JOURNAL OF COMPUTATIONAL PHYSICS 39, 347–363 (1981)

A Diagonal Form of an Implicit Approximate-Factorization Algorithm

T. H. PULLIAM

Ames Research Center, NASA, Moffet Field, California 94035

AND

D. S. CHAUSSEE

Flow Simulations, Inc., 298 Sunnyvale Avenue, Sunnyvale, California 94086

Received December 5, 1979

A modification of an implicit approximate-factorization finite-difference algorithm applied to partial differential equations is presented. This algorithm is applied to the two- and three-dimensional Euler equations in general curvilinear coordinates. The modification transforms the coupled system of equations into an uncoupled diagonal form that requires less computational work. For steady-state applications, the resulting diagonal algorithm retains the stability and accuracy characteristics of the original algorithm. The diagonal algorithm reduces the storage requirement of the implicit solution process and therefore has an important effect on the application of implicit finite-difference schemes to vector processors. Results are presented for realistic two-dimensional transonic flow fields about airfoils. Computation costs are reduced 24–34%.

1. INTRODUCTION

Efficient means of solving the fluid dynamic equations—for example, the inviscid Euler equations—are constantly being sought. Various time-accurate methods, both explicit and implicit, have been proposed. These schemes have been used for unsteady time-accurate computations and also for time-like relaxation to steady-state solutions.

Implicit finite-difference schemes are attractive because of their stability bounds, which allow larger time steps to be taken for either a faster time advance of a time-accurate calculation or as a method to increase convergence rates for steady-state calculations. Even though explicit schemes have more restrictive stability limitations, they generally require less computational work per time step than implicit methods.

The major portion of the computational work in an implicit finite-difference algorithm is contained in the solution of a set of simultaneous equations. When an implicit algorithm is applied to a system of partial differential equations, one obtains block matrix–vector equations that are complicated and time-consuming to solve. A method for uncoupling the solution process, through a diagonalization of the block-

0021-9991/81/020347-17$02.00/0

Copyright © 1981 by Academic Press, Inc.
All rights of reproduction in any form reserved.

matrix structure, is presented. The method is applied to an implicit approximate-factorization algorithm [1] for the two- and three-dimensional inviscid Euler equations in general curvilinear coordinates. The accuracy and stability of the diagonal algorithm are examined and contrasted with the original scheme. The effect of the diagonal algorithm on the application of implicit schemes to vector computers is briefly discussed. Finally, results are presented for realisitic two-dimensional transonic flow fields about airfoils, in which the diagonal algorithm is compared with the standard implicit algorithm.

2. Implicit Algorithm and Diagonal Formulation

A. *Cartesian Equations*

The two-dimensional conservation-law form of the Euler equations cast in Cartesian coordinates and nondimensional variables are

$$\partial_t \bar{q} + \partial_x \bar{E} + \partial_y \bar{F} = 0, \tag{1a}$$

where

$$\bar{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \qquad \bar{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e+p)u \end{bmatrix}, \qquad \bar{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (e+p)v \end{bmatrix}. \tag{1.b}$$

Density ρ is scaled to ρ_∞ (the free-stream density); Cartesian velocities u and v to c_∞ the free-stream speed of sound); and total energy e is nondimensionalized with respect to $\rho_\infty c_\infty^2$. Pressure is obtained from the equation of state for a perfect gas,

$$p = (\gamma - 1)[e - \tfrac{1}{2}\rho(u^2 + v^2)], \tag{2}$$

where γ is the ratio of specific heats.

B. *General Curvilinear Coordinates*

If general curvilinear coordinate transformations are introduced into the Euler equations, where the inertial Cartesian momenta are retained as the dependent variables, the strong conservation-law form is maintained; for example, see Lapidus [2], Viviand [3], and Vinokur [4]. For general curvilinear coordinate transformations

$$\begin{aligned} \xi &= \xi(x, y, t), \\ \eta &= \eta(x, y, t), \\ \tau &= t. \end{aligned} \tag{3}$$

Applied to Eq. (1), we have

$$\partial_\tau \hat{q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} = 0, \tag{4a}$$

$$\hat{q} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \qquad \hat{E} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (e+p)U - \xi_t p \end{bmatrix},$$

$$\hat{F} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (e+p)V - \eta_t p \end{bmatrix}, \tag{4b}$$

and

$$U = \xi_t + \xi_x u + \xi_y v,$$
$$V = \eta_t + \eta_x u + \eta_y v.$$

The metric terms are

$$\begin{aligned} \xi_x &= J y_\eta, \\ \xi_y &= -J x_\eta, \\ \eta_x &= -J y_\xi, \\ \eta_y &= J x_\xi, \\ \xi_t &= -x_\tau \xi_x - y_\tau \xi_y, \\ \eta_t &= -x_\tau \eta_x - y_\tau \eta_y, \end{aligned} \tag{5}$$

with

$$J^{-1} = x_\xi y_\eta - x_\eta y_\xi.$$

C. *Standard Solution Algorithm*

The implicit approximate-factorization algorithm applied to Eqs. (1) has been described in detail by Beam and Warming [5]. Steger [6] has applied the algorithm to the transformed Eqs. (4); details of the equations, the algorithm, and some numerical calculations can be found in that report. The algorithm can be first- or second-order accurate in time with first-order being used in the following discussions. Local time linearization is applied to the nonlinear terms and an approximate factorization of the two-dimensional implicit operator is used to produce locally one-dimensional operators.

An implicit approximate-factorization scheme for Eqs. (4) can be written as

$$(I + h\delta_\xi \hat{A}^n)(I + h\delta_\eta \hat{B}^n)\,\Delta\hat{q}^n = -\Delta t(\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n) = \hat{R}^n, \tag{6}$$

where $\Delta\hat{q}^n = \hat{q}^{n+1} - \hat{q}^n$ and $h = \Delta t$ for first-order accuracy in time.

