K.K. COLLEGE OF ENGINEERING & MANAGEMENT

Branch: Mechanical Engineering

Semester: VI

COMPUTER AIDED DESIGN

COMPUTER AIDED DESIGN

Course Code – ME604

Objectives:

- To provide an overview of how computers can be utilized in mechanical component design

Contents:

Module- I

Fundamentals of Computer Graphics- Product cycle, sequential and concurrent engineering, Computer Aided Design, CAD system architecture, computer graphics, Coordinate systems, 2D and 3D transformations, viewing transformation **(8 hrs)**

Module- II

Geometric Modelling- straight line, representation of curves, Hermite curves, Bezier curves, B-spline curves, rational curves **(5 hrs)**

Module- III

Techniques of surface modelling, plane surface, cylindrical surface, surface of revolution, surface patch, Coons and bicubic patches, Bezier and B-spline surfaces **(6 hrs)**

Module- IV

Fundamental of solid design, parametric space of a solid, surface and curves in a solid, Solid modelling techniques, CSG and B-rep. **(6 hrs)**

Module- V

Visual realism- hidden line-surface-solid removal algorithms, shading, colouring, computer animation **(5 hrs)**

Module- VI

Assembly of parts- assembly modelling, interferences of positions and orientation, tolerance analysis, mass property calculations, mechanism simulation and interefence checking CAD standards- Graphical Kernel System (GKS), standards for vexchange images, Open Graphics Library (OpenGL), Data exchange standards- IGES, STEP, CALS etc., Communication standards **(12 hrs)**

1.2 Product Life Cycle

Figure 1.1 shows the life cycle of a typical product. The product begins with a need which is identified based on customers' and markets' demands. The product goes through two main processes from inception to a finished product: the design process and the manufacturing process. Synthesis and analysis are the two main subprocesses of the design process. The philosophy, functionality, and uniqueness of the product are all determined during synthesis. During synthesis, a design takes the form of sketches and layout drawings that show the relationship among the various product parts. These sketches and drawings can be created using a CAD/CAM system or simply hand-drawn on paper. They are used during brainstorming discussions among various design teams and for presentation purposes.

The analysis subprocess begins with an attempt to put the conceptual design into the context of engineering sciences to evaluate the performance of the expected product. This requires design modeling and simulation. An important aspect of analysis is the "what if" questions that help us to eliminate multiple design choices and find the best solution to each design problem. The outcome of analysis is the design documentation in the form of engineering drawings (also known as blueprints).

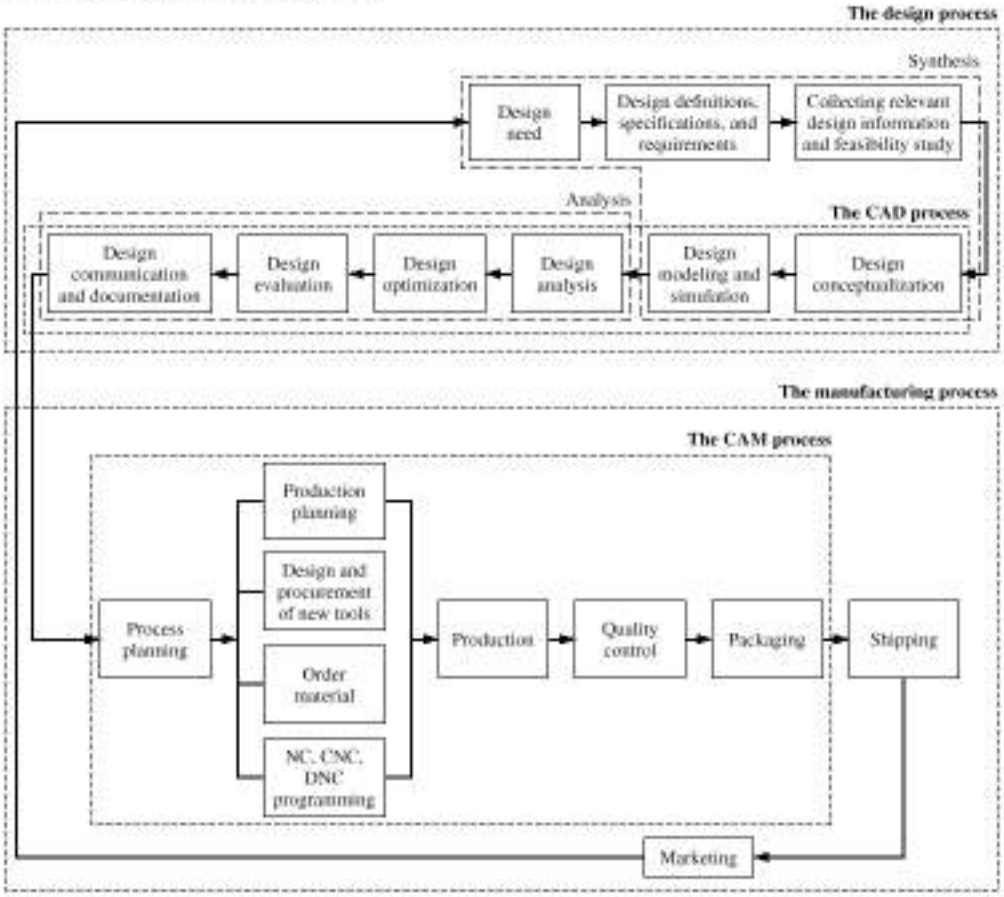


Figure 1.1 A typical product cycle.

1.3.1 Graphics Systems and Hardware

Hardware comprises the *input*, and *display* or *output devices*. Numerous types of graphics systems are in use; those that model one-to-many interaction and others that allow one-to-one interface at a given time. *Mainframe-based systems* use a large mainframe computer on which the software, which is usually a huge code requiring large space for storage, is installed. The system is networked to many designer stations on time-sharing basis with display unit and input devices for each designer. With this setting, intricate assemblies of engineering components, say an aircraft, requiring many human designers can be handled. *Minicomputer* or *Workstation* based systems are smaller in scale than the Mainframe systems with a limited number (one or more) of display and input devices. Both systems employ one-to-many interface wherein more than one designer can interact with a computer. On the contrary, *Microcomputer* (PC) based systems allow only one-to-one interaction at a time. Between the Mainframe, Workstation and PC based systems, the Workstation based system offers advantages of distributed computing and networking potential with lower cost compared with the mainframes.

1.3.2 Input Devices

Keyboard and *mouse* are the primary input devices. In a more involved environment, digitizers, joysticks and tablets are also used. Trackballs and input dials are used to produce complex models. Data gloves, image scanners, touch screens and light pens are some other input devices. A keyboard is used for submitting alphanumeric input, three-dimensional coordinates, and other non-graphic data in 'text' form. A mouse is a small hand held pointing device used to control the position of the cursor on the screen. Below the mouse is a ball. When the mouse is moved on a surface, the amount and direction of movement of the cursor is proportional to that of the mouse. In optical mouse, an optical sensor moving on a special mouse pad having orthogonal grids detects the movements. There are push buttons on top of the mouse beneath the fingers for signaling the execution of an operation, for selecting an object created on the screen within a rectangular area, for making a selection from the pulled down menu, for dragging an object from one part of the screen to other, or for creating drawings and dimensioning. It is an important device used to expedite the drawing operations. A special z-mouse for CAD, animation and virtual reality includes three buttons, a thumb-wheel and a track-ball on top. It gives six degrees of freedom for spatial positioning in x-y-z directions. The z-mouse is used for rotating the object around a desired axis, moving and navigating the viewing position (observer's eye) and the object through a three-dimensional scene.

Trackballs, *space-balls* and *joysticks* are other devices used to create two and three-dimensional drawings with ease. Trackball is a 2-D positioning device whereas space-ball is used for the same in 3-D. A joystick has a vertical lever sticking out of a base box and is used to navigate the screen cursor.

Digitizers are used to create drawings by clicking input coordinates while holding the device over a given 2-D paper drawing. Maps and boundaries in a survey map, for example, can be digitized to create a computer map. *Touch panels* and *light pens* are input devices interacting directly with the computer screen. With touch panels, one can select an area on the screen and observe the details pertaining to that area. They use infrared light emitting diodes (LEDs) along vertical and horizontal edges of the screen, and go into action due to an interruption of the beam when a finger is held closer to the screen. Pencil shaped *light pens* are used to select screen position by detecting the light from the screen. They are sensitive to the short burst of light emitted from the phosphor coating as the electron beam hits the screen. *Scanners* are used to digitize and input a two-dimensional photographic data or text for computer storage or processing. The gradations of the boundaries, gray scale or color of the picture is stored as data arrays which can be used to edit, modify, crop, rotate or scale to enhance and make suitable changes in the image by software designed using geometric transformations and image processing techniques.

1.3.3 Display and Output Devices

Three types of display devices are in use: Cathode ray tube (*CRT*), Plasma Panel Display (PPD) and Liquid Crystal Display (LCD). CRT is a popular display device in use for its low cost and high-resolution color display capabilities. It is a glass tube with a front rectangular panel (screen) and a cylindrical rear tube. A cathode ray gun, when electrically heated, gives out a stream of electrons, which are then focused on the screen by means of positively charged electron-focusing lenses. The position of the focused point is controlled by orthogonal (horizontally and vertically deflecting) set of amplifiers arranged in parallel to the path of the electron beam. A popular method of CRT display is the Raster Scan. In raster scan, the entire screen is divided into a matrix of picture cells called *pixels*. The distance between pixel centers is about 0.25 mm. The total number of pixel sets is usually referred to as *resolution*. Commonly used CRTs are those with resolution of 640×480 (VGA), 1024×768 (XGA) and 1280×1024 (SXGA). With higher resolution, the picture quality is much sharper. As the focused electron beam strikes a pixel, the latter emits light, i.e. the pixel is 'on' and it becomes bright for a small duration of time. The electron beam is made to scan the entire screen line-by-line from top to bottom (525 horizontal lines in American system and 625 lines in European system) at 63.5 microseconds per scan line. The beam keeps on retracing the path. The *refresh rate* is 60Hz, implying that the screen is completely scanned in $1/60^{th}$ of a second (for European system, it is $1/50^{th}$ of a second). In a black and white display, if the pixel intensity is '0', the pixel appears black, and when '1', the pixel is bright. As the electron beam scans through the entire screen, it switches off those pixels which are supposed to be black thus creating a pattern on the screen. For the electron beam to know precisely which pixels are to be kept 'off' during scans, a *frame buffer* is used that is a hardware programmable memory. At least one memory bit ('0' or '1') is needed for each pixel, and there are as many bits allocated in the memory as the number of pixels on display. The entire memory required for displaying all the pixels is called a *bit plane* of the frame buffer.

One bit plane would create only a 'black' and 'white' image, but for a realistic picture, one would need *gray levels* or shades between black and white as well. To control the intensity (or shade) of a pixel one has to use a number of bit planes in a frame buffer. For example, if one uses 3 bit planes in single frame buffer, one can create 8 (or 2^3) combinations of intensity levels (or shades) for the same pixel- 000 (black)-001-010- 011-100-101-110-111(white). The intermediate values will control the intensity of the electron beam falling on the pixel. To have an idea about the amount of memory required for a black and white display with 256×256 (or 2^{16}) pixels, every bit plane will require a memory of $2^{16} = 65,536$ bits. If there are 3 bit planes to control the gray levels, the memory required will be 1,96,608 bits! Since memory is a digital device and the raster action is analog, one needs digital-to-analog converters (DAC). A DAC takes the signal from the frame buffer and produces an equivalent analog signal to operate the electron gun in the CRT.

For *color display*, all colors are generated by a proper combination of 3 basic colors, viz. red, green, and blue. If we assign '0' and '1' to each color in the order given, we can generate 8 colors: black (000), red (100), green (010), blue (001), yellow (110), cyan (011), magenta (101) and white (111). The frame buffer requires a minimum of 3 bit planes—one for each RGB color; this can generate 8 different colors. If more colors are desired, one needs to increase the number of bit planes for each color. For example, if each of the RGB colors has 8 bit planes (a total of 24 bit planes in the frame buffer with three 8-bit DAC), the total number of colors available for picture display would be $2^{24} = 1,67,77,216$! To further enhance the color capabilities, each 8-bit DAC is connected to a color look up memory table. Various methods are employed to decrease the access and display time and enhance the picture sharpness.

2.4 Coordinate Systems

Three types of coordinate systems are needed in order to input, store, and display model geometry and graphics. These are Model Coordinate System (MCS), Working Coordinate System (WCS), and Screen Coordinate System (SCS), respectively. Other names for MCS are database, master, or world coordinate system. Another name for SCS is device coordinate system. Throughout this book, MCS, WCS, and SCS are used. We have covered each briefly in Tutorial 1.8.1.

2.4.1 Model Coordinate System

The **model coordinate system** is defined as the reference space of the model with respect to which all the model geometrical data is stored. It is a cartesian system which forms the default coordinate system used by a particular software program. The *X*, *Y*, and *Z* axes of the MCS can be displayed on the computer screen. The origin of the MCS can be arbitrarily chosen by the user while its orientation is established by the software. The three default sketch planes of a CAD/CAM system define the three planes of the MCS, and their intersection point is the MCS origin. When a CAD designer begins sketching, the origin becomes a corner point of the profile being sketched. The sketch plane defines the orientation of the profile in the model 3D space. This is how we attach the MCS to a geometric model.

In order for the user to communicate properly and effectively with a model database, the relationships between the MCS orthogonal (sketch) planes and the model views must be understood by the user. Typically, the software chooses one of two possible orientations of the MCS in space. As shown in Figure 2.4*a*, the *XY* plane is the horizontal plane and defines the model top view. The front and right side views are consequently defined by the *XZ* and *YZ* planes, respectively. Figure 2.4*b* shows the other possible orientation of the MCS where the *XY* plane is vertical and defines the model front view. As a result, the *XZ* and the *YZ* planes define the top and the right side views, respectively.

Existing CAD/CAM software uses the MCS as the default WCS (see Section 2.4.2). In both orientations, the *XY* plane is the default construction (sketch) plane. If the user utilizes such a plane, the first face of a model to be constructed becomes the top or front view, depending on which MCS is used.

2.4.2 Working Coordinate System

It is often convenient in the development of geometric models and the input of geometric data to refer to an auxiliary coordinate system instead of the MCS. This is usually useful when a desired plane (face) of construction is not easily defined as one of the MCS orthogonal planes, as in the case of inclined faces of a model (see Example 2.1). The user can define a cartesian coordinate system whose XY plane is coincident with the desired plane of construction. That system is the Working Coordinate System, WCS. It is a convenient user-defined system that facilitates geometric construction. It can be established at any position and orientation in space that the user desires. While the user can input data in reference to the WCS, the CAD software performs the necessary transformations to the MCS before storing the data. The ability to use two separate coordinate systems within the same model database in relation to one another gives the user great flexibility. Some commercial software refers to the WCS as is; Unigraphics offers an example. Other software refers to it as a sketch plane (Pro/E and SolidWorks) or construction plane.

A WCS requires three noncollinear points to define its XY plane. The first defines the origin, the first and the second define the X axis, and the third point with the first define the Y axis. The Z axis is determined as the cross product of the two unit vectors in the directions defined by the lines connecting the first and the second (the X axis), and the first and the third points (Y axis). We will use the subscript w to distinguish the WCS axes from those of the MCS. The X_wY_w plane becomes the active sketch (working) or construction plane if the user defines a WCS. In this case, the WCS and its corresponding X_wY_w plane override the MCS and the default sketch plane, respectively. As a matter of fact, the MCS with its default sketch plane is the default WCS with its X_wY_w plane. All CAD/CAM software packages provide users with three standard WCSs (sketch planes) that correspond to the three standard views: `Front`, `Top`, and `Right` sides. The user can define other WCSs or sketch planes.

2.4.3 Screen Coordinate System

In contrast to the MCS and WCS, the **screen coordinate system** (SCS) is defined as a 2D device-dependent coordinate system whose origin is usually located at the lower left corner of the graphics display, as shown in Figure 2.6. The physical dimensions of a device screen (aspect ratio) and the type of device (raster) determine the range of the SCS. The SCS is mostly used in view-related clicks such as definitions of view origin and window or clicking a view to select it for graphics operations.

Figure 2.6 Typical SCS.

A 1024 1024 display has an SCS with a range of (0,0) to (1024,1024). The center of the screen has coordinates of (512,512). This SCS is used by the CAD/CAM software to display relevant graphics by converting directly from MCS coordinates to SCS (physical device) coordinates. A normalized SCS can also be utilized. The range of the SCS can be chosen from (0,0) to (1,1). Such representation can be translated by device-dependent codes to the appropriate physical device coordinates. The third method of defining an SCS is by using the drawing size that the user chooses. If a size A drawing is chosen, the range of the SCS becomes (0,0) to (11,8.5) while size B produces the range (0,0) to (17,11). The rationale behind this method stems from the conventional drawing board so that the drafting paper is represented by the device screen.

A transformation operation from MCS coordinates to SCS coordinates is performed by the software before displaying the model views and graphics. Typically, for a geometric model, there is a data structure to store its geometric data (relative to MCS), and a display file to store its display data (relative to SCS).

2.2.1 Rotation in Two-Dimensions

Consider a rigid body S packed with points $\boldsymbol{P}_i$ $(i = 1, \ldots, n)$ and let a point $\boldsymbol{P}_j(x_j, y_j)$ on S be rotated about the z-axis to $\boldsymbol{P}_j^*(x_j^*, y_j^*)$ by an angle θ. From Figure 2.3, it can be observed that

$$x_j^* = l\cos(\theta+\alpha) = l\cos\alpha\cos\theta - l\sin\alpha\sin\theta$$
$$= x_j\cos\theta - y_j\sin\theta$$

and $$y_j^* = l\sin(\theta+\alpha) = l\cos\alpha\sin\theta + l\sin\alpha\cos\theta$$
$$= x_j\sin\theta + y_j\cos\theta$$

Or in matrix form

$$\begin{bmatrix} x_j^* \\ y_j^* \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x_j \\ y_j \end{bmatrix} \Rightarrow \mathbf{P}_j^* = \mathbf{R}\mathbf{P}_j \quad (2.1)$$

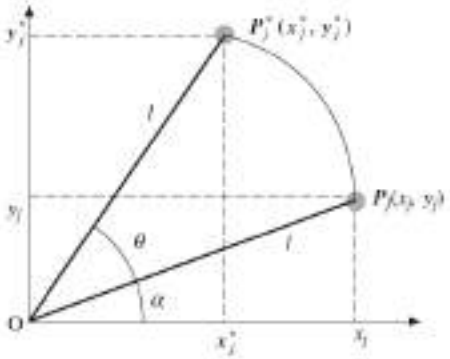


Figure 2.3 Rotation in a plane

where $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is the two-dimensional *rotation matrix*. For S to be rotated by an angle θ, transformation in Eq. (2.1) must be performed simultaneously for all points $\boldsymbol{P}_i$ $(i = 1, \ldots, n)$ such that the entire rigid body reaches the new destination S^*.

2.2.2 Translation in Two-Dimensions: Homogeneous Coordinates

For a rigid body S to be translated along a vector $\mathbf{v}$ such that each point of S shifts by (p, q),

$$x_j^* = x_j + p, \quad y_j^* = y_j + q \Rightarrow \begin{bmatrix} x_j^* \\ y_j^* \end{bmatrix} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} + \begin{bmatrix} p \\ q \end{bmatrix} \Rightarrow \mathbf{P}_j^* = \mathbf{P}_j + \mathbf{v}$$

2.2.3 Combined Rotation and Translation

Consider a point $P(x, y, 1)$ in the x-y plane to be rotated by an angle θ about the z-axis to a position $P_1(x_1, y_1, 1)$ followed by a translation by $\mathbf{v}(p, q)$ to a position $P_2(x_2, y_2, 1)$. Using Eqs. (2.3) and (2.4), we may write

$$\mathbf{P}_1 = \mathbf{RP}, \quad \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

and $$\mathbf{P}_2 = \mathbf{TP}_1, \quad \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{TRP}$$

Thus, $$\mathbf{P}_2 = \begin{bmatrix} \cos\theta & -\sin\theta & p \\ \sin\theta & \cos\theta & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

On the contrary, if translation by $\mathbf{v}$ is followed by rotation about the z-axis by an angle θ to reach P_2^*, then

$$\mathbf{P}_2^* = \mathbf{RTP} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & p\cos\theta - q\sin\theta \\ \sin\theta & \cos\theta & p\sin\theta + q\sin\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.6}$$

2.2.4 Rotation of a Point Q (x_q, y_q, 1) about a Point P (p, q, 1)

Since the rotation matrix **R** about the z-axis and translation matrix **T** in the x-y plane are known from Eqs. (2.4) and (2.3) respectively, rotation of Q about P can be regarded as translating P to coincide with the origin, followed by rotation about the z-axis by an angle θ, and lastly, placing P back to its original position (Figure 2.6). These transformations can be concatenated as

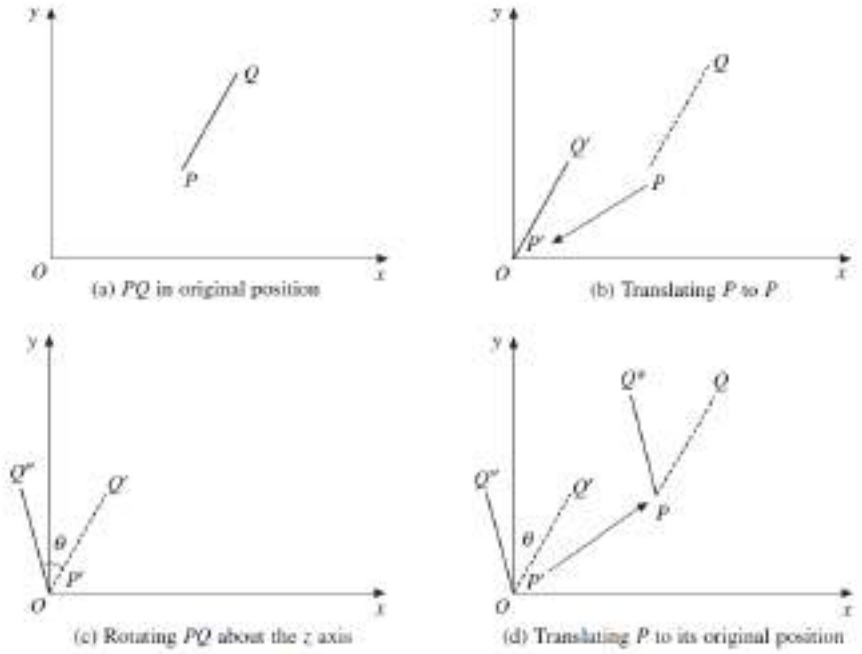


Figure 2.6 Steps to rotate point Q about point P

$$Q^* = \begin{bmatrix} x_q^* \\ y_q^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -p \\ 0 & 1 & -q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} \tag{2.7}$$

2.2.5 Reflection

In 2-D, reflection of an object can be obtained by rotating it through 180° about the axis of reflection. For instance, if an object S in the x-y plane is to be reflected about the x-axis ($y = 0$), reflection of a point (x, y, 1) in S is given by (x^*, y^*, 1) such that

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ -y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{R}_{fx} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.8}$$

Similarly, reflection about the y-axis is described as

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{R}_{fy} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.9}$$

2.2.6 Reflection About an Arbitrary Line

Let D be a point on line L and S be an object in two-dimensional space. It is required to reflect S about L. This reflection can be obtained as a sequence of the following transformations:

(a) Translate point D (p, q, 1) to coincide with the origin O, shifting the line L parallel to itself to a translated position L^*.
(b) Rotate L^* by an angle θ such that it coincides with the y-axis (new position of the line is L^{**}, say).
(c) Reflect S about the y-axis using Eq. (2.9).
(d) Rotate L^{**} through $-\theta$ to bring it back to L^*.
(e) Translate L^* to coincide with its original position L.

The schematic of the procedure is shown in Figure 2.8. The new image S^* is the reflection of S about L and the transformation is given by

$$\begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & -p \\ 0 & 1 & -q \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{T}_{OD}\mathbf{R}(-\theta)\mathbf{R}_{fy}\mathbf{R}(\theta)\mathbf{T}_{DO} \tag{2.10}$$

2.2.7 Reflection Through a Point

A point $\boldsymbol{P}$ (x, y, 1) when reflected through the origin is written as $\boldsymbol{P}^*$ (x^*, y^*, 1) = ($-x$, $-y$, 1) or

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ -y \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{R}_{fo} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.11}$$

For reflection of an object about a point $\boldsymbol{P}_r$, we would require to shift $\boldsymbol{P}_r$ to the origin, perform the above reflection and then transform $\boldsymbol{P}_r$ back to its original position.

2.3.1 Scaling

A point $\boldsymbol{P}$ (x, y, 1) belonging to the object S can be scaled to a new position vector $\boldsymbol{P}^*$ (x^*, y^*, 1) using factors μ_x and μ_y such that

$$x^* = \mu_x x \text{ and } y^* = \mu_y y$$

Or in matrix form

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} \mu_x & 0 & 0 \\ 0 & \mu_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{SP} \tag{2.18}$$

where $\mathbf{S} = \begin{bmatrix} \mu_x & 0 & 0 \\ 0 & \mu_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the scaling matrix. Scale factors μ_x and μ_y are always non-zero and positive. For both μ_x and μ_y less than 1, the geometric model gets *shrunk*. In case of *uniform scaling* when $\mu_x = \mu_y = \mu$, the model gets changed uniformly in size (Figure 2.11) and there is no distortion.

2.3.2 Shear

Consider a matrix $\mathbf{Sh}_x = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ which when applied to a point $\boldsymbol{P}$ (x, y, 1) results in

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + sh_x y \\ y \\ 1 \end{bmatrix} \tag{2.20}$$

which in effect shears the point along the x axis. Likewise, application of $\mathbf{Sh}_y = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ on $\boldsymbol{P}$ yields

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ sh_y x + y \\ 1 \end{bmatrix} \tag{2.21}$$

that is, the new point gets sheared along the y direction.

2.5 Transformations in Three-Dimensions

Matrices developed for transformations in two-dimensions can be modified as per the schema in Eq. (2.23) for use in three-dimensions. For instance, the translation matrix to move a point and thus an object, e.g. in Figure 2.13, by a vector (p, q, r) may be written as

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

Figure 2.13 Translation of a donut along an arbitrary vector

2.5.1 Rotation in Three-Dimensions

The rotation matrix in Eq. (2.4) can be modified to accommodate the three-dimensional homogenous coordinates. For rotation by angle θ about the z-axis (the z coordinate does not change), we get

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

Further, using the cyclic rule for the right-handed coordinate axes, rotation matrices about the x- and y-axis for angles ψ and ϕ can be written, respectively, by inspection as

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.26}$$

Rotation of a point by angles θ, ϕ and ψ (in that order) about the z-, y- and x-axis, respectively, is a useful transformation used often for rigid body rotation. The combined rotation is given as

$$\mathbf{R} = \mathbf{R}_x(\psi)\mathbf{R}_y(\phi)\mathbf{R}_z(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.27}$$

We may as well multiply the three matrices to derive the composite matrix though it is easier to express the transformation in the above form for the purpose of depicting the order of transformations. Also, it is easier to remember the individual transformation matrices than the composite matrix. We may need to rotate an object about a given line. For instance, to rotate an object in Figure 2.14 (a) by 45° about the line $L \equiv y = x$. One way is to rotate the object about the z-axis such that L coincides with the x-axis, perform rotation about the x-axis and then rotate L about the z-axis to its original location. The combined transformation would then be

2.5.2 Scaling in Three-Dimensions

The scaling matrix can be extended from that in a two-dimensional case (Eq. 2.18) as

$$\mathbf{S} = \begin{bmatrix} \mu_x & 0 & 0 & 0 \\ 0 & \mu_y & 0 & 0 \\ 0 & 0 & \mu_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.29}$$

where μ_x, μ_y and μ_z are the scale factors along x, y and z directions, respectively. For uniform overall scaling, $\mu_x = \mu_y = \mu_z = \mu$.

Alternatively,

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix}$$

has the same uniform scaling effect as that of Eq. (2.29). To observe this, we may write

2.5.4 Reflection in Three-Dimensions

Generic reflections about the x-y plane (z becomes $-z$), y-z plane (x becomes $-x$), and z-x plane (y becomes $-y$) can be expressed using the following respective transformations:

$$\mathbf{Rf}_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Rf}_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Rf}_{zx} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.32}$$

2.5.3 Shear in Three-Dimensions

In the 3×3 sub-matrix of the general transformation matrix (2.23), if all diagonal elements including a_{44} are 1, and the elements of 1×3 row sub-matrix and 3×1 column sub-matrix are all zero, we get the shear transformation matrix in three-dimensions, similar to the two-dimensional case. The generic form is

$$\mathbf{Sh} = \begin{bmatrix} 1 & sh_{12} & sh_{13} & 0 \\ sh_{21} & 1 & sh_{23} & 0 \\ sh_{31} & sh_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

whose effect on point $\boldsymbol{P}$ is

$$\begin{bmatrix} x^* \\ y^* \\ z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_{12} & sh_{13} & 0 \\ sh_{21} & 1 & sh_{23} & 0 \\ sh_{31} & sh_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + sh_{12}y + sh_{13}z \\ sh_{23}x + y + sh_{23}z \\ sh_{31}x + sh_{32}y + z \\ 1 \end{bmatrix}$$

Thus, to shear an object only along the y direction, the entries $sh_{12} = sh_{13} = sh_{31} = sh_{32}$ would be 0 while either sh_{21} and sh_{23} or both would be non-zero.

167 160, 195 — 218, 195, 200, 207, 218 → module→2

module→3 → 243-249, 262, 266, 267, 268, 270, 277, 282, 287, 291, 292

module→4 → 262-294, 323-328, 335-345, 351-357, 371-385, 338, 351, 371

module→5 → 517-527, 537, 539, 550, 565, 537-560, 565-569

module→6

4.5.2 Lines

Basic vector parametric equations of straight lines are derived here with two questions in mind. First, how is a line equation converted by the CAD/CAM software into the line database which is at a minimum at the two endpoints of the line? Second, how are the mathematical requirements of an equation correlated with various modifiers available with line commands offered by common user interfaces? Consider the following two cases:

1. A line connecting two points P_1 and P_2, as shown in Fig. 4.14. Define a parameter u such that it has the values 0 and 1 at P_1 and P_2 respectively. Utilizing the triangle OPP_1, the following equation can be written:

$$\mathbf{P} = \mathbf{P}_1 + (\mathbf{P} - \mathbf{P}_1) \tag{4.9}$$

However, the vector $(\mathbf{P} - \mathbf{P}_1)$ is proportional to the vector $\mathbf{P}_2 - \mathbf{P}_1$ such that

$$\mathbf{P} - \mathbf{P}_1 = u(\mathbf{P}_2 - \mathbf{P}_1) \tag{4.10}$$

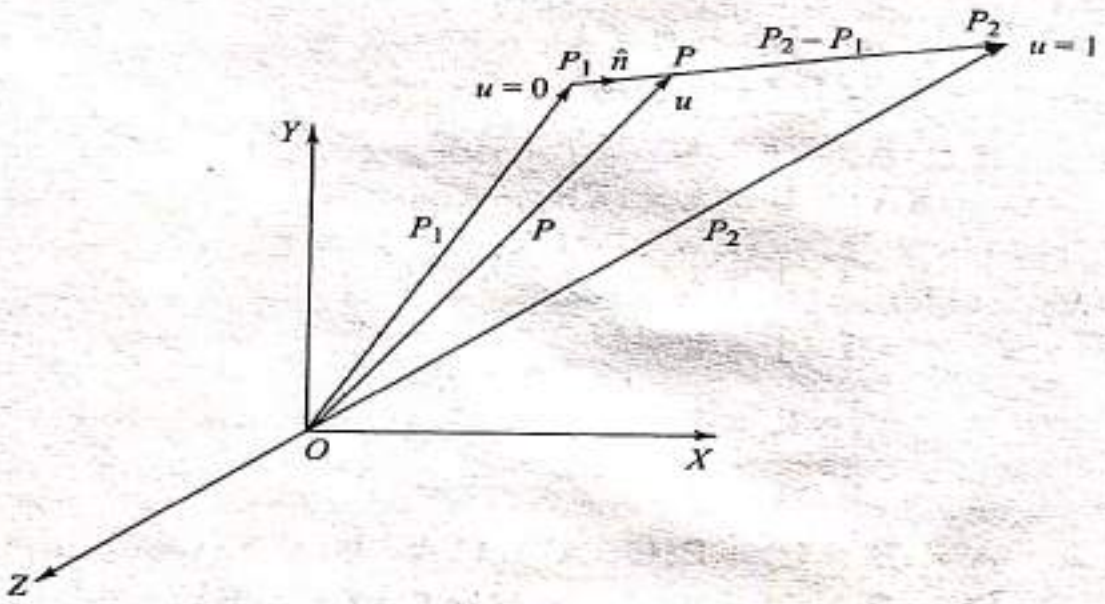


Fig. 4.14 *Line Connecting Two Points P_1 and P_2*

Thus the equation of the line becomes

$$\mathbf{P} = \mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1), \quad 0 \le u \le 1 \tag{4.11}$$

In scalar form, this equation can be written as

$$\left.\begin{aligned} x &= x_1 + u(x_2 - x_1) \\ y &= y_1 + u(y_2 - y_1) \\ z &= z_1 + u(z_2 - z_1) \end{aligned}\right\} \quad 0 \le u \le 1 \tag{4.12}$$

Equation (4.11) defines a line bounded by the endpoints P_1 and P_2 whose associated parametric values are 0 and 1 respectively. Any other point on the line or its extension has a certain value of u which is proportional to the point location, as Fig. 4.15 shows. The coordinates of any point in the figure are obtained by substituting the corresponding u value in Eq. (4.11).

