Generation of Spline Approximations to Tessellations

John F. Dannenhoffer III[1] and Robert Haimes[2]

[1] Syracuse University, Syracuse, NY, USA
jfdannen@syr.edu
[2] Massachusetts Institute of Technology, Cambridge, MA, USA
haimes@mit.edu

Summary. In geometrical modeling, one is often provided a description of a surface that is defined in terms of a triangulation, which is supported by a discrete number of nodes in space. These faceted surface representations are defined to be C-0 continuous, and therefore in general have slope and curvature discontinuities at the triangle sides, unless the tessellation is planar. Unfortunately, analytical and computational methods often require a surface description that has well-defined and smoothly-varying gradients and curvatures; in general spline surfaces possess such properties. Described herein is a process for generating a cubic spline surface that approximates, to within a user-specified tolerance, a given tessellated surface that may be non-convex or multiply-connected. The method combines a local least-squares technique for specifying knot properties as well as an adaptation technique of selecting the necessary knot spacings. This new technique is first described along a curve for illustrative purposes. It is then expanded to the case of the general surface. A reparameterization technique that is required for surfaces with non-smooth parameterizations is described next. Computed results for two configurations are then shown.

1 Introduction

1.1 Background

Creating spline fits of general data is a subject that has received much attention over the years. In fact, there are many books devoted to spline generation and applications[1, 2]. In most of the spline fitting literature, it is assumed that data is known at a specified number of knots, and the spline is constructed so as to pass through the knots and such that it has a specified level of smoothness (most frequently C-2 continuity).

There has been limited literature concerning the generation of splines that fit, in a least-squares sense, a set of data that are not located at the knots[3, 4, 5, 6]. In these cases, the knot locations and/or values can be modified to both minimize the distance between the resulting spline and the data, and to guarantee a specified level of smoothness. These techniques were developed primarily for cases with a larger number of data points than spline knots. Unfortunately, all these techniques are only presented with one independent parameter. Also, if the data are clumped (i.e., there are portions of the parameter's domain that

contain no data points), the only way to apply these schemes is to move the knots to the regions where the data exist.

The extension of these techniques from a curve (one independent parameter) to a surface (two independent parameters) is not straightforward as there are a few other concerns that must be addressed. First, efficient spline evaluation techniques always operate on rectangular grids of knots. Therefore the ability to place knots is restricted; the only degrees of freedom that are available are the values of the dependent variables at the knots (as well as perhaps the total number of knots in each grid direction). Second, surfaces in general can be non-convex and can contain an arbitrary set of holes. Therefore along any constant parameter line there are likely to be gaps, which must be dealt with in an appropriate manner.

1.2 Problem Statement

The problem here is to create a spline surface that approximates, to within a user-specified tolerance, a surface that is defined in terms of a tessellation. In particular, assume that the surface to be fitted is defined in terms of N nodes, at which the space coordinates x_n, y_n, and z_n $(n = 1, \ldots, N)$ are known. Also known at each of the nodes are two parametric coordinates, u_n and v_n, which are defined prior to the execution of this algorithm. The nodes are connected into a tessellation that is defined in terms of T triangles, which are each defined in terms of its three nodes, $n1_t$, $n2_t$, and $n3_t$, $(t = 1, \ldots, T)$ as well as its three neighboring triangles $t1_t$, $t2_t$, and $t3_t$; triangle sides that are at the edge of the surface or which bound a hole within the surface have no neighbor. There is no assumption here that the surface is convex; additionally the surface can contain any number of arbitrarily-shaped and sized holes.

The objective then can be stated as: find the set of uniformly-spaced knots, $\hat{u}_{i,j}$ and $\hat{v}_{i,j}$ $(i = 1, \ldots, I$ and $j = 1, \ldots, J)$ and their corresponding space coordinates ($\hat{x}_{i,j}$, $\hat{y}_{i,j}$, and $\hat{z}_{i,j}$) that can be used to define a cubic spline surface. The cubic-spline surface should be such that the error associated with evaluating the spline at each of the surface's nodes (e_n) is within the user-specified tolerance e_{tol}. The definition of the node errors is given by

$$e_n = \left[(\tilde{x}_n - x_n)^2 + (\tilde{y}_n - y_n)^2 + (\tilde{z}_n - z_n)^2\right]^{1/2} \tag{1}$$

where $\tilde{x}_n$, $\tilde{y}_n$, and $\tilde{z}_n$ are the values that come from an evaluation of the spline at each of the nodes (u_n and v_n).

The problem has been stated in simple terms: given a set of triangles (with (x, y, z) and (u, v)), return a spline surface. The following are just some of the practical applications:

Topological Editing. The Boundary Representation (BRep) that is produced by modern CAD systems is both a function of the configuration and the modeling operations used by the CAD operator. Several modern tools have adopted

the concept of a quilt, in which the faces in the original BRep are merged into quilts (or super-faces). One of the requirements of this merge operation is the ability to create an analytic representation of the merged faces; in fact, the generation of a spline surface for the merged faces is the driving motivation for the work described in this paper. It allows for the formation of a new BRep that stands on its own, where analytic representations of the geometry exist for each entity.

Converting a triangulation to a BRep. Sometimes the geometry of a configuration is only available via a legacy (triangular) representation. The ability to segment a triangulation, parameterize the segments, produce spline approximations, and then trim the resultant surfaces are all necessary techniques for converting a triangulation into the analytic representation that some analytical/computational tools require.

Retriangulation. The given triangulation may not be appropriate for 3D meshing or analysis. CAD systems can provide triangulations in conjunction with the BRep or they may be the result of a quilting operation. More importantly, if the source is a triangulation but the quality is not appropriate, then the tessellation will need to be reconstructed. Many quality surface triangulation schemes require knowledge of both the space coordinates (x, y, z) as well as parametric coordinates, (u, v). Additionally, many of these techniques require smoothly-varying surface slopes and curvatures (if the actual geometry is smooth). A spline surface that is fitted to the original surface naturally provides such a mechanism.

