Neighborhood-Based Element Sizing Control for Finite Element Surface Meshing

Steven J. Owen and Sunil Saigal
Department of Civil and Environmental Engineering, Carnegie Mellon University
and
ANSYS Inc.
275 Technology Drive, Cannonsburg, PA, USA
steve.owen@ansys.com

Abstract

A method is presented for controlling element sizes on the interior of areas during surface meshing. A Delaunay background mesh is defined over which a neighborhood based interpolation scheme is used to interpolate element sizes. A brief description of natural neighbor interpolation is included and compared to linear interpolation. Two specific applications are presented that utilize the sizing function, namely boundary layer meshing and surface curvature refinement. For these applications, criteria used for insertion of additional interior vertices into the background mesh to control element sizing is discussed.

Introduction

In recent years CAD software has become a popular means for generating geometry for the finite element method. The surfaces and volumes described by CAD can come in a variety of forms including parametric surfaces such as NURBS or tessellated geometry consisting of triangle facets. Because CAD models will often have regions of high curvature or very tiny features in the same model with larger or flatter features, automated mesh generation resulting in high quality, well-graded finite elements can be a difficult task. In order to adequately describe all features of the model without generating huge numbers of elements, large transitions in element sizes can be required.

To control element sizes during the mesh generation process, an element sizing function can be defined. When meshing, this function may take into account surface features, proximity to other surfaces, surface curvature as well as physical properties such as boundary layers, surface loads or error norms in determining local element sizes.

While a wide variety of geometric and physical properties may be used in determining element sizes, the question remains as to how these can be combined to define a single smooth sizing function over the domain of the surface. This paper discusses a neighborhood-based interpolation method that can take into account any of the above features while maintaining a smoothly varying sizing function. It will also look at two specific surface meshing problems that can be handled using this approach, namely surface curvature and boundary layer meshing.

There are a great many aspects that can affect the final quality of a surface mesh. This paper focuses on only one of these, that of describing the background sizing function. The definition of the sizing function is defined as a pre-process to the actual surface meshing algorithm. During the meshing process, the sizing function is periodically evaluated providing information to control local element sizes. Although

relatively insignificant in total CPU time compared to the entire surface meshing process, the sizing function can ultimately influence the final quality and grading of the elements, perhaps more than any other phenomenon.

Background Mesh

The proposed method involves first, the definition of a Delaunay background mesh. Other current literature (Canann;1997), (Lohner;1996) also utilizes a background mesh in defining the sizing function. The typical case is to perform a Delaunay tessellation of the boundary nodes. The Delaunay background mesh is defined in the parameter space of the surface. For best results the paramaterization of the surface should be *well behaved* over the surface. This implies that the mapping from 3D space to 2D parametric space, can be described with a constant scaling function. Since, in general, CAD geometry cannot be expected to behave in this manner, a *warped* parametric space may be necessary to describe the parametric domain. Although a detailed description of the warped parametric space is beyond the scope of this paper, it should be noted that a reasonable mapping from 3D space to 2D parametric space is assumed

While the background mesh is usually defined in the parametric space of the surface, the actual meshing algorithm may be in either 2D parametric space or in 3D space. The background mesh is also independent of the actual meshing algorithm used. For this study, the advancing front method is used, but there is no reason it cannot be applied equally well to a Delaunay method. The only connection between mesher and sizing function is through a simple inquiry interface: $d_x = f(x,y)$, that is, given a local coordinate, (x,y) in the parameter space of the surface, return the target element size, d_x.

Figure 1 shows a simple Delaunay tessellation of the boundary nodes of a two dimensional object. Also included in the tesselation are four bounding corner nodes, included in order to facilitate the Delaunay algorithm. Element sizes are usually assigned to the vertices of the background tessellation based on the edge lengths of the boundary segments. Since it is necessary to locate a point in the Delaunay background mesh, a process in itself which can be somewhat time consuming, it is advantageous to maintain a background mesh as sparse as possible, while still maintaining the important sizing details of the model.

