

Anisotropic Tetrahedral Meshing Based on Surface Deformation Techniques

John P. Steinbrenner*
Pointwise Inc., Fort Worth, Texas, 76104

and

J.P. Abelanet †
Basis Software, Keller, Texas, 76248

A technique for the generation of anisotropic tet meshes is introduced that advances vertices belonging to watertight collections of triangles. Non-intersecting triangular fronts are maintained via a surface deformation algorithm that efficiently detects when deformed vertices are either in close proximity of or intersect other triangles and vertices. Vertices on the triangular front are advanced and decimated in directions pointing away from the front, with tetrahedra inserted in the voids left by the deformed front. User-defined cell quality measures guarantee minimal tetrahedra properties in the anisotropic regions. The final fronts serve as the isotropic surface boundaries passed to the isotropic tet mesher. The resulting anisotropic and isotropic meshes are then merged seamlessly. The method can be used to transition from anisotropic to isotropic surface triangles, to match cell size to adjacent meshes, and to create boundary-layer type meshes. Examples are provided for each.

I. Introduction

A cursory literature search reveals that several robust general-purpose tet meshers are available for use in computational analysis [1,2,3,4]. Several of these meshers are isotropic in nature [1,2], in that they attempt to construct tet elements with nearly equilateral angles and approximately equal length edges. The successes of these meshers indicate that isotropic meshes are ideally suited for a large class of problems. In computational fluid dynamics, however, isotropic elements are neither required nor desired in certain regions of the problem domain. For example, in boundary layer regions of the flow field, the gradients of the flow properties are decidedly directional; derivatives normal to the body are orders of magnitude greater than in the streamwise flow direction. To resolve such small-scale phenomena, the mesh cell height at the body must typically be 6-8 orders of magnitude smaller than the nominal size of the geometry. This implies that an isotropic element of this small size could have a volume less than $1/10^{24}$ that of the overall problem domain. Obviously this results in a cell requirement that is generations beyond current state-of-the-art software and hardware, and so the CFD practitioner typically seeks anisotropic tets that are significantly shorter in the geometry normal direction than in the transverse direction.

As a second example, when the flow field is broken into multiple blocks to ease the mesher and/or flow solver burden, it is necessary for cell sizes to be nearly continuous across the boundaries of the blocks. This reduces the likelihood of artificial artifacts to be introduced into the flow solution. If the adjacent block has directional spacing that yields non-uniform cells at the adjacencies (Figure 1), an isotropic mesher will be unable to maintain cell size continuity across the seam.

* Vice President, Research and Development, Pointwise Inc., 213 S. Jennings Ave, Fort Worth, TX 76104, Senior Member.

† President, Basis Software, 1129 Countryhill Dr., Keller, TX 76248.

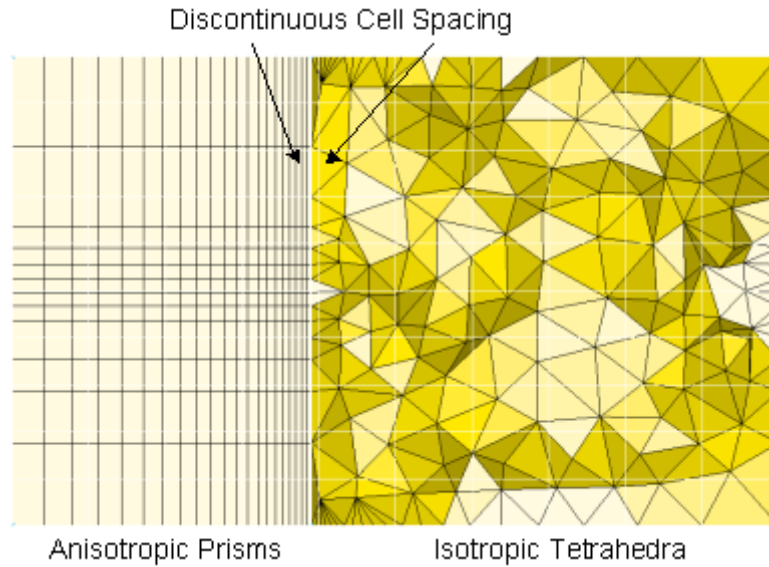


Figure 1. Discontinuous Cells Across Block Boundaries

A third example illustrating the inappropriateness of isotropic cells in CFD problems occurs where the surface elements themselves are anisotropic in shape. Surface anisotropy might be employed in order to resolve a geometrical feature adequately while keeping the local triangle count to a reasonable value. Again, the isotropic mesher has little ability to produce suitable cell quality in these regions (Figure 2).

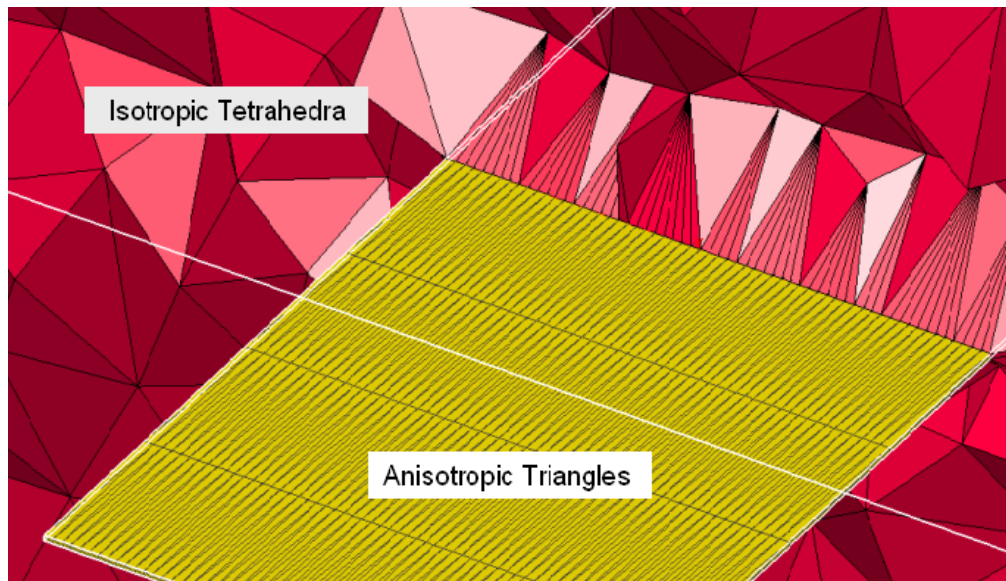


Figure 2. Isotropic Tets Formed on Anisotropic Triangles

In all three of these examples, anisotropic tet cells are needed in regions near the surface meshes, whereas isotropic cells can be used on the flow field interior. Unfortunately, purely isotropic meshers have limited ability to produce acceptable quality elements near the body. This is not an indictment of such methods, but a statement of fact about their applicability to many CFD problems. By their very formulation they attempt to produce meshes of equal size.

This paper addresses the need for anisotropic elements in a tet mesh by separating the anisotropic regions from the isotropic regions automatically, generating meshes on each region, and ultimately merging the two meshes into a single continuous tet representation. The primary enabling technology in this effort is an efficient surface deformation technique (Section II) that maintains a valid topological connection of triangles while insuring that proposed vertex deformations and decimations neither intersect nor come in close proximity to other triangles. The anisotropic algorithm (Section III) deforms and decimates vertices one layer at a time in a manner that maintains strict control of the quality of the tetrahedra formed by tessellating the region between the original and deformed layers. New layers are advanced repeatedly until there are no longer any valid vertices to be deformed, at which time the final isotropic fronts are passed into the isotropic tet mesher and the resulting anisotropic and isotropic tets are combined into a single mesh.

As will be shown in Section III, cells that fail a user-set quality measure are either optimized via a steepest descent algorithm or are discarded. Sections IV and V detail the specifics of the deformation and decimation operators, respectively. Section V also illustrates how vertex decimation can be used to transition smoothly from anisotropic triangles on the surface mesh to isotropic interior tets. Section VI covers an anisotropic triangle generation scheme developed to complement the tet generator. Following this (Section VII) is an explanation of prism recombination and tet block splitting algorithms incorporated to increase flexibility with analysis software. Section VIII describes the lean set of user controls required to operate the algorithm, and also presents several examples of anisotropic/isotropic meshes on complex geometries. Finally, Section IX draws conclusions and describes future work planned with this new capability.

