Achieving high parallel performance for an unstructured unsteady turbomachinery CFD code

N. Hills
University of Surrey
Guildford, UK

ABSTRACT

This paper describes the work done to achieve high parallel performance for an unstructured, unsteady turbomachinery computational fluid dynamics (CFD) code. The aim of the work described here is to be able to scale problems to the thousands of processors that current and future machine architectures will provide. The CFD code is in design use in industry and is also used as a research tool at a number of universities. High parallel scalability has been achieved for a range of turbomachinery test cases, from steady-state hexahedral mesh cases to fully unsteady unstructured mesh cases. This has been achieved by a combination of code modification and consideration of the parallel partitioning strategy and resulting load balancing. A sliding plane option is necessary to run fully unsteady multistage turbomachinery test cases and this has been implemented within the CFD code. Sample CFD calculations of a full turbine including parts of the internal air system are presented.

NOMENCLATURE

$C_{p,loss}$	total pressure loss coefficient
p_0	total pressure
$p_{0,inlet}$	total pressure at inlet

1.0 INTRODUCTION

Traditionally, improvements in gas turbine engine performance and efficiency have been derived from improvement of the components that constitute the main annulus. In recent years, however, as improvements in compressor and turbine performance have become more difficult to achieve, and as secondary air flows have increased to cope with higher turbine inlet temperatures, attention has focused on the interactions of secondary path flows (shroud leakage, cooling, and sealing flows) with the main annulus flow. Consequently, while the main passage flows and the secondary air system flows have traditionally been treated separately in CFD, there is increasingly common interest in establishing efficient and robust solution procedures able to calculate both air system and mainstream flows. The aim of this research was to carry out a simulation of a full turbine including both the main annulus and the secondary air system geometry. This research will contribute to current industry objectives of achieving savings of up to 2% specific fuel consumption from optimisation of the secondary air system and from optimisation of the interaction between the secondary air system and the mainstream flow.

A review of the main CFD methods in general design use for turbomachinery blading is given by Denton and Dawes[1]. Steady-state single row 3D Navier-Stokes solvers have been used for design for some years. Multistage versions of these solvers have been

Paper No. 3100. Manuscript received 25 May 2006, revised 7 February 2007 accepted 12 February 2007.

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

developed by, for example, Ni and Bogoian[2], Adamcyzk *et al*[3], Hah and Wennerstrom[4], Dawes[5], Denton[6], and Jennions and Turner[7]. These steady-state multistage solvers are based on applying circumferential averages of flow quantities at 'mixing planes' placed between the blade rows to approximate the unsteady interactions and this approach is now commonly used for design. These solvers have typically neglected the secondary path flows.

In recent years, there have been extensions of this type of solver to include more of the secondary air system geometry. Wallis *et al*[8] used an unstructured code to model a single blade row including the shroud leakage path. Bohn *et al*[9] modelled a two-stage turbine with shrouded blades, including cavities at the stator hub and rotor tip, although they did not model the full leakage path between the upstream and downstream cavities. Rosic *et al*[10] consider a number of different levels of shroud leakage modelling, including simulating the full geometry, for a three stage low pressure turbine. Gier *et al*[11] and Cherry *et al*[12] both also modelled three stage low pressure turbines including the full shroud leakage geometry. Most, although not all, of these extensions have been based on multi-block structured meshes.

From the internal air system perspective, Chew *et al*[13] give a review of the use of CFD for internal air system problems. Cavity flows are often dominated by rotational effects that lead to strong coupling between the components of the momentum conservation equations. In CFD studies, this often results in slow convergence. Typically, a rotating disc cavity problem will require an order of magnitude more computing time to converge than a 'standard' turbomachinery blading flow problem with the same number of mesh points. As described by Virr *et al*[14], the traditional air system modelling stopped the cavity domain at the exit into the mainstream. However, there has been considerable recent research on the interaction of the mainstream and secondary air system flows in order to predict the level of hot mainstream gas that is ingested into the rotating disc cavity. Hills *et al*[15] performed an unsteady simulation of a turbine stage and rotating disc cavity and showed that the unsteady effects had a significant effect on the level of hot gas ingestion. Cao *et al*[16] carried out computations relevant to an axial steam turbine for a rotating disc cavity and a mainstream section, but without modelling the blades. They showed that instabilities developed in the rim seal, obviously in this case independently of the unsteady blade effects, which again had a significant effect on the level of hot gas ingestion. The prediction of these instabilities depended on the sector size modelled. The presence of these instabilities was confirmed experimentally after the calculations were carried out. Jakoby *et al*[17] carried out an unsteady computation of a 1·5-stage turbine including both rotating disc cavities. Both Jakoby *et al* and Boudet *et al*[18] again showed that instabilities depending on the modelled sector size had an effect on hot gas ingestion. Boudet *et al* also showed that small amplitude low order frequencies (compared to the blade passing frequency) could also have a large effect on hot gas ingestion. Due to the greater complexity of secondary air system geometries, which are commonly determined by mechanical rather than aerodynamic considerations, unstructured meshes have been more commonly used than for mainstream applications.

The aim of the current work was to combine the mainstream and secondary air system modelling approaches and apply them to a real gas turbine, specifically the Rolls-Royce Trent 500 engine. The geometrical complexity involved in including the internal air system geometry means that the use of fully unstructured meshes is attractive as this minimises the length of time spent in the mesh generation process. As discussed, in order to predict the level of hot gas ingestion, the problem needs to be modelled in a fully unsteady manner. In order both to avoid modifying the blade counts, and to capture any instabilities driving ingestion of hot mainstream gas into the secondary air system cavities, large sector (or 360°) models are necessary. All of these factors make the meshes and resulting run times extremely large. Meshes of 100 million nodes and upwards are required for this type of calculation.

Under the UK Applied Aerodynamics Consortium (UKAAC), CPU time for this research has been obtained on the UK's flagship high performance computing service, HPCx. The HPCx machine consists of 96 IBM POWER5 eServer nodes, corresponding to 1536 processors. Up to 1024 processors are available for one job. Each eServer node has 32 Gbyte of memory. Clearly, good parallel scalability is necessary in order to make efficient use of this type of machine. The aim of the work described in this paper was to achieve high parallel performance of the CFD code used for the level of modelling and geometrical complexity necessary to carry out the described simulation on this massively parallel system.

The CFD code used is the Rolls-Royce plc code, HYDRA. This code is employed both within Rolls-Royce for design and as a research tool at its university partners. The code is used in a number of United Kingdom and European Union research programmes, some of which research is described elsewhere in this special edition.

Section 2 describes the code and its parallel implementation further. A basic steady multistage test case is considered in Section 3 and various code improvements necessary to obtain good parallel performance are described. Unsteady simulations are then considered in Section 4. The implementation of the sliding plane necessary to replace the mixing plane for a fully unsteady calculation is described, and also the work done to maintain the parallel performance for these cases. Finally, unstructured meshes, which are made necessary by the geometrical complexity of the secondary air system components, are discussed in Section 5.

