IGTC2007-ABS-13

Enhancing the Productivity and Quality of the CFD Process in Turbomachinery Design

Carlos Felipe Favaretto and Bill Dawes

Cambridge Flow Solutions
Compass House, Vision Park, Histon, Cambridge, CB24 9AD, UK
Phone: +44-1223-257979, FAX: +44-1223-257800, E-mail: felipe.favaretto@cambridgeflowsolutions.com

ABSTRACT

This paper describes the development of two computational tools aimed at tackling difficulties encountered by engineers when conducting CFD analysis during the design phase of modern gas turbines. The first tool is focused on minimizing the computational time and labour required to modify an existing mesh. Traditionally, modifying a mesh requires editing its respective geometry at CAD level, exporting it to the mesh generator, and regenerating the whole mesh from scratch. The authors propose instead a mesh morphing tool which manipulates the nodes directly without the necessity of revisiting the CAD system nor the mesh generator and uses robust optimization techniques to control the quality of the outgoing mesh. The second tool is a Level Set based geometry kernel integrated with an octree-based cut-Cartesian mesh generator, *RANS* solver and post-processor. This novel approach provides rapid and fully automated mesh generation for complex geometries and also allows arbitrary topology edits to the geometry in the spirit of real time computer game. The tool was successfully integrated within the framework of a widely used design optimization system. Turbomachinery applications using the proposed computational tools are presented.

INTRODUCTION

So far, the development in the CFD industry has been focused mainly on improving the accuracy of the solver as well as minimizing computational costs by implementing parallel processing libraries. The low cost and processing power of PC clusters have enabled designers to analyze large number of configurations overnight, providing a rich database of candidate geometries to be chosen from. As a consequence, the bottlenecks in the CFD process have been shifted towards the pre-processing and results visualization steps. Cutting-edge technology in turbomachinery design assisted by CFD requires the ability to generate large computational grids based on detailed CAD models with the least possible user interaction and in a short period of time. The design system must also be flexible enough to allow the designer to freely modify the geometry and rapidly regenerate a high quality mesh. The CAD to mesh process in conventional CFD is simply not robust enough to permit meaningful automation of design space exploration. In addition to that, the CAD to mesh process is done in serial, different software is used for each of the tasks in the process (requiring different licenses, different file formats, different training for engineers) and coupling between CFD and FEA models is not straightforward.

Manuscript Received on April 30, 2007

With all these challenging issues in mind, the authors would like to describe the development of two powerful computational tools. The first tool morphs an existing mesh and it is focused on controlling the quality of the output mesh. The second tool, on the other hand, permits arbitrary topology geometrical changes and its approach is completely different from traditional CFD. In both cases the objective is to bypass the CAD-mesh bottleneck and permit meaningful automation.

MESH MORPHING TOOL

The working principle of the mesh morphing tool is summarized in Fig.1. The software reads in a mesh file from an external CFD code, morphs it and exports a high quality mesh without the necessity of revisiting the original mesh generator. This methodology allows designers to save substantial amount of time for cases where local and moderate changes in the mesh are to be systematically applied.

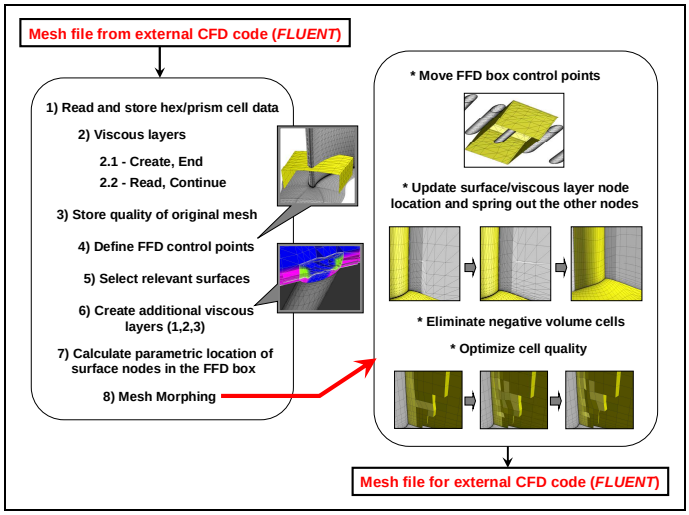


Fig.1 - Working principle of the mesh morphing tool.

