30

NURBS in Structured Grid Generation

Tzu-Yi Yu

Bharat K. Soni

30.1 Introduction

The parametric-based *nonuniform rational B-spline* (NURBS) is a widely utilized representation for geometrical entities in CAD/CAM/CAE systems. The convex hull, local support, shape-preserving forms, affine invariance, and variation diminishing properties of NURBS are extremely attractive in engineering design applications. These properties with associated mathematical and numerical development of NURBS, including evaluation algorithms and knot insertion and degree elevation schemes, are described in detail in Chapters 27 and 28 of this handbook. The first commercial product that used the NURBS to represent geometry came from the Structural Dynamics Research Cooperation (SDRC) in 1983. The Boeing company proposed the NURBS as an IGES (initial graphics exchange specification) standard in 1981, and now the NURBS curve and NURBS surface have been adopted as the IGES geometric entities 126 and 128. The IGES format has become the *de facto* standard IO (input/output) for exchanging data between various CAD/CAM and CAE systems. Recently, the IGES entities 126 and 128 have become increasingly popular in grid (mesh) generation, computational field simulations (CFS) and in general computer aided engineering analysis and simulation systems. In view of this popularity, the NASA Surface Modeling and Grid Generation Steering Committee established a NASA-IGES (a subset of the standard IGES) format in 1992, and has further proposed the NINO (NASA-IGES NURBS ONLY) standard. Detailed description of IGES entities and the NASA-IGES and NINO standards are presented in Chapter 31 of this handbook. Most of the geometrical configurations of interest to practical CFS problems are designed in the CAD/CAM systems and are available to an analyst in an IGES format. The geometry

preparation which is considered as the most critical and labor intensive part of CFS involves the discrete–sculptured definitions of all boundaries/surfaces, with a desired point distribution and smoothness and orthogonality criteria, associated with the domain of interest. The algorithms associated with geometry preparation and structured grid generation based on the NURBS are presented in this chapter.

The NURBS-based geometry preparation for addressing complex CFS problems encountered in industrial environments involves

1. Transformation of widely utilized explicitly/implicitly/discretely defined IGES geometric entities into a common data structure involving NURBS
2. Surface reparametrization for poorly defined surfaces and repairing of faulty surfaces (most common faults involve gaps, overlaps, and undesired discontinuity between neighboring surface patches) and pertinent geometric entities
3. Geometrical operations allowing projections, intersections (surface–surface intersections), composition, union, and other related transformations essential for surface grid generation with desired topological criteria
4. Grid point distribution with desired stretching and quality criteria on domain boundaries/surfaces

The algorithms for transforming widely utilized geometric entities into NURBS, composition of curves and surfaces and their respective NURBS definitions, grid point distribution, and surface/volume reparametrization are presented in this chapter. However, the algorithms for surface reparametrization, approximation of faulty surfaces, and surface–surface intersections are described in Chapter 29 of this handbook.

The transfinite interpolation (*TFI*) technique is a widely used algebraic method for structured grid generation (see Chapter 3). The NURBS formulation of *TFI* method [Yu, 1995] for surface and volume grids is described in this chapter with appropriate illustrations. The NURBS control-volume-based three-dimensional grid generation algorithms for ruled, extruded, and composite volume and volumes of revolution are also presented in this chapter. The applications of these algorithms facilitate the NURBS common data structure for geometry preparation and grid generation.

30.2 NURBS Formulation

The parametric representation of a NURBS curve/surface/volume involves control polygons with weights, knot vectors (vectors in higher dimension), and orders (representative of degree of polynomial in each direction).

A NURBS curve of order k is defined [Farin, 1993] as

$$C(t) = \frac{\sum_{i=0}^{n} W_i d_i N_i^k(t)}{\sum_{i=0}^{n} W_i N_i^k(t)} \tag{30.1}$$

where the d_i, $i = 0, \ldots, n$ denote the deBoor control polygon and the W_i are the weights associated with each control point. The $N_i^k(t)$ is the normalized B-spline basis function of order k and is defined over a knot vector $T = T_i$, $i = 0, \ldots, n + k$ by the recurrence relations

$$N_i^k(t) = \frac{(t - T_i)N_i^{k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N_{i+1}^{k-1}(t)}{T_{i+k} - T_{i+1}}$$
$$N_i^1(t)\begin{cases} = 1 & \text{if } T_i \le t < T_{i+1} \\ = 0 & \text{otherwise} \end{cases} \tag{30.2}$$

Throughout this chapter, it is assumed that the knot vector has the form $T = \{0, \ldots, 0, T_k, \ldots, T_n, 1, \ldots, 1\}$ with the multiplicity k for the knot value 0 and 1 on both ends of the knot vector. If the knot vectors do not match this format, the knot insertion [Yu, 1995] techniques must be used to achieve the multiplicity of k on the ends of the knot vector, and if the end knot values are not 0 and 1, the knot vector must be normalized by the last knot value to match this format. Because shifting and scaling (normalizing) of the knot values does not alter the underlying geometry, the basis function defined in Eq. 30.2 can be normalized appropriately.

The NURBS surface is defined as a tensor product extension of the curve representation in two directions and is formulated as

$$S(s,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} W_{ij} d_{ij} N_i^{k1}(s) N_j^{k2}(t)}{\sum_{i=0}^{m}\sum_{j=0}^{n} W_{ij} N_i^{k1}(s) N_j^{k2}(t)} \tag{30.3}$$

where d_{ij} denotes the 3D control net, and W_{ij} are the *weights* associated with each control point. The $N_i^{k1}(s)$ and $N_j^{k2}(t)$ denote the normalized B-spline basis functions of *order* $k1$ and $k2$ over the two knot vectors $T_1 = T_i$, $i = 0, \ldots, m + k1$ and $T_2 = T_j$, $j = 0, \ldots, n + k2$ in the I and J directions, respectively. The definition of the B-spline basis functions of the NURBS surface is exactly the same as in Eq. 30.2.

The formula for the 3D NURBS volume is defined analogous to the NURBS surface and is a 3D tensor product form written as

$$V(s,t,u) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W_{ijl} d_{ijl} N_i^{k1}(s) N_j^{k2}(t) N_l^{k3}(u)}{\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W_{ijl} N_i^{k1}(s) N_j^{k2}(t) N_l^{k3}(u)} \tag{30.4}$$

The d_{ijl} form the 3D control volume, and the W_{ijl} are *weights* associated with each control point. The $N_i^{k1}(s)$, $N_j^{k2}(t)$, and $N_l^{k3}(u)$ are the normalized B-spline basis functions of *order* $k1$, $k2$, and $k3$ over the three knot vectors $T_1 = T_i$, $i = 0, \ldots, m + k1$, $T_2 = T_j$, $j = 0, \ldots, n + k2$ and $T_3 = T_l$, l= 0, …, $p + k3$ in the I, J, and L directions (i.e., the s, t, u directions), respectively.

30.3 Transforming and Generating Procedures

Transforming procedures for the widely encountered non-NURBS geometric curves and surfaces, including the *TFI* method of surface generation, to NURBS representation are described in this section. To model a NURBS entity, according to Eqs. 30.1–30.4, one should define the control polygons (or control net/volume), weights, knot vectors, and the respective order for the polynomial in each direction.

It is well known that most commonly used geometric entities in engineering design can be analytically transformed to a NURBS representation [Piegl, 1987, 1989, 1991]. However, there are many practical issues not addressed in the transforming procedures published in the open literature. For example, the IGES representation of the implicit conic arc, and important entity, is not contained in those references. The transforming algorithm for a general circular arc (a circular arc with arbitrary starting and ending points) is also missing. The procedure for transforming a surface of revolution into a NURBS is provided only for a 360° revolution, but many grid generation applications require a specified range of angle, such as 60°. Also, several transforming algorithms are never discussed in any of the literature. These include the transfinite interpolation (also known as the *TFI*) for a NURBS surface/volume and the modeling of

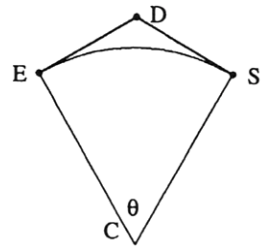

FIGURE 30.1 The basic control triangle for a circular arc.

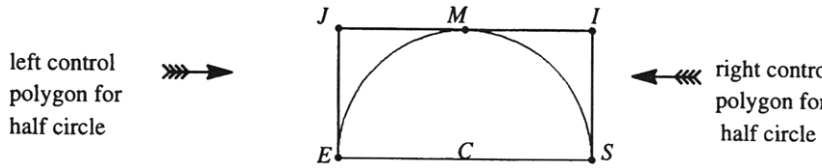

FIGURE 30.2 The NURBS control polygon for a semicircle.

the superellipse as a NURBS curve. The enhancements and generalizations of the transforming procedures were accomplished [Yu, 1995] to meet needs arising from the grid generation process for complex geometries defined in a CAD/CAM system.

These algorithms for transforming various non-NURBS definitions to NURBS representations are described in the following section. (Only the generalized (or enhanced) algorithms are described. The other transforming procedures that can be found in open literature are not repeated here.)

30.3.1 General Circular Arc to NURBS Curve

A circular arc as defined in the IGES standard is represented by a center point, starting point, and ending point within a given constant Z plane. The two endpoints and the center point form an arbitrary sector angle that does not necessarily start from zero. It has been shown that any circular arc with a sector angle less than or equal to 90° can be represented by NURBS [Piegl, 1989, 1991]. The basic control polygon for this NURBS representation is shown in Figure 30.1.

In Figure 30.1 C is the center point, S is the starting point, and E is the ending point. The sector angle SCE (θ) is less than or equal to 90°. The two tangent lines SD and ED intersect at D. The *order* of this control polygon is 3, with the control points S, D, E (hence, the n in Eq. 30.1 will be set to 2) and the *weights* are 1, cos ($\theta/2$), and 1, respectively. The associated knot vector is (0, 0, 0, 1, 1, 1). A circular arc with a sector angle greater than 90° and less than or equal to 180° can be represented by two arcs with one half of the original sector angle. For each of these two sections the previous procedure can be used to evaluate the corresponding control polygon. A 180° circular arc represented by two control polygons is illustrated in Figure 30.2.

From Figure 30.2 the two control polygons can be combined and the common point M can be eliminated. The resulting NURBS information is setting the control polygon to $SIMJE$ (hence, the n is 4), the knot vector to (0, 0, 0, 1/2, 1/2, 1, 1, 1) and the *weights* to (1., cos(θ/n), 1., cos(θ/n), 1.). A similar procedure can be used for circular arcs between 180° and 270° (with n equal to 6) resulting in a final knot vector of (0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1, 1, 1) and *weights* (1., cos(θ/n), 1., cos(θ/n), 1., cos(θ/n), 1., cos(θ/n),1.), and a knot vector of (0, 0, 0, .25, .25, .5, .5, .75, .75, 1, 1, 1) and *weights* (1., cos(θ/n), 1., cos(θ/n), 1., cos(θ/n), 1., cos(θ/n), 1.) for arcs between 270° and 360° for n equal to 8. These four cases are shown in Figure 30.3.

This approach handles all possible circular arcs with no extra computation (such as knot insertion) involved. Furthermore, this algorithm results in good distribution for all cases.

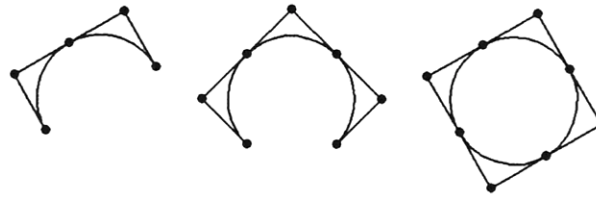

FIGURE 30.3 Arbitrary circular arc with the NURBS control polygons.

