INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **55**:15–27
Published online 19 June 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1277

Parallel solution of lifting rotors in hover and forward flight

C. B. Allen*, †

Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, U.K.

SUMMARY

An implicit unsteady, multiblock, multigrid, upwind solver including mesh deformation capability, and structured multiblock grid generator, are presented and applied to lifting rotors in both hover and forward flight. To allow the use of very fine meshes and, hence, better representation of the flow physics, a parallel version of the code has been developed. It is demonstrated that once the grid density is sufficient to capture enough turns of the tip vortices, hover exhibits oscillatory behaviour of the wake, even using a steady formulation. An unsteady simulation is then presented, and detailed analysis of the time-accurate wake history is performed and compared to theoretical predictions. Forward flight simulations are also presented and, again, grid density effects on the wake formation investigated. Parallel performance of the code using up to 1024 CPU's is also presented. Copyright © 2006 John Wiley & Sons, Ltd.

Received 10 January 2006; Revised 31 March 2006; Accepted 20 April 2006

KEY WORDS: parallel processing; unsteady CFD; multigrid; multiblock; rotor wakes

1. INTRODUCTION

Simulation of lifting rotor flows is expensive for compressible finite-volume codes, as the complex vortical wake needs to be captured over several turns of the blades to obtain the correct blade loads. This requires fine meshes away from the blades and a long numerical integration time for the wake to develop, which can lead to impractical run times. Hence, parallelization of the flow-solver is essential to be able to consider grid densities of practical use for these flows, and this is considered here. A finite-volume, upwind scheme using multiblock structured meshes is presented, which includes steady and unsteady time-stepping options and time-accurate mesh deformation capability. To allow the use of very fine meshes and, hence, better wake and vortex capturing, the code has been parallelized using MPI [1], and is applied to lifting multi-bladed rotors in hover and forward flight.

*Correspondence to: C. B. Allen, Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, U.K.
†E-mail: c.b.allen@bristol.ac.uk

Experimental data for rotors in hover, for example, References [2–5], have suggested the wakes are actually unsteady. This has been confirmed by theoretical stability analyses [6–8], which imply a hovering rotor wake is fundamentally unstable with several unstable modes. Other simulation methods, for example, free-vortex or free-wake [8, 9] or vorticity transport methods [10, 11], with low inherent dissipation, have also shown evidence of unsteady hover wakes, confirming the theoretical and experimental data. However, there has been little published work on wake unsteadiness using compressible CFD simulation, primarily because these flows are normally run with a steady solver, and with insufficient grid densities to capture enough of the wake for any instability to manifest itself. This is considered here.

Forward flight simulation adds the further complications that an unsteady solver must be used, and the entire domain must be simulated (rather than just the part domain associated with one blade for hover cases). This can significantly increase the cost of the simulation, but the solution now evolves faster, and the effect of grid density away from the blades is less significant, since the wake is swept downstream rather than form below the blades. Forward flight is also simulated here and, again, grid dependence of the wake formation is considered.

Steady and unsteady formulations of the flow-solver are first presented, followed by aspects of the parallelization approach used. Inextricably linked to the flow-solver is the grid generation approach, and so this is also discussed for structured multiblock grids for rotors in hover and forward flight. Numerical solutions for lifting rotors are then presented and examined, particularly in terms of vortical wake capturing properties, and grid dependence, using up to 32 million points for a four-bladed case. Parallel performance of the code using up to 1024 CPU's is also presented.

2. FLOW-SOLVER

2.1. Steady approach

Hover simulation requires a rigid but rotating grid. The equations are transformed to a blade-fixed rotating reference frame as in this frame the hover case is a steady problem. Absolute velocities are used. If the frame rotates with angular velocity $\omega = [\Omega_x, \Omega_y, \Omega_z]^T$, and the absolute velocity vector in the rotating frame is denoted by $\mathbf{q_r} = [u_r, v_r, w_r]^T$, the resulting Euler equations in integral form are then:

$$\frac{d}{dt}\int_{V_r} \mathbf{U_r}\, dV_r + \int_{\partial V_r} \mathbf{F_r}\cdot \mathbf{n_r}\, dS_r + \int_{V_r} \mathbf{G_r}\, dV_r = 0 \tag{1}$$

where

$$\mathbf{U_r} = \begin{bmatrix} \rho \\ \rho u_r \\ \rho v_r \\ \rho w_r \\ E \end{bmatrix}, \quad \mathbf{F_r} = \begin{bmatrix} \rho[\mathbf{q_r} - (\omega \times \mathbf{r})] \\ \rho u_r[\mathbf{q_r} - (\omega \times \mathbf{r})] + P\mathbf{i_r} \\ \rho v_r[\mathbf{q_r} - (\omega \times \mathbf{r})] + P\mathbf{j_r} \\ \rho w_r[\mathbf{q_r} - (\omega \times \mathbf{r})] + P\mathbf{k_r} \\ E[\mathbf{q_r} - (\omega \times \mathbf{r})] + P\mathbf{q_r} \end{bmatrix}, \quad \mathbf{G_r} = \begin{bmatrix} 0 \\ \rho(\omega \times \mathbf{q_r})\cdot \mathbf{i_r} \\ \rho(\omega \times \mathbf{q_r})\cdot \mathbf{j_r} \\ \rho(\omega \times \mathbf{q_r})\cdot \mathbf{k_r} \\ 0 \end{bmatrix} \tag{2}$$

