AIAA JOURNAL
Vol. 36, No. 11, November 1998

Parallel Processing Scheme for the Navier–Stokes Equations, Part 1: Scheme Development

N. Ghizawi* and S. Abdallah†
University of Cincinnati, Cincinnati, Ohio 45221-0070

Using a lower and upper decomposition procedure, an implicit parallel processing scheme for the compressible Navier–Stokes equations is developed. In this scheme, the solution for a multidimensional problem is obtained by the superposition of two independent solution cycles. Because of the independence of these two solutions, they can be performed on separate processors simultaneously. To confirm the analytical developments, numerical results are obtained using the proposed scheme and compared with other available solutions and experimental data. A study of other characteristics of the proposed scheme such as stability, symmetry preserving, and effects of domain decomposition on convergence and stability is also conducted.

Introduction

RECENT activities on parallel processing have attracted a great deal of research in computational fluid dynamics (CFD). Progress in CFD is being made in two directions: treating more complex and realistic configurations and developing algorithms based on higher approximations to the full Navier–Stokes equations.

Two basic types of time-marching schemes are used for solving the time-dependent Navier–Stokes equations, namely, explicit and implicit schemes. In this work, only implicit schemes are considered because they are generally highly stable and robust as opposed to explicit schemes. However, as far as parallel processing is concerned, implicit schemes are more difficult to parallelize primarily because of the inherent global spatial dependencies for the solution of large systems in the form of block tridiagonal and/or pentadiagonal matrices.

Approximate factorization methods[1–3] are very popular for solving the Navier–Stokes equations. Some of the employed factorizations produce factors corresponding to each coordinate that are exactly similar to those resulting from the alternating direction implicit (ADI) methods.[1,2,4,5] Other factorizations produce lower-upper (LU) factorizations.[3,6,7] Jameson and Yoon[3] reported that perhaps the biggest drawback of the ADI schemes is the poor high-frequency damping characteristic in three dimensions due to the presence of the error terms of order Δt^3, which may reduce the rate of convergence. In fact, ADI schemes are unconditionally unstable in three dimensions if factorization error-free steady-state solutions (by using the delta form) to the Navier–Stokes equations are sought.[3,7] LU schemes, on the other hand, are unconditionally stable in any number of space dimensions. In addition, inversion of the ADI operators can be costly if applied to a large system of equations. Efficient and robust LU schemes, however, have succeeded in eliminating the disadvantages of the ADI schemes. Therefore, LU implicit schemes are the ones considered in this work.

Upon factorization, the factors (steps) resulting from factoring the multidimensional problem using an LU (or an ADI) scheme are dependent on each other where the solution from the first factor (step) is needed to be able to solve the second step, and so on. Therefore, these steps have to be performed in series, which limits the parallelization features for these types of factorizations. In fact, parallelization in this case is limited to the simultaneous processing of subdomains on separate processors simultaneously. The boundary conditions at the subdomains' interfaces are usually timelagged to enable solving each subdomain separately.

Recently, a new class of implicit schemes for solving the time-dependent Navier–Stokes equations was proposed.[8–11] The novelty of this class of schemes is that the factors (steps) resulting from factoring the multidimensional problem are independent of each other; therefore, parallel processing can be used to enhance computations of large problems. Abdallah[8] proposed an implicit scheme that superposes two or three one-dimensional solutions in two or three dimensions in a finite difference formulation to solve the incompressible Navier–Stokes equations. Ghizawi et al.[9] and Ghizawi and Abdallah[10,11] demonstrated this cycle-independent scheme using a finite volume flux-splitting formulation applied to the thin-layer Navier–Stokes equations. In all of these studies, the resulting factors were all ADI-like factors, and no LU factorizations were pursued.

In this paper, a new lower-upper cycle-independent (LUCI) implicit factorization is proposed to solve the compressible Navier–Stokes equations. The LUCI factorization has the combined advantages of the stability of LU factorizations as well as the cycle independency that gives it better parallel processing functionality. The stability of this scheme is examined by applying the von Neumann stability analysis[12] to the linearized two-dimensional viscous Burger equation. Flow computations for two test cases are performed to check for the symmetry preserving property[3] and to demonstrate the accuracy of the proposed LUCI scheme. These test cases are an inviscid transonic flow over a NACA 0012 airfoil at a zero angle of attack and the interaction of an oblique shock wave with a laminar boundary layer developing over a flat plate leading to the boundary-layer separation.[13] A comparison with other experimental and numerical results is conducted. Finally, effects of time lagging the subdomain boundary information data on the convergence and stability characteristics of the proposed LUCI scheme as well as the symmetric successive overrelaxation (SSOR) scheme[3] (a typical cycle-dependent scheme) are simulated using pseudoparallelism. A staggered biplane configuration is used as the test case to perform this analysis.

Governing Equations

The nondimensional form of the compressible, two-dimensional, time-dependent Navier–Stokes equations (without body force or external heat addition) can be written in

$$\tau = t, \qquad \zeta = \zeta(x, y, t), \qquad \eta = \eta(x, y, t) \tag{1}$$

Applying the coordinate transformation given by Eqs. (1), the two-dimensional Navier–Stokes equations become[14]

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial \zeta} + \frac{\partial G}{\partial \eta} = \frac{\partial F^v}{\partial \zeta} + \frac{\partial G^v}{\partial \eta} \tag{2}$$

Received Dec. 5, 1997; revision received July 10, 1998; accepted for publication July 26, 1998.

*Graduate Assistant, Department of Aerospace Engineering and Engineering Mechanics; currently Aerodynamicist, AlliedSignals Turbocharging Systems, Mail Stop R12, 3201 West Lomita Boulevard, Torrance, CA 90505. Member AIAA.

†Professor, Department of Aerospace Engineering and Engineering Mechanics. Member AIAA.

where Q is the vector of dependent variables, F and G are the inviscid flux vectors and F^v and G^v are the viscous flux vectors.