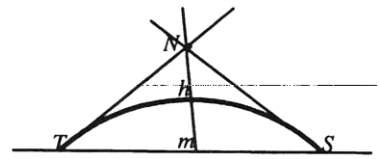

FIGURE 30.4 Basic NURBS control polygon for a conic arc.

30.3.2 Conic Arc to NURBS Curve

The transforming procedure for a conic arc was discussed in [Piegl, 1989, 1991], where the authors described the case of three given control points and changing the weight (conic shape factor) to produce a different family of conic arcs (elliptic, hyperbolic, or parabolic arcs). However, that case is completely different from the one discussed in this section. The conic arc defined in IGES is represented by an implicit form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, with starting point S and ending point T supplied (counterclockwise). The transforming procedure for a basic conic arc is illustrated in Figure 30.4. In this figure, m is the middle point of line TS. Since the two endpoints are known, the two slopes of the tangent lines at the end points can be obtained. The equations describing these two tangent lines can be formed and the intersection point N can be determined. This is accomplished as follows: Differentiate the implicit form of the conic equation to obtain $2Ax + By + Bxy' + 2Cyy' + D + Ey' = 0$. Solving the equation yields

$$y' = (2Ax + By + D)/(-2Cy - Bx - E) \tag{30.5}$$

Substitution of the coordinates of the two endpoints S and T into Eq. 30.5 yields the two desired straight lines. The shoulder point h can then be obtained by solving for the intersection of the line Nm and the given implicit equation. The control triangle is then defined by the polygon SNT (hence, the n is 2 for this case) with *weights* of (1, mh/hN, 1). The *order* can be set to 3 and the knot vector is defined in a manner analogous to the circular arc. As long as this basic control triangle can be found, the procedure used for the circular arc with the sector angle greater than 90° can be applied to the conic arc by simply combining the different control triangles together to form the final control polygon and by setting the proper knot vector. The definition of the sector angle θ for the conic arc is only applied to the elliptic arc; for the parabolic or hyperbolic arcs, three control points are sufficient to form the control polygon. Hence, for a parabolic or hyperbolic arc, the knot vector is always (0., 0., 0., 1.0., 1.0., 1.0) with n equal to 2 Figure 30.5 shows different conic arcs represented by the NURBS using this algorithm. From left to right, (I) elliptic arc with equation $2x^2 + 4xy + 5y^2 - 4x - 22y + 7 = 0$, formed by two NURBS control polygons, (II) parabolic arc with equation $4x^2 - 4xy + y^2 - 2x - 14y + 7 = 0$, (III) hyperbolic arc with equation $2x^2 + 4/3xy - 2y^2 - 16 = 0$.

30.3.3 Cubic Parametric Curve to NURBS Curve

The cubic parametric curve defined in IGES format is a sequence of parametric polynomial segments. More precisely, it is composed of N ($N \geq 1$) pieces of cubic parametric segments. For each parametric curve, it is defined as

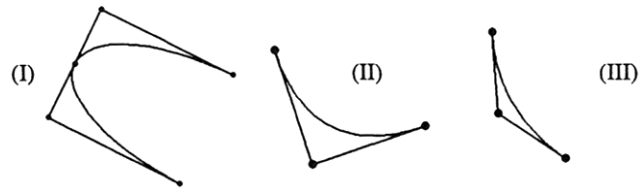

FIGURE 30.5 NURBS control polygon represent different conic arcs.

$$C(u) = a + bt + ct^2 + dt^3 \quad T(i) \le u \le T(i+1) \text{ and } t = u - T(i) \tag{30.6}$$

$T(i)$, i = 1, …, N + 1 are the breakpoints. It has been proven [Farin, 1993] that the cubic Bézier curve is a special case of a B-spline curve with knots vector of (0, 0, 0, 0, 1, 1, 1, 1) (no interior knot value). Also, the B-spline curve is a special case of a NURBS curve with all weights equal to 1. The mathematical transformation from a parametric cubic spline curve in IGES definition to NURBS is accomplished as follows.

The matrix form of each simple cubic parametric curve, according to Eq. 30.6, can be expressed as $C(t) = [1\ t\ t^2\ t^3]\ I_{4\times4}\ [a\ b\ c\ d]^T$ where $I_{4\times4}$ is the identity matrix and $[a\ b\ c\ d]^T$ is the transposed matrix containing the coefficients of the cubic curve. The matrix form of the cubic Bézier curve is expressed as $C(t) = [1\ t\ t^2\ t^3]\ B_{4\times4}\ [b_0\ b_1\ b_2\ b_3]^T$. The $B_{4\times4}$ is the cubic Bézier matrix, and $[b_0\ b_1\ b_2\ b_3]^T$ is the transposed matrix containing the Bézier control polygon. The strategy is to first transform the cubic parametric curve to Bézier form, since a Bézier curve can be treated as a special case of a NURBS curve. Each segment of the parametric spline curve is transformed to a Bézier curve by finding the associated Bézier control polygon. This is done by setting the two matrix equations to be equal, as shown in Eq. 30.7:

$$\text{Bezier} = \begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ -3 & 3 & 0 & 0\\ 3 & -6 & 3 & 0\\ -1 & 3 & -3 & 1\end{bmatrix}\begin{bmatrix}b_0\\ b_1\\ b_2\\ b_3\end{bmatrix} = \text{Cubic curve} = \begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a_i\\ b_i\\ c_i\\ d_i\end{bmatrix} \tag{30.7}$$

and solving the Eq. 30.7 for the Bézier control polygon. Since the cubic parametric spline defined in IGES is composed of N pieces of cubic curves, the range of parametric value t for each piece is not the same as that of the Bézier curve. Hence, a reparameterization of the cubic parametric curve is necessary. For each piece of cubic curve, the coefficients $[a_i\ b_i\ c_i\ d_i]^T$ can be obtained from the given data; therefore, the final equation to solve (for each segment) is

$$\begin{bmatrix}b_0\\ b_1\\ b_2\\ b_3\end{bmatrix} = \frac{1}{3}\begin{bmatrix}3 & 0 & 0 & 0\\ 3 & 1 & 0 & 0\\ 3 & 2 & 1 & 0\\ 3 & 3 & 3 & 3\end{bmatrix}\begin{bmatrix}a_i\\ b_ih\\ c_ih^2\\ d_ih^3\end{bmatrix} \tag{30.8}$$

where $h = T(i + 1) - T(i)$ and $T(i)$ is the break value. After all the Bézier control polygons have been obtained, one can join them together and set the multiplicity of joint knot value equal to 3 to form the final B-spline curve. For example, if two cubic Bézier control polygons are obtained, the final knot vector will be set as (0, 0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1, 1), and the final curve would be C^0 continuous with *order* equal to 4 and all *weights* equal to 1. Figure 30.6 (not applying the knot removal algorithm) demonstrates this approach.

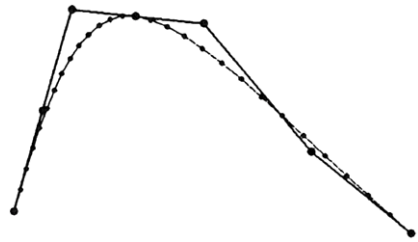

FIGURE 30.6 B-spline control polygon for parametric curve with two segments.

30.3.4 Composite Curve to NURBS Curve

A composite curve in the IGES format is defined as a curve entity consisting of lists of constituent curves. The constituent curve can be any parameterization curve except another composite curve. And this entity is a directed curve, which means the direction of the composite curve is induced by the direction of the constituent curves in the following manner. The start point for the composite curve is the start point of the first curve entity appearing in the defining list, and the terminate point for the composite curve is the terminate point of the last constituent curve appearing in the defining list. Within the defining list itself, the terminate point of each constituent curve entity has the same coordinates as the start point of the succeeding curve entity. It is quite difficult to represent the composite curve precisely without transforming all the constituent curves to the NURBS form. After transforming all curve entities (like straight lines, circular arcs, conic arc, parametric curves and rational B-spline curves), the "NURBS Joining" algorithm for all the constituent NURBS curves is performed to form the NURBS representation for the composite curve. The procedure is illustrated as follows.

Suppose two constituent curves C_1 and C_2 (already transformed to NURBS definition) form a composite curve. Then the first step is to perform the degree of elevation [Piegl, 1991] of the lower-degree curve so that the curves can have the same *order*. The second step is to adjust the knot vector of the second curve C_2 so that the first knot value of the second curve can have the same value as the last knot value of the first curve. Shifting the knot vector will not change the original NURBS curve because the basis function is a "normalized" basis function. The third step is to build up the final knot vector by joining the two knot vectors into one knot vector, and to set that knot value at the joint point to have the multiplicity equal to (*order* –1). For example, if the first knot vector is [0, 0, 0, 1, 1, 1] and the second knot vector is [2, 2, 2, 3, 3, 3], the second knot vector is adjusted by shifting –1 to each value. Thus, the second knot vector becomes [1, 1, 1, 2, 2, 2]. Suppose the *order* of these two curves is 3, then the final knot vector should be [0, 0, 0, 1, 1, 2, 2, 2] (one may notice the interior knot 1 has multiplicity of (*order* –1) = 2). The fourth step is to match the *weights* by timing the ratio of (the last *weight* of the first curve / the first *weight* of the second curve) to all the *weights* of the second curve so that the *weights* at the joint point for the two curves are the same. The last step is to build up the final control polygon and *weights* by throwing the first control point and *weight* of the second curve away and joining the others as one control polygon and one *weights* vector. After these procedures have been applied to all the constituent curves, a composite NURBS curve should be formed. One more procedure that may apply to this final curve is to perform the knot removal to remove the redundant knot vector [Tiller, 1983, 1992]. Figure 30.7 shows this algorithm for transforming the composite curve consisting of, from right to left, one straight line, one circular arc, one straight line, one ellipse arc, and another straight line.

30.3.5 Superellipse to NURBS Curve

A superelliptic arc can be described as Eq. 30.9:

$$\left(\frac{x}{a}\right)^{\eta} + \left(\frac{y}{b}\right)^{\eta} = 1 \tag{30.9}$$

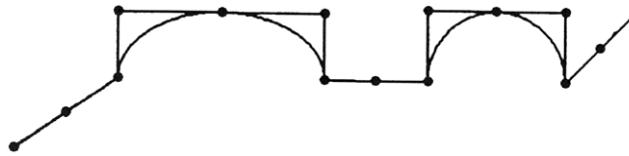

FIGURE 30.7 NURBS control polygon for a composite curve.

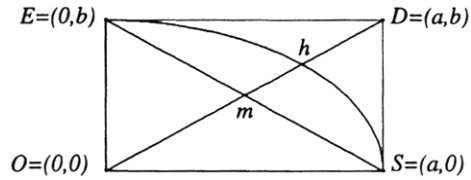

FIGURE 30.8 A superellipse arc with the NURBS control polygon.

where a is the semimajor and b is the semiminor axis of the superellipse. Special cases of Eq. 30.9 include a circle (with $a = b$, and $\eta = 2$), an ellipse (with $a \neq b$, and $y = 2$) and a rectangle (with $a \neq b$, and $\eta = \infty$).

The superellipse is a commonly used geometric description in aerospace design. An example of this is the modeling of a transition duct used for the test of a single-engine nozzle [Reichert et al., 1994]. The transition duct was designed by using a sequence of constant area, superelliptic cross sections according to Eq. 30.9. In this chapter the transforming of this superellipse to a NURBS curve is presented as follows.

