COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING 86 (1991) 245–262
NORTH-HOLLAND

DISTRIBUTED MINIMAL RESIDUAL (DMR) METHOD FOR ACCELERATION OF ITERATIVE ALGORITHMS

Seungsoo LEE* and George S. DULIKRAVICH
The Pennsylvania State University, University Park, PA 16802, USA

Received 23 January 1990

A new method for enhancing the convergence rate of iterative schemes for the numerical integration of systems of partial differential equations has been developed. It is termed the Distributed Minimal Residual (DMR) method, and is based on general Krylov subspace methods. The DMR method differs from the Krylov subspace methods by the fact that the iterative acceleration factors are different from equation to equation in the system. At the same time, the DMR method can be viewed as an incomplete Newton iteration method. The DMR method has been applied to Euler equations of gasdynamics and incompressible Navier–Stokes equations. All numerical test cases were obtained using either explicit four stage Runge–Kutta or Euler implicit time integration. The DMR method was found capable of reducing the computation time by 20–80% depending on the test case. When directly compared with an implicit residual smoothing, the DMR method performed consistently better and more reliably. The formulation for the DMR method is general in nature and can be applied to explicit and implicit iterative algorithms for arbitrary systems of partial differential equations.

1. Introduction

After linearization caused by the discretization, the systems of governing equations associated with, say, fluid flows are recast into the following linear system of algebraic equations:

$$Ax = b\,, \tag{1}$$

where x is the vector of unknowns and A is an $N \times N$ matrix which depends on the discretized scheme, and is assumed to be non-singular. The matrix A is usually sparse and as N becomes larger, it is not economical to solve the system of equations directly. Instead, iterative methods are usually utilized.

The conjugate gradient (CG) method and the conjugate residual (CR) method, are widely used for approximating the solution of the system [1, 2]. Both methods give the exact solution in at most N steps in the absence of round-off errors. However, the CG method and the CR method require the matrix A to be symmetric, positive definite. A large number of generalizations of these methods applicable to systems with a non-symmetric matrix have been made. The success of the generalization of the CG and CR methods is reflected in the introduction of a series of algorithms capable of treating non-symmetric problems (Orthmin [3], Orthdir and

* Presently with Korean Agency for Defense Development, Taejon, Korea.

Orthres [4], GMRES [5, 6]). The Minimal Residual method [7] and the Generalized Nonlinear Minimal Residual method [8, 9] can be thought of as generalizations of the conjugate residual method.

In this paper, a new method of enhancing covergence rate of iterative algorithms for systems of partial differential equations is developed. The method is entitled Distributed Minimal Residual (DMR) method [10–20]. It is related to a general Krylov subspace method. Nevertheless, the DMR method differs from a Krylov subspace method in two aspects. First, the DMR method attempts to improve on a straight application of a Krylov subspace method by using a separate sequence of acceleration factors for each equation in the system. In application of the DMR method to Euler equations of inviscid gasdynamics, for example, the acceleration factors for continuity equation differ from those for the momentum equations and for energy equation. This approach requires fewer consecutive solutions to be stored than required by the GMRES method. The DMR method uses corrections from only two or three consecutive solutions for a successful application. Effectively, the DMR method periodically preconditions the system. Nevertheless, the DMR method does not involve the orthogonalization procedure which most of Krylov subspace methods utilize to reduce the number of numerical operations.

The prime objective of this paper is to develop the theory of the DMR method and to examine the effectiveness of the DMR method by applying it to different systems of partial differential equations: Euler equations of inviscid gasdynamics and incompressible Navier–Stokes equations. The Runge–Kutta time stepping method and the Euler implicit method were used as two basic iterative algorithms. A thorough comparative analysis of the performance of the DMR method [18] involves a varying number of consecutive iteration levels combined, grid clustering, etc.

2. Distributed minimal residual (DMR) method

Let us consider a system of partial differential equations that are integrated iteratively so that their residual vector at iteration level t is given by

$$\boldsymbol{R}^t = \frac{\partial \boldsymbol{E}^t}{\partial x} + \frac{\partial \boldsymbol{F}^t}{\partial y} + \frac{\partial \boldsymbol{G}^t}{\partial z}\,, \tag{2}$$

where $\boldsymbol{E}^t$, $\boldsymbol{F}^t$, $\boldsymbol{G}^t$ are the generalized flux vectors (at iteration level t) that act in the directions x, y, z, respectively. The future residual at iteration level $t+1$ is given by

$$\boldsymbol{R}^{t+1} = \frac{\partial \boldsymbol{E}^{t+1}}{\partial x} + \frac{\partial \boldsymbol{F}^{t+1}}{\partial y} + \frac{\partial \boldsymbol{G}^{t+1}}{\partial z}\,. \tag{3}$$

Assume that each component of the solution vector at iteration level $t+1$ is extrapolated from the previous M consecutive iteration levels. Then, we can say that

$$\begin{aligned}
q_1^{t+1} &= q_1^t + \omega_1^1 \Delta q_1^1 + \omega_1^2 \Delta q_1^2 + \cdots + \omega_1^M \Delta q_1^M\,,\\
q_2^{t+1} &= q_2^t + \omega_2^1 \Delta q_2^1 + \omega_2^2 \Delta q_2^2 + \cdots + \omega_2^M \Delta q_2^M\,,\\
&\vdots\\
q_L^{t+1} &= q_L^t + \omega_L^1 \Delta q_L^1 + \omega_L^2 \Delta q_L^2 + \cdots + \omega_L^M \Delta q_L^M\,.
\end{aligned} \tag{4}$$

Here, the subscripts 1, 2, . . . , L designate the particular component of the solution vector $\boldsymbol{Q}$, that is, the particular equation in the original system (2). The superscripts 1, 2, . . . , M designate the particular iteration level counting backward from the present iteration level, t. Thus, the superscript 1 means the first previous iteration level. The superscript 2 means the second previous iteration level, etc. This can be expressed in a more compact form as

$$\boldsymbol{Q}^{t+1} = \boldsymbol{Q}^{t} + \sum_{m=1}^{M} \boldsymbol{\Theta}^{m} , \tag{5}$$

where

$$\boldsymbol{\Theta}^{m} = \begin{bmatrix} \omega_1^m \Delta_1^m \\ \omega_2^m \Delta_2^m \\ \vdots \\ \omega_L^m \Delta_L^m \end{bmatrix} = \omega_1^m \begin{bmatrix} \Delta_1^m \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \omega_2^m \begin{bmatrix} 0 \\ \Delta_2^m \\ \vdots \\ 0 \end{bmatrix} + \cdots + \omega_L^m \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \Delta_L^m \end{bmatrix} . \tag{6}$$

Here, ω's are the acceleration (weighting) factors to be calculated, Δ's are the iterative corrections computed with the original non-accelerated scheme, M denotes the total number of consecutive iteration levels from which the corrections are combined.

