Chapter

14

# INTRODUCTION TO CURVE GENERATION

## 14.1 INTRODUCTION

Straight lines have a simple mathematical form which makes them easy to deal with in such operations as transformation or clipping. However, many real world objects are inherently smooth and involve curves to represent them. Natural objects are neither perfectly flat nor smoothly curved but often have rough, jagged contours. In this chapter, we shall introduce methods for generating curves.

## 14.2 CURVE GENERATION

There are two approaches to draw curved lines :

- Curve generation algorithm
- Interpolation

### 14.2.1 Curve Generation Algorithm

One approach to curve generation is to develop a curve generation algorithm such as a DDA. With this approach a true curve is created.

Digital differential analyzer algorithm uses the differential equation of the curve. If we know a differential equation for the curve, we can write a digital differential analyzer algorithm which will calculate the coordinates of points on the curve. The differential equations for simple curves such as circles and ellipses are fairly easy to solve and generation algorithms for them can be implemented in hardware.

Let us see the DDA algorithm for generating circular arcs. The equations for the arc coordinates can be written in terms of an angle parameter θ as follows :

$$\left.\begin{aligned} x &= R\cos\theta + x_0 \\ y &= R\sin\theta + y_0 \end{aligned}\right\} \quad \ldots (1)$$

where $(x_0, y_0)$ is the center of curvature and $R$ is the arc radius.

Differentiating equation (1), we get

$$dx = -R \sin\theta \, d\theta$$
$$dy = R \cos\theta \, d\theta$$

From equation (1)

$$R\cos\theta = x - x_0$$
$$R\sin\theta = y - y_0$$

So,

$$dx = -(y - y_0)\, d\theta$$
$$dy = (x - x_0)\, d\theta$$

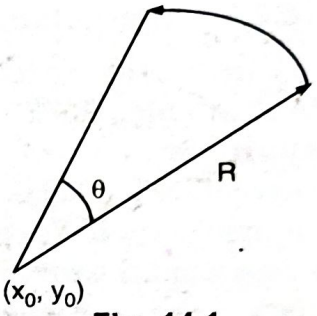


Fig. 14.1

The values of $dx$ and $dy$ indicate the increment in $x$ and $y$ respectively. Thus, to get a new point $(x_2, y_2)$, we can write

$$x_2 = x_1 + dx = x_1 - (y_1 - y_0)\, d\theta$$
$$y_2 = y_1 + dy = y_1 + (x_1 - x_0)\, d\theta$$

These equations form the basis for an arc generation algorithm. From equations, we can see that the next point on the arc is the function of $d\theta$. To have a smooth curve, the neighbouring points on the arc should be close to each other. To achieve this, the value of $d\theta$ should be small enough not to leave gaps in the arc. Usually, the value of $d\theta$ can be determined from the following equation.

$$d\theta = \min(0.01, 1/3.2 * (|x - x_0| + |y - y_0|))$$

**The arc-generating algorithm is as follows :**

**Algorithm**

(1) Read the centre of a curvature $(x_0, y_0)$
(2) Read the arc angle, say $\theta$
(3) Read the starting point of the arc $(x, y)$
(4) Calculate $d\theta$ as

$$d\theta = -\min(0.01, 1/(3.2 * (|x - x_0| + |y - y_0|))$$

(5) Initialize angle = 0
(6) While (angle < θ)

```
do
    {
        Put pixel (x, y, intensity)
        x = x - (y - y0) * dθ
        y = y - (x - x0) * dθ
        angle = angle + dθ
    }
```

(7) Stop.

**Problems in True-Curve Generation Approach**

(1) To specify a curve, we need more information than just its endpoints. This may mean that a different display-file structure should be used.
(2) Another problem can arise from transformations. A line segment when scaled is still a line segment, but other curves may behave differently. *For example*, a circle when scaled in only one direction becomes an ellipse.
(3) If we wished to clip arcs to some window boundary, a new clipping algorithm would have to be developed.
(4) Not every curve which we wish to draw has a simple generation algorithm.

### 14.2.2 Interpolation

In practice we have to deal with complex curves for which no simple mathematical definition is available. Such curves can be drawn using approximation methods. We can draw an approximation to such a curve if we have an array of sample points. We can then guess what the curve should look like between the sample points. If the curve is smooth and our samples are close enough together, we can guess the shape, of the curve.

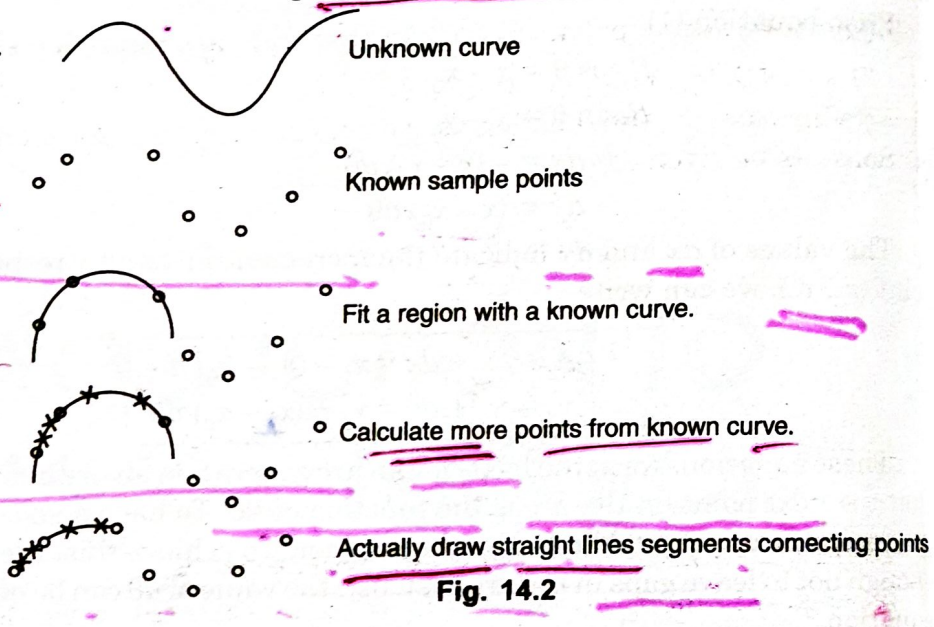



**Fig. 14.2**

Our guess will probably not be exactly right, but it will be close enough for appearances. We fill in portions of the unknown curve with pieces of known curves which pass through the nearby sample points. Since the known and unknown curves share these sample points in a local region, we assume that in this region, the two curves look pretty much alike. We fit a portion of the unknown curve with a curve that we know. Now we can fill in a gap between the sample points by finding the coordinates of points along the known approximating curve and connecting these points with line segments.

The main task in this process is to find the suitable mathematical expression for the known curve. There are polynomial, trigonometric, exponential and other classes of functions that can be used to approximate the curve. Usually polynomial functions in the parametric form are preferred. The polynomial functions in the parametric form can be given as

$$\begin{aligned} x &= f_x(u) \\ y &= f_y(u) \\ z &= f_z(u) \end{aligned}$$

There are a couple of reasons to use parametric form.

(1) It treats all three directions equally.

(2) It allows multiple value. (several $y$ or $z$ values for a given $x$) so that curves can double back or even cross themselves.

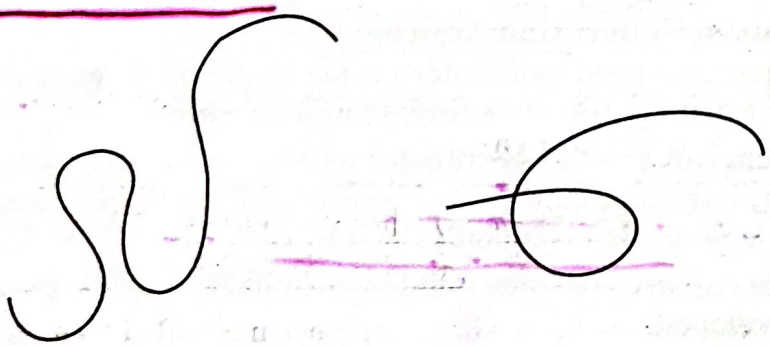


**Fig. 14.3** Representation of curves with double back or crossing themselves.

We have seen that, we have to draw the curve by determining the intermediate points between the known sample points. This can be achieved using interpolation techniques. Let's see the interpolation process.

Suppose we want a polynomial curve that will pass through $n$ sample points.

