

Automated Two-Dimensional Multi-block Meshing using the Medial Object

Jeremy Gould¹, David Martineau¹, Robin Fairey²

¹Aircraft Research Association Ltd, Bedford, UK jgould@ara.co.uk

²TranscenData Europe Ltd, Cambridge, UK rmf2@transcendata.com

Summary. This paper describes the automatic generation of a structured-dominant mesh based on an automatic block decomposition obtained using the medial object. The mesh generator will produce a mesh for an arbitrary two-dimensional geometry, where quadrilateral-dominant or triangular meshing is used when structured meshing fails. The domain is automatically partitioned into blocks (sub-regions), and an appropriate meshing algorithm for meshing each block is then automatically selected. It will also generate conformal interfaces or hanging interfaces between adjacent blocks as required.

Keywords: Medial object, automated block decomposition, mesh generation, structured-dominant mesh, two-dimensional mesh.

1 Introduction

For aerodynamic analysis, it is generally accepted that structured meshes provide greater accuracy and efficiency compared to unstructured meshes. However, considerable manual effort is still required to generate the block decomposition of the flow domain necessary for structured meshing. The work described in this paper automates the generation of a structured-dominant mesh through generation of a block decomposition using the medial object. The mesh generator will produce a structured-dominant mesh for an arbitrary two-dimensional geometry, where quadrilateral-dominant or triangular meshing is used if structured meshing is not possible. The process will automatically partition the domain into blocks (sub-regions), and then select automatically an appropriate meshing algorithm for each block. It will generate conformal interfaces or hanging interfaces between adjacent blocks where required.

The paper provides details of the design of the two-dimensional mesh generation process and is organized as follows. Section 2 describes re-

requirements for the block decomposition of a domain. Section 3 provides an account of how the medial object is used to generate a block decomposition of the flow domain. The meshing process starts with the meshing of block boundary curves, which is described in Section 4. Section 5 describes a solution of the auto-interval assignment problem, which occurs when generating conformal structured meshes in blocks. Section 6 provides an account of the mesh selection strategy used to automatically choose a meshing algorithm for different blocks, and also to automatically place non-matched interfaces between certain blocks. Sections 7 and 8 gives details of the algorithms employed for meshing of block boundaries and the blocks themselves. Finally, Section 9 demonstrates the application of the mesh generator to three representative aerofoil configurations.

2 Block Decomposition Requirements

A block decomposition of the meshing domain is a partition of the domain into non-overlapping regions which cover the whole domain. The overall aim of the block decomposition strategy used for the mesh generator is to decompose the domain into well-shaped regions suitable for structured meshing wherever possible. The requirements are as follows:

1. Blocks should ideally be four-sided and failing this, three-sided.
2. The block decomposition should be conformal. This means that each bounding curve of a block is shared by exactly two blocks unless the curve is part of the geometry or is part of the far-field boundary in which case it belongs to just one block.
3. Also, if a block vertex lies on a bounding curve of the geometry it should be shared by only two quadrilateral blocks, unless the vertex coincides with a discontinuity of the geometry.
4. If a block boundary curve which is not part of the geometry intersects the geometry then it should do so orthogonally.

Ideally, the viscous boundary layer adjacent to the aerodynamic surfaces should be captured within a region of good quality structured mesh. This implies that the block topology should feature quadrilateral blocks adjacent to the aerodynamic surfaces which extend sufficiently far to capture the likely extent of the boundary layer. Additionally the blocking around geometric features, such as trailing edges, should use an appropriate local topology (C-type, O-type or H-type).

3 Block Decomposition using the Medial Object

The medial object (MO) of a region in two dimensions is defined as the set of centres of all maximal inscribed circles in the region [1,2,3,4]. These are circles contained within the region which are not strictly contained within any other circle inside the region.

The block decomposition algorithm uses a shelling based approach and is capable of automatically subdividing a domain into blocks which are mostly four sided, and mostly have angles close to ninety degrees.

Two dimensional shelling is the process of creating an offset profile or shell that is everywhere a fixed distance from the boundary of a trimmed CAD curve. The two dimensional medial object is instrumental in generating shells of this kind.

Building blocks from a shell requires some modifications above and beyond a true 'offset' operation, in which all points on the shell are a fixed distance from the original surface boundary. These include:

- Corner squaring.
- Split generation for most of the shell.
- Additional split generation using the medial axis more directly in narrow gap regions.
- Nested shell construction.

Corner squaring is the principal modification to the shelling process. It creates square corner pieces in place of the arcs that appear at convex corners of the surface being shelled. The motivation for this is to create four-sided blocks, and avoid exclusively O-type topologies (Fig. 1). For two dimensional planar cases, this is a relatively simple process using the tangents of the shell on either side of the corner. Care must be taken however, and the new vertex we are creating to become the fourth corner of the square must be examined to ensure it does not intersect other shell geometry. The two dimensional medial object is again helpful in carrying out this test and determining a better corner placement (Fig. 2).

