Parallel Simulation of Lifting Rotor Wakes

C. B. Allen[a*],

[a]Department of Aerospace Engineering,
University of Bristol,
Bristol, BS8 1TR, United Kingdom.

**Extended Abstract submitted to PCFD05 Coference,
Maryland, May 2005.**

1. INTRODUCTION

It is well known that numerical diffusion inherent in all CFD codes severely compromises the resolution of high flow gradients. This is a serious problem for rotor flow simulation, where the vortical wake capture is essential. 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. This problem is considered here. 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.

In this paper the unsteady formulation of the flow-solver will be presented, along with parallelisation details, followed by grid generation aspects for structured multiblock grids for rotors in hover and forward flight. Numerical solutions for lifting rotors 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 question is: 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 ?

2. UNSTEADY FLOW SOLVER

An unsteady finite-volume upwind scheme is used to solve the integral form of the Euler equations. The code has been developed for structured multiblock meshes, and

*Contact details, Email: c.b.allen@bristol.ac.uk, Tel: 0044 117 9546971.

incorporates an implicit temporal derivative, following Jameson [4], with an explicit-type scheme within each real time-step. Multigrid acceleration is also adopted [1,2,15–17].

2.1. 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.

The solver is coded such that each block boundary simply has a boundary condition tag, a neighbouring block number, and an orientation flag. The only restriction applied is that a boundary can only have one type of tag. Hence, each block only requires NI, NJ, NK, the 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 simply packed and sent to the appropriate processor, and received boundary data unpacked according to the orientation flag. This has the advantage that no halo or multigrid data is required in the grid file and, hence, there is no 'preprocessing' stage. This is important 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. This also means that if the spatial stencil is changed, for example to a higher-order scheme, the number of levels of solution sent and received can simply be increased.

The code has been written to require no global flow or geometry data storage. Only arrays of block dimensions and connectivity are required globally, and this vastly reduces the memory requirements of the code. The 32 million point mesh cases 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.

More discussion of the parallel implementation will be included in the final paper.

3. GRID GENERATION

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 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.

More details in final paper

A wake grid dependence study has been performed, considering the ONERA 7A blade [7]. A typical mesh density for a simulation may be one million points, and so 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, for hover and forward-flight.

Figure 1 shows the computational domain and block boundaries, for the 7A four-bladed case, left is hover and right is forward-flight. The farfield for the forward-flighht mesh is set at 50 chords spanwise and 50 chords vertically, i.e. a cylinder of radius 50 chords, height 100 chords, and for hover the spanwise boundary is 100 chords, and lower is 150 chords. The grid density above and below the rotor disk in forward-flight 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, in the forward flight case, to maintain the maximum proportion of points in the rotor disk area, as this is the region of interest.

Figure 2 shows part of the grid in the rotor disk for each of the six grid densities, for forward fight.

4. 7A ROTOR LIFTING CASES

The hover test case is $M_{Tip} = 0.661$, with a thrust coefficient of around 0.08, and the solver was run in steady mode, in a blade-fixed rotating coordinate system. In fact oscillatory convergence history was demonstrated for 16 and 32 million points. Hence, the 32 million point case is currently being run as an unsteady case, and time history animations produced of vorticity and vorticity temporal derivative, to attempt to understand the physics of the flow.

The forward flight case corresponds to datapoint Dpt0165, where the M_{Tip} 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 BVI effects.

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

$$\mathbf{q}_{freestream} = (\mu M_{Tip} a_\infty \cos\Theta_{inc}, 0, -\mu M_{Tip} a_\infty \sin\Theta_{inc})^T. \quad (1)$$

where Θ_{inc} is the rotor shaft inclination.

Figure 3 shows the vorticity field on selected cutaway 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 shows both the tip vortex path and the effect of numerical diffusion. The grid dependence is clear, with even the 32 million point simulation exhibiting significant diffusion. This is as expected.

The final paper will present detailed evaluataion of grid dependence of more detailed flow features.