2.0 CODE DESCRIPTION

The Rolls-Royce plc code, HYDRA, consists of a suite of nonlinear, linear, and adjoint solvers with associated pre and post processing tools developed collaboratively by Rolls-Royce plc and its university partners. HYDRA is a fully unstructured code using an edge-based data structure with the flow data stored at the cell vertices. The flow equations are integrated around dual control volumes using a standard flux-differencing algorithm. A number of turbulence models are available in the code and the Spalart-Allmaras model[19] has been used throughout the work described here. For the steady state solver (Moiner and Giles[20]), the resulting discrete flow equations are preconditioned using a block Jacobi preconditioner (Moinier[21]) and iterated towards steady-state using the five stage Runge-Kutta scheme of Martinelli[22]. Convergence to steady state is accelerated through the use of an edge-collapsing multigrid algorithm (Crumpton *et al*[23]). For the unsteady solver, a dual time stepping scheme is used (Crumpton and Giles[24]). At Rolls-Royce plc, HYDRA is commonly used within an automated design system and details of this system are given by Shahpar *et al*[25].

The HYDRA solver has been parallelised using the domain decomposition method. The parallelisation within HYDRA was based on the OPLUS library, originally developed at Oxford University[26]. The aim of the library was to separate the necessary parallel programming from the application programming. A standard sequential FORTRAN DO-loop could be converted to a parallel loop by adding calls to OPLUS subroutines at the start of the loop. All the message passing and book-keeping was handled by the library. The domain partitioning strategy used was recursive geometric partitioning, either based on the principal moment of inertia of the original geometry, or on the coordinate axes.

In the original version of OPLUS, a master process handled the partitioning, initialisation, and all I/O. This process communicated with the compute processes via PVM[27] while communication between the compute processes was performed using the BSP library[28,29] (also developed at Oxford University). In 2001, with MPI increasingly becoming standard for message passing, a new version, OPLUS-MPI, was produced by the author which performed all communication via MPI, but kept the same structure of a master process and compute processes.

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

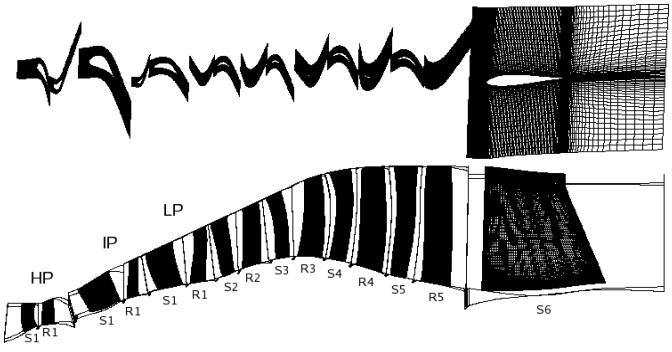


Figure 1. Meshed domain for steady state multistage full turbine model.

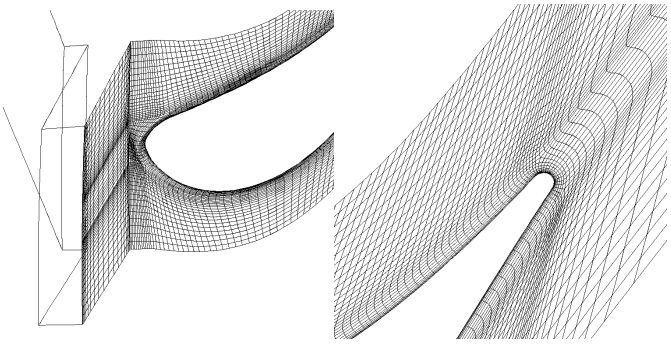


Figure 2. Mesh around HP rotor leading edge and trailing edge.

With increasing problem size, the requirement that a master process read in the geometry and carry out the partitioning and initialisation became a significant constraint. This was particularly true on HPCx as the memory available to any one processor was limited to approximately 1 Gbyte. Hence in 2004, OPLUS-MPI was extensively re-written by the author to perform the partitioning and initialisation entirely in parallel, thereby eliminating the requirement for a master process. This version of OPLUS-MPI formed the starting point for the work described here. Further modifications to OPLUS-MPI have since been carried out and these are discussed in this paper.

3.0 STEADY-STATE TEST CASE

The meshed domain for a steady state multistage turbine calculation for a three shaft engine turbine is shown in Fig. 1. This model consists of one high pressure (HP) stage, one intermediate pressure (IP) stage, and 5·5 low pressure (LP) stages. Solutions for each blade row are calculated in the relevant stationary or rotating frame and mixing planes are placed between each frame. A fast automatic block structured mesh generator developed at Rolls-Royce plc by Shahpar[30] was used to produce the grids for this problem. Since the injected coolant flow (from both blade film cooling flows and leakage flows from the internal air system back into the mainstream) corresponds to around 25% of the mainstream flow, it is impossible to ignore this mass addition for the full multistage turbine model. Hence a blade film cooling model is employed and 'stub' cavities are used to introduce the leakage mass flow. The mesh used consists of approximately 800,000 nodes per blade passage, leading to a total mesh size of 22 million nodes and 67 million edges. Since an automatic mesh generator was used, the mesh is fairly similar in each blade row. A close up of the mesh for the leading and trailing edges of the HP rotor is shown in Fig. 2 with the stub cavity geometry visible. The stub cavity mainstream inlet has been placed downstream of the mixing plane between each blade row. This model represents a state-of-the-art design computation.

The boundary conditions for the model correspond to conditions at cruise. At the inlet, total pressure is assumed constant, but there is a strong radial temperature profile typical of conditions at the exit from the combustion chamber in an engine. This radial temperature profile leads to strong secondary flows in the HP stage.

Figure 3 shows the contours of pressure loss coefficient for the HP NGV in a plane 18% of the NGV axial chord downstream of the trailing edge. The pressure loss coefficient is defined as;

$$C_{p,loss} = \frac{p_{0,inlet} - p_0}{p_{0,inlet}} \quad \ldots (1)$$

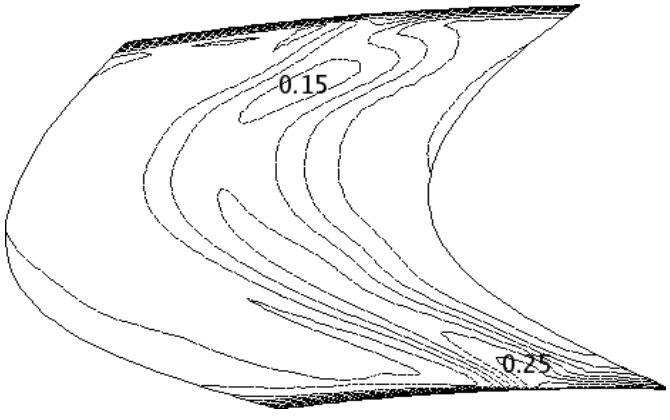


Figure 3. Contours of pressure loss coefficient on an axial plane 18% of axial chord downstream of HP NGV (Contour Interval = 0·05).

