47th AIAA Aerospace Sciences Meeting & Exhibit, 5-8 January 2009, Orlando FL

AIAA-2009-0981

A practical demonstration of scalable, parallel mesh generation

WN Dawes,*
CFD Laboratory, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK.
and
SA Harvey, S Fellows, N.Eccles, D Jaeggi & WP Kellar,
Cambridge Flow Solutions Ltd, Compass House, Vision Park, Cambridge, CB4 9AD, UK

Real-world simulation challenges are getting bigger: virtual aero-engines with multistage blade rows coupled with their secondary air systems & with fully featured geometry; environmental flows at meta-scales over resolved cities; synthetic battlefields. It is clear that the future of simulation is scalable, end-to-end parallelism. To address these challenges we have reported in a sequence of papers a series of inherently parallel building blocks based on the integration of a Level Set based geometry kernel with an octree-based cut-Cartesian mesh generator, RANS flow solver, post-processing and geometry management & editing. The cut-cells which characterize the approach are eliminated by exporting a body-conformal mesh driven by the underpinning Level Set and managed by mesh quality optimization algorithms; this permits third party flow solvers to be deployed. This paper continues this sequence by reporting & demonstrating two main novelties: variable depth volume mesh refinement enabling variable surface mesh refinement and a radical rework of the mesh generation into a bottom-up system based on Space Filling Curves. Also reported are the associated extensions to body-conformal mesh export. Everything is implemented in a scalable, parallel manner. As a practical demonstration, meshes of guaranteed quality are generated for a fully resolved, generic aircraft carrier geometry, a cooled disc brake assembly and a B747 in landing configuration.

I. Introduction

As demand for flow simulations increases, so does application to ever more challenging cases: virtual aero-engines with multistage blade rows coupled with their secondary air systems & with fully featured geometry; environmental flows at meta-scales over resolved cities; synthetic battlefields. These simulations are characterized both by real, complex geometries and also by scale – not just physical scales (which may be widely disparate) but also by scale of mesh resolution needed to support realistic modeling like LES. It is clear that in the future, simulation must employ end-to-end parallelism – from the geometry kernel through the mesh generation and onto the solver/post-processor. This parallelism must be scalable and built on data structures and software architecture paradigms capable of dynamic load balancing.

We have explored one possible way forward and reported our experiences in a sequence of papers. Our chosen methodology was deliberately different from the current, orthodox CFD process chain. The essence of this new approach was the integration of a geometry kernel based on a Level Set approach with an octree-based cut-Cartesian mesh generator, RANS flow solver and post-processing all within a single piece of software. The basic building-block work was reported by Dawes [2005]; the potential to parallelize the entire system was reported in Dawes

* wnd@eng.cam.ac.uk

[2006] and illustrated with prototype versions of our software, *BOXER*. Replacing the cut-cells with body-conformal meshes was described in Dawes et al [2007]; in this work the underpinning Level Set was used with optimization algorithms based on mesh quality metrics to enable the export of meshes with guaranteed quality to drive third party solvers like FLUENT®.

This paper reports recent progress towards our overall goals. The main novelties reported are: first, a generalization of the earlier work to permit variable depth octree refinement to enable variable surface refinement; second, a radical rework of the earlier parallel mesh generation from a simple top-down octree to a bottom-up octree based on Morton coding and Space Filling Curves. Also reported are the associated extensions to permit smooth surface reconstruction from underlying Level Set to allow body-conformal mesh export – with no hanging nodes. All of this is implemented in a scalable manner within a robust C++ architecture.

As a practical demonstration, meshes of guaranteed quality are generated for a fully resolved generic aircraft carrier geometry, a cooled disc brake assembly and a B747 in full landing configuration.

II. Parallel, bottom-up octree mesh generation

In earlier work, reported in Dawes et al [2006] mesh generation based on a very simple top-down octree was described. This starts with a single master cell which then divides in response to the geometry it contains until the final mesh is produced (see Figure 1). This is easy to code and was parallelized using simple coordinate-axis based load balancing; promising early results were obtained & reported.

