Example 7.1.

A parametric cubic curve passes through the points (0, 0), (2, 4), (4, 3), (5, –2) which are parametrized at $u = 0$, 1/4, 3/4, and 1, respectively. Determine the geometric coefficient matrix and the slope of the curve when $u = 0.5$.

Solution.

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 2 \\ 4 \\ 5 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.0156 & 0.0625 & 0.25 & 1 \\ 0.4218 & 0.5625 & 0.75 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} P_0 \\ P_1 \\ P_0' \\ P_1' \end{Bmatrix}$$

$$\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 4 \\ 3 \\ -2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.0156 & 0.0625 & 0.25 & 1 \\ 0.4218 & 0.5625 & 0.75 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} P_0 \\ P_1 \\ P_0' \\ P_1' \end{Bmatrix}$$

$$\begin{Bmatrix} P_0(x, y) \\ P_1(x, y) \\ P_0'(x, y) \\ P_1'(x, y) \end{Bmatrix} = \begin{Bmatrix} (0, 0) \\ (5, -2) \\ (10.33, 22) \\ (4.99, -26) \end{Bmatrix}$$

At $u = 0.5$

$$P'(x) = [0.5^3 \;\; 0.5^2 \;\; 0.5 \;\; 1] \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} 0 \\ 5 \\ 10.33 \\ 4.99 \end{Bmatrix} = 3.67$$

$$P'(y) = [0.5^3 \;\; 0.5^2 \;\; 0.5 \;\; 1] \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} 0 \\ -2 \\ 22 \\ -26 \end{Bmatrix} = -2.0$$

$$\text{Slope} = \frac{P'(y)}{P'(y)} = \frac{-2.0}{3.67} = -0.545$$

7.5.2. Bezier Curve

A Bezier curve is defined by approximating a set of data points (Figure 7.4). Given $n + 1$ points (called **control points**) $P_0, P_1, P_2, \ldots, P_n$ in space, the Bezier curve defined by these control points is:

$$\left.\begin{aligned} P(u) &= \sum_{i=0}^{n} B_{n,i}(u)\, P_i \\ 0 &\le u \le 1 \end{aligned}\right\} \qquad \ldots(7.15)$$

where the coefficients are defined as follows:

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i} \qquad ...(7.16)$$

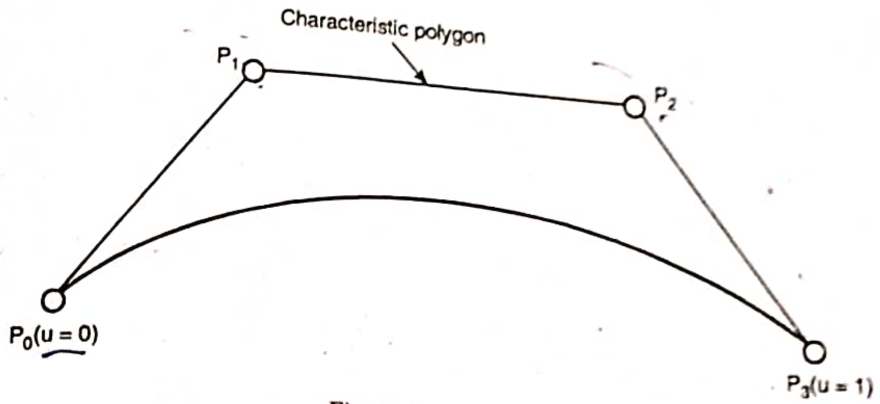


Fig. 7.9. Bezier curve

Therefore, the point that corresponds to u on the Bezier curve is the *weighted* average of all control points, where the weights are the coefficients $B_{n,i}(u)$. The line segments P_0P_1, P_1P_2, ..., P_{n-1}, P_n, called legs, joining in this order form a *characteristic polygon*. Functions $B_{n,i}(u)$, are referred to as the *Bezier basis functions* or *Bernstein polynomials.*

Note that the domain of u is [0, 1]. As a result, all basis functions are non-negative. In Eq. (7.16) since u and i can both be zero and so do 1 – and $n - i$, we adopt the convention that 00 is 1.

7.5.2.1. Properties of Bezier Curves

The following important characteristics of a Bezier curve are:

1. ***The degree of a Bezier curve defined by n + 1 control points is n***

   In each basis function, the exponent of u is $i + (n - i) = n$. Therefore, the degree of the curve is n.

2. ***P(u) passes through P_0 and P_n***

   This is shown in Fig. 7.4. The curve passes through the first and the last control point.

3. ***Non-negativity***

   All basis functions are non-negative.

4. ***Partition of unity***

   The sum of the basis functions at a fixed u is 1. It is not difficult to verify that the basis functions are the coefficients in the binomial expansion of the expression $1 = (u + (1 - u))n$. Hence, their sum is one. Moreover, since they are non-negative, we conclude that the value f any basis function is in the range of 0 and 1. Since all basis functions are in the range of 0 and 1 and sum to one, they can be considered as weights in the computation of a weighted average. More precisely, we could say to *compute P(u), one takes the weight $B_{n,i}(u)$ for control point P_i and sum them together.*

5. ***Convex hull property***

   This means the Bezier curve defined by the given $n + 1$ control points lies completely in the convext hull of the given control points. The convex hull of a set of points is the smallest convex set that contains all points. In the Figure 7.10, the convex hull of the 11 control points is shown. Note that not all control points are on the boundary of the convex hull. For example, control points 3, 4, 5, 8 and 9 are in the interior. The curve, except for the first two end points, lies completely in the convex hull.

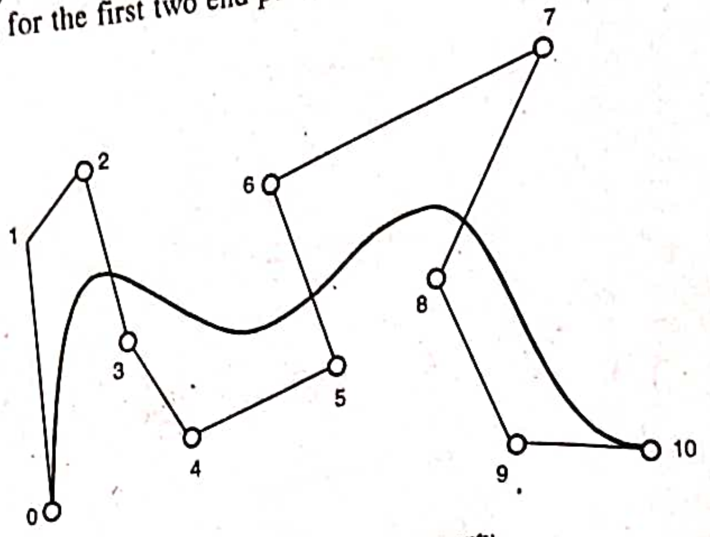


Fig. 7.10. Convex hull property

This property is important because we are guaranteed that the generated curve will be in an understood and computable region and will not go outside of it.

