AIAA 81-0385R

A Strongly Implicit Procedure for Steady Three-Dimensional Transonic Potential Flows

N. L. Sankar,* J. B. Malone,† and Y. Tassa‡
Lockheed-Georgia Company, Marietta, Ga.

A rapid relaxation algorithm for the numerical solution of the conservative full potential equation is described and used to predict transonic potential flow past thick swept wings. A sheared parabolic coordinate system is used to simplify the treatment of boundary conditions and an upwind bias is applied to the density in supersonic regions. A unique feature of the present work is the extension of a strongly implicit procedure, that has, in the past, been applied only to elliptic equations, to the present elliptic-hyperbolic problem. Numerical results are presented to demonstrate that the present method is considerably faster than existing line relaxation procedures.

Introduction

IN the field of computational aerodynamics, much of the current effort is being directed toward finding fast and inexpensive methods for solving the transonic potential flow problem.

For two-dimensional, steady, transonic potential flows, considerable progress has already been made. A variety of techniques such as the AF2 and ADI procedures,[1] the fast Poisson solver algorithms,[2] the multigrid techniques,[3,4] and the strongly implicit procedure,[5] have already been developed and implemented into efficient computer codes.

In three-dimensions, until recently, the successive-line-overrelaxation (SLOR) procedure has been the only procedure to be successfully and efficiently applied to transonic potential flows.

Because a typical three-dimensional problem involves well over 10,000 field points, it is obvious that even a small improvement over the performance that is already achieved by SLOR, will translate into a large reduction in computer time and expenditure. Motivated by this possibility, research workers have recently been investigating the application of implicit procedures such as AF and AF2 algorithms to three-dimensional flows.[6,7] Clearly, other implicit procedures, in addition to AF and AF2 procedures, must be looked into, because of the large potential savings involved.

It is with this purpose, that the strongly implicit procedure (SIP) is studied as a candidate for efficient solution of three-dimensional potential flows. The SIP was chosen as a candidate here because it has been shown in the past to be highly competitive with, if not superior to, AF procedures. Stone[8] first proposed SIP as a solution procedure for multidimensional elliptic partial differential equations. Since that time, it has been demonstrated by a number of research workers that SIP can be applied to parabolic equations such as the unsteady Navier-Stokes equations,[9] as well as to hyperbolic equations in time,[5] and that it performs well even when highly stretched and distorted grids are being used. It is therefore of interest to know whether SIP can be applied as an iterative procedure to three-dimensional steady transonic potential flow efficiently, and whether it may be extended later to treat unsteady potential flows. We address the first task—development of an iterative procedure—in this work.

Presented as Paper 81-0385 at the AIAA 19th Aerospace Sciences Meeting, St. Louis, Mo., Jan. 12-15, 1981; submitted Feb. 23, 1981; revision received Aug. 3, 1981.

*Scientist Associate. Member AIAA.
†Scientist. Member AIAA.
‡Senior Research Scientist. Member AIAA.

In the following sections, we compare the AF and SIP procedures, by applying them to a model problem partially simulating the steady transonic potential flow. Both the advantages and disadvantages of SIP over AF are presented. Subsequently, a numerical procedure is developed for solving three-dimensional transonic potential flow iteratively. Numerical results are presented to establish the reliability and accuracy of the procedure. Specific conclusions are drawn about the convergence speed, computational cost, and memory requirements of the SIP. Some general observations are also made about relevant issues such as vectorization of the SIP.

Relaxation Algorithm

In order to construct an iterative scheme for the solution of the nonlinear mixed elliptic-hyperbolic partial differential equation that arises in transonic potential flow, we first look at the following model problem

$$(A_1\phi_\xi)_\xi + (A_2\phi_\eta)_\eta + (A_3\phi_\zeta)_\zeta = -q \tag{1}$$

A_1, A_2, and A_3 are spatially varying coefficients and may depend on ϕ. For the present purposes we will ignore this dependence on ϕ and assume these coefficients to be known. The function q is a source term which may have a weak dependence on ϕ. We will assume q to be known also.

In this section we will first restate some of the alternating direction implicit (ADI) procedures available in open literature for solution of Eq. (1), and compare these procedures to SIP. The ADI procedures are chosen here for comparison because they are the most efficient procedures currently available in the open literature for solution of steady and unsteady transonic potential flow problems.[1,2,10]

For the solution of Eq. (1), following Jameson,[4] a generalized ADI procedure may be written as

$$\left(L_1 - \frac{\partial}{\partial\xi}A_1\frac{\partial}{\partial\xi}\right)(L_1^{-1})\left(L_1 - \frac{\partial}{\partial\eta}A_2\frac{\partial}{\partial\eta}\right)$$

$$\times (L_1^{-1})\left(L_1 - \frac{\partial}{\partial\zeta}A_3\frac{\partial}{\partial\zeta}\right)C^{n+1} = \omega R(\phi^n) \tag{2}$$

Here L_1 is a generalized operator given by

$$L_1 = \alpha_0 + \alpha_1\partial_\xi^- + \alpha_2\partial_\eta^- + \alpha_3\partial_\zeta^-$$

The negative sign implies that these derivatives are replaced by windward differences in numerical computations. C^{n+1} is the difference $(\phi^{n+1} - \phi^n)$ in the value of function ϕ between

iterations n and $(n+1)$; $R(\phi^n)$ is the residual of Eq. (1) evaluated at the nth iteration; ω, α_0, α_1, α_2, and α_3 are user specified quantities.

The AF2 procedure, developed by Holst for three-dimensional transonic potential flows,[6] deviates from Eq. (2) in some details. Nevertheless, the following observations on ADI are true also for the procedure developed by Holst. For two-dimensional flows, the AF2 procedure used by Holst[1] is similar to Eq. (2) if the operator $[(\partial/\partial\zeta)A_3(\partial/\partial\zeta)]$ and all other $(\partial/\partial\zeta)$ operators are set to zero.

