Application of a parallel rotor CFD code on HPCx

C. B. Allen
Department of Aerospace Engineering,
University of Bristol
Bristol, UK

A. G. Sunderland and R. Johnstone
CCLRC Daresbury Laboratory
UK

ABSTRACT

Aspects of parallel simulation of rotor flows are considered. These flows can be extremely expensive for a compressible finite-volume CFD code, and parallelisation can be essential. The award of HPCx time through the UK Applied Aerodynamics Consortium has allowed large rotor simulations to be performed and wake grid dependence to be investigated. However, there are several issues that need to be investigated when considering very large simulations, including the grid generation process, the parallel flow-solver, including an effective mesh motion approach, and visualisation options. Details of these are presented here, with particular emphasis on the flow-solver parallel performance. A detailed performance analysis of the unsteady flow-solver has been undertaken and the code optimised to improve parallel performance, and details of the parallel scaling performance are presented. The parallel scaling of the code is very good on all the HPC architectures tested here, and this has been recognised by an HPCx Gold Star Capability Incentive award. Results of simulation of a four-bladed lifting rotor in forward flight are also presented, for two mesh densities. It is shown that the solution computed on the serial limit on mesh size, around four million cells, exhibits excessive diffusion, and is of limited use in terms of detailed flow features. The results on a very fine mesh, 32 million cells, have shown a much better solution resolution, and it is also demonstrated that the λ_2 vortex core visualisation option is extremely useful.

NOMENCLATURE

a_∞	freestream acoustic speed
$\mathbf{a}$	acceleration tensor
C_n	normal force coefficient
E	energy
$\mathbf{F}$	flux vector
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	unit vectors in x, y, z directions
$\mathbf{q}$	absolute velocity vector = $(u, v, w)^T$
M_{FF}	rotor forward flight Mach number
M_{Tip}	rotor tip Mach number
$\mathbf{n}$	outward unit normal vector
N	number of blades
P	pressure
$[R]$	rotation matrix
S	cell surface area
$\mathbf{S}$	strain rate tensor
t	time
u, v, w	Cartesian velocity components
x, y, z	Cartesian co-ordinates
$\mathbf{X}$	co-ordinate vector
$\mathbf{U}$	conserved vector
V	cell volume
∂V	cell boundary
γ	ratio of specific heats
λ_2	second eigenvalue of velocity gradient tensor
μ	advance ratio (= M_{FF}/M_{Tip})
ν	viscosity
$\boldsymbol{\omega}$	rotation vector
Ω	vorticity tensor
Ψ	azimuth angle
ρ	density
θ_c	blade cyclic pitch cosine variation angle
θ_s	blade cyclic pitch sine variation angle
Θ	blade pitch angle
Θ_{inc}	blade shaft inclination angle

Paper No. 3095. Manuscript received 25 May 2006, revised version received 14 August 2006, accepted 10 October 2006.

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press

1.0 INTRODUCTION

One of the research themes being pursued within the UK Applied Aerodynamics Consortium is rotor simulation. These flows are computationally expensive for a compressible finite-volume code, as capture of the vortical wake is required over large distances, resulting in large grid sizes and long integration times. Hence, simpler models are often used in initial design, for example free wake or free vortex methods[1,2], or alternative formulations used for wake simulation, for example the vorticity-transport approach[3,4]. However, compressible simulation of these flows is important in the long term, to be able to represent all the flow regimes associated with a rotor blade, and finite-volume schemes are fairly straightforward to parallelise, and so the award of HPCx time to the consortium has allowed some large rotor simulations to be performed, and wake grid dependence to be investigated.

Due to the cost of these simulations, most cases are currently run as isolated rotors to obtain initial load variations. A medium term goal in this area is simulation of the complete vehicle, i.e. main rotor plus fuselage and tail rotor, as the interference effects of each component on the others are vital to produce an optimised design. Furthermore, important issues such as aeroelastic deformations, and subsequent vibration effects, blade-vortex interactions, and acoustic studies, add extra complexity and, hence, requirements for even larger meshes and smaller time-steps. Simulation of these phenomena will require meshes with hundreds of millions of cells, and so the development of each of the components required for these simulations is vital.

There are several possible alternative approaches to attempt to reduce the grid density while maintaining the solution resolution. Grid adaptation is an obvious option, see for example[5,6], but this is expensive, and implementation within an unsteady solver difficult, particularly with the large motion of the tip vortices across the mesh experienced in forward flight simulation. 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[7-10], is another option. This has already been applied to rotor flows, see for example Refs 11 and 12, but as yet there is no theoretical foundation for determining the confinement parameter value. It is also unlikely to help with general diffusion issues, for example it will not help resolution of pressure waves associated with acoustic issues. Chimera overlapping meshes can also be used, where high density meshes can be added to capture the tip vortices, but to specifically capture tip vortices in forward flight requires large motion of the tip vortex meshes which is very expensive and awkward. This type of approach has been adopted in the UTC OVERFLOW code[13,14], and applied to rotor flows[15,16]. However, Ref. 16 discusses 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 spatial scheme. Ref. 17 considers model problems related to rotor tip vortices, and shows improved vortex preservation. However, most of the published test cases are performed on cartesian-type meshes, and these conclusions are questionable for general skewed or stretched meshes. This approach remains the most interesting option, and is likely to be considered in the future.

Initially, some of the major components required for isolated rotor simulation (or any unsteady simulation with moving surfaces and large mesh size) are; a high quality grid generator, an efficient parallel flow-solver, including an effective mesh motion scheme, and visualisation tools. This paper is concerned primarily with the parallel performance of the flow-solver on HPCx, and details of this performance, including optimisation work, are presented. Brief details of the grid generator, mesh motion scheme, and visualisation tools are also presented for completeness.

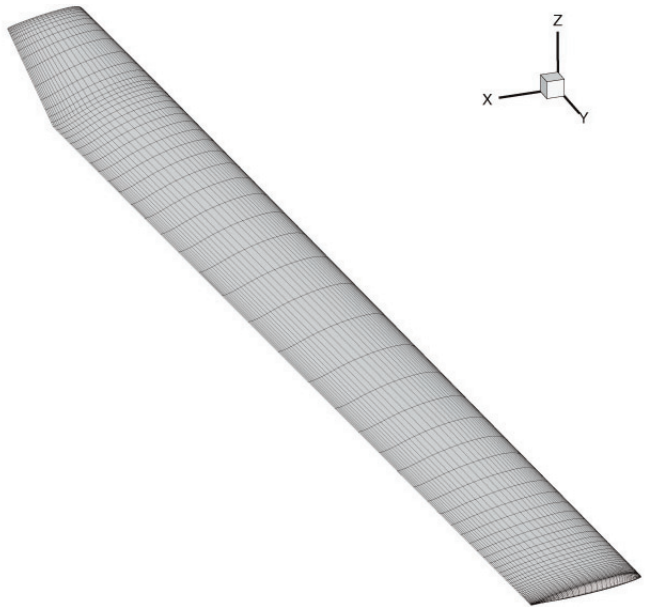


Figure 1. 7A rotor surface mesh.

