ESAIM: M2AN
Vol. 39, N° 3, 2005, pp. 515–535
DOI: 10.1051/m2an:2005021

ESAIM: Mathematical Modelling and Numerical Analysis

LOCAL PRECONDITIONERS FOR STEADY AND UNSTEADY FLOW APPLICATIONS

Eli Turkel[1] and Veer N. Vatsa[2]

Abstract. Preconditioners for hyperbolic systems are numerical artifacts to accelerate the convergence to a steady state. In addition, the preconditioner should also be included in the artificial viscosity or upwinding terms to improve the accuracy of the steady state solution. For time dependent problems we use a dual time stepping approach. The preconditioner affects the convergence rate and the accuracy of the subiterations within each physical time step. We consider two types of local preconditioners: Jacobi and low speed preconditioning. We can express the algorithm in several sets of variables while using only the conservation variables for the flux terms. We compare the effect of these various variable sets on the efficiency and accuracy of the scheme.

Mathematics Subject Classification. 65M06, 76M12.

Plenary lecture, Low Mach Number Flows Conference, June 21-25, 2004, Porquerolles, France.

Introduction

Preconditioning methods for low speed, steady flows have been available in open literature for almost 20 years [16]. Because such preconditioners are designed to modify the path to a steady state, they were originally not appropriate for time dependent calculations. It was later found that, within the context of dual time steps, one can apply these techniques for unsteady flows while maintaining temporal accuracy [14, 29, 32]. Preconditioning was originally used to improve the convergence to a steady state for low speed flows. In this case the ratio of the largest to the smallest eigenvalue is very large and preconditioning can reduce this disparity. Subsequently, it was shown [23] that an appropriate preconditioning can also improve the accuracy of the steady state for low speed flows. In particular, it was proven [23] that a necessary condition for the convergence, as a reference Mach number goes to zero, of a discretization of the compressible equations to that of the incompressible equations is a condition on the scaling of the artificial viscosity or upwinding terms. This can be accomplished by multiplying the pressure gradients in the artificial dissipation by a term proportional to M^2. Most classical finite difference/volume/element methods do not satisfy this condition and, hence, behave poorly for low Mach flows. Including a preconditioning in the artificial viscosity changes the upwinding to satisfy this condition. Numerous computations verify that the preconditioning not only improves the convergence rate to a steady state for low speed flows but also dramatically improves the accuracy of the resultant steady state

Keywords and phrases. Low Mach, preconditioning, Jacobi, Dual time Step, compressible Navier Stokes.

[1] Tel-Aviv University, Israel and NIA, Hampton, VA.
[2] NASA Langley Research Center, Hampton,VA.

[16,19,30]. We shall investigate which of these properties are also true for dual time-stepping schemes. We shall refer to this preconditioner as the low speed preconditioner.

Consider the hyperbolic system of the unsteady Euler equations appended with pseudo-time derivatives. Let t denote the physical time, while τ denotes the pseudo-time used to drive each physical time step to a pseudo-steady state. In quasi-linear form we have

$$\frac{\partial w}{\partial \tau} + \xi \frac{\partial w}{\partial t} + A\frac{\partial w}{\partial x} + B\frac{\partial w}{\partial y} + C\frac{\partial w}{\partial z} = 0 \tag{1}$$

where w refers to a vector of dependent variables. The flux Jacobian matrices A, B, C are symmetric (or simultaneously symmetrizable). We are interested in $\tau \to \infty$. For physically steady state problems $\xi = 0$, while for time dependent problems $\xi = 1$. We discretize the physical time derivative with a backward difference formula (BDF):

$$\frac{\partial w}{\partial t} \sim \frac{c_t w^{n+1} - E(w^n, w^{n-1}, ...)}{\Delta t}, \tag{2}$$

where c_t is a constant that depends on the order of the temporal scheme. We precondition the system by replacing (1) by

$$\mathbf{P}^{-1}\frac{\partial w}{\partial \tau} + \xi \frac{\partial w}{\partial t} + A\frac{\partial w}{\partial x} + B\frac{\partial w}{\partial y} + C\frac{\partial w}{\partial z} = 0. \tag{3}$$

The equations are advanced in pseudo-time by a multistage Runge-Kutta (RK) scheme. Let superscript 0 denote the last artificial time and k be the most recent stage of RK. Let n be the last physical time step calculated and $n+1$ the next physical time step. R^k denotes the spatial derivative terms of the residual at the last stage, k. A typical stage of the RK scheme takes the form

$$w^{k+1} = w^0 - \alpha_k \Delta\tau \mathbf{P} \left(R^k + \frac{c_t w^{n+1} - E(w^n, ...)}{\Delta t} \right),$$

where α_k are the coefficients of the RK scheme. We use a simplified RK scheme where each stage depends on the original w^0 and the residual at the previous stage. The order of accuracy in physical time is determined only by the BDF scheme. In practice, only the inviscid portion of R^k is updated at each stage. The viscous portion is updated, for a five stage scheme, only on the odd stages. The difficulty is that w^{n+1} is not known. So we replace $n+1$ by $k+1$ (*i.e.*, current stage of RK). We reformulate this as

$$w^{k+1} = w^0 - \alpha_k \Delta\tau \mathbf{P} \left(R^k + \frac{c_t w^k - E(w^n, ...)}{\Delta t} \right) - \alpha_k c_t \Delta\tau \mathbf{P} \left(\frac{w^{k+1} - w^k}{\Delta t} \right). \tag{4}$$

The term $\frac{w^{k+1}-w^k}{\Delta t}$ was first suggested by Melson and Sanetrik [11] to make the scheme implicit in the Runge-Kutta algorithm. We define a modified residual

$$(R^*)^k = R^k + \frac{c_t w^k - E(w^n, ...)}{\Delta t}, \tag{5}$$

which denotes the total pseudo-residual that we drive to zero within each physical time step. Thus, within each subiteration we march the pseudo-time τ to infinity until R^* is zero. We call this solution at the end of a physical time step the pseudo-steady state. In practice, we take only a finite number of subiterations within each physical time step so that the pseudo-residual is sufficiently small. In the present study we choose a fixed number of subiterations. However, the goal is to automate the number of subiterations required for achieving consistently small global errors. We then rewrite (4) as

$$w^{k+1} = w^0 - \alpha_k \Delta\tau \mathbf{P} (R^*)^k - \alpha_k c_t \Delta\tau \mathbf{P} \left(\frac{w^{k+1} - w^k}{\Delta t} \right).$$

The last term in this equation is included to increase the the point implicitness of the scheme. This term vanishes in the steady state. Collecting terms, we have

$$\left(I + \alpha_k c_t \frac{\Delta\tau}{\Delta t}\mathbf{P}\right) w^{k+1} = w^0 - \alpha_k \Delta\tau \mathbf{P}(R^*)^k + \alpha_k c_t \frac{\Delta\tau}{\Delta t}\mathbf{P}w^k. \tag{6}$$

The spatial discretization is a central difference formula plus a matrix valued artificial dissipation using second and fourth differences [9, 15, 21]. We describe the second difference formulation for the artificial viscosity for brevity; the fourth differences are treated similarly. We express the dissipation in terms of derivatives rather than differences for presentation only. Consider

$$\frac{\partial w}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = \text{artificial viscosity terms.}$$

As an example, on the numerical level, the residual in the x direction would be

$$R = \frac{\partial F}{\partial x} - \Delta x \frac{\partial}{\partial x}\left(\epsilon_2 \mathbf{P}^{-1}|\mathbf{P}A|\frac{\partial w}{\partial x}\right). \tag{7}$$

ϵ_2 is a scaling factor in the artificial viscosity. A is the Jacobian of F with respect to w. The absolute value of a matrix is found by diagonalizing the matrix and taking absolute values of the eigenvalues (with cutoffs) [15].

We Fourier transform (3) in space and replace w_t by (2). Define $\zeta = \frac{\xi c_t}{\Delta t}$ and

$$\mathbf{G}(\omega_1, \omega_2, \omega_3) = \mathbf{P}\left[-\zeta I + i(A\omega_1 + B\omega_2 + C\omega_3)\right] \tag{8}$$

with $\omega_1^2 + \omega_2^2 + \omega_3^2 = 1$. The condition number is defined as

$$cond\# = \max_{\omega_i}\left|\frac{\lambda_{\max}(\mathbf{G})}{\lambda_{\min}(\mathbf{G})}\right|, \tag{9}$$

where λ denotes an eigenvalue of the matrix. Note that the eigenvalues of $i(A\omega_1 + B\omega_2 + C\omega_3)$ are pure imaginary since the matrices are symmetric. Physically, the condition number (with $\xi = 0$) can be interpreted as the ratio of the fastest speed to the slowest speed in any direction. If viscous terms are included, then we have additional negative real matrices in (8). We stress that if Δt is sufficiently small, then the condition number is close to 1. The methods proposed here will not improve the condition number of such problems and, hence, will not improve the convergence rate to a pseudo-steady state. For the Euler equations, the condition number is approximately inversely proportional to the Mach number. Hence, for low speed steady flows, the preconditioner is expected to be very effective for convergence acceleration.

With a local preconditioner we change the discrete equations at individual grid nodes without introducing new coupling between neighboring nodes. Hence, this technique makes sense only for a system of equations. For a scalar equation, local preconditioning is simply a rescaling of the time variable and has no effect on the numerical solution. This approach is distinct from incomplete LU (ILU) decomposition based preconditioning techniques which couple all the nodes together and, therefore, require more expensive inversion techniques. We choose the matrix $\mathbf{P}$ so as to improve the condition number of the equations at the node point. The assumption is that the better the system is conditioned, the faster the iteration process will approach a steady state. For well-posedness the matrix $\mathbf{P}$ should be symmetric positive definite.

For time dependent problems where the physical time scale is sufficiently small, and so ζ is large, preconditioning can harm the convergence rate. For such problems the preconditioning in the update stage should be turned off and should only affect the artificial viscosity or the upwinding [12, 22]. In this study we consider two different local preconditioners, Jacobi and low speed, to alleviate stiffness associated with disparate characteristic speeds and from a poor condition number. We then formulate a composite preconditioner that combines

the complementary properties of the Jacobi and low speed preconditioners to achieve an efficient scheme for solving flows with embedded low speed flows.

