CHAPTER

TWELVE

Standards and Data Exchange

12.1 Introduction

The central theme of this chapter is the relationship of NURBS geometry, as practiced in this book, to other methods of representing curves and surfaces, and to several important standard formats for the exchange of geometric data between two systems. We have used a specific form of knot vector throughout this book; Section 12.2 discusses another type of knot vector and the conversion between the two types. Section 12.3 describes how NURBS curves and surfaces are defined within the three standards: IGES, STEP, and PHIGS. We remark that our NURBS are compatible with the NURBS of these standards (with the exception that IGES and PHIGS allow discontinuous curves and surfaces). Finally, Section 12.4 gives some general guidelines for exchanging geometric data between a NURBS system and another system.

12.2 Knot Vectors

Knot vectors can be put into two groups, *clamped* and *unclamped*; and within each group there are *uniform* and *nonuniform* knot vectors. Some examples are:

clamped, uniform:

$$\{0,0,0,0,1,2,3,4,4,4,4\}$$

$$\{-0.5,-0.5,-0.5,1,2.5,4,4,4\}$$

clamped, nonuniform:

$$\{0,0,0,2,3,6,7,7,7\}$$

$$\{0,0,0,0,1,2,2,3,4,5,5,5,5\}$$

unclamped, uniform:

$$U_1 = \{-3, -2, -1, 0, 1, 2, 3, 4, 5\}$$

$$U_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$$

unclamped, nonuniform:

$$\{0, 0, 1, 2, 3, 4\}$$

$$\{-2, -1, 0, 4, 5, 6, 7\}$$

Hence clamped/unclamped refers to whether or not the first and last knot values are repeated with multiplicity equal to degree plus one, and uniform/nonuniform refers to the knot spacing. To be uniform and unclamped, all knot spans must be of equal length. To be uniform and clamped, only the "internal" knot spans must be of equal length. The reader has probably noticed that what we are now calling clamped, uniform and nonuniform, is exactly that which we have called *nonperiodic*, uniform and nonuniform, up to now. Following a brief study of unclamped knot vectors, we will offer a flimsy excuse for this changing of the terminology horse in the middle of the knot stream. Since clamped knot vectors are assumed throughout the book, they require no further elaboration here. However, before proceeding to a study of unclamped knot vectors we offer the following reminders relative to clamped knot vectors:

- if we are given a clamped knot vector with $m+1$ knots and end multiplicities of $p+1$, then we require $n+1$ $(= m-p)$ control points to define a pth degree curve; that is, the knot vector determines the degree, number of control points, and the valid parameter range;
- clamped curves can be open or closed, geometrically; if a curve is closed, then whether or not it is closed with C^k continuity depends on the first k and last k internal knot spans, and the first $k+1$ and last $k+1$ weights and control points.

Now consider the unclamped, uniform knot vector, U_1. Using the Cox-deBoor algorithm, Eq. (2.5), we can compute $8-p$ degree p basis functions, for $p = 0, \ldots, 7$. The degree 2 and 3 basis functions, along with a quadratic and cubic curve, are shown in Figures 12.1a and 12.1b. The knot locations are marked on the curves. Notice that the curves are not clamped at the ends, i.e., their start and end points do not coincide with control points. Furthermore, evaluation of the curves at $u \in [u_i, u_{i+1})$ requires the functions $N_{i-p,p}(u), \ldots, N_{i,p}(u)$; thus, the curves are defined only on the parameter range $u \in [u_p, u_{m-p}]$, as is the case for clamped curves. Thus, the quadratic curve of Figure 12.1a is defined on $u \in [-1, 3]$, and the cubic curve of Figure 12.1b is defined on $u \in [0, 2]$. Figure 12.2 shows how to define a closed, cubic curve using the unclamped, uniform knot vector, U_2. Since $m = 11$ and $p = 3$, we require eight control points $(n = 7)$. We wrap around p $(= 3)$ control points at the end; i.e., $\mathbf{P}_5 = \mathbf{P}_0$, $\mathbf{P}_6 = \mathbf{P}_1$ and $\mathbf{P}_7 = \mathbf{P}_2$. The valid parameter range is $u \in [3, 8]$. This yields curve closure, with C^2 continuity.

