JOURNAL OF COMPUTATIONAL PHYSICS **132,** 51–61 (1997)
ARTICLE NO. CP965605

Preconditioning Techniques for the Newton–Krylov Solution of Compressible Flows

M. D. Tidriri*

Department of Mathematics, Iowa State University, Ames, Iowa 50011-2064
E-mail: tidriri@iastate.edu.

Received June 5, 1995

In this paper, we study an efficient strategy for constructing preconditioners for the Newton–Krylov matrix-free methods without forming explicitly the higher order matrix associated with each linear step in the Newton iteration. These preconditioners are formed instead using an explicit derivation of a lower order matrix similar to that associated with a defect-correction procedure. Comparisons of this methodology with the more standard defect-correction procedures, namely, the approximate factorization (AF) for structured grids and the ILU/GMRES for general grids, are then performed. To illustrate the performance of our approaches, we present some numerical applications to the steady solution of a two-dimensional Euler flow.

1. INTRODUCTION

The implicit discretization of compressible flows leads to large sparse linear systems which need to be solved at each time step. In the derivation of this system, one often uses a defect-correction procedure in which the left-hand side of the system is discretized using a lower order approximation than for the right-hand side. This is due to the storage considerations and the computational complexity, as well as the fact that the resulting lower order matrix is better conditioned than the higher order matrix. The resulting methods are only moderately implicit. In the case of structured, body-fitted grids, the linear system can easily be solved using approximate factorization (AF), which is among the most widely used methods for such grids. For unstructured grids, such techniques are no longer valid, and the system is solved using direct or iterative methods. Because of the prohibitive computational costs and the large memory requirements for the solution of the compressible flows, iterative methods are preferred.

In these defect-correction methods, which are used in most CFD computer codes, the mismatch in the right- and left-hand side operators together with the explicit treatment of the boundary conditions leads to several limitations on the CFL number. This results in a slow convergence to the steady state aerodynamic solutions. Many authors have tried to replace explicit boundary conditions with implicit ones (see for instance [24, 18, and 10]). They showed that while high CFL number can be used, no clear advantages in terms of the CPU time as compared to explicit boundary conditions have been drawn. This is due to the mismatch between the right- and left-hand side operators. The Newton–Krylov methods in which the true Jacobian is computed will eliminate this mismatch. However, the derivation of the higher order Jacobian is prohibitive both in terms of the storage considerations and the computational complexity. In the Krylov context only a matrix-vector product is needed. Therefore, the action of the higher order Jacobian on any vector can be computed using a finite difference method through the Newton–Krylov matrix-free methodology.

Newton–Krylov matrix-free methods were introduced first, by Brown and Saad [4] and have been investigated for compressible Euler and Navier–Stokes equations using unstructured grids in [19], [20], and [9]. In [19] and [20], the author has studied both the transonic and supersonic compressible Navier–Stokes flows. In [19], [20], and [9] the authors have used the block diagonal preconditioner. However, most preconditioners require the explicit matrix. If we compute this matrix explicitly, the advantage of the matrix-free method will be lost. To overcome these difficulties and hence to avoid forming the higher order matrix explicitly, we propose to form instead the explicit matrix associated with a lower order approximation similar to that associated with the defect-correction procedure. Therefore, we propose in this paper to precondition the Newton–Krylov matrix-free algorithm by computing the preconditioner associated with a lower order approximation. Thus, the discretization scheme employed to derive the preconditioner is different than that used to compute the action of the Jacobian on a given vector (required in the Krylov methods). The resulting methods combined with implicit

* This work was supported by the National Aeronautics and Space Administration under NASA Contract NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering.

0021-9991/97 $25.00

boundary conditions will eliminate the mismatch between the left- and right-hand side operators, and hence will allow the use of high CFL numbers. Therefore, we present also an efficient CFL strategy in which the CFL may be adaptively advanced according to a relation that computes the current CFL number using the previous CFL number and the norm of the steady residue of the previous iterations.

In TRANAIR [25] the authors used the matrix-free method to solve the full potential problem. The preconditioner is based on an incomplete factorization of an explicitly generated matrix that is an approximation to the full Jacobian on a reduced set of unknowns. This matrix employs the same discretization scheme and order of accuracy used to compute the nonlinear residue except that it does not include the full linearization of the upwinding and the pseudo-unknowns are eliminated which is quite different than the strategy proposed in this paper. Moreover, in strongly transonic flow this matrix gives a significant inadequate definition of the linerarized Newton problem itself [25]. In [2], the Newton method is used to solve a steady subsonic Euler and the Navier–Stokes problems. However, the Jacobian matrix is derived explicitly and the action of the Jacobian on any vector is computed exactly. The authors form explicitly the matrix corresponding to a first-order or second-order discretization scheme which is then used in two different ways: (1) in the preconditioning of the linear system, and (2) as partial matrix-vector products in the iterative solver. When the first-order scheme is used in the right-hand side, the resulting method corresponds to the defect-correction method. However, since the authors study only the subsonic case, the convergence properties will be similar to that of Newton's method for the later stage of the convergence process toward the steady solution. This is not the case for the transonic flow where the mismatch between the right- and left-hand side will prevent the scheme from reaching the Newton convergence properties. Therefore, the left-hand side should be computed using the same order of accuracy as the one used for the right-hand side in order to reach the convergence properties of Newton's method. However, because of the prohibitive computational and storage cost, computing a higher order Jacobian explicitly is not a viable approach. The strategy developed in this paper presents an alternative to this approach.

In the next section, we describe the Euler solver. We then present in Section 3 the proposed methodology. Numerical experiments are presented in Section 4, in which we perform first a study of the defect-correction procedures based on an approximate factorization (AF) method and on Krylov methods with both explicit and implicit boundary conditions. We then study the performance of our methodology, which we compare to these defect-correction procedures. The last section is devoted to some remarks and extensions.

