INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 4, Number 1, Pages 1–15

PARALLEL SIMULATION OF LIFTING ROTOR WAKES IN FORWARD FLIGHT

C. B. ALLEN

Abstract. A finite-volume implicit, unsteady, multiblock, multigrid, upwind solver, and structured multiblock grid generator for rotors are presented, and applied to lifting rotors in forward flight. These flows are particularly expensive to compute as the accurate capture of the detailed vortical wake requires fine meshes away from the blades and, hence, a parallel version of the code has been developed allowing the use of very fine meshes. Parallel performance of the code is presented, and grid dependence of the computed blade loads and wake analysed, by considering wake capturing, total blade load and sectional load variation around the azimuth. It is demonstrated that the vortical wake capture is severely influenced by grid density, and even with 32 million points grid convergence is not approached. It is also shown that if blade loads only are of interest the vorticity dissipation is not a severe problem, and coarser meshes can be used. However, if more detail is required, for example blade-vortex interaction or aero-acoustic analysis, it would appear very difficult to capture the wake to any reasonable accuracy.

Key Words. Numerical simulation, forward flight rotors, parallel processing, wake capturing.

Nomenclature

a_∞	Freestream acoustic speed
A	Cell face area
c	Rotor chord
C_L	Blade load coefficient
C_{load}	Blade sectional load coefficient
E	Energy
F_z	Force in z-direction
$\mathbf{F}$	Flux vector
$\overline{\mathbf{F}}^{\pm}$	Upwinded flux vector components
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Unit vectors in x, y, z directions
k	Cell face index
$\mathbf{q}$	Absolute velocity vector $= [u, v, w]^T$
M_{Tip}	Rotor tip Mach number ($= \Omega R_{Tip}/a_\infty$)
$\mathbf{n}$	Outward unit normal vector
P	Pressure
$\mathbf{r}$	Coordinate vector
$\mathbf{R}$	Residual vector
r	Section radius
R	Rotation matrix
R_{Tip}	Rotor tip radius

Received by the editors May 10, 2005 and, in revised form, November 17, 2005.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.

S	Surface cell area
t	Time
T	Implicit time-stepping coefficients
u, v, w	Cartesian velocity components
x, y, z	Cartesian coordinates
$\mathbf{U}$	Conserved vector
V	Cell volume
α	Explicit time-stepping coefficient
μ	Advance ratio ($= V_{FF}/\Omega R_{Tip}$)
ω	Rotation vector
Ω	Rotational frequency
Ψ	Azimuth angle
ρ	Density
Θ_{inc}	Rotor shaft inclination angle

1. Introduction

Forward flight rotor flows are extremely challenging for numerical simulation codes, due to the particularly harsh flow regimes. The flow in the root region is very low speed, and can contain regions of reverse flow on the retreating side, while the flow in the tip region is normally transonic on the advancing side. The flow around each blade is also strongly influenced by the wake from previous blades, and so the capture of this wake is important. However, numerical diffusion/dispersion inherent in all CFD codes severely compromises the resolution of flow vorticity, and so this is a serious problem for rotor flow simulation. Hover simulation requires the capture of several turns of the tip vortices to compute accurate blade loads, resulting in the requirement for fine meshes away from the surface, and a long numerical integration time for this wake to develop. Forward flight simulation also requires accurate capture of the vortical wake but, depending on the advance ratio, fewer turns need to be captured, as the wake is swept downstream. However, not only does the entire domain need to be solved, rather than the single blade for hover, but the wake is now unsteady, and so an unsteady solver must be used, which is not only more expensive than the steady solver used for hover, but can easily result in even higher numerical diffusion of the wake. Hence, it is extremely expensive to simulate these flows.

Other, cheaper methods can be used for rotor simulation, for example it is common to use a wake model based on a vortex element or vortex-lattice method [1, 2]. However, these can also become expensive for fine resolutions, resulting in simplifying assumptions, often related to vortex roll-up, which can lead to incorrect representations. Another cheaper approach is the vorticity transport formulation, see for example [3, 4], which has very low inherent diffusion, and has been proven to preserve and convect vortices over extremely long distances. Simulations with this method have been demonstrated to capture both unsteady hovering wakes, and wake breakdown in forward flight. However, this approach is, as are free wake models, limited to incompressible flow.

Ultimately, for many aspects of rotor design and analysis, a mainstream CFD solution of the entire, compressible, flow field is desirable, and that approach is considered here for forward flight. Of particular interest is the development and capture of the unsteady vortical wake and, hence, a grid dependence study has been performed. To this end, the simulation code has been parallelised to allow the use of very fine meshes.

The amount of numerical diffusion/dispersion present and, hence, the capture of flow gradients, is of course inextricably linked to the quality of the numerical mesh used. Forward flight will always require a multiblock mesh (if using structured meshes), whereas hover can be simulated with a single block mesh. Previous work [5, 6, 7] has shown that hovering solutions on multiblock meshes exhibit much sharper wake resolution and better convergence than on single block meshes with similar numbers of points, due to the increased grid quality. This agrees with other work, where it has also been shown [8] that vorticity capturing is highly grid size, cell aspect ratio, and grid skewness dependent and, hence it is vital to adopt a multiblock approach, as is adopted here, to attempt to align the mesh with the vortical flow features as well as possible.