1) In ADI procedures the coefficients of the factored Eq. (2) are known a priori and are in fact linear functions of the coefficients of the unfactored equation. No additional storage is needed other than originally used to store the coefficients A_1, A_2, A_3 and quantities ϕ and $R(\phi^n)$.

2) The parameters ω, α_0, α_1, α_2, and α_3 are usually varied between iterations cyclically. They are not spatially varied in practice. That is, no "matrix conditioning" is attempted to give well conditioned equations both in regions where A_1, A_2, and A_3 are large, and in regions where these coefficients are small.

3) In the case of Eq. (2), the cyclic variation of parameters is useful in suppressing all the Fourier components of the error C^{n+1}. If optimal parameters are chosen, it may be analytically estimated that a converged solution is obtained in $O(N\ell nN)$ operations where N is the total number of interior nodes at which ϕ is unknown. The number of operations per iteration are of order $O(N)$. Specifically, once the coefficients A_1, A_2, A_3, and $R(\phi^n)$ are known, three calls to the Thomas tridiagonal algorithm require 15 arithmetic operations per point per iteration.

4) If the coefficients A_1, A_2, and A_3 vary widely, as they will in transonic flow problems solved on a highly stretched grid, an optimal sequence of parameters is difficult to estimate. Thus, the ADI procedure in such situations will require a large number of iterations to suppress all the error components.

5) If the coefficients A_1, A_2, A_3 are not stored but recomputed due to storage limitations, the ADI procedures will require 2 calls to the metric routine. Note that during the first sweep the residual may also be computed. [The AF2 algorithm by Holst[6] requires numerical evaluation of metrics only once, and is therefore more efficient than the algorithm given by Eq. (2).]

With this background on ADI, we turn our attention to the strongly implicit procedure, due to Stone.[8]

SIP Procedure

Like ADI, the SIP is also an approximate factorization procedure. It is applicable equally well as a relaxation solution procedure or as a time marching procedure. In the past, SIP has been applied only to elliptic problems and to parabolic equations in time. But there are no inherent limitations in extending SIP to mixed elliptic-hyperbolic problems and to hyperbolic equations in time.

SIP differs from ADI procedures in that it operates on difference equations rather than on differential operators. The model equation being studied may be discretized in space using central differences for all derivatives. One obtains the following difference equation for model Eq. (1):

$$(E_1+Z_1E_\zeta^-+B_1E_\eta^-+D_1E_\xi^-+F_1E_\xi^++H_1E_\eta^++S_1E_\zeta^+)C^{n+1} = -R_1(\phi^n)-q \tag{3}$$

where $E_\zeta^\pm$, etc., are the shift operators. For example,

$$E_\zeta^\pm C_{ijk}^{n+1} = C_{ijk\pm1}^{n+1}$$

$R_1(\phi^n)$ is the residual of the left-hand side of Eq. (1). The coefficients E_1 etc. are given by

$$Z_1 = A_{3_{ijk-\frac{1}{2}}} \qquad B_1 = A_{2_{ij-\frac{1}{2},k}}$$

$$D_1 = A_{1_{i-\frac{1}{2}jk}} \qquad F_1 = A_{1_{i+\frac{1}{2}jk}}$$

$$H_1 = A_{2_{ij+\frac{1}{2}k}} \qquad S_1 = A_{3_{ijk+\frac{1}{2}}}$$

and

$$E_1 = -(Z_1+B_1+F_1+H_1+S_1+D_1)$$

The SIP replaces the above difference equation (3) by the following factored equation:

$$(d_{ijk}+a_{ijk}E_\zeta^-+b_{ijk}E_\eta^-+c_{ijk}E_\xi^-) \times (1+e_{ijk}E_\xi^++f_{ijk}E_\eta^++g_{ijk}E_\zeta^+)C^{n+1} = R_1(\phi^n)-q \tag{4}$$

where

$$a_{ijk} = Z_1/[1+\alpha(e_{ijk-1}+f_{ijk-1})]$$

$$b_{ijk} = B_1/[1+\alpha(e_{ij-1k}+g_{ij-1k})]$$

$$c_{ijk} = D_1/[1+\alpha(f_{i-1jk}+g_{i-1jk})]$$

$$d_{ijk} = E_1+\alpha[a_{ijk}(e_{ijk-1}+f_{ijk-1})+b_{ijk}(e_{ij-1k}+g_{ij-1k}) +c_{ijk}(f_{i-1jk}+g_{i-1jk})]-[a_{ijk}g_{ijk-1}+b_{ijk}f_{ij-1k} +c_{ijk}e_{i-1jk}]$$

$$e_{ijk} = [F_1-\alpha(a_{ijk}e_{ijk-1}+b_{ijk}e_{ij-1k})]/d_{ijk}$$

$$f_{ijk} = [H_1-\alpha(a_{ijk}f_{ijk-1}+c_{ijk}f_{i-1jk})]/d_{ijk}$$

$$g_{ijk} = [S_1-\alpha(c_{ijk}g_{i-1jk}+b_{ijk}g_{ij-1k})]/d_{ijk}$$

Here α is a relaxation factor cyclically varied between the limits $0<\alpha<1$ during iterations.

The following observations may be made regarding the SIP:

1) The coefficients of the factored equation are nonlinear functions of the coefficients of the model equation, and therefore are not known a priori. Instead they must be computed using the recursive relationship described above. The coefficients in the first factor need not be stored since the first sweep may be performed as soon as these coefficients are computed. The coefficients of the second factor should, however, be stored for later use in the second sweep. Thus, SIP requires storage of three large vectors, e_{ijk}, f_{ijk}, and g_{ijk}, whose dimensions are the same as those of ϕ_{ijk} and C_{ijk}.

2) The factored equation is in the familiar LU form, and may be solved easily. The inversion of LU matrices, and the solution of C^{n+1} is performed efficiently in scalar machines since this is essentially a recursive process. (The question of vectorizing the SIP is taken up in a separate section.)

