B-Spline Curves

B-splines were investigated as early as the nineteenth century by N. Lobachevsky (see Renyi [506], p. 165); they were constructed as convolutions of certain probability distributions.[1] In 1946, I. J. Schoenberg [542] used B-splines for statistical data smoothing, and his paper started the modern theory of spline approximation. For the purposes of this book, the discovery of the recurrence relations for B-splines by C. de Boor [137], M. Cox [129], and L. Mansfield was one of the most important developments in this theory. The recurrence relations were first used by Gordon and Riesenfeld [284] in the context of parametric B-spline curves.

This chapter presents a theory for arbitrary degree B-spline curves. The original development of these curves makes use of divided differences and is mathematically involved and numerically unstable; see de Boor [138] or Schumaker [546]. A different approach to B-splines was taken by de Boor and Höllig [143]; they used the recurrence relations for B-splines as the starting point for the theory. In this chapter, the theory of B-splines is based on an even more fundamental concept: the *blossoming* method proposed by L. Ramshaw [498] and, in a different form, by P. de Casteljau [147]. More literature on blossoms: Gallier [252], Boehm and Prautzsch [87].

8.1 Motivation

B-spline curves consist of many polynomial pieces, offering much more versatility than do Bézier curves. Many B-spline curve properties can be understood by considering just one polynomial piece—that is how we start this chapter.

The Bézier points of a quadratic Bézier curve may be written as blossom values

$$\mathbf{b}[0,0], \mathbf{b}[0,1], \mathbf{b}[1,1].$$

Based on this, we could get the de Casteljau algorithm by repeated use of the identity $t = (1-t)\cdot 0 + t \cdot 1$. The pairs $[0,0], [0,1], [1,1]$ may be viewed as being obtained from the sequence $0, 0, 1, 1$ by taking successive pairs.

Let us now generalize the sequence $0, 0, 1, 1$ to a sequence u_0, u_1, u_2, u_3. The quadratic blossom $\mathbf{b}[u,u]$ may be written as

$$\begin{aligned}\mathbf{b}[u,u] &= \frac{u_2-u}{u_2-u_1}\mathbf{b}[u_1,u] + \frac{u-u_1}{u_2-u_1}\mathbf{b}[u,u_2]\\ &= \frac{u_2-u}{u_2-u_1}\left(\frac{u_2-u}{u_2-u_0}\mathbf{b}[u_0,u_1] + \frac{u-u_0}{u_2-u_0}\mathbf{b}[u_1,u_2]\right)\\ &+ \frac{u-u_1}{u_2-u_1}\left(\frac{u_3-u}{u_3-u_1}\mathbf{b}[u_1,u_2] + \frac{u-u_1}{u_3-u_1}\mathbf{b}[u_2,u_3]\right).\end{aligned}$$

This uses the identity

$$u = \frac{u_2-u}{u_2-u_1}u_1 + \frac{u-u_1}{u_2-u_1}u_2$$

for the first step and the two identities

$$u = \frac{u_2-u}{u_2-u_0}u_0 + \frac{u-u_0}{u_2-u_0}u_2$$

and

$$u = \frac{u_3-u}{u_3-u_1}u_1 + \frac{u-u_1}{u_3-u_1}u_3$$

for the second step. Note that we successively express u in terms of intervals of growing size.

Starting with the $\mathbf{b}[u_i, u_{i+1}]$ as input control points, we may rewrite this as:

$$\begin{array}{lll}\mathbf{b}[u_0,u_1] & & \\ \mathbf{b}[u_1,u_2] & \mathbf{b}[u_1,u] & \\ \mathbf{b}[u_2,u_3] & \mathbf{b}[u,u_2] & \mathbf{b}[u,u].\end{array}$$

This is a first instance of the de Boor generalization of the de Casteljau algorithm. See Example 8.1 for a detailed computation.

Figure 8.1 illustrates the algorithm, but using the knot sequence $u_0, u_1, u_2, u_3 = 0, 1, 3, 4$ and $u = 2.0$.

Example 8.1 **The de Boor algorithm for $n = 2$.**

Let $u_0, u_1, u_2, u_3 = 0, 2, 4, 6$. Let the control points be given by

$$\mathbf{b}[u_0,u_1] = \begin{bmatrix}0\\0\end{bmatrix}, \quad \mathbf{b}[u_1,u_2] = \begin{bmatrix}8\\8\end{bmatrix}, \quad \mathbf{b}[u_2,u_3] = \begin{bmatrix}8\\0\end{bmatrix}.$$

Setting $u = 3$, we now compute the point $\mathbf{b}[3,3]$. At the first level, we compute two points

$$\mathbf{b}[2,3] = \frac{4-3}{4-0}\begin{bmatrix}0\\0\end{bmatrix} + \frac{3-0}{4-0}\begin{bmatrix}8\\8\end{bmatrix} = \begin{bmatrix}6\\6\end{bmatrix}$$

and

$$\mathbf{b}[3,4] = \frac{6-3}{6-2}\begin{bmatrix}8\\8\end{bmatrix} + \frac{3-2}{6-2}\begin{bmatrix}8\\0\end{bmatrix} = \begin{bmatrix}8\\6\end{bmatrix}.$$

Finally,

$$\mathbf{b}[3,3] = \frac{4-3}{4-2}\begin{bmatrix}6\\6\end{bmatrix} + \frac{3-2}{4-2}\begin{bmatrix}8\\6\end{bmatrix} = \begin{bmatrix}7\\6\end{bmatrix}.$$

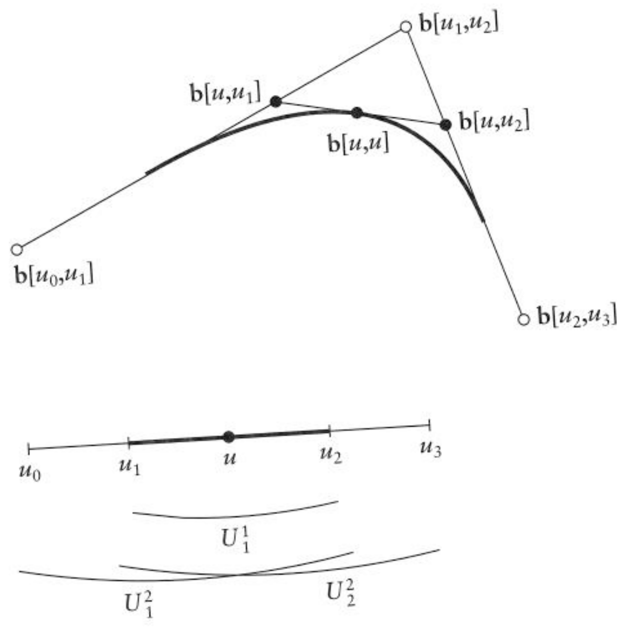


Figure 8.1 The de Boor algorithm: the quadratic case.

8.2 B-Spline Segments

B-spline curves consist of a sequence of polynomial curve segments. In this section, we focus on just one of them.

Let U be an interval $[u_I, u_{I+1}]$ in a sequence $\{u_i\}$ of knots. We define ordered sets U_i^r of successive knots, each containing u_I or u_{I+1}. The set U_i^r is defined such that:

U_i^r consists of $r+1$ successive knots.

u_I is the $(r-i)$th element of U_i^r, with $i=0$ denoting the first of U_i^r's elements.

We also observe

$$U_i^r = U_i^{r+1} \cap U_{i+1}^{r+1}.$$

When the context is unambiguous, we also refer to the U_i^r as *intervals*, having the first and last elements of U_i^r as endpoints. In that context, $U_1^1 = U$. We also define $U = [U_0^0, U_1^0]$ and use the term *interval* only if $U_0^0 \neq U_1^0$.

A degree n curve segment corresponding to the interval U is given by $n+1$ control points $\mathbf{d}_i$ which are defined by

$$\mathbf{d}_i = \mathbf{b}[U_i^{n-1}]; \quad i = 0, \ldots, n. \tag{8.1}$$

The point $\mathbf{x}(u) = \mathbf{b}[u^{<n>}]$ on the curve is recursively computed as

$$\mathbf{d}_i^r(u) = \mathbf{b}[u^{<r>}, U_i^{n-1-r}]; \quad r = 1, \ldots, n; i = 0, \ldots, n-r \tag{8.2}$$

with $\mathbf{x}(u) = \mathbf{d}_0^n(u)$.[2] This is known as the *de Boor algorithm* after Carl de Boor see [137]. See Example 8.2 for the case $n=3$ and Figure 8.2 for an illustration.

