Numerical Solution of Transonic Wing Flowfields

Terry L. Holst*
NASA Ames Research Center, Moffett Field, California
and
Scott D. Thomas*
Informatics General Corporation, Palo Alto, California

A fast, fully implicit, approximate factorization algorithm designed to solve the conservative full-potential equation is used to compute lifting-wing flowfields with embedded transonic flow. The computer program (TWING—transonic wing analysis) uses an elliptic-solver numerical grid-generation routine and is capable of analyzing arbitrary wing geometries with sweep, twist, taper, and section variation. Computed flowfield results for several wing geometries are presented; they demonstrate substantial improvement in convergence speed relative to the FLO28 computer code.

I. Introduction

THE fully implicit AF2, approximate factorization (AF) iteration scheme was first introduced by Ballhaus and Steger[1] for solving the low-frequency (unsteady) transonic small-disturbance equation. Variations of this algorithm have been subsequently applied to the solution of the steady transonic small-disturbance equation for two-dimensional applications[2] and to the steady full-potential equation in both two-dimensional[3-5] and three-dimensional[6-8] applications. In addition, other researchers have recently presented other fully implicit, AF-type algorithms for solving various transonic potential formulations in both two and three dimensions. Some of these algorithms include Sankar et al.,[9] in which the strongly implicit method of Stone[10] was utilized to obtain airfoil and wing solutions, and Jameson,[11] in which a multigrid-enhanced AF scheme was used to obtain airfoil solutions. For all formulations, a significant improvement in convergence speed was obtained relative to the standard transonic relaxation scheme, successive-line overrelaxation (SLOR).

In the present study, experience with the AF2 scheme applied to the conservative full-potential equation for steady transonic wing flowfields is presented. The numerical algorithm used is an extended version of the algorithm presented in Ref. 6. The Ref. 6 algorithm was designed to solve for flowfields about nonlifting wings mounted between parallel walls and as such demonstrated the capability of the AF2 scheme in three-dimensional transonic applications. In the present study this algorithm has been extended to include lift about wings with finite tips. The resulting computer code, called TWING (transonic wing analysis), is capable of handling isolated wing geometries with reasonably general spanwise variations of sweep, taper, twist, and section characteristics. Comparison in terms of both accuracy and convergence speed are made between TWING and the FLO28 (Ref. 12) computer code.

The finite difference spatial-difference scheme used in the present full-potential formulation obtains stability in supersonic regions of flow by adding an artificial viscosity term similar to that introduced in Ref. 13. In the present formulation, however, addition of the artificial viscosity term is achieved by using an upwind bias of the density coefficient. This strategy is significant because it simplifies the technique for including an artificial viscosity term into the residual operator. Other studies[14-16] have used similar steady-state differencing procedures in a wide variety of problems to further substantiate the reliability and flexibility of this differencing procedure.

Presented as Paper 82-0105 at the AIAA 20th Aerospace Sciences Meeting, Orlando, Fla., Jan. 11-14, 1982; submitted Jan. 19, 1982; revision received Aug. 27, 1982. This paper is declared a work of the U.S. Government and therefore is in the public domain.

*Research Scientist. Member AIAA.

II. Measurement of Computational Efficiency

With the advent of many new algorithms designed to improve the computational efficiency of the numerical solution process, the need for quantitatively determining the relative efficiency of each algorithm is at hand. The mere statement that "this algorithm requires only x number of iterations while that algorithm requires y iterations" does not suffice as the needed quantitative unit of measure. The bottom line in determining which code is most efficient from the aerodynamic user's point of view is cost. Although many parameters influence the cost—for example, the amount of storage used, the amount of I/O required, and the local installation charging algorithm—the most important parameter is CPU execution time.

Merely using the CPU execution time is a simple way of assessing the relative efficiency of each computer code. However, there are several other parameters that are also of interest and that address the relative efficiencies of three different areas: 1) the error (or residual) reduction per iteration (r); 2) the CPU time per iteration per grid point (CPU1); and 3) a parameter to measure the mesh topology efficiency (MER). These three parameters are defined as follows:

$$r = \left(|R|^n_{\max} / |R|^1_{\max} \right)^{1/n} \tag{1}$$

$$\text{CPU1} = \text{CPU time (s)}/(n \times \text{NPTS}) \tag{2}$$

$$\text{MER} = \text{NPTS}_{\text{surf}}/(\text{NPTS})^{(L-1)/L} \tag{3}$$

where $|R|^n_{\max}$ is the maximum absolute value residual for the nth iteration; NPTS is the total number of grid points in the finite difference mesh; L is the number of spatial dimensions; and $\text{NPTS}_{\text{surf}}$ is the number of grid points on the aerodynamic surface of interest; for example, for wing-alone calculations $\text{NPTS}_{\text{surf}}$ is the number of grid points on the wing surface.

The r parameter is a measure of the iteration algorithm's ability to reduce the solution residual. A superior definition for this parameter would be to use the solution error instead of residual. This is because the residual operator can give a false indication of convergence. It effectively monitors only the higher frequency errors in the error frequency spectrum (see Refs. 3 and 17 for more discussion on this point).

However, obtaining the error convergence history is not practical; therefore, the easily obtained residual operator is used.

The r parameter is particularly interesting in that it represents an approximation to the algorithm's average spectral radius. Values will generally range from just below 1.0 for explicit or semi-implicit iteration algorithms, indicating very slow convergence, to zero for direct schemes, indicating convergence in one iteration. For nonlinear transonic flow problems no theoretically direct schemes exist. Therefore, a good value of r for transonic problems may be 0.7 to 0.8, which indicates that a reasonable level of convergence could be achieved in about 10 to 20 iterations. One problem associated with using this parameter to make relative comparisons from code to code is that it does not vary linearly with CPU time. In addition, other parameters, such as CPU1 and MER, should be consulted to obtain the ultimate quantitative comparison of computational efficiency.

The next parameter to be discussed is CPU1. This parameter is self-explanatory, being simply the total CPU time in seconds per iteration per grid point; it is a measure of how efficiently a transonic computer code executes independent of the mesh size and the number of iterations required for convergence. This parameter is affected by such things as the computer execution speed, the operating system, compiler options, coding efficiency, and algorithm efficiency. The first three factors vary from computer installation to installation and the last two from code to code. To simplify the situation, comparisons should be performed on the same computer. In the present paper all results have been computed on the Ames CDC 7600 computer with a SCOPE 2.1.4 operating system and an FTN (OPT = 2) compiler.

The last parameter introduced in this section is MER, the mesh efficiency ratio. This parameter measures the efficiency of the mesh topology, that is, the ratio of the number of grid points on the appropriate aerodynamic surface to the number of grid points in an average $(L-1)$-dimensional plane of the same L-dimensional mesh [see Eq. (3)].

