JOURNAL OF COMPUTATIONAL PHYSICS **133,** 129–145 (1997)
ARTICLE NO. CP975662

Pressure-Based Residual Smoothing Operators for Multistage Pseudocompressibility Algorithms

F. Sotiropoulos* and G. Constantinescu†

*School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0355;
†Iowa Institute of Hydraulic Research and Department of Civil Engineering, The University of Iowa, Iowa City, Iowa 52242-1585
E-mail: fsotirop@ce.gatech.edu

Received July 22, 1996; revised February 3, 1997

Implicit residual smoothing operators for accelerating the convergence of explicit, multistage, artificial compressibility algorithms are developed using ideas from pressure-based methods. The velocity derivatives in the continuity equation and the pressure gradient terms in the momentum equations are discretized in time implicitly. The discrete system of equations is linearized in time producing a block implicit operator which is approximately factorized and diagonalized via a similarity transformation. The so-derived diagonal operator depends only on the metrics of the geometric transformation and can, thus, be implemented in an efficient and straightforward manner. It is combined with the standard implicit residual smoothing operator and incorporated in a four-stage Runge–Kutta algorithm also enhanced with local time-stepping and multigrid acceleration. Linear stability analysis for the three-dimensional Navier–Stokes equations and calculations for laminar flows through curved square ducts and pipes demonstrate the damping properties and efficiency of the proposed approach particularly on large-aspect ratio, highly skewed meshes.

INTRODUCTION

Explicit, multistage, time-stepping schemes are becoming increasingly popular for simulating incompressible flows in conjunction with both pressure-based [1–3] and artificial compressibility (AC) algorithms [4–7]. Such schemes are simple to implement and can be readily formulated to take advantage of vector and parallel computers. Their success in modeling complex, three-dimensional flows is largely due to the second-difference implicit residual smoothing (IRS) operator which is incorporated in the basic, explicit, time-stepping scheme [8]. IRS was originally proposed by Lerat [9] to be used in conjunction with the Lax and Wendroff scheme and later was extended to multistage Runge–Kutta time-stepping by Jameson [8]. It is a differential preconditioner [10] which allows the use of larger Courant numbers, enhances robustness, and improves the overall damping properties of the time-stepping procedure. It is, therefore, of crucial importance for designing efficient multigrid algorithms whose performance relies heavily on the damping properties of the basic iterative scheme (multistage, Runge–Kutta in this case). These benefits are achieved at a relatively low computational overhead (typically less than 10% of the total CPU time per iteration) since only inversions of scalar tridiagonal matrices are required.

The objective of this paper is to exploit certain properties of the incompressible flow equations to develop a residual smoothing operator, specifically tailored for explicit, multistage AC algorithms. Such an operator must exhibit the simplicity and computational efficiency of the standard IRS operator and further enhance the damping of high-frequency errors, so that it can be used as an effective multigrid smoother for incompressible flow solutions. To construct such an operator, we explore the possibility of combining ideas from pressure-based, or pressure-Poisson (PP), methods with pseudo-compressible formulations. Consider a PP method in which the momentum equations are advanced in time explicitly [1–3]. In such an algorithm the velocity derivatives in the continuity equation and the pressure gradient terms in momentum equations are discretized implicitly. The continuity and momentum equations are subsequently combined to derive a Poisson equation for the pressure field at the new time level. Solution of this equation requires the inversion of a linear Laplacian operator, which involves only transformation metrics and the time increment. Once the pressure equation is solved, the new pressure field is used to advance the momentum equations in time. An explicit AC method [4–7], on the other hand, advances the continuity and momentum equations in time simultaneously in a coupled fashion. The main advantage of coupling the governing equations is that spatial discretization techniques (scalar and matrix-valued dissipation models, flux-difference splitting upwinding, nonlinear limited schemes, etc.) originally developed for the compressible flow equations can be readily extended to incompressible flows [6, 11, 12]. Coupling also facilitates the implementation of boundary conditions using the method of characteristics [4].

It should be emphasized that, insofar as temporal discretization is concerned, the primary difference between the

0021-9991/97 $25.00

PP and AC formulations described above is the treatment of the continuity equation. In pressure-based methods, the continuity equation is always satisfied at the implicit time level and the pressure field in the momentum equations plays the role of an implicit Lagrange multiplier [13] which projects the velocity field into the solenoidal vector space. In explicit AC formulations the pressure plays a similar role but at a different time level. As shown in [14], for instance, an AC formulation may be viewed as a PP method with the pressure-Poisson equation formulated explicitly.

To incorporate elements from pressure-based methods into coupled, explicit, AC formulations, we propose to employ the same temporal discretization for the AC system as the one used in PP formulations. That is, the velocity derivatives, in the pseudo-compressible continuity equation, and the pressure gradient terms, in the momentum equations are discretized implicitly. Rather than solving the discrete equations in a segregated fashion, however, the resulting system is linearized in time and formulated in delta form. This procedure produces a linear block implicit operator—depending only on the metrics of the geometric transformation—which is approximately factorized, using the standard Beam and Warming [15] approach. The resulting implicit operator is combined with standard implicit residual smoothing to yield a block, pressure-based residual smoothing operator for multistage AC algorithms. Implementation of such an operator is considerably more expensive (50% more CPU time per time step), as compared to the standard IRS, due to the need for inverting block matrices. A computationally more efficient operator may be derived by diagonalizing the three factors of the block operator by invoking similarity transformations of the Jacobian matrices [16]. The resulting diagonal operator requires only 20% more work than the standard IRS per time step, it can be readily implemented in existing multistage flow solvers and yields significant efficiency gains on Cartesian and highly skewed, uniform, and stretched computational meshes. Depending on the spatial discretization of the convective terms, both central and upwind residual smoothing operators can be designed using the proposed approach.

In what follows, we first present the governing equations in Cartesian coordinates and outline an approach for incorporating ideas from pressure-based methods in AC algorithms. Based on this approach, we develop block and diagonal pressure-based smoothing operators for explicit, Runge–Kutta, time-stepping schemes. The stability characteristics of the proposed operators are investigated by employing vector stability analysis for the three-dimensional Navier–Stokes equations. The efficiency of the proposed operators is evaluated by applying them to calculate three-dimensional laminar flows through curved ducts and pipes. The governing equations and the proposed method are formulated in three-dimensional, generalized curvilinear coordinates in an Appendix at the end of this paper.

THE GOVERNING EQUATIONS

For clarity, but without loss of generality, we employ the three-dimensional, incompressible Navier–Stokes equations in Cartesian coordinates to demonstrate the derivation of the proposed method. It should be emphasized, however, that the results presented in subsequent sections of this paper have been obtained using the Navier–Stokes equations in generalized, nonorthogonal curvilinear coordinates (see the Appendix for the curvilinear coordinate version of the proposed method).

The three-dimensional governing equations read in Cartesian coordinates as

$$\Gamma \frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}(E - E_V) + \frac{\partial}{\partial y}(F - F_V) + \frac{\partial}{\partial z}(G - G_V) = 0, \quad (1)$$

where

$$\begin{aligned}
\Gamma &= \mathrm{diag}(\beta, 1, 1, 1) \\
Q &= (p, u, v, w)^{\mathrm{T}} \\
E &= (u, u^2 + p, uv, uw)^{\mathrm{T}} \\
F &= (v, uv, v^2 + p, vw)^{\mathrm{T}} \\
G &= (w, uw, vw, w^2 + p)^{\mathrm{T}} \\
E_V &= \frac{1}{\mathrm{Re}}\left(0, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}\right)^{\mathrm{T}} \\
F_V &= \frac{1}{\mathrm{Re}}\left(0, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial v}{\partial z}\right)^{\mathrm{T}} \\
G_V &= \frac{1}{\mathrm{Re}}\left(0, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}, \frac{\partial w}{\partial z}\right)^{\mathrm{T}}.
\end{aligned} \quad (2)$$

In the above equations, β is a positive constant, p, u, v, and w denote the pressure and velocity components, respectively, and Re is the Reynolds number. Note that setting $\beta = 0$ in Eq. (1) produces the incompressible Navier–Stokes equations, in the form used to derive pressure-based algorithms, while $\beta \neq 0$ corresponds to the pseudo-compressible system of equations which is solved in artificial-compressibility formulations.

A PRESSURE-BASED ARTIFICIAL COMPRESSIBILITY ALGORITHM

In this section we outline a general procedure for combining ideas from pressure-based methods with explicit

AC algorithms to derive a "pressure-based" AC method. To facilitate our discussion let us split the convective flux vectors in Eq. (1) into linear and nonlinear parts,

$$E = E_L + E_N, \quad F = F_L + F_N, \quad G = G_L + G_N, \quad (3)$$

where

$$\begin{aligned} E_L &= (u, p, 0, 0)^{\mathrm{T}}, \quad & E_N &= (0, u^2, uv, uw)^{\mathrm{T}} \\ F_L &= (v, 0, p, 0)^{\mathrm{T}}, \quad & F_N &= (0, uv, v^2, vw)^{\mathrm{T}} \\ G_L &= (w, 0, 0, p)^{\mathrm{T}}, \quad & G_N &= (0, uw, vw, w^2)^{\mathrm{T}}. \end{aligned} \quad (4)$$

That is, the E_L, F_L, and G_L vectors contain the velocity terms in the continuity equation and the pressure gradient terms in the momentum equations, while the E_N, F_N, and G_N vectors contain the remaining nonlinear convective terms. As discussed below, this splitting facilitates the unified formulation of pressure-based and artificial compressibility algorithms. It was first introduced by Merkle *et al.* [17] who compared a PISO-type pressure-based method with density-based algorithms.

By employing a simple, one-stage, Euler-type, temporal integration scheme and incorporating Eqs. (3), Eq. (1) can be discretized in time as

$$\begin{aligned} \Gamma \frac{\Delta Q}{\Delta t} + \frac{\partial}{\partial x}(E_L^k + E_N^n) + \frac{\partial}{\partial y}(F_L^k + F_N^n) \\ + \frac{\partial}{\partial z}(G_L^k + G_N^n) = \left(\frac{\partial E_V}{\partial x} + \frac{\partial F_V}{\partial y} + \frac{\partial G_V}{\partial z}\right)^n, \end{aligned} \quad (5)$$

where

$$\Delta Q = Q^{n+1} - Q^n,$$

n denotes the time level and $k = n$, or $n + 1$, depending on the approach employed to integrate Eq. (5) in time. It should be noted that the Euler temporal integration scheme employed in Eq. (5) serves only to facilitate the development and presentation of the main ideas of this work. Our ultimate objective, which is pursued in a subsequent section, is to implement these ideas in conjunction with multistage, Runge–Kutta temporal integration schemes.