Equation (8.2) may alternatively be written as

$$\mathbf{d}_i^r(u) = (1 - t_{i+1}^{n-r+1})\mathbf{d}_i^{r-1} + t_{i+1}^{n-r+1}\mathbf{d}_{i+1}^{r-1}; \quad r = 1, \ldots, n; i = 0, \ldots, n-r, \tag{8.3}$$

where t_{i+1}^{n-r+1} is the local parameter in the interval U_{i+1}^{n-r+1}.

A geometric interpretation is as follows. Each intermediate control polygon leg $\mathbf{d}_i^r, \mathbf{d}_{i+1}^r$ may be viewed as an affine image of U_{i+1}^{n-r+1}. The point $\mathbf{d}_i^{r+1}$ is then the image of u under that affine map.

For the special knot sequence $0^{<n>}, 1^{<n>}$ and $U = [0,1]$, the de Boor algorithm becomes

$$\mathbf{d}_i^r(u) = \mathbf{b}[u^{<r>}, 0^{<n-r-i>}, 1^{<i>}]; \quad r = 1, \ldots, n; \quad i = 0, \ldots, n-r, \tag{8.4}$$

which is simply the de Casteljau algorithm.

If the parameter u happens to be one of the knots, the algorithm proceeds as before, except that we do not need as many levels of the algorithm. For example, if a quadratic curve segment is defined by $\mathbf{b}[u_0,u_1], \mathbf{b}[u_1,u_2], \mathbf{b}[u_2,u_3]$ and we want to evaluate at $u = u_2$, then two of the intermediate points in the de Boor algorithm are already known, namely, $\mathbf{b}[u_1,u_2]$ and $\mathbf{b}[u_2,u_3]$. From these two, we immediately calculate the desired point $\mathbf{b}[u_2,u_2]$, thus the de Boor algorithm now needs only one level instead of two.

Derivatives of a B-spline curve segment are computed in analogy to the Bézier curve case (5.17)

$$\dot{\mathbf{x}}(u) = n\mathbf{b}[u^{<n-1>}, \vec{1}]. \tag{8.5}$$

Expanding this expression and using the control point notation, this becomes

$$\dot{\mathbf{x}}(u) = \frac{n}{|U|}(\mathbf{d}_1^{n-1} - \mathbf{d}_0^{n-1}), \tag{8.6}$$

2 This notation is different from the one used in previous editions of this book.

Example 8.2 The de Boor algorithm for $n = 3$.

Let part of a knot sequence be given by

$$\ldots u_3, u_4, u_5, u_6, u_7, u_8, \ldots$$

and let $U = [u_5, u_6]$. The standard blossom computation of $\mathbf{b}[u, u, u]$ proceeds as follows:

$$\begin{array}{llll}
\mathbf{b}[u_3, u_4, u_5] & & & \\
\mathbf{b}[u_4, u_5, u_6] & \mathbf{b}[u, u_4, u_5] & & \\
\mathbf{b}[u_5, u_6, u_7] & \mathbf{b}[u, u_5, u_6] & \mathbf{b}[u, u, u_5] & \\
\mathbf{b}[u_6, u_7, u_8] & \mathbf{b}[u, u_6, u_7] & \mathbf{b}[u, u, u_6] & \mathbf{b}[u, u, u].
\end{array}$$

We now write this as

$$\begin{array}{llll}
\mathbf{b}[U_0^2] & & & \\
\mathbf{b}[U_1^2] & \mathbf{b}[u, U_0^1] & & \\
\mathbf{b}[U_2^2] & \mathbf{b}[u, U_1^1] & \mathbf{b}[u, u, U_0^0] & \\
\mathbf{b}[U_3^2] & \mathbf{b}[u, U_2^1] & \mathbf{b}[u, u, U_1^0] & \mathbf{b}[u, u, u].
\end{array}$$

In terms of control points:

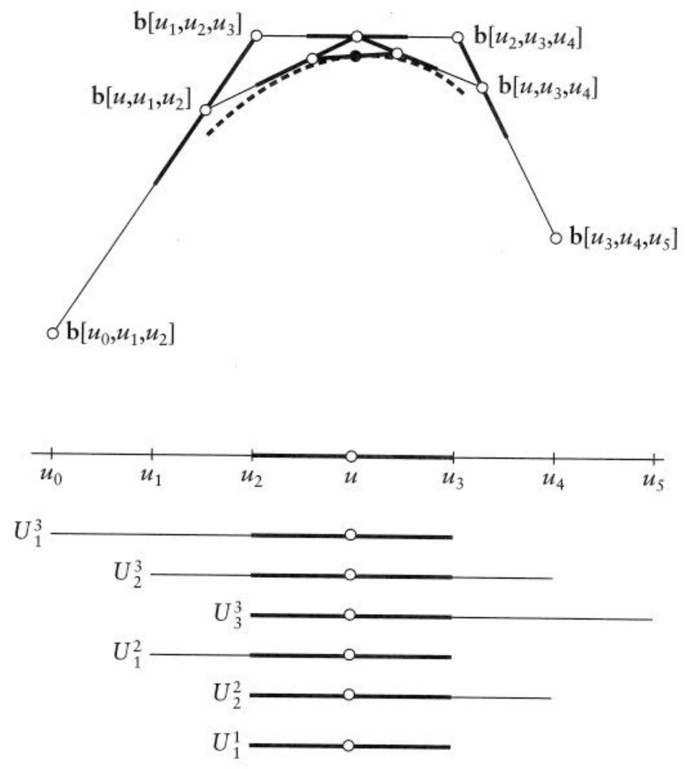
$$\begin{array}{llll}
\mathbf{d}_0 & & & \\
\mathbf{d}_1 & \mathbf{d}_0^1 & & \\
\mathbf{d}_2 & \mathbf{d}_1^1 & \mathbf{d}_0^2 & \\
\mathbf{d}_3 & \mathbf{d}_2^1 & \mathbf{d}_1^2 & \mathbf{d}_0^3.
\end{array}$$

The labeling in the first scheme depends on the subscripts of the knots, whereas the last two employ a relative numbering.

where $|U| = U_1^0 - U_0^0$ denotes the length of the interval U. Thus the last two intermediate points $\mathbf{d}_0^{n-1}$ and $\mathbf{d}_1^{n-1}$ span the curve's tangent, in complete analogy to the de Casteljau algorithm.

Higher derivatives follow the same pattern:

$$\frac{d^r}{du^r}\mathbf{x}(u) = \frac{n!}{(n-r)!}\mathbf{b}[u^{<n-r>}, \vec{1}^{<r>}]. \tag{8.7}$$



$\mathbf{b}[u_1,u_2,u_3]$
$\mathbf{b}[u_2,u_3,u_4]$
$\mathbf{b}[u,u_1,u_2]$
$\mathbf{b}[u,u_3,u_4]$
$\mathbf{b}[u_3,u_4,u_5]$
$\mathbf{b}[u_0,u_1,u_2]$
u_0 u_1 u_2 u u_3 u_4 u_5
U_1^3
U_2^3
U_3^3
U_1^2
U_2^2
U_1^1

Figure 8.2 The de Boor algorithm: a cubic example. The solid point is the result $\mathbf{b}[u, u, u]$; it is on the line through $\mathbf{b}[u, u, u_2]$ and $\mathbf{b}[u, u, u_3]$.

In the case of Bézier curves, we could use the de Casteljau algorithm for curve evaluation, but we could also write a Bézier curve explicitly using Bernstein polynomials. Since we changed the domain geometry, we will now obtain a different explicit representation, using polynomials[3] P_i^n:

$$\mathbf{x}(u) = \sum_{i=0}^{n} \mathbf{d}_i P_i^n(u). \tag{8.8}$$

The polynomials P_i^n satisfy a recursion similar to the one for Bernstein polynomials, and the following derivation is very similar to that case:

3 These will later become building blocks of B-splines.

$$
\begin{aligned}
\mathbf{x}(u) &= \sum_{i=0}^{n-1} \mathbf{d}_i^1 P_i^{n-1}(u) \\
&= \sum_{i=0}^{n-1} (1 - t_{i+1}^n)\mathbf{d}_i P_i^{n-1}(u) + \sum_{i=0}^{n-1} t_{i+1}^n \mathbf{d}_{i+1} P_i^{n-1}(u) \\
&= \sum_{i=0}^{n} (1 - t_{i+1}^n)\mathbf{d}_i P_i^{n-1}(u) + \sum_{i=1}^{n} t_i^n \mathbf{d}_i P_{i-1}^{n-1}(u)
\end{aligned} \tag{8.9}
$$

For the first step, we used the de Boor algorithm, letting t_i^n be the local parameter in the interval U_{i+1}^n. For the second step, we used the convention $P_n^{n-1} \equiv 0$ to modify the first term. Using a similar argument: $P_{-1}^{n-1} \equiv 0$, we may extend the second term to start with $i = 0$. We conclude

$$P_i^n(u) = (1 - t_{i+1}^n)P_i^{n-1}(u) + t_i^n P_{i-1}^{n-1}(u). \tag{8.10}$$

This recursion has to be anchored in order to be useful. This is straightforward for the case $n = 1$:

$$P_0^1(u) = \frac{u_r - U_1^0}{|U|}, \quad P_1^1(u) = \frac{u - U_0^0}{|U|},$$

where $|U| = U_1^0 - U_0^0$. For the special knot sequence $0^{<n>}, 1^{<n>}$ and $U = [0, 1]$, this is the Bernstein recursion.

