AIAA JOURNAL
Vol. 32, No. 12, December 1994

Data-Parallel Lower-Upper Relaxation Method for Reacting Flows

Graham V. Candler* and Michael J. Wright†
University of Minnesota, Minneapolis, Minnesota 55455
and
Jeffrey D. McDonald‡
MasPar Computer Corporation, Sunnyvale, California 94086

The implicit lower-upper symmetric Gauss-Seidel (LU-SGS) method of Yoon and Jameson is modified for use on massively parallel computers. The method has been implemented on the Thinking Machines CM-5 and the MasPar MP-1 and MP-2, where large percentages of the theoretical peak floating point performance are obtained. It is shown that the new data-parallel LU relaxation method has better convergence properties than the original method for two different inviscid compressible flow simulations. The convergence is also improved for five-species reacting air computations. The performance of the method on various partitions of the CM-5 and on the MasPar computers is discussed. The new method shows promise for the efficient simulation of very large perfect gas and reacting flows.

Introduction

THE numerical simulation of reacting flow is a computationally intensive task because the fluid dynamical equations must be solved along with the equations that describe the change in the thermochemical state of the gas. For example, the simulation of the flow of reacting air over a hypersonic vehicle may require the solution of 10 conservation equations at every point (five mass equations, three momentum equations, and two energy equations). The simulation of combustion flows requires the solution of an even larger equation set. The large cost of solving the extended equation set limits the size and complexity of the problems that may be simulated.

Most reacting flow simulations use some form of implicit numerical method because the reaction time scales are usually much smaller than the fluid motion time scales. Therefore, if the simulation is to be performed in a reasonable number of iterations, the chemical reaction source terms must be treated implicitly. For steady-state simulations, it is also attractive to use an implicit numerical method for the fluid dynamics.

The lower-upper symmetric Gauss-Seidel (LU-SGS) method of Yoon and Jameson[1] has several features that make it appealing for the simulation of reacting flows. The method approximates the implicit problem so that it can be solved with a pair of sweeps across the grid without any costly matrix inversions. These approximations make the computational cost of the method scale linearly with the number of equations being solved. Also, because no flux Jacobians must be stored, the memory costs are low. The LU-SGS method has been used successfully for the simulation of many reacting flows. The method converges well on fairly uniform grids, but its convergence rate slows for highly stretched grids. Obayashi[2] shows that the LU-SGS method may be used effectively as a preconditioner for a more exact implicit scheme, which would eliminate this problem.

Massively parallel computers have very large peak floating point performance and are potentially useful for the simulation of reacting flows. However, it is difficult to efficiently implement most implicit methods on massively parallel machines. This is because a true implicit method entails the inversion of a very large sparse matrix, requiring much interprocessor communication. Generally, the communication is reduced by decomposing the computational domain into a number of subdomains.[3] Then the implicit problem is solved on each subdomain. This procedure can be quite complex, due to load balancing and boundary condition issues. For this approach to be effective, it is usually implemented on a multiple-instruction, multiple-data (MIMD) machine.

The current paper takes a different approach. The LU-SGS method is modified to make it amenable for use on a single-instruction, multiple-data (SIMD) computer. This data-parallel approach has several important advantages. The method requires only nearest-neighbor communication, so that built-in and optimized interprocessor communication paths may be used. Many massively parallel computers use a grid-based interconnection network, making these nearest-neighbor communications very fast. The need for a complex domain decomposition is eliminated, and the data are distributed automatically by the compiler in a manner which assures optimum load balancing. The method can be implemented easily in a high-level programming language like FORTRAN90. However, the data-parallel approach has one main drawback. To achieve optimal performance, a structured grid must be used to maintain nearest-neighbor communications. This is not a limitation for most reacting flow problems because they typically have fairly simple geometries.

The paper presents the modifications required to make the LU-SGS method parallelize, and then discusses its implementation on two different types of massively parallel computers, the MasPar MP-1 and MP-2 and the Thinking Machines CM-5. The convergence rate of the method on several inviscid test cases is presented, and its performance on these two computers is discussed.

Data-Parallel Lower-Upper Relaxation Method

The LU-SGS method was developed by Yoon and Jameson[1] in 1987 and has been used for reacting flow simulations by several authors.[4–7] The method is attractive because steady-state solutions may be obtained with a dramatically reduced number of iterations over an explicit method, without a substantial increase in computational cost per iteration. The approach is to make an approximation to the implicit equation that diagonalizes the problem. Then the implicit equation can be solved with simple scalar inversions and a series of sweeps through the flowfield. As formulated, the LU-SGS method does not lend itself to optimal implementation on a massively parallel computer. However, as will be seen, it is possible to make some modifications to parallelize the method.

Presented as Paper 94-0410 at the AIAA 32nd Aerospace Sciences Meeting, Reno, NV, Jan. 10–14, 1994; received March 15, 1994; revision received June 1, 1994; accepted for publication June 13, 1994.

*Associate Professor, Department of Aerospace Engineering and Mechanics. Member AIAA.
†Graduate Research Assistant, Department of Aerospace Engineering and Mechanics. Student Member AIAA.
‡Research Staff. Member AIAA.

A modification to the LU-SGS method must be made to simulate reacting flows efficiently. Eberhardt and Imlay[5] diagonalize the Jacobian of the source vector and use reaction rate limiters to achieve good convergence rates for many reacting flows. Following this work, Hassan et al.[7] modify the set of conserved variables and use a similar diagonalization to obtain improved convergence rates. This approach is now briefly discussed.

