Three-Dimensional Prism-Dominant Mesh Generation for Viscous Flows Around Surface Slope Discontinuities

Juliette Pardue* and Andrey Chernikov†

Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA

The NASA CFD Vision 2030 Study has identified mesh generation to be a significant bottleneck in the CFD pipeline. Human intervention is often required due to a lack of robustness and automation while generating the mesh. An automated approach to generating unstructured three-dimensional, prism-dominant meshes for viscous flows is presented. Meshes comprised of prisms are advantageous because they yield a more accurate solution and have fewer elements than their purely-tetrahedral mesh counterpart. An extrusion-based approach using multiple normals is used where anisotropic prisms are formed from the surface mesh facets and blend prisms are used to fill the cavities between multiple normals. Multiple normals are needed at a node to satisfy the visibility requirement for all incident surface facets. These multiple normals arise at regions of the surface mesh where there are convex discontinuities in the slope of the surface across edges. Nodes where blend region meshes must be formed are classified using an exhaustive enumeration scheme based on the number of convex edges and concave edges that are incident upon the node. Templates are presented to robustly mesh all the enumerated types of blend regions that occur at ridges, cusps, and corner nodes, including non-Lipschitz nodes. Intersections are detected in the boundary layer using an efficient spatial partitioning tree and are treated using Laplacian smoothing of the normal vectors or by reducing the total height of the boundary layer in the identified regions. The remainder of the volume is then tetrahedralized with an isotropic Delaunay mesh generator.

I. Introduction

Mesh generation is a step in the iterative pipeline for analyzing aerospace vehicles as shown in Figure 1. After an analysis of the partial differential equations (PDE) solution on the mesh, the mesh is refined to yield a more favorable error estimation, which is typically a faster operation than generating an entirely new mesh. For computational fluid dynamics (CFD), anisotropic mesh adaptation[1,2] is needed to refine the elements in regions where there is a higher degree of gradation in the flow velocities in one direction. Assuming proper care for the refinement steps, this new mesh will be more accurate in the desired regions with respect to the PDE solution.

Since the end goal is to generate a mesh which accurately and efficiently fits the PDE, the time to achieve this goal is dependent on the number of iterations through the pipeline of mesh generation to PDE solver to analysis. Clearly, this iterative process needs an initial mesh to begin the process. If the initial mesh closely represents the PDE, then fewer iterations through the pipeline are required to achieve a suitable solution. However, if the initial mesh is highly inaccurate with respect to the PDE, then the first iterations through the pipeline will present numerous areas which require refinement. It will take the unsuitable mesh more iterations to reach a similar quality as the suitable initial mesh. The initial mesh sets the pace for the remainder of the iterations through the pipeline as well as the amount of refinement work. For CFD, we need an initial mesh that has highly stretched elements along the direction of the flow to provide the most CPU savings for the PDE solver. An appropriate anisotropic mesh will also require a fewer number

*jpardue@cs.odu.edu
†achernik@cs.odu.edu

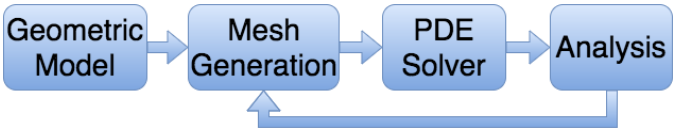


Figure 1. CFD analysis pipeline.

of iterations through the pipeline, thus yielding the fastest overall execution time. Without a well-resolved initial boundary layer, the error estimator may not be able to determine any regions of the mesh to be refined because there is not enough resolution to determine flow characteristics.

The NASA CFD Vision 2030 Study[3] has identified that mesh generation and adaptivity are significant bottlenecks in the CFD pipeline. Hardware advances have allowed for higher resolution simulations, and while flow solver software has matured at a faster rate, mesh generation has become increasingly difficult due to the demand for larger meshes over more complex geometries. The NASA CFD Vision 2030 Study states that mesh generation routinely requires human intervention due to a lack of robustness in treating complex geometries. One of the key findings is a lack of automation, and inherently including robustness, in the mesh generation process: "A single engineer/scientist must be able to conceive, create, analyze, and interpret a large ensemble of related simulations in a time-critical period, without individually managing each simulation, to a pre-specified level of accuracy." Clearly a mesh generator is needed that can be parameterized, executed, and run (without any human interaction during runtime) via a script in order to submit a batch collection of related jobs. Chawner et al.[4] discusses in more detail the geometry and meshing concerns in the NASA CFD Vision 2030 Study along with a history of three decades of meshing evolution which identifies the common unaddressed capabilities over the years. Human intervention has always been a problem with mesh generation and automation has been a request since 1984. In addition to robustness still being requested in the mid-1990s, more efficient mesh generators were also desired because the time to generate a series of related grids was prohibitively long. Chawner et al. also identify the desire for meshes over more complex geometries along with physics-coupled meshing, such as generating an anisotropic boundary layer for Reynolds-Averaged Navier-Stokes flows. Various mesh generation codes use various quality metrics to determine unacceptable elements and various solver codes are more sensitive to certain quality criteria. To summarize, mesh generators need to be able to produce large "high-quality" grids quickly around complex geometries without any human intervention, where "high-quality" is dependent upon the flow solver's discretization.

The summary of the panel discussion at AIAA's 2015 SciTech Conference by Chawner et al.[5] further discusses the aforementioned needs for mesh generators, including mesh generation's utilization of High Performance Computing (HPC) advances. The shortcomings in robustness and the inability for mesh generators to create a valid mesh around complex configurations has become a roadblock for parallel mesh generation. For example, if a mesh generator fails to produce a valid mesh 10% of the time, then attempting to run the mesh generator in parallel with 10 processes will typically lead to a failure. Due to a lack of advancement in fault-tolerance for HPC, if even just one meshing process fails, then all processes abort and a mesh is not produced. Karman, Wyman, and Steinbrenner discuss the challenges of engineering a mesh generator from a commercial perspective,[6] and also cite robustness and automation as one of the main difficulties. Underutilization of HPC resources is due to the scalability and robustness of certain mesh generation approaches, such as the advancing-front method which requires a lot of synchronization to ensure that multiple processes do not create meshes that overlap. In addition to the poor parallel scalability, the advancing-front method is based on heuristics and is not guaranteed to terminate, so this approach is not a good candidate to parallelize. It is clear that the priority for mesh generators has become automation, which requires robustness.

A. Related Work

There have been many developments in the mesh generation field for creating anisotropic meshes. The state-of-the-art approach to creating the initial mesh of highly stretched elements is the advancing-front procedure where elements are extruded from the surface mesh in the normal direction for each surface facet. Many implementations of these methods grow the mesh from the geometry model one layer at a time, where each new element in each layer is checked for intersections with the remainder of the front. The remainder of the

volume, away from the model, can be meshed with an isotropic mesh generator.[7,8] Some approaches use a single normal for each model node, so the boundary layer is treated as an inflation of the surface mesh.[9–12] Aubry et al. present a comparison between the different normal vectors[13] that can be chosen for the single normal for each node. Connell and Braaten[9] create surface meshes on viscous and non-viscous walls using an advancing front procedure, then prisms are extruded from the viscous walls. After the extruded prisms are created, then the exposed quadrilateral faces are collapsed in a stairstep pattern so that only triangular faces are exposed for the isotropic volume mesh generator. Hassan et al.[10] also uses an advancing front procedure to create the initial surface mesh. Anisotropic tetrahedrons are extruded in the normal direction from the surface mesh. At each layer the normals are recalculated and smoothed. After the boundary layer generation terminates, an isotropic advancing front procedure is used to tetrahedralize the remainder of the volume. Pirzadeh[14] uses the advancing layers method with a single normal per node to generate anisotropic tetrahedra along surface vectors of a predetermined triangular surface mesh. One layer is created at a time, and a layer terminates once it gets too close to another front or until a certain grid quality, dictated by the background mesh, is satisfied. After the viscous boundary layer is generated, then the traditional isotropic advancing front procedure is used to fill the remainder of the domain.

Martineau et al.[11] uses an anisotropic, quadrilateral-dominant surface mesh as the initial surface that the prisms will be extruded from. A method of gradation control is also presented which terminates an advancing layer if the previous element is roughly isotropic, or if the new element is too close to other features of the mesh. The marching vector directions are smoothed along with the growth rate in concave regions. Once the boundary layer has been generated, a series of pyramids and tetrahedrons are used to cover each exposed quadrilateral face from the isotropic tetrahedral volume mesh generator. Marcum, Alauzet, and Loseille[12] can produce mixed-element meshes of tetrahedrons, triangular prisms, and hexahedrons. The boundary layer uses varying growth rates and acceleration rates for different regions of the mesh based on the characteristics of the input surface mesh. Anisotropic metric blending is used when two fronts start approaching each other. Their paper looked at the effect of computing the normal in different ways. The normals that they computed and evaluated were: the average normal of incident faces, the most visible normal which minimizes maximum angle between each face normal and the optimal normal, and the least squares normal which minimizes the deviation between each face normal and the optimal normal. They also tested two additional cases which both use a few iterations of Laplacian smoothing: the most visible normal + Laplacian smoothing and the least squares normal + Laplacian smoothing. These two cases were the normals that produced the highest quality meshes with the fewest number of layers that terminated early for many practical cases.