$(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots (x_n, y_n, z_n)$

We will construct the function as the sum of terms, one term for each sample point. These functions can be given as

$$f_x(u) = \sum_{i=1}^{n} x_i B_i(u)$$

$$f_y(u) = \sum_{i=1}^{n} y_i B_i(u)$$

$$f_z(u) = \sum_{i=1}^{n} z_i B_i(u)$$

The functions $B_i(u)$ are called **blending functions.**

For each value of $u$, the blending function determines how much the $i^{th}$ sample point affects the position of the curve. In other words we can say that each sample points tries to pull the curve in its direction and the function $B_i(u)$ gives the strength of the pull. If for some value of $u$, $B_i(u) = 1$ for unique value of $i$ (*i.e.*, $B_i(u) = 0$ for other values of $i$) then $i^{th}$ sample point has complete control of the curve and the curve will pass through $i^{th}$ sample point. For different value of $u$, some other sample point may have complete control of the curve. In such case the curve will pass through that point as well. In general, the blending functions give control of the curve to each of the sample points in turn for different values of $u$. Let's assume that the first sample point $(x_1, y_1, z_1)$ has complete control when $u = -1$, the second when $u = 0$, the third when $u = 1$, and so on. *i.e.*,

when $u = -1 \Rightarrow B_1(u) = 1$ and 0 for $u = 0, 1, 2, \ldots, n-2$

when $u = 0 \Rightarrow B_1(u) = 1$ and 0 for $u = -1, 1, \ldots, n-2$

⋮ ⋮ ⋮

when $u = (n-2) \Rightarrow B_n(u) = 1$ and 0 for $u = -1, 0, \ldots, (n-1)$

To get $B_1(u) = 1$ at $u = -1$ and 0 for $u = 0, 1, 2, \ldots, n-2$, the expression for $B_1(u)$ can be given as

$$B_1(u) = \frac{u(u-1)(u-2)\ldots[u-(n-2)]}{(-1)(-2)\ldots(1-n)}$$

where denominator term is a constant. In general form $i^{th}$ blending function which is 1 at $u = i - 2$ and 0 for other integers can be given as :

$$B_i(u) = \frac{(u+1)(u)(u-1)\ldots[u-(i-3)][u-(i-1)]\ldots[u-(i-2)]}{(i-1)(i-2)(i-3)\ldots(1)(-1)\ldots(i-n)}$$

The approximation of the curve using above expression is called **Lagrange interpolation.** From the above expression blending functions for four sample points can be given as

$$B_1(u) = \frac{u(u-1)(u-2)}{(-1)(-2)(-3)}$$

$$B_2(u) = \frac{(u+1)(u-1)(u-2)}{1(-1)(-2)}$$

$$B_3(u) = \frac{(u+1)u(u-2)}{(2)(1)(-1)}$$

$$B_4(u) = \frac{(u+1)u(u-1)}{(3)(2)(1)}$$

Using these equations (functions) and four sample points, we can construct a curve which passes through the four sample points.

$$x = x_1 B_1(u) + x_2 B_2(u) + x_3 B_3(u) + x_4 B_4(u)$$
$$y = y_1 B_1(u) + y_2 B_2(u) + y_3 B_3(u) + y_4 B_4(u)$$
$$z = z_1 B_1(u) + z_2 B_2(u) + z_3 B_3(u) + z_4 B_4(u)$$

It is possible to get intermediate points between two sampling points by taking values of $u$ between the values of $u$ related to the two sample points under consideration. *For example*, we can find the intermediate points between second and third sample points for which values of $u$ are 0 and 1, respectively; by taking values of $u$ between 0 and 1. This is shown in Fig. 14.4.

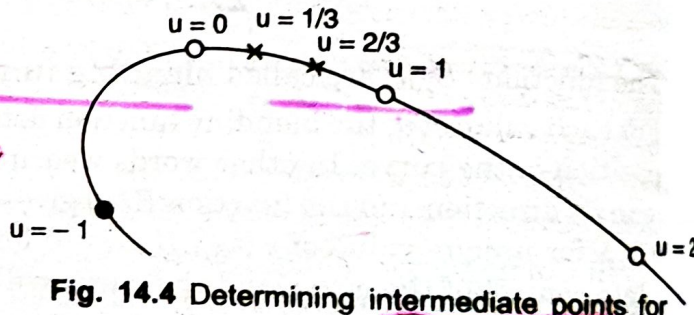



**Fig. 14.4** Determining intermediate points for approximation of curve.

The subsequent intermediate points can be obtained by repeating the same procedure. Finally the points obtained by this procedure are joined by small straight line segments to get the approximated curve.

**Interpolating Algorithm**

1. Get the sample points.
2. Get intermediate values of $u$ to determine intermediate points.
3. Calculate blending function values for middle section of the curve.
4. Calculate blending function values for first section of the curve.
5. Calculate blending function values for the last section of the curve.
6. Multiply the sample points by blending functions to give points on approximation curve.
7. Connect the neighbouring points using straight line segments.
8. Stop.

### 14.2.3 Interpolating Polygons

Blending functions can also be used to round the sides of polygon. Smoothing of each side of the polygon is done by replacing it with several small line segments. We start out with a polygon that has only a few sides and end up with a polygon which has many more sides and appears smoother due to the interpolation. This is illustrated in Fig. 14.5.

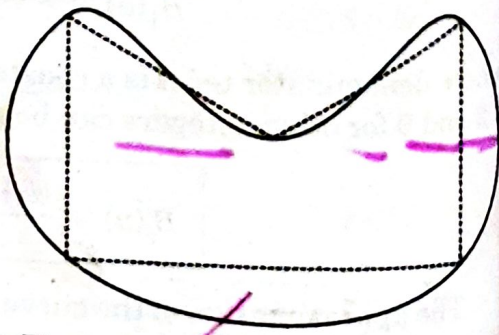


**Fig. 14.5** Smoothing of a polygon.

**Algorithm**

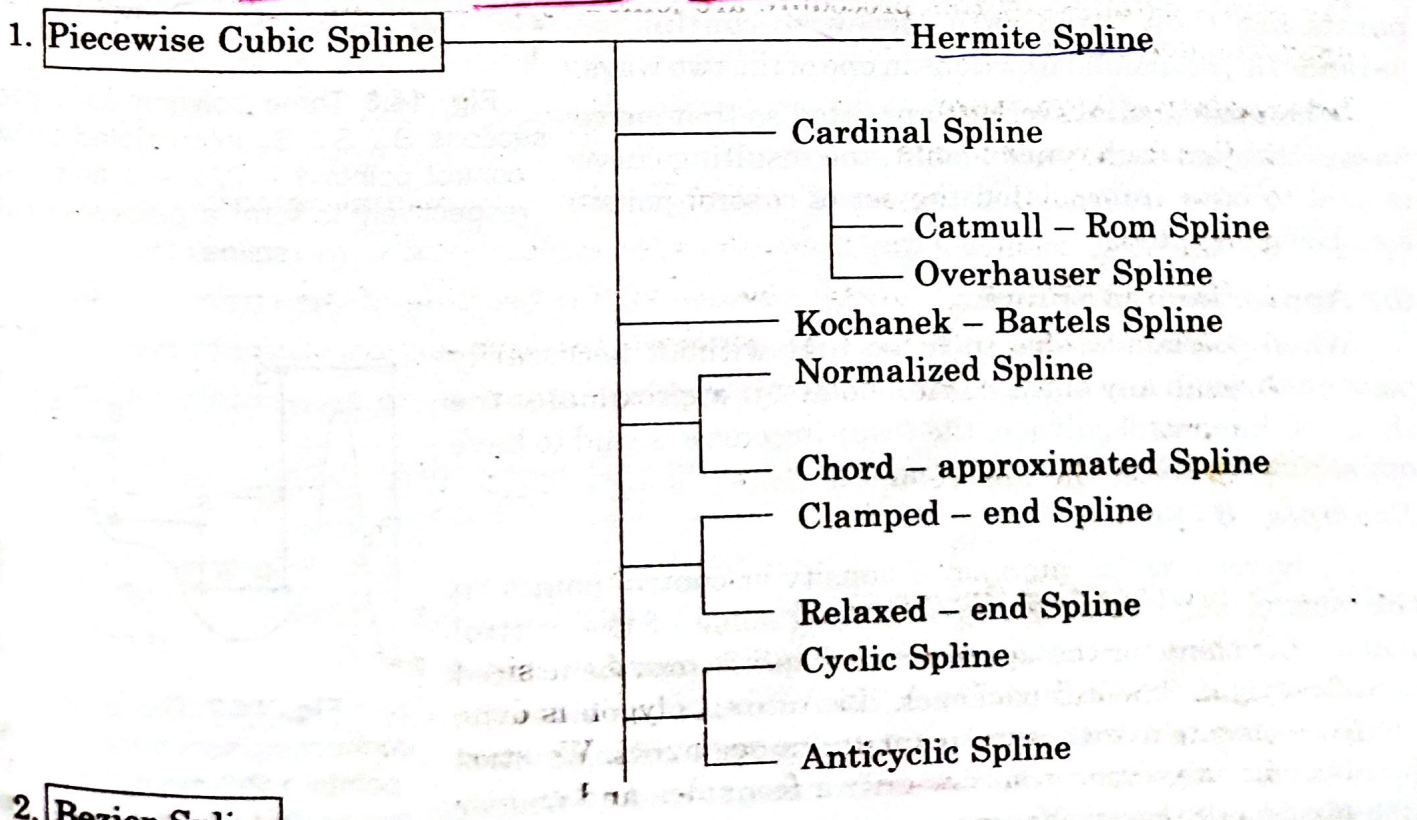
1. Read the original number of polygon sides, say $N$.
2. Read the vertices of original polygon.
3. Read the blending function values.
4. Read the count for the number of small line segments per original polygon side.
5. Smooth one side of original polygon by stepping through four blending functions.
6. Repeat step 5 according to number of small line segments required per original polygon side.
7. Repeat step 5 and 6 for each side of the original polygon.
8. Draw the polygon using small line segments.
9. Stop.

## 14.3 SPLINE REPRESENTATION

In industries like shipbuilding, automobile and aircraft, it is a common practice to layout the model's shape curves in full or near full scale on a large drafting floor. This is done using a long, narrow strip of wood or plastic known as loftman's spline*. Several small lead weights are distributed along the length of such physical splines. By varying the number and position of the lead weights the shape of the spline is changed so that it passes through some designated set of data-points and appears smooth. The concept of mathematical spline curve used in computer aided geometric design is derived from the physical industrial spline.

In Computer Graphics, the term spline curve refers to any composite curve formed with polynomial sections satisfying specified boundary condition at the section endpoints.

Depending on the type of polynomial used and set of boundary conditions satisfied, the following varieties of spline curves are used in graphic applications.




* The skilled craftsmen is shipbuilding industry who perform this task are known a 'loftsmen'.

3. B – Spline
   - Open B – Spline
   - Closed B – Spline
   - Uniform Non rational B – Spline
   - Non-uniform B – Spline
     - Rational B – Spline (NURB)
     - Non-rational B – Spline

4. Beta or β Spline

Inspite of the above varieties of splines they can be broadly classified under following two basic categories depending on how they fit the given set of data points, also known as the *control points.*

(*a*) Interpolated Spline.

