ELSEVIER Computer Aided Geometric Design 13 (1996) 369–386

COMPUTER AIDED GEOMETRIC DESIGN

Elliptic grid generation using NURBS surfaces [☆]

Ahmed Khamayseh [a,*], Bernd Hamann [b]

[a] *Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

[b] *Department of Computer Science, University of California, Davis, CA 95616-8562, USA*

Received July 1993; revised February 1994

Abstract

Recently, there has been a move towards NURBS-based grid generation systems, where the original geometry is given as analytically defined NURBS surfaces. The process of surface grid generation is the computation of an algebraic grid based on the NURBS surface definition and the computation of an elliptic grid based on the algebraic grid. The NURBS format provides a common mathematical representation for both standard analytic shapes and free-form curves and surfaces. The derivatives of the physical coordinates with respect to the parametric coordinates can be evaluated directly. An improved elliptic surface grid generation method for NURBS surfaces is presented. New techniques for computing the control functions and imposing boundary orthogonality are developed.

Keywords: Algebraic grid generation; Approximation; Elliptic grid generation; NURBS curve; NURBS surface; Partial differential equations; Transfinite interpolation

1. Introduction

Elliptic methods for constructing three-dimensional surface and volume grids have been known for many years. Recently, there has been a move towards Non-Uniform Rational B-Splines (NURBS) in grid generation systems. This move follows current trends in geometric modeling and Computer-Aided Geometric Design (CAGD). Since the paper addresses the CAGD and grid generation communities, some basic definitions, algorithms, and approaches regarding NURBS and elliptic grid generation are reviewed. NURBS are becoming the *de facto* industry standard for geometry representation but

[☆] This work was supported by the National Grid Project consortium and the National Science Foundation.

[*] Corresponding author. E-mail: ahmed@lanl.gov.

are still a rather new "tool" to the grid generation community. NURBS allow the representation of nearly all geometries relevant for aircraft and automobile design. They are particularly advantageous for elliptic grid generation due to their numerical stability and efficient (derivative) evaluation (Thompson et al., 1985; Warsi, 1986; Bartels et al., 1987). Currently, only few grid generation systems utilize NURBS.

Grid generation is concerned with the problem of discretizing surfaces and volumes in three-dimensional space. The National Grid Project system, which is currently under development at the NSF Engineering Research Center for Computational Field Simulation at Mississippi State University, utilizes a two-stage grid generation methodology. For each NURBS surface, these two stages are (i) the initial generation of boundary curve grids and (ii) the computation of grid points in the surface's interior. The construction of the desired surface grid is done in two steps. First, an algebraic grid is constructed by interpolation from the boundary of the surface (transfinite interpolation). Second, this grid is smoothed and possibly modified by an iterative procedure. The most successful iterative smoothing schemes are based on elliptic systems of partial differential equations that relate the physical ("x, y, z") and computational ("ξ, η") variables. The elliptic system may be applied to the boundary grids, the interior grids, or both. The elliptic system may preserve the original distribution of grid points or redistribute points, which is commonly done in adaptive grid generation. Orthogonality of the grid may be imposed along certain boundary components of the physical region.

The process of algebraic grid generation for a parametrically defined surface is divided into three steps: (i) forward mapping; (ii) grid generation; and (iii) backward mapping. The forward mapping is the mapping of the three-dimensional physical surface (i.e., a NURBS surface) to a two-dimensional parametric rectangle. Once the forward mapping is completed, the grid is generated in the parametric space using transfinite interpolation (TFI) and then mapped back into physical space. It is important to understand that the type of parametrization does not change the shape of the surface, but it does change the distribution of the points on it. A "poor" parametrization may cause the surface grid to be highly skewed.

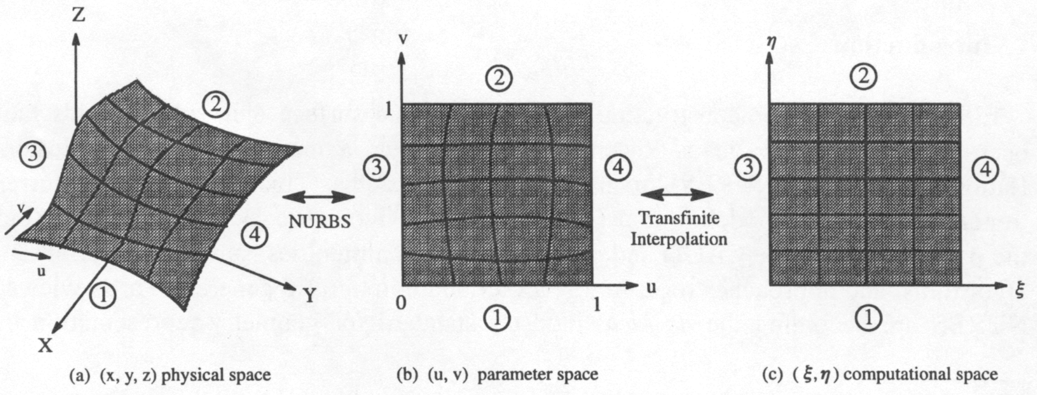
Z
Y
X
v
u
②
③
④
①
NURBS
(a) (x, y, z) physical space
v
1
0
1
u
②
③
④
①
(b) (u, v) parameter space
Transfinite
Interpolation
η
ξ
②
③
④
①
(c) (ξ, η) computational space

Fig. 1. Mapping from physical ("x, y, z") to computational ("ξ, η") space via parametric ("u, v") space.

In classical differential geometry, a surface is viewed as a mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$. Consequently, parametric surfaces (physical variables) are defined in terms of parametric variables. In grid generation, the parametric variables are defined in terms of computational variables, i.e.,

$$s = (x, y, z) = \big(x(u, v), y(u, v), z(u, v)\big) \quad \text{and} \quad (u, v) = \big(u(\xi, \eta), v(\xi, \eta)\big) \tag{1.1}$$

(see Fig. 1).

The construction of an $m \times n$ algebraic grid on a physical surface starts either with the specification of the boundary distribution along the physical boundaries (then mapped to the parametric boundaries) or with the distribution of grid points directly on the parametric boundaries. The parametric values used are denoted by

$$\mathcal{M}_u = \{u_{1,j} < u_{2,j} < \cdots < u_{m,j} \mid j = 1, n\} \quad \text{and} \tag{1.2}$$

$$\mathcal{M}_v = \{v_{i,1} < v_{i,2} < \cdots < v_{i,n} \mid i = 1, m\}. \tag{1.3}$$

The first step in constructing a parametric grid requires a correspondence between the parametric space (u, v) and the computational space (ξ, η) *via* a distribution space (s, t). A uniform spacing in computational space is used, i.e., $\xi = 1, 2, \ldots, m$ and $\eta = 1, 2, \ldots, n$. Therefore, the computational grid, denoted by $\mathcal{G}_{\xi\eta}$, is defined by integer coordinates (i, j), where $i = \xi$ and $j = \eta$. In the next step, the boundary distribution of the (u, v) space is mapped to the boundaries of the (s, t) space (see Fig. 2). This is usually done using an arc length mapping, which is given by the values

$$s_{i,j} = \frac{\displaystyle\sum_{k=2}^{i} \sqrt{(u_{k,j} - u_{k-1,j})^2 + (v_{k,j} - v_{k-1,j})^2}}{\displaystyle\sum_{k=2}^{m} \sqrt{(u_{k,j} - u_{k-1,j})^2 + (v_{k,j} - v_{k-1,j})^2}} \tag{1.4}$$