In this paper the unsteady formulation of the flow-solver will be presented, along with parallelisation details, followed by grid generation aspects for structured multi-block grids for rotors in forward flight. Numerical solutions for lifting rotors in forward flight are then presented and examined. Of particular interest is the grid dependence of the vorticity capturing, and the influence of the vortical wake capturing on the loading of the following blades. The key questions addressed are: how much of the physics can be captured, how much needs to be captured when considering global loads for example, and how expensive is it ? These questions are answered, and possible options, in terms of improving wake capturing, are discussed.

2. Unsteady Flow Solver

The Euler equations in integral form for a mesh rotating in a fixed axis system are

$$\frac{d}{dt}\int_V \mathbf{U}dV + \int_{\partial V} \mathbf{F}.\mathbf{n}dS = 0 \tag{1}$$

where

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

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

$$\mathbf{r}(t) = [R(t)]\mathbf{r}(0) \tag{3}$$

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

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

The equation set is closed by

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

2.1. Spatial Discretisation. A finite-volume upwind scheme is used to solve the integral form of the Euler equations, since by correctly modelling the characteristic behaviour of the flow upwind schemes are naturally dissipative. The flux-vector splitting of van Leer [9, 10] 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 $(\omega \times \mathbf{r})$ terms are evaluated by solving for cell face area moment vectors, and satisfying the resulting conservation condition [14, 12, 13]. (More details of the spatial scheme can be found in [12, 13].)

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 [15]. 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

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

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, $\underline{\overline{\mathbf{F}}}^{\pm}$ 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 [19, 16, 17]. 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 [20, 21]. 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, [12, 13], and for forward flight using an unsteady approach [26, 27, 28].

A V-cycle with relaxed trilinear prolongation operator was found to the most effective scheme for both steady and unsteady simulations. At block and domain boundaries, suitable values of solution updates 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.

2.4. Parallelisation. The code has been parallelised using MPI, to allow the use of fine meshes. However, there is a balance to be struck here, since multigrid is implemented: more CPU's means smaller blocks and, hence, faster solution time per global time-step, but smaller blocks mean fewer multigrid levels can be used, resulting in slower convergence. A pre-processing algorithm has been developed which reads in the mesh in its 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 optimise the mesh dimensions and load balancing.

The solver is coded such that each block boundary simply has a boundary condition tag (listed in table 1), a neighbouring block number, and an orientation flag. The only restriction applied is that a boundary can only have one type of tag.

-1	Solid surface (include in loads calculation)
0	Solid surface
1	Farfield
2	Internal face
3	Periodic downstream
4	Periodic upstream

Table 1: *Boundary Condition Tags.*

Hence, each block only requires NI, NJ, NK, the $NI \times NJ \times NK$ physical coordinates, then six lines, of three integers each, defining the boundary conditions. The spatial stencil is five points in each direction, and so for parallel send/receives, at each internal block boundary, two planes of solution are packed and sent to the appropriate processor, and received boundary data unpacked according to the orientation flag. This has the advantage that no halo, multigrid, or cell connectivity data is required in the grid file and, hence, there is no 'preprocessing' stage. This is significant, as it avoids the mesh having to be processed on a single CPU, which can result in a mesh size limit, or the development of a parallel grid generator/preprocessor. 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 been written to require 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. Avoiding any global storage or collection of flow/geometry data means the code is extremely efficient in terms of memory and load balancing; the 32 million point mesh case shown later can be run with four levels of multigrid on 16 dual-CPU nodes, with only 1GByte RAM per node, i.e. less than 0.5GByte/million points.

3. Grid Generation

Hovering flight can be modelled using a single block mesh, as only one blade need be considered, see for example [12, 5, 13]. However, forward flight simulation requires the complete disk, i.e. all blades, to be modelled, and, hence, a multiblock mesh is required (if using a structured approach). It has previously been shown [5, 6] that multiblock hovering solutions exhibit better convergence and wake capturing than single block solutions anyway, as the grid lines can be aligned more with the flow features.

Hence, a multiblock grid generation tool has been developed which is applicable to fixed- and rotary-wing cases, but is particularly effective for rotor flows. As the flow-solver has been parallelised there is, in theory, no limit on the mesh size that can be used and, hence, it is important that the grid generator does not place a limit on this. The software developed has been coded to be extremely efficient in terms of memory requirement and, as an example, a 64 million point, 408 block, mesh can be generated in around 30 minutes on a P4 machine running Linux, and requires less than 2GBytes RAM. A transfinite interpolation approach [23, 24] is adopted, including improved orthogonality and a periodic transformation.

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 farfield 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 intermediate blocks and blocks that match at the periodic boundaries. A transformation is then applied to make the top and bottom of the solid surface spherical. Previous versions of the grid generator adopted a radial type block structure above the hub, see for example [26, 27]. However, this has now been improved, so that all blocks are generated such that there are no ‘collapsed’ cells, i.e. cells with any zero area faces.

Once the mesh and hub blocks have been generated around one blade, the cylinder is completed by repeating and rotating the computed mesh to fill the entire cylinder. After generating the entire mesh an elliptic smoothing is applied [25]. The smoothing has been coded such that boundaries are also smoothed (for more details of the grid generation see [7, 22]).