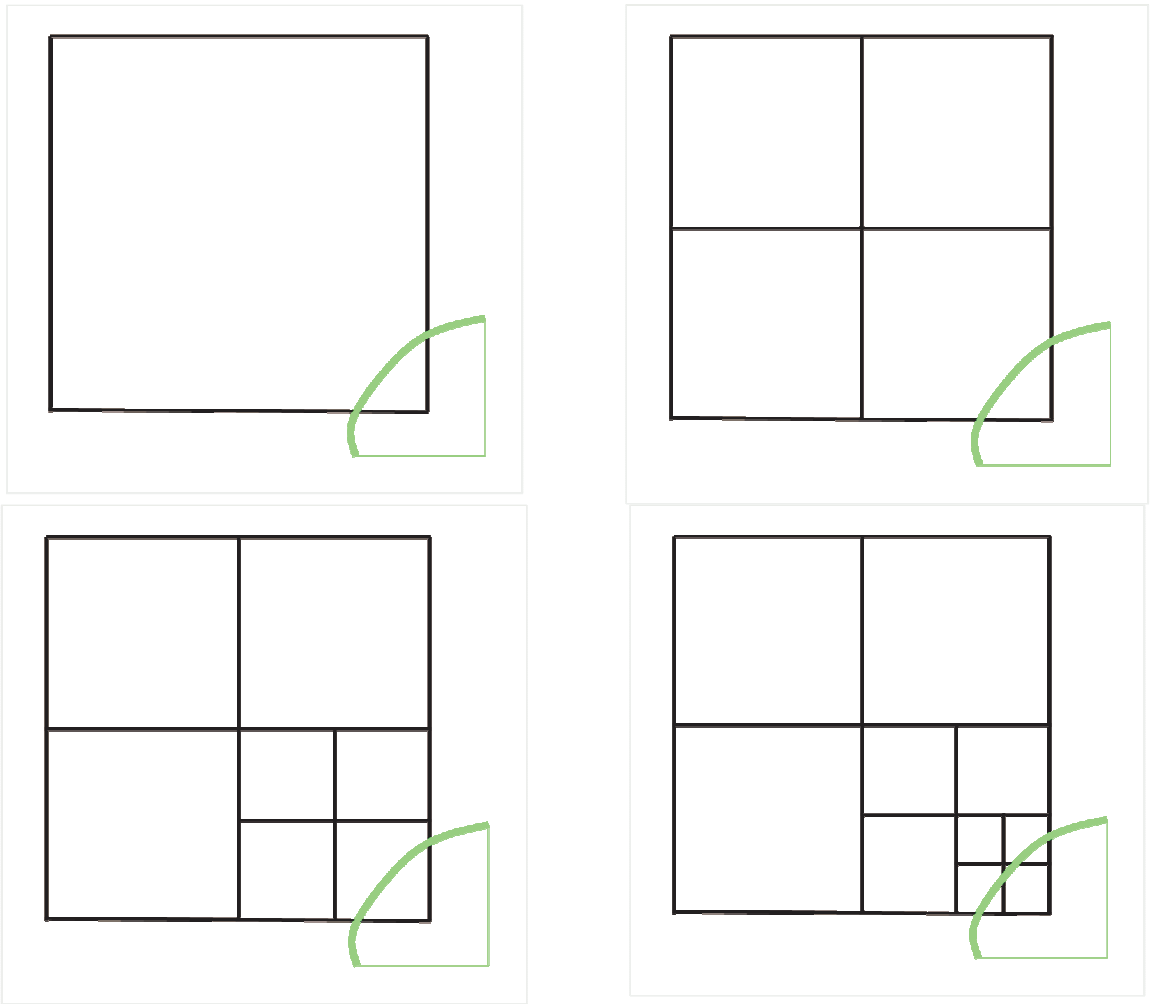

Figure 1: Simple *top-down* octree capturing geometry by division (from top left to bottom right).

However, as this work developed in application to arbitrary geometries it proved difficult to ensure satisfactory load-balancing – which must be done on-the-fly - as the mesh is generated. A better way is to invert the process and generate the mesh from the bottom-up – from the finest cells up the tree to the coarser ones. This sort of approach is less common and typically is based on Space Filling Curves and Morton coding (see for example Tu et al [2007]). However, it is much easier to dynamically load balance this approach and hence achieve parallel scalability.

This approach is summarized in Figure 2 which shows the tree structure associated with a simple quadtree mesh (the three dimensional equivalent octree proceeds in exactly the same way). Each cell is divided into four with *direction codes* 00, 01, 10 & 11. Top-down concatenation of these directional codes leads to the *locational code* of each cell in the mesh. An exactly equivalent but much more convenient method to derive these locational codes is to use Morton coding (Tu et al [2007], Samet [1990], Sagan [1994]) which consists first of expressing the cell coordinates in integer form based on the finest refinement level. Then these integer coordinates are written in binary and interleaved bit by bit. As Figure 2 illustrates, this reproduces the concatenated direction codes.