6. *Bezier curves are tangent to their first and last legs*

Differentiating (7.15) with respect to u, we get

$$\frac{d}{du}P(u) = P'(u) = \sum_{i=0}^{n-1} B_{n-1,i}(u)\left\{n(P_{i+1} - P_i)\right\} \quad ...(7.17)$$

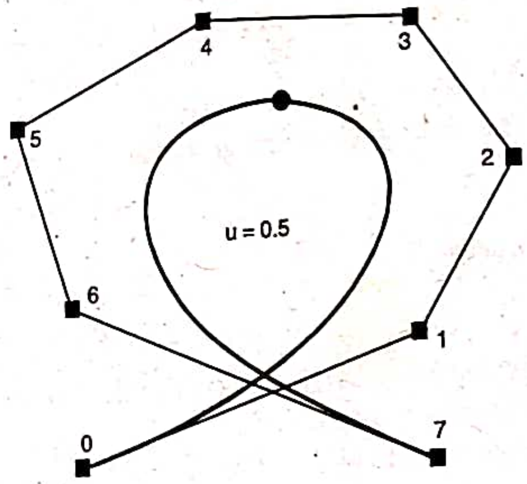


Fig. 7.11. Bezier curves are tangent to their first and last legs.

Letting $u = 0$ and $u = 1$ gives $P'(0) = n(P_1 - P_0)$ and $P'(1) = n(P_n - P_{n-1})$. The first means that the tangent vector at $u = 0$ is in the direction of $P_1 - P_0$ multiplied by n. Therefore, the first leg in the indicated direction is tangent to the Bezier curve. The second means that the tangent vector at $u = 1$ is in the direction of $P_n - P_{n-1}$ multiplier by n. Therefore, the last leg in the indicated direction is tangent to the Bezier curve. Figure 7.11 shows this proper well.

7. Finding a point on a Bezier curve: De Casteljau's Algorithm

Following the construction of a Bezier curve, the next important task is to find the point $P(u)$ on the curve for a particular u. A simple way is to plug u into every basis function, compute the product of each basis function and its corresponding ontrol point, and finally add them together. While this works fine, it is not numerically stable (*i.e.*, could introduce numerical errors during the course of evaluating the Bernstein polynomials).

In what follows, we shall only write down the control point numbers. That is, the control points are 00 for P_0, 01 for P_1, ..., 0*i* for P_i, ..., 0*n* for P_n. The 0s in these numbers indicate the initial or the 0th iteration. Later on, it will be replaced with 1, 2, 3 and so on.

The fundamental concept of de Casteljau's algorithm is to choose a point C in line segment AB such that C divides the line segment AB in a ratio of u:1 – u(*i.e.*, the ratio of the distance between A and C and the distance between A and B is u). Let us find a way to determine the point C (Fig. 7.12)

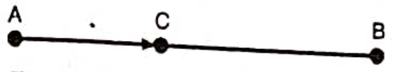


Fig. 7.12. Determining of point C.

The distance from A to B is $B - A$. Since u is a ratio in the range of 0 and 1, point C is located at $u(B - A)$. Taking the position of A into consideration, point C is $A + u(B - A) = (1 - u)A + uB$. Therefore, given a u, $(1 - u)A + uB$ is the point C between A and B that divides AB in a ratio of u:1 – u.

The idea of de Casteljau's algorithm goes as follows:

Suppose we want to find $P(u)$, where u is in [0, 1]. Starting with the first polyline 00, 01, 02, 03, ..., 0*n*, use the above formula to find a point 1i on the leg (*i.e.*, line segment) from 0*i* to 0(*i* + 1) that divides the line segment 0*i* and 0(*i* + 1) in a ratio of u:1 – u. In this way, we will obtain *n* points 10, 11, 12, ..., 1(*n* – 1). They define a new polyline of *n* – 1 legs.

In the Figure 7.13, u is 0.4, 10 is in the leg of 00 and 01, 11 is in the leg of 01 and 02, ..., and 14 is in the leg of 04 and 05. All of these new points are 10, 11, 12, 13 and 14. The new points are numbered as 1*i*'s. Apply the procedure to this new polyline and we shall get a second polyline of *n* – 1 points 20, 21, ..., 2(*n* – 2) and *n* – 2 legs. Starting with this polyline, we can construct a third one of *n* – 2 points 30, 31, ..., 3(*n* – 3) and *n* – 3 legs. Repeating this process *n* times yields a single point n0. De Casteljau provided that this is the point $P(u)$ on the curve that corresponds to u.

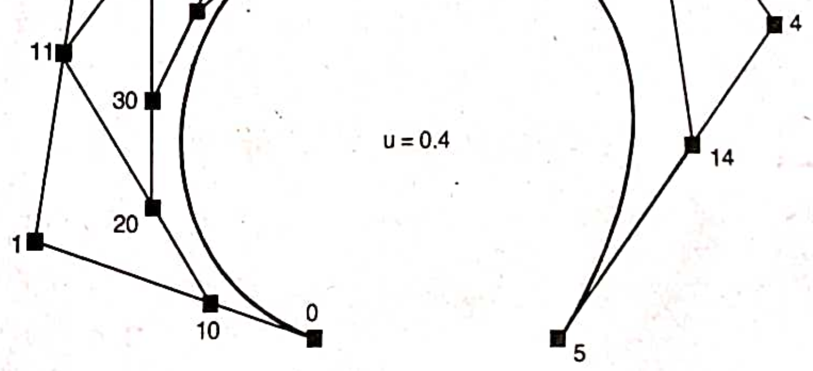


Fig. 7.13. Obtaining polylines.

Let us continue with the Figure 7.13. Let 20 be the point in the leg of 10 and 11 that divides the line segment 10 and 11 in a ratio of u: $1 - u$. Similarly, choose 21 on the leg of 11 and 12, 22 on the leg of 12 and 13 and 23 on the leg of 13 and 14. This gives a third polyline defined by 20, 21, 22 and 23. This third polyline has 4 points and 3 legs. Keep doing this and we shall obtain a new polyline of three points 30, 31 and 32. From this fourth polyline, we have the fifth one of two points 40 and 41. Do it once more, and we have 50, the point $P(0, 4)$ on the curve.

This is the geometric interpretation of de Casteljau's algorithm.

Example 7.2

Determine the cubic Bezier curve.

Solution. The cubic Bezier curve consists of 4 control points P_0, P_1, P_2 and P_3. The mathematical expression for the cubic Bezier curve is given by:

$$\left.\begin{aligned} P(u) &= \sum_{i=0}^{3} B_{3,i}(u)\, P_i \\ 0 &\le u \le 1 \end{aligned}\right\} \qquad \text{...(7.18)}$$

Equation (7.18) can be expanded to give:

$$\left.\begin{aligned} P(u) = P_0(1-u)^3 + 3P_1u(1-u)^2 + 3P_2u^2(1-u) + P_3u^3 \\ 0 \le u \le 1 \end{aligned}\right\} \qquad \text{...(7.19)}$$

In the matrix form

$$P(u) = [u^3 \;\; u^2 \;\; u \;\; 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix} \qquad \text{...(7.20)}$$

Example 7.3

A cubic Bezier curve is described by the four control points: (0, 0), (2, 1), (5, 2), (6, 1). Find the tangent to the curve at $t = 0.25$.

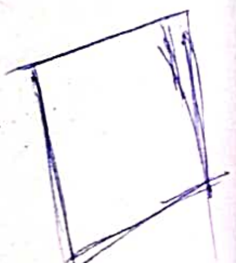
Solution. We use the Bezier cubic polynomial, given in Eq. (7.20), which is:

$$P(u) = [u^3 \;\; u^2 \;\; u \;\; 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix}$$

The control points are (0, 0), (2, 1), (5, 2), (6, 1).

