AIAA JOURNAL
Vol. 36, No. 9, September 1998

Data-Parallel Line Relaxation Method for the Navier–Stokes Equations

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

The Gauss–Seidel line relaxation method is modified for the simulation of viscous flows on massively parallel computers. The resulting data-parallel line relaxation method is shown to have good convergence properties for a series of test cases. The new method requires significantly more memory than the previously developed data-parallel relaxation methods, but it reaches a steady-state solution in much less time for all cases tested to date. In addition, the data-parallel line relaxation method shows good convergence properties even on the high-cell-aspect-ratio grids required to simulate high-Reynolds-number flows. The new method is implemented using message passing on the Cray T3E, and the parallel performance of the method on this machine is discussed. The data-parallel line relaxation method combines the fast convergence of the Gauss–Seidel line relaxation method with a high parallel efficiency and thus shows promise for large-scale simulation of viscous flows.

Introduction

THE numerical simulation of large complex flowfields is a computationally intensive task. In addition, the large disparity between the different length scales encountered in high-Reynolds-number simulations can result in a stiff equation set that usually requires an implicit method to converge to a steady-state solution in a reasonable time. The cost associated with solving this equation set makes the use of a massively parallel supercomputer attractive because these machines have a very large peak performance. However, most implicit methods are inefficient when implemented on a parallel computer. A true implicit method requires the inversion of a large sparse matrix, which involves a great deal of interprocessor communication. This results in poor computational performance. The traditional solution to this problem has been to choose an effective serial algorithm and implement it in parallel using some sort of domain decomposition. This approach has been used with some success,[1] but many of the most effective serial algorithms contain inherent data dependencies and cannot be implemented effectively in parallel without modification.

Another approach for structured meshes is to design a new implicit algorithm that would take advantage of the structured communication pattern. Such an algorithm would be inherently well suited to a data-parallel environment without explicit domain decomposition. The algorithm would be efficient and easy to implement in either data-parallel or message-passing mode and would be portable to a wide variety of parallel architectures. For example, Candler et al.[2] and Wright et al.[3] have shown that it is possible to make some modifications to the Yoon and Jameson lower-upper symmetric Gauss–Seidel (LU-SGS) method[4] that make it almost perfectly data parallel. 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. The DP-LUR method was shown to be attractive for the solution of a variety of viscous problems. However, the method shows a significant degradation of the convergence rate for high-Reynolds-number flows because of the high-cell-aspect-ratio grids needed to resolve the thin boundary layer. Modifications to the DP-LUR method that help to alleviate this problem were presented in Ref. 3, but it remains an issue for high-Reynolds-number flow simulations.

To fully address this convergence degradation on high-cell-aspect-ratio grids, it is necessary to solve the implicit equation in a more closely coupled manner. This is the approach taken in the Gauss–Seidel line relaxation (GSLR) method of MacCormack,[5] which breaks a two-dimensional problem into a series of block tridiagonal matrix solutions in the body-normal direction. This method would be inefficient on a massively parallel machine because the Gauss–Seidel sweeps would require frequent and irregular interprocessor communication. However, it is possible to modify this method using an approach similar to that previously applied to the LU-SGS method. By replacing the Gauss–Seidel sweeps with a series of line relaxation steps, the algorithm can be parallelized effectively. In addition, the potential for a solution bias resulting from the use of the Gauss–Seidel sweeps, which can cause problems in three-dimensional simulations using the GSLR,[6] is removed. This paper discusses the modifications required to create this data-parallel version of the GSLR algorithm. The resulting data-parallel line relaxation (DPLR) method is then compared with the DP-LUR method and the original GSLR method on a variety of viscous problems. Finally, implementation and performance issues on two different parallel computers, the Cray T3E and the Thinking Machines CM-5, are discussed.

Numerical Method

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

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

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

Received May 5, 1997; presented as Paper 97-2046 at the AIAA 13th Computational Fluid Dynamics Conference, Snowmass Village, CO, June 29–July 2, 1997; revision received May 4, 1998; accepted for publication May 14, 1998.

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

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

‡Graduate Research Assistant, Department of Aerospace Engineering and Mechanics and Army High Performance Computing Research Center; currently Research Scientist, Thermosciences Institute, Mail Stop 230-3, NASA Ames Research Center, Moffett Field, CA 94035. Member AIAA.

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

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

to obtain the standard upwind finite volume representation

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

where $R^n_{i,j}$ is the change in the solution due to 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.

For the solution of the Navier–Stokes equations the appropriate implicit viscous Jacobians must be included in Eq. (1). 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. With these definitions Eq. (1) will be unchanged if we simply replace the Euler Jacobians A and B with $\tilde{A}$ and $\tilde{B}$, where

$$\tilde{A}_+ = A_+ - L, \qquad \tilde{A}_- = A_- + L$$
$$\tilde{B}_+ = B_+ - N, \qquad \tilde{B}_- = B_- + N$$

Equation (1) is the basic starting equation for many implicit schemes, including both the DP-LUR and the DPLR methods. From this point, however, the derivation of these two methods is different. We first briefly review the derivation of the DP-LUR method here so that the similarities between the DP-LUR and the new DPLR method may be noted. The full derivation of the DP-LUR method can be found in Refs. 2 and 3.

