INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **43**:785–821 (DOI: 10.1002/fld.497)

The rapid and robust generation of efficient hybrid grids for rans simulations over complete aircraft

J. A. Shaw*,†,[1], S. Stokes[1] and M. A. Lucking[2]

[1]*Aircraft Research Association, Manton Lane, Bedford, MK41 7PF, U.K.*
[2]*BAE SYSTEMS, Warton, Lancashire, PR4 1AX, U.K.*

SUMMARY

A description is given of a method capable of automatically forming efficient, high-quality hybrid meshes for viscous flow calculations over complete aircraft, given engineers' requirements on mesh density and first cell-height. Mixed quadrilateral/triangular surface grids are constructed using the moving-front approach, with point placement and connection optimized to ensure high-quality, anisotropic panels are generated. The volume mesh is built in two stages. A near field mesh is formed using the advancing-layer approach, significantly enhanced through the use of mesh enriching/collapsing techniques and layer-intersection detection/front-termination algorithms. The remainder of the domain is then covered by a cut-cell Cartesian mesh of varying density that conforms to the outer-surface of the near field mesh. Examples of meshes generated using the described methods are presented and the user-input discussed. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: computational grids; three-dimensional flow; Navier–Stokes equation

1. INTRODUCTION

A capability that allows engineers to predict readily, accurately and efficiently high Reynolds number flows over geometrically complex shapes has been a strong requirement of both industry and research establishments for a number of years. To date this need has not been met, primarily due to difficulties encountered in extending the meshing techniques developed for inviscid flows to the creation of the highly anisotropic cells required for viscous flow modelling. Against this background, industry and research-establishments are becoming ever more determined to minimize the risk and maximize the return from their investment in research. Within the U.K., this situation has led to the formation of a collaborative

* Correspondence to: J. A. Shaw, Aircraft Research Association, Manton Lane, Bedford MK41 7PF, U.K.
† E-mail: jshaw@ara.co.uk

Contract/grant sponsor: U.K. Ministry of Defence
Contract/grant sponsor: U.K. Department of Trade and Industry
Contract/grant sponsor: QinetiQ
Contract/grant sponsor: BAE SYSTEMS
Contract/grant sponsor: Airbus U.K.

Received 15 January 2003
Revised 15 January 2003

industry/research-establishment CFD programme, the objective of which is to develop a common, rapid-response, Navier–Stokes capability. The project, entitled SOLAR, is founded on utilizing the collective expertise of BAE SYSTEMS, Airbus UK (AUK), QinetiQ and the Aircraft Research Association (ARA) to deliver the objective.

Ahead of work commencing on the SOLAR project, a detailed requirement capture for the capability was performed. CFD developers and users from BAE SYSTEMS, AUK, QinetiQ and ARA then held discussions to determine the overall technical strategy to meet the requirement. Experiences in the development of methods for simulating both inviscid and high Reynolds number flows were shared and reviewed. These included findings from the use of block-structured [1–5], overlapping [6, 7], unstructured [8, 9], hybrid structured-unstructured [10–14] and Cartesian [15] approaches. The outcome of these meetings was a common view on a strategy for the SOLAR project that would deliver effectively the desired accuracy, flexibility and efficiency.

For the flow solver, a symmetric cell-centred discretization would be adopted, with explicit artificial-dissipation added and multi-grid used to advance to a steady-state solution. Closure would be achieved with a 2-equation differential turbulence model. This scheme had already been found to work well on block-structured grids [5] and the intention was to extend this to the mesh environment described below. Further details of this flow algorithm are given in Reference [16].

For the mesh generation, a hybrid grid strategy was favoured, with the number of hexahedral elements maximized wherever possible. The preference for this approach was based on previous work [11] on structured, unstructured and hybrid grids. The method of grid generation would, however, have to be capable of greater automation, and require less user-expertise, than adopted in earlier efforts [11].

With this in mind, it was crucial that the various elements could be generated without recourse to user-directives. The overall strategy that was settled upon is illustrated in Plate 1, which shows views of a SOLAR mesh for a wing–fuselage configuration. Surface grids composed of quadrilaterals (typically $>99\%$) and triangles would be generated by enhancing the Moving-Front algorithm currently used by BAE SYSTEMS and AUK for solely triangular surface mesh generation. The volume mesh would be formed in two stages, one for the near-field using the advancing-layer approach [17–19] to create hexahedra and prisms, and one for the far-field using embedded regions of Cartesian grid. In the latter case, the cells would be cut [20] to conform to the outer-surface of the near-field grid.

This approach would require fundamental enhancement [21] of the basic advancing-layer approach. A mesh-collapsing technique that had been successfully investigated for marching prismatic elements [14] away from concave boundaries would need to be extended to operate on both prismatic and hexahedral elements. Equally, it was evident that mesh-enrichment would be beneficial in aiding the marching process to perform robustly and produce high quality grids near to convex boundaries. Furthermore, it would be necessary to develop front-intersection detection and front-stopping algorithms to handle situations where the near field mesh from one surface grew towards that emanating from another surface. Store separation, where the configuration of interest is composed of disjoint, often closely located surfaces, is an obvious application requiring this technology.

The remainder of this paper discusses and demonstrates the algorithms developed within this overall surface and volume mesh generation strategy [22]. Following a description of the user-input to the grid generation process in Section 2, the generation of quadrilateral

surface grids is described in Section 3. This is followed in Section 4 by an explanation of the near-field meshing [23], before the detailing of the far field meshing in Section 5. Finally, examples of the application of the algorithms to configurations of current interest are presented in Section 6.

2. USER-INPUT FOR GRID GENERATION

The objective of the SOLAR development team is to *develop a tool that can be used readily by aeronautical engineers to deliver accurate viscous flow solutions within 24 h of finalizing the CAD representation of an aircraft configuration.* This objective has already been met for inviscid calculations, with all user-input performed interactively, prior to the generation of the surface meshes, within a Graphical User Interface front-end (See Plate 2). A requirement that follows from the need to insert the SOLAR technology into the design environment as quickly as possible is that the same front-end should be the user-interface to SOLAR. Furthermore, the amount of additional user-input should be kept to a minimum. For inviscid flow modelling, based on the use of isotropic, tetrahedral meshes, users are required to form the patch-to-patch connectivity of the CAD data and a spatial distribution of sources. The strength of the sources ultimately influences, in turn, the point distribution along each patch boundary, the density of the surface mesh and the clustering of the volume elements [8, 9].

Two additional geometrical inputs have been identified as being necessary for efficient, viscous flow modelling, which requires a high degree of anisotropy in the mesh normal-to-the-surface direction. The first of these is geometric surfaces to represent idealized wakes (these having already been found necessary for improved flow modelling based on viscous–inviscid coupling techniques [24]) and the second is a desired distribution of first cell-height. The former data is set up readily using simple CAD functionality. The latter is specified either as a global, user-defined distance, or, more generally, a separate set of sources are distributed within the domain to control the normal-to-the-wall mesh spacing, with the usual 'inviscid' set of sources solely used to control the across-the-surface mesh spacing. In either case, the user-input can be scaled to build meshes for a range of alternative Reynolds numbers, for which a variation in near-wall spacing is required to maintain flow modelling accuracy.

Further to this, within SOLAR, the user can also introduce a further set of sources that control the across-the-surface mesh spacing for regions where a high degree of anisotropy in the surface mesh is required. Typically, this would be at the leading edge of wings, where usually both the degree of surface curvature and the flow field gradients are significantly greater in the chordwise direction than in the spanwise direction. The benefit of expending the effort to specify this additional data is apparent in terms of the significant reduction in the number of cells in the final mesh and the ensuing reduction in execution time of the flow solver.

For complex cases, experience suggests that set up times for viscous calculations can be as much as 30% more than for inviscid cases. This is considered acceptable and, for comparison, is a very marked improvement in the difference in the user effort required to generate block-structured meshes for inviscid and viscous simulations.

Following the setting up of the user-input data, meshes for the boundaries of the domain are generated. The algorithms that form these meshes are described in the following section. It is noted that these algorithms can either be invoked as a batch process operating on all

surfaces or as the generation of a subset of surface meshes from commands executed within the GUI in which the user-input data is established.

3. SURFACE MESH GENERATION

The nature of a fluid flow is determined by its response to the precise shaping of the configuration's surfaces. This, coupled with the fact that the surface grid acts as an initial condition for field grid generation, makes high-quality surface mesh generation essential in the CFD process. From a user-perspective, surface grids must be readily generated, particularly since realistic configurations are often comprised of hundreds of surfaces and need to be turned-round under very demanding time-scales.

The SOLAR surface mesh generation strategy is based around an Advancing-Front approach [8], where the algorithm seeks to produce anisotropic quadrilateral elements, but builds triangles in preference to constructing low-quality quadrilaterals.

3.1. Geometry decomposition

Numerous surfaces form the overall geometry definition of a complete aircraft. For such complex entities, some surfaces may have geometric properties that would make the generation of efficient meshes of acceptable quality a near to impossible task. For this reason, a strategy has been followed whereby the total surface mesh generation process is performed over a set of zones, where each zone has at least one, and sometimes more, underlying geometric representations. The so-called 'intersection lines', which include lines that bound abutting zones, serve to trim the geometric entities into 'meshing zones'. These delimiting lines form the initial front that is to be advanced from and serve to de-couple each zone from its neighbours, allowing them to be treated as separate entities and hence meshed in parallel. The meshes for each zone are generated in real space and then projected onto the real surface(s) using the underlying surface representation(s). In practice, the CAD representations that SOLAR can use include bi-cubic, Ferguson or NURBS surfaces.

The boundary lines are discretized into segments whose lengths are prescribed by the user-specified spacing distribution discussed in Section 2. Care is taken to ensure that each line is split into an even number of segments, for reasons discussed in Section 3.2.5. The discretized lines for each zone are then formed into consistently ordered loops so that as the boundary front is traversed, the meshing region lies on the left.

3.2. Advancing-front quadrilateral surface mesh generation

This Advancing-Front approach extends the concept of paving algorithms. These form successive loops of cells inwards from the initial front until the surface is completely meshed. The approach has been shown to work well for relatively basic shapes [25, 26], but has proven difficult for arbitrary surfaces where multiple loops can intersect. The method's robustness can be improved by adopting a cell-by-cell approach [27], instead of the loop-by-loop; both to check mesh validity and handle problems. However, the cell-wise approach still forms cells in the same order as the row-wise method.

The looping concept is discarded in favour of an intelligent selection of the best region in which to create cells. Effectively each edge in the front has associated with it a selection

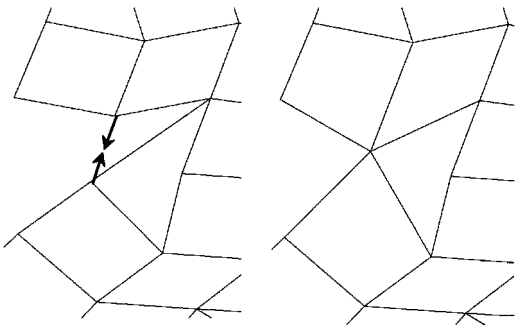

Figure 1. Before and after seaming edges.

weighting; the definition of which is automatically chosen to suit the specific type of zone being meshed. This weighting allows the meshing algorithm to select and treat difficult areas of the front first, for example where there are large gradients in cell size. The probability of advancing out of troublesome areas is thereby maximized and the result is a very robust method that yields high quality cells in all areas of the mesh.

Once an edge has been selected, it forms the basis for the generation of a new quadrilateral. This requires the generation or selection of two points to connect to either end of the edge, creating three new sides. The front must then be updated to reflect the edges that have been added and removed by the cell creation. This process is discussed in more detail below.

3.2.1. Edge selection. Various criteria can be used to determine the selection weighting for an edge. Experience has shown that certain areas are particularly demanding to mesh. These are characterized by adjoining edges that produce a sharp internal-angle and edges that have a small required mesh reference-length in the transverse (in two dimensions) direction, i.e. where the mesh needs to be highly anisotropic. Consequently, the weighting for edges in such regions needs to be high. A simple definition of selection-weighting that works well is

$$\text{weight} = l_{\max}/l + \alpha_{\max}/\alpha \tag{1}$$

where l is required edge length in the transverse direction, α is the angle between a given edge and the one following it in the front and the subscript max denotes the maximum value of these quantities. In practice, l is determined by using the mid-point of the edge as a reference position for input to the algorithms that analyse the isotropic and anisotropic source data.

3.2.2. Special case check. Mesh quality and algorithm robustness are greatly improved if:

- front loops with six or less edges are explicitly closed with the best combination of cells possible;
- Adjacent edges with a small included-angle are seamed together as illustrated in Figure 1.

3.2.3. Point placement & selection. The selected edge forms the basis of a new quadrilateral. Hence, two extra points must be created/selected which, when connected to the existing points

at either end of the selected edge, will form the remaining three faces of the new quadrilateral. A multi-step process is used.

3.2.3.1. Initial placement. An initial 'best-guess' for each of the two point's locations is made, in isolation of knowledge of the other point's position. The placement depends on angles between adjacent edges and the demanded cell size. Note that the algorithm strives to ensure that all edge lengths are as near as possible to the desired value.

3.2.3.2. Co-optimization. The new points are moved until they are in the best position relative to each other. The major factor here is whether the distance between them is compatible with the user-specified desired edge-length. This new *ideal* cell is then checked to see if it should be merged with other areas of the front.

3.2.3.3. Existing point selection. Each new point is assessed to see whether existing points lie close to them. If so, then these existing points are flagged as 'candidate points' to which the new points may be moved. Candidate points are each assessed for the quality of the quadrilateral to which they would lead and ranked accordingly. Use of a candidate point is always favoured over placement of a new point as this prevents points being placed very near to existing edges, which would create poor cells.