3) The total number of arithmetic operations involved in the computation of the coefficients a_{ijk} through g_{ijk}, and the LU solution, is more than the number of operations required by ADI procedure, which essentially involves calling the Thomas tridiagonal algorithm three times at each point. (Three calls to the Thomas algorithm require about 15 arithmetic operations, while the SIP requires about 32 arithmetic operations.) Thus, the CPU time per point per iteration is somewhat higher for the SIP compared to the ADI procedure.

4) In certain situations, the SIP factorization becomes locally exact so that the solution is obtained locally in just one iteration. Consider, for example, a situation where the

coefficient A_1 is much larger than the coefficients A_2 and A_3 ($A_1 \gg A_2$ and $A_1 \gg A_3$). The SIP factorization becomes

$$(\vec{\partial}_\xi A_1 \vec{\partial}_\xi) C^{n+1} = -R_1(\phi^n) - q \tag{5}$$

The ADI factorization for this situation reduces locally to the following form:

$$\left(L_1 - \frac{\partial}{\partial \xi} A_1 \frac{\partial}{\partial \xi}\right) C^{n+1} = \omega R_1(\phi^n) + \omega q \tag{6}$$

Thus, the ADI factorization does not become an exact factorization unless the user externally sets the operator L_1 to zero at these points. No such user intervention is necessary in SIP.

A proof of this observation, under some restrictive conditions, is given in Ref. 11.

5) The SIP has the desirable property of matrix conditioning. That is, it automatically adjusts the diagonals of the first and second factors so that they are always well conditioned, both in the regions where the mesh spacing is small and in those where mesh spacing is large. As noted earlier, ADI does not automatically condition the matrices that arise from the factorization. Even though the user is at privilege to introduce matrix conditioning by varying the parameters α_0, etc., spatially, it is not generally done because this practice leads to an inflexible code not readily adapted to a wide variety of geometries.

The above property of matrix conditioning may be analytically shown for the case when $\alpha = 0$. For other values of α, except $\alpha = 1$, some matrix conditioning is always present, although this can only be verified numerically. For the case $\alpha = 0$, the existence of this property is proved for some restrictive conditions in Ref. 11.

6) It may be shown for relaxation problems SIP converges in $O(N \ell n N)$ equations even if a nonoptimal sequence in α is used, while ADI converges in $O(N \ell n N)$ operations only if an optimal sequence is used. At present this concept may only be verified numerically.

In summary, SIP leads to a relaxation algorithm that requires some additional memory and computing effort. However, it has the remarkable property of matrix conditioning and also the property of becoming locally an exact factorization in certain highly stretched coordinate systems.

Having examined some of the interesting properties of SIP, we now turn our attention to the construction of an efficient algorithm for numerical solution of steady transonic potential flow past thick swept wings. The present algorithm is extended readily to unsteady transonic potential flows. Results for the unsteady transonic potential flow are presented in a separate report.[12]

Governing Equations

The full potential equation in a strong conservation form may be written in a curvilinear coordinate system (ξ, η, ζ) as

$$\left(\frac{\rho U}{J}\right)_\xi + \left(\frac{\rho V}{J}\right)_\eta + \left(\frac{\rho W}{J}\right)_\zeta = 0 \tag{7}$$

where

$$\rho = \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - U\phi_\xi - V\phi_\eta - W\phi_\zeta)\right]^{\frac{1}{\gamma - 1}}$$

$$U = A_1\phi_\xi + A_2\phi_\eta + A_3\phi_\zeta$$

$$V = A_2\phi_\xi + A_4\phi_\eta + A_5\phi_\zeta$$

$$W = A_3\phi_\xi + A_5\phi_\eta + A_6\phi_\zeta$$

$$A_1 = \xi_x^2 + \xi_y^2 + \xi_z^2 \qquad A_2 = \xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z$$

$$A_3 = \xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z \qquad A_4 = \eta_x^2 + \eta_y^2 + \eta_z^2$$

$$A_5 = \eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z \qquad A_6 = \zeta_x^2 + \zeta_y^2 + \zeta_z^2$$

$$J = \xi_x\eta_y\zeta_z - \xi_x\eta_z\zeta_y + \xi_y\eta_z\zeta_x - \xi_y\eta_x\zeta_z + \xi_z\eta_x\zeta_y - \xi_z\eta_y\zeta_x$$

In the above equations ρ is the density normalized with respect to its freestream value; M_∞ is the freestream Mach number; γ is the ratio of specific heats; and U, V, and W are the contravariant components of velocity. A_1, A_2, A_6, etc., are metrics of transformation and J is the Jacobian of transformation.

Grid Generation

In the present work, the well known sheared parabolic coordinate transformation, extensively used by Jameson and his co-workers,[13] is used to construct a body-fitted coordinate system around a thick swept wing mounted on a wall. We briefly describe the grid generation procedure. The transformation consists of a sequence of steps. In the first step, the wing planform, which is an arbitrary shape in the physical coordinate system (x,y,z), is reduced to a rectangle in a coordinate system (x_1, y_1, z_1) by the transformation

$$x_1 = \frac{x - x_0(y)}{c(y)}, \quad z_1 = \frac{z}{c(y)}, \quad y_1 = y \tag{8}$$

Here $x_0(y)$ is the wing leading-edge location at each spanwise station y; $c(y)$ is the wing local chord on the wing, and beyond the wing tip $c(y)$ is set equal to wing chord at the tip. This introduces a discontinuity in the slope of $c(y)$ plotted against y at the tip, but appears to create no adverse effects.

In the second step, at each $y_1 = \text{const}$ station, the airfoil profile is unwrapped about a singular point (x_s, y_1, z_s) located just inside the wing leading edge, according to the transformation:

$$x' + \sqrt{-1}z' = \sqrt{x_1 - x_s + \sqrt{-1}(z_1 - Z_s)}, \quad y' = y_1 \tag{9}$$

Continuation of the square root transformation beyond the wing tip introduces a singular line in the works of Jameson and others requiring special treatment. In the present work, the wing is extended beyond its tip with fluid airfoils of very small but finite thickness. (Since these are fluid airfoils, ϕ must be continuous across them, and appropriate action is taken while applying the boundary conditions on the surface of the airfoil. Continuity is ensured by averaging ϕ at grid points above and below the fluid airfoil. At the leading edge of the fluid airfoil, the value of ϕ is obtained by extrapolation from adjacent points. These statements will be clarified during our discussion on boundary conditions.)

