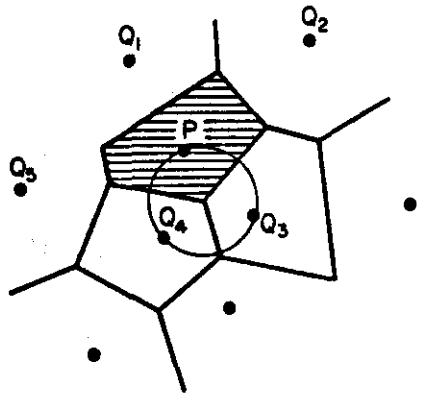


Figure 5(a)

Voronoi Neighborhood around point P

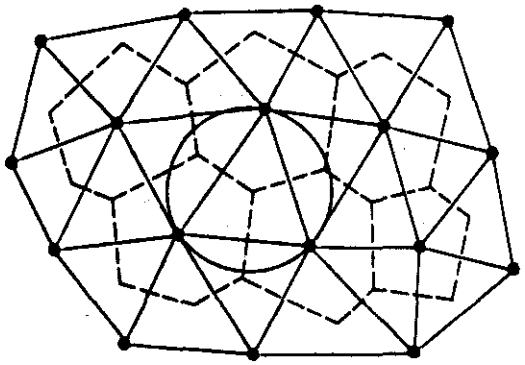


Figure 5(b)

Delaunay Triangulation and Voronoi Diagram

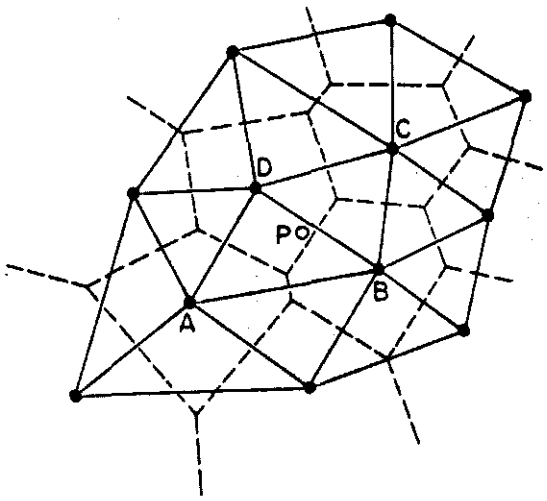


Figure 6(a)

Delaunay Triangulation with new point P prior to re-triangulation

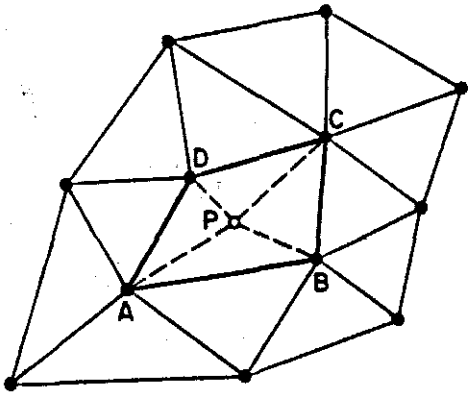


Figure 6(b)

Re-triangulation to include new point P

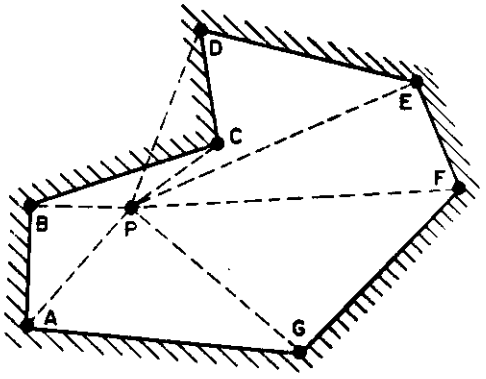


Figure 7(a)

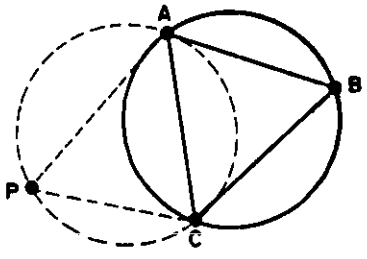


Figure 7(b)