(*b*) Approximated Spline.

**(*a*) Interpolated Spline**

We specify a spline curve by giving a set of coordinate positions, called 'control points', which indicate the general shape of the curve. These control points are then fitted with piecewise continuous parametric polynomial functions in one of the two ways.

When polynomial sections are fitted so that curve passes through each control point, the resulting curve is said to have interpolated the set of control points *e.g.,* Hermite spline.

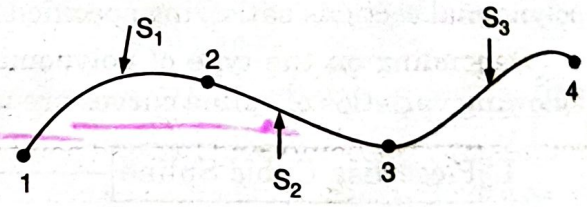



**Fig. 14.6** Three polynomials curve sections $S_1$, $S_2$, $S_3$ interpolated between control points 1 – 2, 2 – 3 and 3 – 4 respectively to form a piecewise cubic spline.

**(*b*) Approximated Spline**

When polynomial are so fitted that without necessarily passing through any of the control points, it approximates the shape of the control polygon, the resulting curve is said to have *approximated* the set of control points.

***Example :*** *B - Spline.*

By increasing the number of density or control points on the control polygon or by repositioning some of the control points, the shape of the approximated splines can be altered significantly. Good CAD packages like Auto CAD provides the facility to 'insert' extra control point and 'move' existing control points in an interactive graphics environment to aid a designer in adjusting the curve shapes.

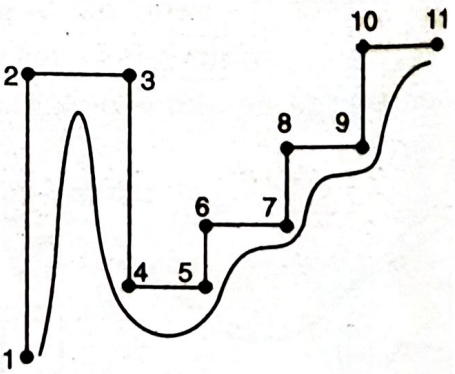



**Fig. 14.7** The polyline connecting sequence of control points 1-2-3-4-5-6-7-8-9-10 is the control polygon. The spline curve shown approximates the shape of the control polygon.

Thus approximated splines are useful in drawing contour lines for Geographic Information System, (GIS) applications or in designing the 'skin' of car bodies, ship hulls, aircraft fuselages, furniture, glassware etc. On the other hand interpolation splines are commonly used to digitize drawing, specify animation paths and graph the data trend of a discrete set of data points.

## 14.4 PIECEWISE CUBIC SPLINE

Given a set of control points, piecewise cubic splines are obtained by fitting each pair of specified points with interpolating cubic polynomial curve sections satisfying some continuity conditions at the common end points.

If there are $n + 1$ control points to fit with a piecewise cubic spline, then we have to construct $n$ interpolating curve pieces joining hands at $n - 1$ interior control points. Each piece of curve is interpolated between a pair of adjacent control points in any of the following methods using different set of boundary conditions.

1. Hermite Interpolation
2. Cardinal Spline Interpolation
3. Kochanck - Bartels Formulation

Here we will discuss only the **Hermite splines**, the simplest of its kind.

### 14.4.1 Hermite Spline

Let us first observe how a single cubic spline piece can be fitted between two given points using Hermite interpolation method. Then we will talk about continuity conditions for smoothly joining adjacent pieces to form a continuous piecewise Hermite spline.

Unlike the cubic polynomial curves which use four points on curve as known geometric quantities, the Hermite interpolation requires two end points and two end point tangents specified. Thus no intermediate control points are required.

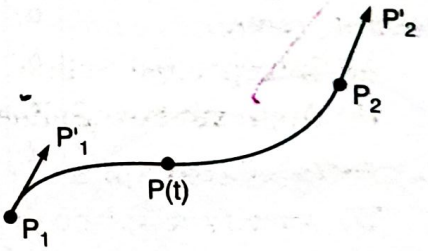



**Fig. 14.8** A Hermite spline-piece defined by two end points $P_1$, $P_2$ and two endpoint tangent vectors $P¢_1$, $P¢_2$.

If $P(u)$ represents a parametric cubic point function for the curve section between control points $P_K$ and $P_{K+1}$ then the boundary conditions that defines this Hermite curve section are

$$\begin{aligned} P(0) &= P_k \\ P(1) &= P_{k+1} \\ P'(0) &= DP_k \\ P'(1) &= DP_{k+1} \end{aligned}$$

With $DP_k$ and $DP_{k+1}$ specifying the values for the parametric derivatives (slope of the curve) at control points $P_k$ and $P_{k+1}$ respectively.

For Hermite–curve section

$$P(u) = au^3 + bu^2 + cu + d \quad \ldots (1)$$

where $0 \le u \le 1$

If $x$ is the component of $P$ then

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$ and similarly for the $y$ and $z$ co-ordinates.

The matrix-equivalent of equation (1) is

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

and $$P'(u) = [3u^2 \quad 2u \quad 1 \quad 0] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Substituting endpoints values 0 and 1 for parameter $u$, we can express the Hermite boundary conditions in the matrix form

$$\begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Solving it

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M_H \begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix}$$

where $M_H$, the Hermite matrix, is the inverse of the boundary constraint matrix.

$$\therefore \quad P[u] = [u^3 \; u^2 \; u \; 1] \; M_H \begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix}$$

Finally, we can determine expressions for the Hermite blending functions by carrying out the matrix multiplication and collecting coefficients for the boundary constraints to obtain the polynomial form :

$$P(u) = P_k(2u^3 - 3u^2 + 1) + P_{k+1}(-2u^3 + 3u^2) + DP_k(u^3 - 2u^2 + u) + DP_{k+1}(u^3 - u^2)$$

$$P(u) = P_k H_0(u) + P_{k+1} H_1(u) + DP_k H_2(u) + DP_{k+1} H_3(u)$$

The polynomial $H_k(u)$ for $k = 0, 1, 2, 3$ are referred to as blending functions because they blend the boundary constraint values to obtain each co-ordinate position along the curve.

Here the basis function matrix $[F] = [B_{0,3}(u) \quad B_{1,3}(u) \quad B_{2,3}(u) \quad B_{3,3}(u)]$

The Bezier geometry matrix

$$[B] = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$[T] = [u^3 \quad u^2 \quad u \quad 1]$$

and the Bezier basis matrix $[M] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

such that $[F] = [T]\,[M]$

Therefore, a cubic Bezier curve controlled by the points $P_0, P_1, P_2, P_3$ is

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

**Properties of Bezier Curve**

1. The basis functions are real.
2. Bezier curve always passes through the first and last control points *i.e.*, curve has same end points as the guiding polygon.
3. The degree of the polynomial defining the curve segment is one less than the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is three, *i.e.*, cubic polynomial.
4. The curve generally follows the shape of the defining polygon.
5. The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segments.
6. The curve lies entirely within the convex hull formed by four control points.
7. The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
8. The curve exhibits the variation diminishing property. This means that the curve does not oscillate about any straight line more often than the defining polygon.
9. The curve is invariant under an affine transformation.

Figure shows the Bezier curve and its four control points. As shown in the figure, Bezier begins at the first control point and ends at the fourth control point.

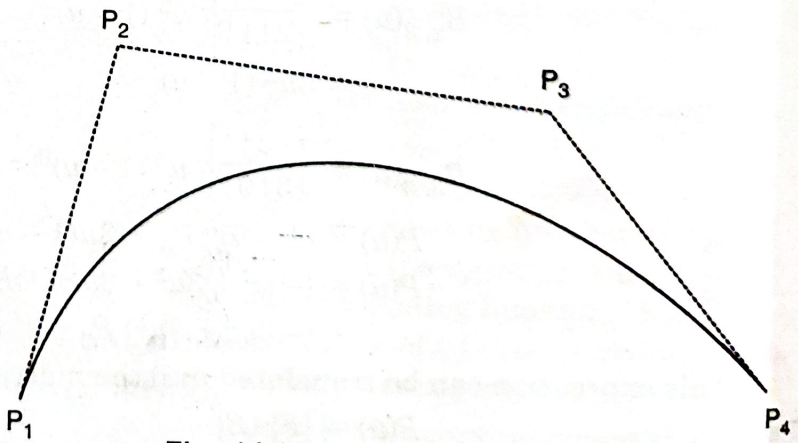



**Fig. 14.12** A cubic Bezier spline.

This means that if we want to connect two Bezier curves, we have to make the first control point of the second Bezier curve match the last control point of the first curve. We can also observe that at the start of the curve, the curve is tangent to the line connecting first and second control points. Similarly at the end of curve, the curve is tangent to the line connecting the third and fourth control point. This means that, to join two Bezier curves smoothly we have to place the third and the fourth control points of the first curve on the same line specified by the first and the second control points of the second curve.

**EXAMPLE 14.1**

**Construct the Bezier curve of order 3 and with 4 polygon vertices $A(1, 1)$, $B(2, 3)$, $C(4, 3)$ and $D(6, 4)$.**

**Solution :** The equation for the Bezier curve is given as

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2 (1-u) P_3 + u^3 P_4$$

for $0 \le u \le 1$

where $P(u)$ is the point on the curve $P_1, P_2, P_3, P_4$.