Larger values of MER indicate more efficient grid topologies, and smaller values indicate less efficient grids. A value of 1.0 is considered average. This parameter is only a qualitative measure of the grid topology efficiency, because different cell-size stretching rates in the directions away from the wing could cause the use of fewer grid points in these directions and, therefore, somewhat different values of MER for the same topology. Nevertheless, this parameter does focus attention on the fact that not all grid topologies are equal and that significant improvements in computational efficiency can be obtained by using more efficient grids.

Each of the parameters just discussed—r, CPU1, and MER—measures the efficiency of different parts of a transonic relaxation computer code. None of these parameters taken alone describes the complete picture with regard to computational efficiency. However, if all these parameters are considered, then a proper evaluation of code performance is obtained.

III. Numerical Algorithm

Governing Equations

The three-dimensional full-potential equation written in strong conservation-law form is given by

$$(\rho\phi_x)_x + (\rho\phi_y)_y + (\rho\phi_z)_z = 0 \tag{4a}$$

$$\rho = \left[1 - \frac{\gamma-1}{\gamma+1}(\phi_x^2 + \phi_y^2 + \phi_z^2)\right]^{1/(\gamma-1)} \tag{4b}$$

The density ρ and velocity components ϕ_x, ϕ_y, and ϕ_z are nondimensionalized by the stagnation density ρ_s and the critical sound speed a_*, respectively; x, y, and z are Cartesian coordinates in the streamwise, spanwise, and vertical directions, respectively, and γ is the ratio of specific heats.

Equation (4) is transformed from the physical domain (Cartesian coordinates) to the computational domain by using a general independent variable transformation. This transformation (Fig. 1), indicated by

$$\xi = \xi(x,y,z) \qquad \eta = \eta(x,y,z) \qquad \zeta = \zeta(x,y,z) \tag{5}$$

maintains the strong conservation-law form of Eq. (4) (Ref. 18). The full-potential equation written in the computational domain (ξ-η-ζ coordinate system) is given by

$$(\rho U/J)_\xi + (\rho V/J)_\eta + (\rho W/J)_\zeta = 0 \tag{6a}$$

$$\rho = \left[1 - \frac{\gamma-1}{\gamma+1}(U\phi_\xi + V\phi_\eta + W\phi_\zeta)\right]^{1/(\gamma-1)} \tag{6b}$$

where

$$\begin{aligned} U &= A_1\phi_\xi + A_4\phi_\eta + A_5\phi_\zeta \\ V &= A_4\phi_\xi + A_2\phi_\eta + A_6\phi_\zeta \\ W &= A_5\phi_\xi + A_6\phi_\eta + A_3\phi_\zeta \end{aligned} \tag{7a}$$

$$\begin{aligned} A_1 &= \xi_x^2 + \xi_y^2 + \xi_z^2 \qquad & A_4 &= \xi_y\eta_y \\ A_2 &= \eta_y^2 & A_5 &= \xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z \\ A_3 &= \zeta_x^2 + \zeta_y^2 + \zeta_z^2 & A_6 &= \zeta_y\eta_y \end{aligned} \tag{7b}$$

and

$$\begin{aligned} J &= (\xi_x\zeta_z - \xi_z\zeta_x)\eta_y \\ &= 1/(x_\xi z_\zeta - x_\zeta z_\xi)y_\eta \end{aligned} \tag{7c}$$

Note that the metric quantities defined by Eqs. (7b) and (7c) have been simplified somewhat from the most general possible form. The assumption that all η = constant surfaces coincide with y = constant planes has been made. This can be expressed mathematically by

$$y_\zeta = 0 \qquad y_\xi = 0 \tag{8}$$

Use of Eq. (8) permits the simplified treatment of wing geometries and does not affect the generality of the three-dimensional spatial-differencing scheme or the fully implicit AF2 iteration scheme.

Several significant advantages are offered by this very general form. The main advantage is that boundaries associated with the physical domain are transformed to boundaries of the computational domain. This aspect is illustrated in Fig. 1, where the physical and computational domains for a typical transformation are shown. The computational coordinates ξ, η, and ζ are in the wraparound, spanwise, and radial-like directions, respectively. The wing/wing-extension boundary transforms to $\zeta = \zeta_{\max}$, and the outer physical boundary transforms to $\zeta = \zeta_{\min}$. The symmetry-plane boundary transforms to $\eta = \eta_{\min}$, and the freestream sidewall boundary transforms to $\eta = \eta_{\max}$. The last two sides of the computational domain are formed by the upper and lower cuts along the vortex sheet.

Grid Generation

The grid-generation scheme used in the present three-dimensional formulation is a simple extension of the two-dimensional scheme presented in Ref. 4. It is written as a separate program and coupled with the flow solver code via

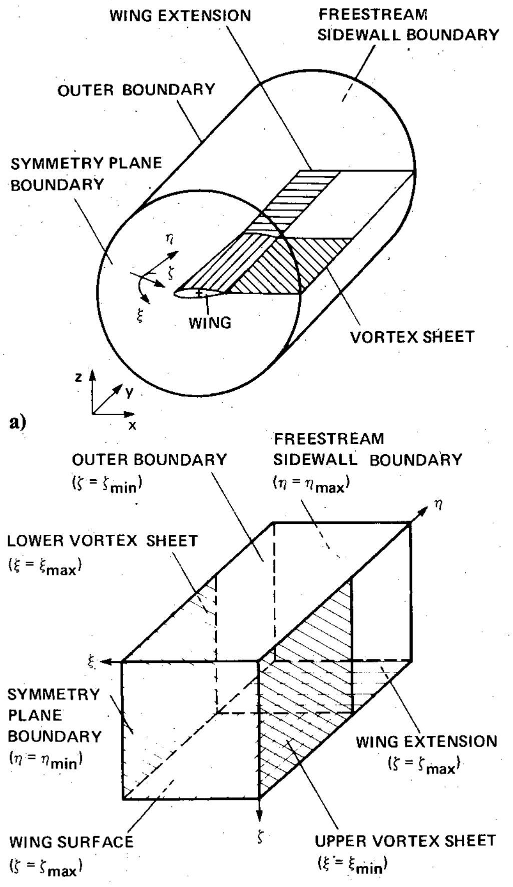
WING EXTENSION
FREESTREAM SIDEWALL BOUNDARY
OUTER BOUNDARY
SYMMETRY PLANE BOUNDARY
WING
VORTEX SHEET
a)
OUTER BOUNDARY ($\zeta=\zeta_{min}$)
FREESTREAM SIDEWALL BOUNDARY ($\eta=\eta_{max}$)
LOWER VORTEX SHEET ($\xi=\xi_{max}$)
SYMMETRY PLANE BOUNDARY ($\eta=\eta_{min}$)
WING EXTENSION ($\zeta=\zeta_{max}$)
WING SURFACE ($\zeta=\zeta_{max}$)
UPPER VORTEX SHEET ($\xi=\xi_{min}$)
b)

Fig. 1 Schematic of general $(x,y,z)\leftrightarrow(\xi,\eta,\zeta)$ transformation: a) physical domain, b) computational domain.

job control language. Thus, one job submittal automatically produces a grid followed by the associated flow solution. The finite difference mesh is generated using a standard two-dimensional algorithm. This requires solution of two Laplace equations

$$\xi_{xx}+\xi_{zz}=0 \qquad \zeta_{xx}+\zeta_{zz}=0 \tag{9}$$

in each spanwise plane used as a defining station. These equations are transformed to (and solved in) the computational domain; that is, ξ and ζ are the independent variables, and x and z are the dependent variables. A fast approximate factorization relaxation algorithm is used to solve the resulting transformed equations.