The tangent vector of the line is given by

$$\mathbf{P}' = \mathbf{P}_2 - \mathbf{P}_1 \tag{4.13}$$

or, in scalar form,

$$\begin{aligned} x' &= x_2 - x_1 \\ y' &= y_2 - y_1 \\ z' &= z_2 - z_1 \end{aligned} \tag{4.14}$$

The independence of the tangent vector from u reflects the constant slope of the straight line. For a two-dimensional line, the known infinite (vertical line) and zero (horizontal line) slope conditions can be generated from Eq. (4.14).

The unit vector $\hat{\mathbf{n}}$ in the direction of the line (Fig. 4.14) is given by

$$\hat{\mathbf{n}} = \frac{\mathbf{P}_2 - \mathbf{P}_1}{L} \tag{4.15}$$

where L is the length of the line:

$$L = |\mathbf{P}_2 - \mathbf{P}_1| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{5.16}$$

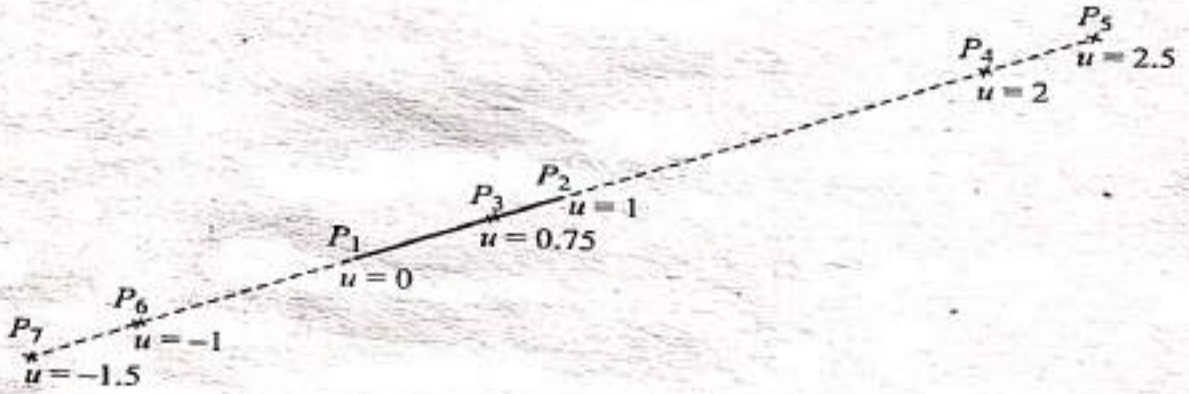


Fig. 4.15 *Locating Points on an Existing Line*

Regardless of the user input to create a line, a line database stores its two endpoints and additional information such as its font, width, color and layer. Equations (4.11) and (4.13) show that the endpoints are enough to provide all geometric properties and characteristics of the line. They are also sufficient to construct and display the line. For reference purposes, CAD/CAM software usually identifies the first point input by the user during line construction as P_1, where $u = 0$. These two equations can be programmed into a subroutine that can reside in a graphics library of the software and which can be invoked, via the user interface, to construct lines. Point commands (or definitions) on most systems provide users with a modifier to specify a u value relative to an entity to generate points on it. In the case of a line, the value is substituted into Eq. (4.11) to find the point coordinates.

2. A line passing through a point P_1 in a direction defined by the unit vector $\hat{\mathbf{n}}$ (Fig. 4.16). Case 1 is considered the basic method to create a line because

it provides the line database directly with the two endpoints. This case and others usually result in generating the endpoints from the user input or given data, as discussed below.

To develop the line equation for this case, consider a general point P on the line at a distance L from P_1. The vector equation of the line becomes (see triangle OP_1P)

$$\mathbf{P} = \mathbf{P}_1 + L\hat{\mathbf{n}}, \quad -\infty \le L \le \infty \tag{4.17}$$

and L is given by

$$L = |\mathbf{P} - \mathbf{P}_1| \tag{4.18}$$

L is the parameter in Eq. (4.17). Thus, the tangent vector is $\hat{\mathbf{n}}$.

Once the user inputs $\hat{\mathbf{P}}_1$, $\hat{\mathbf{n}}$ and L, the point P is calculated using Eq. (4.17) and the line has the two endpoints P_1 and P with u values of 0 and 1 as discussed in case 1.

The following examples show how parametric equations of various line forms can be developed. The examples relate to the most common line commands offered by CAD/CAM software.

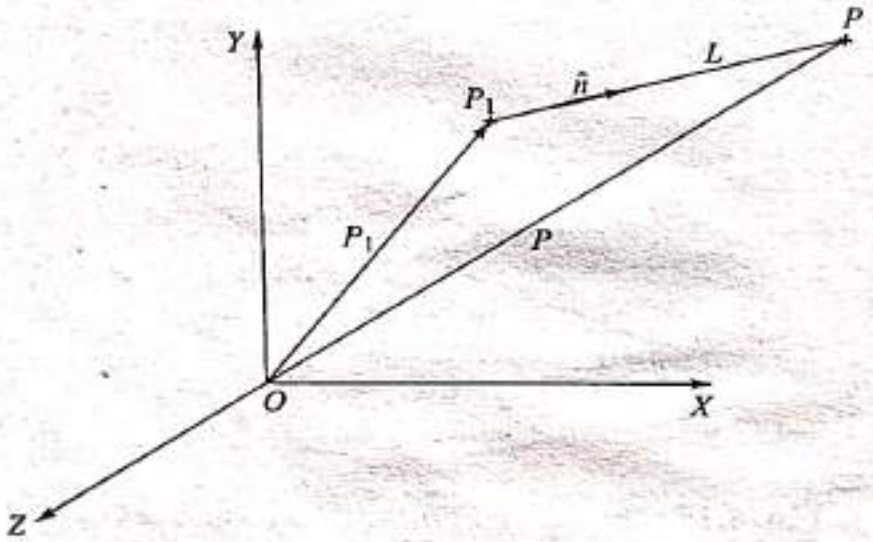


Fig. 4.16 *Line Passing Through P_1 in Direction* $\mathbf{n}$

Example 4.6 Find the equations and endpoints of two lines, one horizontal and the other vertical. Each line begins at and passes through a given point and is clipped by another given point.

Solution Horizontal and vertical lines are usually defined in reference to the current WCS axes. Horizontal lines are parallel to the X axis and vertical lines are parallel to the Y axis. Figure 4.17 shows a typical user working environment where the WCS has a different orientation from the MCS. In this case, the WCS is equivalent to the coordinate system used to develop the line equations. Once the endpoints are calculated from these equations with respect to the WCS, they are transformed to the MCS before the line display or storage.

4.6.1 Hermite Cubic Splines

Parametric spline curves are defined as piecewise polynomial curves with a certain order of continuity. A polynomial of degree N has continuity of derivatives of order $(N-1)$. Parametric cubic splines are used to interpolate to given data, not to design free-form curves as Bezier and B-spline curves do. Splines draw their name from the traditional draftsman's tool called "French curves or splines." The Hermite form of a cubic spline is determined by defining positions and tangent vectors at the data points.

The most commonly used spline curve is a three-dimensional planar curve. The three-dimensional twisted curves are not covered here. For the planar curve, the XY plane of the current WCS is typically used to define the plane of the data points and consequently the plane of the curve. The WCS then serves as the local coordinate system of the spline and is related to the MCS via the proper transformation matrix, as discussed in Chap. 3.

The parametric cubic spline curve (or cubic spline for short) connects two data (end) points and utilizes a cubic equation. Therefore, four conditions are required to determine the coefficients of the equation. When these are the positions of the two endpoints and the two tangent vectors at the points, a Hermite cubic spline results. Thus the Hermite spline is considered as one form of the general parametric cubic spline. The reader is encouraged to extend the forthcoming development of the Hermite cubic spline to a cubic spline defined by four given data points. The parametric equation of a cubic spline segment is given by

$$\mathbf{P}(u) = \sum_{i=0}^{3} \mathbf{C}_i u^i , \quad 0 \le u \le 1 \tag{4.74}$$

where u is the parameter and $\mathbf{C}_i$ are the polynomial (also called algebraic) coefficients. In scalar form this equation is written as

$$\begin{aligned} x(u) &= C_{3x}u^3 + C_{2x}u^2 + C_{1x}u + C_{0x} \\ y(u) &= C_{3y}u^3 + C_{2y}u^2 + C_{1y}u + C_{0y} \\ z(u) &= C_{3z}u^3 + C_{2z}u^2 + C_{1z}u + C_{0z} \end{aligned} \tag{4.75}$$

In an expanded vector form, Eq. (4.74) can be written as

$$\mathbf{P}(u) = \mathbf{C}_3 u^3 + \mathbf{C}_2 u^2 + \mathbf{C}_1 u + \mathbf{C}_0 \tag{4.76}$$

This equation can also be written in a matrix form as

$$\mathbf{P}(u) = \mathbf{U}^T\mathbf{C} \tag{4.77}$$

where $\mathbf{U} = [u^3\ u^2\ u\ 1]^T$ and $\mathbf{C} = [\mathbf{C}_3\ \mathbf{C}_2\ \mathbf{C}_1\ \mathbf{C}_0]^T$. $\mathbf{C}$ is called the coefficients vector.

The tangent vector to the curve at any point is given by differentiating Eq. (4.74) with respect to u to give

$$\mathbf{P}'(u) = \sum_{i=0}^{3} \mathbf{C}_i i u^{i-1}, \quad 0 \le u \le 1 \tag{4.78}$$

In order to find the coefficients $\mathbf{C}_i$, consider the cubic spline curve with the two endpoints P_0 and P_1 shown in Fig. 4.40. Applying the boundary conditions ($\mathbf{P}_0$, $\mathbf{P}'_0$ at $u = 0$ and $\mathbf{P}_1$, $\mathbf{P}'_1$ at $u = 1$), Eqs. (4.74) and (4.78) give

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{C}_0 \\ \mathbf{P}'_0 &= \mathbf{C}_1 \\ \mathbf{P}_1 &= \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0 \\ \mathbf{P}'_1 &= 3\mathbf{C}_3 + 2\mathbf{C}_2 + \mathbf{C}_1 \end{aligned} \tag{4.79}$$

Solving these four equations simultaneously for the coefficients gives

$$\begin{aligned} \mathbf{C}_0 &= \mathbf{P}_0 \\ \mathbf{C}_1 &= \mathbf{P}'_0 \\ \mathbf{C}_2 &= 3(\mathbf{P}_1 - \mathbf{P}_0) - 2(\mathbf{P}'_0 - \mathbf{P}'_1) \\ \mathbf{C}_3 &= 2(\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}'_0 + \mathbf{P}'_1 \end{aligned} \tag{4.80}$$

Substituting Eqs. (4.80) into Eq. (4.76) and rearranging gives

$$\begin{aligned} \mathbf{P}(u) = {} & (2u^3 - 3u^2 + 1)\mathbf{P}_0 + (-2u^3 + 3u^2)\mathbf{P}_1 \\ & + (u^3 - 2u^2 + u)\mathbf{P}'_0 + (u^3 - u^2)\mathbf{P}'_1, \quad 0 \le u \le 1 \end{aligned} \tag{4.81}$$

$\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}'_0$ and $\mathbf{P}'_1$ are called geometric coefficients. The tangent vector becomes

$$\begin{aligned} \mathbf{P}'(u) = {} & (6u^2 - 6u)\mathbf{P}_0 + (-6u^2 + 6u)\mathbf{P}_1 \\ & + (3u^2 - 4u + 1)\mathbf{P}'_0 + (3u^2 - 2u)\mathbf{P}'_1, \quad 0 \le u \le 1 \end{aligned} \tag{4.82}$$

The functions of u in Eqs. (4.81) and (4.82) are called blending functions. The first two functions blend $\mathbf{P}_0$ and $\mathbf{P}_1$ and the second two blend $\mathbf{P}'_0$ and $\mathbf{P}'_1$ to produce the left-hand side in each equation.

Equation (4.81) can be written in a matrix form as

$$\mathbf{P}(u) = \mathbf{U}^T[M_H]\mathbf{V}, \quad 0 \le u \le 1 \tag{4.83}$$

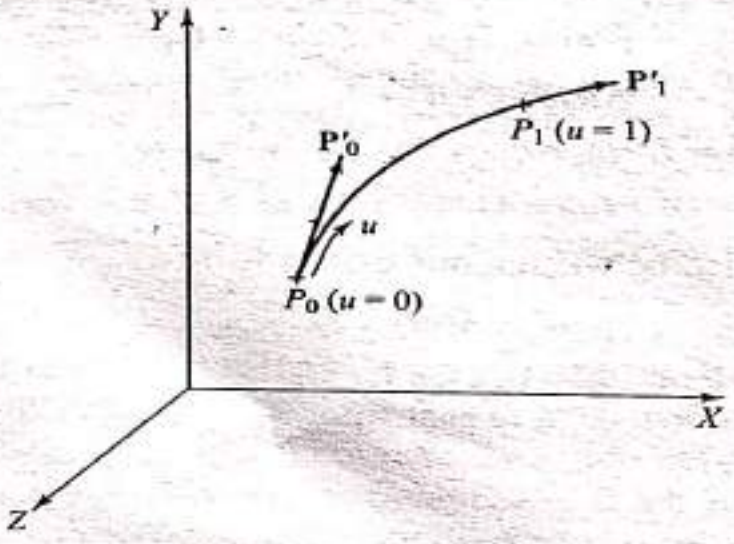


Fig. 4.40 *Hermite Cubic Spline Curve*

where $[M_H]$ is the Hermite matrix and $\mathbf{V}$ is the geometry (or boundary conditions) vector. Both are given by

$$[M_H] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.84}$$

$$\mathbf{V} = [\mathbf{P}_0 \quad \mathbf{P}_1 \quad \mathbf{P'}_0 \quad \mathbf{P'}_1]^T \tag{4.85}$$

Comparing Eqs. (4.77) and (4.83) show that $\mathbf{C} = [M_H]\mathbf{V}$ or $\mathbf{V} = [M_H]^{-1}\,\mathbf{C}$ where

$$[M_H]^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \tag{4.86}$$

Similarly, Eq. (4.82) can be written as

$$\mathbf{P'}(u) = \mathbf{U}^T[M_H]^u\mathbf{V} \tag{4.87}$$

where $[M_H]^u$ is given by

$$[M_H]^u = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.88}$$

Equation (4.81) describes the cubic spline curve in terms of its two end-points and their tangent vectors. The equation shows that the curve passes through the endpoints ($u = 0$ and 1). It also shows that the curve's shape can be controlled by changing its endpoints or its tangent vectors. If the two endpoints P_0 and P_1 are fixed in space, the designer can control the shape of the spline by changing either the magnitudes or the directions of the tangent vectors $\mathbf{P'}_0$ and $\mathbf{P'}_1$. The change of both the magnitudes and the directions is, of course, permissible. However, for planar splines tangent vectors can be replaced by slopes. In this case, a default value, say one, for the lengths of the tangent vectors might be assumed by the software to enable Eq. (4.81) to be used. For example, if the slope at P_0 is given as 30°, then $\mathbf{P'}_0$ becomes [cos 30 sin 30 0]. It is obvious that the slope angle and the components of $\mathbf{P'}_0$ are given relative to the axes of the WCS that is active at the time of creating the spline curve.

Equation (4.81) can also be used to display or plot the spline. Points can be generated on the spline for different values of u between 0 and 1. These points are then transformed to the MCS for display or plotting purposes.

Equation (4:81) is for one cubic spline segment. It can be generalized for any two adjacent spline segments of a spline curve that are to fit a given number of data points. This introduces the problem of blending or joining cubic spline segments which can be stated as follows. Given a set of n points $P_0, P_1, \ldots, P_{n-1}$ and the two

end tangent vectors $\mathbf{P'}_0$ and $\mathbf{P'}_{n-1}$ (Fig. 4.41) connect the points with a cubic spline curve. The spline curve is created as a blend of spline segments connecting the set of points starting from P_0 and ending at P_{n-1}. Tangent vectors at the intermediate points P_1 through P_{n-2} are needed as shown in Eq. (4.81) to compute these segments. To eliminate the need for these vectors, the continuity of curvature at these points can be imposed. To illustrate the procedure, consider eliminating $\mathbf{P'}_1$ between the first two segments that connect points P_0, P_1 and P_2. For curvature continuity between the first two segments, we can write

$$\mathbf{P''}(u_1 = 1) = \mathbf{P''}(u_2 = 0) \tag{4.89}$$

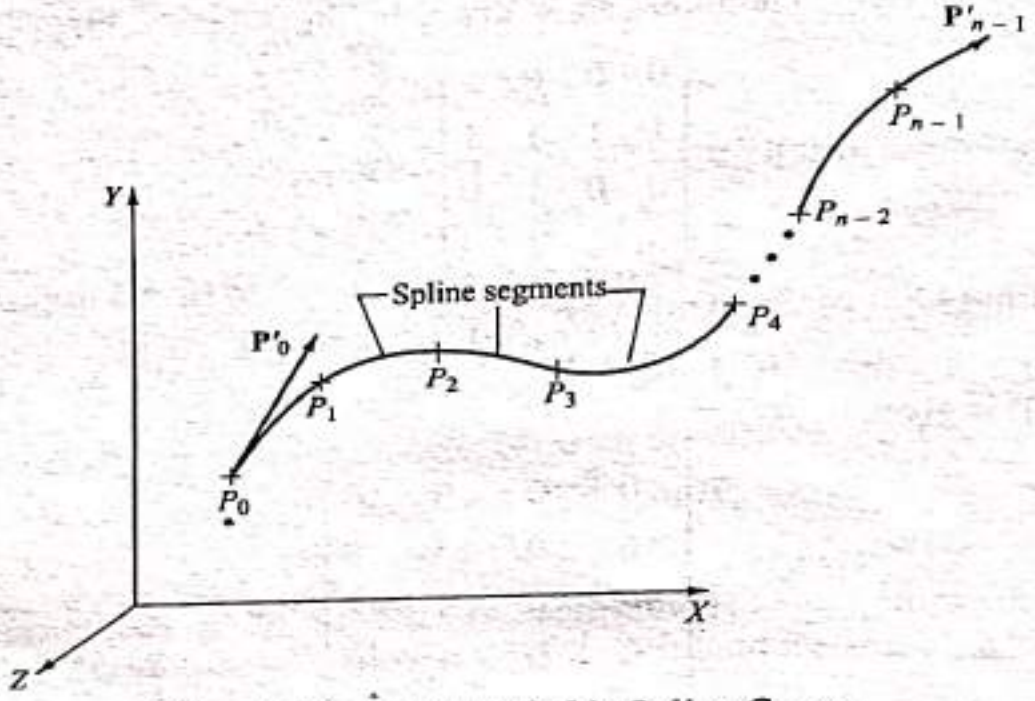


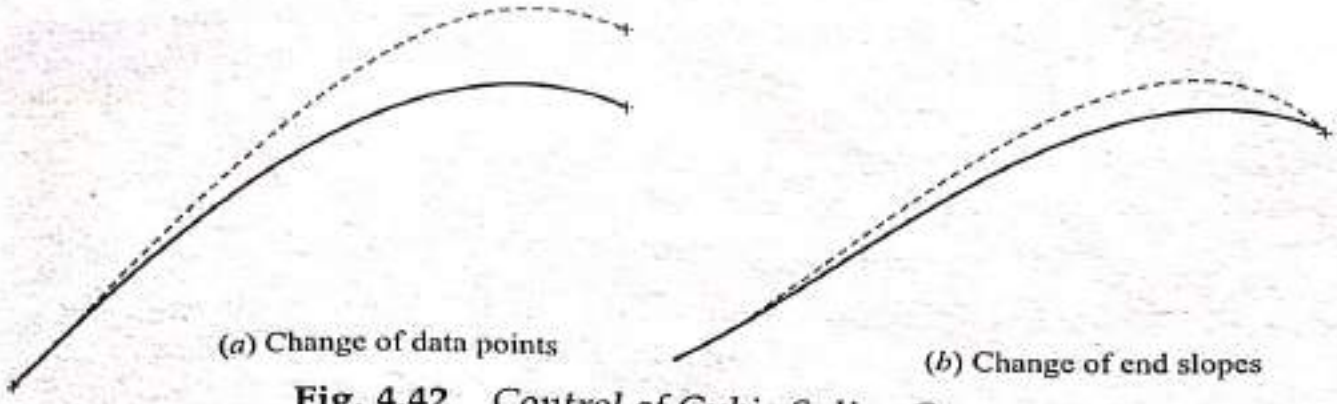
Fig. 4.41 *Hermite Cubic Spline Curve*

where the subscripts of u refer to the segment number. Differentiating Eq. (4.82) and using the result with Eq. (4.89), we obtain

$$\mathbf{P'}_1 = -\tfrac{1}{4}(3\mathbf{P}_0 + \mathbf{P'}_0 - 3\mathbf{P}_2 + \mathbf{P'}_2) \tag{4.90}$$

For more than two segments, a matrix equation can result from repeating this procedure, which can be solved for the intermediate tangent vectors in terms of the data points and the two end tangent vectors $\mathbf{P'}_0$ and $\mathbf{P'}_{n-1}$ Thus, the geometric information of a cubic spline database consists of the set of the data points and the two end tangent vectors.

The use of the cubic splines in design applications is not very popular compared to Bezier or B-spline curves. The control of the curve is not very obvious from the input data due to its global control characteristics. For example, changing the position of a data point or an end slope changes the entire shape of the spline which does not provide the intuitive feel required for design. In addition, the order of the curve is always constant (cubic) regardless of the number of data points. In order to increase the flexibility of the curve, more points must be input, thus creating more splines which are all still of cubic order. Figure 4.42 shows the control aspects of the cubic spline curve.



(a) Change of data points (b) Change of end slopes

Fig. 4.42 *Control of Cubic Spline Curve*

Example 4.18 What shape of a cubic spline curve results if:

(a) $\mathbf{P}_0 = \mathbf{P}_1, \mathbf{P}'_1 = \mathbf{P}'_0$?
(b) $\mathbf{P}_0 = \mathbf{P}_1, \mathbf{P}'_1 = -\mathbf{P}'_0$?

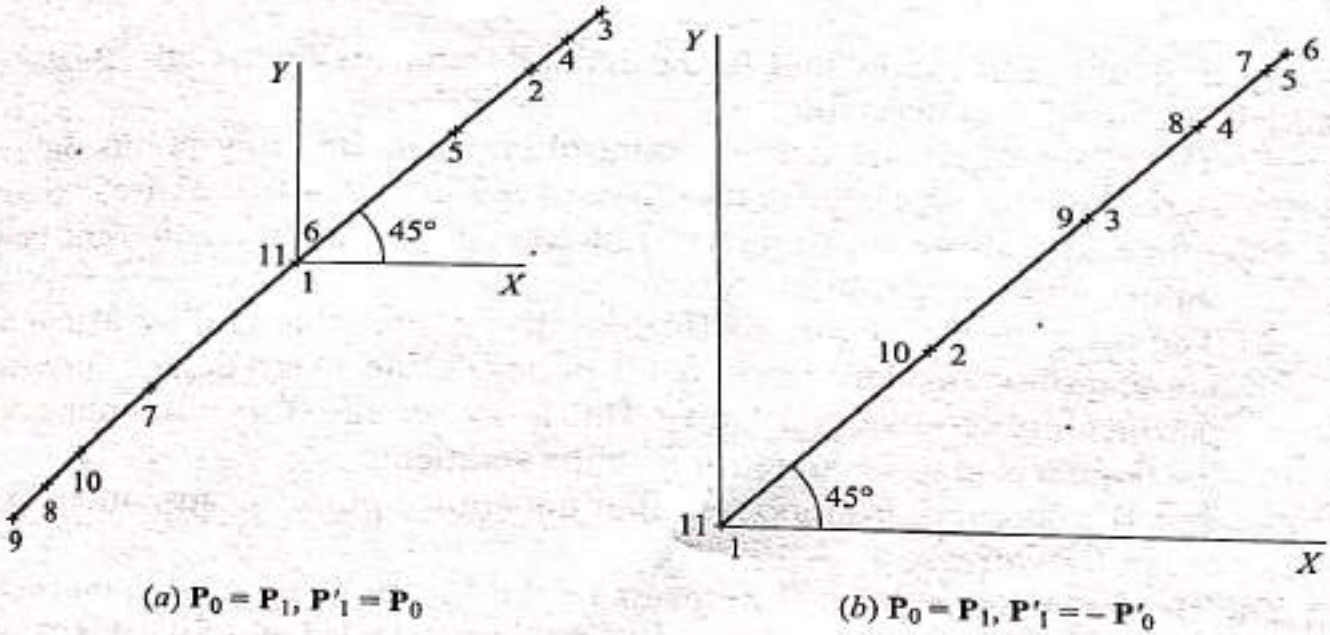
Solution This example illustrates how to create a closed curve using cubic splines. Consider only one segment in this example. Therefore the two endpoints P_0 and P_1 are always identical.

(a) If we substitute the given end conditions into Eqs. (4.81) and (4.82), we get

$$\mathbf{P}(u) = (2u^3 - 3u^2 + u)\mathbf{P}'_0 + \mathbf{P}_0$$

$$\mathbf{P}'(u) = (6u^2 - 6u + 1)\mathbf{P}'_0$$

This spline passes by $\mathbf{P}_0$ at $u = 0, \frac{1}{2}, 1$. Figure 4.43*a* shows the curve for $\mathbf{P}_0 = [0 \quad 0 \quad 0]^T$ and a slope of 45°, that is, $\mathbf{P}'_0 = [1/\sqrt{2} \quad 1/\sqrt{2} \quad 0]^T$. The spline is a straight line in the cartesian space because the slope $y'/x' = y'_0/x'_0$ is constant. Points 1, 2, 3, ..., 11 shown in the figure correspond to values of u



(a) $\mathbf{P}_0 = \mathbf{P}_1, \mathbf{P}'_1 = \mathbf{P}_0$ (b) $\mathbf{P}_0 = \mathbf{P}_1, \mathbf{P}'_1 = -\mathbf{P}'_0$

Fig. 4.43 *Closed Cubic Spline Curves*

equal to 0, 0.1, 0.2,..., 1 respectively. The spline has two extreme points 3 and 9 at $u = 0.2$ and 0.8 respectively. The extreme points can be obtained from solving the equation $\mathbf{P}'(u) = 0$.

(*b*) Similar to (*a*) we obtain

$$\mathbf{P}(u) = (-u^2 + u)\mathbf{P}'_0 + \mathbf{P}_0$$

$$\mathbf{P}'(u) = (-2u + 1)\mathbf{P}'_0$$

This spline has an extreme point at $u = \frac{1}{2}$. It begins the tangent to P'_0 and then overlaps to become the tangent to P'. Figure 4.43*b* shows the curve for P_0 $= [0\ 0\ 0]^T$ and slopes of 45 and 225°. The same analysis made above in (a) can be applied here. The reader may attempt to solve this example on a CAD/CAM system and compare the results using the "verify" command.

4.6.2 Bezier Curves

Cubic splines discussed in the previous section are based on interpolation techniques. Curves resulting from these techniques pass through the given points. Another alternative to create curves is to use approximation techniques which produce curves that do not pass through the given data points. Instead, these points are used to control the shape of the resulting curves. Most often, approximation techniques are preferred over interpolation techniques in curve design due to the added flexibility and the additional intuitive feel provided by the former. Bezier and B-spline curves are examples based on approximation techniques.

Bezier curves and surfaces are credited to P. Bezier of the French car firm Regie Renault who developed (about 1962) and used them in his software system called UNISURF which has been used by designers to define the outer panels of several Renault cars. These curves, known as Bezier curves, were also independently developed by P. DeCasteljau of the French car company Citroen (about 1959) which used it as part of its CAD system. The Bezier UNISURF system was soon published in the literature; this is the reason that the curves now bear Bezier's name.

As its mathematics show shortly, the major differences between the Bezier curve and the cubic spline curve are:

1. The shape of Bezier curve is controlled by its defining points only. First derivatives are not used in the curve development as in the case of the cubic spline. This allows the designer a much better feel for the relationship between input (points) and output (curve).
2. The order or the degree of Bezier curve is variable and is related to the number of points defining it; $n + 1$ points define an nth degree curve which permits higher-order continuity. This is not the case for cubic splines where the degree is always cubic for a spline segment.
3. The Bezier curve is smoother than the cubic spline because it has higher-order derivatives.

The Bezier curve is defined in terms of the locations of $n + 1$ points. These points are called data or control points. They form the vertices of what is called the control or Bezier characteristic polygon which uniquely defines the curve shape as shown in Fig. 4.44. Only the first and the last control points or vertices of the

polygon actually lie on the curve. The other vertices define the order, derivatives and shape of the curve. The curve is also always tangent to the first and last polygon segments. In addition, the curve shape tends to follow the polygon shape. These three observations should enable the user to sketch or predict the curve shape once its control points are given as illustrated in Fig. 4.45. The figure shows that the order of defining the control points changes the polygon definition which changes the resulting curve shape consequently. The arrow depicted on each curve shows its parametrization direction.