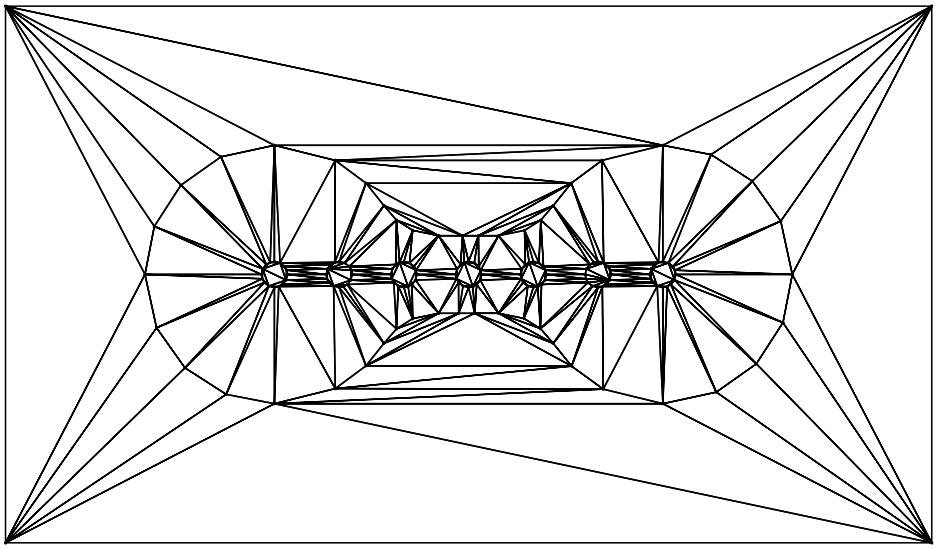

Figure 1 Delaunay tesselation of boundary vertices used for background mesh

The method used for interpolation can greatly effect the quality of the resulting mesh. Although simple linear interpolation is frequently used, some weaknesses have been noted with this method. An improved method, known as *natural neighbor* interpolation is used to define the sizing function.

Linear Interpolation

With linear interpolation, the local element size d_x at location $\mathbf{P_x}$, is defined as:

$$d_x = \sum_{i=0}^{2} w_i d_i \qquad [1]$$

where d_i are the element sizes at each of the three vertices of the triangle in the Delaunay background mesh containing $\mathbf{P_x}$, and w_i are the barycentric coordinates of $\mathbf{P_x}$ within the same triangle; for example:

$$w_0 = \frac{\begin{vmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}}, w_1 = \frac{\begin{vmatrix} x_0 & x & x_2 \\ y_0 & y & y_2 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}}, w_2 = \frac{\begin{vmatrix} x_0 & x_1 & x \\ y_0 & y_1 & y \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}} \qquad [2]$$

In some cases, linear interpolation provides an adequate description of the element sizing function. Poor results can arise when the triangles in the background Delaunay mesh become poorly shaped. Since in most cases, only the boundary nodes are tessellated, long, skinny triangles are typical similar to those shown in Figure 1. As a result, abrupt changes in element size are common resulting in less than desirable element quality and transitions.

Natural Neighbor Interpolation

Because of the limitations of linear interpolation, an alternate method was sought. Natural Neighbor interpolation was first introduced by Sibson (1981) and later further developed by Owen (1992) and Watson (1994). Even though its usage has been commonly for interpolation of scattered data generated from geotechnical applications (Jones;1995), its advantages are also applicable for element sizing. This method provides a smooth interpolation even with poorly shaped triangles in the background mesh. The element size d_x is defined in a similar manner to linear interpolation:

$$d_x = \sum_{i=0}^{n-1} w_i d_i \qquad [3]$$

where n is now the number of neighbor vertices. The distinguishing features of natural neighbor interpolation include the manner in which the neighbor vertices are selected and the computation of the weight, w_i at each of the vertices.

The n vertices included in the interpolant are defined as all vertices belonging to triangles whose circumcircle contain $\mathbf{P_x}$. This criteria is exactly that used in the well-known Delaunay triangulation

algorithm described by Watson (1981). Since the process of computing the neighboring triangles is an integral part of the Delaunay triangulation procedure, this information can not only be used to insert a new vertex into the mesh, but to compute the interpolated element size at the same vertex.