Consider the Euler equations for a two-dimensional reacting flow. A fully implicit representation of this equation is

$$\frac{\boldsymbol{U}^{n+1}-\boldsymbol{U}^n}{\Delta t}+\frac{\partial \boldsymbol{F}^{n+1}}{\partial x}+\frac{\partial \boldsymbol{G}^{n+1}}{\partial y}=\boldsymbol{W}^{n+1}$$

where $\boldsymbol{U}^n$ is the vector of conserved variables at the nth iteration, $\boldsymbol{F}$ and $\boldsymbol{G}$ are the flux vectors in the x and y directions, and $\boldsymbol{W}$ is the source vector that accounts for the rate of chemical reactions and internal energy relaxation. For a five-species representation of vibrationally relaxing air, $\boldsymbol{U}$ is typically chosen as

$$\boldsymbol{U}=(\rho_{N_2},\rho_{O_2},\rho_{NO},\rho_N,\rho_O,\rho u,\rho v,E_v,E)'$$

where ρ_s is the density of species s, ρu and ρv are the momentum densities, and E_v and E are the vibrational and total energy densities. However, Hassan et al.[7] show that a better choice of $\boldsymbol{U}$ for use with the LU-SGS method is

$$\boldsymbol{U}=(\tilde{\rho}_N,\tilde{\rho}_O,\rho_{NO},\rho_N,\rho_O,\rho u,\rho v,E_v,E)'$$

where $\tilde{\rho}_N$ and $\tilde{\rho}_O$ are the densities of nitrogen and oxygen elements, respectively. In this case, we have

$$\frac{\tilde{\rho}_N}{M_N}=2\frac{\rho_{N_2}}{M_{N_2}}+\frac{\rho_{NO}}{M_{NO}}+\frac{\rho_N}{M_N}$$

$$\frac{\tilde{\rho}_O}{M_O}=2\frac{\rho_{O_2}}{M_{O_2}}+\frac{\rho_{NO}}{M_{NO}}+\frac{\rho_O}{M_O}$$

where M_s is the molecular weight of species s. Because of this change of variable, the source vector $\boldsymbol{W}$ is modified. Since elements are conserved during chemical reactions, $\boldsymbol{W}$ is

$$\boldsymbol{W}=(0,0,w_{NO},w_N,w_O,0,0,w_{E_v},0)'$$

This change of variable does not affect the converged solution in any way, but it improves the convergence of the numerical method.[7]

The modified equation set may be implemented efficiently by computing the fluxes of the standard variables and then making the appropriate linear combination of these fluxes to obtain the elemental density fluxes. Thus, this change of variable has a very low cost, with a large improvement in the convergence rate.

Returning to the derivation of the numerical method, we linearize the flux vectors and the source vector using

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

$$\boldsymbol{G}^{n+1}\simeq\boldsymbol{G}^n+B^n\delta\boldsymbol{U}^n,\qquad \boldsymbol{W}^{n+1}\simeq\boldsymbol{W}^n+C^n\delta\boldsymbol{U}^n$$

Then the fluxes are split using a modified Steger-Warming approach[8,9]

$$\boldsymbol{F}=\boldsymbol{F}_+ +\boldsymbol{F}_-,\qquad A=A_+ +A_-$$

and we obtain the first-order finite volume approximation

$$\begin{aligned}&\delta\boldsymbol{U}^n_{i,j}+\frac{\Delta t}{V_{i,j}}\Big\{\left(A_{+i+\frac12,j}S_{i+\frac12,j}\delta\boldsymbol{U}_{i,j}-A_{+i-\frac12,j}S_{i-\frac12,j}\delta\boldsymbol{U}_{i-1,j}\right)\\&+\left(A_{-i+\frac12,j}S_{i+\frac12,j}\delta\boldsymbol{U}_{i+1,j}-A_{-i-\frac12,j}S_{i-\frac12,j}\delta\boldsymbol{U}_{i,j}\right)\\&+\left(B_{+i,j+\frac12}S_{i,j+\frac12}\delta\boldsymbol{U}_{i,j}-B_{+i,j-\frac12}S_{i,j-\frac12}\delta\boldsymbol{U}_{i,j-1}\right)\\&+\left(B_{-i,j+\frac12}S_{i,j+\frac12}\delta\boldsymbol{U}_{i,j+1}-B_{-i,j-\frac12}S_{i,j-\frac12}\delta\boldsymbol{U}_{i,j}\right)\Big\}^n\\&-\Delta t C^n_{i,j}\delta\boldsymbol{U}^n_{i,j}=\Delta t\boldsymbol{R}^n_{i,j}\end{aligned}\tag{1}$$

where $\boldsymbol{R}^n_{i,j}$ is the change in the solution due to the fluxes and source vector at the current time level n.

The LU-SGS method makes an approximation to the diagonalization of the Jacobians of the forward and backward moving fluxes. Only the Jacobians that appear on the left-hand side of Eq. (1) are approximated, therefore, the converged solution is unaffected by this approximation. The diagonalized Jacobians are

$$A_+=\tfrac12(A+\rho_A I),\qquad A_-=\tfrac12(A-\rho_A I)$$

where ρ_A is the spectral radius of the Jacobian A, which for this case is $|u|+a$, where a is the speed of sound. Then the differences between the Jacobians become, for example,

$$A_+ - A_- = \rho_A I = (|u|+a)I$$

Now, if we move the remaining Jacobians to the right-hand side, we obtain

$$\begin{aligned}&\{I+\lambda_A I+\lambda_B I+\Delta t\,\mathrm{diag}(C)\}^n_{i,j}\,\delta\boldsymbol{U}^n_{i,j}=\Delta t\,\boldsymbol{R}^n_{i,j}\\&+\frac{\Delta t}{V_{i,j}}A^n_{+i-\frac12,j}S_{i-\frac12,j}\delta\boldsymbol{U}^n_{i-1,j}-\frac{\Delta t}{V_{i,j}}A^n_{-i+\frac12,j}S_{i+\frac12,j}\delta\boldsymbol{U}^n_{i+1,j}\\&+\frac{\Delta t}{V_{i,j}}B^n_{+i,j-\frac12}S_{i,j-\frac12}\delta\boldsymbol{U}^n_{i,j-1}-\frac{\Delta t}{V_{i,j}}B^n_{-i,j+\frac12}S_{i,j+\frac12}\delta\boldsymbol{U}^n_{i,j+1}\end{aligned}\tag{2}$$

where $\lambda_A=(\Delta t S/V)\rho_A$. With an appropriate diagonalization of the Jacobian of the source vector with respect to the conserved variables, the solution of Eq. (2) is straightforward. With the change in variable discussed earlier, C may be approximated by

$$\{\mathrm{diag}(C)\}_s=\sqrt{\sum_{r=1}^{ne}\left(\frac{\partial w_s}{\partial\tilde{\rho}_r}\right)^2+\sum_{r=ne+1}^{ns}\left(\frac{\partial w_s}{\partial\rho_r}\right)^2}$$

where ne is the number of elements, and ns is the number of species.

