Construction of B-spline Approximations for Use in Numerical Grid Generation

Bernd Hamann

NSF Engineering Research Center for Computational Field Simulation
Mississippi State University
P.O. Box 6176
Mississippi State, Mississippi 39762

and

Department of Computer Science
Mississippi State University
P.O. Drawer CS
Mississippi State, Mississippi 39762

Transmitted by Melvin Scott

ABSTRACT

Grid generation is concerned with discretizing surfaces and volumes in 3D space. The original surfaces defining a geometry might be given as a finite set of triangles/quadrilaterals (discrete form) or as parametrically defined surfaces (analytical form). A new interactive technique is presented for computing a B-spline approximation for geometries given in either form. The method requires user interaction for the selection of subsets of the given surfaces to be approximated. Once all subsets of surfaces have been approximated by B-spline surfaces, these are united yielding an overall C^2 continuous approximation.

1. INTRODUCTION

Grid generation is concerned with discretizing surfaces and volumes for computational field simulation (CFS). A 3D geometry might be given in discrete form (e.g., triangles and/or quadrilaterals) or parametric form (e.g., Bézier, B-spline, and Non-Uniform Rational B-spline (NURBS) surfaces). Grid generation methods construct grid points lying on the surfaces as well

655 Avenue of the Americas, New York, NY 10010 0096-3003/94/$7.00

as in volumes surrounding the surfaces defining a geometry. Unfortunately, the original surface description often contains anomalies, such as discontinuities between surface patches ("gaps") or intersections among surface patches. This paper presents an interactive technique for the correction of such anomalies. The grid generation system ("National Grid Project") currently being developed at the NSF Engineering Research Center for CFS at Mississippi State University uses the method presented.

The fundamental issues in grid generation are described in [1, 2]. Recent advances in grid generation are presented in [3]. The numerical methods required for the technique presented are primarily discussed in the geometric modeling and spline literature. References for these two areas include [4–9]. Most of the notations and algorithms used, in particular when dealing with B-splines, can be found in [4, 6].

The overall approach for creating a B-spline approximation of an arbitrary 3D geometry can be divided into these steps:

(i) Place a block in space.
(ii) Determine the subset of surfaces lying partly inside the block.
(iii) Compute a surface triangulation for this subset.
(iv) Clip the triangulation against the six faces of the block.
(v) Derive approximation conditions by intersecting the triangulation with a family of lines.
(vi) Map these intersections onto the parametric surfaces if the parametric definition is known.
(vii) If intersections can not be found for some lines, compute "artificial" conditions.
(viii) Use the points derived in (v), (vi), and (vii) to compute a local approximant.
(ix) Compute the error of the approximation.
(x) If the error is too large, increase the number of lines and go to (v).
(xi) If multiple surfaces must be approximated, repeat the steps (i)–(x).
(xii) Adjust all B-spline approximants in order to achieve an overall C^2 approximation.

These steps are described in detail in the following sections.

2. SURFACE SELECTION

The selection of subsets of all given surfaces is based on interactively placing blocks (hexahedra with "curved faces") around parts of a given geometry. These hexahedra are constructed as follows: First, the user spec-

ifies four points on the geometry. These four points define a quadrilateral in space. Second, two offset quadrilaterals are computed, one above and one below the user-defined quadrilateral. The offset distance is related to the edge lengths of the user-defined quadrilateral. The two offset surfaces define a block in space whose four side faces are obtained by connecting corresponding point pairs on the upper and lower offset surface. Such a block can be defined formally as follows:

DEFINITION 2.1. A *block* consists of eight vertices

$$\mathbf{v}_{i,j,k} = (x_{i,j,k}, y_{i,j,k}, z_{i,j,k}), \quad i,j,k \in \{0,1\}, \tag{2.1}$$

and twelve edges

$$\begin{aligned} &\overline{\mathbf{v}_{0,j,k}\mathbf{v}_{1,j,k}}, && j,k \in \{0,1\}, \\ &\overline{\mathbf{v}_{i,0,k}\mathbf{v}_{i,1,k}}, && i,k \in \{0,1\}, \quad \text{and} \\ &\overline{\mathbf{v}_{i,j,0}\mathbf{v}_{i,j,1}}, && i,j \in \{0,1\}. \end{aligned} \tag{2.2}$$

The six faces of the block are given by the six quadrilaterals defined by the six ordered (indicated by "<") point sets

$$\begin{aligned} &\{\mathbf{v}_{0,0,0}, \mathbf{v}_{0,0,1}, \mathbf{v}_{0,1,1}, \mathbf{v}_{0,1,0} \mid \mathbf{v}_{0,0,0} < \mathbf{v}_{0,0,1} < \mathbf{v}_{0,1,1} < \mathbf{v}_{0,1,0}\}, \\ &\{\mathbf{v}_{1,0,0}, \mathbf{v}_{1,1,0}, \mathbf{v}_{1,1,1}, \mathbf{v}_{1,0,1} \mid \mathbf{v}_{1,0,0} < \mathbf{v}_{1,1,0} < \mathbf{v}_{1,1,1} < \mathbf{v}_{1,0,1}\}, \\ &\{\mathbf{v}_{0,0,0}, \mathbf{v}_{1,0,0}, \mathbf{v}_{1,0,1}, \mathbf{v}_{0,0,1} \mid \mathbf{v}_{0,0,0} < \mathbf{v}_{1,0,0} < \mathbf{v}_{1,0,1} < \mathbf{v}_{0,0,1}\}, \\ &\{\mathbf{v}_{0,1,0}, \mathbf{v}_{0,1,1}, \mathbf{v}_{1,1,1}, \mathbf{v}_{1,1,0} \mid \mathbf{v}_{0,1,0} < \mathbf{v}_{0,1,1} < \mathbf{v}_{1,1,1} < \mathbf{v}_{1,1,0}\}, \\ &\{\mathbf{v}_{0,0,0}, \mathbf{v}_{0,1,0}, \mathbf{v}_{1,1,0}, \mathbf{v}_{1,0,0} \mid \mathbf{v}_{0,0,0} < \mathbf{v}_{0,1,0} < \mathbf{v}_{1,1,0} < \mathbf{v}_{1,0,0}\}, \quad \text{and} \\ &\{\mathbf{v}_{0,0,1}, \mathbf{v}_{1,0,1}, \mathbf{v}_{1,1,1}, \mathbf{v}_{0,1,1} \mid \mathbf{v}_{0,0,1} < \mathbf{v}_{1,0,1} < \mathbf{v}_{1,1,1} < \mathbf{v}_{0,1,1}\}. \end{aligned} \tag{2.3}$$

The coordinate extrema of the 3D bounding box containing the block are denoted by $X_{\min} = \min\{x_{i,j,k}\}$, $X_{\max} = \max\{x_{i,j,k}\}$, $Y_{\min} = \min\{y_{i,j,k}\}$, $Y_{\max} = \max\{y_{i,j,k}\}$, $Z_{\min} = \min\{z_{i,j,k}\}$, $Z_{\max} = \max\{z_{i,j,k}\}$.