DP-LUR Method

The first step in the derivation of the DP-LUR method is to move all of the off-diagonal terms in Eq. (1) to the right-hand side. The method of Yoon and Jameson[4] is then used to approximate the implicit Jacobians with

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

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. With this approximation the differences between the Jacobians on the diagonal become diagonal matrices, and the solution of the resulting equation is greatly simplified.

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 the resulting equation. Although this method has been shown to be efficient on a serial or vector machine, significant modifications are required to reduce or to eliminate the data dependencies 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 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 using, for $k = 1, k_{\max}$,

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

then

$$\delta U^n_{i,j} = \delta U^{(k_{\max})}_{i,j} \tag{2}$$

where $\lambda_A = \rho_A \Delta t S/V$. For the solution of viscous flows, Eq. (2) can be modified to include the contribution of the appropriate implicit viscous Jacobians by using a spectral radius approximation.[3] With this approach, all data that are required for each relaxation step have already been computed during the previous step. Therefore, the entire relaxation step may be performed simultaneously in parallel without data dependencies. In addition, because 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. Thus, the DP-LUR algorithm is almost perfectly data parallel, and aside from the required nearest-neighbor communication, a massively parallel computer should approach its peak operational performance.

A modified version of this method that improves the convergence rate on high-cell-aspect-ratio grids can be derived from Eq. (2) if the Yoon and Jameson[4] approximation is relaxed and the approximate Jacobians are replaced with their exact counterparts. The resulting full matrix DP-LUR method improves performance by eliminating the overstabilization that is characteristic of all diagonalized methods.[8] The full matrix method is slightly more memory and computationally intensive because it requires the storage and inversion of a single Jacobian matrix at each grid point, but it retains the excellent parallel efficiency of the original method.[3]

DPLR Method

Although both variations of the DP-LUR method are efficient for the simulation of many flows, they are both affected to some extent by the high-cell-aspect-ratio grids necessary to resolve the boundary layers of high-Reynolds-number flows. The dependence of the convergence rate on the cell aspect ratio was reduced by the introduction of the full matrix DP-LUR method, as discussed earlier, but some performance degradation is still apparent. The remaining dependence is primarily due to the fact that these methods place all of the off-diagonal terms on the right-hand side, and thus their effect is only weakly coupled to the diagonal. A more accurate approach would be to move all of the off-diagonal terms back to the left-hand side, as in Eq. (1), and to solve the fully coupled problem using a large block-banded matrix inversion. However, this approach would be extremely expensive and inefficient on a parallel machine. A better approach is possible for viscous external flows if we recognize that the viscous flow gradients will be much stronger in the body-normal (η) direction. Thus, the physical problem is much more strongly coupled in the body-normal direction, and it is possible to move just these body-normal terms back to the left-hand side, resulting in the following:

$$\begin{aligned}&\hat{B}_{i,j}\delta U_{i,j+1} + \hat{A}_{i,j}\delta U_{i,j} - \hat{C}_{i,j}\delta U_{i,j-1}\\ &\quad = -\hat{D}_{i,j}\delta U_{i+1,j} + \hat{E}_{i,j}\delta U_{i-1,j} + \Delta t R^n_{i,j}\end{aligned} \tag{3}$$

where the matrices denoted by the carets are defined from the Jacobians as

$$\begin{aligned}\hat{A}_{i,j} = I + (\Delta t/V_{i,j})\big(&\tilde{A}_{+i+\frac{1}{2},j}S_{i+\frac{1}{2},j} - \tilde{A}_{-i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\\ &+ \tilde{B}_{+i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}} - \tilde{B}_{-i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\big)\end{aligned}$$

$$\hat{B}_{i,j} = (\Delta t/V_{i,j})\tilde{B}_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}$$

$$\hat{C}_{i,j} = (\Delta t/V_{i,j})\tilde{B}_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}$$

$$\hat{D}_{i,j} = (\Delta t/V_{i,j})\tilde{A}_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}$$

$$\hat{E}_{i,j} = (\Delta t/V_{i,j})\tilde{A}_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}$$

In this way it is possible to solve Eq. (3) for all j points at each i location as a series of fully coupled block tridiagonal systems aligned in the body-normal direction. This was the approach used by MacCormack in the GSLR method.[5] In this method the problem is solved via a series of alternating forward and backward sweeps through the flowfield in the i direction, using the latest available data for the terms on the right-hand side. This method works well for two-dimensional flows on a serial or vector machine, but it is not straightforward to extend the method to three-dimensional flows. The obvious approach to the three-dimensional case is to continue to set up the problem as a series of block tridiagonal systems normal to the body and to sweep in both the axial and circumferential directions. However, this approach can lead to a nonphysical bias in the converged solution due to the biasing that is inherent in the Gauss–Seidel sweeping process.[6] In addition, the data dependencies in the Gauss–Seidel sweeps would make the algorithm inefficient on a parallel machine. However, it is possible to eliminate both of these difficulties if we modify the GSLR method by replacing the Gauss–Seidel sweeps with a series of line relaxation steps, using a procedure similar to that outlined in the preceding section for the DP-LUR method. The resulting DPLR method is then described by the following scheme. First, the implicit terms on the right-hand side of Eq. (3) are neglected, and the resulting block tridiagonal system is factored and solved for $\delta U^{(0)}$:

$$\hat{B}_{i,j}\delta U^{(0)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(0)}_{i,j} - \hat{C}_{i,j}\delta U^{(0)}_{i,j-1} = \Delta t R^n_{i,j}$$