On a scalar or vector computer, Eq. (2) is solved using two sweeps, one from the lower-left-hand corner of the grid to the upper-right-hand corner, and then vice versa. That is, first solve for $\delta\boldsymbol{U}^*$ using

$$\begin{aligned}\delta\boldsymbol{U}^*_{i,j}=&\left[I+\lambda^n_A I+\lambda^n_B I+\Delta t\,\mathrm{diag}(C^n)\right]^{-1}_{i,j}\\&\times\left(\Delta t\,\boldsymbol{R}^n_{i,j}+\frac{\Delta t}{V_{i,j}}A^n_{+i-\frac12,j}S_{i-\frac12,j}\delta\boldsymbol{U}^*_{i-1,j}\right.\\&\left.+\frac{\Delta t}{V_{i,j}}B^n_{+i,j-\frac12}S_{i,j-\frac12}\delta\boldsymbol{U}^*_{i,j-1}\right)\end{aligned}\tag{3a}$$

and then obtain $\delta\boldsymbol{U}^n$ using

$$\begin{aligned}\delta\boldsymbol{U}^n_{i,j}=&\delta\boldsymbol{U}^*_{i,j}-\left[I+\lambda^n_A I+\lambda^n_B I+\Delta t\,\mathrm{diag}(C^n)\right]^{-1}_{i,j}\\&\times\frac{\Delta t}{V_{i,j}}\left(A^n_{-i+\frac12,j}S_{i+\frac12,j}\delta\boldsymbol{U}^n_{i+1,j}+B^n_{-i,j+\frac12}S_{i,j+\frac12}\delta\boldsymbol{U}^n_{i,j+1}\right)\end{aligned}\tag{3b}$$

These sweeps can be vectorized with long vector lengths by ordering the data along the grid diagonal.

To solve Eq. (2) efficiently on a fine-grained massively parallel computer, we must modify the method to reduce or eliminate the data dependencies that are inherent in Eq. (2) or in Eq. (3). That is, the solution at time level $n+1$ should depend only on the solution at the previous time level and not on the solution of neighboring points at the new time level $n+1$. Using two sweeps to solve Eq. (2) is inefficient on parallel computers having a large number of processors because not all of the available processors can be kept busy during each sweep. For example, on three-dimensional problems having a grid size of n^3 implemented on a machine having n^2 processors, a geometric analysis shows that it is not possible to keep more than 5/12 of the processors busy during the sweeps, resulting in poor efficiency. If intermediate quantities not impacted by the data dependencies are precomputed before the sweeps, the efficiency can be increased somewhat, but at the expense of using

about twice as much computer memory. The challenge of efficiently implementing the LU-SGS sweeps in parallel is well illustrated by examining performance results obtained by computer vendors on the APPLU NAS Parallel Benchmark pseudoapplication.[10] This SSOR code has the same dependency graph as the LU-SGS method and parallel efficiencies are typically one-half of that obtained on the other two pseudoapplications (APPBT and APPSP) which are ADI-based schemes.[10,11] Despite this relative inefficiency, it is important to note that the solution times are typically about 10% less than the fastest diagonalized ADI-based method (APPSP).

As the target computer becomes coarser grained (i.e., smaller number of more powerful processors), it is possible to use a domain decomposition method to solve Eq. (3) keeping most of the processors busy throughout the sweeps, but only at the expense of irregular interprocessor communication, a complex mapping of the grid to the available processors, and often smaller than optimal vector lengths at each processor.

It is possible to modify the LU-SGS method to ensure that all of the processors are active at all times on any parallel computer during the solution of Eq. (2). With this change, it is possible to obtain a significant fraction of the theoretical floating point performance of the machine. The suggested approach is to perform a series of subiterations using the following scheme. First, the right-hand side, $\mathbf{R}_{i,j}$ is preconditioned to obtain $\delta \boldsymbol{U}^{(0)}$

$$\delta \boldsymbol{U}^{(0)}_{i,j} = \left\{ I + \lambda^n_A I + \lambda^n_B I + \Delta t \, \mathrm{diag}(C^n) \right\}^{-1}_{i,j} \Delta t \, \boldsymbol{R}^n_{i,j}$$

Then the $k_{\max}$ subiterations are made using for $k = 1, k_{\max}$

$$\delta \boldsymbol{U}^{(k)}_{i,j} = \left\{ I + \lambda^n_A I + \lambda^n_B I + \Delta t \, \mathrm{diag}(C^n) \right\}^{-1}_{i,j} \left\{ \Delta t \, \boldsymbol{R}^n_{i,j} + \frac{\Delta t}{V_{i,j}} \left(A^n_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta \boldsymbol{U}^{(k-1)}_{i-1,j} - A^n_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \delta \boldsymbol{U}^{(k-1)}_{i+1,j} + B^n_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \delta \boldsymbol{U}^{(k-1)}_{i,j-1} - B^n_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \delta \boldsymbol{U}^{(k-1)}_{i,j+1} \right) \right\} \tag{4}$$

then

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

With this approach, the data that are required for each subiteration have already been computed during the previous subiteration. Therefore, the entire subiteration may be performed simultaneously, and there are no data dependencies. Thus, aside from nearest-neighbor communication, a massively parallel computer will approach its peak operational performance.

We call this method a data-parallel LU relaxation method. It retains the lower-upper form from the original LU-SGS method, but it uses a series of point Jacobi-like subiterations to solve the implicit problem. We will see that the method has convergence properties that are similar to the LU-SGS method. The new method may be easily and efficiently expressed in a high-level, data-parallel computer language like FORTRAN90 thereby avoiding explicit management of interprocessor communication and synchronization.

Massively Parallel Computers Used

The following computations were performed on two different types of machines, the MasPar MP-1 and MP-2, and the Thinking Machines CM-5. The MasPar and Thinking Machines computers are very different in their architecture, performance, and cost. Thus, it is not the purpose of this paper to compare their performance, but rather to show that the new LU relaxation method works effectively on two different data-parallel architectures.