This establishes values for x and z in each spanwise plane used as a defining station. Coordinate values (x and z) for computational planes between the defining stations are obtained via linear interpolation. A maximum of 25 defining stations is allowed and can be increased easily by increasing program dimensions. For the case of a wing with no taper or section variation, only two defining stations are required, one at the root and one outboard of the tip in the wing-extension region. The root station is user-specified, but the wing-extension station is always chosen as a flat plate. In addition, wing taper, twist, thickness, and sweep variations can be specified at each defining station. The coordinate values in the spanwise direction (y values) are computed from a stretching formula that in its simplest form gives equal spacing over the wing with relatively rapid stretching beyond the tip.

An example grid about the ONERA M6 wing is shown in Figs. 2-5. The grid dimensions used for this case were $89\times25\times18$ (wraparound, spanwise, and radial-like directions, respectively) with 17 spanwise stations on the wing surface. Thus, out of 40,050 computational points, a total of 1513 points were used to define the wing surface. Therefore,

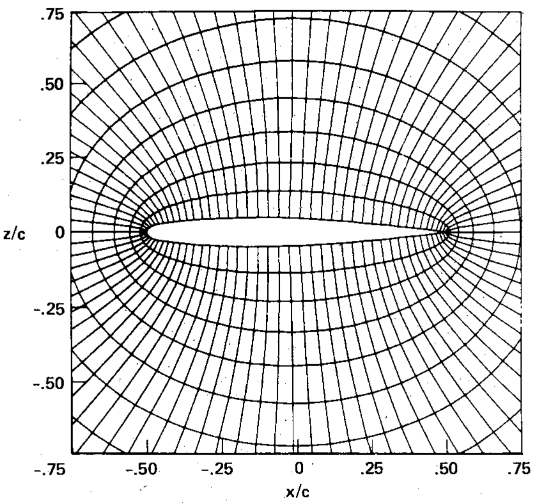

Fig. 2 Numerically generated finite difference mesh about the ONERA M6 wing, station 1 (symmetry plane): $y=0$, 89×18 grid points.

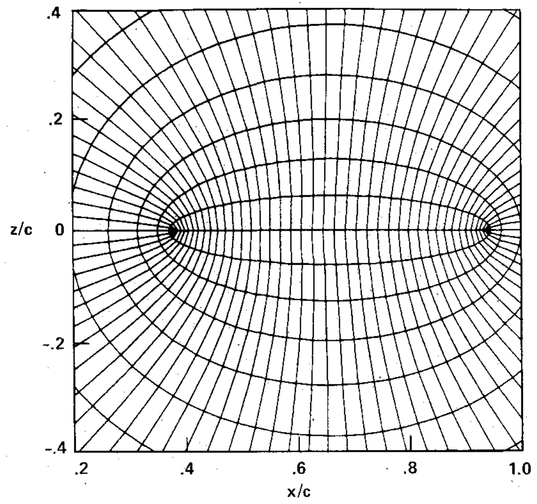

Fig. 3 Numerically generated finite difference mesh about the ONERA M6 wing, station 18 (first plane outboard of the tip): $y=1.53$, 89×18 grid points.

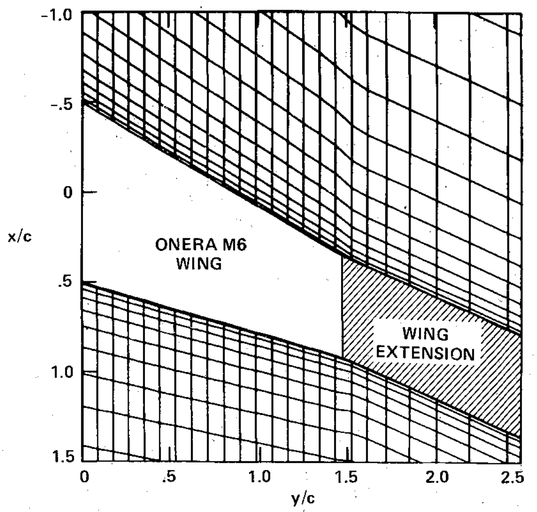

Fig. 4 Three-dimensional grid about the ONERA M6 wing, planform view: $z=0$, 25×18 grid points.

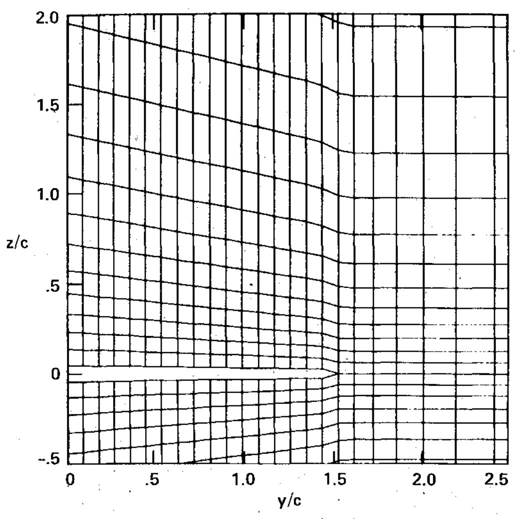

Fig. 5 Three-dimensional grid about the ONERA M6 wing, half-chord view: 25 × 18 grid points.

this mesh has a mesh efficiency ratio (MER) of 1.29, which is somewhat above the nominal value of 1.0, indicating a better than average mesh topology.

For this mesh, three defining stations were used: user-specified stations at the root and tip and the automatic flat-plate section used for all spanwise stations beyond the tip. The outer boundary radius (RADMAX) was set at 6.0 root chords. Generation of the entire three-dimensional mesh, as described, required only about 4 s of computer time on the CDC 7600 computer.

Figures 2 and 3 show local enlargements of the ONERA M6 wing mesh at typical spanwise stations on and off the wing. The mesh is automatically clustered at both the leading and trailing edges in approximately equal amounts. Figures 4 and 5 show additional views of the same mesh. The wing planform including the surrounding mesh ($z=0$ plane) is shown in Fig. 4, and the mesh in the $\xi=$ constant plane corresponding to the wing half-chord position is shown in Fig. 5. Apparent in these views is the wing-tip grid topology and the stretching rate used beyond the tip.

