A PARALLEL NEWTON-KRYLOV METHOD FOR ROTORCRAFT FLOWFIELD CALCULATIONS

Andrew M. Wissink*
MCAT, Inc., NASA Ames Research Center, Moffett Field, CA, 94035

Anastasios S. Lyrintzis†
School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907

Anthony T. Chronopoulos‡
Computer Science, Wayne State University, Detroit, MI 48202

Abstract

This paper explores the use of Krylov subspace iterative methods for implicit solution of rotary-wing flowfields on parallel computers. A Newton-Krylov scheme is proposed which couples conjugate gradient-like iterative methods within the baseline structured-grid Euler/Navier-Stokes flow solver TURNS (Transonic Unsteady Rotor Navier Stokes). Two Krylov methods are studied, Generalized Minimum Residual (GMRES) and Orthogonal s-Step Orthomin (OSOmin). Preconditioning is performed with a parallelized form of the Lower Upper-Symmetric Gauss Seidel (LU-SGS) operator. The scheme is implemented on the IBM SP2 multiprocessor and applied to three-dimensional computations of a rotor in forward flight. The main benefit of the Newton-Krylov scheme is found to be a higher level of time-accuracy in implicit time-stepping. This increases the allowable timestep for time-accurate unsteady calculations, yielding a reduction in the overall solution time.

Introduction

Accurate numerical simulation of the aerodynamics and 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[6-13] 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 where traditional methods exhibit slow convergence, which can occur with very fine viscous grids, certain turbulence models, and with multiple grids. Large memory requirements are 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 very 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.

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[14] and a relatively new scheme called Orthogonal s-Step Orthomin (OSOmin)[15]. They are applied in a matrix-free inexact Newton formulation within the baseline Transonic Unsteady Rotor Navier Stokes (TURNS) code[2,3]. In earlier work[5], an efficient parallel implementation of the implicit Lower Upper-Symmetric Gauss Seidel operator[16] in TURNS was introduced. This operator is used here for preconditioning the Krylov meth-

*Research Scientist, Member AIAA
†Associate Professor, Associate Fellow AIAA
‡Associate Professor

ods. The Newton-Krylov scheme is coded with Message Passing Interface (MPI) message passing and implemented on the IBM SP2 multi-processor. All calculations are restricted to the Euler equations using a non-lifting 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 (Transonic Unsteady Rotor Navier Stokes)[2,3]. TURNS 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 Army, the code is used by various universities and the four major U.S. helicopter companies. The excellent predictive capabilities of TURNS 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

$$\partial_\tau q + \partial_\xi E + \partial_\eta F + \partial_\zeta G = \frac{\epsilon}{Re}\partial_\zeta S \qquad (1)$$

where q is the vector of conserved quantities, E, F, and G, are the inviscid flux vectors, and 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 $\zeta = \zeta(x,y,z,t)$ where the coordinate system x, y, z, t is attached to the blade. TURNS is run in Euler mode (i.e. $\epsilon = 0$) for all calculations presented in this paper.

The inviscid fluxes are evaluated using Roe's upwind differencing[17] 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[18]. Flux limiters are applied so the scheme is Total Variation Diminishing (TVD).

The implicit operator used in TURNS for time-stepping in both steady and unsteady calculations is the Lower-Upper symmetric Gauss-Seidel (LU-SGS) operator of Yoon and Jameson[16]. This operator takes the form

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

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

$$f(q^n) = \delta_\xi E^n + \delta_\eta F^n + \delta_\zeta G^n \qquad (3)$$

The factors D, L, and U are diagonal, lower, and upper tridiagonal matrices, respectively, determined using 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 timestep. A two-step symmetric Gauss Seidel scheme is employed for solution of Eq. (2).

For unsteady time-accurate calculations with LU-SGS, the factorization error is reduced by applying inner relaxation iterations. Using the solution at time level n, the initial condition is set $q^{n+1,0} = q^n$ and LU-SGS is applied to solve the following equation in each inner iteration

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

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

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

Hybrid LU-SGS

An efficient approach for parallelizing the LU-SGS implicit algorithm in TURNS has been introduced by the authors in earlier work[5]. The approach couples a standard domain decomposition implementation of LU-SGS for on-processor computations with the multiple sweep point-relaxation approach of the Data-Parallel Lower Upper Relaxation (DP-LUR) algorithm of Candler and Wright et. al.[19] for efficient inter-processor communications. Because the new approach combines ideas from both of these algorithms, it is referred to as the *hybrid* LU-SGS operator. The algorithm is as follows:

Algorithm: Hybrid LU-SGS

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

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

Communicate $\Delta q^{(i-1)}$ data at processor borders to neighboring processors.