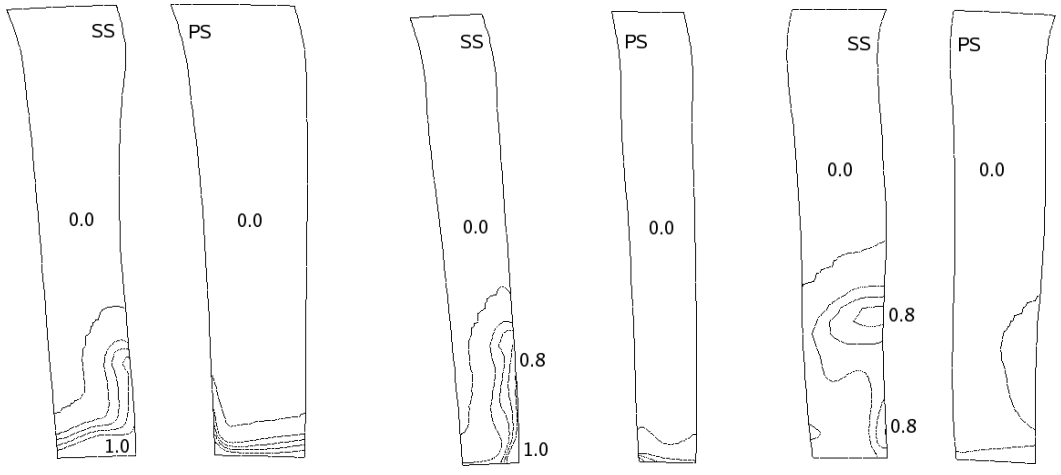
where $p_{0,inlet}$ is the total pressure at inlet and p_0 is the local total pressure. The loss cores of the secondary flow vortices can clearly be seen (and are marked with the loss coefficient value).

One of the aims of this research is to investigate the interaction of the coolant leakage flow with the mainstream flow. In order to trace the leakage flow, a passive scalar is used with a boundary condition of 1 at the inlet for the leakage flow of interest, and 0 at the mainstream inlet. Hence the concentration of this passive scalar describes the concentration of the specified leakage flow in the flow field. Contours of a passive scalar with an inlet boundary condition of 1 for the leakage flow entering via the stub cavity between the HP NGV and HP rotor are shown in Fig. 4 for three axial planes through the HP rotor (at 10%, 50%, and 90% axial chord). It can be seen that the coolant flow becomes entrained by the secondary flows in the blade passage, moving from the pressure side to the suction side of the blade and joining the main passage vortex.

This steady state case was used to test the scaling performance of the code. The scaling results for the initial version of OPLUS-MPI are given in Fig. 5 (labelled as 'Original'). The speed up is given relative to the performance on one 32-processor SMP node, and hence is a non-dimensional factor. The ideal linear speed-up is also shown on the graph. The ideal speed up assumes perfect parallel efficiency, so that, for example, doubling the number of processors will halve the code (wall clock) runtime regardless of the number of processors used initially. The code demonstrates a parallel efficiency of 65% on 256 processors relative to the 32-processor performance. There is then almost no further speed-up obtained from using 512 processors.

The quality of the mesh partitioning has been shown to have a strong effect on parallel performance for CFD codes (see, for example, Gropp *et al*[31]). It is important both to load balance the

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press



(a) 10% axial chord (b) 50% axial chord (c) 90% axial chord

Figure 4. Contours of passive scalar concentration on axial planes through HP rotor.
(coolant flow = 1, mainstream flow = 0, contour Interval = 0·2)

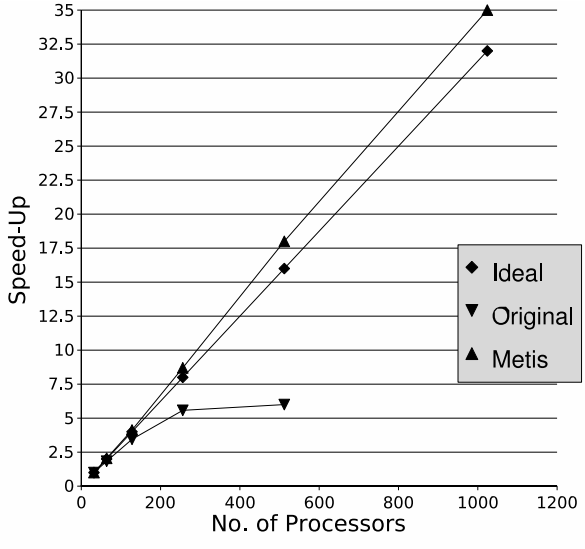


Figure 5. HYDRA Scaling results for steady state model.

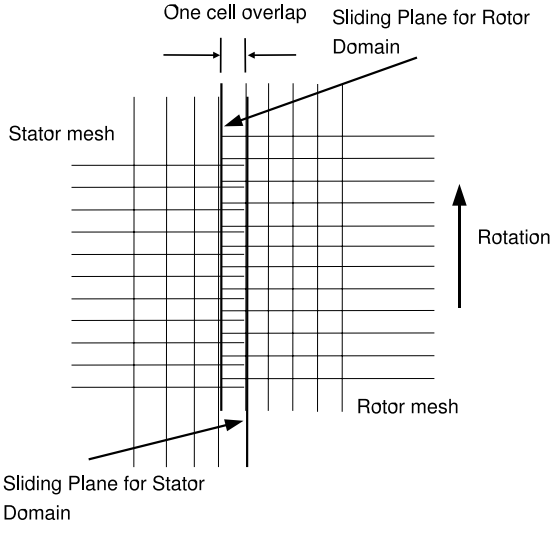


Figure 6. HYDRA sliding plane implementation.

work carried out per processor to avoid processors sitting idle at synchronisation points, and also to minimise the size of the messages passed between processors. In order to improve the parallel scaling performance of HYDRA, the partitioning was changed from the recursive geometric partitioning method described in the introduction to the *k*-way graph partitioning available in the ParMeTis package[32]. In the past few years, MeTis and the parallel version ParMeTis seem to have become the de facto standard partitioning packages used for CFD codes. The default graph partitioning algorithm attempts to minimise the edges cut by the partitions (and hence minimise the message size) while load balancing the nodes.

The parallel performance of the code was also examined using the VAMPIR (Visualisation and Analysis of MPI Resources) profiling package[33]. VAMPIR is a commercial post-mortem trace visualisation tool from Intel GmbH Software and Solutions Group. It uses the profiling extensions to MPI and permits analysis of the MPI events where data is transmitted between processors. Based on this analysis, some re-coding was carried out to minimise code bottlenecks. The re-coding carried out was fairly straight-forward. The output from VAMPIR enables one to determine in which MPI routines the processes are spending the most time. In the initial version of OPLUS-MPI, data was transferred between processes as datatype MPI_Packed to allow for use on heterogeneous clusters. MPI_Pack and MPI_Unpack are relatively expensive operations and the code was spending a high percentage of time in these calls. Consequently, the code was duly modified to transfer data as native datatypes.