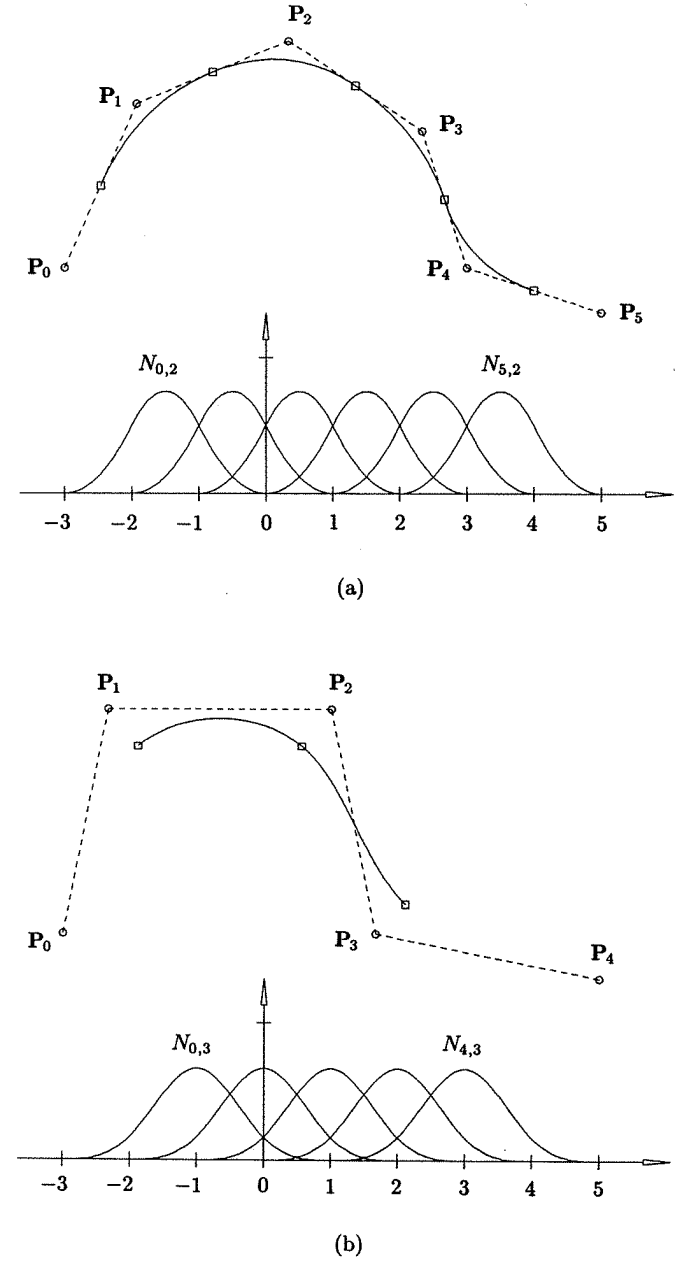


Figure 12.1. Unclamped NURBS curves. (a) Quadratic curve; (b) cubic curve.

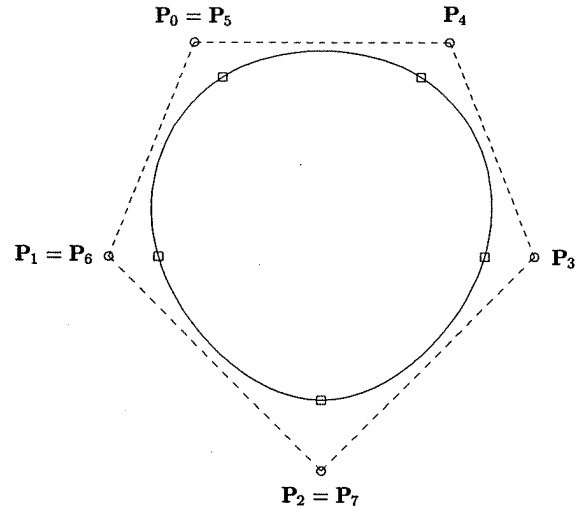


Figure 12.2. Closed C^2 continuous unclamped cubic curve.

Everything stated previously for unclamped, uniform curves also holds for unclamped, nonuniform curves, with the exception of C^{p-1} continuity at the start/end of a closed curve. Since continuity depends on knot values as well as on control point locations, C^{p-1} continuity of a closed, nonuniform curve does not necessarily imply precise wraparound of the control points. Figure 12.3a shows a closed, cubic curve defined on the clamped, uniform knot vector

$$U = \{3, 3, 3, 3, 4, 5, 6, 7, 8, 8, 8, 8\}$$

The curve is C^2 continuous at the start/end point. Figures 12.3b and 12.3c show unclamped versions of (precisely) the same curve; the curve in Figure 12.3b uses the knots

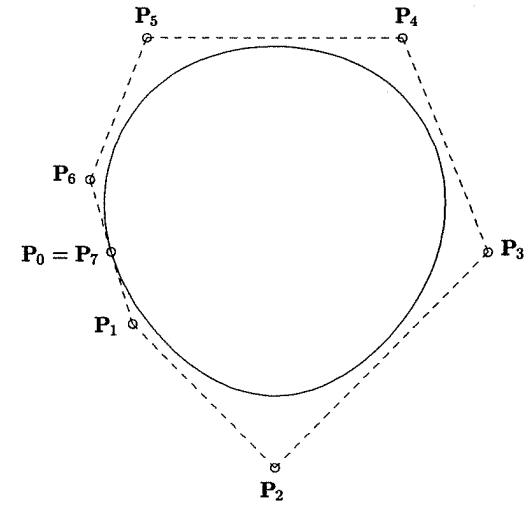
$$U = \{0, 0.5, 2.3, 3, 4, 5, 6, 7, 8, 8.7, 10.5, 11\}$$

and the curve in Figure 12.3c uses the knots

$$U = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$$

It is important to point out here that, algorithmically, there is little difference between clamped and unclamped curves. With only slight modifications, all the fundamental algorithms given in the first five chapters of this book can also be made to work for unclamped curves.

In the early days of B-splines in CAD/CAM (before the early 1980s), very little serious work was done using nonuniform knot vectors, clamped or unclamped. Although terminology has always been confusing in this area, early CAD/CAM workers commonly referred to B-splines as either *nonperiodic* or *periodic*;

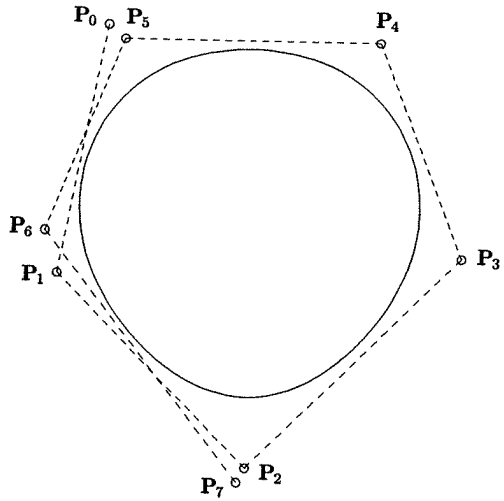


(a)

Figure 12.3. Examples of unclamping a closed curve with different knot vectors. (a) Closed clamped curve; (b) unclamping with no wraparound; (c) unclamping with wraparound.

nonperiodic was synonymous with clamped (as it is in this book), and these were used to define open curves and surfaces (uniform at first, and nonuniform later). Periodic was mathematically equivalent to unclamped, uniform, and these were used to construct C^{p-1} continuous closed curves and surfaces. Referring to Figures 12.1a and 12.1b, one sees that the unclamped, uniform basis functions are all translates of one another, hence they are often called periodic. For periodic B-splines there was no need to store a knot vector, the Cox-deBoor algorithm was not required, and control point wraparound was not necessary to obtain C^{p-1} continuous closed curves. A simple matrix formulation of the (one) basis function, together with a special control point indexing scheme, were sufficient to evaluate periodic B-spline curves (see [Mort85] for a detailed discussion). Although more efficient, this special treatment of periodic B-splines has all but disappeared from practice, probably due to the fact that the time required for software development and maintenance has become much more expensive than CPU time. Clamped and unclamped are newer terms which will probably grow in use, simply because they are more intuitive and accurate. Although one sees these terms rather infrequently in the literature, they seem to be quite prevalent in the CAD/CAM community. We use the term nonperiodic in this book simply because we have not yet managed to break the habit.