Here $\mathbf{G_r}$ is the source term resulting from the transformation, and $\mathbf{r} = [x_r, y_r, z_r]^T$ is the coordinate vector. This is a steady case, as the grid speeds are constant, i.e. the coordinate vector is constant in the rotating frame. The equation set is closed by

$$P = (\gamma - 1)\left[E - \frac{\rho}{2}\mathbf{q}_\mathbf{r}^2\right] \tag{3}$$

2.1.1. Spatial discretization. A finite-volume upwind scheme is used to solve the integral form of the Euler equations (Equation (1)). The flux-vector splitting of Van-Leer [12] is used, along with a third-order spatial interpolation and the continuously differentiable flux limiter due to Anderson *et al.* [13].

To avoid introducing errors due to the evaluation of the rotational terms, the local ($\omega \times \mathbf{r}$) terms are evaluated by solving for cell face area moment vectors, and satisfying the resulting conservation condition [14, 15]. (More details of the spatial scheme can be found in Reference [15].)

2.1.2. Explicit time-stepping scheme. An explicit three-stage Runge–Kutta scheme is used to integrate the equations forward in time, including local time stepping.

2.1.3. Multigrid scheme. It has been shown previously that multigrid is effective for hovering rotors using a steady approach [15], and for forward flight using an unsteady approach [16], and hence, multigrid acceleration is also adopted here. A V-cycle is adopted with a trilinear interpolation of corrections within the prolongation steps.

2.2. Unsteady approach

Forward flight simulation requires an unsteady solution scheme. Furthermore, to allow the cyclic pitch variation of the blades, to balance the forces and moments around the azimuth, the grid must also deform. The Euler equations in integral form for a deforming mesh rotating in a fixed axis system are then:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{U}\,\mathrm{d}V + \int_{\partial V} \mathbf{F}\cdot\mathbf{n}\,\mathrm{d}S = 0 \tag{4}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho[\mathbf{q} - (\omega\times\mathbf{r}(t) + \mathbf{r_t}(t))] \\ \rho u[\mathbf{q} - (\omega\times\mathbf{r}(t) + \mathbf{r_t}(t))] + P\mathbf{i} \\ \rho v[\mathbf{q} - (\omega\times\mathbf{r}(t) + \mathbf{r_t}(t))] + P\mathbf{j} \\ \rho w[\mathbf{q} - (\omega\times\mathbf{r}(t) + \mathbf{r_t}(t))] + P\mathbf{k} \\ E[\mathbf{q} - (\omega\times\mathbf{r}(t) + \mathbf{r_t}(t))] + P\mathbf{q} \end{bmatrix} \tag{5}$$

The coordinate vector $\mathbf{r}(t)$ is time dependent in the fixed axis system, i.e.

$$\mathbf{r}(t) = [R(\Psi)]\mathbf{r}(0) \tag{6}$$

where $[R(\Psi)]$ is the time-dependent rotation matrix (Ψ is the azimuth position, and so $\Psi = \Omega t$). The $\omega \times \mathbf{r}(t)$ term is due to the rigid rotation, but there is also the grid velocity associated with grid deformation, and this is the $\mathbf{r_t}(t)$ term in the above.

The equation set is closed by

$$P = (\gamma - 1)\left[E - \frac{\rho}{2}\mathbf{q}^2\right] \tag{7}$$

2.2.1. Implicit time-stepping scheme. An implicit form of the differential equation for each computational cell is considered:

$$\frac{\partial(V^{n+1}\mathbf{U}^{n+1})}{\partial t} + \mathbf{R}(\mathbf{U}^{n+1}) = 0 \tag{8}$$

where V is the time-dependent cell volume and $\mathbf{R}$ is the upwinded flux integral. The implicit temporal derivative is then approximated by a second-order backward difference, following Jameson [17].

The multigrid explicit multi-stage Runge–Kutta scheme used for steady flows is used to integrate the equations forward in pseudo-time, within each real time-step.

2.2.2. Mesh deformation. The mesh is deformed each time-step using a global parameterization approach. The time-dependent cell volumes are then computed by satisfying the required geometric conservation law [18]. Each blade has its own local axis system, and so everything associated with mesh motion is computed initially at the $t = 0$ ($\Psi = 0$) position, and then rotated to the instantaneous Ψ position. The same second-order backward scheme is used to compute the time-dependent grid velocities as for the flow:

$$\mathbf{r_t}(t) = \frac{1}{2\Delta t}[R(\Psi)]\{3\mathbf{r}^{n+1} - 4\mathbf{r}^n + \mathbf{r}^{n-1}\} \tag{9}$$

2.3. Parallelization

The flow-solver has been parallelized using MPI [1], to allow the use of fine meshes. An algorithm linked to the grid generator reads in the mesh dimensions, in the coarsest (in terms of number of blocks) form, and splits it into more blocks, if required, while attempting to balance the number of points on each CPU, and maintaining the maximum number of multigrid levels. This can be run before the grid generation process, to optimize the mesh dimensions and load balancing.

