13-17 January 2014, National Harbor, Maryland
52nd Aerospace Sciences Meeting

Practical and Reliable Mesh Generation for Complex, Real-world Geometries

A A J Demargne [1], **R O Evans**[2], **P J Tiller**[3]
Cambridge Flow Solutions Ltd, Compass House, Vision Park, Histon, Cambridge CB24 9AD, UK

and

W N Dawes [4]
Cambridge University Engineering Department, Cambridge UK

The following paper examines the key aspects of technology that deliver an advanced hybrid meshing capability that is operational today, and is ideally suited to complex, real-world geometries. The paper shows how particular benefits are derived from detailed algorithmic features at octree level, from the overall meshing procedure and from the software's client-server architecture and its distributed-memory parallel implementation. Two two complex geometry examples are used to demonstrate the benefits in terms of complex geometry capability, of ability to modify geometry and of and ease-of-use.

Nomenclature

ϕ = distance field derived around the surfaces defining the geometry

I. Introduction

The computational resources available today make it possible to obtain CAE simulations on models that are ever more realistic and complex. Such simulations are also very much in demand, in line with the fact that the next generation of technological improvements are generally to be found at ever smaller levels of detail and higher levels of realism. In addition, optimization is moving on from component to system, or at least sub-assembly, level. These trends further exacerbate the bottleneck that is obtaining a suitable discretization (mesh) to run simulations on appropriate fidelity level, real-world, complex geometries.

The focus of the following paper is to demonstrate in clear terms how an easy-to-use hybrid unstructured mesher delivers a reliable, practical CAE gridding capability that transforms industrial simulation cycles, irrespective of geometry complexity and size. In practical terms, the key attributes that make the technology relevant to industrial applications revolve around ease-of-use, reliability and speed. Obtaining a mesh on a challenging geometry should be a systematic process that is not specialized or manpower intensive and can be carried out at least as rapidly as any of the other simulation stages. Reliability and speed of the meshing process ensure that a mesh is generated on a predictable and short timescale. In a design context, this allows the user the opportunity to amend the design, and more broadly allows the simulation process to participate fully in the engineering design cycle.

The paper is structured in two parts. The first part explores the aspects of this specific meshing technology that deliver the tangible benefits and the key attributes mentioned above. Specifically, the paper highlights the aspects of the meshing strategy that confer reliability, robustness, speed and the ability to deform geometry through scripting within the meshing environment.

The second part of the paper focuses on illustrating the key steps in the meshing process through two specific example applications, highlighting clearly how these benefit from the meshing technology. A complete turbocharger is used to highlight the ability of the software to import high-complexity models, and to show the ease with which

[1] Business Development Manager, AIAA Member.
[2] Chief of Applications, AIAA Member.
[3] Senior Applications Engineer, AIAA Member.
[4] Francis Mond Professor of Aeronautical Engineering, CUED, AIAA Senior Member .

the meshing process is set-up and executed. The second example involves the meshing of an entire warship, with a helicopter landing on deck. This application illustrates the ability of the software to deal with a very wide range of lengthscales, and the ease with which geometry deformation can be scripted and implemented largely automatically, enabling the user to explore design spaces through parametric variations.

II. Meshing Technology

BOXERMesh is an unstructured hybrid mesh generator that discretizes geometry directly from CAD, using an Octree-based approach coupled to a distance field. The theoretical background of the technology has been reported previously (Dawes et al.[1,2,3]), and was inspired by early work by Adalsteinsson & Sethian[4], and Baerentzen[5] on volume sculpting with Level Sets.

The meshing process can be viewed as a "volume-to-surface" approach that avoids the prior generation of a surface mesh by "capturing" and "re-constructing" the geometry as the volume mesh is generated. In effect, the current approach focuses on the efficient generation of a runnable mesh, that most closely represents the body.

The underlying "philosophy" of BOXERMesh is generate a mesh that is always runnable, through a process that is entirely dependable and will take a predictable amount of time, rather than being open-ended or able to fail.

The meshing process can be broken down into three key stages: generating an octree mesh based on a distance field, body-morphing the octree mesh to capture the geometry, and growing viscous layers of selected surfaces.

A. Octree Meshing

The user first set-ups a bounding box that will contain the mesh, and defined a "seed" location within that box specifying which region of space on either side of geometry will hold the final mesh. The octree mesh stage begins once that is done.

Next, a distance field ϕ is derived. This defines a "standoff distance" from the body, but also the surface normal (**n**=grad(ϕ) and curvature (k=div(**n**)) (see Fig.1a).