The above square root transformation unwraps the wing, and the region beyond the wing tip as well as any assumed vortex sheet shape behind the wing, into a smoothly varying surface $S(x', y')$. A shearing transformation is now applied:

$$\bar{y} = y' \qquad \bar{x} = x' \qquad \bar{z} = z' - S(x', y') \tag{10}$$

Following the above transformation, stretching is applied to obtain the final computational domain (ξ, η, ζ) where

$$\xi = \xi(\bar{x}) \qquad \eta = \eta(\bar{y}) \qquad \zeta = \zeta(\bar{z}) \tag{11}$$

In the computational domain the grid points are uniformly placed, so that $\Delta\xi = \Delta\eta = \Delta\zeta = 1$. In the physical plane (x,y,z), the grid points are clustered closely near the wing and are highly stretched in other regions.

Numerical Computation of Metrics

The quantities ξ_x, ξ_y, etc., that are needed to compute the metric coefficients A_1, A_2, etc., are related to the quantities x_ξ, x_η, etc., by the relationship

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix}^T = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix}^{-1} \tag{12}$$

In our work x_ξ, x_η, etc., were computed using fourth-order-accurate central differences at interior points. At points adjacent to the computational boundaries, second-order-accurate central differences were used. At points on the boundary, second-order-accurate three-point one-sided differences were used. The metrics A_1, A_2, etc., were not stored but were computed as and when needed.

Care must be taken while choosing higher order formulas for the computation of metrics, as observed by Steger and Caradonna.[10] Even in a region of uniform flow, with density set to unity, the discretized form of Eq. (1) can give a large nonzero residual if the metrics are evaluated using arbitrarily higher order formulas. In our work, it was observed that the use of a reduced potential G, equal to ϕ minus the contribution due to freestream velocity, reduces this residual substantially, even when fourth-order-accurate formulas were used to compute metrics. For clarity, we describe only our formulation based on ϕ in this work.

Discretization of Governing Equations

A second-order-accurate finite difference approximation to Eq. (7) is obtained by replacing all of the spatial derivatives with central difference formulas. For example, a quantity $[(\rho A_1/J)\phi_\xi]_\xi$ is discretized at a point (i,j,k) as

$$\delta_\xi\left(\frac{\rho A_1}{J}\phi_\xi\right) = \left(\frac{\rho A_1}{J}\right)_{i+\frac{1}{2}jk}(\phi_{i+1jk}-\phi_{ijk}) - \left(\frac{\rho A_1}{J}\right)_{i-\frac{1}{2}jk}(\phi_{ijk}-\phi_{i-1jk}) \tag{13}$$

The density and metric coefficients were evaluated at nodal points (i,j,k), and their values at half points were computed using averages.

In order to avoid treatment of a system of nonlinear equations, the density was lagged behind ϕ by one iteration. The cross derivatives were also evaluated at the earlier iteration and moved to the right-hand side. A quantity such as $[(\rho A_2/J)\phi_\eta]_\xi$ was evaluated as

$$\delta_\xi\left[\rho\frac{A_2}{J}\phi_\eta\right] = \tfrac{1}{4}(\rho A_2/J)_{i+\frac{1}{2}jk} \times (\phi_{i+1j+1k}+\phi_{ij+1k}-\phi_{ij-1k}-\phi_{i+1j-1k}) - \tfrac{1}{4}(\rho A_2/J)_{i-\frac{1}{2}jk}(\phi_{ij+1k}+\phi_{i-1j+1k}-\phi_{ij-1k}-\phi_{i-1j-1k}) \tag{14}$$

In the computation of density, similarly, central differences were used to compute the spatial derivatives wherever possible. At the far-field boundary, the density was set to unity. On the wing surface determination of density requires evaluation of the quantity ϕ_ζ. The zero-normal velocity condition was invoked as suggested by Holst,[6] and ϕ_ζ was linked to ϕ_ξ and ϕ_η, which may be evaluated using central differences. Similar treatment was given to points on the end wall and at the wing/end wall interface.

The discretization just described is stable in subsonic regions and second order accurate. In supersonic regions, an upwind bias is needed for stability, and the density is modified using the artificial density concept extensively used by Holst and Ballhaus,[14] Holst,[6] and others. For example, a term such as $(\rho U/J)_\xi$ in Eq. (1) is replaced by $[\hat{\rho}(U/J)]_\xi$, were $\hat{\rho}$ is defined by

$$\hat{\rho}=\rho \text{ in subsonic regions} \tag{15}$$

$$\hat{\rho}=\rho-\nu[(1-\epsilon)\vec{\delta}_\xi\rho+\epsilon\overleftarrow{\delta}_{\xi\xi}\rho] \quad \text{in supersonic regions}$$

Here

$$\nu=\min[1,2(M^2-1)]$$

Both first- and second-order-accurate discretization formulas may be obtained by setting ϵ either to be zero or close to unity. The arrow over the difference operators indicates that windward differences are taken.

In the present work, ϵ was set to zero, giving a first-order-accurate formula in supersonic regions. In a recent work, Steger and Caradonna[10] suggest that the upwind bias may be applied both in the subsonic and supersonic regions. This would eliminate the test on local Mach number, and, in some cases, eliminate the spurious expansions ahead of strong shock waves. Note that by setting $\nu=1$ and $\epsilon=0.8$ and setting

$$\delta_{\xi\xi}\rho \simeq (\rho_{i-2}-2\rho_{i-1}+\rho_i)$$

one would recover the artificial density formulas used by the above authors in some of their numerical results.

One can obtain a rotated difference scheme by applying the artificial bias to density in all three coordinate directions. We have experimented with the above rotated difference scheme, as well as with some other forms of artificial density considered by Hafez et al.[15] In the results presented here, the upwind bias was applied only in the ξ direction.

