AIAA JOURNAL
Vol. 40, No. 5, May 2002

Parallelization of Rotorcraft Aerodynamics Navier–Stokes Codes

Kivanc Ekici* and Anastasios S. Lyrintzis†
Purdue University, West Lafayette, Indiana 47907-1282

The modification of unsteady three-dimensional Navier–Stokes codes for application on massively parallel and distributed computing environments is investigated. Previously, the Euler mode of the Navier–Stokes code TURNS has been parallelized. For the efficient implementation of the Navier–Stokes mode of TURNS on parallel computing systems, several algorithmic changes should be made. The main modification is done on the implicit operator, lower–upper symmetric Gauss–Seidel. Two new implicit operators are used because of convergence problems of traditional operators with high cell aspect ratio grids needed for viscous calculations. Results for Navier–Stokes cases are presented for various operators. The message passing interface protocol is used because of its portability to various parallel architectures.

Nomenclature

a	=	speed of sound
e	=	total energy
p	=	pressure
Pr	=	Prandtl number
Re	=	Reynolds number
u, v, w	=	Cartesian velocities
x, y, z	=	Cartesian coordinates
γ	=	specific heat ratio
μ	=	dynamic viscosity
ξ, η, ζ	=	generalized coordinates
ρ	=	density
Ψ	=	rotor azimuth angle

Introduction

ROTORCRAFT aerodynamics are very complex because three-dimensional unsteady transonic viscous aerodynamic phenomena are involved. The ability to predict the flow around helicopter rotors is vital for the control of high-speed losses, vibration, and noise. The advent of tilt-rotor aircraft in the past few years further complicates rotor aerodynamics.

Computational fluid dynamics (CFD) has proven to be an effective tool for prediction of rotorcraft aerodynamics and noise. Traditional CFD methods used for this purpose have included transonic small disturbance potential models and full-potential models.[1,2] Whereas potential methods can generate a good solution at relatively low cost, they are unable to resolve some flow features that can have a large impact on the aerodynamic and aeroacoustic properties. For instance, the vortical wake produced by the rotor blades plays a dominant role in unsteady load fluctuation, affecting aerodynamic performance. Also, shocks produced by transonic flow near the blade tip, often encountered in forward flight, have a large effect on the high-speed impulsive noise generated by the blades and consequently must be determined accurately. For these reasons, many more recent design analyses[3] use Euler/Navier–Stokes methods.

However, Navier–Stokes methods tend to be computationally demanding, which has delayed their widespread industrial use. Parallel computation offers the potential for cheaper and faster computations. Several vendors have released machines that utilize commodity reduced instruction set computer (RISC) processors that are the same as those used in high-end workstations, for example, IBM SP2 and SGI Origin. These machines generally attain better price/performance than vector processors and have peak execution rates that are several times faster than vector parallel machines such as the Cray C-90. It is also possible to assemble a collection of off-the-shelf hardware components and software packages and achieve high performance at very low prices.[4,5] Several companies are beginning to utilize parallel processing in the form of networked workstations, which sit idle during off hours, to attain supercomputer performance.[6] Unfortunately, many traditional CFD algorithms were developed with serial or vectorized computation in mind, and consequently require significant modification for efficient parallel execution. Also, a certain amount of code rewriting, for example, adding message passing calls, is necessary, which can be a significant time investment for long and well-established codes. These factors have been deterrents to more widespread industrial adoption of parallel processing.

The current study focuses on efforts to improve the efficiency of the TURNS code developed by Srinivasan et al.[7] and Srinivasan and Baeder.[8] TURNS is chosen because it provides a well-known industry standard for rotorcraft aerodynamics. The code uses a third-order accurate, upwind biased scheme that provides shock capturing. Flux limiters are applied so that the scheme is total variation diminishing.

In TURNS the main bottleneck for the parallel optimization is the lower–upper symmetric Gauss–Seidel (LU-SGS) implicit operator, currently used for both steady and unsteady calculations with the code. Candler et al.[9] developed an efficient algorithm called data parallel lower–upper relaxation (DP-LUR) to parallelize LU-SGS for a data parallel environment. Wissink et al.[10–12] developed a new hybrid domain decomposition implementation of the LU-SGS and DP-LUR operators and applied it to the Euler mode of TURNS successfully. This method can be extended to the Navier–Stokes equations. However, the LU-SGS algorithm exhibits poor convergence for the high-cell-aspect-ratio (CAR) grids needed to resolve the boundary layer of high Reynolds flows, as shown, for example, in Refs. 13 and 14. Wright et al.[15] proposed a full matrix modification of the LU relaxation method that works well for the high-aspect-ratio grids. In addition, Wright et al.[16] also developed another type of full matrix method, data parallel line relaxation (DPLR), which is even better for the high-aspect-ratio grids. Neither of these methods have been tested for three-dimensional unsteady flows.

The focus of this study is to develop a new hybrid method similar to that of Wissink using the full matrix approach and to apply this new algorithm together with DPLR to TURNS for the efficient parallelization of rotorcraft aerodynamics codes. Note that the LU-SGS algorithm has been used in a number of well-known CFD codes, for example, INS3D[17] and OVERFLOW,[18] primarily for its stability properties with larger time steps. Therefore, our proposed parallelization methodology will be applicable to several Navier–Stokes

Received 28 February 2001; revision received 24 September 2001; accepted for publication 8 October 2001.

*Research Assistant, School of Aeronautics and Astronautics; currently Research Associate, Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708-0300. Student Member AIAA.

†Associate Professor, School of Aeronautics and Astronautics. Associate Fellow AIAA.

codes. Message passing interface (MPI) will be used because of its portability to different parallel architectures.

Governing Equations

The nondimensional conservation law form of the three-dimensional, thin-layer Navier–Stokes equations in curvilinear space ξ, η, ζ, and τ can be written as[19,20]

$$\partial_\tau q + \partial_\xi E + \partial_\eta F + \partial_\zeta G = Re^{-1}\partial_\zeta S \tag{1}$$

where

$$q = \frac{1}{J}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ e\end{bmatrix}, \qquad E = \frac{1}{J}\begin{bmatrix}\rho U\\ \rho uU + \xi_x p\\ \rho vU + \xi_y p\\ \rho wU + \xi_z p\\ U(e+p) - \xi_t p\end{bmatrix}$$

$$F = \frac{1}{J}\begin{bmatrix}\rho V\\ \rho uV + \eta_x p\\ \rho vV + \eta_y p\\ \rho wV + \eta_z p\\ V(e+p) - \eta_t p\end{bmatrix}, \qquad G = \frac{1}{J}\begin{bmatrix}\rho W\\ \rho uW + \zeta_x p\\ \rho vW + \zeta_y p\\ \rho wW + \zeta_z p\\ W(e+p) - \zeta_t p\end{bmatrix} \tag{2}$$

The contravariant velocities in the preceding equations, namely, U, V, and W, are defined as

$$U = \xi_t + \xi_x u + \xi_y v + \xi_z w$$

$$V = \eta_t + \eta_x u + \eta_y v + \eta_z w$$

$$W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \tag{3}$$

The quantities such as ξ_x, ξ_y, and ξ_z are the metrics of the coordinate transformation, and J is the Jacobian of the transformation. The viscous flux vector for the thin layer approximation S is given by

$$S = \frac{\mu}{J}\begin{bmatrix}0\\ \left(\zeta_x^2+\zeta_y^2+\zeta_z^2\right)u_\zeta + 1/3(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_x\\ \left(\zeta_x^2+\zeta_y^2+\zeta_z^2\right)v_\zeta + 1/3(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_y\\ \left(\zeta_x^2+\zeta_y^2+\zeta_z^2\right)w_\zeta + 1/3(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_z\\ \left(\zeta_x^2+\zeta_y^2+\zeta_z^2\right)m + 1/3(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)n\end{bmatrix} \tag{4}$$

where

$$m = 0.5(u^2+v^2+w^2)_\zeta + [1/Pr(\gamma-1)](a^2)_\zeta$$

$$n = (\zeta_x u + \zeta_y v + \zeta_z w)$$

For the turbulent viscous flows, the viscosity coefficient μ is computed as the sum of the laminar viscosity coefficient μ_l estimated using Sutherland's law and the turbulent viscosity μ_t estimated using the Baldwin–Lomax algebraic turbulence model. The pressure can be obtained from the equation of state given as

$$p = (\gamma-1)\{e - (\rho/2)(u^2+v^2+w^2)\} \tag{5}$$

Numerical Method

The first-order implicit Euler integration in time gives the following form of the equations:

$$\left[I + \Delta\tau\left(\delta_\xi A^n + \delta_\eta B^n + \delta_\zeta C^n - Re^{-1}\delta_\zeta^2 M^n\right)\right]\Delta q^n = -\Delta\tau f(q^n) \tag{6}$$

where

$$f(q^n) = \left(\delta_\xi E^n + \delta_\eta F^n + \delta_\zeta G^n - Re^{-1}\delta_\zeta S^n\right) \tag{7}$$

is the flux evaluated using the upwind-biased scheme of Roe[21] at time level n; A, B, and C are the inviscid Jacobians given by Pulliam and Steger[20]; and M is the viscous Jacobian given by Tysinger and Caughey.[22] The monotone upstream-centered scheme of Anderson et al.[23] for the conservative laws (MUSCL) is used to get high-order accuracy with flux limiters. First, a flux vector splitting is applied to the flux Jacobians A, B, and C, of the inviscid flux vectors E, F, and G. In the ξ direction, A is split into its positive and negative eigenvalue parts, $A = A^+ + A^-$, and the positive matrix is backward differenced, whereas the negative matrix is forward differenced. The same technique is applied to the other inviscid Jacobians in the corresponding directions. The viscous Jacobian M, on the other hand, is simply central differenced in ζ direction. When the backward differenced + terms in L and the forward differenced − terms in U are collected, the system can be divided into lower and upper block-tridiagonal factors with a diagonal factor given by

$$D = I + \Delta\tau(A^+ - A^- + B^+ - B^- + C^+ - C^- + 2M)_{j,k,l}$$

$$L = D - \Delta\tau\left(A^+_{j-1} + B^+_{k-1} + C^+_{l-1} + M_{l-1}\right)$$

$$U = D + \Delta\tau\left(A^-_{j+1} + B^-_{k+1} + C^-_{l+1} - M_{l+1}\right) \tag{8}$$