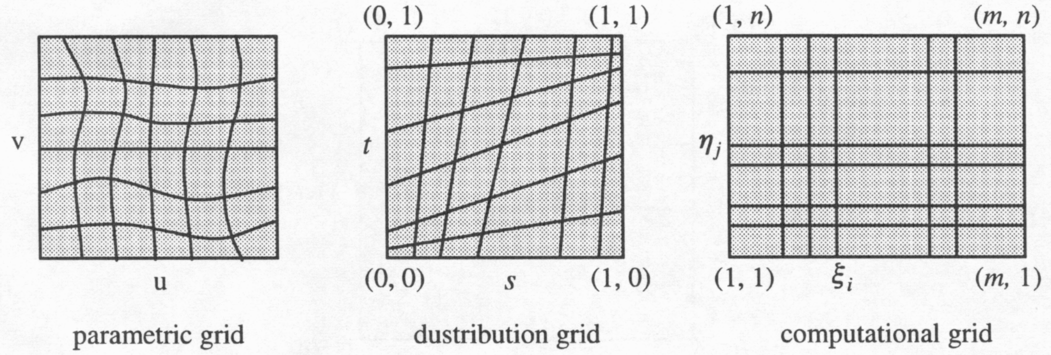

Fig. 2. Mapping from parametric ("u, v") to computational ("ξ, η") space via distribution ("s, t") space.

for $i = 2, \ldots, m$, $j = 1, n$ and

$$t_{i,j} = \frac{\sum_{k=2}^{j} \sqrt{(u_{i,k} - u_{i,k-1})^2 + (v_{i,k} - v_{i,k-1})^2}}{\sum_{k=2}^{n} \sqrt{(u_{i,k} - u_{i,k-1})^2 + (v_{i,k} - v_{i,k-1})^2}} \tag{1.5}$$

for $i = 1, m$, $j = 2, \ldots, n$. It must be noted that $s_{1,1} = 0$ and $t_{1,1} = 0$.

Let

$$\mathcal{M}_s = \{0 = s_{1,j} < s_{2,j} < \cdots < s_{m,j} = 1 \mid j = 1, n\} \tag{1.6}$$

be a partitioning of the s-domain $[0, 1]$ and

$$\mathcal{M}_t = \{0 = t_{i,1} < t_{i,2} < \cdots < t_{i,n} = 1 \mid i = 1, m\} \tag{1.7}$$

a partitioning of the t-domain $[0, 1]$. The distribution grid $\mathcal{G}_{st} = \mathcal{M}_s \otimes \mathcal{M}_t$ partitioning $[0, 1] \times [0, 1]$ is generated by solving for the interior points using

$$s = \frac{s_1 + t_1(s_2 - s_1)}{1 - (s_2 - s_1)(t_2 - t_1)} \quad \text{and} \tag{1.8}$$

$$t = \frac{t_1 + s_1(t_2 - t_1)}{1 - (s_2 - s_1)(t_2 - t_1)} \tag{1.9}$$

(see Fig. 3).

Once the distribution grid is generated, by some standard technique such as tensor product interpolation or TFI (Farin, 1992), the parametric grid $\mathcal{G}_{uv} = \mathcal{M}_u \oplus \mathcal{M}_v$ is constructed, where the parametric values $(u_{i,j}, v_{i,j})$ correspond to the computational nodes $(s_{i,j}, t_{i,j})$. Finally, the surface grid $\mathcal{G}_{xyz}$ is constructed using the physical coordinates of the NURBS representation, i.e.,

$$\boldsymbol{s}_{i,j} = \boldsymbol{s}(u_{i,j}, v_{i,j}), \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \tag{1.10}$$

The most difficult and least developed area in elliptic grid generation is the computation of surface grids. Not only needs the grid to be smoothed but the grid points must stay on

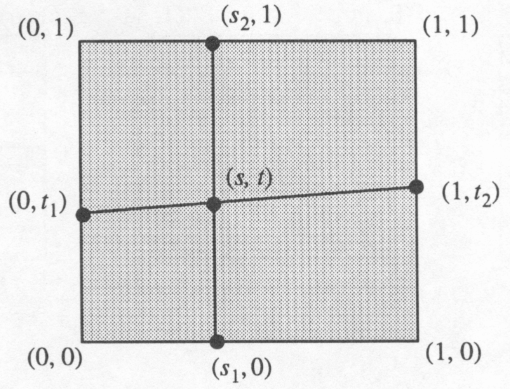

Fig. 3. Construction of distribution grid.

the surface. The simplest technique to create a grid is to work in parametric space rather than in the physical variables of the surface. There are some disadvantages associated with this approach. The differential equations become more complicated and contain two sets of derivatives, the derivatives of the physical variables with respect to the parametric variables ("$x_u, x_v, y_u, y_v, z_u, z_v, x_{uu}, x_{uv}, x_{vv}, \ldots$") and the derivatives of the parametric variables with respect to the computational variables ("$u_\xi, u_\eta, v_\xi, v_\eta, u_{\xi\xi}, u_{\xi\eta}, u_{\eta\eta}, \ldots$").

The advantage of using a NURBS-based geometry definition is the ability to represent both standard analytic shapes (e.g., conics, quadrics, surfaces of revolution, etc.) and free-form curves and surfaces. Therefore, both analytic and free-form shapes are represented precisely, and a unified database can store both. Also, the derivatives of the physical coordinates with respect to the parametric values can be evaluated analytically using the NURBS definition.

This paper discusses some of the current problems in elliptic grid generation on surfaces and describes new features and enhancements that have recently been developed. Grid orthogonality is of particular interest. Although orthogonal grids are generally not necessary for numerical algorithms, there must be some upper bound on the skewness of the grid to limit numerical errors of simulations. Boundary orthogonality is desirable whenever it is necessary to impose Neumann-type boundary conditions on one or more of the physical variables of the problem. Some of the popular ways of treating Neumann boundary conditions are not accurate when the grid is not orthogonal, and others become unstable if the grid is extremely skewed near the boundary (Thompson et al., 1985).

B-spline and NURBS curves and surfaces are described in (Bartels et al., 1987; Piegl, 1991; Farin, 1993). Fundamental concepts of numerical grid generation can be found in (Mastin and Thompson, 1984; Takagi and Miki, 1985; Thompson et al., 1985; Warsi, 1986; George, 1991). Finite difference schemes for the solution of partial differential equations are discussed in (Strikwerda, 1989).

2. Definition and evaluation of NURBS curves and surfaces

It is assumed that the surfaces for which grids are to be constructed are given as NURBS surfaces. This kind of surface/geometry representation has emerged as a *de facto* industry standard over the last years. Since elliptic grid generation requires positional as well as derivative information of surfaces, a short overview regarding the definition of NURBS curves and surfaces is given. For a more detailed discussion of NURBS curves and surfaces, it is referred to (Bartels et al., 1987; Piegl, 1991; Farin, 1993).