Let us take $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

$\therefore$ $P(0) = P_1 = (1, 1)$

$\therefore$
$$P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 P_1 + 3\frac{1}{4}\left(1-\frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2 \left(1-\frac{1}{4}\right) P_3 + \left(\frac{1}{4}\right)^3 P_4$$

$$= \frac{27}{64}(1, 1) + \frac{27}{64}(2, 3) + \frac{9}{64}(4, 3) + \frac{1}{64}(6, 4)$$

$$= \left[\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 6,\ \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 4\right]$$

$$= \left[\frac{123}{64}, \frac{139}{64}\right]$$

$$= (1.9218, 2.1718)$$

$$P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 P_1 + 3\frac{1}{2}\left(1-\frac{1}{2}\right)^2 P_2 + 3\left(\frac{1}{2}\right)^2 \left(1-\frac{1}{2}\right) P_3 + \left(\frac{1}{2}\right)^3 P_4$$

$$= \frac{1}{8}(1, 1) + \frac{3}{8}(2, 3) + \frac{3}{8}(4, 3) + \frac{1}{8}(6, 4)$$

$$= \left[\frac{1}{8}\times 1 + \frac{3}{8}\times 2 + \frac{3}{8}\times 4 + \frac{1}{8}\times 6,\ \frac{1}{8}\times 1 + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}\times 4\right]$$

$$= \left[\frac{25}{8}, \frac{23}{8}\right]$$

$$= (3.125,\ 2.875)$$

$\therefore$
$$P\left(\frac{3}{4}\right) = \left(1-\frac{3}{4}\right)^3 P_1 + 3\frac{3}{4}\left(1-\frac{3}{4}\right)^2 P_2 + 3\left(\frac{3}{4}\right)^2 \left(1-\frac{3}{4}\right) P_3 + \left(\frac{3}{4}\right)^3 P_4$$

$$= \frac{1}{64}P_1 + \frac{9}{64}P_2 + \frac{27}{64}P_3 + \frac{27}{64}P_4$$

$$= \frac{1}{64}(1, 1) + \frac{9}{64}(2, 3) + \frac{27}{64}(4, 3) + \frac{27}{64}(6, 4)$$

$$= \left[\frac{1}{64}\times 1+\frac{9}{64}\times 2+\frac{27}{64}\times 4+\frac{27}{64}\times 6, \frac{1}{64}\times 1+\frac{9}{64}\times 3+\frac{27}{64}\times 3+\frac{27}{64}\times 4\right]$$

$$= \left[\frac{289}{64}, \frac{217}{64}\right] = (4.5156,\ 3.375)$$

$$P(1) = P_3 = (6, 4)$$

The Fig. 14.13, shows the calculated points of the Bezier curve and curve passing throu it.

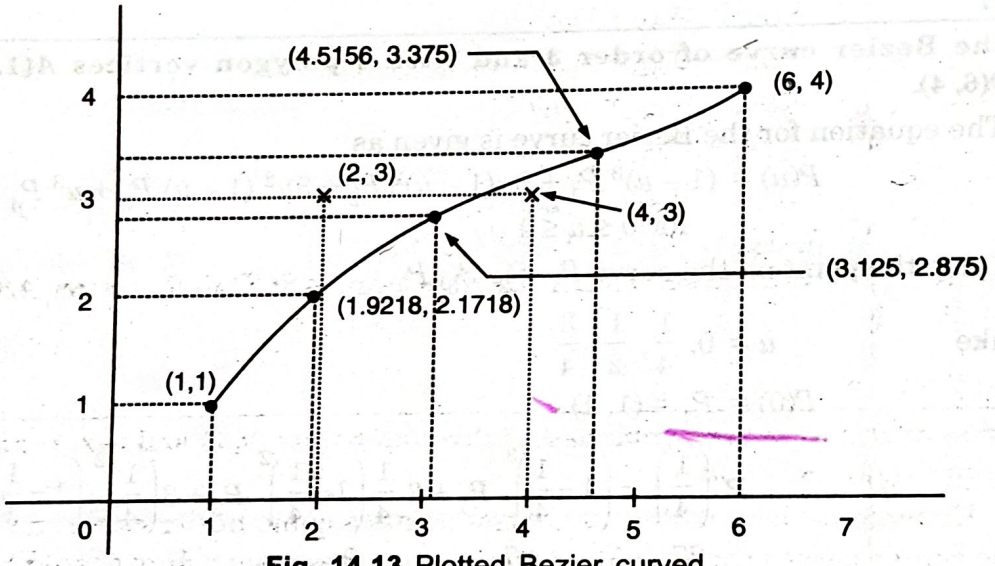



**Fig. 14.13** Plotted Bezier curved.

### Mid-point Approach

Another approach to construct the Bezier curve is called midpoint approach. In this approach the Bezier curve can be constructed simply by taking midpoints. In midpoint approach midpoints of the lines connecting four control points (A, B, C, D) are determined (AB, BC, CD). These midpoints are connected by line segments and their midpoints ABC and BCD are determined. Finally these two midpoints are connected by line segments and its midpoint ABCD is determined.

The point *ABCD* on the Bezier curve divides the original curve into two sections. This makes the points *A*, *AB*, *ABC* and *ABCD* are the control points for the first section and the points *ABCD, BCD, CD* and *D* are the control points for the second section. By considering two sections separately we can get two more sections for each separate section *i.e.*, the original Bezier curve gets divided into four different curves. This process can be repeated to split the curve into smaller sections until we have sections so short that they can be replaced by straight lines or even until the sections are not bigger than individual pixels.

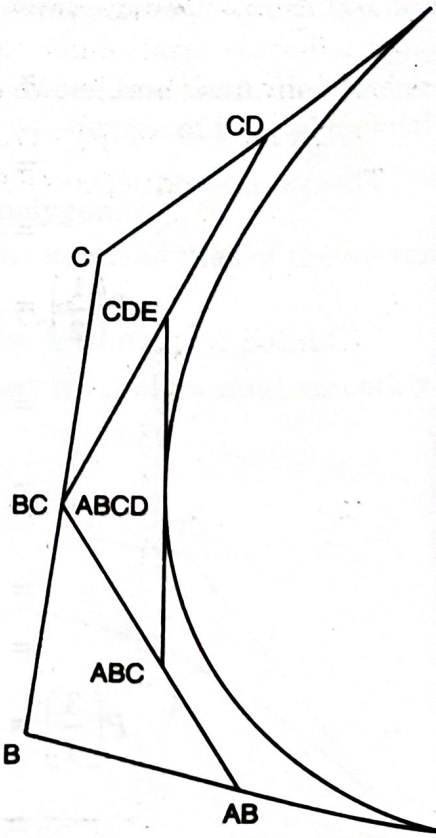



**Fig. 14.14** Subdivision of a Bezier spline.

### Algorithm

1. Get four control points say $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$, $D(x_D, y_D)$

2. Divide the curve represented by points $A$, $B$, $C$ and $D$ in two sections.

$$x_{AB} = (x_A + x_B)/2$$
$$y_{AB} = (y_A + y_B)/2$$
$$x_{BC} = (x_B + x_C)/2$$
$$y_{BC} = (y_B + y_C)/2$$
$$x_{CD} = (x_C + x_D)/2$$
$$y_{CD} = (y_C + y_D)/2$$
$$x_{ABC} = (x_{AB} + x_{BC})/2$$
$$y_{ABC} = (y_{AB} + y_{BC})/2$$
$$x_{BCD} = (x_{BC} + x_{CD})/2$$
$$y_{BCD} = (y_{BC} + y_{CD})/2$$
$$x_{ABCD} = (x_{ABC} + x_{BCD})/2$$
$$y_{ABCD} = (y_{ABC} + y_{BCD})/2$$

3. Repeat the step 2 for section $A$, $AB$, $ABC$ and $ABCD$ and section $ABCD$, $BCD$, $CD$ and $D$.

**EXAMPLE 14.2**

**Find equation of Bezier curve which passes through points (0, 0) and (– 2, 1) and is controlled through points (7, 5) and (2, 0).**

**Solution :** Using four given points, as control points a cubic Bezier curve can be defined. But as the cubic Bezier passes through (0, 0) and (– 2, 1) these two points should be considered as the end control points while the other two points (7, 5) and (2, 0) should be considered as intermediate control points. But depending on the sequence in which we consider the successive control points, we obtain two different equations of cubic Bezier curve.

(*i*) Let $P_0 = (0, 0)$; $P_1 = (7, 5)$; $P_2 = (2, 0)$; $P_3 = (-2, 1)$.

The corresponding cubic Bezier curve is given by

$$P(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 0 \\ -2 & 1 \end{bmatrix} \quad (0 \le u \le 1)$$