This transforming approach is a combination of the circular arc and the conic arc algorithms. Consider a superellipse with semimajor a and semiminor b in the first quadrant, as shown in Figure 30.8. This arc starts at the point $(a, 0)$ and ends at the point $(0, b)$. Two tangent lines intersect at the point (a, b). Similar to the algorithm for the circular arc, these three points can be used as the NURBS control polygon while setting the *order* to be 3 with knot vector (0., 0., 0., 1., 1., 1.). The *weights* at the starting and ending control polygon can be set to 1.0. The only problem remaining is determining the *weight* at the middle point D of the control polygon. This is done similarly to the algorithm of the conic arc. The straight line OD is constructed to intersect with the line SE and the superelliptic arc at the points of m and h. The *weight* at point D is then set as the ratio of (hm/hD).

This approach is self-explanatory. When the exponent of the superellipse η increases, the arc is changing from a circular arc to a rectangular arc, this means that point h is approaching the control point D. Also, the distance of hD is decreasing, and as a result, the *weight* at point D is increased. This situation matches the NURBS theory — a NURBS curve is pulled toward the control point when the *weight* of this control point increases. The mathematical verification can also be done by comparing the h point with the shoulder point evaluation from the NURBS representation. Since the variables (a, b, and η) of the superellipse are all given, the h can be solved from the intersection of the line OD and the arc. On the other hand, after the entire NURBS representation is set up for this superellipse, the shoulder point h can also be evaluated with the parametric value $t = 0.5$. Comparing the locations of these h's, one can find out that the relative deviation is as small as 1.0e – 9. Table 30.1 shows the selected values of the exponent η of the superellipse and the corresponding values of *weights*.

From Table 30.1 one can also notice that in the case of a circular arc or an elliptic arc (when $\eta = 2$), the corresponding *weight* (for the sector angle equal to 90°) is the same as cos(90°/2.0), which has been discussed in circular/elliptic arc section.

30.3.6 Bicubic Parametric Spline Surface to NURBS Surface

The cubic parametric spline surface defined by IGES is composed of M by N cubic patches, as illustrated in Figure 30.9. For each cubic patch, it can be represented as Eq. 30.10:

TABLE 30.1 The Relationship between Exponent η and Weights

η	Weight
2.000000	0.7071067807
2.076143	0.7615055209
2.184741	0.8391550277
2.310944	0.9294727665
2.446475	1.0265482055
2.736506	1.2345144266
2.894152	1.3476587943
3.064489	1.4699782629
3.250206	1.6034070829
3.676614	1.9099667660
3.924127	2.0880154404
4.201364	2.2875017047
4.515468	2.5136151423
4.875638	2.7729511992
5.293192	3.0736854139
5.786112	3.4287875496
6.375087	3.8531827169
7.047038	4.3374610450
7.759080	4.8507150955
8.451551	5.3499221183
9.061041	5.7893464878
9.533431	6.1299460466
9.999865	6.4662654998
10.00000	6.4663630857

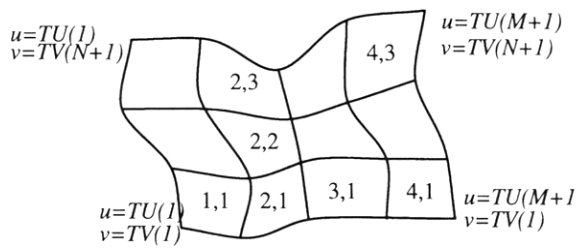

FIGURE 30.9 A 4 × 4 Bicubic parametric patch.

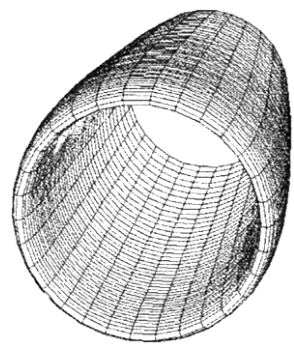

FIGURE 30.10 B-spline surface converted from the bicubic parametric surface.

$$S(u,v) = a + bs + cs^2 + ds^3 + t\left(e + fs + gs^2 + hs^3\right) + t^2\left(k + ls + ms^2 + ns^3\right) + t^3\left(o + ps + qs^2 + rs^3\right) \quad (30.10)$$

Two sets of breakpoint vectors are $TU(i)$, …, $TU(M + 1)$ and $TV(i)$, …, $TV(N + 1)$, where $TU(i) \leqq u \leqq TU(i + 1)$ $i = 1, \ldots, M$, and $s = u - TU(i)$, and $TV(i) \leqq u \leqq TV(i + 1)$ $i = 1, \ldots, N$ and $t = v - TV(i)$.

The strategy for transforming this entity to a B-spline tensor product surface is similar to the one for the cubic parametric spline. The matrix form for the parametric cubic spline surface, according to Eq. 30.10, can be expressed in a matrix form, as shown in Eq. 30.11.

$$S(u,v) = \begin{bmatrix} 1 & s & s^2 & s^3 \end{bmatrix} \begin{bmatrix} a & e & k & o \\ b & f & l & p \\ c & g & m & q \\ d & h & n & r \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \quad (30.11)$$

The matrix form of the Bézier surface with Bézier control points B_{ij} can be expressed as Eq. 30.12.

$$S(u,v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} B_{i,j} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \quad (30.12)$$

The coefficients of this cubic parametric surface are contained in the given data set; therefore, the variables of Eq. 30.11 are all known, and the only unknown for Eq. 30.12 is matrix term of Bézier control points B_{ij}. Hence, the Bézier control points for each bicubic patch are obtained by setting Eq. 30.11 equal to Eq. 30.12 and solving the matrix Eq. 30.13 with reparameterization:

$$\left[B_{ij}\right] = \frac{1}{3}\begin{bmatrix} 3 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 3 & 3 & 3 & 3 \end{bmatrix} \begin{bmatrix} a & eh_2 & kh_2^2 & oh_2^3 \\ bh_1 & fh_1h_2 & lh_1h_2^2 & ph_1h_2^3 \\ ch_1^2 & gh_1^2h_2 & mh_1^2h_2^2 & qh_1^2h_2^3 \\ dh_1^3 & hh_1^3h_2 & nh_1^3h_2^2 & rh_1^3h_2^3 \end{bmatrix} \left(\frac{1}{9}\begin{bmatrix} 9 & 9 & 9 & 9 \\ 0 & 3 & 6 & 9 \\ 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 9 \end{bmatrix} \right) \quad \begin{aligned} h_1 &= TU(i+1) - TU(i) \\ h_2 &= TV(j+1) - TV(j) \end{aligned} \quad (30.13)$$

After all Bézier control patches $B_{i,j}$ are obtained, one can join each subpatch to form the final B-spline surface by setting the multiplicity of the knot value at the junction to 3 in both directions (I and J). Figure 30.10 shows a nacelle of an engine converted from the bicubic parametric surface.

30.3.7 Surface of Revolution to NURBS Surface

The surface of revolution has been discussed in many places [Piegl 1987, 1991] [Piegl and Tiller, 1989] [Tiller, 1992]. However, the more general algorithm is presented here. IGES defines the surface of revolution as the surface formed by rotating a boundary curve (called generatrix) with respect to a straight line (axis of revolution) from a starting angle (not necessarily zero) to an ending angle. This general algorithm for creating the surface grid by NURBS revolution can be stated as follows.

As a first step, the axis of revolution is transformed (translated or rotated or both) so that it is coincident with the Z axis. It is assumed that the generatrix is defined as a NURBS curve with the control polygon d_0 to d_m, *order* equal to k, and *weights* are w_0 to w_m. Next, for each control point d_i (on generatrix $i = 0$, …, m), the surface control net d_{ij} $j = 0, \ldots, n$ at the ith cross section is constructed according to the starting and ending angle by utilizing the circular arc algorithm. Based on the procedure described in Section 30.3.1, n is determined by the sector angle (equal to the difference between ending and starting angles). For example, if the angle is less than 90°, n is equal to 2, if the angle is in the range of 90°–180°, n is equal

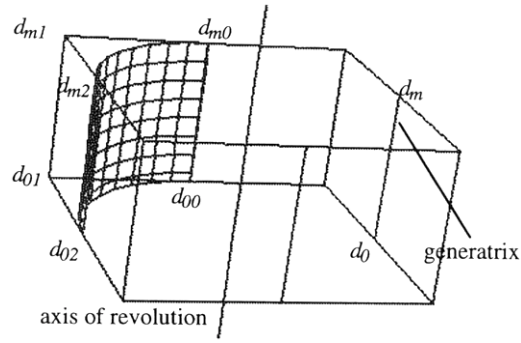

FIGURE 30.11 Illustration of the generalized algorithm for surface of revolution by NURBS.

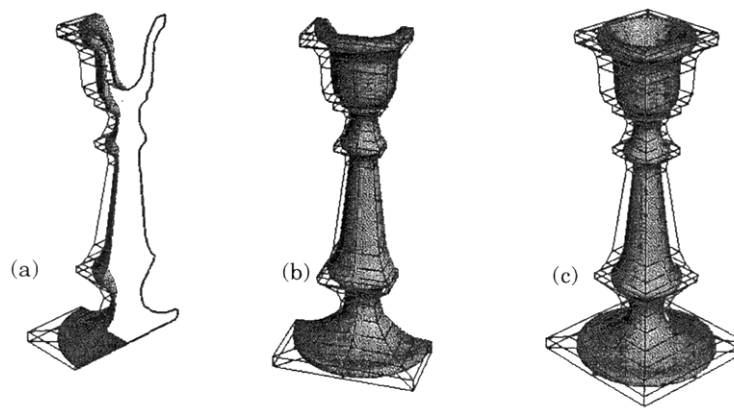

FIGURE 30.12 NURBS surface of revolution. (a) 90° ~ 180°, (b) 90° ~ 270°, (c) full revolution.

to 4, etc. For the section angle θ, the *weights* are set as $W_{ij} = w_i, w_i \cos(\theta/n), w_i, w_i \cos(\theta/n), \ldots,$ (repeat $w_i, w_i \cos(\theta/n)$ with total $n + 1$ terms). The knot vector in direction $I(s)$ is the same as the one of the generatrix while the other one in direction $J(t)$ is determined according to the procedure described in Section 30.3.1. The control net and the *weights* are then transferred back to the original coordinates by reversing the translating/rotating operations. Figure 30.11 shows the construction of the associated control polygon at each cross section for the case of n equal to 2 (section angle equal to 90°).

The final NURBS definition for the constructed surface in Figure 30.11 contains d_{ij} $i = 0, \ldots, m$, $j = 0, \ldots, 2$ as the control net. The *order* and knot vector in the direction I are simply those of the generatrix, while the *order* in the J direction will be set as 3 and the knot vector is set as (0, 0, 0, 1, 1, 1). *Weights* are $W_{ij} = (w_i, w_i\cos(90/2), w_i)$ $i = 0, \ldots, m$ for all j. Figure 30.12 illustrates an example for this algorithm. This figure displays the "candle stand" NURBS control nets as well as the revolved surfaces for different starting and ending angles. The left figure also shows the generatrix.

30.3.8 Transfinite Interpolation for NURBS Surface

In many of the numerical grid generation applications, the H-type grid can be generated easily by the transfinite interpolation algorithm (*TFI*) [Shih, 1994] [Thompson, 1985] (see Chapter 3). As a matter of fact, the *TFI* algorithm is the most frequently used function for the numerical grid generation package.