2.1. NURBS curves

A NURBS curve $\boldsymbol{s}(u) = (x(u), y(u), z(u))$ is a piecewise rational curve given as

$$\boldsymbol{s}(u) = \frac{\boldsymbol{x}(u)}{\omega(u)} = \frac{\sum\limits_{i=0}^{m} \omega_i \boldsymbol{d}_i N_i^k(u)}{\sum\limits_{i=0}^{m} \omega_i N_i^k(u)}, \quad u \in [u_{k-1}, u_{m+1}], \tag{2.1}$$

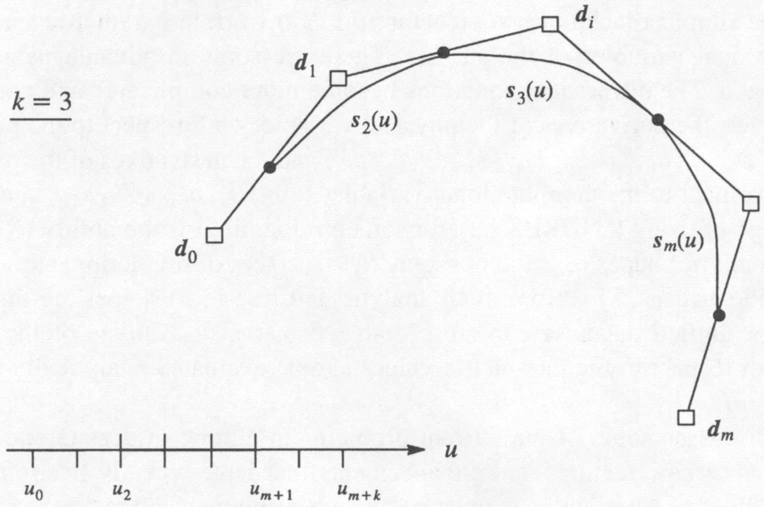

Fig. 4. Control information of NURBS curve.

defined by

- an order k,
- control points $\boldsymbol{d}_i = (x_i, y_i, z_i)$, $i = 0, \ldots, m$,
- real weights ω_i, $i = 0, \ldots, m$,
- a set of real knots $\{u_0, \ldots, u_{m+k} \mid u_i \leqslant u_{i+1},\ i = 0, \ldots, (m+k-1)\}$,
- B-spline basis functions $N_i^k(u)$, $u \in [u_i, u_{i+k}]$, $i = 0, \ldots, m$, where

$$N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u),$$

$$N_i^1(u) = \begin{cases} 1, & \text{if } u_i \leqslant u < u_{i+1}, \\ 0, & \text{otherwise,} \end{cases} \qquad i = 0, \ldots, m, \text{ and}$$

- curve segments $\boldsymbol{s}_i(u)$, $u \in [u_i, u_{i+1}]$, $i = (k-1), \ldots, m$, (see Fig. 4).

2.2. *NURBS surfaces*

A NURBS surface $\boldsymbol{s}(u,v) = (x(u,v), y(u,v), z(u,v))$ is a piecewise rational surface given as

$$\begin{aligned} \boldsymbol{s}(u,v) &= \frac{\boldsymbol{x}(u,v)}{\omega(u,v)} \\ &= \frac{\displaystyle\sum_{j=0}^{n}\sum_{i=0}^{m} \omega_{i,j}\ \boldsymbol{d}_{i,j}\ N_i^k(u) N_j^l(v)}{\displaystyle\sum_{j=0}^{n}\sum_{i=0}^{m} \omega_{i,j} N_i^k(u) N_j^l(v)}, \\ & u \in [u_{k-1}, u_{m+1}],\ v \in [v_{l-1}, v_{n+1}], \end{aligned} \tag{2.2}$$

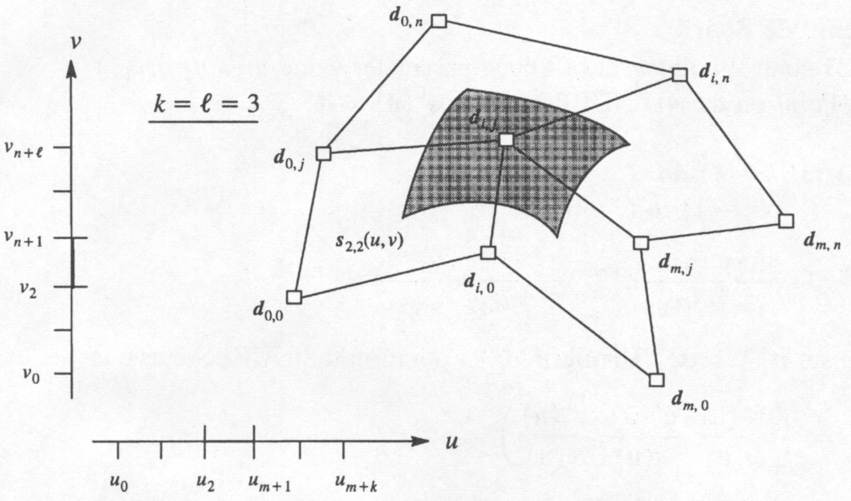

Fig. 5. Control information of NURBS surface.

defined by

- two orders k and l,
- control points $\boldsymbol{d}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$, $i = 0,\dots,m$, $j = 0,\dots,n$,
- real weights $\omega_{i,j}$, $i = 0,\dots,m$, $j = 0,\dots,n$,
- a set of real u-knots, $\{u_0,\dots,u_{m+k} \mid u_i \leqslant u_{i+1},\ i = 0,\dots,(m+k-1)\}$,
- a set of real v-knots, $\{v_0,\dots,v_{n+l} \mid v_j \leqslant v_{j+1},\ j = 0,\dots,(n+l-1)\}$,
- B-spline basis functions $N_i^k(u)$, $u \in [u_i, u_{i+k}]$, $i = 0,\dots,m$,
- B-spline basis functions $N_j^l(v)$, $v \in [v_j, v_{j+l}]$, $j = 0,\dots,n$, and
- surface segments $\boldsymbol{s}_{i,j}(u,v)$, $u \in [u_i, u_{i+1}]$, $i = (k-1),\dots,m$, $v \in [v_j, v_{j+1}]$, $j = (l-1),\dots,n$,

(see Fig. 5).

The indexing scheme for NURBS curves and surfaces follows the notation used in (Bartels et al., 1987). Very often, it is convenient to represent the rational expression for a 3D NURBS curve/surface in the corresponding homogeneous, 4D representation. This requires an evaluation scheme for the 4D representation.

2.3. *The homogeneous de Boor algorithm*

Representing a NURBS curve in its homogeneous form introduces a fourth coordinate. The homogeneous form of a NURBS curve is given as

$$\boldsymbol{s}^*(u) = \big(x^*(u), y^*(u), z^*(u), \omega(u)\big) = \sum_{i=0}^{m} \boldsymbol{d}_i^*\ N_i^k(u), \quad u \in [u_{k-1}, u_{m+1}], \tag{2.3}$$

where $\boldsymbol{d}_i^* = (\omega_i x_i, \omega_i y_i, \omega_i z_i, \omega_i)$ is a 4D control point of a 4D B-spline curve. Instead of using the recursively defined basis functions $N_i^k(u)$ for the computation of a curve point, one commonly uses the numerically more stable "de Boor algorithm" (Farin, 1993). This is an elegant algorithm based on repeated linear interpolation of line segments of the original control polygon. The algorithm can be used to compute a 4D curve point $\boldsymbol{s}^*(u)$:

Algorithm "de Boor"
Input: 4D control points, set of knots, parameter value $u \in [u_I, u_{I+1})$
Output: Point on the 4D NURBS curve, $\boldsymbol{s}^*(u) = \boldsymbol{d}^{*\,k-1}_{I-(k-1)}$

```
for $r = 1$ to $(k-1)$ do
    for $i = I-(k-1)$ to $(I-r)$ do
        $\boldsymbol{d}_i^{*\,r} = \frac{u_{i+k}-u}{u_{i+k}-u_{i+r}}\boldsymbol{d}_i^{*\,r-1} + \frac{u-u_{i+r}}{u_{i+k}-u_{i+r}}\boldsymbol{d}_{i+1}^{*\,r-1};$
```

Here, we set $\boldsymbol{d}_i^{*\,0} = \boldsymbol{d}_i^*$. The desired 3D point on the NURBS curve is

$$\boldsymbol{s}(u) = \left(\frac{x^*(u)}{\omega(u)}, \frac{y^*(u)}{\omega(u)}, \frac{z^*(u)}{\omega(u)}\right). \tag{2.4}$$

This algorithm can be utilized in a similar way to compute 4D surface points. The de Boor algorithm must be applied to all control point rows in the 4D control net and then once to the set of "intermediate" curve control points (one column). This tensor product evaluation scheme is explained in (Farin, 1993). At the end, a 4D surface point is projected into 3D space.