The tangent is given by the derivative of the general equation above,

$$P'(u) = [3u^3 \;\; 2u \;\; 1 \;\; 0]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix}$$



At $u = 0.25$, we get

$$P'(u) = [3(0.25)^2 \quad 2(0.25) \quad 1 \quad 0]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix}$$

$$\begin{bmatrix} 0 & 2 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$ 56 2,1 1

$$= (7.36, 2.91)$$

Po
P1
P2

7.5.3. B-Spline Curve

To design a *B*-spline curve, we need a set of control points, a set of knots and a set of coefficients, one for each control point, so that all curve segments are joined together satisfying certain continuity condition. The computation of the coefficints is perhaps the most complex step because they must ensure certain continuity conditions. Given $(n + 1)$ control points $P_0, P_1, ..., P_n$ and a knot vector $U = \{u_0, u_1, ..., u_m\}$, the *B*-spline curve of degree k defined by these control points and knot vector U is

$$P(u) = \sum_{i=0}^{n} N_{i,k}(u)\, P_i \qquad ...(7.21)$$

where $N_{i,p}(u)$'s are *B*-spline basis functions of degree, the control points called **de Boor** points form the vertices of control polygon.

The basis functions are given by:

$$N_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \qquad ...(7.22)$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \qquad ...(7.23)$$

Equation (7.23) is usually referred to as the **Cox-de Boor recursion formula.** That is, basis function $N_{i,0}(u)$ is 1 if u is in the ith knot span $[u_i, u_{i+1})$. For example, if we have four knots $u_0 = 0$, $u_1 = 1$, $u_2 = 2$ and $u_3 = 3$, knot spans 0, 1 and 2 are $[0, 1)$, $[1, 2)$, $[2, 3)$ and the basis functions of degree 0 are $N_{0,0}(u) = 1$ on $[0, 1)$ and 0 elsewhere, $N_{1,0}(u) = 1$ on $[1, 2)$ and 0 elsewhere, and $N_{2,0}(u) = 1$ on $[2, 3)$ and 0 elsewhere. This is shown in Fig. 7.14.

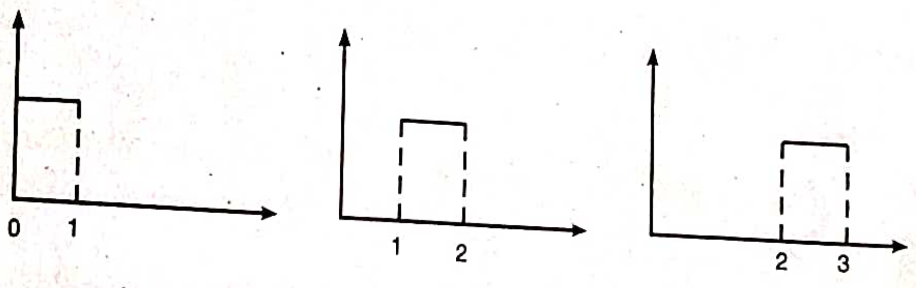


Fig. 7.14. Illustration of Cox-de Boor recusion formula

To understand the way of computing $N_{i,p}(u)$ for k greater than 0, we use the triangular computation scheme. All knot spans are listed on the left (first) column and all degree zero basis functions on the second. This is shown in Fig. 7.15.

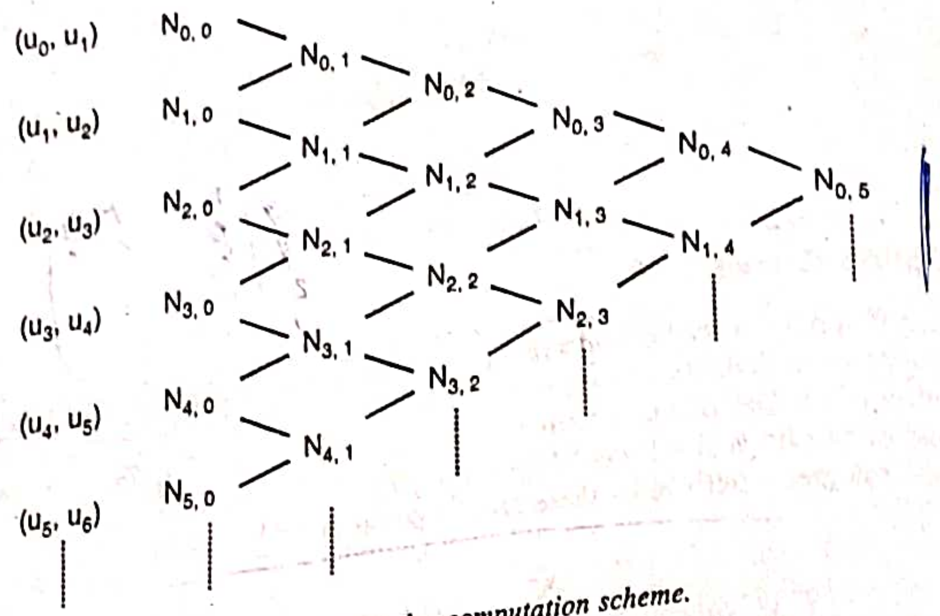


Fig. 7.15. Triangular computation scheme.

To compute $N_{i,1}(u)$, $N_{i,0}(u)$ and N_{i+1}, $0(u)$ are requried. Therefore, we can compute $N_{0,1}(u)$, $N_{1,1}(u)$, ..., $N_{2,1}(u)$, $N_{3,1}(u)$ and so on. All of these $N_{i,1}(u)$'s are written on the third column. Once all $N_{i,1}(u)$'s have been computed, we can compute $N_{i,2}(u)$'s and put them on the fourth column. This process continues until all required $N_{i,k}(u)$'s are computed.

7.5.3.1. Properties of B-spline Curves

The *B*-spline functions have the following properties:

1. $N_{i,k}(u)$ is a degree k polynomial in u.

2. ***Non-negativity***

For all i, k and u, $N_{i,k}(u)$ is non-negative.

3. ***Local support***

$N_{i,k}(u)$ is a non-zero polynomial on $[u_i, u_{i+k+1})$.

4. On any span $[u_i, u_{i+1})$, at most $k + 1$ degree k basis functions are non-zero, namely:

$$N_{i-k,k}(u), N_{i-k+1,k}(u), N_{i-k+2,k}(u), \ldots, N_{i,k}(u)$$

5. ***Partition of unity***

The sum of all non-zero degree k basis functions on span $[u_i, u_{i+1})$ is unity. The previous property shows that $N_{i-k,k}(u), N_{i-k+1,k}(u), N_{i-k+2,k}(u), \ldots, N_{i,k}(u)$ are non-zero on $[u_i, u_{i+1})$. This one states that the sum of these $k + 1$ basis functions is 1.

6. If the number of knots is $m + 1$, the degree of the basis functions is k, and the number of degree k basis functions is $n + 1$, then $m = n + k + 1$.