TFI, also referred to as *Coons–Gordon patch*, is a bivariate interpolation constructed from the superposition of a set of univariate interpolation schemes by the formation of the Boolean sum projector [Thompson, 1985]. In other words, given a set of boundaries (or isoparametric curves), the *TFI* is a

function that constructs the interior surface grid bounded by the given boundaries. The Boolean sum operator for a surface is defined in Eq. 30.14.

$$PS = P_\xi \oplus P_\eta = P_\xi(S) + P_\eta(S) - P_\xi P_\eta(S) \tag{30.14}$$

where $P_\xi(S)$ interpolates the ξ direction of boundaries (the given isoparametric curves) and $P_\eta(S)$ interpolates the η direction of boundaries, while $P_\xi P_\eta(S)$ captures the failures of $P_\xi(S)$ and $P_\eta(S)$. The final surface *PS* bidirectionally interpolates the given curves. There are many functions that can be applied to *TFI*. For example, one can use the linear, quadratic, or even cubic interpolation for function *P* in Eq. 30.14. Taking the linear interpolation for a surface with the resolution *N* by *M* as an example, the Eq. 30.13 can be rewritten as Eq. 30.15:

$$\begin{aligned} R_{ij} = &\left(1-s_{ij}\right)R_{ij} + s_{ij}R_{Nj} + \left(1-t_{ij}\right)R_{i1} + t_{ij}R_{iM} - \\ &\left(\left(1-s_{ij}\right)\left(1-t_{ij}\right)R_{11} + \left(1-s_{ij}\right)t_{ij}R_{1M} + s_{ij}\left(1-t_{ij}\right)R_{N1} + s_{ij}t_{ij}R_{NM}\right) \end{aligned} \tag{30.15}$$

Variables R_{ij} in Eq. 30.15 are the control vertices, which need to be determined. For the NURBS case, the R_{ij} could be d_x, d_y, d_z (control points) and w_{ij} (the *weights*).

This *TFI* function is a fundamental tool for generating grids in many grid applications. However, Eqs. 30.14 and 30.15 *cannot* be applied to NURBS *TFI* directly, because when four NURBS curves are given to generate a NURBS *TFI* surface, the interior control points can be created according to Eq. 30.15 (for the bilinear interpolation) by supplying the control vertices of the boundaries without a problem. The problem comes when determining the interior *weights*. The addition and subtraction operations in Eq. 30.15 may lead to the interior weights being negative or zero values. Any negative *weights* will destroy the convex hull property of a NURBS entity, while any zero *weights* will make the control vertices lose their influence. This obstacle can be overcome by the modified NURBS *TFI* [Lin and Hewitt, 1994]. The formula for this is shown in Eq. 30.16:

$$\left|\begin{matrix} WP(S) \\ W \end{matrix}\right|_{ij} = \left|\begin{matrix} W_\xi W_\eta\left(P_\xi(S) + P_\eta(S) - P_\xi P_\eta(S)\right) \\ W_\xi W_\eta \end{matrix}\right|_{ij} \tag{30.16}$$

Each term of Eq. 30.16 (for the case of linear interpolation of *P*) is defined as follows: the $P_\xi(S)$ represents a NURBS "ruled surface" with *weights* of W_ξ formed in ξ direction (hence, the *order* is 2, knot vector is [0, 0, 1, 1] in ξ direction). $P_\eta(S)$ represents another NURBS "ruled surface" with *weights* of W_η formed in η direction (*order* is 2, knot vector is [0, 0, 1, 1] in η direction) and the $P_\xi P_\eta(S)$ is a NURBS surface constructed by using the four corner points as the control net with *orders* 2 by 2 in ξ and η directions. This is demonstrated in Figure 30.13.

After creating the intermediate surfaces of $P_\xi(S)$, $P_\eta(S)$, and $P_\xi P_\eta(S)$, one has to perform the "knot insertion" [Piegl, 1991] and "degree elevation" [Piegl, 1991] [Tiller, 1992] algorithms to these three surfaces to ensure all of them have the same *orders* and same knot vectors in both the ξ and η directions. If the NURBS surfaces have the same *orders* and same knot vectors, then the dimension of the control net is the same also. Therefore, the control net of the final NURB *TFI* surface can be obtained by adding the control nets of $P_\xi(S)$, $P_\eta(S)$ and subtracting those of $P_\xi P_\eta(S)$, while the *weights* are determined by multiplying W_ξ and W_η.

Comparing the NURBS *TFI* with the traditional *TFI* shows that the NURBS *TFI* needs more computation because the *weights* need to be handled properly. In addition, the knot insertion and degree elevation algorithms need extra computation. However, this function is fundamental and useful when there is a need to create *H*-type grids. Also, when generating the volume grids for a nozzle, this function is particularly useful to create the inlet and outlet surfaces. Figure 30.14 demonstrates this example.

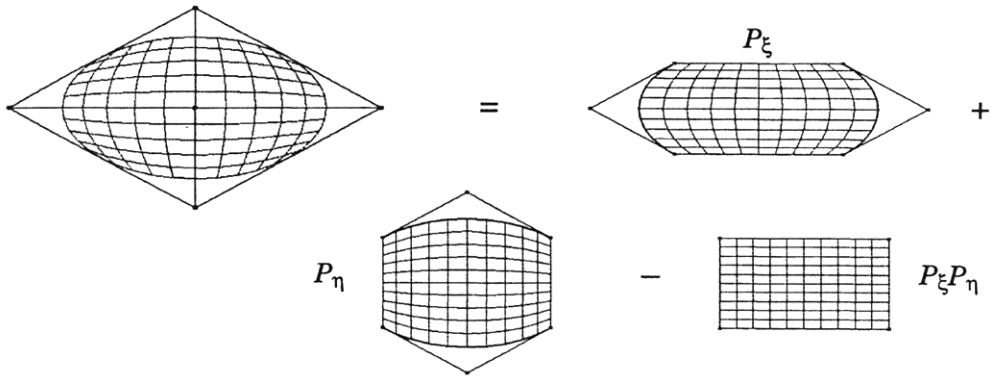

FIGURE 30.13 An illustration of NURBS *TFI* surface.

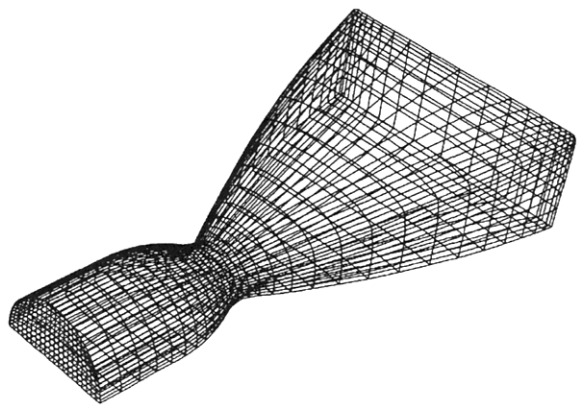

FIGURE 30.14 NURBS *TFI* creates the inlet/outlet surface for a circular–rectangle nozzle.

30.3.9 Cascading Technique for NURBS Surface

As discussed in the previous transforming procedures, the surface of revolution algorithm can be used to model symmetric surfaces. In CFD applications, some of the geometries for analysis are symmetric objects, such as the simulation of the flow passing around a missile. Generally speaking, the surface of revolution algorithm can be used to model a "simple" symmetric surface, but for many CFD applications, the real geometric objects interact with other objects and cannot be modeled by rotating a boundary curve to form a surface of revolution, as shown in Figure 30.15, where a surface blade with the boundary is intersected with a fin. Even though this surface is symmetric, the surface of revolution (SOR) algorithm fails to model it.

This situation also occurs with the "cascade" surface. A cascade surface is usually referred to as the "blade-to-blade" surface in turbomachinery [Shih, 1994]. Even though most of the cascade surfaces are axisymmetric, they cannot be modeled by the NURBS surface revolution algorithm. Also, in the grid generation area, creating the surface grid for the cascade is a challenge. The difficulty of modeling the surface grids for a 3D cascade surface is that when the blade leading edge (or trailing edge) circle radius is too big, such as in a turbine, or if the blade setting angle (the blade angle) is very low, it is hard to generate a well behaved *H*-type grid. Particularly, there is often a grid crossing near the leading edge (or trailing edge) for such a geometry. Traditionally, this kind of surface grid is generated by transforming the 3D surface of the (x, y, z) coordinates to 2D parametric (m', σ) space, griding in the parametric space and then transforming the coordinates back to 3D physical space according to the relation of the (m', σ). Detailed information can be found in [Shih, 1994]. In this section NURBS modeling approach is presented for modeling this type of geometry.

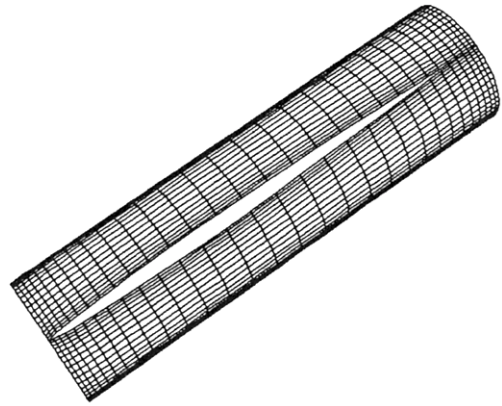

FIGURE 30.15 A symmetric surface blade.

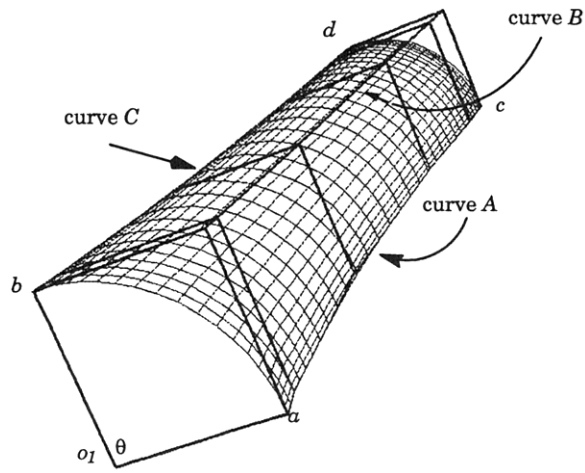

FIGURE 30.16 Illustration of modeling cascade surface by the NURBS control net.

The NURBS algorithm for modeling the cascade surface is described as follows: given a boundary curve of a cascade surface, transform it to a B-spline curve (as curve *A* shown in Figure 30.16) by the interpolation technique. A plane that bisects the surface sector angle (the θ, angle of ao_1b shown in Figure 30.16) is created and then the "mirror" function [Yu, 1995] is used to reflect curve *A* with this plane to create curve *C*. Curve *C* will have the same *order*, knot vector, and number of control points as curve *A*. The next step is to create a straight line on the plane that contains the points o_1, *a*, and *c*. This is done by projecting the control polygons of curve *A* to the plane and setting the *order* and knot vector of the line to be the same as those of curve *A*. After this line is created, the surface of revolution algorithm is performed to rotate this line with respect to the center of o_1o_2 for a total sector angle of θ. A NURBS tabulated cylinder [Yu and Soni, 1995] with sector angle θ will be generated after this step. However, since this surface is not the desired cascade surface, the first and last iso-control polygons (in the axis direction) of this surface must be replaced with the existing B-spline curves *A* and *C*. Because the tabulated cylinder is created by rotating a line that has the same *order* and knot vector as those of curve *A*, it is secure to replace the two control polygons of the surface with *A* and *C* without altering the entire shape of the surface. The control net, with curves *A* and *C* replacing the first and last iso-control polygons, is the final desired NURBS control net. A missile configuration, composed of the surface of revolution and cascade surface, is shown in Figure 30.17 to demonstrate this algorithm.