2.4. *Derivatives of NURBS curves and surfaces*

Derivative information is required for the generation of smooth grids. Therefore, the relevant NURBS curve and surface derivatives are listed. The first and second derivatives of a NURBS curve $\boldsymbol{s}(u)$ are given by

$$\frac{\mathrm{d}}{\mathrm{d}u}\boldsymbol{s}(u) = \boldsymbol{s}_u(u) = \frac{\boldsymbol{x}_u(u)\omega(u) - \boldsymbol{x}(u)\omega_u(u)}{\omega^2(u)} \tag{2.5}$$

and

$$\frac{\mathrm{d}^2}{\mathrm{d}u^2}\boldsymbol{s}(u) = \boldsymbol{s}_{uu}(u) = \Big(\omega(u)(\boldsymbol{x}_{uu}(u)\omega(u) - \boldsymbol{x}(u)\omega_{uu}(u)) \\ -2\omega_u(u)(\boldsymbol{x}_u(u)\omega(u) - \boldsymbol{x}(u)\omega_u(u))\Big)\omega^{-3}(u) \tag{2.6}$$

One can use the de Boor algorithm to compute the terms appearing in these equations by considering the expression for the first derivative of a B-spline curve

$$\boldsymbol{s}(u) = \sum_{i=0}^{m} \boldsymbol{d}_i N_i^k(u),$$

which is

$$\frac{\mathrm{d}}{\mathrm{d}u}\boldsymbol{s}(u) = \sum_{i=0}^{m-1} \frac{k-1}{u_{i+k}-u_{i+1}}(\boldsymbol{d}_{i+1}-\boldsymbol{d}_i)N_{i+1}^{k-1}(u), \quad u \in [u_{k-1}, u_{m+1}]. \tag{2.7}$$

The required partial derivatives of a NURBS surface $\boldsymbol{s}(u,v)$ are given by

$$\begin{aligned}\frac{\partial}{\partial u}\boldsymbol{s}(u,v) = \boldsymbol{s}_u &= \frac{\boldsymbol{x}_u\omega - \boldsymbol{x}\omega_u}{\omega^2},\\ \boldsymbol{s}_v &= \frac{\boldsymbol{x}_v\omega - \boldsymbol{x}\omega_v}{\omega^2},\\ \boldsymbol{s}_{uu} &= \frac{\omega(\boldsymbol{x}_{uu}\omega - \boldsymbol{x}\omega_{uu}) - 2\omega_u(\boldsymbol{x}_u\omega - \boldsymbol{x}\omega_u)}{\omega^3},\\ \boldsymbol{s}_{vv} &= \frac{\omega(\boldsymbol{x}_{vv}\omega - \boldsymbol{x}\omega_{vv}) - 2\omega_v(\boldsymbol{x}_v\omega - \boldsymbol{x}\omega_v)}{\omega^3}, \quad \text{and}\\ \boldsymbol{s}_{uv} &= \frac{\omega(\boldsymbol{x}_{uv}\omega + \boldsymbol{x}_u\omega_v - \boldsymbol{x}_v\omega_u - \boldsymbol{x}\omega_{uv}) - 2\omega_v(\boldsymbol{x}_u\omega - \boldsymbol{x}\omega_u)}{\omega^3}.\end{aligned} \tag{2.8}$$

Several properties of algorithms for NURBS curves and surfaces are discussed in detail in (Piegl, 1991). This brief overview of NURBS curves and surfaces is sufficient for algebraic and elliptic grid generation.

3. Elliptic grid generation

A general method for constructing curvilinear structured grids is based on partial differential equations (Takagi and Miki, 1985; Thompson et al., 1985). The grid points are the solution of an elliptic partial differential equation system with Dirichlet boundary conditions on all boundaries. The use of elliptic systems has a number of advantages. The theory of partial differential equations guarantees that (for some elliptic systems) the mapping between physical and transformed regions will be one-to-one. Another important property is the inherent smoothness in the solution of elliptic systems. As a consequence, the smoothing tendencies allow grids to be generated for any physical configuration without any "overlapping" grid lines. Another advantage is the fact that slope discontinuities on the boundaries are not propagated into the field.

3.1. Elliptic surface equations

The elliptic equations for surface grids are a generalization of Laplace equations for harmonic mappings of planar regions. Let $\mathcal{S}$ be a simply-connected surface defined by the parametric equation

$$\boldsymbol{s} = \boldsymbol{s}(u,v), \quad 0 \leqslant u, v \leqslant 1. \tag{3.1}$$

Further, let u and v be functions of the computational variables ξ and η. A rectilinear grid in the computational square generates a curvilinear grid in the parametric square which maps to a curvilinear grid on the surface. Thus, a uniform grid in the computational space generates a curvilinear grid on the surface. The elliptic system of partial differential equations, which defines the transformation between computational variables and parametric variables, is related to conformal mappings on surfaces (Mastin and Thompson, 1984).

The derivation is based on conformal mappings. This approach utilizes the intrinsic orthogonality and uniformity properties that are inherent to a grid generated by a conformal mapping. Another advantage of using conformal coordinates on parametric surfaces is the fact that they allow solutions of differential equations on surfaces with the same ease as on a rectangle in the cartesian plane.

A surface grid generated by the conformal mapping of a rectangle onto the surface $\mathcal{S}$ is orthogonal and has a constant aspect ratio. These two conditions can be expressed mathematically by the system of equations

$$\boldsymbol{s}_\xi \cdot \boldsymbol{s}_\eta = 0 \quad \text{and} \tag{3.2}$$

$$M|\boldsymbol{s}_\xi| = |\boldsymbol{s}_\eta|, \tag{3.3}$$

where M is the grid aspect ratio. These two equations can be rewritten as

$$x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta = 0 \quad \text{and} \tag{3.4}$$

$$M^2\left(x_\xi^2 + y_\xi^2 + z_\xi^2\right) = x_\eta^2 + y_\eta^2 + z_\eta^2. \tag{3.5}$$

Using the chain rule for differentiation, the above system of equations can be expressed in the form

$$Mu_\xi = av_\eta - bu_\eta \quad \text{and} \tag{3.6}$$

$$Mv_\xi = bv_\eta - cu_\eta, \tag{3.7}$$

where

$$a = \frac{\overline{g}_{22}}{\overline{J}}, \quad b = -\frac{\overline{g}_{12}}{\overline{J}}, \quad c = \frac{\overline{g}_{11}}{\overline{J}},$$
$$\overline{g}_{11} = \boldsymbol{s}_u \cdot \boldsymbol{s}_u, \quad \overline{g}_{12} = \boldsymbol{s}_u \cdot \boldsymbol{s}_v, \quad \overline{g}_{22} = \boldsymbol{s}_v \cdot \boldsymbol{s}_v, \quad \text{and}$$
$$\overline{J} = \sqrt{\overline{g}_{11}\overline{g}_{22} - \overline{g}_{12}^2}.$$