The geometry is then immersed in the background octree grid constructed on the bounding box (and including all cell refinement specifications), and cells are sorted depending on whether they are in a seeded region or outside, or are cut by the geometry (see Fig.1b). At this stage the subdivision of the background cells already takes into account any refinement specified by the user.

The cut cells and those outside are discarded, producing an *integer*, conformal approximation to the true body surface, initialized from the set of exposed quad faces, shown in red in Figure 1c. The remaining set of cells forms the octree mesh (Fig. 1c). In a sense the octree mesh represents the nearest integer approximation of the geometry that is possible with the background grid size and refinements specified. These exposed quad faces represent

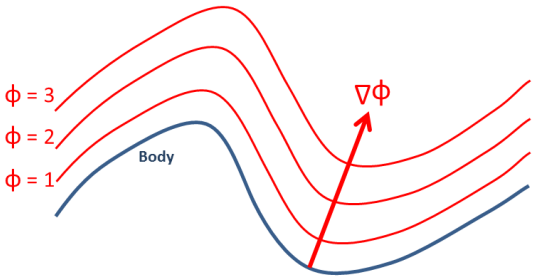

Figure 1a: distance field ϕ

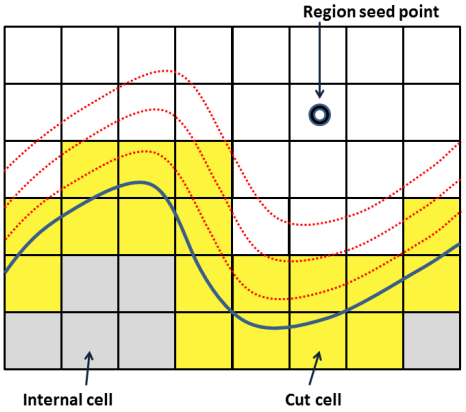

Figure 1b: embedding in background mesh

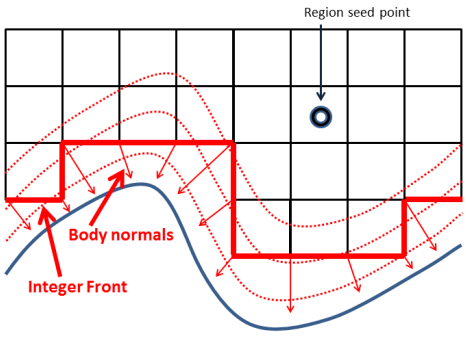

Figure 1c: Octree mesh and body normals

the “integer front”.

In practice, generating an octree mesh is a very fast process, typically lasting a few seconds. This rapidly gives the user a good initial idea of the degree to which the geometry and relevant features will be resolved in the final mesh.

B. Body morphing

The body morphing stage involves node displacements and insertion of hybrid cells (tetrahedral, prisms and pyramids) in a process called hybridisation. Existing nodes from the integer front are displaced onto the geometry following the body normals (shown in Fig. 1c). Those displacements that would lead to compromised cell quality metrics are not allowed to proceed, and new, hybrid elements are inserted between the front and the geometry (see Fig.2a). The emphasis is on producing cells of guaranteed quality, and the.process follows the underlying principle that displacements and insertions are done subject to the mesh always being *solvable*. In other words, the resulting mesh is conformal with the body – unless if to be so would produce an unsolvable mesh.

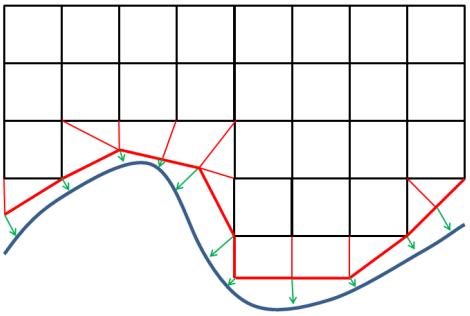

Figure 2a: Hybridisation

This might seem radical but in fact is simply a pragmatic acceptance that if a given mesh cannot support an acceptable flow solution then that mesh is worthless. This clearly places heavy emphasis on the mesh having adequate resolution to support local length scale features – if the mesh is too coarse then the present solvability principle will force the sub-mesh scale feature to be suppressed – this automatic de-featuring is better than a failed mesh or flow solve. This is a critical enabling step for any sort of simulation system which aims at performing design optimization on complex, engineering geometries.

An optimizer is used to finalise the node displacement and insertion of hybrid cells, until all nodes on the front are on the geometry surface (Fig.2b).