The software starts by reading the basic hexahedral/prismatic cell data and generating the cell faces. On the first run of the software the viscous layer flagging mechanism is executed (Fig.2). This process consists of automatically associating nodes in the refined parts of the grid near the wall boundaries (viscous layer) with a root surface node. The basic principle of viscous layer flagging for a surface node with adjacent hexahedral or prism cells is to find the node from these neighboring cells that shares the maximum number of common hexahedral/prism cells with this

particular surface. Since the orientation of the nodes in the cell is known it is possible to easily find one out of three neighboring nodes that matches the criterion. In Fig.2 the reference (root) surface node *4* points to two surface nodes (*1* and *3*) and one internal node (*8*). Therefore, node *8* belongs to level *2* whereas nodes *1, 3* and *4* belong to the root level or level *1*. The process is repeated until all levels (defined by the user) have been flagged. This methodology is implemented in a very straightforward manner allowing layers with mixed cell types (hexahedrons and prisms) to be flagged by the same routine. There are cases, however, in which additional layers need to be created in the regions between existing layers and one single surface node might be the root node for an array of *nlayer(surface 1) x nlayer(surface 2) x nlayer(surface 3)* nodes. Additional routines have been coded to cover these special cases. Once the flagging process is complete the data is stored in a pointer file. This process is only necessary to be done once and for such reason the program is ended in the first run.

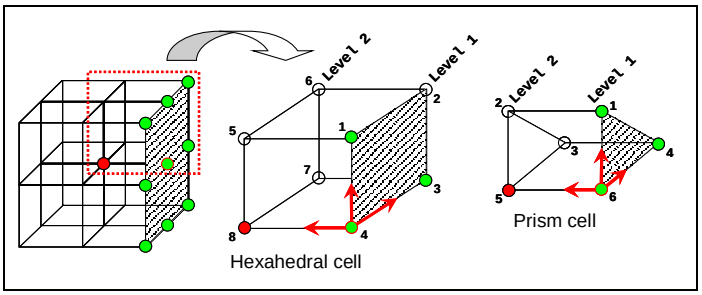


Fig.2 - Viscous layer flagging mechanism.

On the second run of the software the viscous layer data is read (step 2.2) and the mesh quality of the original mesh is stored. A Free-Form Deformation (*FFD*) box is created around the region to be morphed based on the input data provided by the user (box resolution, vectors defining the edges of the box, biasing factors controlling the distance between each control point). The parametric location of the surface nodes in the *FFD* box is calculated by using an *inverse spline* approach (step 7). The control points of the *FFD* box are translated based on the displacement matrix specified by the user in the input file. The displacement of the surface nodes and viscous layer tree nodes, if any, is then updated. The coordinates of all remaining nodes are repositioned by using a spring model analogy (Battina, 1990), producing a smooth mesh between the morphed and unchanged parts of the grid. A quality control algorithm searches the domain for poorly shaped cells and fixes them according to a predefined criterion. This is a fundamental step for the success of the whole *CFD* process since the accuracy of the solution is directly related to cell quality. Step 8 can be part of a loop if a stack of morphed meshes is desired. This whole process has been linked to a commercial CFD solver (*Fluent*) and a high level scripting interface for process integration and optimization (*iSIGHT-FD*). The description of this process integration will be published in the near future.

Mesh quality control

One of the most important routines of the mesh morphing tool is the one controlling the quality of the outgoing mesh. Although the spring model usually produces a smooth mesh there are challenging cases in which cells with negative volume, high skewness or excessively warped faces are unavoidably created. For such cells a *tabu* based mesh optimization algorithm is used.

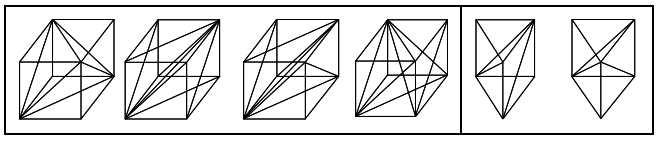


Fig.3 - Subdivision of incoming cells into tetrahedral cells.

The objective of the first run of the mesh quality optimizer is to search and eliminate negative volume cells. The incoming cell is first subdivided into tetrahedral cells using similar approach to the one described by Dompierre et al (1999), shown in Fig.3. The smallest volume among all the permutations of tetrahedral cells is the value stored as the volume of the original cell. The nodes belonging to each negative cell are then flagged. The optimizer moves the node along each of the edges connected to such node and the best case is stored. The process is repeated until the objective is achieved.

The second part of the optimization is to improve the mesh quality parameter *RPI* defined by the following equation:

$$RPI = \frac{V^2}{\left(\sum A\right)^3} \tag{1}$$

where V is the volume of a tetrahedral cell and A its respective area. For every original cell the *RPI* is defined as the lowest *RPI* among all possible permutation of tetrahedral cells. The optimization of *RPI* itself would not be of much use in regions where *good* quality cells with large aspect ratio are expected. For such reason the quality control marker *ΔRPI* was adopted instead:

$$\Delta RPI = \frac{\left|RPI_{morphed} - RPI_{original}\right|}{RPI_{morphed}} \tag{2}$$