The local approximation procedure is restricted to the interior of a block. Therefore, it must be defined when a point lies in the interior of a block. If the original surfaces are Bézier, B-spline, or NURBS surfaces, one can take advantage of their convex hull properties. These three surface types can contain points lying in the interior of a block only if the bounding box of their control nets and the bounding box of the block have a nonempty intersection. Let $\{\mathbf{d}_{i,j} = (d^x_{i,j}, d^y_{i,j}, d^z_{i,j}) \mid i = 0, \ldots, m, j = 0, \ldots, n\}$ denote the set of control points of a particular parametric surface satisfying the convex hull property, and let the coordinate extrema of a 3D bounding box containing all these control points be denoted by $x_{\min} = \min\{d^x_{i,j}\}, x_{\max}$

$= \max\{d^x_{i,j}\}, y_{\min} = \min\{d^x_{i,j}\}, y_{\max} = \max\{d^x_{i,j}\}, z_{\min} = \min\{d^x_{i,j}\}$, and $z_{\max} = \max\{d^x_{i,j}\}$.

A necessary condition for a surface to lie (partly) in the interior of a block is a nonempty intersection of the bounding box of the block and the bounding box of the surface's control points. This can be tested before any clipping of surfaces against the faces of a block is performed. Having determined the set of parametric surfaces that might lie (partly) in the interior of a block, these surfaces are evaluated for the generation of a surface triangulation. A criterion is defined next to decide whether a triangle in this triangulation has a nonempty intersection with the interior of a block.

DEFINITION 2.2. Let $x_i, i = 1, \ldots, 4$, be the (ordered) set of points defining a block face as defined in Definition 2.1. A point $x_0 = (x_0, y_0, z_0)$ is *on the negative side of the face*, if

$$\text{(i)} \quad (p_{1,2,3}(x_0, y_0, z_0) \leq 0 \quad \text{and} \quad p_{1,3,4}(x_0, y_0, z_0) \leq 0) \quad \text{or}$$

$$\text{(ii)} \quad (p_{1,2,4}(x_0, y_0, z_0) \leq 0 \quad \text{and} \quad p_{2,3,4}(x_0, y_0, z_0) \leq 0) \tag{2.4}$$

holds. The plane equation for the (oriented) plane containing the three points $\mathbf{x}_i, \mathbf{x}_j$, and $\mathbf{x}_k$ is $p_{i,j,k}(x, y, z) = a_{i,j,k}(x - x_i) + b_{i,j,k}(y - y_i) + c_{i,j,k}(z - z_i)$, and $\mathbf{n}_{i,j,k} = (a_{i,j,k}, b_{i,j,k}, c_{i,j,k})$ is the outward normal vector defined as $\mathbf{n}_{i,j,k} = (\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)$. If a point $\mathbf{x}_0$ is on the negative sides of all six block faces, it is called an *interior block point*.

THEOREM 2.1. *A triangle with vertices* $\mathbf{v}_1$, $\mathbf{v}_2$, *and* $\mathbf{v}_3$ *cannot contain an interior block point if all its vertices are on the negative side of one block face (all three vertices satisfying* (i) *or all the three vertices satisfying* (ii) *in Definition 2.2) and are on the nonnegative side of the opposite block face (all three vertices violating* (i) *and all three vertices violating* (ii) *in Definition 2.2).*

PROOF. Each point $\mathbf{x}$ in the interior (or on one of the edges) of a triangle can be written as a convex combination $\mathbf{x} = \sum_{r=1}^{3} u_r \mathbf{v}_r$, where $\sum_{r=1}^{3} u_r = 1, u_r \geq 0, r = 1, 2, 3$, and the intersection of the two half-spaces implied by the pairs of plane equations is convex. Therefore, $\mathbf{x}$ lies on the negative side of the same block face and on the nonnegative side of the opposite block face. □

Theorem 2.1 uses a generalization of a 3D bounding box ("cuboid")

to a hexahedron having quadrilaterals as its faces. This theorem is used for the reduction of the number of triangles potentially containing interior block points. All triangles violating the condition stated in Theorem 2.1 are kept and are used for the creation of the local surface approximation.

3. CONDITIONS FOR THE APPROXIMATION

Once the set of all surface triangles lying (partly) in the interior of a block is known, the associated surfaces are locally approximated using a single B-spline surface. In order to determine the control points for this B-spline approximant, one computes a finite point set on the surfaces lying (partly) in the interior of the block and uses these points as approximation conditions. These points are obtained by intersecting lines with the surface triangulation in the interior of a block. The lines themselves are defined in terms of pairs of points lying on the "bottom" and "top" face of the block. This procedure is described next.

Two bilinearly blended surfaces are implied by the two sets of four block face vertices $\{\mathbf{v}_{i,j,0} \mid i,j \in \{0,1\}\}$ ("bottom") and $\{\mathbf{v}_{i,j,1} \mid i,j \in \{0,1\}\}$ ("top"). The two bilinearly blended surfaces are

$$\begin{aligned} \mathbf{s}_k(u,v) &= (1-u)(1-v)\mathbf{v}_{0,0,k} + u(1-v)\mathbf{v}_{1,0,k} + (1-u)u\mathbf{v}_{0,1,k} \\ &\quad + uv\mathbf{v}_{1,1,k} \quad u,v \in [0,1], \qquad k \in \{0,1\}. \end{aligned} \tag{3.1}$$

Evaluating $\mathbf{s}_0$ and $\mathbf{s}_1$ at parameter values (u_I, v_J), $u_I = I/M, I = 0, \ldots, M$, and $v_J = J/N$, $J = 0, \ldots, N$, yields point pairs defining the lines

$$\begin{aligned} \mathbf{l}_{I,J} &= \mathbf{s}_1(u_I, v_J) + t(\mathbf{s}_0(u_I, v_J) - \mathbf{s}_1(u_I, v_J)), \\ &\qquad t \in \mathbb{R}, \qquad I = 0, \ldots, M, \qquad J = 0, \ldots, N. \end{aligned} \tag{3.2}$$

These lines are used to obtain approximation conditions.

The set of triangles lying (partly) in the interior of a block is denoted by

$$\begin{aligned} \mathcal{T} &= \big\{\big(\mathbf{v}_1^i = (x_1^i, y_1^i, z_1^i), \mathbf{v}_2^i = (x_2^i, y_2^i, z_2^i), \\ &\qquad \mathbf{v}_3^i = (x_3^i, y_3^i, z_3^i)\big) \mid i = 1, \ldots, N_t\big\}. \end{aligned} \tag{3.3}$$

Each triangle is contained in a plane given by

$$p^i(x,y,z) = a^i(x - x_1^i) + b^i(y - y_1^i) + c^i(z - z_1^i) = 0, \tag{3.4}$$

where $\mathbf{n}^i = (a^i, b^i, c^i)$ is the plane normal vector $\mathbf{n}^i = (\mathbf{v}_2^i - \mathbf{v}_1^i) \times (\mathbf{v}_3^i - \mathbf{v}_1^i)$.