A wake grid dependence study has been performed, considering the ONERA 7A blade [18]. A typical mesh density for a forward flight simulation may be one million points. To perform the grid dependence study, this density was doubled five times, i.e. meshes of density 1, 2, 4, 8, 16, and 32 million points were generated.

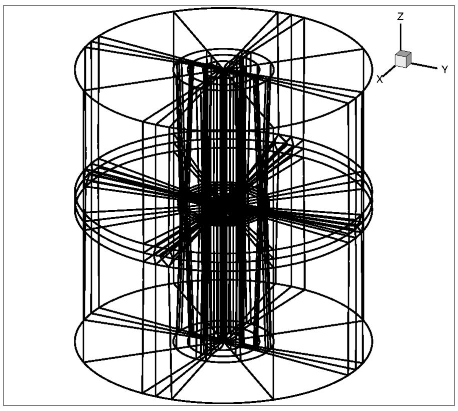


Fig. 1 : *Four-bladed Multiblock Forward flight mesh. Domain and Block Boundaries.*

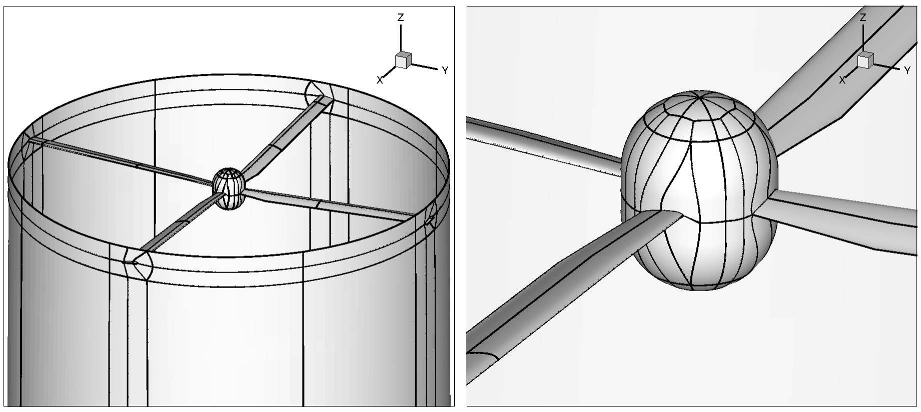


Fig. 2 : *Four-bladed Multiblock Forward flight mesh. Block Structure.*

Figure 1 shows the computational domain and block boundaries, for the 7A four-bladed case. This is an intermediate topology, with 72 blocks associated with each blade, and six blocks above the hub and six below for each blade, giving a total of 336 blocks. For the coarser meshes presented, this number is decreased to 192, and increased to 408 for the 32 million point case, to allow better parallelisation. Figure 2 left shows the block boundaries on the solid surface and a grid plane near the blade tips to show the block structure, and right is a close-up of the hub region, with block boundaries shown. The farfield for the mesh is set at 50 chords spanwise and 50 chords vertically, i.e. a cylinder of radius 50 chords, height 100 chords. The grid density above and below the rotor disk is much smaller than used for hover, since the wake is swept downstream rather than downwards, and so the grid density there is less significant.

These meshes were all individually generated, i.e. coarser meshes have not been generated by removing points from finer ones. This was done to ensure the highest possible grid quality, and also to maintain the maximum proportion of points in the rotor disk area, as this is the region of interest.

Figure 3 shows part of the grid in the rotor disk for each of the six grid densities.