Let n, $k(u)$ be the last degree k basis function. It is non-zero on $[u_n, u_{n+k+1})$. Since it is the last basis function, u_{n+k+1} must be the last knot u_m. Therefore, we have $u_{n+k+1} = u_m$ and $n + k + 1 = m$. In summary, given m and k, let $n = m - k - 1$ and the degree k basis functions are $N_{0,k}(u), N_{1,k}(u), N_{2,k}(u), \ldots$, and $N_{mn,k}(u)$.

7. Basis function $N_{i,k}(u)$ is a composite curve of degree k polynomials with joining points at knots in $[u_i, u_{i+k+1})$.

The form of a *B*-spline curve is very similar to that of a Bezier curve. Unlike a Bezier curve, a *B*-spline curve involves more information, namely: a set of $n + 1$ control points, a knot vector of $m + 1$ knots, and a degree k. Note that n, m and k must satisfy $m = n + k + 1$. More precisely, if we want to define a *B*-spline curve of degree k with $n + 1$ control points, we have to supply $n + k + 2$ knots $u_0, u_1, \ldots, u_{n+k+1}$. On the other hand, if a knot vector of $m + 1$ knots and $n + 1$ control points are given, the degree of the *B*-spline curve is $k = m - n - 1$. The point on the curve that corresponds to a knot u_i, $P(u_i)$, is referred to as a *knot point*. Hence, the knot points divide a *B*-spline curve into curve segments, each of which is defined on a knot span. Although $N_{i,k}(u)$ looks like $B_{n,i}(u)$, the degree of a *B*-spline basis function is an input, while if the knot vector does not have any particular structure, the generated curve will not touch the first and last legs of the control polygon as shown in Fig. 7.16(*a*). This type of *B*-spline curves is called **open** *B*-spline curves.

We may want to clamp the curve so that it is tangent to the first and the last legs at the first and last control points, respectively, as a Bezier curve does. To do so, the first knot and the last knot must be of multiplicity $k + 1$. This will generate the so-called **clamped** *B*-spline curves. See Figure 7.16 (*b*).

By repeating some knots and control points, the generated curve can be a *closed* one. In this case, the start and the end of the generated curve join together forming a closed loop as shown in Fig. 7.16(*c*).

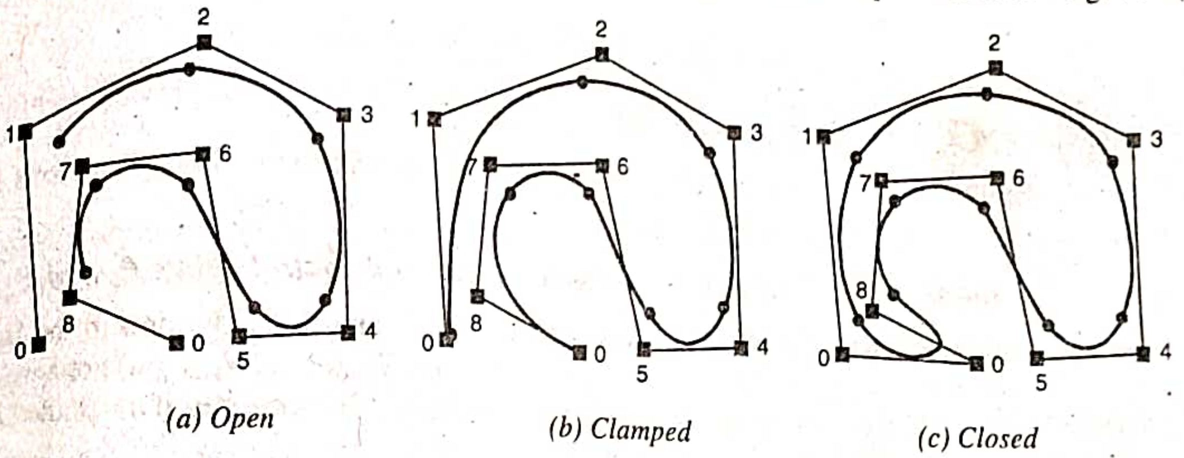


Fig. 7.16. B-spline curves.

Figure 7.16 has $n + 1$ control points ($n = 9$) and $k = 3$. Then, m must be 13 so that the knot vector has 14 knots. To have the clamped effect, the first $k + 1 = 4$ and the last 4 knots must be identical. The remaining $14 - (4 + 4) = 6$ knots can be anywhere in the domain. In fact, the curve is generated with knot vector $U = \{0, 0, 0, 0, 0.14, 0.28, 0.42, 0.57, 0.71, 0.85, 1, 1, 1, 1\}$. Note that except for the first four and last four knots, the middle ones are almost uniformly spaced. The figures also show the corresponding curve segment on each knot span. In fact, the little triangles are the knot points.

B-spline curves share many important properties with Bezier curves, because the former is a generalization of the later. Moreover, *B*-spline curves have more desired properties than Bezier curves. The list below shows some of the most important properties of *B*-spline curves:

In the following, we shall assume a *B*-spline curve $P(u)$ of degree k is defined by $n + 1$ control points and a knot vector $U = \{u_0, u_1, \ldots, u_m\}$ with the first $k + 1$ and last $k + 1$ knots *clamped* (*i.e.*, $u_0 = u_1 = \ldots = u_p$ and $u_{m-k} = u_{m-k+1} = \ldots = u_m$).

8. *B-spline Curve P(u) is a piecewise curve with each component a curve of degree k*

$P(u)$ can be viewed as the union of curve segments defined on each knot span. In Fig. 7.16, where $n = 10$, $m = 14$ and $k = 3$, the first four knots and last four knots are clamped and the 7 internal knots are uniformly spaced. There are eight knot spans, each of which corresponds to a curve segment. In Fig. 7.17 (*a*), these knot points are shown as triangles.

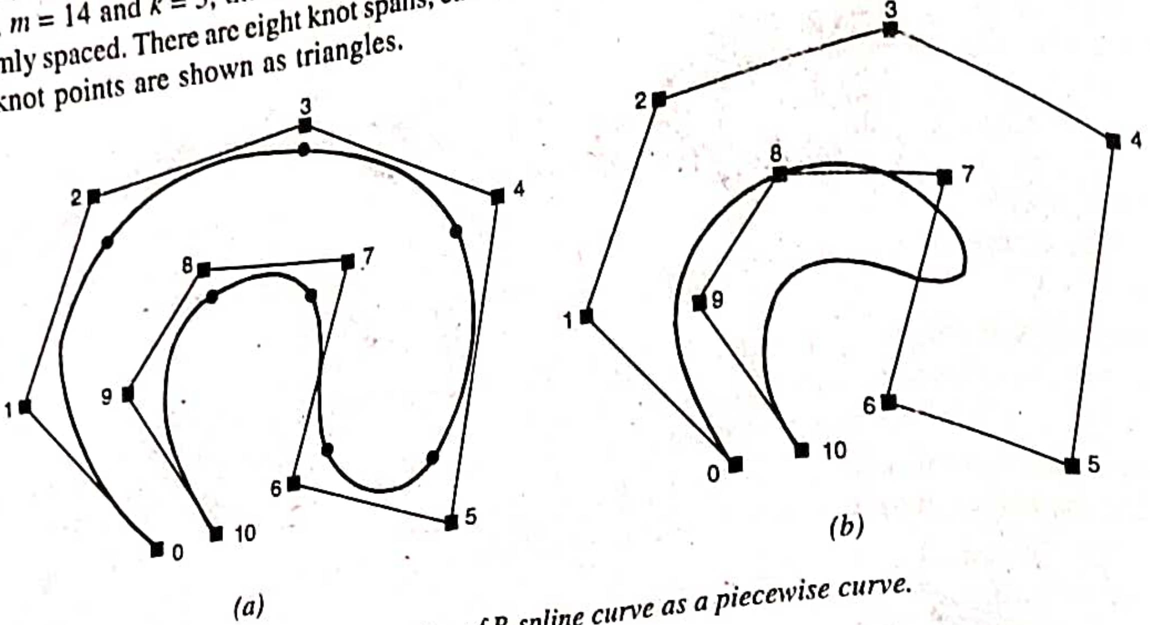


Fig. 7.17. Illustration of B-spline curve as a piecewise curve.