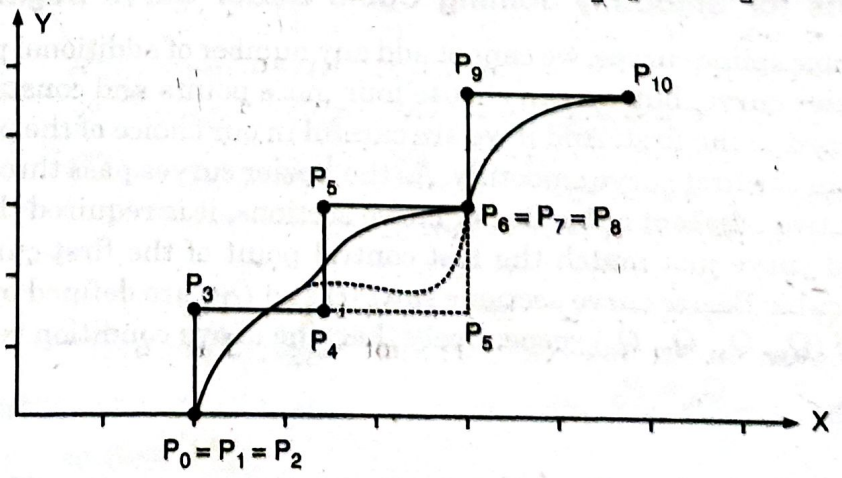



**Fig. 14.15**

$$= [u^3\ u^2\ u\ 1] \begin{bmatrix} 13 & 16 \\ -36 & -30 \\ 21 & 15 \\ 0 & 0 \end{bmatrix}$$

$$= (13u^3 - 36u^2 + 21u)\ (16u^3 - 30u^2 + 15u)$$

(*ii*) If $P_0 = (0, 0)$; $P_1 = (2, 0)$; $P_2 = (7, 5)$ and $P_3 = (-2, 1)$
then the resulting cubic Bezier curve is given by

$$P(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 7 & 5 \\ -2 & 1 \end{bmatrix} \quad (0 \le u \le 1)$$

$$= [u^3\ u^2\ u\ 1] \begin{bmatrix} -17 & -14 \\ 9 & 15 \\ 6 & 0 \\ 0 & 0 \end{bmatrix}$$

$$= [-17u^3 + 9u^2 + 6u]\ [-14u^3 + 15u^2]$$

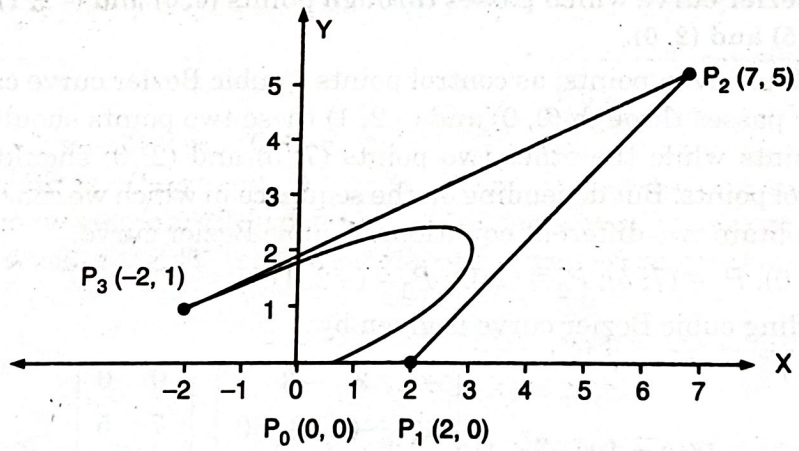



**Fig. 14.16**

### 14.6.1 Conditions for Smoothly Joining Cubic Bezier Curve Segments

Unlike other cubic spline curves, we cannot add any number of additional points and smoothly extend a cubic Bezier curve, but we can choose four more points and construct a second curve which can be attached to the first. And if we are careful in our choice of the points, we can make the second curve join the first curve smoothly. As the Bezier curves pass through the end points, in order to connect two adjacent cubic Bezier curve sections, it is required that the first control point of the second curve just match the last control point of the first curve (*i.e.*, zero-order continuity). If two cubic Bezier curve sections say $P(t)$ and $Q(t)$ are defined by the control points $(P_0, P_1, P_2, P_3)$ and $(Q_0, Q_1, Q_2, Q_3)$ respectively then the above condition requires only

$$Q_0 = P_3 \qquad \ldots (1)$$

$$\Rightarrow \quad Q_2 = P_1 + 2(Q_1 - Q_2) \quad \text{or} \quad Q_2 = P_1 + 2(P_3 + (P_3 - P_2) - P_2)$$

$$\Rightarrow \quad Q_2 = P_1 + 4(P_3 - P_2) \qquad \ldots (3) \quad [\because Q_1 = 2P_3 - P_2]$$

Thus, we see that given a Bezier curve section defined by four control points, we can extend this curve by constructing a second Bezier curve section in continuation by calculating the position of the first three control points of the new section in terms of positions of the last three control points of the previous section, using eqn. (1), (2) and (3) and arbitrarily fixing the position of the last control point (till the curve shape is satisfactory). Similarly a third section can be attached to the previous combined section and the process can be continued to form a continuous piecewise Bezier curve using multiple control points, yet cubic in nature.

**EXAMPLE 14.3**

**Draw the Bezier curve defined by the control points (2, 1), (3, 2), (5, 0) and (6, 2). By properly choosing another set of control points draw a Bezier curve such that the second curve is joined smoothly with the first curve.**

**Solution :** Let the cubic Bezier curve $P(u)$ defined by $P_0 = [2, 1]$, $P_1 = [3, 2]$, $P_2 = [5, 0]$, $P_5 = [6, 2]$ with $0 \le u \le 1$. The curve starts at $P_0$ *i.e.*, $P(u = 0)$ and ends at $P_3$ *i.e.*, $P(u = 1)$.

We have to choose another set of four control points, say $Q_0$, $Q_1$, $Q_2$, $Q_3$ such that the cubic Bezier curve say $Q(u')$, $(0 \le u' \le 1)$ defined by them joins the curve $P(u)$ at $P_3$, smoothly.

From $C^0$ continuity condition,

$$P(u = 1) = Q(u' = 0)$$

$$\Rightarrow P_3 = Q_0$$

$$\Rightarrow Q_0 = [6 \quad 2]$$

Applying $C^1$ continuity at $P_3 = Q_0$

$$P'(u = 1) = Q'(u' = 0)$$

$$\Rightarrow \quad 3(P_3 - P_2) = 3(Q_1 - Q_0)$$

$$\Rightarrow \quad Q_1 = 2P_3 - P_2 \text{ [since } Q_0 = P_3]$$

$$= 2[6 \ \ 2] - [5 \ \ 0]$$

$$= [7 \ \ 4]$$

Applying $C^2$ continuity at $P_3 = Q_0$

$$P''(u = 1) = Q''(u' = 0)$$

$$\Rightarrow \quad 6(P_3 - 2P_2 + P_1) = 6(Q_0 - 2Q_1 + Q_2)$$

$$\Rightarrow \quad Q_2 = P_1 + 4(P_3 - P_2)$$

[since $Q_0 = P_3$ and $Q_1 = 2P_3 - P_2$]

$$= [3 \ \ 2] + 4\,[6 \ \ 2] - 4[5 \ \ 0]$$

$$= [7 \ \ 10]$$

From the fact that the maximum smoothness or degree of continuity achievable at the junction point of two cubic Bezier curves is $C^2$ and the fact that the last control point of the second curve *i.e.*, $Q_3$ doesn't feature in the $C^2$ continuity condition, we are free to choose $Q_3$ without affecting continuity at junction $P_3 = Q_0$. After plotting all the points $P_0$, $P_1$, $P_2$, $P_3 = Q_0$ and $Q_1$ we choose $Q_3$ as (6, 12).

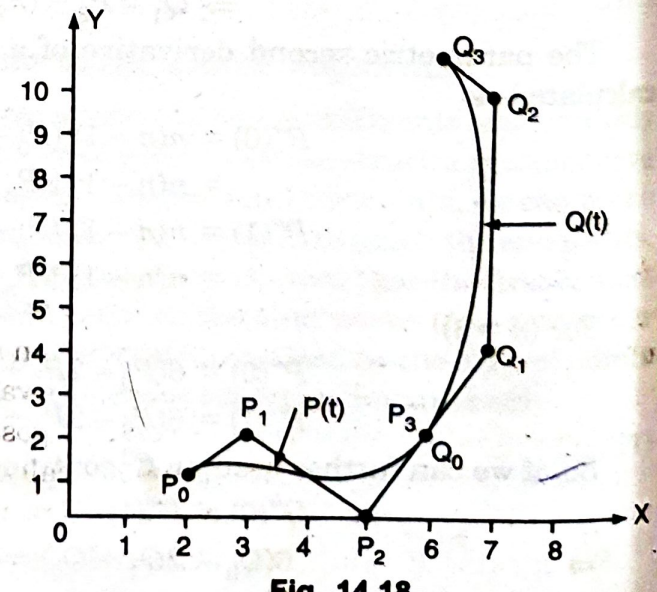



**Fig. 14.18**

## 14.7 B-SPLINE CURVES

We have seen that, a curve generated by using the vertices of a defining polygon is dependent on some interpolation or approximation scheme to establish the relationship between the curve and the polygon. This scheme is provided by the choice of basis function. The Bezier curve produced by the Bernstein basis function has a limited flexibility. First the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve. *For example*, polygon with four vertices results a cubic polynomial curve. The only way to reduce the degree of the curve is to reduce the number of vertices, and conversely the only way to increase the degree of the curve is to increase the number of vertices. The second limiting characteristic is that the value of the blending function is non zero for all parameter values over the entire curve. Due to this change in one vertex, changes the entire curve and this eliminates the ability to produce a local change with in a curve.

There is another basis function, called the *B-spline basis*, which contains the Bernstein basis as a special case. The B-spline basis is non–global. It is nonglobal because each vertex $B_i$ is associated with a unique basis function. Thus, each vertex affects the shape of the curve only over a range of parameter values where its associated basis function is nonzero. The B-spline basis also allows the order of the basis function and hence the degree of the resulting curve is independent on the number of vertices. It is possible to change the degree of the resulting curve without changing the number of vertices of the defining polygon.

If $P(u)$ be the position vectors along the curve as a function of the parameter $u$, a B-spline curve is given by

$$P(u) = \sum_{k=0}^{n} P_k B_{k,d}(u), \quad u_{\min} \le u \le u_{\max}$$

$$2 \le d \le n+1 \qquad \ldots (A)$$

where the $p_k$ are an input set of $n + 1$ control points.

There are several differences between this B-spline formation and that for Bezier splines. The range of parameter $u$ now depends on how we choose the B-spline parameters. And the B-spline blending functions $B_{k,d}$ are polynomials of degree $d - 1$, where parameter $d$ can be chosen to be any integer value in the range from 2 up to the number of control points, $n + 1$.

Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas :

$$B_{k,1}(u) = \begin{cases} 1 & \text{if } u_k \le u \le u_{k+1} \\ 0 & \text{otherwise} \end{cases} \qquad \ldots (B)$$

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d}(u)$$

where each blending function is defined over $d$ subintervals of the total range of $u$. The selected set of subinterval end-points $u_j$ is referred to as a ***knot vector***. We can choose any values for the subinterval endpoints satisfying the relation $u_j \le u_{j+1}$. Values for $u_{\min}$ and $u_{\max}$ then depend on the number of control points we select, the value we choose for parameter $d$ and how we set up the subintervals (knot vector). Since it is possible to choose the elements of the knot vector so that the denominators in the previous calculations can have a value of 0. This formulation assumes that any terms evaluated as 0/0 are to be assigned the value 0.

For a cubic B-spline, the degree $d - 1 = 3$. So $d = 4$ and hence the no. of control points $(n + 1)$ required is atleast 4 ($\because n + 1 \le d$).

Suppose we have four control points $P_0, P_1, P_2, P_3$, We can obtain four cubic B-spline blending functions.

$B_{0,4}(u)$, $B_{1,4}(u)$, $B_{2,4}(u)$ and $B_{3,4}(u)$ from equation (*B*) such that the cubic B-spline fitted to those points is given by

$$P(u) = P_0 B_{0,4}(u) + P_1 B_{1,4}(u) + P_2 B_{2,4}(u) + P_3 B_{3,4}(u) \quad \ldots (C)$$

For knot vector [0 1 2 3 4 5 6 7]

Using the recurrence relation equation (B), we obtain the expression for the four blending functions, by putting $d = 4$ and varying $k$ from 0 through 3.

$$B_{0,4}(u) = \left(\frac{u-u_0}{u_3-u_0}\right) B_{0,3}(u) + \left(\frac{u_4-u}{u_4-u_1}\right) B_{1,3}(u)$$

$$= \left(\frac{u-0}{3-0}\right) B_{0,3}(u) + \left(\frac{4-u}{4-1}\right) B_{1,3}(u)$$

$$= \left(\frac{u}{3}\right) B_{0,3}(u) + \left(\frac{4-u}{3}\right) B_{1,3}(u)$$

$$B_{1,4}(u) = \left(\frac{u-u_1}{u_4-u_1}\right) B_{1,3}(u) + \left(\frac{u_5-u}{u_5-u_2}\right) B_{2,3}(u)$$

$$= \left(\frac{u-1}{3}\right) B_{1,3}(u) + \left(\frac{5-u}{3}\right) B_{2,3}(u)$$

Similarly $$B_{2,4}(u) = \left(\frac{u-2}{3}\right) B_{2,3}(u) + \left(\frac{6-u}{3}\right) B_{3,3}(u)$$

and $$B_{3,4}(u) = \left(\frac{u-3}{3}\right) B_{3,3}(u) + \left(\frac{7-u}{3}\right) B_{4,3}(u)$$

Again putting $d = 3$ and varying $k$ from 0 through 4 in eqn. (B) we get,

$$B_{0,3}(u) = \left(\frac{u}{2}\right) B_{0,2}(u) + \left(\frac{3-u}{2}\right) B_{1,2}(u)$$

$$B_{1,3}(u) = \left(\frac{u-1}{2}\right) B_{1,2}(u) + \left(\frac{4-u}{2}\right) B_{2,2}(u)$$

$$B_{2,3}(u) = \left(\frac{u-2}{2}\right) B_{2,2}(u) + \left(\frac{5-u}{2}\right) B_{3,2}(u)$$

$$B_{3,3}(u) = \left(\frac{u-3}{2}\right) B_{3,2}(u) + \left(\frac{6-u}{2}\right) B_{4,2}(u)$$

$$B_{4,3}(u) = \left(\frac{u-4}{2}\right) B_{4,2}(u) + \left(\frac{7-u}{2}\right) B_{5,2}(u)$$

Finally putting $d = 2$ and varying $k$ from 0 through 5, we get

$$B_{0,2}(u) = (u)\, B_{0,1}(u) + (2-u)\, B_{1,1}(u)$$

$$B_{1,2}(u) = (u-1)\, B_{1,1}(u) + (3-u)\, B_{2,1}(u)$$

$$B_{2,2}(u) = (u-2)\, B_{2,1}(u) + (4-u)\, B_{3,1}(u)$$

$$B_{3,2}(u) = (u-3)\, B_{3,1}(u) + (5-u)\, B_{4,1}(u)$$

$$B_{4,2}(u) = (u-4)\, B_{4,1}(u) + (6-u)\, B_{5,1}(u)$$

$$B_{5,2}(u) = (u-5)\,B_{5,1}(u) + (7-u)\,B_{6,1}(u)$$

We know $B_{k,1}(u) = 1 \quad \text{if} \quad u_k \le u \le u_{k+1}$

else $B_{k,1}(u) = 0$

Hence by varying $k$ from 0 through 6, we get

$$B_{0,1}(u) = 1 \quad \text{if} \quad 0 \le u < 1 \quad \text{else } B_{0,1}(u) = 0$$
$$B_{1,1}(u) = 1 \quad \text{if} \quad 1 \le u < 2 \quad \text{else } B_{1,1}(u) = 0$$
$$B_{2,1}(u) = 1 \quad \text{if} \quad 2 \le u < 3 \quad \text{else } B_{2,1}(u) = 0$$
$$B_{3,1}(u) = 1 \quad \text{if} \quad 3 \le u < 4 \quad \text{else } B_{3,1}(u) = 0$$
$$B_{4,1}(u) = 1 \quad \text{if} \quad 4 \le u < 5 \quad \text{else } B_{4,1}(u) = 0$$
$$B_{5,1}(u) = 1 \quad \text{if} \quad 5 \le u < 6 \quad \text{else } B_{5,1}(u) = 0$$
$$B_{6,1}(u) = 1 \quad \text{if} \quad 6 \le u < 7 \quad \text{else } B_{6,1}(u) = 0$$

Applying these equations we get

$$B_{(0,4)}(u) = \begin{cases} \left(\frac{1}{6}\right)u^3 & \text{for } 0 \le u < 1 \\ \left(\frac{1}{6}\right)u^2(2-u) + \frac{1}{6}u(u-1)(3-u) + \frac{1}{6}(u-1)^2(4-u) & \text{for } 1 \le u < 2 \\ \left(\frac{1}{6}\right)u(3-u)^2 + \left(\frac{1}{6}\right)(4-u)(u-1)(3-u) + \frac{1}{6}(u-2)(4-u)^2 & \text{for } 2 \le u < 3 \\ \left(\frac{1}{6}\right)(4-u)^3 & \text{for } 3 \le u < 4 \end{cases}$$

$$B_{1,4}(u) = \begin{cases} \left(\frac{1}{6}\right)(u-1)^3 & \text{for } 1 \le u < 2 \\ \left(\frac{1}{6}\right)(u-1)^2(3-u) + \frac{1}{6}(u-1)(u-2)(4-u) + \frac{1}{6}(u-2)^2(5-u) & \text{for } 2 \le u < 3 \\ \left(\frac{1}{6}\right)(u-1)(4-u)^2 + \frac{1}{6}(5-u)(u-2)(4-u) + \frac{1}{6}(u-3)(5-u)^2 & \text{for } 3 \le u < 4 \\ \left(\frac{1}{6}\right)(5-u)^3 & \text{for } 4 \le u < 5 \end{cases}$$

$$B_{2,4}(u) = \begin{cases} \left(\frac{1}{6}\right)(u-2)^3 & \text{for } 2 \le u < 3 \\ \left(\frac{1}{6}\right)(u-2)^2(4-u) + \frac{1}{6}(u-2)(u-3)(5-u) + \frac{1}{6}(u-3)^2(6-u) & \text{for } 3 \le u < 4 \\ \left(\frac{1}{6}\right)(u-2) + (5-u)^2 + \frac{1}{6}(6-u)(u-3)(5-u) + \frac{1}{6}(u-4)(6-u)^2 & \text{for } 4 \le u < 5 \\ \left(\frac{1}{6}\right)(6-u)^3 & \text{for } 5 \le u < 6 \end{cases}$$

$$B_{3,4}(u) = \begin{cases} \left(\frac{1}{6}\right)(u-3)^3 & \text{for } 3 \le u < 4 \\ \left(\frac{1}{6}\right)(u-3)^2(5-u) + \frac{1}{6}(u-3)(u-4)(6-u) + \frac{1}{6}(u-4)^2(7-u) & \text{for } 4 \le u < 5 \\ \left(\frac{1}{6}\right)(u-3)(6-u)^2 + \frac{1}{6}(7-u)(u-4)(6-u) + \frac{1}{6}(u-5)(7-u)^2 & \text{for } 5 \le u < 6 \\ \left(\frac{1}{6}\right)(7-u)^3 & \text{for } 6 \le u < 7 \end{cases}$$

The hard work involved in deriving the above functions can be easily avoided by exploiting the inherent relations among them. Can you find the relation ?

Substitute $u - 1$ from $u$ in the expression of $B_{0,4}(u)$ and right shift the parameter domain by 1. We get the expressions for $B_{1,4}(u)$. Similarly $B_{2,4}(u)$ and $B_{3,4}(u)$ are obtained by successively shifting $B_{1,4}(u)$ one position to the right. Now, we plot the cubic B-spline blending functions against parameter $u$.