II. Non-Intersecting Surface Topology Maintenance

The surface deformation algorithm is implemented within the MeshDeformer class in the general-purpose CAD library described in [5]. It is designed for the efficient global determination and prevention of self-intersection in a triangulated surface mesh during continuous local geometric and topological modification. The determination of self-intersection is optimized through the use of an axis-aligned binary space partitioned (BSP) tree, which is automatically updated to reflect any changes to the geometry or topology of the surface mesh. The input is a general triangulated surface mesh, open or closed, possibly with non-manifold edges. When performing geometric modifications, there is a priority queue of all the vertices in the mesh sorted by a user-defined grading function, which can be used repeatedly to perform deformation passes over the entire mesh. This involves proposing a new coordinate for each vertex, as well as a buffer, which is used to create an envelope for the local deformation. The deformed surface is checked for self-intersection and, if none is detected, the new coordinate is accepted. When performing topological modifications such as local decimation, another diagnostic check is used to test a specific set of faces for self-intersection with the rest of the surface mesh. The typical approach is to use a MeshOperator class method to perform a topological modification, check the locally modified faces, and if self-intersection is found, revert the mesh back to its previous state using the provided inverse topological operator; otherwise, the topological modification is accepted.

III. Anisotropic Meshing Algorithm

Creation of the anisotropic tetrahedral elements begins with the construction of a topological representation of the watertight triangles (described above) that serve as the bounding surface elements of the volume to be meshed. A typical B-Rep solid modeling hierarchy [6] is employed, with vertices representing triangle nodes, edges representing the linear distance shared by adjacent triangles, coedges representing the oriented instance of the edges, and face elements representing the triangles themselves. Ancillary data such as vertex normals are also maintained in the topological model. Parallel to the topology data exists a data structure for each vertex in the model. This structure holds data pertinent to the tet generation algorithm, such as the starting (first layer) and current target cell heights and positions, pointers to the vertex topology, and several integer variables marking status and vertex position in the surface and volume meshes. It is important to note that the topology established in this step is necessarily maintained for the entire anisotropic tet generation procedure. This insures a persistent watertight representation during construction.

The heights of the tetrahedral cells to be generated by deforming the surface elements away from the front are computed from the heights of cells of adjacent blocks. These adjacent blocks can contain hexahedral, prism, or tetrahedral elements. If no adjacent block exists, the height is computed from the equivalent equilateral tetrahedral

height for a given vertex. Alternatively (and in fact most frequently), cell heights may be user-specified independently for each component surface mesh defining the watertight front. Users may also specify that a certain set of triangles not advance at all, or instead be treated as a symmetry boundary condition (see Section V).

After the cell heights are formed, vertices on the connected triangle fronts are processed one layer at a time via deformation and decimation of the vertices. Specifics of these operations are discussed in Sections IV and V. Vertices that are marked with symmetry boundary conditions and are also adjacent to at least one vertex scheduled to advance are processed first. These vertices are decimated (removed) from the front by replacing the fan of triangles surrounding the vertex with a reduced set of triangles. The act of decimating the vertex locally advances the front outward by a small amount, and the void left between the original and modified front is replaced by new tetrahedra.

Next, those vertices marked to advance are deformed in the direction of the vertex normal, with the regions between the original and deformed vertex position filled with a small set of tetrahedra. The order of the vertex processing is controlled by the priority queue described earlier that considers vertices adjacent to smaller triangles first. This deformation operation is typically applied to the majority of the active vertices. The final set of vertices processed contains those that failed the previous deformation sweep due to the proposed vertices' close proximity to other vertices on the front. These vertices are decimated (in much the same way as the symmetry b.c. vertices) in order to reduce the level of triangle anisotropy on the front.

When all vertices have been processed, the layer is complete and the algorithm prepares for another sweep. A vertex is halted from subsequent sweeps if any of several criteria are met. First, a vertex is halted if the current cell height is approximately equal to the average of its neighboring triangle edge lengths. This situation indicates a nearly isotropic tetrahedral element. Next, the vertex stops if it fails the intersection and proximity tests with the rest of the front. A vertex is also halted if a previous deformation results in tetrahedra with negative volumes or quality measures (Section VIII) below the user-set thresholds. Finally, a vertex is halted if any of its neighbors were halted in a previous layer. This prevents a vertex from being advanced a substantial number of layers more than its neighbor.

When all vertices have been halted, the anisotropic algorithm is complete and the resulting triangle front is passed to the isotropic tet mesher. This front typically contains triangles that are more isotropic in shape and more uniformly spaced than the original front. This in turn reduces the burden on the isotropic mesher, thereby increasing the likelihood for successful meshing. As a final step, the tetrahedra returned from the isotropic mesher are merged with the existing anisotropic tets into a single volume representation. Since the layer of tet cells on both sides of the final front are relatively isotropic in shape, the transition between the anisotropic and isotropic regions of the overall mesh are usually invisible.

IV. Surface Deformation

The primary operation in the anisotropic tet generation scheme involves the sequential deformation of vertices and the construction of tetrahedra in the void regions formed by the deformation. Vertices are deformed into new positions defined by a normal direction and a ΔS value. The vertex normal direction is initially defined as the area-weighted average of the face normals of the manifold of adjacent triangles, and is then adjusted as necessary to guarantee that its dot product with each of the neighboring triangle normals is positive (visibility test) [7].

For subsequent deformations of the vertex (beyond the first layer of cells), several approaches to recomputing and/or modifying the normals were considered. The most trivial approach, to keep the normals fixed at their original directions for the duration of the deformation algorithm, results in severe discontinuities of cells on the interior of the mesh, particularly near regions of sharp surface turning angles. A second approach is to recompute the normals based on local triangles and the visibility criteria (see above) after each layer is formed. This idea was discarded because it tends to produce instabilities in the overall front movement, especially in regions where local vertex decimation significantly modifies a vertex's adjacent triangle normals (Section V). A third approach is to smooth and store the vertex normals on the initial front, and to apply an affine combination of the unsmoothed and smoothed normals at each vertex, increasing the weighting towards the smoothed values with each layer [4][8]. While this approach requires very few operations to update the normal between layers, it also requires vertex normals in sharp geometric corners to be fixed so that the front does not produce inverted cells. This was found to produce unwanted angular discontinuities in the interior of the mesh.

The approach incorporated for normal calculation involves applying a global smoothing sweep of the normals after each layer is formed. Two passes are used each layer, with an applied relaxation equal to $\frac{1}{2}$ the ratio between the cumulative distance the vertex has been deformed from its original surface mesh position and the equivalent

isotropic edge length at the vertex, clamped between 0 and 1. The two-pass and reduced relaxation approach is used to maintain stability of the normal. The local relaxation is then further reduced, depending on the maximum angle between the surface vertex normal and its adjacent triangle normals. If the triangles adjacent to the vertex are coplanar, there is no further reduction in the relaxation factor. Conversely, if the maximum angle between the vertex normal and the adjacent faces is greater than 30 degrees, the local relaxation is reduced by a factor of 3. The local reduction varies linearly between these two extremes. The additional reduction at sharp surface angles prevents the geometric discontinuities near bodies from being dissipated over too small a span of cells.

The ΔS value applied to the vertex is defined by $\Delta S = \Delta S_0 * \text{growthRate}^{\text{layer}-1}$, where ΔS_0 is the initial cell height prescription described in Section III, growthRate is the prescribed ratio of cell heights between successive layers, and layer is the integer value corresponding to the number of tetrahedral layers behind the vertex.