The Jacobian matrices, $\hat{A}$ and $\hat{B}$, are

$$\hat{A} \text{ or } \hat{B} = \begin{bmatrix} k_t & k_x & k_y & 0 \\ -u\theta + k_x\phi^2 & k_t + \theta - (\gamma-2)k_x u & k_y u - (\gamma-1)k_x v & (\gamma-1)k_x \\ -v\theta + k_y\phi^2 & k_x v - (\gamma-1)k_y u & k_t + \theta - (\gamma-2)k_y v & (\gamma-1)k_y \\ \theta[2\phi^2 - \gamma(e/\rho)] & k_x[\gamma(e/\rho) - \phi^2] & k_y[\gamma(e/\rho) - \phi^2] & \gamma\theta + k_t \\ & -(\gamma-1)u\theta & -(\gamma-1)v\theta & \end{bmatrix}, \tag{7}$$

where $\theta = k_x u + k_y v$ and $\phi^2 = 0.5(\gamma - 1)(u^2 + v^2)$, with $k = \xi$ for $\hat{A}$ and $k = \eta$ for $\hat{B}$. The derivation of Eq. (6) used a local time linearization of the flux vectors, that is,

$$\begin{aligned} \hat{E}^{n+1} &= \hat{E}^n + \hat{A}^n(\hat{q}^{n+1} - \hat{q}^n) + o(\Delta t^2), \\ \hat{F}^{n+1} &= \hat{F}^n + \hat{B}^n(\hat{q}^{n+1} - \hat{q}^n) + o(\Delta t^2), \end{aligned} \tag{8}$$

where $\hat{A}^n = (\partial\hat{E}/\partial\hat{q})^n$ and $\hat{B}^n = (\partial\hat{F}/\partial\hat{q})^n$.

The spatial derivative operators ∂_ξ and ∂_η of Eqs. (4) are approximated with central finite-difference operators δ_ξ and δ_η of either second- or fourth-order accuracy (see [5–7]). Here, second-order central differences are used for the left-hand (implicit) side of Eq. (6) producing block-tridiagonal matrix operators $(I + h\delta_\xi \hat{A}^n)$ and $(I + h\delta_\eta \hat{B}^n)$, which must be inverted sequentially to obtain $\Delta\hat{q}^n$.

Equation (6) can be used for either time-accurate or steady-state computations. For steady-state calculations, $\Delta\hat{q}^n$ approaches zero asymptotically with the solution satisfying the right-hand (explicit) side of Eq. (6), which is the exact steady-state difference equation.

D. *Block-Tridiagonal Matrix Solution Process*

The solution of the matrix equations is obtained through a block lower–upper decomposition (LUD) coupled with forward and backward sweeps. The recursion algorithm for the block-tridiagonal solution process is found in Isaacson and Keller [8]. A relative measure of the work involved in the algorithm can be obtained by looking at the operation counts in terms of the total number of multiplies, adds, and divides. In two-dimensions, the block size is 4×4, and for a typical grid point there are 196 multiplies, 155 adds, and 4 divides. (In three-dimensions the block size is 5×5 and there are 365 multiplies, 325 adds, and 5 divides.) If the work in forming $\hat{A}^n$ and $\hat{B}^n$ is added to the operation counts for the inversion algorithm, the total operation count per grid point for the implicit phase of the integration is 410 multiplies, 326 adds, and 10 divides, a total of 746 operations. The explicit side requires 72 multiplies, 48 adds, and 4 divides, a total of 124 operations per grid count.

An important aspect of the block-tridiagonal solution process is the temporary storage requirement of the recursion algorithm. A close examination of the algorithm reveals that a temporary storage of one block matrix per grid point is required to complete the solution process. In two-dimensions, 16 variables per grid point would be required; in three-dimensions 25 would be required. In some cases, this extra temporary storage can be restrictive.

E. *Diagonal Form of the Implicit Algorithm*

It is quite evident that the block-tridiagonal matrix solutions constitute the major portion of the numerical work of the standard implicit algorithm. Equations (4) are a coupled set and thereby produce a 4×4 block structure for the implicit operators of Eq. (6). If the operators are diagonalized into four scalar operators, the resulting system would be more efficiently solved.

The Jacobian matrices, $\hat{A}$ and $\hat{B}$, have a set of eigenvalues and a complete distinct set of eigenvectors. Similarity transformations (see Warming, Beam, and Hyett [9], or Turkel [10]) can be used to diagonalize $\hat{A}$ and $\hat{B}$, where

$$\hat{A} = T_\xi \hat{\Lambda}_\xi T_\xi^{-1}, \qquad \hat{B} = T_\eta \hat{\Lambda}_\eta T_\eta^{-1} \tag{9a}$$

with

$$\begin{aligned} \hat{\Lambda}_\xi &= D[U, U, U + c(\xi_x^2 + \xi_y^2)^{1/2}, U - c(\xi_x^2 + \xi_y^2)^{1/2}] \\ &= \begin{bmatrix} U & 0 & 0 & 0 \\ 0 & U & 0 & 0 \\ 0 & 0 & U + c(\xi_x^2 + \xi_y^2)^{1/2} & 0 \\ 0 & 0 & 0 & U - c(\xi_x^2 + \xi_y^2)^{1/2} \end{bmatrix}, \\ \hat{\Lambda}_\eta &= D[V, V, V + c(\eta_x^2 + \eta_y^2)^{1/2}, V - c(\eta_x^2 + \eta_y^2)^{1/2}], \end{aligned} \tag{9b}$$

$$T_k = \begin{bmatrix} 1 & 0 & \alpha & \alpha \\ u & \bar{k}_y \rho & \alpha(u + \bar{k}_x c) & \alpha(u - \bar{k}_x c) \\ v & -\bar{k}_x \rho & \alpha(v + \bar{k}_y c) & \alpha(v - \bar{k}_y c) \\ \dfrac{\phi^2}{(\gamma - 1)} & \rho(\bar{k}_y u - \bar{k}_x v) & \alpha\left[\dfrac{\phi^2 + c^2}{(\gamma - 1)} + c\bar{\theta}\right] & \alpha\left[\dfrac{\phi^2 + c^2}{(\gamma - 1)} - c\bar{\theta}\right] \end{bmatrix}, \tag{9c}$$