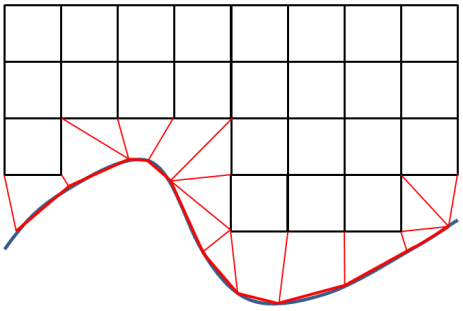

Figure 2b: Body-morphed mesh

C. Viscous layer extrusion

The layer extrusion process involves pushing back the body-morphed mesh by inserting prism elements templated on the surface nodes. The layers are built up one above the other, and the extrusion of each layer is only allowed to proceed as long as the cell quality metrics are met. Any local extrusion that compromises cell quality is halted, while adjacent extrusions that still meet the quality metric are allowed to proceed further.

The layers are specified simply through a first layer height, the number of layers requested and the expansion ratio from one layer to the other.

A final smoothing pass is applied to the “inviscid” part of the mesh once all layers are extruded.

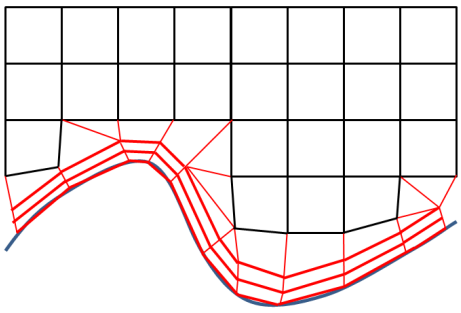

Figure 3a: Viscous layer extrusion

III. Benefits of the present approach

A number of significant benefits arise from the present approach. Some of these benefits stem from the overall meshing procedure, some from detailed choices at an algorithmic level and others simply from the way in which the software has been written and implemented.

A. CAD Tolerance and Geometry Deformation

The fact that the geometry is "captured" and "re-constructed" as the volume mesh is generated has a number of advantages. Specifically, these derive from the software's ability to "re-construct" edges and surface intersections, to the extent that it does not need to have these specified in the original CAD definition of the geometry. As a result, the software is very tolerant of "imperfect" CAD. Surface intersections which are not perfectly coincident (see Fig. 4) will be treated as properly joined as long as defects (e.g gaps) are smaller than the local octree cell size. In other words, the tolerance to CAD imperfections is essentially automatic.

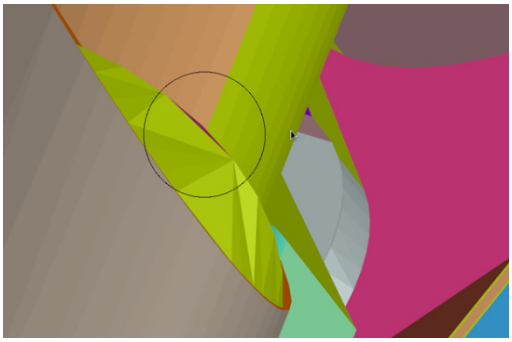

Figure 4: Typical CAD defect treated automatically by BOXERMesh

A second benefit of the treatment of surface intersections is that geometry can be deformed at input into the software, without having to revert to CAD. Again, this is because accurate geometric definitions of edges and surface intersections are not required, so that elements of the geometry (or entire objects) can be displaced one relative to the other and immediately remeshed in that new configuration. This actually delivers a very powerful geometry deformation capability within the meshing environment, that can be very easily scripted to explore parametric design spaces or perform design optimization when combined with a CFD sover, for example. This capability has been reported in Evans et al[6] and Dawes et al[7]. An example of geometry deformation capability is examined below (see section IV-B).

B. Thin Surfaces and Viscous Layers

A key difference between the present approach and other cut-cell octree methods lies in the decision to use an octree integer front that is positioned at some distance from the geometry rather than intersecting it. One of the advantages this brings is in the treatment of thin or zero-thickness surfaces, such as baffles (Fig. 5a). In those cases,