9. *Equality m = n + k + 1 must be satisfied*

Since each control points needs a basis function and the number of basis functions satisfies $m = n + k + 1$.

10. *Clamped B-spline curve C(u) passes through the two end control points P_0 and P_n*

Note that basis function $N_{0,k}(u)$ is the coefficient of control point P_0 and is non-zero on $[u_0, u_{k+1})$. Since $u_0 = u_1 = ... = u_k = 0$ for a clamped *B*-spline curve, $N_{0,0}(u), N_{1,0}(u), ..., N_{k-1,0}(u)$ are zero and only $N_{k,0}(u)$ is non-zero (recall from the triangular computation scheme). Consequently, if $u = 0$, then $N_{0,k}(0)$ is 1 and $P(0) = P_0$. A similar discussion can show $P(1) = P_n$.

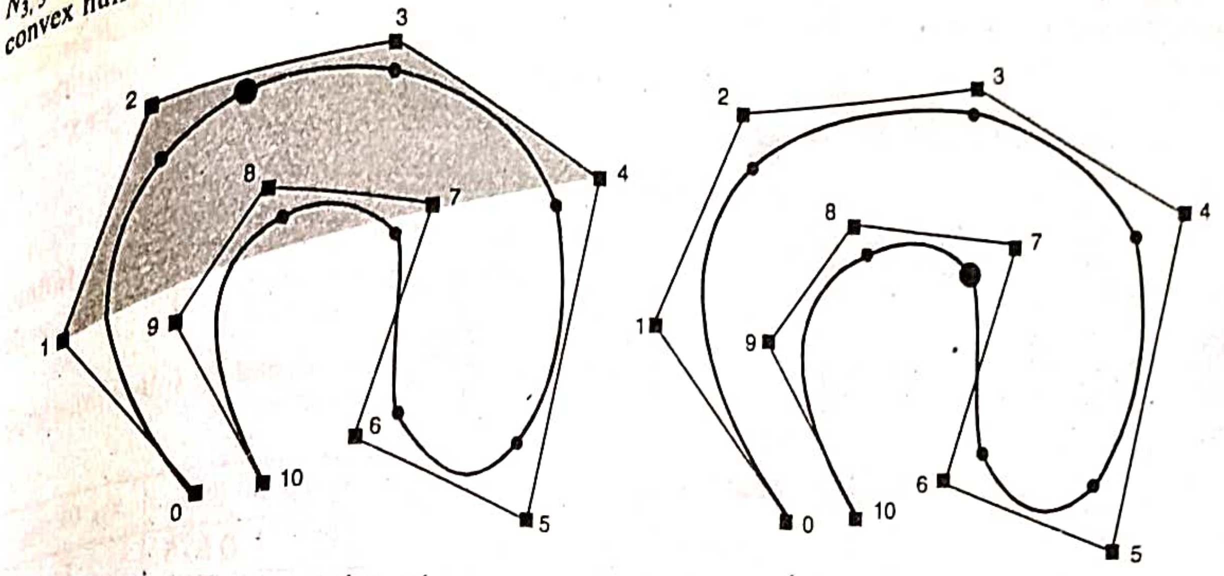
11. *Strong Convex hull property*

A *B*-spline curve is contained in the convex hull of its control polygon. More specifically, if u is in knot span $[u_i, u_{i+1})$, then $P(u)$ is in the convex hull of control points $P_{i-k}, P_{i-k+1}, ..., P_i$. If u is in knot span $[u_i, u_{i+1})$, there are only $k + 1$ basis functions (*i.e.*, $N_{i,k}(u), ..., N_{i-k+1,k}(u), N_{i-k,k}(u)$) non-zero on this knot span. Since $N_{l,k}(u)$ is the coefficient of control point P_l, only $k + 1$ control points $P_i, P_{i-1}, P_{i-2}, ..., P_{i-k}$ have non-zero coefficients. Since on this knot span the basis functions are non-zero and sum to 1, their *weighted* average, $P(u)$, must lie in the convex hull defined by control points $P_i, P_{i-1}, P_{i-2}, ..., P_{i-p}$. The meaning of *strong* is that while $P(u)$ still lies in the convex hull defined by *all* control points, it lies in a much smaller one.

The two *B*-spline curves shown in Fig. 7.18, have 11 control points (*i.e.*, $n = 10$), degree 3 (*i.e.*, $k = 3$) and 15 knots ($m = 14$) with first four and last four knots clamped. Therefore, the number of knot spans is equal to the number curve segments. The knot vector is

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}
0	0	0	0	0.12	0.25	0.37	0.5	0.6	0.75	0.87	1	1	1	1

Figure 7.18(*a*) has *u* in knot span $[u_4, u_5) = [0.12, 0.25)$ and the corresponding point (*i.e.*, $P(u)$) in the second curve segment. Therefore, there are $k + 1 = 4$ basis functions non-zero on this knot span (*i.e.*, $N_{4,3}(u)$, $N_{3,3}(u)$, $N_{2,3}(u)$ and $N_{1,3}(u)$) and thponding control points are P_4, P_3, P_2 and P_1. The shaded area is the convex hull defined by these four points. It is clear that $P(u)$ lies in this convex hull.



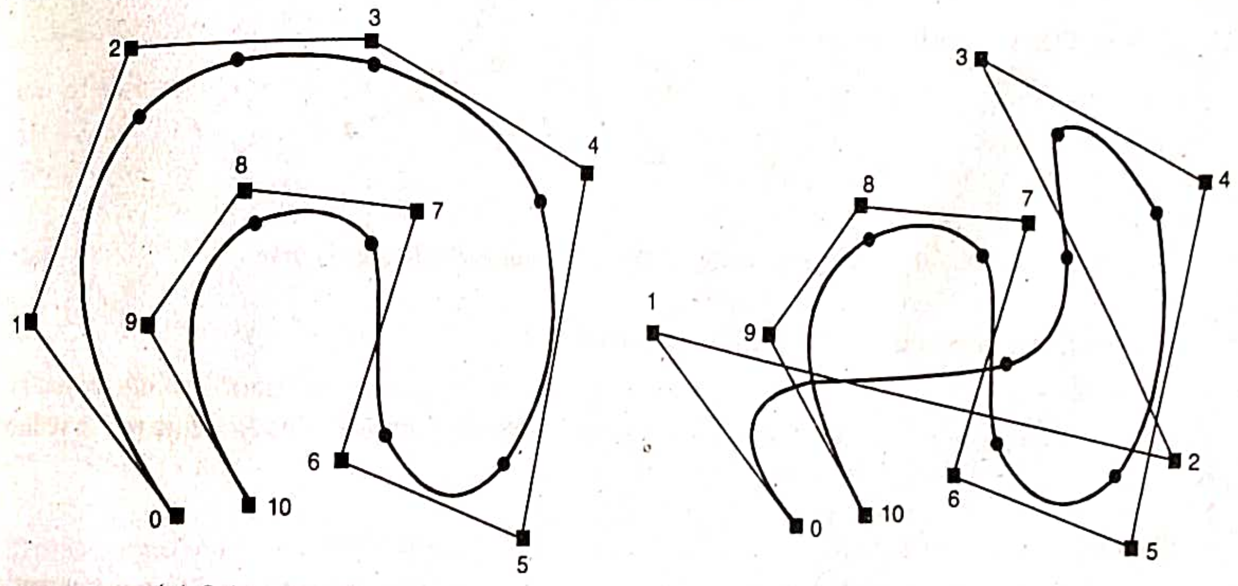
(a) First four knots clamped *(b) Last four knots clamped*