Many systems today use only clamped B-splines. However, both IGES and STEP provide for the exchange of unclamped B-splines. Precise conversions are



(b)

Figure 12.3. (*Continued.*)

possible both ways. Clamping a curve is nothing more than inserting the knots u_p and u_{m-p} until they have multiplicity p, and then discarding the knots and control points lying outside the clamped region. The knot insertion formula, Eq. (5.15), is also valid for unclamped curves, as long as one does not index knots outside the range 0 to m (which does not happen when clamping at u_p and u_{m-p}). We recommend (but leave as an exercise) that the reader write a routine to extract the subcurve between arbitrary parameters, u_1 and u_2, where $u_1 < u_2$. Such a routine not only handles clamping but also has many other uses, including one which we mention in the next section.

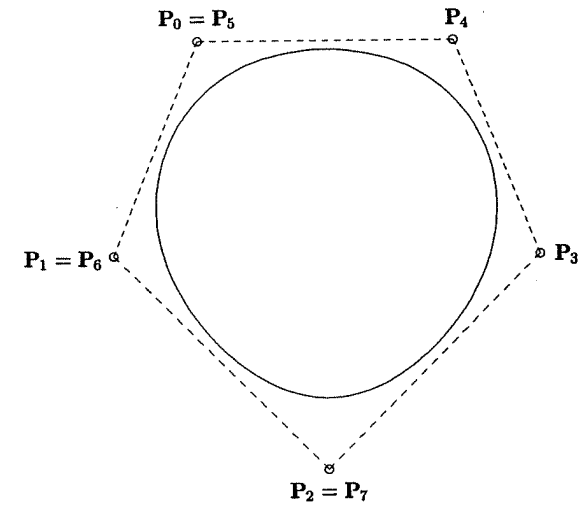
Unclamping is essentially knot removal. Although unclamping is rarely required (yes, there are systems which only handle unclamped B-splines), we present here Algorithm `A12.1`, which computes the $p-1$ new control points at each end and p new knots at each end, all in place. We remind the reader that the result is not unique (see Figures 12.3b and 12.3c). One is free to choose the $2p$ new end knots rather arbitrarily, and the new control point locations depend on the knots. We choose the new knots as follows

$$u_{p-i-1} = u_{p-i} - (u_{n-i+1} - u_{n-i}) \qquad i = 0, \ldots, p-1 \qquad (12.1)$$

and

$$u_{n+i+2} = u_{n+i+1} + (u_{p+i+1} - u_{p+i}) \qquad i = 0, \ldots, p-1 \qquad (12.2)$$

This choice of knots produces wraparound of the control points, as in Figures 12.2 and 12.3c, in case the clamped curve is closed with C^{p-1} continuity. The algorithm is:



(c)

Figure 12.3. (*Continued.*)

```
ALGORITHM A12.1
  UnclampCurve(n,p,U,Pw)
    { /*  Unclamp a clamped curve  */
      /*  Input:  n,p,U,Pw  */
      /*  Output: U,Pw  */
    for (i=0; i<=p-2; i++)   /* Unclamp at left end */
      {
      U[p-i-1] = U[p-i] - (U[n-i+1]-U[n-i]);
      k = p-1;
      for (j=i; j>=0; j--)
        {
        alfa = (U[p]-U[k])/(U[p+j+1]-U[k]);
        Pw[j] = (Pw[j]-alfa*Pw[j+1])/(1.0-alfa);
        k = k-1;
        }
      }
    U[0] = U[1] - (U[n-p+2]-U[n-p+1]);   /* Set first knot */
    for (i=0; i<=p-2; i++)   /* Unclamp at right end */
      {
      U[n+i+2] = U[n+i+1] + (U[p+i+1]-U[p+i]);
      for (j=i; j>=0; j--)
        {
```

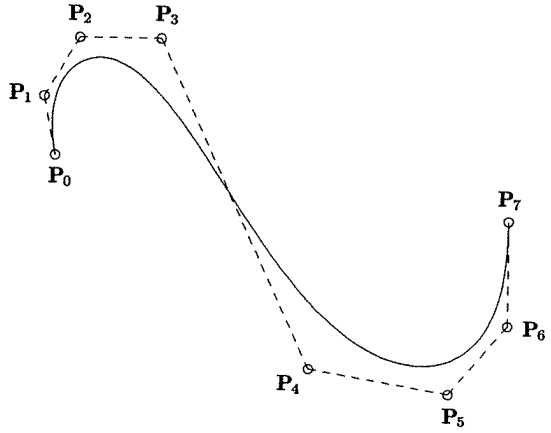
```
        alfa = (U[n+1]-U[n-j])/(U[n-j+i+2]-U[n-j]);
        Pw[n-j] = (Pw[n-j]-(1.0-alfa)*Pw[n-j-1])/alfa;
        }
    }
U[n+p+1] = U[n+p] + (U[2*p]-U[2*p-1]);   /* Set last knot */
}
```

Figures 12.4a–12.4d show the steps in unclamping the left end of a degree 4 curve. Figures 12.4b–12.4d correspond to the completion of the ith step, $i = 0, 1, 2$, respectively. Figure 12.5a shows the quadratic full circle, with knots

$$U = \left\{0,0,0,\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},\frac{3}{4},1,1,1\right\}$$

(see Example Ex7.2). The nine control points are marked. Figure 12.5b illustrates the circle after unclamping by Algorithm A12.1. Notice that the control polygon does not wrap around, in the sense that none of the control points coincide. This is because the circle is only C^0 continuous in homogeneous space. The knot vector after unclamping is

