AIAA JOURNAL
Vol. 34, No. 7, July 1996

Data-Parallel Lower-Upper Relaxation Method for the Navier–Stokes Equations

Michael J. Wright,* Graham V. Candler,† and Marco Prampolini‡
University of Minnesota, Minneapolis, Minnesota 55455

The lower-upper symmetric Gauss–Seidel method is modified for the simulation of viscous flows on massively parallel computers. The resulting diagonal data-parallel lower-upper relaxation (DP-LUR) method is shown to have good convergence properties on many problems. However, the convergence rate decreases on the high cell aspect ratio grids required to simulate high Reynolds number flows. Therefore, the diagonal approximation is relaxed, and a full matrix version of the DP-LUR method is derived. The full matrix method retains the data-parallel properties of the original and reduces the sensitivity of the convergence rate to the aspect ratio of the computational grid. Both methods are implemented on the Thinking Machines CM-5, and a large fraction of the peak theoretical performance of the machine is obtained. The low memory use and high parallel efficiency of the methods make them attractive for large-scale simulation of viscous flows.

Introduction

THE numerical simulation of large complex flowfields can be a computationally intensive task, especially when viscous flows are modeled. The large disparity between the length scales in high Reynolds number flows can result in a very stiff equation set, which usually requires an implicit method to converge to a steady-state solution in a reasonable time.

The large cost associated with solving the Navier–Stokes equations makes the use of a massively parallel supercomputer (MPP) attractive, since such machines have a very large peak performance. However, it is difficult to efficiently implement most implicit methods on an MPP, since a true implicit method requires the inversion of a large sparse matrix, which involves a great deal of interprocessor communication. The traditional solution to this problem is to perform some sort of domain decomposition, which reduces the amount of communication required by solving the implicit problem on a series of local subdomains. This approach has been used with good success, but the resulting algorithms can become costly and complicated because of load balancing and boundary update issues.[1]

Another approach is to seek an implicit method that would be amenable to data-parallel implementation without domain decomposition. Such an algorithm would then be readily portable to a wide variety of parallel architectures, since it is relatively easy to run a data-parallel code on a message-passing machine. The lower-upper symmetric Gauss–Seidel (LU-SGS) method of Yoon and Jameson[2] is a good starting point, because it makes some approximations to the implicit problem that eliminate the need for large block matrix inversions. Candler et al.[3] have shown that it is possible to modify the LU-SGS method, making it almost perfectly data parallel for inviscid flows. The resulting data-parallel lower-upper relaxation (DP-LUR) method replaces the diagonal Gauss–Seidel sweeps of the LU-SGS method with a series of pointwise relaxation steps. With this method all data dependencies are removed, and the computation of each relaxation step becomes perfectly data parallel. Also, the formulation of the method ensures proper load balancing at all times, eliminating the need for a domain decomposition. The resulting algorithm is simple to implement and to use in either data-parallel or message-passing mode and has been shown to be efficient for the large-scale simulation of inviscid flows. A Fortran90 implementation of the method has been tested on two massively parallel architectures, where it achieves a large percentage of the peak theoretical performance of each machine and converges to a steady-state solution in fewer iterations than the LU-SGS algorithm.

The focus of the current paper is the application of the DP-LUR method to the Navier–Stokes equations. The steps necessary to parallelize and diagonalize the implicit viscous terms are discussed. The resulting diagonal DP-LUR method works well for low Reynolds number flows. However, the method exhibits severely degraded convergence properties on the high aspect ratio grids necessary to resolve the boundary layer of high Reynolds number flows, which is a well-known attribute of all LU-SGS type schemes.[4–6] The underlying reasons for this slow convergence are discussed, and modifications are proposed to alleviate the problem, resulting in the full matrix DP-LUR algorithm. The paper then presents comparative results for both of these methods and discusses implementation and performance issues on the Thinking Machines CM-5.

Diagonal DP-LUR

The fully implicit form of the two-dimensional Navier–Stokes equations in curvilinear coordinates is

$$\frac{U^{n+1}-U^{n}}{\Delta t}+\frac{\partial \tilde{F}^{n+1}}{\partial \xi}+\frac{\partial \tilde{G}^{n+1}}{\partial \eta}=0$$

where U is the vector of conserved quantities and $\tilde{F}$ and $\tilde{G}$ are the flux vectors in the ξ (body-tangential) and η (body-normal) directions. The flux vectors can be split into convective and viscous parts,

$$\tilde{F}=F+F_v, \qquad \tilde{G}=G+G_v$$

If we focus on the inviscid problem for now, we can linearize the flux vector using

$$F^{n+1}\simeq F^{n}+\left(\frac{\partial F}{\partial U}\right)^{n}(U^{n+1}-U^{n})=F^{n}+A^{n}\delta U^{n}$$

$$G^{n+1}\simeq G^{n}+B^{n}\delta U^{n}$$

We then split the fluxes according to the sign of the eigenvalues of the Jacobians

$$F=A_{+}U+A_{-}U=F_{+}+F_{-}$$

Received Oct. 26, 1995; revision received March 8, 1996; accepted for publication March 11, 1996. Copyright © 1996 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Graduate Research Assistant, Department of Aerospace Engineering and Mechanics and Army High Performance Computing Research Center. Student Member AIAA.

†Associate Professor, Department of Aerospace Engineering and Mechanics and Army High Performance Computing Research Center. Member AIAA.

‡Visiting Fellow, Department of Aerospace Engineering and Mechanics.

to obtain the upwind finite volume representation

$$\begin{aligned}
&\delta U^n_{i,j} \\
&+ \frac{\Delta t}{V_{i,j}}\Big[\Big(A_{+i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U_{i,j} - A_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U_{i-1,j}\Big) \\
&+ \Big(A_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U_{i+1,j} - A_{-i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U_{i,j}\Big) \\
&+ \Big(B_{+i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U_{i,j} - B_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U_{i,j-1}\Big) \\
&+ \Big(B_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U_{i,j+1} - B_{-i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U_{i,j}\Big)\Big]^n \\
&= \Delta t R^n_{i,j}
\end{aligned} \tag{1}$$

where $R^n_{i,j}$ is the change of the solution as a result of the fluxes at time level n, S is the surface area of the cell face indicated by its indices, and $V_{i,j}$ is the cell volume.