Equation (5) may represent either pressure-based or artificial compressibility formulations. Choosing $\beta = 1$ (or, in general, $\beta > 0$) and $k = n$, for example, produces the standard explicit AC method in which the continuity and momentum equations are coupled and advanced in time simultaneously. On the other hand, selecting $\beta = 0$ and $k = n + 1$ produces an "explicit" pressure-based formulation—the term "explicit" referring herein only to the convective and viscous terms, as the main feature of any pressure-based formulation is the implicit temporal discretization of the continuity equation and the pressure gradient terms in the momentum equations. In this case, the governing equations need to be solved in a segregated fashion. The momentum equations are substituted in the continuity equation to derive a Poisson equation which is solved to obtain the pressure at the $n + 1$ time level. The resulting pressure is subsequently employed to update the velocities using the momentum equations.

Here we propose to combine the two approaches so that the implicit treatment of the pressure and velocity-divergence terms, in pressure-based methods, and the coupled solution of the governing equations, in AC methods, are preserved. This can be accomplished by choosing $\beta = 1$ (i.e., $\Gamma = I$, where I is the identity matrix) and setting $k = n + 1$. Upon linearization, Eq. (5) reads as

$$\left[I + \Delta t\left(\frac{\partial}{\partial x}A + \frac{\partial}{\partial y}B + \frac{\partial}{\partial z}C\right)\right]\Delta Q = -\Delta t R(Q^n), \quad (6)$$

where R is the residual vector,

$$\begin{aligned} R(Q^n) = \Bigg[\frac{\partial}{\partial x}(E - E_V) + \frac{\partial}{\partial y}(F - F_V) \\ + \frac{\partial}{\partial z}(G - G_V)\Bigg]^n, \end{aligned} \quad (7)$$

and A, B, and C are Jacobian matrices:

$$A = \frac{\partial E_L}{\partial Q} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$B = \frac{\partial F_L}{\partial Q} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (8)$$

$$C = \frac{\partial G_L}{\partial Q} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Equation (6) can be factorized using the standard Beam and Warming [15] method as

$$\left[I + \Delta t \frac{\partial}{\partial x} A\right]\left[I + \Delta t \frac{\partial}{\partial y} B\right]\left[I + \Delta t \frac{\partial}{\partial z} C\right]\Delta Q = -\Delta t R(Q^n). \tag{9}$$

Equation (9) appears similar to standard implicit, approximate-factorization AC methods (see, for example, [18]). The main difference between the present formulation and such methods is obviously the fact that the operator in the left-hand side of Eq. (9) is linear, since we have treated implicitly only the divergence and pressure gradient terms.

The three factors in the left-hand side of Eq. (9) are block matrices and their inversion would require significant computational resources. To remedy this situation, we note that the A, B, and C Jacobians have real eigenvalues ($\lambda_{1,2} = 0, \lambda_3 = 1, \lambda_4 = -1$) and linearly independent eigenvectors and can, thus, be diagonalized via a similarity transformation [16]. It is also rather interesting to note that the two nonzero eigenvalues are of opposite sign and equal absolute value. This indicates that every point in the solution domain is equally influenced by upstream and downstream traveling waves. This is consistent with the elliptic character of the pressure gradient terms and continuity equation which were discretized implicitly. This observation further underscores the relation between the proposed formulation and pressure-based methods whose elliptic character is associated with the implicit solution of the Poisson equation for the pressure field.

By implementing the similarity transformation of A, B, and C in Eq. (9), $A = M^{-1}\Lambda_A M$, $B = N^{-1}\Lambda_B N$, and $C = P^{-1}\Lambda_C P$, the diagonal algorithm

$$M\left[I + \Delta t \frac{\partial}{\partial x}\Lambda_A\right]M^{-1}N\left[I + \Delta t \frac{\partial}{\partial y}\Lambda_B\right] N^{-1}P\left[I + \Delta t \frac{\partial}{\partial z}\Lambda_C\right]P^{-1}\Delta Q = -\Delta t R(Q^n) \tag{10}$$

is obtained, where $\Lambda_A = \Lambda_B = \Lambda_C = \mathrm{diag}(0, 0, 1, -1)$, and M, N, and P are the modal matrices of the A, B, and C Jacobian matrices, respectively, given by

$$M = \begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}$$

$$N = \begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & -1 & 0 & 0 \end{pmatrix} \tag{11}$$

$$P = \begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

All terms in the left-hand side of Eq. (10) are constant—in generalized curvilinear coordinates these terms involve the metrics of the geometric transformation (see Appendix)—and can, thus, be computed once and stored. Furthermore, solution of Eq. (10) requires inversions of scalar, tridiagonal matrices.

Finally, it should be noted that Eq. (10) is exactly equivalent to Eq. (9), only on Cartesian, uniformly spaced meshes. For stretched grids (see Eq. (A.11) in the Appendix), Eq. (10) is only an approximation to Eq. (9) as the modal matrices need to be locally frozen in space in order to factor them out of the spatial derivatives. The effect of this approximation on the efficiency of the proposed method on curvilinear grids is discussed in more detail in the ''Results and Discussion'' section of this paper.

EXTENSION TO MULTISTAGE AC ALGORITHMS

To facilitate our subsequent discussion, let us start by presenting the standard explicit, four-stage, Runge–Kutta procedure, enhanced with IRS, as applied to Eq. (1) (for $m = 1$ to 4),

$$Q^{(m+1)} = Q^n - \Delta t \alpha_m \overline{\Im}^{-1} R(Q^{(m)}), \tag{12}$$

where α_m are the Runge–Kutta coefficients and $\overline{\Im}$ is the standard IRS operator defined as

$$\overline{\Im}(\ ) = \left(1 - \varepsilon_x \frac{\partial^2}{\partial x^2}\right)\left(1 - \varepsilon_y \frac{\partial^2}{\partial y^2}\right)\left(1 - \varepsilon_z \frac{\partial^2}{\partial z^2}\right)(\ ), \tag{13}$$

where ε_x, ε_y, and ε_z are positive constants of order one.

Following the ideas developed in the previous section, a pressure-based Runge–Kutta scheme may be constructed

by adopting the four-stage, temporal discretization of Eq. (1) (for $m = 1, 4$),

$$Q^{(m+1)} = Q^n - \Delta t \alpha_m \left\{ \frac{\partial}{\partial x}(E_L^{(m+1)} + E_N^{(m)}) + \cdots - \frac{\partial E_V^{(m)}}{\partial x} - \cdots \right\}, \tag{14}$$

where for clarity only x-direction terms are shown. Upon linearization and approximate factorization of the resulting implicit operator, the above equation reads

$$\left[I + \alpha_m \Delta t \frac{\partial}{\partial x} A \right] \left[I + \alpha_m \Delta t \frac{\partial}{\partial y} B \right] \left[I + \alpha_m \Delta t \frac{\partial}{\partial z} C \right] \Delta Q^{(m)} = -(Q^{(m)} - Q^n) - \Delta t \alpha_m R(Q^{(m)}), \tag{15}$$

where

$$\Delta Q^{(m)} = Q^{(m+1)} - Q^{(m)}. \tag{16}$$

Equation (15) is a pressure-based, Runge–Kutta AC algorithm whose implementation requires the inversion of block matrices. A computationally efficient, diagonal version of Eq. (15)—see Eq. (10)—may be formulated as

$$M \left[I + \alpha_m \Delta t \frac{\partial}{\partial x} \Lambda_A \right] M^{-1} N \left[I + \alpha_m \Delta t \frac{\partial}{\partial y} \Lambda_B \right] N^{-1} P \left[I + \alpha_m \Delta t \frac{\partial}{\partial z} \Lambda_C \right] P^{-1} \Delta Q^{(m)} = -(Q^{(m)} - Q^n) - \Delta t \alpha_m R(Q^{(m)}), \tag{17}$$

where the modal matrices M, N, and P are given by Eqs. (11).

In general, attempts to employ either the block (Eq. (15)) or the diagonal (Eq. (17)) "pressure-based" algorithms were not successful as no converged solutions could be obtained—except for calculations carried out on orthogonal, uniform meshes where both algorithms work well. This should be attributed to the fact that the A, B, and C matrices have two zero eigenvalues (see also Eqs. (8)). Thus, no implicit smoothing is applied to two of the four governing equations which results in numerical instability. For that reason, we propose to combine the pressure-based implicit operators with the standard IRS operator (Eq. (13)) to derive block, $\Im_B$, and diagonal, $\Im_D$, smoothing operators as

$$\Im_B(\ ) = \Im_B^x \Im_B^y \Im_B^z (\ ) \tag{18}$$

with

$$\Im_B^x(\ ) = \left[I + \alpha_m \Delta t \left(\frac{\partial}{\partial x} A - \varepsilon_x \rho(A) I \frac{\partial^2}{\partial x^2} \right) \right] (\ ) \tag{19}$$

and

$$\Im_D(\ ) = M \Im_D^x M^{-1} N \Im_D^y N^{-1} P \Im_D^z P^{-1} (\ ) \tag{20}$$

with

$$\Im_D^x(\ ) = \left[I + \alpha_m \Delta t \left(\frac{\partial}{\partial x} \Lambda_A - \varepsilon_x \rho(A) I \frac{\partial^2}{\partial x^2} \right) \right] (\ ), \tag{21}$$

where $\rho(A)$ is the spectral radius of A included in Eqs. (19) and (21) to scale the second-order dissipative derivatives proportionally to the convective-like terms. Although in Cartesian coordinates the spectral radii of all three Jacobian matrices equal unity, in generalized curvilinear coordinates they depend on the contravariant metric tensor (see the Appendix) and, thus, need to be included for proper scaling. This eigenvalue scaling is similar to that used for constructing explicit, second- and/or fourth-difference artificial dissipation terms for stabilizing central-differencing schemes [19].

The pressure-based residual smoothing operators given in Eqs. (18) and (20) may be incorporated in the multistage algorithm (for $m = 1$ to 4)

$$\Im \Delta Q^{(m)} = -\Delta t \alpha_m R(Q^{(m)}) - (Q^{(m)} - Q^n), \tag{22}$$

where $\Im \equiv \Im_B$ for a block algorithm and $\Im \equiv \Im_D$ for a diagonal algorithm. Relative to the standard multistage algorithm, Eqs. (12) and (13), implementing operators $\Im_B$ or $\Im_D$ in Eq. (22) increases the CPU time per time step by approximately 50% and 20%, respectively. The subsequently presented results indicate that both operators result in significant convergence acceleration in terms of the number of time steps. In most cases, however, the block operator requires approximately the same CPU time as the standard algorithm and, therefore, is not a computationally efficient alternative. Furthermore, it is quite cumbersome to incorporate in an existing Runge–Kutta flow solver as it requires the inversion of block tridiagonal matrices. The diagonal operator, on the other hand, is shown to be very efficient and can be implemented in existing Runge–Kutta AC flow solvers with relatively minimal additional programming work. In the following sections we will pursue both algorithms in order to compare the performance of the block and diagonal operators and investigate the effects of the spatial linearization we adopted in deriving Eq. (20).