8.3 B-Spline Curves

A B-spline curve consists of several polynomial curve segments, each of which may be evaluated using a de Boor algorithm. A B-spline curve is defined by

- the degree n of each curve segment,
- the knot sequence $u_0, \ldots, u_K$, consisting of $K + 1$ knots $u_i \le u_{i+1}$,
- the control polygon $\mathbf{d}_0, \ldots, \mathbf{d}_L$ with $L = K - n + 1$.

Some comments: the numbering of the control points $\mathbf{d}_i$ in this definition is *global*, whereas in Section 8.2 it was *local* relative to an interval U. Each $\mathbf{d}_i$ may be written as a blossom value with n subsequent knots as arguments. Hence the number $L + 1$ of control points equals the number of n-tuples in the knot sequence.

Example 8.3 **Some examples of B-spline curve definitions.**

Let $n = 1$, and let the knot sequence be $0, 1, 2$, hence $K = 2$. There will be control points $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_3$. The curve's domain is $[u_0, u_3]$, and there are two linear curve segments.
Let $n = 2$ with the knot sequence $0, 0.2, 0.4, 0.5, 0.7, 1$, hence $K = 5$. There will be control points $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4$ and three quadratic curve segments. If we now change the knot sequence to $0, 0.2, 0.45, 0.45, 0.7, 1$, then the number of curve segments will drop to two.

Each knot may be repeated in the knot sequence up to n times. In some cases it is approriate to simply list those knots multiple times. For other applications, it is better to list the knot only once and record its *multiplicity* in an integer array. For example, the knot sequence 0.0, 0.0, 1.0, 2.0, 3.0, 3.0, 4.0, 4.0 could be stored as 0.0, 1.0, 2.0, 3.0, 4.0 and a multiplicity array 2, 1, 1, 2, 2.

There is a different de Boor algorithm for each curve segment. Each is "started" with a set of U_i^n, that is, by sequences of $n + 1$ knots. In order for local coordinates to be defined in (8.3), no successive $n + 1$ knots may coincide.

In Section 8.2, we assumed that we could find the requisite U_i^n for each interval U. This is possible only if U is "in the middle" of the knot sequence; more precisely, the first possible de Boor algorithm is defined for $U = [u_{n-1}, u_n]$ and the last one is defined for $U = [u_{K-n}, u_{K-n+1}] = [u_{K-n}, u_L]$. We thus call $[u_{n-1}, u_L]$ the *domain* of the B-spline curve. A B-spline curve has as many curve segments as there are nonzero intervals U in the domain. Example 8.3 illustrates these comments.

For more examples of B-spline curves, see Figure 8.3.

Since a B-spline curve consists of a number of polynomial segments, one might ask for the Bézier form of these segments. For a segment $U = [u_I, u_{I+1}]$ of the curve, we simply evaluate its blossom $\mathbf{b}^U$ and obtain the Bézier points $\mathbf{b}_0^U, \ldots, \mathbf{b}_n^U$ as

$$\mathbf{b}_k^U = \mathbf{b}^U[u_I^{<n-k>}, u_{I+1}^{<k>}].$$

An example is shown in Figure 8.4. Several constituent curve pieces of the same curve are shown in Figure 8.5.

When dealing with B-spline curves, it is convenient to treat it as one curve, not just as a collection of polynomial segments. A point on such a curve is denoted by $\mathbf{d}(u)$, with $u \in [u_{n-1}, u_{K-n+1}]$. In order to evaluate, we perform the following steps:

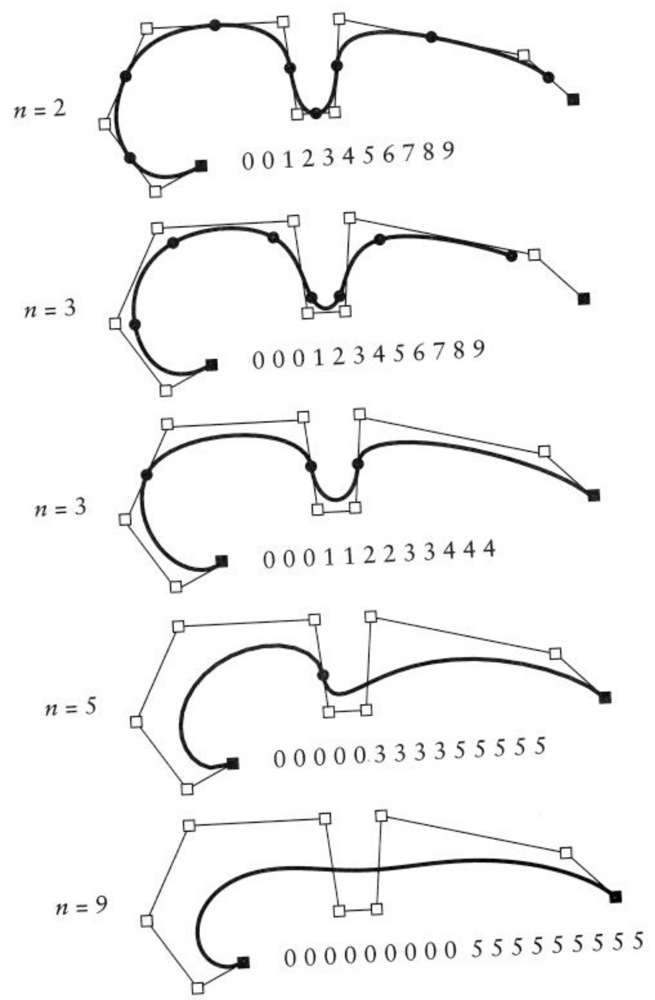


Figure 8.3 B-spline curves: several examples.

1. Find the interval $U = [u_I, u_{I+1})$ that contains u.
2. Find the $n+1$ control points that are relevant for the interval U. They are, using the global numbering, given by $\mathbf{d}_{I-n+1}, \ldots, \mathbf{d}_{I+1}$.
3. Renumber them as $\mathbf{d}_0, \ldots, \mathbf{d}_n$ and evaluate using the de Boor algorithm (8.3).

In terms of the global numbering of knots, we observe that the intervals U_i^k from the previous section are given by

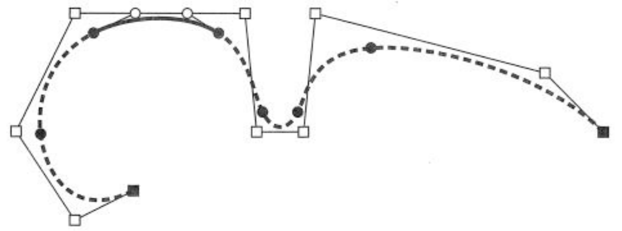


Figure 8.4 Conversion to Bézier form: the Bézier points of a segment of a cubic B-spline curve.

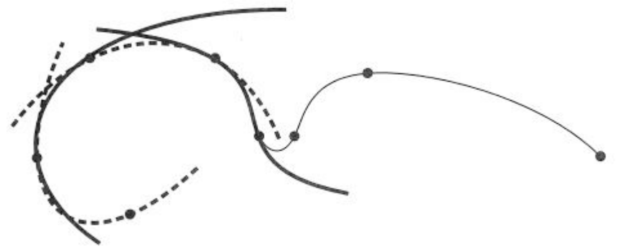


Figure 8.5 Individual curve segments: the first four cubic segments of a cubic B-spline curve are shown, alternating between dashed and black.

$$U_i^k = [u_{I-k+i}, u_{I+i}].$$

The steps in the de Boor algorithm then become

$$\mathbf{d}_i^k(u) = (1-\alpha_i^k)\mathbf{d}_i^{k-1}(u) + \alpha_i^k \mathbf{d}_{i+1}^{k-1}(u)$$

with

$$\alpha_i^k = \frac{u - u_{I-k+i}}{u_{I+i} - u_{I\_k+i}}$$

for $k = r+1, \ldots, n$, and $i = 0, \ldots, n-k$. Here, r denotes the multiplicity of u. (Normally, u is not already in the knot sequence; then, $r = 0$.)