1. Choice of variables

We consider the following sets of variables defined by

$$\begin{aligned} w_c &= (\rho, \rho u, \rho v, \rho w, E), \\ Q &= (p, u, v, w, T), \end{aligned} \tag{10}$$

$$w_0 = (p, u, v, w, S)\ ,\ \mathrm{d}\hat{w}_0 = \left(\frac{\mathrm{d}p}{\rho c}, \mathrm{d}u, \mathrm{d}v, \mathrm{d}w, \mathrm{d}S\right).$$

We shall refer to Q as the primitive variables, w_c as the conservation variables and w_0 as the entropy variables. We evaluate the flux F and the physical time derivatives in conservation variables so that we obtain the correct shock jumps. One implementation of (6) is to use the conservation variables throughout the equation. As the Mach number decreases to zero, the density usually becomes constant and so the conservation variables become less useful. The primitive variables, Q, are frequently used for incompressible flow and are more useful when solving for slightly compressible flow.

If we change from conservation variables to Q variables in the artificial viscosity, we replace (7) by

$$\begin{aligned} R_Q &= \frac{\partial F}{\partial x} - \Delta x \frac{\partial}{\partial x}\left(\epsilon_2 \Gamma^{-1} |\mathbf{P}_Q A_Q| \frac{\partial Q}{\partial x}\right), \\ (R_Q^*)^k &= R_Q^k + \frac{c_t w_c^k - F(w_c^n, ...)}{\Delta t}, \\ \mathbf{P}_Q &= \frac{\partial Q}{\partial w_c} \mathbf{P}_c \frac{\partial w_c}{\partial Q}, \\ \Gamma^{-1} &= \frac{\partial w_c}{\partial Q} \mathbf{P}_Q^{-1}, \quad \Gamma = \mathbf{P}_Q \frac{\partial Q}{\partial w_c} = \frac{\partial Q}{\partial w_c} \mathbf{P}_c . \end{aligned} \tag{11}$$

Multiplying (6) by $\frac{\partial Q}{\partial w_c}$ and since $\Delta w_c = \frac{\partial w_c}{\partial Q} \Delta Q$, we get

$$\left(I + \alpha_k c_t \frac{\Delta \tau}{\Delta t} P_Q\right) Q^{k+1} = Q^0 - \alpha_k \Delta \tau \Gamma (R_Q^*)^k + \alpha_k c_t \frac{\Delta \tau}{\Delta t} P_Q Q^k. \tag{12}$$

After each stage, we calculate w_c^{k+1} using the nonlinear relation between w_c and Q. Once the artificial time derivative approaches zero, the resultant equation is in conservation form, including the physical time derivative. If the physical time derivative were also transformed to Q variables, then we might lose the conservation form and, hence, the correct jump conditions at a shock. The pseudo-residual R^* should also be cast in conservation form so that we get the correct pseudo-steady state at convergence. Preconditioning destroys conservation in the midst of the pseudo-time iteration process, but recovers the conservation form for R^* when the pseudo-time iteration process converges. Because R_Q^* depends on both w_Q and w_c when using Q variables, we do not include the term $\frac{w^{k+1} - w^k}{\Delta t}$ in the update scheme. In this case we must include the physical time step, Δt, in the determination of the pseudo-time step $\Delta \tau$.

A third possibility is to consider a mixture of conservation and Q variables. When evaluating the artificial viscosity we use Q variables as given by (11). However, when updating the variables, we revert to w_c variables. We then get

$$\left(I + \alpha_k c_t \frac{\Delta \tau}{\Delta t} \mathbf{P}_c\right) w_c^{k+1} = w_c^0 - \alpha_k \Delta \tau \mathbf{P}_c (R_Q^*)^k + \alpha_k c_t \frac{\Delta \tau}{\Delta t} \mathbf{P}_c w_c^k. \tag{13}$$

This is the same as (6), except that the artificial viscosity R in w_c variables is replaced by R_Q based on Q variables. Hence, equations (6) and (12) have different numerical pseudo-steady states but equations (12) and (13) have the same numerical pseudo-steady state.

Computations confirm that the variables used in the artificial viscosity have a much larger effect on the pseudo-steady state than the choice of variables used to update the solution. We shall demonstrate that we can efficiently solve the preconditioned equations (6) and (13).

2. Low speed preconditioning

2.1. **Implementation**

For low Mach number flows, the ratio of acoustic to convective speeds is large, which results in an ill-conditioned and stiff system. Hence, we introduce a preconditioner to alleviate this stiffness. Because $\mathbf{P}$ based on conservation variables is a full matrix, we make use of entropy variables, in which the energy equation decouples from the rest of the governing equations. Furthermore, the Jacobian matrix is sparse in these variables. The simplest preconditioner in w_0 variables is given by (see [16, 17, 19, 24])

$$\mathbf{P_0}^{-1} = \begin{pmatrix} \frac{1}{\beta^2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

β is a parameter which should be of the order of the Mach number to approximately equalize all the eigenvalues of $P_0 A_0$.

Let $c^2 = \frac{\gamma p}{\rho}$, $q^2 = u^2 + v^2 + w^2$ and $\hat{q}^2 = \frac{(\gamma-1)q^2}{2}$. Then the Jacobians that connect these variables are

$$\frac{\partial w_0}{\partial w_c} = \begin{pmatrix} \hat{q}^2 & (1-\gamma)u & (1-\gamma)v & (1-\gamma)w & \gamma-1 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \hat{q}^2 - c^2 & (1-\gamma)u & (1-\gamma)v & (1-\gamma)w & \gamma-1 \end{pmatrix},$$

$$\frac{\partial w_c}{\partial w_0} = \begin{pmatrix} \frac{1}{c^2} & 0 & 0 & 0 & -\frac{1}{c^2} \\ \frac{u}{c^2} & \rho & 0 & 0 & -\frac{u}{c^2} \\ \frac{v}{c^2} & 0 & \rho & 0 & -\frac{v}{c^2} \\ \frac{w}{c^2} & 0 & 0 & \rho & -\frac{w}{c^2} \\ \frac{h}{c^2} & \rho u & \rho v & \rho w & -\frac{M^2}{2} \end{pmatrix},$$

where $h = \frac{c^2}{\gamma-1} + \frac{q^2}{2}$ is the specific enthalpy. The preconditioner $\mathbf{P}_c$ in conservation variables is then given by $\mathbf{P}_c = \frac{\partial w_c}{\partial w_0} \mathbf{P}_0 \frac{\partial w_0}{\partial w_c}$. The preconditioner appears in the form of the matrix multiplying a vector. We calculate $\mathbf{P}_c$ times an arbitrary vector $\vec{x} = (x_1, x_2, x_3, x_4, x_5)$ in stages. So,

$$\begin{aligned} \mathbf{P}_c \vec{x} &= \vec{x} + (\beta^2 - 1) y_1 \vec{z}, \\ \mathbf{P}_c^{-1} \vec{x} &= \vec{x} + \left(\frac{1}{\beta^2} - 1 \right) y_1 \vec{z}, \end{aligned} \tag{14}$$

where $y_1 = \frac{\gamma-1}{c^2}\left[\frac{q^2}{2}x_1 - (ux_2 + vx_3 + wx_4) + x_5\right]$ and

$$\vec{z} = \begin{pmatrix} 1 \\ u \\ v \\ w \\ h \end{pmatrix}.$$

In (6) we need to evaluate $(I + \hat{d}\cdot\mathbf{P}_c)^{-1}$ times a vector where $\hat{d} = \alpha_k c_t \frac{\Delta\tau}{\Delta t}$. This is accomplished by

$$(I + \hat{d}\cdot\mathbf{P}_c)^{-1}\vec{x} = \frac{\vec{x} + e y_1 \vec{z}}{1+\hat{d}} \quad , \quad e = \frac{(1-\beta^2)\hat{d}}{1+\beta^2\hat{d}}.$$

For the primitive variables, Q, we have

$$\mathbf{P_Q} = \begin{pmatrix} \beta^2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{(\beta^2-1)T}{c_p p} & 0 & 0 & 0 & 1 \end{pmatrix},$$

where $c_p = \frac{\gamma R}{\gamma-1}$. Define

$$\vec{\chi} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \frac{(\gamma-1)T}{\gamma p} \end{pmatrix}.$$

As above in equations (12) or (13), we need to evaluate $(I + \hat{d}\cdot\mathbf{P}_c)^{-1}$ times a vector. This is accomplished by

$$(I + \hat{d}\cdot\mathbf{P}_Q)^{-1}\vec{x} = \frac{\vec{x} + e x_1 \vec{\chi}}{1+\hat{d}} \quad , \quad e = \frac{\hat{d}(1-\beta^2)}{1+\hat{d}\cdot\beta^2}.$$

2.2. **Choice of parameters**

We need to choose β^2 and the pseudo-time step. When we ignore the term $\frac{w^{k+1}-w^k}{\Delta t}$ in (4), the RK scheme is explicit for the physical time derivative; however, it requires that the pseudo-time step also include a physical time step contribution. The precise form of this term is given in (18). The present analysis is done on the continuous level, except for the source term that arises from discretization of the physical time derivative by a BDF formula. The amplification matrix for a RK scheme is a polynomial in a stage amplification matrix. The total scheme is stable when all the eigenvalues of the stage amplification matrix lie within the stability region of the particular RK scheme. The stage amplification matrix in pseudo-time for the Euler equations in $\hat{w}_0$ variables, in generalized coordinates, is given by

$$\mathbf{G}(\omega_1,\omega_2,\omega_3) = \mathbf{P}_0\left[-\frac{c_t \mathrm{Vol}}{\Delta t} + i(\omega_1 A + \omega_2 B + \omega_3 C)\right], \tag{15}$$

where Vol is the cell volume and A, B, C are the Jacobian matrices of the inviscid flux vectors in the generalized coordinate space dimensions. Matrices A, B, C are symmetric in $d\hat{w}_0$ variables, and so this is a symmetric hyperbolic system. We denote the surface area of the cell as S_{ij} where the first subscript refers to the direction

of the normal and the second is the projection of that normal in each direction. Define the contravariant velocity components as:

$$\begin{aligned} U &= uS_{xx} + vS_{xy} + wS_{xz}\,, \\ V &= uS_{yx} + vS_{yy} + wS_{yz}\,, \\ W &= uS_{zx} + vS_{zy} + wS_{zz}\,. \end{aligned} \tag{16}$$