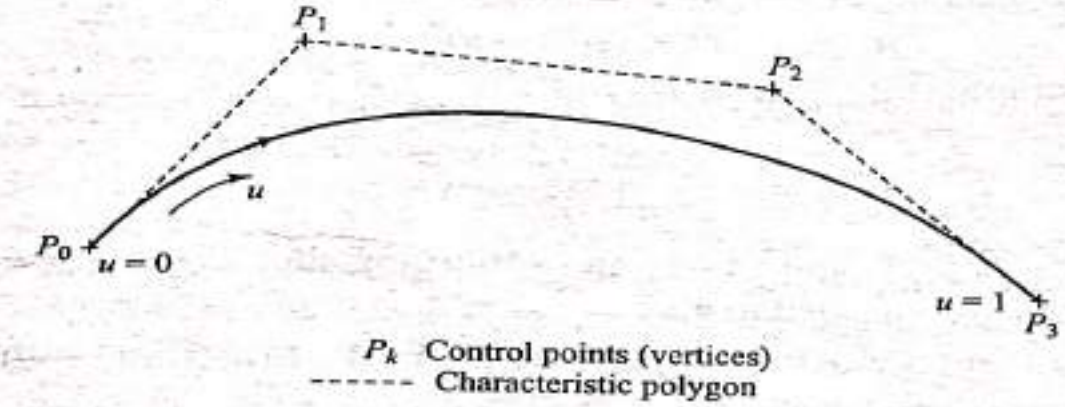


Fig. 4.44 *Cubic Bezier Curve (Nomenclature)*

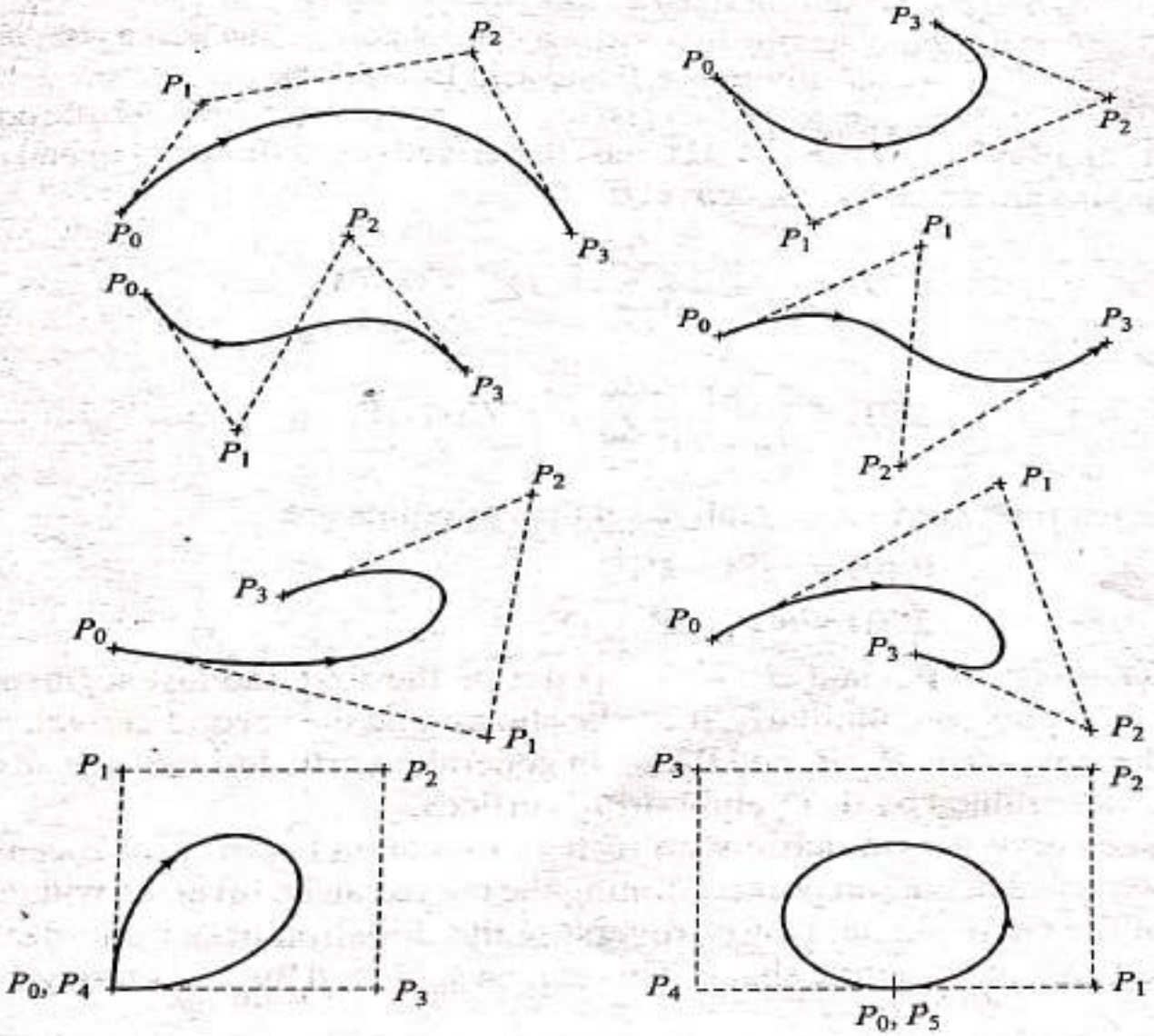


Fig. 4.45 *Cubic Bezier Curves for Various Control Points*

Mathematically, for $n + 1$ control points, the Bezier curve is defined by the following polynomial of degree n:

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(u), \quad 0 \le u \le 1 \tag{4.91}$$

where $P(u)$ is any point on the curve and $\mathbf{P}_i$ is a control point. $B_{i,n}$ are the Bernstein polynomials. Thus, the Bezier curve has a Bernstein basis. The Bernstein polynomial serves as the blending or basis function for the Bezier curve and is given by

$$B_{i,n}(u) = C(n, i)u^i(1-u)^{n-i} \tag{4.92}$$

where $C(n, i)$ is the binomial coefficient

$$C(n, i) = \frac{n!}{i!(n-i)!} \tag{4.93}$$

Utilizing Eqs. (4.92) and (4.93) and observing that $C(n, 0) = C(n, n) = 1$, Eq. (4.91) can be expanded to give

$$\mathbf{P}(u) = \mathbf{P}_0(1-u)^n + \mathbf{P}_1 C(n, 1)u(1-u)^{n-1} + \mathbf{P}_2 C(n, 2)u^2(1-u)^{n-2} + \cdots + \mathbf{P}_{n-1}C(n, n-1)u^{n-1}(1-u) + \mathbf{P}_n u^n, \quad 0 \le u \le 1 \tag{4.94}$$

The characteristics of the Bezier curve are based on the properties of the Bernstein polynomials and can be summarized as follows:

1. The curve interpolates the first and last control points; that is, it passes through P_0 and P_n if we substitute $u = 0$ and 1 in Eq. (4.94).
2. The curve is tangent to the first and last segments of the characteristic polygon. Using Eqs. (4.91) and (4.92), the rth derivatives at the starting and ending points are given by respectively:

$$\mathbf{P}'(0) = \frac{n!}{(n-r)!}\sum_{i=0}^{r} (-1)^{r-i}C(r, i)\mathbf{P}_i \tag{4.95}$$

$$\mathbf{P}'(1) = \frac{n!}{(n-r)!}\sum_{i=0}^{r} (-1)^{i}C(r, i)\hat{\mathbf{P}}_{n-i} \tag{4.96}$$

Therefore, the first derivatives at the endpoints are

$$\mathbf{P}'(0) = n(\mathbf{P}_1 - \mathbf{P}_0) \tag{4.97}$$

$$\mathbf{P}'(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}) \tag{4.98}$$

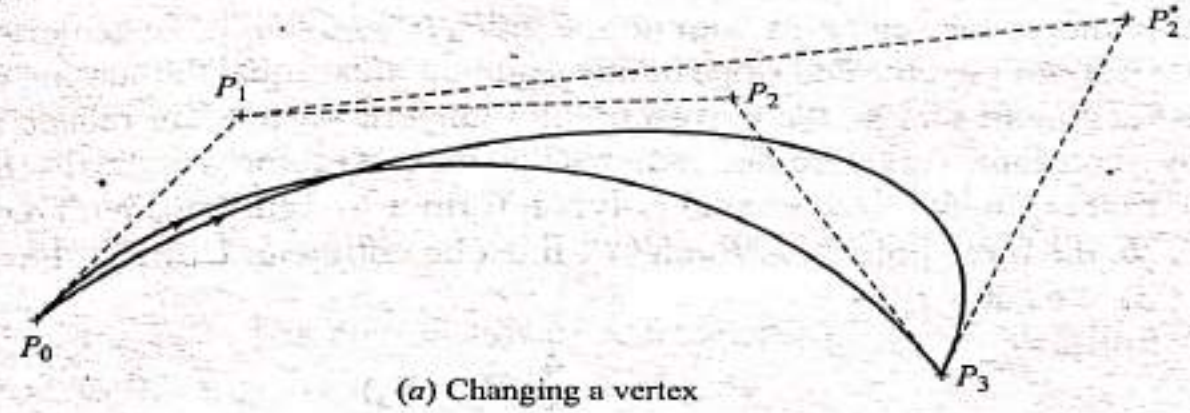
where $(\mathbf{P}_1 - \mathbf{P}_0)$ and $(\mathbf{P}_n - \mathbf{P}_{n-1})$ define the first and last segments of the curve polygon. Similarly, it can be shown that the second derivative at P_0 is determined by $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_2$; or, in general, the rth derivative at an endpoint is determined by its r neighboring vertices.

3. The curve is symmetric with respect to u and $(1 - u)$. This means that the sequence of control points defining the curve can be reversed without change of the curve shape; that is, reversing the direction of parametrization does not change the curve shape. This can be achieved by substituting $1 - u = v$ in

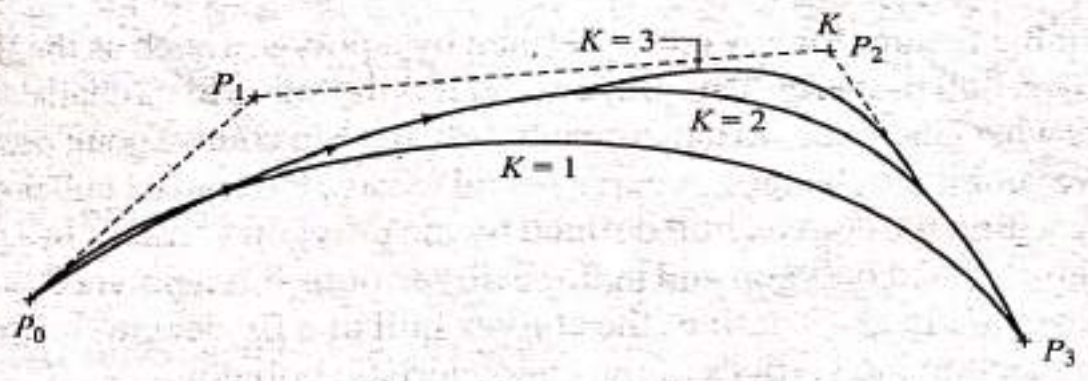
Eq. (4.94) and noticing that $C(n, i) = C(n, n - i)$. This is a result of the fact that $B_{i,n}(u)$ and $B_{n-i,n}(u)$ are symmetric if they are plotted as functions of u.

4. The interpolation polynomial $B_{i,n}(u)$ has a maximum value of $C(n, i)\,(i/n)^i\,(1 - i/n)^{n-i}$ occurring at $u = i/n$ which can be obtained from the equation $d(B_{i,n})/du = 0$. This implies that each control point is most influential on the curve shape at $u = i/n$. For example, for a cubic Bezier curve, $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are most influential when $u = 0$, $\frac{1}{3}$, $\frac{2}{3}$ and 1 respectively. Therefore, each control point is weighed by its blending function for each u value.
5. The curve shape can be modified by either changing one or more vertices of its polygon or by keeping the polygon fixed and specifying multiple coincident points at a vertex, as shown in Fig. 4.46. In Fig. 4.46*a*, the vertex P_2 is pulled to the new position P_2^* and in Fig. 4.46*b*, P_2 is assigned a multiplicity K. The higher the multiplicity, the more the curve is pulled toward P_2.
6. A closed Bezier curve can simply be generated by closing its characteristic polygon or choosing P_0 and P_n to be coincident. Figure 4.45 shows examples of closed curves.
7. For any valid value of u, the sum of the $B_{i,n}$ functions associated with the control points is always equal to unity for any degree of Bezier curve. This fact can be used to check numerical computations and software developments.

Thus far, only one Bezier curve segment is considered. In practical applications, the need may arise to deal with composite curves where various curve segments are blended or joined together. In these applications maintaining continuity of various orders between the segments might be desired. Figure 4.47 shows two

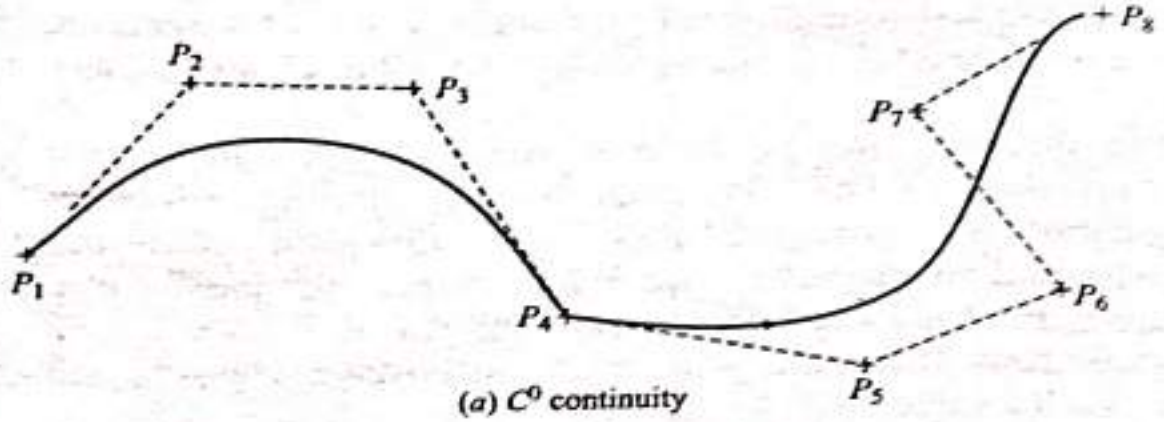


(*a*) Changing a vertex

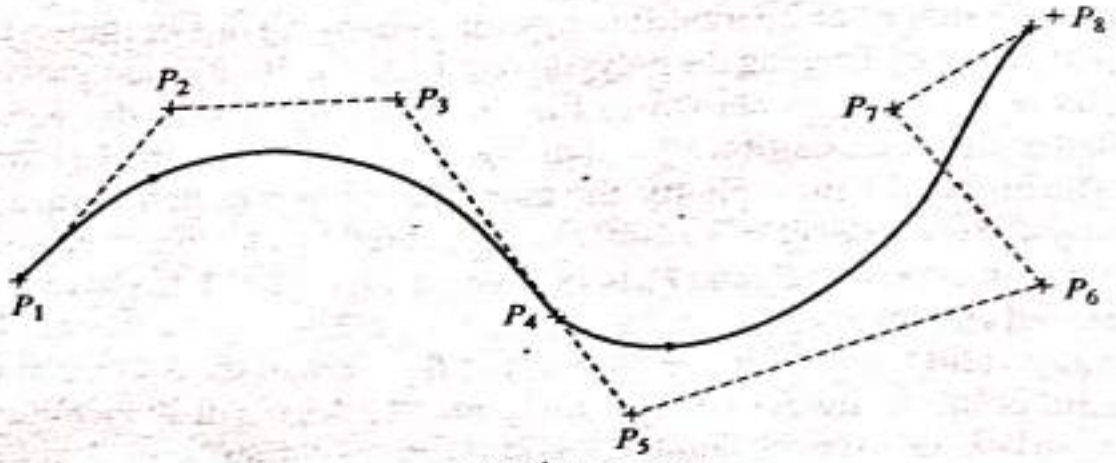


(*b*) Specifying multiple coincident points at a vertex

Fig. 4.46 *Modifications of Cubic Bezier Curve*



(a) C^0 continuity



(b) C^1 continuity

Fig. 4.47 *Blending Bezier Curve Segments*

curve segments defined by the two sets of points P_1, P_2, P_3, P_4 and P_4, P_5, P_6, P_7, P_8. To achieve a zero-order (C^0) continuity, it is sufficient to make one of the end control points of the segments common, e.g., P_4 in Fig. 4.47a. To achieve a first-order (C^1) continuity, the end slope of one segment must equal the starting slope of the next segment; that is, the corresponding tangent vectors are related to each other by a constant. This condition requires that the last segment of the first polygon and the first segment of the second polygon form a straight line. With regards to Fig. 4.47*b*, the three points P_3, P_4 and P_5 must be collinear. Utilizing Eqs. (4.97) and (5.98), we can write

$$\mathbf{P}_4 - \mathbf{P}_3 = \frac{4}{3}(\mathbf{P}_5 - \mathbf{P}_4) \tag{4.99}$$

A most desirable feature for any curve defined by a polygon such as the Bezier curve is the convex hull property. This property relates the curve to its characteristic polygon. This is what guarantees that incremental changes in control point positions produce intuitive geometric changes. A curve is said to have the convex hull property if it lies entirely within the convex hull defined by the polygon vertices. In a plane, the convex hull is a closed polygon and in three dimensions it is a polyhedron. The shaded area shown in Fig. 4.48 defines the convex hull of a Bezier curve. The hull is formed by connecting the vertices of the characteristic polygon.

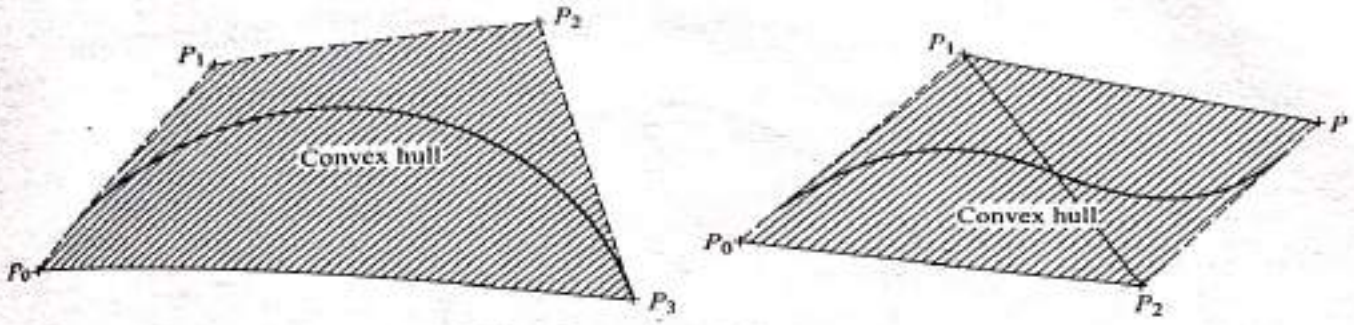


Fig. 4.48 *The Convex Hull of a Bezier Curve*

Curves that possess the convex hull property enjoy some important consequences. If the polygon defining a curve segment degenerates to a straight line, the resulting segment must therefore be linear. Thus a Bezier curve may have locally linear segments embedded in it, which is a useful design feature. Also, the size of the convex hull is an upper bound on the size of the curve itself; that is, the curve always lies inside its convex hull. This is a useful property for graphics functions such as displaying or clipping the curve. For example, instead of testing the curve itself for clipping, its convex hull is tested first and only if it intersects the display window boundaries should the curve itself be. examined. A third consequence of the convex hull property is that the curve never oscillates wildly away from its defining control points because the curve is guaranteed to lie within its convex hull.

From a software point of view, the database of a Bezier curve includes the coordinates of the control points defining its polygon stored in the same order as input by the user. Other information which may obviously be stored include layer, color, name, font and line width of the curve.

While a Bezier curve seems superior to a cubic spline curve, it still has some disadvantages. First, the curve does not pass through the control points which may be inconvenient to some designers. Second, the curve lacks local control. It only has the global control nature. If one control point is changed, the whole curve changes. Therefore, the designer cannot selectively change parts of the curve.

Example ≡ 4.19 The coordinates of four control points relative to a current WCS are given by

$$\mathbf{P}_0 = [2\ \ 2\ \ 0]^T, \quad \mathbf{P}_1 = [2\ \ 3\ \ 0]^T, \quad \mathbf{P}_2 = [3\ \ 3\ \ 0]^T, \text{ and } \mathbf{P}_3 = [3\ \ 2\ \ 0]^T$$

Find the equation of the resulting Bezier curve. Also find points on the curve for $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ and 1.

Solution Equation (4.91) gives

$$\mathbf{P}(u) = \mathbf{P}_0 B_{0,3} + \mathbf{P}_1 B_{1,3} + \mathbf{P}_2 B_{2,3} + \mathbf{P}_3 B_{3,3}, \quad 0 \le u \le 1$$

Using Eqs. (4.92) and (4.93), the above equation becomes

$$\mathbf{P}(u) = \mathbf{P}_0(1-u)^3 + 3\mathbf{P}_1 u(1-u)^2 + 3\mathbf{P}_2 u^2(1-u) + \mathbf{P}_3 u^3, \quad 0 \le u \le 1$$

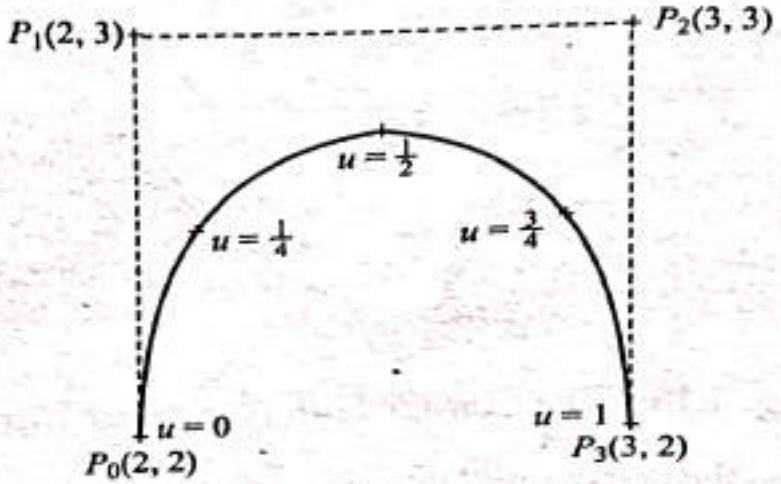


Fig. 4.49 *Bezier Curve and Generated Points*

Substituting the u values into this equation gives $\mathbf{P}(0) = \mathbf{P}_0 = [2\ 2\ 0]^T$

$$\mathbf{P}(0) = \mathbf{P}_0 = [2 \quad 2 \quad 0]^T$$

$$\mathbf{P}\left(\frac{1}{4}\right) = \frac{27}{64}\mathbf{P}_0 + \frac{27}{64}\mathbf{P}_1 + \frac{9}{64}\mathbf{P}_2 + \frac{1}{64}\mathbf{P}_3 = [2.156 \quad 2.563 \quad 0]^T$$

$$\mathbf{P}\left(\frac{1}{2}\right) = \frac{1}{8}\mathbf{P}_0 + \frac{3}{8}\mathbf{P}_1 + \frac{3}{8}\mathbf{P}_2 + \frac{1}{8}\mathbf{P}_3 = [2.5 \quad 2.75 \quad 0]^T$$

$$\mathbf{P}\left(\frac{3}{4}\right) = \frac{1}{64}\mathbf{P}_0 + \frac{9}{64}\mathbf{P}_1 + \frac{27}{64}\ \mathbf{P}_2 + \frac{27}{64}\mathbf{P}_3 = [2.844 \quad 2.563 \quad 0]^T$$

$$\mathbf{P}(1) = \mathbf{P}_3 = [3 \quad 2 \quad 0]^T$$

Observe that $\sum_{i=0}^{3} B_{i,3}$ is always equal to unity for any u value. Figure 4.49 shows the curve and the points.

Example ≡ 4.20 A cubic spline curve is defined by the equation

$$\mathbf{P}(u) = \mathbf{C}_3 u^3 + \mathbf{C}_2 u^2 + \mathbf{C}_1 u + \mathbf{C}_0, \quad 0 \le u \le 1 \tag{4.100}$$

where $\mathbf{C}_3$, $\mathbf{C}_2$, $\mathbf{C}_1$ and $\mathbf{C}_0$ are the polynomial coefficients [see Eq. (4.76)]. Assuming these coefficients are known, find the four control points that define an identical Bezier curve.

Solution The Bezier equation is

$$\mathbf{P}(u) = \mathbf{P}_0 B_{0,3} + \mathbf{P}_1 B_{1,3} + \mathbf{P}_2 B_{2,3} + \mathbf{P}_3 B_{3,3} \tag{4.101}$$

where

$$B_{0,3} = 1 - 3u + 3u^2 - u^3 \qquad B_{1,3} = 3u - 6u^2 + 3u^3$$
$$B_{2,3} = 3u^2 - 3u^3 \qquad B_{3,3} = u^3$$

Substituting all these functions into Eq. (4.101) and rearranging, we obtain

$$\mathbf{P}(u) = (-\mathbf{P}_0 + 3\mathbf{P}_1 - 3\mathbf{P}_2 + \mathbf{P}_3)u^3 + (3\mathbf{P}_0 - 6\mathbf{P}_1 + 3\mathbf{P}_2)u^2 + (-3\mathbf{P}_0 + 3\mathbf{P}_1)u + \mathbf{P}_0 \tag{4.102}$$

Comparing the coefficients of Eqs. (4.100) and (4.102) gives

$$\mathbf{P}_0 = \mathbf{C}_0$$

$$\mathbf{P}_1 = \frac{1}{3}\mathbf{C}_1 + \mathbf{C}_0$$

$$\mathbf{P}_2 = \frac{1}{3}(\mathbf{C}_2 + 2\mathbf{C}_1 + 3\mathbf{C}_0)$$

$$\mathbf{P}_3 = \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0$$

Note: To check these results, observe that the Bezier curve passes through points P_0 and P_3 where $u = 0$ and 1 respectively. These two points are obtained from Eq. (4.100) for the same value of u.

4.63 B-Spline Curves

B-spline curves provide another effective method, besides that of Bezier, of generating curves defined by polygons. In fact, B-spline curves are the proper and powerful generalization of Bezier curves. In addition to sharing most of the characteristics of Bezier curves they enjoy some other unique advantages. They provide local control of the curve shape as opposed to global control by using a special set of blending functions that provide local influence. They also provide the ability to add control points without increasing the degree of the curve.

B-spline curves have the ability to interpolate or approximate a set of given data points. Interpolation is useful in displaying design or engineering results such as stress or displacement distribution in a part while approximation is good to design free-form curves. Interpolation is also useful if the designer has measured data points in hand that must lie on the resulting curve. This section covers only B-spline curves as used for approximation.

In contrast to Bezier curves, the theory of B-spline curves separates the degree of the resulting curve from the number of the given control points. While four control points can always produce a cubic Bezier curve, they can generate a linear, quadratic, or cubic B-spline curve. This flexibility in the degree of the resulting curve is achieved by choosing the basis (blending) functions of B-spline curves with an additional degree of freedom that does not exist in Bernstein polynomials. These basis functions are the B-splines—thus the name B-spline curves.

Similar to Bezier curves, the B-spline curve defined by $n + 1$ control points P_i is given by

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(u), \quad 0 \le u \le u_{\max} \tag{4.103}$$

$N_{i,k}(u)$ are the B-spline functions. Thus B-spline curves have a B-spline basis. The control points (sometimes called deBoor points) form the vertices of the control or deBoor polygon. There are two major differences between Eqs. (4.103) and (4.91). First, the parameter k controls the degree $(k - 1)$ of the resulting B-spline curve and is usually independent of the number of control points except as restricted as shown below. Second, the maximum limit of the parameter u is no longer unity

as it was so chosen arbitrarily for Bezier curves. The B-spline functions have the following properties:

Partition of unity: $\sum_{i=0}^{n} N_{i,k}(u) = 1$

Positivity: $N_{i,k}(u) \geq 0$

Local support: $N_{i,k}(u) = 0 \quad \text{if } u \notin [u_i, u_{i+k+1}]$

Continuity: $N_{i,k}(u)$ is $(k-2)$ times continuously differentiable

The first property ensures that the relationship between the curve and its defining control points is invariant under affine transformations. The second property guarantees that the curve segment lies completely within the convex hull of P_i. The third property indicates that each segment of a B-spline curve is influenced by only k control points or each control point affects only k curve segments. It is useful to notice that the Bernstein polynomial, $B_{i,n}(u)$, has the same first two properties mentioned above.

The B-spline function also has the property of recursion which is defined as

$$N_{i,k}(u) = (u - u_i)\frac{N_{i,k-1}(u)}{u_{i+k-1} - u_i} + (u_{i+k} - u)\frac{N_{i+1,k-1}(u)}{u_{i+k} - u_{i+1}} \tag{4.104}$$

where

$$N_{i,1} = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{4.105}$$

Choose $0/0 = 0$ if the denominators in Eq. (4.104) become zero. Equation (4.105) shows that $N_{i,1}$ is a unit step function.

Because $N_{i,1}$ is constant for $k = 1$, a general value of k produces a polynomial in u of degree $(k-1)$ [see Eq. (4.104)] and therefore a curve of order k and degree $(k-1)$. The u_i are called parametric knots or knot values. These values form a sequence of nondecreasing integers called the knot vector. The values of the u_i depend on whether the B-spline curve is an open (nonperiodic) or closed (periodic) curve. For an open curve, they are given by

$$u_j = \begin{cases} 0, & j < k \\ j - k + 1, & k \leq j \leq n \\ n - k + 2, & j > n \end{cases} \tag{4.106}$$

where

$$0 \leq j \leq n + k \tag{4.107}$$

and the range of u is

$$0 \leq u \leq n - k + 2 \tag{4.108}$$

Relation (4.107) shows that $(n + k + 1)$ knots are needed to create a $(k-1)$ degree curve defined by $(n + 1)$ control points. These knots are evenly spaced over the range of u with unit separation $(\Delta u = 1)$ between noncoincident knots. Multiple (coincident) knots for certain values of u may exist.

While the degree of the resulting B-spline curve is controlled by k, the range of the parameter u as given by Eq. (4.108) implies that there is a limit on k that is determined by the number of the given control points. This limit is found by requiring the upper bound in Eq. (4.108) to be greater than the lower bound for the u range to be valid, that is,

$$n - k + 2 > 0 \tag{4.109}$$

This relation shows that a minimum of two, three and four control points are required to define a linear, quadratic and cubic B-spline curve respectively.

The characteristics of B-spline curves that are useful in design can be summarized as follows:

1. The local control of the curve can be achieved by changing the position of a control point(s), using multiple control points by placing several points at the same location, or by choosing a different degree $(k - 1)$. As mentioned earlier, changing one control point affects only k segments. Figure 4.50 shows the local control for a cubic B-spline curve by moving P_3 to P_3^* and P_3^{**}. The four curve segments surrounding P_3 change only.
2. A nonperiodic B-spline curve passes through the first and last control points P_0 and P_{n+1} and is tangent to the first $(P_1 - P_0)$ and last $(P_{n+1} - P_n)$ segments of the control polygon, similar to the Bezier curve, as shown in Fig. 4.50.

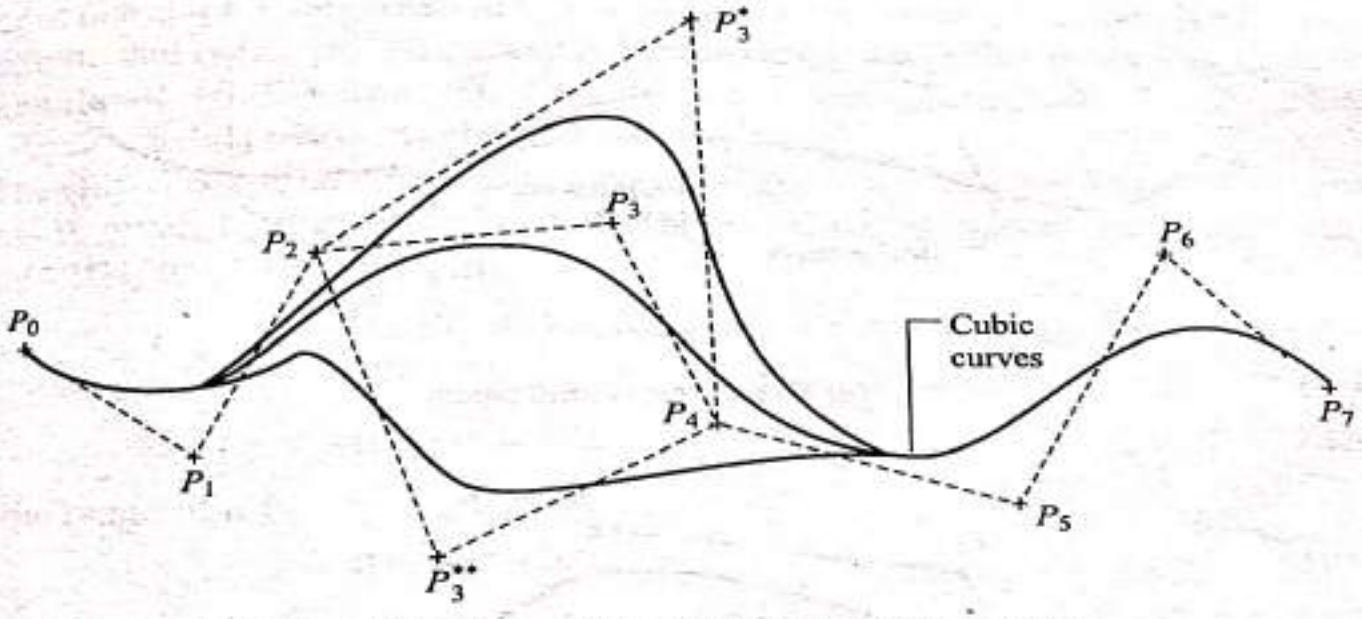


Fig. 4.50 *Local Control of B-spline Curves*

3. Increasing the degree of the curve tightens it. In general, the less the degree, the closer the curve gets to the control points, as shown in Fig. 4.51. When $k = 1$, a zero-degree curve results. The curve then becomes the control points themselves. When $k = 2$, the curve becomes the polygon segments themselves.
4. A second-degree curve is always tangent to the midpoints of all the internal polygon segments (see Fig. 4.51). This is not the case for other degrees.

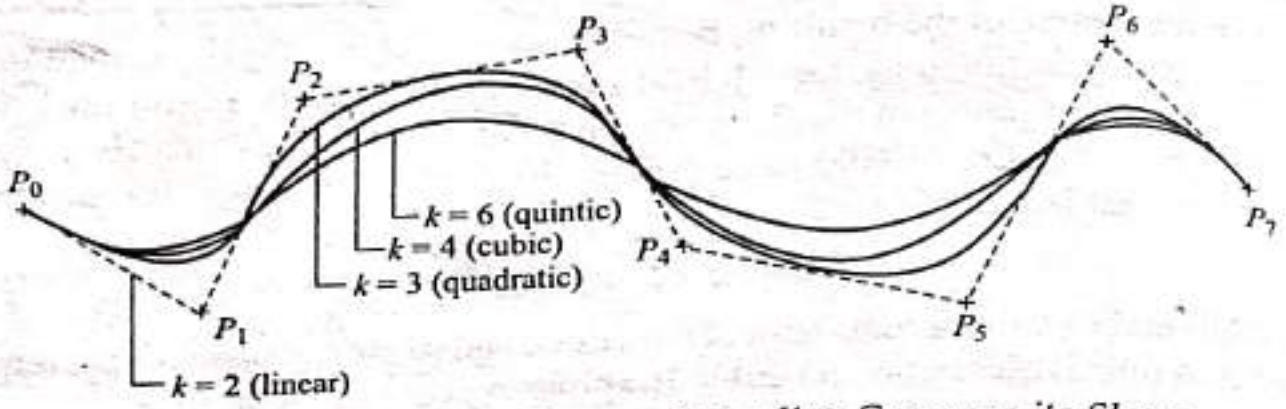
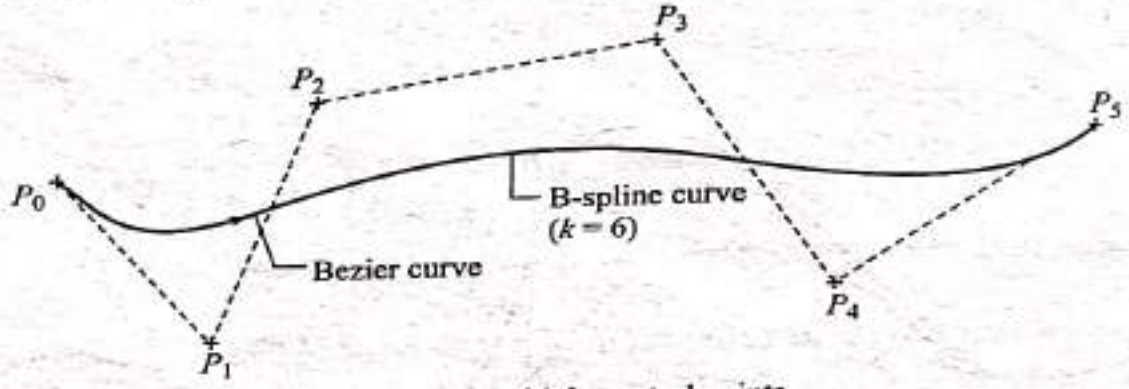
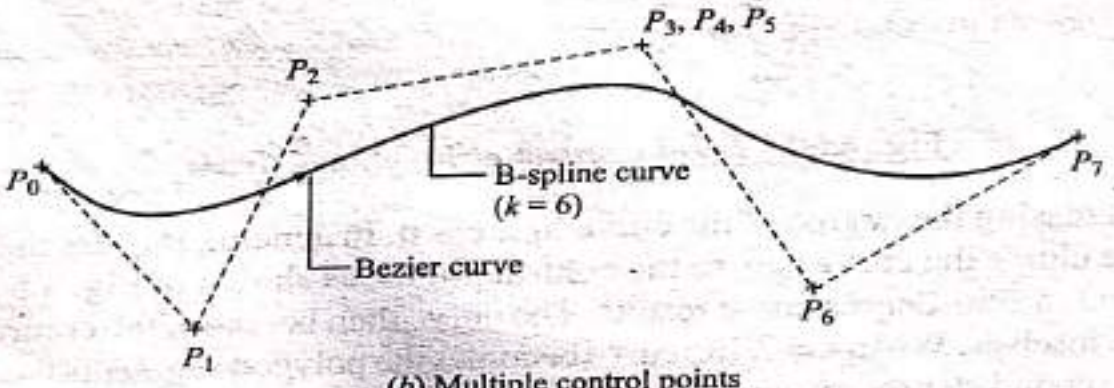


Fig. 4.51 *Effect of the Degree of B-spline Curve on its Shape*

5. If k equals the number of control points $(n + 1)$, then the resulting B-spline curve becomes a Bezier curve (see Fig. 4.52). In this case the range of u becomes zero to one [see Eq. (4.108)] as expected.
6. Multiple control points induce regions of high curvature of a B-spline curve. This is useful when creating sharp corners in the curve (see Fig. 4.53). This effect is equivalent to saying that the curve is pulled more towards a control point by increasing its multiplicity.
7. Increasing the degree of the curve makes it more difficult to control and to calculate accurately. Therefore, a cubic B-spline is sufficient for a large number of applications.



