CONF-9510233--2
SAND95-2358C

Whisker Weaving: Invalid Connectivity Resolution and Primal Construction Algorithm

Timothy J. Tautges

Scott A. Mitchell*

Comp Mechanics and Visualization Dept, Sandia National Laboratories, Albuquerque, NM 87185

ABSTRACT

This paper describes the techniques used to resolve invalid connectivity created as a natural part of the whisker weaving algorithm. These techniques rely on the detection of "repeated hexes" in the STC data, which indicate face pairs which share two edges. The "repeated repeated hex" case is described in detail, including the resolution technique by which a self-intersecting whisker sheet with two independent face loops are created. The algorithm used to contruct the primal of an all-hexahedral mesh (i.e. the actual nodes and hex elements) from the connectivity data contained in the STC is also described. The primal is constructed using a "gift-wrapping" algorithm, where all the mesh edges and hexes containing a particular node are found by traversing between hexes already known to share the node. This algorithm is implemented inside the CUBIT code and is used to generate meshes for several example problems.

INTRODUCTION

The whisker weaving algorithm produces the Spatial Twist Continuum data for an all-hexahedral mesh [1] [2]. The whisker weaving algorithm constructs and represents STC entities in its datastructure that are the dual of all-hex mesh entities; these entities are shown in Table 1. This information represents connectivity of an all-hex mesh only, and contains

Table 1.Correspondence between STC and mesh entities.

STC Entity	Mesh Entity
STC vertex	Hex
STC edge	Face
STC 2-cell	Edge

* This work was supported by the U.S. Department of Energy under contract DE-AC04-76DO00789, by the Applied Mathematical Sciences program, U.S. Department of Energy Research, and by Sandia National Laboratories Cooperative Research and Development Agreement SC93/01247 (USCAR Scientific Computing for Automotive Applications Partnership).

no geometric information except for mesh faces on the geometric surface, which were input to the whisker weaving algorithm.

The STC data produced by whisker weaving is visualized in the form of sheet diagrams. For example, Figures 1 and 2 show the geometry and surface mesh for an example problem

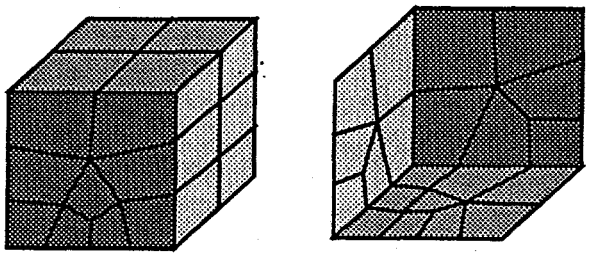


Figure 1. Geometry and surface mesh for the double fold problem.