The proposed deformed position for the vertex is trivially formed from the existing vertex position, the normal direction, and the ΔS value. This position is accepted for deformation, however, only if deforming the vertex (and its surrounding triangles) by a distance equal to $\Delta S' = \Delta S(1+F_s)$ also fails to produce intersections or adjacencies with any other vertices, edges, or triangles on the front. The algorithms and BSP data structures described in Section II allow these intersection tests to be performed efficiently and accurately. The F_s variable in the formulation of $\Delta S'$ above is a user-set safety factor of order 1. The safety factor insures that triangles on the front will maintain a buffer between encroaching fronts that is on the order of the local cell spacing. This provides an adequate amount of space for the final fronts to serve as the boundaries of a region filled with isotropic tets (Figure 3).

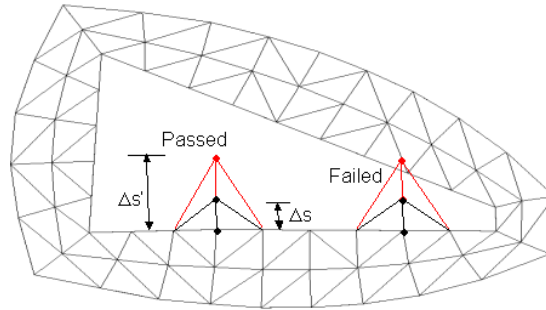


Figure 3. Safety Factor F_s

If the proposed deformed vertex position fails the intersection/proximity tests, the deformation $\Delta S'$ is repeatedly reduced by 50% and re-evaluated until the position passes or until a maximum number of iterations are reached. If a suitable reduced value is obtained, the position is increased halfway to the next level in one final effort to maximize the height of the deformation. Figure 4 illustrates this procedure.

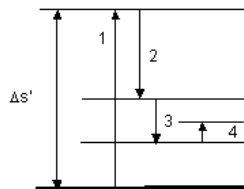


Figure 4. Iterative Deformation Height

When a vertex is deformed from a previous position, it creates a void that may be filled with tetrahedra, provided that the tetrahedra pass user-set geometric quality criteria (Section VIII). As a minimum, each tet in the manifold must possess a positive volume. Optionally, one or more cell skew measures may be queried. If any of these measures fall below the minimum user-set threshold quality requirements, the tets and the deformed vertex are deemed unacceptable. In these cases, a steepest descent algorithm is employed to adjust the vertex position to improve the local mesh quality. This proceeds by normalizing each of the enabled quality measures followed by calculating each measure's gradient at the proposed position. The position is then moved a small distance in the

direction of the gradient, and the resulting cell quality is re-measured. The gradient calculation, vertex movement and cell quality measure sequence is repeated iteratively until the cell quality reaches the user-set threshold, the cell quality stops improving, or until a maximum number of iterations is reached. Optimization is intended only for minor adjustment of the vertex position, and the maximum perturbation is capped at 10% of the originally proposed $\Delta S'$ magnitude. Nevertheless, it has been very successful in repairing negative volume tets as well as cells that do not satisfy minimally acceptable skew criteria.

Tetrahedra are formed and added to the developing mesh at the completion of the deformation layer, i.e., after all vertices have been deformed. A total of N tetrahedra are added for each vertex that satisfies the intersection/proximity and cell quality criteria. N is equal to the number of triangles adjacent to the deformed vertex. Those vertices that were not deformed successfully due to failure to satisfy intersection and quality tests are halted from further deformation.

V. Surface Decimation

Another key operation in the anisotropic tet generation scheme involves the removal, or decimation of vertices from the triangular front. Vertex decimation is required for two specific purposes described later in this section. The basic act of decimation removes a node from a fan of N triangles, replacing it with a fan of $N-2$ triangles that use only the exterior nodes of the original triangle fan. The initial triangulation of the exterior vertices is topologically equivalent to the collapse of the shortest edge in the original fan. This is illustrated in Figure 5a and Figure 5b, which locally reduces the triangulation from 6 to 4 faces. Decimation of the vertex automatically updates the topological representation of the new set of triangles using tools described in Section II.

Since decimation of a vertex changes the surface topology, it is necessary to form tetrahedral elements that link the previous surface to the modified representation. This is accomplished by adding a total of $N-2$ new tetrahedra, each tet defined by the 3 vertices of the new triangulation and the decimated vertex. Before these new tets are added to the developing mesh, they are tested against the same set of quality thresholds employed for surface deformation, namely positive volume and a number of cell skew measures. It is not uncommon for at least one of the new tets to fail the quality criteria, and so systematic edge swapping is applied in an effort to improve the overall quality (Figure 5c). The edges are swapped in multiple passes and the algorithm is repeated each time an overall improvement to the tet collection is computed. Swapping completes when either all tets pass the user-set quality criteria, or it is unable to compute a satisfactory set of tets. In the latter case, the decimated node is reinserted into the surface mesh and the vertex is enabled for deformation on the next layer.

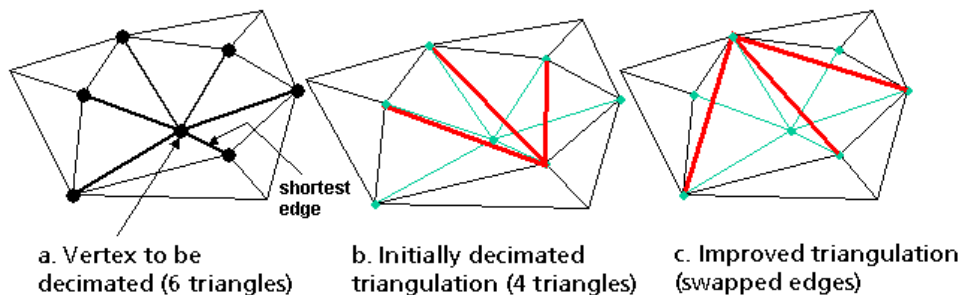


Figure 5. Decimation of a Node

In order for the region between a fan of N triangles surrounding a vertex and the reduced $N-2$ set of decimated triangles to enclose a positive volume (to insure positive volume tets), it is necessary for the fan of triangles to be locally concave at the center vertex. If all vertices are coplanar, for example, positive volume tetrahedra could not be formed. There are two different scenarios in which the decimation algorithm may be applied successfully due to local concavity of the vertex.

The first application of decimation allows prescribed surfaces in the overall mesh to be treated as symmetry boundaries to advancing layers of anisotropic tets. Rather than deform vertices on these prescribed boundaries, those adjacent to recently deformed vertices become candidate vertices for decimation. Decimation is usually successful because of the natural concavity at the decimation point due to its neighbor's deformation. This idea is illustrated in two dimensions in Figure 6. Here, vertices on the horizontal axis are marked for deformation, and vertices on the vertical axis are marked as symmetry boundary condition vertices. After all of horizontal vertices have deformed upward (Figure 6a), the vertex in the corner is decimated by forming a cell from the vertices on its neighboring faces

(Figure 6b). The process is repeated after each layer of deformations takes place (Figure 6c), each time identifying new candidate vertices to be decimated. The overall effect of the procedure is to form an anisotropic mesh from deforming vertices and to have the symmetry boundary condition vertices assimilated into the mesh. An example of using decimation to enforce a prescribed set of surface triangles is shown in Figure 7.

In practice, the global quality of the tetrahedra formed in the vicinity of symmetry vertices is further improved by processing vertices in a topologically alternating order; i.e., the first vertices decimated are those that are not adjacent to any triangles formed from previous decimations. This is achieved by pre-processing the vertices to be decimated with a series of edge swaps. First, all edges containing two vertices scheduled for decimation are identified. Each of these edges is adjacent to 2 triangles, one on the deforming front, and one on the symmetry mesh. Next, the tetrahedron formed by swapping the edge with the edge formed by the 2 opposite vertices of the triangles is computed. If the tet formed passes a positive volume check along with the user-set cell quality criteria, the front is modified locally and the cell is added to the mesh. Prefacing the decimation step with a series of edge swaps has a twofold advantage with respect to cell quality. First, cells formed by the edge swap tend to be of high quality because their maximum bend angle is usually the angle between the symmetry plane and the surface, nominally 90 degrees. Secondly, swapping the edges creates a number of concave pockets of adjacent triangles that are more readily amenable to successful vertex decimation. The edge swap procedure is schematically illustrated in Figure 8.

