JOURNAL OF COMPUTATIONAL PHYSICS 92, 1–66 (1991)

A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics*

DAVID P. YOUNG, ROBIN G. MELVIN, AND MICHAEL B. BIETERMAN

Boeing Computer Services, M/S: 7L-21, P.O. Box 24346, Seattle, Washington 98124-0346

FORRESTER T. JOHNSON AND SATISH S. SAMANT

Boeing Commercial Airplanes, M/S: 7C-36, P.O. Box 3707, Seattle, Washington 98124-2207

AND

JOHN E. BUSSOLETTI

Boeing Advanced Systems, M/S: 7C-36, P.O. Box 3707, Seattle, Washington 98124-2207

A new finite element method for solving important linear and nonlinear boundary value problems arising in computational physics is described in this paper. The method is designed to handle general three-dimensional regions, boundary conditions, and material properties. The boundaries are described by piecewise planar surfaces on which boundary conditions are imposed. The method uses box finite elements defined by a Cartesian grid that is independent of the boundary definition. Local refinements are performed by dividing a box element into eight similar box elements. The discretization uses trilinear approximations on the box elements with special element stiffness matrices for boxes cut by any boundary surface. This discretization process is automated and does not require the generation of a boundary conforming grid. The resulting (possibly nonlinear) discrete system is solved using a preconditioned GMRES algorithm. The primary preconditioner is a sparse matrix solver using a dynamic drop tolerance in the decomposition phase. Results are presented for aerodynamics problems with up to 400,000 elements. demonstrating the accuracy and efficiency of the method.
© 1991 Academic Press, Inc.

1. INTRODUCTION

Many engineering designs are geometrically complex. Commercial aircraft, for example, have nacelles, stabilizers, slats, flaps, and ailerons, in addition to wings and fuselages. The geometry of typical military aircraft can be even more complex.

* Partially supported by NASA Contract NAS2-12513.

The design of such complex configurations presents challenging mathematical modeling problems in aerodynamics, acoustics, electromagnetics, and structures. Often, these problems take the form of partial differential equations on unbounded domains with many types of boundary conditions depending on the application. Also, geometric and physical complexity of the problem requires a large number of degrees of freedom. For transonic flow about a complete aircraft involving shocks and slip surfaces, several hundred thousand degrees of freedom are required by standard second-order finite element, finite volume, or finite difference methods to produce accurate answers in the entire flow field.

A computational method for such problems should be reliable, accurate, flexible, and efficient. In two space dimensions, for elliptic boundary value problems, adaptive finite element methods can meet such criteria [1–4]. In three space dimensions, panel methods (boundary integral methods) can have the desired degree of geometry flexibility, but are limited to linear flow [5]. In this paper, the full potential equation was chosen because it models important nonlinear effects and was well enough understood to allow development of a truly reliable engineering tool.

Many methods in aerodynamics have used body fitted structured or block structured grids that allow each grid cell to be treated in a similar fashion with minimal storage [6–9]. This introduces the subsidiary problem of generating a body conforming grid. A second approach that allows arbitrary refinement is to use tetrahedral finite elements [10–14]. This approach also requires the generation of a body conforming mesh.

A third approach uses Cartesian grids with special operators near the boundary [15–20]. We have chosen to use this approach because the body fitted grid generation problem is eliminated. The method presented in this paper is a finite element method that uses rectangular box elements. The finite element formulation enables us to construct special boundary operators easily. The grid is based on a uniform Cartesian global grid with local refinement. Because problems of practical interest often have many different length scales, this local grid refinement is essential. A refinement is made by dividing a given box element into eight similar boxes. In this way every element that does not intersect the boundary is geometrically similar and has an identical element stiffness matrix, minimizing storage.

An important feature of many of the problems we consider is the presence of a non-local far field condition, e.g., the Sommerfeld radiation condition. The use of a Green's function defined on a uniform global grid in conjunction with the fast fourier transform allows the boundary of the computational grid to be very close to the object [20–23]. This greatly reduces the number of finite elements needed to solve a given problem.

To ensure robust convergence, we have developed a solution procedure for the discrete equations that employs a combination of two preconditioners for a GMRES driver [24, 25]. One of these preconditioners is a sparse direct solver with a drop tolerance [26]. The other is a Poisson solver on the uniform global grid that ensures that the far field boundary condition is satisfied [27].

In Section 2 we define the aerodynamics problem to be considered. Section 3 contains a discussion of the discretization. Section 4 describes the octtree data structure used to store the refinements. Section 5 gives the solution procedure for linear problems. Section 6 discusses the sparse linear algebra used in one of the preconditioners. Section 7 presents computational results for linear problems in aerodynamics. Section 8 contains a description of the iterative algorithm used for the nonlinear full potential equation of aerodynamics. Section 9 presents results for some aerodynamic transonic flow cases. The computational cost of our implementation of the method is discussed in Section 10. Section 11 discusses two other application areas, acoustics and electromagnetics, where this method has been applied. Section 12 discusses useful features not described in this paper. This is followed with concluding remarks.

2. Problem Definition

The full potential equation of aerodynamics is

$$\mathscr{F}(\Phi) \equiv \nabla \cdot \rho \nabla \Phi = 0, \tag{1}$$

where the density is given by

$$\rho = \rho_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - \frac{q^2}{q_\infty^2} \right) \right]^{1/(\gamma - 1)}. \tag{2}$$

Here, with $\mathbf{V}_\infty$ taken to be a uniform onset flow, Φ is the total velocity potential to be determined, $q = \|\nabla\Phi\|_2$ is the local speed, $q_\infty = \|\mathbf{V}_\infty\|_2$ is the freestream speed, ρ_∞ is the freestream density, M_∞ is the freestream Mach number, and γ is the ratio of specific heats. Equation (1) describes irrotational compressible flow. In addition, boundary conditions are required to define a well-posed problem. The far field condition is

$$\phi = \mathcal{O}\left(\frac{1}{R}\right) \tag{3}$$

as $x \to -\infty$, i.e., upstream of the object. Here, the perturbation potential is given by $\phi = \Phi - \Phi_\infty$, where $\nabla\Phi_\infty = \mathbf{V}_\infty$. On impermeable surfaces, the normal mass flux condition is $\rho(\partial\Phi/\partial n) = 0$. On other surfaces, such as engine inlets, the normal mass flux is a specified function, $\rho(\partial\Phi/\partial n) = g_1$. On other surfaces we impose the Dirichlet condition $\Phi = g_3$. On engine exhaust surfaces, tangential flow can be prohibited by specifying g_3 to be constant. Wake surfaces must extend downstream from lifting components such as wings. These surfaces allow nonzero circulation in potential flow and can be thought of as thin sheets of concentrated vorticity [28]. The boundary conditions on a wake are

$$\hat{n} \cdot \Delta(\rho \, \nabla\Phi) = 0 \tag{4}$$

and

$$\Delta p = 0, \tag{5}$$

where

$$p = p_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - \frac{q^2}{q_\infty^2} \right) \right]^{\gamma/(\gamma-1)} \tag{6}$$

and Δ represents the jump across the wake surface. Equations (4) is an expression of conervation of mass across the wake. Equation (5) is required for conservation of normal momentum. Equation (5) is often linearized about the freestream pressure $p = p_\infty$, assuming small perturbation velocity $\nabla\phi$. This leads to the equivalent Dirichlet condition that $\Delta\Phi$ is constant along the wake in the direction of $\mathbf{V}_\infty$. The circulation μ at the trailing edge is determined by a Kutta condition imposed there.

The full potential equation is a consequence of the Bateman variational principle, namely, that the integral of pressure over the flow field is stationary [29]. This principle can be used to derive finite element formulas for the full potential equation as described below in Section 3.4. In subsonic flow, the Bateman integral is maximized. A generalization of the Bateman variational principle which incorporates the boundary conditions described above is that the functional

$$\begin{aligned} J = & \int_\Omega p \, dV + \int_{\partial\Omega_1} g_1 \Phi \, dS \\ & - \int_{\partial\Omega_2} \alpha \left(\rho \frac{\partial \Phi}{\partial n} \right) (\Delta\Phi - \mu) \, dS \\ & + \int_{\partial\Omega_3} \rho \frac{\partial \Phi}{\partial n} (\Phi - g_3) \, dS \end{aligned} \tag{7}$$

is stationary. Here, g_1 is the given mass flux data on $\partial\Omega_1$, $\Delta\Phi$ is the jump in Φ across the wake surface $\partial\Omega_2$, μ is the unknown representing the jump in Φ on $\partial\Omega_2$ determined by Eq. (5), α denotes the average of the upper surface and lower surface values, and g_3 is the given Dirichlet data on $\partial\Omega_3$. The function μ is itself unknown and is determined by Eq. (5). To achieve a stable numerical formulation, the treatment of Dirichlet boundary conditions and wake surfaces must be modified. In addition, the natural Neumann condition must be modified to account for boundary curvature, since the solution is often sensitive to this quantity and the boundary is discretized using flat panels. These modifications are described below in Section 3.6.

A minor modification of the above formulation allows the simulation of flows involving regions of differing total temperature and pressure. The flow in each separate region is still potential as long as total temperature and pressure are constant in the region, but pressure and density must be redefined in the following way:

$$p = p_\infty r_p \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - \frac{q^2}{q_\infty^2 r_T}\right)\right]^{\gamma/(\gamma-1)} \tag{8}$$

$$\rho = \rho_\infty \frac{r_p}{r_T} \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - \frac{q^2}{q_\infty^2 r_T}\right)\right]^{1/(\gamma-1)} \tag{9}$$

Here, r_p is the ratio of the total pressure in the region to the freestream total pressure and r_T is the ratio of total temperature in the region to freestream total temperature. The regions are assumed to be separated by fixed wake surfaces on which two jump boundary conditions are applied. The first is the standard static pressure continuity condition equation (5). If the total pressure and/or temperature differences across the wake are large, the pressure formula cannot be linearized, i.e., μ is not constant in the downstream direction. The second condition is similar to Eq. (4) but requires a modification to make the answer less sensitive to wake position when total pressure and temperature differences are large. Equation (4) is replaced by

$$\hat{n} \cdot \Delta \mathbf{W}^* = 0, \tag{10}$$

where

$$\mathbf{W}^* = \frac{\rho_\infty q_\infty}{\rho_0 q_0} \rho \nabla \Phi. \tag{11}$$

Here, q_0 is the velocity magnitude which makes $p = p_\infty$ in the given region and ρ_0 is the density at this velocity. Equation (4) becomes a natural jump boundary condition for $\Phi^* = q_\infty \Phi / q_0$ if the Bateman Principle is modified so that

$$J = \int_\Omega p^* \, dV, \tag{12}$$

where

$$p^* = p \frac{\rho_\infty q_\infty^2}{\rho_0 q_0^2}. \tag{13}$$

3. Discretization

In this section, we describe the discretization of the variational problem using finite elements on a grid consisting of rectangular boxes. First, the computational grid and its generation are described. This is followed by a description of the element trial functions. The restriction to a finite computational grid is then justified. Then, the derivation of element stiffness matrices and the treatment of the density ρ are described. Grid interfaces between different levels of refinement are then discussed, followed by discussions of modifications of the Bateman principle and of artificial dissipation.

3.1. *Computational Grid*

The boundary surfaces of objects to be modeled are described using networks of locally flat surface patches called panels [5]. This input format allows relatively simple specification of complicated surfaces. Various boundary conditions can be specified on these surfaces. Different formulas for density can also be specified on either side of these boundary surfaces. This panel description is identical to the input format of panel method codes that are extensively used for linear flow analysis [5]. Once an input is prepared, a panel method linear flow analysis, a full potential solution, an acoustic analysis, and an electromagnetic analysis (see Section 11) can all be performed. Current geometry software can easily generate panelings that are sufficiently dense to make the piecewise flat panel assumption a negligable source of error compared to other discretization errors. The volume grid, on the other hand, is generated automatically and can be controlled using certain criteria. The process for constructing the grid is described below.

We start with a coarse uniform rectangular grid, called the global grid, that contains all boundary surfaces but is otherwise independent of the boundary surfaces. It is assumed that outside this global grid, the discrete finite element operator can be approximated by some constant coefficient linear operator (see Section 3.3 below). The global grid is used in enforcing the far field condition and for one of the preconditioners (see Section 3.5 below). The global grid is locally refined in a hierarchical manner, i.e., any grid box can be refined into eight geometrically similar boxes of equal volume. This process is repeated to give a grid with any desired local resolution and is controlled by two criteria.

The first criterion for local refinement is based on the length scale of the surface panels used to describe the boundary. Every box element that is sufficiently close to a panel is refined if some weighted length scale associated with the panel is smaller than the length scale associated with that box element. This criterion is effective in providing local refinement near the boundary surface. In this way, it is possible to provide denser surface paneling to effect more local refinement near certain parts of the boundary if it is known that the solution will have stronger gradients in that region.

The second criterion allows refinement away from boundary surfaces. Special "regions of interest or disinterest" can be prescribed, each with desired minimum and maximum refinement levels. All the box elements in these special regions are refined recursively until the minimum level is reached. Further refinement depends on the first criterion. The special regions are hexahedral and provide a fair amount of flexibility in generating off surface refinement. Such refinement is useful in problems where large gradients such as shock waves may exist away from the boundary surfaces. Solution adaptivity significantly augments these two grid refinement criteria and will be described elsewhere.

Once all specified refinements are done, the grid is "legalized" so that two boxes abutting on a face or an edge differ by at most one refinement level. This ensures sparse stencils for the finite element operators and simplifies certain data structures [30], but allows sufficiently rapid changes in grid level.

The locally refined box elements formed by the process described above are usually an unstructured collection of box elements. However, these elements are completely described by an octtree data structure [31, 32], described in detail in Section 4 below, to minimize storage.

3.2. *Element Trial Functions*

The boundary value problem is discretized using a finite element method based on this grid. Every element is identical geometrically except for a scale factor. The element trial function is the standard trilinear one parameterized by eight unknowns, one located at each corner of the box element as shown in Fig. 1. In order to generate as compact a stencil as possible, we wanted the trial functions to generate the standard seven-point operator for Poisson's equation on a uniform grid. This can be accomplished by adding certain second-order terms to the trilinear trial function as given in Fig. 2. These terms introduce no additional degrees of freedom. Moreover, they vanish on all faces of the element and thus do not affect the continuity of the basis.

3.3. *Finite Computational Domain*

The restriction to a finite computational domain can be justified in the following way. Suppose that the partial differential operator $\mathscr{F}$ is equal to a constant coefficient differential operator $\mathscr{T}$ everywhere outside a finite rectangular region. Let $\mathscr{G}$ be a Green's function for $\mathscr{T}$ such that $\mathscr{T}(\mathscr{G} * Q) = Q$ for all Q, where $*$ is the

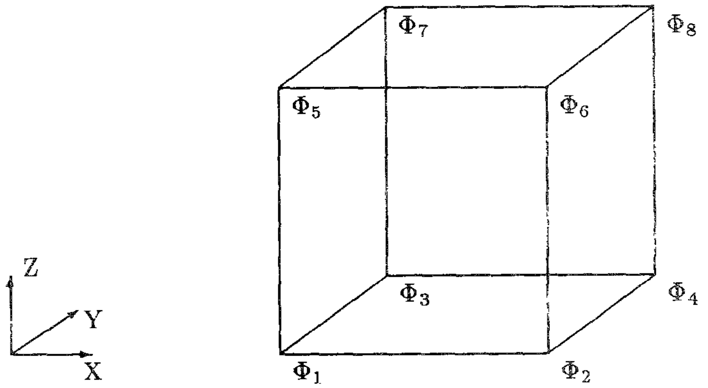


FIG. 1. Box finite element with eight corner unknowns.

$$
\begin{aligned}
\Phi(x,y,z) &= a + bx + cy + dz + exy + fyz + gxz + hxyz \\
&+ \xi_1 \bar{e} F(x)F(y)G(z) + \xi_2 \bar{f} F(y)F(z)G(x) \\
&+ \xi_3 \bar{g} F(x)F(z)G(y) + \xi_4 h F(x)F(y)F(z)
\end{aligned}
$$

where

$$
\begin{aligned}
a &= \Phi_1 \\
b &= \frac{\Phi_2 - \Phi_1}{\Delta x} \\
c &= \frac{\Phi_3 - \Phi_1}{\Delta y} \\
d &= \frac{\Phi_5 - \Phi_1}{\Delta z} \\
e &= \frac{\Phi_4 + \Phi_1 - \Phi_2 - \Phi_3}{\Delta x \Delta y} \\
f &= \frac{\Phi_7 + \Phi_1 - \Phi_3 - \Phi_5}{\Delta y \Delta z} \\
g &= \frac{\Phi_6 + \Phi_1 - \Phi_2 - \Phi_5}{\Delta x \Delta z} \\
h &= \frac{\Phi_8 + \Phi_5 + \Phi_3 + \Phi_2 - \Phi_7 - \Phi_6 - \Phi_4 - \Phi_1}{\Delta x \Delta y \Delta z} \\
F(x) &= x(x - \Delta x)(x - \frac{\Delta x}{2}) \\
G(x) &= x(x - \Delta x) \\
\bar{e} &= e + \frac{\Delta z}{2} h \\
\bar{f} &= f + \frac{\Delta x}{2} h \\
\bar{g} &= g + \frac{\Delta y}{2} h \\
\xi_1 &= K \left[1 + \frac{5(\Delta x \Delta y)^2}{21(\Delta z)^2(\Delta x^2 + \Delta y^2)} \right]^{-\frac{1}{2}} \\
\xi_2 &= K \left[1 + \frac{5(\Delta y \Delta z)^2}{21(\Delta x)^2(\Delta y^2 + \Delta z^2)} \right]^{-\frac{1}{2}} \\
\xi_3 &= K \left[1 + \frac{5(\Delta x \Delta z)^2}{21(\Delta y)^2(\Delta x^2 + \Delta z^2)} \right]^{-\frac{1}{2}} \\
\xi_4 &= \frac{280\sqrt{10}}{(\Delta x \Delta y \Delta z)^2} \\
K &= \frac{120}{(\Delta x \Delta y \Delta z)^2} \sqrt{\frac{35}{6}}
\end{aligned}
$$

FIG. 2. The element trial function in terms of its values at the eight corners of the element.

convolution operator. We further assume that $\Phi = \mathscr{G} * Q$ satisfies the far field condition for all Q which are identically zero outside the finite box. (This will be the case if $\mathscr{G}$ satisfies the far field condition.) Then the original differential equation $\mathscr{F}\Phi = 0$ is equivalent to

$$Q + (\mathscr{F} - \mathscr{T})\mathscr{G} * Q = 0. \tag{14}$$

Outside the finite rectangular region, $\mathscr{F} = \mathscr{T}$ so that $Q = 0$. Thus, the unknowns Q are confined to a bounded region. If the continuous operators are replaced by discrete ones, the same argument holds. Thus, the computational grid can be restricted to the region where the discrete operator F is not approximated well by a discrete far field operator T. The requirement on T is that a discrete Greens function G be available that satisfies an appropriate discrete far field condition and $T(G * Q) = Q$ for all Q. In practice, this means that T is a constant coefficient elliptic operator discretized on a uniform Cartesian grid [20–23, 27]. Thus, for the full potential equation, the computational domain need only cover the region where nonlinear flow occurs. For a wing in transonic flow, the grid typically terminates one or two chord lengths away. (Wakes, which extend to infinity, are an exception which produce sources and sinks that extend to infinity downstream of the configuration. By assuming that such sources and sinks are constant in the downstream direction, their influence can be computed by using a downstream Green's function [27]. This enables the termination of the computational domain a short distance downstream of the configuration.)

The far field operator is taken to be the Prandtl–Glauert operator,

$$\mathscr{T}\Phi = (1 - M_\infty^2)\,\Phi_{xx} + \Phi_{yy} + \Phi_{zz}, \tag{15}$$

which is a linearization of the full potential equation (1) about $\mathbf{V}_\infty$. The discrete operator T is the standard seven-point finite difference discretization of $\mathscr{T}$. The computational domain must include one plane of unrefined global grid boxes on each face of the computational domain, where $F = T$. These unrefined boxes must not be cut by any boundary surface (except boxes on the downstream face of the computational domain which can be cut by wakes) and must remain unrefined throughout the refinement process. This is required because the source Q is assumed to be zero on the boundary points of the global grid. This condition is satisfied if $F = T$ for boxes on the faces of the computational domain.

It is convenient to define three classes of boxes in such a grid. *Far field boxes* are boxes where $F \equiv T$. All such boxes are identical except for a scale factor depending on their level of refinement. We assume that all boxes on the outer boundary of the global grid are far field boxes. *Near field boxes* are boxes not cut by any boundary surface, but where $F \neq T$. Such boxes are by far the most numerous and have identical element stiffness matrices up to a factor depending on the refinement level and the local values of the coefficients of $\mathscr{F}$. *Boundary boxes* are those cut by a boundary surface. Each such box may contain several flow regions (see Section 3.4) each with its own unique element trial function and associated element stiffness matrix.

Due to the nonlinearity of the full potential equation, operators for computing the velocity at the centroids of each flow region within a boundary box are also needed.

3.4. *Finite Element Operators*

Element stiffness matrices are generated by taking variations of the functional J with respect to each degree of freedom and using the local nature of the element basis functions. The density ρ is replaced by a piecewise constant function. In Eq. (8) the pressure p has the property that

$$\frac{\partial p}{\partial u} = -\rho u, \qquad \frac{\partial p}{\partial v} = -\rho v, \qquad \frac{\partial p}{\partial w} = -\rho w, \tag{16}$$

where $(u\ v, w) = \nabla \Phi$. Thus, taking a variation of the volume integral in Eq. (8) is equivalei to taking a variation of the energy functional

$$a(\Phi, \Phi) = -\int_\Omega \rho\, \nabla\Phi \cdot \nabla\Phi\, dV \tag{17}$$

with density fixed. If only natural boundary conditions are present, variations of J are given by

$$\begin{aligned} \delta J &= -\int_\Omega \rho \nabla\Phi \cdot \nabla\, \delta\Phi\, dV \\ &= -\sum_i \int_{\Omega_i} \rho\, \nabla\Phi \cdot \nabla\, \delta\Phi\, dV \\ &\simeq -\sum_i \rho_i \int_{\Omega_i} \nabla\Phi \cdot \nabla\, \delta\Phi\, dV. \end{aligned} \tag{18}$$

Here, ρ_i is the value of ρ at the centroid of the elemental region Ω_i. The last step in Eq. (18) is equivalent to replacing ρ by a piecewise constant function [33]. Equation (18) defines the element stiffness matrices by considering variations of J with respect to each of the eight corner unknowns of the element. Thus, every element not cut by a boundary has the same element stiffness matrix up to a constant factor that depends only on the refinement level of the element and ρ_i. This results in large savings in storage. The multiplying factor ρ_i must be computed each iteration.