Non-blocking sends and receives were used throughout, and, after this initial modification, the re-coding consisted of attempting to only complete each send or receive after carrying out all possible computation. The original OPLUS design[26] divided all loops into two parts: the first part only over data that was wholly local to the process, the second part over the halo data. Hence the message sends and receives are posted before the first part of the loop but do not

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

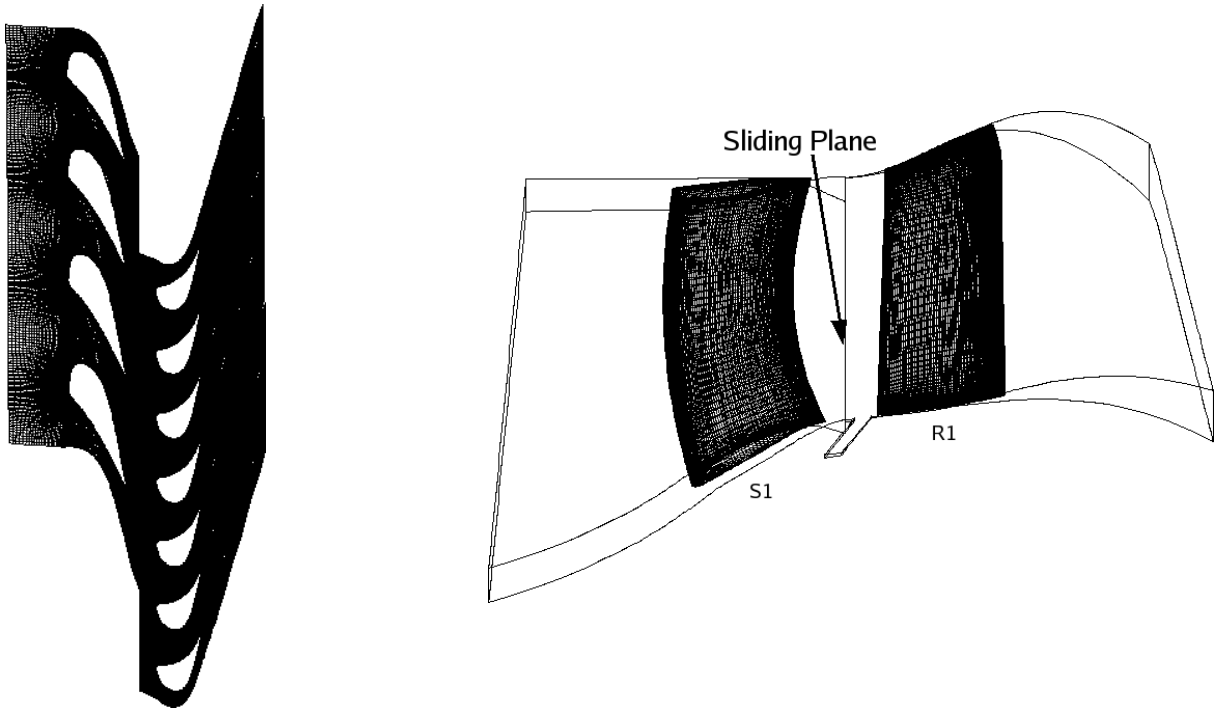


Figure 7. Unsteady HP calculation meshed domain.

need to be completed until after the first part of the loop and before the second part. This design already allows a large amount of computation to be carried out before it is necessary to complete the sends and receives. The additional re-coding carried out focused on performing the message processing optimally. So, for example, when filling send buffers, logic was added to determine if any send buffers had no corresponding pending sends, so that all these buffers could be filled before completing any outstanding sends. Similarly, when receiving messages, once one receive was completed, all possible processing was carried out on the received buffer before completing another receive.

The improved scaling results are shown in Fig. 5 labelled as 'Metis'. Super-linear performance is shown up to 1024 processors (corresponding to around 20k nodes per processor). The super-linear performance is presumably due to better cache utilisation at the higher number of processors. Most of the performance gain shown in Fig. 5 came from the code changes due to the profiling. Only for 512 and 1024 processors did the MeTis partitioning provide any speed-up over the original geometric recursive partitioning method. Codes that demonstrate excellent scaling properties on HPCx can apply for 'Seals of Approval' that qualify them for discounted use of CPU time. As a result of the scaling performance demonstrated in Fig. 4, HYDRA was awarded a gold seal of approval.

4.0 UNSTEADY TEST CASE

For a fully unsteady calculation, the mixing plane of Section 3 needs to be replaced by a time-accurate sliding plane treatment. The first implementation of a sliding plane treatment appears to have been by Rai[34] and a general discussion of the issues is given by Martelli[35].

This section describes the implementation of a sliding plane capability into HYDRA. The strategy adopted was to use meshes for the two frames which overlap by one cell row. This is illustrated for a 2D mesh in Fig. 6. The mesh overlap is created during a pre-processing step where the mesh spacing is adjusted at the sliding plane boundary and the two meshes are each extruded by one cell row. In 2D, the extruded mesh consists of quadrilateral cells (regardless of whether the original mesh consists of quadrilateral or triangular cells). In 3D, the extruded mesh consists of hexahedral cells (if a quadrilateral face on the boundary has been extruded) or prismatic cells (if a triangular face on the boundary has been extruded). In 3D, the sliding plane can take a general shape, providing it is axisymmetric with respect to the axis of rotation.

At each time step, the meshes are moved relative to each other. At the start of each time step sub-iteration, the solution variables are interpolated (with the relevant change of frame added) from the interior plane of the stationary mesh onto the exterior plane of the rotating mesh, and similarly from the interior plane of the rotating mesh onto the exterior plane of the stationary mesh. These values are then used as the boundary conditions for the relevant zone. At convergence of the time-step, these values are consistent between the zones.

Since the mesh is constructed so that there is a one cell overlay, the interpolation step requires a search only on a 1D line (for a 2D simulation) or on a 2D plane (for a 3D simulation) to locate the face containing each vertex. If a more general overlay was used, a full 2D or 3D search would need to be carried out.

The meshed domain for the test case used to evaluate the parallel performance is shown in Fig. 7. This test case consisted of just the HP stage from the previous model, but now modelling a sector of 4 vanes and 7 blades, and adding a sliding plane between the blade rows. This gave a mesh size of 7 million points and 22 million edges. Initial scaling results for unsteady calculations with this test case were somewhat inconsistent between test cases with reasonable scaling performance obtained for some test cases and extremely poor scaling performance obtained for others. Overall, though, the scaling performance was significantly worse than for the steady state test case of Section 3. This was discovered to be due to poor load balancing of the sliding plane interpolation phase of the calculation. The mesh partitioning described above assumes constant work per node and takes no account of the work associated with the sliding plane interpolation which only occurs for the sliding plane nodes. A synchronisation point occurs after the sliding plane interpolation phase, hence it was possible for one processor to be carrying out the entire sliding plane interpolation while all others were sitting idle waiting for this processor to finish.

