RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Efficient Helicopter Aerodynamic and Aeroacoustic Predictions on Parallel Computers

Andrew M. Wissink
Anastasios S. Lyrintzis
Roger C. Strawn
Leonid Oliker
Rupak Biswas

RIACS Technical Report 96.04 January 1996

Paper No. AIAA-96-0153, presented at the *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, Nevada, January 15–18, 1996

Efficient Helicopter Aerodynamic and Aeroacoustic Predictions on Parallel Computers

Andrew M. Wissink
Anastasios S. Lyrintzis
Roger C. Strawn
Leonid Oliker
Rupak Biswas

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 212, Columbia, MD 21044, (410) 730-2656

Work reported herein was supported by NASA via Contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA). Work was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035-1000.

EFFICIENT HELICOPTER AERODYNAMIC AND AEROACOUSTIC PREDICTIONS ON PARALLEL COMPUTERS*

Andrew M. Wissink†
Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN 55455

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

Roger C. Strawn§
US Army AFDD, Mail Stop 258-1, NASA Ames Research Center, Moffett Field, CA 94035

Leonid Oliker¶
RIACS, Mail Stop T27A-1, NASA Ames Research Center, Moffett Field, CA 94035

Rupak Biswas‖
RIACS, Mail Stop T27A-1, NASA Ames Research Center, Moffett Field, CA 94035

Abstract

This paper presents parallel implementations of two codes used in a combined CFD/Kirchhoff methodology to predict the aerodynamics and aeroacoustics properties of helicopters. The rotorcraft Navier-Stokes code, TURNS, computes the aerodynamic flowfield near the helicopter blades and the Kirchhoff acoustics code computes the noise in the far field, using the TURNS solution as input. The overall parallel strategy adds MPI message passing calls to the existing serial codes to allow for communication between processors. As a result, the total code modifications required for parallel execution are relatively small. The biggest bottleneck in running the TURNS code in parallel comes from the LU-SGS algorithm that solves the implicit system of equations. We use a new hybrid domain decomposition implementation of LU-SGS to obtain good parallel performance on the SP-2. TURNS demonstrates excellent parallel speedups for quasi-steady and unsteady three-dimensional calculations of a helicopter blade in forward flight. The execution rate attained by the code on 114 processors is six times faster than the same cases run on one processor of the Cray C-90. The parallel Kirchhoff code also shows excellent parallel speedups and fast execution rates. As a performance demonstration, unsteady acoustic pressures are computed at 1886 far-field observer locations for a sample acoustics problem. The calculation requires over two hundred hours of CPU time on one C-90 processor but takes only a few hours on 80 processors of the SP2. The resultant far-field acoustic field is analyzed with state-of-the-art audio and video rendering of the propagating acoustic signals.

Introduction

Accurate numerical simulation of the aerodynamic and aeroacoustic properties of rotorcraft in hover and forward flight is a complex and challenging problem. In recent years, three-dimensional unsteady Euler/Navier-Stokes methods [1–5] have been proven to accurately compute the aerodynamics and aeroacoustics properties in the region near the blade. The TURNS (Transonic Unsteady Rotor Navier Stokes) code, developed by Srinivasan et al. [3,4] at NASA Ames, is one such example. TURNS has been demonstrated in a number of studies to compute accurately the aerodynamic flowfield and near-field acoustics, both high-speed impulsive (HSI) and blade-vortex interaction (BVI) noise, of a helicopter rotor (without fuselage) in hover and forward flight in subsonic and transonic conditions [3-10]. The code has been applied in the framework of overset grids [6,7] and has been coupled with an unstructured grid adaptive method [8] for better resolution of flow features. Unlike more traditionally-used full potential methods that require use of a linear wake model to resolve the tip vortex, TURNS captures the tip vortex and the entire vortical wake as part of the overall flowfield solution.

In addition to the desire for high aerodynamic performance, modern helicopter designs also aim for low noise. The two types of noise that cause problems for helicopters are high speed impulsive (HSI), characterized by a strong acoustic disturbance occurring over a short period of time, and blade-vortex interactions

*Paper No. AIAA 96-0153, presented at the 34th AIAA Aerospace Science Meeting, Reno, NV, Jan. 1996.

†NASA Fellow, Student member AIAA
‡Associate Professor, Member AIAA
§Research Scientist, Member AIAA
¶Graduate Student, University of Colorado
‖Research Scientist, Member AIAA

(BVI's) which occur when the rotor blades interact with their vortical wake systems. Prediction of far-field noise is not trivial, since traditional CFD methods are too dissipative to carry the near-field pressure signals into the far field. Recent progress in the prediction of far-field helicopter noise have come from combined CFD/Kirchhoff methods. A CFD method, usually a Full Potential or Euler/Navier-Stokes solver, is used close to the rotor blades to capture the near-field acoustic nonlinearities. The solutions are then interpolated onto a surface that surrounds the CFD domain and a Kirchhoff integration is performed to carry the acoustic signal into the far field.

The advantage of the hybrid CFD/Kirchhoff methods is that they account for nonlinear acoustic propagation modeled by CFD close to the rotor blades, whereas away from the blades, a Kirchhoff integral avoids dissipation and dispersion errors associated with any CFD solver. By comparison, acoustic analogy methods use CFD information only on the blade surface and do not account for the nonlinearities (e.g. supersonic regions for transonically moving blades) near the blade.

The CFD/Kirchhoff approach was first demonstrated for rotorcraft applications in two-dimensional cases [11–13] and has since been applied in three-dimensions [14–19]. Kirchhoff surfaces have been applied in a rotating reference frame (rotating with the helicopter blades) [14,15,19] as well as a nonrotating frame [9,18], with the Kirchhoff surface enveloping the entire rotating hub and blades. Strawn et al. [20] showed both approaches give identical results. The Kirchhoff approach has shown a higher level of accuracy in predicting far field noise to more traditionally used methods. Baeder et al. [9] showed the Kirchhoff method to be more accurate than acoustic analogy without quadrapoles for hover HSI noise with tip Mach numbers higher than 0.7. Lyrintzis et al. [21] also found Kirchhoff's method to be more accurate than the acoustic analogy results in the WOPWOP code [22] for unsteady HSI noise, and about the same for low-speed BVI noise. A thorough review on the use of Kirchhoff's formula in computational aeroacoustics is given by Lyrintzis [23].

Although the combined CFD/Kirchhoff approach has been proven to give accurate results, it is computationally demanding. A three dimensional unsteady Navier-Stokes computation with TURNS, and a Kirchhoff computation at a large number of far-field observer points, can require many hours of CPU time on supercomputers of Cray-class.

Parallel computation offers the potential for cheaper and faster computations. Several vendors have released machines that utilize commodity RISC processors that are the same as those used in high-end workstations (e.g. IBM SP2, Intel Paragon, SGI Power Challenge). These machines generally attain better price/performance than vector processors and have peak execution rates that are several times faster than vector parallel machines like the Cray C-90. Some companies are beginning to utilize parallel processing in the form of networked workstations, that sit idle during off-hours, to attain supercomputer performance [24].

Parallel processing has not, in general, been well-received by the engineering community [25] due mainly to the difficulties in writing parallel programs and their lack of portability, and the extensive algorithm modifications that are required for efficient parallelism. The first problem has been circumvented somewhat with the advent of standardized parallel languages (e.g. Message Passing Interface - MPI, Parallel Virtual Language - PVM) that have made parallel codes portable and easier to implement. In this paper, we introduce algorithm modifications that make both the TURNS code and an existing Kirchhoff code parallelizable. The modifications to the existing code are small and relatively easy to implement.

Finally, although both the CFD and Kirchhoff solvers are applied in this paper to helicopter problems, the parallelization strategies are not unique to this application and could readily be used for other problems.

Computational Fluid Dynamics Model

The structured-grid Euler/Navier-Stokes solver TURNS [3,4] is used to compute the aerodynamic flowfield close to the helicopter rotor. This CFD code solves the Navier-Stokes equations about rotating helicopter blades. Since viscous effects are small for the cases considered in this paper, the TURNS code is run in the Euler mode. All nonlinear effects on the acoustic propagation are accurately modeled within the framework of the Euler equations.