The density ρ is a nonlinear function of the velocity and is evaluated at the centroid of each element every iteration. Thus, discrete formulas for velocity at the centroids in terms of the unknowns at the corners of the element are needed. Since all far and near field box elements are similar, only one velocity formula need be stored, resulting in additional large savings in storage. This approximation of the operator coefficients maintains second-order accuracy for the potential in the L_2 norm and first-order accuracy in the energy norm [33].

Since boundary conditions on a surface can induce discontinuities in Φ or $\nabla\Phi$, one element trial function is needed in the boundary boxes for each connected subset of the box. Such connected subsets are referred to as D regions (see Fig. 3). Each D region is bounded by a subset of the boundary surface as well as possibly subsets of the box faces. D regions are not defined in stagnation regions such as the interior of wings or bodies. In such regions, the solution is of no interest and we define the operator $\mathscr{F}$ by $\mathscr{F}\Phi \equiv \phi = 0$ which corresponds to perturbation stagnation. One can merely set $\phi = 0$ at all grid points of boxes completely inside stagnation regions and finite element discretization is not needed. Thus, a box cut by a boundary surface with flow on one side but stagnation on the other would have only one D region corresponding to the flow region, for example, region D_3 in Fig. 3. The element trial function for each such D region is parameterized by unique unknowns at the corners of the grid box. Corner unknowns on the other side of a boundary surface from their D region can be viewed as extrapolated values and are denoted by Ψ. It is therefore possible to have more than one element trial function in a given box and more than one unknown at a grid point. This is the case in Fig. 3 for D regions D_1 and D_2, where a wake divides an element. The Ψ_L unknowns

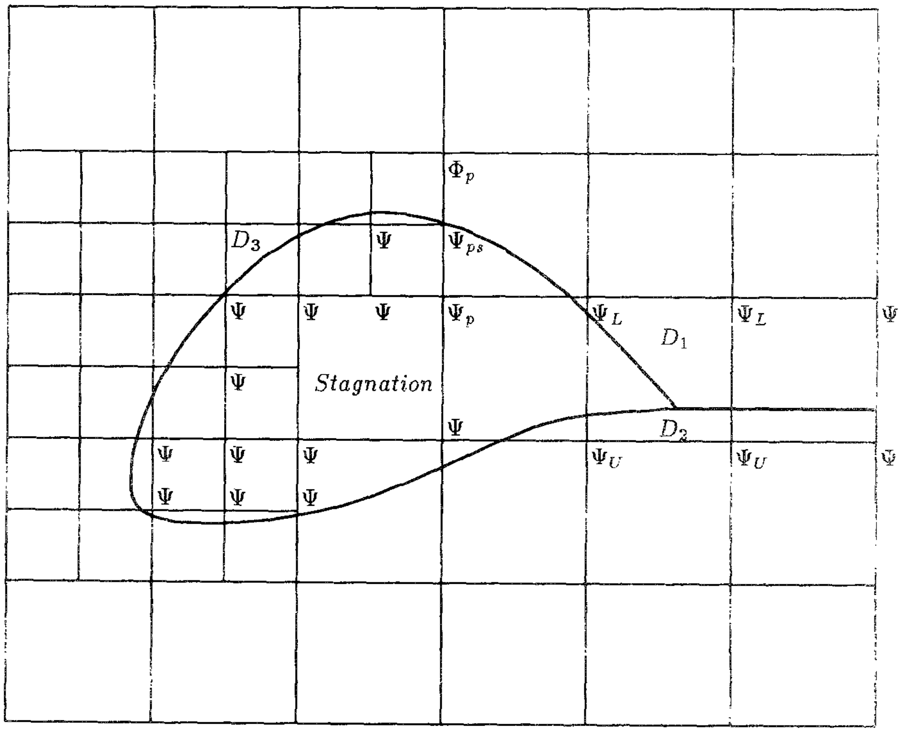


FIG. 3. Placement of unknowns. All grid points have Φ unknowns.

correspond to the element trial function in D_2 and the Ψ_U unknowns correspond to the element trial function in D_1. This represents only a slight complication since each element trial function is still parameterized by eight unique unknowns. There is a one-to-one correspondence between element trial functions in the box and D regions. Each D region also has its own element stiffness matrix. The density is evaluated at the centroid of the D region so that special velocity operators are required. These operators give the velocity at the centroid in terms of the eight corner unknowns associated with the D region.

Each D region has its own distinct element stiffness matrix which must be stored. However, these elements represent typically only 10 to 20% of the elements needed to give an accurate solution of the boundary value problem. Hence, the storage required is acceptable. The element stiffness matrices are derived from an expanded version of Eq. (18) including appropriate surface integral terms. The domain of integration for the volume integral is the region cut off by the relevant boundary surfaces, i.e., the D region in question. The domain for the surface integrals is the part of the surface inside the D region. The integrand is a product of polynomials, which is itself a polynomial. Thus, volume moments must be computed over the D region. We will restrict our attention to the volume moment

$$\mathrm{H}(I, J, K) = \int_D x^{I-1} y^{J-1} z^{K-1}\, dV. \tag{19}$$

Since the boundary is parameterized by piecewise flat panels, this moment can be computed exactly via the following procedure. By Gauss' theorem

$$\mathrm{H}(I, J, K) = \frac{1}{I+J+K} \int_S x^{I-1} y^{J-1} z^{K-1} (\hat{n} \cdot \mathbf{R})\, dS, \tag{20}$$

where S is the bounding surface and $\mathbf{R} = (x, y, z)$. Since S is the union of flat surfaces S_i,

$$\mathrm{H}(I, J, K) = \frac{1}{I+J+K} \sum_i \mathrm{F}_{S_i} \tag{21}$$

$$\mathrm{F}_{S_i} = n_x \mathrm{F}(I+1, J, K) + n_y \mathrm{F}(I, J+1, K) + n_z \mathrm{F}(I, J, K+1), \tag{22}$$

where

$$\mathrm{F}(I, J, K) = \int_{S_i} x^{I-1} y^{J-1} z^{K-1}\, dS. \tag{23}$$

Each S_i is assumed to be polygon whose perimeter is the union of straight lines T_{ij}. Using Stokes' theorem, $\mathrm{F}(I, J, K)$ for the fixed i may be evaluated recursively by the formula

$$F(I, J, K) = \frac{1}{I+J+K-1}\left[(\hat{n}\cdot\mathbf{R})\,F_n + \sum_{l} E_\nu\right] \tag{24}$$

$$F_n = n_x(I-1)\,F(I-1, J, K) + n_y(J-1)\,F(I, J-1, K) + n_z(K-1)\,F(I, J, K-1)$$

$$E_\nu = \nu_x E(I+1, J, K) + \nu_y E(I, J+1, K) + \nu_z E(I, J, K+1),$$

where with $\hat{t}$ denoting the edge tangent vector, $\hat{\nu}$ is the edge normal vector $\hat{t}\otimes\hat{n}$, and $E(I,J, K)$ is defined by

$$E(I, J, K) = \int_{T_{ij}} x^{I-1}y^{J-1}z^{K-1}\,dl. \tag{25}$$

Using simple one-dimensional integration formulas $E(I, J, K)$ may be evaluated recursively by the formula

$$E(I, J, K) = \frac{1}{I+J+K-2}\,[E_\zeta + D_t] \tag{26}$$

$$E_\zeta = (I-1)\,\zeta_x E(I-1, J, K) + (J-1)\,\zeta_y E(I, J-1, K) + (K-1)\,\zeta_z E(I, J, K-1)$$

$$D_t = t_x D(I+1, J, K) + t_y D(I, J+1, K) + t_z D(I, J, K+1),$$

where $\zeta = (\hat{t}\otimes\mathbf{R})\otimes\hat{t}$ (constant along T_{ij}), and $D(I, J, K) = x^{I-1}y^{J-1}z^{K-1}|_1^2$, where 1 and 2 represent the initial and final points of T_{ij}. Thus, the original integrals (19) defined over a complicated volume can be systematically reduced to point evaluations at vertices of the bounding surface. The surface moments arising out of the surface integrals in Eq. (7) can be computed starting with Eq. (23). Note that the location of the centroid in each D region can be computed from the zero and first-order moments.

There remains, of course, the problem of identifying D regions and their bounding surfaces in a given boundary box. This is done in three stages. First, for each panel, we construct a list of grid boxes containing any part of the panel. Because the grid is rectangular and hierarchical in nature it is relatively easy to isolate the subset of boxes which are located within a neighborhood of a given panel. Moreover, because the boxes are rectangular and the panels are divided into flat triangles it is straightforward to determine if boxes in a neighborhood, in fact, contain any part of the given panel. This list is then inverted to find all the panels contained in a given boundary box. In the second stage we construct a list of equivalence classes of panel sides for each boundary box. A panel side is either the upper or lower surface of a panel. An equivalence class consists of all panel sides which are connected to each other through panel edges. A panel side is connected to another panel side if the two panels share a common edge that is partially or wholly contained within the given boundary box and if there is no intervening panel also connected to that edge. In the third stage we identify separate connected regions of the boundary box. This is done by choosing points on different panel side equivalence classes and then joining them with straight lines. The set of panels cutting these straight lines is examined and the panel side equivalence classes of panels responsible for successive cuts are identified as members of a new equivalence class of panel sides which bound a connected region. Polygonal subsets of a face of the boundary box are included in such an equivalence class whenever a panel side is discovered to intersect the face. This algorithm determines connected regions within a boundary box. However, it is also necessary to determine which such regions are connected to regions in adjacent boxes. This is because of the necessity of maintaining continuity of elemental basis functions across box faces and edges. For this purpose we keep track of which panels in each boundary box region intersect box faces. These intersections are compared with those in an adjacent box and connections between regions are established. The Ψ parameters at common nodes of connected regions are then identified.

3.5. *Grid Interfaces*

In the finite element method, conservation of mass results if the element basis functions are continuous. This property can be retained in the presence of grid refinement by introducing the notion of pseudo-unknown. A pseudo-unknown is defined as an unknown located at a node that is on the boundary of some element but is not located at a corner of this element. This can occur only at a coarse-to-fine grid interface. In the two-dimensional case pictured in Fig. 4, Φ_1 is a pseudo-unknown whose parents are Φ_2 and Φ_3. In the situation pictured in Fig. 3, Ψ_{ps} is a pseudo-unknown whose parents are Φ_p and Ψ_p. In order to maintain continuity of the element basis functions across element boundaries, Φ_1 in Fig. 4 must be $\frac{1}{2}(\Phi_2 + \Phi_3)$, i.e., the average of its parents. In three dimensions, pseudo-unknowns can occur at the midpoints of element edges or the centers of element faces. For a

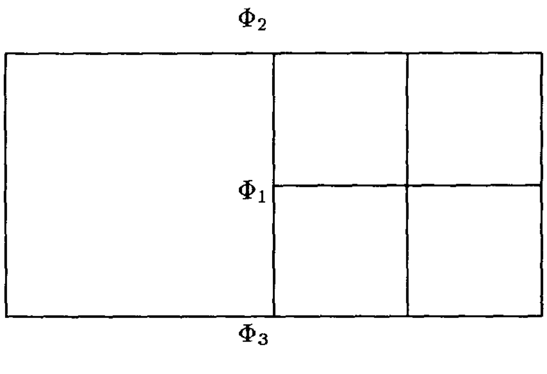


FIG. 4. In two dimensions, Φ_2 and Φ_3 are parents of Φ_1.

pseudo-unknown Φ_1 at the center of an element face with four parents Φ_2, Φ_3, Φ_4, and Φ_5,

$$\Phi_1 = \tfrac{1}{4}(\Phi_2 + \Phi_3 + \Phi_4 + \Phi_5). \tag{27}$$

Thus, pseudo-unknowns are not true degrees of freedom. They could be eliminated at the beginning from the element stiffness matrices by using Eq. (27). This would result in loss of uniformity in these matrices, many special cases, and loss of vectorization. Instead these unknowns are treated as degrees of freedom when the element stiffness matrices are generated. In the process of evaluating the discrete operator F, pseudo-unknowns are first assigned values by averaging their parent unknowns. Residuals of the governing equations are produced at these unknowns but are then distributed to the residuals for their parents. This technique has the advantage that every element stiffness matrix produces contributions only to the eight corner unknowns of its box element, thereby simplifying the generation of the stiffness matrices and enhancing vectorization. Vectorizing over large blocks of similar elements can be done using an outer loop over the eight corner unknowns and an inner loop over the elements in the block. This process of distributing residuals to parents is justified by Eq. (27) and a straightforward application of the chain rule

$$\frac{dJ}{d\Phi_2} = \frac{\partial J}{\partial \Phi_2} + \frac{\partial J}{\partial \Phi_1}\frac{\partial \Phi_1}{\partial \Phi_2} = \frac{\partial J}{\partial \Phi_2} + \frac{1}{4}\frac{\partial J}{\partial \Phi_1}. \tag{28}$$

Thus, the residual of the discrete version of Eq. (18) calculated at Φ_1 should be equally distributed to the residuals for the four parent unknowns.

3.6. *Modifications to the Bateman Principle*

The introduction of the last surface integral in the variational principle (7) enforces a Dirichlet condition on $\partial\Omega_3$. Equation (7) can then be used to calculate the element stiffness matrices in a finite element formulation. It turns out that the resultant discrete problem is somewhat unstable. In certain instances one can show that the boundary Ψ unknowns actually satisfy a discrete Helmholtz equation and an oscillatory solution is, in fact, obtained. This phenomenon is probably related to the fact that J is no longer maximized in subsonic flow with Dirichlet data. This suggests a remedy which we have implemented and has been found to be very reliable numerically. The last integral in (7) is replaced by

$$\int_{\partial\Omega_3} \left[\rho \frac{\partial \Phi}{\partial n}(\Phi - g_3) - \frac{\rho}{2\,\Delta l}(\Phi - g_3)^2 \right] dS, \tag{29}$$

where Δl is the minimum diameter of the box containing the basis function. A similar term may be added to the second integral.

All surfaces are represented by piecewise flat panels. Resultant discontinuities in slope from panel to panel will be reflected in the solution as the grid is refined. In most cases, this effect is spurious, since the surface slope discontinuities are artifacts of the panel description of the surface. To eliminate this problem, we simulate a curved surface by adding to the variation of Eq. (7) a surface integral

$$\delta J = \delta J + \int_{\partial \Omega_1} (\rho \, \nabla \Phi \cdot \hat{n} - \hat{n}^*) \, \delta \Phi \, dS, \tag{30}$$

where $\hat{n}^*$ is a polynomial interpolation of $\hat{n}$ and $\partial \Phi$ denotes the variation of Φ. The endpoints for the polynomial interpolation of the normal are user controlled. This allows discontinuities in slope where they are actually present in the underlying geometry.

3.7. *Dissipation*

In the full potential case, standard first-order upwinding of the density is used to produce the artificial viscosity required when supersonic flow is present [7, 34]. Such an upwinding is given by replacing ρ in the full potential equation with

$$\tilde{\rho} = \rho - \mu \hat{V} \cdot \nabla_{-} \rho, \tag{31}$$

where $\hat{V}$ is the normalized local velocity and $\nabla_{-} \rho$ is an upwind undivided difference. In Eq. (31), μ is the switching function given by

$$\mu = \max(0, 1 - M_c^2 / M^2), \tag{32}$$

where M is the local Mach number and M_c is the cutoff Mach number assigned the value $M_c^2 = 0.95$ chosen to introduce dissipation just below Mach 1.0. We upwind on a face basis with a stencil that is precomputed. Every element has six faces. For each face a density is chosen for upwinding in the operator generation phase of the algorithm. For a uniform grid with no boundaries, each box has a single box adjacent to it across each of its six faces. In case of grid refinement, there are two other cases. If the adjacent box is refined, the density used for upwinding is obtained by averaging the densities for the four adjacent refined elements. If the adjacent box is coarser, then three densities are averaged. In two dimensions, the possibilities for upwinding to the left across an edge are illustrated in Fig. 5.

We define the upwinded density $\tilde{\rho}$ by

$$\tilde{\rho} = \rho + \mu \sum_{i=1}^{6} \max(-\hat{V} \cdot \mathbf{n}_i, 0) \, s(i) \sum_j C_{i,j} (\rho_{i,j} - \rho), \tag{33}$$

where i runs over the six faces of the given box, j runs over the densities averaged to obtain the density upwinded to, $C_{i,j}$ is the coefficient for each of the four densities contributing to the density upwinded to, $\hat{V}$ is the normalized velocity at the centroid of the given element, $\mathbf{n}_i$ is the outward pointing normal to face i of the element, and $s(i)$ is a blending function to make the upwinding differentiable. This upwinding is first-order accurate, introducing an error comparable to replacing the

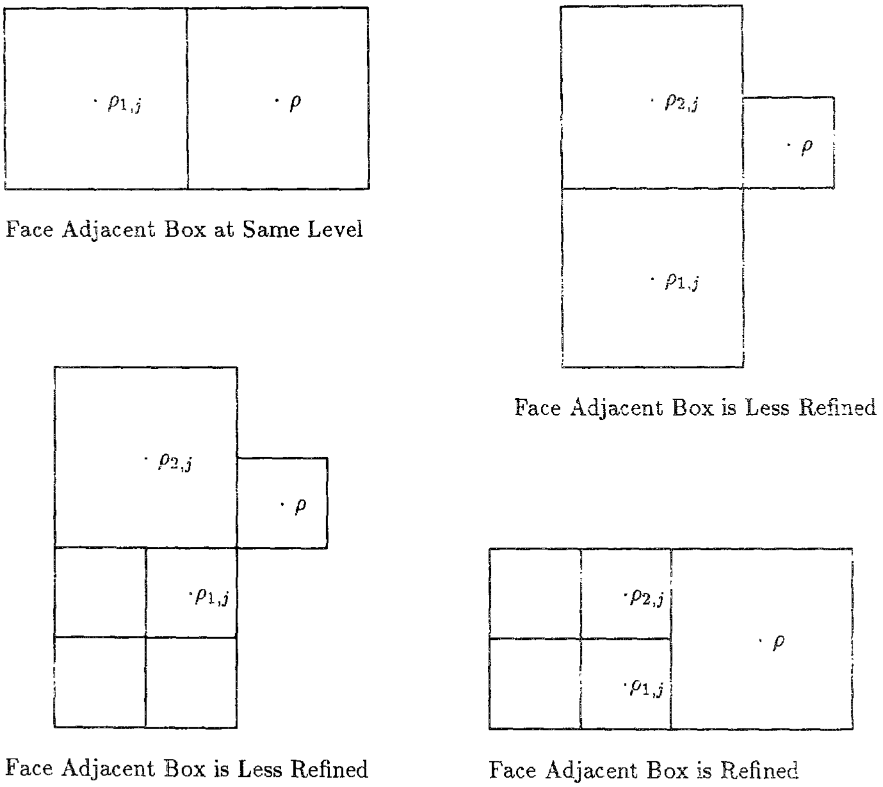

$\rho_{1,j}$
ρ
Face Adjacent Box at Same Level
$\rho_{2,j}$
ρ
$\rho_{1,j}$
Face Adjacent Box is Less Refined
$\rho_{2,j}$
ρ
$\rho_{1,j}$
Face Adjacent Box is Less Refined
$\rho_{2,j}$
ρ
$\rho_{1,j}$
Face Adjacent Box is Refined

FIG. 5. Upwinding stencils in two dimensions for negative x edge.

density ρ with a piecewise constant approximation in each element. In the case of D regions, special operators must be constructed based on local information about box adjacency. We will not discuss this aspect of the method in detail except to say that all the information needed is extracted from the octtree and D region information in a preprocessing step.

3.8. *Accuracy of the Discretization*

At first glance, one might think that this discretization would suffer from approximation problems, since high aspect ratio D regions are often present. By keeping the trial functions parameterized by values at the corners of similar boxes, uniformity of the basis [33] is guaranteed. Thus, in the limit, standard approximation theory and finite element error estimates hold. Another issue for the discretization described here is conditioning. We have no theoretical guarantee of good conditioning as the mesh is refined. However, in practice, conditioning problems have been encountered only when very small D regions are present (six or seven orders of magnitude smaller than neighboring grid cells).

The asymptotic convergence of the method has been verified with uniform grids for the case of a sphere in incompressible flow where an analytic solution is available [20]. Sections 7 and 9 contain computational examples that demonstrate the method's accuracy for locally refined grids.

4. Octtree Data Structure

We have developed a compact data structure which contains essentially all the information regarding the refined grid. Asymptotically this data structure requires storage equal to $\frac{10}{7}$ the number of unrefined boxes for the box information. However, the factor is closer to 2 in practice because node information must also be stored. The data structure allows efficient extraction of a variety of information, such as the location of nodes and element centroids, box size, box level, node indices, box adjacency, and identity of boundary boxes.

We use a modification of an octtree-type data structure described by Samet [32]. At the beginning of the octtree data structure certain overhead information that describes the size of the grid is stored. This is followed by two blocks of data, each as long as the size of the global grid, describing its refinement. Then we follow the branch data elements of the tree. These describe the hierarchical refinement and form the bulk of the octtree.

A typical branch data element in an octtree is illustrated in Fig. 6. The first word in a branch data element points to the father box, the next eight words point to the refinement branches of the sons if any, and the last word contains an accumulation index specifying the number of nodes encountered up to that point in the octtree.

A null pointer (value zero) in the son entry in any branch data element represents an unrefined box. Any unrefined box cut by a boundary is identified by a negative number (equal to its index in a list of such boxes) placed in the son entry in a branch data element. This is a convenient and compact way of accounting for the presence of the boundary.

We have extended the octtree data structure further to accommodate nodal information. Nodes are indexed by assigning the index of a box to the node at its

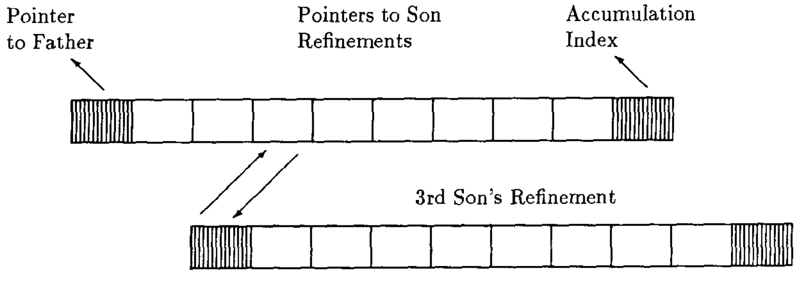


Fig. 6. Octtree data structure element.