Within the solver each block boundary only requires one boundary condition tag, a neighbouring block number and an orientation flag to apply the correct internal or external conditions. Hence, each block only requires the grid dimensions, the physical grid point coordinates, then six lines, of three integers each, defining the boundary conditions. All sends, at each internal block boundary, are performed by packing solution data and sending to the appropriate processor. Received boundary data are then unpacked according to the orientation flag. This has the advantage that no cell connectivity data are required in the grid file and, hence, there is no preprocessing or domain decomposition stage. A separate file can be written for each block, and a header file created which defines the block–CPU relationship and the name of each block file. The code has also been written such that all parallel communication is performed in specific subroutines, and this means there is no message passing in any of the fundamental routines.

The most significant aspect of this data structure is that there is no global flow or geometry data storage. Only arrays of block dimensions and connectivity are required globally, which are only 1D arrays of length *nblocks*, and this vastly reduces the memory requirements of the code. All I/O is also handled locally, and so there is only one subroutine in the entire code that requires any global data collection, and that is the monitoring routine. This simply collects the residuals

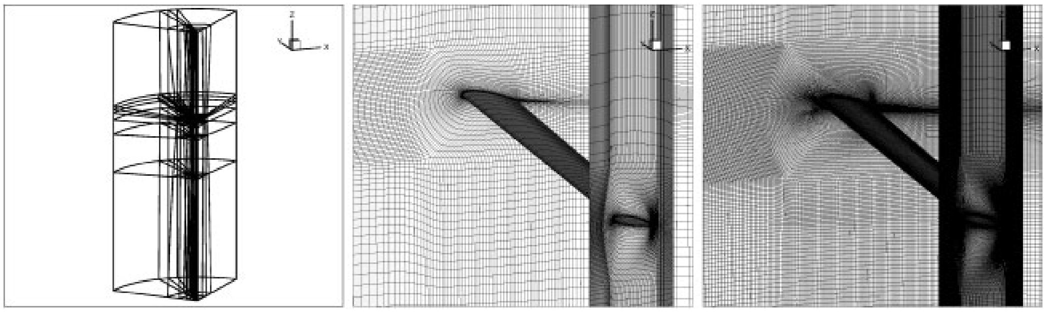

Figure 1. Four-bladed multiblock hover mesh domain and block boundaries, and mesh in selected planes, 2 and 32×10^6 points.

and load components from each CPU, and writes to the output file. Avoiding any global storage or collection of flow/geometry data means the code is extremely efficient in terms of memory; the 32 million point mesh cases shown later can be run with four levels of multigrid on 16 dual-CPU nodes, with only 1 GB RAM per node, i.e. less than 0.5 GB/million points.

3. GRID GENERATION

An efficient grid generator has been developed that can generate multiblock meshes for fixed wings and rotors in hover and forward flight. The grid is generated by first generating a blade-fixed grid around the solid surface, using a variable periodic transformation [19]. From this near-blade grid, cylindrical grids are generated upstream, downstream, above and below the blade to fill the required domain. This domain is a $(1/N)$th part cylinder, where N is the number of blades. In forward flight extra blocks are added to mesh the hub, which is transformed to a spherical shape, and the flow regions above and below (these are not required for hover). The mesh is then repeated and rotated, to fill the entire domain.

A transfinite interpolation algorithm [20, 21] is used, with improved orthogonality and stretching functions, to compute the grid within each block. After generating the entire mesh an elliptic smoothing is applied [22].

A wake grid dependence study has been performed, considering the ONERA 7A blade [23], and meshes of density 2, 8, and 32 million points were generated. Figure 1 shows the domain and block boundaries for a 120-block periodic hover mesh, and the grid in selected grid planes (hub, rotor surface, and planes near the tip) for 2 and 32 million points, for a four-bladed case. Figure 2 shows domain and block boundaries for a 192 block forward flight mesh, and views of the rotor disk grid for the 2 and 32 million point meshes.

4. 7A ROTOR HOVER SIMULATION

The case simulated was $M_{\text{Tip}} = 0.661$, and was run with meshes of density 2, 8, and 32 million points. The spanwise far-field, i.e. radius of sector, was set at 80 chords, the upper boundary

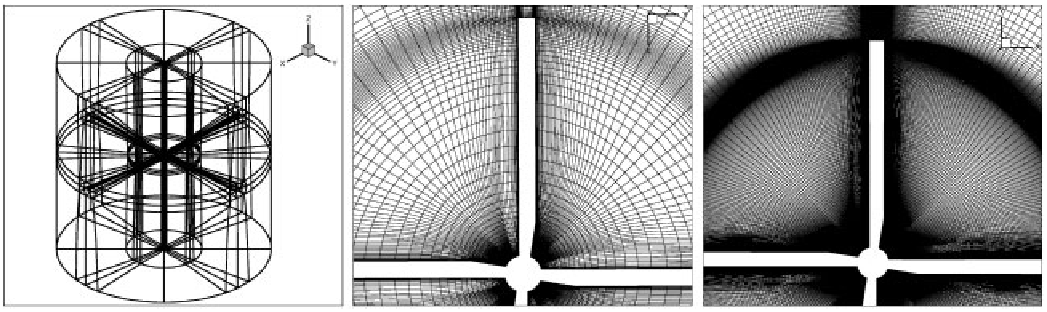

Figure 2. Four-bladed multiblock forward flight mesh domain and block boundaries, and mesh in rotor disk, 2 and 32×10^6 points.