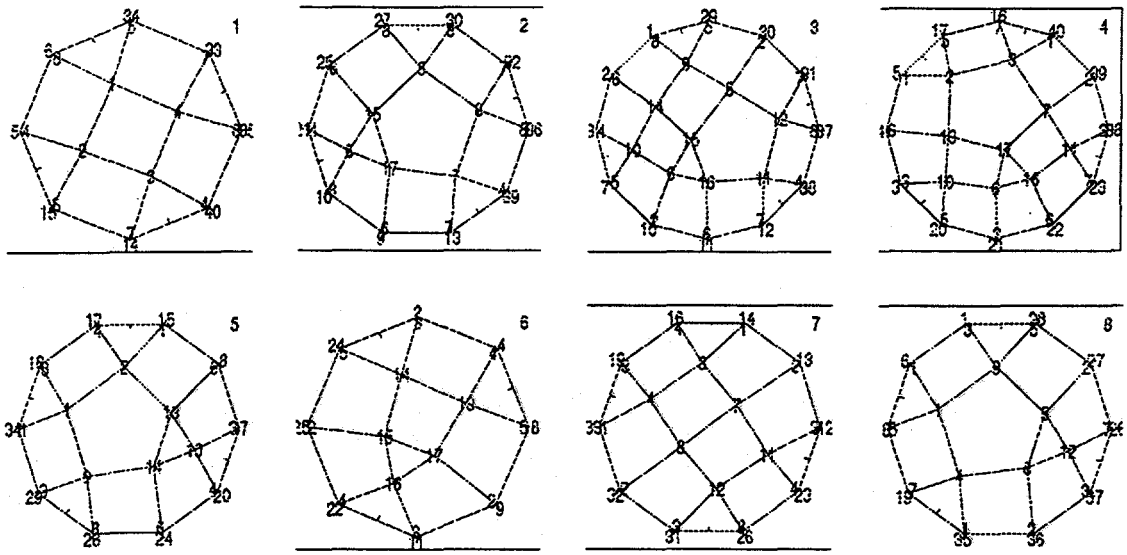


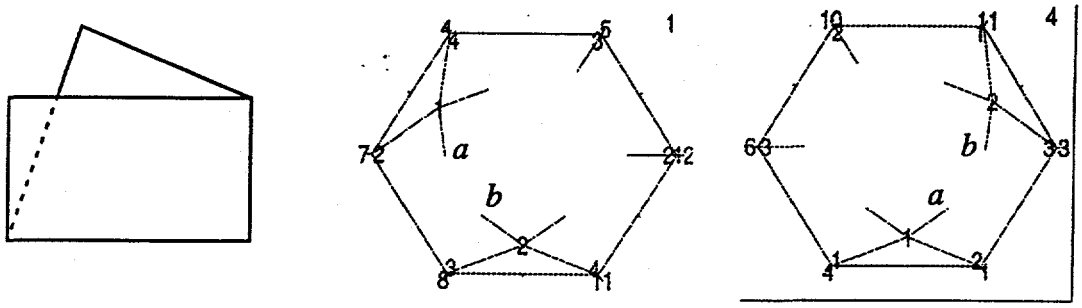
Figure 2. Final sheets for the double fold example.

(the "double fold" problem) and the whisker sheets of the fully woven all-hex mesh. For a description of these diagrams, see [1]. This problem is used to describe the primal construction algorithm later in this paper.

This paper is organized as follows. Section 2 discusses the algorithm now used to detect and resolve invalid connectivity cases during whisker weaving. This section also introduces new strategies for resolving the repeated repeated hex case. Section 3 describes the primal construction algorithm. Section 4 gives conclusions and discusses future work.

INVALID CONNECTIVITY DETECTION AND RESOLUTION

The creation of invalid hexahedral mesh connectivity has been observed in both the whisker weaving[3] and plastering[4] algorithms. The most common type of invalid connectivity is the case of two mesh faces sharing two edges; this is shown in Figure 3a.



(a) (b)

Figure 3. Invalid connectivity case of two mesh faces sharing two edges. a) Hex space representation; b) Mesh

This case is represented in STC data as a pair of STC edges which appear in two distinct STC 2-cells[3]. For example, in Figure 3b, the STC edges labelled *a* and *b* represent mesh faces which share two edges. This situation is referred to as a repeated STC edge pair, since the pair of STC edges representing two faces is repeated in two distinct STC 2-cells. This situation is also referred to as a repeated hex pair, since there is often a hex pair that is repeated in the same way (although this is not required); for example, in Figure 3b, hex elements 1 and 2 are in both STC 2-cells.

Various strategies exist for resolving invalid mesh connectivity in the whisker weaving algorithm, depending on where the invalid connectivity occurs (e.g. on or below the meshing boundary). However, the detection of these invalidities can be based entirely on repeated STC edges, as described above, since they all contain one or more face pairs which share two edges. There are 4 general types of invalidities currently observed in whisker weaving; their detection and resolution is summarized in Table 2.

Table 2. Four cases of invalid connectivity, and their detection and resolution techniques.

Type:	Detection:	Resolution:
Seam	Repeated STC edge pair (see Figure 3b)	Join 2 chords.
Knife	2 STC edges representing same face adjacent.	Join chord to itself; forms knife element in mesh.
Remove Hex Seam	Repeated STC edge pair, with one STC edge (one face) not on meshing front.	Remove all hexes after embedded STC edge on that chord; seam with other chord.

Table 2. Four cases of invalid connectivity, and their detection and resolution techniques.

Repeated Repeated Hex	Two STC edge pairs which share one STC edge, resolving one pair creates the other pair.	Ignore, hoping this invalidity will disappear.