Then (15) becomes

$$G(\omega_1,\omega_2,\omega_3) = -\frac{c_t \mathbf{P}_0\, Vol}{\Delta t} + i\omega_1 \begin{pmatrix} \beta^2 U & \beta^2 cS_{xx} & \beta^2 cS_{xy} & \beta^2 cS_{xz} \\ cS_{xx} & U & 0 & 0 \\ cS_{xy} & 0 & U & 0 \\ cS_{xz} & 0 & 0 & U \end{pmatrix} + i\omega_2 \begin{pmatrix} \beta^2 V & \beta^2 cS_{yx} & \beta^2 cS_{yy} & \beta^2 cS_{yz} \\ cS_{yx} & V & 0 & 0 \\ cS_{yy} & 0 & V & 0 \\ cS_{yz} & 0 & 0 & V \end{pmatrix}$$
$$+ i\omega_3 \begin{pmatrix} \beta^2 W & \beta^2 cS_{zx} & \beta^2 cS_{zy} & \beta^2 cS_{zz} \\ cS_{zx} & W & 0 & 0 \\ cS_{zy} & 0 & W & 0 \\ cS_{zz} & 0 & 0 & W \end{pmatrix}.$$

Let

$$|q| = \sqrt{U^2 + V^2 + W^2}$$

and

$$\omega_1 = \frac{U}{|q|} \quad \omega_2 = \frac{V}{|q|} \quad \omega_3 = \frac{W}{|q|}.$$

Therefore, $\omega_1^2+\omega_2^2+\omega_3^2=1$. Define

$$\begin{aligned} \hat{U} &= \frac{S_{xx}U + S_{yx}V + S_{zx}W}{|q|}\,, \\ \hat{V} &= \frac{S_{xy}U + S_{yy}V + S_{zy}W}{|q|}\,, \\ \hat{W} &= \frac{S_{xz}U + S_{yz}V + S_{zz}W}{|q|}\,, \\ \hat{q}^2 &= \hat{U}^2 + \hat{V}^2 + \hat{W}^2 \end{aligned}$$

where $\hat{U}, \hat{V}, \hat{W}$ depend on the velocity components u, v, w and the geometry metrics. This gives

$$G = \begin{pmatrix} \beta^2 D & \beta^2 c\hat{U} & \beta^2 c\hat{V} & \beta^2 c\hat{W} \\ c\hat{U} & D & 0 & 0 \\ c\hat{V} & 0 & D & 0 \\ c\hat{W} & 0 & 0 & D \end{pmatrix},$$

where:

$$D = -\frac{c_t \mathrm{Vol}}{\Delta t} + i|q|.$$

The eigenvalues are $\lambda_0 = D$ and

$$\lambda_\pm = \frac{\beta^2+1}{2} D \pm \sqrt{\left(\frac{\beta^2-1}{2}\right)^2 D^2 + \beta^2 c^2 \hat{q}^2}. \tag{17}$$

Because D is a complex number, so is $\lambda_\pm$. We define $\lambda_{\text{inv}} = \max(|\lambda_+|, |\lambda_-|)$. The artificial time step is determined by demanding that $\lambda_\pm$ be within the stability domain of the RK scheme. Since $\lambda_\pm$ is a complex number, this leads to a condition that depends on the details of the stability curve. Hence, we replace this by a condition on the real and imaginary parts separately. We use a similar argument to account for viscous terms in the eigenvalues and the artificial time step [20]. The formula we use for calculating the artificial time step, $\Delta\tau \sim \frac{1}{\lambda}$, is given by

$$\frac{1}{\Delta\tau} = \frac{1}{\Delta\tau_{\text{ss}}} + \frac{K_\tau}{c_t \Delta t}\,, \tag{18}$$

where $\Delta\tau_{\text{ss}}$ is the steady state (without dual time-stepping) artificial time step which is a sum of inviscid and viscous contributions. For most of the results presented in this paper, $K_\tau = 1$. Using an implicit formula for the physical time derivative as derived above allows $K_\tau = 0$. However, for robustness, we usually choose $K_\tau = 1$. Even when $K_\tau = 0$, $\Delta\tau$ depends on the physical time step, Δt, through λ which is a function of β^2. For preconditioning based on primitive variables, we do not include $\frac{w^{k+1}-w^k}{\Delta t}$ in (4) and so we choose a higher value for $K_\tau = 2$.

The major difficulty in determining β is that D is a complex number. Hence, we cannot choose a real β to (approximately) equalize the eigenvalues λ_0 and $\lambda_\pm$. Ignoring β^2 compared with 1, we would like D^2 to be approximately equal to $\beta^2 c^2 \hat{q}^2$. However, one term is real and the other term is complex. Furthermore, the square root of a complex quantity combines the real and imaginary parts in every term. For inviscid steady state flow, all the terms are "imaginary," and so we can cancel "i" and deal only with real quantities. β^2_{inv} is chosen so that $\beta c\hat{q} = |D| = |q|$. For low Mach numbers, β^2 is small and so $\lambda_\pm \sim \frac{1+\sqrt{5}}{2} D$. We then choose β_{inv} as a term that depends on D plus a cutoff to prevent β from becoming too small. This cutoff depends on a global quantity M_{ref}. We choose

$$\begin{aligned} \beta^2_{\text{inv}} &= K_1 \frac{q^2}{c^2 \hat{q}^2} + K_2 M^2_{\text{ref}}\,, \\ \beta^2_{\text{ss}} &= K_3(\text{Re}_\Delta)\beta^2_{\text{inv}}\,. \end{aligned} \tag{19}$$

For a uniform Cartesian mesh, $\frac{q^2}{c^2\hat{q}^2}$ reduces to M^2. The formula for β^2 with dual time stepping is then given by

$$\beta^2 = \beta^2_{\text{ss}} + K_\beta \left(\frac{c_t \text{Vol}}{c|S|\Delta t} \right)^p\,, \tag{20}$$

where K_1, K_2, K_3, K_β are constants, Re_Δ is the cell Reynolds number, and $|S|^2$ is a sum of the squares of all surface metrics (6 in two dimensions and 9 in three dimensions). M is the local Mach number and M_{ref} is a reference Mach number, which is representative of the free stream Mach number for low speed flows. We recalculate M_{ref} after each physical time step. Based on numerical experimentation, $p = \frac{1}{2}$ in (20) yields the most consistent results. Note that β^2 depends on $\frac{1}{\Delta t}$ even when using an implicit method for the time derivative term. We also account for viscous effects in computing β^2 [20] through K_3. Because we do not let β^2 exceed unity, the preconditioning is turned off locally in the farfield for external problems, where the cell volumes are large. Similarly, when Δt is small enough, then preconditioning is turned off globally.

When the contribution of the physical time step is large enough, then the preconditioning does not improve the convergence of the subiterations, but it is still useful for improving the accuracy of the numerical solution. Hence, we use different values of β for the preconditioning for the update procedure and for the artificial viscosity. We denote by K^{avis}_β the value of K_β in (20) when used within the artificial viscosity. For dual time stepping this will generally be smaller than the value of K^{update}_β used in the update procedure.

3. Residual smoothing

Implicit residual smoothing is a technique to smooth the residuals and allow a larger time step [6]. It also changes the damping characteristics of the scheme. In continuous form, it is given by

$$\left(I - \epsilon \frac{\partial^2}{\partial x^2}\right) R^{\text{smoothed}} = R^{\text{original}},$$

R^{original} corresponds to $(R^*)^k$ in (6). This is done in each coordinate direction and at each stage of the RK update. For a one dimensional steady state problem, ϵ is chosen to be

$$\epsilon = \frac{1}{4}\left[\left(\frac{\lambda}{\lambda^*}\right)^2 - 1\right], \tag{21}$$

where λ is the CFL number of the smoothed scheme and λ^* is the CFL number of the unsmoothed scheme. Hence, λ^* should be determined by the stability theory of the RK method being used. For the five stage RK method used here, $\lambda^* = 3.748$. However, for the various preconditioners, we frequently reduce λ^* from its theoretical value. In multi-dimensions, we also account for the ratios of the spectral radius in various directions [10, 31].

For dual time steps we need to reanalyze the determination of ϵ. As before, we denote the physical time by t and the artificial time by τ. We consider the one dimensional scalar equation, $u_\tau = u_x - \frac{c_t}{\Delta t}u$. We apply residual smoothing to the change $E = u^k - u^0$ at the k-th RK stage. This yields

$$E_i - \varepsilon(E_{i+1} - 2E_i + E_{i-1}) = \Delta\tau\left(\frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{c_t}{\Delta t}\right).$$

Using Fourier transform, we get

$$z(\theta) = \frac{i\lambda\sin(\theta) - \frac{c_t\Delta\tau}{\Delta t}}{1 + 2\varepsilon(1 - \cos(\theta))}.$$

z is the Fourier transform of the variable being updated in the RK scheme. By standard ODE analysis, the scheme is absolutely stable if z lies within the stability domain of the RK scheme. We have a difficulty since z is a complex quantity, and so the stability is dependent on the details of the shape of the stability domain. Instead, we replace this requirement by the simpler one that we only look at the imaginary and real parts of z and demand that they lie within the appropriate intervals on the imaginary and real axes, respectively. This yields

$$\left|\frac{\lambda\sin(\theta)}{1 + 2\varepsilon(1 - \cos(\theta))}\right| \le \lambda_I^*,$$
$$\frac{c_t\frac{\Delta\tau}{\Delta t}}{1 + 2\varepsilon(1 - \cos(\theta))} \le \lambda_R^*,$$

where λ_I^* and λ_R^* are the limits of the RK stability region along the imaginary and negative real axis, respectively. Usually we choose θ so as to maximize the first term. This yields

$$\cos(\theta) = \frac{2\varepsilon}{1 + 2\varepsilon},$$

and as a consequence we get

$$\varepsilon \ge \frac{(\frac{\lambda}{\lambda_I^*})^2 - 1}{4}.$$

Substituting this into the second requirement we get

$$\frac{(1+2\varepsilon)c_t\frac{\Delta\tau}{\Delta t}}{1+4\varepsilon} \le \lambda_R.$$

Define $r = \frac{\lambda_R}{c_t\frac{\Delta\tau}{\Delta t}}$. Then we require

$$\varepsilon \le \frac{1-r}{2(2r-1)}.$$

So in addition to the usual condition on ε, we have a new restriction that

$$\Delta\tau \le \frac{\lambda_R \Delta t}{c_t}\left(1+\frac{2\varepsilon}{1+2\varepsilon}\right). \tag{22}$$

In practice, we have found that instead of restricting $\Delta\tau$ to satisfy (22), we can reduce the value of λ^*. For non-preconditioned steady flow computations, we set $\lambda^* = 3.748$ in (21) for the five stage RK scheme. For a low speed preconditioned steady state calculation, this is reduced to $\lambda^* = 3.25$. For dual time steps with Jacobi preconditioning, this is changed to $\lambda^* = 2.4$. When using dual time steps with low speed preconditioning, this is further reduced to $\lambda^* = 2.0$. As shown above, the residual smoothing parameter should be proportional to $\frac{1}{\Delta t}$.

