SCI INSTITUTE
TECHNICAL REPORT

Hexahedral Mesh Generation Constraints

Jason F. Shepherd

UUSCI-2006-010

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

February 13, 2006

Abstract:

Hexahedral finite element meshes have historically offered some mathematical benefit over tetrahedral finite element meshes in terms of reduced error, smaller element counts, and improved reliability, especially with respect to finite element analyses within highly elastic, and plastic, structural domains. However, hexahedral finite element mesh generation continues to be extremely difficult to perform and automate, with hexahedral mesh generation taking several orders of magnitude longer in time to complete over current tetrahedral mesh generators. In this paper, I focus on delineating the known constraints associated with hexahedral meshes, formulating these constraints utilizing the dual of the hexahedral mesh. Utilizing these constraints, it will be possible to highlight areas where better knowledge and incorporation of these constraints can augment existing algorithms, predict failure of specific methods, and suggest some additional methods for extending the class of geometries which can be hexahedrally meshed.

THE UNIVERSITY OF UTAH

Hexahedral Mesh Generation Constraints

Jason F. Shepherd[1]

Scientific Computing and Imaging Institute, Salt Lake City, UT
jfsheph@sci.utah.edu

Hexahedral finite element meshes have historically offered some mathematical benefit over tetrahedral finite element meshes in terms of reduced error, smaller element counts, and improved reliability, especially with respect to finite element analyses within highly elastic, and plastic, structural domains. However, hexahedral finite element mesh generation continues to be extremely difficult to perform and automate, with hexahedral mesh generation taking several orders of magnitude longer in time to complete over current tetrahedral mesh generators. In this paper, I focus on delineating the known constraints associated with hexahedral meshes, formulating these constraints utilizing the dual of the hexahedral mesh. Utilizing these constraints, it will be possible to highlight areas where better knowledge and incorporation of these constraints can augment existing algorithms, predict failure of specific methods, and suggest some additional methods for extending the class of geometries which can be hexahedrally meshed.

1 Introduction

Numerical approximation methods, including finite element, finite difference, and finite volume methods, are mathematical methods used to model various scientific and engineering phenomena for a wide variety of disciplines, including structural mechanics, dynamics, heat transfer, and computational fluid dynamics. Because of the flexibility of these methods, the problems to which they can be applied to obtain solutions is growing and expanding daily, and is currently being utilized in fields as diverse as cellular microbiology and quantum chromodynamics to star and galaxy formation studies (for example, see [17, 21, 18]).

From problem formulation to final solution, a numerical approximation is typically separated into three distinct steps, i.e. model creation, simulation or analysis, and result visualization. In the scientific community, these three areas receive a great amount of attention from various research groups according to

specific difficulties associated with each area. In model creation, a great deal of effort is currently placed on geometric creation and mesh generation. The meshes produced are transformed into a set of matrix equations to 'simulate' the physics of the phenomena being modeled. These matrices are solved utilizing better and faster numerical solvers, and produce voluminous amounts of data. This resultant data is processed using a variety of complex data-mining and visualization techniques in an effort to derive useful conclusions from the model.

In this paper, we will be targeting the mesh generation area associated with the various numerical approximation methods. For example, to perform a finite element analysis, the problem domain is first discretized into a finite set of sub-domains, where each sub-domain is known as an element and are typically one, two, or three dimensional entities. Taken together, the set of finite elements over the problem domain is known as a 'mesh'. This description of a finite element mesh is also similar for many of the other numerical approximation methods.

The most common forms of elements utilized in numerical approximations are triangles or quadrilaterals in two-dimensions, and tetrahedral or hexahedral elements in three-dimensions. While some approximation methods allow mixtures of theses element types, homogenous meshes are typically preferred, where possible.

To reduce the amount of time to prepare a model, automated meshing algorithms have been developed for creating triangular, quadrilateral, and tetrahedral meshes for a very generalized class of geometries. In the case of tetrahedral meshing, algorithms are available which can generate greater than 400 thousand tetrahedra per minute [16]. Automated hexahedral mesh generation algorithms, however, are available for only a very limited class of geometries.

Despite the availability of an automated hexahedral mesh generation algorithm, hexahedral meshes are often preferred over tetrahedral meshes, when available, for a variety of reasons:

1. Tetrahedral meshes typically require 4-10 times more elements than a hexahedral mesh to obtain the same level of accuracy [32, 11].
2. In some types of numerical approximations (i.e. high deformation structural finite element analysis), tetrahedral elements will be mathematically 'stiffer' due to a reduced number of degrees of freedom associated with a tetrahedral element [1, 10]. This problem is also known as 'tet-locking'.
3. There is also a preference by many analysts to utilize a hexahedral mesh, when available.

Because of the limited class of geometries for which hexahedral meshes can be built, a great deal of time to generate a hex mesh involves decomposing (cutting up) a model into pieces for which a known hexahedral mesh generation algorithm will succeed. The processing of geometry for hexahedral mesh

can take several months for a generalized model, whereas tetrahedral meshes can often be created in a matter of hours or days [33, 34].

Because of the ongoing need and desire for hexahedral meshes, developing a better understanding of the underlying constraints which make hexahedral meshing difficult could result in dramatic reductions in the amount of time necessary to prepare a hexahedral finite element model for analysis.

2 Background

Numerous algorithms exist for producing hexahedral meshes. However, no one algorithm is completely successful at generating provably correct and robust hexahedral meshes. The most basic form of a hexahedral mesh generation algorithm stems from the mesh of a cuboid (i.e. subdivisions of a single hexahedral element). For complex geometries, meshes can be obtained by decomposing the initial geometry into a collection of cuboids. For geometries that are encountered frequently, the decomposition can be retained, or stored, as a 'primitive' decomposition, and when the common shape is encountered again meshing can be nearly automatic [7].

Because decomposition of a model into cuboids and/or primitive shapes can be tedious, complex, and difficult to automate, alternative algorithms were sought which did not require geometric decomposition. One particular algorithm for hexahedral mesh generation, plastering [6], was modeled after the successful 'paving' quadrilateral mesh generation algorithm [4]. Paving generates unstructured quadrilateral meshes without first requiring geometric decomposition of the surface.

Following the approach taken in paving, plastering utilizes and advancing front approach of inserting hexahedron into the interior of a solid in the following manner. Given a solid with a prescribed quadrilateral mesh on its boundary, insert a hexahedral element interior to the solid where one of more boundary faces of the inserted hex contains at least one of the quadrilaterals on the solid's boundary. The boundary of the solid is updated such that the new boundary takes into account the inserted hex. This process is repeated iteratively, advancing the boundary of the solid inward until closure can be obtained.

One of the difficulties encountered with the plastering method is that as the advancing front begins to close, matching faces on the front with other faces can be combinatorially difficult, or impossible, resulting in solids with void regions left in the mesh where the boundary did not close. Finding a hexahedral mesh for these void regions can be an extremely difficult task.

2.1 The Dual of a Hexahedral Mesh

In late 1993, hexahedral mesh generation for these types of difficult problems were viewed as a topological problem by Murdoch [24, 25] and Thurston [30].

They realized that utilizing the dual subdivision of a hexahedral mesh yields a structure of surfaces (sheets), where the structure of the sheets determines the structure of the hexahedral mesh (and vice versa) according to some specific topological requirements. The structure of these dual sheets was dubbed the "Spatial Twist Continuum" by Murdoch in his doctoral dissertation [24].

For a given quadrilateral mesh on a surface (disk), the dual of the mesh is created by drawing a line segment across each quadrilateral and connecting the opposite edges of the quadrilateral. For a quadrilateral, there are two such line segments, one for each of the opposite pairs of edges on a quadrilateral element. By iterating on this process for an entire surface mesh, it is soon realized that each of line segments from a quadrilateral connect neighbor to neighbor until the line segments form either a closed curve, or the resulting curve has end points at the surface boundary. Each curve in the dual is dubbed a 'chord' in the quadrilateral mesh. An example of a mesh (also known as the 'primal'), with its dual, is shown in Figure 1.

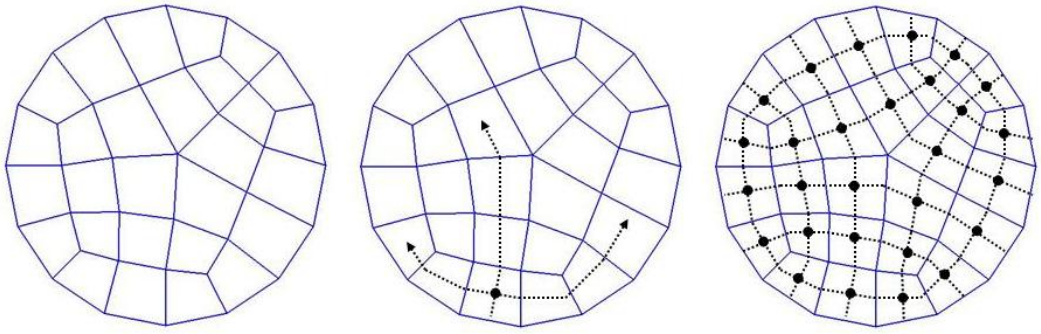

Fig. 1. A quadrilateral mesh on a circular disk (left). The dual of a quadrilateral mesh is created by line segments (chords) connecting opposite edges of an individual quad, and traversing all of the edges of all of the quads on the mesh (middle). The intersection of two chords is called a centroid and is dual to a quadrilateral. The complete dual is on the right.