2. DESCRIPTION OF THE EULER SOLVER

2.1. *Governing Equations*

The two-dimensional Euler equations in conservative form are

$$W_t + F(W)_x + G(W)_y = 0, \tag{1}$$

where $W = (\rho, \rho u, \rho v, e)^{\mathrm{T}}$, $F = (\rho u, \rho u^2 + p, \rho uv, u(e + p))^{\mathrm{T}}$, and $G = (\rho u, \rho uv, \rho v^2 + p, v(e + p))^{\mathrm{T}}$. In these expressions, ρ is the density, u and v are the velocity components, e is the internal energy, p is the pressure defined by $p = (\gamma - 1)(e - \rho(u^2 + v^2)/2)$, and γ is a constant with $\gamma \approx 1.4$ for air.

After changing the variables into the curvilinear coordinates

$$\tau = t, \qquad \xi = \xi(x, y), \qquad \eta = \eta(x, y),$$

we obtain the following set of equations

$$\tilde{W}_\tau + (\tilde{F})_\xi + (\tilde{G})_\eta = 0. \tag{2}$$

Here $\tilde{W}$ and the contravariant flux vectors, $\tilde{F}$ and $\tilde{G}$, are defined in terms of the Cartesian fluxes and the Jacobian determinant of the coordinate system transformation, through the relations

$$\begin{aligned} \tilde{W} &= J^{-1}W, \\ \tilde{F} &= J^{-1}(\xi_t W + \xi_x F + \xi_y G), \\ \tilde{G} &= J^{-1}(\eta_t W + \eta_x F + \eta_y G), \end{aligned}$$

and

$$\begin{aligned} J &= \frac{\partial(\xi, \eta, \tau)}{\partial(x, y, t)} \\ &= \det \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix}. \end{aligned}$$

From now on, the tilde in the expressions of $\tilde{W}$, $\tilde{F}$, and $\tilde{G}$ will be omitted.

2.2. *Finite Volume Scheme*

An implicit finite volume discretization of the equation (2) can be written as

$$\begin{aligned} &(W_{i,j}^{n+1} - W_{i,j}^{n})\,\Delta\xi\,\Delta\eta + (F_{i+1/2,j}^{n+1} - F_{i-1/2,j}^{n+1})\,\Delta\eta\,\Delta\tau \\ &\quad + (G_{i,j+1/2}^{n+1} - G_{i,j-1/2}^{n+1})\,\Delta\xi\,\Delta\tau = 0, \end{aligned} \tag{3}$$

where the values are taken at the center of either the cell (i, j) or the interfaces of the cell (i, j) and its neighbours, and the superscripts refer to the iteration in time. To compute the fluxes above, we shall use a flux splitting approach, which is defined for F by (see [17])

$$F = F^+ + F^-,$$

with similar expressions for G. F^+ is associated with the positive eigenvalues whereas F^- is associated with the negative ones, and G^+, G^- are defined analogously.

Letting $\delta W = W_{i,j}^{n+1} - W_{i,j}^n$, the implicit split-flux discretization of (3) is given by

$$\delta W^n + \Delta\tau(\delta_\xi(F^+ + F^-)^{n+1} + \delta_\eta(G^+ + G^-)^{n+1}) = 0,$$

where δ_ξ is defined by

$$\delta_\xi F = \frac{1}{\Delta\xi}[F_{i+1/2,j} - F_{i-1/2,j}] \tag{4}$$

and δ_η is defined similarly. This yields the following nonlinear system

$$f(W^{n+1}) = 0. \tag{5}$$

This nonlinear system will be solved by using the proposed approach of this paper, which is based on a Newton–Krylov method (see future sections). We describe here the more standard defect-correction method, which is based on the following linearization of first-order in time of the above nonlinear system

$$[I + \Delta\tau(\delta_\xi^i A^+\cdot + \delta_\xi^i A^-\cdot + \delta_\eta^i B^+\cdot + \delta_\eta^i B^-\cdot)]\,\delta W^n = -\Delta\tau(\delta_\xi^e F^n + \delta_\eta^e G^n).$$

The superscripts i and e above indicate that the implicit and explicit operators are discretized using different schemes. The dots indicate that the difference operators apply to the product of the Jacobian matrices with δW^n. The matrices A^+, A^-, B^+, and B^- are defined by

$$A^+ = \frac{\partial F^+}{\partial W}, \qquad A^- = \frac{\partial F^-}{\partial W},$$
$$B^+ = \frac{\partial G^+}{\partial W}, \qquad B^- = \frac{\partial G^-}{\partial W}.$$

The compact form of the above equation corresponds to the defect-correction procedure

$$\mathbf{A}\,\delta W^n = b. \tag{6}$$

The different fluxes above are computed using Roe's approximate Riemann solver [14]. Three limiters are employed: minmod, Superbee, and Van Leer. The Jacobians are evaluated using the first-order Roe scheme or the first-order flux-vector split scheme [17], which corresponds to the true partials of the positive and negative flux vectors as described earlier. However, in the context of the defect-correction method, the flux-vector split scheme has been shown to give improved convergence rates over the Roe matrices. Therefore, for the defect-correction approach, the Jacobian matrices corresponding to the flux-vector split scheme are used in the left-hand side. This results in inconsistent left- and right-hand-side operators.

Remark 2.1. For most CFD codes, the implicit spatial differences are only first-order accurate. The matrix obtained using a higher order approximation is very large, requires a lot of storage and a large operation count in its evaluation, and may be very difficult to invert.

Following this remark, the implicit spatial differences (the left-hand side) in Eq. (6) are approximated through a first-order accurate scheme. The explicit spatial differences (right-hand side) in Eq. (6) are approximated using the higher order formulations of Roe's scheme, which are based on the work of Osher and Chakravarthy [13].

2.3. *Explicit Boundary Conditions*

The boundary conditions are derived using the locally one-dimensional characteristic variable boundary conditions, which yield (for the derivations see for example [12]):

2.3.1. *Farfield-Subsonic Inflow.*

$$P_b = (1/2)P_a + P_i + \mathrm{sign}(\lambda_k^i)\rho_o c_o[\bar{k}_x(u_a - u_i) + \bar{k}_y(v_a - v_i)]$$
$$\rho_b = \rho_a + [(P_b - P_a)/c_o^2]$$
$$u_b = u_a + \bar{k}_x[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i)$$
$$v_b = v_a + \bar{k}_y[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i).$$

The point a in these expressions is outside the computational domain, point b is on the computational boundary, and i is inside the computational domain.