The weights, w_i at each of the neighboring vertices are defined using what Sibson (1981) first described as *local coordinates*. Local coordinates are often thought of as generalized barycentric coordinates. While barycentric coordinates can be defined as describing space with respect to three points in $\mathbf{R}^2$, local coordinates define space with respect to an arbitrary number of points. The local coordinate, w_i is defined as follows:

$$w_i = \frac{\kappa(\pi_i)}{\kappa(\pi)} \qquad [4]$$

where $\kappa(\pi)$ is the area of the Voronoi polygon, π defined after $\mathbf{P_x}$ is inserted into the domain and $\kappa(\pi_i)$ is the difference in areas of the Voronoi polygon π_i at vertex *i* before and after $\mathbf{P_x}$ is inserted. The Voronoi polygon associated with vertex *i* is that space in $\mathbf{R}^2$ closer to vertex *i* than any other vertex in the mesh. More precisely, the Voronoi polygon π_i associated with vertex $\mathbf{P_i}$ is defined as:

$$\pi_i = \left\{ x \in \mathbf{R}^2 : \left| x - P_i \right| < \left| x - P_j \right|, j = 1, \ldots N, j \neq i \right\} \qquad [5]$$

where *N* is the total number of vertices in the mesh. Figure 2 shows a graphical representation of local coordinates. In this example, the interpolant is $\mathbf{P_x}$. Solid lines represent the Voronoi polygons for each vertex in the domain after temporarily inserting $\mathbf{P_x}$ into the domain. The dashed lines represent the Voronoi polygons before insertion. In this case, vertices 1, 4, 5, 6 and 9 are selected as neighbors, and used to weight the interpolation. The area, $\kappa(\pi)$ is the Voronoi polygon defined by $\mathbf{P_x}$ and $\kappa(\pi_i)$ are the difference in Voronoi areas of polygons before and after $\mathbf{P_x}$ is inserted for each neighbor vertex. From this example it can be seen that the sum of $\kappa(\pi_i)$ will be $\kappa(\pi)$, hence, from equation 4, the sum of weights, w_i, will be one.

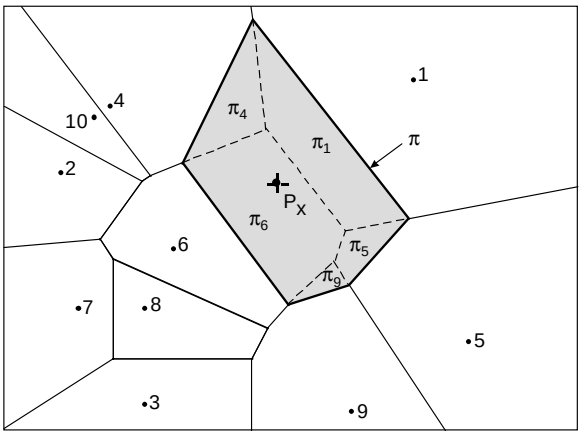

Figure 2. Voronoi polygons used in the definition of natural neighbor *local coordinates*.

A detailed description of the calculation of the local coordinates is provided by the author (1992) and Watson (1994). The author describes a simple method for computing the areas of Voronoi polygons based

on triangle circumcenters. Watson describes a more efficient method, which avoids the explicit formulation of the Voronoi polgons.

Applications

In many cases, it is sufficient to use only the boundary nodes of the area as input to describe the sizing function. If the boundary divisions have been adequately defined and there is very little surface curvature within the area, the resulting sizing function can produce very high quality elements. The sizing may not be sufficient when internal surface features, not described by line divisions are required, or if physical phenomenon requiring smaller element sizes close to boundaries is required. In these cases, additional internal nodes may be inserted into the background mesh to more precisely define the element sizing function.

Boundary Layers

Boundary layers are frequently required in computational fluid dynamics or electromagnetic problems. Several layers of small, uniformly sized elements are required immediately adjacent to a boundary. In order to avoid huge numbers of elements in the simulation, the element sizes should transition from the boundary layers to a much larger element size on the interior of the mesh. In order to capture the physics of the simulation, element size transitions from the boundary to the interior can be as much as 100 or 1000 to 1. Many methods have been proposed for meshing surfaces with boundary layers (Clements;1997), (Pirzadeh;1993). Inserting additional sizing vertices into the background mesh appears to be a promising solution for controlling the element size on meshes requiring boundary layers.