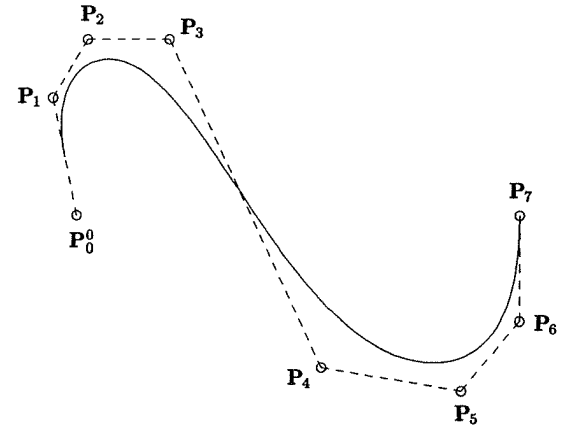
$$U = \left\{-\frac{1}{4},-\frac{1}{4},0,\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},\frac{3}{4},1,\frac{5}{4},\frac{5}{4}\right\}$$



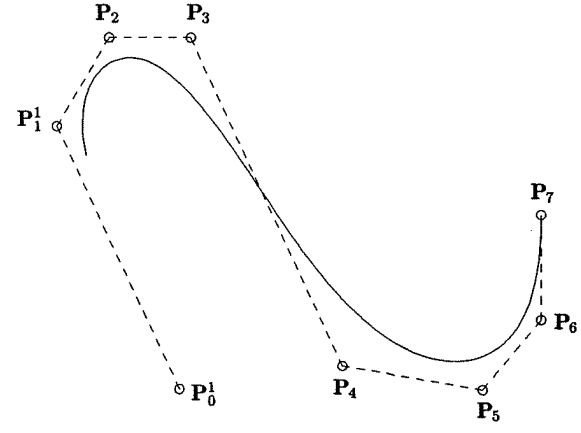
(a)

Figure 12.4. The process of unclamping a quartic curve at its left end. (a) Original clamped curve; (b) $i = 0; \rightarrow \mathbf{P}_0^0$ computed; (c) $i = 1; \rightarrow \mathbf{P}_0^1$ and $\mathbf{P}_1^1$ computed; (d) $i = 2; \rightarrow \mathbf{P}_0^2, \mathbf{P}_1^2$, and $\mathbf{P}_2^2$ computed.

Clearly, the extension of these results to surfaces is straightforward. A surface is unclamped in the u (v) direction by applying the computations of Algorithm A12.1 to the $m+1$ rows ($n+1$ columns) of control points. Figures 12.6a–12.6d show examples of surface unclamping. The original degree (3, 2) surface is shown in Figure 12.6a. Unclamping in the u direction is illustrated in Figure 12.6b, whereas unclamping in the v direction is pictured in Figure 12.6c. Figure 12.6d shows unclamping in both directions.

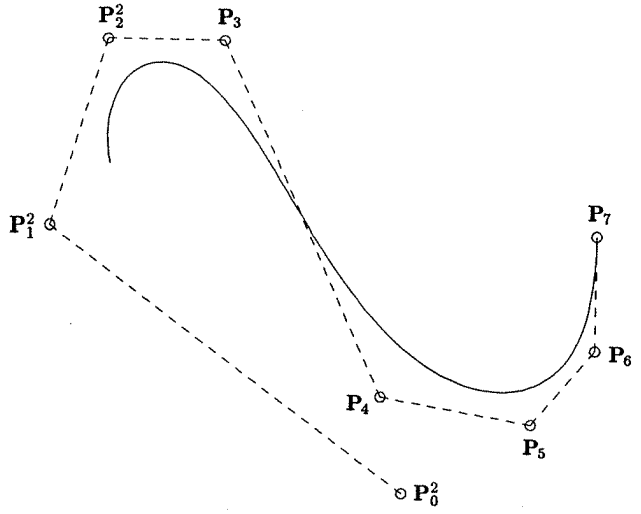


(b)



(c)

Figure 12.4. (*Continued.*)



(d)

Figure 12.4. (*Continued.*)

12.3 NURBS Within the Standards

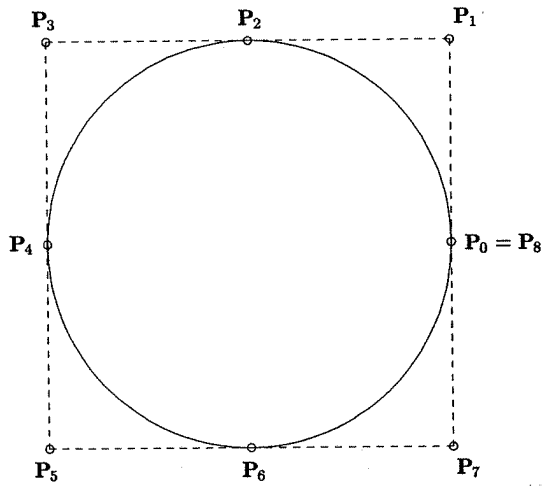
NURBS are not only a *de facto* standard throughout the CAD/CAM/CAE industry, but they are also incorporated into several international and American national standards:

- IGES : Initial Graphics Exchange Specification;
- STEP : Standard for the Exchange of Product Model Data;
- PHIGS : Programmer's Hierarchical Interactive Graphics System.

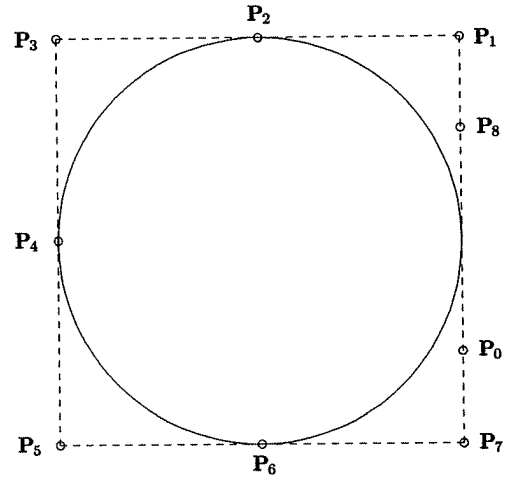
The definition of NURBS within these standards is the topic of this section. The purpose is to convey, conceptually, what defines a NURBS in each of the standards. To this end, we adopt an informal style and avoid the specific terminology of each standard. The reader should consult the relevant documents for a more rigorous study of these standards [PHIG92; IGE93; STEP94]. For additional reading see Bloor and Owen [Blor91] and Vergeest [Verg91] for STEP, and Howard [Howa91a] and Howard et al. [Howa91b] for PHIGS. Anyone designing a new NURBS system should ensure that it accommodates these standards as they relate to NURBS.