The strategies for performing simple seams and for forming knife elements using STC data have been known for some time and are not reviewed here; for details, see [1] and [3]. The removed hex seam case can be envisioned as follows. In STC space, a repeated STC edge pair is created where one of the STC edges is not on the current meshing front. In hex space, this is analogous to a pair of faces being formed which share two edges, where one of the faces is not on the meshing front. Figure 5a shows this situation, where the

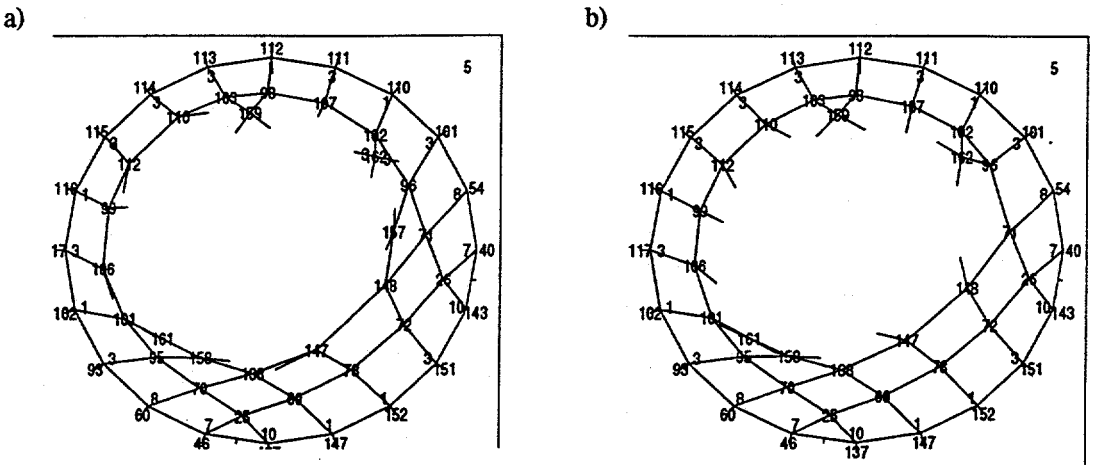


Figure 4. Removed hex seam case; a) before resolution; b) after resolution.

embedded repeated STC edge is between hexes 157 and 96 (the other sheet on which this chord appears is not shown). After removing hex 157, the chord beginning at face 101 can be joined with the blind chord originating at hex 162; the result is shown in Figure 5b. In effect, the removed hex seam operation is choosing to remove hexes from one chord so that a seam operation can take place. This has the effect of minimizing the number of hexes in a particular mesh, although this strategy is not guaranteed to be optimal.

The resolution scheme shown in Table 2 for the repeated repeated hex case is less than satisfying, and can result in invalid connectivity persisting in a mesh that is not part of the small set of invalid elements already allowed (these elements are the knife and doublet elements). Also, the resolution of invalid connectivity using the techniques listed in Table 2 was critical in moving the whisker weaving algorithm to closure for more complicated problems like the macaroni problem [1].

To review, the repeated repeated hex case appears as two pairs of repeated STC edges, where the pairs share an STC edge. Joining two of the STC edges resolves one pair, but the resolution of the other pair requires breaking the chord just joined and joining one of the pieces with the third STC edge. This arrangement in hex space, shown in Figure 6 ,

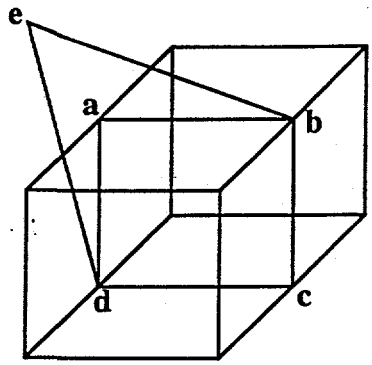


Figure 5. Repeated repeated hex case in hex space.

corresponds to two faces which share two edges, with one face (face **adeb**) on the meshing front and one face (face **abcd**) not on the front. This case is distinct from the removed hex seam case in that the chord going through face **abcd** is completed.

Instead of leaving this invalidity in place, this problem can be resolved by closing the face on the meshing front (by joining nodes **a** and **e** of face **adeb** in Figure 5). Since this face is part of a hex element on one side, this hex element becomes a knife element. Also, in general, the two sheets passing through this face are not the same. Since knife elements must by definition contain self-intersecting sheets[1], this results in the merging of the two sheets passing through the collapsed face. Right after the merge operation, the merged sheet has two face loops, which are represented as inner and outer loops on the sheet diagram. In the case of a repeated repeated hex case motivating the sheet merge, the only chord joining the inner and outer loops after the merge is the chord which terminates in a knife element.