Using Taylor series expansion in time for $\boldsymbol{R}^{t+1}$ and truncating the terms that are higher than second order in Δt, (3) becomes approximately

$$\boldsymbol{R}^{t+1} = \boldsymbol{R}^{t} + \sum_{m=1}^{M} \left[\frac{\partial}{\partial x} \boldsymbol{A}^{t} \cdot + \frac{\partial}{\partial y} \boldsymbol{B}^{t} \cdot + \frac{\partial}{\partial z} \boldsymbol{C}^{t} \cdot \right] \boldsymbol{\Theta}^{m} . \tag{7}$$

The global domain residual can be defined as

$$R^{t} = \sum_{D} \boldsymbol{R}^{t^{\mathrm{t}}} \boldsymbol{R}^{t} , \tag{8}$$

where Σ_D denotes summation over the computational domain D, and the superscript t represents transpose of a vector. In order to minimize the future global residual, R^{t+1}, the ω's are determined from the following conditions:

$$\frac{\partial R^{t+1}}{\partial \omega_r^m} = 0 . \tag{9}$$

From (8) this leads to

$$\begin{aligned} -\sum_{D} \boldsymbol{R}^{t^{\mathrm{t}}} & \left[\frac{\partial}{\partial x} \boldsymbol{A}^{t} \cdot + \frac{\partial}{\partial y} \boldsymbol{B}^{t} \cdot + \frac{\partial}{\partial z} \boldsymbol{C}^{t} \cdot \right] \frac{\partial \boldsymbol{\Theta}^{m}}{\partial \omega_r^m} \\ &= \sum_{D} \sum_{n}^{M} \left\{ \left[\frac{\partial}{\partial x} \boldsymbol{A}^{t} \cdot + \frac{\partial}{\partial y} \boldsymbol{B}^{t} \cdot + \frac{\partial}{\partial z} \boldsymbol{C}^{t} \cdot \right] \boldsymbol{\Theta}^{n} \right\}^{\mathrm{t}} \\ &\quad \cdot \left\{ \left[\frac{\partial}{\partial x} \boldsymbol{A}^{t} \cdot + \frac{\partial}{\partial y} \boldsymbol{B}^{t} \cdot + \frac{\partial}{\partial z} \boldsymbol{C}^{t} \cdot \right] \frac{\partial \boldsymbol{\Theta}^{m}}{\partial \omega_r^m} \right\} , \end{aligned} \tag{10}$$

where

$$\frac{\partial \boldsymbol{\Theta}^m}{\partial \omega_r^m} = [\Delta_p^m \delta_{pr}] \tag{11}$$

and δ_{pr} is the Kronecker delta.

However, from (6) we have that

$$\boldsymbol{\Theta}^n = \sum_q^L \omega_q^n \frac{\partial \boldsymbol{\Theta}^n}{\partial \omega_q^n} . \tag{12}$$

Noticing that $\partial \boldsymbol{\Theta}^n / \partial \omega_q^n$ is not a function of ω, it follows that

$$\begin{aligned} -\sum_D \boldsymbol{R}'^{\mathrm{t}} \left[\frac{\partial}{\partial x} \boldsymbol{A}' \cdot + \frac{\partial}{\partial y} \boldsymbol{B}' \cdot + \frac{\partial}{\partial z} \boldsymbol{C}' \cdot \right] \frac{\partial \boldsymbol{\Theta}^m}{\partial \omega_r^m} \\ = \sum_D \sum_n^M \sum_q^L \omega_q^n \left\{ \left[\frac{\partial}{\partial x} \boldsymbol{A}' \cdot + \frac{\partial}{\partial y} \boldsymbol{B}' \cdot + \frac{\partial}{\partial z} \boldsymbol{C}' \cdot \right] \frac{\partial \boldsymbol{\Theta}^n}{\partial \omega_q^n} \right\}^{\mathrm{t}} \\ \cdot \left\{ \left[\frac{\partial}{\partial x} \boldsymbol{A}' \cdot + \frac{\partial}{\partial y} \boldsymbol{B}' \cdot + \frac{\partial}{\partial z} \boldsymbol{C}' \cdot \right] \frac{\partial \boldsymbol{\Theta}^m}{\partial \omega_r^m} \right\} . \end{aligned} \tag{13}$$

Let

$$a_q^n = \left[\frac{\partial}{\partial x} \boldsymbol{A}' \cdot + \frac{\partial}{\partial y} \boldsymbol{B}' \cdot + \frac{\partial}{\partial z} \boldsymbol{C}' \cdot \right] \frac{\partial \boldsymbol{\Theta}^n}{\partial \omega_q^n} . \tag{14}$$

Then (13) becomes

$$-\sum_D \boldsymbol{R}'^{\mathrm{t}} \boldsymbol{a}_r^m = \sum_n^M \sum_q^L \sum_D \omega_q^n \boldsymbol{a}_q^{n\mathrm{t}} \boldsymbol{a}_r^m . \tag{15}$$

For simplicity, let

$$c_{qr}^{nm} = \sum_D \boldsymbol{a}_q^{n\mathrm{t}} \boldsymbol{a}_r^m \tag{16}$$

and

$$b_r^m = -\sum_D \boldsymbol{R}'^{\mathrm{t}} \boldsymbol{a}_r^m . \tag{17}$$

Then, the system of algebraic equations (15) can be written as

$$\sum_n^M \sum_q^L \omega_q^n c_{qr}^{nm} = b_r^m \tag{18}$$

or

$$\begin{bmatrix} c_{11}^{11} & c_{21}^{11} & \cdots & c_{L1}^{11} & c_{11}^{21} & \cdots & c_{L1}^{M1} \\ c_{12}^{11} & c_{22}^{11} & \cdots & c_{L2}^{11} & c_{12}^{21} & \cdots & c_{L2}^{M1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{1L}^{11} & c_{2L}^{11} & \cdots & c_{LL}^{11} & c_{1L}^{21} & \cdots & c_{LL}^{M1} \\ c_{11}^{12} & c_{21}^{12} & \cdots & c_{L1}^{12} & c_{11}^{22} & \cdots & c_{L1}^{11} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{1L}^{1M} & c_{2L}^{1M} & \cdots & c_{LL}^{1M} & c_{1L}^{11} & \cdots & c_{LL}^{MM} \end{bmatrix} \begin{bmatrix} \omega_1^1 \\ \omega_2^1 \\ \vdots \\ \omega_L^1 \\ \omega_1^2 \\ \vdots \\ \omega_L^M \end{bmatrix} = \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_L^1 \\ b_1^2 \\ \vdots \\ b_L^M \end{bmatrix}, \tag{19}$$

representing the system of $L \times M$ linear algebraic equations for the $L \times M$ optimum acceleration factors ω. For example, if we are periodically to combine corrections from $M = 2$ consecutive iteration levels to extrapolate the solution and to solve a system of $L = 4$ partial differential equations, we need to solve simultaneously $L \times M = 8$ algebraic equations for 8 values of ω. In practice [10–20] M is less than five, making this operation relatively inexpensive.

Notice that when the convergence is achieved, the b's become zero (17), thus making the ω's zero. In other words, the accuracy of the fully converged solution will not be affected by using the DMR method. Furthermore, if the matrix c_{qr}^{nm} is positive definite, it can be shown easily that the ω's minimize the global residual, R^{t+1}, at iteration level $t+1$. This fact can be used as a criterion for determining whether or not the DMR method should be applied at the particular iteration level. Using a different sequence of acceleration factors for each partial differential equation in the original system is equivalent to using a different time step for each equation or selectively preconditioning the system. The DMR method, therefore, can be understood as the combination of a preconditioning method and a Krylov subspace method. Also, we can think of the DMR method as an incomplete Newton iteration. This point can be illustrated by the following fact. When the acceleration factors vary not only from equation to equation, but also from grid point to grid point, and when we use one set of acceleration factors for each iteration level, that is, $M = 1$ in the DMR formulation, it can be shown that the DMR method is equivalent to the Newton iterations. As a result, the DMR no longer needs other schemes to provide the corrections.

The storage requirement of the DMR method for a two-dimensional problem is approximately $N \times L \times M \times (L+1) + 2 \times N \times L \times (L+1)$, while the storage requirement of the basic scheme is $6 \times N \times L + 10 \times N$ for the Runge–Kutta method and $2 \times N \times L^2 + 4 \times N \times L + 10 \times N$ for the Euler implicit method. Here, N is the number of grid points. This requirement is estimated assuming that the Jacobian matrices and metrics are evaluated once and stored. The number of operations involved in the application of the DMR method is $\mathrm{MLN}(5L+1)$ for a two-dimensional problem.

3. Application of the DMR method to Euler equations of gasdynamics

The introduction of the successful numerical algorithms such as the Euler implicit method and the explicit Runge–Kutta time stepping method made it relatively inexpensive to perform the numerical integration of the systems of partial differential equations governing compressible flows. Most of such algorithms, however, suffer from slow convergence at low Mach numbers. The reasons for this are the rapidly increased stiffness and the singular behavior of the original system of compressible flow equations at low Mach numbers. The singular behavior of the system near Mach number zero can be removed by eliminating the singularity of the system by a perturbation technique [21, 22]. The stiffness of the system at low Mach numbers can be reduced by preconditioning the system [22, 23]. The DMR method is used to alleviate the difficulty associated with the increased stiffness of the Euler equations for low Mach number compressible flows.