The LU-SGS method of Yoon and Jameson[2] is a logical choice for implicit time advancement, because it makes some simplifications to the implicit equation that diagonalize the problem and allow steady-state solutions to be obtained with a dramatically reduced number of iterations over an explicit method, without a substantial increase in the computational cost per iteration. In addition, the extension to three-dimensional flows is straightforward, unlike many implicit methods. Although the method as formulated does not lend itself to implementation on a parallel machine, we will see that it is possible to make some modifications to the algorithm that make it inherently data parallel. Following the method of Yoon and Jameson,[2] we approximate the implicit Jacobians

$$A_+ = \tfrac{1}{2}(A + \rho_A I), \qquad A_- = \tfrac{1}{2}(A - \rho_A I)$$

where ρ_A is the spectral radius of the Jacobian A, given by the magnitude of the largest eigenvalue $|u| + a$, where a is the speed of sound. Then differences between the Jacobians become, for example,

$$A_+ - A_- = \rho_A I$$

If we move the off-diagonal terms in Eq. (1) to the right-hand side, the resulting implicit equation becomes

$$\begin{aligned}
&(I + \lambda_A I + \lambda_B I)^n_{i,j}\delta U^n_{i,j} = \Delta t R^n_{i,j} \\
&+ \frac{\Delta t}{V_{i,j}}A^n_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^n_{i-1,j} - \frac{\Delta t}{V_{i,j}}A^n_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^n_{i+1,j} \\
&+ \frac{\Delta t}{V_{i,j}}B^n_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^n_{i,j-1} - \frac{\Delta t}{V_{i,j}}B^n_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^n_{i,j+1}
\end{aligned} \tag{2}$$

where $\lambda_A = (\Delta t S/V)\rho_A$. With this approximation, Eq. (2) can be solved without any costly matrix inversions. Note that we are making approximations only to the implicit side of the problem; therefore the steady-state solution will be unaffected.

The LU-SGS algorithm employs a series of corner-to-corner sweeps through the flowfield using the latest available data for the off-diagonal terms to solve Eq. (2). This method is efficient on a serial or vector machine. However, significant modifications are required to reduce or eliminate the data dependencies that are inherent in Eq. (2) and to make the method parallelize effectively. The DP-LUR approach solves this problem by replacing the Gauss–Seidel sweeps with a series of pointwise relaxation steps using the following scheme. First, the off-diagonal terms are neglected and the right-hand side, $R_{i,j}$, is divided by the diagonal operator to obtain $\delta U^{(0)}$:

$$\delta U^{(0)}_{i,j} = \big(I + \lambda^n_A I + \lambda^n_B I\big)^{-1}_{i,j}\Delta t R^n_{i,j}$$

Then a series of $k_{\max}$ relaxation steps are made with $k = 1, \ldots, k_{\max}$:

$$\begin{aligned}
&\delta U^{(k)}_{i,j} = \big(I + \lambda^n_B I + \lambda^n_B I\big)^{-1}_{i,j}\Big\{\Delta t R^n_{i,j} \\
&+ \frac{\Delta t}{V_{i,j}}\Big[A^n_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^{(k-1)}_{i-1,j} - A^n_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^{(k-1)}_{i+1,j} \\
&+ B^n_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^{(k-1)}_{i,j-1} - B^n_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^{(k-1)}_{i,j+1}\Big]\Big\}
\end{aligned} \tag{3}$$

then

$$\delta U^{n+1}_{i,j} = \delta U^{(k_{\max})}_{i,j}$$

With this approach, all data required for each relaxation step have already been computed during the previous step. Therefore, the entire relaxation step is performed simultaneously in parallel without any data dependencies, and all communication can be handled by optimized nearest-neighbor routines. In addition, since the same pointwise calculation is performed on each computational cell, load balancing will be ensured as long as the data are evenly distributed across the available processors. The resulting algorithm was shown to be efficient for the simulation of perfect gas and reacting inviscid flowfields, and in fact with $k_{\max} = 4$ it converges to a steady-state solution in fewer iterations than the original LU-SGS method.[3] The low memory usage of the method makes it possible to solve up to 32 million grid points on a 512-processor CM-5 with 16 gigabytes (GB) of available memory. This fact, together with the nearly perfect scalability of the algorithm to larger numbers of processing nodes, makes it attractive for the solution of very large problems.

For the solution of viscous flows, Eq. (2) or (3) must be modified to include the contribution of the appropriate viscous terms. Following the method of Tysinger and Caughey,[7] we can linearize the viscous flux vectors F_v and G_v, assuming that the transport coefficients are locally constant, to obtain

$$F^{n+1}_v \simeq F^n_v + \frac{\partial}{\partial\xi}(L\delta U)^n, \qquad G^{n+1}_v \simeq G^n_v + \frac{\partial}{\partial\eta}(N\delta U)^n$$

where the viscous Jacobians L and N are evaluated in such a way that they are functions of the vector of conserved quantities U and not the derivatives of U. An alternate derivation by Gnoffo[8] results in a slightly different form for the Jacobian matrices, but the two approaches give nearly equivalent performance when implemented in the DP-LUR method. To maintain the diagonal nature of the method, an approximation must be made to the viscous Jacobians that appear on the diagonal. One approach is to use the spectral radii of L and N, given by

$$\rho_L = \rho_N = \frac{\gamma\mu}{\rho Pr}\left(\frac{S}{J}\right)$$

where Pr is the Prandtl number and J is the determinant of the transformation matrix between Cartesian and curvilinear coordinates.