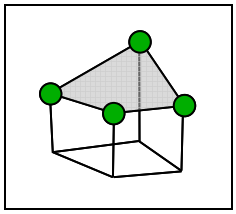


Fig.4 - Warped face of a hexahedral cell

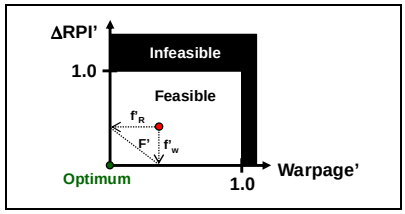


Fig.5 - Objective space for optimization of hexahedral/prismatic cell quality

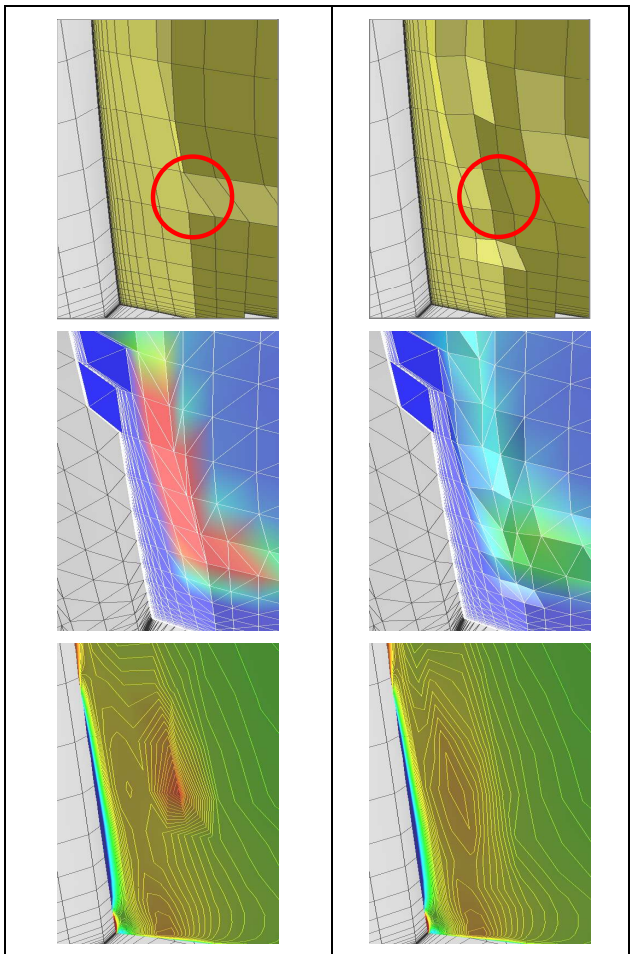
In some cases using *ΔRPI* as the sole figure of merit may cause the optimizer to find acceptable quality tetrahedrons inside a poor quality hexahedron/prism (Fig.4). This is because there is no constraint in terms of the flatness of the faces of the hexahedral/prism cell. For such reason, the face warpage metric was introduced. The cross-product of each pair of edges of a quadratic face is calculated, resulting in a *normal* for each of the four nodes. Warpage is defined as the worst case angle deviation among the *normals*. The nodal value of *ΔRPI* (f_R) is normalized by the highest value among all relevant nodes whereas the nodal value for warpage (f_W) is normalized by the fixed value of *0.5*. The figure of merit for the optimization is defined as:

$$F' = \sqrt{f_W'^2 + f_R'^2} \tag{3}$$

where F' is the objective function, f'_W the warpage and f'_R the *ΔRPI*, all in the normalized objective space (Fig.5). The optimization terminates when the objective criterion for *ΔRPI* (2.0) is reached. If the convergence criterion is not satisfied the upper limit for *ΔRPI* is redefined as the highest value from the previous run and the optimization is restarted.

In order to test the mesh quality control routine, a hexahedral structured mesh around a turbine blade was morphed. The top part of Fig.6a shows the morphed mesh before running the quality control routine. Note the trapezoidal shape of one of the low quality cells (red circle). In total, 203 nodes were flagged for quality improvement. The center plots in Fig.6 show the *ΔRPI* distribution

on a crystal cut of the mesh (hexahedral cells subdivided into tetrahedral cells). The red color indicates low mesh quality (high *ΔRPI*) whereas blue indicates high quality (low *ΔRPI*). It is also important to note that the color representation is node based. The value of a scalar for a particular node is actually the lowest value among all neighboring cells. After running 10 optimization steps the number of selected nodes for quality improvement was reduced to 16 and the largest *ΔRPI* was reduced from *34801.0* to *2.26*, which is already an acceptable value (center right plot of Fig.6b). In the top part of Fig.6b it is shown how the nodes belonging to low quality cells were moved around, slightly reducing the quality of some cells (by introducing a small amount of warpage) but improving the overall quality. In the bottom part of Fig.6 contours of *y-velocity* gradient in the *y-direction* calculated from *Fluent* results are shown. The impact of mesh quality on the solution can be qualitatively observed. The red core observed in the early stages of the mesh quality optimization is gradually smoothed out.



a) Before quality control b) After 10 steps

Fig.6 - Example of quality control routine acting on poorly shaped cells (top: hexahedral mesh; center: *ΔRPI*; bottom: *y*-velocity gradient in the *y*-direction).

