Advancing Layer Surface Mesh Generation

Jasmeet Singh* and Carl Ollivier-Gooch†
University of British Columbia, Vancouver, British Columbia, V6T1Z4

A new method of generating advancing layer surface meshes is demonstrated. We generate semi structured quad-dominant meshes with the ability to have local control over the aspect ratio. Semi-structured two-dimensional anisotropic meshes in the boundary layer regions have been shown to have superior fluid flow simulation results. However, the discretization of the surfaces poses challenges in replicating the same for volume meshes. Beginning from an isotropic surface triangulation, we incrementally replace that mesh with an anisotropic mesh. Point placement in layers and local reconnection helps produce a valid surface mesh at each step of mesh generation. We demonstrate the ability of the meshing algorithm to tackle fairly complex geometries and coarse initial surface discretization.

I. Introduction

The need to generate advancing layer meshes is prevalent in mechanical engineering and aerospace applications [1]. Boundary layer meshes generated by advancing layer strategy have found applications even in biomedical computing [2]. Wherever there is a need to refine the elements around the boundary of an object, a high-aspect ratio advancing layer mesh is desired for carrying out the simulations. Such a mesh helps in capturing the gradients normal to the direction of a specified boundary without refining the region along the boudnary. Hence, computational cost and memory usage are considerably reduced.

In many aerodynamic applications, a boundary layer volume mesh with anisotropy in specific directions is very much desired. Some of the initial attempts at generating stretched element meshes in two-dimensions used a Delaunay mesh and a locally mapped space to get the required anisotropy [3, 4]. Other techniques either used an approach of using a locally structured or semi structured mesh for the regions requiring high anisotropy[5, 6] or using an advancing layer strategy to generate the mesh in layers while terminating locally whenever necessary([7–9]). Several techniques have also been developed to generate boundary layer volume meshes [10–14].

The methods stated above work on simplical meshes, ie meshes with triangles in two dimensions and tetrahedra in three dimensions. These simplical meshes have an advantage of good flexibility in descretizing the domain, especially if it contains complex features. However, the vertex connectivity of simplical mesh elements is high. The number of edges in a tetrahedral mesh is about seven times the number of vertices. On the other hand, in a hexahedral mesh, the number of edges is only about three times the number of vertices (asymptotically). The cost of vertex-based discretization is directly proportional to the number of edges in a mesh. Hence, non-simplical meshes are substantially more efficient than simplical meshes for a given number of unknowns or grid points. Also, regular arrays of nonsimplical elements may also enhance accuracy, owing to a local cancellation of truncation errors that may not occur on groups of nonsimilar simplical elements[15]. Quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [16]. The advantages of non-simplical mesh elements have resulted in fully non-simplical mesh generation techniques [17, 18]. However, simplical mesh elements are necessary to deal with the complexity in the geometry. Hence, a hybrid mesh containing both simplical and non-simplical mesh elements is a reasonable choice for mesh generation. We generate a surface mesh which is quad dominant with some triangles in it.

Stretched volume meshes are generated from an initial triangulation of a surface. These surfaces are usually represented by NURBS in various CAD packages. Generation of the surface mesh requires a separate methodology or algorithm as compared to the two-dimensional mesh generation as the mesh elements include a third dimension. The surface mesh can be created through a parametric mapping from two-dimensional mesh or by a direct three-dimensional method. In the first method, the mesh can be created by mapping a two-dimensional Delaunay mesh onto a surface[19, 20] using surface parametrization. This method, however, doesn't always form well shaped surface elements if the surface derivatives vary widely over the domain [21]. Also, these methods do not intend to generate an anisotropic surface mesh. Analogous method has also been introduced to generate the surface mesh using advancing front technique by utilizing a metric derived from the first fundamental form of the surface [22, 23].

*Graduate Research Assistant, Mechanical Engineering, jasmeet.singh.mec11@iitbhu.ac.in
†Professor, Mechanical Engineering, cfog@mech.ubc.ca; Associate Fellow, AIAA

Another way of creating a surface mesh is by creating mesh elements directly on the surface geometry. Lao and Lu[24] have formulated an advancing front approach for arbitrary three dimensional surfaces. This approach creates triangular elements on the surface geometry. It uses surface normal, tangent and projection computations to generate the surface mesh. In another work [25], they use this triangular mesh to generate an all quad mesh. For this method to work successfully, the underlying triangular mesh needs to be sufficiently well resolved in geometrically complex features of the mesh. Also, meshes developed by this method also do not involve anisotropic surface mesh elements.

We present a surface mesh algorithm that creates a surface discretization with required directional anisotropy and mostly quad elements. Such a surface mesh can be a stepping stone towards an anisotropic hex-dominant boundary layer volume mesh. The surface mesh generation algorithm is based on a closed advancing front method. The method needs minimal user input and automatically generates an advancing layer mesh from a given input surface triangulation. The input surface is first segmented into several sub-surfaces so that each of them can be meshed independently. Boundaries of the surfaces are identified and the points on the boundary iteratively march towards the interior of the surface. Initial extrusion length and growth ratio can be defined to give the required level of anisotropy at each point of the mesh or for a given boundary. A valid surface mesh is produced after each layer which gives the user freedom to advance until a given level of anisotropy is reached or until the advancing front routine terminates.

However important may the advancing layer mesh be, its creation poses challenges for geometries that are complicated. Sharp corners and highly curved regions in the input geometry pose robustness issues for most mesh generators. Our advancing front routine includes several validity checks to avoid these issues. However, we are working on making it more robust for complex geometries.

II. Methodology

The surface mesh algorithm presented in this paper starts by importing a surface triangulation of and creating an interpolated model of the surface. Steps taken from importing surface triangulation to the mesh completion are detailed in the following subsections.

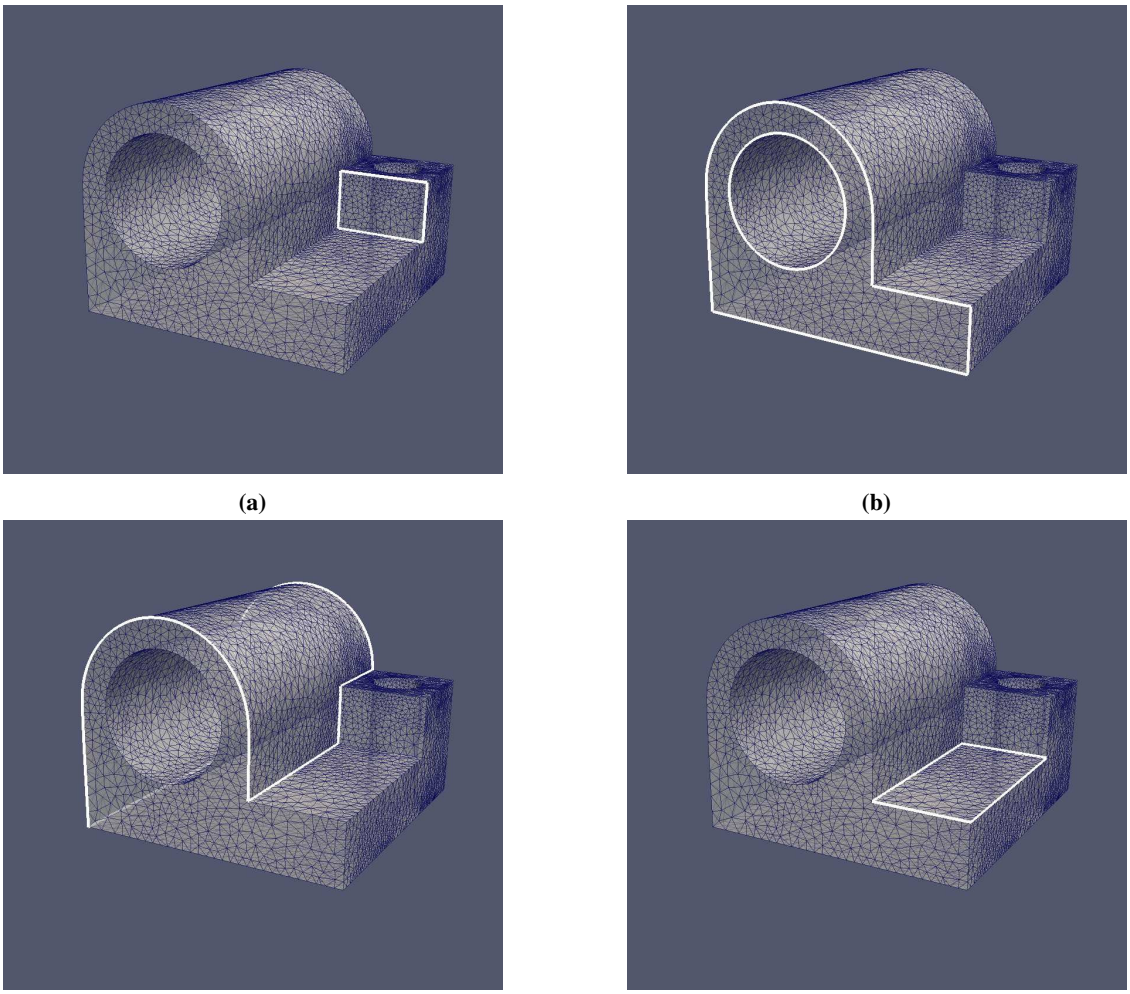
A. Surface Import and Segmentation

The surface mesh generator reads a surface triangulation as a faceted geometry. It then uses the Common Geometry Module (CGM)[26] facet based geometry kernel to create an interpolated geometry from the facet based triangulation. G^1 continuous curves and surfaces are created from triangles which form quartic Bezier triangle patches. The geometry imported by CGM is used as the background surface to create the mesh. Robust and cost efficient geometry and topology query operations in CGM help us in creating the surface mesh. The most fundamental requirements of the surface mesh from the geometry are - a closest point or point projection routine and surface normal evaluations. These operations are frequently used to project point onto the underlying geometry, compare the normal of a triangle to the normal at a point on the underlying surface, etc.. After the surface geometry is created, the imported surface is segmented into sub-surfaces based on the the dihedral angle between neighbouring facets. This results in creation of sub-surfaces which together form the imported surface. An example of a surface triangulation with some of its sub-surfaces is shown in Figure 1. The input geometry is segmented into ten sub-surfaces (four shown in the figure). The boundaries of these sub-surfaces are highlighted. These sub-surfaces are meshed separately by the advancing layer surface mesh algorithm.