Then a series of $k_{\max}$ relaxation steps are made using, for $k = 1, k_{\max}$,

$$\hat{B}_{i,j}\delta U^{(k)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(k)}_{i,j} - \hat{C}_{i,j}\delta U^{(k)}_{i,j-1} = -\hat{D}_{i,j}\delta U^{(k-1)}_{i+1,j} + \hat{E}_{i,j}\delta U^{(k-1)}_{i-1,j} + \Delta t R^n_{i,j}$$

then

$$\delta U^n_{i,j} = \delta U^{(k_{\max})}_{i,j} \tag{4}$$

Boundary conditions are implemented by folding the contribution of the boundary cells into the appropriate matrix in a manner identical to that used for GSLR.[5] The DPLR method requires a single lower-upper (LU) factorization and $k_{\max} + 1$ backsubstitutions per iteration. By using this approach, all of the data required for each relaxation step have already been computed during the previous step. Therefore, as long as the data are distributed on the processors in such a way that the body-normal direction is entirely local, the entire relaxation step can be performed simultaneously in parallel with no data dependencies. In addition, because the i-direction, off-diagonal terms are all equally lagged by one relaxation step, the biasing problem no longer exists, and implementation of a three-dimensional version of the algorithm becomes straightforward. The DPLR approach will use significantly more memory than the DP-LUR methods because five Jacobian matrices (seven for three-dimensional flows) must now be computed and stored at each grid point as compared with one for the full matrix DP-LUR method and none for the diagonal method. For perfect gas flows the DPLR method uses about twice as much memory as the DP-LUR method for the two-dimensional implementation and four times as much for the three-dimensional version. However, the DPLR method uses no more memory than the original GSLR method and should converge much faster than either DP-LUR approach due to the more exact formulation of the implicit operator.

Results

The perfect gas implementation of the DPLR method has been tested on two- and three-dimensional geometries with an 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 blunt body, with Reynolds numbers Re based on the freestream conditions and body length varying from 3×10^4 to 3×10^8. 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.

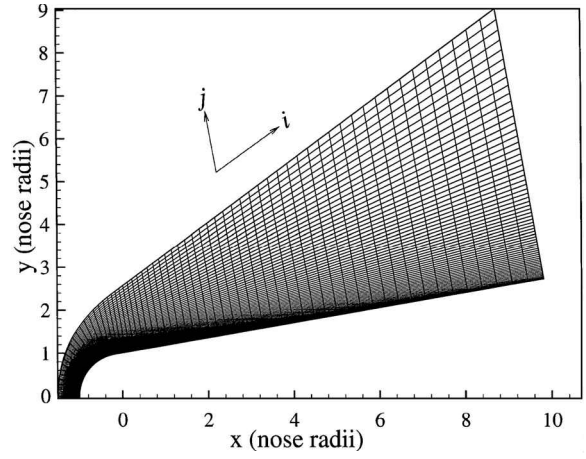


Fig. 1 Sample 128 × 128 cylinder-wedge grid. Every fourth grid point is shown.

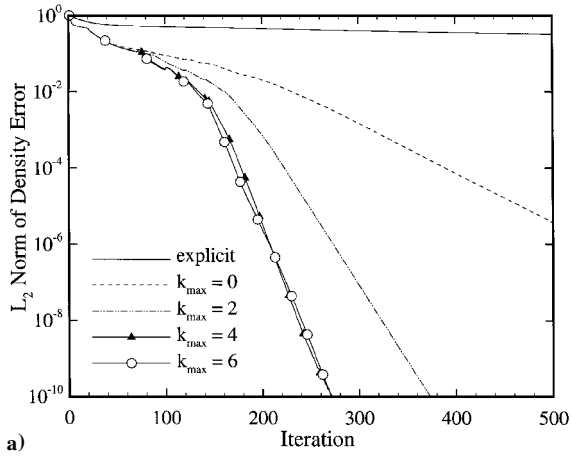
The boundary-layer resolution is measured with the wall variable

$$y_+ = \rho y u_* / \mu$$

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 $y_+ \leq 1$ for the first cell above the body surface. The grid for each case is then exponentially stretched from the wall to the outer boundary.