Considering that vertices on the deformation front are formed independently from the symmetry surface meshes, it is necessary to construct the symmetry meshes a priori in a manner that is consistent with the anticipated deformation attributes (initial spacing and growth rate). Section VI describes a surface anisotropic triangle algorithm developed for this purpose.

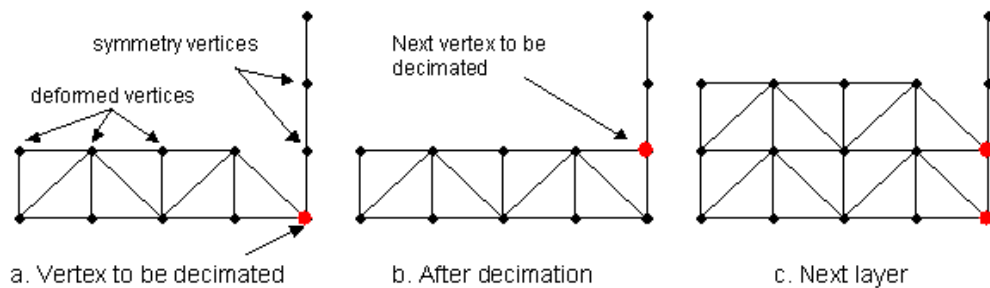


Figure 6. Decimation as a Boundary Condition

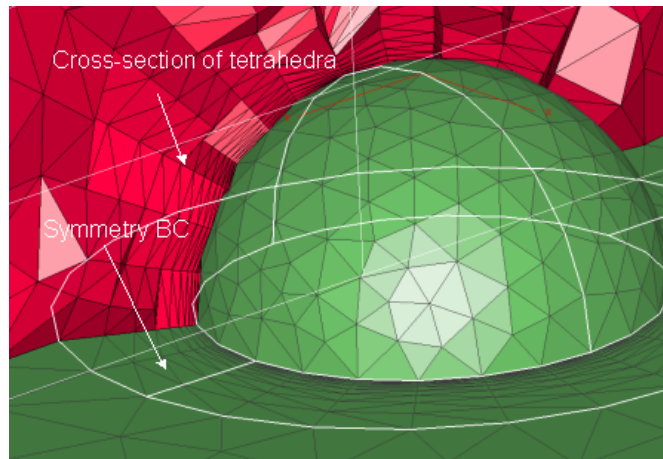


Figure 7. Example Symmetry Boundary Condition

The second application of decimation allows tetrahedra to blend smoothly from anisotropic triangles defined on the initial surfaces of the front to isotropic tets on the interior of the mesh. A smooth transition is maintained through a modified form of the intersection/proximity tests used for vertex deformation. During the deformation phase of the meshing procedure, a vertex is first tested to see if its deformed or extended (via the safety factor F_s) location

intersects the rest of the fronts. If it does not, it is further tested by perturbing its location in the direction of each adjacent edge in the triangle fan (Figure 9). If any of the perturbed vertices intersect the rest of the surface, the vertex is not deformed; instead it is marked for decimation later after all vertices are deformed.

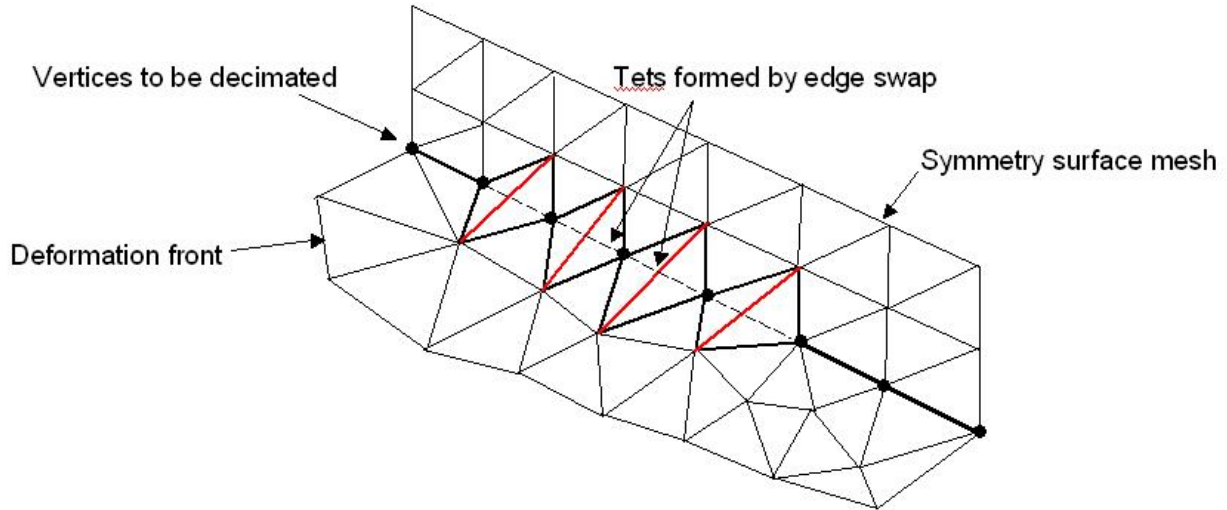


Figure 8. Edge Swap Processing Prior to Decimation

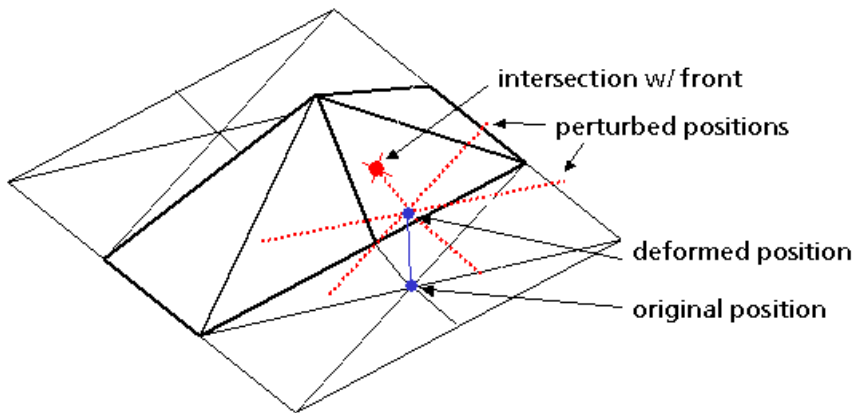


Figure 9. Perturbation of Deformed Vertex

The perturbation distance is set equal to the local prescribed cell height ΔS multiplied by a user-set blend factor F_b , nominally set to 0.5. With this type of formulation the perturbation distance grows as the layers and cell heights grow. In practice this tends to mark every j^{th} ($j=2$ or 3) vertex for decimation in the direction of the anisotropy when the layer corresponding to a cell height equal to the anisotropic edge distance is reached. Vertices marked for decimation (those failing the perturbation test) are generally surrounded by successfully deformed vertices. This provides the necessary concavity at the vertex that allows decimation to produce positive-volume tetrahedra. An example of the smooth transition from anisotropic to isotropic tets that results from this method is displayed in Figure 10. Note that for smaller values of the F_b , more layers would be required to decimate the mesh to the same degree, and vice-versa.