4. Jacobi preconditioning

The Jacobi preconditioning is based on adding a matrix-based artificial viscosity and then choosing P^{-1} as the terms on the diagonal (*i.e.*, the coefficient of w_{ij}). The result for a central difference scheme is

$$P_J^{-1} = \zeta I + |A| + |B| + |C|. \tag{23}$$

This approach has been proposed by Allmaras [2] and Pierce and Giles [13] for steady state flows. The good high frequency damping characteristics of the Jacobi preconditioner make it an ideal candidate for coupling with a multigrid scheme. Because this formulation connects the preconditioning with the artificial viscosity (or upwinding), the matrix P is affected by the details of the discretization. However, equation (23) has also been used with other artificial viscosities such as CUSP (see Caughey and Jameson [4]). We prefer to view the preconditioner represented by equation (23) as a matrix or characteristic inverse time step (see [26] for a similar view). A multistage, non-preconditioned RK scheme uses an artificial time step given by

$$\Delta\tau = \frac{CFL}{\zeta + \rho(A) + \rho(B) + \rho(C)}, \tag{24}$$

where ρ is the spectral radius and CFL is a number chosen to achieve stability. A matrix time step for the Jacobi preconditioner replaces this by

$$\Delta\tau = CFL\left(\zeta I + |A| + |B| + |C|\right)^{-1}. \tag{25}$$

In calculating the absolute value of the matrices, one needs to cutoff the eigenvalues to prevent them from becoming too small. We do this by not allowing any eigenvalue to be less than a given percentage of the maximum eigenvalue. Within the artificial viscosity we cutoff the acoustic eigenvalues at 30% and the convective eigenvalue at 10% of the maximum [15]. Within the Jacobi preconditioning all eigenvalues are cutoff at 30% of the maximum eigenvalue.

The preconditioning techniques described here have been incorporated in the TLNS3D code [27, 28]. The standard TLNS3D code solves the generalized thin layer Reynolds-averaged Navier-Stokes equations, and uses residual smoothing and multigrid to accelerate the convergence to a steady state. We wish to include all of these acceleration techniques when using the Jacobi preconditioning. Though some researchers have avoided

the use of residual smoothing with the Jacobi preconditioner [13, 25], we have found no difficulty in including both the residual smoothing and the Jacobi preconditioning. In fact, they complement each other since the Jacobi preconditioning is local while the residual smoothing is global in nature due to the implicit operators in each coordinate direction. This is especially important in the presence of high aspect ratio cells, which are essential for resolving boundary layers in viscous flows.

The Jacobi preconditioning for the k-th stage of a RK algorithm is given as

$$(\zeta I + |A| + |B| + |C|)\Delta w = \alpha_k CFL \cdot \mathrm{Res} \tag{26}$$

where α_k is the stage coefficient of the RK scheme. For a three dimensional problem, each of the matrices is 5×5. As suggested by Caughey and Jameson [8], and Hosseini and Alonso [5], we transform the equation to entropy variables $w_0 = (\frac{\mathrm{d}p}{\rho c}, \mathrm{d}u, \mathrm{d}v, \mathrm{d}w, \mathrm{d}S)$. This has the advantage that the flux Jacobian matrices are symmetric and also that the entropy equation reduces to a scalar equation that decouples from the others. Hence, we need only to operate with a 4×4 matrix rather than a 5×5 system, which results in appreciable savings in computational costs. The formulas for the absolute values are presented later. One then calculates the LU factors explicitly for a 4×4 matrix. Using a Cholesky decomposition for a symmetric matrix, fewer elements need to be calculated. However, four square roots are evaluated in this approach (Caughey and Jameson [4] reduced this to three square roots by clever programming). Computationally, we found that using the non-symmetric LU form required about the same computer time for this small matrix, since no square roots are required. The storage is larger for the non-symmetric decomposition. Since this matrix is not stored globally, it has an insignificant impact on memory requirements. The extra work in the Jacobi preconditioning is mainly in defining the elements of the matrix rather than the inversion, and it typically adds about 10% to the total running time for a compressible turbulent flow code.

For viscous problems, Caughey and Jameson [4] replaced the entropy variables by a transformation suggested by Abarbanel and Gottlieb [1]. This can also be expressed as using a different set of variables [19]. Even though symmetry is preserved, the resultant absolute values constitute a full 5×5 matrix. Since this substantially adds to the computing time, we used the entropy variables instead and approximate the viscous terms by a diagonal matrix corresponding to the additional term in the time step calculation of the standard code. Hence, it is just an addition to the diagonal term in (25). For high Reynolds number flows, this viscous correction is small enough that it does not justify the additional computational time required for inverting a full 5×5 matrix at all nodes.

In generalized coordinates we define the contravariant velocity $U = uS_{xx} + vS_{xy} + wS_{xz}$ where S_{ij} are the elements of the surface area tensor. In entropy variables, the flux Jacobian matrix A is given by

$$\mathbf{A} = \begin{pmatrix} U & cS_{xx} & cS_{xy} & cS_{xz} & 0 \\ cS_{xx} & U & 0 & 0 & 0 \\ cS_{xy} & 0 & U & 0 & 0 \\ cS_{xz} & 0 & 0 & U & 0 \\ 0 & 0 & 0 & 0 & U \end{pmatrix}.$$

Let λ_1, λ_2 be the eigenvalues of A and define

$$\begin{aligned} R_1 &= \frac{|\lambda_1| + |\lambda_2|}{2}, \\ R_2 &= \frac{|\lambda_1| - |\lambda_2|}{2}, \\ R_3 &= R_1 - |U|, \end{aligned}$$

and $|S| = \sqrt{S_{xx}^2 + S_{xy}^2 + S_{xz}^2}$. Define the normalized surface metrics

$$\hat{S}_x = \frac{S_{xx}}{|S|}; \qquad \hat{S}_y = \frac{S_{xy}}{|S|}; \qquad \hat{S}_z = \frac{S_{xz}}{|S|}.$$

Then the absolute value is given by (symmetric terms suppressed)

$$|A| = \begin{pmatrix} R_1 & \hat{S}_x R_2 & \hat{S}_y R_2 & \hat{S}_z R_2 & 0 \\ . & |U| + \hat{S}_x{}^2 R_3 & \hat{S}_x \hat{S}_y R_3 & \hat{S}_x \hat{S}_z R_3 & 0 \\ . & . & |U| + \hat{S}_y{}^2 R_3 & \hat{S}_y \hat{S}_z R_3 & 0 \\ . & . & . & |U| + \hat{S}_z{}^2 R_3 & 0 \\ . & . & . & . & |U| \end{pmatrix}. \tag{27}$$

To get a better intuition of the matrix $|A|$ we consider the subsonic case with $0 \le u, v, w \le c$. In Cartesian coordinates we have $S_{xx}=1$, $S_{xy}=S_{xz}=0$. Then $R_1=c$, $R_2=u$, $R_3=c-u$ and

$$|A| = \begin{pmatrix} c & u & 0 & 0 & 0 \\ u & c & 0 & 0 & 0 \\ 0 & 0 & u & 0 & 0 \\ 0 & 0 & 0 & u & 0 \\ 0 & 0 & 0 & 0 & u \end{pmatrix}.$$

Let d be the number of dimensions (2 for 2-D and 3 for 3-D flows). Then

$$|A| + |B| + |C| = \begin{pmatrix} d \cdot c & u & v & w & 0 \\ u & v+w+c & 0 & 0 & 0 \\ v & 0 & u+w+c & 0 & 0 \\ w & 0 & 0 & u+v+c & 0 \\ 0 & 0 & 0 & 0 & u+v+w \end{pmatrix}.$$

For $u, v, w \ll c$,

$$(|A| + |B| + |C|)^{-1} \sim \frac{1}{c} \mathrm{diag}\left(\frac{1}{d}, 1, 1, 1, \frac{c}{|u| + |v| + |w|}\right).$$

The inverse of the Jacobian is a diagonal matrix (up to errors of $O(M)$). On the other hand, $\Delta\tau \sim \frac{1}{cd}$. So,

$$\frac{(|A|+|B|+|C|)^{-1}}{\Delta\tau} \sim \mathrm{diag}\left(1, d, d, d, \frac{cd}{|u| + |v| + |w|}\right).$$

Hence, for most of the variables (except for entropy), there is a (maximum) factor of $d \sim 2, 3$ variation in the time step. Thus, as a matrix time step, Jacobi preconditioning mainly affects the entropy equation. However, the major advantage of the Jacobi preconditioning is the damping of the high frequencies, which is useful for multigrid convergence [18].

