CAD/CAM THEORY AND PRACTICE

Ibrahim Zeid

Department of Mechanical Engineering
Northeastern University

McGraw-Hill, Inc.

New York St. Louis San Francisco Auckland Bogotá Caracas
Hamburg Lisbon London Madrid Mexico Milan Montreal New Delhi
Paris San Juan São Paulo Singapore Sydney Tokyo Toronto

CHAPTER 5

TYPES AND MATHEMATICAL REPRESENTATIONS OF CURVES

5.1 INTRODUCTION

CAD tools have been defined in Chap. 1 as the melting pot of three disciplines: design, geometric modeling, and computer graphics. While the latter discipline is covered in Part III, this part discusses geometric modeling and its relevance to CAD/CAM. Early CAD/CAM systems focused on improving the productivity of draftsmen. More recently, they have focused on modeling engineering objects. As a result, geometric models that once were more than adequate for drafting purposes are not acceptable for engineering applications. A basic requirement, therefore, is that a geometric model should be an unambiguous representation of its corresponding object. That is to say, the model should be unique and complete to all engineering functions from documentation (drafting and shading) to engineering analysis to manufacturing.

A geometric model of an object and its related database have been defined in Chap. 3. The three types of geometric models, wireframes, surfaces, and solids, are covered in Chaps. 5, 6, and 7 respectively. Each chapter presents the available types of entities of the modeling technique and their related mathematical representations to enable good understanding of how and when to use these entities in engineering applications. Users usually have to decide on the type of modeling technique based on the ease of using the technique during the construction phase and on the expected utilization of the resulting database later in the design and manufacturing processes. Regardless of the chosen technique, the user constructs a geometric model of an object on a CAD/CAM system by inputting the object data as required by the modeling technique via the user interface provided by the software. The software then converts such data into a mathematical representation which it stores in the model database for later use. The user may retrieve and/or modify the model during the design and/or manufacturing processes.

To convey the importance of geometric modeling to the CAD/CAM process, one may refer to other engineering disciplines and make the following analogy. Geometric modeling to CAD/CAM is as important as governing equilibrium equations to classical engineering fields as mechanics and thermal fluids. From an engineering point of view, modeling of objects is by itself unimportant. Rather, it is a means (tool) to enable useful engineering analysis and judgment. As a matter of fact, the amount of time and effort a designer spends in creating a geometric model cannot be justified unless the resulting database is utilized by the applications module discussed in Chap. 3.

The need to study the mathematical basis of geometric modeling is manyfold. From a strictly modeling point of view, it provides a good understanding of terminology encountered in the CAD/CAM field as well as CAD/CAM system documentation. It also enables users to decide intelligently on the types of entities necessary to use in a particular model to meet certain geometric requirements such as slopes and/or curvatures. In addition, users become able to interpret any unexpected results they may encounter from using a particular CAD/CAM system. Moreover, those who are involved in the decision-making process and evaluations of CAD/CAM systems become equipped with better evaluation criteria.

From an engineering and design point of view, studying geometric modeling provides engineers and designers with new sets of tools and capabilities that they can use in their daily engineering assignments. This is an important issue because, historically, engineers cannot think in terms of tools they have not learned to use or been exposed to. The tools are powerful if utilized innovatively in engineering applications. It is usually left to the individual imagination to apply these tools usefully to applications in a new context. For example, the mere fact that CAD/CAM databases are centralized and associative provides great capabilities that are utilized in Sec. 5.8 of this chapter. These capabilities are usually more efficient than writing analyses and plotting programs on conventional computers.

Having established the need for geometric modeling, what is the most useful geometric model to engineering applications? Unfortunately, there is no direct answer to this question. Nevertheless, the following answer may be offered. In this book, the answer has two levels. At one level, engineers may agree that some sort of geometry is required to carry engineering analysis. The degree of geometric detail depends on the analysis procedure that utilizes the geometry. Engineers may also agree that there is no model that is sufficient to study all behavioral aspects of an engineering component or a system. A machine part, for example, can be modeled as a lumped mass rigid body on one occasion or as a distributed mass continuum on another occasion.

At the second level, the adequacy of geometry or a geometric model to an analysis procedure is decided by its related useful attributes to that procedure. Attributes of geometry is never an issue for manual procedures because the engineer's mind coordinates all the related facts and information. In computer-based modeling and analysis, attributes attached to geometric models determine their relevance to design, analysis, and manufacturing. Current geometric models offered by CAD/CAM systems seem to have adequate and enough geometric and visualization (colors and shades) attributes. Based on these attributes, they are utilized successfully in applications such as mass property calculations, mechanism analysis, finite element modeling, and NC. If these attributes are insufficient or must be reorganized for other applications, then new software must be written based on the existing database structure of the geometric model which may be modified to accept new attributes; or a completely new structure may have to be developed. In conclusion, the study of existing geometric models provides designers and engineers with the capabilities and limitations of these models and paves the road for them to utilize the attributes of these models or create new ones to benefit new engineering applications.

This chapter covers the available types and most useful mathematical representations of curves. Sections 5.1 through 5.7 cover the basic related topics. Section 5.8 applies these topics to design and engineering applications to demonstrate their usefulness.

5.2 WIREFRAME MODELS

A wireframe model of an object is the simplest, but mose verbose, geometric model that can be used to represent it mathematically in the computer. It is sometimes referred to as a stick figure or an edge representation of the object. The word "wireframe" is related to the fact that one may imagine a wire that is bent to follow the object edges to generate the model. Typically, a wireframe model consists entirely of points, lines, arcs and circles, conics, and curves. Wireframe modeling is the most commonly used technique and all commercial CAD/CAM systems are wireframe-based.

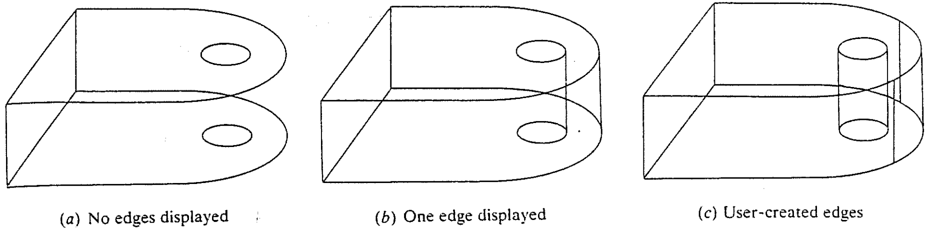
Early wireframe modeling techniques developed in the 1960s were strictly two dimensional and were designed to automate drafting and simple NC. Two-dimensional wireframe models contained enough useful information to perform the NC work. Users had to construct geometry in the desired various views independently due to the lack of centralization and associativity of the resulting database. Later in the early 1970s, the centralized associative database concept enabled modeling of three-dimensional objects as wireframe models that can be subject to three-dimensional transformations. Creating geometry in one view is automatically projected and displayed in other views. This represents a substantial saving and flexibility over manual design and drafting.

In constructing a wireframe model on a CAD/CAM system, the user should follow the modeling guidelines discussed in Chap. 3. The detailed step-by-step procedures to create models may vary according to system capabilities and the user's individual habits. In addition to the commands required to create the common wireframe entities, users are provided with other tools that facilitate the model construction. Part IV of the book covers these tools in detail. These tools, in general, help users to manage geometry as well as to avoid unnecessary calculations. For example, typical CAD/CAM systems provide users with possibly three modes to input coordinates: cartesian, cylindrical, or spherical. Each mode has explicit or implicit inputs. Explicit input could be absolute or incremental coordinates. Implicit input involves user digitizes. Another example is the geometric modifiers which automatically identify specific locations such as end- or midpoints of entities that are convenient to access once these entities are created by the system.

Despite its many disadvantages, the major advantages of wireframe modeling is its simplicity to construct. Therefore, it does not require as much computer time and memory as does surface or solid modeling. However, the user or terminal time needed to prepare and/or input data is substantial and increases rapidly with the complexity of the object being modeled. Wireframe modeling is considered a natural extension of traditional methods of drafting. Consequently, it does not require extensive training of users; nor does it demand the use of unusual terminology as surfaces and solids. Wireframe models form the basis for surface models. Most existing surface algorithms require wireframe entities to generate surfaces (refer to Chap. 6). Lastly, the CPU time required to retrieve, edit, or update a wireframe model is usually small compared to surface or solid models.

The disadvantages of wireframe models are manyfold. Primarily, these models are usually ambiguous representations of real objects and rely heavily on human interpretation. A wireframe model of a box offers a typical example where the model may represent more than one object depending on which face(s) is assumed to exist. Models of complex designs having many edges become very confusing and perhaps even impossible to interpret. To overcome this confusion, lines can be hidden, dashed, or blanked. If done manually, these operations are very tedious, error-prone, and can result in "nonsense" objects. Automatic hidden line removal algorithms based on wireframe modeling are usually helpful. Another disadvantage is the lack of visual coherence and information to determine the object profile. The obvious example is the representation of a hole or a curved portion of the object. In most systems, the hole is displayed as two parallel circles separated by the hole length. Some systems may connect a line between the two circles on one side of the hole. In many cases, users add edges of the hole for appearance purposes at the drafting mode or may use a cylindrical surface to represent the hole which introduces problems later on during the model clean-up phase. In adding the edges, inexperienced users tend to attempt to create tangent lines beweeen the hole circles which obviously does not work. Figure 5-1 shows possible cases to display holes and/or curved ends of objects. Representing the intersection of plane faces with cylinders, cylinders with cylinders, or tangent surfaces in general is usually a problem in wireframe modeling and requires user manipulations.

Wireframe models are also considered lengthy or verbose when it comes to the amount of defining data and command sequence required to construct them. For example, compare the creation of a simple box as a wireframe and as a solid. In the latter, the location of one corner, the length, width, and height are the required input while in the former the coordinates of at least four corners of one face, the depth, and the edge connectivity are required, considering the box as a two-and-a-half-dimensional object. In other words, both topological and geometrical data are needed to construct wireframe models while solids require only geometrical data (refer to Chap. 7 for the difference between topology and geometry). In addition, WCSs and construction planes always need to be defined to facilitate model construction.


(*a*) No edges displayed

(*b*) One edge displayed

(*c*) User-created edges

FIGURE 5-1
Displaying holes and curved ends in wireframe models.

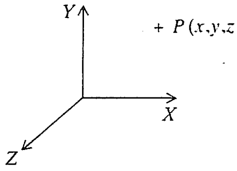
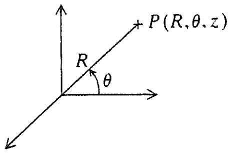
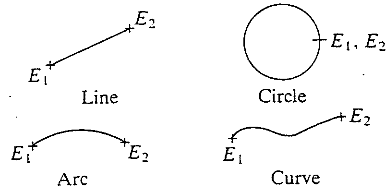
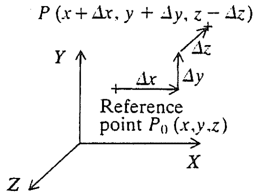
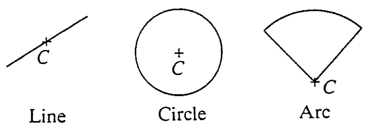
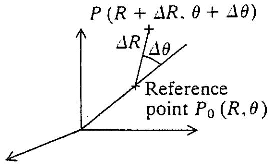
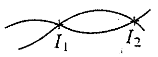
From an application, and consequently engineering, point of view, wireframe models are of limited use. Unless the object is two-and-a-half dimensional, volume and mass properties, NC tool path generation, cross-sectioning, and interference detections cannot be calculated. The model can, however, be used in manual finite element modeling and tolerance analysis.

Despite the above-mentioned limitations, wireframe models are expected to last and may extend to certain classes of solid modeling. For example, some solid modelers (such as Medusa) are based on wireframe input. The simplicity of the geometrical concepts based on wireframe modeling makes them attractive to use to introduce users to the CAD/CAM field. In addition, at early design stages, designers might just need a sketchpad to try various ideas. Wireframes are ideal to provide them with such a capability. From an industrial point of view, wireframe models may be sufficient to many design and manufacturing needs. From a practical point of view, many companies have large amounts of wireframe databases that are worth millions of dollars and man-hours and therefore make it impossible to get rid of wireframe technology.

5.3 WIREFRAME ENTITIES

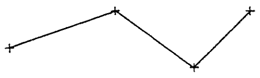
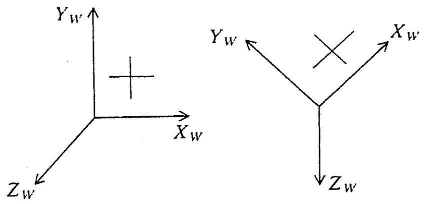
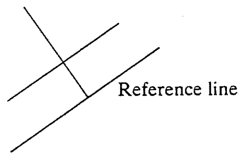
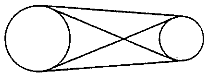
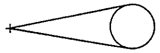
All existing CAD/CAM systems provide users with basic wireframe entities which can be divided into analytic and synthetic entities. Analytic entities are points, lines, arcs and circles, fillets and chamfers, and conics (ellipses, parabolas, and hyperbolas). Synthetic entities include various types of spline [cubic spline, B-spline, β(beta)-spline, ν(nu)-spline] and Bezier curves. The mathematical properties of each entity and how it is used in engineering applications or converted into a user interface are covered in the remainder of the chapter. It is quite common for a user to be faced with many modifiers that can be used to create a particular entity on a particular CAD/CAM system. Knowledge of the basics of such an entity can only increase user productivity. Choosing the proper modifier for a given input can save unnecessary calculations. Also, knowing a curve behavior in relation to its input data can save time trying to achieve the impossible.

TABLE 5.1
Methods of defining points

Explicit methods	Implicit methods
1. Absolute cartesian co-ordinates 	A digitize d (with or without an active grid) + d Coordinates of resulting point can be obtained by using the "verify" command. Coordinates are measured relative to the MCS as discussed in Chap. 3
2. Absolute cylindrical coordinates  (Spherical coordinates are seldom used in practice)	Endpoint of an existing entity†  E₂ E₁ Line E₁, E₂ Circle E₁ E₂ Arc E₂ E₁ Curve
3. Incremental cartesian coordinates  P (x+Δx, y + Δy, z − Δz) Δz Y Δx Δy Reference point P₀ (x,y,z) X Z	Centerpoint (origin) of an existing entity† 
4. Incremental cylindrical coordinates  P (R + ΔR, θ + Δθ) ΔR Δθ Reference point P₀ (R,θ) (Some CAD/CAM systems require moving the current WCS to the reference point P_0 to avoid unexpected results)	Intersection point of two existing entities† 

† More details of these methods are covered in Sec. 11.2, Chap. 11.

TABLE 5.2
Methods of defining lines

Method	Illustration
1. Points defined by any method of Table 5.1	
2. Horizontal (parallel to the X axis of the current WCS) or vertical (parallel to the Y axis of the current WCS)	
3. Parallel or perpendicular to an existing line	
4. Tangent to existing entities	 One of the four possibilities is obtained depending on user digitizes of the two circles  One of the two possibilities is obtained

Tables 5.1 to 5.5 show the most common methods utilized by CAD/CAM systems to create wireframe entities. Readers are advised to compare and/or modify these methods to what their respective systems offer and what user interfaces of these systems require. As an example, defining a point using the endpoint method requires the " END" or "PND (point end)" modifier on the Computervision or Calma system respectively. Such similarities among systems exist in all

software modules, simply because they all share the same theory. Readers are also advised to find commands corresponding to these tables on their particular CAD/CAM systems. The following are some examples to illustrate using wireframe modeling techniques. These examples are independent of any system or user interface and readers can simply convert them into a command sequence of their choice.

TABLE 5.3
Methods of defining arcs and circles

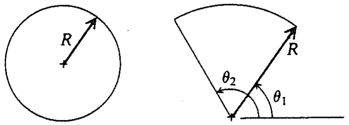
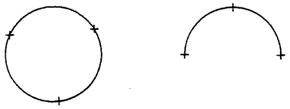
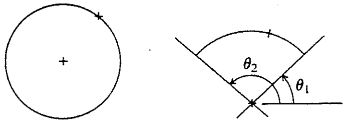
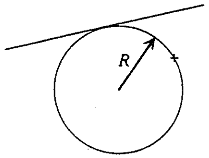
Method	Illustration
1. Radius or diameter and center. In the case of an arc, beginning and ending angles θ_1 and θ_2 are required	
2. Three points defined by any method of Table 5.1	
3. Center and a point on the circle	
4. Tangent to line, pass through a given point, and with a given radius	

TABLE 5.4
Methods of defining ellipses and parabolas

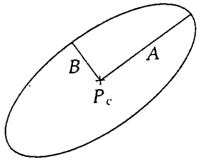
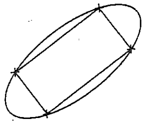
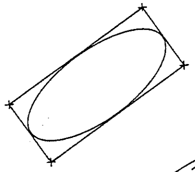
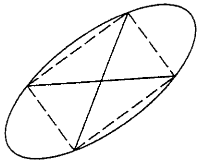
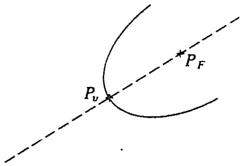
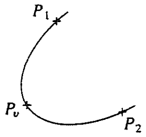
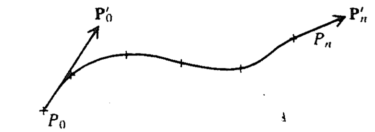
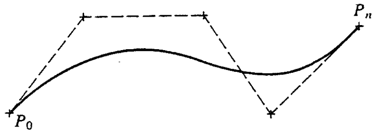
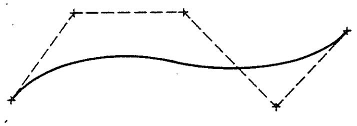
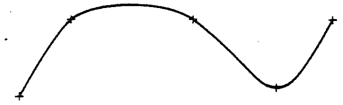
Methods	Illustration
1. *Ellipses*	
(*a*) Center and axes lengths	
(*b*) Four points	 
(*c*) Two conjugate diameters	
2. *Parabolas*	
(*a*) Vertex and focus	
(*b*) Three points	

TABLE 5.5
Methods of defining synthetic curves

Method	Illustration
1. *Cubic spline* A given set of data points and start and end slopes	
2. *Bezier curves* A given set of data points	
3. *B-spline curves* (*a*) Approximate a given set of data points	
(*b*) Interpolate a given set of data points	

Example 5.1. For the guide bracket shown in Fig. 5-2:

(*a*) Create the model database utilizing a CAD/CAM system.
(*b*) Obtain the orthographic views of the model.
(*c*) Obtain a final drawing of the model.

Solution

(*a*) An efficient planning strategy is required before logging into the CAD/CAM system. Examining Fig. 5-2 reveals that the geometric model of the guide bracket is a two-and-a-half-dimensional model. It is symmetric with respect to the model central vertical plane if the cut on the top right corner is ignored. Let us choose the origin of the MCS at point A shown in the figure. The orientation of the MCS is chosen properly, as discussed in Chap. 3, so that the front view of the bracket is defined as shown. Assume the orientation shown to facilitate discussion. The following steps may be followed to construct the model:

1. Follow the part setup procedure of the CAD/CAM system. Define an isometric view to begin constructing the model.

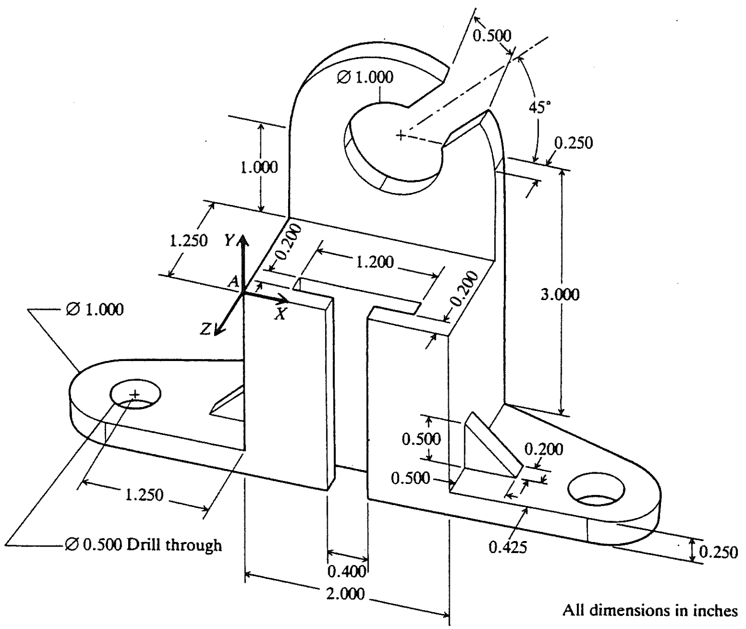
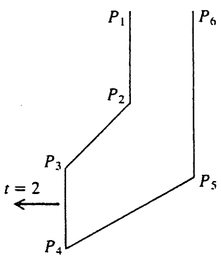


FIGURE 5-2
Guide bracket.

2. Begin by constructing the right face using a line command with the following sequence of point input:

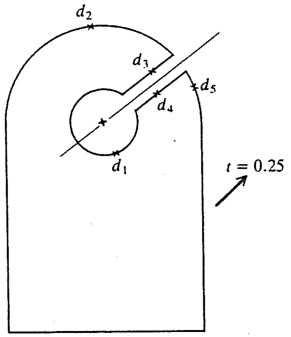
Point	Coordinate input relative to MCS		
	x	y	z
P_1	2[†]	1.00	−1.25
P_2	–[‡]	$\Delta y = -1.00$[§]	–
P_3	–	–	0
P_4	–	$y = -2$	–
P_5	–	–	$z = -1.5$
P_6	–	$\Delta y = 3.00$	–

[†] Absolute coordinate input.
[‡] System defaults to the last input value.
[§] Relative (incremental) coordinate input. Incremental coordinate input is very efficient in this case.



3. Project that face a distance of 2 inches in the direction shown above (X axis).

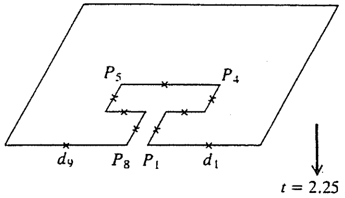
4. Construct the top part of the front face of the model as shown below:



Trimming, offsetting, circle, and arc commands are useful. Cylindrical input is needed to create the line at 45°.

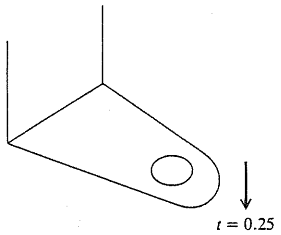
5. Project the entities labeled by digitizes d_1 to d_5 a distance of 0.25 inch in the direction shown above (Z axis).
6. Create the slot in the top face using a line command with the point sequence shown below:

Point	Coordinate input relative to MCS		
	x	y	z
P_1	1.2	0	0
P_2	—	—	$\Delta z = -0.2$
⋮	⋮	⋮	⋮
P_8	—	—	0



Trim the front horizontal line between points P_1 and P_8.

7. Project the entities labeled by digitizes d_1 to d_9 a distance of 2.25 inches in the direction shown above (Y axis).
8. Construct the top face of the right flange of the bracket as shown below:

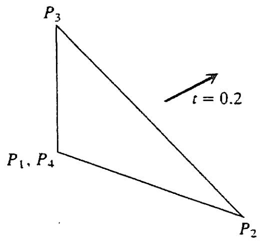


First construct the two circles and then the two tangents. Trim the outer circle to finish the construction.

9. Project that top face a distance of 0.25 inch in the direction shown (Y axis).

10. Construct the front face of the right support (wedge) using a line command with the following point sequence:

Point	Coordinate input relative to MCS		
	x	y	z
P_1	2	−2	−0.425
P_2	$\Delta x = 0.5$	—	—
P_3	2	−1.5	—
P_4	—	−2	—



11. Project that face a distance of 0.2 inch in the direction shown (Z axis).
12. Using the mirror command, copy the entities that resulted in steps 8 to 11 to create the left flange of the model. The mirror plane is the vertical central plane of the model. To facilitate using this command, a different layer (see Chap. 11) should be used for steps 8 to 11 to isolate the entities to be mirrored from the others.
13. Connect any missing lines and obtain a hard copy or a plot of the model.
14. File the model to save it.

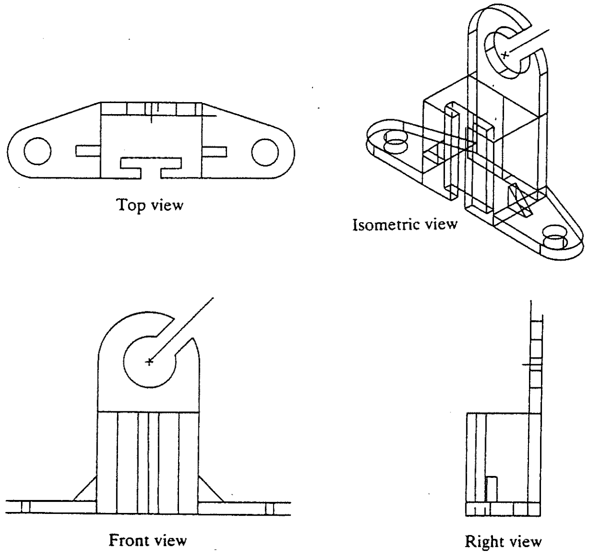


FIGURE 5-3
Orthographic views of guide bracket.

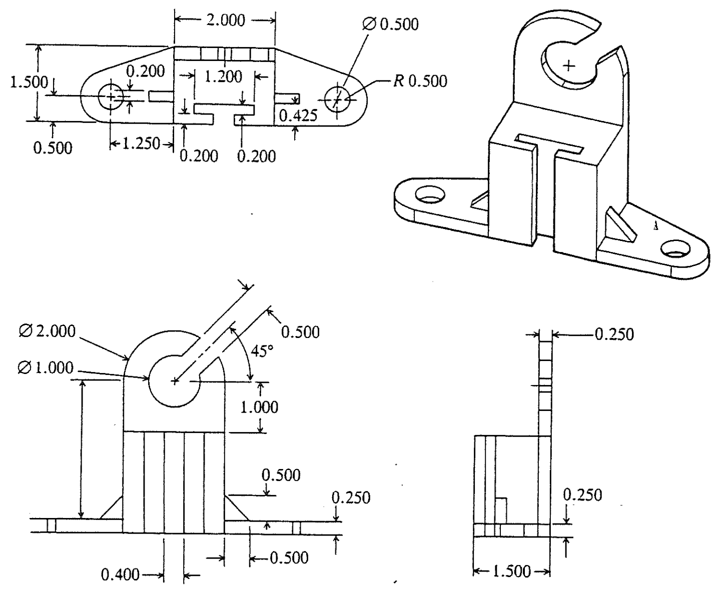
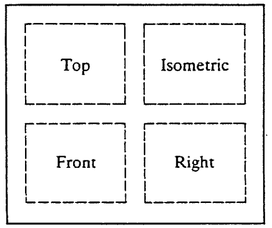


FIGURE 5-4
Final drawing of guide bracket.

(*b*) Using a new drawing or screen layout, define views as shown below:



Follow all discussions in Chap. 3 to define these views. The result is shown Fig. 5-3.

(*c*) Using the model clean-up procedure on the system, the model views are cleaned according to the drafting conventions and the drawing of the model is shown in Fig. 5-4.

This example illustrates most of the experiences encountered in creating two-and-a-half-dimensional wireframe models on major CAD/CAM systems with their related drafting work.

Example 5.2. For the stop block shown in Fig. 5-5:

(*a*) Create the model database utilizing a CAD/CAM system.
(*b*) Obtain the isometric view of the model.
(*c*) Obtain the final drawing of the model.

Solution

(*a*) Unlike Example 5.1, the geometric model of the stop block is given by its three orthographic views. All existing mechanical designs are available in this form where their related information are stored in blueprints. While scanning techniques exist to convert blueprints into CAD databases automatically, or at least semi-automatically, this example illustrates how to create a geometric model database from its three orthogonal views. The planning strategy in this case consists of choosing the proper location and orientation of the database MCS and of trying to construct the model by working in four views simultaneously. The general guiding rule should be to minimize the amount of calculations by taking advantage of the centralization and associativity property of the database. For example, the part of the block that is oriented at 30° should be constructed in the front view first and then finished off in the top and right views. Assuming an MCS as shown, the following steps may be followed to create the model database:

1. Follow the part setup procedure of the CAD/CAM system. Define four views (front, top, right, and isometric) to construct the model.

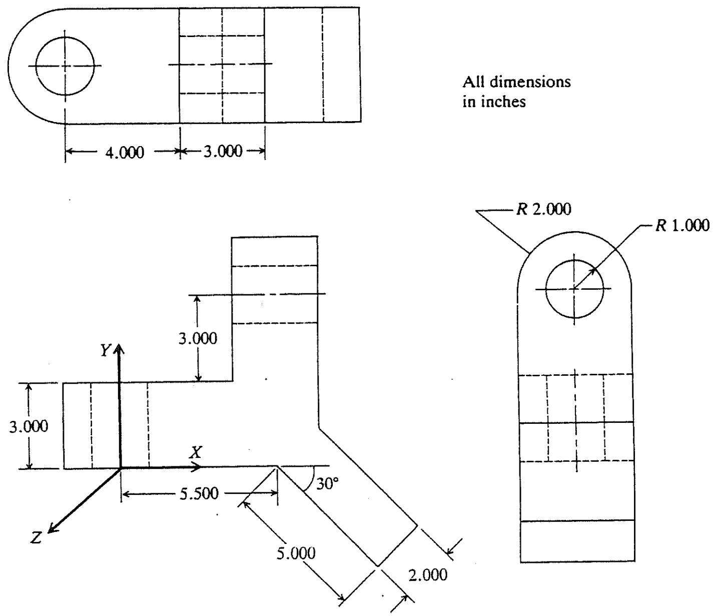
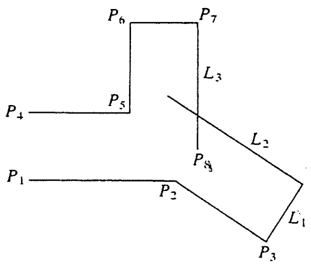


FIGURE 5-5
Stop block.

2. Begin by constructing the front view as shown below:

Point	Coordinate input relative to MCS		
	x	y	z
P_1	0	0	0
P_2	5.5	–	–
P_3	$R = 5$	$\theta = -30$	–
P_4	0	3	0
P_5	$\Delta x = 4$	–	–
P_6	–	$\Delta y = 5$	–
P_7	$\Delta x = 3$	–	–
P_8	–	$\Delta y = -7$	–



Connect P_1, P_2, and P_3 using a line command. Connect P_4, P_5, P_6, P_7, and P_8 using a line command. Create line L_1 with a perpendicular line command. Create line L_2 with a parallel line command. Trim lines L_2 and L_3 to their intersection point.

3. Complete the creation of the top view. Notice that all entities created in the front view are properly projected and displayed in the other three views. An existing entity can be chosen (located) by digitizing it in any view where it is most feasible and accessible. This usually makes construction much simpler.
4. Complete the creation of the right side view. The isometric view is now automatically completed and the four views are shown in Fig. 5-6.

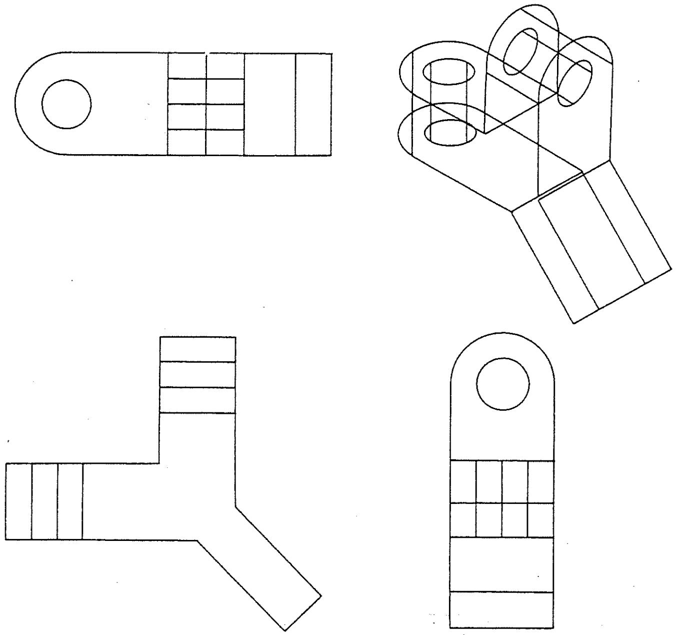


FIGURE 5-6
Views of stop block.

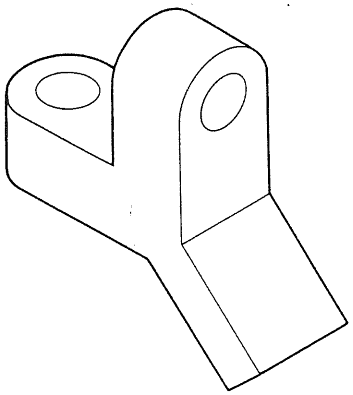


FIGURE 5-7
Isometric view of stop block.

(*b*) Access a new drawing or screen layout and define the whole screen as an isometric view. Clean up the view to obtain Fig. 5-7.

(*c*) Using the clean-up procedure as discussed in Example 5.1, the final drawing of the model is shown in Fig. 5-8.

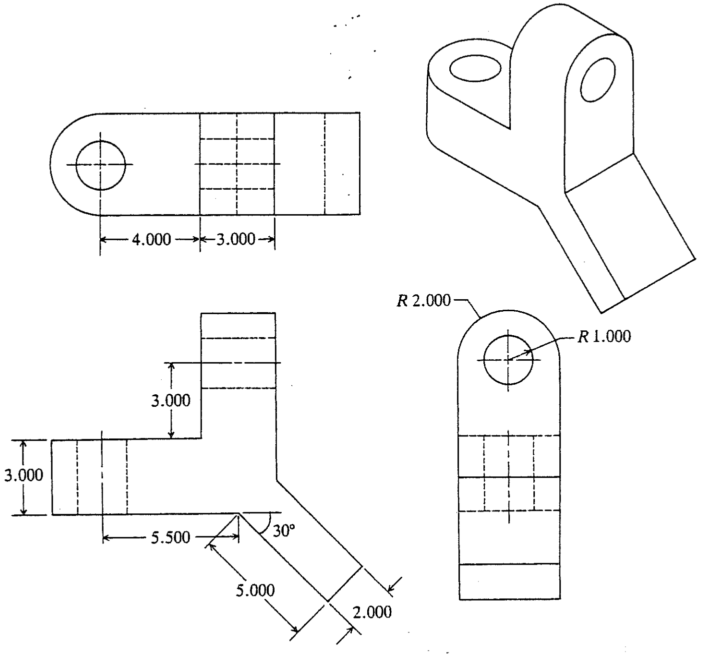
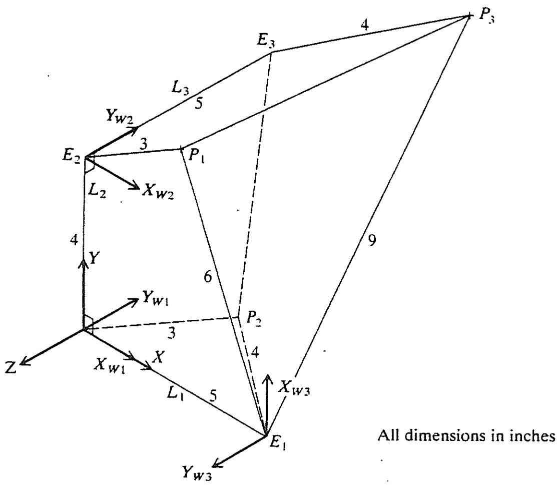


FIGURE 5-8
Final drawing of stop block.



All dimensions in inches

FIGURE 5-9
Adjustment block.

Example 5.3. Create the database of the three-dimensional model shown in Fig. 5-9.

Solution. This is a three-dimensional geometric model. Its construction does not usually follow any general guidelines. Typically, coordinates of its corners are required for construction. However, the general rule of minimizing calculations and utilizing capabilities of various CAD/CAM functions must always be followed. The shortest way to construct this model is discussed below:

1. After the part setup procedure, define an isometric view. The MCS is chosen as shown in Fig. 5-9.
2. Construct line L_1 between points (0, 0, 0) and (5, 0, 0) and L_2 between points (0, 0, 0) and (0, 4, 0).
3. Construct two circles to find the corner P_1. The first circle has a center at E_1 (endpoint of L_1) and a radius of 6, and the second has a center at E_2 (end of line L_2) and a radius of 3. The intersection of the two circles defines P_1.
4. Connect E_1 and P_1 and E_2 and P_1 using the proper line command.
5. Define a $(\text{WCS})_1$ such that its XY plane coincides with the bottom of the model which is in turn assumed to lie in the XZ plane of the MCS.
6. Find point P_2 and construct the bottom edges in the same way as in steps 3 and 4.
7. Construct L_3 by connecting E_2 and point (0, 5, 4). The coordinates of the point are measured with respect to the active $(\text{WCS})_1$.
8. Construct two circles to find the corner P_3. Define a $(\text{WCS})_2$ such that its XY plane coincides with the top view of the model. Construct a circle with center E_3 and radius 4. Define another $(\text{WCS})_3$ with its XY plane coincident with the right view of the model. Construct a circle with center E_1 and radius 9. The intersection of the two circles defines P_3.
9. Connect E_1 and P_3, E_3 and P_3, P_1 and P_3, and P_2 and E_3 to complete the model construction as shown in Fig. 5-10.

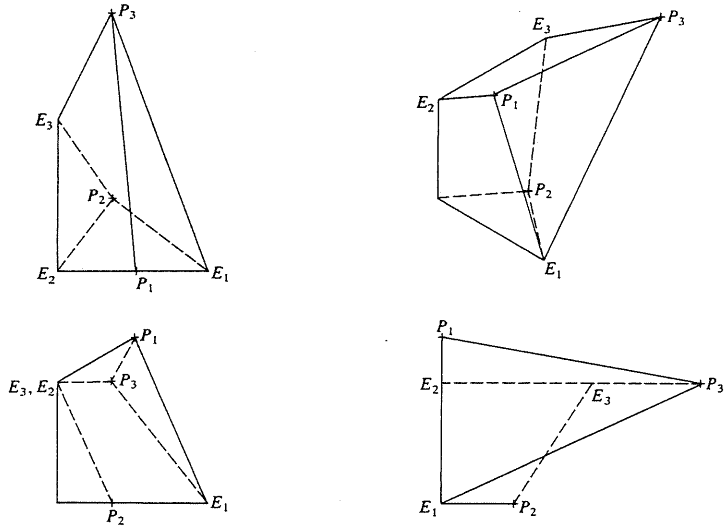


FIGURE 5-10
Views of adjustment block.

5.4 CURVE REPRESENTATION

Examples in the previous section have been chosen to require only simple geometric entities (lines and circles) to provide a good understanding of issues encountered in practicing CAD/CAM technology. However, many applications (automotive and aerospace industries) require other general curves to meet various shape constraints (continuity and/or curvature). The remainder of this chapter covers the basics of these curves.

A geometric description of curves defining an object can be tackled in several ways. A curve can be described by arrays of coordinate data or by an analytic equation. The coordinate array method is impractical for obvious reasons. The storage required can be excessively large and the computation to transform the data from one form to another is cumbersome. In addition, the exact shape of the curve is not known, therefore impairing exact computations such as intersections of curves and physical properties of objects (e.g., volume calculations). From a design point of view, it becomes difficult to redesign shapes of existing objects via the coordinate array method. Analytic equations of curves provide designers with information such as the effect of data points on curve behavior, control, continuity, and curvature.

The treatment of curves in computer graphics and CAD/CAM is different from that in analytic geometry or approximation theory. Curves describing engineering objects are generally smooth and well-behaved. In addition, not every available form of a curve equation is efficient to use in CAD/CAM software due to either computation or programming problems. For example, a curve

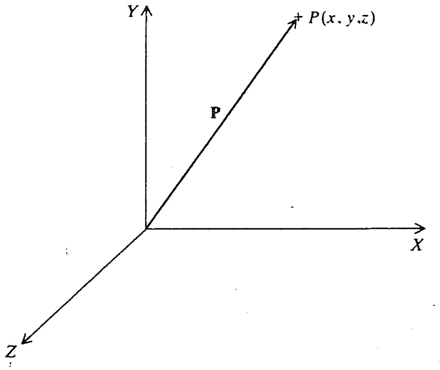


FIGURE 5-11
Position vector of point P.

equation that results in a division by zero while calculating the curve slope causes overflow and errors in calculations. Similarly, if the intersection of two curves is to be found by solving their two equations numerically, the forms of the two equations may be inadequate to program due to the known problems with numerical solutions. In addition, considering that most design data of objects are available in a discrete form, mainly key points, the curve equation should be able to accept points and/or tangent values as input from the designer.

Curves can be described mathematically by nonparametric or parametric equations. Nonparametric equations can be explicit or implicit. For a nonparametric curve, the coordinates y and z of a point on the curve are expressed as two separate functions of the third coordinate x as the independent variable [see Eq. (5.1)]. This curve representation is known as the nonparametric explicit form. If the coordinates x, y, and z are related together by two functions [see Eq. (5.2)], a nonparametric implicit form results. For a parametric curve, on the other hand, a parameter is introduced and the coordinates x, y, and z are expressed as functions of this parameter [see Eq. (5.3)].

Explicit nonparametric representation of a general three-dimensional curve takes the form:

$$\mathbf{P} = [x \quad y \quad z]^T = [x \quad f(x) \quad g(x)]^T \tag{5.1}$$

where $\mathbf{P}$ is the position vector of point P as shown in Fig. 5-11. Equation (5.1) is a one-to-one relationship. Thus, this form cannot be used to represent closed (e.g., circles) or multivalued curves (e.g., parabolas). The implicit nonparametric representation can solve this problem and is given by the intersection of two surfaces as

$$\begin{aligned} F(x, y, z) &= 0 \\ G(x, y, z) &= 0 \end{aligned} \tag{5.2}$$

However, the equation must be solved to find its roots (y and z values) if a certain value of x is given. This may be inconvenient and lengthy. Other limitations of nonparametric representations of curves are:

1. If the slope of a curve at a point is vertical or near vertical, its value becomes infinity or very large, a difficult condition to deal with both computationally and programming-wise. Other ill-defined mathematical conditions may result.
2. Shapes of most engineering objects are intrinsically independent of any coordinate system. What determines the shape of an object is the relationship between its data points themselves and not between these points and some arbitrary coordinate system.
3. If the curve is to be displayed as a series of points or straight line segments, the computations involved could be extensive.

Parametric representation of curves overcomes all of the above difficulties. It allows closed and multiple-valued functions to be easily defined and replaces the use of slopes with that of tangent vectors, as will be introduced shortly. In the case of commonly used curves such as conics and cubics, these equations are polynomials rather than equations involving roots. Hence, the parametric form is not only more general but it is also well suited to computations and display. In addition, this form has properties that are attractive to CAD/CAM and the interactive environment, as will be seen later in this chapter.

In parametric form, each point on a curve is expressed as a function of a parameter u. The parameter acts as a local coordinate for points on the curve. The parametric equation for a three-dimensional curve in space takes the following vector form:

$$\mathbf{P}(u) = [x \quad y \quad z]^T = [x(u) \quad y(u) \quad z(u)]^T, \qquad u_{\min} \le u \le u_{\max} \tag{5.3}$$

Equation (5.3) implies that the coordinates of a point on the curve are the components of its position vector. It is a one-to-one mapping from the parametric space (euclidean space E^1 in u values) to the cartesian space (E^3 in x, y, z values), as shown in Fig. 5-12. The parametric curve is bounded by two parametric values $u_{\min}$ and $u_{\max}$. It is, however, convenient to normalize the parametric variable u to have the limits 0 and 1. The positive sense on the curve is the sense in which u increases (Fig. 5-12).

The parametric form as given by Eq. (5.3) facilitates many of the useful related computations in geometric modeling. To check whether a given point lies on the curve or not reduces to finding the corresponding u values and checking whether that value lies in the stated u range. Points on the curve can be computed by substituting the proper parametric values into Eq. (5.3). Geometrical transformations (as discussed in Chap. 9) can be performed directly on parametric equations. Parametric geometry can be easily expressed in terms of vectors and matrices which enables the use of simple computation techniques to solve complex analytical geometry problems. In addition, common forms for curves which are extendable to surfaces can be found. For example, a cubic polynomial, and a cubic polynomial for surfaces, can describe a three-dimensional curve sufficiently in space. Furthermore, curves defined by Eq. (5.3) are inherently bounded and, therefore, no additional geometric data is needed to define boundaries. Lastly, the parametric form is better suited for display by the special graphics hardware. Numerical values of coordinates of points on a curve can control the deflection of the electron beam of a graphics display, as discussed in Chap. 2.

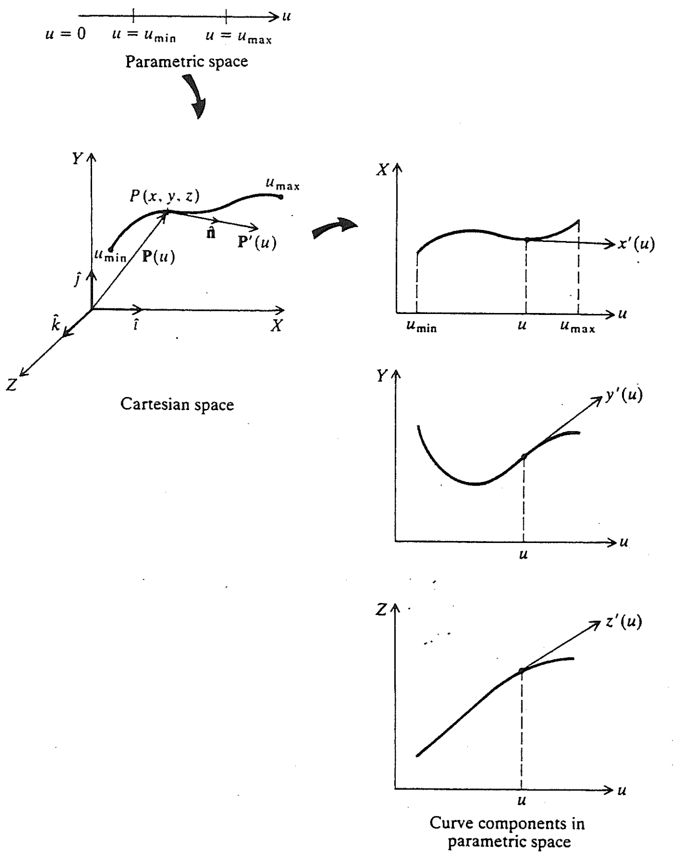


FIGURE 5-12
Parametric representation of a three-dimensional curve.

To evaluate the slope of a parametric curve at an arbitrary point on it, the concept of the tangent vector must be introduced. As shown in Fig. 5-12, the tangent vector is defined as vector $\mathbf{P}'(u)$ in the cartesian space such that

$$\mathbf{P}'(u) = \frac{d\mathbf{P}(u)}{du} \tag{5.4}$$

Substituting Eq. (5.3) into Eq. (5.4) yields the components of the tangent vector in the parametric space as

$$\mathbf{P}'(u) = [x' \quad y' \quad z']^T = [x'(u) \quad y'(u) \quad z'(u)]^T, \qquad u_{\min} \leq u \leq u_{\max} \tag{5.5}$$

where $x'(u)$, $y'(u)$, and $z'(u)$ are the first parametric derivatives (with respect to u) of the position vector components $x(u)$, $y(u)$, and $z(u)$ respectively. The slopes of the curve are given by the ratios of the components of the tangent vector:

$$\frac{dy}{dx} = \frac{dy/du}{dx/du} = \frac{y'}{x'}$$
$$\frac{dz}{dy} = \frac{z'}{y'} \quad \text{and} \quad \frac{dx}{dz} = \frac{x'}{y'} \tag{5.6}$$

The tangent vector has the same direction as the tangent to the curve—hence the name "tangent vector." The magnitude of the vector is given by

$$|\mathbf{P}'(u)| = \sqrt{x'^2 + y'^2 + z'^2} \tag{5.7}$$

and the direction cosines of the vector are given by

$$\hat{\mathbf{n}} = \frac{\mathbf{P}'(u)}{|\mathbf{P}'(u)|} = n_x\hat{\mathbf{i}} + n_y\hat{\mathbf{j}} + n_z\hat{\mathbf{k}} \tag{5.8}$$

where $\hat{\mathbf{n}}$ is the unit vector (Fig. 5-12) with cartesian space components n_x, n_y, and n_z. The magnitude of the tangent vectors at the two ends of a curve affects its shape and can be used to control it, as will be seen later.

There are two categories of curves that can be represented parametrically: analytic and synthetic. Analytic curves are defined as those that can be described by analytic equations such as lines, circles, and conics. Synthetic curves are the ones that are described by a set of data points (control points) such as splines and Bezier curves. Parametric polynomials usually fit the control points. While analytic curves provide very compact forms to represent shapes and simplify the computation of related properties such as areas and volumes, they are not attractive to deal with interactively. Alternatively, synthetic curves provide designers with greater flexibility and control of a curve shape by changing the positions of the control points. Global as well as local control of the shape can be obtained as discussed in Sec. 5.6.