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

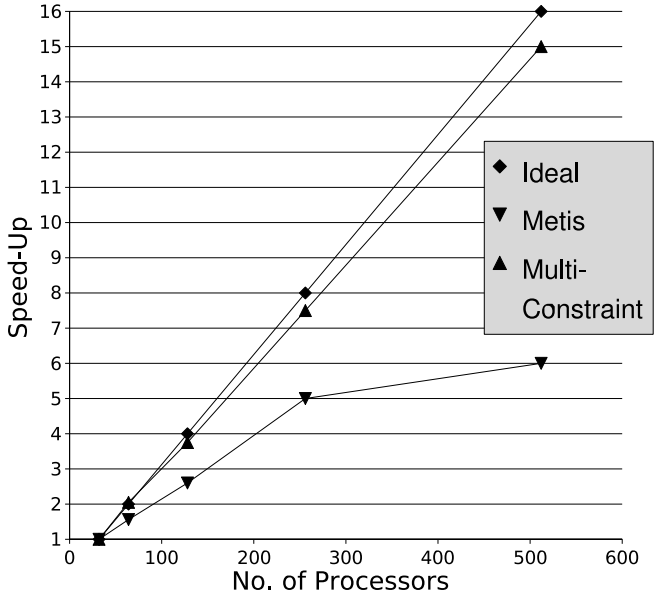


Figure 8. HYDRA scaling for unsteady sliding plane calculation.

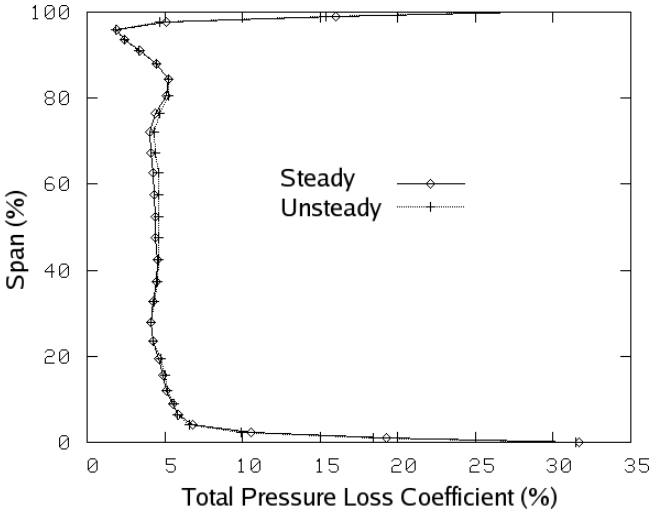


Figure 9. Circumferentially mass-averaged pressure loss coefficient *vs* span.

The ParMeTis package offers a multi-constraint partitioning routine to deal with multi-phase calculations[36]. Multi-phase calculations are effectively those with a synchronisation point between two calculations which require a different quantity to be load balanced. Due to the synchronisation point, it is not just sufficient to sum up the relative times for each phase and to compute a partitioning based on this sum. For an unsteady calculation with a sliding plane case, the number of nodes needs to be load balanced for the standard flux calculation and flow variable update, while the number of sliding plane nodes needs to be load balanced separately for the interpolation phase.

Scaling results for both the MeTis partitioning of the previous section and the multi-constraint partitioning strategy are shown in Fig. 8. Again the speed-up is given relative to the performance on one 32-processor SMP node and the ideal linear speed-up is also shown on the graph for comparison. As can be seen, the multi-constraint partitioning returns the scaling performance nearly to that obtained for the steady-state simulations. The slight degradation of performance may be due to the somewhat poorer partition (in terms of minimising the edge cut) that results from requiring the additional load-balancing constraint.

The full three-shaft engine turbine described in Section 3 was then run with the HP stage replaced by the unsteady HP stage model described in this section. Figure 9 gives the spanwise variation of circumferentially mass-averaged total pressure loss coefficient for the HP NGV for the unsteady and steady solutions, and it can be seen that the difference is negligible. In the rotor, however, the details of the secondary flows are strongly influenced by unsteady effects, as can be seen in Fig. 10, which shows the passive scalar contours for the HP leakage flow (equivalent to Fig. 4 for the steady-state case). There are clear differences between the secondary flows in the different blade passages, corresponding to different instantaneous positions of the rotor with respect to the NGV. The change in the details of the secondary flows can also be seen in Fig. 11, showing the circumferentially mass-averaged whirl angle approximately half an axial chord downstream of the HP rotor.

5.0 UNSTRUCTURED MESH TEST CASE

As the geometrical complexity of the cases being run increases, so do the advantages of using unstructured meshes. A Pro/Engineer model of the HP stage of the turbine being modelled is shown in Fig. 12. As can be seen, the task of meshing this geometry using a block structured approach is considerable. Instead, an unstructured mesh has been generated using the commercial mesh generator ICEMCFD. Prismatic cells are grown away from the solid surfaces for the boundary layer mesh and tetrahedra are used elsewhere in the domain.

Again, initial scaling results for problems using this type of mesh were inconsistent between test cases and again this turned out to be due to poor load balancing. The majority of work (around 75%) carried out by HYDRA is done in the edge based loops calculating the fluxes, while around 20% of the work is done in node based loops (adding source terms and updating the variables). The default load balancing using ParMetis balances the number of nodes between the processors. While for meshes using one type of cell (either fully hexahedral or full tetrahedral) load balancing the number of nodes also approximately load balances the number of edges, this is no longer as true for the mixed meshes considered here. It is necessary to balance the work load associated with each node. This can be done within ParMetis by attaching a weight to each node based on the number of edges attached to it, and the relative amount of work associated with the edge based loops and the node based loops. This method was implemented within OPLUS-MPI and this duly restored the parallel performance to that obtained on the block structured hexahedral meshes used in Sections 3 and 4.

The domain for an unsteady unstructured calculation for the HP stage including the secondary air system cavities is shown in Fig. 13. The model again consisted of a sector of 8 vanes and 14 blades. The increase in sector size from the unsteady model in Section 4 is to maintain the correct number of bolts in the secondary air system cavity. Only one vane, one blade, and one bolt are shown in Fig. 9 for clarity. This mesh consisted of 19 million nodes and 91 million edges. There are two sliding planes between the vane and blade and between the secondary air system cavity and mainstream (which is necessary due to the presence of stationary bolts and rotating cover

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

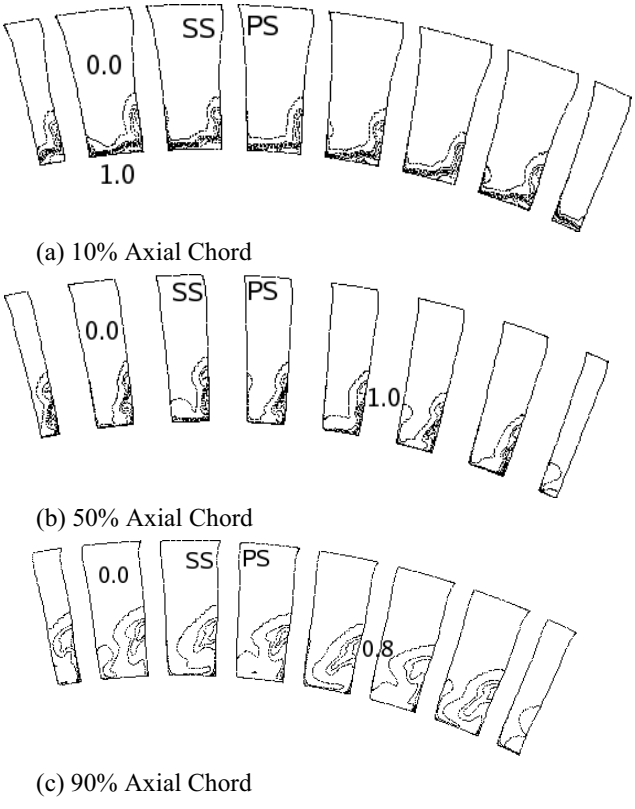


Figure 10. Contours of passive scalar concentration on axial planes through HP rotor for unsteady calculation (coolant flow = 1, mainstream flow = 0, contour interval = 0·2).