Test case: leading edge filleting

Leading edge modification of inlet guide vanes is a technique that has been used by turbine designers to reduce secondary flows and minimize the associated aerodynamic losses (Becz et al., 2003) as well as to reduce adiabatic wall temperatures (Lethander et al., 2003). Finding the optimum shape of the junction between leading edge and endwall is a challenging task and is usually done by experience or with the help of an optimizer. The objective of this test case is to demonstrate the potentiality of the mesh morphing tool for integration within a fully automated design optimization system.

The control points of the *FFD* box near the leading edge were gradually displaced along the spanwise direction to generate a smooth convex surface, as shown in Figs.7 and 8. The main complication with respect to mesh quality was handled by a viscous layer blending approach.

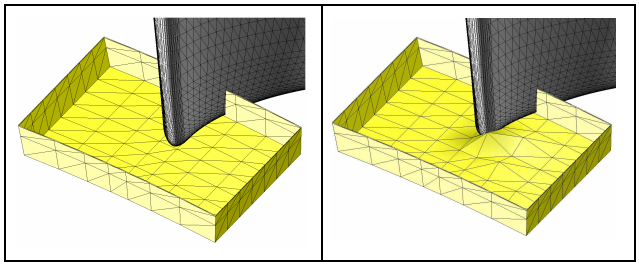


Fig.7 - *FFD* box for original and morphed meshes.

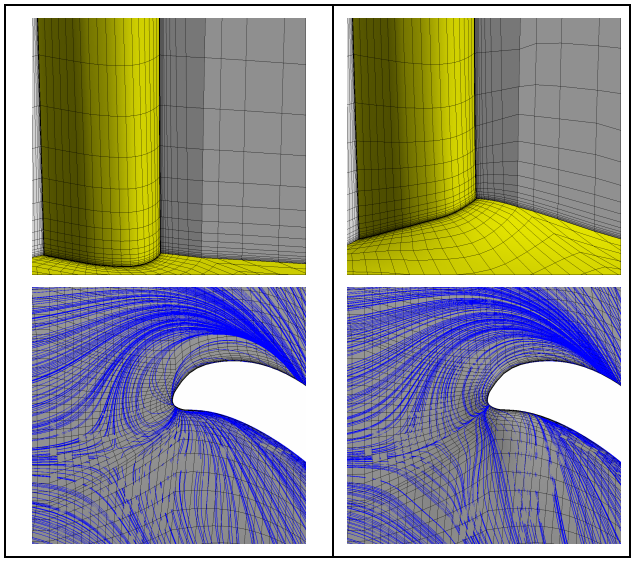


Fig.8 - Limiting streamlines and crystal cut of original and morphed meshes.

In the bottom part of Fig.8 limiting streamlines are shown for the original and morphed meshes. The solution was obtained by running *Fluent 6.2* with *k-ω SST* model and second order scheme for all variables. It is clearly observed that the location of the saddle point was shifted towards the leading edge due to the convex shape of the fillet. The endwall separation line was moved towards the leading edge of the blade, indicating that the penetration of the inlet boundary layer for the filleted case is smaller than the original one. As a consequence, the formation of the new boundary layer beneath the passage vortex occurs at an earlier stage for the filleted leading edge configuration.

AN INTEGRATED, PARALLEL, GEOMETRY ENGINE, MESH GENERATOR, FLOW SOLVER AND POST- PROCESSOR

This second computational tool uses a completely different approach from traditional CFD codes. The architecture of the software provides an integration of the solid modeling directly with the mesh generation and with the flow solution (Fig.9). The solid model is initialized by the import of a tessellated surface from a variety of potential sources, such as *STL* format. The solid model is then captured on the adaptive, unstructured Cartesian hexahedral mesh very efficiently by cutting the tessellated boundaries using basic computer graphics constructs developed for interactive 3D gaming. The uniqueness of this software is in its capabilities of providing the user a flexible graphical user interface for topological

editing, or *flow sculpting*, and rapidly updating the flow solution for the morphed geometry.

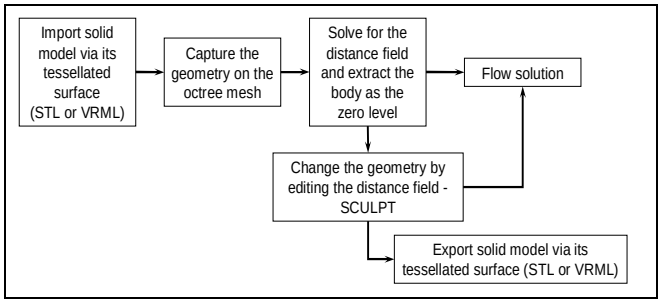


Fig.9 - Working principle of the integrated geometry engine, mesh generator, flow solver and post-processor.

