

Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition

Joseph R. Tristano, Steven J. Owen and Scott A. Canann
ANSYS, Inc.
275 Technology Drive
Canonsburg, PA 15317
and
Carnegie Mellon University
Department of Environmental and Civil Engineering
Pittsburgh PA
{ joe.tristano, steve.owen, scott.canann }@ansys.com

Abstract

A method is presented for meshing 3D CAD surfaces in parametric space using an advancing front approach and a metric map to govern the size and shape of the triangles in the parametric space. The creation of the metric map will be discussed. The advancing front mesher generates triangles based on the metric map, stretching them in order to capture the change in parameterization of the surface. The benefits of this algorithm include better quality elements without having to do costly real space calculations.

Keywords: Triangulation, free surface meshing, Riemannian metric, CAE, finite elements

1. Introduction

1.1 Importance of work

The finite element method is a powerful tool for today's engineering community. One of the barriers to automating finite element analysis is robust automatic mesh generation on CAD surfaces. There are many manual, semi-automatic, and automatic methods available today, and all have their own advantages and drawbacks¹. Current commercial codes tend to use either advancing front or Delaunay² triangulation to generate surface meshes. These methods, while very robust in two dimensions, are not as effective on 3-dimensional parametric surfaces, common in CAD models produced in industry. Direct three-dimensional extensions of Delaunay³ and advancing front⁴ techniques have been proposed, but tend to be slower and less robust. For this reason, two dimensional methods are preferred, utilizing a parametric surface mapping where appropriate. However, standard two-dimensional methods cannot be used directly since the mapping from parametric space to real space can produce distorted elements. A method is needed to place anisotropic triangles in two dimensions that will map back to isotropic triangles in three dimensions. A metric map based on the first fundamental form of the surface can be used. While a significant amount of research has gone into meshing surfaces using a metric map with Delaunay

triangulation⁵, little work has been done in the area of advancing front^{6,7}. This paper proposes an expanded use of the metric map for use in advancing front mesh generation.

1.2 Previous work

Previous work in this area has been done by George et al at INRIA^{5,8} and others^{9,10} who have investigated the use of the metric and metric map quite extensively with the use of a Delaunay kernel. Möller and Hansbo investigated using a metric to define the shape of triangles in parametric space but did not propose an algorithm to produce the mesh⁶. Cuillère⁷ has also explored the use of a metric in meshing parametric surfaces using advancing front. He explored the notion of an orthogonal vector in Riemannian space (see section 3.4.3.3, equation 14) quite thoroughly.

1.3 Overview of paper

The blend of advancing front and a metric map is new since previous work with the advancing front method has been in parametric space (either directly meshing the parametric space¹¹ or modifying the space¹²) or directly on the 3D surface⁴.

This paper describes the details of generating a finite element mesh in the parametric space of a CAD surface using a Riemannian metric map. Section 2 presents the overall algorithm. Section 3 gives the details of the algorithm. Section 4 presents some results and comparison to other methods. Section 5 draws conclusions and discusses a few areas where work is still needed.

2. Overall Algorithm

As with any advancing front method, the algorithm begins with a set of boundary loop segments, defined as the initial “front”. Triangles are constructed from the front segments and grow towards the interior of the domain, “advancing the front as it proceeds”. More specifically, the proposed algorithm involves the following steps:

- Discretize the boundary (section 3.2)
- Compute background mesh (section 3.3)
- Orient the front segments (section 3.4.1)
- Initialize the fronts (section 3.4.2)
- Process the fronts (section 3.4.3)

For each front:

- Check metric of nodes on front to determine appropriate method for distance calculation (section 3.4.3.1)
- Check to see if the angle between adjacent fronts is below a threshold angle. If it is, see if the formation of a triangle is possible (section 3.4.3.2)
- Determine the best location for a candidate node based on interpolation of the background mesh. (section 3.4.3.3)
- Find all nearby nodes on the current front (section 3.4.3.4)
- See if any of the nearby nodes form an acceptable triangle (section 3.4.3.5)
- If all of the above methods fail, use a brute force method to go through all the remaining nodes on the front and form the best triangle (section 3.4.3.6)
- Check boundary node normals (section 3.4.3.7)
- Form triangle

- Update the front.

The proposed algorithm can be used for both flat and curved surfaces. While curved surfaces require the use of a metric map, flat surfaces can use the identity matrix for the metric, provided an initial transformation of the three dimensional boundary nodes to the x-y plane is first accomplished.

3. Details of the Algorithm

This section will define the metric and it's uses. It will also describe the creation of the background mesh and the surface mesh that utilize the metric.