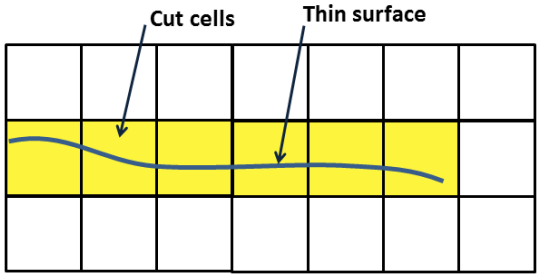

Figure 5a: Thin surface – cut cells

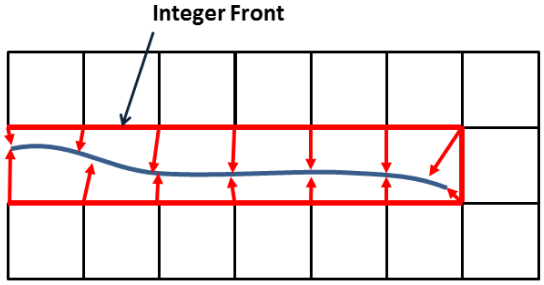

Figure 5b: Integer front nodes automatically morph to the correct side of the surface

the displacement of integer front nodes along distance field body normals guarantees that nodes will automatically find the correct surface to morph onto (Fig. 5b). This avoids the topological issues caused by nodes morphing to the wrong side of the thin or zero-thickness surface, and the manual work required to either pre-empt or fix those issues to achieve a usable mesh. The present approach guarantees that the treatment of such thin surfaces is essentially automatic, and most importantly that it leads to topologically valid, good quality cells. In turn, this contributes to the robustness, consistency and dependability of the software, which can be used perfectly reliably to generate meshes

in automatic (batch) mode, facilitating its integration into existing workflows and simulation processes, and permitting it's use in design optimization.

Another advantage of the integer front standoff distance is that the task of extruding viscous layers is made easier by the buffer space provided around the geometry. It becomes possible to grow layers further away from the surface, both because the underlying topology is more favourable and because the extrusion itself is less constrained. Furthermore, as the quality of the cells resulting from each step in the inflation process is checked prior to the extrusion itsef, layers can be grown into narrow gaps and surfaces in close proximity without cells colliding and being pushed into one another. This also leads to a layer growth that tapers down automatically to accommodate the reduced space available.

Viscous layers around a landing gear strut taken from the NASA / Gulfstream "Partially Dressed Closed Cavity – Nose Landing Gear" geometry submitted as one of the reference problems to the AIAA BANC-II workshop[8] (test case 4), is shown in Figure 6. The layers can be seen to extend significantly into the freestream, wrapping around features and tapering down into spaces with tight clearances.

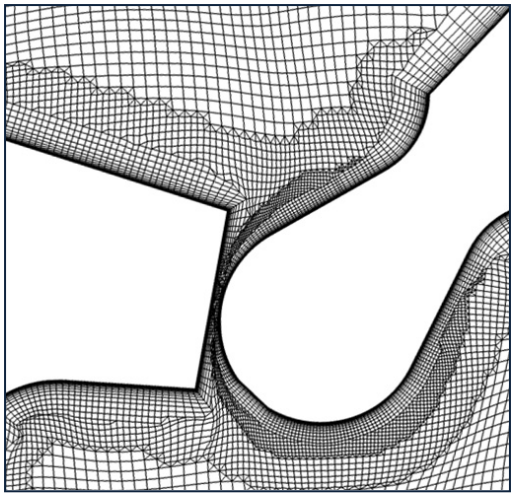

Figure 6: Viscous layers around a landing gear strut – automatically tapering into narrow gaps (detail)

C. Computational Flexibility and Speed from Parallelization

The software is structured following a simple client-server architecture: the server performs the core meshing task, driven by the client (GUI), or from the command line or as batch job. Significantly, from the onset, the software was written as distributed memory parallel code, which presents significant scalability advantages, particularly for large, high cell count meshes that have higher memory requirements. In those cases, the ability to distribute the memory requirement over the computational nodes, each node only needing its own small subset of the total memory, as opposed to having to pool the memory centrally for all the nodes to access, is a major benefit.

All stages of the meshing process described in section A are parallel, including the extrusion of viscous layers. This confers significant speed to the software. The following sections provide examples of complex geometry, multi-million cell meshes generated on the timescales of an hour or so.

Together, the architecture and parallelization allow the software to run on the widest range of hardware platforms, literally from a laptop to an HPC cluster to the cloud, whilst delivering excellent performance scalability. Users have demonstrated performance retention in meshes exceeding 1Bn cells.

IV. Example Applications

The second part of this paper focuses on illustrating some of the characteristics and benefits of the current approach through two specific example applications. The first example is a complete turbocharger assembly that presents complex geometrical domains and features. The second example involves the meshing of an entire warship, with a helicopter landing on the rear deck in the ship's air wake.

