INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **45**:973–984 (DOI: 10.1002/fld.711)

An unsteady multiblock multigrid scheme for lifting forward flight rotor simulation

C. B. Allen*,†

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

SUMMARY

Numerical simulation of multi-bladed lifting rotors in forward flight is considered. The flow-solver presented is multiblock and unsteady, which is essential for forward flight, and also includes multigrid acceleration to reduce run-times. A structured multiblock grid generator specifically for rotor blades has also been developed and is presented here. Previous work has shown that hovering lifting rotor flows are particularly expensive to simulate, since the capture of the vortical wake below the disc requires a long numerical integration time; more than 20 revolutions for an unsteady simulation, or more than 40 000 time-steps for a single grid steady simulation. It is demonstrated here that only two or three revolutions are required to obtain a converged solution for forward flight, since the wake is swept downstream. This requires less than $1.5\times$ the run-time of a steady hovering simulation, for the same grid density around each blade, even though an unsteady simulation is required and the complete disk must be solved rather than one blade as in hover. It is demonstrated that very fine meshes are required to capture the unsteady tip vortex motion, and the effects on blade loading of blade–vortex interaction and rotor shaft inclination are also considered. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: numerical simulation; unsteady flows; grid generation; rotor flows; forward flight

1. INTRODUCTION

Numerical simulation of the flow about a helicopter rotor places severe demands on a CFD code, due to the non-linear, highly three dimensional, and unsteady flow regime. Hover simulation requires the accurate capture of the vortical wake over several turns, to obtain the correct wake influence on the blade loading. Forward flight simulation also requires accurate capture of the vortical wake, but this is now unsteady, and can easily be diffused by the numerical scheme. Hence, hover can be simulated with a steady code, but a very long integration time is required for the wake to develop, while forward flight requires a more expensive unsteady code but, since the wake is swept downstream, a shorter numerical integration time should be necessary. The requirements on the numerical mesh are fundamentally different for a rotary-wing flow than a fixed-wing flow, since the vortical wake capture requires a much higher grid

*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

Published online 4 May 2004

Received 23 June 2003
Revised 22 January 2004

density away from the surface than for a fixed-wing case, where far-field mesh and solution are normally of little significance and so this, combined with the long integration time, means rotary-wing run-times can be impractical. Hence, these flows are ideal for parallelization [1] and/or a multigrid approach to reduce run-times [2].

Hovering flight can be simulated as a steady problem, in a blade-fixed rotating co-ordinate system, but forward flight will always require a full unsteady simulation. Furthermore, forward flight also requires a multiblock mesh (if using structured meshes), whereas hover can be simulated with a single block mesh. Previous work [3–5] has shown that hovering solutions on multiblock meshes exhibit sharper wake resolution and better convergence than on single block meshes with similar numbers of points, due to the increased grid quality. It was also shown that simulating hovering flow with an unsteady scheme was significantly more expensive than with a steady solver (in a blade-fixed rotating co-ordinate system). More than 20 revolutions were required for convergence, which required almost six times the CPU time of the steady solution. However, it was summised that this was a feature of the case, wherein many turns of the wake need to be captured before convergence, and that forward flight would be cheaper since the wake is swept downstream and capture of many turns is not essential. This paper examines this supposition, by considering numerical simulation of lifting forward flight using an unsteady flow-solver with multiblock meshes.

It has been demonstrated previously [7] that the numerical dissipation used by central-difference schemes has a significant effect on hovering solutions, in terms of wake capturing and hence convergence rate. An upwind Euler solver is used in the current work, since this accurately models the physics of the flow, in terms of characteristic behaviour, and so is naturally dissipative. A finite-volume solver is presented, based on the flux-vector splitting of Van-Leer [8].

In this paper, the unsteady flow-solver is presented, followed by grid generation aspects for structured multiblock grids for rotors in forward flight. Numerical solutions for two-bladed rotors in forward flight are then presented and examined. Tip vortex trajectories, and the influence of the vortical wake on the loading of the following blade is also considered.

2. UNSTEADY SOLVER

For forward flight simulation a full unsteady simulation is essential. The Euler equations in integral form for a mesh rotating in a fixed axis system are

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{U}\,\mathrm{d}V + \int_{\partial V} \mathbf{F}.\mathbf{n}\,\mathrm{d}S = 0 \tag{1}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho[\mathbf{q} - (\boldsymbol{\omega} \times \mathbf{r}(t))] \\ \rho u[\mathbf{q} - (\boldsymbol{\omega} \times \mathbf{r}(t))] + P\mathbf{i} \\ \rho v[\mathbf{q} - (\boldsymbol{\omega} \times \mathbf{r}(t))] + P\mathbf{j} \\ \rho w[\mathbf{q} - (\boldsymbol{\omega} \times \mathbf{r}(t))] + P\mathbf{k} \\ E[\mathbf{q} - (\boldsymbol{\omega} \times \mathbf{r}(t))] + P\mathbf{q} \end{bmatrix} \tag{2}$$

In the above V is the domain, ∂V the domain boundary, $\mathbf{n}$ the outward unit normal, and dS an elemental area of the boundary. The co-ordinate vector $\mathbf{r}(t)=[x, y, z]^{\mathrm{T}}(t)$ is time dependent in the fixed axis system, i.e.

$$\mathbf{r}(t)=[\mathbf{R}(t)][x, y, z]^{\mathrm{T}}(0) \tag{3}$$

where $[\mathbf{R}(t)]$ is the time-dependent rotation matrix. The z-axis is taken as the rotation axis here, and so

$$[\mathbf{R}(t)]=\begin{bmatrix} \cos(\Omega_z t) & \sin(\Omega_z t) & 0 \\ -\sin(\Omega_z t) & \cos(\Omega_z t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The equation set is closed by

$$P=(\gamma-1)\left[E-\frac{\rho}{2}\mathbf{q}^2\right] \tag{5}$$

2.1. Spatial discretization

A finite-volume upwind scheme is used to solve the integral form of the Euler equations (Equation (1)), since by correctly modelling the characteristic behaviour of the flow upwind schemes are naturally dissipative. The flux-vector splitting of Van-Leer [8, 9] is used.