The imported geometry in CGM is stored in memory while we create the surface mesh. The segmented surface gives us the freedom of meshing sub-surfaces independently. The ability of meshing sub-surfaces independently allows us to decide the anisotropy direction independently for the surfaces and can also be helpful if the surface mesh generation is required to run in parallel.

B. Boundary Initialization

After importing the surface triangulation, we have a valid underlying surface representation with us. Also, segmented sub-surfaces and their boundaries provide us with the boundary we need to march off of. The mesh generation routine starts by initializing the boundaries of each sub-surface of the surface by marking each point on the boundary as a candidate marching point which form the starting layer in the mesh. The points at the boundary of the mesh serve as the parent points for the first layer inserted into the mesh. The data structure for each boundary front vertex stores its adjacent edges so as to identify the marching directions which is explained in detail in the next subsection. Each edge stores the direction into the interior of the sub-surface it bounds with respect to the surface normal. As the sub-segments sharing a common boundary are meshed independently, it is easy to identify the normal to an edge along the sub-surface



(a) **(b)**

(c) **(d)**

Fig. 1 An example input triangulation with some of the segmented surfaces.

for advancing the front in the mesh generation algorithm.

C. Advancing Layer Routine- Point placement

For each of the sub-surfaces of the geometry, the advancing layer routine iteratively picks a point from its boundary and extrudes it in a given direction. After evaluating the extruded point, we project the point onto the underlying surface. This process is set up to be of two steps for simplicity, accuracy and computational efficiency as will be explained later.

In the first step, we extrude the parent point to get the extruded point. We would interchangeably call the extruded points as the kid points as they represent the successors of their parent points from the previous layer. The direction of this extrusion is set up to be the average of the normals of the parent vertex's adjacent edges in the tangential plane of the sub-surface. In other words, the normals of the adjacent two edges of a vertex on the underlying sub-surface are averaged to get the extrusion direction. Consider Figure 2; we need to find the extrusion direction of a point P on the boundary. In the underlying triangulation T, PB and PA are the edges adjacent to P on the boundary of the surface S. We first find the points B_m and A_m, which are the mid-points of quartic-Bezier curves PB and PA which are constructed by CGM as a part of constructing the quartic-Bezier triangular surface patches from the underlying triangulation T. Next, we find the normal directions on the surface at points B_m and A_m which are labeled as N_b and N_a in the figure. We cross product the vector PB with N_b to get the direction T_b. The vector T_b is normal to the edge PB as well as tangential to the surface S. Similarly, we find the vector T_a. The direction of extrusion PK is chosen to be the average of the direction of T_a and T_b.

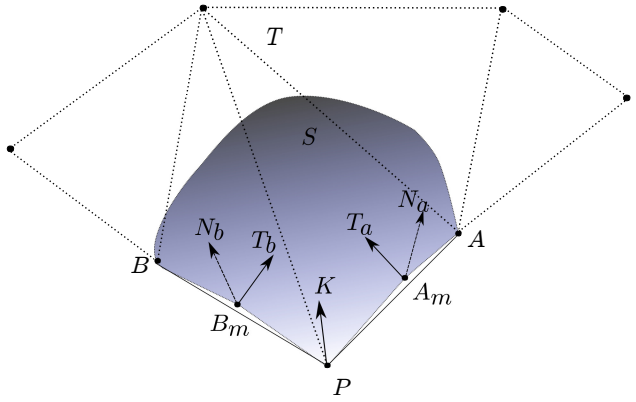


Fig. 2 Calculation of the extrude direction for a point P in the advancing front

The initial extrusion length is an input parameter provided by the user. This length would be taken as the extrusion length when the boundary points are extruded for the first time to the interior of the surface. This length can vary with boundary vertices as the points are extruded independently. Hence, this extrude length can either be supplied by the user for all the points of the boundary separately or as a single value for all the boundary points. To obtain best quality quad elements, we scale the extrusion length at a given vertex on the advancing layer with respect to the interior angle between the direction vectors T_a and T_b. If the vertex is a concave corner vertex, the extrusion length is increased so as to create good quality quad elements in the next layer. This process is described in detail in section II.G.

After we have extruded the point, we project it on to the underlying geometry. This operation ensures that all the points we insert in the mesh are on the underlying geometry. Errors here would compound in subsequent layers. Points are inserted in the mesh and the mesh elements are subdivided to include the new point. The candidate point for insertion can subdivide an existing triangle to replace the previous triangle with three new ones, or can subdivide an edge to replace existing two triangles with four new ones. To find the best triangle or edge for subdivision, we first make a guess for the triangle to insert the point. Any triangle in the surface interior adjacent to the point being extruded is chosen. Starting from this triangle, we iteratively jump to the best edge or triangle by comparing the barycentric coordinates of the new point with respect to the triangle in consideration. This technique suffers from two disadvantages. First, we need to compare double precision values of barycentric coordinates for making a decision on which triangle to choose for insertion. If the values are too close, the point might be inserted in the wrong triangle and would eventually lead to deviation of the mesh from the underlying surface. Second, the process of iteratively finding the right triangle for insertion might end up being in an infinite loop. Both of these problems are substantially reduced with a good isotropic initial triangulation. However, we add several validity tests to avoid these problems even for a coarse initial triangulation. These include orientation checks of the triangles formed with respect to the surface, thresholding the maximum deviation of the newly formed triangle from the surface and thresholding the dihedral angles between two triangles on the surface. We use an epsilon value of 10^{-5} while comparing the values of barycentric coordinates to zero. Also, we insert the point on a face rather than in a triangle when the ratio of the second-smallest barycentric coordinate to the smallest one is more than a set threshold (10^2). This helps us avoid very skinny triangles with large obtuse angles and also helps in avoiding several unnecessary face swapping in the mesh.

After advancing one layer to the surface interior, we increase the extrude length at each point by a factor. This factor, called the growth ratio, specifies the anisotropic layer-on-layer extrusion length growth as we march on the surface. A value of growth ratio between 1.1 and 1.4 gives us satisfactory anisotropy at the boundaries of the mesh.

D. Local Reconnection for Quality

A mesh observer object is initialized for the mesh data structure. The observer monitors all the changes happening to the mesh. These include creation and deletion of vertices, edges and cells in the mesh. An edge swapping manager and quality criterion for deciding a swap are also initialized for the mesh. The primary quality criterion we chose for an edge swap is maximization of the minimum angle in the pair of triangles adjacent to the edge before and after the swap. However, only this criterion is not enough for improving surface mesh quality. This is because a better quality pair of triangles may result by edge swapping according to this criterion, however, they might deviate quite a bit from the underlying surface. Hence, we threshold the maximum deviation of the resultant triangles from the surface. This

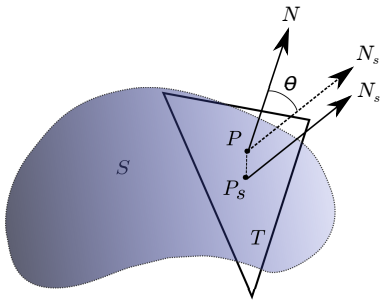


Fig. 3 Triangle T deviates from the underlying surface S. P is the centroid of T and P_s is the projection of P on S. The deviation is calculated as the interior angle between the normal to the triangle PN and the normal to the surface at P_s, which is P_sN_s. The angle θ represents the deviation here. The deviation is exaggerated for illustration purposes.

deviation is calculated as the angle between the normal to the triangle and the normal at the centriod of the triangle projected onto the surface. See Figure 3 for reference. Deviation of a triangle T with respect to a surface S is shown. P is the centroid of the triangle and P_s is the projection of the centroid onto the surface. PN is the normal to the triangle and P_sN_s is the normal to the surface at P_s. The interior angle θ is considered as the deviation of the triangle from the surface. A upper bound is set for θ to limit the deviation of resultant triangles from the surface. The bound is set to 30° for the surfaces meshes we are dealing with. This bound is sensitive to the refinement in the initial triangulation. A higher value of θ would be appropriate for a coarse initial triangulation while the value can be set low for a fine initial triangulation.

We queue up the edges of the triangles affected by the insertion of the new point and do edge swapping for these edges to improve on the existing mesh. Figure 4 shows the process of point insertion and local reconnection in two steps. First, a point is inserted into the mesh at the desired location. This results in subdivision of a triangle as can be seen in figure 4b. Subsequently, edge swaps occur to improve the mesh quality. Two iterations of edge swap can be seen in figure 4c and 4d. We do not swap the edges which comprise the front or the edge between parent and child points. Doing so would invalidate the advancing front. Hence, only interior faces of the surface or the diagonals of the quad elements generated are swapped by the face swapping algorithm. After edge swapping, the algorithm moves on to the next point in the current layer to repeat the same process.

E. Front Recovery

In an advancing layer mesh generation algorithm, the accurate representation of the boundary by anisotropic layers generated from it is very important. Also, generation of quad-elements from triangles with the right aspect ratios is far easier if we retain the boundary representation as we advance multiple layers. Hence, after all points of the parent layer are extruded, we implement a front recovery routine which recovers all edges between kid vertices.

Front recovery is done by selecting a pair of points from the parent layer and then forcing an edge between the points extruded from these parent points; that is by forcing an edge between the kid vertices in the extruded layer. An edge is forced between the kid vertices by iteratively swapping any edge that lies topologically between the two kid points. This kind of forced swapping in the mesh might seem to decrease mesh quality. However, with the right boundary discretization, it helps in giving us the high-aspect ratio elements in the mesh that we desire. After all the kid vertices are connected by this method, validity checks are run to ensure that the extruded layer is well defined and there are no discrepancies in this layer as it will serve as the parent layer for the next one. This incorporates checking the connectivity of each vertex with its adjacent edges and the connectivity of each edge with its end points. Figure 5 shows how exactly we recover an edge in the front by iteratively swapping the edges which obstruct the edge creation between two kid vertices, P_1 and P_2 in the figure. The edges which are topologically obstructing kid vertices P_1 and P_2 are swapped iteratively. A test is setup for each swap to avoid the flipping of triangles on the surface. After all the obstructing edges are swapped, a front edge between the kid vertices P_1 and P_2 would be recovered as shown in green in Figure 5e.

