AIAA JOURNAL
Vol. 37, No. 10, October 1999

Parallel Newton–Krylov Method for Rotary-Wing Flowfield Calculations

Andrew M. Wissink*
NASA Ames Research Center, Moffett Field, California 94035-1000
Anastasios S. Lyrintzis†
Purdue University, West Lafayette, Indiana 47907
and
Anthony T. Chronopoulos‡
University of Texas at San Antonio, San Antonio, Texas 78249

The use of Krylov subspace iterative methods for the implicit solution of rotary-wing flowfields on parallel computers is explored. A Newton–Krylov scheme is proposed that couples conjugate-gradient-like iterative methods within the baseline structured-grid Euler/Navier–Stokes flow solver, transonic unsteady rotor Navier–Stokes. Two Krylov methods are studied, generalized minimum residual and orthogonal *s*-step orthomin. Preconditioning is performed with a parallelized form of the lower–upper symmetric Gauss–Seidel operator. The scheme is implemented on the IBM SP2 multiprocessor and applied to three-dimensional computations of a rotor in forward flight. The Newton–Krylov scheme is found to be more robust and to attain a higher level of time accuracy in implicit time stepping, increasing the allowable time step. The method yields approximately a 20% reduction in solution time with the same level of accuracy in time-accurate calculations but requires more memory than do more traditional implicit techniqes.

Introduction

THE accurate numerical simulation of the aerodynamics and the aeroacoustics of rotary-wing aircraft is a complex and challenging problem. Three-dimensional unsteady Euler/Navier–Stokes computational fluid dynamics (CFD) methods are widely used,[1–4] but their application to large problems is limited by the amount of computer time they require. Efficient utilization of parallel processing is one effective means of speeding up these calculations.[5] Another is the use of more efficient numerical solution methods.

In recent years, a number of researchers[6–14] have reported benefits in the use of conjugate-gradient-like Krylov subspace iterative solvers for nonlinear CFD problems. Krylov methods are used in conjunction with more traditional implicit solution methods, which act as a preconditioner, to accelerate the nonlinear convergence in the implicit solution. They are particularly useful for problems for which traditional methods exhibit slow convergence, which can occur with very fine viscous grids, certain turbulence models, and with multiple grids. A large memory requirement is the main drawback associated with Krylov methods. This has limited their application mainly to two-dimensional problems in the past, although some three-dimensional calculations have been successfully performed recently.[11,13]

Recent advances in parallel processing technology may encourage more widespread use of conjugate-gradient-like schemes within the CFD community. The methods are amenable to parallel processing because most operations are performed on large vectors that can be easily distributed. Further, the large memory capacity available on modern distributed-memory parallel machines can effectively lift many of the storage restrictions that have limited their use in the past. It is reasonable to postulate that Krylov methods will be applicable to relatively large three-dimensional problems in the not-too-distant future. Keyes et al.[15] point out that the scalability of Newton–Krylov methods make them well suited for CFD calculations on large-scale massively parallel petaflop computer architectures.

In this paper, we investigate the performance of Krylov subspace iterative solvers applied to three-dimensional calculations of a rotor in forward flight. Our goal is to provide insight into the performance of these methods for typical large-scale rotary-wing aerodynamics computations. Two iterative methods are tested: the popular generalized minimum residual (GMRES) method[16] and a relatively new scheme called orthogonal *s*-step orthomin[17] (OSOmin). They are applied in a matrix-free inexact Newton formulation within the baseline transonic unsteady rotor Navier–Stokes (TURNS) code.[2,3] In an earlier work,[5] an efficient parallel implementation of the implicit lower–upper symmetric Gauss–Seidel (LU-SGS) operator[18] in TURNS was introduced. This operator is used here for preconditioning the Krylov methods. The Newton–Krylov scheme is coded with message-passing interface (MPI) message passing and implemented on the IBM SP2 multiprocessor. All calculations are restricted to the Euler equations by use of a nonlifting rotor, but the approach is readily extendible to viscous flows.

Baseline Numerical Method

The baseline numerical method is the structured-grid Euler/Navier–Stokes solver, TURNS.[2,3] The TURNS code was developed by Srinivasan in conjunction with the U.S. Army Aeroflightdynamics Directorate at NASA Ames Research Center. It is used for calculating the flowfield of a helicopter rotor (without fuselage) in hover and forward flight conditions. In addition to NASA and the U.S. Army, various universities and the four major U.S. helicopter companies use the code. The excellent predictive capabilities of the TURNS code for lifting rotors in hover and forward-flight conditions, in both subsonic and transonic flow regimes, have been validated against wind-tunnel data in other studies.[2–4]

The governing equations solved by the TURNS code are the three-dimensional unsteady compressible thin-layer Navier–Stokes equations, applied in conservative form in a generalized body-fitted curvilinear coordinate system:

$$\partial_\tau \boldsymbol{q} + \partial_\xi \boldsymbol{E} + \partial_\eta \boldsymbol{F} + \partial_\zeta \boldsymbol{G} = (\sigma/Re)\partial_\zeta \boldsymbol{S} \tag{1}$$

where $\boldsymbol{q}$ is the vector of conserved quantities; $\boldsymbol{E}$, $\boldsymbol{F}$, and $\boldsymbol{G}$, are the inviscid flux vectors; and $\boldsymbol{S}$ is the viscous flux vector. The generalized coordinates are $\tau = t$, $\xi = \xi(x, y, z, t)$, $\eta = \eta(x, y, z, t)$, and

Received 5 June 1997; revision received 1 February 1999; accepted for publication 12 March 1999. This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

*Research Scientist, MCAT, Inc., MS 258-1. Member AIAA.
†Associate Professor, School of Aeronautics and Astronautics. Associate Fellow AIAA.
‡Associate Professor, Division of Computer Science, 6900 North Loop 1604 West.

$\zeta = \zeta(x, y, z, t)$, where the coordinate system x, y, z, t is attached to the blade. The TURNS code is run in Euler mode (i.e., $\sigma = 0$) for all calculations presented in this paper.

The inviscid fluxes are evaluated with Roe's upwind differencing[19] in all three directions. The use of upwinding obviates the need for user-specified artificial dissipation and improves the shock capturing in transonic flowfields. The spatial differencing scheme is third-order accurate with the higher-order accuracy obtained using van Leer's (MUSCL) approach.[20] Flux limiters are applied so that the scheme is total variation diminishing.

The implicit operator used in the TURNS code for time stepping in both steady and unsteady calculations is the LU-SGS operator of Yoon and Jameson.[18] This operator takes the form

$$LD^{-1}U\,\Delta \boldsymbol{q}^n = -\Delta t \boldsymbol{f}(\boldsymbol{q}^n) \tag{2}$$

where $\Delta \boldsymbol{q}^n = \boldsymbol{q}^{n+1} - \boldsymbol{q}^n$ and $\boldsymbol{f}(\boldsymbol{q}^n)$ is the spatially differenced right-hand-side vector:

$$\boldsymbol{f}(\boldsymbol{q}^n) = \delta_\xi \boldsymbol{E}^n + \delta_\eta \boldsymbol{F}^n + \delta_\zeta \boldsymbol{G}^n \tag{3}$$

The factors D, L, and U are diagonal, lower, and upper tridiagonal matrices, respectively, determined with a spectral approximation for the flux Jacobians. The use of a spectral approximation places the largest terms on the diagonal matrix, which ensures diagonal dominance and allows the method to converge for any time step. A two-step symmetric Gauss–Seidel scheme is used for the solution of Eq. (2).