If the parametric and computational variables are exchanged in (3.6) and (3.7) so that the parametric variables become the independent variables, the system becomes

$$-M\eta_u = b\xi_u + c\xi_v \quad \text{and} \tag{3.8}$$

$$M\eta_v = a\xi_u + b\xi_v. \tag{3.9}$$

This first-order system is analogous to Beltrami's system of equations for the quasi-conformal mapping of planar regions. It follows that the computational variables ξ and η are solutions of the following second-order linear homogeneous elliptic system ($\Phi = \Psi = 0$):

$$\overline{g}_{22}\xi_{uu} - 2\overline{g}_{12}\xi_{uv} + \overline{g}_{11}\xi_{vv} + (\Delta_2 u)\xi_u + (\Delta_2 v)\xi_v = \Phi \quad \text{and} \tag{3.10}$$

$$\overline{g}_{22}\eta_{uu} - 2\overline{g}_{12}\eta_{uv} + \overline{g}_{11}\eta_{vv} + (\Delta_2 u)\eta_u + (\Delta_2 v)\eta_v = \Psi. \tag{3.11}$$

The Beltramians $\Delta_2 u$ and $\Delta_2 v$ are given by

$$\Delta_2 u = \overline{J}(a_u + b_v) = \overline{J}\left[\frac{\partial}{\partial u}\left(\frac{\overline{g}_{22}}{\overline{J}}\right) - \frac{\partial}{\partial v}\left(\frac{\overline{g}_{12}}{\overline{J}}\right)\right] \quad \text{and} \tag{3.12}$$

$$\Delta_2 v = \overline{J}(b_u + c_v) = \overline{J}\left[\frac{\partial}{\partial v}\left(\frac{\overline{g}_{11}}{\overline{J}}\right) - \frac{\partial}{\partial u}\left(\frac{\overline{g}_{12}}{\overline{J}}\right)\right]. \tag{3.13}$$

This system is the basis for elliptic methods generating surface grids. The source terms (or control functions) Φ and Ψ are added to allow control over the distribution of grid points on the surface. Typically, the points in the computational space are given, and the points in the parametric space must be computed. Therefore, it is convenient to exchange variables again so that the computational variables ξ and η are the independent variables. The transformation given by (3.10) and (3.11) is reduced to the following system of equations for which the parametric variables u and v are solutions of the following quasi-linear elliptic system:

$$g_{22}(u_{\xi\xi} + \mathcal{P}u_\xi) - 2g_{12}u_{\xi\eta} + g_{11}(u_{\eta\eta} + \mathcal{Q}u_\eta) = J^2\Delta_2 u \quad \text{and} \tag{3.14}$$

$$g_{22}(v_{\xi\xi} + \mathcal{P}v_\xi) - 2g_{12}v_{\xi\eta} + g_{11}(v_{\eta\eta} + \mathcal{Q}v_\eta) = J^2\Delta_2 v, \tag{3.15}$$

where

$$g_{11} = \overline{g}_{11}u_\xi^2 + 2\overline{g}_{12}u_\xi v_\xi + \overline{g}_{22}v_\xi^2,$$
$$g_{12} = \overline{g}_{11}u_\xi u_\eta + \overline{g}_{12}(u_\xi v_\eta + u_\eta v_\xi) + \overline{g}_{22}v_\xi v_\eta,$$
$$g_{22} = \overline{g}_{11}u_\eta^2 + 2\overline{g}_{12}u_\eta v_\eta + \overline{g}_{22}v_\eta^2,$$
$$\mathcal{P} = \frac{\overline{J}J^2}{g_{22}}\Phi, \quad \mathcal{Q} = \frac{\overline{J}J^2}{g_{11}}\Psi, \quad \text{and} \quad J = u_\xi v_\eta - u_\eta v_\xi.$$

The functions $\mathcal{P}$ and $\mathcal{Q}$ are used to control the grid point distribution.

The use of conformal mappings for mapping rectangular regions onto surfaces and deriving the elliptic generation system ((3.14) and (3.15)) has been discussed. Solving the generation system is accomplished using a finite difference discretization and an iterative method. For a more detailed discussion about this approach, it is referred to (Khamayseh, 1994). Some of the theoretical results from harmonic mappings of planar regions can be verified for solutions of the above system. The same generation system can be derived from a differential geometric point of view (Warsi, 1986). The system of partial differential equations may be solved with either Dirichlet- or Neumann-type boundary conditions, depending on whether the grid points on the boundary of the surface are to remain fixed or are allowed to move.

3.2. Control functions

The control functions $\mathcal{P}$ and $\mathcal{Q}$ must be selected so that the grid has the required distribution of grid points on the surface. Using $\mathcal{P} = \mathcal{Q} = 0$ tends to generate a grid with a uniform spacing. In most cases, there is a need to concentrate points in a certain area on the surface such as along particular boundary segments. The first step of computing control functions is the estimation of $\mathcal{P}$ and $\mathcal{Q}$ from some initial grid. The derivative terms in the elliptic system can be computed from given grid points, leaving two unknowns, $\mathcal{P}$ and $\mathcal{Q}$, which can be determined from the two equations (3.14) and (3.15). These control function values are smoothed so that the final elliptic grid will be smoother as well as "more orthogonal" than the initial grid.

The control functions can be computed from an initial grid only if the Jacobian of the transformation from computational to parametric space is nonvanishing. Since this may not always be the case for interpolated grids, there is the option of computing the control functions from the boundary distribution. The appropriate values for the control functions on the boundary can be derived by assuming that the grid lines are orthogonal to the boundary and the spacing normal to the boundary is uniform. The development follows that for two-dimensional planar regions (Thompson et al., 1985). The formulas for $\mathcal{P}$ and $\mathcal{Q}$ in the surface equations (3.14) and (3.15) are

$$\mathcal{P} = -\frac{s_{\xi\xi}}{s_\xi} + s_\xi \kappa_1 \quad \text{and} \tag{3.16}$$

$$\mathcal{Q} = -\frac{s_{\eta\eta}}{s_\eta} + s_\eta \kappa_2. \tag{3.17}$$

The variable s is the arc length along the boundary of the surface, and κ_1 and κ_2 denote the curvatures of the boundary curves $\xi =$ constant and $\eta =$ constant, respectively. The curvatures are

$$\kappa_1 = -\sqrt{\frac{g}{g_{22}^3}}\varGamma_{22}^1 \quad \text{and} \quad \kappa_2 = \sqrt{\frac{g}{g_{11}^3}}\varGamma_{11}^2,$$

where the Christoffel symbols $\varGamma_{ij}^k$ are given by

$$\varGamma_{11}^2 = \frac{1}{2g}\left[g_{11}\left(2\frac{\partial g_{12}}{\partial \xi} - \frac{\partial g_{11}}{\partial \eta}\right) - g_{12}\frac{\partial g_{11}}{\partial \xi}\right],$$

$$\varGamma_{22}^1 = \frac{1}{2g}\left[g_{22}\left(2\frac{\partial g_{12}}{\partial \eta} - \frac{\partial g_{22}}{\partial \xi}\right) - g_{12}\frac{\partial g_{22}}{\partial \eta}\right], \quad \text{and}$$

$$g = g_{11}g_{22} - (g_{12})^2.$$

The arc length derivatives for $\mathcal{P}$ are evaluated on the $\eta =$ constant boundary curves and then interpolated in the interior of the computational region, while the arc length derivatives of $\mathcal{Q}$ are evaluated on $\xi =$ constant and then interpolated in the interior. In a similar manner, the curvature of each control function can be computed on opposite boundaries and then interpolated. The curvatures cannot be computed solely from information on the boundary curves. They also depend on information off the boundaries. All arc length derivatives and curvatures, and therefore $\mathcal{P}$ and $\mathcal{Q}$, can be computed throughout the computational region in this manner. This second control option can be used to determine the distribution of grid points on a surface, even if the initial grid is of such poor quality that it cannot be used itself to generate appropriate control functions.