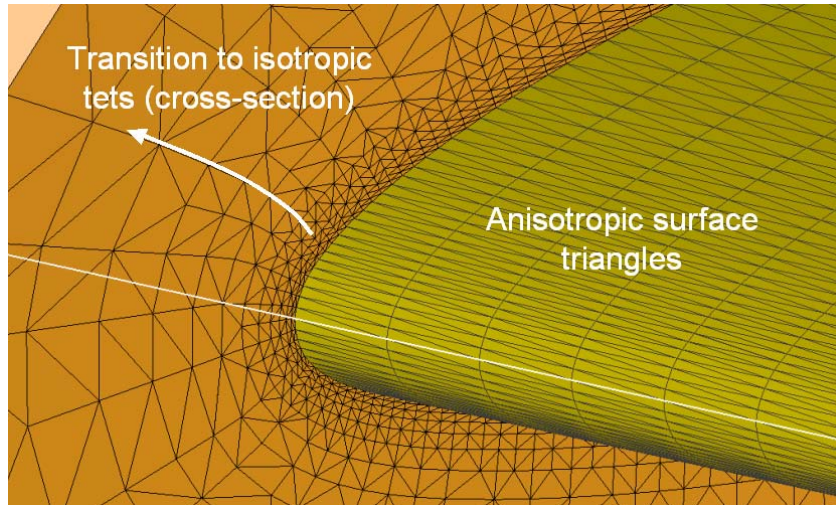


Figure 10. Example Transition to Isotropic Tets

VI. Anisotropic Triangle Generation

In the anisotropic tet meshing algorithm described above, cell layers are constructed in a direction normal to the underlying geometry. If any planes of symmetry exist in the mesh, vertex deformation near the symmetry surfaces is nominally parallel to the symmetry planes. The vertex decimation algorithm allows deforming layers to encapsulate layers of triangle cells on the symmetry meshes. This suggests that for acceptable cell quality, the triangles on the symmetry planes must possess similar anisotropic properties as the tet mesh being generated.

It was therefore necessary to develop an anisotropic triangle mesher complementary to the tet mesher. The basis of this effort was an existing Delaunay isotropic surface mesh generator. While Delaunay methods of this type have been in use for a number of years, their extension to anisotropy has seen only limited success. One of the more popular techniques is to employ a local field distortion to effect anisotropy into the mesh [9][10], but meshes produced in this way suffer from the fact that there is no bound on the maximum included angle. In the case of an anisotropic mesh with aspect ratios greater than 1000:1, for example, the resulting elements can have included angles much greater than 179 degrees, rendering these elements useless for most types of analysis.

The method presented here generates anisotropic unstructured surface meshes with elements that can have aspect ratios well-beyond 1000:1, but with maximum included angles that are bounded and typically near 90 degrees. It employs an **advancing normal** technique [11][12], meaning that nodes are inserted by advancing along curves emanating normal to the boundary of the domain to be meshed. The result is a semi-structured mesh comprised primarily of anisotropic, quadrilateral elements in layers that are parallel to the boundary. The elements normally vary gradually from highly anisotropic at the boundary, to nearly isotropic in the interior of the domain.

Unlike traditional approaches to advancing front and advancing normal meshing, which directly define the mesh as new nodes are inserted, this method only indirectly defines the mesh as it proceeds. The key concept is that there exists, at any stage of the mesh refinement, a valid triangulation of the initial boundary plus the nodes that have been successfully inserted. When a new node is proposed, standard meshing operators are employed to determine which existing triangle contains the proposed node, and how to update the mesh if the new node is accepted. The result is an algorithm that is extremely stable and robust - changes to the node distribution scheme will have no effect on the reliability of the basic algorithm.

The algorithm consists of four primary steps. First, each loop of the mesh is loaded into memory, and normals to each vertex are computed and adjusted if the angles between successive vertices are too large. This greatly improves the quality of cells emanating from sharp corners. In the second step, layers of semi-structured cells are formed. Vertex normals are smoothed relative to each other to improve cell orthogonality as they approach the interior of the mesh. Proposed vertex locations are then computed based on the normal, initial spacing, and the prescribed cell growth rate. A 3-D anisotropic transform is then constructed to control edge swaps during vertex insertion. This transform creates a distinct bias in favor of recovering edges that are parallel to the original boundary, and it also permits use of the Delaunay-type vertex insertion functions already in place. The proposed vertex is then inserted

into the mesh, with failing vertices marked for local termination. A pass of isotropic edge swapping is then employed, which improves the quality of cells near colliding layers. All of the new elements comprising the semi-structured layer are then frozen. The third primary step involves an isotropic edge-swapping pass to improve the quality of the coarse cells in the interior. Finally, the interior of the mesh is refined using the standard isotropic Delaunay techniques.

Among the more important aspects of the algorithm are the criteria for vertex rejection, described below.

- The spacing at the vertex is zero (no prescribed advancement)
- The normal vertex spacing is approximately the same as the tangential node spacing, resulting in isotropic triangles
- The tangential spacing is substantially larger than the edge length specified by the background function
- The tangential spacing is substantially smaller than the original tangential spacing along the boundary
- The new vertex would lie outside the mesh, inside an element that has been frozen, too close to an existing vertex, or too close to a cell that has been frozen. (This globally detects collisions between fronts)
- The new vertex cannot be connected to its predecessor along the same normal
- The new vertex does not form a quad element

An example of the anisotropic triangle meshing algorithm is provided in Figure 11.

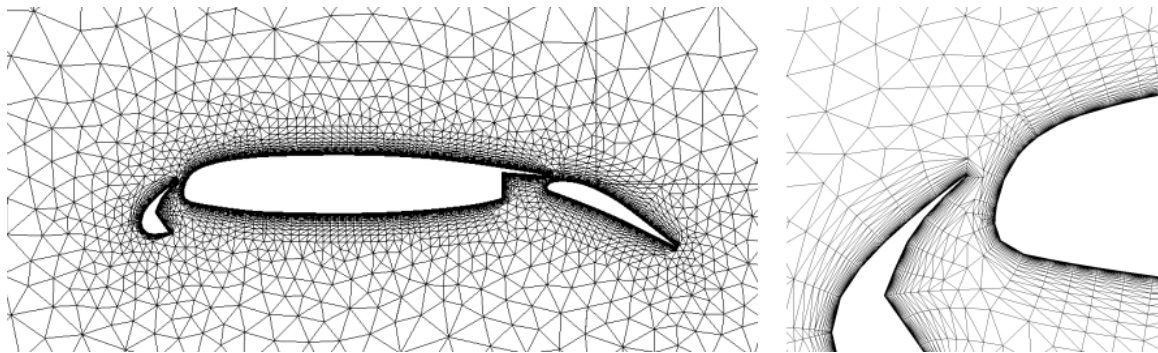


Figure 11. Anisotropic Triangle Meshing on Airfoil

VII. Tet Block Post-Processing

The high aspect ratio and associated small volume of anisotropic cells near the geometry makes it commonplace to produce an order of magnitude more cells than would typically be formed with a purely isotropic approach. The resulting mesh blocks can become unwieldy, to the point where representing the entire mesh by a single domain renders it impossible to export for analysis within the memory constraints of a 32-bit application. Two possible solutions to this problem are to reduce the overall cell count in the mesh via tet-to-prism recombination, and to subdivide the problem domain automatically into more manageable blocks. As described below, each solution has been implemented in a way that does not compromise the anisotropic mesh quality.

For small initial spacings with moderate growth rates, it is usually possible to deform the entire mesh completely for a number of layers before the first vertex movement is terminated due to cell quality failure, collision with another part of the front, or full growth to an isotropic size. As long as a layer is completely advanced, each triangle on the previous layer will have exactly 3 tetrahedra superimposed on it, formed by advancing each of the triangle vertices individually. Since the triangles on the original front are replaced with an equivalent set of triangles, the topology of the triangle front is exactly maintained across the layer, suggesting that tets in the layer can be replaced by an equivalent prism layer with 1/3 as many cells. Specifically, each set of 3 tets superimposed on a triangle is replaced with a single prism cell consisting of 2 triangle faces and 3 quadrilateral faces.

The algorithm constructed for this purpose attempts to form as many prism layers as possible. The number of suitable layers is dependent on the results of four global tests, each potentially reducing the number of viable layers for prism recombination. First, the history of the vertex deformation process is reviewed, returning the number of layers formed for each independent front before the first vertex deformation failure is encountered. Next, the symmetry surface meshes are checked to determine the maximum number of layers in which there exists a 1-1

relationship between vertices in successive layers. Actual symmetry decimation history is then reviewed to count the number of layers in which all of the vertices on symmetry surfaces were properly decimated. Finally, a geometric check of all potential recombined prism cells is made to determine the number of layers in which all new prisms pass the positive volume and user-set quality criteria. The minimum of these 4 numbers corresponds to the number of prism layers that will be recombined from the anisotropic tet block. The act of recombination causes the anisotropic tet block to be reduced and a total of N new prism blocks to be formed, where N is equal to the number of independent fronts defined by geometrically disjoint collections of surface meshes and collections of surface meshes surrounded totally by symmetry meshes. Note that the recombination of tets into prisms will not degrade the quality of the mesh in any measurable way, because all prisms are subject to the same cell quality criteria as their originating tets. This quality evaluation on the prism element is necessary because it is possible for each of the 3 tets to pass a quality criterion while its combined prism does not.