For boundary layer meshing, some user input is required. For each line defined as a boundary, at least three values must be specified:

1. α = element size at boundary
2. β = thickness of boundary layer
3. χ = thickness of transition layer

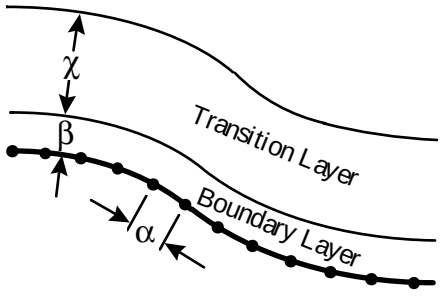

Figure 3 Boundary and transition layers adjacent to model boundary.

Alternatively, the user may prefer to enter an exact number of elements through the boundary layer, instead of β, or a growth ratio indicating the maximum aspect ratio allowable in the transition layer instead of χ. In either case, the user input is first converted into the required α, β and χ distances.

For every node on the boundary, up to two new vertices can be inserted into the background mesh. As indicated in Figure 4, points $\mathbf{P_c}$, and $\mathbf{P_t}$ are generated on a line normal to the boundary at boundary vertex $\mathbf{P_b}$. To avoid geometric evaluations, the normal vector, $\mathbf{N_b}$ to the line is defined as the bisector of the two adjacent edges to $\mathbf{P_b}$. Therefore $\mathbf{P_c}$ and $\mathbf{P_t}$ are defined as:

$$\mathbf{P}_c = \mathbf{P}_b + \left\|\mathbf{N}_b\right\| \cdot \beta\,, \qquad \mathbf{P}_t = \mathbf{P}_b + \left\|\mathbf{N}_b\right\| \cdot \left(\beta + \chi\right) \tag{6}$$

$$\text{where } \left\|\mathbf{N}_b\right\| = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \left\|\mathbf{P}_{b1} - \mathbf{P}_{b0}\right\| \tag{7}$$

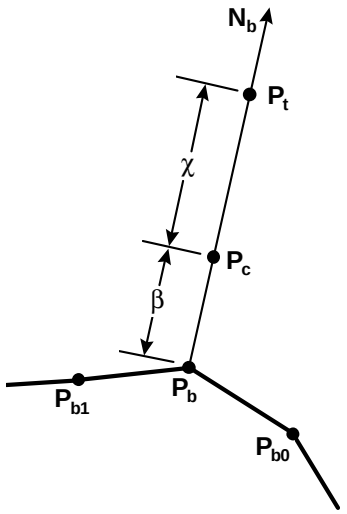

Figure 4 Placement of sizing vertices at the boundary and transition layers.

Once $\mathbf{P_c}$ and $\mathbf{P_t}$ have been located, the element size at these vertices are defined. The element size, d_c at $\mathbf{P_c}$ is simply the user supplied α. d_c can also be defined as the average length of the edges immediately adjacent to $\mathbf{P_b}$. The element size, d_t at $\mathbf{P_t}$ is defined from a user supplied internal element size, d_{global}. In the absence of a user supplied d_{global}, d_t can be defined as the average length of all boundary intervals not defined as boundary layers.

Inserting all vertices, $\mathbf{P_c}$, and $\mathbf{P_t}$ into the background mesh is in most cases far more than is needed to adequately describe the element sizing function. For most cases it is sufficient to insert new vertices at intervals no closer than d_{global}. For very large transitions where α is very small with respect to d_{global}, it may be necessary to insert $\mathbf{P_c}$ at the boundary layer at closer intervals.

As an additional measure to limit the number of vertices in the background mesh, an interpolation is first done at the proposed location of the new vertex. In this case the interpolation has relatively little overhead since the triangles needed for natural neighbor interpolation are the same as those used for a Watson (1982) type node insertion procedure. The proposed vertex is only inserted if the interpolated element size at the new vertex is different by more than a predefined percentage, ε, from the new element size, d_i at the vertex. For this application, ε of 5% was used.

One problem which can arise from this procedure occurs when the boundary layers or transition layers of opposing boundaries intersect. Vertices with widely varying element size definitions can end up being placed close together resulting in very poor element quality. To ensure this does not occur, $\mathbf{P_t}$ cannot be within a radius of $\beta+\chi$ from any other boundary edge. A local distance check is made to ensure this does not occur. In the event $\mathbf{P_t}$ is closer to a nearby edge than $\beta+\chi$, the distance χ is iteratively cut back until this requirement is satisfied. If this occurs, a new equivalent d_t must also be defined. The new d_t is linearly interpolated between the old d_t and the size d_c at $\mathbf{P_c}$.