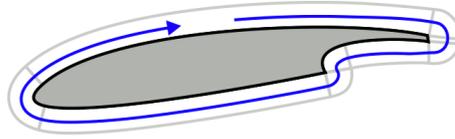


Fig. 1. O-type topology from unmodified shell

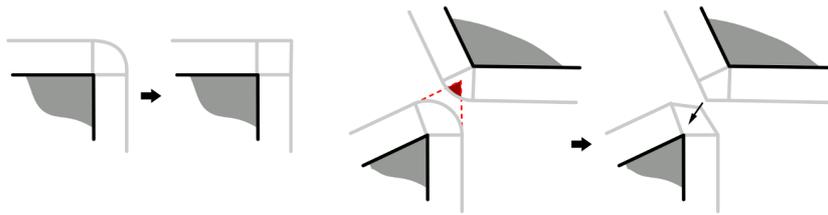


Fig. 2. Corner squaring

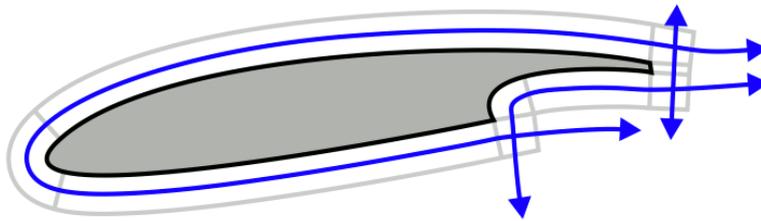


Fig. 3. H-type topology from shell with squared corners

The corner squaring process removes most of the three-sided blocks, and also gives a locally H-type topology (Fig. 3). This technique works for general two-dimensional profiles, and is not specific to aerofoil geometries. Some three sided blocks remain, in highly concave regions or where the shell intersects itself in models with multiple element aerofoils.

Split generation is the splitting of a shell into blocks. A shell is divided into blocks using straight line segments (orthogonal to the geometry if the block is adjacent to the geometry). This is always safe to do, by construction of the two dimensional medial object.

Where a narrow gap occurs, for example between two elements of a multi-element aerofoil, the available space is not wide enough to accommodate two full shell thicknesses, and so the two parts of the shell must join together and be reduced in height. In these regions, the two-dimensional MO will run through the middle of the gap, and sections of the MO can be used directly as block boundary geometry (Fig. 4). This technique also applies when there is detail on the surface boundary that is completely contained within the shell thickness.

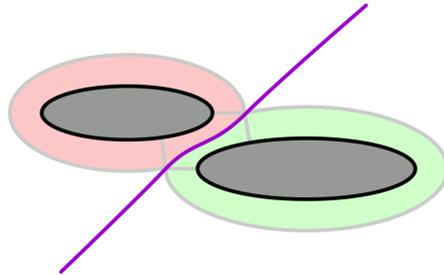


Fig. 4. Medial object used as block boundary

Nested shell construction proceeds by building one layer of shell, before generating a completely new two dimensional medial object on the remaining domain. This is feasible since each medial object computation is relatively cheap.

4 Meshing Block Boundary Curves

The meshing of blocks and block boundaries requires specification of a target mesh density which is provided as input.

As well as the spacing information, refinement of the background spacing is desirable during meshing in order to improve mesh quality. At block boundaries between structured and unstructured zones, there is potential for a large mismatch in spacing, due to the propagation of the curve discretisation through a set of topologically parallel curves. This potential problem is avoided by generating additional mesh sizing information along the boundary of structured zones adjacent to any unstructured zones.

Each boundary curve of a block has an associated marker indicating the nature of that curve. The marker distinguishes between curves belonging to the geometry, curves on the far-field boundary, curves capturing a special flow feature (e.g. a wake plane), or curves in the interior of the domain. These markers are important for meshing because they determine which blocks will be meshed with a geometrical-growth (Navier-Stokes) meshing algorithm.

The first phase of the meshing process involves the generation of an initial discretisation of the curves bounding each block in the input geometry. The initial meshing is required to make subsequent decisions about how to modify the block topology if required. The initial curve meshing is therefore performed independently of any constraints which may be imposed

later, due to structured blocks for example. The curves forming the boundaries of each block are meshed initially using either a standard spacing-based discretisation, or a Navier-Stokes algorithm.

The choice between the above two approaches is based on whether the curve is connected to the end of a curve that belongs to the geometry or to a curve belonging to a special flow feature, which is not itself of one of these types. If it is, then it is meshed using the Navier-Stokes algorithm, otherwise the spacing-based method is used.

Viscous meshes are generated directly, as opposed to generating a mesh based on the Euler background spacing field and then obtaining the viscous mesh through a subsequent refinement process.

5 Auto-interval Assignment

Certain meshing algorithms impose constraints on the numbers of nodes in the meshes of the boundary curves of a block. For example, for a four-sided block to have a structured mesh, it is required that the opposite edges are discretised with the same number of nodes. A quadrilateral meshing algorithm requires an even number of intervals around the whole boundary mesh of the block. The problem of auto-interval assignment is the problem of simultaneously satisfying the constraints for each block so that a conformal mesh can be generated for the whole domain.