Set $\Delta q^{(i)} = \Delta q^{(i-1)}$ at borders

Perform LU-SGS sweeps locally on each processor, computing $\Delta q^{(i)}$ over each subdomain

End Do

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

The hybrid LU-SGS implicit algorithm can be implemented in parallel in the same way as one would implement an explicit scheme. That is, the global domain is divided into subdomains and each processor operates on its own data. Only nearest-neighbor communication is required between the subdomains. Previously reported results[5] indicate the method achieves good parallel performance and maintains nearly identical convergence to the original LU-SGS operator with $i_{sweep} = 2$.

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 timestep

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

where $\Delta q^n = q^{n+1} - q^n$ and $f(q^n)$ 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 implicit timestep. However, Newton's method in its exact form is not applicable to most CFD problems of interest because the CPU time and storage required to exactly solve 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[21] 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 employed in this work is deferred to the next section.

Formation and storage of the Jacobian term, $\frac{\partial f}{\partial q}$, in Eq. (5) can be difficult and costly. Krylov solvers have the nice property that the Jacobian matrix is only used 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 w):

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

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. $f(q+\varepsilon w)$) 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 Eq. (6) can affect the nonlinear convergence of the method if not 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 q and w are comparably scaled, ε should ideally be near the square root of the machine roundoff, $\sqrt{\varepsilon_{mach}}$, which is 10^{-7} to 10^{-8} in double precision accuracy. The entries in the q vector are non-dimensionalized such that each entry has a value of approximately unity. The w vector is scaled within the iterative methods such that its root-mean-square is approximately unity, so each entry has a value of about $1/\sqrt{N}$ (N is the dimension of the vector). Thus, a simple yet accurate determination of ε is

$$\varepsilon = \sqrt{N \cdot \varepsilon_{mach}} \qquad (7)$$

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 in order to maintain convergence in the nonlinear solution. Dembo et al.[21] have proven that the nonlinear iterations will converge as long as the linear solution accuracy is at least

$$\|f(q^n) + f'(q^n)\Delta q^n\|_2 \leq \eta \|f(q^n)\|_2 \qquad (8)$$

where $0 < \eta \leq 1$. 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 timestep, sub-iterations of the iterative solver are performed until Eq. (8) is satisfied. A maximum of 20 sub-iterations is specified, 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 conjugate gradient (CG) method[21]. The convergence of CG is only assured 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 CG have been proposed for nonsymmetric systems. These nonsymmetric generalizations can be divided into two main categories, Lanczos-based methods and Arnoldi-based methods.

Lanczos-based methods include the Conjugate Gradient Squared (CGS)[22] method, stabilized variants of the Biconjugate-gradient method (Bi-CGSTAB)[23], and methods based on the Quasi-Minimum Residual idea (QMR)[24]. The approach used in deriving these methods from 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 to become erratic which can negatively affect the nonlinear convergence. Also, Lanczos-based methods require the transpose of the Jacobian (i.e. $A^T A$) for matrix-vector multiplies. Computation of A^T requires an 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 Generalized Minimum Residual (GMRES)[14] method, Generalized Conjugate Residual[25], and Orthomin[26]. 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 all direction vectors in the Krylov subspace to be stored so 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[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[14]. The application of GMRES in the context of nonlinear CFD problems is described in detail in a number of references[6,10,11,13,28] and the interested reader is deferred to these for a more thorough description. 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 Eq. (8) after reaching the fixed Krylov dimension, the method is restarted using the current solution as the initial guess.

The second iterative method used is the Orthogonal s-Step Orthomin (OSOmin) method of Chronopoulos and Swanson[15]. The so called "s-step" class of iterative methods are 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 [27] for a more general discussion of s-step methods). In 1991, Chronopoulos[27] introduced an s-step version of the classical nonsymmetric Orthomin(k) method. This version was modified to maintain orthogonality between the different s directions using 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 $s = m$ (see [27]).

Both GMRES and OSOmin are capable of solving nonsymmetric linear systems with symmetric part (i.e. $\frac{A+A^T}{2}$) positive definite (all positive eigenvalues). In earlier work[28], the authors showed 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 GMRES and OSOmin are about the same.