Spatial Discretization

The spatial derivatives in the residual vector, Eq. (7), are discretized by employing: (i) second-order accurate, three-point central, finite differencing for the viscous terms; (ii) central or upwind differencing for the convective terms. When central differencing is employed for the convective terms, third-order, fourth-difference artificial dissipation terms are explicitly added in the right-hand side of Eq. (7). These terms are scaled using the nonisotropic eigenvalue scaling procedure, suitable for large-aspect-ratio meshes, proposed by Martinelli [19]. Upwind approximations of the convective terms, on the other hand, are constructed by employing flux-difference splitting. In the present study a second-order accurate upwind scheme is employed. The details of the central and upwind discretization procedures can be found in [6].

The convective-like terms in the left-hand side of Eqs. (9) (or (15)) and (10) (or (17)) are discretized, depending on the discretization of the convective terms in the right-hand side, using either central or upwind differencing. Upwind approximations for these terms are constructed by employing a straightforward flux-splitting procedure since, unlike the E, F, and G flux vectors (see Eq. (2)), the E_L, F_L, and G_L flux vectors (see Eqs. (4)) are homogenous. Therefore, they can be readily split into positive and negative parts by splitting the eigenvalue matrices Λ_A, Λ_B, and Λ_C—for example, $\Lambda_A = \Lambda_A^+ + \Lambda_A^-$, where $\Lambda_A^+ = \text{diag}(0, 0, 1, 0)$ and $\Lambda_A^- = \text{diag}(0, 0, 0, -1)$—and by defining positive and negative Jacobian matrices as $A^{\pm} = M^{-1}\Lambda_A^{\pm}M$ (see the Appendix for expressions in generalized curvilinear coordinates). It should be noted that, regardless of the order of accuracy of the upwind scheme employed in the right-hand side, first-order accurate upwinding is always used for the left-hand-side terms in order to preserve the tridiagonal structure of the matrices to be inverted. Obviously such an approximation has no effect on the accuracy of the steady-state solution as the left-hand side is always driven to zero at convergence.

Multigrid Acceleration

The standard and proposed residual smoothing operators are implemented in a four-stage, Runge–Kutta algorithm—with coefficients $\alpha_m = \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1$, for $m = 1, 2, 3, 4$, respectively—enhanced with multigrid acceleration [20] and local time stepping. The multigrid method employs a V-cycle algorithm with three grid levels and has both full- and semi-coarsening capabilities [6, 7]. One iteration is performed on the finest mesh, and two and three iterations are performed on the two coarser meshes, respectively. During prolongation, the coarse grid residuals are smoothed using the standard constant-coefficient IRS operator (Eq. (13)).

Boundary Conditions

For the curved duct and pipe geometries considered in this work, boundary conditions need to be specified at inlet, outlet, solid wall, and plane-of-symmetry boundaries. Dirichlet conditions are imposed for the three velocity components at the inlet boundary and on solid walls (no-slip, no-flux). At the outflow boundary the velocity components are obtained by linear extrapolation from the interior nodes, a treatment which is equivalent to setting the diffusion terms in the streamwise direction to zero. On a plane-of-symmetry boundary, the normal velocity component is set to zero while the tangential components are computed using Neumann (zero normal derivative) symmetry conditions. The pressure at all boundaries of the computational domain is computed via linear extrapolation, except, of course, the plane-of-symmetry where its normal derivative is set to zero. The various types of boundary conditions can be formulated in a unified fashion as

$$\Delta Q_N = S^a \Delta Q_{N\pm1} + S^b \Delta Q_{N\pm2}, \tag{23}$$

where S^a and S^b are diagonal coefficient matrices, whose entries depend on the type of boundary under consideration, and N denotes a boundary node. For a z = constant plane-of-symmetry boundary, for example, $S^a = \text{diag}(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 0)$ and $S^b = -\text{diag}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$.

The procedure adopted for implementing the boundary conditions is crucial for the success of the pressure-based residual smoothing operator, particularly in nonorthogonal, highly skewed meshes. For the block algorithm, Eq. (18), boundary conditions may be implemented implicitly in a straightforward manner, similar to that used in Beam and Warming type methods [15]. Correct treatment of the boundary conditions is somewhat more involved, however, when using the diagonal algorithm (Eq. (20)). The procedures for implementing boundary conditions for both algorithms are described below.

Block Algorithm. The block operator, Eq. (18), is inverted in three successive steps:

$$\Im_B^x \Delta Q^* = R^{(m)} \tag{24a}$$

$$\Im_B^y \Delta Q^{**} = \Delta Q^* \tag{24b}$$

$$\Im_B^z \Delta Q^{(m)} = \Delta Q^{**}. \tag{24c}$$

Let us consider the x-direction sweep and assume that a boundary of the computational domain is located at $i = im$. At the node adjacent to $i = im$, Eq. (24a), discretized with a three-point differencing scheme, reads

$$C_{im-1}^- \Delta Q_{im-2}^* + C_{im-1}^o \Delta Q_{im-1}^* + C_{im-1}^+ \Delta Q_{im}^* = R^{(m)}, \tag{25}$$

where C^-, C^o, and C^+ are coefficient matrices depending on the spatial discretization scheme. Boundary conditions in the above equation may be implemented implicitly by expressing ΔQ^*_{im} in terms of ΔQ^*_{im-1} and ΔQ^*_{im-2}, using Eq. (23)

$$\begin{aligned}(C^-_{im-1} + C^+_{im-1}S^b)\,\Delta Q^*_{im-2} \\ + (C^o_{im-1} + C^+_{im-1}S^a)\,\Delta Q^*_{im-1} = R^{(m)}.\end{aligned} \tag{26}$$

Similar procedures may be adopted for the sweeps in the y- and z-directions.

Diagonal Algorithm. The diagonal residual smoothing operator, Eq. (20), is inverted in the following steps:

$$\Im^x_D\,\overline{\Delta Q}^* = M^{-1}R^{(m)} \tag{27a}$$

$$\Delta Q^* = M\,\overline{\Delta Q}^* \tag{27b}$$

$$\Im^y_D\,\overline{\Delta Q}^{**} = N^{-1}\,\Delta Q^* \tag{27c}$$

$$\Delta Q^{**} = N\,\overline{\Delta Q}^{**} \tag{27d}$$

$$\Im^z_D\,\overline{\Delta Q}^{(m)} = P^{-1}\,\Delta Q^{**} \tag{27e}$$

$$\Delta Q^{(m)} = P\,\overline{\Delta Q}^{(m)}. \tag{27f}$$

Inversion of the diagonal operators in Eqs. (a), (c), and (e) above requires boundary conditions for $\overline{\Delta Q}^*$, $\overline{\Delta Q}^{**}$, and $\overline{\Delta Q}^{(m)}$, respectively. Unlike the corresponding variables without overbars in the above equations—whose components represent temporal changes in pressure and velocity—the $\overline{\Delta Q}$ variables are by definition (see Eqs. (27b), (27d), and (27e) above) linear combinations of physical flow quantities. That is, Eq. (23) is not applicable in terms of $\overline{\Delta Q}$ and, therefore, boundary conditions cannot be applied in a similar fashion as discussed above for the block algorithm. A simple solution is to set $\overline{\Delta Q}^*$, $\overline{\Delta Q}^{**}$, and $\overline{\Delta Q}^{(m)}$ equal to zero at all boundary nodes. This explicit implementation, however, was found to work well only on orthogonal meshes (the square duct case discussed subsequently) but resulted in dramatic deterioration of the convergence rate on highly skewed meshes (the pipe bend case discussed below). A more rigorous, albeit also explicit, boundary condition implementation procedure may be formulated as follows:

Consider the x-sweep operator (Eq. (27a)) near the boundary $i = im$,

$$\begin{aligned}\overline{C}^-_{im-1}\,\overline{\Delta Q}^*_{im-2} + \overline{C}^o_{im-1}\,\overline{\Delta Q}^*_{im-1} + \overline{C}^+_{im-1}\,\overline{\Delta Q}^*_{im} \\ = M^{-1}_{im-1}R^{(m)}_{im-1},\end{aligned} \tag{28}$$

where the $\overline{C}^{\pm,o}_i$ coefficients depend on the spatial discretization scheme. As discussed above, the general extrapolation procedure, Eq. (23), is applicable only to ΔQ^*_{im} but not to $\overline{\Delta Q}^*_{im}$. To formulate boundary conditions for the latter, we multiply Eq. (23)—written in terms of ΔQ^*_{im}—from the left with M^{-1}_{im}, and using Eq. (27b) we obtain

$$\overline{\Delta Q}^*_{im} = M^{-1}_{im}S^a\,\Delta Q^*_{im-1} + M^{-1}_{im}S^b\,\Delta Q^*_{im-2}. \tag{29}$$

The corrections that appear in the right-hand side of this equation are approximated explicitly and set equal to the residuals at the corresponding nodes, i.e., $\Delta Q^*_i = R^{(m)}_i$, thus, yielding

$$\overline{\Delta Q}^*_{im} = M^{-1}_{im}S^aR^{(m)}_{im-1} + M^{-1}_{im}S^bR^{(m)}_{im-2}. \tag{30}$$

This relation provides a consistent explicit boundary condition for $\overline{\Delta Q}^*_{im}$. Incorporating Eq. (30) into Eq. (28), the near-boundary form of the x-operator is obtained:

$$\begin{aligned}\overline{C}^-_{im-1}\,\overline{\Delta Q}^*_{im-2} + \overline{C}^o_{im-1}\,\overline{\Delta Q}^*_{im-1} = M^{-1}_{im-1}R^{(m)}_{im-1} \\ - \overline{C}^o_{im-1}(M^{-1}_{im}S^aR^{(m)}_{im-1} + M^{-1}_{im}S^bR^{(m)}_{im-2}).\end{aligned} \tag{31}$$

Similar procedures are adopted for the y- and z-direction operators.

LINEAR STABILITY ANALYSIS

To investigate the effect of the proposed residual smoothing operators on the damping properties of the multistage iterative algorithm, we employ Von Neumann vector stability analysis for the three-dimensional Navier–Stokes equations. The equations of motion are linearized in time and space and transformed in Fourier space to derive an equation that relates the amplitude of the error at the "n + 1" time step to that at the "n" time step, $\hat{Q}^{n+1} = G\hat{Q}^n$, where $\hat{Q}$ is the error amplitude and G is the amplification matrix (4×4 matrix for the three-dimensional Navier–Stokes equations). The stability characteristics of the system of equations may be determined by examining the magnitude of the eigenvalues of G for the entire wave number range. Unstable behavior is associated with eigenvalues larger than unity. Eigenvalues less than unity, but very close to it, indicate very sluggish convergence rates, while eigenvalues approaching zero indicate rapid error damping and, thus, fast convergence.