Figure 5 shows the background mesh generated for a contrived model where the change in element size is 1:200. An equivalent β thickness was computed from a user input of five elements through the thickness of the boundary layer. The χ thickness was derived from user input from the maximum growth ratio of 2:1. The resulting contours generated from natural neighbor interpolation of the element size are also displayed on Figure 5. Figure 6 shows the resulting triangles generated using an advancing front triangle meshing algorithm which utilizes the background size information. Figure 7 is a closeup view of the boundary layers of Figure 6. Note the overlapping boundary layers and change in element sizes between layers.

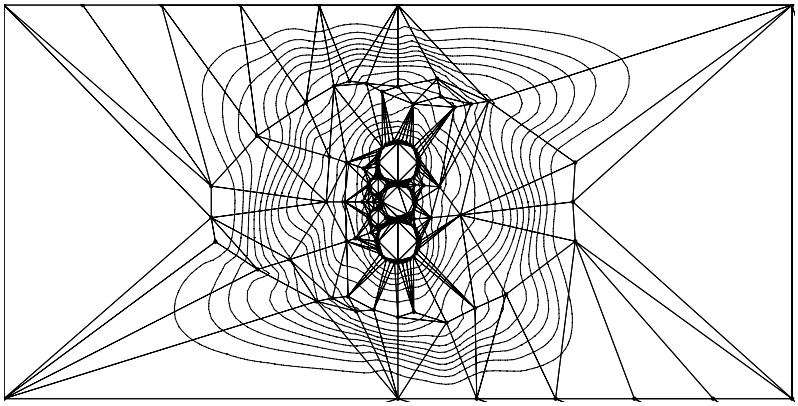

Figure 5. Background mesh used for boundary layer mesh. Countours generated from natural neighbor interpolation are over-layed