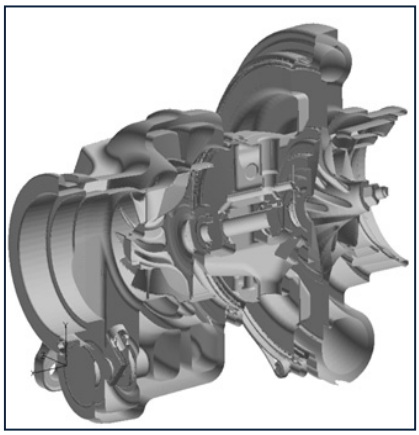

Figure 7: Complete turbocharger – section through CAD model

A. Complete Turbocharger

The geometry in this example is a complete turbocharger, used by courtesy of The Holset Engineering Co. This particular design is typically used in mid-range diesel engine applications, from 90kW to 240kW, and is found in light trucks, minivans and larger pick-up trucks. The turbocharger is a variable geometry design, using a sliding side wall to regulate the area at turbine entry. The system is oil-lubricated and water

cooled. Typical operating speeds are in the region of 150,000rpm, and maximum boost is around 4atm.

The CAD defining the geometry is of "manufacturing" grade, in other words it includes all the components and dimensioning required for assembly. In a CFD analysis context this generally means that there is excessive information and that the user will have to spend time defeaturing the CAD to retain only the information and detail that is needed for the CFD analysis. In the present approach, the tolerance to CAD and ability to handle arbitrary complexity mean that the CAD cleanup and defeaturing stage is not required, and the CAD is imported directly into BOXERMesh.

The turbocharger consists of separate compressor (atmospheric air) and turbine (exhaust gas) domains, as well as lubricating oil andcooling water cavities. The geometry is inserted into a background octree mesh via a bounding box and each of these 4 domains is seeded and meshed separately. The mesh set-up stage took of order 20 minutes. The compressor domain ran on 48 processors and took approximately 15 minutes to mesh (in blue in Fig.8a). The larger turbine domain, which includes vanes, took approximately 40 minutes to mesh (in red in Fig. 8a).

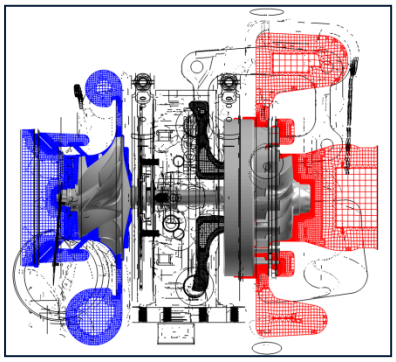

Figure 8a: Compressore (blue) and turbine (red) domains

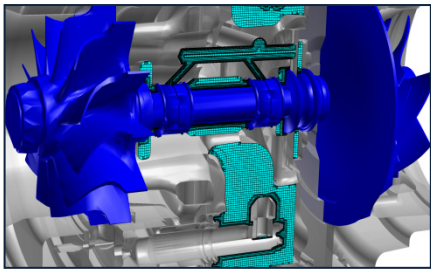

Figure 8b: Mesh on a section through the oil cavity domain

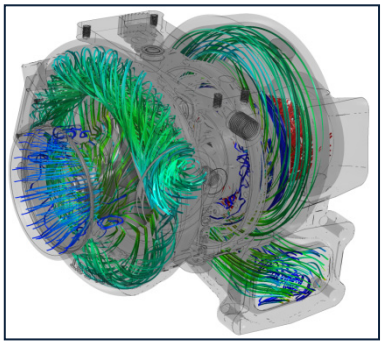

Figure 8c: Flow solution on the compressor and turbine domains - streamlines coloured by Mach no.

The ability to extract automatically complex domains is illustrated in Fig. 8b, showing a section through the meshed oil lubrication cavity. The resulting meshes are immediately runnable (Fig. 8c) and do not need repair. They can be solved separately or in a conjugate mode. Geometry not needed to define the selected domain is simply and automatically discarded without user input.

More generally, the point made here is that the present approach is ideally suited to the meshing of highly complex geometries, and is sufficiently easy-to-use for the meshing to be done with minimal user intervention. Furthermore, the ability to import and use highly-detailed CAD directly, as well as the fact that the user avoids the mesh repair stage altogether, combine to deliver a considerable gain in time and productivity.

B. Warship with Helicopter Landing on Rear Deck

The original geometry for this example was obtained from Trimble 3D Warehouse[9], and is publicly available for use. The geometry is based on an Arleigh Burke-class destroyer, and includes both the ship and the helicopter on the deck. Both are very well detailed, retaining features such as individual railings, antennae, and individual windshield wipers on the helicopter (Fig. 9).