Boundary Conditions

The discretized form of Eq. (7) was applied at all of the interior points as well as points on the end wall. At the points on the end wall, the boundary condition is $V=0$, which was imposed by setting

$$\left[\frac{\rho V}{J}\right]_{ij-\frac{1}{2}k} = -\left[\frac{\rho V}{J}\right]_{ij+\frac{1}{2}k} \tag{16}$$

at these points.

On the wing surface, the zero normal velocity boundary condition gives

$$W=A_3\phi_\xi+A_5\phi_\eta+A_6\phi_\zeta=0$$

or

$$\phi_\zeta=-[(A_5\phi_\eta+A_3\phi_\xi)/A_6] \tag{17}$$

The terms on the right-hand side were evaluated at the known iteration level. The term on the left-hand side was written as

$$\phi_\zeta=(4\phi_{ij2}-3\phi_{ij1}-\phi_{ij3})/2 \tag{18}$$

thus linking the point on the wing surface $(i,j,1)$ with points in its neighborhood. Equation (17) was implicitly built into the discretized form of Eq. (7) while the latter was applied at $(i,j,2)$.

At points on the vortex sheet, and also on the fluid airfoil sections, the velocity potential was explicitly updated after the velocity potential was determined at all interior points at the new iteration level.

Considering first the points on the vortex sheet, there are two computational points for each physical point, one on either side of the vortex sheet. At these points the conditions that the normal velocity be continuous and that the jump in potential be the same as the jump at the wing trailing edge

immediately upstream lead to two simultaneous equations which may be solved for the values of ϕ at these two computational points.

At points on the fluid airfoil section, everywhere except at the nose, there are two computational points, one on the upper surface and the other on the lower surface of the airfoil. The same conditions as those applied at the vortex sheet may be applied at these points except that the jump in velocity potential is zero.

Special treatment is needed at the nose points of the fluid airfoil section because there is only one computational point for each physical point. Because mass can not be created inside the fluid airfoil, neglecting spanwise variation, we required $\int \rho\phi_\eta ds$ to be zero at each station. Since $\rho\phi_\eta$ at all other points of the fluid airfoil exactly balance and cancel, at the nose we require $\phi_\eta = 0$. This is achieved by setting $\phi_{i,j,1} = \phi_{i,j,2}$ to a first-order accuracy, where the point $(i,j,1)$ is the nose point.

Results and Discussion

A computer code has been written that is capable of treating steady transonic flow past swept wings. Because the governing equations are solved in a curvilinear coordinate system, the program is capable of analyzing wings of arbitrary geometric characteristics. Only the coordinates of the body fitted grid system need to be specified. The code may easily be extended to treat wing-body combinations as well. However, in the numerical results presented here, only wing-alone geometries are considered.

A first test case studied was the nonlifting potential flow past a wing of high aspect ratio equal to 100, and a sweep angle Λ of 30 deg mounted between wind tunnel walls. A NACA 64A006 airfoil section, whose thickness was scaled down by a factor of cosΛ was used. The freestream Mach number was set to 0.8/cosΛ. A sheared parabolic coordinate system, with 130 nodes in the streamwise (ξ) direction, 20 nodes in the ζ direction, and 5 spanwise stations in the η direction, was used. At the end of 40 iterations, the surface pressure had converged within plottable accuracy to the properly scaled two-dimensional solution predicted by the simple sweep theory, as seen in Fig. 1. For this case the maximum residual dropped by more than three orders of magnitude in about 90 iterations.

A series of calculations for a swept wing mounted between wind tunnel walls, with NACA0015 airfoil sections, was carried out. The wing sweep angle was 25 deg, the freestream Mach number was 0.86, and the aspect ratio was set to 9.5. At the end of about 54 iterations, the solution converged to the numerical results obtained by Holst using the AF2 procedure,[6] as seen in Fig. 2.

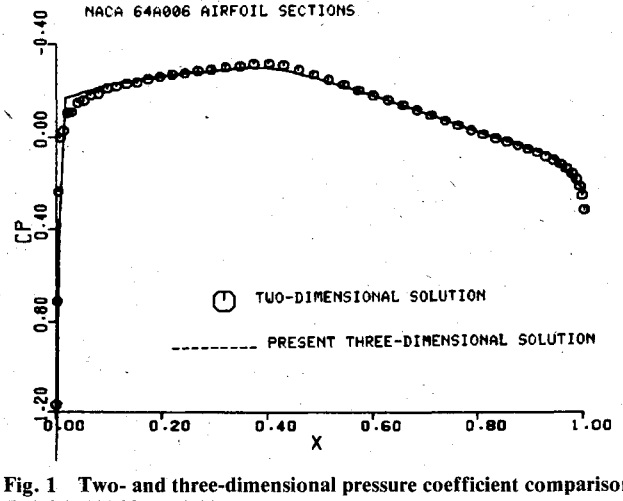

Fig. 1 Two- and three-dimensional pressure coefficient comparisons (NACA 64A006 airfoil).

The accuracy, reliability, and convergence speed of the present algorithm for practical problems was tested by studying lifting and nonlifting transonic potential flows past the ONERA D wing. This wing has been extensively studied both experimentally[14] and numerically[7,11] and is therefore an excellent candidate for evaluation of new algorithms. A $130 \times 24 \times 17$ grid was used in the present study for nonlifting flows, including 100 nodes on the wing at each spanwise station in the ξ direction and 16 spanwise cells on the wing. For lifting flows, a $160 \times 28 \times 17$ grid with 20 spanwise cells was used.