For clarity, but without loss of generality, the stability analysis presented below is carried out by focusing only on the magnitude of the largest eigenvalue of G—i.e., the spectral radius of G—which will be denoted as the amplification factor g of the numerical scheme. For the

standard IRS and proposed multistage procedures outlined above, g is a function of

$$g = g(\omega_{x_i}; \sigma_{x_i}; \Omega_{x_i}; \alpha_m; \varepsilon_{x_i}; u_i) \quad \text{for } i = 1, 2, 3; m = 1 \text{ to } 4,$$

where, ω_{x_i} are the phase angles along each spatial direction, $\sigma_{x_i} = \Delta t/\Delta x_i$, Ω_{x_i} is the Von Neumann number ($\equiv \Delta t/(\Delta x_i^2\ \mathrm{Re})$), ε_{x_i} are the standard implicit residual smoothing coefficients (ε_x, ε_y, and ε_z) and u_i are the Cartesian velocity components (u, v, and w). When central-differencing is employed for discretizing the convective terms g also depends on ε, the coefficient of the explicitly added artificial dissipation terms. For all cases studied below, g is evaluated numerically and its contours are plotted—for given σ_{x_i}, Ω_{x_i}, α_m, ε_{x_i}, and u_i—at ω_{x_3} = const planes in terms of ω_{x_1} and ω_{x_2}.

It should be noted that since the Von Neumann method is only applicable to linear equations, the block and diagonal residual smoothing operators developed herein should be expected to exhibit identical stability characteristics. Also, our subsequent analysis is restricted to the single-grid version of the various algorithms and, thus, does not account for the effects of multigrid acceleration. This notwithstanding, however, the performance of the proposed residual smoothing operator in a multigrid framework may be readily extrapolated from a single grid stability analysis by focusing our attention at the high frequency end of the spectrum which is the only relevant frequency range for multigrid algorithms.

In what follows, we compare the stability characteristics of the standard IRS multistage algorithm, Eq. (12), with those of the proposed algorithm, Eq. (18) or (20). For each algorithm, we investigate the effect of the spatial discretization scheme by considering both central and upwind second-order accurate discretizations of the convective terms. We also examine the effect of cell aspect ratio AR—defined as $AR = \max(\lambda_{x_2}/\lambda_{x_1}, \lambda_{x_3}/\lambda_{x_1})$, where $\lambda_{x_i} \equiv \rho_i/\Delta x_i$ and ρ_i ($\equiv |u_i| + \sqrt{u_i^2 + \beta}$) denote the spectral radii of the convective Jacobian matrices (ρ_1, for example, is the spectral radius of $\partial E/\partial Q$)—by comparing the relative behavior of the two algorithms for AR = 1 and 1000. For all cases, the Courant and Von Neumann numbers, as well as the residual smoothing coefficients were selected to optimize the damping properties of each algorithm.

Figures 1 (central differencing) and 2 (upwind differencing) compare the stability characteristics of the standard and proposed residual smoothing operators for the case of uniform mesh (AR = 1). It is seen that, regardless of the spatial discretization scheme employed, the proposed operator yields substantially smaller, compared to the standard IRS operator, values for the amplification factor in the vicinity of $\omega_x = \omega_y = \pi$ for all four ω_z = constant sections shown. This trend suggests that the pressure-based operator should be very effective in damping high frequency errors, thus providing an excellent smoother for a multigrid procedure. The effect of the proposed algorithm is even more pronounced in Figs. 3 and 4 which compare the stability characteristics of two residual smoothing operators for a large-aspect-ratio case (AR = 1000). For both spatial discretization schemes, the pressure-based operator consistently produces substantially smaller amplification factors everywhere and particularly in the high frequency range of phase angles. Obviously this is a very encouraging result as it suggests that the proposed approach may be a powerful tool for accelerating the convergence on large-aspect-ratio meshes which are necessary for simulating turbulent flows of practical interest. The subsequently presented computations confirm most of the trends observed in Figs. 1 to 4.

TEST CASES AND COMPUTATIONAL DETAILS

To investigate the relative efficiency of the standard and proposed residual smoothing operators, we apply them to calculate laminar flows through strongly curved 90° bends of square and circular cross section. As mentioned above these calculations are carried out using the three-dimensional Navier–Stokes equations in generalized, curvilinear coordinates (see Appendix). For the square duct case, the Reynolds number, based on hydraulic diameter and bulk velocity, is Re = 790 and fully developed flow is specified at the entrance of the bend. For the pipe bend, on the other hand, plug flow is specified at the entrance of the bend and Re = 1100.

Figure 5 shows typical plane-of-symmetry and cross-sectional views of the computational grid for the duct and pipe geometries. Two meshes are employed to discretize the duct geometry: a uniform mesh with $61 \times 21 \times 13$ nodes (case 1a), and a stretched mesh (minimum near-wall spacing 8×10^{-4}) with $61 \times 41 \times 21$ nodes (case 1b) in the streamwise, radial, and normal directions, respectively. The pipe bend is discretized using $69 \times 41 \times 21$ grid nodes (case 2) with minimum near-wall spacing 1×10^{-4}. It should be noted that the cross-sectional mesh topology for the pipe bend (see Fig. 5) produces highly-skewed grid lines near the corners of the cross section. This topology was specifically selected to investigate the combined effects of mesh skewness and aspect ratio on the robustness and performance of the standard and proposed smoothing operators.

For all cases, calculations were carried out with the standard (Eq. 13)) and the block and diagonal versions of the proposed pressure-based (Eqs. (18) and (20)) residual smoothing operators. Each algorithm was employed in conjunction with both central (IRS_C, PBB_C, and PBD_C will denote the standard and pressure-based block and diagonal algorithms, respectively) with explicitly

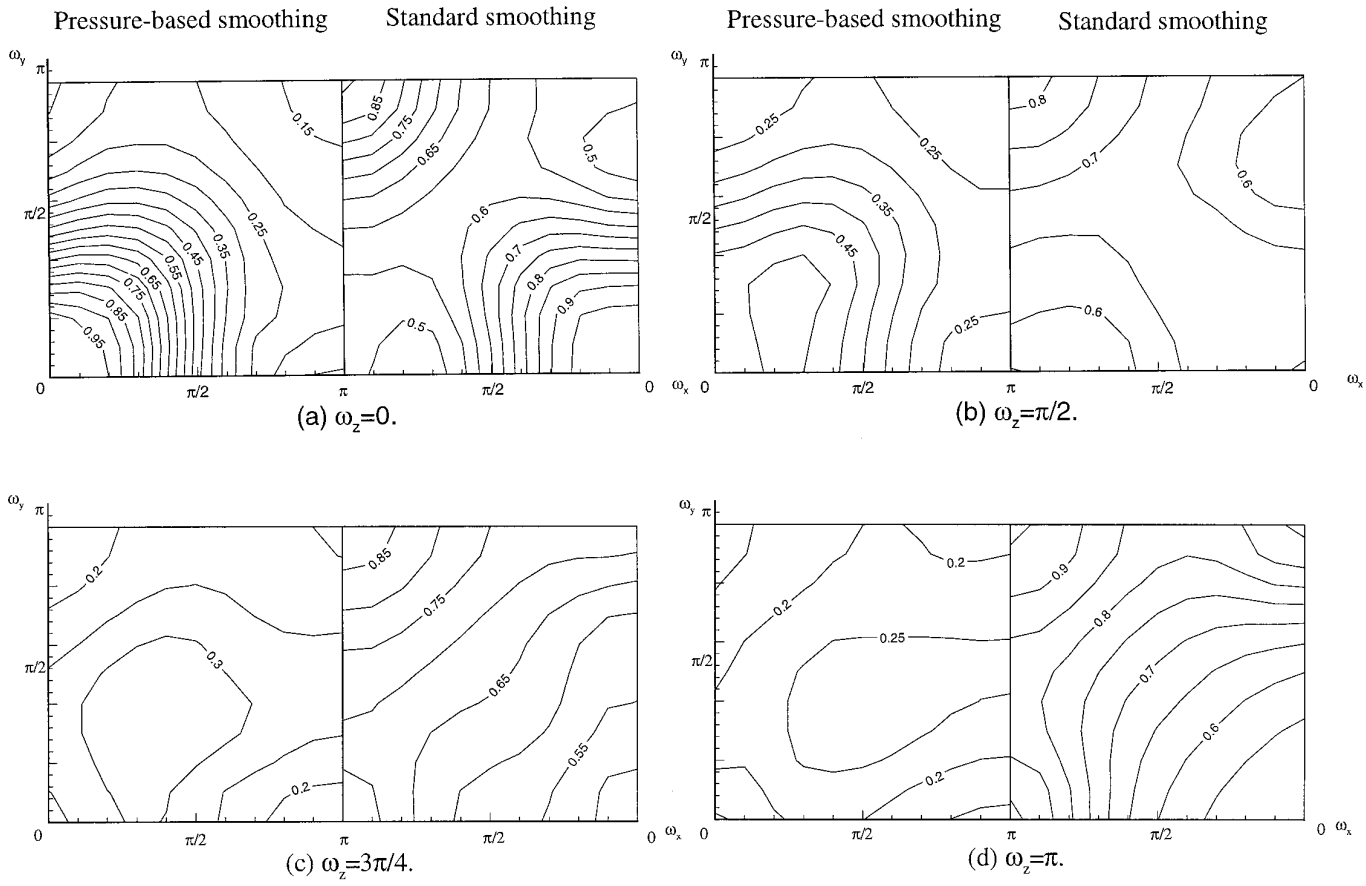

FIG. 1. Linear stability analysis (central differencing; aspect ratio AR = 1; Re = 10^3). Pressure-based smoothing (CFL = 2.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.55$, $\varepsilon = 0.06$). Standard smoothing (CFL = 1.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.35$, $\varepsilon = 0.06$).

added artificial dissipation, and second-order accurate flux-difference splitting upwind differencing (IRS_U, PBB_U, and PBD_U)—only a representative sample of these calculations, carefully selected to demonstrate the main findings of our work without confusing the reader, are shown in the subsequent figures. Case 1a is calculated with both the single-grid and multigrid versions of the various algorithms while all other cases are calculated using only multigrid acceleration.

The Courant–Friedrich–Lewis (CFL) number and smoothing coefficients for all cases were optimized via numerical experimentation with guidance from linear stability analysis. For the standard IRS algorithm the optimum CFL number was found to be CFL = 6.0, regardless of the spatial discretization scheme employed. The optimum CFL numbers for the block and diagonal pressure-based operators, on the other hand, were found to be significantly smaller, ranging between CFL = 2 to 3.5. For all cases, the optimal residual smoothing coefficients are of order one (ranging between 0.5 to 3.0, depending on the grid stretching and skewness). For the cases where central differencing was employed to discretize the convective terms, the artificial dissipation coefficient was set equal to 0.004.

To facilitate our subsequent discussion, the relative efficiency of the various algorithms is assessed, based on the work they require to reduce the residuals by four orders of magnitude (corresponding residual level of the order 10^{-5}). The CPU unit in all subsequently presented figures is defined as the CPU time required for one single grid iteration with the standard IRS algorithm. That is, the horizontal axis in these figures has been scaled to provide a direct assessment of the relative efficiency of the various algorithms in terms of actual CPU time per time step. All calculations were carried out on a HP-750 workstation using single-precision arithmetic.

RESULTS AND DISCUSSION

Figures 6 and 7 compare the performance of the various algorithms for the uniform grid case (case 1a). Figure 6 shows the convergence histories for the IRS_C, PBB_C, and PBD_C operators employed in conjunction with the single-grid version of the multistage iterative procedure.