3.2.3.4. Surface projections. Created points are projected to the surface at various stages during mesh generation to ensure surface conformity. However, this process is an expensive operation and so its use is minimized as much as possible. The surface projection, as well as calculation of local surface normal, are the only operations that a surface must support. This means that the surface mesh generator can work with a variety of different surfaces, from simple spline definitions through to commercial CAD-type representations.

3.2.4. Point connection & cell formation. Each new edge is checked for validity, it must:

- not intersect an existing edge;
- lie on the meshing side of its adjoining edges.

It is possible that a valid set of connections cannot be found, in which case quadrilateral formation using this particular edge is abandoned. A new edge is then picked to restart the process, however, instead of basing the selection on the aforementioned selection weighting (Section 3.2.1), one of the 'failed' edge's neighbours are picked. In this way, the algorithm works around the area of difficulty until a successful solution is achieved.

3.2.5. Front maintenance. After adding a new quadrilateral, the front has to be updated to reflect the edges that have been removed and/or added, whilst maintaining the 'mesh on the left' rule. There are several ways in which edge connectivity can be altered, depending on the type of cell addition. For example, two loops can be merged into one or *vice versa*.

A condition for the creation of an all-quadrilateral mesh is that the boundary must consist of an even number of segments. The boundary intersection lines are discretized with this in mind (Section 3.1), but the even number must be preserved by subsequent connections if the possibility of an all-quadrilateral mesh is to be maintained. Consequently, when a front loop is split or joined, a check can be made for an even number of faces in each loop. In practice,

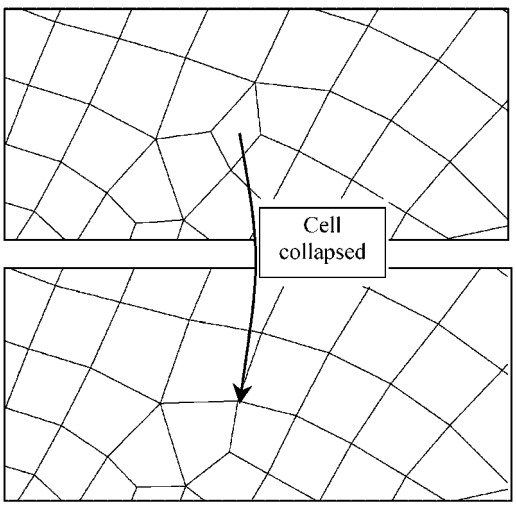

Figure 2. Topological operation to improve mesh quality.

however, the algorithm produces an acceptably small number of triangles (typically $<0.5\%$) without this explicit detection and so it is often ignored for speed.

3.3. Post meshing refinements

The raw mesh can be improved in two basic ways; topological changes and smoothing.

3.3.1. Topological improvements. One type of improvement is edge swapping. This is performed in regions of the mesh where the initial topology is not adequately modelling the curvature of the surface. Other operators [28, 29] add or remove cells to improve node connectivity, as illustrated in Figure 2.

3.3.2. Smoothing. A simple Laplacian smoothing technique is applied to the mesh. A constraint that must be observed is that points on the surface intersection lines must not be moved. This again means that each surface can be smoothed separately and in parallel if desired.

3.4. Surface mesh reconstruction

The volume mesh generator requires each surface mesh face to be ordered in an anti-clockwise direction when viewed from the flow domain. Hence, the final step of the surface mesh generator is to ensure that the cells for each surface are correctly orientated. Counting the number of times a vector from the surface intersects all other surfaces achieves this.

3.5. Examples of application of surface mesh generation technology

Using the algorithms described, surface meshes have been generated on a wide range of realistic configurations, comprised of very many surfaces. Two examples of surface meshes for alternative builds of the Tornado aircraft are given in Figures 3 and 4. In the first,

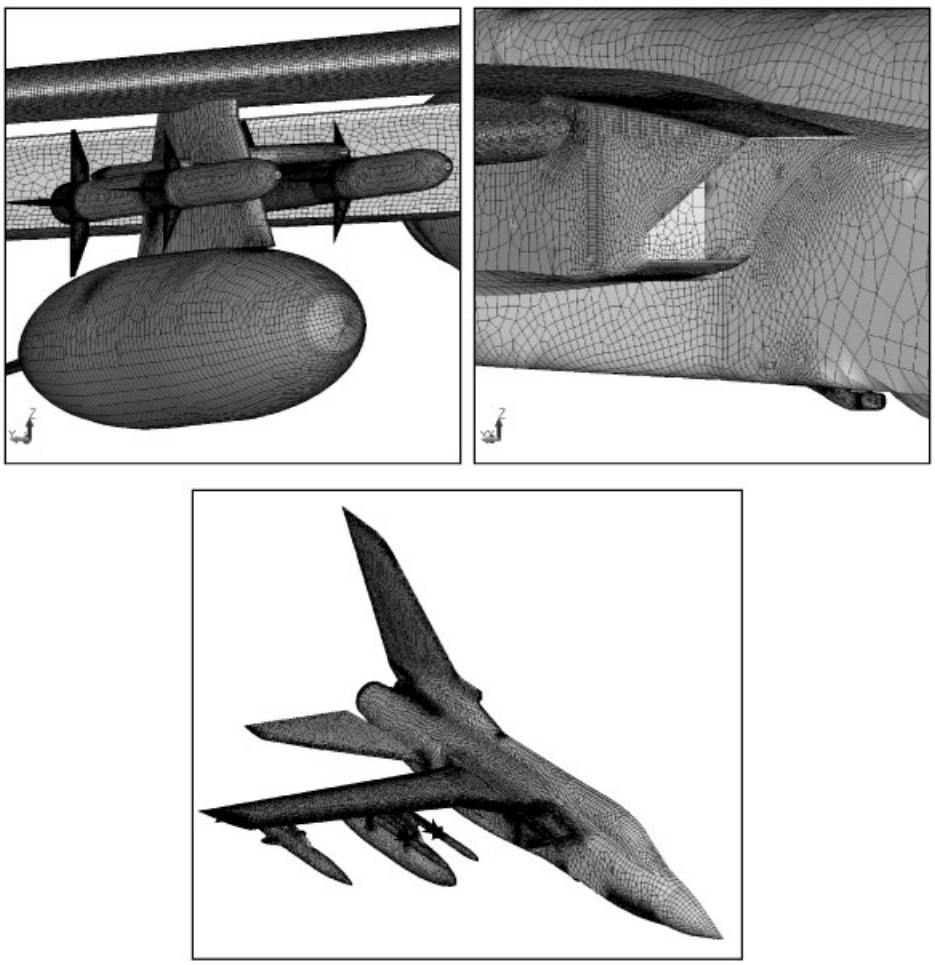

Figure 3. Global and close-up views of surface mesh on Tornado aircraft laden with stores generated using only isotropic sources.

geometrically complex example (Figure 3), the aircraft is laden with multiple stores. The general quality of the quadrilateral faces is high, with a high degree of regularity in the mesh connectivity in geometrically simple regions, such as on the surface of the large fuel tank. This mesh was generated using isotropic sources only. Hence, the surface is unnecessarily highly resolved in some directions to ensure that the surface curvature in other directions is adequately modelled. In the second example (Figure 4), anisotropic sources have been used. The figure focuses on showing the detail of the mesh around the intake region. The aspect ratio of the faces in this region is about 10:1, giving an efficient coverage of a complex region of geometry over which flow gradients will be significant in the direction of the anisotropy throughout the operating conditions of the aircraft.

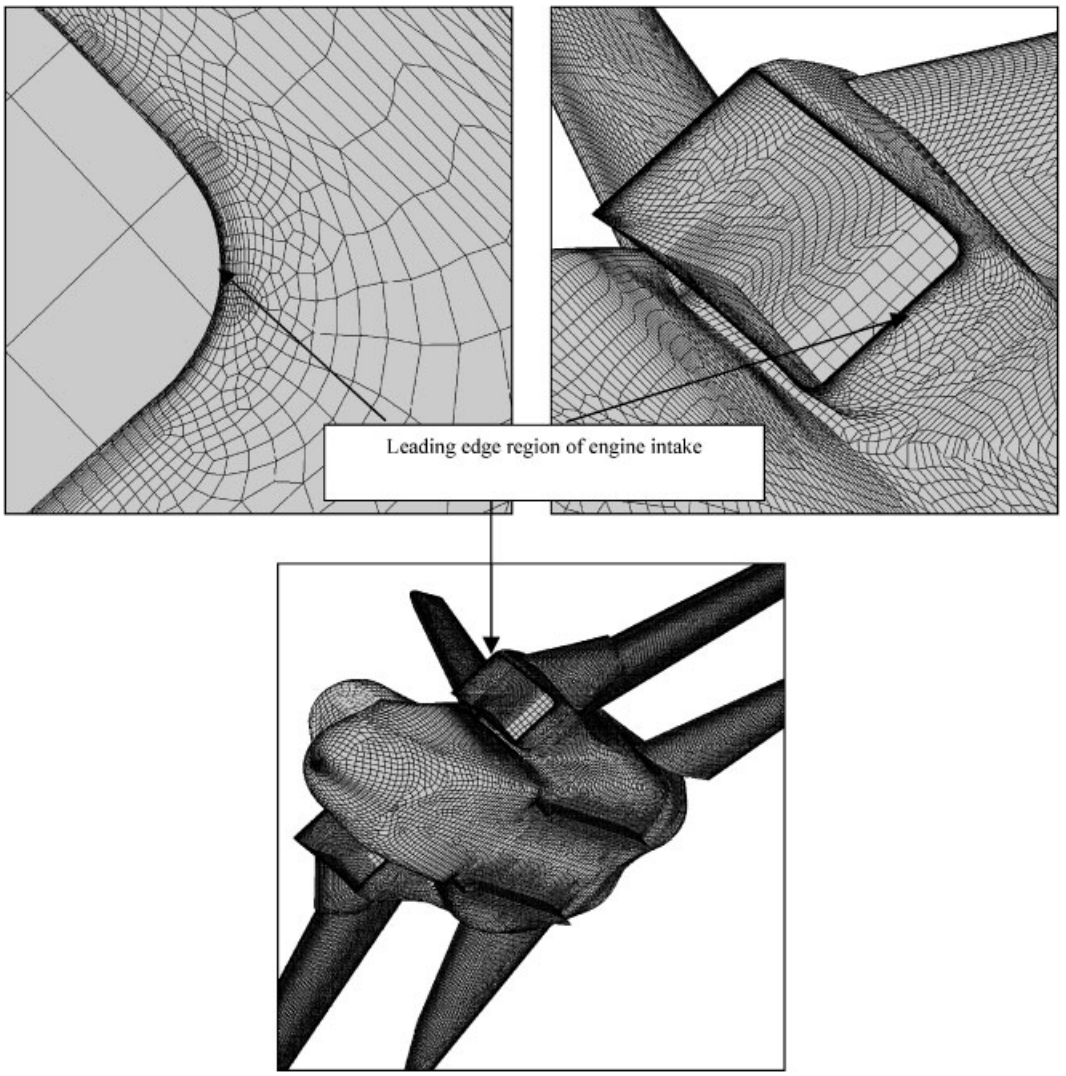

Figure 4. Global and close-up views of surface grid on Tornado aircraft demonstrating the generation of high-aspect ratio quadrilaterals via the use of anisotropic sources.

4. GENERATION OF THE NEAR-FIELD VOLUME MESH

Principal requirements of a near-field mesh generation technique for viscous flows are believed to be precise control of near-wall spacing and high mesh quality. The former of these is known to be important to the accuracy of simulations performed using differential turbulence models [4], whilst the latter has been found to be an important quality of a grid in validation studies [11]. Semi-structured methods, based on marching away from a pre-defined surface grid, meet both requirements. Marching-distance is defined easily and a regular topology for the

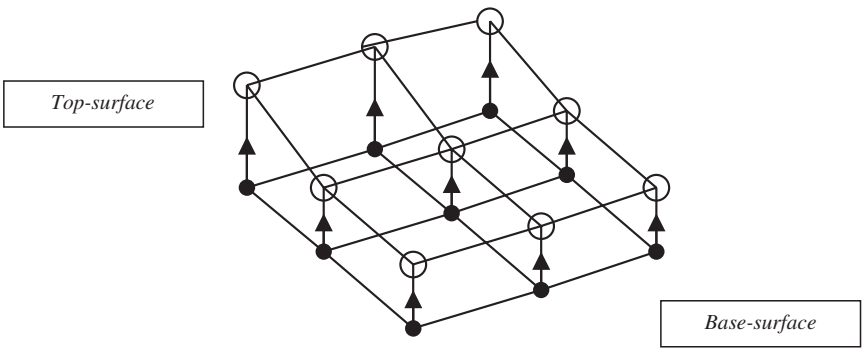

Figure 5. Schematic of Advancing-layer Approach to Grid Generation.

grid normal to the surface is ensured. A high degree of mesh orthogonality over the majority of the surface is guaranteed, also. The advancing-layer approach [17–19] is the most popular technique for generating such grids.

4.1. Introductory discussion on advancing-layer mesh generation

This section has three focal points which are: an outlining of the basic advancing-layer technique, a discussion on the limitations of the approach and an overview of the strategy followed to overcome these limitations.

4.1.1. The basic technique. The advancing-layer technique [17–19] produces a grid as a result of the recursive generation of a layer of grid and its subsequent placement on top of the previously generated layers. Each layer has a *base-surface* and a *top-surface*. The layer is initiated by a *base-surface* whose points and connectivity are known, the surface elements being either quadrilaterals or triangles. A predefined surface grid (see Figures 3 and 4) forms the *base-surface* for the first layer and this is projected out into the flow domain to attain the *top-surface* of layer 1 (see Figure 5). The *top-surface* for layer 1 is copied to be the *base-surface* for layer 2 and the layer growth is repeated until the complete domain has been covered. The process is viewed as an inflation of the initial surface grid (analogous to blowing up a balloon), since the surface topology is maintained in each layer whilst the shape of the *top-surface* tends towards a sphere as the grid is advanced.