(*a*) No multiple control points



(*b*) Multiple control points

Fig. 4.52 *Identical B-spline and Bezier Curves*

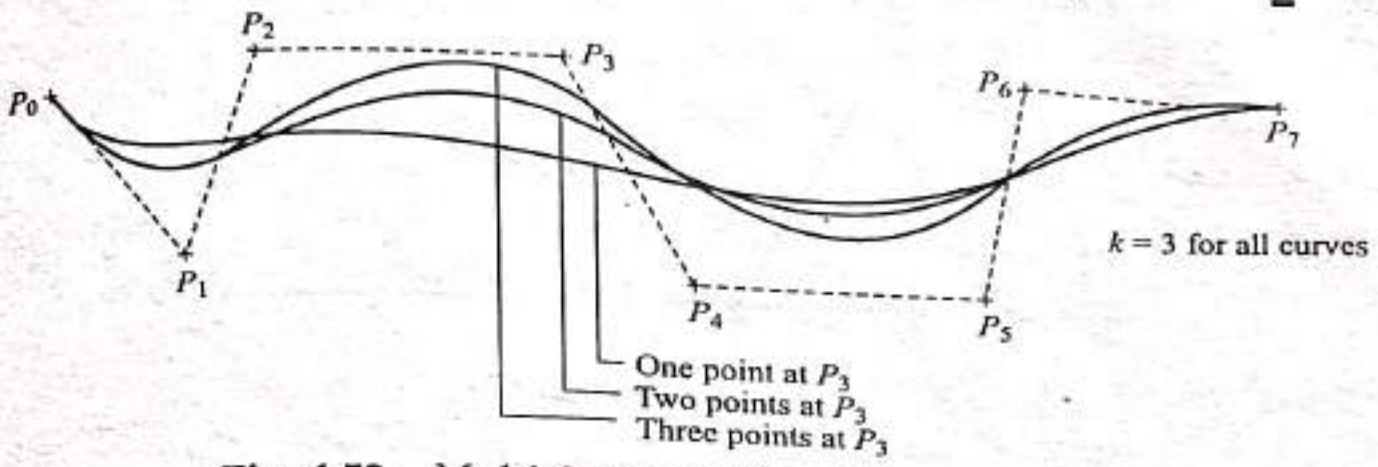


Fig. 4.53 *Multiple Control Point B-spline Curves*

Thus far, open or nonperiodic B-spline curves have been discussed. The same theory can be extended to cover closed or periodic B-spline curves. The only difference between open and closed curves is in the choice of the knots and the basic functions. Equations (4.106) to (4.108) determine the knots and the spacing between them for open curves. Closed curves utilize periodic B-spline functions as their basis with knots at the integers. These basis functions are cyclic translates of a single canonical function with a period (interval) of k for support. For example, for a closed B-spline curve of order 2 ($k = 2$) or a degree 1 ($k - 1$), the basis function is linear, has a nonzero value in the interval (0, 2) only and has a maximum value of one at $u = 1$, as shown in Fig. 4.54. The knot vector in this case is [0 1 2]. Quadratic and cubic closed curves have quadratic and cubic basis functions with intervals of (0, 3) and (0, 4) and knot vectors of [0 1 2 3] and [0 1 2 3 4] respectively.

The closed B-spline curve of degree $(k-1)$ or order k defined by $(n+1)$ control points is given by Eq. (4.103) as the open curve. However, for closed curves Eqs. (4.104) to (4.108) become

$$N_{i,k}(u) = N_{0,k}((u - i + n + 1) \bmod (n+1)) \tag{4.110}$$

$$u_j = j, \quad 0 \le j \le n+1 \tag{4.111}$$

$$0 < j \le n+1 \tag{4.112}$$

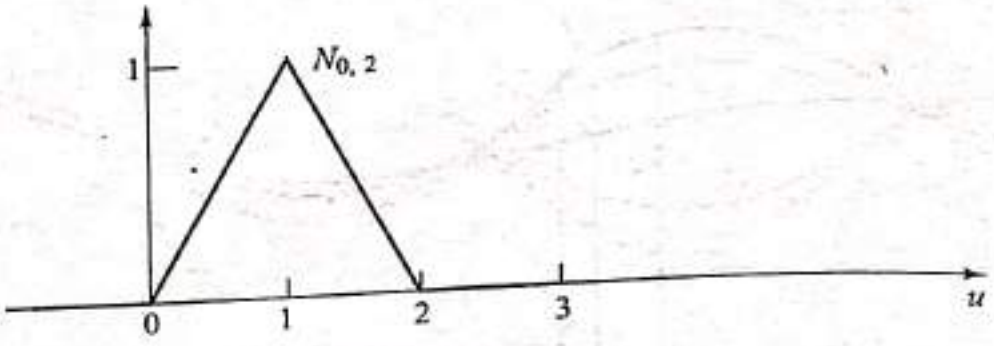
and the range of u is

$$0 \le u \le n+1 \tag{4.113}$$

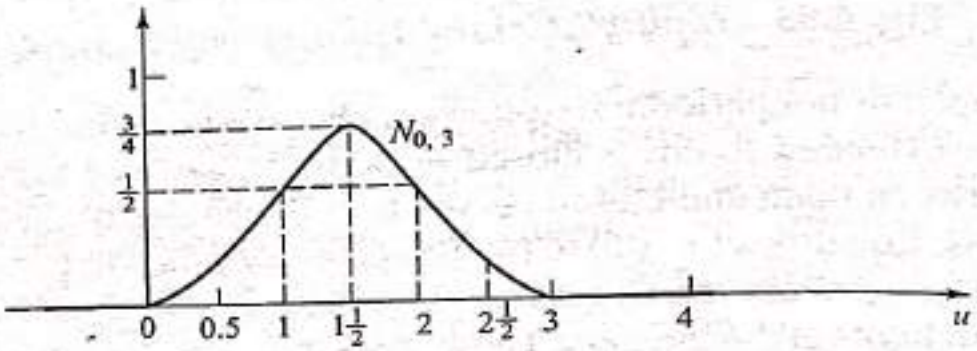
The mod $(n+1)$ in Eq. (4.110) is the modulo function. It is defined as

$$A \bmod n = \begin{cases} A, & A < n \\ 0, & A = n \\ \text{remainder of } A/n, & A > n \end{cases} \tag{4.114}$$

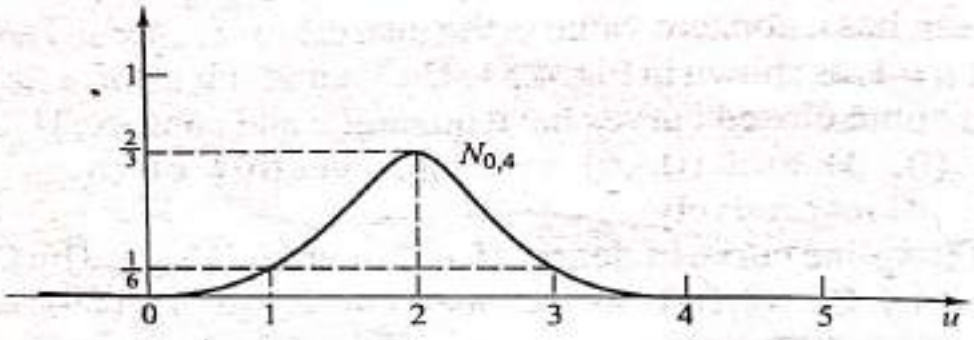
For example, 3.5 mod 6 = 3.5, 6 mod 6 = 0 and 7 mod 6 = 1. The mod function enables the periodic (cyclic) translation [mod $(n+1)$] of the canonical basis function $N_{0,k}$. $N_{0,k}$ *is* the same as for open curves and can be calculated using Eqs. (4.104) and (4.105).



(a) Linear function ($k = 2$)



(b) Quadratic function ($k = 3$)



(c) Cubic function ($k = 4$)

Fig. 4.54 *Periodic B-spline Basis Functions*

Like open curves, closed B-spline curves enjoy the properties of partition of unity, positivity, local support and continuity. They also share the same characteristics of the open curves except that they do not pass through the first and last control points and therefore are not tangent to the first and last segments of the control polygon. In representing closed curves, closed polygons are used where the first and last control points are connected by a polygon segment. It should be noticed that a closed B-spline curve can not be generated by simply using an open curve with the first and last control points being the same (coincident). The resulting curve is only C^0 continuous, as shown in Fig. 4.55. Only if the first and last segments of the polygon are colinear does a C^1 continuous curve result as in a Bezier curve.

Based on the above theory, the database of a B-spline curve includes the type of curve (open or closed), its order k or degree $(k - 1)$ and the coordinates of the control points defining its polygon stored in the same order as input by the user. Other information such as layer, color, name, font and line width of the curve may be stored.

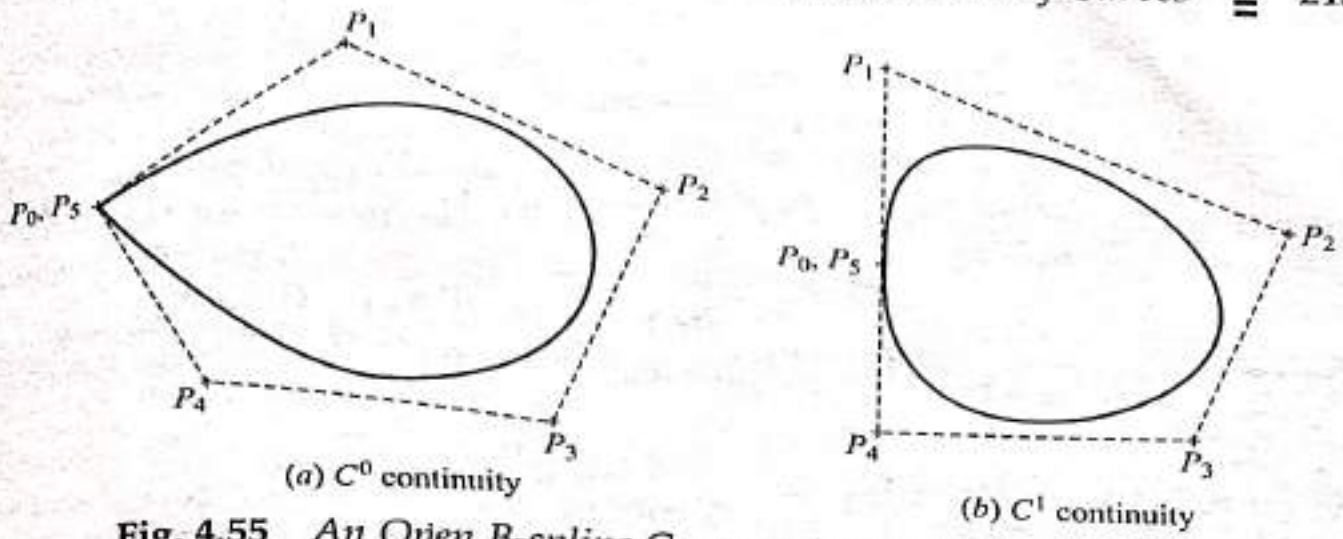


Fig. 4.55 *An Open B-spline Curve with P_0 and P_n Coincident*

Example ≡ 4.21 Find the equation of a cubic B-spline curve defined by the same control points as in Example 4.19. How does the curve compare with the Bezier curve?

Solution This cubic spline has $k = 4$ and $n = 3$. Eight knots are needed to calculate the B-spline functions. Equation (4.106) gives the knot vector

$$[u_0 \quad u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7] \text{ as } [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]$$

The range of u [Eq. (4.108)] is $0 \le u \le 1$. Equation (4.103) gives

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4} + \mathbf{P}_1 N_{1,4} + \mathbf{P}_2 N_{2,4} + \mathbf{P}_3 N_{3,4}, \quad 0 \le u \le 1 \qquad (4.115)$$

To calculate the above B-spline functions, use Eqs. (4.104) and (4.105) together with the knot vector as follows:

$$N_{0,1} = N_{1,1} = N_{2,1} = \begin{cases} 1, & u = 0 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{3,1} = \begin{cases} 1, & 0 \le u \le 1 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{4,1} = N_{5,1} = N_{6,1} = \begin{cases} 1, & u = 1 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{0,2} = (u - u_0)\frac{N_{0,1}}{u_1 - u_0} + (u_2 - u)\frac{N_{1,1}}{u_2 - u_1} = \frac{uN_{0,1}}{0} + \frac{(-u)N_{1,1}}{0} = 0$$

$$N_{1,2} = (u - u_1)\frac{N_{1,1}}{u_2 - u_1} + (u_3 - u)\frac{N_{2,1}}{u_3 - u_2} = \frac{uN_{1,1}}{0} + \frac{(-u)N_{2,1}}{0} = 0$$

$$N_{2,2} = (u - u_2)\frac{N_{2,1}}{u_3 - u_2} + (u_4 - u)\frac{N_{3,1}}{u_4 - u_3} = \frac{uN_{2,1}}{0} + \frac{(1-u)N_{3,1}}{1} = (1 - u)N_{3,1}$$

$$N_{3,2} = (u-u_3)\frac{N_{3,1}}{u_4-u_3} + (u_5-u)\frac{N_{4,1}}{u_5-u_4} = uN_{3,1} + \frac{(1-u)N_{4,1}}{0} = uN_{3,1}$$

$$N_{4,2} = (u-u_4)\frac{N_{4,1}}{u_5-u_4} + (u_6-u)\frac{N_{5,1}}{u_6-u_5} = (u-1)\frac{N_{4,1}}{0} + \frac{(1-u)N_{5,1}}{0} = 0$$

$$N_{5,2} = (u-u_5)\frac{N_{5,1}}{u_6-u_5} + (u_7-u)\frac{N_{6,1}}{u_7-u_6} = \frac{(u-1)N_{5,1}}{0} + \frac{(1-u)N_{6,1}}{0} = 0$$

$$N_{0,3} = (u-u_0)\frac{N_{0,2}}{u_2-u_0} + (u_3-u)\frac{N_{1,2}}{u_3-u_1} = u\frac{0}{0} + (-u)\frac{0}{0} = 0$$

$$N_{1,3} = (u-u_1)\frac{N_{1,2}}{u_3-u_1} + (u_4-u)\frac{N_{2,2}}{u_4-u_2} = u\frac{N_{1,2}}{0} + \frac{(1-u)N_{2,2}}{1} = (1-u)^2 N_{3,1}$$

$$N_{2,3} = (u-u_2)\frac{N_{2,2}}{u_4-u_2} + (u_5-u)\frac{N_{3,2}}{u_5-u_3} = uN_{2,2} + (1-u)N_{3,2} = 2u(1-u)N_{3,1} = 0$$

$$N_{3,3} = (u-u_3)\frac{N_{3,2}}{u_5-u_3} + (u_6-u)\frac{N_{4,2}}{u_6-u_4} = u^2 N_{3,1} + (1-u)\frac{N_{4,2}}{0} = u^2 N_{3,1}$$

$$N_{4,3} = (u-u_4)\frac{N_{4,2}}{u_6-u_4} + (u_7-u)\frac{N_{5,2}}{u_7-u_5} = (u-1)\frac{N_{4,2}}{0} + (1-u)\frac{N_{5,2}}{0} = 0$$

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

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

$$N_{2,4} = (u-u_2)\frac{N_{2,3}}{u_5-u_2} + (u_6-u)\frac{N_{3,3}}{u_6-u_3} = 3u^2(1-u)N_{3,1}$$

$$N_{3,4} = (u-u_3)\frac{N_{3,3}}{u_6-u_3} + (u_7-u)\frac{N_{4,3}}{u_7-u_4} = u^3 N_{3,1}$$

Substituting $N_{i,4}$ into Eq. (4.115) gives

$$\mathbf{P}(u) = [\mathbf{P}_0(1-u)^3 + 3\mathbf{P}_1 u(1-u)^2 + 3\mathbf{P}_2 u^2(1-u) + \mathbf{P}_3 u^3]N_{3,1}, \quad 0 \le u \le 1$$

Substituting $N_{3,1}$ into this equation gives the curve equation as

$$\mathbf{P}(u) = \mathbf{P}_0(1-u)^3 + 3\mathbf{P}_1 u(1-u)^2 + 3\mathbf{P}_2 u^2(1-u) + \mathbf{P}_3 u^3, \quad 0 \le u \le 1$$

This equation is the same as the one for the Bezier curve in Example 4.19. Thus the cubic B-spline curve defined by four control points is identical to the cubic Bezier curve defined by the same points. This fact can be generalized for a $(k-1)$-degree curve as mentioned earlier.

There are two observations that are worth mentioning here. First, the sum of the two subscripts (i, k) of any B-spline function $N_{i,k}$ cannot exceed $(n + k)$. This gives a control on how far to go to calculate $N_{i,k}$. In this example six functions of $N_{i,1}$, five of $N_{i,2}$ and four of $N_{,3}$ were needed such that (6 + 1) for the first, (5 + 2) for the second and (4 + 3) for the last are always equal to 7 $(n + k)$. Second, whenever the limits of u for any $N_{i,1}$ are equal, the u range becomes one point.

Example 4.22 Find the equation of a closed (periodic) B-spline curve defined by four control points.

Solution This closed cubic spline has $k = 4$, $n = 3$. Using Eqs. (4.111) to (4.113), the knot vector $[u_0 \quad u_1 \quad u_2 \quad u_3 \quad u_4]$ is the integers [0 1 2 3 4] and the range of u is $0 \le u \le 4$. Equation (4.103) gives the curve equation as

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4} + \mathbf{P}_1 N_{1,4} + \mathbf{P}_2 N_{2,4} + \mathbf{P}_3 N_{3,4}, \quad 0 \le u \le 4 \tag{4.116}$$

To calculate the above B-spline functions, use Eq. (4.110) to obtain

$$N_{0,4}(u) = N_{0,4}((u + 4) \bmod 4)$$
$$N_{1,4}(u) = N_{0,4}((u + 3) \bmod 4)$$
$$N_{2,4}(u) = N_{0,4}((u + 2) \bmod 4)$$
$$N_{3,4}(u) = N_{0,4}((u + 1) \bmod 4)$$

In the above equations, $N_{0,4}$ on the right-hand side is the function for the open curve and on the left-hand side is the periodic function for the closed curve. Substituting these equations into Eq. (4.116) we get

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4}((u + 4) \bmod 4) + \mathbf{P}_1 N_{0,4}((u + 3) \bmod 4)$$
$$+ \mathbf{P}_2 N_{0,4}((u + 2) \bmod 4) + \mathbf{P}_3 N_{0,4}((u + 1) \bmod 4), \quad 0 \le u \le 4 \tag{4.117}$$

In Eq. (4.117), the function $N_{0,4}$ has various arguments, which can be found if specific values of u are used. To find $N_{0,4}$, similar calculations to the previous example 4.21 are performed using the above knot vector as follows:

$$N_{0,1} = \begin{cases} 1, & 0 \le u \le 1 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{1,1} = \begin{cases} 1, & 1 \le u \le 2 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{2,1} = \begin{cases} 1, & 2 \le u \le 3 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{3,1} = \begin{cases} 1, & 3 \le u \le 4 \\ 0, & \text{elsewhere} \end{cases}$$

$$N_{0,2} = (u - u_0)\frac{N_{0,1}}{u_1 - u_0} + (u_2 - u)\frac{N_{1,1}}{u_2 - u_1} = uN_{0,1} + (2 - u)N_{1,1}$$

$$N_{1,2} = (u-u_1)\frac{N_{1,1}}{u_2-u_1} + (u_3-u)\frac{N_{2,1}}{u_3-u_2} = (u-1)N_{1,1} + (3-u)N_{2,1}$$

$$N_{2,2} = (u-u_2)\frac{N_{2,1}}{u_3-u_2} + (u_4-u)\frac{N_{3,1}}{u_4-u_3} = (u-2)N_{2,1} + (4-u)N_{3,1}$$

$$N_{0,3} = (u-u_0)\frac{N_{0,2}}{u_2-u_0} + (u_3-u)\frac{N_{1,2}}{u_3-u_1} = \frac{1}{2}uN_{0,2} + \frac{1}{2}(3-u)N_{1,2}$$

$$= \frac{1}{2}u^2N_{0,1} + \frac{1}{2}[u(2-u) + (3-u)(u-1)]N_{1,1} + \frac{1}{2}(3-u)^2N_{2,1}$$

$$N_{1,3} = (u-u_1)\frac{N_{1,2}}{u_3-u_1} + (u_4-u)\frac{N_{2,2}}{u_4-u_2} = \frac{1}{2}(u-1)\,N_{1,2}\,\frac{1}{2}(4-u)N_{2,2}$$

$$= \frac{1}{2}(u-1)^2N_{1,1} + \frac{1}{2}[(u-1)(3-u) + (u-2)(4-u)]N_{2,1} + \frac{1}{2}(4-u)^2N_{3,1}$$

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

$$= \frac{1}{6}\{u^3N_{0,1} + [u^2(2-u) + u(3-u)(u-1) + (4-u)(u-1)^2]N_{1,1}$$

$$+ [u(3-u)^2 + (4-u)(u-1)(3-u) + (4-u)^2(u-2)]N_{2,1} + (4-u)^3N_{3,1}\}$$

or

$$N_{0,4} = \frac{1}{6}[u^3N_{0,1} + (-3u^3 + 12u^2 - 12u + 4)N_{1,1} + (3u^2 - 24u^2 + 60u - 44)N_{2,1}$$

$$+ (-u^3 + 12u^2 - 48u + 64)N_{3,1}]$$

Due to the non-zero values of the functions $N_{i,1}$ for various intervals of u, the above equation can be written as

$$N_{0,4}(u) = \begin{cases} \frac{1}{6}u^3, & 0 \le u \le 1 \\ \frac{1}{6}(-3u^3 + 12u^2 - 12u + 4), & 1 \le u \le 2 \\ \frac{1}{6}(3u^3 - 24u^2 + 60u - 44), & 2 \le u \le 3 \\ \frac{1}{6}(-u^3 + 12u^2 - 48u + 64), & 3 \le u \le 4 \end{cases} \tag{4.118}$$

To check the correctness of the above expression of $N_{0,4}(u)$, one would expect to obtain Fig. 4.54*c* if this function is plotted. Indeed, this figure is the plot of $N_{0,4}$. If $u = 0$, 1, 2, 3 and 4 are substituted into this function, the corresponding values of $N_{0,4}$ that are shown in the figure are obtained.

Equations (4.117) and (4.118) together can be used to evaluate points on the closed B-spline curve for display or plotting purposes. As an illustration, consider the following points:

$$\begin{aligned}\mathbf{P}(0) &= \mathbf{P}_0 N_{0,4}(4 \bmod 4) + \mathbf{P}_1 N_{0,4}(3 \bmod 4) \\ &\quad + \mathbf{P}_2 N_{0,4}(2 \bmod 4) + \mathbf{P}_3 N_{0,4}(1 \bmod 4) \\ &= \mathbf{P}_0 N_{0,4}(0) + \mathbf{P}_1 N_{0,4}(3) + \mathbf{P}_2 N_{0,4}(2) + \mathbf{P}_3 N_{0,4}(1) \\ &= \frac{1}{6}\mathbf{P}_1 + \frac{2}{3}\mathbf{P}_2 + \frac{1}{6}\mathbf{P}_3.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbf{P}(0.5) &= \mathbf{P}_0 N_{0,4}(0.5) + \mathbf{P}_1 N_{0,4}(3.5) + \mathbf{P}_2 N_{0,4}(2.5) + \mathbf{P}_3 N_{0,4}(1.5) \\ &= \frac{1}{48}\mathbf{P}_0 + \frac{1}{48}\mathbf{P}_1 + \frac{23}{48}\mathbf{P}_2 + \frac{23}{48}\mathbf{P}_3\end{aligned}$$

$$\begin{aligned}\mathbf{P}(1) &= \mathbf{P}_0 N_{0,4}(1) + \mathbf{P}_1 N_{0,4}(0) + \mathbf{P}_2 N_{0,4}(3) + \mathbf{P}3\, N_{0,4}(2) \\ &= \frac{1}{6}\mathbf{P}_0 + \frac{1}{6}\mathbf{P}_2 + \frac{2}{3}\mathbf{P}_3\end{aligned}$$

$$\mathbf{P}(2) = \mathbf{P}_0 N_{0,4}(2) + \mathbf{P}_1 N_{0,4}(1) + \mathbf{P}_2 N_{0,4}(0) + \mathbf{P}_3 N_{0,4}(3) = \frac{2}{3}\mathbf{P}_0 + \frac{1}{6}\mathbf{P}_1 + \frac{1}{6}\mathbf{P}_3$$

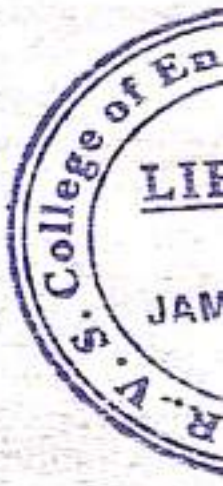
$$\mathbf{P}(3) = \mathbf{P}_0 N_{0,4}(3) + \mathbf{P}_1 N_{0,4}(2) + \mathbf{P}_2 N_{0,4}(1) + \mathbf{P}_3 N_{0,4}(0) = \frac{1}{6}\mathbf{P}_0 + \frac{2}{3}\mathbf{P}_1 + \frac{1}{6}\mathbf{P}_2$$

$$\mathbf{P}(4) = \mathbf{P}_0 N_{0,4}(0) + \mathbf{P}_1 N_{0,4}(3) + \mathbf{P}_2 N_{0,4}(2) + \mathbf{P}_3 N_{0,4}(1) = \frac{1}{6}\mathbf{P}_0 + \frac{2}{3}\mathbf{P}_2 + \frac{1}{6}\mathbf{P}_3$$

In the above calculations, notice the cyclic rotation of the $N_{0,4}$ coefficients of the control points for the various values of u excluding $u = 0.5$. Notice also the effect of the canonical (symmetric) form of $N_{0,4}$ on the coefficients of the control points. If the u values are 0.5, 1.5, 2.5 and 3.5, or other values separated by unity, a similar cyclic rotation of the coefficients is expected. Finally, notice that $P(0)$ and $P(4)$ are equal, which ensures obtaining a closed B-spline curve.

The theory of the B-spline has been extended further to allow more control of the curve shape and continuity. For example, β-spline (beta-spline) and ν-spline (nu-spline) curves provide manipulation of the curve shape and maintain its geometric continuity rather than its parametric continuity as provided by B-spline curves. The β-spline (sometimes called the spline in tension) curve is a generalization of the uniform cubic B-spline curve. The β-spline curve provides the designer with two additional parameters: the bias and the tension to control the shape of the curve. Therefore, the control points and the degree of the β-spline curve can remain fixed and yet the curve shape can be manipulated.

Although the β-spline curve is capable of applying tension at each control point, its formulation as piecewise hyperbolic sines and cosines makes its computation expensive. The ν-spline curve is therefore developed as a piecewise polynomial alternative to the spline in tension.



4.6.4 Rational Curves

A rational curve is defined by the algebraic ratio of two polynomials while a nonrational curve [Eq. (4.103) gives an example] is defined by one polynomial. Rational curves draw their theories from projective geometry. They are important because of their invariance under projective transformation; that is, the perspective image of a rational curve is a rational curve. Rational Bezier curves, rational B-spline and β-spline curves, rational conic sections, rational cubics and rational surfaces have been formulated. The most widely used rational curves are NURBS (nonuniform rational B-splines). A brief description of rational B-spline curves is given below.

The formulation of rational curves requires the introduction of homogeneous space and the homogeneous coordinates. This subject is covered in detail in Chap. 8 (Sec. 8.2.5). The homogeneous space is four-dimensional space. A point in E^3 with coordinates (x, y, z) is represented in the homogeneous space by the coordinates (x^*, y^*, z^*, h), where h is a scalar factor. The relationship between the two types of coordinates is given by Eq. (8.59).

A rational B-spline curve defined by $n + 1$ control points P_i is given by

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i R_{i,k}(u), \quad 0 \le u \le u_{\max} \tag{4.119}$$

$R_{i,k}(u)$ are the rational B-spline basis functions and are given by

$$R_{i,k}(u) = \frac{h_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)} \tag{4.120}$$

The above equation shows that $R_{i,k}(u)$ are a generalization of the nonrational basis functions $N_{i,k}(u)$. If we substitute $h_i = 1$ in the equation, $R_{i,k}(u) = N_{i,k}(u)$. The rational basis functions $R_{i,k}(u)$ have nearly all the analytic and geometric characteristics of their nonrational B-spline counterparts. All the discussions covered in Sec. 4.6.3 apply here.

The main difference between rational and nonrational B-spline curves is the ability to use h_i at each control point to control the behavior of the rational B-splines (or rational curves in general). Thus, similarly to the knot vector, one can define a homogeneous coordinate vector $H = [h_0 \quad h_1 \quad h_2 \quad h_3 \ldots h_n]^T$ at the control points $P_0, P_1, \ldots, P_n$ of the rational B-spline curve. The choice of the H vector controls the behavior of the curve.

A rational B-spline is considered a unified representation that can define a variety of curves and surfaces. The premise is that it can represent all wireframe, surface and solid entities. This allows unification and conversion from one modeling technique to another. Such an approach has some drawbacks, including the loss of information on simple shapes. For example, if a circular cylinder (hole) is represented by a B-spline, some data on the specific curve type may be lost unless it is carried along. Data including the fact that the part feature was a cylinder would be useful to manufacturing to identify it as a hole to be drilled or bored rather than a surface to be milled.

are useful when solving surface intersection problems. Based on Eq. (5.1), the z coordinate of a point on the sphere is to be rewritten as a function of x and y. Squaring and adding the first two equations give

$$\frac{(x-x_0)^2}{R^2}+\frac{(y-y_0)^2}{R^2}=\cos^2 u=1-\sin^2 u$$

which gives

$$\sin u=\sqrt{1-\frac{(x-x_0)^2}{R^2}-\frac{(y-y_0)^2}{R^2}}$$

By substituting $\sin u$ into the z coordinate equation, $f(x, y)$ for a sphere becomes

$$z=z_0+\sqrt{R^2-(x-x_0)^2-(y-y_0)^2}$$

The sphere implicit equation can now be written as

$$\mathbf{P}=\left[x \quad y \quad z_0+\sqrt{R^2-(x-x_0)^2-(y-y_0)^2}\right]^T$$

It is obvious that the square of the z coordinate gives the classical sphere equation:

$$(x-x_0)^2+(y-y_0)^2+(z-z_0)^2=R^2$$

5.5 ≡ PARAMETRIC REPRESENTATION OF ANALYTIC SURFACES

This section covers the basics of the parametric equations of analytic surfaces most often encountered in surface modeling and design. The background provided in this section and the next one should enable users of CAD/CAM technology to realize the limitations of each surface available to them for modeling and design as well as cope with the documentation of surface commands.

The distinction between the WCS and MCS has been ignored in the development of the parametric equations of the various surfaces to avoid confusion. The transformation between the two systems is obvious and has already been discussed in Chap. 4. In addition, the terms "surface" and "patch" are used interchangeably in this chapter. However, in a more general sense a surface is considered the superset since a surface can contain one or more patch.