$$T_k^{-1} = \begin{bmatrix} \left(1 - \dfrac{\phi^2}{c^2}\right) & (\gamma - 1)c^{-2}u & (\gamma - 1)c^{-2}v & -(\gamma - 1)c^{-2} \\ -\rho^{-1}(\bar{k}_y u - \bar{k}_x v) & \bar{k}_y \rho^{-1} & -\bar{k}_x \rho^{-1} & 0 \\ \beta(\phi^2 - c\bar{\theta}) & \beta[\bar{k}_x c - (\gamma - 1)u] & \beta[\bar{k}_y c - (\gamma - 1)v] & \beta(\gamma - 1) \\ \beta(\phi^2 + c\bar{\theta}) & -\beta[\bar{k}_x c + (\gamma - 1)u] & -\beta[\bar{k}_y c + (\gamma - 1)v] & \beta(\gamma - 1) \end{bmatrix} \tag{9d}$$

and $\alpha = \rho/(\sqrt{2}c)$, $\beta = 1/(\sqrt{2}\rho c)$, $\bar{\theta} = \bar{k}_x u + \bar{k}_y v$, and, for example, $\bar{k}_x = k_x/(k_x^2 + k_y^2)^{1/2}$. Relations exist between T_ξ and T_η of the form

$$N = T_\xi^{-1} T_\eta, \qquad N^{-1} = T_\eta^{-1} T_\xi, \tag{10a}$$

where

$$\hat{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & -\mu m_2 & \mu m_2 \\ 0 & \mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & -\mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix},$$

$$\hat{N}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & \mu m_2 & -\mu m_2 \\ 0 & -\mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & \mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix}, \tag{10b}$$

with $m_1 = (\tilde{\xi}_x \tilde{\eta}_y + \tilde{\xi}_y \tilde{\eta}_y)$, $m_2 = (\tilde{\xi}_x \tilde{\eta}_y - \tilde{\xi}_y \tilde{\eta}_x)$, and $\mu = 1/\sqrt{2}$. Note that $\hat{N}$ and $\hat{N}^{-1}$ are independent of the flow variables.

After applying the identities, Eqs. (9) in Eqs. (6), we have the diagonal form

$$[(T_\xi T_\xi^{-1})^n + h\delta_\xi(T_\xi \hat{\Lambda}_\xi T_\xi^{-1})^n][(T_\eta T_\eta^{-1})^n + h\delta_\eta(T_\eta \hat{\Lambda}_\eta T_\eta^{-1})^n]\,\Delta\hat{q}^n = \hat{R}^n, \tag{11}$$

where the identities $T_\xi T_\xi^{-1} = I$ and $T_\eta T_\eta^{-1} = I$ are used.

A modified form of Eq. (11) is constructed by moving T_ξ and T_η outside of the difference operators δ_ξ and δ_η, respectively. This results in the "diagonal" form of the algorithm

$$T_\xi^n(I + h\delta_\xi \hat{\Lambda}_\xi^n)(T_\xi^{-1})^n T_\eta^n(I + h\delta_\eta \hat{\Lambda}_\eta^n)(T_\eta^{-1})^n\,\Delta\hat{q}^n = \hat{R}^n. \tag{12}$$

Since T_ξ and T_η are functions of ξ and η, the modification has introduced an error; the error will be examined in Section 3. Equation (12) can be simplified by using the relations of Eqs. (10) to produce

$$T_\xi^n(I + h\delta_\xi \hat{\Lambda}_\xi^n)\,\hat{N}(I + h\delta_\eta \hat{\Lambda}_\eta^n)(T_\eta^{-1})^n\,\Delta\hat{q}^n = \hat{R}^n. \tag{13}$$

The new implicit operators, $(I + h\delta_\xi \hat{\Lambda}_\xi)$ and $(I + h\delta_\eta \hat{\Lambda}_\eta)$, are still block-tridiagonal, but now the blocks are diagonal in form so that the operators reduce to four independent scalar tridiagonal operators. This has a large positive effect on the solution process discussed below.

Similar reductions of the block operators have been introduced previously by Steger [11] (for the conservative law form) and Briley and McDonald [12] (for the nonconservative form). In these cases the authors point out that the basic structure of the "cartesian form" of the block Jacobian matrices can be rearranged such that the block implicit operators reduce to a series of scalar operators and one smaller ranked block operator. For instance, in two dimensions, one obtains two scalar tridiagonal operators and a 2×2 block implicit operator. Unfortunately, the general curvilinear transformations destroy the reducible structure of the equations. The diagonal algorithm is not so affected by the transformations.

Yanenko and Kovenja [13] have also introduced algorithms which are constructed with scalar sweeps. In this case the physical processes are split, where the convective

terms give scalar implicit operators and the pressure terms give a block implicit operator.

The solution process for the implicit part of Eq. (13) consists of: (1) $S_1 = (T_\xi^{-1})^n \hat{R}^n$, a matrix-vector multiply at each grid point, since T_ξ^{-1} is known analytically; (2) $[I + h\delta_\xi \hat{A}_\xi^n] S_2 = S_1$, solution of four scalar tridiagonal equations; (3) $S_3 = \hat{N}^{-1} S_2$, a matrix-vector multiply at each point; (4) $[I + h\delta_\eta \hat{A}_\eta^n] S_4 = S_3$, four more scalar tridiagonal equations; (5) $\Delta\hat{q}^n = T_\eta^n S_4$, another set of matrix-vector multiplies, and (6) $\hat{q}^{n+1} = \hat{q}^n + \Delta\hat{q}^n$ to update the solution. This contrasts with the two block-tridiagonal matrix solutions required in Eq. (6).

An operation count for the diagonal form of the implicit algorithm yields 233 multiplies, 125 adds, and 26 divides, or 384 operations. This is a significant reduction from the 746 operations required for the standard algorithm. The net effect is a reduction in the total computing costs for the solution of the Euler equations with an implicit numerical algorithm.