Geometry is imported into BOXERMesh retaining the full feature tree. This enables important downstream productivity benefits by allowing full boundary condition associativity. Another advantage, exploited here, is that individual elements of geometry can be identified and a variety of spatial transformations applied to these elements, rapidly generating a new configuration. Commonly used transformation include translations, rotations, and scalings, applied to all or part of a geometry. The transformations are applied through simple

Figure 9: CAD geometry of the ship and helicopter

scripting, making use of a utility that reads in, executes and outputs BOXERMesh files. As mentioned in section III-A, deformation of geometries within the meshing environment is made possible by the way in which the present approach treats surface intersections.

In this example the helicopter is brought to land on the deck of the ship, by modifying its position sequentially through four different stages involving translations and rotations. In the last stage, the oleo on the tail landing gear is also compressed, to illustrate the ease with which specific parts of a geometry can be deformed.

The meshing process for this example is illustrated as a flow diagram in Fig. 10. The process is driven through a Python script. The script reads in the original CAD parts and then defines the desired displacements. The script then loops over the sequence of positions, and for each one displaces the helicopter and then imports the resulting transformed geometry and the unchanged ship geometry into the mesher, where these are combined to form the model. BOXERMesh's internal solid geometry model allows the new simulation model to be assembled via simple Boolean summation with geometry intersections, collisions and gap closures handled automatically.

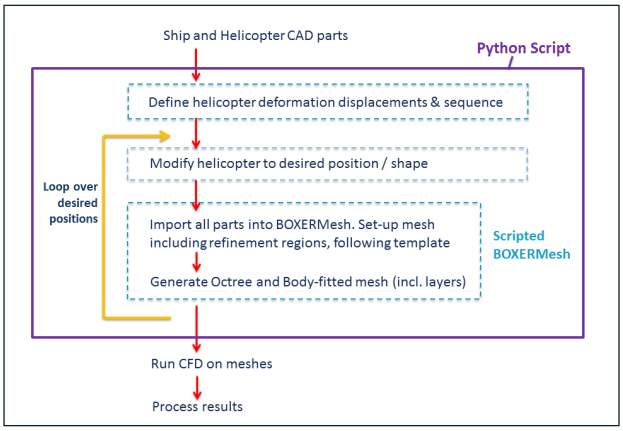

Figure 10: Process flow diagram - ranging over sequence of helicopter positions relative to the ship

The Python script invokes a pre-defined template file that is used to set-up the meshing parameters, and guide the actual meshing run. The resulting mesh is saved and the script iterates to the next position until the desired sequence is completed. The output from the script is a series of runnable meshes, together with a log of operations, including statistics and timings.

The key point here is that all these modifications to the geometry are carried out within what can be described as the meshing environment, without reverting to CAD in any way. This is of very practical benefit to the user, as it delivers the capability to explore design variants and spaces in an automated way, and lends itself naturally to integration into broader design optimization processes.

Setting the process up including initial import of CAD, initial BOXERMesh run to generate the meshing set-up template, and preparing the Python script took about 1 day. This again highlights the ease with which such an intricate geometry can be set-up and meshed.

The cell count for these meshes remained effectively constant at 46 Million cells, with 28.5 Million nodes. Typical meshing times were about 100 seconds for the octree mesh and 72 minutes for the body-fitted mesh, running on 48 processors. This is actually noticeably slower than the average speeds achieved on most geometries, which is typically close to 0.02 Mcells/min/core. The reason for this slight increase in meshing time is that this case has a particurlarly large number of very small features which are detected by the software, and then resolved automatically through additional "feature capture" iterations, which take longer. In this example, small features include all railings and antennae on the ship, of which there are many (see Fig.11). This point provides a clear illustration of how actual software performance (speed) on complex geometries can be case specific, influenced by detailed features and range of lengthscales, as well as evidently by user inputs such as refinements and the number of layers. It also highlights the software's inherent ability to deal with a very wide range of lengthscales – close to 10^5 in this example. Other meshes