Each line $\mathbf{l}_{I,J}$ is intersected with each plane p^i. When an intersection is found, it is determined whether the intersection points $\mathbf{y}^i$ is in the interior (or on the boundary) of some triangle (contained in plane p^i) by computing the value

$$\omega = \arccos\left(\frac{\mathbf{a}\times\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right) + \arccos\left(\frac{\mathbf{b}\times\mathbf{c}}{\|\mathbf{b}\|\|\mathbf{c}\|}\right) + \arccos\left(\frac{\mathbf{c}\times\mathbf{a}}{\|\mathbf{c}\|\|\mathbf{a}\|}\right), \quad (3.5)$$

where $\mathbf{a} = \mathbf{v}_1^i - \mathbf{y}^i$, $\mathbf{b} = \mathbf{v}_2^i - \mathbf{y}^i$, $\mathbf{c} = \mathbf{v}_3^i - \mathbf{y}^i$, and $\|\|$ indicates the Euclidean norm. If $\omega = 2\pi$ the intersection point $\mathbf{y}^i$ is a point in the interior (or on the boundary) of this particular triangle, otherwise, it is outside. Special care is required when the line $\mathbf{l}_{I,J}$ is parallel to or contained in a plane.

A line $\mathbf{l}_{I,J}$ might not intersect any triangle at all, might intersect several triangles, or might be contained in the plane containing a triangle. Among all intersections in the interior (or on the boundary) of triangles, one selects the intersection that is closest to $\mathbf{s}_1(u_I, v_J)$, closest to the "top" face of the block. If a line is contained in the plane defined by a triangle, one computes where the line $\mathbf{l}_{I,J}$ intersects the triangle's edges and selects the point closest to $\mathbf{s}_1(u_I, v_J)$. The selected intersection point is denoted by $\mathbf{x}_{I,J}$ as in Figure 1.

Some lines might not intersect any triangles. In this case, "artificial" approximation conditions are derived using the following approach: Each intersection point $\mathbf{x}_{I,J}$ that has been found can be written as a linear combination of the two points $\mathbf{s}_0(u_I, v_J)$ and $\mathbf{s}_1(u_I, v_J)$, i.e.

$$\mathbf{x}_{I,J} = (1 - t_{I,J})\mathbf{s}_1(u_I, v_J) + t_{I,J}\mathbf{s}_0(u_I, v_J), \qquad t_{I,J} \in \mathbb{R}. \quad (3.6)$$

A bivariate approximation technique can be used to compute parameter values $t_{I,J}$ for lines without intersection points.

Hardy's reciprocal multiquadric is used to solve this problem (see [10]). The bivariate interpolation conditions are

$$t_{I,J} = t(u_I, v_J) = \sum_{j\in\{0,\ldots,N\}} \sum_{i\in\{0,\ldots,M\}} c_{i,j}\left(R + (u_I - u_i)^2 + (v_J - v_j)^2\right)^{-\gamma},$$
$$I \in \{0,\ldots,M\}, \qquad J \in \{0,\ldots,N\}, \quad (3.7)$$

where one uses values $t_{I,J}, u_I, u_i, v_J$, and v_j for which an intersection point $\mathbf{x}_{I,J}$ is known. For an equidistant parametrization (i.e., $u_{i+1} - u_i = \delta_u$ and $v_{j+1} - v_j = \delta_v$), the value $R = 0.5(\delta_u + \delta_v)$ generally yields good results. Using the value $\gamma = 0.001$ in Hardy's reciprocal multiquadric works well in most practical cases tested.

The coefficients $c_{i,j}$ appearing in (3.7) are computed by solving the implied linear system of equations. Additional parameter values $t_{I,J}$ can be

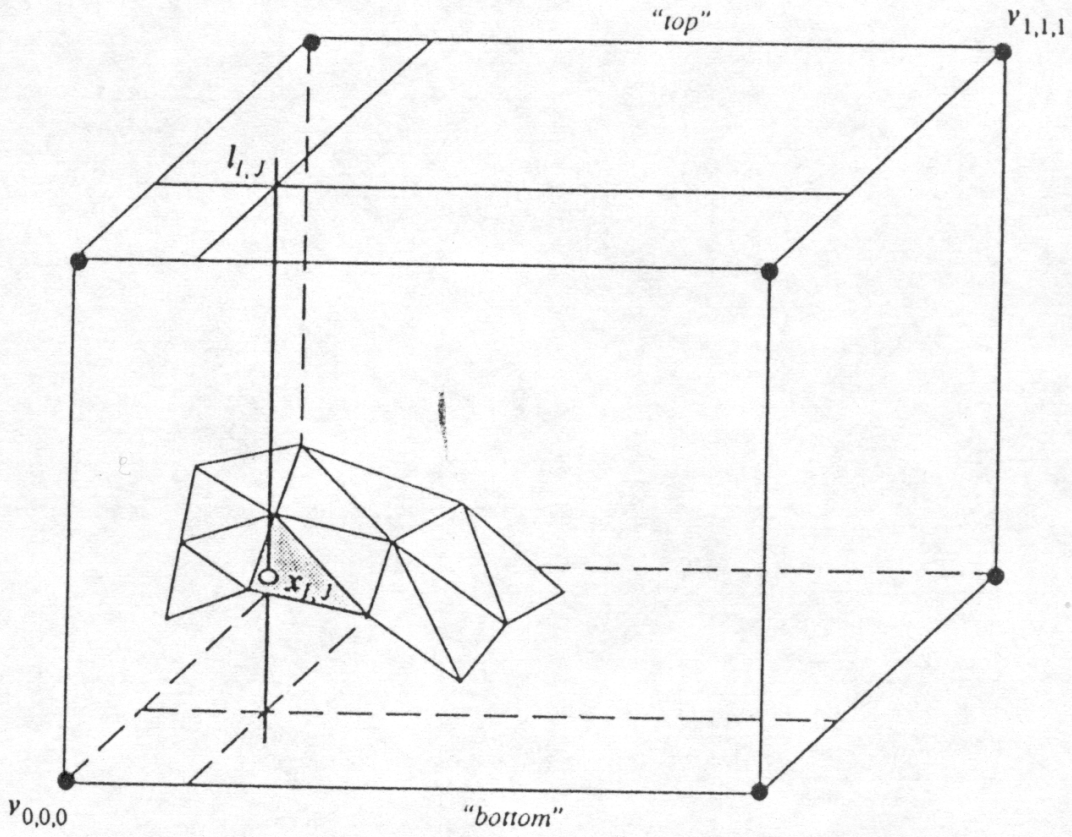

FIG. 1. Obtaining approximation conditions.

computed by simply evaluating (3.7) for $(u, v) = (u_I, v_J)$. The bivariate interpolation problem can be made more efficient and localized by considering only a certain number of known $t_{I,J}$ values "around" a line without intersection points. The final approximation condition for a line $\mathbf{l}_{I,J}$ is obtained by evaluating (3.6) for the additionally computed values $t_{I,J}$. By following this sequence for deriving approximation conditions, one guarantees $(M+1)(N+1)$ approximation conditions. They are used to determine the B-spline approximation for all the surfaces lying (partly) in a block.