There are two main aspects to the generation of a layer, namely the evaluation of a marching-direction and marching-distance. Each requires detailed attention to realize the benefits the approach offers. These topics are discussed in detail in Sections 4.2.3 and 4.2.4.

4.1.2. Limitations of method. Two principal problems have limited the use of the advancing-layer technique for forming meshes for modelling high Reynolds number flows over complex geometries, e.g. a fighter aircraft laden with stores, as depicted in Figure 3.

The first issue revolves around robustness and grid quality. In particular, the advancing-layer technique is renowned for encountering difficulties in marching away from:

- Concave boundaries (e.g. wing–fuselage junctions)
- Convex boundaries (e.g. leading edge of Tornado intake shown in Figure 4)

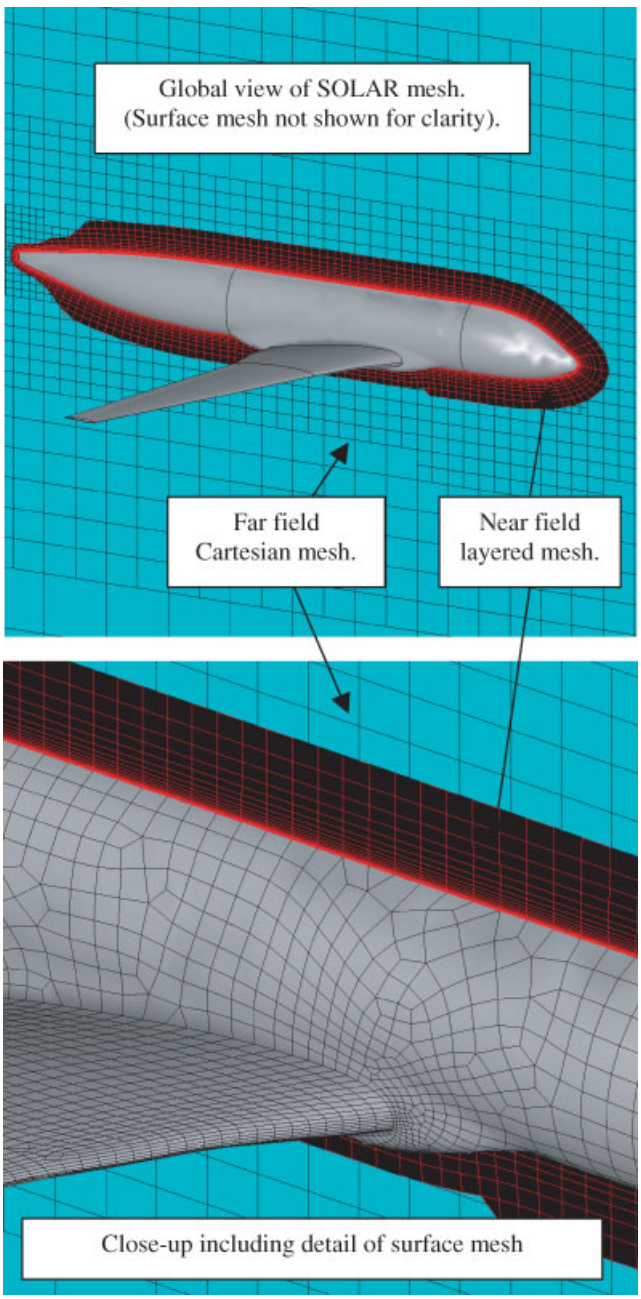

Plate 1. Global and close-up views of SOLAR mesh generated for civil wing–fuselage configuration.

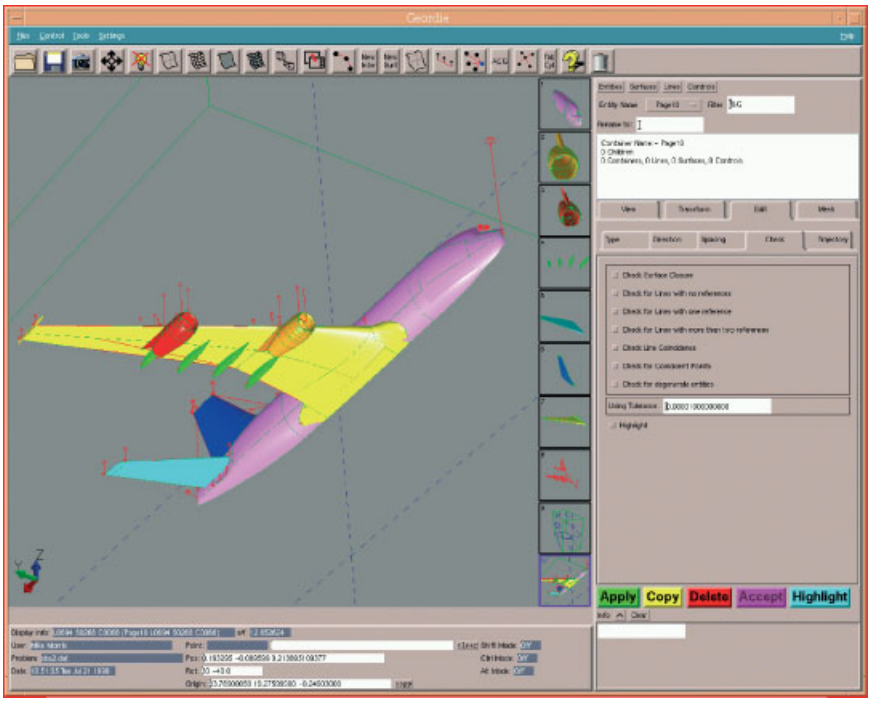

Plate 2. GUI interface showing geometry and positioning and extent of influence of sources.

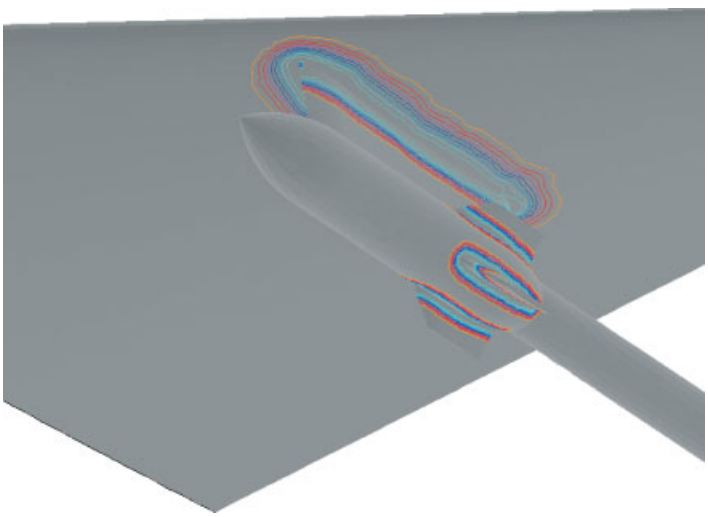

Plate 3. Sub-Surface of wing-pylon-store configuration that is used for smoothing normals.

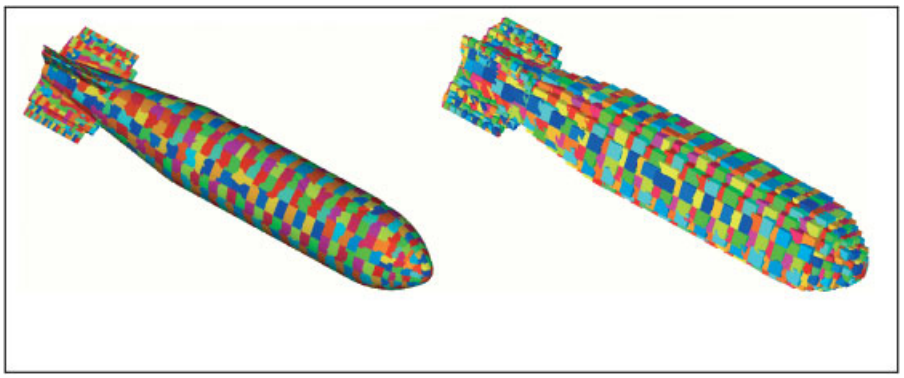

Plate 4. Grouped faces and bounding boxes for a single missile configuration.

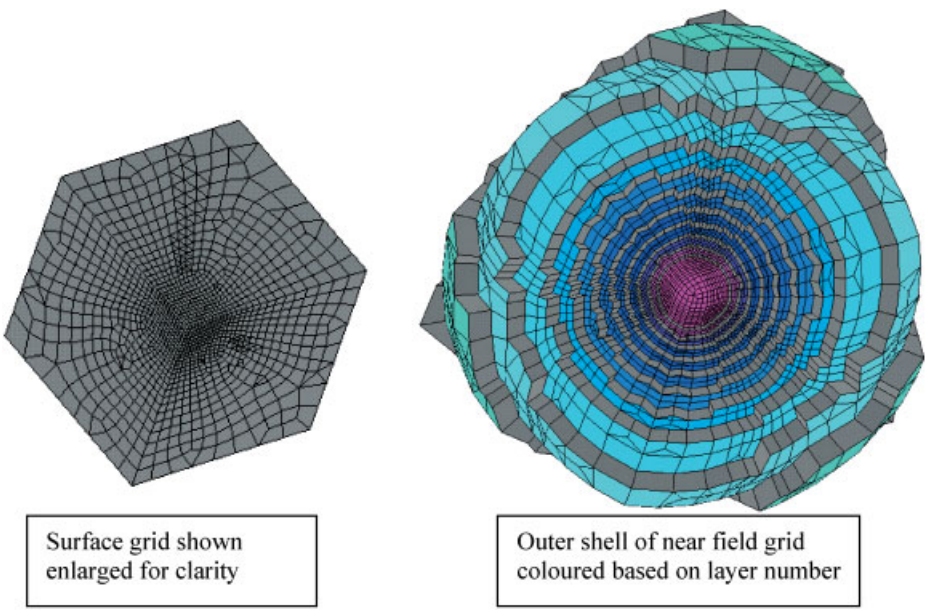

Plate 5. Surface grid and outer boundary of near field grid showing local layer termination based on achieving desired cell size.

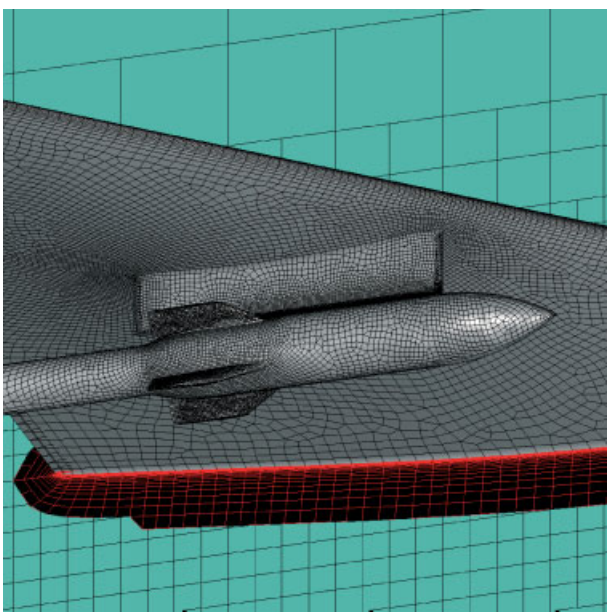

Plate 6. SOLAR mesh generated around a generic wing/pylon/store configuration.

- A combination of concave and convex boundaries (e.g. trailing edge of wing–fuselage junction.)

The second issue concerns the lack of applicability to configurations with:

- Disconnected surfaces (e.g. modelling released stores still in close proximity with the parent aircraft).
- Closely coupled surfaces (e.g. deployed high-lift devices)
- Internal geometries (e.g. intake ducts)

In addition to these limitations, the use of the two-stage 'layered then cut-cell Cartesian' approach to mesh generation within SOLAR places additional demands on the near-field mesh generator. For in the near/far-field interface region, the layer mesh generator must form cells with a height approximately matching the characteristic length of the cells the Cartesian generator will create.

Overcoming the aforementioned limitations, addressing the near/far-field interface constraint and developing robust algorithms for marching-distance and marching-direction evaluation are the main foci of the work covered in the remainder of Section 4 of this paper.

4.1.3. Strategy for overcoming limitations and addressing interface constraint. The two major limitations highlighted in the preceding discussion are overcome through the combination of a number of techniques. These are overviewed in this sub-section in a prelude to the discussion of the algorithmic details given in Sections 4.2.

4.1.3.1. Grid quality and method robustness. The importance of the evaluation of the marching-direction and marching-distance in enabling the advancing-layer technique to be used to march a reasonable distance away from a surface has already been touched on. These topics were given considerable attention by Chappell [13, 14] and have received further consideration in the current work, which has both applied the method to more demanding test cases and looked to reduce the user-input to the algorithms to a minimum.

However, the key thrust that has been followed to improving grid quality and method robustness is the use of element-collapsing and element-enriching within a layer, building on the use of element-collapsing for prismatic grid generation that was demonstrated by Chappell [14].

The principal role of element collapsing is to facilitate marching away from concave regions as illustrated for a simple two-dimensional case in Figure 6. In this case, the possibility of normal grid lines converging and ultimately intersecting can be removed since the collapsing allows some of the normal grid lines to terminate. Sensors are used to detect the convergence of normals within a layer and appropriate edges on the top of a layer are collapsed to a point.

In a converse manner, element-enrichment can be performed to improve the approaches' robustness in marching away from convex boundaries, as illustrated for a simple two-dimensional test case in Figure 7. In this case, the normal grid lines are diverging and the addition of points on the base edges of a layer reduces the possibility of creating large jumps in cell sizes for adjacent cells in a given layer. It also has the added benefit of enriching the mesh in potentially key aerodynamic regions.