Analysis

A generic implicit method for solving Eq. (2) is

$$\Delta Q^n + \Delta\tau\left(\delta_i F^{n+1} + \delta_j G^{n+1}\right) = \Delta\tau\left(\delta_i F^{v(n+1)} + \delta_j G^{v(n+1)}\right) \tag{3a}$$

where

$$\Delta Q^n = Q^{n+1} - Q^n \tag{3b}$$

and the various δ are central difference operators in computational space.

As suggested by Gatlin,[15] the diffusive terms can be treated explicitly as time-lagged source terms. Pulliam and Steger[5] routinely neglected the implicit viscous terms when applying the NASA Ames Research Center codes to steady viscous flows and convection-dominated flows. Therefore, it is assumed that

$$F^{v(n+1)} = F^{v(n)} + \mathcal{O}(\Delta\tau) \tag{4}$$

with similar expressions for G^v.

Applying Eq. (4), splitting the inviscid fluxes according to the sign of their eigenvalues, linearizing them, and rearranging, Eq. (3a) becomes

$$\left[I + \Delta\tau\left(\delta_i^- A^+ + \delta_i^+ A^- + \delta_j^- B^+ + \delta_j^+ B^-\right)\right]\Delta Q^n = R^n \tag{5a}$$

where

$$R^n = \Delta\tau\left(\delta_i F^{v(n)} + \delta_j G^{v(n)} - \delta_i^- F^{+(n)} - \delta_i^+ F^{-(n)} - \delta_j^- G^{+(n)} - \delta_j^+ G^{-(n)}\right) \tag{5b}$$

$$A^+ = \frac{\partial F^+}{\partial Q}, \qquad A^- = \frac{\partial F^-}{\partial Q} \tag{5c}$$

with similar definitions for B^+ and B^-. The δ_i^- and δ_j^- denote the backward difference operators and δ_i^+ and δ_j^+ denote the forward difference operators.

Two famous schemes for evaluating the numerical fluxes at cell faces are the flux difference split (FDS) schemes[14,16,17] and the flux vector split (FVS) schemes.[18,19] However, as suggested by Whitfield et al.,[20] improved results and convergence can be obtained if the implicit portion of Eq. (5a) is based on the FVS scheme and the explicit portion (residual vector) is based on the FDS algorithm. The FVS scheme used here is based on Steger–Warming splitting,[21] and the FDS scheme is based on Roe's approximate Reiman solver.[22] Therefore, Eq. (5b) is modified to be

$$R^n = \Delta t\left(\delta_i F^{v(n)} + \delta_j G^{v(n)} - \delta_i F_r^{(n)} - \delta_j G_r^{(n)}\right) \tag{5d}$$

where the subscript r refers to the fact that Roe's averaged variables[22] are used to evaluate the fluxes and the eigenvalues at the cell face. The evaluation of the split fluxes in Eq. (5d) is based on the MUSCL technique with the minmod flux-limiter function used to avoid oscillatory solutions.[23] Third-order extrapolation at a cell interface is perfomed using the formulas given in Ref. 23. The method used to evaluate the viscous diffusive terms in the residual R^n is done by a simple central difference from first derivative values computed on opposing cell faces.

Lower-Upper Cycle-Independent Method

Equation (5a) can be discretized and rearranged to give

$$N\Delta Q^n_{i,j} + \Delta\tau\left(A^-_{i+1,j}\Delta Q^n_{i+1,j} - A^+_{i-1,j}\Delta Q^n_{i-1,j}\right) + \Delta\tau\left(B^-_{i,j+1}\Delta Q^n_{i,j+1} - B^+_{i,j-1}\Delta Q^n_{i,j-1}\right) = R^n \tag{6a}$$

where

$$N = \left[I + \Delta\tau\left(A^+_{i,j} - A^-_{i,j}\right) + \Delta\tau\left(B^+_{i,j} - B^-_{i,j}\right)\right] \tag{6b}$$

Now, we propose to solve Eq. (6a) using

$$\Delta Q^n = \Delta Q^n_1 + \Delta Q^n_2 - N^{-1}R^n \tag{7}$$

where ΔQ^n_1 and ΔQ^n_2 are computed according to

$$N\Delta Q^n_{1_{i,j}} - \Delta\tau\left(A^+_{i-1,j}\Delta Q^n_{1_{i-1,j}} + B^+_{i,j-1}\Delta Q^n_{1_{i,j-1}}\right) = R^n \tag{8a}$$

$$N\Delta Q^n_{2_{i,j}} + \Delta\tau\left(A^-_{i+1,j}\Delta Q^n_{2_{i+1,j}} + B^-_{i,j+1}\Delta Q^n_{2_{i,j+1}}\right) = R^n \tag{8b}$$

Now the LUCI solution procedure is summarized in solving Eqs. (8a) and (8b) at every time step and then using Eq. (7) to update our solution before we move on to the next time step. This is done until convergence is achieved. Equations (8a) and (8b) may be written as

$$L\Delta Q^n_1 = R^n \tag{9a}$$

$$U\Delta Q^n_2 = R^n \tag{9b}$$

where

$$L = \left[I + \Delta\tau\left(\delta_i^- A^+ + \delta_j^- B^+ - A^- - B^-\right)\right] \tag{10a}$$

$$U = \left[I + \Delta\tau\left(\delta_i^+ A^- + \delta_j^+ B^- + A^+ + B^+\right)\right] \tag{10b}$$

Note here that the lower and upper operators given by Eqs. (10a) and (10b) are the same as those arrived at by Jameson and Yoon[3] in deriving their lower-upper SSOR scheme. However, in the SSOR scheme, the lower and upper steps are solved sequentially. The lower operator step is solved first, and then the upper operator step is done afterwards using the solution achieved in the first step as a source term. On the other hand, in the present LUCI formulation, the lower and upper operator steps [Eqs. (10a) and (10b), respectively] are independent of each other; therefore, they can be implemented concurrently. However, an extra step is needed [Eq. (7)] to get the actual solution. Note that this extra step involves only the superposition of ΔQ^n_1, ΔQ^n_2, and R^n at every point separately; i.e., no system of equations needs to be solved.

