NURBS AND GRID GENERATION

ROBERT E. BARNHILL*, GERALD FARIN†, AND BERND HAMANN‡

Abstract. This paper provides a basic overview of NURBS and their application to numerical grid generation. Curve/surface smoothing, accelerated grid generation, and the use of NURBS in a practical grid generation system are discussed.

Key words. B-spline, NURBS, smoothing, structured grid generation, surface approximation, unstructured grid generation.

1. Introduction. We have recently been exposed to the area of numerical grid generation by Joe F. Thompson's research group at the NSF Engineering Research Center (ERC) for Computational Field Simulation (CFS), Mississippi State University. Some of the grid problems have strong connection with spline curves and surfaces, referred to as NURBS (Non-Uniform Rational B-Splines) in this paper. We shall discuss how some curve and surface design techniques have shown promise in the grid generation area.

To begin with, we need some spline basics. For a detailed description of cubic B-splines, as well as their conversion to the piecewise Bézier form, see [Farin '92]. A cubic B-spline curve is defined by a knot sequence $u_0 < \dots < u_L$ and a control polygon $\mathbf{d}_{-1}, \dots, \mathbf{d}_{L+1}$. The B-spline curve is C^2 overall and a cubic curve over each interval $[u_i, u_{i+1}]$. This cubic has a Bézier polygon $[\mathbf{b}_{3i}, \mathbf{b}_{3i+1}, \mathbf{b}_{3i+2}, \mathbf{b}_{3i+3}]$, found from

$$\mathbf{b}_{3i} = \frac{\Delta_i}{\Delta_{i-1}+\Delta_i}\mathbf{b}_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1}+\Delta_i}\mathbf{b}_{3i+1}, \tag{1.1}$$

where

$$\begin{aligned} \mathbf{b}_{3i+1} &= \frac{\Delta_i + \Delta_{i+1}}{\Delta}\mathbf{d}_i + \frac{\Delta_{i-1}}{\Delta}\mathbf{d}_{i+1}, \\ \mathbf{b}_{3i+2} &= \frac{\Delta_{i+1}}{\Delta}\mathbf{d}_i + \frac{\Delta_{i-1}+\Delta_i}{\Delta}\mathbf{d}_{i+1}, \\ \Delta &= \Delta_{i-1} + \Delta_i + \Delta_{i+1} \qquad \text{and} \\ \Delta_i &= u_{i+1} - u_i. \end{aligned} \tag{1.2}$$

The Bézier polygon $[\mathbf{b}_{3i}, \mathbf{b}_{3i+1}, \mathbf{b}_{3i+2}, \mathbf{b}_{3i+3}]$ defines a parametric curve

$$\mathbf{b}_i(t) = (1-t)^3\mathbf{b}_{3i} + 3t(1-t)^2\mathbf{b}_{3i+1} + 3t^2(1-t)\mathbf{b}_{3i+2} + t^3\mathbf{b}_{3i+3}, \tag{1.3}$$

* Office of the Vice President for Research and Strategic Initiatives, Arizona State University, Tempe, AZ 85287-2703, U.S.A.

† Department of Computer Science, Arizona State University, Tempe, AZ 85287-5406, U.S.A.

‡ NSF Engineering Research Center for Computational Field Simulation/Department of Computer Science, Mississippi State University, P.O. Box 6176, Mississippi State, MS 39762, U.S.A.

with $t = (u - u_i)/\Delta_i$. The collection of all L Bézier curves forms a twice differentiable parametric curve: It is 2D if the $\mathbf{d}_i$ lie in one plane, and 3D otherwise. The above equations show how to derive the curve from the coefficients $\mathbf{d}_i$. One can also solve the inverse problem, namely prescribing data points $\mathbf{x}_i$ and then finding $\mathbf{d}_i$ such that the curve defined by them passes through the given $\mathbf{x}_i$. This problem, cubic spline interpolation, leads to a linear system for the $\mathbf{d}_i$ and is easily solved, see [Farin '92].

The above definitions just address standard B-spline curves and their piecewise Bézier representation. *Rational* B-splines, or NURBS, are defined in an analogous way. Simply imagine that our B-spline curve is not defined in 3D space, but in 4D space. All the above formulas still hold, except that now they have four instead of three components per involved point. This 4D B-spline curve may be *projected* into 3D space by a projection through the origin onto the hyperplane $\omega = 1$, where ω denotes the fourth coordinate in 4D space. This projection is accomplished by the process of dividing through by the fourth coordinate, a process known as "inhomogenizing." The resulting curve is a rational B-spline curve.

2. Curve smoothing. We have worked for quite some time on devising methods to improve the shape of a B-spline curve (see [Farin & Sapidis '89], [Farin *et al.* '87], and [Sapidis & Farin '90]). One method can be found in [Farin & Worsey '91]. The idea behind it, *degree reduction fairing*, is as follows: The condition

$$\Delta^3\mathbf{b}_{3i} = \mathbf{b}_{3i} - 3\mathbf{b}_{3i+1} + 3\mathbf{b}_{3i+2} - \mathbf{b}_{3i+3} = 0 \tag{2.1}$$

means that the cubic defined by $\mathbf{b}_{3i}, \mathbf{b}_{3i+1}, \mathbf{b}_{3i+2}, \mathbf{b}_{3i+3}$ is actually a quadratic curve, *i.e.*, a parabola. It seems reasonable to hope for a better curve shape if all cubic pieces are close to being parabolic. The rationale is that parabolas are simpler in shape (*i.e.*, have less information content) than cubics, and when we try to "combat" digitizing errors, we also "combat" an overabundance of information in the curve shape.

We could rewrite (2.1) in terms of the B-spline coefficients $\mathbf{d}_i$ according to equations (1.1) and (1.2). This would lead to a global linear system for new control points $\hat{\mathbf{d}}_i$. There is a simpler geometric method that achieves the same goal. In addition, it will have the property of being local.

A cubic may be approximated by a quadratic by the process of *degree reduction*, see [Farin '92], [Forrest '72], and [Watkins & Worsey '88]. The approximating quadratic, with Bézier points $\mathbf{c}_{2i}, \mathbf{c}_{2i+1}, \mathbf{c}_{2i+2}$, may be obtained as

$$\mathbf{c}_{2i} = \mathbf{b}_{3i}, \quad \mathbf{c}_{2i+1} = \mathbf{m}_i, \quad \mathbf{c}_{2i+2} = \mathbf{b}_{3i+3}, \tag{2.2}$$

where

$$\mathbf{m}_i = \frac{1}{2}\Big(\frac{3}{2}\mathbf{b}_{3i+1} - \frac{1}{2}\mathbf{b}_{3i}\Big) + \frac{1}{2}\Big(\frac{3}{2}\mathbf{b}_{3i+2} - \frac{1}{2}\mathbf{b}_{3i+3}\Big). \tag{2.3}$$

Other meaningful ways to define $\mathbf{m}_i$ may exist, but (2.3) was sufficient for all applications so far.