The fact that each curve segment is only affected by $n+1$ control points is called the *local control property*.

We also use the notion of a B-spline blossom $\mathbf{d}[v_1, \ldots, v_n]$, keeping in mind that each domain interval U has its own blossom $\mathbf{b}^U$ and that consequently $\mathbf{d}[v_1, \ldots, v_n]$ is piecewise defined.

B-spline curves enjoy all properties of Bézier curves, such as affine invariance, variation diminution, etc. Some of these properties are more pronounced now because of the local control property. Take the example of the convex hull

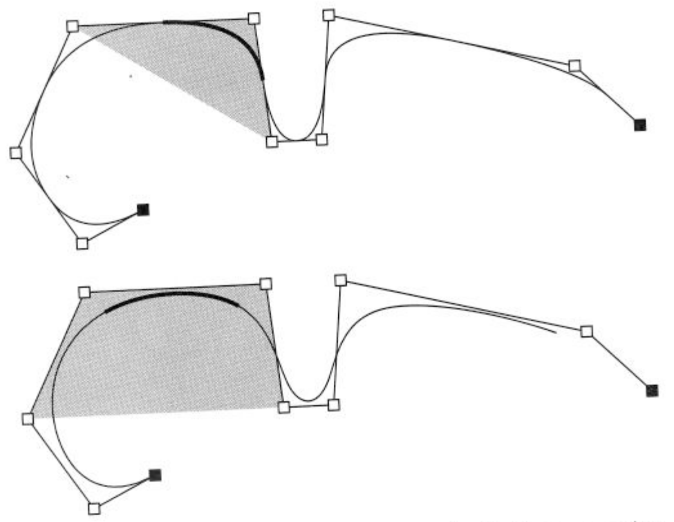


Figure 8.6 The local convex hull property: top, quadratic; bottom, cubic.

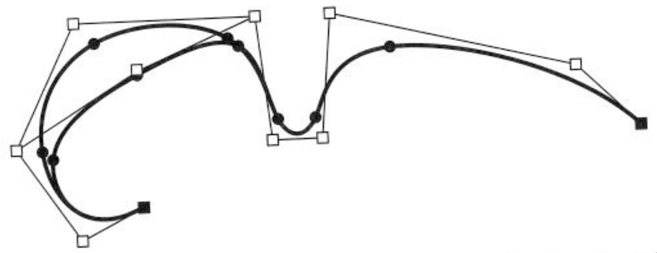


Figure 8.7 The local control property: changing one control point of a cubic B-spline curve has only a local effect.

property: the curve is in the convex hull of its control polygon, but also each segment is in the convex hull of its defining $n+1$ control points; see Figure 8.6.

A consequence of the local control property is that changing one control point will only affect the curve locally. This is illustrated in Figure 8.7.

8.4 **Knot Insertion**

Consider a B-spline curve segment defined over an interval U. It is defined by all blossom values $\mathbf{d}[U_i^{n-1}]$; $i=0,\ldots,n$ where each n-tuple of successive knots U_i^{n-1} contains at least one of the endpoints of U. If we now split U into two segments by inserting a new knot $\hat{u}$, the curve will have two corresponding

Example 8.4 **Knot insertion.**

In Figure 8.8, just one de Boor step is carried out for the parameter value $\hat{u}$. The two new resulting points, in blossom notation, are $\mathbf{b}[u_1,\hat{u}]$ and $\mathbf{b}[\hat{u},u_2]$. Let us now consider the points

$$\mathbf{b}[u_0,u_1], \mathbf{b}[u_1,\hat{u}], \mathbf{b}[\hat{u},u_2].$$

These are the B-spline control points of our curve $\mathbf{b}[u,u]$ for the interval $[u_1,\hat{u}]$! Similarly, the B-spline control points for the interval $[\hat{u},u_2]$ are given by

$$\mathbf{b}[u_1,\hat{u}], \mathbf{b}[\hat{u},u_2], \mathbf{b}[u_2,u_3].$$

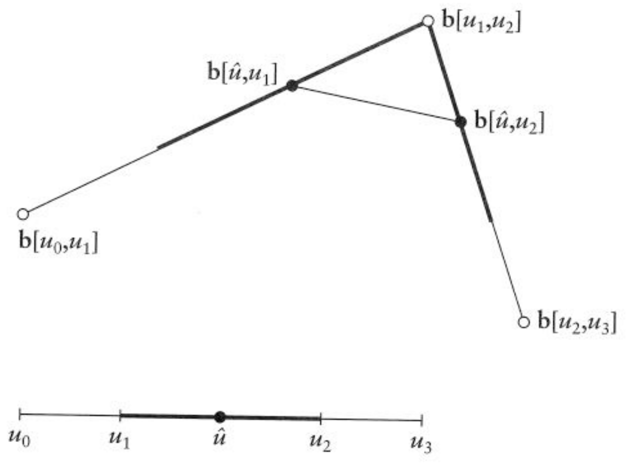


Figure 8.8 Knot insertion: a quadratic example.

segments. What are the control points for these two segments? The answer is surprisingly simple: all blossom values $\mathbf{d}[\hat{U}_i^{n-1}]$; $i=0,\ldots,n+1$ where each n-tuple of successive knots $\hat{U}_i^{n-i}$ contains at least one of the endpoints of U. This result is due to W. Boehm [68], although it was not originally derived using blossoms. See Example 8.4.

Knot insertion works since B-spline control points are nothing but blossom values of successive knots—now they involve the new knot $\hat{u}$. We may also view the process of knot insertion as one level of the de Boor algorithm, as illustrated in Example 8.4.

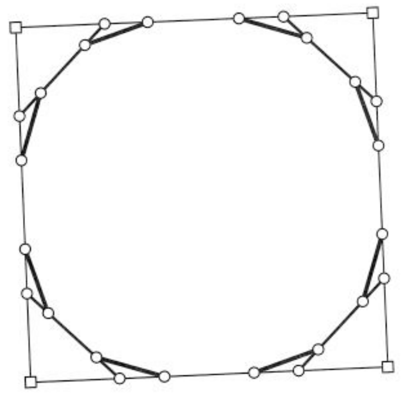


Figure 8.9 Chaikin's algorithm: starting with a (closed) control polygon B-spline curve, two levels of the algorithm are shown.

An interesting application of repeated knot insertion is due to G. Chaikin [105]. Consider a quadratic B-spline curve over a uniform knot sequence. Insert a new knot at the midpoint of every interval of the knot sequence. If the "old" curve had control vertices $\mathbf{d}_i$ and those of the new one are $\mathbf{d}_i^{(1)}$, it is easy to show that

$$\mathbf{d}_{2i-1}^{(1)} = \frac{3}{4}\mathbf{d}_i + \frac{1}{4}\mathbf{d}_{i-1} \quad \text{and} \quad \mathbf{d}_{2i}^{(1)} = \frac{3}{4}\mathbf{d}_i + \frac{1}{4}\mathbf{d}_{i+1}.$$

If this procedure is repeated infinitely often, the resulting sequence of polygons will converge to the original curve, as follows from our previous considerations. Figure 8.9 shows the example of a closed quadratic B-spline curve; two levels of the iteration are shown.

Chaikin's algorithm may be described as *corner cutting*: at each step, we chisel away the corners of the initial polygon. This process is, on a high level, similar to that of degree elevation for Bézier curves, which is also a convergent process. One may ask if corner-cutting processes will always converge to a smooth curve. The answer is *yes*, with some mild stipulations on the corner-cutting process, and was first proved by de Boor [140]. One may thus use a corner-cutting procedure to *define* a curve—and only very few of the curves thus generated are piecewise polynomial! Recent work has been carried out by Prautzsch and Micchelli [495] and [426], based on earlier results by de Rham [150], [151].

R. Riesenfeld [508] realized that Chaikin's algorithm actually generates uniform quadratic B-spline curves. A general algorithm for the simultaneous insertion of several knots into a B-spline curve has been developed by Cohen, Lyche, and Riesenfeld [121]. This so-called Oslo algorithm needs a theory of discrete B-splines for its development (see Bartels, Beatty, and Barsky [47]).

8.5 Degree Elevation

We may degree elevate in (almost) the same way we could degree elevate Bézier curves using (6.2). The difference: a given B-spline is a piecewise degree n curve over a given knot sequence. Its differentiability is determined by the knot multiplicities. If we write it as a piecewise degree $n+1$ curve, we need to increase the multiplicity of every knot by one, thus maintaining the original differentiability properties. For example, if we degree elevate a C^0 piecewise linear curve to piecewise quadratic, it is still C^0. But for a piecewise quadratic to be C^0, it has to have double knots. Let us denote the knots in this augmented knot sequence by $\hat{u}_i$.