Multidisciplinary Coupling. One of the keys to multi-disciplinary analysis, such as fluid/structure interaction, is the transfer of analysis data on the surface between disciplines. Direct transfer of data from one analysis's grid nodes to the other's is not a well defined operation because the nodes are generally not co-located. If a smooth approximation to the analysis data can be constructed and tied to the analytic surface, then the transfer can be made simply. Therefore, given a surface parameterization and a scalar or vector field at the nodes, the goal is to provide an interpolant for the field data on the surface. This requires the same algorithms as described in this paper, but instead of (x, y, z), field values are used in the fitting process. The end result can be used to interpolate these quantities at a specific (u, v) on the surface and therefore onto the other discipline's mesh nodes.

Behavioral Morphing. The analysis of geometric shapes that change in response to the physical phenomena being modeled is a challenging problem. An obvious example is a structural member under stress, but the motivating example here is in turbomachinery, where the analysis is performed *hot* (i.e. at temperature and up to rotational speed) but the manufactured part (in the CAD system) is *cold*. Given a parameterization that defines the surface for an individual face in the BRep, it is possible to transform the given displacement field (however it is supplied) into an interpolating function, as described above, based on the surface's (u, v) space. It is then simple to create a new

surface that is displaced from the original surface; $(x, y, z) = f(u, v)$ just becomes $(x, y, z) = f(u, v) + g(u, v)$ where g is the displacement interpolating function.

1.3 General Approach

The proposed solution to this problem is given by:

1. Pick a 2×2 array of knots $(\hat{u}_{i,j}, \hat{v}_{i,j})$ that surrounds the parametric coordinates in the original tessellated surface.
2. Find the space coordinates at the knots that minimizes, in a least-squares sense, the errors between each of the nodes and their spline counterparts.
3. If the maximum error if less than a user-specified tolerance, STOP.
4. Otherwise
   a) refine the number of knots in the i direction and then re-execute the least-squares algorithm (and its error)
   b) refine the number of knots in the j direction and then re-execute the least-squares algorithm (and its error)
   c) keep the knots (from a or b above) that have the smaller maximum error and repeat back to step 3.

1.4 Roadmap

Each of these steps is described below. This paper begins with a derivation of the proposed scheme along a curve (i.e., only one parametric coordinate, u), in order to both analytically and graphically describe the algorithm. Sample results of the basic algorithm are shown and the need for *smoothing* is highlighted; the smoothing formulation used here is then presented and more sample results are shown.

The section that follows re-derives the scheme for a surface with two parametric coordinates (u and v). The two-dimensional scheme can result in a very large sparse matrix; the technique used to solve the system is described. This section also concludes with some sample results.

One of the limitations of the proposed technique is that it requires that the provided (u, v) parameterization be relatively smooth. A section that shows what happens when you do not have smoothness, the method used to regularize the parameterization, and the resulting approximation follows.

Sample results for a two configurations follow. The paper concludes with a summary of the scheme and its benefits.

2 Spline Approximation for a Curve

In this section, the basic scheme is derived for the case of a curve in space that is defined in terms of one parametric coordinate, u. Although this result is sometimes useful by itself, it is presented here to simplify the algebra and so that a graphical interpretation of the results can be readily made.

2.1 Ferguson Splines

A Ferguson spline is a special form of a piece-wise cubic spline that was popularized by D. Ferguson[7]. In particular, consider a spline fit of one space coordinate x as a function of u. The characteristics of the Ferguson spline are:

- the spline is defined by a uniformly-spaced set of knots $\hat{u}_i$ $(i = 1, \ldots, I)$ so that $\hat{u}_i - \hat{u}_{i-1} =$ constant.
- as with all piece-wise cubic splines, the spline has C-2 continuity throughout;
- the particular flavor of Ferguson splines used here has natural end conditions (that is, the curvature vanishes at the end points $\hat{u}_1$ and $\hat{u}_I$. (In general, Ferguson splines for which the end slopes are prescribed can be generated, but are not considered here); and
- the spline creation process finds and stores the slopes $(\hat{X}_i)$ at each of the knots.

Given the above, the cubic in each interval i to $i+1$ can be written as

$$\tilde{x} = \tilde{A}\hat{x}_i + \tilde{B}\hat{x}_{i+1} + \tilde{C}\hat{X}_i + \tilde{D}\hat{X}_{i+1} \tag{2}$$

where the basis functions are given by

$$\tilde{A} = 1 - 3\tilde{s}^2 + 2\tilde{s}^3 \qquad \tilde{B} = 3\tilde{s}^2 - 2\tilde{s}^3$$
$$\tilde{C} = \tilde{s} - 2\tilde{s}^2 + \tilde{s}^3 \qquad \tilde{D} = -\tilde{s}^2 + \tilde{s}^3$$

and where $\tilde{s} = (\tilde{u} - \hat{u}_i)/(\hat{u}_{i+1} - \hat{u}_i)$ is the fractional distance of $\tilde{u}$ between $\hat{u}_i$ and $\hat{u}_{i+1}$.

To set up the spline, we want to make the curvatures match at the interior knots. Using the above, this can be written as

$$3\hat{x}_{i-1} - 3\hat{x}_{i+1} + \hat{X}_{i-1} + 4\hat{X}_i + \hat{X}_{i+1} = 0 \qquad (i = 2, \ldots, I-1) \tag{3}$$

The natural end conditions imply that the curvature vanishes at the endpoints. This can be written

$$3\hat{x}_1 - 3\hat{x}_2 + 2\hat{X}_1 + \hat{X}_2 = 0 \tag{4}$$
$$3\hat{x}_{I-1} - 3\hat{x}_I + \hat{X}_{I-1} + 2\hat{X}_I = 0 \tag{5}$$

The above I equations form a linear set for the $\hat{X}_i$ with a given set of $\hat{x}_i$. The resulting matrix equation is tridiagonal, it can be solved in $O(I)$ time.