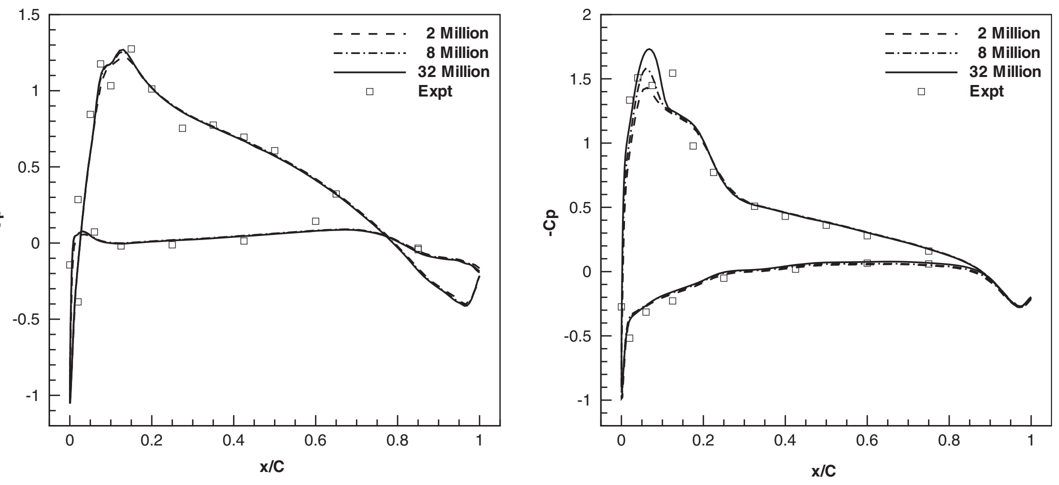

Figure 3. Four-bladed 7A hover. Pressure coefficient at $r/R = 0.7$ and 0.92.

at 80 chords, and lower boundary at 150 chords. Characteristic conditions are applied at these boundaries.

Figure 3 shows the surface pressure coefficient variation at two radial stations, for the three mesh densities, compared to experiment. This shows excellent agreement, although it also demonstrates the grid dependence as there is a noticeable difference between the 8 million and 32 million cases at $r/R = 0.92$.

The vorticity in the downstream periodic plane near the rotor is presented in Figure 4 for the three grid densities (this is the plane 45° behind the blade). The position of the rotor is also shown for clarity, and all figures show 151 contour levels between 0.01 and 2.0. The lower-right figure is a wider view of the 32 million point case to show how many vortex turns are captured. The grid dependence is clearly demonstrated here. The vortex sheet is also much more evident in the 32 million point case, with around one and a half turns captured.

Figure 5 shows the convergence histories of the steady simulations. The residual for the 32 million point case becomes periodic, showing that unsteadiness in the wake is now being captured.