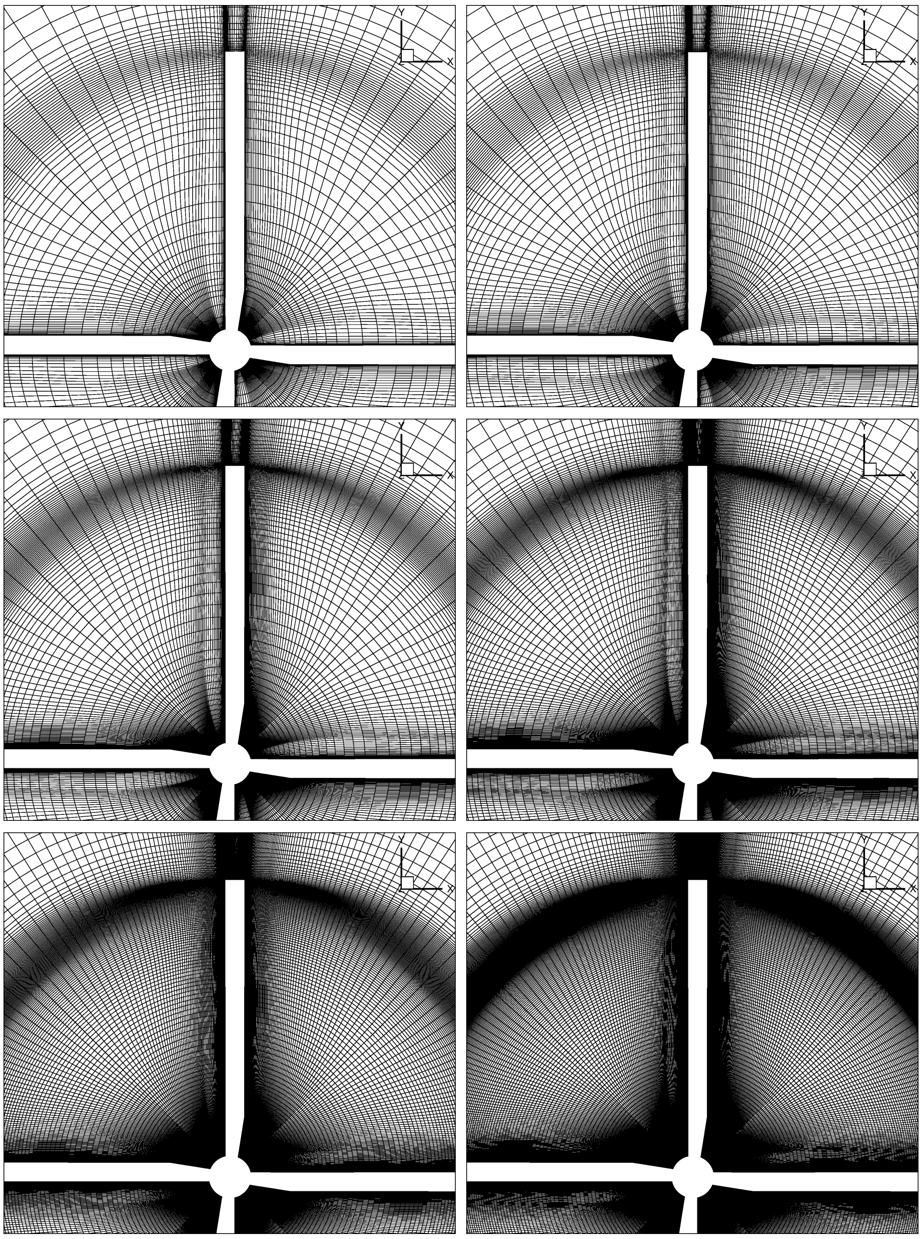


Fig. 3 : *Four-bladed Multiblock Forward flight mesh.*
Rotor disk grid, $1, 2, 4, 8, 16, 32 \times 10^6$ *points.*

4. 7A Rotor in Lifting Forward Flight

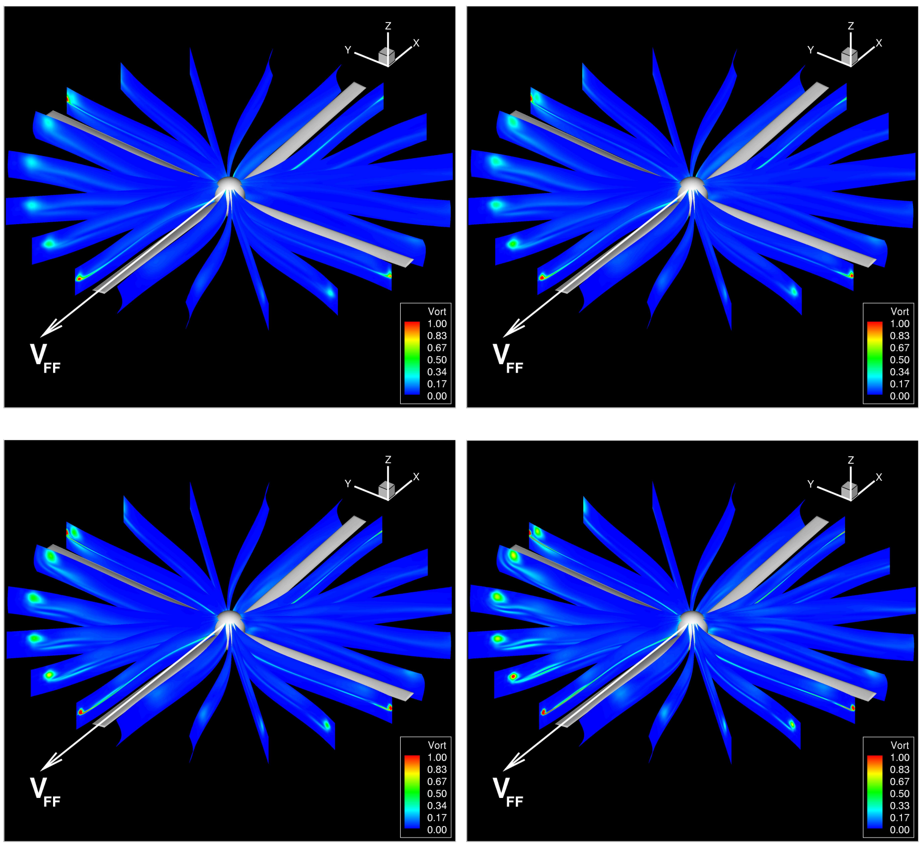
A lifting forward flight test case was run with the 7A rotor. (The code has previously been validated for forward flight, see for example [26, 27, 28].) The case corresponds to datapoint Dpt0165, where the tip Mach number is 0.618, and the advance ratio, μ, set to 0.214. The case also has a shaft inclination of -3.72 degrees, i.e. backward, so has significant blade-vortex interaction (BVI) effects. This test case in fact has a time-dependent blade pitch, but that is not considered here, and the case run with a rigid rotor. This then results in higher loading on the advancing side and, hence, stronger tip vortices in this region, which make a wake capturing study more meaningful. The inclusion of surface and mesh motion is an important stage of the development, but has little effect on the wake capturing grid dependence analysis considered here.