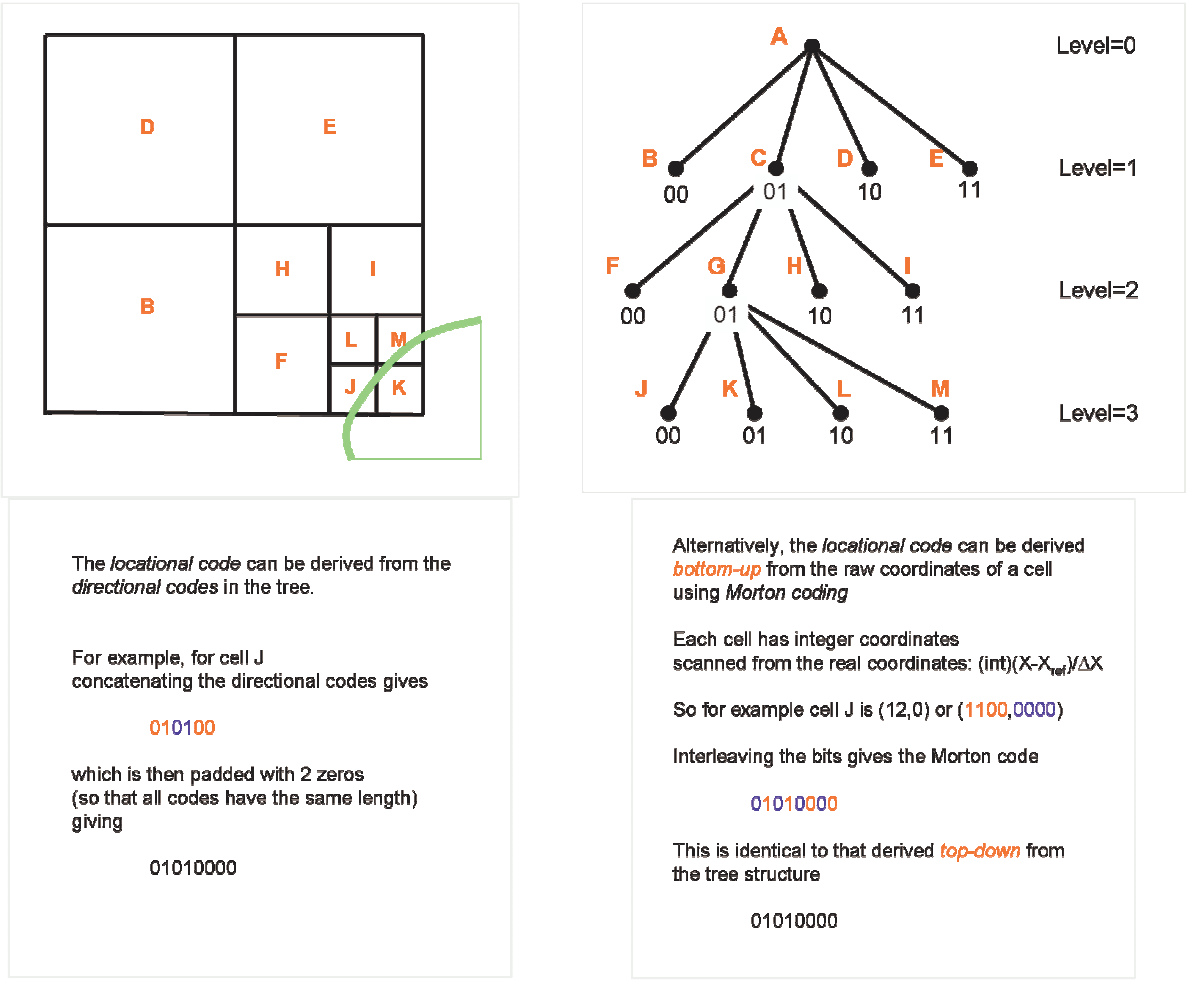

Figure 2: The tree structure with top-down directional codes leading to locational codes contrasted with locational codes derived directly from cell spatial coordinates using Morton coding.

This approach is *much* more convenient because it is *bottom up* and does *not* need the whole tree structure to be known – or even exist. In fact in our parallel implementation the whole mesh and associated tree only exist implicitly within each node of the PC cluster which holds only the local mesh and tree fragment. Thus, as illustrated in Figure 3, mesh generation proceeds in exactly the opposite direction to that shown earlier in Figure 1.

First the geometry is "scanned" into voxel cell form with scale corresponding to the lowest refinement level. Next appropriate siblings are added; then cells of the same or higher level are added by agglomeration moving outwards from the body. This is repeated recursively until the entire domain is covered. The agglomeration must preserve the h:2h transition rule – but this is trivial to manage on-the-fly.

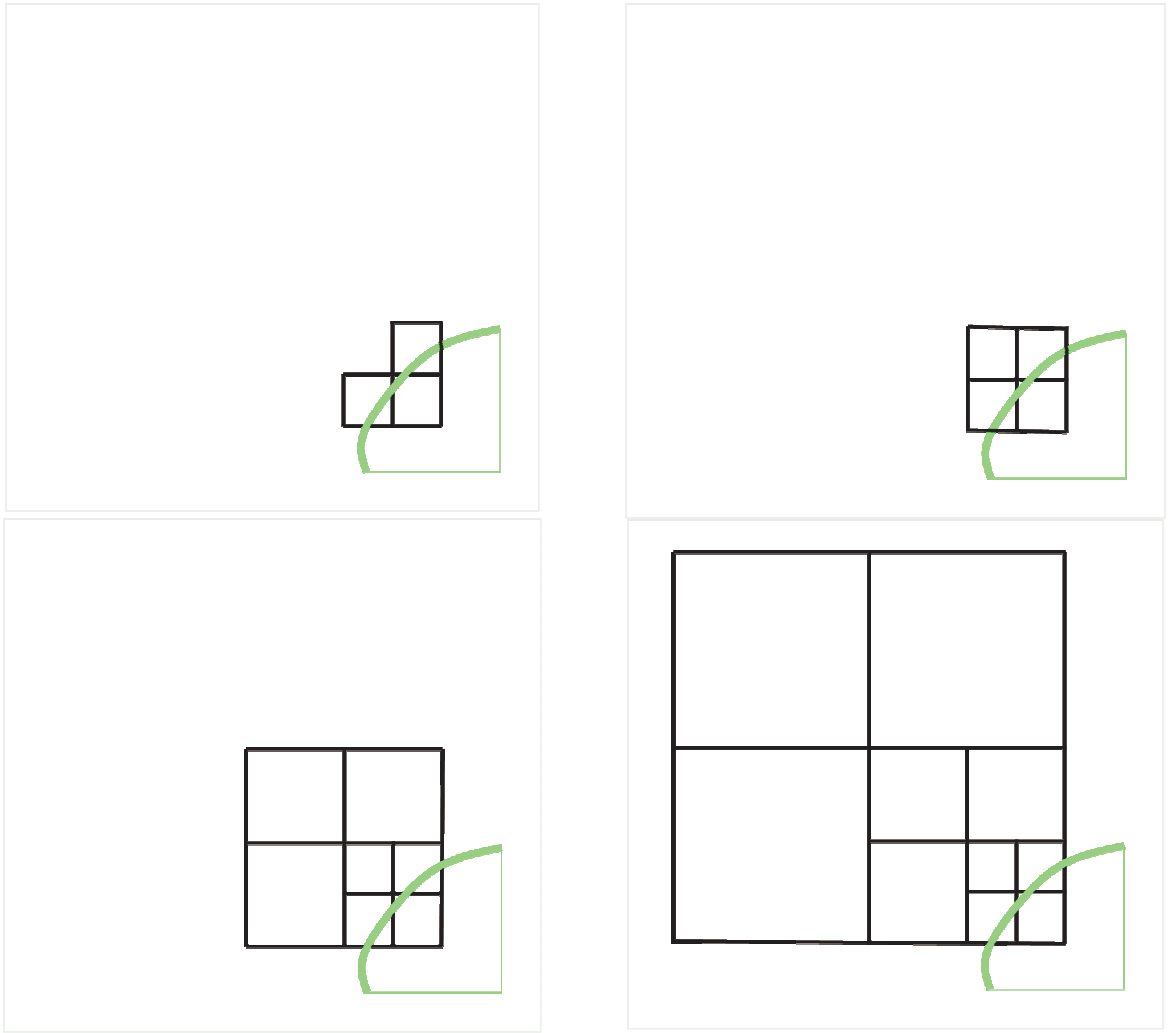

Figure 3: More useful *bottom-up* octree capturing geometry by agglomeration (from top left to bottom right).