The resulting algorithm is then

$$\begin{aligned}
&(I + \lambda_A I + 2\lambda_L I + \lambda_B I + 2\lambda_N I)^n_{i,j}\delta U^n_{i,j} = \Delta t R^n_{i,j} \\
&+ \frac{\Delta t}{V_{i,j}}(A_+ - L)^n_{i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^n_{i-1,j} \\
&- \frac{\Delta t}{V_{i,j}}(A_- + L)^n_{i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^n_{i+1,j} \\
&+ \frac{\Delta t}{V_{i,j}}(B_+ - N)^n_{i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^n_{i,j-1} \\
&- \frac{\Delta t}{V_{i,j}}(B_- + N)^n_{i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^n_{i,j+1}
\end{aligned} \tag{4}$$

where $\lambda_L = (\Delta t S/V)\rho_L$. Note that if the computational grid is constructed so that η is the body-normal direction, the ξ-direction viscous Jacobians typically will be much smaller than those in the η direction and can be eliminated from Eq. (4) with no effect on the performance of the algorithm. This is equivalent to making a thin-layer approximation on the implicit side only and therefore will

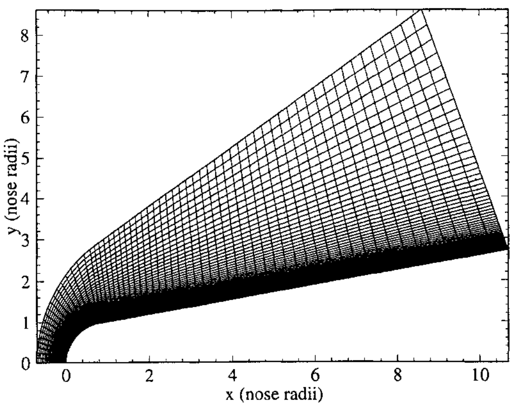


Fig. 1 Sample 128 × 128 cylinder–wedge grid. Every other grid point is shown.

not affect the steady-state solution. The approach of Tysinger and Caughey, which has been used successfully by several authors,[4,5] in conjunction with the LU-SGS method, can be readily adapted to the DP-LUR algorithm by replacing the diagonal operator and off-diagonal Jacobian matrices in Eq. (3) with their counterparts in Eq. (4). The diagonal viscous DP-LUR algorithm thus retains all of the benefits of the inviscid implementation and requires no additional interprocessor communication.

The viscous implementation of the diagonal DP-LUR method has been tested on two- and three-dimensional geometries, with the emphasis on evaluating the convergence properties and parallel performance of the new method. The primary test case for this paper is the Mach 15 perfect gas flow over a cylinder–wedge body, with Reynolds numbers based on freestream conditions and the body length varying from 3×10^3 to 3×10^7. A sample 128 × 128 grid for this problem is shown in Fig. 1. The three-dimensional computations were performed on multiple planes of the same two-dimensional grids, which makes it easy to directly compare the convergence properties of the two- and three-dimensional implementations of the method. However, the method has been used with good results on several true three-dimensional flows.[9] The boundary-layer resolution is measured with the wall variable

$$y_+ = \frac{\rho y u_*}{\mu}$$

where u_* is the friction velocity, given in terms of the wall stress τ_w by $u_* = \sqrt{(\tau_w/\rho)}$. For a well-resolved boundary layer, the mesh spacing is typically chosen so that $y_+ \le 1$ for the first cell above the body surface. The grid for each case is then exponentially stretched from the wall to the outer boundary, which results in approximately 70 points in the boundary layer at all Reynolds numbers for the baseline 128 × 128 grids.

Although the results presented here are based on modified first-order Steger–Warming flux vector splitting for the explicit fluxes,[10] the derivation of the implicit algorithm is general and can be used with many explicit methods. Results have also been obtained using a Harten–Yee upwind non-MUSCL total variation diminishing scheme,[11] with comparable convergence properties. All implicit cases were run with essentially infinite time step [Courant–Friedrichs–Lewy (CFL) number $\approx 10^4$]. However, note that, because of the approximate nature of the implicit operator, the CFL number does not represent the true elapsed time of each iteration. Thus, little improvement in convergence is obtained by taking still larger time steps.

The implicit off-diagonal viscous coupling terms in Eq. (4) facilitate the diffusion of viscous effects through the flowfield, and therefore proper treatment of these terms can be very important to the convergence and stability of the algorithm. There are two possible ways to handle these off-diagonal terms; the full L and N matrices can be used, as shown earlier, or all viscous Jacobians can be approximated by their spectral radii, as in

$$L \simeq \rho_L I, \qquad N \simeq \rho_N I$$

Figure 2 plots the convergence histories for these methods for an $Re = 3 \times 10^4$ flow. As expected, the coupling terms greatly improve

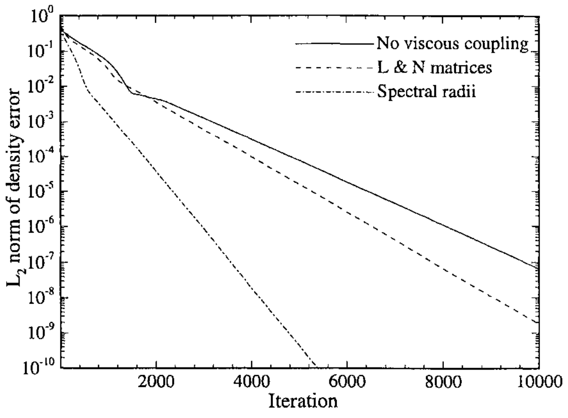


Fig. 2 Convergence histories for the diagonal DP-LUR method showing effect of off-diagonal viscous coupling. Cylinder–wedge body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$ for the first cell above the body.

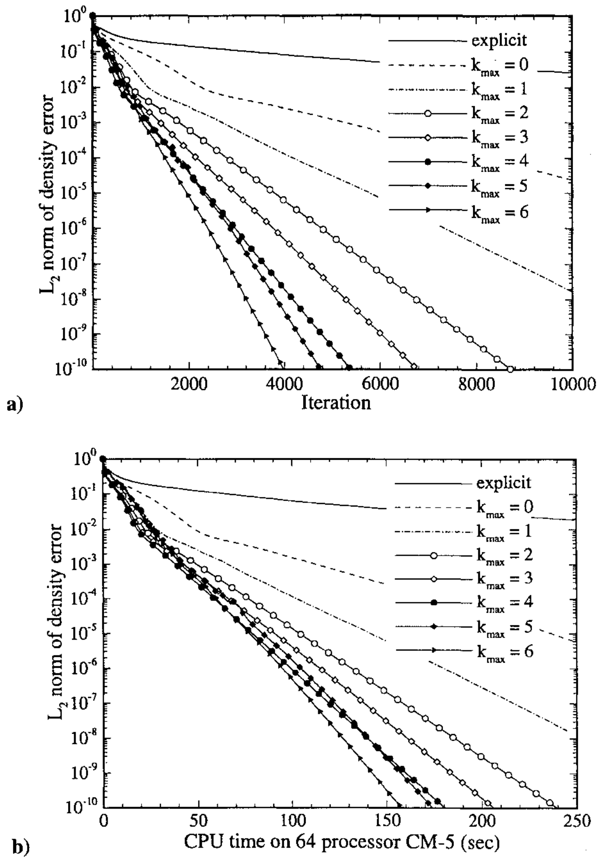


Fig. 3 a) Convergence histories and b) CPU times on 64-processor CM-5 for the diagonal DP-LUR method showing influence of $k_{\max}$. Cylinder–wedge body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$.

the convergence rate of the algorithm and are necessary for stability at some flow conditions. For all cases run to date the spectral radius approximation is the most efficient approach; therefore it will be used for all of the diagonal DP-LUR results presented here.