The flux integral for each cell is evaluated by defining a local orthogonal axis system at each cell face, and summing the upwinded components. The upwind interpolations are computed using a third-order spatial interpolation [11]. High-order schemes suffer from spurious oscillations in regions of high-flow quantity gradients, and so a flux limiter is required, and the continuously differentiable one due to Anderson *et al.* [11] was chosen.

To avoid introducing errors due to the evaluation of the rotational terms, the local $(\boldsymbol{\omega}\times\mathbf{r})$ terms are evaluated by solving for cell face area moment vectors, and satisfying the resulting conservation condition [10, 2, 6]. (More details of the spatial scheme can be found in References [2, 6].)

2.2. Implicit time-stepping scheme

An implicit form of the differential equation for each computational cell is considered,

$$\frac{\partial(V^{n+1}\mathbf{U}^{n+1})}{\partial t}+\mathbf{R}(\mathbf{U}^{n+1})=0 \tag{6}$$

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 [12]. However, the computations are started by moving the blade into a stationary fluid, and so a small initial time-step is required. This time-step can be increased during the computation, so a variable time-step scheme is adopted. Δt^{n+1} is the time-step from time level n to $n+1$, and Δt^n the time-step from time level $n-1$ to n. The 'pseudo-time' formulation is used to reduce the unsteady problem to a series of steady problems. A multi-stage

time-stepping scheme is then used to converge the solution at each real time step. For example integrating from pseudo-time-level m to $m+1$, the scheme would be

$$\mathbf{U}^{m+\alpha_j} = \mathbf{U}^m - \alpha_j \frac{\Delta\tau}{V^{n+1}} \left\{ T^{n+1} V^{n+1} \mathbf{U}^{m+\alpha_{j-1}} - T^n V^n \mathbf{U}^n \right.$$
$$\left. + T^{n-1} V^{n-1} \mathbf{U}^{n-1} + \sum_{k=1}^{6} [R]_k^{-1} [\bar{\mathbf{F}}^+ (\mathbf{U}^+)_k^{m+\alpha_{j-1}} + \bar{\mathbf{F}}^- (\mathbf{U}^-)_k^{m+\alpha_{j-1}}] A_k^{n+1} \right\} \tag{7}$$

where

$$T^{n+1} = \frac{(2\Delta t^{n+1} + \Delta t^n)}{(\Delta t^{n+1} + \Delta t^n)\Delta t^{n+1}} \tag{8}$$

$$T^n = \frac{(\Delta t^{n+1} + \Delta t^n)}{\Delta t^{n+1} \Delta t^n} \tag{9}$$

$$T^{n-1} = \frac{\Delta t^{n+1}}{(\Delta t^{n+1} + \Delta t^n)\Delta t^n} \tag{10}$$

and $\alpha_{0,1,2,3} = 0, \frac{1}{4}, \frac{1}{2}, 1$. k represents the six cell faces, $\Delta\tau$ is the pseudo-time-step, $[R]$ is the local rotation matrix, $\bar{\mathbf{F}}^{\pm}$ are the upwinded flux components, and A the cell face area. There is no limit to the size of the real time step that can be taken and this leads to a large reduction in CPU time compared to an explicit scheme [13–15]. The time step is limited by accuracy rather than stability, which is the reason small time-steps are used at the beginning of the computation. This approach means that the instantaneous grid positions and speeds, and the geometric quantities (normal vectors, area moment vectors, etc.), only have to be recomputed once every real time-step, and remain constant during the pseudo-time iterations. As there is no blade motion, i.e. no motion in addition to the rigid rotation, accounted for here, the cell volumes are constant for this case.

2.3. *Multigrid scheme*

Explicit time-stepping schemes lend themselves well to multigrid acceleration, see for example References [16, 17]. In this approach, errors computed on the finest mesh are transferred to progressively coarser meshes to compute the corrections to the fine mesh solution. As larger time-steps are allowed on the coarser meshes, errors are propagated out of the domain faster than is possible on the fine mesh, resulting in faster convergence. It has been shown previously that multigrid is effective for hovering rotors using a steady approach [2, 6]. An unsteady solver is used here, but an explicit-type solver is used within each real time-step, and so multigrid should also be effective here.

The scheme is presented here using N to represent the mesh number ($N=1$ is finest), V is the cell volume, $\hat{\mathbf{U}}$ and $\tilde{\mathbf{U}}$ represent the restricted and smoothed (updated) solution respectively, and I_N and I^N are the number of iterations performed on mesh N in the decreasing and increasing mesh density directions. $\mathbf{R}$ is used to represent the

residual, i.e.

$$\mathbf{R}(\mathbf{U}^{m+\alpha_{j-1}},\mathbf{U}^n,\mathbf{U}^{n-1}) = T^{n+1}V^{n+1}\mathbf{U}^{m+\alpha_{j-1}} - T^nV^n\mathbf{U}^n + T^{n-1}V^{n-1}\mathbf{U}^{n-1} + \sum_{k=1}^{6}[R]_k^{-1} [\overline{\mathbf{F}}^+(\mathbf{U}^+)_k^{m+\alpha_{j-1}} + \overline{\mathbf{F}}^-(\mathbf{U}^-)_k^{m+\alpha_{j-1}}]A_k^{n+1} \tag{11}$$

Then for a simple 'V' cycle, the scheme can be written as (these operations are performed for every cell)