2.2 Least-Squares Problem

In the above technique, it was assumed that the knot values $\hat{x}_i$ were known and we had to find the slopes $\hat{X}_i$ that yielded C-2 continuity. In this section, we do not *a priori* know the knot values and thus have to find the $\hat{x}_i$ as well as the $\hat{X}_i$.

Assume that we are given the desired number of knots, I. The first step is to define a vector of knot coordinates, $\hat{u}_i$, that are both evenly spaced and such

all the u_n fall between $\hat{u}_1$ and $\hat{u}_I$. If we define a small buffer, b, to ensure that the spline always interpolates (and never extrapolates), then the above can be written

$$\hat{u}_1 = u_{\min} - b(u_{\max} - u_{\min})$$
$$\hat{u}_I = u_{\max} + b(u_{\max} - u_{\min})$$

where

$$u_{\min} = \min_{n=1}^{N} u_n \qquad \text{and} \qquad u_{\max} = \max_{n=1}^{N} u_n$$

For all cases shown here, $b = 0.10$.

Combining these, we finally get

$$\hat{u}_i = \left[\frac{(1+b)(I-i) - b(i-1)}{I-1}\right] u_{\min} + \left[\frac{(1+b)(i-1) - b(I-i)}{I-1}\right] u_{\max} \tag{6}$$

The sum-squared error can then be defined (using Eq. 1) as

$$E = \sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} \left[(A_n \hat{x}_{in} + B_n \hat{x}_{in+1} + C_n \hat{X}_{in} + D_n \hat{X}_{in+1}) - x_n\right]^2 \tag{7}$$

where in in the knot immediately preceding u_n so that $\hat{u}_{in} \le u_n < \hat{u}_{in+1}$. Also, $A_n = 1 - 3s_n^2 + 2s_n^3$, where $s_n = (u_n - \hat{u}_{in})/(\hat{u}_{in+1} - \hat{u}_n)$.

To minimize E, we force the I partial derivatives, $\partial E/\partial \hat{x}_i$ to vanish

$$\frac{\partial E}{\partial \hat{x}_i} = \sum_{n=1}^{N} 2(A_n \hat{x}_i + B_n \hat{x}_{i+1} + C_n \hat{X}_i + D_n \hat{X}_{i+1}) F_n = 0 \tag{8}$$

where

$$F_n = \begin{cases} A_n \text{ if } \hat{u}_{i-1} \le u_n < \hat{u}_i \\ B_n \text{ if } \hat{u}_i \le u_n < \hat{u}_{i+1} \\ 0 \quad \text{otherwise} \end{cases}$$

The above I equations can be rearranged into the form

$$S_1 \hat{x}_{i-1} + S_2 \hat{x}_i + S_3 \hat{x}_{i+1} + S_4 \hat{X}_{i-1} + S_5 \hat{X}_i + S_6 \hat{X}_{i+1} = S_7 \tag{9}$$

with the shorthands

$$S_1 \equiv \sum_{n \in nm_i} A_n B_n \qquad S_2 \equiv \sum_{n \in nm_i} B_n^2 + \sum_{n \in np_i} A_n^2$$
$$S_3 \equiv \sum_{n \in np_i} A_n B_n \qquad S_4 \equiv \sum_{n \in nm_i} B_n C_n$$
$$S_5 \equiv \sum_{n \in nm_i} B_n D_n + \sum_{n \in np_i} A_n C_n \qquad S_6 \equiv \sum_{n \in np_i} A_n D_n$$
$$S_7 \equiv \sum_{n \in nm_i} B_n x_n + \sum_{n \in np_i} A_n x_n$$

where nm_i is the set of nodes for which $\hat{u}_{i-1} \leq u_n < \hat{u}_i$ and np_i is the corresponding set when $\hat{u}_i \leq u_n < \hat{u}_{i+1}$.

We can now form a large matrix equation from Eqs. 3 to 5 and Eq. 9, which can be written in the general form

$$\begin{bmatrix} M_1 & M_2 \\ M_5 & M_6 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{X} \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{10}$$

Here the sub-matrices M_1, M_2, and R_1 are taken directly from Eq. 9 and sub-matrices M_5 and M_6 are taken from Eqs. 3 to 5. The resulting matrix equation, which is quite sparse, can be solved in any convenient manner.

2.3 Numerical Solution

The above algorithm can be applied to a sample data set that was obtained by taking a constant-v-parameter cut through one of the surfaces of an example configuration. The original triangulation for the example surface is plotted in the (u, v) plane in Fig. 1, with the selected cut at $v = 2$ shown by the horizontal line. Fig. 2(a) shows the $x(u)$ data along this cut as a series of circles. One can