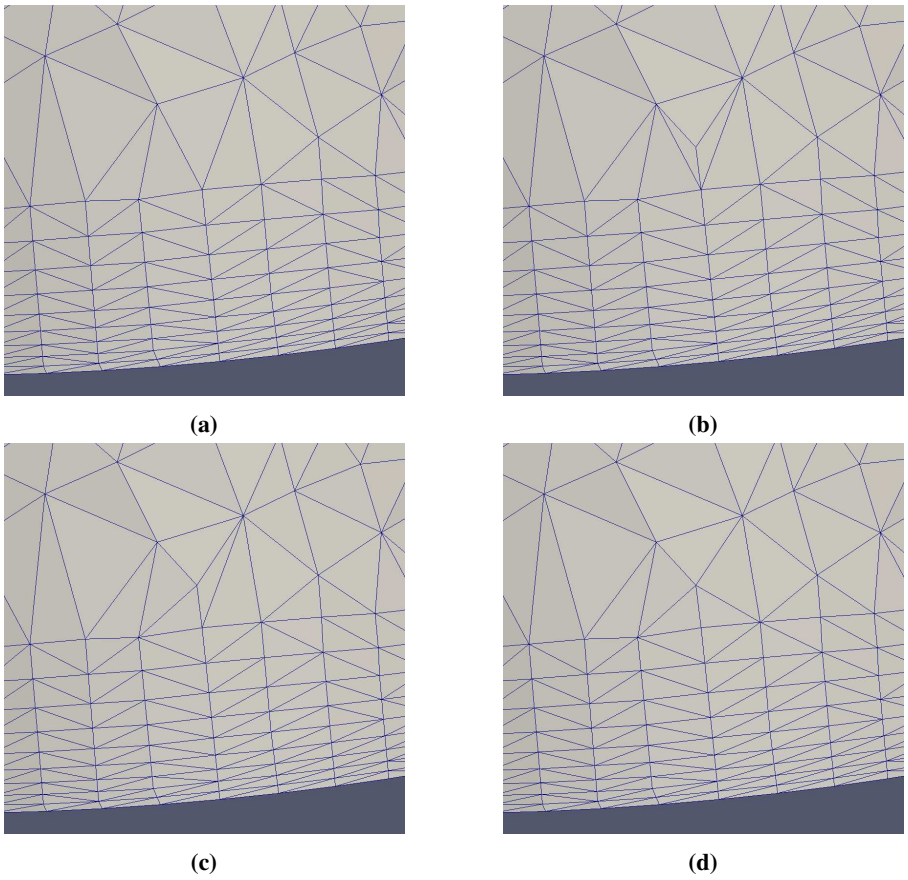


Fig. 4 Point Insertion and local reconnection for quality shown in four steps. (a) is the initial state of the mesh. A point is inserted in the mesh after extrusion from the parent point, projection onto the surface and finding the right triangle to insert. The triangle is subdivided into three new triangles as shown in (b). To improve the mesh quality after point insertion, swapping is done based on maximization of the minimum angle in a triangle. Two swaps occur as shown in figure (c) and (d) to improve mesh quality.

F. Edge Collapse and Interior Improvement

As the advancing front moves into surface interior, we decimate interior vertices of the initial triangulation as well as boundary vertices which are too close to each other through edge-collapse.

Once we have recovered the advancing front by iterative edge swaps to connect the kid points in the mesh, we check for vertices on the front that are too close to each other relative to the extrusion length. For instance, vertices which are near a concave corner could encroach each other if they are not decimated. Another example would be vertices on the advancing front which are far away from surface boundaries because the extrusion length has grown by this point. Decimating vertices from the front which are at a substantial distance from surface boundaries helps prevent the cell aspect ratio, ie front edge length over extrusion length, from dropping below one.

To check for vertices to decimate, we iterate through the vertices in the front and identify the ones which are too close to their neighbours. Vertex decimation is done through the edge-collapse routine as described in [27]. The threshold edge length between two points on the front is set to be $2\tan(\pi/8)$ times the average extrusion length at those points. This value is set so as to minimize the normalized maximum deviation of angle from 90° for quad elements. Hence, the threshold ensures that the anisotropic properties are retained for several layers into the surface. All short edges on the front are collapsed using the edge-collapse algorithm. An example of an edge-collapse on the front is shown in Figure 6. Here, two points P_1 and P_2 are collapsed into a single point which forms a part of the next front on the surface.

As the advancing front marches into the surface, vertices in the interior of the surface immediately next to the front are decimated to make way for the advancing layers. Before extruding a point on the advancing front, we check if any point in the surface interior with which it shares an edge agrees with the following condition. If it does, we decimate the

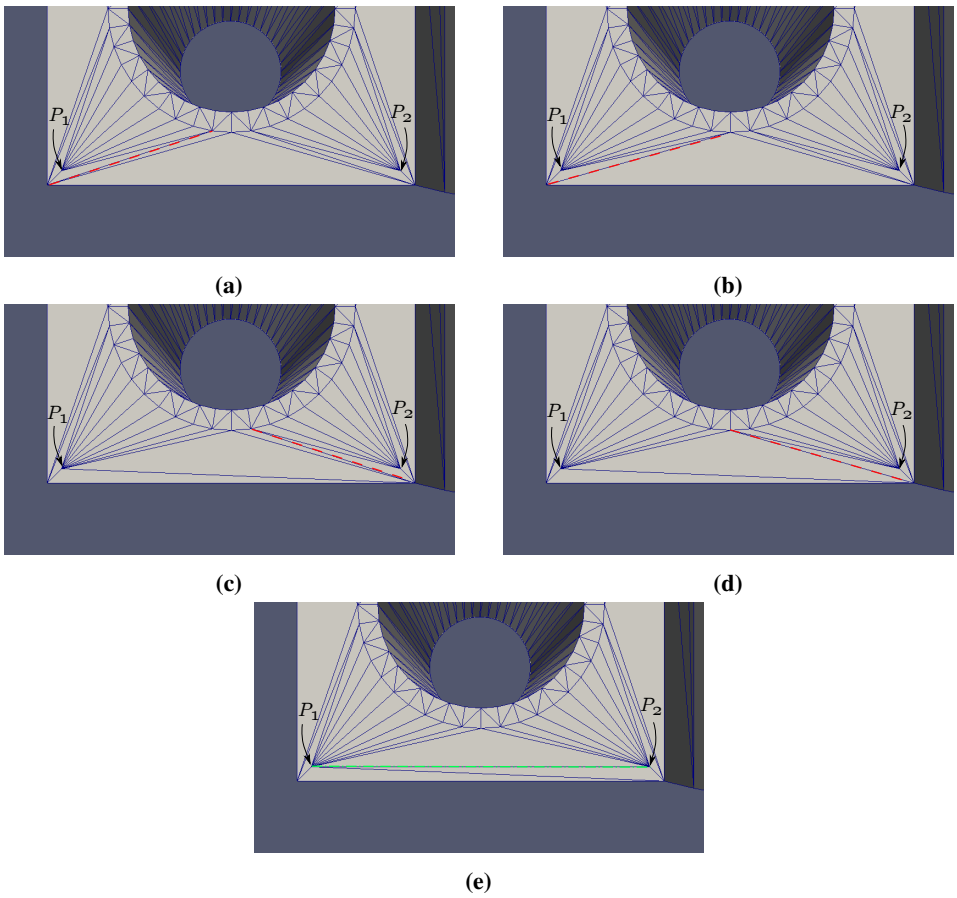


Fig. 5 Front Recovery: Point P_1 and point P_2 here represent the two kid points generated from the boundary of the surface. We try to force a connection between the two points by iteratively swapping edges which topologically obstruct their connection. Red dashed lines represents the edge chosen to be swapped next. In (e), the green edge is the edge recovered and would serve as a part of the next front in the mesh. Note that this example is for front recovery illustration purposes and the initial boundary discretization is too coarse to get a good-quality advancing layer mesh.

interior vertex.

$$d < max\left(l_1/\sqrt{2},\ l_2/\sqrt{2},\ c \times extrudeLength\right) \tag{1}$$

Here d is the distance between the point on the advancing front and the interior point, l_1 and l_2 are the lengths of adjacent edges of the point on the front, c is a constant whose value is set as 2 and $extrudeLength$ is the extrusion length at the vertex on the front. This condition ensures decimation of vertices in the surface interior which are close to the advancing front and avoids any encroachment of surface interior vertices on the advancing layers. Topological and geometrical checks are done before an edge can be collapsed. These include a threshold for the ratio of area of generated triangles and a limit on the dihedral angle between the adjacent triangles created by the collapse. The area threshold is set to be 10^8. Also, edge-collapse is successful only if the triangles resulting from it are be within a limit of $\theta < 30°$ from the surface(see Figure 3). The threshold for the maximum dihedral angle between any two adjacent triangles that result from the edge-collapse is set to 40°. The quality criterion used for edge collapse is again maximization of the minimum angle in the triangles thus produced. The best edge for collapse is chosen when decimating the interior vertices using this quality criterion. This is in contrast to the vertex decimation on the advancing front where the candidate edge for collapse is already identified. An illustrative example for surface interior vertex decimation can be seen in Figure 7.

After we have a recovered the front and decimated encroaching vertices in the surface interior as well as on the advancing front through edge collapse, we queue up the immediate interior edges of the surface and swap them for

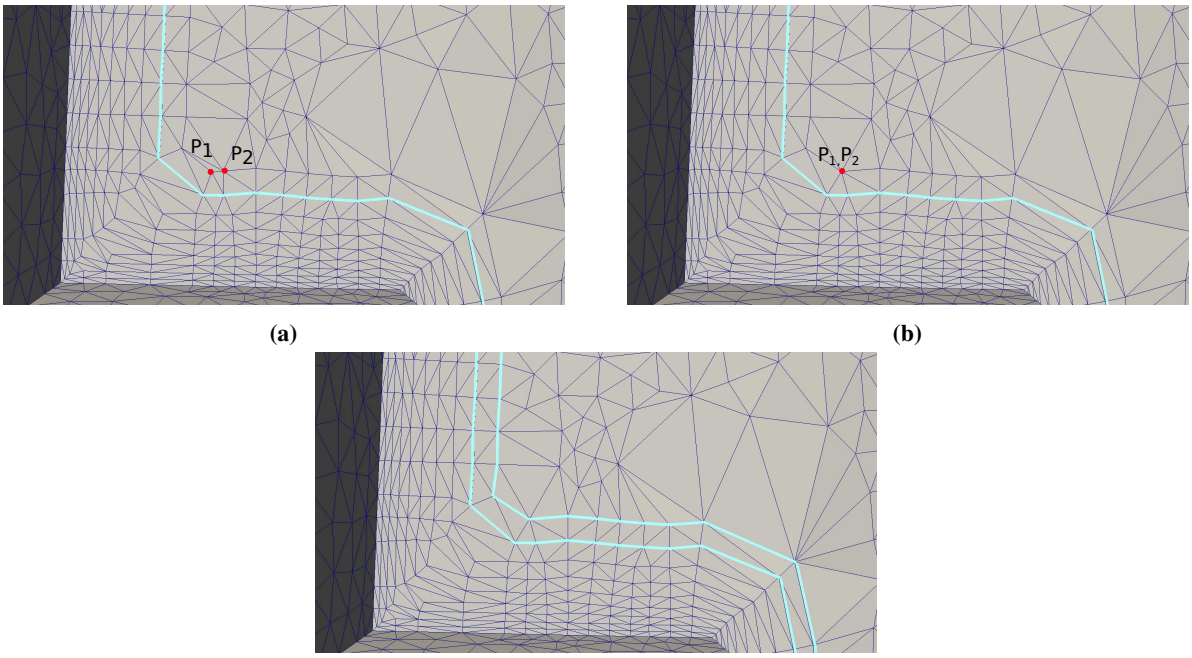


Fig. 6 Edge collapse on an advancing front to avoid encroachment of points. In (a), two points in the kid layer P_1 and P_2 are sufficiently close to each other. Their parent layer is highlighted. If both the points advance to the next layer, then the next front would fail to recover. Hence, the edge between them is chosen to collapse. (b) shows the result of the edge collapse. The new location of both the points is the average position of their initial location. (c) shows how the next front looks like.

maximizing mesh quality. This step is included so that we have a good interior triangulation at each step of the advancing layer routine. This step ensures that we do not have bad triangles causing problems as we continue to march.