5. COMPUTATIONAL REQUIREMENTS

As mentioned earlier, the grid generation software is very cheap, requiring around 20 minutes and 1GByte RAM to generate the 32 million point mesh.

The hover cases and smaller forward flight 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 and 32 million point forward flight cases were 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.

Figure 4 shows the parallel performance of the code[2]. The case was not run on under 32 CPU's due to lack of memory, but a speed-up factor of one was set for this case, and subsequent cases scaled from this.

6. CONCLUSIONS

Numerical simulation of multi-bladed lifting rotor flows in hover and forward flight has been presented. An implicit, upwind, unsteady, multiblock, multigrid scheme has been developed and validated, 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 performance has been demonstrated.

It has been demonstrated that the numerical dissipation inherent in the solver seriously limits the accuracy of wake capturing. Even the solutions computed using 32 million points exhibits significant wake diffusion resulting in the vorticity diffusing quickly behind each blade.

More detailed studies of local flow features have been performed, for example blade-vortex interactions, and will be presented in final paper.

Acknowledgements

The HPCx computer time was provided through the UK Applied Aerodynamics Consortium (of which the author is a member) under EPSRC grant EP/S91130/01.

REFERENCES

1. 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.
2. 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.
3. Obayashi, S., "*Freestream Capturing for Moving Coordinates in Three Dimensions*", AIAA Journal, Vol 30, No 4, 1992, pp1125-1128.
4. Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings", AIAA Paper 91-1596.
5. Allen, C. B., "*The Reduction of Numerical Entropy Generated by Unsteady Shock-waves*", Aeronautical Journal, Vol. 101, No. 1001, January 1997, pp9-16.
6. Allen, C. B., "*Grid Adaptation for Unsteady Flow Computations*", I. Mech. E. Journal of Aerospace Engineering, Part G4, Vol. 211, 1997, pp237-250.
7. 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*

[2]These timings have been certified by the Terascaling Team at the National Supercomputing Centre

and Aeroacoutics (HELISHAPE) - Test Documentation and Representative Results", 22nd European Rotorcraft Forum, Brighton, U.K., 1996.
8. 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
9. Jameson, A., "*Transonic Flow Calculations*", Princeton University, Report MAE 1751, 1984.
10. Jameson, A., "*Time-Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*", AIAA Paper 91-1596, 1991.
11. Allen, C. B., "*CHIMERA Volume Grid Generation within the EROS Code*", I. Mech. E. Journal of Aerospace Engineering, Part G, 2000.
12. 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.
13. 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.
14. 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.
15. Allen, C.B., "*Numerical Simulation of Lifting Forward Flight*", AIAA paper 2003-4080, 21st AIAA Applied Aerodynamics Conference, Florida, June 2003.
16. 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.
17. 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.