F. *Three-Dimensional Algorithm*

The three-dimensional Euler equations in generalized coordinates are

$$\partial_\tau \hat{q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} + \partial_\zeta \hat{G} = 0 \tag{14a}$$

where

$$\hat{q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \hat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (e + p) U - \xi_t p \end{bmatrix}, \quad \hat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_y p \\ (e + p) V - \eta_t p \end{bmatrix},$$

$$\hat{G} = J^{-1} \begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (e + p) W - \zeta_t p \end{bmatrix}, \tag{14b}$$

$$U = \xi_t + \xi_x u + \xi_y v + \xi_z w,$$
$$V = \eta_t + \eta_x u + \eta_y v + \eta_z w,$$
$$W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w.$$

The metric terms are

$$\left.\begin{aligned}\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta)\\ \xi_y &= J(z_\eta x_\zeta - x_\eta z_\zeta)\\ \xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta)\end{aligned}\right\},\quad \left.\begin{aligned}\eta_x &= J(z_\zeta y_\xi - y_\zeta z_\xi)\\ \eta_y &= J(x_\zeta z_\xi - x_\zeta z_\xi)\\ \eta_z &= J(y_\zeta x_\xi - x_\zeta y_\xi)\end{aligned}\right\},$$

$$\left.\begin{aligned}\zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta)\\ \zeta_y &= J(x_\eta z_\xi - x_\xi z_\eta)\\ \zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta)\end{aligned}\right\},\quad \left.\begin{aligned}\xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z\\ \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z\\ \zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z\end{aligned}\right\}, \tag{15}$$

and

$$J^{-1} = x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi.$$

The standard implicit algorithm is described in detail in [14] and is written as

$$(I + h\delta_\xi \hat{A}^n)(I + h\delta_\eta \hat{B}^n)(I + h\delta_\zeta \hat{C}^n)\Delta\hat{q} = -h(\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n + \delta_\zeta \hat{G}^n) = \hat{R}^n. \tag{16a}$$

with

$$\hat{A}, \hat{B}, \text{ or } \hat{C} = \begin{bmatrix} k_t & k_x & k_y & k_z & 0 \\ k_x\phi^2 - u\theta & k_t + \theta - k_x(\gamma-2)u & k_y u - (\gamma-1)k_x v & k_z u - (\gamma-1)k_x w & k_x(\gamma-1) \\ k_y\phi^2 - v\theta & k_x v - k_y(\gamma-1)u & k_t + \theta - k_y(\gamma-2)v & k_z v - (\gamma-1)k_y w & k_y(\gamma-1) \\ k_z\phi^2 - w\theta & k_x w - k_z(\gamma-1)u & k_y w - k_z(\gamma-2)v & k_t + \theta - k_z(\gamma-2)w & k_z(\gamma-1) \\ \theta[2\phi^2 - \gamma(e/\rho)] & \{k_x[\gamma(e/\rho) - \phi^2] & \{k_y[\gamma(e/\rho) - \phi^2] & \{k_z[\gamma(e/\rho) - \phi^2] & k_t + \gamma\theta \\ & -(\gamma-1)u\theta\} & -(\gamma-1)v\theta\} & -(\gamma-1)w\theta\} & \end{bmatrix}, \tag{16b}$$

where $\phi^2 = 0.5(\gamma-1)(u^2+v^2+w^2)$, $\theta = k_x u + k_y v + k_z w$, and, for example, to obtain $\hat{A}$, $k = \xi$.

The similarity transformations for the Jacobian matrices $\hat{A}$, $\hat{B}$, and $\hat{C}$ are

$$A = T_\xi \hat{A}_\xi T_\xi^{-1}, \qquad B = T_\eta \hat{A}_\eta T_\eta^{-1}, \qquad C = T_\zeta \hat{A}_\zeta T_\zeta^{-1}, \tag{17a}$$

where

$$\begin{aligned}\hat{A}_\xi &= D[U, U, U, U + c(\xi_x^2 + \xi_y^2 + \xi_z^2)^{1/2}, U - c(\xi_x^2 + \xi_y^2 + \xi_z^2)^{1/2}],\\ \hat{A}_\eta &= D[V, V, V, V + c(\eta_x^2 + \eta_y^2 + \eta_z^2)^{1/2}, V - c(\eta_x^2 + \eta_y^2 + \eta_z^2)^{1/2}],\\ \hat{A}_\zeta &= D[W, W, W, W + c(\zeta_x^2 + \zeta_y^2 + \zeta_z^2)^{1/2}, W - c(\zeta_x^2 + \zeta_y^2 + \zeta_z^2)^{1/2}].\end{aligned} \tag{17b}$$

with

$$T_k = \begin{bmatrix} \bar{k}_x & \bar{k}_y & \bar{k}_z \\ \bar{k}_x u & \bar{k}_y u - \bar{k}_z \rho & \bar{k}_z u + \bar{k}_y \rho \\ \bar{k}_x v + \bar{k}_z \rho & \bar{k}_y v & \bar{k}_z v - \bar{k}_x \rho \\ \bar{k}_x w - \bar{k}_y \rho & \bar{k}_y w + \bar{k}_x \rho & \bar{k}_z w \\ \left[\bar{k}_x \dfrac{\phi^2}{(\gamma-1)}\right. & \left[\bar{k}_y \dfrac{\phi^2}{(\gamma-1)}\right. & \left[\bar{k}_z \dfrac{\phi^2}{(\gamma-1)}\right. \\ \left. + \rho(\bar{k}_z v - \bar{k}_y w)\right] & \left. + \rho(\bar{k}_x w - \bar{k}_z u)\right] & \left. + \rho(\bar{k}_y u - \bar{k}_x v)\right] \end{bmatrix}$$