Example 5.4. The nonparametric implicit equation of a circle with a center at the origin and radius R is given by $x^2 + y^2 = R^2$. Find the circle parametric equation. Using the resulting equation, find the slopes at the angles 0, 45, and 90°.

Solution. In general, the parametric equation of a curve is not unique and can take various forms. For a circle, let

$$x = R\cos 2\pi u, \qquad 0 \leq u \leq 1$$

Substituting into the circle equation gives $y = R\sin 2\pi u$ and the parametric equation of the circle becomes

$$\mathbf{P}(u) = [R\cos 2\pi u \quad R\sin 2\pi u]^T, \qquad 0 \leq u \leq 1$$

and the tangent vector is

$$\mathbf{P}'(u) = [-2\pi R\sin 2\pi u \quad 2\pi R\cos 2\pi u]^T, \qquad 0 \leq u \leq 1$$

The parameter u takes the values 0, 0.125, and 0.25 at the angles 0, 45, and 90° respectively, and the corresponding tangent vectors are given by

$$\mathbf{P}'(0) = [0 \quad 2\pi R]^T$$
$$\mathbf{P}'(0.125) = [-2\pi R/\sqrt{2} \quad 2\pi R/\sqrt{2}]^T$$
$$\mathbf{P}'(0.25) = [-2\pi R \quad 0]^T$$

For a two-dimensional curve, the slope is given by y'/x'. The slope at the given angles are calculated as ∞, -1, and 0.

Example 5.5. The parametric equation of the helix shown in Fig. 5-13 with a radius a and a pitch b is given by

$$\mathbf{P}(u) = [a \cos 2\pi u \quad a \sin 2\pi u \quad 2b\pi u]^T, \qquad 0 \le u \le 1$$

Find the nonparametric equation of the helix.

Solution. The helix equation gives

$$x = a \cos 2\pi u \qquad y = a \sin 2\pi u \qquad z = 2b\pi u$$

Solving the first equation for u and substituting in the other two gives the following explicit nonparametric equation [compare with Eq. (5.1)]:

$$\mathbf{P} = \left[x \quad a \sin\left(\cos^{-1}\left(\frac{x}{a}\right)\right) \quad b \cos^{-1}\left(\frac{x}{a}\right)\right]^T$$

Solving the Z component for u and substituting in the x and y equations gives the implicit nonparametric equation as [compare with Eq. (5.2)]

$$x - a \cos\left(\frac{z}{b}\right) = 0$$
$$y - a \sin\left(\frac{z}{b}\right) = 0$$

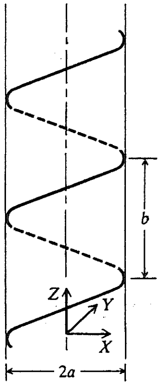


FIGURE 5-13
A helix curve.

5.5 PARAMETRIC REPRESENTATION OF ANALYTIC CURVES

This section covers the basics of the parametric equations of analytic curves that are most widely utilized in wireframe modeling. These developments are related, whenever possible, to the common practice encountered on CAD/CAM systems. This should enable users to fully realize the input parameters they deal with and usually find in system documentation. It should also help developers who may be interested in writing their own CAD/CAM software.

Parametric equations and their developments are presented in vector form. The benefits of the vector form include developing a unified approach and consistent notation to treat both two-dimensional and three-dimensional curves in addition to yielding concise equations that are more convenient to program. The following section provides a quick review of the most relevant vector algebra and analysis needed here. Additional equations can be found in standard linear algebra textbooks.

5.5.1 Review of Vector Algebra

Let **A**, **B**, and **C** be independent vectors, $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ be unit vectors in the X, Y, and Z directions respectively, and K be a constant.

1. Magnitude of a vector is

$$|\mathbf{A}| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

where A_x, A_y, and A_z are the cartesian components of the vector **A**.

2. The unit vector in the direction of **A** is

$$\hat{\mathbf{n}}_A = \frac{\mathbf{A}}{|\mathbf{A}|} = n_{Ax}\hat{\mathbf{i}} + n_{Ay}\hat{\mathbf{j}} + n_{Az}\hat{\mathbf{k}}$$

The components of $\hat{\mathbf{n}}_A$ are also the direction cosines of the vector **A**.

3. If two vectors **A** and **B** are equal, then

$$A_x = B_x \qquad A_y = B_y \qquad \text{and} \qquad A_z = B_z$$

4. The scalar (dot or inner) product of two vectors **A** and **B** is a scalar value given by

$$\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A} = A_x B_x + A_y B_y + A_z B_z = |\mathbf{A}|\,|\mathbf{B}| \cos\theta$$

where θ is the angle between **A** and **B**. Therefore the angle θ between two vectors is given by

$$\cos\theta = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}|\,|\mathbf{B}|}$$

The scalar product can give the component of a vector **A** in the direction of another vector **B** as

$$\mathbf{A} \cdot \hat{\mathbf{n}}_B = |\mathbf{A}| \cos\theta$$

Other properties of the scalar product are:

$$\mathbf{A}\cdot\mathbf{A} = |\mathbf{A}|^2$$
$$\mathbf{A}\cdot\mathbf{B} = \mathbf{B}\cdot\mathbf{A}$$
$$\mathbf{A}\cdot(\mathbf{B}+\mathbf{C}) = \mathbf{A}\cdot\mathbf{B}+\mathbf{A}\cdot\mathbf{C}$$
$$(K\mathbf{A})\cdot\mathbf{B} = \mathbf{A}\cdot(K\mathbf{B}) = K(\mathbf{A}\cdot\mathbf{B})$$
$$\hat{\mathbf{i}}\cdot\hat{\mathbf{i}} = \hat{\mathbf{j}}\cdot\hat{\mathbf{j}} = \hat{\mathbf{k}}\cdot\hat{\mathbf{k}} = 1 \qquad \hat{\mathbf{i}}\cdot\hat{\mathbf{j}} = \hat{\mathbf{j}}\cdot\hat{\mathbf{k}} = \hat{\mathbf{k}}\cdot\hat{\mathbf{i}} = 0$$

5. The vector (cross) product of two vectors **A** and **B** is a vector perpendicular to the plane formed by **A** and **B** and is given by

$$\mathbf{A}\times\mathbf{B} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix}$$
$$= (A_yB_z - A_zB_y)\hat{\mathbf{i}} + (A_zB_x - A_xB_z)\hat{\mathbf{j}} + (A_xB_y - A_yB_x)\hat{\mathbf{k}}$$
$$= (|\mathbf{A}|\,|\mathbf{B}|\sin\theta)\hat{\mathbf{I}}$$

where $\hat{\mathbf{I}}$ is a unit vector in a direction perpendicular to the plane of **A** and **B** and a sense determined by the advancement of the tip of a right-hand screw when it is rotated from **A** to **B** (the right-hand rule). Utilizing both the scalar and vector products, the angle θ between two vectors can be written as

$$\tan\theta = \frac{|\mathbf{A}\times\mathbf{B}|}{\mathbf{A}\cdot\mathbf{B}}$$

The vector product can give the component of a vector **A** in a direction perpendicular to another vector **B** as

$$|\mathbf{A}\times\hat{\mathbf{n}}_B| = |\mathbf{A}|\sin\theta$$

Other properties of the vector product are:

$$\mathbf{A}\times\mathbf{B} = -\mathbf{B}\times\mathbf{A}$$
$$|\mathbf{A}\times\mathbf{B}|^2 = |\mathbf{A}|^2|\mathbf{B}|^2\sin^2\theta$$
$$\mathbf{A}\times(\mathbf{B}+\mathbf{C}) = \mathbf{A}\times\mathbf{B}+\mathbf{A}\times\mathbf{C}$$
$$(K\mathbf{A})\times\mathbf{B} = \mathbf{A}\times(K\mathbf{B}) = K(\mathbf{A}\times\mathbf{B})$$
$$\hat{\mathbf{i}}\times\hat{\mathbf{i}} = \hat{\mathbf{j}}\times\hat{\mathbf{j}} = \hat{\mathbf{k}}\times\hat{\mathbf{k}} = \mathbf{0}$$
$$\hat{\mathbf{i}}\times\hat{\mathbf{j}} = \hat{\mathbf{k}}$$
$$\hat{\mathbf{j}}\times\hat{\mathbf{k}} = \hat{\mathbf{i}}$$
$$\hat{\mathbf{k}}\times\hat{\mathbf{i}} = \hat{\mathbf{j}}$$
$$\mathbf{A}\times(\mathbf{B}\times\mathbf{C}) = \mathbf{B}(\mathbf{C}\cdot\mathbf{A}) - \mathbf{C}(\mathbf{A}\cdot\mathbf{B}) \qquad \text{(vector triple product)}$$
$$\mathbf{A}\cdot(\mathbf{B}\times\mathbf{C}) = \mathbf{B}\cdot(\mathbf{C}\times\mathbf{A}) = \mathbf{C}\cdot(\mathbf{A}\times\mathbf{B})$$
$$= \begin{vmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{vmatrix} \qquad \text{(scalar triple product)}$$

6. Two vectors **A** and **B** are parallel if and only if

$$\hat{\mathbf{n}}_A\cdot\hat{\mathbf{n}}_B = 1 \qquad \text{or} \qquad |\hat{\mathbf{n}}_A\times\hat{\mathbf{n}}_B| = 0$$

7. Two vectors **A** and **B** are perpendicular if and only if

$$\hat{\mathbf{n}}_A\cdot\hat{\mathbf{n}}_B = 0 \qquad \text{or} \qquad |\hat{\mathbf{n}}_A\times\hat{\mathbf{n}}_B| = 1$$

The conditions for parallelism and perpendicularity are written as shown above to reflect the expected skew symmetry of the two properties. For calculation purposes, it might be useful to replace $|\hat{\mathbf{n}}_A\times\hat{\mathbf{n}}_B| = 0$ by $\hat{\mathbf{n}}_A\times\hat{\mathbf{n}}_B = \mathbf{0}$ or $\mathbf{A}\times\mathbf{B} = \mathbf{0}$ for parallel lines and replace $\hat{\mathbf{n}}_A\cdot\hat{\mathbf{n}}_B = 0$ by $\mathbf{A}\cdot\mathbf{B} = 0$ for perpendicular lines.

5.5.2 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. 5-14. 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:

$$\mathbf{P} = \mathbf{P}_1 + (\mathbf{P} - \mathbf{P}_1) \tag{5.9}$$

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

$$\mathbf{P} - \mathbf{P}_1 = u(\mathbf{P}_2 - \mathbf{P}_1) \tag{5.10}$$

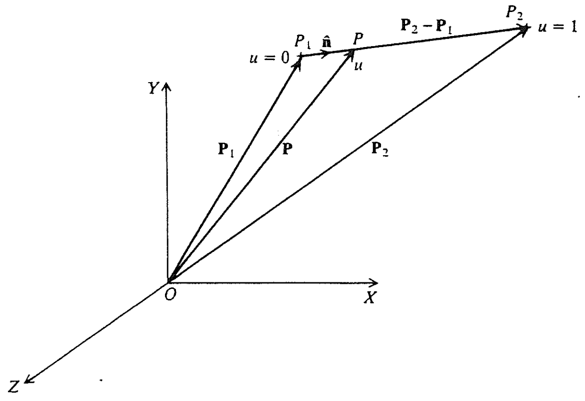


FIGURE 5-14
Line connecting two points P_1 and P_2.

Thus the equation of the line becomes

$$\mathbf{P} = \mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1), \qquad 0 \le u \le 1 \tag{5.11}$$

In scalar form, this equation can be written as

$$\left.\begin{aligned} x &= x_1 + u(x_2 - x_1) \\ y &= y_1 + u(y_2 - y_1) \\ z &= z_1 + u(z_2 - z_1) \end{aligned}\right\} \qquad 0 \le u \le 1 \tag{5.12}$$

Equation (5.11) 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. 5-15 shows. The coordinates of any point in the figure are obtained by substituting the corresponding u value in Eq. (5.11).

The tangent vector of the line is given by

$$\mathbf{P}' = \mathbf{P}_2 - \mathbf{P}_1 \tag{5.13}$$

or, in scalar form,

$$\begin{aligned} x' &= x_2 - x_1 \\ y' &= y_2 - y_1 \\ z' &= z_2 - z_1 \end{aligned} \tag{5.14}$$

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. (5.14).

The unit vector $\hat{\mathbf{n}}$ in the direction of the line (Fig. 5-14) is given by

$$\hat{\mathbf{n}} = \frac{\mathbf{P}_2 - \mathbf{P}_1}{L} \tag{5.15}$$

where L is the length of the line:

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

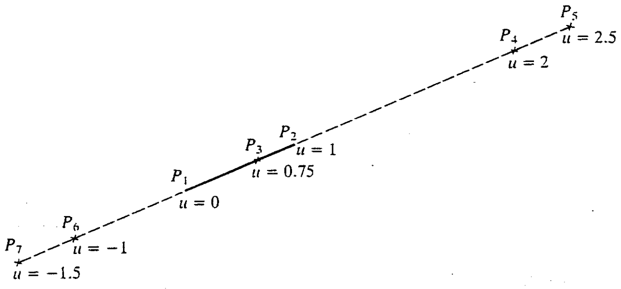


FIGURE 5-15
Locating points on an existing line.

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 (5.11) and (5.13) 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. (5.11) 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. 5-16). 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)

   $$\mathbf{P} = \mathbf{P}_1 + L\hat{\mathbf{n}}, \qquad -\infty \le L \le \infty \tag{5.17}$$

   and L is given by

   $$L = |\mathbf{P} - \mathbf{P}_1| \tag{5.18}$$

   L is the parameter in Eq. (5.17). 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. (5.17), 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.

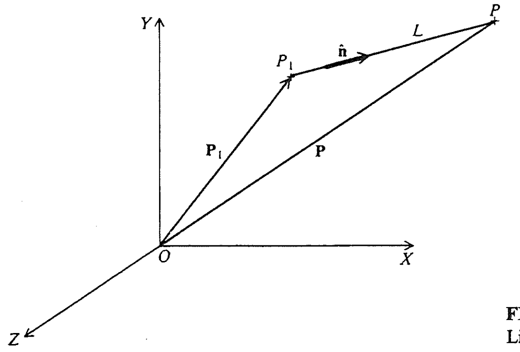


FIGURE 5-16
Line passing through P_1 in direction **n**.

Example 5.6. 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 5-17 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.

Assume that three points P_1, P_2, and P_3 are given. The vertical line passes through P_1 and ends at P_2 while the horizontal line passes through P_1 and ends at P_3. In general, the two lines cannot pass through points P_2 or P_3. Therefore, the ends are determined by projecting the points onto the lines as shown. Using Eq. (5.17), the line equations are

Vertical: $$\mathbf{P} = \mathbf{P}_1 + L\hat{\mathbf{n}}_1, \qquad 0 \le L \le L_1$$

or

$$x_W = x_{1W}$$
$$y_W = y_{1W} + L$$
$$z_W = z_{1W}$$

where

$$L_1 = y_{2W} - y_{1W}$$

and the endpoints are (x_{1W}, y_{1W}, z_{1W}) and $(x_{1W}, y_{1W} + L_1, z_{1W})$

Horizontal: $$\mathbf{P} = \mathbf{P}_1 + L\hat{\mathbf{n}}_2, \qquad 0 \le L \le L_2$$

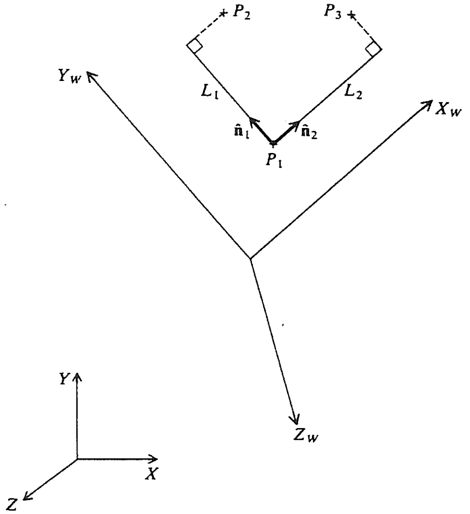


FIGURE 5-17
Horizontal and vertical lines.

or

$$x_W = x_{1W} + L$$
$$y_W = y_{1W}$$
$$z_W = z_{1W}$$

where

$$L_2 = x_{3W} - x_{1W}$$

and the endpoints are (x_{1W}, y_{1W}, z_{1W}) and $(x_{1W} + L_2, y_{1W}, z_{1W})$.

Example 5.7. Find the equation and endpoints of a line that passes through a point P_1, parallel to an existing line, and is trimmed by point P_2 as shown in Fig. 5-18.

Solution. To minimize confusion, the differentiation between the orientations of the WCS and the MCS is ignored and the position vectors of the various points are omitted from the figure. Assume that the existing line has the two endpoints P_3 and P_4, a length L_1, and a direction defined by the unit vector $\hat{\mathbf{n}}_1$. The new line has the same direction $\hat{\mathbf{n}}_1$, a length L_2, and endpoints P_1 and P_5. P_5 is the projection of P_2 onto the line.

The equation of the new line is found by substituting the proper vectors into Eq. (5.17). The unit vector $\hat{\mathbf{n}}_1$ is given by

$$\hat{\mathbf{n}}_1 = \frac{\mathbf{P}_4 - \mathbf{P}_3}{L_1}$$

where L_1 is a known value [Eq. (5.16)]. L_2 can be found from the following equation:

$$\begin{aligned} L_2 &= \hat{\mathbf{n}}_1 \cdot (\mathbf{P}_2 - \mathbf{P}_1) \\ &= \frac{\mathbf{P}_4 - \mathbf{P}_3}{L_1} \cdot (\mathbf{P}_2 - \mathbf{P}_1) \\ &= \frac{(x_4 - x_3)(x_2 - x_1) + (y_4 - y_3)(y_2 - y_1) + (z_4 - z_3)(z_2 - z_1)}{\sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2 + (z_4 - z_3)^2}} \end{aligned}$$

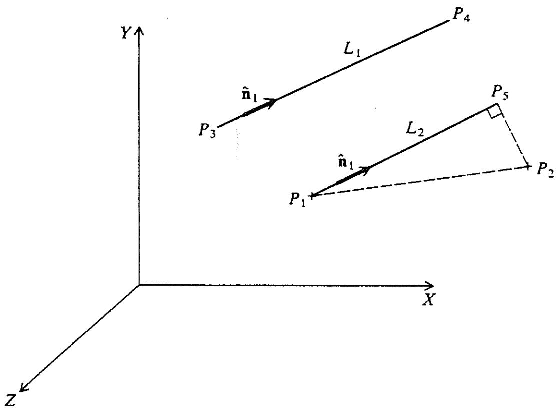


FIGURE 5-18
Line parallel to an existing line.

The equation of the new line becomes

$$\mathbf{P} = \mathbf{P}_1 + L\hat{\mathbf{n}}_1, \qquad 0 \le L \le L_2$$

and its two endpoints are $P_1(x_1, y_1, z_1)$ and $P_5(x_1 + L_2 n_{1x}, y_1 + L_2 n_{1y}, z_1 + L_2 n_{1z})$. As numerical examples, consider the simple two-dimensional case of constructing lines parallel to existing horizontal and vertical ones. Verify the above equations for the horizontal lines case $P_1(3, 2)$, $P_2(7, 4)$, $P_3(2, 3)$, and $P_4(5, 3)$. Repeat for the vertical lines case $P_1(8, 9)$, $P_2(9, 2)$, $P_3(5, 3)$, and $P_4(5, 9)$. Readers can also utilize the corresponding command to this example to construct the line on their CAD/CAM systems, verify the line to obtain its length and endpoints, and then compare with the results obtained here.

Example 5.8. Relate the following CAD/CAM commands to their mathematical foundations:

(*a*) The command that measures the angle between two intersecting lines.
(*b*) The command that finds the distance between a point and a line.

Solution

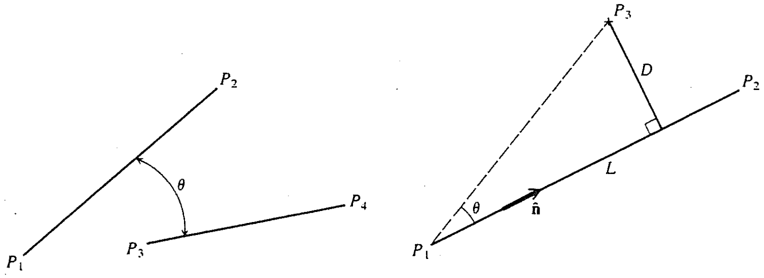
(*a*) If the endpoints of the two lines are P_1, P_2 and P_3, P_4 (Fig. 5-19*a*), the angle measurement command uses the equation

$$\cos\theta = \frac{(\mathbf{P}_2 - \mathbf{P}_1)\cdot(\mathbf{P}_4 - \mathbf{P}_3)}{|\mathbf{P}_2 - \mathbf{P}_1|\,|\mathbf{P}_4 - \mathbf{P}_3|}$$

$$= \frac{(x_2 - x_1)(x_4 - x_3) + (y_2 - y_1)(y_4 - y_3) + (z_2 - z_1)(z_4 - z_3)}{\sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2][(x_4 - x_3)^2 + (y_4 - y_3)^2 + (z_4 - z_3)^2]}}$$

(*b*) Figure 5-19*b* shows the distance D from point P_3 to the line whose endpoints are P_1 and P_2 and direction is $\hat{\mathbf{n}}$. Its length is L. D is given by

$$D = |(\mathbf{P}_3 - \mathbf{P}_1) \times \hat{\mathbf{n}}| = \left|(\mathbf{P}_3 - \mathbf{P}_1) \times \frac{(\mathbf{P}_2 - \mathbf{P}_1)}{|\mathbf{P}_2 - \mathbf{P}_1|}\right| = \left\|\begin{matrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \dfrac{x_2 - x_1}{L} & \dfrac{y_2 - y_1}{L} & \dfrac{z_2 - z_1}{L} \end{matrix}\right\|$$



(*a*) Angle measurement command

(*b*) Distance measurement command

FIGURE 5-19
Data related to line commands.

(where, on the furtherest right-hand side, the inner vertical bars indicate the determinant and the outer vertical bars indicate the magnitude of the resulting vector) or

$$D = \frac{1}{L}\,\mathrm{SQRT}\{[(y_3 - y_1)(z_2 - z_1) - (z_3 - z_1)(y_2 - y_1)]^2$$
$$+ [(z_3 - z_1)(x_2 - x_1) - (x_3 - x_1)(z_2 - z_1)]^2$$
$$+ [(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1)]^2\}$$

where SQRT is the square root. For a two-dimensional case, this equation reduces to

$$D = \frac{1}{L}\,[(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1)]$$

Example 5.9. Find the unit tangent vector in the direction of a line:

(*a*) Parallel to an existing line.
(*b*) Perpendicular to an existing line.

Solution. The conditions of parallelism or perpendicularity of two lines given in the vector algebra reviewed in Sec. 5.5.1 are useful if the vector equations of the two lines exist. If one equation is not available, which is mostly the case in practical problems, the conditions should be reduced to find the unit tangent vector of the missing line in terms of the existing one. Figure 5-20 shows the existing line as L_1 with a known unit tangent vector $\hat{\mathbf{n}}_1$. The unit tangent vector $\hat{\mathbf{n}}_2$ is to be found in terms of $\hat{\mathbf{n}}_1$.

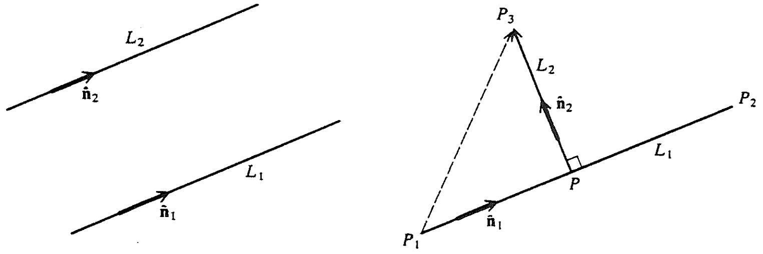
(*a*) For L_1 and L_2 to be parallel (Fig. 5-20*a*),

$$\hat{\mathbf{n}}_2 = \hat{\mathbf{n}}_1$$

or

$$[n_{2x} \quad n_{2y} \quad n_{2z}]^T = [n_{1x} \quad n_{1y} \quad n_{1z}]^T$$

This equation defines an infinite number of lines in an infinite number of planes in space. Additional geometric conditions are required to define a specific line. This equation is equivalent to the condition of the equality of the two slopes of



(*a*) Parallel lines

(*b*) Perpendicular lines

FIGURE 5-20
Unit tangent vectors of various lines.

the two lines in the two-dimensional case. Neglecting the Z component, the above equation gives the condition for this case as

$$[n_{2x} \quad n_{2y}]^T = [n_{1x} \quad n_{1y}]^T$$

or

$$\frac{n_{2y}}{n_{2x}} = \frac{n_{1y}}{n_{1x}}$$

or

$$m_2 = m_1$$

where m_1 and m_2 are the slopes of the lines.

(*b*) For L_1 and L_2 to be perpendicular (Fig. 5-20*b*),

$$\hat{\mathbf{n}}_2 \cdot \hat{\mathbf{n}}_1 = 0$$

or

$$n_{1x}n_{2x} + n_{1y}n_{2y} + n_{1z}n_{2z} = 0 \tag{5.19}$$

Additional equations are needed to solve for $\hat{\mathbf{n}}_2$. The following two equations can be written:

$$|\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2| = 1$$

or

$$(n_{1y}n_{2z} - n_{1z}n_{2y})^2 + (n_{1z}n_{2x} - n_{1x}n_{2z})^2 + (n_{1x}n_{2y} - n_{1y}n_{2x})^2 = 1 \tag{5.20}$$

and

$$\hat{\mathbf{n}}_2 \cdot \hat{\mathbf{n}}_2 = |\hat{\mathbf{n}}_2|^2$$

or

$$n_{2x}^2 + n_{2y}^2 + n_{2z}^2 = 1 \tag{5.21}$$

However, only two equations out of the above three are independent. For example, squaring Eq. (5.19) and adding it to Eq. (5.20) results in Eq. (5.21) if the identity $n_{1x}^2 + n_{1y}^2 + n_{1z}^2 = 1$ is used. Therefore, only two equations are available to solve for n_{2x}, n_{2y}, and n_{2z}, which implies that only two of these components can be obtained as functions of the third. This situation results from the fact that the perpendicular line L_2 to L_1 at point P shown in Fig. 5-20 is not unique. A plane perpendicular to L_1 at P defines the locus of L_2.

As a more defined case, assume that point P_3 is known and L_2 is the perpendicular line from P_3 to L_1. Using the above equations to solve for $\hat{\mathbf{n}}_2$ is usually cumbersome. Instead, utilizing the triangle P_1PP_3, the following equation can be written:

$$\mathbf{P}_1\mathbf{P}_3 = \mathbf{P}_1\mathbf{P} + \mathbf{P}\mathbf{P}_3$$

or

$$\mathbf{P}_3 - \mathbf{P}_1 = [(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_1]\hat{\mathbf{n}}_1 + D\hat{\mathbf{n}}_2 \tag{5.22}$$

where D is the perpendicular distance between P_3 and L_1 and is given in the previous example. Equation (5.22) gives $\hat{\mathbf{n}}_2$ as

$$\hat{\mathbf{n}}_2 = \frac{1}{D}\{\mathbf{P}_3 - \mathbf{P}_1 - [(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_1]\hat{\mathbf{n}}_1\}$$

In scalar form,

$$n_{2x} = \frac{1}{D}[(x_3 - x_1)(1 - n_{1x}^2) - (y_3 - y_1)n_{1x}n_{1y} - (z_3 - z_1)n_{1x}n_{1z}]$$

$$n_{2y} = \frac{1}{D}[(y_3 - y_1)(1 - n_{1y}^2) - (x_3 - x_1)n_{1x}n_{1y} - (z_3 - z_1)n_{1y}n_{1z}]$$

$$n_{2z} = \frac{1}{D}[(z_3 - z_1)(1 - n_{1z}^2) - (y_3 - y_1)n_{1y}n_{1z} - (x_3 - x_1)n_{1x}n_{1z}]$$

For the two-dimensional case, Eq. (5.22) becomes

$$x_3 - x_1 = [(x_3 - x_1)n_{1x} + (y_3 - y_1)n_{1y}]n_{1x} + Dn_{2x}$$

$$y_3 - y_1 = [(x_3 - x_1)n_{1x} + (y_3 - y_1)n_{1y}]n_{1y} + Dn_{2y}$$

Multiplying the first and second equations by n_{1x} and n_{1y} respectively and adding them, utilizing the identity $n_{1x}^2 + n_{1y}^2 = 1$, we get

$$D(n_{2x}n_{1x} + n_{2y}n_{1y}) = 0$$

or

$$\frac{n_{2y}}{n_{2x}} = -\frac{n_{1x}}{n_{1y}}$$

or

$$\frac{n_{2y}}{n_{2x}}\frac{n_{1y}}{n_{1x}} = -1$$

or

$$m_1m_2 = -1$$

which is a known result from two-dimensional analytic geometry. In the two-dimensional case $\hat{\mathbf{n}}_2$ can be chosen such that

$$n_{2x} = n_{1y} \qquad n_{2y} = -n_{1x}$$

or

$$n_{2x} = -n_{1y} \qquad n_{2y} = n_{1x}$$

Note: it is left to the reader to show that Eq. (5.22) can be reduced to the condition $\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2 = 0$.

Other useful line cases are assigned as problems at the end of the chapter.

5.5.3 Circles

Circles and circular arcs are among the most common entities used in wireframe modeling. Circles and circular arcs together with straight lines are sufficient to construct a large percentage of existing mechanical parts and components in practice. Besides other information, a circle database stores its radius and center as its essential geometric data. If the plane of the circle cannot be defined from the user input data as in the case of specifying a center and a radius, it is typically assumed by the software to be the XY plane of the current WCS at the time of construction. Regardless of the user input information to create a circle, such information is always converted into a radius and center by the software. This section presents some cases to show how such conversion is possible. Other cases are left to the reader as problems at the end of the chapter.

The basic parametric equation of a circle can be written as (refer to Fig. 5-21)

$$\left.\begin{aligned} x &= x_c + R\cos u \\ y &= y_c + R\sin u \\ z &= z_c \end{aligned}\right\} \quad 0 \le u \le 2\pi \tag{5.23}$$

assuming that the plane of the circle is the XY plane for simplicity. In this equation, the parameter u is the angle measured from the X axis to any point P on the circle. This parameter is used by commercial software to locate points at certain

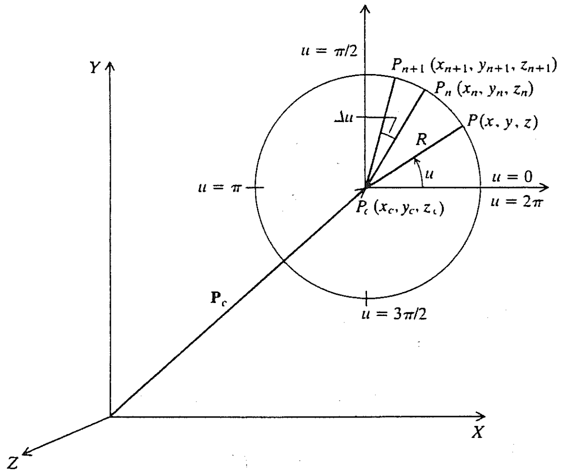


FIGURE 5-21
Circle defined by a radius and center.

angles on the circle for construction purposes. Some certain values of u are shown in Fig. 5-21.

For display purposes, Eq. (5.23) can be used to generate points on the circle circumference by incrementing u from 0 to 360. These points are in turn connected with line segments to display the circle. However, this is an inefficient way due to computing the trigonometric functions in the equation for each point. A less computational method is to write Eq. (5.23) in an incremental form. Assuming there is an increment Δu between two consecutive points $P(x_n, y_n, z_n)$ and $P(x_{n+1}, y_{n+1}, z_{n+1})$ on the circle circumference, the following recursive relationship can be written:

$$\begin{aligned} x_n &= x_c + R \cos u \\ y_n &= y_c + R \sin u \\ x_{n+1} &= x_c + R \cos (u + \Delta u) \\ y_{n+1} &= y_c + R \sin (u + \Delta u) \\ z_{n+1} &= z_n \end{aligned} \tag{5.24}$$

Expanding the x_{n+1} and y_{n+1} equation gives

$$\begin{aligned} x_{n+1} &= x_c + (x_n - x_c) \cos \Delta u - (y_n - y_c) \sin \Delta u \\ y_{n+1} &= y_c + (y_n - y_c) \cos \Delta u + (x_n - x_c) \sin \Delta u \\ z_{n+1} &= z_n \end{aligned} \tag{5.25}$$

Thus, the circle can start from an arbitrary point and successive points with equal spacing can be calculated recursively. Cos Δu and sin Δu have to be calculated

only once, which eliminates computation of trigonometric functions for each point. This algorithm is useful for hardware implementation to speed up the circle generation and display.

Circular arcs are considered a special case of circles. Therefore, all discussions covered here regarding circles can easily be extended to arcs. A circular arc equation can be written as

$$\left.\begin{aligned} x &= x_c + R \cos u \\ y &= y_c + R \sin u \\ z &= z_c \end{aligned}\right\} \quad u_s \le u \le u_e \tag{5.26}$$

where u_s and u_e are the starting and ending angles of the arc respectively. An arc database includes its center and radius, as a circle, as well as its starting and ending angles. Most user inputs offered by software to create arcs are similar to these offered to create circles. In fact, some software packages do not even offer arcs, in which case users have to create circles and then trim them using the proper trimming boundaries. As indicated from Eq. (5.26) and Fig. 5-21, the arc always connects its beginning and ending points in a counterclockwise direction. This rule is usually the default of most CAD/CAM packages when the user input is not sufficient to determine the arc position in space. For example, Fig. 5-22 shows the two possibilities to create an arc given two input points P_1 and P_2 to define its diameter. In this case, the arc is obviously half a circle and P_1 is always its starting point as it is input first by the user.

Following are some examples that show how various geometric data and constraints, which can be thought of as user inputs required by software packages, can be converted to a radius and center before its storage by the software in the corresponding circle database. They also show that three constraints (points and/or tangent vectors) are required to create a circle except for the obvious case of a center and radius. In all these examples, the reader can verify the resulting equations by comparing their results with those of an accessible CAD/CAM software package.

Example 5.19. Find the radius and the center of a circle whose diameter is given by two points.

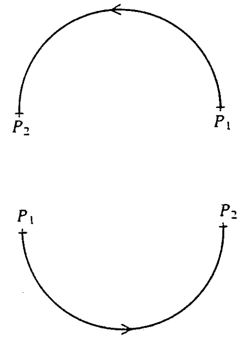


FIGURE 5-22
Arc creation following counterclockwise direction.

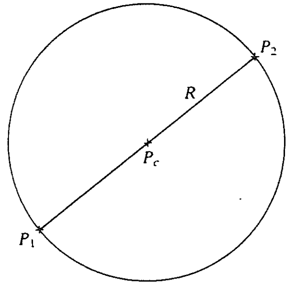


FIGURE 5-23
Circle defined by diameter P_1P_2.

Solution. Assume the circle diameter is given by the two points P_1 and P_2 as shown in Fig. 5-23. The circle radius and center are

$$R = \tfrac{1}{2}\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

$$\mathbf{P}_c = \tfrac{1}{2}(\mathbf{P}_1 + \mathbf{P}_2)$$

or

$$[x_c \quad y_c \quad z_c]^T = \begin{bmatrix} \dfrac{x_1 + x_2}{2} & \dfrac{y_1 + y_2}{2} & \dfrac{z_1 + z_2}{2} \end{bmatrix}^T$$

Example 5.11. Find the radius and the center of a circle passing through three points.

Solution. Figure 5-24 shows the three given points P_1, P_2, and P_3 that the circle must pass through. The circle radius and center are shown as P_c and R respectively. From analytic geometry, P_c is the intersection of the perpendicular lines to the chords P_1P_2, P_2P_3, and P_1P_3 from their midpoints P_4, P_5, and P_6 respectively. The unit vectors defining the directions of these chords in space are known and given by

$$\hat{\mathbf{n}}_1 = \frac{\mathbf{P}_2 - \mathbf{P}_1}{|\mathbf{P}_2 - \mathbf{P}_1|} \qquad \hat{\mathbf{n}}_2 = \frac{\mathbf{P}_3 - \mathbf{P}_2}{|\mathbf{P}_3 - \mathbf{P}_2|} \qquad \hat{\mathbf{n}}_3 = \frac{\mathbf{P}_1 - \mathbf{P}_3}{|\mathbf{P}_1 - \mathbf{P}_3|}$$

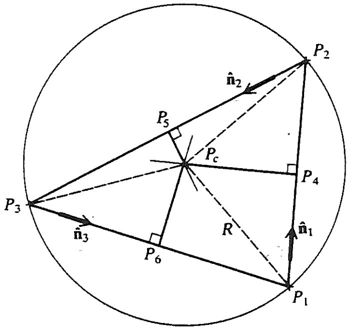


FIGURE 5-24
Circle passing through three points.

To find the center of the circle, P_c, the following three equations can be written:

$$(\mathbf{P}_c - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_1 = \frac{|\mathbf{P}_2 - \mathbf{P}_1|}{2}$$

$$(\mathbf{P}_c - \mathbf{P}_2) \cdot \hat{\mathbf{n}}_2 = \frac{|\mathbf{P}_3 - \mathbf{P}_2|}{2} \tag{5.27}$$

$$(\mathbf{P}_c - \mathbf{P}_3) \cdot \hat{\mathbf{n}}_3 = \frac{|\mathbf{P}_1 - \mathbf{P}_3|}{2}$$

Each of these vector equations implies that the component of a vector radius in the direction of any of the three chords is equal to half the chord length. These equations can be used to solve for P_c (x_c, y_c, z_c). Expanding and rearranging the equations in a matrix form yields

$$\begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ n_{3x} & n_{3y} & n_{3z} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{5.28}$$

where

$$b_1 = \frac{|\mathbf{P}_2 - \mathbf{P}_1|}{2} + (x_1 n_{1x} + y_1 n_{1y} + z_1 n_{1z})$$

$$b_2 = \frac{|\mathbf{P}_3 - \mathbf{P}_2|}{2} + (x_2 n_{2x} + y_2 n_{2y} + z_2 n_{2z})$$

$$b_3 = \frac{|\mathbf{P}_1 - \mathbf{P}_3|}{2} + (x_3 n_{3x} + y_3 n_{3y} + z_3 n_{3z})$$

The matrix equation (5.28) has the form $[A]\mathbf{P}_c = \mathbf{b}$. Therefore,

$$\mathbf{P}_c = [A]^{-1}\mathbf{b} = \frac{\text{Adj }([A])}{|A|}\mathbf{b}$$

where Adj ([A]) and $|A|$ are the adjoint matrix and determinant of [A] respectively. Adj ([A]) is the matrix $[C]^T$ where [C] is the matrix formed by the cofactors C_{ij} of the elements a_{ij} of [A]. The cofactor C_{ij} is given by

$$C_{ij} = (-1)^{i+j}M_{ij}$$

where M_{ij} is a unique scalar associated with the element a_{ij} and is defined as the determinant of the $(n - 1) \times (n - 1)$ matrix obtained from the $n \times n$ matrix [A] by crossing out the ith row and jth column. Thus

$$\mathbf{P}_c = \frac{[C]^T}{|A|}\mathbf{b}$$

and

$$|A| = n_{1x}(n_{2y}n_{3z} - n_{2z}n_{3y}) - n_{1y}(n_{2x}n_{3z} - n_{2z}n_{3x}) + n_{1z}(n_{2x}n_{3y} - n_{2y}n_{3x})$$

$$[C] = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

The elements of $[C]$ are given by

$$\begin{array}{lll} C_{11} = n_{2y}n_{3z} - n_{2z}n_{3y} & C_{12} = n_{2z}n_{3x} - n_{2x}n_{3z} & C_{13} = n_{2x}n_{3y} - n_{2y}n_{3x} \\ C_{21} = n_{1z}n_{3y} - n_{1y}n_{3z} & C_{22} = n_{1x}n_{3z} - n_{1z}n_{3x} & C_{23} = n_{1y}n_{3x} - n_{1x}n_{3y} \\ C_{31} = n_{1y}n_{2z} - n_{1z}n_{2y} & C_{32} = n_{1z}n_{2x} - n_{1x}n_{2z} & C_{33} = n_{1x}n_{2y} - n_{1y}n_{2x} \end{array}$$

The coordinates of the center can now be written as

$$x_c = \frac{1}{|A|}(C_{11}b_1 + C_{21}b_2 + C_{31}b_3)$$

$$y_c = \frac{1}{|A|}(C_{12}b_1 + C_{22}b_2 + C_{32}b_3)$$

$$z_c = \frac{1}{|A|}(C_{13}b_1 + C_{23}b_2 + C_{33}b_3)$$

The radius R is the distance between P_c and any of the three data points, that is,

$$R = |\mathbf{P}_c - \mathbf{P}_1| = |\mathbf{P}_c - \mathbf{P}_2| = |\mathbf{P}_c - \mathbf{P}_3|$$

or, for example,

$$R = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2 + (z_c - z_1)^2}$$

For the two-dimensional case, only two of the three relationships shown in Eq. (5.27) are sufficient to find the center $P_c(x_c, y_c)$. The third equation can be used to check the results. Using the first two relationships, the following matrix equations can be written:

$$\begin{bmatrix} n_{1x} & n_{1y} \\ n_{2x} & n_{2y} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where

$$b_1 = \frac{|\mathbf{P}_2 - \mathbf{P}_1|}{2} + (x_1 n_{1x} + y_1 n_{1y})$$

$$b_2 = \frac{|\mathbf{P}_3 - \mathbf{P}_2|}{2} + (x_2 n_{2x} + y_2 n_{2y})$$

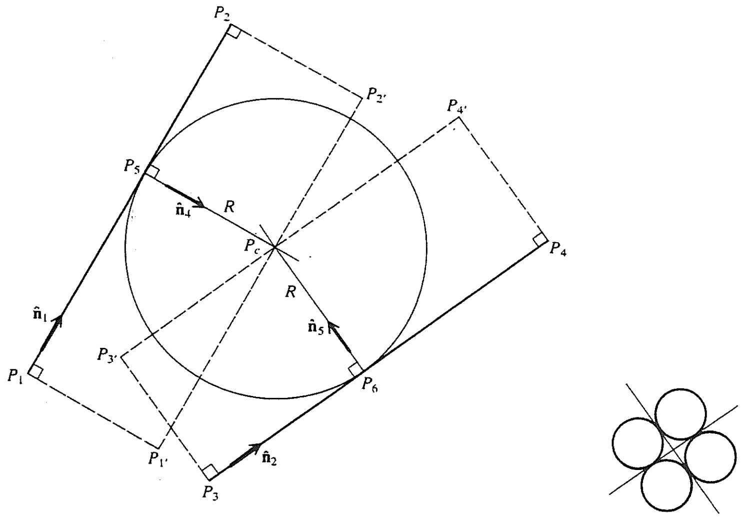
Similar to the three-dimensional case, the center is given by

$$x_c = \frac{n_{2y}b_1 - n_{1y}b_2}{n_{1x}n_{2y} - n_{1y}n_{2x}}$$

$$y_c = \frac{n_{1x}b_2 - n_{2x}b_1}{n_{1x}n_{2y} - n_{1y}n_{2x}}$$

Example 5.12. Find the center of a circle that is tangent to two known lines with a given radius.

Solution. This case is shown in Fig. 5-25. The two existing lines are defined by the point pairs (P_1, P_2) and (P_3, P_4). The unit vectors $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ define the directions of the lines in space. The center of the circle P_c is the intersection of the two normals to the two lines at the tangency points. The unit vectors $\hat{\mathbf{n}}_4$ and $\hat{\mathbf{n}}_5$ define the directions of these two normals in space.



(*a*) One solution

(*b*) Multiple solutions

FIGURE 5-25
Circle tangent to two lines with given radius R.

Conceptually, P_c can be found by finding the intersection of the two normals by solving their two vector equations. However, this route is cumbersome. Instead, P_c is found as the intersection of the two lines $P_{1'}P_{2'}$ and $P_{3'}P_{4'}$ that are parallel to P_1P_2 and P_3P_4 at distance R respectively. The following development is based on the observation that the two lines P_1P_2 and P_3P_4 define the plane of the circle and all the other lines and points shown in Fig. 5-25 lie in this plane.

The unit vectors can be defined as

$$\hat{\mathbf{n}}_1 = \frac{\mathbf{P}_2 - \mathbf{P}_1}{|\mathbf{P}_2 - \mathbf{P}_1|} \qquad \hat{\mathbf{n}}_2 = \frac{\mathbf{P}_4 - \mathbf{P}_3}{|\mathbf{P}_4 - \mathbf{P}_3|} \qquad \hat{\mathbf{n}}_3 = \frac{\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2}{|\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2|}$$

$$\hat{\mathbf{n}}_4 = \hat{\mathbf{n}}_3 \times \hat{\mathbf{n}}_1 \qquad \hat{\mathbf{n}}_5 = \hat{\mathbf{n}}_2 \times \hat{\mathbf{n}}_3$$

where $\hat{\mathbf{n}}_3$ is the unit vector perpendicular to the plane of the circle. The endpoints of the parallel lines are given by

$$\mathbf{P}_{1'} = \mathbf{P}_1 + R\hat{\mathbf{n}}_4 \qquad \mathbf{P}_{2'} = \mathbf{P}_2 + R\hat{\mathbf{n}}_4$$

and

$$\mathbf{P}_{3'} = \mathbf{P}_3 + R\hat{\mathbf{n}}_5 \qquad \mathbf{P}_{4'} = \mathbf{P}_4 + R\hat{\mathbf{n}}_5$$

Thus, the parametric vector equations of these parallel lines becomes

$$\mathbf{P} = \mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1) + R\hat{\mathbf{n}}_4 \tag{5.29}$$

$$\mathbf{P} = \mathbf{P}_3 + v(\mathbf{P}_4 - \mathbf{P}_3) + R\hat{\mathbf{n}}_5 \tag{5.30}$$

Therefore, the intersection point of the two lines is defined by the equation

$$\mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1) + R\hat{\mathbf{n}}_4 = \mathbf{P}_3 + v(\mathbf{P}_4 - \mathbf{P}_3) + R\hat{\mathbf{n}}_5 \qquad (5.31)$$

This vector equation can be solved for either u or v which, in turn, can be substituted into Eq. (5.29) or Eq. (5.30) to find P_c. If this equation is solved in scalar form, two components, say in the X and Y directions, give u and v. The third component, in the Z direction, can be used to check the computations involved.

To find u, take the scalar product of Eq. (5.31) with $\hat{\mathbf{n}}_5$. This gives

$$u = \frac{(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_5 + (1 - \hat{\mathbf{n}}_4 \cdot \hat{\mathbf{n}}_5)R}{(\mathbf{P}_2 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_5} \qquad (5.32)$$

Substituting this value in Eq. (5.29) gives

$$\mathbf{P}_c = \mathbf{P}_1 + \left[\frac{(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_5 + (1 - \hat{\mathbf{n}}_4 \cdot \hat{\mathbf{n}}_5)R}{(\mathbf{P}_2 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_5}\right](\mathbf{P}_2 - \mathbf{P}_1) + R\hat{\mathbf{n}}_4 \qquad (5.33)$$

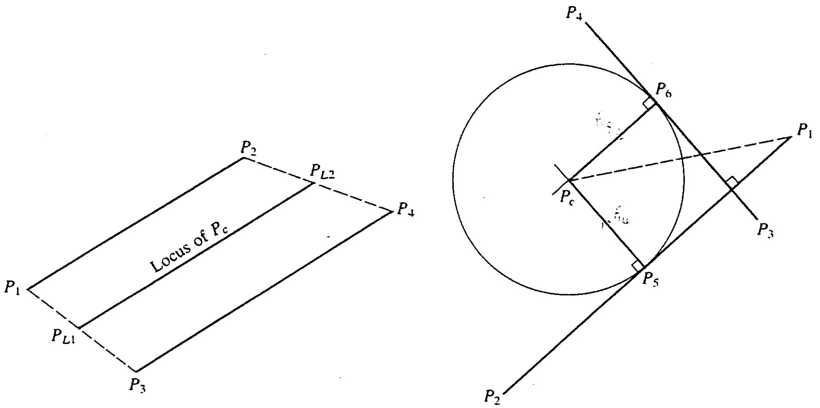
which can easily be written in scalar form to yield the coordinates x_c, y_c, and z_c of the centerpoint P_c. The two-dimensional case exhibits no special characteristics from the three-dimensional case.

The above development was not concerned with the existence of the four multiple solutions shown in Fig. 5-25*b* if the two known lines, and not their extensions, intersect. In such a case, the software package often follows a certain convention or requests the user to digitize the quadrant where the circle is to reside. One common convention is that the circle becomes tangent to the closest line segments chosen by the user while digitizing the two known lines to identify them. For all solutions, the above development is valid.