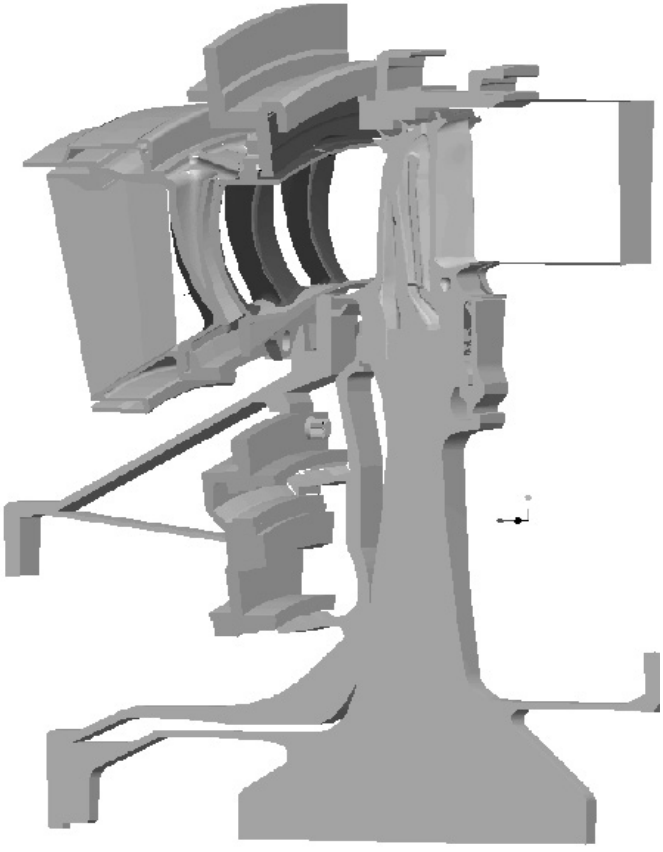


Figure 12. Pro/engineer model of HP stage.

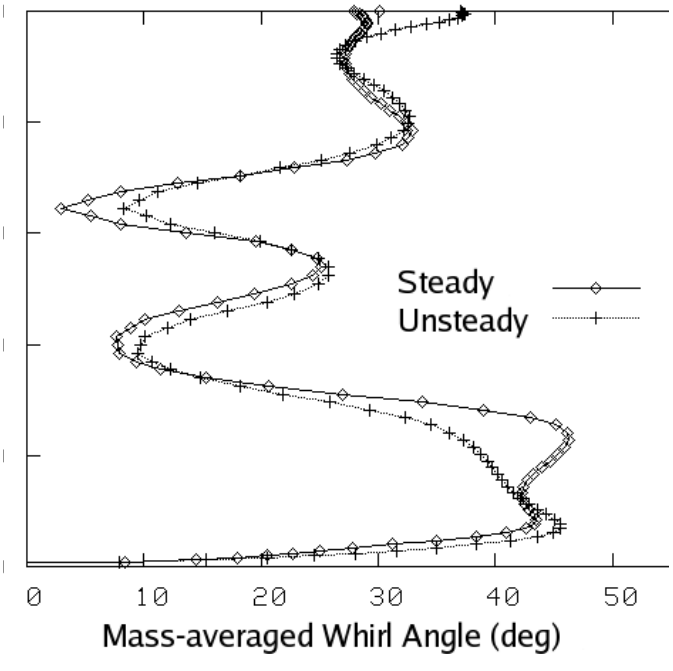


Figure 11. Circumferentially mass-averaged whirl angle *vs* span.

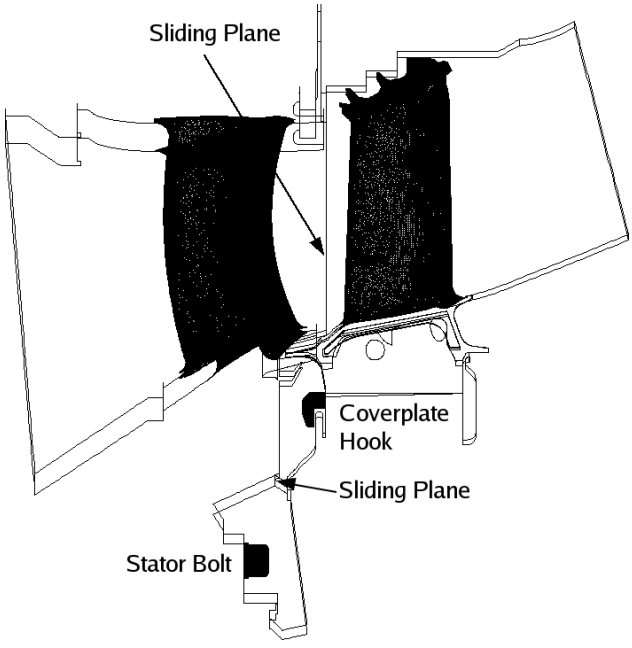


Figure 13. Unsteady unstructured mesh calculation domain.