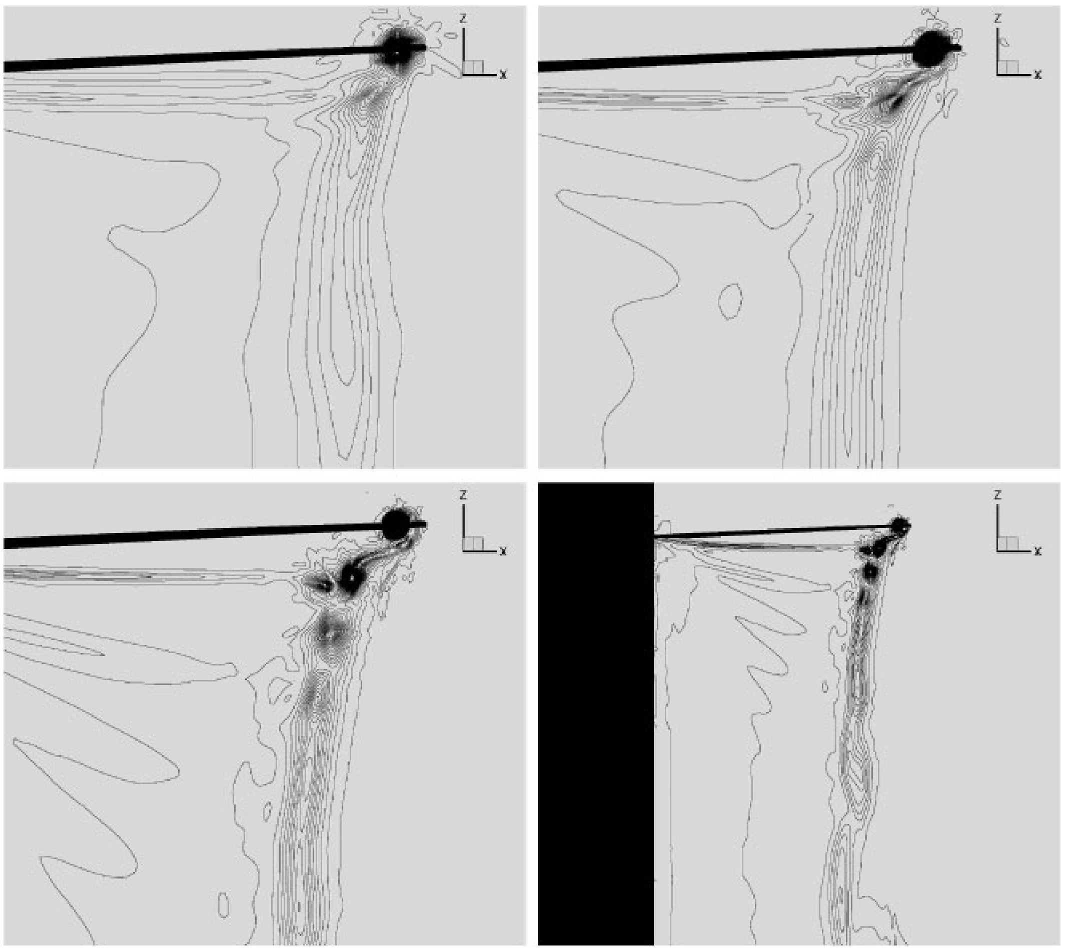

Figure 4. Four-bladed 7A hover. Vorticity contours in downstream periodic plane, 2×10^6 (top-left), 8×10^6 (top-right), 32×10^6 points.

As discussed in the Introduction, previous work has shown that hover is not actually a steady case, but little, if any, has been presented on simulation of unsteady hover flows using finite-volume CFD methods. Stability analysis [8] shows that the wake should actually be in periodic equilibrium, but is unstable to all perturbations. Hence, any experimental or numerical result is almost certain to become unstable, due to any perturbation.

5. UNSTEADY 7A HOVER SIMULATION

This case was also run as an unsteady case. A 32 million point 'steady' solution was used as the initial solution, and the unsteady solver started with a few very small time-steps, and this was

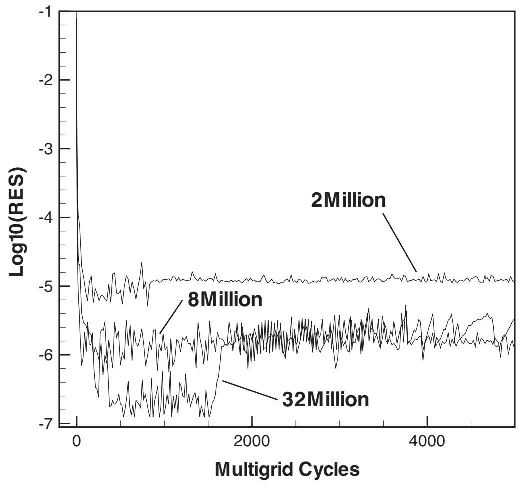

Figure 5. Four-bladed 7A hover simulation convergence histories.

gradually increased to a constant 90 time-steps per revolution. The solution remained unsteady for 40 computed revolutions.

5.1. *Time history analysis*

The vorticity time history computed on the downstream periodic plane was analysed to extract the unsteady motion frequencies. To reduce the amount of data, the periodic plane was split into several smaller planes, each of which was further split into 12 sub-blocks, and the total vorticity in each sub-block summed at each time-step. Fourier transforms of this quantity were then computed over the entire simulation.

Figure 6 (left) shows the Fourier coefficients for one of the planes, for the frequency range, relative to the rotation frequency, of 0 to 100, and right is the lower frequency range only for this plane. Hence, there are distinct peaks around 1, 4, and 8 times the rotation frequency and another at 90 times rotation frequency. The higher peak is due to the time-stepping input, as there are 90 time-steps per revolution. Similar results were found in the other planes.

5.2. *Theoretical comparison*

Linear stability analysis of hovering rotor wakes has previously been published [8], and showed that all wave numbers (modes) are unstable to all perturbations. Hence, any mode which is perturbed will be unstable. In the unsteady simulation above, three distinct modes have been captured, and these are all multiples of the rotor frequency. Hence, these can be assumed to have been perturbed by the numerical time-stepping scheme. However, there are expected modes that have not been excited, for example, the modes associated with vortex pairing. These are associated with wave numbers $N(k+1/2)$ where N is the number of blades, and k is any integer $\geqslant 0$. These are physical modes, and are not captured due to the dissipation of the vortices. Hence, the wake is unstable to perturbations, as predicted, but the instabilities appear to be due to the numerics and not the physics.