Spatial Differencing

A finite difference approximation for Eq. (4), suitable for both subsonic and supersonic flow regions, is given by

$$\overleftarrow{\delta_\xi}(\tilde{\rho}U/J)_{i+½,j,k}+\overleftarrow{\delta_\eta}(\bar{\rho}V/J)_{i,j+½,k}+\overleftarrow{\delta_\zeta}(\hat{\rho}W/J)_{i,j,k+½}=0 \qquad (10)$$

where the operators $\overleftarrow{\delta_\xi}(\ )$, $\overleftarrow{\delta_\eta}(\ )$, and $\overleftarrow{\delta_\zeta}(\ )$ are first-order-accurate, backward-difference operators in the ξ, η, and ζ directions, respectively. The density coefficients $\tilde{\rho}$, $\bar{\rho}$, and $\hat{\rho}$ are defined by

$$\tilde{\rho}_{i+½,j,k}=[(1-\nu)\rho]_{i+½,j,k}+\nu_{i+½,j,k}(\rho_{i+r+½,j,k})$$

$$\bar{\rho}_{i,j+½,k}=[(1-\nu)\rho]_{i,j+½,k}+\nu_{i,j+½,k}(\rho_{i,j+s+½,k})$$

$$\hat{\rho}_{i,j,k+½}=[(1-\nu)\rho]_{i,j,k+½}+\nu_{i,j,k+½}(\rho_{i,j,k+t+½}) \qquad (11)$$

where

$$r=\pm 1 \text{ when } U_{i+½,j,k}\lesseqgtr 0$$

$$s=\pm 1 \text{ when } V_{i,j+½,k}\lesseqgtr 0$$

$$t=\pm 1 \text{ when } W_{i,j,k+½}\lesseqgtr 0 \qquad (12)$$

$$\nu_{i+½,j,k}=\begin{cases}\max[(M^2_{i,j,k}-1)C,0] & \text{for } U_{i+½,j,k}>0\\ \max[(M^2_{i+1,j,k}-1)C,0] & \text{for } U_{i+½,j,k}<0\end{cases} \qquad (13)$$

The quantity $M_{i,j,k}$ is the local Mach number, C is a user-specified constant (usually between 1.0 and 2.0); and the quantities U, V, and W are the contravariant velocity components computed using standard, second-order-accurate finite difference formulas. The density ρ is computed from the second-order-accurate discretized version of Eq. (6b) and is stored at half points in the finite difference mesh (i.e., at $i+½,j,k$). Values needed at $i,j+½,k$ or $i,j,k+½$ are obtained using simple four-element averages.

Use of the density coefficient given by Eqs. (11) is equivalent to the addition of an appropriately differenced artificial viscosity term.[3,4] This effectively maintains an upwind influence in the differencing scheme for supersonic regions anywhere in the finite difference mesh for any orientation of the velocity vector, thus approximating a rotated differencing scheme. Use of upwinding along all three coordinate directions is not necessary in most cases. As a result, the present version of TWING has included in its spatial-differencing algorithm the ability to choose which coordinate directions will be upwinded and which will not.

The spatial-differencing scheme given by Eqs. (10-13) is centrally differenced and second-order-accurate in subsonic regions. In supersonic regions, the differencing is a combination of 1) the second-order-accurate central differencing used in subsonic regions, and 2) the first-order-accurate upwind differencing resulting from the upwind evaluation of the density. As the flow becomes increasingly supersonic the scheme is increasingly retarded in the upwind direction.

AF2 Iteration Scheme

The AF2 fully implicit iteration scheme used in the present study can be expressed in a three-step format given by
Step 1:

$$\left(\alpha\mp\alpha\beta_\eta\left|\frac{V}{J}\right|_{i,j,k}\overset{\rightleftarrows}{\delta_\eta}-\frac{1}{A_k}\overrightarrow{\delta_\eta}A_j\overleftarrow{\delta_\eta}\right)g^n_{i,j}$$

$$=\alpha\omega L\phi^n_{i,j,k}+\alpha A_{k+1}f^n_{i,j,k+1} \qquad (14a)$$

Step 2:

$$\left(A_k\mp\beta_\xi\overset{\rightleftarrows}{\delta_\xi}-\frac{1}{\alpha}\overrightarrow{\delta_\xi}A_i\overleftarrow{\delta_\xi}\right)f^n_{i,j,k}=g^n_{i,j} \qquad (14b)$$

Step 3:

$$(\alpha+\overleftarrow{\delta_\zeta})C^n_{i,j,k}=f^n_{i,j,k} \qquad (14c)$$

where the n superscript is an iteration index; α is an acceleration parameter; $L\phi^n_{i,j,k}$ is the nth iteration residual operator [defined by Eq. (10)]; ω is a relaxation factor, equal to 1.8 for all cases presented; $g^n_{i,j}$ is an intermediate result stored at each grid point in a given k plane, that is, g requires only a two-dimensional array of storage; and $f^n_{i,j,k}$ is an intermediate result stored at each point in the finite difference mesh. The A_i, A_j, and A_k coefficients are defined by

$$A_i=(\tilde{\rho}A_1/J)^n_{i-½,j,k}$$

$$A_j=(\bar{\rho}A_2/J)^n_{i,j-½,k}$$

$$A_k=(\hat{\rho}A_3/J)^n_{i,j,k-½} \qquad (15)$$

and the density coefficients $\tilde{\rho}$, $\bar{\rho}$, and $\hat{\rho}$ are defined by Eq. (11).

In step 1, the g array is obtained by solving a tridiagonal matrix equation for each $\xi=$constant line in the kth plane. In step 2, the f array is obtained from g by solving a tridiagonal matrix equation for each $\eta=$constant line, again for just the kth plane. Next, step 1 is used to obtain the g array for the

$k-1$ plane, and then step 2 is used to obtain the f array for the $k-1$ plane, etc. This process continues until all values of f in the three-dimensional mesh are established. Then, by using step 3, the correction array

$$C^n_{i,j,k} = \phi^{n+1}_{i,j,k} - \phi^n_{i,j,k}$$

is obtained from the f array by solving a simple bidiagonal matrix equation for each ζ line in the entire finite difference mesh. The nature of the AF2 factorization places a sweep-direction restriction on the step 1-2 combination and on step 3. The step 1-2 combination must be swept in the direction of the decreasing k subscript; that is, from the wing boundary toward the outer boundary (see Fig. 1). The step 3 sweep must proceed in just the opposite direction; that is, from the outer boundary toward the wing. There are no sweep-direction limitations placed on any of the three sweeps due to flow direction.