Each intersection point $\mathbf{x}_{I,J}$ lying in the interior (or on the boundary) of some triangle formed by three surface vertices is mapped onto the associated surface, provided its parametric definition is known. This is achieved by expressing the intersection point in terms of barycentric coordinates, using these barycentric coordinates to get a parameter tuple in the surface's domain, and computing a point on the surface. Writing the intersection point $\mathbf{x}_{I,J}$ as

$$\mathbf{x}_{I,J} = \overline{u}_1 \mathbf{v}_1 + \overline{u}_2 \mathbf{v}_2 + \overline{u}_3 \mathbf{v}_3, \tag{3.8}$$

where $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$ are the vertices of the triangle containing $\mathbf{x}_{I,J}$, one

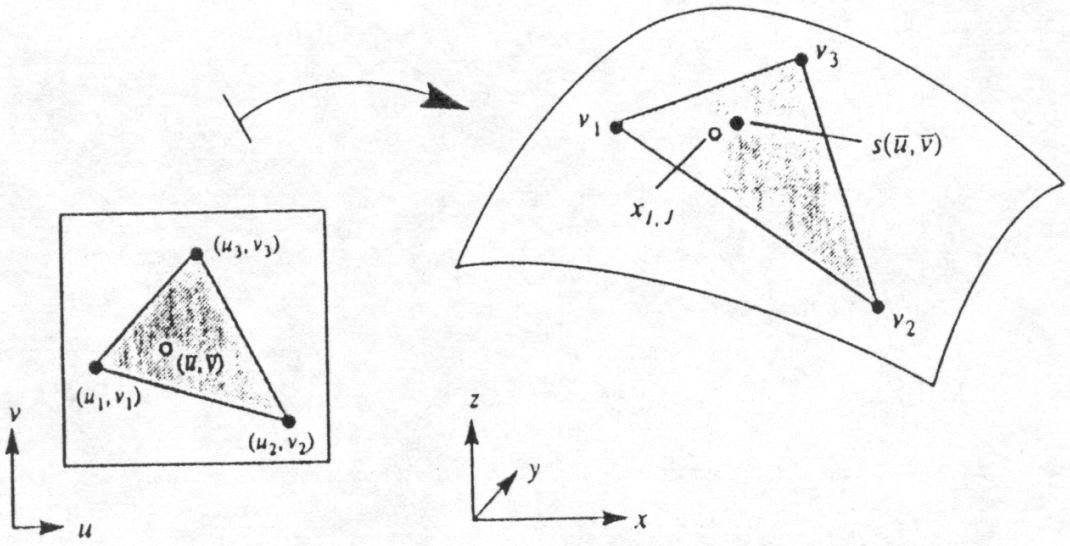

FIG. 2. Computing point on surface using barycentric coordinates.

computes the parameter tuple

$$(\overline{u}, \overline{v}) = \overline{u}_1(u_1, v_1) + \overline{u}_2(u_2, v_2) + \overline{u}_3(u_3, v_3), \tag{3.9}$$

where $(u_k, v_k), k = 1, 2, 3$, is the parameter tuple of vertex $\mathbf{v}_k$. The surface $\mathbf{s}$ containing $\mathbf{v}_k, k = 1, 2, 3$, is evaluated at $(\overline{u}, \overline{v})$, and $\mathbf{x}_{I,J}$ is replaced by $\mathbf{s}(\overline{u}, \overline{v})$ using it as the final approximation condition. This principle is illustrated in Figure 2.

Obviously, the set of surface points generated in this process depends on the orientation of the block. This is due to the fact that the surface points are obtained by intersecting line segments defined by corresponding point pairs on the "top" and "bottom" face of the block with the given geometry. changing the orientation of the block leads to different line segments and therefore to a different set of intersections with the geometry.

4. B-SPLINE APPROXIMATION

The approximation conditions derived in the previous steps are used to construct a locally approximating B-spline surface. The definition of a B-spline surface is as follows:

DEFINITION 4.1. A B-*spline surface* $\mathbf{s}(u,v)$ is a piecewise polynomial surface, denoted by

$$\mathbf{s}(u,v) = \sum_{j=0}^{n}\sum_{i=0}^{m} \mathbf{d}_{i,j} N_{i,k}(u) N_{j,l}(v),$$
$$u \in [u_{k-1}, u_{m+1}], \qquad v \in [v_{l-1}, v_{n+1}], \tag{4.1}$$

and defined by

- two orders k and l,
- a set of 3D control points, $\{\mathbf{d}_{0,0}, \ldots, \mathbf{d}_{m,n}\}$,
- a set of real u knots, $\{u_0, \ldots, u_{m+k} \mid u_i \le u_{i+1}, i = 0, \ldots, (m+k-1)\}$,
- a set of real v knots, $\{v_0, \ldots, v_{n+1} \mid v_j \le v_{j+1}, j = 0, \ldots, (n+l-1)\}$,
- B-spline basis functions $N_{i,k}(u), u \in [u_i, u_{i+k}], i = 0, \ldots, m$, where

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u),$$
$$N_{i,1}(u) = \begin{cases} 1, & \text{if } u_i \le u < u_{i+1}, \\ 0, & \text{otherwise,} \end{cases} \qquad i = 0, \ldots, m, \tag{4.2}$$

- and B-spline basis functions $N_{j,l}(v), v \in [v_j, v_{j+l}], j = 0, \ldots, n$, defined analogously to $N_{i,k}(u)$.

The curves $\mathbf{c}_0(v) = \mathbf{s}(u_{k-1}, v), \mathbf{c}_1(v) = \mathbf{s}(u_{m+1}, v), v \in [v_{l-1}, v_{n+1}]$, and $\bar{\mathbf{c}}_0(u) = \mathbf{s}(u, v_{l-1})$ and $\bar{\mathbf{c}}_1(u) = \mathbf{s}(u, v_{n+1}), u \in [u_{k-1}, u_{m+1}]$, are called *boundary curves*, and the points $\mathbf{x}_{0,0} = \mathbf{s}(u_{k-1}, v_{l-1}), \mathbf{x}_{1,0} = \mathbf{s}(u_{m+1}, v_{l-1})$, $\mathbf{x}_{0,1} = \mathbf{s}(u_{k-1}, v_{n+1})$, and $\mathbf{x}_{1,1} = \mathbf{s}(u_{m+1}, v_{n+1})$ are called *corner points* of the B-spline surface $\mathbf{s}(u,v)$.

Using this definition, one easily derives the system of linear equations for interpolating the set of points $\{\mathbf{x}_{I,J} \mid I = 0, \ldots, M, J = 0, \ldots, N\}$. The system is given by

$$\mathbf{x}_{I,J} = \mathbf{s}(\bar{u}_I, \bar{v}_J) = \sum_{j=0}^{n}\sum_{i=0}^{m} \mathbf{d}_{i,j} N_{i,k}(\bar{u}_I) N_{j,l}(v_J),$$
$$I = 0, \ldots, M, \quad J = 0, \ldots, N. \tag{4.3}$$

Using a piecewise bicubic approach ($k = l = 4$), and interpolating at the knots, i.e., $\bar{u}_I = u_{k-1+I} = u_{3+I}, I = 0, \ldots, M$, and $\bar{v}_J = v_{l-1+J} =$

v_{3+J}, $J = 0, \ldots, N$, the system of equations becomes

$$\mathbf{x}_{I,J} = \mathbf{s}(u_{3+I}, v_{3+J}) = \sum_{j=0}^{N+2} \sum_{i=0}^{M+2} \mathbf{d}_{i,j} N_{i,4}(u_{3+I}), N_{j,4}(v_{3+J}),$$
$$I = 0, \ldots, M, \qquad J = 0, \ldots, N. \tag{4.4}$$

One can use an equidistant knot distribution, i.e., $u_i = i/(M+6)$, $i = 0, \ldots, (M+6)$, and $v_j = j/(N+6)$, $j = 0, \ldots, (N+6)$, or a knot distribution considering the Euclidean distances of the points to be interpolated as in Figure 3.