The case was run as an unsteady simulation, using 180 real time-steps per revolution, by spinning the entire rotor mesh at Ω_z with a uniform freestream onflow velocity of

$$\mathbf{q}_{freestream} = (\mu M_{Tip} a_\infty \cos\Theta_{inc}, 0, -\mu M_{Tip} a_\infty \sin\Theta_{inc})^T. \tag{11}$$

where Θ_{inc} is the rotor shaft inclination. As mentioned above, meshes of density 1, 2, 4, 8, 16, and 32 million points were used.

4.1. Results and Discussion. Figure 4 shows the vorticity field on selected cut-away planes in the rotor disc (V_{FF} shows the forward flight direction of the rotor), for the various grid densities (the scale is the same in all pictures). This is the solution after four complete revolutions have been computed (the rotation sense is anti-clockwise). This shows both the tip vortex path and the effect of numerical diffusion/dispersion. The grid dependence is clear, with even the 32 million point simulation exhibiting diffusion. This is as expected.



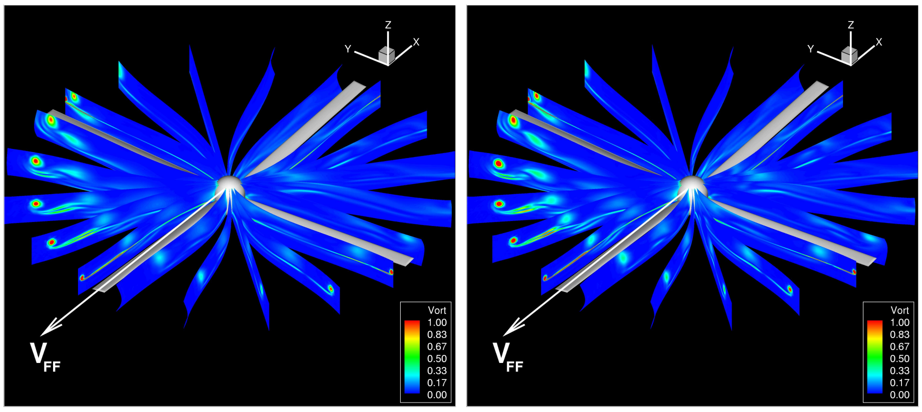


Fig. 4: *7A Forward Flight Vorticity Shading.*

Clearly, as density is increased more influence of the wakes of previous blades is captured, but more points are still needed. To overcome this problem, there are several possible options. Grid adaptation is an obvious option, see for example [29, 30], but this is expensive, and implementation within an unsteady solver difficult, particularly with the large motion of the tip vortices across the mesh. Structured mesh adaptation can be extremely expensive, and unstructured adaptation, i.e. adding and removing points, can lead to conservation problems, and can add dissipation by affecting the spatial stencil.

The 'vorticity confinement' technique of Steinhoff and co-workers [31]-[34], is another option. This has been applied to rotor flows, see for example [35, 36], but as yet there is no theoretical foundation for determining the confinement parameter value. It also does not help with the general diffusion issue, for example it will not help acoustic problems, where pressure waves need to be resolved.

Chimera overlapping meshes could also be used, where high density meshes can be added to capture the tip vortices, but to specifically capture the tip vortices in forward flight requires a large motion of the tip vortex mesh which, again, is expensive and awkward. This type of approach has been adopted in the UTC OVERFLOW code [37, 38], and applied to rotor flows, see for example [39, 40]. However, it was stated in [40] that there were tip vortex grid location convergence problems, which have yet to be resolved, even for hover.

Perhaps the best option, in terms of general diffusion/dispersion, is to use a high-order scheme, see for example [8] wherein model problems related to rotor tip vortices are considered. It was stated in this paper that *".... it is more efficient to use a higher-order scheme rather than to increase the number of points with a lower order method."*. However, it must be remembered that most of the published test cases are performed on cartesian-type meshes, and these conclusions are questionable for general skewed or stretched meshes, but this remains an interesting approach.

Figure 5 shows blade load coefficient around the azimuth, for the six grid densities, defined as:

$$C_L = \frac{F_z}{\frac{1}{2}\rho_\infty(\Omega R_{Tip})^2 R_{Tip} c} \tag{12}$$

where

$$F_z = \sum_{blade}^{surface\ cells} P A_z \tag{13}$$

where

$$A_z = \mathbf{n}.\mathbf{k}S \tag{14}$$

and **n** is the surface cell outward unit normal, **k** is the z-direction unit vectro, and S is the surface cell area. Ψ is the azimuth angle, and is defined as zero when the blade is aligned with the onflow, i.e. facing backward along the fuselage.

Hence, if only blade loads are required, the grid density is not as significant as if more detailed quantities are of interest.