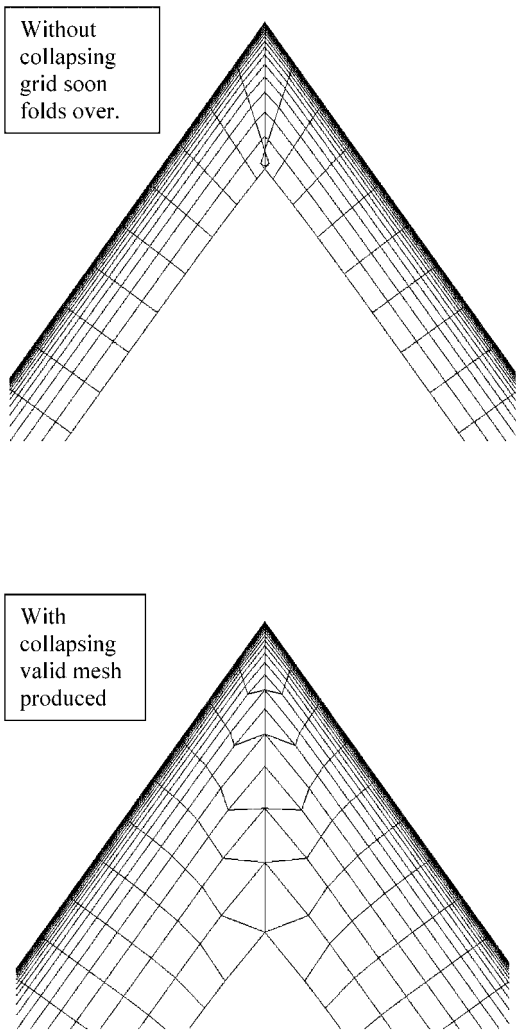

Figure 6. Mesh-collapsing allows marching approach to mesh concave boundaries (N.B same first cell height used for both grids).

Collapsing and enriching also have important secondary roles that include maintaining grid quality and helping to promote a smooth change in cell volume at the interface between the near and far-field grid.

A consequence of both element-collapsing and element-enriching strategies is that within a layer the elements' connectivity changes as the grid is grown away from the surface (this is in direct contrast to the standard advancing-layer approach where the same grid topology is maintained for each layer). Thus, the topology of each layer has the potential to be adjusted automatically to meet the particular requirements of a given layer. A key feature of unlocking

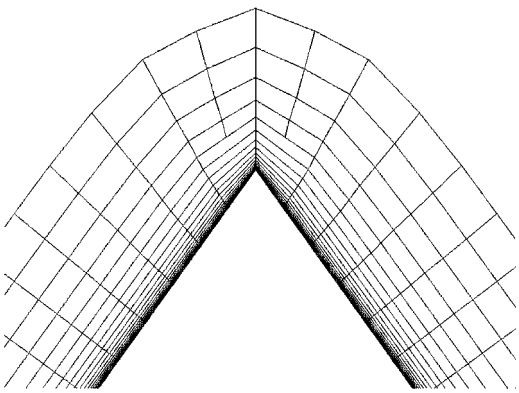

Figure 7. Example of mesh-enrichment.

this potential automation is the development of suitable sensors for invoking the collapsing and enriching operations, as discussed in Section 4.2.

4.1.3.2. Range of applicability of method. Even with both the most sophisticated techniques for marching-direction and marching-distance evaluation and the use of element-collapsing and element-enriching, the basic advancing-layer technique is never going to be able to mesh completely multiply connected domains. The topological limitations of the approach ensure this, since the intersection of layers grown from one set of surfaces with those grown from another set of surfaces cannot be prevented. Indeed, the same premise can be anticipated, if not proven, for very closely coupled, or annular, surfaces.

Fundamental extensions to the approach are therefore needed before it can be applied to general configurations. The route taken in this work is to develop techniques to:

- Detect that cells from within the current layer intersect the outer surface of the current grid.
- Remove intersecting cells, decrenellate the boundary and stop growing in these regions of the layer.
- Update the layer's *top-surface* so that the marching can continue from the remaining exposed faces.

These extensions are illustrated in Figure 8 and discussed in detail in Section 4.2.

4.1.3.3. Addressing the near-field/far-field interface constraint. In the conventional operation of the advancing-layer approach, a mesh is generated by marching over a number of layers until the complete domain is covered. In the mixed layered/Cartesian cut-cell approach taken here, the growth of the grid must terminate when the cells have a height that matches the characteristic length that the far-field mesh will have. A number of approaches to this have been investigated during the course of the work, including strategies involving specifying the total number of layers to grow and reducing locally the mesh growth rate once the desired height has been attained. The approach that has been found most satisfactory however is a combination of the techniques discussed in Section 4.2. This includes local mesh-enrichment

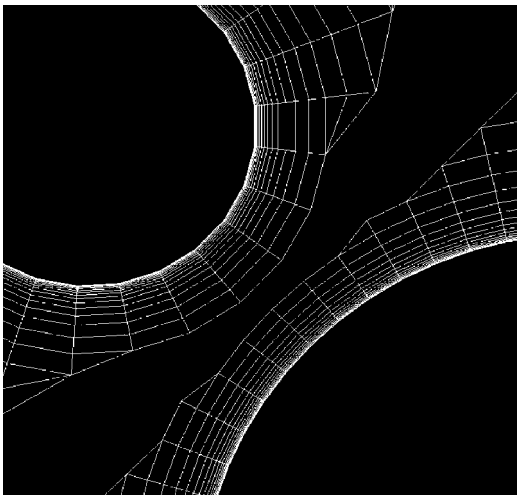

Figure 8. Local front stopping driven by detection of intersection of cells, followed by removal of cells and decrenellation of boundary.

and mesh-collapsing to achieve the desired lengths for edges that lie in the *base*/*top-surface* of a layer and local front stopping once the marching-distance matches the desired edge-length.

4.2. Improving grid quality, method robustness and range of applicability of the advancing-layer method

Since the advancing-layer approach involves the recursive generation of layers, it is useful to provide an overview of the order in which the mesh generation algorithms are applied within the generation of each layer, see Figure 9.

This provides a framework for introducing the techniques that have been developed to extend the robustness and applicability of the method.

Study of Figure 9 shows that:

1. *The-Layer* is initiated with a *base-surface*, which for layer 1 is the initial set of surface grids, and for layer n is the *top-surface* of layer $n-1$.
2. Sensors determine whether quadrilateral faces should be split or points should be added to the *base-surface*. In either case, the faces and edges of the *base-surface* are redefined and hanging-face data for inter-layer connectivity is created.
3. Marching-directions are evaluated for all nodes on the *base-surface*.
4. Marching-distances are determined for all nodes on the *base-surface*.
5. The *base-surface* is projected to give the *top-surface*, thereby creating the cells within the layer.
6. Tests are performed to determine if cells within the layer need to be removed, which include:

   (a) Face-edge intersection tests to detect whether cells in the layer intersect the current 'outer-shell' of the near field grid. These tests are sufficient to ensure that no other region of grid is intersected due to the marching nature of the method.

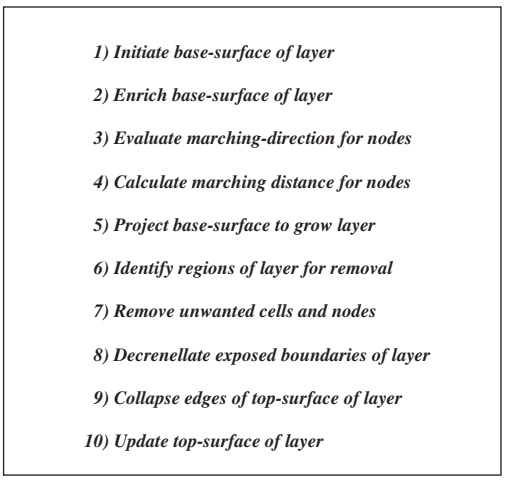

Figure 9. Order of operations in construction of layer.

(b) Cell height monitoring to terminate layer growth locally once the marching distance for a cell (height) exceeds the local characteristic lengths given by the background sources.
(c) Cell quality checks to ensure that cells with negative volumes or high skewness are identified.

7. If cells are flagged for removal, the data are used to determine which nodes and cells contained within the layer should be removed. The removal of these cells ensures that no further growth in these regions occurs in subsequent layers.
8. The process of decrenellating the boundary of the retained region is invoked. Once a layer has had cells removed from it, all subsequent layers will contain cells that need decrenellating.
9. Sensors determine whether edges should be collapsed on the *top-surface*, in which case the faces and nodes of the *top-surface* are redefined, as are the cells and their intra-layer connectivity.
10. Finally, the *top-surface* of the layer is updated in preparation for copying it to form the *base-surface* of subsequent layers.

The following sections of this paper describe the algorithms employed at each of these stages.

4.2.1. Initiation of the base-surface. The *base-surface* is initialized simply, either for the first layer via the input surface grid or for subsequent layers by copying the contents of the *top-surface* of the previous layer. The class that supports the *base-surface* contains information about the co-ordinates of nodes, constraints on node-normal directions, the nodes that form faces and the connectivity of faces.

4.2.2. Mesh enrichment. Mesh enrichment takes two forms. Firstly, the *base-surface* of the layer is analysed to determine whether any of the quadrilateral faces should be split in half to give two triangular faces. The principal motives for this are to remove poor quality

quadrilaterals (large internal angles) and to modify topological structures that restrict the validity of applying the mesh collapsing techniques. Secondly, nodes are added along edges of the *base-surface*. The motives here are threefold; to improve the robustness of the method in regions where the normals of adjacent nodes are rapidly diverging, to ensure a smooth change in cell volume at the interface between the near and far field grid and to promote grid structures that facilitate the application of the mesh collapsing techniques.

The mesh enrichment entails two basic operations on the given *base-surface* of a layer. The faces of the *base-surface* are redefined to account for the added nodes and edges under the strict constraint that all edges that existed in the initial *base-surface* are present in the refined *base-surface*, even if this is as the union of more than one edge. The resulting new *base-surface* faces will either match one-to-one or one-to-many with the initial *base-surface* (which is the *top-surface* of the previous layer). Where there is a one-to-many match, hanging-face data must be created for the inter-layer connectivity. The following sub-sections cover these topics.

4.2.2.1. Sensors for enriching mesh. The sensors that have been developed to identify that a quadrilateral face should be split into a pair of triangular faces are based on detecting:

- Two-point singularities, i.e. where two quadrilaterals abut each other on two of their faces. In such a case, the collapsing of a non-common edge for each quadrilateral would produce an invalid mesh because four faces (rather than two) would meet at a common edge.
- Large internal angles of quadrilaterals, i.e. $>150°$. This test removes poor quality quadrilaterals, in particular chevrons (a quadrilateral with an internal angle greater than $180°$), which are both difficult to march away from and a source of potential mesh induced errors in subsequent flow solutions.

The sensors that have been developed to identify that nodes should be added to the *base-surface* are based on detecting, for each edge in the mesh, whether:

- The difference in direction of the marching-vectors at the two end points exceeds a given tolerance.
- The current edge-length is a factor of 1.75 greater than the desired mesh length scale given by the background sources.
- Refining the current edge will create a grid structure that will facilitate the application of the mesh collapsing techniques.

As each edge is flagged for refinement, a node is added to the list of surface mesh nodes. Its position is determined and its global number stored in an edge-based array ready for its subsequent connectivity to the surface mesh.

4.2.2.2. Face enrichment following node addition. For each face in the initial *base-surface*, flags associated with the forming edges are examined to determine whether nodes have been added to enrich the grid. If this is the case, the face needs to be replaced in the *base-surface* by a number of new faces. The precise rebuilding of the faces depends on whether the initial face is a quadrilateral or triangle and the topology of the edges that have been enriched. Geometric tests are also used to determine best grid quality for scenarios where more than one permutation is possible. In certain situations, such as all four edges of a quadrilateral

being enriched, an additional node is added at the centre of the face (in the case cited, to allow the construction of four child quadrilaterals). Once the faces have been redefined, the edges and connectivity of the *base-surface* are re-determined.

4.2.2.3. Creating hanging face data. The cell-centred SOLAR flow solver is capable of treating the one-to-many abutment of cell faces, i.e. hanging faces, where a parent face is abutted by more than one child face. In this case, the parent faces lie on the *top-surface* of the previous layer and child faces lie on the *base-surface* of the current layer. For each parent face, the data that needs to be established is the number of child faces, the forming nodes of the child and parent faces and the cell numbers associated with the elements containing the child and parent faces. This data is readily determined, in part by storing the face of the *top-surface* of layer $n-1$ that each face of the *base-surface* of layer n abuts. Directly copying the *base-surface* of the current layer from the *top-surface* of the previous layer gives a simple initialization of this data, which is updated as either existing faces are subdivided to form new faces, or as cells are removed from the layer.

4.2.3. Calculation of marching-direction. The marching-direction needs to be evaluated as a unit vector for each node on the *base-surface*. This is done through the careful construction of a vector for each node, followed by a smoothing of vectors.

4.2.3.1. Calculation of initial normal vectors at nodes. The following algorithm was devised for calculating the marching-direction for each node:

- Loop over all the faces in a node's manifold. Group together approximately planer faces. This will be referred to as grouping 1.
- Loop over each group of planer faces and merge together groups which are approximately anti-parallel. This will be referred to as grouping 2.
- Calculate a normalized average of all the face normals within each group, giving a group normal vector.
- Calculate a normalized average of the group normals to give the marching-direction vector.

A necessary condition for an advancing-layer mesh to produce positive cells is that after a node has been projected on to the *top-surface* of the current layer, it has to remain visible from all of the nodes in its surrounding base-surface manifold. To ensure normal visibility the following algorithm, based on work presented in Reference [19], has been implemented.