According to [6], one solves the (underdetermined) system (4.4) by first interpolating $(N+1)$ rows of points given by the sets $\{\mathbf{x}_I = \mathbf{x}_{I,J} \mid I = 0, \ldots, M\}_{J=0}^{N}$ using certain end conditions (e.g., natural, clamped, or Bessel). The results of this step are $(N+1)$ rows of "intermediate" control points, denoted by $\{\bar{\mathbf{d}}_{i,J} \mid i = 0, \ldots, (M+2)\}_{J=0}^{N}$. Second, one interpolates the $(M+3)$ columns of "intermediate" control points, given by the sets $\{\bar{\mathbf{d}}_j = \bar{d}_{i,J} \mid j = 0, \ldots, N\}_{i=0}^{M+2}$. This results in the set of the desired control points for a tensor product B-spline surface, $\{\mathbf{d}_{i,j} \mid i = 0, \ldots, (M+2), j = 0, \ldots, (N+2)\}$.

5. REPRESENTING THE TOPOLOGY OF THE OVERALL APPROXIMATION

During the process of constructing the approximating B-spline surfaces, one must determine their topological connectivities and store them. For all following processing steps, it is essential to know which boundary curves of a particular B-spline surface are shared by which other B-spline surfaces. Since all the surfaces created are topologically four-sided entities, they can have a maximum of four neighbors. Our application is restricted to particular geometries, which are defined next.

DEFINITION 5.1. *A connected, knot-to-knot B-spline surface geometry* is a finite set of at least two B-spline surfaces satisfying the following conditions:

(i) Each boundary curve of a B-spline surface is shared by at most two surfaces (no bifurcations).
(ii) A corner point of a B-spline surface can be shared by any number of surfaces.

FIG. 3. Information involved in B-spline approximation.

(iii) Each surface in the set of all B-spline surfaces has at least one point along one of its boundary curves in common with another surface (connectivity).
(iv) If a corner point of a B-spline surface is shared by a second B-spline surface, this point is also a corner point of the second surface (knot-to-knot property).

In grid generation, such geometries are referred to as "full-face interface" geometries.

DEFINITION 5.2. Let $\mathbf{s}(u, v)$ be a B-spline surface in a connected, knot-to-knot B-spline surface geometry. The *left neighbor* of $\mathbf{s}(u, v)$ is the B-spline surface sharing the curve $\mathbf{s}(u_{k-1}, v)$, the *right neighbor* is the B-spline surface sharing the curve $\mathbf{s}(u_{m+1}, v)$ the *bottom neighbor* is the B-spline surface sharing the curve $\mathbf{s}(u, v_{l-1})$, and the *top neighbor* is the B-spline surface sharing the curve $\mathbf{s}(u, v_{n+1})$.

For each local B-spline approximant, at most four neighbors are identified and stored.

6. ERROR ESTIMATION

In order to evaluate the quality of the approximation, one needs an error measure determining the difference between the subset of approximated surfaces and a single approximating B-spline surface. A discrete measure is used for the computation of the difference between surfaces. Assuming that one knows the exact definition (order, knots, and control points) of the subset of surfaces being approximated, one can compute their distances to the approximating B-spline surface.

First, points on the approximating B-spline surface $\mathbf{s}^{\mathrm{app}}(u, v)$ are generated. More specifically, one computes a finite point set $\mathcal{P} = \{\mathbf{x}_j = \mathbf{s}^{\mathrm{app}}(u_j, v_j) = (x_j, y_j, z_j) \mid j = 1, \ldots, K\}$. Since one knows the triangulation of the subset of surfaces being approximated, one can determine the shortest distance between a point $\mathbf{x}_j$ and the triangles in the triangulation. The distance between $\mathbf{x}_j$ and a triangle with vertices $\mathbf{v}_1^i$, $\mathbf{v}_2^i$, and $\mathbf{v}_3^i$ is defined as $\sum_{k=1}^{3} \|\mathbf{x}_j - \mathbf{v}_k^i\|$. The triangle for which this expression is minimal is identified. Its vertices are denoted by $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$. The plane containing this triangle is given by

$$Ax + By + Cz + D = 0, \tag{6.1}$$

where (A, B, C) is the unit normal vector $\mathbf{n}$ of the triangle,

$$\mathbf{n} = (A, B, C) = \frac{(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})}{\|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})\|}, \tag{6.2}$$

and $D = -(\mathbf{n} \cdot \mathbf{a})$. The (signed) perpendicular distance d_j between the plane (6.1) and the point $\mathbf{x}_j$ is given by

$$d_j = \mathbf{n} \times \mathbf{x}_j + D. \tag{6.3}$$

Since d_j is the distance of the point $\mathbf{y}_j$ in a plane (6.1) having shortest distance to $\mathbf{x}_j$, this point is given by

$$\mathbf{y}_j = \mathbf{x}_j - d_j \mathbf{n}. \tag{6.4}$$

The point $\mathbf{y}_j$ is expressed in terms of barycentric coordinates,

$$\mathbf{y}_j = \bar{u}_1 \mathbf{a} + \bar{u}_2 \mathbf{b} + \bar{u}_3 \mathbf{c}, \tag{6.5}$$

and the triple $(\bar{u}_1, \bar{u}_2, u_3)$ is used to compute a point on the corresponding parametric surface. Knowing the parameter tuples (u_1, v_1), (u_2, v_2), and (u_3, v_3) associated with the vertices a, b, and c, one computes the tuple

$$(\bar{u}, \bar{v}) = \bar{u}_1 (u_1, v_1) + \bar{u}_2 (u_2, v_2) + \bar{u}_3 (u_3, v_3) \tag{6.6}$$

and uses it for the generation of the point $\mathbf{s}_j = \mathbf{s}(\bar{u}, \bar{v})$ lying on the same original surface as the vertices a, b, and c. Therefore, the (approximate) distance D_j between the point $\mathbf{x}_j$ and $\mathbf{s}_j$, is given by

$$D_j = \sqrt{(\mathbf{x}_j - \mathbf{s}_j)(\mathbf{x}_j - \mathbf{s}_j)}. \tag{6.7}$$

The root-mean-square (RMS) error E is defined as

$$E = \sqrt{\frac{1}{K} \sum_{j=1}^{K} D_j^2}, \tag{6.8}$$

and it is used as a measure for the distance between the approximating B-spline surface and the subset of approximated surfaces.