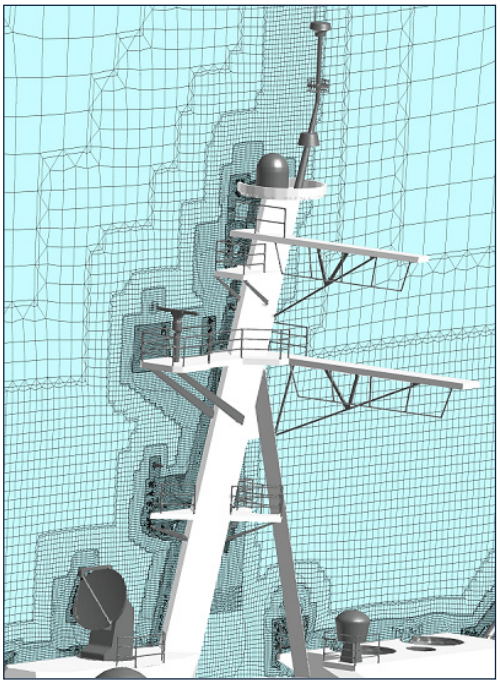

Figure 11: Detail of the mesh around the ship mast and antennae

have been generated on models with ranges in excess of 10^8. The software is capable of working with lengthscale ranges of 10^{11} . Pictures of the resulting meshes for each of the four helicopter position are shown in Fig. 12.

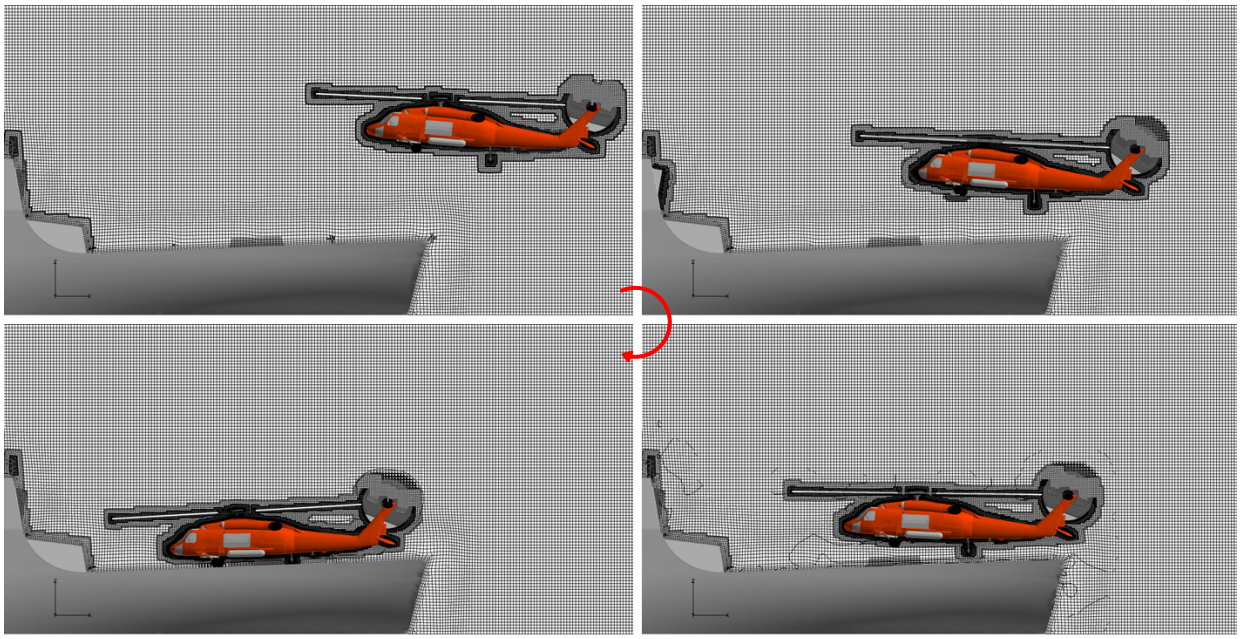

Figure 12: Scripted displacement sequence – landing the helicopter on the ship's deck

As mentioned, the resulting meshes are ready to be used, and no "repair" activity is required by the user. Side views of the mesh and of a CFD solution on the first helicopter position are shown in Fig. 13, with the ship sailing into the wind at 14 knots. The boundary conditions used are approximate, and verify that the meshes are indeed runnable, rather than being physically representative and accurate. The CFD was run using Fluent 14.5.0 and showed good convergence.