For unsteady time-accurate calculations with LU-SGS, the factorization error is reduced when subiterations are applied. By use of the solution at time level n, the initial condition is set $\boldsymbol{q}^{n+1,0} = \boldsymbol{q}^n$, and LU-SGS is applied to solve the following equation in each inner iteration:

$$(LD^{-1}U)^{n+1,m}\,\Delta \boldsymbol{q}^{n+1,m} = -\Delta t\left[\frac{\boldsymbol{q}^{n+1,m} - \boldsymbol{q}^n}{\Delta t} + \boldsymbol{f}(\boldsymbol{q}^{n+1,m})\right] \tag{4}$$

where $\Delta \boldsymbol{q}^{n+1,m} = \boldsymbol{q}^{n+1,m+1} - \boldsymbol{q}^{n+1,m}$. In Eq. (4), n refers to the nonlinear iteration or time step and m to the subiteration. Three subiterations were used for the cases in this work. On completion of the subiterations, the solution at the next time level is $\boldsymbol{q}^{n+1} = \boldsymbol{q}^{n+1,m_{\max}}$.

Additional algorithm details of the TURNS code are given in Ref. 3.

LU-SGS Parallelization

An efficient approach for parallelizing the LU-SGS implicit algorithm in TURNS has been introduced by the authors in an earlier work.[5] The approach is based on the data-parallel lower–upper relaxation (DP-LUR) operator of Candler et al.[21]

DP-LUR is an efficient parallel modification of LU-SGS for data-parallel-type parallel implementations. The algorithm uses the same factorization technique used in the LU-SGS algorithm, based on a spectral approximation of the flux Jacobians. However, it replaces the symmetric Gauss–Seidel sweeps, which are difficult to parallelize, with a point-relaxation method. Multiple relaxation iterations (generally 3–6) of the point-relaxation method are applied at each nonlinear iteration. The relaxation sweeps make the method amenable to parallel processing because it can be easily load balanced with only nearest-neighbor communication. Further details of DP-LUR are explained in Ref. 21.

An alternative approach for parallelizing the LU-SGS algorithm, which is based on the DP-LUR algorithm but designed specifically for multiple-instruction multiple-data parallel implementations (i.e., use of message passing), was introduced in an earlier work.[5] Once the computational space has been divided into subdomains, the original LU-SGS algorithm is applied simultaneously to each processor subdomain. Then border data between the subdomains are communicated by the relaxation-type approach of DP-LUR. The use of multiple relaxation sweeps is retained to enhance the robustness of the original algorithm lost in the domain decomposition. Because the method combines aspects of both LU-SGS and DP-LUR, it is referred to as hybrid LU-SGS. The algorithm is as follows.

Algorithm 1: Hybrid LU-SGS

$$\Delta \boldsymbol{q}^{(0)} = -D^{-1} \cdot \Delta t \boldsymbol{f}(\boldsymbol{q}^n)$$

For $i = 1, \ldots, i_{\text{sweep}}$, do
  communicate $\Delta \boldsymbol{q}^{(i-1)}$ data at processor borders to neighboring processors
  set $\Delta \boldsymbol{q}^{(i)} = \Delta \boldsymbol{q}^{(i-1)}$ at borders
  perform LU-SGS sweeps locally on each processor, computing $\Delta \boldsymbol{q}^{(i)}$ over each subdomain
End for

$$\Delta \boldsymbol{q}^n = \Delta \boldsymbol{q}^{(i_{\text{sweep}})}$$

On a single processor, the hybrid LU-SGS is identical to the original LU-SGS algorithm. On many processors (in the limit as the number of processors approaches the number of grid points), the algorithm becomes identical to DP-LUR. Like DP-LUR, hybrid LU-SGS can be implemented such that it is completely load balanced with only nearest-neighbor communication required between the subdomains. Hybrid LU-SGS was found to require fewer relaxation iterations at each nonlinear iteration and is consequently more computationally efficient for parallel calculations with the TURNS code by use of the third-order-accurate upwind-differenced method used in this work. The method converges for all cases tested with $i_{\text{sweep}} = 1$, but it experiences a slight reduction in convergence over the original LU-SGS algorithm. With $i_{\text{sweep}} = 2$, however, the method shows essentially convergence identical to that of the original LU-SGS, even with large numbers of processor subdomains. Further details of the hybrid LU-SGS algorithm are given in Ref. 5.

Inexact Newton's Method

Fully implicit Newton's method is the most robust technique for solving systems of nonlinear equations. To implement Newton's method, the fully coupled set of governing equations are linearized about time level n, which produces a large linear system at each nonlinear iteration:

$$\left[I + \Delta t\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^n\right]\Delta \boldsymbol{q}^n = -\Delta t \boldsymbol{f}(\boldsymbol{q}^n) \tag{5}$$

where $\Delta \boldsymbol{q}^n = \boldsymbol{q}^{n+1} - \boldsymbol{q}^n$ and $\boldsymbol{f}(\boldsymbol{q}^n)$ denotes the spatially differenced convective terms given in Eq. (3). If the linear system in Eq. (5) is solved exactly at each time level, the method becomes Newton's method exactly and is capable of achieving quadratic convergence and is completely time accurate with no restriction on the time step used for the nonlinear iteration. However, Newton's method in its exact form is not applicable to most CFD problems of interest because the CPU time and storage required for exactly solving the sparse linear system with a direct method is too costly.

An efficient alternative to the exact method is an inexact Newton method. An inexact Newton method refers to use of an approximate technique for solution of the linear system arising in Eq. (5). In CFD applications, this linear system becomes very large and sparse, and iterative methods based on the conjugate-gradient (CG) method of Hestenes and Stiefel[22] have been found to be very successful in determining an approximate solution to this type of system. These CG-type methods work on the principle that the residual of the linear system is minimized over a Krylov subspace and are therefore commonly referred to as Krylov methods. Further discussion of the Krylov methods used in this work is deferred to the next section.

Formation and storage of the Jacobian term $(\partial \boldsymbol{f}/\partial \boldsymbol{q})$ in Eq. (5) can be difficult and costly. Krylov solvers have the nice property that the Jacobian matrix is used only in matrix–vector multiplies, for which the following finite-difference numerical approximation can be used (to compute the product of the Jacobian times arbitrary vector $\boldsymbol{w}$):

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\boldsymbol{w} \approx \frac{\boldsymbol{f}(\boldsymbol{q} + \varepsilon \boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{q})}{\varepsilon} \tag{6}$$

The existence of the numerical matrix–vector approximation is important because it allows the use of nearly consistent left- and right-hand sides in the solution with a matrix-free approach. That is, the large cost of computing and storing the Jacobian at each nonlinear iteration is avoided.

This advantage does not come without other costs, however. The numerical derivative requires a function evaluation [i.e., $\boldsymbol{f}(\boldsymbol{q}+\varepsilon\boldsymbol{w})$] at every approximate matrix–vector multiply, which may be less efficient than an actual sparse-matrix multiplication. Also, the finite-difference approximation of the Jacobian is less accurate than an exact determination. Nevertheless, the amount of storage saved by utilizing the numerical approximation is significant. The matrix-free approach has been successfully applied in a number of other works.[7,11–13]

The choice of ε in approximation (6) can affect the nonlinear convergence of the method and should be chosen carefully. It is desirable to use as small a value as possible to increase the accuracy of the finite-difference approximation, but too small a choice will lead to numerical roundoff errors. When $\boldsymbol{q}$ and $\boldsymbol{w}$ are comparably scaled, ε should ideally be near the square root of the machine roundoff, $\sqrt{\varepsilon_{\text{mach}}}$, which is 10^{-7}–10^{-8} in double-precision accuracy. The entries in the $\boldsymbol{q}$ vector are nondimensionalized such that each entry has a value of approximately unity. The $\boldsymbol{w}$ vector is scaled within the Krylov methods such that its root mean square is approximately unity, so each entry has a value of approximately $1/\sqrt{N}$ (N is the dimension of the vector). Thus, a simple yet accurate determination of ε is

$$\varepsilon = \sqrt{N \cdot \varepsilon_{\text{mach}}} \tag{7}$$

This choice was also proposed by Cai et al.[12]

An important consideration in the use of approximate iterative methods is what level of linear accuracy is required within each nonlinear iteration for maintaining convergence in the nonlinear solution. Dembo et al.[23] have proven that the nonlinear iterations will converge as long as the linear solution accuracy is at least

$$\|\boldsymbol{f}(\boldsymbol{q}^n) + \boldsymbol{f}'(\boldsymbol{q}^n)\Delta\boldsymbol{q}^n\|_2 \le \eta\|\boldsymbol{f}(\boldsymbol{q}^n)\|_2 \tag{8}$$

where $0 < \eta \le 1$. That is, the L2 norm of the linear residual is less than or equal to that of the nonlinear residual. In enforcing this nonlinear convergence criteria, a certain fixed value of η is specified and, at each nonlinear iteration, linear iterations of the Krylov solver are performed until relation (8) is satisfied. A maximum of 20 linear iterations is specified in the code, but this limit is rarely reached.