The effect of the number of relaxation steps ($k_{\max}$) on convergence is shown in Fig. 3a for the two-dimensional cylinder wedge at $Re = 3 \times 10^4$ and $y_+ = 1$. The diagonal viscous DP-LUR method exhibits convergence characteristics similar to the inviscid algorithm, with convergence steadily improving as $k_{\max}$ increases. Figure 3b plots the convergence histories for the same case vs computer time on a 64-processor CM-5. Because each relaxation step has a small cost compared with the evaluation of the explicit fluxes, we see that increasing $k_{\max}$ also improves the cost effectiveness of the method. However, note that this improvement diminishes rapidly for high

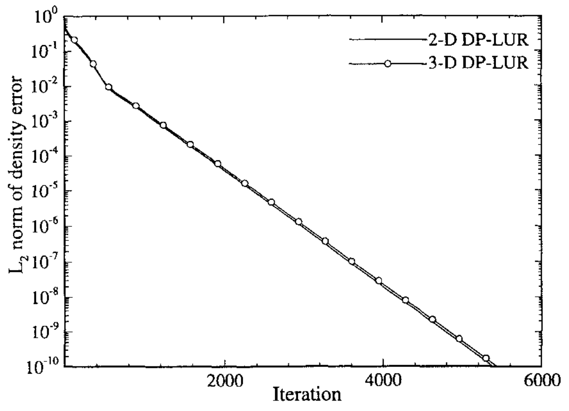


Fig. 4 Convergence histories for the two- and three-dimensional versions of the diagonal DP-LUR method. Cylinder–wedge body at M_∞ = 15 and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$.

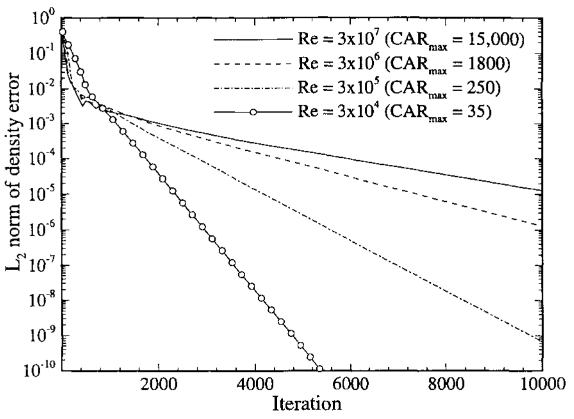


Fig. 5 Convergence histories for the diagonal DP-LUR method showing influence of Reynolds number. Cylinder–wedge body at M_∞ = 15; 128 × 128 grid with $y_+ = 1$.

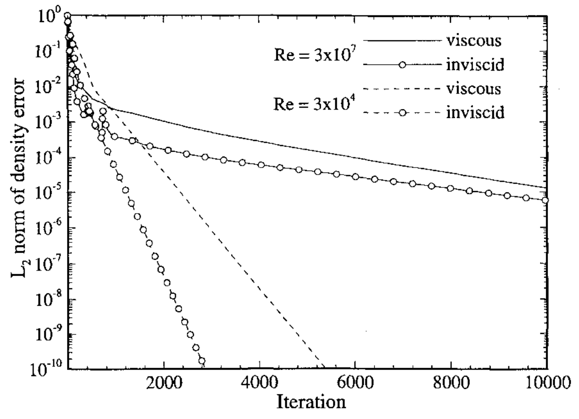


Fig. 6 Convergence histories for the viscous and inviscid implementations of the diagonal DP-LUR method. Cylinder–wedge body at M_∞ = 15; 128 × 128 grid with $y_+ = 1$ for each case.

values of $k_{\max}$. For the case shown in Fig. 3, the improvement in cost effectiveness between $k_{\max} = 2$ and 4 is about 25%, whereas the improvement from $k_{\max} = 4$ to 6 is about 12%, and the improvement from $k_{\max} = 6$ to 8 (not shown) is only 6%. It is theoretically possible to use any number of relaxation steps; however, our experience has shown that using values of $k_{\max}$ greater than 4 can cause the algorithm to become less stable for more complex problems, which reduces the maximum allowed CFL number and slows the overall convergence rate. Therefore, all of the results presented next were run at $k_{\max} = 4$, which has proven to give the best combination of numerical stability and cost effectiveness for a variety of problems.

Figure 4 plots the convergence rate of the two- and three-dimensional diagonal DP-LUR methods for the same case shown in Fig. 3. We see that there is essentially no difference in the performance of the two implementations. This result holds for all cases tested in this paper.

Figure 5 shows the convergence histories of the algorithm for several viscous flows with Reynolds numbers from 3×10^4 to 3×10^7. The 128 × 128 grid for each case was chosen so that $y_+ = 1$ for the first cell above the body, ensuring that the boundary layer is equally well resolved for all cases. Since the thickness of the boundary layer decreases with increasing Reynolds number, the maximum cell aspect ratio (CAR) of the grid must increase as well to meet the $y_+ = 1$ requirement. For the test cases in Fig. 5, the maximum CAR ranges from 35 for $Re = 3 \times 10^4$ to about 15,000 for $Re = 3 \times 10^7$. We see that, although the algorithm converges very well for low Reynolds number (low CAR) flows, the convergence rate decreases rapidly as the Reynolds number and, therefore, CAR increase. This is not surprising, since it is well known that methods of this type show significant degradation of convergence rate on high CAR grids.[4–6] The underlying reason for this can be seen easily by examining the diagonal term in Eq. (4). Assuming a rectangular grid for simplicity, with x tangential and y normal to the body, the scalar terms on the diagonal are inversely proportional to the mesh spacing, as in

$$\lambda_A = \frac{\Delta t\, S}{V}\rho_A \simeq \frac{\Delta t}{\Delta x}\rho_A$$