5. Preconditioning Squared

Let β be given by (20). We consider the simplest low speed preconditioning in entropy variables given by

$$P_0 = \mathrm{diag}(\beta^2, 1, 1, 1, 1). \tag{28}$$

We combine the low speed preconditioning with Jacobi preconditioning by starting with an artificial viscosity based on the low speed preconditioning for increased accuracy and then forming the Jacobi preconditioning for better convergence rates [18]. Let P be the low speed preconditioning, and let the physical time derivative be represented by (2). Then the preconditioned scheme (showing only the second-order dissipation) is given by

$$\begin{aligned} P_J^{-1} &= {P_0}^{-1}\left(\zeta P_0 + |P_0 A| + |P_0 B| + |P_0 C|\right), \\ P_J^{-1}\Delta w &= \frac{c_t w^{n+1} - E(w^n, w^{n-1}, ...)}{\Delta t} + F_x + G_y + H_z \\ &\quad - \frac{\Delta x}{2}\left[({P_0}^{-1}|P_0 A| w_x)_x + ({P_0}^{-1}|P_0 B| w_y)_y + ({P_0}^{-1}|P_0 C| w_z)_z\right] \equiv \mathrm{Res}. \end{aligned} \tag{29}$$

We redefine the entropy variables for the preconditioned scheme so as to include β. So,

$$\mathrm{d}\hat{w}_0 = \left(\frac{\mathrm{d}p}{\rho\beta c}, u, v, w, T\right). \tag{30}$$

In these variables the matrices P_0 and $|P_0 A| + |P_0 B| + |P_0 C|$ are symmetric. The eigenvalues of each matrix are given by

$$\begin{aligned} \lambda_\pm &= \frac{(\beta^2+1)U \pm \sqrt{(\beta^2-1)^2U^2 + 4\beta^2 c^2}}{2}, \\ \lambda_{\max} &= \frac{(\beta^2+1)|U| + \sqrt{(\beta^2-1)^2U^2 + 4\beta^2 c^2}}{2}. \end{aligned}$$

Define

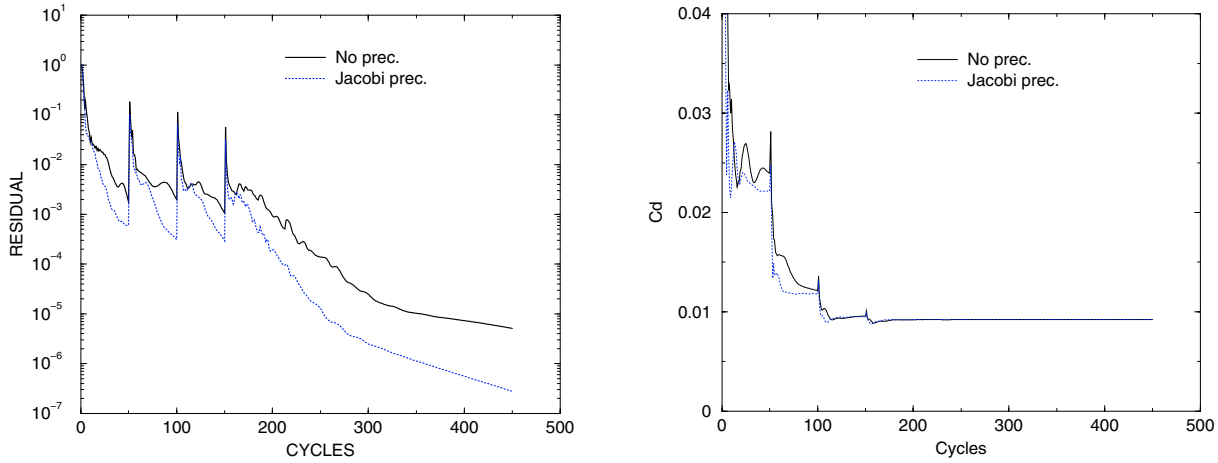
$$\begin{aligned} &|\Lambda_+| = \max(|\lambda_+|, \epsilon_n \lambda_{\max}), \qquad |\Lambda_-| = \max(|\lambda_-|, \epsilon_n \lambda_{\max}), \qquad |\Lambda_0| = \max(|U|, \epsilon_l \lambda_{\max}), \\ &R_1 = \frac{(\lambda_+ - U)|\Lambda_+| - (\lambda_- - U)|\Lambda_-|}{\lambda_+ - \lambda_-}, \qquad S_1 = \frac{(\lambda_+ - U)|\Lambda_-| - (\lambda_- - U)|\Lambda_+|}{\lambda_+ - \lambda_-}, \\ &R_2 = \beta c \frac{|\Lambda_+| - |\Lambda_-|}{\lambda_+ - \lambda_-}, \qquad R_3 = S_1 - |\Lambda_0|, \end{aligned}$$

where U is the contravariant velocity in each direction and ϵ_n and ϵ_l are constants. Typical values are $\epsilon_n = 0.3$ and $\epsilon_l = 0.1$ for the artificial viscosity and $\epsilon_n = \epsilon_l = 0.3$ within the Jacobi preconditioning. Then

$$|P_0 A| = \begin{pmatrix} R_1 & \hat{S}_x R_2 & \hat{S}_y R_2 & \hat{S}_z R_2 & 0 \\ . & |\Lambda_0| + \hat{S}_x{}^2 R_3 & \hat{S}_x \hat{S}_y R_3 & \hat{S}_x \hat{S}_z R_3 & 0 \\ . & . & |\Lambda_0| + \hat{S}_y{}^2 R_3 & \hat{S}_y \hat{S}_z R_3 & 0 \\ . & . & . & |\Lambda_0| + \hat{S}_z{}^2 R_3 & 0 \\ . & . & . & . & |\Lambda_0| \end{pmatrix}. \tag{31}$$

The same formulas hold for $|P_0 B|$ and $|P_0 C|$ with the appropriate surface metrics. The update scheme, (26), then becomes

$$\Delta w = \frac{\partial w}{\partial w_0}\left(\zeta P_0 + |P_0 A| + |P_0 B| + |P_0 C|\right)^{-1} P_0 \frac{\partial w_0}{\partial w} \alpha_k \Delta\tau \mathrm{Res}. \tag{32}$$



(a) Residual (b) Drag

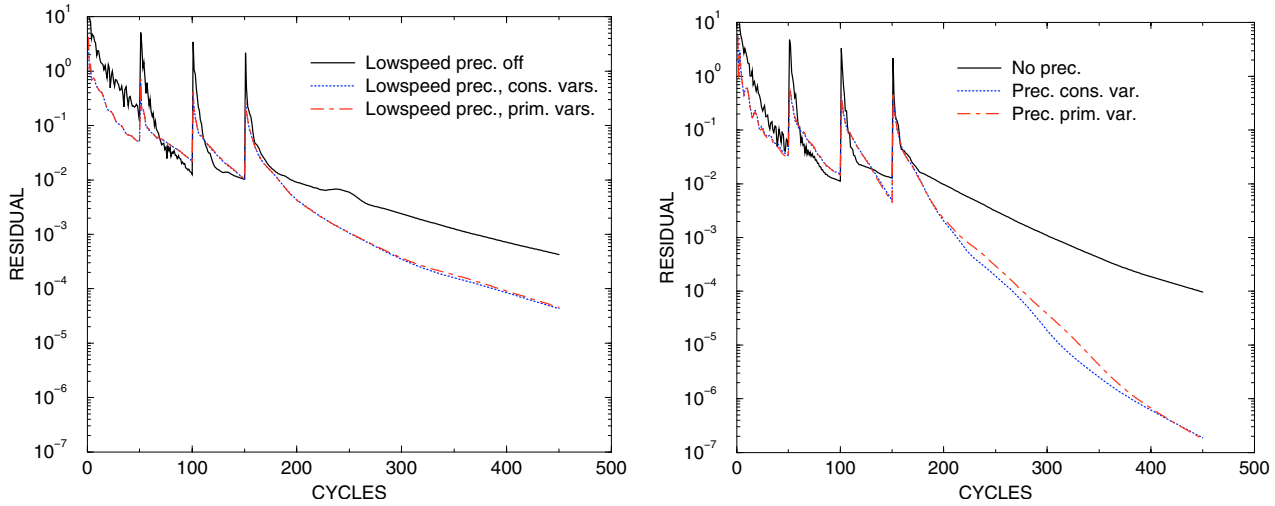
FIGURE 1. RAE2822 convergence history, $M_\infty = 0.73$.

6. RESULTS

The results are computed using TLNSD, a finite volume central difference code augmented by a matrix artificial viscosity. The equations are advanced in time with a dual time stepping scheme. A five stage RK scheme, accelerated by residual smoothing and multigrid [9, 15, 27], is used for advancing the solutions in pseudo-time. Second order BDF formulation is used for discretizing the physical time derivatives. Without low speed preconditioning, we use a RK scheme with the coefficients (0.25, 0.16667, 0.375, 0.5, 1.0). With low speed preconditioning, the RK coefficients are chosen as (0.25, 0.18, 0.40, 0.51, 1.0). The artificial viscosity is partially updated only on the odd stages of the multistage method using fractions of 0.56 and 0.44 on the third and fifth stages, respectively. There is also a small dependence of β on the viscous terms for the artificial viscosity [20]. The other parameters are identical with the exception of $\frac{\lambda}{\lambda^*}$ in (21) as described above. The residual smoothing coefficients depend on the aspect ratio [10, 31]. For all preconditioning cases, $\beta^2_{\min} = M^2_\infty$.