The single normal approach, even using Laplacian smoothing, creates poorly shaped volume elements around nodes of the surface mesh that have large deviations between its incident surface facets' normal vectors, such as the trailing edge region. If a facet's nodes' chosen marching vectors have an almost zero dot product with the surface facet's original normal vector then the extruded prism will be flattening out and approaching zero volume. This is known as the visibility requirement because the normal of each node for a triangle must be visible to the computed normal of the triangle. Figure 2 shows the grid lines around the sharp trailing edge of a cone, commonly seen on payloads, that has been generated using a single normal per node and smoothing those normal direction. The grid lines show that some of the volume elements extruded from the surface elements incident upon the trailing edge node have a near zero volume since their grid lines are perpendicular to the surface element's normal.

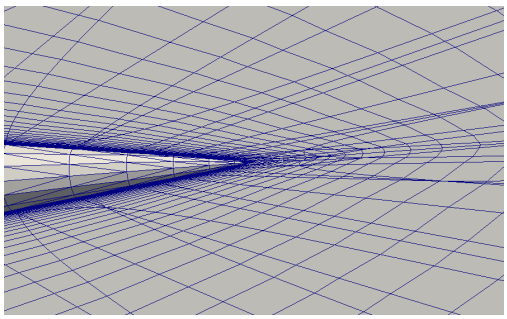


Figure 2. Grid lines for a trailing edge slice.

Many authors began introducing multiple normals to treat these regions that have large discontinuities in the slope of the surface.[15–19] Normals are chosen that satisfy a certain visibility requirement such as the dot product between the node's chosen normal and an incident surface facet's calculated normal being above a threshold. When all of a node's incident surface facets satisfy the visibility requirement, then only one normal is needed for the node; otherwise, multiple normals are needed. However, when multiple normals are introduced, the cavity between these multiple normals has highly anisotropic faces exposed. The isotropic volume mesh generator cannot handle these highly stretched faces, so blend meshes need to be created to fill the cavity and cover the anisotropic faces.

Garimella's dissertation[15] presents an anisotropic tetrahedral mesh generator using multiple normals that can treat non-manifold geometry. As seen in previous work using the single normal approach, Garimella also uses Laplacian smoothing for the marching vectors. Garimella defines templates for treating the blend regions incident along edges, where both nodes have two normals, and cusp edges, where one node has two normals and the other node has one normal. However, Garimella does not discuss the blend regions incident upon corner nodes and proposes "that the vertex blends be created using general mesh generation techniques tailored for the purpose of filling these gaps."[15] Jansen, Shephard, and Beall[16] present an extrusion-based approach for generating anisotropic prism-dominant meshes using multiple normals. Edge blend regions are treated with hexahedrons, and neighboring edge blend regions must have the same number of hexahedrons so that the shared quadrilateral faces are compatible. Vertex blends are not discussed in detail. Instead of using Laplacian smoothing, a method of gradation control is used to determine if layers need to be added or removed. Ito, Shih, and Soni use an advancing front method to generate the anisotropic prisms, and blend hexahedrons at sharp convex edges.[17] Concave regions are treated using Laplacian smoothing of the marching directions. Their approach can handle a surface mesh with mixed triangles and quadrilaterals because prisms are used. The triangular prism and hexahedron extruded from a neighboring triangle and quadrilateral face, respectively, will be compatible because they share the same interior quadrilateral face incident upon the common edge of the surface triangle and surface quadrilateral, see Figure 3. A method of gradation control is used to terminate regions of the front once the resulting elements are isotropic, or if the front approaches other features of the mesh. Once the outer layer of the boundary layer is isotropic, an isotropic advancing front method is used to fill the remainder of the volume.

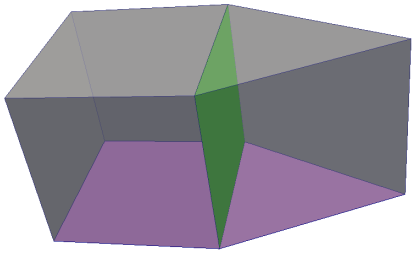


Figure 3. Compatibility of shared quadrilateral face (green) for prisms extruded from neighboring quadrilateral and triangle (purple).

Aubry and Lohner[18] present a method which enumerates many different types of surface slope discontinuities that arise from an input surface mesh where multiple normals are allowed. These classifications are divided into three groups (ridges, cusps, and corners) based on the convexity or concavity of a node's adjacent edges. The different ridges and corners are treated on a case-by-case basis based on the number of concave and convex edges that come into the node. Surfaces can be defined as viscous or non-viscous, and anisotropic tetrahedrons are extruded from the triangular surface mesh. Laplacian smoothing is used to treat concave regions of the mesh. Non-Lipschitz vertices are not discussed in this work. A non-Lipschitz vertex is a node that cannot be treated using a single normal because there is no single normal that is visible from every incident surface facet. Aubry et al.[19] builds upon the previous work by using generalized spherical Voronoi diagrams to determine the topology of the boundary layer around ridges and corners. Compared to Aubry and Lohner's earlier work,[18] this approach provided a more reliable technique to treat complex corners, where multiple convex and multiple concave ridges are incident upon a single node.

In this paper, an approach to generating a boundary layer mesh that is prism-dominant is presented. Prism-dominant meshes have been shown to have better accuracy than their purely-tetrahedral counterparts. Park and Anderson compared the convergence and accuracy of solutions computed from a finite-volume scheme on hexahedral meshes, a finite-element scheme on tetrahedral meshes, and a finite-volume scheme on tetrahedral meshes.[20] Varying mesh sizes were used and the largest tetrahedral mesh for the finite-volume scheme was not able to accurately resolve the same flow features as the smallest hexahedral mesh for the finite-volume scheme. The velocity profiles were also compared and the finite-volume scheme over hexahedral meshes and the finite-element scheme over tetrahedral meshes both agree with the reference solution. However, there is a large disparity in the velocity profile for the finite-volume scheme over tetrahedral meshes compared to the reference solution. This approach also addresses the main concerns of the NASA CFD Vision 2030 Study and responses which requested a mesh generator that is robust, automated, and efficient. This approach:

- generates a high-fidelity, anisotropic boundary layer mesh from a user-defined growth function
- uses multiple nodal normals at convex surface slope discontinuities
- generates blend region meshes between multiple normals for ridges, cusps, and corners
- resolves intersections in the anisotropic boundary layer
- generates an isotropic, graded Delaunay inviscid region mesh
- is a push-button mesh generator, no human interaction is required after startup

II. Methodology

The first step for extruding a viscous boundary layer from the surface mesh is to analyze the surface mesh's edges. The angle at each edge is calculated as the angle between the two incident surface triangles' normals. Based on the winding of the triangles, it is determined if this edge causes a concavity or convexity. All concavities above five degrees (to account for small deviations on the surface) are marked as concave ridges. For convex edges, if this angle is above a user-defined threshold, then the edge is marked as a convex ridge. Multiple normals are needed at these convex ridges. The ridges at each node are analyzed to determine if the node is a ridge point, cusp point, or corner point based on if a node is incident on one ridge, two ridges, or three or more ridges, respectively. Triangular prisms are extruded from the surface triangles. Prism-dominant blend meshes are formed along ridges, cusps, and corners. Intersections are resolved using Laplacian smoothing or by reducing the local height of the boundary layer. Finally, exposed quadrilateral faces on the exterior of the boundary layer are treated before being passed off to the isotropic tetrahedral mesh generator. Figure 4 shows the pseudocode for the flow of the algorithm. The different classifications for nodes (Line 9) are shown in Table 1. In this section, these different node configurations are shown on models and broken down based on the number of incident ridges and the convexity or concavity of the incident ridges. Their specific mesh generation process is described and the resulting boundary layer is shown.

```
1  FOR EACH surface mesh edge
2      Compare signed angle between the two adjacent triangles' normals and classify edge
3      IF angle is convex and above a user-defined threshold THEN convex ridge
4      ELSE IF angle is concave and at least 5 degrees THEN concave ridge
5      ELSE normal edge
6      END IF
7  END FOR EACH
8  FOR EACH surface mesh node
9      Classify the node by the number of incident convex ridges and concave ridges
10     IF #convex == 0 THEN compute a single normal
11     ELSE IF #convex == 1 THEN use two normals, one per triangle of the convex ridge
12     ELSE use #convex normals, one for each set of triangles between successive convex ridges
13     END IF
14     Use node classification to determine initial blend regions
15 END FOR EACH
16 Handle intersections
17 Create and index points along each normal with respect to growth function
18 Create and index extruded triangular prisms for each surface mesh triangle
19 Create and index the blend volume elements for each blend region cavity
20 Split exposed outer layer quadrilateral faces and form isotropic inviscid region
```