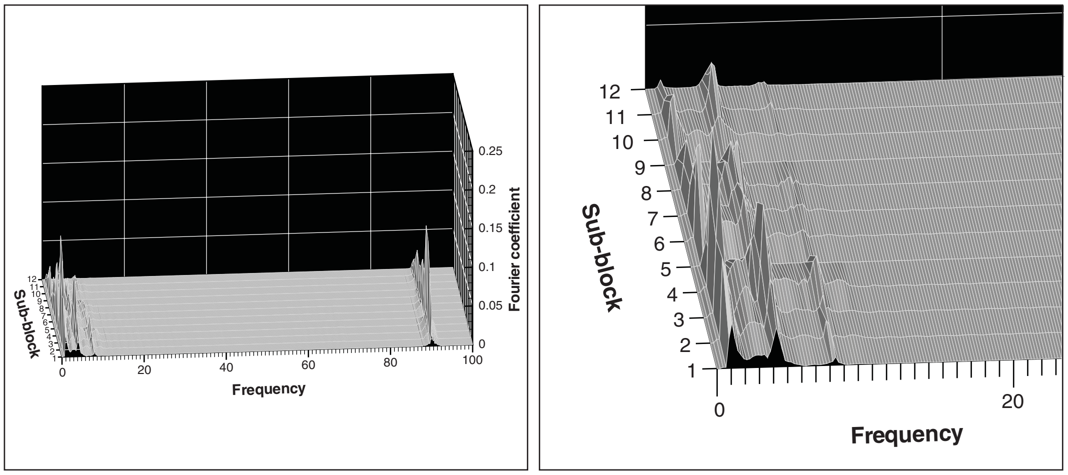

Figure 6. Fourier transform of total vorticity time-history in sub-blocks.

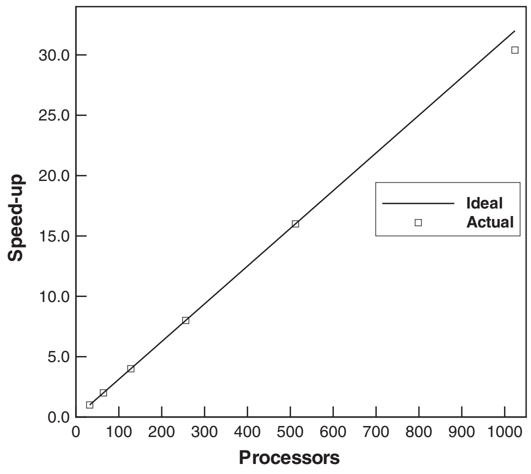

Figure 7. Parallel performance of the code.

6. PARALLEL PERFORMANCE OF THE CODE

The code has been run on the National Supercomputer HPCx, and benchmarks performed. Figure 7 shows the speed-up obtained using from 32 up to 1024 CPU's (actually for a 20 million point case).[‡] A factor of one was used for 32 CPUs as fewer than this were not used. Hence, the code has been awarded a Gold Star for scaling performance by the National Supercomputing Centre.

[‡]Figures produced by Andy Sunderland of the Terascaling Team at the National Supercomputing Centre.

7. THE 7A ROTOR IN LIFTING FORWARD FLIGHT

A lifting forward flight test case was also run with the 7A rotor. The case considered has a tip Mach number of 0.618, and an advance ratio, μ, of 0.214 ($\mu = M_{\mathrm{FF}}/M_{\mathrm{Tip}}$). The case also has a shaft inclination of $-3.72°$, i.e. backward, so has significant blade vortex interaction effects. The collective pitch setting is 2.74°, and the cyclic pitch variation,

$$\Theta(\Psi) = \theta_c \cos(\Psi) + \theta_s \sin(\Psi) \tag{10}$$

is determined by: $\theta_c = 2.66°$, $\theta_s = -2.24°$.

The case was run as an unsteady simulation, using 360 real time-steps per revolution, and the third revolution was taken as converged. Meshes of density 2, 8, and 32 million points were again used.