The software has a built-in flow 3D *RANS* flow solver which was adapted from an existing unstructured mesh *RANS* solver. For additional information readers are encouraged to refer to the papers by Dawes (2005), Dawes (2006) and Dawes et al (2007).

Test case: internal cooling system

The employment of new concepts in the design of blades for modern gas turbines has pushed the orthodox CFD process to its limit. Interesting CFD calculations have been conducted by several authors for complex geometries such as internal cooling systems with arrays of pin fins, cooling holes, dust holes and ribs (Bucchieri et al, 2006, Kulasekharan and Prasad, 2006). In most cases the geometry is parameterized and the mesh is regenerated according to the new configuration that the optimizer is analyzing. It is convenient for some CFD analysts to instruct the mesh generator to create an unstructured mesh, create the prismatic viscous layers and finally generate an unstructured mesh using tetrahedral cells. There are other engineers who prefer to work with templates for creating multi-block structured meshes. In any case, the amount of time required to set-up the scripts for driving the mesh generator can be substantially large. In addition to that, the fact that a template must be defined *a priori* constrains the design space for the optimizer. The robustness of the system is also compromised since the mesh generator will simply abort if, for instance, one pin fin overlaps a neighbouring pin fin.

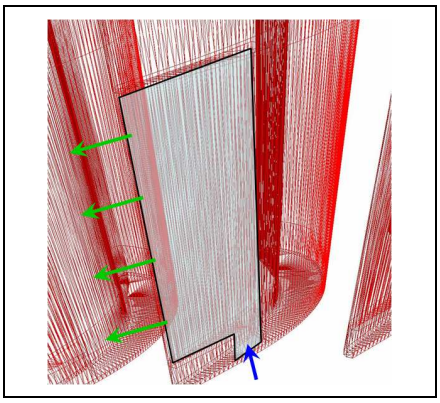


Fig.10 - Triangulated surfaces for a cascade with internal cavity.

The present methodology, on the other hand, works in a completely different way. The triangulated surfaces shown in Fig.10 were generated from a generic cascade data. An internal cavity was also added to the model in order to produce an approximation to the internal cooling system of a real blade. The coolant flow enters the cavity from the hub (blue arrow) and exits at the trailing edge slot (green arrows). The triangulated surfaces are therefore the starting point for the software. Once the input data is read, a user-defined uniform Cartesian mesh is generated and the cells which are intersected by the triangles are flagged as cut cells. The mesh is then refined and the process is repeated until the maximum number of refinements has been reached. Figures 11a and 12a show the fluid mesh for pitchwise and spanwise planes, respectively. The solid mesh is shown in Figs.11b and 12b. The fact that the software generates both fluid and solid mesh at once makes the approach very attractive for multi-disciplinary problems.

The crucial part for the topology editing or *sculpting* capability of the software lies on the Level Set Method (Osher and Sethian, 1988). Each cell has associated with it the signed distance to the nearest point on the body (the triangulated surfaces in Fig.10), known as a *distance field*. Boundaries are represented as the zero isosurface of the Level Set (green color in Figs.11c and 12c). The blue color in the figures represents the negative distance (solid cells) whereas the red color represents the positive distance (fluid cells). *Sculpting* means simply editing the distance field using a simple voxel-wise logic.

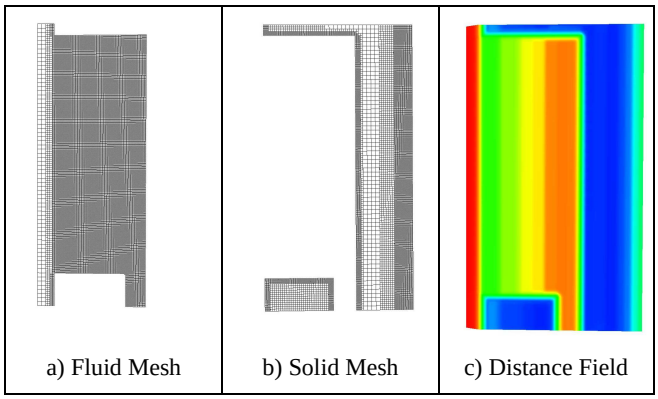


Fig.11 - Initial geometry (pitchwise plane).

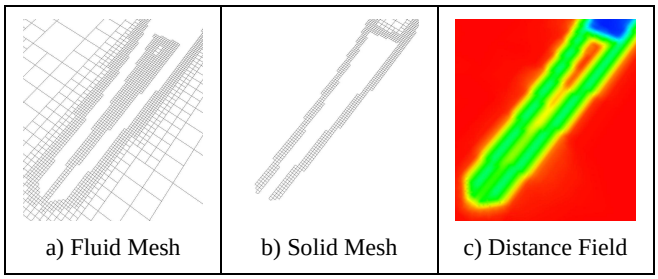


Fig.12 - Initial geometry (spanwise plane).