Srinivasan [1] has modified the TURNS code to compute the aerodynamics of isolated BVI's. Recently, Baeder and Srinivasan [10] used this modified version of TURNS to compute the BVI noise of an isolated rotor blade interacting with an upstream-generated vortex. They showed good agreement with the experimental data of Caradonna et al. [26] for time-dependent surface pressures during the BVI event. This same version of the TURNS code is used for the calculations in this paper. However, an additional module has been added to evaluate and store the appropriate pressure information on the Kirchhoff surfaces. More detailed information about the TURNS algorithm is given in the remainder of this section.

The governing equations for the TURNS code are the unsteady, compressible, three-dimensional thin layer Navier-Stokes equations. These equations are applied in conservation form in a generalized body-conforming curvilinear coordinate system

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

where $\tau = t$, $\xi = \xi(x,y,z,t)$, $\eta = \eta(x,y,z,t)$, and $\zeta = \zeta(x,y,z,t)$. The coordinate system (x,y,z,t) is attached to the blade. The vector of conserved quantities

is Q, and the inviscid flux vectors E, F, and G are

$$Q = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \qquad E = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U H - \xi_t p \end{bmatrix} \tag{2}$$

$$F = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V H - \eta_t p \end{bmatrix} \quad G = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W H - \zeta_t p \end{bmatrix}$$

where $H = (e+p)$ and U, V, and W, are the contravariant velocity components (e.g. $U = \xi_t + \xi_x u + \xi_y v + \xi_z w$). The cartesian velocity components u, v, and w are defined in the x, y, and z directions, respectively. The quantities ξ_t, ξ_x, ξ_y, and ξ_z are the coordinate transformation metrics and J is the Jacobian of the transformation. The pressure p is related to the conserved quantities through the perfect gas equation of state

$$p = (\gamma - 1)\left\{e - \frac{\rho}{2}(u^2 + v^2 + w^2)\right\} \tag{3}$$

The viscous flux vector S is incorporated in the code but the calculations given in this paper are all inviscid (i.e. $\epsilon = 0$ in Eq. (1)) so the viscous terms are not described here. Details can be found in [3].

The inviscid fluxes are evaluated using Roe's upwind differencing [27] in all three directions. The use of upwinding obviates the need for user-specified artificial dissipation and improves the shock capturing in transonic flowfields. Third order accuracy is obtained using van Leer's MUSCL approach [28] and flux limiters are applied so the scheme is Total Variation Diminishing (TVD).

The final Euler discretized form of Eq. (1) in implicit delta form is

$$[I + h(\delta_\xi A^n + \delta_\eta B^n + \delta_\zeta C^n)]\,\Delta Q^n = -hRHS^n \tag{4}$$

where

$$RHS^n = \partial_\xi E^n + \partial_\eta F^n + \partial_\zeta G^n \tag{5}$$

I is the identity matrix, h is the timestep and $\Delta Q^n = Q^{n+1} - Q^n$. The 5×5 matrices A, B and C are the Jacobians of the flux vectors with respect to the conserved quantities (e.g. $A = \frac{\partial E}{\partial Q}$).

Implicit LU-SGS Operator

TURNS uses the two-factor LU-SGS (Lower-Upper Symmetric Gauss Seidel) algorithm of Yoon and Jameson [29] for the implicit time step. The LU-SGS algorithm has been used in a number of well-known CFD codes (e.g. INS3D [30], OVERFLOW [31]) primarily for it's stability properties. Classic implicit methods such as Beam-Warming approximate factorization have a large factorization error (of order Δt^3) which further restricts the size of the time step. The two-factor LU-SGS method has enhanced stability along with a reduction in factorization error (order Δt^2) that make it an attractive alternative.

LU-SGS resembles a typical LU factorization scheme with diagonal preconditioning to increase robustness. The scalar diagonal terms are obtained by use of approximate Jacobians, avoiding costly matrix inversions. First, the Jacobian terms A, B, and C in Eq. (4) are split into " + " and " − " parts, with positive parts constituting only the positive eigenvalues and negative parts constituting only the negative eigenvalues. The positive matrix is backward differenced and the negative matrix is forward differenced, as follows

$$\begin{aligned} A_\xi &= \delta_\xi^- A^+ + \delta_\xi^+ A^- \\ &= A_j^+ - A_{j-1}^+ + A_{j+1}^- - A_j^- \end{aligned} \tag{6}$$

This splitting ensures diagonal dominance. The Jacobians are then approximated using the following spectral approximation

$$A^\pm = s_A^\pm \frac{1}{2}(A \pm \rho_A I) \pm \varepsilon \rho_A I \tag{7}$$

where ρ_A is the spectral radius of A (in the ξ direction).

$$\rho_A = max\,[|\lambda_A|] = |U| + a\,|\nabla\xi| \tag{8}$$

ε is some small value (e.g. .001), and $s_A^\pm$ is defined as

$$s_A^\pm = \begin{cases} 1 & if \pm (U \pm a\,|\nabla\xi|) \geq 0 \\ 0 & otherwise \end{cases} \tag{9}$$

The same procedure is used in the η and ζ directions to form the B and C terms.

Substituting these approximate Jacobians into the implicit equation (4), the equation takes the form

$$LD^{-1}U\Delta Q^n = -hRHS^n \tag{10}$$

where

$$\begin{aligned} D &= I + h(\rho_A + \rho_B + \rho_C)_{j,k,l} \\ L &= D - h\left(A_{j-1}^+ + B_{k-1}^+ + C_{l-1}^+\right) \\ U &= D + h\left(A_{j+1}^- + B_{k+1}^- + C_{l+1}^-\right) \end{aligned} \tag{11}$$

D is a diagonal matrix, and the two-step LU decomposition can be performed by

$$\begin{aligned} L\Delta Q^* &= -hRHS^n \\ U\Delta Q^n &= D\Delta Q^* \end{aligned} \tag{12}$$

which can also be written as the following algorithm

Algorithm 1: LU-SGS

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

$$\delta Q^*_{j,k,l} = D^{-1}h\left[\begin{array}{l} -RHS^n + (A^+_{j-1}\delta Q^*_{j-1} \\ +B^+_{k-1}\delta Q^*_{k-1} + C^+_{l-1}\delta Q^*_{l-1}) \end{array}\right]$$

End Do

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

$$\delta Q_{j,k,l} = \delta Q^*_{j,k,l} - D^{-1}h\left[\begin{array}{l} A^-_{j+1}\delta Q^*_{j+1} + B^-_{k+1} \\ \delta Q^*_{k+1} + C^-_{l+1}\delta Q^*_{l+1} \end{array}\right]$$

End Do

Time Stepping

Time stepping in TURNS is done using the implicit LU-SGS operator. The LU-SGS scheme introduces a factorization error of $O(\Delta t^2)$ and is consequently not time-accurate. A time-accurate solution can be determined using inner iterations to relax the factorization error. Using the solution at time level n, the initial condition is set $Q^{n+1,0} = Q^n$ and the following equation is solved using LU-SGS

$$\left[LD^{-1}U\right]^{n+1,m}\Delta q = -h\left(\frac{Q^{n+1,m} - Q^n}{h} + RHS^{n+1,m}\right) \quad (13)$$

where RHS is the same as Eq. (5) and

$$\Delta q = Q^{n+1,m+1} - Q^{n+1,m} \quad (14)$$

Here, n refers to the time level and m to the iteration level. In our tests, three inner iterations were used at each timestep. Upon completion of the inner iterations, the solution at the next time level is $Q^{n+1} = Q^{n+1,m_{max}}$.

TURNS Parallel Implementation

For parallel implementation of TURNS, we wanted to use an approach that would require relatively minor changes to the existing code, since the code is relatively long (6000 lines), and we wanted the resulting code to be portable. For these reasons, a MIMD (i.e. requiring message passing) approach is chosen over a SIMD (Single Instruction Single Data) or data-parallel approach. Message passing codes are generally more portable to different parallel architectures (e.g. from massively parallel supercomputers to workstation clusters) and the message passing subroutine calls can be added to the existing Fortran 77 code, as opposed to rewriting the entire code in a High Performance Fortran type language (e.g. CMFortran). To ensure portability, a set of generic message passing subroutines is used. With this protocol, the specific message passing commands can be altered in one line of the code rather than throughout, making conversion to different message passing languages, such as PVM (Parallel Virtual Machine), a relatively short procedure.