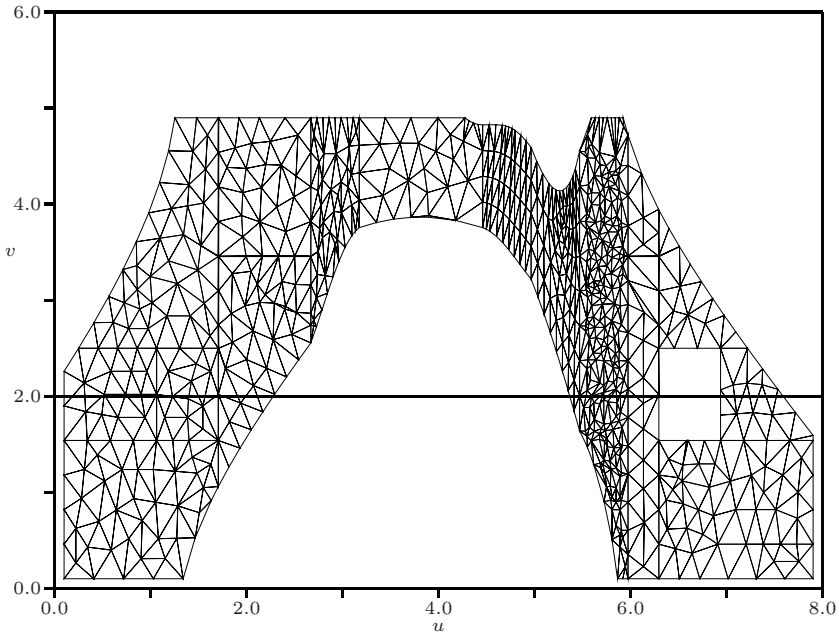
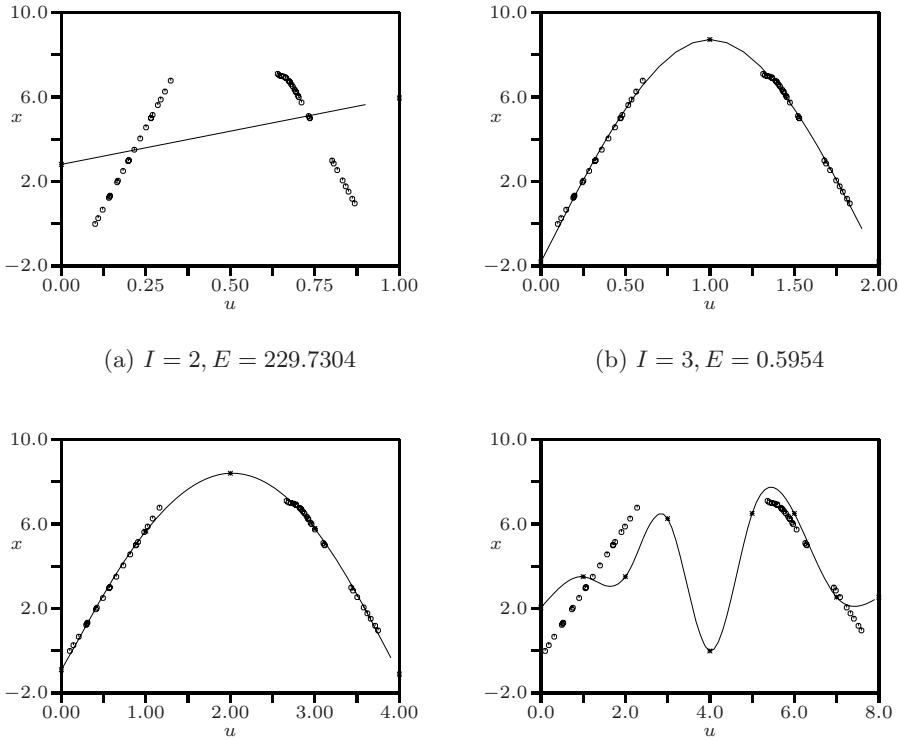


Fig. 1. Prescribed triangulation for a test configuration. The horizontal line shows the cut used in Figs. 2 and 3. This triangulation is the result of merging 13 faces in an original BRep.



(a) $I = 2, E = 229.7304$

(b) $I = 3, E = 0.5954$

(c) $I = 5, E = 0.6366$

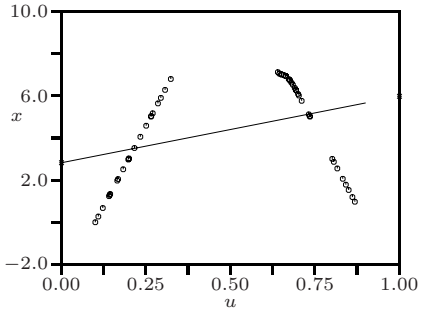
(d) $I = 9, E = 78.3642$

Fig. 2. Spline approximations along a curve without smoothing ($\mu = 0$)

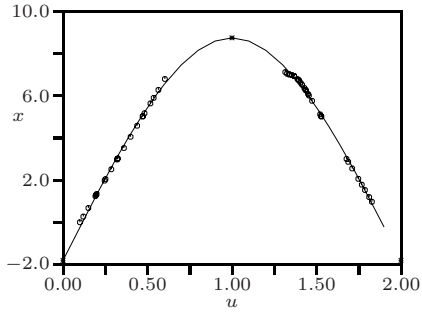
note three regions in which data is defined; the gap in the middle comes from the shape of the outer boundary of the triangulation whereas the smaller gap near the right comes from the rectangular hole in the triangulation.

The first approximation, shown in Fig. 2(a), has only two knots, resulting in a spline that is simply a straight line. The caption of the figure states that when the number of knots, I is 2 the sum-squared error, E, is about 200; clearly this is not a good approximation. In part (b) of the figure, the number of knot-intervals was doubled (taking I from 2 to 3); here the fit is visually much better and the error has decreased to a little more than 1/2. When $I = 5$ the fit is about as good as before. Unfortunately when the number of knots is increased to $I = 9$ the spline becomes worse, with wild oscillations present. A careful look at the figure shows that the oscillation occurs in the region where there is insufficient data to support the spline. In other words, in the vicinity of the oscillations, the number of degrees of freedom in the spline knots is locally greater that the

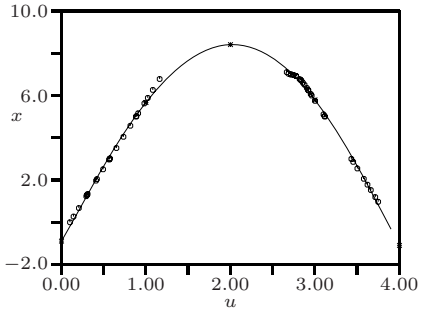
number of nodes for which an error is being minimized. Also, examination of the matrix equation 10 reveals that there are some rows that are not diagonally dominant (and in fact can become degenerate).