G. Extrusion Length Scaling At Concave Corners

The advancing layer surface mesh takes a fixed initial extrusion length and grows the layers from it using a user supplied growth ratio. However, having a fixed initial extrusion length for all the vertices at the boundary would lead to collisions at concave corners of the mesh. Hence, we scale the extrusion length at the vertices of the advancing layer to take into account the concavity of the layer at that particular vertex.

Let us consider a concave corner vertex P in the mesh located at an advancing layer. The adjacent edges to P on the layer are PR and PQ. The included angle between these two edges is denoted as 2θ. Let's assume that the length of the edge PQ is l (see Figure 8) and the extrusion length for the layer at which vertex P is located as h. We need to find a reasonable extrusion direction and extrusion length for the corner point P.

For good quad cells in the surface mesh, i.e. quad cells with angles as close to right angles as possible, we need the point P to 'race' a bit initially towards the interior to catch up with its adjacent points. This would be helpful in avoiding mesh collisions. Hence, we formulate a method to find the scaling factor for the extrusion length.

The line PC is constructed at equal angles from edge PR and PQ. Let us place the point X extruded from point P on the line PC. In an orthogonal structured mesh scenario, if the angle between PR and PQ was 180 degrees, the extrusion direction should have been along the edge PA and the extruded point would be at a distance h from the point P, which coincides with point A in the figure. The quad thus formed would have been $PABQ$. However, this is not the case. We have a skewed interior angle between PQ and PR. If we place the extruded point at a distance h from P along PC, the mesh is going to fold onto itself in the next few layers. To avoid this, we take an alternative approach. We place the point X further away from distance h. This distance is calculated using the following strategy.

The area of the quad $PABQ$ would have been $h \times l$ in the orthogonal mesh case. Our strategy is to put the point X on the line PC such that the area of quad $PXBQ$ is also $h \times l$.

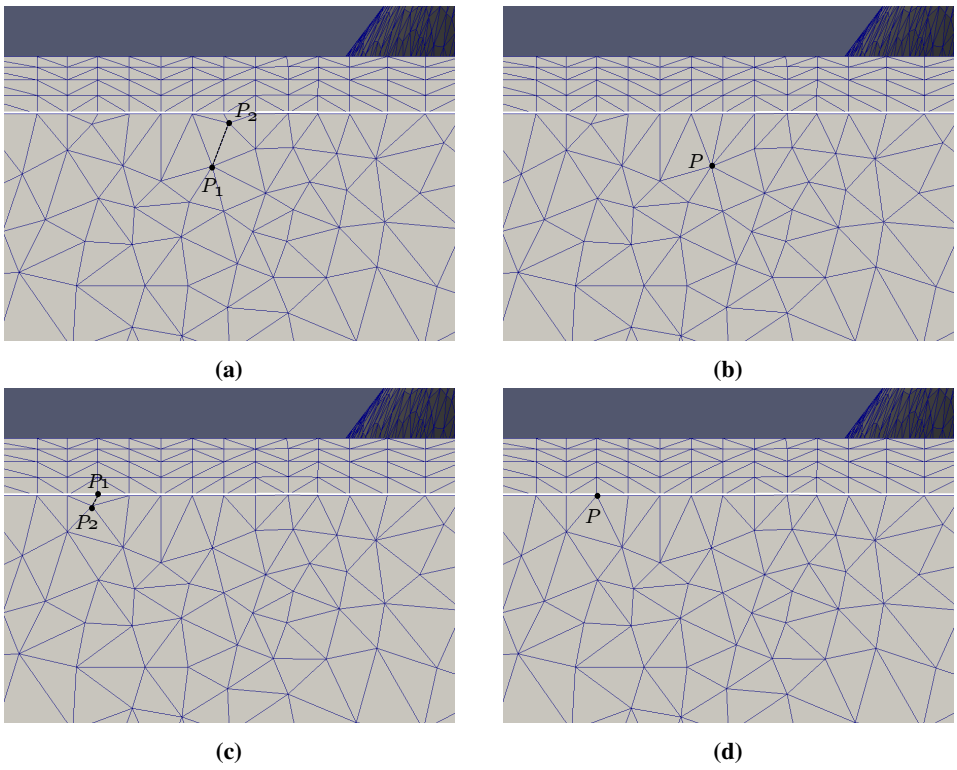


Fig. 7 Interior vertex decimation through edge collapse. The highlighted line shows the advancing front. In (a), vertex P_2 is about to encroach the front. Hence, the best vertex for collapse is chosen among its neighbours. The best vertex for edge collapse here is P_1. Hence, P_2 is collapsed on to P_1. The connectivity after the edge collapse is shown in (b) where vertex P represents the collapsed vertex. Similarly, in (c), vertex P_2 is about to encroach the advancing front and is collapsed onto vertex P_1 which is on the advancing front itself. The new connectivity is shown in (d) where all the possibly encroaching vertices for the advancing layer are decimated.

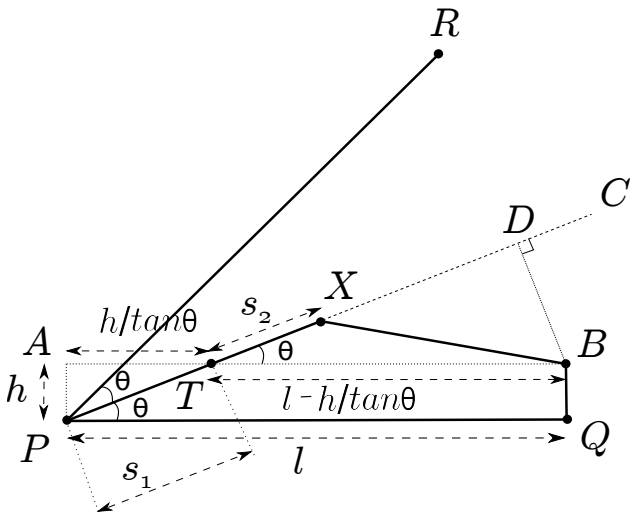


Fig. 8 Extrusion

Let us assume that the line PC intersects AB at point T. The area of the quad $PTBQ$ is common to both quads $PABQ$ and $PXBQ$. Hence, to make the area of the quads $PABQ$ and $PXBQ$ the same, we just have to equate the area of $\triangle APT$

and $\triangle XTB$.

$$Area(\triangle ATP) = Area(\triangle TXB) \tag{2}$$

Using this equation, we can find the length PX. Let us assume that line segment PT is of length s_1 and TX is of length s_2. Hence, $PX = s_1 + s_2$. s_1 can be easily computed as $h/\sin\theta$. To find s_2, we use the area constraint from equation 2. Area of triangles $\triangle ATP$ and $\triangle TXB$ can be easily computed in terms of variables h, l and θ.

$$Area(\triangle ATP) = \frac{1}{2} \,.\, h \,.\, \frac{h}{\tan\theta} \tag{3}$$

$$Area(\triangle TXB) = \frac{1}{2} s_2 \left(l - \frac{h}{\tan\theta} \right) \sin\theta \tag{4}$$

Solving for s_2 using equations 2, 3 and 4

$$\begin{aligned} \frac{1}{2}\frac{h^2}{\tan\theta} &= \frac{1}{2} s_2 \left(l - \frac{h}{\tan\theta} \right) \sin\theta \\ \implies s_2 &= \frac{h}{\sin\theta} . \frac{1}{(l/h)\tan\theta - 1} \end{aligned} \tag{5}$$

Finally PX is computed as

$$\begin{aligned} PX &= s_1 + s_2 \\ &= \frac{h}{\sin\theta} \left(1 + \frac{1}{(l/h)\tan\theta - 1} \right) \end{aligned} \tag{6}$$

Let l/h be denoted as the aspect ratio A.

$$PX = \frac{h}{\sin\theta} \left(1 + \frac{1}{A\tan\theta - 1} \right) \tag{7}$$

If the lengths of the adjacent edges PQ and PR are different, we would want to consider both the aspect ratios to calculate the length PX and then average it. This gives us

$$PX = \frac{h}{2\sin\theta} \left[2 + \frac{1}{A_1\tan\theta - 1} + \frac{1}{A_2\tan\theta - 1} \right] \tag{8}$$

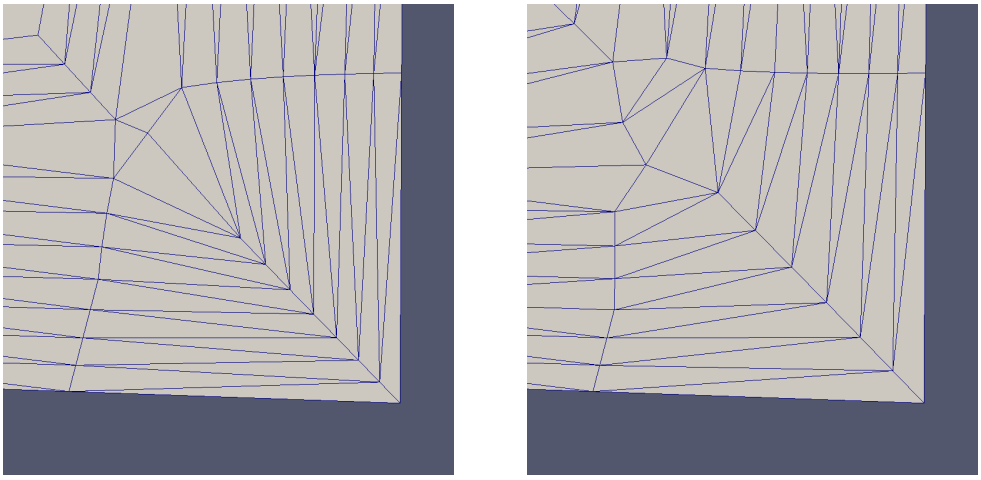
And the extrusion factor f is simply PX/h

$$f = \frac{1}{2\sin\theta} \left[2 + \frac{1}{A_1\tan\theta - 1} + \frac{1}{A_2\tan\theta - 1} \right] \tag{9}$$

Hence, the extrusion factor f depends on the half included angle θ and the aspect ratios A_1 and A_2 at the vertex. We use a couple of constraints while using this extrusion factor. First, we ensure that $A > cot\theta$, so that we never get negative values for f. Secondly, we limit the extrusion length at any vertex to be less than the minimum of the length of adjacent edges on the boundary. This is done because the extrusion factor can be really large for small angles and small aspect ratios. Hence, to avoid generating skinny elements at concave corners of the mesh, we limit the extrusion length at any vertex such that the resultant aspect ratio at that vertex would always be more than 1. Figure 9 shows an example of scaling the extrusion length at the concave corner. Along with preventing layers to fold onto themselves, this technique also helps in improving the quality of quad cells generated in the mesh.