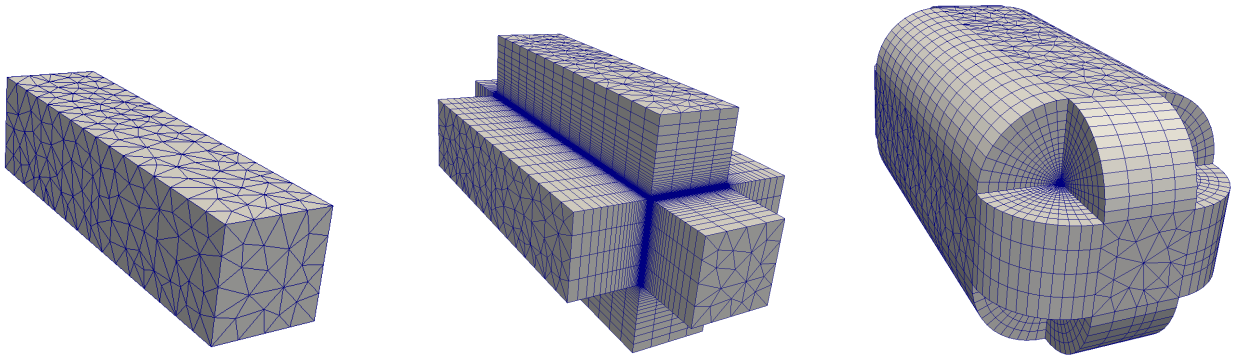
Figure 4. Psuedocode showing the overall flow of the algorithm, with an emphasis on node classification

A. Surface Triangle Extrusion

Triangular prisms are extruded from the surface triangles. For models with C^1 continuity, no blend regions will be exposed and the average of unique normals is used for each nodal normal. A visibility check is performed to ensure that the normal is visible to the computed normal of all incident surface triangles. If the dot product of these two normals is not positive, then the prism will be invalid. For models with C^0 continuity like the box model in Figure 5(a), anisotropic quadrilateral faces will be exposed in the blend regions. The exposed quadrilateral faces appear in regions where a node has multiple normals, such as along the 12 edges of a box, shown in Figure 5(b). The inviscid region is generated using an off-the-shelf, isotropic Delaunay tetrahedralizer. The tetrahedral mesh generator requires that all exposed boundary faces be triangular so that boundary layer prisms do not need to be subdivided. These exposed triangular faces should be roughly isotropic so that poor-quality elements are not created. The cavities formed by these blend regions needs to be meshed so that there are no highly-stretched faces or quadrilateral faces exposed to the isotropic mesher.

Table 1. The exhaustive classifications for nodes, based on the number of incident convex and concave ridges.

Classification	# Convex Ridges	# Concave Ridges
Normal Node	0	0
Convex Cusp Node	1	0
Concave Cusp Node	0	1
Convex-Convex Ridge Node	2	0
Convex-Concave Ridge Node	1	1
Concave-Concave Ridge Node	0	2
Pure Concave Corner Node	0	3+
Mostly Concave Corner Node	1	2+
Dual Convex Corner Node	2	1+
Pure Convex Corner Node	3+	0
Complex Corner Node	3+	1+



(a) Surface mesh.

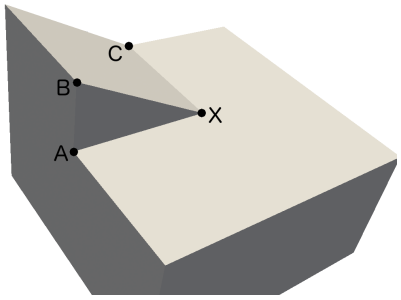
(b) Extruded triangular prisms.

(c) Boundary layer with convex ridge blend prisms.

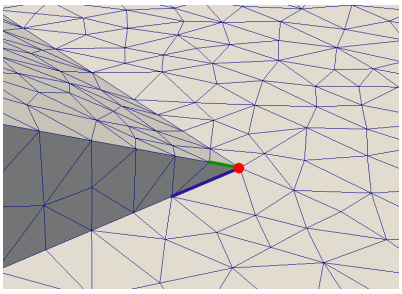
Figure 5. The exposed anisotropic quadrilateral faces in the blend region around a box geometry surface mesh after the triangular prisms have been extruded. The final image shows the blend hexahedra added along the ridge cavities.

B. Ridges

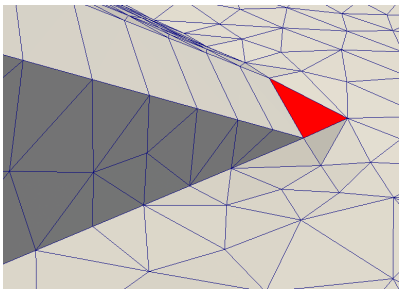
A ridge point is defined as a point that is incident upon exactly two ridges. In this section, the three different types of ridge points are defined and discussed. These three types depend on the convexity of the two incident ridges: convex-convex, convex-concave, and concave-concave. Nodes incident upon convex ridges need multiple normals to satisfy the visibility requirements at the convexity. A single normal would cause poorly shaped elements to be formed around the convex ridge. Multiple normals will need to be added so that the outer layer blend prisms have approximately the same volume as the neighboring outer layer triangular prisms that are extruded from the surface triangles incident upon the convex ridge. The exposed blend cavities with the exposed quadrilateral faces in Figure 5(b) are all in regions with convex-convex ridge points. Figure 5(c) shows the blend hexahedra added to provide a smooth transition between the triangular prisms extruded from the surface triangles. All neighboring convex ridges must have the same number of hexahedrons so that the quadrilateral faces are conformal. There are 12 groups of neighboring convex ridges (corresponding to the 12 edges of a box), separated by the corner node cavities (not shown for visual clarity). A single triangular prism is used instead of a hexahedron for the first layer element that forms from the convex ridge. The six vertices of the triangular prisms are the two surface nodes incident upon the convex ridge and the first interior vertex of the four normals (two per node).



(a) Box-wedge model around concave-convex ridge point X, concave ridge XA, convex ridge XB, and normal edge XC.



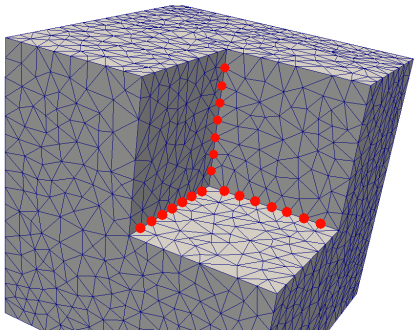
(b) Zoomed in view of the surface mesh showing the concave-convex ridge point and incident ridges.



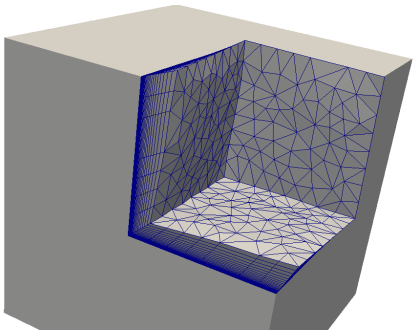
(c) Extruded triangular prisms from the concave-convex ridge point.

Figure 6. The surface and mesh around a box-wedge model. The surface mesh has a single concave-convex ridge point connected to a series of convex-convex ridges. The boundary layer is shown and the single stack of triangular prisms incident upon the concave-convex ridge point.

An uncommon case is the concave-convex ridge point, shown in Figure 6(a), which has one convex ridge, XB, and one concave ridge, XA. Figure 6(b) shows the zoomed in surface mesh around the concave-convex ridge point with the ridges shown. The node adjacent to the concave-convex ridge point across the convex ridge has two normals, but the concave-convex ridge point only has one normal. So blend elements need to be formed along this convex ridge where one node has two normals and the other node has one normal. This is known as a convex cusp, which will be detailed in the next section. Figure 6(c) shows the boundary layer around the box-wedge surface. The stack of convex cusp blend elements is below the red enclosing triangle. A series of blend hexahedra are adjacent to the convex cusp while the region around the concave ridge uses a single normal per node which are smoothed using Laplacian smoothing. Figure 7(a) shows a series of concave-concave ridges along the interior of the cavity. Since the region is concave, we know that all of the surface triangles incident along the ridge will be visible with a single normal. The average normal of the unique normals is calculated for the ridge nodes. Figure 7(b) shows the resulting boundary layer for the cavity after Laplacian smoothing.