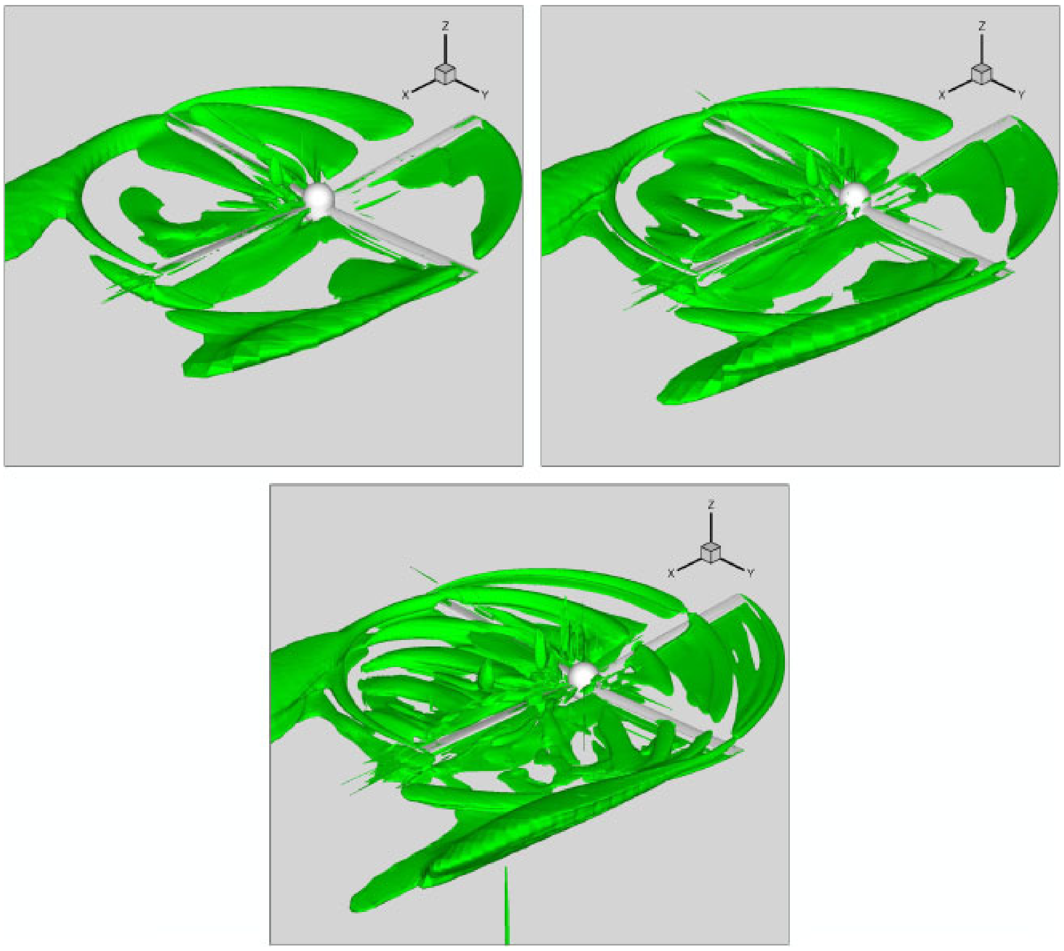

Figure 8. 7A Forward flight λ_2 isosurface, 2×10^6 (top-left), 8×10^6 (top-right), 32×10^6 points.

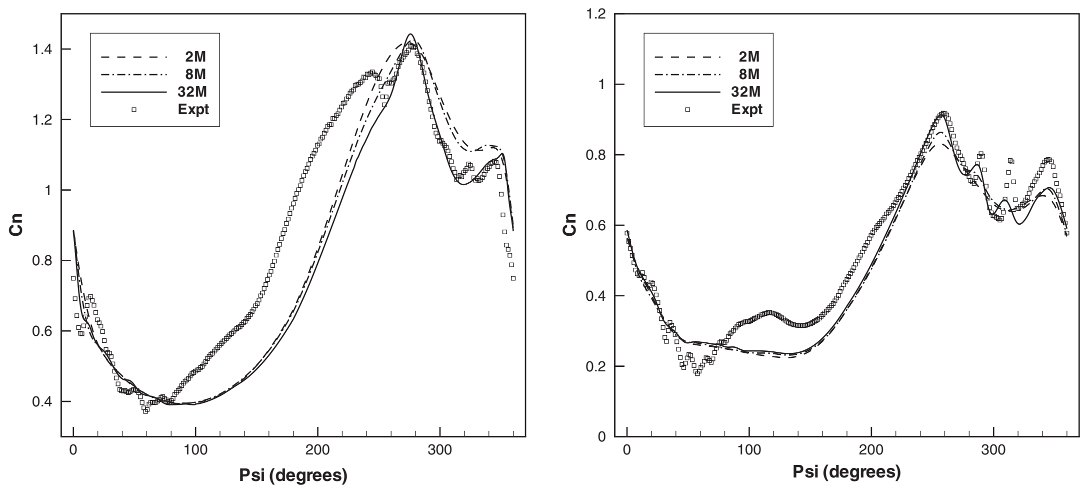

Figure 9. 7A Forward flight sectional blade loading. $r/R = 0.50$ and 0.82.

Vorticity contours (or shading) are often used for visualization. However, this is not always ideal, particularly for this type of flow where the rotor hub can create significant vorticity. Hence, the λ_2 quantity [24, 25] has been used here instead. This can identify local pressure minima associated with spin alone, so can be used to identify vortex cores more accurately than simply using local vorticity magnitude. Figure 8 shows an isosurface of λ_2 (at a small negative value) for the 2, 8, and 32 million point grid densities, at the $\Psi = 0^\circ$ position. This shows the paths of the tip vortices. The grid dependence is also clear, with more details emerging, due to preservation of the vortical wake over longer distances, as the density is increased.

To investigate this further, the sectional blade load coefficient was evaluated around the azimuth, at two radial stations. The results for the three mesh densities are compared with experimental data in Figure 9. Hence, the general trends are captured well, even for the coarsest mesh, but more details are beginning to emerge at high densities.

8. CONCLUSIONS

An implicit unsteady, multiblock, multigrid, upwind scheme, with deforming mesh capability, has been developed and validated, and used to compute four-bladed lifting test cases in hover and forward flight. A structured multiblock grid generation algorithm for rotors has also been developed and presented. Furthermore, to allow the use of fine meshes, vital for these type of flows, the flow code has been parallelized.

For hovering rotor solutions it has been shown that once sufficient points are used to capture enough turns of the wake, the solution does not converge to a steady state solution, instead exhibiting oscillatory wakes. This agrees with other published results using less diffusive formulations, theoretical stability analysis, and experiments. An unsteady simulation of the same case has also been performed, and detailed analysis of the time-accurate wake history has shown that three distinct modes have been captured, with frequencies of 1, 4, and 8 times the rotational frequency.

However, other unsteady modes, for example, those associated with vortex pairing, have not been captured. Hence, it must be concluded that the wake is indeed unstable to perturbations, as predicted by theory, but the instabilities are due to the numerics rather than the physics.