Iterative Methods

Over the past two decades, a number of efficient Krylov subspace iterative methods have been developed for solving large sparse linear systems. These methods are formulated as generalizations of the well-known CG method.[22] The convergence of CG is ensured only for symmetric positive definite linear systems, but most CFD applications of interest (e.g., transonic flow) generate nonsymmetric linear systems. A number of generalizations of the CG have been proposed for nonsymmetric systems. These nonsymmetric generalizations can be divided into two main categories, (biorthogonal) Lanczos-based methods and Arnoldi-based methods.

Lanczos-based methods include the CG squared[24] method, stabilized variants of the biconjugate-gradient method,[25] and methods based on the quasi-minimum residual idea.[26] The approach used in deriving these methods from the CG is to relax the minimization property while keeping the efficient three-term-recurrence relations. This allows the size of the Krylov subspace to grow (making the implicit solution more robust) without an increase in memory. However, relaxing the minimization property can cause the linear convergence of the norm of the residual to become erratic, which can negatively affect the nonlinear convergence. Also, biorthogonal Lanczos and biconjugate-gradient-type methods require the transpose of the Jacobian for matrix–vector multiplies. The computation of A^T requires an explicit determination of the Jacobian matrix A, rendering them inapplicable with a matrix-free implementation approach.

Arnoldi-based schemes are formulated with the approach of relaxing the three-term-recurrence relations while keeping the residual minimization property. Some examples of Arnoldi-based schemes include the GMRES method,[16] the generalized conjugate residual method,[27] the generalized conjugate-gradient least-squares method,[28] and orthomin.[29] As a result of keeping the residual minimization property, the convergence of these schemes tends to be more stable. However, relaxing the three-term recurrences requires that all direction vectors in the Krylov subspace be stored so that storage costs increase linearly with the dimension of the Krylov subspace.

The two iterative methods chosen for this work are Arnoldi-based schemes, for three reasons. First, the erratic convergence typically associated with Lanczos-based schemes is viewed as a deterrent to the acceptance of Krylov methods for a wide range of CFD problems. Second, Lanczos-based schemes cannot be implemented within the matrix-free approach. Third, separate studies by Ajmani and Liou[9] and McHugh and Knoll[7] have determined that the GMRES Arnoldi-based method was more efficient than several Lanczos-based schemes for solution of the Navier–Stokes equations.

The first iterative method applied in this work is the GMRES method of Saad and Shultz.[16] The application of the GMRES method within the context of nonlinear CFD problems is described in detail in a number of references.[6,10,11,13,30] A restarted version of the algorithm is used, GMRES(m), where m is the dimension of the Krylov subspace. With the restarted version, the Krylov subspace size is fixed, and if the linear solution does not satisfy the nonlinear convergence requirements in relation (8) after the fixed Krylov dimension is reached, the method is restarted with the current solution as the initial guess.

The second iterative method used is the OSOmin method of Chronopoulos and Swanson.[17] The so-called s-step class of iterative methods is formulated to be more parallelizable implementations of standard iterative methods. Some of the advantages associated with s-step methods include a higher degree of robustness, better parallelization potential, and reduced memory contention for shared-memory parallel machines (see Ref. 28 for a more general discussion of s-step methods). In 1991, Chronopoulos[31] introduced an s-step version of the classical nonsymmetric orthomin (k) method. This version was modified to maintain orthogonality between the different s directions by use of a modified Gram–Schmidt algorithm, which allows larger numbers of s steps (up to 16). The resulting OSOmin(s, k) method is theoretically proven to maintain the same level of robustness as GMRES(m) when $s = m$ (Ref. 28).

Both the GMRES and the OSOmin methods are proved to solve nonsymmetric linear systems with symmetric parts [i.e., $(A + A^T)/2$] positive definite (i.e., with all positive eigenvalues). In an earlier work,[30] the authors showed that OSOmin(s, k) outperformed GMRES(m) for solution of the steady two-dimensional transonic small disturbance equation on the vectorized shared-memory Cray C90.

Storage is a major consideration for the solution of three-dimensional problems, and the predominant total storage costs for the baseline TURNS code with and without the Krylov methods are shown in Table 1. Note that when $k = 1$ and $s = m$, the storage requirements of the GMRES and the OSOmin methods are approximately the same.

Preconditioning

The convergence rate of Krylov solvers is sensitive to the condition number (i.e., eigenvalue spectrum) of the coefficient matrix of the linear system. A preconditioner can be used to cluster the eigenvalues and thereby accelerate the solution of the iterative method. The proper choice of a preconditioner is essential for efficiency.

A preconditioner is applied in the following way: A preconditioning matrix P^{-1} is added to the left of the original unpreconditioned linear system in Eq. (5) and results in the following new linear system to be solved at each nonlinear iteration n:

$$P^{-1}\left[I + \Delta t\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^n\right]\Delta\boldsymbol{q}^n = -\Delta t\, P^{-1}\boldsymbol{f}(\boldsymbol{q}^n) \tag{9}$$

Table 1 Storage requirements[a]

Method	Storage
Baseline TURNS	$3N$
TURNS + GMRES(m)	$3N + (m+4)\cdot N$
TURNS + OSOmin(s, k)	$3N + (s\cdot k + 3)\cdot N$

[a] N = number of gridpoints × 5 (number of dependent variables in three dimensions).

For a preconditioner to be effective, it must perform a reasonable approximation to the inverse of the linear system and it must be able to perform this approximation at low cost (CPU time).

One of the more popular types of preconditioners is that based on incomplete factorizations [e.g., incomplete lower–upper (ILU) factorization]. Ajmani et al.[8] found the lower–upper symmetric successive overrelaxation (LU-SSOR) method of Yoon and Jameson[18] (of which LU-SGS is a subset) to be more efficient than ILU factorization for inexact Newton solution of transonic and subsonic two-dimensional Navier–Stokes flows. Considering these results and the fact that an effective parallelization strategy exists for LU-SGS (i.e., hybrid LU-SGS), it is an attractive preconditioning choice for our application.

Parallel Implementation

The flowfield domain is laid out on an array of processors by a single-program multiple-data parallel implementation strategy, which preserves the original structure of the code. The three-dimensional flowfield domain is divided in the wraparound and spanwise directions to form a two-dimensional array of processor subdomains, as shown in Fig. 1. Each processor executes a version of the code simultaneously for the portion of the flowfield that it holds. Coordinates are assigned to the processors to determine the global values of the data each holds. Border data are communicated between processors, and a single layer of ghost cells stores this communicated data. The MPI software routes communication between the processor subdomains.