Several important items to note can be gleaned from the dual representation of the quadrilateral mesh:

1. The intersection point of two chords is known as a 'centroid', and a centroid is dual to a quadrilateral in primal space.
2. Each chord represents a 'stack' of quadrilaterals in the primal mesh.
3. All chords must either have endpoints on the boundary of the surface, or they must form a closed curve within the boundary of the surface. (I.e. there is no chord with an endpoint in the interior of the surface.)
4. All chords intersect at nearly orthogonal angles to one another.
5. Chords cannot be tangent to other chords.

6. The 'size' of the primal mesh and the number of elements local to an area of the surface is a function of the density of chords and chord intersections relative to that locale on the surface.
7. Due to observation 3 above, the parity of the edges around a surface admitting a quadrilateral mesh must be even.

Extending these observations of quadrilateral meshes to hexahedral meshes, we can formulate the dual of a hexahedral mesh. Since each hexahedral element will consists of three pairs of opposing quadrilaterals (similar to the two opposing edges for quadrilateral elements), we can draw a line segment between the centers of each of the opposing faces of the hexahedra. For the dual of a single hex, we can already begin to draw several conclusions:

1. The three line segments within a hexahedral element intersect at a single point, which again is known as a centroid. A centroid in a 3D mesh is dual to a hex element in the primal mesh.
2. The three segments intersect nearly orthogonally to each other.

By incorporating additional hexes adjacent to this single hexahedron, we observe that there are some additional interactions occurring within the dual space that are not readily apparent with just the chords and centroids. As more hexahedra are added to the mesh, we observe that, in similar fashion to the chords of a quadrilateral mesh, the chords within a hexahedral mesh define a stack of hexahedra. However, we also observe that these stacks now interact in two directions resulting in 'layers' of elements. A layer of elements corresponds to a surface in the dual space of the mesh. This dual surface has been referred to as a 'sheet' [22], an 'interior surface' [12], or as a 'twist plane' [25, 24]. For the purposes of this paper, we will utilize the term 'sheet' to apply to these dual surfaces. An example dual subdivision of a hexahedral mesh is shown in Figure 2 for clarity.

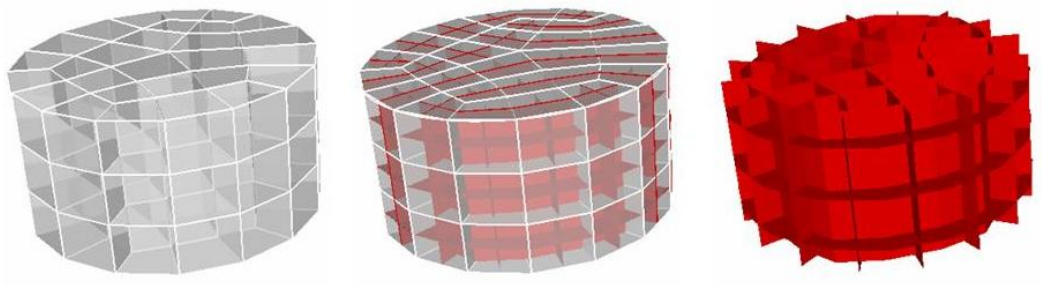

Fig. 2. Example of a cylinder meshed with hexahedra (left). The picture in the middle shows the hex mesh with its dual subdivision in red. The sheets of the mesh are shown on the right.

Incorporating the sheets into the dual space of a hexahedral mesh, we observe the following interactions:

1. There are no instances of more than three sheets intersecting at a single point. An intersection of three sheets at a single point is called a 'centroid'. A centroid is the dual form of a single hexahedron in primal space.
2. Each sheet in the dual space represents a 'layer' of hexahedral elements in the primal mesh.
3. All sheets either form a closed shell within the mesh space, or have terminating edges around the boundary of the mesh (i.e. there is no sheet which terminates in the interior of the hexahedral mesh.)
4. All sheets intersect nearly orthogonally with any other sheet.
5. Sheets cannot be tangent to other sheets.
6. The 'size' of the primal mesh and the number of elements local to an area of the surface is a function of the density of sheets and the triple intersections of sheets relative to that locale within the mesh.
7. There exists a mapping for each sheet within the hexahedral mesh to a quadrilateral mesh on that sheet. Utilizing observation 3 above, and observation 7 from quadrilateral duals, it is apparent that the parity of quadrilaterals on the boundary of the hexahedral mesh must be even. (A proof of this can be found in [22])

Utilizing these observations with the theorems of topology, Thurston theorized [30] that for a given solid, any quadrilateral mesh composed of an even number of quadrilaterals should admit a compatible hexahedral mesh within the solid. This theory was later proved by Mitchell in [22], and shown to have linear complexity by Eppstein in [12]. These proofs, however, have not resulted in any practical algorithms for generating hexahedral meshes in an arbitrary solid which will be suitable for analytic use.

So, while it is true that, topologically speaking, any solid on whose boundary contains a quadrilateral mesh with even parity will admit a compatible hexahedral mesh, there must also exist some geometric and quality requirements which must be satisfied to enable a hexahedral mesh to be usable for analytic methods, such as finite element analysis.

3 Methods

In this section, we will explore and derive some of the constraints for hexahedral mesh generation. By understanding the constraints, we can propose some possible extensions to existing algorithms, along with some new approaches, to enhance the class of geometries for which these hexahedral meshing algorithms are applicable. The last part of this section will highlight some algorithms which may be utilized to help satisfy some of the constraints for various situations.

3.1 Hexahedral Mesh Constraints

Topologic Constraints

In delineating the topological requirements of a hexahedral mesh, we will utilize the constraints given by Mitchell in his proof of hexahedral mesh existence [22]. While his proof started by considering a solid homeomorphic to a ball with an even-parity quadrilateral mesh on the boundary, it should be recognized that the constraints given will also apply to any arrangement of sheets within a solid where no boundary mesh on that solid has been specified. That is, if we start with an arrangement of sheets and place these sheets interior to a solid, the topological constraints enumerated below will still hold with only minor concessions to incorporate the boundary of the solid which does not contain a quadrilateral mesh. The requirements for sheets near the geometric boundary will be discussed in the next section.

Before outlining the topological constraints, we will define some additional needed terminology. First, let us clarify our definition of sheet to be the following: For a given problem domain in $\Re^3$ (where the boundary of the space is defined by the boundary of the solid(s) to be meshed), a sheet is a homotopic surface which effectively divides the problem domain into two separate regions. This corresponds to the observation made earlier that a sheet must either be closed within the boundary of the mesh space, or else it must span the mesh space. Utilizing the above definition for a sheet, a collection of sheets will intersect to form the following collection of sub-entities, as follows (see also Figure 3):

- A centroid (i.e. a 0D element) occurs at the intersection (local) of 3 sheets. (It is possible that a single sheet may self-intersect, such that a single sheet is effectively 2 (or all 3) of the local sheets needed to form a centroid.)
- A chord (i.e. a 1D element) is produced along the intersection (local) of 2 sheets.
- A 2-cell (i.e. a 2D element) is a polytope on a sheet resulting from an intersection with 1, or more, other sheets, where the polytope boundary are the chords produced by the sheet intersection.
- A 3-cell (i.e. a 3D element) is a volumetric polytope resulting from the division of the original space by one, or more, sheets, where the boundary of the 3-cell are the 2-cells enclosing the sub-space.

Utilizing the above definitions, the topological requirements, from Mitchell [22], for a given arrangement of sheets to produce a hex mesh are as follows:

1. Each internal 2-cell is contained in exactly two distinct 3-cells.
2. Each face contains at least one lower dimensional face (excepting centroids).
3. Each chord segment must contain two distinct centroids.
4. Every internal cell contains at most one surface cell of one lower dimension.