12.3.1 IGES

IGES, an American National Standard (ANS), is the most widely used format for exchanging product data among today's CAD/CAM/CAE systems. Developed



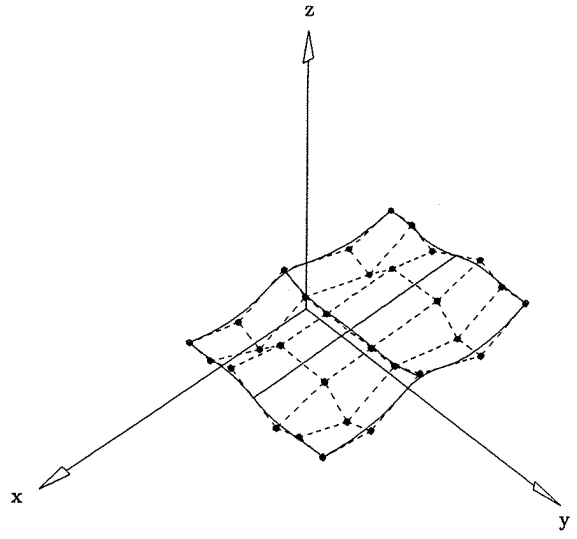
(a)



(b)

Figure 12.5. Unclamping the nine-point NURBS circle. (a) Original curve; (b) unclamped curve.

in the early 1980s, IGES specifies formats for exchanging graphics and geometry data, with support for various applications such as drafting, circuit design, finite elements, and piping. Curves, surfaces, and three-dimensional solids are supported. A NURBS curve is specified in IGES by:

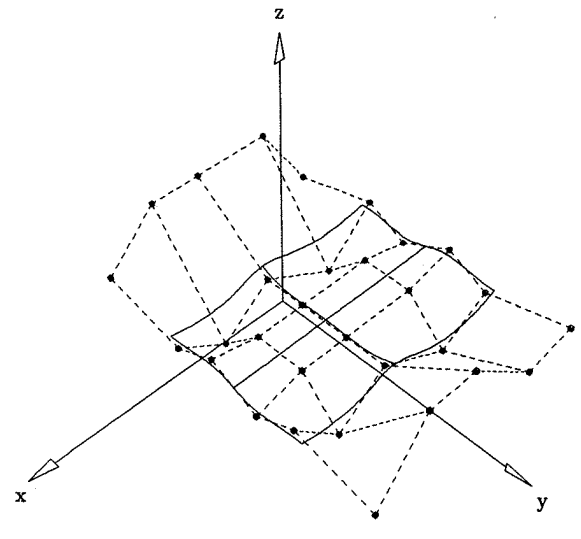


(a)

Figure 12.6. Unclamping a degree (3,2) surface. (a) Original surface; (b) unclamping in the u direction; (c) unclamping in the v direction; (d) unclamping in both directions.

- degree, p, and number of control points, $n+1$;
- Euclidean control points, $\mathbf{P}_i$, and weights, w_i;
- a knot vector, U, containing $m+1 = n+p+2$ knots;
- start and end parameter values, s_0 and s_1;
- other nonessential but useful information: whether the curve is planar or nonplanar, open or closed, truly rational (the w_i not all equal) or nonrational, periodic or nonperiodic; furthermore, a curve can be tagged as a special type, e.g., a line, or a circular or a conic arc.

Notice that control points and weights are separate items in IGES; there is no concept of homogeneous control points, $\mathbf{P}_i^w$. Only positive weights are allowed. The terms "periodic" and "nonperiodic" are not defined in the IGES manual, and therefore they are meaningless. Most often they are probably interpreted to mean unclamped and clamped, respectively, as defined in Section 12.2. The only constraints on the knots are $m = n+p+1$ and $u_{i-1} \le u_i$ for $i = 1, \ldots, m$. Thus just about anything is allowed, including internal knots with multiplicity greater than p. The parameters s_0 and s_1 are an integral part of the definition, because they define the start and end points of the curve. The only restriction is that $u_0 \le s_0 < s_1 \le u_m$. That is, the intended curve can be a proper subcurve of the larger curve defined by the knots $u_0, \ldots, u_m$. A system which allows only clamped curves and no s_0 and s_1 values can accommodate IGES NURBS curves



(b)

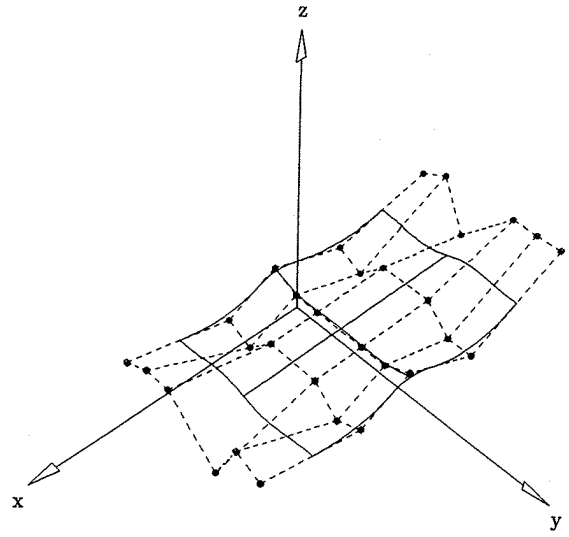
Figure 12.6. (*Continued.*)

(including unclamped curves) by applying subcurve extraction, which is simply knot insertion at s_0 and s_1, by using Algorithm `A5.1`.