2.3.2. *Farfield-Subsonic Outflow.*

$$P_b = P_a$$
$$\rho_b = \rho_a + [(P_b - P_a)/c_o^2]$$
$$u_b = u_a + \bar{k}_x[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i)$$
$$v_b = v_a + \bar{k}_y[(P_a - P_b)/(\rho_o c_o)]\mathrm{sign}(\lambda_k^i).$$

2.3.3. *Impermeable Surface.*

$$P_b = P_r \mp \rho_o c_o$$
$$u_b = u_r - \bar{k}_x(\bar{k}_x u_r + \bar{k}_y v_r)$$
$$v_b = v_r - \bar{k}_y(\bar{k}_x u_r + \bar{k}_y v_r).$$

Here the point r is the center of the first cell from the boundary and the minus sign in the first equation is used if r is in the positive k direction from the boundary, and the plus sign is used if r is in the negative direction from the boundary.

2.3.4. *Farfield-Supersonic Inflow.* In this case all eigenvalues have the same sign. Since we have an in-inflow case, all variables are specified.

2.3.5. *Farfield-Supersonic Outflow.* In this case also, all eigenvalues have the same sign. Here, however, we have an outflow case; therefore, all variables must be obtained from the solution in the computational domain. All variables are extrapolated from inside the computational domain to the boundary.

2.4. *Implicit Boundary Conditions*

In the implicit form, the above boundary conditions can be written in the form of operators formulated as functions of the conservation vector W

$$f_b(W) = 0 \tag{7}$$

and are implemented implicitly through

$$\frac{\partial f_b}{\partial W}\delta W = -f_b(W).$$

Using these implicit boundary conditions, the author showed in [21] that, starting from a small initial CFL number (10), the CFL may be adaptively advanced according to

$$\mathrm{CFL}^{n+1} = \mathrm{CFL}^n \cdot \frac{\|f(W)\|^{n-1}}{\|f(W)\|^n},$$

where the superscript refers to the iteration in time. As the results in Section 4 will show, this is the key to the successful implementation of the preconditioned Newton–Krylov matrix-free method studied in this paper.

3. DESCRIPTION OF THE METHODOLOGY

Newton–Krylov methods, first proposed by Brown and Saad [4], have been investigated for compressible Euler and Navier–Stokes equations using unstructured grids in [19, 20, 9], and for structured grids in [5, 6, 21]. In [19, 20], the author has studied both transonic and supersonic compressible Navier–Stokes flows. In [5, 6, 21, and 22], the author has studied the convection-diffusion problem and the transonic compressible Euler flows. A study of the strategy developed in this paper was performed by the author in the report [21]. This approach focuses on the construction of efficient preconditioners for the Newton–Krylov matrix-free algorithm without forming explicitly the higher order matrix representation usually required in the standard Newton methods. In [22] the author has combined this preconditioned Newton–Krylov matrix-free algorithm with the Schwarz domain decomposition methods in order to obtain an efficient parallel algorithm.

3.1. *Newton's Method*

Consider the following nonlinear system of equations

$$f(W) = 0, \tag{8}$$

where f is a nonlinear function from $\mathbb{R}^N$ to $\mathbb{R}^N$ ($N = 2$ or 3). Newton's method applied to (8) results in the following iteration:

- Define an initial guess δW_0.
- For $k = 0, 1, 2, \ldots$ until convergence do

Solve

$$J(W_k)\delta W_k = -f(W_k). \tag{9}$$

Set

$$W_{k+1} = W_k + \delta W_k. \tag{10}$$

where $J(W_k) = (\partial f/\partial W)(W_k)$ is the system Jacobian.

For the compressible Euler case (see Section 2) this Jacobian corresponds to a higher order matrix representation. Using direct methods to solve the system (9), the memory requirements and the computational complexity are prohibitive. In this case iterative methods are preferred and the system (9) is solved only approximately. The resulting method is called the inexact Newton method [7] and corresponds to the following iteration

- Define an initial guess δW_0
- For $k = 0, 1, 2, \ldots$ until convergence do

Solve

$$J(W_k)\delta W_k = -f(W_k). \tag{11}$$

Set

$$W_{k+1} = W_k + \alpha\delta W_k. \tag{12}$$

where $J(W_k) = (\partial f/\partial W)(W_k)$ denotes the system Jacobian as before, and α is a parameter selected using a line search or the trust region method ([4, 8]).

3.2. *Krylov Methods*

The iterative methods which we will use to solve the linear system (11), which we rewrite as

$$J\delta w = -f, \tag{13}$$

where f and its Jacobian J are evaluated at the current iterate, are the Krylov methods. If w_0 is an initial guess for the true solution of (13), then letting $w = w_0 + Z$, we have the equivalent system

$$JZ = r^0,$$

where $r^0 = -f - Jw_0$ is the initial residual. Let K_m be the Krylov subspace

$$K_m := \text{Span}\{r^0, Jr^0, ..., J^{m-1}r^0\}.$$

Arnoldi's method and GMRES both find an approximate solution

$$w_m = w_0 + Z_m, \qquad \text{with } Z_m \in K_m,$$

such that either

$$(-f - Jw_m) \perp K_m$$

for Arnoldi's method or

$$\|f + Jw_m\|_2 = \min_{w \in w_0 + K_m} \|f + Jw\|_2 \qquad (= \min_{Z \in K_m} \|r^0 - JZ\|_2)$$

for GMRES. Here, $\|.\|_2$ denotes the Euclidian norm on $\mathbb{R}^N$ and orthogonality is meant in the usual Euclidian sense.

In these Krylov methods, only the action of the Jacobian J times a vector w, and not J explicitly, is required. In the context of problem (8), this action can be approximated by a difference quotient of the form

$$J(u)w \approx \frac{f(u + \varepsilon w) - f(u)}{\varepsilon},$$

where u is the current approximation to a root of (8) and ε is a scalar. Selecting an optimal parameter ε might be a difficult problem. If ε is too small then the rounding errors made in the numerator are amplified by a factor of order $1/\varepsilon$ which leads to an inaccurate result. If, on the other hand, ε is too large then the approximation of $J(u)w$ will be poor. Any reasonable choice of ε should attempt to reach a compromise between these two difficulties. The technique for choosing the scalar ε we use here is

$$\varepsilon = \frac{\sqrt{\varepsilon_{\text{mach}}}}{\|w\|_2^2} \cdot \max\{|(u, w)|, \text{typ}\, u|w|\},$$

where $|w| = (|w_1|, ..., |w_n|)^{\mathrm{T}}$, and typ u is a given value depending on u and the problem to be solved. The Krylov method retained in this paper is GMRES. For more detail we refer the reader to [4].