DO FOR $NB = 1 \rightarrow NBLOCK$

Perform I_1 iterations (smoothing sweeps) on mesh 1 (finest)

$$\mathbf{U}_1^{m+\alpha_j} = \mathbf{U}_1^m - \alpha_j \frac{\Delta t}{V_N^{n+1}}\mathbf{R}(\mathbf{U}_1^{m+\alpha_{j-1}},\mathbf{U}_1^n,\mathbf{U}_1^{n-1}), \quad j = 1\ldots3$$

$$\Rightarrow \tilde{\mathbf{U}}_1$$

set $\mathbf{f}_1 = 0$

DO FOR $N = 2 \rightarrow NMESH$

Restrict solution to next finest mesh ($\hat{}$ is meaningless for $N = 1$)

$$\hat{\mathbf{U}}_N = \frac{\sum_{\text{cell}=1}^{8}\tilde{\mathbf{U}}_{N-1}V_{N-1}^{n+1}}{\sum_{\text{cell}=1}^{8}V_{N-1}^{n+1}}, \quad \hat{\mathbf{U}}_N^n = \frac{\sum_{\text{cell}=1}^{8}\hat{\mathbf{U}}_{N-1}^nV_{N-1}^{n+1}}{\sum_{\text{cell}=1}^{8}V_{N-1}^{n+1}}, \quad \hat{\mathbf{U}}_N^{n-1} = \frac{\sum_{\text{cell}=1}^{8}\hat{\mathbf{U}}_{N-1}^{n-1}V_{N-1}^{n+1}}{\sum_{\text{cell}=1}^{8}V_{N-1}^{n+1}}$$

Evaluate forcing function

$$\mathbf{f}_N = \mathbf{R}_N(\hat{\mathbf{U}}_N,\hat{\mathbf{U}}_N^n,\hat{\mathbf{U}}_N^{n-1}) - \textstyle\sum_{\text{cell}=1}^{8}\{\mathbf{R}_{N-1}(\tilde{\mathbf{U}}_{N-1},\hat{\mathbf{U}}_{N-1}^n,\hat{\mathbf{U}}_{N-1}^{n-1}) - \mathbf{f}_{N-1}\}$$

Perform I_N iterations on mesh N

$$\mathbf{U}_N^{m+\alpha_j} = \mathbf{U}_N^m - \alpha_j\frac{\Delta t}{V_N^{n+1}}(\mathbf{R}_N(\mathbf{U}_N^{m+\alpha_{j-1}},\hat{\mathbf{U}}_N^n,\hat{\mathbf{U}}_N^{n-1}) - \mathbf{f}_N), \quad j = 1\ldots3$$

$$\Rightarrow \tilde{\mathbf{U}}_N$$

ENDDO

DO FOR $N = NMESH - 1 \rightarrow 1$

Compute corrections on coarser mesh

$$\Delta\mathbf{U}_{N+1} = \tilde{\mathbf{U}}_{N+1} - \hat{\mathbf{U}}_{N+1}$$

Pass, prolong, corrections to current mesh

$$\Rightarrow \Delta\mathbf{U}_N$$

$$\tilde{\mathbf{U}}_N = \hat{\mathbf{U}}_N + \Delta\mathbf{U}_N$$

Perform I^N iterations on mesh N

$$\mathbf{U}_N^{m+\alpha_j} = \mathbf{U}_N^m - \alpha_j\frac{\Delta t}{V_N^{n+1}}(\mathbf{R}_N(\mathbf{U}_N^{m+\alpha_{j-1}},\hat{\mathbf{U}}_N^n,\hat{\mathbf{U}}_N^{n-1}) - \mathbf{f}_N), \quad j = 1\ldots3$$

$$\Rightarrow \tilde{\mathbf{U}}_N$$

ENDDO

ENDDO

It has been shown previously [6] that a V-cycle with relaxed trilinear prolongation operator is the most effective scheme for the single block steady version of current code, and that was also found for multiblock and unsteady simulations.

Table I. Boundary condition tags.

−1	Solid surface (include in loads calculation)
0	Solid surface
1	Far field
2	Internal face
3	Periodic downstream
4	Periodic upstream

At block and domain boundaries, suitable values of $\Delta\mathbf{U}$ are prescribed in halo cells, such that the relevant boundary conditions are satisfied and the same trilinear interpolation scheme can be used for every point.

It should be noted here that the multiblock nature means that fewer levels are possible. Only four mesh levels were used here even though 4 million points are used. Five or six levels could be used for 4 million points in a single block grid.

2.4. Boundary conditions

The solver is coded such that each block boundary has a boundary condition tag (listed in Table I), a neighbouring block number, and an orientation flag. The only restriction applied is that a boundary can only have one type of tag. This was done primarily to avoid possible connectivity/index problems when extending the code to include multigrid.

Tags 2, 3, and 4 require a neighbouring block number and orientation flag. At upper, lower, and spanwise far-field boundaries, characteristic based conditions are applied, accounting for moving boundary speeds.

3. GRID GENERATION

Hovering flight can be modelled using a single block mesh, as only one blade need be considered, see for example References [2, 3, 6]. However, forward flight simulation requires a multiblock mesh and, furthermore, it was shown in References [3, 4] that multiblock solutions exhibit better convergence and wake capturing than single block solutions.

The multiblock grid is generated by first generating a small extent blade-fixed C-H grid around the solid surface. This is generated in a similar manner to a single block grid, i.e. incorporating a periodic transformation [18, 3], but with a variable fraction of the disk and a small far-field distance. From this near blade grid, which has a rectangular type far field, 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.