Second, to avoid costly computation times for the inversion of the 5×5 D matrix, an efficient spectral radius approximation proposed by Yoon and Kwak[14] is applied to the flux Jacobians:

$$A^\pm = \tfrac{1}{2}(A \pm \rho_A I) \tag{9}$$

where ρ_A is the spectral radius of A in the ξ direction. This approximation results in a very simple form of D:

$$D = I + \Delta\tau(\rho_A I + \rho_B I + \rho_C I + 2\rho_M I)_{j,k,l} \tag{10}$$

for which the inversion is just a scalar operation.

The LU-SGS scheme proposed by Yoon and Jameson[24] is used for the time integration of the implicit equations. The resulting system of equations is solved by applying a two-step symmetric Gauss–Seidel algorithm as follows:

$$L\Delta q^* = -\Delta\tau f(q^n), \qquad U\Delta q^n = D\Delta q^* \tag{11}$$

In Eq. (11), Δq^* is the intermediate solution and Δq^n is the update to the vector of dependent variables at time step n, $q^{n+1} = q^n + \Delta q^n$. Note that the approximations are made only to the left-hand side of the system to improve the convergence to the steady-state solution; hence, they will not effect the accuracy of the scheme.

Time Accuracy

The implicit algorithm implemented in TURNS is used for steady-state as well as time-accurate solutions. For steady-state problems, a first-order scheme in time is used, and the convergence to the steady state is accelerated using large time steps. On the other hand, for time-accurate problems, the equations are integrated through time using a second-order scheme starting from a meaningful initial condition, usually the steady-state solution. In this case, the time step is chosen according to the timescale of the problem.

LU-SGS has some factorization error associated with the approximations made on the left-hand side of the algorithm. This destroys the good convergence characteristics in large time steps for steady-state solutions and puts a limit to the size of the time steps used. In addition, for time-accurate solutions, the factorization error still exists, but it can be reduced by using inner relaxation steps.

For steady-state problems one can vary $\Delta\tau$ in space, which can be interpreted as an attempt to use a uniform Courant–Friedrichs–Lewy number throughout the flowfield. The space-varying time step approach is especially effective for grids that have very fine to very coarse spacings in the domain. The Navier–Stokes grids used in this study are good examples of such wide variety length scale grids. In TURNS, a purely geometric variation of the time-step approach is used[25,26] in the following form:

$$\Delta\tau = \Delta\tau|_{\text{ref}}\frac{1+\varepsilon\sqrt{J}}{1+\sqrt{J}} \tag{12}$$

to accelerate the steady-state convergence. In Eq. (12), $\Delta\tau|_{\text{ref}}$ is the fixed reference time step, ε is a small constant, for example, 0.002,

and J is the Jacobian of the transformation. On the other hand, for unsteady solutions, the time step used in the calculations is a fixed value.

Boundary Conditions

The boundary conditions are applied explicitly, which means that the values of the conserved variables are updated at the end of each iteration. The Cartesian velocities u, v, and w are calculated from the extrapolation of the contravariant velocities. On the surface, the tangency condition is imposed for the Euler calculations, whereas a no-slip condition is imposed for the Navier–Stokes calculations. These two conditions require the contravariant velocities $W = 0$ and $U = V = W = 0$, respectively. The density and the pressure on the surface are calculated by zeroth-order extrapolations, whereas the total energy is determined from the equation of state.

Parallelization

Previously, the parallelization of the Euler mode of TURNS was done by Wissink[11] in a domain decomposition fashion using MPI routines. The parallelization of the code in multiple instruction multiple data (MIMD) fashion was chosen for two main reasons. First, code rewriting from FORTRAN 77 to FORTRAN 90, which is required by single instruction multiple data, is avoided. Second, the MIMD approach makes the program more portable to different kinds of parallel architectures.

A simple domain decomposition strategy, which uses the values from the previous iteration of the current time step on processor boundaries, is chosen. This strategy was used before by Wissink et al.,[27] and no significant degradation of performance was observed for up to 456 processors. The alternative, of course, is to do a formal domain decomposition as explained in Ref. 28. However, this approach is time consuming to implement in a legacy code, and because the first, simple strategy has worked well previously, it was used in this study.

The computational grid is decomposed to equally spaced partitions to prevent load imbalance, and a copy of the code is run on each processor. On the boundaries of each partition, an overlapping strategy is used to calculate the conserved variables. For example, the conserved variables on the left boundaries of each partition is calculated as the last interior points near the right boundary of the previous partition. Likewise, the conserved variables on the right boundaries of each partition is calculated as the first interior points near the left boundary of the next partition. This single overlap strategy and a typical two-dimensional grid partitioning are given in Figs. 1 and 2, respectively.

After all of the processors are done with their solutions, the values on the boundaries are sent to and received from the neighboring processors using MPI_SEND and MPI_RECEIVE subroutines, and the calculations for the next iteration can be started. For small number of points per processor, we expect to see some performance degradation because the boundary values are less accurate.

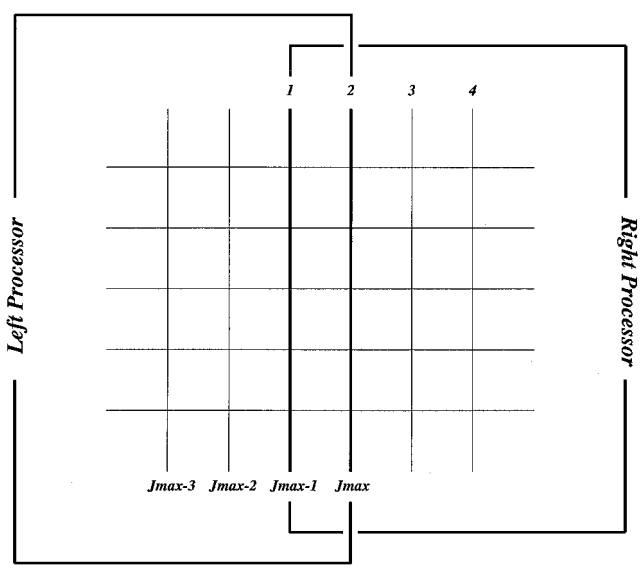

Fig. 1 Single overlap strategy.

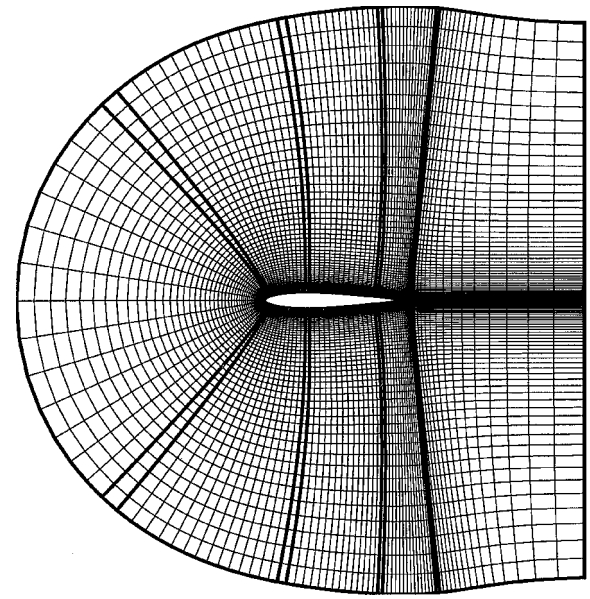

Fig. 2 Partitions of a typical two-dimensional grid.

The partitioning of the grid is allowed only in the streamwise and radial directions, as shown in Fig. 3, to eliminate the load imbalance when applying the boundary conditions.

In addition, the grid collapses to a singular plane at outboard the tip of the blade and across the wake. At these locations, to satisfy the continuity of the flow quantities, values are interpolated between the processors on each side of the singular plane. Partitioning in the normal direction would make some of the processors sit idle while the others would perform the interpolation, which would cause load imbalance. Therefore, partitioning of the computational domain is done in the streamwise and radial directions only.