$$\begin{bmatrix} \alpha & \alpha \\ \alpha(u + \bar{k}_x c) & \alpha(u - \bar{k}_x c) \\ \alpha(v + \bar{k}_y c) & \alpha(v - \bar{k}_y c) \\ \alpha(w + \bar{k}_z c) & \alpha(w - \bar{k}_z c) \\ \left\{\alpha\left[\dfrac{\phi^2 + c^2}{(\gamma-1)} + c\bar{\theta}\right]\right\} & \left\{\alpha\left[\dfrac{\phi^2 + c^2}{(\gamma-1)} - c\bar{\theta}\right]\right\} \end{bmatrix}, \quad (17c)$$

and

$$T_k^{-1} = \begin{bmatrix} \left[\bar{k}_x\left(1 - \dfrac{\phi^2}{c^2}\right)\right. & \bar{k}_x(\gamma-1)\, uc^{-2} & \left[\bar{k}_z\rho^{-1}\right. \\ \left. - \rho^{-1}(\bar{k}_z v - \bar{k}_y w)\right] & & \left. + \bar{k}_x(\gamma-1)\, vc^{-2}\right] \\ \left[\bar{k}_y\left(1 - \dfrac{\phi^2}{c^2}\right)\right. & [-\bar{k}_z\rho^{-1} & \bar{k}_y(\gamma-1)\, vc^{-2} \\ \left. - \rho^{-1}(\bar{k}_x w - \bar{k}_z u)\right] & \left. + \bar{k}_y(\gamma-1)\, uc^{-2}\right] & \\ \left[\bar{k}_z\left(1 - \dfrac{\phi^2}{c^2}\right)\right. & [\bar{k}_y\rho^{-1} & [-\bar{k}_x\rho^{-1} \\ \left. - \rho^{-1}(\bar{k}_y u - \bar{k}_x v)\right] & \left. + \bar{k}_z(\gamma-1)\, uc^{-2}\right] & \left. + \bar{k}_z(\gamma-1)\, vc^{-2}\right] \\ \beta(\phi^2 - c\bar{\theta}) & \beta[\bar{k}_x c - (\gamma-1)u] & \beta[\bar{k}_y c - (\gamma-1)v] \\ \beta(\phi^2 + c\bar{\theta}) & -\beta[\bar{k}_x c + (\gamma+1)u] & -\beta[\bar{k}_y c + (\gamma-1)v] \end{bmatrix}$$

$$\begin{bmatrix} [-\bar{k}_y\rho^{-1} + \bar{k}_x(\gamma-1)\, wc^{-2}] & -\bar{k}_x(\gamma-1)\, c^{-2} \\ [\bar{k}_x\rho^{-1} + \bar{k}_y(\gamma-1)\, wc^{-2}] & -\bar{k}_y(\gamma-1)\, c^{-2} \\ \bar{k}_z(\gamma-1)\, wc^{-2} & -\bar{k}_z(\gamma-1)\, c^{-2} \\ \beta[\bar{k}_z c - (\gamma-1)w] & \beta(\gamma-1) \\ -\beta[\bar{k}_z c + (\gamma-1)w] & \beta(\gamma-1) \end{bmatrix}, \quad (17d)$$

where $\bar{\theta} = \bar{k}_x u + \bar{k}_y v + \bar{k}_z w$ and, for example, $\bar{k}_x = k_x/(k_x^2 + k_y^2 + k_z^2)^{1/2}$, etc.

Following the same procedure as in the two-dimensional development, that is, inserting the identities Eqs. (17) into Eqs. (16) and moving the matrices T_k outside the spatial difference operators, we obtain the diagonal form

$$T_\xi^n(I + h\delta_\xi \hat{A}_\xi^n)\,\hat{N}(I + h\delta_\eta \hat{A}_\eta^n)\,\hat{P}(I + h\delta_\zeta \hat{A}_\zeta^n)(T_\zeta^{-1})^n\,\Delta\hat{q}^n = \hat{R}^n, \tag{18a}$$

with $\hat{N} = T_\xi^{-1}T_\eta$, $\hat{N}^{-1} = T_\eta^{-1}T_\xi$, $\hat{P} = T_\eta^{-1}T_\zeta$, $\hat{P}^{-1} = T_\zeta^{-1}T_\eta$, where

$$T_k^{-1}T_l = \begin{bmatrix} m_1 & m_2 & m_3 & -\mu m_4 & \mu m_4 \\ -m_2 & m_1 & m_4 & \mu m_3 & -\mu m_3 \\ -m_3 & -m_4 & m_1 & -\mu m_2 & \mu m_2 \\ \mu m_4 & -\mu m_3 & \mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ -\mu m_4 & \mu m_3 & -\mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix}$$

$$m_1 = \bar{k}_x\bar{l}_x + \bar{k}_y\bar{l}_y + \bar{k}_z\bar{l}_z, \qquad m_3 = \bar{k}_x\bar{l}_z - \bar{k}_z\bar{l}_x,$$
$$m_2 = \bar{k}_x\bar{l}_y - \bar{k}_y\bar{l}_x, \qquad m_4 = \bar{k}_y\bar{l}_z - \bar{k}_z\bar{l}_y.$$

The three-dimensional diagonal algorithm requires 448 multiplies, 265 adds, and 41 divides, a total of 754 operations. The standard algorithm requires 1281 multiplies, 1061 adds, and 18 divides, a total of 2360 operations.

3. Accuracy and Stability

A. *Steady-State Accuracy*

The spatial accuracy of the standard and diagonalized algorithm for steady-state problems (i.e., where $\Delta\hat{q}$ goes to zero) is determined by the type of differencing used in forming $\hat{R}^n$, the steady-state equation. Since the modification that produces the diagonal algorithm does not affect $\hat{R}^n$, both schemes will have the same steady-state solution (if we assume that the steady-state solution is independent of the convergence path, i.e., that the steady state is unique).