Integration with a design optimization system. The described software covers all steps in a CFD analysis, from geometry import to results visualization. In order to take full advantage of its powerful capabilities the software was integrated with a widely used high level scripting interface, *iSIGHT-FD 2.0*. One great advantage of this integration is that it provides the user unlimited flexibility and robustness for modifying the geometry according to the instructions from the optimizer. In practice, any possible manufacturing operation can be performed on the model in an analogous way to an actual *NC* machine. The other advantage is that the optimization process script in *iSIGHT-FD* can be easily added to a greater process within its framework. This would be very convenient for large corporations that have chosen to use such system for integrating the different disciplines involved in the design process.

A configuration file (Fig.13) is used as a means of defining the parameters for the editing tools, such as tool type, editing mode (remove or add material), tool radius, tool length and tool location. This file is read by both software packages and it is the main source of data exchange between them.

The high level scripting interface for the design optimization system is relatively easy to use. The methodology is similar to other

visual programming interfaces, such as Visual Basic, in which the user starts with an empty canvas and gradually populates it by dragging and dropping icons (or components). Figure 14 shows the script for conducting an optimization task. The icon named *DOE1* controls a design of experiments task. Double-clicking on the icon will open a window in which the user can choose the type of approach for *DOE*, the design variables, constraints and cost function. For the present test case, a fixed number of 15 cylindrical tools was chosen and the design variables were the spanwise location of each tool and its radius. The idea was to create an array of pin fins in the internal cavity by adding material to the mesh. Each cylindrical tool works as a piece of metal to be deposited in the cavity. The *DOE* algorithm generated hundreds of random combinations of the 30 design variables. The *Update ConfigTools* icon writes the new values of the design variables to the common configuration file for the software (*BoXeR* icon*)* to read. The software imports the triangulated surfaces, generates the mesh, calculates the distance field for the original geometry, reads in the editing tool data from the configuration file, calculates the distance field for each of the editing tools, combines the distance field from the original geometry and the editing tool, generates the edited mesh and exports encapsulated postscript files of screen shots captured during the process.

```
15          --> Number of tools
2           --> Tool 1: type = cylinder
-1          --> Tool 1: operation = add
-0.00169    --> Tool 1: x-coordinate
0.0175      --> Tool 1: y-coordinate
0.091666    --> Tool 1: z-coordinate
0.009666    --> Tool 1: radius
0.011154    --> Tool 1: length
0.0         --> Tool 1: angle 1
0.0         --> Tool 1: angle 2
0.0         --> Tool 1: angle 3
2           --> Tool 2: type = cylinder
-1          --> Tool 2: operation = add
-0.00169    --> Tool 2: x-coordinate
0.0175      --> Tool 2: y-coordinate
0.136666    --> Tool 2: z-coordinate
0.0060      --> Tool 2: radius
0.011154    --> Tool 2: length
0.0         --> Tool 2: angle 1
0.0         --> Tool 2: angle 2
0.0         --> Tool 2: angle 3
.
.
.
```

Fig.13 - Configuration file for defining the editing operations.

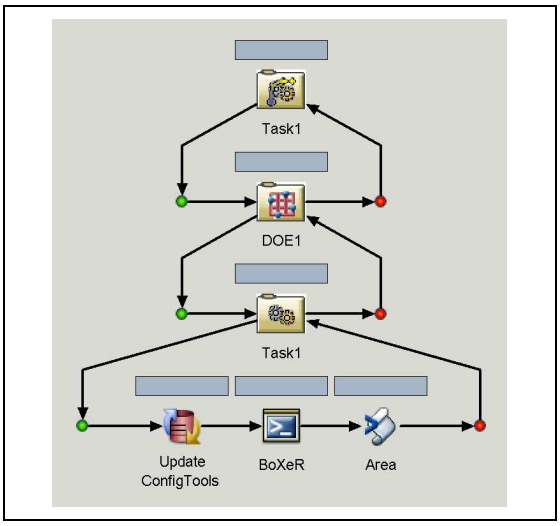


Fig.14 - The high level scripting interface (*iSIGHT-FD*).

The CPU time required for the execution of the CFD code for one configuration took in average 75s on a 64-bit workstation for a mesh of approximately 2.1 million cells. It must be mentioned that the process of rendering the plots and saving to file (purely for demonstration purposes) took 20% of the total CPU time.

Figure 15 presents some of the screen shots for the editing using a cylindrical tool. The top left figure shows the original triangulated surfaces in red and the cylindrical tools in blue. The top center figure shows the distance field. The blue region inside the green circles indicates the existence of solid cells. The top right figure shows the fluid cells after the edit. The proposed approach proves to be extremely flexible, allowing a full exploration of the design space. In the bottom part of the figure the distance field contour plot and the fluid and solid meshes are shown from a different viewpoint.