A point of interest here is the possible graphical solution to find the point P_c which is useful to verify the above results if the case of constructing a circle tangent to two lines is not provided by the software, but the other simpler case of a radius and center is. In terms of a typical CAD/CAM software package, the graphical solution consists of defining a WCS utilizing the endpoints of the two known lines. The XY plane of WCS becomes the plane of the lines. Using the parallel line command, the user can construct the two lines $P_{1'}P_{2'}$ and $P_{3'}P_{4'}$. The circle can now be constructed with the radius R and P_c as the intersection of these two parallel lines. Using the verify command the user can obtain the coordinates of P_c and compare with the results of Eq. (5.33).

Two special cases related to Eqs. (5.32) and (5.33) are discussed here. First, consider the case when the two known lines are parallel, as shown in Fig. 5-26*a*. In this case $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_2$, $\hat{\mathbf{n}}_4 = -\hat{\mathbf{n}}_5$, $\hat{\mathbf{n}}_4 \cdot \hat{\mathbf{n}}_5 = -1$, and $(\mathbf{P}_2 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_5 = 0$. Therefore, $u \to \infty$ and P_c is not defined. However, the locus of P_c is defined as the parallel line to the known lines at the middle distance between them, as shown in the figure. In this case the software can override the user input of R and replaces it by half the perpendicular distance between the two lines. Such a distance can be computed as shown in Sec. 5.5.2. The two endpoints of the locus of P_c are given by

$$\mathbf{P}_{L1} = \frac{\mathbf{P}_1 + \mathbf{P}_3}{2} \qquad \mathbf{P}_{L2} = \frac{\mathbf{P}_2 + \mathbf{P}_4}{2}$$



(*a*) Parallel lines

(*b*) Perpendicular lines

FIGURE 5-26
Circles tangent to parallel and perpendicular lines.

An infinite number of circles exists with centers on the locus. The software can either display a warning message to the user or choose point P_{L1} or P_{L2} as a default center.

The second case is shown in Fig. 5-26*b*, where the two known lines are perpendicular to each other. In this case $\hat{\mathbf{n}}_4 = \hat{\mathbf{n}}_2$, $\hat{\mathbf{n}}_5 = \hat{\mathbf{n}}_1$, and $\hat{\mathbf{n}}_4 \cdot \hat{\mathbf{n}}_5 = 0$. Equations (5.33) and (5.32) become

$$u = \frac{(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_1 + R}{|\mathbf{P}_2 - \mathbf{P}_1|}$$

$$\mathbf{P}_c = \mathbf{P}_1 + [(\mathbf{P}_3 - \mathbf{P}_1) \cdot \hat{\mathbf{n}}_1 + R]\hat{\mathbf{n}}_1 + R\hat{\mathbf{n}}_2$$

This equation for P_c is also obvious if we consider the triangle $P_1P_cP_5$ and write the vector equation $\mathbf{P}_1\mathbf{P}_c = \mathbf{P}_1\mathbf{P}_5 + \mathbf{P}_5\mathbf{P}_c$.

The above development can be extended to fillets connecting lines as follows. Once P_c is known from Eq. (5.33), the two points P_5 and P_6 (Fig. 5-25*a*) are given by

$$\mathbf{P}_5 = \mathbf{P}_c - R\hat{\mathbf{n}}_4$$

$$\mathbf{P}_6 = \mathbf{P}_c - R\hat{\mathbf{n}}_5$$

These two points define the beginning and the end of the fillet and can be used to construct the proper part of the circle that forms the fillet.

5.5.4 Ellipses

Mathematically the ellipse is a curve generated by a point moving in space such that at any position the sum of its distances from two fixed points (foci) is constant and equal to the major diameter. Each focus is located on the major axis of the ellipse at a distance from its center equal to $\sqrt{A^2 - B^2}$ (A and B are the major and minor radii). Circular holes and forms become ellipses when they are viewed obliquely relative to their planes.

The development of the parametric equation and other related characteristics of ellipses, elliptic arcs, and fillets are similar to those of circles, circular arcs, and fillets. However, four conditions (points and/or tangent vectors) are required to define the geometric shape of an ellipse as compared to three conditions to define a circle. The default plane of an ellipse, as in a circle, is the XY plane of the current WCS at the time of construction if the user input is not enough to define the ellipse plane, as in the case of inputing the center, half of the length of the major axis, and half of the length of the minor axis. The database of an ellipse usually stores user input as a centerpoint, half the length of the major axis, half the length of the minor axis, and other information (orientation, starting and ending angles, layer, font, name, color, etc.).

Figure 5-27 shows an ellipse with point P_c as the center and the lengths of half of the major and minor axes are A and B respectively. The parametric equation of an ellipse can be written as

$$\left.\begin{aligned} x &= x_c + A \cos u \\ y &= y_c + B \sin u \\ z &= z_c \end{aligned}\right\} \quad 0 \le u \le 2\pi \qquad (5.34)$$

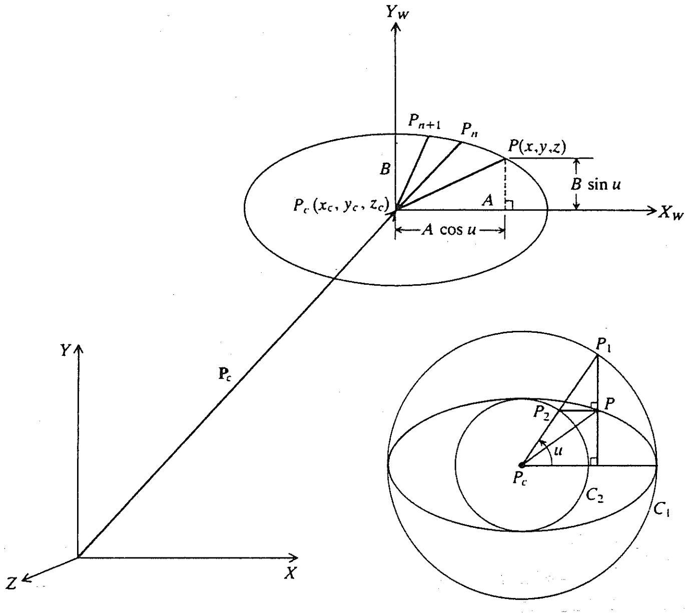


FIGURE 5-27
Ellipse defined by a center, major and minor axes.

assuming the plane of the ellipse is the XY plane. The parameter u is the angle as in the case of a circle. However, for a point P shown in the figure, it is not the angle between the line PP_c and the major axis of the ellipse. Instead, it is defined as shown. To find point P on the ellipse that corresponds to an angle u, the two concentric circles C_1 and C_2 are constructed with centers at P_c and radii of A and B respectively. A radial line is constructed at the angle u to intersect both circles at points P_1 and P_2 respectively. If a line parallel to the minor axis is drawn from P_1 and a line parallel to the major axis is drawn from P_2, the intersection of these two lines defines the point P.

Similar development as in the case of a circle results in the following recursive relationships which are useful for generating points on the ellipse for display purposes without excessive evaluations of trigonometric functions:

$$\begin{aligned} x_{n+1} &= x_c + (x_n - x_c) \cos \Delta u - \frac{A}{B}(y_n - y_c) \sin \Delta u \\ y_{n+1} &= y_c + (y_n - y_c) \cos \Delta u + \frac{A}{B}(x_n - x_c) \sin \Delta u \\ z_{n+1} &= z_n \end{aligned} \qquad (5.35)$$

If the ellipse major axis is inclined with an angle α relative to the X axis as shown in Fig. 5-28, the ellipse equation becomes

$$\left.\begin{aligned} x &= x_c + A \cos u \cos \alpha - B \sin u \sin \alpha \\ y &= y_c + A \cos u \sin \alpha + B \sin u \cos \alpha \\ z &= z_c \end{aligned}\right\} \quad 0 \le u \le 2\pi \qquad (5.36)$$

Equations (5.36) cannot be reduced to a recursive relationship similar to what is given by Eqs. (5.35). Instead these equations can be written as

$$\begin{aligned} x_{n+1} &= x_c + A \cos (u_n + \Delta u) \cos \alpha - B \sin (u_n + \Delta u) \sin \alpha \\ y_{n+1} &= y_c + A \cos (u_n + \Delta u) \sin \alpha + B \sin (u_n + \Delta u) \cos \alpha \\ z_{n+1} &= z_n \end{aligned} \qquad (5.37)$$

where $u_n = (n - 1)u$. The first point corresponds to $n = 0$ which lies at the end of the major axis. In addition, $\cos (u_n + \Delta u)$ and $\sin (u_n + \Delta u)$ are evaluated from the double-angle formulas for cosine and sine. If the calculations from the previous point for $\sin u_n$ and $\cos u_n$ are stored temporarily, $\cos (u_n + \Delta u)$ and $\sin (u_n + \Delta u)$ can be evaluated without calculating trigonometric functions for each point. Cos Δu and sin Δu need to be calculated only once. Therefore, computational savings similar to the circle case can be achieved.

Example 5.13. Find the center, the lengths of half the axes, and the orientation in space of an ellipse defined by:

(*a*) Its circumscribing rectangle.
(*b*) One of its internal rectangles.

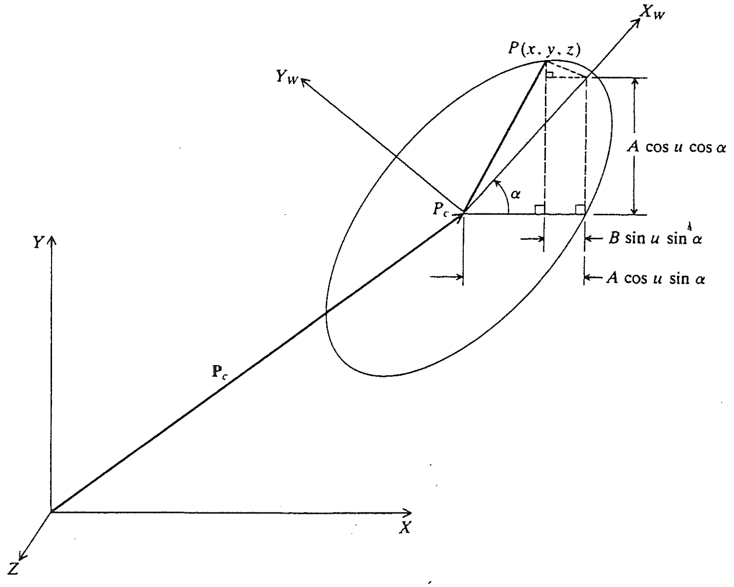


FIGURE 5-28
An inclined ellipse.

Solution. Both cases are equivalent to defining an ellipse by four points. The user interface can let the user input the three corner points of the rectangle or one corner point, the lengths of the rectangle sides, and an orientation.

(*a*) From Fig. 5-29*a*:

$$\mathbf{P}_c = \tfrac{1}{2}(\mathbf{P}_1 + \mathbf{P}_3) = \tfrac{1}{2}(\mathbf{P}_2 + \mathbf{P}_4)$$

$$\mathbf{P}_H = \tfrac{1}{2}(\mathbf{P}_2 + \mathbf{P}_3)$$

$$\mathbf{P}_v = \tfrac{1}{2}(\mathbf{P}_3 + \mathbf{P}_4)$$

$$A = |\mathbf{P}_c - \mathbf{P}_4| = \sqrt{(x_c - x_4)^2 + (y_c - y_4)^2 + (z_c - z_4)^2}$$

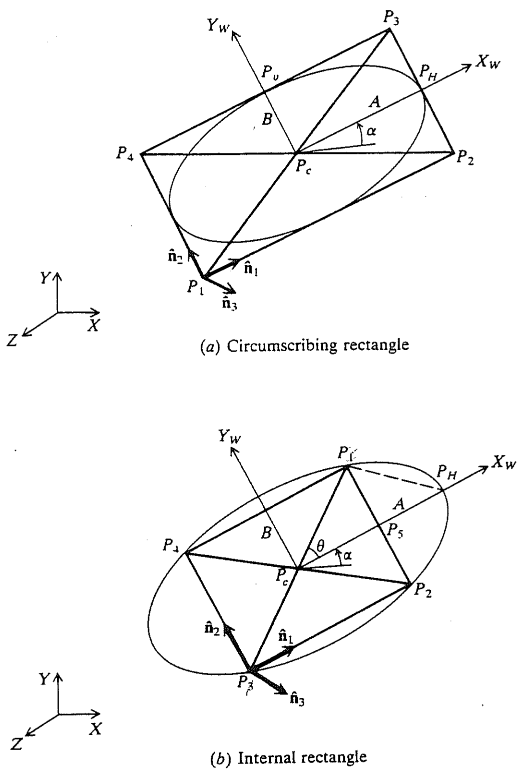
$$B = |\mathbf{P}_c - \mathbf{P}_v| = \sqrt{(x_c - x_v)^2 + (y_c - y_v)^2 + (z_c - z_v)^2}$$

The orientation of the ellipse in space can be defined by the unit vectors $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$ instead of the angle α. These vectors are given by

$$\hat{\mathbf{n}}_1 = \frac{\mathbf{P}_2 - \mathbf{P}_1}{|\mathbf{P}_2 - \mathbf{P}_1|} \qquad \hat{\mathbf{n}}_2 = \frac{\mathbf{P}_4 - \mathbf{P}_1}{|\mathbf{P}_4 - \mathbf{P}_1|} \qquad \hat{\mathbf{n}}_3 = \hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2$$

To display the ellipse, points on its circumference can be computed in the local (WCS) system $X_W Y_W Z_W$ utilizing the equation

$$\left.\begin{aligned} x_W &= A \cos u \\ y_W &= B \cos u \\ z_W &= 0 \end{aligned}\right\} \qquad 0 \le u \le 2\pi$$



(*a*) Circumscribing rectangle

(*b*) Internal rectangle

FIGURE 5-29
An ellipse defined by four points.

These points can then be transformed to the MCS utilizing Eq. (3.3) as follows:

$$[x \quad y \quad z \quad 1]^T = \left[\begin{array}{ccc:c} n_{1x} & n_{2x} & n_{3x} & x_c \\ n_{1y} & n_{2y} & n_{3y} & y_c \\ n_{1z} & n_{2z} & n_{3z} & z_c \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x_W \\ y_W \\ 0 \\ 1 \end{bmatrix}$$

The transformed points can be input to the graphics display driver to display the ellipse.

(*b*) P_c, $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$ can be calculated as in case (*a*). To find A and B, one can write

$$\mathbf{P}_5 = \tfrac{1}{2}(\mathbf{P}_2 + \mathbf{P}_3)$$

In reference to the WCS coordinate system, the x and y coordinates of point P_3 are $|\mathbf{P}_5 - \mathbf{P}_c|$ and $|\mathbf{P}_3 - \mathbf{P}_5|$ respectively. Substituting into the ellipse equation,

$$\frac{|\mathbf{P}_5 - \mathbf{P}_c|^2}{A^2} + \frac{|\mathbf{P}_3 - \mathbf{P}_5|^2}{B^2} = 1 \qquad (5.38)$$

To write another equation in A and B, consider the triangle $P_c P_3 P_H$ and write the law of cosines as

$$|\mathbf{P}_3 - \mathbf{P}_H|^2 = |\mathbf{P}_3 - \mathbf{P}_c|^2 + A^2 - 2A|\mathbf{P}_3 - \mathbf{P}_c| \cos \theta \tag{5.39}$$

The angle θ can be computed as the angle between the two vectors $(\mathbf{P}_5 - \mathbf{P}_c)$ and $(\mathbf{P}_3 - \mathbf{P}_c)$. To calculate the length $|\mathbf{P}_3 - \mathbf{P}_H|$, $\mathbf{P}_H$ which has the coordinates $(A, 0, 0)$ in the WCS system must be transformed to the MCS. Consequently its MCS coordinates become $(x_c + n_{1x}A,\ y_c + n_{1y}A,\ z_c + n_{1z}A)$. Substituting these coordinates into Eq. (5.39), a second-order equation in A results which can be solved for A. Then Eq. (5.38) can be used to solve for B. The remainder of the solution (finding the ellipse orientation and displaying it) is identical to case (*a*).

Example 5.14. Find the center, the lengths of half the axes, and the orientation of an ellipse given two of its conjugate diameters.

Solution. This is a case of defining an ellipse by four points that form two of its conjugate diameters. Figure 5-30 shows the two conjugate diameters and the relevant unit vectors. The center of the ellipse is given by

$$\mathbf{P}_c = \tfrac{1}{2}(\mathbf{P}_1 + \mathbf{P}_3) = \tfrac{1}{2}(\mathbf{P}_2 + \mathbf{P}_4)$$

The unit vectors $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$ are given by

$$\hat{\mathbf{n}}_1 = \frac{\mathbf{P}_3 - \mathbf{P}_1}{|\mathbf{P}_3 - \mathbf{P}_1|} \qquad \hat{\mathbf{n}}_2 = \frac{\mathbf{P}_4 - \mathbf{P}_2}{|\mathbf{P}_4 - \mathbf{P}_2|} \qquad \hat{\mathbf{n}}_3 = \frac{\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2}{|\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2|}$$

Utilizing the ellipse equation for points P_2 and P_3, the following two equations can be written:

$$\frac{(L_1 \cos \theta_1)^2}{A^2} + \frac{(L_1 \sin \theta_1)^2}{R^2} = 1 \tag{5.40}$$

$$\frac{(L_2 \cos \theta_2)^2}{A^2} + \frac{(L_2 \sin \theta_2)^2}{B^2} = 1 \tag{5.41}$$

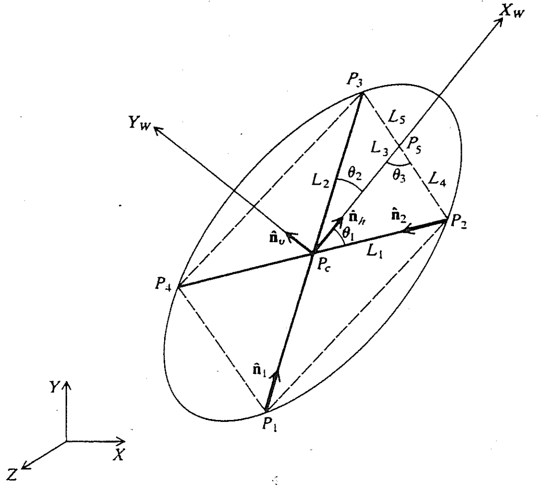


FIGURE 5-30
An ellipse defined by two conjugate diameters.

where

$$L_1 = |\mathbf{P}_2 - \mathbf{P}_c| \qquad L_2 = |\mathbf{P}_3 - \mathbf{P}_c|$$

The angles θ_1 and θ_2 can be related together as follows:

$$\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2 = \cos (\theta_1 + \theta_2)$$

or

$$\theta_1 + \theta_2 = K \tag{5.42}$$

where K is a known value. In addition, applying the law of sines to the triangles $P_c P_3 P_5$ and $P_c P_2 P_5$, we can write

$$\frac{L_1}{\sin \theta_3} = \frac{L_4}{\sin \theta_1} \tag{5.43}$$

$$\frac{L_2}{\sin (180 - \theta_3)} = \frac{L_5}{\sin \theta_2} \tag{5.44}$$

where

$$L_4 = |\mathbf{P}_2 - \mathbf{P}_5| \qquad L_5 = L_6 - L_4 \qquad L_6 = |\mathbf{P}_3 - \mathbf{P}_2|$$

Dividing Eq. (5.44) by (5.43) and using (5.42) gives

$$\frac{\sin \theta_1}{\sin (K - \theta_1)} = \frac{L_2 L_4}{L_1(L_6 - L_4)} \tag{5.45}$$

Applying the law of cosines to the same triangles gives

$$L_4^2 = L_1^2 + L_3^2 - 2L_1 L_3 \cos \theta_1 \tag{5.46}$$

$$(L_6 - L_4)^2 = L_2^2 + L_3^2 - 2L_2 L_3 \cos \theta_2 \tag{5.47}$$

Equations (5.40), (5.41), (5.42), (5.45), (5.46), and (5.47) form six equations to be solved for A, B, θ_1, θ_2, L_3, and L_4. Subtracting Eqs. (5.47) and (5.46) and using (5.42) gives

$$L_3 = \frac{L_2^2 - L_1^2 - L_6^2 + 2L_4 L_6}{2[L_2 \cos (K - \theta_1) - L_1 \cos \theta_1]} \tag{5.48}$$

If Eq. (5.48) is substituted into (5.46) and the resulting L_4 is substituted into (5.45), a nonlinear equation in θ_1 results which can be solved for θ_1. Consequently θ_2 can be found from Eq. (5.42). Equations (5.40) and (5.41) can therefore be solved for A and B.

The orientation of the ellipse can be found by determining the unit vectors $\hat{\mathbf{n}}_h$ and $\hat{\mathbf{n}}_v$ that define the directions of the major and the minor axes respectively. To find $\hat{\mathbf{n}}_h$, the following three equations can be written:

$$\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_h = \cos \theta_2$$
$$\hat{\mathbf{n}}_2 \cdot \hat{\mathbf{n}}_h = \cos \theta_1$$
$$\hat{\mathbf{n}}_3 \cdot \hat{\mathbf{n}}_h = 0$$

These equations can be solved for the components n_{hx}, n_{hy}, and n_{hz} using the matrix approach utilized in Example 5.11. The unit vector $\hat{\mathbf{n}}_v$ can be found as

$$\hat{\mathbf{n}}_v = \hat{\mathbf{n}}_3 \times \hat{\mathbf{n}}_h$$

With $\hat{\mathbf{n}}_h$, $\hat{\mathbf{n}}_v$, and $\hat{\mathbf{n}}_3$ known, the orientation of the ellipse is completely defined in space and transformation of points on the ellipse from the WCS system to the MCS can be performed for display and plotting purposes.

Note: case (*b*) of Example 5.13 is a special case of this example.

Example 5.15. Find the tangent to an ellipse from a given point P_1 outside the ellipse.

Solution. Two tangents can be drawn to the ellipse from P_1 as shown in Fig. 5-31. Assume the tangency point is P_T. First, transform P_1 from the MCS to the ellipse local WCS system using the equation

$$\mathbf{P}_1 = [T]\mathbf{P}_{1W} \tag{5.49}$$

The transformation matrix $[T]$ is known because the orientation of the ellipse is known. P_{1W} holds the local coordinates of P_1 which should be $[x_{1W} \quad y_{1W} \quad 0]$. Therefore:

$$\mathbf{P}_{1W} = [T]^{-1}\mathbf{P}_1$$

The tangent vector to the ellipse is given by

$$\mathbf{P}' = [-A \sin u \quad B \cos u \quad 0]^T$$

At point P_T, this vector becomes

$$\mathbf{P}' = [-A \sin u_T \quad B \cos u_T \quad 0]^T$$

and the slope of the tangent is given by

$$S = -\frac{B \cos u_T}{A \sin u_T}$$

This slope S can also be found using the two points P_1 and P_T as

$$S = \frac{y_{1W} - B \sin u_T}{x_{1W} - A \cos u_T} \tag{5.50}$$

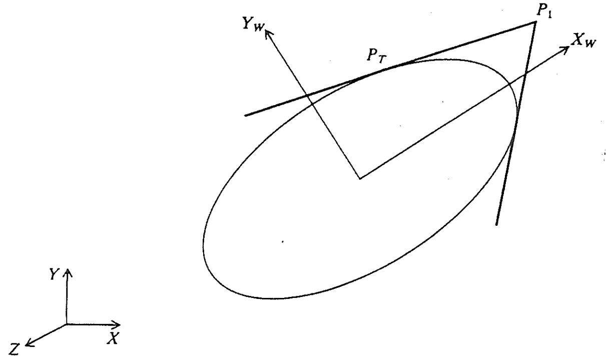


FIGURE 5-31
Tangent to an ellipse from an outside point.

Therefore,

$$\frac{y_{1W} - B \sin u_T}{x_{1W} - A \cos u_T} = -\frac{B \cos u_T}{A \sin u_T}$$

which gives

$$x_1 B \cos u_T + y_1 A \sin u_T = AB$$

or

$$K_1 \cos u_T + K_2 \sin u_T = AB$$

where $K_1 = x_1 B$ and $K_2 = y_1 A$. Define an angle γ such that $\gamma = \tan^{-1}(K_1/K_2)$. Thus, the above equation can be rewritten as

$$\sin(\gamma + u_T) = \frac{AB}{\sqrt{K_1^2 + K_2^2}}$$

or

$$u_T = \sin^{-1}\left(\frac{AB}{\sqrt{K_1^2 + K_2^2}}\right) - \gamma$$

The arcsine function gives two angles that result in two tangents.

Once u_T is known, the local coordinates of P_T become

$$\mathbf{P}_{TW} = [x_{TW} \quad y_{TW} \quad 0]^T = [A \cos u_T \quad B \sin u_T \quad 0]^T$$

$\mathbf{P}_{TW}$ can be substituted in Eq. (5.49) to obtain $\mathbf{P}_T$. Therefore, the tangent is defined and stored in the database by the two endpoints P_1 and P_T. In practice, the CAD/CAM software can ask the user to digitize close to the desired tangent so that the other one is eliminated.

5.5.5 Parabolas

The parabola is defined mathematically as a curve generated by a point that moves such that its distance from a fixed point (the focus $\mathbf{P}_F$) is always equal to its distance to a fixed line (the directrix) as shown in Fig. 5-32. The vertex P_v is the intersection point of the parabola with its axis of symmetry. It is located midway between the directrix and the focus. The focus lies on the axis of symmetry. Useful applications of the parabolic curve in engineering design include its use in parabolic sound and light reflectors, radar antennas, and in bridge arches.

Three conditions are required to define a parabola, a parabolic curve, or a parabolic arc. The default plane of a parabola is the XY plane of the current WCS at the time of construction. The database of a parabola usually stores the coordinates of its vertex, distances y_{HW} and y_{LW}, that define its endpoints as shown in Fig. 5-32, the distance A between the focus and the vertex (the focal distance), and the orientation angle α. Unlike the ellipse, the parabola is not a closed curve. Thus, the two endpoints determine the amount of the parabola to be displayed.

Assuming the local coordinate system of the parabola as shown in Fig. 5-32, its parametric equation can be written as

$$\left.\begin{aligned} x &= x_v + Au^2 \\ y &= y_v + 2Au \\ z &= z_v \end{aligned}\right\} \quad 0 \le u \le \infty \tag{5.51}$$

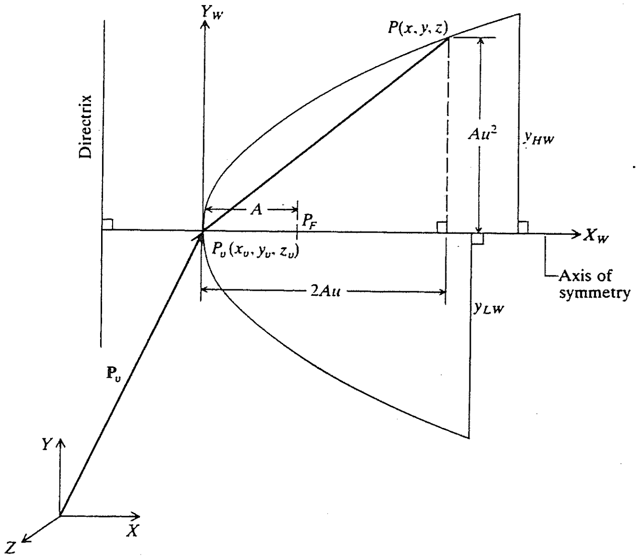


FIGURE 5-32
Basic geometry of a parabola.

If the range of the y coordinate is limited to y_{HW} and y_{LW} for positive and negative values respectively, the corresponding u values become

$$u_H = \frac{y_{HW}}{2A} \qquad\qquad (5.52)$$
$$u_L = \frac{y_{LW}}{2A}$$

The recursive relationships to generate points on the parabola are obtained by substituting $u_n + \Delta u$ for points $n + 1$. This gives

$$\begin{aligned} x_{n+1} &= x_n + (y_n - y_v)\,\Delta u + A(\Delta u)^2 \\ y_{n+1} &= y_n + 2A\,\Delta u \qquad\qquad (5.53) \\ z_{n+1} &= z_n \end{aligned}$$

If the parabolic axis of symmetry is inclined with an angle α as shown in Fig. 5-33, its equation becomes

$$\begin{aligned} x &= x_v + Au^2 \cos\alpha - 2Au \sin\alpha \\ y &= y_v + Au^2 \sin\alpha + 2Au \cos\alpha \qquad\qquad (5.54) \\ z &= z_v \end{aligned}$$

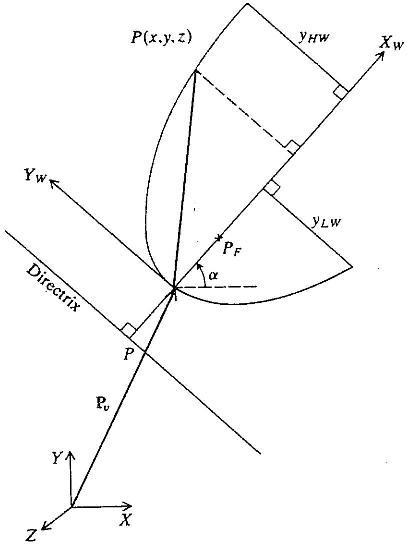


FIGURE 5-33
An inclined parabola.

and the recursive relationships reduce to

$$\begin{aligned} x_{n+1} &= x_n \cos\alpha + (1 - \cos\alpha)x_v + (\Delta u \cos\alpha - \sin\alpha)(y_n - y_v) \\ &\quad + A\,\Delta u(\Delta u \cos\alpha - 2\sin\alpha) \\ y_{n+1} &= (\cos\alpha + \Delta u \sin\alpha)y_n + (1 - \cos\alpha - \Delta u \sin\alpha)y_v \qquad\qquad (5.55) \\ &\quad + (x_n - x_v)\sin\alpha + A\,\Delta u(\Delta u \sin\alpha + 2\cos\alpha) \\ z_{n+1} &= z_n \end{aligned}$$

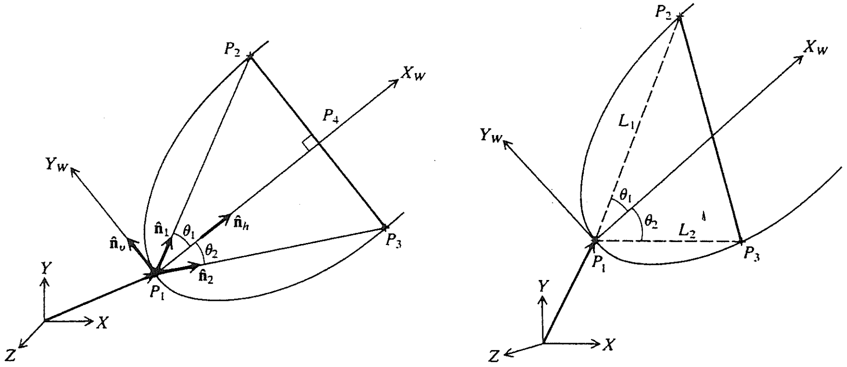
An alternative to Eqs. (5.55) would be to use Eqs. (5.53) to generate the points in the WCS local coordinate system of the parabola and then utilize Eq. (3.3) to transform these points to the MCS before displaying or plotting them. If the $X_W Y_W$ and the XY planes coincide, the transformation matrix becomes

$$[T] = \left[\begin{array}{ccc|c} \cos\alpha & \sin\alpha & 0 & x_v \\ -\sin\alpha & \cos\alpha & 0 & y_v \\ 0 & 0 & 1 & z_v \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad\qquad (5.56)$$

If the two planes are different, then the unit vectors $\hat{\mathbf{n}}_{xW}$, $\hat{\mathbf{n}}_{yW}$, and $\hat{\mathbf{n}}_{zW}$ that define the orientation of the parabolic local coordinate system relative to the MCS must be calculated before using Eq. (3.3).

Example 5.16. Find the focal distance and the orientation in space of a parabola that passes through three points, one of which is the vertex.

Solution. Figure 5-34 shows two cases in which point P_1 is the vertex. In Fig. 5-34*a*, the other two points P_2 and P_3 define a line perpendicular to the X_W axis of the



(*a*) $\mathbf{P}_2 - \mathbf{P}_3$ is perpendicular to the X_W axis

(*b*) General case

FIGURE 5-34
A parabola passing through three points.

parabola while Fig. 5-34*b* shows the general case. The solution for the first case can be found as follows. The angles θ_1 and θ_2 in this case are equal. Thus,

$$\mathbf{P}_4 = \frac{\mathbf{P}_2 + \mathbf{P}_3}{2}$$

The angle θ_1 can be found from

$$\tan\theta_1 = \frac{|\mathbf{P}_2 - \mathbf{P}_4|}{|\mathbf{P}_4 - \mathbf{P}_1|}$$

Using the parabolic equation $y_W^2 = 4Ax_W$, the focal distance A can be written as

$$A = \frac{|\mathbf{P}_2 - \mathbf{P}_4|^2}{4|\mathbf{P}_4 - \mathbf{P}_1|}$$

The orientation of the parabola is determined by the unit vectors $\hat{\mathbf{n}}_h$ and $\hat{\mathbf{n}}_v$ along the X_W and Y_W axes and the vector $\hat{\mathbf{n}}_3$ which is perpendicular to its plane. These vectors are given as

$$\hat{\mathbf{n}}_h = \frac{\hat{\mathbf{n}}_1 + \hat{\mathbf{n}}_2}{|\hat{\mathbf{n}}_1 + \hat{\mathbf{n}}_2|}$$

$$\hat{\mathbf{n}}_3 = \frac{\hat{\mathbf{n}}_2 \times \hat{\mathbf{n}}_1}{|\hat{\mathbf{n}}_2 \times \hat{\mathbf{n}}_1|} = \frac{\hat{\mathbf{n}}_2 \times \hat{\mathbf{n}}_1}{\sin(\theta_1 + \theta_2)}$$

$$\hat{\mathbf{n}}_v = \hat{\mathbf{n}}_3 \times \hat{\mathbf{n}}_h$$

where the unit vectors $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ can easily be computed from the given points.

Unlike the above case, the angles θ_1 and θ_2 are not equal for the second case (Fig. 5-34*b*). Applying the parabolic equations to points P_2 and P_3, we can write respectively

$$L_1 \sin^2\theta_1 = 4A\cos\theta_1 \tag{5.57}$$

$$L_2 \sin^2\theta_2 = 4A\cos\theta_2 \tag{5.58}$$

where $L_1 = |\mathbf{P}_2 - \mathbf{P}_1|$ and $L_2 = |\mathbf{P}_3 - \mathbf{P}_1|$. In addition,

$$\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2 = \cos(\theta_1 + \theta_2)$$

or

$$\theta_1 + \theta_2 = K \tag{5.59}$$

where

$$K = \cos^{-1}(\mathbf{n}_1 \cdot \hat{\mathbf{n}}_2)$$

The solution of Eqs. (5.57), (5.58), and (5.59) gives A, θ_1, and θ_2. If we divide Eq. (5.57) by (5.58) and use (5.59), we obtain

$$\frac{L_1}{L_2}\left[\frac{\sin\theta_1}{\sin(K - \theta_1)}\right]^2 = \frac{\cos\theta_1}{\cos(K - \theta_1)}$$

which is a nonlinear equation in θ_1. One way to solve it would be to plot both the left- and right-hand sides as functions of θ_1 $(0 \le \theta_1 \le 90)$ and find the intersection point using the intersection modifier provided by the CAD/CAM software. Once θ_1 is found, Eq. (5.57) can be solved for A and (5.59) solved for θ_2. To find the orientation of the parabola, write

$$\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_h = \cos\theta_1$$

$$\hat{\mathbf{n}}_2 \cdot \hat{\mathbf{n}}_h = \cos\theta_2$$

$$\hat{\mathbf{n}}_3 \cdot \hat{\mathbf{n}}_h = 0$$

which can be solved for the components of $\hat{\mathbf{n}}_h$. Then

$$\hat{\mathbf{n}}_v = \hat{\mathbf{n}}_3 \times \hat{\mathbf{n}}_h$$

5.5.6 Hyperbolas

A hyperbola is described mathematically as a curve generated by a point moving such that at any position the difference of its distances from the fixed points (foci) F and F' is a constant and equal to the transverse axis of the hyperbola. Figure 5-35 shows the geometry of a hyperbola.

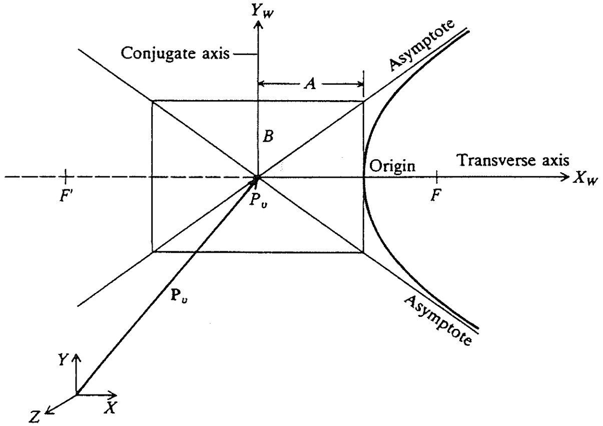


FIGURE 5-35
Hyperbola geometry.

The parametric equation of a hyperbola is given by

$$
\begin{aligned}
x &= x_v + A \cosh u \\
y &= y_v + B \sinh u \qquad (5.60) \\
z &= z_v
\end{aligned}
$$

This equation is based on the nonparametric implicit equation of the hyperbola which can be written as

$$\frac{(x - x_v)^2}{A^2} - \frac{(y - y_v)^2}{B^2} = 1 \qquad (5.61)$$

by utilizing the identity $\cosh^2 u - \sinh^2 u = 1$.

Similar to the ellipse developments, equations of an inclined hyperbola and recursive relationships can be derived. Also, the examples covered in the ellipse section can be extended to hyperbolas.

5.5.7 Conics

Conic curves or conic sections form the most general form of quadratic curves. Lines, circles, ellipses, parabolas, and hyperbolas covered in the previous sections are all special forms of conic curves. They all can be generated when a right circular cone of revolution is cut by planes at different angles relative to the cone axis—thus the derivation of the name conics. Straight lines result from intersecting a cone with a plane parallel to its axis and passing through its vertex. Circles result when the cone is sectioned by a plane perpendicular to its axis while ellipses, parabolas, and hyperbolas are generated when the plane is inclined to the axis by various angles.

The general implicit nonparametric quadratic equation that describes a planar conic curve has five coefficients if the coefficient of the term x^2 is made equal to one; that is, normalize the equation by dividing it by this coefficient if it is not one. Thus, five conditions are required to completely define a conic curve. As presented in the previous sections, these reduce to two conditions to define lines, three for circles and parabolas, and four for ellipses and hyperbolas.

The conic parametric equation can be developed if five conditions are specified. Two cases are discussed here: specifying five points or three points and two tangent vectors. The development is based on the observation that a quadratic equation can be written as the product of two linear equations. Figure 5-36 shows a conic curve passing through points P_1 to P_5. Define the two pairs of lines (L_1, L_2) and (L_3, L_4) shown in the figure. Their equations are

$$L_1 = 0 \qquad L_2 = 0 \qquad L_3 = 0 \qquad L_4 = 0 \qquad (5.62)$$

The four intersection points of these pairs are given by points P_1 to P_4. Let us define the two conics:

$$L_1 L_2 = 0 \qquad L_3 L_4 = 0 \qquad (5.63)$$

Each one of these conics passes through points P_1 to P_4 but not necessarily through point P_5. However, any linear combination of these two conics represents another conic (since it is quadratic) which passes through their intersection points P_1 to P_4. For example, the equation

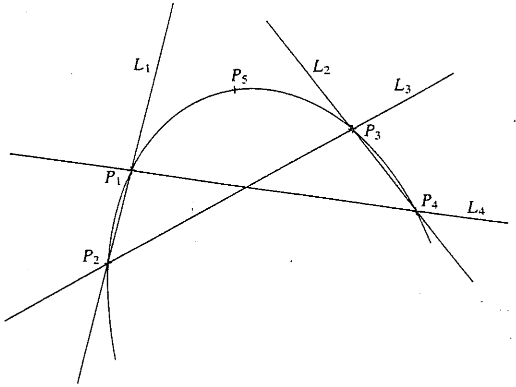


FIGURE 5-36
A conic curve defined by five points.

$$aL_1L_2 + bL_3L_4 = 0 \qquad (5.64)$$

represents such a conic. For the conic to pass through point P_5 its coordinates are substituted into Eq. (5.64) to find the ratio b/a. A more convenient form of Eq. (5.64) is

$$(1 - u)L_1L_2 + uL_3L_4 = 0, \qquad 0 \le u \le 1 \qquad (5.65)$$

Equation (5.65) gives the parametric equation of a conic curve with the parameter u. Changing the value of u results in a family (or pencil) of conics, two of which are $L_1L_2 = 0$ $(u = 0)$ and $L_3L_4 = 0$ $(u = 1)$. To use Eq. (5.65), four data points are used to find the equations for the lines L_1 to L_4 and the fifth point is used to find the u value.

The case of a conic defined by three points and two tangent vectors is considered as an adaptation of the above case. If we make lines L_1 and L_2 tangent to the conic curve, points P_1 and P_2 become one point (the tangent point) and P_3 and P_4 become another point. Consequently, the two lines L_3 and L_4 are merged into one line. Figure 5-37 shows the conic geometry in this case. The definition of this conic curve is equivalent to a four-point definition: the two points of tangency P_1 and P_2, the intersection point P_4 of the two tangents L_1 and L_2, and a fourth point P_3, known as the shoulder point. P_3 must always be chosen inside the triangle $P_1P_2P_4$ to ensure the continuity of the conic curve segment that lies inside the triangle between P_1 and P_2. Equation (5.65) is then reduced for this case to

$$(1 - u)L_1L_2 + uL_3^2 = 0, \qquad 0 \le u \le 1 \qquad (5.66)$$

Similar to the first case, points P_1, P_2, and P_4 determine the equations for the lines L_1 to L_3 and the point P_3 is used to find the u value.

The choice of the position of P_3 determines the type of resulting conic curve. If P_3 is the midpoint of the line joining the midpoints of the tangents L_1

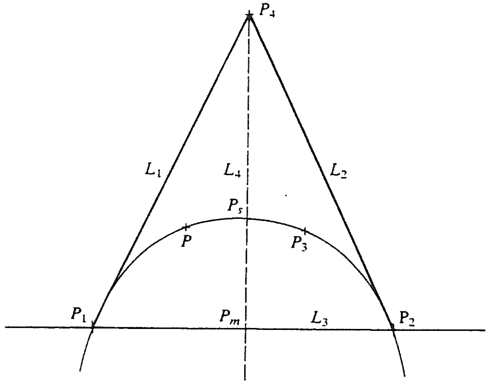


FIGURE 5-37
A conic curve defined by three points and two tangent vectors.

and L_2, then the conic is a parabola. P_3 becomes the vertex of the parabola and the line connecting P_3 and P_4 becomes its axis of symmetry. It is also obvious that the distance between P_3 and P_4 is equal to the distance between P_3 and the intersection point of the axis of symmetry and L_3 (a known characteristic of parabolas). If P_3 lies inside the parabola and line L_3, the resulting conic is an ellipse. If it is outside the parabola, a hyperbolic curve results. To check which type of conic curve results from a given input data, let P_m be the midpoint of the line L_3 and let the line L_4 intersect the conic curve at the point P_s. The parametric equation of line L_4 is $\mathbf{P} = \mathbf{P}_m + u(\mathbf{P}_4 - \mathbf{P}_m)$. Let the parameter u take the value s at P_s. Then $(\mathbf{P}_s - \mathbf{P}_m) = s(\mathbf{P}_4 - \mathbf{P}_m)$. Therefore, the conic curve is parabolic if $s = \frac{1}{2}$, elliptic if $s < \frac{1}{2}$, and hyperbolic if $s > \frac{1}{2}$. A further test is necessary to determine whether an eliptic curve is circular. If $|\mathbf{P}_4 - \mathbf{P}_1| = |\mathbf{P}_2 - \mathbf{P}_4|$ and

$$\frac{s^2}{1 - s^2} = \frac{|\mathbf{P}_2 - \mathbf{P}_1|^2}{4|\mathbf{P}_4 - \mathbf{P}_1|\,|\mathbf{P}_4 - \mathbf{P}_2|} \tag{5.67}$$

then the arc is circular.

Example 5.17. Find the equation of a conic curve defined by five points P_1 to P_5.

Solution. Equation (5.65) must be reduced further to be able to utilize it to generate points on the conic curve for display purposes. The approach taken in this example is to create a local (WCS) coordinate system $(X_W\, Y_W\, Z_W)$ in the plane of the conic curve (Fig. 5-38), write Eq. (5.65) in this system, generate points, and lastly transform these points to the global MCS (XYZ). The input point P_2 is taken as the origin of the local system and the vector $(\mathbf{P}_4 - \mathbf{P}_2)$ as its X_W axis. To define the system, the unit vectors along the axes are calculated as

$$\begin{aligned} \hat{\mathbf{n}}_1 &= \frac{\mathbf{P}_4 - \mathbf{P}_2}{|\mathbf{P}_4 - \mathbf{P}_2|} \qquad \hat{\mathbf{n}}_4 = \frac{\mathbf{P}_1 - \mathbf{P}_2}{|\mathbf{P}_1 - \mathbf{P}_2|} \\ \hat{\mathbf{n}}_3 &= \frac{\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_4}{|\hat{\mathbf{n}}_1 \times \mathbf{n}_4|} \\ \hat{\mathbf{n}}_2 &= \hat{\mathbf{n}}_3 \times \hat{\mathbf{n}}_1 \end{aligned} \tag{5.68}$$

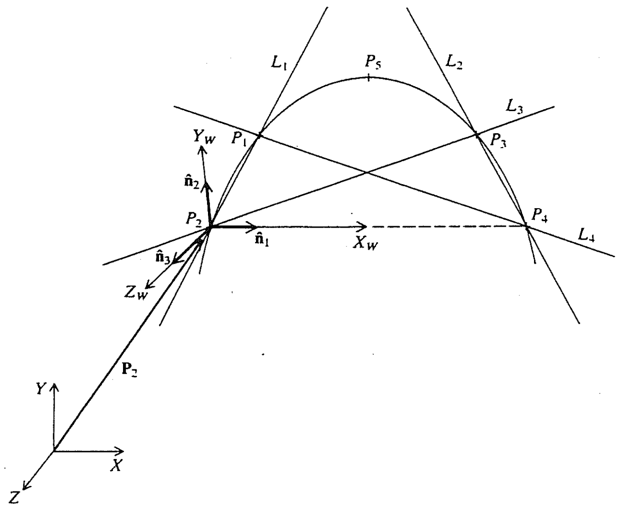


FIGURE 5-38
Local coordinate system of a conic curve.

At this point, it might be useful to ensure that the five points the user has inputted lie in one plane (the conic curve plane). This can simply be achieved by checking whether the scalar (dot) products of the vector $\hat{\mathbf{n}}_3$ with the vectors $(\mathbf{P}_3 - \mathbf{P}_2)$ and $(\mathbf{P}_3 - \mathbf{P}_5)$ are zeros or not.

The points P_1 to P_5 can be transformed from the MCS to the local system using the equation

$$[x \quad y \quad z \quad 1]^T = [T]\begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix} \tag{5.69}$$

or

$$[x_W \quad y_W \quad z_W \quad 1]^T = [T]^{-1}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.70}$$

where

$$[T] = \left[\begin{array}{ccc|c} n_{1x} & n_{2x} & n_{3x} & x_2 \\ n_{1y} & n_{2y} & n_{3y} & y_2 \\ n_{1z} & n_{2z} & n_{3z} & z_2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{5.71}$$

Notice that the Z_W components of all the points will be zeros.

The equation of a line can be written as

$$Ax_W + By_W + C = 0$$

or

$$L = x_W + \frac{B}{A} y_W + \frac{C}{A} = 0 \tag{5.72}$$

For each of the lines L_1 to L_4, the local coordinates of its two known points can be substituted into the above equation to find the two ratios B/A and C/A. Utilizing the resulting equations with Eq. (5.65) gives an implicit equation of the conic curve that has the general form

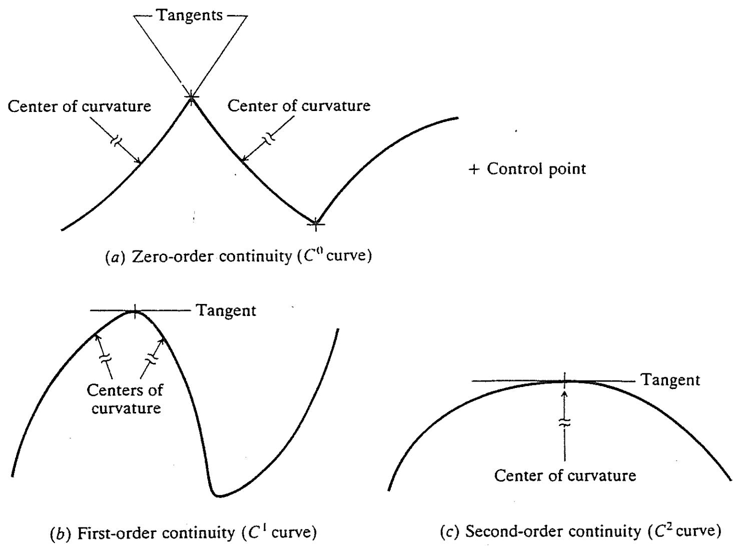
$$f(x_W, y_W, u) = 0, \qquad 0 \leq u \leq 1 \tag{5.73}$$

Substituting the local coordinates of P_5 into this equation gives the value of u. Now, this equation can be used to generate points on the conic curve. These points can be transformed to the MCS using Eq. (5.69) for display, plotting, and storage purposes. Notice that this approach can also apply to Eq. (5.66).