If the error E exceeds a prescribed tolerance, the number of approximation conditions used in the approximation is increased until the error is smaller than the tolerance. It is necessary to use the same number of

approximation conditions for the computation of neighboring B-spline surfaces. It is essential that each pair of neighbors has $(M+3)(N+3)$ control points (see (4.4)). This is required for the adjustment of the approximating B-spline surfaces' control points to define an overall C^2 approximation.

7. UNITING B-SPLINE SURFACES

The process of approximating a subset of surfaces by a single B-spline surface has been explained in the previous sections. In the context of approximating an entire aircraft, car body, or ship hull, it is necessary to compute several local B-spline approximants. Once computed, they must be adjusted such that each pair of neighbor B-spline surfaces satisfies continuity conditions along the shared boundary curve. Furthermore, it must be taken care of continuity conditions at surface corner points where multiple B-spline surfaces come together.

Considering only connected, knot-to-knot B-spline surface geometries (Definition 5.1), a geometry is constructed such that neighbor B-spline surfaces satisfy certain continuity conditions along their common boundary curve. In the following, it is described how to adjust the control information (control points and knots) of piecewise bicubic B-spline surfaces ($k = l = 4$) in order to achieve C^2 continuity. The principle can easily be generalized to higher-order continuity and surfaces of arbitrary order.

THEOREM 7.1. *Two B-spline surfaces* $\mathbf{s}(u,v) = \sum_{j=0}^{n} \sum_{i=0}^{m} \mathbf{d}_{i,j}\, N_{i,4}(u) N_{j,4}(v)$, $u \in [u_3, u_{m+1}]$, $v \in [v_3, v_{n+1}]$, *and* $\bar{\mathbf{s}}(\bar{u}, \bar{v}) = \sum_{j=0}^{n} \sum_{i=0}^{\bar{m}} \bar{\mathbf{d}}_{i,j}\, N_{i,4}(\bar{u}) N_{j,4}(\bar{v})$, $\bar{u} \in [\bar{u}_3, \bar{u}_{\bar{m}+1}]$, $\bar{v} \in [\bar{v}_3, \bar{v}_{n+1}]$, *are* C^2 *continuous along* $u = u_{m+1}$ $(\bar{u} = \bar{u}_3)$, *if their control points and knots satisfy the following conditions:*

$$\begin{aligned} \mathbf{d}_{m-2+I,j} &= \bar{\mathbf{d}}_{I,j}, \qquad I = 0,\ldots,2, \qquad j = 0,\ldots,n, \\ \Delta u_{m-2+I} &= \Delta \bar{u}_I, \qquad I = 0,\ldots,5, \qquad \text{and} \\ \Delta v_j &= \Delta \bar{v}_j, \qquad j = 0,\ldots,(n+3), \end{aligned} \tag{7.1}$$

where $\Delta u_i = u_{i+1} - u_i$ *and* $\Delta v_j = v_{j+1} - v_j$.

PROOF. See [5, 6]. □

This theorem is the basis for adjusting the control points and knots for pairs of B-spline surfaces that must be C^2 continuous along a common boundary curve. Considering connected, knot-to-knot B-spline sur-

face geometries implies that any number of B-spline surfaces can share a common surface corner point. It is necessary to ensure continuity at such common surface corner points as well. It turns out that it is possible to satisfy this additional condition during the process of enforcing C^2 continuity along common boundary curves. It is also possible to enforce continuity along all shared surface boundary curves in a first and continuity at all shared surface corner points in a second processing step.

The algorithm for the adjustment of control information is straightforward. Each control point $\mathbf{d}_{i,j}$ of each B-spline surface is assigned an "adjustment counter" $c_{i,j}$ indicating how many control points have been averaged for its computation. Initially, these counters are set to "1" for all control points of all B-spline surfaces. The following pseudo-code represents the algorithm used for adjusting control points and knots in order to obtain a pair of surfaces that is C^2 continuous along a shared boundary curve. It must be mentioned that all surfaces must be cubic B-spline surfaces having at least six rows and six columns of control points.

ALGORITHM 1. *Adjusting control points and knots.*

Input: Two cubic B-spline surfaces (control points and knots), $\mathbf{s}(u,v)$ and $\bar{\mathbf{s}}(\bar{u},\bar{v})$

Output: Two adjusted cubic B-spline surfaces (control points and knots), C^2 continuous along common boundary curve $\mathbf{c}(v) = \bar{\mathbf{c}}_0(\bar{v})$ (see Definition 4.1)

{

/* Adjusting control points */

$\mathbf{d}_{m-2+I,j} = (c_{m-2+I,j}\mathbf{d}_{m-2+I,j} + \bar{c}_{I,j}\overline{\mathbf{d}}_{I,j})/(c_{m-2+I,j} + \bar{c}_{I,j});$ $(I = 0,1,2, \quad j = 0,\ldots,n)$

$\overline{\mathbf{d}}_{I,j} = \mathbf{d}_{m-2+I,j};$ $(I = 0,1,2, \quad j = 0,\ldots,n)$

$c_{m-2+I,j} = c_{m-2+I,j} + \bar{c}_{I,j};$ $(I = 0,1,2, \quad j = 0,\ldots,n)$

$\bar{c}_{I,j} = c_{m-2+I};$ $(I = 0,1,2, \quad j = 0,\ldots,n)$

/* Adjusting knots */

$\delta_I = (\Delta u_{m-2+I} + \Delta\bar{u}_I)/2;$ $(I = 0,\ldots,5)$

$u_{m-2+I} = u_{m-2} + \sum_{r=0}^{I-1}\delta_r;$ $(I = 1,\ldots,6)$

$\bar{u}_I = \bar{u}_0 + \sum_{r=0}^{I-1}\delta_r;$ $(I = 1,\ldots,6)$

$\epsilon_I = (\Delta v_j + \Delta\bar{v}_j)/2;$ $(j = 0,\ldots,(n+3))$

$v_j = (v_0 + \bar{v}_0)/2 + \sum_{r=0}^{j-1}\epsilon_r;$ $(j = 0,\ldots,(n+4))$

$\bar{v}_j = v_j;$ $(j = 0,\ldots,(n+4))$

}

REMARK. In the implementation, $\mathbf{d}_{m-2+l,j}$, and $\bar{\mathbf{d}}_{l,j}$ represent pointers to the corresponding control point coordinate vectors. These pointers must be updated throughout the algorithm to ensure C^2 continuity at common surface corner points.

This algorithm averages control points along boundary curves and the lengths of knot intervals at the ends of parameter domains. In order to be applicable to all four boundary curves of a surface, the control points and knots used in the adjustment must be selected in accordance with the particular boundary curve. The scheme applies to any number of surfaces sharing a common corner point. Figure 4, shows the case of three surfaces sharing a common corner point. The control point of surface $\mathbf{s}_k$ are denoted by $\mathbf{d}^k_{i,j}$.

The adjustment algorithm ensures C^2 continuity at each surface corner point. B-spline control points "around" a common surface corner point are the average of the control points of all the B-spline surfaces coming together at the common corner point.

8. EXAMPLES

Figure 5 shows an approximation for two intersecting surfaces. The original surfaces are shaded darker than the approximating surface.