Diagonal Operators

In this section, two parallel implicit operators that use the spectral approximation of LU-SGS scheme previously implemented in TURNS are described. The use of spectral approximation retains the ease of computation, as well as the advantage of using large time steps for the steady-state solution.

LU-SGS Operator

The LU-SGS algorithm is the basis of the parallel implicit operators implemented in TURNS. In the two-step symmetric Gauss–Seidel method (see Ref. 11), there are off-diagonal terms coming from L and U on the left-hand side of both equations. If we move these off-diagonal terms to the right-hand side, the resulting system becomes diagonal, and the updates of the conserved quantities can easily be obtained. The algorithm for the LU-SGS implicit operator can be written as follows.

Algorithm 1(LU-SGS):

Do $j, k, l = 1, \ldots, J_{\max}, K_{\max}, L_{\max}$

$$\Delta q^*_{j,k,l} = D^{-1}\Delta\tau\left[-f(q^n) + A^+_{j-1}\Delta q^*_{j-1} + B^+_{k-1}\Delta q^*_{k-1} + C^+_{l-1}\Delta q^*_{l-1} + M_{l-1}\Delta q^*_{l-1}\right]$$

EndDo

Do $j, k, l = J_{\max}, K_{\max}, L_{\max}, \ldots, 1$

$$\Delta q^n_{j,k,l} = \Delta q^*_{j,k,l} - D^{-1}\Delta\tau\left[A^-_{j+1}\Delta q^n_{j+1} + B^-_{k+1}\Delta q^n_{k+1} + C^-_{l+1}\Delta q^n_{l+1} - M_{l+1}\Delta q^n_{l+1}\right]$$

EndDo

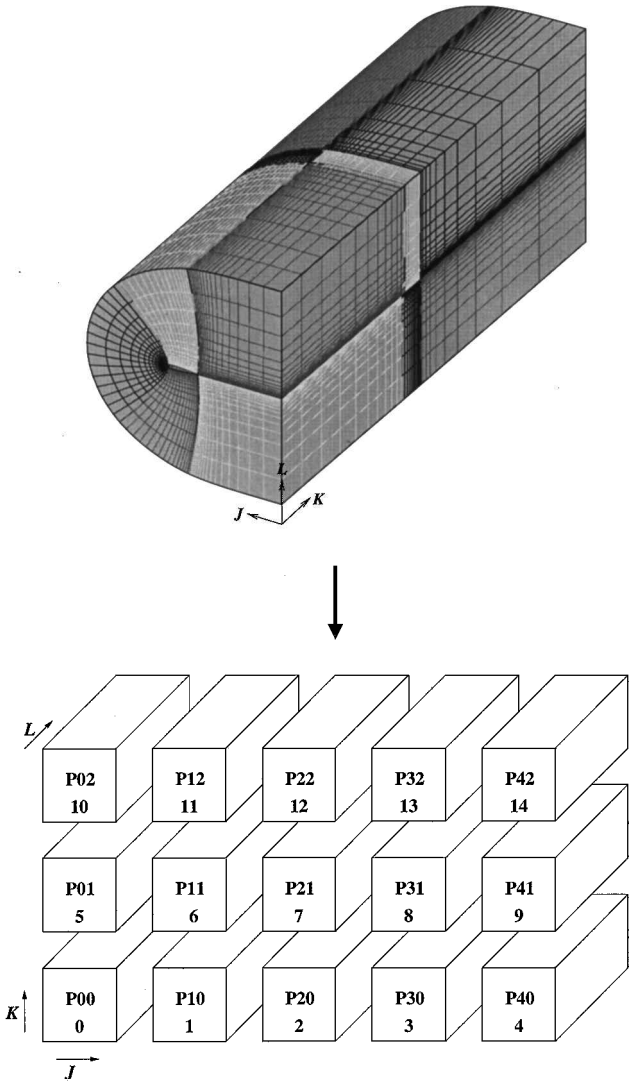

Fig. 3 Partitioning strategy of the computational grid.

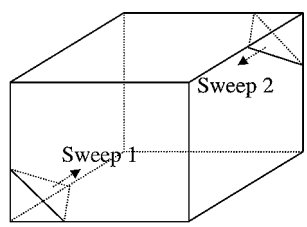

Fig. 4 Schematic diagram of LU-SGS sweeps.

In the first step of the solution, the intermediate solution vector Δq^* is obtained by sweeps that start form the lower left corner and proceed to the upper right corner of the computational grid. The final solution is then obtained by sweeps in the reverse direction. The direction of sweeps in the computational domain is shown in Fig. 4. The LU-SGS method uses hyperplanes of different sizes throughout the grid. Although this approach can easily be vectorized, it is difficult to implement the method on the distributed-memory computers because of the load balancing problems caused by the variable-size hyperplanes and the large amount of communication due to the recursion between planes. For example, if we look at the forward sweep, the point (j, k, l) can begin the computation only when points $(j-1, k, l)$, $(j, k-1, l)$, and $(j, k, l-1)$ have completed the step. Therefore, this data dependency causes load balancing problem, and not all of the processors can be kept busy at the same time. Overall, the LU-SGS algorithm in its original form is not very suitable for distributed-memory parallelization, and hence, some algorithmic changes should be made.

DP-LUR Operator

A very efficient point-relaxation implementation of the LU-SGS method for the data-parallel environment, DP-LUR, has been introduced by Candler et al.[9] for solving hypersonic flow problems. Basically, this method replaces the two symmetric Gauss–Seidel sweeps with Jacobi sweeps and moves all of the off-diagonal terms to the right-hand side. The procedure for this method can be written as follows.

Algorithm 2 (DP-LUR):

$$\Delta q^{(0)}_{j,k,l} = -D^{-1} \cdot \Delta\tau f(q^n)$$

Do $i = 1, \ldots, i_{\text{sweep}}$

Do $j, k, l = 1, \ldots, J_{\max_{\text{local}}}, K_{\max_{\text{local}}}, L_{\max_{\text{local}}}$

$$\Delta q^{(i)}_{j,k,l} = D^{-1}\Delta\tau\left[-f(q^n) + A^+_{j-1}\Delta q^{(i-1)}_{j-1} + B^+_{k-1}\Delta q^{(i-1)}_{k-1} + C^+_{l-1}\Delta q^{(i-1)}_{l-1} + M_{l-1}\Delta q^{(i-1)}_{l-1} - A^-_{j+1}\Delta q^{(i-1)}_{j+1} - B^-_{k+1}\Delta q^{(i-1)}_{k+1} - C^-_{l+1}\Delta q^{(i-1)}_{l+1} + M_{l+1}\Delta q^{(i-1)}_{l+1}\right]$$

EndDo

EndDo

$$\Delta q^n_{j,k,l} = \Delta q^{(i_{\text{sweep}})}$$

DP-LUR uses only the nearest neighbor data due to the point-relaxation strategy and, therefore, allows simultaneous computations on multiple processors with each processor holding the local data. The data that lie on the borders are communicated after each domain sweep. This algorithm is implemented in TURNS in such a way that the computations are completely load balanced with communications occurring only at the borders of each partition. In addition, because the algorithm is matrix free, it is memory efficient, which allows us to solve large-size problems.

Although DP-LUR is more suitable for parallel computing, it may need more computations to converge, compared to the LU-SGS method. The main reason for this is that a point-relaxation method usually has a lower rate of convergence than a Gauss–Seidel method.

Hybrid LU-SGS Operator

To reduce the computations needed for DP-LUR, Wissink et al.[10] and Wissink[11] have introduced a new algorithm that combines the high convergence rate of LU-SGS with the efficient interprocessor communication of DP-LUR. This method, called the hybrid LU-SGS method, performs LU-SGS solutions locally on each processor and uses the point-relaxation sweeping of DP-LUR to communicate the border data. The procedure is given as follows.

Algorithm 3 (Hybrid LU-SGS):

$$\Delta q^{(0)}_{j,k,l} = -D^{-1} \cdot \Delta\tau f(q^n)$$

Do $i = 1, \ldots, i_{\text{sweep}}$

Communicate border data $\Delta q^{(i-1)}_{j,k,l}$

On the borders $\Delta q^*_{j,k,l} = \Delta q^{(i-1)}_{j,k,l}$

Do $j, k, l = 1, \ldots, J_{\max_{\text{local}}}, K_{\max_{\text{local}}}, L_{\max_{\text{local}}}$

$$\Delta q^*_{j,k,l} = D^{-1}\Delta\tau\left[-f(q^n) + A^+_{j-1}\Delta q^*_{j-1} + B^+_{k-1}\Delta q^*_{k-1} + C^+_{l-1}\Delta q^*_{l-1} + M_{l-1}\Delta q^*_{l-1}\right]$$

EndDo

Do $j, k, l = J_{\max_{\text{local}}}, K_{\max_{\text{local}}}, L_{\max_{\text{local}}}, \ldots, 1$

$$\Delta q^n_{j,k,l} = \Delta q^*_{j,k,l} - D^{-1}\Delta\tau\left[A^-_{j+1}\Delta q^n_{j+1} + B^-_{k+1}\Delta q^n_{k+1} + C^-_{l+1}\Delta q^n_{l+1} - M_{l+1}\Delta q^n_{l+1}\right]$$

EndDo

EndDo

$$\Delta q^n_{j,k,l} = \Delta q^{(i_{\text{sweep}})}$$

The preceding diagonal methods, that is, LU-SGS, DP-LUR, and hybrid LU-SGS, work well with the Euler mode of TURNS. However, when dealing with the high CAR grids, which are needed to resolve the boundary layer in viscous flows, they either show poor or no convergence. This is mainly due to the spectral approximation, which overstabilizes the solution by dramatically increasing the diagonal elements of D for high CAR grids.