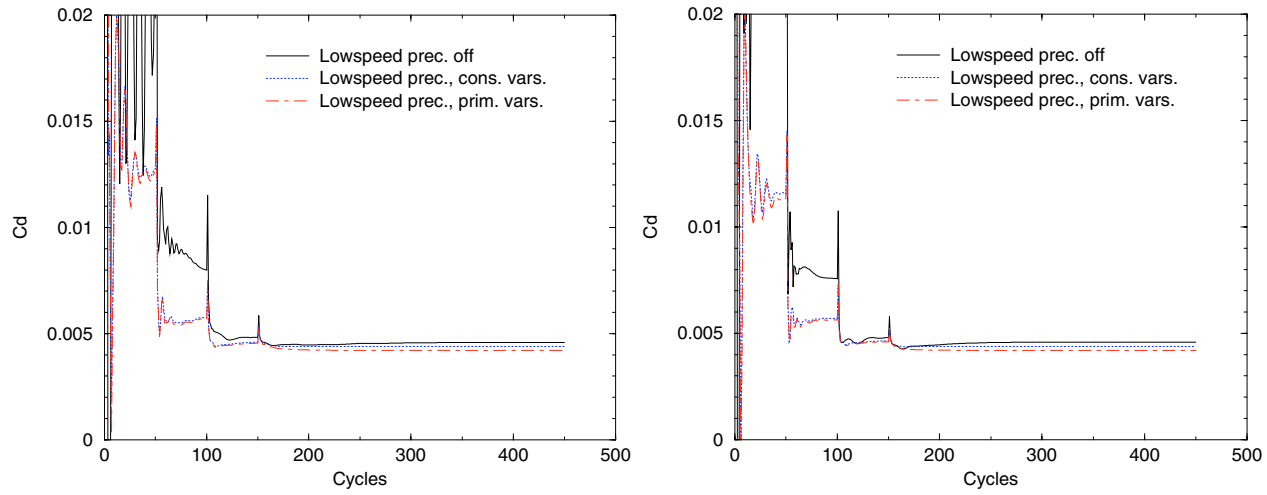
6.1. **RAE2822 airfoil**

We first examine the use of both Jacobi and low speed preconditioners for steady flow. We consider a two-dimensional RAE2822 airfoil using a 320×64 C grid. We first consider a transonic case with an inflow Mach number, $M_\infty = 0.73$, and an angle of attack, $\alpha = 2.79°$. The Reynolds number is 6.5 million, and the turbulent flow is simulated with a Baldwin-Lomax turbulence model. We use a Full Multi Grid (FMG) scheme [3] with 50 iterations on each of the three coarse meshes and 300 iterations on the finest mesh. The only algorithmic change associated with the Jacobi preconditioner is in the update stage where the residual is multiplied by the low speed preconditioning matrix, P, followed by either $\Delta\tau$ or else by $(|PA|+|PB|)^{-1}$. In Figure 1a we compare the convergence rate for the standard code with that produced with Jacobi preconditioning. The residual shown in Figure 1a reflects the change from n to $n+1$ in ρu. For the transonic flow case, we do not use low speed preconditioning. The Jacobi preconditioning results in an improvement in the convergence rate. In Figure 1b the drag coefficient for the same case also shows improved convergence with Jacobi preconditioning. The Jacobi preconditioning has no impact on the final value of the drag, as expected.



(a) no Jacobi preconditioning (b) Jacobi preconditioning

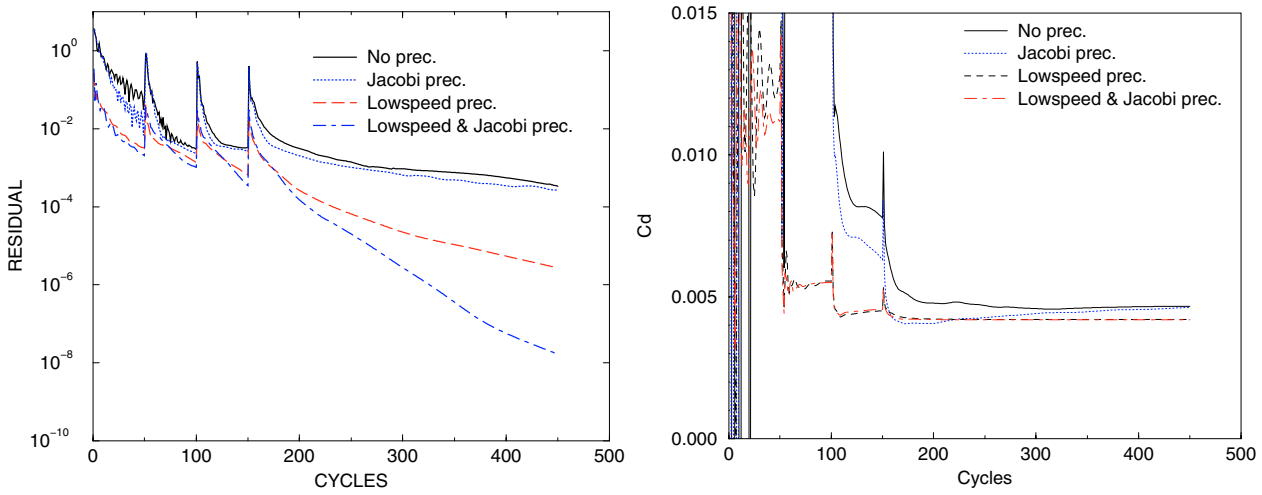
FIGURE 2. RAE2822 residual history, $M_\infty = 0.20$.



(a) no Jacobi preconditioning (b) Jacobi preconditioning

FIGURE 3. RAE2822 drag history, $M_\infty = 0.20$.

We next compute a low speed case with $M_\infty = 0.2$. In Figure 2a we compare the impact of low speed preconditioning without Jacobi preconditioning on the convergence of the residuals. The residual shown reflects the change from n to $n+1$ in ρu for the conservation variable and in u for the primitive variable formulation of preconditioning. Since the density is almost constant at low Mach numbers, these residuals are comparable in magnitude. The low speed preconditioning improves the convergence rate significantly. The two preconditioners based on the different set of variables are almost indistinguishable. In Figure 2b we show the results using the Jacobi preconditioner both by itself and combined with low speed preconditioning. The Jacobi preconditioning improves both the non-preconditioned and the low speed preconditioned cases. The cpu time required for the low speed preconditioned code is approximately an additional 20% of the total run time compared to



(a) Residual (b) Drag

FIGURE 4. RAE2822 convergence history, $M_\infty = 0.05$.

the baseline code. This includes both the contributions to the update and the artificial viscosity. The Jacobi preconditioning requires an additional 10% cpu time. The gains in residual convergence are more significant with Jacobi preconditioning, especially if one wishes to reduce the residual to very low levels. In Figures 3a and 3b we show the convergence of the drag. The steady state drag is the same for the non-preconditioned algorithm and Jacobi preconditioning since we only change the update procedure but not the residual. However, the low speed preconditioning is included in the artificial dissipation and so the low speed preconditioning changes the steady state numerical solution. Turkel, Fiterman and van Leer [23] have proven that only the low speed preconditioned residual gives the correct solution as the Mach number approaches zero. The theory is based on a linearized system and so it does not distinguish between preconditioning using conservation or primitive variables.

We conclude, in Figures 4a and 4b, with results for an inflow Mach number of 0.05. Since both preconditioners produced similar results, we include only the one based on primitive variables. We see that the Jacobi preconditioning by itself helps relatively little for this low Mach number flow. In contrast, using the low speed preconditioning gives a large improvement in the residual convergence. The combined low speed and Jacobi preconditioning gives a dramatic improvement, yielding 8 orders of magnitude decrease in the residual in 300 multigrid cycles. The residual is reduced by about 11 orders of magnitude. For a five stage RK formula, this is equivalent to 1500 explicit sweeps through the grid. Such rapid convergence represents a significant improvement for low-speed, viscous, turbulent flow computations on high aspect ratio grids.

6.2. **Vortex**

We now consider time dependent flow using dual time steps. We begin with an inviscid case for which the exact solution is known, and so we can assess the effect of the preconditioners on the accuracy. We solve for an inviscid vortex propagating in the x direction. We first define several quantities in terms of x_0, which represents the center of vortex at time, $t=0$, and c_0 and c_1, which are free parameters.

$$c_2 = \frac{\gamma - 1}{2\gamma} \frac{\rho_0}{p_0} c_0^2 c_1^2,$$
$$arg = 1 - (x - x_0 - c_0 t)^2 - y^2.$$

TABLE 1. Vortex Motion: $M = 0.05$.

(a) Error in u at T=60

CFL_{phys}	no precondition	precondition
0.25	4.547×10^{-4}	2.048×10^{-4}
0.50	4.235×10^{-4}	1.972×10^{-4}
1.0	4.480×10^{-4}	3.808×10^{-4}
2.0	6.854×10^{-4}	1.038×10^{-3}
3.0	9.062×10^{-4}	1.238×10^{-3}
4.0	1.006×10^{-3}	1.265×10^{-3}
5.0	1.011×10^{-3}	1.173×10^{-3}

(b) Error at 10 physical time steps

CFL_{phys}	no precondition	precondition
0.25	7.101×10^{-5}	3.442×10^{-5}
0.50	1.000×10^{-4}	3.958×10^{-5}
1.0	1.581×10^{-4}	5.285×10^{-5}
2.0	3.402×10^{-4}	3.858×10^{-4}
3.0	6.666×10^{-4}	7.976×10^{-4}
4.0	8.865×10^{-4}	1.045×10^{-3}
5.0	1.011×10^{-3}	1.173×10^{-3}

The exact solution for this problem in the non-dimensional variables of TLNS3D code is as follows:

$$
\begin{aligned}
u(x,y,t) &= c_0\left(1 - c_1 y \mathrm{e}^{0.5\ arg}\right), \\
v(x,y,t) &= c_0 c_1 (x - x_0 - c_0 t)\mathrm{e}^{0.5\ arg}, \\
\rho(x,y,t) &= \rho_0\left(1 - c_2 \mathrm{e}^{arg}\right)^{\frac{1}{\gamma-1}}, \\
p(x,y,t) &= p_0\left(1 - c_2 \mathrm{e}^{arg}\right)^{\frac{\gamma}{\gamma-1}}.
\end{aligned}
$$