Preconditioning

The convergence rate of Krylov solvers is very 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.

Table 1: Storage requirements - N = no. gridpoints $\times$ 5 (no. dependent variables in 3D).

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

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 timestep n.

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

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 are those based on incomplete factorizations (e.g. Incomplete Lower Upper factorization - ILU). Ajmani[8] found the LU-SSOR method of Yoon and Jameson[16] (to which LU-SGS is a subset) to be more efficient than ILU 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 layed out on an array of processors using a Single Program Multiple Data (SPMD) 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 global values of the data each holds. Border data is communicated between processors, and a single layer of ghost-cells stores this communicated data. The Message Passing Interface (MPI) software routes communication between the processor subdomains.

There are essentially four main steps of the inexact Newton algorithm; 1) explicit flux evaluation using Roe-upwinded third-order accurate spatial discretization to form the right-hand-side vector, 2) preconditioning using hybrid LU-SGS, 3) implicit solution by the Krylov subspace solver, and 4) explicit

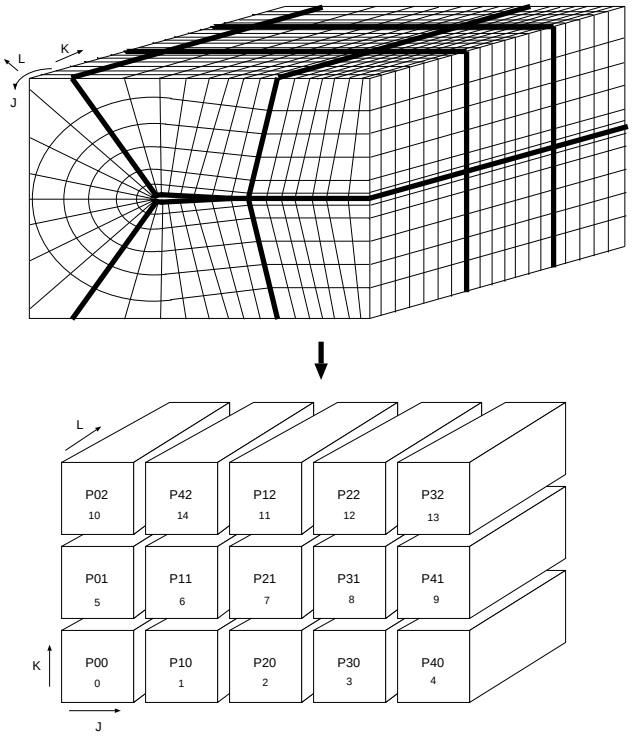

Figure 1: Partitioning the three-dimensional domain on a two-dimensional array of processors.

application of boundary conditions. The communication required in step 1) is straightforward. After the flux vectors are determined using 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 using 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 earlier. The communication pattern for this step is nearest-neighbor and 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 $2\log_2 p$ 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 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 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. By breaking the domain 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 which can significantly reduce parallel performance on large numbers of processors. While 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.

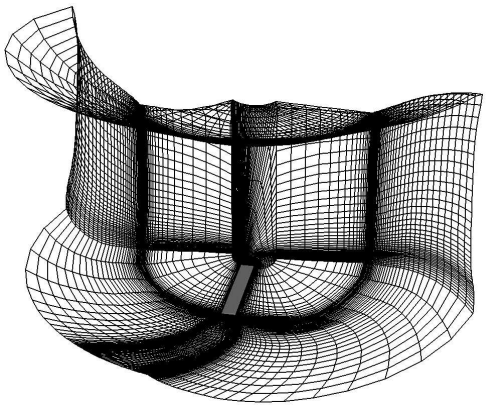

Figure 2: $135 \times 50 \times 35$ C-H grid

Computed Results

The parallelized inexact Newton implementation of TURNS is tested on the 160 node IBM SP2 at NASA Ames Research Center. The scheme is used to compute the quasi-steady (i.e. blade-fixed) and unsteady flowfield 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 non-lifting test case.

The flow is computed about a two-bladed symmetric untwisted Operational Load Survey (OLS) helicopter blade rotating with tip Mach number $M_{tip} = 0.665$ 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 1/7-scale model of the main rotor for the 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 TURNS for this particular test case have been evaluated in other studies by Strawn et al.[29] so this investigation will focus only on 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 [30].