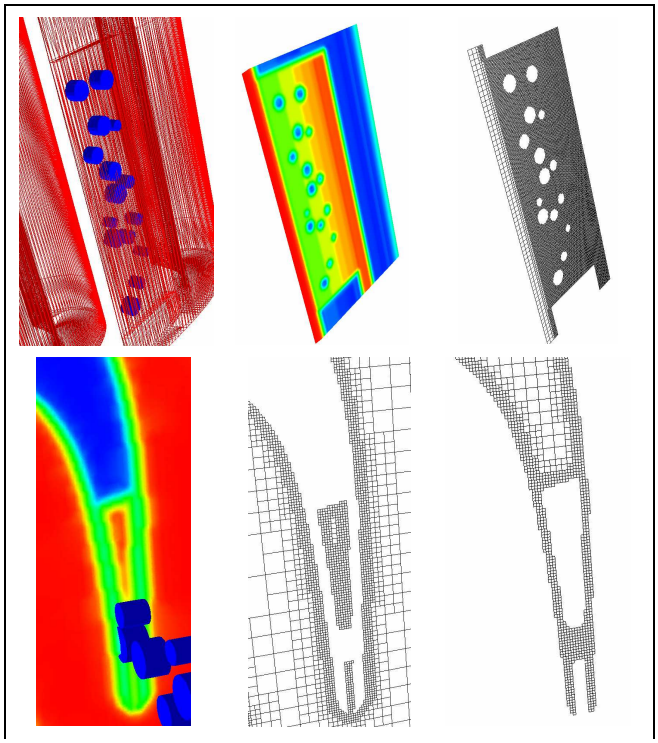


Fig.15 - Topology editing using a cylindrical tool.

Figure 16 shows similar test case results but using an arbitrary tool (profiled fin). The *STL* model for the tool was generated using two widely used software packages (*Gambit* and *TGrid*). The editing tool was easily imported into the software by adding an extra line to the configuration file pointing to the location and name of the file containing the triangulated surfaces for the customized tool. This is a very important feature in the software since the editing may also be regarded as a means to construct a complex geometry starting from a simple blade and adding all the other parts using the editing tool. For instance, in the case of unshrouded blades, the designer might be interested in investigating the effects of the squealer shape or type on the heat transfer in the tip region. In this case the input geometry for the software could be a triangulated surface representing the blade without the squealer and the editing tools could be the triangulated surfaces for different types of squealers. The design variables could be the squealer type and height.

The flow solution for the castellated fluid meshes shown in Figs.15 and 16 could be either run directly by the built-in standard solver, or a *ghost cell* (Viecelli, 1971) based solver or exported as a conformal mesh without the hanging nodes to a third party CFD solver, such as *Fluent*. The aim of the current paper, however, is to present the capabilities of the software to rapidly generate and edit meshes from an *STL* file with minimum user interaction. The

linkage of the solver or third party solvers into the process is currently ongoing and will be reported in future publications.

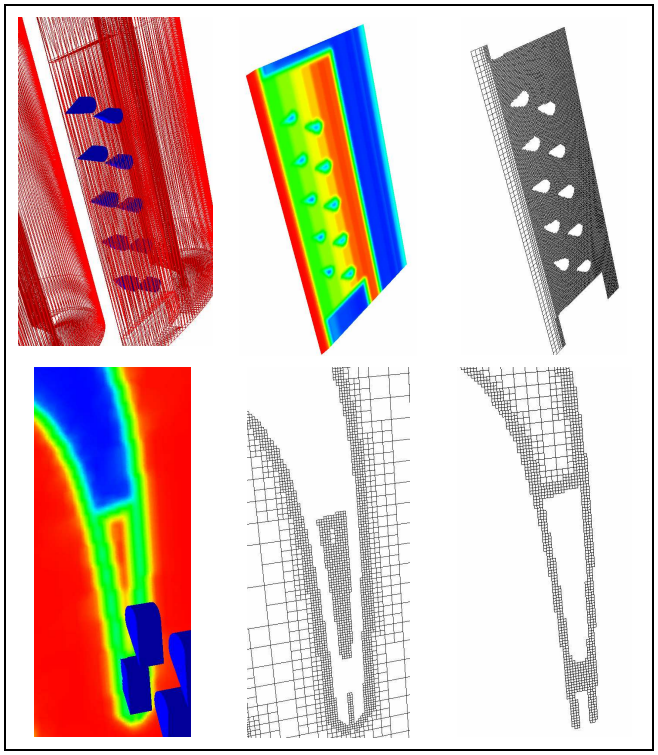


Fig.16 - Topology editing using an arbitrary tool.