Fig. 7.18. Illustration of convex hull property.

The *B*-spline curve in Fig. 7.18(*b*) is defined in the same way. However, *u* is in $[u_9, u_{10}) = [0.75, 0.87)$ and the non-zero basis functions are $N_{9,3}(u)$, $N_{7,3}(u)$ and $N_{6,3}(u)$. The corresponding control points are P_9, P_8, P_7 and P_6.

12. *Local modification scheme—changing the position of control point P_i only affects the curve $P(u)$ on interval $[u_i, u_{i+k+1})$*

This follows from another important property of *B*-spline basis functions. Recall that $N_{i,k}(u)$ is non-zero on interval $[u_i, u_{i+k+1})$. If *u* is not in this interval, $N_{i,k}(u)P_i$ has no effect in computing $P(u)$ since $N_{i,k}(u)$ is zero. On the other hand, if *u* is in the indicated interval, $N_{i,k}(u)$ is non-zero. If P_i changes its position, $N_{i,k}(u)\, P_i$ is changed and consequently $P(u)$ is changed.



(a) Original position *(b) Effect of moving P_2*

Fig. 7.19. Illustration local modification scheme.

The *B*-spline curves shown in Fig. 7.19 are defined with the same parameters are in the previous convex hull example. We intent to move control point P_2. The coefficient of this control point is $N_{2,3}(u)$ and the interval on which this coefficient is non-zero is $[u_2, u_{2+3+1}) = [u_2, u_6) = [0, 0.37)$. Since $u_2 = u_3 = 0$, only three segments that correspond to $[u_3, u_4)$ (the domain of the first curve segment), $[u_4, u_5)$ (the domain of the second curve segment) and $[u_5, u_6)$ (the domain of the third curve segment) will be affected. The Figure 7.19(*b*) shows the result of moving P_2 to the lower right corner. As you can see, only the first, second and third curve segments change their shapes and all remaining curve segments stay in their original place without any change.

13. P(u) is C^{k-l} continuous at a knot of multiplicity l

If u is not a knot, $P(u)$ is in the middle of a curve segment of degree k and is therefore infinitely differtiable. If u is a knot in the non-zero domain of $N_{i,k}(u)$, since the latter is only C^{k-l} continuous, so does $P(u)$.

The *B*-spline curve (Fig. 7.20) has 18 control points (*i.e.*, $n = 17$), degree 4, and the following clamped knot vectors.

u_0 to u_4	u_5	u_6 and u_7	u_8	u_9 to u_{11}	u_{12}	u_{13} to u_{16}	u_{17}	u_{18} to u_{22}
0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1

Thus, u_6 is a double knot, u_9 is a triple knot and u_{13} is a quadruple knot. Consequently, $C(u)$ is of C_4 continuous at any point that is not a knot, C_3 continuous at all simple knots, C_2 continuous at u_6, C_1 continuous at u_9, C_0 continuous at u_{13}.

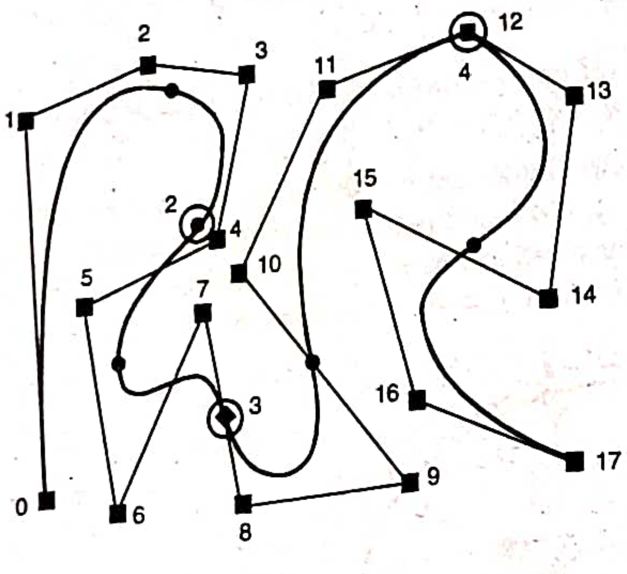


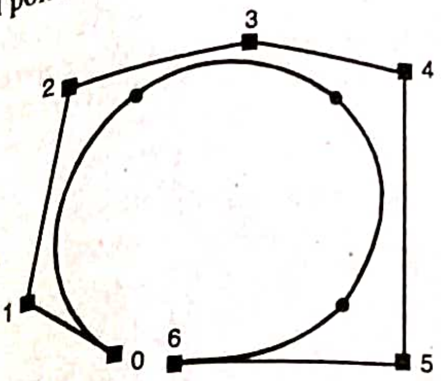
Fig. 7.20. Illustration of continuity property of B*-spline curve.*

14. Bezier curves are special cases of* B*-spline curves

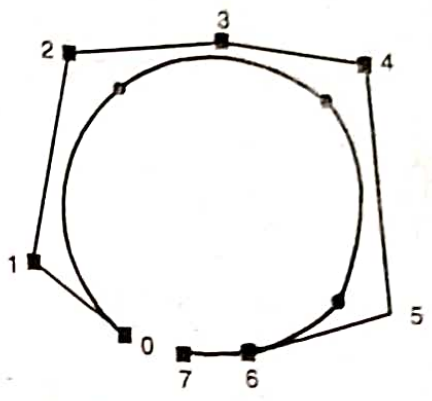
If $n = k$ (*i.e.*, the degree of a *B*-spline curve is equal to n, the number of control points minus 1), and there are $2(k + 1) = 2(n + 1)$ knots with $k + 1$ of them clamped at each end, this *B*-spline curve reduces to a Bezier curve.