3.1. Euler equations for compressible flows

The Euler equations for a two-dimensional unsteady inviscid flow expressed in a generalized

non-orthogonal curvilinear coordinates (ξ, η) without body forces or heat transfer, can be written in a vector form as

$$\frac{\partial \tilde{Q}}{\partial \tau} + \frac{\partial \tilde{E}}{\partial \xi} + \frac{\partial \tilde{F}}{\partial \eta} = 0 , \tag{20}$$

where

$$\tilde{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e_0 \end{bmatrix}, \qquad \tilde{E} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho U u + p\xi_x \\ \rho U v + p\xi_y \\ (e_0 + p)U \end{bmatrix}, \qquad \tilde{F} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho V u + p\eta_x \\ \rho V v + p\eta_y \\ (e_0 + p)V \end{bmatrix}. \tag{21}$$

The subscripts x and y represent first (partial) derivatives with respect to x and y, respectively. Here τ is the time, ρ is the density, p is the thermodynamic pressure, e_0 is the total energy per unit volume, while u and v are the Cartesian velocity components along the x and y axis, respectively. J is the Jacobian determinant, $\partial(\xi, \eta)/\partial(x, y)$, while U, V are the contravariant velocity vector components defined as

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}. \tag{22}$$

3.2. Numerical algorithm

The artificial dissipation suggested by Steger and Kutler [24] was used in the form

$$D(J\tilde{Q}) = \frac{\varepsilon}{8J\,\Delta\tau}\,\nabla^4[J\tilde{Q}] , \tag{23}$$

where ∇^4 is the biharmonic differential operator in ξ, η coordinates. The residual vector $\hat{R}$ of Euler equations for compressible flow including the artificial dissipation is

$$\hat{R} = \frac{\partial \tilde{E}}{\partial \xi} + \frac{\partial \tilde{F}}{\partial \eta} + \frac{\varepsilon}{8J\,\Delta\tau}\,\nabla^4[J\tilde{Q}] . \tag{24}$$

After discretization, the governing equations become a set of ordinary differential equations, which can be integrated by the Runge–Kutta time stepping method [25].

$$\begin{aligned} &\tilde{Q}^0 = \tilde{Q}^t , \\ &\Delta\tilde{Q}^k = -\alpha_k\,\Delta\tau\,\hat{R}^{k-1} , \quad k = 1, 2, \ldots, K , \\ &\tilde{Q}^{t+1} = \tilde{Q}^t + \Delta\tilde{Q}^K , \end{aligned} \tag{25}$$

where α_k are the coefficients for each of the K stages of the Runge–Kutta scheme required to advance the solution from the iteration level t to the iteration level $t+1$. For example, $\alpha_k = 1/4,\ 1/3,\ 1/2$ and 1 for the four stage Runge–Kutta scheme.

The time steps for each direction are estimated [26] from

$$\Delta\tau_\xi = \frac{\mathrm{CFL}}{|U| + c(\xi_x^2 + \xi_y^2)^{1/2}}, \qquad \Delta\tau_\eta = \frac{\mathrm{CFL}}{|V| + c(\eta_x^2 + \eta_y^2)^{1/2}}, \tag{26}$$

where c is the local speed of sound and CFL is the Courant–Friedrichs–Lewy number. The maximum time step is given as

$$\Delta\tau = \frac{\Delta\tau_\xi \, \Delta\tau_\eta}{\Delta\tau_\xi + \Delta\tau_\eta}. \tag{27}$$

The implicit characteristic boundary procedure of Chakravarthy [27] was used, though the scheme itself is explicit. Entropy per unit mass ($s = p/\rho^\gamma$), total enthalpy per unit mass, $h_0 = (e_0 + p)/\rho$, and flow angle ($\tan(\alpha) = v/u$) are specified at the inflow boundary. For a subsonic downstream outflow boundary ($\xi = \text{constant}$), the equation corresponding to the negative eigenvalue, $U - c(\xi_x^2 + \xi_y^2)^{1/2}$, is substituted with a constant back pressure, p_b. For a solid wall boundary ($\eta = \text{constant}$), the equation corresponding to the positive eigenvalue, $V + c(\eta_x^2 + \eta_y^2)^{1/2}$, is substituted with a tangency boundary condition, $V = 0$.

Upon applying the DMR method to the system of Euler equations of gasdynamics, (14) becomes

$$a_q^m = \left[\frac{\partial}{\partial\xi} \tilde{A}^t \cdot + \frac{\partial}{\partial\eta} \tilde{B}^t \cdot + DJ \cdot \right] \frac{\partial \Theta^m}{\partial \omega_q^m}, \tag{28}$$

where $\tilde{A}$ and $\tilde{B}$ are the Jacobian matrices in the transformed coordinates

$$\tilde{A} = \frac{\partial \tilde{E}}{\partial \tilde{Q}}, \qquad \tilde{B} = \frac{\partial \tilde{F}}{\partial \tilde{Q}}. \tag{29}$$

Inclusion of the boundary points when computing a_q^m did not make any difference in the convergence rate.

3.3. Results for compressible Euler equations

A two-dimensional flow analysis code has been developed in FORTRAN according to the previous theory for Euler equations of gasdynamics. All computational results were obtained on CRAY-YMP at NAS facility using automatic vectorization.

The test case for the code was flow around a circular cylinder. The outer boundary of the computational domain was located at 20 times the radius of the cylinder. A 66×32 cell computational grid was used in this test case. The computations were performed with and without the DMR method in conjunction with the four stage Runge–Kutta (RK) scheme. The convergence histories are plotted in terms of the number of iterations and in terms of the CPU time (Fig. 1). The maximum allowable CFL number (CFL = 2.8) was used in both accelerated and non-accelerated computations. The free stream Mach number was chosen to be 0.05 which is practically an incompressible flow. The DMR method saves 60% of the total CPU time in this critical flow test case. The surface pressure coefficient (Fig. 2) matches well with the potential solution.

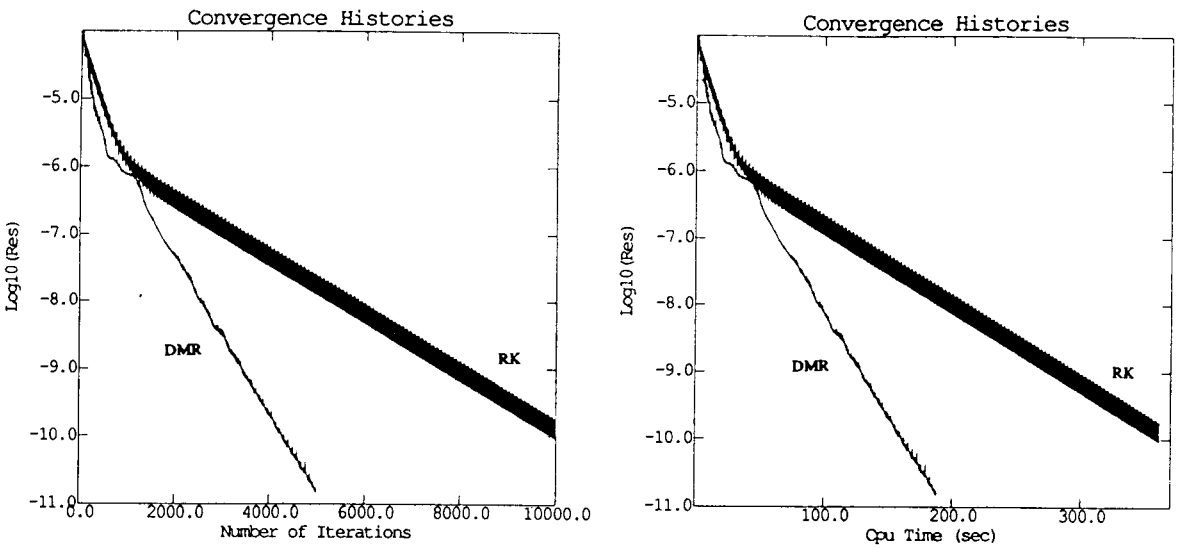

Fig. 1. Convergence histories for the inviscid flow around a circular cylinder with $M_\infty = 0.05$.