3.3. *Preconditioned Newton–Krylov Matrix-Free Methods*

The combination of the Krylov matrix-free methods and the inexact Newton method described above results in the Newton–Krylov matrix-free algorithm introduced in [4]. Although the matrix-free method is attractive because it does not form the matrix explicitly, the matrix is still required for preconditioning purposes. In [19, 20, 9] the authors settled for a compromise that uses a block-diagonal preconditioner. In this case the Newton–Krylov matrix-free algorithm writes

- Define δW_0^n, an initial guess.
- For $k = 0, 1, 2, ...$ until convergence do

  Solve

$$D^{-1} \frac{f(W_k^n + \varepsilon \delta W_k^n) - f(W_k^n)}{\varepsilon} = -D^{-1} f(W_k^n). \tag{14}$$

  Set

$$W_{k+1}^n = W_k^n + \alpha \delta W_k^n,$$

where D is a block diagonal matrix, and α is the parameter introduced in Section 3.1. Most preconditioners require the matrix explicitly. This is true for the ILU preconditioner. However, as we mentioned earlier, the prohibitive memory requirements and the computational complexity for the higher order matrix representation, whether by analytical or numerical means, make the explicit calculation of such a matrix a difficult problem. Moreover, if we compute this matrix explicitly the advantage of the matrix-free method will be lost. In order to overcome these difficulties, instead of computing the preconditioner from the system Jacobian (higher order matrix, which is unavailable explicitly for our matrix-free version), we propose to form only, as in the defect-correction procedure (6), the explicit Jacobian matrix associated with a lower order approximation of the Jacobian. We derive then an ILU preconditioner based on a lower order approximation to the true Jacobian. This includes: (a) the Jacobian of a lower order discretization, (b) and the Jacobian obtained using a lower order discreti-

zation that allows a less expensive analytical evaluation of elements.

Applying the resulting method to the fully implicit nonlinear system (5) and (7) yields the following algorithm:

- Define δW_0^n, an initial guess.
- For $k = 0, 1, 2, \ldots$ until convergence do

Solve

$$M^{-1}\frac{f(W_k^n + \varepsilon\delta W_k^n) - f(W_k^n)}{\varepsilon} = -M^{-1}f(W_k^n). \quad (15)$$

Set

$$W_{k+1}^n = W_k^n + \alpha\delta W_k^n$$

with α the parameter introduced in Section 3.1. The preconditioner M^{-1} is constructed using an approximation similar to that used to derive the matrix $\mathbf{A}$ of the defect-correction procedure (6) as described above (points (a) and (b)). This results in a combined discretization in which for each linear step (15) of the Newton iteration the preconditioner is not derived from the actual higher order system (11). Instead, this preconditioner is derived using an approximation of the Jacobian matrix that employs a lower order discretization in a fashion similar to defect-correction procedure.

4. NUMERICAL RESULTS

The test problem on which we studied the performance of the methodology developed here corresponds to a NACA0012 steady transonic airfoil at an angle of attack of 1.25° and a freestream Mach number of 0.8. Since the standard solution of this problem can be obtained using the C-grids 128×32 cells, this mesh is the one retained in this study. For a study of the methodology proposed here on a coarser mesh we refer the reader to [21]. In all computations performed herein the solution obtained agrees with the standard one [12]. The initial code employs the discretization, described in Section 2 with explicit boundary conditions, over a body-fitted grid. It was developed initially in [12] and used a linear solver of an approximate factorization (AF) type (see for example [3]).

Before studying the preconditioned Newton–Krylov matrix-free methods proposed in this paper, we shall first study and compare the performance of the defect-correction procedures of AF and ILU/GMRES types. This preliminary study will provide us with a basis for comparisons of our methodology. We start by performing this study using explicit boundary conditions. We then study the effect on the defect-correction procedure based on the ILU/GMRES method of replacing the explicit boundary conditions by implicit ones. Finally, we study and compare our approach with these defect-correction methods. All calculations were performed on the same Sparc20 machine. Since we are dealing with different methods which require varying amounts of work at each time step, we believe that the CPU time is the only true measure for comparing them. However, we also compare the iteration counts versus the steady-state residual norm for some cases.

In our simulations the incomplete factorization method we have used corresponds to ILU(0) in which we discard any elements that would introduce fill, and there was no restart for the GMRES accelerator we used. The convergence criterion for the resulting linear solver (either in the defect-correction procedures or within each linear step of the Newton method) corresponds to a relative reduction in the norm of the residual equal to 10^{-3}.

4.1. *Defect-Correction Procedures*

4.1.1. *Explicit Boundary Conditions.* We start by comparing the results obtained using the approximate factorization AF method and ILU/GMRES when the boundary conditions are explicit. For these defect-correction procedures, which correspond in their compact form to the relation (6), the matrix $\mathbf{A}$ is computed using a first-order Van Leer scheme while the right-hand side of the system is computed using Roe's scheme. In the approximate factorization method the maximum CFL allowed was equal to 15, while it was equal only to 5 for the ILU/GMRES method. In both cases we have used the maximum CFL allowed. We observe that, to reach the same level of accuracy, the CPU time necessary for the AF method is almost twice as much as the time necessary to reach the same level of accuracy with the Krylov method (ILU/GMRES) as can be seen in Fig. 1, which shows the logarithm of the steady state residual norm versus the CPU time.