- Loop over the manifold faces and determine which pair of faces intersects to form the most acute wedge, refer to these as faces f1 and f2.
- Constrain the marching vector to lay on the bisection plane of faces f1 and f2.

4.2.3.2. Smoothing of marching-direction. Once these algorithms have been applied to calculate a sensible marching vector for each node on the *base-surface*, a Laplacian smoothing operator is used to remove any discontinuities in the direction of the marching vectors. This is particularly important in concave regions where the discontinuities could lead to the mesh

folding. The formulation of the operator is [18]

$$V_{\mathrm{p}}^{t+1} = (1-\omega)\hat{V}_{\mathrm{p}}^{t} + \frac{\omega}{N_{\mathrm{p}}} \sum_{n=1}^{N_{\mathrm{p}}} \hat{V}_{n}^{t} \tag{2}$$

where $\hat{V}_{\mathrm{p}}$ is the unit surface vector for point p, $\hat{V}_n$ is the nth neighbouring normal vector of p, N_{p} is the number of neighbours, ω_{p} is relaxation parameter, and t is an iteration level. If the marching vector is required to lie in a given plane, constraints on the normal vectors are imposed both prior to and following the application of the smoothing operator.

For efficiency reasons the smoothing is split into two stages. Firstly, the following algorithm is implemented to identify the nodes in the vicinity of concave regions, where a significant amount of smoothing of normal vectors is required:

- Sets up a node based array, r, and initialize to zero.
- Set $r_{\mathrm{p}} = 1$ at all concave nodes, p, on the surface.
- Smooth the value of r over the surface.
- Construct a *sub-surface* made up of faces whose nodes all have a value of r greater than a given tolerance.
- Using r_{p} as the relaxation factor ω_{p}, smooth the marching-vectors over the *sub-surface*.
- Reintegrate the *sub-surface* with the original surface.

The Laplacian Smoothing operator is then applied for a few iterations over the whole surface using the following formulation for ω_{p}, the relaxation factor:

$$\omega_{\mathrm{p}} = \mathrm{MIN}(1.0, \mathrm{nl}/25) \tag{3}$$

where nl is the layer number.

This approach realizes the same benefits as global smoothing of the entire surface for a large number of iterations but with huge savings in computational cost. An example of a typical *sub-surface* for an aircraft is illustrated in Plate 3.

4.2.4. Marching-distance evaluation. The marching-distance for each *base-surface* node is usually calculated by applying an exponential function since this provides the required control of the normal-to-the-wall spacing of the grid. The exponential function can be tailored to achieve a specified first cell-height for the *base-surface* of the first layer, as well as allowing the cell volumes to grow in magnitude by a constant growth factor. The desired first cell-height is obtained by deriving values from the source distribution specified by the user within the GUI. Furthermore, these values can be globally scaled to allow the same distribution of desired cell-height to be used for a range of Reynolds numbers. Within SOLAR, users have the option to have a linear region of mesh spacing close to the wall, prior to the application of the exponential growth normal to the surface. It is believed this facility may benefit the accuracy of the prediction of quantities such as skin friction, although a thorough validation with the SOLAR flow solver has yet to be undertaken.

To increase the robustness of the advancing-layer method, the marching-distance is modified [13, 14] to smooth out regions of high curvature. The modification factor is calculated in two stages. Firstly, a factor is calculated that, when multiplied with the initial marching-distance, will ensure the local curvature of any concave region on the *top-surface* of the layer does not exceed that on the *base-surface* of the layer. Secondly, this factor is corrected so as to

reduce curvature over the *top-surface* of the layer. This process involves further increasing the factor in concave regions and decreasing it in convex regions. As with the marching-direction evaluation, the robustness of the method and quality of the resulting grids is improved by the application of a Laplacian smoothing filter.

4.2.5. Projection of base-surface to grow layer. Once a marching-distance and a unit magnitude marching-direction have been evaluated for all nodes, it is a straightforward task to multiply this data to produce the increments to the *base-surface* that are required to produce the *top-surface* of the layer.

4.2.6. Identification of regions for termination of local layer growth. As has already been discussed, there are a number of reasons for the requirement to terminate locally the growth of a layer. These include the removal of cells that intersect other cells (either in the current layer or previous layers), the removal of cells of poor quality (either negative volume or high skewness) and the removal of cells where the marching-distance exceeds the other characteristic length scales of the cell. The algorithms that are invoked to flag these cells are discussed in detail in this section.

4.2.6.1. Detecting layer intersection. The problem of detecting the intersection of a given layer with either the existing grid (which does not include the layer being generated), or itself, is potentially computationally intensive. The required effort can be reduced to testing the intersection of the set of edges that define the cells in the layer with each face that bounds the outer-shells of both the current grid and layer. The faces forming the *base-surface* of the layer can be considered equivalent to the faces forming the *top-surface* of the grid (Of course, exact equivalence is violated by the possibility of enrichment having occurred in the *base-surface* of the layer). Hence, in addition to the faces bounding the layer, the only faces of the grid that need to be considered are the exposed faces introduced by the layer-stopping and decrenellation processes applied to previous layers (see Section 4.2.7). These faces are easily identified within the grid, since a particular boundary condition is associated with them as part of the process of identifying the faces that the far-field grid will have to be cut to match. Adding the bounding side faces and faces from the *top-surface* of the layer to the set of grid faces completes the definition that the outer-shell of the grid would have if the layer had already been added to it.

The faces in this outer shell are tested for intersection with vectors defined by connecting nodes in the *base-surface* of the layer with their corresponding ones in the *top-surface*, i.e. the intersection vectors are effectively along the smoothed marching vectors. However, they are given a magnitude that exceeds the local marching distance (multiplied by 2.0, currently) so that if a layer is getting close to intersecting the outer-shell, then the cells are flagged as intersecting. This avoids situations where the gap between the grown cells is very small, causing a large jump in cell size between the near and far-field mesh.

The same faces are then tested for intersection with the vectors formed from their own edges.

If, and only if, either of the above set of edge/face intersection tests identify a region of intersection, then the faces of the *base-surface* of the layer are tested for intersection with the 'through the layer' edge vectors that were used in the first set of tests.

Quadrilateral faces within the grid are subdivided into triangles for the intersection testing, split along the diagonal that produces the most convex, and therefore most likely to intersect, surface.

The priority in developing the algorithm for this section of the system is efficiency, as the worst case cost for the algorithm is $O(\text{nNode}^{*}n\text{Face})$. Developers of advancing-front type unstructured grid generators [8] have used alternating-digital-tree structures (ADT) for explicit edge–face intersection calculations, however an ADT is not the most appropriate structure for the specific problem here. For an ADT, calculation of edge–face intersections is rapid; however, tree set up time can be prohibitive. In Advancing-Front calculations, the set up time is ameliorated due to its dynamic nature; for the advancing-layer approach, a new tree needs to be set up for each layer, thus magnifying the set up cost.

Introducing relatively cheap tests that nullify the need to check explicitly for edge–face intersection can substantially reduce this cost. The approach that has been used is based on one that has been exploited successfully in the development of a rapid hidden line removal algorithm for vector postscript picture production. The principal is, to employ a recursive-bisection approach to partition the outer-shell surface into sub-domains, each containing approximately equal numbers of surface faces. The geometric extent of these sub-domains is then evaluated and stored as bounding boxes, see Plate 4.

By comparing the co-ordinates of the end points of an edge with those of the bounding box of a domain, large numbers of tests on possible edge–face intersections are avoided.

In practice, it is important to get a balance between the amount of CPU time spent performing the edge/bounding-box tests (i.e. the time spent avoiding edge–face intersection checking) and the time spent performing edge–face intersection tests. This can be achieved by defining the total CPU cost to be:

$$\text{Cost} = n\text{edges} * (a * k1 * x + b * k2/x) \tag{4}$$

where $n$edges is the number of edges, $k1$ is the number of domains an edge is likely to intersect, $k2$ is the number of faces in the surface, and x is the number of faces in a domain. a and b are constants that represent the computational effort for the two sets of tests, respectively. Thus, the cost can be minimized by dynamically changing the number of faces contained in a domain to be:

$$x = \text{sqrt}(b * k2/a * k1) \tag{5}$$

To optimize the intersection calculation further, the tree can be stored between iterations. This allows the triangle-domain mapping of the tree to be re-used for each layer (if the number of triangles and their connectivity remain constant). The new triangle locations are written to the tree and the limits of the domain are redefined.

The data created by the intersection routine stores the cell number that each edge intersects.

4.2.6.2. Termination based on achieving desired edge length of cell. Layer growth is locally terminated once the average marching distance, for a given cell, becomes greater than the desired mesh length scale given by the background sources. The motivation for this approach is two-fold. Firstly, it ensures a reasonable match in cell size and aspect ratio between the near-field and far-field meshes. Secondly, it minimizes the number of cells in the near-field mesh thus improving the efficiency of the subsequent flow calculation. Plate 5 demonstrates

the application of this approach to a cube with a surface mesh that is clustered towards a single vertex.

4.2.6.3. Termination based on poor mesh quality. Mesh quality is assessed currently either on mesh volume or cell skewness, the latter assessed simply via the dot product of the unit normal vectors to the *top* and *base* face of a cell.

4.2.7. Removal of nodes and cells from a layer. Once the basic data for the cells that are to be removed from a layer has been defined, certain topological tests are performed on the remaining nodes and faces of the layer to ensure that a valid, consistent set of data is created. The tests, which are performed iteratively on the *top-surface* of the layer until no more nodes or cells are added to the list, are to flag for removal:

- Faces that have a forming node flagged for removal.
- Faces that are not connected to at least two other faces. This has the effect of smoothing off the boundary of the retained region.
- Faces that contain nodes that are part of two connected contours (i.e. the node at the centre of a figure '8'). Such structures are not allowed due to the data structures used in both the near and far field mesh generators.
- Nodes that are not part of the retained faces of the *base-*/*top-surface*.
- 'Brothers and sisters' of child hanging-faces that have already been identified for removal.
- Any quad that has either of its diagonally opposite corners on the boundary of the domain. This is to prevent having to decrenellate (see later) this particular type of element.

Overall then, after these various topological tests have converged for the *top-surface*, there is the potential for a significantly larger region of cells to be flagged for removal than that identified by the tests outlined in the Section 4.2.6.

The cells and nodes are then actually removed from the layer. The new boundary nodes are then set and the *base-surface* and *top-surface* of the layer are redefined in terms of the retained nodes and faces. The cell list and intra-layer connectivity are recreated and all node and face based data that has been established for use in the subsequent collapsing algorithms (see Section 4.2.9) is appropriately sifted. During all the renumbering processes, the various mappings of data need to be stored to manage the correct addition of the layer to the existing grid.

4.2.8. Decrenellation of cells in layer. There are potential drawbacks with adopting a strategy by which cells and nodes are removed from a layer. The shell that bounds the layered grid is highly crenellated. Furthermore, there is the potential both for cells of significantly different aspect ratios to be in close proximity to each other and for highly anisotropic faces of the layered mesh to abut the far field mesh. Such situations would lead to the Cartesian cut-cell far field method producing highly irregular polyhedra. Also, there would be a lack of smoothness in the variation of cell mesh-quality and mesh-property measures, which in turn could be expected to affect both the robustness and accuracy of the subsequent flow solution. These undesirable features in the mesh can be removed by smoothing off the jagged boundary produced by the local stopping of the mesh, a process that is referred to here as mesh decrenellation.

There are two possible approaches that could be taken to mesh decrenellation. One would be to generate the complete near field grid and then to smooth off the boundary. The alternative is to smooth off each layer, as it is grown. The latter approach has been favoured based on mesh quality considerations.

In considering how best to perform the decrenellation process, it is very attractive to consider the conceptually simple idea of collapsing the vertical edges of the faces that bound the region of the layer that has been removed. The drawback with this approach, for the work reported here, comes purely from an implementation point of view, given the structure of the code at the time of developing the decrenellation capability. In particular, such an approach would lead to the same nodes lying in both the *base-surface* and the *top-surface* of the layer, which in turn would have a significant impact on a number of existing algorithms. Instead, the approach taken, which is described below, is based on the deletion of certain nodes on the *top-surface* of the layer and the subsequent rebuilding of the affected faces and elements.

4.2.8.1. Checking the validity of the top-surface after decrenellation. Just as checks have to be performed to verify that the basic *top-surface* is valid after elements have been removed, then the topological properties of the *top-surface* that will exist after the decrenellation process has been invoked also needs to be verified. This is done prior to the decrenellation process itself. The current *top-surface* is simply copied to a temporary surface and the boundary conditions set to identify the boundary which the *top-surface* would have if all the presently flagged faces and nodes were removed. The tests that are performed to check the integrity of the surface are performed at the end of each iterative loop of the tests described in Section 4.2.7. They are:

- If any faces have all nodes on the new boundary of the grid, they are flagged for removal.
- If an interior edge of the surface has faces either side of it that will be removed by the decrenellation process, then the edge itself will form a ridge on the outer-shell. In such cases, both faces are flagged for removal.
- If an interior node of the surface is not connected by an edge to any other interior node of the surface, then the decrenellation process would leave an isolated node that is not connected to any face. Subsequent algorithms would break down if such situations occur and hence the node is flagged for removal.
- If an interior node of the surface is connected to two or more distinct groups of faces that will be removed by the decrenellation process, then the node is flagged for removal. In this case, the removal is to avoid the creation of non-smooth ridges and the creation of a post-decrenellation *top-surface* that has two contours meeting at a node.

4.2.8.2. Strategy for decrenellation. To support the explanation of the decrenellation process, the simple example of growing a mesh away from a coarse, regular surface mesh on a flat plate is considered. Assume that a single node in the middle of one of the *top-surface* of layer n has been flagged for removal. Based on the earlier discussion, the four surrounding elements (hexahedra) are then identified for removal.