The quadratic thus defined may be brought back to cubic form by the process of *degree elevation*. That is to say, we can find a cubic Bézier curve with control vertices $\hat{\mathbf{b}}_{3i}, \hat{\mathbf{b}}_{3i+1}, \hat{\mathbf{b}}_{3i+2}, \hat{\mathbf{b}}_{3i+3}$, which is identical to the given quadratic with vertices $\mathbf{b}_{3i}, \mathbf{m}_i, \mathbf{b}_{3i+3}$. These vertices are given by

$$\begin{aligned} \hat{\mathbf{b}}_{3i} &= \mathbf{b}_{3i}, \\ \hat{\mathbf{b}}_{3i+1} &= (\mathbf{b}_{3i} + 2\mathbf{m}_i)/3, \\ \hat{\mathbf{b}}_{3i+2} &= (\mathbf{b}_{3i+3} + 2\mathbf{m}_i)/3, \qquad \text{and} \\ \hat{\mathbf{b}}_{3i+3} &= \mathbf{b}_{3i+3}. \end{aligned} \tag{2.4}$$

The involved Bézier points may be expressed in terms of the B-spline vertices $\mathbf{d}_{i-1}, \mathbf{d}_i, \mathbf{d}_{i+1}$, and $\mathbf{d}_{i+2}$ according to the formulas given above. Let's agree to keep $\mathbf{d}_{i-1}$ and $\mathbf{d}_{i+2}$ fixed and to only change the other two control vertices. This simplification leads to the following algorithm to fair the i^{th} curve segment:

step 1 Given the initial B-spline curve, compute the Bézier points $\mathbf{b}_{3i}, \mathbf{b}_{3i+1}, \mathbf{b}_{3i+2}, \mathbf{b}_{3i+3}$.
step 2 Compute the quadratic approximation to the cubic defined by these Bézier points.
step 3 Degree-elevate that quadratic, resulting in a cubic control polygon $\mathbf{b}_{3i}, \hat{\mathbf{b}}_{3i+1}, \hat{\mathbf{b}}_{3i+2}, \mathbf{b}_{3i+3}$.
step 4 The new $\hat{\mathbf{d}}_i$ and $\hat{\mathbf{d}}_{i+1}$ are on the straight line through $\hat{\mathbf{b}}_{3i+1}, \hat{\mathbf{b}}_{3i+2}$. The involved ratios are known from (1.2). They are illustrated in Figure 2.1.

We now give the formulas for each step.

step 1 The necessary formulas are given by equations (1.1) and (1.2) in the Introduction.
step 2 This involves the computation of $\mathbf{m}_i$, which is given in (2.3).
step 3 We obtain $\hat{\mathbf{b}}_{3i+1}, \hat{\mathbf{b}}_{3i+2}$ from (2.4).
step 4 The desired new control points $\hat{\mathbf{d}}_i$ and $\hat{\mathbf{d}}_{i+1}$ are given by

$$\hat{\mathbf{d}}_i = \frac{(\Delta_{i-1} + \Delta_i)\hat{\mathbf{b}}_{3i+1} - \Delta_{i-1}\hat{\mathbf{b}}_{3i+2}}{\Delta_i} \tag{2.5}$$

and

$$\hat{\mathbf{d}}_{i+1} = \frac{(\Delta_i + \Delta_{i+1})\hat{\mathbf{b}}_{3i+2} - \Delta_{i+1}\hat{\mathbf{b}}_{3i+1}}{\Delta_i}, \tag{2.6}$$

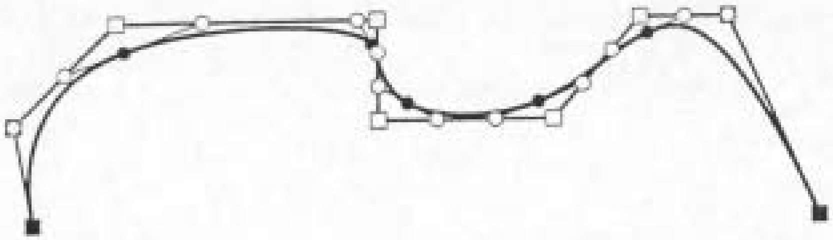

FIG. 2.1. *Cubic B-spline curves: the relationship between B-spline control polygon and Bézier control points.*

where i ranges from 0 to L. In order to avoid problems for $i = 0$ and $i = L$, we introduce additional knots $u_1 = u_0$ and $u_{L+1} = u_L$. The control points $\mathbf{d}_{-1}$ and $\mathbf{d}_{L+1}$, being the first and last points of the curve, are not changed.

step 5 The $\hat{\mathbf{d}}_i$ in (2.5) was already computed as $\hat{\mathbf{d}}_{i+1}$ for the previous i. We average the result from (2.5) with the previous value:

$$\hat{\mathbf{d}}_i = \frac{1}{2}\frac{(\Delta_{i-1}+\Delta_i)\hat{\mathbf{b}}_{3i+1} - \Delta_{i-1}\hat{\mathbf{b}}_{3i+2}}{\Delta_i} + \frac{1}{2}\hat{\mathbf{d}}_i. \tag{2.7}$$

This is not the only way to avoid a double definition of control points—we have obtained good results using it.

It should be kept in mind that the above algorithm does not produce a curve which consists of quadratic (*i.e.*, degree reduced) segments only:

a) Only $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$ are changed, whereas also $\mathbf{d}_{i-1}$ and $\mathbf{d}_{i+2}$ have to be changed in order to build a degree-reduced cubic, and
b) the result of the $(i-1)^{st}$ step is distorted in the i^{th} step by (2.7).

It may be argued that b) takes away from the idea of degree-reducing each curve segment. This is true, and we presently know only one argument to explain why b) does not pose a serious problem: The method works!

As an example of the method's performance, see Figure 2.2. Plots of the respective curves are not shown – to the scale that is possible for this paper, they look identical. That is not to say that they *are* the same! In fact, their properties differ significantly, be it for fluid flow applications or for aesthetic design criteria.

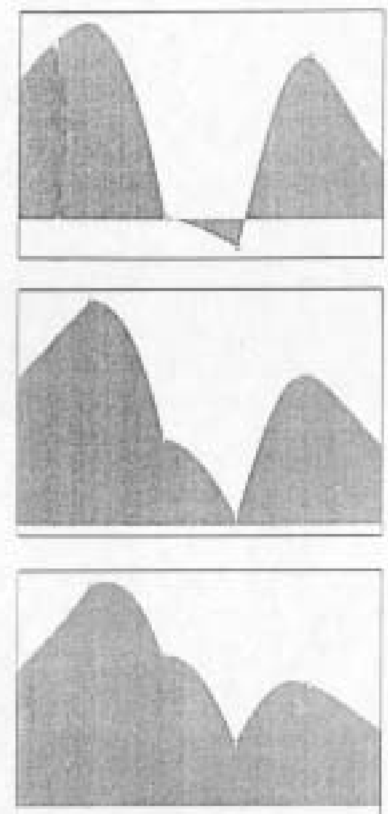

FIG. 2.2. *Curve fairing—top: the curvature of an initial curve; middle: after applying the fairing method twice; bottom: after applying it five times.*

An open question remains: does a "nice" shape of a curve guarantee optimal flow around an object of which it is an essential shape feature? Intuitively, the answer is positive, but we also know that flow properties change considerably depending on speed. It may therefore be adequate to ask for a "flow plot" of a curve instead of a curvature plot. Such a "flow plot" would certainly take velocity as input, but we do not know what precisely it would be, much less how to devise methods on how to optimize it.

3. Surface smoothing. Once imperfections are detected in a tensor product B-spline surface (see [de Boor '78] and [Farin '92]), one would want methods to remove them without time-consuming interactive adjustment of control polygons.

The following algorithm fairs a tensor product B-spline surface: let $\{\mathbf{d}_{i,j}\}$ be the control net of that surface with knot sequences $\{u_i\}$ and $\{v_j\}$. Fair the surface by first interpreting all rows of the control net as B-spline control polygons and then applying the curve fairing algorithm to each of them. In the second step, interpret all columns of the resulting control net as B-spline polygons and apply the curve fairing algorithm to each of them. The final control net will correspond to a surface that is fairer than the original one.