Initiation of step 1 at the wing boundary ($k=\text{NK}$) requires knowledge of f at NK+1, which is generally unobtainable. A simple solution is to set f at NK+1 equal to zero. The $f_\eta=0$ boundary condition, which is used in Ref. 4 for two-dimensional problems, is not used here because the present AF2 factorization does not easily allow such a boundary condition on f. Because the present iteration scheme is written in the correction form, f must approach zero as the solution converges. The $f=0$ boundary condition is therefore consistent with the steady-state solution. Additional boundary conditions for g are required at $\eta=\eta_{\min}$ (symmetry plane boundary) and $\eta=\eta_{\max}$ (freestream sidewall boundary). These conditions are implemented by imposing

$$(g_\eta)_{i,1}=0 \quad \text{and} \quad g_{i,\text{NJ}}=0$$

The quantity α appearing in Eq. (14) can be considered as Δt^{-1}. Best convergence characteristics are obtained when α is cycled over a sequence of values. The small values are particularly effective for reducing the low-frequency errors, and the large values are particularly effective for reducing the high-frequency errors. The α sequence given in Ref. 3 has been used for the three-dimensional cases presented herein.

With this version of AF2 the ζ-direction difference operator is split between two steps. This generates a $\phi_{\zeta t}$-type term which can be useful to the iteration process as time-like dissipation. (See Refs. 3 and 4 for a more detailed discussion of this aspect. A comparison with the ADI factorization is also given in Ref. 3.) In addition, $\phi_{\eta t}$- and $\phi_{\xi t}$-type terms have been added inside the brackets of steps 1 and 2, respectively [see Eqs. (14a) and (14b)]. The parameters β_ξ and β_η are specified by the user to maintain stable convergence. For all cases presented herein β_η was set to zero and β_ξ was set between 0.05 and 0.1 in supersonic regions and to zero in subsonic regions. The larger values of β_ξ are required for larger regions of supersonic flow. The double-arrow notation on the δ_ξ- and δ_η-difference operators in Eqs. (14a) and (14b) indicates that the difference direction is always upwind. The sign in front of each operator is chosen so as to increase the magnitude of the diagonal term in the first and second tridiagonal matrix sweeps.

Boundary Conditions

The wing-surface boundary condition is that of flow tangency (i.e., no flow through the wing surface), and requires that the ζ contravariant velocity component at the wing surface be zero ($W=0$). This boundary condition is implemented by a reflection condition. In other expressions, where ϕ_ζ is required at the wing surface, the $W=0$ boundary condition is used again to obtain

$$\phi_\zeta|_{\text{wing}} = -(A_5/A_3)\phi_\xi - (A_6/A_3)\phi_\eta \tag{16}$$

Thus, a value of ϕ_ζ at the wing surface can be obtained without using a one-sided difference on ϕ. Along the symmetry plane boundary, flow tangency is also specified ($V=0$).

Along the freestream sidewall and outer boundaries the initial freestream distribution of ϕ is held fixed for nonlifting calculations. For lifting calculations, the outer boundary ϕ distribution is updated by the usual compressible vortex solution with circulation strength Γ_j. At the end of each iteration the circulation is recomputed at each span station from the trailing-edge velocity potential jump

$$\tilde{\Gamma}^n_j = (\phi^n_{U_{\text{TE}}} - \phi^n_{L_{\text{TE}}})_j \tag{17}$$

and then either overrelaxed or underrelaxed by using

$$\Gamma^n_j = \text{RGAM}\tilde{\Gamma}^n_j + (1-\text{RGAM})\Gamma^{n-1}_j \tag{18}$$

After the nth level circulation is updated, the residual calculation is performed. In the residual calculation, the jump in velocity potential along the vortex sheet is explicitly included. When differencing the correction across the vortex sheet, the jump condition becomes

$$\Gamma^{n+1}_j - \Gamma^n_j = (C^n_U - C^n_L)_j \tag{19}$$

which is difficult to impose since Γ^{n+1}_j is unknown. An alternative is to estimate the value of Γ^{n+1}_j from

$$\overline{\Gamma^{n+1}_j} = 2\Gamma^n_j - \Gamma^{n-1}_j \tag{20}$$

and then to use the estimated value in Eq. (19). Unfortunately, this lift calculation procedure had difficulty converging because the Γ^n_j and $\overline{\Gamma^{n+1}_j}$ quantities oscillated 180 deg out of phase. This problem was removed by modifying the Γ^n_j calculation

$$\Gamma^n_j|_{\text{modified}} = \tfrac{1}{2}(\Gamma^n_j + \overline{\Gamma^n_j}) \tag{21}$$

where $\overline{\Gamma^n_j}$ is the extrapolated value of $\overline{\Gamma^{n+1}_j}$ from the previous iteration.

The RGAM parameter used in Eq. (18) is user specified. Best performance is achieved with a relatively large value of RGAM (e.g., 1.5) for the first several iterations (e.g., $1 \le n \le 12$) and then a smaller value (e.g., 0.9) for $n>12$. This has the effect of establishing the circulation distribution very rapidly.

The spatial-differencing procedure used outboard of the tip on the wing extension and downstream of the trailing edge on the vortex sheet is virtually identical to that used in the flowfield interior. This is achieved in the ζ direction at the wing-extension surface by using difference formulas which utilize both sides of the wing-extension. Across the vortex sheet ξ-difference formulas can be constructed by assuming ξ to be a periodic coordinate. Difficulties along the wing-extension leading- and trailing-edge mapping singularities have been alleviated by using an average of residual values adjacent to each mapping singularity instead of the standard finite difference formula. In addition, some averaging of the density is used in these locations to ensure smoothness of the resulting solution.

IV. Computed Results

The TWING computer code discussed in the previous section is evaluated in this section by presenting a range of computed examples. The first of two cases presented is basically two-dimensional in nature and is compared with well-established two-dimensional results. For an additional degree of comparison, results from the three-dimensional FLO28 computer code[12] (wing-alone mode) are included. The FLO28 code solves the conservative full potential equation using a finite-volume spatial-difference scheme and the SLOR iteration scheme. This formulation, like that of TWING, is second-order accurate in subsonic regions of flow and first-order accurate in supersonic regions. For these two-dimensional comparisons TWING and FLO28 were run with zero sweep, no taper, and with a large aspect ratio ($\mathcal{R}=40$). The mesh dimensions used for each three-dimensional code were the same as those for a standard three-dimensional calculation. The TWING results were obtained with a mesh of $89\times25\times18=40{,}050$ points (wraparound, spanwise, and radial-like directions, respectively) with $89\times17=1513$ points on the wing surface. A typical finite difference grid used by TWING with these dimensions has already been presented and discussed (Figs. 2-5).

All FLO28 results have been computed on a mesh of $120\times16\times28=53{,}760$ points (wraparound, normal, and spanwise directions, respectively), with $75\times18=1350$ points on the wing surface. This is the finest mesh available for this version of FLO28 (called FLO28L), because FLO28 is core-contained on the CDC 7600 computer. That is, the disk I/O associated with the standard version of FLO28 has been removed to improve the code's execution efficiency. This change incurs the sacrifice that fine mesh cases are no longer allowed. The TWING code uses disk I/O and, therefore, is not restricted to such a coarse mesh.