Connections between new point and points on boundary of deleted cavity

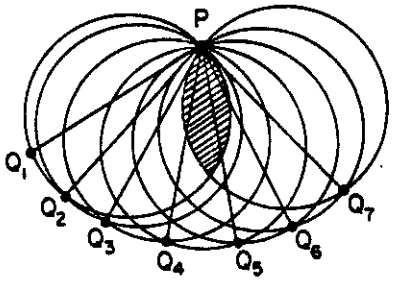


Figure 8

Example of inefficient point introduction

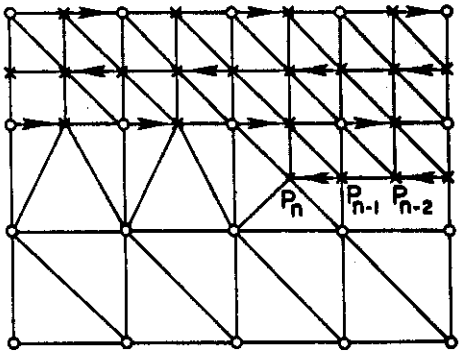


Figure 9

Introduction of points by refinement process

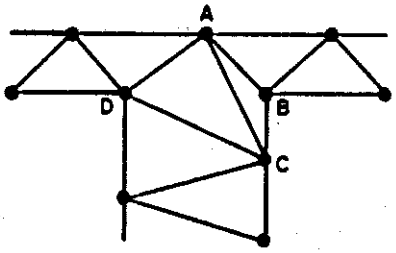


Figure 10(a)

Triangulation of two components with connection AC across interface

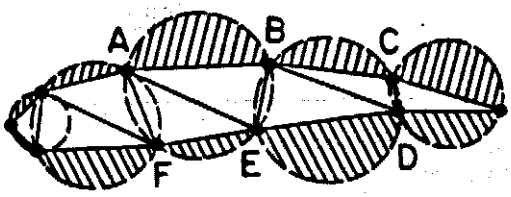


Figure 11(a)

Delaunay Spheres for triangles forming on airfoil

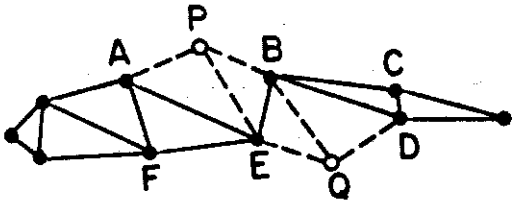


Figure 11(b)

Surface breakthrough by connections to points outside airfoil

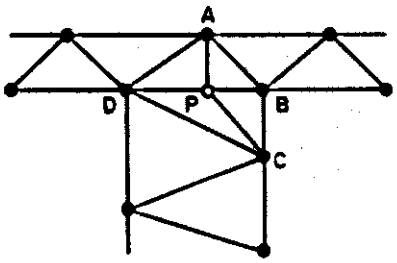


Figure 10(b)