Although the results presented here are based on modified first-order Steger–Warming flux vector splitting for the numerical flux evaluation,[9] note that the derivation of the implicit algorithm is general and can be used with many flux evaluation methods. In fact, Liou and Van Leer[10] have shown that the use of Steger–Warming splitting can be effective and robust for the implicit part of the problem, even when the fluxes are evaluated by a different method. The DPLR method is in all instances more sensitive to the size of the implicit time step than the DP-LUR method, as expected, because the DPLR method involves a more exact representation of the problem, and therefore the size of the time step will have more physical meaning. In all cases presented here, the implicit time step was chosen to correspond to a Courant–Friedrichs–Lewy (CFL) number of 1 in the first iteration and was rapidly increased to its maximum stable value. The size of the maximum stable time step for each case was governed primarily by the freestream conditions, with little or no limitation due to the mesh spacing. Therefore the maximum CFL number varied considerably from case to case.

Figure 2a shows the effect of the number of relaxation steps ($k_{\max}$) on the convergence rate of the method on the two-dimensional cylinder-wedge geometry at a Reynolds number of 3×10^4 and $y_+ = 1$. In Fig. 2a the explicit solution was obtained using a standard first-order Euler method with the maximum stable time step (CFL = 0.1). The line marked $k_{\max} = 0$ corresponds to performing just the initial block tridiagonal solution along each i line, with no implicit coupling in the i direction. The $k_{\max} = 0$ approach is similar to that proposed by Wang and Widhopf[11] and later implemented in parallel by Taylor and Wang.[12] Although the $k_{\max} = 0$ approach offers a significant improvement over the explicit method, we see from Fig. 2a that the convergence rate of the method improves when the effect of the i-direction coupling terms is included ($k_{\max} > 0$). The DPLR method shows a dependence on $k_{\max}$ similar to that of the DP-LUR methods, with the convergence rate steadily improving as $k_{\max}$ increases, up to $k_{\max} = 6$. Figure 2b shows the effect of the number of relaxation steps on the cost of the method, evaluated as total CPU time on an eight-processor Cray T3E-900. Because the cost of evaluating the Jacobian matrices and setting up the LU factored block tridiagonal system is much greater than the cost of performing additional backsubstitutions, we see that increasing $k_{\max}$ also improves the cost effectiveness of the method, up to $k_{\max} = 4$. As shown in Fig. 2b, values of $k_{\max}$ larger than 4 give little improvement in convergence and are not cost effective. Therefore, all of the results presented in this paper were run at $k_{\max} = 4$.



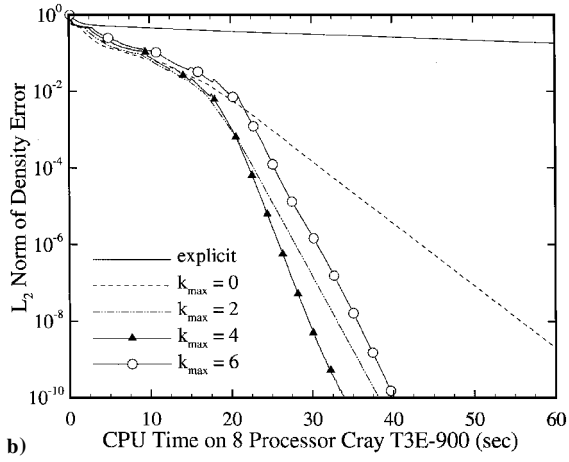


Fig. 2 a) Convergence histories and b) CPU times on an eight-processor Cray T3E-900 for the DPLR method showing influence of k_{max}: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$ for the first cell above the body.

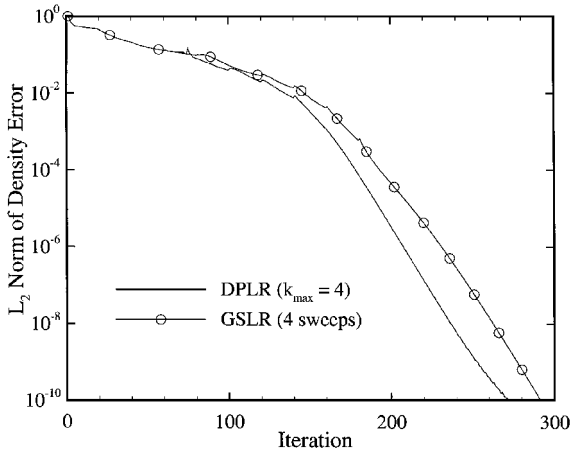


Fig. 3 Convergence histories for the DPLR method as compared with the original GSLR method: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$.

The convergence rate of the new DPLR method is compared with the original GSLR method in Fig. 3. Both methods are tested at the same conditions as in Fig. 2. The methods behave similarly, with the convergence rate increasing significantly after about 150 iterations. The slower convergence rate at the beginning of the solution is due to the motion of the bow shock through the computational grid. Because this is a highly nonlinear process, the shock will move at most one computational cell per iteration. However, once the bow shock has reached its final location, the block tridiagonal solutions rapidly drive the error norm toward machine zero. We see