4.1.2. *Implicit Boundary Conditions.* We first compare the different results obtained using different CFL numbers. The results are presented in Fig. 2 where we show a comparison of the logarithm of the steady residual norm versus the CPU time. These calculations are performed using a CFL number equal to 5, 100, and 500 respectively. From these comparisons we observe that using a high CFL number improves the convergence rate. However, when the CFL number is larger than 100, no further improvement can be obtained. This is due to the mismatch of the left- and right-hand side operators. We will see in the next section that this drawback can be removed using the preconditioned Newton–Krylov matrix-free methodology described in Section 3. We now validate the CFL strategy described in Section 2. In Fig. 3 we compare the results obtained using a CFL number of 100 with those obtained using the CFL strategy described in Section 2. This comparison shows the feasibility of these techniques and their validation. It also shows that the converged solution is obtained in about the same CPU time. In Fig. 4, we show

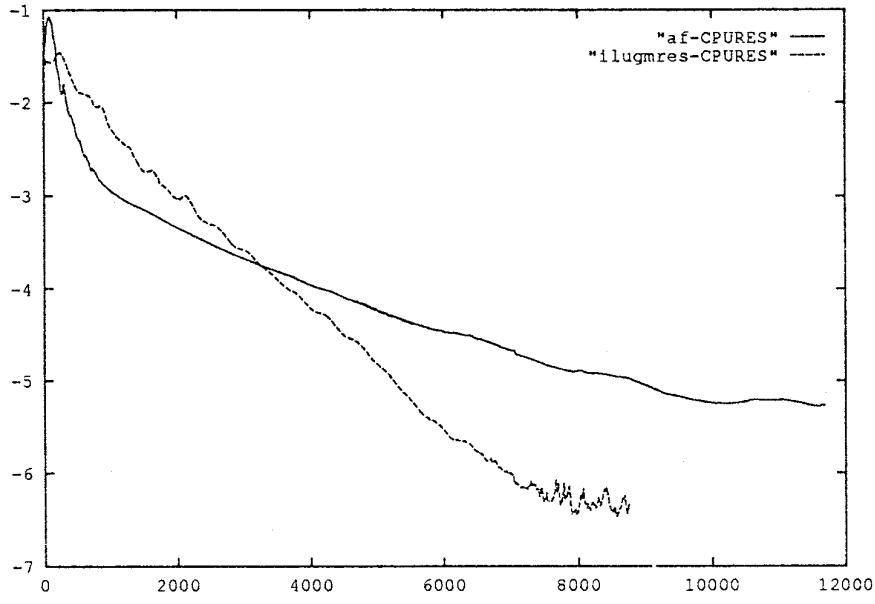

FIG. 1. Steady-state residual versus CPU time for approximate factorization (AF) and ILU/GMRES solvers.

the CFL versus the iteration count. To illustrate the performance of the implicit boundary conditions as compared to the explicit ones, we show in Fig. 5 the comparison of the steady state residual norm versus the CPU time for the converged solution obtained using explicit boundary conditions with a CFL number equal to 5 and using the implicit boundary conditions with a CFL equal to 100. We observe that an improvement in terms of the CPU time is obtained when we use the implicit boundary conditions as compared to the explicit ones. This comparison highlights the gain obtained using the implicit boundary conditions when the Krylov methods are used as linear solvers. It should be noted that, in using the AF solver, the implicit boundary conditions do not improve the convergence rate because the AF method is based on an approximation of a first-order of the linear system to be solved.

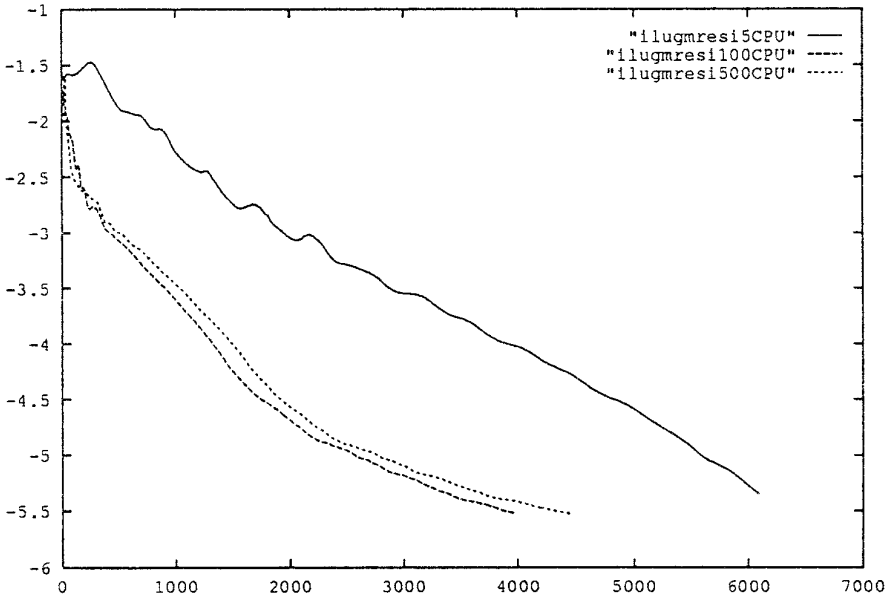

FIG. 2. Steady-state residual versus CPU time for ILU/GMRES solver with different CFL: 5, 100, and 500.

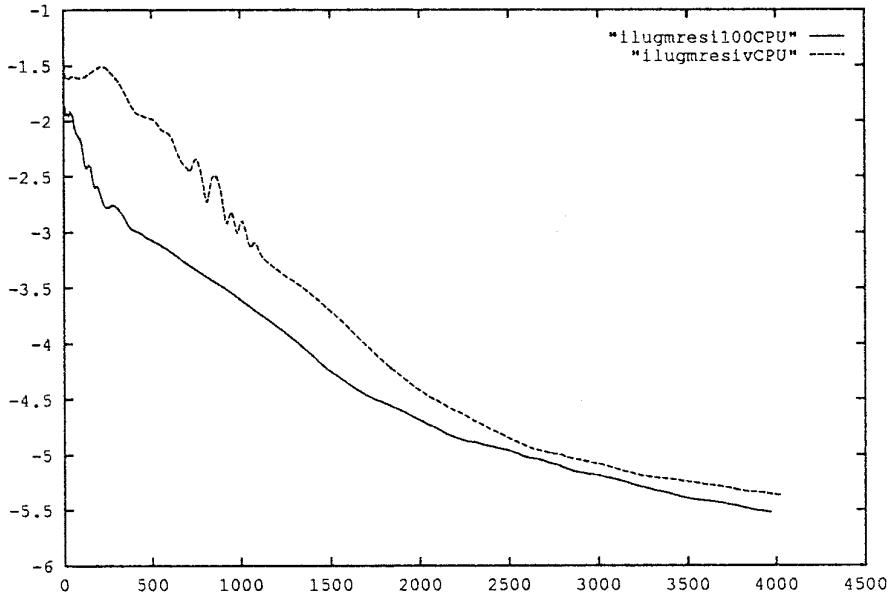

FIG. 3. Steady-state residual versus CPU time for ILU/GMRES solvers with CFL constant equal 100, and with adaptively increasing CFL.