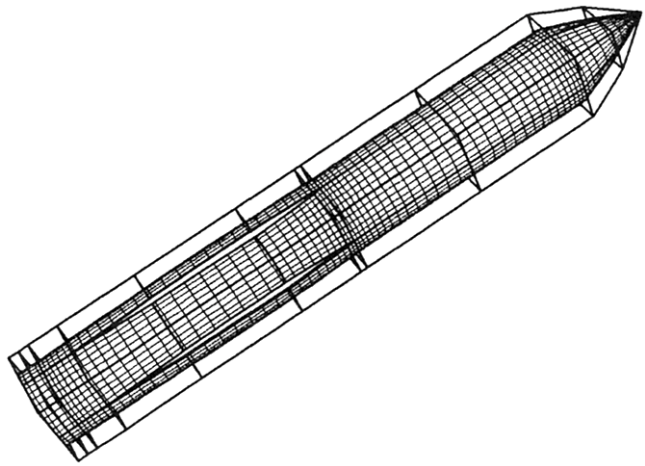

FIGURE 30.17 A missile surface grid modeled by the NURBS control net.

30.4 Grid Redistribution

The NURBS entity (curve, surface, or volume) is presented as a parametric format, and a grid point on a NURBS entity is generated by evaluating the parametric value t (or s, t for surface, s, t, u for the volume). The placement of grid points on the physical geometry of interest with the desired stretching/concentration criteria is of key importance for CFS analysis. This in turn requires the reparametrization of parametric values t (s, t for surface and s, t, u for the volume) such that when NURBS formula is evaluated, the desired distribution criteria are met on the physical geometry. For example, evenly distributed parametric values t may not result in a sequence of evenly distributed grid points of $C(t)$ on the physical NURBS curve. The location of the control polygon, the value of *weights*, and even the knot vector are all possible factors in evaluation of the NURBS entity. Changing any of these factors could result in an unexpected (or undesired) packing of the grid points (lines or surface) in the physical geometry. This situation is referred to as "bad parametrization" and is remedied by performing reparametrization. The problem in this case is calculating the proper parametric values to obtain the desired distribution on the physical NURBS entity without altering the NURBS definition (control polygon, *weights*, and knot vector). The reparametrization algorithm is presented for the three-dimensional NURBS volume entity. The respective algorithms for the one-dimensional (curve) and the two-dimensional (surface) NURBS entities can be easily deduced from the three-dimensional scheme.

30.4.1 Reparametrization Algorithm

Before discussing the algorithms, it is necessary to define several notations. For the NURBS tensor product volume with resolution ni, nj, and nl, the 3D arrays are defined as follows:

1. $(vs_1(i,j,l), vt_1(i,j,l), vu_1(i,j,l))$, $i = 1, \ldots, ni$, $j = 1, \ldots, nj$, and $l = 1, \ldots, nl$ are the parametric distribution volume associated with the desired distribution of the volume in physical space.
2. $(vs_2(i,j,l), vt_2(i,j,l), vu_2(i,j,l))$, $i = 1, \ldots, ni$, $j = 1, \ldots, nj$, and $l = 1, \ldots, nl$ are the normalized chord length of the volume with desired distribution in direction I, J, and L, respectively.
3. $(vs_3(i,j,l), vt_3(i,j,l), vu_3(i,j,l))$, $i = 1, \ldots, ni$, $j = 1, \ldots, nj$, and $l = 1, \ldots, nl$ are the normalized chord length of the volume evaluated at parametric values $(vs_1(i,j,l), vt_1(i,j,l), vu_1(i,j,l))$, $i = 1, \ldots, ni$, $j = 1, \ldots, nj$, and $l = 1, \ldots, nl$.

These variables are explained as follows: If the designer would like to have the final volume grid, say, evenly distributed, then $(vs_2(i,j,l), vt_2(i,j,l))$ would be a 3D array that contains the even distribution, and $(vs_1(i,j,l), vt_1(i,j,l))$ would be the parametric values that are to be determined such that $(vs_3(i,j,l), vt_3(i,j,l))$, the normalized

chord length of final volume, would be the same as ($vs_2(i,j,l)$, $vt_2(i,j,l)$) or within certain tolerance. The algorithm for finding the desired parametric values is illustrated by the pseudo-code Algorithm I:

Algorithm I

For *each* parametric value, search the index of *I*, *J*, and *L* such that

$$
\begin{aligned}
&\left(vs_2(i,j,l), vt_2(i,j,l), vu_2(i,j,l)\right) \quad \text{is located within the cells of} \\
&\left(vs_3(I,J,L), vt_3(I,J,L), vu_3(I,J,L)\right) \\
&\left(vs_3(I+1,J,L), vt_3(I+1,J,L), vu_3(I+1,J,L)\right) \\
&\left(vs_3(I,J+1,L), vt_3(I,J+1,L), vu_3(I,J+1,L)\right) \\
&\left(vs_3(I,J,L+1), vt_3(I,J,L+1), vu_3(I,J,L+1)\right) \\
&\left(vs_3(I+1,J+1,L), vt_3(I+1,J+1,L), vu_3(I+1,J+1,L)\right) \\
&\left(vs_3(I+1,J,L+1), vt_3(I+1,J,L+1), vu_3(I+1,J,L+1)\right) \\
&\left(vs_3(I,J+1,L+1), vt_3(I,J+1,L+1), vu_3(I,J+1,L+1)\right) \\
&\left(vs_3(I+1,J+1,L+1), vt_3(I+1,J+1,L+1), vu_3(I+1,J+1,L+1)\right)
\end{aligned}
$$

After the (*I, J, L*) is found, for *each* parametric value, solve the α, β, and γ for Eq. 30.17:

$$
\begin{aligned}
&\left(vs_2(i,j,l), vt_2(i,j,l), vu_2(i,j,l)\right) = (1-\alpha)(1-\beta)(1-\gamma)\left(vs_3(I,J,L), vt_3(I,J,L), vu_3(I,J,L)\right) \\
&+\alpha(1-\beta)(1-\gamma)\left(vs_3(I+1,J,L), vt_3(I+1,J,L), vu_3(I+1,J,L)\right) \\
&+(1-\alpha)\beta(1-\gamma)\left(vs_3(I,J+1,L), vt_3(I,J+1,L), vu_3(I,J+1,L)\right) \\
&+(1-\alpha)(1-\beta)\gamma\left(vs_3(I,J,L+1), vt_3(I,J,L+1), vu_3(I,J,L+1)\right) \\
&+\alpha\beta(1-\gamma)\left(vs_3(I+1,J+1,L), vt_3(I+1,J+1,L), vu_3(I+1,J+1,L)\right) \\
&+\alpha(1-\beta)\gamma\left(vs_3(I+1,J,L+1), vt_3(I+1,J,L+1), vu_3(I+1,J,L+1)\right) \\
&+(1-\alpha)\beta\gamma\left(vs_3(I,J+1,L+1), vt_3(I,J+1,L+1), vu_3(I,J+1,L+1)\right) \\
&+\alpha\beta\gamma\left(vs_3(I+1,J+1,L+1), vt_3(I+1,J+1,L+1), vu_3(I+1,J+1,L+1)\right)
\end{aligned}
\tag{30.17}
$$

After α, β, and γ are obtained, the new parametric values are determined as shown in Eq. 30.18:

$$
\begin{aligned}
&\left(vs(i,j,l), vt(i,j,l), vu(i,j,l)\right) = (1-\alpha)(1-\beta)(1-\gamma)\left(vs_1(I,J,L), vt_1(I,J,L), vu_1(I,J,L)\right) \\
&+\alpha(1-\beta)(1-\gamma)\left(vs_1(I+1,J,L), vt_1(I+1,J,L), vu_1(I+1,J,L)\right) \\
&+(1-\alpha)\beta(1-\gamma)\left(vs_1(I,J+1,L), vt_1(I,J+1,L), vu_1(I,J+1,L)\right) \\
&+(1-\alpha)(1-\beta)\gamma\left(vs_1(I,J,L+1), vt_1(I,J,L+1), vu_1(I,J,L+1)\right) \\
&+\alpha\beta(1-\gamma)\left(vs_1(I+1,J+1,L), vt_1(I+1,J+1,L), vu_1(I+1,J+1,L)\right) \\
&+\alpha(1-\beta)\gamma\left(vs_1(I+1,J,L+1), vt_1(I+1,J,L+1), vu_1(I+1,J,L+1)\right) \\
&+(1-\alpha)\beta\gamma\left(vs_1(I,J+1,L+1), vt_1(I,J+1,L+1), vu_1(I,J+1,L+1)\right) \\
&+\alpha\beta\gamma\left(vs_1(I+1,J+1,L+1), vt_1(I+1,J+1,L+1), vu_1(I+1,J+1,L+1)\right)
\end{aligned}
\tag{30.18}
$$

Finally, replace the $vs_1(i,j,l)$, $vt_1(i,j,l)$, $vu_1(i,j,l)$ with $vs(i,j,l)$, $vt(i,j,l)$, $vu(i,j,l)$.

30.4.2 Singularity Control

It can easily be shown that the reparametrization algorithm presented here will fail if the underlying NURBS curve/surface/volume contains any singularities. Using a NURBS curve as an example, if all the points on this curve collapse to one point, then this NURBS curve is a singular curve. The same definition can be applied to NURBS surface and volume. If any one of the iso-parametric lines on a NURBS surface collapses to one point, then this line is defined as a singular line. For a volume, if any one of the iso-planes on a NURBS volume collapses to a line, then this plane is a singular plane. When these singularity problems occur, the total arc length from calculating it according to the reparameterization algorithms will be zero. Since the normalized arc lengths are obtained by dividing the total arc length by the individual ones, this will lead to the operation of 0/0, which is mathematically undefined. These singularity problems are often encountered in CFS applications requiring structured grids. For example, the surface grid that represents the canopy of an aircraft has a singular line at the nose position: a surface grid that represents the missile has a singular line in the nose position; and the volume grid of a cylinder (or any cylindrical pipe) has a singular plane at the axial direction. In each of these cases, the singularity problems occurred because the control points collapse to one point (for the surface case) or one line (for the volume case). While evaluating the NURBS entities with certain parametric values by utilizing these collapsed control points, the singularity problem arises. Hence, it is necessary to enhance the algorithm to handle this problem.

The strategy for the enhancement of the algorithm is related to the machine accuracy (also called machine precision). The machine accuracy, commonly represented as symbol ε, is defined as the smallest positive real number such that $1 + \varepsilon > 1$. On the *Silicon Graphics Personal Iris,* this number is equal to 10.0^{-16} (double precision). In many numerical simulations, this number is needed to represent the finite precision arithmetic of the computer architecture. For example, the convergence criteria of an iteration scheme is dependent upon the machine accuracy. A variable expected to be *zero* in numerical representation may not be reached due to the finite precision of the computer memory representation. Therefore, in many numerical applications, the exact *zero* is replaced by a value related to ε; for example, if a variable is less than $\sqrt{\varepsilon}$; then that variable is assumed to be equal to *zero*. This concept is also utilized to avoid the singularity problems. Using the NURBS surface with a singular line (say, occurring at grid line index $i = 0$) as an example, the grid line evaluated with the parametric values of $(ss_1(0,j), st_1(0,j), j = 0, \ldots, nj$ will shrink to one singular point due to the control vertices d_{0j} collapsing to one point. However, if one perturbs these parametric values by a small value, perhaps $\sqrt{\varepsilon}$ and then reevaluates the surface, the returned grid line will not be the same as the singular one since these parametric values are no longer exactly *zero*. Instead, it will return a grid line with a small but recognizable total arc length. Even though the total arc length is small, the normalization process will make the values of $(ss_3(0,j), st_3(0,j)\ J = 0, \ldots, nj$ to 0.0 ~ 1.0 and will avoid the uncertain situation of 0/0.