(a) Surface mesh showing the concave-concave ridge points.



(b) Extruded and smoothed triangular prisms in the cavity.

Figure 7. The surface mesh around a cube model missing an octant with the concave-concave ridge points identified. The boundary layer is shown only for the triangular prisms in the cavity.

C. Cusps

A cusp point is defined as a point that is incident upon exactly one ridge. The cusp point can either be concave or convex with respect to the incident ridge. For a concave cusp, the concave cusp node is in a region where all of its incident surface triangles have approximately the same normal direction. If the node adjacent to the concave cusp point is only incident upon concave ridges, then the region around the concave ridges including the concave cusp point will only need one normal per node and will have that normal direction smoothed with Laplacian smoothing. For convex cusps, as shown in Figure 6(b),

the convex cusp node will only have one normal, but the node adjacent to the convex cusp node will have at least two normals. There may be more normals since the convex-convex ridge adjacent to the convex cusp may have been split as seen by the multiple hexahedra extruding from the convex-convex ridges in Figure 5(c). The elements formed from a convex cusp are all triangular prisms, except the first element. The first element is a five-vertex wedge, Figure 8. The top face is triangular which conforms to the second layer triangular prism, and the triangular face incident upon the convex-convex ridge point conforms to the adjacent convex ridge's first layer triangular prism. The quadrilateral faces of the five-vertex wedge are joined at two edges and conform to the quadrilateral faces of the first layer triangular prisms extruded from the neighboring surface triangles. Most flow solvers do not support an arbitrary five-vertex wedge, so the element needs to be divided without splitting the two quadrilateral and two triangular faces. There is no possible way to subdivide the five-vertex wedge into basic elements, so a steiner point is added in the interior and connected to the two triangular faces and two quadrilateral faces to form two tetrahedra and two pyramids, respectively. However, since the two quadrilateral faces share three vertices, the resulting pyramids may have a near zero volume, so the steiner point needs to be placed near the shared edge of the two triangular faces. Each endpoint of the shared edge is incident upon exactly one of the quadrilateral faces. The steiner point will be inserted along a vector from the midpoint of the shared edge. The vector will be the average of the two triangular faces' interior pointing normals. The steiner point should be very close to the shared edge, but not the centroid of the five-vertex wedge. This ensures that the two pyramids do not have a zero volume.

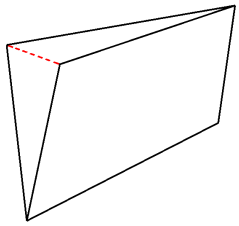


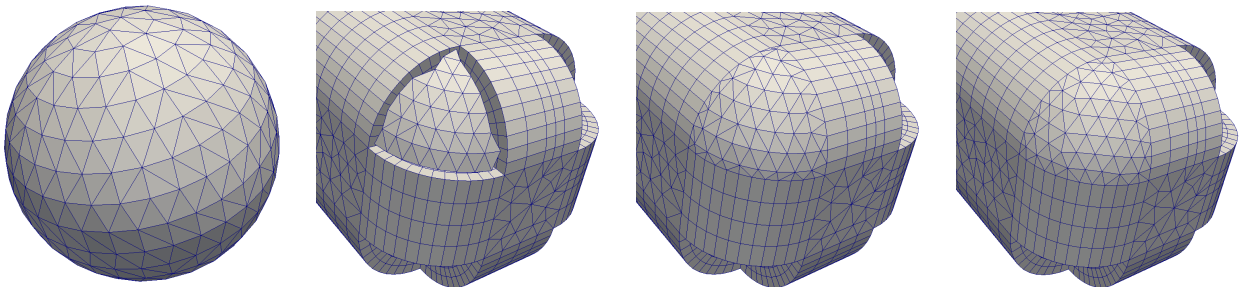
Figure 8. A five vertex wedge is decomposed after a steiner point is inserted in the interior near the midpoint of the red edge.

D. Corners

Corner discontinuities are defined at a node which has at least three incident ridges. There are many different combinations that can arise for corner configurations depending on the number of incident convex ridges and concave ridges. In this section, the different corner cases are broken down into groups and the different configurations are shown and meshed. When a corner node has multiple normals, a cavity will exist at the corner node. The shape of the cavity is dependent upon the incident concave and convex ridges as well as the connectivity of the blend prisms formed along the ridges.

1. Pure Concave & Pure Convex Corner Nodes

The simplest case to handle is the pure concave corner, where there are three or more incident concave ridges. Figure 7(a) has a single pure concave corner node where the three red dotted lines meet at the interior of the cavity. The average normal of the incident surface triangles' normals are used as the single normal for this node. The node's normal will be included in the group of normals in the concavity that will be smoothed using Laplacian smoothing. The opposite case is the pure convex corner, where there are three or more incident convex ridges. Figure 5(c) shows the cavity formed by three incident convex ridges. There are many exposed quadrilateral faces due to the multiple hexahedra from the added normals. The vertices



(a) A uniform sphere which will overlay the cavity.

(b) The clipped points and triangles inside the cavity.

(c) The stitched gap.

(d) Resulting surface after Laplacian smoothing.

Figure 9. Steps to generate the outer layer topology for the blend region at pure convex corner nodes.

on the outer layer of the cavity are fixed and lie on the surface of a sphere since all of the normals that form the cavity originate from the same corner point and the total height of the boundary layer is constant at this step. Figure 9 shows the steps for generating the blend mesh around the pure convex corner. First a regular spherical mesh is generated with the radius as the height of the boundary layer and the origin as the corner node. Next, all of the vertices and triangles inside the cavity are chosen. This exposes a gap between the spherical surface mesh and the outer vertices of the pure convex corner cavity which needs to be triangulated. The final step is a couple iterations of Laplacian smoothing so that neighboring triangles have roughly the same area. After the outer layer topology has been determined, then normal vectors are cast down from the outer layer points down to the corner node. Stacks of triangular prisms are formed at each layer, except the first, below each of the outer triangles. This ensures that the exposed quadrilateral faces of the neighboring hexahedra are conformal with the quadrilateral faces of the triangular prisms. The first layer elements are tetrahedra so that the triangular faces are compatible with convex ridges' first layer triangular prisms. This approach is preferred over an advancing front method, which is based on heuristics and has no proof of termination because the front may not be able to be closed. This approach can always create an outer layer surface mesh covering the cavity. This is because there will always exist a valid triangulation for the cavity. We start with a valid triangulation on the uniform sphere. Some subset of points will lie inside the cavity, but not all. This creates the gap region which always has a valid triangulation. On the rare occasion the subset of points inside the cavity is empty, then a single point will be used at the centroid of the cavity. All edges along the cavity will form triangles connecting to the centroid. This approach is general and is independent of the number of incident convex ridges and independent of the number of normals incident to the corner node. Figure 10 shows the pure convex corner incident upon four convex ridges. Each ridge has a different number of normals which yields a different number of blend hexahedra.

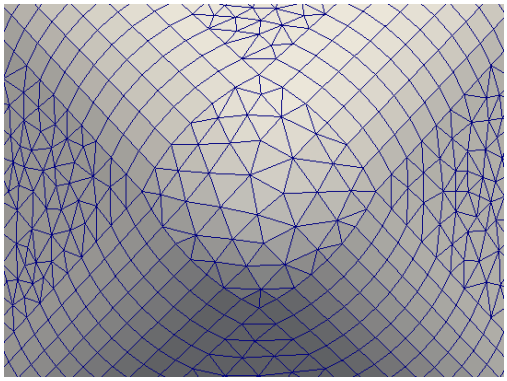
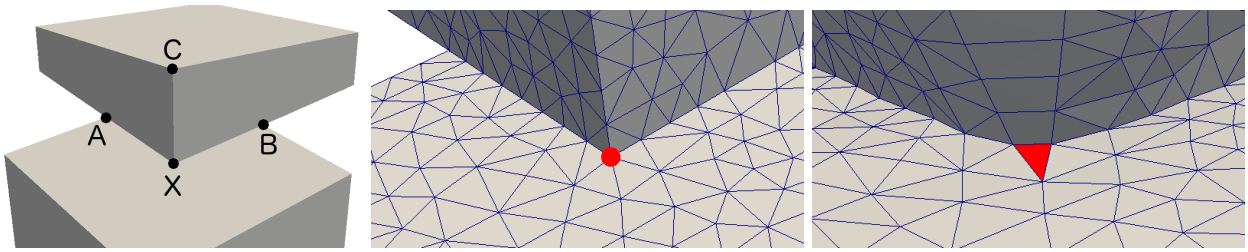


Figure 10. A pure convex corner incident upon four convex ridges.