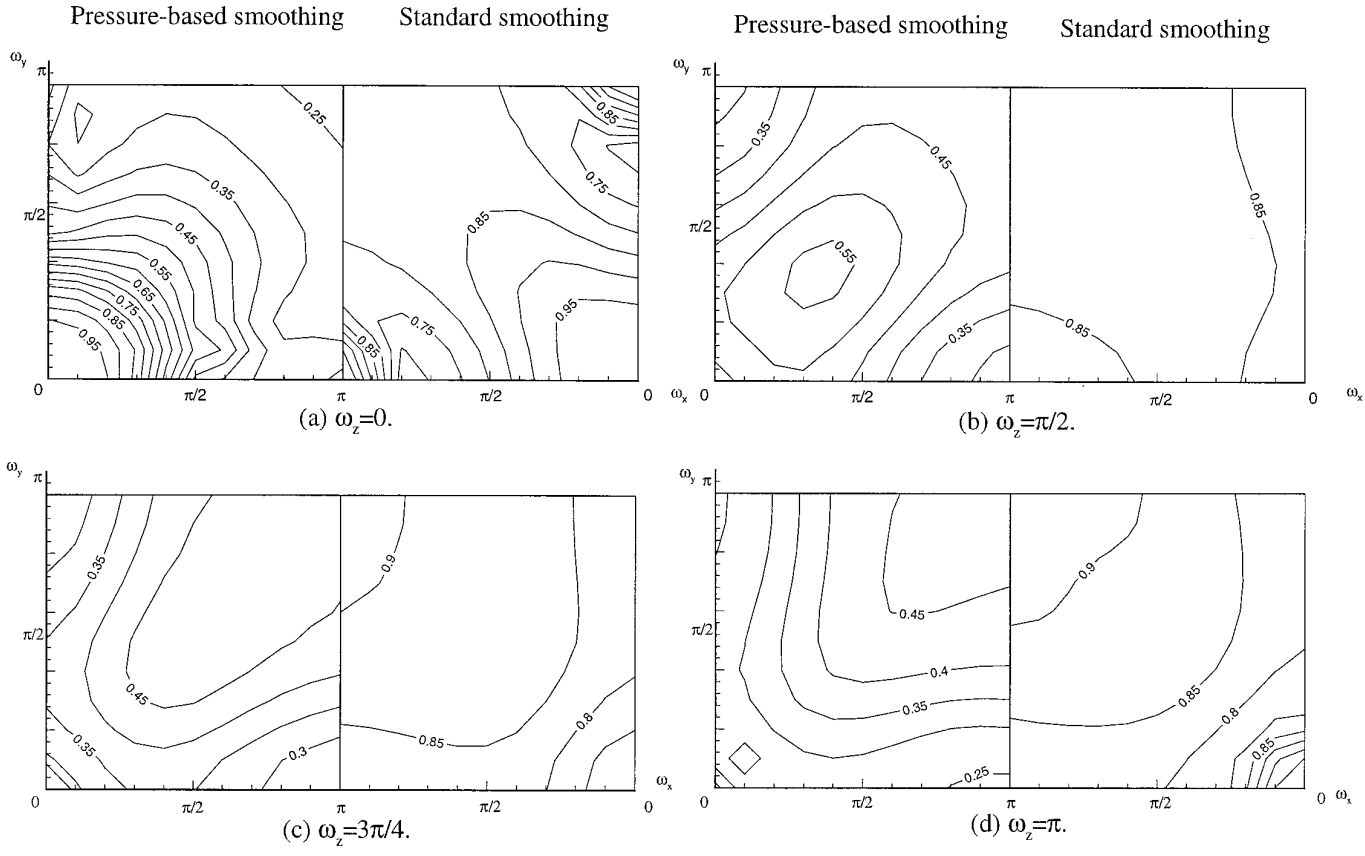

FIG. 2. Linear stability analysis (second-order upwind; aspect ratio AR = 1; Re = 10^3). Pressure-based smoothing (CFL = 3.0, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.25$). Standard smoothing (CFL = 1.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.8$).

IRS_C and PBB_C are seen to converge at almost the same rate. The diagonal version of the proposed algorithm (PBD_C), on the other hand, converges at a substantially faster rate, requiring 30% less CPU units to achieve convergence. For a uniform grid case one would expect that both the block and diagonal operators developed herein (PBB and PBD, respectively), should converge at more or less the same rate—since the grid metrics are constant and, thus, the spatial linearizations involved in deriving PBD do not introduce any error—when convergence history is plotted in terms of number of time steps. Figure 6 does confirm this observation, as the difference between the PBB and PBD convergence rates is approximately equal to the difference between the CPU time per time step required by each algorithm—that is, when plotted in terms of number of time steps, without considering the computational overhead of each operator, both algorithms converge at nearly identical rates. Figure 6 also confirms the stability analysis results for the uniform grid case (AR = 1) which, as discussed above, indicate that the proposed smoothing operator can damp very effectively almost the entire range of error frequencies (see Fig. 1). Finally, we should point out that the reasons for the oscillatory convergence exhibited by the PBB algorithm are not entirely clear. One possible reason could be the different approaches adopted for implementing boundary conditions, as discussed in a previous section.

Implementing the various smoothing operators in conjunction with a three-level, full-coarsening, V-cycle multigrid procedure has a dramatic effect on the performance of all three methods as shown in Fig. 7. All methods converge at significantly faster rates, as compared to their single-grid versions, although the general trends regarding their relative performance are similar to those observed in Fig. 6. Namely, the difference in CPU time required for convergence between PPB and PPD is approximately 25%—indicating that the two methods converge at similar rates in terms of number of multigrid cycles—and the PPD operator requires approximately 40% less CPU units than the IRS operator to achieve convergence. For this case, however, the pressure-based block algorithm (PPB) converges about 20% faster than IRS. An important observation which follows from Figs. 6 and 7 is that the relative efficiency of the pressure-based operators increases when implemented with multigrid acceleration. This trend is consistent with the stability results shown in Figs. 1 and 2,

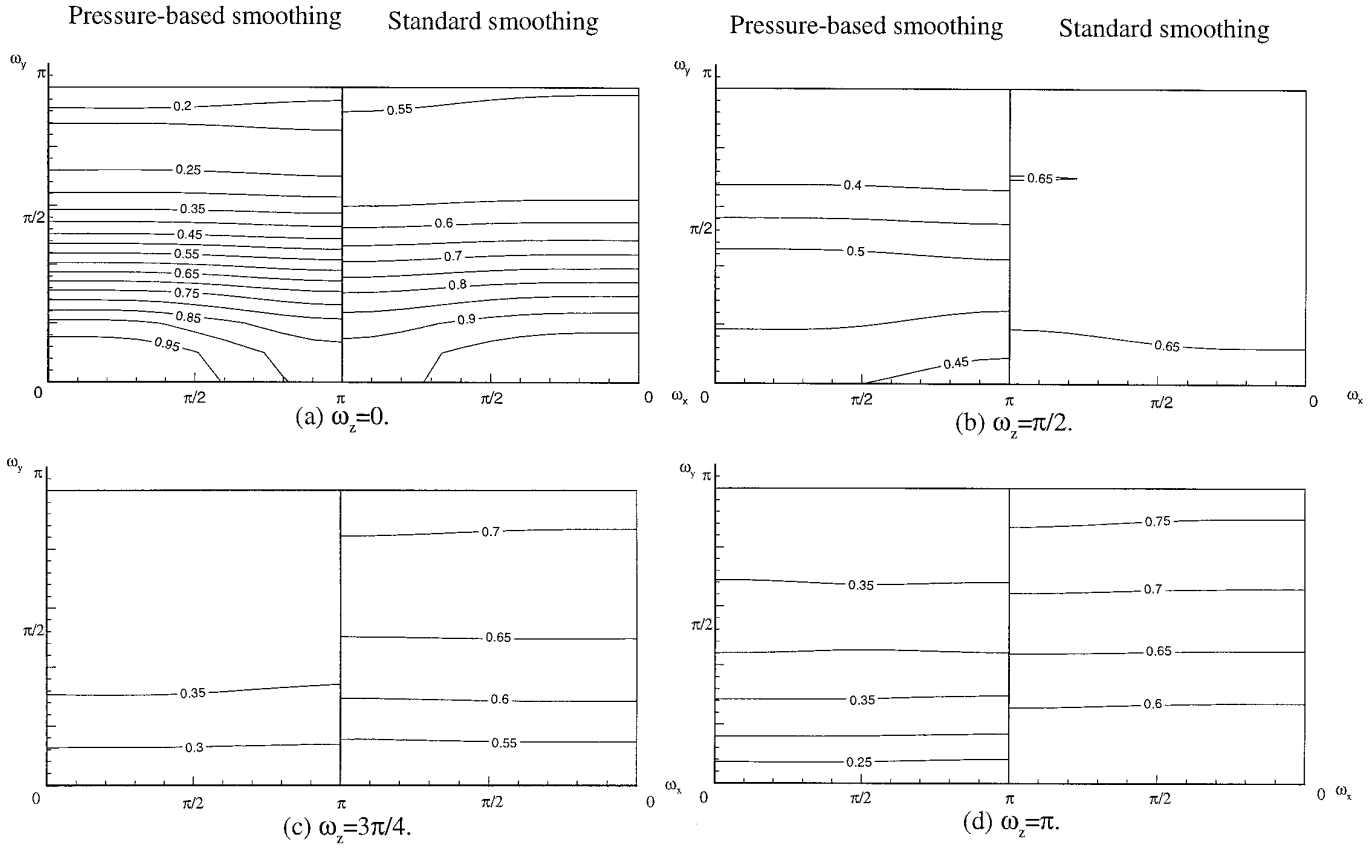

FIG. 3. Linear stability analysis (Ccntral differencing; aspect ratio AR = 1000; Re = 10^5). Pressure-based smoothing (CFL = 2.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.7$, $\varepsilon = 0.06$). Standard smoothing (CFL = 1.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.35$, $\varepsilon = 0.06$).

which clearly suggest that the pressure-based operator should be a powerful multigrid smoother due to its profound effectiveness in damping high frequency errors.