With the constants

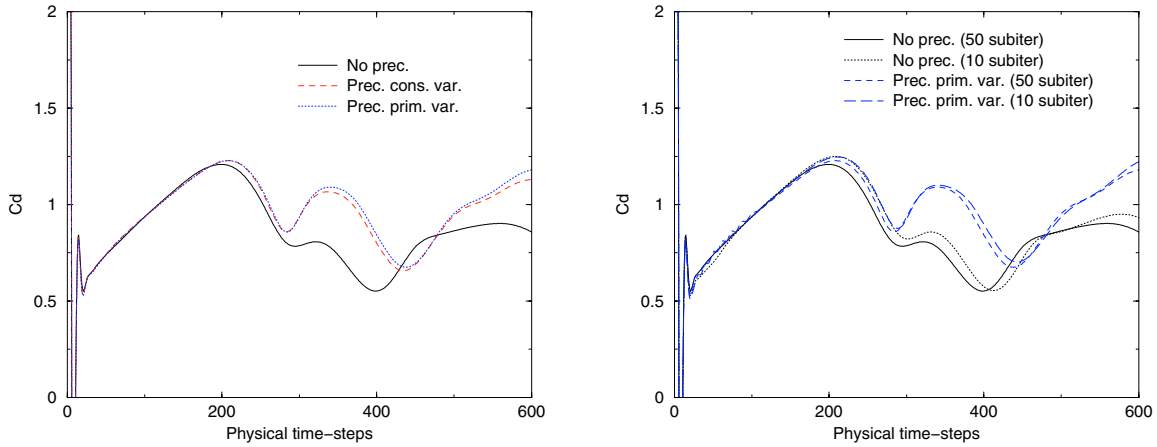
$$
\begin{aligned}
c_0 &= \sqrt{\gamma} M \qquad c_1 = \frac{1}{2\pi} \qquad x_0 = 10 \\
\rho_0 &= 1 \qquad p_0 = 1 \quad \text{So} \quad c_2 = \frac{\gamma - 1}{2} c_1^2 M^2.
\end{aligned}
$$

This vortex also satisfies the incompressibility condition $u_x + v_y = 0$. We define $CFL_{\text{phys}} = \frac{c_0 \Delta t}{\Delta x}$. The computational domain is $5 \le x \le 35$, $-5 \le y \le 5$ with a uniformly spaced Cartesian grid with 97x33 nodes yielding $\Delta x = \Delta y = .3125$. We discretize the Euler equations using a fourth order difference. We choose $M = 0.05$. So $CFL = 1$ corresponds to $\Delta t = \frac{6.25}{\sqrt{1.4}} \sim 5.28$. We perform 50 subiterations at each physical time step.

In Table 1a we present the L_2 errors in u for a fixed physical time of 60 (*i.e.* 60 physical time steps at $CFL = 1$). So, analytically the center of the vortex moves horizontally from $x = 10$ to $x = 28.75$. This enables us to see the growth of the errors for a larger time. In Table 1b we present the error after 10 physical time steps. Hence, different CFL correspond to different physical times. We compare the non-preconditioned and preconditioned algorithms where the preconditioned algorithm is based on conservation variables. Use of primitive variables did not change the results very much.

Note that for $M = 0.05$ flow at a physical time step corresponding to CFL = 1, $M^2 \sim 2.5 \times 10^{-3}$ while $(\frac{c_t \text{Vol}}{c|S|^2 \Delta t})^{\frac{1}{2}} \sim 3.8 \times 10^{-1}$. Thus, the physical time term dominates other contributions in the calculation of β, (20), by two orders of magnitude. On the other hand for the same time step we have $\frac{1}{\Delta \tau_{\text{ss}}} \sim .53$, while $\frac{K_\tau}{c_t \Delta t} \sim .13$ and so the contribution of the physical time step to the artificial time step in (18) is small. Although $K_\tau = 0$ is stable, we choose $K_\tau = 1$ in (18) for improved robustness.

As described above, we consider the use of two separate values for β, for the convergence acceleration and the artificial dissipation. The only difference in (19) is the constant K_β. To distinguish them, we denote as K_β^{avis} the constant which affects the magnitude of the artificial viscosity. Normally, we choose K_β^{avis} as small as possible without destroying convergence so as to gain accuracy. The constant K_β^{update} used in the update stage does not affect accuracy and is chosen only to improve the convergence rate. We choose $K_\beta^{\text{update}} = 0.3$ for all the cases.



(a) 50 subiterations (b) 10 subiterations

FIGURE 5. Effects of subiterations on time history of c_L for NACA0012, $\alpha = 30°$.

The standard value chosen for $K_\beta^{\text{avis}} = 0.015$, which is more than an order of magnitude smaller than K_β^{update} and, therefore, it minimizes the magnitude of the artificial viscosity. Increasing K_β^{avis} for these larger physical time steps reduces the error for the preconditioned case to be similar to the non-preconditioned scheme. Hence, the formula (20) gives the minimum β in the artificial viscosity for stability but not necessarily the optimal value. For $CFL \leq 1$ the optimal K_β^{avis} is the smallest for which the subiterations are stable. However, for a larger physical time step, we get better accuracy with a larger coefficient in the artificial viscosity than the minimum required. Hence, preconditioning improves the accuracy for $CFL_{\text{phys}} \leq 1$ but the accuracy deteriorates for larger physical time steps.

6.3. **Time Dependent NACA0012**

We next consider turbulent flow around a NACA0012 airfoil. The grid is an O mesh with 141×61 nodes. The angle of attack is $\alpha = 30°$, and the inflow Mach number is $M_\infty = 0.05$. The high angle of attack causes the flow to be unsteady especially in the wake region. We use the Spalart-Allmaras one equation turbulence model. The solution is calculated for 600 physical time cycles. We compute 50 cycles (each a 5 stage RK with multigrid and residual smoothing) within each physical time cycle. In Figure 5a we show the lift as a function of the physical time. We see that for smaller times, the solutions are essentially the same. However, for longer times, the preconditioning results differ from that of the non-preconditioned code. We stress that the use of preconditioning affects the accuracy of the solution and so changes the values of the lift and drag. There is no analytic solution for this problem and, hence, no easy way to determine the correct solution. The proof of Turkel, Fiterman and van Leer [23], that preconditioning improves the accuracy for low Mach number flows, applies to steady state flows. Nevertheless, the analysis applies to time dependent flows based on similar scaling arguments. This lends some credence to the results obtained with the preconditioned code. There are also smaller differences between the preconditioned codes based on primitive or conservation variables. In Figure 5b we display the same case where only 10 subiterations are done for each physical time step. As time progresses we see that in all cases, the lift begins to differ. However, the differences are much smaller with the preconditioned algorithm, implying that the preconditioned algorithm is more robust with respect to lowering the number of subiterations. This is important when the number of iterations is computed based on some error criteria rather than being fixed in advance. The differences in lift are caused by the different level of artificial viscosity between the non-preconditioned and preconditioned schemes. To see this even more clearly, we vary the value of β^2 used in the artificial viscosity of the preconditioned scheme. The coefficient in β^2 in (19) used in the

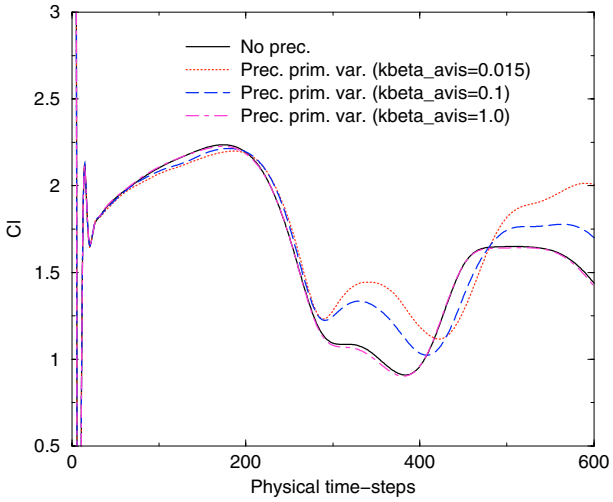


FIGURE 6. Effect of changing K_β^{avis}.

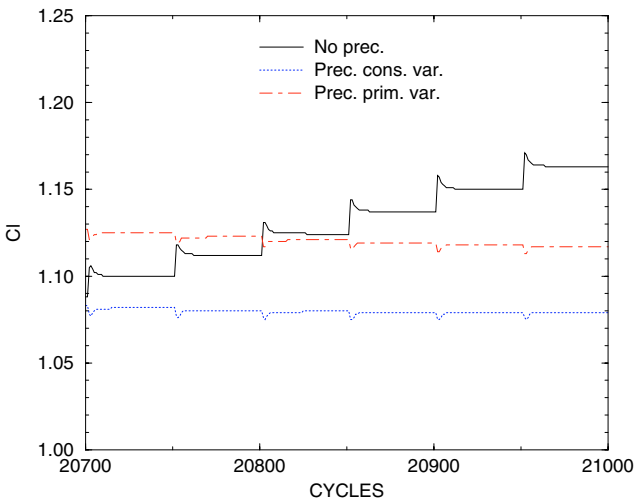
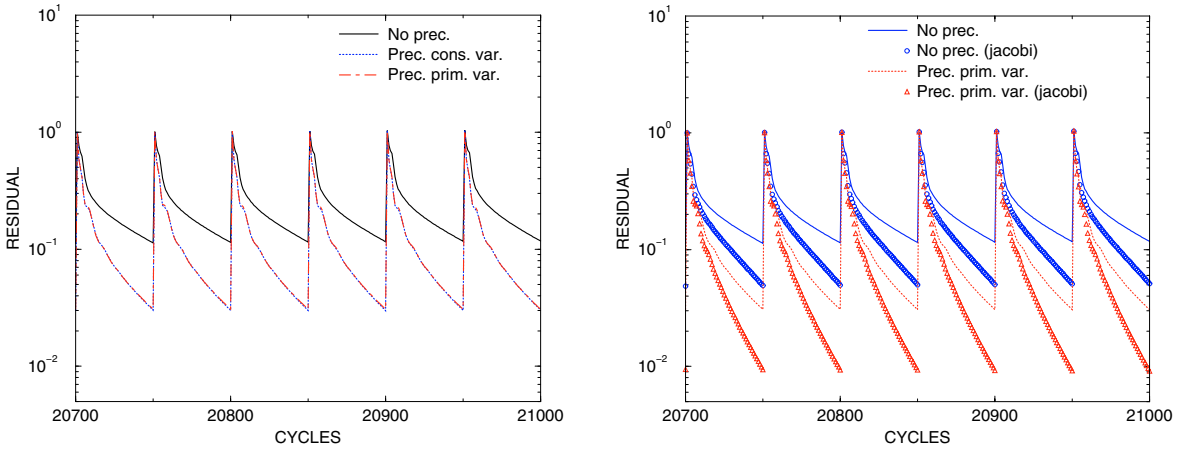


FIGURE 7. Convergence of c_L within subiterations.