This algorithm will remove "extraneous information" from the surface—it is not tailored toward grid generation. However, we feel that a simple surface should fair better than one with bumps, in almost all applications, ranging from grids to aesthetics.

Our CAGD-based approach to measure the quality of a surface was first introduced in [Klass '80]. It is a simulation of a method used by stylists to see if the shape of a car is acceptable. A car is placed in a room with parallel fluorescent strip lights on the ceiling. These lights are reflected in the polished car surface, producing so-called "reflection lines." If each of them is fair (see above), then the surface is said to be fair itself. Figure 3.1 gives an example of degree reduction surface fairing. The reflection line that is shown corresponds to a light source perpendicular to the image plane and infinitely high over the surface.

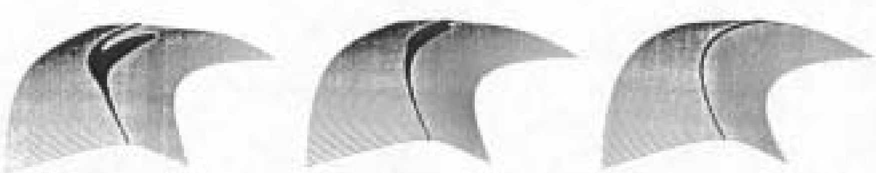

FIG. 3.1. *Surface fairing—left: a reflection line of the initial surface; middle: after fairing twice; bottom: after fairing four times.*

Other methods for surface fairing exist; they aim for the enforcement of convexity constraints in tensor product spline surfaces. Such methods are discussed in [Andersson *et al.* '87], [Jones '87], and [Kaufmann & Klass '88].

4. Accelerated grid generation. To start, we give some background information. Suppose we are given four arbitrary curves $\mathbf{c}_1(u)$, $\mathbf{c}_2(u)$ and $\mathbf{d}_1(v)$, $\mathbf{d}_2(v)$, defined over $u \in [0,1]$ and $v \in [0,1]$, respectively. Compute the bilinear Coons patch $\mathbf{x}(u,v)$ that has these four curves as boundary curves:

$$\mathbf{x}(u,v) = [(1-u) \quad u]\begin{bmatrix}\mathbf{x}(0,v)\\ \mathbf{x}(1,v)\end{bmatrix} + [\mathbf{x}(u,0) \quad \mathbf{x}(u,1)]\begin{bmatrix}(1-v)\\ v\end{bmatrix} - [(1-u)u]\begin{bmatrix}\mathbf{x}(0,0) & \mathbf{x}(0,1)\\ \mathbf{x}(1,0) & \mathbf{x}(1,1)\end{bmatrix}\begin{bmatrix}(1-v)\\ v\end{bmatrix}, \quad u,v \in [0,1]. \tag{4.1}$$

We have been introduced to the following problem in grid generation: One is given four boundary curves, represented by data points on each curve. A NURBS surface is desired "in between" the given curves. Here is a solution that has been described to us:

1. Apply the Coons formula (4.1), also referred to as "transfinite interpolation," to the data points in order to get a surface grid.
2. Use tensor product interpolation to find an interpolating B-spline surface (for information on surface interpolation, see [de Boor '78] and [Farin '92]).

The process may be accelerated considerably by applying the Coons formula in a more direct way.

Suppose that each boundary curve is given by its B-spline polygon—from an interpolation process to the boundary data, say. More precisely, assume we are given four control polygons $[\mathbf{d}_{0,0},\ldots,\mathbf{d}_{L,0}]$, $[\mathbf{d}_{0,M},\ldots,\mathbf{d}_{L,M}]$, $[\mathbf{d}_{0,0},\ldots,\mathbf{d}_{0,M}]$, and $[\mathbf{d}_{L,0},\ldots,\mathbf{d}_{L,M}]$. The following formula computes a complete B-spline control net that has the given polygons as its boundaries:

$$\mathbf{d}_{i,j} = [1-i/L \quad i/L]\begin{bmatrix}\mathbf{d}_{0,j}\\ \mathbf{d}_{L,j}\end{bmatrix} + [\mathbf{d}_{i,0} \quad \mathbf{d}_{i,M}]\begin{bmatrix}1-j/M\\ j/M\end{bmatrix} - [1-i/L \quad i/L]\begin{bmatrix}\mathbf{d}_{0,0} & \mathbf{d}_{0,1}\\ \mathbf{d}_{1,0} & \mathbf{d}_{1,1}\end{bmatrix}\begin{bmatrix}1-j/M\\ j/M\end{bmatrix}, \quad i=0,\ldots,L,\ j=0,\ldots,M. \tag{4.2}$$

As it turns out, that surface is precisely the Coons patch to the original boundary data! The proof is straightforward and relies on the fact that B-spline methods have linear precision. The accelerated method only has to perform interpolation on the four boundary curves, whereas the previously described method has to perform surface interpolation, which is an order of magnitude more complex. For 3D grids, the approach is completely analogous. An example of a Coons construction is given in Figure 4.1. The four boundary curves are cubic B-splines. The Coons formula is then applied to the boundary control polygons.

5. The National Grid Project and NURBS. Few years ago, the areas of geometric modeling and grid generation were viewed as indepen-

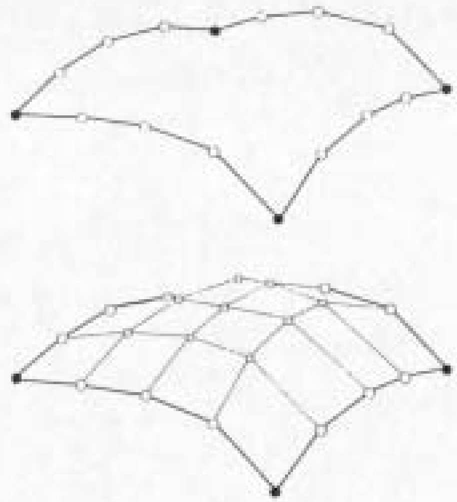

FIG. 4.1. *The Coons formula – a control net (bottom) is formed by "filling in" control points between four boundary control polygons (top).*

dent scientific/engineering fields. The reason for this is the fact that most people working in the area of grid generation have an engineering background, whereas experts in geometric modeling typically have a mathematics or computer science background. In 1991, a team effort headed by Joe F. Thompson was started at the NSF ERC for Computational Field Simulation (CFS), whose goal was the development of a next-generation grid generation system. This effort, the "National Grid Project" (NGP), has led to a grid generation system that unifies geometric modeling, structured grid generation, and unstructured grid generation functionality. The geometry representation of the system is entirely based on NURBS.

The interdisciplinary nature of the NSF ERC for CFS made it possible to design a completely new grid generation system. The NGP system converts given CAD data (IGES format) to a "standard" internal data format for NURBS curves and surfaces. In addition, the system provides a set of useful CAD functions needed for the grid generation process. These CAD functions enable the user to construct his own geometry from scratch, correct or add to a given geometry, or define curves and surfaces defining the field boundaries around a geometry to be used for field simulation.

The decision to use NURBS affects most modules of the overall grid generation system. Primarily, the internal CAD system and the structured and unstructured grid generation modules are affected by the choice of NURBS. The system supports the conversion of the most common IGES

entities used in aircraft, car body, and ship hull design. Among these entities are *composite curve*, *parametric spline curve/surface*, *ruled surface*, *surface of revolution*, and *rational B-spline curve/surface*. Whenever possible, the given IGES data files are converted exactly into the corresponding NURBS representation. NURBS approximations of given IGES data are generated only if an exact conversion is not possible.