The MasPar machines are SIMD computers with the processing elements arranged in a two-dimensional grid interconnection network. On the MasPar machines, the computational speed of the large number of reduced instruction set computer (RISC) processors (1K–16K) is of the same order as the communication bandwidth. Because of this balance, it is possible to do nearest-neighbor communications without incurring major performance costs. The peak performance of the available 8K processor MP-1 is 0.6 Gflops, and the 8K processor MP-2 has a peak speed of 3.2 Gflops.

The Thinking Machines CM-5 used in this work is a 896-processor machine located at the University of Minnesota Army High Performance Computing Research Center. The machine is regularly configured in smaller power-of-two sized machines between 32 and 512 nodes to make it accessible to a variety of problems. Each processor has four vector units that each have a peak performance of 32 Mflops. Thus, the 512-node machine has a theoretical peak performance of 65.5 Gflops. The processors are arranged on a fat-tree interconnection network, which has the geometry of a tree, but with an increasing number of wires closer to the root. Relative to the speed of the processors, the network is very slow because it has a large latency, and its bandwidth is only 5 Mbytes/s. Thus, it is mandatory to minimize the communication between the processors. However, the nearest neighbor communication routines in FORTRAN90 such as CSHIFT are highly optimized and are generally much faster than general router communication.

The data-parallel LU relaxation method was coded in FORTRAN90 for both architectures. There are some small differences in the codes because of the difference in the communication cost between the machines. The most important cost-saving measure involves the evaluation of the off-diagonal implicit terms that appear on the right-hand side of Eq. (4). For example, $A^n_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta \boldsymbol{U}^{(k-1)}_{i-1,j}$ is a vector quantity that can be computed with local data and then shifted to the right and added. Thus, only the vector needs to be communicated, rather than the Jacobian itself. This approach also drastically reduces the amount of memory required for the implicit method.

Another advantage of the LU relaxation method is that it is relatively insensitive to small changes in implementation. This allows several further approximations which reduce the amount of CPU time and memory required with only a small decrease in convergence rate. First, it is possible to use stored face-centered values for surface normals and cosines, which eliminates the need to either store cell-centered values or recompute them at every time step. Also, the implicit boundary conditions may be treated by setting the boundary $\delta \boldsymbol{U}$ to zero before beginning the implicit subiterations. This greatly reduces the cost of boundary updates, which can be very expensive on the CM-5, and makes the code almost perfectly parallel.

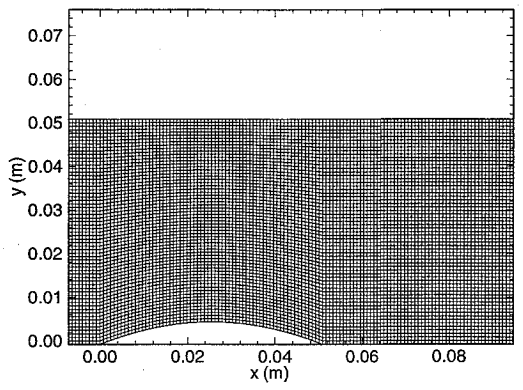
Results and Discussion

In this section, results of the data-parallel LU relaxation method are presented on two different two-dimensional and three-dimensional geometries. Results are presented first for perfect gas solutions, and then extended to five-species reacting flows with a single vibrational temperature. Finally, the floating-point performance of the two massively parallel computers used in this study is discussed.

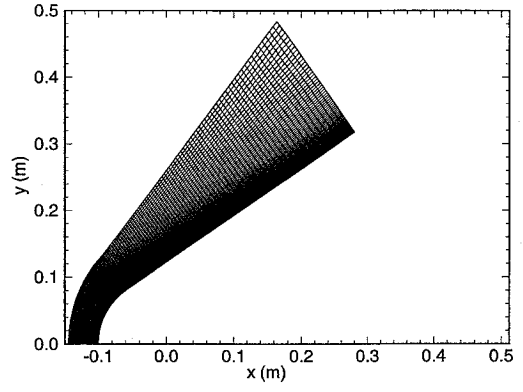
Two different flows were used as test cases for this work. The first is the Mach 2.5 flow over a circular arc airfoil. The mesh is equally spaced in each direction. A sample 128×64 grid is shown in Fig. 1a. The second test case is the Mach 15 flow over a circle-wedge blunt body. The nose radius is 10 cm and the wedge angle is 35 deg. Figure 1b is a plot of a 128×64 grid used to represent this body. The reacting flow simulations were only performed on the circle-wedge geometry. The three-dimensional computations are performed on multiple planes of the same grids, which makes it easy to compare the convergence of the two- and three-dimensional implementations of the data-parallel LU relaxation method.

The explicit fluxes, represented by $\boldsymbol{R}_{i,j}$ in Eq. (3a), are calculated using a modified Steger-Warming flux vector splitting method.[8] The original LU-SGS method has been used with a wide variety of higher order accurate spatial differencing methods[4–7] without significant changes in the convergence properties. Thus, the results presented here are representative of those that would be obtained with other differencing schemes.

All implicit cases were run with essentially infinite time step [Courant-Friedricks-Lewy (CFL) = 10^4]. However, because of the approximate nature of the implicit method, this very large CFL number does not represent the true elapsed time of each iteration. Thus, little improvement in convergence is obtained with larger time steps.



a)



b)

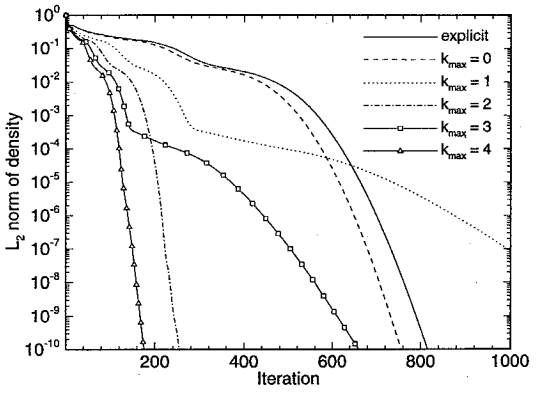
Fig. 1 Sample 128 × 64 grid: a) circular arc airfoil and b) circle-wedge blunt body grid.