3.1 Definition of the metric

For 3D surfaces with a parameterization denoted by \mathcal{O} there is a metric of the tangent plane at every point P defined as:

$$[M]_{\mathbf{P}} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \quad [1]$$

where

$$\begin{aligned} E &= \vec{\tau}_1 \cdot \vec{\tau}_1 \\ F &= \vec{\tau}_1 \cdot \vec{\tau}_2 \\ G &= \vec{\tau}_2 \cdot \vec{\tau}_2 \end{aligned} \quad [2]$$

and

$$\begin{aligned} \vec{\tau}_1 &= \vec{\sigma}'_u(u, v) \\ \vec{\tau}_2 &= \vec{\sigma}'_v(u, v) \end{aligned} \quad [3]$$

where $\vec{\sigma}'_u(u, v)$ and $\vec{\sigma}'_v(u, v)$ are the gradient vectors at the point P on the surface.

The 3D distance between two points along the surface, A and B, as measured from point A can be computed using their coordinates in parametric space and the metric at A:

$$\|AB\|_{M_A} = \sqrt{E_A(u_B - u_A)^2 + 2F_A(u_B - u_A)(v_B - v_A) + G_A(v_B - v_A)^2} \quad [4]$$

This function only gives the distance with respect to the metric at A. The distance as computed from both A and B is needed during the meshing process. For a *well behaved* surface the distance can be computed as the average of the distance measured from A and the distance measured from B:

$$\|AB\| = \frac{\|AB\|_{M_A} + \|AB\|_{M_B}}{2} \quad [5]$$

Well behaved implies a minimal deviation of the surface derivatives over the surface. The behavior of the surface is determined during the creation of the background mesh (section 3.3). When the surface is not *well behaved*, numerical integration must be done using a finite number of integration points, N (typically 5), along the line segment between A and B.

$$\begin{aligned}
 d_{3D} &= \sum_{i=1}^n d_i \\
 d_i &= \sqrt{\vec{P_i P_{i+1}} \cdot \left([M]_{avg} \right)_i \cdot \vec{P_i P_{i+1}}} \\
 \left([M]_{avg} \right)_i &= \frac{[M]_i + [M]_{i+1}}{2}
 \end{aligned} \tag{6}$$

3.2 Discretization of the Boundary

The boundary loops of the surface must be discretized in such a way that well shaped elements can be created. A method such as smart sizing¹³ can be used where the boundary is discretized and then refined based on the proximity and curvature of the lines in the model. This allows the mesher to capture the curvature of the surface by putting more element divisions on highly curved boundaries.

3.3 Construction of metric & size map

The metric stated above is used to compute 3D distances in a 2D, parametric domain. Since 3D sizes are stored on the 2D domain, the metric allows for the creation of distortion free triangles while performing all of the meshing in 2D parametric space. This is accomplished by using the metric to distort the triangles in the parametric space so they are distortion free when mapped back to real space.

For the sake of efficiency, a background mesh is defined to control element sizing and metric values. The background mesh consists of selected points in the parametric domain where both size and metric are known exactly. The mesher can utilize this information to interpolate local size and metric data as a function of the parametric u, v coordinates. For example:

$$\begin{aligned}
 size &= f(u, v) \\
 [M]_p &= f(u, v)
 \end{aligned} \tag{7}$$

One of the benefits of computing the metric map and size map using the same background mesh is that all sizing information and metric information is stored in one place. A second benefit is that the metric of a point in parametric space can be determined by a simple interpolation rather than a costly evaluation of the surface followed by the computation of the metric. An additional benefit is the ability to refine the background mesh based on size and metric values at the same time on the same mesh. This, as described by Owen¹⁴, is a fast and convenient way to compute essentially two different maps on one mesh.

The initial background mesh is a Delaunay tessellation of a subset of the domain's boundary nodes in parametric space. Boundary nodes are inserted into the background mesh only if their resulting contribution to the size map would affect it significantly.

The parametric space does not need to be *well behaved* in order for this method to work well. As a matter of fact, one of the strengths of this method is its ability to handle surfaces that do not have a *well behaved* parametric space.

Once the background mesh is generated, the 3D sizes and metrics are stored at the nodes of the 2D background mesh for later reference.

If the element size must change on the interior of the mesh due to surface curvature or due to the metric changes caused by variations in the mapping between parametric space and global space, internal nodes must be inserted into the background mesh so that an accurate representation of the size and metric can be produced.

The internal nodes in the background mesh are placed by using a quadtree decomposition of the parametric space. The quadtree is initialized by evaluating the tangent vectors at each of the points in an $N \times N$ grid. A reasonable value for N is 10. New nodes are inserted as needed at the centroid of each quadtree leaf. The quadtree leaves are refined based on the ratio the deviation of the mapping of the surface from parametric space to real space. This ratio is defined as the maximum magnitude to the minimum magnitude of the tangent vectors at the four corners of the leaf. If this ratio exceeds the maximum slope ratio (1.5 seems to work well for most surfaces) the leaf is refined. If the leaves need to be refined, the surface is not considered to be locally *well behaved*. The level of refinement is limited to a maximum number of iterations as well as by the real world length of the diagonal of the smallest quadtree leaf. If the diagonal length in real space is less than twice the minimum size then refinement of the background mesh is stopped. In addition to being inserted to fully capture transitions in the parametric mapping, nodes are also inserted to capture size transitions and surface curvature (as in Owen¹⁴).