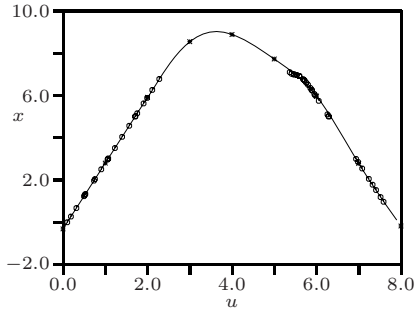
(a) $I = 2, E = 229.7304$



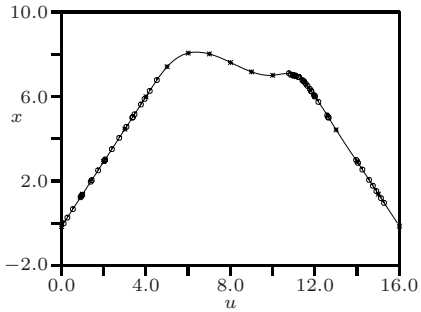
(b) $I = 3, E = 0.5952$



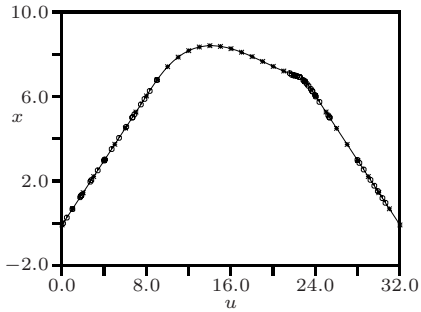
(c) $I = 5, E = 0.6369$



(d) $I = 9, E = 0.1672$



(e) $I = 17, E = 0.0013$



(f) $I = 33, E = 0.0003$

Fig. 3. Spline approximations along a curve with smoothing ($\mu = 0.001$)

2.4 Smoothing

To fix the above, there are two modifications to the basic least-squares equations that appear to be necessary. First for interior knots, it has been found useful to add a very small amount of Laplacian *smoothing* to the least-squares equations. In particular, adding

$$\mu\left(-\hat{x}_{i-1}+2\hat{x}_i-\hat{x}_{i+1}-\frac{1}{2}\hat{X}_{i-1}+\frac{1}{2}\hat{X}_{i+1}\right) \qquad (i=2,\ldots,I-1)$$

to the i-th Eq. 10 has been found to work very well, where μ is a very small parameter (here taken to be $\mu = 0.001$). Note that in regions where there exists enough data for least-square minimization, these smoothing terms are negligible. However in regions where there are no nodes (that is, both nm_i and np_i are empty sets), the smoothing restores the diagonal dominance in the matrix.

The other modification is associated with the knots at the ends of the spline. At those knots, the above smoothing cannot be used since there is not an outboard knot. For these cases, it has been found sufficient to transfer the *natural end condition* associated with the Ferguson splines to the first inboard knot. For example, if the first rows of M_1, M_2, and R_1 are all zero, we can change that equation to a zero-curvature condition at the inboard knot, or

$$3\hat{x}_1-3\hat{x}_2+2\hat{X}_1+4\hat{X}_2=0 \tag{11}$$

A similar modification is added in the case where the last row of M_1, M_2, and R_1 are all zero, which occurs when there are no nodes between the last two knots.

Fig. 3 shows the results of applying this scheme, with smoothing, to the nodes that was shown previously in Fig. 2. Note that as more knots I are added, the shape of the spline better approximates the shape of the node data. Note in particular how smooth the spline is in the region that does not contains any nodes. Also shown in the figure is the sum-squared error, E, which clearly decreases as the number of knots increases.

3 Spline Approximation for a Surface

The extension of the above techniques from a curve to a surface is straightforward, but algebraically tedious. In this section, the most important aspects of the extension are discussed.

Surfaces are defined in terms of two parametric coordinates, u and v. Hence the knots at which the spline data is known form a uniformly-spaced two-dimensional array; the parametric coordinates at the knots are thus $\hat{u}_{i,j}$ and $\hat{v}_{i,j}$, where $i=1,\ldots,I$ and $j=1,\ldots,J$. The knot values of the dependent variables are $\hat{x}_{i,j}$, $\hat{y}_{i,j}$, and $\hat{z}_{i,j}$; for illustrative purposes, only $\hat{x}_{i,j}$ will be considered here.

3.1 Ferguson Splines

Creation of a Ferguson spline for a surface involves the determination of the slopes in both parametric directions, $\hat{X}^u_{i,j}$ and $\hat{X}^v_{i,j}$, as well as the cross-derivatives $\hat{X}^{uv}_{i,j}$. In practice when all the knot values $\hat{x}_{i,j}$ are known, the slopes

$\hat{X}^u_{i,j}$ are found by applying Eqs. 3 to 5 independently along each constant j line; then an analogous technique for finding $\hat{X}^v_{i,j}$ is used independently along each constant i line. Finally, the cross-derivatives $\hat{X}^{uv}_{i,j}$ can be found by applying Eqs. 3 to 5 along constant i lines to the $\hat{X}^v_{i,j}$ found previously (or alternatively by fitting $\hat{X}^u_{i,j}$ along constant j lines).

The evaluation of a Ferguson spline at $(\tilde{u}, \tilde{v})$ is performed using a two-dimensional form of Eq. 2, where there are now 16 basis functions that are polynomial functions of $\tilde{s}$ (the fractional distance of $\tilde{u}$ between $\hat{u}_i$ and $\hat{u}_{i+1}$) and $\tilde{t}$ (the fractional distance of $\tilde{v}$ between $\hat{v}_j$ and $\hat{v}_{j+1}$) .

3.2 Least-Squares Problem

The least-squares problem for finding the $\hat{x}_{i,j}$ that minimize the error E is also conceptually straightforward.

