Boundary Control For Automated Sweeping of Finite Element Meshes

Robert A. Kerr
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Civil and Environmental Engineering Commons

BYU ScholarsArchive Citation

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
BOUNDARY CONTROL FOR AUTOMATED SWEEPING OF FINITE ELEMENT MESHES

ROBERT A. KERR
BOUNDARY CONTROL FOR AUTOMATED SWEEPING
OF FINITE ELEMENT MESHES

by

Robert A. Kerr

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Civil and Environmental Engineering
Brigham Young University
December 1999
of a thesis submitted by

Robert A. Kerr

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

11/9/99
Date

11/10/99
Date

11/9/99
Date

Steven E. Benzley, Chair

Richard J. Balling

David W. Jensen
As chair of the candidate's graduate committee, I have read the thesis of Robert A. Kerr in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials, including figures, tables, and charts, are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Nov 9, 1997
Date

Steven E. Benzley,
Chair, Graduate Committee

Accepted for the Department

T. Leslie Youd
Chair, Department of Civil and Environmental Engineering

Accepted for the College

Douglas M. Chabries
Dean, College of Engineering and Technology
Finite element analysis depends greatly upon a high-quality mesh to be able to provide reasonably accurate answers to engineering problems. Models that need to be analyzed using finite element analysis are becoming increasingly more complex, and correspondingly harder to mesh with good quality. Skew is one quality metric which can cause problems with finite element analyses. This thesis explains how skew is calculated, then discusses two common sources of skew: multiply-linked surfaces with interval constraints, and biased edge meshes.

Two methods of lessening skew in surface meshes are then presented: the skew control algorithm, and the curve morphing algorithm. These algorithms are discussed in detail, with representative graphics. Examples which demonstrate the skew which arises from the above-mentioned sources are presented. These models are then subjected to the algorithms discussed in the thesis, and a comparison of the skew
measure for each example is presented. Finally, areas of possible future work are presented and the possible detrimental effects that the skew control algorithm can exert on the quality metrics of aspect ratio and mesh size gradation are discussed.
ACKNOWLEDGMENTS

How do I thank the many people who’ve helped in this work? First, I must give thanks to Dr. Benzley, advisor extraordinaire. I’ve been extremely grateful for the opportunity I’ve had to work for and with him, in my schooling and elsewhere. Next, I am grateful for my patrons at Sandia National Laboratories*. Dave White has been an invaluable guide as well as friend. Tim Tautges and Scott Mitchell have also provided me with needed guidance and input in the work I’ve done. My friends at BYU, Jason Shepherd and especially Steve Jankovich have supported me and helped me with design decisions, daily drudgery, and delightful discourses. My parents certainly deserve mention for supporting me in my extended schooling, and their never-failing love. Dan McRae deserves mention for the rubber-band idea for curve morphing.

Most importantly I must give thanks for the three most important people in my life. I’m eternally grateful for my Father in Heaven, for the inspiration He’s given me and the help He’s provided me. My wife Keri has been enthusiastic and supportive of me, and her attitude, and that of my darling son Michael, make me feel like I’m on top of the world. Thank you all.

*This work was partly funded by Sandia National Laboratories, operated for the U.S. DOE under contract No. DE-AL04-94AL8500. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. DOE.
List of Figures

Figure 1  Rectangular solid demonstrating mapped mesh on front face. ........ 3
Figure 2  Surface with mapped mesh demonstrating a logical corner placed
          somewhere other than on an actual corner. .......................... 4
Figure 3  Surface with convex regions meshed with a mapping algorithm. .... 5
Figure 4  Beginning surface before subdivision. ............................... 6
Figure 5  One possible subdivision of the surface. ............................. 6
Figure 6  Result of submapping the surface. .................................. 7
Figure 7  Volume ready for a swept mesh. ..................................... 8
Figure 8  Sweepable volume with source surface meshed. ....................... 9
Figure 9  Volume showing final swept mesh. ................................... 9
Figure 10 Multiply-connected set of surfaces which could lead to skew problems.
               ................................................................. 11
Figure 11 Quadrilateral mesh element showing representative angle ‘A’. ..... 13
Figure 12 Multiply-connected surfaces. .......................................... 14
Figure 13 Multiple surfaces showing interval settings on right-hand side. ..... 15
Figure 14 Interval propagation across surfaces. ................................ 15
Figure 15 Final interval propagation across surface 1. ......................... 16
Figure 16 Final mesh on surface 1, showing a large value of skew. .......... 17
Figure 17 Simple surface showing biased mesh on right-hand side. .......... 17
Figure 18 Simple surface showing regular mesh on side opposite biased mesh. 18
Figure 19 Simple surface with resulting skew problems. ....................... 18
Figure 20 Surfaces showing a mesh with controlled skew. ..................... 20
Figure 21 Surface showing the four vertex types. ............................. 23
Figure 22 Surfaces showing which end is the smallest. Notice that vertices are
          marked by dark triangles. ........................................... 23
Figure 23 Surfaces after splitting curve into two curves and inserting a new pseudo edge. Note the new virtual vertex. .................................................. 24
Figure 24 Surfaces showing loop about to be subdivided. ........................................ 26
Figure 25 Surfaces showing final subdivision of loops. ........................................ 26
Figure 26 Surfaces showing final subdivisions with representative interval groups labeled. ................................................................. 27
Figure 27 Surfaces after cleanup of skew control entities. .................................. 28
Figure 28 Model with non-planar front surface. ............................................... 29
Figure 29 Non-planar surface showing I-J classification. ....................................... 30
Figure 30 Non-planar surface showing the I-positions of the I vertices. .............. 31
Figure 31 Non-planar surface showing propagation of a vertex to all affected curves. ................................................................. 32
Figure 32 Non-planar surface showing end result of vertex propagation. .......... 33
Figure 33 Surface with non-manifold curve. ...................................................... 34
Figure 34 Model with potential skew problems. .................................................. 36
Figure 35 Hooked surface before meshing. .......................................................... 37
Figure 36 Hooked surface after submapped meshing without skew control. ....... 37
Figure 37 Hooked surface after processing by the skew control algorithm. ........... 38
Figure 38 Hooked surface after meshing with skew control. ................................ 39
Figure 39 Skewed mesh generated on linked surfaces. ....................................... 40
Figure 40 Less-skewed mesh generated with skew control algorithm on linked surfaces. ................................................................. 40
Figure 41 Reproduction of simple surface showing biased mesh on right-hand side. ................................................................. 42
Figure 42 Simple surface with skewed mesh caused by a biased side mesh. ........ 46
Figure 43 Simple surface with non-skewed mesh aided by the curve morphing algorithm. ................................................................. 47
Figure 44 Stepped surface with biased mesh on top and left curves. .................... 48
Figure 45 Stepped surface with skewed mesh. ..................................................... 49
Figure 46 Stepped surface with less-skewed mesh due to the curve morphing algorithm. ................................................................. 49
Figure 47  Surface showing biased mesh on top curve. ...................... 51
Figure 48  Surface showing loss of bias on top curve after skew control algorithm. ................................................................. 51
Figure 49  Model with top surface consisting of more than one loop. ........ 52
Figure 50  Hooked surface before skew control processing. ..................... 53
Figure 51  Hooked surface after processing by the skew control algorithm. .... 53
Figure 52  Surface which needs narrow-neck loop separation for correct calculation of vertex placement. ............................... 54
List of Tables