An example of a repeated repeated hex case, before and after resolution by collapsing a face, is shown in Figure 6. Note that the knife element formed is represented in Figure 6c by the chord joining the inner and outer loops on the sheet diagram (labelled **c-c**). This example, from the macaroni problem, is subsequently closed by whisker weaving. The resulting mesh contains more knife elements and fewer doublets than were produced with the old resolution technique for the repeated repeated hex case. This technique for resolving repeated repeated hex cases will be investigated in greater detail to ascertain whether there are real benefits to performing this operation.

The pseudo code now used for detection and resolution of the various cases of connectivity checking is summarized in Figure 7.

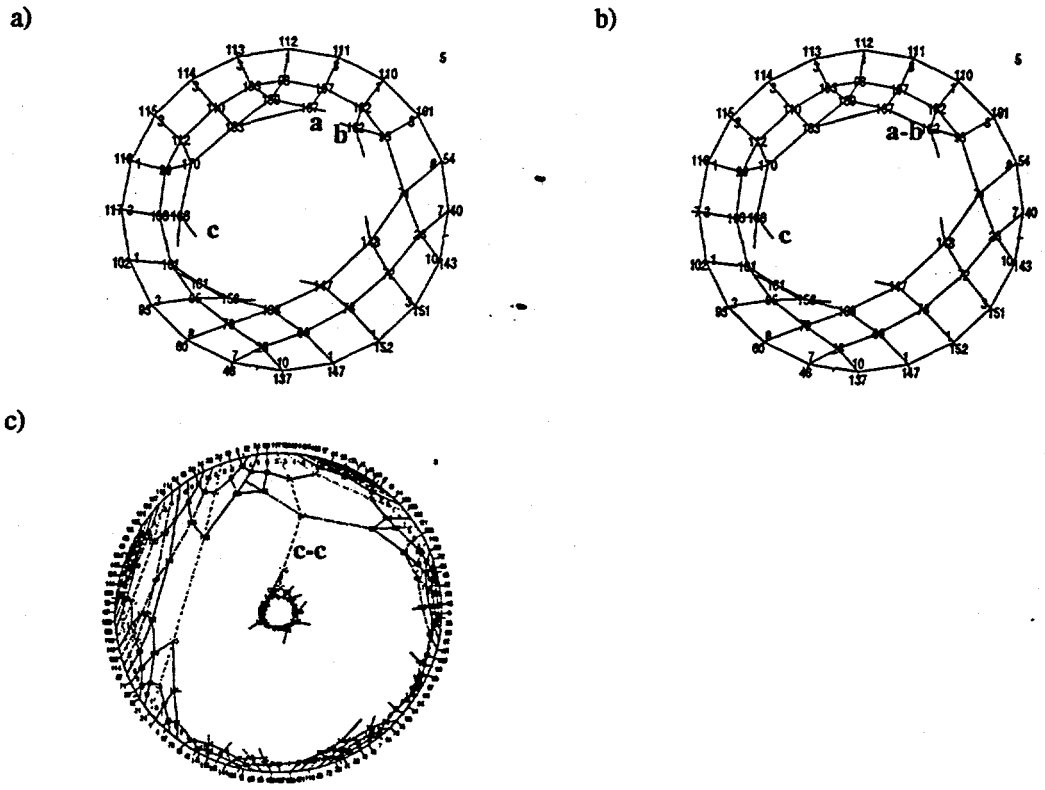


Figure 6. Example of a repeated repeated hex case and resolution by collapsing a face. a) Sheet diagram showing 3 STC edges (a, b, c) forming 2 pairs of repeated STC edges. b) Joining of 2 STC edges (a-b) to complete a chord. c) Resolution of third STC edge by merging sheets and joining with its other sheetchord (c-c).

- For each sheetchord on sheetchord list:
  - Check for repeated hex; if one exists:
    - If a simple join can be done, do it
    - Else if embedded repeated STC edge,
      - Remove hexes after repeated STC edge on the associated chord
      - If repeated repeated hex case, resolve by collapsing a face
      - Else append this sheetchord to the sheetchord list
    - Else remove one repeated STC edge by removing a hex

Figure 7. Pseudo code for detection and resolution of invalid connectivity in whisker weaing.