Therefore, the diagonal term in Eq. (4) becomes

$$\{I + \lambda_A I + 2\lambda_L I + \lambda_B I + 2\lambda_N I\}$$

$$\simeq \left[I + \frac{\Delta t}{\Delta x}(\rho_A + 2\rho_L)I + \frac{\Delta t}{\Delta y}(\rho_B + 2\rho_N)I\right]$$

From this it is apparent that as the CAR in the boundary layer increases ($\Delta x \gg \Delta y$), the magnitude of the diagonal operator will be increasingly dominated by the terms inversely proportional to Δy. This will eventually cause the solution to become overstabilized in the x direction, slowing the convergence rate. This result also applies to three-dimensional flows, except that there are now three relevant cell lengths, one normal and two tangential to the body. In all cases the level of overstabilization and, therefore, the degradation of the convergence rate are driven by the maximum CAR, which for most external flows will be given approximately by the ratio of the mesh spacing in one of the body-tangential directions to the spacing in the body-normal direction. In addition, examination of the flowfield indicates that for the high CAR cases the cells nearest to the body evolve slowly compared with the rest of the flow, which means that, even after the global error norm has fallen several orders of magnitude, the solution may still be changing near the body surface.

Note that this problem is solely a function of the computational grid and consequently should be observed for inviscid flows as well. This is demonstrated in Fig. 6, which compares the convergence of the viscous and inviscid algorithms for two of the cases in Fig. 5. As expected, the performance of the algorithm on high CAR grids is nearly independent of viscosity. There is a slight degradation in convergence rate for the viscous problem, however, which becomes more pronounced as the Reynolds number decreases. This effect, which can also be seen in Fig. 6, is caused by the slower time scales involved in the evolution of the viscous portions of the flow.

There have been many methods proposed to resolve this high CAR convergence problem, including local time stepping, modified Jacobian splitting,[12] nonisotropic diagonal terms,[13,14] and multigrid methods,[6] but none of these modifications fully addresses the fundamental problem with the diagonalized operator.

Full Matrix DP-LUR

The obvious way to avoid poor performance on high aspect ratio grids is to remove all of the simplifications discussed earlier and move some or all of the off-diagonal coupling terms back to the left-hand side of Eq. (2). However, this would require the inversion of a large block matrix at every time step, which would be both computationally and memory intensive and would destroy the data-parallel nature of the current algorithm. Another possibility arises if we recognize the fact that the diagonalization given by Yoon and Jameson[2] essentially acts to ensure numerical stability by increasing the diagonal dominance of the system. As stated earlier, this approximation works well on low CAR grids, but as the CAR increases, the diagonal dominance increases as well, resulting in an overstabilized algorithm that converges very slowly. A better approach for this problem would be to remove the approximation altogether and to use the exact flux Jacobians on and off the diagonal for both the Euler and viscous terms. This results in

$$\left\{I + \frac{\Delta t}{V_{i,j}}\left[(A_+ - L)_{i+\frac{1}{2},j}S_{i+\frac{1}{2},j} - (A_- + L)_{i-\frac{1}{2},j}S_{i-\frac{1}{2},j} + (B_+ - N)_{i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}} - (B_- + N)_{i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\right]\right\}^n \delta \boldsymbol{U}^n_{i,j} = \Delta t \boldsymbol{R}^n_{i,j}$$
$$+ \frac{\Delta t}{V_{i,j}}(A_+ - L)^n_{i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta \boldsymbol{U}^n_{i-1,j} - \frac{\Delta t}{V_{i,j}}(A_- + L)^n_{i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta \boldsymbol{U}^n_{i+1,j}$$
$$+ \frac{\Delta t}{V_{i,j}}(B_+ - N)^n_{i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta \boldsymbol{U}^n_{i,j-1} - \frac{\Delta t}{V_{i,j}}(B_- + N)^n_{i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta \boldsymbol{U}^n_{i,j+1} \tag{5}$$

The exact Euler Jacobians are given by

$$A_\pm = C_A^{-1}\Lambda_{A\pm}C_A$$

where C_A^{-1} and C_A are the left- and right-hand eigenvector matrices and $\Lambda_{A\pm}$ is the diagonal matrix of positive or negative eigenvalues. The full matrix form of the DP-LUR method then follows logically if we use the same approach to solve Eq. (5) as shown in Eq. (3). The resulting algorithm would still require a large amount of communication or storage to implement in data parallel, since all of the Jacobians should be evaluated at the indicated cell face; however, the method can be further modified by calculating all Jacobians at the local cell center with almost no degradation of convergence rate. In addition, if we assume that adjacent cell face areas on the diagonal are approximately equal,

$$S_{i+\frac{1}{2},j} \simeq S_{i-\frac{1}{2},j} \simeq S_I, \qquad S_{i,j+\frac{1}{2}} \simeq S_{i,j-\frac{1}{2}} \simeq S_J$$

and recognize that

$$\Lambda_+ - \Lambda_- = |\Lambda|$$

the method simplifies further. The full matrix DP-LUR method is then given by

$$\delta \boldsymbol{U}^{(0)}_{i,j} = \left[I + \frac{\Delta t}{V_{i,j}}\left(C_A^{-1}|\Lambda_A|C_A S_I - 2LS_I + C_B^{-1}|\Lambda_B|C_B S_J - 2NS_J\right)\right]^{-1}_{i,j} \Delta t \boldsymbol{R}^n_{i,j}$$

and the series of $k_{\max}$ relaxation steps with $k = 1, \ldots, k_{\max}$:

$$\delta \boldsymbol{U}^{(k)}_{i,j} = \left[I + \frac{\Delta t}{V_{i,j}}\left(C_A^{-1}|\Lambda_A|C_A S_I - 2LS_I + C_B^{-1}|\Lambda_B|C_B S_J - 2NS_J\right)\right]^{-1}_{i,j} \left[\Delta t \boldsymbol{R}^n_{i,j} + \frac{\Delta t}{V_{i,j}}(A_+ - L)^n_{i-1,j}S_{i-\frac{1}{2},j}\delta \boldsymbol{U}^{(k-1)}_{i-1,j} - \frac{\Delta t}{V_{i,j}}(A_- + L)^n_{i+1,j}S_{i+\frac{1}{2},j}\delta \boldsymbol{U}^{(k-1)}_{i+,1,j} + \frac{\Delta t}{V_{i,j}}(B_+ - N)^n_{i,j-1}S_{i,j-\frac{1}{2}}\delta \boldsymbol{U}^{(k-1)}_{i,j-1} - \frac{\Delta t}{V_{i,j}}(B_- + N)^n_{i,j+1}S_{i,j+\frac{1}{2}}\delta \boldsymbol{U}^{(k-1)}_{i,j+1}\right] \tag{6}$$

then

$$\delta \boldsymbol{U}^{n+1}_{i,j} = \delta \boldsymbol{U}^{(k_{\max})}_{i,j}$$

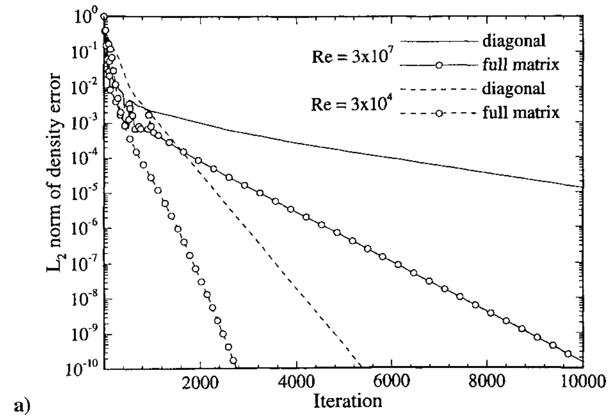
The full matrix method is somewhat more computationally and memory intensive than the diagonal version, since an $n_{\text{eq}} \times n_{\text{eq}}$ matrix must be stored and inverted at every grid point, where n_{eq} is the number of conservation equations in the system. However, that calculation requires only local data and must be performed just once per time step. With the preceding simplifications, the algorithm remains inherently data parallel and free of all data dependencies. Also, it requires no more communication per iteration than the diagonal version.

One potential problem with this method is that by removing the Yoon and Jameson approximation[2] we are no longer assured of diagonal dominance for all situations and therefore would no longer expect the algorithm to be unconditionally stable in the linear limit. However, experience has shown that the method is still numerically robust for both high and low CAR grids, as will be seen later. In addition, since Eq. (6) is a better physical representation of the problem than Eq. (4), the full matrix method should converge much better than the diagonal method on complex flows. One result of using this more physical representation is that the numerical stability and convergence rate are sensitive to the magnitude of the implicit time step, rather than the CFL number. In fact, for all of the cylinder–wedge flows in this paper, the maximum stable time step was governed solely by the freestream conditions, independent of the mesh spacing (and therefore independent of the CFL number).

The performance of the full matrix method is compared with the diagonal method in Figs. 7a and 7b. The computational grids are identical to those used in Fig. 6. All full matrix results were run at $k_{\max} = 4$, which remains the most stable implementation of the method. Results are shown here only for the two-dimensional implementation; however, the three-dimensional full matrix algorithm exhibits nearly identical convergence properties. In Fig. 7a, which plots convergence vs number of iterations for two different Reynolds numbers, it can be seen that the full matrix method requires significantly fewer iterations to reach steady state. The same results are plotted against computer time on a 64-processor CM-5 in Fig. 7b. Although each iteration of the more complex full matrix method takes about 1.7 times as long as the diagonal method for both the two- and three-dimensional implementations, it is clearly the most cost-effective approach, especially for high Reynolds number flows. For the cases shown in Fig. 7, the full matrix method converges about 1.3 times faster for the $Re = 3 \times 10^4$ flow and 2 times faster for the $Re = 3 \times 10^7$ flow.

Most grids over complex two- and three-dimensional geometries have regions of very high aspect ratio cells, even for low Reynolds number or inviscid flow simulations. Therefore, it is useful to perform a parametric study of the effects of CAR on the convergence of the two methods. Figure 8a presents the computer time on a 64-processor CM-5 required for the error norm to fall 10 orders of magnitude for the two-dimensional inviscid algorithms as a function of the maximum CAR. We see that the full matrix method is slightly faster than the diagonal method even for the lowest CAR grid tested (CAR = 10). This is an interesting result, because the full matrix method was developed specifically to improve high aspect ratio performance. As the CAR increases, both methods require more iterations to reach steady state, but the full matrix method is affected by the grid to a lesser extent and converges more than two times faster on the highest CAR grid tested (CAR = 10,000).

In Fig. 8b we also plot computer time for 10 orders of error norm reduction vs CAR, but this time for a low Reynolds number



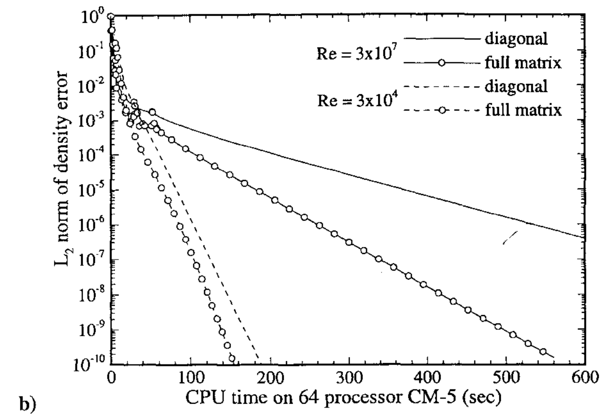
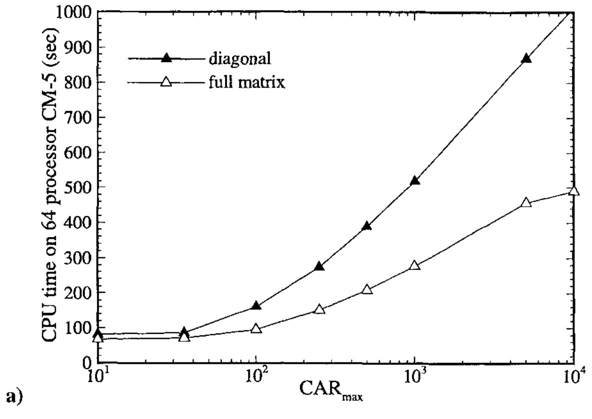


Fig. 7 a) Convergence histories and b) CPU times on 64-processor CM-5 for the full matrix and diagonal DP-LUR methods. Cylinder–wedge body at M_∞ = 15; 128 × 128 grid with y_+ = 1 for each case.