Table 1  Comparison of skew for hooked surface with and without skew control.  39
Table 2  Comparison of skew for middle surface of linked surfaces with and without skew control. ................................................................. 41
Table 3  Comparison of skew for simple surface with and without curve morphing. ................................................................. 47
Table 4  Comparison of skew for stepped surface with and without curve morphing. ................................................................. 49
1 - Introduction

It has been said that “Engineering is the art of molding materials we do not fully understand into shapes we cannot fully analyze, and preventing the public from realizing the full extent of our ignorance.” [9] While most engineering professionals would take issue with the last of those statements, our conscience demands that we cede the point that we do not fully understand the materials we work with, and many times cannot fully analyze the structures built with those materials.

However, through the use of finite element analysis (FEA), it has become much easier to perform analyses of structures that, while not being perfect analyses, are at least close enough to reality to allow the confident construction of many engineering works. The art of finite element analysis has been around for over 30 years, and in that time has progressed into a well-understood and relatively easily-used discipline. Many programs exist that will analyze a given solid model subject to different loading cases, and return answers that are within a few percentage points of the “accurate” solution. One would think that because of how far the art of FEA has progressed, engineers would not have any problems studying and solving all the problems they face. Unfortunately, this utopian ideal is hindered by the necessary prerequisite step of generating a finite element mesh which will serve to sufficiently describe the model for the FEA program. In fact, most of the time spent in performing an analysis is spent in creating the model and meshing it. [2] One of the
goals of current meshing research is to automate as much as possible the generation of the finite element mesh for a given model. If the time-to-mesh could be decreased substantially, the engineer would see a corresponding decrease in time-to-analyze, and would be able to solve a larger number of problems in the time allotted.

Of course, as the state of engineering progresses, the models that need to be meshed become more and more complicated, thus requiring meshing algorithms of ever-increasing capabilities. Many different algorithms have been developed in an effort to attain this goal of automatic meshing. Two of the most widely-used surface-meshing algorithms are mapping and its descendant, submapping. [13,15] These two algorithms are used extensively because they generate a structured mesh rather easily. Structured, in a meshing environment, is defined as a surface mesh in which each node is connected to four other nodes. [7] Volume mapping, a three-dimensional analogue to surface mapping, and sweeping, which will be defined later, are two volume meshing algorithms that depend on there being mapped meshes on most or all of the surfaces of a model. [17] Unfortunately, there are many cases where the geometry of the solid model is such that generating a mapped mesh on a part is very problematic. This thesis will briefly explain mapping, submapping and sweeping, then will discuss certain problems encountered in generating mapped meshes. Methods for lessening or eliminating some of these problems before the meshing takes place will be demonstrated, and examples will be presented which will show the effectiveness of the algorithms developed herein.
2 - Existing Technology

2.1 - Mapping

Parameter space mapping on surfaces, hereafter referred to as “mapping”, is an algorithm for creating a structured mesh on a given surface. Structured, as defined previously, is a surface mesh wherein each node is connected to four other nodes. [7] This generates a regular grid of quadrilateral mesh elements over a surface. Figure 1 shows a rectangular prism with a mapped mesh on the front face. Mapping is a highly useful technique, because it generates a mesh with usually well-formed elements close to the boundary of the surface, and there are no irregular nodes (nodes which are connected to more or fewer than four other nodes). [20] Through the technique of
mapping, the analyst is usually able to generate a mesh that is:

- Boundary Sensitive: the well-shaped elements around the boundary yield good results for the analysis.
- Orientation Insensitive: the algorithm will generate the same type of elements in the same configuration in spite of changes in orientation of the underlying surface.
- Regularly Shaped: the resulting mesh is structured, which yields well-shaped elements. [5]

In order to generate a mapped mesh, four logical corners are picked from the surface. These four corners will determine the corners of the mapped mesh. The logical corners need not necessarily correspond to actual corners of the surface under consideration, although typically better results are obtained if that correspondence is present. The sides connecting those four logical corners are then designated as the sides of the mapped mesh. Figure 2 demonstrates a surface in which logical corners do not correspond with actual corners.

![Figure 2](image)

**Figure 2** Surface with mapped mesh demonstrating a logical corner placed somewhere other than on an actual corner.
One of the necessary prerequisites for a successful mapped mesh is that opposite sides of the mapped surface have the same number of mesh edges, or intervals. Therefore, appropriate interval assignments on the surface are vital to the success of the mapping algorithm. Much work has been done on the problem of automatically determining interval assignments to generate acceptable meshes. [14, 17] In many instances, the use of an interval linear program allows the setting of intervals for an entire set of surfaces and volumes, [18] allowing a feasible solution for the meshing problem.