3.3. Boundary orthogonality

Imposing boundary orthogonality for the physical surface grid allows grid points to move along the boundary. The appropriate boundary conditions must be imposed on the elliptic system in the parametric region so that the grid is orthogonal on the surface. Two grid lines will be orthogonal on the surface if the inner product $\boldsymbol{s}_\xi \cdot \boldsymbol{s}_\eta$ vanishes. This can be expanded using the chain rule yielding the equation

$$\overline{g}_{11}u_\xi u_\eta + \overline{g}_{22}v_\xi v_\eta + \overline{g}_{12}(u_\xi v_\eta + u_\eta v_\xi) = 0. \tag{3.18}$$

The orthogonality condition is used to formulate derivative boundary conditions for the elliptic system. If the boundary curves $u = 0$ and $u = 1$ are considered, the orthogonality condition reduces to

$$\overline{g}_{22}v_\xi + \overline{g}_{12}u_\xi = 0. \tag{3.19}$$

Similarly, orthogonality for the curves $v = 0$ and $v = 1$ is imposed by

$$\overline{g}_{11}u_\eta + \overline{g}_{12}v_\eta = 0. \tag{3.20}$$

The elliptic system, the given u and v values on the boundary, and the orthogonality conditions form a mixed boundary value problem. Equation (3.19) determines the values for v along the boundary, where $u = 0$ and $u = 1$, and equation (3.20) determines the values for u, where $v = 0$ and $v = 1$.

The above discussion shows how derivative boundary conditions can be used to impose orthogonality. This requires "moving" grid points along the boundary of the surface. Orthogonality can also be imposed by adjusting the control functions near the boundary and keeping the boundary points fixed. Since there are two control functions, not only can the angle at the boundary be chosen but the distance to the first grid line from the boundary as well. Assuming that the grid lines are orthogonal, the values for $\mathcal{P}$ and $\mathcal{Q}$ can be determined from the elliptic system (3.14) and (3.15) along a boundary segment, where a specified grid spacing is desired. The distance is used for the value of g_{11} or g_{22}. Some of the other derivatives in the equations for $\mathcal{P}$ and $\mathcal{Q}$ can be computed from the fixed values of u and v on the boundary. The derivatives that must be computed using interior values are updated during the iterative solution procedure of the elliptic system. At each iteration, the boundary control functions are smoothly blended with the interior control functions to generate a smooth grid that is orthogonal and has a specified spacing along assigned boundary segments.

4. Examples

Five examples are presented to demonstrate the method. The first three examples (Figs. 6, 7 and 8) show an arbitrary bicubic NURBS surface, a torus (periodic case), and a cone (singular case). Each figure contains the shaded NURBS surface with its defining control points and the algebraic and elliptically smoothed surface grid. The algebraic grids are constructed using transfinite interpolation of the boundary curves. The elliptic grids are generated using the algebraic grids and solving the elliptic system. Clearly, the elliptic grid is smoother and "more uniform".

The next two examples show algebraic and elliptically smoothed surface grids on real-world geometries. The first geometry (Fig. 9) shows the space shuttle. The elliptic grid is generated using control functions $\mathcal{P} = \mathcal{Q} = 0$, and the boundary points are allowed to move so that the grid is orthogonal on the boundary. Thus, the initial spacing on the boundaries is not maintained. The second geometry (Fig. 10) shows an F15 aircraft having a degenerate boundary (edge collapsed to a point). The elliptic grid has been generated using control functions computed from the algebraic grid by solving equations (3.14) and (3.15) for $\mathcal{P}$ and $\mathcal{Q}$. Orthogonality is achieved on the boundaries by using

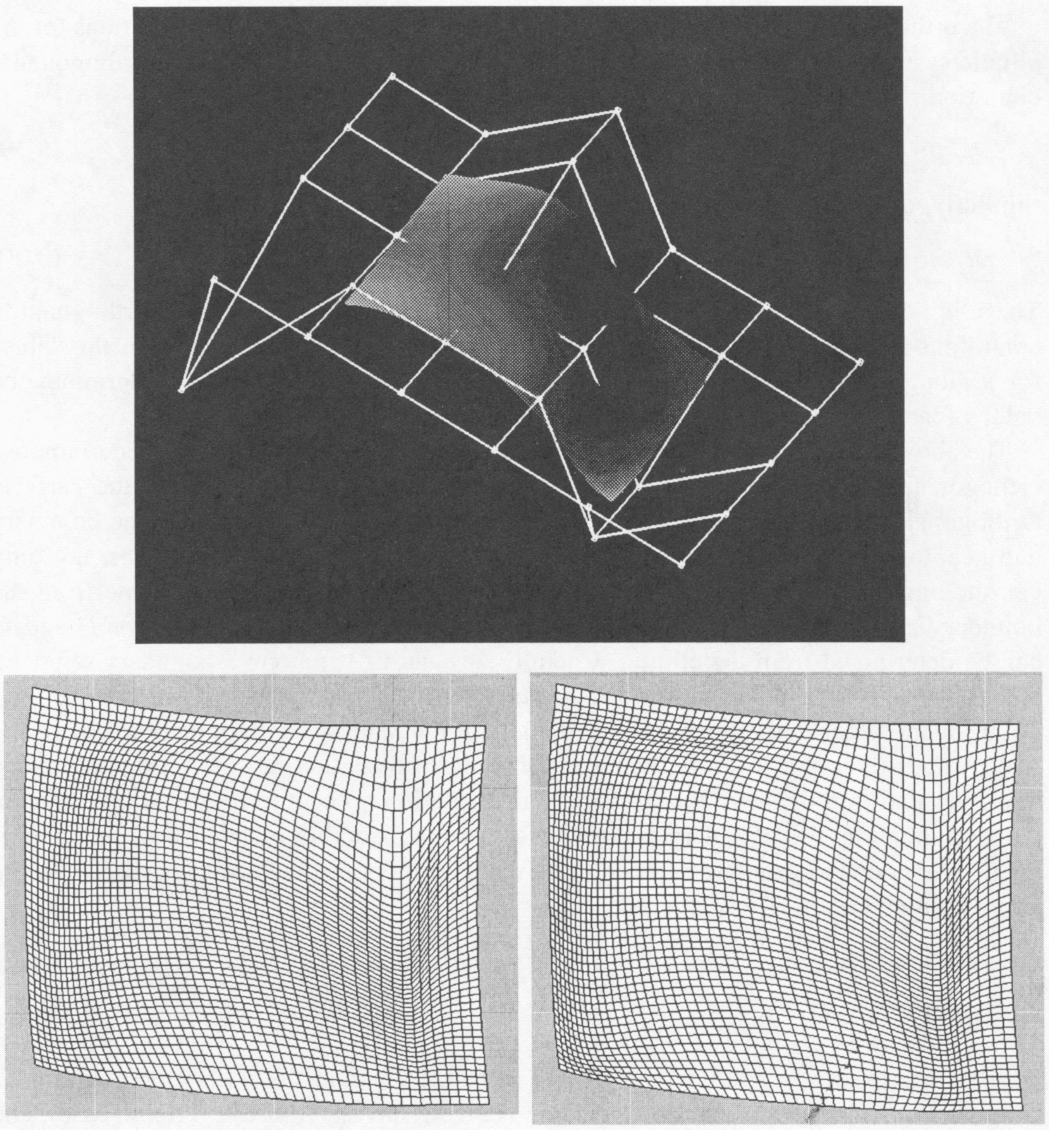

Fig. 6. Bicubic surface (shaded surface (top), algebraic grid (bottom-left), and elliptic grid (bottom-right)).