5.6 PARAMETRIC REPRESENTATION OF SYNTHETIC CURVES

Analytic curves, described in the previous section, are usually not sufficient to meet geometric design requirements of mechanical parts. Products such as car bodies, ship hulls, airplane fuselage and wings, propeller blades, shoe insoles, and bottles are a few examples that require free-form, or synthetic, curves and surfaces. The need for synthetic curves in design arises on two occasions: when a curve is represented by a collection of measured data points and when an existing curve must change to meet new design requirements. In the latter occasion, the designer would need a curve representation that is directly related to the data points and is flexible enough to bend, twist, or change the curve shape by changing one or more data points. Data points are usually called control points and the curve itself is called an interpolant if it passes through all the data points.

Mathematically, synthetic curves represent a curve-fitting problem to construct a smooth curve that passes through given data points. Therefore, polynomials are the typical form of these curves. Various continuity requirements can be specified at the data points to impose various degrees of smoothness of the resulting curve. The order of continuity becomes important when a complex curve is modeled by several curve segments pieced together end to end. Zero-order continuity (C^0) yields a position continuous curve. First (C^1)- and second (C^2)-order continuities imply slope and curvature continuous curves respectively. A C^1 curve is the minimum acceptable curve for engineering design. Figure 5-39 shows a geometrical interpretation of these orders of continuity. A cubic polynomial is the minimum-order polynomial that can guarantee the generation of C^0, C^1, or C^2 curves. In addition, the cubic polynomial is the lowest-degree polynomial that permits inflection within a curve segment and that allows representation of nonplanar (twisted) three-dimensional curves in space. Higher-order polynomials are not commonly used in CAD/CAM because they tend to oscillate about control points, are computationally inconvenient, and are uneconomical of storing curve and surface representations in the computer.



(a) Zero-order continuity (C^0 curve)

(b) First-order continuity (C^1 curve)

(c) Second-order continuity (C^2 curve)

FIGURE 5-39
Various orders of continuity of curves.

The type of input data and its influence on the control of the resulting synthetic curve determine the use and effectiveness of the curve in design. For example, curve segments that require positions of control points and/or tangent vectors at these points are easier to deal with and gather data for than those that might require curvature information. Also, the designer may prefer to control the shape of the curve locally instead of globally by changing the control point(s). If changing a control point results in changing the curve locally in the vicinity of that point, local control of the curve is achieved; otherwise global control results.

Major CAD/CAM systems provide three types of synthetic curves: Hermite cubic spline, Bezier, and B-spline curves. The cubic spline curve passes through the data points and therefore is an interpolant. Bezier and B-spline curves in general approximate the data points, that is, they do not pass through them. Under certain conditions, the B-spline curve can be an interpolant, as will be seen in Sec. 5.6.3. Both the cubic spline and Bezier curves have a first-order continuity and the B-spline curve has a second-order continuity. The formulation of each curve is discussed below.

5.6.1 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

$$\mathbf{P}(u) = \sum_{i=0}^{3} \mathbf{C}_i u^i, \qquad 0 \le u \le 1 \tag{5.74}$$

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

$$\begin{aligned} x(u) &= C_{3x}u^3 + C_{2x}u^2 + C_{1x}u + C_{0x} \\ y(u) &= C_{3y}u^3 + C_{2y}u^2 + C_{1y}u + C_{0y} \\ z(u) &= C_{3z}u^3 + C_{2z}u^2 + C_{1z}u + C_{0z} \end{aligned} \tag{5.75}$$

In an expanded vector form, Eq. (5.74) can be written as

$$\mathbf{P}(u) = \mathbf{C}_3 u^3 + \mathbf{C}_2 u^2 + \mathbf{C}_1 u + \mathbf{C}_0 \tag{5.76}$$

This equation can also be written in a matrix form as

$$\mathbf{P}(u) = \mathbf{U}^T\mathbf{C} \tag{5.77}$$

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

The tangent vector to the curve at any point is given by differentiating Eq. (5.74) with respect to u to give

$$\mathbf{P}'(u) = \sum_{i=0}^{3} \mathbf{C}_i i u^{i-1}, \qquad 0 \le u \le 1 \tag{5.78}$$

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. 5-40. Applying the boundary conditions ($\mathbf{P}_0$, $\mathbf{P}'_0$ at $u = 0$ and $\mathbf{P}_1$, $\mathbf{P}'_1$ at $u = 1$), Eqs. (5.74) and (5.78) give

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{C}_0 \\ \mathbf{P}'_0 &= \mathbf{C}_1 \\ \mathbf{P}_1 &= \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0 \\ \mathbf{P}'_1 &= 3\mathbf{C}_3 + 2\mathbf{C}_2 + \mathbf{C}_1 \end{aligned} \tag{5.79}$$

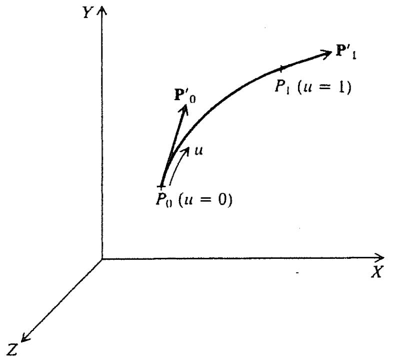


FIGURE 5-40
Hermite cubic spline curve.

Solving these four equations simultaneously for the coefficients gives

$$\begin{aligned} \mathbf{C}_0 &= \mathbf{P}_0 \\ \mathbf{C}_1 &= \mathbf{P}'_0 \\ \mathbf{C}_2 &= 3(\mathbf{P}_1 - \mathbf{P}_0) - 2(\mathbf{P}'_0 - \mathbf{P}'_1) \\ \mathbf{C}_3 &= 2(\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}'_0 + \mathbf{P}'_1 \end{aligned} \tag{5.80}$$

Substituting Eqs. (5.80) into Eq. (5.76) and rearranging gives

$$\begin{aligned} \mathbf{P}(u) = {} & (2u^3 - 3u^2 + 1)\mathbf{P}_0 + (-2u^3 + 3u^2)\mathbf{P}_1 \\ & + (u^3 - 2u^2 + u)\mathbf{P}'_0 + (u^3 - u^2)\mathbf{P}'_1, \qquad 0 \le u \le 1 \end{aligned} \tag{5.81}$$

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

$$\begin{aligned} \mathbf{P}'(u) = {} & (6u^2 - 6u)\mathbf{P}_0 + (-6u^2 + 6u)\mathbf{P}_1 \\ & + (3u^2 - 4u + 1)\mathbf{P}'_0 + (3u^2 - 2u)\mathbf{P}'_1, \qquad 0 \le u \le 1 \end{aligned} \tag{5.82}$$

The functions of u in Eqs. (5.81) and (5.82) 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 (5.81) can be written in a matrix form as

$$\mathbf{P}(u) = \mathbf{U}^T[M_H]\mathbf{V}, \qquad 0 \le u \le 1 \tag{5.83}$$

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

$$[M_H] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.84}$$

$$\mathbf{V} = [\mathbf{P}_0 \quad \mathbf{P}_1 \quad \mathbf{P}'_0 \quad \mathbf{P}'_1]^T \tag{5.85}$$

Comparing Eqs. (5.77) and (5.83) show that $\mathbf{C} = [M_H]\mathbf{V}$ or $\mathbf{V} = [M_H]^{-1}\mathbf{C}$ where

$$[M_H]^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \tag{5.86}$$

Similarly, Eq. (5.82) can be written as

$$\mathbf{P}'(u) = \mathbf{U}^T[M_H]^u\mathbf{V} \tag{5.87}$$

where $[M_H]^u$ is given by

$$[M_H]^u = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.88}$$

Equation (5.81) describes the cubic spline curve in terms of its two endpoints 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. (5.81) 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 (5.81) 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 (5.81) 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. 5-41) 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. (5.81) 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

$$\mathbf{P}''(u_1 = 1) = \mathbf{P}''(u_2 = 0) \tag{5.89}$$

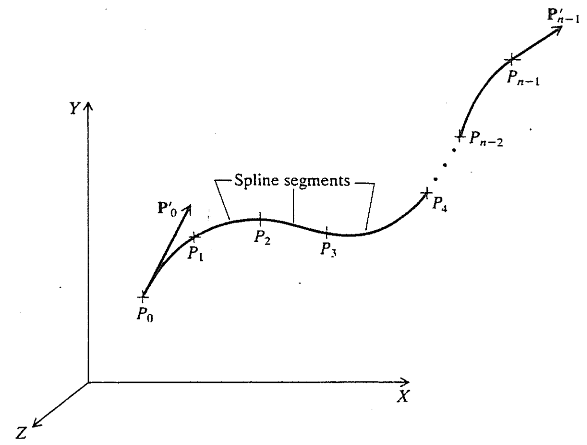


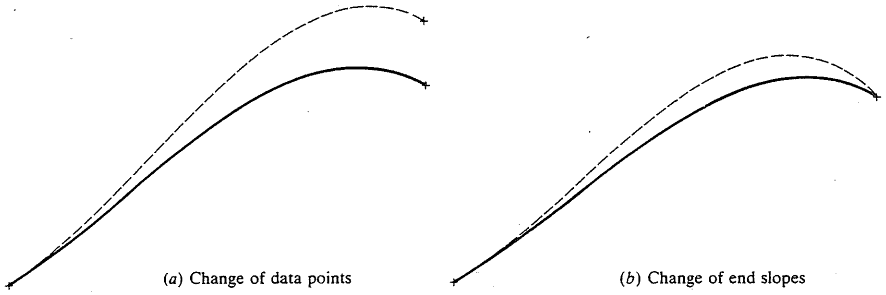
FIGURE 5-41
Hermite cubic spline curve.

where the subscripts of u refer to the segment number. Differentiating Eq. (5.82) and using the result with Eq. (5.89), we obtain

$$\mathbf{P}'_1 = -\tfrac{1}{4}(3\mathbf{P}_0 + \mathbf{P}'_0 - 3\mathbf{P}_2 + \mathbf{P}'_2) \tag{5.90}$$

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. $P'_1 = -\frac{1}{4}(3P_1 + P'_1 - 3P_2 + P'_2)$

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,


(a) Change of data points

(b) Change of end slopes

FIGURE 5-42
Control of cubic spline curve.

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 5-42 shows the control aspects of the cubic spline curve.

Example 5.18. 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. (5.81) and (5.82), we get

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

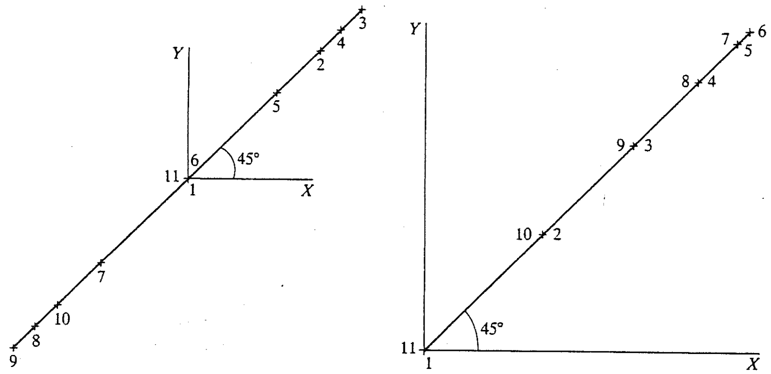
$$\mathbf{P}'(u) = (6u^2 - 6u + 1)\mathbf{P}'_0$$

This spline passes by $\mathbf{P}_0$ at $u = 0, \frac{1}{2}, 1$. Figure 5-43*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 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 $\mathbf{P}'(u) = \mathbf{0}$.

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

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

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



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

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

FIGURE 5-43
Closed cubic spline curves.

This spline has an extreme point at $u = \frac{1}{2}$. It begins the tangent to $\mathbf{P}'_0$ and then overlaps to become the tangent to $\mathbf{P}'_1$. Figure 5-43*b* shows the curve for $\mathbf{P}_0 = [0 \quad 0 \quad 0]^T$ 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.

5.6.2 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. 5-44. 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

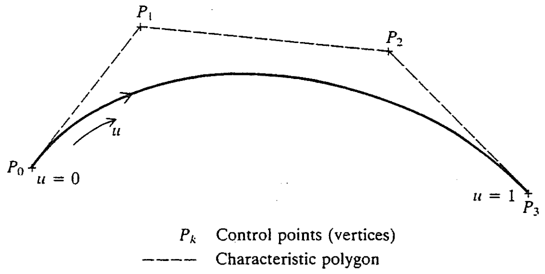


FIGURE 5-44
Cubic Bezier curve (nomenclature).

predict the curve shape once its control points are given as illustrated in Fig. 5-45. 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.

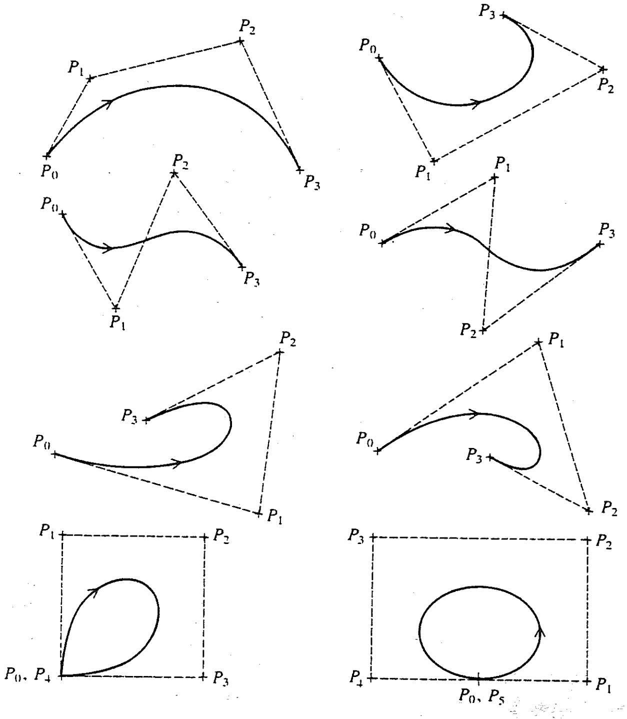


FIGURE 5-45
Cubic Bezier curves for various control points.

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

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(u), \qquad 0 \le u \le 1 \tag{5.91}$$

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

$$B_{i,n}(u) = C(n, i)u^i(1 - u)^{n-i} \tag{5.92}$$

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

$$C(n, i) = \frac{n!}{i!\,(n - i)!} \tag{5.93}$$

Utilizing Eqs. (5.92) and (5.93) and observing that $C(n, 0) = C(n, n) = 1$, Eq. (5.91) can be expanded to give

$$\mathbf{P}(u) = \mathbf{P}_0(1 - u)^n + \mathbf{P}_1 C(n, 1)u(1 - u)^{n-1} + \mathbf{P}_2\, C(n, 2)u^2(1 - u)^{n-2} \\ + \cdots + \mathbf{P}_{n-1}C(n, n - 1)u^{n-1}(1 - u) + \mathbf{P}_n u^n, \qquad 0 \le u \le 1 \tag{5.94}$$

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. (5.94).
2. The curve is tangent to the first and last segments of the characteristic polygon. Using Eqs. (5.91) and (5.92), the rth derivatives at the starting and ending points are given by respectively:

$$\mathbf{P}^r(0) = \frac{n!}{(n - r)!} \sum_{i=0}^{r} (-1)^{r-i} C(r, i)\mathbf{P}_i \tag{5.95}$$

$$\mathbf{P}^r(1) = \frac{n!}{(n - r)!} \sum_{i=0}^{r} (-1)^{i} C(r, i)\mathbf{P}_{n-i} \tag{5.96}$$

Therefore, the first derivatives at the endpoints are

$$\mathbf{P}'(0) = n(\mathbf{P}_1 - \mathbf{P}_0) \tag{5.97}$$

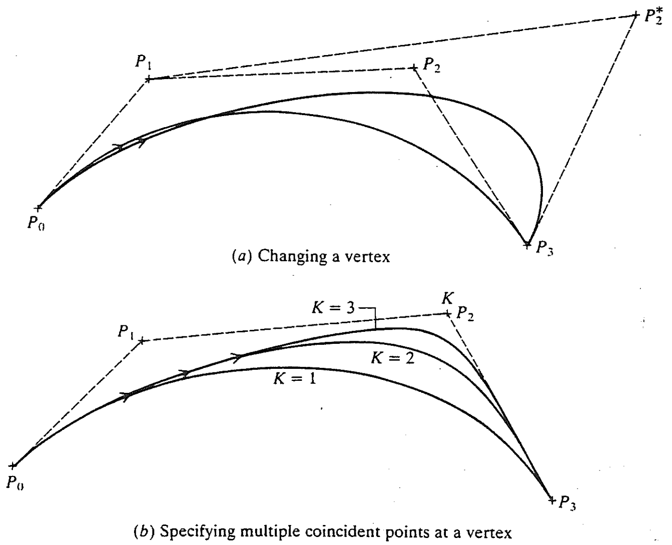
$$\mathbf{P}'(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}) \tag{5.98}$$

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. (5.94) 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. 5-46. In Fig. 5-46*a*, the vertex P_2 is pulled to the new position P_2^* and in Fig. 5-46*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 5-45 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.

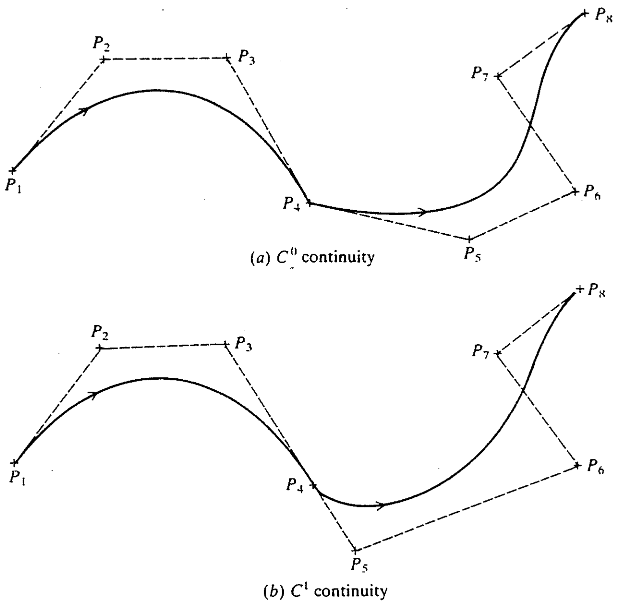
Thus far, only one Bezier curve segment is considered. In practical applications, the need may arise to deal with composite curves where various curve segments are blended or joined together. In these applications maintaining continuity of various orders between the segments might be desired. Figure 5-47 shows two 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. 5-47*a*. 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. 5-47*b*, the three points P_3, P_4, and P_5 must be collinear. Utilizing Eqs. (5.97) and (5.98), we can write



(*a*) Changing a vertex

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

FIGURE 5-46
Modifications of cubic Bezier curve.



(*a*) C^0 continuity

(*b*) C^1 continuity

FIGURE 5-47
Blending Bezier curve segments.

$$\mathbf{P}_4 - \mathbf{P}_3 = \tfrac{4}{3}(\mathbf{P}_5 - \mathbf{P}_4) \tag{5.99}$$

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. 5-48 defines the convex hull of a Bezier curve. The hull is formed by connecting the vertices of the characteristic polygon.

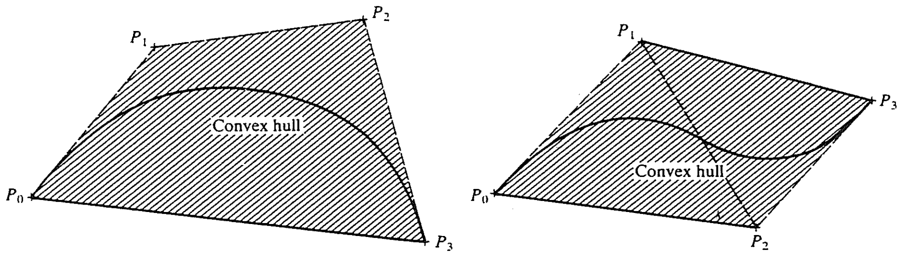


FIGURE 5-48
The convex hull of a Bezier curve.

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 5.19. The coordinates of four control points relative to a current WCS are given by

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

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 (5.91) gives

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

Using Eqs. (5.92) and (5.93), the above equation becomes

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

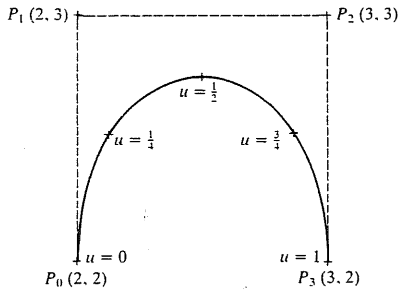


FIGURE 5-49
Bezier curve and generated points.

Substituting the u values into this equation gives

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

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

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

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

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

Observe that $\sum_{i=0}^{3} B_{i,3}$ is always equal to unity for any u value. Figure 5-49 shows the curve and the points.

Example 5.20. A cubic spline curve is defined by the equation

$$\mathbf{P}(u) = \mathbf{C}_3 u^3 + \mathbf{C}_2 u^2 + \mathbf{C}_1 u + \mathbf{C}_0, \qquad 0 \le u \le 1 \tag{5.100}$$

where $\mathbf{C}_3$, $\mathbf{C}_2$, $\mathbf{C}_1$, and $\mathbf{C}_0$ are the polynomial coefficients [see Eq. (5.76)]. Assuming these coefficients are known, find the four control points that define an identical Bezier curve.

Solution. The Bezier equation is

$$\mathbf{P}(u) = \mathbf{P}_0 B_{0,3} + \mathbf{P}_1 B_{1,3} + \mathbf{P}_2 B_{2,3} + \mathbf{P}_3 B_{3,3} \tag{5.101}$$

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. (5.101) and rearranging, we obtain

$$\mathbf{P}(u) = (-\mathbf{P}_0 + 3\mathbf{P}_1 - 3\mathbf{P}_2 + \mathbf{P}_3)u^3 + (3\mathbf{P}_0 - 6\mathbf{P}_1 + 3\mathbf{P}_2)u^2 + (-3\mathbf{P}_0 + 3\mathbf{P}_1)u + \mathbf{P}_0 \tag{5.102}$$

Comparing the coefficients of Eqs. (5.100) and (5.102) gives

$$\mathbf{P}_0 = \mathbf{C}_0$$
$$\mathbf{P}_1 = \tfrac{1}{3}\mathbf{C}_1 + \mathbf{C}_0$$
$$\mathbf{P}_2 = \tfrac{1}{3}(\mathbf{C}_2 + 2\mathbf{C}_1 + 3\mathbf{C}_0)$$
$$\mathbf{P}_3 = \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0$$

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. (5.100) for the same value of u.

5.6.3 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

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(u), \qquad 0 \le u \le u_{\max} \tag{5.103}$$

$N_{i,k}(u)$ 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. (5.103) and (5.91). 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:	$N_{i,k}(u) \ge 0$
Local support:	$N_{i,k}(u) = 0 \quad$ 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

$$N_{i,k}(u) = (u - u_i)\frac{N_{i,k-1}(u)}{u_{i+k-1} - u_i} + (u_{i+k} - u)\frac{N_{i+1,k-1}(u)}{u_{i+k} - u_{i+1}} \tag{5.104}$$

where

$$N_{i,1} = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{5.105}$$

Choose $0/0 = 0$ if the denominators in Eq. (5.104) become zero. Equation (5.105) 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. (5.104)] 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

$$u_j = \begin{cases} 0, & j < k \\ j - k + 1, & k \le j \le n \\ n - k + 2, & j > n \end{cases} \tag{5.106}$$

where

$$0 \le j \le n + k \tag{5.107}$$

and the range of u is

$$0 \le u \le n - k + 2 \tag{5.108}$$

Relation (5.107) 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.

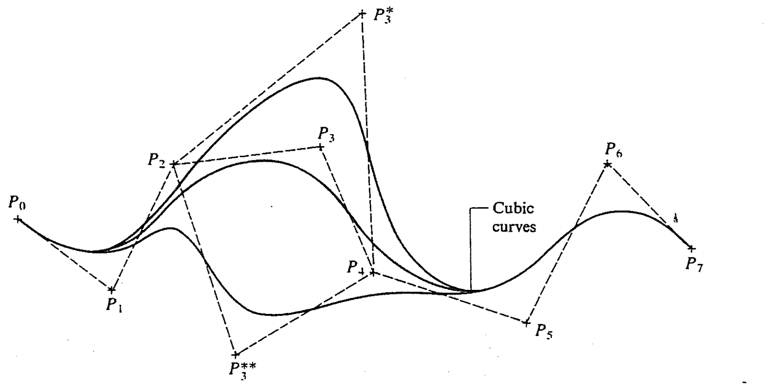


FIGURE 5-50
Local control of B-spline curves.

While the degree of the resulting B-spline curve is controlled by k, the range of the parameter u as given by Eq. (5.108) 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. (5.108) to be greater than the lower bound for the u range to be valid, that is,

$$n - k + 2 > 0 \tag{5.109}$$

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 5-50 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. 5-50.
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. 5-51. 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. 5-51). This is not the case for other degrees.

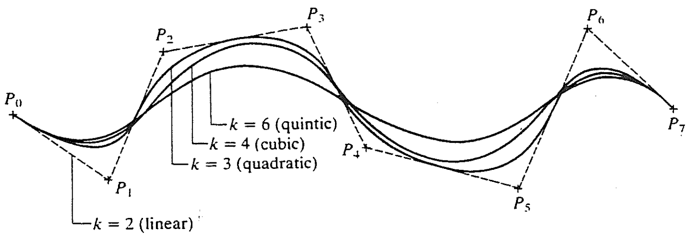
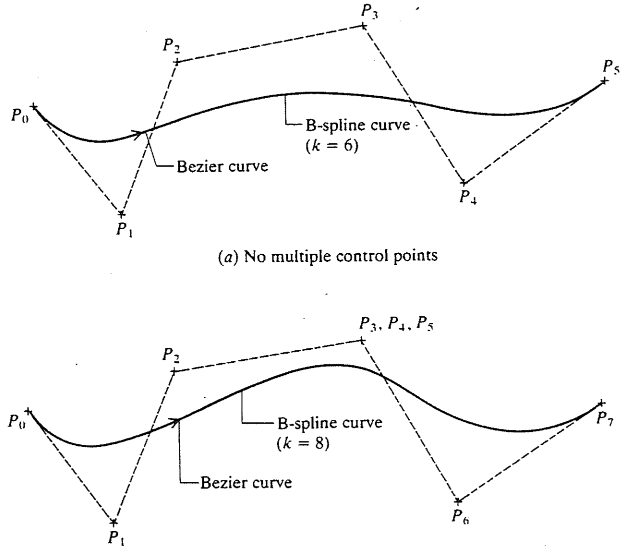


FIGURE 5-51
Effect of the degree of B-spline curve on its shape.

5. If k equals the number of control points $(n + 1)$, then the resulting B-spline curve becomes a Bezier curve (see Fig. 5-52). In this case the range of u becomes zero to one [see Eq. (5.108)] 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. 5-53). 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

FIGURE 5-52
Identical B-spline and Bezier curves.

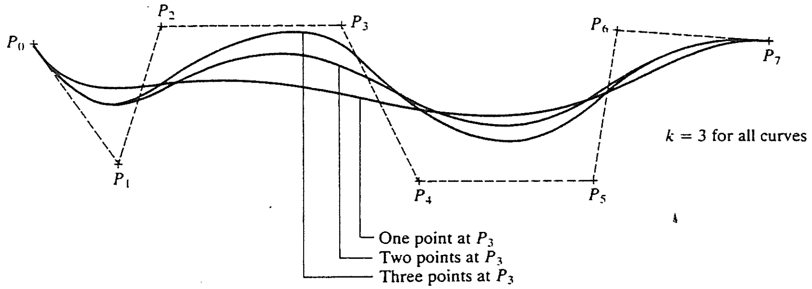


FIGURE 5-53
Multiple control point B-spline curves.

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 (5.106) to (5.108) 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. 5-54. The knot vector in this case is $[0 \quad 1 \quad 2]$. Quadratic and cubic closed curves have quadratic and cubic basis functions with intervals of (0, 3) and (0, 4) and knot vectors of $[0 \quad 1 \quad 2 \quad 3]$ and $[0 \quad 1 \quad 2 \quad 3 \quad 4]$ respectively.

The closed B-spline curve of degree $(k - 1)$ or order k defined by $(n + 1)$ control points is given by Eq. (5.103) as the open curve. However, for closed curves Eqs. (5.104) to (5.108) become

$$N_{i,k}(u) = N_{0,k}((u - i + n + 1) \bmod (n + 1)) \tag{5.110}$$

$$u_j = j, \qquad 0 \le j \le n + 1 \tag{5.111}$$

$$0 < j \le n + 1 \tag{5.112}$$

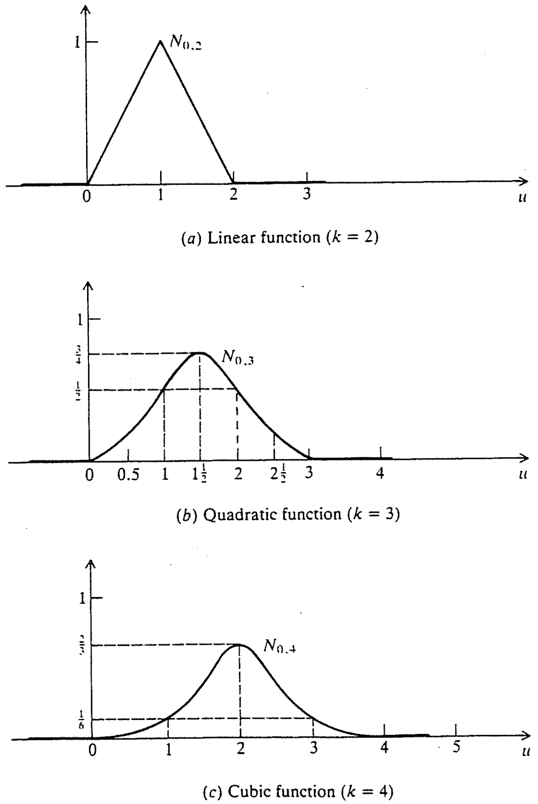
and the range of u is

$$0 \le u \le n + 1 \tag{5.113}$$

The mod $(n + 1)$ in Eq. (5.110) is the modulo function. It is defined as

$$A \bmod n = \begin{cases} A, & A < n \\ 0, & A = n \\ \text{remainder of } A/n, & A > n \end{cases} \tag{5.114}$$

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. (5.104) and (5.105).



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

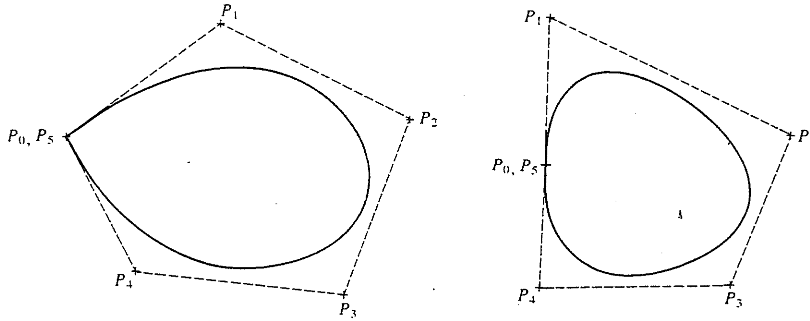
(*b*) Quadratic function ($k = 3$)

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

FIGURE 5-54
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. 5-55. 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.



(a) C^0 continuity

(b) C^1 continuity

FIGURE 5-55
An open B-spline curve with P_0 and P_n coincident.

Example 5.21. Find the equation of a cubic B-spline curve defined by the same control points as in Example 5.19. 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 (5.106) gives the knot vector

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

The range of u [Eq. (5.108)] is $0 \le u \le 1$. Equation (5.103) gives

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4} + \mathbf{P}_1 N_{1,4} + \mathbf{P}_2 N_{2,4} + \mathbf{P}_3 N_{3,4}, \qquad 0 \le u \le 1 \qquad (5.115)$$

To calculate the above B-spline functions, use Eqs. (5.104) and (5.105) together with the knot vector as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Substituting $N_{i,4}$ into Eq. (5.115) gives

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

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

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

This equation is the same as the one for the Bezier curve in Example 5.19. 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 $N_{i,3}$ 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 5.22. 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. (5.111) to (5.113), the knot vector $[u_0 \quad u_1 \quad u_2 \quad u_3 \quad u_4]$ is the integers $[0 \quad 1 \quad 2 \quad 3 \quad 4]$ and the range of u is $0 \le u \le 4$. Equation (5.103) gives the curve equation as

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4} + \mathbf{P}_1 N_{1,4} + \mathbf{P}_2 N_{2,4} + \mathbf{P}_3 N_{3,4}, \qquad 0 \le u \le 4 \tag{5.116}$$

To calculate the above B-spline functions, use Eq. (5.110) 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. (5.116) we get

$$\mathbf{P}(u) = \mathbf{P}_0 N_{0,4}((u + 4) \bmod 4) + \mathbf{P}_1 N_{0,4}((u + 3) \bmod 4)$$
$$+ \mathbf{P}_2 N_{0,4}((u + 2) \bmod 4) + \mathbf{P}_3 N_{0,4}((u + 1) \bmod 4), \qquad 0 \le u \le 4 \tag{5.117}$$

In Eq. (5.117), 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 5.21 are performed using the above knot vector as follows:

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

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

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

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

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

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

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

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

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

$$N_{0,4} = (u - u_0)\frac{N_{0,3}}{u_3 - u_0} + (u_4 - u)\frac{N_{1,3}}{u_4 - u_1} = \frac{1}{3}uN_{0,3} + \frac{1}{3}(4 - u)N_{1,3}$$
$$= \tfrac{1}{6}\{u^3 N_{0,1} + [u^2(2 - u) + u(3 - u)(u - 1) + (4 - u)(u - 1)^2]N_{1,1}$$
$$+ [u(3 - u)^2 + (4 - u)(u - 1)(3 - u) + (4 - u)^2(u - 2)]N_{2,1} + (4 - u)^3 N_{3,1}\}$$

or

$$N_{0,4} = \tfrac{1}{6}[u^3 N_{0,1} + (-3u^3 + 12u^2 - 12u + 4)N_{1,1} + (3u^3 - 24u^2 + 60u - 44)N_{2,1}$$
$$+ (-u^3 + 12u^2 - 48u + 64)N_{3,1}]$$

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

$$N_{0,4}(u) = \begin{cases} \tfrac{1}{6}u^3, & 0 \le u \le 1 \\ \tfrac{1}{6}(-3u^3 + 12u^2 - 12u + 4), & 1 \le u \le 2 \\ \tfrac{1}{6}(3u^3 - 24u^2 + 60u - 44), & 2 \le u \le 3 \\ \tfrac{1}{6}(-u^3 + 12u^2 - 48u + 64), & 3 \le u \le 4 \end{cases} \tag{5.118}$$

To check the correctness of the above expression of $N_{0,4}(u)$, one would expect to obtain Fig. 5-54*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 (5.117) and (5.118) 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:

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

Similarly,

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

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

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

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

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

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.

5.6.4 Rational Curves

A rational curve is defined by the algebraic ratio of two polynomials while a nonrational curve [Eq. (5.103) 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. 9 (Sec. 9.2.5). 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. (9.59).

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

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i R_{i,k}(u), \qquad 0 \le u \le u_{\max} \tag{5.119}$$

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

$$R_{i,k}(u) = \frac{h_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)} \tag{5.120}$$

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. 5.6.3 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 $H = [h_0 \quad h_1 \quad h_2 \quad h_3 \quad \cdots \quad h_n]^T$ 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.

5.7 CURVE MANIPULATIONS

Analytic and synthetic curves essential to wireframe modeling have been presented in Secs. 5.5 and 5.6. The effective use of these curves in a design and manufacturing environment depends mainly on their manipulation to achieve goals in hand. A user might want to blend Bezier and B-spline curves with a certain continuity requirement, or the intersection of two curves in space might provide the coordinates of an important point to engineering calculations or modeling of a part. The next sections cover some useful features of curve manipulations. Refer to Chap. 12 for more details on the implementation of curve manipulations in CAD/CAM softwear.

5.7.1 Displaying

Displaying curves provides the designer with a means of visualizing geometric models within the limits of the wireframe modeling technique. Various colors can be assigned to various curves for identification and other purposes. To display a curve many closely spaced points on it are generated for various permissible values of the parameter u. The curve parametric equation is obviously used to generate the coordinates of these points. The display processing unit (see Chap. 2) receives these coordinates and changes them to two-dimensional coordinates relative to the device coordinate system of the display terminal. The resulting points are displayed and are connected by short-line segments. Line-generation hardware exists (refer to Chap. 2) for creating and displaying these line segments.

To display a straight line, the two endpoints are fed to the line-generation hardware to generate intermediate points and connect them by short-line segments. Equations of curves, such as circles and conic sections, that involve trigonometric functions are usually rewritten to minimize computation time to generate points on these curves for display purposes [see Eqs. (5.25), (5.35), (5.37), and (5.55)]. As mentioned in Chap. 2, displaying a curve and representing it in its database are two different things.

5.7.2 Evaluating Points on Curves

Points on curves are generated for display purposes as discussed previously or for other purposes. For example, finite element modeling techniques (see Chap. 18) requires generating nodal points (nodes) on the model to be used later in finite element analysis. In most CAD/CAM systems, the function of evaluating points on curves is made available to the user via a "generate point on" command or its equivalent.

It is also mentioned in the previous section that a curve parametric equation is used to evaluate points on it. Evaluation methods must be efficient and fast for interactive purposes as well as capable of producing enough points to display a smooth curve. The obvious method of calculating the coordinates of points on a curve by substituting successive values of its parameter into its equation is inefficient. Incremental methods proved more efficient.

The forward difference technique to evaluate a curve polynomial equation at equal intervals of its parameter is the most common incremental method. Consider applying the method to a cubic polynomial as an example. If the curve is given by

$$\mathbf{P}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}, \qquad 0 \le u \le 1 \tag{5.121}$$

and if $(n + 1)$ equally spaced points are to be evaluated for the u range, then $\Delta u = 1/n$. The following equations can be used to initialize the method:

$$\begin{aligned} \mathbf{P}(0) &= \mathbf{d} \\ \Delta_1 \mathbf{P}_0 &= \mathbf{a}(\Delta u)^3 + \mathbf{b}(\Delta u)^2 + \mathbf{c}(\Delta u) \\ \Delta_2 \mathbf{P}_0 &= 6\mathbf{a}(\Delta u)^3 + 2\mathbf{b}(\Delta u)^2 \\ \Delta_3 \mathbf{P}_0 &= 6\mathbf{a}(\Delta u)^3 \end{aligned} \tag{5.122}$$

Any successive point can be evaluated from

$$\begin{aligned} \mathbf{P}_{i+1} &= \mathbf{P}_i + \Delta_1 \mathbf{P}_i, \qquad 0 \le i \le n \\ \Delta_1 \mathbf{P}_{i+1} &= \Delta_1 \mathbf{P}_i + \Delta_2 \mathbf{P}_i \\ \Delta_2 \mathbf{P}_{i+1} &= \Delta_2 \mathbf{P}_i + \Delta_3 \mathbf{P}_i \end{aligned} \tag{5.123}$$

These equations show that only three additions are needed to generate any point and $3n$ additions generate the n points. In general, for a polynomial of degree n, n additions are required to calculate one point.

Evaluating a point on a curve given by its parametric value, u, is sometimes called the direct point solution. It entails evaluating three polynomials in u, one for each coordinate of the point. The inverse problem is another form of evaluating a point on a curve. Given a point on or close to a curve in terms of its cartesian coordinates x, y, and z, find the corresponding u value. This problem arises, for example, if tangent vectors are to be evaluated at certain locations on the curve. The solution of this problem is called the inverse point solution and requires the solution of a nonlinear polynomial in u via numerical methods (see Prob. 5.18 at the end of the chapter).

5.7.3 Blending

The construction of composite curves from the various types of parametric curve segments covered in Secs. 5.5 and 5.6 forms the core of the blending problem which can be stated as follows. Given two curve segments $\mathbf{P}_1(u_1)$, $0 < u_1 < a$, $\mathbf{P}_2(u_2)$, $0 < u_2 < b$, find the conditions for the two segments to be continuous at the joint. Notice that the upper limits on u_1 and u_2 are taken to be a and b, and not 1, for generality (B-spline curves have an upper limit that is not unity in general). Three classes of continuity at the joint can be considered (see Fig. 5-39). The first is C^0 continuity; that is, the ending point of the first curve and the starting point of the second curve are the same. This gives

$$\mathbf{P}_1(a) = \mathbf{P}_2(0) \tag{5.124}$$

If the two segments are to be C^1 continuous as well, they must have slope continuity, that is,

$$\begin{aligned} \mathbf{P}'_1(a) &= \alpha_1 \mathbf{T} \\ \mathbf{P}'_2(0) &= \alpha_2 \mathbf{T} \end{aligned} \tag{5.125}$$

where α_1 and α_2 are constants and $\mathbf{T}$ is the common unit tangent vector at the joint. It has already been shown how to use curve characteristics for blending purposes as in the case with Bezier curves. Another example is the blending of a Bezier curve and an open B-spline curve. For slope continuity at the joint, the last segment of the control polygon of the former and the first segment of the control polygon of the latter must be colinear.

The third useful class of continuity is if the curvature (C^2 continuity) is to be continuous at the joint in addition to position and slope. To achieve curvature continuity is less straightforward and requires the binormal vector to a curve at a point. Figure 5-56 shows the tangent unit vector $\mathbf{T}$, the normal unit vector $\mathbf{N}$, the center of curvature O, and the radius of curvature ρ at point P on a curve

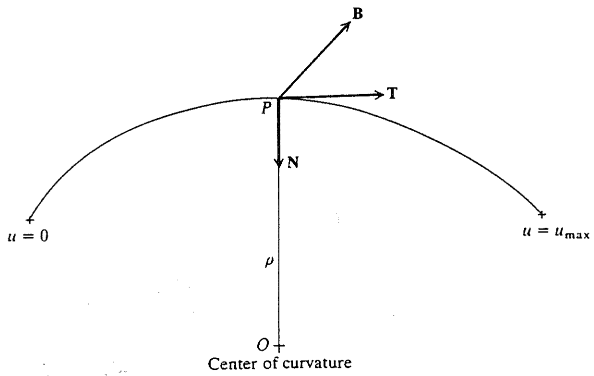


FIGURE 5-56
Binormal vector to a curve.

segment. The curvature at P is defined as $1/\rho$. The binormal vector **B** is defined as

$$\mathbf{B} = \mathbf{T} \times \mathbf{N} \tag{5.126}$$

The curvature is related to the curve derivatives through the vector **B** by the following equation:

$$\frac{1}{\rho}\mathbf{B} = \frac{\mathbf{P}' \times \mathbf{P}''}{|\mathbf{P}'|^3} \tag{5.127}$$

where $\mathbf{P}''$ is the second derivative with respect to the parameter u. The following condition can then be written for curvature continuity at the joint:

$$\frac{\mathbf{P}'_1(a) \times \mathbf{P}''_1(a)}{|\mathbf{P}'_1(a)|^3} = \frac{\mathbf{P}'_2(0) \times \mathbf{P}''_2(0)}{|\mathbf{P}'_2(0)|^3} \tag{5.128}$$

Substituting Eqs. (5.125) into the above equation gives

$$\mathbf{T} \times \mathbf{P}''_1(a) = \left(\frac{\alpha_1}{\alpha_2}\right)^2 \mathbf{T} \times \mathbf{P}''_2(0) \tag{5.129}$$

This equation can be satisfied if

$$\mathbf{P}''_2(0) = \left(\frac{\alpha_2}{\alpha_1}\right)^2 \mathbf{P}''_1(a) \tag{5.130}$$

or, in general, if

$$\mathbf{P}''_2(0) = \left(\frac{\alpha_2}{\alpha_1}\right)^2 \mathbf{P}''_1(0) + \gamma \mathbf{P}'_1(a) \tag{5.131}$$

where γ is an arbitrary scalar which is chosen as zero for practical purposes.

5.7.4 Segmentation

Segmentation or curve splitting is defined as replacing one existing curve by one or more curve segments of the same curve type such that the shape of the composite curve is identical to that of the original curve. Segmentation is a very useful feature for CAD/CAM systems; it is implemented as a "divide entity" command. Model clean-up for drafting and documentation purposes is an example where an entity (curve) might be divided into two at the line of sight of another (normally the two entities do not intersect in space). One of the resulting segments is then removed or blanked out. Another example is when a closed curve has to be split for modeling purposes. Consider the case of creating a ruled surface using a closed rectangle and a circle as rails of the surface (see Chap. 6). In this case the circle is split into four segments at the intersections of the rectangle diagonals with the circle. Each of the resulting four segments is then used with the proper rectangle side to create the four ruled surfaces shown in Fig. 5-57.

Mathematically, curve segmentation is a reparametrization or parameter transformation of the curve. Splitting lines, circles, and conics is a simple problem. To split a line connecting two points P_0 and P_1 at a point P_2, all that is needed is to define two new lines connecting the point pairs (P_0, P_2) and (P_2, P_1). For circles and conics, the angle corresponding to the splitting point together with the starting and ending angles of the original curve defines the proper range of the parameter u (in this case the angle) for the resulting two segments. In other words, the parameter transformation in the case of lines, circles, and conics is trivial.

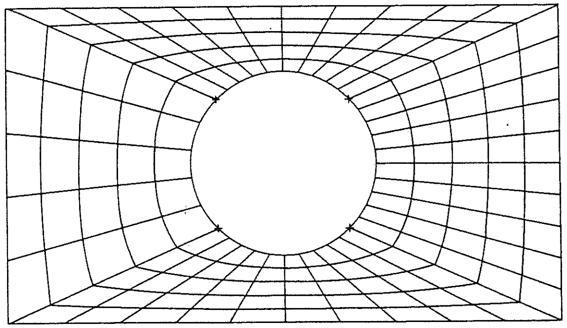


FIGURE 5-57
Segmentation of a circle for modeling purposes.

Polynomial curves such as cubic splines, Bezier curves, and B-spline curves require a different parameter transformation. If the degree of the polynomial defining a curve is to be unchanged, which is the case in segmentation, the transformation must be linear. Let us assume that a polynomial curve is defined over the range $u = u_0, u_m$. To split the curve at a point defined by $u = u_1$ means that the first and the second segments are to be defined over the range $u = u_0, u_1$ and $u = u_1, u_m$ respectively. A new parameter v is introduced for each segment such that its range is $v = 0, 1$ (see Fig. 5-58). The parameter transformation takes the form:

$$\begin{aligned} &u = u_0 + (u_1 - u_0)v && \text{for the first segment} \\ \text{and} \quad &u = u_1 + (u_m - u_1)v && \text{for the second segment} \end{aligned} \tag{5.132}$$

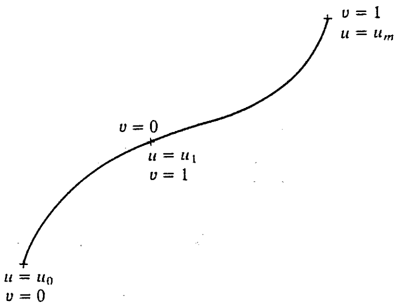


FIGURE 5-58
Reparametrization of a segmented curve.

It is clear that $v = 0$, 1 corresponds to the proper u values as required. If Eq. (5.132) is substituted into the equation of a given curve, the proper equation of each segment, and consequently its database, can be obtained in terms of the parameter v. To facilitate this substitution in the curve polynomial equation where u is raised to different powers, the following equation can be used:

$$u^n = \sum_{r=0}^{n} \binom{n}{r} u_0^r (\Delta u_0 \, v)^{n-r} \qquad (5.133)$$

where $\Delta u_0 = u_1 - u_0$. A similar equation can be written for the second segment. If the curve is to be divided into more than two segments, Eqs. (5.132) and (5.133) can be applied successively.

Example 5.23. A cubic spline connecting two points is to be divided by a user into two segments. Find the endpoints and slopes for each segment.