In tandem with this is an implicit, bottom-up generation of the tree structure extracting direction codes from the Morton codes as illustrated in Figure 4. Each node of the PC cluster only holds – and only needs to hold - sufficient local tree data to find parents, children and neighbours/siblings.

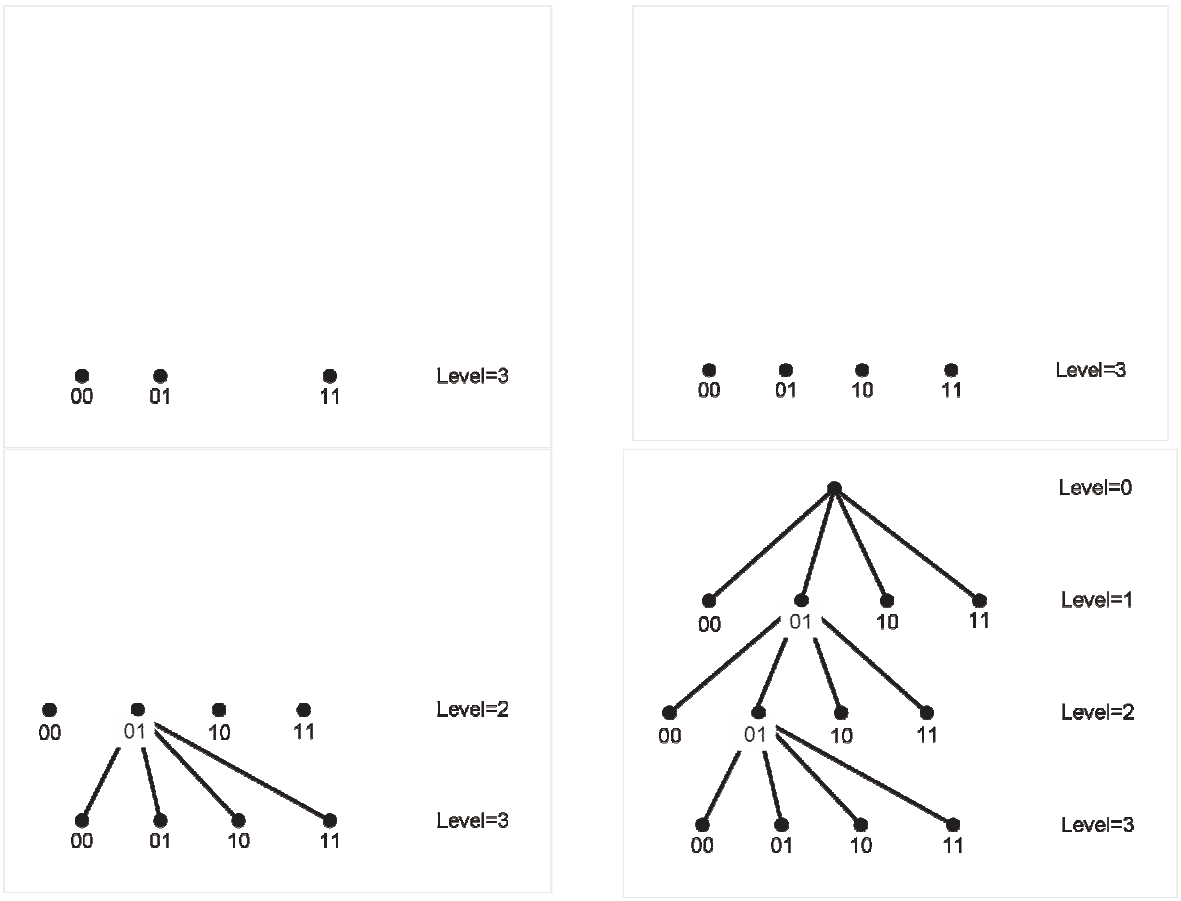

Figure 4: Implicit, bottom-up generation of the tree structure (from top left to bottom right).

Key to a successful parallel implementation is the ability to load balance on-the-fly as the mesh is created. We achieve this using the Space Filling Curve (SFC) associated with the Morton coding (Samet [1990], Sagan [1994]) and using ideas presented by Aftosmis at al [2004]. As Figure 5 illustrates, the direction codes 00, 01, 10 & 11 have an associated curve – a SFC called the Morton N-curve. When applied to the simple mesh (top right of Figure 5) cells B to M are connected in a particular way, sorted by ascending locational code – this is the SFC of this mesh. Storing the mesh this way is very efficient and also local since the associated tree is only present implicitly.

Simple but effective domain partitioning can be performed by dividing this SFC amongst the available PC nodes. As mesh generation proceeds and the SFC is filled with ever more entries, dynamic load balancing can easily take place by moving cells appropriately from up or down the SFC into neighbouring PC nodes. Tree data, needed for parents/siblings/children/neighbours, can be derived naturally & efficiently from the locational codes in the SFC.

Finally, at the same time as revising *BOXER*'s fundamental data structures we took the opportunity to allow variable depth refinement – and hence variable surface mesh scales – and accordingly had to extend the surface reconstruction for the body-conformal export (see the description in Dawes et al [2007] of this reconstruction for uniformly refined mesh depth). Variable depth refinement is implemented by developing a mesh to a chosen refinement floor, representing this by an implicit tree and then refining below that floor if desired – for example driven by body geometry curvature.