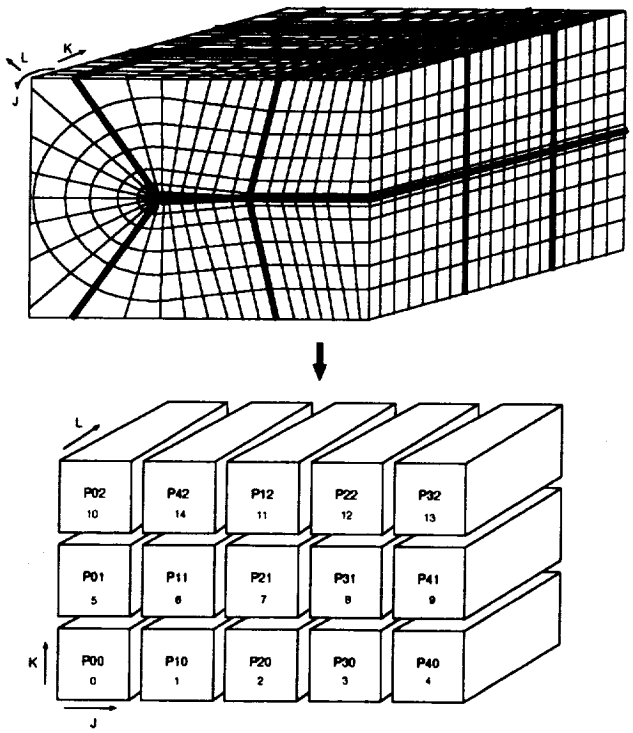

Figure 1: Partitioning the three-dimensional domain on a two-dimensional array of processors.

A domain decomposition strategy that preserves the original construct of the code is used for parallel implementation. The domain is divided in the wraparound and spanwise directions into subdomains. The normal direction is left intact, so the entire flowfield domain is layed out on a two-dimensional array of processors, as demonstrated in Fig. 1. Data is communicated between processors, and a single layer of ghost-cells is used to store this communicated data. Although this approach leads to non-square subdomains, and optimal parallelism is achieved with square subdomains, the normal direction is left undivided to prevent a load imbalance from occurring during the application of the boundary conditions at the C-plane. An averaging of data is performed between nodes lying on either side of the plane that requires communication of data across this plane. If the normal direction was divided, all processors not holding data on the C-plane would sit idle while the data is communicated. In the present implementation, all processors participate in this communication, eliminating the load imbalance.

There are essentially three main portions of the solution algorithm in TURNS. The first is the formation of RHS. The communication required for this step is trivial. 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 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 and constitutes a small percentage of the overall communication. The

second portion of the solution algorithm is application of the boundary conditions. This can be done local to each processor, with exception to the averaging of data across the C-plane, described above. The third part of the algorithm is the implicit solve using LU-SGS. This portion of the code is not trivial to implement in parallel.

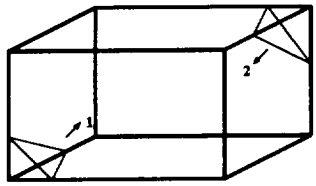

Figure 2: Domain sweeping strategy used by LU-SGS algorithm. Can vectorize on hyperplanes where $j+k+l = const.$

In the LU-SGS algorithm (Algorithm 1), the first step sweeps in the positive direction updating δQ^*. The second step then computes δQ by sweeping back through the domain in the opposite direction. This algorithm can be vectorized using a hyperplane approach, as outlined in ref. [32]. Vectorization is done across hyperplanes in which $j+k+l=const$, as outlined in Fig. 2. While the hyperplane approach leads to good vector execution rates, it is difficult to parallelize for two reasons; 1) the size of the hyperplanes vary throughout the grid, leading to load balancing problems, and 2) there is a recursion between the planes, leading to a large amount of communication.

Parallelization of the LU-SGS algorithm has been addressed by other researchers. Barszcz et al. [34] implemented the LU-SSOR algorithm, which is similar to LU-SGS, on a parallel machine by restructuring the data-layout using a skew-hyperplane approach. Although they were able to extract good parallelism with this approach the data-layout is complex and the restructuring of data on the left hand side in turn causes the right hand side layout to be skewed and extra communication is required. Several researchers have proposed modifications of the LU-SGS algorithm to make it more parallelizable. Candler and Wright [33–35] have investigated a modification called Data-Parallel LU Relaxation (DP-LUR), which has shown excellent results in a data-parallel environment. Wong et al. [36] have investigated a domain decomposition implementation of LU-SGS. For two-dimensional steady state reacting flow problems, they found that, while the convergence rate of the operator is reduced with the domain breakup, the affect is relatively weak (e.g. with 64 subdomains, the number of iterations increases by less than 20%).

In refs. [37,38], the DP-LUR algorithm is implemented in place of LU-SGS in TURNS and tested for three-dimensional rotorcraft calculations on the Thinking Machines CM-5 using a message-passing implementation. The DP-LUR method showed excellent parallelism with a low percentage of communication but the method required about three times the work of LU-SGS to maintain the same convergence rate.

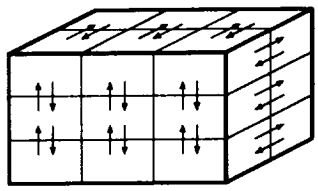

Figure 3: Jacobi Communication Strategy of the Hybrid algorithm. Load balanced parallelism with nearest neighbor communication.

We propose in ref. [37] a modification of DP-LUR that was found to be more efficient with the message passing approach. The original DP-LUR algorithm uses Jacobi sweeps in place of the Gauss-Seidel sweeps used in LU-SGS. This has the benefit that all computations can be done locally to each processor and very little communication is required. However, because Jacobi has a slower convergence rate than Gauss-Seidel, multiple sweeps of the domain (e.g. 3-6) are performed to maintain convergence, and the algorithm consequently requires more work. In our modified approach, the original Gauss-Seidel sweeps in LU-SGS are performed locally on each processor while retaining the Jacobi communication strategy used in DP-LUR for communications. Essentially, the modified algorithm entails using Symmetric Gauss-Seidel sweeps for on-processor computations and a Jacobi method for inter-processor communications. Figure 3 describes this idea.

The multiple sweep idea is retained from DP-LUR to improve robustness for cases where the breakup of the domain causes convergence difficulties. Because this modified approach retains properties of both DP-LUR and LU-SGS, we refer to it as a Hybrid parallel implementation of LU-SGS.

Algorithm 2: Hybrid LU-SGS

$$\delta Q^{(0)}_{j,k,l} = D^{-1} \cdot hRHS^n$$

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

Communicate border $\delta Q^{(i-1)}_{j,k,l}$ data to neighboring processors.

$$\delta Q^{*(i)}_{j,k,l} = \delta Q^{(i-1)}_{j,k,l}$$

Perform LU-SGS sweeps (Algorithm 1) locally on each processor, computing $\delta Q^{(i)}_{j,k,l}$

End Do

$$\delta Q_{j,k,l} = \delta Q^{(i_{sweep})}_{j,k,l}$$

The Hybrid algorithm is easily implemented in the domain decomposition strategy. Since original LU-SGS is used for on-processor computations, little code modification is required of the LU-SGS algorithm already in

TURNS. Message passing is used to communicate the border data amongst neighboring processors, and then the computations proceed locally. On a single processor (with 1 sweep), the Hybrid method is identical to the original LU-SGS algorithm.

Kirchhoff Integration Method

The CFD/Kirchhoff method consists of a CFD solution, which is done in this case using TURNS, near the rotor blades followed by a Kirchhoff integration to propagate the acoustic signals to the far field. Since viscous effects are small for the cases considered in this paper, the TURNS code is run in the Euler mode. All nonlinear effects on the acoustic propagation are accurately modeled within the framework of the Euler equations.