The first stage of the decrenellation process is to identify which of the remaining hexahedra have either two or six nodes on the side boundary of the retained volume. Each of these hexahedra is split into two prisms, with the diagonal cut always chosen to connect between nodes that are on the boundary and nodes that are not. This in turn requires the intra-layer

and inter-layer connectivity data to be updated. This topological action greatly simplifies the subsequent decrenellation process.

The next stage is to flag for removal the nodes that lie on the *top-surface* and form part of the boundary of the retained region. In turn, the faces containing these nodes are then also flagged for removal from the *top-surface*, followed by the reconstruction of a valid *top-surface*.

Given a valid *base-surface* and *top-surface*, it is then possible to redefine the cells in the grid to account for the deleted nodes and faces. Cells that have not had any nodes removed simply need to have the *top-face* nodes renumbered to account for the removed nodes. The remaining cells need to be rebuilt from the retained, renumbered nodes. The rebuilding strategy is straightforward, based on knowledge of how many nodes lie on the base of the cell and how many nodes are retained in the *top-surface*.

A more demanding task is to update the intra-layer connectivity that defines the face-based connectivity of the mesh used by the flow solver. Again, this involves renumbering of existing data and a number of alternative scenarios need to be considered. Also required is the creation of additional connectivity data arising from the decrenellation of prismatic elements to pyramids that have a triangular base and quadrilateral side face that abuts an internal element of the layer. This connectivity data relates to the triangular top-face of the pyramid, which will be part of the outer shell of the near-field grid.

4.2.9. Mesh collapsing for concave boundaries. As described earlier, it has been found to be critical to allow the mesh to coarsen as layers are advanced away from surfaces with either high concave curvature or slope-discontinuities that give rise to neighbouring nodes having rapidly converging normals.

Collapsing is performed by developing sensors that firstly identify regions and then reduce edges of the mesh to points. Three basic operations are performed on the *top-surface* of a layer. The edges of the *top-surface* are analysed to determine which need to be collapsed and a certain priority associated with the edge depending upon how important the collapsing is deemed to be. The edges of the grid are then collapsed in turn, based on their position within a heap data structure, with checks operating to ensure that invalid collapses are not allowed. Finally, the prisms and hexahedra within the layer are redefined as alternative elements following the edge collapsing.

In practice, collapsing of edges can only be applied to one edge of a polygon at a time. It has therefore been necessary to develop the algorithms to enable the repeated application of the collapsing sensors and mechanisms over a number of iterations (typically four) for each layer to ensure that all the required collapses are performed.

4.2.9.1. Sensors for collapsing mesh. The primary function of the edge collapsing is to prevent the normal grid lines from crossing over as the mesh is projected away from a surface that has either concave-curvature or slope-discontinuities. The edge collapsing also has two secondary functions, firstly to improve mesh quality and secondly to facilitate the generation of approximately unit aspect ratio cells within the final layer of the mesh. The requirement for having unit aspect ratio cells within the final layer is to ensure a smooth transition in cell size between the near and far-field meshes. To give the edge collapsing this range of functionality, three sensors have been developed to determine which edges on the current surface require collapsing.

The first sensor is based on predicting the number of layers, nl, for which each face can be projected until the normal grid lines will eventually crossover. If, for a given face, the normal grid lines are not predicted to cross over within a pre-specified number of layers, then nl is set to zero. If nl takes nonzero values at the faces on either side of a given edge, then that edge is flagged for collapsing. The weighting given to this collapse is equal to the minimum, over the two faces, of $(1 + 1/\mathrm{nl})$. If nl is nonzero at an isolated face, then the implication is that it is a quadrilateral face with converging marching vectors emanating from disconnected nodes. If this is the case then the shortest edge of this face and the edge opposite are both flagged for collapsing.

The second sensor is based on monitoring the change in aspect ratio of each cell face, before and after the surface has been projected to form a dummy *top-surface* for the subsequent layer. If there is a predicted increase in aspect ratio for the two faces either side of a given edge then that edge is flagged up for collapsing. The weighting given to this collapse is based on the average increase in aspect ratio for the two faces.

The third sensor is aimed at achieving unit aspect ratio cells in the final grid layer. To this end, a test has been devised which is based on the ratio

$$\mathrm{el}/((\mathrm{nl}/\mathrm{total\_nl}) * 1.5 * \mathrm{md}) \tag{6}$$

where, el is edge length, nl is the current layer number, total_nl is the total number of layers to be generated and md is the marching-distance. If, for a given edge this ratio takes a value greater than one, then the edge is flagged for collapsing with weighting equal to the value of the ratio.

4.2.9.2. Edge-collapsing. The edge collapsing is applied to the edges of the *top-surface* of a layer, with the edges taken in turn from the heap structure. The collapsing is performed by operations on the edge's two end nodes, repositioning one node and removing the other from the grid. If neither, or both, of the end nodes of an edge lie on the boundary of the domain, the edge is collapsed by positioning the retained node at the bisector of the edge. If only one of the end nodes lies on the boundary of the domain, its position is chosen for the retained node to maintain boundary integrity. The collapsing of an edge is only accepted if the elements in the layer maintain a positive volume and the normal grid lines do not become too skewed.

The power of the edge-collapsing is illustrated in Figure 10 where a grid is grown for 50 layers away from a missile, without using any of the local layer termination techniques described earlier. It is clear to see from this figure how the topology of the *top-surface* of layers varies as the grid is grown.

4.2.10. Updating the top-surface of the layer. Following the modifications to the *top-surface* of a layer that result from edge collapsing, the data describing the faces and cells internal to a layer needs to be updated. Specifically, the faces that connect the *bottom-surface* to the *top-surface* of a layer are no longer simply quadrilaterals, but may be triangles. Equally, the *top-faces* of elements that were either initially triangles or quadrilaterals could now be lines or points.

Once the *top-surface* has been redefined, the definition of the layer is complete. The layer can now be added to the grid and the *top-surface* copied to become the *base-surface*

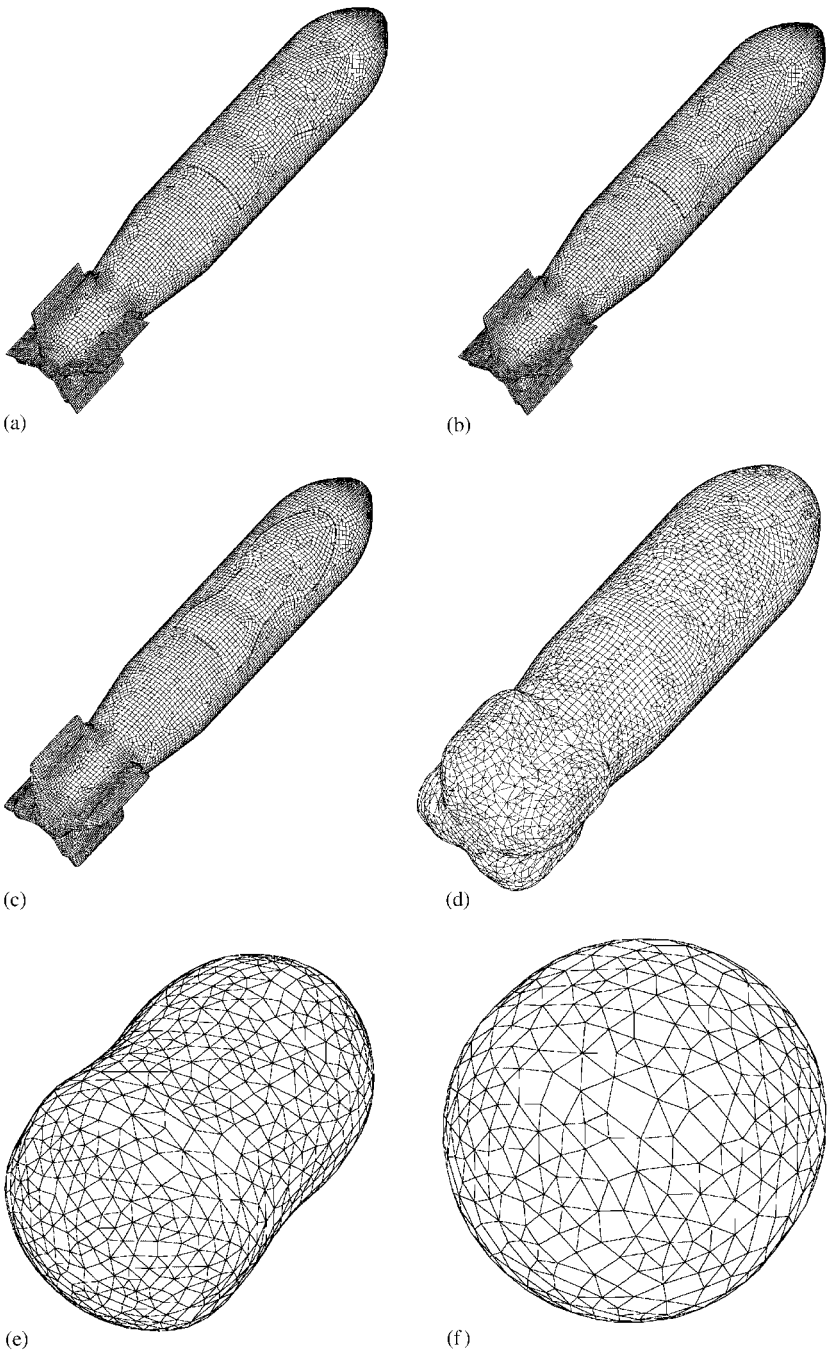

Figure 10. (a) Surface mesh on cruciform store; (b) *Top-surface* of layer 10; (c) *Top-surface* of layer 20; (d) *Top-surface* of layer 30; (e) *Top-surface* of layer 40; (f) *Top-surface* of layer 50.

of the next layer. This process continues until the generation of the near field grid is completed.

4.3. Data structures employed

To ensure that the layered grid generation is efficient in terms of both execution times and memory requirements, effective data structures [30] must be employed to store and manipulate data. The data structures used throughout the algorithms described in Section 4.2 include static lists, stacks, heap sorts, trees, dynamic-arrays and dynamic-lists.

5. FAR-FIELD MESH GENERATION

The far-field mesh is formed in two main stages. Initially a Cartesian mesh of varying density is constructed to cover the whole domain, with spacing locally determined from the same user-specified sources that control the surface mesh density. The cells that lie completely interior to the outer-boundary of the near-field grid are then identified and removed from the grid and the Cartesian cells that intersect this boundary are cut to conform to it.

5.1. Generation of Cartesian mesh

The Cartesian region of the far field mesh is initially generated by splitting the overall dimensions of the box bounding the flow domain into a number of regular intervals. This initial grid is then refined based on the strengths of sources distributed within the mesh. These sources are typically distributed near to the geometric surfaces since they are used to influence the density of the surface meshes. The sources define a desired edge-length for the grid and the Cartesian mesh is refined automatically to achieve cells with the desired local edge-length.

5.2. Construction of Cartesian cut cells

All cells of the Cartesian mesh that either lie completely within the configuration surface or the near-field mesh are removed from the grid. Of the remaining cells, some intersect the outer-boundary of the near-field mesh. These cells need to be cut to conform to the planes defined by the faces on the outer-boundary of the near-field mesh and the polygonal representation of the cell reconstructed to reflect this. If the cutting process creates very small cells, then they are removed by merging them with a neighbouring larger cell to form one cell.

Figure 11 shows a generic cut cell produced by a part of the outer-boundary of the near-field grid intersecting a Cartesian cell. For illustrative purposes, the outer boundary shown is composed of triangles only. In this case the lower part of the cube lies outside the boundary of the near-field grid so the upper part is discarded and the remaining cut cell consists of eight faces: the bottom face in its entirety; the four side faces made up of edges of the cut-cell and line segments formed by the intersection of the near-field boundary faces with the side-faces of the cube; finally the polygons formed where the cube clips the outer boundary faces. Construction of polygons such as abcd is easily implemented using a technique common in computer graphics applications, the Sutherland-Hodgman clipping algorithm [20]. The resulting clipped polygon is generated as a list of vertices that are naturally ordered in the same sense as

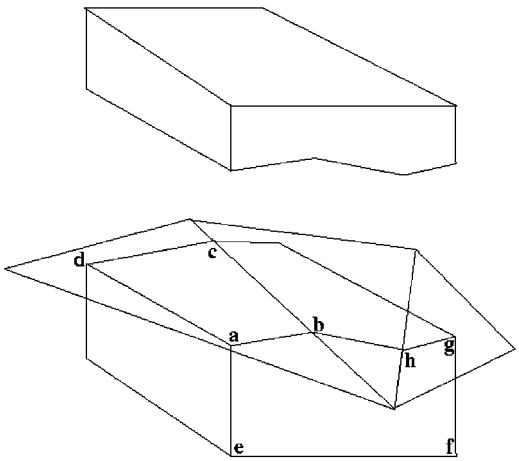

Figure 11. Generic cut-cell.

the vertices of the boundary face. Connectivity of the resulting set of clipped polygons takes the form of a so-called winged-edge data structure, which relates edges to faces and adjacent edges in a way that greatly facilitates edge and face traversal. To complete the creation of the cut-cell, the faces such as abhgfe, in Figure 11, must be built by connecting edges of clipped polygons to edges of the cube.

The complete cut cell construction thus consists, in general, of an arbitrary number of faces, each of which has an arbitrary number of edges. The faces must be formed into a list associated with each cut cell and account must be taken of the general case where a Cartesian cell can be cut to form more than one cut cell.

Figure 12 shows an instance of this that has been constructed to be easy to visualize. Where the near-field boundary cuts the cell, it consists of quadrilaterals that form a 'ridge' as shown. The part of the cut-cell lying outside the boundary consists of two prisms. The cut-cell construction algorithm involves traversing the list of edges generated by the cutting process and if untraversed edges remain in this list after a closed polyhedron has been constructed then a further cut-cell is constructed.