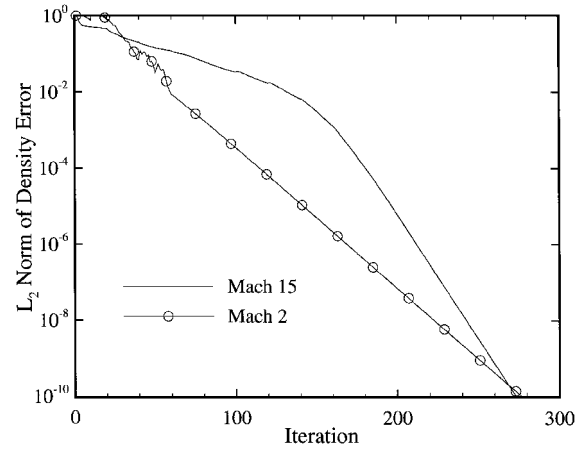


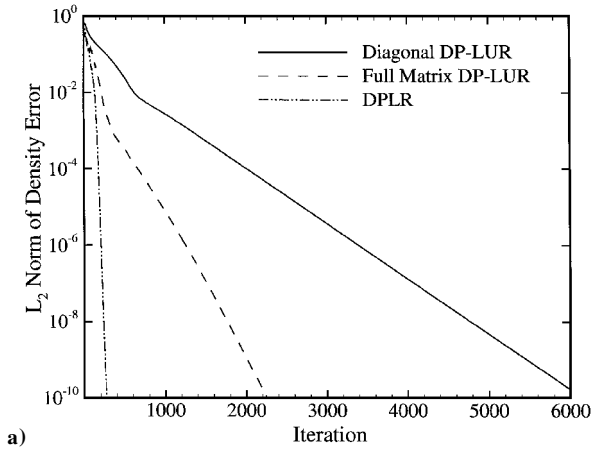
Fig. 4 Convergence histories for the DPLR method on a high-Mach-number and low-supersonic-Mach-number flow: two-dimensional cylinder-wedge blunt body at $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$ for each case.

that, although both methods achieve a 10-order-of-magnitude reduction in the density error norm in fewer than 300 iterations, the DPLR method using $k_{max} = 4$ actually performs a little better than the GSLR method with four sweeps through the flowfield. This is because the DPLR method allows larger time steps to be used for this case. If both methods are run with the same time step, their performance is nearly identical. This is surprising because the GSLR method always uses latest available data and thus should allow information to travel across the entire computational domain during each implicit iteration, whereas with the DPLR method information can travel only k_{max} cells per iteration in the axial (i) direction. However, both methods are identical in their treatment of the body-normal terms.

Figure 4 compares the convergence rate of the DPLR method on two cylinder-wedge flows at Mach 15 and 2. Both cases are run at a Reynolds number of 3×10^4. We see that both cases reach a 10-order-of-magnitude reduction in the density error norm in fewer than 300 iterations. However, there are differences in the convergence histories. The low-Mach-number case requires fewer iterations for the bow shock to reach its final location because the low-Mach-number flow is less nonlinear. In addition, once the shock has reached its final location, the convergence rate of the Mach 2 flow is slower than that for the Mach 15 flow, due to the longer characteristic flow time. Similar results are obtained at other Mach numbers. This shows that the DPLR method can be an effective tool for the solution of both supersonic and hypersonic flows.

The performance of the DPLR method is also examined for three-dimensional flows, using multiple identical planes of the baseline 128×128 cylinder-wedge blunt body grid. In all of the cases, there is essentially no difference in the convergence histories between the two-dimensional and three-dimensional implementations. By performing the three-dimensional calculations on multiple planes of a two-dimensional grid, we can also easily check for any evidence of the solution bias that is exhibited by the three-dimensional GSLR method. This effect is always more noticeable in such cases because any solution bias will tend to create a nonphysical crossflow velocity in the direction in which the multiple planes are projected. This will be evident even before any other differences can be detected between the solutions on different planes. In all of the cases tested to date, the maximum crossflow velocity in the three-dimensional flowfield is more than 10 orders of magnitude smaller than the freestream velocity. We feel that this value is sufficiently small to be attributed to machine roundoff errors, and thus the DPLR method has eliminated the solution bias problem.

The new DPLR method is compared with the diagonal and full matrix DP-LUR methods in Fig. 5 for the cylinder-wedge blunt body at a Reynolds number of 3×10^4 and $y_+ = 1$. All three methods are run at $k_{max} = 4$. We see in Fig. 5a that the DPLR method is very efficient, achieving a 10-order-of-magnitude reduction in the density error norm in fewer than 300 iterations, as compared with



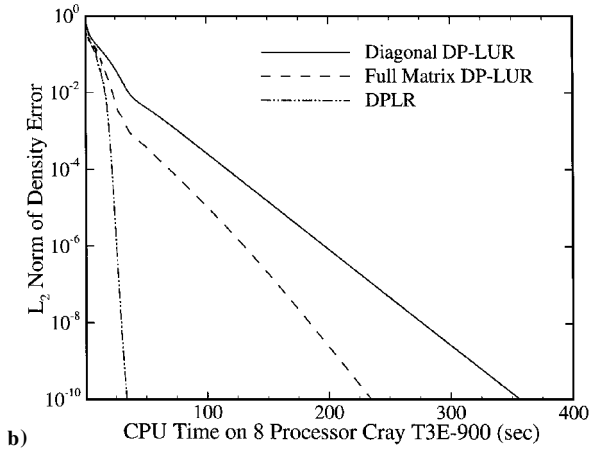


Fig. 5 a) Convergence histories and b) CPU times on an eight-processor Cray T3E-900 for the DPLR method as compared with the two DP-LUR methods: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$.