4.2. *Preconditioned Newton–Krylov Matrix-Free Algorithm*

We study now the preconditioned Newton–Krylov matrix-free method developed in Section 3 with full implicit boundary conditions (see (2.4)). This method corresponds to the algorithm (15). In this algorithm, the fluxes involved in the finite difference quotient are computed using the same scheme as that used to compute the flux in the right-hand side, which corresponds to the higher order Roe scheme. The techniques used in the choice of the finite differencing parameter are described in Section 3. The preconditioner M^{-1} in (15) corresponds to an incomplete factorization without fill (ILU(0)). As in the defect correction procedure, the matrix M is computed using a first-order Van Leer scheme. To take full advantage of the convergence properties of the Newton method, and thus to allow a more rapid asymp-

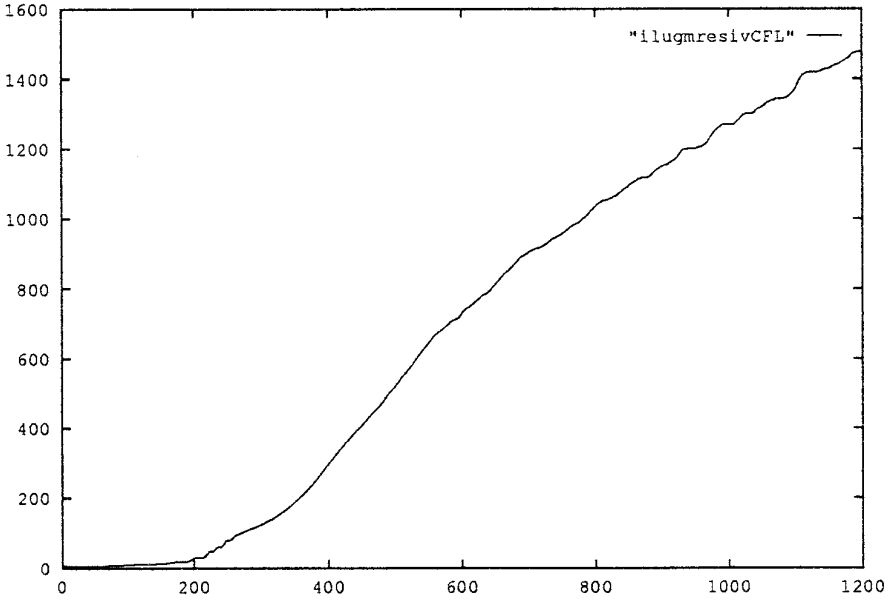

FIG. 4. CFL versus iteration count for ILU/GMRES solvers.

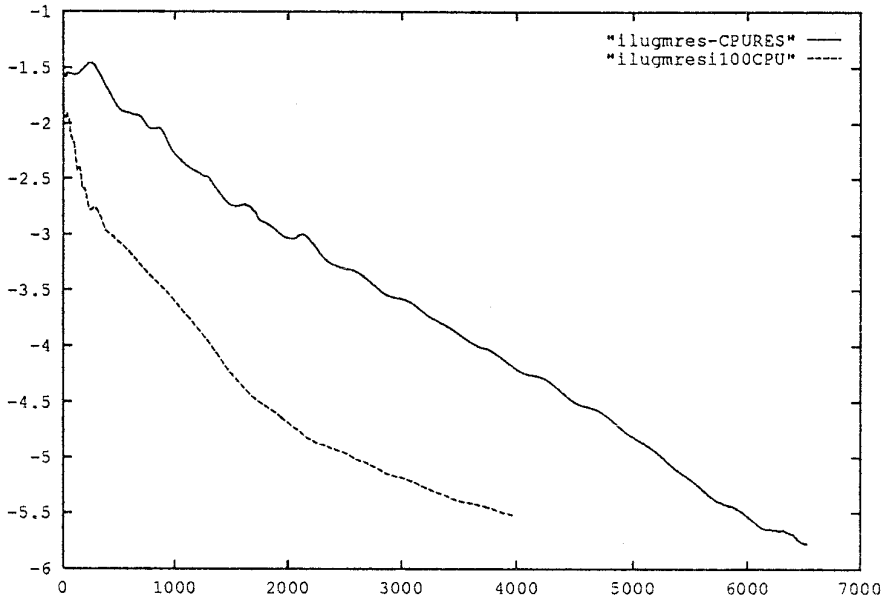

FIG. 5. Steady-state residual versus CPU time for ILU/GMRES solvers with explicit boundary conditions and implicit boundary conditions.

totic convergence to the steady state solution, we use the CFL strategy described in Section 2 and validated above.

To study the performance of this preconditioned Newton–Krylov matrix-free algorithm, we compare it with the defect correction method of ILU/GMRES type studied above in which the CFL is equal to 100. In both cases, the boundary conditions are implicit. The results are presented in Fig. 6. This figure shows the logarithm of the steady residual norm versus the CPU time. For each implicit time step we perform four Newton iterations. The stopping criterion corresponds to a steady residual norm of 10^{-9}. The starting CFL number is equal to 60, which allows us to accelerate the early stages of the nonlinear iteration, resulting in a distinct advantage over polyalgorithmic procedures. In fact, one may attempt for example to combine the Newton–Krylov methodology with the defect correction procedure in order to accelerate the early stages of the nonlinear iteration, especially in this transonic regime (see Section 5). The ending CFL number is equal to 28603.