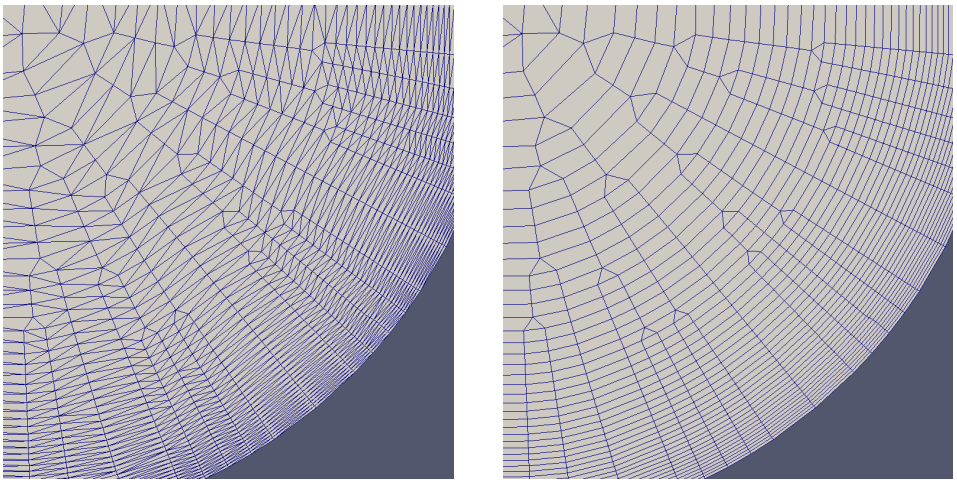
H. Combining Triangular Elements to Quads

Generating advancing layers over the surface proceeds by inserting points in the initial triangulation. After generating a layer of the mesh, we can combine the triangular cells into quadrilateral cells by removing edges from the mesh. Edges are removed from the finished layers so as to generate superior quality quadrilateral cells. As the layers advance from the boundary towards the tangential direction onto the surface, the quadrilateral elements generated retain the boundary topology several layers into the interior of the surface. The quality of a quadrilateral element is calculated as the inverse of the maximum deviation of its interior angles from 90 degrees. This criterion is selected so as to prefer rectangular elements and retain the boundary representation on the surface mesh. Multiple iterations of this subroutine are run for the same layer during mesh generation to ensure as few triangular elements are generated per layer as possible.



(a) (b)

Fig. 9 Extrusion length scaling is illustrated using a concave corner. In (a), the layers are about to fold onto one another because of constant extrusion length at each layer. (b) shows the vertex at the corner having a larger scaled extrusion length so as to maintain higher quad quality and prevent layer collision.



(a) (b)

Fig. 10 Combining triangular cells to quadrilateral cells based on the quality of the quadrilateral cells generated. For any given quadrilateral element, the inverse of the maximum deviation of its interior angles from 90° is adopted as the measure of quality.

I. Smoothing

Applying a smoothing methodology to a boundary layer mesh is non-trivial. On the one hand, smoothing improves the mesh element quality, while on the other hand it can diffuse the boundary topology in the advancing layers of the mesh. Hence, a simple Laplacian smoothing technique will not serve our purpose. Optimization based approaches could be employed to smooth the mesh by moving the mesh nodes so as to maximize a quality metric[28]. However, even in such approaches, the boundary layers of the mesh are kept fixed and the isotropic regions of the mesh are smoothed. Additionally, this approach is quite computationally intensive. The smoothing methodology that we use for the advancing layer mesh is physically-based smoothing. The mesh nodes exert forces on each other and each mesh node is subsequently moved so as to balance out all the forces. This procedure of smoothing is also called as spring based smoothing as the edges of the mesh act as springs which pull or push the mesh nodes away or towards each other.

Several spring forces are applied to a given mesh node in the smoothing procedure. Each spring constant is denoted by k_{spring}. The forces are:

- **Parent force** - force which keeps the distance of the vertex to its parent (from which it was extruded) closer to the initial extrusion length, l_{ideal} between the two.

$$f_{parent} = k_{extrusion} * \frac{l_{real} - l_{ideal}}{l_{real} + l_{ideal}} * ||parentLocation - vertLocation|| \tag{10}$$

- **Kid force** - similar to the parent force, this one keeps the distance between the node and its kid closer to the initial extrusion length.

$$f_{kid} = k_{extrusion} * \frac{l_{real} - l_{ideal}}{l_{real} + l_{ideal}} * ||kidLocation - vertLocation|| \tag{11}$$

- **Neighbour force** - force to maintain uniform spacing of mesh nodes for each layer.

$$f_{neighbour} = k_{neighbour} * \frac{(vertLocation - leftVertLocation) + (vertLocation - rightVertLocation)}{2.0} \tag{12}$$

- **Original location** - a force is added to restrict the movement of the vertex from the location at which it was placed in the mesh before smoothing. This term also ensures that the deviation of the mesh nodes from the initial surface representation is as little as possible.

$$f_{original} = k_{original} * (origialLocation - currentLocation) \tag{13}$$

These forces are summed up to give the mesh node a resultant force and the node is moved according to the total force. Several iterations (around 5-10) of smoothing are done per layer as the mesh advances towards surface interior. A limit is set to the vertex movement per smoothing iteration. The distance a vertex moves per iteration of the smoothing algorithm is kept at 5% of the extrusion length at that vertex. This helps in limiting the movement of the vertex when the initial boundary discretization is too coarse. The limitation of choosing a spring-based smoothing technique is that we have to carefully identify the spring constants, k_{spring} for our algorithm. However, once these constants are identified, they seem to work reasonably well for all the cases that we run. The value of $k_{extrusion}$ we use is 0.01, $k_{neighbour}$ is kept at 0.02 and $k_{original}$ at 0.1. Figure 11 shows an advancing layer surface mesh with and without smoothing applied to the mesh vertices. It can be seen that smoothing helps improve the vertex spacing along the advancing layer, thereby making the aspect ratio more uniform at a given front.

J. Front Collision and Redefinition

In our advancing layer mesh generation subroutine, we use extrusion length scaling, edge collapse on the front and vertex smoothing to maintain sufficiently good mesh quality and sustain boundary topology. These procedures help to avoid front collisions in many cases and at least postpone the front collisions to further layers. However, front collisions are inevitable while marching layers on a bounded surface. Hence, a subroutine to prevent front collision and redefine the front is required. We deploy two separate methods to deal with collisions.

For concave corners of the mesh which might collide onto themselves, we preemptively close off the corner before the layers collide. Consider a concave corner as shown in figure 12. Point A is a corner vertex at the mesh front CAB. The extrusion length at the front is h. The length of the edge AB is l. If point B is extruded with the current front definition, the extruded point D would cross the angle bisector of $\angle BAC$, AX and will invalidate the front. This is likely to occur for smaller aspect ratio (AR) values and small values of included angle $2\,\theta$. To avoid collision in the next front,

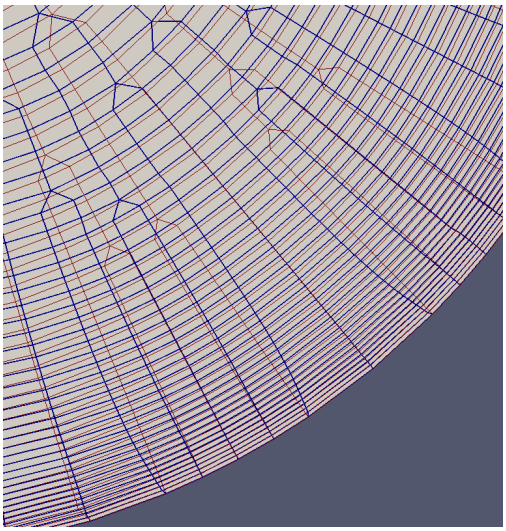


Fig. 11 Zoomed in view of smoothing applied to an advancing layer surface mesh. Dark blue lines represent the smoothed mesh and flat red lines represent the mesh without smoothing. We can see that the smoothed version of the mesh helps to attain uniform aspect ratio at each layer.