Any interpolation method can be used to create and evaluate the background mesh, but Owen¹⁴ illustrates that natural neighbor interpolation is an attractive method to use since it is C^1 continuous at the nodes of the background mesh. This is important in representing the metric, since it is undesirable to have discontinuities in the metric map. The method also reduces the amount of “banding” i.e., bands of unnecessarily small elements.

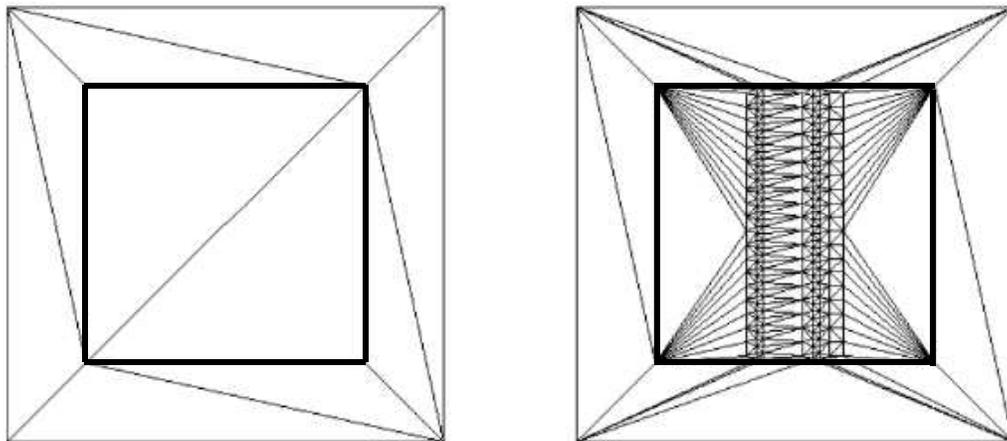


Figure 1 Background mesh before and after refinement

3.4 Advancing front algorithm using the metric map

After the background mesh has been created, the meshing process can begin. While the basic algorithm was outlined in Section 2, the details will be presented here.

3.4.1 Orient front segments

A set of closed boundary loops (closed set of singly connected edges) made up of the initial front serves as input to the advancing front mesher. Boundary loops are oriented such that the exterior loops follow a counter-clockwise direction and the interior loops follow a clockwise direction.

3.4.2 Initialize the fronts

The first step after the boundary has been discretized and the size/metric map has been computed is the initialization of the front:

- Sizes and metrics are interpolated from the background mesh and stored with the boundary nodes. The 3D front length is computed and stored in the front data structure using the distance calculations from Section 3.1.
- For each edge on the front, the equation of the line using the u, v coordinates of the edge's nodes is computed for intersection calculations to be performed later.
- Experience has shown that meshing smaller fronts first can be advantageous. To accomplish this, the fronts are hashed according to their 3D size. The fronts are stored in a collection of bins; each one holding a specified range of sizes. Bin sizes are defined logarithmically, where the front size range for a bin containing small fronts is narrower than for larger fronts. During the meshing process, fronts are first processed from the bin containing the smallest fronts. It was found that sorting based on 3D size is better than 2D size because of problems found with mapping the mesh back to 3D space where the parametric space is highly distorted. Sorting based on 3D distances also helps to better capture size transitions in the mesh.
- The final step in front initialization is to check for intersecting boundary segments. This step is performed in order to check for erroneous or poor input. Each front segment is checked against any non-adjacent front. If any of the boundary fronts intersect, the mesher returns an error code.

3.4.3 Process the fronts

The fronts are now processed from the smallest 3D sized front to the largest, as described in the following sections.

3.4.3.1 Check metric of nodes on the front

Since distance measurement in Riemannian space varies from one location to another throughout the parametric space, special care must be taken where large size transitions occur, or where the parametric space is locally not well behaved.

For each front, the mesher must ensure that the metrics at its end nodes do not exceed the maximum slope ratio defined during the creation of the background mesh. If the ratio is exceeded, distance calculations must be integrated to better approximate the true 3D distance. (Note that in further discussions of distance calculations the ratio of the metrics must be checked in order to determine which distance calculation should be used, i.e. averaging (equation 5) or integrating (equation 6)).

3.4.3.2 Check if the angle between adjacent fronts is below threshold angle

The first step in processing a front is checking the 3D angle it makes between its two adjacent fronts. Using the 3D lengths of the three sides, the law of cosines is used to compute the angle at the desired node as in the following:

$$\theta_A = \cos^{-1} \left(\frac{\|CA\|_{3D}^2 + \|AB\|_{3D}^2 - \|BC\|_{3D}^2}{2\|CA\|_{3D}\|AB\|_{3D}} \right) \quad [8]$$

If the angle is below the threshold angle of 75 degrees as shown in Figure 2, the triangle formed from A, B and C becomes a candidate triangle. Before forming triangle ABC, segment BC must first be checked to ensure it does not intersect any other fronts. This can be done as outlined in section 3.4.3.5.