The effect of pressure-based smoothing is even more dramatic on large aspect ratio meshes. Figure 8 compares the convergence histories of various smoothing operators (IRS_C, IRS_U, PPB_C, PPB_C, PPD_C, and PPD_U) for case 1b. All calculations shown in this figure have been carried out using a three-level, full-coarsening, V-cycle multigrid procedure. Regardless of the spatial discretization scheme, the proposed operators yield substantial savings in the CPU time required for convergence. More specifically, PPD_C and PPD_U require approximately 54% and 58% less work to achieve convergence as compared to IRS_C and IRS_U, respectively. Yet another very encouraging trend is the fact that both block and diagonal pressure-based algorithms converge at nearly the same rates. This implies that in terms of number of multigrid cycles required for convergence, the block operators converge faster. This is to be expected since the spatial linearizations involved in deriving PPD introduce significant errors on nonuniform meshes. The efficiency of the diagonal operator, however, is sufficient to compensate for these errors since, as seen in Fig. 8, both diagonal operators converge, in terms of CPU units, somewhat faster than their block counterparts. Another interesting trend revealed by these comparisons is with regard to the effect of spatial discretization on the efficiency of the multigrid procedure. In general, the upwind algorithms are seen to be more efficient than their central counterparts (see also [6]). This trend, however, cannot be readily explained by the linear stability analysis results presented above which, in fact, indicate that the opposite should be true. One could attribute this discrepancy between calculations and Von Neumann analysis to the limitations of the latter which does not account for non-linearities and the influence of boundary conditions. This not withstanding, the comparisons shown in Fig. 8 do confirm the stability results insofar as the relative performance of the various algorithms on large aspect ratio meshes is concerned.

The effect of grid aspect ratio on efficiency is further underscored by the convergence histories shown in Fig. 9 (case 1b). This figure compares the performance of IRS_C and PPD_C implemented in conjunction with both full- (IRS_C_fc and PPD_C_fc) and semi-coarsening (IRS_C_sc and PPD_C_sc) multigrid procedures—coarse

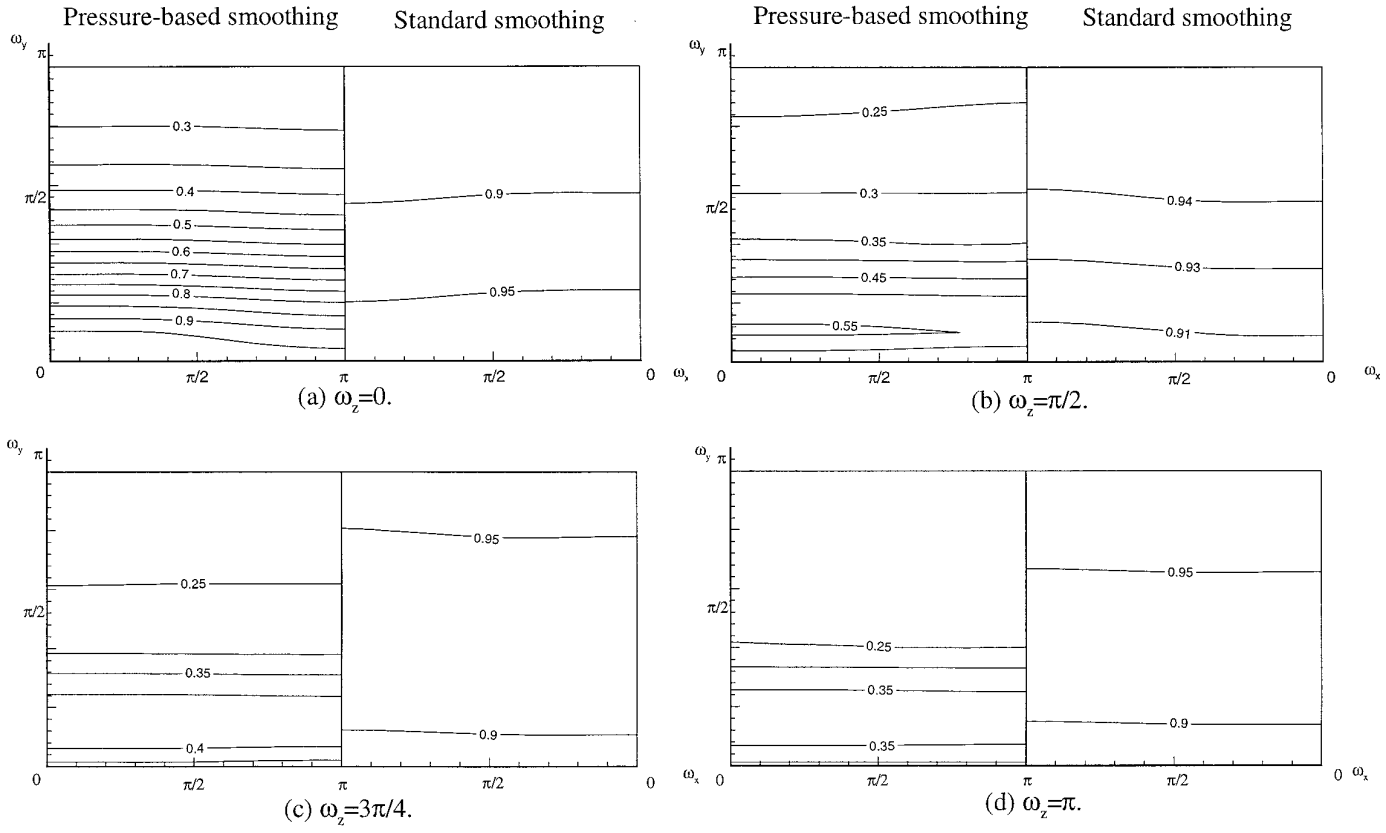

FIG. 4. Linear stability analysis (second-order upwind; aspect ratio AR = 1000; Re = 10^5). Pressure-based smoothing (CFL = 5.0, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.1$). Standard smoothing (CFL = 1.5, $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0.7$).

grids are constructed by coarsening the fine mesh along all three or only the lateral and transverse spatial directions, respectively. For the curved duct test case considered herein (case 1a), where the grid spacing in the streamwise direction is uniform, full-coarsening would tend to preserve and/or increase the fine mesh aspect ratios, while semi-coarsening would have the opposite effect. For that reason, semi-coarsening is known to enhance the performance of multigrid methods [6, 7] for duct geometries. This trend is also demonstrated in Fig. 9. The IRS_C operator with semi-coarsening converges approximately 35% faster as compared to its full-coarsening counterpart while the corresponding acceleration for the PPD_C operator is only 20%—regarding the relative efficiency of the two operators, PPD converges 45% and 57% faster than IRS for semi- and full-coarsening, respectively. The relatively smaller effect that semi-coarsening has on the efficiency of the pressure-based algorithm indicates that the proposed operator is less sensitive to and more effective in dealing with large aspect-ratio meshes, as compared to the standard IRS operator. This trend is consistent with the previously discussed results of the linear stability analysis which indicate that pressure-based smoothing enhances damping of the high frequency errors on large aspect ratio meshes. It is important to emphasize that this is a very encouraging feature of the proposed method since semi-coarsening should be expected to work well only on meshes where the distribution of grid nodes along one spatial direction is more or less uniform. This is obviously very difficult to achieve in general three-dimensional geometries of practical interest. For such cases semi-coarsening may not have the same favorable impact as for the curved duct case on the efficiency of the standard IRS operator and the benefits from adopting the proposed pressure-based smoothing may be even greater. Obviously these observations are only speculations based on the results of linear stability analysis and the present calculations with curved duct geometries. Further work is required to test the efficiency of the proposed method when applied to calculate real-life three-dimensional geometries.

The combined effect of grid stretching and skewness on the performance of the various smoothing operators is shown in Fig. 10, which compares the convergence histories of IRS_C, IRS_U, PPD_C, and PPD_U for case 2. These

results were obtained using a three-grid level, V-cycle algorithm with full-coarsening. It is seen that, regardless of the spatial discretization scheme, the proposed smoothing operator yields substantial efficiency gains, although its performance does not appear to be as impressive as for the curved duct case. After a very steep initial reduction of the residual by approximately 4.5 orders of magnitude, the convergence rates of both the PPD_U and PPD_C slow down somewhat and the residual curves exhibit consistent high frequency oscillations. Examination of the evolution, with time steps, of the computed solutions, however, indicates that the proposed pressure-based operator yields converged to plotting scale solutions in less than 400 CPU units, while the IRS-based algorithm is still converging, even after 1000 CPU units. This is shown in Fig. 11 which depicts the ''time'' evolution of pressure coefficient profiles plotted along the inner and outer walls of the pipe at the plane of symmetry for the PPD_U and IRS_U algorithms—similar results are obtained for the central-differencing versions of the two operators. Therefore, even for the highly skewed mesh employed to discretize the pipe bend (case 2) the proposed algorithm improves the convergence rate of the standard IRS method by more than 50%.

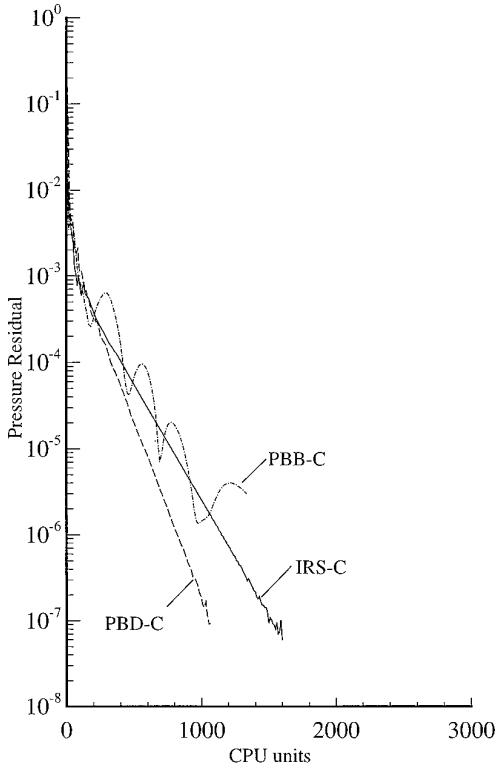

FIG. 6. Single-grid convergence histories for Case 1a (central-differencing): IRS: implicit residual smoothing, Eq. (13); PBB: pressure-based block operator, Eq. (18); PBD: pressure-based diagonal operator, Eq. (20).

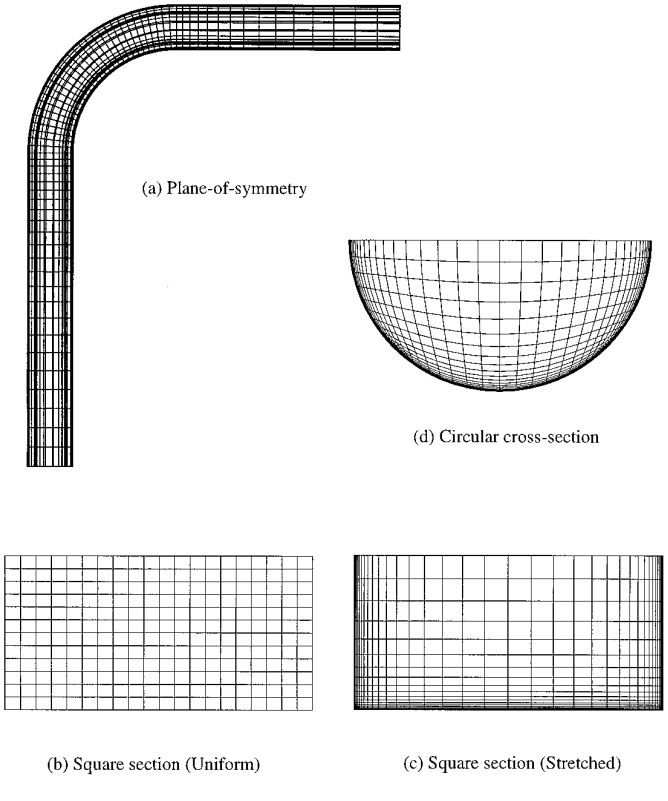

FIG. 5. Plane-of-symmetry and cross-sectional views of computational mesh: (a) mesh at the symmetry plane; (b) square cross-section (uniform mesh); (c) square cross-section (stretched mesh); (d) circular cross section.