Full Matrix Operators

To overcome the convergence problem due to the high CAR grids, Wright et al.[15,16] proposed a class of methods, which removes the spectral approximation and uses the exact forms of the Jacobian matrices. The Jacobian A can be split into positive and negative parts as follows:

$$A = X_A \Lambda_A X_A^{-1}$$

$$A = X_A\left(\Lambda_A^+ + \Lambda_A^-\right)X_A^{-1} = A^+ + A^- \tag{13}$$

where X_A and X_A^{-1} are the left- and right-eigenvector matrices and Λ_A is the diagonal eigenvalue matrix. Knowing the right- and left-eigenvector matrices and the eigenvalues of the inviscid flux Jacobians, the difference in the positive and negative parts can be written as

$$A^+ - A^- = X_A \Lambda_A^+ X_A^{-1} - X_A \Lambda_A^- X_A^{-1}$$

$$A^+ - A^- = X_A |\Lambda_A| X_A^{-1} \tag{14}$$

In addition, it is known that the sum of the positive and negative parts is equal to the Jacobian itself. This leads us to the following expressions:

$$D = I + \Delta\tau\left(X_A|\Lambda_A|X_A^{-1} + X_B|\Lambda_B|X_B^{-1} + X_C|\Lambda_C|X_C^{-1} + 2M\right)_{j,k,l}$$

$$A^+ = \left(A + X_A|\Lambda_A|X_A^{-1}\right)/2$$

$$A^- = \left(A - X_A|\Lambda_A|X_A^{-1}\right)/2 \tag{15}$$

The nondiagonal terms on the right-hand side also use the exact forms of the Jacobians for the full matrix methods. With the use of the true split Jacobians, the solution becomes more computationally and memory intensive than the diagonal versions. At each grid point, the 5×5 D matrix must be inverted and stored now. However, to decrease the memory needs, the off-diagonal Jacobians are computed on the fly and discarded at each inner sweep. Because all of this additional computation is local, the amount of communication used is exactly the same as the diagonal method. Although the amount of computation is greater than the diagonal method, the use of the exact forms of the Jacobians increases the overall convergence rate compared to the diagonal methods. The only significant problem with this approach is that when the spectral radius approximation is removed, smaller time steps must be used compared to the diagonal methods. However, experience has shown that the full-matrix approach has a much higher convergence rate, and the limitation on the time step is not important for the steady-state problems.

Full Matrix Hybrid (FM-Hybrid) Operator

The idea of using the advantage of DP-LUR communication and the LU-SGS convergence rate introduced by Wissink et al.[10] and Wissink[11] can further be extended by the use of true Jacobians, to enable handling of high CAR grids, in the LU-SGS algorithm. The resulting new method, which will be called the full matrix hybrid (FM-Hybrid) method, uses the full form of the Jacobians, solves the problem using LU-SGS locally, and uses DP-LUR for interprocessor communication. Because there is no change in the data communication between the processors, the new algorithm needs exactly the same amount of communication as the hybrid LU-SGS method. The amount of computations is greater than the diagonal method. However, the results show that this new method is better than all of the previous ones in terms of the wallclock time used on an IBM SP2 computer because the number of iterations needed is much lower. The algorithm for this new method can be written as follows.

Algorithm 4 (FM-Hybrid):

$$D^{-1} = \left(I + \Delta\tau\left(X_A|\Lambda_A|X_A^{-1} + X_B|\Lambda_B|X_B^{-1} + X_C|\Lambda_C|X_C^{-1} + 2M\right)_{j,k,l}\right)^{-1}$$

$$\Delta q_{j,k,l}^{(0)} = -D^{-1} \cdot \Delta\tau f(q^n)$$

Do $i = 1, \ldots, i_{\text{sweep}}$

Communicate border data $\Delta q_{j,k,l}^{(i-1)}$

On the borders $\Delta q_{j,k,l}^{*} = \Delta q_{j,k,l}^{(i-1)}$

Do $j, k, l = 1, \ldots, J_{\max_{\text{local}}}, K_{\max_{\text{local}}}, L_{\max_{\text{local}}}$

$$\Delta q_{j,k,l}^{*} = D^{-1}\Delta\tau\left[-f(q^n) + A_{j-1}^{+}\Delta q_{j-1}^{*} + B_{k-1}^{+}\Delta q_{k-1}^{*} + C_{l-1}^{+}\Delta q_{l-1}^{*} + M_{l-1}\Delta q_{l-1}^{*}\right]$$

EndDo

Do $j, k, l = J_{\max_{\text{local}}}, K_{\max_{\text{local}}}, L_{\max_{\text{local}}}, \ldots, 1$

$$\Delta q_{j,k,l}^{n} = \Delta q_{j,k,l}^{*} - D^{-1}\Delta\tau\left[A_{j+1}^{-}\Delta q_{j+1}^{n} + B_{k+1}^{-}\Delta q_{k+1}^{n} + C_{l+1}^{-}\Delta q_{l+1}^{n} - M_{l+1}\Delta q_{l+1}^{n}\right]$$

EndDo

EndDo

$$\Delta q_{j,k,l}^{n} = \Delta q^{(i_{\text{sweep}})}$$

DPLR Operator

Another method for the solution of Navier–Stokes equations with high CAR grids, proposed by Wright et al.,[16] is the last method to be applied to TURNS. This method, DPLR, is based on the Gauss–Seidel line relaxation (GSLR) method of MacCormack,[29] and it basically moves the off-diagonal terms in the body-normal direction ζ on the right-hand side back to the left-hand side and solves a block-tridiagonal system of equations. This makes the problem more physical and strongly coupled in ζ direction, where the viscous flow gradients are strongest. When started from Eqs. (11) and (8), the resulting tridiagonal system can be written as

$$\tilde{C}^{-}\Delta q_{l+1}^{n} + D\Delta q_{l}^{n} - \tilde{C}^{+}\Delta q_{l-1}^{n} = \Delta\tau\left[-f(q^n) + A_{j-1}^{+}\Delta q_{j-1}^{n} - A_{j+1}^{-}\Delta q_{j+1}^{n} + B_{k-1}^{+}\Delta q_{k-1}^{n} - B_{k+1}^{-}\Delta q_{k+1}^{n}\right] \tag{16}$$

where

$$\tilde{C}^{-} = \Delta\tau\left(C_{l+1}^{-} - M_{l+1}\right), \qquad \tilde{C}^{+} = \Delta\tau\left(C_{l-1}^{+} + M_{l-1}\right) \tag{17}$$

The DPLR method uses a series of line-relaxation steps similar to the point-relaxation stepping of DP-LUR, whereas the GSLR method uses Gauss–Seidel sweeps. By the use of line relaxation, which is similar to the point relaxation of the earlier methods, the data dependencies are minimized, and the code can easily be parallelized. First, the left-hand side of Eq. (16) is decomposed to its lower and upper matrices, and the resulting system is solved by ignoring the off-diagonal terms on the right-hand side. Then, a series of inner sweeps are performed on the system using the solution from the previous relaxation step. The algorithm can be written as follows.

Algorithm 5 (DPLR):

$$\tilde{C}^{-}\Delta q_{l+1}^{(0)} + D\Delta q_{l}^{(0)} - \tilde{C}^{+}\Delta q_{l-1}^{(0)} = -\Delta\tau f(q^n)$$

Do $i = 1, \ldots, i_{\text{sweep}}$

Communicate border data

$$\tilde{C}^{-}\Delta q_{l+1}^{(i)} + D\Delta q_{l}^{(i)} - \tilde{C}^{+}\Delta q_{l-1}^{(i)} = \Delta\tau\Big[-f(q^n) + A_{j-1}^{+}\Delta q_{j-1}^{(i-1)} - A_{j+1}^{-}\Delta q_{j+1}^{(i-1)} + B_{k-1}^{+}\Delta q_{k-1}^{(i-1)} - B_{k+1}^{-}\Delta q_{k+1}^{(i-1)}\Big]$$

EndDo

$$\Delta q_{j,k,l}^{n} = \Delta q^{(i_{\text{sweep}})}$$

With the way TURNS is parallelized now, all of the data in the body-normal direction ζ are local (Fig. 2). Therefore, there are no data dependency problems for the block-tridiagonal system, and the entire procedure can be performed simultaneously in parallel. Wright et al.[16] have successfully implemented this algorithm to Navier–Stokes equations and found that it is superior to the diagonal and full-matrix DP-LUR algorithms. In this study, we will implement DPLR in TURNS and compare its performance with the rest of the implicit operators, especially with the new FM-Hybrid operator introduced in the preceding section.