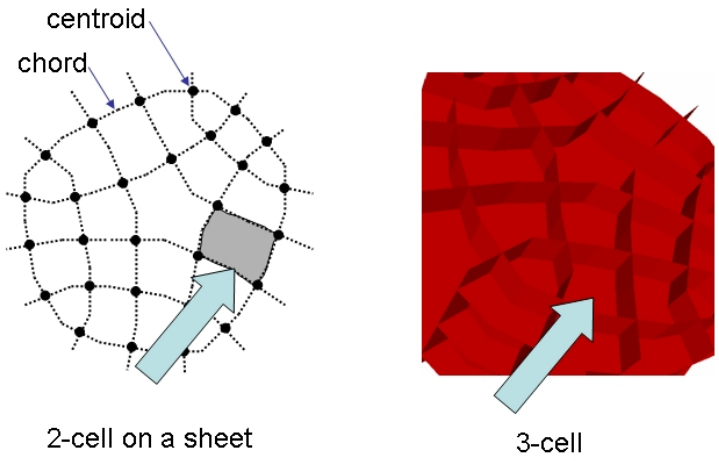

Fig. 3. Entities in the Dual

5. Each internal chord segment must be contained in exactly four distinct 2-cells.
6. Each centroid is contained in six chord segments. Note, also, that each chord segment at a centroid is paired with another chord segment belonging to the same chord.
7. Two 3-cells have, at most, one 2-cell in common.

When any of the conditions above are violated, the result will be either a degeneracy or void regions in the resulting hexahedral mesh. Proofs for each individual requirement can be found in [22].

For purposes of aiding the reader's intuition, these constraints can in most cases be viewed as:

1. Only three sheets can intersect at any given centroid.
2. Sheets cannot be tangent with another sheet.
3. Sheets must span the space, or form a closed surface within the space.
4. When traversing the centroids along a single chord, consecutive instances of a single centroid are not permitted.

Boundary Constraints

In outlining the topological constraints of sheets for hexahedral meshes, we paid very little attention to the geometric boundary of the space, or solid. In this section, we make several observations regarding the geometric boundary of solids and formulate the additional requirements necessary to ensure compatibility of the interior sheets with the geometric boundary of a space.

For clarity, we need to define what is meant by a solid geometry into which we will be placing a mesh. A solid geometry consists of five main entity types, namely vertices, curves, surfaces, solids (or volumes), and collections of volumes (these may be referred to as an 'assembly'). These entities are often arranged in a hierarchical structure, as shown in Table 1. This hierarchical structure is often referred to as the topology of the solid geometry. (Please note the distinction between the mesh topology and the geometric topology of the solid.)

While there may be special cases to the hierarchy shown in the table (i.e. curves without endpoints, surfaces without curves, etc.), most of these special cases can be ignored, or remedied by introducing the necessary entities to match the definitions shown in the table without affecting the solid geometry representation.

Geometric Entity	Bounding Entities
Vertex	None
Curve	Two Vertices
Surface	One or more Curves
Volume	One or more Surfaces

Table 1. Hierarchical arrangement of geometric entities.

Understanding the interaction between these entity types is important to understanding how the geometric constraints on hexahedral meshes help to capture the geometric entities. For instance, in order to mesh a surface, the mesh topology of the surface must somehow incorporate the curves and vertices 'owned' by that surface. For many solid geometries, it may be easy to capture the geometric shape of the object, but can be very difficult to capture the geometric topology of the object.

One other item that we should identify is the distinction between mesh entities and geometric entities. We will utilize the following terminology, shown in Table 2, in order to distinguish between mesh entities and geometric entities. From this table, we can also note the hierarchical relationship inherit in the mesh entities, which is similar to the relationship between geometric entities noted earlier.

Dimensionality	Geometric Entity	Mesh Entity
0D	Vertex	Node
1D	Curve	Edge
2D	Surface	Quadrilateral (or Triangle, etc.)
3D	Volume	Hexahedron (or Tetrahedron, etc.)

Table 2. Relationships between geometric entities and mesh entities.

At this point, we can also construct a table (Table 3) indicating entity correspondence between the dual and primal spaces for hexahedral meshes.

Primal Entity	Dimension	Dual Entity	Dimension
Hex	3	Centroid	0
Quad	2	Chord	1
Edge	1	2-Cell	2
Node	0	3-Cell	3

Table 3. Conversions between dual and primal entities.

Geometric Surface Constraints

A couple of interesting observations can be made from Table 3. First, note that a chord is dual to a quadrilateral. Since chords were drawn between hexahedra, the chord between two centroids was dual to the quad shared by these two hexahedra. Therefore, in some sense the chord can be viewed as an approximation to the normal of the quadrilateral between two hexes. At the boundary of our geometry, we will want to align the chords with the local surface normals to obtain a reasonable quadrilateral mesh on the boundary surface. Otherwise, the resulting quadrilaterals on the surface will not appropriately approximate the geometry.

The second interesting observation is that since there are 3 pairs of chords associated with each centroid, a chord whose endpoint is on the geometric boundary is roughly orthogonal to the other two chords. By definition, a chord can only be contained in at most two sheets. Because one chord approximates the normal, the other two chords emanating from the centroid must exist within a single sheet. Therefore, in an area which is local to our original chord resides a single sheet which is geometrically similar to the boundary surface but is offset by the length of the boundary terminating chord. Utilizing this observation, we can formulate the following constraint (see Figure 4) .

> For each surface of our solid, there exists one sheet within the solid which is geometrically similar to the local boundary surface but offset a distance which is a function of the size of the mesh local to that boundary (i.e. the local chord length).

Geometric Curve Constraints

With respect to curves on the boundary, we can make a similar deduction that for each curve on the boundary, there must be one or more chords which, in a local sense, are geometrically similar to the boundary curve, but offset a distance which is again a function of the mesh size in the boundary curve's locale. Since each chord is created from the intersection of two sheets, we can infer the following constraint (see Figure 5):

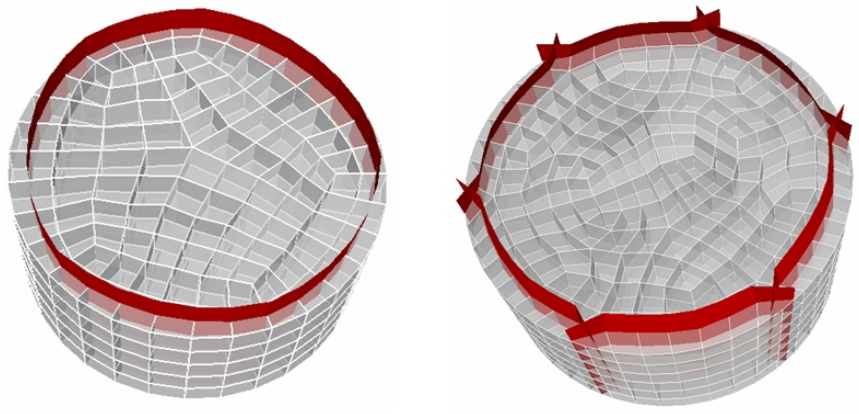

Fig. 4. Image showing how a sheet captures the geometric boundary. The picture on the right shows a single sheet capturing the cylindrical surface, while the picture on the left (of a different mesh) shows the same surface being captured with multiple sheets.

> There will be one intersecting sheet pairs within the solid corresponding locally to each curve on the boundary, which are offset a distance related to the mesh size local to the curve.

Geometric Vertex Constraints

Continuing the arguments from above as we move down the element hierarchy to determine the constraints for each vertex on our boundary, we can show that a vertex on the boundary will correspond to a centroid which is offset from the vertex by a distance related to the local mesh size. However, there is a complicating difference with vertices, in that there it is not necessary for a vertex to correspond to a single centroid. Rather, a vertex may correspond to many centroids within a single solid.

We can re-write this observation in terms of sheets, such that, for each centroid, there exist a local, triple-set of sheets whose intersection form a centroid which corresponds to a vertex. However, because a vertex on the boundary can correspond to one or more centroids, there will be cases when this vertex will correspond to more than one triple-sheet pairing. A few examples are shown in Figure 6.

Our constraint in relation to geometric vertices is:

> There will be at least one triple-sheet pairing which corresponds to each vertex on the boundary. This triple-sheet pair is equivalent to a centroid, and is offset a distance related to the mesh size local to the vertex.

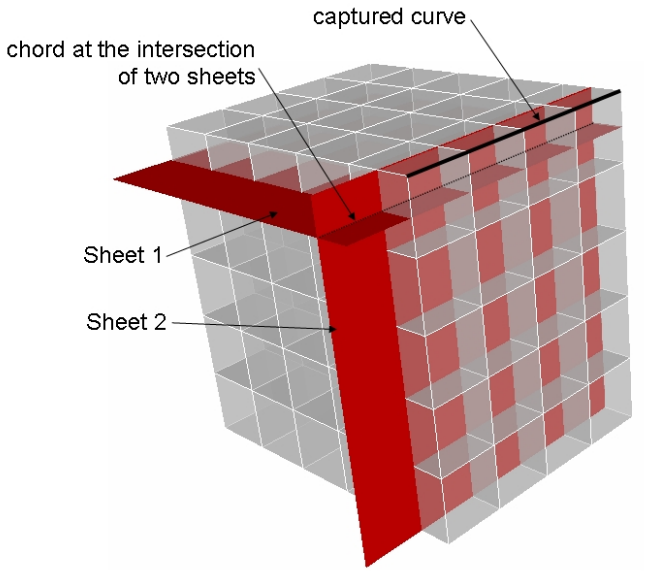

Fig. 5. Capturing a curve utilizing an offset chord (from the intersection of two sheets).