Two flow conditions were studied. In the first study, the angle of attack was set to zero, and the freestream Mach number was set to 0.923. The iterative solution rapidly converged to the steady solution, and at the end of 99 iterations the surface pressure distribution had converged, at least to three significant digits, everywhere. The converged solutions are plotted at several spanwise stations in Figs. 3-6, and are compared with experiments.[14]

In Fig. 7, the surface pressure solution at midspan is compared with a nonconservative transonic potential flow solution obtained by executing the FLO-22 code for the same flow conditions. It is clear from the above figures that the present algorithm gives numerical results that are in agreement with experiments and other existing codes. Note that the FLO-22 code was used here for comparison, because it gives nearly identical results, even at the shock, as the more

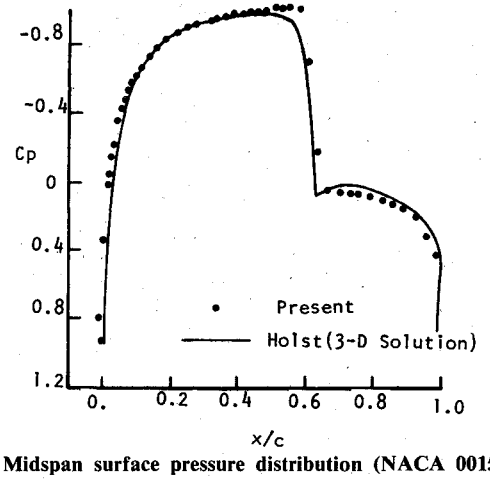

Fig. 2 Midspan surface pressure distribution (NACA 0015 wing, $\mathcal{R} = 10$, $\lambda = 25$ deg, $\alpha = 0$ deg, Mach No. = 0.86).

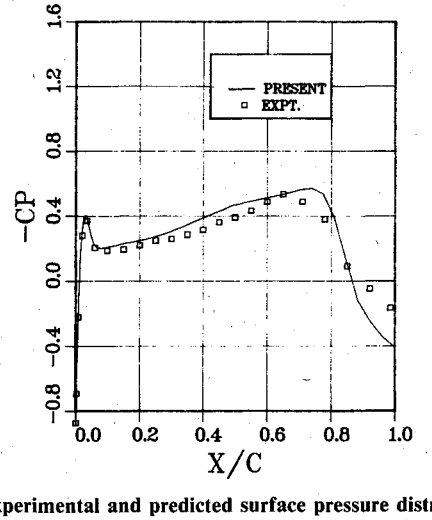

Fig. 3 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923, 20% span).

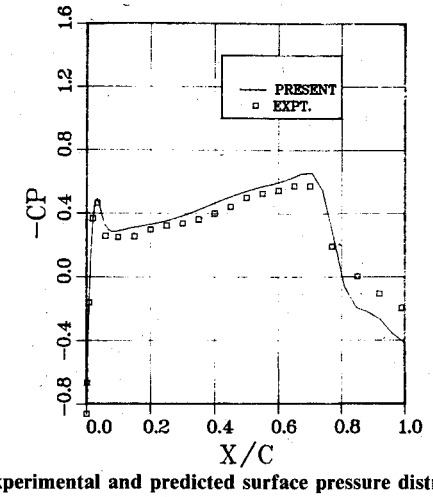

Fig. 4 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923, 45% span).

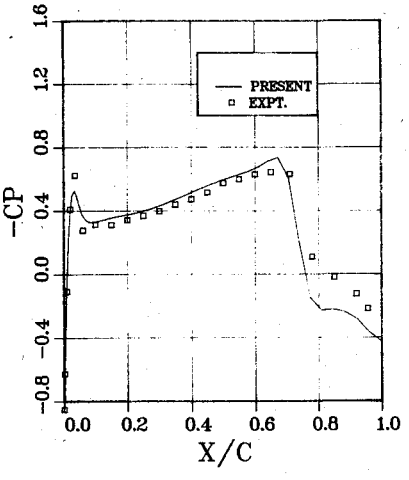

Fig. 5 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923, 65% span).

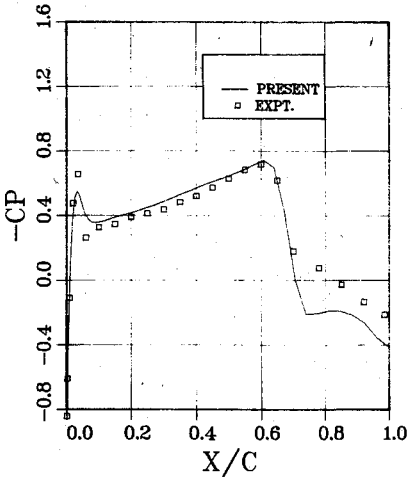

Fig. 6 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923, 80% span).

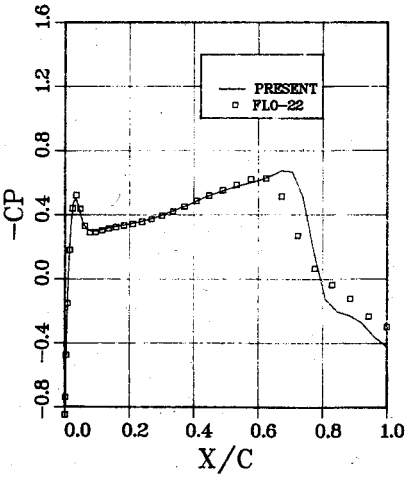

Fig. 7 Midspan pressure distribution compared to solution from FLO-22 (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923).

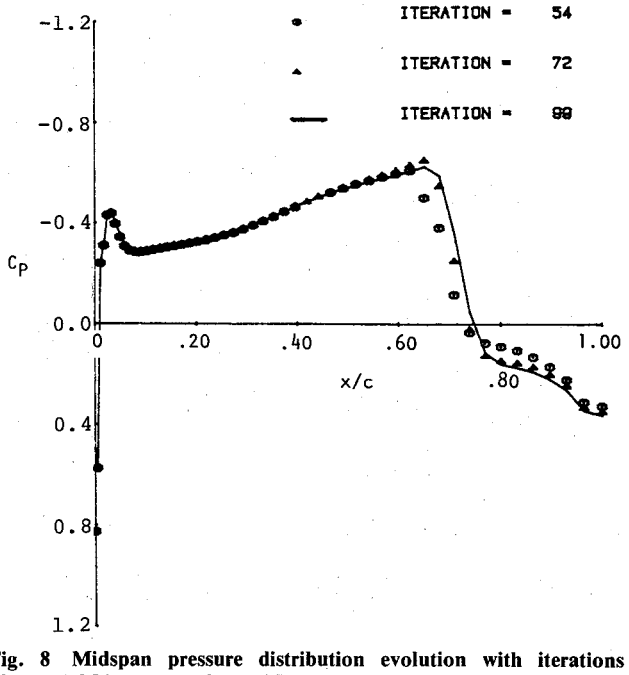

Fig. 8 Midspan pressure distribution evolution with iterations (ONERA M6 wing, $\alpha = 0$ deg, Mach No. = 0.923).