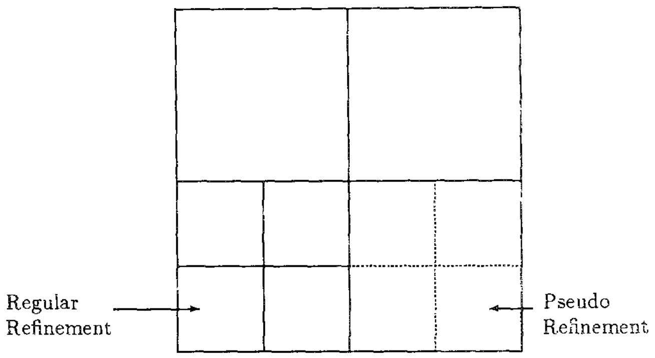


FIG. 7. Some details of the hierarchical refinement.

lower-left-near corner. In order to account for all nodes at refinement interfaces we perform pseudo refinement (see Fig. 7). This allows us to keep track of the nodes as well as the boxes using the same octtree with only a modest increase in storage. Boxes added by the pseudo refinement are used only to identify nodes and are not finite elements.

Even though the octtree data structure described above is able to reflect an arbitrary collection of hierarchically refined grid cells, it is convenient to restrict the refinement pattern. We require that no two face or edge neighbors in a "legal" refined grid differ by more than one level (see Fig. 8). This rule prevents pathologically large stencils under certain circumstances, but allows refinement down to an arbitrary level within one adjacent coarse grid box.

The location of a node or a centroid of an unrefined box can be calculated by climbing up to the ancestor in the global grid and then using the global box information. Adjacency or box-to-box connectivity information can be obtained by starting at a box and climbing up the octtree to the root of the branch that includes that box and climbing down a complementary path to the neighboring box.

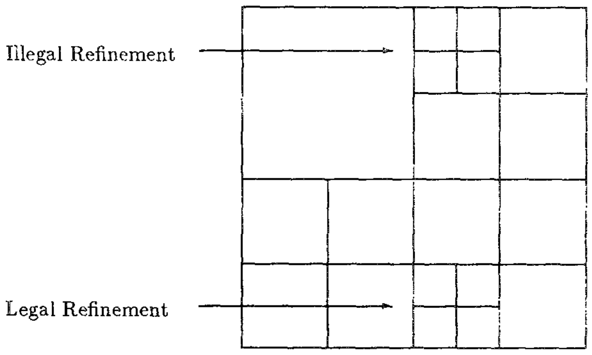


FIG. 8. Legal and illegal refinements.

5. Linear Solution Algorithm

The solution technique we use was designed for nonlinear problems. However, it is useful to first describe the algorithm in the special case of a linear boundary value problem, e.g., the Prandtl–Glauert equation in aerodynamics. We will consider the generic linear potential equation

$$\nabla \cdot (\rho \nabla \Phi) = f \tag{34}$$

with $\rho = \rho(x, y, z)$ assumed to be given and strictly positive. Boundary conditions are those described earlier in Section 2 for the full potential equation. The finite element operator described in Section 2 will be denoted by L. It is defined over the whole grid except on the boundary of the global grid and is evaluated in the standard way by multiplying each element stiffness matrix by the current vector of unknown values. T is the constant coefficient discrete Poisson operator on the global grid. In order to enforce the far field condition (3), source unknowns Q are introduced on the global grid and are defined by $T\Phi = Q$ as discussed in Section 3. Except in the vicinity of wakes, we assume that $L = T$ on the boundary and exterior of the computational domain so that nonzero sources Q are confined to the interior of the global grid. At global grid points, we replace the unknown Φ by Q. Since Q is known to be zero on the boundary of the global grid, the residual does not need to be computed there. Since L is defined in terms of Φ, it is necessary to be able to compute Φ when Q is given, i.e., to compute $T^{-1}Q$. The result of applying T^{-1} in the present implementation of the method always satisfies the discrete version of the far field condition (3). This is true because T^{-1} is evaluated by convolution with a discrete Green's function G which itself satisfies this discrete far field condition [20]. Thus, on the global grid, $\Phi = T^{-1}Q = G * Q$. In the following, we will denote the extrapolated values in boundary boxes by Ψ and all other variables on the refined grid by Φ. The doublet parameters on wakes are denoted by μ. We thus want to solve the linear system of equations

$$L \begin{pmatrix} T^{-1}Q \\ \Phi \\ \Psi \\ \mu \end{pmatrix} = f. \tag{35}$$

An iterative procedure is chosen for this purpose. Since the system (depending on boundary conditions) is non-symmetric and non-definite we chose the GMRES method of Saad and Schultz [24] as the basic iterative solver. An iterative method was required because the problem is generally too large to be solved by any direct method and because of the nature of the far field condition. The discrete far field condition is not sparse and would be very complicated to discretize in matrix form. The T^{-1} operator already acts as an effective right preconditioner for the global chosen to approximate the problem near the internal boundaries. For this purpose, we define the reduced set to consist of all unknowns that are located at corners of boundary boxes, refined boxes, or boxes with total pressure or temperature different from freestream values. The doublet parameters μ are also included but stagnation unknowns are not. Closure unknowns are those unknowns outside the reduced set but in the stiffness matrix stencil of some unknown in the reduced set. The preconditioner N is taken to be the global stiffness matrix restricted to the reduced set. Special boundary conditions are applied at closure unknowns to close the system. We have found that it is quite feasible to do a direct sparse incomplete factorization of N. This works for the following reasons. First, the reduced set is often significantly smaller than the total number of degrees of freedom in the problem. Second, a drop tolerance can be introduced into the sparse elimination process allowing small elements in the decomposition to be dropped as they are generated. This has a cascading effect and dramatically reduces fill [26]. Third, a grid-based nested disssection ordering can be generated for this reduced set. This ordering reduces fill during elimination and reduces the total amount of work. In the full potential case, the drop tolerance is the most effective strategy. Large regions of refined grid restrict the effectiveness of the first and third factors. Figure 9 shows the reduced set and a possible first dissector for a grid for a sphere case.

The boundary condition at closure unknowns is an approximation to the far field condition for the original problem, i.e., $\phi = 0$. Note that there is some overlap between the Q unknowns on the global grid preconditioned by T^{-1} and those in the reduced set preconditioned by N^{-1}. For these unknown sources Q at global grid points in the reduced set, an additional preconditioner T must be applied on the left to make the equation dimensionally correct.

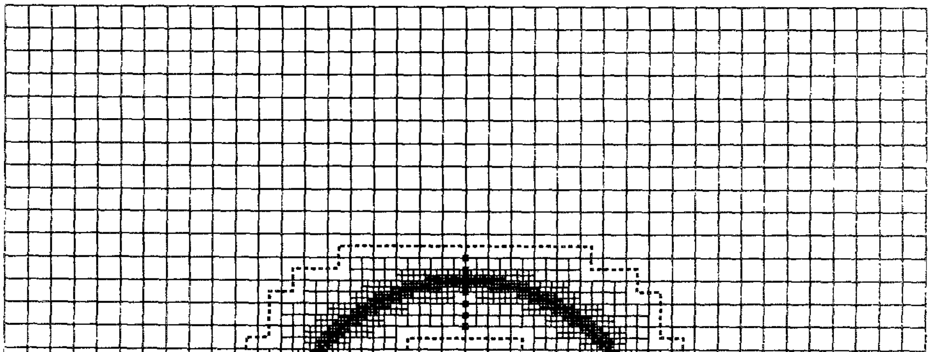


FIG. 9. Reduced set and possible first dissector.

There are then a total of five classes of unknowns in a given transonic flow problem. They are: $Q^{(1)}$, the source unknowns at global grid points which are not in the reduced set and not in stagnation regions; $Q^{(2)}$, the source unknowns at global grid points in the reduced set or in stagnation regions; Φ, the values of the velocity potential at points on locally refined grids; Ψ, the values of velocity potential in the boundary basis functions; and finally, μ, the doublet strengths at leading edges of wake networks.

The preconditioned equation at reduced set unknowns can then be written as

$$TN^{-1}(f - LT^{-1}\mathscr{X}) = 0, \tag{36}$$

where

$$\mathscr{X} = \begin{pmatrix} Q^{(1)} \\ Q^{(2)} \\ \Phi \\ \Psi \\ \mu \end{pmatrix}. \tag{37}$$

The operators T and N are defined as

$$T = \begin{pmatrix} T_{(1)(1)} & T_{(1)(2)} & 0 & 0 & 0 \\ T_{(2)(1)} & T_{(2)(2)} & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{pmatrix} \tag{38}$$

$$N = \begin{pmatrix} I & 0 & 0 & 0 & 0 \\ 0 & N_{(2)(2)} & N_{(2)(3)} & N_{(2)(4)} & N_{(2)(5)} \\ 0 & N_{(3)(2)} & N_{(3)(3)} & N_{(3)(4)} & N_{(3)(5)} \\ 0 & N_{(4)(2)} & N_{(4)(3)} & N_{(4)(4)} & N_{(4)(5)} \\ 0 & N_{(5)(2)} & N_{(5)(3)} & N_{(5)(4)} & N_{(5)(5)} \end{pmatrix}. \tag{39}$$

To achieve invariance with respect to units, some scaling of the source unknowns Q relative to the field unknowns is done. Note that from dimensional analysis alone, a source differs from a field quantity by a factor of the inverse length squared. Source scale factors are defined so that the GMRES convergence history is independent of the physical units used to define the problem.

The calculation of the preconditioned residual R (the function evaluation subroutine for GMRES) can be described as follows. For unknowns $Q^{(1)}$ on the global grid but not in the interior of the reduced set,

$$R(Q^{(1)}) = f - L\begin{pmatrix} T^{-1}\begin{pmatrix} Q^{(1)} \\ Q^{(2)} \end{pmatrix} \\ \Phi \\ \Psi \\ \mu \end{pmatrix}. \tag{40}$$

For unknowns $Q^{(2)}$ on the global grid and in the interior of the reduced set or located at global grid points in stagnation regions,

$$R(Q^{(2)}) = TN^{-1}\left(f - L\begin{pmatrix} T^{-1}\begin{pmatrix} Q^{(1)} \\ Q^{(2)} \end{pmatrix} \\ \Phi \\ \Psi \\ \mu \end{pmatrix}\right). \tag{41}$$

In (41), special account must be taken of unknowns Q located in stagnation regions, such as the interior of wings and fuselages. For these unknowns, it is important to realize that N^{-1} is just the identity, $f = 0$, and $L\Phi = \phi$. Thus T is applied to the global grid unknowns in the reduced set, closure point unknowns, and stagnation unknowns. But the input for T at stagnation unknowns comes from a different process than that used for the other two classes of Q unknowns. Another special class of Q unknowns in (41) are those at closure points. N^{-1} does apply to the residual at these points producing input values for T to give residual values for points in the reduced set. But for these closure unknowns, the residual equation is actually (40).

For unknowns Φ not located on the global grid and for all unknowns Ψ and μ,

$$\begin{pmatrix} R(\Phi) \\ R(\Psi) \\ R(\mu) \end{pmatrix} = N^{-1}L\begin{pmatrix} T^{-1}\begin{pmatrix} Q^{(1)} \\ Q^{(2)} \end{pmatrix} \\ \Phi \\ \Psi \\ \mu \end{pmatrix}. \tag{42}$$

For Φ unknowns located at points not in the global grid but in stagnation regions, Eq. (42) must be modified. The residual is given by $R(\Phi) = L\Phi = \phi$.

The convergence rate of this method depends on the drop tolerance used in the sparse solver. When the reduced set is essentially two-dimensional, nested dissection will be particularly effective and no drop tolerance is needed. In this case, the method typically converges in 10 to 20 iterations. (Here converged means that the residual has been reduced by 10 orders of magnitude from the initial residual.) This shows that the reduced set problem is in general an excellent preconditioner. Introducing a drop tolerance sufficient to reduce fill and work by an order of magnitude typically causes the number of iterations required for convergence to at most double.

6. Sparse Solver

In this section, the general purpose sparse solver that is used to factor the preconditioner N mentioned above is summarized. The sparse solver was designed for general usage to solve very large problems. It has a general input capability allowing contributions to matrix elements to be entered in any order. These contributions are sorted and combined to produce the final matrix. This feature is particularly convenient with finite elements, where element stiffness matrices can be generated in any order. The solver is out-of-core so that quite large problems can be solved on current computers. Gaussian elimination is performed by block rows, additional blocks being created as fill is generated. The blocks are stored on a random file. If the blocks reside on a CRAY SSD, for example, the block transfers can be made almost as fast as transfers from main memory. A drop tolerance can be used to drop small elements in the L and U factors as they are generated. Each element in the decomposition is compared to the magnitude of the diagonal entry of the current row and dropped when this ratio is smaller than the tolerance. For a great many full potential cases, we find that good preconditioning is obtained even if the total size of the incomplete L and U factors is only twice the number of nonzeros in the original matrix [26]. In all cases we have considered, physically based nested dissection ordering is feasible. This ordering is often quite valuable in reducing the cost of the decomposition. We will describe in detail only two features of this solver in the context of its use as a preconditioner in a finite element method. Other features are detailed in [26].

6.1. *Sorting*

Contributions to the global stiffness matrix are generated on an element by element basis, i.e., element stiffness matrices are input one by one. This order is unrelated to the ordering of the unknowns used for the decomposition. Thus, the contributions must be sorted and coalesced to produce the final global stiffness matrix. This process consists of four steps and is illustrated in the case of two blocks and a 3 by 3 matrix in Fig. 10. Each contribution is described by a numerical value (denoted by a letter) and by row and column indices. The four steps are as follows:

1. All contributions are collected in equal sized blocks and stored out of core.

2. Each block is returned to main memory and sorted by row index, resulting in an ordered sequence of row groups within each block. A row group is a group of contributions all having the same row index. A simple bucket sort algorithm

Contributions input in two blocks

a	b	c	d	e	f	g	h	i	j	k	l	m	
1,2	1,3	1,1	3,1	2,1	1,1	3,2	2,3	3,2	1,2	3,3	2,2	2,3	

Sort each block into row groups

a	b	c	f	d	e	g	j	l	h	m	i	k	
1,2	1,3	1,1	1,1	3,1	3,3	3,2	1,2	2,2	2,3	2,3	3,2	3,3	

Merge two blocks into a chain of blocks

a	b	c	f	j	l	h	m	d	e	g	i	k	
1,2	1,3	1,1	1,1	1,2	2,2	2,3	2,3	3,1	3,3	3,2	3,2	3,3	

Coalesce all contributions to the same matrix element

c+f	a+j	b	l	h+m			d	g+i	k+e				
1,1	1,2	1,3	2,2	2,3			3,1	3,2	3,3				

FIG. 10. Sorting and merging procedure.

3. These sorted blocks are then merged into ordered chains of blocks. An ordered chain is a chain of blocks such that all contributions in a given block have row indices less than or equal to those in all subsequent contributions in that block and the remaining blocks of the chain. Merging two chains consists in interleaving their row groups so that the resulting chain is also sorted by row index. At a given stage, the two shortest chains are always merged. Ultimately, the result is a single chain of blocks. Because the contributions are not sorted by column index at this stage, all movement of elements can be done by row group. This allows vectorization of the merge algorithm.

4. All contributions to the same matrix element are then coalesced, i.e., within each row group, contributions with common column indices are added to form a single contribution. Coalescing can be done without first sorting each row group by column index.

We note here that most elements of the global stiffness matrix have contributions from eight element stiffness matrices in subsonic regions and as many as 64 in supersonic regions. Thus, the number of contributions may be up to 125 times greater than the number of elements in the assembled global stiffness matrix. In order to minimize storage requirements, the above four-step process is performed repeatedly. (It can be performed at any point in the generation of contributions to the stiffness matrix.) When this is done, chains which have already been formed are not merged until new chains of equal size exist. Row groups are sorted by column index using a bucket sort only after completion of contribution input and coalescing.

6.2. *Physically Based Nested Dissection Ordering*

For large problems, a sparse matrix preconditioner is practical only if the decomposition is also sparse. This is true not only because of storage limitations, but also because of the CPU time required for forward and back substitution. One key to maintaining sparsity is a good permutation ordering for the rows and columns of the matrix. For sparse matrices resulting from standard discretizations of elliptic partial differential equations on uniform rectangular grids, nested dissection has been shown to be asymptotically optimal [36].

In the current implementation, local grid refinement leads naturally to an ordering of the unknowns that is not very favorable for sparse factorization. This defect has been remedied by implementing a physically based version of nested dissection suitable for grids with local refinements. One advantage of this method is that it does not require an examination of the graph of the matrix. The algorithm acts recursively on subsets of nodes (grid points). The first such subset is the set of all nodes in the reduced set. For a set of nodes $\mathcal{N}$ the algorithm finds a set of nodes $\mathcal{N}_d$ called a dissector. We write $\mathcal{N} = \mathcal{N}_d \cup \mathcal{N}_l \cup \mathcal{N}_r$, where $\mathcal{N}_l$ consists of nodes on one side of $\mathcal{N}_d$ and $\mathcal{N}_r$ those on the other. The dissector has the property that an unknown at any node on one side has a stencil that does not include unknowns located at nodes on the other side. The permutation is produced by ordering the nodes in the dissector last. Figure 11 shows the block structure of this matrix. The

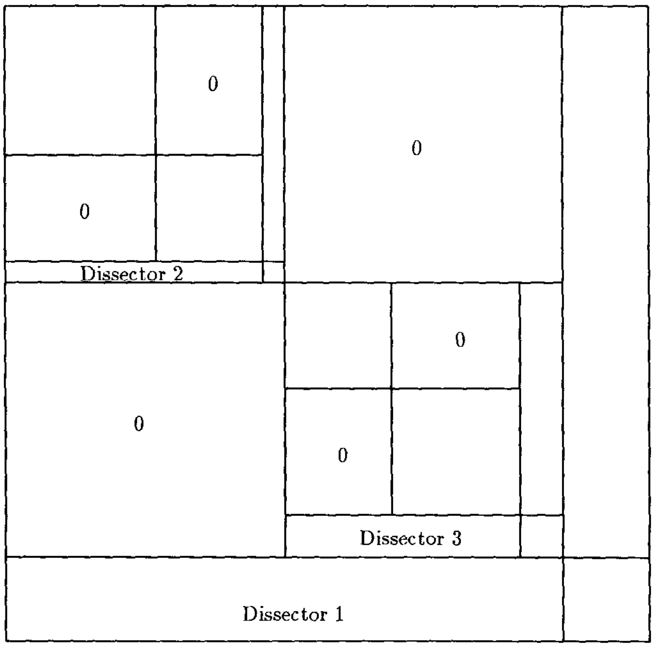


FIG. 11. Block structure of a sparse matrix ordered with nested dissection.

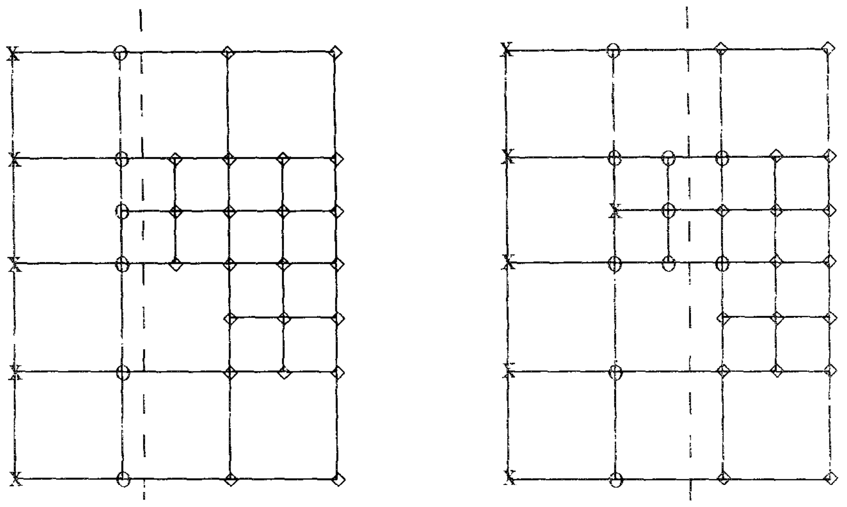


FIG. 12. Examples of cutting planes and nodes in resulting dissector: – – –, cutting plane; ×, node in $\mathcal{N}_l$; ○, node in dissector $\mathcal{N}_d$; ◇, node in $\mathcal{N}_r$.

blocks of zeros remain intact, preserving sparsity during the decomposition. For a structured grid a plane of points forms a suitable dissector for a 27-point stencil.

Dissectors are generated by first taking a cutting plane perpendicular to a coordinate axis and finding the set $\mathcal{B}$ of all boxes intersecting this plane by interrogating the octtree data structure. Taking the case when the cutting plane is perpendicular to the x axis, the dissector is taken to consist of all nodes on the left hand (negative x) face of boxes in $\mathcal{B}$ that are also in $\mathcal{N}$. This will provide a dissector except near pseudo-nodes where the stencil is altered. When a pseudo-node is in the dissector its parent nodes must also be included in the dissector. Figure 12 shows examples (in two dimensions) of cutting planes and corresponding sets of nodes in the dissectors.

There are two guiding principles that aid in producing an effective nested dissection ordering. The first is that the components $\mathcal{N}_l$ and $\mathcal{N}_r$ resulting from the dissection be of approximately equal size. The second is that the dissectors contain as few unknowns as possible. (The size of the dissectors can vary due to local refinement and the location of boundaries.) These principles can conflict and some compromise is necessary. The cutting plane is selected to have x coordinate equal to that of the node in $\mathcal{N}$ with median x coordinate. Dissectors perpendicular to each of the three coordinate axes are tested and the one yielding the smallest dissector is chosen. The process is repeated recursively on the newly formed components resulting from each dissection until all remaining components contain fewer than 50 nodes.

The above algorithm is modified when regions of supersonic flow are present in the full potential case because upwinding of the density enlarges the stencil (see

Section 3). If the cutting plane intersects a box which contains supersonic flow or is adjacent to such a box then all the nodes of the box are included in the dissector.

7. Computational Results for Linear Flow

We have implemented this method in a computer program called TRANAIR. This program was written for the CRAY X-MP computer but has also been run on a CRAY Y-MP and on a large central memory machine, namely, the CRAY 2. Detailed comparisons of TRANAIR results with those of a linear panel code [5] used extensively at Boeing have been made. Linear flow is modeled using the Prandtl–Glauert equation (15). In this section, we discuss solutions for three geometries: a sphere, the ONERA M6 wing, and the F16 fighter aircraft configuration.