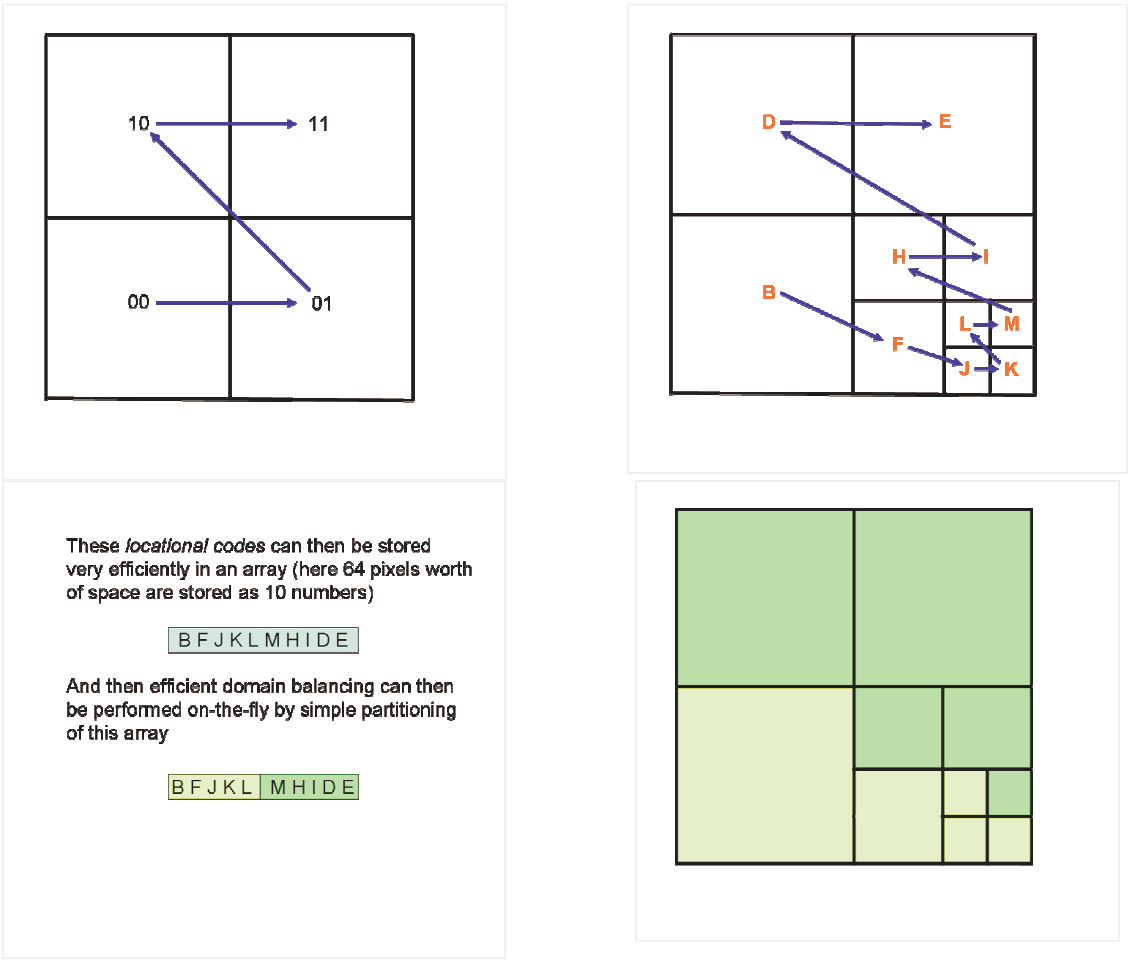

Figure 5: The Space Filling Curve associated with Morton coding and the associated domain partitioning.

III. Sample results

A generic aircraft carrier

For sample results first our software, *BOXER*, was applied to a fully featured generic aircraft carrier geometry. The resulting 25M cell mesh took approximately 15 minutes to generate from STL import to body-conformal mesh export in a format suitable for the FLUENT® flow solver. Around 8 of the 15 minutes was I/O and about 5 minutes was spent optimizing the quality of the exported mesh. A master node plus 7 server nodes were used with about 28Gb total RAM used at peak.

Figure 6 shows an overview of the generic aircraft carrier with a vertical mesh slice through the body-conformal export from the underlying octree mesh showing the octree hanging nodes replaced by hex/pyramid/tet cells to allow flow solution via third-party codes; the colouring is by PC processor number. Figure 6 also shows exported body-conformal meshes showing the surface smoothness & again rendered by PC processor number. Finally the Figure shows the body-conformal surface meshes themselves displaying variable refinement levels; a detail view near the bow shows the benefit of variable depth refinement – as well as the hull scale, the anchor itself is resolved.