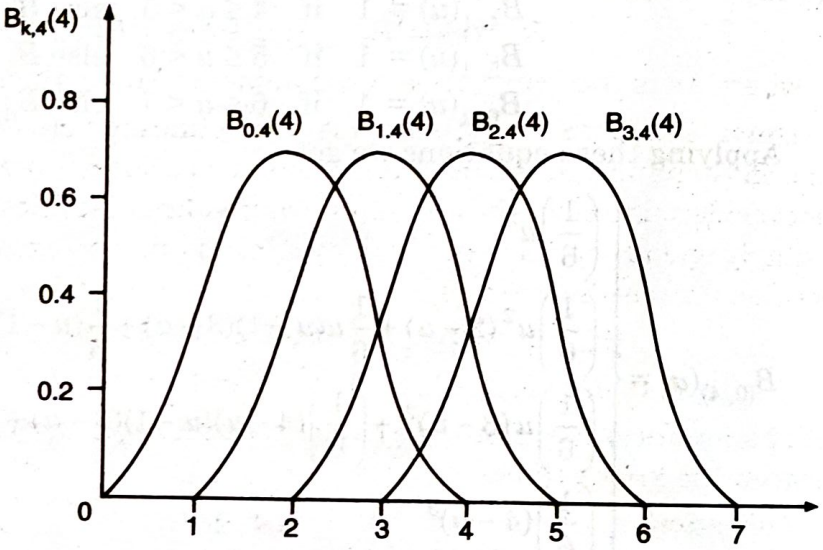



Fig. 14.19

**B-spline curves have the following properties :**

(1) The polynomial curve has degree $d - 1$ and $C^{d-2}$ continuity over the range of $u$.

(2) For $n + 1$ control points, the curve is described with $n + 1$ blending functions.

(3) Each blending function $B_{k,d}$ is defined over $d$ subintervals of the total range of $u$, starting at knot value $u_k$.

(4) The range of parameter $u$ is divided into $n + d$ sub–intervals by the $n + d + 1$ values specified in the knot vector.

(5) With knot values labeled as $(u_0, u_1, \ldots u_{n+d})$ the resulting B-spline curve is defined only in the interval from knot value $u_{d-1}$ upto knot value $u_{n+1}$.

(6) Each section of the spline curve (between two successive knot values) is influenced by $d$ control points.

(7) Any one control point can affect the shape of at most $d$ curve sections.

In addition, a B-spline curves lies within the convex hull of at most $d + 1$ control points, so that B-splines are tightly bound to the input positions. For any value of $u$ in the interval from knot value $u_{d-1}$ to $u_{n+1}$, the sum over all basis functions is 1 :

$$\sum_{k=0}^{n} B_{k,d}(u) = 1$$

Given the control-point positions and the value of parameter $d$, we then need to specify the knot values to obtain the blending functions.

There are three general classifications for knot vectors :

(a) Uniform
(b) Open-uniform
(c) Non-uniform

When the spacing between the knot values is constant, the resulting curve is called a *uniform B-spline. For example,* the integer knot vector [0 1 2 3 4 5 6 7] we have used for the 4-control point cubic B-spline.

Uniform B-splines have *periodic* blending functions. That is, for given values of $n$ and $d$, all blending functions have the same shape. Each successive blending function is simply a shifted version of the previous function.

$$B_{k,\,d}(u) = B_{k+1,\,d}(u + \Delta u) = B_{k+2,\,d}(u + 2\Delta u)$$

where $\Delta u$ is the interval between adjacent knot values.

***Open, Uniform B-splines*** class of B-splines is a cross between uniform B-splines and non-uniform B-splines. Sometimes it is treated as a special type of uniform B-spline and sometimes it is considered to be in the non–uniform B-spline classification. For the open, uniform B-splines or simply open B-splines, the knot spacing is uniform except at the ends where knot values are repeated $d$ times. *For example,*

$[0, 0, 1, 2, 3, 3]$ $\quad d = 2$ and $n = 3$

$[0, 0, 0, 0, 1, 2, 2, 2, 2]$ $\quad d = 4$ and $n = 4$

with nonuniform B-splines, we can choose multiple internal knot values and unequal spacing between the knot values.

*For example,* $[0, 1, 2, 3, 3, 4]$

$[0, 2, 2, 3, 3, 6]$

Non-unifrom B-splines provide increased flexibility in controlling a curve shape. With unequally spaced intervals in the knot vector, we obtain different shapes for the blending functions in different intervals, which can be used to adjust spline shapes. By increasing knot multiplicity, we produce subtle variations in curve shape and even introduce discontinuities. Multiple knot values also reduce the continuity by 1 for each repeat of a particular value.

## 14.8 BEZIER SURFACES AND B-SPLINE SURFACES

Two sets of orthogonal Bezier curves can be used to design an object surface by specifying by an input mesh of control points. The parametric vector function for the Bezier surface is formed as the Cartesian product of Bezier blending functions.

$$P(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,\,k}\, B_{j,\,m}(v)\, B_{k,\,n}(u)$$

where $P_{j,\,k}$ specifying the location of the $(m + 1)$ by $(n + 1)$ control points.

Bezier surfaces have the same properties as Bezier curves and they provide a convenient method for interactive design applications.

Formulation of a *B-spline surface* is similar to that for Bezier surface. We can obtain a vector point function over a B-spline surface using the Cartesian product of B-spline blending functions in the form.

$$P(u, v) = \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} P_{k_1,\,k_2}\, B_{k_1,\,d_1}(u)\, B_{k_2,\,d_2}(v)$$

where the vector values for $P_{k_1, k_2}$ specify positions of the $(n_1 + 1)$ by $(n_2 + 1)$ control points.

**EXAMPLE 14.4**

**A Bezier curve is to be drawn given the control points $P_1(40, 40)$, $P_2(10, 40)$, $P_3(60, 60)$, $P_4(60, 0)$. Calculate the coordinates of the points on the curve corresponding to the parameter $t = 0.2$, 0.4, 0.6. Draw a rough sketch of the curve and show coordinates of various points on it.**

**Solve the above problem for a cubic B-spline fitted to the control points given**

**Solution :** Any point $P(t)$ corresponding to parameter value $t$ on the Bezier curve defined by $P_1, P_2, P_3, P_4$ is given by

$$P(t) = [t^3 \; t^2 \; t \; 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 40 & 40 \\ 10 & 40 \\ 60 & 60 \\ 60 & 0 \end{pmatrix} \qquad (0 \le t \le 1)$$

$$= [t^3 \; t^2 \; t \; 1] \begin{pmatrix} -130 & -100 \\ 240 & 60 \\ -90 & 0 \\ 40 & 40 \end{pmatrix}$$

$$\Rightarrow \qquad P(t) = [(-130t^3 + 240t^2 - 90t + 40) \; (-100t^3 + 60t^2 + 40)]$$

For $t = 0.2$

$$P(t) = P(0.2) = [(-130(0.2)^3 + 240(0.2)^2 - 90(0.2) + 40) \; (-100(0.2)^2 + 60(0.2)^2 + 40)]$$

$$= [30.56 \quad 41.6]$$

Similarly $P(0.4) = [34.08 \quad 43.2]$

$P(0.6) = [44.32 \quad 40]$

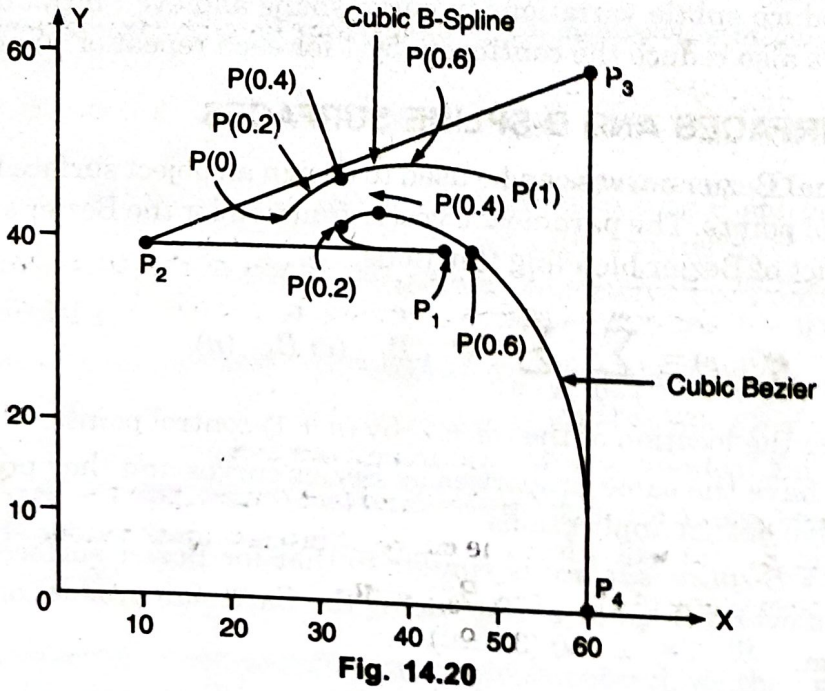



Fig. 14.20

The equation of a periodic cubic B-spline with same set of control points $P_1, P_2, P_3, P_4$ is given by

$$P(t) = [t^3 \; t^2 \; t \; 1] \left(\frac{1}{6}\right) \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} 40 & 40 \\ 10 & 40 \\ 60 & 60 \\ 60 & 0 \end{pmatrix} \qquad (0 \le t \le 1)$$

$$= [t^3 \; t^2 \; t \; 1] \frac{1}{6} \begin{pmatrix} -130 & -100 \\ 240 & 60 \\ 60 & 60 \\ 140 & 260 \end{pmatrix}$$

$$= \left[\frac{1}{6}(-130t^3 + 240t^2 + 60t + 140) \; \frac{1}{6}(-100t^3 + 60t^2 + 60t + 260)\right]$$

$\Rightarrow$ $P(0.2) = [26.76 \quad 45.60]$

$P(0.4) = [32.35 \quad 47.87]$

$P(0.6) = [39.05 \quad 49.33]$

$P(0) = [23.33 \quad 43.33]$

$P(1) = [51.67 \quad 46.67]$

**EXAMPLE 14.5**

**The Bezier blending function is given as**

$$B_{k,n}(u) = C(n, k)\, u^k (1-u)^{n-k}$$

where $$C(n, k) = \frac{n!}{k!(n-k)!}$$

**Generate an expression for the blending functions of a cubic bezier curve. Plot rough sketch of the blending function generated.**

**Solution :** Here $n = 3$

So this curve contain four control points hence four blending function such as

$B_{0,3}(u) = (1-u)^3$ since ${}^3C_0 = 1$

$B_{1,3}(u) = 3u(1-u)^2$ since ${}^3C_1 = 3$

$B_{2,3}(u) = 3u^2(1-u)$ since ${}^3C_2 = 3$

$B_{3,3}(u) = u^3$ since ${}^3C_3 = 1$

Because the blending function determines the shape of the curve for the values of the parameter $u$. When $u = 0$, the only non-zero blending function is $B_{0,3}$ which is equal to 1, *i.e.*,

when $$B_{0,3}(0) = (1-0)^3 = 1$$

$u = 1$ then

$B_{3,3}(1) = (1)^3 = 1$ and others are zero.