Solution. Instead of substituting Eq. (5.133) into Eq. (5.81) of the spline directly and reducing the result, it is more efficient to use the matrix form given by Eq. (5.83). The vector **U** in this equation can be written in terms of the parameter v using Eq. (5.133) as:

$$[u^3 \quad u^2 \quad u \quad 1] = [v^3 \quad v^2 \quad v \quad 1] \begin{bmatrix} \Delta u_0^3 & 0 & 0 & 0 \\ 3u_0 \, \Delta u_0^2 & \Delta u_0^2 & 0 & 0 \\ 3u_0^2 \, \Delta u_0 & 2u_0 \, \Delta u_0 & \Delta u_0 & 0 \\ u_0^3 & u_0^2 & u_0 & 1 \end{bmatrix}$$

or in a matrix form

$$\mathbf{U}^T = \mathbf{v}^T[T]$$

Substituting this equation into Eq. (5.83), we obtain

$$\mathbf{P}(u) = \mathbf{v}^T[T][M_H]\mathbf{V} = \mathbf{P}^*(v)$$

This equation can be rewritten in a similar form to Eq. (5.83) as

$$\mathbf{P}^*(v) = \mathbf{v}^T[M_H][M_H]^{-1}[T][M_H]\mathbf{V} = \mathbf{v}^T[M_H]\mathbf{V}^*$$

Therefore, the modified geometry, or boundary conditions, vector of the first spline segment is given by

$$\mathbf{V}^* = [M_H]^{-1}[T][M_H]\mathbf{V}$$

Expanding this equation and utilizing Eqs. (5.84) to (5.86) and $\Delta u_0 = u_1 - u_0$, we get

$$\begin{aligned} \mathbf{P}_0^* &= (2u_0^3 - 3u_0^2 - 1)\mathbf{P}_0 + (-2u_0^3 + 3u_0^2)\mathbf{P}_1 \\ &\quad + (u_0^3 - 2u_0^2 + u_0)\mathbf{P}'_0 + (u_0^3 - u_0^2)\mathbf{P}'_1 \\ \mathbf{P}_1^* &= (2u_1 - 3u_1^2 - 1)\mathbf{P}_0 + (-2u_1^3 + 3u_1^2)\mathbf{P}_1 \\ &\quad + (u_1^3 - 2u_1^2 + u_1)\mathbf{P}'_0 + (u_1^3 - u_1^2)\mathbf{P}'_1 \\ \mathbf{P}_0^{*\prime} &= (u_1 - u_0)[(6u_0^2 - 6u_0)\mathbf{P}_0 + (-6u_0^2 + 6u_0)\mathbf{P}_1 \\ &\quad + (3u_0^2 - 4u_0 + 1)\mathbf{P}'_0 + (3u_0^2 - 2u_0)\mathbf{P}'_1] \\ \mathbf{P}_1^{*\prime} &= (u_1 - u_0)[(6u_1^2 - 6u_1)\mathbf{P}_0 + (-6u_1^2 + 6u_1)\mathbf{P}_1 \\ &\quad + (3u_1^2 - 4u_1 + 1)\mathbf{P}'_0 + (3u_1^2 - 2u_1)\mathbf{P}'_1] \end{aligned}$$

where $\mathbf{P}_i^{*\prime} = d\mathbf{P}_i^*/dv$ and $\mathbf{P}'_i = d\mathbf{P}_i/du$. Comparing the above four equations with Eqs. (5.81) and (5.82) reveals that

$$\mathbf{P}_0^* = \mathbf{P}_0 \qquad \mathbf{P}_1^* = \mathbf{P}(u_1)$$

$$\mathbf{P}_0^{*\prime} = (u_1 - u_0)\mathbf{P}'_0 \qquad \mathbf{P}_1^{*\prime} = (u_1 - u_0)\mathbf{P}'(u_1)$$

These equations conclude that the first spline segment has the endpoints P_0 and $P(u_1)$ which lie on the original curve and its end tangent vectors are related to the end vectors of the original curve by the factor $(u_1 - u_0)$. The boundary conditions for the second segment can, therefore, be written as

$$\mathbf{P}_0^* = \mathbf{P}(u_1) \qquad \mathbf{P}_1^* = \mathbf{P}_1$$

$$\mathbf{P}_0^{*\prime} = (u_m - u_1)\mathbf{P}'(u_1) \qquad \mathbf{P}_1^{*\prime} = (u_m - u_1)\mathbf{P}'_1$$

While the endpoints of each segment are logically understood, the above equations show that the magnitudes of the tangent vectors are scaled by the range $(u_1 - u_0)$ [or $(u_m - u_1)$] of the parametric variable to preserve the directions of the tangent vectors and thus the shape of the curve. This scale factor can be derived in another way as follows:

$$\mathbf{P}^{*\prime} = \frac{d\mathbf{P}^*}{dv} = \frac{d\mathbf{P}}{dv} = \frac{d\mathbf{P}^*}{du}\frac{du}{dv} = \frac{d\mathbf{P}}{du}\frac{du}{dv}$$

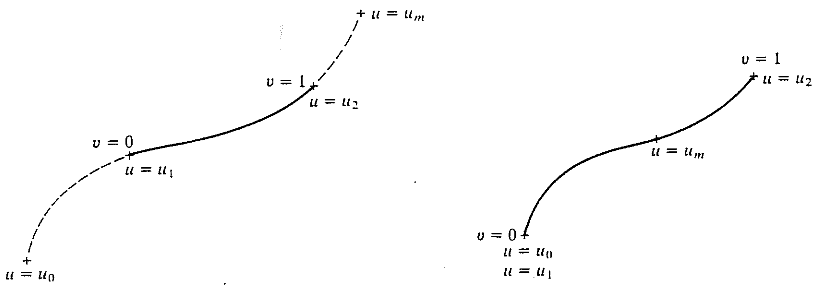
$$= \frac{du}{dv}\mathbf{P}'$$

Substituting Eq. (5.132) gives the same result as above.

Similar development can be applied to split a Bezier curve or a B-spline curve. Refer to the problems at the end of the chapter.

5.7.5 Trimming

Trimming curves or entities is a very useful function provided by all CAD/CAM systems. Trimming can truncate or extend a curve. Trimming is mathematically identical to segmentation covered in the previous section. The only difference between the two is that the result of trimming a curve is only one segment of the



(*a*) Truncated curve

(*b*) Extended curve

FIGURE 5-59
Reparametrization of a trimmed curve.

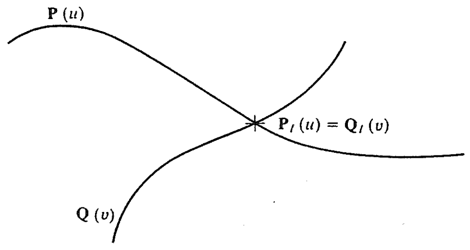


FIGURE 5-60
Intersection of two parametric curves in space.

curve bounded by the trimming boundaries. All the mathematical treatment of the segmentation applies here. The trimming function requires evaluating the equation of the desired segment of the curves, deleting the original curve, and then storing and displaying the desired segment. Figure 5-59 shows the reparametrization of a trimmed curve. Extending general curves (Fig. 5-59*b*) such as cubic splines, Bezier curves, and B-spline curves may not be recommended because the curve behavior outside its original interval $u_0 < u < u_m$ may not be predictable.

5.7.6 Intersection

The intersection point P_I (see Fig. 5-60) of two parametric curves $\mathbf{P}(u)$ and $\mathbf{Q}(v)$ in three-dimensional space requires the solution of the following equation in the parameters u and v:

$$\mathbf{P}(u) - \mathbf{Q}(v) = \mathbf{0} \tag{5.134}$$

This equation represents three scalar equations that take the polynomial nonlinear form generally in the two parameter unknowns. One way to find u and v is to solve the X and Y components of the equation simultaneously, that is,

$$\begin{aligned} P_x(u) - Q_x(v) &= 0 \\ P_y(u) - Q_y(v) &= 0 \end{aligned} \tag{5.135}$$

and then use the Z component, $P_z(u) - Q_z(v) = 0$, to verify the solution.

The roots of Eq. (5.134) or (5.135) can be found by numerical analysis methods such as the Newton-Raphson method. This method requires an initial guess which can be determined interactively. The user is usually asked by the CAD/CAM software to digitize the two curves whose intersection is required. These digitizes could be possibly changed to an initial guess to start the solution. If more than one intersection point exists, the user must digitize close to the desired intersection point.

Example 5.24. Find the intersection point of two lines.

Solution. Assume the equations of the lines are

$$\mathbf{P}(u) = \mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0) \tag{5.136}$$

$$\mathbf{Q}(v) = \mathbf{Q}_0 + v(\mathbf{Q}_1 - \mathbf{Q}_0) \tag{5.137}$$

The intersection point is given by the solution of the equation

$$\mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0) = \mathbf{Q}_0 + v(\mathbf{Q}_1 - \mathbf{Q}_0) \tag{5.138}$$

Taking the dot product of the above equation with the vector $(\mathbf{Q}_0 \times \mathbf{Q}_1)$ we obtain

$$(\mathbf{Q}_0 \times \mathbf{Q}_1) \cdot [\mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0)] = 0$$

since $(\mathbf{Q}_0 \times \mathbf{Q}_1)$ is perpendicular to the plane of $\mathbf{Q}_0$ and $\mathbf{Q}_1$ and hence to the two vectors. Solving the above equation for u gives

$$u = -\frac{(\mathbf{Q}_0 \times \mathbf{Q}_1) \cdot \mathbf{P}_0}{(\mathbf{Q}_0 \times \mathbf{Q}_1) \cdot (\mathbf{P}_1 - \mathbf{P}_0)} \tag{5.139}$$

Thus, u is given in terms of the endpoints of the two lines. Substituting Eq. (5.139) into (5.136) results in the coordinates of the intersection point which are usually calculated in terms of the MCS of the given model or part. Equation (5.138) could have been solved for v instead of u to give

$$v = -\frac{(\mathbf{P}_0 \times \mathbf{P}_1) \cdot \mathbf{Q}_0}{(\mathbf{P}_0 \times \mathbf{P}_1) \cdot (\mathbf{Q}_1 - \mathbf{Q}_0)}$$

The above approach can be extended to find the intersection of a line given by $\mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0)$ and a curve given by $\mathbf{Q}(v)$ such as a Bezier or B-spline curve. In this case u should be eliminated from the two equations to obtain

$$(\mathbf{P}_0 \times \mathbf{P}_1) \cdot \mathbf{Q}(v) = 0$$

5.7.7 Transformation

Manipulation or transformation of geometric entities during model construction or creating the model database offers a distinct advantage of CAD/CAM technology over traditional drafting methods. With transformation techniques, the designer can project, translate, rotate, mirror, and scale various entities. Section 5.3 shows how two-and-a-half-dimensional objects can be constructed easily using a "project" command. It also shows how the "mirror" command helps construct symmetric objects. Transformation is also useful in studying the motion of mechanisms, robots, and other objects in space. Animation techniques (Chap. 13) may be based on transforming an object into various positions and then replaying these positions continuously.

Simple transformations such as those mentioned above are usually referred to as rigid-body transformations. They can be directly applied to parametric representations of geometric models. To translate a model of a car, all points, lines, curves, and surfaces forming the model must be translated.

Homogeneous transformation, as discussed in Chap. 9, offers a concise matrix form to perform all rigid-body transformations as matrix multiplications, which is a desired feature from the software development point of view. Equation (3.3) gives the general homogeneous transformation matrix $[T]$. The proper choice of the elements of this matrix produces the various rigid-body transformations.

One of the main characteristics of rigid-body transformations is that geometric properties of curves, surfaces, and solids are invariant under these transformations. For example, originally parallel or perpendicular straight lines remain so after transformations. Intersection points of curves are transformed into the new intersection points, that is, one-to-one transformation. Chapter 9 discusses in more detail the subject of geometric transformation.

5.8 DESIGN AND ENGINEERING APPLICATIONS

This section presents some examples that show how theories covered in this chapter are applied to design and engineering problems. Consequently, it shows how existing CAD/CAM systems can be stretched beyond just using them for drafting, geometric modeling, and beyond what application modules of these systems offer. The reader is advised to work these examples on an actual CAD/CAM system.

Example 5.25. For the state of plane stress shown in Fig. 5-61, determine:

(*a*) The principal stresses.
(*b*) The state of stress exerted on plane *a-a*.

Solution. The solution of this typical problem is to use Mohr's circle which is a graphical method. The centralization of CAD/CAM databases can be used to simplify the use of the method. The designer follows the part setup procedure covered in Chap. 3 and utilizes the default view and construction plane. Then two lines are created, one horizontal and one vertical, passing through the origin of the MCS. These form the σ and τ axes (see Fig. 5-62). The stresses at the X and Y planes can be used to form the endpoints P_1 and P_2 of a line. These points have the coordinates $(-400, -600, 0)$ and $(800, 600, 0)$. Now a circle can be constructed whose center is the intersection of the line with the σ axis and whose diameter is that line. The intersection points P_3 and P_4 of the circle with the σ axis are the principal stresses. Their coordinates are $(\sigma_{\min}, 0, 0)$ and $(\sigma_{\max}, 0, 0)$, and $\sigma_{\max}$ and $\sigma_{\min}$ can simply be obtained using the "verify" command available on the CAD/CAM system. The orientation of the principal planes can be found by evaluating the angle α between the σ axis and the line using the "measure angle" command.

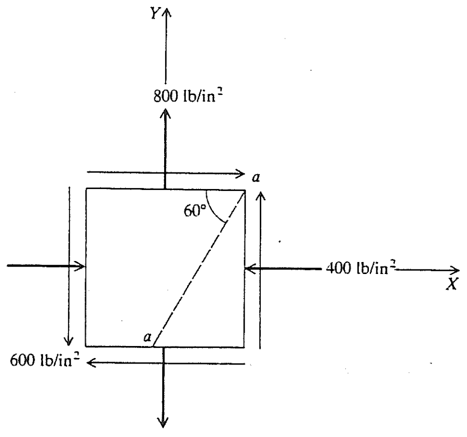
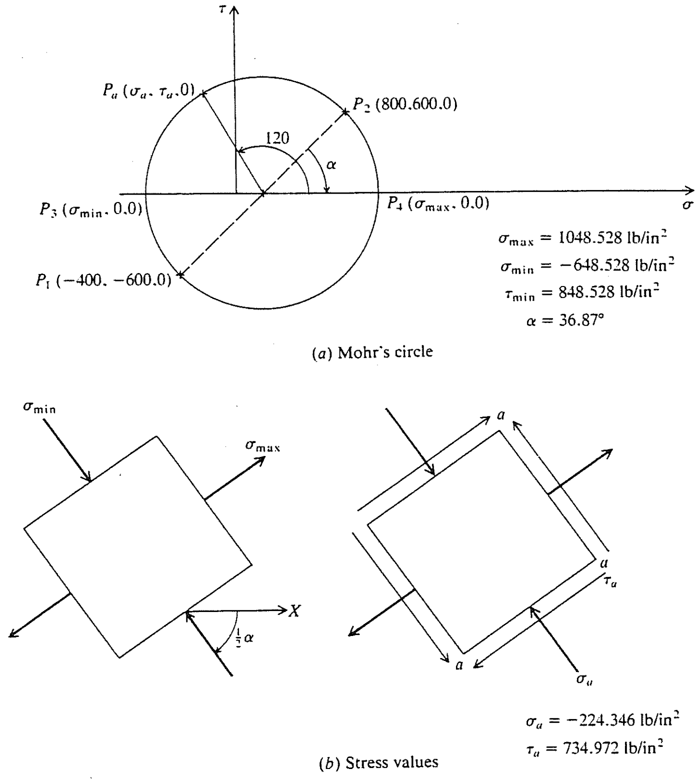


FIGURE 5-61
Stress state.


τ
P_a $(\sigma_a, \tau_a, 0)$
P_2 (800, 600, 0)
120
α
P_3 $(\sigma_{\min}, 0, 0)$
P_4 $(\sigma_{\max}, 0, 0)$
σ
P_1 (−400, −600, 0)
$\sigma_{\max}$ = 1048.528 lb/in²
$\sigma_{\min}$ = −648.528 lb/in²
$\tau_{\min}$ = 848.528 lb/in²
α = 36.87°
$\sigma_{\min}$
$\sigma_{\max}$
X
$\frac{1}{2}\alpha$
a
a
τ_a
a
σ_a
σ_a = −224.346 lb/in²
τ_a = 734.972 lb/in²

(*a*) Mohr's circle

(*b*) Stress values

FIGURE 5-62
Stress calculations via Mohr's circle.

To find the state of stress on plane *a-a*, a line that passes through the circle center and has an angle of 120° with the σ axis can be constructed. Its intersection point with the circle gives the state of stress on this plane. The complete solution is shown in Fig. 5-62.

The above procedure can be automated by writing a macro (refer to Chap. 15). The ease with which the designer can learn macro programming versus learning a full programming language makes this procedure attractive.

Example 5.26. The bar *AB* shown in Fig. 5-63 moves with its ends in contact with the horizontal and vertical walls. Assuming a plane motion, find the locus of point *C* on the bar for $D = 6$ inches and $L = 20$ inches. What is the locus if *C* is the centerpoint of the rod? Prove your answer.

Solution. The planning strategy to solve this problem begins by trying to generate enough points on the locus and then interpolating these points by a B-spline curve. Follow a similar part setup procedure as in the previous example. Create two lines,

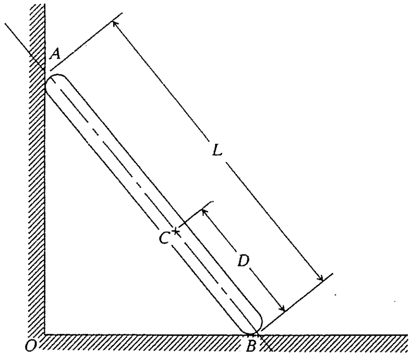


FIGURE 5-63
Bar AB in plane motion.

horizontal and vertical, passing through the origin of the MCS to represent the two walls. The bar AB is represented by a line of length 20 inches (see Fig. 5-64).

The extreme positions of point C are at distances D and $L - D$ from point O when the bar AB coincides with the vertical and horizontal walls respectively. To find any other point on the locus, let us start from the vertical position of AB and increment its motion until it becomes horizontal. Thus, we start when point A is at the location $(0, L, 0)$. Assuming that ten points are enough to generate the locus, point A has the coordinates $(0, L - m\,\Delta L, 0)$ where $\Delta L = L/10$ and $0 < m < 10$. To find point C for any position of the bar, create a circle with center at A and radius equal to L. The intersection point between the circle and the horizontal line gives point B and, therefore, the orientation of the bar. Point C can be located on AB using a point command with a parameter u value equal to D/L or $(1 - D/L)$ depending on whether point B or A is input first in the line command used to create the line. The algorithm to generate points on the locus can therefore be written as

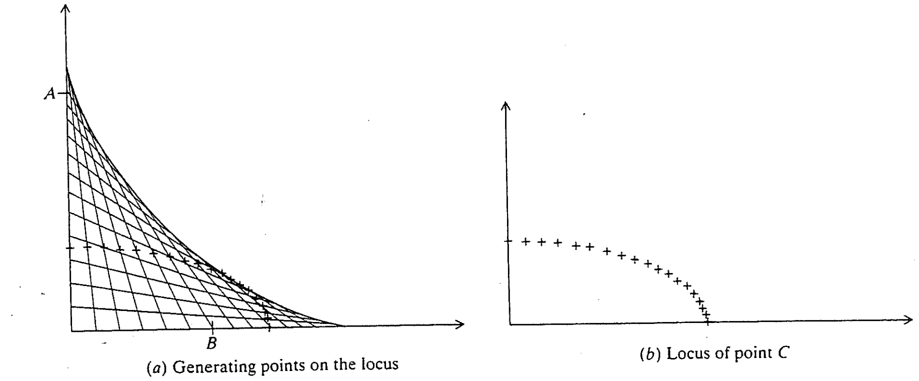
```
LOOP C1 = circle with $P_c = A$, $R = L$
         L1 = Line connecting $P_0 = A$ and $P_1$ = Intersection of C1 and horizontal line
         C = point at $u = 1 - D/L$
         A = point at $(0, L - m\,\Delta L, 0)$
         when m is equal to 10 exit
         Go to LOOP
```

Once all the points are generated, they are connected with a B-spline curve via a B-spline command. The locus is shown in Fig. 5-64. To facilitate the management of all graphics entities during construction, it is recommended that all possible aids such as layers, colors, and fonts be used (see Chap. 11). This method lends itself to macro programming which is useful in parametric design studies.

If point C is the center of AB, the locus becomes a circle with center at O and radius equal to the distance OC. Indeed, the locus of C is an ellipse in general with center at O and major and minor axes of lengths equal to $(L - D)$ and D if $D < L/2$ and vice versa if $D > L/2$. To prove this, consider Fig. 5-65, which gives the following coordinates of point C:

$$x = (L - D)\cos\theta \qquad y = D\sin\theta \qquad z = 0$$

which is an equation of an ellipse. If $D = L/2$, an equation of a circle results.



(a) Generating points on the locus

(b) Locus of point C

FIGURE 5-64
Locus of point C on bar AB.

Example 5.27. Minimize the function

$$\phi(x, y) = (x - 3)^2 + (y - 3)^2$$

subject to the constraints

$$\phi_1 \equiv x \geq 0$$

$$\phi_2 \equiv y \geq 0$$

$$\phi_3 \equiv x + y - 4 \leq 0$$

Solution. After following the part setup, construct the horizontal and vertical lines passing through the origin of the MCS. The above constraint set is the shaded triangle shown in Fig. 5-66. The function ϕ is a circle with a center at $C(3, 3, 0)$ and has a minimum value at that point ignoring the constraints. With the constraints, the minimum corresponds to the circle that is tangent to the line ϕ_3. The tangency point is the intersection point P between ϕ_3 and the line connecting the origin O and the center C. Verifying this point gives its coordinates as $(2, 2, 0)$. Therefore, $\phi_{\min} = \phi(2, 2) = 2$.

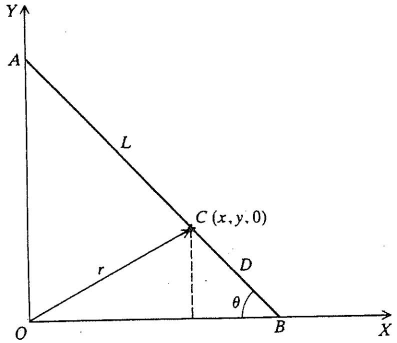


FIGURE 5-65
Analytic development of locus of point C.

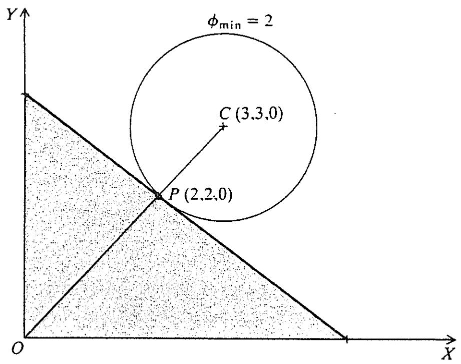


FIGURE 5-66
Minimum of the function $\phi(x, y)$.

Example 5.28. A four-bar mechanism is shown Fig. 5-67. The input angular velocity of the link AB is $\omega_1 = 200$ r/min clockwise. Point E, the center of the link CB, is connected to a valve that is not shown in the figure. The mechanism is to be redesigned such that:

Design criterion. The maximum linear velocity of point E, $v_{E,\max}$, must be greater than its current value by at least 5 inches/s.

Design constraints. (1) Only L and LL lengths can change and (2) AB must rotate the full 360°.

Solution. The solution to this design problem requires the velocity analysis of mechanisms. Either the relative velocity or instantaneous center concept can be utilized. The latter is used here because it lends itself to CAD/CAM techniques. The velocity analysis is shown in Fig. 5-68. The instantaneous center of the mechanism is the intersection point I of links AB and CD for any configuration of the mechanism. The velocity analysis gives

$$\begin{aligned} v_B &= \omega_1 L = \omega_2 L_1 \\ \omega_2 &= \omega_1 L / L_1 \\ v_E &= \omega_2 L_2 = \omega_1 L L_2 / L_1 \end{aligned} \tag{5.140}$$

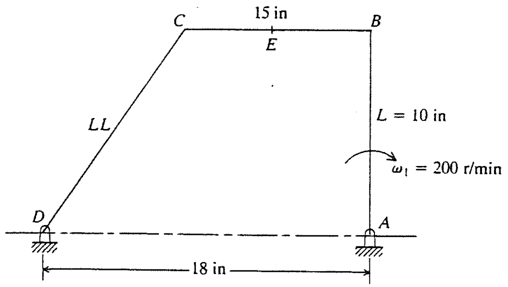


FIGURE 5-67
Four-bar mechanism.

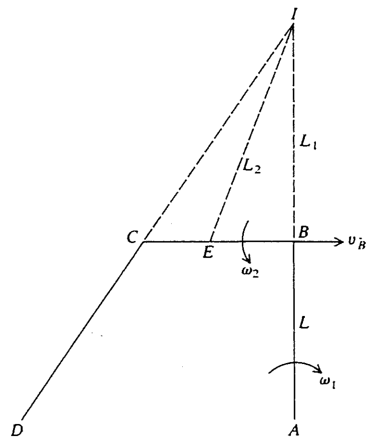


FIGURE 5-68
Velocity analysis of mechanism shown in Fig. 5-67.

The last equation gives the velocity of point E in terms of the input velocity and the lengths L, L_1, and L_2. To obtain $v_{E,\max}$, rotate the link AB around point A an angle θ, construct the mechanism in the new position, find L_1 and L_2, and substitute in Eqs. (5.140) to find v_E. Repeat for the admissible range of the angle θ. Connect the resulting points [each point has coordinates $(\theta, v_E, 0)$] with a B-spline curve. Find $v_{E,\max}$ from the curve. To construct the mechanism at any angle θ, rotate AB about A, the required angle. This defines point B. Point C is the intersection of two circles. The first has a center at B and a radius of 15 in and the second has a center at D and a radius equal to LL. Thus, the admissible range of angle of rotation (θ) of link AB for a certain set of dimensions is found when the above two circles do not intersect.

With the above strategy in mind, the designer can choose various values for L and LL in an attempt to achieve the design goal and the given design constraints. The major commands needed are "rotate, measure distance, B-spline" commands in addition to layer and utility (delete. . .) commands.

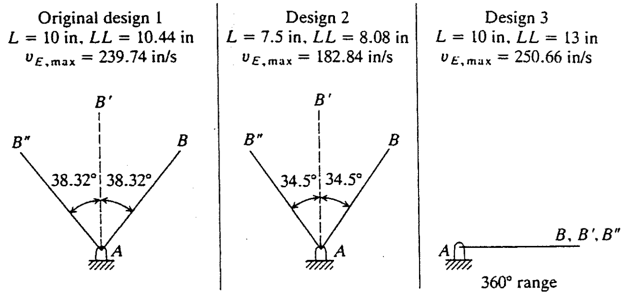

Original design 1
$L = 10$ in, $LL = 10.44$ in
$v_{E,\max} = 239.74$ in/s
Design 2
$L = 7.5$ in, $LL = 8.08$ in
$v_{E,\max} = 182.84$ in/s
Design 3
$L = 10$ in, $LL = 13$ in
$v_{E,\max} = 250.66$ in/s
B'
B"
B
38.32°
38.32°
A
B'
B"
B
34.5°
34.5°
A
A
B, B', B"
360° range

FIGURE 5-69
Results for three design iterations.

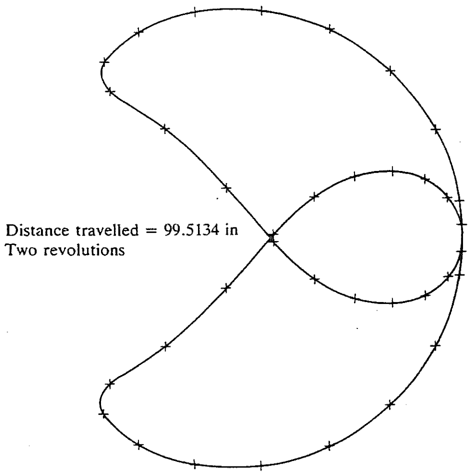


FIGURE 5-70
Locus of point E for design 3.

Figure 5-69 shows the results for two designs in addition to the original. Design 3 is the final design. Figure 5-70 shows the locus of point E for this design and Fig. 5-71 shows the velocity curves of E for the three designs. Notice that for the final design $v_{E,\max} = 250.656$ inch/s, which meets the design goal.

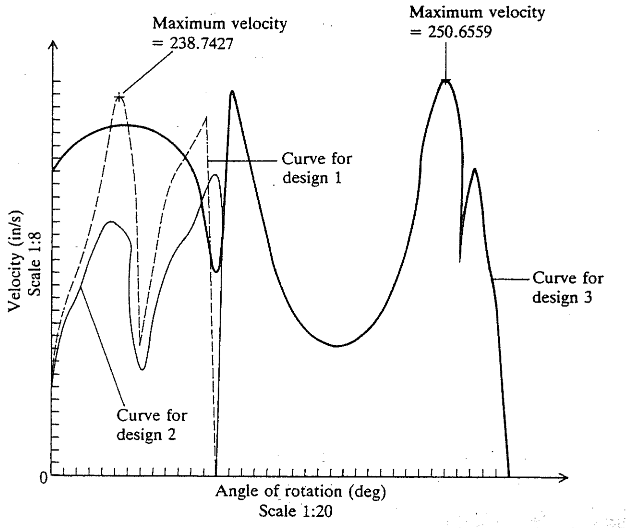
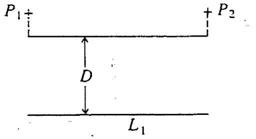


FIGURE 5-71
Velocity curves of designs 1, 2, and 3.

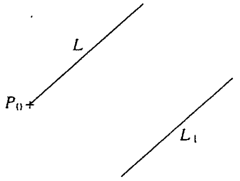
PROBLEMS

Part 1: Theory

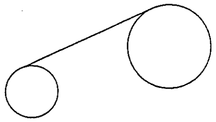
5.1. Find the equation and endpoints of each of the following lines:



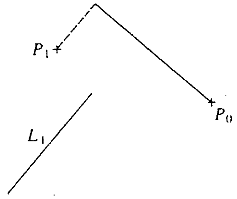
(*a*) Parallel to L_1 at distance D and bounded by points P_1 and P_2



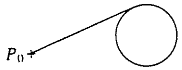
(*b*) Parallel to L_1, passes through P_0, and has length L



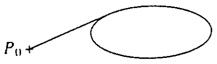
(*c*) Tangent to two given circles



(*d*) Perpendicular to L_1, passes through P_0, and bounded by P_1



(*e*) Tangent to a given circle from a given point P_0



(*f*) Tangent to a given ellipse from a given point P_0

5.2. Find the length of the common perpendicular to two skew lines.

5.3. Find the radius and the center of the following circles:
(*a*) Tangent to a given circle and a given line with a given radius.
(*b*) Tangent to two lines and passing through a given point.
(*c*) Passing through two points and tangent to a line.
(*d*) Tangent to a line, passing through a point, and with a given radius.

5.4. Find the center and the major and minor radii of an ellipse defined by two points and two slopes.

5.5. Find the intersection point of two tangent lines at two known points on an ellipse.

5.6. Find the tangent to an ellipse:
(*a*) At any given point on its circumference.
(*b*) From a point outside the ellipse.

5.7. As Prob. 5.6 but for a parabola.

5.8. Figure P5-8 shows a cubic spline curve consisting of two segments 1 and 2 with end conditions $\mathbf{P}_0$, $\mathbf{P}'_0$, $\mathbf{P}_1$, P'_{11}, and $\mathbf{P}_1$, $\mathbf{P}'_{12}$, P_2, $\mathbf{P}'_2$. The second subscripts in the tangent vectors at P_1 refer to the segment number. If $\mathbf{P}_{11} = \mathbf{R}$ and $\mathbf{P}_{12} = K\mathbf{R}$, where K is a constant, prove that the curve can only be C^1 continuous at P_1 and C^2 elsewhere.

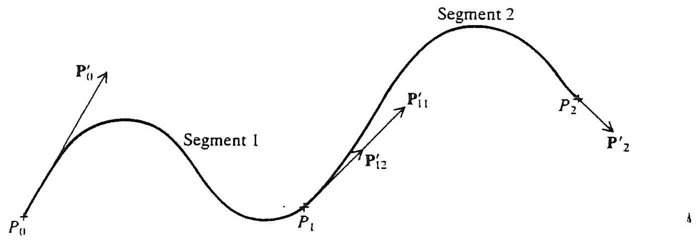


FIGURE P5-8

5.9. Find the normal vector to a cubic spline curve at any of its points.

5.10. For a cubic Bezier curve, carry a similar matrix formulation to a cubic spline. Compare $[M_B]$ and V for the two curves.

5.11. Find the condition that a cubic Bezier curve degenerates to a straight line connecting $\mathbf{P}_0$ and $\mathbf{P}_3$.

5.12. Derive a method by which you can force a Bezier curve to pass through a given point in addition to the starting and ending points of its polygon. Achieve that by changing the position of only one control point, say $\mathbf{P}_2$.

5.13. Investigate the statement "each segment of a B-spline curve is influenced by only k control points or each control point affects only k curve segments." Use $n = 3$, $k = 2$, 3, 4.

5.14. Find the equation of an open quadratic B-spline curve defined by five control points.

5.15. For an open cubic B-spline curve defined by n control points, carry a similar matrix formulation to a cubic spline. Compare $[M_S]$ and V with those of the cubic spline and Bezier curves.

Hint: Start with $n = 5$, that is, six control points, and then generalize the resulting $[M_S]$.

5.16. As Prob. 5.14 but for a closed B-spline curve.

5.17. Derive Eqs. (5.122) and (5.123).

5.18. Given a point Q and a parametric curve in the cartesian space, find the closest point P on the curve to Q.

Hint: Find P such that $(\mathbf{Q} - \mathbf{P})$ is perpendicular to the tangent vector.

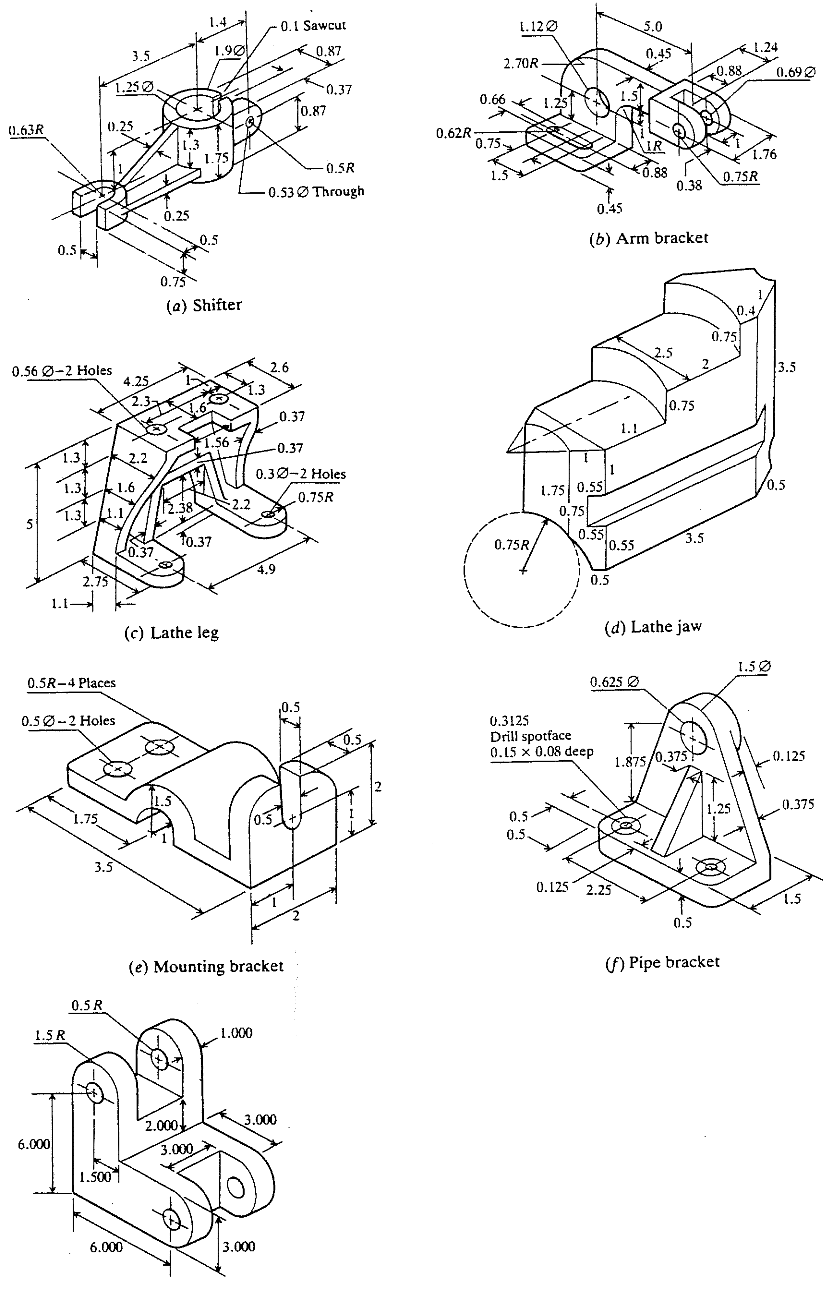
5.19. A cubic Bezier curve is to be divided by a designer into two segments. Find the modified polygon points for each segment.

Part 2: Laboratory

5.20. Obtain the three orthographic views (front, top, and right side) and a perspective view looking along the Z axis at a distance 20 inches from the parts shown in Fig. P5-20 (all dimensions in inches). Obtain final drawings of these views. Follow the model clean-up and documentation procedure on your particular CAD/CAM system. Obtain the standard six isometric views of each model. Clean up each view.

5.21. Using your CAD/CAM system, investigate the line, circle, ellipse, parabola, conics, cubic spline, Bezier curve, and B-spline curve commands and their related modifiers. Relate these modifiers to their theoretical background covered in this chapter.

5.22. Choose a mechanical element such as a gear and generate its geometric model.



(*a*) Shifter

(*b*) Arm bracket

(*c*) Lathe leg

(*d*) Lathe jaw

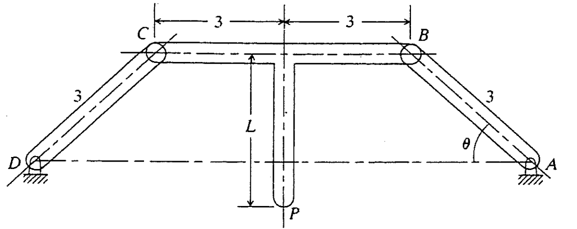
(*e*) Mounting bracket

(*f*) Pipe bracket

(*g*) Support bracket

FIGURE P5-20

5.23. In Fig. P5-23, the similar links AB and CD rotate about the fixed pins at A and D. Find the locus of point P for $0 < \theta < 180$ for $L = 7$ and 3.5 inches. How does L affect the locus?



All dimensions in inches

FIGURE P5-23

5.24. The slider crank mechanism shown in Fig. P5-24 is part of a machine. The sensor comes in contact with the centerpoint of the connecting rod BC only when this point is in its extreme position. The sensor has to change position to meet the following design requirements:

Design objective (goal): sensor should move 0.5 in outward.

Design constraints: $a + b$ must be kept to a minimum to minimize the mechanism weight (a and b have same cross section and same material).

Find the new lengths a and b.

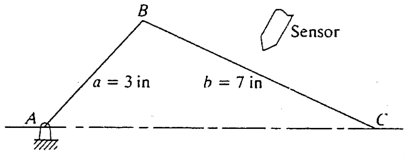


FIGURE P5-24

Part 3: Programming

5.25. Based on the theory presented in this chapter, write a program that takes the proper user input and display lines, circles, ellipses, and parabolas.

5.26. An ellipse at a general orientation can be generated and displayed in two ways:

(*a*) Use Eq. (5.37) to generate points ready for display directly.

(*b*) Use the equations $x_L = A \cos u$, $y_L = B \sin u$, $z_L = 0$ (see Fig. 5-28) and use the transformation

$$[x \quad y \quad z \quad 1]^T = \left[\begin{array}{ccc:c} \cos\alpha & \sin\alpha & 0 & x_c \\ -\sin\alpha & \cos\alpha & 0 & y_c \\ 0 & 0 & 1 & z_c \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x_L \\ y_L \\ 0 \\ 1 \end{bmatrix}$$

Program both ways and compare the CPU times of both cases.

5.27. As Prob. 5.26 but for a parabola [use Eq. (5.55)].

5.28. Write a program for a Hermite cubic spline connecting two points.

5.29. As Prob. 5.28 but for a cubic Bezier curve.

5.30. As Prob. 5.28 but for both open and closed B-spline curves.

BIBLIOGRAPHY

Abhyankar, S. S., and C. Bajaj: "Automatic Parametrization of Rational Curves and Surfaces 1: Conics and Conicoids," *Computer Aided Des.*, vol. 19, no. 1, pp. 11–14, 1987.

Barnhill, R. E., and R. F. Riesenfeld (Eds.): *Computer Aided Geometric Design*, Academic Press, New York, 1974.

Beer, F. P., and E. R. Johnston: *Mechanics of Materials*, McGraw-Hill, New York, 1981.

Boehm, W.: "A Survey of Curve and Surface Methods in CAGD," *Computer Aided Geometric Des. (CAGD) J.*, vol. 1, pp. 1–60, 1984.

Boehm, W.: "Triangular Spline Algorithms," *CAGD J.*, vol. 2, pp. 61–67, 1985.

Boehm, W.: "Curvature Continuous Curves and Surfaces," *CAGD J.*, vol. 2, pp. 313–323, 1985.

Boehm, W.: "Multivariate Spline Methods in CAGD," *Computer Aided Des. J.*, vol. 18, no. 2, pp. 102–104, 1986.

Boehm, W.: "Curvature Continuous Curves and Surfaces," *Computer Aided Des. J.*, vol. 18, no. 2, pp. 105–106, 1986.

Casen, S. H.: *Geometric Principles and Procedures for Computer Graphic Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

Chang, T. C., and R. A. Wysk: *An Introduction to Automated Process Planning Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1985.

Cohen, E., T. Lyche, and L. L. Schumaker: "Algorithms for Degree-Raising of Splines," *ACM Trans. on Graphics*, vol. 4, no. 3, pp. 171–181, 1985.

Cohen, E., and L. L. Schumaker: "Rates of Convergence of Control Polygons," *CAGD J.*, vol. 2, pp. 229–235, 1985.

Dakken, T.: "Finding Intersections of B-Spline Represented Geometrics Using Recursive Subdivision Techniques," *CAGD J.*, vol. 2, pp. 189–195, 1985.

Delvos, F. J.: "Bernoulli Functions and Periodic B-Splines," *Computing*, vol. 38, pp. 23–31, 1987.

Encarnacao, J., and E. G. Schlechtendahl: *Computer Aided Design; Fundamentals and System Architectures*, Springer-Verlag, New York, 1983.

Farin, G.: "Some Remarks on V2-Splines," *CAGD J.*, vol. 2, pp. 325–328, 1985.

Faux, I. D., and M. J. Pratt: *Computational Geometry for Design and Manufacture*, Ellis Horwood (John Wiley), Chichester, West Sussex, 1979.

Foley, J. D., and A. Van Dam: *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

Forrest, A. R.: "Curves and Surfaces for Computer-Aided Designs," Ph.D.Thesis, University of Cambridge, 1968.

Gardan, Y., and M. Lucas: *Interactive Graphics in CAD*, Kogan Page, London, 1984.

Giloi, W. K.: *Interactive Computer Graphics; Data Structures, Algorithms, Languages*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

Goldman, R. N.: "Vector Elimination: A Technique for the Implicitization, Inversion, and Intersection of Planar Parametric Rational Polynomial Curves," *CAGD J.*, vol. 1, pp. 327–356, 1984.

Goldman, R. N.: "The Method of Resolvents: A Technique for the Implicitization, Inversion, and Intersection of Non-Planar, Parametric, Rational Cubic Curves," *CAGD J.*, vol. 2, pp. 237–255, 1985.

Goldman, R. N., and D. J. Filip: "Conversion from Bezier Rectangles to Bezier Triangles," *Computer Aided Des. J.*, vol. 19, no. 1, pp. 25–27, 1987.

Goodman, T. N. T., and K. Unsworth: "Manipulating Shape and Producing Geometric Continuity in Spline Curves," *IEEE CG&A*, pp. 50–56, February 1986.

Groover, M. P., and E. W. Zimmers: *CAD/CAM; Computer-Aided Design and Manufacturing*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

Hagen, H.: "Geometric Spline Curves," *CAGD J.*, vol. 2, pp. 223–227, 1985.

Haug, E. J., and J. S. Arora: *Applied Optimum Design*, John Wiley, New York, 1979.

Lasser, D.: "Bernstein-Bezier Representation of Volumes," *CAGD J.*, vol. 2, pp. 145–149, 1985.

Lee, E. T. Y.: "Some Remarks Concerning B-Splines," *CAGD J.*, vol. 2, pp. 307–311, 1985.

Luzadder, W. J.: *Innovative Design with an Introduction to Design Graphics*, Prentice-Hall, Englewood Cliffs, N.Y., 1975.

Maccallum, K. J., and J. M. Zhang: "Curve Smoothing Techniques Using B-Splines," *The Computer J.*, vol. 29, no. 6, pp. 564–571, 1986.

Mortenson, M. E.: *Geometric Modeling*, John Wiley, New York, 1985.

Newman, W. M., and R. F. Sproull: *Principles of Interactive Computer Graphics*, 2d ed., McGraw-Hill, New York, 1979.

Nielson, G. M.: "Rectangular Splines," *IEEE CG&A*, pp. 35–40, February 1986.

Patterson, R. R.: "Projective Transformation of the Parameter of a Bernstein-Bezier Curve," *ACM Trans. on Graphics*, vol. 4, no. 4, pp. 276–290, 1985.

Piegl, L.: "Recursive Algorithms for the Representation of Parametric Curves and Surfaces," *Computer Aided Des. J.*, vol. 17, no. 5, pp. 225–229, 1985.

Piegl, L.: "Infinite Control Points, A Method for Representing Surfaces of Revolution Using Boundary Data," *IEEE CG&A*, pp. 45–55, March 1987.

Piegl, L.: "Interactive Data Interpolation by Rational Bezier Curves," *IEEE CG&A*, pp. 45–58, April 1987.

Rogers, D. F., and J. A. Adams: *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York, 1976.

Sablonniere, P.: "Bernstein-Bezier Methods for the Construction of Bivariate Spline Approximations," *CAGD J.*, vol. 2, pp. 29–36, 1985.

Sederberg, T. W.: "Degenerate Parametric Curves," *CAGD J.*, vol. 1, pp. 301–307, 1984.

Vandoni, C. E. (Ed.): *Eurographics '85*, Elsevier Science, New York, 1985.

Wilson, P. R.: "Conic Representations for Shape Description," *IEEE CG&A*, pp. 23–30, April 1987.

CHAPTER 6

TYPES AND MATHEMATICAL REPRESENTATIONS OF SURFACES

6.1 INTRODUCTION

Shape design and representation of complex objects such as car, ship, and airplane bodies as well as castings cannot be achieved utilizing wireframe modeling covered in the previous chapter. In such cases, surface modeling must be utilized to describe objects precisely and accurately. Due to the richness in information of surface models, their use in engineering and design environments can be extended beyond just geometric design and representation. They are usually used in various applications such as calculating mass properties, checking for interference between mating parts, generating cross-sectioned views, generating finite element meshes, and generating NC tool paths for continuous path machining.

Creating surfaces in general has some quantitative data, such as a set of points and tangents, and some qualitative data, such as intuition of the desired shape and smoothness. Quantitative and qualitative data can be thought of as hard and soft data respectively. Surface formulation must provide the designer with the flexibility to use both types of data in a simple form that is suitable for interactive use. Similar to curves, available surface techniques can interpolate or approximate the given hard data. As will be seen in this chapter, the Bezier surface is a form of approximation and the B-spline surface is a form of interpolation.

In addition to using surfaces to model geometric objects, they can also be used to fit experimental data, tables of numbers, and discretized solutions of differential equations. In all these cases, the multidimensional surface problem arises. The problem can be stated as follows. Given positional data (points in three-dimensional space) and a variable value at each point, how can the surface representing the variable distribution be constructed? Consider, for example, constructing the pressure surface, or distribution, on an oblique airplane wing or constructing the stress distribution in a mechanical part. The construction of these four-dimensional surfaces usually involves finding the proper contours of the given variable (pressure or stress) and then displaying them in a color-coded form.

The choice of the surface form depends upon the application; that is, there is no single solution for all problems. For example, the surface form used to model a car body may not be adequate to model the human heart. The choice of surface form may depend on manufacturing methods needed to produce the surface. It is usually preferred to choose a surface that can be produced using three-axis machining instead of five-axis machining to reduce the manufacturing cost. However, all surface forms must be easy to differentiate to determine surface tangents, normals, and curvatures. Polynomial functions are an obvious choice. Polynomials of higher orders are not appropriate from a surface design point of view due to their large number of coefficients which may make it difficult to control the resulting surface or may introduce unwanted oscillations in the surface. For most practical surface applications, cubic polynomials are sufficient.

It is desirable, if possible, to choose a surface mathematical description or form that is applicable to both surface design and surface representation. Surface representation involves using given data to display and view the surface. It can take place in *n*-space, as the pressure surface problem cited above shows. Surface design involves using key given data and making interactive changes to obtain the desired surface. Surface design usually takes place in three-dimensional space. In this chapter, we concern ourselves with surface design only.

The most obvious and inefficient way to describe a surface is to list sufficient points on it. This approach is cumbersome and cannot be used to derive any surface properties. Instead, it is common to employ some form of interpolation scheme and use fewer points. It is even more convenient if analytic forms of surfaces exist all the time. When analytic forms are not available, surfaces are defined in patches that are connected together similarly to curves, which can be defined in a piecewise manner.