Let V^n be a sequence of $n+1$ real numbers $v_1, \ldots, v_{n+1}$. Let $V^n|v_i$ denote the sequence V^n with the value v_i removed. Then the degree $n+1$ blossom $\hat{\mathbf{b}}$ may be expressed in terms of the degree n blossom $\mathbf{b}$ via

$$\hat{\mathbf{b}}[V^{(n+1)}] = \frac{1}{n+1}\left(\mathbf{b}[V^{(n+1)}|v_1] + \ldots + \mathbf{b}[V^{(n+1)}|v_{n+1}]\right). \tag{8.11}$$

The proof is identical to that for degree elevation of Bézier curves. The control points are then recovered from the blossom as before (see Example 8.5).

The inverse process—*degree reduction* is more important for practical applications. Following the example of the analogous Bézier case, we write the elevation process as a matrix product and invert it by a least squares technique for the reduction process; see Section 6.4. This method is described in detail in [617]. Other methods exist, see [481] and [624].

Example 8.5 B-spline degree elevation and blossoms.

Let a cubic B-spline curve be defined over $\{u_0 = u_1 = u_2, u_3, \ldots\}$. Then the interval $[u_4, u_5]$ corresponds to $[\hat{u}_7, \hat{u}_8]$. We denote the corresponding blossoms by $\mathbf{d}_4[a, b, c]$ and $\mathbf{d}_7[a, b, c, d]$. The new control point $\hat{\mathbf{d}}_4$ is computed as follows:

$$\begin{aligned}
\hat{\mathbf{d}}_4 &= \hat{\mathbf{d}}_7[\hat{u}_4, \hat{u}_5, \hat{u}_6, \hat{u}_7] \\
&= \frac{1}{4}\left(\mathbf{d}_4[\hat{u}_4, \hat{u}_5, \hat{u}_6] + \mathbf{d}_4[\hat{u}_4, \hat{u}_5, \hat{u}_7] + \mathbf{d}_4[\hat{u}_4, \hat{u}_6, \hat{u}_7] + \mathbf{d}_4[\hat{u}_5, \hat{u}_6, \hat{u}_7]\right) \\
&= \frac{1}{2}\left(\mathbf{d}_4[u_3, u_3, u_4] + \mathbf{d}_4[u_3, u_4, u_4]\right).
\end{aligned}$$

For the last step, we have used $\hat{u}_4 = \hat{u}_5 = u_3$ and $\hat{u}_6 = \hat{u}_7 = u_4$.

8.6 Greville Abscissae

Let $l[u]=u$ be the blossom of the (nonparametric) linear function u. If we want to write this linear blossom as a quadratic one: $l^2[u,v]=l[u]$, we easily see that

$$l^{(2)}[u,v]=\frac{1}{2}l[u]+\frac{1}{2}l[v]$$

gives the desired quadratic form of our linear blossom. If we asked for a cubic form $l^{(3)}[u,v,w]$ of $l[u]$, we find that

$$l^{(3)}[u,v,w]=\frac{1}{3}l^2[v,w]+\frac{1}{3}l^2[u,w]+\frac{1}{3}l^2[u,v].$$

If we denote a degree n version of the linear blossom by $l^{(n)}[V^n]$ with $V^n=v_1,\ldots,v_n$, it follows that

$$l^{(n)}[V^n]=\frac{1}{n}(v_1+\ldots+v_n).$$

The proof is by induction and was anchored by the earlier examples. The inductive step starts with the degree elevation formula (8.11):[4]

$$l^{(n+1)}=\frac{1}{n+1}\sum_{i=1}^{n+1}\frac{1}{n}[V^{(n+1)}|v_i]$$

This is easily transformed to

$$l^{n+1}=\frac{1}{n+1}(v_1+\ldots+v_{n+1}),$$

thus finishing the proof.

If we are given a knot sequence $u_0,\ldots,u_K$ and a degree n, then we know that any B-spline function $d(u)$ has control vertices $d[u_0,\ldots,u_{n-1}],\ldots,d[u_{K-n+1},\ldots,u_K]$. In the case of a linear function l, we thus have control vertices

$$\frac{1}{n}(u_0+\ldots+u_{n-1}),\ldots,\frac{1}{n}(u_{K-n+1}+\ldots+u_n).$$

4 We do not have to work with augmented knot sequences here since we always deal with *one* linear function.

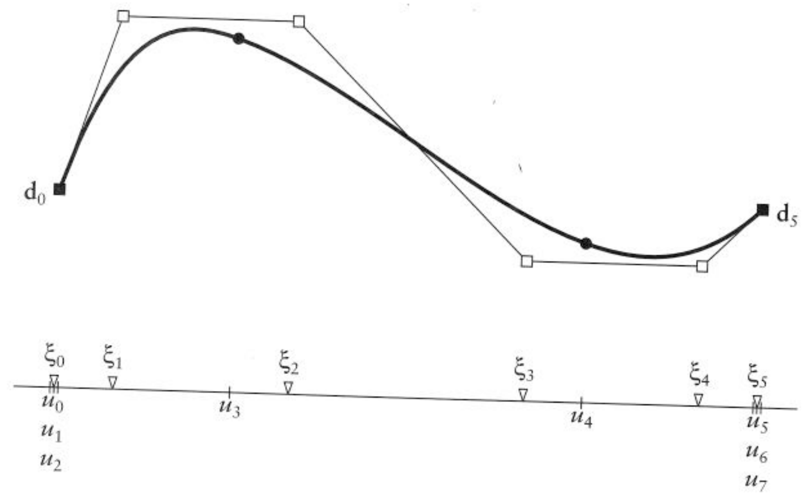


Figure 8.10 Nonparametric B-spline curves: a cubic example.

These terms are called *Greville abscissae* and are abbreviated as

$$\xi_i=\frac{1}{n}(u_i+\ldots+u_{i+n-1}).$$

A *nonparametric* B-spline function $d(u)$ may thus be written as a parametric curve with points

$$\mathbf{d}_i=\begin{bmatrix}\xi_i\\ d_i\end{bmatrix};\quad i=0,\ldots,L$$

with the usual $L=K-n+1$. Figure 8.10 gives an example.

For the special case of the knot sequence $0^{<n>},1^{<n>}$, we obtain $\xi_i=\frac{i}{n}$, as already encountered in Section 6.5.

8.7 Smoothness

A B-spline curve consists of several polynomial segments, one for each domain interval U. What is the smoothness of this piecewise curve?

Figures 8.11, 8.12, and 8.13 show how knot multiplicities affect smoothness. In general, if a knot $\hat{u}$ is of multiplicity r, then a B-spline curve of degree n has smoothness C^{n-r} at that knot. This follows from considering the osculants

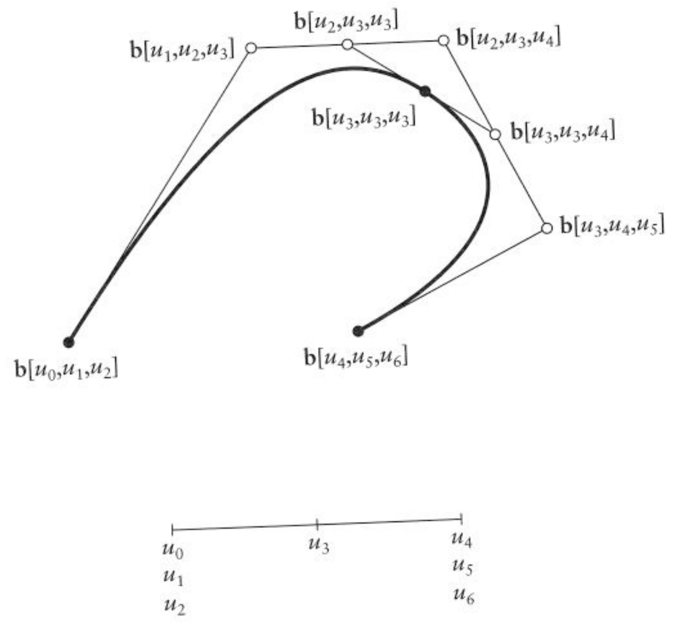

$\mathbf{b}[u_2,u_3,u_3]$
$\mathbf{b}[u_1,u_2,u_3]$
$\mathbf{b}[u_2,u_3,u_4]$
$\mathbf{b}[u_3,u_3,u_3]$
$\mathbf{b}[u_3,u_3,u_4]$
$\mathbf{b}[u_3,u_4,u_5]$
$\mathbf{b}[u_0,u_1,u_2]$
$\mathbf{b}[u_4,u_5,u_6]$
u_0 u_1 u_2
u_3
u_4 u_5 u_6

Figure 8.11 Smoothness: an interior knot of multiplicity one results in a C^2 piecewise cubic curve.