An approach to solving the interval assignment problem for structured meshing was proposed by Mitchell in [5]. In Mitchell's paper the auto-interval assignment problem is stated as a mixed-integer linear program. This method involves tackling the determination of all the intervals simultaneously through the definition of constraints, which depend on the meshing algorithm to be applied to each block. However, only the constraint for structured blocks is applicable in this work, and although the linear programming approach is still applicable, a simpler and more robust approach was chosen. The technique used here is based on the notion of curve clan introduced in [6], and provides a way of obtaining solutions to the interval assignment problem for structured meshing based on the spacing specified for the domain.

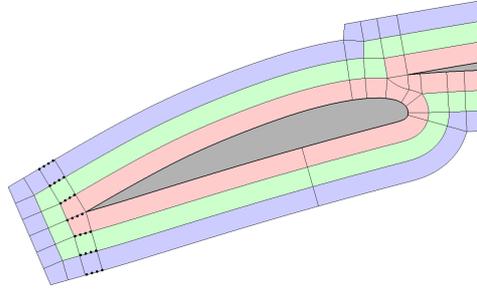


Fig. 5. A curve clan.

A *curve clan* is a set of topologically parallel boundary curves of four-sided blocks in a block decomposition of a domain. The term topologically parallel means that any two curves in a clan are related by a sequence of opposite edges of neighbouring four-sided blocks. The definition is extended in this work to include three-sided blocks by identifying a pair of opposite curves which share the same boundary condition and have the closest match in terms of target mesh density based on the background spacing. If three-sided blocks are allowed, then the determination of curve clans for a given block decomposition is not entirely topological, but depends also on the spacing. In Fig. 5, the set of dashed curves form a curve clan.

Collecting together all such curves into a curve clan, and then meshing all curves in the clan together, will ensure that the requirement for structured meshes mentioned above can be met. Furthermore, curve clans can be meshed independently of each other, and therefore curve meshing is parallelizable at the clan level.

Any discretisation of the curves in the curve clan would be a solution to the interval assignment problem described earlier, as long as all the curves in the clan share this discretisation. For the purposes of this prototype, the discretisation is selected from the curve in the clan with the most nodes in its curve mesh, based on the initial meshing of the curves using either spacing-based or Navier-Stokes mode, as described above. This curve will be called the *clan leader* of the curve clan. The remaining curves in the clan are then meshed by mapping the mesh of the clan leader on to each of the other curves in the clan.

The idea of *curve dependency* is used to facilitate the meshing of one curve (*dependent curve*) by copying the parameterisation of the mesh of another curve (*master curve*). In the case of curve clans, the clan leader will be the master curve for all the other curves in the clan.

6 Mesh Algorithm Selection and Non-conformal Interfaces

The strategy used to select a suitable meshing algorithm for each block in the decomposition of the domain makes use of curve clans and curve dependencies.

A key concept introduced in the new mesh generator is that of a hierarchy of mesh generation algorithms. The aim of this is to provide the best quality meshes without sacrificing robustness.

One of the main aims of the prototype mesh generator is to generate structured-dominant meshes, in order to exploit the improved accuracy and potential efficiency benefits of structured meshes in the flow solver. Hence, structured meshing is the first algorithm in the meshing algorithm hierarchy. A necessary condition that a block must satisfy for it to be given a structured mesh is that it should have three or four bounding curves. Thus blocks with five sides or more are not meshed with a structured meshing algorithm.

The hierarchical meshing approach works by attempting to mesh each block with the “best” meshing algorithm possible, subject to certain constraints. In the two-dimensional meshing prototype, these constraints are purely topological; however the constraints could well include geometric considerations. If structured meshing fails for a block, then an attempt is made to mesh the block using the unstructured quadrilateral-dominant meshing algorithm. If this fails, then triangular meshing will be used.

Experience of this approach in the current work has shown that success criteria need to be given more consideration. In the current implementation, surface mesh quality metrics, including self-intersection checks, are only conducted after all the surface meshes have been generated. It is also clear from the testing that sometimes, a good quality unstructured quad-dominant mesh would be preferable to a poor quality structured mesh, and that the strictly hierarchical meshing algorithm selection described above is not necessarily optimal. This highlights the need for quantifiable definitions of acceptable mesh quality metrics to enable automatic selection of the most appropriate algorithm. However, the approach has successfully demonstrated the improved robustness through the ability to resort to alternative meshing algorithms in the event of meshing failure.