5.5.1 Plane Surface

The parametric equation of a plane can take different forms depending on the given data. Consider first the case of a plane defined by three points P_0, P_1 and P_2 as shown in Fig. 5.22. Assume that the point P_0 defines $u = 0$ and $v = 0$ and the vectors $(\mathbf{P}_1-\mathbf{P}_0)$ and $(\mathbf{P}_2-\mathbf{P}_0)$ define the u and v directions respectively. Assume also that the domains for u and v are [0, 1]. The position vector of any point P on the plane can be now written as

$$\mathbf{P}(u, v)=\mathbf{P}_0+u(\mathbf{P}_1-\mathbf{P}_0)+v(\mathbf{P}_2-\mathbf{P}_0), \qquad 0\le u\le 1, 0\le v\le 1 \qquad (5.25)$$

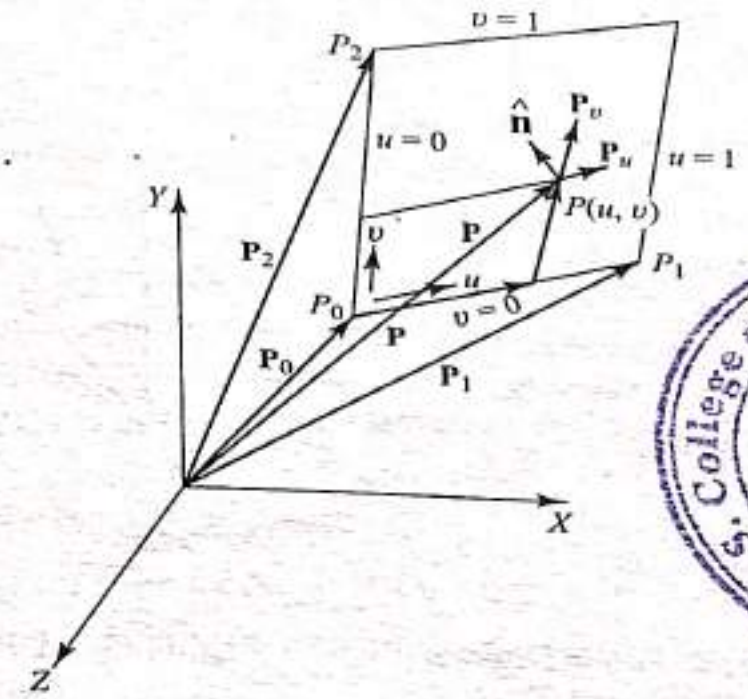


Fig. 5.22 *A Plane Patch Defined by Three Points*

The above equation can be seen as the bilinear form of Eq. (5.11). Utilizing Eqs. (5.4) and (5.5), the tangent vectors at point P are

$$\mathbf{P}_u(u, v) = \mathbf{P}_1 - \mathbf{P}_0 \qquad \mathbf{P}_v(u, v) = \mathbf{P}_2 - \mathbf{P}_0, \qquad 0 \le u \le 1, 0 \le v \le 1 \quad (5.26)$$

and the surface normal is

$$\hat{\mathbf{n}}(u, v) = \frac{(\mathbf{P}_1 - \mathbf{P}_0) \times (\mathbf{P}_2 - \mathbf{P}_0)}{|(\mathbf{P}_1 - \mathbf{P}_0) \times (\mathbf{P}_2 - \mathbf{P}_0)|}, \quad 0 \le u \le 1, 0 \le v \le 1 \qquad (5.27)$$

which is constant for any point on the plane. As for the curvature of the plane, it is equal to zero [see Eq. (5.20)] because all the second fundamental coefficients of the plane are zeros [see Eq. (5.19)].

Another case of constructing a plane surface is when the surface passes through a point P_0 and contains two directions defined by the unit vectors $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$ as shown in Fig. 5.23. Similar to the above case, the plane equation can be written as

$$\mathbf{P}(u, v) = \mathbf{P}_0 + uL_u\hat{\mathbf{r}} + vL_v\hat{\mathbf{s}}, \qquad 0 \le u \le 1, 0 \le v \le 1 \qquad (5.28)$$

This equation is also considered as the bilinear form of Eq. (5.17). The equation assumes a plane of dimensions L_u and L_v that may be set to unity.

The above two cases can be combined to provide the equation of a plane surface that passes through two points P_0 and P_1 and is parallel to the unit vector $\hat{\mathbf{r}}$. In this case, we can write

$$\mathbf{P}(u, v) = \mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0) + vL_v\hat{\mathbf{r}}, \qquad 0 \le u \le 1, 0 \le v \le 1 \qquad (5.29)$$

The last case to be considered is for a plane that passes through a point P_0 and is perpendicular to a given direction $\hat{\mathbf{n}}$. Figure. 5.24 shows this case. The vector $\hat{\mathbf{n}}$ is normal to any vector in the plane. Thus,

$$(\mathbf{P} - \mathbf{P}_0) \cdot \hat{\mathbf{n}} = 0 \qquad (5.30)$$

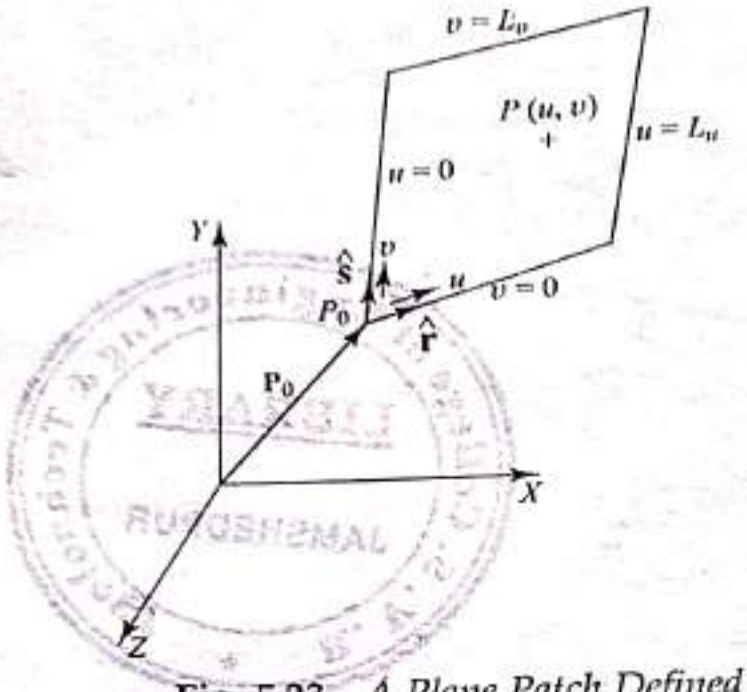


Fig. 5.23 *A Plane Patch Defined by a Point and Two Directions*

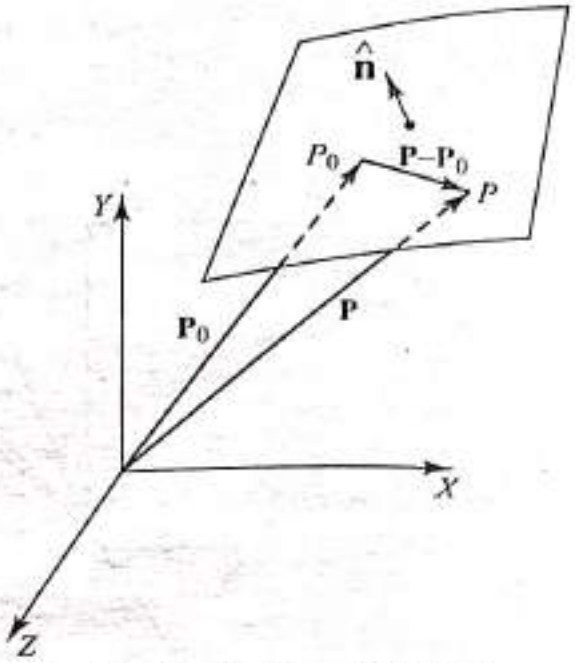


Fig. 5.24 *A Plane Patch Passing through Point P_0 and Normal to $\hat{\mathbf{n}}$.*

which is a nonparametric equation of the plane surface. A parametric equation can be developed by using Eq. (5.30) to generate two points on the surface which can be used with P_0 in Eq. (5.25). Planes that are perpendicular to the axes of a current WCS are special cases of Eq. (5.30). For example, in the case of a plane perpendicular to the X axis, $\hat{\mathbf{n}}$ is (1, 0, 0) and the plane equation is $x = x_0$.

A database structure of a plane surface can be seen to include its unit normal $\hat{\mathbf{n}}$, a point on the plane P_0 and u and v axes defined in terms of the MCS coordinates. For example, if a plane passes through the points $P_0(0, 0, 0)$, $P_1(2, 0, 0)$ and $P_2(0, 0, 2)$ as shown in Fig. 5.25, the verification of the plane surface entities shows the entity is a plane passing through $P_0(0, 0, 0)$ and has a unit normal of $(0, -1, 0)$. In addition, the u and v axes are defined by the coordinates (1, 0, 0) and (0, 0, 1) respectively.

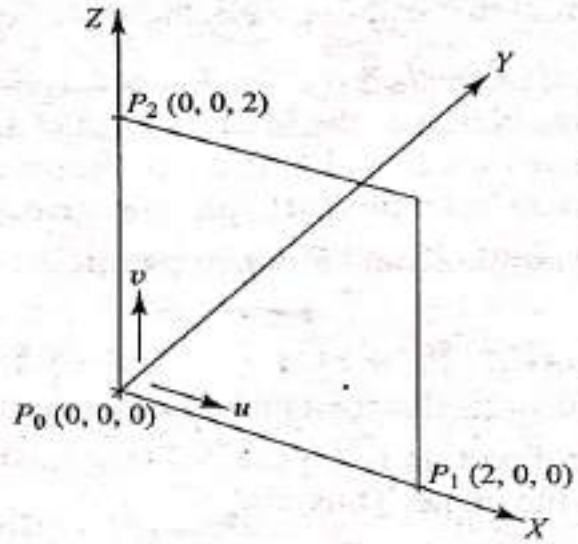


Fig. 5.25 *Plane Surface Construction*

Example 5.3 Find the minimum distance between a point in space and a plane surface.

Solution The minimum distance between a point and a plane is also the perpendicular distance from the point onto the plane. Let us assume the plane equation is given by

$$\mathbf{P} = \mathbf{P}_0 + u\hat{\mathbf{r}} + v\hat{\mathbf{s}}, \qquad 0 \le u \le 1, 0 \le v < 1 \tag{5.31}$$

This assumption can be made in the light of the database structure described above. From Fig. 5.26, it is obvious that the perpendicular vector from point Q to the plane is parallel to its normal $\hat{\mathbf{n}}$. Thus, we can write

$$\mathbf{P} = \mathbf{Q} - D\hat{\mathbf{n}} \tag{5.32}$$

By using Eq. (5.31) in (5.32), we get

$$\mathbf{P}_0 + u\hat{\mathbf{r}} + v\hat{\mathbf{s}} = \mathbf{Q} - D\hat{\mathbf{n}} \tag{5.33}$$

Equation (5.33) can be rewritten in a matrix form as

$$\begin{bmatrix} r_x & s_x & n_x \\ r_y & s_y & n_y \\ r_z & s_z & n_z \end{bmatrix} \begin{bmatrix} u \\ v \\ D \end{bmatrix} = \begin{bmatrix} x_Q - x_0 \\ y_Q - y_0 \\ z_Q - z_0 \end{bmatrix} \tag{5.34}$$

where r_x, r_y and r_z are the components of the unit vector $\hat{\mathbf{r}}$. Similarly, the components of the other vectors are given in Eq. (5.34). Solving Eq. (5.34) (see Chap. 4) gives the normal distance D and u and v, which can give the point P.

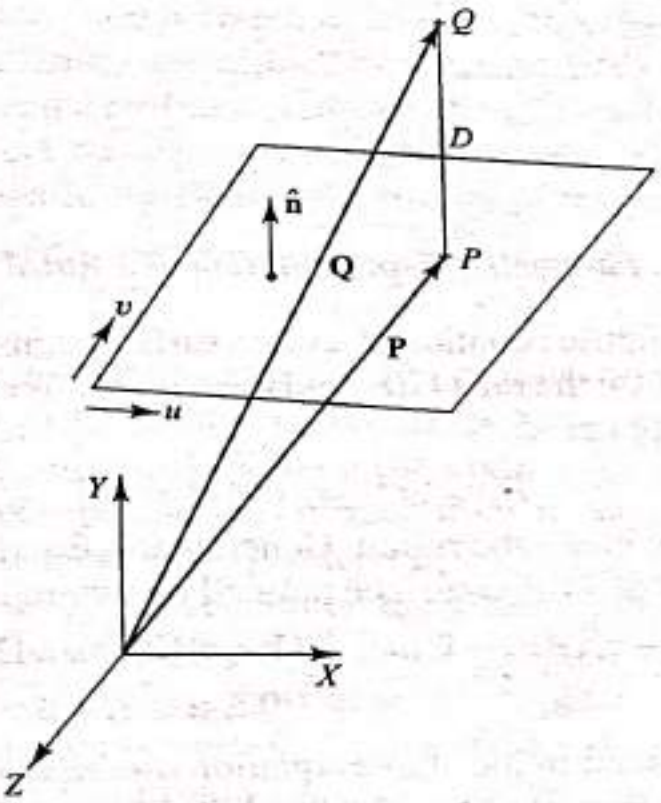


Fig. 5.26 *Minimum Distance between a Point and Plane Surface*

5.5.2 Ruled Surface

A ruled surface is generated by joining corresponding points on two space curves (rails) $\mathbf{G}(u)$ and $\mathbf{Q}(u)$ by straight lines (also called rulings or generators), as shown in Fig. 5.27. The main characteristic of a ruled surface is that there is at least one straight line passing through the point $P(u, v)$ and lying entirely in the surface. In addition, every developable surface is a ruled surface. Cones and cylinders are examples of ruled surfaces and the plane surface covered in Sec. 5.5.1 is considered the simplest of all ruled surfaces.

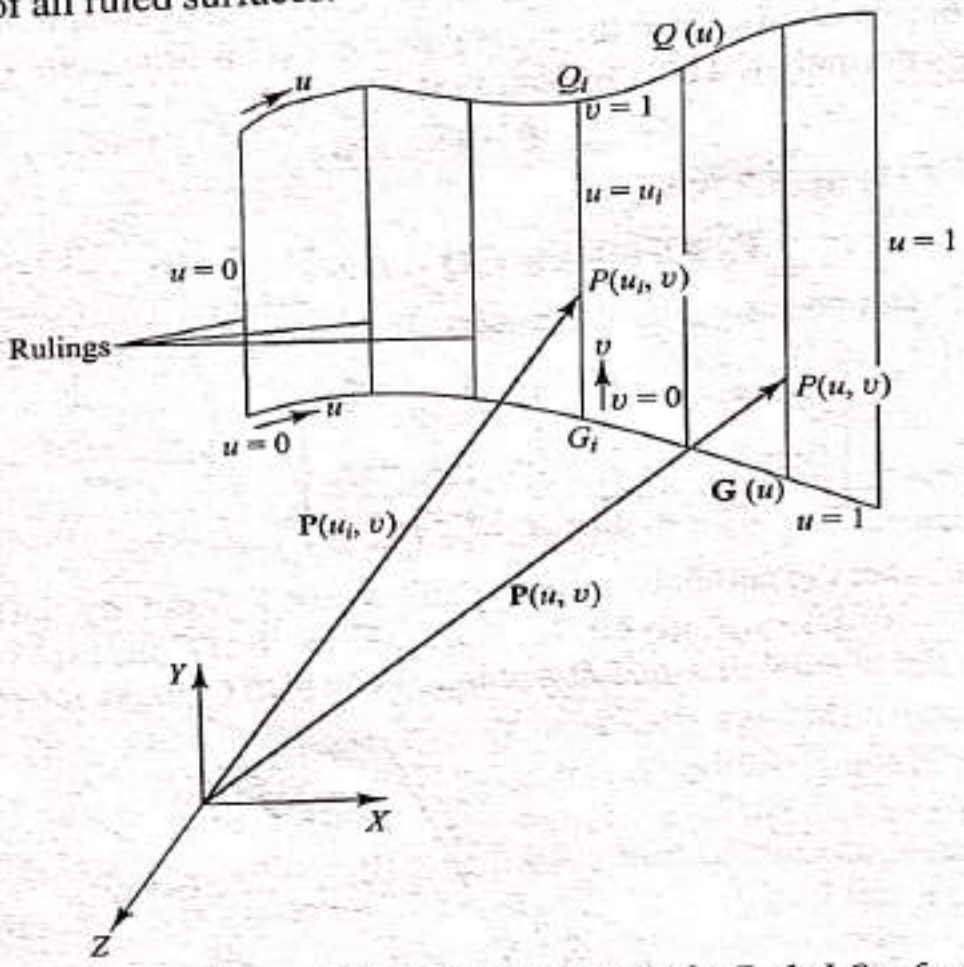


Fig. 5.27 *Parametric Representation of a Ruled Surface*

To develop the parametric equation of a ruled surface, consider the ruling $u = u_i$ joining points G_i and Q_i on the rails $\mathbf{G}(u)$ and $\mathbf{Q}(u)$ respectively. Using Eq. (5.11), the equation of the ruling becomes

$$\mathbf{P}(u_i, v) = \mathbf{G}_i + v(\mathbf{Q}_i - \mathbf{G}_i) \tag{5.35}$$

where v is the parameter along the ruling. Generalizing Eq. (5.35) for any ruling, the parametric equation of a ruled surface defined by two rails is

$$\mathbf{P}(u, v) = \mathbf{G}(u) + v[\mathbf{Q}(u) - \mathbf{G}(u)] = (1 - v)\mathbf{G}(u) + v\mathbf{Q}(u),$$
$$0 \le u \le 1, 0 \le v \le 1 \tag{5.36}$$

Holding the u value constant in the above equation produces the rulings given by Eq. (5.35) in the v direction of the surface, while holding the v value constant yields curves in the u direction which are a linear blend of the rails. In fact, $\mathbf{G}(u)$

and $\mathbf{Q}(u)$ are $\mathbf{P}(u, 0)$ and $\mathbf{P}(u, 1)$ respectively. Therefore, the closer the value of v to zero, the greater the influence of $\mathbf{G}(u)$ and the less the influence of $\mathbf{Q}(u)$ on the v = constant curve. Similarly, the influence of $\mathbf{Q}(u)$ on the ruled surface geometry increases when the v value approaches unity (see Fig. 5.1).

Based on Fig. 5.27 and Eq. (5.35), it is now obvious why digitizing the wrong ends of the rails produces the undesirable ruled surface as shown in Fig. 5.1*a*. In addition, Eq. (5.36), together with Eqs. (5.15), (5.19) and (5.20), shows that a ruled surface can only allow curvature in the u direction of the surface provided that the rails have curvatures. The surface curvature in the v direction (along the rulings) is zero and thus a ruled surface cannot be used to model surface patches that have curvatures in two directions.

5.5.3 Surface of Revolution

The rotation of a planar curve an angle v about an axis of rotation creates a circle (if $v = 360$) for each point on the curve whose center lies on the axis of rotation and whose radius $r_z(u)$ is variable, as shown in Fig. 5.28. The planar curve and the circles are called the profile and parallels respectively while the various positions of the profile around the axis are called meridians.

The planar curve and the axis of rotation form the plane of zero angle, that is, $v = 0$. To derive the parametric equation of a surface of revolution, a local *coordinate* system with a Z axis coincident with the axis of rotation is assumed as shown in Fig. 5.28. This local system shown by the subscript L can be created as follows. Choose the perpendicular direction from the point $u = 0$ on the profile as the X_L axis and the intersection point between X_L and Z_L as the origin of the local system. The Y_L axis is automatically determined by the right-hand rule. Now, consider a point $\mathbf{G}(u) = \mathbf{P}(u, 0)$ on the profile that rotates an angle v about Z_L when the profile rotates the same angle. Considering the shaded triangle which is perpendicular to the Z_L axis, the parametric equation of the surface of revolution can be written as

$$\mathbf{P}(u, v) = r_z(u) \cos v\, \hat{\mathbf{n}}_1 + r_z(u) \sin v\, \hat{\mathbf{n}}_2 + z_L(u)\, \hat{\mathbf{n}}_3,$$
$$0 \le u \le 1, 0 \le v \le 2\pi \qquad (5.37)$$

If we choose $z_L(u) = u$ for each point on the profile, Eq. (5.37) gives the local coordinates (x_L, y_L, z_L) of a point $P(u, v)$ as $[r_z(u) \cos v, r_z(u) \sin v, u]$. The local coordinates are transformed to MCS coordinates before displaying the surface using Eq. (3.3) where the rotation matrix is formed from $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$ and $\hat{\mathbf{n}}_3$ and the position of the origin of the local system is given by P_L (see Fig. 5.28).

The database of a surface of revolution must include its profile, axis of rotation and the angle of rotation as starting and ending angles. Whenever the user requests the display of the surface with a mesh size $m \times n$, the u range is divided equally into $(m - 1)$ divisions and m values of u are obtained. Similarly, the v range is divided equally into n values and Eq. (5.37) is used to generate points on the surface.

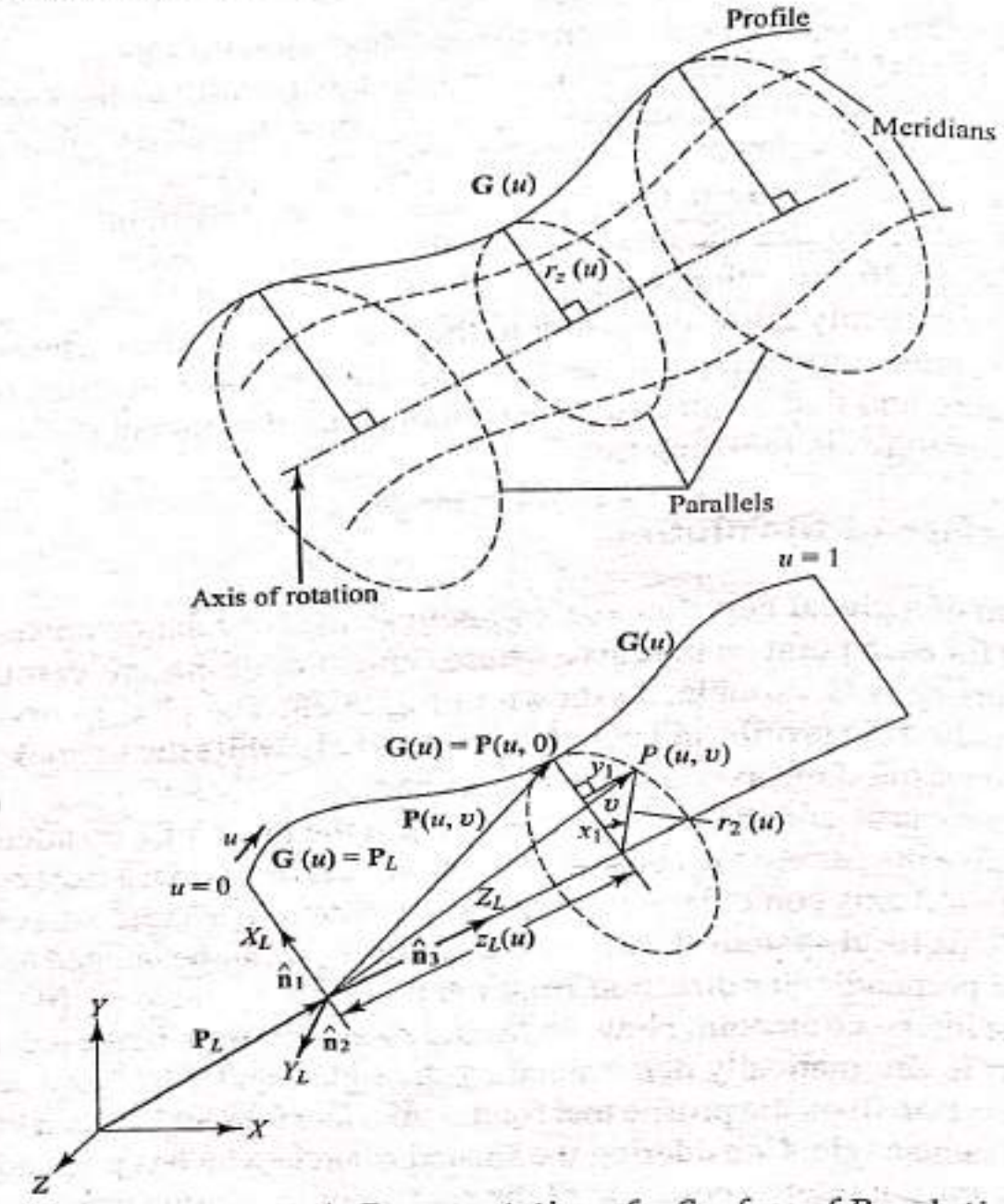


Fig. 5.28 *Parametric Representation of a Surface of Revolution*

5.5.4 Tabulated Cylinder

A tabulated cylinder has been defined as a surface that results from translating a space planar curve along a given direction. It can also be defined as a surface that is generated by moving a straight line (called generatrix) along a given planar curve (called directrix). The straight line always stays parallel to a fixed given vector that defines the v direction of the cylinder as shown in Fig. 5.29. The planar curve $\mathbf{G}(u)$ can be any wireframe entities. The position vector of any point $P(u, v)$ on the surface can be written as

$$\mathbf{P}(u, v) = \mathbf{G}(u) + v\,\hat{\mathbf{n}}_v, \qquad 0 \le u \le u_{max},\ 0 \le v \le v_{max} \tag{5.38}$$

From a user point of view, $\mathbf{G}(u)$ is the desired curve the user digitizes to form the cylinder, v is the cylinder length and $\hat{\mathbf{n}}_v$ is the cylinder axis. The representation of $\mathbf{G}(u)$ is already available in the database at the time of creating it. The cylinder length v is input in the form of lower and higher bounds where the difference

between them gives the length. A zero value of the lower bound indicates the plane of the directrix. The user inputs the cylinder axis as two points that are used to determine $\hat{\mathbf{n}}_v$ which is the unit vector along the axis.

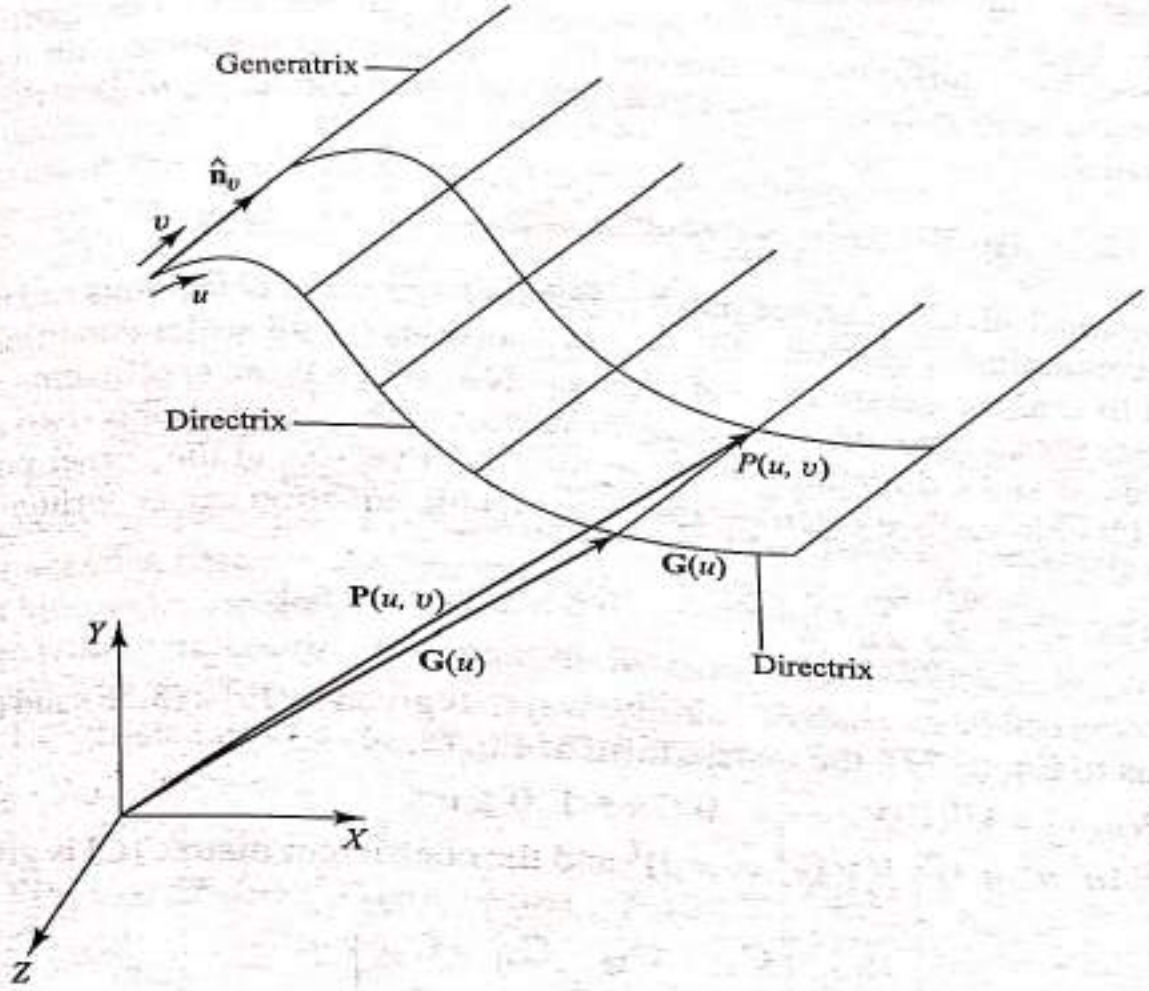


Fig. 5.29 *Parametric Representation of a Tabulated Cylinder*

As seen from Eq. (5.38), the database of a tabulated cylinder includes its directrix, the unit vector $\hat{\mathbf{n}}_v$ and the lower and upper bounds of the cylinder. The display of a tabulated cylinder with a mesh $m \times n$ follows the same approach as discussed with surfaces of revolution.

5.6 PARAMETRIC REPRESENTATION OF SYNTHETIC SURFACES

The arguments regarding the needs for synthetic curves discussed in Sec. 4.6 of Chap. 4 apply here. Synthetic surfaces provide designers with better surface design tools than analytic surfaces. Consider the design of blade surfaces in jet aircraft engines. The design of these surfaces is usually based on aerodynamic and fluid-flow simulations, often incorporating thermal and mechanical stress deformation. These simulations yield ordered sets of discrete streamline points which must then be connected accurately by surfaces. Any small deviation in these aerodynamic surfaces can significantly degrade performance. Another example is the creation of blending surfaces typically encountered in designing dies for injection molding of plastic products.

For continuity purposes, the parametric representation of synthetic surfaces is presented below in a similar form to curves. Surfaces covered are bicubic, Bezier, B-spline, Coons, blending offset, triangular, sculptured and rational surfaces. All these surfaces are based on polynomial forms. Surfaces using other forms such as Fourier series are not considered here. Although Fourier series can approximate any curve given sufficient conditions, the computations involved with them are greater than with polynomials. Therefore, they are not suited to general use in CAD/CAM.

5.6.1 Hermite Bicubic Surface

The parametric bicubic surface patch connects four corner data points and utilizes a bicubic equation. Therefore, 16 vector conditions (or 48 scalar conditions) are required to find the coefficients of the equation. When these coefficients are the four corner data points, the eight tangent vectors at the corner points (two at each point in the u and v directions) and the four twist vectors at the corner points, a Hermite bicubic surface patch results. The bicubic equation can be written as

$$\mathbf{P}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} C_{ij} u^i v^j, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{5.39}$$

This equation can be expanded in similar ways, as given by Eqs. (5.75) and (5.76). Analogous to Eq. (5.77), the matrix form of Eq. (5.39) is

$$\mathbf{P}(u, v) = \mathbf{U}^T[C]\mathbf{V}, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{5.40}$$

where $U = [u^3\ u^2\ u\ 1]^T$, $V = [v^3\ v^2\ v\ 1]^T$ and the coefficient matrix $[C]$ is given by

$$[C] = \begin{bmatrix} \mathbf{C}_{33} & \mathbf{C}_{32} & \mathbf{C}_{31} & \mathbf{C}_{30} \\ \mathbf{C}_{23} & \mathbf{C}_{22} & \mathbf{C}_{21} & \mathbf{C}_{20} \\ \mathbf{C}_{13} & \mathbf{C}_{12} & \mathbf{C}_{11} & \mathbf{C}_{10} \\ \mathbf{C}_{03} & \mathbf{C}_{02} & \mathbf{C}_{01} & \mathbf{C}_{00} \end{bmatrix} \tag{5.41}$$

In order to determine the coefficients $\mathbf{C}_i$, consider the patch shown in Fig. 5.18. Applying the boundary conditions into Eq. (5.40), solving for the coefficients and rearranging give the following final equation of a bicubic patch:

$$\mathbf{P}(u, v) = \mathbf{U}^T[M_H][B][M_H]^T\mathbf{V}, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{5.42}$$

where $[M_H]$ is given by Eq. (4.84) and $[B]$, the geometry or boundary condition matrix, is

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{v00} & \mathbf{P}_{v01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{v10} & \mathbf{P}_{v11} \\ \hdashline \mathbf{P}_{u00} & \mathbf{P}_{u01} & \mathbf{P}_{uv00} & \mathbf{P}_{uv01} \\ \mathbf{P}_{u10} & \mathbf{P}_{u11} & \mathbf{P}_{uv10} & \mathbf{P}_{uv11} \end{array}\right] \tag{5.43}$$

The matrix $[B]$ is partitioned as shown above to indicate the grouping of the similar boundary conditions. It can also be written as

$$[B] = \left[\begin{array}{c|c} [P] & [P_v] \\ \hline [P_u] & [P_{uv}] \end{array}\right] \tag{5.44}$$

where $[P]$, $[P_u]$, $[P_v]$ and $[P_{uv}]$ are the submatrices of the corner points, corner u-tangent vectors, corner v-tangent vectors and the corner twist vectors respectively. The tangent and twist vectors at any point on the surface are given by

$$\mathbf{P}_u(u, v) = \mathbf{U}^T[M_H]^u[B]M_H]^T\mathbf{V} \tag{5.45}$$

$$\mathbf{P}_u(u, v) = \mathbf{U}^T[M_H][B]M_H]^{vT}\mathbf{V} \tag{5.46}$$

$$\mathbf{P}_{uv}(u, v) = \mathbf{U}^T[M_H]^u[B]M_H]^{vT}\mathbf{V} \tag{5.47}$$

where $[M_H\}^u$ or $[MH]^v$ is given by Eq. (4.88).