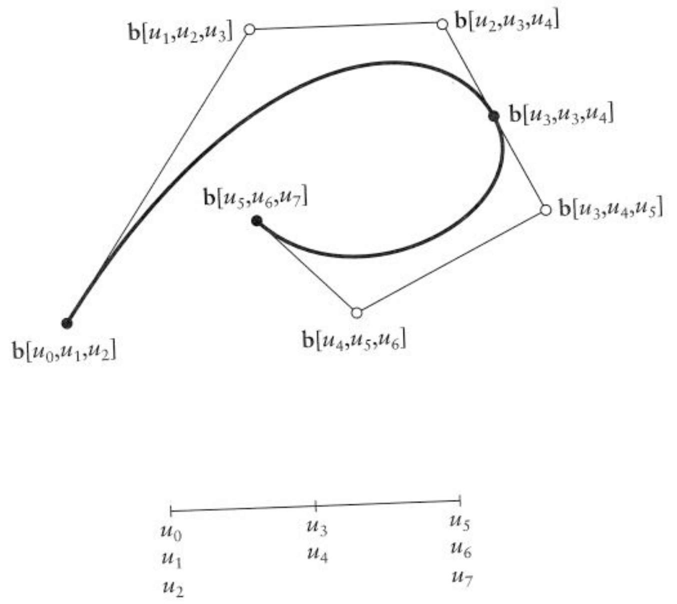

$\mathbf{b}[u_1,u_2,u_3]$
$\mathbf{b}[u_2,u_3,u_4]$
$\mathbf{b}[u_3,u_3,u_4]$
$\mathbf{b}[u_5,u_6,u_7]$
$\mathbf{b}[u_3,u_4,u_5]$
$\mathbf{b}[u_0,u_1,u_2]$
$\mathbf{b}[u_4,u_5,u_6]$
u_0 u_1 u_2
u_3 u_4
u_5 u_6 u_7

Figure 8.12 Smoothness: an interior knot of multiplicity two results in a C^1 piecewise cubic curve.

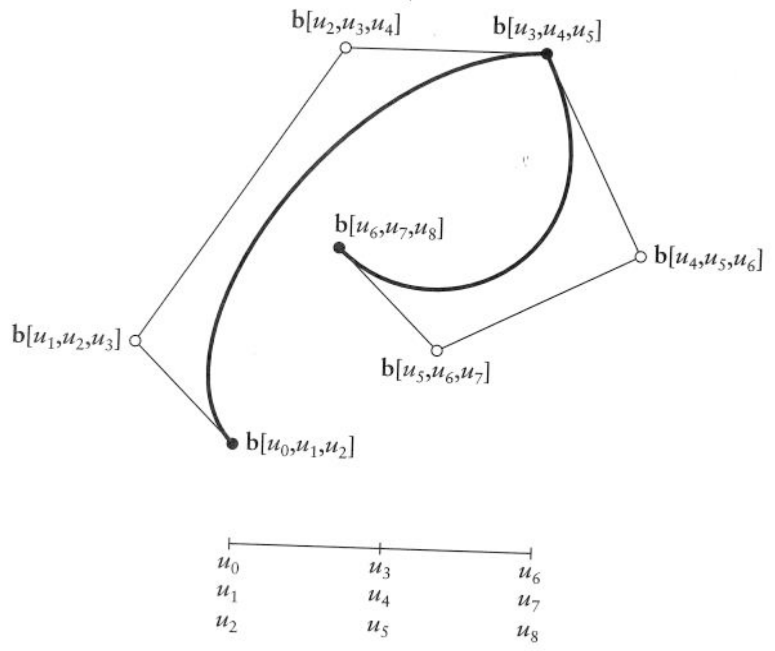

$\mathbf{b}[u_2,u_3,u_4]$
$\mathbf{b}[u_3,u_4,u_5]$
$\mathbf{b}[u_6,u_7,u_8]$
$\mathbf{b}[u_4,u_5,u_6]$
$\mathbf{b}[u_1,u_2,u_3]$
$\mathbf{b}[u_5,u_6,u_7]$
$\mathbf{b}[u_0,u_1,u_2]$
u_0 u_1 u_2
u_3 u_4 u_5
u_6 u_7 u_8

Figure 8.13 Smoothness: an interior knot of multiplicity three results in a C^0 piecewise cubic curve.

at $\hat{u}$.[5] The highest-order osculant is given by

$$\mathbf{o}^{n-r}(u) = \mathbf{b}[\hat{u}^{<r>}, u^{<n-r>}],$$

assuring continuity of derivatives up to order $n-r$. Higher-order continuity is possible, but cannot be guaranteed.

An important special case is given by *piecewise Bézier curves*. These are B-spline curves of degree n where each knot is of full multiplicity n. In general, such curves will only be C^0, but under certain conditions, they may be smoother.

For concreteness, take two cubic Bézier curves with control polygons $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ and $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_2$, defined over a knot sequence $u_0, u_0, u_0, u_1, u_1, u_1, u_2, u_2, u_2$. They are C^0 if $\mathbf{b}_3 = \mathbf{c}_0$, or, in terms of the associated blossoms, if $\mathbf{b}[u_1,u_1,u_1] = \mathbf{c}[u_1,u_1,u_1]$. Two such curves are shown in Figure 8.14.

The two curves are C^1 if they may be written as a B-spline curve with a double, not a triple knot u_1. Then our triple knot at u_1 is the result of knot insertion and the three points $\mathbf{b}_2, \mathbf{b}_3, \mathbf{c}_1$ are collinear and in the ratio $\Delta_0 : \Delta_1$ with $\Delta_0 = u_1 - u_0$

5 The osculant of order r of an n^{th} degree polynomial curve $\mathbf{x}(u)$ at paramter value $\hat{u}$ is the degree r polynomial that agrees with $\mathbf{x}$ for all derivatives up to order r.

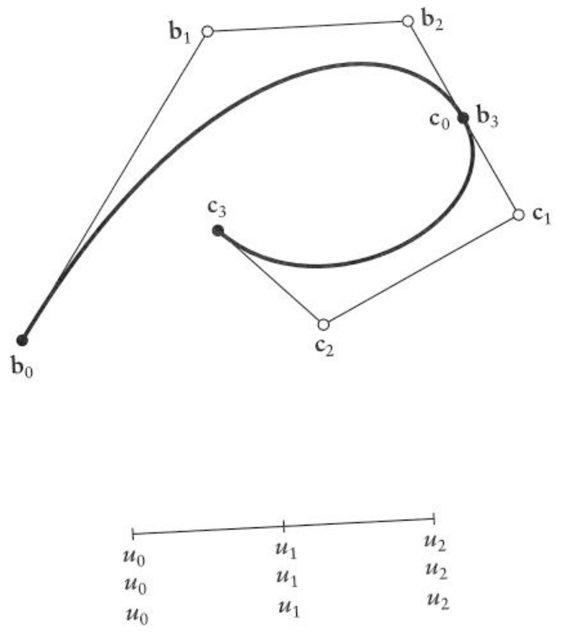


Figure 8.14 Smoothness of Bézier curves: the C^1 case.

and $\Delta_1 = u_2 - u_1$. In terms of blossoms:

$$c_1 = b[u_1, u_1, u_2] \quad \text{and} \quad b_2 = c[u_0, u_1, u_1].$$

For C^2 smoothness, the knot u_1 must have been the result of *two* knot insertions. It follows that

$$b[u_0, u_1, u_2] = c[u_0, u_1, u_2]. \tag{8.12}$$

is our desired C^2 condition. It is illustrated in Figure 8.15.

If we are to check if two given Bézier curves are C^2 or not, all we have to do is construct the two points appearing in (8.12). If they disagree, as in Figure 8.16, we conclude that the given curve is not C^2.

In most practical cases, a C^2 check would have to check for *approximate* satisfaction of (8.12), since reals or floats are rarely equal. In other words, a tolerance has to be used. The practical value of (8.12) lies in the fact that it is amenable to using a point tolerance that determines when two distinct points are to be considered the same point. Checking for C^2 smoothness by comparing second derivatives would require a different, and less intuitive, tolerance.