Surface creation on existing CAD/CAM systems usually requires wireframe entities as a start. A system might request two boundary entities (rails of the surface) to create a ruled surface or might require one entity (generator curve) to create a surface of revolution. All analytic and synthetic wireframe entities covered in the previous chapter can be used to generate surfaces. In order to visualize surfaces on a graphics display, a mesh, say $m \times n$ in size, is usually displayed. The mesh size is controllable by the user. CAD/CAM systems must provide their users with a wide range of surfaces and surface manipulations as discussed in this chapter.

This chapter covers both theoretical and practical aspects of surfaces. Throughout the chapter, related issues to constructing surfaces on CAD/CAM systems are covered. Sections 6.2 and 6.3 are dedicated to some of these issues which may be helpful to users. Sections 6.4 to 6.7 cover the mathematical representations of most popular surfaces. Section 6.8 applies the chapter material to design and engineering applications.

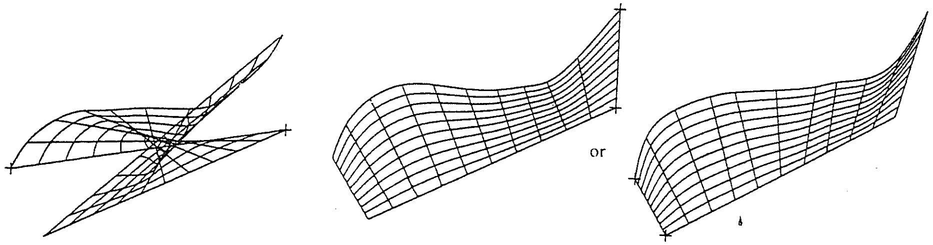
6.2 SURFACE MODELS

A surface model of an object is a more complete and less ambiguous representation than its wireframe model. It is also richer in its associated geometric contents, which makes it more suitable for engineering and design applications. A surface model can be used, for example, to drive the cutter of a machine tool while a wireframe model cannot. Surface models take the modeling of an object one step beyond wireframe models by providing information on surfaces connecting the object edges. Typically, a surface model consists of wireframe entities that form the basis to create surface entities. Surface description is usually tackled as an extension to the wireframe representation. Analytic and synthetic surface entities are available and provided by most CAD/CAM systems.

Surface modeling has been developing rapidly due to the shortcomings and inconveniences of wireframe modeling. The former is considered an extension of the latter. In general, a wireframe model can be extracted from a surface model by deleting or blanking all surface entities. Databases of surface models are centralized and associative. Thus, manipulating surface entities in one view is automatically reflected in other views. These entities can also be subjected to three-dimensional geometric transformations.

Despite their similar look, there is a fundamental difference between surface and solid models. Surface models define only the geometry of their corresponding objects. They store no information regarding the topology of these objects. As an example, if there are two surface entities that share a wireframe entity (edge), neither the surfaces nor the entity store such information. There is also a fundamental difference between creating surface and solid models. To create a surface model, the user begins by constructing wireframe entities and then connecting them appropriately with the proper surface entities. For solids based on boundary representation, either faces, edges, and vertices are created or solid primitives are input which are converted internally by the software to faces, edges, and vertices. Refer to the next chapter on solid modeling.

In constructing a surface model on a CAD/CAM system, the user should follow the modeling guidelines discussed in Chap. 3. All the design tools provided by CAD/CAM systems and covered in Part IV of the book are applicable to surface models. Only geometric modifiers are meaningless and cannot be applied to surface entities. From a practical point of view, it is more convenient to construct a surface model in an isometric view to enable clear display and visualization of its entities. Visualization of a surface is aided by the addition of artificial fairing lines (called mesh) which criss-cross the surface and so break it up into a network of interconnected patches. If wireframe entities are to be digitized to generate surfaces the user must do so in an ordered manner to ensure that the right surfaces are created. Figure 6-1 shows how the wrong ruled surface is created if the surface rails are digitized near the wrong ends. The +'s in the figure indicates the digitized locations. Another practical tip in constructing surface models is the change in mesh size of a surface entity. The most obvious way is to delete the surface entity and then reconstruct it with the desired mesh size. A better solution is to simply change the size of the mesh and then regenerate the surface display. Figure 6-2 shows surfaces of revolutions with a mesh size of 4 × 6 and Fig. 6-3 shows the regeneration of the surfaces with a 20 × 20 mesh



(*a*) Wrong digitized locations

(*b*) Proper digitized locations

FIGURE 6-1
Construction of improper and proper ruled surfaces.

size. It should be mentioned here that the finer the mesh size of surface entities in a model, the longer the CPU time to construct the entities and to update the graphics display, and the longer it takes to plot the surface model. Finally, some CAD/CAM systems do not permit their users to delete wireframe entities used to create surface entities unless the latter are deleted first.

Surface models have considerable advantages over wireframe models. They are less ambiguous. They provide hidden line and surface algorithms to add realism to the displayed geometry. Shading algorithms are only available for surface and solid models. From an application point of view, surface models can be utilized in volume and mass property calculations, finite element modeling, NC path generation, cross sectioning, and interference detections.

Despite the above-cited advantages, surface models have few disadvantages. Surface modeling does not lend itself to drafting background. It therefore requires more training and mathematical background on the user's part. Surface models are generally more complex and thus require more terminal and CPU time and computer storage to create than wireframe models. They are still ambiguous in some applications, as is the case when determining which of an object's surfaces define its volume. Surface models are sometimes awkward to create and may require unnecessary manipulations of wireframe entities. Consider the example of a surface with holes or cuts in it. Surface patches may have to be created with the aid of intermediate wireframe entities to create the full surface. Refer to Fig. 5-57 where the circle has to be divided to create the ruled surfaces.

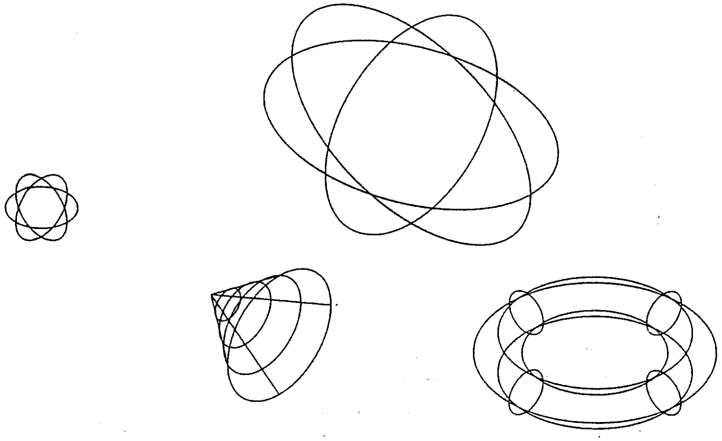


FIGURE 6-2
Surfaces of revolution with a 4 × 4 mesh size.

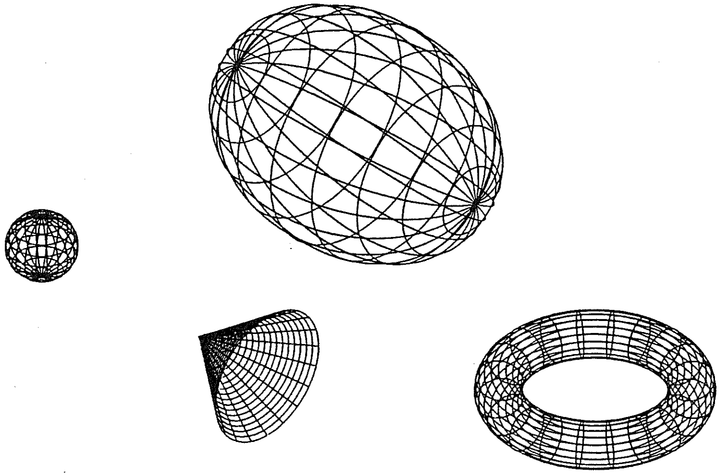
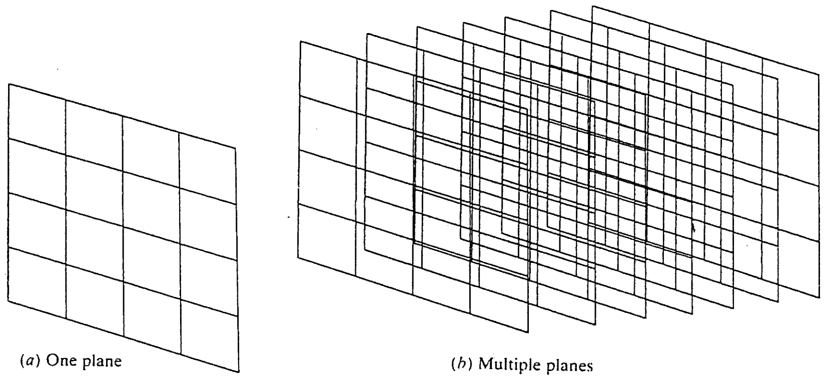


FIGURE 6-3
Surfaces of revolution with a 20 × 20 mesh size.

Surfaces and wireframes form the core of all existing CAD/CAM systems. Surface descriptions and capabilities are generalized to provide modeling flexibilities. For example, Gordon surfaces are considered alternatives, with generalization and improvements, to Bezier and Coons surfaces. Triangular Bezier and Coons patches are available for the case of arbitrarily located input data. Additional triangular patches also exist. As a result, new surface possibilities are created for those situations in which four-sided topology cannot be assumed.

6.3 SURFACE ENTITIES

Similar to wireframe entities, existing CAD/CAM systems provide designers with both analytic and synthetic surface entities. Analytic entities include plane surface, ruled surface, surface of revolution, and tabulated cylinder. Synthetic entities include the bicubic Hermite spline surface, B-spline surface, rectangular and triangular Bezier patches, rectangular and triangular Coons patches, and Gordon surface. The mathematical properties of some of these entities are covered in this chapter for two purposes. First, it enables users to correctly choose the proper surface entity for the proper application. For example, a ruled



(*a*) One plane

(*b*) Multiple planes

FIGURE 6-4
Plane surface.

surface is a linear surface and does not permit any twist while a B-spline surface is a general surface. Second, users will be in a position to better understand CAD/CAM documentation and the related modifiers to each surface entity command available on a system. The following are descriptions of major surface entities provided by CAD/CAM systems:

1. *Plane surface.* This is the simplest surface. It requires three noncoincident points to define an infinite plane. The plane surface can be used to generate cross-sectional views by intersecting a surface model with it, generate cross sections for mass property calculations, or other similar applications where a plane is needed. Figure 6-4 shows a plane surface.

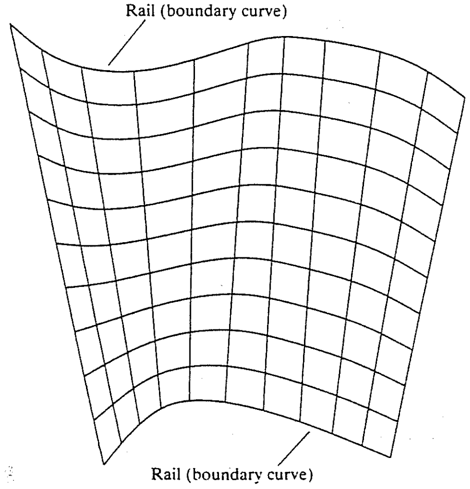


FIGURE 6-5
Ruled surface.

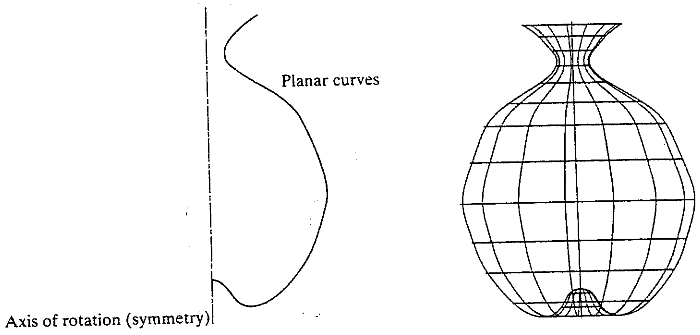


FIGURE 6-6
Surface of revolution.

2. *Ruled (lofted) surface.* This is a linear surface. It interpolates linearly between two boundary curves that define the surface (rails). Rails can be any wireframe entity. This entity is ideal to represent surfaces that do not have any twists or kinks. Figure 6-5 gives some examples.
3. *Surface of revolution.* This is an axisymmetric surface that can model axisymmetric objects. It is generated by rotating a planar wireframe entity in space about the axis of symmetry a certain angle (Fig. 6-6).
4. *Tabulated cylinder.* This is a surface generated by translating a planar curve a certain distance along a specified direction (axis of the cylinder) as shown in Fig. 6-7. The plane of the curve is perpendicular to the axis of the cylinder. It

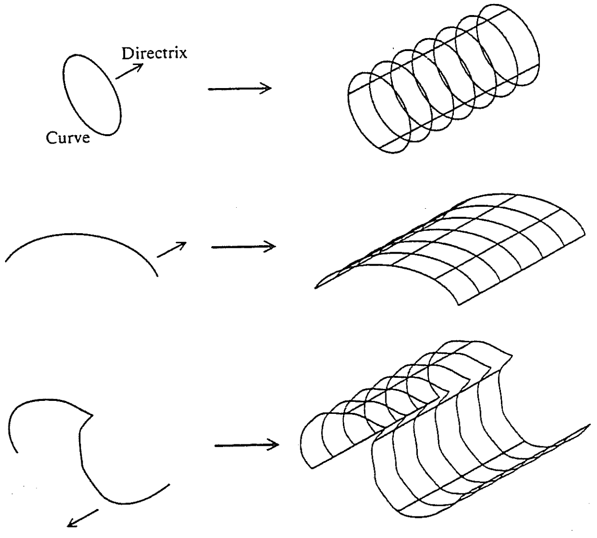


FIGURE 6-7
Tabulated cylinder.

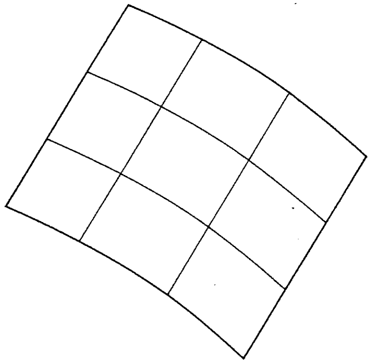
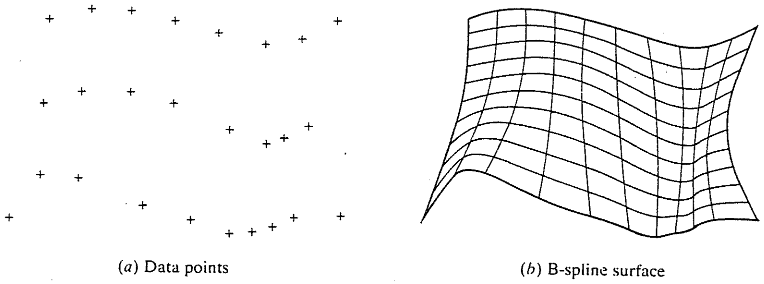


FIGURE 6-8
Bezier surface.

is used to generate surfaces that have identical curved cross sections. The word "tabulated" is borrowed from the APT language terminology.

5. *Bezier surface.* This is a surface that approximates given input data. It is different from the previous surfaces in that it is a synthetic surface. Similarly to the Bezier curve, it does not pass through all given data points. It is a general surface that permits, twists, and kinks.(Fig. 6-8). The Bezier surface allows only global control of the surface.
6. *B-spline surface.* This is a surface that can approximate or interpolate given input data (Fig. 6-9). It is a synthetic surface. It is a general surface like the Bezier surface but with the advantage of permitting local control of the surface.
7. *Coons patch.* The above surfaces are used with either open boundaries or given data points. The Coons patch is used to create a surface using curves that form closed boundaries (Fig. 6-10).
8. *Fillet surface.* This is a B-spline surface that blends two surfaces together (Fig. 6-11). The two original surfaces may or may not be trimmed.
9. *Offset surface.* Existing surfaces can be offset to create new ones identical in shape but may have different dimensions. It is a useful surface to use to speed up surface construction. For example, to create a hollow cylinder, the outer or inner cylinder can be created using a cylinder command and the other one can be created by an offset command. Offset surface command becomes very efficient to use if the original surface is a composite one. Figure 6-12 shows an offset surface.



(*a*) Data points

(*b*) B-spline surface

FIGURE 6-9
B-spline surface.

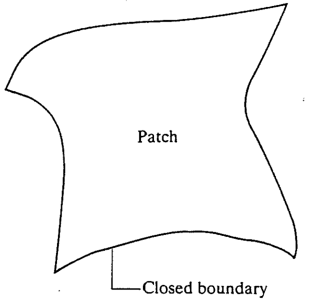


FIGURE 6-10
Coons patch.

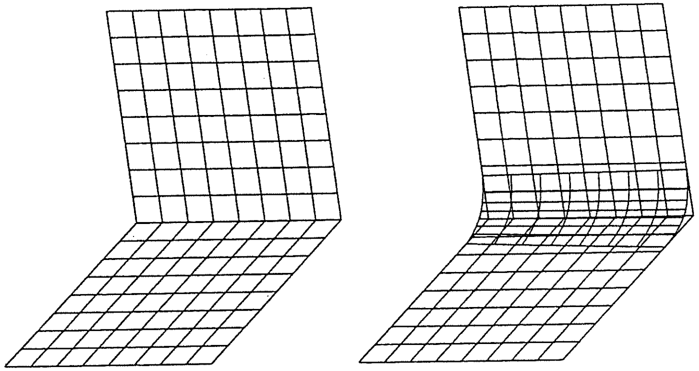


FIGURE 6-11
Fillet surface.

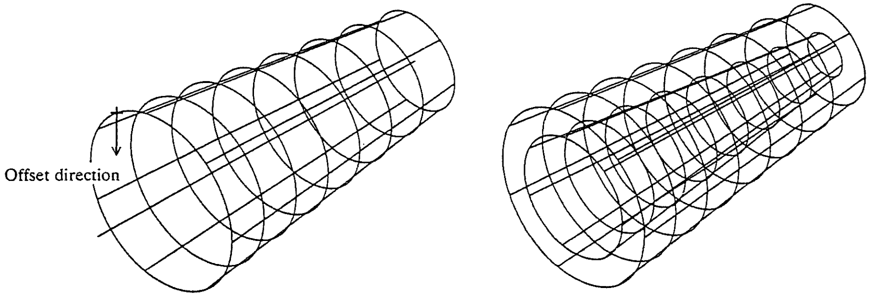


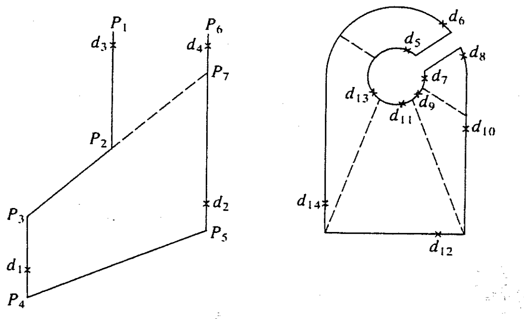
FIGURE 6-12
Offset surface.

Example 6.1. Create the surface model of the guide bracket shown in Fig. 5-2.

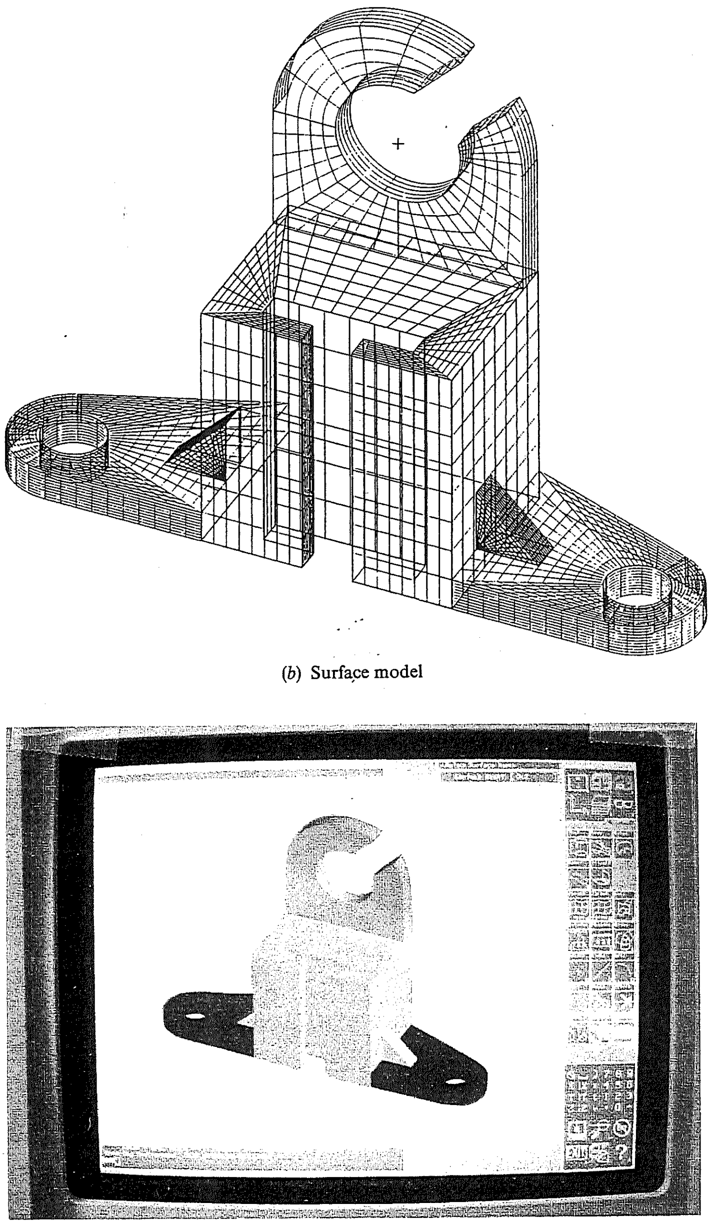
Solution. Before creating the surface model, the wireframe model of the bracket created in Example 5.1 is required and its database is assumed to be available for this example. A quick look at Fig. 5-2 reveals that ruled surfaces and tabulated cylinders are sufficient to construct the surface model. The following steps may be followed to construct the surface model:

1. Retrieve the wireframe model database of the bracket. It might be more practical to copy the wireframe model and use the copy to create the surfaces so that if the database is corrupted during surface creation, the original database can be copied again.
2. Select new layer(s) for surface entities to facilitate managing surface and wireframe entities.
3. Create surfaces on the right face of the model by using a ruled surface command. Referring to Fig. 6-13*a*, the line P_5P_6 is divided first into two entities at the intersection point (P_7) of lines P_2P_3 and P_5P_6. The ruled surface command is used twice to create two surfaces using lines P_3P_4 and P_5P_7 (identified by digitizes d_1 and d_2 in Fig. 6-13*a*) as rails for the first surface and lines P_1P_2 and P_6P_7 (d_3 and d_4 in the figure) as rails for the other surfaces.
4. Use a mirror or duplicate command to copy these surfaces to create surfaces on the left face of the model.
5. Create ruled surfaces on the front face of the top part of the model as shown in Fig. 6-13*a*. Five surfaces are constructed. These surfaces use entities d_5 and d_6; d_7 and d_8; d_9 and d_{10}; d_{11} and d_{12}; and d_{13} and d_{14} as rails. The small circle has to be divided into five arc segments to create these surfaces.
6. Create the surface of the hole shown in Fig. 6-13*a*. Use a tabulated cylinder command with the circle as the cylinder generator.
7. Follow a similar approach to construct all the surfaces of the model. During construction, the user encounters either dividing existing entities or creating new ones to define the appropriate rails of the various ruled surfaces. Figure 6-13*b* shows the completed surface model while Fig. 6-13*c* shows a shaded image of the surface model for better visualization.

The above example illustrates most of the experiences encountered in creating surface models on major CAD/CAM systems. The long construction time and user inconveniences due to dividing or creating entities make surface models



(*a*) Intermediate construction



(*b*) Surface model

(*c*) Shaded image

FIGURE 6-13
Surface model of the guide bracket of Example 5.1.

somewhat difficult to create and give more appeal to solid modeling. However, the major advantage of surface modeling is in its generality and capability to handle any kind of surface including sculptured surfaces.

6.4 SURFACE REPRESENTATION

Surface representation is considered, in many aspects, an extension of curve representation covered in the previous chapter. The nonparametric and parametric forms of curves can be extended to surfaces as will be covered later in this chapter. Similarly, the treatment of surfaces in computer graphics and CAD/CAM requires developing the proper equations and algorithms for both computation and programming purposes. Moreover, surface description is usually related to machining requirements to manufacture the surface. The surface description must successfully drive a tool to generate its path. Chapter 20 covers more on machining.

Surfaces can be described mathematically in three-dimensional space by nonparametric or parametric equations. There are several methods to fit nonparametric surfaces to a given set of data points. These fall into two categories. In the first, one equation is fitted to pass through all the points while in the second the data points are used to develop a series of surface patches that are connected together with at least position and first-derivative continuity. In both categories, the equation of the surface or surface patch is given by

$$\mathbf{P} = [x \quad y \quad z]^T = [x \ y \ \ f(x, y)]^T \tag{6.1}$$

where **P** is the position vector of a point on the surface as shown in Fig. 6-14. The natural form of the function $f(x, y)$ for a surface to pass through all the given data points is a polynomial, that is,

$$z = f(x, y) = \sum_{m=0}^{p} \sum_{n=0}^{q} a_{mn} x^m y^n \tag{6.2}$$

where the surface is described by an XY grid of size $(p + 1) \times (q + 1)$ points.

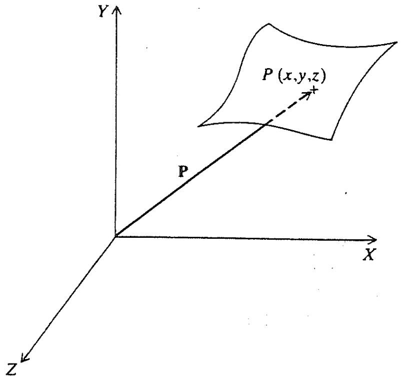


FIGURE 6-14
Point P on a nonparametric surface patch.

The nonparametric surface representation suffers from all the disadvantages, when compared with parametric surface representation, that nonparametric curves suffer from when compared with parametric curves. However, nonparametric surfaces do have some advantages when it comes to solving surface intersection problems, but these advantages do not warrant their use in CAD/CAM.

The parametric representation of a surface means a continuous, vector-valued function $\mathbf{P}(u, v)$ of two variables, or parameters, u and v, where the variables are allowed to range over some connected region of the uv plane and, as they do so, $\mathbf{P}(u, v)$ assumes every position on the surface. The function $\mathbf{P}(u, v)$ at certain u and v values is the point on the surface at these values. The most general way to describe the parametric equation of a three-dimensional curved surface in space is

$$\mathbf{P}(u, v) = [x \quad y \quad z]^T = [x(u, v) \quad y(u, v) \quad z(u, v)]^T,$$
$$u_{\min} \le u \le u_{\max},\ v_{\min} \le v \le v_{\max} \tag{6.3}$$

As with curves, Eq. (6.3) gives the coordinates of a point on the surface as the components of its position vector. It uniquely maps the parametric space (E^2 in u and v values) to the cartesian space (E^3 in x, y, and z), as shown in Fig. 6-15. The parametric variables u and v are constrained to intervals bounded by minimum and maximum values. In most surfaces, these intervals are [0, 1] for both u and v.

Equation (6.3) suggests that a general three-dimensional surface can be modeled by dividing it into an assembly of topological patches. A patch is considered the basic mathematical element to model a composite surface. Some surfaces may consist of one patch only while others may be a few patches connected together. Figure 6-16 shows a two-patch surface where the u and v values are [0, 1]. The topology of a patch may be rectangular or triangular as shown in Fig. 6-17. Triangular patches add more flexibility in surface modeling because they do not require ordered rectangular arrays of data points to create the surface as the rectangular patches do. Both triangular and rectangular Coons and Bezier patches are available.

Analogous to curves, there are analytic and synthetic surfaces. Analytic surfaces are based on wireframe entities and include the plane surface, ruled surface, surface of revolution, and tabulated cylinder. Synthetic surfaces are formed from a given set of data points or curves and include the bicubic, Bezier, B-spline, and Coons patches. There are few methods to generate synthetic surfaces such as the tensor product method, rational method, and blending method. The rational method develops rational surfaces which is an extension of rational curves. The blending method approximates a surface by piecewise surfaces.

The tensor product method is the most popular method and is widely used in surface modeling. Its widespread use is largely due to its simple separable nature involving only products of univariate basis functions, usually polynomials. It introduces no new conceptual complications due to the higher dimensionality of a surface over a curve. The properties of tensor product surfaces can easily be deduced from properties of the underlying curve schemes. The tensor product formulation is a mapping of a rectangular domain described by the u and v

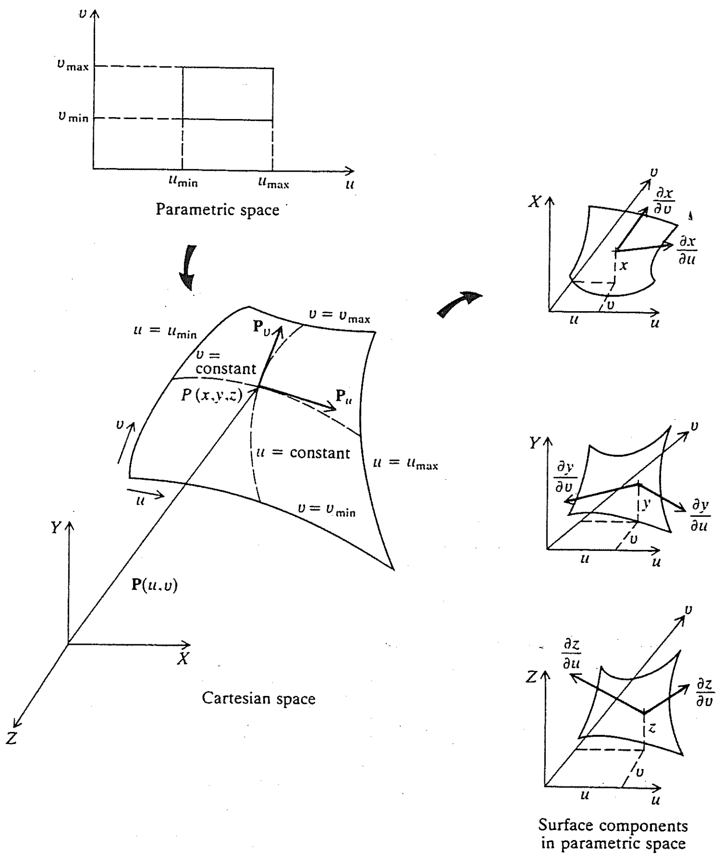


FIGURE 6-15
Parametric representation of a three-dimensional surface.

values; e.g., $0 \leq u \leq 1$ and $0 \leq v \leq 1$. Tensor product surfaces fit naturally onto rectangular patches. In addition, they have an explicit unique orientation (triangulation of a surface is not unique) and special parametric or coordinate directions associated with each independent parametric variable.

There is a set of boundary conditions associated with a rectangular patch. There are sixteen vectors and four boundary curves as shown in Fig. 6-18. The vectors are four position vectors for the four corner points $P(0, 0)$, $P(1, 0)$, $P(1, 1)$, and $P(0, 1)$; eight tangent vectors (two at each corner); and four twist vectors at the corner points (see below for the definition of a twist vector). The four boundary curves are described by holding one parametric variable fixed at one of its limiting values and allowing the other to change freely. The boundary curves are then defined by the curve equations $u = 0$, $u = 1$, $v = 0$, and $v = 1$.

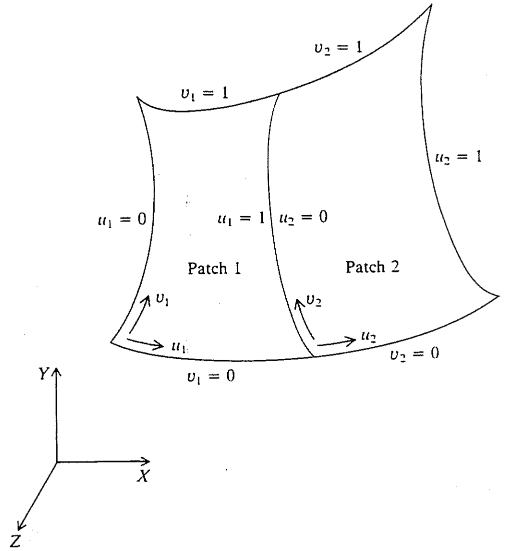
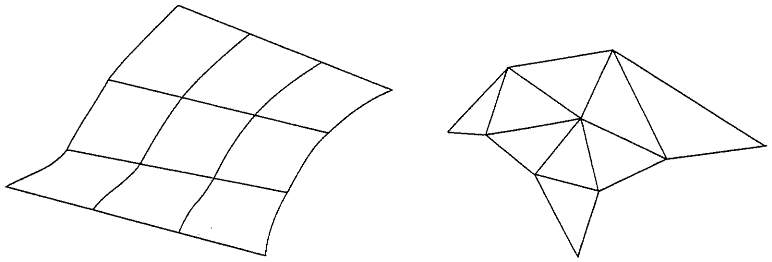


FIGURE 6-16
Two-patch parametric surface.

To generate curves on a surface patch, one can fix the value of one of the parametric variables, say u, to obtain a curve in terms of the other variable, v. By continuing this process first for one variable and then for the other using a certain set of arbitrary values in the permissible domain, a network of two parametric families of curves are generated. Only one curve of each family passes through any point $P(u, v)$ on the surface. The positive sense of any of these curves is the sense in which its nonfixed parameter increases. As mentioned earlier, the user can specify a mesh size, say $m \times n$, to display a surface on the graphics display. That mesh size determines the number of curves in each family. The curves of each family are usually equally spaced in the permissible interval of the corresponding parametric variable.



(a) Rectangular patches

(b) Triangular patches

FIGURE 6-17
Surfaces composed of rectangular and triangular patches.

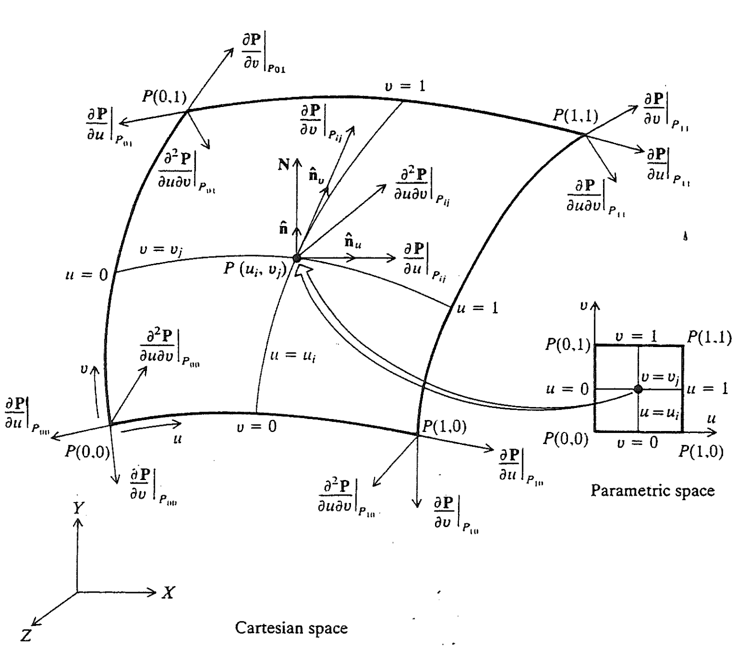

$\left.\frac{\partial \mathbf{P}}{\partial v}\right|_{P_{01}}$
$P(0,1)$
$v = 1$
$\left.\frac{\partial \mathbf{P}}{\partial u}\right|_{P_{01}}$
$\left.\frac{\partial \mathbf{P}}{\partial v}\right|_{P_{ij}}$
$P(1,1)$
$\left.\frac{\partial \mathbf{P}}{\partial v}\right|_{P_{11}}$
$\left.\frac{\partial^2 \mathbf{P}}{\partial u \partial v}\right|_{P_{01}}$
$\mathbf{N}$
$\hat{\mathbf{n}}_v$
$\left.\frac{\partial^2 \mathbf{P}}{\partial u \partial v}\right|_{P_{ij}}$
$\left.\frac{\partial \mathbf{P}}{\partial u}\right|_{P_{11}}$
$\left.\frac{\partial \mathbf{P}}{\partial u \partial v}\right|_{P_{11}}$
$\hat{\mathbf{n}}$
$\hat{\mathbf{n}}_u$
$v = v_j$
$\left.\frac{\partial \mathbf{P}}{\partial u}\right|_{P_{ij}}$
$u = 0$
$P\ (u_i,\ v_j)$
$u = 1$
v
$\left.\frac{\partial^2 \mathbf{P}}{\partial u \partial v}\right|_{P_{00}}$
$u = u_i$
$P(0,1)$
$v = 1$
$P(1,1)$
v
$u = 0$
$v = v_j$
$u = 1$
$\left.\frac{\partial \mathbf{P}}{\partial u}\right|_{P_{00}}$
$u = u_i$
u
$P(1,0)$
u
$v = 0$
$P(0,0)$
$P(0,0)$
$v = 0$
$P(1,0)$
$\left.\frac{\partial \mathbf{P}}{\partial u}\right|_{P_{10}}$
$\left.\frac{\partial \mathbf{P}}{\partial v}\right|_{P_{00}}$
Parametric space
$\left.\frac{\partial^2 \mathbf{P}}{\partial u \partial v}\right|_{P_{10}}$
$\left.\frac{\partial \mathbf{P}}{\partial v}\right|_{P_{10}}$
Y
X
Z
Cartesian space

FIGURE 6-18
A parametric surface patch with its boundary conditions.

Geometric surface analysis forms an important factor in using surfaces for other purposes than just displaying them. For example, knowing the tangent vectors to a surface enables driving a cutting tool along the surface to machine it and knowing the normal vectors to the surface provides the proper directions for the tool to approach and retract from the surface. Differential geometry plays a central role in the analysis of surfaces. The measurements of lengths and areas, the specification of directions and angles, and the definition of curvature on a surface are all formulated in differential terms. The parametric surface $P(u, v)$ is directly amenable to differential analysis. There are intrinsic differential characteristics of a surface such as the unit normal and the principal curvatures and directions which are independent of parametrization. These characteristics require introducing few parametric derivatives.

The tangent vector concept that is introduced in Chap. 5 can be extended to surfaces. The tangent vector at any point $P(u, v)$ on the surface is obtained by holding one parameter constant and differentiating with respect to the other. Therefore, there are two tangent vectors, a tangent to each of the intersecting curves passing through the point as shown in Fig. 6-18. These vectors are given by

$$\mathbf{P}_u(u, v) = \frac{\partial \mathbf{P}}{\partial u} = \frac{\partial x}{\partial u}\hat{\mathbf{i}} + \frac{\partial y}{\partial u}\hat{\mathbf{j}} + \frac{\partial z}{\partial u}\hat{\mathbf{k}}, \qquad u_{\min} \leq u \leq {}_{\max},\ v_{\min} \leq v \leq v_{\max} \tag{6.4}$$

along the v = constant curve, and

$$\mathbf{P}_v(u, v) = \frac{\partial \mathbf{P}}{\partial v} = \frac{\partial x}{\partial v}\hat{\mathbf{i}} + \frac{\partial y}{\partial v}\hat{\mathbf{j}} + \frac{\partial z}{\partial v}\hat{\mathbf{k}}, \qquad u_{\min} \leq u \leq u_{\max},\ v_{\min} \leq v \leq v_{\max} \tag{6.5}$$

along the u = constant curve. These two equations can be combined to give

$$\begin{bmatrix} \mathbf{P}_u \\ \mathbf{P}_v \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{bmatrix} \tag{6.6}$$

These components of each unit vector are shown in Fig. 6-15. If the dot product of the two tangent vectors given by Eqs. (6.4) and (6.5) is equal to zero at a point on a surface, then the two vectors are perpendicular to one another at that point. Figure 6-18 shows the tangent vectors at the corner points of a rectangular patch and at point P_{ij}. The notation $\partial\mathbf{P}/\partial u|_{P_{ij}}$, for example, means that the derivative is calculated at the point P_{ij} defined by $u = u_i$ and $v = v_j$. Tangent vectors are useful in determining boundary conditions for patching surfaces together as well as defining the motion of cutters along the surfaces during machining processes. The magnitudes and unit vectors of the tangent vectors are given by

$$\begin{aligned} |\mathbf{P}_u| &= \sqrt{\left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 + \left(\frac{\partial z}{\partial u}\right)^2} \\ |\mathbf{P}_v| &= \sqrt{\left(\frac{\partial x}{\partial v}\right)^2 + \left(\frac{\partial y}{\partial v}\right)^2 + \left(\frac{\partial z}{\partial v}\right)^2} \end{aligned} \tag{6.7}$$

and

$$\hat{\mathbf{n}}_u = \frac{\mathbf{P}_u}{|\mathbf{P}_u|} \qquad \hat{\mathbf{n}}_v = \frac{\mathbf{P}_v}{|\mathbf{P}_v|} \tag{6.8}$$

The slopes to a given curve on a surface can be evaluated (refer to Chap. 5), although they are less significant and seldom used in surface analysis.

The twist vector at a point on a surface is said to measure the twist in the surface at the point. It is the rate of change of the tangent vector $\mathbf{P}_u$ with respect to v or $\mathbf{P}_v$ with respect to u, or it is the cross (mixed) derivative vector at the point. Figure 6-19 shows the geometric interpretation of the twist vector. If we increment u and v by Δu and Δv respectively and draw the tangent vectors as shown, the incremental changes in $\mathbf{P}_u$ and $\mathbf{P}_v$ at point P, whose position vector is $\mathbf{P}(u, v)$, are obtained by translating $\mathbf{P}_u(u, v + \Delta v)$ and $\mathbf{P}_v(u + \Delta u, v)$ to P and forming the two triangles shown. The incremental rate of change of the two tangent vectors become $\Delta\mathbf{P}u/\Delta v$ and $\Delta\mathbf{P}v/\Delta u$, and the infinitesimal rate of change is given by the following limits:

$$\begin{aligned} \underset{\Delta v \to 0}{\text{Limit}} \frac{\Delta \mathbf{P}_u}{\Delta v} &= \frac{\partial \mathbf{P}_u}{\partial v} = \frac{\partial^2 \mathbf{P}}{\partial u\, \partial v} = \mathbf{P}_{uv} \\ \underset{\Delta u \to 0}{\text{Limit}} \frac{\Delta \mathbf{P}_v}{\Delta u} &= \frac{\partial \mathbf{P}_v}{\partial u} = \frac{\partial^2 \mathbf{P}}{\partial u\, \partial v} = \mathbf{P}_{uv} \end{aligned} \tag{6.9}$$

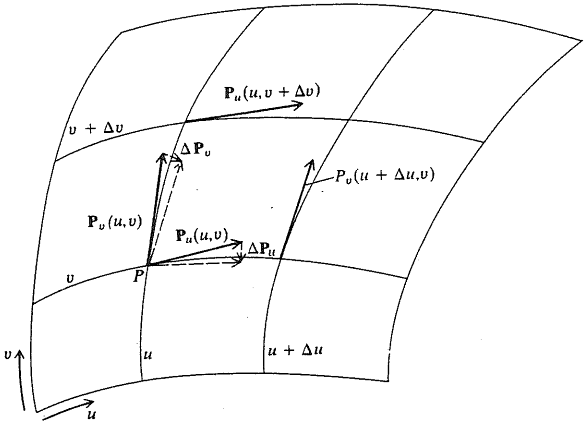


FIGURE 6-19
Geometric interpretation of twist vectors.

The twist vector can be written in terms of its cartesian components as

$$\mathbf{P}_{uv} = \left[\frac{\partial^2 x}{\partial u\, \partial v}\ \frac{\partial^2 y}{\partial u\, \partial v}\ \frac{\partial^2 z}{\partial u\, \partial v}\right]^T = \frac{\partial^2 x}{\partial u\, \partial v}\,\hat{\mathbf{i}} + \frac{\partial^2 y}{\partial u\, \partial v}\,\hat{\mathbf{j}} + \frac{\partial^2 z}{\partial u\, \partial v}\,\hat{\mathbf{k}},$$

$$u_{\min} \le u \le u_{\max},\ v_{\min} \le v \le v_{\max} \tag{6.10}$$

The twist vector depends on both the surface geometric characteristics and its parametrization. Due to the latter dependency, interpreting the twist vector in geometrical terms may be misleading since $\mathbf{P}_{uv} \neq 0$ does not necessarily imply a twist in a surface. For example, a flat plane is not a twisted surface. However, depending on its parametric equation, $\mathbf{P}_{uv}$ may or may not be zero.

The normal to a surface is another important analytical property. It is used to calculate cutter offsets for three-dimensional NC programming to machine surfaces, volume calculations, and shading of a surface model. The surface normal at a point is a vector which is perpendicular to both tangent vectors at the point (Fig. 6-18); that is,

$$\mathbf{N}(u, v) = \frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v} = \mathbf{P}_u \times \mathbf{P}_v \tag{6.11}$$

and the unit normal vector is given by

$$\hat{\mathbf{n}} = \frac{\mathbf{N}}{|\mathbf{N}|} = \frac{\mathbf{P}_u \times \mathbf{P}_v}{|\mathbf{P}_u \times \mathbf{P}_v|} \tag{6.12}$$

The order of the cross-product in Eq. (6.11) can be reversed and still defines the normal vector. The sense of $\mathbf{N}$, or $\hat{\mathbf{n}}$, is chosen to suit the application. In machining, the sense of $\hat{\mathbf{n}}$ is usually chosen so that $\hat{\mathbf{n}}$ points away from the surface being machined. In volume calculations, the sense of $\hat{\mathbf{n}}$ is chosen positive when pointing toward existing material and negative when pointing to holes in the part.

The surface normal is zero when $\mathbf{P}_u \times \mathbf{P}_v = \mathbf{0}$. This occurs at points lying on a cusp, ridge, or a self-intersecting surface. It can also occur when the two derivatives $\mathbf{P}_u$ and $\mathbf{P}_v$ are parallel, or when one of them has a zero magnitude. The latter cases correspond to a pathological parametrization which can be remedied.

The calculation of the distance between two points on a curved surface forms an important part of surface analysis. In general, two distinct points on a surface can be connected by many different paths, of different lengths, on the surface. The paths that have minimum lengths are analogous to a straight line connecting two points in euclidean space and are known as geodesics. Surface geodesics can, for example, provide optimized motion planning across a curved surface for numerical control machining, robot programming, and winding of coils around a rotor. The infinitesimal distance between two points (u, v) and $(u + du, v + dv)$ on a surface is given by

$$ds^2 = \mathbf{P}_u \cdot \mathbf{P}_u\, du^2 + 2\mathbf{P}_u \cdot \mathbf{P}_v\, du\, dv + \mathbf{P}_v \cdot \mathbf{P}_v\, dv^2 \tag{6.13}$$

Equation (6.13) is often called the first fundamental quadratic form of a surface and is written as

$$ds^2 = E\, du^2 + 2F\, du\, dv + G\, dv^2 \tag{6.14}$$

where

$$E(u, v) = \mathbf{P}_u \cdot \mathbf{P}_u \qquad F(u, v) = \mathbf{P}_u \cdot \mathbf{P}_v \qquad G(u, v) = \mathbf{P}_v \cdot \mathbf{P}_v \tag{6.15}$$

E, F, and G are the first fundamental, or metric, coefficients of the surface. These coefficients provide the basis for the measurement of lengths and areas, and the specification of directions and angles on a surface.

The distance between two points $P(u_a, v_a)$ and $P(u_b, v_b)$, shown in Fig. 6-20, is obtained by integrating Eq. (6.14) along a specified path $\{u = u(t), v = v(t)\}$ on the surface to give

$$S = \int_{t_a}^{t_b} \sqrt{Eu'^2 + 2Fu'v' + Gv'^2}\, dt \tag{6.16}$$

where $u' = du/dt$ and $v' = dv/dt$. The minimum S is the geodesic between the two points.

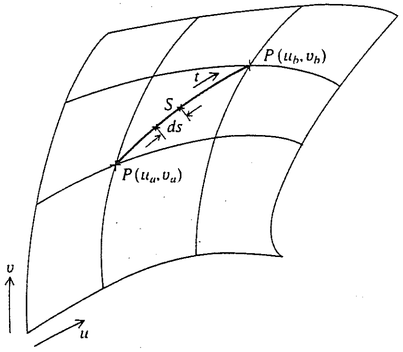


FIGURE 6-20
Surface geodesics.