The first case presented is a sphere with radius 0.8 in linear potential flow with $M_\infty = 0$. This is a nontrivial problem for a Cartesian grid method since the surface intersects the grid in many different ways depending on the location on the sphere. Four grids were used to test the accuracy of TRANAIR. In Fig. 13 the paneling used for the coarse and medium grids is shown. With the fine and uniform grids,

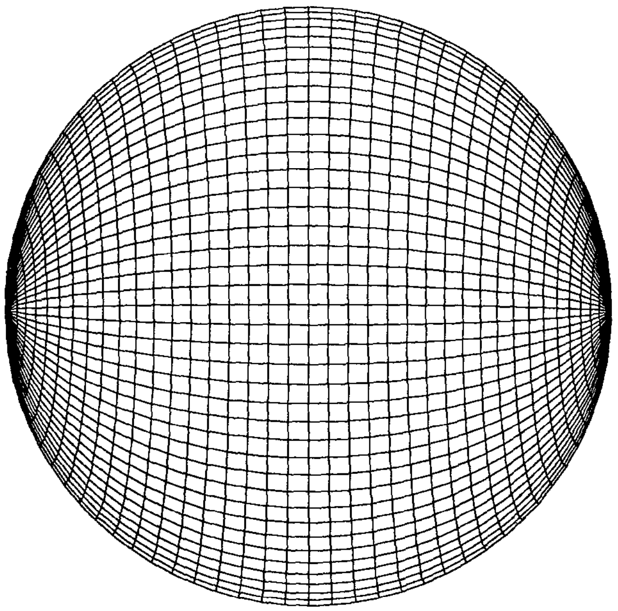


FIG. 13. Paneling used for sphere in linear flow, 1600 panels.

the paneling was doubled in each direction, resulting in 6400 panels. Planar cuts through the four grids are shown Fig. 14. The uniform grid had 123,680 elements. Its far field boundary was somewhat closer to the sphere. The coarse, medium, and fine grids had 10,356, 35,456, and 149,515 elements, respectively. These cases were run with one plane of symmetry. Only half of each cut is shown since each cut is symmetric about a second place of symmetry. Stagnation regions (those totally inside the sphere) are eliminated in a preprocessing step and are not included in the element totals.

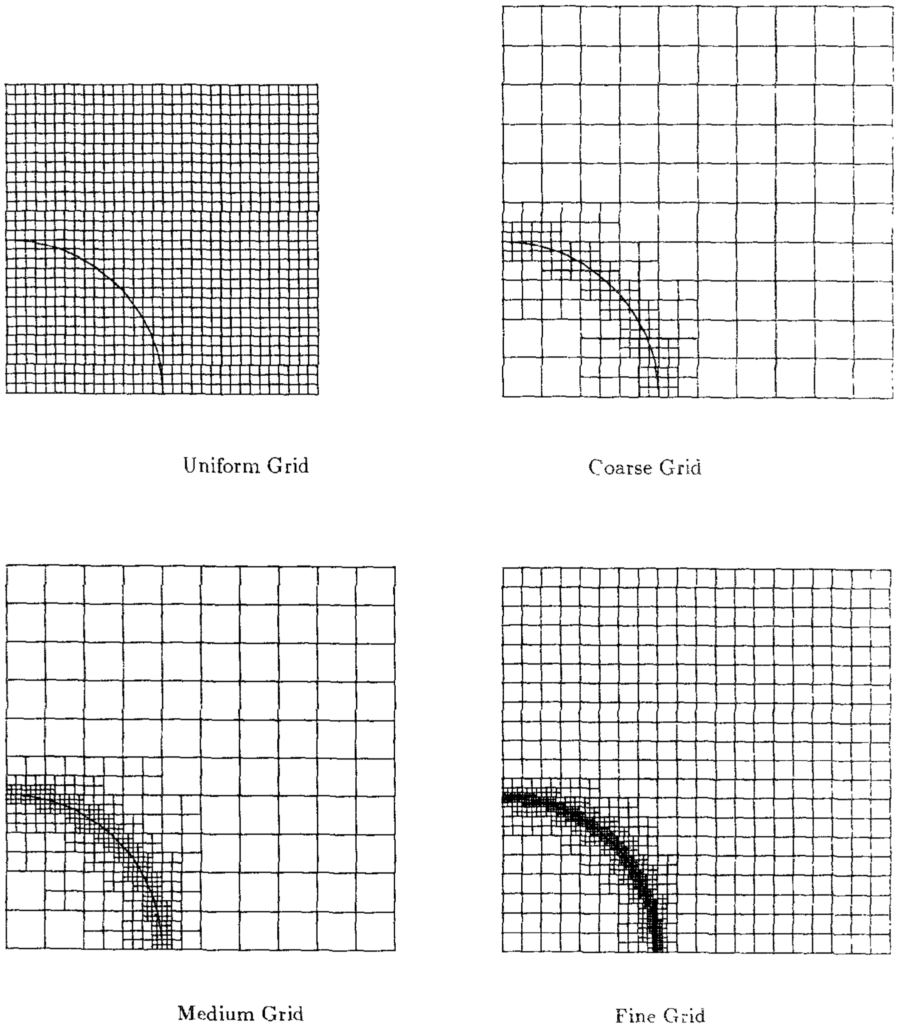
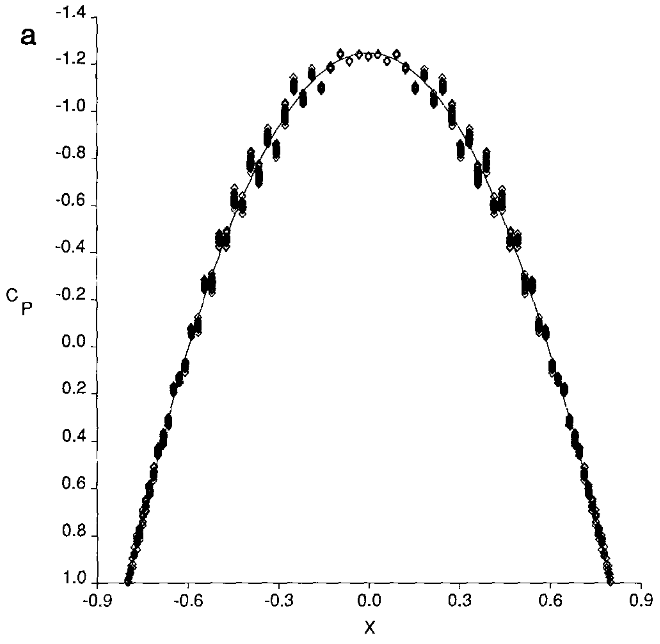


FIG. 14. Cuts through four grids for a sphere in linear flow. $M_\infty = 0$.



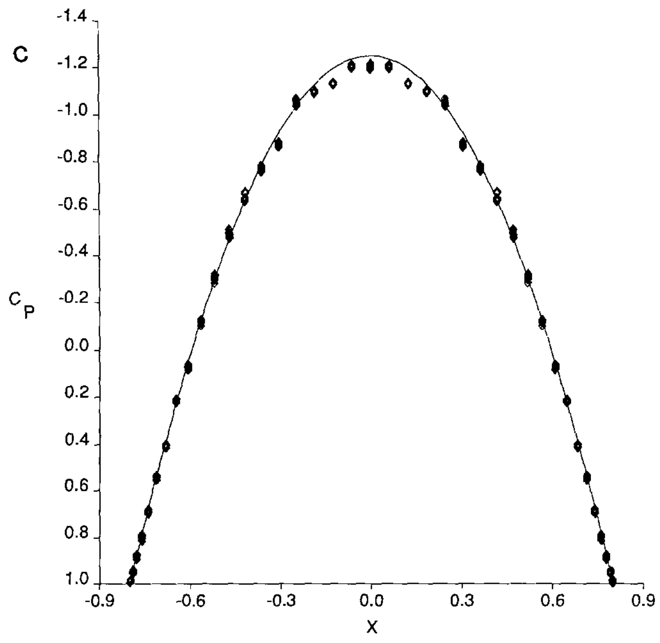
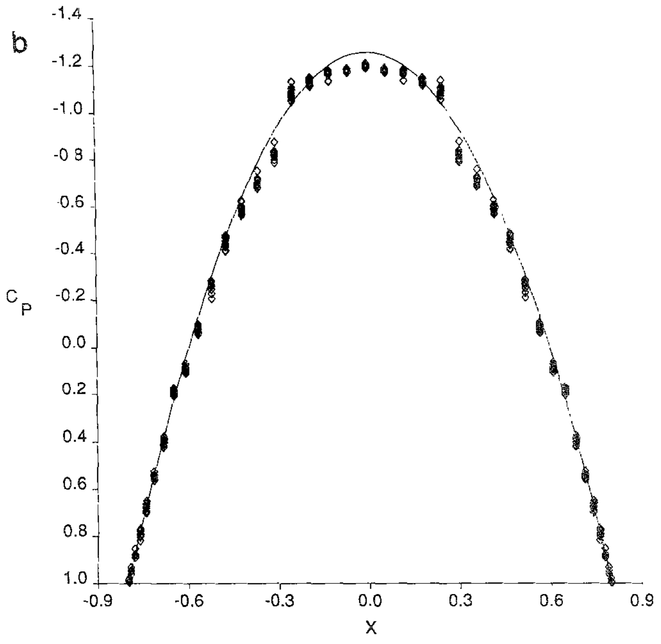


FIG. 15. Solutions on four grids for a sphere in linear flow, $M_\infty = 0$. (a) Uniform grid; (b) coarse grid; (c) medium grid; (d) fine grid; ———, analytic solution; $\diamond$, TRANAIR.



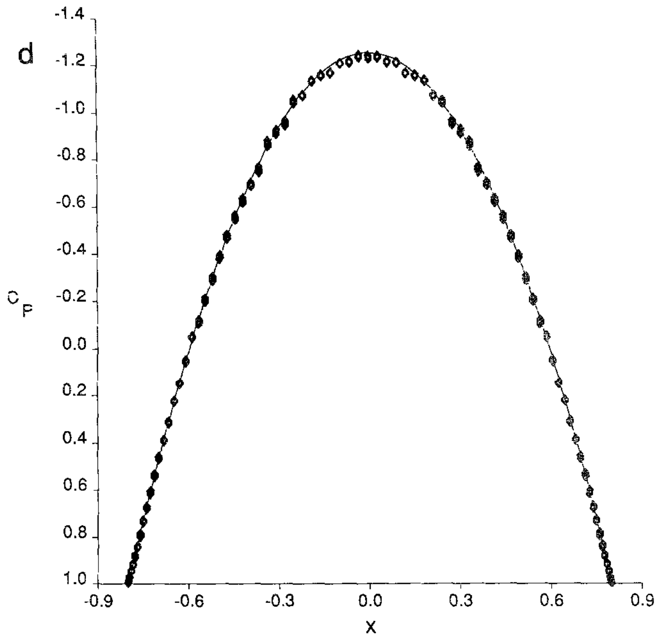
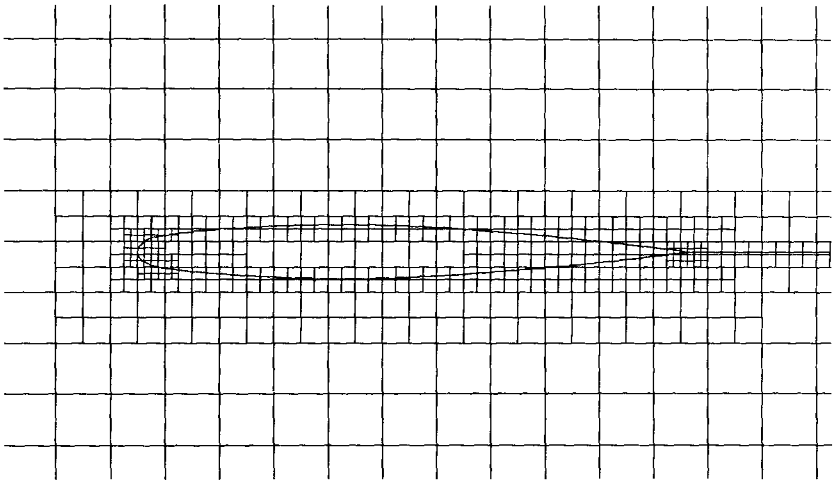


FIG. 15—*Continued*


Coarse Grid

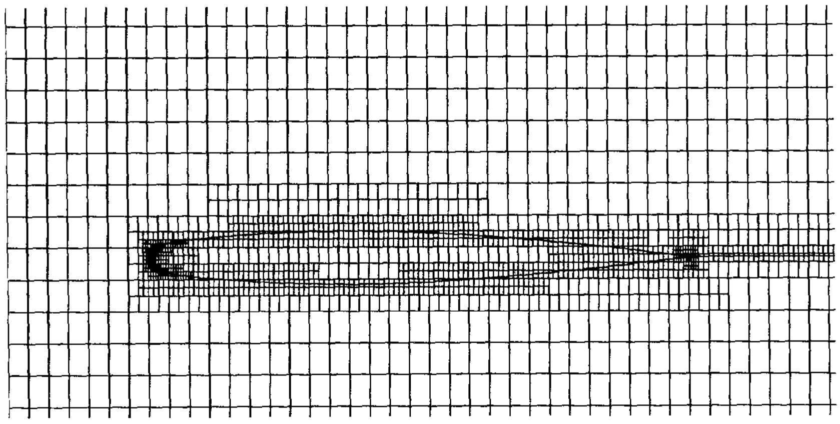

Fine Grid

FIG. 16. Cuts through two grids for the ONERA M6 wing in linear flow, $M_\infty = 0$, $\alpha = 3.06°$.

In aerodynamics, pressure is generally represented by its non-dimensional counterpart defined as

$$C_p = \frac{p - p_\infty}{\rho_\infty q_\infty^2/2}, \tag{43}$$

where p is local pressure, ρ_∞, p_∞, and q_∞ are the freestream density, local pressure, and velocity magnitude. In Fig. 15 the surface pressure for the sphere is plotted as a function of x. In all cases presented in this paper, the solution is displayed at panel corner points. Velocities and hence pressures are computed by simply evaluating the gradient of the basis functions at these points. If the panels are small compared to the local grid cell size, i.e., there are many panel corner points in a single grid cell, plotted C_p results will show a stairstep effect. This is due to the fact that the gradient of the basis function is almost constant in each cell. To eliminate this effect, standard finite element post processing algorithms, see, for example, [37, 38], are available to smooth the velocities. One such algorithm has been recently implemented but was not used for the results shown in this paper. The results generated with post processing are not significantly different in most cases. However, for the sphere cases, post processing improves the symmetry of the results

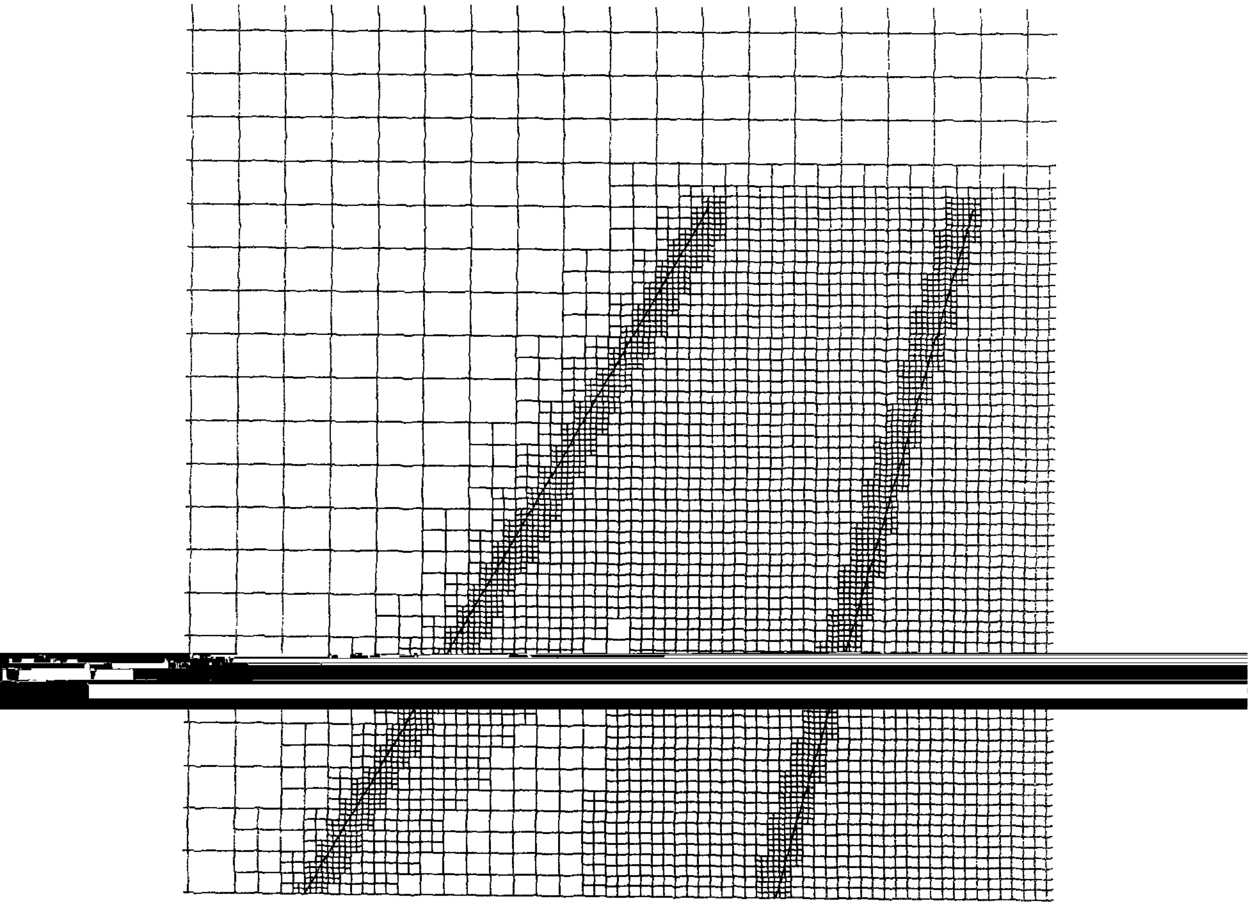


FIG. 17. Waterline cut through ONERA M6 coarse grid.

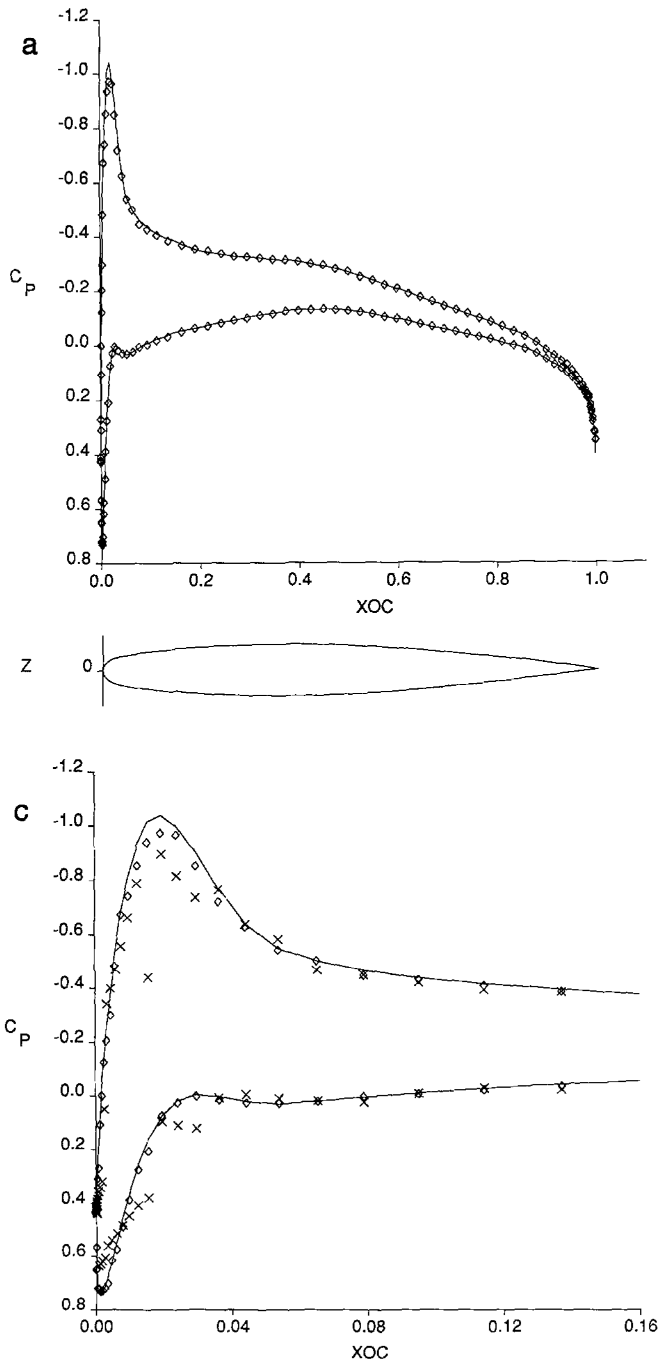
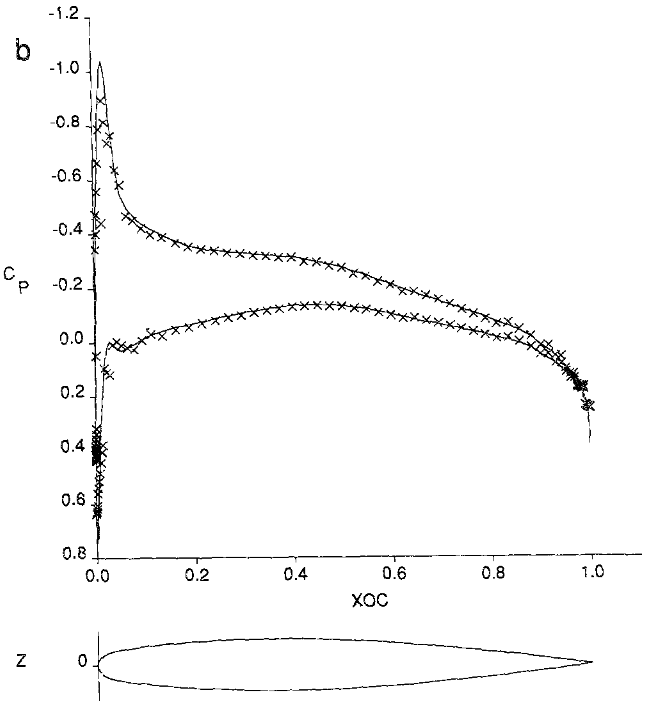

a
C_P
-1.2
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0
XOC
Z
0
c
0.00
0.04
0.08
0.12
0.16

FIGURE 18



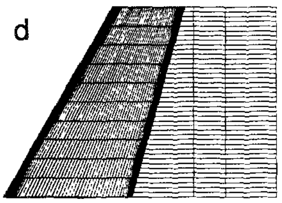
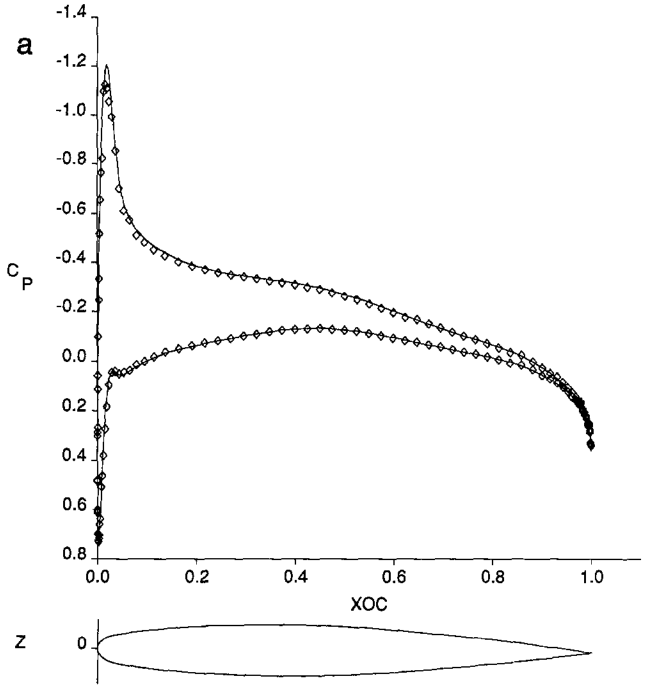


FIG. 18. Solutions for the ONEAR M6 wing in linear flow at 20% span, $M_\infty = 0$, $\alpha = 3.06°$; (a) fine grid; (b) coarse grid; (c) leading edge closeup; (d) paneling; ———, panel method; ◇, TRANAIR fine grid; ×, TRANAR coarse grid.