Results for Parallel Implicit Operators

To test and compare the convergence characteristics of the different methods, two-dimensional steady-state and three-dimensional quasi-steady and unsteady computations are carried out in TURNS. More detailed results can be found in Ref. 30. The two-dimensional calculations are performed on 27 processors, whereas the quasi-steady three-dimensional calculations are performed on 36, and the unsteady three-dimensional calculations are performed on 16 processors. The 80-node 272-processor IBM SP2 computer of the Purdue University is used for the runs.

Two-Dimensional Steady-State Results

In this section two-dimensional, Navier–Stokes solutions for two different airfoils are presented. First, solutions for a transonic flow around a Royal Aerospace Establishment (RAE) 2822 airfoil using different implicit operators are presented and compared with the experimental data. Second, the performance of the implicit operators is investigated for a transonic flow around a NACA 0012 airfoil for grids with different cell aspect ratios.

All of the calculations are performed on 137×50 C grids. The partitioning of the grid in the wraparound direction gives each processor 7 grid points in the wraparound direction and 50 grid points in the normal direction. Nondimensional spatially varying time steps $\Delta\tau$ are used for all methods to increase the convergence to steady state. The value of the reference time step $\Delta\tau|_{\text{ref}}$ used in Eq. (12) is different for all methods. The diagonal operators usually allow larger values compared to the full matrix methods. The runs are stopped when the global density residual reached $1E-11$.

RAE 2822 Airfoil ($CAR_{max} = 19{,}400$)

The accuracy of the serial and vectorized versions of the TURNS code has been previously validated in Refs. 7, 8, and 19. In this section, we will investigate the numerical accuracy of the parallelized version of the code using different implicit operators. For a steady-state case, one expects to get identical results for different implicit operators as long as the right-hand side of the system is kept the same.

The two-dimensional viscous transonic numerical results are compared with the experimental data of an RAE 2822 supercritical airfoil. The experimental data are presented by Cook et al.[31] and are extensively used in the literature for code validation purposes. The calculation here corresponds to case 6 with a freestream Mach number of 0.726, an angle of attack of 2.92 deg, and a Reynolds number of 6.5×10^6. Because of the wall effects in the wind tunnel, a correction is necessary in the angle of attack. In these calculations, the value of the numerical angle of attack is taken to be 2.43 deg. The outer boundary of the grid is located about eight chords from the body. The minimum spacing of the grid in the normal direction near the surface is 0.0001 chords, which corresponds to a maximum CAR of 19,400. A section of the grid is shown in Fig. 5.

The surface pressure distributions for well-converged solutions are presented in Fig. 6. The numerical solutions obtained using different implicit operators are identical. This means that the choice of the implicit method does not affect the accuracy. In addition, the numerical results are in good agreement with the experimental data except for the location of the shock. Although not shown here, plots of the pressure and the Mach number contours for the converged solution are also identical for all of the implicit operators.

NACA 0012 Airfoil ($CAR_{max} = 2200$)

The main objective of this study is the efficient parallelization of TURNS for the high CAR grids needed to resolve the boundary layer of the viscous flows. Therefore, only comparison for the performance of the methods will be carried out from this point on. However, note that the converged solutions obtained using different methods are identical and that the choice of the implicit operator does not change the accuracy of the code.

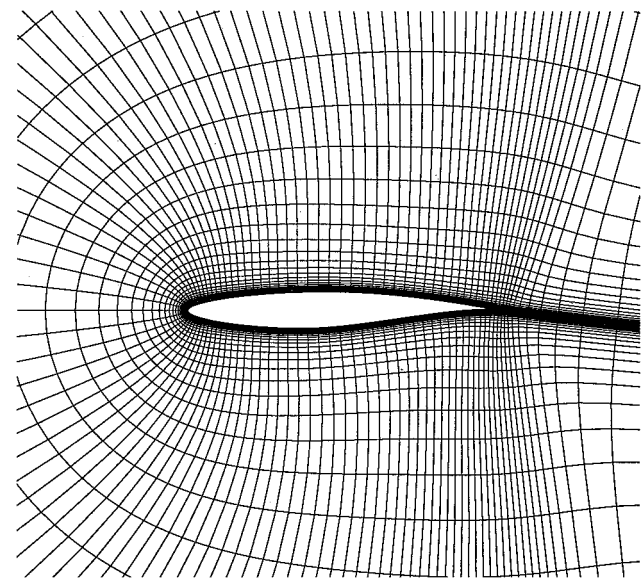

Fig. 5 Computational grid around RAE 2822 airfoil.

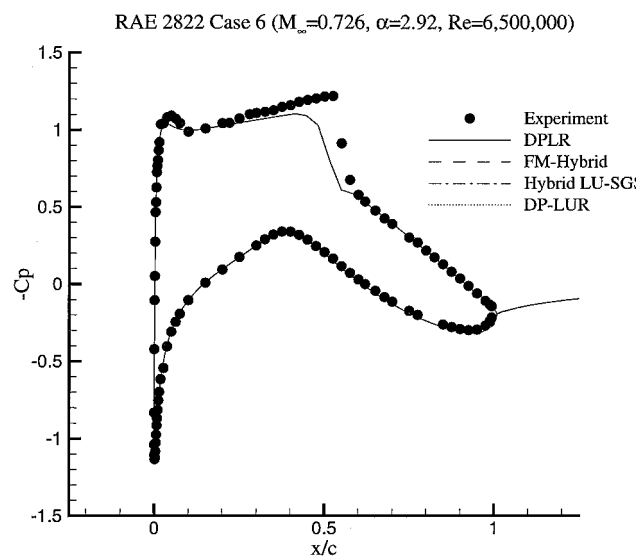

Fig. 6 Computed and experimental pressure distributions for RAE 2822 airfoil.

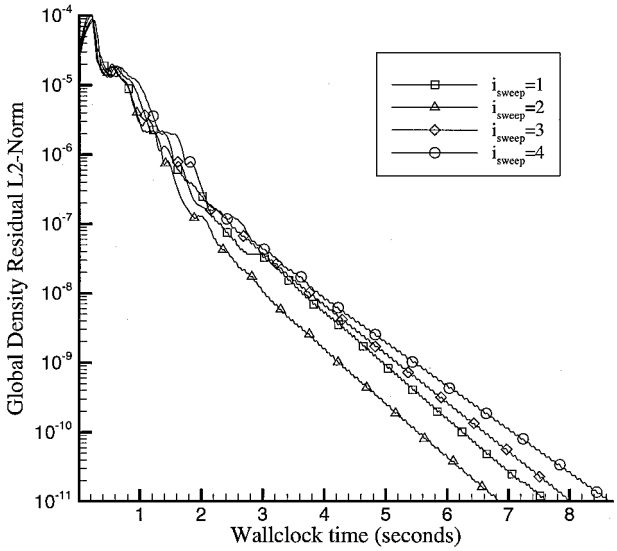

Fig. 7 Effect of number of inner sweeps on the wallclock time for FM-Hybrid method.

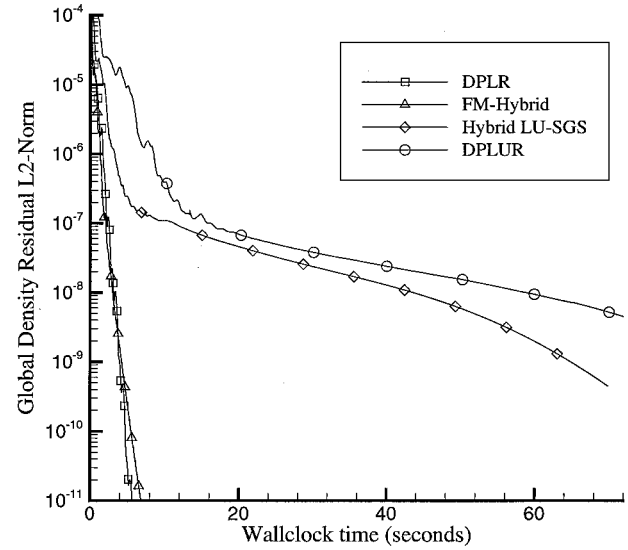

Fig. 8 Comparison of parallel implicit operators for optimal number of inner sweeps (NACA 0012, $CAR_{max} = 2200$).

The case tested here is a two-dimensional, viscous, transonic flow around a NACA 0012 airfoil with a freestream Mach number of 0.80, a 0-deg angle of attack, and a Reynolds number of 2.75×10^6. The maximum CAR is 2200 in the computational domain. The calculations are carried out for different reference time steps for each method, and the values corresponding to the maximum allowable time steps are used.

The effect of the number of inner sweeps is studied for different implicit operators. In Fig. 7, the effect of the number of inner sweeps on convergence is shown for the FM-Hybrid method. For this case, the method converges for all sweeps, and $i_{sweep} = 2$ has the best convergence for the wallclock time. Similarly, computations for the other methods are also performed for different number of inner sweeps, and i_{sweep} with the best convergence is chosen for each method ($i_{sweep} = 4$ for DP-LUR, $i_{sweep} = 1$ for hybrid LU-SGS, and $i_{sweep} = 3$ for DPLR).