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

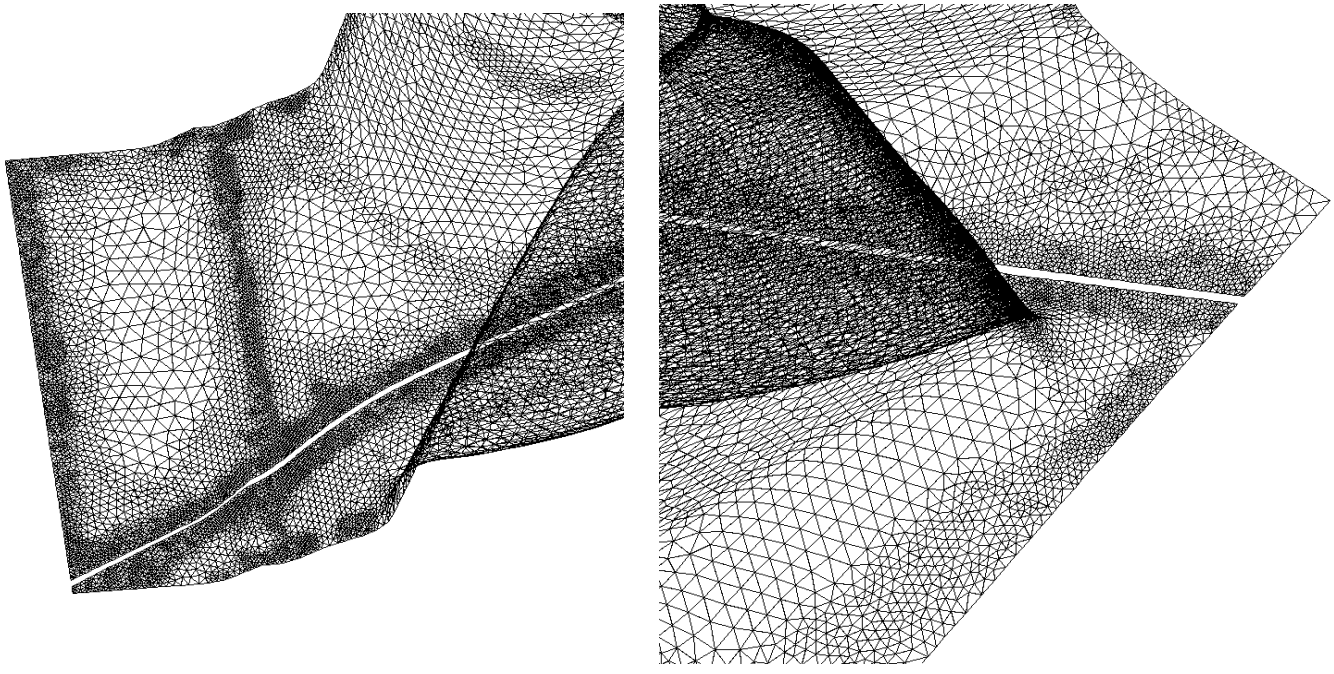


Figure 14. Leading and trailing edge unstructured mesh for HP rotor.

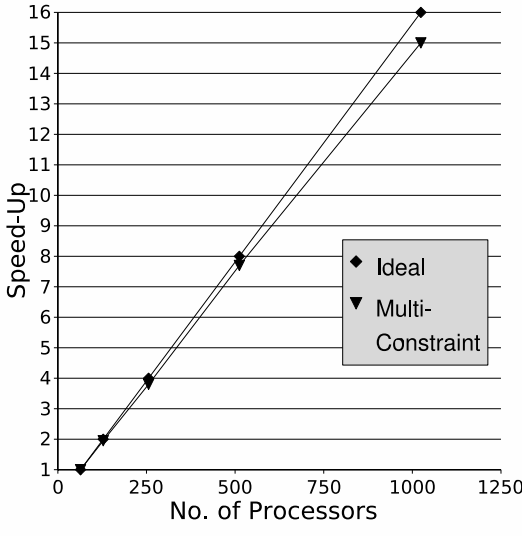


Figure 15. Scaling for Unsteady Unstructured Mesh calculation.

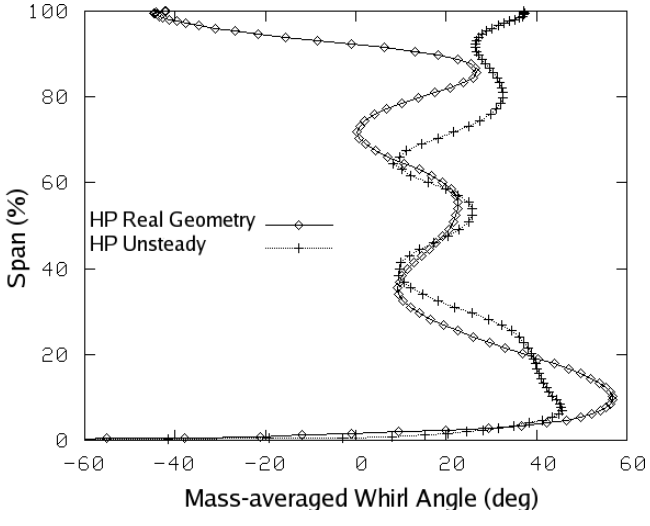


Figure 16. Circumferentially mass-averaged whirl angle *vs* span.

plate hooks). The mesh around the HP rotor leading and trailing edges is shown in Fig. 14 for comparison with the structured mesh of Fig. 2. The gap in the mesh in the centre of the blade passage is because the inter-platform gap and rotor damper is also being modelled. Scaling results for this problem are shown in Fig. 10. Due to the larger problem size, for this case speed up is shown relative to the performance on two 32-processor SMP nodes and the ideal linear speed-up is also shown on the graph for comparison. As can be seen, even for this extremely demanding test case, near linear speed-up is obtained up to 1024 processors.

Again the full three-shaft engine turbine described in Section 3 was run, only this time with the HP stage replaced by the real engine geometry HP stage model described in this section. The spanwise variation of circumferentially mass-averaged total pressure loss coefficient for the HP NGV was negligible between the real geometry and idealised geometry cases. However, the details of the secondary flows in the rotor are affected by the real geometry. Figure 16 shows the comparison of the circumferentially mass-averaged whirl angle approximately half an axial chord downstream of the HP rotor between the idealised geometry case and the real geometry case. The largest difference between the two cases is at the rotor tip, which is caused by the inclusion of the rotor blade tip gap (which can be seen in Fig. 13) in the real geometry model. Detailed consideration of the differences between the real geometry and idealised geometry cases will also require consideration of the level of numerical diffusion present in both models. While there is considerable experience of the mesh sizes required to obtain reasonable grid independence for the automatic structured mesh generator used, this level of experience has not yet been acquired for the use of unstructured meshes and for more complex geometries.

https://doi.org/10.1017/S0001924000004449 Published online by Cambridge University Press

6.0 CONCLUSIONS

The work done to obtain highly efficient parallel performance for an unstructured, unsteady turbomachinery CFD code, HYDRA, for a range of test cases has been described and near-linear scaling has been demonstrated. In order to achieve this level of performance, it has been necessary to consider each additional modelling complexity separately and ensure that no degradation of parallel performance occurred as they were added. Codes that demonstrate excellent scaling properties on HPCx can apply for 'Seals of Approval' that qualify them for discounted use of CPU time and HYDRA has been awarded a Gold Seal of Approval. The focus of the work has been on achieving the capability to carry out calculations for a full turbine and internal air system, requiring meshes over several hundred of millions of points. The ability to scale problems to thousands of processors opens the possibility of exploring engine models with much more detailed physical modelling and greater component interaction than has previously been achievable. Sample CFD calculations of the kind now possible have been presented. Future work will now concentrate on ensuring the numerical accuracy of the simulations and on detailed consideration of the physical mechanisms involved in the interactions between the internal air system and the mainstream flow.

ACKNOWLEDGEMENTS

This work has been funded by the Research Councils UK under an RCUK Academic Fellowship and by Rolls-Royce plc. This is gratefully acknowledged by the author, as is the help and interest of A. Smith, L. Lapworth, and T. Scanlon of Rolls-Royce plc. The author would also like to thank EPSRC for their support of the UK Applied Aerodynamics Consortium (UKAAC) under grant GR/S91130/01 and the CCP12 programme.