2.2 - Submapping

Mapping, while being a powerful method for meshing surfaces, is only applicable to fully-convex regions. A region is fully convex if and only if all of the interior angles are less than $180^\circ$. This guarantees that there are not interior corners. Mapping also can generate poor quality meshes when the region to be mapped does not lend itself to easy classification of logical corners. Figure 3 is an example of a

![Figure 3 Surface with convex regions meshed with a mapping algorithm.](image-url)
plane surface on which mapping is unable to generate acceptable elements because of concave sections, which interfere with ability of the mapping algorithm to pick corners correctly. The quality of this mesh is completely unacceptable because of the great numbers of mesh elements which overlap. Surfaces like this obviously need to be meshed using a different algorithm, such as submapping. Submapping is a type of surface decomposition algorithm, which has as its goal the division of a surface into mappable regions. The algorithm accomplishes this task by searching for logical points of division for a surface, such as interior corners. When these subdivision points are found, the submapping algorithm divides the surface into mappable regions, and then proceeds to map the subregions. [19] Figures 4 and 5 demonstrate the process of subdivision used in the submapping algorithm. Figure 6 shows the mesh

![Figure 4](beginning-surface-before-subdivision.png)  ![Figure 5](one-possible-subdivision-of-surface.png)

Figure 4: Beginning surface before subdivision.  
Figure 5: One possible subdivision of the surface.
resulting from the use of the submapping algorithm. As can be seen, in spite of a few skewed elements, this is a much better result than that given by the mapping algorithm shown in Figure 3.

![Figure 6 Result of submapping the surface.](image)

2.3 - Sweeping

Mapping and submapping are powerful tools for generating surface and volume meshes. This thesis refers to them predominantly in their role in surface meshing, but they are easily extendible to three-dimensional volumes. One of the most used volume meshing schemes is sweeping. Studies have noted that over 50% of real-world meshing problems involve sweeping. [13, 15] Thus, any improvements in the sweeping algorithm will serve to increase the positive results of the sweeping algorithm.
The sweeping algorithm is a method for generating a structured or semi-structured mesh in a volume. To be sweepable, a volume must have topologically similar source and target surfaces connected by mappable or submappable side surfaces. These side surfaces are known as linking surfaces. These types of volumes are known as two and one-half-dimensional solids, because they can be meshed by projecting the mesh of the source surface layer by layer through the volume, until the target surface is encountered. [17] Figure 7 shows a volume, ready for meshing. It fits the criteria for a sweepable volume because the source and target surfaces, i.e. the ends, are topologically similar, and all sides of the volume are mappable. In Figure 8 the source surface has been meshed, and in Figure 9 the mesh has been completed. Because of the requirement that the linking surfaces be mappable or submappable, the sweeping algorithm is greatly dependent on the success of the mapping and submapping algorithms.
Figure 8  Sweepable volume with source surface meshed.

Figure 9  Volume showing final swept mesh.
3 - Statement of Problem

3.1 - Interval Matching

To generate a structured mesh, it is vital that the intervals on the boundaries be assigned correctly. As was stated above, an interval linear program can be used to aid in the task of assigning a compatible set of intervals for the entire volume. This linear program modifies the intervals on the surfaces and their boundaries, which will hereafter be referred to as curves. If the user has assigned a certain number of intervals to a curve, then that assignment is respected, and the linear program attempts to accommodate those intervals in the mesh of the rest of the volume. It is incumbent upon the user to not overconstrain the volume, thus forcing the linear program to return an impossible solution to the interval problem. Usually the user will assign a general set of intervals to the volume, with perhaps some specific settings on important features. The linear program will then usually be able to obtain an acceptable solution. Unfortunately, the linear program is not designed to solve for a set of intervals that will provide the most regular mesh, but rather to solve for a set of intervals that will least disturb the current general settings. The solutions provided by the linear program will in certain cases lead to meshes that, while technically acceptable, have severe quality problems. This is very visible in cases such as that shown in Figure 10, in which the model consists of a group of connected parts. Interval assignments on the outer sides of the model will propagate across the whole
model, producing some rather poor quality meshes. This interval propagation problem happens to a much greater extent with surfaces that are to be meshed using a mapped or submapped mesh than it does to surfaces which would have an unstructured mesh. An unstructured mesh is one in which the nodes may be connected to any number of elements. Thus, unstructured meshes do not have the requirement of equal intervals on opposite sides of the surfaces.

3.2 - Mesh Quality

It is well-known that elements with high quality yield better results than do elements with low quality. Quality is a measure of how much the mesh element varies from the ideal. In quadrilateral meshing, the ideal is a square. That is, the elements of the mesh should have 90-degree angles, and have sides that are equal in length. As mesh elements vary from this ideal, the errors in the finite element analysis increase.
Analysts need a way to quantify and categorize the quality of a mesh. [1,8] This need has given rise to measurements of quality such as:

- aspect ratio, which indicates the ratio of length to breadth of the element.
- taper, which indicates the tendency of one side of an element to be shorter than the opposite side.
- Jacobian, which is the minimum pointwise volume of the local map at the four corners and center of the quadrilateral. [12]
- skew, which provides a measure of the variation of the sides of the element from the optimal right-angle configuration. [6, 16]

As stated, these quality metrics measure different departures from the ideal. As a quality measure moves away from the ideal, the analysis results also depart from the ideal result. For example, elements with high aspect ratios will perform more stiffly in the analysis than will elements with lower aspect ratios, and elements with a negative Jacobian measure will cause severe problems with the analysis, and will yield answers that are highly inaccurate, if indeed results are obtainable.

Skew is another quality issue that can cause problems with the analysis. Although it is always laudable to have elements with no quality problems, that is almost never possible. The search for good quality elements then becomes an optimization problem, with the analyst's time being weighed against the desire for high quality, and sometimes even the quality metrics become competing objectives. In such a case, the analyst will try to lessen the overall quality problems, but will also need to decide which quality problem is more important than the others. Usually,
skew is a quality problem that analysts will try to minimize. This desire to lessen skew is sometimes thwarted by the tendency of the mapping and sweeping algorithms to propagate interval assignments throughout a large model, and can also be affected by irregular intervals on the sides of surfaces.