Perfect Gas Results

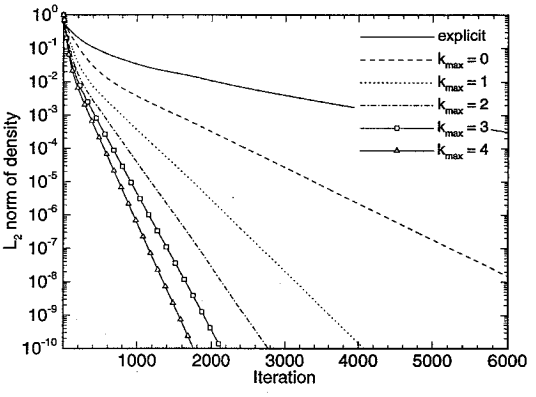
A series of computations was performed on each grid to show how the variation of k_{max} affects the convergence rate of the data-parallel LU relaxation method. Figure 2 shows these results plotted against the number of iterations and computer time for both test cases on a 128 × 64 grid. We can see from Figs. 2a and 2b that both cases show steady improvement in performance as k_{max} increases, although the slender body exhibits poor results for odd values of k_{max}. The convergence rates for values of k_{max} greater than 4 are not plotted because there is no improvement in convergence rate for these cases. The larger number of iterations required to converge the blunt-body flow is caused by the motion of the strong bow shock wave through the grid. The steady-state subsonic region cannot be established until the bow shock has reached its final location.

Figures 2c and 2d plot the convergence rates vs the computer time on a 32-processor partition of the CM-5. Here we see that for the circular arc airfoil, the cost of using $k_{max} = 2$ or 4 is similar. However, because each subiteration has a relatively small cost, $k_{max} = 4$ is the most cost effective for the blunt-body calculation. The data-parallel LU relaxation method converges about twice as fast as the explicit method for the airfoil and at least seven times faster for the blunt-body geometry. Thus, even without the stringent time-step limitation associated with chemically reacting flow simulations, the data-parallel LU relaxation method is attractive.

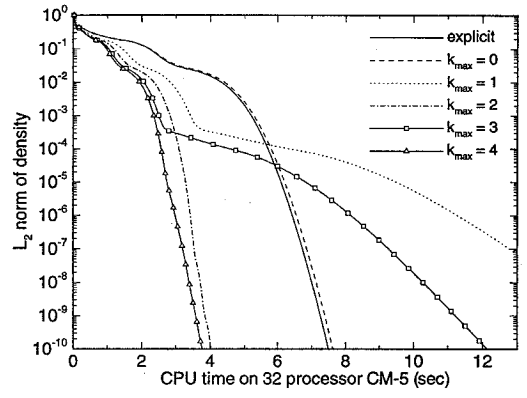
Figures 3a and 3b compare the convergence rate of the new data-parallel LU relaxation method with that of the original LU-SGS method. The same test cases are used on the 128 × 64 grids. We see that the original method and $k_{max} = 2$ give similar performance, whereas $k_{max} = 4$ converges in fewer iterations. This result is surprising because the LU-SGS method uses sweeps to solve Eq. (3), which should allow information to travel across the entire grid during each implicit time step; whereas with the data-parallel LU relaxation method, data can only travel k_{max} grid points per time step. Thus, it would appear that the convergence properties of the new method should be inferior to the old method. However, in practice this is not the case. It is meaningless to compare the computer time required



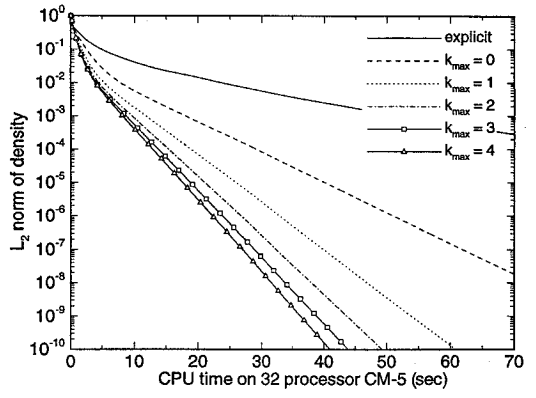
a)



b)

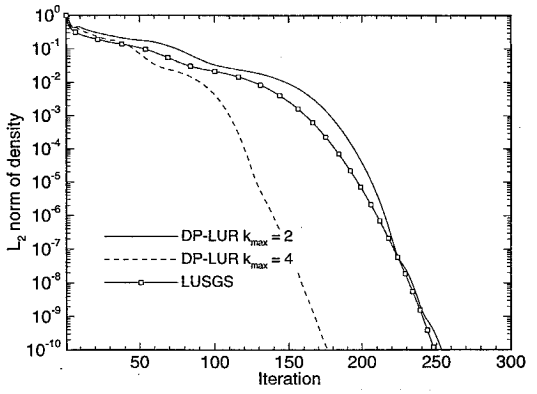


c)

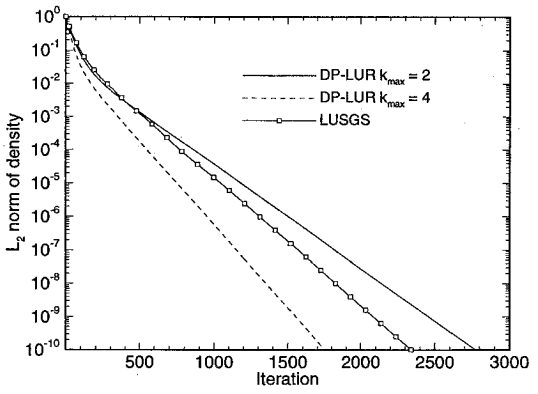


d)

Fig. 2 Convergence histories for the data-parallel LU relaxation method showing influence of k_{max} 128 × 64 grid: a) perfect gas circular arc airfoil at $M_\infty = 2.5$; b) perfect gas circle-wedge blunt body at $M_\infty = 15$; c) perfect gas circular arc airfoil at $M_\infty = 2.5$; and d) perfect gas circle-wedge blunt body at $M_\infty = 15$.



a)



b)

Fig. 3 Convergence histories for the data-parallel LU relaxation and original LU-SGS methods, 128 × 64 grid: a) perfect gas circular arc airfoil at $M_\infty = 2.5$ and b) perfect gas circle-wedge blunt body at $M_\infty = 15$.