REFERENCES

1. DENTON, J.D. and DAWES, W.N. Computational fluid dynamics for turbomachinery design, Proc Istn Mech Engrs, 1999, **213** Part C, pp 107-204.
2. NI, R.H. and BOGOIAN, J.C. Prediction of 3D multistage turbine flow field using a multigrid Euler solver, AIAA paper 89-0203, 1989.
3. ADAMCYZK, J.J., CELESTINA, M., BEACH, T.A. and BARNETT, M. Simulation of viscous flow within a multistage turbine, *ASME J Turbomachinery*, 1990, **112**.
4. HAH, C. and WENNERSTROM, A. Three-dimensional flowfields inside a transonic compressor with swept blades, *ASME J Turbomachinery*, 1991, **113**.
5. DAWES, W.N. Towards improved throughflow capability: the use of 3D viscous flow solvers in a multistage environment, *ASME J Turbomachinery*, 1992, **114**.
6. DENTON, J.D. The calculation of three-dimensional viscous flow through multistage turbomachinery, *ASME J Turbomachinery*, 1992, **114**.
7. JENNIONS, I.K. and TURNER, M.G. Three-dimensional Navier-Stokes computations of transonic fan using an explicit flow solver and an Implicit *k-e* Solver, *ASME J Turbomachinery*, 1993, **115**.
8. WALLIS, A.M., DENTON, J.D. and DEMARGNE, A.A. The control of shroud leakage flows to reduce aerodynamic losses in a low aspect ratio, shrouded axial flow turbine, ASME paper 2000-475, 2000.
9. BOHN, D.E, BALKOWSKI, I., MA, H. and TUEMMERS, C. Influence of open and closed shroud cavities on the flowfield in a 2-Stage turbine with shrouded blades, ASME paper 2003-38436, 2003.
10. ROSIC, B., DENTON, J.D. and PULLAN, G. The importance of shroud leakage modelling in multistage turbine flow calculations, ASME paper 2005-68469, 2005.
11. GIER, J., STUBERT, B., BROUILLET, B. and DEVITO, L. Interaction of shroud leakage flow and main flow in a three-stage LP turbine, ASME paper 2003-38025, 2003.
12. CHERRY, D., WADIA, A., BEACOCK, R., SUBRAMIAN, M. and VITT, P. Analytical investigation of a low pressure turbine with and without flowpath endwall gaps, seals, and clearance features, ASME paper 2005-68492, 2005.
13. CHEW, J.W., HILLS, N.J., HORNSBY, C. and YOUNG, C. Recent developments in application of CFD to turbomachinery internal air systems, 5th European Turbomachinery Conference (ETC5), 2003.
14. VIRR, G.P., CHEW, J.W. and COUPLAND, J. Application of computational fluid dynamics to turbine disc cavities, *ASME J Turbomachinery*, 1993, **116**, pp 701-708.
15. HILLS, N.J., CHEW, J.W. and TURNER, A.B. Computational and mathematical modelling of turbine rim seal ingestion, *ASME J of Turbomachinery*, **124**, 2002.
16. CAO, C., CHEW, J.W., MILLINGTON, P.R. and HOGG, S.I. Interaction of rim seal and annulus flows in an axial flow turbine, ASME paper 2003-38368, 2003.
17. JAKOBY, R., ZIERER, T., LINDBLAD, K., LARSSON, J., DEVITO, L., BOHN, D., FUNCKE, J. and DECKER, A. Numerical simulation of the unsteady flow field in an axial gas turbine rim seal configuration. ASME paper 2004-53829, 2004.
18. BOUDET, J., AUTEF, V.N.D., CHEW, J.W., HILLS, N.J. and GENTILHOMME, O. Numerical simulation of rim seal flows in axial turbines, *Aeronaut J*, August 2005, (1089), **109**, pp 361-372.
19. SPALART, P.R. and ALLMARAS, S.R. A one-equation turbulence model for aerodynamic flows, *La Recherche Aerospatiale*, **1**, pp 5-21.
20. MOINIER, P. and GILES, M.B. Preconditioned Euler and Navier-Stokes calculations on unstructured grids, 6th ICFD Conference on Numerical Methods for Fluid Dynamics, 1998, Oxford, UK.
21. MOINIER, P. Algorithm Developments for an Unstructured Viscous Flow Solver, PhD thesis, 1999, University of Oxford, UK.
22. MARTINELLI, L., Calculations of Viscous Flows with a Multigrid Method, PhD. thesis, 1987, Dept of Mech and Aerospace Eng, Princeton University, USA.
23. CRUMPTON, P.I., MULLER, J-D. and GILES, M.B. Edge-based multigrid schemes and preconditioning for hybrid grids. *AIAA J*, **40**, pages 1954-1960, 2002.
24. P.I. CRUMPTON and M.B. GILES. Implicit time accurate solutions on unstructured dynamic grids. AIAA paper 95-1671, 1995.
25. SHAHPAR, S., GIACCHE, D. and LAPWORTH, L. Multi-objective design and optimisation of bypass outlet guide vanes. ASME paper GT-2003-38700, 2003.
26. BURGESS, D.A., CRUMPTON, P.I. and GILES, M.B. *A Parallel Framework for Unstructured Grid Solvers*. In DECKER, K.M. and REHMANN, R.M. (Eds), *Programming Environments for Massively Parallel Distributed Systems*, 1994, pp 97-106, Birkhauser.
27. SUNDERAM, V.S. PVM: A framework for parallel distributed computing, concurrency, *Practice and Experience*, **2**, 1990, pp 315-339.
28. MCCOLL, W.F. *Scalable Computing. in Computer Science Today: Recent Trends and Developments*, Springer-Verlag, 1995.
29. VALIANT, L.G. A Bridging model for parallel computation. communications of the ACM, 1990, **33**, pp 103-111.
30. SHAHPAR, S. PADRAM: Parametric design and rapid meshing system for turbomachinery optimisation, ASME paper GT-2003-38698, 2003.
31. GROPP, W.D., KAUSHIK, D.K., KEYS, D.E., and SMITH, B.F. High Performance Parallel Implicit CFD, *Parallel Computing*, 2001, **27**, pp 337-362.
32. KARYPSIS, G. and KUMAR, V. A FAST and high quality scheme for partitioning irregular graphs, *SIAM J Scientific Computing, 1999*, **20**, pp 359-392.
33. Pallas high performance computing products, http://www.pallas.com
34. RAI, M.M. Unsteady 3D Navier-Stokes simulation of turbine rotor-stator interaction, AIAA paper 87-2058, 1987.
35. MARTELLI, F. Unsteady flow modelling in a turbine stage, annals of the New York Academy of Sciences, 2000, **934**, pp 80-95.
36. KARYPSIS, G. and KUMAR, V. Multilevel algorithms for multi-constraint graph partitioning, Proc. Supercomputing '98, 1998.