Quasi-Steady

The nonlinear convergence with the inexact Newton scheme for a quasi-steady calculation with blade azimuth angle at $\psi = 0^o$ is shown in Figs. 3 and 4. Both figures show the convergence of the L2-norm of the residual ($||f(q)||_2$) vs. timesteps and vs. wallclock time on 19 processors of the SP2. Figure 3 shows results using the nonlinear convergence criterion in Eq. (8) with $\eta = 0.95$ (i.e. multiple iterations of the Krylov method applied at each timestep until the criteria is met) whereas Fig. 4 shows the results using only a single iteration of the Krylov method at each timestep. The inexact Newton cases are compared against the baseline case using 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 the 19 processor case shown. The maximum residual ($||f(q)||_\infty$) was also determined and showed similar results.

The hybrid LU-SGS method uses $i_{sweep} = 2$ because this was found in [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 three and five (that is, $m = 3, 5$ in GMRES and $s = 3, 5$ in OSOmin) because previous results[30] 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 one so the total storage costs for the Newton-GMRES and Newton-OSOmin comparison is essentially the same.

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 timestep than when multiple iterations of the Krylov method coupled with the nonlinear convergence criteria in Eq. (8) is 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

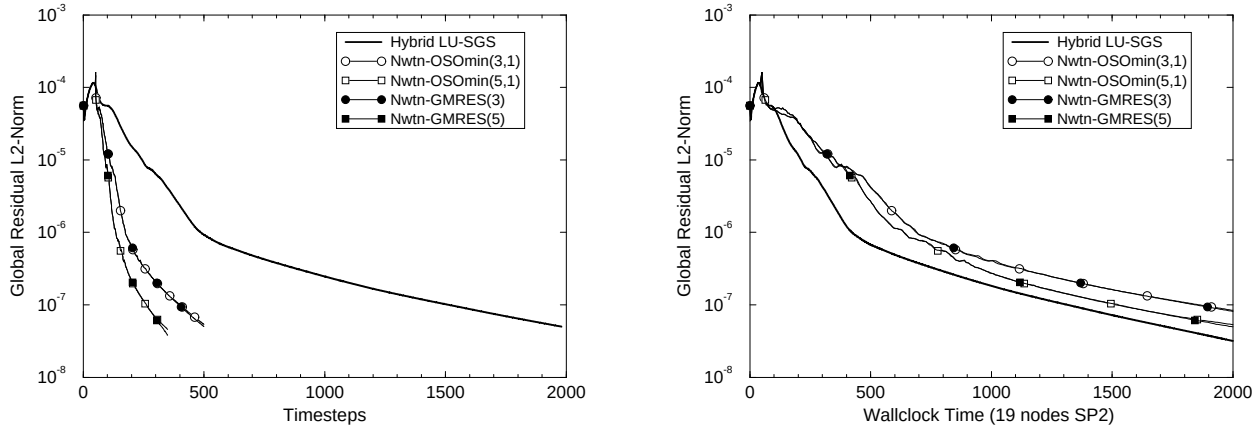

Figure 3: Convergence of Newton-Krylov method (compared to baseline hybrid LU-SGS method) on 19 processors of the IBM SP2, with nonlinear convergence Eq. (8) with $\eta = 0.95$ enforced at each timestep.