Similar to the cubic spline, the bicubic form permits C^1 continuity from one patch to the next. The necessary two conditions are to have the same curves (C^0 continuity) and the same direction of the tangent vectors (C^1 continuity) across the common edge between the two patches. The magnitude of the tangent vectors does not have to be the same.

Before writing the continuity conditions in terms of the $[B]$ matrix, let us expand Eqs. (5.42) and (5.45) to (5.47) to see what influences the position and tangent vectors. Equations (5.42) and (5.45) to (5.47) give

$$\mathbf{P}(u, v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)][B]\begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{5.48}$$

$$\mathbf{P}_u(u, v) = [G_1(u) \quad G_2(u) \quad G_3(u) \quad G_4(u)][B]\begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{5.49}$$

$$\mathbf{P}_v(u, v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)][B]\begin{bmatrix} G_1(v) \\ G_2(v) \\ G_3(v) \\ G_4(v) \end{bmatrix} \tag{5.50}$$

$$\mathbf{P}_{uv}(u, v) = [G_1(u) \quad G_2(u) \quad G_3(u) \quad G_4(u)][B]\begin{bmatrix} G_1(v) \\ G_2(v) \\ G_3(v) \\ G_4(v) \end{bmatrix} \tag{5.51}$$

where

$$F_1(x) = 2x^3 - 3x^2 + 1$$
$$F_2(x) = -2x^3 + 3x^2 \qquad (5.52)$$
$$F_3(x) = x^3 - 2x^2 + x$$
$$F_4(x) = x^3 - x^2$$

and

$$G_1(x) = 6x^2 - 6x$$
$$G_2(x) = -6x^2 + 6x \qquad (5.53)$$
$$G_3(x) = 3x^2 - 4x + 1$$
$$G_4(x) = 3x^2 - 2x$$

For $u = 0$ and $u = 1$ edges these equations become

$$\begin{bmatrix} \mathbf{P}(0, v) \\ \mathbf{P}(1, v) \\ \mathbf{P}_u(0, v) \\ \mathbf{P}_u(1, v) \end{bmatrix} = [B] \begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \qquad (5.54)$$

Similarly, for $v = 0$ and $v = 1$ edges we can write

$$\begin{bmatrix} \mathbf{P}(u, 0) \\ \mathbf{P}(u, 1) \\ \mathbf{P}_v(u, 0) \\ \mathbf{P}_v(u, 1) \end{bmatrix} = [B] \begin{bmatrix} F_1(u) \\ F_2(u) \\ F_3(u) \\ F_4(u) \end{bmatrix} \qquad (5.54)$$

Equation (5.54) shows that the corner v-tangent and twist vectors affect the position and tangent vectors respectively all along the $u = 0$ and $u = 1$ edges except at $v = 0$ and 1 where F_3 and F_4 are both zero.

To write the blending conditions for C^1 continuity between two patches, consider the surface shown in Fig. 5.16. These conditions to connect patch 1 and patch 2 along the u edges are

$$[\mathbf{P}(0, v)]_{\text{patch 2}} = [\mathbf{P}(1, v)]_{\text{patch 1}} \qquad C^0 \text{ continuity}$$
$$[\mathbf{P}_u(0, v)]_{\text{patch 2}} = K[\mathbf{P}_u(1, v)]_{\text{patch 1}} \qquad C^1 \text{ continuity} \qquad (5.56)$$

where K is a constant. Equation (5.56) can be interpreted as blending an infinite number of cubic spline segments on each patch (each corresponding to a particular v value) with C^1 continuity across the u edge. Utilizing Eq. (5.54), Eq. (5.56) can be expressed in terms of the rows of the $[B]$ matrix as shown in Fig. 5.30. The figure shows the constrained elements of each matrix only. Empty elements of each matrix are unconstrained and can have arbitrary values. It is left to the reader to find similar matrices to join the two patches at the $v = 1$ and $v = 0$ edges or any other combination.

The Ferguson surface (also called the F-surface patch) is considered a bicubic surface patch with zero twist vectors at the patch corners, as shown in this special

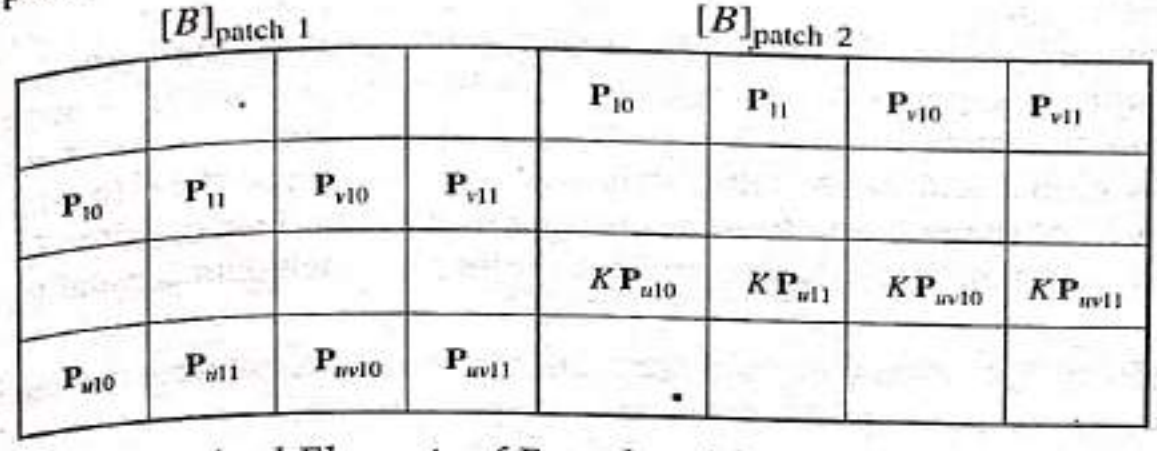


Fig. 5.30 *Constrained Elements of Boundary Matrix [B] to Blend Two Bicubic Patches along a u Edge*

Fig. 5.31. Thus, the boundary matrix for the F-surface patch becomes

$$[B] = \left[\begin{array}{cc|cc} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{v00} & \mathbf{P}_{v01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{v10} & \mathbf{P}_{v11} \\ \hline \mathbf{P}_{u00} & \mathbf{P}_{u01} & 0 & 0 \\ \mathbf{P}_{u10} & \mathbf{P}_{u11} & 0 & 0 \end{array}\right] \tag{5.57}$$

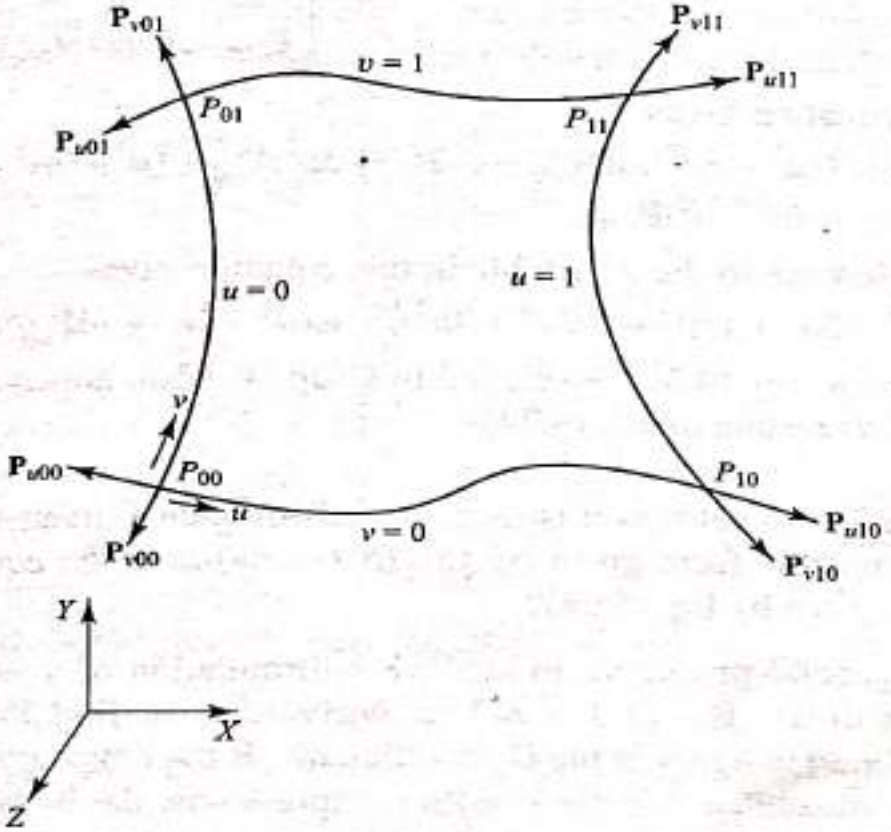


Fig. 5.31 *The F-surface Patch*

surface is useful in design and machining applications. The tangent vectors at the corner points can be approximated in terms of the corner positions using the direction and the length of chord lines joining the corner points. Hence, the designer does not

have to input tangent vector information and the computations required to calculate the surface parameters are simplified. This is useful if tool paths are to be generated to mill the surface.

The characteristics of the bicubic surface path are very similar to those of the cubic spline. The patch can be used to fit a bicubic surface to an array of data points homomorphic to an $m \times n$ rectangular grid. The control of the resulting surface is global and is not intuitively based on the input data. In addition, the requirement of tangent and twist vectors as input data does not fit very well the design environment because the intuitive feeling for such data is usually not clear.

Example ≡ 5.4 Show that a bicubic surface patch degenerates to a cubic spline if the four corner points of the patch are collapsed to two.

Solution Consider the surface patch shown in Fig. 5.18. Let us assume that the $v = 1$ edge coincides with the $v = 0$ edge. In this case, the corners P_{00} and P_{01} coincide and so do the corners P_{10} and P_{11}. All the derivatives with respect to v are set to zero and the u-tangent vectors at the coincident corners are equal to one another. Finally let us choose the value of v to equal 1. Substituting all these values into Eq. (5.48), we obtain

$$\mathbf{P}(u, 1) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{00} & 0 & 0 \\ \mathbf{P}_{10} & \mathbf{P}_{10} & 0 & 0 \\ \mathbf{P}_{u00} & \mathbf{P}_{u00} & 0 & 0 \\ \mathbf{P}_{u10} & \mathbf{P}_{u10} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Expanding the equation gives

$$\mathbf{P}(u, 1) = (2u^3 - 3u^2 + 1)\mathbf{P}_{00} + (-2u^3 + 3u^2)\mathbf{P}_{10} + (u^3 - 2u^2 + u)\mathbf{P}_{u00} + (u^3 - u^2)\mathbf{P}_{u10}$$

Dropping the reference to the v variable in this equation gives

$$\mathbf{P}(u) = (2u^3 - 3u^2 + 1)\mathbf{P}_0 + (-2u^3 + 3u^2)\mathbf{P}_1 + (u^3 - 2u^2 + u)\mathbf{P}_{u0} + (u^3 - u^2)P_{u1}$$

which is identical to Eq. (4.81) developed in Chap. 4. Also, notice that substituting $v = 1$ in Eq. (5.39) results in Eq. (4.74).

Example ≡ 5.5 Sometimes it is useful to reformulate a given surface equation in terms of the bicubic form given by Eq. (5.42). What is the equivalent bicubic patch to a plane given by Eq. (5.28)?

Solution The general procedure to achieve reformulation of a surface form to a bicubic form is to use Eq. (5.39) and its derivatives to find $\mathbf{P}(u, v)$, $\mathbf{P}_u(u, v)$, $\mathbf{P}_v(u, v)$ and $\mathbf{P}_{uv}(u, v)$ in terms of the $\mathbf{C}_{ij}$ coefficients. If the boundary values of u and v (0 and 1) are substituted into the resulting expressions, the boundary conditions (elements of $[B]$) of the patch can be written in terms of these coefficients. The resulting general functions take the form:

$$\mathbf{B}_{mn} = f(\mathbf{C}_{ij})$$

where $\mathbf{B}_{mn}$ is an element of $[B]$.

If the equation of a given surface is compared to Eq. (5.39), the corresponding set of C_{ij} coefficients are obtained which can in turn be substituted into the above equation to provide $[B]$ of the equivalent bicubic patch.

Let us apply the above general procedure to the plane given by Eq. (5.28). Comparing this equation to Eq. (5.39) gives

$$\mathbf{C}_{00} = \mathbf{P}_0$$

$$\mathbf{C}_{10} = \mathbf{L}_u \hat{\mathbf{r}}$$

$$\mathbf{C}_{01} = \mathbf{L}_v \hat{\mathbf{s}}$$

and all the other coefficients are zeros. The $[B]$ matrix then becomes

$$[B] = \left[\begin{array}{cc:cc} P_0 & P_0 + L_v\hat{s} & L_v\hat{s} & L_v\hat{s} \\ P_0 + L_u\hat{r} & P_0 + L_u\hat{r} + L_v\hat{s} & L_v\hat{s} & L_v\hat{s} \\ \hdashline L_u\hat{r} & L_u\hat{r} & 0 & 0 \\ L_u\hat{r} & L_u\hat{r} & 0 & 0 \end{array}\right]$$

Example ≣ 5.6 A bicubic surface patch passes through the point P_{00} and has tangent vectors at the point as $\mathbf{P}_{u00}$ and $\mathbf{P}_{v00}$. Show that the patch is planar if the other corner vectors are defined as linear functions of $\mathbf{P}_{u00}$ and $\mathbf{P}_{v00}$ and the corner twist vectors are zeros.

Solution Figure 5.32 shows the above defined bicubic patch. The corner position and tangent vectors can be defined by the following linear functions:

$$\begin{bmatrix} \mathbf{P}_{10} \\ \mathbf{P}_{01} \\ \mathbf{P}_{11} \\ \mathbf{P}_{u10} \\ \mathbf{P}_{u01} \\ \mathbf{P}_{u11} \\ \mathbf{P}_{v10} \\ \mathbf{P}_{v01} \\ \mathbf{P}_{v11} \end{bmatrix} = \begin{bmatrix} 1 & a_1 & b_1 \\ 1 & a_2 & b_2 \\ 1 & a_3 & b_3 \\ 0 & a_4 & b_4 \\ 0 & a_5 & b_5 \\ 0 & a_6 & b_6 \\ 0 & a_7 & b_7 \\ 0 & a_8 & b_8 \\ 0 & a_9 & b_9 \end{bmatrix} \begin{bmatrix} P_{00} \\ P_{u00} \\ P_{v00} \end{bmatrix} \tag{5.58}$$

Substituting the above equation into Eq. (5.43) we get

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{00} + a_2\mathbf{P}_{u00} + b_2\mathbf{P}_{v00} & \mathbf{P}_{v00} & a_8\mathbf{P}_{u00} + b_8\mathbf{P}_{v00} \\ \mathbf{P}_0 + a_1\mathbf{P}_{u00} + b_1\mathbf{P}_{v00} & \mathbf{P}_{00} + a_3\mathbf{P}_{u00} + b_3\mathbf{P}_{v00} & a_7\mathbf{P}_{u00} + b_7\mathbf{P}_{v00} & a_9\mathbf{P}_{v00} + b_9\mathbf{P}_{v00} \\ \hdashline \mathbf{P}_{u00} & a_5\mathbf{P}_{u00} + b_5\mathbf{P}_{v00} & 0 & 0 \\ a_4\mathbf{P}_{u00} + b_4\mathbf{P}_{v00} & a_6\mathbf{P}_{u00} + b_6\mathbf{P}_{v00} & 0 & 0 \end{array}\right] \tag{5.59}$$

If the bicubic patch whose geometry matrix $[B]$ is given by Eq. (5.59) is planar, the following equation can be written for any point on the patch:

$$\mathbf{N}_{00} \cdot [\mathbf{P}(u, v) - \mathbf{P}_{00}] = 0 \tag{5.60}$$

which states that the normal vector at point P_{00} must be perpendicular to any vector $[\mathbf{P}(u, v) - \mathbf{P}_{00}]$ in the plane, as shown in Fig. 5.32. To prove that Eq. (5.60) is valid for any point on the patch under investigation, substitute Eq. (5.59) into Eq. (5.48) and reduce the result to obtain

$$\begin{aligned}
\mathbf{P}(u, v) = \mathbf{P}_{00}[&F_1(u)F_1(v) + F_2(u)F_1(v) + F_1(u)F_2(v) + F_2(u)F_2(v)] \\
&+ \mathbf{P}_{u00}[a_1F_2(u)F_1(v) + F_3(u)F_1(v) + a_4F_4(u)F_1(v) + a_2F_1(u)f_2(v) \\
&\quad + a_3F_2(u)F_2(v) + a_5F_3(u)F_2(v) + a_6F_4(u)F_2(v) \\
&\quad + a_7F_2(u)F_3(v) + a_8F_1(u)F_4(v) + a_9F_2(u)F_4(v) \\
&+ \mathbf{P}_{v00}[b_1F_2(u)F_1(v) + b_4F_4(u)F_1(v) + b_2F_1(u)F_2(v) + b_3F_2(u)F_2(v) \\
&\quad + b_5F_3(u)F_2(v) + b_6F_4(u)F_2(v) + F_1(u)F_3(v) + b_7F_2(u)F_3(v) \\
&\quad + b_8F_1(u)F_4(v) + b_9F_2(u)F_4(v)]
\end{aligned}$$

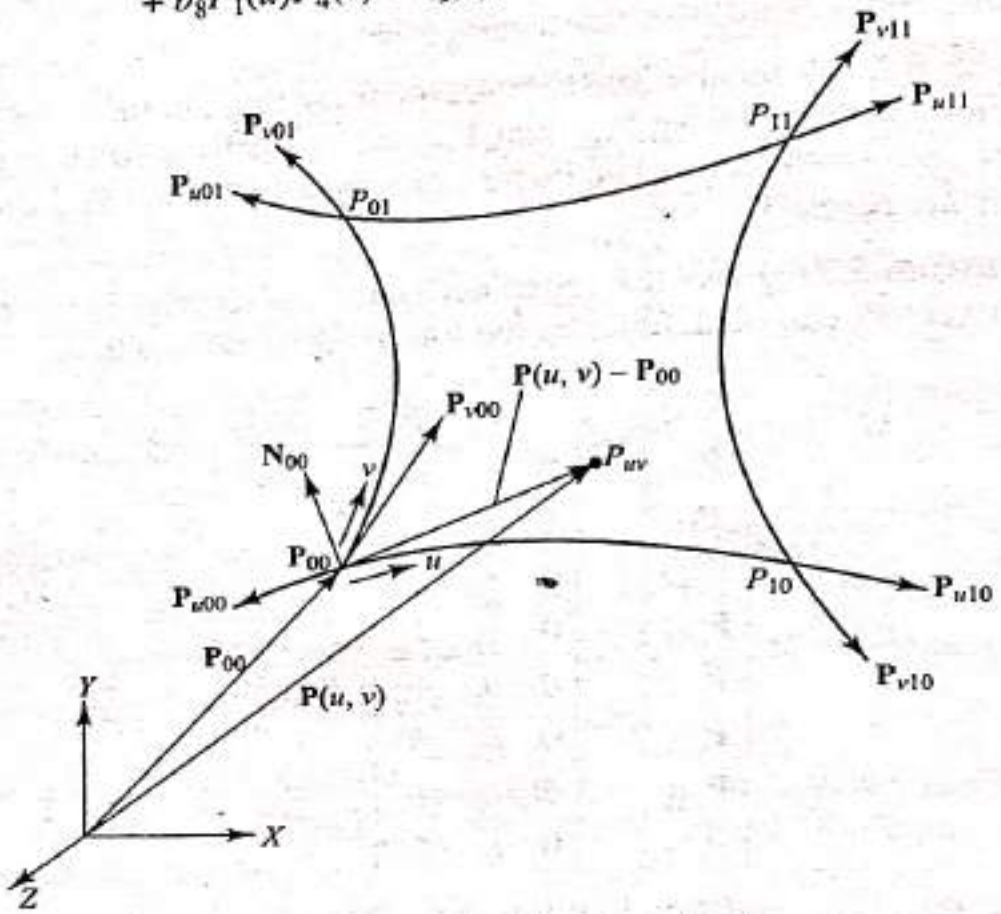


Fig. 5.32 *A Bicubic Plane Patch*

The coefficient of $\mathbf{P}_{00}$ in the above equation is equal to unity if Eq. (5.52) is used. The coefficients of $\mathbf{P}_{u00}$ and $\mathbf{P}_{v00}$ are functions of u and v only. Assuming these functions aref(u, v) and $g(u, v)$ respectively, the above equation can be written as

$$\mathbf{P}(u, v) - \mathbf{P}_{00} = f(u, v)\mathbf{P}_{u00} + g(u, v)\mathbf{P}_{v00} \tag{5.61}$$

The normal vector $\mathbf{N}_{00}$ can be written as

$$\mathbf{N}_{00} = \mathbf{P}_{u00} \times \mathbf{P}_{v00} \tag{5.62}$$

Substitute Eqs. (5.61) and (5.62) into the left-hand side of Eq. (5.60). This gives

$$(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot [f(u, v)\mathbf{P}_{u00} + g(u, v)\mathbf{P}_{v00}]$$
$$= f(u, v)(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot \mathbf{P}_{u00} + g(u, v)(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot \mathbf{P}_{v00} \quad (5.63)$$

The right-hand side of this equation is equal to zero regardless of the values of u and v. Therefore, Eq. (5.60) is valid for any point and the bicubic patch is planar.

The above technique can be generalized to test for planarity/nonplanarity of a bicubic patch based on its geometry matrix $[B]$. Let us define the matrix $[S]$ as

$$[S] = [M_H][B][M_H]^T \quad (5.64)$$

If a bicubic patch is to be planar, the elements s_{ij} of $[S]$ must satisfy the following equation:

$$N_{x00} \sum_{i=1}^{4}\sum_{j=1}^{4} \frac{s_{ijx}}{(5-i)(5-j)} + N_{y00} \sum_{i=0}^{4}\sum_{j=1}^{4} \frac{s_{ijy}}{(5-1)(5-j)}$$
$$+ N_{z00} \sum_{i=1}^{4}\sum_{j=1}^{4} \frac{s_{ijz}}{(5-i)(5-j)} - K = 0 \quad (5.65)$$

where N_{x00}, N_{y00} and N_{z00} are the components of the normal vector N_{00} and K is defined by

$$K = \mathbf{N}_{00} \cdot \mathbf{P}_{00} \quad (5.66)$$

The practical implication of this example is the ability to construct planes with curved boundaries as opposed to straight boundaries, discussed in Sec. 5.5.1.

5.6.2 Bezier Surface

A tensor product Bezier surface is an extension of the Bezier curve in two parametric directions u and v. An orderly set of data or control points is used to build a

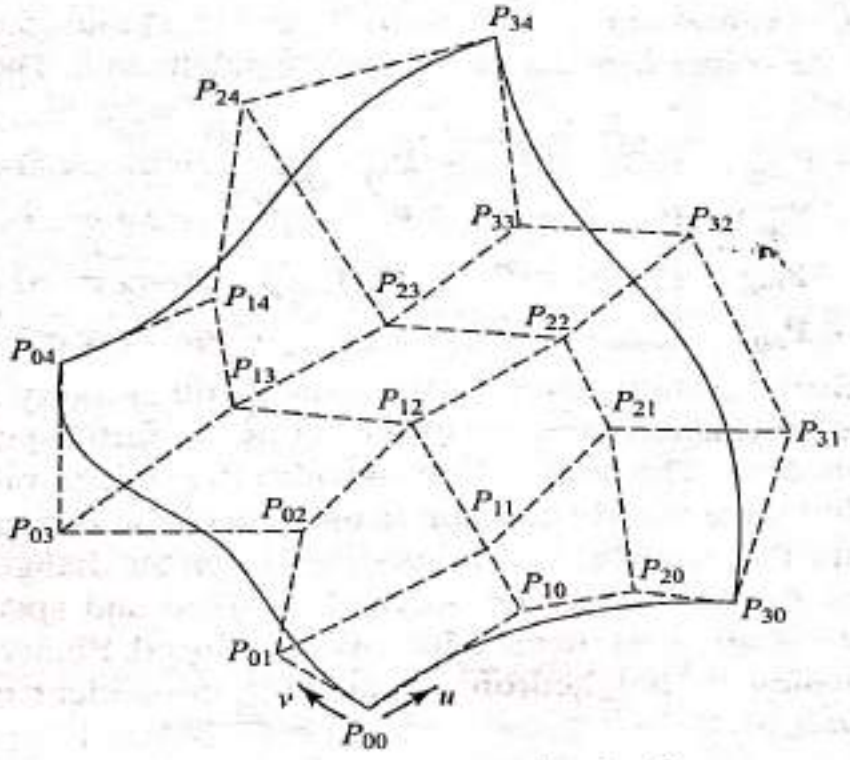


Fig. 5.33 *A 4 × 5 Bezier Surface*

topologically rectangular surface as shown in Fig. 5.33. The surface equation can be written by extending Eq. (4.91); that is,

$$\mathbf{P}(u, v) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{P}_{ij} B_{i,n}(u) B_{j,m}(v) \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{5.67}$$

where $\mathbf{P}(u, v)$ is any point on the surface and $\mathbf{P}_{ij}$ are the control points. These points form the vertices of the control or characteristic .polyhedron (shown dashed in Fig. 5.33) of the resulting Bezier surface. The points are arranged in an $(n + 1) \times (m + 1)$ rectangular array, as seen from the above equation. In comparison with Eq. (5.94) of the Bezier curve, expanding Eq. (5.67) gives

$$\begin{aligned}
\mathbf{P}(u, v) = {} & \mathbf{P}_{00}(1-u)^n(1-v)^m + \mathbf{P}_{11}C(n,1)C(m,1)uv(1-u)^{n-1}(1-v)^{m-1} \\
& + \mathbf{P}_{22}C(n,2)C(m,2)u^2v^2(1-u)^{n-2}(1-v)^{m-2} + \ldots \\
& + \mathbf{P}_{(n-1)(m-1)}C(n,n-1)C(m,m-1)u^{n-1}v^{m-1}(1-u)(1-v) \\
& + \mathbf{P}_{nm}u^nv^m \qquad \text{(Symmetric terms in } u \text{ and } v\text{)} \\
& + \mathbf{P}_{10}C(n,1)u(1-u)^{n-1}(1-v)^m + \mathbf{P}_{01}C(m,1)v(1-u)^n(1-v)^{m-1} \\
& + \mathbf{P}_{20}C(n,2)u^2(1-u)^{n-2}(1-v)^m + \mathbf{P}_{02}C(m,2)v^2(1-u)^n(1-v)^{m-2} \\
& + \ldots + \mathbf{P}_{n0}u^n(1-v)^m + \mathbf{P}_{0m}v^m(1-u)^n \\
& + \ldots + \mathbf{P}_{(n-1)(m-2)}C(n,n-1)C(m,m-2)u^{n-1}v^{m-2}(1-u)(1-v)^2 \\
& + \mathbf{P}_{(n-2)(m-1)}C(n,n-2)C(m,m-1)u^{n-2}v^{m-1}(1-u)^2(1-v) \\
& + \mathbf{P}_{(n-1)m}C(n,n-1)u^{n-1}v^m(1-u) + \mathbf{P}_{n(m-1)}u^nv^{m-1}(1-v) \qquad \text{(Nonsymmetric terms in } u \text{ and } v\text{)}
\end{aligned} \tag{5.68}$$

The characteristics of the Bezier surface are the same as those of the Bezier curve. The surface interpolates the four corner control points (see Fig. 5.33) if we substitute the (u, v) values of (0, 0), (1, 0) (0, 1) and (1,1) into (5.68). The surface is also tangent to the corner segments of the control polyhedron. The tangent vectors at the corners are:

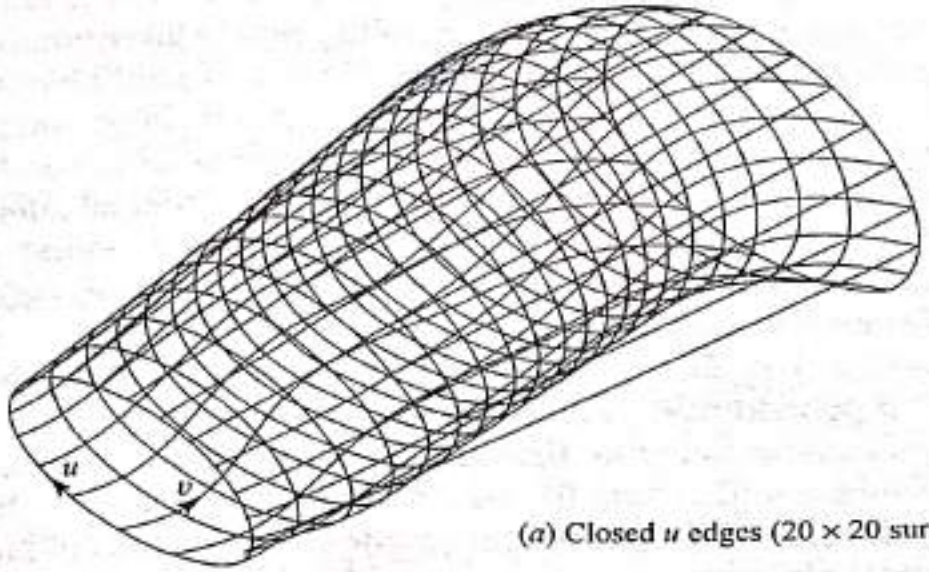
$$\begin{aligned}
&\mathbf{P}_{u00} = n(\mathbf{P}_{10} - \mathbf{P}_{00}) \quad && \mathbf{P}_{un0} = n(\mathbf{P}_{n0} - \mathbf{P}_{(n-1)0}) && \text{along } v = 0 \text{ edge} \\
&\mathbf{P}_{u0m} = n(\mathbf{P}_{1m} - \mathbf{P}_{0m}) && \mathbf{P}_{umn} = n(\mathbf{P}_{nm} - \mathbf{P}_{(n-1)m}) && \text{along } v = 1 \text{ edge} \\
&\mathbf{P}_{v00} = m(\mathbf{P}_{01} - \mathbf{P}_{00}) && \mathbf{P}_{v0m} = m(\mathbf{P}_{0m} - \mathbf{P}_{0(m-1)}) && \text{along } u = 0 \text{ edge} \\
&\mathbf{P}_{vn0} = m(\mathbf{P}_{n1} - \mathbf{P}_{n0}) && \mathbf{P}_{vnm} = m(\mathbf{P}_{nm} - \mathbf{P}_{n(m-1)}) && \text{along } u = 1 \text{ edge}
\end{aligned} \tag{5.69}$$

In addition, the Bezier surface possesses the convex hull property. The convex hull in this case is the polyhedron formed by connecting the furthest control points on the control polyhedron. The convex hull includes the control polyhedron of the surface as it includes the control polygon in the case of the Bezier curve.

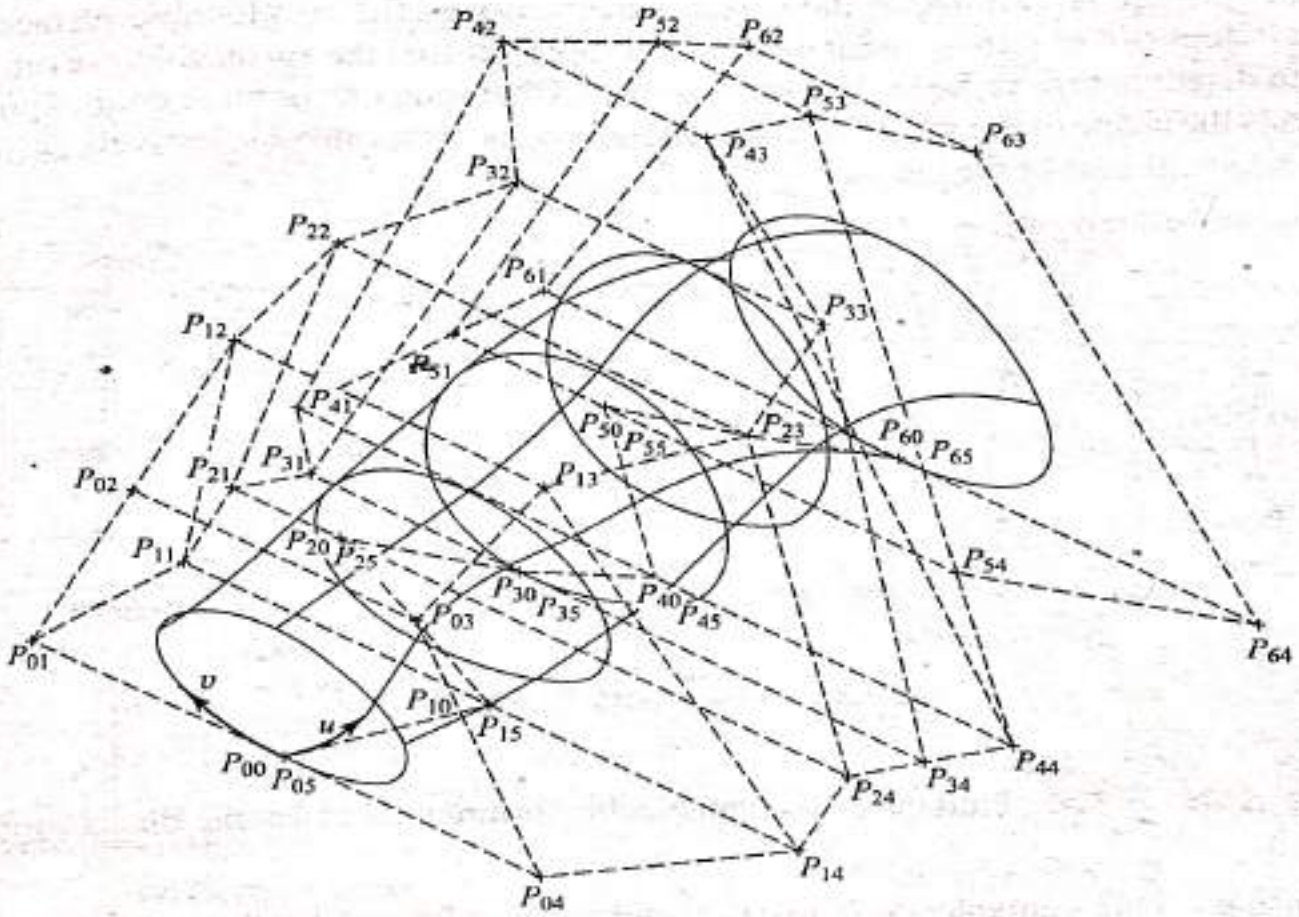
The shape of the Bezier surface can be modified by either changing some vertices of its polyhedron or by keeping the polyhedron fixed and specifying multiple coincident points of some vertices. Moreover, a closed Bezier surface can be generated by closing its polyhedron or choosing coincident corner points as illustrated in Fig. 5.34.