The Kirchhoff formulation from Farassat and Myers [39] is used to evaluate the acoustic pressure, P, at a fixed observer location, $\vec{x}$, and observer time, t. In the general case, the Kirchhoff surface, S, can deform and can also have arbitrary motion. The formula can be written as:

$$P(\vec{x},t) = \frac{1}{4\pi}\int_S \left[\frac{E_1}{|\vec{r}|(1-M_r)} + \frac{E_2 P}{r^2(1-M_r)}\right]_\tau \mathrm{d}S. \tag{15}$$

The expressions for E_1 and E_2 are given as:

$$E_1 = (M_n^2-1)P_n + M_n\vec{M}_t\cdot\nabla_2 P - \frac{M_n\dot{P}}{a_\infty} + \frac{(\dot{n}_r - \dot{M}_n - \dot{n}_M)P + (\cos\theta - M_n)\dot{P}}{a_\infty(1-M_r)} + \frac{\dot{M}_r(\cos\theta - M_n)P}{a_\infty(1-M_r)^2}, \tag{16}$$

$$E_2 = \frac{(1-M^2)(\cos\theta - M_n)}{(1-M_r)^2}. \tag{17}$$

The Kirchhoff surface is assumed to be moving with Mach number $\vec{M}$. The distance between a point on the Kirchhoff surface and the observer is given by $|\vec{r}|$. M_n and M_r are the components of $\vec{M}$ along the local surface normal, $\vec{n}$, and the radiation direction, $\vec{r}$. $\vec{M}_t$ is the Mach number tangent to the Kirchhoff surface, P_n is the derivative of P along the surface normal, and $\nabla_2 P$ is the gradient of the pressure on the Kirchhoff surface. The freestream speed of sound is assumed uniform at a_∞, and θ is the angle between $\vec{n}$ and $\vec{r}$. The dot over any quantity denotes its partial derivative with respect to time, t. The terms $\dot{n}_r$ and $\dot{n}_M$ are defined as $\dot{\vec{n}}\cdot\vec{r}$ and $\dot{\vec{n}}\cdot\vec{M}$, respectively. The simplified form for E_2 in Eq. (17) is taken from Myers and Hausmann [40].

Figure 4 shows a representative Kirchhoff surface in the rotating reference frame. The surface moves with the rotor blade and coincides exactly with a portion of the CFD mesh. The pressure and its spatial and temporal derivatives on the Kirchhoff surface are computed by the TURNS code and stored at every azimuthal degree around the rotor disk. The pressure gradients are computed using the same coordinate transformation metrics that are used to evaluate the fluxes in the flow solver. The chain rule is used to convert the pressure gradients from computational space to inertial space for the Kirchhoff integration.

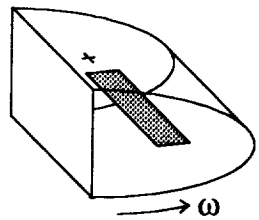

Figure 4: The rotating Kirchhoff surface is attached to the helicopter blade.

The Farassat and Myers [39] Kirchhoff formulation in Eqs. (15–17) allows both rotation and translation of the Kirchhoff surface. Note that the entire integral in Eq. (15) must be evaluated at the time of emission for the acoustic signal, τ, which is sometimes called the retarded time. The calculation of the proper retarded time values for each point on the surface is the most difficult step in the evaluation of the Kirchhoff integral. Extensive details are given in Ref. [20].

One major problem with this rotating-surface Kirchhoff method is that Eq. (14) requires the surface to move subsonically. This requirement places a restriction on the outer radial location for the rotating surface. If it is too far from the rotor blade tip, it will have supersonic motion, which is not allowed. On the other hand, if the surface is too close, the acoustic solution may be inaccurate. To overcome this problem, a second implementation uses a nonrotating Kirchhoff surface that completely surrounds the spinning rotor blades. Once the rotational motion is removed from the Kirchhoff surface, its translational motion is always subsonic. However, one disadvantage of this formulation is that it requires interpolations from the rotating-grid CFD solution onto the nonrotating Kirchhoff surface.

Reference [20] presents results for both types of Kirchhoff surfaces. Excellent agreement with experimental data was obtained for both HSI and BVI noise with each Kirchhoff method. Figure 5 shows one of the comparisons from that paper for HSI noise from the Army's AH-1 helicopter with the OLS rotor blades. Computed far-field noise shows excellent agreement with the experimental data which is taken from Schmitz et al. [41].

Kirchhoff Parallel Implementation

Implementation of the rotating and nonrotating Kirchhoff integrations on parallel computers is straightforward due to the regular nature of the computations. The Kirchhoff surface is divided into patches and each

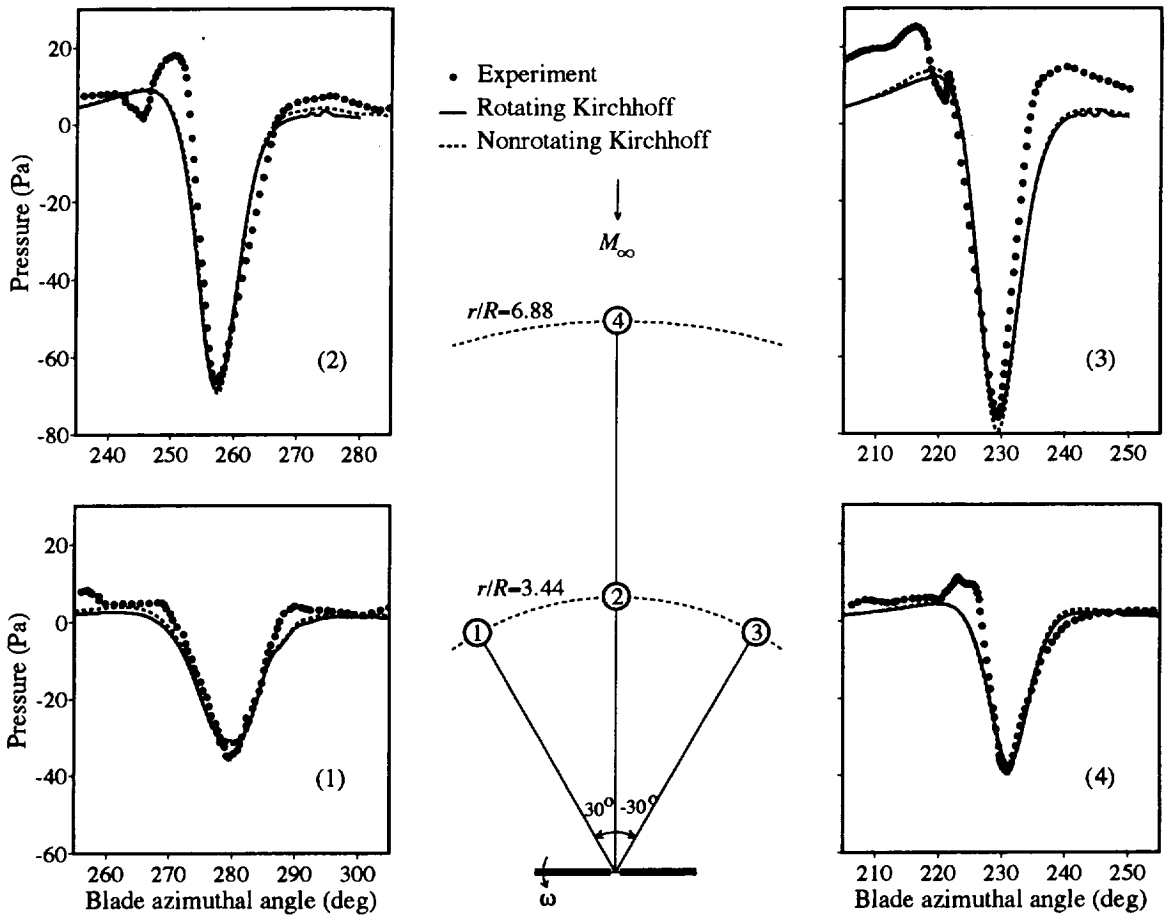

Figure 5: Comparison of acoustic pressures with experimental data at four different microphone locations for an AH-1 blade with $M_{at} = 0.837$. All microphones are in the plane of the rotor.

patch is assigned to a processor. Virtually no communication is needed among processors except for a final summation of the integral contributions from each surface patch. In this paper, parallel results are presented only for the rotating Kirchhoff code. Work continues on implementing the nonrotating Kirchhoff code and results will be presented in reference [42].

The Kirchhoff integration is implemented using a MIMD message passing paradigm and is portable to any distributed-memory architecture that supports FORTRAN 90 and the MPI library. Details of the implementation are as follows. The Kirchhoff surface and associated pressure files are stored as two dimensional arrays of size $(imax, jmax)$, which must be partitioned and distributed among the processors. A "host" processor is designated to read and scatter the data to the working set, P, of processors. Each processor receives a contiguous one-dimensional strip of $(imax \times jmax)/P$ data elements, plus an additional overlapped section of $imax$ elements. This allows local, communication-free, nearest-neighbor computations. Note that the data could have also been distributed in two-dimensional blocks of size $(imax/P, jmax/P)$, thereby reducing the space required for the overlapped elements while adversely affecting cache performance and complicating the local index calculations. Due to the relatively small size of the rows, the former less complex approach was taken.