Since Bezier curve passes through the first and last control point so, the other two blending functions $B_{1,3}(u)$ and $B_{2,3}(u)$ will affect the curve at intermediate values of $u$.

Now, the maximum value of $B_{1,3}$ and $B_{2,3}$ at $u$.

$$B'_{1,3} = (1-u)(3-9u)$$

For $$B'_{1,3} = 0, \quad u = 1, \frac{1}{3}$$

So, $B_{1,3}$ is maximum at $u = \frac{1}{3}$ and $B_{2,3}$ is maximum at $u = \frac{2}{3}$

Because there is not a single blending function which non-zero over the entire range of So, Bezier curve does not has local control of the curve shape *i.e.*, repositioning of any cont point, affects the entire curve.

The four blending functions for cubic Bezier curve are as follows :

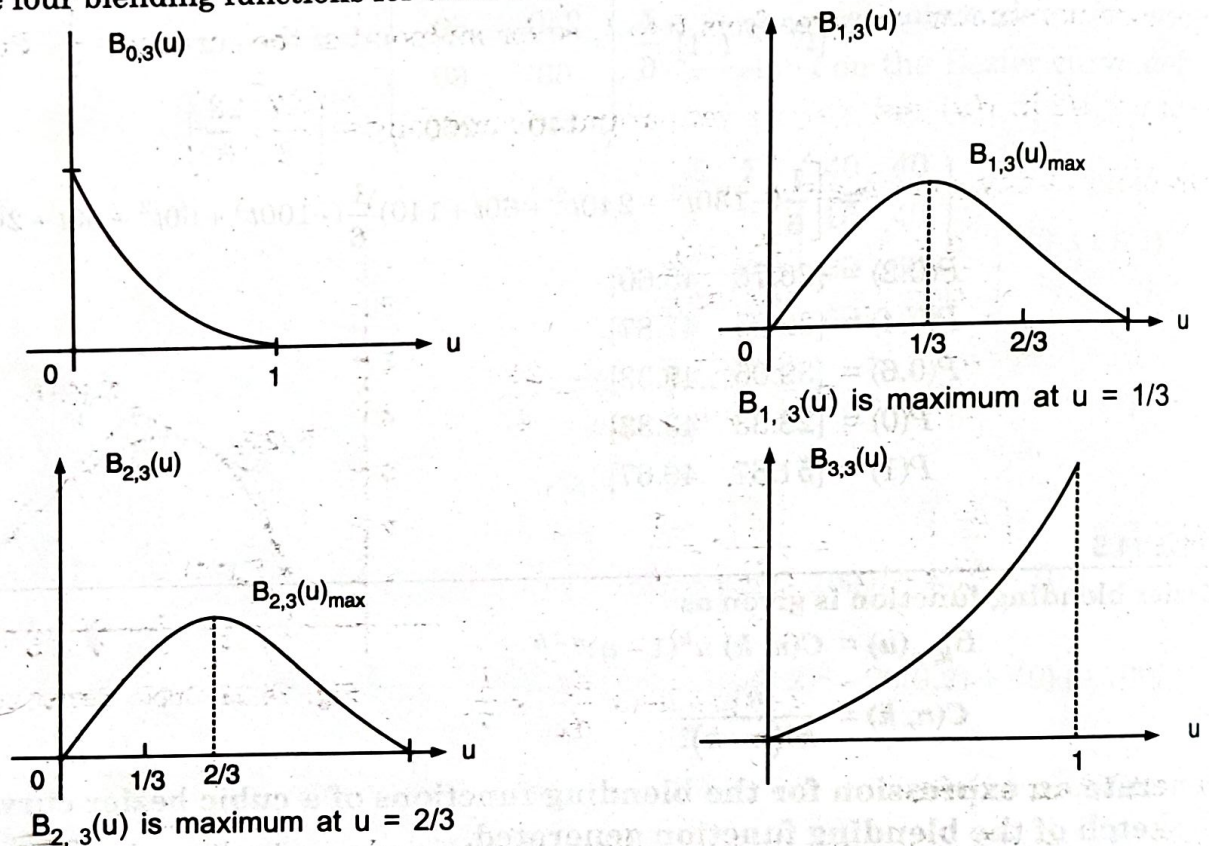



**Fig. 14.21**

**EXAMPLE 14.6**

**What do you understand by Bezier curve ? A cubic curve is defined by the points (1, 1), (2, 3) ( 4) and (6, 1). Calculate the co-ordinates of parametric midpoint of this curve and verify th its gradient *dy/dx* is 1/7 at this point. Sketch the curve ?**

**Solution :** Here four control points. So there will be four blending function *i.e.*,

$$B_{0,3}(u) = (1-u)^3$$
$$B_{1,3}(u) = 3u(1-u)^2$$
$$B_{2,3}(u) = 3u^2(1-u)$$
$$B_{3,3}(u) = u^3$$

The $x$-component of Bezier curve

$$x(u) = x_0 B_{0,3}(u) + x_1 B_{1,3}(u) + x_2 B_{2,3}(u) + x_3 B_{3,3}(u)$$
$$x(u) = (1-u)^3 + 6u(1-u)^2 + 12u^2(1-u) + 6u^3 \quad \ldots$$

Similarly $y$-component of Bezier curve

$$y(u) = (1-u)^3 + 9u(1-u)^2 + 12u^2(1-u) + u^3 \quad \ldots$$

Differentiating eqn. (1) w.r.t $u$.

$$\frac{dx}{du} = -3(1-u)^2 - 12u(1-u) + 6(1-u)^2 - 12u^2 + 24u(1-u) + 18u^2$$

Similarly differentiating eqn (2) w.r.t $u$.

$$\frac{dy}{du} = -3(1-u)^2\,u(1-u) + 9(1-u)^2 - 12u^2 + 24(1-u)u + 3u^2$$

Because parameter $u$ ranges from 0 to 1, so for midpoint of the curve $u = \frac{1}{2}$. Putting the value of $u = 1/2$ in $x(u)$ and $y(u)$, we get parametric midpoint $= \left(\frac{27}{8}, \frac{23}{8}\right)$.

Gradient at any point is given by

$$\left(\frac{dy}{dx}\right)_u = \frac{(dy/du)_u}{(dx/du)_u}$$

$$\left(\frac{dy}{du}\right)_{u=1/2} = \frac{3}{4}\left(\frac{dx}{du}\right)_{u=1/2} = \frac{21}{4}$$

So,
$$\left(\frac{dy}{dx}\right)_{u=1/2} = \frac{\left(\frac{3}{4}\right)}{\left(\frac{21}{4}\right)} = \frac{1}{7}$$

*i.e.*, gradient $\frac{dy}{dx}$ is $\frac{1}{7}$.

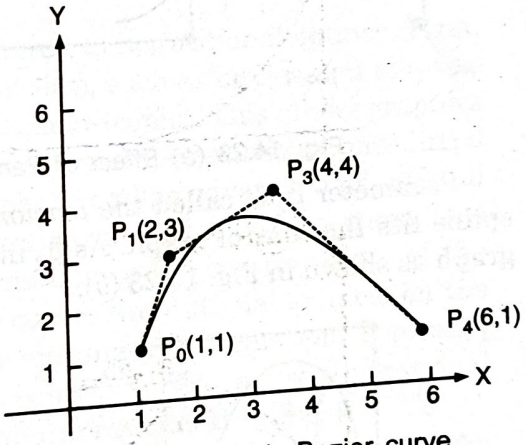



**Fig. 14.22** Cubic Bezier curve.

## 14.9 BETA-SPLINES

A generalization of B-splines are the beta-splines, also referred to as β-splines, that are formulated by imposing geometric continuity conditions on the first and second parametric derivatives. The continuity parameters for beta-splines are called β parameters.

**Beta-Spline Continuity Conditions**

For a specified knot vector, we can designate the spline sections to the left and right of a particular knot $u_j$ with the position vectors $P_{j-1}(u)$ and $P_j(u)$.

Zero-order continuity (*positional continuity*), $G^0$, at $u_j$ is obtained by requiring

$$P_{j-1}(u_j) = P_j(u_j)$$

First-order continuity (*unit tangent continuity*), $G^1$, is obtained by requiring tangent vectors to be proportional :

$$\beta_1 P'_{j-1}(u_j) = P'_j(u_j), \qquad \beta_1 > 0$$

Here, parametric first derivatives are proportional, and the unit tangent vectors are continuous across the knot.

Second-order continuity (*curvature vector continuity*), $G^2$, is imposed with the condition

$$\beta_1^2 P''_{j-1}(u_j) = \beta_2 P'_{j-1}(u_j) = P''_j(u_j)$$

where $\beta_2$ can be assigned any real number, and $\beta_1 > 0$. The curvature vector provides a measure of the amount of bending of the curve at position $u_j$. When $\beta_1 = 1$ and $\beta_2 = 0$, beta-splines reduce to B-splines.