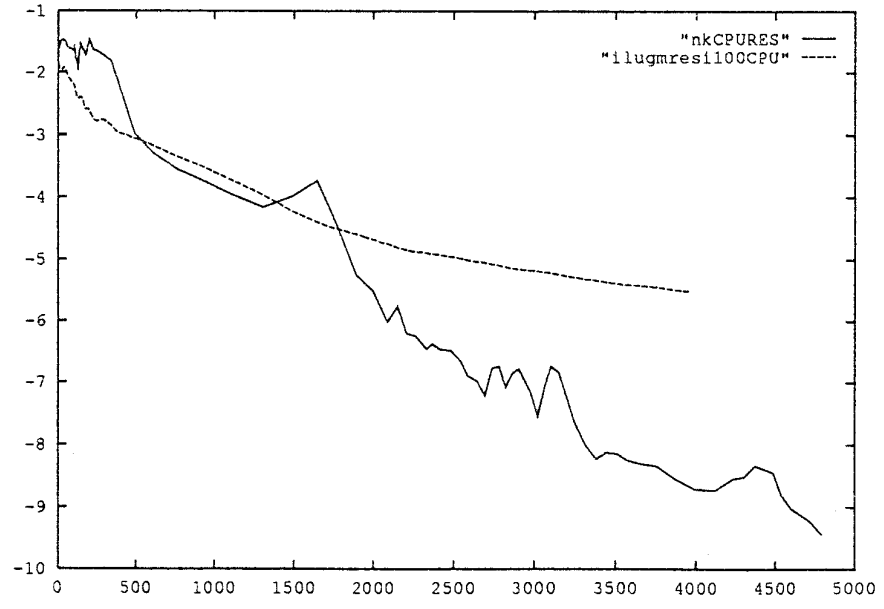

FIG. 6. Steady-state residual versus CPU time for defect correction (ILU/GMRES) and Newton–Krylov matrix-free solvers.

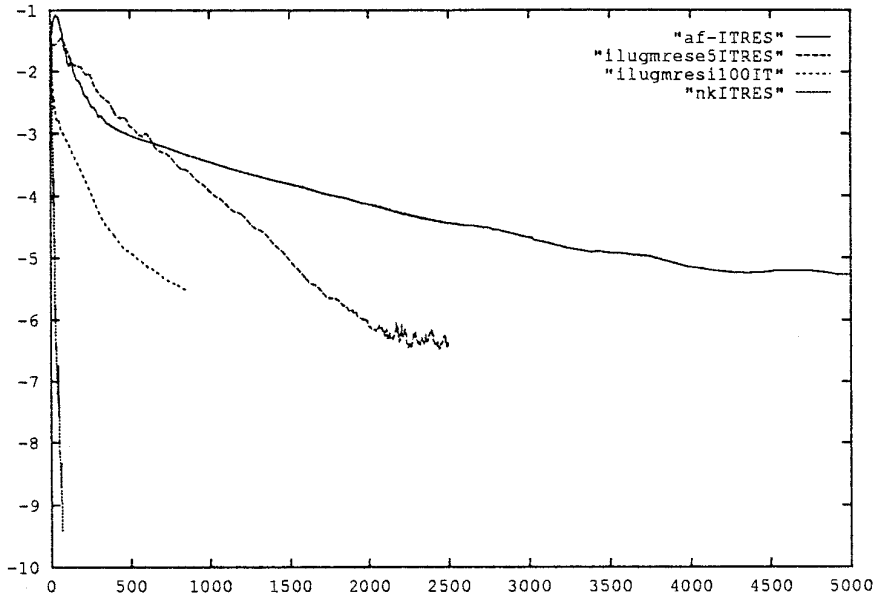

FIG. 7. Steady-state residual versus iteration count for approximate factorization (AF), ILU/GMRES with explicit boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton–Krylov matrix-free solvers.

In Figs. 7 and 8 we show a comparison of the logarithm of the steady state residual norm versus the iteration counts and the CPU time for the preconditioned Newton–Krylov method proposed in this paper and the three other defect-correction methods studied above. Clearly, we observe that the preconditioned Newton–Krylov matrix-free outperforms all of the other three methods.

5. CONCLUSIONS

In this study we have demonstrated the performance of the preconditioned Newton–Krylov matrix-free algorithm proposed in this paper.

We have performed only comparisons of the Krylov methods with the approximate factorization (AF) meth-

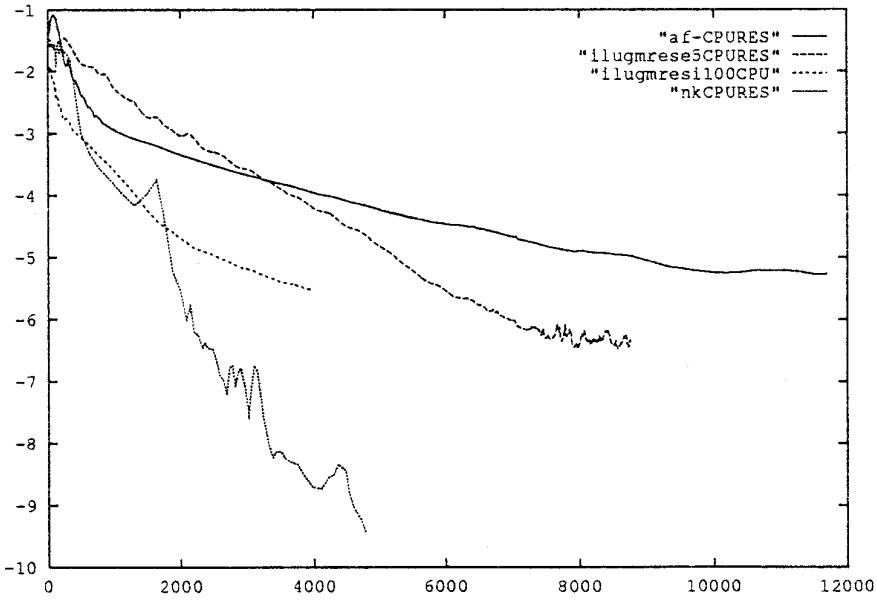

FIG. 8. Steady-state residual versus CPU time for approximate factorization (AF), ILU/GMRES with explicit boundary conditions, ILU/GMRES with implicit boundary conditions, and Newton–Krylov matrix-free solvers.

ods. However, in [23], it was shown that the ILU/GMRES method outperforms the relaxation methods, therefore we can conclude that the methodologies developed here outperform the most widely used methods in CFD, namely the AF method for structured grids and the relaxation methods for general grids.