With the increase in Mach number, and especially in the transonic regime, the effectiveness of the present formulation of the DMR method deteriorates [10–20]. This brings up the question of whether it would be more appropriate to use a characteristic decomposition and apply different sequences of ω's to each characteristic variable [28]. The other equally appropriate approach would be to use the locally streamline aligned (canonical or natural) coordinate system instead of the x, y, z Cartesian system [29]. The first suggestion implies that certain variables (or information) propagates faster than the others. The second concept suggests that the same information propagates faster in certain directions than the others. A

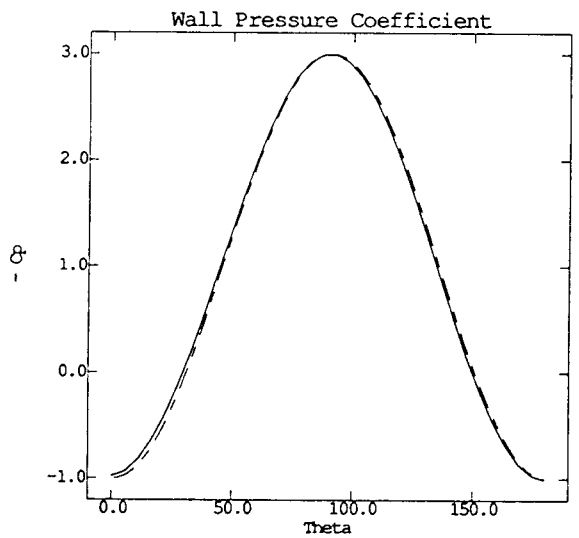

Fig. 2. Wall pressure coefficient distribution for the inviscid flow around a circular cylinder with $M_\infty = 0.05$.

combination of the two concepts seems to be the most promising avenue of future research on the general DMR methodology.

4. Application of the DMR method to incompressible Navier–Stokes equations

The main difficulties associated with the incompressible flow computations are caused by the absence of a time derivative term in the continuity equation. One of the methods for solving the incompressible Navier–Stokes equations was originated by Chorin [30]. In this concept, an artificially time dependent derivative term $\partial(p/\beta)/\partial t$ is added to the continuity equation with a user specified control parameter β. The artificial time derivative added to the continuity equation diminishes as the solution converges to its steady state. The added term forces the system to be of a mixed parabolic–hyperbolic type, which allows the use of time marching techniques. Later, Choi and Merkle [31] and Kwak et al. [32] used an Alternating Direction Implicit (ADI) method in conjunction with the artificial compressibility method.

4.1. *Incompressible Navier–Stokes equations*

The two-dimensional Navier–Stokes equations in the general non-orthogonal curvilinear coordinates ξ, η are given as

$$\partial\tilde{Q}/\partial\tau + \partial\tilde{E}/\partial\xi + \partial\tilde{F}/\partial\eta = \tilde{D}^2 . \tag{30}$$

The solution vector and the flux vectors in the transformed coordinates are given as

$$\tilde{Q} = \frac{1}{J}\begin{bmatrix} p/\beta \\ u \\ v \end{bmatrix}, \qquad \tilde{E} = \frac{1}{J}\begin{bmatrix} U \\ Uu + \xi_x p \\ Uv + \xi_y p \end{bmatrix}, \qquad \tilde{F} = \frac{1}{J}\begin{bmatrix} V \\ Vu + \eta_x p \\ Vv + \eta_y p \end{bmatrix}, \tag{31}$$

where p is the pressure. Notice that the artificial compressibility has been added in the continuity equation. The physical viscous terms in the general coordinates are given by

$$\tilde{D}^2 = \left[\frac{S}{J}\, g_{ij}(J\tilde{Q})_{,j}\right]_{,i} \tag{32}$$

where g_{ij} is the contravariant matrix tensor

$$g_{ij} = \nabla x_i' \nabla x_j' . \tag{33}$$

Here, x_i' means ξ or η depending on the index i

$$S = \frac{1}{\mathrm{Re}}\,\mathrm{diag}(0, 1, 1) , \tag{34}$$

where Re is the Reynolds number.

The Navier–Stokes equations are mixed parabolic–hyperbolic partial differential equations. According to the eigenvalue of the hyperbolic part of the equations, the Jacobian matrices in

the transformed coordinates have real eigenvalues

$$\tilde{A} = \frac{\partial \tilde{E}}{\partial \tilde{Q}} = K(U, \xi_x \cdot \xi_y), \qquad \tilde{B} = \frac{\partial \tilde{F}}{\partial \tilde{Q}} = K(V, \eta_x, \eta_y), \tag{35, 36}$$

where the matrix K is defined as

$$K(k, k_1, k_2) = \begin{bmatrix} 0 & k_1 & k_2 \\ \beta k_1 & k + k_1 u & k_2 u \\ \beta k_2 & k_1 v & k + k_2 v \end{bmatrix}. \tag{37}$$

Here, k_1 and k_2 are either ξ_x and ξ_y or η_x and η_y depending on the direction to be considered, and $k = k_1 u + k_2 v$. The eigenvalues of the matrix K are given by

$$\Lambda = \mathrm{diag}(k - c, k + c, k), \tag{38}$$

where the equivalent speed of sound, c, is given as

$$c = \sqrt{k^2 + \beta(k_1^2 + k_2^2)}. \tag{39}$$

Notice that one of the eigenvalues is negative. This means that the incompressible flow is equivalently 'subsonic' in the sense of different signs of the eigenvalues and that c will influence stiffness of the system. Thus, the direction of characteristics should be considered when applying boundary conditions.

4.2. Numerical methods

The residual vector including the fourth order artificial dissipation (23) is defined as

$$\hat{R} = \partial \tilde{E}/\partial \xi + \partial \tilde{F}/\partial \eta - \tilde{D}^2 + D. \tag{40}$$

After spatial derivative terms were discretized, the governing equations were integrated either by the explicit Runge–Kutta time-stepping algorithm (25) or by an Euler implicit method with approximate factorization [33]. To reduce the computational effort, the artificial dissipation and the viscous part of the residual are calculated only once every global time level and kept unchanged during the four stages of the Runge–Kutta scheme. This does not deteriorate the stability of the time stepping algorithm.

The Euler implicit scheme with factorization for the incompressible Navier–Stokes equations gives

$$\left[\mathrm{I} + \Delta\tau\left\{\frac{\partial}{\partial \xi}\tilde{A}\cdot - \frac{\partial}{\partial \xi}\left(\frac{Sg_{11}}{J}\frac{\partial}{\partial \xi}J\cdot\right)\right\}\right]\left[\mathrm{I} + \Delta\tau\left\{\frac{\partial}{\partial \eta}\tilde{B}\cdot - \frac{\partial}{\partial \eta}\left(\frac{Sg_{22}}{J}\frac{\partial}{\partial \eta}J\cdot\right)\right\}\right]\Delta\tilde{Q} = -\Delta\tau\hat{R}. \tag{41}$$