There are essentially four main steps of the inexact Newton algorithm: 1) explicit flux evaluation by Roe-upwinded third-order-accurate spatial discretization to form the right-hand-side vector, 2) preconditioning by hybrid LU-SGS, 3) implicit solution by the Krylov subspace solver, and 4) explicit application of boundary conditions. The communication required in step 1 is straightforward. After the flux vectors are determined with the MUSCL routine, they are communicated and stored in the ghost layer. Then Roe differencing is applied (this additional communication step could be avoided by use of a ghost layer of two cells, but the present approach was easier to implement into the existing code). Preconditioning with hybrid LU-SGS in step 2 was explained above. The communication pattern for this step is nearest neighbor, and communications are performed only after the interior domain updates (i.e., after each sweep). The two Krylov subspace solvers utilized in step 3 perform, in addition to matrix-times-vector operations, two main numerical operations: SAXPY's and dot products. SAXPY's, or vector updates, are performed locally and require no communication. Global dot products are straightforward to parallelize: Local dot products are formed at each processor and a global sum operation (MPI-REDUCE) is used to compute the global product. This operation requires $\log_2 p$ messages, where p is the number of processors (the exact number of messages for the reduce operation may depend on how the MPI collective communication operations are implemented for the particular parallel architecture). Overall, both GMRES and OSOmin are quite scalable and easy to parallelize.

Application of the boundary conditions in step 4 can be done locally on each processor, with the exception of the averaging of data across the C-plane overlap behind the trailing edge of the rotor blades. Processors that contain data on the blade surface do not participate in the averaging but spend time invoking the flow-tangency

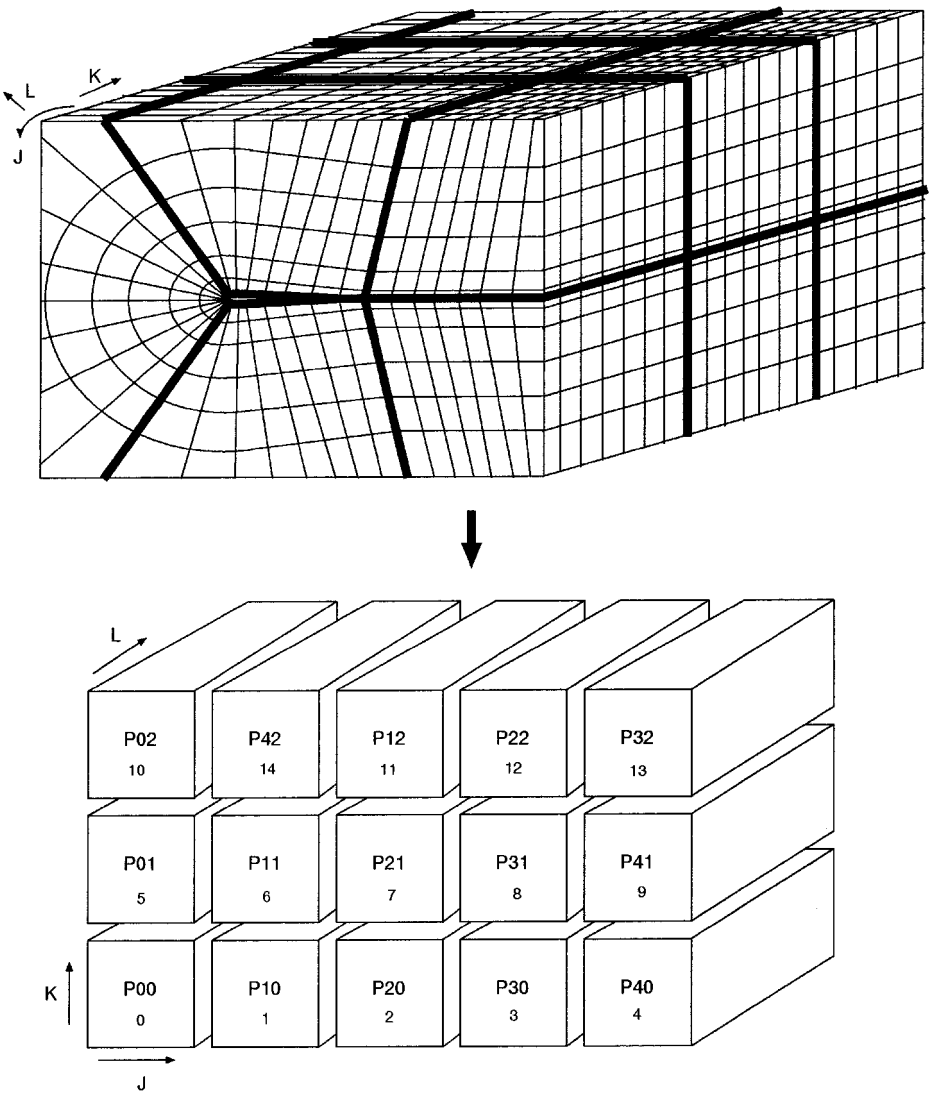


Fig. 1 Partitioning the three-dimensional domain on a two-dimensional array of processors.

boundary condition. Thus a good degree of load balance between processors is maintained during application of the boundary conditions. It should be noted here that load balance concerns caused us to split the flowfield subdomains in only two directions rather than three. If the domain were broken in the normal direction, interior processors would be required to sit idle during the communication step required for application of the boundary conditions at the C plane. This introduces a load imbalance that can significantly reduce parallel performance on large numbers of processors. Although breaking the domain in all three directions yields square subdomains, thereby minimizing the amount of data communicated, the inefficiency caused by the idle processors during the boundary-condition application is expected to outweigh the efficiency gained by use of square subdomains.

Computed Results

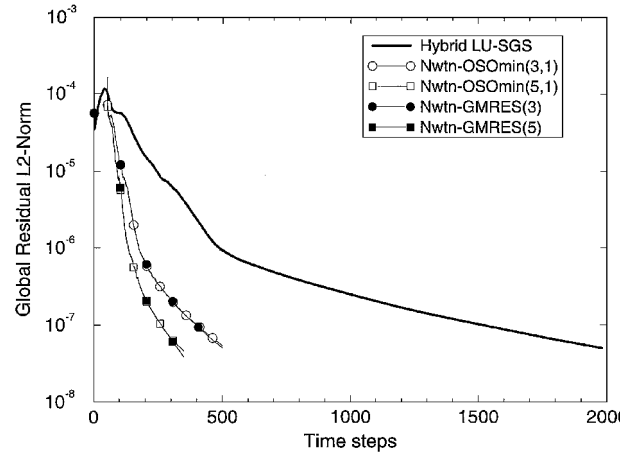
The parallelized inexact Newton implementation of the TURNS code is tested on the 160-node IBM SP2 at NASA Ames Research Center. The scheme is used to compute the quasisteady (i.e., blade-fixed) and unsteady flowfields of a rotating helicopter rotor (without fuselage) in forward flight. Viscous effects have not yet been included in the parallel implementation, so all calculations are performed in Euler mode for a nonlifting test case.

The flow is computed about a two-bladed symmetric untwisted operational load survey (OLS) helicopter blade rotating with tip Mach number $M_{\text{tip}} = 0.665$ and moving forward with a forward-flight advance ratio of $\mu = 0.258$. The OLS blade has a sectional airfoil thickness to chord ratio of 9.71% and is a $\frac{1}{7}$-scale model of the main rotor for the U.S. Army's AH-1 helicopter. A $135 \times 50 \times 35$ C–H type grid is used (shown in Fig. 2). The grid extends out to 2 rotor radii from the hub in the plane of the rotor and 1.5 rotor radii above and below the plane. The computed results with the TURNS code for this particular test case have been evaluated in other studies by Strawn et al.,[32] so this investigation will focus on only the numerical and parallel performance of the method.