The normal to a Bezier surface at any point can be calculated by substituting Eq. (5.67) into (5.11) to obtain

$$\mathbf{N}(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij}\frac{\partial B_{i,n}(u)}{\partial u}B_{j,m}(v) \times \sum_{k=0}^{n}\sum_{l=0}^{m} P_{kl}B_{k,n}(u)\frac{\partial B_{l,m}(v)}{\partial v}$$



(*a*) Closed u edges (20×20 surface mesh)



(*b*) Closed v edges (polygon and 4×4 surface mesh)

Fig. 5.34 *Closed Bezier Surface*

$$= \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{n}\sum_{l=0}^{m} \frac{\partial B_{i,n}}{\partial u} B_{j,m}(v)\, B_{k,n}(u) \frac{\partial B_{l,m}(v)}{\partial v} \mathbf{P}_{ij} \times \mathbf{P}_{kl} \qquad (5.70)$$

When expanding this equation, it should be noted that $P_{ij} \times P_{kl} = 0$ if $i = k$ and $j = l$ and that $\mathbf{P}_{ij} \times \mathbf{P}_{kl} = -\mathbf{P}_{kl} \times \mathbf{P}_{ij}$.

As with the Bezier curve, the degree of Bezier surface is tied to the number of control points. Surfaces requiring great design flexibility need a large control point array and would, therefore, have a high polynomial degree. To achieve required design flexibility while keeping the surface degree manageable, large surfaces are generally designed by piecing together smaller surface patches of lower degrees. This keeps the overall degree of the surface low but requires a special attention to ensure that appropriate continuity is maintained across patch boundaries. A composite Bezier surface can have C^0 (positional) and/or C^1 (tangent) continuity. A positional continuity between, say, two patches requires that the common boundary curve between the two patches must have a common boundary polygon between the two characteristic polyhedrons (see Fig. 5.35*a*). For tangent continuity across the boundary, the segments, attached to the common boundary polygon, of one patch polyhedron must be colinear with the corresponding segments of the other patch polyhedron, as shown in Fig. 5.35*b*. This implies that the tangent planes of the patches at the common boundary curve are coincident.

In a design environment, the Bezier surface is superior to a bicubic surface in that it does not require tangent or twist vectors to define the surface. However, its main disadvantage is the lack of local control. Changing one or more control point affects the shape of the whole surface. Therefore, the user cannot selectively change the shape of part of the surface.

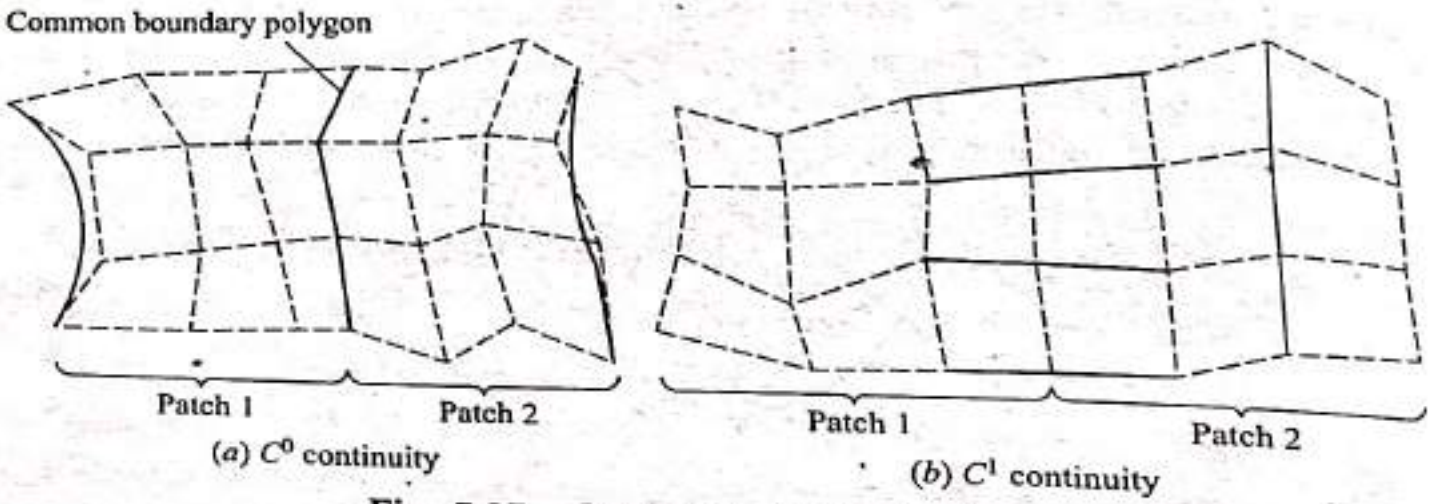


Fig. 5.35 *Composite Bezier Surface*

Example ≡ 5.7 Find the equivalent bicubic formulation of a cubic Bezier surface patch.

Solution This equivalence is usually useful for software development purposes where, say, one subroutine can handle the generation of more than one surface. It is also useful in obtaining a better understanding of the surface characteristic.

Figure 5.36 shows a cubic Bezier patch. Substituting $n = 3$ and $m = 3$ into Eq. (5.67), the patch equation is

$$\mathbf{P}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij}\, B_{i,3}(u)\, B_{j,3}(v), \qquad 0 \le u \le 1,\ 0 \le v \le 1 \tag{5.71}$$

This equation can be expanded to give

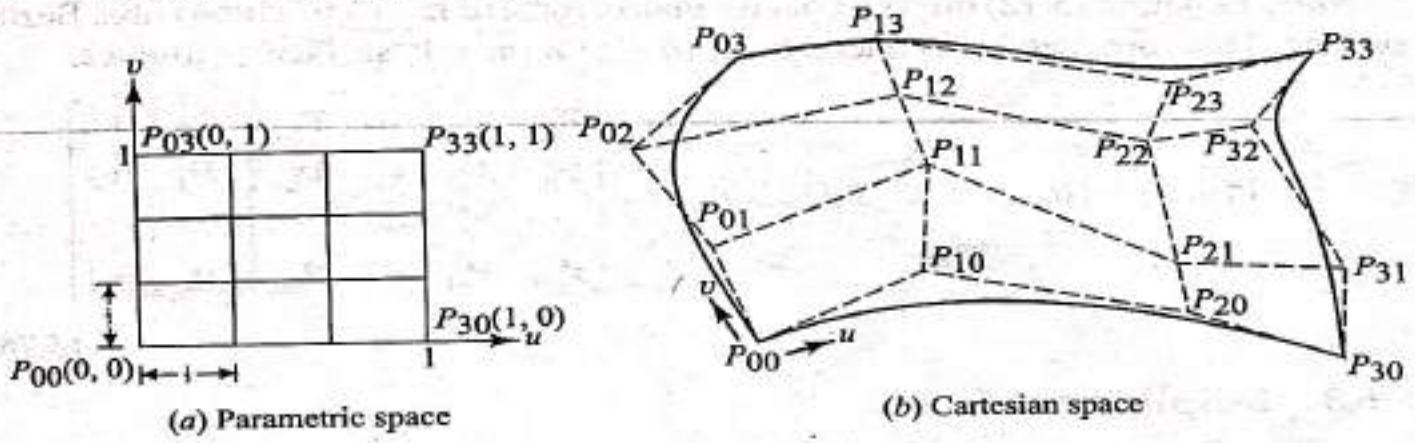
$$\begin{aligned}\mathbf{P}(u, v) = {} & \sum_{i=0}^{3} B_{i,3}(u)[\mathbf{P}_{i0}B_{0,3}(v) + \mathbf{P}_{i1}B_{1,3}(v) + \mathbf{P}_{i2}B_{2,3}(v) + \mathbf{P}_{i3}B_{3,3}(v)] \\ & + B_{0,3}(u)[\mathbf{P}_{00}B_{0,3}(v) + \mathbf{P}_{01}B_{1,3}(v) + \mathbf{P}_{02}B_{2,3}(v) + \mathbf{P}_{03}B_{3,3}(v)] \\ & + B_{1,3}(u)[\mathbf{P}_{10}B_{0,3}(v) + \mathbf{P}_{11}B_{1,3}(v) + \mathbf{P}_{12}B_{2,3}(v) + \mathbf{P}_{13}B_{3,3}(v) \\ & + B_{2,3}(u)[\mathbf{P}_{20}B_{0,3}(v) + \mathbf{P}_{21}B_{1,3}(v) + \mathbf{P}_{22}B_{2,3}(v) + \mathbf{P}_{23}B_{3,3}(v)] \\ & + B_{3,3}(u)[\mathbf{P}_{30}B_{0,3}(v) + \mathbf{P}_{31}B_{1,3}(v) + \mathbf{P}_{32}B_{2,3}(v) + \mathbf{P}_{33}B_{3,3}(v)]\end{aligned}$$

This equation can be written in a matrix form as

$$\mathbf{P}(u, v) = [B_{0,3}(u)\ \ B_{1,3}(u)\ \ B_{2,3}(u)\ \ B_{3,3}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix} \begin{bmatrix} B_{0,3}(v) \\ B_{1,3}(v) \\ B_{2,3}(v) \\ B_{3,3}(v) \end{bmatrix} \tag{5.72}$$

or $$\mathbf{P}(u, v) = [(1-u)^3\ \ 3u(1-u)^2\ \ 3u^2(1-u)\ \ u^3][P] \begin{bmatrix} (1-v)^3 \\ 3v(1-v)^2 \\ 3v^2(1-v) \\ v^3 \end{bmatrix}$$

or $$\mathbf{P}(u, v) = \mathbf{U}^T[M_B][P][M_B]^T\mathbf{V} \tag{5.73}$$



(a) Parametric space (b) Cartesian space

Fig. 5.36 *A Cubic Bezier Patch*

where the subscript B denotes Bezier and

$$[P] = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}$$

and

$$[M_B] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.74}$$

and the **U** and **V** vectors are $[u^3\ u^2\ u\ 1]^T$ and $[v^3\ v^2\ v\ 1]^T$ respectively. Notice that $[M_B]$ given by Eq. (5.74) is the same matrix for the cubic Bezier curve (see Prob. 4.9 in Chap. 4).

Equating Eq. (5.73) with (5.42) gives

$$\mathbf{U}^T[M_H][B][M_H]^T\mathbf{V} = \mathbf{U}^T[M_B][P][M_B]^T\mathbf{V}$$

or

$$[M_H][B][M_H]^T = [M_B][P][M_B]^T$$

Solving for $[B]$ gives

$$[B] = [M_H]^{-1}[M_B][P][M_B]^T[M_H]^{T-1}$$

Using Eq. (4.86) for $[M_H]^{-1}$, this equation can be reduced to give

$$[B] = \left[\begin{array}{cc|cc} \mathbf{P}_{00} & \mathbf{P}_{03} & 3(\mathbf{P}_{01}-\mathbf{P}_{00}) & 3(\mathbf{P}_{03}-\mathbf{P}_{02}) \\ \mathbf{P}_{30} & \mathbf{P}_{33} & 3(\mathbf{P}_{31}-\mathbf{P}_{30}) & 3(\mathbf{P}_{33}-\mathbf{P}_{32}) \\ \hline 3(\mathbf{P}_{10}-\mathbf{P}_{00}) & 3(\mathbf{P}_{13}-\mathbf{P}_{03}) & 9(\mathbf{P}_{00}-\mathbf{P}_{10}-\mathbf{P}_{01}+\mathbf{P}_{11}) & 9(\mathbf{P}_{02}-\mathbf{P}_{12}-\mathbf{P}_{03}+\mathbf{P}_{13}) \\ 3(\mathbf{P}_{30}-\mathbf{P}_{20}) & 3(\mathbf{P}_{33}-\mathbf{P}_{23}) & 9(\mathbf{P}_{20}-\mathbf{P}_{21}-\mathbf{P}_{30}+\mathbf{P}_{31}) & 9(\mathbf{P}_{22}-\mathbf{P}_{23}-\mathbf{P}_{32}+\mathbf{P}_{33}) \end{array}\right] \tag{5.75}$$

Comparing this equation with Eq. (5.43) for the bicubic patch reveals that the tangent and twist vectors of the Bezier surface are expressed in terms of the vertices of its characteristic polyhedron.

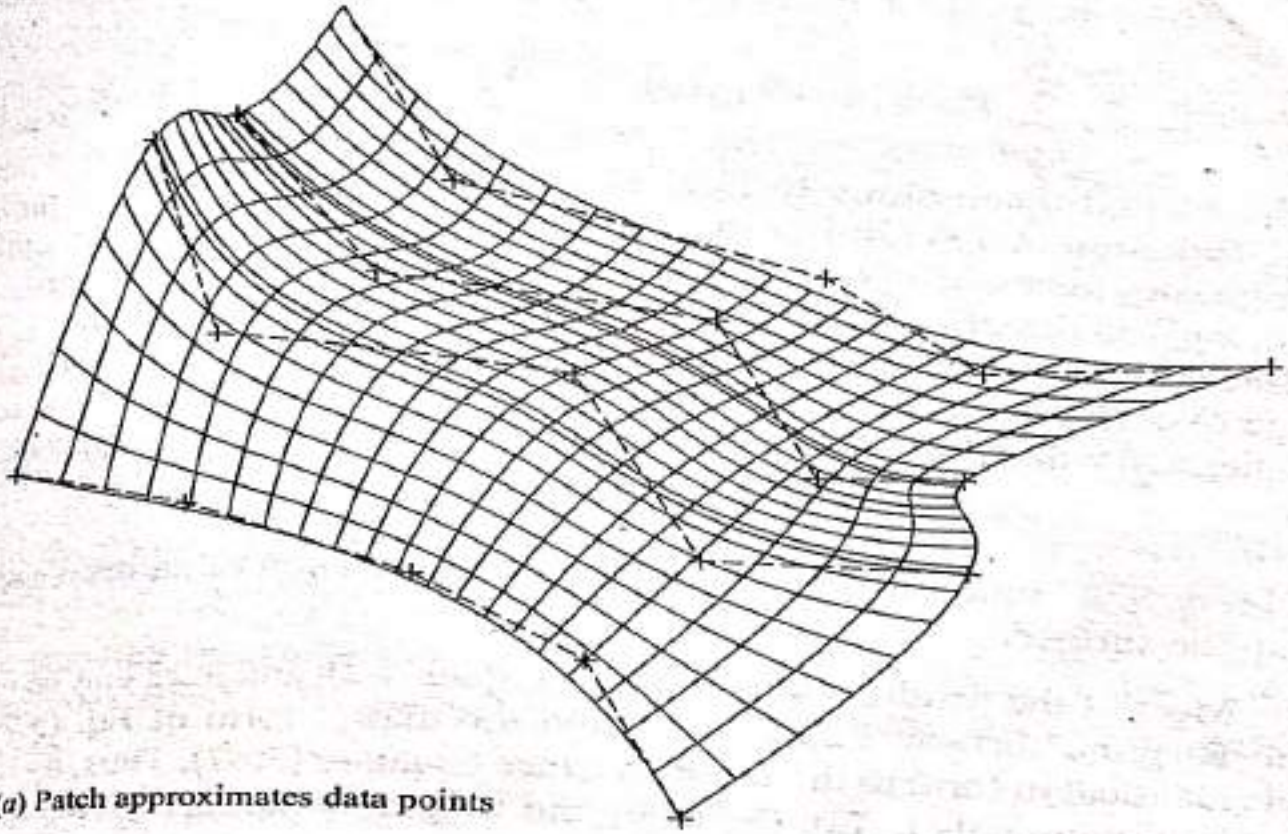
Note: Equation (5.72) offers a concise matrix form of Eq. (5.67) for a cubic Bezier surface. This form can be extended to an $(n+1) \times (m+1)$ surface as follows:

$$\mathbf{P}(u, v) = [B_{0,n}(u)\ B_{1,n}(u)\ \ldots\ B_{n,n}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \cdots & \mathbf{P}_{0m} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1m} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{n0} & \mathbf{P}_{n1} & \cdots & \mathbf{P}_{nm} \end{bmatrix} \begin{bmatrix} B_{0,m}(v) \\ B_{1,m}(v) \\ \vdots \\ B_{n,m}(v) \end{bmatrix} \tag{5.76}$$

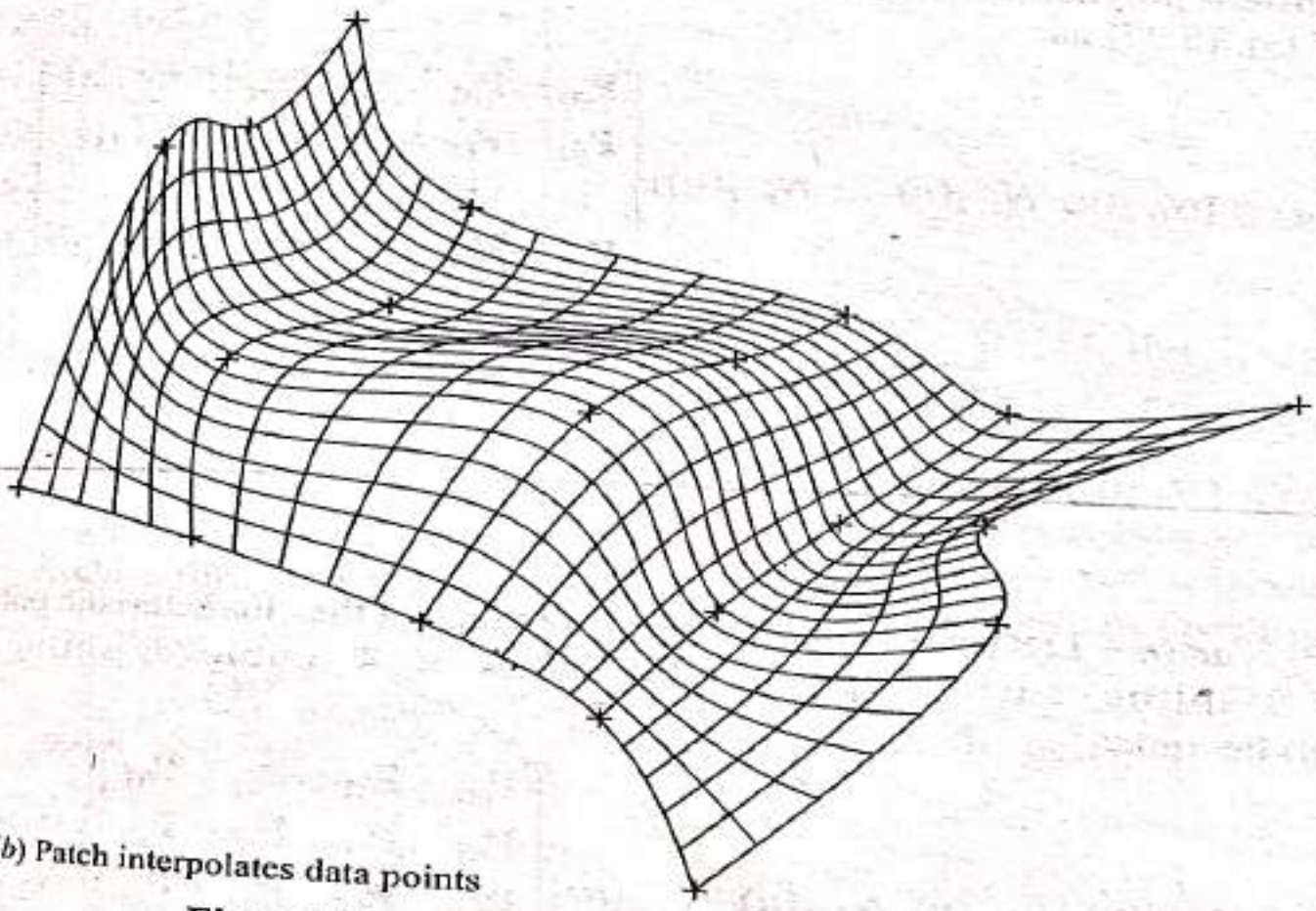
5.6.3 B-Spline Surface

The same tensor product method used with Bezier curves can extend B-splines to describe B-spline surfaces. A rectangular set of data (control) points creates the

surface. This set forms the vertices of the characteristic polyhedron that approximates and controls the shape of the resulting surface. A B-spline surface can approximate or interpolate the vertices of the polyhedron as shown in Fig. 5.37.



(*a*) Patch approximates data points



(*b*) Patch interpolates data points

Fig. 5.37 *4 × 5 B-spline Surface Patches*

The degree of the surface is independent of the number of control points and continuity is automatically maintained throughout the surface by virtue of the form of blending functions. As a result, surface intersections can easily be managed.

A B-spline surface patch defined by an $(n+1)\times(m+1)$ array of control points is given by extending Eq. (4.103) into two dimensions:

$$\mathbf{P}(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij} N_{i,k}(u) N_{j,l}(v), \qquad 0 \le u \le u_{\max}, \qquad 0 \le v \le v_{\max} \quad (5.77)$$

All the related discussions to Eqs. (4.104) and (4.105) apply to the above equation. Equation (4.77) implies that knot vectors in both u and v directions are constant but not necessarily equal. Other formulations could allow various knot vectors in a given direction to increase the flexibility of local control.

B-spline surfaces have the same characteristics as B-spline curves. Their major advantage over Bezier surfaces is the local control. Composite B-spline surfaces can be generated with C^0 and/or C^1 continuity in the same way as composite Bezier surfaces.

Example ≡ 5.8 Find the equivalent bicubic formulation of an open and closed cubic B-spline surface.

Solution Most of the results obtained in Examples 4.21 and 4.22 can be extended to a cubic B-spline surface. First, let us find the matrix form of Eq. (5.77). This equation is identical in form to the Bezier surface equation (5.67). Thus, by replacing the Bernstein polynomials in Eq. (5.76) by the B-spline functions yields the matrix form of Eq. (5.77) as

$$\mathbf{P}(u, v) = [N_{0,k}(u)\ N_{1,k}(u)\ \cdots\ N_{n,k}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \cdots & \mathbf{P}_{0m} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1m} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{n0} & \mathbf{P}_{n1} & \cdots & \mathbf{P}_{nm} \end{bmatrix} \begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix} \quad (5.78)$$

or

$$\mathbf{P}(u, v) = [N_{0,k}(u)\ N_{1,k}(u)\ \cdots\ N_{n,k}(u)][\mathbf{P}]] \begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix} \quad (5.79)$$

where $[P]$ is an $(n+1)\times(m+1)$ matrix of the vertices of the characteristic polyhedron of the B-spline surface patch. For a 4×4 cubic B-spline patch, Eq. (5.78) becomes

$$\mathbf{P}(u, v) = [N_{0,4}(u)\ N_{1,4}(u)\ N_{2,4}(u)\ N_{3,4}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix} \begin{bmatrix} N_{0,4}(v) \\ N_{1,4}(v) \\ N_{2,4}(v) \\ N_{3,4}(v) \end{bmatrix}$$

For the open patch, the B-spline functions are the same as the Bernstein polynomials of the previous example (refer to Example 4.21) and the equivalent bicubic formulation results in Eq. (5.75). For, say, a 5×6 open cubic B-spline patch, we get

$$\mathbf{P}(u, v) = [N_{0,4}(u) \;\; N_{1,4}(u) \;\; N_{2,4}(u) \;\; N_{3,4}(u) \;\; N_{4,4}(u)][P] \begin{bmatrix} N_{0,4}(v) \\ N_{1,4}(v) \\ N_{2,4}(v) \\ N_{3,4}(v) \\ N_{4,4}(v) \\ N_{5,4}(v) \end{bmatrix} \quad \begin{matrix} 0 \le u \le 2, \\ 0 \le v \le 3 \end{matrix}$$

where

$$[P] = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} & \mathbf{P}_{04} & \mathbf{P}_{05} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \mathbf{P}_{14} & \mathbf{P}_{15} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} & \mathbf{P}_{24} & \mathbf{P}_{25} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} & \mathbf{P}_{34} & \mathbf{P}_{35} \\ \mathbf{P}_{40} & \mathbf{P}_{41} & \mathbf{P}_{42} & \mathbf{P}_{43} & \mathbf{P}_{44} & \mathbf{P}_{45} \end{bmatrix}$$

All the B-spline functions shown in this equation are calculated by following the procedure described in Example 4.21. After this is done, the above equation can be reduced to

$$\left[\begin{array}{c|c|c} \mathbf{P}_{11}(u, v) & \mathbf{P}_{12}(u, v) & \mathbf{P}_{13}(u, v) \\ 0 \le u \le 1, & 0 \le u \le 1, & 0 \le u \le 1, \\ 0 \le v \le 1, & 1 \le v \le 2, & 2 \le v \le 3, \\ \hline \mathbf{P}_{21}(u, v) & \mathbf{P}_{22}(u, v) & \mathbf{P}_{23}(u, v) \\ 1 \le u \le 2, & 1 \le u \le 2, & 1 \le u \le 2, \\ 0 \le v \le 1, & 1 \le v \le 2, & 2 \le v \le 3, \end{array}\right]$$

$$= \left[\begin{array}{c|c|c} \mathbf{U}^T[M_S][P_{11}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{12}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{13}][M_S]^T\mathbf{V} \\ \hline \mathbf{U}^T[M_S][P_{21}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{22}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{23}][M_S]^T\mathbf{V} \end{array}\right] \tag{5.80}$$

where $[P_{11}]$, $[P_{12}]$,..., $[P_{23}]$ are partitions of $[P]$. Each partition is 4×4 and is different from its neighbor (moving in the row direction of $[P]$) by one row or by one column (moving in the column direction of $[P]$). The general form of any partition is given by

$$[P_{ij}] = \begin{bmatrix} \mathbf{P}_{(i-1)(j-1)} & \mathbf{P}_{(i-1)j} & \mathbf{P}_{(i-1)(j+1)} & \mathbf{P}_{(i-1)(j+2)} \\ \mathbf{P}_{i(j-1)} & \mathbf{P}_{ij} & \mathbf{P}_{i(j+1)} & \mathbf{P}_{i(j+2)} \\ \mathbf{P}_{(i+1)(j-1)} & \mathbf{P}_{(i+1)j} & \mathbf{P}_{(i+1)(j+1)} & \mathbf{P}_{(i+1)(j+2)} \\ \mathbf{P}_{(i+2)(j-1)} & \mathbf{P}_{(i+2)j} & \mathbf{P}_{(i+2)(j+1)} & \mathbf{P}_{(i+2)(j+2)} \end{bmatrix}$$

The matrix $[M_S]$ is the same as for cubic B-spline curves. It is given by

$$[M_S] = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{5.81}$$

Equation (5.80) can be written in a more concise form as

$$\mathbf{P}_{ij}(u, v) = \mathbf{U}^T[M_S][\mathbf{P}_{ij}][M_S]^T\mathbf{V}$$
$$1 \le i \le 2,\ 1 \le j \le 3;\ (i-1) \le u \le i,\ (j-1) \le v \le j \tag{5.82}$$

Equating the above equation with the bicubic equation, the equivalent $[B_{ij}]$ matrix becomes

$$[B_{ij}] = [M_H]^{-1}[M_S][P_{ij}][M_S]^T[M_H]^{T-1} \tag{5.83}$$

Notice that the above procedure can be extended to an $n \times m$ cubic B-spline surface.

A similar procedure can be followed for a closed cubic B-spline patch. The difference comes in the form of the B-spline functions. For a 4×4 cubic B-spline patch closed in the u direction, the v direction, or both directions, the following three equations can be written respectively:

$$\mathbf{P}(u, v) = [N_{0,4}((u+4) \bmod 4),\ N_{0,4}((u+3) \bmod 4),\ N_{0,4}((u+2) \bmod 4),\ N_{0,4}((u+1) \bmod 4)]$$
$$\times [P]\begin{bmatrix} N_{0,4}(v) \\ N_{1,4}(v) \\ N_{2,4}(v) \\ N_{3,4}(v) \end{bmatrix}$$

$$\mathbf{P}(u, v) = [N_{0,4}(u)\ \ N_{1,4}(u)\ \ N_{2,4}(u)\ \ N_{3,4}(u)][P]\begin{bmatrix} N_{0,4}((v+4) \bmod 4) \\ N_{0,4}((v+3) \bmod 4) \\ N_{0,4}((v+2) \bmod 4) \\ N_{0,4}((v+1) \bmod 4) \end{bmatrix}$$

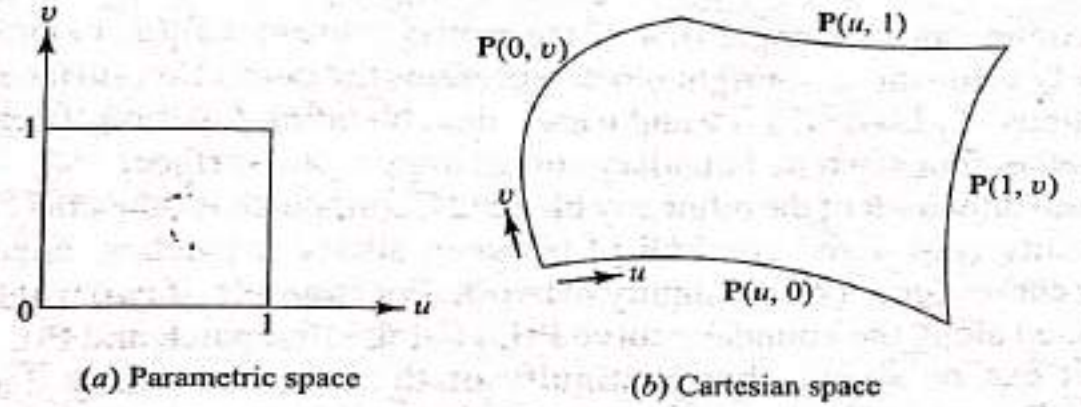
and $$\mathbf{P}(u, v) = [N_{0,4}((u+4) \bmod 4),\ N_{0,4}((u+3) \bmod 4,\ N_{0,4}((u+2) \bmod 4),\ N_{0,4}((u+1) \bmod 4)]$$
$$\times [P]\begin{bmatrix} N_{0,4}((v+4) \bmod 4) \\ N_{0,4}((v+3) \bmod 4) \\ N_{0,4}((v+2) \bmod 4) \\ N_{0,4}((v+1) \bmod 4) \end{bmatrix}$$

The closed B-spline functions have been evaluated in Example 4.22. Investigating Eqs. (4.118) reveals that there are four different expressions for the matrix $[B]$ [see Eq. (5.83) above] depending on the value of u, v, or u and v. For, say, a 5×6 closed

cubic B-spline patch, the above procedure is repeated but with the functions $[N_{0,4}((u+5) \bmod 5), N_{0,4}((u+4) \bmod 5), N_{0,4}((u+3) \bmod 5), N_{0,4}((u+2) \bmod 5), N_{0,4}((u+1) \bmod 5)]$ and $[N_{0,4}((u+6) \bmod 6), N_{0,4}((u+5) \bmod 6), N_{0,4}((u+4) \bmod 6), N_{0,4}((u+3) \bmod 6), N_{0,4}((u+2) \bmod 6), N_{0,4}((u+1) \bmod 6)]$. The control point matrix $[P]$ is a 5×6 matrix as in the case of the open patch.

5.6.4 Coons Surface

All the surface methods introduced thus far share one common philosophy; that is, they all require a finite number of data points to generate the respective surfaces. In contrast, a Coons surface patch is a form of "transfinite interpolation" which indicates that the Coons scheme interpolates to an infinite number of data points, that is, to all points of a curve segment, to generate the surface. The Coons patch is particularly useful in blending four prescribed intersecting curves which form a closed boundary as shown in Fig. 5.38. The figure shows the given four boundary curves as $\mathbf{P}(u, 0)$, $\mathbf{P}(1, v)$, $\mathbf{P}(u, 1)$ and $\mathbf{P}(0, u)$. It is assumed that u and v range from 0 to 1 along these boundaries and that each pair of opposite boundary curves are identically parametrized. Development of the Coons surface patch centers around answering the following question: what is a suitable well-behaved function $\mathbf{P}(u, v)$ which blends the four given boundary curves and which satisfies the boundary conditions, that is, reduces to the correct boundary curve when $u = 0$, $u = 1$, $v = 0$ and $v = 1$?