4.3. Time step limitations and boundary conditions

The allowable time increments of the explicit scheme are severely restricted by the stability limit, while for an implicit scheme the time step restrictions are caused by the factorization

errors. The time step is determined by considering the hyperbolic part of the system and the parabolic part of the system separately and by combining these time steps, as suggested by MacCormack and Baldwin [26]. The system becomes hyperbolic when viscosity is neglected. Then, the stability bound of the resulting system is determined by the CFL (Courant–Friedrichs–Lewy) number. The maximum allowable time steps for each of the coordinate directions are defined as

$$\Delta\tau_{h\xi} = \frac{\mathrm{CFL}}{|U| + c_\xi}, \qquad \Delta\tau_{h\eta} = \frac{\mathrm{CFL}}{|V| + c_\eta}, \tag{42}$$

so that the combined maximum time step for the hyperbolic part of the system is defined by

$$\Delta\tau_h = \Delta\tau_{h\xi}\,\Delta\tau_{h\eta}/(\Delta\tau_{h\xi} + \Delta\tau_{h\eta})\,. \tag{43}$$

When the convective part of the acceleration is neglected, the system becomes of parabolic type. The stability of the parabolic type system is dictated by the non-dimensional number σ (Von Neumann number). For each generalized coordinate direction, the maximum time steps are defined by

$$\Delta\tau_{p\xi} = \frac{\sigma\,\mathrm{Re}}{g_{11}}, \qquad \Delta\tau_{p\eta} = \frac{\sigma\,\mathrm{Re}}{g_{22}}, \tag{44}$$

and the combined maximum time step for the parabolic part is given by

$$\Delta\tau_p = \Delta\tau_{p\xi}\,\Delta\tau_{p\eta}/(\Delta\tau_{p\xi} + \Delta\tau_{p\eta})\,. \tag{45}$$

The total maximum time step is estimated conservatively as

$$\Delta\tau = \Delta\tau_h\,\Delta\tau_p/(\Delta\tau_h + \Delta\tau_p)\,. \tag{46}$$

For the explicit Runge–Kutta method, (46) was used to estimate the maximum time step. However, for the Euler implicit method, only CFL limitation was used to compute the time step, that is,

$$\Delta\tau = \Delta\tau_{h\xi}\,\Delta\tau_{h\eta}/(\Delta\tau_{h\xi} + \Delta\tau_{h\eta})\,. \tag{47}$$

It was assumed that the flow is inviscid at the inlet and at the exit plane thus causing the system of equations to be hyperbolic in time near the inlet and exit. As stated earlier, the incompressible Navier–Stokes equations have one negative eigenvalue and the rest of the eigenvalues are positive. Thus, one equation should be considered with two boundary conditions at the inlet. At the exit, two equations with one boundary condition must be applied. At the inlet u and v velocity vector components were specified, while the back pressure p was specified at the exit. Also, the flow is assumed to be locally one-dimensional at the inlet and exit boundaries in order to transform locally the equation into the characteristic form. At the solid wall, the velocity vector components were set to zero, and the surface pressure was extrapolated from the grid points next to the wall from the condition that $\partial p/\partial n = 0$.

4.4. Residual smoothing

One of the successful attempts to accelerate the convergence of the Runge–Kutta scheme is Implicit Residual Smoothing (IRS) introduced by Jameson and Baker [34]. With this method, it is possible to use much higher values of CFL. The residual is smoothed through the following equation:

$$[1 - \theta\delta_\xi^2][1 - \theta\delta_\eta^2]\hat{\boldsymbol{R}}^* = \hat{\boldsymbol{R}}\,, \tag{48}$$

where δ^2 designates the central difference operator for a respective second derivative and θ is the smoothing coefficient. Thus, when using the IRS we have to solve two scalar tri-diagonal matrices. Since their coefficients are constants, the tri-diagonal matrices are decomposed into upper and lower bi-diagonal matrices so that at every application of the IRS only forward and backward substitutions are needed to get the smoothed residual.

The application of the DMR method to incompressible Navier–Stokes equations differs from the formulation for its application to the Euler equations of gasdynamics only by the following term:

$$\boldsymbol{a}_q^m = \left[\frac{\partial}{\partial\xi}\tilde{\boldsymbol{A}}^t\cdot + \frac{\partial}{\partial\eta}\tilde{\boldsymbol{B}}^t\cdot - \boldsymbol{D}^2\boldsymbol{J}\cdot + \boldsymbol{D}\boldsymbol{J}\cdot\right]\frac{\partial\boldsymbol{\Theta}^m}{\partial\omega_q^m}\,. \tag{49}$$

4.5. Computational results for Navier–Stokes equations

A steady, laminar, viscous flow normal to a solid wall (Hiemenz flow) was the first test case. Reason for this choice of the test case is that the analytic solution to the Hiemenz flow is known [35]. The accuracy of the codes (the explicit Runge–Kutta method and the Euler implicit method) can be verified by comparing the computed solution with the analytic one.

The flow corresponding to the Reynolds number 400 based on the free stream velocity and a body dimension, R_0 of the wall was computed with and without the DMR method in conjunction with explicit and implicit codes. The computational grid consisted of 60×29 cells, and the dimensions of the computational domain were $H = R_0$ and $L = 2R_0$. In the case of an explicit Runge–Kutta (RK) method, the maximum allowable CFL number of 2.8 was used and the Von Neumann number was $\sigma = 0.4$. A small amount of the fourth order artificial dissipation was added to get a smooth solution ($\varepsilon = 0.05$). Using numerical experimentation it was found that the fastest convergence is obtained with the artificial compressibility coefficient $\beta = 2$, and that the DMR method should be applied every 10 iterations by combining 3 consecutive solutions.

The computed distribution of the wall surface velocity gradient, $\partial u/\partial y$, was compared with that of the analytic solution (Fig. 3), showing an excellent agreement. Figure 4 shows that the residual was reduced 12 orders of magnitude in 5000 iterations without the DMR method, while the same reduction in residual could be achieved in 2000 iterations with the DMR method indicating a 60% reduction in CPU time. The implicit residual smoothing was also implemented with and without the DMR method. The basic RK method gives the slowest convergence, the IRS gives faster convergence than the basic RK method, while the DMR method gave an even better convergence. The most rapid convergence in terms of the number of iterations was achieved by combining the implicit residual smoothing and the DMR method. However, the DMR method alone needed the least CPU time.

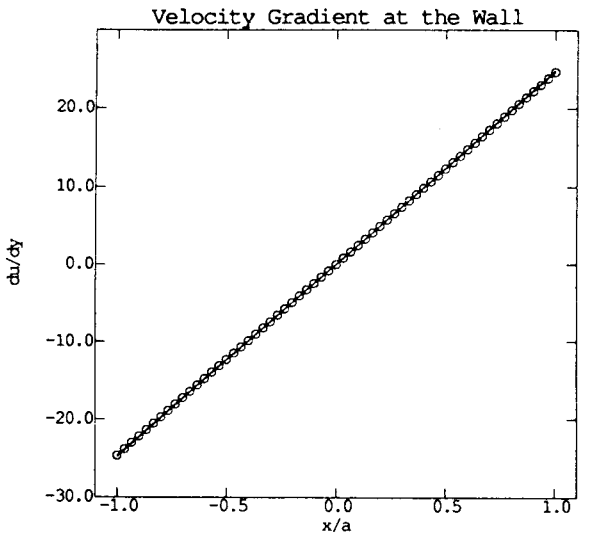

Fig. 3. Distributions of wall surface velocity gradient for Hiemenz flow (RK: solid line; analytic solution: circles).