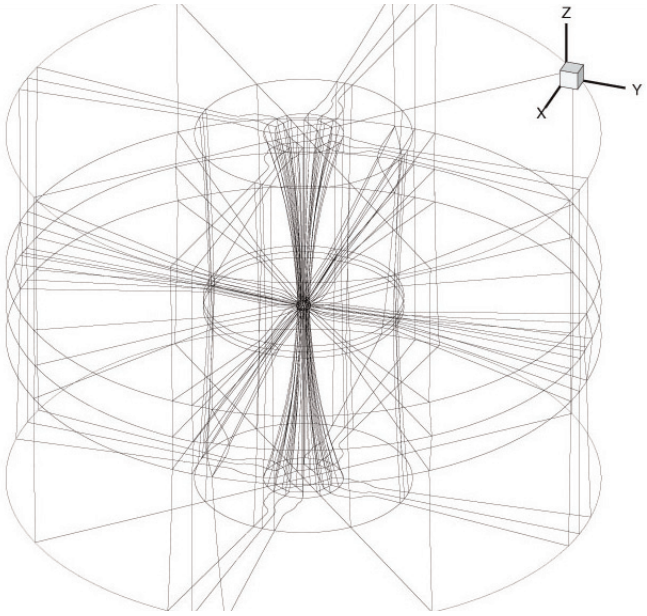


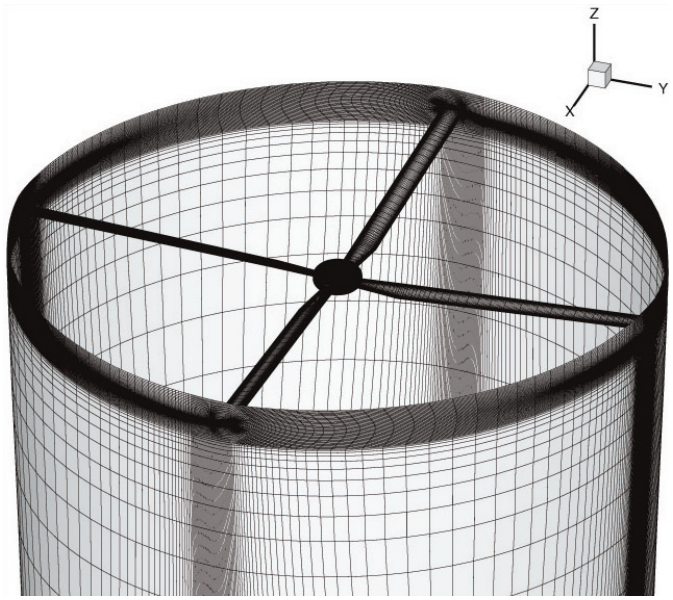
Figure 2. 7A rotor forward flight mesh domain and block boundaries.

2.0 MESH GENERATION SOFTWARE

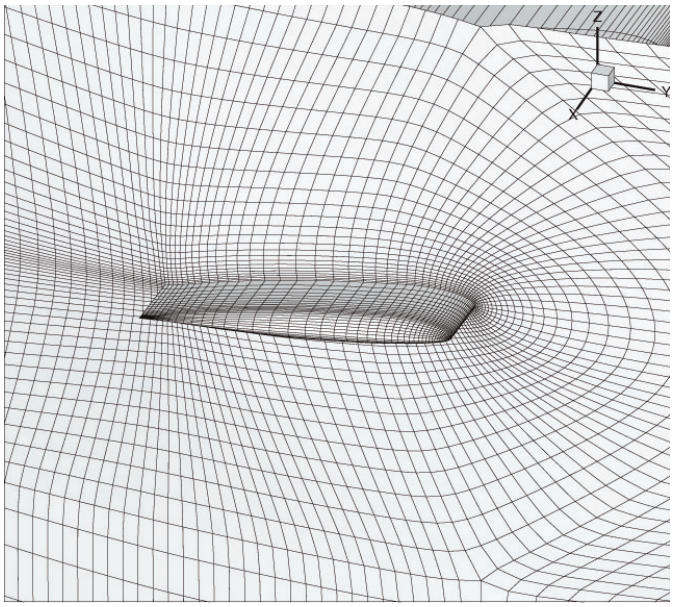
Structured meshes are used in the current research, and forward flight simulation cannot use a monoblock mesh due to the topology requirements. It was also shown in Refs 18 and 19 that multiblock hovering solutions exhibit better convergence and wake capturing than those on single block grids, due to improved grid quality possible.

For clean wings and rotors, standard block structures can be defined to remove some of the user input from the process, as this can be a bottle-neck in the overall simulation process. Multiblock meshes for rotors are generated by first generating a structured surface mesh from input patches. A small extent blade-fixed mesh is then generated around this solid surface. From this near blade mesh, cylindrical meshes are generated upstream, downstream, above and

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press



(a)



(b)

Figure 3. Four-bladed multiblock forward flight mesh. Selected mesh planes.

below the blade to fill the required domain. For hover, the hub region, i.e. inboard of the blade root, is not important and a cylindrical surface can be used, as there is no flow through this region. For forward flight there is of course flow through this region, and so the solid surface associated with the hub must be represented, with a fluid mesh above and below to allow flow through this region. Hence, mesh blocks above and below the hub are generated, and a transformation applied to map the solid surface to an elliptical shape. Periodicity is exploited to complete the mesh for the other blades.

A transfinite interpolation algorithm[20,21] is used to compute the grid within each block. After generating the entire mesh, elliptic smoothing is applied[22]. The smoothing has been coded such that boundaries are also smoothed (for more details of the mesh generation see Refs 23 and 24).