The first fundamental form gives the distance element ds which lies in the tangent plane of the surface at $P(u, v)$ and, therefore, yields no information on how the surface curves away from the tangent plane at that point. To investigate the surface curvature, another distance perpendicular to the tangent plane at $P(u, v)$ is introduced and given by

$$\tfrac{1}{2}\, dh^2 = \hat{\mathbf{n}} \cdot \mathbf{P}_{uu}\, du^2 + 2\hat{\mathbf{n}} \cdot \mathbf{P}_{uv}\, du\, dv + \hat{\mathbf{n}} \cdot \mathbf{P}_{vv}\, dv^2 \qquad (6.17)$$

Similarly to Eq. (6.13), Eq. (6.17) is often called the second fundamental quadratic form of a surface and is written as

$$\tfrac{1}{2}\, dh^2 = L\, du^2 + 2M\, du\, dv + N\, dv^2 \qquad (6.18)$$

where

$$L(u, v) = \hat{\mathbf{n}} \cdot \mathbf{P}_{uu} \qquad M(u, v) = \hat{\mathbf{n}} \cdot \mathbf{P}_{uv} \qquad N(u, v) = \hat{\mathbf{n}} \cdot \mathbf{P}_{vv} \qquad (6.19)$$

L, M, and N are the second fundamental coefficients of the surface and form the basis for defining and analyzing the curvature of a surface.

A surface curvature at a point $P(u, v)$ is defined as the curvature of the normal section curve that lies on the surface and passes by the point. A normal section curve is the intersection curve of a plane passing through the normal $\hat{\mathbf{n}}$ at the point and the surface. Obviously, there can exist a family of planes, and therefore a family of normal section curves, that can contain $\hat{\mathbf{n}}$. The surface curvature at a point on a normal section curve given by the form $\{u = u(t),\ v = v(t)\}$ can be written as

$$\mathscr{K} = \frac{Lu'^2 + 2Mu'v' + Nv'^2}{Eu'^2 + 2Fu'v' + Gv'^2} \qquad (6.20)$$

and the radius of curvature at the point is $\rho = 1/\mathscr{K}$. The sense of curvature must be chosen. One convention is to choose the sign in Eq. (6.20) to give positive curvature for convex surfaces and negative curvature for concave surfaces.

Few types of surface curvatures exist. The gaussian curvature K and mean curvature H are defined by

$$\left.\begin{aligned} K &= \frac{LN - M^2}{EG - F^2} \\ H &= \frac{EN + GL - 2FM}{2(EG - F^2)} \end{aligned}\right\} \qquad (6.21)$$

Equation (6.20) gives the surface curvature in any direction at point $P(u, v)$. However, it can be used to obtain the principal curvatures which are the upper (maximum) and lower (minimum) bounds on the curvature at the point:

$$\begin{aligned} \mathscr{K}_{\max} &= H + \sqrt{H^2 - K} \\ \mathscr{K}_{\min} &= H - \sqrt{H^2 - K} \end{aligned} \qquad (6.22)$$

The gaussian and mean curvatures can be written in terms of Eqs. (6.22) as

$$\begin{aligned} K &= \mathscr{K}_{\max}\mathscr{K}_{\min} \\ H &= \tfrac{1}{2}(\mathscr{K}_{\max} + \mathscr{K}_{\min}) \end{aligned} \qquad (6.23)$$

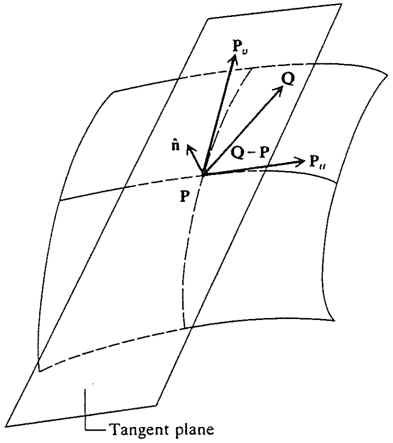


FIGURE 6-21
Tangent plane to a surface.

The gaussian curvature can be positive (as in a hill), negative (as in a saddle point), or zero (as in ruled or cylindrical surfaces) depending on the signs at $\mathscr{K}_{\max}$ and $\mathscr{K}_{\min}$. Surfaces that have zero gaussian curvature everywhere are called developable, that is, they can be laid flat on a plane without stretching, tearing, or distorting them. Some CAD/CAM systems display gaussian curvature contour maps to convey information about local surface shape.

From a practical point of view, the principal curvatures are of primary interest. For example, to machine a surface with a spherical cutter, it is important to ensure that the cutter radius is smaller than the smallest concave radius of curvature of the surface if gouging is to be avoided.

Analogous to the family of planes that are perpendicular to a surface and contain the normal $\hat{\mathbf{n}}$, a tangent plane to a surface at a point can be defined. It is the common plane on which all the tangent vectors to the surface at a point lie. To develop the equation of a tangent plane, consider a point q in a tangent plane to a given surface at a given point P, as shown in Fig. 6-21. The vectors $\mathbf{P}_u$, $\mathbf{P}_v$, and $\mathbf{Q} - \mathbf{P}$ lie on the plane. The normal $\hat{\mathbf{n}}$ is normal to any vector in the plane. Thus, we can write

$$\hat{\mathbf{n}} \cdot (\mathbf{Q} - \mathbf{P}) = (\mathbf{P}_u \times \mathbf{P}_v) \cdot (\mathbf{Q} - \mathbf{P}) = 0 \qquad (6.24)$$

which gives the equation of the tangent plane.

Example 6.2. The parametric equation of a sphere of radius R and a center at point $P_0(x_0, y_0, z_0)$ is given by

$$\begin{aligned} x &= x_0 + R\cos u \cos v \\ y &= y_0 + R\cos u \sin v \qquad -\frac{\pi}{2} \le u \le \frac{\pi}{2},\ 0 \le v \le 2\pi \\ z &= z_0 + R \sin u \end{aligned}$$

Find the sphere implicit equation.

Solution. Equation (6.1) suggests a useful and systematic way of developing the implicit equation of a surface from its parametric form. Implicit equations of surfaces are useful when solving surface intersection problems. Based on Eq. (6.1), 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

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

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

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

6.5 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. 5. 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.

6.5.1 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. 6-22. 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

$$\mathbf{P}(u, v)=\mathbf{P}_0+u(\mathbf{P}_1-\mathbf{P}_0)+v(\mathbf{P}_2-\mathbf{P}_0), \qquad 0\le u\le 1, 0\le v\le 1 \quad (6.25)$$

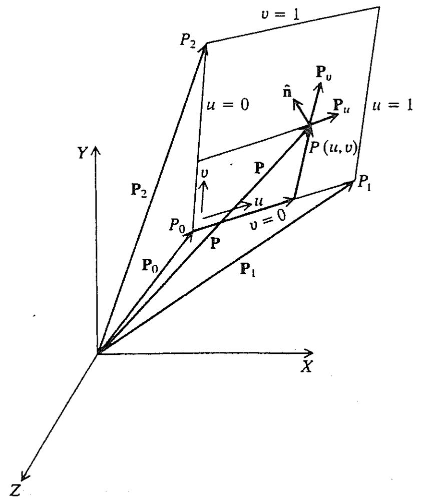


FIGURE 6-22
A plane patch defined by three points.

The above equation can be seen as the bilinear form of Eq. (5.11). Utilizing Eqs. (6.4) and (6.5), the tangent vectors at point P are

$$\mathbf{P}_u(u, v)=\mathbf{P}_1-\mathbf{P}_0 \qquad \mathbf{P}_v(u, v)=\mathbf{P}_2-\mathbf{P}_0, \qquad 0\le u\le 1, 0\le v\le 1 \quad (6.26)$$

and the surface normal is

$$\hat{\mathbf{n}}(u, v)=\frac{(\mathbf{P}_1-\mathbf{P}_0)\times(\mathbf{P}_2-\mathbf{P}_0)}{|(\mathbf{P}_1-\mathbf{P}_0)\times(\mathbf{P}_2-\mathbf{P}_0)|}, \qquad 0\le u\le 1, 0\le v\le 1 \quad (6.27)$$

which is constant for any point on the plane. As for the curvature of the plane, it is equal to zero [see Eq. (6.20)] because all the second fundamental coefficients of the plane are zeros [see Eq. (6.19)].

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. 6-23. Similar to the above case, the plane equation can be written as

$$\mathbf{P}(u, v)=\mathbf{P}_0+uL_u\hat{\mathbf{r}}+vL_v\hat{\mathbf{s}}, \qquad 0\le u\le 1, 0\le v\le 1 \quad (6.28)$$

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

$$\mathbf{P}(u, v)=\mathbf{P}_0+u(\mathbf{P}_1-\mathbf{P}_0)+vL_v\hat{\mathbf{r}}, \qquad 0\le u\le 1, 0\le v\le 1 \quad (6.29)$$

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 6-24 shows this case. The vector $\hat{\mathbf{n}}$ is normal to any vector in the plane. Thus,

$$(\mathbf{P}-\mathbf{P}_0)\cdot\hat{\mathbf{n}}=0 \quad (6.30)$$

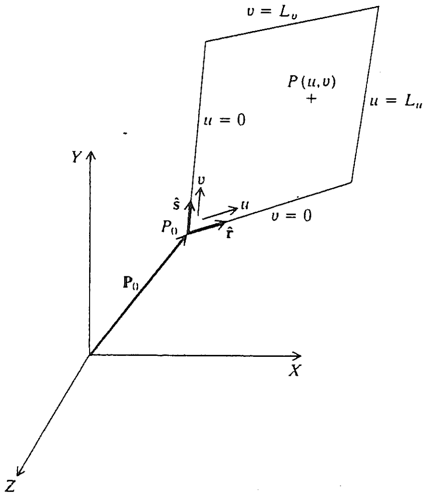


FIGURE 6-23
A plane patch defined by a point and two directions.

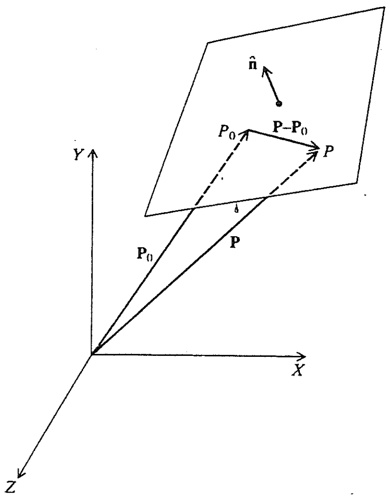


FIGURE 6-24
A plane patch passing through point P_0 and normal to $\hat{\mathbf{n}}$.

which is a nonparametric equation of the plane surface. A parametric equation can be developed by using Eq. (6.30) to generate two points on the surface which can be used with P_0 in Eq. (6.25). Planes that are perpendicular to the axes of a current WCS are special cases of Eq. (6.30). 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. 6-25, 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.

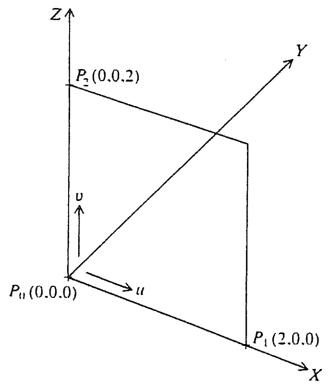


FIGURE 6-25
Plane surface construction.

Example 6.3. 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

$$\mathbf{P} = \mathbf{P}_0 + u\hat{\mathbf{r}} + v\hat{\mathbf{s}}, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{6.31}$$

This assumption can be made in the light of the database structure described above. From Fig. 6-26, 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

$$\mathbf{P} = \mathbf{Q} - D\hat{\mathbf{n}} \tag{6.32}$$

By using Eq. (6.31) in (6.32), we get

$$\mathbf{P}_0 + u\hat{\mathbf{r}} + v\hat{\mathbf{s}} = \mathbf{Q} - D\hat{\mathbf{n}} \tag{6.33}$$

Equation (6.33) can be rewritten in a matrix form as

$$\begin{bmatrix} r_x & s_x & n_x \\ r_y & s_y & n_y \\ r_z & s_z & n_z \end{bmatrix} \begin{bmatrix} u \\ v \\ D \end{bmatrix} = \begin{bmatrix} x_Q - x_0 \\ y_Q - y_0 \\ z_Q - z_0 \end{bmatrix} \tag{6.34}$$

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. (6.34). Solving Eq. (6.34) (see Chap. 5) gives the normal distance D and u and v, which can give the point P.

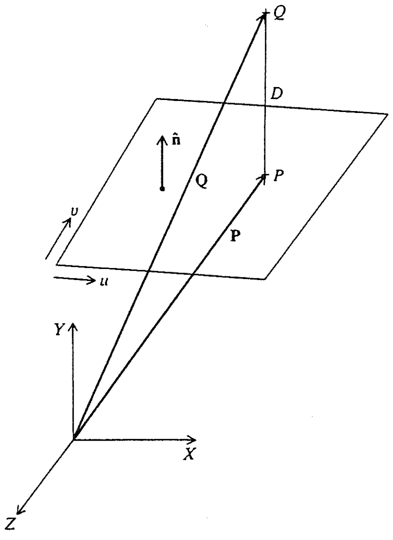


FIGURE 6-26
Minimum distance between a point and plane surface.

6.5.2 Ruled Surface

A ruled surface is generated by joining corresponding points on two space curves (rails) **G**(u) and **Q**(u) by straight lines (also called rulings or generators), as shown in Fig. 6-27. 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. 6.5.1 is considered the simplest of all ruled surfaces.

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 **G**(u) and **Q**(u) respectively. Using Eq. (5.11), the equation of the ruling becomes

$$\mathbf{P}(u_i, v) = \mathbf{G}_i + v(\mathbf{Q}_i - \mathbf{G}_i) \qquad (6.35)$$

where v is the parameter along the ruling. Generalizing Eq. (6.35) for any ruling, the parametric equation of a ruled surface defined by two rails is

$$\mathbf{P}(u, v) = \mathbf{G}(u) + v[\mathbf{Q}(u) - \mathbf{G}(u)] = (1 - v)\mathbf{G}(u) + v\mathbf{Q}(u),$$
$$0 \le u \le 1, 0 \le v \le 1 \qquad (6.36)$$

Holding the u value constant in the above equation produces the rulings given by Eq. (6.35) in the v direction of the surface, while holding the v value constant

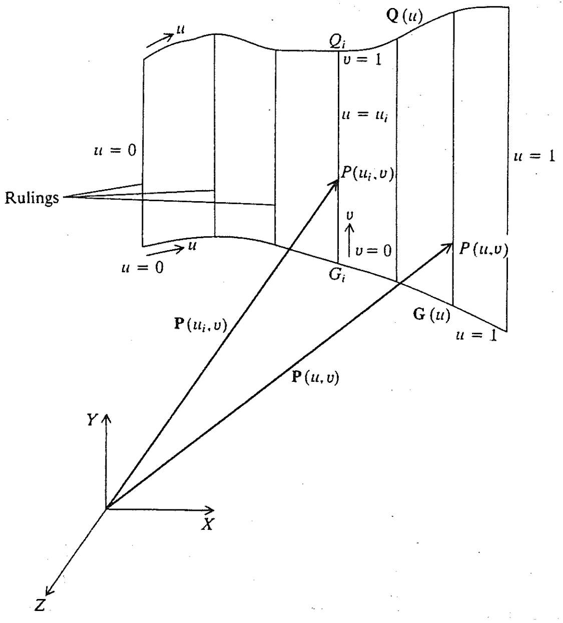


FIGURE 6-27
Parametric representation of a ruled surface.

yields curves in the u direction which are a linear blend of the rails. In fact, **G**(u) and **Q**(u) are **P**(u, 0) and **P**(u, 1) respectively. Therefore, the closer the value of v to zero, the greater the influence of **G**(u) and the less the influence of **Q**(u) on the v = constant curve. Similarly, the influence of **Q**(u) on the ruled surface geometry increases when the v value approaches unity (see Fig. 6-1).

Based on Fig. 6-27 and Eq. (6.35), it is now obvious why digitizing the wrong ends of the rails produces the undesirable ruled surface as shown in Fig. 6-1*a*. In addition, Eq. (6.36), together with Eqs. (6.15), (6.19), and (6.20), 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.

6.5.3 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. 6-28. 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. 6-28. 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

$$\mathbf{P}(u, v) = r_z(u) \cos v\hat{\mathbf{n}}_1 + r_z(u) \sin v\hat{\mathbf{n}}_2 + z_L(u)\hat{\mathbf{n}}_3,$$
$$0 \le u \le 1, 0 \le v \le 2\pi \qquad (6.37)$$

If we choose $z_L(u) = u$ for each point on the profile, Eq. (6.37) 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 $\mathbf{P}_L$ (see Fig. 6-28).

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. (6.37) is used to generate points on the surface.

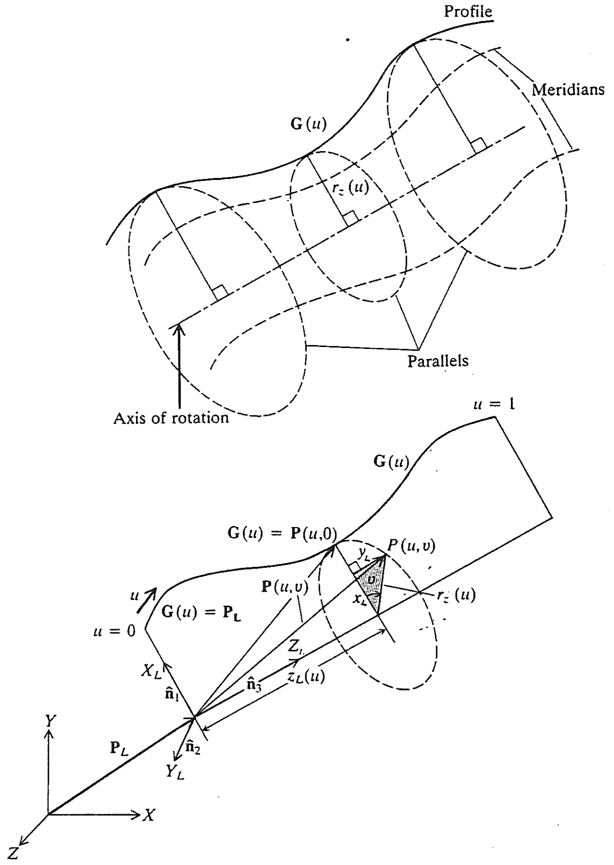


FIGURE 6-28
Parametric representation of a surface of revolution.

6.5.4 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. 6-29. 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

$$\mathbf{P}(u, v) = \mathbf{G}(u) + v\hat{\mathbf{n}}_v, \qquad 0 \le u \le u_{\max}, \; 0 \le v \le v_{\max} \tag{6.38}$$

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.

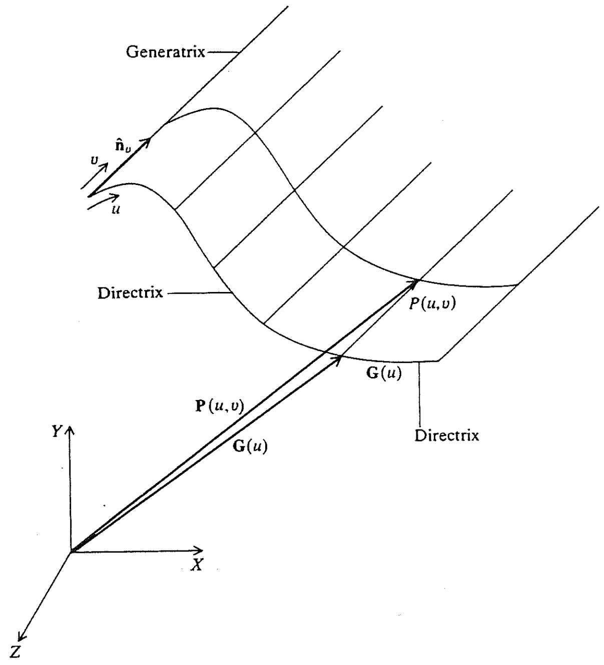


FIGURE 6-29
Parametric representation of a tabulated cylinder.

As seen from Eq. (6.38), 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.

6.6 PARAMETRIC REPRESENTATION OF SYNTHETIC SURFACES

The arguments regarding the needs for synthetic curves discussed in Sec. 5.6 of Chap. 5 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.

6.6.1 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

$$\mathbf{P}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{C}_{ij} u^i v^j, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{6.39}$$

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. (6.39) is

$$\mathbf{P}(u, v) = \mathbf{U}^T[C]\mathbf{V}, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{6.40}$$

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

$$[C] = \begin{bmatrix} \mathbf{C}_{33} & \mathbf{C}_{32} & \mathbf{C}_{31} & \mathbf{C}_{30} \\ \mathbf{C}_{23} & \mathbf{C}_{22} & \mathbf{C}_{21} & \mathbf{C}_{20} \\ \mathbf{C}_{13} & \mathbf{C}_{12} & \mathbf{C}_{11} & \mathbf{C}_{10} \\ \mathbf{C}_{03} & \mathbf{C}_{02} & \mathbf{C}_{01} & \mathbf{C}_{00} \end{bmatrix} \tag{6.41}$$

In order to determine the coefficients $\mathbf{C}_i$, consider the patch shown in Fig. 6-18. Applying the boundary conditions into Eq. (6.40), solving for the coefficients, and rearranging give the following final equation of a bicubic patch:

$$\mathbf{P}(u, v) = \mathbf{U}^T[M_H][B][M_H]^T\mathbf{V}, \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{6.42}$$

where $[M_H]$ is given by Eq. (5.84) and $[B]$, the geometry or boundary condition matrix, is

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{v00} & \mathbf{P}_{v01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{v10} & \mathbf{P}_{v11} \\ \hdashline \mathbf{P}_{u00} & \mathbf{P}_{u01} & \mathbf{P}_{uv00} & \mathbf{P}_{uv01} \\ \mathbf{P}_{u10} & \mathbf{P}_{u11} & \mathbf{P}_{uv10} & \mathbf{P}_{uv11} \end{array}\right] \tag{6.43}$$

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

$$[B] = \left[\begin{array}{c:c} [P] & [P_v] \\ \hdashline [P_u] & [P_{uv}] \end{array}\right] \tag{6.44}$$

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

$$\mathbf{P}_u(u, v) = \mathbf{U}^T[M_H]^u[B][M_H]^T\mathbf{V} \tag{6.45}$$

$$\mathbf{P}_v(u, v) = \mathbf{U}^T[M_H][B][M_H]^{vT}\mathbf{V} \tag{6.46}$$

$$\mathbf{P}_{uv}(u, v) = \mathbf{U}^T[M_H]^u[B][M_H]^{vT}\mathbf{V} \tag{6.47}$$

where $[M_H]^u$ or $[M_H]^v$ is given by Eq. (5.88).

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. (6.42) and (6.45) to (6.47) to see what influences the position and tangent vectors. Equations (6.42) and (6.45) to (6.47) give

$$\mathbf{P}(u, v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)][B]\begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{6.48}$$

$$\mathbf{P}_u(u, v) = [G_1(u) \quad G_2(u) \quad G_3(u) \quad G_4(u)][B]\begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{6.49}$$

$$\mathbf{P}_v(u, v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)][B]\begin{bmatrix} G_1(v) \\ G_2(v) \\ G_3(v) \\ G_4(v) \end{bmatrix} \tag{6.50}$$

$$\mathbf{P}_{uv}(u, v) = [G_1(u) \quad G_2(u) \quad G_3(u) \quad G_4(u)][B]\begin{bmatrix} G_1(v) \\ G_2(v) \\ G_3(v) \\ G_4(v) \end{bmatrix} \tag{6.51}$$

where

$$
\begin{aligned}
F_1(x) &= 2x^3 - 3x^2 + 1 \\
F_2(x) &= -2x^3 + 3x^2 \\
F_3(x) &= x^3 - 2x^2 + x \\
F_4(x) &= x^3 - x^2
\end{aligned}
\tag{6.52}
$$

and

$$
\begin{aligned}
G_1(x) &= 6x^2 - 6x \\
G_2(x) &= -6x^2 + 6x \\
G_3(x) &= 3x^2 - 4x + 1 \\
G_4(x) &= 3x^2 - 2x
\end{aligned}
\tag{6.53}
$$

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

$$
\begin{bmatrix} \mathbf{P}(0, v) \\ \mathbf{P}(1, v) \\ \mathbf{P}_u(0, v) \\ \mathbf{P}_u(1, v) \end{bmatrix} = [B] \begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{6.54}
$$

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

$$
\begin{bmatrix} \mathbf{P}(u, 0) \\ \mathbf{P}(u, 1) \\ \mathbf{P}_v(u, 0) \\ \mathbf{P}_v(u, 1) \end{bmatrix} = [B] \begin{bmatrix} F_1(u) \\ F_2(u) \\ F_3(u) \\ F_4(u) \end{bmatrix} \tag{6.55}
$$

Equation (6.54) 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. 6-16. These conditions to connect patch 1 and patch 2 along the u edges are

$$
\begin{aligned}
[\mathbf{P}(0, v)]_{\text{patch 2}} &= [\mathbf{P}(1, v)]_{\text{patch 1}} && C^0 \text{ continuity} \\
[\mathbf{P}_u(0, v)]_{\text{patch 2}} &= K[\mathbf{P}_u(1, v)]_{\text{patch 1}} && C^1 \text{ continuity}
\end{aligned}
\tag{6.56}
$$

where K is a constant. Equation (6.56) 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. (6.54), Eq. (6.56) can be expressed in terms of the rows of the $[B]$ matrix as shown in Fig. 6-30. 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.

$[B]_{\text{patch 1}}$				$[B]_{\text{patch 2}}$			
				$\mathbf{P}_{10}$	$\mathbf{P}_{11}$	$\mathbf{P}_{v10}$	$\mathbf{P}_{v11}$
$\mathbf{P}_{10}$	$\mathbf{P}_{11}$	$\mathbf{P}_{v10}$	$\mathbf{P}_{v11}$				
				$K\,\mathbf{P}_{u10}$	$K\,\mathbf{P}_{u11}$	$K\,\mathbf{P}_{uv10}$	$K\,\mathbf{P}_{uv11}$
$\mathbf{P}_{u10}$	$\mathbf{P}_{u11}$	$\mathbf{P}_{uv10}$	$\mathbf{P}_{uv11}$				

FIGURE 6-30
Constrained elements of boundary matrix $[B]$ to blend two bicubic patches along a u edge.

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 Fig. 6-31. Thus, the boundary matrix for the F-surface patch becomes

$$
[B] = \left[\begin{array}{cc|cc} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{v00} & \mathbf{P}_{v01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{v10} & \mathbf{P}_{v11} \\ \hline \mathbf{P}_{u00} & \mathbf{P}_{u01} & 0 & 0 \\ \mathbf{P}_{u10} & \mathbf{P}_{u11} & 0 & 0 \end{array}\right] \tag{6.57}
$$

This special 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

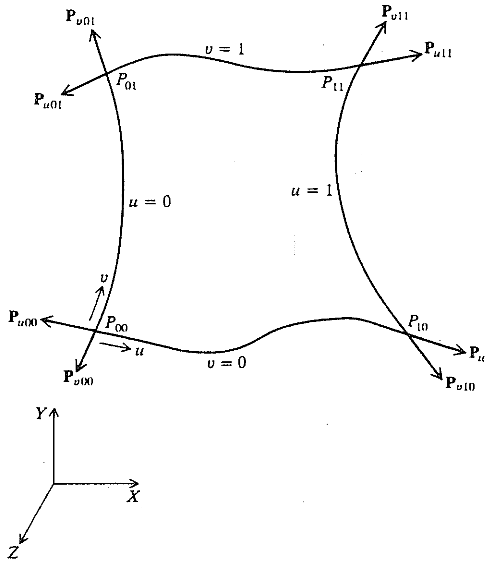


FIGURE 6-31
The F-surface patch.

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 6.4. 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. 6-18. 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. (6.48), 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

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

which is identical to Eq. (5.81) developed in Chap. 5. Also, notice that substituting $v = 1$ in Eq. (6.39) results in Eq. (5.74).

Example 6.5. Sometimes it is useful to reformulate a given surface equation in terms of the bicubic form given by Eq. (6.42). What is the equivalent bicubic patch to a plane given by Eq. (6.28)?

Solution. The general procedure to achieve reformulation of a surface form to a bicubic form is to use Eq. (6.39) 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. (6.39), 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. (6.28). Comparing this equation to Eq. (6.39) gives

$$\mathbf{C}_{00} = \mathbf{P}_0$$
$$\mathbf{C}_{10} = L_u\hat{\mathbf{r}}$$
$$\mathbf{C}_{01} = L_v\hat{\mathbf{s}}$$

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

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_0 & \mathbf{P}_0 + L_v\hat{s} & L_v\hat{\mathbf{s}} & L_v\hat{\mathbf{s}} \\ \mathbf{P}_0 + L_u\hat{\mathbf{r}} & \mathbf{P}_0 + L_u\hat{\mathbf{r}} + L_v\hat{\mathbf{s}} & L_v\hat{\mathbf{s}} & L_v\hat{\mathbf{s}} \\ \hdashline L_u\hat{\mathbf{r}} & L_u\hat{\mathbf{r}} & 0 & 0 \\ L_u\hat{\mathbf{r}} & L_u\hat{\mathbf{r}} & 0 & 0 \end{array}\right]$$

Example 6.6. 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 6-32 shows the above defined bicubic patch. The corner position and tangent vectors can be defined by the following linear functions:

$$\begin{bmatrix} \mathbf{P}_{10} \\ \mathbf{P}_{01} \\ \mathbf{P}_{11} \\ \mathbf{P}_{u10} \\ \mathbf{P}_{u01} \\ \mathbf{P}_{u11} \\ \mathbf{P}_{v10} \\ \mathbf{P}_{v01} \\ \mathbf{P}_{v11} \end{bmatrix} = \begin{bmatrix} 1 & a_1 & b_1 \\ 1 & a_2 & b_2 \\ 1 & a_3 & b_3 \\ 0 & a_4 & b_4 \\ 0 & a_5 & b_5 \\ 0 & a_6 & b_6 \\ 0 & a_7 & b_7 \\ 0 & a_8 & b_8 \\ 0 & a_9 & b_9 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{00} \\ \mathbf{P}_{u00} \\ \mathbf{P}_{v00} \end{bmatrix} \tag{6.58}$$

Substituting the above equation into Eq. (6.43) we get

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{00} + a_2\mathbf{P}_{u00} + b_2\mathbf{P}_{v00} & \mathbf{P}_{v00} & a_8\mathbf{P}_{u00} + b_8\mathbf{P}_{v00} \\ \mathbf{P}_0 + a_1\mathbf{P}_{u00} + b_1\mathbf{P}_{v00} & \mathbf{P}_{00} + a_3\mathbf{P}_{u00} + b_3\mathbf{P}_{v00} & a_7\mathbf{P}_{u00} + b_7\mathbf{P}_{v00} & a_9\mathbf{P}_{v00} + b_9\mathbf{P}_{v00} \\ \hdashline \mathbf{P}_{u00} & a_5\mathbf{P}_{u00} + b_5\mathbf{P}_{v00} & 0 & 0 \\ a_4\mathbf{P}_{u00} + b_4\mathbf{P}_{v00} & a_6\mathbf{P}_{u00} + b_6\mathbf{P}_{v00} & 0 & 0 \end{array}\right] \tag{6.59}$$

If the bicubic patch whose geometry matrix $[B]$ is given by Eq. (6.59) is planar, the following equation can be written for any point on the patch:

$$\mathbf{N}_{00} \cdot [\mathbf{P}(u, v) - \mathbf{P}_{00}] = 0 \tag{6.60}$$

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. 6-32. To prove that Eq. (6.60) is

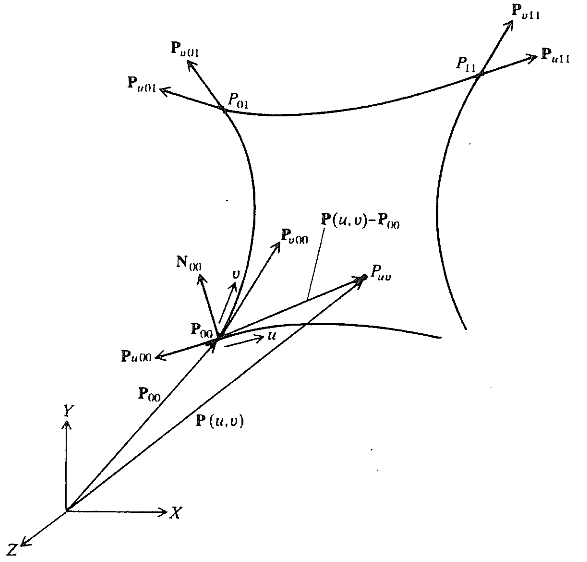


FIGURE 6-32
A bicubic plane patch.

valid for any point on the patch under investi,
(6.48) and reduce the result to obtain

$$\mathbf{P}(u, v) = \mathbf{P}_{00}[F_1(u)F_1(v) + F_2(u)F_1(v) + F_1(u)F$$
$$+ \mathbf{P}_{u00}[a_1 F_2(u)F_1(v) + F_3(u)F_1(v) + a_4$$
$$+ a_3 F_2(u)F_2(v) + a_5 F_3(u)F_2(v)$$
$$+ a_7 F_2(u)F_3(v) + a_8 F_1(u)F_4(v) +$$
$$+ \mathbf{P}_{v00}[b_1 F_2(u)F_1(v) + b_4 F_4(u)F_1(v) + b_2 F$$
$$+ b_5 F_3(u)F_2(v) + b_6 F_4(u)F_2(v) + F$$
$$+ b_8 F_1(u)F_4(v) + b_9 F_2(u)F_4(v)]$$

The coefficient of $\mathbf{P}_{00}$ in the above equation is equal to uı
The coefficients of $\mathbf{P}_{u00}$ and $\mathbf{P}_{v00}$ are functions of u and ι
functions are $f(u, v)$ and $g(u, v)$ respectively, the above equation

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

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

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

Substitute Eqs. (6.61) and (6.62) into the left-hand side of Eq. (6.60).

$$(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot [f(u, v)\mathbf{P}_{u00} + g(u, v)\mathbf{P}_{v00}]$$
$$= f(u, v)(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot \mathbf{P}_{u00} + g(u, v)(\mathbf{P}_{u00} \times \mathbf{P}_{v00}) \cdot \mathbf{P}_{v00} \qquad (6.63)$$

The right-hand side of this equation is equal to zero regardless of the values of u and v. Therefore, Eq. (6.60) 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

$$[S] = [M_H][B][M_H]^T \qquad (6.64)$$

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

$$N_{x00} \sum_{i=1}^{4} \sum_{j=1}^{4} \frac{s_{ijx}}{(5-i)(5-j)} + N_{y00} \sum_{i=1}^{4} \sum_{j=1}^{4} \frac{s_{ijy}}{(5-i)(5-j)}$$
$$+ N_{z00} \sum_{i=1}^{4} \sum_{j=1}^{4} \frac{s_{ijz}}{(5-i)(5-j)} - K = 0 \qquad (6.65)$$

$_{x00}$, N_{y00}, and N_{z00} are the components of the normal vector N_{00} and K l by

$$K = \mathbf{N}_{00} \cdot \mathbf{P}_{00} \qquad (6.66)$$

al implication of this example is the ability to construct planes with daries as opposed to straight boundaries, discussed in Sec. 6.5.1.

Surface

t Bezier surface is an extension of the Bezier curve in two para- u and v. An orderly set of data or control points is used to build ctangular surface as shown in Fig. 6-33. The surface equation

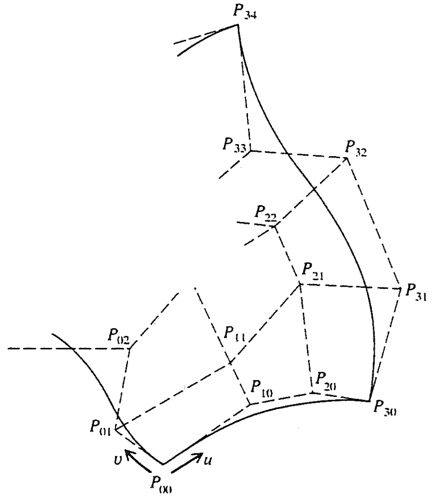


FIGURE 6-33
A 4 × 5 Bezier surface.

can be written by extending Eq. (5.91); that is,

$$\mathbf{P}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{ij} B_{i,n}(u) B_{j,m}(v), \qquad 0 \le u \le 1, 0 \le v \le 1 \tag{6.67}$$

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. 6-33) 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. (6.67) gives

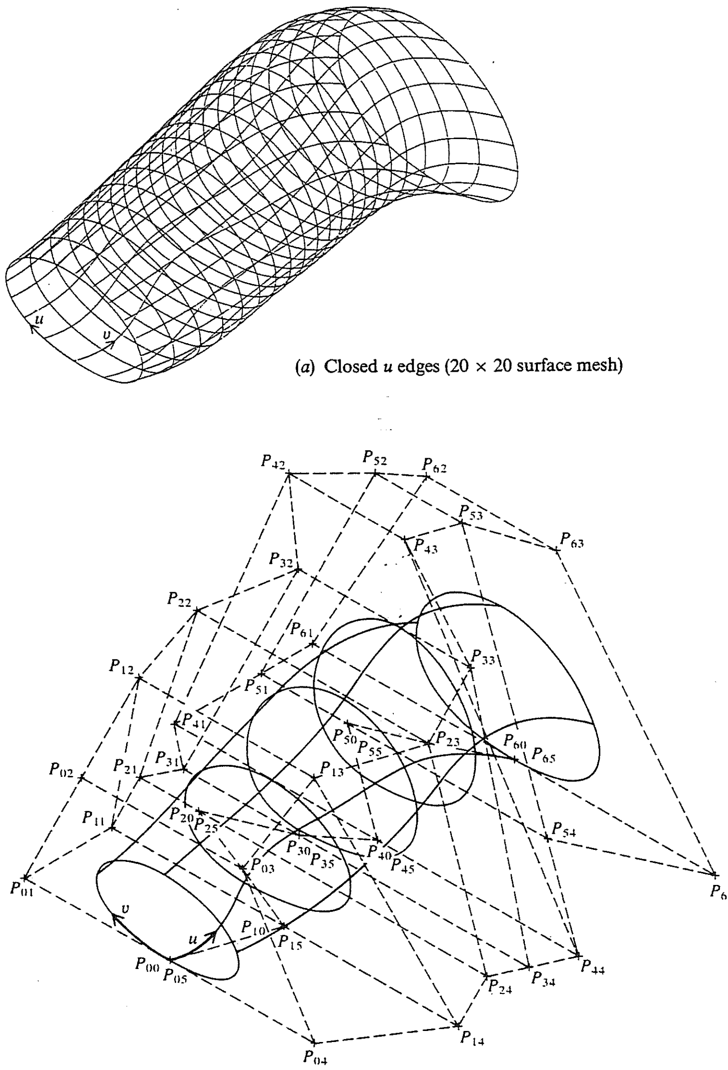
$$\begin{aligned}
\mathbf{P}(u, v) = {} & \left.\begin{aligned} & \mathbf{P}_{00}(1-u)^n(1-v)^m + \mathbf{P}_{11}C(n, 1)C(m, 1)uv(1-u)^{n-1}(1-v)^{m-1} \\
& + \mathbf{P}_{22}\, C(n, 2)C(m, 2)u^2v^2(1-u)^{n-2}(1-v)^{m-2} + \cdots \\
& + \mathbf{P}_{(n-1)(m-1)}C(n, n-1)C(m, m-1)u^{n-1}v^{m-1}(1-u)(1-v) \\
& + \mathbf{P}_{nm}\, u^n v^m \end{aligned}\right] \text{Symmetric terms in } u \text{ and } v \\
\text{Nonsymmetric terms in } u \text{ and } v \; & \left[\begin{aligned} & + \mathbf{P}_{10}\, C(n, 1)u(1-u)^{n-1}(1-v)^m + \mathbf{P}_{01}C(m, 1)v(1-u)^n(1-v)^{m-1} \\
& + \mathbf{P}_{20}\, C(n, 2)u^2(1-u)^{n-2}(1-v)^m + \mathbf{P}_{02}\, C(m, 2)v^2(1-u)^n(1-v)^{m-2} \\
& + \cdots + \mathbf{P}_{n0}\, u^n(1-v)^m + \mathbf{P}_{0m}\, v^m(1-u)^n \\
& + \cdots + \mathbf{P}_{(n-1)(m-2)}\, C(n, n-1)C(m, m-2)u^{n-1}v^{m-2}(1-u)(1-v)^2 \\
& + \mathbf{P}_{(n-2)(m-1)}C(n, n-2)C(m, m-1)u^{n-2}v^{m-1}(1-u)^2(1-v) \\
& + \mathbf{P}_{(n-1)m}\, C(n, n-1)u^{n-1}v^m(1-u) + \mathbf{P}_{n(m-1)}u^n v^{m-1}(1-v) \end{aligned}\right.
\end{aligned} \tag{6.68}$$

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. 6-33) if we substitute the (u, v) values of (0, 0), (1, 0) (0, 1), and (1, 1) into (6.68). The surface is also tangent to the corner segments of the control polyhedron. The tangent vectors at the corners are:

$$\begin{aligned}
\mathbf{P}_{u00} &= n(\mathbf{P}_{10} - \mathbf{P}_{00}) & \mathbf{P}_{un0} &= n(\mathbf{P}_{n0} - \mathbf{P}_{(n-1)0}) & &\text{along } v = 0 \text{ edge} \\
\mathbf{P}_{u0m} &= n(\mathbf{P}_{1m} - \mathbf{P}_{0m}) & \mathbf{P}_{unm} &= n(\mathbf{P}_{nm} - \mathbf{P}_{(n-1)m}) & &\text{along } v = 1 \text{ edge} \\
\mathbf{P}_{v00} &= m(\mathbf{P}_{01} - \mathbf{P}_{00}) & \mathbf{P}_{v0m} &= m(\mathbf{P}_{0m} - \mathbf{P}_{0(m-1)}) & &\text{along } u = 0 \text{ edge} \\
\mathbf{P}_{vn0} &= m(\mathbf{P}_{n1} - \mathbf{P}_{n0}) & \mathbf{P}_{vnm} &= m(\mathbf{P}_{nm} - \mathbf{P}_{n(m-1)}) & &\text{along } u = 1 \text{ edge}
\end{aligned} \tag{6.69}$$

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. 6-34.



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

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

FIGURE 6-34
Closed Bezier surface.

The normal to a Bezier surface at any point can be calculated by substituting Eq. (6.67) into (6.11) to obtain

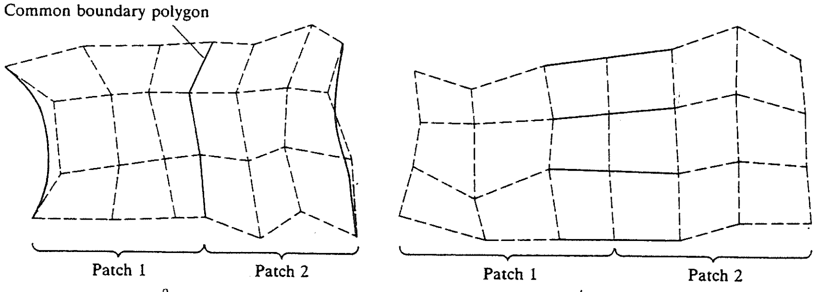
$$\mathbf{N}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{ij} \frac{\partial B_{i,n}(u)}{\partial u} B_{j,m}(v) \times \sum_{k=0}^{n} \sum_{l=0}^{m} \mathbf{P}_{kl} B_{k,n}(u) \frac{\partial B_{l,m}(v)}{\partial v}$$

$$= \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{n} \sum_{l=0}^{m} \frac{\partial B_{i,n}}{\partial u} B_{j,m}(v) B_{k,n}(u) \frac{\partial B_{l,m}(v)}{\partial v} \mathbf{P}_{ij} \times \mathbf{P}_{kl} \qquad (6.70)$$

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. 6-35*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. 6-35*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.



(*a*) C^0 continuity

(*b*) C^1 continuity

FIGURE 6-35
Composite Bezier surface.

Example 6.7. 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 6-36 shows a cubic Bezier patch. Substituting $n = 3$ and $m = 3$ into Eq. (6.67), the patch equation is

$$\mathbf{P}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{P}_{ij} B_{i,3}(u) B_{j,3}(v), \qquad 0 \le u \le 1, 0 \le v \le 1 \qquad (6.71)$$

This equation can be expanded to give

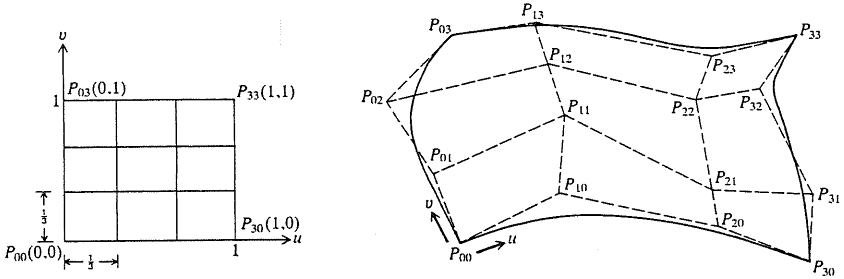
$$\begin{aligned}\mathbf{P}(u, v) &= \sum_{i=0}^{3} B_{i,3}(u)[\mathbf{P}_{i0} B_{0,3}(v) + \mathbf{P}_{i1} B_{1,3}(v) + \mathbf{P}_{i2} B_{2,3}(v) + \mathbf{P}_{i3} B_{3,3}(v)]\\ &= B_{0,3}(u)[\mathbf{P}_{00} B_{0,3}(v) + \mathbf{P}_{01} B_{1,3}(v) + \mathbf{P}_{02} B_{2,3}(v) + \mathbf{P}_{03} B_{3,3}(v)]\\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

This equation can be written in a matrix form as

$$\mathbf{P}(u, v) = [B_{0,3}(u) \quad B_{1,3}(u) \quad B_{2,3}(u) \quad B_{3,3}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix} \begin{bmatrix} B_{0,3}(v) \\ B_{1,3}(v) \\ B_{2,3}(v) \\ B_{3,3}(v) \end{bmatrix} \qquad (6.72)$$

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

$$\text{or} \quad \mathbf{P}(u, v) = \mathbf{U}^T[M_B][P][M_B]^T\mathbf{V} \qquad (6.73)$$



(*a*) Parametric space

(*b*) Cartesian space

FIGURE 6-36
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

$$[M_B] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.74}$$

and the $\mathbf{U}$ and $\mathbf{V}$ vectors are $[u^3 \quad u^2 \quad u \quad 1]^T$ and $[v^3 \quad v^2 \quad v \quad 1]^T$ respectively. Notice that $[M_B]$ given by Eq. (6.74) is the same matrix for the cubic Bezier curve (see Prob. 5.10 in Chap. 5).

Equating Eq. (6.73) with (6.42) 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. (5.86) for $[M_H]^{-1}$, this equation can be reduced to give

$$[B] = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{03} & 3(\mathbf{P}_{01} - \mathbf{P}_{00}) & 3(\mathbf{P}_{03} - \mathbf{P}_{02}) \\ \mathbf{P}_{30} & \mathbf{P}_{33} & 3(\mathbf{P}_{31} - \mathbf{P}_{30}) & 3(\mathbf{P}_{33} - \mathbf{P}_{32}) \\ \hdashline 3(\mathbf{P}_{10} - \mathbf{P}_{00}) & 3(\mathbf{P}_{13} - \mathbf{P}_{03}) & 9(\mathbf{P}_{00} - \mathbf{P}_{10} - \mathbf{P}_{01} + \mathbf{P}_{11}) & 9(\mathbf{P}_{02} - \mathbf{P}_{12} - \mathbf{P}_{03} + \mathbf{P}_{13}) \\ 3(\mathbf{P}_{30} - \mathbf{P}_{20}) & 3(\mathbf{P}_{33} - \mathbf{P}_{23}) & 9(\mathbf{P}_{20} - \mathbf{P}_{21} - \mathbf{P}_{30} + \mathbf{P}_{31}) & 9(\mathbf{P}_{22} - \mathbf{P}_{23} - \mathbf{P}_{32} + \mathbf{P}_{33}) \end{array}\right] \tag{6.75}$$

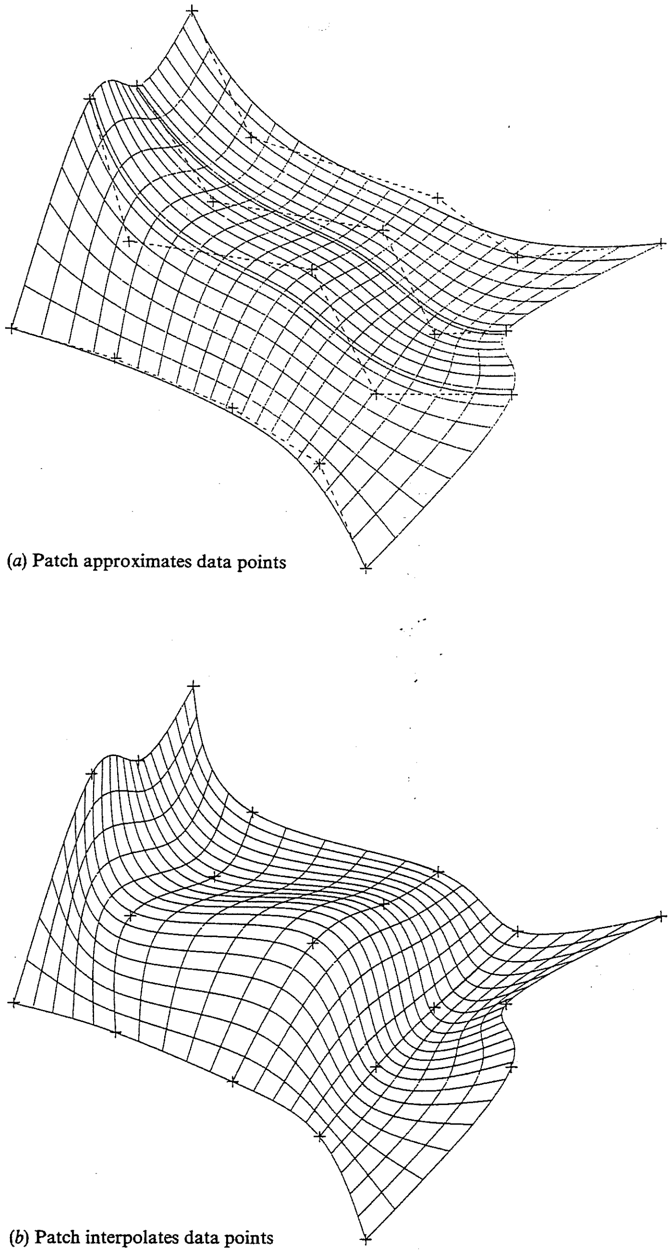
Comparing this equation with Eq. (6.43) for the bicubic patch reveals that the tangent and twist vectors of the Bezier surface are expressed in terms of the vertices of its characteristic polyhedron.