In this case, utilizing the sheets in meeting this constraint is helpful instead of the centroids because there are usually geometric curves emanating from each of the vertices. The sheets utilized to capture the geometric surfaces and curves will also be the same sheets which capture the vertices. The convergence of many sheets around some vertices must be handled with care to maintain the topological constraints local to a vertex (i.e. Only three sheets can intersect at a single point, etc.).

Quality Constraints

Specifying constraints in relation to the quality of the mesh is often difficult because a complete understanding of how mesh quality affects analysis error is largely unkown. The only consistent constraint placed on mesh generation is that the jacobian (scaled) of each mesh element must be positive (i.e. non-inverted). In this section, we will consider how the geometric properties of a sheet affect the overall quality of the resulting hex mesh, while the ultimate goal will be to determine the necessary conditions on a sheet to have a resulting non-inverted mesh. We will begin by first discussing the sheets relating to the ideal mesh as developed for finite elements, and then by discussing how geometric and topologic modifications to the ideal sheets change the underlying quality of the resulting hexahedral meshes.

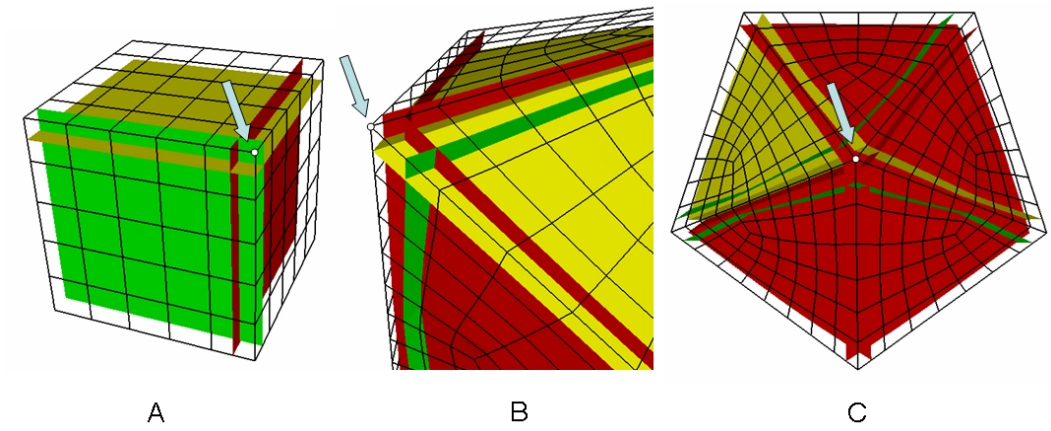

Fig. 6. At least three sheets are necessary to capture a geometric vertex in a given mesh topology (A). However, more than one triple-pair of sheets is not exclusive for each vertex. For geometric vertices whose valence is higher than three, more than one triple pair is necessary to capture the all of the geometric features related to the vertex. A four-sided pyramid (B) requires four sheets (creating two distinct centroids, or two triple-pairs) and a five-sided pyramid (C) requires five sheets (creating three distinct centroids, or three triple-pairs) to succinctly capture the geometric features associated with the vertex. In (B), there are two red sheets, one green, and one yellow shown. In (C), there are two red, two green and one yellow sheet shown.

Quality Considerations

In terms of hexahedra, the ideal element for which finite element basis functions are formulated is a cube of six quadrilaterals of equal area, with eight edges of equal length, and which are mutually orthogonal to each other. Such cubes are easily subdivided by dividing each edge by two and each quadrilateral into four smaller, but mutually equivalent quadrilaterals. The arrangement of the sheets in dual space for such an ideal mesh is simply a collection of Cartesian planes which subdivide the mesh as described earlier in the dual construction of a mesh (see Figure 7).

We can, again, make some general observations on this arrangement of sheets as compared to other possible arrangements, as follows:

1. From the topological constraints earlier, we know that at most three sheets can intersect at a single point. From the ideal mesh, it is apparent that a perfectly orthogonal intersection of the three sheets is desired. Deviations from orthogonal intersections of the sheets will produce 'skewing' of the sheets (see Figure 8).
2. Element sizing information is determined directionally as a function of the local density of the sheets in an area. To increase the element size in one direction, decrease the density of locally parallel arrangement of sheets.

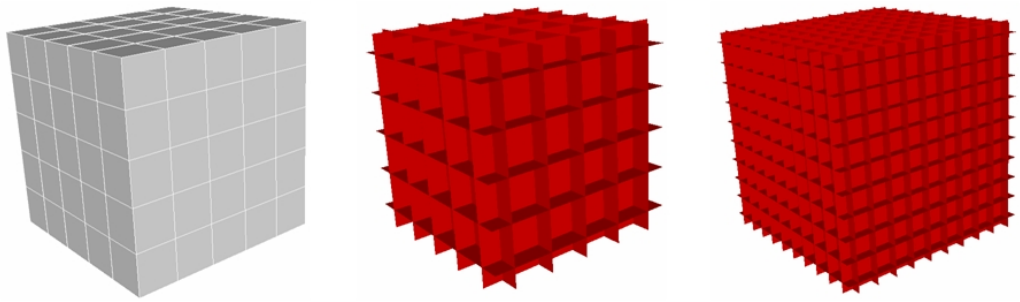

Fig. 7. The ideal mesh (left) with the induced sheet arrangements at two different mesh sizes (the sheet arrangement on the right is twice as dense as the sheet arrangement in the middle).

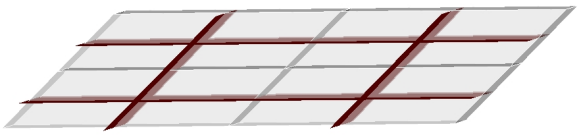

Fig. 8. As sheet intersections deviate from orthogonality, the skew of the mesh increases.

A dramatic transition in the density of sheets is undesirable for quality reasons (see Figure 9).

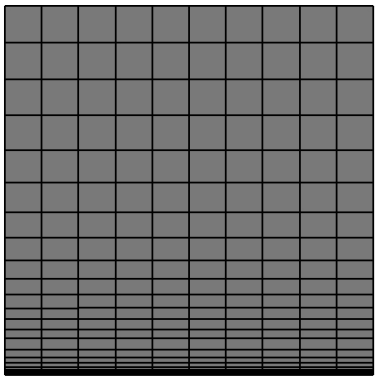

Fig. 9. Note that as the sheet density increases in one direction only, the element aspect ratio increases as can be seen in this figure.

3. The ideal mesh contains perfectly planar sheets. Therefore, we desire the local curvature of a sheet to have as maximal an absolute value as possible.

Curvature of the sheets causes 'keystoning' of the elements, where the length of one edge is substantially different than it's opposite edge. This phenomenon can be readily seen in Figure 10, as we increase the curvature of a single sheet of elements.

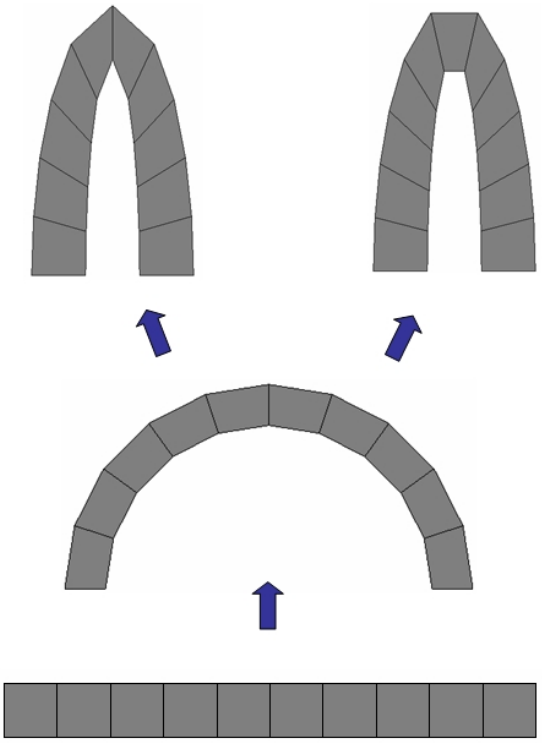

Fig. 10. Increasing the curvature of a single sheet results in 'keystoning' of elements where one edge shrinks in size while the other grows as the curvature of the sheet increases.

4. With the exception of self-intersecting curves connecting self-intersection point along the boundary of a sheet, sheets are not allowed to self-intersect internally within the mesh space.
5. The regular topological arrangement of the sheets, as shown in the ideal mesh, is also desirable (and is essentially required in finite difference calculations). In finite element methods this requirement has some flexibility, but maintenance of this regular topology, where possible, has some additional benefits from several algorithmic standpoints including element numbering, matrix formulation, compression, etc. Additionally, alternative topological structures have a tendency to interfere with the ability to obtain optimal quality values for constraints 1 and 3 above, although

it is possible to use alternative topological arrangements (and may be necessary for some classes of solids) with acceptable quality results.