Figure 6 shows a real-world car body configuration containing both "gaps" and intersecting surfaces. The original geometry is shown in the top portion of the figure and the continuous approximation is shown in the bottom portion.

9. CONCLUSIONS

An interactive technique for approximating surfaces by a set of B-spline surfaces has been described. The approximation process is semi-automatic in the sense that the user still has to specify subsets of surfaces by picking four appropriate surface points. Once these subsets have been determined, they are approximated without any further user interaction. Having created all approximating B-spline surfaces, C^2 continuity is automatically enforced along all surface boundary curves and at all surface corner points.

This approximation strategy has been designed for the particular needs of grid generation. Given the CAD definition of a 3D geometry, the set of surfaces defining the geometry might contain anomalies, such as discontinu-

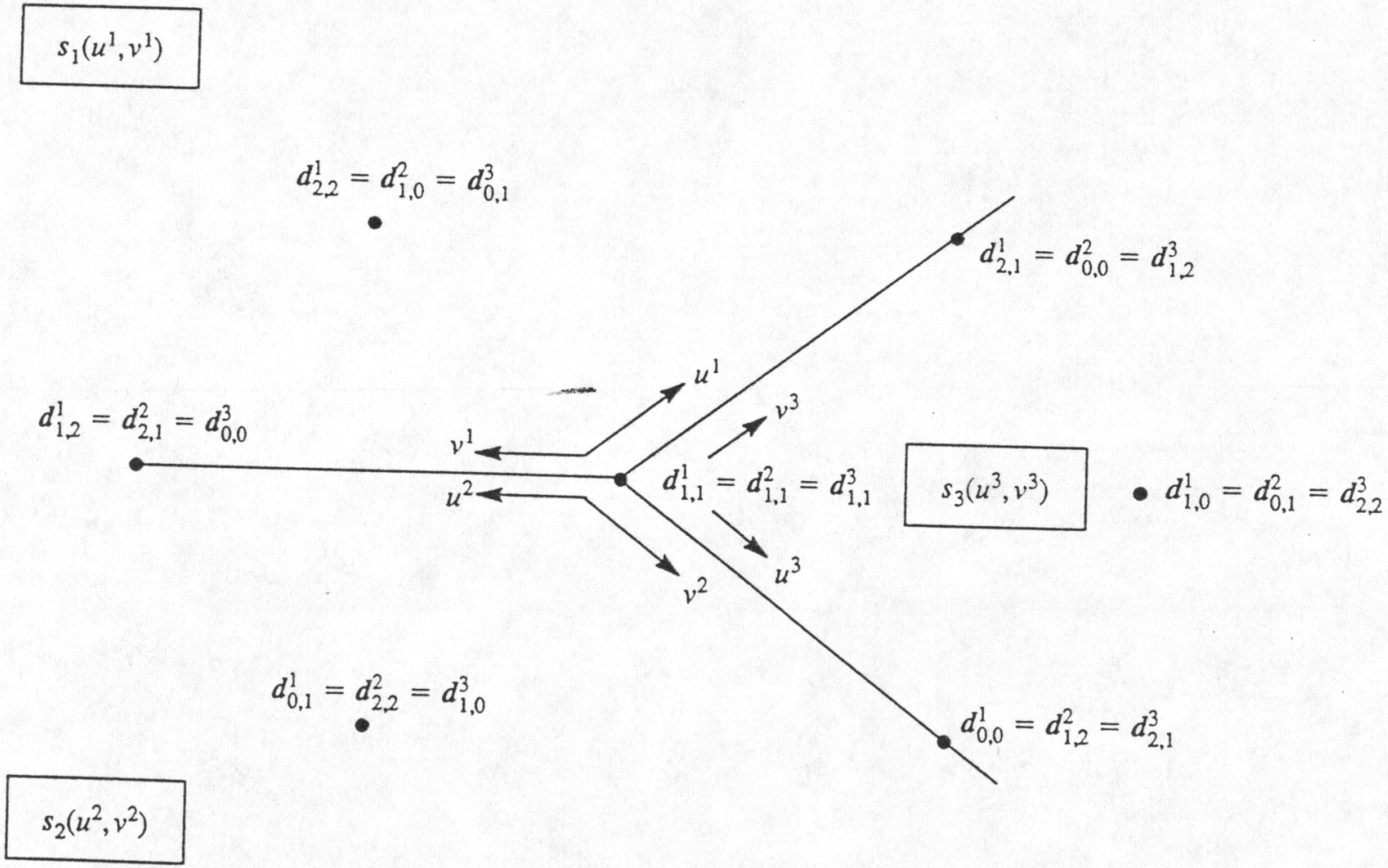

FIG. 4. Control point equalities for three surfaces sharing common corner point.