Triangulation of two components with no connections across interface

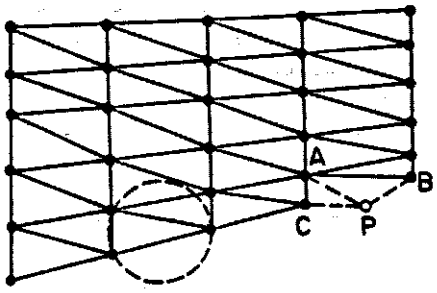


Figure 12

Breakthrough at a wing leading edge

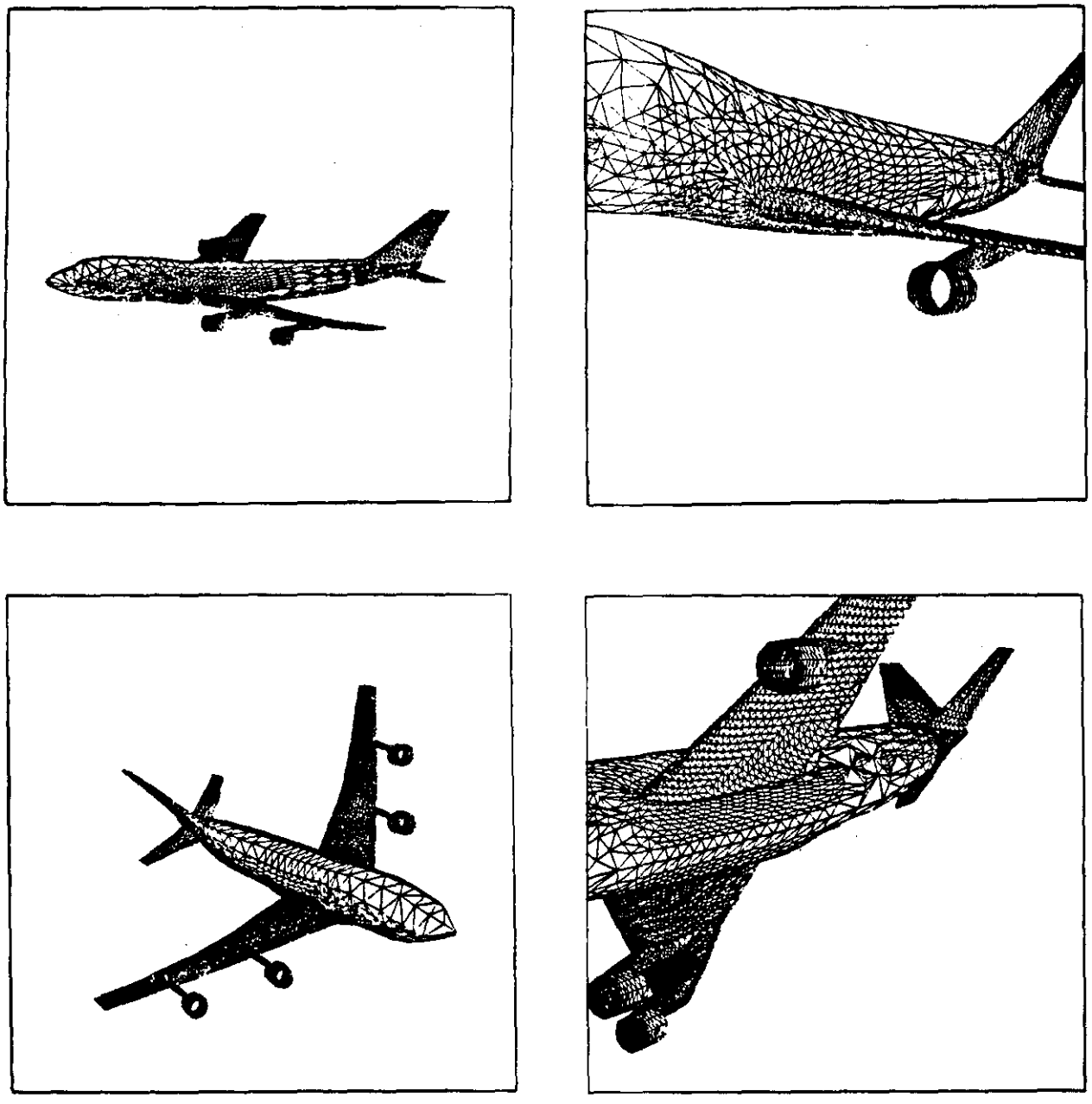


Figure 13

Mesh for the Boeing 747-200