Determining Quality Constraints

The quality of a hexahedral mesh, as determined from its dual subdivision, is largely unexplored. From the quality considerations enumerated earlier, it is known that there exists a correlation between the conformation of the sheet and the ultimate quality of the hexahedral mesh. However, it is unknown what interactions are allowable to obtain a non-inverted mesh. While, the final constraints are not currently known, some case studies are may be of value in determining some of the necessary conditions for producing quality hexahedral meshes.

- *Low sheet curvature is important for high quality* - Figure 11 shows a sheet from a volume which was meshed via the whisker weaving algorithm [13]. At the base of the trough in this sheet is a collection of inverted elements which are currently untangle-able [15, 31]. Mesh smoothing can only improve the sheet conformation in a limited fashion due to how this sheet must interact with the surrounding sheets. Some efforts involving mesh topology reconfiguration (for instance, mesh-flipping [2, 3, 29]) may aid our ability to untangle these meshes, but the underlying constraint involved with how this particular configuration prohibits untangling of the mesh is unknown.

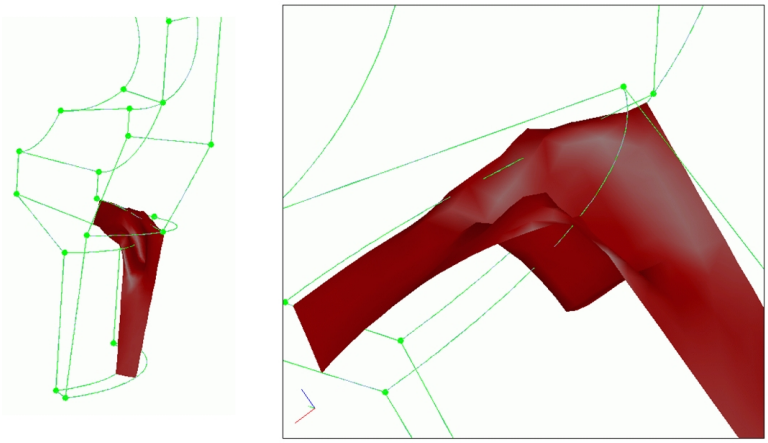

Fig. 11. Two views of a sheet with a high curvature trough in a solid. The high curvature at the trough of the sheet shown corresponds to hex elements which cannot be untangled in the resulting mesh.

- *The surface mesh is important for high quality* - In some algorithms, the boundary of a sheet is defined by the dual cycle of a quadrilateral mesh on a surface. Some of the boundaries defined by arbitrary quadrilateral meshes imply extremely complex geometric definitions for the resulting sheet that corresponds to the defined boundary. One such boundary is shown in Figure 12, where the dual cycle wraps around itself several times in one area of the solid. Because of current inability to untangle the resulting mesh in this object, there must be a quality constraint which is not being enforced in the generation of this particular mesh topology.

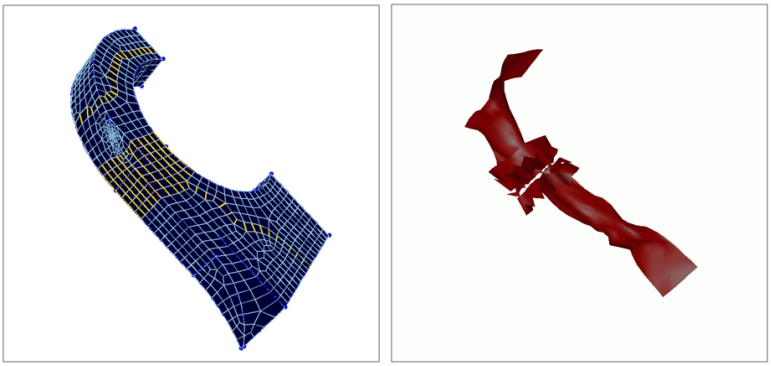

Fig. 12. The surface mesh (shown on the left) has a dual cycle which spirals up to the top of the image. The resulting hex mesh, generated with WhiskerWeaving, cannot be untangled. The sheet generated by WhiskerWeaving is shown on the right.

- *The mesh approximates the sheet* - Recent work done by Suzuki, et al. [27] has explicitly defined the sheets interior to a solid. By generating an interior surface for the dual cycle of Schneider's pyramid [26] and subsequently satisfying all necessary topologic constraints for a hex mesh, it was felt that a reasonable quality mesh would result. However, the resulting mesh had elements which were untangle-able. Comparing the sheet defined by the hex elements with the sheet created by the authors showed a very rought approximation of the desired interior surface (see Figure 13. Increasing the number of elements subsequently improved some areas of quality, but the underlying constraint to direct where to place more elements to guarantee a non-inverted mesh is still missing.

These case studies indicate that there is still a great deal of information which is not currently understood with regards to generation of quality (or, at least, non-inverted) hexahedral meshes. The considerations listed at the beginning of this section will play an important role in understanding the necessary hexahedral quality constraints, however, a broader understanding of these considerations with the interaction of multiple sheets is still not well

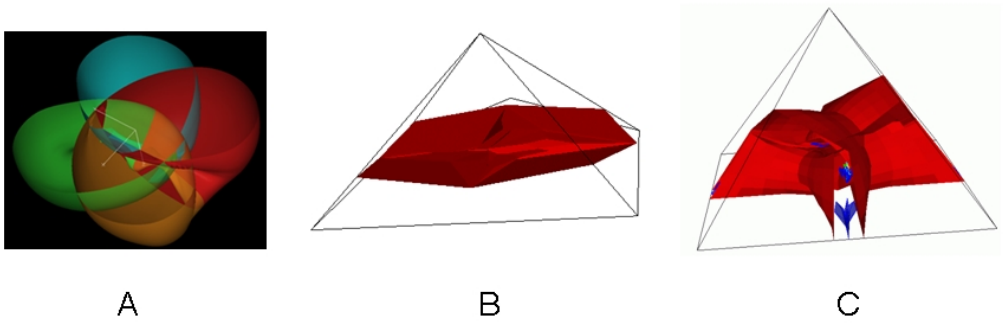

Fig. 13. The boundary and interior surface, shown in (A), correspond to the boundary and an interior surface for Schneider's pyramid [26]. The resulting hex mesh has a sheet (B) which approximates the surface in (A), but the facets tend to flatten the intended sheet conformation. By increasing the resolution (via dicing [20]) the approximation of the surface is better and has fewer regions of negative jacobian elements (some negative jacobian elements are shown in the image).

understood. A better understanding of these interactions will be useful in directing current and on-going research.

Algorithmic Constraints

In a presentation to attendees at the 9th International Meshing Roundtable, Blacker [5] re-iterated several algorithmic considerations which are utilized in evaluating the performance of a hexahedral mesh generation algorithms. These considerations can be viewed as additional constraints on a hexahedral mesh generation algorithm, because failure to satisfy any of these constraints would result in an algorithm which would not be utilized. These additional constraints are, for the most part, similar to the constraints listed above; however, because some additional practical constraints are also present, we will list the considerations in their entirety.

- *Geometric Generality* - "The algorithm should handle a large class of geometries with arbitrary complexity and detail."
  This consideration questions how well an algorithm incorporates the requirements listed in the previous three sections. In hexahedral mesh generation, most algorithms will assume that one set of constraints are fixed and then work to optimize the other two constraint sets. By fixing one set of constraints, an algorithm will only work for the geometric set of problems for which the constraint set is fixed.
- *Geometric Matching* - "The algorithm should contain the geometric features identified in the solid being meshed."
  In some sense, this consideration asks how well an algorithm satisfies the geometric constraints of a hexahedral mesh. In some cases, an algorithm

working under the assumption of a fixed mesh topology may have difficulty capturing specific geometric features since the mesh topology utilized will not be readily homeomorphic to the geometric boundary.

- *Boundary Sensitivity* - "The algorithm should produce high-quality elements at the domain boundary."
  Historically, this consideration dealt with the regularity of the mesh topology at the boundary coupled with the how nearly orthogonal the sheets at the local boundary locations are with respect to the boundary surface.
- *Orientation Insensitivity* - "The orientation of the geometry should not affect the mesh generated by the algorithm."
  Another way of saying this would be: "The algorithm should produce identically (or nearly so) meshes despite rigid body transformations of the geometry." This consideration can normally be satisfied by anchoring the mesh topology to the geometry.
- *Bad Geometry Tolerant* - "The algorithm should be able to operate despite gaps, overlaps, holes, etc. in the geometry."
  Upon determination of a mesh topology which generally satisfies the geometric and quality constraints, a general operation in most algorithms is to assign each individual mesh entity to an owning geometric entity. With 'bad geometry' these assignments can be ambiguous at times. The algorithms ability to deal with these ambiguities is the subject of this consideration.
- *Size Control* - "The algorithm should match the desired element sizing constraints throughout the domain."
  Because of the topological constraints of hexahedral mesh generation require a sheet to span the mesh space; at times it can be difficult to control the sheet density in local areas while still satisfying the mesh quality constraints. This consideration pertains to how well an algorithm is able to balance these oft-times conflicting requirements.
- *Speed* - Generate reasonably large meshes (>1M elements) in 'interactive' time. Tetrahedral meshing speeds are desirable.
  Tetrahedral mesh generation algorithms are currently available which are generating on the order of millions of elements per minute on single processor machines. A desirable hexahedral algorithm would be able to generate hexes in the same order of time, while still adequately meeting the previous algorithmic considerations for its operation class of geometries.