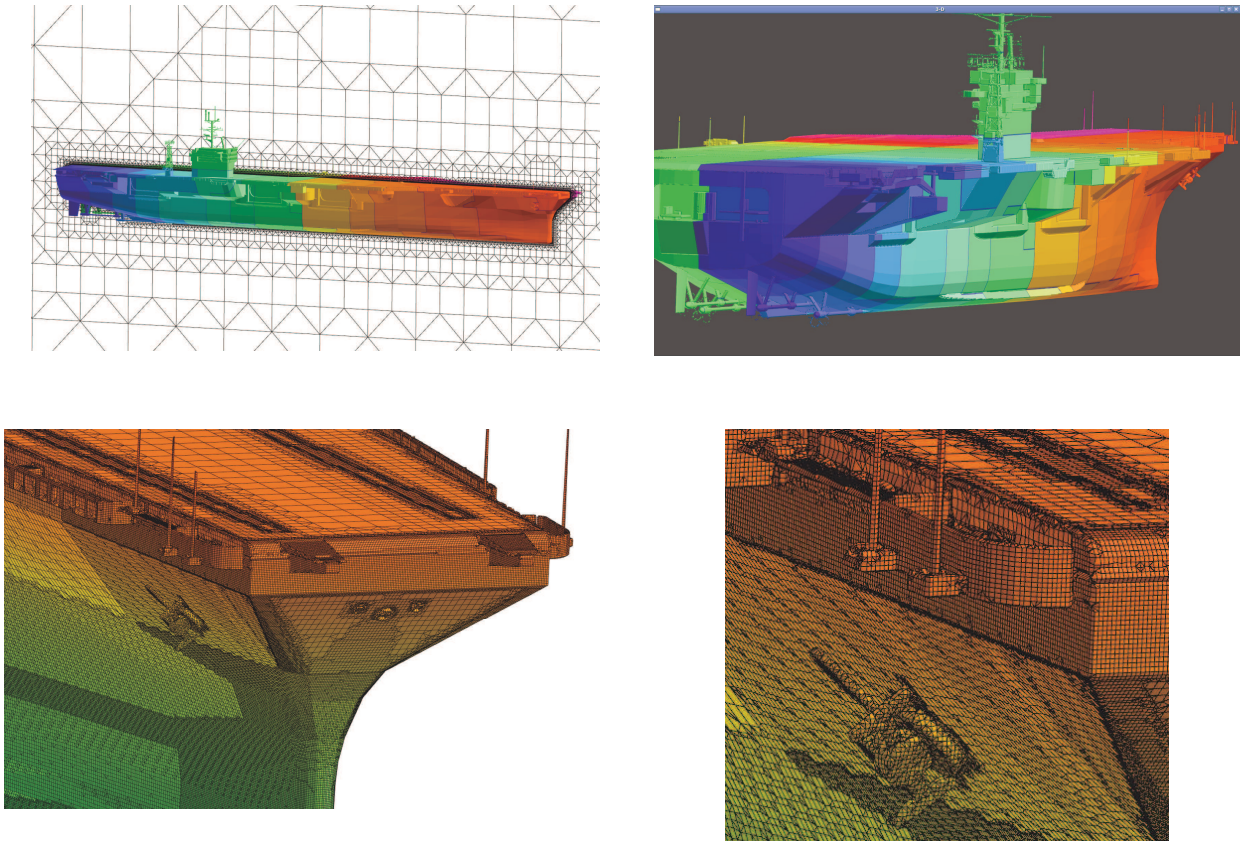

Figure 6: Overview of the generic carrier with a vertical mesh slice through the body-conformal export with the octree hanging nodes replaced by hex/pyramid/tet cells; the colouring is by PC processor number (top left); exported body-conformal meshes showing the surface smoothness & rendered by PC processor number (top right); (bottom row) body-conformal surface meshes showing variable refinement levels; detail view near the bow showing the resolution of the anchor.

A cooled disc brake assembly

The next example is a generic, cooled disc brake assembly; this is used simply for illustration – in practice the entire associated open-wheeled racing car would be meshed also. The assembly was exported from a UG solid in STL format, imported direct into *BOXER* and then exported as a body conformal mesh with hanging nodes removed.

Figure 7 shows (top left) the smooth exported surface of the assembly with (top right) an overview of the mesh. The bottom row shows detail views displaying the curvature sensitive variable depth refinement in action – in particular the disc edges and the cooling hole perimeters are resolved with a refinement several levels below the background floor. Overall the mesh contained 18.2M cells and took only 6min45 on 48 processors (including 1min30 for disc writing and general NFS slowness) from start to finish.