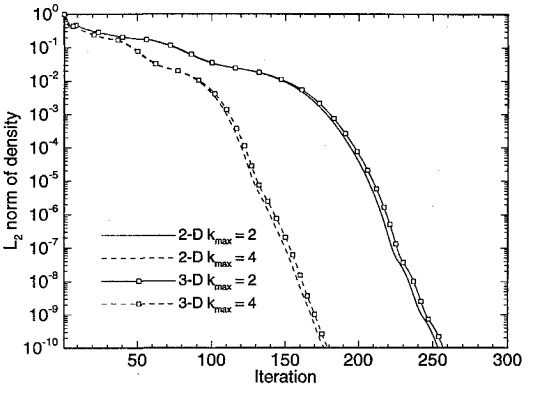


Fig. 4 Comparison of convergence rates for two- and three-dimensional versions of the data-parallel LU relaxation method, 128 × 64 grid: perfect gas circular arc airfoil at $M_\infty = 2.5$.

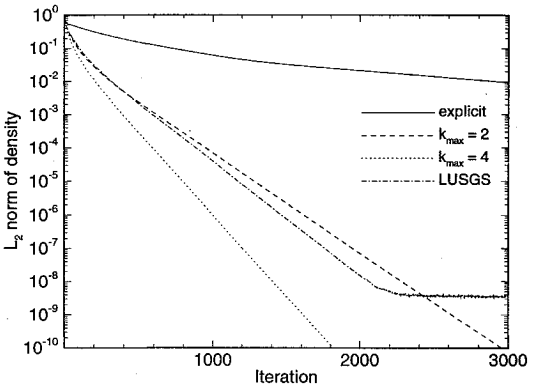


Fig. 5 Convergence histories for the data-parallel LU relaxation and original LU-SGS methods, 128 × 64 grid: five species reacting air circle-wedge blunt body at $M_\infty = 15$ and 60-km altitude.

for the two methods because they are designed for use on different computer architectures. However, it is possible to compare the floating-point operations required to implement the implicit portion of each method. The original LU-SGS method requires 481 floating-point operations per grid point for a two-dimensional perfect gas, whereas the new LU relaxation method takes 490 operations for two subiterations and 869 operations for four subiterations. Thus, leaving aside machine architecture concerns, the original method and the new method with $k_{\max} = 2$ would require essentially the same computer time.

Figure 4 plots the convergence rate of the two- and three-dimensional LU relaxation methods for the perfect gas circular arc airfoil. We see that there is essentially no change in the convergence rate for this case. The same behavior is seen for the blunt body.

Five-Species Reacting Air Results

The new data-parallel LU relaxation method was implemented for a five-species chemical and vibrational nonequilibrium model for air. Standard reaction rates are used[12] along with the elemental density formulation already discussed.

Figure 5 presents a comparison of the convergence history for the explicit method, the original LU-SGS method, and the data-parallel LU relaxation method. The freestream conditions are Mach 15 air at 60 km for the blunt-body geometry. We see a slightly degraded convergence rate for this case as compared the nonreacting perfect gas flow. There is a 10 order-of-magnitude reduction in the residual after 1800 iterations for $k_{\max} = 4$ and after 3000 iterations for $k_{\max} = 2$ (compared with 1700 and 2800 for the perfect gas, respectively). Again, the $k_{\max} = 4$ convergence is superior to the original LU-SGS method. Many other flow conditions have been tested. Generally, the convergence rate degrades with increasing stiffness (increasing density or speed), however, Fig. 5 is indicative of the performance obtained for most cases.

The results presented in the preceding figures show that the data-parallel LU relaxation method achieves convergence in fewer iterations than the original scalar-vector version. It has been shown to yield nearly the same convergence rate for a reacting flow as for a perfect gas flow. The next section discusses the performance of the method on two different massively parallel computer architectures.

Parallel Performance of Method

The data-parallel LU relaxation method was implemented on the MasPar MP-1 and MP-2 and on the Thinking Machines CM-5 in FORTRAN90. The codes were optimized using the performance monitors available on the machines. The intrinsic data-parallel communication operations such as CSHIFT were used, and no extraordinary measures were used to improve the performance of the method.

A number of computations were performed to determine how the performance of the method changes with the size of the problem and the size of the partition of the CM-5. For the two-dimensional code, Fig. 6a shows that the performance of the method depends primarily on the number of points on each processor. For any partition size, if there are 256 grid points per processor, each processor runs at 19 Mflops. If there are 32K points (which is the memory limit), each processor performs at 38 Mflops. Thus, the method is perfectly scalable from small numbers of processors to the full 512-node machine. The performance increases with the number of grid points per processor because that implies an increased vector length. Figure 6b plots the same information in a different format. Here, the overall performance of each partition is plotted after it has been scaled to the equivalent performance on a 512-processor CM-5. (This scaling is valid because Fig. 6a shows that the performance per processor is independent of the partition size.) Figure 6b shows that the performance of each partition increases with the number of grid points in the calculation, and that the equivalent performance asymptotes to 19.5 Gflops.

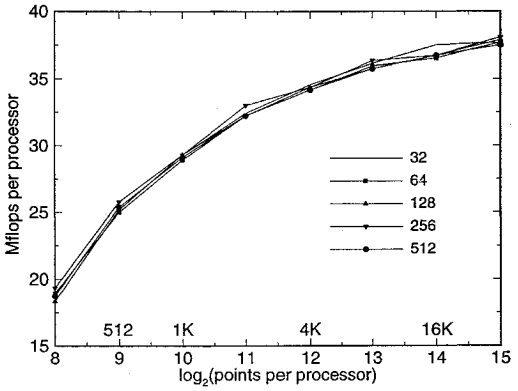


Fig. 6a Per-processor floating-point performance of the two-dimensional perfect gas data-parallel LU relaxation method as a function of the number of grid points per processor for different partitions of CM-5.