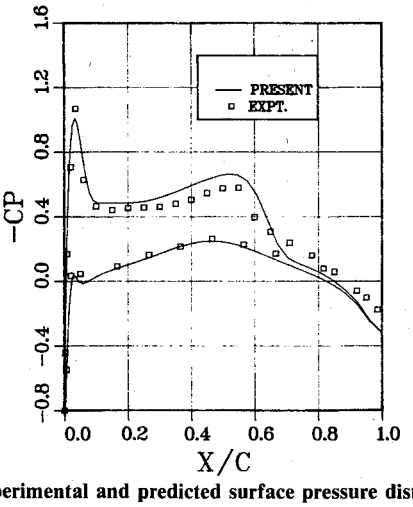

Fig. 9 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 3.06$ deg, Mach No. = 0.84, 20% span).

expensive, but fully conservative, FLO-27 code for this wing.

Some indications of the convergence speed that can be expected from the present relaxation procedure may be obtained by examining Fig. 8, where the surface-pressure distribution at midspan is plotted at 54, 72, and 99 iterations. It is clear that a converged pressure distribution is achieved in about 54 iterations everywhere except at the shock, and that by 72 iterations even near the shock the pressure distribution has converged.

In the second study involving the ONERA D wing, the angle of attack was set to 3.06 deg, and the freestream Mach number was set to 0.84. The introduction of another unknown, namely, the circulation generated by every spanwise station, slows down the convergence for this case somewhat compared to the previous nonlifting case; in addition, the total number of nodes for the lifting case is about 30% more than that for the nonlifting case, and this contributed an increase in the number of iterations needed for convergence. In our computations, a converged surface pressure distribution was obtained in about 120 iterations.

In Figs. 9-11 the surface pressure distributions at several spanwise stations are plotted and compared with experimental results. These lifting wing results were computed using the reduced potential formulation, with density computed at midpoints. In Ref. 11, results for the full potential-nodal density formulation are given.

The above results indicate that SIP converges very rapidly to a reasonably accurate solution. But there is still some room for improvement, particularly with regard to solution accuracy. In the nonlifting flow case, the predicted shock location at the 65% outboard station is ahead of the experimental results, while the exactly opposite effect would be expected. This appears to be caused by large discretization errors due to the rapid stretching of the coordinates. In the lifting case, the forward shock is not captured properly. This is caused by the fact that the amount of artificial viscosity used in the supersonic region was larger than that suggested by a Murman-Cole type of switching, leading to an extreme smearing of weak shock waves. Currently, work is being carried out to improve the accuracy of the present algorithm.

Computer Time and Storage Requirements

All of the computations were performed on a VAX 11/780 minicomputer system. This computer has virtual addressing capability and therefore can handle programs that require very large memory. We have, nevertheless, kept the memory requirements of the computer code to a minimum, so that the present code can be executed on the CDC 7600 class of computers. In the code five large arrays are used to store ϕ, $R(\phi^n)$, e, f, and g. Note that e, f, and g are required in the SIP factorization procedure. All other quantities, including the x, y, and z coordinates of the grid, the density, and the metric coefficients, are recomputed as and when needed. We estimate that a three-dimensional problem involving 64,000 nodes may be solved on the CDC 7600 if both the LCM and SCM memory of the machine amounting to a total of 384K, are used. If ϕ is not stored, but is sequentially buffered into and out of the central core memory, even larger problems involving, say, 80,000 nodes, may be solved.

The CPU time per iteration of the present computer code was evaluated by running, for a few iterations, a test problem involving 30,000 nodes on a CDC 7600 available on the CYBERNET system. Based on this test, we estimate that a problem involving 46,000 nodes would require 5.8 s per point per iteration using the code described here, and this may be compared with the 4.5 s CPU time used by the AF2 algorithm described by Holst[6] for three-dimensional flows with a comparable total number of nodes. Since the present computer program is purely a research code, some reduction in CPU time can be achieved by careful reprogramming of the code.

Based on the above CPU time estimate, a practical problem on a $130 \times 17 \times 28$ grid that involves 100 nodes on the wing in the streamwise direction and 20 spanwise cells on the wing may be solved on the CDC 7600 in about 13-15 min based on 100 iterations. With careful reprogramming, the CPU time may be reduced to about 10 min or less. In its present form, the code is twice as fast as existing line relaxation procedures such as FLO-27. Currently, grid sequencing is being built into the existing code, and this essentially involves getting a good starting solution by solving the problem on a coarse grid, with 8-64 times fewer points. With the grid sequencing, the SIP is expected to be 5 times faster than FLO-27.