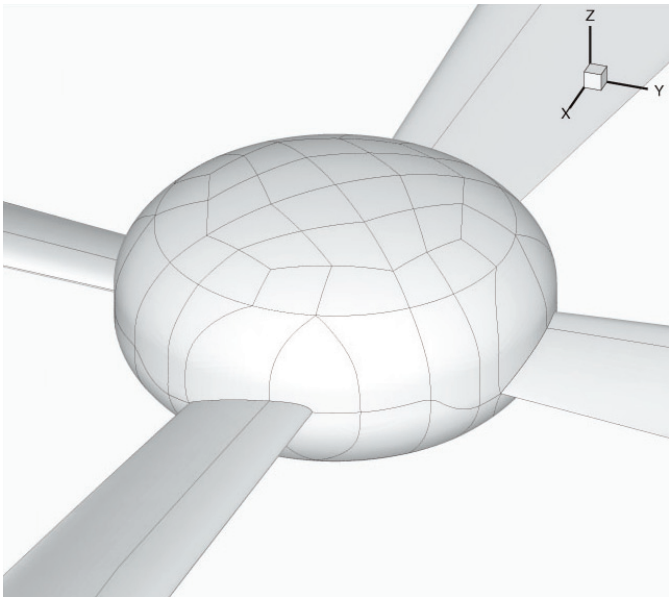


Figure 4. Four-bladed multiblock forward flight mesh. Hub block structure.

Figure 1 shows the surface mesh for the ONERA 7A[25] blade considered later. Figure 2 shows the domain and block boundaries, for a four-bladed 7A mesh (this is simplest topology with 208 blocks). The domain is a cylinder of height five tip radii, diameter six tip radii. Figure 3 shows two views of the solid surface (blades plus hub) mesh and selected planes near the blade tip (the lower image is an enlarged version of the upper image, to show the grid structure), and Fig. 4 shows the hub shape and block structure. The mesh shown contains around four million points in total.

The entire generation process is driven automatically from the input surface patches, and a user-input target number of grid points. All topologies and block structures are transparent to the user. The software can generate a 64 million point, 500 block, four-bladed forward flight mesh in around 15 minutes on a Pentium IV running Linux, and requires less than 2GBytes of RAM.

3.0 FLOW SOLVER

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

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{U}\mathrm{d}V + \int_{\partial V} \mathbf{F}.\mathbf{n}\mathrm{d}S = 0 \qquad \ldots (1)$$

Where

$$\mathbf{U}=\begin{bmatrix} r \\ ru \\ rv \\ rw \\ E \end{bmatrix}, \mathbf{F}=\begin{bmatrix} r\left[\mathbf{q}-\left(w\times\mathbf{X}(t)+\mathbf{X}_t(t)\right)\right] \\ ru\left[\mathbf{q}-\left(w\times\mathbf{X}(t)+\mathbf{X}_t(t)\right)\right]+P\mathbf{i} \\ rv\left[\mathbf{q}-\left(w\times\mathbf{X}(t)+\mathbf{X}_t(t)\right)\right]+P\mathbf{j} \\ rw\left[\mathbf{q}-\left(w\times\mathbf{X}(t)+\mathbf{X}_t(t)\right)\right]+P\mathbf{k} \\ E\left[\mathbf{q}-\left(w\times\mathbf{X}(t)+\mathbf{X}_t(t)\right)\right]+P\mathbf{q} \end{bmatrix}. \qquad \ldots (2)$$

The co-ordinate vector $\mathbf{X}(t)$ is time dependent in the fixed axis system, i.e.

$$\mathbf{X}(t)=\left[R(\psi)\right]\mathbf{X}(0) \qquad \ldots (3)$$

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press

where $[R(\Psi)]$ is the time-dependent rotation matrix. The $\boldsymbol{\omega} \times \mathbf{X}(t)$ term is due to the rigid rotation, but there is also a grid velocity associated with grid deformation, and this is the $\mathbf{X}_t(t)$ term in the above.

The equation set is closed by

$$P = (\gamma - 1)\left[E - \frac{\rho}{2}\mathbf{q}^2 \right] \quad \ldots (4)$$

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[26,27] is used.

To avoid introducing errors due to the evaluation of the rotational-terms, the local $(\boldsymbol{\omega} \times \mathbf{X})$ terms are evaluated by solving for cell face area moment vectors, and satisfying the resulting conservation condition[28-30].

An implicit form of the differential equation for each computational cell is considered, and the temporal derivative approximated by a second-order backward difference, following Jameson[31]. An explicit multi-stage Runge-Kutta scheme is then used to integrate the equations forward in pseudo-time, within each real time-step. Explicit time-stepping schemes lend themselves well to multigrid acceleration, see for example Refs 32 and 33, and multigrid is adopted here[24].

3.0.1 Mesh deformation

In forward flight the blades must undergo cyclic variation of pitch, flap and lead-lag angles around the azimuth to balance the loads and moments on the rotor disk. For example, on the advancing side, when the local onflow speed is the rotation speed plus a forward flight component, the pitch must be a minimum, while it must be a maximum on the retreating side, where the local onflow speed is the rotation speed minus a forward flight component. Hence, the pitch angle is varied using cyclic angles, the magnitude of which depend on the advance ratio (the ratio of forward flight speed to rotational speed). To simulate this the mesh needs to deform, and since large meshes are used here, the motion scheme must be efficient.

The mesh is deformed each time-step using a global parameterisation approach, which removes the connectivity requirement of each grid point, i.e. each point can moved independently to its neighbours. This means the scheme can be applied to any mesh type, but also means it is ideally parallel since the grid points on each CPU can be moved independently, meaning no communication is required between processors. Global weighting parameters, in terms of volume position and connectivity with each 'moving' surface, are evaluated for each point in the mesh. These do not change through the simulation, so can be computed once at the beginning of the simulation. The position of each mesh point at each time-level is then determined solely by the weighted functions of the instantaneous surface positions. A mesh density of 32 million cells is used in the example later, and for meshes of this size it is clearly vital to have a cheap mesh motion scheme. The current approach scales linearly with CPUs, and costs less than 1% of a real time-step for the 32 million cell case shown.

The time-dependent cell volumes are computed each time step by satisfying the required geometric conservation law[34]. 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:

$$\mathbf{X}_t(t) = \frac{1}{2\Delta t}\left[R(\psi) \right]\left\{ 3\mathbf{X}^{n+1} - 4\mathbf{X}^{n} + \mathbf{X}^{n-1} \right\} \quad \ldots (5)$$

(More details of the mesh motion scheme can be found in Ref. 35.)