The internal CAD system provides the tools required to construct one's own geometries, field boundaries, and blocks. All CAD operations operate on NURBS. These are some of the CAD functions available in the NGP system (or currently under development):

- Cubic spline interpolation for curves and surfaces with user-specified end conditions
- Ruled surface, surface of revolution, and sweep surface
- Transfinite interpolation (Coons patch)
- Conic sections and quadric surfaces
- Splitting curves/surfaces in their parametric spaces
- Union of curves/surfaces
- Derivative computation, *e.g.*, curvature computation
- Curve/surface projection onto a given geometry
- Curves/surfaces defined on surfaces (*i.e.*, curves/surfaces defined in the parameter space of existing surfaces)
- Offset curves/surfaces
- Curve-curve, curve-surface, and surface-surface intersection
- Reparametrization of NURBS curves/surfaces

This list of CAD functions is by no means complete, but provides an insight into the system's capabilities. New CAD functions that turn out to be useful for the overall block and grid generation process are permanently added to the system. Major development work is still being devoted to the representation and manipulation of trimmed entities resulting from surface-surface intersection and to the problem of performing CAD operations "across" the parameter spaces of multiple NURBS curves/surfaces. Figure 5.1 shows an example of a complex aircraft geometry and its surrounding far field boundary created entirely with the NGP CAD system. Many of the CAD concepts, algorithms, and implementations found in the NGP system are based on the methods discussed in [Bartels *et al.* '87], [Farin '92], [Hosaka '92], [Piegl '91], and [Yamaguchi '88]. Several aspects of the overall NGP system are discussed in [Hamann & Sarraga '94].

6. Using NURBS for interactive geometry correction. For practical grid generation purposes, a given geometry around which a grid is to be constructed must be error-free, *i.e.*, a geometry must not contain any discontinuities (overlapping patches, gaps between patches, or surface intersections). This is crucial for the generation of a valid grid. Unfortunately, most given CAD data contain countless such errors. This has always been

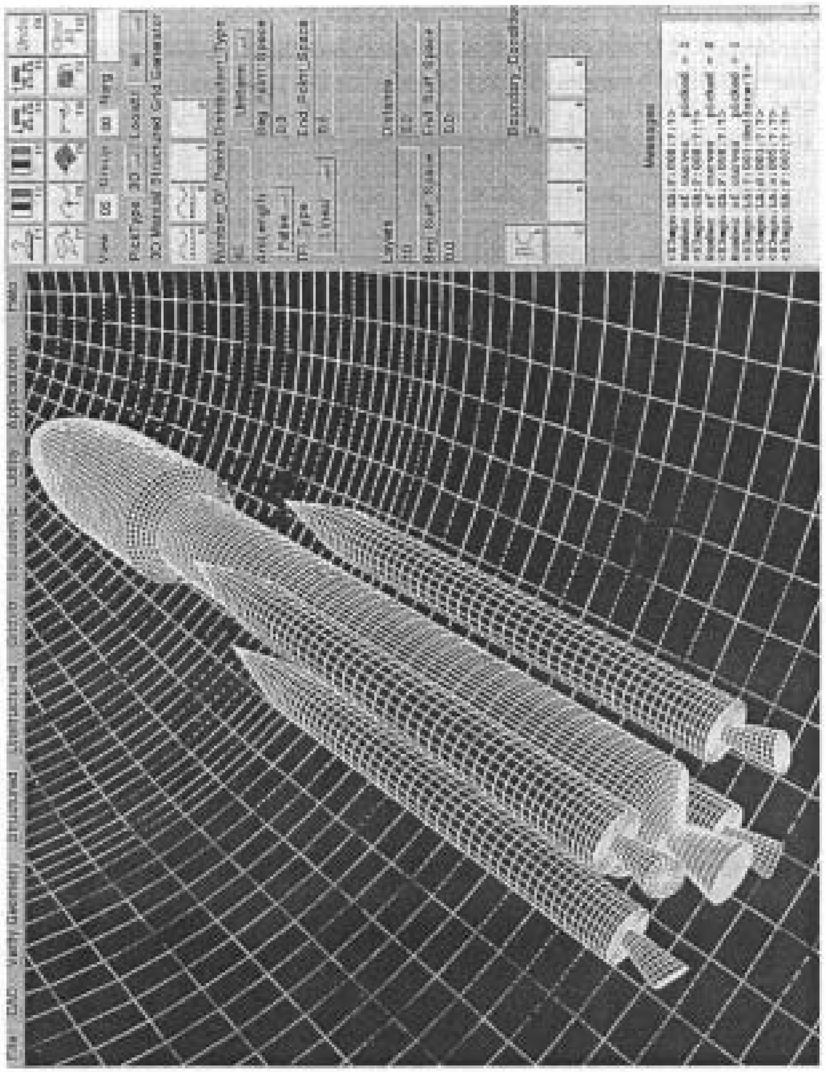

FIG. 5.1. *Geometry and part of far field boundary constructed with the NGP system image generated by John B. Whitmire, NSF ERC, Mississippi State University).*

a main reason that makes grid generation so time-consuming.

The NGP system provides a geometry correction technique that requires the user to interactively specify combinations of points and surfaces in regions containing undesired discontinuities. The method then automatically computes a surface that locally approximates the given geometry and "removes" the discontinuities. Usually, this process must be executed in many different regions of a geometry. Eventually, a new model of the geometry is obtained that consists partly of original NURBS surfaces and partly of NURBS surfaces generated by the interactive correction technique. The resulting model is free of discontinuities and may be used for grid generation.

The geometry correction technique is based on constructing an initial local surface approximant (Coons patch) that is projected onto the given geometry, thereby defining a new NURBS surface. The originally given geometry is replaced by a combination of original NURBS surfaces and local surface approximants. The overall approximation process requires these steps:

(i) Definition of four surface boundary curves
(ii) Computation of points on the bilinear Coons patch implied by these four boundary curves
(iii) Projection of these points onto the given geometry
(iv) Generation of additional points whenever certain points on the Coons patch can not be projected onto the geometry
(v) Interpolation of the points resulting from step (iii) and step (iv)

An error estimate is computed for the local surface approximation. Eventually, the system will choose the number of projections depending on a maximally allowed error tolerance that is specified by the user. Existing curves of a given geometry, *e.g.*, boundary curves of surfaces, can be preserved by the method. The four curves specified by the user are blended by a bilinear Coons patch $\mathbf{x}(u,v)$ (see Section 4) that is then projected onto the given geometry. The Coons patch is uniformly evaluated in parameter space, and the resulting points on the Coons patch are projected onto the geometry.

Denoting the points on the Coons patch by $\mathbf{x}_{i,j}$, the outward unit normal vectors at these points are given by

$$\mathbf{n}_{i,j} = \mathbf{n}(u_{i,j}, v_{i,j}) = \frac{\frac{\partial}{\partial u}\mathbf{x}(u_{i,j}, v_{i,j}) \times \frac{\partial}{\partial v}\mathbf{x}(u_{i,j}, v_{i,j})}{\left\| \frac{\partial}{\partial u}\mathbf{x}(u_{i,j}, v_{i,j}) \times \frac{\partial}{\partial v}\mathbf{x}(u_{i,j}, v_{i,j}) \right\|}, \tag{6.1}$$

where $\| \; \|$ denotes the Euclidean norm. A family of line segments, defined by the points $\mathbf{x}_{i,j}$, the normals $\mathbf{n}_{i,j}$, and a fixed line segment length, is intersected with the given geometry. This process is called "projection." If a line segment has multiple intersections with the given geometry, the one closest to the Coons patch is chosen.