If structured meshing is used for the majority of the blocks in the flow domain, with conformal interfaces between blocks, it can give rise to the problem of high density structured mesh being propagated out to regions with a low target mesh density. This problem is typical in conventional multi-block structured meshing tools. However, in the prototype mesh generator, two possible remedies for this situation are allowed. The first is

to select an unstructured meshing algorithm for a particular block. This effectively splits into two the curve clans running in both directions through the block, allowing them to be meshed with different discretisations. The second approach is to introduce a non-conformal interface in the block topology. Again, this splits a curve clan into two separate clans, but only in one direction. The non-conformal interface is implemented by replacing a single curve in the underlying geometry with two topologically distinct, but physically coincident curves.

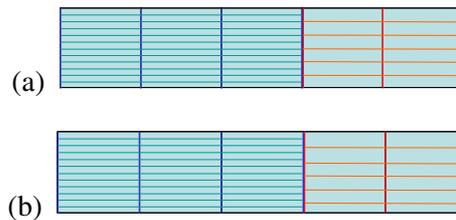


Fig. 6. (a) A hanging interface, (b) a non-matched interface.

There are two types of non-conformal interface used in the block topology. The first type is a *hanging interface* where one curve is a refinement or coarsening of the other (Fig. 6a). This introduces a new type of dependency between the curves. The second type is a *non-matched* interface (Fig. 6b) where the two coincident curves can have completely independent discretisations, and have no dependency on each other.

A hanging or non-matched interface can be used to generate structured mesh which conforms more closely to the target background spacing, but the mesh will of course, no longer be conformal.

The automatic block meshing algorithm selection strategy initially assumes that all three and four-sided blocks will be meshed using a structured algorithm, and all other blocks will be meshed using an unstructured algorithm. Block boundary curves are selected to be hanging or non-matched interfaces or structured/unstructured interfaces by analyzing curve clans. The idea is to compare spacing along the different curves in a curve clan. A curve is chosen to be a non-conformal interface when there is a large decrease in spacing between that curve and the next or previous curve in the clan. This is implemented as follows. Recall that all the boundary curves of the block decomposition are given an initial mesh. These meshes are used to calculate the ratio of the number of nodes in the mesh of each clan curve to the number of nodes in the mesh of the clan leader. A sequence of Booleans is generated for the curve clan by comparing the sequence of ratios to a configurable threshold value. A ratio above the threshold is mapped to true (T), otherwise it is mapped to false (F).

Setting every curve with a false value to be a hanging interface may lead to too many hanging interfaces. Therefore, a form of smoothing is applied, which replaces the patterns of the form TFT and FTF in the sequence of true/false values associated with a clan by the patterns TTT and FFF, respectively. After the smoothing operation interface curves are selected by searching for patterns of the form FTT, TTF, FFT and TFF. If either pattern is found, then the interface curve will be the curve in the clan corresponding to the middle symbol. Only one split is allowed in each clan.

Once the decisions have been made about where to introduce unstructured blocks or non-conformal interfaces, the curve clans are re-determined for the modified topology. The above process is then repeated for the new set of curve clans, and more interfaces introduced if necessary. This cycle is repeated until no new non-conformal interfaces or unstructured blocks are deemed necessary.

7 Block Interface Meshing

As mentioned above, block boundaries are initially meshed using either standard spacing-based discretisation, or an advancing-layer style discretisation. The advancing-layer curve meshing algorithm is based on the following parameters: first cell height, number of linear layers and growth rate. The algorithm uses a geometric expansion of the projected edge length which terminates when the projected edge length matches the background spacing. The remainder of the curve is then meshed using the conventional spacing-based discretisation technique.

Following the initial curve meshing, the curves are re-meshed based on the constraints imposed by the blocking topology and the choice of meshing algorithms and block interface types. The re-meshing of the curves is based entirely upon the curve dependencies. The re-meshing involves an iterative loop over the curves, only meshing a curve if it either has no dependency, or if its *master* curve (i.e. the curve upon which it is dependant) has already been re-meshed. The curve dependency can take one of two forms: *direct* or *hanging*.

In the case of a direct dependency, the parametric distribution is copied from the master curve. A small amount of Laplacian smoothing is applied to the curve distribution as this was seen to improve the mesh quality in certain situations, for example, smoothing out the Navier-Stokes refinement away from the aerofoil geometry.

At a hanging interface, two possibilities exist for generation of the dependent curve mesh: refinement of the master mesh or coarsening of the

master mesh. Refinement of a master mesh is the simplest operation, as it simply involves sub-dividing each segment of the master mesh into a specified number of edges. Coarsening of a master mesh is slightly more involved. The coarse mesh is generated through sub-sampling of the master mesh using a specified coarsening factor, however, the master mesh nodes must then be projected onto the segments of the coarse mesh in order to ensure conservation of volume at the interface and avoid creation of gaps or overlaps in the mesh.

Experience of applying hanging interfaces to realistic aerofoil geometries revealed that a fixed refinement or coarsening factor is not necessarily desirable, since the background spacing can vary considerably along the block boundary. To account for this, a modified algorithm was implemented for curve coarsening based on a variable coarsening factor. This modified algorithm uses a rather different approach to the curve remeshing. The curve is first meshed using the standard sourcing-based approach, and then the nodes are snapped to the closest nodes in the master mesh. As with the fixed-ratio coarsening, the master mesh nodes must then be projected onto the segments of the coarse mesh.