Figure 6 shows normal force coefficient variation around the azimuth at r/R = 0.5, and figure 7 at r/R = 0.82. The normal force coefficient is defined at any section r as:

$$C_n = \frac{F_z}{\frac{1}{2}\rho_\infty(\Omega R_{Tip}\frac{r}{R_{Tip}} + \mu\Omega R_{Tip}\sin(\Psi))^2 c} \tag{15}$$

where, in this case

$$F_z = \sum_{blade}^{section\ cells} PdS_x \tag{16}$$

where

$$dS_x = \frac{A_z}{\Delta y} \tag{17}$$

and r is the section radius.

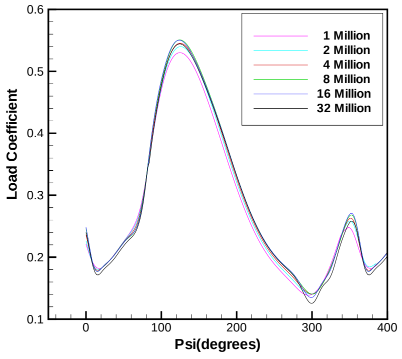


Fig. 5: *7A Forward Flight Blade Loading.*

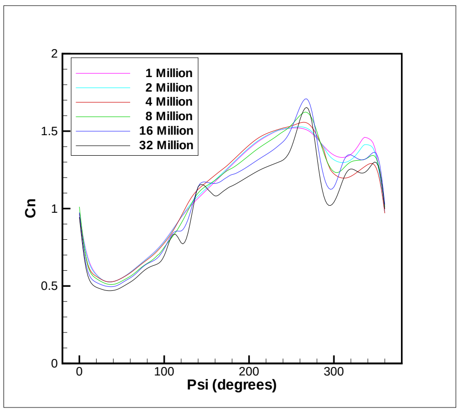


Fig. 6: *7A Forward Flight Section Normal Force Variation,* $r/R_{Tip} = 0.5$.

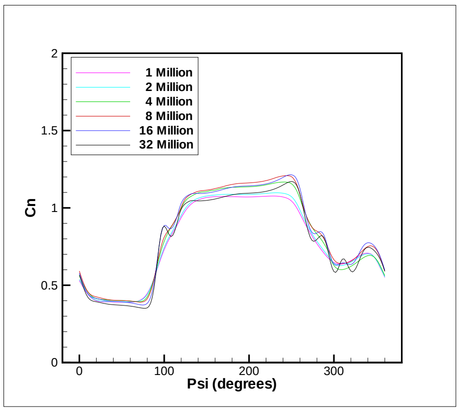


Fig. 7: *7A Forward Flight Section Normal Force Variation,* $r/R_{Tip} = 0.82$.

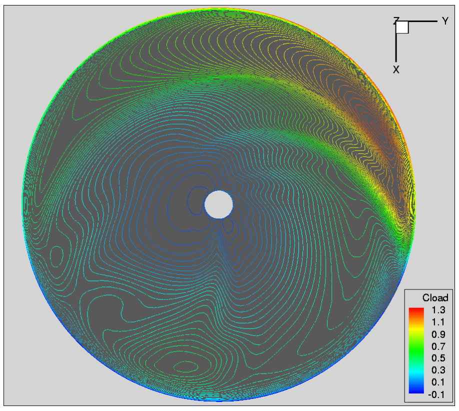


Fig. 8: *7A Forward Flight Blade Disk Loading.*

The blade disk loading for the 32 million point case is also shown if figure 8, which shows C_{load} around the azimuth, defined as:

$$C_{load} = \frac{F_z}{\frac{1}{2}\rho_\infty(\Omega R_{Tip})^2 c} \tag{18}$$

where F_z is as defined in (16).

This clearly shows the blade-vortex interactions, which are particularly strong on the advancing side.

Figures 5 and 6 demonstrate how, with less than 16 million points there is little, if any, evidence of blade-vortex interaction being captured. However, it is also clear that the solution is far from converging to a grid independent solution, even with 32 million points and, hence, more points are required.

4.2. Computational Requirements and Parallel Performance. The 1, 2, 4, 8, and 16 million point cases were run on the Beowulf cluster in the LACMS (Laboratory for Advanced Computation in the Mathematical Sciences), at the Department of Mathematics at Bristol. This consists of 80 nodes, each comprising two 1GHz P3's with 1GByte RAM, and these cases were run on upto 48 CPU's. The 16 million point case required around 14 days on 48 P3's to compute four revolutions. The 32 million point case was run on the national HPCx 1600 CPU machine. This consists of 50 nodes, each comprising 32 1.7GHz IBM CPU's with 32GBytes of shared memory. This case required four days on 256 CPU's to compute four revolutions, i.e. around 25000 CPU hours.

The code has been officially benchmarked by the Terascaling Team, at the National Supercomputer Centre. Figure 9 shows the speed-up obtained using from 32 upto 1024 CPU's for a 20 million point case [1]. A factor of one was used for 32 CPUs as fewer than this were not used. Hence, the code exhibits excellent scaling efficiency.