First, the parametric coordinates at each of the knots, $\hat{u}_{i,j}$ and $\hat{v}_{i,j}$, are found by independently scaling the u and v via equations similar to Eq. 6. Note that the sums used here are over all nodes. Then an equation that is analogous to Eq. 9 is formed, where the sums are over the set of nodes that are in the four quadrilateral cells that surround (i, j). Next, a large, sparse matrix equation

$$\begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ M_5 & M_6 & 0 & 0 \\ M_7 & 0 & M_8 & 0 \\ 0 & 0 & M_9 & M_{10} \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{X}^u \\ \hat{X}^v \\ \hat{X}^{uv} \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

is formed. Here, the sub-matrices M_1 to M_4 and R_1 are written directly from the analog of Eq. 9. The second set of sub-matrices (M_5 and M_6) come from the spline-fit of $\hat{x}$ in the i-direction (which yields $\hat{X}_u$). In a similar way, the third set of sub-matrices (M_7 and M_8) come from fitting $\hat{x}$ in the j-direction. Finally, M_9 and M_{10} come from splining $\hat{X}^v$ in the i-direction. Note that since the spline data $\hat{x}_{i,j}$ are not known prior to generating the least-square fit, the spline generation process requires that all three spline-generation sweeps be done simultaneously.

Smoothing is added as above, by adding the Laplacian operator

$$\begin{aligned} &\mu(4\hat{x}_{i,j} - \hat{x}_{i-1.j} - \hat{x}_{i+1.j} - \hat{x}_{i,j-1} - \hat{x}_{i,j+1}) \\ &+\frac{\mu}{2}(-\hat{X}^u_{i-1,j} + \hat{X}^u_{i+1,j} - \hat{X}^v_{i,j-1} + \hat{X}^v_{i,j+1}) \end{aligned}$$

to the matrix equation (as above). Finally, if any of the boundary knots ($i = 1$, or $i = I$, or $j = 1$, or $j = J$) have a degenerate equation in the matrix (because the are no nodes in the cells adjacent to the knot), the natural end condition is applied at the adjacent inboard knot (for example, at $(2, j)$).

The above matrix equation is rather large, but is very sparse. In fact each of the sub-matrices M is pentadiagonal. To see how large the matrix is, consider the generation of a 65×17 array of knots, which results in a square matrix

that contains almost 20 million elements, fewer than one-half percent of which are non-zero. Clearly forming and solving the full matrix is unacceptable and thus sparse-matrix methods must be used. Here, the sparse-matrix data storage format and the bi-conjugate gradient solver described in [8] are used. The only delicate part of using this sparse-matrix technique is the direct creation of the matrix in sparse-matrix form.

4 Overall Algorithm

The above section describes the technique used to find the knot data, $\hat{x}_{i,j}$, for a spline that matches, in a least-squares sense, the data values prescribed at the nodes, x_n. Since in general we want to generate fits for all three space coordinates, (x, y, z), the above technique can be applied successively to x_n, y_n, and z_n. However, the objective is to find the knots that result in splines that are within a user-specified tolerance, e_{tol}, of the knot data. Above, a general technique is outlined; this section expands on that description.

As mentioned above, the method starts out with a 2×2 array of knots; the technique described above is then applied with $I = 2$ and $J = 2$ and the error at each node, e_n, is computed for each space coordinate. The maximum of all these errors

$$e_{\max} = \max_{n=1}^{N} (e_n^x,\ e_n^y,\ e_n^z)$$

is compared with the tolerance, e_{tol} to determine if the knot set is sufficient. This coarse knot grid turns out to be sufficient for cases in which the tessellated surface is planar.

If $e_{\max}$ is larger than e_{tol}, then we do not have a sufficient number of knots. In this case, two refinements are made:

- First the number of knot intervals in the i-direction is doubled, or $I_{\text{new}} \leftarrow 2I - 1$. The least-squares algorithm is executed on a $I_{\text{new}} \times J$ grid and the maximum resulting error, called e_I, is computed.
- Next the number of knot intervals in the j-direction is doubled, or $J_{\text{new}} \leftarrow 2J - 1$. The least-squares algorithm is executed on a $I \times J_{\text{new}}$ grid and the maximum resulting error, called e_J, is computed.

The grid of knots with the smaller error becomes the new set of knots. For example, if $e_I < e_J$, then the new grid will be $I_{new} \times J$.

If the error associated with this new grid (just computed) is smaller than e_{tol}, the overall procedure terminates; otherwise the process is repeated until either a sufficient grid is found or until a maximum grid size is reached; in practice, the latter does not happen if the specified tolerance is reasonable. For all the cases shown here, e_{tol} is set to 10^{-4} of the largest dimension (in (x, y, z)) of the surface.

4.1 Improvements in the Parameterization

After exercising the above on over 2500 surfaces taken from both test and real-world configurations, it was found that the above algorithm sometimes results in

a set of knots that exhibit large oscillations; other times, the algorithm produces good results within the trimming curves of the surface, but exhibits unwanted behavior beyond the surface's trimming curve, indicating that the spline would not be suitable for extrapolating a small distance beyond the edge of the surface. (This extrapolation might be needed for an algorithm such as surface-surface intersection.) Careful examination of these cases showed that this unwanted behavior was always associated with nearly-infinite slopes in the underlying parameterization.

Clearly the answer to this is to reparameterize so that these nearly-infinite slopes are eliminated. For a curve, this reparameterization is fairly easy to accomplish through stretching. Unfortunately, this reparameterization for a surface is not quite as straightforward.

Stated simply, the objective of the reparameterization is to make the spacings of the nodes in parametric (u, v) space to be proportional to the node spacing in terms of the space coordinates, (x, y, z). Here we have adopted the reparameterization proposed by Floater[9]. The first step is to find the mean value weights at each node due to its neighbors. For a triangle side between nodes $n1$ and $n2$, the mean-value weight is given by

$$w_{n1,n2} = \frac{\tan(\alpha/2) + \tan(\beta/2)}{d_{n1,n2}} \tag{13}$$

where α is the angle at node $n1$ in the triangle to the left of side $[n1, n2]$, β is the angle at node $n1$ in the triangle to the right of side $[n1, n2]$, and $d_{n1,n2}$ is the distance between the nodes. All of these angle and distance calculations are made using the space (x, y, z) coordinates.