In the instances where only a relatively small number of prism layers are extracted from an anisotropic mesh, the remaining mesh may still be too large for use as a single domain in the analysis software. Rather than having the user split the overall domain into smaller regions and run the anisotropic mesher on each region, it is more logical to form a single anisotropic block, splitting it for export after completion. In this way any artifacts in the anisotropic mesh due to pre-splitting into multiple blocks beforehand are avoided.

The algorithm developed for this purpose is designed to split the collection of tets in a way that preserves the original block boundaries. As such the resulting reduced blocks are usually concentric in nature. The procedure begins by forming a list of all of the triangles on the outer surface of the block, and by assigning their vertices a layer value of 1. Next, all other vertices are assigned layer numbers based on their topological distance from the surface. This is done by querying each cell containing a vertex with a marked layer number, and by assigning all unmarked vertices in the cell an incremented value of the layer, repeating the procedure until all vertices are assigned a value. Next, tetrahedra are introduced into the triangular front, one layer at a time. If a given tet has the proper layer number (a tet's layer number is equal to largest layer number among its vertices) and it contains a triangle on the front, the triangular front is locally modified to cover the tet. For simple cases this amounts to removing the triangle and replacing it with the tet's other 3 triangles. The process is repeated until either all tets on the first layer are covered by the front, or the number of tets behind the front exceeds a pre-determined target size calculated from the number of cells in the block and the user-set block cell limit.

Several special heuristics were necessarily added to the cell-covering algorithm to insure that the front has no local pinching and forms no orphaned pockets of cells during the triangle front advancement process. These rules insure that the triangle front is valid after each modification. In some instances the tet covering is unable to encapsulate all tets adjacent to the outer block surface. This can happen, for example, if the outer surface is in close proximity to an inner surface. In these cases, block vertices are again assigned layer numbers, and a new triangular front is formed, comprised initially of the 4 triangles on a tetrahedron found to lie near the topological center of the mesh. The triangular front is then grown outward from this tet, covering cells in the same way as before.

When complete, there will generally exist two blocks of tetrahedra – one containing approximately the number of target cells, and the other containing all remaining tets in the block. The splitting algorithm is applied recursively to the larger block until the blocks all satisfy the user-set maximum cell count, or until topology prevents the blocks from being subdivided further. The algorithm is also applied to prism blocks formed from the anisotropic tet elements. Prism blocks are split at the layer level into blocks that satisfy the same user-set requirement.

VIII. Applications

The anisotropic tet and tri meshing algorithms introduced in this work have been fully implemented into the Gridgen software [13][14], a general-purpose grid generation package. A minimal set of user parameters is required to control their operation. Unless indicated otherwise, the attributes below are available for both triangles and tets.

TotalLayers – The total number of tri/tet layers that will be generated before the algorithm is stopped. This is in fact an upper limit, because all vertices on the front could be halted due to size and/or intersection constraints prior to the prescribed number. If the value is set to 0, the entire surface/volume mesh will be isotropic.

NumPrismLayers - (tet only) (\leq **TotalLayers**) Specifies the number of layers to be deformed before vertex decimation is enabled. Since decimation changes the overall triangle topology, suppressing these two operations increases the likelihood of the front remaining topologically constant. Each constant-topology layer of tets could

later be recombined into prism elements, resulting in a threefold reduction in the number of cells with no reduction in solution accuracy.

Δs_0 – The cell height for the first layer. As described in Section III, the cell heights assigned at the vertex level control the quality of the mesh. Alternatively, symmetry boundary conditions (tet only) can be applied, deformation can be disabled, or isotropic elements can be assigned. These values are applied at the curve/surface mesh level.

growthRate – The ratio of cell heights applied between successive layers.

F_s (safety factor) – (tet only) Multiplicative factor of the cell height at which the deformed vertex must not intersect the triangle front. This maintains a controllable buffer between encroaching fronts that is easily filled with isotropic tet cells.

F_b (blend factor) – (tet only) Multiplicative factor of the cell height at which the deformed vertex is perturbed in the direction of the adjacent edges. If any intersections of the perturbed vertex are detected, the vertex is marked for decimation. Small values of **F_b** increase the number of layers with which anisotropic cells are blended into isotropic elements.

Quality Criteria – (tet only) All tetrahedra created by deformation and decimation are subject to a positive volume constraint. Four other quality criteria may also be enabled, including Angle Skew, Centroid Skew, Volume Skew and Max Angle [15]. Tets that fail to meet all enable criteria are discarded.

Quality Delay – (tet only) The number of layers formed before a deforming vertex failing the cell quality criteria (other than negative volume cells) will be terminated. This parameter is used to preserve the topology of the deforming front even after a failing cell is found, so that as many layers of the mesh as possible may be recombined into prism elements.

The anisotropic tet meshing procedure has been applied to the DLR-F6 geometry, a benchmark case provided in the 2nd AIAA Drag Prediction Workshop held in 2003 [16]. The DLR-F6 geometry is a wing-fuselage-pylon-nacelle configuration imported into Gridgen as an IGES [17] file (Figure 12). The 58 trimmed surfaces within the IGES representation were then stitched into a watertight solid model [5] consisting of 5 engineering surfaces, or quilts, separated at geometric discontinuities. Each quilt was then automatically meshed with refinement based on proximity to the edge curves, the local curvature, and the local bending angles. An outer triangular surface mesh was also constructed, as were symmetry triangular meshes at the fuselage midline. The fully enclosed triangular meshes were then passed into the anisotropic mesher with a surface spacing of 0.0006, a growth rate of 1.5, and nominal safety and blend factors of 0.5. A total of 30 anisotropic layers were formed in approximately 20 minutes on a 3.2 GHz P4 computer, resulting in a total of 3.47 million tets. Near-body tets were then recombined into a prism block of 16 layers (0.75 million cells) and a nominally isotropic tet block of 1.2 million cells, amounting to a 44% cell reduction. The sharp trailing edge of the wing caused some cells in the vicinity to have relatively high maximum bend angles, but only 17 of the 3+ million cells were greater than 175 degrees, with the maximum still less than 179 degrees.

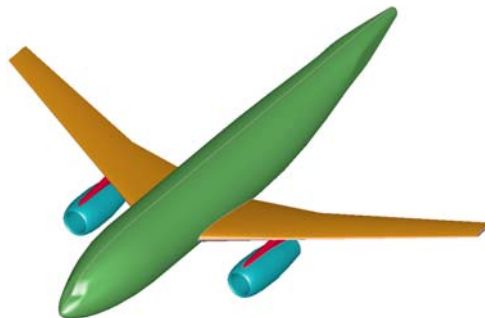


Figure 12. DLR-F6 geometry

Figure 13 depicts a fuselage-station cut of cells near the wing root quarter chord. Note the degree of clustering near the body and the smooth transition to isotropy away from the body. Also note the smoothness with which

tetrahedra connect to the triangles on the plane of symmetry. This required the surface mesh triangles to be generated with identical Δs_0 and $growthRate$ values as the anisotropic tet mesh. Figure 14 presents a closer view of the tetrahedra in the concave corners in the wing-pylon-nacelle junctions. Cells were able to deform out of these corners to the desired degree, adequately resolving the boundary layer region.