To perform Eq. (7), the inverse of the matrix N is needed. This matrix is a 5×5 matrix in the three-dimensional case, and so one possibility is to invert it directly in a symbolic fashion (this is done only once) and to use it when needed. Another possibility, which is used in this work, defines the plus and minus flux Jacobians in the following way,

$$A^+ = 0.5(A + \gamma_A I), \qquad A^- = 0.5(A - \gamma_A I) \tag{11a}$$

$$B^+ = 0.5(B + \gamma_B I), \qquad B^- = 0.5(B - \gamma_B I) \tag{11b}$$

where

$$\gamma_A \geq \max(|\lambda_A|) \tag{12a}$$

$$\gamma_B \geq \max(|\lambda_B|) \tag{12b}$$

where λ_A and λ_B are the spectral radii of the associated flux Jacobians. The purpose of using this method is to make the matrix N diagonal, which can be inverted efficiently without using large storage. A proof that Eq. (7) is a solution for Eq. (6a) is given by Ghizawi.[24] Furthermore, the factorization error (FE) is shown to be

$$\begin{aligned}\mathrm{FE} = {} & -\Delta\tau^2\left(\delta_i^+ A^- + \delta_j^+ B^- + A^- + B^-\right)N^{-1} \\ & \times \left(\delta_i^- A^+ + \delta_j^- B^+ - A^- - B^-\right)\Delta Q^n_1 - \Delta\tau^2 \\ & \times \left(\delta_i^- A^+ + \delta_j^- B^+ - A^+ - B^+\right)N^{-1} \\ & \times \left(\delta_i^+ A^- + \delta_j^+ B^- + A^+ + B^+\right)\Delta Q^n_2\end{aligned} \tag{13}$$

Note that this error tends to zero as the steady-state solution is approached (ΔQ^n_1 and ΔQ^n_2 both approach zero at steady state). Therefore, factorization error-free steady-state solutions are achieved.

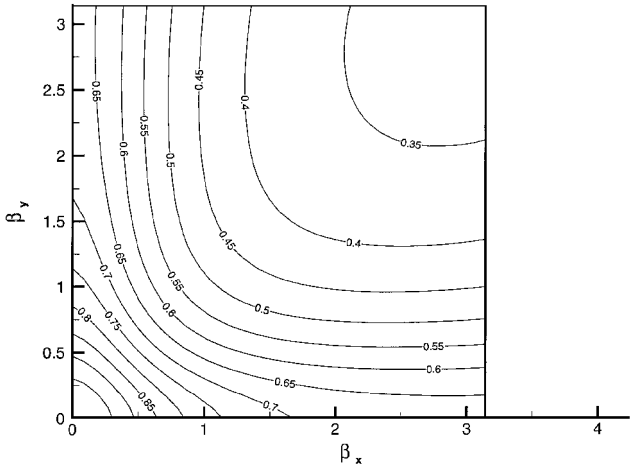


Fig. 1 Amplification factor contours for the LUCI scheme (CFL = 10^5 and $Re_\Delta \rightarrow \infty$).

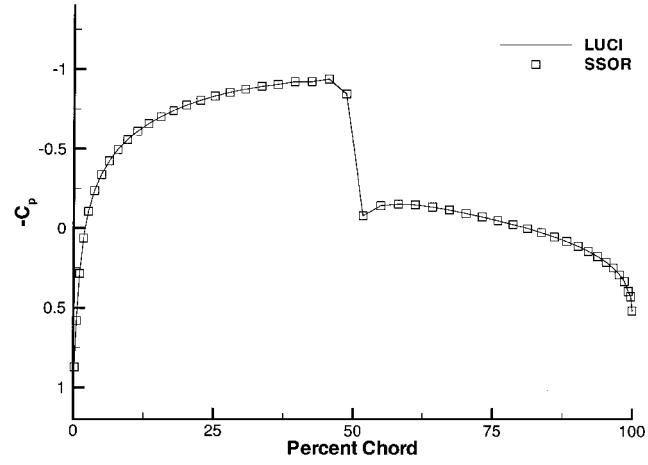


Fig. 2 Comparison of the results for the surface pressure coefficient for the NACA 0012 airfoil problem.

Stability Analysis

The stability of the present scheme was examined by applying the von Neumann stability analysis to the simple model problem

$$U_\tau + \lambda_1 U_x + \lambda_2 U_y = \nu(U_{xx} + U_{yy}) \tag{14}$$

Figure 1 shows the amplification factor G contours obtained for a Courant–Friedrichs–Lewy (CFL) number of 10^5 and a mesh Reynolds number $Re_\Delta \rightarrow \infty$ (inviscid case) plotted against the wave numbers in the x and y directions, β_x and β_y, respectively. The values of the amplification factor do not exceed 1 for any combination of the wave numbers. Thus, it appears that the LUCI scheme is unconditionally stable in this case.

Numerical experiments with the present scheme applied to the Navier–Stokes equations have confirmed the same conclusion mentioned earlier, namely, the unconditional stability. However, because of the explicit treatment of the diffusive terms, the scheme is highly stable for high-Reynolds-number flows (convection-dominated flows), whereas for low-Reynolds-number flows the scheme becomes unstable. This is certainly a disadvantage of time lagging the diffusive terms, but it also has the advantage of not destroying the simply and efficiently solved tridiagonal systems of the schemes developed for the Euler equations. Therefore, it is possible to achieve useful viscous solutions for a wide class of aerodynamic problems with only a modest increase in cost over inviscid analyses.