In the subsonic case, the influence of the approximate determination of the Jacobians becomes rather small and the nearly quadratic convergence of the Newton's method is obtained. The appearance of a discontinuity in the transonic flow increases the inconsistency between the approximate Jacobian and the flux terms on the right-hand side. Furthermore, the process of shock capturing and positioning degrades the convergence rate and thus more iterations are required to drop the residual to the same order of magnitude as in the subsonic case. For completely supersonic inviscid flows, the information has to be propagated only in one direction, resulting in a faster convergence compared with the transonic case despite the complex shock system (see for instance [16]). Therefore our focus was only on the transonic case.

Since the early stages of the nonlinear iteration may be slow, one may attempt to combine the Newton–Krylov matrix-free strategy with a standard defect correction procedure. However, the resulting polyalgorithm may not be easy to handle, and it is not suitable for the parallel computing environment [22]. In our study, here, we have shown that high CFL number can be used at the early stages, which leads to an efficient algorithm. This results in a distinct advantage over the polyalgorithmic approach.

ACKNOWLEDGMENTS

The author would like to thank the editor and the referees for their suggestions and criticisms which led to a significant improvements of the original version of this paper.

REFERENCES

1. T. J. Barth, *"Analysis of Implicit Local Linearization Techniques for Upwind and TVD Algorithms,"* AIAA Paper 87-0595, Jan. 1987.
2. T. J. Barth and S. W. Linton, *"An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation,"* AIAA Paper 95-0221, Jan. 1995.
3. R. M. Beam and R. F. Warming, *An Implicit Factored Scheme for the Compressible Navier–Stokes Equations,* AIAA J. **16,** No. 4 (Apr. 1978), 393–402.
4. P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Statist. Comp.* **11**(1990), 450–481.
5. X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Parallel implicit methods for aerodynamics, in* "Proceedings of the Seventh International Conference on Domain Decomposition Methods for Partial Differential Equations," (D. Keyes and J. Xu, Eds.), American Math. Society, Providence, RI, 1994.
6. X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton–Krylov–Schwarz Methods in CFD, in* "Proceedings of the International Workshop on the Navier–Stokes Equations," Notes in Numerical Fluid Mechanics (R. Rannacher, Ed.), Vieweg Verlag, Braunschweig, 1994.
7. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods, SIAM J. Numer. Anal.* **19,** No. 2 (April 1982), 400–408.
8. J. E. Dennis, Jr. and R. B. Schnabel, *"Numerical Methods for Unconstrained Optimization and Nonlinear Equations,"* Prentice–Hall, Englewood Cliffs, NJ, 1983.
9. Z. Johan, T. J. R. Hugues, K. K. Mathur, and S. L. Johnsson, *"A Data Parallel Finite Element Method for Computational Fluid Dynamics on the Connection Machine System,"* Computer Methods in Applied Mechanics and Engineering, Vol. 99, pp. 113–124, 1992.
10. M.-S. Liou and B. Van Leer, *Choice of Implicit and Explicit Operators for the Upwind Differencing Method,* Paper AIAA-88-0624, AIAA, 1988.
11. P. R. McHugh and D. A. Knoll, *"Inexact Newton's Method Solutions to the Incompressible Navier–Stokes and Energy Equations Using Standard and Matrix-Free Implementations,"* Paper, AIAA, 1993.
12. J. S. Mounts, D. M. Belk, and D. L. Whitfield, *"Program EAGLE User's Manual," Vol. IV, "Multiblock Implicit, Steady-State Euler Code,"* Air Force Armament Laboratory TR-88-117, Vol. IV, Sep. 1988.
13. S. Osher and S. R. Chakravarthy, *"Very High Order Accurate TVD Schemes,"* Report 84-44, ICASE, Sep. 1984.
14. P. L. Roe, *Approximate Riemann solvers, parameter vector, and difference schemes, J. Comp. Phys.* **43**(1981), 357–372.
15. Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM J. Sci. Statist. Comp. **7**(1986), 865–869.
16. E. Scholl and H.-H. Fruhauf, *An accurate and efficient implicit upwind solver for the Navier–Stokes equations, in* "Proceedings of the International Workshop on the Navier–Stokes Equations," Notes in Numerical Fluid Mechanics (R. Rannacher, Eds.), Vieweg Verlag, Braunschweig, 1994.
17. J. L. Steger and R. F. Warming, *Flux vector splitting of the inviscid gas dynamics equations with applications to finite-difference methods, J. Comp. Phys.* **40**(1981), 263–293.
18. W. T. Thomkins, Jr. and R. H. Bush, *Boundary treatments for implicit solutions to Euler and Navier–Stokes equations, J. Comp. Phys.* **48**(1982), 302–311.
19. M. D. Tidriri, *"Couplage d'approximations et de modèles de types différent dans le calcul d'écoulements externes,"* Ph.D. thesis, Univ. of Paris XI, May 1992.
20. M. D. Tidriri, *Domain decompositions for compressible Navier–Stokes equations, J. Comp. Phys.* **119**(1995), 271–282.
21. M. D. Tidriri, *"Krylov Methods for Compressible Flows,"* ICASE-CR 95-48 Report, June 1995.
22. M. D. Tidriri, *"Schwarz-Based Algorithms for Compressible Flows,"* ICASE-CR 96-4 Report, Jan. 1996.
23. V. Venkatakrishnan and D. J. Mavriplis, *"Implicit Solvers for Unstructured Meshes,"* ICASE Report 91-40, May 1991.
24. H. C. Yee, R. M. Beam, and R. F. Warming, *Boundary approximations for implicit schemes for one-dimensional inviscid equations of gas dynamics, AIAA J.* **20,** No. 9 (Sep. 1982).
25. D. P. Young, C. C. Ashcraft, R. G. Melvin, M. B. Bieterman, W. P. Huffman, F. T. Johnson, C. L. Hilmes, and J. E. Bussoletti, *"Ordering and Incomplete Factorization Issues for Matrices Arising from the TRANAIR CFD Code,"* Boeing Computer Services Report BCSTECH-93-025, Aug. 1993.