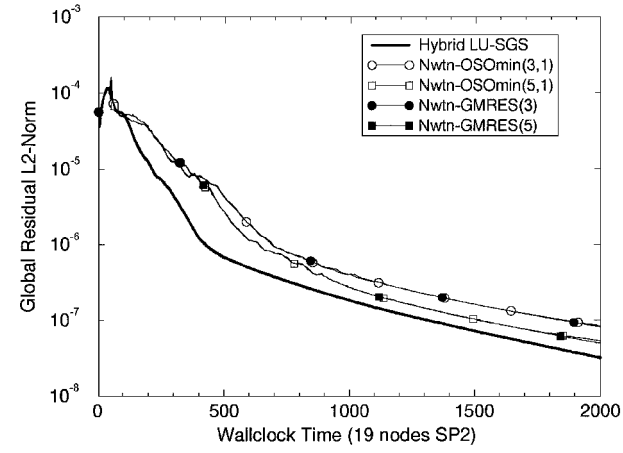
Results from this case only are reported here, but the scheme was also tested under a variety of conditions (i.e., subsonic and transonic flow), including two-dimensional test problems. These results are reported in Ref. 33.

Quasisteady

The nonlinear convergence with the inexact Newton scheme for a quasisteady calculation with blade azimuth angle at $\psi = 0$ deg is shown in Figs. 3 and 4. Figures 3a and 4a show the convergence of the L2 norm of the residual ($\|\boldsymbol{f}(\boldsymbol{q})\|_2$) vs time steps (nonlinear iterations), and Figs. 3b and 4b show the convergence vs wallclock time on 19 SP2 processors. The results in Fig. 3 use the nonlinear convergence criterion in relation (8) with $\eta = 0.95$ (i.e., multiple iterations of the Krylov method applied at each nonlinear iteration until the criteria is met), whereas Fig. 4 shows the results with only a single iteration of the Krylov method used at each nonlinear iteration. The inexact Newton cases are compared against the baseline case by use of the hybrid LU-SGS method only. Other processor partitions were also tested and, aside from the differences in wallclock solution time, the curves are essentially identical to those of the 19-processor case shown. The maximum residual ($\|\boldsymbol{f}(\boldsymbol{q})\|_\infty$) was also determined and showed similar results.



a) Nonlinear iterations



b) Wallclock time on 19 IBM SP2 processors

Fig. 3 Convergence of Newton–Krylov method with the nonlinear convergence of relation (8) enforced at each nonlinear iteration.

The hybrid LU-SGS method uses $i_{\text{sweep}} = 2$ because this was found in Ref. 5 to give nearly identical convergence to the original LU-SGS method for any number of processors. The iterative methods use Krylov subspace dimensions of 3 and 5 (that is, $m = 3, 5$ in GMRES and $s = 3, 5$ in OSOmin) because previous results[33] with a two-dimensional test case showed these values gave slightly better wallclock times than others. It should be noted, however, that the overall effect of the Krylov subspace dimension on the wallclock performance was found to be small. In OSOmin, k is set to 1 so the total storage costs for the Newton–GMRES and Newton–OSOmin comparison is essentially the same.

A comparison of Figs. 3 and 4 indicates that the Newton method is slightly more efficient when only a single iteration of the Krylov solver is applied at each nonlinear iteration than when multiple iterations of the Krylov method coupled with the nonlinear convergence criteria in relation (8) are used. This is most likely due to the fact that determination of the linear residual requires an extra matrix–vector multiply at the end of every linear iteration, which is used to determine only the residual vector to find whether the nonlinear convergence criteria have been satisfied. It is not required if the number of linear iterations is fixed. Considering that the matrix–vector multiplies constitute the most expensive operation, this additional operation at each nonlinear iteration can yield a noticeable reduction in efficiency. A more detailed study[33] showed no performance gains

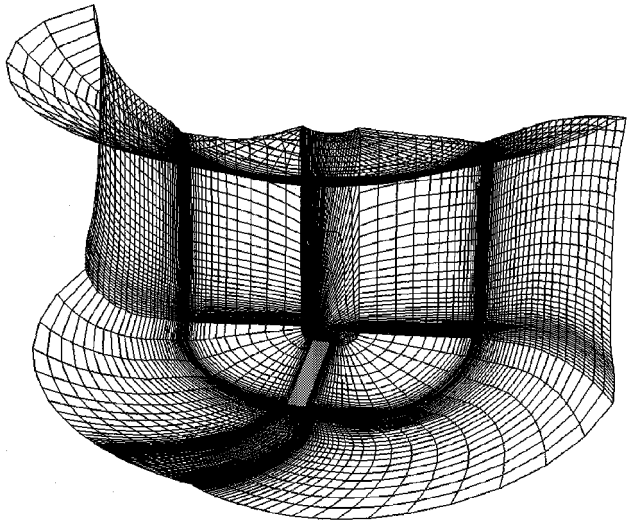
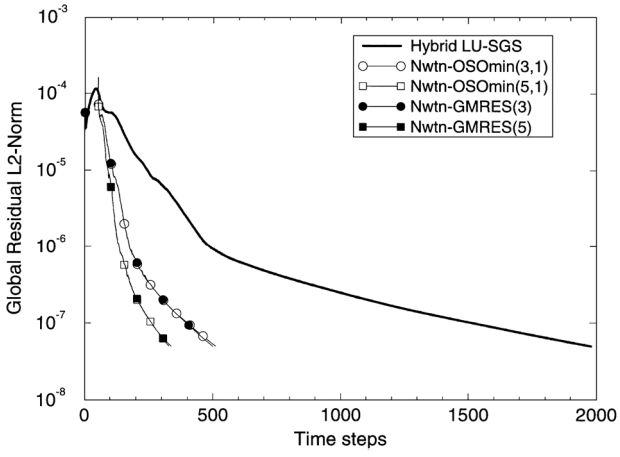
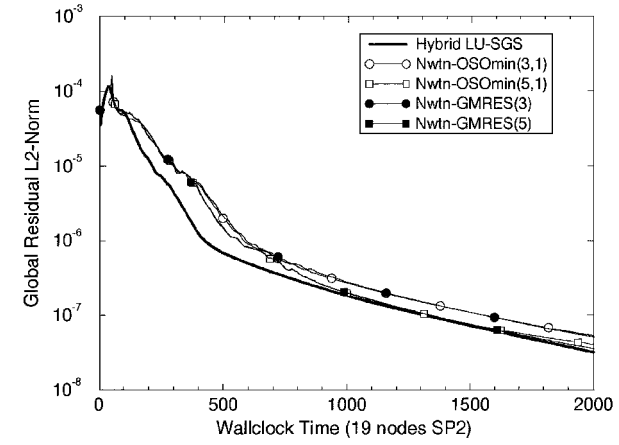


Fig. 2 135 × 50 × 35 C–H grid.



a) Nonlinear iterations



b) Wallclock time on 19 IBM SP2 processors

Fig. 4 Convergence of Newton–Krylov method with a single iteration of Krylov solver at each nonlinear iteration.

for various values of η and evaluation strategies for the residual. Thus the one-iteration algorithm is used in subsequent computations.

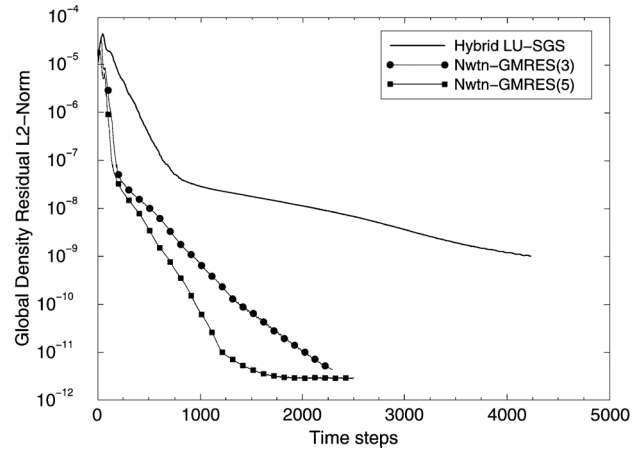
The Newton–Krylov approach shows improvement in the nonlinear convergence rate with increasing Krylov subspace dimension, but the effect on wallclock solution time is small because the time per nonlinear iteration increases by approximately the same factor as the reduction in number of nonlinear iterations. For the forced nonlinear convergence case in Fig. 3, the Newton–Krylov methods show slightly worse efficiency than hybrid LU-SGS methods. However, with the single-iteration case in Fig. 4, the efficiency is slightly worse in the initial nonlinear iterations but becomes approximately the same as that of the hybrid LU-SGS method as the solution converges. Both GMRES and OSOmin methods show nearly identical results with the same Krylov dimension.