update procedure is fixed at $K_\beta^{\text{update}} = 0.3$. In Figure 6 we vary between $K_\beta^{\text{avis}} = 0.015$ (standard) to $K_\beta^{\text{avis}} = 0.1$ to $K_\beta^{\text{avis}} = 1.0$. We compare the preconditioning scheme, based on primitive variables, with the various values of K_β and the non-preconditioned code. We see that at the highest level, $K_\beta^{\text{avis}} = 1.0$, the preconditioned and non-preconditioned values for c_L basically coincide. As with the vortex, the dominant term in (20) is the physical time dependent term $(\frac{c_t \text{Vol}}{c|S|^2 \Delta t})^{\frac{1}{2}}$ which varies between 0.2 and 1.3 in the computational domain. Hence, when $K_\beta^{\text{avis}} = 0.015$, the preconditioning is changing the artificial viscosity. However, when $K_\beta^{\text{avis}} = 1.0$, then β^2 becomes greater than 1 over large portions of the domain and is capped at $\beta^2 = 1$. This means there is no preconditioning in the artificial viscosity over much of the domain. The contribution of the physical time step, including the volume of the cell, towards the artificial time step (18) also varies over the grid. At other grid points it is comparable or smaller than the steady state contributions. As a consequence we chose $K_\tau = 1$ for preconditioning with conservation variables even though an implicit formula was used. For the primitive variables, we do not use the implicit term $\frac{w^{k+1} - w^k}{\Delta t}$ in (4) and so chose $K_\tau = 2$. Even though this decreases the artificial time step, nevertheless, we still achieve a better convergence rate within the subiterations.

In Figure 7 we display the lift for the case with 50 subiterations per physical time step. We see that the preconditioning changes the calculated value of lift. All the methods converge the lift to within graphical accuracy at each physical time step. However, the preconditioning is still converging faster. In Figure 8a we display the residual for the case with 50 subiterations per physical time step (u residual for preconditioning based on primitive variables and ρu residual for all other cases). The residuals are normalized so that they start at 1.0. The time frame period is about half way through the 600 physical time cycles. Low speed preconditioning improves the convergence. There is a negligible difference between the two preconditioners based on either conservation or primitive variables. Furthermore, we see that displaying the residuals of u or ρu makes no difference. We next add Jacobi preconditioning to the previous computations, with and without low speed preconditioning. To avoid too many graphs, we display only the low speed preconditioning based on primitive variables. In Figure 8b we see that adding the Jacobi preconditioning improves the convergence rate of both the non-preconditioned and low speed preconditioned algorithms.

In conclusion for this time dependent case the low speed preconditioning improves the convergence rate of the residual. Jacobi preconditioning further improves the convergence rate. No difference between various sets of variables was found. However, both the preconditioning and the set of variables did affect the value of the lift and drag.



(a) Effect of low speed preconditioning (b) Effect of combined low speed and Jacobi preconditioning

FIGURE 8. Residual history, within subiterations, for NACA0012.

7. Concluding remarks

Jacobi and low speed preconditioning techniques have been developed for a central difference algorithm to treat both steady state and time dependent problems. Jacobi preconditioning is shown to improve the efficiency of the baseline TLNS3D code for steady flow over a RAE 2822 airfoil at transonic Mach numbers. The efficiency gain is obtained on top of the efficiency gained by the residual smoothing and multigrid acceleration techniques which are integral parts of the TLNS3D code. For lower speed flows, the low speed preconditioning improves the convergence rate while the Jacobi preconditioner by itself does not improve the convergence rate. The convergence rate is significantly improved by combining the Jacobi and low speed preconditionings. In addition, for low Mach number flows, the low speed preconditioning improves the accuracy of the steady state solution while the Jacobi preconditioning does not affect the steady state.

If the physical time step, within dual time stepping, is sufficiently small, then the preconditioning does not improve the convergence rate. In this case, we choose $\beta = 1$, thereby turning off the preconditioning in the update stage. However, for typical parameters, as used in a NACA 0012 test case, the low speed preconditioner did improve the convergence of the subiterations towards the pseudo-steady state. The Jacobi preconditioner further improved the convergence rate within each physical time step. In all cases the low speed preconditioning affects the artificial viscosity. To minimize the magnitude of the artificial viscosity, we use different values of β in the update stage and the artificial viscosity.

References

[1] S. Abarbanel and D. Gottlieb, Time splitting for two and three-dimensional Navier-Stokes equations with mixed derivatives. *J. Comput. Phys.* **41** (1981) 1–33.
[2] S. Allmaras, *Analysis of a Local Matrix Preconditioner for the 2-D Navier-Stokes Equations*. AIAA Paper 1993-3330 (1993).
[3] A. Brandt, Multi-level adaptive solutions to boundary value problems. *Math. Comp.* **31** (1977) 333–390.
[4] D.A. Caughey and A. Jameson, *Fast Preconditioned Multigrid Solution of the Euler and Navier-Stokes Equations for Steady Compressible Flows*. AIAA Paper 2002-0963 (2002).
[5] K. Hosseini and J.J. Alonso, *Practical Implementation and Improvement of Preconditioning Methods for Explicit Multistage Flow Solvers*. AIAA Paper 2004-0763 (2004).
[6] A. Jameson, The Evolution of Computational Methods in Aerodynamics. *ASME J. Appl. Mech.* **50** (1983) 1052–1070.
[7] A. Jameson, *Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows past Airfoils and Wings*. AIAA Paper 1991-1596 (1991).
[8] A. Jameson and D.A. Caughey, *How Many Steps are Required to Solve the Euler equations of Steady, Compressible Flow: In Search of a Fast Solution Algorithm*. AIAA Paper 2001-2673 (2001).

[9] A. Jameson, W. Schmidt and E. Turkel, *Numerical Solutions of the Euler Equations by a Finite Volume Method using Runge-Kutta Time-Stepping Schemes.* AIAA Paper 1981-1259 (1981).
[10] L. Martinelli and A. Jameson, *Validation of a Multigrid Method for the Reynolds Averaged Equations.* AIAA Paper 1988-0414 (1988).
[11] N.D. Melson and M.D. Sanetrik, Multigrid Acceleration of Time-Accurate Navier-Stokes Calculations, in *7th Copper Mountain Conference on Multigrid Methods* (1995).
[12] S.A. Pandya, S. Venkateswaran and T.H. Pulliam, *Implementation of Preconditioned Dual-Time Procedures in OVERFLOW.* AIAA paper 2003-0072 (2003).
[13] N.A. Pierce and M.B. Giles, Preconditioned multigrid methods for compressible flow codes on stretched meshes. *J. Comput. Phys.* **136** (1997) 425–445.
[14] J.S. Shuen, K.H. Chen and Y.H. Choi, *A Time-Accurate Algorithm for Chemical Non-Equilibrium Viscous Flows at All Speeds.* AIAA Paper 1992-3639 (1992).
[15] R.C. Swanson and E. Turkel, On central difference and upwind schemes. *J. Comput. Phys.* **101** (1992) 292–306.
[16] E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations. *J. Comput. Phys.* **72** (1987) 277–298.
[17] E. Turkel, A review of preconditioning methods for fluid dynamics. *Appl. Numer. Math.* **12** (1993) 257–284.
[18] E. Turkel, *Preconditioning-Squared Methods for Multidimensional Aerodynamics.* AIAA Paper 1997-2025 (1997).
[19] E. Turkel, Preconditioning Techniques in Computational Fluid Dynamics. *An. Rev. Fluid Mech.* **31** (1999) 385–416.
[20] E. Turkel, *Robust Preconditioning for Steady and Unsteady Viscous Flows.* AIAA Paper 2002-0962 (2002).
[21] E. Turkel and V.N. Vatsa, Effect of artificial viscosity on three-dimensional flow solutions. *AIAA Journal* **32** (1993) 39–45.
[22] E. Turkel and V.N. Vatsa, *Choice of Variables and Preconditioning for Time Dependent Problems.* AIAA Paper 2003-3692 (2003).
[23] E. Turkel, A. Fiterman and B. van Leer, Preconditioning and the Limit to the Incompressible Flow Equations, in *Computing the Future: Frontiers of Computational Fluid Dynamics 1994*, D.A. Caughey and M.M. Hafez Eds., Wiley Publishing (1994) 215–234.
[24] E. Turkel, V.N. Vatsa and R. Radespiel, *Preconditioning Methods for Low Speed Flow.* AIAA Paper 1996-2460 (1996).
[25] E. Turkel, V.N. Vatsa and V. Venkatakrishnan, *Uni-directional Implicit Acceleration Techniques. 14th AIAA Computational Fluid Dynamics Conference.* AIAA paper 1999-3265 (1999).
[26] B. van Leer, W.T. Lee and P.L. Roe, *Characteristic Time-Stepping or Local Preconditioning of the Euler Equations.* AIAA Paper 1991-1552 (1991).
[27] V.N. Vatsa and B.W. Wedan, Development of a Multigrid Code for 3-d Navier-Stokes Equations and its Application to a Grid-refinement Study. *Comput. Fluids* **18** (1990) 391–403.
[28] V.N. Vatsa, M.D. Sanetrik and E.B. Parlette, A Multigrid Based Multiblock Flow Solver for Practical Aerodynamic Configurations, in *Computing the Future: Frontiers of Computational Fluid Dynamics 1994*, D.A. Caughey and M.M. Hafez Eds., Wiley Publishing (1994) 414–447.
[29] S. Venkateswaran and L. Merkle, *Dual Time Stepping and Preconditioning for Unsteady Computations.* AIAA Paper 1995-0078 (1995).
[30] S. Venkateswaran, D. Li and L. Merkle, *Influence of Stagnation Regions on Preconditioned Solutions at Low Speeds.* AIAA Paper 2003-0435 (2003).
[31] L.B. Wigton and R.C. Swanson, Variable Coefficient Implicit Residual Smoothing, *12th International Conference on Numerical Methods in Fluid Dynamics* (1990).
[32] J.P. Withington, J.S. Shuen and V. Yang, *A Time Accurate, Implicit Method for Chemically Reacting Flows at All Mach Numbers.* AIAA Paper 1991-0581 (1991).