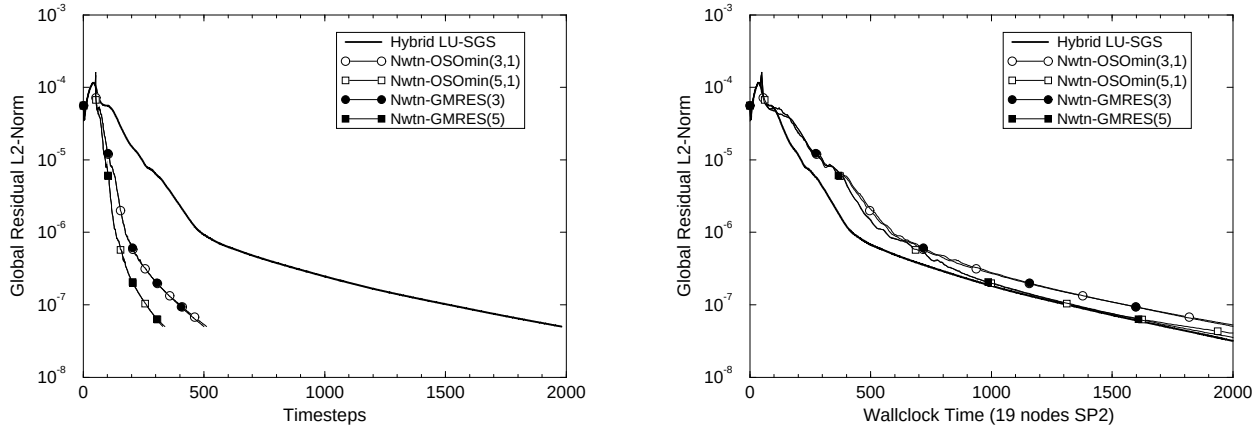

Figure 4: Convergence of Newton-Krylov method with only a single iteration of Krylov solver at each timestep.

every linear iteration, which is used only to determine the residual vector in order to whether or not the nonlinear convergence criteria has 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 timestep can yield a noticeable reduction in efficiency. A more detailed study[30] showed no performance gains 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 iteration increases by about the same factor as the reduction in number of iterations. For the forced nonlinear convergence case in Fig. 3, the Newton-Krylov methods show slightly worse efficiency than hybrid LU-SGS. However, with the single sub-iteration case in Fig. 4, the efficiency is slightly worse in the initial timesteps but becomes about the same as hybrid LU-SGS as the solution converges. Both GMRES and OSOmin show nearly identical results with the same Krylov dimension.

The parallel performance of methods is reported in Table 2. Shown are the average time per timestep, percentage communication, and parallel speedup for the baseline and Newton-Krylov methods on 4, 8, 19, 57, and 114 processors of the IBM SP2. The percentage communication is determined by timing all routines that invoke communication (any MPI routines) and comparing with the total average time per timestep. Parallel speedups are determined by comparing the average time per timestep to the 4 processor case.

Overall, all of the methods give comparable par-

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

	Time/Iter	% Comm	Speedup
4 Processors			
Hybrid LUSGS	4.07 sec	2.4%	1
Nwtn-GMRES(3)	18.78 sec	2.5%	1
Nwtn-GMRES(5)	26.16 sec	2.2%	1
Nwtn-OSOmin(3,1)	18.58 sec	2.1%	1
Nwtn-OSOmin(5,1)	26.35 sec	2.2%	1
8 Processors			$opt = 2$
Hybrid LUSGS	2.17 sec	4.6%	1.87
Nwtn-GMRES(3)	10.65 sec	4.1%	1.76
Nwtn-GMRES(5)	14.92 sec	4.2%	1.75
Nwtn-OSOmin(3,1)	10.68 sec	4.2%	1.74
Nwtn-OSOmin(5,1)	14.94 sec	4.8%	1.76
19 Processors			$opt = 4.75$
Hybrid LUSGS	.874 sec	5.1%	4.66
Nwtn-GMRES(3)	4.14 sec	5.4%	4.54
Nwtn-GMRES(5)	5.81 sec	5.4%	4.51
Nwtn-OSOmin(3,1)	4.13 sec	5.3%	4.50
Nwtn-OSOmin(5,1)	5.82 sec	5.4%	4.52
57 Processors			$opt = 14.25$
Hybrid LUSGS	.307 sec	8.9%	13.25
Nwtn-GMRES(3)	1.45 sec	9.7%	12.95
Nwtn-GMRES(5)	2.05 sec	10.1%	12.76
Nwtn-OSOmin(3,1)	1.42 sec	9.6%	13.08
Nwtn-OSOmin(5,1)	1.97 sec	9.9%	13.37
114 Processors			$opt = 28.5$
Hybrid LUSGS	.173 sec	11.9%	23.52
Nwtn-GMRES(3)	.885 sec	13.5%	21.22
Nwtn-GMRES(5)	1.23 sec	13.2%	21.26
Nwtn-OSOmin(3,1)	.823 sec	12.3%	22.58
Nwtn-OSOmin(5,1)	1.19 sec	13.4%	22.14

allel performance. There are no significant differences in the parallel speedup, although the baseline method (hybrid LU-SGS) and Newton-OSOmin show slightly better speedups than Newton-GMRES 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 show similar performance but there are a few subtle differences. On lower numbers of processors (i.e. 4 and 8), Newton-OSOmin requires slightly more time per iteration than Newton-GMRES because OSOmin requires slightly more work. However, OSOmin is found to achieve slightly better parallel on larger numbers of processors. Hence, the time per iteration of Newton-OSOmin is slightly faster than Newton-GMRES on 114 processors.