A transfinite interpolation algorithm [19, 20] is used to compute the grid within each block. After generating the entire mesh an elliptic smoothing is applied [21]. The smoothing has been coded such that boundaries are also smoothed (for more details of the grid generation see References [5, 18]). The cylinder is then completed by rotating the computed mesh to fill the entire cylinder.

The major difference between meshes for hover and forward flight cases is representation of the hub. In the hovering case, it is simple to have the hub (small radius cylinder at the

root of all blades) running to the far-field boundaries above and below the blades, as there is no flow through it. That is not the case in forward flight, since there is flow through this region. The solid surface is taken to around one chord above and below the blades, and then the hub regions above and below the blades are filled by generating cylindrical meshes. A transformation is then applied to make the top and bottom of the solid surface spherical.

The geometry considered in the next section is that of the well-known Caradonna–Tung two-bladed rotor [22]. This is a rotor with no twist or taper, with a constant NACA0012 section. The hovering case has aspect ratio six, and blade incidence of 8°, and this is a standard test case. Hence, a forward flight simulation was also performed with this rotor. Plate 1(a) shows the solid surface used, i.e. the two blades and hub, and Plate 1(b) shows the solid surface mesh and block boundaries. The surface mesh on each blade is 129×65, and the local blade-attached mesh $129 \times 65 \times 33$. There are 36 blocks associated with each blade, and eight blocks above the hub and eight below, giving a total of 88 blocks, and around 4.0 million points in total. The grid density below the rotor disk is smaller than used for hover, since the wake is swept downstream rather than downwards. Plate 2(a) shows the block boundaries and the domain, and (b) shows the solid surface and the midplane. Plate 3(a) shows the solid surface plus selected planes near the blade tip, and block boundaries, and (b) the grid in the rotor disc. The far field for the mesh is set at 15 chords spanwise and 20 chords vertically, i.e. a cylinder of radius 15 chords, height 40 chords.

4. VALIDATION

The code was first validated for a simple forward flight case, the non-lifting Caradonna–Tung case, with aspect ratio 7 [23]. As wake capturing was unimportant here, the same blade-fixed mesh density was used, i.e. $129 \times 65 \times 33$ local volume mesh, but coarser blocks away from the blades. The block topology was the same, with 88 blocks and total number of points of around 1.4×10^6.

4.1. Non-lifting results

The case simulated was $M_{\text{Tip}} = 0.8$ and $\mu = 0.2$. Here μ is the advance ratio, or $M_{\text{FF}}/M_{\text{Tip}}$. This was run with 60 time-steps per rotation, i.e.

$$\Delta t = \frac{2\pi}{60\Omega_z} \tag{12}$$

and the initial time-step was $\frac{1}{5}$ of this, and was increased to the constant value over the first 10 time-steps.

Plate 4 shows surface relative Mach contours on the blade and hub solid surface, at 30° intervals, on the third revolution. The forward flight velocity of the rotor is in the negative x direction. (The simulations presented here and later are actually run by spinning the entire rotor mesh at Ω_z with a freestream onflow velocity of $(\mu M_{\text{Tip}}, 0, 0)$.)

Figure 1 shows computed pressure coefficient compared to experimental data [23]. The data is all for the radial station $r/R = 0.89$. This shows excellent agreement.

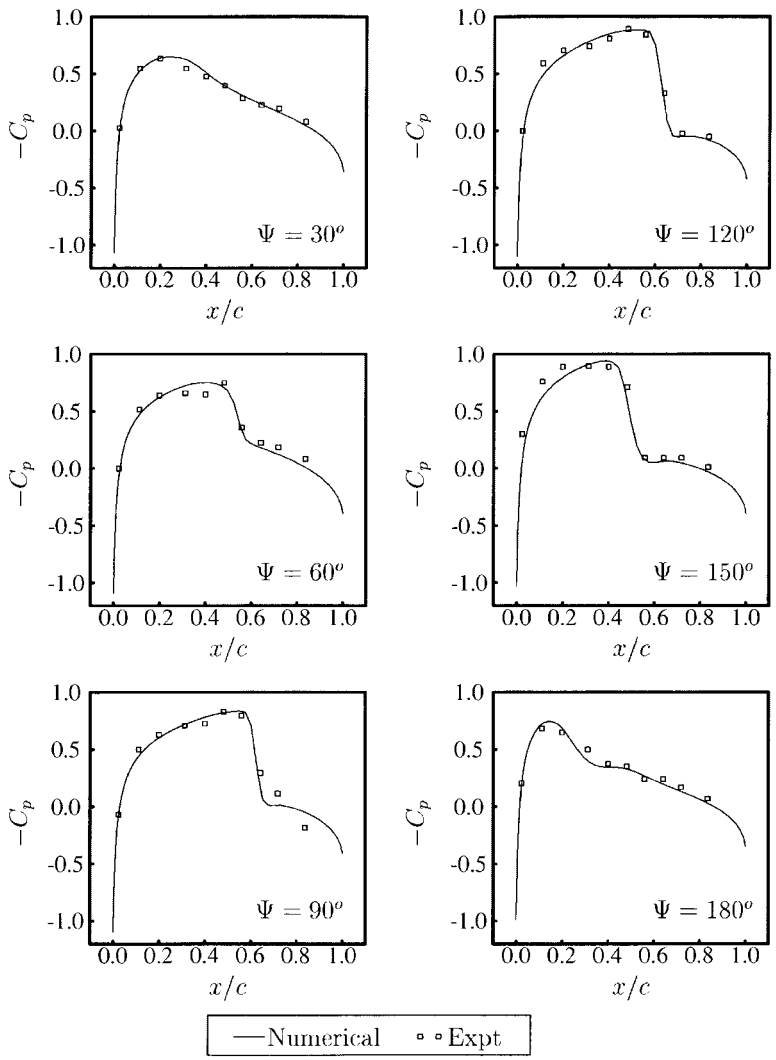


Figure 1. Numerical and experimental pressure coefficient.