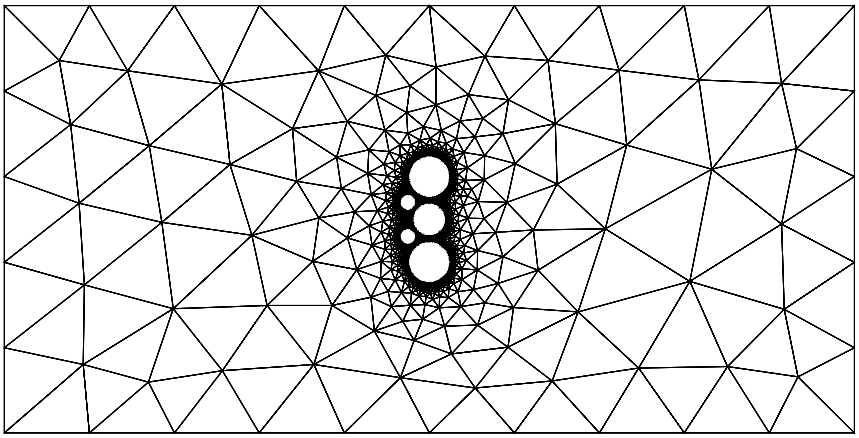

Figure 6. Boundary layer mesh with 1:200 transition

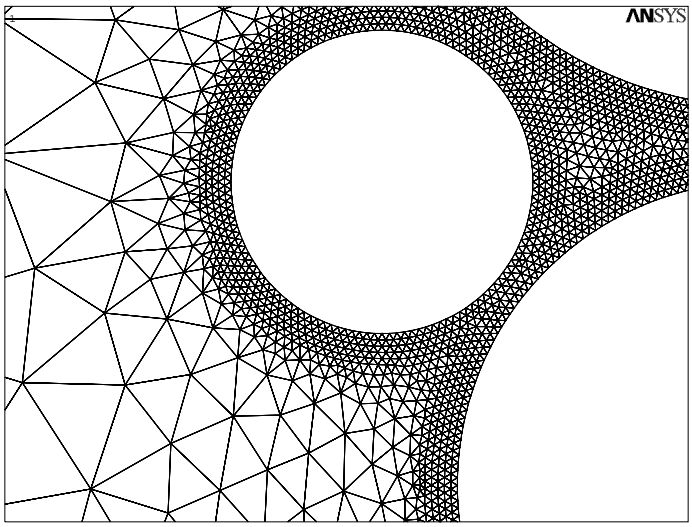

Figure 7. Close-up of boundary layers in Figure 6

Surface Curvature

If areas have internal surface features, inserting additional sizing vertices can drastically improve element quality. What is typically used to control element sizing on curved surfaces is a user specified maximum spanning angle, ϕ. This angle defines a limit as to how far the element edges may deviate from the actual geometry. Some applications use the maximum spanning angle or other discretization criteria during the meshing process (deCougny;1996). By incorporating the sizing based on curvature and maximum spanning angle into the background sizing function, the mesher does not have to compute additional normal calculations for each triangle. The maximum element size based on ϕ, is defined by the following:

$$d_\phi = 2r_c \sin\left(\frac{\phi}{2}\right) \qquad [8]$$

where r_c is the radius of curvature. The radius of curvature, r_c can be approximated from the normals at the surface. The radius of curvature between two locations on the surface that are a linear distance of h apart and whose normals are $\mathbf{N_A}$ and $\mathbf{N_B}$ is approximated by:

$$r_c = \frac{h/2}{\sqrt{\frac{1-\|\mathbf{N}_A\|\cdot\|\mathbf{N}_B\|}{2}}} \qquad [9]$$

Figure 8 shows graphically the relationships from which equation 9 is derived. The normals, $\mathbf{N_A}$ and $\mathbf{N_B}$ with vector **AB** form a triangle from which trigonometric relationships can be developed and where θ is defined as, $\mathbf{N}_A \cdot \mathbf{N}_B$. This relationship is exact for a quadratic surface (ie. sphere, cylinder) but is only an approximation for arbitrary surfaces. Even though exact curvature can be easily computed for some parametric surface types, for faceted geometry, curvature is generally not readily available and can be time

consuming to compute. For this reason, normals are used to approximate the radius of curvature. The accuracy of the resulting radius of curvature will in general improve as h gets small.

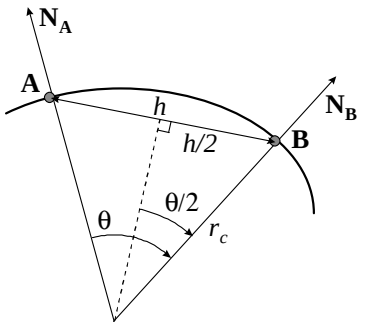

Figure 8. Approximation of radius of curvature between two points **A,B** on a surface.

Depending on whether the mesher works directly in parametric space or in 3D space it may also be necessary to apply a scaling factor to the resulting maximum element size, d_ϕ computed. If parametric space is used, a local scaling factor that maps the size d_ϕ to an equivalent size, d_ϕ' in parametric space is needed as follows:

$$d'_\phi = \left[\frac{h'}{r_c \cos^{-1}\left(\left\| \mathbf{N}_A \right\| \cdot \left\| \mathbf{N}_B \right\| \right)} \right] d_\phi \qquad [10]$$

where h' is the distance in parametric space between points A and B on the surface. The denominator in the above scale factor is simply an expression for the arc length in world space on the surface between the same two points.

Now that a controlling maximum element size has been defined based on a user defined maximum spanning angle, the question still remains as to where new sizing vertices should be placed into the background mesh so that the element sizing function will better describe the surface curvature. It is clear that a random or a gridded distribution of vertices in the background mesh will be inadequate since regions of maximum curvature can very easily be missed. It is typical that only in these regions of high curvature that additional sizing vertices need to be placed. The approach taken for this application was to use a quadtree decomposition of the parametric space of the area.

The quadtree is initialized by evaluating the normals over an N by M grid of points on the surface. N and M should be chosen so that a reasonable initial representation of the surface can be established. A maximum value for N or M defined as 10, appeared to provide sufficient information for most surfaces tested. Each set of four points defining a quadrilateral in the N by M grid serve as the root of a quadtree. The dot product between the normals at the corners of the quadrilateral are computed. The minimum dot product between any two corners is used to approximate the radius of curvature, r_c and maximum element size d_ϕ. The current leaf of the quadtree is subdivided into its four child quadrilaterals if the angle spanned over the quadrilateral is less than the maximum spanning angle, ϕ. The quadtree subdivision is also limited by a minimum allowable element size or maximum number of subdivisions. This is to ensure the quadtree does not go forever on surfaces with sharp folds or discontinuities.