3.3 - The Skew Problem

3.3.1 - Definition of Skew

Skew is defined in the literature as the maximum absolute value of the cosine of the angle between edges at the center of the quadrilateral. In other words, it is the cosine of the angles formed by the two lines which pass through the midpoints of the sides of the quadrilateral. [16] In Figure 11, a quadrilateral element is shown, with one of the angles labeled as angle A. The absolute value of the cosine of this angle would be the measure of skew for this element. The other angles formed by these lines

![Figure 11 Quadrilateral mesh element showing representative angle 'A'.](image)
are complementary angles, and yield the same value for the cosine, except for sign. As can be seen, the value of skew ranges between 0.0 and 1.0, with the optimal value being 0. Negative numbers are avoided by applying the absolute value operation.

3.3.2 - Skew from Interval Propagation

Mapped meshes, by their nature, depend on propagation of interval assignments. [14, 19] Skew usually is not a problem in small or simple models, however in a multiply-connected model, as was shown in Figure 10, interval assignments propagate throughout the model. This propagation of interval assignments can lead to skew problems. In Figure 12, interval settings on the ends of surfaces 2, 3, 4 and 5 will propagate across those surfaces and affect the sides of surface 1. Surface 1 will then have to have its right-hand side edges set to the same number of intervals as is on the left-hand side. If one of the right-hand side curves has a comparatively

![Figure 12 Multiply-connected surfaces.](image)
high interval count, the mesh on surface 1 could become greatly skewed. Figure 13 shows, for example, a possible set of intervals for the right-hand side of surfaces 2, 3 and 4, and the left-hand side of surface 5. These intervals will propagate across the surfaces, as shown in Figure 14. After propagating across the surfaces, the intervals on the common edges between surface 1 and the other three surfaces are firmly set. A
mesh then needs to be generated on surface 1. Most mesh generating software assigns equally-spaced intervals as the default. Therefore, the left-hand side interval assignment on surface 1 would look like that shown in Figure 15. The final meshing

![Figure 15 Final interval propagation across surface 1.](image)

of the surfaces would then be done, yielding a final mesh on surface 1 as is shown in Figure 16. As can be seen, the mesh on this surface has a large degree of skew.

What is required to reduce skew is to develop methods that will transfer the interval assignments from one side of the surface to the other in a manner that will preserve the relative interval spacings on different sections of the curves. One possible solution is to set intervals on opposite sides to be the equal. Unfortunately, this method does not take into account the possibility that these opposite edges may be of different lengths, which would still engender skew. Thus, manually setting intervals on all the curves yields only limited success, engendering the same mesh shown in Figure 16.
As can be seen, manual intervention, besides taking much longer, is not better than what would be done automatically.

3.3.3 - Skew from Irregular Mesh Spacing

Another facet to the skew problem is that of irregular mesh spacing along the edges. If a curve is meshed with a bias, as shown in Figure 17, it will have a greater
concentration of mesh nodes at one end than the opposite end. Because of the tendency of mesh generators to equally space mesh nodes, the default mesh on the left-hand side will be similar to that shown in Figure 18. This situation will generate skew problems on the surface, as shown in Figure 19.

These circumstances are what has prompted the development of the algorithms which will be discussed in the rest of this thesis.

Figure 18 Simple surface showing regular mesh on side opposite biased mesh.

Figure 19 Simple surface with resulting skew problems.
4 - The Skew Control Algorithm

4.1 - Overview

The purpose of this algorithm is to manage the interval settings on surfaces that will probably exhibit unacceptable skew. The skew control algorithm is intended for use previous to meshing, although the user may assign approximate sizes and intervals before its invocation. If the user has specified this information, the algorithm will respect those settings. Certain of the calculations in the algorithm depend on the mesh size on a surface. Therefore, if the user has not set a mesh size, one will be calculated automatically using the autosizing algorithm, which will be explained later.

Given a surface or set of surfaces such as that discussed above, the skew control algorithm developed here will partition the edges of the surface and set up a system of matching edges across a surface or multiple surfaces. Once this is done, the interval count on these matched edges is set to be equal, so that a created mesh has little or no skew. Instead of the skewed mesh that was shown in Figure 16, the final mesh would result as depicted in Figure 20.
4.2 - Details of the Skew Control Algorithm

The skew control algorithm has the following 7 steps:

1. Approximate the affected surfaces with pseudo geometry.
2. Create a loop of edges around the base surface.
3. Find the smallest projection on the surface.
4. Separate this small feature from the rest of the surface using a pseudo edge.
5. Continue separating small sections until all loops consist of only four edges.
6. Step through loops setting up interval assignments for opposite edges.
7. Clean up pseudo geometry.

These steps will be explained one-by-one with representative illustrations.
4.2.1 - Approximate the affected surfaces.

The skew control algorithm depends on an approximation of the surfaces to be meshed. These surfaces will be referred to as the base surfaces. The set of surfaces originally shown in Figure 12 will be used as an example. The algorithm uses pseudo geometry known as skew control entities for this approximation. The curves and vertices which make up the real surface are used as templates to create skew control edges and skew control vertices. These skew control entities are the basis for almost all the work in the algorithm. Skew control entities only need to possess a minimum of information about the entities which define them. Each skew control edge knows what vertices define it, and each skew control vertex knows its position in three-space and its type. Vertex types will be defined later. Because of this sparsity of information, the memory overhead in using these pseudo-entities is small.

4.2.2 - Create a loop of edges around the base surfaces.

The skew control edges and vertices that have been created from the base surfaces are placed in lists which maintain their order. Each base surface is approximated by one list. This list holds pointers to the edges and vertices which define the surface and form a loop. These loops, which are the base for this algorithm, define the base surfaces throughout the rest of the algorithm. This example set of base surfaces would translate into five loops, one for each of the surfaces. Some of the curves and vertices will be in more than one list, if that curve is shared between two surfaces.
4.2.3 - Find the smallest projection in the surface.