SUMMARY AND CONCLUSIONS

A pressure-based residual smoothing operator for explicit, multistage, artificial compressibility methods was derived by incorporating a temporal discretization scheme common among pressure-Poisson methods into pseudo-compressible formulations. A similarity transformation was employed to derive a computationally efficient diagonal pressure-based operator which was implemented in a four-stage Runge–Kutta algorithm enhanced with multigrid acceleration. Both central and upwind differencing formulations were developed and investigated.

Vector stability analysis for the coupled three-dimensional Navier–Stokes equations indicates that the proposed pressure-based residual smoothing operator substantially enhances the damping of high-frequency errors on large aspect ratio meshes and can, thus, be very effective in conjunction with multigrid acceleration. This was confirmed by numerical experiments for laminar flow through strongly curved ducts and pipes using both uniform and

highly stretched, as well as orthogonal and highly skewed, meshes. Depending on the computational mesh, the proposed approach requires between 30% to 60% less work for achieving convergence, as compared to the standard residual smoothing operator and is significantly less sensitive to mesh aspect ratio. Furthermore, it is simple to implement and can be readily incorporated in existing multistage codes which utilize the constant-coefficient implicit residual smoothing. It is, therefore, a promising tool for accelerating convergence in calculations of complex, three-dimensional flows of engineering interest. The proposed approach can be extended to unsteady flows by combining it with dual time-stepping procedures.

APPENDIX

In this appendix we present the governing equations and the proposed pressure-based residual smoothing operator in three-dimensional curvilinear coordinates.

Governing Equations (repeated indices imply summation),

$$\Gamma \frac{\partial Q}{\partial t} + J \frac{\partial}{\partial \xi^j}(F^j - F_V^j) = 0, \tag{A.1}$$

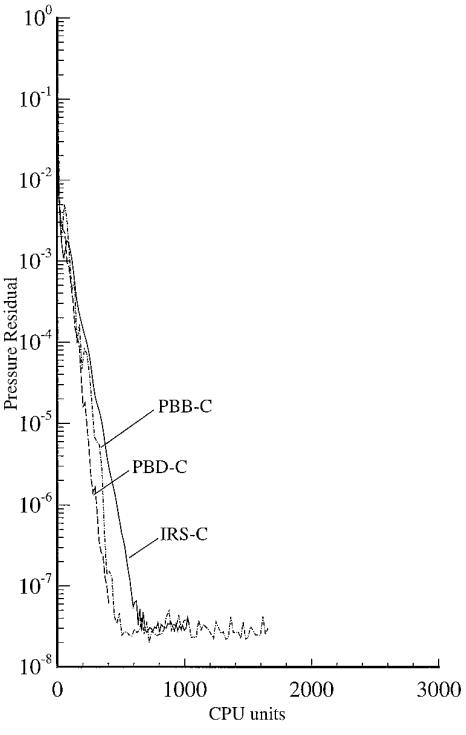

FIG. 7. Multigrid convergence histories for Case 1a (central-differencing (xxx-C); full-coarsening): IRS: implicit residual smoothing, Eq. (13); PBB: pressure-based block operator, Eq. (18); PBD: pressure-based diagonal operator, Eq. (20).

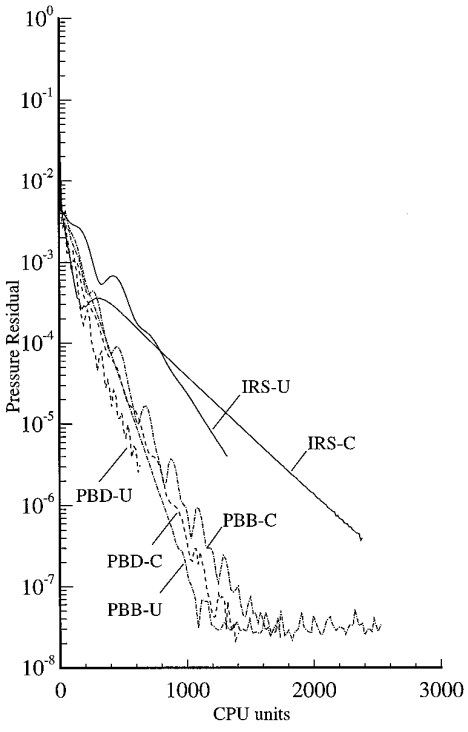

FIG. 8. Multigrid convergence histories for Case 1b (central (xxx-C) versus upwind (xxx-U) differencing; full coarsening): IRS: implicit residual smoothing, Eq. (13); PBB: pressure-based block operator, Eq. (18); PBD: pressure-based diagonal operator, Eq. (20).

where

$$Q = [p, u_1, u_2, u_3]^{\mathrm{T}}$$

$$F^j = \frac{1}{J}[U^j, u_1 U^j + p\xi^j_{x_1}, u_2 U^j + p\xi^j_{x_2}, u_3 U^j + p\xi^j_{x_3}]^{\mathrm{T}} \tag{A.2}$$

$$F_V^j = \frac{1}{J}\frac{1}{\mathrm{Re}}\left[0, g^{mj}\frac{\partial u_1}{\partial \xi^m}, g^{mj}\frac{\partial u_2}{\partial \xi^m}, g^{mj}\frac{\partial u_3}{\partial \xi^m}\right]^{\mathrm{T}}.$$

In the above equations, p is the static pressure, u_i are the Cartesian velocity components, x_i are the Cartesian coordinates, J is the Jacobian of the geometric transformation, $\xi^i_{x_j}$ are the metrics of the geometric transformation, U^j are the contravariant velocity components ($U^j = u_i \xi^i_{x_j}$), and g^{ij} are the components of the contravariant metric tensor ($g^{ij} = \xi^i_{x_k}\xi^j_{x_k}$).

The linear part, F^j_L, of the convective flux vector F^j, necessary for constructing the pressure-based operators, reads in curvilinear coordinates as

$$F^j_L = \frac{1}{J}[U^j, p\xi^j_{x_1}, p\xi^j_{x_2}, p\xi^j_{x_3}]^{\mathrm{T}}. \tag{A.3}$$

THE PROPOSED ALGORITHM. (*No* summation over repeated indices).

(i) *Block operator*

$$\Im_B(\ ) = \Im_B^{\xi^1}\Im_B^{\xi^2}\Im_B^{\xi^3}, \tag{A.4}$$

where

$$\Im_B^{\xi^j}(\ ) = \left[I + \alpha_m \Delta t \left(\frac{\partial}{\partial \xi^j} A^j - \varepsilon_{\xi^j} \rho(A^j) I \frac{\partial^2}{\partial (\xi^j)^2} \right) \right] (\ ) \tag{A.5}$$

$$A_j = \frac{\partial F_L^j}{\partial Q} = \begin{pmatrix} 0 & \xi^j_{x_1} & \xi^j_{x_2} & \xi^j_{x_3} \\ \xi^j_{x_1} & 0 & 0 & 0 \\ \xi^j_{x_2} & 0 & 0 & 0 \\ \xi^j_{x_3} & 0 & 0 & 0 \end{pmatrix} \tag{A.6}$$

$$\rho(A^j) = \sqrt{g^{jj}}. \tag{A.7}$$

(ii) *Diagonal operator*

$$\Im_D(\ ) = M_1 \Im_D^{\xi^1} M_1^{-1} M_2 \Im_D^{\xi^2} M_2^{-1} M_3 \Im_D^{\xi^3} M_3^{-1}, \tag{A.8}$$

where

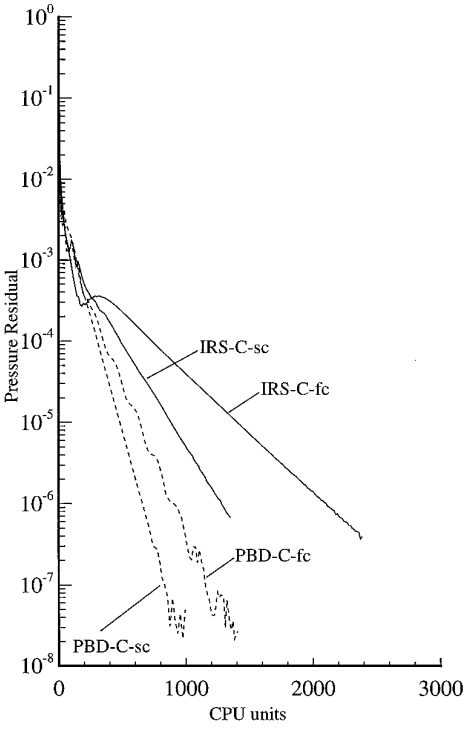

FIG. 9. Effect of grid coarsening strategies on multigrid performance (Case 1b; central differencing (xxx-C); full (fc) versus semi-coarsening (sc) strategies): IRS: implicit residual smoothing, Eq. (13); PBD: pressure-based diagonal operator, Eq. (20).

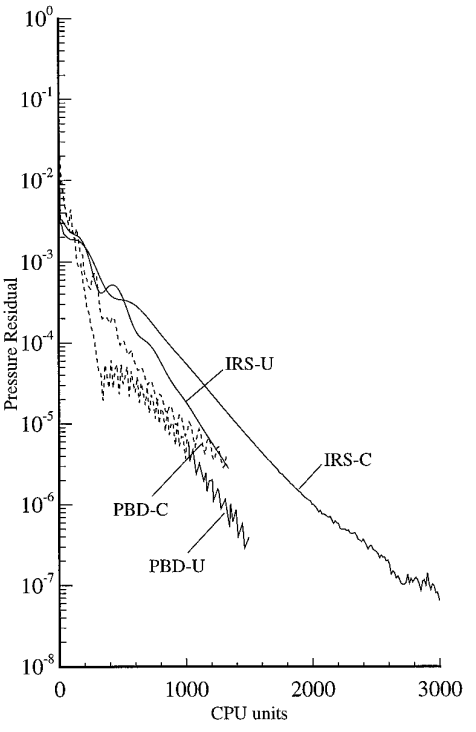

FIG. 10. Multigrid convergence histories for Case 2 (central (xxx-C) versus upwind (xxx-U) differencing; full coarsening): IRS: implicit residual smoothing, Eq. (13); PBD: pressure-based diagonal operator, Eq. (20).