The TWING cross-sectional grid uses the "O" mesh topology, and the FLO28 cross-sectional grid is of the "C" mesh variety. This produces a somewhat less efficient grid for FLO28 (MER = 0.95) compared with TWING (MER = 1.29) and is the basic reason for comparing solutions computed on different size meshes.

The two-dimensional result presented in Fig. 6 shows a comparison of pressure coefficient distributions from TWING, FLO28, TAIR (Ref. 19), and FLO6 (Ref. 13). The latter two codes solve the conservative full-potential equation in two dimensions for airfoil geometries. This lifting, transonic calculation was computed for an NACA 0012 airfoil at $M_\infty=0.75$ and $\alpha=1$ deg. The TWING, FLO6, and TAIR results are all in very good agreement even at the shock, which is somewhat surprising since the TWING cross-sectional mesh is relatively coarse. (The FLO6 results were computed with a 148×32 mesh and the TAIR results with a 149×30 mesh.) The apparent reason TWING produces such a good shock capture on such a coarse mesh is as follows. First, the spatial differencing algorithm used by TWING computes and stores the density at $i+\frac{1}{2},j,k$ which tends to favor shocks that are mesh-fit along $\xi=$ constant surfaces. Since in this case, the shock is almost exactly mesh-fit by $\xi=$ constant surfaces, a nearly optimal shock capture is realized. For many three-dimensional calculations in which the shock is approximately swept with the leading and trailing edges of the wing, this nearly optimal shock-capture property will produce improved shock-capture results.

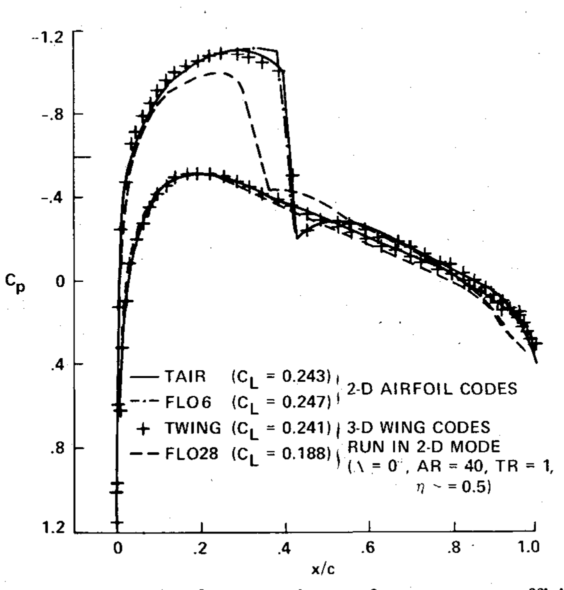

Fig. 6 Two-dimensional comparison of pressure coefficient distributions: NACA 0012 airfoil, $M_\infty=0.75$, $\alpha=1$ deg.

The FLO28 result of Fig. 6 underpredicts the lift by about 23%; therefore, it produces a weaker shock almost 10% of chord upstream of the other results. This particular FLO28 calculation was produced with a coarse-medium-fine mesh sequence with 50, 50, and 200 iterations, respectively. A second calculation with double the number of fine-mesh iterations produced no significant change in the solution. The possibility of tip effects existing at the midspan location was studied by running the same solution for several $\mathcal{R}$ values. No significant change in the midspan solution was found for the $\mathcal{R}=40$ solution relative to solutions with higher $\mathcal{R}$.

One possible explanation for the underprediction of lift displayed by FLO28 is the trailing-edge mesh coarseness. The Kutta condition, which determines the amount of circulation required to match pressure at the trailing edge, is satisfied in both TWING and FLO28 at the trailing edge. The TWING trailing-edge mesh spacing is 0.2% of chord; this compares with 5.3% for FLO28. Use of such a locally coarse mesh in the vicinity of the trailing edge may produce (or contribute to) the type of inaccuracy causing the observed underprediction of lift.

The FLO28 shock thickness displayed in Fig. 6 is about 7 or 8% of chord; this compares with about 3 or 4% for TWING. Even though the FLO28 mesh had only 75 points on the wing surface and TWING had 89, the mesh spacing at the FLO28 shock is *finer* than the corresponding TWING shock mesh spacing (3.1% of chord for FLO28 and 3.4% for TWING). Thus, the outstanding shock-capture capability associated with the TWING spatial-differencing scheme is again emphasized.

The second case considered involves the ONERA M6 wing at $M_\infty=0.84$ and $\alpha=3.06$ deg. The leading-edge sweep angle is 30 deg, the taper ratio is 0.5624, and the aspect ratio is 3.8 (see Ref. 20 for more details about the geometry). The finite difference grid used by TWING for this calculation has already been presented in Figs. 2-5. Basically, it consists of $89\times25\times18=40{,}050$ points with $89\times17=1513$ points on the wing surface. The results of this calculation are compared in Fig. 7 with the experimental results of Ref. 20 and with numerical results computed from FLO28 (wing-alone mode). The FLO28 results were computed on a mesh with the same dimensions used in the previous two-dimensional comparisons, that is, $120\times16\times28=53{,}760$ points with $75\times18=1350$ points on the wing surface.

Figure 7 shows a comparison of pressure coefficient distributions for four different wing stations ($\eta=0.20$, 0.44, 0.65, and 0.90). This particular solution has a double shock structure, including a supersonic-to-supersonic oblique shock swept approximately parallel to the wing leading edge, followed downstream by a nearly normal shock. The two shocks merge and form a single shock near the wing tip. The TWING shock-capture quality for the supersonic-to-supersonic shock is reasonably good at $\eta=0.20$ but degrades considerably for $\eta=0.44$ and 0.65. Although the FLO28 supersonic-to-supersonic shock capture is only fair, it is somewhat better than that of the TWING solution. The primary reason for this is the relatively large amount of grid clustering used near the leading edge for the FLO28 mesh. For instance, at 10% of chord, the surface streamwise mesh spacing in the present FLO28 mesh is about 1.3% of chord and that of the TWING mesh is 2.0%. The supersonic-to-supersonic shock capture for both solutions would be improved with more mesh points.