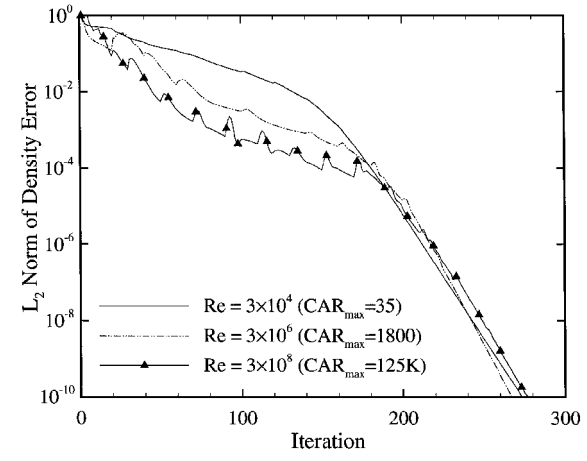


Fig. 6 Convergence histories for the DPLR method showing influence of Reynolds number: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$; 128 × 128 grid with $y_+ = 1$ for each case.

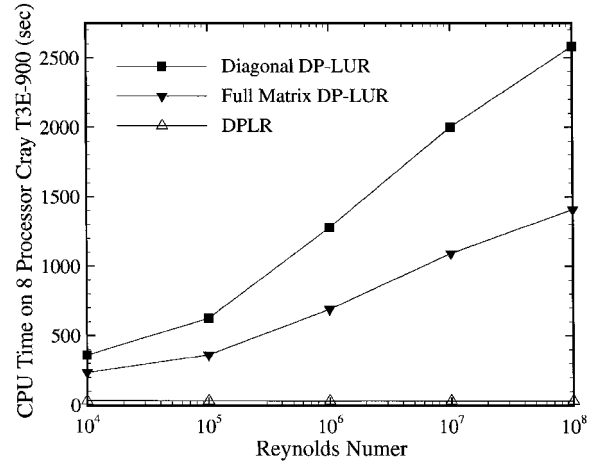


Fig. 7 CPU times on an eight-processor Cray T3E-900 required to achieve 10 orders of density error norm convergence for the DPLR and DP-LUR methods as a function of the Reynolds number: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128 × 128 grid with $y_+ = 1$ for each case.

2300 iterations for the full matrix method and 6200 iterations for the diagonal DP-LUR method. However, convergence in fewer iterations does not necessarily imply that the method will be more cost effective on a massively parallel machine. It is also necessary to know how much each iteration of the method will cost. Therefore, to examine the effectiveness of the new algorithm, Fig. 5b compares the cost of the methods, plotted as CPU time on an eight-processor Cray T3E-900. From Fig. 5b we see that the DPLR method also has a high parallel efficiency and is by far the most cost effective of the three methods. For this problem the line relaxation method reaches a 10-order-of-magnitude reduction in the density error norm in just 35 s, compared with 235 s for the full matrix method and 359 s for the diagonal method. This shows that the DPLR method can potentially be a powerful tool in the simulation of viscous flows.

Figure 6 shows the convergence histories of the DPLR method for three viscous flows with Reynolds numbers ranging from 3×10^4 to 3×10^8. The 128×128 grid for each case was chosen so that $y_+ = 1$ for the first cell above the body, ensuring that the boundary layers for all cases are equally well resolved. Because the boundary-layer thickness 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. 6, the maximum CAR ranges from 35 for $Re = 3 \times 10^4$ to about 125,000 for $Re = 3 \times 10^8$. We can see that, although each of the cases behaves differently during the early phases of the flow evolution, all converge with the same terminal slope after the bow shock reaches its final location. In addition, there is essentially no increase in the number of iterations required to reach steady state as the Reynolds number (and therefore the CAR) is increased.

Figure 7 compares the convergence properties of the DPLR method with the DP-LUR methods on several viscous flows with Reynolds numbers ranging from 3×10^4 to 3×10^8. Once again, the 128×128 grid for each case was chosen so that $y_+ = 1$ for the first cell above the body. As the Reynolds number is increased by four orders of magnitude, the amount of computer time required to achieve a 10-order-of-magnitude reduction in the density error norm remains constant for the DPLR method, whereas the time increases by a factor of 6 for the full matrix method and 7 for the diagonal method. This shows that the more exact implicit operator used in the DPLR method eliminates the convergence degradation on high-cell-aspect-ratio grids.

Figure 8 compares the viscous and inviscid implementations of the DPLR method for two of the cases in Fig. 6. The grid for each case was chosen to satisfy the $y_+ = 1$ condition for the viscous solution. The inviscid solutions were then obtained on the same computational grids. We see that the convergence rates for the viscous and inviscid versions of the method are almost identical. This is in direct contrast to the DP-LUR method, which always requires more iterations to converge the viscous solution. This again shows the benefit of moving the body-normal terms back to the left-hand side of the equation and coupling them directly to the diagonal.