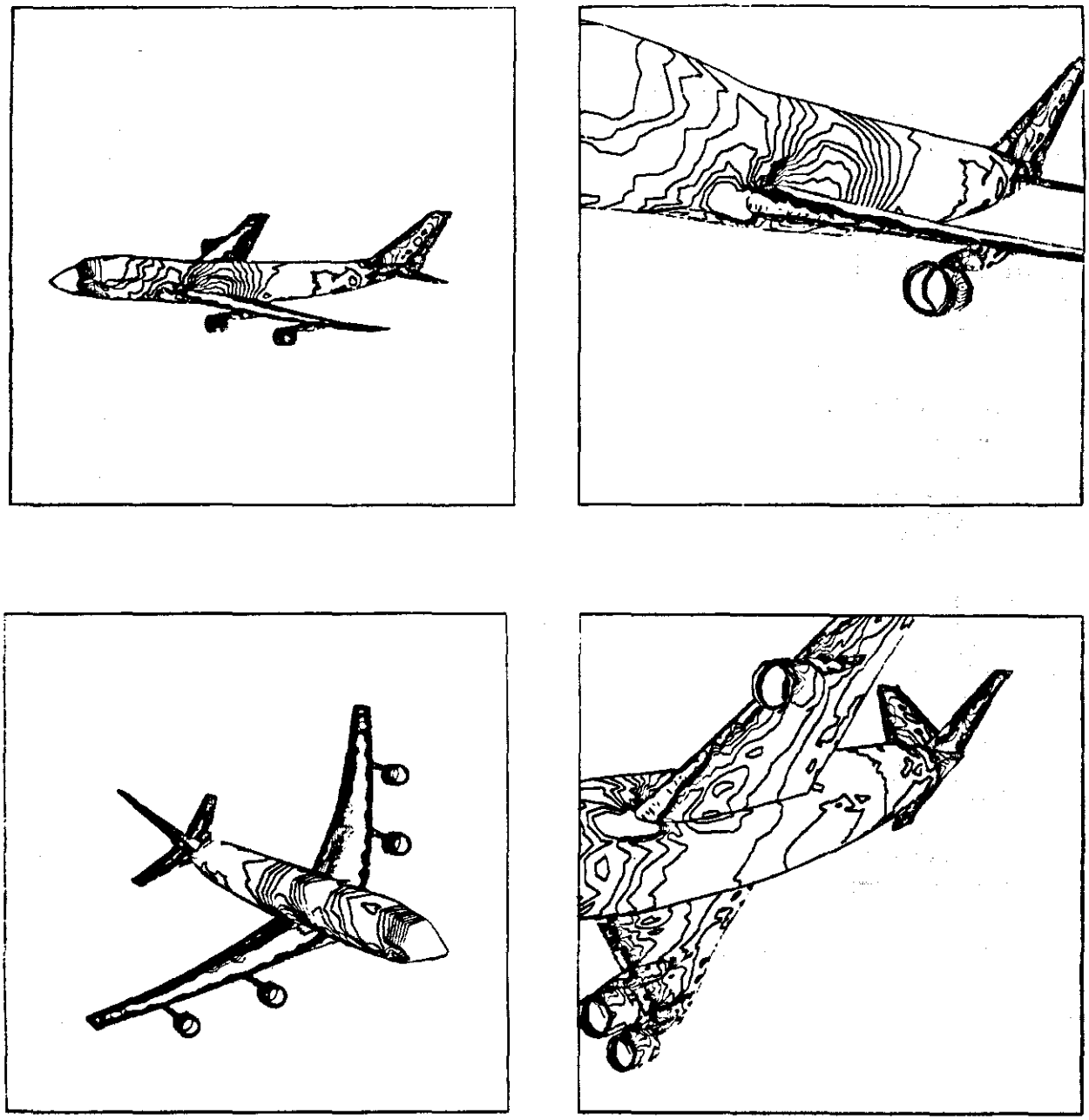
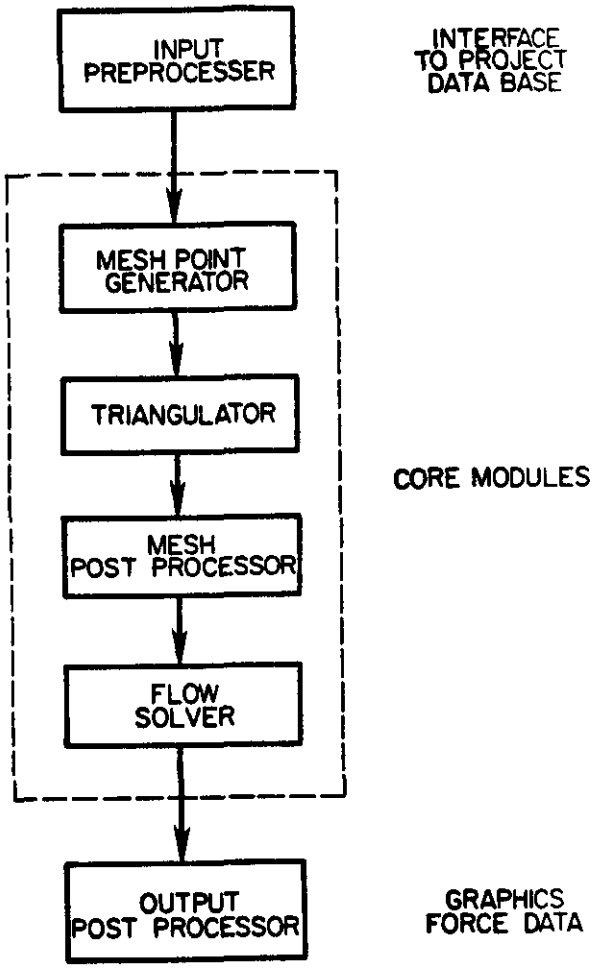


Figure 14

Surface pressure contours for the Boeing 747-200
Mach .84 Angle of attack 2.73 degrees



MODULAR PROGRAM STRUCTURE

FIGURE 15