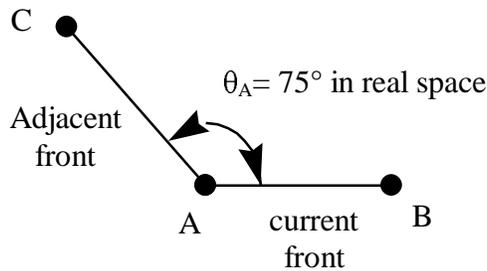


Figure 2 Small angle triangle creation

To ensure that high aspect ratio triangles are not created, an additional check is made before the triangle is created. If the aspect ratio of edge length to front length is more than twice the user controllable growth ratio, g_r , (usually 2.0) the adjacent edge will be split.

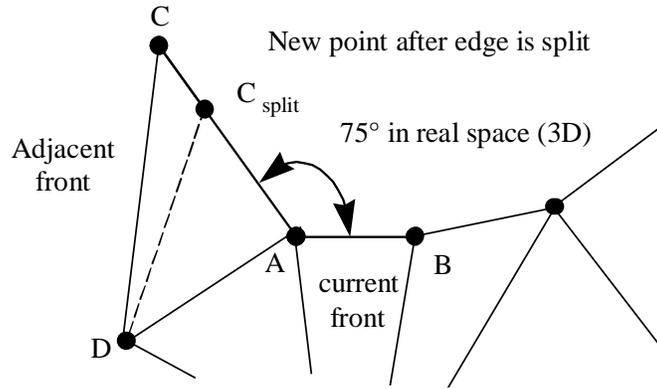


Figure 3 Splitting of triangle because of aspect ratio

For example, in Figure 3, edge AB is the current front and edge AC is an adjacent front with an angle less than 75° . Since edge AC exceeds the maximum transition ratio, the adjacent front AC is split.

$$\frac{\|CA\|_{3D}}{\|AB\|_{3D}} \geq g_r \quad [9]$$

Edge AC is split at location C_{split} , forming a new node, where the length of the new edge AC_{split} is defined as:

$$\|AC_{split}\|_{3D} = \|AC\|_{3D} \frac{\psi_A}{\psi_A + \psi_C} \quad [10]$$

where ψ_A , and ψ_C , are the desired sizes at A and C.

The AC is split at C_{split} in order to create a division that best matches the desired sizes at A and C.

The new triangles CDC_{split} and DAC_{split} are then formed from triangle ACD. The triangle ABC_{split} is then created and the front is updated.

3.4.3.3 Determining the best location for a new node

If a triangle cannot be created using the adjacent fronts, as described above, the location of a candidate node, N_c , that would form an ideal triangle is determined. All of the nodes on the current front, N_{Fi} , that are within a specified 3D distance from N_c are found and sorted by their distance to N_c (see section 3.4.3.4 for details). The closest node that forms a valid triangle with the current front is selected. The following describes this process in more detail.

To find the ideal size, ψ_{3D} of the new triangle, T_n , from front AB the following may be used:

$$\psi_{3D} = \min\left(\frac{\psi_A + \psi_B}{2}, \|AB\|_{3D} \cdot g_r\right), g_r > 1 \quad [11]$$

where ψ_A, ψ_B are the 3D element sizes interpolated from the background mesh at A and B respectively and g_r is a user defined maximum growth ratio.

For flat, non-parametric surfaces, the location of N_c can be constructed by forming an isosceles triangle with front AB, where edge length $\|AN_c\|_{3D} = \|BN_c\|_{3D} = \psi_{3D}$, and the height, h_{3D} , of T_n is defined as:

$$h_{3D} = \sqrt{\psi_{3D}^2 - \frac{\|AB\|_{3D}^2}{4}} \quad [12]$$

Since most parametric mappings are not uniform, $\|AN_c\|_{2D} \neq \|BN_c\|_{2D}$, even though $\|AN_c\|_{3D} = \|BN_c\|_{3D}$.

However, N_c can be located by computing an equivalent h_{2D} and normal vector, \vec{N}_{2D} to front AB, as shown in Figure 4. For example let:

$$\vec{m}_{AB} = \frac{\vec{B}_{2D} + \vec{A}_{2D}}{2} \quad [13]$$

$$\vec{N}_{2D} = \begin{Bmatrix} u_{N_{2D}} \\ v_{N_{2D}} \end{Bmatrix} = \begin{bmatrix} -(F_{m_{AB}} + G_{m_{AB}}) \\ E_{m_{AB}} + F_{m_{AB}} \end{bmatrix} \vec{V}_{AB} \quad [14]$$

where \vec{V}_{AB} is a unit vector pointing from A to B and \vec{m}_{AB} is the midpoint of AB.