When the optimal number of inner sweeps for each of the implicit operators is found, the performance of the implicit methods can be compared with each other (Fig. 8). The full matrix methods, DPLR and FM-Hybrid, have significantly better convergence compared to the diagonal methods, hybrid LU-SGS and DP-LUR. The convergence criterion is met in 5.62 and 6.86 s for DPLR and FM-Hybrid, and the global density residual is down to $4.47E-10$ and $4.52E-09$ for hybrid LU-SGS and DP-LUR in approximately 72 s.

The selection of the optimal number of inner sweeps for each method by experimentation is not very practical because it requires multiple runs of a case for the comparison of the best case performance. Although, the number of inner sweeps for each method is problem dependent, our experience has shown that the optimal value of the sweeps are three to four for DPLR, one to two for FM-Hybrid, one for hybrid LU-SGS, and four for DP-LUR for most of the problems. In a similar study, Wright[32] used four inner sweeps for DP-LUR and DPLR methods because of cost effectiveness and improved stability. In another study by Wissink,[11] one inner sweep usually showed slightly better performance than two inner sweeps for the hybrid LU-SGS method. Therefore, for the rest of this study four inner sweeps will be used for DP-LUR and DPLR methods, and one inner sweep will be used for hybrid LU-SGS and FM-Hybrid methods. Figure 9 compares the performance of the parallel implicit operators using these typical values of the inner sweeps, and it is apparent that there is no significant difference when optimal values of inner sweeps are used.

NACA 0012 Airfoil ($CAR_{max} = 22000$)

The final two-dimensional case tested is identical to the preceding one except that the grid clustering is increased near the body. This is done to investigate how well DPLR and FM-Hybrid perform

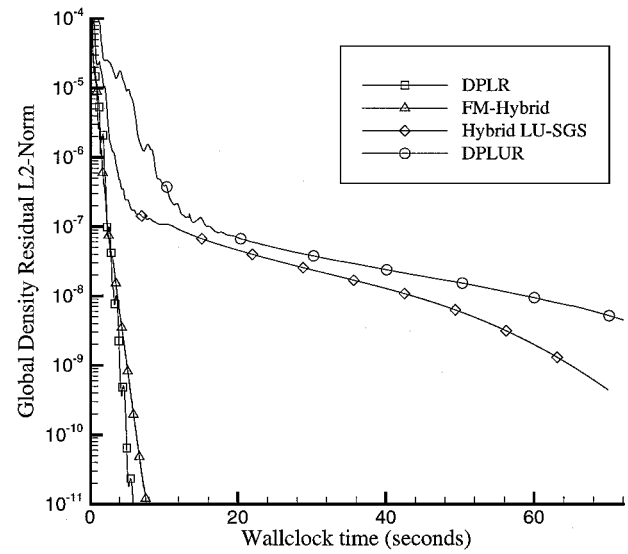

Fig. 9 Comparison of parallel implicit operators for typical number of inner sweeps (NACA 0012, $CAR_{max} = 2200$).

compared to the diagonal methods for very high CAR grids. The number of grid points in wraparound and normal directions are 137×50, as in the preceding cases. The computational grid used for this case has a maximum CAR of 22,000.

Figure 10 shows the wallclock time vs global density residual for the implicit operators. This time, the convergence criterion is met for DPLR and FM-Hybrid methods in 8.49 and 10.50 s, whereas the density residual goes down to $1.19E-08$ in 70.17 s for hybrid LU-SGS and $2.50E-08$ in 72.00 s for DP-LUR.

Thus, the increase in the CAR of the grid does not effect the convergence rate of the full matrix methods, whereas it slows the convergence rate of the diagonal methods. The DPLR and FM-Hybrid methods show comparable performance, with DPLR performing slightly better and both outperforming the diagonal methods. This conclusion does not change, according to a previous study,[33] where the same $\Delta\tau$ was used for all methods.

Three-Dimensional Viscous Quasi-Steady Results

For the three-dimensional quasi-steady cases, the operational load survey (OLS) rotor blade of an AH-1 helicopter is selected. The OLS rotor has an aspect ratio of 9.22 and a maximum thickness ratio of 9.71% chord. The test case examined has a tip Mach number of 0.664 and an advance ratio of 0.258 with a Reynolds number of

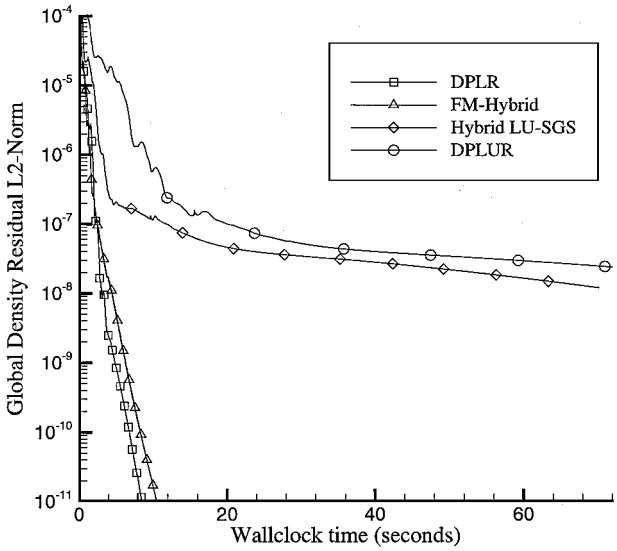

Fig. 10 Comparison of parallel implicit operators (NACA 0012, $CAR_{max} = 22{,}000$).

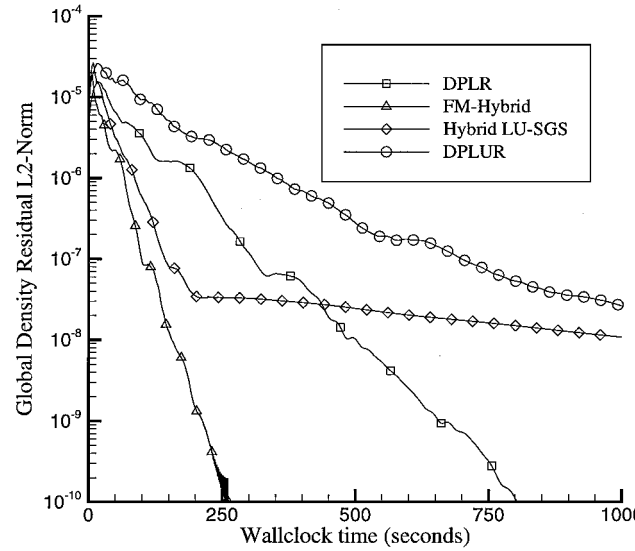

Fig. 12 Comparison of parallel implicit operators (three-dimensional OLS).

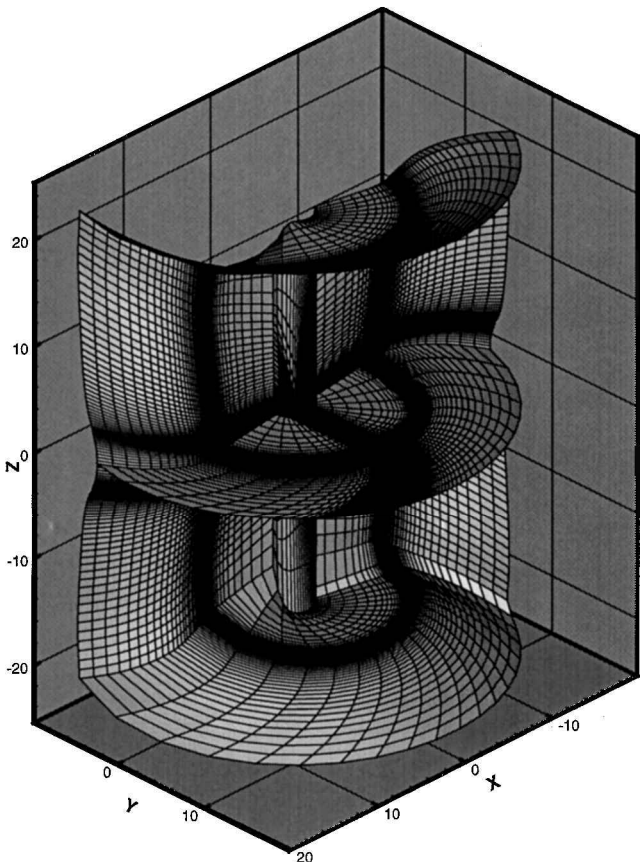

Fig. 11 Three-dimensional Navier–Stokes grid.

2.75×10^6. In a quasi-steady computation, the blade is fixed, and the air is circulated around the rotor. In forward-flight cases, an additional freestream flow is imposed.

The calculations are performed on a $137 \times 50 \times 50$ body-conforming grid, constructed by stacking and bending two-dimensional C–H-type grids. The minimum spacing of the grid in the normal direction near the surface is 0.0001 chords. This corresponds approximately to a maximum CAR of 12,000. The partitioning of the grid is done in the wraparound and radial directions. Each processor has 17 grid points in the wraparound direction, 14 grid points in the radial direction, and 50 grid points in the normal direction. The three-dimensional grid is shown in Fig. 11.