2. Mostly Concave Corner Node

A mostly concave corner node is a corner node that is incident upon exactly one convex ridge while the rest are concave. Since there is only one convex ridge incident upon this corner node, the node needs to be treated as a convex cusp node and will have two normals. Figure 11(a) shows the model of a two box model with the corner node and ridges identified. Figure 11(b) shows the surface mesh zoomed in around the corner node. For the two normals at this corner node, the contributing surface triangles need to be divided into two groups, as described by line 11 of the pseudocode in Figure 4. Clearly the initial surface triangle for each group will be either of the two surface triangles incident upon the convex ridge. If the incident surface triangles to the corner node are wound based on the corner node's adjacent edges, then the two groups need to be contiguous to avoid intersecting or twisted elements. The first dividing edge of the cycle



(a) Two box model showing corner point X, convex ridge XC, and concave ridges XA and XB.

(b) Surface mesh around a two box model. The corner point is marked.

(c) The resulting stack of blend elements. The top face of the blend elements is shown.

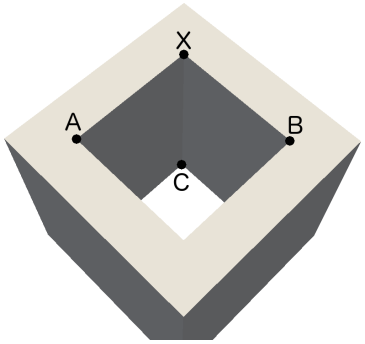
Figure 11. A corner case where only one of the incident ridges are convex. There are two concave ridges shown at the corner point. A convex cusp procedure is used to create a stack of blend triangular prisms.

is the convex ridge, so the other dividing edge can be any other adjacent edge to the corner node, except a concave ridge. It is advantageous to set the dividing edge as the edge that has the largest convex angle (but not large enough to be marked as a convex ridge) between the edge's incident surface triangles because it helps to prevent flat, poor-quality elements for the first layer pyramids and tetrahedra. Figure 11(c) shows the boundary layer around this corner configuration. The stack of blend triangular prisms is beneath the marked triangle. The region surrounding the concavity has been smoothed with Laplacian smoothing.

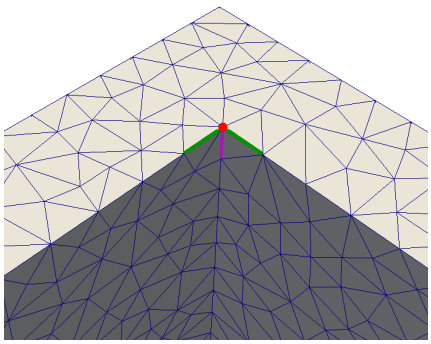
3. Dual Convex Corner Node

The next case is for corner points that are incident upon exactly two convex ridges. There is at least one concave ridge incident upon the corner point, but there may be more. Figure 12(a) shows the model for a box with a hole from the top face through to the bottom face. At the interior corners of this hollowed out region of the box, there are two incident convex ridges and one incident concave ridge. Figure 12(b) shows a zoomed in view of the surface mesh for one of the corner points. There are multiple normals along the convex ridges, but a single normal along the concave ridge. The corner point has two normals: one for the surface triangles inside the concavity, and the other normal for the surface triangles on the top face where the hole is. The average of the unique normals for each set of surface triangles is used for the direction of the two normals. Since all of the concave ridges at the corner node have exactly one normal, the corner node is treated as a convex-convex-ridge node. The two incident convex ridges are declared as neighbors and share the two normals at the corner point. Nothing needs to be done with the concave ridge. Laplacian smoothing is used along the neighboring blend regions near the corner node to smooth the directions of the blend region normals so the neighboring elements have similar volumes and so that no non-planar faces are created. The normals that are emitted from other surface triangles are not smoothed, only the normals incident upon blend regions.

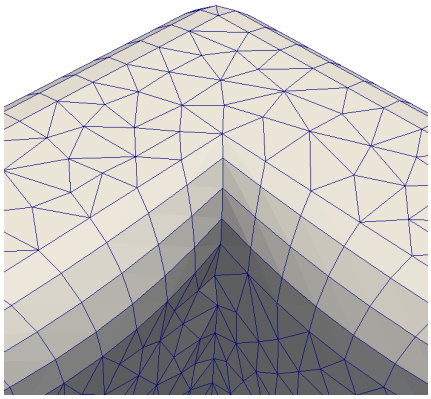
A special case exists for dual convex corner nodes that are incident upon two convex ridges and two concave ridges. If the incident surface triangles are ordered in a cycle based upon neighboring edges, there is a pattern with traversing across these edges between the triangles: concave ridge, convex ridge, concave ridge, convex ridge, and repeat. This is known as a non-Lipschitz node (or saddle point) and the blend volume elements formed in this region will be degenerate due to the variation in direction between normals of the incident ridges. For this reason, a single normal is not sufficient in this region. Figure 13(a) shows the non-Lipschitz node for a model where two cubes are joined. Laplacian smoothing is used along the neighboring blend regions which contain a non-Lipschitz node in order to smooth the directions of the blend region normals so there is a smooth transition between the alignment, orthogonality, and volume of the neighboring elements near the non-Lipschitz node. The normals that are emitted from other surface triangles are not smoothed, only the normals incident upon blend regions. Figure 13(b) shows the blend elements created



(a) Model of a hollowed out box showing corner point X, concave ridge XC, and convex ridges XA and XB.

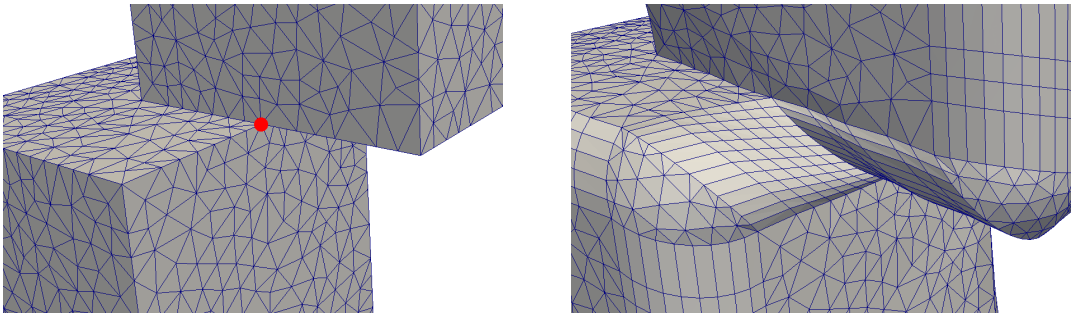


(b) Zoomed in view of the surface mesh of a hollowed out box. The corner point is shown with two green convex ridges and one purple concave ridge.



(c) The clipped points and triangles that lie inside the cavity. The normals inside the concavity has also been smoothed due to intersections.

Figure 12. A corner case where two of the incident ridges are convex and the remainder are concave. The corner point is treated as a convex-convex-ridge point.



(a) Surface mesh of a two joined cubes with the non-Lipschitz node.

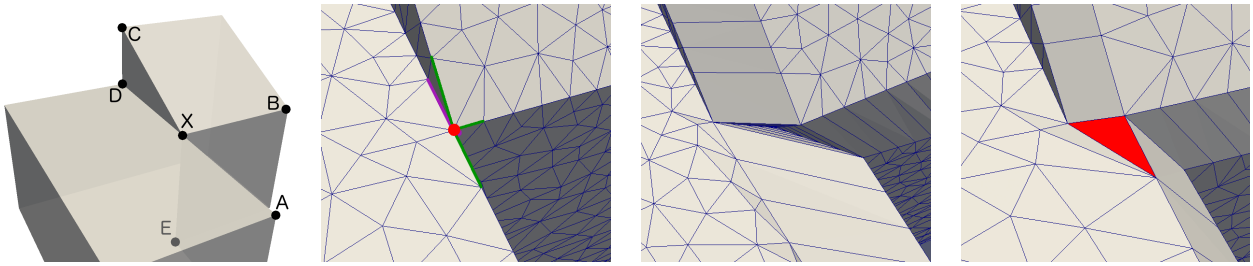
(b) The resulting blend elements after Laplacian smoothing was used along the blend normals.

Figure 13. Treatment of a non-Lipschitz node by Laplacian smoothing.

after the smoothing process. A check is performed for the two blend ridges incident upon the non-Lipschitz node to determine if the first layer triangular prism is ill-formed. If it is, then the normal directions are too twisted to form a valid triangular prism along the ridge, so the local height of the boundary layer is reduced for these ridges until the first layer triangular prism is valid.