The associated algorithm applied to a 3D NURBS volume is presented as Algorithm II, and Figure 30.18 shows a 3D NURBS cylindric pipe evaluated with even parametric values. Notice that in its L direction, the surface degenerates to a singular line. The result of reparameterization for this volume is shown in Figure 30.19.

Algorithm II

$$
\begin{aligned}
&for(k = 0; k < nk; k++)\\
&for(j = 0; j < nj; j++) \quad vs_1(0, j, k) += \varepsilon; vs_1(ni-1, j, k) -= \varepsilon;\\
&for(k = 0; k < nk; k++)\\
&for(i = 0; i < ni; i++) \quad vt_1(i, 0, k) += \varepsilon; ct_1(i, nj-1, k) -= \varepsilon;\\
&for(j = 0; j < nj; j++)\\
&for(i = 0; i < ni; i++) \quad vu_1(i, j, 0) += \varepsilon; vu_1(i, j, nj-1) -= \varepsilon;
\end{aligned}
$$

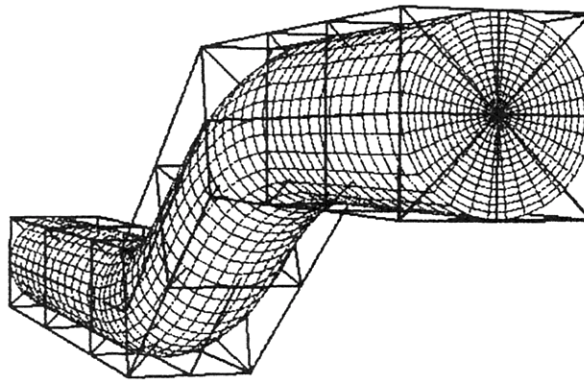

FIGURE 30.18 A NURBS volume grid with a singular plane in flow direction.

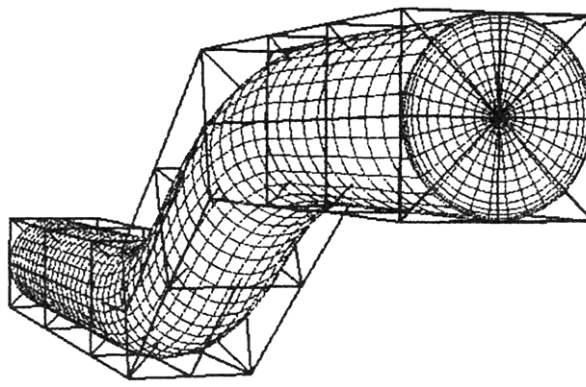

FIGURE 30.19 The reparameterization algorithm for a NURBS volume grid with singularity.

30.5 Volume Grid Generation by NURBS Control Volume

Volume grid generation algorithms have been utilized in many CFD analysis procedures. A widely used technique to algebraically generate a three-dimensional volume grid is to utilize the transfinite interpolation algorithm based on the bounding surface grids. However, the volume generation techniques are seldom applied to CAD/CAM applications. Even though NURBS representation has been widely used in many industry applications, the geometry modeled by the NURBS volume approaches are seldom discussed in the computer-aided geometry design (CAGD) literature. In this chapter, using the NURBS volume to model the geometry for the volume grid is presented. Instead of storing the surface/volume grid points, one can store the associated control polygon (or control net for the surface/volume) with the associated weights to reduce the memory requirement. This is especially useful for volume grid generation. Even though computer memory availability has been dramatically improved, a complicated geometry usually consumes a great deal of computer memory for the volume grid. Storing the NURBS control net to reduce the size of an entire volume grid is demonstrated in the examples of this section.

The ultimate objective is to explore various NURBS control volume options applicable to three-dimensional grid generation. In this section, the development of NURBS ruled volume, NURBS extruded volume, NURBS volume of revolution, NURBS composite volume, and transfinite interpolation (*TFI*) NURBS volume are discussed.

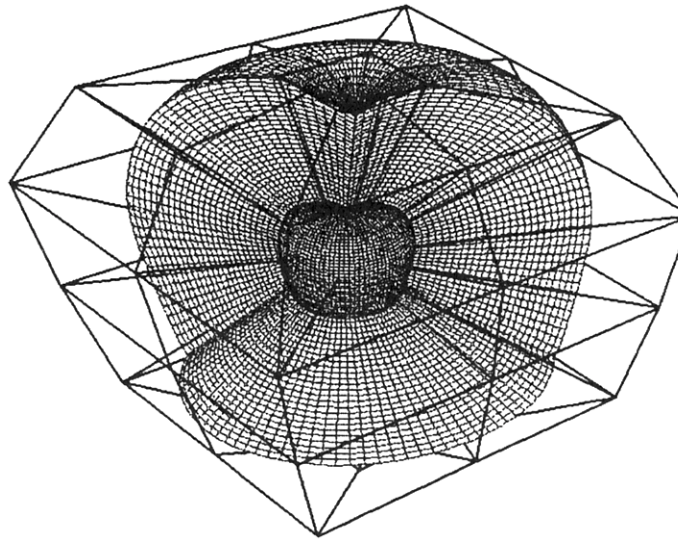

FIGURE 30.20 A volume grid created by NURBS control volume (ruled volume option).

30.5.1 Ruled Volume

The easiest 3D NURBS volume to generate is the ruled NURBS volume. The algorithm is described as follows: Given two NURBS surfaces, the first step to form a ruled volume is to make the knot vectors of the surfaces be in the same range of [(0~l]. Next, considering the *I* direction of both surfaces, the degree raising technique is used to raise the low degree (*order* – 1) of the surface. This procedure will yield a new knot vector and new control net. If the new knot vector differs from the other knot vector, then the knot insertion algorithm is performed to merge them into one final knot vector. Then these steps are applied to the knot vectors in the *J* direction of both surfaces. After this step, the two NURBS surfaces will have the same *orders* and the same knot vectors in both *I* and *J* directions. This means that the resolutions of the control nets of both surfaces will be the same. Finally, connect the corresponding control points together to form the 3D NURBS volume. The orders and knot vectors of the final volume in the *I* and *J* directions will be the same as those of the surfaces after degree elevation and knot insertion, and the *order* in *L* direction will be set as 2 with the knot vector set as (0,0,1,1). Figure 30.20 shows a 3D "apple-like" NURBS volume and its volume grid, while Figure 30.2l shows a missile configuration with the control volume formed by this algorithm.

30.5.2 An Extruded Volume

The generation of a NURBS extruded volume is an extension of the extruded surface definition. The extruded surface is defined as a surface formed by moving a line segment parallel to itself along a curve. In other words, given a NURBS curve, one can generate another curve by extruding the given curve with a distance α along a vector *V*. Similarly, the NURBS extruded volume is defined as follows: Given a NURBS surface with control net d_{ij} and the associate *weights*, knot vectors and *orders*, the new surface $\bar{d}_{ij}$ can be generated by "extruding" the given surface with a distance α along a vector *V*. Mathematically, this new NURBS extruded surface can be described as $\bar{d}_{ij} = d_{ij} + \alpha V$ with the same *orders*, same *weights* and same knot vectors as those of the given surface. After this step the algorithm of "ruled volume" can be applied to these surfaces to form a final NURBS volume. Figure 30.22 shows a 3D NURBS extruded volume.

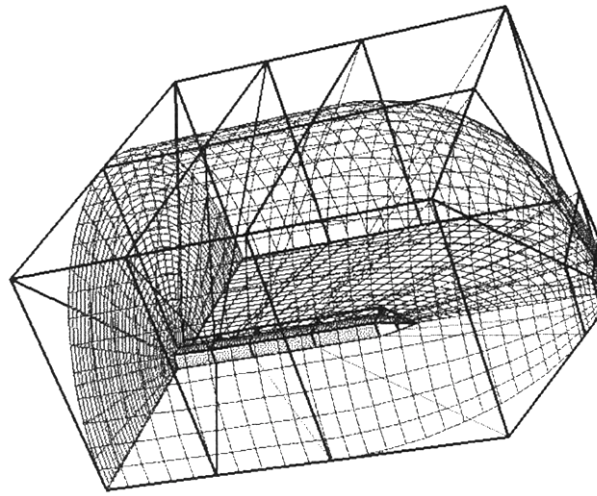

FIGURE 30.21 A missile volume grid modeled by NURBS ruled volume.

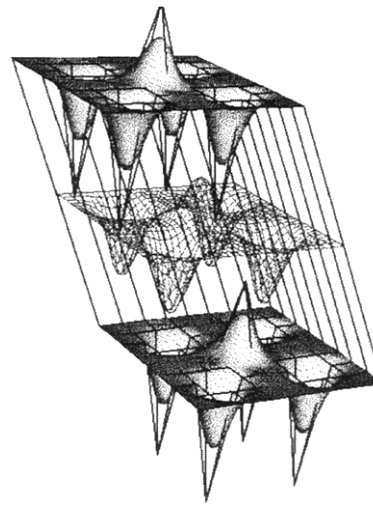

FIGURE 30.22 A volume grid created by the NURBS extruded volume option.

30.5.3 Volume of Revolution

Another commonly used approach for generating volume grids is the "revolution" method. A revolution resulting in a surface is known as a "surface of revolution," while a revolution resulting in a volume is then known as a "volume of revolution." The fact that this modeling technique can be used only for symmetric geometries is not limiting, since many objects in real-world applications, such as turbomachinery configurations, are symmetric. The extension of a volume revolution modeled by NURBS is presented as follows: the definition of surface of revolution is a surface formed by rotating a given curve with respect to an arbitrary straight line from a starting angle to an ending angle. Likewise, the volume of revolution is defined as a volume formed by rotating a given NURBS surface with respect to an arbitrary axis of revolution from any starting angle to an ending angle.