Parallel Performance

The DPLR algorithm is inherently data parallel by design and requires no asynchronous communication or computation. This makes the code readily portable to a variety of parallel architectures with only minor modifications because it is relatively easy to modify a data-parallel code to run effectively on a message-passing machine, whereas the reverse is not necessarily possible. The DPLR method

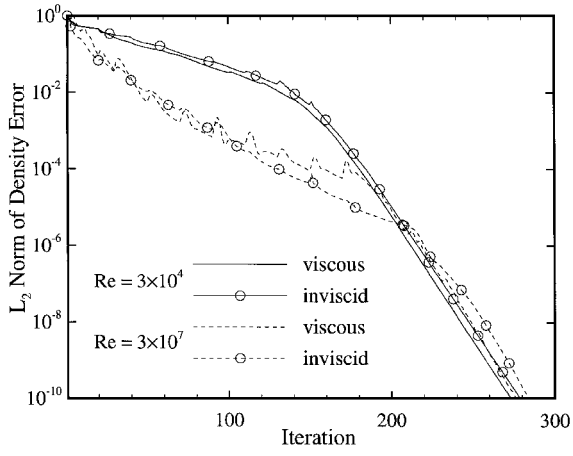


Fig. 8 Convergence histories for the viscous and inviscid implementation of the DPLR method: two-dimensional cylinder-wedge blunt body at $M_\infty = 15$; 128 × 128 grid with $y_+ = 1$ for each case.

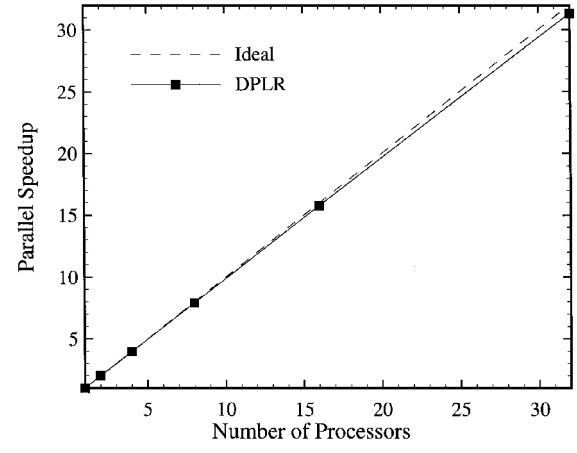


Fig. 9 Parallel efficiency for the two-dimensional viscous DPLR method on the T3E-900: 512 × 512 computational grid used.

was implemented and tested on two different massively parallel architectures. First, a message-passing version using the Message-Passing Interface (MPI) standard for interprocessor communication was implemented on the 272-processor Cray T3E-900 located at Network Computing Services, Inc. (formerly the Minnesota Supercomputer Center). Each processor of this machine has 128 Mbytes of memory and a theoretical peak performance of 900 Mflops. In addition, a data-parallel version was implemented on the 896-processor Thinking Machines CM-5 located at the University of Minnesota Army High Performance Computing Research Center. Each processor of this machine has 32 Mbytes of memory and four vector units, yielding a theoretical peak performance of 128 Mflops per processor.

The data-parallel implementation of the new DPLR method should retain the perfect scalability and high parallel efficiency that characterized the DP-LUR method[2] because the algorithm design is very similar. However, it is difficult to show this on the CM-5 because it is a vector parallel machine, and thus large vector lengths are required to ensure good performance. In addition, it has been shown that only the number of points in the dimensions that are spread across all of the nodes should be used when evaluating the vector length.[2] To solve the block tridiagonal system required by the DPLR method without interprocessor communication, it is necessary that all j points corresponding to a particular i location be entirely on processor, whereas with the DP-LUR method both the i and j directions can be spread across the available processors. Therefore, when the 128 × 128 test case presented in this paper is run on a 64-processor CM-5 with four vector units per processor, the vector length will be 64 for the DP-LUR method but will actually be less than 1 for the DPLR method. This means that the DPLR method is not using the vector hardware on the processors. Therefore we would expect the two-dimensional DPLR method to be very slow on the CM-5 due to the small vector length, even though it may have a high parallel efficiency. This is indeed the case for the implementation tested. For the 128 × 128 test case presented here, the DPLR method runs at only 1.39 Mflops per processor, as compared with 13.2 for the diagonal DP-LUR method and 20.7 for the full matrix method. This would not be a problem on a machine without vector hardware. Thus, although the DPLR method should be efficient on many data-parallel architectures, it is difficult to directly calculate the parallel efficiency of the method on the CM-5.

The performance of the message-passing implementation of the method on the T3E is easier to evaluate because there is no vector hardware on this machine. In this implementation, the data are distributed across the processors by breaking the problem in the i direction. Communication latency is masked by using nonblocking sends and receives in MPI. The parallel speedup curve for the two-dimensional DPLR algorithm on the T3E-900 is presented in Fig. 9. We see that the method has almost perfect speedup, up to the maximum number of processors tested. In fact, on 32 processors the speedup is 31.3, which corresponds to a parallel efficiency of 0.98.