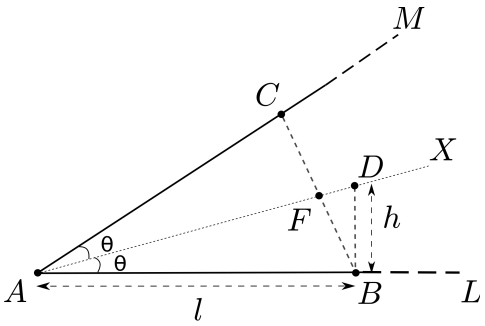


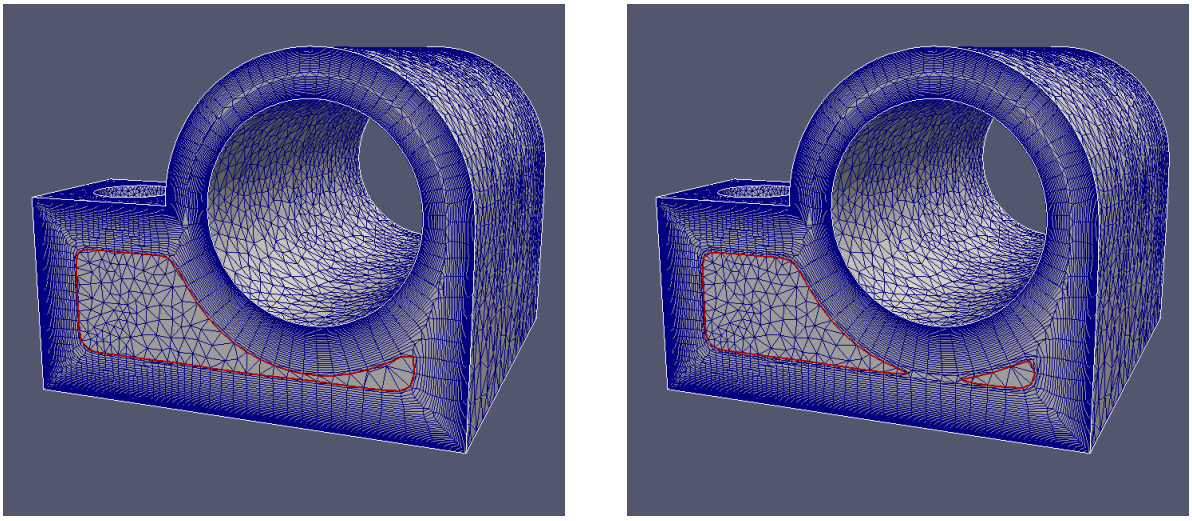
Fig. 12 Handling collision at concave corners to avoid layer overlap. Corner point A is replaced by point F so as to form the new front CFB. F is the surface-projected mid-point of points B and C. This corner collision handling is done until all the half-interior angles, θ are within the limit described by equation 14.

we insert a new point F into the mesh, located at the surface projection of the mid-point of B and C. F takes the place of A as a front vertex and this procedure is repeated until no point which is extruded from the front crosses the angle bisector of its neighbours. The value of angle θ where this occurs can be written as $\theta = \tan^{-1}(1/AR)$, where AR is the aspect ratio at the corner vertex. Hence, we identify corner as any vertex on the front which has half interior angle θ less than a threshold times this limit.

$$\theta_{corner} < threshold * \tan^{-1}(1/AR) \tag{14}$$

The threshold is taken as 1.5 for our mesh generation method to resolve all corners collision instances.

Apart from handling collisions at the corners of the mesh, we also need to take care of the front collisions which happen elsewhere. Consider Figure 13a where the advancing layer surface mesh generation algorithm is meshing one of the surfaces of the geometry shown in Figure 1 . The advancing front is shown in red in the figure. After advancing several layers from the boundary, it reaches a point where it is about to collapse onto itself. Hence, some sort of front redefinition is necessary before moving on to the next layer. To achieve this, we first iterate through all the points in the front to identify the encroached points. These points are the ones whose distance from any edge in the front, except its adjacent edges, is less than a threshold (taken as 2.4) times the extrusion length at that point. As our surface mesh is closed, we can utilize the vertex connectivity to find out the possible vertex-front edge encroachment. Hence, for a given point on the advancing front, only the edges on the front that share a triangle with that point are checked for encroachment. This saves a lot of computational cost while identifying encroached vertices. We assume that the input



(a) (b)

Fig. 13 Handling front collisions. (a) shows the advancing front which is about to collide onto itself. Using the vertex connectivity at the advancing layer and the proximity of the advancing layer vertices from each other, some of the front vertices are removed from the front and the front is redefined. The redefined front extruded to the next layer is shown in (b). From this iteration of the front, it would proceed with only the vertices on the front shown in red.

boundary discretization and surface triangulation are fine enough to resolve the complex features of the geometry, so that we don't have to iterate through all the edges on the front to find whether a given point is encroached or not.

After these encroached points and their corresponding edges are identified, we reconnect the non-encroached points so as to form the new front. The encroached points are moved out of the advancing front and the front is hence redefined. In figure 13b, we show the redefined front extruded to the next layer. In this case, the front is divided into two loops, each of which continues to march independently towards the interior of the surface. After dealing with the concave corner collision cases, this approach helps us tackle front collisions at other regions of the front and completes the front redefinition process wherever the advancing front might collide onto itself.

The concave corner detection and front redefinition routine associated with it works well to close off the corners. However, the vertex placement at such concave corners is not ideal every time. It tends to produce skinny triangular elements if there are multiple corners adjacent to each other because of bad boundary discretization. Hence, this is a limitation of the current formulation. Figure 14 shows such a case. A zoomed in view of the trailing edge of an airfoil is

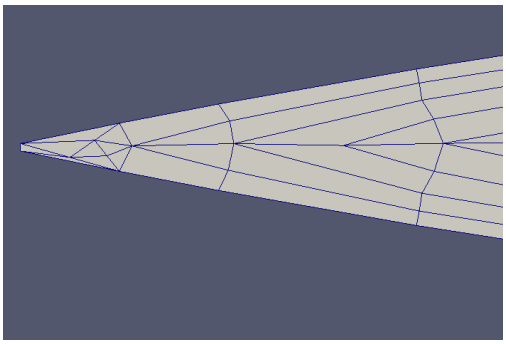


Fig. 14 Limitation of the concave corner collision detection and front redefinition subroutine. Zoomed in view of the trailing edge of an airfoil is shown. The subroutine successfully closes off the two concave corners present a the tip of the trailing edge, but produces a couple of skinny triangles in the mesh with non-ideal vertex placement.

shown. Two sharp corners at the boundary of the surface are successfully closed by the mesh and the front continues to advance towards the interior of the surface. However, the quality of the triangular elements generated is not ideal. Better vertex placement and a different smoothing subroutine for such vertices could produce better quality mesh elements for such cases.

K. Overall Mesh Generation Algorithm

In the previous section, we have detailed the steps in the advancing layer mesh generation routine. Here, we summarize the algorithm in the form of a pseudo code in Algorithm 1.

Algorithm 1 Overall Mesh Generation algorithm

```
1:  procedure SurfMesh::CreateMesh(triangulation T, extrusion length, growth ratio)
2:      Create surface geometry S ← T
3:      Initialize Advancing Layer F from Surface Boundary
4:      while F ≠ ∅ do                          ▷ Advance until Front F isn't empty
5:          AdvanceLayer(F)
6:          Validate the new Layer
7:      Export SurfMesh
8:  procedure SurfMesh::AdvanceLayer(F)
9:      Delete Encroaching Interior Vertices
10:     Redefine front F Until No Vertex Encroachment
11:     Extrude All Points of F and Insert Into Mesh
12:     Recover front F'                        ▷ Through forced edge creation between kid vertices
13:     Improve Mesh Quality by Edge Swapping   ▷ In the immediate interior of mesh
14:     Collapse Short Edges in the Front
15:     Combine triangular elements to quads
16:     Smooth Vertices
17:     Update Extrusion Length for vertices
```

III. Assumptions

We make a few assumptions on the input triangulation for our mesh generation algorithm to successfully create a valid mesh. The first one is that the triangulation of the surface is fine enough to resolve the geometric complexities in the object. If the input triangulation is too coarse, the encroaching interior vertex deletion subroutine will create mesh elements that deviate significantly from the surface or even fail. Hence, for highly curved regions of the surface, the input triangulation should be within 30 degrees deviation from the surface. However, this issue can be tackled by refining the triangulation using any isotropic refinement algorithm as long as it retains the features of the input object. Coarse triangulations are acceptable for objects without any geometric complexities.

Another assumption we have made for the input triangulation is that it would have well defined boundaries where the advancing layers can march out from. Sharp corners in the input geometry are automatically identified as boundaries of the mesh but the boundary discretization of blunt corners needs to be supplied as an input to the advancing layer surface mesh generator.

IV. Examples

We use the surface mesh generation algorithm described here to mesh some geometries. We illustrate how the advancing layer algorithm works by creating several advancing fronts from the boundary of input triangulation. We take the NACA 0018 airfoil and create a solid model by extruding it. After generating a triangulation from the model, we mesh the airfoil using our advancing layer routine. Figure 15 shows a mesh of NACA 0018 airfoil. Figure 15a and 15c show a mesh with a growth ratio of 1.05. Figure 15b and 15d show the same geometry meshed using growth ratio 1.1. The meshes can be seen to have anisotropic refinement at the leading edge, the trailing edge and at the boundary of the end caps of the airfoil. The description of the boundary of the airfoil is carried several layers into the mesh. Most of the elements generated are quads with nearly right angles. A few triangles can be seen wherever the mesh has to conform to

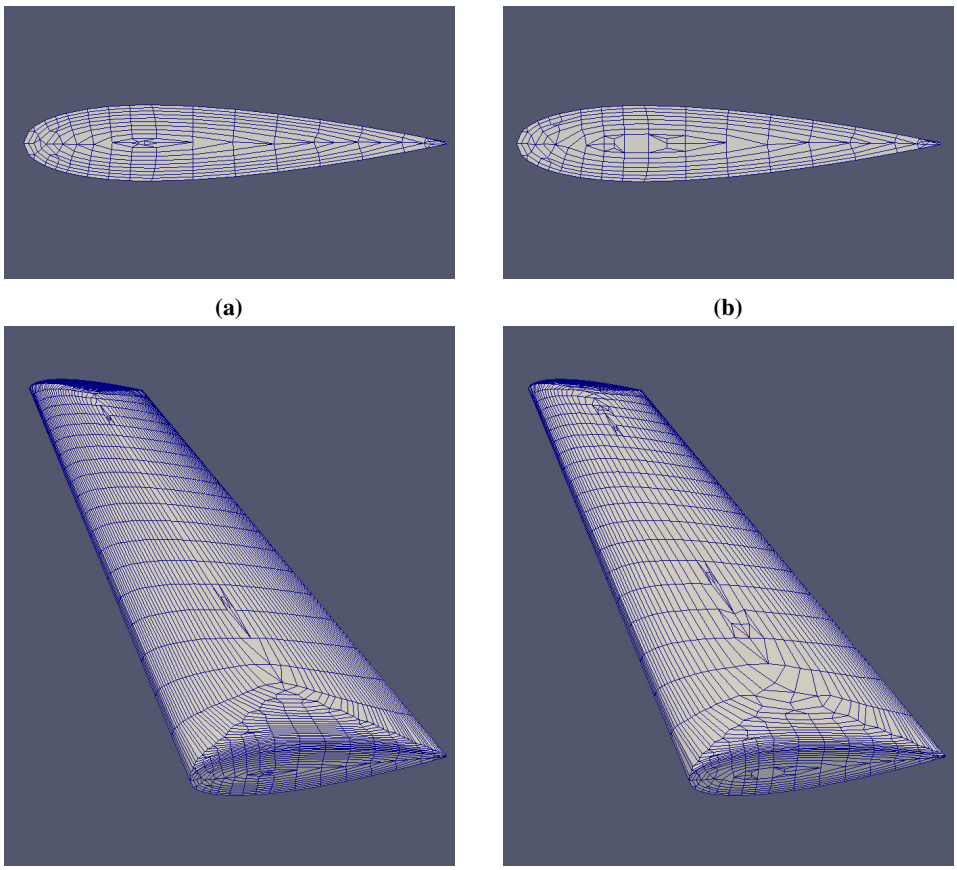


Fig. 15 NACA0018 airfoil meshed with our advancing layer surface mesh algorithm. Meshes shown on the left ((a) and (c)) correspond to a growth ratio of 1.05 while ones shown on the right ((b) and (d) correspond to a growth ratio of 1.1. Anisotropic refinement of the mesh can be seen at the leading edge, trailing edge, and both end caps of the airfoil.