5. LIFTING RESULTS

A lifting forward flight test case was run with the Caradonna–Tung two-bladed rotor. The standard hovering geometry was used, i.e. aspect ratio 6, 8° blade incidence. The tip Mach number was set at 0.7, and the advance ratio, μ, set to 0.2857, i.e. a 0.2 forward flight Mach number. Again 60 real time-steps were used per revolution.

The lifting cases considered here are not representative of a real rotor, where the pitch would vary around the azimuth to trim the rotor. The consideration of a correctly trimmed rotor is the next stage of the research, but the current results are presented to demonstrate the

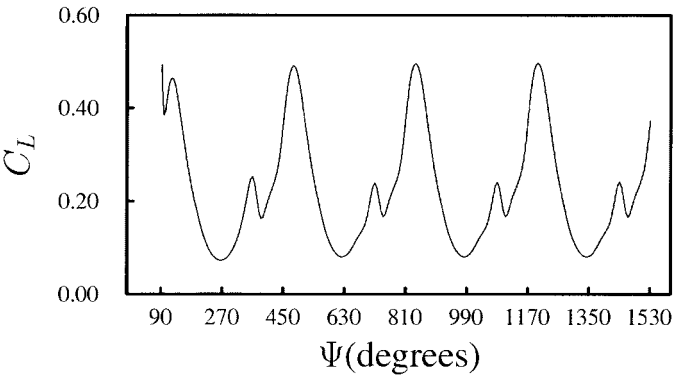


Figure 2. Force coefficient variation for one blade.

developed technology both in terms of grid generation and flow-solver, to examine the blade load variations, and consider wake capturing implications for forward flight simulations.

Plate 5 shows blade and hub upper surface relative Mach contours (57 contours are plotted between 0.0 and 1.5). The solution is at $\pi/3$ intervals over the third revolution.

It is interesting to consider the load variation on each blade around the azimuth. A force coefficient for each blade is defined as

$$C_L = \frac{F_z}{\frac{1}{2}\rho(\Omega R_{\mathrm{Tip}})^2 R_{\mathrm{Tip}} c} \tag{13}$$

where F_z is the force on the blade in the z direction. This is clearly related to the thrust coefficient, but will be labelled a lift, or load, coefficient to avoid confusion with the common use of thrust coefficient, which is normally taken as a value for the complete rotor for each flight condition.

Figure 2 shows the load history for one blade over four revolutions. Zero degrees is taken as pointing along the x-axis, i.e. away from the forward flight direction, and the simulation is started from the 90° position. The loss of loading due to the interaction of the blade with the tip vortex from the previous blade is clearly demonstrated here. The variation shows that the solution, in terms of blade loads, is periodic on the second revolution. Clearly, more than this may be needed to consider, for example, main rotor-tail rotor wake interaction but for blade loads the second revolution may be taken as the converged solution. This is to be expected, as the forward flight velocity means the wake is swept downstream. This is in sharp contrast to hovering simulations, where at least twenty revolutions were needed for the wake below the disc to develop over several turns [4].

Plate 6 shows the vorticity field when the lead blade is at 150°, 210°, and 270° rotation (V_{FF} shows the forward flight direction of the rotor.) This shows both the tip vortex path and the variation of tip vortex strength around the azimuth. The tip vortex shed on the advancing side is significantly stronger than on the retreating side. It is also worth noting that the vortex appears to be diffused, both in terms of strength and size, fairly quickly even though there are 4 million points. This has serious implications if main vortex-tail vortex interaction is to be simulated.

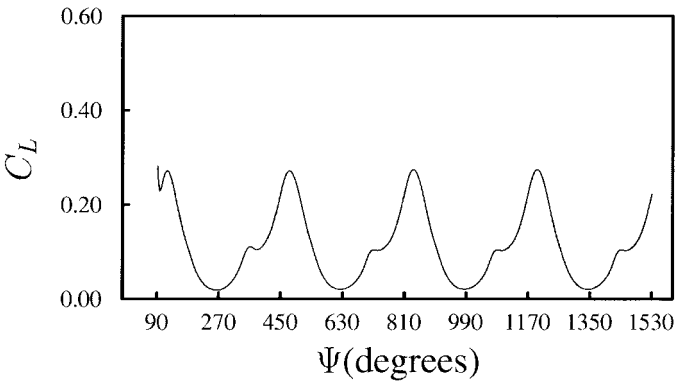


Figure 3. Force coefficient variation for one blade. shaft inclined 10°.

5.1. *More realistic case*

The previous case was run with forward flight direction normal to the rotor axis, i.e. there is no mechanism to generate forward thrust. As a more representative case, the previous case was run with the rotor shaft inclined forward at 10° to the z-axis, i.e. 17% of the thrust is in the forward direction, and the effect on the loading considered. Figure 3 shows the leading blade load coefficient history. As expected the loading is decreased due to the inclination and also, since the wake is swept further below the following blade, the loss of loading is less significant than in the previous case. Again the loading is periodic after the first revolution of the simulation.

5.2. *Computational requirements*

These simulations were run using 60 time-steps per revolution. With this time-step, using four-level multigrid, an average of around 70 effective pseudo-time-steps per real time step were required for convergence. This means that for two revolutions 120 time-steps were performed with 4 million points. This equates to around 3.3×10^{10} cell updates for a converged solution, whereas a two-bladed hovering solution using the steady form of the solver, also with four-level multigrid, required around 9000 time-steps with 2.5 million points, i.e. 2.3×10^{10} cell updates. Hence, forward flight simulation is only around $1.5\times$ the cost of hovering simulation for similar grid density around each blade, even though an unsteady solver must be used and the complete domain simulated. The entire simulation required around two weeks on a single EV6 500 MHz processor on a Compaq UNIX machine. (The code is currently being parallelized.)