Scheme Validation on a Single Processor

Two numerical examples of two-dimensional flow computations are included to demonstrate the accuracy of the proposed LUCI scheme. The first test problem is that of an inviscid transonic flow ($M_\infty = 0.8$) over a NACA 0012 airfoil at a zero angle of attack. Even though this is a classical problem, it is an important test case that it is often used to evaluate the accuracy of any given scheme.[3] This nonlifting flow case is modeled by solving the flow over both the lower and upper surfaces of the airfoil without invoking the symmetry in any way. Inaccurate schemes may fail to preserve symmetry, and techniques such as alternating sweeps or using different systems of equations for the lower and upper half-planes are resorted to get symmetrical solutions.[3]

An O-mesh was used to solve this problem with 100 cells in the direction around the airfoil and 48 cells in the radial direction. A CFL number of 10^5 was used to perform the present computations. To show the symmetry in the results, the pressure coefficient C_P distributions on both the lower and upper surfaces of the NACA 0012 airfoil are given in Fig. 2, and the pressure contours are shown in Fig. 3. The C_P distributions on the lower and upper surfaces of the airfoil match with a maximum percentage difference of 10^{-4}. The basic flow features of the induced shock wave and the symmetry can also be seen in Fig. 3. It is important to emphasize here that symmetry was not accounted for in any way (no alternating sweeps or modified systems of equations for different halves of the plane). Instead, it came out naturally with the solution. This is probably due to the unbiased nature of the cycle-independent class of schemes, as both cycles (steps) have the same right-hand side (which is symmetric to start with).

The popular problem of the interaction of an oblique shock wave with a laminar boundary layer developing over a flat plate[25–27] is used here as our second test case. The shock wave angle ϕ equals 32.6 deg, and this shock is of sufficient strength to cause separation of the boundary layer. The freestream Mach number equals 2.0. The computational domain was defined by $-0.01 \leq x \leq 0.3$ and $0.0 \leq y \leq 0.1215$ with the flat plate starting from $x = 0.03$ (all dimensions are in feet). The shock wave is imposed at $x = 0.0$ and $y = 0.1215$ ft; therefore, its direction intersects the flat plate at $x = 0.19$ ft. The Reynolds number based on a length of 0.16 ft (the length from the leading edge to the shock impingement point) and the freestream conditions equals 2.96×10^5. This model problem corresponds to one of the experiments of Hakkinen et al.[13]

The initial mesh generated for this problem was a Cartesian mesh that contained 32×45 grid points. Uniform (no stretching) distribution of grid points ($\Delta x = 0.01$ ft) was used. Exponential stretching was employed in the y direction with Δy at the flat plate being equal to 0.0001 ft. Two other Cartesian meshes with two (63×89) and three (94×133) times the initial mesh density were generated. Again, a CFL number of 10^5 was used to perform the present computations.

Figure 4 shows the results of the grid refinement study on the wall pressure distribution. Some differences between the results of the first and the second grids in the separation zone and the leading-edge shock area can be seen, whereas no noticeable differences exist between these results for the second and third grids in the separation zone and only slight differences exist in the leading-edge shock area. The solutions from all of the three grids show the leading-edge shock, the initial compression due to the separating boundary layer, and the final recompression due to the turning at the wall. Expansion over the separation bubble can be seen clearly in the results of the second and the third grids.

Comparisons of the wall pressure and the skin-friction coefficient distributions achieved using the LUCI and the SSOR schemes with the experimental data of Hakkinen et al.[13] are given in Figs. 5 and 6, respectively. An excellent agreement between the wall pressure measurements and the present results is achieved, whereas for the skin-friction coefficient the agreement is reasonably good. The absence of experimental data for the skin-friction coefficient in the separation zone is due to the fact that no skin-friction measurements were possible in that zone.[13]

Figure 7 compares the pressure contour pictures obtained using the LUCI and the SSOR schemes for this problem. Both pictures show the weak leading-edge shock, the incident shock, the compression waves ahead of separation, the expansion waves behind the separation bubble followed by recompression, and finally the main reflected shock. The similarity between the two pictures is very clear.

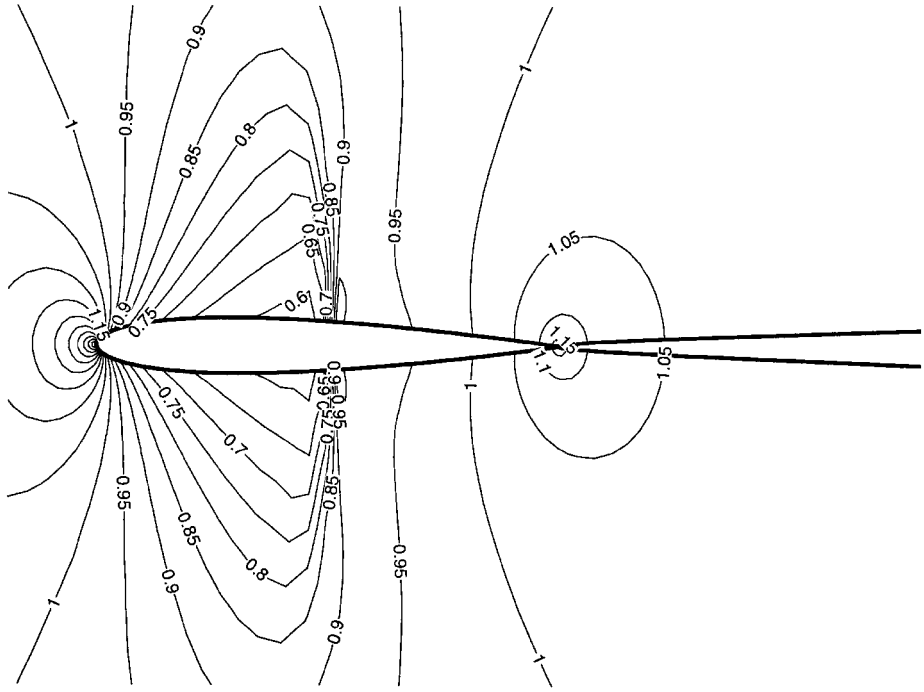


Fig. 3 Pressure contours for transonic flow ($M_\infty = 0.8$) around the NACA 0012 airfoil (LUCI scheme).