4. Complex Corner Node

The final case is the most complex and requires some extra checks. Nodes that are incident upon three or more convex ridges and at least one concave ridge are classified as complex corner nodes. Figure 14(a) shows a model around a complex corner with three convex ridges and two concave ridges incident upon the corner node. Concave ridge XE is shown with the transparent model. Figure 14(b) shows a zoomed in view of the surface mesh around the complex corner with the ridges identified. Concave ridge XE is not able to be seen, but it is between the two lower convex ridges, XA and XB. The difficulty with the complex corner is that the blend cavity is almost shaped liked the pure convex corner cavity. However, the concave ridges cause the original cavity to be ill-formed as in Figure 14(c). If the complex corner was treated as a pure convex corner, then the blend triangular prisms formed in the blend cavity would be nearly flat and have a zero volume. Complex corners are treated by changing the topology of the cavity to fix the degenerate cavity. The topology of the cavity is changed by gluing the exposed interior faces of pairs of blend prisms together. When two convex ridges are joined, the cavity becomes the same topologically as the convex cusp cavity. This means that the complex corner node needs to be classified as multiples of convex-convex ridge nodes and convex cusp nodes. Figure 14(d) shows the resulting boundary layer after two of the convex ridges, XA and XB, are joined together. The newly formed convex cusp cavity is filled using the same approach as convex cusp nodes. The stack of triangular prisms formed from the convex cusp is below the marked triangle.



(a) A box-wedge model showing complex corner X; convex ridges XA, XB, and XC; and concave ridges XD and XE.

(b) Surface mesh showing a complex corner with concave (purple) and convex (green) ridges shown.

(c) The original degenerate cavity that would yield flat prisms.

(d) The resulting blend elements after two convex ridges were joined and a convex cusp is added.

Figure 14. Treatment of a complex corner by classifying the node as a convex-convex ridge point and also a convex cusp point. Ridges along XA are joined with ridges along XB to open up the degenerate cavity.

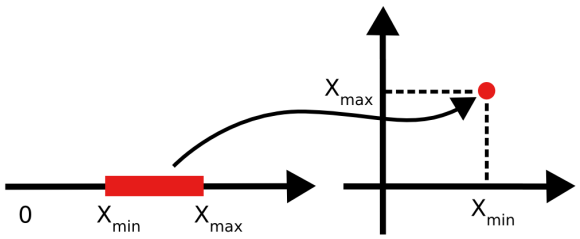


Figure 15. 1-Dimensional analog for the alternating digital tree, showing the mapping of the 1-dimensional extent box to 2-dimensional space

E. Intersections

Once the nodal normals have been determined, a set of intersection tests is used to determine if any normals cause an intersection with any of the boundary layer prisms. It is too computationally expensive to perform a brute-force, exhaustive search by comparing every normal segment to every prism to check for intersections, approximately an $\mathcal{O}(n^2)$ operation where n is the number of prisms. An alternating digital tree (ADT)[21] is used to perform the intersection checks. The ADT works by mapping a d-dimensional space to a $(d+d)$-dimensional half-space. The 1-dimensional analog is shown in Fig. 15. In three-dimensions, the ADT is used with a searching algorithm that determines if a six-dimensional point lies within a particular six-dimensional space subregion. The ADT is populated with the boundary layer prisms. The three-dimensional extent box is computed for each prism and inserted into the ADT as a six-dimensional point. Now each normal is converted to a line segment where the base is on the surface and the endpoint is on the exterior surface of the boundary layer. By projecting the line segments' extent boxes to six-dimensional points, a particular line segment's extent box (as a six-dimensional point) can be tested to see if it intersects with any of the n prisms' extent boxes (also as six-dimensional points) in $\mathcal{O}(\log n)$ time where n is the number of prisms. The identified extent box intersections do not guarantee that a normal segment will intersect another element's boundary layer, but it significantly and efficiently reduces the search space. Intersection checks using computational geometry are then performed for each identified normal and prism pair. If there are a set of intersections in a contiguous region, then Laplacian smoothing is used in these regions to smooth the normal directions to resolve the intersections. Additional non-intersecting normals in the neighborhood of the contiguous region are added to the set of normals to smooth in order to provide a steady and gradual deviation for the normal directions as they move away from the concavity. Figure 7(b) shows the resulting mesh after the contiguous intersections for a concave corner region have been resolved using Laplacian smoothing. The Laplacian smoothing operation is desirable for these important feature regions because it does not affect the overall element density of the region and preserves the semi-structured pattern instead of clipping the rays and introducing unstructured elements close to the model. The smoothing operation also helps to minimize the discretization error of the flow solution by preserving the orthogonality, alignment, and stretching of the prisms. Laplacian smoothing of the normal direction vectors will not work for non-contiguous intersections. Figure 16 shows the surface mesh for two spheres. The two fronts between the spheres will grow towards each other and may cause an intersection. For the identified intersections, the local height of the boundary layer in the region is reduced. This preserves the number of elements in this region of the boundary layer. The points along a normal vector where the local height of the boundary layer was reduced are then packed closer together, providing more resolution in these critical regions where intersections occur. For the test line segments, the non-surface point is set to be a little further than the point on the outer border of the boundary layer, to detect and account for the situation where two walls

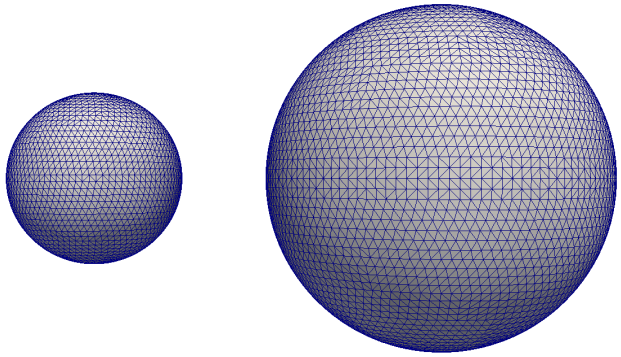


Figure 16. Surface mesh of two different-sized spheres.

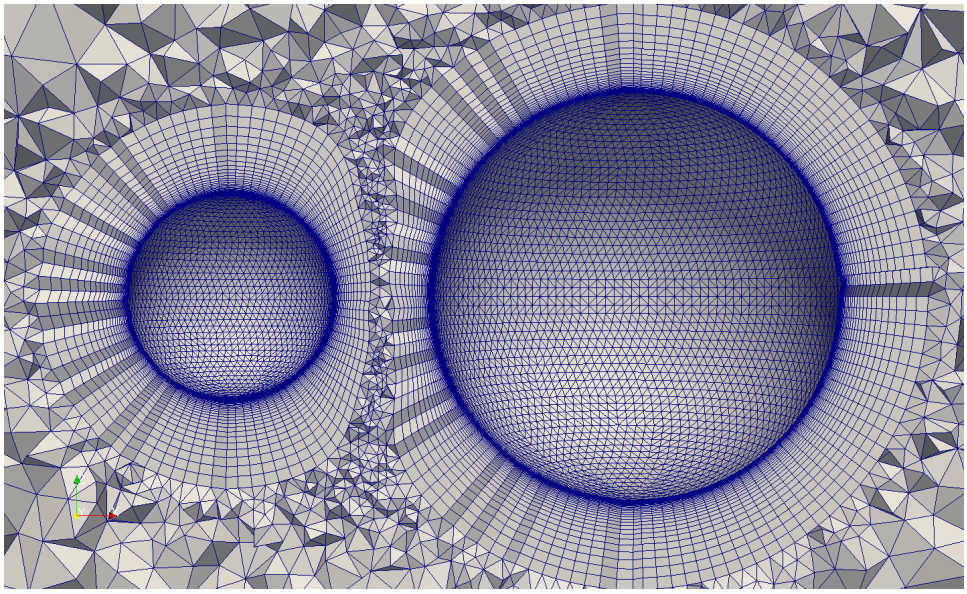


Figure 17. The resolved non-contiguous intersections after reducing the local height of the boundary layer for two fronts that are extruding towards each other.

may be extruding towards each other and come very close to meeting but do not cross over. Terminating too close to another front would cause sliver triangles to be created for the inviscid region inside the small gap. Figure 17 shows the resulting boundary layer after the local height of the boundary layer has been reduced to prevent the two fronts from overlapping. The isotropic inviscid region is also shown between the two boundary layer fronts where the local height of the boundary layer was reduced a little extra to ensure that we do not have a narrow gap between the two fronts, which would cause the isotropic volume mesher to create sliver tetrahedra in this region.

F. Isotropic Inviscid Region

Once a valid boundary layer volume mesh has been created near the model, the remainder of the volume away from the model is tetrahedralized using an off-the-shelf three-dimensional Delaunay mesh generator, TetGen.[8] Since TetGen uses only tetrahedra to discretize the domain, we need to ensure that there are only triangular faces exposed in the boundary layer. If quadrilateral faces are exposed, then TetGen will need to split these quadrilateral faces into triangular faces. We do not want TetGen to split any face, quadrilateral or triangular, on the exterior of the boundary layer, because this will create a non-conformal mesh, so we use a provided flag that disallows TetGen from inserting vertices on input facets. This forces us to do two things. The first is that all exterior quadrilateral faces and incident volume element need to be split before we pass the exterior surface to TetGen. The second is that we must generate the farfield domain and discretize the farfield faces.