Figure 12 is a comparison of the four methods with each other. FM-Hybrid performs better than any other method. The computational intensity of DPLR enters the picture with this test case. Although DPLR requires a lower number of iterations, overall FM-Hybrid is approximately three times faster. This is because a DPLR iteration is approximately 2.5 times slower than an FM-Hybrid iteration. The high CAR of the grid is apparently effecting the convergence rate of the diagonal methods. As a result, the FM-Hybrid method performs better than any other methods used for this problem.

Three-Dimensional Viscous Time-Accurate Unsteady Results

The three-dimensional unsteady computations are performed for the same configuration used in the preceding section. The quasi-steady solution at $\Psi = 0$ deg is used as the starting solution, and time-accurate calculations are performed for one full revolution of the blade. Constant time steps are used for all of the methods. At each time iteration, the whole computational grid is rotated with the blade, and three relaxation iterations are performed. The reason for using relaxation iterations is to reduce the factorization error in the implicit operator.

First, an unsteady solution is obtained for a very small time step that corresponds to a $\frac{1}{20}$ deg azimuth per time step using the FM-Hybrid operator. The FM-Hybrid operator is chosen for this run because it uses exact forms of the Jacobians on the left-hand side of the system. The pressure coefficient is evaluated at the quarter chord and a radial position of $r/R = 0.864$ of the blade. This time-accurate pressure coefficient is then used as a baseline solution, and results obtained by using larger time steps are compared with it. In our computations, we found that smaller time steps for DPLR had to be used because of the inherent time-step restrictions of the method compared to the rest of the operators. Therefore, for time-accurate calculations we concluded that DPLR is not efficient, and results from this method will not be presented. This may be due to the explicit boundary conditions used in TURNS, which are difficult to change to the implicit boundary conditions needed for the best convergence with DPLR.

A time step corresponding to $\Delta\Psi = 0.10$ deg azimuth angle is used for the runs, and relative error in pressure coefficients compared to the baseline solution is plotted in Fig. 13. It can be seen from Fig. 13 that the FM-Hybrid method has a smaller error compared to the hybrid LU-SGS and DP-LUR methods. Therefore, one can obtain the same level of accuracy either by using larger time steps or a smaller number of subiterations for FM-Hybrid. Figure 14 shows the error for three subiterations when $\Delta\Psi = 0.10$ deg is used for hybrid LU-SGS and DP-LUR and $\Delta\Psi = 0.50$ deg is used for FM-Hybrid. Also shown in Fig. 14 is the error for FM-Hybrid for a single subiteration and $\Delta\Psi = 0.10$ deg. It is apparent that the error

Table 1 Comparison of implicit operators for unsteady calculations

Method	$\Delta\Psi$, deg	Subiterations	Average error	Wallclock time, s
FM-Hybrid	0.50	3	9.01×10^{-4}	1700
FM-Hybrid	0.10	1	5.37×10^{-4}	2777
FM-Hybrid	0.10	3	0.86×10^{-4}	8458
Hybrid LU-SGS	0.10	3	15.2×10^{-4}	6119
DP-LUR	0.10	3	9.91×10^{-4}	10643

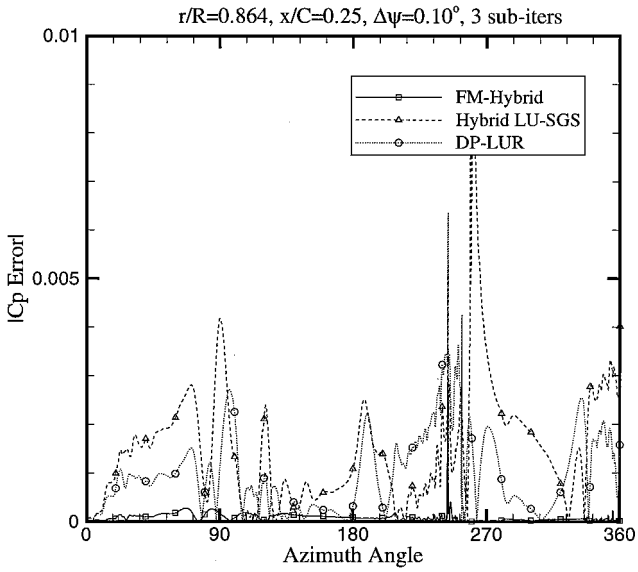

Fig. 13 Unsteady C_p error for implicit operators: three subiterations.

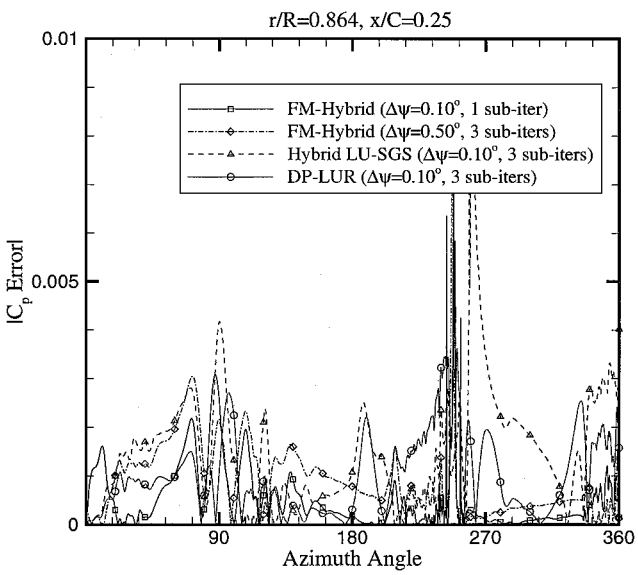

Fig. 14 Unsteady C_p error for implicit operators.

for FM-Hybrid using either larger time steps or a smaller number of subiterations is comparable with the errors for hybrid LU-SGS and DP-LUR.

The average of the errors and the total time required for different methods are given in Table 1. The use of larger time steps for the FM-Hybrid method yields faster overall solution times, although it is slower for each iteration for the same level of accuracy. For the case where three subiterations are used for all methods and $\Delta\Psi = 0.50$ deg for FM-Hybrid and $\Delta\Psi = 0.10$ deg for hybrid LU-SGS and DP-LUR, FM-Hybrid yields 72 and 84% solution time savings over hybrid LU-SGS and DP-LUR, respectively, with similar error levels. When the azimuth increment is fixed to 0.10 deg and single subiteration is used for FM-Hybrid, one can obtain 55% savings over hybrid LU-SGS and 74% savings over DP-LUR.

Parallel Performance

Previously, the Euler mode of TURNS was modified by Wissink[11] for parallel architectures. In that study (compared to the computational time required on a single CPU Cray C-90), the new program resulted in a 12-fold decrease for hybrid LU-SGS and an 8-fold decrease for DP-LUR on 114 processors of an IBM SP2 computer. With the current improvements to the implicit solvers, we expect faster computation times.

To investigate the parallel efficiency of the implicit operators implemented in TURNS, the three-dimensional viscous quasi-steady case studied earlier is run for different number of processors. The iterations are stopped when the global density residual dropped to $1E-08$. The minimum number of processors used in this study is nine. Therefore, the parallel speedup for the algorithms is defined by $T(9)/T(P)$, where $T(P)$ is the total time required when P number of processors are used. In that sense, the ideal curve speedup is defined as $P/9$.

The parallel speedup of the methods is presented in Fig. 15. It is seen that both of the methods have very good speedups. FM-Hybrid has superlinear speedup, whereas DPLR is just below the ideal speedup line. Having a superlinear speedup indicates that the cache misses are reduced as we increase the number of processors. Although not shown here, the parallel speedup could be affected when more number of processors are used for the calculations. Because there are few number of points in one direction, the use of the values from the previous time step at the boundaries could degrade the convergence resulting in less efficient calculations. For more realistic calculations with larger grids, a larger number of processors is possible.

As mentioned before, the parallel performance of the implicit methods is based on the three-dimensional viscous quasi-steady calculations. However, note that the parallel performance is similar for two-dimensional viscous steady-state and three-dimensional viscous unsteady calculations.