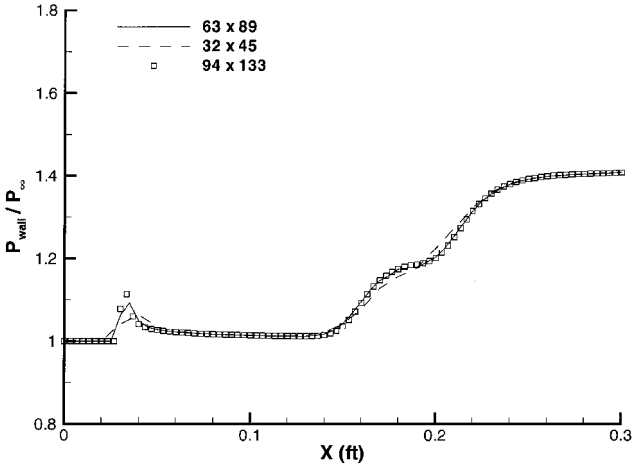


Fig. 4 Wall pressure distribution for different grid sizes (LUCI scheme).

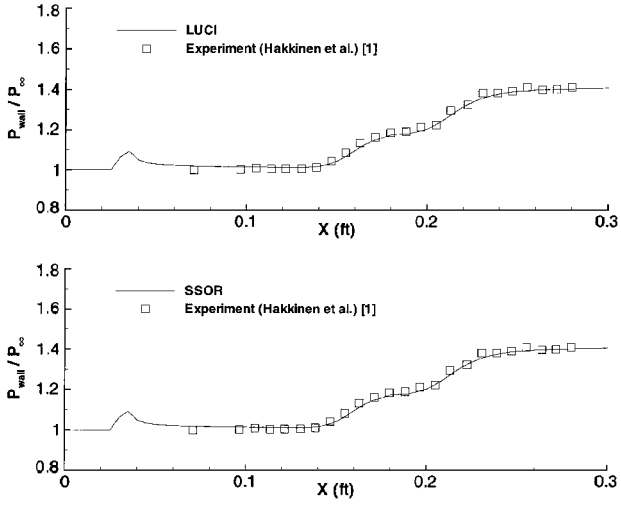


Fig. 5 Comparison of the wall pressure distribution with experimental data.

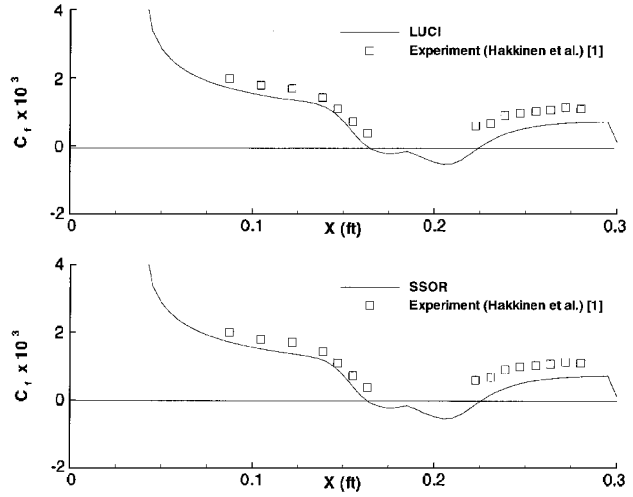
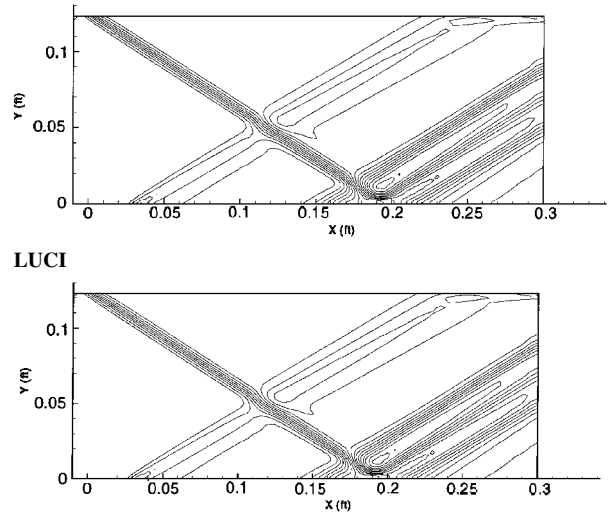


Fig. 6 Comparison of the skin-friction coefficient with experimental data.



LUCI

SSOR

Fig. 7 Comparison of the pressure contours for the shock/boundary-layer interaction problem.

Domain Decomposition: Effects on Stability and Convergence

In this section, parallelism of the LUCI and the SSOR schemes due only to domain decomposition (no cycle independence effects) will be analyzed. To be able to solve any problem in parallel, the problem domain has to be split or decomposed in some way. One of the most important domain decomposition techniques is geometric decomposition.[28] Once the problem domain has been decomposed into several subdomains or blocks, each block or group of blocks gets assigned to a specific processor. Time lagging of flow information at the block interfaces is usually used to enable one to solve each block (subdomain) separately. The purpose of this section is to study the effects of this time lagging of the boundary conditions on the convergence and stability characteristics of the implicit solution scheme. Studying these effects is important for any future successful implementation of the LUCI and the SSOR schemes (or similar schemes) in a massive parallel processing environment; therefore, a numerical experiment was conducted to study these effects.

The test problem we used in our experiment was the staggered biplane configuration[29] shown in Fig. 8. A hybrid C-H-C grid was used to solve this problem. Each of the C-grids contained 51×51 grid points, whereas the H-grid for the channel between the two airfoils contained 101×26 grid points. The number of blocks varied in our experiment between 3 and 1850. Square and rectangular blocks (depending on the specific number of blocks and the grid size in each direction) of nodes were used in this experiment.

If a one block per processor mapping is used, then a maximum of 1850 processors are needed to perform our experiment. This was not feasible for us; therefore, pseudoparallelism was used to simulate the desired stability and convergence deterioration effects using only a single processor. The decomposed (multiblock) problem was put on a single processor, and each block was solved independently. In other words, the blocks were solved in series (one at a time). To do so, flow information at the blocks interfaces was time lagged to simulate the effects of the real parallel implementation on convergence and stability. In fact, this approach can be applied to any partitioning, simulating any number of processors. It alleviates the need to actually have the processors physically available to be able to study the desired convergence and stability characteristics in massively parallel environments.