6. CONCLUSIONS

Numerical simulation of multi-bladed lifting rotor flows in forward flight has been presented. An implicit unsteady multiblock multigrid scheme has been developed and validated, and used to compute forward flight test cases. A structured multiblock grid generation algorithm for rotors has also been presented.

For hovering rotor solutions it has been shown previously that over 20 revolutions need to be computed for convergence, as the vortical wake must be captured below the blades. However, it is demonstrated here that much fewer revolutions need to be computed in the forward flight case as the wake is swept downstream away from the blades, so the slow wake development demonstrated for hover is not a problem. In the cases considered here, only two revolutions are required, and this means forward flight is only around $1.5\times$ the cost of a steady hover simulation, even though an unsteady solver is required, and the complete rotor disk, including the hub regions, must be meshed. (The memory requirement is doubled, though.)

Results show the blade loading variation around the azimuth, the tip vortex path, and the effect the tip vortex has on blade loading. The disturbance in loading due to the influence of the vortex is shown to be significantly reduced when the rotor shaft is inclined forward. However, it is also demonstrated that the vortex diffuses fairly quickly behind each blade, and so to simulate the effects of the wake on the fuselage and/or the tail rotor is likely to require much finer meshes than considered here.

ACKNOWLEDGEMENTS

The author would like to thank Agusta-Westland Helicopters Ltd., who funded some of the grid generation research, for their continued support.

REFERENCES

1. Allen CB, Jones DP. Parallel implementation of an upwind Euler solver for inviscid hovering rotor flows. *Aeronautical Journal* 1999; **103**(1021):129–138.
2. Allen CB. Multigrid acceleration of an upwind Euler code for hovering rotor flows. *The Aeronautical Journal* 2001; **105**(1051):517–524.
3. Allen CB. Time-stepping and grid effects on convergence of inviscid rotor flows. *AIAA Paper 2002-2817*, *Proceedings 20th Applied Aerodynamics Conference*, St Louis, June 2002.
4. Allen CB. Convergence of steady and unsteady inviscid hovering rotor solutions. *International Journal for Numerical Methods in Fluids* 2003; **41**(9):931–949.
5. Allen CB. The effect of grid topology and density on inviscid hovering rotor solutions. *I M E Journal of Aerospace Engineering*, *Part G* 1999; **213**:81–95.
6. Allen CB. Multigrid convergence of inviscid fixed- and rotary-wing flows. *International Journal for Numerical Methods in Fluids* 2002; **39**(2):121–140.
7. Raddatz J, Rouzard O. Calculations of multibladed rotors in hover using 3D Euler methods of DLR and Onera. *Paper 11*, *21st European Rotorcraft Forum*, St. Petersburg, Russia, August 1995.
8. Van-Leer B. Flux-vector splitting for the Euler equations. *Lecture Notes in Physics*, vol. 170. Springer: Berlin, 1982; 507–512.
9. Parpia IH. Van-Leer flux-vector splitting in moving coordinates. *AIAA Journal* 1988; **26**:113–115.
10. Obayashi S. Freestream capturing for moving coordinates in three dimensions. *AIAA Journal* 1992; **30**(4): 1125–1128.
11. Anderson WK, Thomas JL, Van-Leer B. Comparison of finite volume flux vector splittings for the Euler equations. *AIAA Journal* 1986; **24**:1453–1460.
12. Jameson A. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA Paper 91-1596*.
13. Allen CB. The reduction of numerical entropy generated by unsteady shockwaves. *Aeronautical Journal* 1997; **101**(1001):9–16.
14. Allen CB. Grid adaptation for unsteady flow computations. *I M E Journal of Aerospace Engineering*, *Part G4*, 1997; **211**:237–250.
15. Gaitonde AL. A dual-time method for the solution of the unsteady Euler equations. *The Aeronautical Journal of the Royal Aeronautical Society* 1994; **98**:283–291.
16. Jameson A. Transonic Flow Calculations. Princeton University, *Report MAE* 1751, 1984.

17. Jameson A. Time-dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA Paper 91-1596*, 1991.
18. Allen CB. CHIMERA volume grid generation within the EROS code. *I M E Journal of Aerospace Engineering*, *Part G* 2000; **412**:125–141.
19. 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.
20. Eriksson LE. Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation. *AIAA Journal* 1982; **20**(10):1313–1320.
21. 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.
22. Caradonna FX, Tung C. Experimental and analytical studies of a model helicopter rotor in hover. *NASA TM*-81232, September 1981.
23. Caradonna FX, Laub GH, Tung C. An experimental investigation of the parallel blade-vortex interaction. *Proceedings of the 10th European Rotorcraft Forum*, The Hague, Netherlands, 1984; 4.1–4.30 (Also NASA TM-86005, 1984).

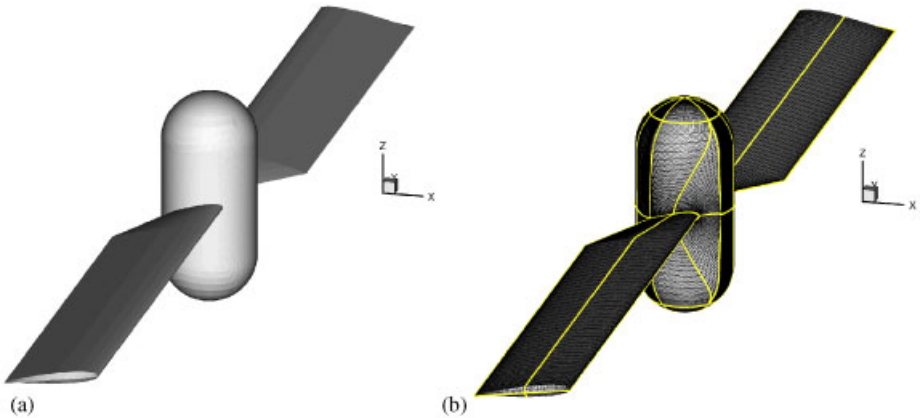


Plate 1. (a) Solid surface, (b) solid surface mesh and block boundaries.