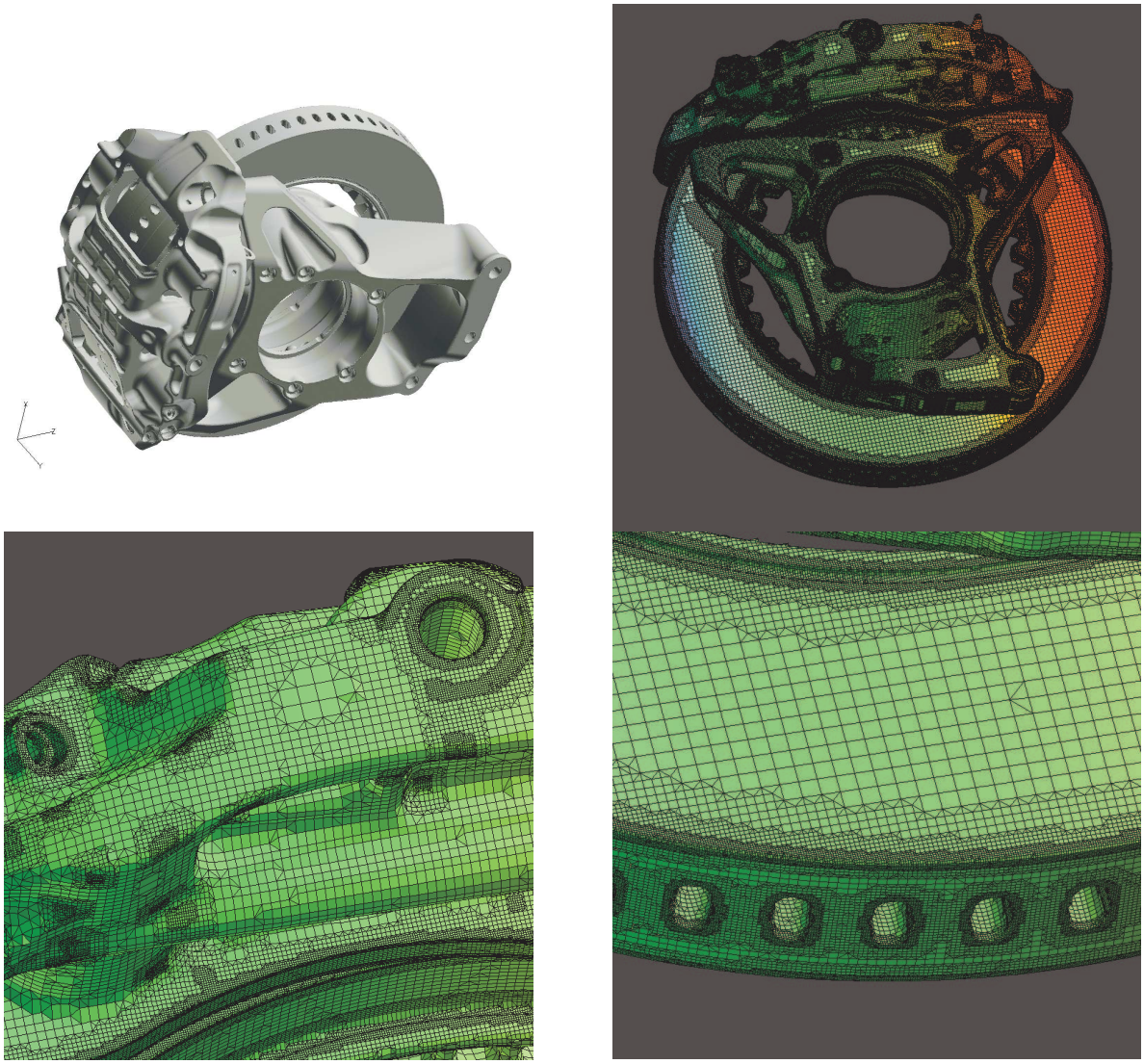

Figure 7: A cooled disc brake assembly – exported, body conformal mesh showing: (top left) the smooth surface; (top right) an overview of the mesh; (bottom row) detail views showing the curvature sensitive variable depth refinement in action.

A B747 in full landing configuration

The final example shows a capability available within the serial, development version of *BOXER* and currently being implemented in the parallel version – viscous layers. The layers are inserted into the mesh, guided by the underpinning Level Set distance field and all the time under the management of an optimizer which prevents the formation of cells with unacceptable mesh quality. This means that in practice clean, layer meshes can easily be constructed on relatively smooth areas of geometry (where there are likely to be boundary layers which can be effectively resolved on layer meshes) whereas near corners the mesh returns to isotropic (which is also what fluid dynamical resolution requires).

Current capability is illustrated by a (rather under-resolved) flow solution of the flow around a B747 in full landing configuration – with all slats, flaps & wheels deployed. Figure 8 shows various views of the body-conformal mesh generated by *BOXER* from imported STL then exported into FLUENT® and solved at approximately approach

speed and angle of attack. The mesh contains around 18M cells and each wetted surface has up to seven viscous layers. Probably a factor of ten more mesh resolution would be needed for a meaningful fluid dynamic simulation but the best test of – and proof of – mesh quality is a flow solution and it is in that spirit that this is presented here.