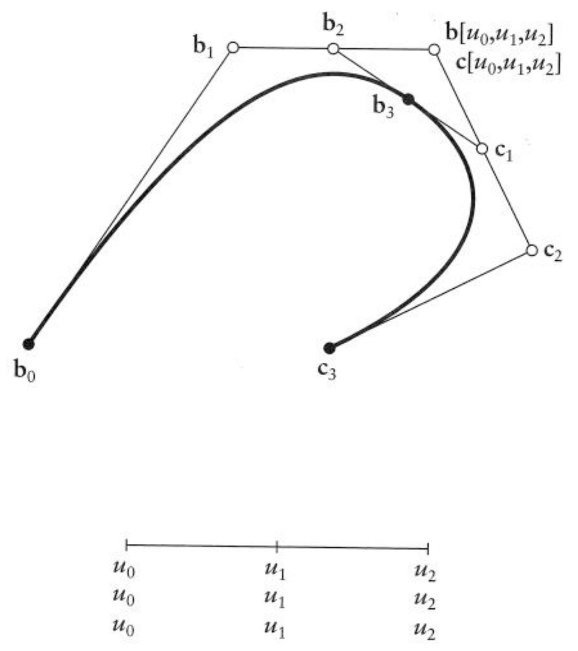


Figure 8.15 Smoothness of Bézier curves: the C^2 case.

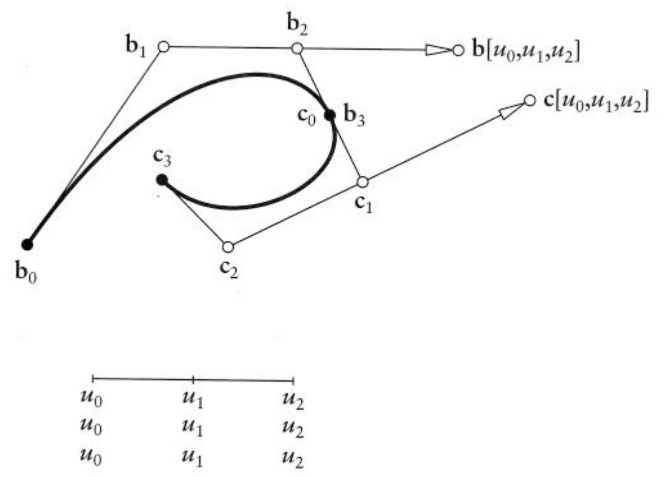


Figure 8.16 Smoothness of Bézier curves: the C^2 condition is violated.

8.8 B-Splines

Consider a knot sequence $u_0, \ldots, u_M$ and the set of piecewise polynomials of degree n defined over it, where each function in that set is $n - r_i$ times continuously differentiable at knot u_i. All these piecewise polynomials form a linear space, with dimension

$$\dim = (n+1) + \sum_{i=1}^{M-1} r_i. \tag{8.13}$$

For a proof, suppose we want to construct an element of our piecewise polynomial linear space. The number of independent constraints that we can impose on an arbitrary element, or its number of *degrees of freedom*, is equal to the dimension of the considered linear space. We may start by completely specifying the first polynomial segment, defined over $[u_0, u_1]$; we can do this in $n+1$ ways, which is the number of coefficients that we can specify for a polynomial of degree n. The next polynomial segment, defined over $[u_1, u_2]$, must agree with the first segment in position and $n - r_1$ derivatives at u_1, thus leaving only r_1 coefficients to be chosen for the second segment. Continuing further, we obtain (8.13).

We are interested in B-spline curves that are piecewise polynomials over the special knot sequence $[u_{n-1}, u_L]$. The dimension of the linear space that they form is $L+1$, which also happens to be the number of B-spline vertices for a curve in this space. If we can define $L+1$ linearly independent piecewise polynomials in our linear function space, we have found a basis for this space. We proceed as follows.

Define functions $N_i^n(u)$, called *B-splines* by defining their de Boor ordinates to satisfy $d_i = 1$ and $d_j = 0$ for all $j \neq i$. The $N_i^n(u)$ are clearly elements of the linear space formed by all piecewise polynomials over $[u_{n-1}, u_L]$. They have *local support*:

$$N_i^n(u) \neq 0 \text{ only if } u \in [u_{i-1}, u_{i+n}].$$

This follows because knot insertion, and hence the de Boor algorithm, is a local operation; if a new knot is inserted, only those Greville abscissae that are "close" will be affected.

B-splines also have *minimal support:* if a piecewise polynomial with the same smoothness properties over the same knot vector has less support than N_i^n, it must be the zero function. All piecewise polynomials defined over $[u_{i-1}, u_{i+n}]$, the support region of N_i^n, are elements of a function space of dimension $2n+1$, according to (8.13). A support region that is one interval "shorter" defines a function space of dimension $2n$. The requirement of vanishing $n - r_{i-1}$ derivatives at u_{i-1} and of vanishing $n - r_{i+n}$ derivatives at u_{i+n} imposes $2n$ conditions on any element in the linear space of functions over $[u_{i-1}, u_{i+n-1}]$. The additional requirement of assuming a nonzero value at some point in the support region raises the number of independent constraints to $2n+1$, too many to be satisfied by an element of the function space with dimension $2n$.

Another important property of the N_i^n is their *linear independence*. To demonstrate this independence, we must verify that

$$\sum_{j=0}^{L} c_j N_j^n(u) \equiv 0 \tag{8.14}$$

implies $c_j = 0$ for all j. It is sufficient to concentrate on one interval $[u_I, u_{I+1}]$ with $u_I < u_{I+1}$. Because of the local support property of B-splines, (8.14) reduces to

$$\sum_{j=I-n+1}^{I+1} c_j N_j^n(u) \equiv 0 \quad \text{for } u \in [u_I, u_{I+1}].$$

We have completed our proof if we can show that the linear space of piecewise polynomials defined over $[u_{I-n}, u_{I+n+1}]$ does not contain a nonzero element that vanishes over $[u_I, u_{I+1}]$. Such a piecewise polynomial cannot exist: it would have to be a nonzero local support function over $[u_{I+1}, u_{I+n+1}]$. The existence of such a function would contradict the fact that B-splines are of *minimal* local support.

Because the B-splines N_i^n are linearly independent, every piecewise polynomial s over $[u_{n-1}, u_L]$ may be written uniquely in the form

$$s(u) = \sum_{j=0}^{L} d_j N_j^n(u). \tag{8.15}$$

The B-splines thus form a *basis* for this space. This reveals the origin of their name, which is short for Basis splines. Figure 8.17 gives examples of some cubic B-splines.

If we set all $d_i = 1$ in (8.15), the function $s(u)$ will be identically equal to 1, thus asserting that B-splines form a *partition of unity*.

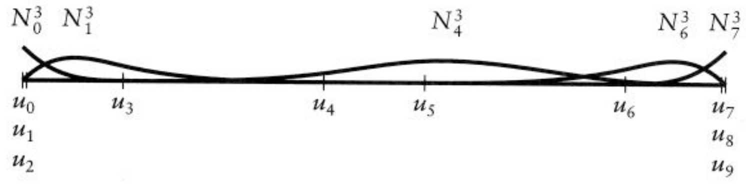


Figure 8.17 B-splines: some cubic examples.

B-spline curves are simply the parametric equivalent of (8.15):

$$\mathbf{x}(u) = \sum_{j=0}^{L} \mathbf{d}_j N_j^n(u).$$

Just as the de Casteljau algorithm for Bézier curves is related to the recursion of Bernstein polynomials, the de Boor algorithm yields a recursion for B-splines. It is given by

$$N_l^n(u) = \frac{u - u_{l-1}}{u_{l+n-1} - u_{l-1}} N_l^{n-1}(u) + \frac{u_{l+n} - u}{u_{l+n} - u_l} N_{l+1}^{n-1}(u), \qquad (8.16)$$

with the "anchor" for the recursion being given by

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}. \qquad (8.17)$$

Its proof relates the local recursion (8.10) to the global indexing scheme. An example is shown in Figure 8.18.

Equation (8.16) is due to L. Mansfield, C. de Boor, and M. Cox; see de Boor [137] and Cox [129]. For an illustration of (8.16), see Figure 8.18. This formula shows that a B-spline of degree n is a strictly convex combination of two lower-degree ones; it is therefore a very stable formula from a numerical viewpoint. If B-spline curves must be evaluated repeatedly at the same parameter values u_k, it is a good idea to compute the values for $N_i^n(u_k)$ using (8.16) and then to store them.

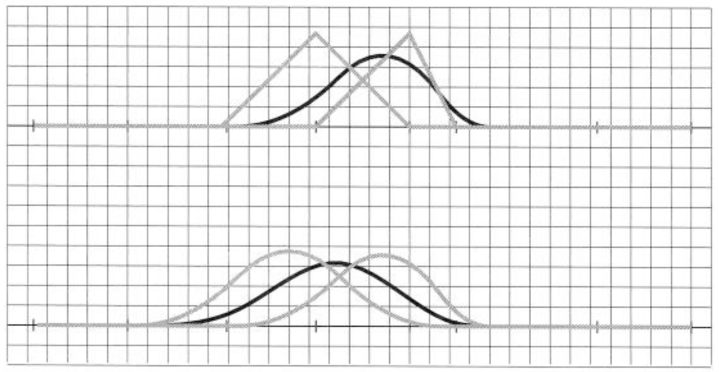


Figure 8.18 The B-spline recursion: top, two linear B-splines yield a quadratic one; bottom, two quadratic B-splines yield a cubic one.