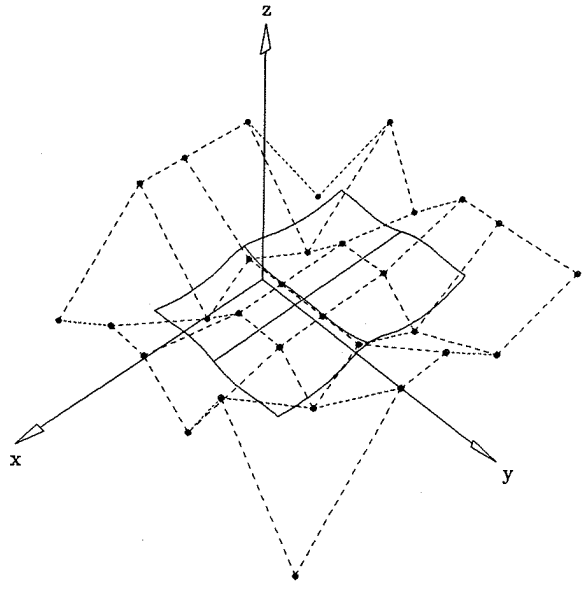
IGES NURBS surfaces are defined analogously. Control points and weights are separate; weights must be positive. The only restrictions on the u and v knots are $r = n+p+1$, $s = m+q+1$, $u_{i-1} \le u_i$ for $i = 1, \ldots, r$, and $v_{j-1} \le v_j$ for $j = 1, \ldots, s$, where p and $n+1$, and q and $m+1$, are the degrees and the numbers of control points in the u and v directions, respectively. Parameters s_0, s_1 and t_0, t_1 define the intended surface, which can be a proper subsurface of the surface defined by the given knots, weights, and control points. If necessary, knot insertion is used to extract and clamp the intended surface. There are informational flags indicating whether the surface is open or closed, and periodic (?) or nonperiodic (?) in each direction, and whether the surface is truly rational (not all $w_{i,j}$ equal) or nonrational. An IGES NURBS surface can be tagged as one of nine special types: plane, circular cylinder, cone, sphere, torus, surface of revolution, general cylinder, ruled surface, or general quadric surface.

12.3.2 STEP

STEP is an emerging international standard for the exchange of product model data. It represents a major international effort, and the complexity, breadth, and depth of STEP dwarfs that of IGES. NURBS curves and surfaces are defined in the so-called Part 42, that part of STEP concerned with the basic geometry



(c)



(d)

Figure 12.6. (*Continued.*)

and topology of a product model. In fact, because of the generality, numerical stability, and broad industrial acceptance of the NURBS representation, it was chosen as the *exclusive* method of exchanging general spline and single-span polynomial and rational curves and surfaces through STEP. Part 42 includes curves, surfaces, and three-dimensional solids. A NURBS curve is specified in STEP by:

- degree, p, and number of control points, $n+1$;
- Euclidean control points, $\mathbf{P}_i$, and weights, w_i (there is no concept of $\mathbf{P}_i^w$); only positive weights are allowed;
- a knot vector, U, containing $m+1 = n+p+2$ knots, and satisfying $u_{i-1} \leq u_i$ for $i = 1, \ldots, m$; STEP bounds the multiplicity of knots – the first and last knot values can have multiplicity at most $p+1$, and the maximum multiplicity of all internal knot values is p;
- additional information such as the type of knot vector, whether the curve is open or closed, whether or not it is self-intersecting, and a type tag if applicable (polyline, circle, conic).

The concept of "trimming" a curve by parameters s_0 and s_1, $u_0 \leq s_0 < s_1 \leq u_m$, also exists in STEP, but at a higher level; it is not embedded in the definition of the curve. STEP distinguishes four types of NURBS curves, based on the form of their knot vectors: uniform, quasi-uniform, Bézier, and nonuniform. A knot vector is uniform if $u_i - u_{i-1} = d$ for $i = 1, \ldots, m$ and some positive constant d. This corresponds to unclamped and uniform in the terminology of Section 12.2. Quasi-uniform corresponds to clamped and uniform, i.e., $u_0 = \cdots = u_p$, $u_{m-p} = \cdots = u_m$, and $u_i - u_{i-1} = d$ for $i = p+1, \ldots, m-p$ and some positive constant, d. Bézier means that the *distinct* knot values are equally spaced, and that the first and last have multiplicity $p+1$ and all internal ones have multiplicity p, i.e., the curve is piecewise Bézier. Any knot vector not falling into one of these three categories is called nonuniform.

A STEP NURBS surface is defined analogously. Control points and weights are separate, and weights must be positive. A surface is uniform, quasi-uniform, or (piecewise) Bézier only if both its u and v knots are of the respective type; otherwise it is nonuniform. There is a self-intersection flag, an open/closed flag for both u and v directions, and, if applicable, a tag indicating a special type, such as plane, cylinder, ruled surface, etc.

12.3.3 PHIGS

Whereas IGES and STEP are data exchange standards, PHIGS is an international standard specifying a device-independent interactive graphics programming interface. PHIGS emerged in the mid 1980s, and NURBS were incorporated into the standard in 1992 as part of the PHIGS PLUS extension [Howa91b]. A PHIGS implementation supporting NURBS allows an application programmer to pass down NURBS curves and surfaces directly for display. A NURBS curve is defined in PHIGS by:

- order (= *degree* + 1), number of control points, and number of knots;
- the Euclidean control points, $\mathbf{P}_i$, if the curve is nonrational, otherwise the four-dimensional homogeneous control points, $\mathbf{P}_i^w$; a flag indicates rationality. Weights must be positive;
- the knots; only clamped knot vectors are allowed, either uniform or nonuniform. There is no bound on the multiplicity of knots;
- trimming parameters s_0 and s_1, defining the actual start and end of the intended curve.

NURBS surfaces are defined analogously, except there are no trimming parameters. More general trimming loops, consisting of NURBS curves in the surface's parameter domain, are allowed.

12.4 Data Exchange to and from a NURBS System

Throughout this section we assume two curve and surface systems, A and B, and we study the exchange of data between them. We assume that System A was designed and implemented based upon this book, i.e., it uses clamped NURBS curves and surfaces of arbitrary degree. For the sake of practicality and compatibility with most existing systems, we also require that all weights be positive, and that the multiplicity of internal knots not exceed the degree. For the moment we make no assumptions about how geometry is represented in System B.