5.3. Removal of small cells

Arbitrarily small cells may be generated by the cutting process leading to small time-steps, hence flow solution convergence problems, and inaccuracies arising from large differences in size between adjacent cells. A simple cell merging procedure is implemented to overcome these difficulties.

Figure 13 shows a 'typical' scenario. By way of illustration, the intersection of the near-field boundary and the two Cartesian cells is defined by the quadrilaterals ijqr and turq. The volume of prism-shaped cut-cell C is smaller than the specified minimum so it is merged with cut cell A. Cut cell A is chosen, as it possesses the largest shared face with cell C. This shared face is removed from the global face list and the other faces tagged as belonging to cell C are re-tagged as belonging to cell A, thus completing the merged cell.

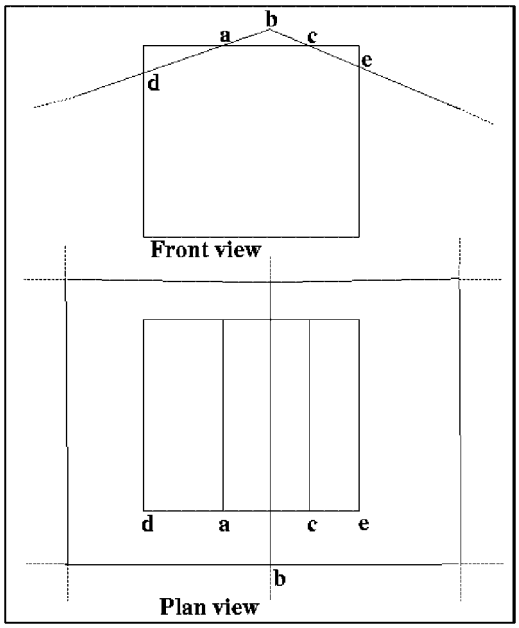

Figure 12. Creation of more than one cut-cell.

6. EXAMPLES

The mesh generation capability described has been developed for use in a wide range of application fields, including power plant installation and store carriage and release. The examples discussed below cover the areas mentioned.

6.1. Mesh for research civil aircraft

The requirement to simulate routinely the flow over complete aircraft such as the civil aircraft shown in Figure 14(a) is one of the main requirements of the SOLAR system. The configuration was described by 45 CAD surfaces, for which the bounding point distributions were derived from sources, the strength of which were chosen to promote suitable mesh density in regions of either high curvature or particular interest. Meshes were generated on each surface, and displayed, and various measures of mesh quality cycled through before each was accepted. On occasions, when the grid was not quite as desired, perhaps either because the mesh density varied too quickly or greater anisotropy was desired, the parameters associated with particular sources were modified and the mesh regenerated. All this was performed interactively via the GUI interface.

Figures 14(a) and 15(a) show the surface mesh generated for the research civil aircraft. In this case, the prime complexity in the generation of a volume mesh comes from the topology of the pylon and its intersection with, and abutment to, parts of the wing and nacelle. This complexity has led previously to significant manpower being invested in the development of block-topologies to generate block-structured grids for viscous flow modelling

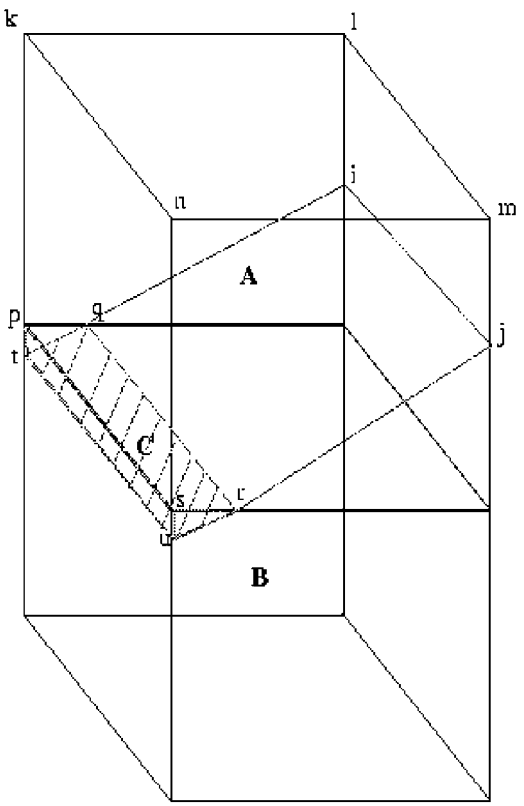

Figure 13. Removal of small cells from grid.

around this configuration. Furthermore, notable additional manpower was needed for each significant geometric variation occurring within the study of the integrated design of the wing-pylon and nacelle. In contrast, the technology discussed in this paper has allowed alternative builds of this configuration to be addressed in rapid succession.

Figures 14(b) and 15(b) show views of the outer shell of the near field volume mesh generated around this configuration. The figures serve to indicate the extent of the growth of the near field mesh, the overall quality of the volume mesh and the smoothness of the interface between the near and far field mesh. In Figure 16, the overall percentage of each layer of cells that are hexahedra is plotted. It can be seen that this starts off at more than 99%, with hardly any deterioration over the first 20 layers. Then, with the marching-distance getting ever larger, the tendency for grid lines to converge in junction regions increases, leading to the sensors switching on the edge-collapsing routines. This leads to the quadrilateral faces in the *top-surface* of the layers either being modified to triangles or compressed to lines or points. Around layer 30, growth away from regions with tightly spaced cells will begin to terminate because the mesh cells will have reached unit aspect ratio. The decrenellation process will be active at this point, which is a source for some of the hexahedra getting converted into prisms, from which subsequent prisms will grow. Hence, the percentage of cells that are hexahedra

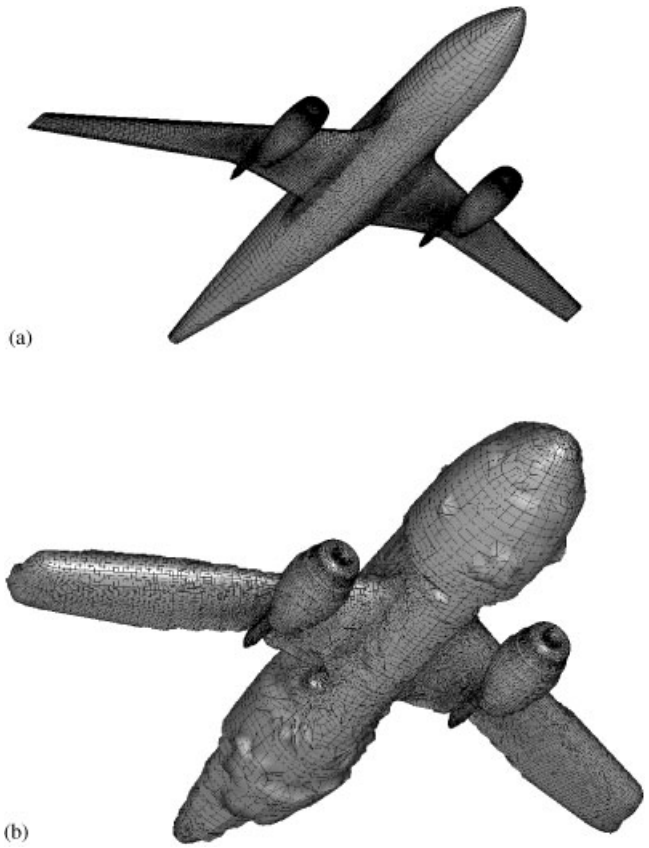

Figure 14. (a) Surface grid on civil aircraft; (b) The outer shell of the near field volume grid.

drops off quite quickly until, by the terminal layer (41), only about 2/3rds remain hexahedra. Overall though, in excess of 98% of the cells in the near field mesh are hexahedra.

A mesh with the same point density, but decomposed into tetrahedra, would contain about a factor of 5 more cells, with notable implications for the flow solver in terms of additional memory requirements, computational effort per time step and convergence rates. Thus, the idea that is promoted in this paper of using hybrid grids composed primarily of hexahedra offers potentially significant cost reductions over completely unstructured approaches.

6.2. Store carriage and release

An important application area for the SOLAR CFD system is that of store carriage and release. For general application in this area, the volume mesh generator needs to be able to generate automatically non-overlapping meshes around configurations with closely coupled surfaces, such as a store in carriage under a pylon. Test cases are presented in Sections 6.2.1 and 6.2.2 that demonstrate the meeting of this requirement.

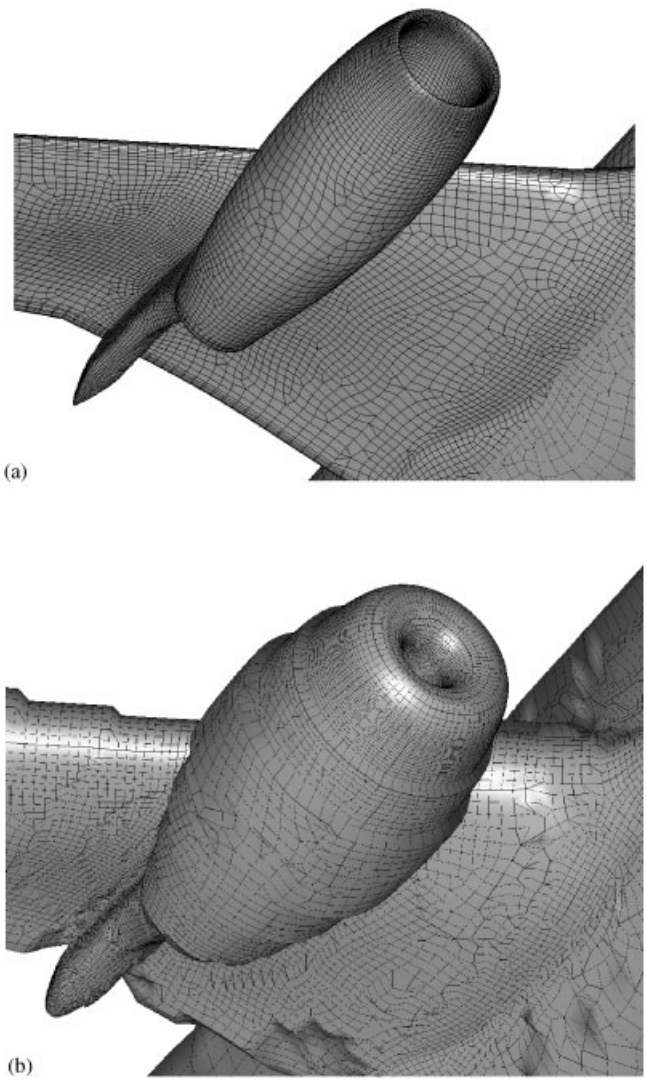

Figure 15. (a) A close up of the mesh on the nacelle; (b) A close up of the outer shell of the near field mesh in the vicinity of the nacelle.

6.2.1. Generic wing/pylon/store configuration. Plate 6 shows the complete surface mesh generated for a generic wing/pylon/store configuration. As discussed above, the major challenge to the volume meshing capability for this type of configuration is the small gap region between the store and the underside of the pylon. The ability of the method to treat this region can be judged by viewing constant cuts through the volume mesh, as shown in Figures 17–19.

Here, it can be deduced that the local-layer growth terminates based on the detection of overlapping regions of mesh rather than on achieving a desired marching-distance. The

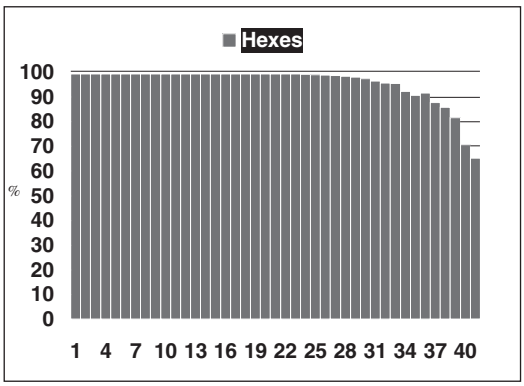

Figure 16. Percentage of elements per layer that are hexahedra.

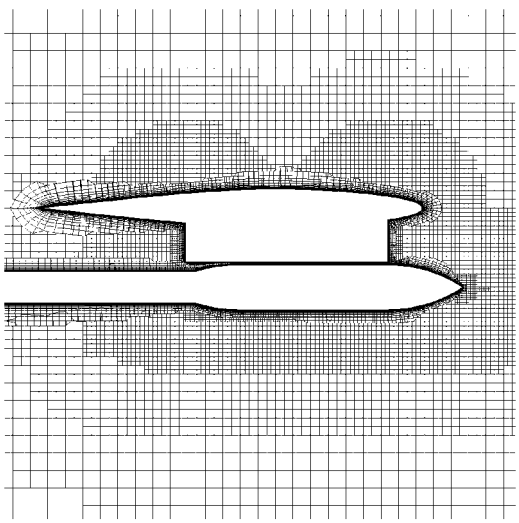

Figure 17. A cut of constant y though the SOLAR mesh.

overall quality of the mesh is good, with the cut-cell mesh matching well to the inner grid.