Also investigated in this section is the percentage of the time spent in communication for the two-dimensional viscous problem. The results are presented in Table 2 for both 9 and 27 processors used in

Table 2 Percentage of the time spent for different implicit operators for the two-dimensional problem

Method	i_{sweep}	Number of processors	Communication %
DPLR	4	9	10.5
DPLR	4	27	23.4
FM-Hybrid	1	9	11.0
FM-Hybrid	1	27	21.4
Hybrid LU-SGS	1	9	12.7
Hybrid LU-SGS	1	27	25.8
DP-LUR	4	9	14.3
DP-LUR	4	27	28.7

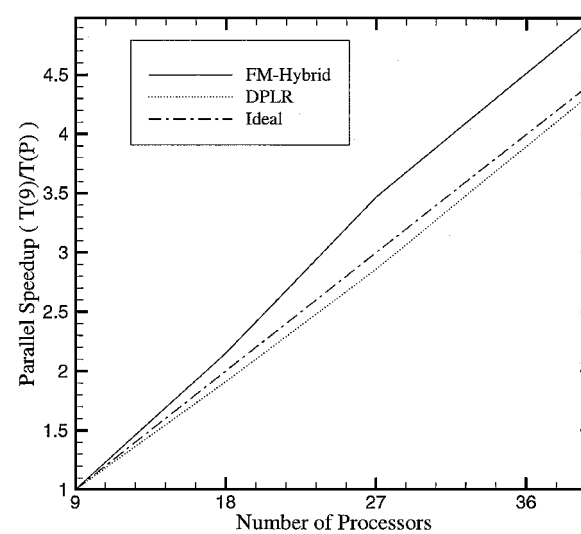

Fig. 15 Parallel speedup of implicit operators.

the solution of this problem. It appears that the percent of communication for all methods seems somewhat high. This is mainly due to the relatively small problem size, that is, 850 and 350 grid points per processor for 9 and 27 processors, respectively. If a larger problem were used, all of the costs such as timers and barrier synchronizations that appear artificially high in the smaller problem would be lower. Also, note that when the number of processors is increased, the percent communication increases. There are two main reasons for this. First, the absolute communication time increases, and second, because the number of grid points per processor decreases, the computational intensity decreases. One can also see that the communication percentage for the full matrix methods are lower than the diagonal methods. This is mainly because full matrix methods are more computationally intensive compared to the diagonal methods, which can mask the communication time better.

Conclusions

Parallelization techniques for rotorcraft aerodynamics were studied. Two algorithmic changes are made for the efficient parallelization of the Navier–Stokes mode of TURNS.

A new implicit operator, FM-Hybrid, based on the hybrid LU-SGS operator of Wissink[11] and the DPLR implicit operator developed by Wright et al.,[16] has been implemented in TURNS. Both the hybrid LU-SGS and DPLR methods use the exact flux Jacobians and remove the spectral radius approach in the LU-SGS algorithm. The main reason for using exact forms of the Jacobians is to remove the overstabilizing effect of the spectral radius approximation for high CAR grids, which are needed for viscous computations.

The operators are parallelized using message passing, and their efficiency is compared with DP-LUR and hybrid LU-SGS methods on an IBM SP2 supercomputer. Remarkable performance increase over the diagonal methods has been obtained with the implementation of these operators. The FM-Hybrid method seems to perform slightly better than DPLR in the three-dimensional, quasi-steady Navier–Stokes case. On the other hand, the DPLR method performs better than FM-Hybrid for two-dimensional test cases regardless of the maximum CAR of the grid. We have also tested these operators for the first time for unsteady three-dimensional calculations. We found that DPLR is not efficient for time-accurate calculations. However, FM-Hybrid performs better than the diagonal-based methods, and significant CPU time savings, for example, 72% vs hybrid LU-SGS and 84% vs DP-LUR, can obtained.

Acknowledgments

The first author would like to thank the Scientific and Technical Research Council of Turkey and Purdue Research Foundation for their support. We also thank R. C. Strawn from NASA Ames Research Center and A. M. Wissink from Lawrence Livermore National Laboratory for their help with the TURNS code.

References

[1]Strawn, R. C., and Tung, C., "The Prediction of Transonic Loading on Advancing Helicopter Rotors," NASA TM-88238, April 1986.

[2]Strawn, R., and Caradonna, F., "Conservative Full Potential Model for Unsteady Transonic Rotor Flows," *AIAA Journal*, Vol. 25, No. 2, 1987, pp. 193–198.

[3]Srinivasan, G. R., and Sankar, L. N., "Status of Euler and Navier–Stokes CFD Methods for Helicopter Applications," *Proceedings of the 2nd AHS International Aeromechanics Specialists' Conference*, Vol. 2, American Helicopter Society, Alexandria, VA, 1995, pp. 6-1–6-19.

[4]Knight, D. D., "Parallel Computing in Computational Fluid Dynamics," *77th AGARD FDP Meeting and Symposium on Progress and Challenges in CFD Methods and Algorithms*, AGARD, No. 578, April 1996, pp. 3-1–3-14.

[5]Anirudh, M., and Long, L. N., "Unsteady Separated Flow Simulations Using a Cluster of Workstations," AIAA Paper 2000-0272, Jan. 2000.

[6]Sterling, T., "How to Build a Beowulf: Assembling, Programming, and Using a Clustered PC Do-it-Yourself Supercomputer," Supercomputing 1997 Conf., San Jose, CA, Nov. 1997.

[7]Srinivasan, G. R., Baeder, J. D., Obayashi, S., and McCroskey, W. J., "Flowfield of a Lifting Rotor in Hover: A Navier–Stokes Simulation," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2371–2378.

[8]Srinivasan, G. R., and Baeder, J. D., "TURNS: A Free-Wake Euler/Navier–Stokes Numerical Method for Helicopter Rotors," *AIAA Journal*, Vol. 31, No. 5, 1993, pp. 959–962.

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

[10]Wissink, A. M., Lyrintzis, A. S., and Strawn, R. C., "Parallelization of a Three-Dimensional Flow Solver for Euler Rotorcraft Aerodynamics," *AIAA Journal*, Vol. 34, No. 11, 1996, pp. 2276–2283.

[11]Wissink, A. M., "Efficient Parallel Implicit Methods for Rotory–Wing Calculations," Ph.D. Dissertation, Dept. of Aerospace Engineering and Engineering Mechanics, Univ. of Minnesota, Minneapolis, MN, May 1997.

[12]Wissink, A. M., Lyrintzis, A. S., and Chronopoulos, A. T., "Parallel Newton–Krylov Method for Rotary–Wing Flowfield Calculations," *AIAA Journal*, Vol. 37, No. 10, 1999, pp. 1213–1221.

[13]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.

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

[15]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. 1371–1378.

[16]Wright, M. J., Candler, G. V., and Bose, D., "Data-Parallel Line Relaxation Method for the Navier–Stokes Equations," *AIAA Journal*, Vol. 36, No. 9, 1998, pp. 1603–1609.

[17]Yoon, S., and Kwak, D., "Three-Dimensional Incompressible Navier–Stokes Solver Using Lower–Upper Symmetric-Gauss–Seidel Algorithm," *AIAA Journal*, Vol. 22, No. 6, 1991, pp. 874, 875.

[18]Kandula, M., and Bunning, P. G., "Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in OVERFLOW Thin-Layer Navier–Stokes Code," AIAA Paper 94-2357, June 1994.

[19]Srinivasan, G. R., Raghavan, V., Duque, E. P. N., and McCroskey, W. J., "Flowfield Analysis of Modern Helicopter Rotors in Hover by Navier–Stokes Method," *Journal of the American Helicopter Society*, Vol. 38, No. 3, 1993, pp. 3–10.

[20]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.

[21]Roe, P. L., "Approximate Reimann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 3, 1981, pp. 357–372.

[22]Tysinger, T. L., and Caughey, D. A., "Alternating Direction Implicit Methods for the Navier–Stokes Equations," *AIAA Journal*, Vol. 30, No. 8, 1992, pp. 2158–2161.

[23]Van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, 1979, pp. 101–136.

[24]Yoon, S., and Jameson, A., "Lower–Upper Symmetric Gauss–Seidel Method for the Euler and Navier–Stokes Equations," *AIAA Journal*, Vol. 26, 1988, pp. 1025, 1026.

[25]Srinivasan, G. R., Chyu, W. J., and Steger, J. L., "Computation of Simple Three-Dimensional Wing–Vortex Interaction in Transonic Flow," AIAA Paper 81-1206, June 1981.

[26]Pulliam, T. H., "Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings," Lecture Notes, von Kármán Inst. for Fluid Dynamics Lecture Series, Brussels, Jan. 1986.

[27]Wissink, A. M., Lyrintzis, A. S., and Strawn, R. C., "On Improving Parallelism in the Transonic Unsteady Rotor Navier–Stokes (TURNS) Code," *77th AGARD FDP Meeting and Symposium on Progress and Challenges in CFD Methods and Algorithms*, AGARD, No. 578, 1996, pp. 6-1–6-11.

[28]Ortega, J. M., *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum, New York, 1988.

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

[30]Ekici, K., "Parallel Computing Techniques for Rotorcraft Aerodynamics," Ph.D. Dissertation, School of Aeronautics and Astronautics, Purdue Univ., West Lafayette, IN, Dec. 2001.

[31]Cook, P. H., McDonald, M. A., and Firmin, M. C. P., "Aerofoil RAE2822 Pressure Distributions and Boundary Layer and Wake Measurements," *Experimental Data Base for Computer Program Assessment*, AR-138, AGARD, 1979.

[32]Wright, M. J., "A Family of Data Relaxation Methods for the Navier–Stokes Equations," Ph.D. Dissertation, Dept. of Aerospace Engineering and Engineering Mechanics, Univ. of Minnesota, Minneapolis, MN, June 1997.

[33]Ekici, K., and Lyrintzis, A. S., "Parallel Computing Techniques for Rotorcraft Aerodynamics," AIAA Paper 2000-2617, June 2000.

A. Plotkin
Associate Editor