CONCLUDING REMARKS

The development of two innovative computational tools for turbomachinery design was presented. Both software packages were designed to bypass the CAD-mesh bottleneck and permit meaningful automation. The successful integration of the second computational tool with a widely used process integration and design optimization system was demonstrated.

The mesh morphing tool described in the first part of the paper was designed for cases where systematic and moderate changes to a large computational mesh are desired. It is best used in the intermediate step of a design process assisted by CFD, where the initial mesh and result data are available and the analyst is satisfied with the quality of the solution. From that stage the designer would probably be wishing to evaluate the effect of a particular geometrical parameter on a certain figure-of-merit. In order to avoid the trouble of manually modifying the geometry at CAD level and going through the labour intensive task of regenerating the mesh, the analyst may instead setup the mesh morphing tool to make changes to the mesh automatically. The free-form deformation (*FFD*) approach was used as a means to morph the surface mesh. This technique allows the designer to survey a far wider design space than a restricted set of design parameters. The displacement of one control point of the *FFD* box allows the variation of several engineering parameters at once. The free-form deformation routines as well as the mesh quality optimizer are part of a library of functions that were used by the second computational tool.

The second computational tool, on the other hand, provides unlimited flexibility for editing the computational domain. The methodology is ideal for rapid prototyping during the early phases of the design process. It is designed for speed and for minimum user interaction during the mesh generation. The built-in automatic CAD clean-up tool eliminates the tedious task of fixing dirty geometries. The CAD-mesh process is handled internally by the software, restricting the user interaction to simply defining the computational domain extension and number of mesh refinement levels. The robustness of the software makes it a powerful tool for conducting truly automated design optimization. The integration of the software with a design optimization system was straightforward and the ease of use of both software packages is certainly encouraging for engineers willing to conduct complex real-world design optimization. The integration of the solver or a third party CFD solver into the design optimization framework is currently underway and results will be published in the near future. Exploring the natural conjugate format of the mesh generated by the software for multi-disciplinary problems is another task that the authors are planning for the near future. Some of the capabilities of the mesh morphing tool, such as free-form deformation for editing the triangulated surfaces and mesh quality optimization for the conformal mesh export routine, were also incorporated into this software.

ACKNOWLEDGEMENTS

The authors would like to thank Engineous Software Inc. for granting a free trial license for the development of the current project.

REFERENCES

Battina, J.T., 1990, “Unsteady Euler airfoil solutions using unstructured dynamic meshes”, *AIAA Journal*, 28(8):1381-1288.

Becz, S., Majewski, M.S., Langston, L.S., 2003, “Leading Edge Modification Effects on Turbine Cascade Endwall Losses”, *ASME Paper 2003-38898.*

Bucchieri, G., Galbiati, M., Coutandin, D., Zecchi, S., “Optimisation Techniques Applied to the Design of Gas Turbine Blades Cooling Systems”, *ASME Paper GT-2006-90771*.

Dawes, W.N., Harvey, S.A., Fellows, S., Favaretto, C.F., Velivelli, A., 2007, “Viscous Layer Meshes from Level Sets on Cartesian Meshes”, *AIAA-2007-0555*.

Dawes, W.N., 2006, “Towards a fully parallel integrated geometry kernel, mesh generator, flow solver & post-processor”, *AIAA-2006-0942*.

Dawes, W.N., 2005, “Building Blocks Towards VR-Based Flow Sculpting”, *AIAA-2005-1156*.

Dompierre, J., Labbé, P., Vallet, M.G., Camarero, R., “How to Subdivide Pyramids, Prisms and Hexahedra into Tetrahedra”, *8th Internation Meshing Roundtable*, California.

Engineous Software Inc., “iSIGHT-FD User’s Guide”, Version 2.0.

Fluent Inc., “Fluent User’s Guide”, Version 6.2, 2005.

Kellar, W.P., 2002, “Geometry Modelling in Computational Fluid Dynamics and Design Optimization”, PhD Dissertation, University of Cambridge.

Kulasekharan, N., Prasad, B.V.S.S.S., 2006, “Influence of Rib Turbulators on Pin-Fin Heat Transfer in the Trailing Region of Gas Turbine Vane - A Numerical Study”, *ASME-GT-2006-91124*.

Lethander, A.T., Thole, K.A., Zess, G., Wagner, J., 2003, “Optimizing the Vane-Endwall Junction to Reduce Adiabatic Temperatures in a Turbine Vane Passage”, *ASME Paper GT-2003-38939*.

Osher, S., Sethian, J.A., 1988, “Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations”, *Journal of Computational Physics*, 79, 12-.

Samareh, J.A., 2004, “Aerodynamic Shape Optimization Based on Free-form Deformation”, AIAA-2004-4630.

Viecelli, J.A, 1971, “A computational method for incompressible flows bounded by moving walls”, Journal of Computational Physics, 8, 119-.