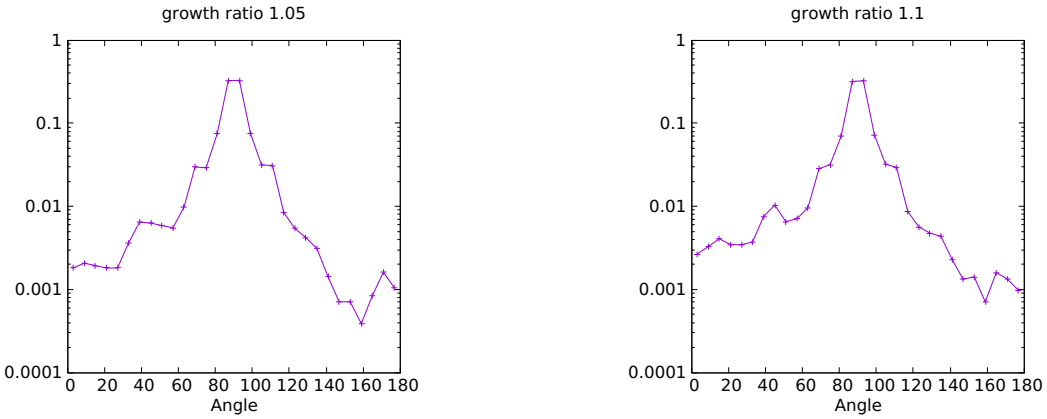
the geometry without folding onto itself.

Table 1 shows the mesh parameters of the two cases shown in figure. In the higher growth ratio case, 96.22 percent of the angles in the mesh elements are between 45 and 135 degrees while for the lower growth ratio case, 97.38 percent of the angles are in that range. In figure 16, we plot the distribution of angles for the mesh elements. As can be seen from the distribution, for both the cases, most of the angles are within a 30° range of 90°. The distribution is plotted on a log scale to highlight the variation from the mean value. The distribution of angles is more spread out for the higher growth ratio case due to a higher number of triangles in that case. A few angles are close to 0° and 180°. This is because of the sharp trailing edge of the geometry, which has to be dealt with using the corner collision algorithm described in section II.J. Overall, the algorithm produces quad dominant meshes with a very few skinny cells.

To illustrate the robustness of the advancing layer algorihtm, we run it on a geometry which has surfaces with a variety of concave corners at their boundaries. Figure 18 shows the resultant meshes generated on such a geometry for two growth ratios- 1.04 and 1.08. It can be seen that the required anisotropic stretched elements are created at the boundary of the surface. Figure 17c and 17d show magnified view of a 30° concave corner in the mesh. Even though the mesh is advancing from the corner towards the surface interior, it successfully recovers the front at each iteration of the advancing layer routine. Also, the quality of the mesh elements remains consistent as the mesh grows from the boundary. The boundary topology is successfully retained several layers into the advancing layer mesh generation process. Figure **??** shows the distribution of interior angles of the mesh elements of these meshes. The distribution is more flattened out as there are higher number of triangles in the mesh due to more number of concave corners. Still, the number of skinny triangles in the mesh is negligible. The percentage of quads generated for the low growth ratio mesh

Table 1 Mesh parameters for NACA0018 Airfoil (chord length $c = 1$)

extrusion length	growth ratio	number of quads	number of tris	% quads	% angles between 45°-135°
0.007	1.05	3750	162	95.86	97.38
0.007	1.1	2678	198	93.11	96.22



(a) **(b)**

Fig. 16 Distribution of interior angles of the cells for the meshes generated for NACA 0018 airfoil. Most of the angles of the cells in the mesh are around 90° with a very few skinny angles.

is 96.43% and for the high growth ratio mesh is 95.82%. Hence, even though the mesh has a lot of concave corners, more than 95% of the mesh elements are non-simplical quad elements.

We also illustrate the advancing layer mesh generation routine by generating a surface mesh from the geometry shown previously in Figure 1. The advancing layer surface mesh generated for the geometry is shown in Figure 19. The sub-surface boundaries are highlighted in the figure. Highly stretched elements are created at the boundary of the geometry. As the mesh grows towards surface interior, the advancing layer grows in size, hence generating the required level of anisotropy at the boundaries. The aspect ratio for this mesh at the boundary varies with the boundary vertices but is set to be around 9-10. Various views of the surface mesh are shown in the figure. This illustrates that the mesh generator is capable of handling a variety of surface topologies. A close up view of colliding layers is shown in Figure 19f . The number of vertices in the advancing layer drop as the layers start to collide. Eventually, there are no vertices left on the layer to march and the surface mesh for the sub-surface is complete.

V. Conclusion

An advancing layer mesh generation algorithm was described in the paper. The algorithm takes an isotropic input triangulation of the surface and proceeds to produce a quad-dominant anisotropic mesh one layer at a time. The meshes produced by the algorithm have the required level of anisotropy at well defined boundaries of the surface. The mesh generator is capable of producing meshes from fairly complex surface topologies with multiple concave corners. It produces meshes with primarily rectangular quad elements and a few trianglular elements. The distribution of interior angles of the mesh generated is centered around 90° with less than 1% of the angles below 20° or above 160°. Anisotropic surface meshes produced from the algorithm could be used as a reasonable input to a three-dimensional mesh generator.

References

[1] Sbardella, L., Sayma, A., and Imregun, M., "Semi-structured meshes for axial turbomachinery blades," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 5, 2000, pp. 569–584.

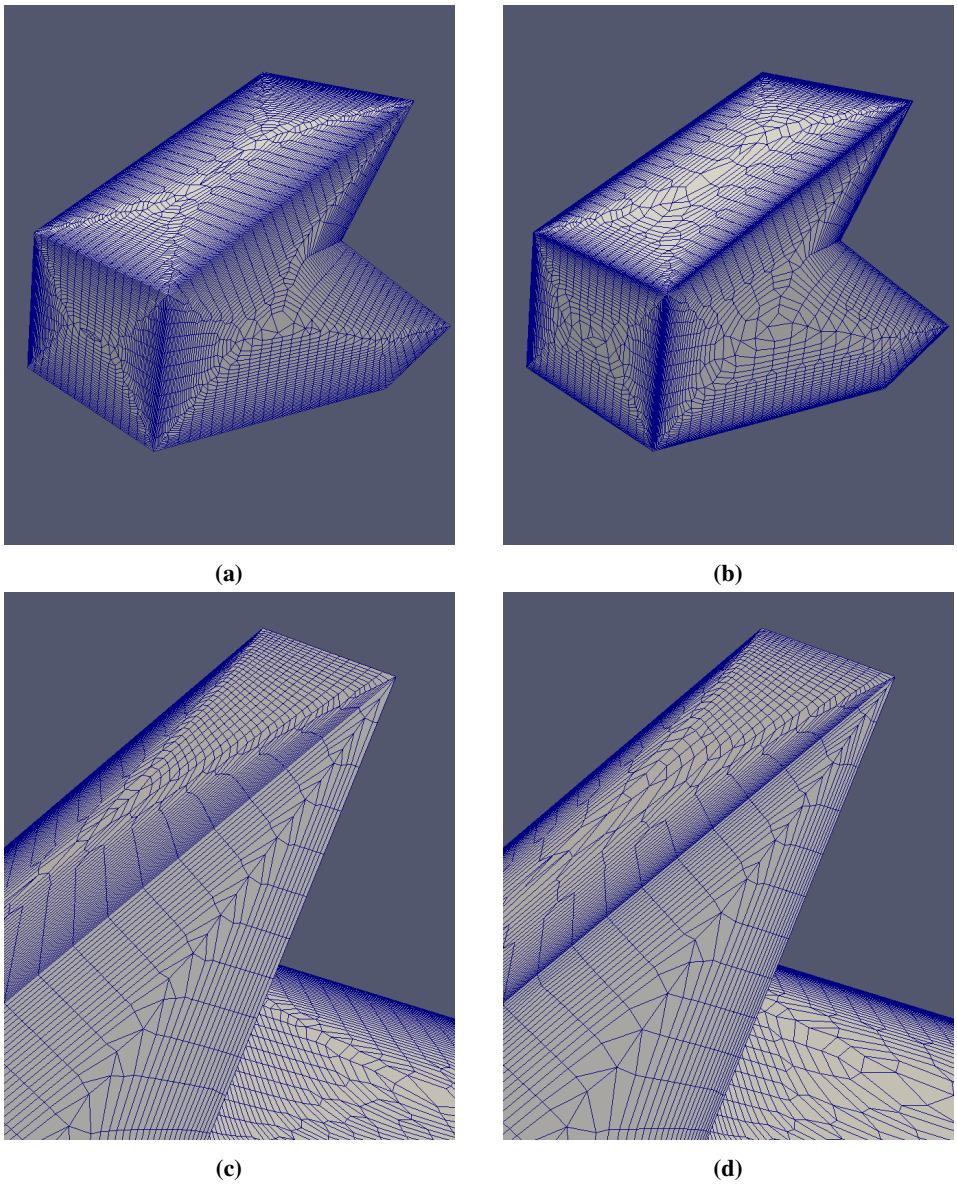
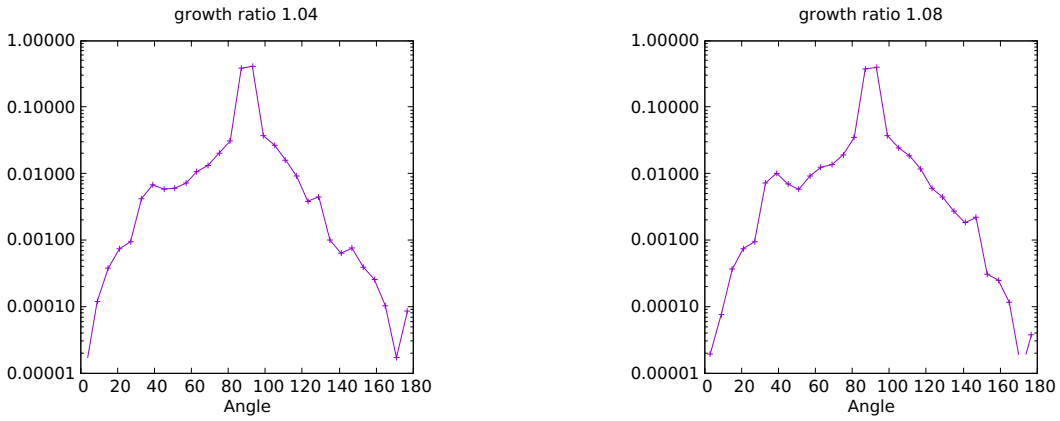


Fig. 17 Geometry with a variety of concave corners at the boundary is meshed with advancing layer surface mesh generation subroutine. On the left, the geometry is meshed with a growth ratio of 1.04 while on the right, it is meshed with a growth ratio of 1.08. The initial extrusion length of the mesh on the right is 40% less than that on the left. (c) and (d) show the magnified view of a 30° concave corner corresponding to each of the meshes.