3.2 Constraint-satisfying Methods

Over the years, several methods have been developed which make it possible to improve the flexibility of existing hexahedral meshing algorithms. In this section, we will highlight some of these methods and show how they help to satisfy some of the fundamental constraints for hexahedral meshing of solid models. In particular, we will discuss methods for inserting and extracting

sheets in existing mesh topologies. The methods we will highlight are pillowing [23], dicing [20, 19], mesh-cutting [9], grafting [14], and sheet extraction [8].

Inserting Sheets

Pillowing

During the development of the whisker weaving algorithm [28, 13], Mitchell et al. consistently encountered meshes where two neighboring hexes shared two faces, called a 'doublet' (see Figure 14). This situation is undesirable because it is practically impossible to move the nodal locations in such meshes to generate a reasonable jacobian values within the mesh for subsequent analyses. A simple, but powerful, technique called 'pillowing' [23] was developed to locate and place a mesh refinement that effectively removed the doublets from the mesh. In terms of the dual of the mesh, pillowing is essentially a sheet insertion operation, where a new sheet is inserted which effectively splits the doublet hexes into multiple hexes, and eliminating the problematic mesh topology.

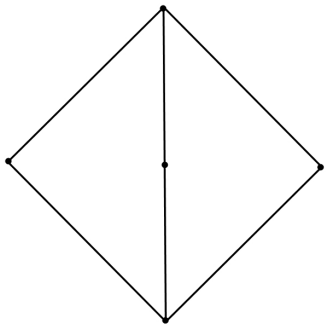

Fig. 14. A quadrilateral doublet, where two adjacet quadrilaterals share two edges. Similar types of doublets occur in 3D with adjacent hexes sharing two or more quadrilaterals. The scaled jacobian for both elements, as shown, is zero, and while node movement strategies can improve the jacobian value for one of the two quadrilaterals, simultaneous improvement of the jacobian value for both quadrilaterals is not possible.

The pillowing method turns out to be powerful, not for it's ability to remove doublets, rather it provides a fairly straight-forward approach to insert sheets into existing meshes. The sheets can be inserted utilizing the primal elements of an existing mesh, and without explicitly creating a geometric definition for the sheet and calculating the intersections with the other local sheets in the space.

The basic pillowing algorithm is as follows:

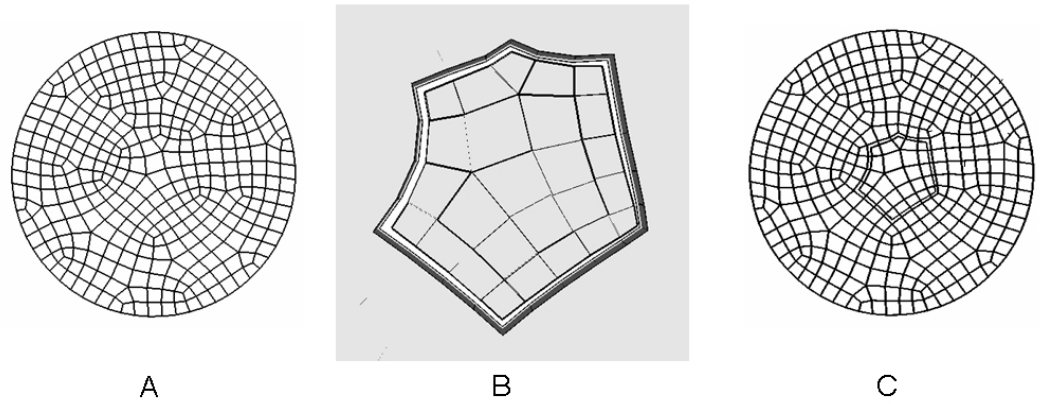

Fig. 15. A basic pillowing operation starts with an initial mesh (A) from which a subset of elements is defined to create a shrink set. The shrink set is separated from the original mesh and 'shrunk' (B), and a new layer of elements (i.e. a dual sheet) is inserted (C) to fill the void left by the shrinking process.

1. *Define a shrink set* - For our purposes, this step involves dividing the existing mesh into two sets of elements: one set for each of the half-spaces defined by the sheet to be inserted. One of these two sets of hexahedral elements comprises the shrink set. The choice of which one should be the shrink set is arbitrary, although the best algorithmic choice will be the set with the fewest number of elements.
2. *Shrink the shrink set* - This step essentially creates a gap region between the two previous element sets (see Figure 15. The difficulty in this step involves splitting the shared nodes, edges, and quads in the existing mesh, while maintaining the appropriate correspondence of the mesh entities with the geometric topology.
3. *Connect with a layer of elements* - This step results in a fully-conformal mesh with the new sheet inserted between the original two element sets. To complete this step, an edge is inserted between each node that was separated during the 'shrinking' operation. Utilizing the quadrilaterals on the boundary between the two sets of hexes, along with these new edges, it is fairly straight-forward to determine the connectivity of all of the hexes in this new layer.

It is often desirable to perform a smoothing operation on the resulting mesh after the new sheet has been inserted to obtain better nodal placement and higher quality elements. The speed of the pillowing algorithm is largely dependent on the time needed to find the shrink set. The number of new hexahedra created will be equal to the number of quadrilaterals on the boundary of the shrink set.

Dicing

The dicing algorithm [20, 19] was created to very efficiently generate very large, refined meshes from existing coarse meshes. The generation of these very large, refined meshes is accomplished by copying the existing sheets and placing them in a parallel configuration to the sheet being copied. The basic method for dicing is as follows:

1. *Define the sheet to be diced* - An edge in a hexahedral mesh can only correspond to a single sheet in the dual. Utilizing one edge, the opposite edges of the hexahedron can be deduced as belonging to the same sheet via the definition of the dual of the hexahedral mesh. It is then possible to iterate until all of the edges associated with a single sheet in the dual are found.
2. *Dice the edges* - With the list of edges found in the previous step, dicing then splits (dices) all of these edges the specified number of times. If for instance, we wish to copy the sheet one time, then each of the edges is split once resulting in two new edges.
3. *Form the new sheets* - With each of the edges associated with the hexahedral sheet split, we can again utilize the idea that an edge can be associate with a single sheet in the mesh and form a new layer of hexahedra for each split in the original set of edges, where the hexahedral connectivity is similar to the original hexahedral layer before the edges were split.

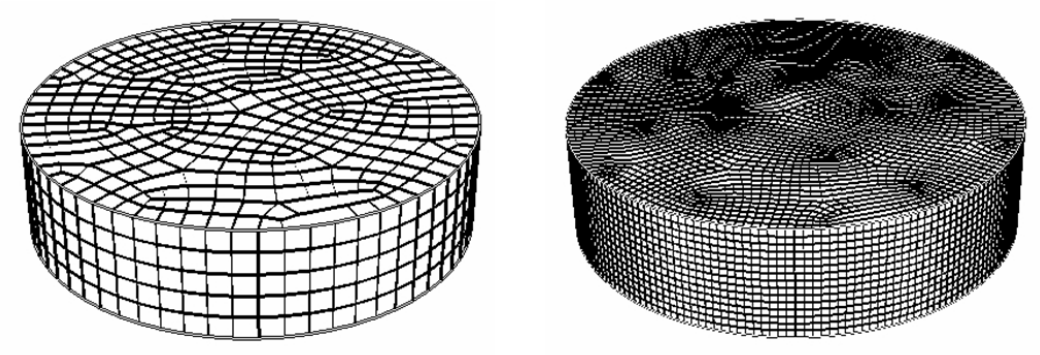

Fig. 16. The original mesh (left) contains 1805 hex elements before dicing. Each sheet in the original mesh is copied three times resulting in a mesh that is 3^3 larger, with 48735 hex elements.

Utilizing the dicing method, the number of elements increases as the cube of the dicing value. For instance, if an existing mesh is diced four times (i.e. each of the sheets in the existing mesh is copied four times), the resulting mesh

would have 64X as many elements as the original mesh. Because all search operations can be performed directly, the dicing algorithm can produce large meshes at very efficient speeds (see Figure 16).