When exchanging a curve or surface between two systems which represent geometry differently, there are three possible scenarios:

- a mathematically precise conversion is possible, which means that floating point roundoff is the only error incurred. The geometry is both geometrically and parametrically equivalent in the two systems; an example is the conversion between B-spline and piecewise power basis forms given in Section 6.6;
- the conversion is geometrically precise, but the parameterization is altered; an example is the circle, parameterized by trigonometric functions in System B. Notice that if the circle is used to form certain types of surfaces in Systems A and B (such as ruled, skinned, Gordon and Coons), the resulting surface geometry can be different;
- mathematically there exists no precise conversion between the two systems; in this case, an approximation to some specified tolerance is the best that can be done. For example, if System B allows only cubic curves, then higher degree curves from System A must be approximated.

For our purposes we distinguish two types of geometry:

- simple curves and surfaces of analytic geometry, such as lines, circles, conics, planes, circular or conic cylinders and cones, spheres, and tori;
- everything else.

Assuming that System B can represent the analytic types, the exchange of these between Systems A and B is geometrically precise in both directions. Consider the direction from B to A. Most systems represent the analytic types either in a rational form (e.g., rational power basis or Bézier), or in a form from which it is easy to derive the geometric characteristics such as center, axes, radii, vertices, etc. Section 6.6 shows how to convert from piecewise rational power basis or Bézier form to NURBS; going from other rational forms to NURBS is similar. Constructing the NURBS representations of the analytic types from their geometric characteristics is the topic of Chapters 7 and 8. Passing the analytic types from System A to B is more difficult. The problem is that the NURBS representation of a curve or surface is not unique. For example, there are infinitely many combinations of degree, knots, control points, and weights which produce the full unit circle centered at the origin. Hence it is not simple to determine if a NURBS curve is an analytic type and, if so, to compute its geometric characteristics. Section 7.6 presents a method to determine the geometric characteristics of a quadratic rational Bézier curve. However, degree reduction and knot insertion may be required to bring the NURBS curve into piecewise quadratic rational form (if possible). Moreover, the NURBS curve is a conic only if it yields the same geometric characteristics (to within a tolerance) on each Bézier segment. We emphasize that we make assumptions only about what constitutes a NURBS curve or surface in System A; we make no assumptions about how the specific analytic types are constructed as NURBS. Indeed, such geometry may be created in another system, in NURBS form, but using different combinations of knots, weights, and control points than presented in this book, and then passed into System A. Providing type-tags for NURBS geometry, as allowed in IGES and STEP (see Section 12.3), offers some help in this regard; but it does not eliminate the problem.

Now consider more complex geometry, ranging from general ruled and revolved surfaces to offsets, blends, fillets, and general free-form curves and surfaces. System B might use three methods to represent such geometry:

- implicit equations of the form $f(x, y) = 0$ or $f(x, y, z) = 0$;
- procedural definitions;
- one of the many forms of polynomial or rational splines.

Theoretically, a precise conversion of a NURBS to piecewise implicit form is possible, using techniques known as *implicitization* (e.g., see [deMo84; Sede84; Hoff89] and the references therein). However, it is not computationally practical; precise conversion from implicit to NURBS form is generally not possible, and thus approximation is required. In practice, implicit equations are rarely used to represent complex geometry, therefore we pursue the topic no further.

We say that a curve or surface is defined procedurally if it is not defined directly in terms of explicit coefficients or functions, but rather indirectly by means of other *base* curves or surfaces, together with a procedure or formula to compute points on the intended geometry from points on the base geometry.

Offset curves and *surfaces* are examples of procedurally defined geometry. An offset surface, $\mathbf{O}(u,v)$, is specified by

$$\mathbf{O}(u,v) = \mathbf{S}(u,v) + d\mathbf{N}(u,v) \tag{12.3}$$

where $\mathbf{S}(u,v)$ is the base surface, $d \neq 0$ is a constant scalar, and $\mathbf{N}(u,v)$ is the unit length normal vector of $\mathbf{S}(u,v)$ at (u,v). Offset curves and surfaces are mathematically complex and, in general, can rarely be represented precisely in NURBS form (see [Till84; Faro86; Coqu87b; Hosc88]). Blend and fillet surfaces are sometimes defined procedurally [Choi91], and Filip and Ball [Fili89] describe a procedural method of skinning. Such surfaces can seldom be passed into a NURBS (or any other) system without approximation.

Most CAD/CAM/CAE systems use piecewise polynomial or rational curves and surfaces to represent complex geometry. We refer to these, collectively, as splines. There are many different types of splines; but no matter what fancy name it has, a spline curve (surface) simply consists of a number of segments (patches), pieced together with some form of continuity. For brevity, we restrict our discussion to curves for the remainder of this section, but similar statements and algorithms hold for surfaces. In order to determine if a particular type of spline curve has an equivalent NURBS representation (and vice versa), one must consider three things:

- degree;
- rationality;
- continuity.

Degree and rationality are easy. Degree can be precisely raised, but not lowered; and a polynomial is a rational function, but the converse is not true. Since our System A allows rational curves of arbitrary degree, there are no problems in this regard in bringing geometry from System B into System A. However, if System B restricts degree or allows only nonrational geometry, then the corresponding curves must be approximated in going from System A to System B.

Continuity is more complicated. Let $\mathbf{C}(u)$ be a spline curve consisting of the segments $\mathbf{C}_i(u)$, $i = 1, \ldots, m$. $\mathbf{C}_i(u)$ is defined on the interval $[u_{i-1}, u_i]$, where $u_0 < u_1 < \cdots < u_{m-1} < u_m$. We call the values $\{u_i\}$ the *breakpoints*; if $\mathbf{C}(u)$ is in B-spline form, then the $\{u_i\}$ are the distinct knot values. Each pair of segments, $\mathbf{C}_i(u)$ and $\mathbf{C}_{i+1}(u)$, joins with some *type* and *order* of continuity, and the fundamental question in data exchange always boils down to whether or not the receiving system can represent the segments, connected with the appropriate continuity. By continuity type we mean parametric (C) versus geometric (G) continuity; see Section 9.3.1 and [Bars89, 90; Hosc93] and the references therein for more details on geometric continuity and various types of splines constructed to have G continuity. We remind the reader that:

- G^1 continuity at u_i means that the first derivative vectors, $\mathbf{C}_i'(u_i)$ and $\mathbf{C}_{i+1}'(u_i)$, point in the same direction but can have different magnitudes;
- for B-splines, knot multiplicity indicates continuity of the basis functions, which is generally the parametric continuity of the curve in homogeneous space;
- reparameterization of a curve segment changes its parametric continuity with its neighbors, but not its geometric continuity.