15. Finding a point on a* B*-spline curve—de Casteljau's Algorithm

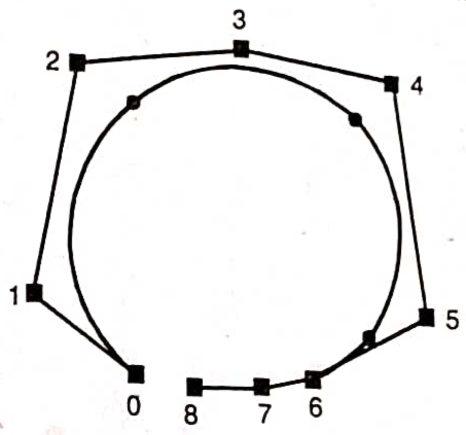
De Boor's algorithm is a generalization of de Casteljau's algorithm. It provides a fast and numerically stable way for finding a point on a *B*-spline curve given a u in the domain. Recall from a property of *multiple knots* that increasing the multiplicity of an internal knot decreases the number of non-zero basis *functions at this knot*. In fact, if the multiplicity of this knot is l, there are at most $k - l + 1$ non-zero basis *functions at this knot*. Consequently, at a knot of multiplicity p, there will be only one non-zero basis *function whose* value at this knot is one because of the property of partition of unity. Let this knot be *ui*. *If u is* u_i, since $N_{i,k}(u)$ is non-zero on $[u_i, u_{i+1})$, the point on the curve $P(u)$ is affected by *exactly one control point* P_i. More precisely, we actually have $P(u) = N_{i,k}(u)\, P_i = P_i$!



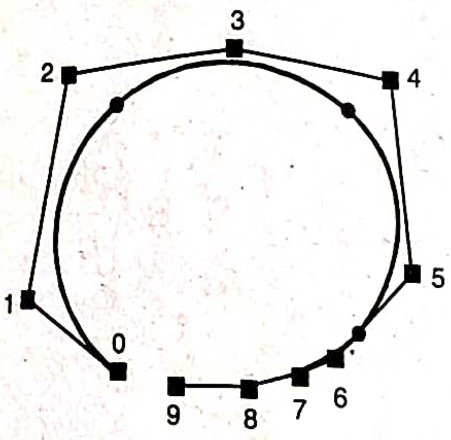
(a) Original position



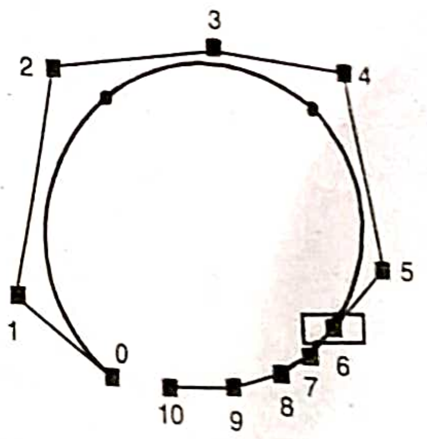
(b) Effect of inserting one point



(c) Effect of inserting two points



(d) Effect of inserting three points



(e) Effect of inserting four points

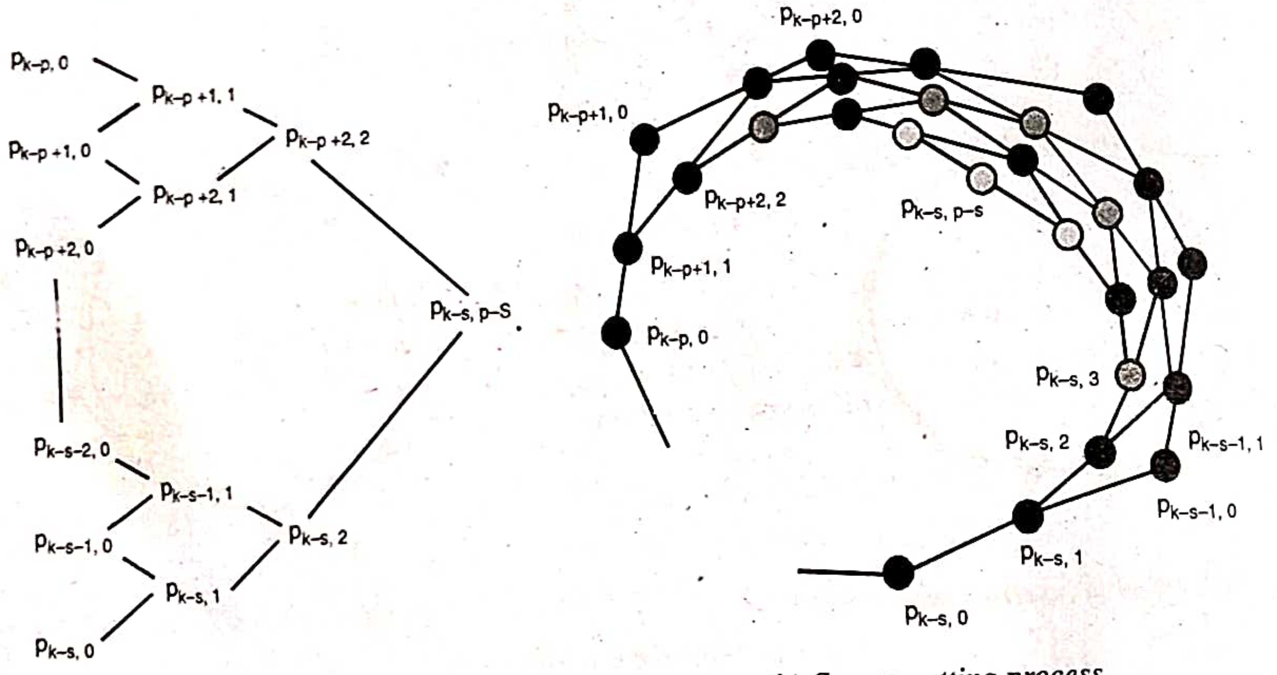
Fig. 7.21. Effect of inserting knots

So, what is the point of this interesting or somewhat strange property ? Simply: if a knot u is inserted **repeatedly so that its multiplicity is k, the last generated new control point is the point on the curve**

that corresponds to u. Why is this so ? After inserting u multiple times, making its multiplicity k, the triangular computation scheme yields one point. Because the given B-spline curve must pass this new point, it is the point on the curve corresponding to u. This observation provides us with a technique for finding $P(u)$ on the curve. We just insert u until its multiplicity becomes k and the *last* point is $P(u)$!

Figure 7.21(a) is a B-spline curve of degre 4 defined by 7 conrol point. To compute $P(0.9)$, where 0.9 is not a knot, $u = 0.9$ is inserted 4 (degree) times. Figure 7.21 (b) and 7.21(c) show the rsults after the first and second insertion. Thus, two new control points are added near the lower right corner. Note that the control polygon is closer to the curve than the original. The results of the third insertion is shown in Fig. 7.21 (d). The fourth insertion yields the point on the curve (Fig. 7.21 (e)). Therefore, after four insertions, $P(0.9)$ is a control point.

```
Input: A value $u$
Output: The point on the curve, $P(u)$
If $u$ lies in $[u_l, u_{l+1})$ and $u! = u_l$, let $h = k$ (i.e., inserting $u$ $k$ times) and $s = 0$;
If $u = u_l$ and $u_l$ is a knot of multiplicity $s$, let $h = k - s$ (i.e., inserting $u$ $k - s$ times);
Copy the affected control points $P_{l-s}, P_{l-s-1}, P_{l-s-2}, \ldots, P_{l-k+1}$ and $P_{l-k}$ to a new array and rename
them as $P_{l-s,0}, P_{l-s-1,0}, P_{l-s-2,0}, \ldots, P_{l-k+1,0}$;
for $r := 1$ to $h$ do
    for $i := l - k + r$ to $l - s$ do
        begin
            Let $a_{i,r} = \dfrac{(u - u_i)}{(u_{i+k-r+1} - 1)}$
            Let $P_{i,r} = (1 - a_{i,r})\, P_{i-1,r-1} + a_{i,r} P_{i,r-1}$
        end
    $P_{l-s,k-s}$ is the point $P(u)$.
```