Geometric Capture with Sheets

Mesh Cutting

The mesh-cutting [9] method is an effective approach for capturing geometric surfaces within an existing mesh topology. The mesh-cutting method utilizes the pillowing and dicing methods mentioned previously to insert two sheets which are geometrically similar to the surface to be captured. By utilizing two sheets, the result is a layer of quadrilaterals, shared by the hexes in the two sheets, which can be viewed as a set of facets geometrically approximating the surface. The mesh-cutting method entails the following steps:

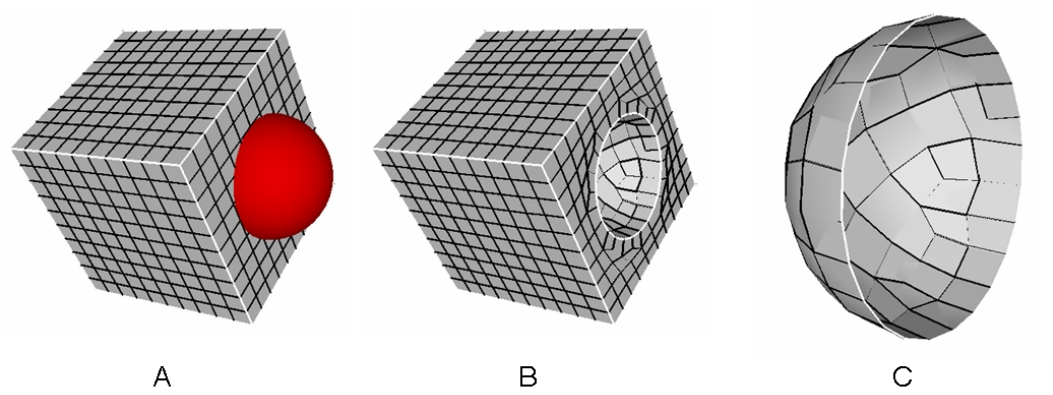

Fig. 17. Meshcutting utilizes an existing mesh and inserts new sheets to capture a geometric surface(the existing mesh is shown in (A) where the red, spherical surface is the surface to be captured.) The reulting mesh after meshcutting is shown in (B), with a close-up of the quadrilaterals on the captured surface being shown in (C).

1. *Define the pillowing shrink set* - Utilizing the surface that is to be captured in the mesh, we divide the existing mesh into two sets of elements. One of these element sets will be the shrink set, and a sheet (pillow) is inserted between the two sets of elements.
2. *Dice the new sheet* - We split the newly inserted sheet into two sheets utilizing an approach similar to dicing.
3. *Move the shared quadrilaterals to the surface* - With two new sheets defined in the mesh topology, we can find all of the quadrilateral which are

shared by the hexes between the two sheets. These quadrilaterals become the mesh on the surface we are attempting to capture (see Figure 17).

A caveat with this method is that the existing mesh topology must be fine enough to capture the detail of the surface to be inserted. Because the resulting quadrilaterals only approximate the inserted surface, if the resulting quadrilateral mesh is too coarse, the surface may not be approximated adequately enough to be resolved.

One other item to remember with this method is that since a geometric surface is being utilized to define a sheet within the mesh space it may be necessary to have the ability to extend the geometric surface in some fashion such that it meets the requirements on a sheet that it divide the mesh space. If the geometric surface is trimmed, for instance, the trimmed surface may not adequately divide the space being meshed making it necessary to provide a continuation to the surface definition to the boundary of the mesh space.

Grafting

The term 'grafting' is derived from the process of grafting a branch from one tree into the stem, or trunk, of another tree. In meshing, the grafting method was initially to be utilized for allowing a branch mesh to be inserted into the linking surface of a previously hexahedrally swept volume [14]. The grafting method would then offer a reasonably generalized approach to multi-axis sweeping.

In reality, grafting is a method which essentially captures geometric curves on previously meshed surfaces. Assuming that a hexahedral mesh exists which captures to geometry of the surface where you would like a closed set of curves placed, the method for creating a graft (i.e. capturing the geometric curve) can be outlined as follows (see also Figure 18):

1. *Create a pillowing shrink set* - In the case of grafting, the shrink set is typically defined as the set of hexes which have one quadrilateral owned by the surface and which are interior to the closed set of curves (with respect to the surface).
2. *Insert the pillow (sheet)* - By inserting the second sheet, we have essentially satisfied the hexahedral constraint for capturing a geometric curve. That is, we now have two sheets which generate a chord in the mesh which is offset from the set of curves which were the input to the grafting algorithm.

At this point, there is often some database adjustments also necessary to ensure that the new mesh entities are associated with the correct geometric entities, but the curve is essentially capture when the second sheet is inserted in conjunction with the initial set of sheets that captured the geometric surface. A similar method can be used to capture a single curve, rather than a set of curves, but it is still necessary that the sheet that is inserted to capture the curve must satisfy the definition of a sheet. That is, the sheet must completely divide the mesh space into two regions.

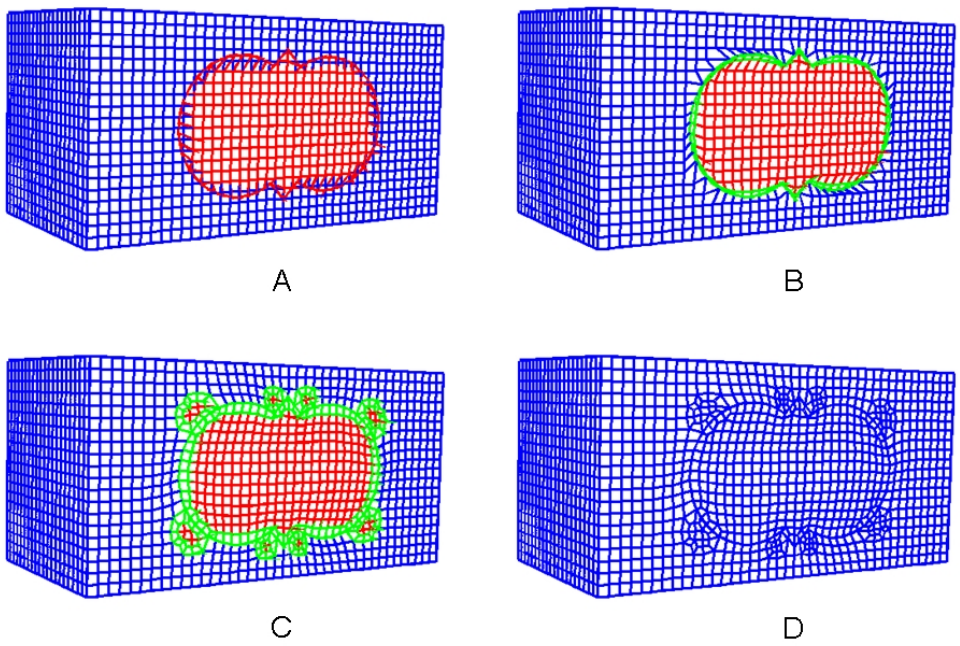

Fig. 18. In grafting, a shrink set on a existing meshed volume is defined (A) and a new sheet is inserted via a pillowing operation (B). Once the new sheet has been inserted, the nodes are positioned along the curve to be captured via a smoothing operation (C). Additional pillows can also be inserted to remove any doublet hexes that may have been created (C). The resulting mesh topology captures the geometric curve (D).

One caveat with this method: because there is no explicit steps taken to capture the geometric vertices associated with each of the curves being grafted, there is a requirement that the resolution of the trunk mesh be fine enough to be able to capture all of the curve's endpoints by moving existing nodes in the final mesh to the vertex locations. The movement of the nodes to the vertex locations must be done intelligently to avoid destroying the required mesh topology necessary to correctly capture the curve. While other sheets may be added to avoid this problem, the addition of more sheets to capture the vertices may have the negative effect of locally refining the mesh sizes and/or mesh topologies that are not as aesthetically pleasing as may be desired.

Removing Sheets

Sheet Extraction

One of the nicest things about working with the sheets in hexahedral meshing, is that all of the processes are easily reversible. It is just as easy (or easier) to

insert a sheet as it is to remove a sheet completely from the mesh. A method for extracting a sheet is detailed in [8], where the basic steps can be outlined as follows:

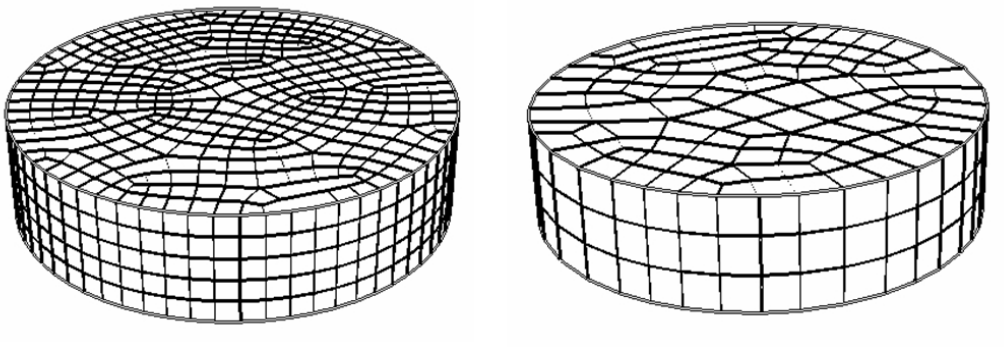

Fig. 19. Original mesh, shown on left, with 1805 hex elements. After removing approximately half of the sheets in the original mesh, the resulting mesh (right) has 342 hex elements.