(a sensitive test of accuracy). This post processing can be summarized briefly as follows. Velocities at nodes are obtained by averaging basis function velocities from surrounding elements. Velocities at any point are then obtained by trilinear interpolation using the nodal values. This results in continuous velocities and hence continuous surface pressure distributions.

In Fig. 15, data at all circumferential stations (panel corner points) are plotted. For the 1600 panel case, there are 20 stations at each x value. The scatter of surface



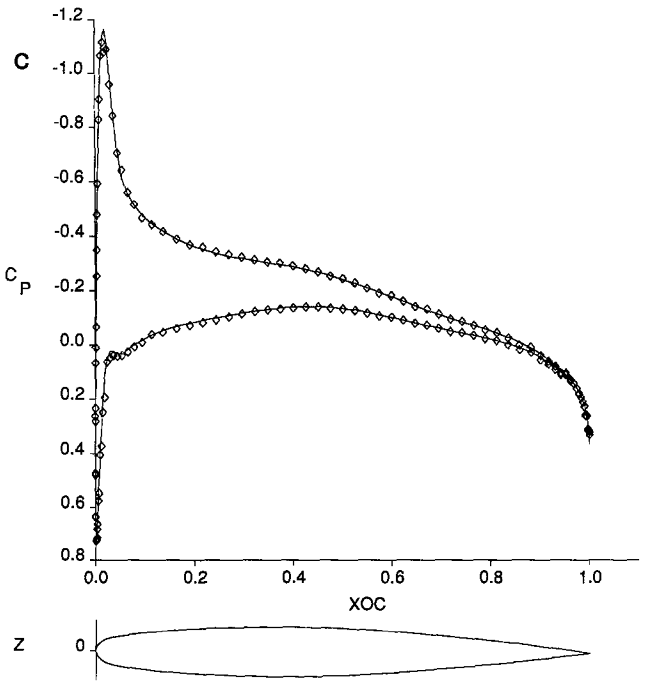
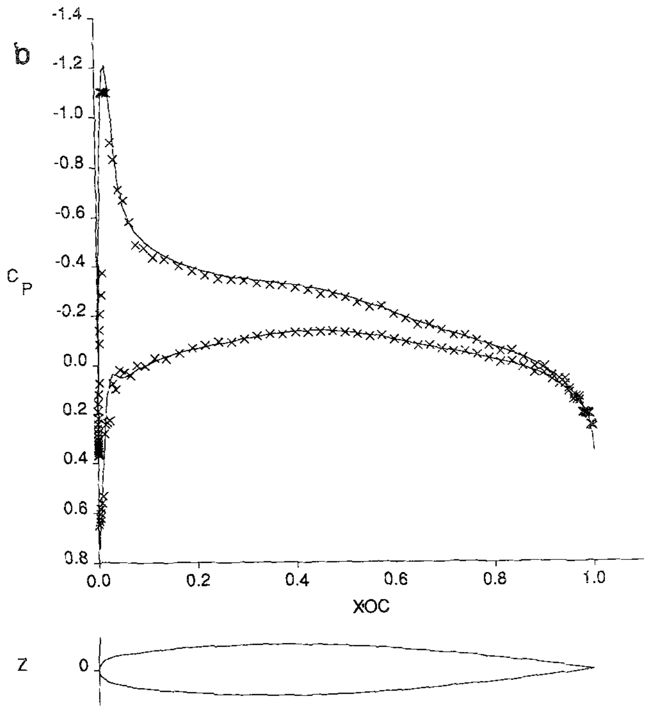


FIG. 19. Solutions for the ONERA M6 wing in linear flow at 60% span, $M_\infty = 0$, $\alpha = 3.06°$; (a) 60% span, fine grid; (b) 60% span, coarse grid; (c) 80% span, fine grid; (d) 80% span, coarse grid; ———, panel method; ◇, TRANAIR fine grid; ×, TRANAIR coarse grid.



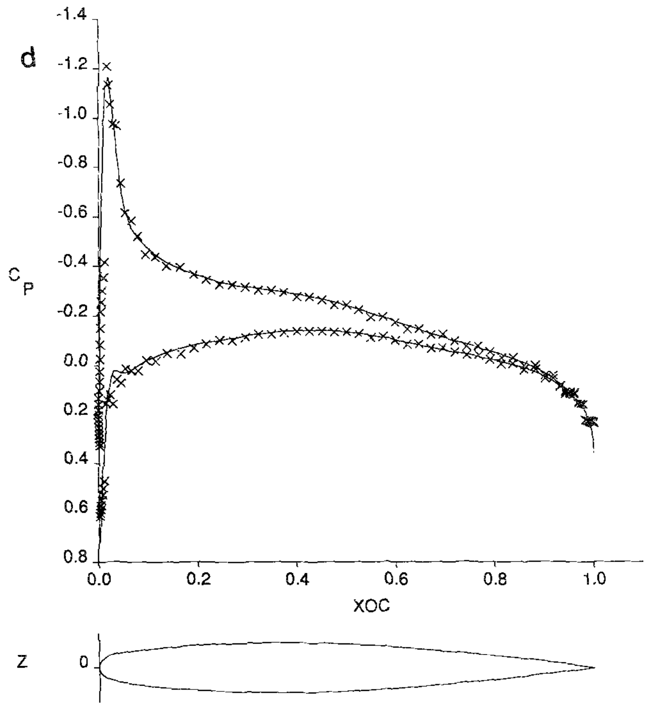


FIG. 19—*Continued*

pressure at a constant x coordinate is due to the Cartesian nature of the grid and gives a good measure of the overall accuracy of the TRANAIR solution. For this problem, an analytic solution is available. The solution accuracy improves significantly as the grid is refined. The expected quadratic convergence rate in potential as the grid is refined has been verified in this case using an earlier uniform grid version of TRANAIR [20].

A second case is an ONERA M6 wing with about 1800 panels. The flow conditions are $M_\infty = 0$ and angle of attack $\alpha = 3.06°$. The panels have a very high aspect ratio, being much longer spanwise than chordwise (pictured in Fig. 18). This paneling is adequate for a panel method because the solution also changes much more rapidly chordwise than spanwise. The number of panels needed to describe the variation of the surface unknowns (sources and doublets) in a panel method is relatively small, making this a favorable case for the panel method. TRANAIR was run on a coarse and a fine grid, the coarse grid having 35,188 elements and the fine grid having 249,305 elements. Vertical cuts through the two grids at the plane of symmetry are shown in Fig. 16. Figure 17 shows a waterline cut through the coarser of these two grids. The clustering of fine grid cells at the leading and trailing edges is necessary to resolve high velocity gradients even for linear flow about a simple wing. Figure 18 compares surface pressure at the 20% span station. The solid line is a solution obtained with the panel method. Note that the fine grid TRANAIR solution agrees well with the panel method solution. The leading edge is enlarged in the third plot. Figure 19 shows two other stations from these same solutions.

An F16 fighter aircraft configuration shown in Fig. 20 was analyzed in both codes. The flow conditions are $M_\infty = 0.6$ and $\alpha = 4.0°$. The configuration has 3510 panels, making it quite expensive for panel methods. The TRANAIR run had 162,850 elements. Figure 21 compares surface pressure on the wing at two stations.

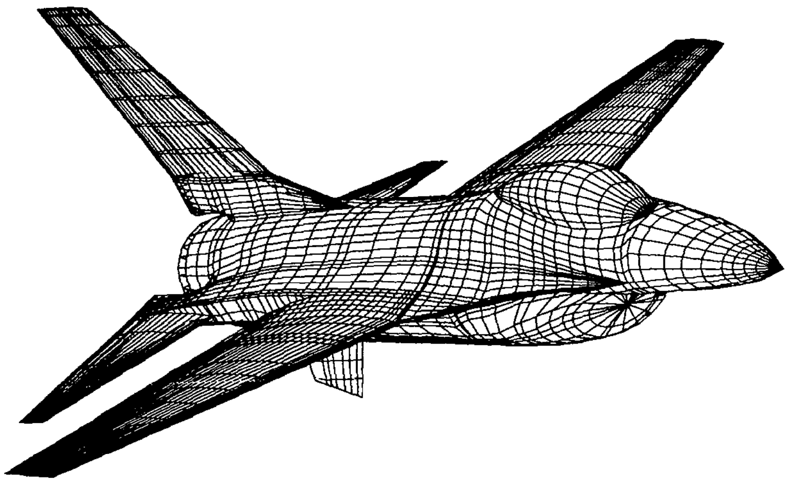
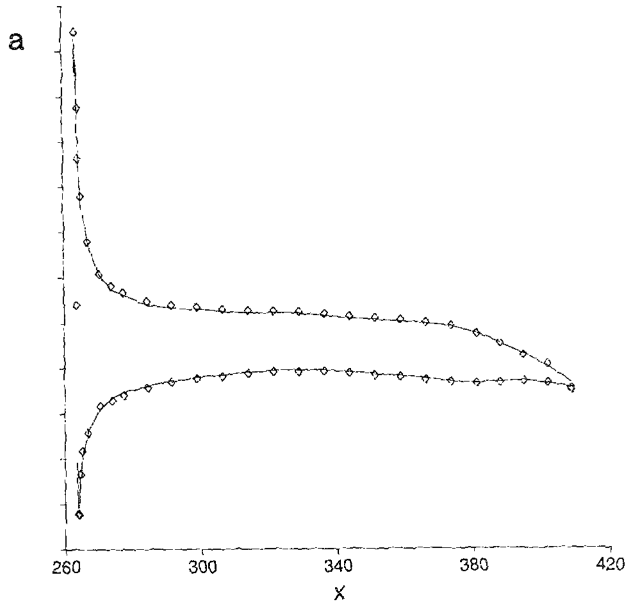


FIG. 20. F16 Aircraft configuration.



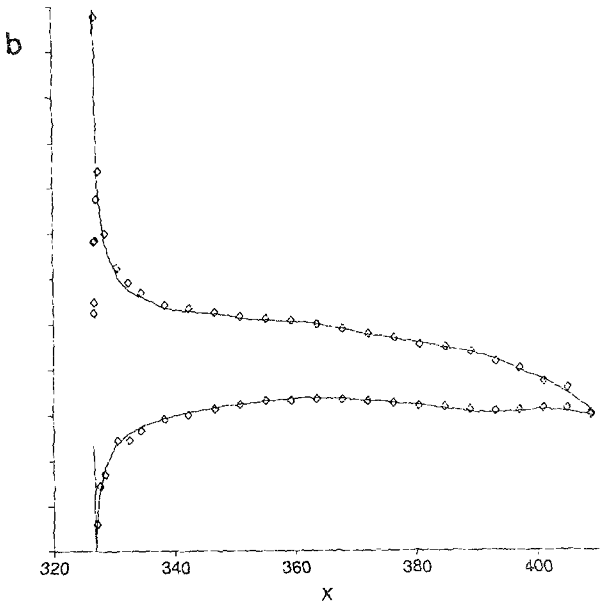


FIG. 21. Surface pressure at two stations on the F16 wing, $M_\infty = 0.6$, $\alpha = 4.0°$: (a) 75% span; (b) 33% span; ——, panel method; ◇, TRANAIR.

It is worth noting that the F16 with tanks and missiles that we discuss later in the Section 9 has 7143 panels, making it too big to run in the panel method. This case, however, runs quite easily in TRANAIR at moderate cost.

8. Nonlinear Solution Algorithm

When the problem is nonlinear, a damped Newton's method is used. Each iteration requires the solution of a linear problem of the type discussed in Section 5. This is accomplished using the preconditioned GMRES algorithm [24, 25].

Newton's method can be described as follows. Suppose we wish to solve the nonlinear system of equations

$$F(x) = 0. \tag{44}$$

Given an initial approximate solution x^0, for $n = 0, 1, 2, \ldots$ until the residual is sufficiently small, set

$$x^{n+1} = x^n + \lambda(\delta x^{n+1}), \tag{45}$$

where δx^{n+1} is the solution of the linear system

$$\bar{F}_{x^n}(\delta x^{n+1}) = -F(x^n) \tag{46}$$

and λ is a step length to be determined. Here $\bar{F}_{x^n}$ is the Jacobian for F linearized about x^n. This linear operator can be defined by giving its action on any vector y,

$$\bar{F}_x(y) = \lim_{\varepsilon \to 0} \frac{F(x + \varepsilon y) - F(x)}{\varepsilon}. \tag{47}$$

The step length λ is selected so that in some appropriate norm, $\|F(x_{n+1})\| < \|F(x^n)\|$. The GMRES algorithm can be used to solve Eq. (46). This algorithm requires only the ability to calculate the action of the linear operator $\bar{F}_x$ on any vector y. Equation (47) can be used to approximate this action

$$\bar{F}_x(y) \simeq \frac{F(x + \varepsilon y) - F(x)}{\varepsilon}, \tag{48}$$

where ε is small in some appropriate sense. Thus, the linear problem, Eq. (46), can ~~be solved without ever explicitly generating the Jacobian for the full nonlinear~~ problem.

To control the cost of the method, the system (46) is solved only approximately with GMRES, i.e., δx^{n+1} satisfies

$$\|\bar{F}_{x^n}(\delta x^{n+1}) + F(x^n)\| < \eta.$$

This makes the method an inexact Newton method [39]. If η is constant, the method converges linearly. If η goes to 0 as convergence takes place, the convergence is superlinear. More details can be found in [25].

Because problems of practical interest are large and not well conditioned, a preconditioned GMRES algorithm is needed to solve Eq. (46) effectively. For the full potential equation, where F is now the nonlinear discrete full potential operator described previously in Section 3, the preconditioning is identical to that used for linear systems and given in Eq. (36). If f is replaced by $-F(x^n)$, L by $\bar{F}_{x^n}$, and $T^{-1}\mathscr{X}$ by x, Eq. (36) is the same as Eq. (46) preconditioned on the left by TN^{-1}. For convenience, the finite difference formula (48) is applied to $TN^{-1}F$ rather than F. The matrix forms of T, N, and T^{-1} are given in Section 5. N is the Jacobian for F about the current solution restricted to a reduced set as described in Section 5 above. The definition of the reduced set is also modified to include all elements where upwinding is used. The matrix N need not be computed every Newton step, an old one often being a sufficiently good preconditioner. The convergence rate for these linearized problems is the same as that discussed in Section 5.

The Jacobian N is generated on an element by element basis using the element stiffness matrices and the density function ρ and its derivatives evaluated at element centroids. For unknowns one of whose eight contributing elements has upwinding in effect, the row of the Jacobian depends on more than just these eight elements. The algorithm can be simplified by applying the chain rule to the calculation of a Jacobian entry,

$$\frac{dF(\Phi)_i}{d\Phi_j} = \frac{\partial F(\Phi)_i}{\partial \Phi_j} + \sum_k \frac{\partial F(\Phi)_i}{\partial \tilde{\rho}_k} \cdot \frac{\partial \tilde{\rho}_k}{\partial \Phi_j}. \tag{49}$$

where $\tilde{\rho}$ is given by Eq. (31). The first term on the right is the contribution from the subsonic stencil, i.e., the element stiffness matrices. The second term is generated using a sparse matrix–matrix multiply. This technique enables vectorization even though the upwinding is element dependent.

Newton's method is rarely globally convergent. Also, its convergence rate is generally quadratic only sufficiently close to the solution. In the full potential case, the initial iterate is taken to be $\phi = 0$, which usually is not a good approximation to the solution. Thus, it is not surprising that Newton's method works well only for small problems or those without shocks. Damping of Newton's must be introduced for large transonic problems to prevent divergence or very slow convergence. This is accomplished in several ways.

We use a damping strategy based on the residual for Eq. (44) due to Bank and Rose [40] for determining the step length λ. This strategy was chosen because of its simplicity and ease of implementation.

Another damping strategy that has proved useful in some cases is to further limit λ so that the solution iterate x^{n+1} does not have local Mach numbers greater that some prescribed cutoff value. This prevents spurious large velocities from causing stagnation of convergence. In the ONERA M6 wing results reported below, this strategy was used with a local Mach number cutoff of $\sqrt{5}$.

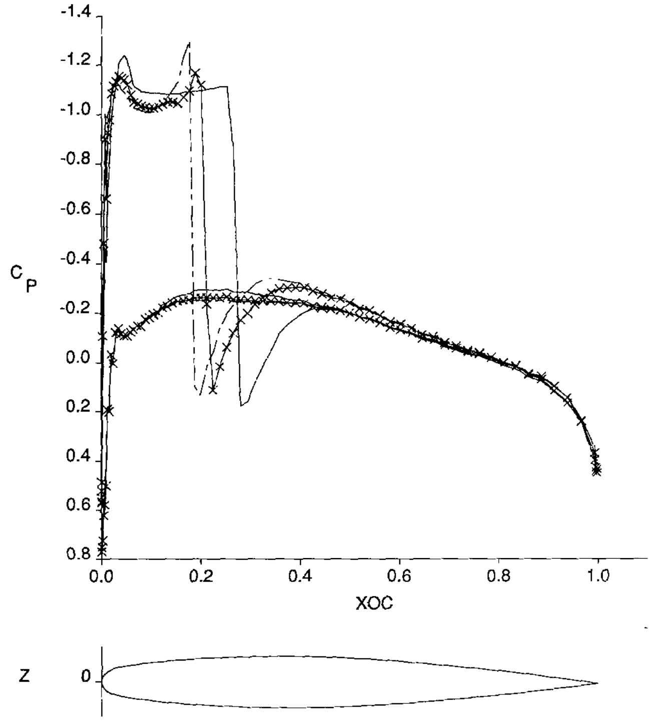


FIG. 22. Iterates for Newton's method with residual and local mach number damping for ONERA M6 wing case, $M_\infty = 0.84$, $\alpha = 3.06°$, 91% span station; ———, converged solution; — — —, sixth Newton iterate; ×, twelfth Newton iterate.

However, local damping procedures of this kind are only adequate by themselves in cases that almost converge anyway. In difficult transonic cases, convergence of Newton's method can stagnate due to the formation of a steep shock in the wrong location early in the iterative process. Once this occurs, a local method can rarely move the shock more than one grid point per iteration, resulting in very slow convergence.

To improve convergence, a problem-dependent dissipation parameter is introduced. This parameter is used in a continuation process called viscosity damping. Initially, the problem is modified by multiplying the switching function of Eq. (32) by a moderate constant (1.5 to 3.0) and by reducing the cutoff Mach number. This has the effect of increasing the amount of artificial viscosity and applying it to a larger part of the flow field. After several Newton steps, the problem is modified by reducing the multiplying factor and raising the cutoff Mach number. This process is repeated until the desired level of dissipation is reached. The continuation parameter is reduced in discrete steps, two to four usually being sufficient. This

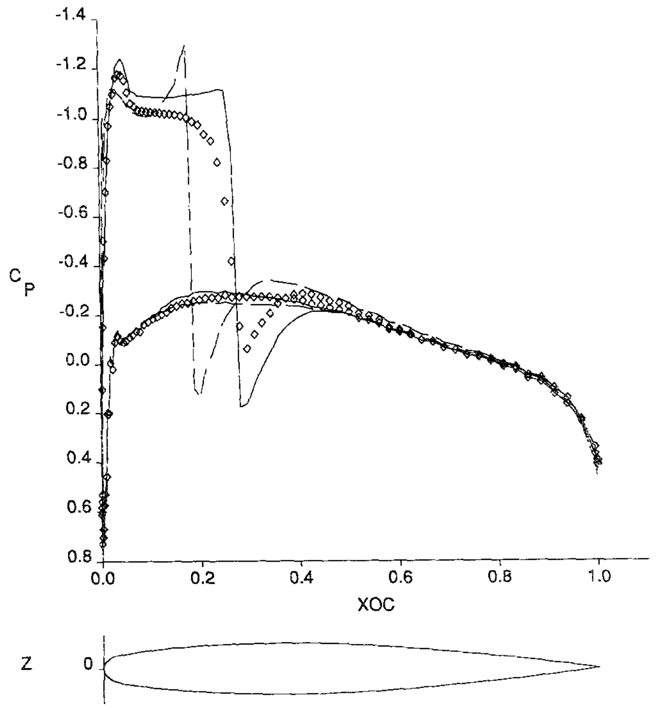


FIG. 23. Partially converged iterate for the second continuation step using viscosity damping for ONERA M6 wing case, $M_\infty = 0.84$, $\alpha = 3.06°$, 91% span station: ———, converged solution; — — —, sixth Newton iterate; ◇, seventh viscous damping iterate.

continuation process has been found to work very well. It has the effect of locating supersonic zones and shock positions fairly early in the process, even though the shocks are quite smeared. The effect of viscosity damping can be seen in the case of the ONERA M6 wing at $M_\infty = 0.84$ and angle of attack $\alpha = 3.06°$ on a grid having about 311,000 elements. This is a case discussed in Section 9 and has a strong shock outboard. If Newton's method is used with an initial iterate $\phi = 0$ and the two damping strategies discussed above for limiting λ, the convergence is very slow. Figure 22 shows the Newton iterate after 6 and 12 Newton steps. The final converged solution is shown for reference. Newton's method is moving the shock toward the correct location very slowly. When viscosity damping is used, convergence is rapid after the initial viscous problems are partially solved. A partially converged solution at the second continuation step (Newton step 7) is shown in Fig. 23. Figure 24 shows the convergence histories for both runs. Each Newton step required 30 to 50 linear GMRES iterations. The residual jumps in this figure correspond to discrete changes in the continuation parameter. The drawback of this continuation approach is the high cost of even partially solving the viscous problems that are introduced.