(a) Parametric space (b) Cartesian space

Fig. 5.38 *Boundaries of Coons Surface Patch*

Let us first consider the case of a bilinearly blended Coons patch which interpolates to the four boundary curves shown in Fig. 5.38. For this case, it is useful to recall that a ruled surface interpolates linearly between two given boundary curves in one direction as shown by Eq. (5.36). Therefore, the superposition of two ruled surfaces connecting the two pairs of boundary curves might satisfy the boundary curve conditions and produces the Coons patch. Let us investigate this claim. Utilizing Eq. (5.36) in the v and u directions gives respectively

$$\mathbf{P}_1(u, v) = (1 - u)\mathbf{P}(0, v) + u\mathbf{P}(1, v) \tag{5.84}$$

and

$$\mathbf{P}_2(u, v) = (1 - v)\mathbf{P}(u, 0) + v\mathbf{P}(u, 1) \tag{5.85}$$

Adding these two equations gives the surface

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) + \mathbf{P}_2(u, v) \tag{5.86}$$

The resulting surface patch described by Eq. (5.86) does not satisfy the boundary conditions. For example, substituting $v = 0$ and 1 into this equation gives respectively

$$\mathbf{P}(u, 0) = \mathbf{P}(u, 0) + [(1 - u)\mathbf{P}(0, 0) + u\mathbf{P}(1, 0)] \tag{5.87}$$

$$\mathbf{P}(u, 1) = \mathbf{P}(u, 1) + [(1 - u)\mathbf{P}(0, 1) + u\mathbf{P}(1, 1)] \tag{5.88}$$

These two equations show that the terms in square brackets are extra and should be eliminated to recover the original boundary curves. These terms define the boundaries of an unwanted surface $P_3(u, v)$ which is embedded in Eq. (5.86). This surface can be defined by linear interpolation in the v direction, that is,

$$\mathbf{P}_3(u, v) = (1 - v)[(1 - u)\mathbf{P}(0, 0) + u\mathbf{P}(1, 0)] + v[(1 - u)\mathbf{P}(0, 1) + u\mathbf{P}(1, 1)] \tag{5.89}$$

Subtracting $\mathbf{P}_3(u, v)$ (called the "correction surface") from Eq. (5.86) gives

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) + \mathbf{P}_2(u, v) - \mathbf{P}_3(u, v) \tag{5.90}$$

or

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) \oplus \mathbf{P}_2(u, v) \tag{5.91}$$

where $\oplus$ defines the "boolean sum" which is $\mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_3$. The surface $\mathbf{P}(u, v)$ given by the above equation defines the bilinear Coons patch connecting the four boundary curves shown in Fig. 5.38. Figure 5.39 shows the graphical representation of Eq. (5.91) and its matrix form is

$$\mathbf{P}(u, v) = -[-1 \quad (1 - u) \quad u]\left[\begin{array}{c|cc} 0 & \mathbf{P}(u, 0) & \mathbf{P}(u, 1) \\ \hline \mathbf{P}(0, v) & \mathbf{P}(0, 0) & \mathbf{P}(0, 1) \\ \mathbf{P}(1, v) & \mathbf{P}(1, 0) & \mathbf{P}(1, 1) \end{array}\right]\left[\begin{array}{c} -1 \\ 1 - v \\ v \end{array}\right] \tag{5.92}$$

The left column and the upper row of the matrix represent $\mathbf{P}_1(u, v)$ and $\mathbf{P}_2(u, v)$ respectively while the lower right block represents the correction surface $\mathbf{P}_3(u, v)$. The functions -1, $1 - u$, u, $1 - v$ and v are called blending functions because they blend together four separate boundary curves to give one surface.

The main drawback of the bilinearly blended Coons patch is that it only provides C^0 continuity (positional continuity) between adjacent patches, even if their boundary curves form a C^1 continuity network. For example, if two patches are to be connected along the boundary curve $\mathbf{P}(1, v)$ of the first patch and $\mathbf{P}(0, v)$ of the second, it can be shown that continuity of the cross-boundary derivatives (Fig. 5.40) $\mathbf{P}_u(1, v)$ and $\mathbf{P}_u(0, v)$ of the two patches cannot be made equal (see Prob. 5.7 at the end of the chapter).

Gradient continuity across boundaries of patches of a composite Coons surface is essential for practical applications. If, for example, a network of C^1 curves is given as shown in Fig. 5.41, it becomes very desirable to form a composite Coons surface which is smooth or C^1 continuous, that is, it provides continuity of cross-boundary derivatives between patches. Investigation of Eq. (5.92) shows that the choice of blending functions controls the behavior of the resulting Coons patch. If the cubic Hermite polynomials $F_1(x)$ and $F_2(x)$ given in Eq. (5.52) are used instead of the linear polynomials $(1 - u)$ and u respectively, a bicubically blended Coons patch results. This patch guarantees C^1 continuity between patches. Substituting $F_1(x)$ and $F_2(x)$ into Eqs. (5.84) and (5.85) gives

$$\mathbf{P}_1(u, v) = (2u^3 - 3u^2 + 1)\mathbf{P}(0, v) + (-2u^3 + 3u^2)\mathbf{P}(1, v) \tag{5.93}$$

$$\mathbf{P}_2(u, v) = (2v^3 - 3v^2 + 1)\mathbf{P}(u, 0) + (-2v^3 + 3v^2)\mathbf{P}(u, 1) \tag{5.94}$$

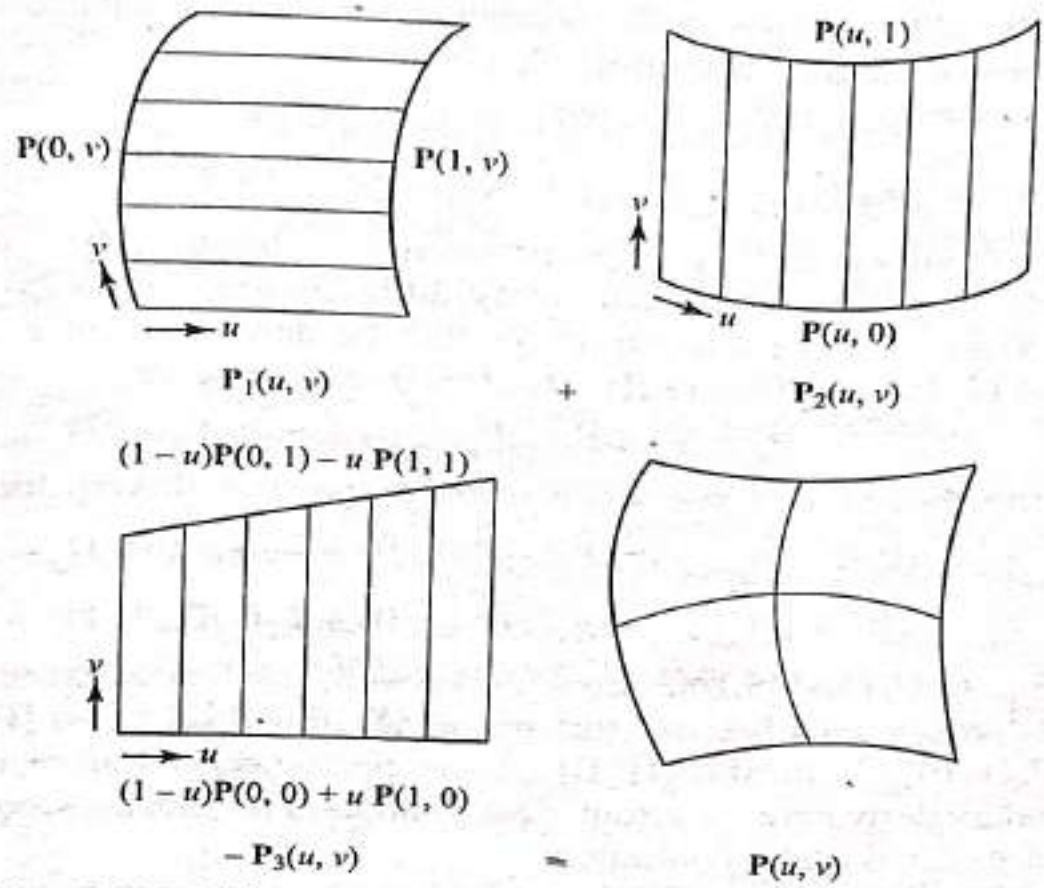


Fig. 5.39 *Bilinearly Blended Coons Patch (boolean sum)*

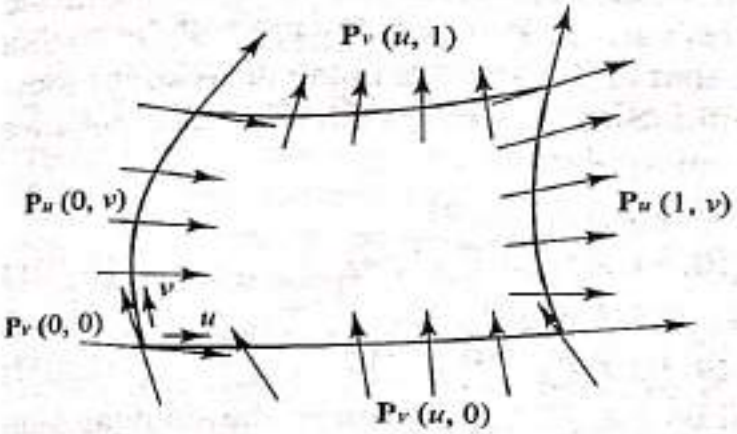


Fig. 5.40 *Cross-boundary Derivatives*

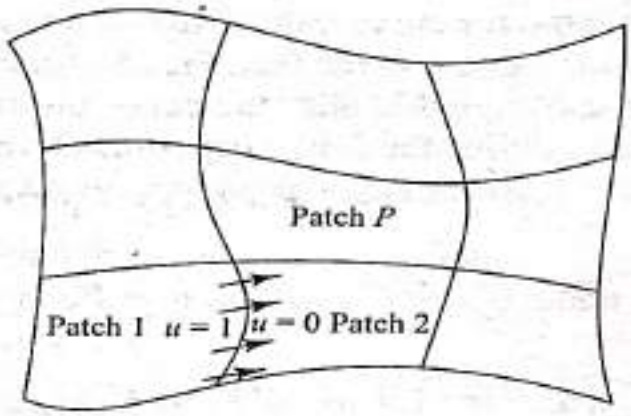


Fig. 5.41 *A Composed Coons Surface Formed by a Network of C^1 Boundary Curves*

Similar to the bilinear patch, the boolean sum $P_1 \oplus P_2$ can be formed and the matrix equation of the bicubic Coons patch becomes

$$\mathbf{P}(u, v) = -[-1 \quad F_1(u) \quad F_2(u)] \begin{bmatrix} 0 & \mathbf{P}(u, 0) & \mathbf{P}(u, 1) \\ \mathbf{P}(0, v) & \mathbf{P}(0, 0) & \mathbf{P}(0, 1) \\ \mathbf{P}(1, v) & \mathbf{P}(1, 0) & \mathbf{P}(1, 1) \end{bmatrix} \begin{bmatrix} -1 \\ F_1(v) \\ F_2(v) \end{bmatrix} \tag{5.95}$$

As can easily be seen from Eqs. (5.92) and (5.95), the correction surface is usually formed by applying the blending functions to the corner data alone.

To check continuity across patch boundaries, let us consider patch 1 and patch 2 in Fig. 5.41. For C^0 and C^1 continuity we should have

$$[\mathbf{P}(0, v)]_{\text{patch 2}} = [\mathbf{P}(1, v))]_{\text{patch 1}} \tag{5.96}$$

$$[\mathbf{P}_u(0, v)]_{\text{patch 2}} = [\mathbf{P}_u(1, v)]_{\text{patch 1}} \tag{5.97}$$

C^0 continuity is automatically satisfied between the two patches because they share the same boundary curve. For C^1 continuity, differentiating Eq. (5.95) with respect to u, using $G_1(x)$ and $G_2(x)$ in Eq. (5.53) for the derivatives of F_1 and F_2 and noticing that $G_1(0) = G_2(0) = G_1(1) = G_2(1) = 0$, we can write

$$\mathbf{P}_u(u, v) = \mathbf{F}_1(v)\mathbf{P}_u(u, 0) + \mathbf{F}_2(v)\mathbf{P}_u(u, 1) \tag{5.98}$$

At the common boundary curve between the two patches, this equation becomes

$$[\mathbf{P}_u(0, v)]_{\text{patch 2}} = F_1(v)\mathbf{P}_u(0, 0) + F_2(v)\mathbf{P}_u(0, 1) \tag{5.99}$$

and

$$[\mathbf{P}_u(1, v)]_{\text{patch 1}} = F_1(v)\mathbf{P}_u(1, 0) + F_2(v)\mathbf{P}_u(1, 1) \tag{5.100}$$

Based on Eqs. (5.99) and (5.100), Eq. (5.97) is satisfied if the network of boundary curves is C^1 continuous because this makes $[\mathbf{P}_u(0, 0)]_{\text{patch 2}}$ and $[\mathbf{P}_u(0, 1)]_{\text{patch 2}}$ equal to $[\mathbf{P}_u(1, 0)]_{\text{patch 1}}$ and $[\mathbf{P}_u(1, 1)]_{\text{patch 1}}$ respectively. Therefore, continuity of cross-boundary derivatives is automatically satisfied for bicubic Coons patches if the boundary curves are C^1 continuous.

The bicubic Coons patch as defined by Eq. (5.95) is easy to use in a design environment because only the four boundary curves are needed. However, a more flexible composite C^1 bicubic Coons surface can be developed if, together with the boundary curves, the cross-boundary derivatives $\mathbf{P}_u(0, v)$, $\mathbf{P}_u(1, v)$, $\mathbf{P}_v(u, 0)$ and $\mathbf{P}_v(u, 1)$ are given (see Fig. 5.40). Note that at the corners these derivatives must be compatible with the curve information. Similar to Eqs. (5.93) and (5.94), we can define the following cubic Hermite interpolants:

$$\begin{aligned}\mathbf{P}_1(u, v) = {} & F_1(u)\mathbf{P}(0, v) + F_2(u)\mathbf{P}(1, v) \\ & + F_3(u)\mathbf{P}_u(0, v) + F_4(u)\mathbf{P}_u(1, v)\end{aligned} \tag{5.101}$$

and

$$\begin{aligned}\mathbf{P}_2(u, v) = {} & F_1(v)\,\mathbf{P}\,(u, 0) + F_2(v)\mathbf{P}(u, 1) \\ & + F_3(v)\mathbf{P}_v(u, 0) + F_4(v)\mathbf{P}_v(u, 1)\end{aligned} \tag{5.102}$$

where the functions F_1 to F_4 are given by Eq. (5.52). Forming the boolean sum $(\mathbf{P}_1 \oplus \mathbf{P}_2)$ of the above two equations results in the bicubic Coons patch that incorporates cross-boundary derivatives. The introduction of the cross-boundary derivatives causes the twist vectors at the corners of the patch to appear. The matrix equation of this Coons patch can be written as

$$\mathbf{P}(u, v) = -\,[-1 \quad F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)]$$

$$\times \begin{bmatrix} 0 & \mathbf{P}(u,0) & \mathbf{P}(u,1) & \mathbf{P}_v(u,0) & \mathbf{P}_v(u,1) \\ \mathbf{P}(0,v) & \mathbf{P}(0,0) & \mathbf{P}(0,1) & \mathbf{P}_v(0,0) & \mathbf{P}_v(0,1) \\ \mathbf{P}(1,v) & \mathbf{P}(1,0) & \mathbf{P}(1,1) & \mathbf{P}_v(1,0) & \mathbf{P}_v(1,1) \\ \mathbf{P}_u(0,v) & \mathbf{P}_u(0,0) & \mathbf{P}_u(0,1) & \mathbf{P}_{uv}(0,0) & \mathbf{P}_{uv}(0,1) \\ \mathbf{P}_u(1,v) & \mathbf{P}_u(1,0) & \mathbf{P}_u(1,1) & \mathbf{P}_{uv}(1,0) & \mathbf{P}_{uv}(1,1) \end{bmatrix} \begin{bmatrix} -1 \\ F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{5.103}$$

The upper 3×3 matrix determines the patch defined previously by Eq. (5.95). The left column and the upper row represent $P_1(u, v)$ and $P_2(u, v)$ respectively, while the lower right 4×4 matrix represents the bicubic tensor product surface discussed in Sec. 5.6.1 [see Eq. (5.43)]. Equation (5.103) shows that every bicubically blended Coons surface reproduces a bicubic surface. On the other hand, bicubically blended Coons patches cannot be described in general by bicubic tensor product surfaces. Thus, Coon formulation can describe a much richer variety of surfaces than do tensor product surfaces.

Example ≣ 5.9 Show that if the boundary curves of a bilinear Coons patch are co-planar, the resulting patch is also planar.

Solution We need to show that the surface normal at any point on the surface is normal to the plane of the boundary curves. Based on Eq. (5.90), the surface tangent vectors are

$$\begin{aligned}\mathbf{P}_u(u, v) &= [\mathbf{P}_u(u, 0) + \mathbf{P}(1, v) - \mathbf{P}(0, v) + \mathbf{P}(0, 0) - \mathbf{P}(1, 0)] \\ &\quad + v[\mathbf{P}_u(u, 1) - \mathbf{P}_u(u, 0) + \mathbf{P}(1, 0) - \mathbf{P}(0, 0) - \mathbf{P}(1, 1) + \mathbf{P}(0, 1)] \\ &= \mathbf{A} + v\mathbf{B}\end{aligned}$$

and

$$\begin{aligned}\mathbf{P}_v(u, v) &= [\mathbf{P}_v(0, v) + \mathbf{P}(u, 1) - \mathbf{P}(u, 0) + \mathbf{P}(0, 0) - \mathbf{P}(0, 1)] \\ &\quad + u[\mathbf{P}_v(1, v) - \mathbf{P}_v(0, v) - \mathbf{P}(0, 0) + \mathbf{P}(0, 1) + \mathbf{P}(1, 0) - \mathbf{P}(1, 1)] \\ &= \mathbf{C} + u\mathbf{D}\end{aligned}$$

It can easily be shown that $\mathbf{P}_u(u, v)$ and $\mathbf{P}_v(u, v)$ lie in the plane of the boundary curves. Considering $\mathbf{P}_u(u, v)$, the vectors $\mathbf{P}_u(u, 0)$, $\mathbf{P}(1, v) - \mathbf{P}(0, v)$ and $\mathbf{P}(0, 0) - \mathbf{P}(1, 0)$ lie in the given plane. By investigating the coefficient of v, the vectors $\mathbf{P}_u(u, 1)$, $\mathbf{P}_u(u, 0)$, $\mathbf{P}(1, 0) - \mathbf{P}(0, 0)$ and $\mathbf{P}(0, 1) - \mathbf{P}(1, 1)$ also lie in the given plane. Therefore, vectors A and B lie in the plane. Consequently, the tangent vector $\mathbf{P}_u(u, v)$ to the surface at any point (u, v) lies in the plane of the boundary curves. The same argument can be extended to $\mathbf{P}_v(u, v)$.

The surface normal is given by

$$\mathbf{N}(u, v) = \mathbf{P}_u \times \mathbf{P}_v = (\mathbf{A} + v\mathbf{B}) \times (\mathbf{C} + u\mathbf{D})$$

which is perpendicular to the plane of $\mathbf{P}_u$ and $\mathbf{P}_v$ and, therefore, the plane of the boundary curves. Thus for any point on the surface, the direction of the surface normal is constant (the magnitude depends on the point) or the unit normal is fixed in space. Knowing that the plane surface is the only surface that has a fixed unit normal, we conclude that a bilinear Coons patch degenerates to a plane if its boundary curves are coplanar. Thus, this patch can be used to create planes with curved boundaries similar to the bicubic surface covered in Example 5.6. Note, however, that a bicubic Coons patch or any other patch that has nonlinear blending functions does not reduce to a plane when all its boundaries are coplanar.

5.6.5 Blending Surface

This is a surface that connects two nonadjacent surfaces or patches. The blending surface is usually created to manifest C^0 and C^1 continuity with the two given patches. The fillet surface shown in Fig. 5.11 is considered a special case of a blending surface. Figure 5.42 shows a general blending surface. A bicubic surface

can be used to blend patch 1 and patch 2 with both C^0 and C^1 continuity. The corner points P_1, P_2, P_3 and P_4 of the blending surface and their related tangent and twist vectors are readily available from the two patches. Therefore, the $[B]$ matrix of the blending surface can be evaluated. A bicubic blending surface is suitable to blend cubic patches, that is, bicubic Bezier or B-spline patches.

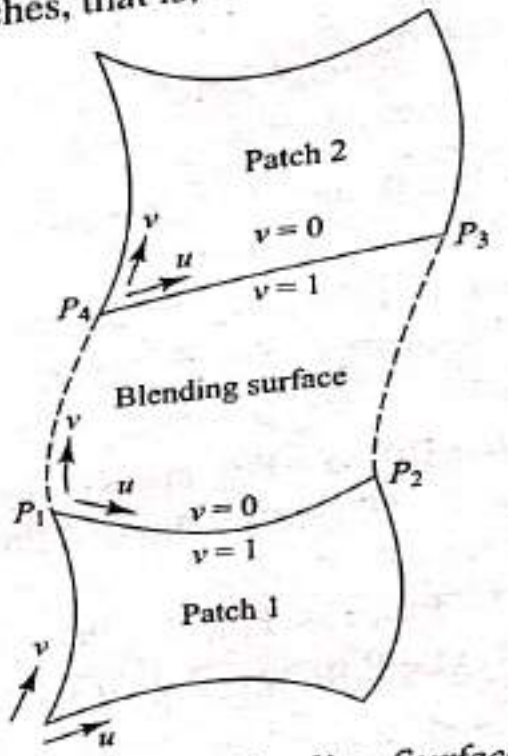


Fig. 5.42 *A Blending Surface*

For patches of other orders, a B-spline blending surface may be generated in the following scenario. A set of points and their related v-tangent vectors beginning with P_1 and ending with P_2 can be generated along the $v = 1$ edge of patch 1. Similarly, a corresponding set can be generated along the $v = 0$ edge of patch 2. Cubic spline curves can now be created between the two sets. These curves can be used to generate an ordered rectangular set of points that can be connected with the B-spline surface which becomes the blending surface. Some CAD/CAM systems allow users to connect a given set of curves with a B-spline surface directly. In the case of the fillet surface shown in Fig. 5.11, a fillet radius is used to generate the surface. Here, the rectangular set of points to create the B-spline surface can be generated by creating fillets between corresponding v = constant curves on both patches. In turn, points can be generated on these fillets.

5.6.6 Offset Surface

If an original patch and an offset direction are given as shown in Fig. 5.12, the equation of the resulting offset patch can be written as:

$$\mathbf{P}(u, v)_{\text{offset}} = \mathbf{P}(u, v) + \hat{\mathbf{n}}\,(u, v)\, d(u, v) \tag{5.104}$$

where $\mathbf{P}(u, v)$, $\hat{\mathbf{n}}\,(u, v)$ and $d(u, v)$ are the original surface, the unit normal vector at point (u, v) on the original surface and the offset distance at point (u, v) on the original surface respectively. The unit normal $\hat{\mathbf{n}}\,(u, v)$ is the offset direction shown in Fig. 5.12. The distance $d(u, v)$ enables generating uniform or tapered thickness surfaces depending on whether $d(u, v)$ is constant or varies linearly in u, v or both.

5.6.7 Triangular Patches

Triangular patches are useful if the given surface data points form a triangle or if a given surface cannot be modeled by rectangular patches only and may require at least one triangular patch. In tensor product surfaces, the parameters are u and v and the parametric domain is defined by the unit square of $0 \le u \le 1$ and $0 \le v \le 1$. In triangulation techniques, three parameters u, v and w are used and the parametric domain is defined by a symmetric unit triangle of $0 \le u \le 1$, $0 \le v \le 1$ and $0 \le w \le 1$, as shown in Fig. 5.43. The coordinates u, v and w are called "barycentric coordinates." While the coordinate w is not independent of u and v (note that $u + v + w = 1$ for any point in the domain), it is introduced to emphasize the symmetry properties of the barycentric coordinates.

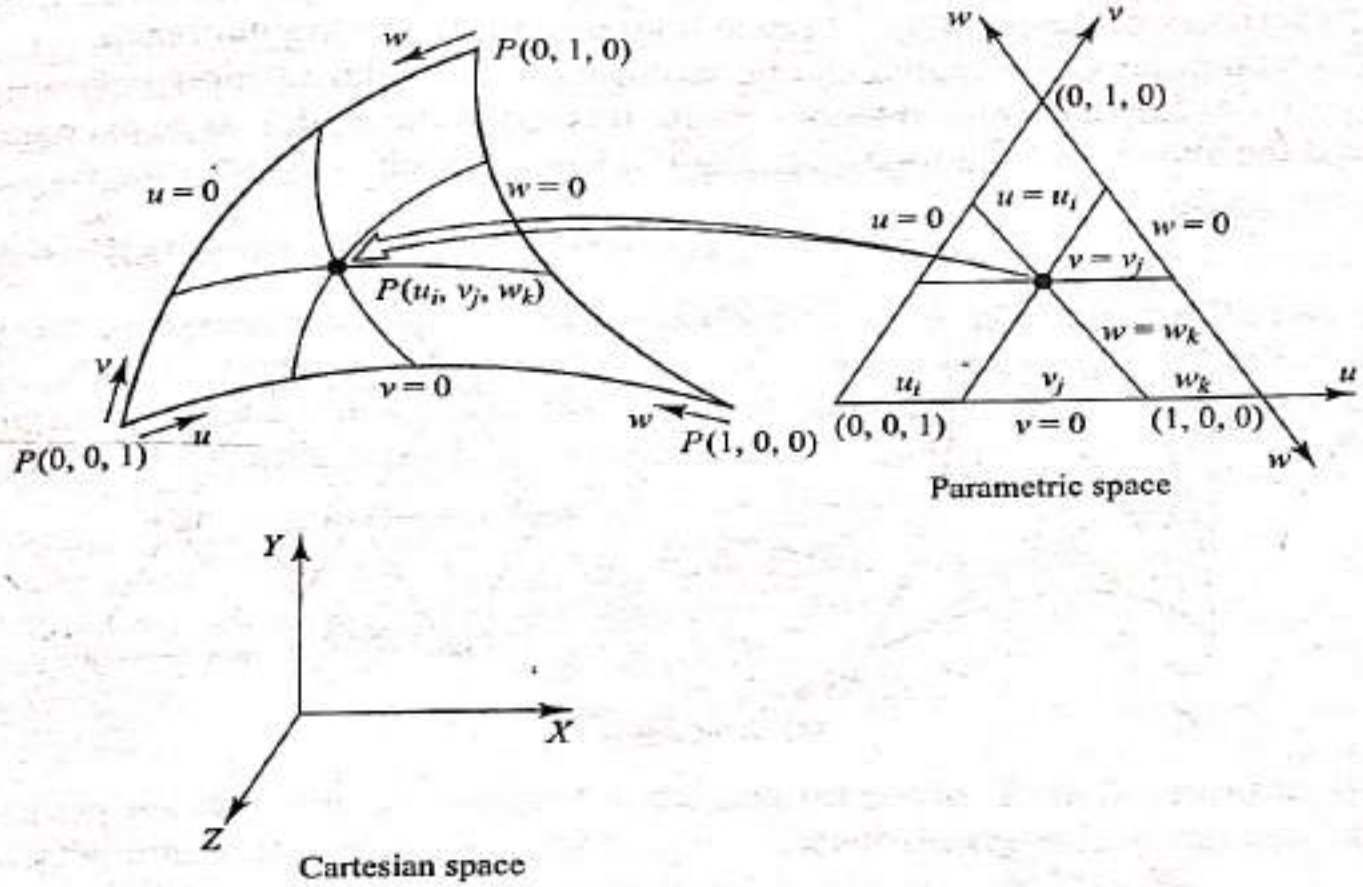


Fig. 5.43 *Representation of a Triangular Patch*

The formulation of triangular polynomial patches follows a somewhat similar pattern to that of tensor product patches. For example, a triangular Bezier patch is defined by

$$\mathbf{P}(u, v, w) = \sum_{i, j, k} \mathbf{P}_{ijk}\, B_{i, j, k, n}(u, v, w), \quad 0 \le u \le 1, 0 \le v \le 1, 0 \le w \le 1 \tag{5.105}$$

where $i, j, k \ge 0$, $i + j + k = n$ and n is the degree of the patch. The $B_{i, j, k, n}$ are Bernstein polynomials of degree n:

$$B_{i,j,k,n} = \frac{n!}{i!\, j!k!} u^i v^j w^k \tag{5.106}$$

The coefficients $\mathbf{P}_{i,j,k}$ are the control or data points that form the vertices of the control polygon. The number of data points required to define a Bezier patch of degree n is given by $(n+1)(n+2)/2$. Figure 5.44 shows cubic and quartic triangular Bezier patches with their related Bernstein polynomials. The order of inputting data points should follow the pyramid organization of a Bernstein polynomial shown in the figure. For example, 15 points are required to create a quartic Bezier patch and must be input in five rows. The first row has five points $(n + 1)$ and each successive row has one point less than its predecessor until we reach the final row that has only one point. This pattern of input can be achieved symmetrically from any direction as shown. Note that the degree of the triangular Bezier patch is the same in all directions, in contrast to the rectangular patch which can have n- and m-degree polynomials in u and v directions respectively. However, all the characteristics of the rectangular patch hold true for the triangular patch.

A rectangular Coons patch can be modified in a similar fashion as described above to develop a triangular Coons patch. It is left to the reader, as an exercise, to extend the above formulation to a triangular Coons patch.

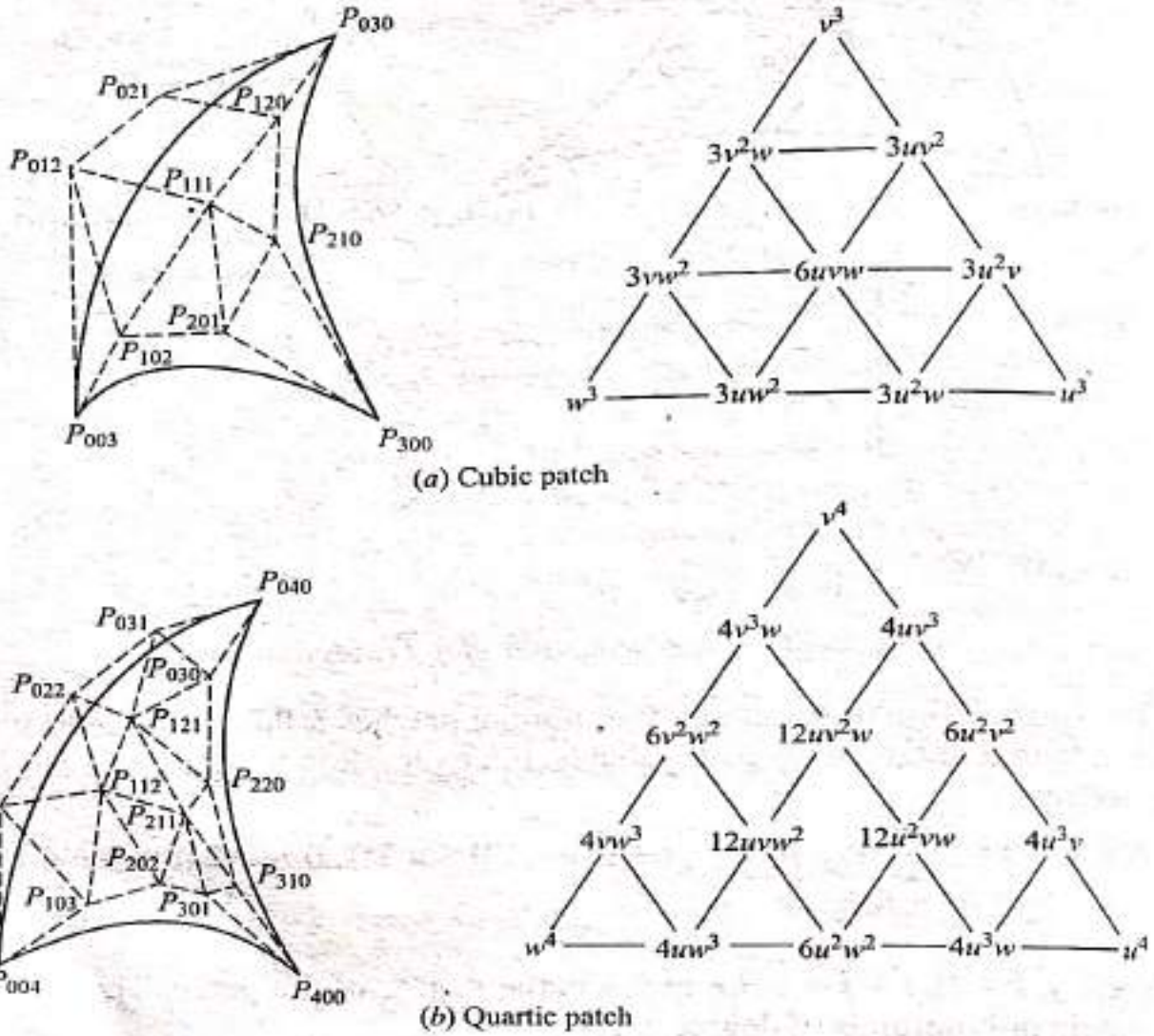


Fig. 5.44 *Triangular Bezier Patches*

160, 195, 200, 207, 218 → module-2

195 - 218

245 - 249

module-3 → 262, 266, 267, 268, 270, 278, 282, 287, 291

292

262 - 294

328

module-4 → 338, 351, 371 →

323 - 343

335 - 345

351 - 357

371 - 385

module-5 → 517, 537, 539, 550, 565 (517 - 527) (534 - 560)

565 - 569

module-6 →