Results

Both TURNS and the Kirchhoff code are implemented on the 160 node IBM SP2 multiprocessor located at NASA Ames. The standard MPI (Message Passing Interface) library is used for message passing in the codes. The TURNS source code is written in Fortran 77 while the Kirchhoff code is in Fortran 90. The codes are combined to compute the near-field CFD solution and far-field acoustic patterns for a nonlifting symmetric OLS blade rotating with advancing tip Mach number $M_{at} = 0.837$ (tip Mach number 0.665 and freestream Mach number 0.172). 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. The following sections present results of the CFD solution using TURNS and the far-field noise prediction using the Kirchhoff method.

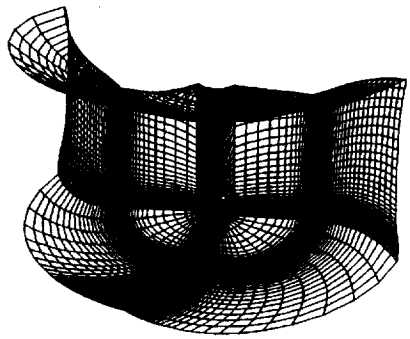

Figure 6: Upper half of the $135 \times 50 \times 35$ C-H type grid used for OLS airfoil calculations on the CM-5.

TURNS Results

The TURNS code is run in Euler mode, using a $135 \times 50 \times 35$ C-H type grid, with the domain extending eight chords in all directions. The upper half of the grid is shown in Fig. 6.

Before an unsteady computation can be performed, a quasi-steady starting solution must first be computed. In the quasi-steady case, the blade is held fixed at zero degrees azimuth while the tip and freesteam conditions are the set to that for the corresponding unsteady solution. A spatially varying timestep, described in [33], is used to generate the quasi-steady solution. Since the solution is not time-accurate, the quasi-steady case provides a convenient way to quantify the convergence qualities of the Hybrid modification of LU-SGS on different processor partitions.

TURNS is run for the quasi-steady case on 1, 4, 8, 19, 57, and 114 processors. For the 4–8 processor cases, the wraparound direction in the grid is left intact while the spanwise direction is subdivided into 4 and 8 subdomains, respectively. For the 19, 57, and 114 processor cases, the wraparound direction is subdivided into a 19 subdomains, and the spanwise direction is subdivided into 3 and 6 subdomains, respectively. Due to the specific grid used, we were unable to use a more variable distribution of processors in the wraparound direction. The only factors of the 133 points in the wraparound direction are 7 and 19. With 7 subdomains, a processor boundary is placed at the trailing edge of the blade, which results in adverse convergence difficulties. Thus, we were forced to use only 19 subdomains in the wraparound direction. Use of other meshes in future calculations should allow more versatility to the choice of processor subdomains.

The convergence criteria used for the quasi-steady case is the global L2 density norm is reduced by three orders of magnitude to 10^{-8}. The iterations are stopped when this criteria is reached. The number of iterations required to meet this convergence criteria, the time per iteration and parallel speedup of the time per iteration, the percentage of time spent in communication, and total time statistics are given for this quasi-steady case in Table 1. The results presented use 1 inner sweep in the hybrid LU-SGS algorithm. These were found to give the best CPU time results. In ref. [37], results of these same cases run on the Thinking Machines CM-5 are presented. The results from the CM-5 also indicated that 1 sweep gives the best overall CPU time.

The convergence characteristics of TURNS for this quasi-steady case are shown in plots of the global density L2 norm and the maximum density vs. number of iterations in Figs. 7 and 8, respectively. The behavior of the maximum residual, after 500 iterations, is essentially the same for all processor partitions tested. Some difference is observed in the behavior of the L2 density norm, however. With 1, 4, and 8 processors, the convergence curves are nearly identical. With 19, 57, and 114 processors, the convergence curves are also nearly identical, but are worse than that of 1 processor. On 1 processor, the convergence rate of the density norm is relatively constant until it reaches 3×10^{-8}, at which time there is a leveling off and the method shows a new, slower convergence rate. Similar behavior is shown with 19 processors, but the point at which the leveling off occurs is at about 1×10^{-7}. As a result, approximately 30% more iterations are required to achieve the convergence criteria for the 19, 57, and 114 processor cases.

It is unknown to us why the density norm convergence phenomenon occurs in this way, but it does illustrate the difference in subdividing domains in the wraparound and spanwise directions. With the 1–8 processor cases and and the 19–114 processor cases, where the number of subdomains in the wraparound direction is kept constant and the number in the spanwise direction is varied, there is little difference in convergence. There is, however, substantial difference between the 1 and 19 processor cases, where the spanwise direction is left intact while the wraparound direction is subdivided. This makes physical sense since the flow variables have larger gradients in the wraparound direction during convergence, so subdividing in that direction should cause a reduction in convergence.

On 1 processor of the SP2, TURNS was found to have an execution rate of 21.5 Mflops/second. This is considerably lower than the 100 Mflops/second we were able to achieve with basic test codes, indicating TURNS is not optimized for the RISC chip architecture used on the nodes of the SP2. We expect this is due to portions of unstreamed data in certain parts of the code which cause cache misses. To further support this argument, superlinear parallel speedups are exhibited by the code, which is usually indicative of cache misses. Due the the length of the code, we have not yet made substantial efforts to restructure inefficient portions of the code but will look into this further in the future. It should be noted, however, that TURNS is a long and complex code; Despite modifications to improve vectorization for execution on the Cray C-90, the code still runs at only one quarter the peak speed, on one processor. Therefore, it should be expected that the code will not run at the peak rate on the RISC processor either.

Despite the poor single processor performance,

Table 1: Table 1 - Timing Results on the IBM SP2 for quasi-steady inviscid calculation with TURNS. Nonlifting OLS blade with $M_{at} = 0.837$, $135 \times 50 \times 35$ mesh, global density norm reduced to 10^{-8}.

Processors	#Wraparound	#Spanwise	Iterations	Time/It	Speedup	% Comm.	Tot. Time
1	1	1	2123	15.19 sec		0%	537.5 min
4	1	4	2130	3.92 sec	3.9	1.9%	139.2 min
8	1	8	2165	2.08 sec	7.3	4.0%	75.1 min
19	19	1	2769	0.74 sec	20.6	12.3%	34.2 min
57	19	3	2771	0.26 sec	59.1	16.4%	12.0 min
114	19	6	2783	0.14 sec	107.8	20.0%	6.5 min

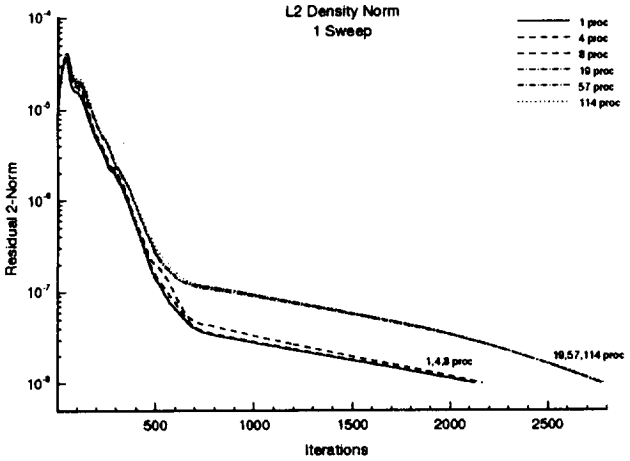

Figure 7: Convergence of global density L2 norm for quasi-steady problem with different processor partitions of the SP2.