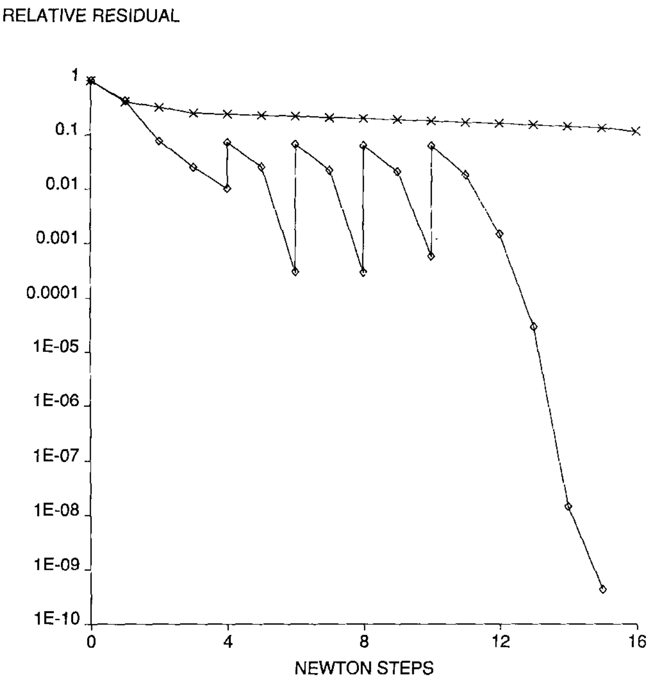


FIG. 24. Convergence histories for Newton's method with various damping strategies for ONERA M6 wing case, $M_\infty = 0.84$, $\alpha = 3.06°$; ×, residual and local Mach number damping; ◇, residual, local Mach number, and viscosity damping.

Grid sequencing has a big payoff and reduces the need for damping of Newton's method. Grid sequencing consists of solving the problem first on a coarse grid, interpolating this solution to a finer grid, and using the interpolant as the initial guess to solve the finer grid problem. This technique will be discussed in detail elsewhere.

9. COMPUTATIONAL RESULTS FOR NONLINEAR FLOW

To demonstrate the nonlinear capabilities of the method, we present solutions for a wide variety of three-dimensional configurations. These include complex fighter and transport configurations as well as the geometries used in Section 7 above for linear flow.

The first case is a sphere with radius 0.8 at a freestream Mach number of 0.7 ($M_\infty = 0.7$). At this condition, the flow is transonic and contains a strong shock. This case was used to test the effectiveness of the upwinding used in TRANAIR. Ideally, the solution should be axially symmetric. The coarse and medium grids pictured in Fig. 25 were used. They correspond to two of the grids used for linear flow about this object. Figure 26 shows surface Mach numbers for both grids as a function of x. Once again, values at all circumferential stations are plotted. The quality and symmetry of the solution are noticeably better on the medium grid. The medium grid also captures the well-known re-expansion phenomenon at the foot of the shock.

A standard aerodynamic test case is the flow about the ONERA M6 wing at $M_\infty = 0.84$ and $\alpha = 3.06°$. This case has an oblique supersonic to supersonic shock

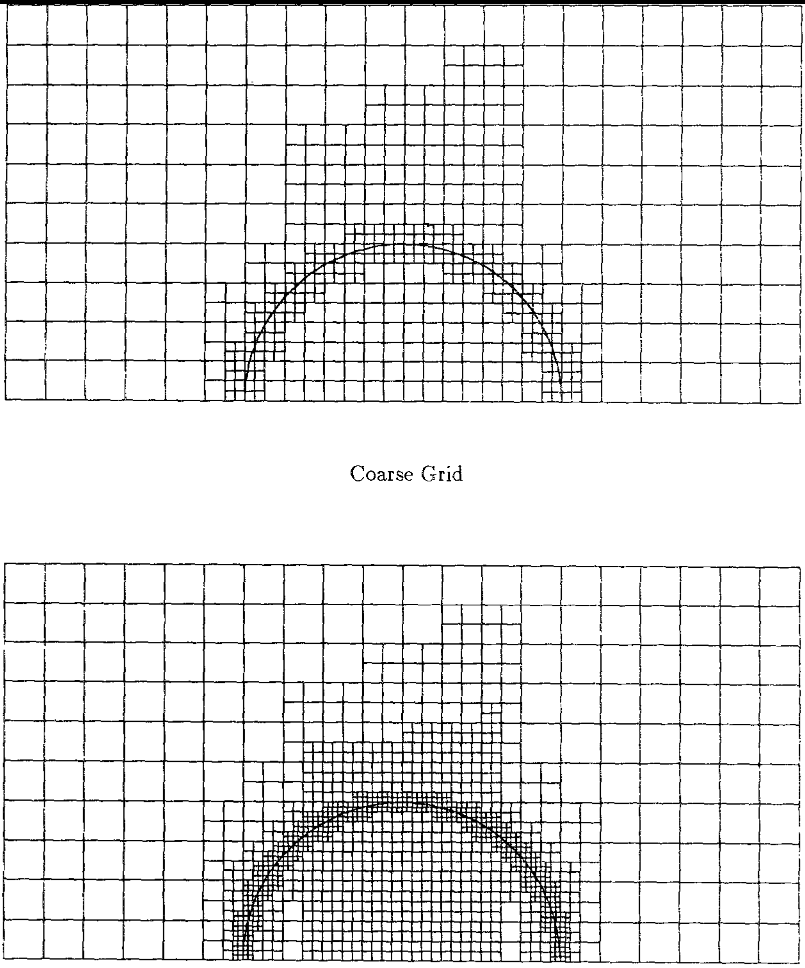
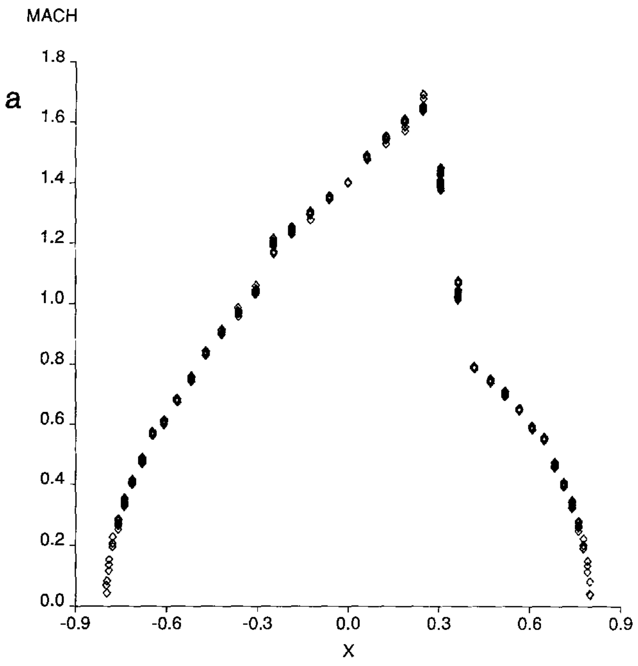


FIG. 25. Cuts through two grids for a sphere in transonic flow, $M_\infty = 0.7$.



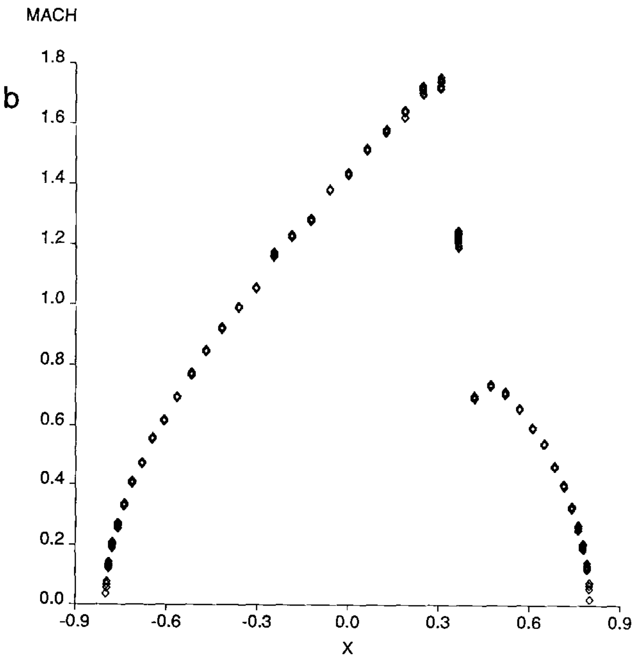
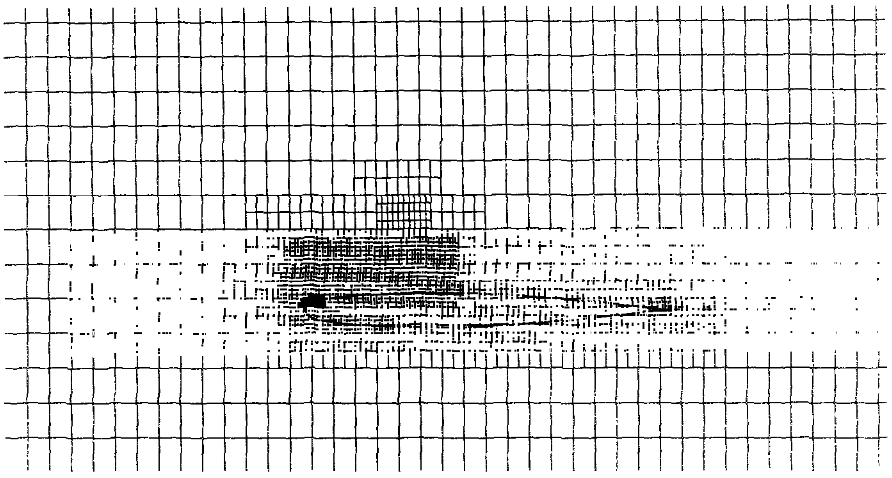
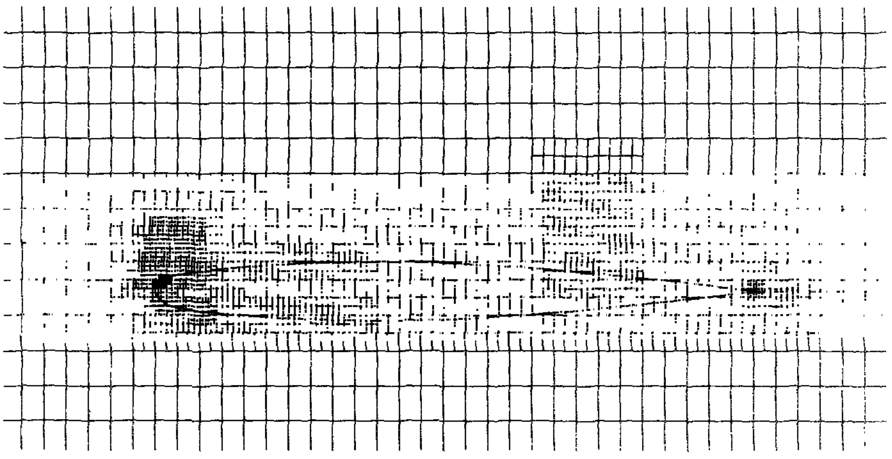


FIG. 26. Surface Mach numbers for sphere, $M_\infty = 0.7$: (a) coarse grid; (b) medium grid.

as well as a supersonic to subsonic shock. Here, TRANAIR results are compared to those obtained using the FLO28 code of Jameson [6] and a grid with 364,000 points. FLO28 is a full potential code widely used throughout the aerospace industry that is well suited to simple wing geometries. The TRANAIR grid had about 311,000 elements. Dense grids were used in both codes to accurately capture the oblique shock. In Fig. 27 two vertical cuts through the TRANAIR grid for this case are shown. Figure 28 is a waterline cut through the grid, and Fig. 29 compares



90% Span



Plane of Symmetry (Root)

FIG. 27. Two cuts through grid for ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06°$.

surface pressures at four stations with those obtained with FLO28. The TRANAIR solution agrees quite well with the FLO28 solution using first-order dissipation. It is unclear in this case whether the second-order dissipation FLO28 solution offers improved accuracy. In this problem, TRANAIR obtained comparable accuracy at comparable cost. Viscosity damping is required in this case (see Section 8) and accounts for about $\frac{2}{3}$ of the TRANAIR iteration steps.

The F16 shown in Fig. 20 of Section 7 was analyzed at $M_\infty = 0.9$ and $\alpha = 4.0°$. The TRANAIR grid had about 189,000 elements. A comparison of computed surface pressure with wind tunnel data at two wing stations is shown in Fig. 30. The agreement is good considering the fact that boundary layer effects are not yet included in TRANAIR. The shock position at the outboard station is affected by the fact that the wind tunnel model had wing tip missile launchers installed but the panel model provided to us for the computation did not. Another configuration of interest is the F16 with tanks and missiles shown in Fig. 31. The grid generation

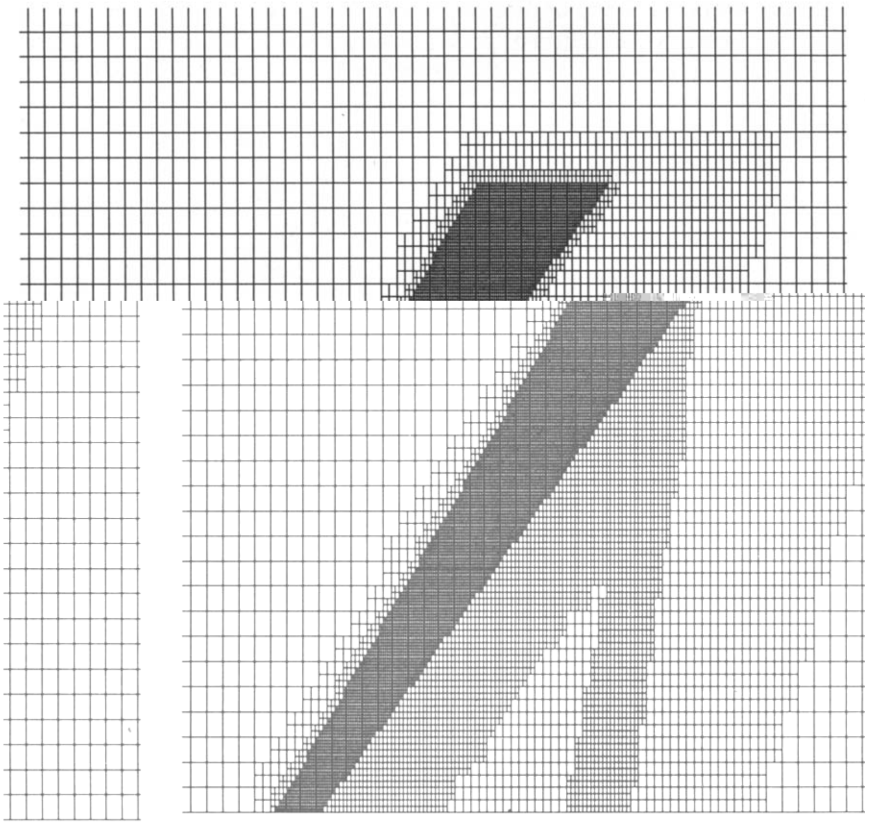


FIG. 28. Waterline cut through grid for ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06°$.

required makes this case very difficult for surface-fitted grid codes. The TRANAIR grid contained 215,960 elements. Figure 32 shows three plane cuts through the computational grids. Figure 33 compares computed surface pressure just inboard and just outboard of the tank strut with TRANAIR results for the F16 without tanks and missiles. The effect of the tank is as expected.

The final case considered is flow about a 747-200 transport configuration with wing, body, struts, and nacelles. The geometry description includes about 23,000 panels (cf. Fig. 34) and is many times more complex than that which can be handled by panel codes. Flow conditions were taken to be $\alpha = 2.7^\circ$ and $M_\infty = 0.8$. This is approximately the largest freestream Mach number at which an inviscid solver can obtain accurate results without boundary layer coupling.

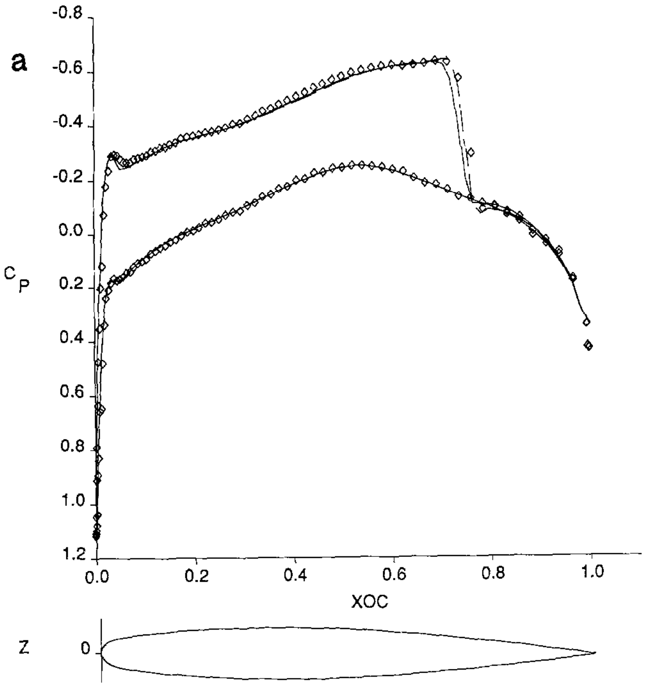
The grid used for this problem consisted of approximately 219,000 finite elements. Figure 35 shows two cuts through the grid. The cut shown in Fig. 35A is a yz plane cut and passes through the outboard nacelle strut and core cowl and through the prescribed wakes behind the inboard strut and nacelle. The cut shown in Fig. 35B is an xz plane cut through the outboard nacelle.

Figure 36 compares TRANAIR results with wind tunnel pressure data at four span stations of the wing. Overall, one sees very good agreement with experiment. Most of the differences are seen in the upper surface pressures and are attributable to viscous boundary layer effects not currently modeled in TRANAIR. By comparing lower surface pressure profiles, one can clearly see the effect of the outboard nacelle at the 69% span station, which is shown in Fig. 35B. Near the leading edge of the wing at the 69% span station the TRANAIR solution contains a supersonic to supersonic shock whose presence is supported by the experimental data.

10. Cost of the Method

Elapsed CPU times and external storage requirements of the present implementation of the method presented in this paper are summarized in the log–log plots shown in Figs. 37, 38, and 39. Data for the figures was obtained from single processor CRAY X-MP runs for almost all of the problems described in Sections 7 and 9, as well as other typical aerodynamics applications. In all nonlinear cases, the [illegible]

that the method has several user-specified parameters that affect performance, for example, the drop tolerance used in the sparse solver, the number of continuation steps used in the viscosity damping strategy, and the number of Jacobians computed. In several cases where a preliminary version of grid sequencing was used, total CPU requirements were reduced by a factor of 2 over those reported here and external storage requirements by 20%. Solution CPU cost was reduced by a factor of 3. Two such cases are the transonic sphere and the transonic ONERA M6 wing discussed in Section 9. These results will be reported elsewhere.



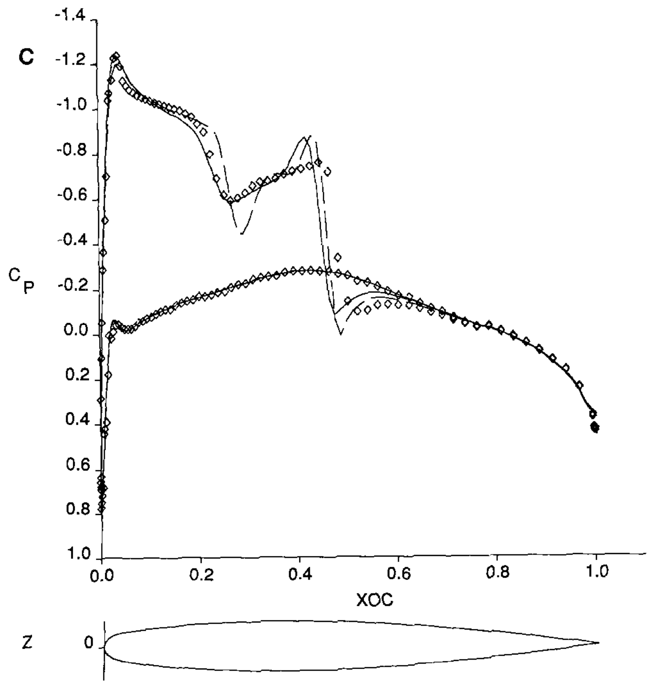


FIG. 29. Comparison of surface pressure at four span stations on ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06°$: (a) plane of symmetry (root); (b) 44% span; (c) 70% span; (d) 91% span; ———, FLO 28 first-order viscosity; – –, FLO 28 second-order viscosity; ◇, TRANAIR.