PRIMAL CONSTRUCTION ALGORITHM

The output of the whisker weaving algorithm is a specification of all-hex mesh connectivity in the form of STC data. In particular, connectivity information for hex elements, mesh faces and mesh edges is given (the dual counterparts to these entities are constructed in the STC data as part of the whisker weaving algorithm). However, the nodal connectivity is

not represented directly in the STC data. This information must be extracted from the STC data before the mesh primal, consisting of nodes and hex elements, can be constructed.

The primal construction algorithm is based on gathering all the edges and hexes that include a node, then allocating that node. A rough description of the algorithm is given in this section. The example in Figures 1 and 2 will be used to illustrate the steps of the algorithm. In the example, 2-cells are identified by a sequence of SheetChordPoint identifiers. For example, the "h1-f6-f5-h2" 2-cell on sheet 1 denotes the 2-cell bounded by hex 1, face 6, face 5, and hex 2 (see Figure 2).

1. Construct all the whisker cells.

In STC data, the whisker cells are unique; that is, they provide a one-to-one matching between STC entities (STC 2-cells) and mesh entities (mesh edges). This is in contrast to 3 STC vertices representing a hex, and to 2 STC edges representing a face. Since the 2-cells are not constructed explicitely during whisker weaving, they must be constructed here. During this step each hex is given a list of 2-cells on which it appears (which corresponds to the 12 edges contained by that hex).

2. Choose a first 2-cell (cell A).

Cell A corresponds to a mesh edge which has not yet been "wrapped" (i.e. it has a node which has not been wrapped). For example, in Figure 3, the h1-h2-h3-h4 2-cell is chosen.

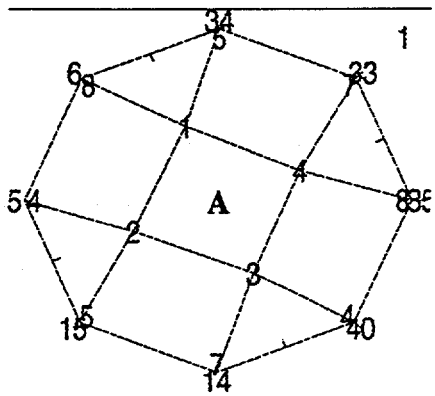


Figure 8. First 2-cell (A), hex (h1) and face (h1-h2) chosen.

3. Find a second cell (cell B) which shares a node with the cell A.

The 2-cells A and B correspond to two mesh edges which share a node. Cell B is found by choosing a hex in the first cell, choosing one of the mesh faces (STC edges) coming from that hex on that cell, and choosing one of the two other cells containing the other STC edge corresponding to that face.

For example, the first hex chosen in the 2-cell A is hex 1, represented by SheetChordPoint h1a. One of the STC edges in that cell is chosen arbitrarily, say the h1-h2 edge (using a

similar specification to the 2-cell specification). This STC edge represents a mesh face, which is represented by exactly one other STC edge on another sheet. The other STC edge is found using the whisker cell list that is kept for each whisker hex; this edge must not contain point h1a but must contain SheetChordPoints corresponding to hexes 1 and 2. Figure 4 shows the other STC edge (h1-h2), which is on sheet 5.

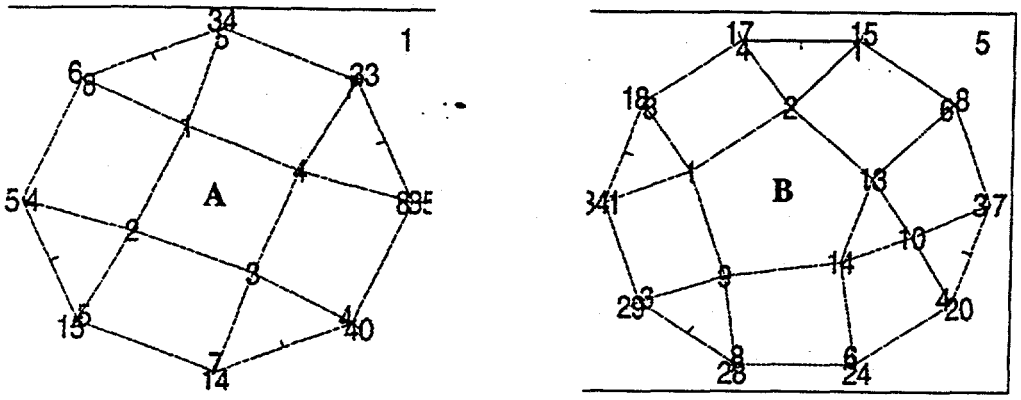


Figure 9. Second STC edge representing face h1-f6.