Obviously, checking for instability is straightforward, because the solution will eventually diverge if the scheme is not stable for a certain partitioning. To identify convergence, the values of the lift and drag coefficients for the forward and aft airfoils were monitored. The local flow parameters (lift and drag coefficients for the forward and the aft airfoils) were considered to be converged when the absolute percentages of the amplitudes of their oscillations relative to their mean values within a certain period (for example, 1000 iterations) were equal to or less than a certain tolerance (3%).

Seven runs were conducted to simulate the stability and convergence characteristics of the LUCI and the SSOR schemes as the number of processors increases. The number of blocks (simulated processors) for each of these runs were as follows: 3, 48, 234, 850, 1302, 1564, and 1850. A CFL value of 10^5 was used in all of these runs. Results for the stability will be presented first, and then the convergence results will follow.

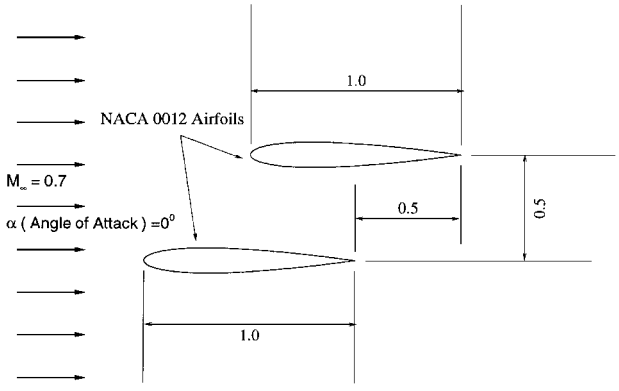


Fig. 8 Staggered biplane configuration.

Table 1 Stability results for a CFL value of 10^5

	Number of blocks						
Scheme	3	48	234	850	1302	1564	1850
LUCI	Stable	Stable	Stable	Stable	Stable	Stable	Stable
SSOR	Stable	Stable	Stable	Stable	Stable	Unstable	Unstable

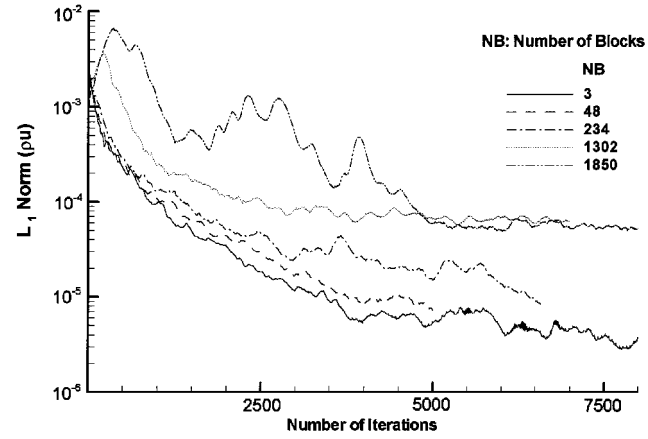


Fig. 9 Convergence history for the LUCI scheme for several partitionings.

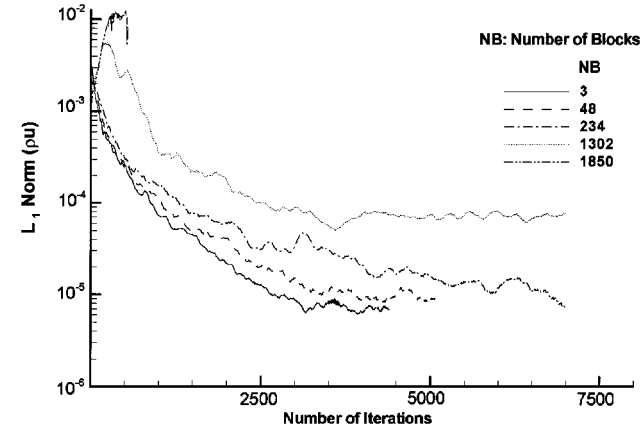


Fig. 10 Convergence history for the SSOR scheme for several partitionings.

Stability Results

The achieved stability results for both the LUCI and the SSOR schemes are summarized in Table 1. These results indicate that the stability of the SSOR scheme breaks down when the number of processors (blocks) exceeds or equals 1564. The LUCI scheme, on the other hand, maintains its favorable unconditional stability characteristics through the whole range for the experiment. It is still stable even in the extreme situation when the number of blocks equals 1850. Based on these results, we can say that, for the specific problem we are solving and for the specific value of the CFL number we used, the LUCI scheme exhibits superior stability characteristics relative to the SSOR scheme when the number of processors (blocks) becomes very large (in the massive range).

Convergence Results

The convergence results for both the LUCI and the SSOR schemes are shown in Figs. 9–12. Figures 9 and 10 show the effect on the convergence of the L_1 norm for both the LUCI and the SSOR schemes, respectively, as the simulated number of processors (number of blocks) varies. For better clarity, only some of the test cases we ran are shown in these figures. The case when the number of blocks equals 1850 is shown in Fig. 9 only to show how the instability started. The first thing we notice from these figures is that convergence is definitely affected by the change in the number of blocks. As the number of blocks increases, the convergence behavior becomes

Table 2 Local convergence of C_L for the aft airfoil using the LUCI scheme

	Number of blocks				
Parameter	3	48	234	1302	1850
Number of iterations	2547	3991	5601	5931	6859
DET, %	0	56.7	119.9	132.9	169.3
Terminal C_L value	0.59446	0.59484	0.59478	0.59447	0.59521

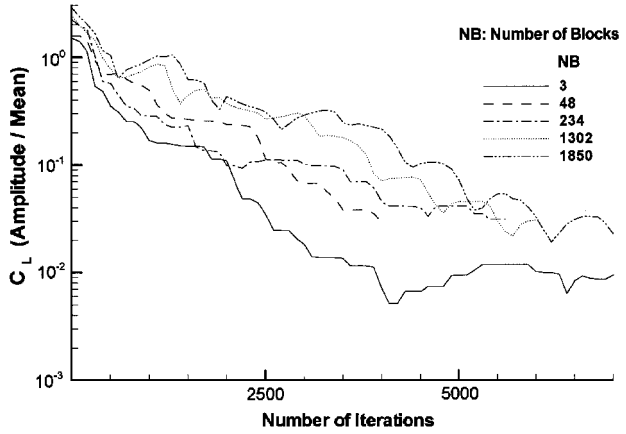


Fig. 11 Convergence of C_L for the aft airfoil for several partitionings (LUCI).