The skew control algorithm implements a type of “blocking” subdivision. “Blocking” refers to the process of dividing up the surface into blocks, or four-sided areas. This blocking algorithm starts with the smallest areas, filling them with blocks, then expands to the larger ones. The starting step is to find the smallest end. For some geometries, this is not an easy location to find, and much depends on the definition of “end”. An “end” is defined as a set of edges that are bounded by two vertices which are known as End_Types. As can be seen, the type of the skew control vertices has a great impact on the definition of an end. There are four types of vertices, based on the angle of the edges which share the vertex:

1. End_Type, with an angle close to $90^\circ$.
2. Side_Type, with an angle close to $180^\circ$.
3. Corner_Type, with an angle close to $270^\circ$.
4. Reversal_Type, with an angle close to $360^\circ$. [19]

Figure 21 shows all four types. A skew control vertex is assigned a type based on the type of the underlying real vertex. If there is no underlying vertex, the type is computed based on the angle of the connected skew control edges. The base surface’s loop is searched for the “end” that is shortest, and that end is used for the next operation. As shown in Figure 22, the far right-hand end should be picked first.
Figure 21  Surface showing the four vertex types.

Figure 22  Surfaces showing which end is the smallest. Notice that vertices are marked by dark triangles.
4.2.4 - Separate the smallest projection from the rest of the loop.

When the shortest end has been found, the algorithm then compares the two edges that form the sides of the block. At this point there are two possibilities: either one side will be longer than the other, or the two will be the same length (within a tolerance). If the first case applies, the algorithm splits the longer of the two sides so that the resulting two sides are equal. Of course, this splitting can happen within a range of lengths, depending on local geometry and the desired tolerance. If the pseudo edge to be split has an underlying geometry edge, that geometry edge is split too, and a virtual vertex is inserted. After the split of the pseudo edge, this first case becomes identical to the second case. When two vertices are at an equal distance from the end, a new pseudo edge is created between the two vertices. The edges and vertices comprising the block being replaced are then removed from the old loop, and the new edge is inserted in their place. A new loop is created using the newly

Figure 23 Surfaces after splitting curve into two curves and inserting a new pseudo edge. Note the new virtual vertex.
independent edges and vertices and the new edge, as shown in Figure 23. Where once there was one loop defining the surface, now there are two. An important feature of this new loop is that it consists of only four edges.

4.2.5 - Continue separating small sections until all loops consist of only four edges.

Now there are six loops to consider, instead of the previous five. Each of these loops is checked to see if they consist of more than four edges. If one is found, the previous step is repeated and the next smallest projection is separated from that loop and put into its own loop. Now, because of the possibility of an edge being in two loops at the same time, it is vital that the loops stay current regarding which edges belong to them. It would cause a severe problem if an edge were split but the loops containing the old edge still had pointers to that obsolete edge. Pointers to these owning loops are of necessity another piece of data contained in each skew control edge. Each edge keeps a list of loops it belongs to. With this information, when an edge is subdivided, as in step 4, the algorithm can access this list of loops and update each one of them so that they contain the correct information. A side effect of this is that a loop that previously only had four edges might end up having more through the splitting of one of its included edges. This necessitates traversing the set of loops multiple times until all the loops have only four edges. An example of this increase in edges is seen in Figure 24, which shows the right-side curve of the left-hand loop (shown in bold in the figure) being subdivided. The next time that loop is examined, it will need to be processed again, because of the increase in number of edges. This is
shown in Figure 25 where all the needed subdivisions have taken place. In this figure, the inverted triangles mark all the vertices in the new surface, both the original vertices, and those virtual vertices where underlying curves have been subdivided.
4.2.6 - Step through loops setting up interval assignments for opposite edges.

Once all of the loops satisfy the requirement of having only four edges, the setting of constraints can be done. Each loop is examined to find an edge that has an underlying geometry edge (or owner). This edge is then marked and the loop is searched for the edge opposite the marked one. If the opposite edge has an underlying owner, then those two edges are set to have equal intervals. If this opposite edge does not have an owner, the algorithm asks the edge for its list of owning loops and these loops are searched in the same manner. In this way, geometry edges that are at either end of a series of loops are found and then set to have equal intervals. Figure 26 shows the final state of the example surfaces. A representative sampling of the sets of edges which have the same interval setting have been labeled. Because each edge can

Figure 26 Surfaces showing final subdivisions with representative interval groups labeled.
belong to any number of loops, the interval assignment on one edge may propagate to many different edges on many different surfaces, but this is handled transparently. The algorithm does not need to differentiate between surfaces, because its working environment consists exclusively of loops and owners.

4.2.7 - Clean up pseudo geometry.

After the interval assignment is complete, the skew control entities can be safely deleted. The only changes done to the underlying geometry are that where edges had to be split, there are now two edges and a vertex (Figure 27), and interval assignments have been made to all the curves in the base surfaces.

![Figure 27 Surfaces after cleanup of skew control entities.](image-url)
4.3 - The Skew Control Algorithm–Special Cases

4.3.1 - Non-planar Surfaces

Non-planar surfaces present a particular problem for the skew control algorithm. Because the underlying geometry is not flat, the approximations done in creating skew control entities can lead to errors in the creation of new vertices. Because of this, non-planar surfaces are treated in a different manner. The algorithm creates the pseudo-curves as is done in the normal algorithm, but instead of creating the four-sided loops that are used to subdivide regular planar surfaces, the algorithm propagates vertices to all possible curves. For example, the model shown in Figure 28 has a non-planar front surface which will serve as the example for the description of the methods used in the skew control algorithm on non-planar surfaces.

![Figure 28 Model with non-planar front surface.](image-url)
There are four steps to the non-planar skew control algorithm:

1. Each curve on a surface is classified as either a Positive I, Negative I, Positive J or Negative J curve. This classification is shown in Figure 29.

2. The algorithm traverses the surface’s loop, processing the vertices on the loop. The first vertex is marked, and its location in I-J space is calculated. This vertex is known as the source vertex, and the curve immediately before it in the loop is known as the source curve.

3. Each curve in the loop is then examined to determine whether the source vertex lies in between the start and end of the curve. This comparison is only done if the classification of the source curve is of the same general classification as the current curve, i.e. if the source
curve and current curve are both I or both J. This is because a source vertex on a -I curve cannot influence the +J or -J curves, but it can affect both a +I or -I curve.

4. When a current curve is found which will be affected by the source vertex, that curve is split, and the underlying owner is split too. This adds to the number of vertices in the loop, but this new vertex does not add to the search. If a source vertex affects a certain set of curves, then any vertices created in this step will only affect the original set of curves, and will only affect those in the same places as the original source vertex. Therefore these newly created vertices do not need to be propagated to the rest of the surface.