A comment on end knot multiplicities: the widespread data format IGES uses two additional knots at the ends of the knot sequence; in our terms, it adds knots u_{-1} and u_{L+2n-1}. The reason is that formulas like (8.16) seemingly require the presence of these knots. Since they are multiplied only by zero factors, their values have no influence on any computation. There is no reason to store completely inconsequential data, and hence the "leaner" notation of this chapter.

8.9 B-Spline Basics

Here, we present a collection of the most important formulas and definitions of this chapter. As before, n is the (maximal) degree of each polynomial segment, $L+1$ is the number of control points, and K is the number of intervals.

Knot sequence: $\{u_0, \ldots, u_K\}$.

Control points: $\mathbf{d}_0, \ldots, \mathbf{d}_L$, with $L = K - n + 1$.

Domain: Curve is only defined over $[u_{n-1}, \ldots, u_L]$.

Greville abscissae: $\xi_i = \frac{1}{n}(u_i + \cdots + u_{i+n-1})$.

Support: N_i^n is nonnegative over $[u_{i-1}, u_{i+n}]$.

Knot insertion: To insert $u_I \le u < u_{I+1}$, first find new Greville abscissae $\hat{\xi}_i$, then set new $d_i = P(\hat{\xi}_i)$.

de Boor algorithm: Given $u_I \le u < u_{I+1}$, renumber the relevant control points $\mathbf{d}_{I-n+1}, \ldots, \mathbf{d}_{I+1}$ as $\mathbf{d}_0, \ldots, \mathbf{d}_n$ and then set

$$\mathbf{d}_i^k(u) = (1 - \alpha_i^k)\mathbf{d}_i^{k-1}(u) + \alpha_i^k \mathbf{d}_{i+1}^{k-1}(u)$$

with

$$\alpha_i^k = \frac{u - u_{I+i+1}}{u_{I-n+k+i} - u_{I+i+1}}$$

for $k = r+1, \ldots, n$, and $i = 0, \ldots, n-k$. Here, r denotes the multiplicity of u. (Normally, u is not already in the knot sequence; then, $r = 0$.)

Mansfield, de Boor, Cox recursion:

$$N_l^n(u) = \frac{u - u_{l-1}}{u_{l+n-1} - u_{l-1}} N_l^{n-1}(u) + \frac{u_{l+n} - u}{u_{l+n} - u_l} N_{l+1}^{n-1}(u).$$

Derivative:

$$\frac{\mathrm{d}}{\mathrm{d}u}\mathrm{N}_l^n(u) = \frac{n}{u_{n+l-1} - u_{l-1}}\mathrm{N}_l^{n-1}(u) - \frac{n}{u_{l+n} - u_l}\mathrm{N}_{l+1}^{n-1}(u).$$

Derivative of B-spline curve:

$$\frac{\mathrm{d}}{\mathrm{d}u}s(u) = n\sum_{i=0}^{L-1} \frac{\Delta \mathbf{d}_i}{u_{n+i-1} - u_{i-1}}\mathrm{N}_i^{n-1}(u).$$

Degree elevation:

$$\mathrm{N}_i^n(u) = \frac{1}{n+1}\sum_{j=i-1}^{n+i} \mathrm{N}_i^{n+1}(u; u_j),$$

where $\mathrm{N}_i^{n+1}(u; u_j)$ is defined over the original knot sequence except that the knot u_j has its multiplicity increased by one. This identity was discovered by H. Prautzsch in 1984 [493]. Another reference is Barry and Goldman [39].

8.10 Implementation

Here is the header for the de Boor algorithm code:

```
float deboor(degree,coeff,knot,u,i)
/*   uses de Boor algorithm to compute one
     coordinate on B-spline curve for param. value u in interval i.
Input: degree:          polynomial degree of each piece of curve
       coeff:           B-spline control points
       knot:            knot sequence
       u:               evaluation abscissa
       i:               u's interval: u[i]<= u < u[i+1]
Output:                 coordinate value.
*/
```

This program does not need to know about L. The next program generates a set of points on a whole B-spline curve—for one coordinate, to be honest—so it has to be called twice for a 2D curve and three times for a 3D curve.

```
bspl_to_points(degree,l,coeff,knot,dense,points,point_num)
/*    generates points on B-spline curve. (one coordinate)
Input: degree:          polynomial degree of each piece of curve
       l:               number of active  intervals
```

```
         coeff:         B-spline control points
         knot:          knot sequence: knot[0]...knot[l+2*degree-2]
         dense:         how many points per segment
Output:points:          output array with function values.
         point_num:     how many points are generated. That number is
                        easier computed here than in the calling program:
                        no points are generated between multiple knots.
*/
```

The main program `deboormain.c` generates a postscript plot of a B-spline curve. A sample input file is in `bspl.dat`; it creates the outline of the letter **r** from Figure 5.11.

As a second example, the input data for the y-values of the curve in Figure 8.10 are

```
degree = 3; l = 3; coeff = 1,4,4,0,0,1;
knot = 0,0,0,3,9,12,12,12; dense = 10.
```

Next, we include a B-spline blossom routine:

```
deboor_blossom(control,degree,deboor,deboor_wts,
              knot,uvec,interval,point,point_wt)

/*

 FUNCTION: deBoor algorithm to evaluate a B-spline curve blossom.
           For polynomial or rational curves.


 INPUT:    control[] .......... [0]: indicates type of input curve
                                     0 = polynomial
                                     1 = rational
                                [1]: indicates if input/output is
                                     in R3 or R4;
                                     3 = R3
                                     4 = R4
           degree ............. polynomial degree of each piece
                                of the input curve, must be <=20
           deboor[][3] ........ deboor control points
           deboor_wts[] ....... rational weights associated with
                                the control points if control[0]=1;
                                otherwise weights not used
```

```
                 knot[] ............. knot sequence with multiplicities
                                      entered explicitly
                 uvec[]  ............ blossom (parameter) values
                                      to evaluate
                 interval ........... interval within knot sequence
                                      with which to evaluate wrt u
                                      (typically: i=interval then
                                      knot[i]<= u < knot[i+1])

OUTPUT:          point[3] ........... evaluation point;
                                      depending on control[] values,
                                      this point will be in R3 or R4
                 point_wt ........... if control[0]=1 then this is the
                                      rational weight associated with
                                      the point

*/
```

8.11 Problems

1 For the case of a planar parametric B-spline curve, does symmetry of the polygon with respect to the y-axis imply that same symmetry for the curve?

***2** Derive (8.16) from (8.10).

***3** Find the Bézier points of the closed B-spline curves of degree four whose control polygons consist of the edges of a square and have (a) uniform knot spacing and simple knots and (b) uniform knot spacing and knots all with multiplicity two.

P1 Use `de_boor_blossom` to program degree elevation for B-spline curves.

9 Constructing Spline Curves

A *spline* is a flexible rod of wood or plastic. It has its origins in shipbuilding, where splines were used to draft the curves (ribs) that define a ship body. Early uses go back to the 1600s, and are documented in [450]. Although mechanical splines are used less frequently now, the underlying principle still gives rise to new algorithms.

9.1 Greville Interpolation

In Chapter 7, we saw how to pass a polynomial curve of degree n through $n+1$ data points $\mathbf{p}_0, \ldots, \mathbf{p}_n$ with parameter values $t_0, \ldots, t_n$. The key to the solvability of the problem was simple: the number of knowns (the data points with parameter values) had to equal the number of unknowns (the polynomial coefficients).

Something quite analogous happens in a spline context. A spline curve of degree n is defined over a knot sequence $u_0, \ldots, u_K$. Such a knot sequence has $K-n+2$ Greville abscissae ξ_i and hence the spline curve has $L+1 = K-n+2$ B-spline control points $\mathbf{d}_0, \ldots, \mathbf{d}_L$.

In view of these numbers, the following is a meaningful interpolation problem:

Given: A knot sequence $u_0, \ldots, u_K$ and a degree n, also a set of data points $\mathbf{p}_0 \cdots \mathbf{p}_L$ with $L = K - n + 1$.

Find: A set of B-spline control points $\mathbf{d}_0, \ldots, \mathbf{d}_L$ such that the resulting curve $\mathbf{x}(u)$ satisfies

$$\mathbf{x}(\xi_i) = \mathbf{p}_i; \quad i = 0, \ldots, L. \tag{9.1}$$