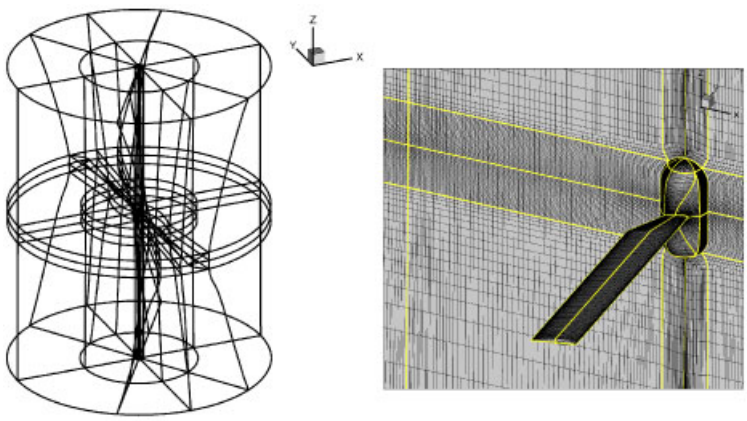


Plate 2. Block boundaries and solid surface and midplane.

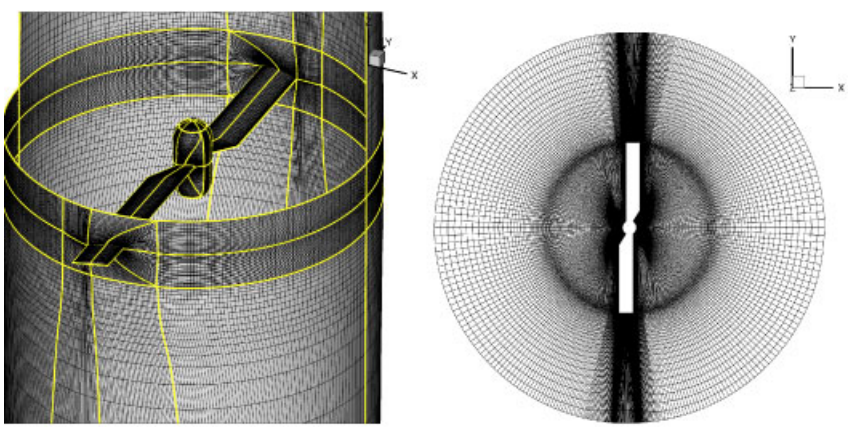


Plate 3. Solid surface and selected planes near tip and grid in rotor disc.