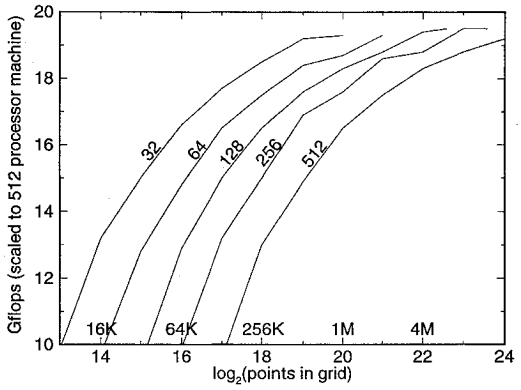


Fig. 6b Floating-point performance of the two-dimensional perfect gas data-parallel LU relaxation method as a function of the number of grid points on different partitions of CM-5; performance scaled to a 512-processor CM-5.

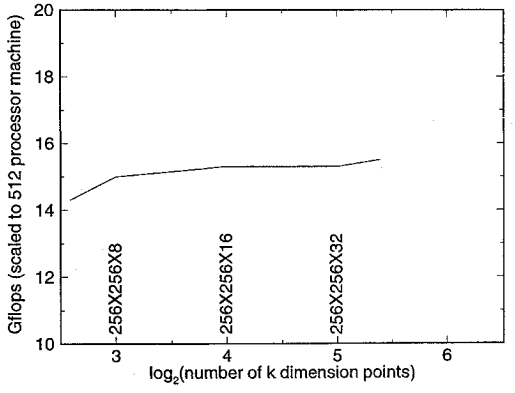


Fig. 7 Floating-point performance of the three-dimensional perfect gas data-parallel LU relaxation method with different serial dimension (k) lengths, results obtained on a 32-processor CM-5; performance scaled to a 512-processor CM-5.

Figure 7 presents the performance of the three-dimensional perfect gas code as a function of the number of grid points in the third dimension (k), which is treated as a serial dimension. That is, no interprocessor communications occur along this axis. Therefore, the performance of the code does not vary appreciably as this dimension is changed except for k dimensions less than 8. Because there are no nonlocal communications in the serial direction, the three-dimensional code performs better overall than the two-dimensional code for the same number of points in the parallel (i-j) plane. This may be seen in Fig. 8.

The MasPar machines do not have the same type of performance variation shown for the CM-5. Provided that the grid in the parallel dimensions (i-j plane) is an even multiple of the number of processors, there is little variation in the overall performance of the method when the grid size is varied. This is because the nodes are scalar processors and because the machine's grid-based architecture correctly matches the structure of the grid.

Table 1 Number of floating-point operations[a] per grid point required for each portion of the data-parallel LU relaxation method, for the four implementations of the method

Function	2-D perf	3-D perf	2-D chem	3-D chem
Explicit	1036	1521	2177	3005
Implicit[b]	869	1680	2363	4723
Chemistry	—	—	1472	1472
Total	1905	3201	6012	9200

[a] Arithmetic counted as one operation, square root and exponentiation counted as eight operations.
[b] for $k_{max} = 4$.

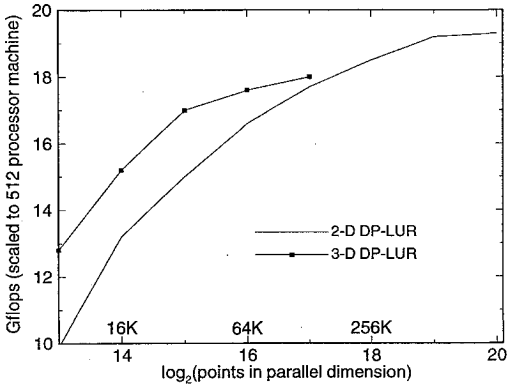


Fig. 8 Floating-point performance of the two- and three-dimensional perfect gas data-parallel LU relaxation methods as a function of the parallel dimension (i-j) size, results obtained on a 32-processor CM-5; performance scaled to a 512-processor CM-5.

Table 1 presents the number of floating-point operations per grid point required for each of the four different implementations of the method (two- and three-dimensional versions of the perfect gas and reacting gas algorithms). The explicit subroutine includes calculation of the fluxes as well as the solution update to the next time level. The implicit portion contains the implementation of the LU relaxation method, and the chemistry subroutine entails the calculation of the source term vector and required implicit derivatives. There are several things to note about the number of floating-point operations required for each portion of the codes. For $k_{max} = 4$, the implicit portion of the method requires about the same number of floating-point operations as the calculation of the fluxes. The three-dimensional version of the code requires about 50% more operations than the two-dimensional code. When the conservation equations required to represent the chemical reactions and vibrational relaxation are included, the number of floating-point operations essentially doubles for the flux calculations and triples for the implicit portion of the code. The doubling of the cost for the flux calculation is a result of solving five additional equations at each grid point. The increased cost of the implicit portion of the code comes from having to compute more complicated pressure derivatives for the reacting code. Also, the computation of the vibrational temperature is very intensive.

Tables 2 and 3 present the floating-point performance of each section of the data-parallel LU relaxation method for specific grids. Table 2 shows the performance of the Thinking Machines CM-5 for the four different versions of the method (two-dimensional and three-dimensional perfect and reacting gases). The results were obtained on a 32-processor partition of the CM-5 and were then scaled to the performance that would be obtained on a 512-processor machine. Again, this scaling is done so that it is easy to compare relative performance across different partition sizes. Because the performance of the machine depends on the number of grid points per processor,

Table 2 Floating-point performance[a] for each version of the data-parallel LU relaxation code on the 32-processor partition of the CM-5[b] scaled to 512-processor machine

Function	2-D perf, Gflops	3-D perf, Gflops	2-D chem, Gflops	3-D chem, Gflops
Explicit	20.0	17.8	16.9	17.1
Implicit	18.3	18.2	15.7	15.1
Chemistry	——	——	12.0	11.8
Total	19.3	18.0	14.1	13.9

[a]Based on beta version of the software and, consequently, not necessarily representative of the performance of the full version of this software.