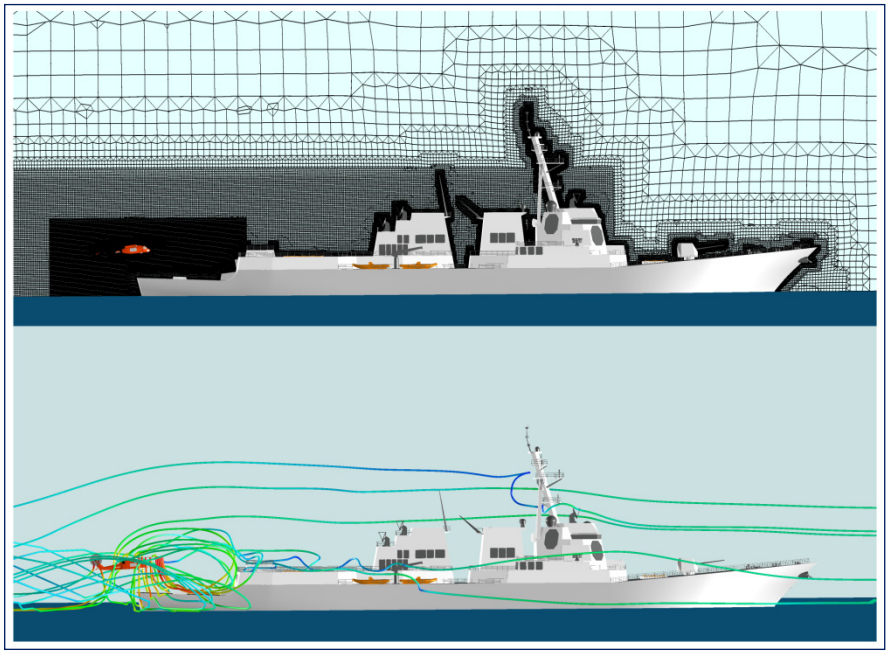

Figure 13: Side view of the mesh and a flow solution

V. Conclusion

The objective of this paper is to highlight key aspects of technology that deliver an advanced hybrid meshing capability that is ideally suited to complex, real-world geometries. The paper shows how particular benefits are derived from detailed algorithmic features, from the overall meshing procedure and from the way the software is written and implemented. These benefits are demonstrated on two complex geometry examples.

The specific treatment of surfaces and geometry delivers both a tolerance to CAD imperfections and the ability to deform geometry within the meshing environment. The use of a distance field coupled to an integer octree front that stands-off from the geometry allows the software to deal automatically with thin or zero-thickness surfaces. Importantly, this ensures that the meshing process has control of cell quality, producing meshes that are immeadiately usable, and underpining the robustness and reliability of the entire meshing process. It also contributes to delivering a very comprehensive and powerful viscous layers extrusion capability.

The present approach is implemented as distributed-memory parallel software, which gives it excellent scalability and therefore the ability to tackle very large meshes. It also gives it speed, with meshes generated in a matter of hours and minutes. An added advantage is the use of a client-server architecture, which allows the software to run equally well on hardware platforms ranging from laptops to HPC clusters to the cloud.

More generally, the paper demonstrates that through these key features and techniques, the present approach delivers a robust, reliable and easy-to-use meshing capability that tackles geometries of arbitrary complexity and size, in a truly parallelized and scalable way. This high-performance simulation capability is in operation now.

References

[1]Dawes WN "Building Blocks Towards VR-Based Flow Sculpting" 43rd AIAA Aerospace Sciences Meeting & Exhibit, 10-13 January 2005, Reno, NV, AIAA-2005-1156

[2]Dawes WN, Kellar WP, Harvey SA "Viscous Layer Meshes from Level Sets on Cartesian Meshes" 45th AIAA Aerospace Sciences Meeting & Exhibit, 8-11 January 2007, Reno, NV, AIAA-2007-0555

[3]Dawes WN, Kellar WP, Harvey SA "A practical demonstration of scalable parallel mesh generation" 47th AIAA Aerospace Sciences Meeting & Exhibit, 9-12 January 2009, Orlando, FL, AIAA-2009-0981

[4]Adalsteinsson D & Sethian JA, "A level set approach to a unified model for etching, deposition & lithography II: three dimensional simulations" J.Comput.Phys, 122, pp348-366, 1995

[5]Baerentzen A, "Volume sculpting: intuitive, interactive 3D shape modelling" IMM, May 2001

[6]Evans RO, Dawes WN, Zhang Q "Application of Design of Experiment to a Gas Turbine Cascade Test Cell" ASME Turbo Expo 2013: Power for Land, Sea and Air, June 3-7, 2013, San Antonio, TX, ASME GT2013-94314

[7]Dawes WN, Kellar WP, Harvey SA "Towards topology-free optimisation: an application to turbine internal cooling geometries" 46th AIAA Aerospace Sciences Meeting & Exhibit, 7-10 January 2008, Reno, NV, AIAA-2008-925

[8]Second Workshop on Benchmark problems for Airframe Noise Computations (BANC – II), AIAA, Aeroacoustics and Fluid Dynamics Technical Committees, June 7-8, 2012, Colorado Springs, CO

[9]Trimble 3D Warehouse; http//:http://sketchup.google.com/3dwarehouse/