Figures 30 and 31 demonstrate the algorithm. Figure 30 shows the surface with the I
positions labeled for all the I vertices. Vertex A is located at position 5 in I-space, so all curves that cross the 5 position will have a virtual vertex created at the 5 location on that curve. Figure 31 demonstrates the surface with vertex A propagated to all affected curves.

![Figure 31](image)

**Figure 31** Non-planar surface showing propagation of a vertex to all affected curves.

...affected curves. The vertices on the J curves will be propagated in the same manner. Figure 32 shows the non-planar surface after all vertex propagation is finished. This design for the handling of non-planar surfaces will succeed in locating vertices where they are needed, but it can also lead to a greater number of virtual vertices being created than is strictly necessary. Although this is a slight drawback, it is expected that most non-planar surfaces will not be so complicated that they will cause an excess of vertex creation.
4.3.2 - Surfaces with non-manifold Curves

If a loop of edges defines a surface, then any edge which appears in that loop more than once is known as a non-manifold curve. Surfaces in which a non-manifold curve appears, such as that shown in Figure 33, have a potential for problems. When a non-manifold curve is split, special care needs to be taken so that every occurrence of the non-manifold curve is replaced by the two new curves. If one of these occurrences is missed, then the algorithm will yield unanticipated results when that curve is encountered again. The pseudo-geometry which represents that curve will not have been kept up-to-date with the actual underlying geometry, and the algorithm has a great probability of failing.
4.4 - The Autosizing Algorithm

The autosizing algorithm, described in Appendix A, is an automated method of determining an appropriate size for a mesh on a given model. Frequently the analyst must devote time to the question of what size of mesh to assign to a given model. The autosizing algorithm was written to give the analyst a rapid estimate for an acceptable mesh size. The autosizing algorithm has 3 steps:

1. The algorithm gathers information about the lengths of the curve in the model.
2. The algorithm applies a scaling equation to this information based on a default scaling factor.
3. The result from this equation is then used as the mesh size for the model.
The default scaling factor is a number between zero and 11, which allows the analyst to affect the outcome of the scaling equation without needing to know the sizes of the parts of the model. The analyst is thus able to apply the autosizing algorithm to a model and be reasonably certain of having a decent size of mesh. The autosizing algorithm is also useful in giving the analyst information as to the general sizes of the model, so that a mesh size can be determined manually.
5 - The Skew Control Algorithm: Examples

The skew control algorithm shows a lot of potential in models which have many surfaces which affect one another. Descriptions and results for two of these models follow.

5.1 - Multiply-impacted “Hooked” surface

The model shown in Figure 34 has a multitude of multiply-connected surfaces that can cause skew. For example, the lower right-hand surface from this model.

Figure 34 Model with potential skew problems.
shown in Figure 35, is rather complicated, with many other surfaces touching it. If
meshed without skew control, using a submapping algorithm, the mesh would be
created as shown in Figure 36. As can be seen, there is significant skew evident on

Figure 35  Hooked surface before meshing.

Figure 36  Hooked surface after submapped meshing without skew control.
the surface. However, if the skew control algorithm is applied to this surface, the resulting mesh demonstrates much less skew. The skew control algorithm processes this surface, inserting virtual vertices in places it deems appropriate. Since the user has already requested a certain mesh size on the model, this is taken into account in the final mesh, by transferring interval assignments onto newly created curves. The surface in question now has some new vertices, in addition to those previously present. This is shown in Figure 37. The mesh that results from this changed surface is shown in Figure 38. As this figure shows, the mesh is noticeably much less skewed. Use of the skew control algorithm does result in an increase in the number of edges and vertices in the model, but this has not proven to be a major concern. The goal of the algorithm, that of reducing skew on the model, has been met. The mesh of the “hooked” surface
is of good quality, and the measure of skew has decreased dramatically, as is shown in Table 1.

![Figure 38](image)

**Figure 38** Hooked surface after meshing with skew control.

<table>
<thead>
<tr>
<th>Hooked Surface</th>
<th>Non-controlled</th>
<th>Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Skew</td>
<td>0.7081</td>
<td>0.1184</td>
</tr>
<tr>
<td>Minimum Skew</td>
<td>0.02594</td>
<td>0.00</td>
</tr>
<tr>
<td>Average Skew</td>
<td>0.2384</td>
<td>0.01638</td>
</tr>
</tbody>
</table>

**Table 1** Comparison of skew for hooked surface with and without skew control.

5.2 - Multiply-connected surfaces

Another example of skew control is shown in Figures 39 and 40. This is a model similar to the example which was used to demonstrate the steps of the algorithm, although there are some minor differences in the model and hence in the
final mesh. Figure 39 shows the mesh that is generated without having applied the skew control algorithm. The surfaces were meshed by a submapping algorithm, after the intervals on most of the sides were set. Although this thesis concentrates on the surface in the middle, it is also obvious that there are skew problems in other surfaces.

Figure 40 demonstrates the model with a skew controlled mesh. As can be seen, there

Figure 39 Skewed mesh generated on linked surfaces.

Figure 40 Less-skewed mesh generated with skew control algorithm on linked surfaces.
is very little skew on the middle surface, although the other surfaces, because of their inherently skewed nature, still demonstrate some skew. Table 2 compares the values of skew resulting on the middle surface both with and without skew control. These values are comparable to the values shown for the previous example.

**Table 2** Comparison of skew for middle surface of linked surfaces with and without skew control.

<table>
<thead>
<tr>
<th>Middle Surface</th>
<th>Non-controlled</th>
<th>Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Skew</td>
<td>0.6548</td>
<td>2.601 x 10^{-14}</td>
</tr>
<tr>
<td>Minimum Skew</td>
<td>0.07927</td>
<td>6.661 x 10^{-17}</td>
</tr>
<tr>
<td>Average Skew</td>
<td>0.4354</td>
<td>6.898 x 10^{-15}</td>
</tr>
</tbody>
</table>

6 - The Curve Morphing Algorithm

6.1 - Overview

Another source of skew for mapped and submapped surfaces is irregular node placement on the edges of the surface. As was mentioned previously, if one side of a surface has a non-regular distribution, perhaps generated by a biasing algorithm, as was shown previously and reproduced here as Figure 41, then the opposite side of the face needs to have the same type of distribution, or the mesh will be skewed. The solution to this is to use an algorithm such as curve morphing. Morphing is the process whereby a mesh is transferred from one entity to another, preserving the connectivity of the mesh and as much of the original shape as is possible. [10] Curve

Figure 41 Reproduction of simple surface showing biased mesh on right-hand side.
morphing is the subset of the morphing algorithms which applies to the copying of a mesh, usually non-regular, from one curve or set of curves onto another curve or set of curves. This algorithm can be productively used in the above-mentioned cases in order to lessen skew.