The measured execution rates of the code on various processors of the SP2 applied to this problem are shown in Fig. 5. The Megaflop rate for each

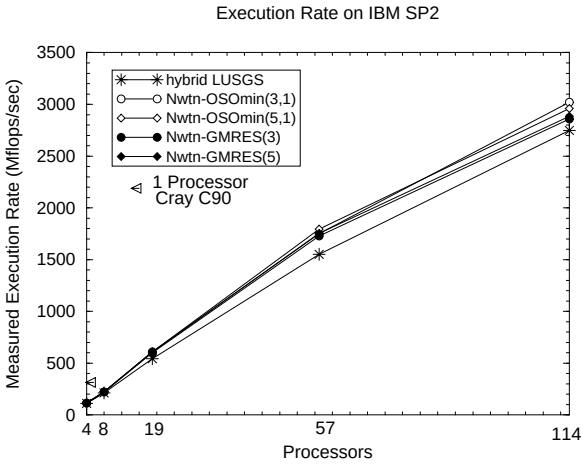

Figure 5: Execution rate attained on various processors of the SP2 for 236K gridpoint problem.

processor partition is measured with IBM's parallel hardware performance monitor (phpm) 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 measured rate on the C90 using Cray's hardware performance monitor is slightly different for each method but is shown as a single averaged point in Fig. 5 for convenience (actual rates on the C90 are 320 Megaflops for the baseline TURNS code, 340 Megaflops for Newton-GMRES, and 360 Megaflops for Newton-OSOmin). The Newton-Krylov scheme shows slightly better Megaflop 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 focussed 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 employed, making the left and 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 sub-iterations of the standard LU-SGS method at each

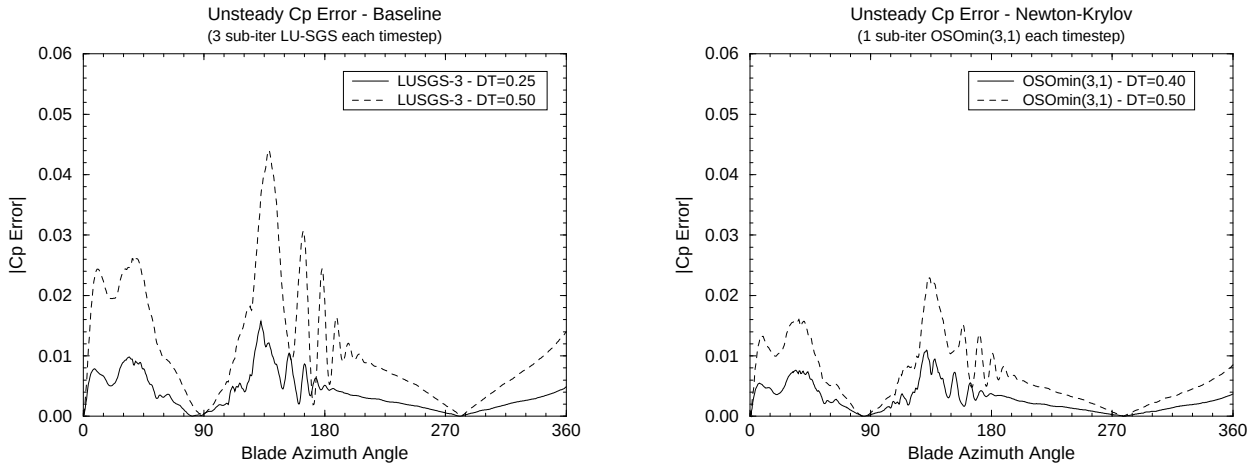

Figure 6: Unsteady error in Cp value at a representative point on the blade (1/4 chord, r/R=0.8). Calculation of 1 rev using baseline method with three sub-iterations of hybrid LU-SGS (*left*) and Newton-OSOmin(3,1) (*right*) (results with GMRES(3) identical to OSOmin(3,1)).

timestep, an time-accurate unsteady solution can be obtained using a timestep that corresponds to 1/4 degree of blade revolution per timestep ($\Delta\psi = 0.25^0$). We seek to match this result with the Krylov methods and compare the performance.