Forward flight simulations using the same rotor have also been presented, and again significant grid dependence of the vortical wake capture demonstrated. The local λ_2 value has been used for visualization of this flow, and is shown to be very useful to identify vortex trajectories, rather than the local vorticity vector magnitude that is often used.

Parallel performance of the code has been presented, showing that almost linear scaling has been achieved using up to 1024 CPU's.

ACKNOWLEDGEMENTS

The author would like to thank postgraduate student James Rodgers for his contribution to the grid generation codes and Agusta-Westland, who funded part of the grid generation research, for their continued support. Also EPSRC (the Engineering and Physical Sciences Research Council) for funding time on HPCx, through EPSRC grant EP/S91123/01, and Roderick Johnstone, of Daresbury Laboratories, for his help with the λ_2 work.

.

REFERENCES

1. Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. *MPI*: *The Complete Reference*. The MIT Press: Cambridge, MA, 1996.
2. Landgrebe AJ. The wake geometry of a hovering rotor and its influence on rotor performance. *Journal of the American Helicopter Society* 1972; **17**(4):2–15.
3. Tangler JL, Wohlfeld RM, Miley SJ. An experimental investigation of vortex stability, tip shapes, compressibility, and noise for hovering model rotors. *NASA CR-2305*, 1973.
4. Martin PB, Bhagwat MJ, Leishman JG. Strobed laser-sheet visualization of a helicopter rotor wake. *Paper PF118*, *Proceedings of PSFVIP-2*, Honolulu, HI, 1999.
5. Caradonna F, Hendley E, Silva M, Huang S, Komerath N, Reddy U, Mahalingam R, Funk R, Wong O, Ames R, Darden L, Villareal L, Gregory J. Performance measurements and wake characteristics of a model rotor in axial flight. *Journal of the AHS* 1999; **44**(2):101–108.
6. Gupta BP, Loewy RG. Theoretical analysis of the aerodynamic stability of multiple, interdigitated helical vortices. *AIAA Journal* 1974; **12**(10):1381–1387.
7. Jain R, Conlisk AT, Mahalingam R, Komerath NM. Interaction of tip-vortices in the wake of a two-bladed rotor. *Proceedings of the 54th Annual AHS Forum*, Washington, DC, 1998.
8. Bhagwat MJ, Leishman JG. On the aerodynamic stability of helicopter rotor wakes. *Proceedings of the 56th Annual AHS Forum*, Virginia Beach, VA, May 2000.
9. Chung KH, Na SU, Jeon WH, Lee DJ. A study of rotor tip-vortex roll-up phenomenon by using time-marching free-wake method. *Proceedings of the 56th Annual AHS Forum*, Virginia Beach, VA, May 2000.
10. Brown RE, Line AJ, Ahlin GA. Fuselage and tail-rotor interference effects on helicopter wake development in descending flight. *Proceedings of the 60th American Helicopter Society Annual Forum*, Baltimore, MA, 2004.
11. Line AJ, Brown RE. Efficient high-resolution wake modelling using the vorticity transport equation. *Proceedings of the 60th American Helicopter Society Annual Forum*, Baltimore, MA, 2004.
12. Parpia IH. Van-leer flux-vector splitting in moving coordinates. *AIAA Journal* 1988; **26**:113–115.
13. Anderson WK, Thomas JL, Van-Leer B. Comparison of finite volume flux vector splittings for the Euler equations. *AIAA Journal* 1986; **24**:1453–1460.
14. Obayashi S. Freestream capturing for moving coordinates in three dimensions. *AIAA Journal* 1992; **30**(4):1125–1128.
15. Allen CB. Multigrid convergence of inviscid fixed- and rotary-wing flows. *International Journal for Numerical Methods in Fluids* 2002; **39**(2):121–140.

16. Allen CB. An unsteady multiblock multigrid scheme for lifting forward flight simulation. *International Journal for Numerical Methods in Fluids* 2004; **45**(7):973–984.
17. Jameson A. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA Paper 91-1596*.
18. Thomas PD, Lombard CK. Geometric conservation laws and its application to flow computations on moving grid. *AIAA Journal* 1979; **17**(10):1030–1037.
19. Allen CB. CHIMERA volume grid generation within the EROS code. *I. Mech. E. Journal of Aerospace Engineering* 2000; **214**:125–141.
20. Gordon WJ, Hall CA. Construction of curvilinear coordinate systems and applications of mesh generation. *International Journal for Numerical Methods in Engineering* 1973; **7**:461–477.
21. Eriksson LE. Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation. *AIAA Journal* 1982; **20**(10):1313–1320.
22. Thompson JF. A general three dimensional elliptic grid generation system on a composite block-structure. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**:377–411.
23. Schultz K-J, Splettstoesser W, Junker B, Wagner W, Scheoll E, Arnauld G, Mercker E, Fertis D. A parametric wind tunnel test on rotorcraft aerodynamics and aeroacoutics (HELISHAPE)—test documentation and representative results. *22nd European Rotorcraft Forum*, Brighton, U.K., 1996.
24. Jeong J, Hussain F. On the identification of a vortex. *JFM* 1995; **285**:69–94.
25. Müller K, Rist U, Wagner S. Enhanced visualization of late-stage transitional structures using vortex identification and automatic feature extraction. *Proceedings ECCOMAS Congress*, 1998.