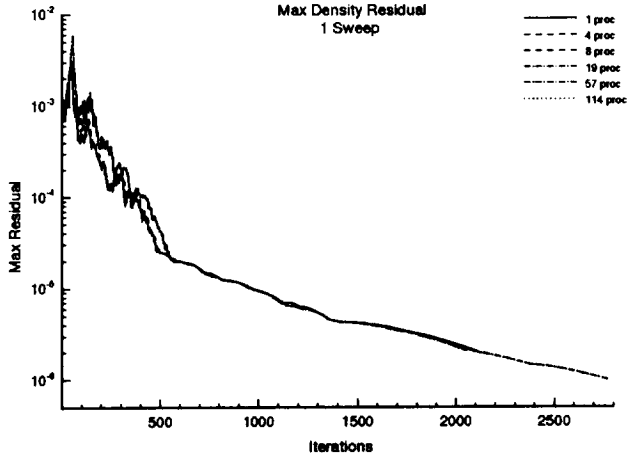

Figure 8: Convergence of maximum global density for quasi-steady problem with different processor partitions of the SP2.

the code exhibits good parallel speedups and has an overall fast execution rate. On 114 processors, the time/iteration of the code is about six times faster than the performance attained on one processor of the Cray C-90. Although more iterations are required on the SP-2 for this case, the entire calculation is done in about 6 minutes. By comparison, this calculation requires about 30 minutes on the Cray C-90.

Using the quasi-steady solution as a starting solution, a time-accurate unsteady flow calculation for one complete revolution of the AH-1 blade is run with TURNS. The same solution mesh is used to start the unsteady calculation, but the mesh is set to rotate with the blade. A timestep corresponding to 0.25 deg azimuth/timestep is used, so a complete revolution is done in 1440 timesteps. Three inner relaxation iterations are performed in each timestep.

Table 2 shows the time/timestep, parallel speedup, percentage of communication, and total times for the unsteady case run on the same processor partitions used in the quasi-steady case. A plot of the global density residual norm vs. time for the unsteady case is shown in Fig. 9 and the parallel speedup exhibited by the code on different processor partitions of the SP2 for this case is shown in Fig. 10.

The time/timestep for the unsteady case is approximately three times that for the quasi-steady run, due to the use of 3 subiterations at each timestep. The percentage communication and parallel speedups are nearly the same as in the quasi-steady run. The domain breakup has a similar effect on the global density residual for the unsteady case as it did for the quasi-steady case; The convergence on 4 and 8 processors is essentially the same as 1 processor, while on 19–114 processors, there is a noticeable reduction. We anticipate that using more sweeps inside the Hybrid method will reduce this effect and we are investigating this further.

The benefit of parallel computation is apparent from the overall solution time. On one processor of the Cray C-90, this calculation required a little over an hour. On 114 processors of the SP2, the calculation time is only 10 minutes, a reduction by a factor of about 6.

Kirchhoff Code Results

The computation time for a single evaluation of a far-field observer pressure on one processor of the SP2

Table 2: Table 2 - Timing Results for a 1 revolution unsteady inviscid calculation with TURNS on the SP2. OLS blade, $M_{at} = 0.837$, 3 inner iterations per timestep, 1440 timesteps with each timestep = 0.25 deg azimuth.

Processors	#Wraparound	#Spanwise	Time/Timestep	Speedup	% Comm.	Tot. Time
1	1	1	45.48 sec		0%	1092 min
4	1	4	11.72 sec	3.9	2.0%	281.3 min
8	1	8	6.26 sec	7.3	3.7%	150.2 min
19	19	1	2.16 sec	21.1	9.8%	51.8 min
57	19	3	0.74 sec	61.4	13.1%	17.8 min
114	19	6	0.43 sec	106.5	20.8%	10.3 min

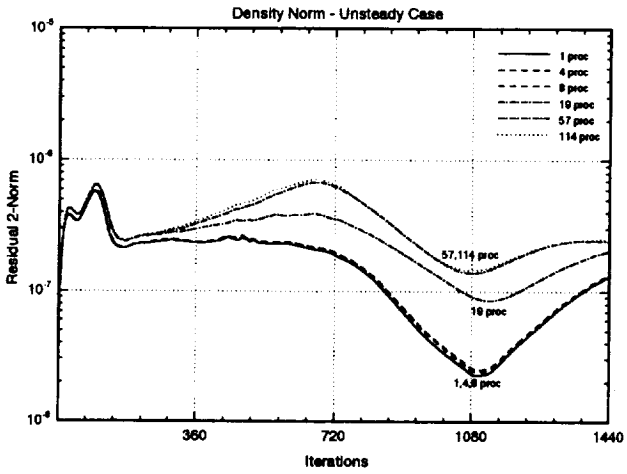

Figure 9: Global density norm history for unsteady calculation on the SP2. One revolution (1440 timesteps at 0.25 deg azimuth/timestep) of AH-1 blade with $M_{at} = 0.837$.

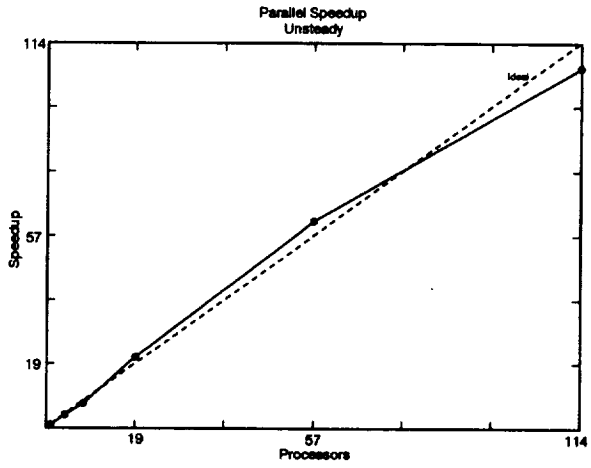

Figure 10: Parallel speedups on SP2 for unsteady calculation.

is about 1.6 seconds. This compares very favorably with a CPU requirement of 1.5 seconds on a single processor of a Cray C-90 computer. The reason for this excellent performance of the SP2 relative to the C-90 is that each integration point on the Kirchhoff surface requires an iterative solution for the nonlinear retarded time equation. This retarded time calculation does not vectorize on the C-90 and the performance is therefore poor. The SP2 currently does not have a vector processing capability; hence, the solution to the retarded time equation does not adversely affect performance. Since no communication is required until a final global summation, and the I/O is independent of the number of processors, the actual run time on the SP2 scales by over 99% as the number of processors increases. As a result, with 80 processors on the SP2, we can compute a periodic time history of 360 pressure evaluations for 1886 observers in few hours. On one processor of the Cray C90, this calculation would require over 200 hours of CPU time.

Visual Postprocessing

Figure 11 shows a sample grid of far-field observer locations for the OLS rotor blade. These grid points are located in the plane of the rotor between 4 and 7 blade radii from the rotor hub. Overall, this grid contains a total of 1886 observer points. Time histories of acoustic pressure were computed for each of these observer locations using the parallel Kirchhoff code on the SP2. These unsteady results are then animated and visualized on an SGI graphics workstation.

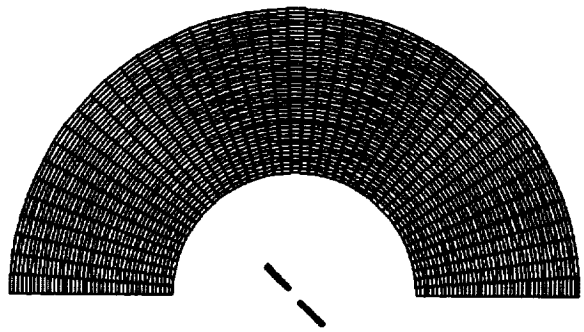

Figure 11: Far-field grid of observer points for the Kirchhoff acoustics simulation.