Since we are restricting the boundary layer to have the same number of anisotropic layers extruded from each node, the only time there will be exposed quadrilateral faces is when there are convex-convex ridge points on the surface. This means that a hexahedra will be exposed in a blend region on the outer layer of the boundary layer. There are two local operations to treat these exposed hexahedra. The first option is to form a pyramid from the exposed quadrilateral face which would shield the quadrilateral face with four triangular faces. However, it is more computationally expensive to extrude a pyramid from the quadrilateral face, because we need to perform intersection checks to ensure the newly formed triangular faces do not intersect with any other volume elements. Additionally, flat pyramids may need to be created in regions where there are concavities or narrow gaps to avoid causing an intersection. The other approach is to insert a steiner point at the centroid of the hexahedra and connect the six quadrilateral faces to the steiner point to form six pyramids. Finally, the pyramid which has the quadrilateral base on the exterior of the boundary layer is split into two tetrahedra, which splits the quadrilateral face into two triangular faces. Since the

outer layer hexahedra are roughly isotropic, a steiner point can be added without introducing flat volume elements. This approach is advantageous because we do not need to perform any intersection checks since we are only decomposing the hexahedra, instead of extruding a new pyramid from the hexahedra. After the boundary layer is enclosed with only triangular faces, the desired isotropic edge length metric needs to be set for all vertices on the exterior of the boundary layer. The isotropic edge length metric is computed from each vertex's incident edges. This ensures a smooth gradation in element volume between the outer layer boundary layer elements and the neighboring inviscid region tetrahedra.

The farfield domain is generated using two user-specified parameters: the distance to the farfield from the model in chord lengths and the desired edge length metric at the farfield vertices. The farfield is generated using a right-triangular uniform tiling procedure where all triangular faces have the same area and the isotropic metric for all of the farfield vertices is set to the user-defined edge length. All of the triangular faces, vertices, and isotropic edge length metrics of the farfield and the exterior surface of the boundary layer are passed off to TetGen. There is a smooth gradation of the element volumes going from the exterior surface of the boundary layer to the farfield because TetGen uses linear interpolation to determine the isotropic edge length metric for all newly created vertices. After the remainder of the volume has been tetrahedralized, TetGen returns the mesh of vertices and tetrahedra. These vertices and tetrahedra will need to be reindexed so that the unique ids are consecutive once they are added to the boundary layer elements and vertices.

III. Convergence Study

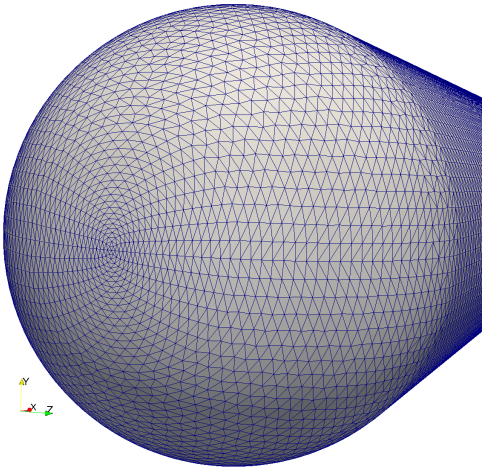
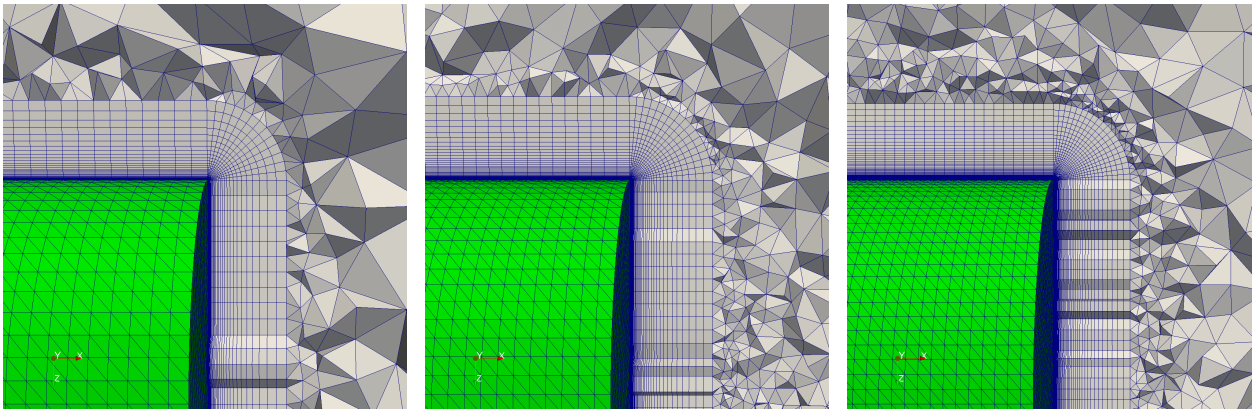


Figure 18. Hemisphere cylinder surface mesh at the hemisphere.

The meshes generated with this approach have been validated against reference solutions from the NASA Langley Research Center's Turbulence Modeling Resource Website. The hemisphere cylinder described by Nishikawa[22] is used for validation. The hemisphere cylinder parameters are for a reference diameter of 1 and a length of 10, where the chord length is based on cylinder diameter. A custom surface mesh generator was used which packed more points towards the leading edge of the hemisphere, Fig. 18. The simulation was run at mach speed 0.6, Reynolds number of 0.35 million, and angle of attack at 10 degrees. The farfield is located 50 chord lengths away from the hemisphere cylinder. The freestream flow conditions specify the reference temperature to be 540 Rankine. The governing equations are the Navier-Stokes equations using the Spalart-Allmaras turbulence model with negative turbulence variable provisions. The boundary conditions for the hemisphere cylinder surface are viscous and the no-slip condition is explicitly set. The bounding box of the domain has its boundary condition set to be the farfield by using Riemann invariants. The reference value for the coefficient of lift given by Diskin[23] for the largest mesh with over 70 million vertices for a prism-dominant mesh is 0.034116.

The algorithm presented in this paper is used to generate a family of meshes around hemisphere cylinders. A scaling factor of $\sqrt[3]{2}^{-1}$ is used to vary the density of the boundary discretization, the thickness of the first layer, the growth rate of the boundary layer's geometric growth function, and the number of layers in the



(a) Coarse mesh (b) Medium mesh (c) Fine mesh

Figure 19. Family of hemisphere cylinder meshes for a slice at the trailing edge.

Surface Vertices	Total Vertices	Total Volume Elements	CL
15236	789889	1781579	0.035650
23300	1556819	3621231	0.034358
37665	3150819	7472505	0.034217

Table 2. Geometry and simulation data for the convergence study

boundary layer. The first model and mesh are generated using a scaling factor of one. Each subsequent model and mesh use a scaling factor of $\sqrt[3]{2}^{-i}$ for the i^{th} model (starting at zero). Using the scaling factor, the total height of the boundary layer remains invariant between the different meshes, but the number of vertices in the boundary layer doubles from each previous mesh. Fig. 19 show the family of three meshes used for this convergence study and Table 2 shows the mesh statistics and the results of each simulation. All simulations were fully converged to machine zero. For the largest mesh, the coefficient of lift has approximately a 0.3% error with the reference value, 0.034116.

Table 3 shows the breakdown of volume elements. The majority of the elements in the boundary layer are prisms. If the input surface mesh contains mostly quadrilateral facets, then the majority of the boundary layer mesh will be comprised of hexahedra. Pyramids and tetrahedra are used to split the outer layer of hexahedra into triangular faces so that TetGen can be used to generate the isotropic inviscid region. The largest mesh generated took less than a minute for over three million vertices.

Meshing Time	Triangular Prisms	Hexahedra	Pyramids	Boundary Layer Tetrahedra	Inviscid Region Tetrahedra
15.42 s	1371892	35776	4160	1664	368087
22.86 s	2657298	72930	6630	2652	881721
41.77 s	5350146	138000	10000	4000	1970359

Table 3. Breakdown of volume elements for the convergence study

IV. Future Work

This work is the first step towards extending our parallel two-dimensional approach[24] to handle three-dimensional domains. For the two-dimensional work, the extrusion-based approach with multiple normals was also used, and a strong scaling speedup of 180 over the fastest sequential algorithm was observed when using 256 processes, roughly a 70% efficiency. This approach decomposed the boundary layer into multiple subdomains that can be triangulated independently. The inviscid region was also decoupled in such a way that two processes can concurrently refine neighboring subdomains without having to communicate. Intersections are detected efficiently using the aforementioned ADT. The two-dimensional work was improved upon and the manuscript[25] is currently under review for the Association for Computing Machinery's (ACM) Transactions on Mathematical Software (TOMS) journal. For the ACM TOMS manuscript, we improved the performance through benchmarking and optimization. A 79% parallel weak scaling efficiency on 1024 distributed memory nodes, and a strong scaling speedup of 368 over the fastest sequential algorithm using 512 processes, a 72% parallel efficiency, is shown through numerical experiments.