6.2 - Details of the Curve Morphing Algorithm

The curve morphing algorithm is a simple method which consists of three steps:

1. Setup of the scaling information.
2. Transferring the nodes.
3. Arranging nodes around vertices.

These steps will now be explained in more detail.

6.2.1 - Setup of the scaling information

The algorithm begins by summing the lengths of the source and the target edges. This simple arithmetic operation enables the algorithm to generate the necessary scale. This scale will be used to convert the source mesh length to a suitable target mesh length. This fraction, which is mainly for use in situations in which the total length of the source edges is different than that of the target edges, is stored for use in the next step.
6.2.2 - Transferring the nodes.

Once the scaling information is obtained, the algorithm starts at one end of the list of source mesh edges. The first mesh edge is found, and the length of it is measured. This length is then multiplied by the scaling factor to yield a target length. The algorithm then moves to the beginning point of the target curves, and from that point measures along the curve the target length. A node is then inserted at the end of the target length, and the process repeats until all the nodes on the set of source curves have been measured.

6.2.3 - Arranging nodes around vertices.

A difficult aspect of curve morphing is ensuring that the nodes being placed on the target curves fall correctly on the vertices. If there is only one source curve and one target curve, this is not a problem, because the source nodes will automatically fall on the correct target locations. But, if there are multiple target curves, special care must be taken to ensure that each vertex receives a node. Because of the scaling factor and the possibly-differing lengths for the source and target curves, it is highly likely that a node will not land serendipitously on a target vertex. The algorithm uses the following criteria to determine what to do in this case:

- If the next node would be placed “close” to the vertex then that node is assigned to the target vertex, and the source length is adjusted to compensate for the difference between the ideal placement and the actual placement. “Close” in this case means within a certain tolerance
value. Good results have been obtained with a tolerance value of half of the current mesh length.

- If the next node falls outside of the tolerance value defined above, it is considered to be too far away from the vertex. In this case the previous node is moved to the vertex, and the source length is adjusted accordingly.
7 - The Curve Morphing Algorithm: Examples

7.1 - Simple Surface

The surface used previously serves as a first example of curve morphing. As was shown previously, without utilizing the curve morphing algorithm, the mesh on the simple square surface is skewed (shown in Figure 42). As is evident, the normal mapping algorithm regularly spaces the nodes on the side of the surface opposite the biased mesh. This skews the surface mesh. This type of surface is not one that the skew control algorithm can handle. The skew on this surface is not caused by vertices and curves, but rather the actual mesh distribution. As a rule, geometry considerations

Figure 42 Simple surface with skewed mesh caused by a biased side mesh.
should be handled by the skew control algorithm, while mesh distribution
considerations should be handled by the curve morphing algorithm. The skew control
algorithm would not change the skew on this surface, but the curve morphing
algorithm was written to handle just these types of cases. Figure 43 demonstrates how

![Figure 43](image)

Figure 43 Simple surface with non-skewed mesh aided by the curve morphing algorithm.

the surface would be meshed if the curve morphing algorithm were employed to copy
the biasing from the right-hand side to the left-hand side. Obviously there is much
less skew in this example. Table 3 compares the values for skew for both cases.

<table>
<thead>
<tr>
<th>Simple Surface</th>
<th>Non-curve morphed</th>
<th>Curve morphed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Skew</td>
<td>0.3544</td>
<td>7.133 x 10^{-15}</td>
</tr>
<tr>
<td>Minimum Skew</td>
<td>0.04373</td>
<td>0.0000</td>
</tr>
<tr>
<td>Average Skew</td>
<td>0.2348</td>
<td>1.071 x 10^{-15}</td>
</tr>
</tbody>
</table>
7.2 - Stepped Surface

The surface shown in Figure 44 has a biased nodal distribution along the top and left curves. As can be seen, the nodes along the top of the surface will not fall exactly on the ends of the curves comprising the bottom of the surface. The submapping algorithm will attempt to regularly distribute the same number of nodes along the bottom of the surface as exists along the top surface, without regard to the relative spacing of those nodes. The same will happen with the nodes on the left and right sides of the surface. This leads to the mesh generated by the submapping algorithm shown in Figure 45. Obviously the biasing of the top and left curves has caused skew problems with the mesh in this example. However, if the user specifies that the top curve is to be copied or morphed to the bottom set of curves, the resulting mesh appears as in Figure 46. The curve morphing algorithm has not solved all the
skew problems with this surface, but those that still exist are due to the submapping algorithm. Table 4 compares the values of skew for the stepped surface. As can be seen, the skew has decreased considerably, in spite of the submapping problems.

**Table 4** Comparison of skew for stepped surface with and without curve morphing.

<table>
<thead>
<tr>
<th>Simple Surface</th>
<th>Non-curve morphed</th>
<th>Curve morphed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Skew</td>
<td>0.8008</td>
<td>0.3519</td>
</tr>
<tr>
<td>Minimum Skew</td>
<td>0.00</td>
<td>2.368⁻¹⁶</td>
</tr>
<tr>
<td>Average Skew</td>
<td>0.5362</td>
<td>0.05098</td>
</tr>
</tbody>
</table>
8 - Further Work

Although the algorithms developed herein are fairly robust, and have been seen to provide good results in many cases, there are some issues that need to be addressed to make these more widely applicable tools. Many of the remaining areas of debate exist because of particular design decisions that need to be made. Some of these areas are:

- The skew control algorithm only works on surfaces that are submappable, i.e. of blocky, roughly four-sided sections or subsections. This is a limitation that will probably be modified in the future. There has been some discussion as to the need to implement a skew control algorithm for surfaces that would need to be meshed with an unstructured mesh.