Initially, meshing of non-matched curve interfaces might seem trivial, since the two curves are simply meshed independently from one another. However, it is essential that volume is conserved at the interface, and that there are no gaps or overlaps in the mesh. For the implementation of the prototype, non-matched interfaces were therefore only allowed at straight block boundaries.

8 Block Meshing

The main algorithm used to generate structured meshes is linear transfinite interpolation. The method extends to triangular blocks, by collapsing one of the boundary edges of the block.

For a rectangular block, linear transfinite interpolation will give a perfectly rectilinear surface mesh. However, as the shape of the block deviates from being rectangular the quality of the mesh will deteriorate. For such cases, the mesh lines attached to the aerofoil will not be orthogonal to the aerofoil surface. For particularly badly shaped blocks, linear transfinite interpolation may not be one-to-one which will cause ‘folds’ in the surface mesh.

For this prototype meshing capability, an alternative algorithm was implemented for structured meshing of “viscous” blocks. The alternative exploits the fact that, by construction, the blocks adjacent to the surface of

the aerofoil geometry will have straight edges in the direction perpendicular to the aerofoil surface. This allows the structured mesh generation to be reduced to a linear interpolation of the distributions along the perpendicular block edges.

Hermite transfinite interpolation [7] was also used. This technique is able to generate a mesh for a non-rectangular block which has mesh curves orthogonal to the boundary curves of the block. This method was modified to allow normal vectors to be specified only along a wall-type boundary curve, where orthogonality of the mesh is most critical.

9 Results and Discussion

The mesh generator has been demonstrated on three test cases: a clean wing case, a high-lift aerofoil, and a clean aerofoil with a leading edge ice shape. Viscous meshes suitable for RANS simulation of nominally attached flows have been generated for all cases.

The first test case corresponds to the computation of flow around a single element two-dimensional aerofoil section at the transonic cruise condition. The test case used is the RAE 2822 aerofoil section, which exhibits regions of concave and convex geometry. The test case has a sharp (discontinuous) trailing edge. The block topology and mesh generated for the RAE 2822 aerofoil are shown in Fig. 7a and Fig. 7b, respectively, based on six layers of shelling. The block topology and mesh demonstrate the desirable C-topology around the aerofoil resulting from the corner squaring process.

A series of meshes were generated for the RAE 2822 configuration using various options to accommodate changes in target mesh density:

- a. Default mesh generated with structured blocks wherever possible and only conformal interfaces
- b. Structured-dominant mesh with hanging interfaces of fixed ratio.
- c. Structured-dominant mesh with hanging interfaces of variable ratio.
- d. Structured-dominant mesh with non-matched interfaces.
- e. Completely conformal mesh generated using largely structured meshing, but allowing use of unstructured blocks to accommodate changes in density.

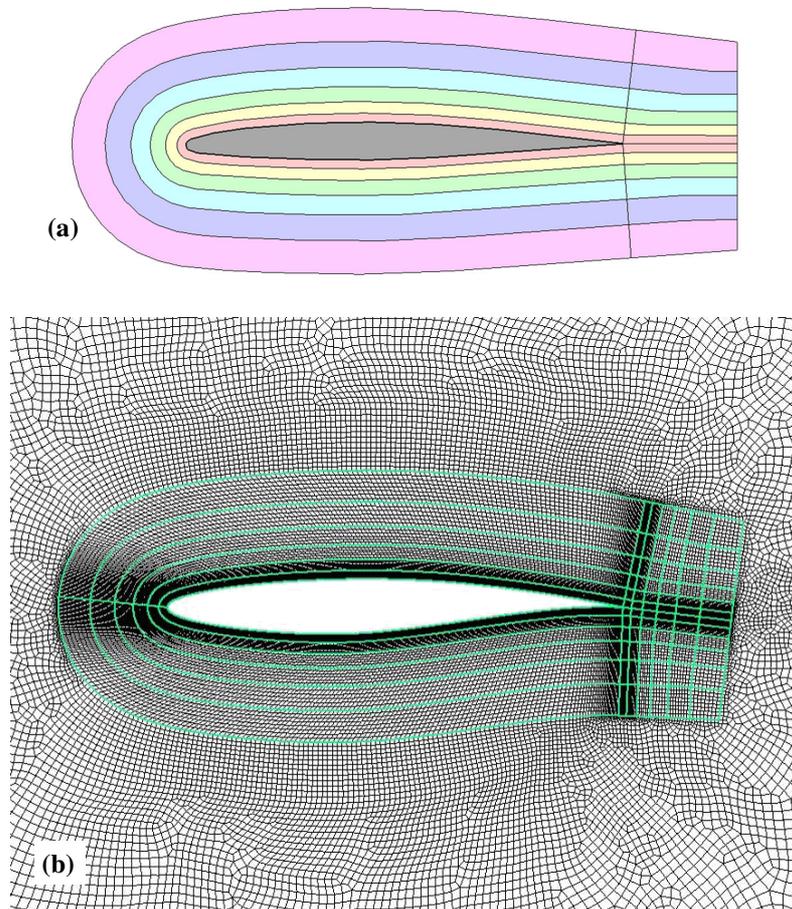


Fig. 7. (a) Block topology, and (b) mesh generated on RAE 2822 aerofoil test case.