(a) **(b)**

Fig. 18 Distribution of interior angles of the cells for the meshes generated for the geometry with a variety of concave corners (shown in figure ??). Most of the angles are in 45°-135° range with a very few skinny traingles.

[2] Dyedov, V., Einstein, D. R., Jiao, X., Kuprat, A. P., Carson, J. P., and Del Pin, F., "Variational generation of prismatic boundary-layer meshes for biomedical computing," *International Journal for Numerical Methods in Engineering*, Vol. 79, No. 8, 2009, pp. 907–945.

[3] Mavriplis, D. J., "Adaptive mesh generation for viscous flows using triangulation," *Journal of computational Physics*, Vol. 90, No. 2, 1990, pp. 271–291.

[4] Castro-Díaz, M., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic unstructured mesh adaption for flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, No. 4, 1997, pp. 475–491.

[5] Nakahashi, K., "FDM-FEM zonal approach for viscous flow computations over multiple-bodies," *25th AIAA Aerospace Sciences Meeting*, 1987, p. 604.

[6] Kallinderis, Y., Khawaja, A., and McMorris, H., "Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries," *AIAA journal*, Vol. 34, No. 2, 1996, pp. 291–298.

[7] Pirzadeh, S., "Unstructured viscous grid generation by the advancing-layers method," *AIAA journal*, Vol. 32, No. 8, 1994, pp. 1735–1737.

[8] Löhner, R., "Matching semi-structured and unstructured grids for Navier-Stokes calculations," *11th Computational Fluid Dynamics Conference*, 1993, p. 3348.

[9] Connell, S. D., and Braaten, M. E., "Semistructured mesh generation for three-dimensional Navier-Stokes calculations," *AIAA journal*, Vol. 33, No. 6, 1995, pp. 1017–1024.

[10] Garimella, R. V., and Shephard, M. S., "Boundary layer mesh generation for viscous flow simulations," *International Journal for Numerical Methods in Engineering*, Vol. 49, No. 1-2, 2000, pp. 193–218.

[11] Ito, Y., and Nakahashi, K., "Unstructured Mesh Generation for Viscous Flow Computations." *IMR*, 2002, pp. 367–377.

[12] Sahni, O., Jansen, K. E., Shephard, M. S., Taylor, C. A., and Beall, M. W., "Adaptive boundary layer meshing for viscous flow simulations," *Engineering with Computers*, Vol. 24, No. 3, 2008, p. 267.

[13] Ito, Y., Shih, A. M., Soni, B. K., and Nakahashi, K., "Multiple marching direction approach to generate high quality hybrid meshes," *AIAA journal*, Vol. 45, No. 1, 2007, pp. 162–167.

[14] Aubry, R., and Löhner, R., "Generation of viscous grids at ridges and corners," *International journal for numerical methods in engineering*, Vol. 77, No. 9, 2009, pp. 1247–1289.

[15] Mavriplis, D., "Unstructured grid techniques," *Annual Review of Fluid Mechanics*, Vol. 29, No. 1, 1997, pp. 473–514.

[16] Aftosmis, M., Gaitonde, D., and Tavares, T. S., "On the accuracy, stability, and monotonicity of various reconstruction algorithms for unstructured meshes," 1994.

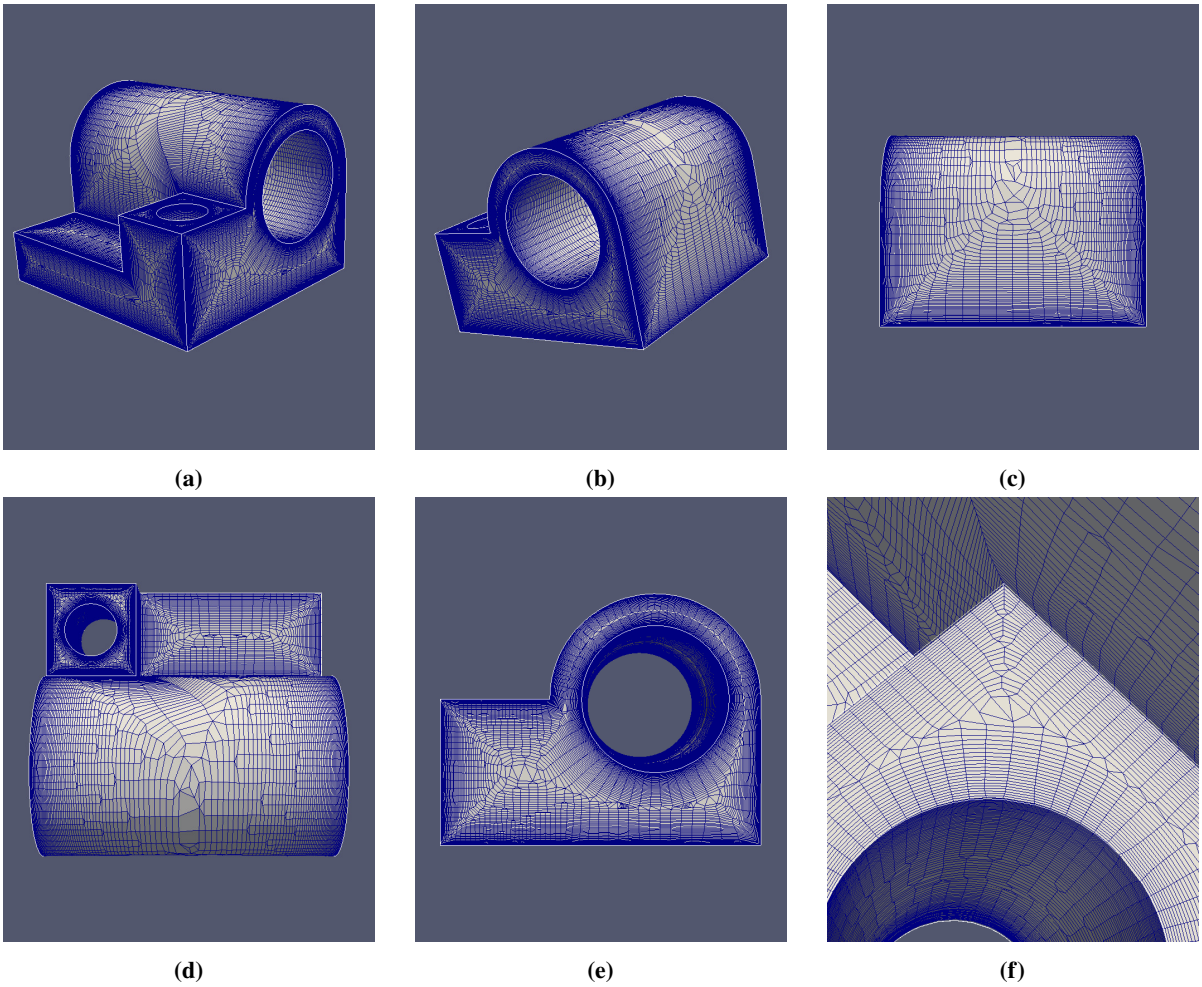


Fig. 19 Geometry shown in Figure 1 is meshed using the advancing layer surface mesh generation routine. The algorithm is able to tackle a variety of surface topologies while generating anisotropic meshes. Given level of anisotropy can be obtained on the boundaries of the mesh with the algorithm.

[17] Blacker, T. D., and Stephenson, M. B., "Paving: A new approach to automated quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 811–847.

[18] Zhu, J., Zienkiewicz, O., Hinton, E., and Wu, J., "A new approach to the development of automatic quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 849–866.

[19] George, P.-L., and Borouchaki, H., "Delaunay triangulation and meshing," 1998.

[20] Chen, H., and Bishop, J., "Delaunay triangulation for curved surfaces," *Meshing Roundtable*, 1997, pp. 115–127.

[21] Owen, S. J., "A survey of unstructured mesh generation technology." *IMR*, 1998, pp. 239–267.

[22] Cuillière, J.-C., "An adaptive method for the automatic triangulation of 3D parametric surfaces," *Computer-aided design*, Vol. 30, No. 2, 1998, pp. 139–149.

[23] Tristano, J. R., Owen, S. J., and Canann, S. A., "Advancing front surface mesh generation in parametric space using a riemannian surface definition." *IMR*, 1998, pp. 429–445.

[24] Lan, T., and Lo, S., "Finite element mesh generation over analytical curved surfaces," *Computers & Structures*, Vol. 59, No. 2, 1996, pp. 301–309.

[25] Lau, T., Lo, S., and Lee, C., "Generation of quadrilateral mesh over analytical curved surfaces," *Finite Elements in Analysis and Design*, Vol. 27, No. 3, 1997, pp. 251–272.

[26] Tautges, T. J., "The Common Geometry Module (CGM)." Tech. rep., Sandia National Laboratories, 2004.

[27] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., "Mesh optimization," Tech. rep., Washington Univ Seattle Dept OF Computer Science And Engineering, 1994.

[28] Canann, S. A., Tristano, J. R., Staten, M. L., et al., "An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes." *IMR*, Citeseer, 1998, pp. 479–494.