4.0 FLOW-SOLVER PERFORMANCE

4.1 Parallelisation

The flow-solver has been parallelised using MPI[36], to allow the use of fine meshes. An algorithm linked to the mesh 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 mesh generation process, to optimise the mesh dimensions and load balancing.

The most significant aspect of this data structure is that the code has been written to use 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 significantly reduces the memory requirements of the code. 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 is then unpacked according to the orientation flag. This means no cell connectivity data is required in the grid file and, hence, there is no preprocessing or domain decomposition stage. 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 and load components from each CPU, and writes to the output file.

The global parameterisation quantities required to move the mesh at each time step only need to be computed once at the beginning of the calculation, independently to the flow solution. These quantities are written to file, one per block, and are not kept in memory within the flow-solver. At each real time step each block can be updated independently to the other blocks, and so each CPU simply reads in the required data for each block assigned to it in turn, moves the mesh, then deallocates that space. Hence, as mentioned earlier, this process is perfectly parallel.

Avoiding any global storage or collection of flow/geometry data means the code is extremely efficient in terms of memory; less than 0·5GByte RAM/million cells are required, even using four levels of multigrid.

4.2 HPCx hardware details

HPCx is the UK's flagship high performance computing service[37]. It is a large IBM p5-575 cluster whose configuration is specifically designed for high-availability capability computing. HPCx is a joint venture between the Daresbury Laboratory of the Council for the Central Laboratories of the Research Councils (CCLRC)[38] and Edinburgh Parallel Computing Centre (EPCC) at the University of Edinburgh[39]. The current phase of HPCx has 96 IBM p690+ Regatta nodes for computation with a total of 1·60TBytes of memory and the high performance switch (HPS) for message passing, also formally known as Federation. Each Regatta node houses 16 1·5GHz IBM p5-575 processors, making 1,536 processors in total. Results are also shown for the previous HPCx system: a 1600 processor p690+ cluster with HPS.

For comparison, the HPCx performance is shown against two other UK-based HPC systems. Firstly Newton[40], a 512 processor SGI Altix 3700 which has 384 1·3GHz Itanium2 processors and 128 1·5GHz processors with 1TByte of memory and the NUMAflex interconnect, based at the University of Manchester. Secondly SCARF (Scientific Computing Application Resource for Facilities)[41], a compute cluster run by the CCLRC e-Science Centre. It comprises of 128 dual CPU AMD Opteron 248MSI 1000-D processors with 2·2GHz clock speed. Message passing is achieved via a Myrinet M3F-PCIXD-2 interconnect.

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press

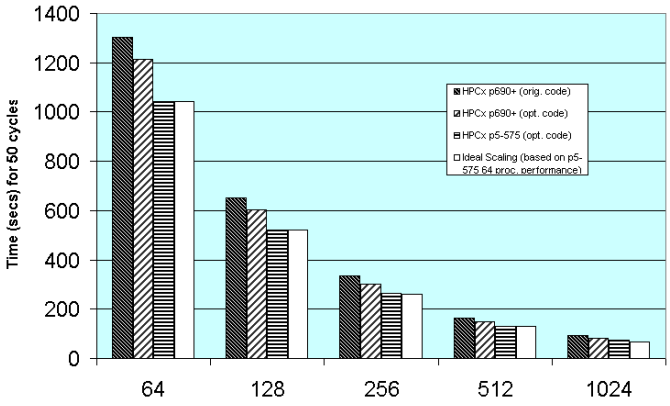


Figure 5. Parallel performance of flow-solver on HPCx.

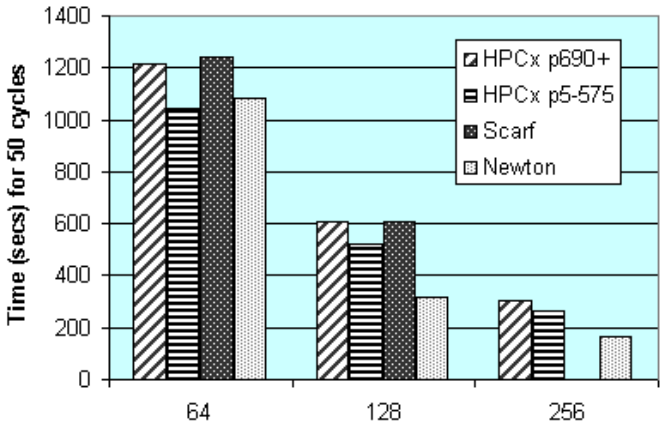


Figure 6. Parallel performance of flow-solver on several HPC platforms.

4.3 Parallel performance on HPCx and other HPC systems

The parallel performance of the original flow-solver on HPCx was generally very good. However, profiling the parallel code on HPCx with the Vampir analysis tool[42] identified inefficiencies associated with the data exchange across block boundaries for both planes of solution during cycles of the iterative solver. The structure of this message passing operation was split into two distinct stages separated by global calls to MPI_Barrier. Removing all global synchronisations introduces more flexibility into the communication pattern and processors need only wait for their neighbours to send data before proceeding with the next cycle of the solver. This asynchronicity allows the system to schedule tasks more efficiently and therefore reduces communication overheads significantly, particularly on the HPCx system. A fuller description of the analysis and optimisation can be found in Ref. 43. The superior performance of the optimised code on the new p5-575 nodes is chiefly due to improvements in the memory heirarchy of the new nodes[44].

The benchmark runs were performed on a four-bladed rotor case in forward flight. Simulations were run using a mesh density of 20M points. The original blocking resulted in 208 blocks, and these were split into smaller blocks for the simulations on larger numbers of CPUs, and the resulting mesh contained 145 × 145 points in the surface mesh on each blade.

The scaling performance on HPCx for both the original code and the optimised version of the code are shown in Fig. 5. From the p690+ figures, it is shown that the performance gain with the optimised version is more marked at large processor counts resulting in a reduction in compute time of 7·5% on 64 processors and 11·5% on 1,024 processors. The speed-up for the optimised code from 64 processors to 1,024 processors is a factor of 14 98 (linear scaling corresponds to a factor of 16). The performance on each processor averages 475Mflop/s, which is around average for CFD codes on HPCx. The excellent scaling properties of the code has been recognised by a Gold star rating under the HPCx Capability Incentive Award Scheme[45]. This entitles users of the code to specially discounted resources on HPCx.