Finally, to locate N_c in parametric space, N_{c2D} , the following can be used:

$$\vec{N}_{c2D} = \vec{m}_{AB} + \vec{N}_{2D} \cdot h_{2D} \quad [15]$$

where

$$h_{2D} = \frac{h_{3D}}{\sqrt{E_{m_{AB}} u_{N_{2D}}^2 + 2F_{m_{AB}} u_{N_{2D}} v_{N_{2D}} + G_{m_{AB}} v_{N_{2D}}^2}} \quad [16]$$

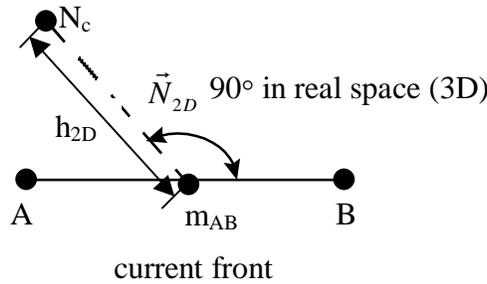


Figure 4 Locating ideal node location

3.4.3.4 Find nearby nodes

It is not always necessary to create a new node at N_c . In many cases, there will be an existing node, N_f , that is close enough to fulfill the local size requirements. A search radius, r_{3D} defined as:

$$r_{3D} = \min(\psi_c, \psi_{3D}) \cdot s_f, \quad 0 < s_f < 1 \quad [17]$$

where ψ_c is the interpolated element size at N_c , and s_f is a shrink factor limiting the radius of acceptable nodes, thereby limiting the size transition in the mesh and reducing the number of distance calculations. A typical range for s_f is between .8 and .4.

Since the search radius is a 3D distance, an ellipse based on the metric at N_c must be used in parametric space to calculate a bounding box to reduce the number of distance calculations. The major radius of this ellipse is used as the width of the bounding box.

$$r_{2D} = \frac{r_{3D}}{\sqrt{E_{N_c} u_{N_c}^2 + 2F_{N_c} u_{N_c} v_{N_c} + G_{N_c} v_{N_c}^2}} \quad [18]$$

Equation 18 is in the same form as an ellipse in polar coordinates centered about the origin. It can then be transformed into the general form of an ellipse in Cartesian coordinates.

$$Ax^2 + By^2 + Cxy + F = 0 \quad [19]$$

where:

$$A = E_{N_c}, B = G_{N_c}, C = 2F_{N_c}, F = r_{3D}^2, x = r_{2D}^2 u_{N_c}, y = r_{2D}^2 v_{N_c} \quad [20]$$

By rotating the ellipse to align with the Cartesian coordinate axis, equation 19 may be reduced and the major radius defined as:

$$a = \frac{1}{\sqrt{\min(A_{rot}, B_{rot})}} \quad [21]$$

where

$$A_{rot} = \left(A \cos^2 \theta + B \sin^2 \theta + \frac{C \sin 2\theta}{2} \right) F^{-1}$$

$$B_{rot} = \left(B \cos^2 \theta + A \sin^2 \theta - \frac{C \sin 2\theta}{2} \right) F^{-1} \quad [22]$$

where

$$\cot \theta = \frac{B - A}{C}$$

The major radius can now be used as the parametric space bounding box width.

A bounding box with width $2a$ centered at N_c can be used to quickly filter nodes, N_{Fi} , on the front that do not fall within the ellipse. The distance equations mentioned in section 3.1 can be used to determine if they are actually within the search radius while not violating the maximum growth ratio γ_r .

3.4.3.5 Validity checks for candidate Triangles

There are three tests that a triangle must pass to be accepted: zero area test, internal node test, and front intersection test.

The first triangle validity check determines if the triangle has zero area in parametric space. Zero area triangles cannot be created.

If T_n passes the zero area test, it must be checked for nodes interior to it or nodes too close to the boundary edges of nearby triangles. The nodes close to the triangle are checked with this same test, referred to as the interior node test.

The area (barycentric) coordinates of a node on the front, N_{Fi} , which lies within the bounding box of the current front are computed with respect to T_n (Figure 5). If any of these coordinates are less than a specified empirical tolerance, N_c must be tested again using the approximate 3D area coordinates. The 3D area of a triangle, γ_{3D} , can be defined from Heron's formula as.

$$\gamma_{3D} = \sqrt{s(s - \|AB\|_{3D})(s - \|BC\|_{3D})(s - \|CA\|_{3D})} \quad [23]$$

where

$$s = \frac{\|AB\|_{3D} + \|BC\|_{3D} + \|CA\|_{3D}}{2} \quad [24]$$

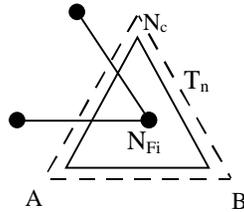


Figure 5 Test for node inside triangle

If any of these area coordinates are algebraically less than an empirical tolerance (currently set to - 0.0154321012) then the triangle passes the interior node test.