The downstream, nearly normal shock comparison requires thoughtful consideration. The FLO28 results at $\eta=0.20$ and 0.44 are in very good agreement with experiment whereas the TWING results predict a somewhat stronger shock 6-8% of chord farther downstream. Generally speaking, conservative

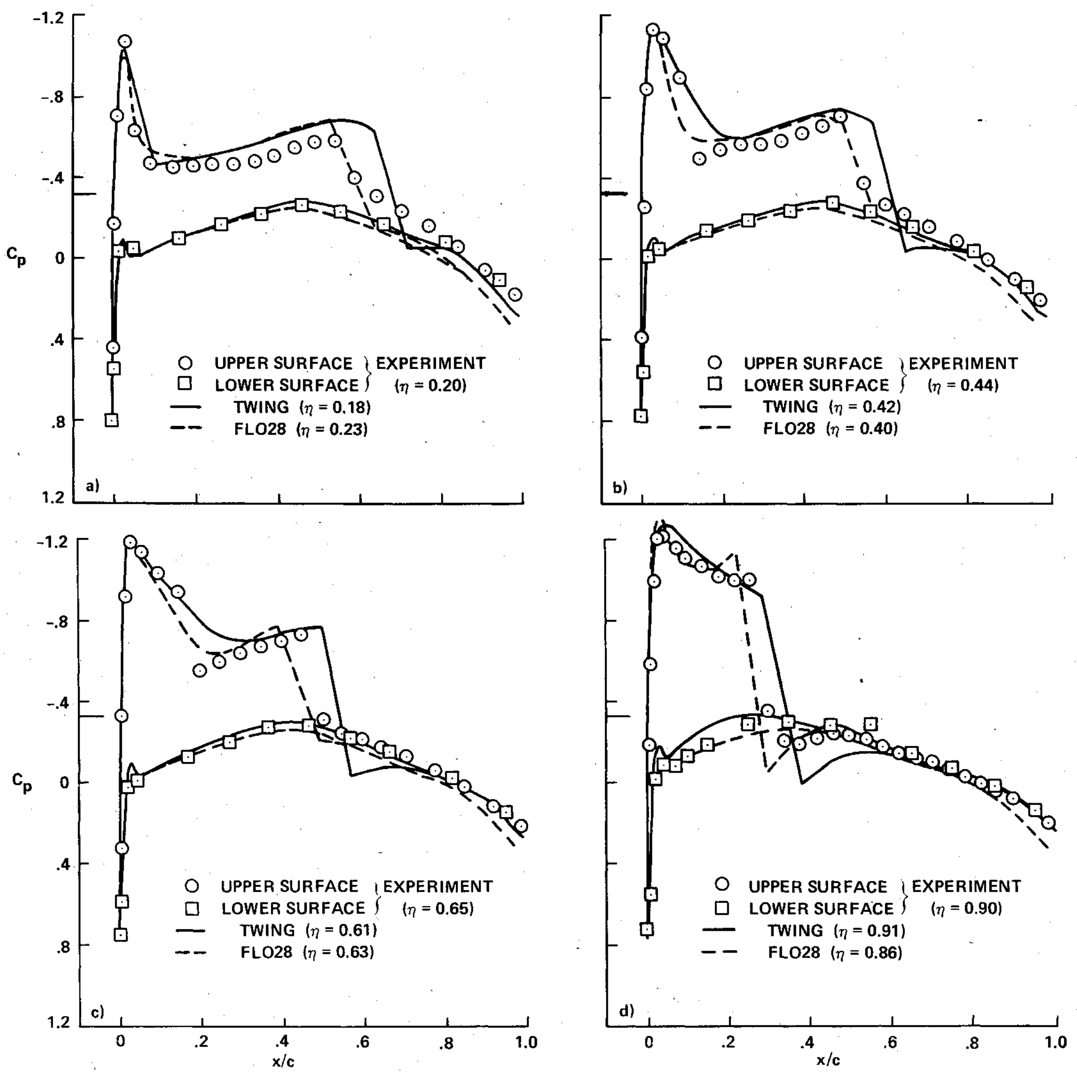

Fig. 7 Pressure coefficient comparisons: ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06$ deg. a) $\eta \cong 0.20$; b) $\eta \cong 0.44$; c) $\eta \cong 0.65$; d) $\eta = 0.90$.

inviscid results overpredict experimental shock strengths and produce shock positions downstream of the experimental values. Of course, the magnitude of these discrepancies is greatly influenced by viscous effects, for example, the shock/boundary-layer interaction. This fact, and the previous finding regarding the tendency of FLO28 to underpredict lift, leads to the conclusion that the overprediction of shock strength and position offered by the TWING solution is correct (in trend) and that the "good" agreement in shock position displayed by the FLO28 solution is fortuitous. This hypothesis is further substantiated by the agreement shown in the comparisons at $\eta = 0.65$ and 0.90. At these locations the FLO28 aft shock position is somewhat upstream of the experiment; however, the TWING shock is still slightly downstream, leaving some room for viscous correction.

The convergence properties of TWING established from the ONERA M6 wing calculation just discussed, are examined in Fig. 8 and Table 1. The rate of development of the pressure coefficient distribution at one selected span station ($\eta = 0.18$) is presented in Fig. 8. The solution after 20 iterations (54 s of CPU time) is compared with the fully converged solution after 80 iterations. Very little disagreement exists, except at the shock, where about six or eight points have not quite converged to their final steady-state positions.

A TWING/FLO28 comparison of various solution parameters including convergence information from the ONERA M6 wing calculation just discussed is presented in Table 1. The convergence parameters from both codes have been approximately optimized by a trial-and-error process. The TWING results are based on an 83-iteration run in which the center-span lift changed by 0.03% in the last 30 iterations. The FLO28 results are based on a coarse-medium-fine mesh sequence with 50, 50, and 283 iterations, respectively. The total lift for this run changed by 0.49% in the last 100 iterations of the fine mesh. Immediately obvious is that TWING sets up the lift much faster than FLO28—8.3 times faster for 95% of the lift and 11.6 times faster for 98% of the lift. In addition, the average residual reduction per iteration, r [see Eq. (1)] is much larger for TWING (0.918) than for FLO28 (0.985). Actually, this comparison is somewhat misleading, as it always is, when the CPU time required per iteration is not considered. For the present case, each TWING iteration required only 2.7 s, whereas each FLO28 iteration required 6.0 s. Therefore, $r = 0.918$ means that the average residual reduction with TWING every 2.7 s is 0.918. Based on a 6.0-s unit of time, the effective TWING residual reduction would be $r_{\text{eff}} = 0.827$!

The reason for the large difference in CPU time per iteration between TWING and FLO28 is not clear. Of course, one explanation for part of the difference is in the mesh-size differences between the two calculations. Perhaps, another reason is associated with the residual operator section of code (spatial-differencing scheme) where most of either code's CPU time is spent. The finite-volume scheme presented in Ref. 12 seems to be somewhat more complicated and, therefore, more costly in execution than the finite difference spatial-differencing scheme presently used in TWING. In addition, the FLO28 code recomputes metrics every iteration

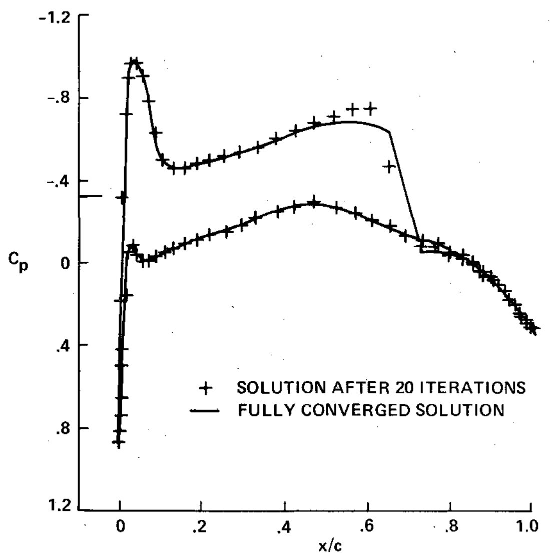

Fig. 8 Comparison of partially and fully converged pressure coefficient distributions at $\eta = 0.18$: ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06$ deg.