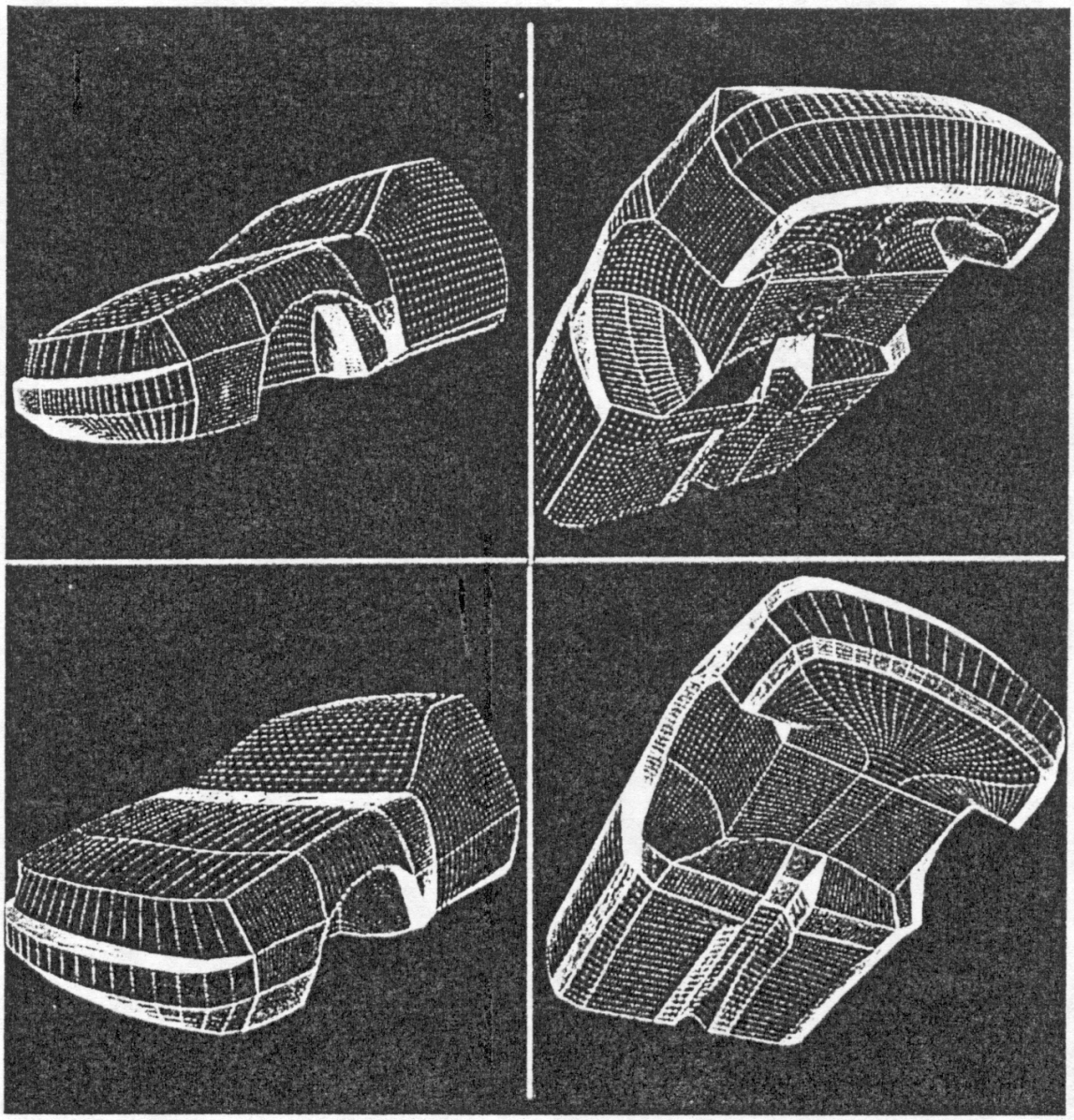

FIG. 6. Approximation of car body (top: original, bottom, approximation.)

Thanks go to all the members on the National Grid Project team, in particular to the people who have implemented and tested the algorithm, Ming-Laing Chen, J. Adam Gaither, Brian A. Jean, Shekhar Mahadevan, Kelly L. Parmley, and Po-Yu Tsai. John F. Dannenhoffer III, C. Wayne Mastin, Bharat K. Soni, Joe F. Thompson, and Nigel P. Weatherill have contributed significantly to this paper. This is gratefully acknowledged.

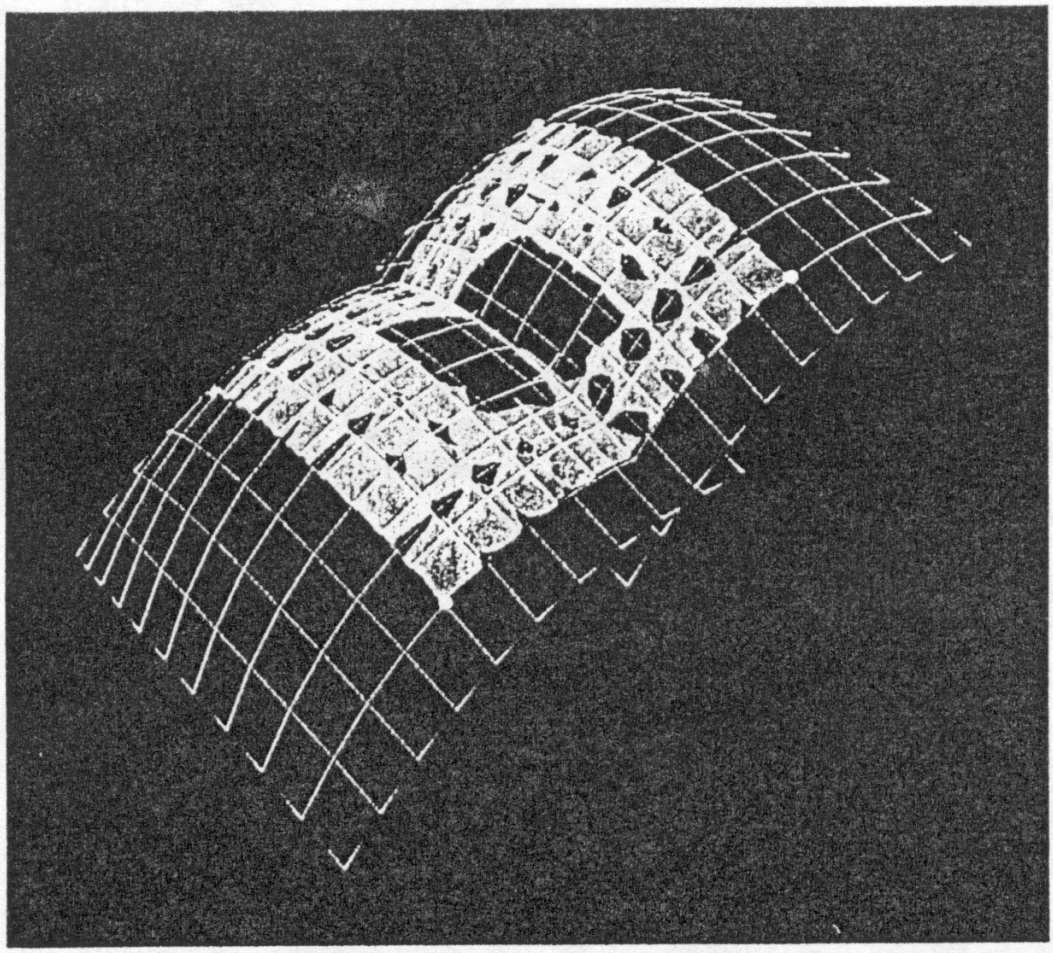

FIG. 5. Approximation of two intersecting surfaces.

ities along boundary curves of two patches or surface-surface intersections. In order to create a valid 3D grid around such surfaces, it is necessary to remove these anomalies from the geometry definition. The approach discussed in this paper has been developed for this purpose.

Further research will be concerned with automatically identifying the subsets of surfaces to be approximated, i.e., the user will no longer have to interactively specify these subsets. Another issue that will be addressed is the problem of dealing with nonconvex surfaces, which might not be approximated properly when using the current technique. This is due to the fact that many line-surface intersections can be found in the process of deriving the approximation conditions (see Section 3). The approximating B-spline surfaces are internally stored as NURBS surfaces. It will be investigated whether the weights in the NURBS representation can be used for the reduction of the approximation error (see Section 6).

This research was supported by the National Grid Project consortium.

REFERENCES

1 P. L. George, *Automatic Mesh Generation*, Wiley, New York, NY, 1991.

2 J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation*, North–Holland, New York, NY, 1985.

3 J. E. Castillo, *Mathematical Aspects of Numerical Grid Generation*, SIAM, Philadelphia, PA, 1991.

4 R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

5 C. de Boor, *A Practical Guide to Splines*, Applied Mathematical Sciences, Vol. 27, Springer–Verlag, New York, NY, 1978.

6 G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 3rd ed., Academic Press, San Diego, CA, 1993.

7 I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood Publishers, New York, NY, 1979.

8 M. Hosaka, *Modeling of Curves and Surfaces in CAD/CAM*, Springer–Verlag, New York, NY, 1992.

9 F. Yamaguchi, *Curves and Surfaces in Computer Aided Geometric Design*, Springer–Verlag, New York, NY, 1988.

10 R. Franke, Scattered data interpolation: Tests of some methods, *Math. Comp.*, 38, 181–200 (1982).