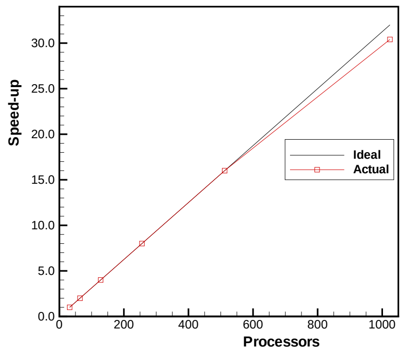


Fig. 9 : *Parallel Performance of the Code.*

5. Conclusions

Numerical simulation of multi-bladed lifting rotor flows in forward flight has been presented. An implicit, upwind, unsteady, multiblock, multigrid scheme has been developed, and used to compute lifting test cases. An efficient structured multiblock grid generation algorithm for rotors has also been developed and presented. Furthermore, the flow code has been parallelised to allow use of very fine meshes, and excellent parallel efficiency demonstrated.

It has been demonstrated that the numerical dissipation/dispersion inherent in the solver limits the accuracy of wake capturing. Even the solution computed using 32 million points exhibits wake diffusion/dispersion. It is shown that if only total blade loads are of interest this is not a huge problem, but if more detailed data is required, for example BVI effects are to be analised, it does not seem sensible to

[1]These figures have been certified by Andy Sunderland of the Terascaling Team at the National Supercomputing Centre

attempt to capture all the physics with a standard CFD code. Grid convergence is not approached even with a 32 million point simulation, which cost 25000 CPU hours.

Acknowledgements

The author would like to thank postgraduate student James Rodgers for his contribution to the grid generator, and Agusta-Westland, who funded part of the grid generation research, for their continued support. Thanks must also go to Dr's Ian Stewart and Jason Hogan-O'Neill, and Professor Steve Wiggins, of the University of Bristol for granting access to the LACMS (Laboratory for Advanced Computation in the Mathematical Sciences) 160 CPU Beowulf cluster, and to EPSRC for funding time on the HPCx 1600 CPU machine for the UK Applied Aerodynamics Consortium (of which the author is a member). This time was provided by EPSRC under contract GR/S91123/1. Thanks are also due to Andy Sunderland of the Terascaling Team at the National Supercomputing Centre for performing the scaling tests.

References

[1] Egolf, A., "*Helicopter Free Wake Predictions of Complex Wake Structures Under Blade-Vortex Interaction Operating Conditions*", 44th Annual Forum of the American Helicopter Society, June, 1988.

[2] Bagai, A. and Leishman, J.G., "*Rotor Free-Wake Modeling using a Pseudo-Implicit Technique - Including Comparison with Experiment*", Journal of the American Helicopter Society, Vol. 40, No. 3, 1995, pp29-41.

[3] Brown, R.E., Line, A.J. and Ahlin, G.A., "*Fuselage and Tail-Rotor Interference Effects on Helicopter Wake Development in Descending Flight*", Proceedings 60th American Helicopter Society Annual Forum, Baltimore, Maryland, June 2004.

[4] Line, A.J. and Brown, R.E., "*Efficient High-Resolution Wake Modelling using the Vorticity Transport Equation*", Proceedings 60th American Helicopter Society Annual Forum, Baltimore, Maryland, June 2004.

[5] Allen, C.B., "*Time-Stepping and Grid Effects on Convergence of Inviscid Rotor Flows*", AIAA paper 2002-2817, proceedings 20^{th} Applied Aerodynamics Conference, St Louis, June 2002.

[6] Allen, C.B. "*Convergence of Steady and Unsteady Inviscid Hovering Rotor Solutions*", International Journal for Numerical Methods in Fluids, Vol 41, No 9, 2003, pp931-949.

[7] Allen, C.B., "*The Effect of Grid Topology and Density on Inviscid Hovering Rotor Solutions*", I. Mech. E. Journal of Aerospace Engineering, Part G, Vol. 213, 1999, pp81-95.

[8] Wake, B.E. and Choi, D., "*Investigation of High-Order Upwinded Differencing for Vortex Convection*", AIAA Journal, Vol. 34, No. 2, pp332-337, 1996.

[9] Van-Leer, B., "*Flux-Vector Splitting for the Euler Equations*", Lecture Notes in Physics, Vol. 170, 1982, pp. 507-512.

[10] Parpia, I. H., "*Van-Leer Flux-Vector Splitting in Moving Coordinates*", AIAA Journal, Vol. 26, January 1988, pp. 113-115.

[11] Anderson, W. K. Thomas, J. L. and Van-Leer, B., "*Comparison of Finite Volume Flux Vector Splittings for the Euler Equations*", AIAA Journal, Vol. 24, September 1986, pp. 1453-1460.

[12] Allen, C.B., "*Multigrid Acceleration of an Upwind Euler Code for Hovering Rotor Flows*", The Aeronautical Journal, Vol. 105, No. 1051, September 2001, pp517-524.

[13] Allen, C.B., "*Multigrid Convergence of Inviscid Fixed- and Rotary-Wing Flows*", International Journal for Numerical Methods in Fluids, Vol. 39, No. 2, 2002, pp121-140.

[14] Obayashi, S., "*Freestream Capturing for Moving Coordinates in Three Dimensions*", AIAA Journal, Vol 30, No 4, 1992, pp1125-1128.

[15] Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings", AIAA Paper 91-1596.

[16] Allen, C. B., "*The Reduction of Numerical Entropy Generated by Unsteady Shockwaves*", Aeronautical Journal, Vol. 101, No. 1001, January 1997, pp9-16.