The different meshes corresponding to these alternative strategies are shown in Fig. 8 (a) to (e) and illustrate how the variation in target mesh density can be handled in a variety of ways.

The second test case is the L1T2 geometry, a multi-element high-lift configuration at take-off or landing. This system involves three elements, a leading-edge slat, main element and trailing-edge flap. Fig. 9 shows the detail of the medial object computed on the L1T2 test case, which includes segments passing through the gaps between the elements, and also segments which approach the cove region of the slat and main element.

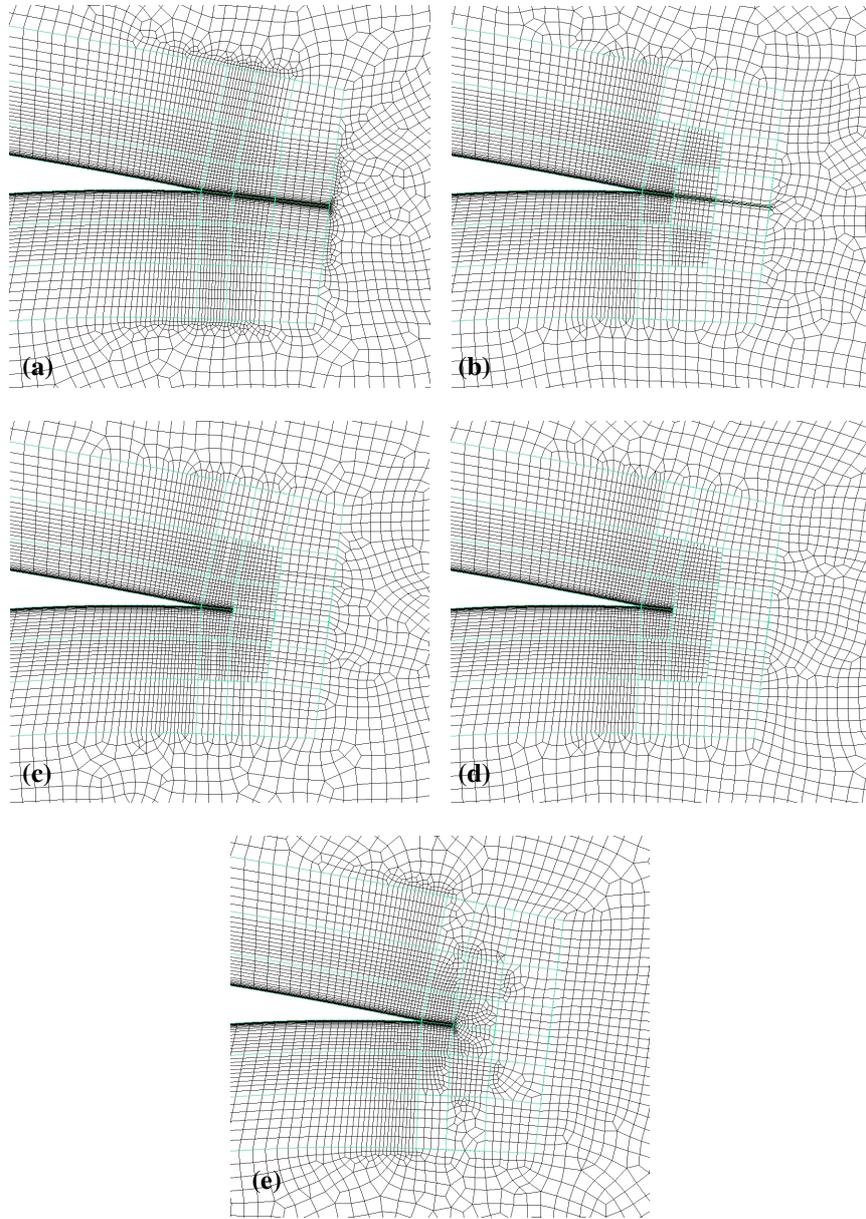


Fig. 8. RAE2822 mesh at trailing edge of aerofoil with (a) no interfaces, (b) fixed ratio hanging interfaces, (c) variable ratio hanging interfaces, (d) non-matched interfaces, and (e) unstructured blocks.

The block topology for this case is shown in Fig. 10, illustrating how the medial object is used to form part of the block boundaries in gap regions. Various details of the structured-dominant mesh generated for the L1T2 test case are shown in Fig. 11. The effect of using only conformal interfaces in this mesh can be seen in the propagation of the refined mesh downstream from each of the trailing edges of the three elements.