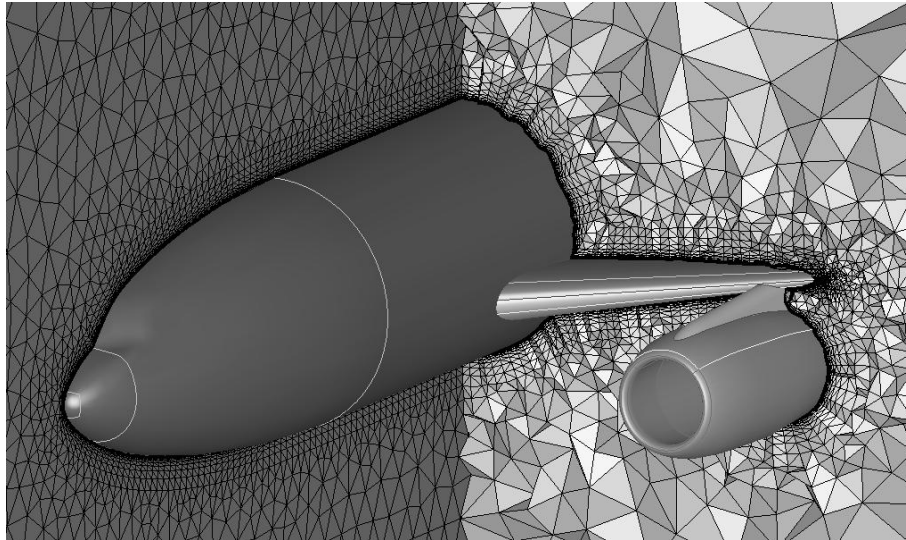


Figure 13. Cross-Sectional Cut of Anisotropic Tets on DLR-F6

The second application of the anisotropic tet meshing procedure is a grid around a trapezoidal multi-element wing attached to a simplified fuselage shape. As shown in Figure 15, the wing consists of 3 fully disjoint geometric components in close proximity to each other. The grid was constructed with isotropic surface meshes on the wing surfaces and the fuselage. A symmetry surface grid was formed on the $y=0$ plane using the anisotropic surface mesher with an initial spacing of 0.015, a $growthRate$ of 1.3 and a total of 20 requested layers. These same attributes were then applied to the anisotropic tet mesher, which resulted in a total of 3.2 million tet elements. Planar fuselage and wing cuts from this mesh are displayed in Figure 16 and Figure 17, respectively.

To improve resolution of the leading edges of the three wing components, a second mesh was also constructed, this time with anisotropic triangular elements biased in the direction of local curvature. For this example, an anisotropic triangle mesh serving as a symmetry surface to the tet mesher was placed on the flat section of the fuselage (see Figure 11). Figure 18 illustrates the successful transition of cells from the anisotropic triangles on the wing surface to isotropic tet cells on the interior of the mesh.