First, an unsteady solution is run with a very small timestep that corresponds to 1/10 degree azimuth per timestep ($\Delta\psi = 0.10^0$). The baseline hybrid LU-SGS method with three sub-iterations at each timestep is used for this run. The time-varying pressure coefficient is recorded at a representative location on the blade (one-quarter chord and r/R=0.80). Then, cases are run with larger timesteps and the resulting unsteady pressure coefficients are compared to the $\Delta\psi = 0.10^0$ result to determine the error.

Figure 6 shows the pressure coefficient error using the baseline and inexact Newton method with different timesteps. The left side of the figure shows the error resulting from a timestep of $\Delta\psi = 0.25^0$ and $\Delta\psi = 0.50^0$ with three sub-iterations of LU-SGS at each timestep (denoted LUSGS-3 in the figure). The right side of the figure shows the errors with timesteps of $\Delta\psi = 0.40^0$ and $\Delta\psi = 0.50^0$ using Newton-OSOmin(3,1) with a single iteration of OSOmin(3,1) at each timestep. It is apparent from the figures that the error from LUSGS-3 with $\Delta\psi = 0.25^0$ and Newton-OSOmin with $\Delta\psi = 0.40^0$ and $\Delta\psi = 0.50^0$ are comparable.

With LUSGS-3 using $\Delta\psi = 0.25^0$ considered the baseline case, Fig. 7 shows a closeup comparison of the errors with Newton-OSOmin using $\Delta\psi = 0.40^0$ and $\Delta\psi = 0.50^0$. The error with $\Delta\psi = 0.40^0$ is slightly lower than the baseline, and the error with $\Delta\psi = 0.50^0$ is slightly larger. All are very close, however. Newton-GMRES(3) was also tried and gives essentially identical results to Newton-OSOmin(3,1). Different spanwise locations were also tested (reported in [30]) and show similar results.

By allowing the use of larger timesteps with the same level of accuracy, the inexact Newton method can yield faster overall solution times. Table 3 lists the total time required to complete a full 360^o unsteady solution on 19 processors of the SP2 using three methods; 1) three sub-iterations of LU-SGS with a timestep of $\Delta\psi = 0.25^0$, 2) Newton-OSOmin(3,1) with $\Delta\psi = 0.40^0$, 3) Newton-OSOmin(3,1) with $\Delta\psi = 0.50^0$. The total time is determined from the time per timestep data for each method in Table 2. With $\Delta\psi = 0.40^0$, the total solution time with Newton-OSOmin is reduced by a about 5% over hybrid LU-SGS alone. With $\Delta\psi = 0.50^0$, it is reduced by about 30%. Similar results are achieved with Newton-GMRES. Thus, the inexact Newton algorithm is expected to yield wall-clock solution time savings on the order of 10%-20% for the same level of time-accuracy.

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

	Timestep	Soln. Time
Hybrid LUSGS (3 subits)	$\Delta\psi = 0.25^0$	3844 sec
Nwtn-OSOmin(3,1)	$\Delta\psi = 0.40^0$	3717 sec
Nwtn-OSOmin(3,1)	$\Delta\psi = 0.50^0$	2973 sec

Concluding Remarks

A parallelized Newton-Krylov algorithm is investigated for structured-grid calculations of the

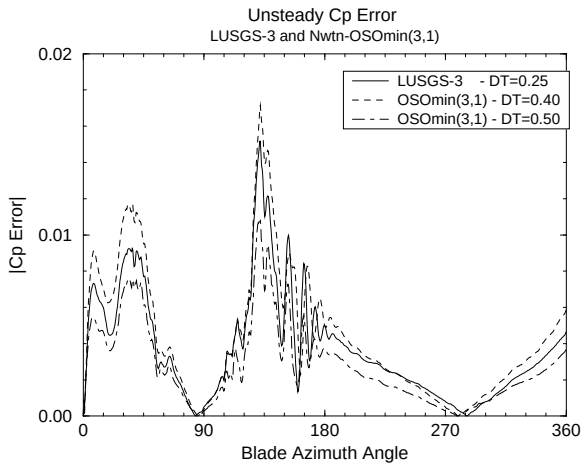

Figure 7: Detailed comparison of unsteady Cp error; LUSGS-3 with timestep $\Delta\psi = 0.25^0$ vs. Newton-OSOmin(3,1) with $\Delta\psi = 0.40^0$ and 0.50.

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 quasi-steady and unsteady three-dimensional Euler computations of a non-lifting helicopter blade on the IBM SP2 multiprocessor.