If the triangle passes the interior node test, the triangle then goes through the front intersection test. This test checks that the triangle does not intersect any of the current fronts. This intersection test is performed in parametric space.

3.4.3.6 Brute force method

If any of the candidate nodes (nearby nodes plus the ideal node N_c), N_{ci} , fail to create a valid triangle, then the brute force method presented in this section is used to create a triangle. The brute force method consists of looping through all of the nodes on the front and finding the best shaped triangle. A distortion metric¹⁵, α , is used to determine the quality of the triangle. For flat non-parametric surfaces, α is defined as:

$$\alpha = 2\sqrt{3} \frac{\overrightarrow{N_c A} \times \overrightarrow{N_c B} \cdot \hat{n}}{\|N_c A\|^2 + \|AB\|^2 + \|BN_c\|^2} \quad [25]$$

The direction vectors $\overrightarrow{N_c A}$ and $\overrightarrow{N_c B}$ are not known on the parametric surface, so the area squared, γ_{3D}^2 , must be computed using equation 23 above. Using this, the distortion metric in Riemannian space, α_{3D} , can be defined as:

$$\alpha_{3D} = 2\sqrt{3} \frac{2\gamma_{3D}^2}{\|N_c A\|_{3D}^2 + \|AB\|_{3D}^2 + \|BN_c\|_{3D}^2} \quad [26]$$

The distortion metric alone is not always the best method to create the new triangle because size criterion may be violated even though one triangle may be of better quality than another. It was found that the best triangle is created by penalizing the distortion metric of the triangle based on a function of the ratio of desired triangle size and longest triangle edge length. This function is illustrated below:

$$penalty = \begin{cases} xe^{(1-x)} - 0.4\sin\pi x & 0 < x < 1 \\ 0.5(xe^{(1-x)} - \frac{1}{9}x + \frac{10}{9}) & 1 \leq x < \infty \end{cases} \quad [27]$$

where

$$x = \frac{(\psi_A + \psi_B)/2}{\max(\|AN_c\|_{3D}, \|BN_c\|_{3D})} \quad [28]$$

where ψ_A and ψ_B are the desired sizes at A and B.

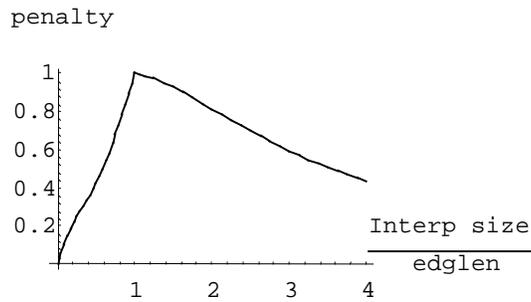


Figure 6 Penalty function for distortion metric

The new alpha then becomes:

$$\alpha_{new} = \alpha_{3D} \cdot penalty \quad [29]$$

3.4.3.7 Checking boundary normals

There are some cases where degenerate surfaces such as cones have highly distorted parametric space. In such cases it is necessary to check the angle spanned by a triangle connected to boundary nodes to prevent the element from “flattening” out (Figure 7).

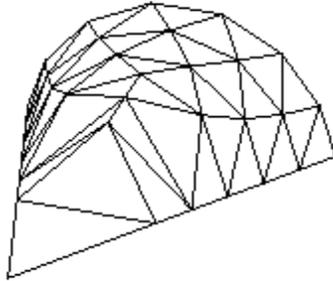


Figure 7 Flattened out cone tip

This check is done only when two or three of the triangle’s nodes lie on mutually exclusive boundary curves. The angle spanned by the edge that cuts across the domain from one boundary to another is checked by computing the angle between the surface normals of the boundary nodes. If the computed angle is below the user controllable maximum spanning angle (usually between 5 and 60 degrees), the triangle is then split at its midpoint along the edge that cuts across the domain.

For example, Let nodes A and B of triangle ABC (Figure 8) lie on different boundary curves. Let the surface normals at nodes A and B be \vec{N}_A and \vec{N}_B . If the dot product of \vec{N}_A and \vec{N}_B exceeds the maximum spanning angle then triangle ABC will be split at its 3D midpoint P along edge AB.