Since the original surfaces might be discontinuous, certain points on the Coons patch can not be projected onto the original geometry - at least not in normal direction $\mathbf{n}_{i,j}$. Considering all the projections that could be generated, each of these projections, denoted by $\mathbf{p}_{i,j}$, can be represented as a linear combination of the end points of the line segment associated with the point $\mathbf{x}_{i,j}$ on the Coons patch and the associated normal vector $\mathbf{n}_{i,j}$. Thus, each projection $\mathbf{p}_{i,j}$ has some associated (linear) parameter $t_{i,j}$.

Thus, the problem of "missing projections" becomes a bivariate approximation problem: Parameter values $t_{i,j}$ must be approximated for all points $\mathbf{x}_{i,j}$ without projection. Hardy's reciprocal multiquadric method is used for this bivariate scattered approximation problem (see [Franke '82]).

The system of linear equations to be solved is given by

$$t_{i,j} = t(u_{i,j}, v_{i,j}) = \sum_{J \in \{0,\ldots,N\}} \sum_{I \in \{0,\ldots,M\}} c_{I,J} \left(R + (u_{I,J} - u_{i,j})^2 + (v_{I,J} - v_{i,j})^2 \right)^{-\gamma}, \quad i \in \{0,\ldots,M\},\ j \in \{0,\ldots,N\}, \tag{6.2}$$

where only those values $t_{i,j}$, $u_{I,J}$, $u_{i,j}$, $v_{I,J}$, and $v_{i,j}$ are considered for which a projection has been found. The value $\gamma = 0.5$ yields reasonable results, but "optimal" values for γ and R are currently not known. The NGP system actually uses a localized version of Hardy's reciprocal multiquadric method.

Once the coefficients $c_{I,J}$ are known, additional points can be computed, "artificial projections," for which the associated points $\mathbf{x}_{i,j}$ on the Coons patch could not be projected onto the geometry. The $(M+1) \times (N+1)$ points obtained by projection and Hardy's method are interpolated by a C^1 continuous, bicubic NURBS surface (see [Farin '92]). The geometry correction technique is described in greater detail in [Hamann '94], [Hamann & Jean '94], and [Soni & Hamann '93]. It is planned to further improve the quality of the resulting surface approximations and minimize the necessary user input by automating the method as much as possible. An example of a given car body configuration is shown in Figure 6.1. The upper two views show the original CAD description containing "holes," and the lower two views show the continuous approximation of it.

7. Structured and unstructured grid generation using NURBS. The fundamental concepts of numerical grid generation are discussed in [George '91] and [Thompson *et al.* '85]. Recent advances in grid generation technology are presented in [Castillo '91], and an extensive literature review of numerical grid generation can be found in [Thompson & Weatherill '93].

Both the structured and unstructured grid generation modules of the NGP system utilize NURBS curve and surface representations. The generation of structured grid of a single NURBS surface requires two steps. The first step is the generation of boundary curve grids having a user-specified point distribution, and the second step is the generation of grid points in

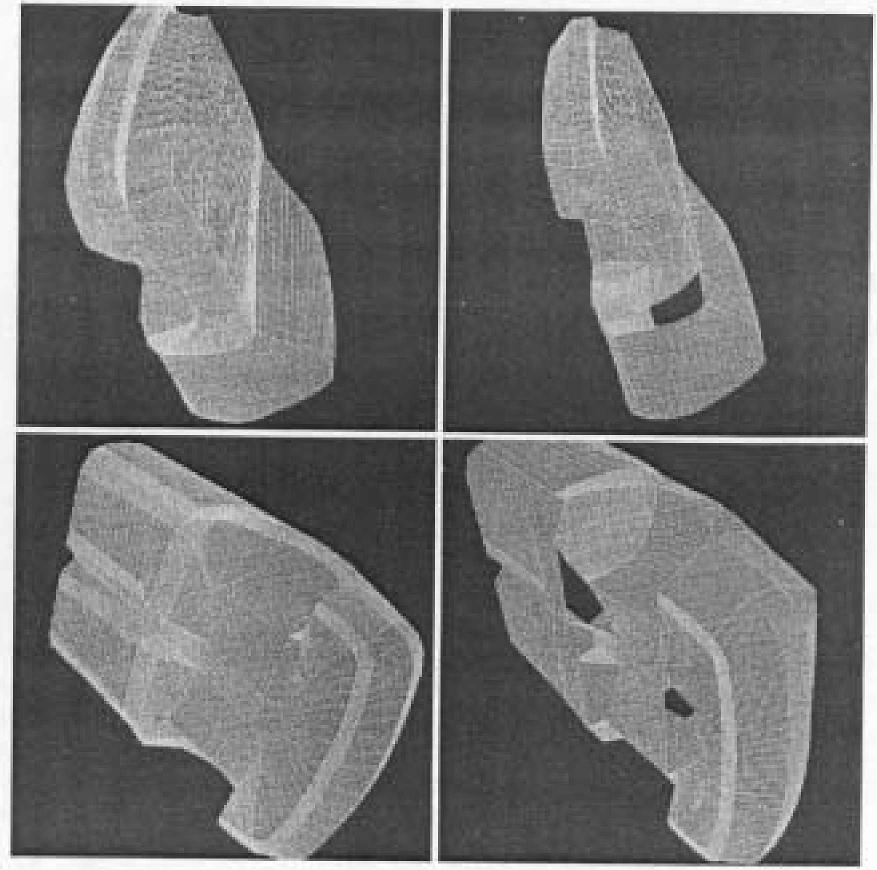

FIG. 6.1. *Car body geometry with "holes" and its approximation (images generated by Brian A. Jean, NSF ERC, Mississippi State University).*

the surface's interior. An initial surface grid is computed by performing transfinite interpolation of the boundary curve grids. This initial surface grid is then iteratively "smoothed," which yields a surface grid whose grid lines intersect orthogonally. This process involves the partial derivatives of a parametric surface, which can be computed directly from the NURBS representation.

The iterative smoothing technique that is used in the NGP system is based on solving an elliptic partial differential equation system with Dirichlet (or Neumann) boundary conditions relating *physical* (x, y, z), *parametric* (u, v), and *computational* (ξ, η) variables. Thus, a single NURBS surface $\mathbf{s}(u, v)$ is viewed as

$$\begin{aligned} \mathbf{s}(u,v) &= (x(u,v), y(u,v), z(u,v)) \\ &= \Big(x\big(u(\xi,\eta), v(\xi,\eta)\big), y\big(u(\xi,\eta), v(\xi,\eta)\big), z\big(u(\xi,\eta), v(\xi,\eta)\big)\Big). \end{aligned} \tag{7.1}$$