Among other three-dimensional potential flow procedures, the AF2 method described by Holst promises to be another implicit procedure with excellent convergence properties, and for the NACA0015 case reported here, the AF2 procedure apparently requires essentially the same number of iterations as the SIP. Clearly, any detailed comparison of SIP and AF2 will require that identical problems be solved on comparable grids with equal grid points. Such a detailed comparison is not attempted here. It may also be mentioned that the AF procedure documented by Chattot and co-workers[7] requires, based on a total number of 49,000 nodes, 15-30 min on the CDC 7600, and appears to be another promising development for the implicit iterative solution of transonic potential flow. A unique feature of the SIP, in contrast to the AF, AF2, and line relaxation procedures described here, is that it may be

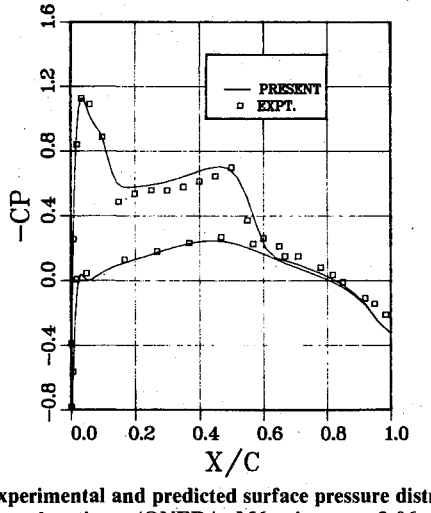

Fig. 10 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 3.06$ deg, Mach No. = 0.84, 45% span).

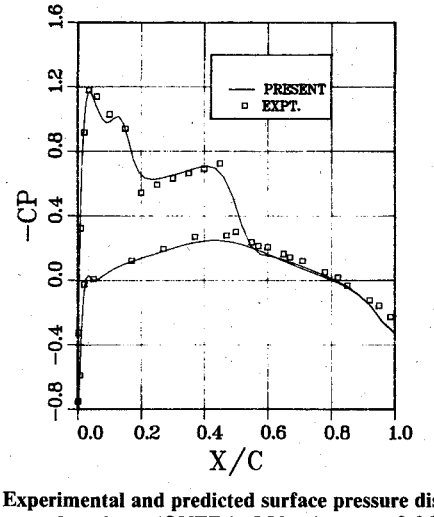

Fig. 11 Experimental and predicted surface pressure distributions at selected span locations (ONERA M6 wing, $\alpha = 3.06$ deg, Mach No. = 0.84, 65% span).

readily extended to three-dimensional unsteady flows with little or no modification in the factorization algorithm, thereby resulting in a single computer code that is capable of solving steady and unsteady problems simply by changing a parameter in the input data.[12]

Vectorization of the SIP

The iterative procedure, as described in the present work, at first glance, can not be easily vectorized because of the large number of recursive operations. Some simple modifications rectify this shortcoming. For example, the L and U factors in the SIP may be broken up into smaller operators, in the spirit of the Douglas-Gunn factorization, each smaller operator involving one side differences only in the ξ, η, or ζ directions. Then the inversion of these operators may be vectorized in much the same way as an ADI procedure would be vectorized. This breaking up of operators may be done without losing the matrix conditioning properties of SIP. Similarly, modifications are possible in the recursive computations of the factors e, f, and g. Other computations, such as evaluation of metrics, density, residual $R(\phi^n)$ etc., may be vectorized without any modifications.

Concluding Remarks

An algorithm has been developed that is capable of treating practical transonic lifting potential flow past three-dimensional geometries. A reliable converged solution is obtained in less than 120 iterations, and this is a significant improvement over existing line relaxation algorithms, which will require, in the absence of grid sequencing, hundreds of iterations to obtain a converged solution. The algorithm can handle highly stretched grids without loss in convergence speed, and no precise estimates of the optimal sequence of iteration parameters are necessary. Any reasonable sequence in α may be adapted. With very little modification in the basic factorization, the code may be extended to treat unsteady transonic potential flows as well.

References

[1] Holst, T. L., "An Implicit Algorithm for the Conservative Transonic Full Potential Equation Using an Arbitrary Mesh," AIAA Paper 78-1113, 1978.

[2] Jameson, A., "Accelerated Iterative Schemes for Transonic Flow Calculations using Fast Poisson Solvers," New York Univ., New York, ERDA Rept. C00-3077-82, March 1975.

[3] Brandt, A., "Multi-level Adaptive Solution to Boundary Value Problems," *Mathematics of Computation,* Vol. 31, 1977, pp. 333-391.

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

[5] Sankar, N. L. and Tassa, Y., "An Algorithm for Unsteady Transonic Potential Flow Past Airfoils," *Seventh International Conference on Numerical Methods in Fluid Dynamics Proceedings*, Stanford/Ames, June 1980, pp. 367-372.

[6] Holst, T. L., "A Fast Conservative Algorithm for Solving the Transonic Full Potential Equation," *AIAA Fourth Computational Fluid Dynamics Conference Proceedings*, July 1979, pp. 109-121.

[7] Chattot, J. J., Coulombiex, C., and Silva Tome, C., "Calculs D'E coulements Transsoniques Autor D'Ailes," *La Recherche Aerospatiale*, April 1978, pp. 143-159.

[8] 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.

[9] Sampath, S., "A Numerical Study of Incompressible Viscous Flow Around Airfoils," Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Ga., 1977.

[10] Steger, J. L. and Caradonna, F. X., "A Conservative Implicit Finite Difference Algorithm for the Unsteady Transonic Full Equation," AIAA Paper 80-1368, July 1980.

[11] Sankar, N. L., Malone, J. B., and Tassa, Y., "A Strongly Implicit Procedure for Steady Three-Dimensional Transonic Potential Flows," AIAA Paper 81-0385, Jan. 1981.

[12] Sankar, N. L., Malone, J. B., and Tassa, Y., "An Implicit Conservative Algorithm for Steady and Unsteady Three-Dimensional Transonic Potential Flows," *AIAA Fifth Computational Fluid Dynamics Conference Proceedings,* June 1981, pp. 199-212.

[13] Jameson, A. and Caughey, D. A., "A Finite Volume Method for Transonic Potential Flow Calculation," *AIAA Third Computational Fluid Dynamics Conference Proceedings*, 1977, pp. 35-54.

[14] Holst, T. L. and Ballhaus, W. F., "Conservative Implicit Schemes for the Full Potential Equation Applied to Transonic Flows," NASA TM-78469, March 1978.

[15] 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, 1978.

[16] Monnerie, B. and Charpin, F., "Essais de Buffeting d'une aile en fleche en transsonique," *10th Colloque d'Aerodynamique Appliquee,* Lille, France, Nov. 1973.