[17] Allen, C. B., "*Grid Adaptation for Unsteady Flow Computations*", I. Mech. E. Journal of Aerospace Engineering, Part G4, Vol. 211, 1997, pp237-250.

[18] 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.

[19] Gaitonde, A.L., "*A Dual-Time Method for the Solution of the unsteady Euler Equations* " The Aeronautical Journal of the Royal Aeronautical Society, Oct. 1994, pp 283-291

[20] Jameson, A., "*Transonic Flow Calculations*", Princeton University, Report MAE 1751, 1984.

[21] Jameson, A., "*Time-Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*", AIAA Paper 91-1596, 1991.

[22] Allen, C. B., "*CHIMERA Volume Grid Generation within the EROS Code*", I. Mech. E. Journal of Aerospace Engineering, Part G, 2000.

[23] Gordon, W. J. and Hall, C. A., "*Construction of Curvilinear Coordinate Systems and Applications of Mesh Generation*", International Journal of Numerical Methods in Engineering, Vol. 7, 1973, pp. 461-477.

[24] Eriksson, L. E., "*Generation of Boundary-Conforming Grids Around Wing-Body Configurations Using Transfinite Interpolation*", AIAA Journal, Vol. 20, No. 10, 1982, pp. 1313-1320.

[25] Thompson, J. F., "*A General Three Dimensional Elliptic Grid Generation System on a Composite Block-Structure*", Computer Methods in Applied Mechanics and Engineering, Vol. 64, 1987, pp. 377-411.

[26] Allen, C.B., "*Numerical Simulation of Lifting Forward Flight*", AIAA paper 2003-4080, 21st AIAA Applied Aerodynamics Conference, Florida, June 2003.

[27] Allen, C.B., "*An Unsteady Multiblock Multigrid Scheme for Lifting Forward Flight Simulation*", International Journal for Numerical Methods in Fluids, Vol. 45, No. 7, 2004.

[28] Allen, C.B., "*Numerical Simulation of Lifting Rotors in Hover, Ground Effect, and Forward Flight*", AIAA paper 2004-5288, 22nd AIAA Applied Aerodynamics Conference, Providence, RI, August 2004.

[29] Tang, L., Baeder, J.D., "*Improved Euler Simulation of Hovering Rotor Tip Vortices with Validation*", 55th American Helicopter Society Annual Forum, Montreal, Canada, May 1999.

[30] Murayama, M., Nakahashi, K., Sawada, K., "*Numerical Simulation of Vortex Breakdown Using Adaptive Grid Refinement with Vortex-Center Identification*", AIAA Paper 2000-0806, 2000.

[31] Steinhoff, J., Wang, C., Underhill, D., Mersch, T., Wenren, Y., "*Computational Vorticity Confinement: A Non-Diffusive Eulerian Method for Vortex-Dominated Flows*", UTSI Preprint, 1992.

[32] Steinhoff, J., Underhill, D., "*Modification of the Euler Equations for "Vorticity Confinement": Application to the Computation of Interacting Vortex Rings*", Physics of Fluids, Vol, 31, pp2738-2744, 1994.

[33] Steinhoff, J., "*"Vorticity Confinement": A New Techique for Computing Vortex Dominated Flows*", in *Frontiers of Computational Fluid Dynamics*, eds. D.A. Caughey and M.M. Hafez, John Wiley and Son, pp235-263, 1994.

[34] Wang, C.M., Steinhoff, J., Wenren, Y., "*Numerical Vorticity Confinement for Vortex-Solid Body Interaction Problems*", AIAA Journal, Vol. 33, pp1447-1453, 1995.

[35] Wenren, Y., Fan, M., Dietz, W., Hu, G., Braun, C., Steinhoff, J., Grossman, B., "*Efficient Eulerian Computation of Realistic Rotorcraft Flows using Vorticity Confinement. A Survey of Recent Results*", AIAA Paper 2001-2642, 2001.

[36] Biava, M., Vigevano, L., "*Assessment of the Vorticity Confinement Technique Applied to Rotorcraft Flows*", AIAA Paper 2003-3524, Orlando, 2003.

[37] Slotnick, J.P., Kandula, M., Buning, P.G., Martin, F.W., "*Numerical Simulation of the Space Shuttle Launch Vehicle Flowfield with Real Gas Solid Rocket Motor Plume Effects*", AIAA93-0521, 31st Aerospace Sciences Meeting, Reno, Nv, 1993.

[38] Kandula, M. and Buning, P.G., "*Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in OVERFLOW Thin-Layer Navier-Stokes Code*", AIAA94-2357, 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO, June 1994.

[39] Ahmad, J. and Duque, E.P.N., "*Helicopter Rotor Blade Computation in Unsteady Flows Using Moving Overset Grids*", Journal of Aircraft, Vol. 33, No. 1, pp54-60, 1996.

[40] Egolf, T.A., Wake, B.E., Berezin, C., "*Recent Rotor Wake Simulation and Modelling Studies at United Technologies Corporation*", AIAA2000-0115, 38th Aerospace Sciences Meeting, Reno, Nv, 2000.

Department of Aerospace Engineering, University of Bristol, BS8 1TR
E-mail: `c.b.allen@bristol.ac.uk`
URL: `http://www.aer.bris.ac.uk/contact/academic/allen1.shtml`