The elliptic system to be solved is given by the two equations

$$\begin{aligned} g_{22}(u_{\xi\xi} + Pu_\xi) - 2g_{12}u_{\xi\eta} + g_{11}(u_{\eta\eta} + Qu_\eta) &= J^2 \Delta_2 u \quad \text{and} \\ g_{22}(v_{\xi\xi} + Pv_\xi) - 2g_{12}v_{\xi\eta} + g_{11}(v_{\eta\eta} + Qv_\eta) &= J^2 \Delta_2 v, \end{aligned} \tag{7.2}$$

where

$$\begin{aligned} g_{11} &= \bar{g}_{11}u_\xi^2 + 2\bar{g}_{12}u_\xi v_\xi + \bar{g}_{22}v_\xi^2, \\ g_{12} &= \bar{g}_{11}u_\xi u_\eta + \bar{g}_{12}(u_\xi v_\eta + u_\eta v_\xi) + \bar{g}_{22}v_\xi v_\eta, \\ g_{22} &= \bar{g}_{11}u_\eta^2 + 2\bar{g}_{12}u_\eta v_\eta + \bar{g}_{22}v_\eta^2, \\ \Delta_2 u &= \bar{J}\left[\frac{\partial}{\partial u}\left(\frac{\bar{g}_{22}}{\bar{J}}\right) - \frac{\partial}{\partial v}\left(\frac{\bar{g}_{12}}{\bar{J}}\right)\right], \\ \Delta_2 v &= \bar{J}\left[\frac{\partial}{\partial v}\left(\frac{\bar{g}_{11}}{\bar{J}}\right) - \frac{\partial}{\partial u}\left(\frac{\bar{g}_{12}}{\bar{J}}\right)\right], \quad \text{and} \\ \bar{g}_{11} &= \mathbf{s}_u \cdot \mathbf{s}_u, \quad \bar{g}_{12} = \mathbf{s}_u \cdot \mathbf{s}_v, \quad \bar{g}_{22} = \mathbf{s}_v \cdot \mathbf{s}_v, \\ J &= u_\xi v_\eta - u_\eta v_\xi, \quad \text{and} \quad \bar{J} = \sqrt{\bar{g}_{11}\bar{g}_{22} - \bar{g}_{12}^2}. \end{aligned} \tag{7.3}$$

The functions P and Q control the grid point distribution. Using $P = Q = 0$ leads to a uniformly spaced grid. To solve the elliptic system, P and Q must be estimated from some initial grid that (nearly) has the desired point distribution. Alternatively, the functions P and Q can be computed based on the user-specified boundary curve distributions by performing transfinite interpolation. These control functions are iteratively smoothed yielding the final grid. Often, grid line orthogonality is required. This implies that $\mathbf{s}_\xi \cdot \mathbf{s}_\eta = 0$ must hold and that grid points can move on the boundary curves. A more detailed discussion is given in [Khamayseh & Hamann '94].

The same principles generalize to the 3D volume case using NURBS volumes defined over cuboids. This is described in [Thompson *et al.* '85]. Figure 7.1 shows the initial surface grid (obtained by transfinite interpolation of the boundary curve grids) and the elliptically smoothed surface grid of the space shuttle.

The representation of parametric curves and surfaces as NURBS in the NGP system has turned out to be extremely beneficial for the grid generation process. The evaluation and differentiation of NURBS is based on a geometric algorithm ("de Boor algorithm," see [Farin '92]), which is fast and numerically stable.

The generation of unstructured grids, *i.e.*, grids defined by triangles (surface grids) and tetrahedra (volume grids), is relatively independent of the underlying curve and surface representation. In the NGP system, the unstructured grid generation module mainly depends on the evaluation of NURBS. The generation of an unstructured surface grid is based on computing the Delaunay triangulation of a set of scattered points in a surface's parameter space, which might lead to poor surface triangulations due to the parametrization. This problem is currently being investigated.

The generation of unstructured volume grids surrounding a 3D geometry is based on computing the 3D Delaunay triangulation of a set of scattered points in the surrounding field. The variation of point densities is realized by using "point" and "line sources," from which the grid point density decreases in some user-specified fashion. These concepts are discussed in [Weatherill '92]. Unlike structured grid generation, unstructured grid generation requires little user input and operates highly automatically. Unfortunately, many numerical field solution algorithms can only handle structured grids, which is the reason why the NGP system was designed to support both types of grids. Figure 7.2 shows an unstructured surface/volume grid for a car body configuration.

Recently, an alternative approach to the Delaunay-based grid generation technique has been developed (see [Hamann *et al.* '94]). This alternative approach is based on intersecting the edges in an initial volume triangulation with a given geometry, extracting the "valid" part (*i.e.*, the exterior or interior part of the volume triangulation) of this initial volume triangulation, and iteratively inserting grid points in the field. The grid point density decreases with increasing distance to the geometry. It also depends on the local surface curvature. The technique is completely automatic.

The initial volume triangulation consists of a uniform "density" of tetrahedra, and it is iteratively improved by inserting points until a desired density is obtained, which reflects the distance to and the curvature of the geometry nearby. This techniques works for closed geometries (of arbitrary topological genus) only. A closed geometry allows to characterize each 3D point as an exterior, as an interior, or as a surface point. These are the steps involved in generating an unstructured 3D volume grid:

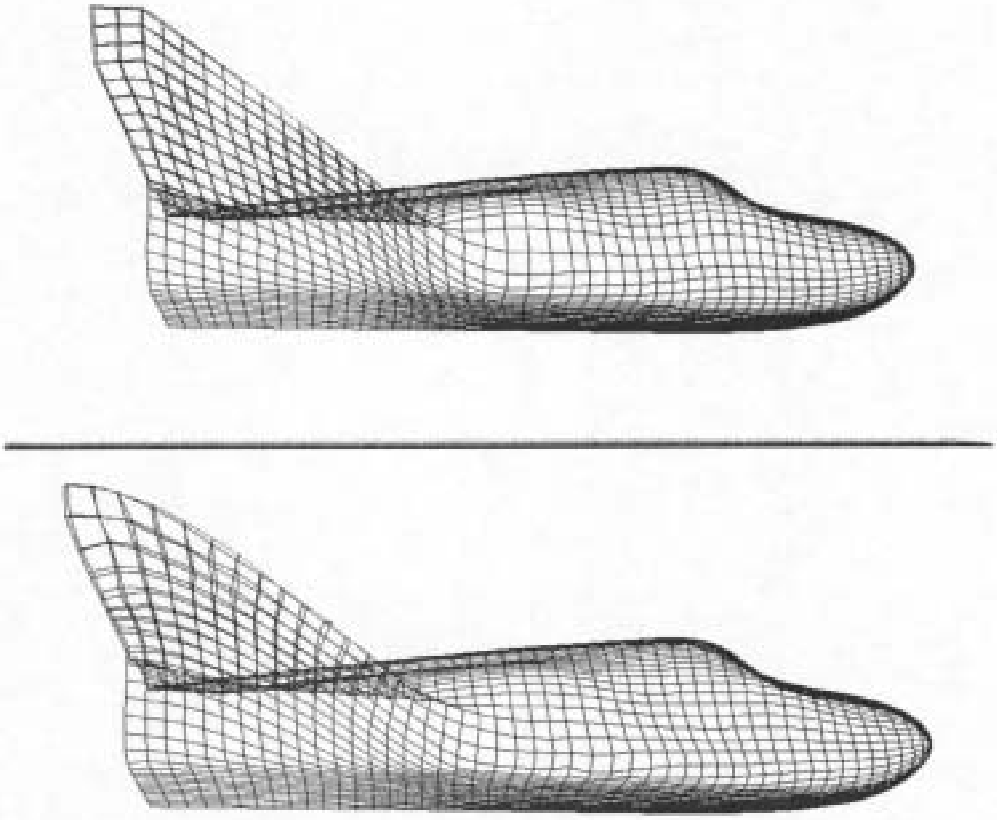

FIG. 7.1. *Initial (top) and elliptically smoothed grid (bottom) of space shuttle (image generated by Ahmed Khamayseh, NSF ERC, Mississippi State University).*