The sustained performance for the two-dimensional and three-dimensional implementations of the DPLR method on the T3E-900 is about 75 Mflops per node, which is only 8.3% of the peak theoretical performance of the machine. This number seems quite low, but it is comparable to other published results. The NAS 2 parallel benchmark results offer the best comparison because these codes have been written to simulate actual computational fluid dynamics applications, and the individual machine vendors have not been allowed to perform assembler-level optimizations to the source code. Unfortunately, NAS 2 benchmark results have not yet been published for the T3E. However, results from the T3D for the block tridiagonal (BT) benchmark show a performance of about 10 Mflops per processor.[13] In addition, results for the NAS 1 benchmarks, which have been published for both the T3D and T3E-600 (600 Mflops peak performance), show that the sustained speed on the T3E-600 is typically about 3.3 times that on the T3D.[14] This would result in a performance of about 33 Mflops per processor on the T3E-600 for the BT benchmark or about 50 Mflops per processor on the T3E-900, assuming perfect scalability. Therefore, the 75 Mflops per processor obtained for the DPLR method seems reasonable. However, it is possible that further optimizations can be made to the source code that would increase the performance.

Conclusions

The GSLR method has been modified to make it amenable to the solution of the Navier–Stokes equations on massively parallel supercomputers. The resulting DPLR method replaces the Gauss–Seidel sweeps of the original with a series of line relaxation steps. In this manner all of the data dependencies in the original method are removed, and each relaxation step becomes almost perfectly data parallel. Because of its design, the new method can be easily implemented in either the data-parallel or message-passing programming styles. The method also retains the good convergence properties of the original GSLR method, and in fact with four relaxation steps it converges in fewer iterations for the test cases considered. In addition, the relaxation steps eliminate the solution bias problem exhibited by the three-dimensional GSLR method, and thus the DPLR method can easily be extended to the solution of three-dimensional flows.

The DPLR method uses considerably more memory than either of the previously developed DP-LUR methods but demonstrates a dramatic improvement in cost effectiveness, reaching steady state in about 15% of the time on a Cray T3E. In addition, both DP-LUR methods showed a degradation of the convergence rate when high-Reynolds-number flows were simulated. However, the DPLR method is more strongly coupled in the body-normal direction and thus has good convergence properties at all Reynolds numbers.

The new method has been implemented using message passing on the Cray T3E-900 and shows nearly perfect speedup, with a parallel efficiency of 98% even when 32 processors are used. The single node performance of the method is about 75 Mflops per processor, which is only 8.3% of the peak theoretical performance of the machine.

However, this value is comparable to data obtained for the NAS 2 parallel benchmarks and does not detract from the high parallel efficiency of the method.

In short, the high parallel efficiency and good convergence characteristics of the DPLR method make it attractive for the solution of very large compressible flow problems.

Acknowledgments

The authors were supported by the NASA Langley Research Center under Contract NAG-1-1498 and the Army Research Office under Grant DAAH04-93-G-0089. This work was also supported 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, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Computer time on the Cray T3E was provided by the Minnesota Supercomputer Institute.

References

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

[2]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.

[3]Wright, M. J., Candler, G. V., and Prampolini, M., "Data-Parallel Lower-Upper Relaxation Method for the Navier–Stokes Equations," *AIAA Journal*, Vol. 34, No. 7, 1996, pp. 1371–1377.

[4]Yoon, S., and Jameson, A., "A Lower-Upper Symmetric Gauss-Seidel Method for the Euler and Navier–Stokes Equations," *AIAA Journal*, Vol. 26, No. 9, 1988, pp. 1025, 1026.

[5]MacCormack, R. W., "Current Status of the Numerical Solutions of the Navier–Stokes Equations," AIAA Paper 85-0032, Jan. 1985.

[6]MacCormack, R. W., "Solution of the Navier–Stokes Equations in Three Dimensions," AIAA Paper 90-1520, June 1990.

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

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

[9]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.

[10]Liou, M. S., and Van Leer, B., "Choice of Implicit and Explicit Operators for the Upwind Differencing Method," AIAA Paper 88-0624, Jan. 1988.

[11]Wang, J. C., and Widhopf, G. F., "An Efficient Finite Volume TVD Scheme for Steady State Solutions of the 3-D Compressible Euler/Navier–Stokes Equations," AIAA Paper 90-1523, June 1990.

[12]Taylor, S., and Wang, J. C., "Launch Vehicle Simulations Using a Concurrent Implicit Navier–Stokes Solver," AIAA Paper 95-0223, Jan. 1995.

[13]Saphir, W., Woo, A., and Yarrow, M., "The NAS Parallel Benchmarks 2.1 Results," Numerical Aerospace Simulation Facility, NAS TR NAS-96-010, NASA Ames Research Center, Moffett Field, CA, Aug. 1996.

[14]Saini, S., and Bailey, D. H., "NAS Parallel Benchmark (Version 1.0) Results 11-96," Numerical Aerospace Simulation Facility, NAS TR NAS-96-018, NASA Ames Research Center, Moffett Field, CA, Nov. 1996.

D. S. McRae
Associate Editor