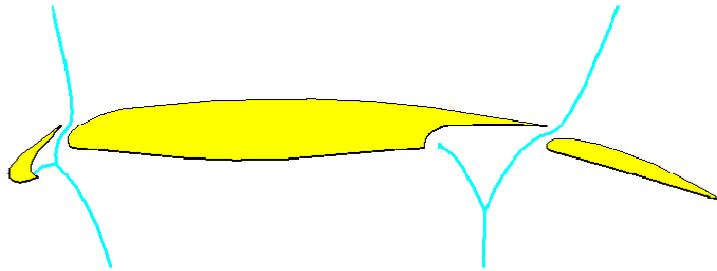


Fig. 9. Detail of the medial object close to the L1T2 geometry

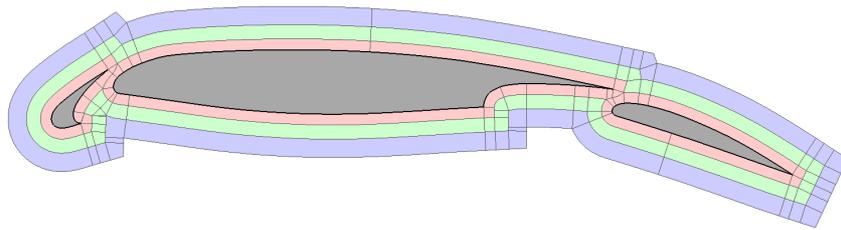


Fig. 10. Block topology generated on the L1T2 test case

The final test case corresponds to a clean aerofoil configuration with leading edge ice shape. The test case used is the C6 configuration from the NATO-RTO Workshop on Ice Accretion Simulation in 2000 at CIRA, Italy [8]. The medial object and block topology generated for the C6 icing case are shown in Fig. 12. The shelling offset is relatively small compared with the chord length, in order to capture the detail of the ice shape.

Detail of the mesh on the C6 icing test case is shown in Fig. 13a and 13b. In this case triangular meshing was used to generate the mesh in the far-field region of the flow domain as a result of a failure in the quad-dominant advancing-front algorithm, demonstrating the hierarchical meshing approach. The detail of the mesh in Fig. 13b demonstrates the potential of this automatic blocking and mesh generation process to handle viscous mesh generation around complex geometric features.

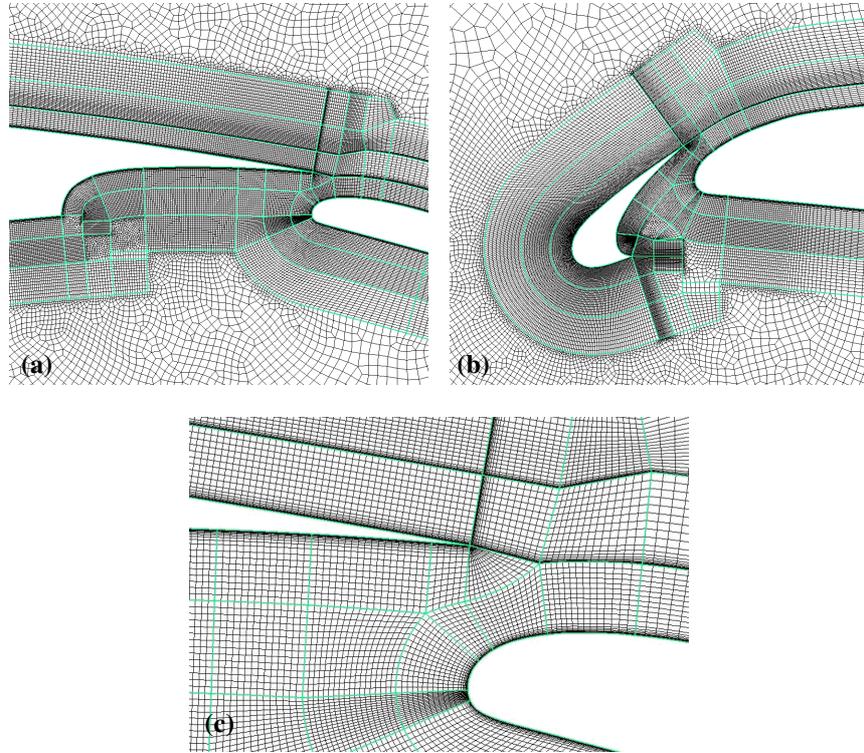


Fig. 11. Detail of the mesh on the L1T2 test case at (a) region around the slat, (b) cove region of main element, and (c) gap between main element and flap.