the second type of orthogonality, where grid lines are orthogonal to the boundaries and the grid points distribution on the boundaries remains unchanged. Orthogonality is not applied on the singular boundary.

Unfortunately, the described grid generation strategy has one drawback when applied to NURBS surfaces directly. If the number of grid points to be generated on a NURBS surface is significantly smaller than the number of control points $\boldsymbol{d}_{i,j}$, convergence problems may arise. In the implementation, this problem is solved by first evaluating the given NURBS surface at a resolution similar to the number of grid points to be computed. A bicubic B-spline surface is determined interpolating the points on the original NURBS

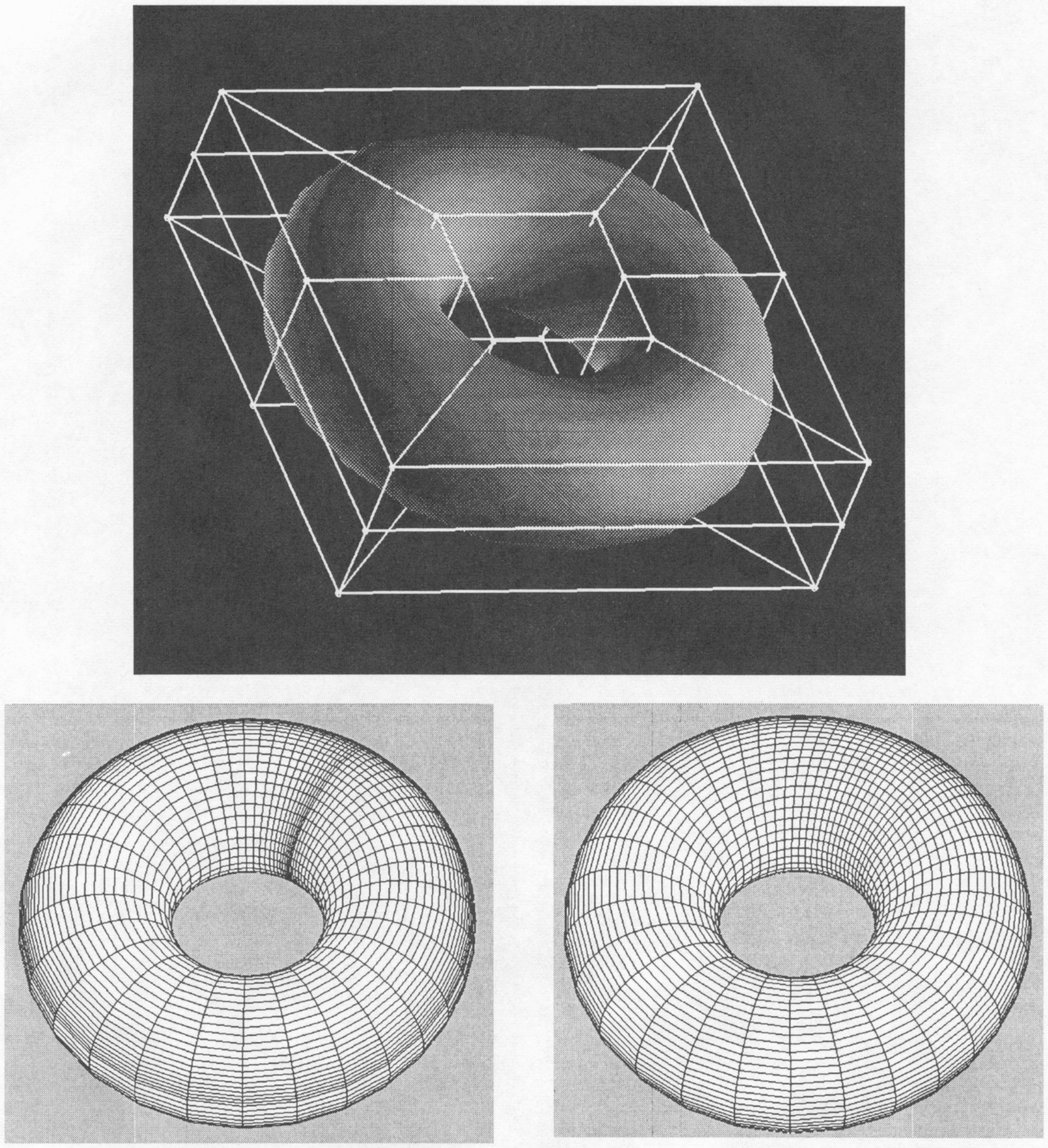

Fig. 7. Torus (shaded surface (top), algebraic grid (bottom-left), and elliptic grid (bottom-right)).

surface. A surface grid is then computed for the B-spline surface, and the values for u and v are finally used to evaluate the original NURBS surface. It has been found that this approach works well for all practical applications that have been tested.

5. Conclusion

This paper shows how grid generation can be applied directly to NURBS surfaces and how the derivative information of NURBS is utilized. The elliptic generation system is