For quasi-steady calculations, the Newton-Krylov algorithm has a much faster convergence rate than the baseline approach (hybrid LU-SGS alone) but the wallclock solution time remains about the same for both. However, for time-accurate unsteady calculations, the Newton-Krylov algorithm maintains a higher degree of consistency in the implicit solution and consequently allows use of larger timesteps for the same level of accuracy. This, in turn, leads to reductions in the total solution time of 10%-20%.

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 very 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 show similar performance but OSOmin gives slightly better parallel speedups on larger processor partitions.

Although this work focussed on 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. multi-blocked or overset) would be an interesting extension of the present work.

Acknowledgments

The first author was supported by a NASA Graduate Student Fellowship from the High Performance Computing and Communications Program (HPCCP). Computer time on the IBM SP2 was provided by a grant from the Computational Aerosciences Division at NASA Ames. Additional computer time was also provided by a grant from the Pittsburgh Supercomputing Center. The third author acknowledges a CRAY/W.S.U. 1996-1997 grant and NSF support under grant CCR-9496327. The authors would like to acknowledge Dr. Roger Strawn for his advice during the course of this work, and Dr. 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 2nd AHS International Aeromechanics Specialists' Conference*, Vol. II, Bridgeport, CT, Oct. 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, Oct. 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, July 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, May 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, Nov. 1996, pp. 2276-2283.

[6] Wigton, L.B., Yu, N.J., and Young, D.P, "GMRES Acceleration of Computational Fluid Dynamics Codes," AIAA-85-1494, Presented at the AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, OH, July 1985.

[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, Dec. 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, Presented at the AIAA 32nd Aerospace Sciences Meeting, Reno, NV, Jan. 1994.

[9] Ajmani, K., Liou, M.-S., "Implicit Conjugate-Gradient Solvers on Distributed-Memory Architectures," AIAA 95-1065-CP, Presented at the 12th AIAA Computational Fluid Dynamics Conference, New Orleans, LA, 1995.

[10] Rogers, S.E., "Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations," *AIAA Journal*, Vol. 33, No. 11, Nov. 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, Oct. 1994, pp. 1978-1984.

[12] Cai, X.-C., Keyes, D.E., and Venkatakrishnan, V., "Newton-Krylov-Schwartz: An Implicit Solver for CFD," ICASE Report No. 95-87, 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," AIAA 95-1733-CP, Presented at the 12th AIAA Computational Fluid Dynamics Conference, New Orleans, LA, 1995.

[14] Saad, Y., and Shultz, M., "GMRES: a Generalized Minimum Residual Algorithm for Solving Non-symmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, 1996, pp. 856-869.

[15] Chronopoulos, A.T., and Swanson, C.D., "Parallel Iterative S-step Methods for Unsymmetric Linear Systems," *Parallel Computing*, Vol. 22/5, 1996, pp. 623-641.

[16] 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.

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

[18] 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.

[19] 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, Dec. 1994, pp. 2380-2386.

[20] Dembo, R.S., Eisenstat, S.C., and Steighaug, T., "Inexact Newton Methods," *SIAM Journal of Numerical Analysis*, **19**, 1982, pp. 400-408.

[21] Hestenes, M.R., and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," *J. Res. Natl. Bur. Standards*, Vol. 49, 1954, pp. 409-435.

[22] Sonneveld, P., "CGS: A Fast Lanzos-type Solver for Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comp.*, **10**, 1:36, 1989.

[23] Van der Vorst, H.A., "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comp.*, **13**, 2:631, (1992).

[24] Freund, R.W., "A Transpose-Free Quasi-Minimum Residual Algorithm for Non-Hermitian Linear Systems," *SIAM J. Sci. Comp.*, **14**, 2:470, 1993.

[25] Eisenstat, S.C., Elman, H.C., and Schultz, M.H., "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations," *SIAM Journal of Numerical Analysis*, Vol. 20, 1983, pp. 345-357.

[26] Vinsome, P.K.W., "ORTHOMIN, an Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations," *Society of Petroleum Engineers of AIME*, SPE 5729, 1976.

[27] Chronopoulos, A.T., "s-Step Iterative Methods for (Non)symmetric (In)definite Linear Systems," *SIAM J. Num. Anal.*, Vol. 28, No. 6, 1991, pp. 1776-1789.

[28] 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.

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

[30] Wissink, A.M., "Efficient Parallel Implicit Methods for Rotary-Wing Aerodynamics Calculations," PhD. Thesis, University of Minnesota, May 1997.