The performance of the code on the four computing platforms decribed earlier is shown in Fig. 6. The code scales very well on all the systems, but parallel performance is best on the Altix 3000 system (Newton). This is largely due to the excellent memory bandwidth of the Altix combined with good message passing performance from the NUMAflex architecture. At the time of benchmarking, runs on SCARF were restricted to a maximum of 128 processors but it is evident that, at least upto this level of parallelism, competitive performance and good scaling can be obtained from the pairing of AMD Opteron processors with commodity switch.

5.0 7A ROTOR IN LIFTING FORWARD FLIGHT

A lifting forward flight test case was run with the 7A rotor. (The flow solver has previously been validated for forward flight, see for example Ref. 24.) 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, Θ_{inc}, of –3·72 degrees, i.e. backward, so has significant BVI effects. The collective pitch setting is 2·74°, and the cyclic pitch variation,

$$\Theta(\psi) = \theta_c \mathrm{Cos}(\psi) + \theta_s \mathrm{Sin}(\psi) \quad \ldots (6)$$

is determined by:

$$\theta_c = 2{\cdot}66^\circ \qquad \theta_s = -2{\cdot}24^\circ \quad \ldots (7)$$

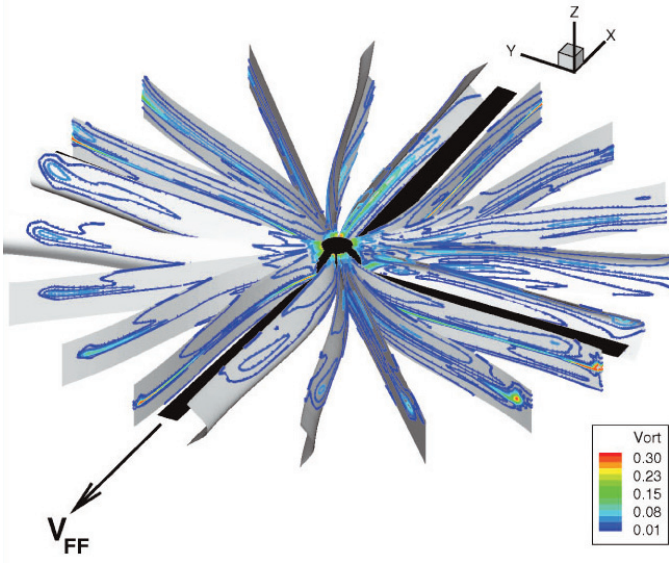
The case was run as an unsteady simulation, using 360 real time-steps per revolution, by spinning the entire rotor mesh at Ω_z with a uniform freestream onflow velocity of ($\mu M_{Tip} a_\infty \mathrm{Cos}\Theta_{inc}$, 0, $-\mu M_{Tip} a_\infty \mathrm{Sin}\Theta_{inc}$), and the third revolution is taken as converged.

Two mesh densities were considered; four million and 32 million cells. Four million cells is the limit of serial capability for a 32bit machine, as this just fits within 2GBytes RAM. Hence, this was run as the serial benchmark, and a parallel simulation run with double the mesh density in each parametric direction to give 32 million cells.

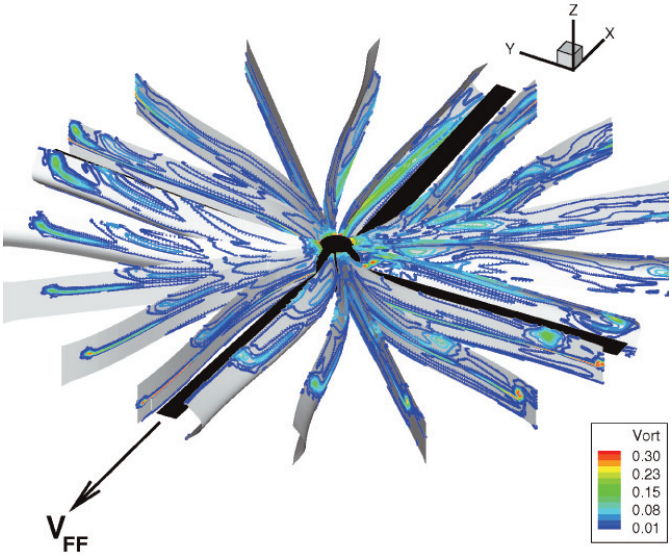
Figure 7 shows contours of the vorticity field on selected cutaway planes in the rotor disc (V_{FF} shows the forward flight direction of the rotor), for the 4 and 32 million cell grid densities. This shows the tip vortex paths around the azimuth. The grid dependence is also clear, with more details emerging, due to preservation of the vortical wake over longer distances, on the finer mesh. It is clear on the four million cell mesh that the tip vortex from each blade is barely reaching the following blade before being completely diffused.

To investigate this further, and to validate the code, the sectional blade load coefficient was evaluated around the azimuth, at two radial stations. The results for the two meshes are compared with experimental data in Fig. 8. Hence, the general trends are captured well, for both meshes, but more details are beginning to emerge on the finer mesh. The peaks and troughs in the experimental data are due to blade-vortex interactions and, hence, it is clear that these are not being captured at all on the four million cell mesh. The 32 million cell simulation has captured the positions of the interactions, but not the correct strengths.

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press

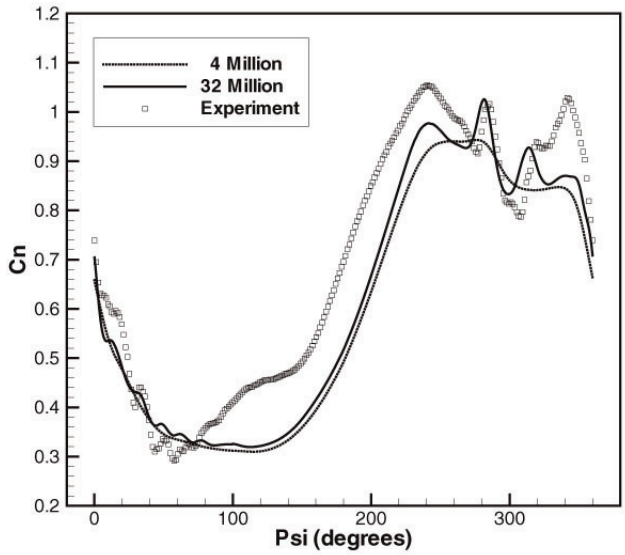


(a)