B. *Stability*

One of the advantages of using the diagonal algorithm over the standard algorithm is that steady-state solutions, with the same accuracy as for the standard algorithm, can be obtained with less computational effort and cost. The question then arises as to the stability of the diagonal algorithm. Warming and Beam [15] have presented the linear stability analysis for the standard scheme when applied to constant coefficient partial differential equations. Their analysis shows unconditional stability, but in actual practice for nonlinear problems with nonperiodic boundary conditions, stability bounds have been encountered (although much less stringent ones than the explicit stability bounds). For a particular problem, one must rely on numerical experimentation to determine the actual stability bounds. In fact, linear stability analysis for the three-dimensional delta form of the approximate factorization

algorithm shows unconditional instability. The instability, though, is a weak one and can be controlled by numerical dissipation.

For constant coefficient matrices $\hat{A}$ and $\hat{B}$, the diagonal algorithm reduces to the standard algorithm because the eigenvector matrices are also constant. Therefore, the linear stability analysis of Warming and Beam [15] also holds for the diagonal algorithm. Numerical experiments have shown identical stability characteristics for the two algorithms. Convergence rates are unaffected and time-step limitations are the same for both schemes.

C. *Time Accuracy*

For simplicity, the unsteady characteristics of the diagonal algorithm can be investigated by examining the algorithm applied to the one-dimensional fluid dynamic equations. The one-dimensional equations are

$$\partial_t \bar{q} + \partial_x \bar{E} = 0, \tag{19a}$$

where

$$\bar{q} = \begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix}, \qquad \bar{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (e+p)u \end{bmatrix}. \tag{19b}$$

The standard solution algorithm for Eqs. (19) is

$$(I + h\delta_x A^n)\,\Delta\bar{q} = -h\delta_x \bar{E}^n = \bar{R}^n \tag{20a}$$

with

$$\Delta\bar{q} = (\bar{q}^{n+1} - \bar{q}^n), \qquad h = \Delta t,$$

and

$$A^n = \left(\frac{\partial \bar{E}}{\partial \bar{q}}\right)^n = \begin{bmatrix} 0 & 1 & 0 \\ (\gamma-3)u^2/2 & -(\gamma-3)u & \gamma-1 \\ [-\gamma u(e/\rho) + (\gamma-1)u^3] & [\gamma(e/\rho) - (3/2)(\gamma-1)u^2] & \gamma u \end{bmatrix}. \tag{20b}$$

Equations (20) are first-order accurate in time, when the first-order Euler implicit scheme

$$\Delta\bar{q} = (\bar{q}^{n+1} - \bar{q}^n) = -h\partial_t \bar{q}^{n+1} + o(h^2) \tag{21}$$

and the second-order local time linearization of the flux vector

$$\bar{E}^{n+1} = \bar{E}^n + A^n(\bar{q}^{n+1} - \bar{q}^n) + o(h^2) \tag{22}$$

are used.

The diagonalized form of Eqs. (20) are

$$X(I + h\delta_x \Lambda_A^n)(X^{-1})^n \Delta\bar{q} = \bar{R}^n, \tag{23a}$$

where

$$\Lambda_A = D(u, u+c, u-c) = \begin{bmatrix} u & 0 & o \\ 0 & u+c & 0 \\ 0 & 0 & u-c \end{bmatrix}, \tag{23b}$$

and

$$X = \begin{bmatrix} 1 & \alpha & \alpha \\ u & \alpha(u+c) & \alpha(u-c) \\ \dfrac{u^2}{2} & \alpha\left[\dfrac{u^2}{2} + uc + \dfrac{c^2}{(\gamma-1)}\right] & \alpha\left[\dfrac{u^2}{2} - uc + \dfrac{c^2}{(\gamma-1)}\right] \end{bmatrix},$$

$$X^{-1} = \begin{bmatrix} 1 - \dfrac{u^2}{2}(\gamma-1)c^{-2} & (\gamma-1)\,uc^{-2} & -(\gamma-1)c^{-2} \\ \beta\left[(\gamma-1)\dfrac{u^2}{2} - uc\right] & \beta[c - (\gamma-1)u] & \beta(\gamma-1) \\ \beta\left[(\gamma-1)\dfrac{u^2}{2} + uc\right] & -\beta[c + (\gamma-1)u] & \beta(\gamma-1) \end{bmatrix}. \tag{23c}$$

First we note that although Eqs. (20) are in conservation-law form, Eqs. (23) are not, due to the variable coefficient X in front of the derivative δ_x. This nonconservative form is only for the unsteady part of the equations, whereas the steady-state equations (which are unaffected by the diagonalization) remain in conservation-law form. The nonconservative nature of Eqs. (23) for unsteady calculation has been demonstrated numerically for a one-dimensional shock tube problem (Section 5).

The effect of the diagonalization on the time accuracy can be examined by subtracting Eqs. (23) from Eqs. (20), which produces

$$r = h\delta_x[X^n \Lambda_A^n (X^{-1})^n \Delta\bar{q}^n] - hX^n \delta_x[\Lambda_A^n (X^{-1})^n \Delta\bar{q}^n], \tag{24}$$

where the similarity transformation $A = X\Lambda_A X^{-1}$ is used to replace A in Eqs. (20).

Chain rule for discrete differences is used on the first term of Eq. (24) giving

$$\begin{aligned} R &= h\delta_x(X^n)\,\Lambda_A^n (X^{-1})^n \Delta\bar{q}^n + hX^n \delta_x[\Lambda_A^n (X^{-1})^n \Delta\bar{q}^n] \\ &\quad - hX^n \delta_x[\Lambda_A^n (X^{-1})^n \Delta\bar{q}^n] + O(h\,\Delta\bar{q}^n\,\Delta x^p) \\ &= h\delta_x(X^n)\,\Lambda_A^n (X^{-1})^n \Delta\bar{q}^n + O(h\,\Delta\bar{q}^n\,\Delta x^p), \end{aligned} \tag{25}$$

where the chain rule produces an $O(\Delta x^p)$ error depending on the order (p) of the difference operator δ_x.

Now, by Eq. (21), $\Delta\bar{q}^n$ is $O(h)$ so that

$$r = O(h^2) + O(h^2 \Delta x^p) = O(h^2). \tag{26}$$

Therefore, the error introduced by the diagonalization is first order in time.