Table 1 Comparison of pertinent TWING and FLO28 code statistics (ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06$ deg)

Parameter	FLO28	TWING	FLO28/ TWING ratio
Grid	$120 \times 16 \times 28$ (53,760)	$89 \times 25 \times 18$ (40,050)	1.34
Surface grid	75×18 (1350)	89×17 (1513)	0.89
r [see Eq. (1)]	0.985	0.918	—
CPU1 [see Eq. (2)]	1.11×10^{-4}	0.672×10^{-4}	1.66
MER [see Eq. (3)]	0.95	1.29	0.74
CPU time required for 95% lift, s	467	56	8.3
CPU time required for 98% lift, s	742	64	11.6

to save storage while the TWING code reads the already-computed metrics from disk during each iteration. The disk I/O may be cheaper in terms of CPU time than the operations associated with metric recomputation. At any rate, the number of extra operations associated with the AF2 iteration scheme (TWING), because of three sweeps relative to one sweep for the SLOR scheme (FLO28), is small and could easily be overshadowed by these other factors.

V. Conclusions

Results from a revised transonic-wing analysis computer code (TWING) have been presented and compared with independent numerical and experimental results. The present algorithm uses a fully implicit approximate factorization iteration scheme to solve the full-potential equation in conservative form and produces about an order-of-magnitude improvement in computational efficiency over a standard successive-line overrelaxation iteration scheme. TWING results computed in the two-dimensional mode, that is, with no sweep, taper, or twist and with a very large aspect ratio, agree well with other two-dimensional conservative inviscid results. The TWING shock (strength and position) is generally stronger and exists farther downstream than experimental shock results; therefore, it produces the expected trend for conservative inviscid results when compared with experiment. In addition, the finite difference spatial-difference scheme used in TWING produces a sharper shock profile (for supersonic-to-subsonic shocks) than for several other numerical results presented.

References

[1]Ballhaus, W. F. and Steger, J. L., "Implicit Approximate Factorization Schemes for the Low-Frequency Transonic Equation," NASA TM X-73,082, 1975.

[2]Ballhaus, W. F., Jameson, A., and Albert, J., "Implicit Approximate Factorization Schemes for the Efficient Solution of Steady Transonic Flow Problems," *AIAA Journal,* Vol. 16, June 1978, pp. 573-579.

[3]Holst, T. L. and Ballhaus, W. F., "Fast Conservative Schemes for the Full Potential Equation Applied to Transonic Flows," NASA TM-78469, 1978; also *AIAA Journal,* Vol. 17, Feb. 1979, pp. 145-152.

[4]Holst, T. L., "An Implicit Algorithm for the Conservative, Transonic Full Potential Equation Using an Arbitrary Mesh," AIAA Paper 78-1113, July 1978; also *AIAA Journal,* Vol. 17, Oct. 1979, pp. 1038-1045.

[5]Baker, T. J., "Potential Flow Calculation by the Approximate Factorization Method," *Journal of Computational Physics,* Vol. 42, July 1981, pp. 1-19.

[6]Holst, T. L., "Fast, Conservative Algorithm for Solving the Transonic Full-Potential Equation," AIAA Paper 79-1456, July 1979; also *AIAA Journal,* Vol. 18, Dec. 1980, pp. 1431-1439.

[7]Baker, T. J. and Forsey, C. R., "A Fast Algorithm for the Calculation of Transonic Flow Over Wing/Body Combinations," *Proceedings of the 5th AIAA Computational Fluid Dynamics Conference,* Palo Alto, Calif., June 1981, pp. 189-198.

[8]Benek, J., Steinhoff, J., and Jameson, A., "Application of Approximate Factorization to Three-Dimensional Transonic Potential Flow Calculations," Presented at the 5th AIAA Computational Fluid Dynamics Conference, Palo Alto, Calif., June 1981.

[9]Sankar, N. L., Malone, J. B., and Tassa, Y., "A Strongly Implicit Procedure for Steady Three-Dimensional Transonic Potential Flow," AIAA Paper 81-0385, Jan. 1981; also *AIAA Journal,* Vol. 20, May 1982, pp. 598-605.

[10]Stone, H. L., "Iterative Solution of Implicit Approximations of Multi-Dimensional Partial Differential Equations," *SIAM Journal of Numerical Analysis,* Vol. 5, No. 3, 1968, pp. 530-558.

[11]Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method," *Proceedings of the 4th AIAA Computational Fluid Dynamics Conference,* Williamsburg, Va., July 1979, pp. 122-146.

[12]Caughey, D. A. and Jameson, A., "Numerical Calculation of Transonic Potential Flow about Wing-Body Combinations," *AIAA Journal,* Vol. 17, Feb. 1979, pp. 175-181.

[13]Jameson, A., "Transonic Potential Flow Calculations Using Conservative Form," *AIAA Second Computational Fluid Dynamics Conference Proceedings,* June 1975, pp. 148-155.

[14]Hafez, M. M., Murman, E. M., and South, J. C., "Artificial Compressibility Methods for Numerical Solution of Transonic Full Potential Equation," AIAA Paper 78-1148, July 1978; also *AIAA Journal,* Vol. 17, Aug. 1979, pp. 838-844.

[15]Eberle, A., "A Finite Volume Method for Calculating Transonic Potential Flow Around Wings from the Pressure Minimum Integral," Tech. Translation, NASA TM-75,324, 1978.

[16]Eberle, A., "Transonic Potential Flow Computations by Finite Elements: Airfoil and Wing Analysis, Airfoil Optimization," Lecture presented at the DGLR/GARTEUR 6 Symposium, Transonic Configurations, Bad Harzburg, Germany, June 1978.

[17]Ballhaus, W. F., "A Fast Implicit Solution Procedure for Transonic Flows," *Lecture Notes in Physics, Computing Methods in Applied Sciences and Engineering,* Vol. 91, Springer-Verlag, New York, 1977, pp. 90-102.

[18]Steger, J. L., "Implicit Finite-Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils," AIAA Paper 77-665, June 1977; also *AIAA Journal,* Vol. 16, July 1978, pp. 679-686.

[19]Dougherty, F. C., Holst, T. L., Gundy, K. L., and Thomas, S. D., "TAIR—A Transonic Airfoil Analysis Computer Code," NASA TM-81,296, May 1981.

[20]Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," AGARD Report AR-138, May 1979.