The general algorithm is outlined as follows: the first step is translating/rotating the axis of revolution by a proper transformation matrix so that it is coincident with the Z axis. This transformation matrix

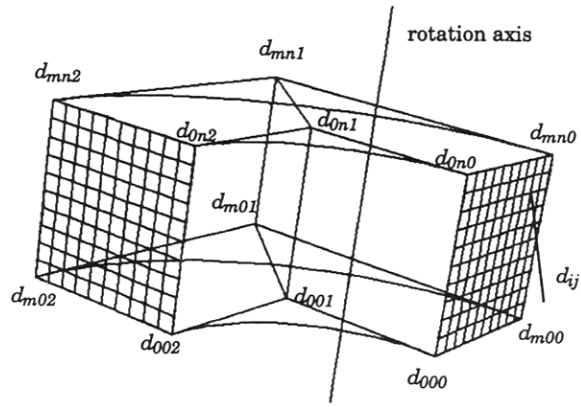

FIGURE 30.23 Illustration of constructing the NURBS volume of revolution.

is also applied to the given NURBS surface so that the entire surface can be kept in the same position as the axis of rotation. It is assumed that the surface is defined (or transformed) as NURBS with the control net d_{ij} order $k1$ and $k2$, *weights* W_{ij} and two knot vectors. The second step is to construct, for each control net d_{ij} (on the generatrix $i = 0, \ldots, m$, $j = 0, \ldots, n$), the control volume $d_{ijl}\, l = 0 \ldots p$ at each jth cross section through the starting and ending angle by utilizing the circular arc algorithm. In other words, this approach constructs the NURBS control net at each J-constant plane by revolving the control polygon d_{ij} with respect to L direction and then "stacks" them together to form a final NURBS volume. Figure 30.23 demonstrates this approach. The general procedure of generating the NURBS circular arc is described in a previous section. The p for the last dimension of control volume is determined by the sector angle θ (equal to the difference between ending and starting angle). For example, if the angle is less than 90°, p is equal to 2. If the angle is in the range of 90° ~ 180°, p is equal to 4; if in the range of 180° ~ 270°, p is 6; if it is greater than 270°, p should be 8. For the sector angle θ, the *weights* are set as (in each J constant plane, $J = 0, .. n$) $W_{ijp} = w_{ij}, w_{ij} \cos(\theta/p), w_{ij}, w_{ij}, \cos(\theta/p), \ldots\ i = 0, \ldots, m$ (repeat $w_{ij}, w_{ij}\cos(\theta/p)$ with total $p + 1$ terms). The knot vectors in directions of $I(s)$ and $J(t)$ are the same as the ones of the given surface, while the knot vector in direction $L(u)$ is determined according to the circular arc procedure. For example, when p is 2, the associated knot vector is set as (0, 0, 0, 1, 1, 1); for the case of p equal to 4, the knot vector is set as (0, 0, 0, 1/2, 1/2, 1, 1, 1); for the case of p equal to 6, the knot vector is set as (0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1,1,1); and for the case of p equal to 8, the knot vector is (0,0,0, .25, .25, .5, .5, .75, .75,1,1,1). Also, the *orders* in I and J are set to be $k1$ and $k2$ (as the orders of the original surface), while 3 is set as the *order* of L direction.

Because the NURBS has the translate and rotate invariant properties, the inverse transformation matrix can be applied to the control volume (without altering the *weights* and knot vectors) returning the volume to the original coordinates. Figure 30.21 shows a 3D volume grid and its control volume, according to this algorithm. This example was developed by revolving the *TFI* surface from 0° to 180°. Because the NURBS surface *TFI* technique needs four boundary curves to define a surface, this results in an "*H*" type surface grid. Revolving this "*H*" type *TFI* surface creates "*H*" type NURBS control volume and yields the "*H*" volume grid. This topology can be changed by revolving another "0" type NURBS surface to form an "0" type volume grid, as shown in Figure 30.24. Notice that the sizes of this control volume are only $3 \times 3 \times 5$ (for "*H*" type grids) and $9 \times 9 \times 5$ (for "0" type grids), yet the resolution of the entire volume grid can be any number (for this case, $31 \times 31 \times 61$).

30.5.4 Composite Volume

A composite NURBS volume is defined as a volume consisting of lists of constituent volumes. The composing procedure is stated as follows: Suppose two constituent NURBS volumes V_1 and V_2 form a composite volume. Assume that V_1 has control volume $d_1[0{:}m_1, 0{:}n_1, 0{:}l_1]$, weight W_1 $[0{:}m_1, 0{:}n_1,$

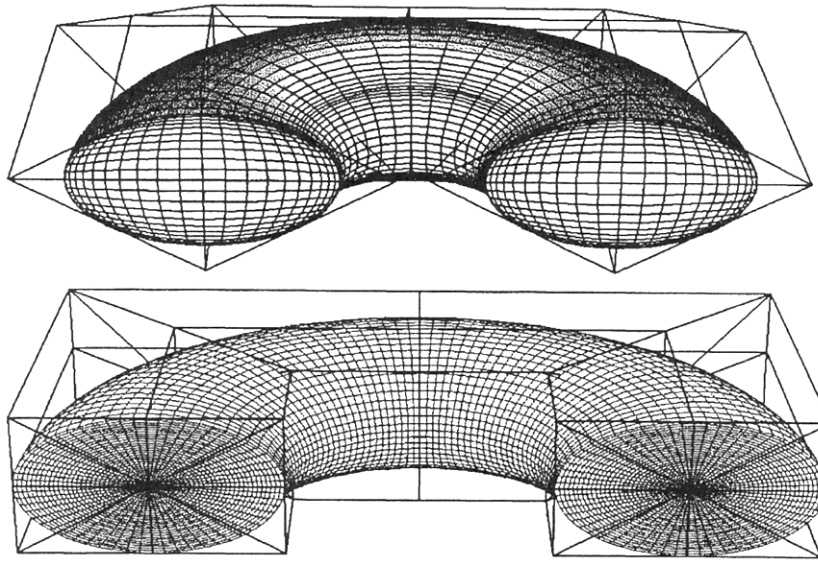

FIGURE 30.24 *H* and *O* type volume grids created by NURBS volume of revolution.

0:l_1], three knot vectors $knot\_i)_1$, $knot\_j)_1$, $knot\_l)_1$ and orders $k\_i)_1$, $k\_j)_1$, $k\_l)_1$ while V_2 has control volume d_2[0:m_2, 0:n_2, 0:l_2], weight W_2[0:m_2, 0:n_2, 0:l_2], three knot vectors knot $knot\_i)_2$, $knot\_j)_2$, $knot\_l)_2$ and orders $k\_i)_2$, $k\_j)_2$, $k\_l)_2$. There are many possible combinations of the two volumes joined together. For example, one can join the volumes in the *I* direction with the interface of the *J*, *L* surface, or join in the *L* direction with the interface of the *I*, *J* surface, etc. Even though there are many cases, the procedure is similar. When joining in the *I* direction, for example, the first step is to perform the degree elevation to V_1 and V_2 so that these two volumes can have compatible degrees in the *I*, *J*, and *L* directions. If the two knot vectors in the *J* direction for V_1 and V_2 are not the same, they are merged together by setting the final knot vector as {$knot\_j)_1$ / $knot\_j)_2$}; then the knot insertion is applied to V_1 and V_2 in the *J* dimension. The same procedure should be applied to the *L* direction if $knot\_l)_1$ and $knot\_l)_2$ are not the same. After this step, V_1 and V_2 will have the same degree in three directions, and the number of control points and knot vectors in the *J* and *L* directions will be the same. The second step is to adjust the knot vector $knot\_i)_2$ so that its first knot value can be the same as the last knot value $knot\_i)_1$. Shifting the knot vector will not change the original NURBS because the basis function is a "normalized" basis function. The third step is to build up the final knot vector by joining the two knot vectors into one knot vector and setting that knot value at the joint point to have the multiplicity equal to (*order* –1). For example, if the knot vector $knot\_i)_1$ is [0., 0., 0., 1., 1., 1.] and the knot vector $knot\_i)_2$ is [2., 2., 2., 3., 3., 3.], the second knot vector is adjusted by shifting –1 to each value. Thus, the $knot\_i)_2$ becomes [1,1,1,2,2,2]. Suppose the final *order* of these two volumes in the *I* direction is 3; then, the final knot vector should be [0., 0., 0., 1., 1., 2., 2., 2.] (notice the interior knot 1 has multiplicity of (*order* –1) = 2). The fourth step is to match the *weights* at the interface surface by multiplying the ratio of $W_1[m_1, j, l]$ / $W_2[0, j, l]$ to $W_2[i, j, l]$ for $i = 0, \ldots, m_2$, $j = 0, \ldots, n_1$, and $l = 0, \ldots, l1$]. The last step is to construct the final control volume and *weights* by removing the d_2[0:0, 0:n_2, 0:l_2] and W_2[0:0, 0:n_2, 0:l_2] and joining the others as one control volume and *weights*. Figure 30.25 demonstrates this algorithm.

Generally speaking, it is difficult to model a complicated geometry by a single NURBS control volume. However, one can construct the individual control volume and then utilize this composite algorithm to merge for a final volume. Figure 30.26 demonstrates the flexibility and the advantage of this approach. The NURBS control volume is used to model the internal pipes. The griding of the turning portions of the pipe can be constructed by "volume of revolution" without any difficulties. Assembling all the sub-NURBS volumes makes the final single block NURBS control volume.

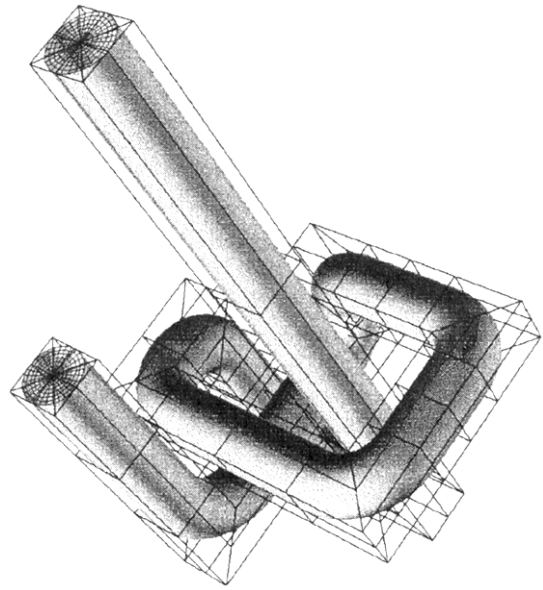

FIGURE 30.25 A volume grid for the turning pipe created by NURBS composite volume.

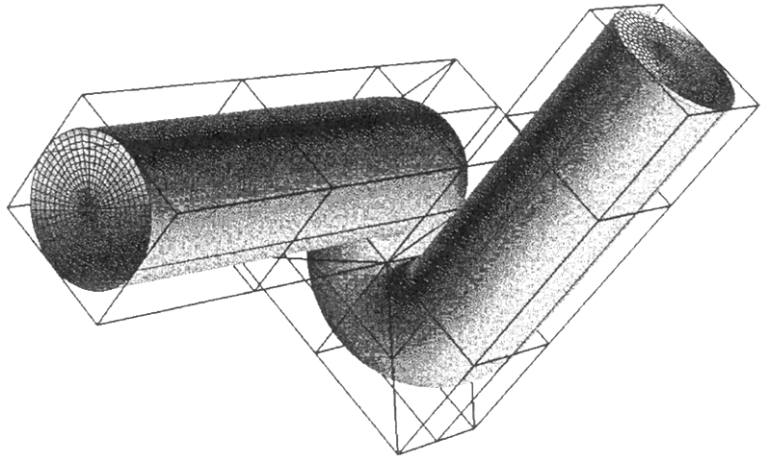

FIGURE 30.26 Volume grid created by NURBS composite volume.