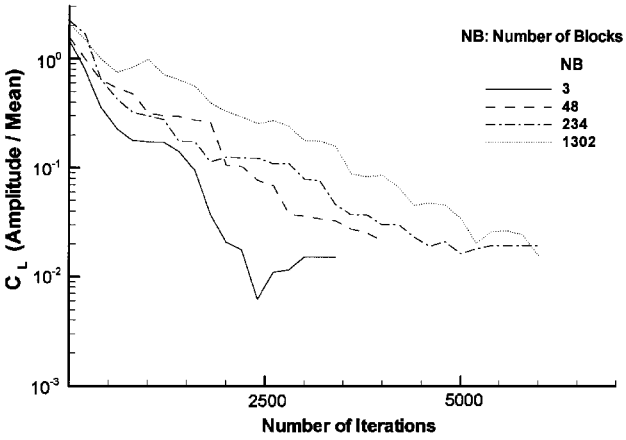


Fig. 12 Convergence of C_L for the aft airfoil for several partitionings (SSOR).

rougher (oscillations start having larger amplitudes) and generally worse (slower convergence rate). For a small number of blocks, the effects on convergence become clearer in the late stages of the iteration process (at larger numbers of iterations). However, when the number of blocks increases, the effects on convergence can be seen very clearly in the very early stages of the iteration process (at small numbers of iterations), and these effects are maintained thereafter. Moreover, the final level to which the L_1 norm drops changes as the number of blocks increases. From Fig. 9, it also appears that the reduction in the level to which the L_1 norm for the LUCI scheme drops coincides with the SSOR scheme becoming unstable. However, even with this convergence degradation, the values of the lift and drag coefficients converged within reasonable engineering accuracy as can be seen in Tables 2 and 3 for the lift coefficient for the aft airfoil.

Figures 11 and 12 show the convergence of the lift coefficient C_L for the aft airfoil using both the LUCI and the SSOR schemes, respectively. Again, local convergence deterioration can be seen in these figures as well. If we define an acceptable local convergence level to be such that the ratio of the amplitude of the oscillation in the value of C_L for the aft airfoil to its mean does not exceed 3%, we can construct Tables 2 and 3.

Table 3 Local convergence of C_L for the aft airfoil using the SSOR scheme

	Number of blocks			
Parameter	3	48	234	1302
Number of iterations	1929	3552	4262	5033
DET, %	0	84.1	120.9	160.9
Terminal C_L value	0.59448	0.59462	0.59468	0.595228

The parameter DET(%) is defined as follows:

$$\mathrm{DET}(\%) = \frac{\mathrm{NOITR\ (NB)} - \mathrm{NOITR\ (3)}}{\mathrm{NOITR\ (3)}} \times 100\%$$

where NB is the number of blocks, and NOITR is the number of iterations needed to satisfy the 3% local convergence criteria.

The case when the number of blocks equals 1850 for the SSOR scheme is not shown in these tables because the scheme is unstable for this case (see Table 1).

In Tables 2 and 3, the parameter DET(%) is the percentage increase in the number of iterations needed to converge as the number of blocks (simulated processors) increases. It thus gives us a quantitative idea about the convergence deterioration experienced by both the LUCI and the SSOR schemes as the number of blocks increases. It can be seen from these tables that the SSOR scheme typically takes fewer iterations to converge relative to the LUCI scheme for a certain number of blocks. However, the SSOR scheme experiences more deterioration in its convergence relative to the LUCI scheme. In fact, when the number of blocks (simulated processors) increases beyond a certain value (1564 in this case), the SSOR scheme becomes unstable, and thus the use of other schemes (the LUCI scheme, for example) is needed.

Concluding Remarks

It is believed that the future for computational fluid dynamics is going to be greatly influenced by the development of parallel architecture computers. A lower-upper cycle-independent implicit algorithm for solving the compressible Navier–Stokes equations on parallel computers has been proposed. Using this algorithm, the solution of a multidimensional problem is split into the solution of two independent problems for any number of space dimensions. Because of the independency of these one-dimensional solutions, this algorithm has the potential to be efficiently implemented on parallel machines. Unconditional stability of the proposed LUCI scheme was verified.

Two test cases were run on a single processor using the LUCI scheme. These runs confirmed the symmetry-preserving property as well as the accuracy of the LUCI scheme. Comparisons with the SSOR computations and experimental data were very good.

Pseudoparallelism was employed to study the effects of time lagging of the boundary conditions at the blocks' interfaces on the convergence and stability of the LUCI and the SSOR schemes. These effects were simulated on a single processor for a range of processors that varied between 3 and 1850. It was found that for the specific test case and for the specific domain decomposition used the stability of the SSOR scheme broke down when the number of processors exceeded 1564. However, the LUCI scheme maintained its unconditional stability over the whole range. Convergence of both the LUCI and the SSOR schemes deteriorated as the number of processors increased. However, the deterioration in the SSOR case was more than that in the LUCI case.

References

[1] Beam, R. M., and Warming, R. F., "An Implicit Finite-Difference Algorithms for Hyperbolic Systems in Conservation-Law Form," *Journal of Computational Physics*, Vol. 22, No. 1, 1976, pp. 87–110.

[2] Warming, R. F., and Beam, R. M., "On the Construction and Application of Implicit Factored Schemes for Conservation Laws," *Proceedings of the Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics*, edited by H. B. Keller, American Mathematical Society, Providence, RI, 1978, pp. 85–129.

[3]Jameson, A., and Yoon, S., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, 1987, pp. 929–935.