1. *Define the sheet to be extracted* - Because an edge in a hexahedral mesh can only correspond to a single sheet in the dual, this step can be easily accomplished by specifying a single edge in the mesh. From this single edge, the primal mesh can be iteratively traversed to determine all of the edges which correspond to the sheet to be extracted.
2. *Collapse the edges* - With the list of edges found in the previous step, the nodes of each of the edges can be merged, effectively removing the sheet from the mesh.

There are some special circumstances that must be avoided when extracting sheets in order to avoid either degenerating the mesh or producing a mesh which is no longer conformal with the geometric topology. These situations can be avoided by checking to ensure that one of the edges to be collapsed is not the only edge on a curve, or that the nodes in an edge are not owned by different curves, etc.

4 Conclusion

A thorough understanding of the constraints associated with hexahedral mesh generation may provide the insight necessary to dramatically reduce the time required to generate these types of meshes for use in finite element analyses. Hexahedral meshes are currently necessary in many engineering analyses, most specifically highly elastic and plastic structural analyses where tetrahedral

meshes suffer tet-locking. Due to the difficulty and complexity of generating hexahedral meshes, however, tetrahedral meshes are often the only means available to perform the finite element analysis.

In this paper, we have outlined many of the necessary constraints for generating a hexahedral mesh as derived from the dual representation. By incorporating methods to satisfy ignored, or overlooked, constraints in existing hexahedral mesh generation algorithms, we can extend the class of geometries for which these algorithms might be applied. Several existing methods for satisfying individual constraints are also highlighted.

We propose several research areas to be explored to validate the list of constraints. These research tasks demonstrate application to new areas, augmentation of existing algorithms, and development of new algorithms for hexahedral mesh generation. A timeline for completing these research tasks was outlined for completion of the doctoral dissertation.

References

1. S. E. Benzley, E. Perry, K. Merkley, and B. Clark. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings, 4th International Meshing Roundtable*, pages 179–191. Sandia National Laboratories, October 1995.
2. M. Bern and D. Eppstein. Flipping cubical meshes. In *Proceedings, 10th International Meshing Roundtable*, pages 19–29. Sandia National Laboratories, October 2001.
3. M. Bern, D. Eppstein, and J. Erickson. Flipping cubical meshes. *Engineering with Computers*, 18(3):173–187, 2002.
4. T. D. Blacker. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.
5. T. D. Blacker. Meeting the challenge for automated conformal hexahedral meshing. In *Proceedings, 9th International Meshing Roundtable*, pages 11–19. Sandia National Laboratories, October 2000.
6. T. D. Blacker and R. J. Meyers. Seams and wedges in plastering: A 3d hexahedral mesh generation algorithm. *Engineering With Computers*, 2(9):83–93, 1993.
7. T. D. Blacker, J. L. Mitchiner, L. R. Phillips, and Y. Lin. Knowledge system approach to automated two-dimensional quadrilateral mesh generation. *Computers in Engineering*, 3:153–162, 1988.
8. M. J. Borden, S. E. Benzley, and J. F. Shepherd. Coarsening and sheet extraction for all-hexahedral meshes. In *Proceedings, 11th International Meshing Roundtable*, pages 147–152. Sandia National Laboratories, September 2002.
9. M. J. Borden, J. F. Shepherd, and S. E. Benzley. Mesh cutting: Fitting simple all-hexahedral meshes to complex geometries. In *Proceedings, 8th International Society of Grid Generation Conference*, 2002.
10. M. L. Bussler and A. Ramesh. The eight-node hexahedral elements in fea of part designs. *Foundry Management and Technology*, pages 26–28, November 1993.

11. A. O. Cifuentes and A. Kalbag. A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. *Finite Elements in Analysis and Design*, 12(3-4):313–318, 1992.
12. D. Eppstein. Linear complexity hexahedral mesh generation. In *12th ACM Symposium on Computational Geometry*, pages 58–67. ACM, 1996.
13. N. T. Folwell and S. A. Mitchell. Reliable whisker weaving via curve contraction. *Engineering With Computers*, 15:292–302, 1999.
14. S. R. Jankovich, S. E. Benzley, J. F. Shepherd, and S. A. Mitchell. The graft tool: An all-hexahedral transition algorithm for creating multi-directional swept volume mesh. In *Proceedings, 8th International Meshing Roundtable*, pages 387–392. Sandia National Laboratories, October 1999.
15. P. Knupp and S. A. Mitchell. Integration of mesh optimization with 3d all-hex mesh generation, ldrd subcase 3504340000, final report. SAND 99-2852, October 1999.
16. M. Loriot. Tetmesh-ghs3d v3.1 the fast, reliable, high quality tetrahedral mesh generator and optimiser (see http://www.simulog.fr/mesh/tetmesh3p1d-wp.pdf).
17. S. Means, A. J. Smith, J. F. Shepherd, J. Shadid, J. Fowler, R. Wojcikiewicz, T. Mazel, G. D. Smith, and B. S. Wilson. Impact of geometry on spatial distributions of intralumenal endoplasmic reticulum calcium under variant ip3r channel distributions. *to appear*.
18. D. L. Meier. Multidimensional astrophysical structural and dynamical analysis. i. development of a nonlinear finite element approach. *Astrophys. J.*, 518:788–813, 1999.
19. D. J. Melander. *Generation of Multi-Million Element Meshes for Solid Model-Based Geometries: The Dicer Algorithm.* Published Master's Thesis, Brigham Young University, April 1997.
20. D. J. Melander, T. J. Tautges, and S. E. Benzley. Generation of multi-million element meshes for solid model-based geometries: The dicer algorithm. *AMD - Trends in Unstructured Mesh Generation*, 220:131–135, July 1997.
21. K. A. Milton. Finite-element quantum field theory. In *Proceedings of the XIVth International Symposium on Lattice Field Theory*, volume Nucl. Phys. B(Proc. Suppl.) 53 (1997), pages 847–849, 1996.
22. S. A. Mitchell. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volumes. In *13th Annual Symposium on Theoretical Aspects of Computer Science*, volume Lecture Notes in Computer Science: 1046, pages 465–476, 1996.
23. S. A. Mitchell and T. J. Tautges. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In *Proceedings, 4th International Meshing Roundtable*, pages 231–240. Sandia National Laboratories, October 1995.
24. P. J. Murdoch. *The Spatial Twist Continuum: A Dual Representation of the All Hexahedral Finite Element Mesh.* Published Doctoral Dissertation, Brigham Young University, December 1995.
25. P. J. Murdoch and S. E. Benzley. The spatial twist continuum. In *Proceedings, 4th International Meshing Roundtable*, pages 243–251. Sandia National Laboratories, October 1995.
26. Schneiders Pyramid Open Problem, http://www-users.informatik.rwth-aachen.de/ roberts/open.html.

27. T. Suzuki, S. Takahashi, and J. F. Shepherd. Practical interior surface generation method for all-hexahedral meshing. In *Proceedings, 14th International Meshing Roundtable*, pages 377–397. Sandia National Laboratories, September 2005.
28. T. J. Tautges, T. D. Blacker, and S. A. Mitchell. Whisker weaving: A connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering*, 39:3327–3349, 1996.
29. T. J. Tautges and S. Knoop. Topology modification of hexahedral meshes using atomic dual-based operations. In *Proceedings, 12th International Meshing Roundtable*, pages 415–423. Sandia National Laboratories, September 2003.
30. B. Thurston. Geometry in action: Hexahedral decomposition of polyhedra (available from http://www.ics.uci.edu/ eppstein/gina/thurston-hexahedra), October 1993.
31. P. Vachal, R. V. Garimella, and M. J. Shashkov. Mesh untangling. LAU-UR-02-7271, T-7 Summer Report 2002.
32. V. I. Weingarten. The controversy over hex or tet meshing. *Machine Design*, pages 74–78, April 18, 1994.
33. D. R. White, R. W. Leland, S. Saigal, and S. J. Owen. The meshing complexity of a solid: An introduction. In *Proceedings, 10th International Meshing Roundtable*, pages 373–384. Sandia National Laboratories, October 2001.
34. D. R. White, S. Saigal, and S. J. Owen. Meshing complexity of single part cad models. In *Proceedings, 12th International Meshing Roundtable*, pages 121–134. Sandia National Laboratories, September 2003.