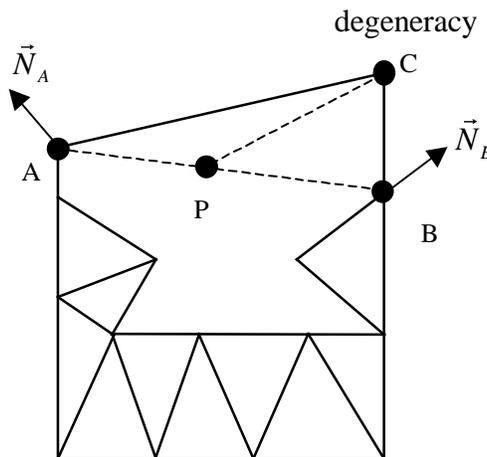


Figure 8 Splitting of flat cone tip in parametric space

4. Results and Comparisons

In this section, the results of the Riemannian space advancing front mesher are compared against the warped parametric space and direct 3D meshers available in the ANSYS program. The warped parametric space mesher¹² is a mesher in the ANSYS program that meshes parametric surfaces in a different manner. This method reparametrizes the surface selectively evaluating surface derivatives ($\Delta\mathbf{u}$, $\Delta\mathbf{v}$) over the domain and adjusting local u,v values to hold the magnitude of $\Delta\mathbf{u}$, $\Delta\mathbf{v}$ roughly constant. Times presented in this section incorporate the total meshing time after boundary discretization. This includes, projecting the 3D boundary nodes to 2D parametric space, background mesh creation, advancing front meshing, cleanup and smoothing, mapping to 3D, and storage of elements to the ANSYS database.

Figures 9(a,b,c) and Table 1 illustrate the advantages of meshing parametric surfaces using a Riemannian surface definition rather than changing the parametric space. The poorly parametrized surface has surface derivatives that are not orthogonal. This phenomena yields poorly shaped, stretched triangles when meshed with the warped parametric space mesher. However, when the surface is meshed using the Riemannian space mesher the resulting triangles are well shaped.

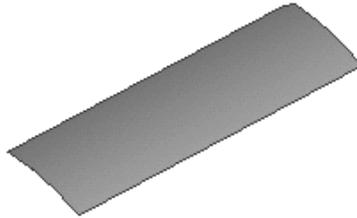


Figure 9a Poorly parameterized surface

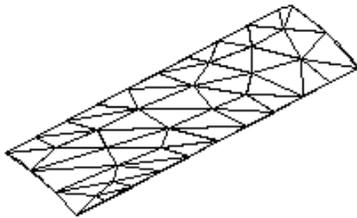


Figure 9b With warped parametric space

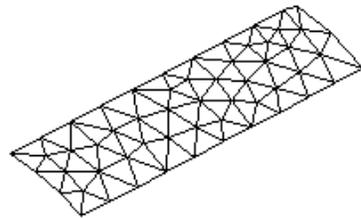


Figure 9c With Riemannian space mesher

Table 1 Distortion metrics of surface mesh

Mesher	Triangles	min α	avg. α
Riemannian space	64	0.778	0.961
warped parametric space	50	0.320	0.743

Figure 10 and Table 2 illustrate the drastic speed improvements that the Riemannian space mesher has over the direct 3D mesher with both meshers yielding meshes of equivalent quality.

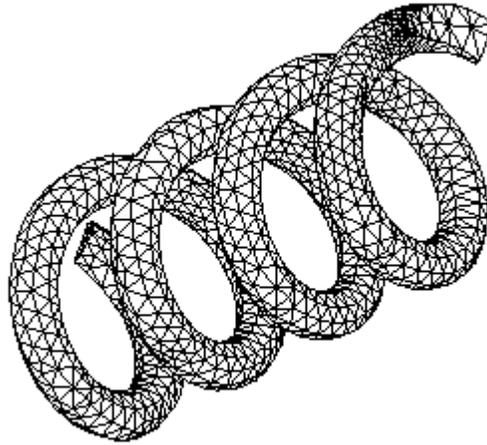


Figure 10 Spring

Table 2 Spring statistics

Mesher	Triangles	min α	Avg. α	time	elements/sec
Riemannian space	3336	0.489	0.923	9.93	335.95
warped parametric space	4356	0.307	0.922	21.89	198.99
direct 3D	3506	0.481	0.927	64.38	54.45

Figures 11(a,b,c) and Table 3 illustrate the overall quality and speed improvements over the warped parametric space and direct 3D meshers for a general CAD surface. The stretched triangles in the warped parametric space mesh are caused by non-orthogonal surface derivatives. The Riemannian space mesher resolves those problems. It also resolves the problems of costly real space calculations done by the direct 3D mesher.