Since System A allows arbitrary knot spacing and multiplicity, it is capable of representing precisely, geometrically and parametrically, virtually all polynomial and rational spline curves which are at least C^0 continuous. However, going from A to B can require either an approximation or reparameterization of the segments (which does not change the overall curve geometry). For example, if System B uses rational B-splines but does not allow multiple internal knots, then a C^{p-2} continuous B-spline curve with double internal knots must be approximated. If System B allows multiple knots but requires distinct knots (breakpoints) to be equally spaced, then all curves of System A can be passed geometrically precisely into System B; however, reparameterization is required, and parametric continuity is lost in the process. The algorithm is:

1. insert all internal knots until they have multiplicity p, thus obtaining a piecewise Bézier curve; this yields the control points for the curve in System B;
2. let
$$d = \frac{u_m - u_0}{m}$$
and define the knot vector for System B to be
$$U = \{\underbrace{u_0, \ldots, u_0}_{p+1}, \underbrace{u_1, \ldots, u_1}_{p}, \ldots, \underbrace{u_{m-1}, \ldots, u_{m-1}}_{p}, \underbrace{u_m, \ldots, u_m}_{p+1}\}$$
where $\qquad u_i = u_0 + i \cdot d \qquad i = 1, \ldots, m-1$
3. apply knot removal (to a zero tolerance), but all breakpoints must remain with multiplicity at least one so that they are equally spaced.

Note that if this algorithm is used to pass a curve from A to B and then back to A, the original and the returned curves in A differ parametrically and in their number of knots. Notice also that, even though the curves are geometrically equivalent, if they are used in System A to construct ruled surfaces (for example) then the resulting surfaces may not be geometrically equivalent.

Regardless of the type of spline used in System B, the continuity issue is comprised of two parts:

- does the representation allow for nonuniformly spaced breakpoints;
- what types and orders of continuity are possible?

We remark that the first part can be answered even for systems in which there is no concept of global breakpoints. For example, many geometrically continuous splines are constructed by local methods using a local parameter t, $t \in [0,1]$,

for each segment. Such systems may allow for a global parameterization, in which case global breakpoints, u_i, are available, and nonuniform spacing is most probably allowed. In systems with no concept of a global parameterization, equally spaced global breakpoints are implied. As seen previously, reparameterization can be used to transform nonuniformly spaced breakpoints into uniformly spaced ones; however, parametric continuity is altered in the process. Conversely, reparameterization which transforms uniformly spaced breakpoints to nonuniformly spaced ones is useful in converting (geometrically precisely) a G^k continuous spline to a C^k continuous B-spline (e.g., this was done in Section 9.3.2, Eqs. [9.37]–[9.39]). With regard to the second part, one must determine the minimum order of continuity allowed. For example, a system based on fifth-degree G^2 continuous splines can represent all quintic G^k continuous splines, for $k \geq 2$; however, a quintic G^k continuous spline with $k < 2$ must be approximated. As another example, curves of a B-spline system allowing only single internal knots are C^{p-1} continuous. Hence all orders of continuity are possible, but the degree must be at least one greater than the minimum desired continuity (i.e., the minimum order of continuity may depend on the degree).

We now summarize the exchange of spline curves between Systems A and B. All spline curves which are at least C^0 continuous can be passed precisely (geometrically and parametrically) from System B to A. Multiple knots are necessary for G continuous curves, but, if desired, reparameterization can be used to achieve C continuity and lower the multiplicity of internal knots. The use of reparameterized curves in certain types of surface constructions (e.g., ruled surfaces) can change surface geometry. A typical algorithm to effect the conversion first computes either the power basis or Bézier coefficients for each segment, and it then applies the techniques of Section 6.6 to build the NURBS representation.

The conversion from System A to System B can be more complicated. Algorithm A12.2 gives the logical thought process necessary to determine how such a conversion should proceed. Denote the pth degree NURBS curve by $\mathbf{C}$, and assume System B can represent splines of degree k, where $p_{\min} \leq k \leq p_{\max}$.

```
ALGORITHM A12.2
  if (C is rational and System B only represents polynomials)
    {
    Approximate C;
    return;
    }
  if (p > p_max)
    {
    Approximate C;
    return;
    }
  if (p < p_min)
    {
    DegreeElevateCurve(C);    /* to degree p_min */
    return;
    }
  Let m be the minimum order continuity allowed in System B
    for a degree p curve.
  Determine the minimum order, k, of continuity at any
    breakpoint of C.
  if (k < m)
    {
    Approximate C;
    return;
    }
  Extract the segments of C (knot insertion), and convert them
    to the segment coefficients required by System B
    (reparameterization may be necessary).
  If applicable, load the global breakpoints into System B.
```


Determining the minimum continuity values, m and k, requires elaboration. There is C continuity and G continuity, and for rational curves there is continuity in homogeneous space versus continuity in Euclidean space. The values m and k must refer to the type of continuity on which System B's splines are based. For example, suppose System B uses a rational G^2 continuous spline. This spline is based on tangent and curvature continuity in Euclidean space, hence the value k must be based on the G continuity of System A's NURBS curve in Euclidean space. Notice that the quadratic full circle of Example Ex7.2 is precisely representable as a G^2 spline in System B, even though, as a NURBS curve, it is only C^1 continuous in Euclidean space and even has cusps in homogeneous space.

We close this chapter with the observation that the NURBS form is not only a powerful and flexible method for curve and surface design, but it is also clearly today's most suitable representation for the exchange of complex curve and surface data.