The implicit code was also exercised for the test of the Hiemenz flow with the same flow and grid conditions as in the test case for the explicit code (Re = 400). The computed surface velocity gradient distribution was compared with the analytic solution (Fig. 5). Good agreement can be observed. A CFL number of 10 was used in this computation. Also, the fourth order artificial dissipation with $\varepsilon = 0.25$ was added. The optimal value of the artificial compressibility coefficient β was found by numerical experiments to be $\beta = 5$. When applied to the implicit Euler scheme every 5 iterations by combining 5 consecutive solutions, the DMR method was found to give the fastest convergence. Figure 6 shows that the DMR method offers 60% reduction in CPU time indicating that the DMR method can be successfully

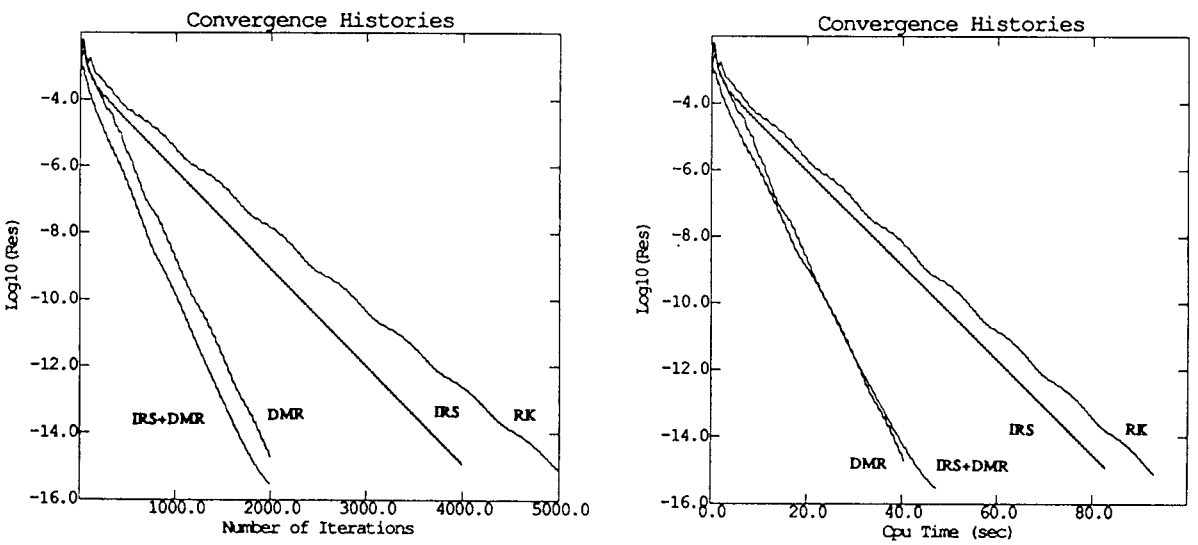

Fig. 4. Convergence histories of the RK method for Hiemenz flow with Re = 400.

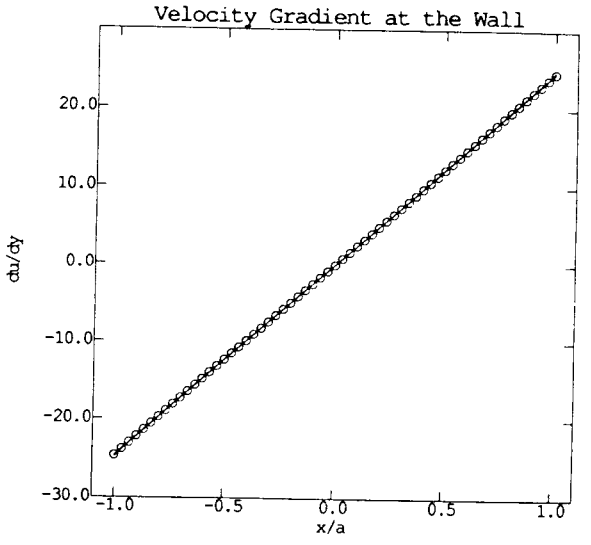

Fig. 5. Distributions of wall surface velocity gradient for Hiemenz flow (Euler implicit: solid line; analytic solution: circles).

applied to implicit methods. Our results indicate that the basic implicit code is considerably slower compared with the RK method. This should not be a general situation, since our version of the Euler implicit scheme was not vectorized. On the other hand, the RK method was fully vectorized.

The next test case was a laminar flow around a circular cylinder. The computational highly clustered grid of 66×44 cells was used. Flow with Reynolds numbers of 20 was computed with the RK method and the Euler implicit method. The CFL and Von Neumann numbers were CFL = 2.8 and $\sigma = 0.4$, respectively, for the RK method, and CFL = 10 was used for the Euler

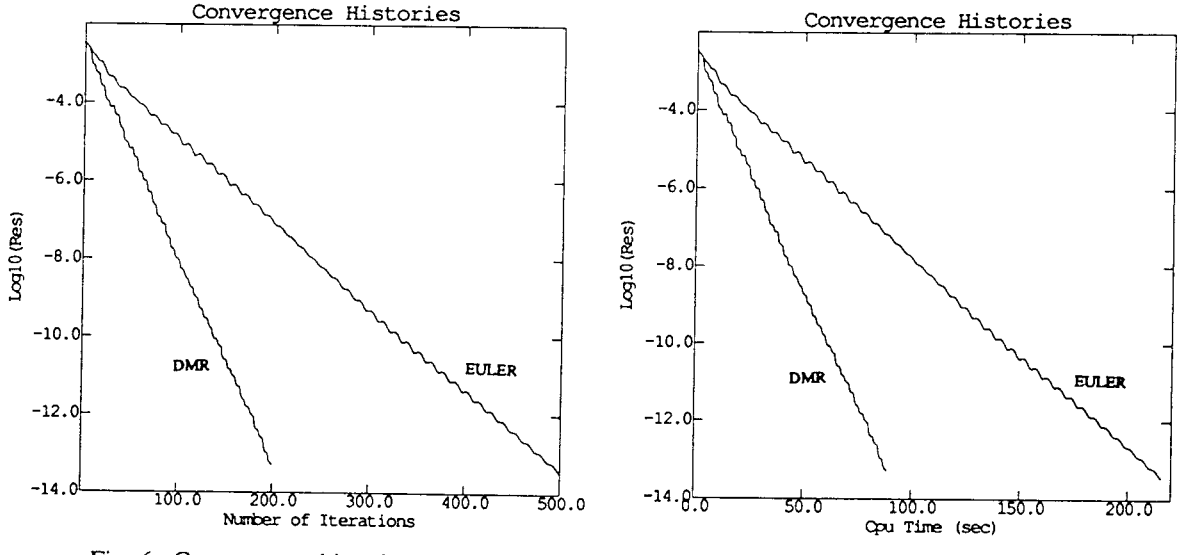

Fig. 6. Convergence histories of the Euler implicit method for Hiemenz flow with Re = 400.

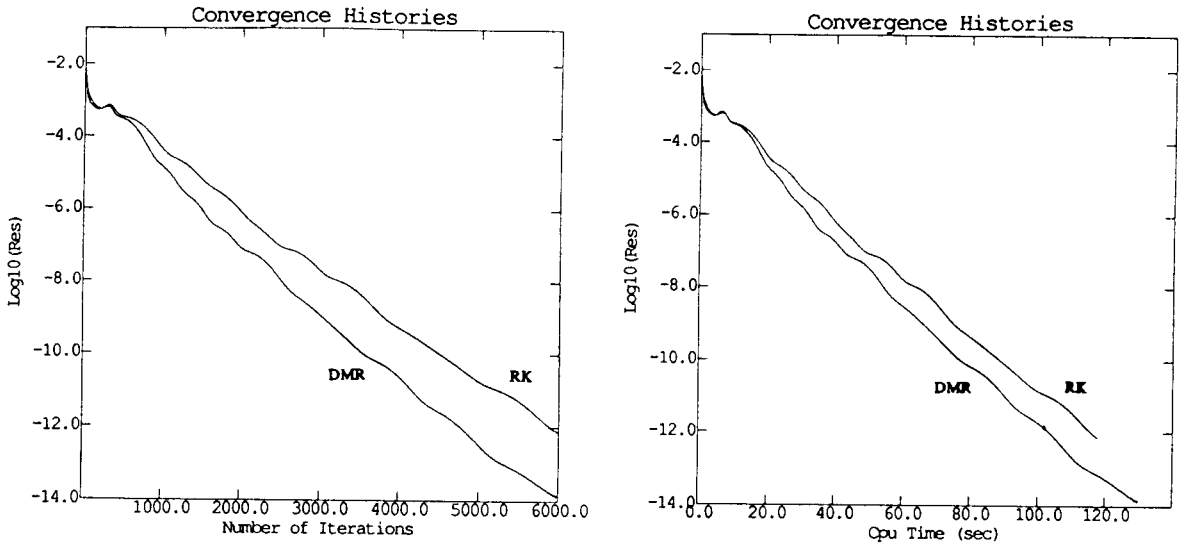

Fig. 7. Convergence histories for viscous flow around a circular cylinder with Re = 20 (RK).

implicit method. The DMR method was applied every 30 iterations for the RK method, and every 10 iterations for the Euler implicit method. For both methods, two consecutive iteration levels were used with the DMR method, though these combinations of the number of iterations and the frequency of the DMR application are not optimal. The artificial compressibility coefficient was $\beta = 1$ for both methods. The convergence histories of the RK method and the Euler implicit method with and without the DMR method are presented in Figs. 7 and 8. The DMR method offers more savings with the Euler implicit method than with the RK method. The wall pressure distributions and the wall vorticity distributions were compared with the computational results of Choi and Merkle [31] in Figs. 9 and 10. Reasonable agreement with the other results can be observed.