Figure 12 shows four snapshots from an animation of the resulting acoustic field. The acoustic pressure

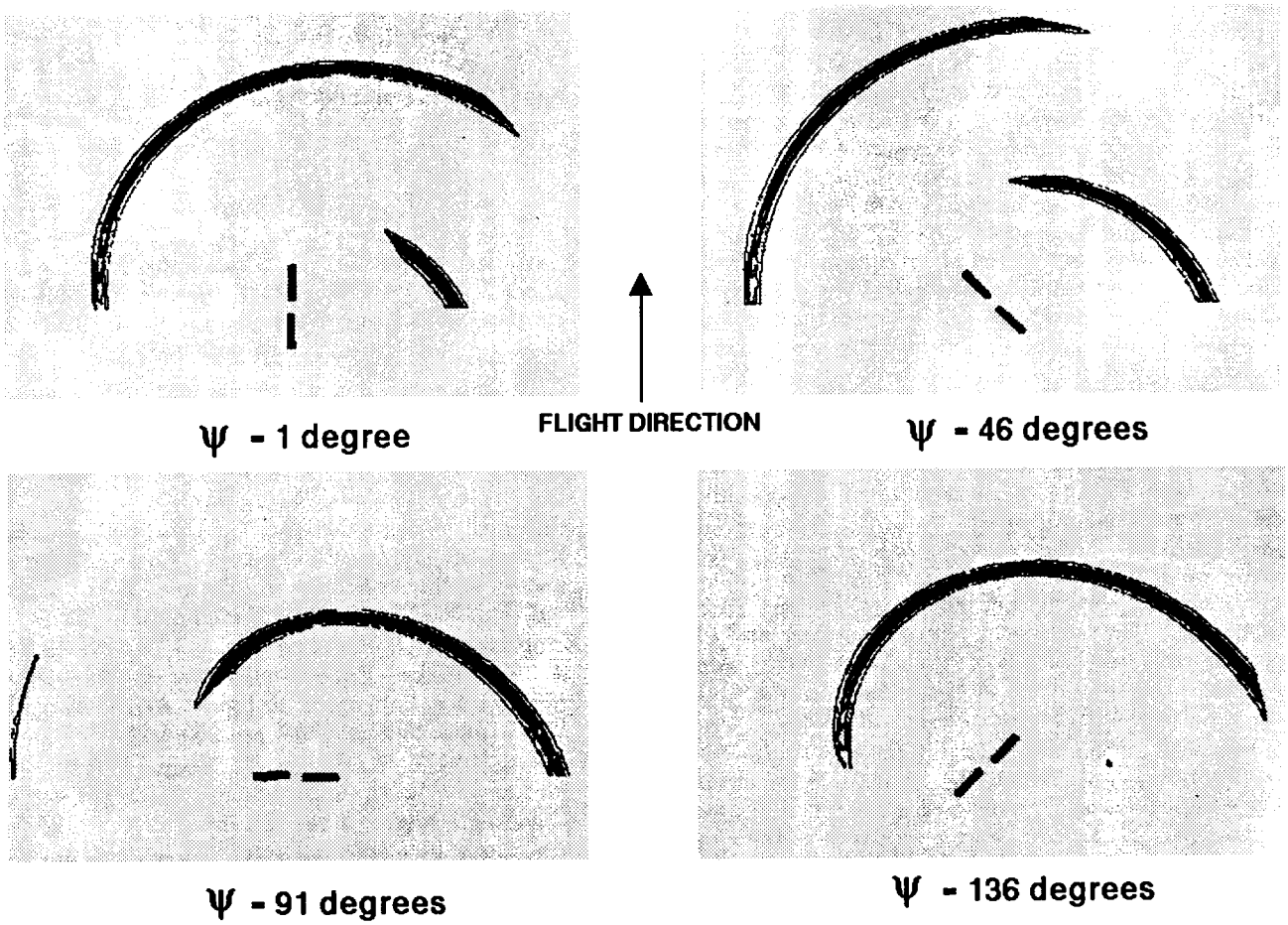

Figure 12: Contours of computed far-field acoustic pressure for several different blade azimuth positions. The pressure contours are scaled by the radius from the rotor hub.

fields are scaled by the distance from the rotor hub, which helps to show the directivity of the noise in the far field. At each blade azimuthal location, the maximum scaled acoustic amplitude contours contours show that the maximum directivity of the acoustic signal is along the direction of flight, directly ahead of the rotor blade and slightly to the advancing side. This is consistent with data obtained from windtunnel experiments and flight tests for similar conditions.

The animated results give a clear picture of the unsteady propagation of acoustic signals in the far field. The advantage to this approach is that the entire field can be viewed at once. This conveys much more information to the viewer than what can be seen from a handful of far-field experimental microphones. The importance of this additional information is even greater for BVI noise where the far-field acoustic propagation is much more complex than that seen in Fig. 12.

Audio Postprocessing

Time-dependent CFD/Kirchhoff rotorcraft acoustics simulations are well-suited for audio playback. The CFD/Kirchhoff integration provides pressure data at discrete locations in the far-field such as those shown in Fig. 11. Audio playback is obtained by using the Stereophonic Acoustics Software Library that has been recently developed at NASA Ames Research Center. This software package allows the user to interactively select individual grid points where pressure data can be played over headphones or stereo speakers connected to an SGI Indigo computer. The digital pressures for the entire field are scanned for maximum and minimum amplitudes and then scaled for 16-bit stereo sound. The user can add a slight phase shift between the signals for each ear. This phase shift simulates the inter-aural time delay that conveys spatialized acoustic perceptions to the human brain.

The sample results in Fig. 12 have been processed for the audio playback described above. Although we cannot effectively convey the audio results in this paper, the consensus opinion of those who have heard it is that the simulation is very realistic. A video will be shown at the presentation demonstrating this.

Concluding Remarks

An efficient parallelization of two codes used in a combined CFD/Kirchhoff methodology is presented. An Euler/Navier Stokes rotorcraft CFD code called TURNS and an existing rotating Kirchhoff code are both implemented on the 160 node IBM SP2 multiprocessor at NASA Ames. The two codes are used together to predict both the near-field aerodynamics and

far-field aeroacoustic characteristics of an AH-1 blade rotating with $M_{at} = 0.837$. TURNS is used to compute the inviscid flowfield in the region near the blades and the Kirchhoff code, using the CFD results, computes the far-field noise characteristics. Some algorithm changes are required to allow the codes to have efficient parallel execution, but a domain decomposition implementation is used that requires relatively small modifications to the existing codes. Communications are performed by adding standard MPI message passing calls to to the Fortran code, and techniques are used to ensure easy portability.

The main portion of TURNS that presents difficulty for parallel implementation is the LU-SGS algorithm, used for the implicit timestep. A modified Hybrid domain decomposition implementation of LU-SGS is utilized. Although some reduction in convergence is observed with this implicit algorithm, the reduction is small and the method shows excellent parallel speedups. Three dimensional quasi-steady and unsteady calculations are performed with TURNS using up to 114 processors, and the time/iteration for these cases is reduced by a factor of 6 over the Cray C-90.

Implementation of the Kirchhoff solver on the SP2 is relatively straightforward due to the regular nature of the computations. The method is nearly as efficient on 1 processor of the SP2 as on 1 processor of the Cray C-90 and shows nearly perfect parallel speedups on the SP2. The unsteady far-field acoustic properties are calculated at 1886 observer locations, and visual as well as audio postprocessing is performed using the Stereophonic Acoustics Software Library at NASA Ames to gain better insight to the far-field noise characteristics. These calculations, which would take over 200 hours of CPU time on one processor of the Cray C-90, are performed on the SP2 in a few hours. Ref. [42] will contain a more complete description and more results from the parallel Kirchhoff method and the associated video and audio postprocessing.

Finally, although the results presented here are for a relatively simple nonlifting two-blade configuration, the same CFD/Kirchhoff algorithms discussed in this paper are used on more advanced configurations such as the V-22 tiltrotor [43]. The use of parallel processing will allow larger and more complex problems to be solved in the future. Also, while the CFD/Kirchhoff approach is applied here to helicopter noise prediction, the parallelization strategies are not unique to this application and the approaches could readily be applied to other problems.

Acknowledgments

The first author was supported by a NASA Graduate Student Fellowship. This work was supported by allocation grants from the Minnesota Supercomputer Institute (MSI). Computer time on the IBM SP2 was provided by a grant from the Computational Aerosciences branch of NASA Ames. The authors wish to acknowledge the support of Dr. G.R. Srinivasan of JAI Associates for his assistance with the TURNS code.

References

[1] Srinivasan, G.R., "A Free-Wake Euler and Navier-Stokes CFD Method and its Application to Helicopter Rotors Including Dynamic Stall," JAI Associates, Inc., Technical Report 93-01, November 1993.

[2] Wake, B.E., and Sankar, L.N., "Solution of Navier-Stokes Equations for the Flow over a Rotor Blade," *Journal of the American Helicopter Society*, Vol. 34, April 1989, pp. 13-23.

[3] 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.

[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] 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.

[6] Srinivasan, G.R., and Ahmad, J.U., "Navier Stokes Simulation of Rotor-Body Flowfields in hover Using Overset Grids," *Proceedings of the Nineteenth European Rotorcraft Forum*, Paper No. C15, September 1993, Cernobbio Italy.