The second 2-cell is chosen from one of the two on either side of the second STC edge. In mesh space, choosing a second 2-cell corresponds to finding a second edge that is part of a given face and that shares a node with the first mesh edge chosen. The choice between 2 2-cells corresponds to choosing one of two opposite edges on a face which shares a node with one of the other edges on that face. In this example, the h1-h9-h14-h13-h2 2-cell is chosen (see Figure 4).

Cells A and B now define a node that has not yet been allocated, unless that node lies on the surface (more on that later). The cells are put on a new list of whisker cells. This list is used in the next step to keep track of all 2-cells (mesh edges) sharing the node being wrapped.

4. Wrap the node.

A node is "wrapped" by finding all the mesh edges and hexes which contain that node. This is done using a "gift-wrapping" algorithm, where a traversal from hex to hex is done, picking up all hexes and edges sharing that node along the way.

The starting point for the wrapping operation is three whisker cells (mesh edges) and at least one hex, all of which contain the wrapped node. Given two edges which share a node in an all-hex mesh, this defines a face. If that face is on the interior (i.e. if one or two of the edges is interior), there are exactly two hexes which share that face, otherwise one hex contains that face. These hexes can be derived from the whisker weaving data in a straightforward manner. By testing all combinations of edge pairs, adding new edges to the end of the list as they are found, all the hexes and edges around a given node can be found.

In our example, we have two 2-cells (A and B) on our list. Given these two 2-cells, a third 2-cell can be found which shares the shared node. This is done in the following manner. A hex common to both 2-cells is found; in this case, 2-cells A and B share hex 1. A common face (that is, a pair of STC edges representing the same face) can also be found using the procedure described in step 3. In this example, the h1-h2 face is chosen. The information we have now consists of a hex element, and a face and two edges in that element. The two edges share a node, which is contained in that face and hex. This is depicted in mesh space in Figure 5. The third 2-cell corresponds to the third edge in the

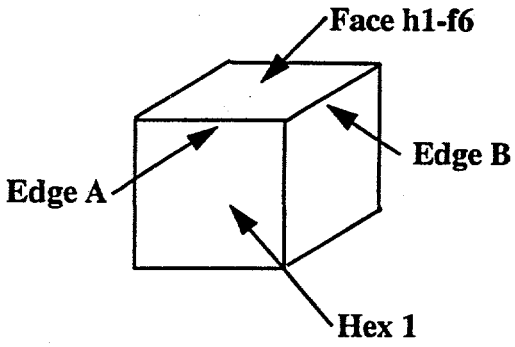


Figure 10. Mesh space entities represented by 2-cells A and B, hex 1 and face h1-f6.

chosen hex which shares the node being wrapped. Figure 5 shows that that third edge is shared by two faces, each of which shares an edge (edge A or edge B) with face h1-h2. The two faces are represented in STC space by the STC edges connected to hex 1 on cells A and B on the other side of edge h1-h2. So, from Figure 4, the two faces are h1-h4 on cell A and h1-h9 on cell B. Since these faces share an edge, they should be part of the same 2-cell somewhere else. Because this third edge is orthogonal to the other two (in a hex principal direction sense), the third 2-cell contains the third STC vertex representing hex 1. The third STC vertex for hex occurs on sheet 8, shown in Figure 6. There are four STC 2-cells which

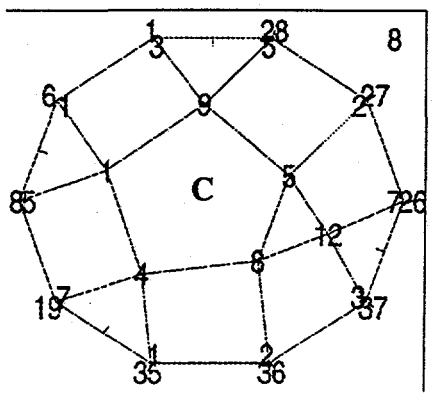


Figure 11. Sheet containing third cell (cell C).

contain hex 1; the third cell must contain faces h1-h4 and h1-h9. Thus, the third 2-cell is the h1-h4-h8-h5-h9 2-cell. If this third 2-cell does not yet appear on the list of 2-cells, it is added at this time.