30.5.5 Transfinite Interpolation Volume

Similar to the NURBS *TFI* surface algorithm [Thompson et al., 1985], this approach is frequently used to generate an *H*-type volume grid (see Chapter 3) Instead of providing four NURBS curves for a *TFI* surface, this algorithm requires six NURBS surfaces to generate a NURBS *TFI* volume. This algorithm is the extension from the surface to volume. The *Boolean* sum equation is defined as Eq. 30.19

$$PV = P_\xi \oplus P_\eta \oplus P_\zeta = P_\xi V + P_\eta V + P_\zeta V - P_\xi P_\eta V - P_\eta P_\zeta V - P_\xi P_\zeta V + P_\xi P_\eta P_\zeta V \tag{30.19}$$

The *P* could be any interpolation function, such as the linear, quadratic hermit or the cubic interpolation. The traditional definitions of each term in Eq. 30.19 can be found in [Thompson 1985, 1992]. However,

as one can find the formulas from the reference, the traditional *TFI* approach [Thompson, 1985], [Soni, 1993] cannot be applied to the process of generating a NURBS *TFI* control volume because the addition and subtraction operations in Eq. 30.19 may lead to zero or negative *weights* in the interior control volume. Any zero *weight* will make the corresponding control point lose its influence and the negative *weights* will create undesirable grids, such as the unbounded grids or crossing grids. Hence, when applying Eq. 30.19 to a NURBS *TFI* volume, it is necessary to redefine the individual terms listed in Eq. 30.19. The procedure is as follows: Suppose the six NURBS surfaces are all predefined, and the surfaces of S_1 and S_2 are used for the ξ direction. S_1 and S_2 have the same orders of *k2, k3,* and same number of control points of $n \times L$ (refer to Eq. 30.3). If the orders of these two surfaces do not match, one should perform the degree-raising algorithm to the low degree surface. If the resolutions of the control points of S_1 and S_2 are different, then the knot insertion algorithm should be used to make them the same. The same procedures should be applied to the surface of S_3, S_4 (with the *orders* of $k1$, $k3$ and the resolutions of control net of $m \times L$) and S_5, S_6 (with the *orders* of $k1$, $k2$ and the resolutions of control net of $m \times n$). After this step, each term for a linear NURBS *TFI* volume can be defined as follows: the $P_\xi V$ is a NURBS volume that is created by using the surfaces of S_1 and S_2 with the algorithm of "ruled NURBS volume" (described in the previous section of this chapter). Hence, the *three* orders of $P_\xi V$ are 2, $k2$, and $k3$, while the resolution of the control volume is $2 \times n \times L$. The same procedures should be applied to $P_\eta V$ and $P_\zeta V$. Therefore, the *orders* of $P_\eta V$ are $k1$, 2, $k3$ with the resolution of the control volume of $m \times 2 \times L$, while the *orders* of $P_\zeta V$ are $k1$, $k2$, 2 with the resolution of the control volume of $m \times n \times 2$. $P_\xi P_\eta(V)$ is a NURBS volume that is created by utilizing the boundaries (in ζ direction) of S_1, S_2 and the corner points of S_3, S_4, S_5, S_6. In other words, it has *orders* of 2, 2, $k3$ and the dimension of control volume of 2, 2, L. The $P_\eta P_\zeta(V)$ and $P_\xi P_\zeta(V)$ are defined analogously — the *orders* of the $P_\eta P_\zeta(V)$ are $k1$, 2, 2 and the resolution of the control volume is m, 2, 2, while the *orders* of the $P_\xi P_\zeta(V)$ are 2, *k2, 2* and the resolution of the control volume is *2, n,* 2. The last term of $P_\xi P_\eta P_\zeta(V)$ is simply a NURBS control volume constructed by all the corner points of the six surfaces. Hence, the *orders* of this volume are 2, 2, 2 and the size of control volume is $2 \times 2 \times 2$. These seven control volumes are illustrated in Figure 30.27.

After these seven intermediate control volumes are created, Eq. 30.20 below should be used for the final linear NURBS TFL This equation will avoid the creation of any undesired interior *weights*.

$$\left|\begin{matrix} WP \\ W \end{matrix}\right|_{ijk} = \left|\begin{matrix} W_\xi W_\eta W_\zeta \left(P_\xi + P_\eta + P_\zeta - P_\xi P_\eta - P_\eta P_\zeta + P_\xi P_\eta P_\zeta\right) \\ W_\xi W_\eta W_\zeta \end{matrix}\right|_{ijk} \tag{30.20}$$

In addition to the algorithm of NURBS *TFI* surface, one has to perform the knot insertion and degree elevation algorithms on all seven intermediate control volumes to ensure all of them have the same orders and same knot vectors in all the ξ, η, and ζ directions, respectively. After this step is completed, the sizes of all the control volumes will be the same. Hence, the final control volume for NURBS *TFI* can be obtained by adding the corresponding control points of $P_\xi(V)$, $P_\eta(V)$, $P_\zeta(V)$, $P_\xi P_\eta P_\zeta(V)$ and subtracting those of $P_\xi P_\eta(V)$, $P_\eta P_\zeta(V)$ and $P_\xi P_\zeta(V)$, while the *weights* are determined by multiplication of W_ξ, W_η, and W_ζ. Figure 30.28 shows an *H*-type nozzle generated according to this approach.

30.6 Conclusion and Summary

The geometry modeling techniques used in computer-aided geometric design have been extended and applied to numerical grid generation for CFS simulation. The generalized algorithms that convert the non-NURBS entities to NURBS (or B-spline) representations have been presented. These algorithms can be utilized to bridge the gap between the grid generation and the CAD/CAM systems. The formulation of NURBS has been extended from curves, surfaces to full 3D NURBS control volumes to model the CFS configurations. The development of the redistribution schemes on volume grids with singularity is

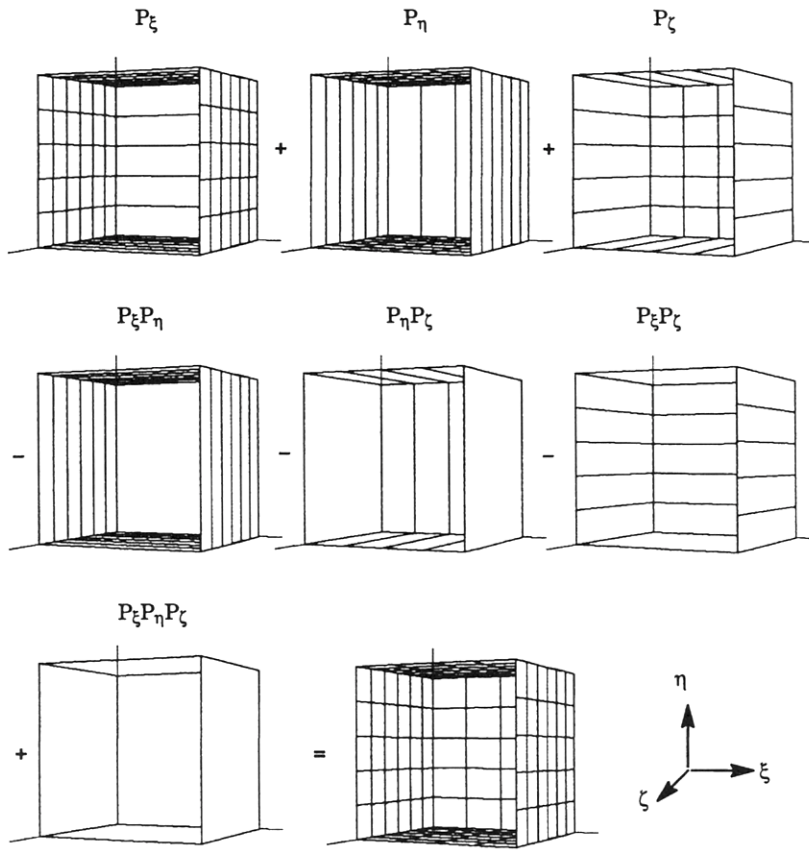

FIGURE 30.27 Illustration of NURBS *TFI* volume.

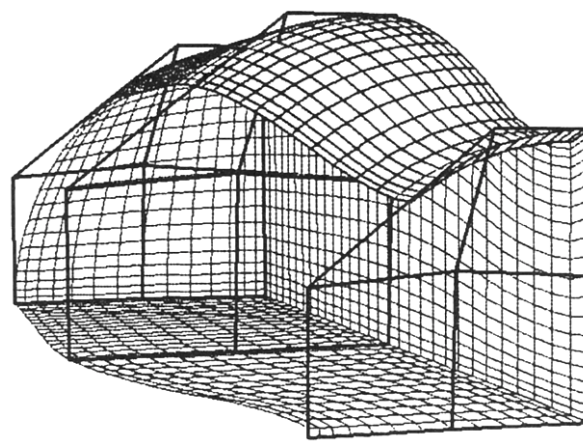

FIGURE 30.28 A volume grid for a nozzle created by NURBS volume *TFI* option.

demonstrated by computational examples. The applications of these reparametrization techniques to precise grid distribution control with accurate geometry fidelity have been demonstrated. In addition, the applications of NURBS to grid generation presented in this chapter have proven the versatility of NURBS in the CFS simulation processes.

Acknowledgment

Part of the contents of this chapter (Figures 30.1–30.6, 30.9, 30.11–12, Eqs. 30.5, 30.7, 30.8, 30.10, 30.11) were reprinted from T. Y. Yu and B. K. Soni, *Computer-Aided Design,* pp 147–157, with kind permission from Elsevier Science Ltd. The algorithms presented in this chapter were developed under a contract from NASA/Marshall Space Flight Center at Mississippi State University.

References

Farin, G.E., *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 3rd Edition. Academic Press, 1993.

Lin, F. and Hewitt, W.T., Expressing Coons–Gordon surfaces as NURBS, *J. of Computer Aided Design*. Feb.1994, Vol. 26, No. 2, pp 145–155.

Piegl, L., Infinite Control Points – A method for representing surfaces of revolution using boundary data, *IEEE Computer Graphics & Applications*. Mar. 1987, Vol. 7, No. 3, pp 45–55.

Peigl, L., Interactive data interpolation by rational Bézier curves, *IEEE Computer Graphics & Applications*. Apr. 1987, Vol. 7, No. 4, pp 45–58.

Piegl, L., On NURBS: a survey, *IEEE Computer Graphics & Applications.* Jan. 1991, Vol. 11, No. 1, pp 57–71.

Piegl, L., Rational B-Spline curves and surfaces for CAD and graphics, *State of the Art in Computer Graphics Visualization and Modeling*. Rogers, D.F. and Earnshaw, R.A., (Eds.), Springer-Verlag, 1991, pp 225–269.

Piegl, L. and Tiller, W., Curve and surface constructions using rational B-splines, *Computer Aided Design*. 1987, Vol. 19, No. 9, pp 485–498.

Piegl, L. and Tiller, W., A menagerie of rational B-splines, *IEEE Computer Graphics & Applications*. Sept. 1989, Vol. 9, No. 5, pp 48–56.

Reichert, B.A., Hingst, W.R., Okiishi, T.H., Circular-to-rectangular transition duct flow without and with inlet swirl, *J. of Propulsion and Power.* Jan.-Feb. 1994, Vol. 10, No. 1, pp 88–100.

Shih, M.H., Towards a comprehensive computational simulation system for turbomachinery, Ph.D. dissertation, Mississippi State University, May 1994.

Soni, B.K., Grid generation for internal flow configurations, *Computers and Mathematics with Applications*. 1993, Vol. 24, No. 5/6, pp 151–163.

Thompson, J.F., A survey of grid generation techniques in computational fluid dynamics, *AIAA-83-0447*, AIAA 21st Aerospace Sciences Meeting, 1983.

Thompson, J.F., National grid project, *Computing Systems in Engineering*.1992, Vol. 3, No. 1–4, pp 393–399.

Thompson, J.F., Warsi, Z.U.A., Mastin, C.W., *Numerical Grid Generation Foundations and Applications.* North-Holland, 1985.

Tiller, W., Rational B-splines for curve and surface representation, *IEEE Computer Graphics & Applications*. Sep.1983, Vol. 3, No. 10, pp 61–69.

Tiller, W., Knot removal algorithms for NURBS curves and surfaces, *Computer Aided Design*. Aug. 1992, Vol. 24, No. 8, pp 445–453.

Yu, T.Y., CAGD Techniques in grid generation, Ph.D. dissertation, Mississippi State University, Dec. 1995.

Yu, T.Y., and Soni, B.K., The application of NURBS in numerical grid generation, *J. of Computer-Aided Design*. Feb. 1995, Vol. 27, No. 2, pp 147–157.