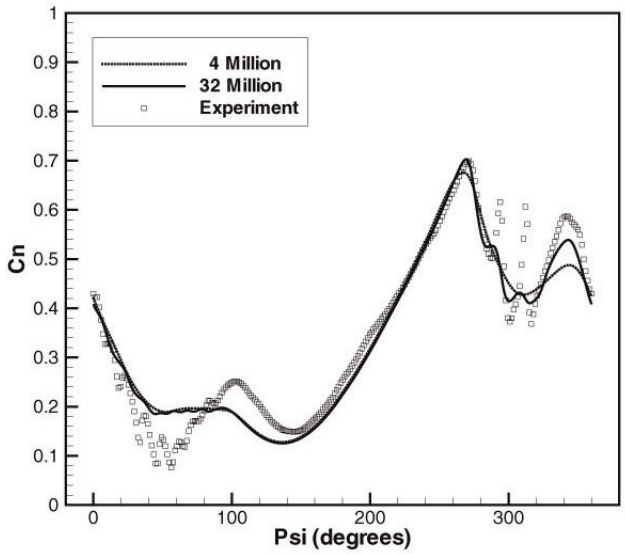


(b)

Figure 7. 7A forward flight vorticity contours for 4 and 32m cells.



(a)



(b)

Figure 8. 7A forward flight sectional blade loading, r/R = 0·7, 0·92.

6.0 VORTEX TRAJECTORY EXTRACTION

It is common to visualise vorticity magnitude on mesh planes, as presented above. This quantity is easy to calculate and Galilean invariant. However, when considering flowfield data, for example visualising isosurfaces, it is necessary to determine the correct threshold for vorticity magnitude isosurfaces. The ideal threshold is usually a function of the local radial extent of the vortex and thus its age (and also, in general, different for different vortices), and vorticity generated by boundary layers or solid surface boundary conditions often tends to obscure the interaction of vortices with solid surfaces (which is often of interest). Hence, it is not normally possible to obtain detailed vortex trajectories in this manner.

As a pressure minimum is expected in a vortex core, pressure isosurfaces can also be used (or better, techniques that detect pressure minima[46]), but unsteady effects can create non-vortical pressure minima, and viscous effects can mask those that are in fact vortices. The λ_2 method due to Jeong and Hussain[47] is designed to remove these effects.

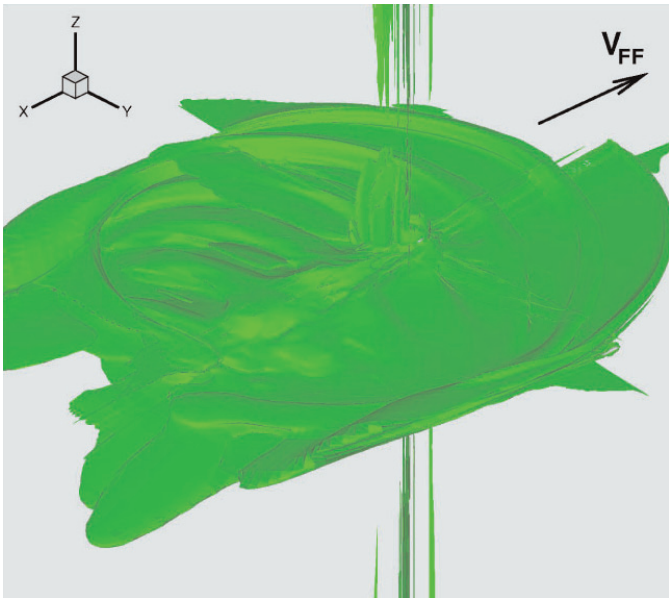
This method is developed as follows: by taking the gradient of the incompressible Navier-Stokes equations, the Hessian ($p_{,ij}$) of the pressure may be expressed as

$$-\frac{1}{\rho}p_{,ij} = a_{i,j} + \upsilon u_{i,jkk} \qquad \ldots (8)$$

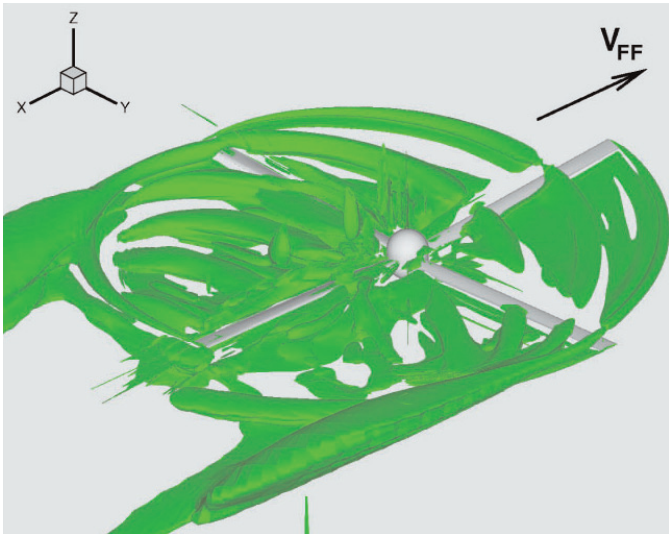
where the comma denotes partial differentiation, and indices imply summation; a is the acceleration tensor, which is decomposed into symmetric and antisymmetric parts:

$$a_{i,j} = \underbrace{\left[\frac{\mathrm{D}S_{ij}}{\mathrm{D}t} + \Omega_{ik}\Omega_{kj} + S_{ik}S_{kj}\right]}_{symmetric} + \underbrace{\left[\frac{\mathrm{D}\Omega_{ij}}{\mathrm{D}t} + \Omega_{ik}S_{kj} + S_{ik}\Omega_{kj}\right]}_{antisymmetric} \qquad \ldots (9)$$

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press



(a)



(b)

Figure 9. 7A forward flight vorticity and λ_2 isosurface, 32m cells.

Here **S** is the strain rate tensor $\frac{1}{2}(u_{i,j} + u_{j,i})$, and Ω is the vorticity tensor $\frac{1}{2}(u_{i,j} - u_{j,i})$; these are the symmetric and antisymmetric components of $\nabla \mathbf{q}$ respectively.

The antisymmetric part of Equation (9) describes inviscid vorticity transport. The intention is to relate $p_{,i,j}$ to vortical motion only, neglecting viscous and unsteady effects. Hence the second term on the RHS of Equation (8) is ignored, and the inviscid vorticity transport equation is assumed to be satisfied, and therefore also removed, as are unsteady terms.