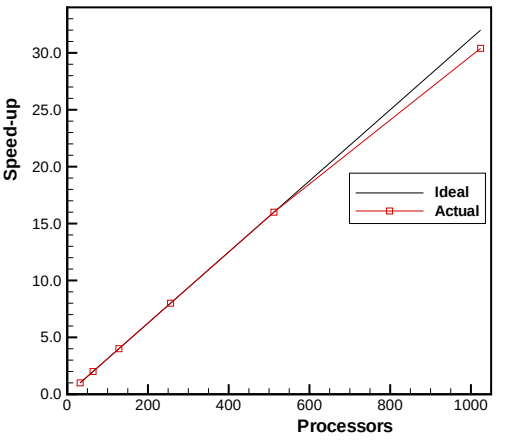

Fig. 4: *Parallel Performance.*

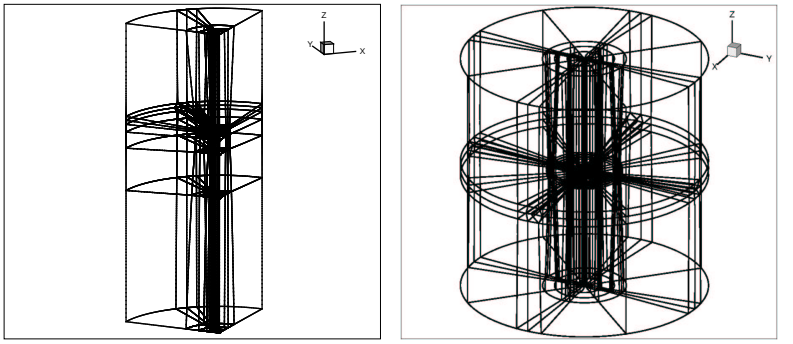

Fig. 1 : *Multiblock Domain and Block Boundaries. Hover and Forward Flight.*

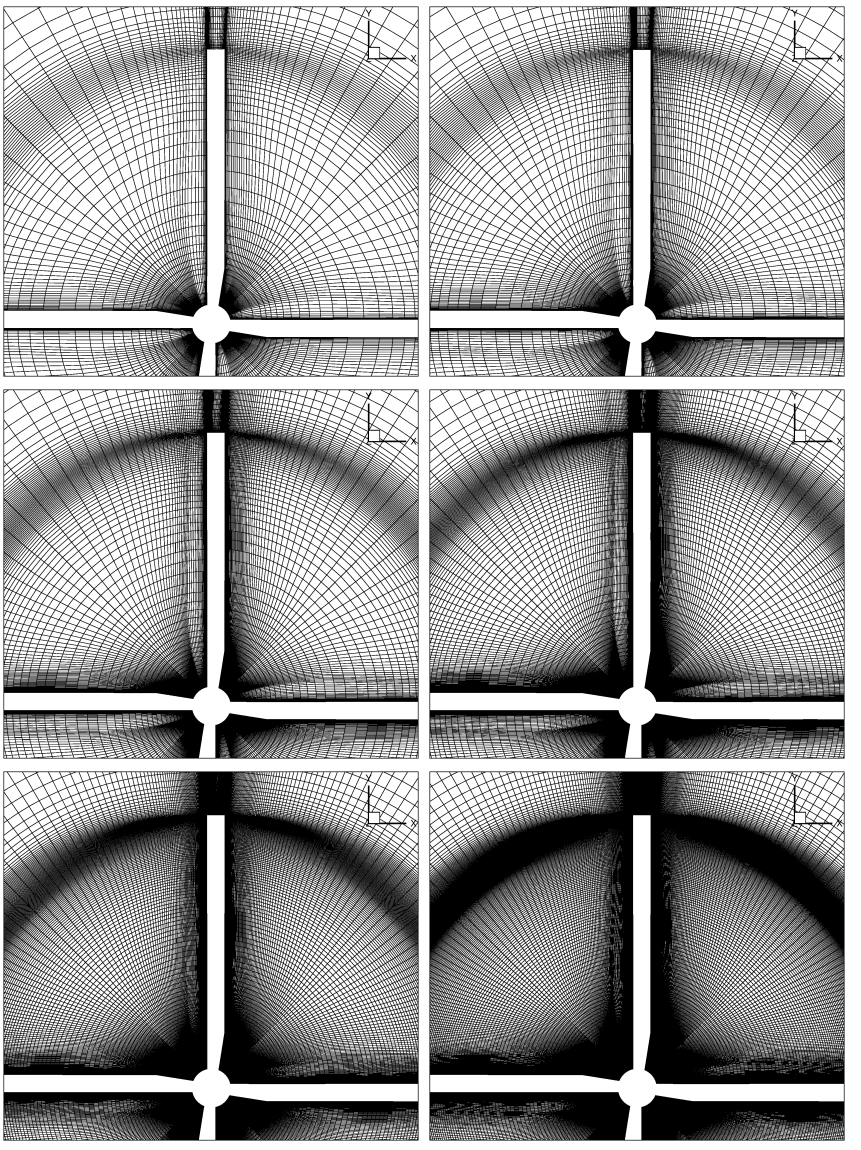

Fig. 2 : *Multiblock Forward Flight Mesh. Rotor disk grid,* $1, 2, 4, 8, 16, 32 \times 10^6$ *points.*