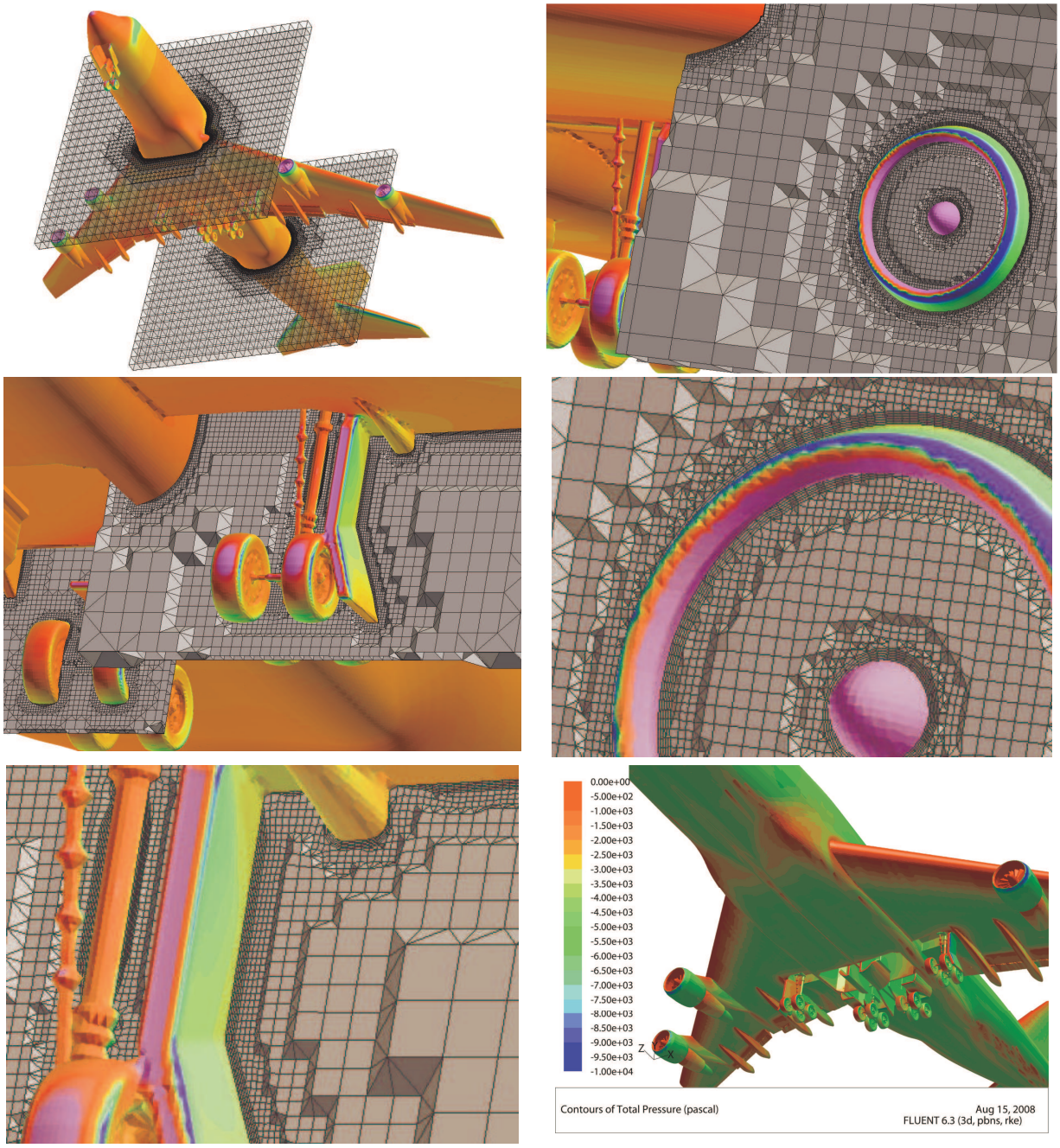

Figure 8: A B747 in full landing configuration; various mesh views showing both the overall mesh generated by *BOXER* with viscous layers together with a flow solution obtained using FLUENT®.

IV. Concluding Remarks

This paper has described recent extensions to the *BOXER* paradigm. The main novelties reported were: first, a generalization of the earlier work to permit variable depth octree refinement to enable variable surface refinement; second, a radical rework of the earlier parallel mesh generation from a simple top-down octree to a bottom-up octree based on Morton coding and Space Filling Curves. Also reported were the associated extensions to permit smooth surface reconstruction from underlying Level Set to allow body-conformal mesh export – with no hanging nodes.

As a practical demonstration, meshes of guaranteed quality were successfully generated for a fully resolved, generic aircraft carrier geometry, a cooled disc brake assembly and a B747 in full landing configuration.

V. References

Adalsteinsson D & Sethian JA "A level set approach to a unified model for etching, deposition & lithography II: three dimensional simulations" J.Comput.Phys, 122, 348-366, 1995

Aftosmis MJ, Berger MJ & Melton JE, "Robust and efficient Cartesian mesh generation for component-based geometry" AIAA J., 36, 6, 952-, 1998

Aftosmis MJ, Berger MJ & Murman SM, "Application of space filling curves to Cartesian methods for CFD" AIAA Paper 2004-1232, Reno January 2004

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

Dawes WN, Dhanasekaran PC, Demargne AAJ, Kellar WP & Savill AM, "Reducing bottlenecks in the CAD-to-mesh-to-solution cycle time to allow CFD to participate in design" ASME Journal of Turbomachinery, 2002

Dawes WN, Kellar WP, Harvey SA, Dhanasekaran PC, Savill AM & Cant RS "Managing the geometry is limiting the ability of CFD to manage the flow" AIAA Paper 2003-3732, 33rd AIAA Fluid Dynamics Conference, 23-26 June 2003, Hilton Walt Disney World, Orlando

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

Dawes WN "Towards a fully integrated parallel geometry kernel, mesh generator, flow solver & post-processor", 44th AIAA Aerospace Sciences Meeting & Exhibit, 9-12 January 2006, Reno, NV, AIAA-2006-45023

Dawes WN, Harvey SA, Fellows S, Favaretto CF & Vellivelli A "Viscous Layer Meshes from Level Sets on Cartesian Meshes", 45th AIAA Aerospace Sciences Meeting & Exhibit, 8-11 January 2007, Reno, NV, AIAA-2007-0555

Galyean TA & Hughes JF "Sculpting: an interactive volumetric modelling technique" ACM Trans., Computer Graphics, vol.25, no.4, pp 267-274, 1991

Glassner AS "Graphics Gems" Academic Press, 1990+

Haimes R & Follen GL "Computational Analysis Programming Interface" Proc. 6th Int. Conf. On Numerical Grid Generation in Computational Field Simulations, Eds. Cross, Eiseman, Hauser, Soni & Thompson, July 1998.

Osher S & Sethian JA "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations" J.Comput.Phys, 79, 12-, 1988

Perng K-L, Wang W-T, Flanagan M & Ouhyoung M "A real-time 3D virtual sculpting tool based on modified marching cubes", SIGGRAPH, 2001

Samareh, JA "Status & Future of Geometry Modelling and Grid Generation of Design and Optimisation" J.of Aircraft, Vol 36, no1, Feb 1999

Samet H "The design and analysis of spatial data structures" Addison-Wesley Series on Computer Science and Information Processing, Addison-Wesley 1990

Sagan H "Space Filling Curves" Springer-Verlag 1994

Sussman M, Smereka P & Osher S "A level set approach for computing solutions to incompressible two-phase flow" J.Comput.Phys, 114, pp 146-159, 1994

Tu T, Yi H, Ramirez-Guzman L, Bielak J, Ghattas O, Ma K-L & O'Hallaron DR "From mesh generation to scientific visualization: an end-to-end approach to parallel super-computing" SC2006, Tampa FL, 2006

Viecelli JA "A computing method for incompressible flows bounded by moving walls" J.Comput.Phys, 8, 119-, 1971

Yerry MA & Sheppard MS "Automatic three-dimensional mesh generation by the modified octree technique" Int.J.for Num.Methods in Engineering, vol.20, 1965-1990,1983