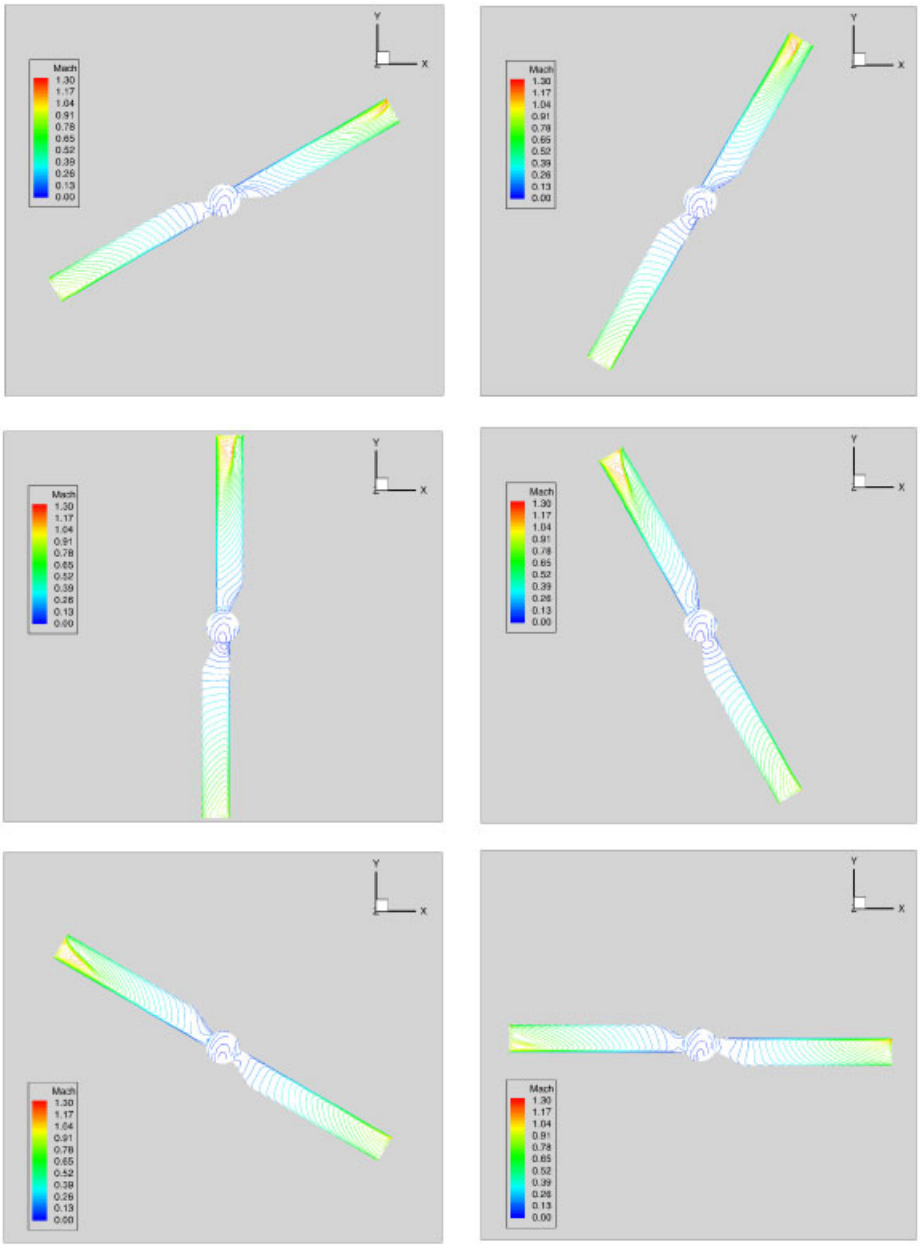


Plate 4. Solid surface relative mach contours.

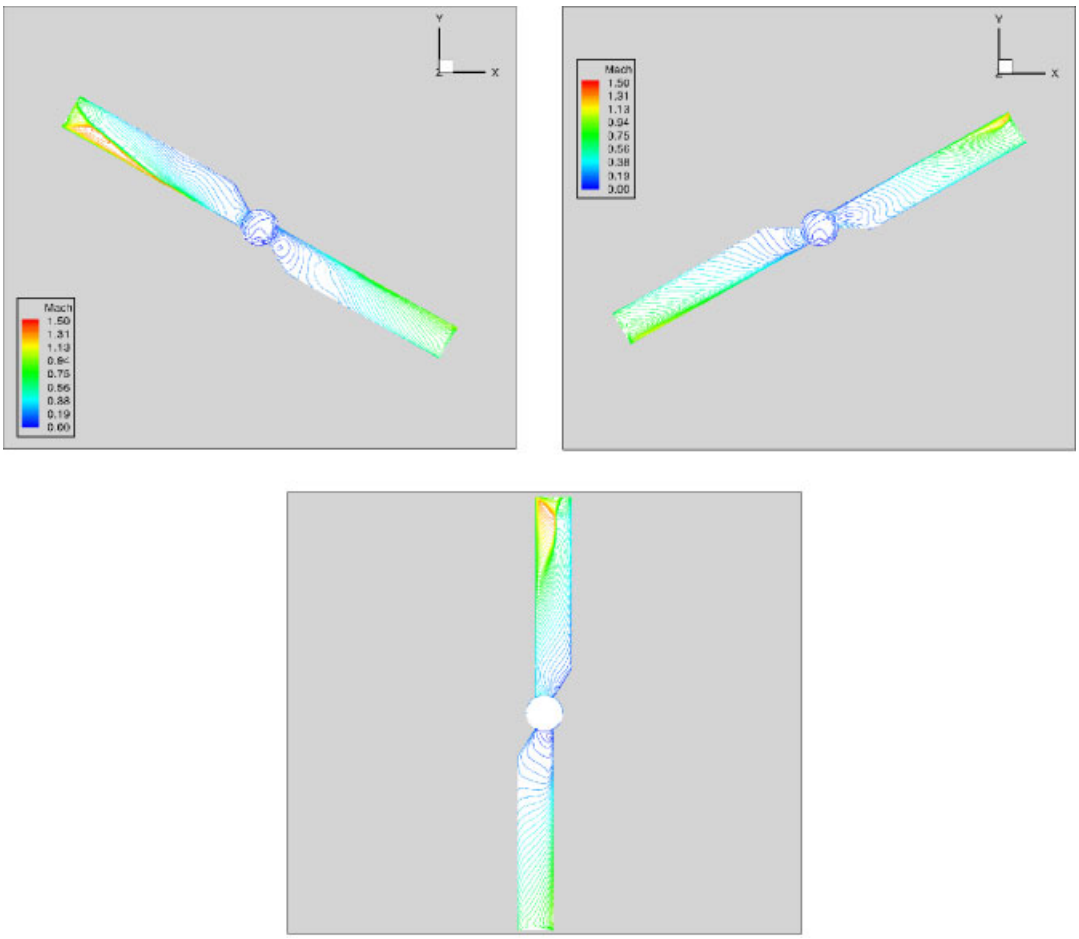
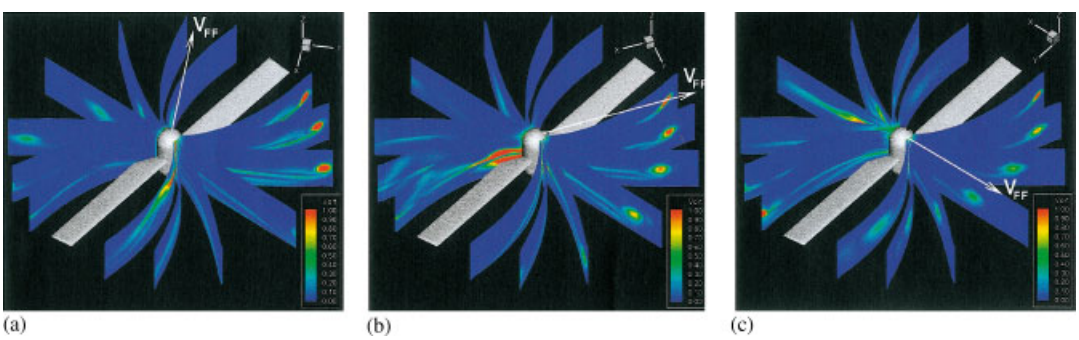


Plate 5. Solid surface mach contours over one revolution.



(a) (b) (c)

Plate 6. Vorticity field with lead blade at 150°, 210°, 270°.