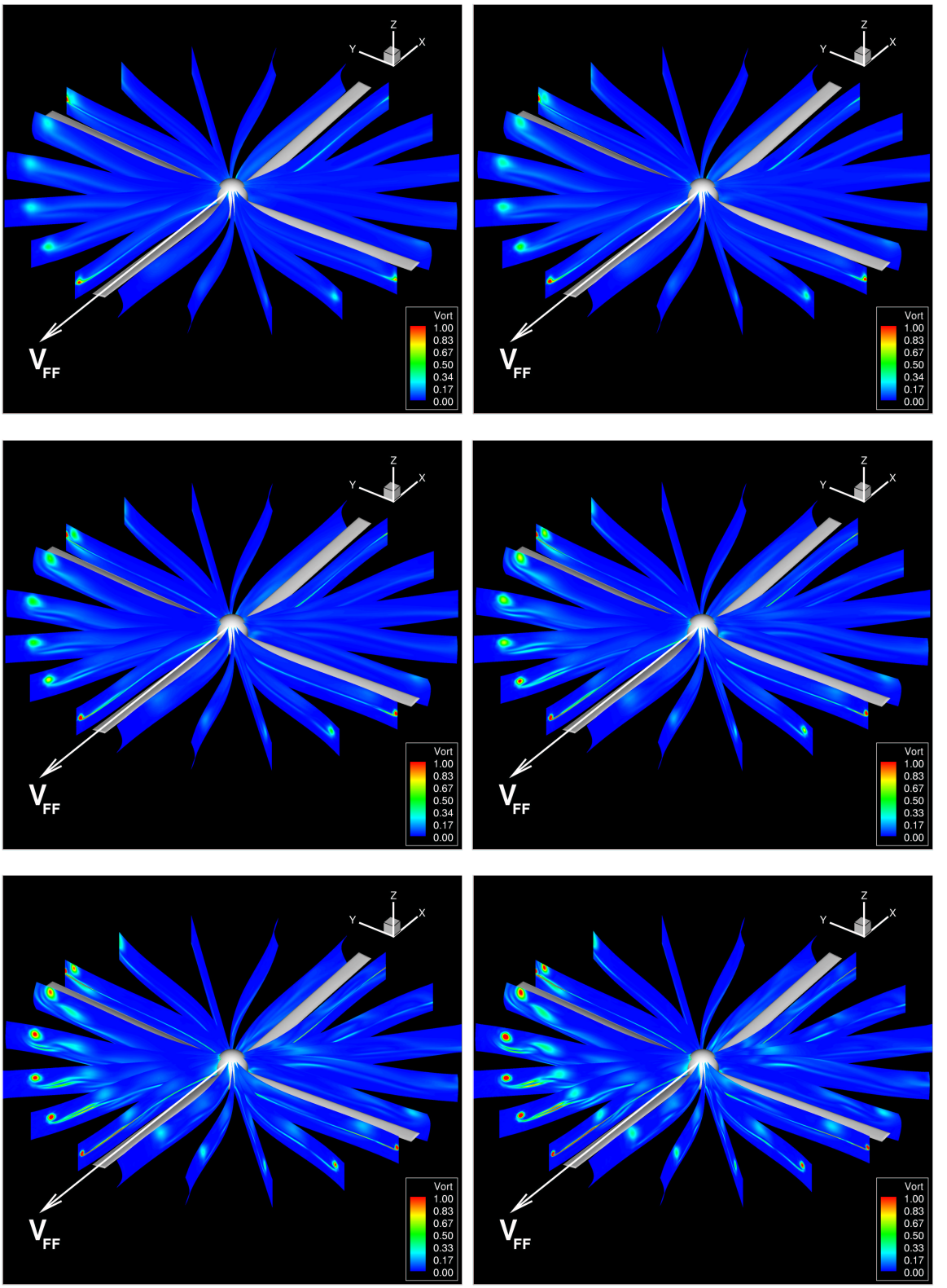

Fig. 3: *7A Forward Flight Vorticity Shading.*