[b]Two-dimensional cases run on a 512 × 1024 grid, three-dimensional cases run on a 256 × 512 × 8 grid.

Table 3 Floating-point performance for the three-dimensional chemically reacting flow version of the data-parallel LU relaxation code on the 8K processor MP-1 and MP-2[a]

Function	MP-1, Gflops	MP-2, Gflops
Explicit	0.377	1.141
Implicit	0.476	1.690
Chemistry	0.297	0.919
Total	0.382	1.240

[a]Single-precision arithmetic on a 128 × 64 grid.

the results presented in Table 2 should only be used to compare the relative efficiencies of each portion of the code. Note that for the perfect gas codes, the implicit portion of the code runs very efficiently on the CM-5, with a speed of up to 18.3 Gflops obtained. This is a sizable portion (28%) of the peak theoretical speed of a 512-processor CM-5, which is 65.5 Gflops. The explicit portion of the code runs very efficiently with an overall performance of 20.0 Gflops (31% of peak) for the two-dimensional case. The fluxes are computed at a rate of 22.2 Gflops, whereas the solution update runs at only 1.7 Gflops. The update to the boundary requires a router communication, making it very costly (about 10% of the total time of the entire explicit code). The evaluation of the chemical source vector is considerably slower than the other portions of the code, which is surprising because the source vector calculation is entirely pointwise and requires no nonlocal data. The reduced performance appears to be the result of the implementation of the exponentiation operation on the CM-5. At each point it is necessary to compute 37 exponentials during the evaluation of the chemical source term; these operations take about 50% of the total time of the source term evaluation, which requires a total of 1472 floating-point operations.

Table 3 shows the performance of the three-dimensional reacting gas data-parallel LU relaxation method on the MasPar MP-1 and MP-2. In this case, we achieve a higher percentage of the theoretical peak performance of the machine. The theoretical peak speed of an 8K processor MP-1 is 0.60 Gflops; the code obtains 0.38 Gflops (64% of peak) overall and 0.48 Gflops (79% of peak) in the implicit portion of the code. Again, the implementation of the exponential function appears to cause the chemistry routine to be slower than the other portions of the code. Smaller percentages of the 3.2-Gflops peak theoretical speed of the 8K processor MP-2 are obtained. Overall, the MP-2 runs 3.25 times faster than the MP-1, which is 39% of the peak speed.

Conclusions

The lower-upper symmetric Gauss-Seidel method of Yoon and Jameson has been modified to make it amenable to data-parallel machines. The resulting data-parallel lower-upper relaxation method replaces the Gauss-Seidel sweeps of the original method with a series of point Jacobi-like subiterations. In this manner all data dependencies are removed, and the new method becomes almost perfectly parallel. The data-parallel LU relaxation method eliminates the need for the complicated domain decomposition routines required by more exact implicit parallel methods and is easy to implement on a wide variety of parallel architectures in either SIMD or MIMD mode. The method also retains the convergence properties of the original LU-SGS algorithm, and with four subiterations converges in fewer iterations for all inviscid cases tested to date.

This new method is implemented on two different massively parallel architectures, the MasPar MP-1 and MP-2 and the Thinking Machines CM-5, with large percentages of the peak theoretical performance of each machine obtained. Floating-point performance on the CM-5 is primarily a function of the number of grid points per processor, due to the vector nature of the machine, and the algorithm is shown to be perfectly scalable to larger numbers of processors. For very large perfect gas problems the 512-processor CM-5 runs at about 20 Gflops, although the performance of the three-dimensional algorithm is slightly limited due to memory considerations. However, the overall memory use is low, making it possible to run 32M point perfect gas grids on the 512-processor machine. The reacting air code is somewhat slower due to the calculation of the required exponential functions and uses about twice as much memory. The floating-point performance on the MasPar machines is also good, with an overall performance of 0.38 Gflops on the MP-1 and 1.25 Gflops on the MP-2 for the perfect gas codes. As expected, the performance on the MasPar machines is independent of problem size as long as load balancing is ensured.

The high parallel efficiency and low memory requirements of this new data-parallel LU relaxation method make it useful for the solution of very large inviscid perfect gas and reacting flow problems.

Acknowledgments

The first two authors were supported by the NASA Langley Research Center Contract NAG-1-1498. This work was supported in part by the Army Research Office Contract DAALO3-89-0038 with the University of Minnesota Army High Performance Computing Research Center (AHPCRC) and the Department of Defense Shared Resource Center at the AHPCRC. The time on the MasPar computers was granted by the MasPar Computer Corp.

References

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

[2]Obayashi, S., "Numerical Simulation of Underexpanded Plumes Using Upwind Algorithms," AIAA Paper 88-4360, July 1988.

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

[4]Park, C., and Yoon, S., "Calculation of Real-Gas Effects on Blunt-Body Trim Angles," AIAA Paper 89-0685, Jan. 1989.

[5]Eberhardt, S., and Imlay, S. T., "A Diagonal Implicit Scheme for Computing Flows with Finite-Rate Chemistry," AIAA Paper 90-1577, June 1990.

[6]Imlay, S., Roberts, D., Soestrisno, M., and Eberhardt, S., "Nonequilibrium Thermochemical Calculations Using Diagonal Implicit Scheme," AIAA Paper 91-0468, Jan. 1991.

[7]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.

[8]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.

[9]Van der Vegt, J. J. W., "Assessment of Flux Vector Splitting for Viscous Compressible Flows," AIAA Paper 91-0242, Jan. 1991.

[10]Bailey, D., Barton, J., Lasinski, T., and Simon, H., eds., "The NAS Parallel Benchmarks," NASA TM-103863, July 1993.

[11]Bailey, D., Barszcz, E., Dagum, L., and Simon, H., "NAS Parallel Benchmark Results 3-94," NASA RNR TR-RNR-94-006, March 1994.

[12]Park, C., Howe, J. T., Jaffe, R. L., and Candler, G. V., "Review of Chemical-Kinetic Problems of Future NASA Missions, II: Mars Entries," *Journal of Thermophysics and Heat Transfer,* Vol. 8, No. 1, 1994, pp. 9–23.