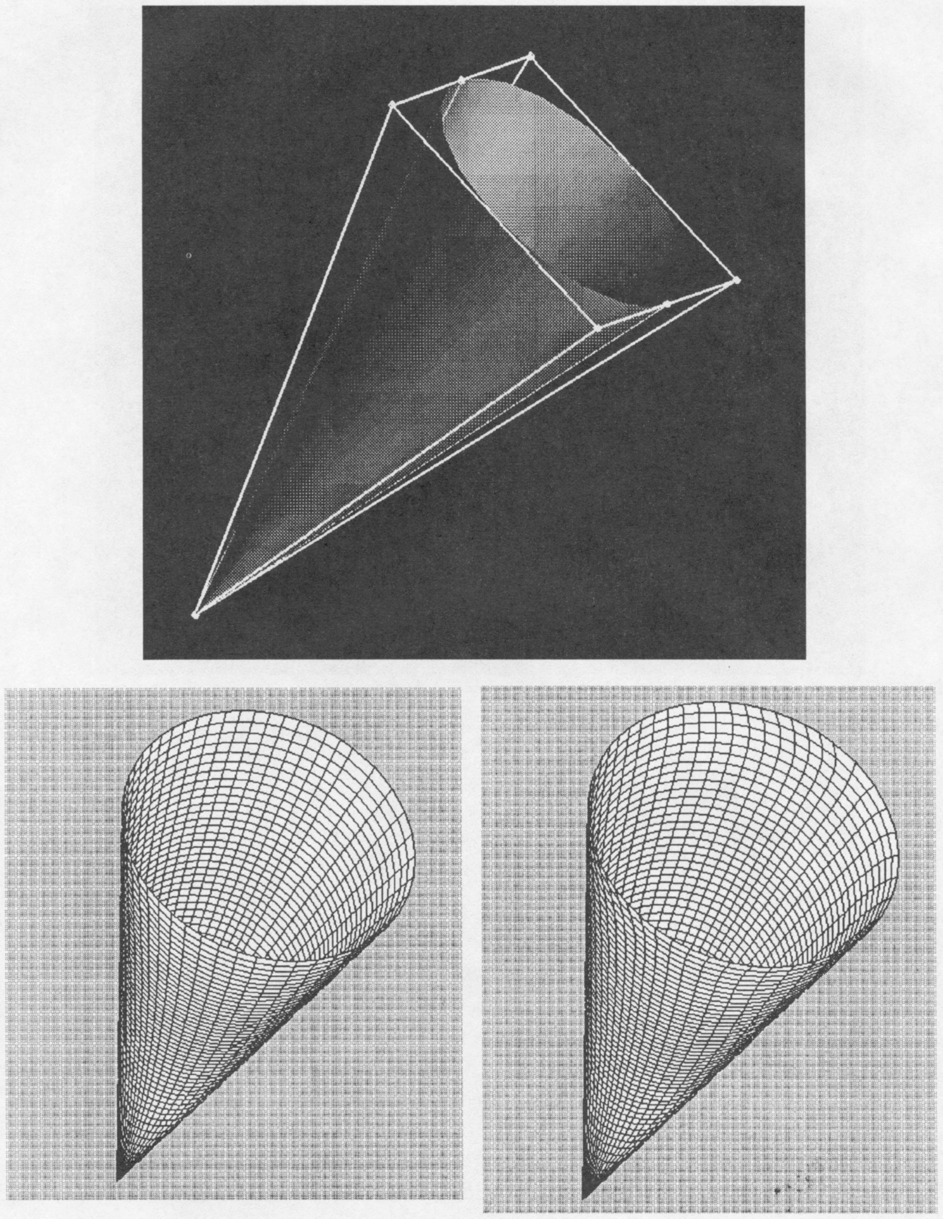

Fig. 8. Cone (shaded surface (top), algebraic grid (bottom-left), and elliptic grid (bottom-right)).

solved using a finite difference discretization and the successive-over-relaxation (SOR) method. The control function and orthogonality techniques allow a great degree of freedom in designing surface grids. Orthogonality on the boundaries is enforced by checking if the boundary is regular, degenerate, or periodic. The two techniques for achieving orthogonality are applied on regular boundaries and are applied on one boundary at a time. If the boundary is degenerate, orthogonality is not applied, but if a periodic boundary

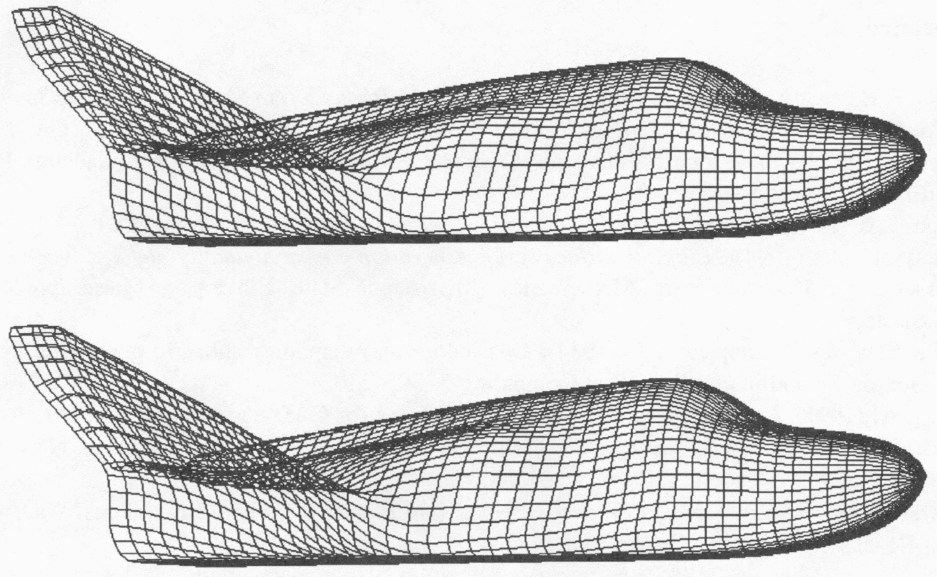

Fig. 9. Space shuttle (algebraic grid (top) and elliptic grid (bottom)).

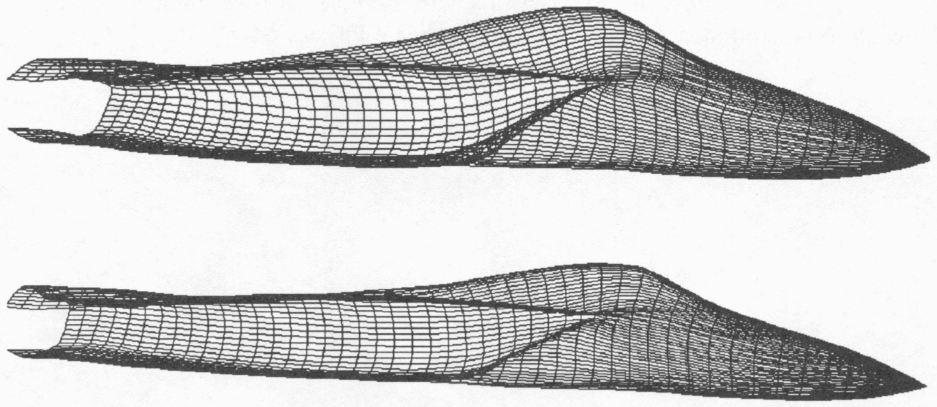

Fig. 10. F15 aircraft (algebraic grid (top) and elliptic grid (bottom)).

is encountered, the SOR method is also applied on the common boundary to ensure a smooth transition along the common boundaries.

Acknowledgements

Special thanks go to Brian A. Jean and Holly Beeland for preparing the figures and examples. We thank C. Wayne Mastin, Bharat K. Soni, Joe F. Thompson and Z.U.A. Warsi for their advice and support.

References

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

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

George, L.P. (1991), *Automatic Mesh Generation*, Wiley and Sons, New York, NY.

Khamayseh, A. (1994), *Elliptic surface grid generation on analytically defined geometries*, Dissertation, Department of Mathematics, Mississippi State University, Mississippi State, Mississippi.

Mastin, C.W. and Thompson, J.F. (1984), Quasiconformal mappings and grid generation, SIAM Journal on Scientific and Statistical Computing 5, 305–310.

Piegl, L.A. (1991), Rational B-spline curves and surfaces for CAD and graphics, in: D.F. Rogers and R.A. Earnshaw, eds., *State of the Art in Computer Graphics*, Springer-Verlag, New York, NY, 225–269.

Strikwerda, J.C. (1989), *Finite Difference Schemes and Partial Differential Equations*, Wadsworth and Brooks/Cole, Pacific Grove, CA, 1989.

Takagi, T. and Miki, K. (1985), Numerical generation of boundary-fitted curvilinear coordinate systems for arbitrarily curved surfaces, Journal of Computational Physics 58, 67–79.

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

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