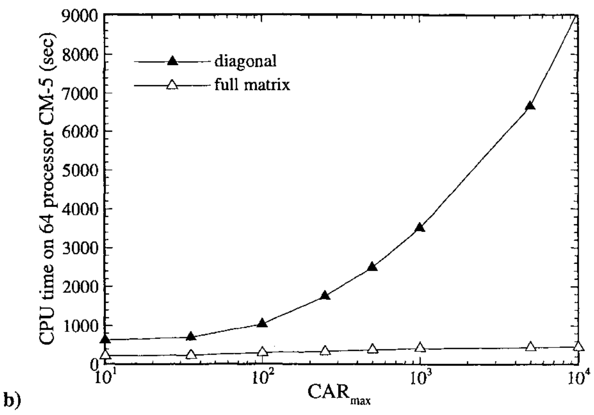


Fig. 8 CPU times on 64 processor CM-5 to achieve 10 orders of error norm convergence for the two-dimensional full matrix and diagonal DP-LUR methods as a function of maximum CAR: a) inviscid and b) viscous with $Re = 3 \times 10^3$. Cylinder–wedge body at M_∞ = 15; 128 × 128 grid with y_+ = 1 for each case.

($Re = 3 \times 10^3$) flow. For this case the full matrix method is superior on all grids and, in fact, is about 20 times faster on the highest aspect ratio grid tested. It is also interesting to directly measure the effect that increasing the CAR has on each of these methods. Dividing the computer time required for the highest CAR grid (10,000) by the time for the lowest aspect ratio grid (10), we see that, whereas the diagonal code takes 14 times longer for the high CAR case, the time required for the full matrix method only doubles. This shows that the more physical approach of the full matrix method is also better suited to the solution of highly viscous (low Reynolds number) flows.

The results presented in Fig. 8 show that the use of the full matrix method reduces, but does not eliminate, the effect of CAR on convergence rate. The remaining dependence is probably a result of the off-diagonal terms being on the right-hand side of Eq. (6), which means that their effect is only indirectly coupled to the diagonal and is, in fact, lagged by one relaxation step. However, recent work by Buelow et al.[15] suggests that the high CAR convergence of an alternating direction implicit scheme can be improved significantly with the use of viscous preconditioning and the method of characteristics boundary conditions. These methods have not been implemented in this work, but it is possible that they may further improve the convergence of the DP-LUR method.

Another important result from Fig. 8 is that, for all cases run to date, the full matrix method achieves convergence faster than the diagonal method. However, this does not mean that the diagonal method should be discarded. The diagonal method will remain useful for very large simulations because it does not require the storage of any Jacobian matrices and therefore uses about 30% less memory than the full matrix approach. The simulation of reacting flows is another area in which there will be a definite tradeoff between the methods. The large equation sets encountered in these simulations make the diagonal method very attractive, because the full matrix method requires the formation and inversion of an exact Jacobian, and therefore the time and memory required will scale, at least to some extent, with the square of the number of equations. However, the more exact treatment of the source term Jacobian that is possible with the full matrix approach may greatly improve the stability and convergence of numerically stiff problems.

Parallel Performance

The viscous DP-LUR method was written in CMFortran and implemented on the 512-processor Thinking Machines CM-5 located at the University of Minnesota Army High Performance Computing Research Center. Each processor of this machine has 32 MB of memory and four vector units, each with a peak performance of 32 Mflops; therefore the entire 512-node machine has a theoretical peak performance of 64 Gflops and 16 GB of available memory. Interprocessor communication is very slow compared with the computational speed of the processors, primarily because of the large latency. However, communication routines between nearest neighbors, such as the circular shift, are highly optimized and are usually much faster than general router communication. Thus, it is necessary to minimize the total interprocessor communication in the algorithm and use the nearest neighbor routines whenever possible to obtain good performance.

Although the CM-5 supports both data-parallel and message-passing programming environments, the algorithm was implemented only in data parallel, since as formulated it is inherently data parallel and requires no asynchronous communication or computation. Although it is relatively easy to modify a data-parallel code to run effectively on a message-passing machine, the reverse is not usually possible. Therefore this algorithm is quite general and should run efficiently on a variety of parallel architectures with only minor modifications.

Both the diagonal and full matrix versions of the viscous DP-LUR algorithm retain the perfect scalability and high parallel efficiency that characterized the original method.[3] The per-processor performance of each method increases with the number of points on each processing node, as shown in Fig. 9, and is independent of the total number of processors. The primary reason for the increase in performance for larger numbers of grid points per processor is the

Table 1 Sustained floating point performance in Gflops for the four implementations of the viscous DP-LUR code on a 512-processor CM-5

Implementation	Gflop rate
Two-dimensional diagonal	16.5
Three-dimensional diagonal	15.3
Two-dimensional full matrix	19.8
Three-dimensional full matrix	18.9

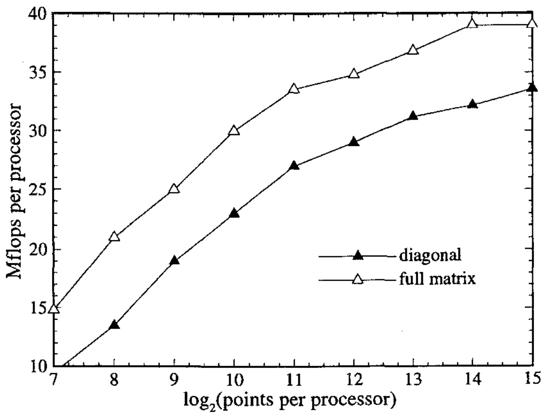


Fig. 9 Per-processor floating point performance of the two-dimensional full matrix and diagonal DP-LUR methods as a function of number of grid points on each processor.

vector nature of the machine; more points on each processor implies a greater vector length. We also see that the full matrix method is consistently faster than the diagonal method when both algorithms are run using the same number of grid points. This result is expected, because the full matrix method is more computationally intensive but requires no additional communication and therefore has a higher parallel efficiency. The peak per-processor performance, obtained when all of the available memory is used, is about 40 Mflops for the two-dimensional full matrix algorithm and 34 Mflops for the diagonal version, as shown in Fig. 9. Performance of the three-dimensional implementations is similar.