Currently, the code is sequential, but is being designed with decisions that exploit data parallelism and with a level of abstraction that allows the code to be flexible and easy to extend to be parallel. This requires diligent book keeping. Each model node has an associated set of growth curve ids. Each growth curve keeps track of it's first interior vertex id and the number of points that it will insert along this growth curve. Growth curves have sequential ids. So growth curve with id 0 has it's first interior vertex's id as the number of input surface nodes, the nodes on the surface of the model. Growth curve id 1 has a first interior vertex id as $\#surface\_nodes + GrowthCurve_0(\#interior\_nodes)$. An all-gather operation can be performed where each process computes the sum of all of it's growth curves' layer counts. Then each process can determine its first local insert vertex id from all of the other process' with a lower rank. This allows each process to determine the vertex ids for all of the triangular prisms in its subset of the domain without further communication.

V. Conclusion

An approach to generating an anisotropic semi-structured boundary layer around complex configurations has been demonstrated. The approach uses an extrusion-based method with multiple nodal normals where there are sharp convexities on the model's surface. The input for our application is a triangular surface mesh, but a surface mesh with mixed triangles and quadrilaterals can be handled the similarly. A triangular prism and hexahedron extruded from a neighboring triangle and quadrilateral face, respectively, will be compatible because they share the same interior quadrilateral face incident upon the common edge of the surface triangle and surface quadrilateral. Triangular prisms are extruded from surface triangles, and prismatic elements are extruded from mesh edges and mesh nodes in between the sharp convex regions where there are multiple normals. An exhaustive classification of node types is presented. A node's type is dependent on the number of incident ridges, edges where there is a sharp change in the slope across the model's surface. A node may be a regular node with no incident ridges, a cusp node with one incident ridge, a ridge node with two incidentincident ridges, and a corner node with three or more incident ridges. These classifications are broken down further depending on the convexity and concavity of the incident ridges. Non-Lipschitz nodes, a special case for corner nodes, are also treated. Intersections in the boundary layer are detected efficiently using a spatial tree. Local, contiguous intersections are treated using Laplacian smoothing on the normal vectors while non-local intersections are treated by reducing the local height of the boundary layer in the overlapping region. Before the inviscid region can be generated with the isotropic tetrahedral mesh generator, all exposed quadrilateral faces are split by locally decomposing only outer layer hexahedra without the decomposition propagating to neighboring elements. The farfield bounding box is generated using uniform right triangles and is parameterized in terms of chord lengths between model and farfield and the desired minimum edge length for the farfield bounding box.

The input geometry, boundary layer growth function parameters, and farfield bounding box parameters are the only inputs to this automated process. The application is written in C++14 and is only dependent on TetGen for the inviscid region, making the application as a whole, a lightweight and portable mesh generator for aerospace applications with viscous flows. Using a high-fidelity mesh to begin the iterative CFD pipeline will yield a final, acceptable mesh in fewer iterations than an ill-suited initial mesh. Constructing this initial mesh efficiently without human intervention eliminates an expensive bottleneck in the development process.

Acknowledgments

This work was supported (in part) by the National Institute of Aerospace (http://www.nianet.org) and NSF grant CCF-1439079. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIA or the NSF. We thank Mike Park and Boris Diskin for their discussions and comments.

References

[1]Tam, A., Ait-Ali-Yahia, D., Robichaud, M., Moore, M., Kozel, V., and Habashi, W., "Anisotropic mesh adaptation for 3D flows on structured and unstructured grids," *Computer Methods in Applied Mechanics and Engineering*, Vol. 189, No. 4, 2000, pp. 1205–1230.

[2]Formaggia, L., Micheletti, S., and Perotto, S., "Anisotropic mesh adaptation in computational fluid dynamics: application to the advection–diffusion–reaction and the Stokes problems," *Applied Numerical Mathematics*, Vol. 51, No. 4, 2004, pp. 511–533.

[3]Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA CR-2014-218178, Langley Research Center, March 2014.

[4]Chawner, J. R., Dannenhoffer, III, J. F., and Taylor, N. J., "Geometry, Mesh Generation, and the CFD 2030 Vision," Tech. rep., 2015.

[5]Chawner, J. R., Dannenhoffer, III, J. F., Dey, S., Thornburg, H., Jones, W. T., Slotnick, J., and Taylor, N. J., "The Path to and State of Geometry and Meshing in 2030: Panel Summary," Tech. rep., 2015.

[6]Karman, S. L., Wyman, N., and Steinbrenner, J. P., "Mesh Generation Challenges: A Commercial Software Perspective," Tech. rep., 2017.

[7]Schöberl, J., "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules," *Computing and visualization in science*, Vol. 1, No. 1, 1997, pp. 41–52.

[8]Si, H., "TetGen, a Delaunay-based quality tetrahedral mesh generator," *Association for Computing Machinery's Transactions on Mathematical Software*, Vol. 41, No. 2, 2015, pp. 11.

[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]Hassan, O., Morgan, K., Probert, E., and Peraire, J., "Unstructured tetrahedral mesh generation for three-dimensional viscous flows," *International Journal for Numerical Methods in Engineering*, Vol. 39, No. 4, 1996, pp. 549–567.

[11]Martineau, D., Stokes, S., Munday, S., Jackson, A., Gribben, B., and Verhoeven, N., "Anisotropic hybrid mesh generation for industrial RANS applications," *AIAA Paper 2006-534*, 2006.

[12]Marcum, D. L., Alauzet, F., and Loseille, A., "On a robust boundary layer mesh generation process," *AIAA Paper 2017-0585*, 2017.

[13]Aubry, R., Mestreau, E., Dey, S., Karamete, B., and Gayman, D., "On the 'most normal' normal-Part 2," *Finite Elements in Analysis and Design*, Vol. 97, 2015, pp. 54–63.

[14]Pirzadeh, S. Z., "Three-Dimensional Unstructured Viscous Grids by the Advancing-Layers Method," *AIAA Journal*, Vol. 34, No. 1, Jan. 1996, pp. 43–49.

[15]Garimella, R. V., *Anisotropic Tetrahedral Mesh Generation*, Ph.D. thesis, Rensselaer Polytecnic Institute, 1998.

[16]Jansen, K. E., Shephard, M. S., and Beall, M. W., "On Anisotropic Mesh Generation and Quality Control in Complex Flow Problems." *International Meshing Roundtable*, 2001, pp. 341–349.

[17]Ito, Y., Shih, A. M., Soni, B. K., and Nakahashi, K., "An approach to generate high quality unstructured hybrid meshes," *AIAA Paper 2006-530*, 2006.

[18]Löhner, R., "Generation of viscous grids with ridges and corners," *AIAA Paper 2007-3832*, 2007.

[19]Aubry, R., Dey, S., Mestreau, E., and Karamete, B., "Boundary layer mesh generation on arbitrary geometries," *International Journal for Numerical Methods in Engineering*, Vol. 00, 2017, pp. 1–20.

[20]Park, M. A. and Anderson, W. K., "Spatial Convergence of Three Dimensional Turbulent Flows," Tech. rep., 2016.

[21]Bonet, J. and Peraire, J., "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," *International Journal for Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1–17.

[22]Nishikawa, H. and Diskin, B., "Customized Grid Generation Codes for Benchmark Three-Dimensional Flows," AIAA Paper 2018–1101, 2018.

[23]Diskin, B., Anderson, W. K., Pandya, M. J., Rumsey, C. L., Thomas, J. L., Liu, Y., and Nishikawa, H., "Grid Convergence for Three Dimensional Benchmark Turbulent Flows," AIAA Paper 2018–1102, 2018.

[24]Pardue, J. and Chernikov, A., "Parallel Two-Dimensional Unstructured Anisotropic Delaunay Mesh Generation of Complex Domains for Aerospace Applications," *45th International Conference on Parallel Processing*, 2016, pp. 608–617.

[25]Pardue, J. and Chernikov, A., "Algorithm xxx: An Efficient Parallel Anisotropic Unstructured Delaunay Mesh Generator for Two-Dimensional Aerospace Computational Fluid Dynamics Analysis," *Association for Computing Machinery's Transactions on Mathematical Software*, (submitted for publication).