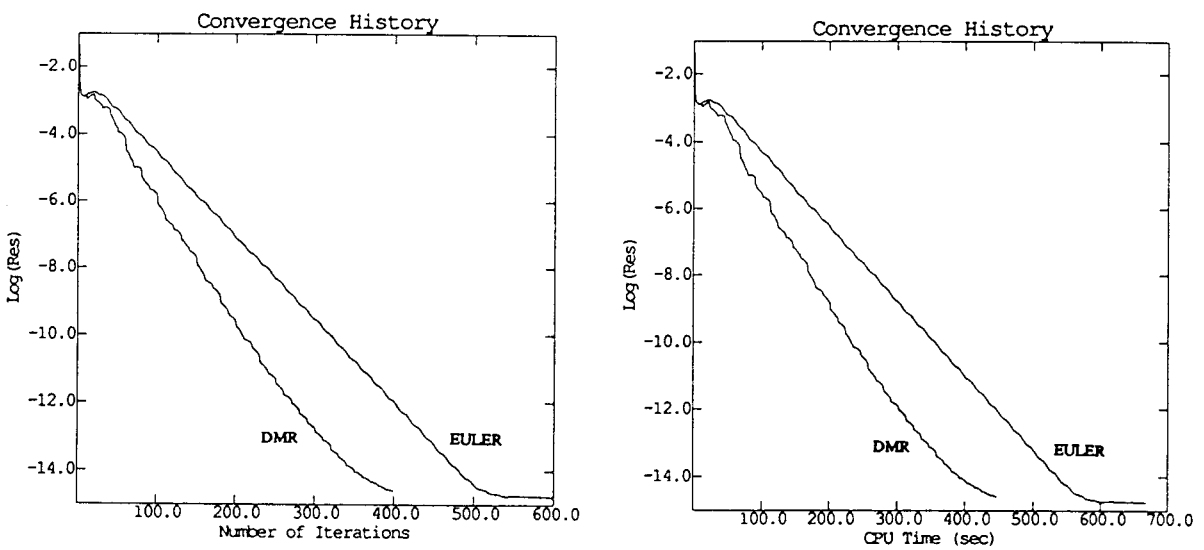

Fig. 8. Convergence histories for viscous flow around a circular cylinder with Re = 20 (Euler implicit).

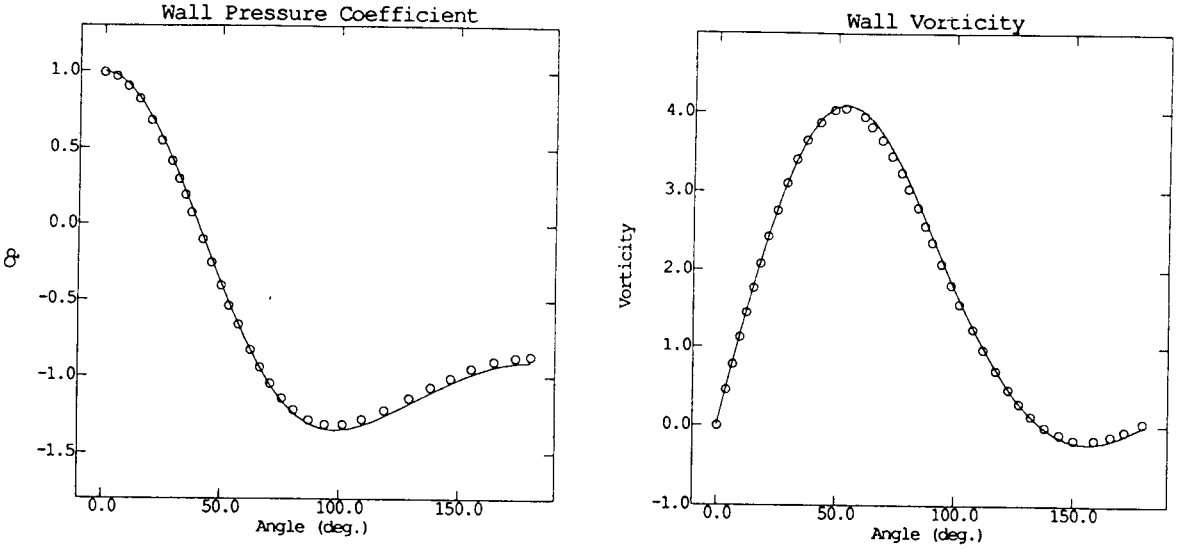

Fig. 9. Wall pressure coefficient distributions and vorticity distributions for flow around a circular cylinder at Re = 20 (RK). (a) Pressure coefficient. (b) Vorticity.

Although the DMR method requires information from only a few consecutive iteration levels, determination of the optimal number of the levels M combined during each application of the DMR method needs to be resolved. In addition, the question of optimal timing or optimal frequency of application of the DMR method remains to be resolved. An approach based on monitoring the slope and curvature of the residual versus iteration number curve [36] is a possibility. A more reliable formulation would be the one which involves both the number of iteration levels combined, M, and the optimal number of non-accelerated iterations, N, as

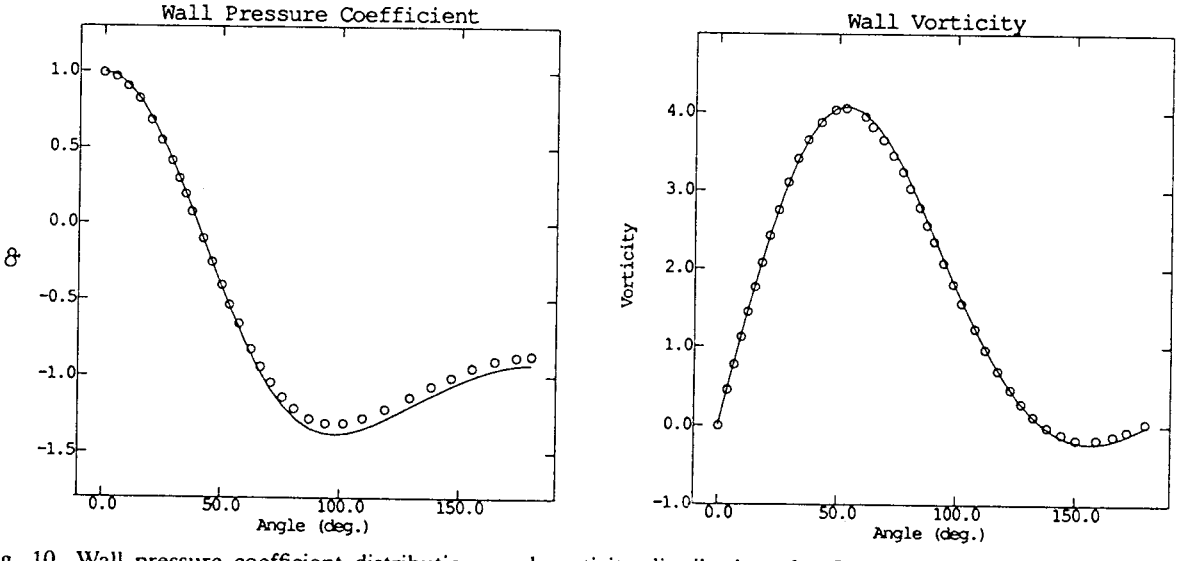

Fig. 10. Wall pressure coefficient distributions and vorticity distributions for flow around a circular cylinder at Re = 40 (Euler implicit).

integral parts of the future residual, $\boldsymbol{R}^{t+1}$. Hence, the optimal values for M and N could be obtained in the same way as the optimal values for ω.