(a) Points and their relations *(b) Corner cutting process*

Fig. 7.22. Illustration of de Casteljau's algorithm

In Figure 7.22(a), all $P_{i,0}$'s are on the left column. From the 0th column and coefficients $a_{i,1}$, one can compute $P_{i,1}$'s. From this first column and coefficients $a_{i,2}$'s, the second column is computed and so on. Since there are $(l-s)-(l-k)+1 = k-s+1$ points on the 0th column and since each column has one point less than the previous one, it takes $k-s$ columns to reduce the number of points on a column to 1. This is why the last point is $P_{l-k,l-s}$. Figure 7.22(b) shows the corner-cutting process.

Although this process looks like the one obtained from de Casteljau's algorithm, they are very different. **First,** under de Casteljau's algorithm, the dividing points are computed with a pair of numbers $1-u$ and u that never change throughout the computation procedure, while under de Boor's algorithm these pairs of numbers are different and depend on the column number and control point number. **Second,** in de Boor's algorithm only $k+1$ affected control points are involved in the computation, while de Casteljau's algorithm uses all control points. Since control points P_{l-k} to P_l define a convex hull that contains the curve segment on knot span $[u_l, u_{l+1})$, the computation of de Boor's algorithm is performed within the corresponding convex hull.

16. Types of B-spline Curves

There are several types of B-spline curves, discussed as follows:

B-spline open curves. If the first and last knots do not have multiplicity $k+1$, where k is the degree of a B-spline curve, the curve will not be tangent to the first and last legs at the first and last control points, respectively. The curve is an open B-spline curve (Fig. 7.23).

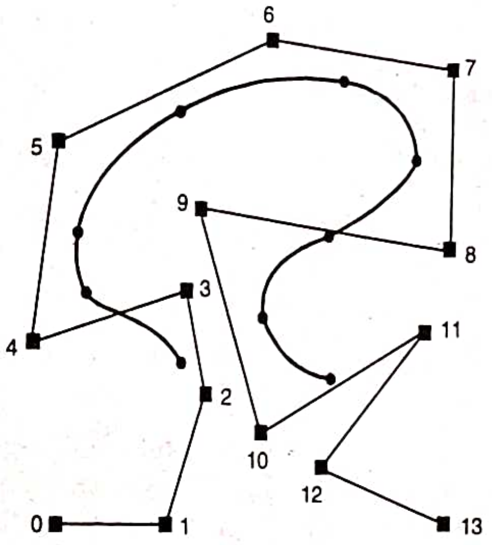


Fig. 7.23. Open curve.

Consider a B-spline curve of degree 6 (*i.e.*, $k = 6$) defined by 14 control points (*i.e.*, $n = 13$). The number of knots is 21 (*i.e.*, $m = n + k + 1 = 20$). If the knot vector is *uniform*, the knots are 0, 0.05, 0.10, 0.15, ..., 0.90, 0.95 and 1.0. The *open* curve is defined on $[u_k, u_{n-k}] = [u_6, u_{14}] = [0.3, 0.7]$ and is not tangent to the first and last legs. The Figure 7.23 shows the curve.

The values of u_i are given by:

$$u_j = \begin{cases} 0 & \text{if } j < k \\ j-k & \text{if } k \le j \le n \\ n-k+1 & \text{if } j > n \end{cases} \qquad \text{...(7.24)}$$

where $\qquad 0 \le j \le n+k+1$

and the range of u is :

$$0 \leq u \leq n-k+1 \qquad \text{...(7.25)}$$

For example, an open B-spline curve of degree 4 defined by 9 control points (*i.e.*, $n = 8$) and a uniform knot vector {0, 1/13, 2/13, 3/13, ..., 12/13, 1}. Describe the change betweeen an open curve and a clamped one.

If we change the second knot 1/13 to 0 making 0 a double knot, the result is curve-1 in the Fig. 7.24.

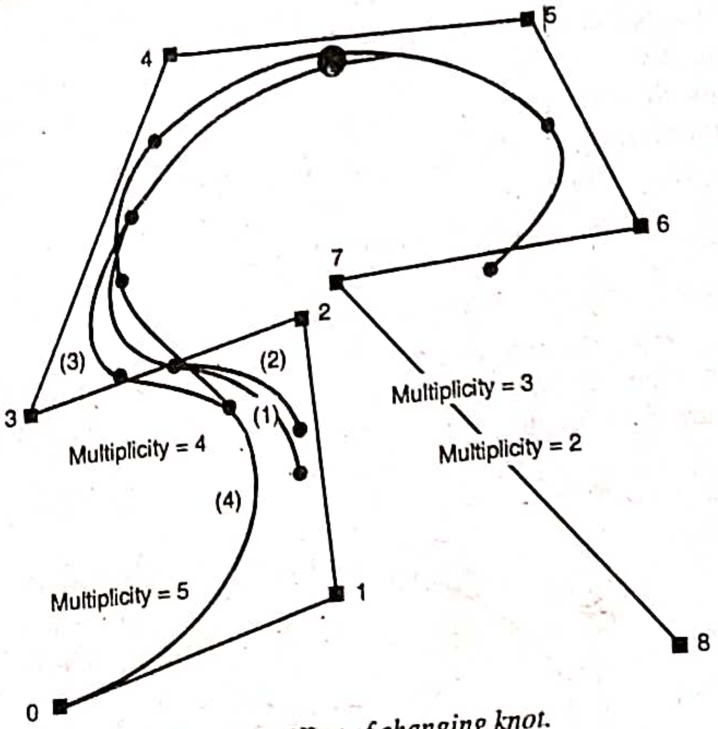


Fig. 7.24. Effect of changing knot.

In fact, this curve and the original one with 0 being a simple knot are almost identical. Now, if we change the third knot 2/13 to 0 making0 a knot of multiplicity 3, the result is curve-2.

If the fourth knot 3/13 is change to 0 (multiplicity 4), the resulting curve is curve-3. As you can see, these three open curves are not very different from each other.

Now, let us make the fifth knot 4/13 to 0. Since 0 is know a knot of multiplicity 5 (*i.e.*, $k + 1$), the curve not only passes through the first control point but also is tangent to the first leg to the control polygon (*i.e.*, clamped). As you can see from curve-4, the shape of the curve changes drastically by pulling one end of it to the first control point. The same holds true if we make the last 5 knots to 1.

B-spline closed curves. There are many ways to generate closed curves. The simple ones are either wrapping control points or wrapping knot vectors. The values of *ui* are given by:

$$u_j = j \qquad \text{...(7.26)}$$
$$0 \leq j \leq n+1$$

and the range of u is:

$$0 \leq u \leq n+1 \qquad \text{...(7.27)}$$

Example 7.4

Find the equation of a cubic B-spline curve. Compare the same with the Bezier curve.

Solution. The cubic spline has $n = 3$, $k = 3$