$$\Im_D^{\xi^j}(\ ) = \left[I + \alpha_m \Delta t \left(\frac{\partial}{\partial \xi^j} \Lambda^j - \varepsilon_{\xi^j} \rho(A^j) I \frac{\partial^2}{\partial (\xi^j)^2} \right) \right] (\ ) \tag{A.9}$$

$$\Lambda^j = \mathrm{diag}(0, 0, \sqrt{g^{jj}}, -\sqrt{g^{jj}}) \tag{A.10}$$

$$M_j = \begin{pmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-(\xi^j_{x_2} + \xi^j_{x_3})}{s^j} & \frac{\xi^j_{x_1}(\xi^j_{x_3} - \xi^j_{x_2})}{s^j \sqrt{g^{jj}}} & \frac{\xi^j_{x_1}}{\sqrt{2}} & \frac{-\xi^j_{x_1}}{\sqrt{2}} \\ \frac{\xi^j_{x_1}}{s^j} & \frac{g^{jj} + \xi^j_{x_2}(\xi^j_{x_3} - \xi^j_{x_2})}{s^j \sqrt{g^{jj}}} & \frac{\xi^j_{x_2}}{\sqrt{2}} & \frac{-\xi^j_{x_2}}{\sqrt{2}} \\ \frac{\xi^j_{x_1}}{s^j} & \frac{-(g^{jj} - \xi^j_{x_3}(\xi^j_{x_3} - \xi^j_{x_2}))}{s^j \sqrt{g^{jj}}} & \frac{\xi^j_{x_3}}{\sqrt{2}} & \frac{-\xi^j_{x_3}}{\sqrt{2}} \end{pmatrix}$$

$$s^j = \sqrt{2g^{jj} - (\xi^j_{x_2} - \xi^j_{x_3})^2}. \tag{A.11}$$

Finally, the following matrices are necessary for constructing upwind pressure-based operators:

$$A_j^+ = \frac{1}{2} \begin{pmatrix} \sqrt{g^{jj}} & \xi^j_{x_1} & \xi^j_{x_2} & \xi^j_{x_3} \\ \xi^j_{x_1} & (\xi^j_{x_1})^2/\sqrt{g^{jj}} & \xi^j_{x_1}\xi^j_{x_2}/\sqrt{g^{jj}} & \xi^j_{x_1}\xi^j_{x_3}/\sqrt{g^{jj}} \\ \xi^j_{x_2} & \xi^j_{x_1}\xi^j_{x_2}/\sqrt{g^{jj}} & (\xi^j_{x_2})^2/\sqrt{g^{jj}} & \xi^j_{x_2}\xi^j_{x_3}/\sqrt{g^{jj}} \\ \xi^j_{x_3} & \xi^j_{x_1}\xi^j_{x_3}/\sqrt{g^{jj}} & \xi^j_{x_2}\xi^j_{x_3}/\sqrt{g^{jj}} & (\xi^j_{x_3})^2/\sqrt{g^{jj}} \end{pmatrix}$$

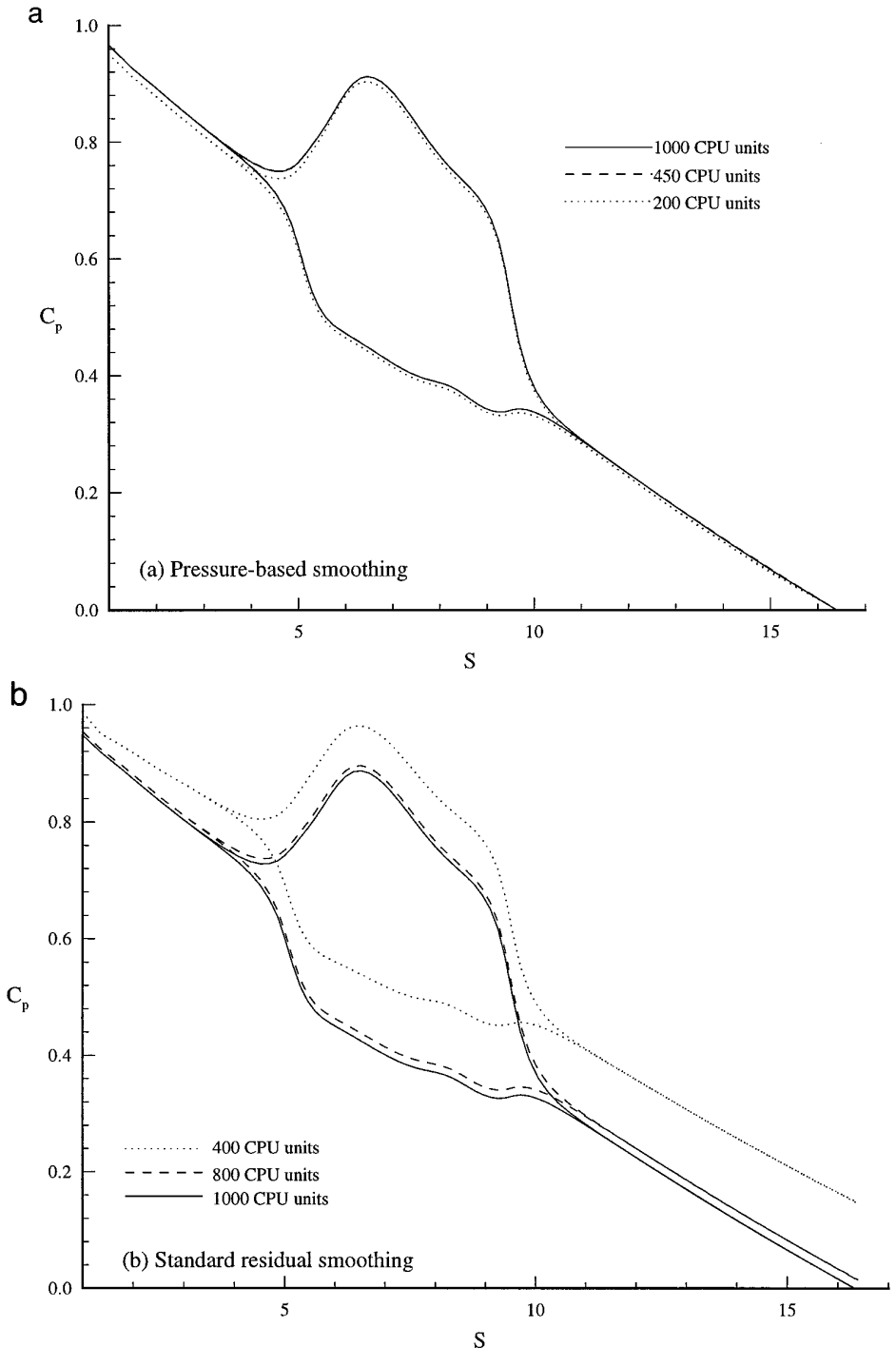

FIG. 11. Convergence history of pressure coefficient along the inner and outer walls of the pipe bend on the plane of symmetry (Case 2; upwind differencing; full-coarsening): (a) pressure-based residual smoothing, Eq. (20); (b) standard residual smoothing, Eq. (13).

$$A_j^- = A_j - A_j^+; \tag{A.12}$$

$$\Lambda_j^+ = \mathrm{diag}(0, 0, \sqrt{g^{jj}}, 0)$$

$$\Lambda_j^- = \mathrm{diag}(0, 0, 0, -\sqrt{g^{jj}}). \tag{A.13}$$

ACKNOWLEDGMENTS

This work was supported by a grant from the Electric Power Research Institute (EPRI), monitored by Mr. J. L. Tsou, and carried out under the general guidance and supervision of Professor V. C. Patel. Some calculations during the development stage of this work were carried

out on the Cray C90 at the San Diego Supercomputing Center (SDSC).

REFERENCES

1. F. Sotiropoulos and S. Adballah, A primitive variable method for the solution of 3D, incompressible, viscous flows, *J. Comput. Phys.* **103**(2), 336 (1992).
2. F. Sotiropoulos, W. J. Kim, and V. C. Patel, A computational comparison of two incompressible Navier–Stokes solvers in three-dimensional laminar flows, *Comput. & Fluids* **23**(4), 627 (1994).
3. F. Sotiropoulos and V. C. Patel, Prediction of turbulent flow through a transition duct using a second-moment closure, *AIAA J.* **32,** No. 11, 2194 (1994).
4. C. L. Merkle and P. Y.-L. Tsai, ''Application of Runge–Kutta Schemes to Incompressible Flows,'' AIAA Paper 86–0553, 1986.
5. J. Farmer, L. Martinelli, and A. Jameson, A fast multigrid method for solving the nonlinear ship wave problem, in *Proceedings, Sixth Int. Conf. on Numerical Ship Hydrodynamics,* p. 155 (National Academy Press, Washington, DC, 1994).
6. F. B. Lin and F. Sotiropoulos, Assessment of artificial dissipation models for three-dimensional incompressible flow solutions, *ASME J. Fluids Eng.,* to appear.
7. F. B. Lin and F. Sotiropoulos, Strongly-coupled multigrid method for 3-D incompressible flows using near-wall turbulence closures, *ASME J. Fluids Eng.,* to appear.
8. A. Jameson, W. Schmidt, and E. Turkel, ''Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge–Kutta Time-Stepping Schemes,'' AIAA Paper 81-1259, 1981.
9. H. Hollanders, A. Lerat, and R. Peyret, Three-dimensional calculations of transonic viscous flows by an implicit method, *AIAA J.* **23,** 1670 (1985).
10. E. Turkel, ''Review of Preconditioning Methods for Fluid Dynamics,'' NASA Contractor Report 189712, 1992.
11. C. L. Merkle and M. Athavale, ''Time-Accurate Unsteady Incompressible Flow Algorithms Based on Artificial Compressibility,'' AIAA Paper 87-1137, 1987.
12. S. E. Rogers, D. Kwak, and C. Kiris, Steady and unsteady solutions of the incompressible Navier–Stokes equations,'' *AIAA J.* **29,** No. 4, 603 (1991).
13. P. M. Gresho and R. L. Sani, On pressure boundary conditions for the incompressible Navier–Stokes equations, *Int. J. Numer. Methods Fluids* **7,** 1111 (1987).
14. R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow,* (Springer-Verlag, New York, 1983).
15. R. M. Beam and R. F. Warming, An implicit finite-difference algorithm for hyperbolic systems in conservation-law form, *J. Comput. Physics* **22,** 87 (1976).
16. T. H. Pulliam and D. S. Chaussee, A diagonal form of an implicit approximate-factorization algorithm, *J. Comput. Physics* **39,** 347 (1981).
17. C. L. Merkle, S. Venkateswaran and P. E. O. Buelow, ''The Relationship Between Pressure-Based and Density-Based Algorithms,'' AIAA Paper 92–0425, 1992.
18. D. Kwak, J. L. C. Chang, S. P. Shanks, and S. Chakravarthy, A three-dimensional incompressible Navier–Stokes flow solver using primitive variables, *AIAA J.* **24**(3), 390 (1986).
19. L. Martinelli, *Calculations of Viscous Flows with a Multigrid Method,* Ph.D. thesis, MAE Department, Princeton Univ., 1987.
20. A. Jameson, Solution of the Euler equations by a multigrid method, *Appl. Math. Comput.* **13,** 327 (1983).