Equation (8) becomes

$$-\frac{1}{\rho} p_{,ij} = \Omega_{ik}\Omega_{kj} + S_{ik}S_{kj} = \Omega^2 + \mathbf{S}^2 \qquad \ldots (10)$$

By the second derivative test for the presence of local extrema, there will be a local maximum of $-p$ and hence minimum of p in a plane if the Hessian $p_{,i,j}$ has two negative eigenvalues. Hence the sort of pressure minimum associated with a vortex core (we require a minimum in a plane), would be expected whenever the second largest eigenvalue λ_2 of $\Omega^2 + \mathbf{S}^2$ is less than zero (in practice, an arbitrary small negative value improves results[48], by reducing spurious false positives due to imperfect numerical differentiation).

Figure 9 shows an isosurface of vorticity magnitude for the domain of the 32 million cell mesh. This has a value of non-dimensional vorticity of 0·1 (the scale in Fig. 7 is 0·01-0·3). There are some spikes in vorticity apparent above and below the hub, and these are due to a combination of slightly non-smooth grid distributions across block boundaries here and the post-processing algorithm, and so are non-physical. The blades and hub are shown but are hidden by the surface. This gives an idea of the general structure of the flow, but many of the details are obscured. This is where the λ_2 quantity is useful. Figure 9 also shows an isosurface of λ_2 (at a small negative number to remove effects of imperfect numerical integration). A value of –0·001 is used here. The vortex core paths are now much clearer, as vorticity not associated with a local pressure trough is not shown. This type of visualisation is likely to be essential for future extensions, for example considering the vortex interactions with fuselage and tail rotor.

It should be noted that three computed revolutions may be insufficient for the wake and super-vortices downstream of the rotor disk to fully develop. However, the mesh outboard of the disk stretches away exponentially, and so very little is captured here, anyway. Hence, the blade flow features, for example the sectional force coefficient above, are unlikely to be affected by flow in this region.

7.0 CONCLUSIONS

Aspects of parallel simulation of rotor flows have been considered and presented. These flows can be extremely expensive for a compressible finite-volume CFD code, and parallelisation is essential. However, there are several issues that need to be investigated when considering very large simulations, including the parallel flow-solver, the grid generation process, mesh motion approach, and visualisation options. These have been presented here, with particular emphasis on the flow-solver parallel performance and optimisation.

Details of the implicit unsteady multiblock multigrid flow solver have been presented, along with parallelisation issues. Examples of the capability of the structured multiblock mesh generation algorithm for rotors have also been presented.

A detailed parallel performance analysis of the code has been undertaken and the code has been optimised to improve parallel performance. The parallel scaling of the code is very good on all the HPC architectures tested here, and this has been recognised by an HPCx Gold Star Capability Incentive award. A highly scalable flow-solver is clearly essential for future applications, including full vehicle simulation.

Results of simulation of a four-bladed lifting rotor in forward flight have been presented, and grid dependence considered. It has been shown how the solution computed on the serial limit on mesh size, around four million cells, exhibits excessive diffusion and, although still very useful for global values of integrated loads, is of limited use in terms of detailed flow features. The results on a very fine mesh, 32 million cells, have shown a much better solution resolution, and it has been demonstrated that the λ_2 vortex core visualisation option is extremely useful.

Considering these results it is likely that at least 100 million cells would be required to use the current code for full helicopter simulation in forward flight. However, this is likely to require around 50 hours on 1,024 CPUs (at current HPCx performance) for one revolution of the disk, and two or three would be required for periodicity, so 150,000 CPU hours. Hence, an order of magnitude reduction is needed in terms of mesh cells required, or an order of magnitude increase in terms of code/CPU/compiler performance, to obtain an 'overnight' turn-around time for these type of calculations.

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press

ACKNOWLEDGEMENTS

The authors would like to thank postgraduate student James Rodgers for his contribution to the grid generation codes and Agusta-Westland, who funded part of the mesh generation research, for their continued support. Also EPSRC (the Engineering and Physical Sciences Research Council) for funding time (through grant GR/S91130/01) on the HPCx machine for the UK Applied Aerodynamics Consortium.