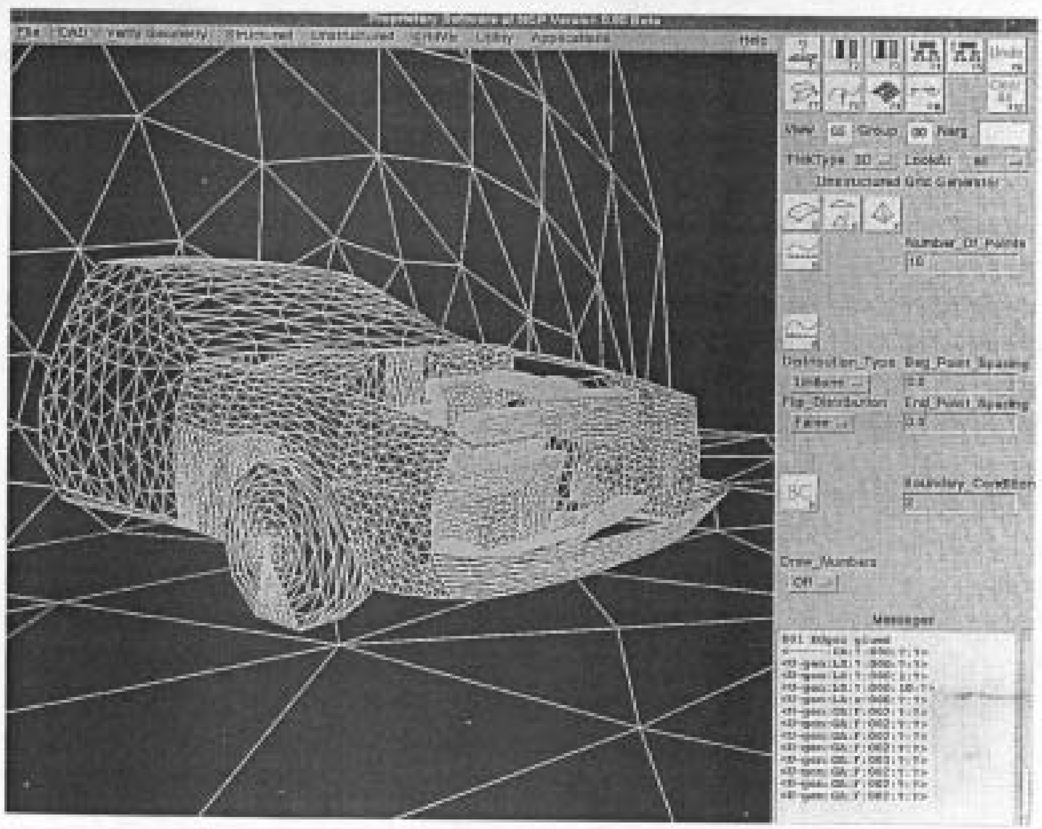

FIG. 7.2. *Unstructured surface/volume grid generated with the NGP' system (image generated by Kelly L. Parmley, NSF ERC, Mississippi State University).*

(i) Computation of an initial, uniform triangulation of a bounding volume that contains the entire geometry
(ii) Computation of intersections of all edges in the initial triangulation with the geometry
(iii) Extraction of that part of the initial triangulation that lies outside (or, alternatively, inside) the given closed geometry
(iv) Insertion of additional grid points into the triangulation until the desired density of tetrahedra is achieved

One of the basic principles underlying this technique is the association of an "expected volume" with a tetrahedron that has a particular distance from the geometry. In general, the distance between a tetrahedron and the geometry is measured as the shortest (perpendicular) distance between the centroid of a tetrahedron and the geometry. Denoting the distance between a tetrahedron $\mathcal{T}$ and the geometry by d, the expected volume of $\mathcal{T}$ is defined as

$$V(d) = \frac{d_{\max} - d}{d_{\max} - d_{\min}} V_{\min} + \frac{d - d_{\min}}{d_{\max} - d_{\min}} V_{\max}, \tag{7.4}$$

where $V_{\min}$ is the minimum and $V_{\max}$ the maximum volume of all tetrahedral volumes, and $d_{\min}$ is the minimum and $d_{\max}$ the maximum distance between the tetrahedra and the geometry (considering all tetrahedra).

In the implementation, it is the goal to obtain a tetrahedral volume for each tetrahedron that differs very little from the expected volume $V(d)$. Formula (7.4) can be modified further in order to account for surface curvature and a non-linear decrease of tetrahedral volumes with respect to distance (see [Hamann *et al.* '94]). Eventually, the triangulation is improved by performing edge-triangle swapping for convex hexahedra with triangular faces in order to avoid "long," "skinny" tetrahedra. Figure 7.3 shows a slice of the resulting volume grid surrounding a wing. Tetrahedra with relatively smaller volumes are found closer to the surface.

8. Future research. We have described a method to smooth 1D input data, *i.e.*, curves, based on geometric criteria. It should be beneficial to supplement the geometric criteria by more application specific ones. The same is, of course, true for surfaces.

We have not exploited the flexibility that is offered by using NURBS instead of the more traditional B-splines. Methods are needed that make judicious use of the extra parameters available in NURBS, for example by ensuring that cylindrical surfaces are reproduced *exactly*, not only approximately.

The above surface considerations (Section 3 and 4) apply to structured grids. Smooth surfaces through unstructured grids are provided by triangular or tetrahedral surface schemes. As an example, the Clough-Tocher element, see [Strang & Fix '73], had been discovered as a surface generation method by Barnhill (see [Barnhill '77]). It is a piecewise polynomial

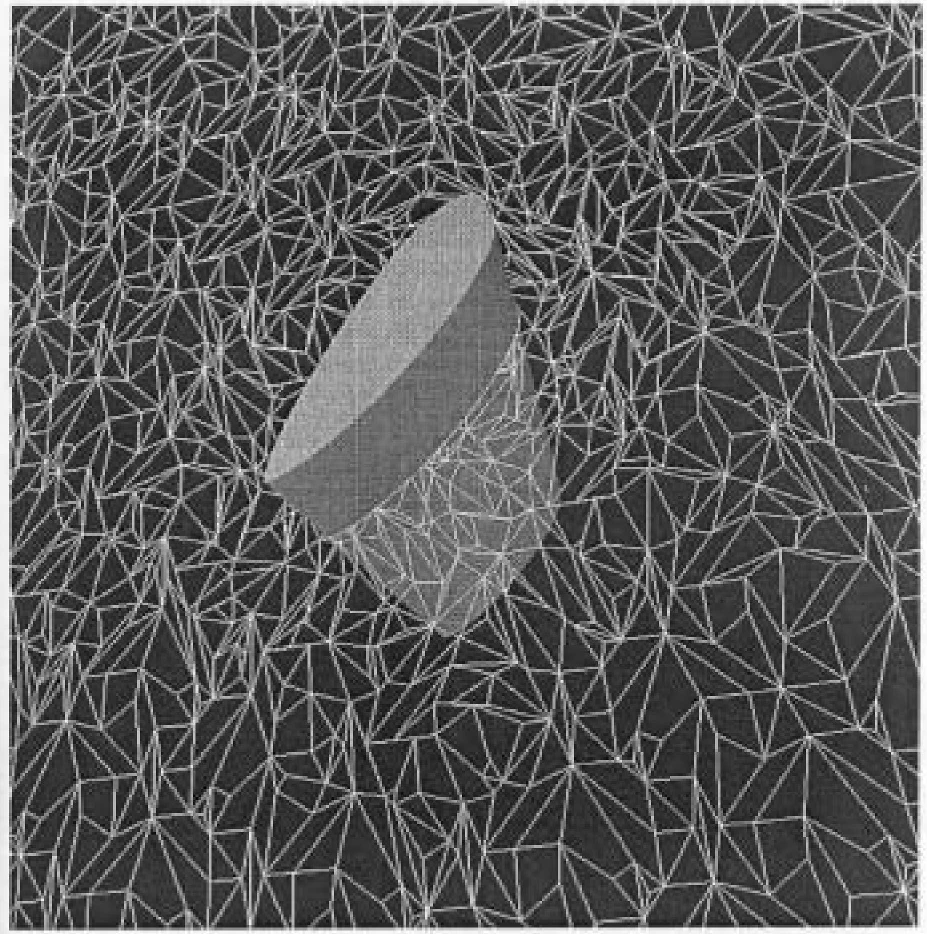

FIG. 7.3. *Slice of unstructured volume grid outside wing (image generated by Guangshi Hong, Department of Computer Engineering, Mississippi State University).*

element and is most conveniently written in terms of multivariate Bernstein polynomials, see [Farin '86]. A generalization to higher dimensions is described in [Worsey & Farin '87].