A new vertex is considered for insertion into the background mesh only at the local maximum quadtree level. That is, any quadtree leaf that is not split into four children defines a potential vertex at the centroid of its quadrilateral. The size, d_x defined at the vertex will be the smaller of the element size, d_ϕ, defined from the surface curvature, or d_x defined from an interpolation at the same point before any modifications for curvature. Similar to boundary layers, a vertex is only inserted into the background mesh if the resulting change to the local element size is greater than ε.

Although in many cases, the background mesh is sufficient to control the quality and sizing of the mesh, some additional measures can also be taken. If the advancing front method is used for meshing, it is possible that for large transitions in element size, the smaller element regions can be missed. Typically the size of a new element at the front is determined from an interpolation of the background mesh. For large element sizes, the location interpolated may miss the smaller regions. This can generally be avoided by keeping an ordered list of fronts sorted by size while meshing. If the smallest fronts are always dealt with first, the sizing function can be better captured. A set of bins containing fronts of approximately the same size is sufficient for maintaining the size ordering. Alternatively, rather than a single interpolation into the background mesh, multiple interpolations can be performed at the vertices and/or centroid of the proposed new triangle in the mesh. A weighted average of the computed sizes can be used.

Figure 9 shows the background mesh for a simple parametric surface along with the resulting contours generated using natural neighbor interpolation. Figure 10 shows the resulting triangle mesh using both a maximum spanning angle, ϕ, of 15 degrees and 30 degrees.

Conclusion

Natural Neighbor interpolation has been proposed as a new method for providing element size information to surface meshing algorithms. High quality elements can be generated in situations requiring very large transitions or discretization of highly curved surfaces. While only two specific criteria were used for defining element sizing information, many others can be developed and combined. Future directions will inevitably be application to three dimensions. Three dimensional boundary layer meshing is also an important topic, particularly for CFD applications. While the interpolation method presented has been discussed with respect to isotropic sizing, extensions to incorporate anisotropic criteria will also be considered.