REFERENCES

1. BHAGWAT, M.J. and LEISHMAN, J.G. On the aerodynamic stability of helicopter rotor wakes, May 2000, Proceedings of the 56th Annual AHS Forum, Virginia Beach, VA.
2. CHUNG, K.H., NA, S.U., JEON, W.H. and LEE, D.J. A study of rotor tip vortex roll-up phenomenon by using time-marching free-wake method, May 2000, Proceedings of the 56th Annual AHS Forum, Virginia Beach, VA.
3. BROWN, R.E., LINE, A.J. and AHLIN, G.A. Fuselage and tail-rotor interference effects on helicopter wake development in descending flight, June 2004, Proceedings 60th American Helicopter Society Annual Forum, Baltimore, Maryland.
4. LINE, A.J. and BROWN, R.E. Efficient high-resolution wake modelling using the vorticity transport equation, June 2004, Proceedings 60th American Helicopter Society Annual Forum, Baltimore, Maryland.
5. TANG, L. and BAEDER, J.D. Improved Euler simulation of hovering rotor tip vortices with validation, May 1999, 55th American Helicopter Society Annual Forum, Montreal, Canada.
6. MURAYAMA, M., NAKAHASHI, K. and SAWADA, K. Numerical simulation of vortex breakdown using adaptive grid refinement with vortex-center identification, 2000, AIAA Paper 2000-0806.
7. STEINHOFF, J., WANG, C., UNDERHILL, D., MERSCH, T. and WENREN, Y. Computational vorticity confinement: a non-diffusive Eulerian method for vortex-dominated flows, 1992, UTSI Preprint.
8. STEINHOFF, J. and UNDERHILL, D. Modification of the Euler equations for 'vorticity confinement': application to the computation of interacting vortex rings, *Physics of Fluids*, 1994, **31**, pp 2738-2744.
9. STEINHOFF, J. Vorticity confinement: a new techique for computing vortex dominated flows, *Frontiers of Computational Fluid Dynamics*, 1994, pp 235-263, CAUGHEY, D.A. and HAFEZ, M.M. (Eds), John Wiley and Son.
10. WANG, C.M., STEINHOFF, J. and WENREN, Y. Numerical vorticity confinement for vortex-solid body interaction problems, *AIAA J*, 1995, **33**, pp 1447-1453.
11. WENREN, Y., FAN, M., DIETZ, W., HU, G., BRAUN, C., STEINHOFF, J. and GROSSMAN, B. Efficient Eulerian computation of realistic rotorcraft flows using vorticity confinement. a survey of recent results, 2001, AIAA Paper 2001-2642.
12. BIAVA, M. and VIGEVANO, L. Assessment of the vorticity confinement technique applied to rotorcraft flows, 2003, AIAA Paper 2003-3524, Orlando.
13. SLOTNICK, J.P., KANDULA, M., BUNING, P.G. and MARTIN, F.W. Numerical simulation of the Space Shuttle launch vehicle flowfield with real gas solid rocket motor plume effects, 1993, AIAA93-0521, 31st Aerospace Sciences Meeting, Reno, NV.
14. KANDULA, M. and BUNING, P.G. Implementation of LU-SGS algorithm and Roe upwinding scheme in OVERFLOW thin-layer Navier-Stokes code, June 1994, AIAA94-2357, 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO.
15. AHMAD, J. and DUQUE, E.P.N. Helicopter rotor blade computation in unsteady flows using moving overset grids, *J Aircr*, 1996, **33**, (1), pp 54-60.
16. EGOLF, T.A., WAKE, B.E. and BEREZIN, C. Recent rotor wake simulation and modelling studies at United Technologies Corporation, 2000, AIAA2000-0115, 38th Aerospace Sciences Meeting, Reno, NV.
17. WAKE, B.E. and CHOI, D. Investigation of high-order upwinded differencing for vortex convection, *AIAA J*, 1996, **34**, (2), pp 332-337.
18. ALLEN, C.B. Time-stepping and grid effects on convergence of inviscid rotor flows, June 2002, AIAA paper 2002-2817, proceedings 20th Applied Aerodynamics Conference, St Louis.
19. ALLEN, C.B. Convergence of steady and unsteady inviscid hovering rotor solutions, *Int J for Numerical Methods in Fluids*, 2003, **41**, (9), pp 931-949.
20. GORDON, W.J. and HALL, C.A. Construction of curvilinear coordinate systems and applications of mesh generation, *Int J of Numerical Methods in Engineering*, 1973, **7**, pp 461-477.
21. ERIKSSON, L.E. Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation, *AIAA J*, 1982, **20**, (10), pp 1313-1320.
22. THOMPSON, J.F. A General three dimensional elliptic grid generation system on a composite block-structure, *Computer Methods in Applied Mechanics and Engineering*, 1987, **64**, pp 377-411.
23. ALLEN, C.B. CHIMERA volume grid generation within the EROS code, *I Mech E. J Aerospace Engineering*, 2000, Part G.
24. ALLEN, C.B. An unsteady multiblock multigrid scheme for lifting forward flight dimulation, *Int J for Numerical Methods in Fluids*, 2004, **45**, (7), pp 973-984.
25. SCHULTZ, K.-J., SPLETTSTOESSER, W., JUNKER, B., WAGNER, W., SCHEOLL, E., ARNAULD, G., MERCKER, E. and FERTIS, D. A Parametric wind tunnel test on rotorcraft aerodynamics and aeroacoutics (HELISHAPE) — test documentation and representative results, 1996, 22nd European Rotorcraft Forum, Brighton, UK.
26. VAN-LEER, B. Flux-vector splitting for the Euler equations, *Lecture Notes in Physics*, 1982, **170**, pp 507-512.
27. PARPIA, I.H. Van-Leer flux-vector splitting in moving coordinates, *AIAA J*, January 1988, **26**, pp 113-115.
28. OBAYASHI, S. Freestream capturing for moving coordinates in three dimensions, *AIAA J*, 1992, **30**, (4), pp 1125-1128.
29. ALLEN, C.B. Multigrid acceleration of an upwind Euler code for hovering rotor flows, *Aeronaut J*, September 2001, **105**, (1051), pp 517-524.
30. ALLEN, C.B. Multigrid convergence of inviscid fixed- and rotary-wing flows, *Int J for Numerical Methods in Fluids*, 2002, **39**, (2), pp 121-140.
31. JAMESON, A. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, AIAA Paper 91-1596.
32. JAMESON, A. Transonic flow calculations, 1984, Princeton University, Report MAE 1751.
33 JAMESON, A. Time-dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, 1991, AIAA Paper 91-1596.
34. THOMAS, P.D. and LOMBARD, C.K. Geometric conservation laws and its application to flow computations on moving grid, *AIAA J*, 1979, **17**, (10), pp 1030-1037.
35. ALLEN, C.B. Parallel universal approach to mesh motion and application to rotors in forward flight, to appear *Int J for Numerical Methods in Engineering*.
36. SNIR, M., OTTO, S., HUSS-LEDERMAN, S., WALKER, D. and DONGARRA, J. *MPI: The Complete Reference*, 1996, The MIT Press, Cambridge, Massachusetts.
37. HPCx: The UK's world-class service for world-class research, http://www.hpcx.ac.uk.
38. Central Laboratory of the Research Councils, http://www.cclrc.ac.uk.
39. Edinburgh Parallel Computing Centre, http://epcc.ed.ac.uk.
40. Computer Services for Academic Research, http://www.csar.cfs.ac.uk.
41. Scientific Computing Application Resource for Facilities, http://hpcsg.esc.rl.ac.uk/scarf/groups/index.html.
42. Vampir & VampirTrace, http://www.pallas.com/e/products/index.htm.
43. SUNDERLAND, A.G., EMERSON, D.R. and ALLEN, C.B. Parallel performance of a UKAAC helicopter code on HPCx and other large-scale facilities, May 2005, Proceedings 17th International Parallel CFD Conference, Maryland.
44. GRAY, A. *et al* A performance comparison of HPCx Phase 2a to Phase 2, HPCx Technical Report HPCxTR0602, http://www.hpcx.ac.uk/research/hpc/.
45. Capability Incentives on HPCx, http://www.hpcx.ac.uk/services/policies/capability.html.
46. SINGER, B. and BANKS, D. A predictor-corrector scheme for vortex identification, 1994, ICASE Report No. 94-11, NASA Langley.
47. JEONG, J. and HUSSAIN, F. On the identification of a vortex, *J Fluid Mechanics*, 1995, **285**, pp 69-94.
48. LIN, C-L., MCWILLIAMS, J.C., MOENG, C-H. and SULLIVAN, P.P. Coherent structures and dynamics in a neutrally stratified planetary boundary layer flow, *Physics of Fluids*, 1996, **8**, (10), pp 2626-2639.

https://doi.org/10.1017/S0001924000004401 Published online by Cambridge University Press