Figure 5a shows the result of Newton–GMRES and hybrid LU-SGS quasisteady calculation carried out over a large number of nonlinear iterations. Convergence of the hybrid LU-SGS method stalls after a 4-order-of-magnitude reduction in the residual, whereas the Newton–Krylov method converges to nearly machine zero. The Newton–GMRES method with $m = 3$ converges to order 10^{-12} and to order 10^{-16} with $m = 5$. It should be noted that the standard LU-SGS algorithm also stalled for this case so the behavior is not a by-product of the parallel hybrid LU-SGS implementation. Figure 5b shows the nonlinear convergence vs CPU time comparison on 19 processors. This result implies that the Newton–Krylov method is a more numerically robust nonlinear solver, although the convergence of hybrid LU-SGS is probably sufficient for most CFD problems of interest.

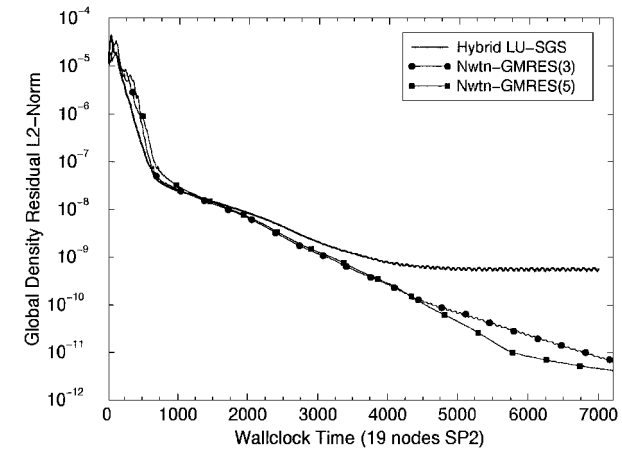
The parallel performance of methods is reported in Table 2. Shown are the average time per nonlinear iteration, percentage

Table 2 Parallel performance statistics for the baseline (hybrid LU-SGS), Newton–GMRES, and Newton–OSOmin methods on different processors of the SP2

Method	Time/iteration, s	%Communication	Speed up
4 Processors			
Hybrid LU-SGS	4.07	2.4	1
Nwtn–GMRES(3)	18.78	2.5	1
Nwtn–GMRES(5)	26.16	2.2	1
Nwtn–OSOmin(3,1)	18.58	2.1	1
Nwtn–OSOmin(5,1)	26.35	2.2	1
8 Processors			opt = 2
Hybrid LU-SGS	2.17	4.6	1.87
Nwtn–GMRES(3)	10.65	4.1	1.76
Nwtn–GMRES(5)	14.92	4.2	1.75
Nwtn–OSOmin(3,1)	10.68	4.2	1.74
Nwtn–OSOmin(5,1)	14.94	4.8	1.76
19 Processors			opt = 4.75
Hybrid LU-SGS	0.874	5.1	4.66
Nwtn–GMRES(3)	4.14	5.4	4.54
Nwtn–GMRES(5)	5.81	5.4	4.51
Nwtn–OSOmin(3,1)	4.13	5.3	4.50
Nwtn–OSOmin(5,1)	5.82	5.4	4.52
57 Processors			opt = 14.25
Hybrid LU-SGS	0.307	8.9	13.25
Nwtn–GMRES(3)	1.45	9.7	12.95
Nwtn–GMRES(5)	2.05	10.1	12.76
Nwtn–OSOmin(3,1)	1.42	9.6	13.08
Nwtn–OSOmin(5,1)	1.97	9.9	13.37
114 Processors			opt = 28.5
Hybrid LU-SGS	0.173	11.9	23.52
Nwtn–GMRES(3)	0.885	13.5	21.22
Nwtn–GMRES(5)	1.23	13.2	21.26
Nwtn–OSOmin(3,1)	0.823	12.3	22.58
Nwtn–OSOmin(5,1)	1.19	13.4	22.14



a) Nonlinear iterations



b) Wallclock time on 19 IBM SP2 processors

Fig. 5 Convergence of Newton–Krylov method carried to machine zero.

communication, and parallel speed up for the baseline and Newton–Krylov methods on 4, 8, 19, 57, and 114 IBM SP2 processors. The percentage communication is determined by the timing of all routines that invoke communication (any MPI routines) and comparison with the total average time per nonlinear iteration. Parallel speed ups are determined by comparison of the average time per nonlinear iteration with the 4-processor case.

Overall, the methods all demonstrate comparable parallel performance. There are no significant differences in the parallel speed up, although the baseline method (hybrid LU-SGS) and the Newton–OSOmin method show slightly better speed ups than the Newton–GMRES method on 114 processors. There is a noticeable increase in the percentage of communication for the Newton–Krylov method on larger numbers of processors. This is probably due to the larger number of global dot product operations in the Krylov solvers, for which the communications do not scale as well as the border communications as the number of processors grows.

GMRES and OSOmin give similar performances but there are a few subtle differences. On lower numbers of processors (i.e., 4 and 8), the Newton–OSOmin method requires slightly more time per nonlinear iteration than the Newton–GMRES method because OSOmin requires slightly more work. However, OSOmin is found to achieve slightly better parallelism on larger numbers of processors. Hence the time per nonlinear iteration of Newton–OSOmin is slightly faster than Newton–GMRES on 114 processors.

The measured execution rates of the code on the various SP2 processors applied to this problem are shown in Fig. 6. The megaflop (Mflop) rate for each processor partition is measured with IBM's parallel hardware performance monitor software. The execution rate on a single processor of the Cray C90 is also shown for comparison. The C90 version of the code is slightly different in that it uses a vectorized form of the original LU-SGS operator rather than the hybrid LU-SGS operator used on the SP2. Also, the rate measured on the C90 with Cray's hardware performance monitor is slightly different for each method but is shown as a single averaged point in Fig. 6 for convenience (actual rates on the C90 are 320 Mflops for the baseline TURNS code, 340 Mflops for Newton–GMRES, and 360 Mflops for Newton–OSOmin). The Newton–Krylov scheme shows slightly better Mflop per second rates than the baseline hybrid LU-SGS scheme, and OSOmin appears to show slightly better performance than GMRES.

It should be noted that our efforts focused primarily on attaining efficient parallel performance, and only a small effort was made to optimize the code for the reduced instruction set cache (RISC) processors on the SP2. The total execution rate could be enhanced (perhaps substantially) if further efforts were undertaken to optimize the single-processor performance of the code. The execution rate is also expected to improve with larger problem sizes.

Time-Accurate Unsteady

The Newton–Krylov approach allows for a higher degree of time accuracy for implicit time stepping because a more exact form of the left-hand-side Jacobian is used, making the left- and the right-hand sides more consistent. The method is studied here for a time-accurate computation of a single revolution of the OLS blade in forward flight.

Srinivasan[4] has shown that, by using three subiterations of the standard LU-SGS method at each nonlinear iteration, a time-accurate unsteady solution can be obtained by using a time step that corresponds to a $\frac{1}{4}$ degree of blade revolution per time step ($\Delta\psi = 0.25$ deg). We seek to match this result with the Krylov methods and compare the performance.