- Biased intervals settings are not propagated correctly onto split curves when using the skew control algorithm. As Figures 47 and 48 show, even though the user has specified a biased mesh on the top curve, after the skew control algorithm subdivides the curve, the biasing has disappeared. While general interval settings are handled automatically, the biasing is not preserved. Ideally, the curve morphing algorithm would be called in this situation, but this has not been studied yet.
Figure 47  Surface showing biased mesh on top curve.

Figure 48  Surface showing loss of bias on top curve after skew control algorithm.
Surfaces with more than one loop, i.e. surfaces with holes in them, an example of which is shown in Figure 49, cannot be processed by the skew control algorithm. These types of surfaces need to be modified before the algorithm can be applied. The addition of an automatic surface cracking algorithm, which will create a curve connecting an inner loop to an outer loop, will convert two-loop surfaces into one-loop surfaces. This type of surface will then become a one-loop case with a non-manifold edge, which the skew control algorithm can process.

![Figure 49 Model with top surface consisting of more than one loop.](image)

Virtual vertex creation, while enabling the algorithm to easily set up the constraint equations for the final model, adds extraneous vertices to the model. As can be seen by comparing Figures 50 and 51, the skew control algorithm sometimes inserts a multitude of vertices that are not strictly necessary for the definition of the solid model. Although this creation of vertices has not proven to affect the model excessively, it
would be beneficial to develop a different way of setting up the constraint equations. It is hoped that through use of the curve morphing algorithm the skew control algorithm will be able to correctly transfer the mesh from one side of a surface to another without the use of virtual vertices.
Other design decisions, such as tolerancing of curve intersections, and where to place target vertices on the target curve need to be made in order to enhance the algorithm's ability to handle complicated geometry. In cases that involve curved sides, or sides with many connected edges, (see Figure 52 for example) the algorithm needs to have a better apparatus for deciding final location of the virtual vertices. Efforts are currently under way to implement a method of narrow-neck loop separation. This will better allow the algorithm to place the vertices so as to ensure a higher-quality mesh.

Figure 52 Surface which needs narrow-neck loop separation for correct calculation of vertex placement.

In certain models, the configuration of the surfaces allows the vertices on a given surface to recursively impact that same surface. This can sometimes lead to a loop in which the skew control algorithm places an extraordinary number of vertices on the model, until the limits of the
tolerance are reached. In order to be robust, the algorithm needs a method to detect cases like this, so that these models can be processed without problem.

- The skew control algorithm, while reducing skew, affects other quality measures. In certain models the skew control algorithm leads to the creation of elements with a very large aspect ratio. In addition, the mesh on a skew controlled surface may also exhibit a large size difference between adjoining mesh elements. As a consequence, the analyst will need to understand the requirements for the mesh before making the decision to use these algorithms or some other skew control method.
9 - Conclusion

In spite of the areas mentioned above that need to be studied further, the skew control algorithm has been proven to provide a good method for lessening skew on multiple linked surfaces. Using a combination of the skew control algorithm and the curve morphing algorithm can greatly decrease the skew on meshes generated by mapping and submapping algorithms. This combination of algorithms has a great potential for increasing the success rate of volumes which can be meshed by sweeping. Because sweeping has proven to be of such widespread use to analysts, this improved success rate should enable the faster meshing of many different geometries, thereby increasing the productivity of the analysts.

The new approach taken in this thesis, that of progressive subdivision of surfaces as a prior step to setting the intervals on sections of geometric curves to be equal, shows promise as a robust algorithm that will be able to lessen the time-to-mesh for complicated geometries, especially those that can be meshed by sweeping. Although this algorithm has not proven successful on self-impacting surfaces and although work stills needs to be done for the case of surfaces which require unstructured meshes, the skew control algorithm promises to provide a worthwhile tool in the analysts' arsenal. However, care must be taken when using the skew control algorithm to ensure that the resulting mesh has acceptable aspect ratio and size.
gradation, since the algorithm presented here can have a detrimental effect on those quality metrics.

The curve morphing algorithm, which in addition to allowing the skew control algorithm to better preserve the biasing of the mesh on the curves and eventually avoid the creation of virtual vertices, also gives the analyst a useful tool for lessening skew on individual surfaces. These two algorithms combined provide a very widely applicable and robust tool, which will make large, complicated models much easier to mesh satisfactorily.
References


17 Shepherd, J., Interval Matching and Control for Hexahedral Mesh Generation of Swept Volumes, Published Master’s Thesis at Brigham Young University, 1999.


19 White, D., Automatic, Quadrilateral and Hexahedral Meshing of Pseudo-Cartesian Geometries Using Virtual Subdivision, Published Master’s Thesis at Brigham Young University, 1996.

Appendix A: The Autosizing Algorithm

The autosizing algorithm is explained in this appendix. The information presented is taken from a presentation given by the author at Sandia National Laboratories. [11]

**Auto-sizing**

**What is it?**

- Automated Abstract Size
  - Setting/Specification Tool
    - Automated--User called, but work performed by Cubit
    - Abstract--Unitless relative size range
    - Size Setting--Default size of 6 for all geometries, then scaled to actual part size
    - Specification--Allow user to interact with sizing tool if desired
Auto-sizing
Why is it needed?

- Moving towards a more-automatic meshing of simple geometries.
  - Aimed at beginning users, or users with simple geometries.
  - Gives a "good" first estimate of a decent size for the mesh on a part.

Auto-sizing
How is it done?

- Find minimum and maximum curve lengths in model.
- Apply a weighted scaling equation involving the unitless size and the min/max curve lengths
- Optionally add feature-sizing to the edges
Auto-sizing

Scaling equations

Two possible scaling equations:

- UL
- IL
- IS
- LS

L = Longest Curve
S = Shortest Curve

Default of 6

Optional power function

Auto-sizing

Future work

- Feature-sizing speed-up
- Skew Control Problem
- Automatic invocation if needed
Auto-sizing

Future work

- Feature-sizing speed-up
- Skew Control Problem