5. Conclusions

The present form of the DMR method was found capable of reducing the computational time 20–80% depending on the test case. When directly compared with an implicit residual smoothing, the DMR method performed consistently better and more reliably. The DMR method requires considerably less additional memory compared to the GMRES method. The new method was successfully applied to both explicit and implicit algorithms.

Acknowledgment

The authors are grateful for the initial support provided by the AFOSR/NM and for the Cray computing time provided by the NAS facility at NASA Ames Research Center. Computing equipment was donated by Apple Computer, Inc.

References

[1] Q.Q. Huynh, Iterative preconditioned gradient–Newton type methods for modern shock computations, Report No. 107, Dept. of Computer Sciences, Uppsala University, 1987.
[2] D.K. Faddeev and V.N. Faddeeva, Computational Methods of Linear Algebra, Translated by R.C. Williams (Freeman, San Francisco, CA, 1963).
[3] P.K.W. Vinsome, ORTHMIN, an iterative method for solving sparse sets of simultaneous linear equations, Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976.
[4] D.M. Young and K.C. Jea, Generalized conjugate–gradient acceleration of non-symmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.
[5] Y. Saad and M. Schultz, Conjugate gradient-like algorithms for solving non-symmetric linear systems, Math. Comp. 44 (170) (1985).
[6] L.B. Wigton, N.J. Yu and D.P. Young, GMRES acceleration of computational fluid dynamics codes, AIAA Paper 85-1494 (1985).
[7] M. Hafez, E. Parlette and M.D. Salas, Convergence acceleration of iterative solutions of Euler equations for transonic flow computations, AIAA Paper 85-1641 (July 1985).
[8] C.-Y. Huang and G.S. Dulikravich, Fast iterative algorithms based on optimized explicit time stepping, Comput. Methods Appl. Mech. Engrg. 63 (July 1987) 15–36.
[9] C.-Y. Huang, S.R. Kennon and S.G. Dulikravich, Generalized non-linear minimal residual (GNLMR) method for iterative algorithms, J. Comput. Appl. Math. 16 (November 1986) 215–232.
[10] S. Lee, G.S. Dulikravich and D. Dorney, Distributed minimal residual (DMR) method for explicit algorithms applied to nonlinear systems, Presented at the Conference on Iterative Method for Large Linear Systems, Austin, Texas, 19–21 October 1988.
[11] G.S. Dulikravich, D.J. Dorney and S. Lee, Iterative acceleration and physically based dissipation for Euler equations of gasdynamics, in: O. Baysal, ed., Proc. ASME WAM'88, Symposia on Advances and Applications in Computational Fluid Dynamics, FED—Vol. 66 (1988) 81–92.
[12] S. Lee, G.S. Dulikravich and D.J. Dorney, Acceleration of iterative algorithms for Euler equations of gasdynamics, AIAA Paper 89-0097, Reno, NV, January 1989; also AIAA J. 28 (5) (May 1990) 939–942.

[13] S. Lee and G.S. Dulikravich, Accelerated computation of viscous, steady incompressible flows, ASME paper 89-GT-45, Gas Turbine and Aeroengine Congress and Exposition, Toronto, Canada, 4–8 June 1989.
[14] S. Lee and G.S. Dulikravich, A fast iterative algorithm for incompressible Navier–Stokes equations, Proceedings of the 10th Brazilian Congress of Mechanical Engineering, Rio de Janeiro, 7–10 December 1989.
[15] S. Lee and G.S. Dulikravich, Computer simulation of convective cooling effectiveness inside turning passages, in: R. Gaumont, ed., Proceedings of the 16th Northeast Bioengineering Conference, The Pennsylvania State University, PA, 26–27 March 1990.
[16] S. Lee and G.S. Dulikravich, Acceleration of iterative algorithms for incompressible Navier–Stokes equations, T. Mauteuffel, ed., Proceedings of the Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 1–5 April 1990.
[17] S. Lee and G.S. Dulikravich, Acceleration of viscous incompressible flows with heat transfer, in. N.-X. Chen, ed., Proceedings of the First International Conference on Experimental and Computational Aerothermodynamics of Internal Flows, Beijing, People's Rep. of China, 7–11 July 1990; also Numer. Methods Heat Transfer (to appear).
[18] S. Lee, Acceleration of iterative algorithms for Euler and Navier–Stokes equations, Ph. D. Thesis, The Department of Aerospace Engineering, The Pennsylvania State University, May 1990.
[19] S. Lee and G.S. Dulikravich, Acceleration of iterative algorithms using distributed minimal residual method, Proceedings of the 2nd World Congress on Computational Mechanics, Stuttgart, Germany, 27–31 August 1990.
[20] S. Lee and G.S. Dulikravich, Distributed minimal residual (DMR) method for acceleration of iterative algorithms, in: M.-S. Liou, ed., Proceedings of the CFD Symposium on Aeropropulsion, NASA Lewis Research Center, Cleveland, OH, 24–26 April 1990.
[21] W.R. Briley, H. McDonald and S.J. Shamroth, A low Mach number Euler formulation and application to time-iterative LBI schemes, AIAA J. 21 (10) (October 1983) 1467–1469.
[22] Y. Choi, Computation of low Mach number compressible flow, Ph. D. Thesis, The Department of Mechanical Engineering, The Pennsylvania State University, May 1989.
[23] E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations, J. Comput. Phys. 72 (2) (October 1987) 277–298.
[24] J.L. Steger and P. Kutler, Implicit finite-difference procedure for the computation of vortex wakes, AIAA J. 15 (7) (July 1977) 581–590.
[25] A. Jameson, W. Schmidt and E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping scheme, AIAA Paper 81-1259, Palo Alto, CA, June 1981.
[26] R.W. MacCormack and B.S. Baldwin, A numerical method for solving the Navier–Stokes equations with application to shock-boundary layer interaction, AIAA Paper 75-1, January 1975.
[27] S.R. Chakravarthy, Euler equations—Implicit schemes and implicit boundary conditions, AIAA Paper 82-0228, AIAA 20th Aerospace Science Meeting, Orlando, Florida, 11–14 January 1982.
[28] S.R. Kennon, Private communications, March 1990.
[29] B. Van Leer, Private communications, March 1988.
[30] A.J. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12–26.
[31] D. Choi and C.L. Merkle, Application of time-iterative schemes to incompressible flow, AIAA J. 23 (10) (October 1985) 1518–1524.
[32] D. Kwak, J.L.C. Chang, S.P. Shanks and S.R. Chakravarthy, An incompressible Navier–Stokes flow solver in three dimensional curvilinear coordinate system using primitive variables, AIAA J. 24 (3) (March 1986) 390–396.
[33] R.M. Beam and R.F. Warming, Implicit numerical methods for the compressible Navier–Stokes and Euler equations, Lecture Notes Series 1982-04, Von Karman Institute for Fluid Dynamics, Belgium, 1982.
[34] A. Jameson and T.J. Baker, Solution of the Euler equations for complex configurations, Proceedings of AIAA 6th Computational Fluid Dynamics Conference (AIAA, New York, 1983) 293–302.
[35] R.L. Panton, Incompressible Flow (Wiley, New York, NY, 1984).
[36] S.R. Kennon and G.S. Dulikravich, Optimum acceleration factors for iterative solution of linear and nonlinear differential systems, Comput. Methods Appl. Mech. Engrg. 47 (1984) 357–367.