If the common face is in the interior of the mesh, there is a second hex that shares that face. That hex is represented by the other hex on the chosen STC edge. In our example, the hex opposite hex 1 that shares face h1-h2 is obviously hex 2. Using the same procedure described in the previous paragraphs, a third 2-cell (edge) is found which shares a node with edges represented by 2-cells A and B and which is contained in hex 2. The reader can verify that this 2-cell is cell h2-h13-h17-h7-h3 (cell D) shown in Figure 7. If this 2-cell is

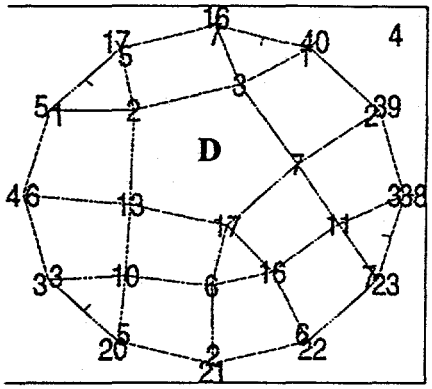


Figure 12. Sheet containing fourth cell (cell D).

not yet on the list of 2-cells, it is added. Also, if hex 2 is not on the hex list, it is added.

To review, given two STC 2-cells (mesh edges), a common STC edge (face) can be found, and at least one common STC vertex (hex), or two hexes if the face is on the interior, can be found. From the two hexes and the two edges, two additional edges can be found which share the node being wrapped.

To find all the edges and hexes sharing a given node, all permutations of 2-cell (edge) pairs on the list are tested. This is guaranteed to find all the hexes and edges sharing a node, because in a non-degenerate hexahedral mesh, you can traverse between any two hexes that share a node by travelling through faces which also share that node.

5. Allocate and distribute the node.

The cell list and hex list now contain all the edges and hexes which share a node, respectively. If any of the edges lies on the surface (i.e. both of its nodes are defined), the node in question has already been allocated. (An interior edge can have both of its nodes defined too, but since this algorithm finds all instances of edges connected to an interior node, you wouldn't get this edge after all of its interior nodes were defined.) If not, allocate a node. Put the node on all the connected edges, and add the node to each hex's node list. If the node was allocated, compute the position based on the connected whisker hex's

physical position (ok if that position hasn't yet been computed - the result will just be the origin).

6. Repeat steps 3-5 for the first cell so that both nodes for the edge defined.

7. Repeat steps 2-6 for all cells.

After this is done, all nodes have been allocated and defined.

8. Compute whisker hex node ordering.

The node ordering in a hexahedral element is significant since it is used to compute the jacobian and other metrics for the element. The ordering of nodes is specified by the Exodus database standard [3], and is shown in Figure 8. Node ordering is computed by

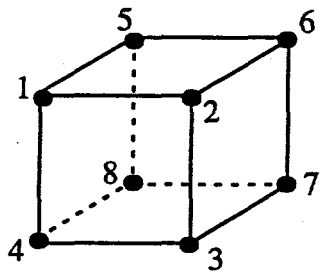


Figure 13. Hexahedral element node ordering, from Exodus.

finding 4 nodes which form a face and the other 4 nodes connected to the first 4 and which are not on the face. These nodes are then ordered according to the Exodus specification. These nodes are passed in to the constructor for a NodeHex, a class inside CUBIT which represents hexahedral elements consisting only of nodes.

9. Smooth the mesh.

The smoothing scheme selected for the volume being meshed is used. Emperical evidence shows that a length-weighted laplacian smoothing scheme tends to work best, although this has received limited testing.

Note that this step computes the physical positions of the nodes. In most cases, when the interior nodes were allocated, they were positioned at the origin. This initial position may be changed in the future to ensure that initially the nodes are inside the volume being meshed (this position should be available from the solid modeler). This method places demands on the smoothing algorithms available, but one could argue that these algorithms must be robust anyway.