[7] Duque, E.P.N., and Srinivasan, G.R., "Numerical Simulation of a Hovering Rotor Using Embedded Grids," *Proceedings of the 48th Annual Forum of the American Helicopter Society*, Washington DC, June 1992.

[8] Duque, E.P.N., "A Structured/Unstructured Embedded Grid Solver for Helicopter Rotor Flows," *Proceedings of the 50th Annual Forum of the American Helicopter Society*, Vol. II, Washington DC, May 1994, pp. 1249-1257.

[9] Baeder, J.D., Gallman, J.M., and Yu, Y.H., "A Computational Study of the Aeroacoustics of Rotors in Hover," *Proceedings of 49th Annual Forum of the American Helicopter Society*, St. Louis, Missouri, May 1993, pp. 55-71.

[10] Baeder, J.D., and Srinivasan, G.R., "Computational Aeroacoustic Study of Isolated Blade Vortex Interaction Noise," AHS Specialists' Aeromechanics Conference, San Francisco, CA, Jan 1994.

[11] George, A. R., and Lyrintzis, A. S., "Acoustics of Transonic Blade-Vortex Interactions," *AIAA Journal*, Vol. 26, No. 7, Jul. 1988, pp. 769–776.

[12] Lyrintzis, A. S., and George, A. R., "Far-Field Noise of Transonic Blade-Vortex Interactions," *Journal of the American Helicopter Society*, Vol. 34, No. 3, July 1989, pp. 30-39.

[13] Lyrintzis, A. S., and Xue, Y., "A Study of the Noise

Mechanisms of Transonic Blade-Vortex Interactions," *AIAA Journal*, Vol. 29, No. 10, Oct. 1991, pp. 1562-1572.

[14] Lyrintzis, A. S., Kilaras, M. S., and Xue, Y., "Transonic 3-D BVI Noise Using a Rotating Kirchhoff Formulation for Advancing Rotors," *Proceedings of the 50th AHS Annual Forum*, Vol. I, Washington, DC, May 1994. pp. 115-127.

[15] Lyrintzis, A. S., Xue, Y., and Kilaras, M. S., "The Use of a Rotating Kirchhoff Formulation for High-Speed Impulsive Noise," AIAA Paper 94-0463, presented at the 32nd AIAA Aerospace Sciences Conference, Jan. 1994.

[16] Strawn, R. C., and Biswas, R., "Computation of Helicopter Rotor Noise in Forward Flight," *Journal of the American Helicopter Society*, Vol. 40, No. 3, July 1995, pp. 66-72.

[17] Strawn, R. C., and Biswas, R., "Numerical Simulations of Helicopter Aerodynamics and Acoustics," presented at the 6th International Congress on Computational and Applied Mathematics, Leuven, Belgium, Jul. 1994.

[18] Strawn, R.C., Biswas, R., and Garceau, M., "Unstructured Adaptive Mesh Computations of Rotorcraft High-Speed Impulsive Noise," *Journal of Aircraft*, Vol. 32, No. 4, July-Aug. 1995, pp. 754-760.

[19] Xue, Y., and Lyrintzis, A. S., "Rotating Kirchhoff Method for Three-Dimensional Transonic Blade-Vortex Interaction Hover Noise," *AIAA Journal*, Vol. 32, No. 7, Jul. 1994, pp. 1350–1359.

[20] Strawn, R. C., Biswas, R., and Lyrintzis, A. S., "Helicopter Noise Predictions using Kirchhoff Methods," Proceedings of the 51st Annual Forum of the American Helicopter Society, 9–11, May, 1995, Fort Worth, Texas, Vol. I, pp. 495-508; to appear in *Journal of Computational Acoustics*.

[21] Lyrintzis, A. S, Koutsavdis, E. K., and Strawn, R. C., "A Comparison of Computational Aeroacoustic Prediction Methods," *Proceedings of the AHS Aeromechanics Specialists' Conference*, Stratford, CT, Oct. 1995, Vol. I, pp. 3-58–69.

[22] Brentner, K. S., "Prediction of Helicopter Rotor Discrete Frequency Noise," NASA Technical Memorandum 87721, Oct. 1986.

[23] Lyrintzis, A. S., "Review: The Use of Kirchhoff's Method in Computational Aeroacoustics," *ASME Journal of Fluids Engineering*, Vol. 116, Dec. 1994, pp. 665–675.

[24] Fischberg, C., Rhie, C., Zacharias, R., Bradley, P., and DesSurealt, T., "Using Hundreds of Workstations for Production Running of Parallel CFD Applications," in *Parallel Computing 1995*, 1995.

[25] Knight, D.D., "Parallel Computing in Computational Fluid Dynamics," Presented at the 77th AGARD FDP Meeting, Paper 3, Oct. 2–5, Seville, Spain.

[26] Caradonna, F. X., Strawn, R. C., and Bridgeman, J. O., "An Experimental and Computational Study of Rotor-Vortex Interactions," Vertica, Vol. 12, No. 4, 1988, pp. 315–327.

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

[28] 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.

[29] 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.

[30] Yoon, S., and Kwak, D., "Three-Dimensional Incompressible Navier Stokes Solver Using Lower-Upper Symmetric-Gauss-Seidel Algorithm," *AIAA Journal*, Vol. 29, No. 6, June 1991, pp. 874-875.

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

[32] Barszcz, E., Fatoohi, R., Venkatakrishnan, V., and Weeratunga, S., "Solution of Regular, Sparse Triangular Linear Systems on Vector and Distributed-Memory Multiprocessors," NASA Report RNR-93-007, April 1993.

[33] Candler, G.V., Olynick, D.R., "Hypersonic Flow Simulations Using a Diagonal Implicit Method," presented at the 10th International Conference on Computing Methods in Applied Sciences and Engineering, Paris France, Feb. 1992.

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

[35] Wright, M.J., Candler, G.V., and Prampolini, M. "A Data Parallel LU Relaxation Method for the Navier Stokes Equations," AIAA Paper 95-1750, 1995.

[36] Wong, C.C., Blottner, F.G., and Payne, J.L., "A Domain Decomposition Study of Massively Parallel Computing in Compressible Gas Dynamics," AIAA Paper 95-0572, Jan. 1995.

[37] Wissink, A.W., Lyrintzis, A.S., and Strawn, R.C., "On Improving Parallelism in the Transonic Unsteady Rotor Navier Stokes (TURNS) Code", Presented at the 77th AGARD FDP Meeting, Paper 6, Oct. 2–5, Seville, Spain.

[38] Wissink, A.M., Lyrintzis, A.S., and Strawn, R.C. "On the Parallelization of the Transonic Unsteady Rotor Navier Stokes Code," presented at the Computational Aerosciences Conference, March 11–13, 1995, Santa Clara, CA.

[39] Farassat, F., and Myers, M. K., "Extension of Kirchhoff's Formula to Radiation from Moving Surfaces," *Journal of Sound and Vibration*, Vol. 123, No. 3, 1988, pp. 451–460.

[40] Myers, M. K., and Hausmann, J. S., "On the Application of the Kirchhoff Formula for Moving Surfaces," *Journal of Sound and Vibration*, Vol. 139, 1990, pp. 174–178.

[41] Schmitz, F. H., Boxwell, D. A., Splettstoesser, W. R., and Schultz, K. J., "Model-Rotor High-Speed Impulsive Noise: Full-Scale Comparisons and Parametric Variations," Vertica, Vol 8, No. 4, 1984, pp. 395-422.

[42] Strawn, R.C., Oliker, L., DeRyke, D., and Biswas, R. "New Computational Methods for the Prediction and Analysis of Helicopter Acoustics," to be presented at the 2nd AIAA/CEAS Aeroacoustics Conference, State College, PA, May 6–8, 1996.

[43] Duque, E. P. N., Strawn, R. C., Ahmad, J., and Biswas, R., "An Overset Grid Navier-Stokes/Kirchhoff-Surface Method for Rotorcraft Aeroacoustics Predictions," AIAA Paper #96-0152, presented at the 34th Aerospace Sciences Meeting, Reno, NV, Jan. 15–18, 1996.

RIACS
Mail Stop T041-5
NASA Ames Research Center
Moffett Field, CA 94035