6.2.2. BL755 missile released from a Tornado. The geometry model that was used for a Tornado/bl755 configuration included a description of the aircraft's intake and the diverter that is positioned between the intake and the fuselage. A view of the mesh created around this configuration is presented in Figure 20. For the volume mesh generation the most challenging aspect of this configuration is the diverter region. This is due to the combination of the concave junction regions, where the diverter is connected to the intake and fuselage, and the proximity of the inboard side of the intake to the fuselage. A cut of constant x through the

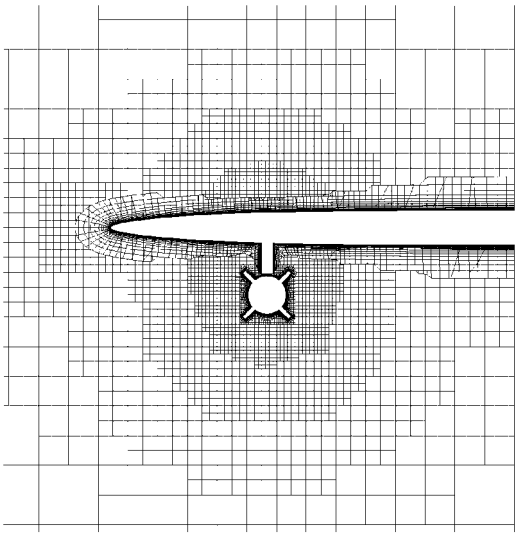

Figure 18. A cut of constant x though the SOLAR mesh.

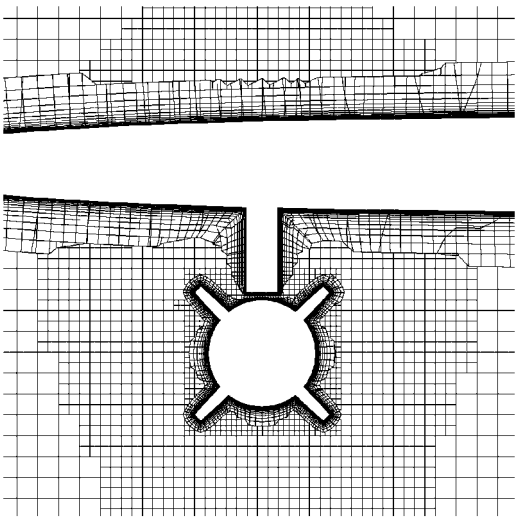

Figure 19. A close up view of the cut of constant x through the SOLAR mesh.

volume mesh in this region is presented in Figures 21(a) and 21(b). Note that the intersection of the near-field mesh is automatically prevented by the local termination of the marching process. Figure 22 illustrates a cut of constant y through the Tornado fuselage and the bl755 store. Overall the mesh quality is good for what is a challenging test case.

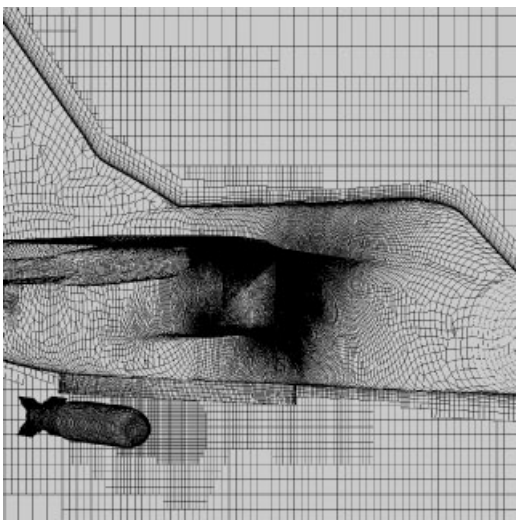

Figure 20. SOLAR mesh generated around the Tornado/bl755 configuration.

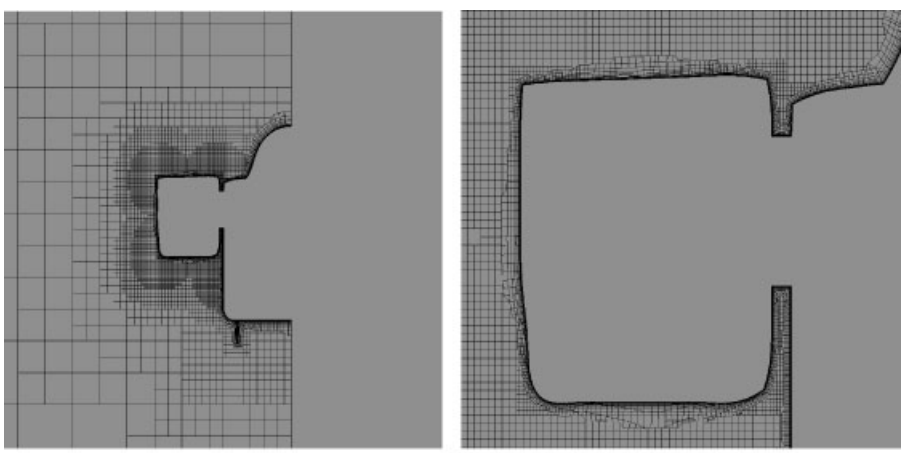

Figure 21. (a) A cut of constant x through the SOLAR mesh; (b) A close up view of the constant x cut through the intake.

7. CONCLUSIONS

It is concluded that:

- A highly automated mesh generation capability for use in conjunction with viscous flow solvers has been developed. For engineers who are used to the requirements for setting up input to unstructured mesh generators for inviscid flows, the effort required to become

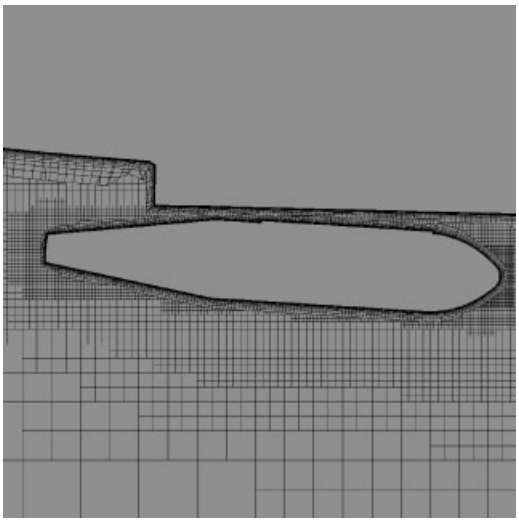

Figure 22. A cut of constant y through the SOLAR mesh.

proficient in using this new capability is minimal. This has enabled the rapid insertion of this new technology into use in the research and development and design environments.

- Mixed quadrilateral–triangular meshes can be formed on general surfaces using the moving-front approach. The quality of these meshes is ensured to be high through careful selection of the best region to mesh next and selective determination of best position to place a node. Smoothing algorithms and topological operations serve to further mesh quality. Typically, 99% of the polygons in the surface meshes will be quadrilaterals.
- The advancing-layer technique provides a suitable basis for creating the highly anisotropic cells required near to surfaces for efficient viscous flow modelling. However, great care is needed in the evaluation of the marching-direction and marching-distance to produce a robust capability.
- Augmenting the advancing-layer technique with cell collapsing and enriching techniques, which allow the topology of the layer to alter as the mesh is grown, significantly enhances the range of problems that the approach can handle robustly.
- The addition of front-intersection detection and local-layer termination techniques to the advancing-layer method enables closely coupled configurations to be treated and allows the growth of the near-field mesh to extend to cover the region where viscous effects dominate.
- The use of a Cartesian mesh of variable density provides an efficient covering of the far field. Cutting the cells to conform to the boundary of the near-field mesh is computationally inexpensive relative to the cost of reconstructing the faces of the cells. The quality of the mesh at the interface between the near and far field meshes can be greatly improved through the merging of small cells with neighbouring larger ones.
- The U.K industry/research-establishment initiative to fund and resource jointly a team of specialists from across the collaborating organizations to develop a capability to meet a common requirement has realized significant benefits rapidly.

ACKNOWLEDGEMENTS

The work reported here has been carried out with the support of the UK Ministry of Defence, the UK Department of Trade and Industry, QinetiQ, BAE SYSTEMS and Airbus UK. The work has also benefited from significant contributions from others, most notably Jamil Appa, Matthew Leatham, James Cooper and Thomas Blaylock. Finally, we wish to acknowledge the input to the work from other members of the SOLAR development team.

REFERENCES

1. Weatherill NP, Forsey CR. Grid generation and flow calculations for aircraft geometries. *Journal of Aircraft* 1985; **22**(10).
2. Allwright SE. Techniques in multi-block domain decomposition and surface grid generation. In *Proceedings of 2nd International Conference on Numerical Grid Generation in Computational Fluid Mechanics.* Sengupta S, Häuser J, Eiseman PR, Thompson JF (eds). Pineridge Press: Swansea, 1988; 559–568.
3. Peace AJ, May NE, Pocock MF, Shaw JA. Inviscid and viscous flow modelling of complex aircraft configurations using the CFD simulation system SAUNA. *ICAS Paper* 94-2-6.3, 1994.
4. Hunt DL, May NE. Practical use of transport turbulence models in aerospace – CFD implementation and applications. AIAA-99-3137. *17th AIAA Applied Aerodynamics Conference*, Norfolk, Virginia, June 1999.
5. Kalitzen G, Gould ARB, Benton JJ. Application of two-equation turbulence models in aircraft design. AIAA 96-0327. *34th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 1996.
6. Albone CM. Embedded meshes of controllable quality synthesised from elementary geometric features. AIAA 92-0663. *30th AIAA Aerospace Sciences Meeting*, Reno, NV, January 1992.
7. Blaylock TA. Application of the FAME method to the simulation of store separation from a combat aircraft at transonic speed. In *Proceedings of 5th International Conference in Numerical Grid Generation in Computational Field Simulations*. Soni BK, Thompson JF, Hauser J, Eiseman PR (eds). Mississippi State University, 1996; 805–814.
8. Morgan K, Peraire J, Peiro J. Unstructured mesh methods for compressible flows. *AGARD Report 787*, Special course on Unstructured Grid Methods for Advection Dominated Flows, 5.1-5.39, 1992.
9. Weatherill NP, Hassan O. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering* 1994; **37**: 2005–2039.
10. Shaw JA. Hybrid Grids. Chapter 23, In *The Handbook of Grid Generation*. Thompson JF, Soni BK, Weatherill NP (eds). CRC Press: Boca Raton, 1998; 23–18.
11. Shaw JA, Peace AJ. Simulating three-dimensional aeronautical flows on mixed block-structured/semi-structured/unstructured grids. *ICAS-98-2-7.1*, Melbourne, Australia, September 1998.
12. Leatham M, Chappell JA. On the rapid generation of hybrid meshes due to design driven geometry perturbation. In *Numerical Grid Generation in Computational Field Simulation*. Cross *et al*, MA (eds). Mississippi State University, 1998.
13. Chappell JA, Shaw JA, Leatham M. The generation of hybrid grids incorporating prismatic regions for viscous flow calculations. In *Proceedings of 5th International Conference in Numerical Grid Generation in Computational Field Simulations*. Soni BK, Thompson JF, Hauser J, Eiseman PR (eds). 1996; 537–546.
14. Chappell JA. Marching grid generation with element modification and its role in construction of hybrid meshes for viscous flow calculations. *ARA CR M358/1*, January 1999.
15. Smith RJ, Leschziner MA. A novel cartesian grid method for complex aerodynamic CFD applications. In *Proceedings of 5th International Conference in Numerical Grid Generation in Computational Field Simulations.* Soni BK, Thompson JF, Hauser J, Eiseman PR (eds). 1996; 709–718.
16. Appa J, Hughes R, Porter L, Woods PD, Hunt DL, Rham S. Generating rapid response Navier–Stokes solutions on hybrid meshes using two-equation turbulence models. *AIAA Paper* 2000-2677, Denver, Colorado, June 2000.
17. Pirzadeh S. Unstructured viscous grid generation by advancing layers method. *AIAA Paper* 93-3453-CP, 1993.
18. Marchant MJ, Weatherill NP. Unstructured grid generation for viscous flow simulations. *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*. Weatherill NP, Eiseman PR, Häuser J, Thompson JF (eds). Pineridge Press: Swansea, 1994; 151–162.
19. Kallinderis Y. Discretization of complex 3D flow domains with adaptive hybrid grids. In *Proceedings of 2nd International Conference on Numerical Grid Generation in Computational Fluid Mechanics.* Sengupta S, Häuser J, Eiseman PR, Thompson JF (eds). 1996; 505–515.
20. Sutherland IE, Hodgman GW. Reentrant polygon clipping. *Communication of the ACM* 1974; **17**(1):32–42.
21. Stokes S, Shaw JA, Leatham M. The automatic generation of grids for viscous flow simulations. *ARA CR* M371/3, October 2001.

22. Leatham M, Stokes S, Shaw JA, Cooper J, Appa J, Blaylock TA. Automatic mesh generation for rapid-response Navier–Stokes calculations. *AIAA Paper* 2000-2247, Denver, Colorado, June 2000.
23. Stokes S, Leatham M, Shaw JA. Three-dimensional hybrid mesh generation for viscous flows using the advancing-layer approach with automatic layer topology modification. *Numerical Grid Generation in Computational Field Simulations*. Soni BK, Thompson JF, Häuser J, Eiseman PR (eds). ISGG, Mississippi State University: MS, USA, 2000; 233–242.
24. Szmelter J. Viscous coupling techniques using unstructured and multiblock meshes. *ICAS*-96-1.7.4, Sorento, Italy, September 1996.
25. Blacker TD, Stephenson MB. Paving a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
26. Cass RJ, Benzley SE, Meyers RJ, Blacker TD. Generalized 3D paving: an automated quadrilateral surface mesh generation algorithm. *International Journal for Numerical Methods in Engineering* 1996; **39**(9):1475–1490.
27. White DR, Kinney P. Redesign of the paving algorithm: robustness enhancements through element by element meshing. *6th International Meshing Roundtable*, October 1997, Albuquerque NM.
28. Lee CK, Lo SH. A new scheme for the generation of a graded quadrilateral mesh. *Computers & Structures* 1994; **52**(5):847–857.
29. Canann SA, Muthukrishnan SN, Phillips RK. Topological improvement procedures for quadrilateral finite element meshes. *Engineering with Computers* 1998; **14**:168–177.
30. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in fortran. *The Art of Scientific Computing* (2nd edn). 1992.