4. Application to Vector Processors

One important feature of the diagonal algorithm is the reduction in temporary storage needed to complete the implicit integrations. As noted in Section 2, for two-dimensional calculations a 4×4 matrix is needed at each point, a 5×5 matrix in three dimensions. Each of the implicit operators in the standard algorithm is one dimensional and, on conventional machines, where each variable is a point function, the temporary storage requirement would be 16 times the maximum number of grid points for any direction. On vector machines, the variables are vectors, and the storage requirement is multiplied by the vector length. The Illiac IV computer and the Cray 1 have vector lengths of 64; for the STAR 100C, long vectors of the order of 500 are required. Therefore, the temporary storage requirements quickly swamp the memory of the vector machines for any reasonable grid sizes.

The diagonal algorithm reduces the above requirements by 15/16 for 2-D and by 24/25 for 3-D, since the operators are scalar, not block, tridiagonals. This reduction makes the application of implicit solvers, on vector machines, more reasonable.

5. Results

Numerical experiments have been performed comparing the standard implicit algorithm with the diagonal form of the algorithm. In all cases, the two algorithms compare quite well for convergence, steady-state spatial accuracy, and numerical stability.

In Fig. 1, results of a one-dimensional shock tube calculation are shown for both

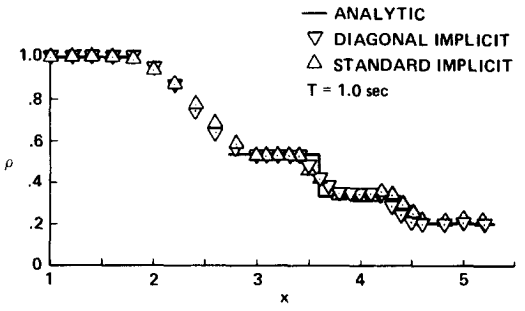

Fig. 1. Unsteady shock tube results.

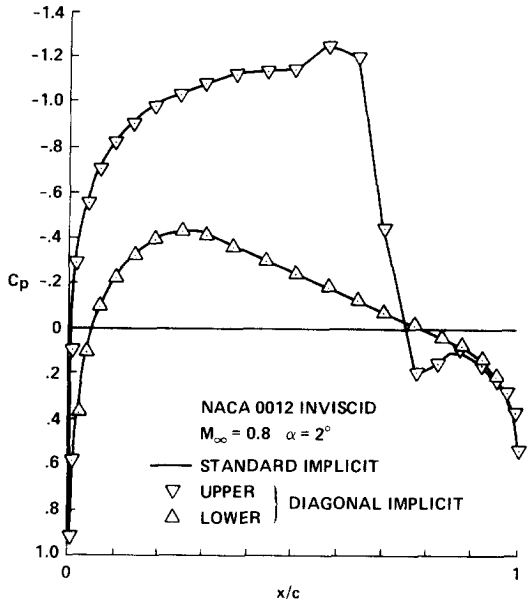

FIG. 2. Comparison for NACA 0012.

the standard and diagonal algorithm. Initially, a density ratio of 5 exists between two chambers separated by a diaphragm. After the diaphragm is removed, an expansion wave moves to the left and a contact surface and shock move to the right. As seen in Fig. 1, both methods predict the correct expansion, contact surface, and shock jumps. But in the case of the diagonalized algorithm, the shock speed is off by 5%, which is indicative of a nonconservative scheme. The scheme does predict the overall features of the flow field correctly and, therefore, seems to be applicable to unsteady flows without shocks.

A two-dimensional implicit airfoil code [6] was modified to the diagonal form and used to compare the two schemes. Geometry, application of boundary conditions, spatial accuracy, and approximations are discussed in detail in [6]. Computed pressure coefficients are shown in Figs. 2 and 3 for two airfoils at Mach number, $M_\infty = 0.8$, and angle of attack, $\alpha = 2°$. The first (Fig. 2) is for a NACA 0012 airfoil and the second (Fig. 3) for a NACA 64A410. In both cases, the solutions are fully converged and compare exactly with the solutions obtained with the standard algorithm. The convergence history for the second airfoil, the NACA 64A410, is shown in Fig. 4. The figure shows the total residual, which is the sum of the square of the residuals over all the grid points, plotted against the number of times steps N. A slight perturbation in the time step was introduced at $N = 300$. The convergence of both schemes is almost identical even after the perturbation. This confirms that the transient part of the solution is being computed by the diagonal algorithm and that it is consistent with that of the standard algorithm. In the airfoil calculations an initial

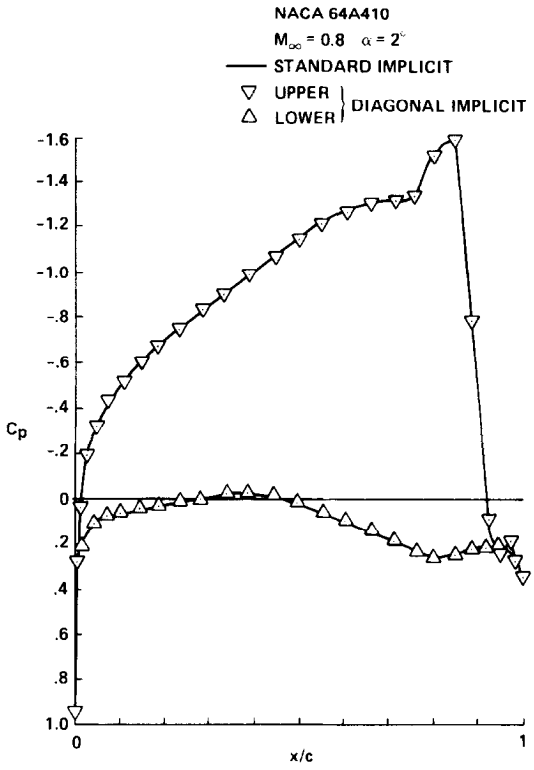

FIG. 3. Comparison for NACA 64A410.

free stream condition was used and the solutions developed to the final results. Even though the motion of the shock as it develops may be incorrect, due to the nonconservative nature of the implicit part of the diagonal algorithm, the final converged position is determined by the steady-state finite-difference equations which are identical for the diagonal and standard algorithms. The residual shown is not sensitive to the shock position since it is a global quantity. In any event, the development of the solution from arbitrary initial conditions will be almost identical for the two algorithms since the finite-difference equations are really not that much different from each other. Also, in all cases examined, there was no change in the stability characteristics when the diagonal form was used.