First, an unsteady solution is run with a very small time step that corresponds to $\frac{1}{10}$ degree azimuth per time step ($\Delta\psi = 0.10$ deg). The baseline hybrid LU-SGS method with three subiterations at each nonlinear iteration is used for this run. The time-varying pressure coefficient is recorded at a representative location on the blade ($\frac{1}{4}$ chord and $r/R = 0.80$). Then cases are run with larger time steps, and the resulting unsteady pressure coefficients are compared with the $\Delta\psi = 0.10$ deg result to determine the error.

Figures 7a and 7b show the pressure coefficient error obtained with the baseline and the inexact Newton methods with different time steps. Figure 7a shows the error resulting from time steps of $\Delta\psi = 0.25$ and 0.50 deg with three subiterations of LU-SGS at each nonlinear iteration (denoted by LUSGS-3). Figure 7b shows the errors with time steps of $\Delta\psi = 0.40$ and 0.50 deg obtained with Newton–OSOmin(3,1) with a single iteration of OSOmin(3,1) at each nonlinear iteration. It is apparent from the figures that the error from LUSGS-3 with $\Delta\psi = 0.25$ deg and Newton–OSOmin with $\Delta\psi = 0.40$ and 0.50 deg is comparable.

With LUSGS-3, in which $\Delta\psi = 0.25$ deg is considered the baseline case, Fig. 8 shows a close-up comparison of the errors obtained with Newton–OSOmin with $\Delta\psi = 0.40$ and 0.50 deg. The error

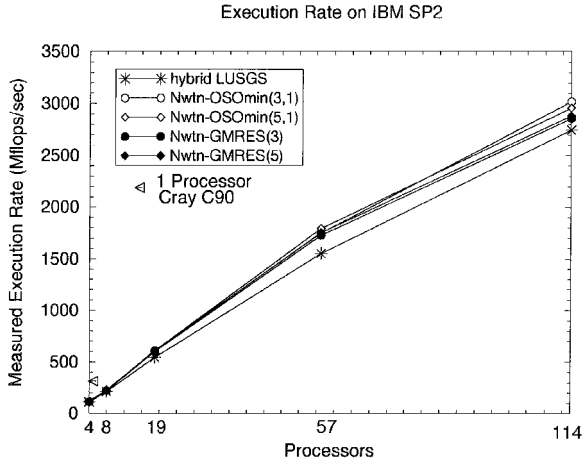


Fig. 6 Execution rate attained on various SP2 processors for 236 × 10^3 grid-point problem.

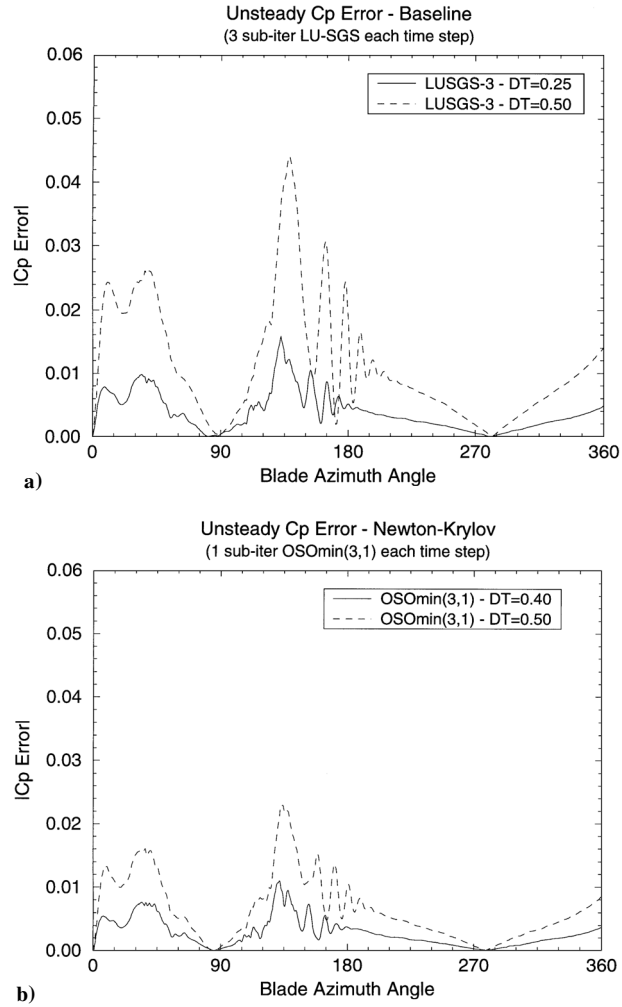


Fig. 7 Unsteady *Cp* error at $\frac{1}{4}$ chord, *r/R* = 0.8. Calculation of 1 rev by use of a) three subiterations of hybrid LU-SGS and b) one iteration of OSOmin(3,1) [results identical with those of GMRES(3)].

Table 3 Total solution time for time-accurate unsteady calculation of a full 360-deg blade revolution on SP2 19 processors

Method	Time step, deg	Solution time, s
Hybrid LU-SGS (3 subiterations)	$\Delta\psi = 0.25$	3844
Nwtn–OSOmin(3,1)	$\Delta\psi = 0.40$	3717
Nwtn–OSOmin(3,1)	$\Delta\psi = 0.50$	2973

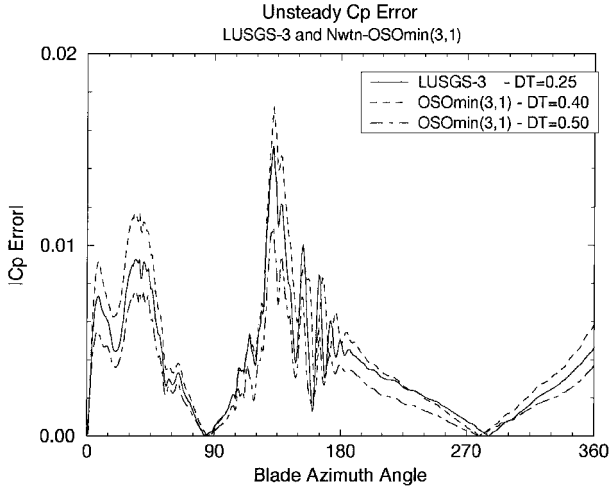


Fig. 8 Detailed comparison of unsteady *Cp* error: LUSGS-3 with time step $\Delta\psi = 0.25$ deg vs Newton–OSOmin(3,1) with $\Delta\psi = 0.40$ and 0.50 deg.

with $\Delta\psi = 0.40$ deg is slightly lower than the baseline, and the error with $\Delta\psi = 0.50$ deg is slightly larger. All are very close, however. Newton–GMRES(3) was also tried and gives results that are essentially identical to those of Newton–OSOmin(3,1). Different spanwise locations were also tested (reported in Ref. 33) and show similar results.

By allowing the use of larger time steps with the same level of accuracy, the inexact Newton method can yield faster overall solution times. Table 3 lists the total time required for completing a full 360 deg unsteady solution on 19 SP2 processors with three methods: 1) three subiterations of LU-SGS with a time step of $\Delta\psi = 0.25$ deg, 2) Newton–OSOmin(3,1) with $\Delta\psi = 0.40$ deg, and 3) Newton–OSOmin(3,1) with $\Delta\psi = 0.50$ deg. The total time is determined from the time per time-step data for each method in Table 2. With $\Delta\psi = 0.40$ deg, the total solution time with Newton–OSOmin is reduced by approximately 5% over that of the hybrid LU-SGS alone. With $\Delta\psi = 0.50$ deg, it is reduced by approximately 30%. Similar results are achieved with Newton–GMRES. Thus the inexact Newton algorithm is expected to yield wallclock solution time savings of the order of 10–20% for the same level of time accuracy.