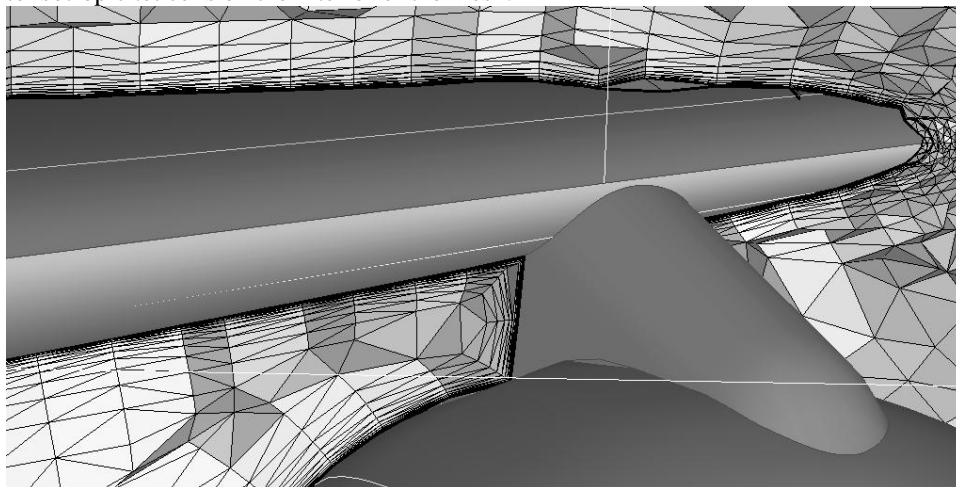


Figure 14. Closeup of Wing-Pylon-Fuselage Junction on DLR-F6

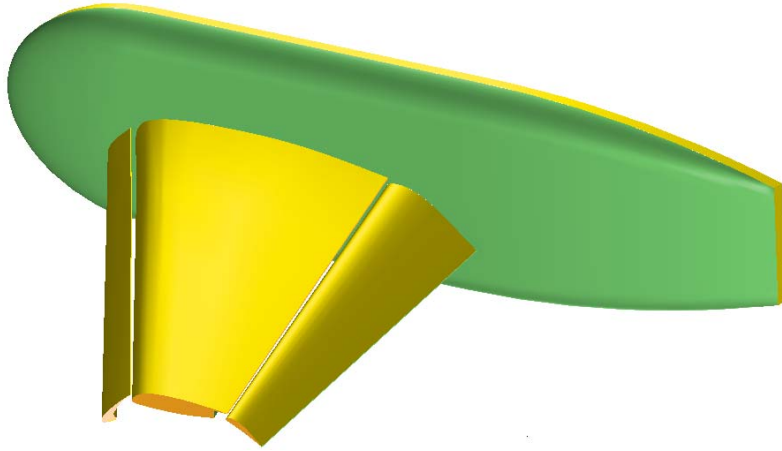


Figure 15. Trapezoidal Wing Geometry

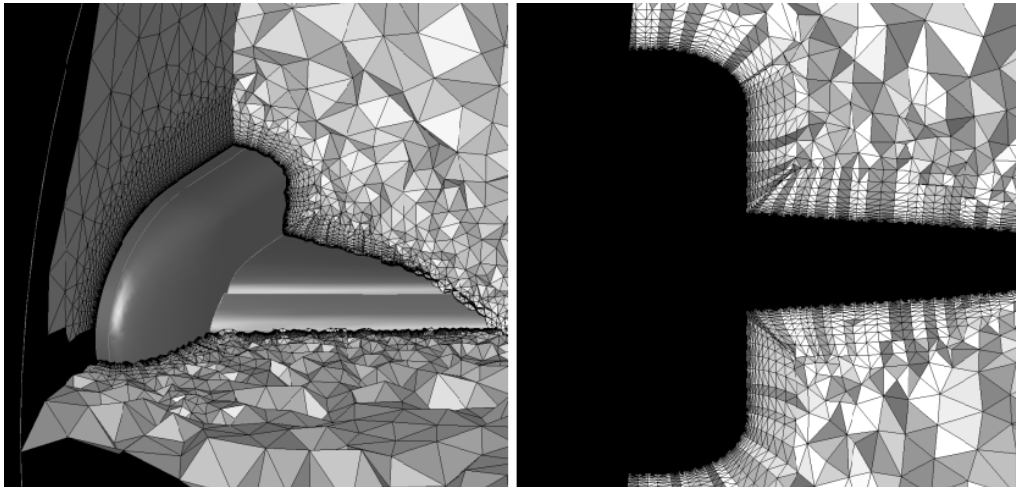


Figure 16. Trap Wing Anisotropic Mesh Fuselage Cross-Sections

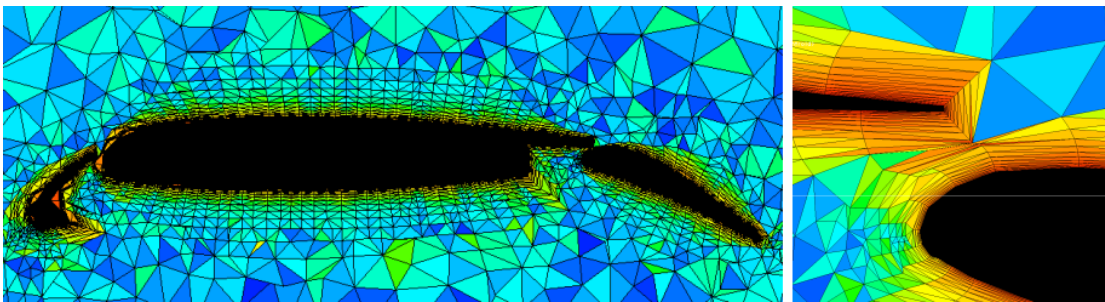


Figure 17. Trap Wing Anisotropic Mesh Wing Cross-Sections

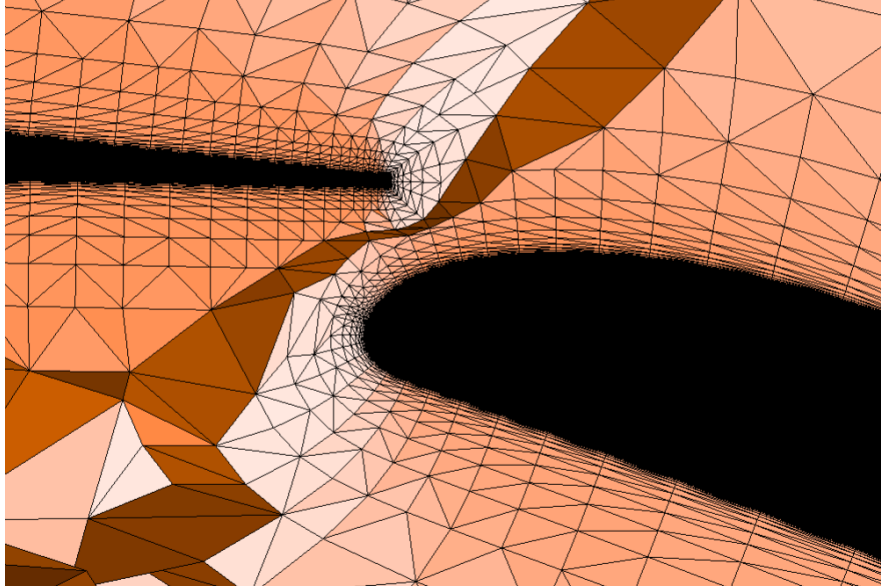


Figure 18. Anisotropic to Isotropic Tet Transition on Trap Wing

IX. Conclusions and Future Work

This paper introduces a method of generating anisotropic tetrahedral cells that is based on repeatedly deforming vertices on watertight triangular fronts. The use of binary space partition trees allows efficient and accurate deformation of the vertices while preventing intersection with other fronts. Proposed vertex deformations are accepted as long as the tetrahedra formed by perturbing the vertices from their previous position satisfies a positive volume as well as a user-set array of quality criteria. Vertex decimation techniques have been introduced that allow triangle cells on a symmetry surface to be agglomerated by tets formed from neighboring deforming vertices. Vertex decimation has also been incorporated successfully to provide a smooth transition from anisotropic triangle elements on the geometry surface to isotropic tetrahedra further into the interior. When the anisotropic mesher has completed, a generally isotropic surface triangulation is then passed into an isotropic tet mesher, with the resulting tets combined with the anisotropic tets to form a seamless block. Post-processing methods have been described that facilitate the use of these meshes with analysis software. Specifically, a prism reconstruction algorithm has the potential to reduce the number of cells in the entire mesh by up to a factor of 3, and an automatic tet splitting algorithm reduces individual block size to a user-prescribed value.

Though the success of this method described in this paper gives credence to its viability, there are a few open issues with the current implementation. First, conspicuously absent among the deformation and decimation operations is a refinement operator that would insert vertices onto the front at triangle centroids or edge bisectors. Such an operator is occasionally required when the normals of adjacent vertices point away from each other, thereby increasing the size of the local triangle beyond its target size. A procedure for cell refinement was developed for this method, and was in fact described in some detail in earlier drafts of this paper. However, a harmonious co-implementation of the deformation, decimation, and refinement operations has not yet been developed. In the current implementation, vertices introduced via triangular refinement tend to be terminated prematurely due to the combination of intersections with the rest of the front, local heights reaching the isotropic limit, and termination of adjacent vertices at an earlier layer. In order to allow the fronts to deform for as many layers as possible, cell refinement operations are currently disabled in the working code. This is obviously a topic for later investigation.

Though cell quality failure and front intersection/proximity are the two geometric events that trigger termination of a deforming vertex, the majority of the vertices are terminated because they lie adjacent to a vertex terminated at a previous layer. This latter criterion can result in meshes that appear to have terminated prematurely in relatively benign regions of the mesh geometry. Unfortunately, this type of vertex termination is necessary to prevent multiple layers of tets from connecting to the same terminated vertex. Experimental controls that extend the number of layers (beyond 1) before a vertex and its terminated neighbor is halted are already in place, but thus far have seen limited success. Fortunately, this problem can be eliminated completely in many instances with more careful construction of the surface mesh to be deformed. Nevertheless, avoidance of this issue regardless of the surface meshes is an obvious design goal.

Finally, the next version of these techniques will also likely address code efficiency. The vertex decimation and the tet block splitting algorithms are two areas already identified for closer investigation via source code profiling.

X. References

- ¹Baker, T.J., Vassberg, J.C., "Tetrahedral Mesh Generation and Optimization.", 6th International Conference on Numerical Grid Generation in Computational Field Simulation, Cross, M., ed., International Society of Grid Generation, pp 337-350, (1998).
- ²George, P.L., Hecht, F., and Saltel, E., "Automatic Mesh Generator With Specified Boundary," Computational Methods in Applied Mechanical Engineering, Vol 92, pp. 269-288, 1991.
- ³Pirzadeh, S., "Unstructured Viscous Grid Generation by Advancing-Layers Method," AIAA Paper 93-3453-CP, 1993.
- ⁴Marcum, David L., "Generation of Unstructured Grids for Viscous Flow Applications," AIAA Paper 95-0212, 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan, 1995.
- ⁵Eccles, N.C., Steinbrenner, J.P., and Abelanet, J.P., "Solid Modeling and Fault Tolerant Meshing- Two Complementary Strategies," AIAA Paper 2005-5237, AIAA Nth Computational Fluid Dynamics Conference, Toronto, Canada, June 2005.
- ⁶Zeid, I., "CAD/CAM Theory and Practice", McGraw-Hill Series in Mechanical Engineering, McGraw-Hill Inc., ISBN 0-07-072857-7, 1991.
- ⁷Kallinderis, Y. and Ward, S., "Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations," 10th Applied Aerodynamics Conference, AIAA Paper 92-2721, Palo Alto, CA, June 1992.
- ⁸Karman, Jr., Steve L., "Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing," AIAA Paper 2006-0531, 44th Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan, 2006.
- ⁹Bossen, F.J., P.S. Heckbert, "A Pliant Method for Anisotropic Mesh Generation," Proceedings of the 5th International Meshing Roundtable, pp. 64-74, Pittsburgh, PA, October, 1996.
- ¹⁰Castro Diaz, M.J., and F. Hecht, "Anisotropic Surface Mesh Generation," INRIA Research Report No. 2672, 1995.
- ¹¹Hassan, O., M. Marchant, K. Morgan, E. Probert, N. Weatherill, and D. Marcum, "The Numerical Simulation of 3D Turbulent Transonic Flows Using Unstructured Grids," AIAA Paper 94-2346, June, 1994.
- ¹²Marcum, D., "Efficient Generation of High-Quality Unstructured Surface and Volume Grids," Proceedings of the 9th International Meshing Roundtable, New Orleans, LA, October, 2000.
- ¹³Chawner, J.R., Steinbrenner, J.P., and Wyman, N., "Hybrid Grid Generation for Complex Geometries Using Gridgen," 7th International Grid Conference, Whistler, British Columbia, Canada, 2000.
- ¹⁴Steinbrenner, John .P., Nick J. Wyman and John R. Chawner, "Fast Surface Meshing on Imperfect CAD Models", 9th International Grid Conference, New Orleans, LA, 2000.
- ¹⁵Gridgen V15 Users Manual, Pointwise, Inc., ondemandmanuals.com, 2006.
- ¹⁶2nd AIAA CFD Drag Prediction Workshop, 2003, <http://www.cfd-online.com/Forum/news.cgi/read/875>.
- ¹⁷Reed, K., "The Initial Graphics Exchange Specification (IGES)," NISTIR 4412, National Institute of Standards and Technology, 1991.