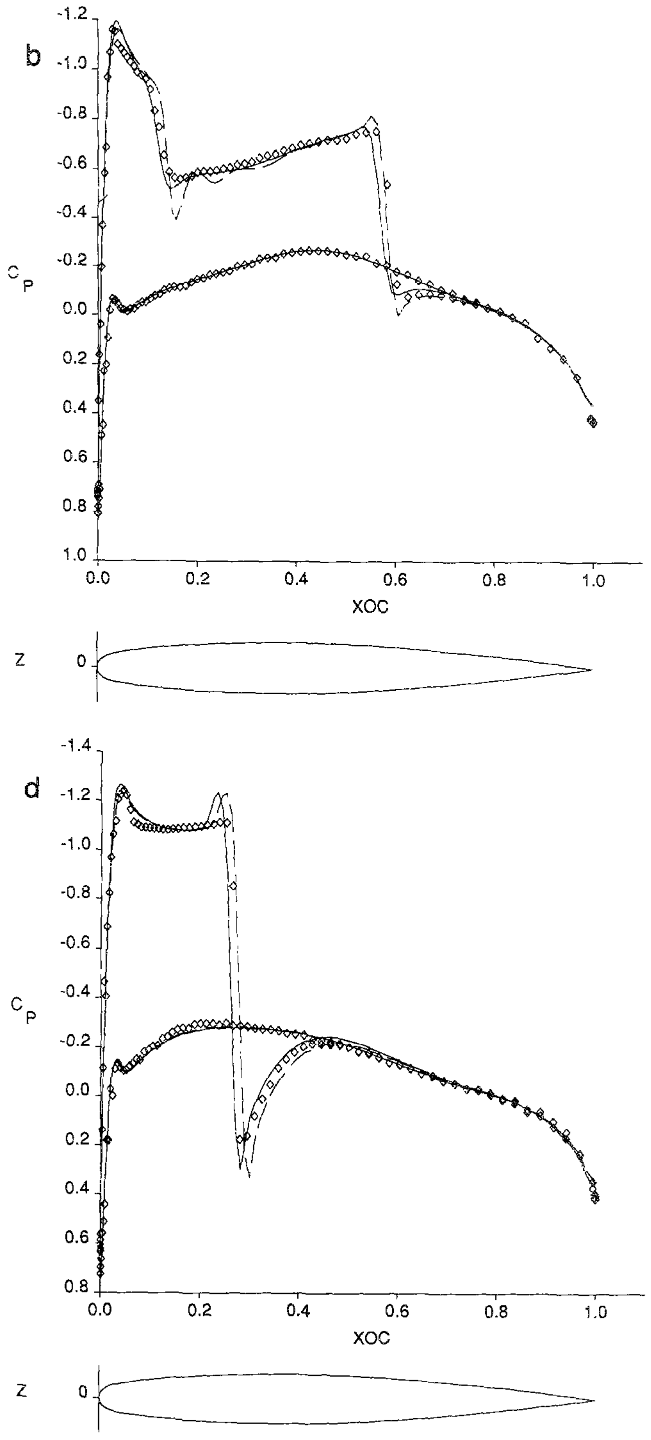
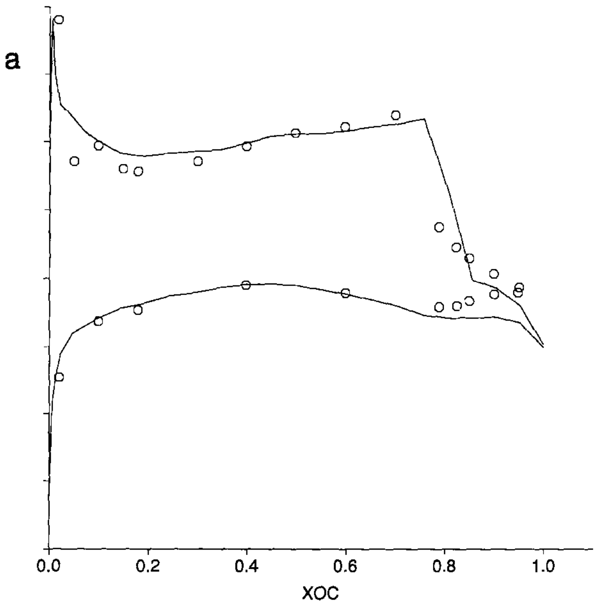


FIG. 29—*Continued*



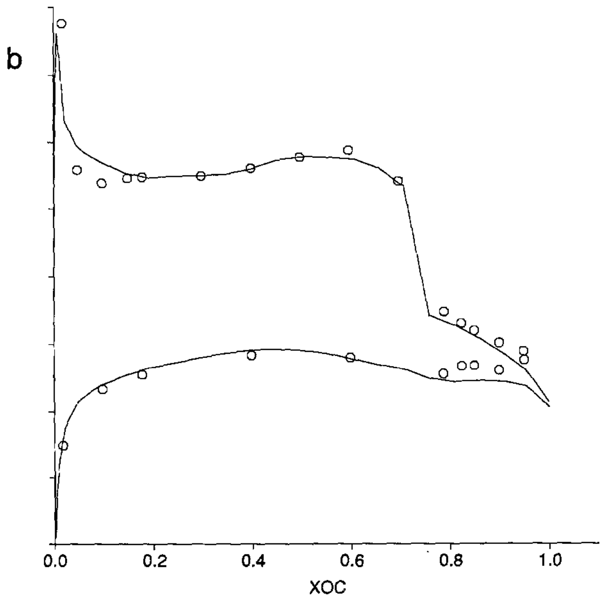


FIG. 30. Wing pressures for the F16, $M_\infty = 0.9$, $\alpha = 4.0°$: (a) 45% span; (b) 71% span; ——, TRANAIR; ○, wind tunnel test data. (Note: lines are TRANAIR and ○ are wind tunnel test data.)

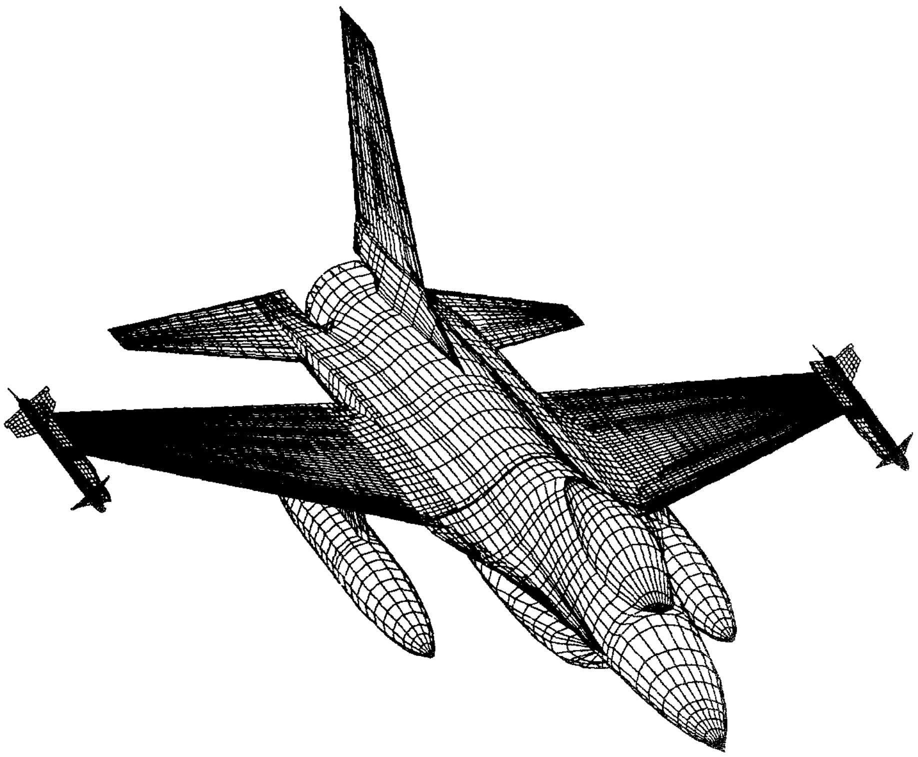
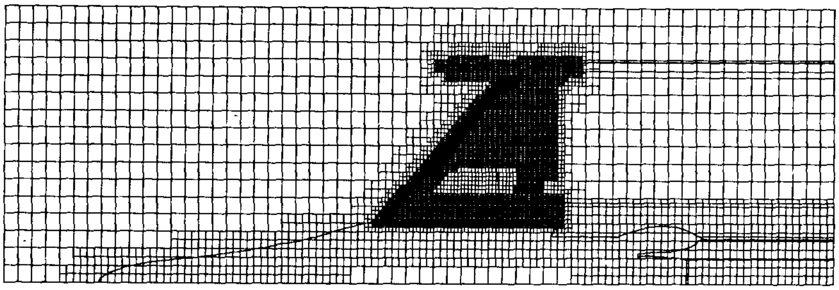


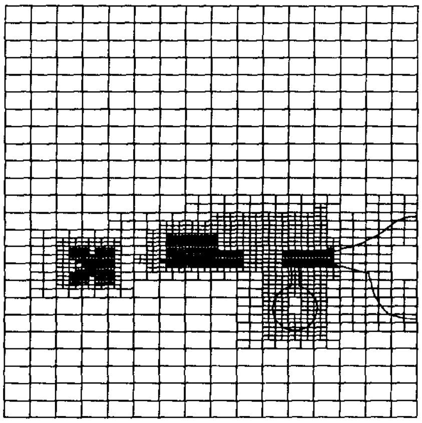
FIG. 31. F16 aircraft configuration with tanks and missiles.

Figure 37 indicates that the overhead CPU cost of the method increases linearly with the number of degrees of freedom N. Overhead consists primarily of generating the locally refined grid and computing the finite element operators. Figure 38 summarizes the method's solution CPU time, which is the total CPU time less that for overhead. The least squares linear fits of the data shown in the figure indicate that solution CPU times increase approximately as $\mathcal{O}(N^{1.1})$ and $\mathcal{O}(N^{1.2})$ as $N \to \infty$ for linear and nonlinear problems, respectively. These data are sensitive to selection of a good dynamic drop tolerance in the method's reduced set sparse solver preconditioner and we do not know if our choices are optimal. Based on asymptotic analysis [41] for linear problems, it is reasonable to expect solution CPU times to grow no more rapidly than $\mathcal{O}(N^{7/6}) = \mathcal{O}(N^{1.17})$. Other data [42] suggest that $\mathcal{O}(N)$ is possible, however.

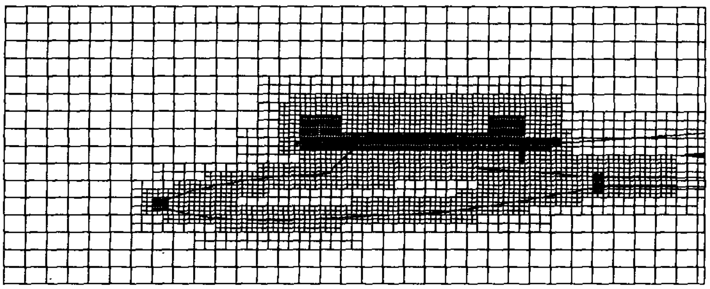
The Cray SSD storage required by the method is summarized in Fig. 39. While it is not possible to be rigorous, one sees that storage increases approximately linearly with N. This is sensitive to the value chosen for the drop tolerance used in the decomposition of the reduced set Jacobian matrix described in Section 5. The number of nonzeros in this matrix is usually 20 to $50N_r$, where N_r is the number



Waterline Cut Through wing

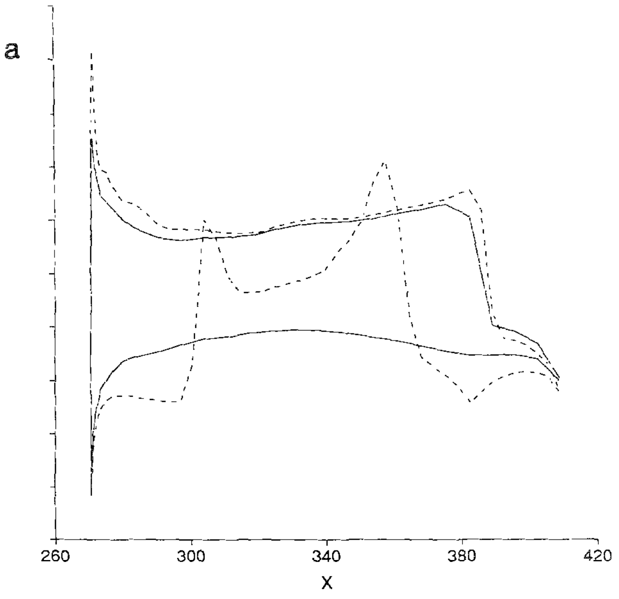


Vertical Cut



Cut Through Underwing Tank and Strut

FIG. 32. Cuts through the grid for the F16 with tanks and missiles, $M_\infty = 0.9$, $\alpha = 4.0°$.



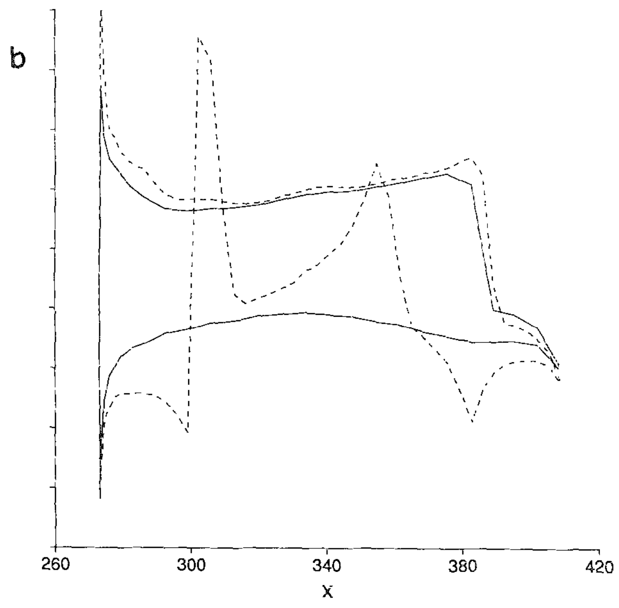


FIG. 33. Computed wing pressures for the F16 with tanks and missiles, $M_\infty = 0.9$, $\alpha = 4.0°$: (a) inboard side of strut; (b) outboard side of strut; ———, F16 without tanks and missiles; - - -, F16 with tanks and missiles.

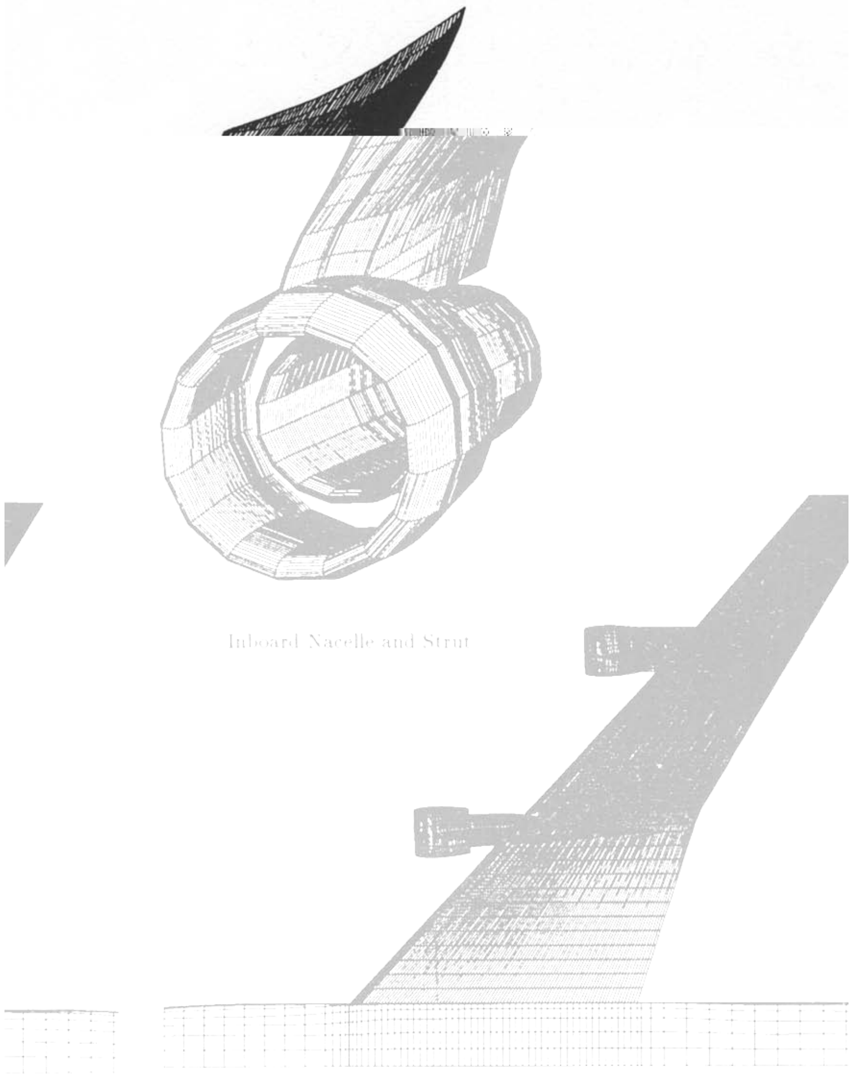
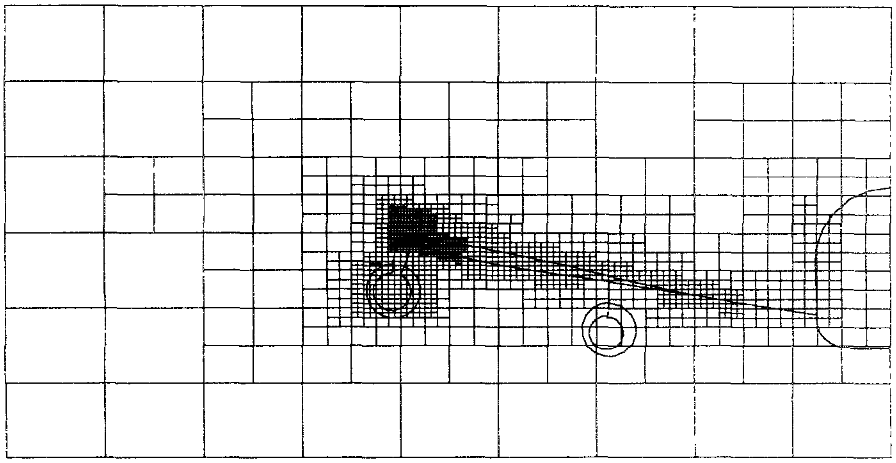
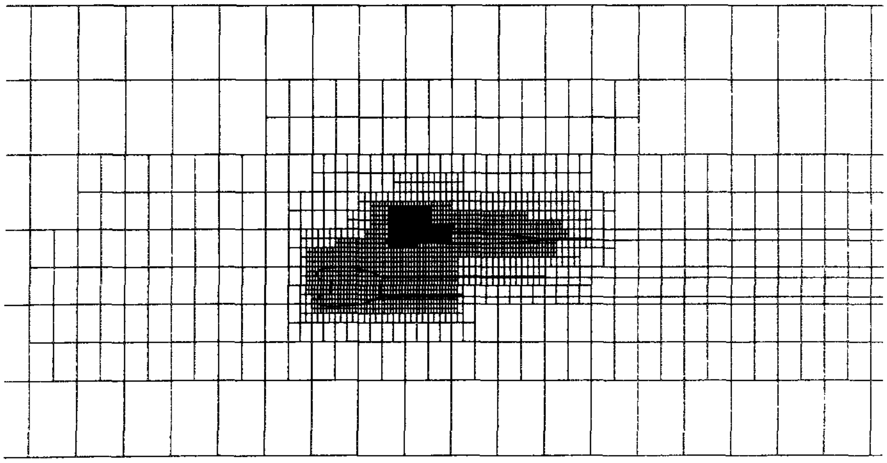


FIG. 34. 747-200 transport configuration.

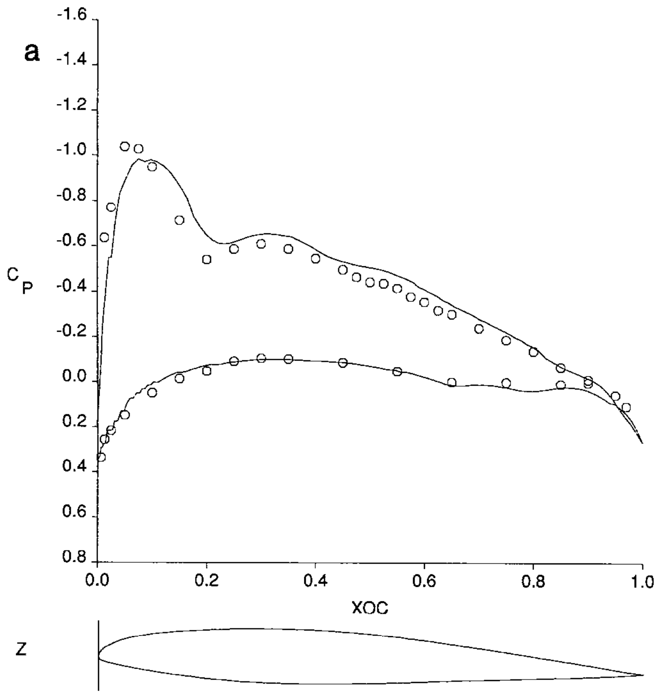


A. Constant x Cut Behind Inboard Nacelle



B. Constant y Cut Through Outboard Nacelle

FIG. 35. Two cuts through TRANAIR grid for 747-200 case.



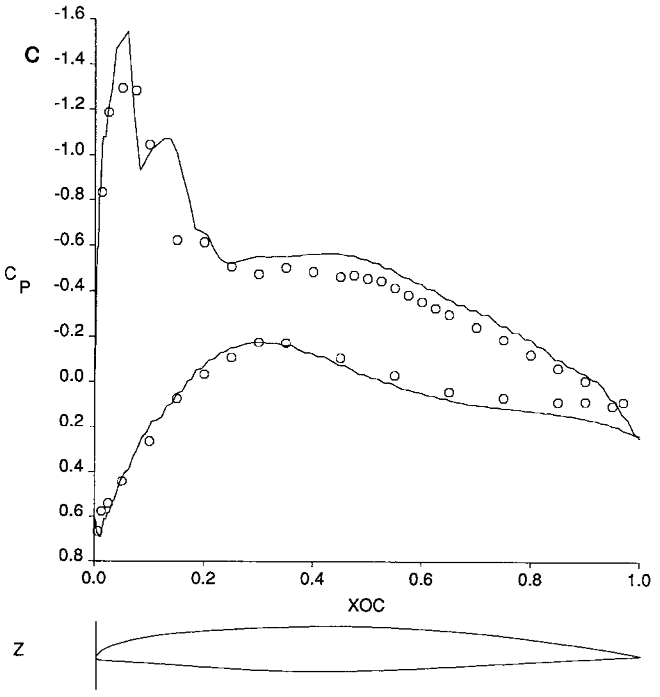
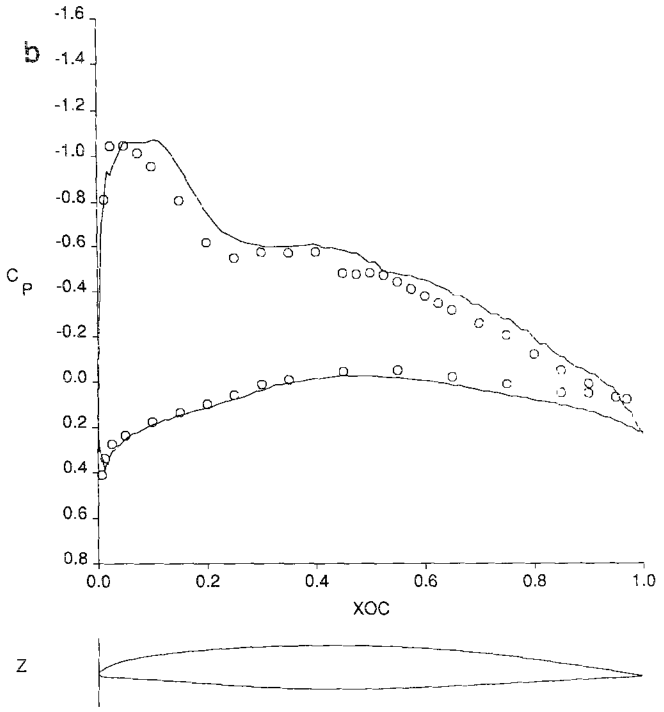


FIG. 36. Wing pressures for 747-200, $M_{\infty} = 0.8$, $\alpha = 2.7°$. (a) 26% span; (b) 60% span: (c) 69% span; (d) 81% span; ○, wind tunnel test data; ———, TRANAIR.