Conclusion

A parallelized Newton–Krylov algorithm is investigated for structured-grid calculations of the flowfield of a helicopter rotor. Two preconditioned conjugate-gradient-like iterative methods are implemented within the baseline TURNS code: the well-known GMRES method and a relatively new *s*-step modification of the classical orthomin method called orthogonal *s*-step Orthomin (OSOmin). A parallel implementation of the LU-SGS operator is applied for left preconditioning, and the implementation is matrix free. The numerical and parallel performance is evaluated for quasisteady and unsteady three-dimensional Euler computations of a nonlifting helicopter blade on the IBM SP2 multiprocessor.

For quasisteady calculations, the Newton–Krylov algorithm shows some improvement over the baseline hybrid LU-SGS method in converging the solution to machine zero. The hybrid LU-SGS method stalls after a residual reduction of ~4 orders of magnitude. Before stall, the computational time required for the two methods are similar. For time-accurate unsteady calculations, the Newton–Krylov algorithm allows use of larger time steps for the same level of accuracy and leads to reductions in the total solution time by 10–20%. However, the Krylov methods require considerably more memory, and the reduction in CPU time may not justify the memory increase.

The parallel performance of the Krylov methods is good, but the overall parallel performance of the baseline method was not enhanced appreciably with their addition. The baseline method alone demonstrates good parallel performance (up to 114 processors tested) so, despite the high degree of parallelism inherent in the Krylov methods, their incorporation did not significantly enhance the overall parallel efficiency of the code. OSOmin and GMRES gives similar performances but OSOmin gives slightly better parallel speed ups on larger processor partitions.

This study was, to our knowledge, the first known application of Krylov methods for large-scale three-dimensional rotary-wing flowfield applications. Overall, we did not find substantial gains in their use for the inviscid calculations presented here. Follow-up work should include a study with a more complex flowfield (e.g., high Reynolds number viscous flows) as a number of authors have demonstrated substantial gains by using Krylov methods for such cases. Although this work focused on the solution of the Euler equations, the approach is readily adaptable to viscous flows as well. Future application of the Newton–Krylov approach to multiple grid solutions (e.g., multiblocked or overset) would be an interesting extension of the present work.

Acknowledgments

A. M. Wissink was supported by a NASA Graduate Student Fellowship from the High Performance Computing and Communications Program. Computer time on the IBM SP2 was provided by a grant from the Computational Aerosciences Division at NASA Ames Research Center. Additional computer time was also provided by a grant from the Pittsburgh Supercomputing Center. A. T. Chronopoulos acknowledges supercomputer time provided by San Diego Supercomputing Center, a Silicon Graphics, Inc./Cray 1996–1997 grant, and U.S. National Science Foundation support under Grant CCR-9496327. The authors acknowledge Roger Strawn for his advice during the course of this work and G. R. Srinivasan for his assistance with the TURNS code.

References

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

[2] 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.

[3] Srinivasan, G. R., Raghavan, V., Duque, E. P. N., and McCroskey, W. J., "Flowfield of a Lifting Rotor in Hover by a Navier–Stokes Method," *Journal of the American Helicopter Society*, Vol. 38, No. 3, 1993, pp. 3–13.

[4] 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.

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

[6] Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes," *Proceedings of the AIAA Seventh Computational Fluid Dynamics Conference*, AIAA, New York, 1985, pp. 67–74.

[7] McHugh, P. R., and Knoll, D. A., "Comparison of Standard and Matrix-Free Implementations of Several Newton–Krylov Solvers," *AIAA Journal*, Vol. 32, No. 12, 1994, pp. 2394–2400.

[8] Ajmani, K., Liou, M.-S., and Dyson, R. W., "Preconditioned Implicit Solvers for the Navier Stokes Equations on Distributed-Memory Machines," AIAA Paper 94-0408, Jan. 1994.

[9] Ajmani, K., and Liou, M.-S., "Implicit Conjugate-Gradient Solvers on Distributed-Memory Architectures," *Proceedings of the AIAA Twelfth Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1995, pp. 550–559.

[10]Rogers, S. E., "Comparison of Implicit Schemes for the Incompressible Navier–Stokes Equations," *AIAA Journal*, Vol. 33, No. 11, 1995, pp. 2066–2072.

[11]Hixon, R., Tsung, F. L., and Sankar, L. N., "Comparison of Two Methods for Solving Three-Dimensional Unsteady Compressible Viscous Flows," *AIAA Journal*, Vol. 32, No. 10, 1994, pp. 1978–1984.

[12]Cai, X.-C., Keyes, D. E., and Venkatakrishnan, V., "Newton–Krylov–Schwartz: An Implicit Solver for CFD," Inst. for Computer Applications in Science and Engineering, Rept. 95-87, Hampton, VA, Dec. 1995.

[13]Neilsen, E. J., Anderson, W. K., Walters, R. W., and Keyes, D. E., "Application of Newton–Krylov Methodology to a Three-Dimensional Unstructured Euler Code," *Proceedings of the AIAA Twelfth Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1995, pp. 981–990.

[14]Orkwis, P. D., and McRae, D. S., "Newton's Method Solver for the Axisymmetric Navier–Stokes Equations," *AIAA Journal*, Vol. 30, No. 6, 1992, pp. 1507–1514.

[15]Keyes, D. E., Kaushik, D. K., and Smith, B. F., "Prospects for CFD on Petaflops Systems," Inst. for Computer Applications in Science and Engineering, Rept. 97-73, Hampton, VA, Dec. 1997.

[16]Saad, Y., and Shultz, M., "GMRES: A Generalized Minimum Residual Algorithm for Solving Non-Symmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[17]Chronopoulos, A. T., and Swanson, C. D., "Parallel Iterative S-Step Methods for Unsymmetric Linear Systems," *Parallel Computing*, Vol. 22, No. 5, 1996, pp. 623–641.

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

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

[20]Anderson, W. K., Thomas, J. L., and van Leer, B., "A Comparison of Finite Volume Flux Vector Splittings for the Euler Equations," AIAA Paper 85-0122, Jan. 1985.

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

[22]Hestenes, M. R., and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, Vol. 49, No. 6, 1954, pp. 409–435.

[23]Dembo, R. S., Eisenstat, S. C., and Steighaug, T., "Inexact Newton Methods," *SIAM Journal on Numerical Analysis*, Vol. 19, No. 2, 1982, pp. 400–408.

[24]Sonneveld, P., "CGS: A Fast Lanzos-Type Solver for Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 10, No. 1, 1989, p. 36.

[25]Van der Vorst, H. A., "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 2, 1992, pp. 631–644.

[26]Freund, R. W., "A Transpose-Free Quasi-Minimum Residual Algorithm for Non-Hermitian Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 14, No. 2, 1993, pp. 470–482.

[27]Eisenstat, S. C., Elman, H. C., and Schultz, M. H., "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations," *SIAM Journal on Numerical Analysis*, Vol. 20, No. 2, 1983, pp. 345–357.

[28]Axelsson, O., "A Generalized Conjugate Gradient, Least Squares Method," *Journal of Numerical Mathematics*, Vol. 51, No. 2, 1987, pp. 209–227.

[29]Vinsome, P. K. W., "ORTHOMIN, an Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations," Society of Petroleum Engineers of the American Inst. of Mining, Metallurgical, and Petroleum Engineers, Rept. SPE 5729, Richardson, TX, 1976.

[30]Wissink, A. M., Lyrintzis, A. S., and Chronopoulos, A. T., "Efficient Iterative Methods Applied to the Solution of Transonic Flows," *Journal of Computational Physics*, Vol. 123, No. 31, 1996, pp. 379–396.

[31]Chronopoulos, A. T., "S-Step Iterative Methods for (Non)Symmetric (In)Definite Linear Systems," *SIAM Journal on Numerical Analysis*, Vol. 28, No. 6, 1991, pp. 1776–1789.

[32]Strawn, R. C., Biswas, R., and Lyrintzis, A. S., "Helicopter Noise Predictions Using Kirchhoff Methods," *Journal of Computational Acoustics*, Vol. 4, No. 3, 1996, pp. 321–338.

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

D. S. McRae
Associate Editor