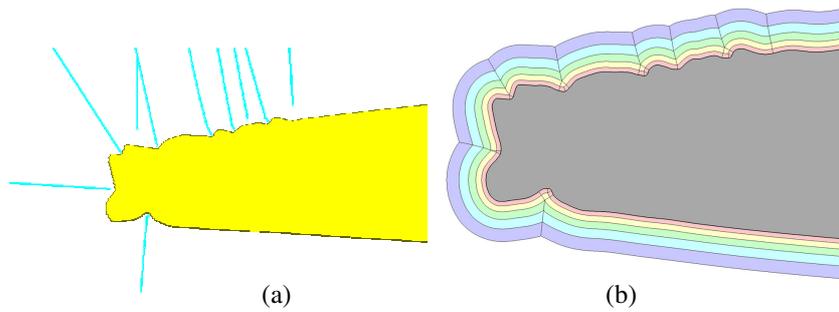


Fig. 12. Detail of the (a) medial object, and (b) block topology at the leading edge of the C6 test case.

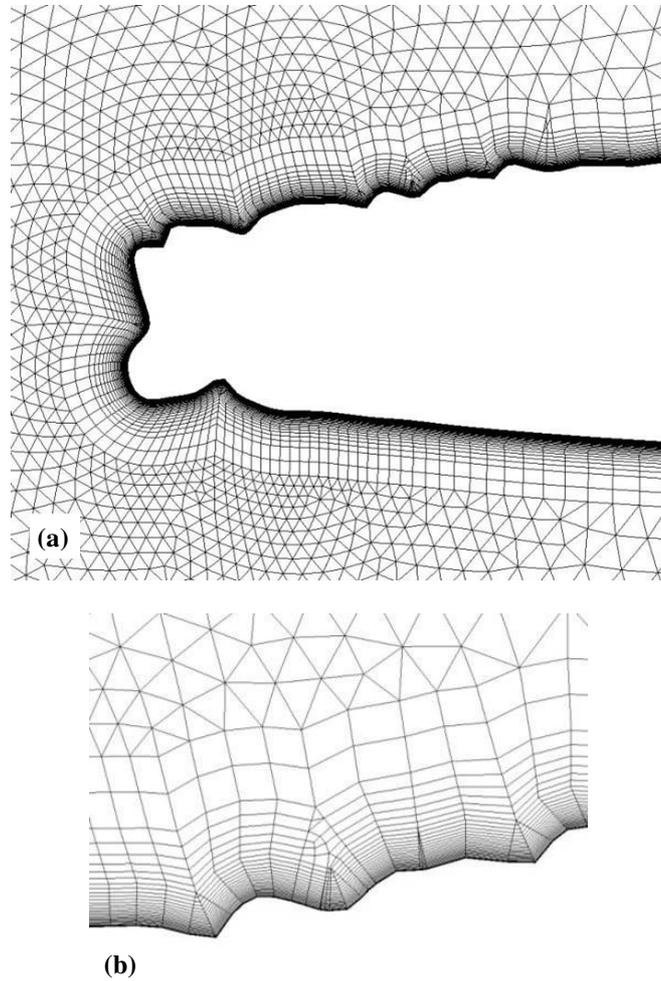


Fig. 13. Mesh on the C6 icing test case around (a) the leading edge, and (b) a detail of the ice shape.

10 Conclusions

A new prototype mesh generator has been presented which will produce a structured-dominant mesh for an arbitrary two-dimensional geometry. The process will automatically partition the domain into sub-regions or blocks using the medial object. An appropriate meshing algorithm is automati-

cally selected to generate each block mesh. Quadrilateral-dominant or triangular meshing is used if structured meshing is not possible. The process will generate conformal interfaces or hanging interfaces between adjacent blocks where required.

Acknowledgement. The work reported here has been carried out under the ANSD project jointly funded by Airbus, ARA, TranscenData and the UK Technology Strategy Board. The medial object computation and automatic blocking technology was developed in the context of TranscenData Europe Ltd's CADfix product [9].

References

1. Blum, H A (1967) Transformation for Extracting New Descriptors of Shape. In: Models for the Perception of Speech and Visual Form
2. Sheehy D S, Armstrong C G, Robinson D J (1995) Computing the medial surface of a solid from a domain Delaunay triangulation. In: Proc ACM/IEEE Symposium on Solid Modeling and Applications, Salt Lake City
3. Price M, Stops C, Butlin G (1995) A Medial Object Toolkit for Meshing and Other Applications. In: Proceedings of the 4th International Meshing Roundtable
4. Robinson T, Fairey R, Armstrong C, Ou H, Butlin G (2008) Automated Mixed Dimensional Modelling with the Medial Object. Proceedings of the 17th International Meshing Roundtable
5. Mitchell S A (1997) High fidelity interval assignment. In: Proceedings 6th International Meshing Roundtable
6. Beatty K, Mukherjee N (2010) A Transfinite Meshing Approach for Body-In-White Analyses. In: Proceedings of the 19th International Meshing Roundtable
7. Faux I D, and Pratt M J (1979) Computational Geometry for Design and Manufacture. Ellis Horwood, Chichester
8. Ice Accretion Simulation Evaluation Test (2001) Report of the Applied Vehicle Technology Panel (AVT) Task Group AVT-006, RTO Technical Report 38
9. www.transcendata.com/cadfix.htm