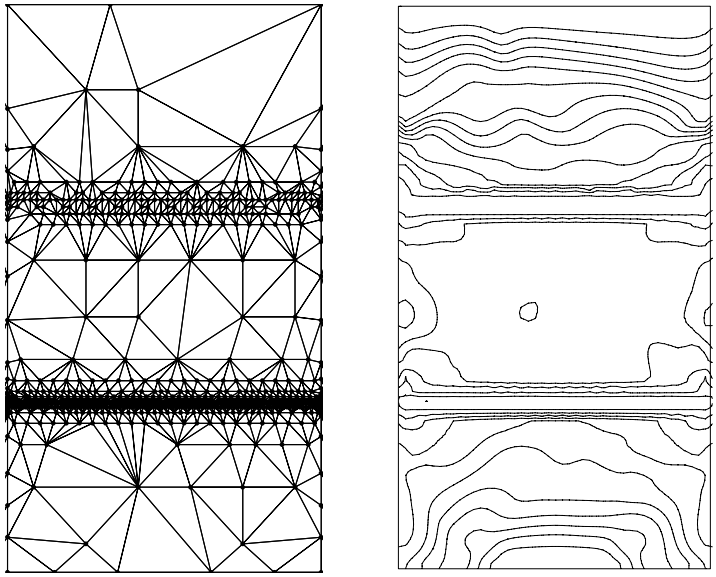

Figure 9. Background mesh and contours of sizing function for parametric surface

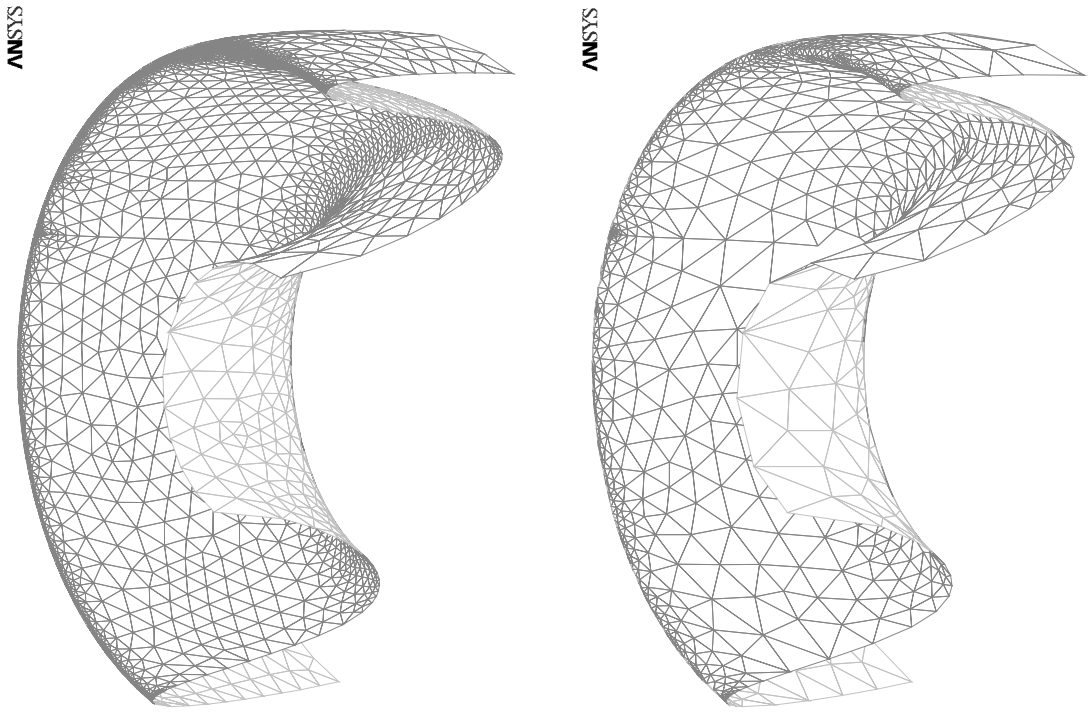

Figure 10. Parametric surface meshed with two different max spanning angles (ϕ). Left ϕ=15 degrees, right: ϕ=30 degrees

References

Canann, Scott A., Yong-Cheng Liu and Anton V. Mobley (1997), "Automatic 3D Surface Meshing to Address Today's Industrial Needs", *Finite Elements in Analysis and Design*. Elsevier, Vol. 25, pp.185-198

Cunha, Alex, Scott A. Canann and Sunil Saigal (1997), "Automatic Boundary Sizing for 2D and 3D Meshes", *AMD-Vol.220 Trends in Unstructured Mesh Generation*, ASME, pp.65-72

Clements, Jan, Ted Blacker and Steven Benzley (1997), "Automated Mesh Generation in Boundary Layer Regions Using Special Elements", *AMD-Vol.220 Trends in Unstructured Mesh Generation*, ASME, pp.137-143.

deCougny, H.L., and M.S. Shephard (1996), "Surface Meshing Using Vertex Insertion", *Proceedings, 5th International Meshing Roundtable*, Sandia National Laboratories, pp. 243-256

Jones, Norman L, Steven J. Owen and Ernest C. Perrie (1995), "Plume Characterization With Natural Neighbor Interpolation", *Proceedings GeoEnvironment 2000,* ASCE, pp.331-345

Lohner, Rainald (1996), "Extensions and Improvements of the Advancing Front Grid Generation Technique," *Communications in Numerical Methods in Engineering*, John Wiley & Sons, Vol. 12, pp. 683-702

Owen, Steven J., (1992), *An Implementation of Natural Neighbor Interpolation in Three Dimensions*, Master's Thesis, Brigham Young University, 119p.

Pirzadeh, Shahyar, (1993), "Unstructured Viscous Grid Generation by Advancing-Layers Method," *AIAA-93-3453-CP*, pp.420-434

Sibson, R., (1981), "A Brief Description of Natural Neighbor Interpolation," *Interpreting Multivariate Data*, John Wiley & Sons, New York, pp. 21-36

Watson, David, F. (1994), nngridr: An Implementation of Natural Neighbor Interpolation

Watson, David, F. (1981), "Computing the Delaunay Tesselation with Application to Voronoi Polytopes", *The Computer Journal*, Vol 24(2) pp. 167-172