Computation times for the cases that were run on a CDC 7600 computer are shown in Table 1. Cases 1 and 2 are the two airfoil solutions; they have a consistent savings of over 30% in CPU time. Also included is the computation time for a two-dimensional inlet calculation [16], which shows a 34% saving in CPU time. The difference from the airfoil calculations is due to other changes in the code, such as boundary conditions, geometry, and output.

TABLE 1

Run-Time Comparisons

Case No.	Body	N	Computer time (sec)		Saving in CPU time (%)
			Standard	Diagonal	
		Two-Dimensional Calculation			
1	NACA 0012	1200	855.7	580.5	32
2	NACA 64A410	1200	992.7	673.1	32
3	2-D Inlet	200	174.7	115.0	34
		Three-Dimensional Calculation			
1	Prop-fan	50	347.8	266.0	24
2	Hemi-cyl.	50	345.0	245.4	29

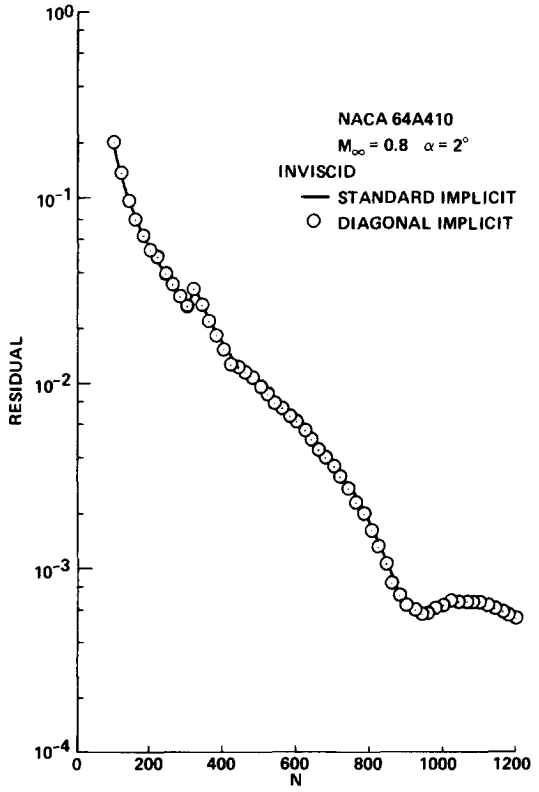

FIG. 4. Residual for NACA 64A410 computation.

Finally, preliminary calculations have been made using the three-dimensional form of the diagonalization. Here the three-dimensional code of Pulliam and Steger [14] was modified and applied to flow through a prop-fan configuration that has curved-twisted propeller blades, and to a hemisphere–cylinder configuration. Details of the hemisphere–cylinder calculations can be found in [14]. Results of the prop-fan calculations are as yet unpublished. Computation times for 50 time steps are shown in Table 1. These savings are less than what might be expected from the operation counts due to the fact that the three-dimensional program on the CDC 7600 is *I/O* (input/output) bound. Also, the three-dimensional flow field program has a lot of overhead computation for boundary conditions, metric calculations, and geometry, which lessen the relative savings from a change in the algorithm.

6. Summary

A diagonal form of an approximate-factorization implicit finite-difference algorithm has been developed. It is more efficient than the standard form yet retains many of the original accuracy and stability characteristics of the standard form. The algorithm has been applied to the Euler equations in Cartesian and general curvilinear coordinates. Results show computer time savings of up to 34% for realistic calculations. The new algorithm has an effect on the development of implicit schemes for vector computers since it reduces the temporary storage requirements of the implicit solution process.

References

1. R. M. Beam and R. F. Warming, *AIAA J.* **16** (1978), 393.
2. A. Lapidus, *J. Comput. Phys.* **2** (1967), 154.
3. H. Viviand, *Recherche Aerospatiale* **1** (1974), 65.
4. M. Vinokur, *J. Comput. Phys.* **14** (1974), 105.
5. R. M. Beam and R. F. Warming, *J. Comput. Phys.* **22** (1976), 87.
6. J. L. Steger, *AIAA J.* **16** (1978), 679.
7. J. L. Steger and P. Kutler, "Implicit Finite-Difference Procedures for the Computation of Vortex Wakes," AIAA Paper 76-385, Albuquerque, N.M., 1976.
8. E. Isaacson and H. B. Keller, "Analysis of Numberical Methods," p. 58, Wiley, New York/London/Sydney, 1969.
9. R. F. Warming, R. M. Beam, and B. J. Hyett, *Math. Comp.* **29** (1975), 1037.
10. E. Turkel, *Math. Comp.* **27** (1973), 729.
11. J. L. Steger, *Comput. Methods Appl. Mech. Engrg.* **13** (1978), 185.
12. W. R. Briley and H. McDonald, *J. Comput. Phys.* **24** (1977), 372.
13. N. N. Yanenko and V. M. Kovenja, *Soviet Math. Dokl.* **18** (1977), 260.
14. T. H. Pulliam and J. L. Steger, *AIAA J.* **18** (1980), 159.
15. R. F. Warming and R. M. Beam, *SIAM–AMS Proc.* **11** (1977).
16. D. S. Chaussee and T. H. Pulliam, "A Diagonal Form of an Implicit Approximate-Factorization Algorithm as Applied to the Calculation of the Inviscid and Viscous Supersonic Flow Fields of Two-Dimensional Inlets," AIAA Paper 80-67, Pasadena, Calif., Jan. 1980.