Next two sparse matrix equations are formed. For each interior node in the surface, we can write

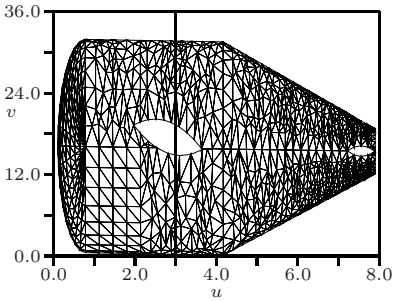
$$u_n \sum_m w_{m,n} + \sum_m (w_{m,n} u_m) = 0 \qquad (n = 1, \ldots, N) \tag{14}$$

$$v_n \sum_m w_{m,n} + \sum_m (w_{m,n} v_m) = 0 \qquad (n = 1, \ldots, N) \tag{15}$$

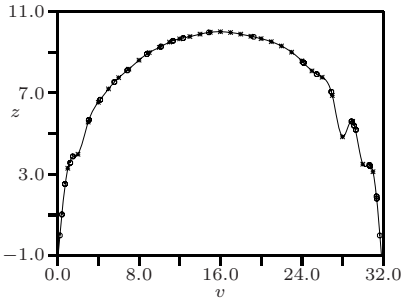
The parametric coordinates at boundary nodes are not allowed to move. These equations are solved using the sparse-matrix, bi-conjugate gradient technique mentioned above.

Fig. 4(a) show the tessellation that was provided; it was formed by projecting the nodes on a fuselage surface in the spanwise direction; that is $u_n = x_n$ and $v_n = y_n$. Note how the parameterization is squeezed near the top and bottom edges due to the fact that the surface is nearly parallel to the $x - z$ plane. A vertical cut was made through the parameterization at $u = 3$ and the corresponding spline fit of $z(v)$ is shown in part (b). Notice the oscillations that arise due to the nearly infinite slope, $\frac{dz}{dv}$, at the ends (which correspond to the top and bottom of the vertical cut).

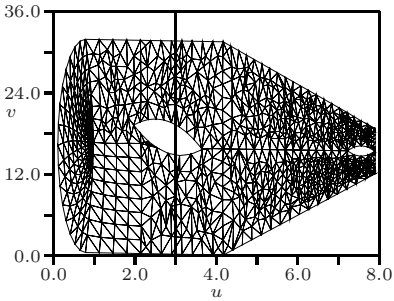
The above reparameterization technique was then applied, resulting in the parameterization in part (c) of the figure. Since the nodes on the surface were originally placed such that the spacings between adjacent nodes was approximately



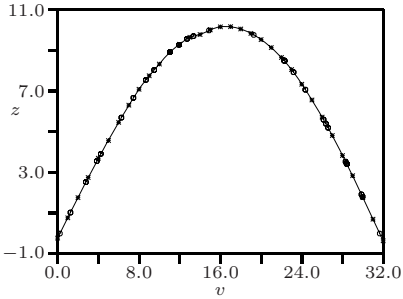
(a) original surface tessellation. The triangles near the bottom and top are highly stretched.



(b) spline approximation along $u = 3$ for the original surface tessellation.



(c) reparameterized surface tessellation. The triangles near the bottom and top are more isotropic than before.



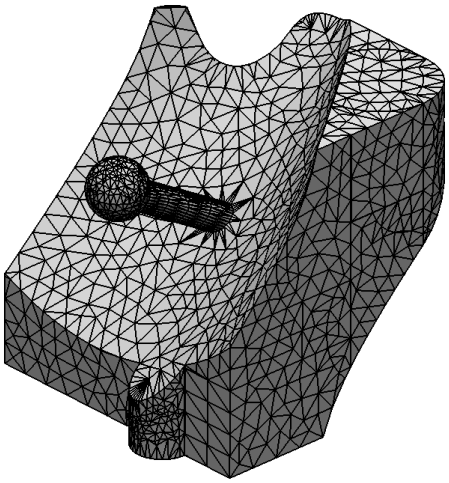
(d) improved spline approximation along $u = 3$ for the reparameterized tessellation

Fig. 4. Effect of reparameterizing a surface

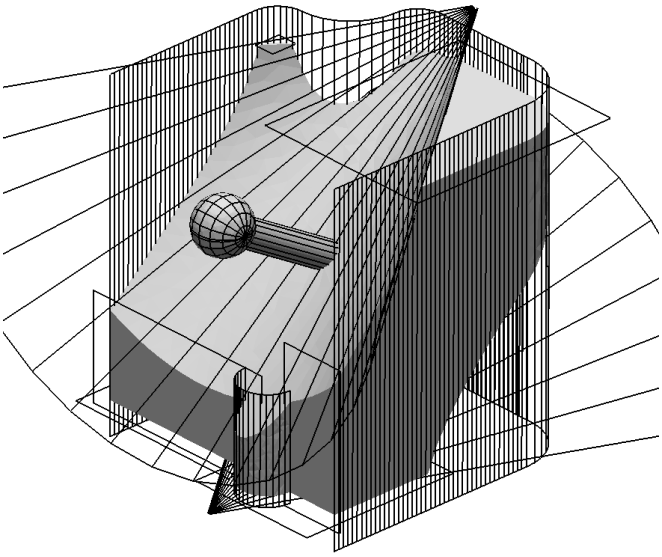
constant, the reparameterization results in approximately-constant parametric spacings. The spline technique applied on this new parameterization is shown in Fig. 4(d). Here the oscillations have been eliminated and the error between the resulting spline and the nodes is negligible. Application of this reparameterization to the entire test suite shows that all the oscillations were eliminated.