10. Check the hexes for inversion.

The connectivity inside a hex element can be inverted such that the hex has negative volume, according to the Exodus definition of a hex. For example, in Figure 8, nodes 1 and 3 can be exchanged and the same can be done with nodes 5 and 7. The resulting hex element is identical to the valid hex in terms of connectivity, but will have a negative volume as computed by an analysis code. The node ordering for each hex is currently inferred by tracing chords in from the surface mesh, under the assumption that the surface mesh face ordering is correct.

11. Check the resulting mesh for incomplete edges (edges with less than two nodes defined).

EXAMPLES

This algorithm has been tested on the double_fold problem shown in Figure 1, and results in the mesh shown in Figure 9. Other meshes produced with this algorithm are shown in

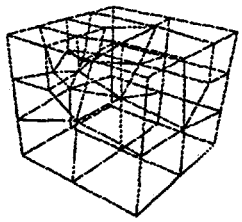


Figure 14. Mesh produced for the double_fold problem.

Figures 10. Although the resulting meshes haven't been run through an analysis, they do

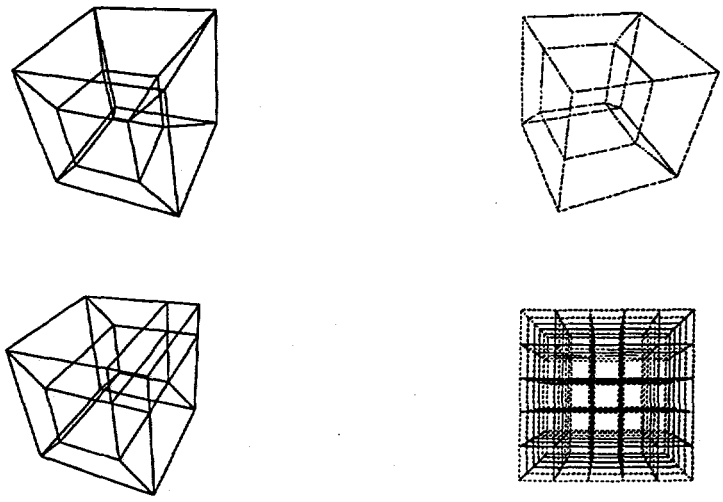


Figure 15. Meshes produced for the small hex-center hex (left), small corner hex (middle) and small self-intersection (right) problems.

have positive volumes, as computed by the NUMBERS program [4].

CONCLUSIONS

The detection of cases of invalid connectivity in whisker weaving can be based fully on repeated STC edges. One special connectivity case, the repeated repeated hex case, can be handled by collapsing a face, producing a knife and sometimes a self-intersecting sheet with multiple face loops. This resolution strategy will be investigated further to see if it holds any advantages over the current strategy of ignoring this case.

The whisker weaving algorithm is able to produce meshes for simple geometries which have non-trivial surface meshes. The algorithm described in this paper takes the connectivity information generated by the whisker weaving algorithm and produces real nodes and hexes. The resulting mesh contains elements which have positive volume and should be useable for finite element analysis.

REFERENCES

1. T. J. Tautges, T. D. Blacker, S. A. Mitchell, "The Wisker Weaving Algorithm: A Connectivity-Based Method for Constructing All–Hexahedral Finite Element Meshes", submitted to Int J Num Meth Eng.

2. P. Murdoch, S. S. Benzley, T. D. Blacker, S. A. Mitchell,"The Spatial Twist Continuum:A Connectivity-Based Method for Representing All–Hexahedral Finite Element Meshes", submitted to Int J Num Meth Eng.

3. T. J. Tautges, S. A. Mitchell, "The Wisker Weaving Algorithm for Constructing All–Hexahedral Finite Element Meshes", 3rd International Meshing Roundtable, Oct 24-25, 1994, Albuquerque, NM.

4. Jim Hipp, Randy Lober, "Plastering: Automated All-Hexahedral Mesh Generation Through Connectivity Resolution", 3rd International Meshing Roundtable, Oct 24-25, 1994, Albuquerque, NM.

5. Larry A. Schoof, Victor R. Yarberry,,"EXODUS II: A Finite Element Data Model", SAND92-2137, Sandia National Laboratories, Albuquerque, NM, September 1994.

6. Gregory D. Sjaardema,,"NUMBERS: A Collection of Utilities for Pre- and Post-processing Two- and Three-Dimensional EXODUS Finite Element Models", SAND88-0737, Sandia National Laboratories, Albuquerque, NM, March 1989.