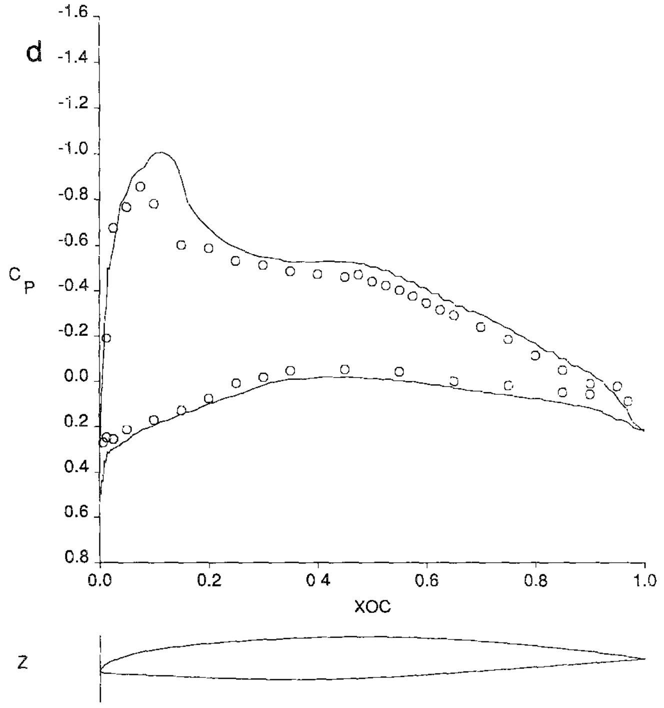


FIG. 36—*Continued*

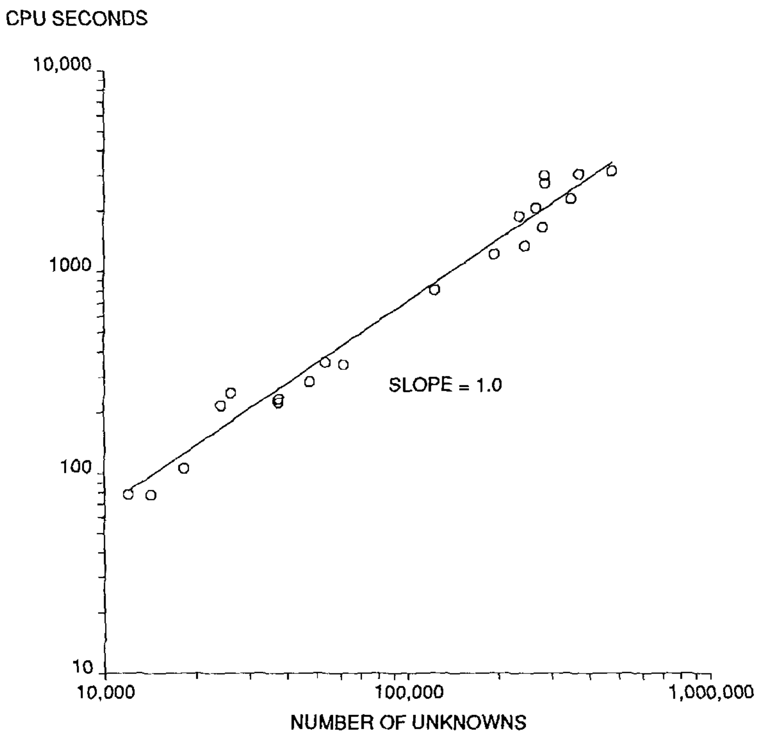


FIG. 37. Overhead CPU cost for TRANAIR.

of unknowns in the reduced set. N_r is usually between $0.65N$ and $0.7N$. The choice of the drop tolerance is guided by the following rule of thumb. If the number of nonzeros in the incomplete factors of the Jacobian matrix is about twice the number of nonzeros in the original Jacobian matrix, the incomplete factorization is a good preconditioner. Each nonzero in the decomposition requires two words of SSD storage because of column indices. Thus, the SSD storage required for the incomplete factorization would be between $50N$ and $140N$. Were a drop tolerance not employed, one could expect [41] the storage requirement in this phase to be at least $\mathcal{O}(N_r^{4/3})$. This would make many of the applications considered here unfeasible on today's supercomputers, since in most of the present runs storage for the incomplete decomposition with drop tolerance accounts for 30 to 40% of the total SSD space used.

11. HELMHOLTZ APPLICATIONS

While the present paper is devoted to aerodynamics, in two other application areas, acoustics and electromagnetics, similar techniques are applicable [20]. In

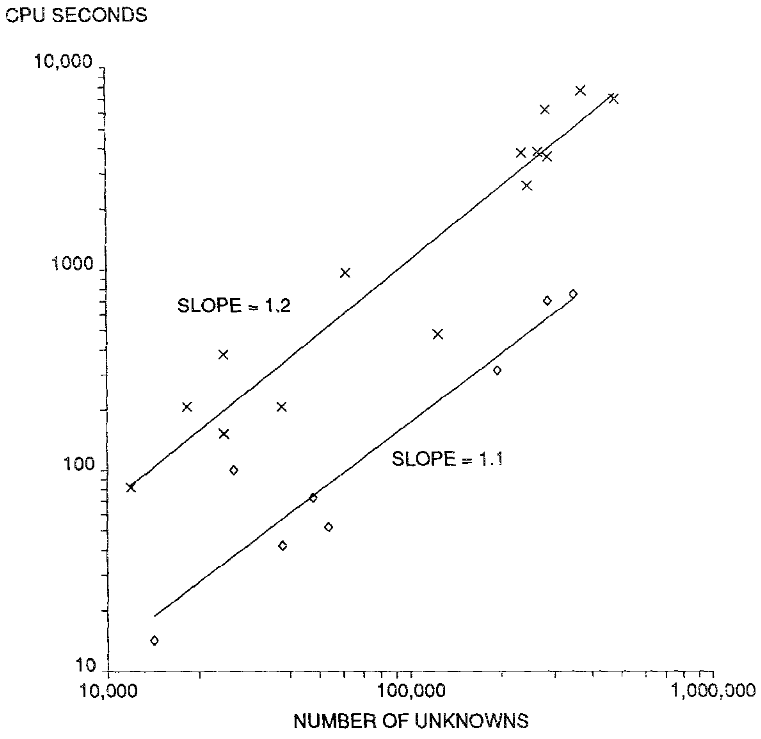


FIG. 38. TRANAIR solution CPU cost: ×, nonlinear problems: ◇, linear problems

time harmonic acoustics with frequency ω, a generalization of the scalar linear Helmholtz equation

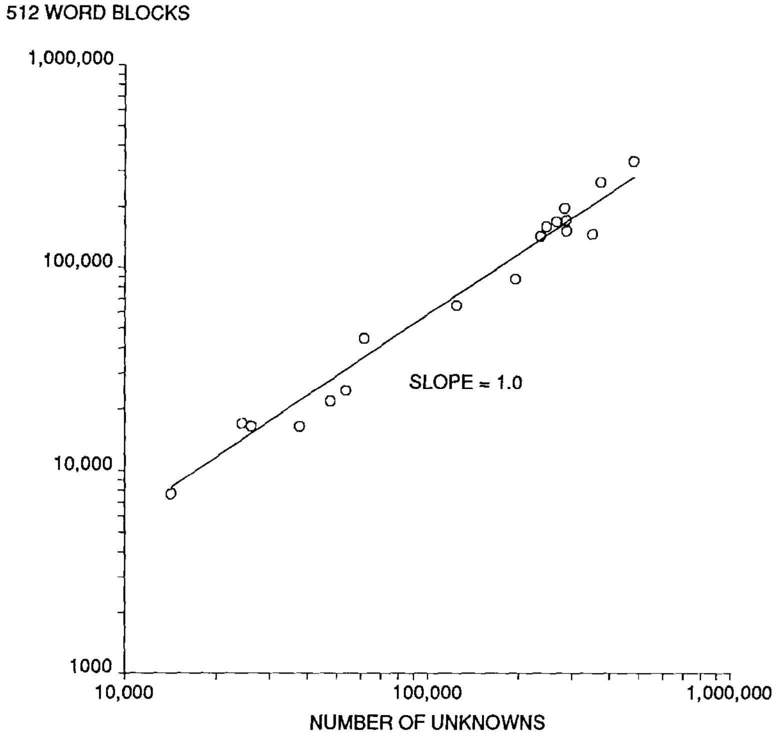
$$\mathscr{L}(\phi) \equiv \nabla \cdot \rho \, \nabla\phi + k^2\phi = f \tag{50}$$

is to be solved, where $k = k(x, y, z) = \omega/c(x, y, z)$ is the wave number, $c(x, y, z)$ is the local speed of sound, and $\rho = \rho(x, y, z)$ is the density of the medium. All variables are complex valued and the variational principle involves the standard Galerkin functional

$$J = \int_{\Omega} (\rho \, \nabla\phi \cdot \nabla\phi - k^2\phi^2 + 2f\phi) \, dV. \tag{51}$$

In this case the natural boundary condition is $\rho(\partial\phi/\partial n) = 0$ and the far field condition is the Sommerfeld radiation condition

$$\frac{\partial\phi}{\partial R} - \imath k\phi = \mathcal{O}\left(\frac{1}{R}\right) \tag{52}$$



[illegible]

as $R \to \infty$. Results obtained in acoustics using the method can be found in Ref. [20, 43].

In electromagnetics, time harmonic Maxwell's equations with frequency ω reduce to a generalized vector Helmholtz equation

$$\mathscr{L}(\mathbf{H}) \equiv \mathbf{H} - \beta \nabla \times (\alpha \nabla \times \mathbf{H}) = 0, \tag{53}$$

where $\mathbf{H}$ is the magnetic field, $\beta^{-1} = -\iota\omega\mu$, $\alpha^{-1} = \iota\omega\varepsilon_{\text{real}} + \sigma$, ε is the electric permittivity, μ is the magnetic permeability, and σ is the electric conductivity. All variables are complex valued. For scattering from a conducting body with boundary $\partial\Omega_1$, the functional can be taken to be

$$J = \frac{1}{2\iota\omega} \int_\Omega \left(\frac{1}{\alpha} \mathbf{E} \cdot \mathbf{E} - \frac{1}{\beta} \mathbf{H} \cdot \mathbf{H} \right) dV + \frac{1}{\iota\omega} \int_{\partial\Omega_1} (\mathbf{H} \cdot (\mathbf{n} \times \mathbf{E}))\, dS, \tag{54}$$

where $\mathbf{E}$ is the electric field, $\mathbf{E} = \alpha \nabla \times \mathbf{H}$, and the boundary condition is $\mathbf{n} \times \mathbf{E} = 0$ on $\partial\Omega_1$ with an appropriate far field condition [44]. This is a nonstandard functional that is applicable to lossy materials [45]. Results of applying the method to this problem can be found in [45].

These problems can be treated with virtually the same numerical method as that given above for the full potential of aerodynamics, with certain simplifications. For example, no artificial dissipation is needed in either acoustics or electromagnetics. In what follows, only the main differences will be discussed.

In both of these areas, computational experience has been limited to cases using a uniform grid, i.e., a grid without local refinement, even though the full local refinement capability is available. Because of the necessity of resolving the wave throughout the computational domain, such a uniform grid is not very wasteful for many problems of interest (for example, scattering from a perfect conductor). Also, there may be grid interface effects to be considered when local refinement is used.

In the acoustics case, $\mathscr{T}$, the far field operator discussed in Section 3.3, is a constant coefficient version of the Helmholtz operator in Eq. (50), where the coefficients are those of the external medium. In electromagnetics, $\mathscr{T}$ is a constant coefficient version of Eq. (53).

The operator $\mathscr{L}$ in both electromagnetics and acoustics is linear and the operator coefficients (α, β, ρ, and k^2) are all functions of the spatial variables only. In the finite element formulation, these coefficients are approximated by piecewise constant functions. In many cases, only a few materials are present, so only a few values of coefficients need to be stored. In acoustics, the standard trilinear element basis function can be used. In electromagnetics, however, the unknown is a vector and use of a trilinear basis function for each component results in a large stencil. A vector bilinear element basis function has been developed that is bilinear for each component of the vector. This element and the resulting discrete operator are described in detail in [45].

In electromagnetics, the equivalent of a stagnation region is a perfect conductor. Boxes interior to perfect conductors do not generate D regions, and the equation is simply that the field is zero there.

The sparse solver preconditioner is used in a somewhat different way in acoustics and electromagnetics. The reduced set is defined in the same way, but the boundary condition at closure points must be a discrete approximation to the appropriate far field radiation condition. When a uniform grid is used, the closure points are quite close to the configuration boundaries. In some cases, such as those involving interior resonance, this results in poor preconditioning. In these cases, the reduced set must be extended so that the closure points are outside the resonance region. This phenomenon is believed to be related to the fact that the closure point radiation condition is not even approximately satisfied by the actual solution.

In acoustics and electromagnetics, nested dissection is quite effective, since the reduced set is often just a thin layer of points near boundary surfaces. This is the case, e.g., for electromagnetic scattering from a conducting body at a fairly high frequency in which a fine global grid is required. This situation produces a reduced set that is essentially two-dimensional. Thus, the problems can often be solved without using a drop tolerance. When a drop tolerance is used, the resulting system can become unstable without some added dissipation. This dissipation can be introduced by adding an imaginary component to the wavenumber in the N operator, simulating a lossy material in which waves are damped with distance and preventing resonance. When this is done, significant degradation of the convergence rate results. For many large scattering problems, the SSD storage required for the sparse decomposition is the limiting factor for the method, so that accepting significant slowing of convergence is necessary to obtain an answer.

Since these problems are linear, no Newton method is needed and a simple linear GMRES method is used. With a complete factorization of the reduced set problem, convergence is typically achieved in 10 to 20 GMRES iterations, indicating very good preconditioning.

More details concerning these applications can be found in Refs. [20, 43, 45].

12. Future Directions

The method described in this paper is fully implemented for aerodynamics applications in the computer program TRANAIR, which is a valuable engineering tool for aircraft analysis and design. Most of the method's important algorithms also have been implemented for acoustics and electromagnetics applications. However, there are several capabilities that could substantially improve the present methodology and these are described here.

As discussed above, grid sequencing decreases CPU time and external storage significantly and provides more reliable convergence. This procedure is implemented in TRANAIR and will be discussed in detail elsewhere.

The current technique for generating the computational grid requires user knowledge of solution characteristics to input parameters that control refinement where it is required. While general rules about required grid can be given based on the type of object being modeled, this process is not easily applicable to new situations and is suboptimal. An automatic solution adaptive refinement capability would greatly increase the reliability of the method and reduce its cost. Such a capability has been implemented and is being tested.

At high subsonic Mach numbers, boundary layer effects substantially change shock strength and position. Inclusion of these effects would extend the applicability of the method. This will be done by coupling the current code with a boundary layer or thin layer Navier–Stokes capability.

The finite element discretization discussed in this paper generalizes immediately to the case of higher order elements. The advantage of a Cartesian grid finite element method is evident since the storage required for the element stiffness matrix grows rapidly as the order of the element is increased. When almost all of the elements have the same element stiffness matrix, very large savings in storage can be achieved.

13. CONCLUSION

A new finite element method has been presented that is designed for automatic generation of discretizations for complicated geometries in three space dimensions without using standard grid generation techniques. Computational results have been presented demonstrating the accuracy and reliability of the method.

ACKNOWLEDGMENTS

Many people aided us in this work. Larry Wigton, Ki Lee, Richard Burkhart, and Bjarne Everson were members of the team that produced the uniform grid code [20, 46], many of whose components are used in the present method. Gautam Sengupta and Richard Burkhart have been invaluable in the acoustics applications. Mike Madson and Alex Woo of NASA AMES have helped us in many ways over the years. In particular, Mike Madson provided us with the F16 geometries, the test data, and much help in running these cases.

REFERENCES

1. R. E. BANK, in *Elliptic Problem Solvers*, edited by Martin Schultz (Academic Press, New York, 1981), p. 1.
2. R. E. BANK, University of California, San Diego, Department of Mathematics Technical Report, 1988 (unpublished).
3. I. BABUŠKA AND W. C. RHEINBOLDT, in *Elliptic Problem Solvers II*, edited by G Birkhoff and A. Schoenstadt (Academic Press, New York, 1984), p. 345.
4. B. A. SZABO, in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, edited by I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. de A. Oliveira (Wiley, New York, 1986), p. 61.
5. F. T. JOHNSON, NASA CR-3079, 1979 (unpublished).
6. A. JAMESON, Princeton University Department of Mechanical and Aerospace Engineering Report 1651, 1983 (unpublished).
7. T. L. HOLST AND W. F. BALLHAUS, *AIAA J.* **17**, 145 (1979).
8. A. JAMESON, W. SCHMIDT, AND E. TURKEL, AIAA Paper 81-1259, 1981 (unpublished).
9. A. JAMESON AND T. J. BAKER, AIAA Paper 83-1929, 1983 (unpublished).
10. A. JAMESON. T. J. BAKER, AND N. P. WEATHERILL, AIAA Paper 86-0103, 1986 (unpublished).
11. R. B. PELZ AND A. JAMESON, AIAA Paper 83-1922, 1983 (unpublished).
12. R. LÖHNER, K. MORGAN, J. PERAIRE, AND O. C. ZIENKIEWICZ, AIAA Paper 85-1531, 1985 (unpublished).
13. R. LÖHNER, K. MORGAN, AND O. C. ZIENKIEWICZ, in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, edited by I. Babuška, O.C. Zienkiewicz, J. Gago, and E. R. de A. Oliveira (Wiley, New York, 1986), p. 281.
14. R. GLOWINSKI, *Numerical Methods for Nonlinear Variational Problems* (Springer-Verlag, New York, 1984).
15. B. WEDAN AND J. C. SOUTH, AIAA Paper 83-1889, 1983 (unpublished).
16. T. A. REYHNER, NASA CR-3814 (unpublished).
17. R. J. LEVEQUE, *J. Comput. Phys.* **78**, 36 (1988).
18. R. J. LEVEQUE, in *Numerical Methods for Fluid Dynamics III*, edited by J. W. Morton and M. J. Baines (Clarendon Press, Oxford, 1988), p. 375.
19. F. T. JOHNSON, R. M. JAMES, J. E. BUSSOLETTI, A. C. WOO, AND D. P. YOUNG, AIAA Paper 82-0953, 1982 (unpublished).

20. P. E. Rubbert, J. E. Bussoletti, F. T. Johnson, K. W. Sidwell, W. S. Rowe, S. S. Samant, G. SenGupta, W. H. Weatherill, R. H. Burkhart, B. L. Everson, D. P. Young, and A. C. Woo, in *Comptational Mechanics — Advances and Trends*, edited by A. K. Noor (Amer. Soc. Mech. Eng., New York, 1986), p. 49.
21. O. Buneman, *J. Comput. Phys.* **8**, 500 (1971).
22. R. A. James, *J. Comput. Phys.* **25**, 71 (1977).
23. R. W. Hockney, in *Methods in Computational Physics, Vol. 9* (Academic Press, New York, 1969), p. 135.
24. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
25. L. B. Wigton, N. J. Yu, and D. P. Young, AIAA Paper 85-1494, 1985 (unpublished).
26. D. P. Young, R. G. Melvin, F. T. Johnson, J. E. Bussoletti, L. B. Wigton, and S. S. Samant, *SIAM J. Sci. Stat. Comput.* **10**, 1186 (1989).
27. NASA Ames Research Center Contractor's Report, Contract NAS2-11851, 1987 (unpublished).
28. P. E. Rubbert and G. R. Saaris, AIAA Paper 72-188, 1972 (unpublished).
29. H. Bateman, *Proc. Nat. Acad. Sci.* **16**, 816 (1930).
30. A. Weiser, Yale University Department of Computer Science Technical Report 213, 1981 (unpublished).
31. S. S. Samant, J. E. Bussoletti, F. T. Johnson, R. G. Melvin, and D. P. Young, in *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, Vol. 323* (Springer-Verlag, Berlin, 1989), p. 518.
32. H. Samet, *Comput. Surveys* **16**, 68 (1984).
33. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method* (Prentice–Hall, Englewood Cliffs, NJ, 1973).
34. M. Hafez, E. M. Murman, and J. C. South, AIAA Paper 78-1148, 1978 (unpublished).
35. D. E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching* (Addison–Wesley, London, 1973).
36. A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice–Hall, Englewood Cliffs, NJ, 1981).
37. O. C. Zienkiewicz, L. Xi-Kui, and S. Nakazawa, *Commun. Appl. Num. Methods* **1**, 3 (1985).
38. J. H. Bramble and A. H. Schatz, *Math. Comput.* **31**, 94 (1977).
39. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *SIAM J. Num. Anal.* **19**, 400 (1982).
40. R. E. Bank and D. J. Rose, *Num. Math.* **37**, 279 (1981).
41. O. Axelson and V. A. Barker, *Finite Element Solutions of Boundary Value Problems: Theory and Computation* (Academic Press, New York, 1984).
42. Z. Zlatev, J. Wazniewski, and K. Schaumburg, *SIAM J. Sci. Stat. Comput.* **3**, 486 (1982).
43. G. SenGupta, AIAA Paper 87-2669, 1987 (unpublished).
44. C. Müller, *Foundations of the Mathematical Theory of Electromagnatic Waves* (Springer-Verlag, Berlin, 1969).
45. Air Force Flight Dynamics Laboratory Technical Report AFFDL-TR-87-3082, 1987 (unpublished).
46. S. S. Samant, J. E. Bussoletti, F. T. Johnson, R. H. Burkhart, B. L. Everson, R. G. Melvin, D. P. Young, L. L. Erickson, M. D. Madson, and A. C. Woo, AIAA Paper 87-0034, 1987 (unpublished).