5 Computational Examples

The spline generation scheme proposed here has been applied to over 100 configurations, comprised of over 2500 surfaces. Figs. 5 and 6 show two of these configurations. The BRep produced by the CAD systems, along with CAPRI's[10] tessellation is shown in part (a) of each figure. Shown in part (b) of each figure is the grid of knots that were automatically produced by this scheme. One can clearly

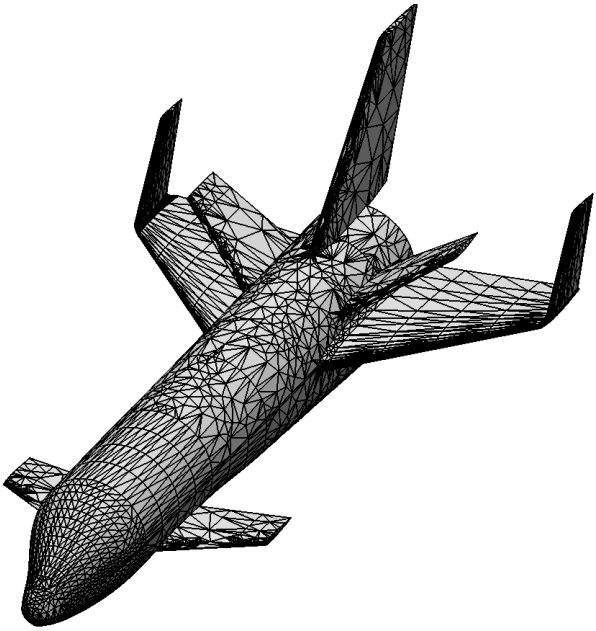


(a) BRep with given tessellation

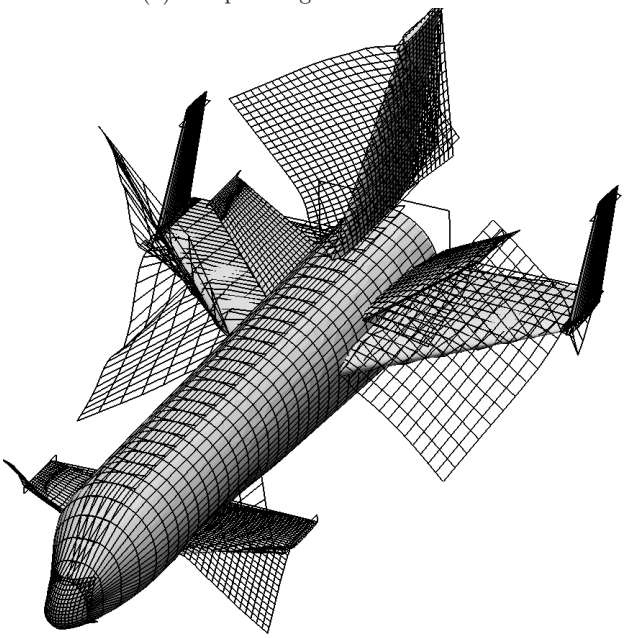


(b) BRep with grid of knots for each surface

Fig. 5. Test case 1, consisting of 17 surfaces (which were created by merging the original 82 faces in the Parasolid part). Note that the curved back surface is the surface used in Figs 1 to 3.



(a) BRep with given tessellation



(b) BRep with grid of knots for each surface

Fig. 6. Rocket launch vehicle, consisting of 40 surfaces (which were created by merging the original 83 faces in the Pro/ENGINEER part)

that most of the knots match the original configuration to within the specified tolerance, which was set to 10^{-4} times the configuration size. One can also see the extrapolation beyond the edges of the configuration that is necessary in order to have a rectangular grid of knots for each surface; this is especially apparent forward of the wing, canard, and empenage leading edges in Fig. 6. The total CPU time needed for the two cases was 4.3 and 49.9 seconds, respectively, on a MacBook-Pro with a 2.6 GHz Intel Core 2 Duo processor running under OSX 10.5.

6 Summary

A new process for generating a cubic spline surface that approximates, to within a user-specified tolerance, a given tessellated surface that may be non-convex or multiply-connected is described. The method combines a local least-squares technique for specifying knot properties as well as an adaptation technique of selecting the necessary knot spacings.

This new technique is fully described along curves, with all necessary equations. It is then extended to the case of the general surface; here the key equations are given together with an explanation as to how to derive the other equations by extending the curve method. Also, a reparameterization technique that is required for surfaces with poorly-defined parameterizations is described.

The new technique has been applied to over 100 configurations, composed of over 2500 surfaces. In all cases, the method was able to find spline approximations that were within the user-specified tolerance. Computed results for two configurations are shown.

The development of this technique was motivated by the need to create an approximation for surfaces that were created by merging other surfaces. However, this technique has wider applicability, such as to support retriangulation of surfaces, multi-disciplinary coupling, and behavior morphing.

Note that the development here used Ferguson splines as the approximating function, which are a subset of general NURBs. Therefore this technique provides a natural way for generating a NURB representation of a tessellated surface.

Acknowledgements

This work was funded by the NASA Fundamental Aeronautics Program under Cooperative Agreement NNX07AV29A. May-Fun Liou is the Technical Monitor.

References

1. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design, 2nd edn. Academic Press, London (1990)
2. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, Heidelberg (1997)
3. Jupp, D.L.B.: Approximation to Data by Splines with Free Knots. SIAM Journal of Numerical Analysis 15(2), 328–343 (1978)

4. Eilers, P.H.C., Marx, B.D.: Splines, Knots, and Penalties (unpublished, 2004), http://www.stat.lsu.edu/faculty/marx/splines_knots_penalties.pdf
5. Schwetlick, H., Schutze, T.: Least Squares Approximation by Splines with Free Knots. BIT Numerical Mathematics 35(3), 361–384 (1995)
6. Ruppert, D.: Selecting the Number of Knots for Penalized Splines. Journal of Computational and Graphical Statistics 11(4), 735–757 (2002)
7. Ferguson, D.: Construction of curves and surfaces using numerical optimization techniques. Computer Aided Design 18(1), 15–21 (1986)
8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in Fortran, 2nd edn., pp. 71–82. Cambridge University Press, Cambridge (1992)
9. Floater, M.S.: Mean value coordinates. Computer Aided Geometric Design 20, 19–27 (2003)
10. Haimes, R., Aftosmis, M.J.: Watertight Anisotropic Surface Meshing Using Quadrilateral Patches. In: 13th International Meshing Roundtable (2004), www.imr.sandia.gov/papers/imr13/haimes.pdf