It has been outlined to what extent NURBS are being used in a practical grid generation system. The approximation of any given CAD data containing errors by a minimal number of approximating NURBS surfaces remains an open question (Section 6). Another open research problem is the *completely automatic* approximation of any given CAD geometry containing undesired holes, intersections, and overlapping patches.

For grid generation purposes, in particular unstructured grid generation, it is extremely important that NURBS surfaces are reasonably parametrized, *i.e.*, relative metric information in parameter space and on the surface should differ very little. Appropriate reparametrization algorithms are currently being developed.

9. Acknowledgements. The efforts of Joe F. Thompson to introduce us to the area of numerical grid generation are greatly appreciated. Peter R. Eiseman has supplied valuable background information. We thank all members of the National Grid Project research and development team at the NSF Engineering Research Center for Computational Field Simulation, Mississippi State University. This research was supported in part by NSF grant DMC-8807747 and by DoE grant DE-FG02-87ER25041 to Arizona State University, NSF grant ASC-9210439 to Mississippi State University, and the National Grid Project consortium.

REFERENCES

[1] R. ANDERSSON, E. ANDERSSON, M. BOMAN, B. DAHLBERG, T. ELMROTH, AND B. JOHANNSON, *The automatic generation of convex surfaces*, in *The Mathematics of Surfaces II* (ed., R.R. MARTIN) Oxford University Press, New York 1987, pp. 427–445.

[2] R.E. BARNHILL, *Representation and approximation of surfaces*, in *Mathematical Software III* (ed., J.R. RICE) Academic Press, San Diego 1977, pp. 69–120.

[3] 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, Inc., Los Altos 1987.

[4] J.E. CASTILLO, *Mathematical Aspects of Numerical Grid Generation*, SIAM, Philadelphia 1991.

[5] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York 1978.

[6] G. FARIN, *Triangular Bernstein-Bézier patches*, Computer Aided Geometric Design **3** (1986), pp. 83–128.

[7] G. FARIN, *Curves and Surfaces for Computer Aided Geometric Design* (third edition) Academic Press, San Diego 1992.

[8] G. FARIN, G. REIN, N. SAPIDIS, AND A.J. WORSEY, *Fairing cubic B-spline curves*, Computer Aided Geometric Design **4** (1987), pp. 91–104.

[9] G. FARIN, AND N. SAPIDIS, *Curvature and the fairness of curves and surfaces*, IEEE Computer Graphics and Applications **9** (1989), pp. 52–57.

[10] G. FARIN, AND A.J. WORSEY, *Reparameterization and degree elevation for rational Bézier curves*, in *NURBS for Curve and Surface Design* (ed., G. FARIN) SIAM, Philadelphia 1991, pp. 47–57.

[11] A. FORREST, *Interactive interpolation and approximation by Bézier polynomials*, The Computer J. **15** (1972), pp. 71–79.

[12] R. FRANKE, *Scattered data interpolation: Tests of some methods*, Math. Comp. **38** (1982), pp. 181–200.

[13] P.L. GEORGE, *Automatic Mesh Generation*, Wiley & Sons, New York 1991.

[14] B. HAMANN, *Construction of B-spline approximations for use in numerical grid generation*, Applied Mathematics and Computation (to appear).

[15] B. HAMANN, J.L. CHEN, AND G. HONG, *Automatic generation of unstructured grids for volumes outside or inside closed surfaces*, in *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields* (eds., N.P. WEATHERILL, P.R. EISEMAN, J. HÄUSER, AND J.F. THOMPSON) Pineridge Press Ltd., Swansea, U.K., 1994, pp. 187–197.

[16] B. HAMANN AND B.A. JEAN, *Interactive surface correction based on a local approximation scheme*, *Finite Elements, Grid Generation, and Geometric Design* (eds., B. HAMANN AND R.F. SARRAGA) SIAM, Philadelphia (to appear).

[17] B. HAMANN AND R.F. SARRAGA, *Finite Elements, Grid Generation, and Geometric Design*, SIAM, Philadelphia (to appear).

[18] M. HOSAKA, *Modeling of Curves and Surfaces in CAD/CAM*, Springer-Verlag, New York 1992.

[19] A. JONES, *Shape control of curves and surfaces through constrained optimization*, *Geometric Modeling: New Trends and Algorithms*, (ed., G. FARIN) SIAM, Philadelphia 1987, pp. 265–279.

[20] E. KAUFMANN AND R. KLASS, *Smoothing surfaces using reflection lines for families of splines*, Computer Aided Design **20** (1988), pp. 312–316.

[21] A. KHAMAYSEH AND B. HAMANN, *Elliptic grid generation using NURBS surfaces*, *Finite Elements, Grid Generation, and Geometric Design* (eds., B. HAMANN AND R.F. SARRAGA) SIAM, Philadelphia (to appear).

[22] R. KLASS, *Correction of local surface irregularities using reflection lines*, Computer Aided Design **12** (1980), pp. 73–77.

[23] L.A. PIEGL, *Rational B-spline curves and surfaces for CAD and graphics*, in *State of the Art in Computer Graphics* (eds., D.F. ROGERS AND R.A. EARNSHAW) Springer-Verlag, New York 1991, pp. 225–269.

[24] N. SAPIDIS AND G. FARIN, *Automatic fairing algorithm for B-spline curves*, Computer Aided Design **22** (1990), pp. 121–129.

[25] B.K. SONI AND B. HAMANN, *Computational geometry tools in grid generation*, in *Advances in Hydro-Science & -Engineering* (ed., S.S.Y. WANG) **Vol. I** (Part B) (1993), pp. 2004–2009.

[26] G. STRANG AND G. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs 1973.

[27] J.F. THOMPSON, Z.U.A. WARSI AND C.W. MASTIN, *Numerical Grid Generation*, North-Holland, New York 1985.

[28] J.F. THOMPSON AND N.P. WEATHERILL, *Aspects of numerical grid generation: current science and art*, Proceedings of the 11th AIAA Applied Aerodynamics Conference, Monterey, August 1993.

[29] Z.U.A. WARSI, *Numerical grid generation in arbitrary surfaces through a second-order differential geometric model*, Journal of Computational Physics **64** (1986), pp. 82–96.

[30] M. WATKINS AND A.J. WORSEY, DEGREE REDUCTION FOR BÉZIER CURVES, Computer Aided Design **20** (1988), pp. 398–405.

[31] N.P. WEATHERILL, *The Delaunay triangulation in CFD*, Computers and Mathematics with Applications **24** (1992), pp. 129–150.

[32] A.J. WORSEY AND G. FARIN, *An n-dimensional Clough-Tocher element*, Constructive Approximation **3** (1987), pp. 99–110.

[33] F. YAMAGUCHI, *Curves and Surfaces in Computer Aided Geometric Design*, Springer-Verlag, New York 1988.