[4]Pulliam, T. H., and Steger, J. L., "Implicit Finite Difference Simulations of Three-Dimensional Compressible Flow," *AIAA Journal*, Vol. 18, No. 2, 1980, pp. 159–167.

[5]Pulliam, T. H., and Steger, J. L., "Recent Improvements in Efficiency, Accuracy, and Convergence for Implicit Approximate Factorization Algorithms," AIAA Paper 85-0360, Jan. 1985.

[6]Jameson, A., and Turkel, E., "Implicit Schemes and LU Decompositions," *Mathematics of Computation*, Vol. 37, No. 156, 1981, pp. 385–397.

[7]Obayashi, S., and Kuwakara, K., "LU Factorization of an Implicit Scheme for the Compressible Navier–Stokes Equations," AIAA Paper 84-1670, June 1984.

[8]Abdallah, S., "Parallel Processing for Implicit Solutions of the Compressible Navier–Stokes Equations," *AIAA Journal*, Vol. 32, No. 12, 1994, pp. 2469–2471.

[9]Ghizawi, N. A., Abdallah, S., and Lee, T., "Cycle-Independent Implicit Algorithm for the Navier–Stokes Equations," *Proceedings of the ASME International Congress on Fluid Dynamics and Propulsion* (Cairo, Egypt), American Society of Mechanical Engineers and Cairo University, Vol. 2, 1996, pp. 322–331.

[10]Ghizawi, N. A., and Abdallah, S., "An Update on an Implicit Parallel Processing Algorithm for the Compressible Navier–Stokes Equations," *Proceedings of SES 1995, 32nd Annual Technical Meeting* (New Orleans, LA), Society of Engineering Science, College of Engineering, Univ. of New Orleans, and School of Engineering, Tulane Univ., 1995, pp. 1, 2.

[11]Ghizawi, N. A., and Abdallah, S., "Implicit Parallel Processing Algorithm for the Compressible Navier–Stokes Equations," *Proceedings of the 6th International Symposium on Computational Fluid Dynamics* (Lake Tahoe, NV), Univ. of California, Davis, CA, 1995, pp. 35–40.

[12]Anderson, J. D., *Computational Fluid Dynamics: The Basics with Applications*, McGraw–Hill, New York, 1995, pp. 70–83.

[13]Hakkinen, R. J., Greber, I., Trilling, L., and Arbarbanel, S. S., "The Interaction of an Oblique Shock Wave with a Laminar Boundary Layer," NASA Memorandum 2-18-59W, March 1959.

[14]Osher, S., "Numerical Solution of Singular Perturbation Problems and Hyperbolic Systems of Conservation Laws," *North Holland Mathematical Studies No. 47*, edited by S. Axelsson, L. S. Fraud, and A. Van der Sluis, North-Holland, Amsterdam, 1981, pp. 179–205.

[15]Gatlin, B., "An Implicit, Upwind Method for Obtaining Symbiotic Solutions to the Thin-Layer Navier–Stokes Equations," Ph.D. Dissertation, Dept. of Mechanical and Nuclear Engineering, Mississippi State Univ., Mississippi State, MS, Aug. 1987.

[16]Roe, P. L., "The Use of the Riemann Problem in Finite Difference Schemes," *Seventh International Conference on Numerical Methods in Fluid Dynamics*, Springer–Verlag, Berlin, 1981, pp. 354–359.

[17]Harten, A., and Lax, P. D., "A Random Choice Finite Difference Scheme for Hyperbolic Conservation Laws," *SIAM Journal of Numerical Analysis*, Vol. 18, No. 2, 1981, pp. 289–315.

[18]Van Leer, B., Thomas, J. L., Roe, P. L., and Newsome, R. W., "A Comparison of Numerical Flux Formulas for the Euler and Navier–Stokes Equations," AIAA Paper 87-1104, June 1987.

[19]Van Leer, B., "Flux-Vector Splitting for the Euler Equations," Vol. 170, Lecture Notes in Physics, Springer–Verlag, Berlin, 1982, pp. 507–512; also *Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics*, Springer–Verlag, Aachen, West Germany, 1982.

[20]Whitfield, D. L., Janus, J. M., and Simpson, L. R., "Implicit Finite Volume High Resolution Wave-Split Scheme for Solving the Unsteady Three-Dimensional Euler and Navier–Stokes Equations on Stationary or Dynamic Grids," Mississippi State Univ., Rept. MSSU-EIRS-ASE-88-2, Mississippi State, MS, Feb. 1988.

[21]Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods," *Journal of Computational Physics*, Vol. 40, No. 2, 1981, pp. 263–293.

[22]Roe, P. L., "Approximate Reiman Solvers, Parameter Vector, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[23]Tsai, Y.-L. P., and Hsieh, K. C., "Comparative Study of Computational Efficiency of Two LU Schemes for Non-Equilibrium Reacting Flows," AIAA Paper 90-0396, Jan. 1990.

[24]Ghizawi, N., "Lower Upper Cycle Independent Implicit Parallel Processing Algorithm for the Compressible Navier–Stokes Equations," Ph.D. Dissertation, Dept. of Aerospace Engineering and Engineering Mechanics, Univ. of Cincinnati, Cincinnati, OH, Sept. 1997.

[25]Beam, R. M., and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier–Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393–402.

[26]Chakravarthy, S. R., Szema, K.-Y., Goldberg, U. C., Gorski, J. J., and Osher, S., "Application of a New Class of High Accuracy TVD Schemes to the Navier–Stokes Equations," AIAA Paper 85-0165, Jan. 1985.

[27]Thomas, J. L., and Walters, R. W., "Upwind Relaxation for the Navier–Stokes Equations," *AIAA Journal*, Vol. 25, No. 4, 1987, pp. 527–534.

[28]Baker, L., and Smith, B. J., *Parallel Programming*, McGraw–Hill, New York, 1996, Chap. 4.

[29]Clarke, D. K., Salas, M. D., and Hassan, H. A., "Euler Calculations for Multielement Airfoils Using Cartesian Grids," *AIAA Journal*, Vol. 24, No. 3, 1986, pp. 353–358.

K. Kailasanath
Associate Editor