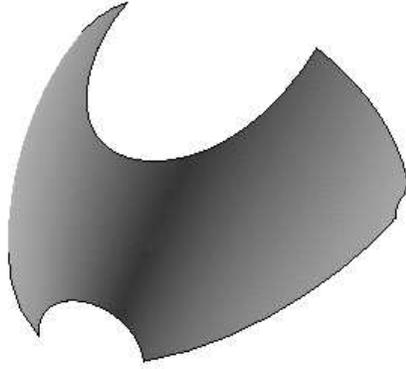


Figure 11a CAD Surface

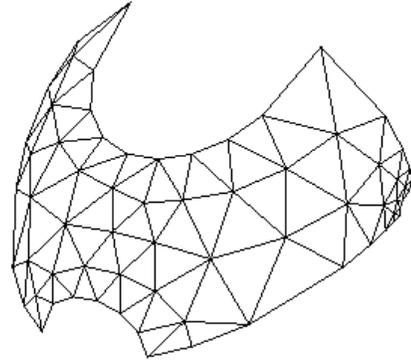


Figure 11b With Riemannian space mesher

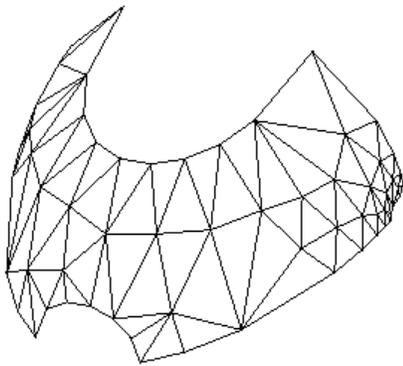


Figure 11c With warped parametric space

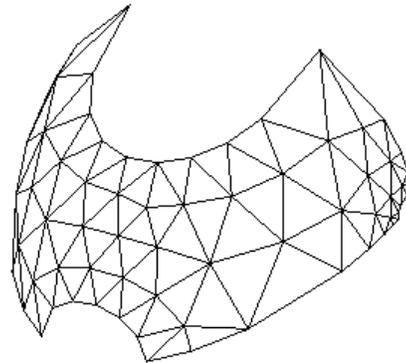


Figure 11d With direct 3D mesher

Table 3 CAD surface statistics

Mesher	Triangles	min α	Avg. α	time	elements/sec
Riemannian space	121	0.650	0.910	0.36	336.11
warped parametric space	115	0.409	0.821	0.29	396.55
direct 3D	125	0.459	0.907	0.74	168.91

5. Conclusion

A method for meshing 3D parametric surfaces using the advancing front method with a Riemannian surface definition was presented. The details of the creation of the metric map used to determine the amount of distortion of the elements in parametric space were given along with the details of an advancing front algorithm that utilizes the metric map.

Meshing surfaces using an advancing front with a Riemannian surface definition proves to be a valuable technique for meshing surfaces common in CAE. This method overcomes anomalies found in the warped parametric space and direct 3D methods currently used in the ANSYS program. Well shaped triangles are produced by this method. Any of three of the methods compared robust and capable of creating high quality meshes for most geometries. However, having all three meshers available in the ANSYS program greatly increases the likelihood of successfully meshing any arbitrary collection of surfaces and hence fully automated constrained tetrahedral meshing. Future areas of work may include the combination of a warped parametric space and Riemannian space mesher in order to create better mappings for high aspect ratio surfaces.

6. References

- 1 Ho-Le, K., Finite element mesh generation methods: a review and classification, *Computer Aided Design*, Vol 20(1) , 27-38, 1988.
- 2 Chen, H., and J. Bishop, Delaunay Triangulation for Curved Surfaces, *6th International Meshing Roundtable Proceedings*, pp.115-127, 1997.
- 3 Chew, Paul L., Guaranteed-Quality Mesh Generation for Curved Surfaces, *9th Annual Computational Geometry*, Vol 73, 1993.
- 4 Lohner,R., Extensions and Improvements of the Advancing Front Grid Generation Technique, *Communications in Numerical Methods in Engineering*, John Wiley & Sons, Ltd, Vol 12, pp.683-702, 1996.
- 5 George P. L., and H. Borouchaki, *Delaunay Triangulation and Meshing Application to Finite Elements*, Editions HERMES, Paris, 1998.
- 6 Moller, P., and P Hansbo, On advancing front mesh generation in three dimensions, *IJNME*, Vol. 38, pp.3551-3569, 1995.
- 7 Cuillère, J. C., An adaptive method for the automatic triangulation of 3D parametric surfaces, *Computer Aided Design*, Vol 30(2), pp.139-149, 1998.
- 8 Castro Díaz, M. J., and F. Hect, Anisotropic Surface Mesh Generation, *INRIA Research Report*, N° 2672, 1995.
- 9 Shimada, K., Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles, *6th International Meshing Roundtable Proceedings*, pp.63-74, 1996.
- 10 Bossen, F. J., P. S. Heckbert, A Pliant Method fo Anisotropic Mesh Generation, *5th International Meshing Roundtable Proceedings*, pp.375-390, 1997.
- 11 Lo, S.H., A new mesh generation scheme for arbitrary planar domains, *IJNME*, Vol. 21, pp1403-1426, 1985.
- 12 Canann, S. A., Y. C. Liu and A. V. Mobley, Automatic 3D Surface Meshing to Address Today's Industrial Needs, *Finite Elements Anal. Des.*, Vol 25, pp.185-198, 1997.
- 13 Cunha, A., S. A. Canann and S. Saigal, Automatic Boundary Sizing for 2D and 3D Meshes, *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ASME, pp.65-72, July 1997.
- 14 Owen, S., Neighborhood-based element sizing control for finite element surface meshing, *6th International Meshing Roundtable Proceedings*, pp..43-154, 1997.
- 15 Lo, S.H. and C.K. Lee, On Using Meshes of Mixed Element Types in Adaptive Finite Element Analysis, *Finite Elements in Analysis and Design*, Elsevier, Vol 11, pp.307-336, 1992.