Table 1 presents the sustained performance for each implementation of the viscous DP-LUR method, obtained by running the maximum grid size allowed by memory limitations on a given machine. Performance is presented for a 512-processor machine, but the results can be scaled to any number of processors. The fastest algorithm on the CM-5 is the two-dimensional full matrix code, which runs at 19.8 Gflops or about 30% of the theoretical performance. However, even the slowest of the four versions runs at over 20% of peak. The viscous algorithms shown in Table 1 are all about 10% slower than the inviscid versions, as a result of the increased communication involved in the calculation of the flow derivatives required for the explicit viscous fluxes. Finally, we see that the peak performance of the full matrix code is about 20% faster than the diagonal version, as a result of the higher parallel efficiency of the full matrix algorithm described earlier.

Conclusions

The LU-SGS method has been modified to make it amenable to the solution of the Navier-Stokes equations on data-parallel machines. The resulting diagonal DP-LUR method is almost perfectly parallel and displays good stability and convergence properties on many viscous flows. However, the diagonal method converges very slowly on the high aspect ratio grids necessary to adequately resolve the boundary layer of high Reynolds number flows. We show that this poor convergence is caused in part by the diagonal approximation; we therefore relax this assumption and derive a new method. This full matrix DP-LUR method retains the numerical stability of the diagonal method, and its convergence rate is less sensitive to grids with areas of high CAR. Each iteration of the more exact full matrix method takes about 1.7 times as long as the diagonal. However, the full matrix method reaches steady state faster than the diagonal method on all problems tested and is as much as 20 times faster for the simulation of low Reynolds number flows on high aspect ratio grids.

Both methods are implemented in data parallel on the Thinking Machines CM-5, where they show the high parallel efficiency and perfect scalability that characterize the original Euler solver. Their formulation makes it easy to implement the methods in either data-parallel or message-passing modes; therefore they should be readily portable to a variety of different parallel architectures. Floating point performance on the CM-5 is primarily a function of the number of grid points per processor, because of the vector nature of the machine. For very large two-dimensional flow simulations, the 512-processor CM-5 runs at about 16.5 Gflops for the diagonal code and 19.8 Gflops for the full matrix version, which in each case is over 25% of the peak theoretical performance of the machine. Peak performance for three-dimensional flows is slightly lower. In addition, the memory usage of both methods is quite low, making it possible to run up to 32×10^6 grid points for the diagonal method and 24×10^6 grid points for the full matrix method on a 512-processor machine with 16 GB of memory.

In short, the high parallel efficiency, low memory requirements, and numerical stability of the viscous DP-LUR methods make them potentially useful for the solution of very large problems.

Acknowledgments

The authors were supported by NASA Langley Research Center Contract NAG-1-1498. This work is also sponsored in part by the U.S. Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory Cooperative Agreement DAAH04-95-2-0003/Contract DAAH04-95-C-0008.

References

[1]Simon, H. D. (ed.), *Parallel Computational Fluid Dynamics Implementations and Results*, MIT Press, Cambridge, MA, 1992.

[2]Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper 87-0600, Jan. 1987.

[3]Candler, G. V., Wright, M. J., and McDonald, J. D., "Data-Parallel Lower-Upper Relaxation Method for Reacting Flows," *AIAA Journal*, Vol. 32, No. 12, 1994, pp. 2380–2386.

[4]Hassan, B., Candler, G. V., and Olynick, D. R., "The Effect of Thermo-Chemical Nonequilibrium on the Aerodynamics of Aerobraking Vehicles," *Journal of Spacecraft and Rockets*, Vol. 30, No. 6, 1993, pp. 647–655.

[5]Candler, G. V., and Olynick, D. R., "Hypersonic Flow Simulations Using a Diagonal Implicit Method," *Proceedings of the 10th International Conference on Computing Methods in Applied Science and Engineering*, edited by R. Glowinski, Nova Science Publishers, New York, 1992, pp. 29–48.

[6]Yoon, S., and Kwak, D., "Multigrid Convergence of an Implicit Symmetric Relaxation Scheme," *AIAA Journal*, Vol. 32, No. 5, 1994, pp. 950–955.

[7]Tysinger, T., and Caughey, D., "Implicit Multigrid Algorithm for the Navier–Stokes Equations," AIAA Paper 91-0242, Jan. 1991.

[8]Gnoffo, P. A., "An Upwind-Biased, Point Implicit Relaxation Algorithm for Viscous, Compressible Perfect-Gas Flows," NASA TP 2953, Feb. 1990.

[9]Levin, D. A., Collins, R. J., Candler, G. V., Wright, M. J., and Erdman, P. W., "Examination of OH Ultraviolet Radiation from Shock Heated Air," *Journal of Thermophysics and Heat Transfer*, Vol. 10, No. 2, 1996, pp. 200–208.

[10]MacCormack, R. W., and Candler, G. V., "The Solution of the Navier–Stokes Equations Using Gauss–Seidel Line Relaxation," *Computers and Fluids*, Vol. 17, No. 1, 1989, pp. 135–150.

[11]Yee, H. C., "A Class of High-Resolution Explicit and Implicit Shock Capturing Methods," NASA TM 101088, Feb. 1989.

[12]Obayashi, S., and Guruswamy, G. P., "Convergence Acceleration of an Aeroelastic Navier–Stokes Solver," AIAA Paper 94-2268, June 1994.

[13]Swanson, R., and Turkel, E., "A Multistage Time-Stepping Scheme for the Navier–Stokes Equations," AIAA Paper 85-0035, Jan. 1985.

[14]Vatsa, V., and Wedan, B., "Development of a Multigrid Code for 3-D Navier–Stokes Equations and Its Application to a Grid Refinement Study," *Computers and Fluids*, Vol. 18, No. 4, 1990, pp. 391–403.

[15]Buelow, P. E., Venkateswaran, S., and Merkle, C. L., "Effect of Grid Aspect Ratio on Convergence," *AIAA Journal*, Vol. 32, No. 12, 1994, pp. 2401–2408.