Note: equation (6.72) offers a concise matrix form of Eq. (6.67) for a cubic Bezier surface. This form can be extended to an $(n + 1) \times (m + 1)$ surface as follows:

$$\mathbf{P}(u, v) = [B_{0,n}(u) \quad B_{1,n}(u) \quad \cdots \quad B_{n,n}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \cdots & \mathbf{P}_{0m} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1m} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{n0} & \mathbf{P}_{n1} & \cdots & \mathbf{P}_{nm} \end{bmatrix} \begin{bmatrix} B_{0,m}(v) \\ B_{1,m}(v) \\ \vdots \\ B_{n,m}(v) \end{bmatrix} \tag{6.76}$$

6.6.3 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. 6-37.



(*a*) Patch approximates data points

(*b*) Patch interpolates data points

FIGURE 6-37
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. (5.103) into two dimensions:

$$\mathbf{P}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{ij} N_{i,k}(u) N_{j,l}(v), \qquad 0 \le u \le u_{\max},\ 0 \le v \le v_{\max} \tag{6.77}$$

All the related discussions to Eqs. (5.104) and (5.105) apply to the above equation. Equation (6.77) 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 6.8. Find the equivalent bicubic formulation of an open and closed cubic B-spline surface.

Solution. Most of the results obtained in Examples 5.21 and 5.22 can be extended to a cubic B-spline surface. First, let us find the matrix form of Eq. (6.77). This equation is identical in form to the Bezier surface equation (6.67). Thus, by replacing the Bernstein polynomials in Eq. (6.76) by the B-spline functions yields the matrix form of Eq. (6.77) as

$$\mathbf{P}(u, v) = [N_{0,k}(u) \quad N_{1,k}(u) \quad \cdots \quad N_{n,k}(u)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \cdots & \mathbf{P}_{0m} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1m} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{n0} & \mathbf{P}_{n1} & \cdots & \mathbf{P}_{nm} \end{bmatrix} \begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix} \tag{6.78}$$

or

$$\mathbf{P}(u, v) = [N_{0,k}(u) \quad N_{1,k}(u) \quad \cdots \quad N_{n,k}(u)][P] \begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix} \tag{6.79}$$

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. (6.78) becomes

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

For the open patch, the B-spline functions are the same as the Bernstein polynomials of the previous example (refer to Example 5.21) and the equivalent bicubic formulation results in Eq. (6.75). For, say, a 5 × 6 open cubic B-spline patch, we get

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

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 5.21. After this is done, the above equation can be reduced to

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

$$= \left[\begin{array}{c:c:c} \mathbf{U}^T[M_S][P_{11}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{12}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{13}][M_S]^T\mathbf{V} \\ \hdashline \mathbf{U}^T[M_S][P_{21}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{22}][M_S]^T\mathbf{V} & \mathbf{U}^T[M_S][P_{23}][M_S]^T\mathbf{V} \end{array}\right] \tag{6.80}$$

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 (see Prob. 5.15 in Chap. 5). It is given by

$$[M_S] = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{6.81}$$

Equation (6.80) can be written in a more concise form as

$$\mathbf{P}_{ij}(u, v) = \mathbf{U}^T[M_S][\mathbf{P}_{ij}][M_S]^T\mathbf{V}$$

$$1 \le i \le 2,\ 1 \le j \le 3;\ (i - 1) \le u \le i,\ (j - 1) \le v \le j \tag{6.82}$$

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

$$[B_{ij}] = [M_H]^{-1}[M_S][P_{ij}][M_S]^T[M_H]^{T-1} \tag{6.83}$$

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:

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

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

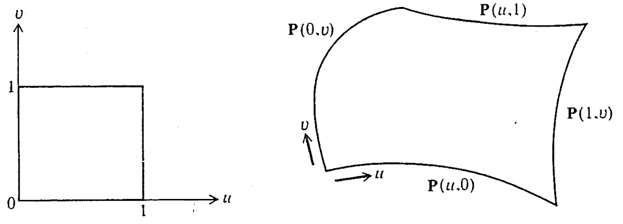
and $\mathbf{P}(u, v) = [N_{0,4}((u + 4) \bmod 4),\ N_{0,4}((u + 3) \bmod 4),\ N_{0,4}((u + 2) \bmod 4),\ N_{0,4}((u + 1) \bmod 4)]$

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

The closed B-spline functions have been evaluated in Example 5.22. Investigating Eqs. (5.118) reveals that there are four different expressions for the matrix $[B]$ [see Eq. (6.83) 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.

6.6.4 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. 6-38. The figure shows the given four boundary curves as $\mathbf{P}(u, 0)$, $\mathbf{P}(1, v)$, $\mathbf{P}(u, 1)$, and $\mathbf{P}(0, v)$. 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

FIGURE 6-38
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. 6-38. 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. (6.36). 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. (6.36) in the v and u directions gives respectively

$$\mathbf{P}_1(u, v) = (1 - u)\mathbf{P}(0, v) + u\mathbf{P}(1, v) \tag{6.84}$$

and

$$\mathbf{P}_2(u, v) = (1 - v)\mathbf{P}(u, 0) + v\mathbf{P}(u, 1) \tag{6.85}$$

Adding these two equations gives the surface

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) + \mathbf{P}_2(u, v) \tag{6.86}$$

The resulting surface patch described by Eq. (6.86) does not satisfy the boundary conditions. For example, substituting $v = 0$ and 1 into this equation gives respectively

$$\mathbf{P}(u, 0) = \mathbf{P}(u, 0) + [(1 - u)\mathbf{P}(0, 0) + u\mathbf{P}(1, 0)] \tag{6.87}$$

$$\mathbf{P}(u, 1) = \mathbf{P}(u, 1) + [(1 - u)\mathbf{P}(0, 1) + u\mathbf{P}(1, 1)] \tag{6.88}$$

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. (6.86). This surface can be defined by linear interpolation in the v direction, that is,

$$\mathbf{P}_3(u, v) = (1 - v)[(1 - u)\mathbf{P}(0, 0) + u\mathbf{P}(1, 0)] + v[(1 - u)\mathbf{P}(0, 1) + u\mathbf{P}(1, 1)] \tag{6.89}$$

Subtracting $\mathbf{P}_3(u, v)$ (called the "correction surface") from Eq. (6.86) gives

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) + \mathbf{P}_2(u, v) - \mathbf{P}_3(u, v) \qquad (6.90)$$

or

$$\mathbf{P}(u, v) = \mathbf{P}_1(u, v) \oplus \mathbf{P}_2(u, v) \qquad (6.91)$$

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. 6-38. Figure 6-39 shows the graphical representation of Eq. (6.91) and its matrix form is

$$\mathbf{P}(u, v) = -[-1 \quad (1-u) \quad u]\left[\begin{array}{c|cc} 0 & \mathbf{P}(u, 0) & \mathbf{P}(u, 1) \\ \hline \mathbf{P}(0, v) & \mathbf{P}(0, 0) & \mathbf{P}(0, 1) \\ \mathbf{P}(1, v) & \mathbf{P}(1, 0) & \mathbf{P}(1, 1) \end{array}\right]\left[\begin{array}{c} -1 \\ 1-v \\ v \end{array}\right] \qquad (6.92)$$

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. 6-40) $\mathbf{P}_u(1, v)$ and $\mathbf{P}_u(0, v)$ of the two patches cannot be made equal (see Prob. 6.10 at the end of the chapter).

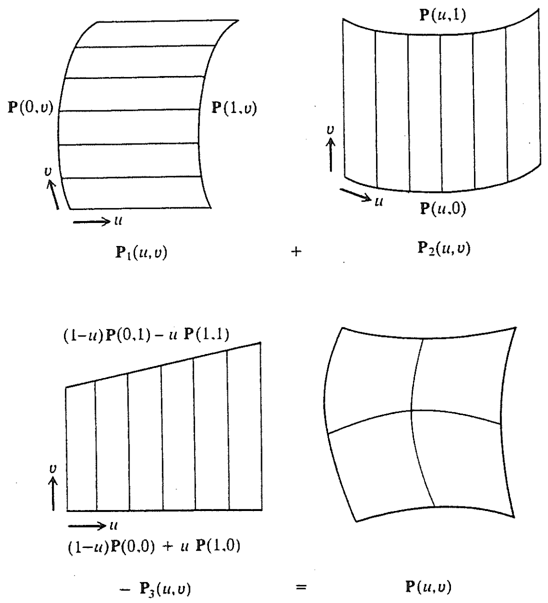


FIGURE 6-39
Bilinearly blended Coons patch (boolean sum).

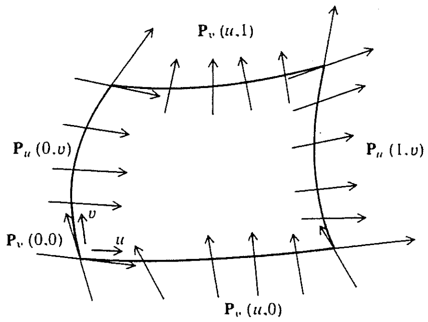


FIGURE 6-40
Cross-boundary derivatives.

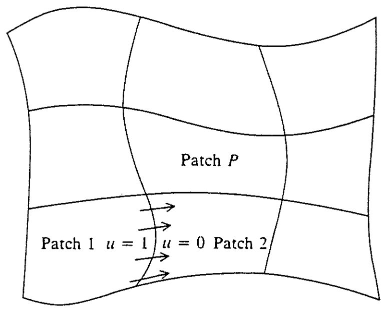


FIGURE 6-41
A composed Coons surface formed by a network of C^1 boundary curves.

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. 6-41, 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. (6.92) 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. (6.52) 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. (6.84) and (6.85) gives

$$\mathbf{P}_1(u, v) = (2u^3 - 3u^2 + 1)\mathbf{P}(0, v) + (-2u^3 + 3u^2)\mathbf{P}(1, v) \qquad (6.93)$$

$$\mathbf{P}_2(u, v) = (2v^3 - 3v^2 + 1)\mathbf{P}(u, 0) + (-2v^3 + 3v^2)\mathbf{P}(u, 1) \qquad (6.94)$$

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

$$\mathbf{P}(u, v) = -[-1 \quad F_1(u) \quad F_2(u)]\left[\begin{array}{c|cc} 0 & \mathbf{P}(u, 0) & \mathbf{P}(u, 1) \\ \hline \mathbf{P}(0, v) & \mathbf{P}(0, 0) & \mathbf{P}(0, 1) \\ \mathbf{P}(1, v) & \mathbf{P}(1, 0) & \mathbf{P}(1, 1) \end{array}\right]\left[\begin{array}{c} -1 \\ F_1(v) \\ F_2(v) \end{array}\right] \qquad (6.95)$$

As can easily be seen from Eqs. (6.92) and (6.95), 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. 6-41. For C^0 and C^1 continuity we should have

$$[\mathbf{P}(0, v)]_{\text{patch 2}} = [\mathbf{P}(1, v)]_{\text{patch 1}} \qquad (6.96)$$

$$[\mathbf{P}_u(0, v)]_{\text{patch 2}} = [\mathbf{P}_u(1, v)]_{\text{patch 1}} \qquad (6.97)$$

C^0 continuity is automatically satisfied between the two patches because they share the same boundary curve. For C^1 continuity, differentiating Eq. (6.95) with respect to u, using $G_1(x)$ and $G_2(x)$ in Eq. (6.53) 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

$$\mathbf{P}_u(u, v) = F_1(v)\mathbf{P}_u(u, 0) + F_2(v)\mathbf{P}_u(u, 1) \tag{6.98}$$

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

$$[\mathbf{P}_u(0, v)]_{\text{patch 2}} = F_1(v)\mathbf{P}_u(0, 0) + F_2(v)\mathbf{P}_u(0, 1) \tag{6.99}$$

and
$$[\mathbf{P}_u(1, v)]_{\text{patch 1}} = F_1(v)\mathbf{P}_u(1, 0) + F_2(v)\mathbf{P}_u(1, 1) \tag{6.100}$$

Based on Eqs. (6.99) and (6.100), Eq. (6.97) 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. (6.95) 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. 6-40). Note that at the corners these derivatives must be compatible with the curve information. Similar to Eq. (6.93) and (6.94), we can define the following cubic Hermite interpolants:

$$\mathbf{P}_1(u, v) = F_1(u)\mathbf{P}(0, v) + F_2(u)\mathbf{P}(1, v) + F_3(u)\mathbf{P}_u(0, v) + F_4(u)\mathbf{P}_u(1, v) \tag{6.101}$$

and
$$\mathbf{P}_2(u, v) = F_1(v)\mathbf{P}(u, 0) + F_2(v)\mathbf{P}(u, 1) + F_3(v)\mathbf{P}_v(u, 0) + F_4(v)\mathbf{P}_v(u, 1) \tag{6.102}$$

where the functions F_1 to F_4 are given by Eq. (6.52). 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 (see Prob. 6.11 at the end of the chapter). The matrix equation of this Coons patch can be written as

$$\mathbf{P}(u, v) = -[-1 \quad F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)]$$
$$\times \left[\begin{array}{c:cc:cc} 0 & \mathbf{P}(u, 0) & \mathbf{P}(u, 1) & \mathbf{P}_v(u, 0) & \mathbf{P}_v(u, 1) \\ \hdashline \mathbf{P}(0, v) & \mathbf{P}(0, 0) & \mathbf{P}(0, 1) & \mathbf{P}_v(0, 0) & \mathbf{P}_v(0, 1) \\ \mathbf{P}(1, v) & \mathbf{P}(1, 0) & \mathbf{P}(1, 1) & \mathbf{P}_v(1, 0) & \mathbf{P}_v(1, 1) \\ \hdashline \mathbf{P}_u(0, v) & \mathbf{P}_u(0, 0) & \mathbf{P}_u(0, 1) & \mathbf{P}_{uv}(0, 0) & \mathbf{P}_{uv}(0, 1) \\ \mathbf{P}_u(1, v) & \mathbf{P}_u(1, 0) & \mathbf{P}_u(1, 1) & \mathbf{P}_{uv}(1, 0) & \mathbf{P}_{uv}(1, 1) \end{array}\right] \begin{bmatrix} -1 \\ F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \tag{6.103}$$

The upper 3 × 3 matrix determines the patch defined previously by Eq. (6.95). The left column and the upper row represent $\mathbf{P}_1(u, v)$ and $\mathbf{P}_2(u, v)$ respectively, while the lower right 4 × 4 matrix represents the bicubic tensor product surface discussed in Sec. 6.6.1 [see Eq. (6.43)]. Equation (6.103) 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 6.9. Show that if the boundary curves of a bilinear Coons patch are coplanar, 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. (6.90), 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 $\mathbf{A}$ and $\mathbf{B}$ 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 6.6. 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.

6.6.5 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. 6-11 is considered a special case of a blending surface. Figure 6-42 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.

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

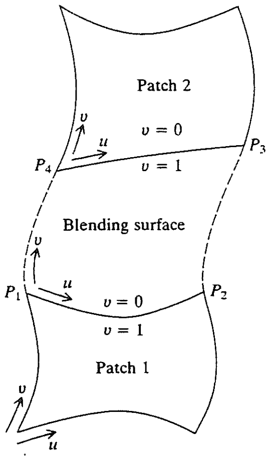


FIGURE 6-42
A blending surface.

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. 6-11, 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.

6.6.6 Offset Surface

If an original patch and an offset direction are given as shown in Fig. 6-12, the equation of the resulting offset patch can be written as:

$$\mathbf{P}(u, v)_{\text{offset}} = \mathbf{P}(u, v) + \hat{\mathbf{n}}(u, v)\, d(u, v) \qquad (6.104)$$

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. 6-12. 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.

6.6.7 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

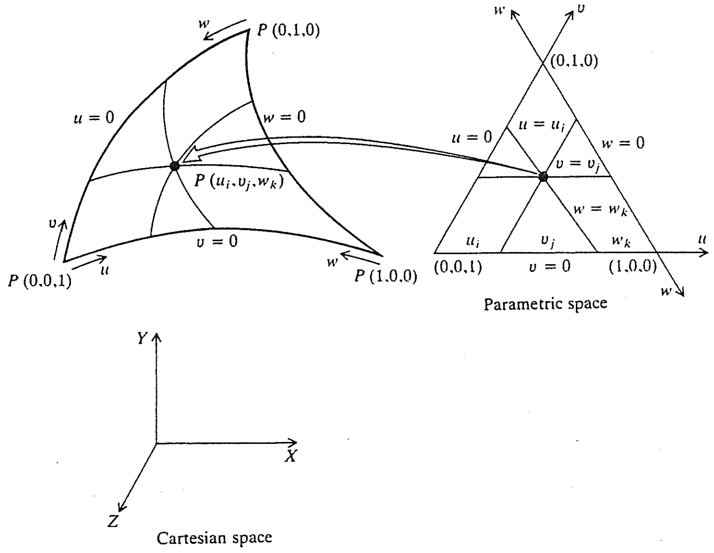


FIGURE 6-43
Representation of a triangular patch.

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. 6-43. 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.

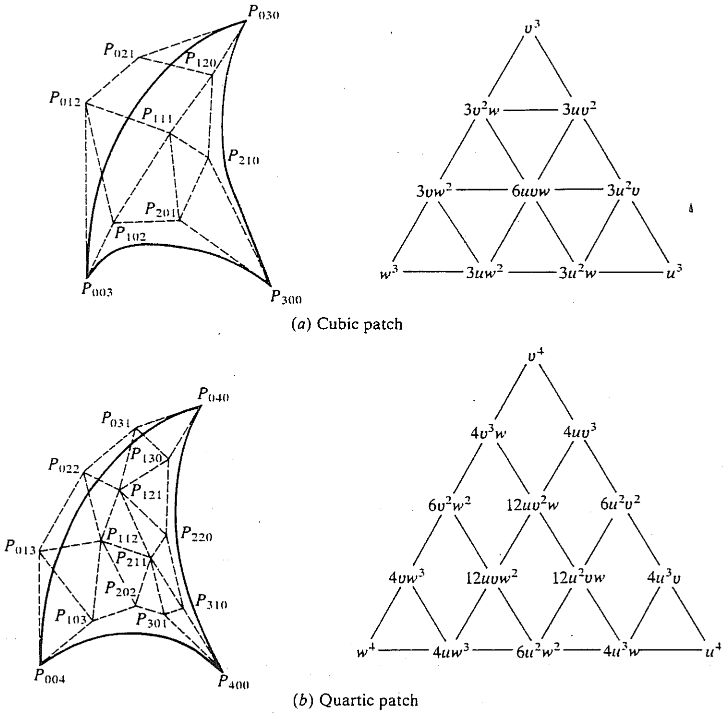
The formulation of triangular polynomial patches follows a somewhat similar pattern to that of tensor product patches. For example, a triangular Bezier patch is defined by

$$\mathbf{P}(u, v, w) = \sum_{i, j, k} \mathbf{P}_{ijk} B_{i, j, k, n}(u, v, w), \qquad 0 \le u \le 1,\ 0 \le v \le 1,\ 0 \le w \le 1 \quad (6.105)$$

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:

$$B_{i, j, k, n} = \frac{n!}{i!\,j!\,k!} u^i v^j w^k \qquad (6.106)$$

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 6-44 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



(a) Cubic patch

(b) Quartic patch

FIGURE 6-44
Triangular Bezier patches.

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.

6.6.8 Sculptured Surface

Any of the surface patches introduced thus far is seldom enough by itself to model complex surfaces typically encountered in design and manufacturing applications. These complex surfaces are known as sculptured or free-form surfaces.

They arise extensively in automotive die and mold-making, aerospace, glass, cameras, shoes, and appliance industries—to name a few.

A sculptured surface is defined as a collection or sum of interconnected and bounded parametric patches together with blending and interpolation formulas. The surface must be susceptible to APT, or other machining languages, processing for NC machine tools. The analytic and synthetic patches described thus far can be used to create a sculptured surface. From a modeling viewpoint, a sculptured surface can be divided into the proper patches which can be created to produce a C^0 or C^1 continuous surface. The two-patch surfaces (e.g., Fig. 6-16) discussed throughout this chapter thus far are considered sculptured surfaces.

6.6.9 Rational Parametric Surface

The rational parametric surface is considered a very general surface. The shape of the surface is controlled by weights (scalar factors) assigned to each control point. A rational surface is defined by the algebraic ratio of two polynomials. A rational tensor product surface can be represented as

$$\mathbf{P}(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{ij} h_{ij} F_i(u) F_j(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} h_{ij} F_i(u) F_j(v)} \tag{6.107}$$

where h_{ij} are the weights assigned to the control points. Rational bicubic, Bezier, and B-spline surfaces are available. The discussion regarding rational curves covered in Sec. 5.6.4 can be extended to rational surfaces.

6.7 SURFACE MANIPULATIONS

As with curves, surface manipulation provides the designer with the capabilities to use surfaces effectively in design applications. For example, if two surfaces are to be welded by a robot, their intersection curve provides the path the robot should follow. Therefore, the robot trajectory planning becomes very accurate. This section covers some useful features of surface manipulations.

6.7.1 Displaying

The simplest method to display surfaces is to generate a mesh of curves on the surface by holding one parameter constant at a time. Most surface commands on CAD/CAM systems have a mesh size modifier. Using this size and the surface equation, the curves of the mesh are evaluated. The display of these curves follows the same techniques discussed in Sec. 5.7.1. For example, to display a Bezier surface of mesh size 3 x 4, three curves that have constant v values ($v = 0, \frac{1}{2}, 1$) and four curves that have constant u values ($u = 0, \frac{1}{3}, \frac{2}{3}, 1$) are evaluated using Eq. (6.67) and displayed. This method is not very efficient as fine details of the surface may be lost unless the mesh is very dense, in which case it becomes slow and expensive to generate, display, and/or update the surface. This method of displaying a surface by a mesh of curves is sometimes referred to as a wireframe display of a surface.

To improve the visualization of a surface, surface normals can be displayed as straight lines in addition to the wireframe display. If the lengths of these straight lines are made long enough, any small variation in surface shape becomes evident. Another good method to use to display surfaces is shading. Shading techniques are discussed in more detail in Chap. 10. Various surfaces can be shaded with various colors. Surface curvatures and other related properties can also be displayed via shading. If surfaces are nested, inner shaded surfaces can be viewed by using X-ray and transparency techniques.

6.7.2 Evaluating Points and Curves on Surfaces

Generating points on surfaces is useful for applications such as finite element modeling and analysis. The "generate point on" command or its equivalent, provided by CAD/CAM systems and mentioned in Sec. 5.7.2, is also available for surfaces. The obvious method of calculating points on a surface is by substituting their respective u and v values into the surface equation. The forward difference technique for evaluating points on curves described by Eq. (5.121) can be extended to surfaces by using the equation for constant values of v. If the specific v value is substituted into the surface equation, Eq. (5.121) results. Therefore, Eqs. (5.122) and (5.123) can be applied to generate the points for the given v value. This procedure can be repeated for all v values to generate the desired points on the surface.

Similarly to curves, evaluating a point on a surface for given u and v values is the direct point solution, which consists of evaluating three polynomials in u and v, one for each coordinate of the point. The inverse problem, which requires finding the corresponding u and v values if the x, y, and z coordinates of the point are given, arises if tangent vectors at the point are to be evaluated or if two coordinates of the point are given and we must find the third coordinate. The solution of this problem (the inverse point solution) requires solving two nonlinear polynomials in u and v simultaneously via numeric methods (see Prob. 6.12 at the end of the chapter).

6.7.3 Segmentation

The segmentation problem of a given surface is identical to that of a curve, which is discussed in Sec. 5.7.4. Surface segmentation is a reparametrization or parameter transformation of the surface while keeping the degree of its polynomial in u and v unchanged.

Let us assume a surface patch is defined over the range $u = u_0$, u_m and $v = v_0$, v_m and it is desired to split it at a point P_1 defined by (u_1, v_1) as shown in Fig. 6-45. If the point is defined by its cartesian coordinates, the inverse point solution must be found first to obtain u_1 and v_1. Let us introduce the new parameters u^1 and v^1 such that their ranges are (0, 1) for each resulting subpatch (see Fig. 6-45).

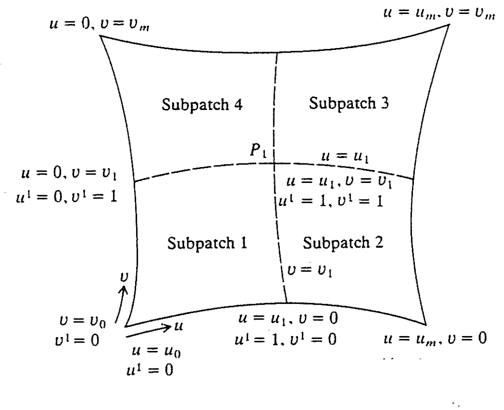


FIGURE 6-45
Reparametrization of a segmented surface patch.

The parameter transformation takes the form:

$$\left.\begin{aligned} u^1 &= u_0 + (u_1 - u_0)u \\ v^1 &= v_0 + (v_1 - v_0)v \end{aligned}\right\} \quad \text{for subpatch 1} \qquad (6.108)$$

Similar equations can be written for the other subpatches. Notice that $u^1 = 0, 1$ and $v^1 = 0, 1$ correspond to the proper values of u and v for the subpatch. If the surface is described by a polynomial, Eq. (5.133) and a similar one for v can be used to facilitate substitutions in the surface equation.

If the surface is to be divided into two segments or subpatches along the $u = u_1$ or $v = v_1$ curve instead of four segments, Eq. (6.108) becomes

$$\left.\begin{aligned} u^1 &= u_0 + (u_1 - u_0)u \\ v^1 &= v_0 + (v_m - v_0)v \end{aligned}\right\} \quad \text{for the first segment} \qquad (6.109)$$

if segmentation is done along the $u = u_1$ curve, and

$$\left.\begin{aligned} u^1 &= u_0 + (u_m - u_0)u \\ v^1 &= v_0 + (v_1 - v_0)v \end{aligned}\right\} \quad \text{for the first segment} \qquad (6.110)$$

if segmentation is done along the $v = v_1$ curve.

Example 6.10. A bicubic surface patch is to be divided by a designer into four subpatches. Find the boundary conditions of each of the resulting subpatches.

Solution. Using Eq. (5.133) in its matrix form found in Example 5.23, Eq. (6.42) can be written as

$$\mathbf{P}(u, v) = \mathbf{U}^{1T}[T]^u[M_H][B][M_H]^T[T]^{vT}\mathbf{V}^1 = \mathbf{P}^*(u^1, v^1)$$

This equation can be rewritten in the form of Eq. (6.42) as

$$\begin{aligned} \mathbf{P}^*(u^1, v^1) &= \mathbf{U}^{1T}[M_H][M_H]^{-1}[T]^u[M_H][B][M_H]^T[T]^{vT}[M_H]^{T^{-1}}[M_H]^T\mathbf{V}^1 \\ &= \mathbf{U}^{1T}[M_H][B]^*[M_H]^T\mathbf{V}^1 \end{aligned}$$

where

$$[B]^* = [M_H]^{-1}[T]^u[M_H][B][M_H]^T[T]^{vT}[M_H]^{-1T}$$

$$= [M_H]^{-1}[T]^u[M_H][B][[M_H]^{-1}[T]^v[M_H]]^T$$

or
$$[B]^* = [A]^u[B][A]^{vT} \tag{6.111}$$

where $[A] = [M_H]^{-1}[T][M_H]$. Using Eqs. (5.84) and (5.86) and the $[T]$ matrix from Example 5.23, $[A]^u$ is given by

$$[A]^u = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta u_0^3 & 0 & 0 & 0 \\ 0 & \Delta u_0^2 & 0 & 0 \\ 0 & 0 & \Delta u_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The above diagonal $[T]$ matrix resulted from assuming $u_0 = 0$. Similarly, we assume $v_0 = 0$. These assumptions do not restrict the solution method but they rather simplify the matrix manipulations (the reader can re-solve Example 5.23 with $u_0 = 0$ for clarification of this point if necessary). Performing the above matrix multiplications and utilizing Eqs. (6.52) and (6.53) in the result, we obtain

$$[A]^u = \begin{bmatrix} 1 & 0 & 0 & 0 \\ F_1(\Delta u_0) & F_2(\Delta u_0) & F_3(\Delta u_0) & F_4(\Delta u_0) \\ 0 & 0 & \Delta u_0 & 0 \\ \Delta u_0\, G_1(\Delta u_0) & \Delta u_0\, G_2(\Delta u_0) & \Delta u_0\, G_3(\Delta u_0) & \Delta u_0\, G_4(\Delta u_0) \end{bmatrix}$$

The matrix $[A]^{vT}$ is obtained from the above matrix by replacing Δu_0 by Δv_0 and transposing the result to get

$$[A]^{vT} = \begin{bmatrix} 1 & F_1(\Delta v_0) & 0 & \Delta v_0\, G_1(\Delta v_0) \\ 0 & F_2(\Delta v_0) & 0 & \Delta v_0\, G_2(\Delta v_0) \\ 0 & F_3(\Delta v_0) & \Delta v_0 & \Delta v_0\, G_3(\Delta v_0) \\ 0 & F_4(\Delta v_0) & 0 & \Delta v_0\, G_4(\Delta v_0) \end{bmatrix}$$

Substituting $[A]^u$ and $[A]^{vT}$ into Eq. (6.111) gives the geometry or boundary condition matrix for subpatch 1 shown in Fig. 6-45. To reduce the results into a useful form, the cubic polynomial interpolation given by Eqs. (5.81) and (5.82) must be recognized during matrix multiplications. In Eq. (6.111), assume $[A]^u[B] = [D]$. Thus, the elements of $[D]$ are

$$\mathbf{d}_{11} = \mathbf{P}_{00}$$

$$\mathbf{d}_{12} = \mathbf{P}_{01}$$

$$\mathbf{d}_{13} = \mathbf{P}_{v00}$$

$$\mathbf{d}_{14} = \mathbf{P}_{v01}$$

$$\mathbf{d}_{21} = \mathbf{P}_{00}F_1(u_1) + \mathbf{P}_{10}F_2(u_1) + \mathbf{P}_{u00}F_3(u_1) + \mathbf{P}_{u10}F_4(u_1)$$

We have substituted u_1 for Δu_0 based on the assumption that $u_0 = 0$. The right-hand side of the above equation is a cubic spline of the form given by Eq. (5.81) evaluated at u_1. It interpolates the boundary conditions at the corners P_{00} and P_{10} of the original patch. Therefore:

$$\mathbf{d}_{21} = \mathbf{P}(u_{1,\,o})$$

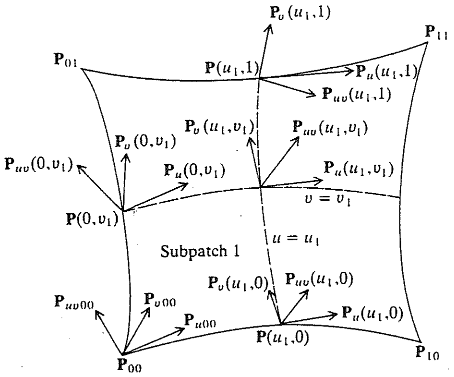


FIGURE 6-46
Related vectors to subpatch 1.

This point is shown in Fig. 6-46. Repeating this interpretation process and referring to Fig. 6-46, we obtain

$$\mathbf{d}_{22} = \mathbf{P}_{01}F_1(u_1) + \mathbf{P}_{11}F_2(u_1) + \mathbf{P}_{u01}F_3(u_1) + \mathbf{P}_{u11}F_4(u_1) = \mathbf{P}(u_1, 1)$$

$$\mathbf{d}_{23} = \mathbf{P}_{v00}F_1(u_1) + \mathbf{P}_{v10}F_2(u_1) + \mathbf{P}_{uv00}F_3(u_1) + \mathbf{P}_{uv10}F_4(u_1) = \mathbf{P}_v(u_1, 0)$$

In the above equation, it is obvious that the cubic polynomial interpolates the v-tangent vector between the corners P_{00} and P_{10}.

$$\mathbf{d}_{24} = \mathbf{P}_{v01}F_1(u_1) + \mathbf{P}_{v11}F_2(u_1) + \mathbf{P}_{uv01}F_3(u_1) + \mathbf{P}_{uv11}F_4(u_1) = \mathbf{P}_v(u_1, 1)$$

$$\mathbf{d}_{31} = \Delta u_0\,\mathbf{P}_{u00}$$

$$\mathbf{d}_{32} = \Delta u_0\,\mathbf{P}_{u01}$$

$$\mathbf{d}_{33} = \Delta u_0\,\mathbf{P}_{uv00}$$

$$\mathbf{d}_{34} = \Delta u_0\,\mathbf{P}_{uv01}$$

$$\mathbf{d}_{41} = \Delta u_0[\mathbf{P}_{00}G_1(u_1) + \mathbf{P}_{10}G_2(u_1) + \mathbf{P}_{u00}G_3(u_1) + \mathbf{P}_{u10}G_4(u_1)]$$

Using Eq. (5.82), we can write

$$\mathbf{d}_{41} = \Delta u_0\,\mathbf{P}_u(u_1, 0)$$

$$\mathbf{d}_{42} = \Delta u_0[\mathbf{P}_{01}G_1(u_1) + \mathbf{P}_{11}G_2(u_1) + \mathbf{P}_{u01}G_3(u_1) + \mathbf{P}_{u11}G_4(u_1)] = \Delta u_0\,\mathbf{P}_u(u_1, 1)$$

$$\mathbf{d}_{43} = \Delta u_0[\mathbf{P}_{v00}G_1(u_1) + \mathbf{P}_{v10}G_2(u_1) + \mathbf{P}_{uv00}G_3(u_1) + \mathbf{P}_{uv10}G_4(u_1)] = \Delta u_0\,\mathbf{P}_{uv}(u_1, 0)$$

$$\mathbf{d}_{44} = \Delta u_0[\mathbf{P}_{v01}G_1(u_1) + \mathbf{P}_{v11}G_2(u_1) + \mathbf{P}_{uv01}G_3(u_1) + \mathbf{P}_{uv11}G_4(u_1)] = \Delta u_0\,\mathbf{P}_{uv}(u_1, 1)$$

Returning to Eq. (6.111), the elements of $[B]^*$ result from multiplying $[D]$ and $[A]^{vT}$:

$$\mathbf{b}^*_{11} = \mathbf{P}_{00}$$

$$\mathbf{b}^*_{12} = \mathbf{P}_{00}F_1(v_1) + \mathbf{P}_{01}F_2(v_1) + \mathbf{P}_{v00}F_3(v_1) + \mathbf{P}_{v01}F_4(v_1) = \mathbf{P}(0, v_1)$$

$$\mathbf{b}^*_{13} = \Delta v_0\,\mathbf{P}_{v00}$$

$$\mathbf{b}^*_{14} = \Delta v_0[\mathbf{P}_{00}G_1(v_1) + \mathbf{P}_{01}G_2(v_1) + \mathbf{P}_{v00}G_3(v_1) + \mathbf{P}_{v01}G_4(v_1)] = \Delta v_0\,\mathbf{P}_v(0, v_1)$$

$$\mathbf{b}^*_{21} = \mathbf{P}(u_1, 0)$$

$$\mathbf{b}_{22}^* = \mathbf{P}(u_1, 0)F_1(v_1) + \mathbf{P}(u_1, 1)F_2(v_1) + \mathbf{P}_v(u_1, 0)F_3(v_1) + \mathbf{P}_v(u_1, 1)F_4(v_1)$$

$$= \mathbf{P}(u_1, v_1) = \mathbf{P}_1$$

$$\mathbf{b}_{23}^* = \Delta v_0 \mathbf{P}_v(u_1, 0)$$

$$\mathbf{b}_{24}^* = \Delta v_0[\mathbf{P}(u_1, 0)G_1(v_1) + \mathbf{P}(u_1, 1)G_2(v_1) + \mathbf{P}_v(u_1, 0)G_3(v_1) + \mathbf{P}_v(u_1, 1)G_4(v_1)]$$

$$= \Delta v_0 \mathbf{P}_v(u_1, v_1)$$

$$\mathbf{b}_{31}^* = \Delta u_0 \mathbf{P}_{u00}$$

$$\mathbf{b}_{32}^* = \Delta u_0[\mathbf{P}_{u00} F_1(v_1) + \mathbf{P}_{u01} F_2(v_1) + \mathbf{P}_{uv00} F_3(v_1) + \mathbf{P}_{uv01} F_4(v_1)]$$

$$= \Delta u_0 \mathbf{P}_u(0, v_1)$$

$$\mathbf{b}_{33}^* = \Delta u_0 \Delta v_0 \mathbf{P}_{uv00}$$

$$\mathbf{b}_{34}^* = \Delta u_0 \Delta v_0[\mathbf{P}_{u00} G_1(v_1) + \mathbf{P}_{u01} G_2(v_1) + \mathbf{P}_{uv00} G_3(v_1) + \mathbf{P}_{uv01} G_4(v_1)]$$

$$= \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(0, v_1)$$

$$\mathbf{b}_{41}^* = \Delta u_0 \mathbf{P}_u(u_1, 0)$$

$$\mathbf{b}_{42}^* = \Delta u_0[\mathbf{P}_u(u_1, 0)F_1(v_1) + \mathbf{P}_u(u_1, 1)F_2(v_1) + \mathbf{P}_{uv}(u_1, 0)F_3(v_1) + \mathbf{P}_{uv}(u_1, 1)F_4(v_1)]$$

$$= \Delta u_0 \mathbf{P}_u(u_1, v_1)$$

$$\mathbf{b}_{43}^* = \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(u_1, 0)$$

$$\mathbf{b}_{44}^* = \Delta u_0 \Delta v_0[\mathbf{P}_u(u_1, 0)G_1(v_1) + \mathbf{P}_u(u_1, 1)G_2(v_1) + \mathbf{P}_{uv}(u_1, 0)G_3(v_1) + \mathbf{P}_{uv}(u_1, 1)G_4(v_1)]$$

$$= \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(u_1, v_1)$$

The boundary condition matrix $[B]^*$ for subpatch 1 takes the form:

$$[B]^* = \left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}(0, v_1) & \Delta v_0 \mathbf{P}_{v00} & \Delta v_0 \mathbf{P}_v(0, v_1) \\ \mathbf{P}(u_1, 0) & \mathbf{P}(u_1, v_1) & \Delta v_0 \mathbf{P}_v(u_1, 0) & \Delta v_0 \mathbf{P}_v(u_1, v_1) \\ \hdashline \Delta u_0 \mathbf{P}_{u00} & \Delta u_0 \mathbf{P}_u(0, v_1) & \Delta u_0 \Delta v_0 \mathbf{P}_{uv00} & \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(0, v_1) \\ \Delta u_0 \mathbf{P}_u(u_1, 0) & \Delta u_0 \mathbf{P}_u(u_1, v_1) & \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(u_1, 0) & \Delta u_0 \Delta v_0 \mathbf{P}_{uv}(u_1, v_1) \end{array}\right]$$

This result is consistent with Example 5.23. The corner points of subpatch 1 are those evaluated from the original patch. However, the subpatch tangent vectors have a factor of Δu_0 or Δv_0 multiplying those obtained from the original patch. There is also a factor of $\Delta u_0 \, \Delta v_0$ associated with the twist vectors. These factors ensure the adherence to the shape of the original patch. It is obvious that surface normals do not reverse directions. The $[B]$ matrices for the other three subpatches can now easily be written by substituting the proper Δu, Δv, and the vectors into the above matrix.

If the original patch is to be divided into two segments along either the $u = u_1$ or $v = v_1$ curve, the two boundary condition matrices can be written in a similar way.

6.7.4 Trimming

Trimming of surface entities is useful for engineering applications. It helps eliminate unnecessary calculations on the user's part. Surface trimming can be treated as a segmentation or intersection problem. If the surface is to be trimmed between two point trimming boundaries, we then have a segmentation problem at hand. If, on the other hand, a surface is to be trimmed to another surface, the intersection curve of the two surfaces must be found first and then the desired surface is trimmed to it. Figure 6-47 shows an example of trimming surfaces. A Bezier surface (identified by the digitize d_1) is trimmed between the two points identified by the two digitizes d_2 and d_3.

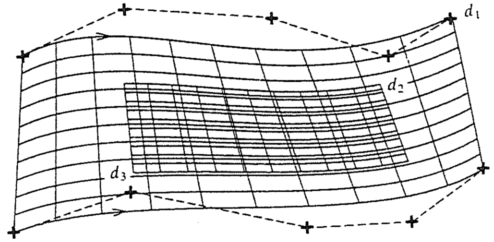


FIGURE 6-47
Surface trimming.

6.7.5 Intersection

The intersection problem involving surfaces is complex and nonlinear in nature. It depends on whether it is a surface/curve or surface/surface intersection as well as on the representation, parametric or implicit, of the involved surfaces and/or curves. In surface-to-curve intersection, the intersection problem is defined by the equation

$$\mathbf{P}(u, v) - \mathbf{P}(w) = \mathbf{0} \tag{6.112}$$

where $\mathbf{P}(u, v) = \mathbf{0}$ and $\mathbf{P}(w) = \mathbf{0}$ are the parametric equations of the surface and the curve respectively. Equation (6.112) is a system of three scalar equations, generally nonlinear, in three unknown u, v, and w. There are efficient iterative methods that can solve Eq. (6.112) such as the Newton-Raphson method. The detailed solution of Eq. (6.112) is left to the reader as an exercise.

The problem of surface-to-surface intersection is defined by the equation

$$\mathbf{P}(u, v) - \mathbf{P}(t, w) = \mathbf{0} \tag{6.113}$$

where $\mathbf{P}(u, v) = \mathbf{0}$ and $\mathbf{P}(t, w) = \mathbf{0}$ are the equations of the two surfaces. The vector equation (6.113) corresponds to three scalar equations in four variables u, v, t, and w. To solve this overspecified problem, some methods hold one of the unknowns constant and therefore reduce the original surface/surface intersection problem into a surface/curve intersection. Other methods introduce a new constrained function of the four variables. In both approaches, the solutions are iterative and may yield any combination of curves (closed or open) and isolated points. Solutions for sculptured surfaces usually employ curve-tracing, subdivision (divide-and-conquer method), or a combination of both techniques. Some of these solutions assume a given class of surface types, such as Bezier or B-spline representation. Figure 6-48 shows an example of surface/surface intersection. Some CAD/CAM systems implement the special case of a plane intersecting a surface as a "cut plane" command.

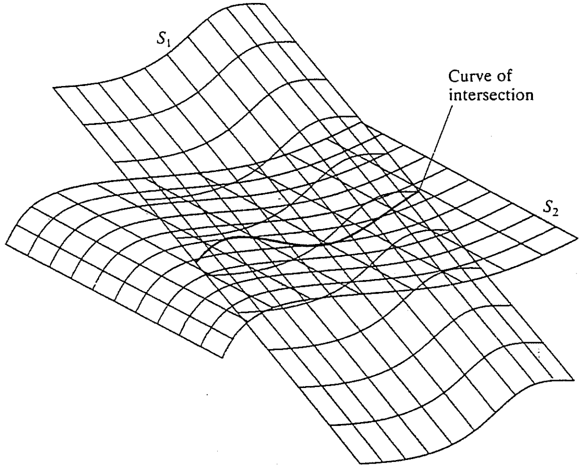
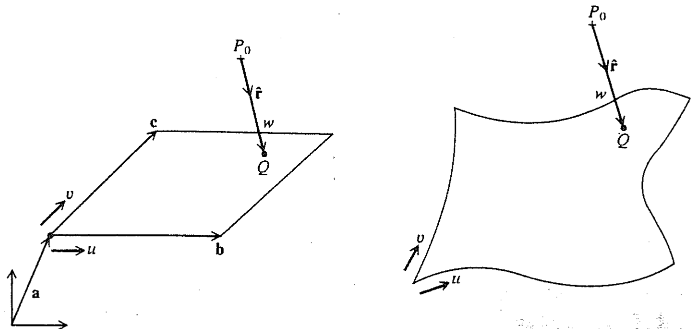


FIGURE 6-48
Surface/surface intersection.

6.7.6 Projection

Projecting an entity onto a plane or a surface is useful in applications such as determining shadows or finding the position of the entity relative to the plane or the surface. Entities that can be projected include points, lines, curves, or surfaces. Projecting a point onto a plane or a surface forms the basic problem shown in Fig. 6-49*a*. Point P_0 is projected along the direction *r* onto the given plane. It is desired to calculate the coordinates of the projected point *Q*. The plane equation, based on Eq. (6.28), can be written as follows:

$$\mathbf{P}(u, v) = \mathbf{a} + u\mathbf{b} + v\mathbf{c} \tag{6.114}$$



(*a*) Planar surface

(*b*) General surface

FIGURE 6-49
Projecting a point onto a surface.

where the vectors **a**, **b**, and **c** are shown in Fig. 6-49*a*. The equation of the projection line is given by

$$\mathbf{P}(w) = \mathbf{P}_0 + w\hat{\mathbf{r}} \tag{6.115}$$

The projection point *Q* is the intersection point between the line and the plane; that is, the following equation must be solved for *u*, *v*, and *w*:

$$\mathbf{P}(u, v) - \mathbf{P}(w) = \mathbf{0} \tag{6.116}$$

or

$$\mathbf{a} + u\mathbf{b} + v\mathbf{c} = \mathbf{P}_0 + w\hat{\mathbf{r}} \tag{6.117}$$

To solve for *w*, dot-multiply both sides of the above equation by $(\mathbf{b} \times \mathbf{c})$ to get

$$(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} = (\mathbf{b} \times \mathbf{c}) \cdot (\mathbf{P}_0 + w\hat{\mathbf{r}}) \tag{6.118}$$

since $(\mathbf{b} \times \mathbf{c})$ is perpendicular to both **b** and **c**. Equation (6.118) gives

$$w = \frac{(\mathbf{b} \times \mathbf{c}) \cdot (\mathbf{a} - \mathbf{P}_0)}{(\mathbf{b} \times \mathbf{c}) \cdot \hat{\mathbf{r}}} \tag{6.119}$$

Similarly, we can write

$$u = \frac{(\mathbf{c} \times \hat{\mathbf{r}}) \cdot (\mathbf{P}_0 - \mathbf{a})}{(\mathbf{c} \times \hat{\mathbf{r}}) \cdot \mathbf{b}} \tag{6.120}$$

and

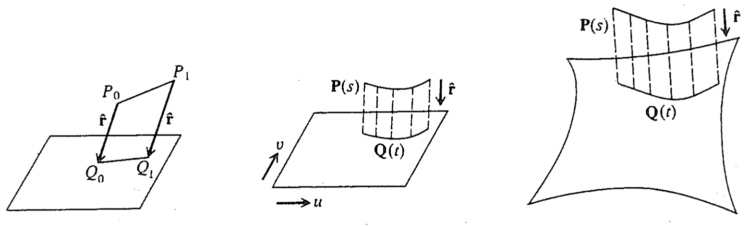
$$v = \frac{(\mathbf{b} \times \hat{\mathbf{r}}) \cdot (\mathbf{P}_0 - \mathbf{a})}{(\mathbf{b} \times \hat{\mathbf{r}}) \cdot \mathbf{c}} \tag{6.121}$$

The projection point *Q* results by substituting Eq. (6.119) into (6.115) or Eqs. (6.120) and (6.121) into (6.114).

If point P_0 is to be projected onto a general surface as shown in Fig. 6-49*b*, Eq. (6.114) is replaced by the surface equation and Eq. (6.116) becomes a nonlinear equation similar to Eq. (6.112).

The above approach can be extended to projecting curves and surfaces onto a given surface as shown in Figs. 6-50 and 6-51. The projection of a straight line passing by the two endpoints P_0 and P_1 (see Fig. 6-50*a*) along the direction $\hat{\mathbf{r}}$ involves projecting the two points using Eqs. (6.119) to (6.121) and then connecting Q_0 and Q_1 by a straight line which, of course, must lie in the given plane.

The projection of a general curve $\mathbf{P}(s)$ onto a plane (Fig. 6-50*b*) or a general surface (Fig. 6-50*c*) requires repetitive solution of Eq. (6.112). One simple strategy



(*a*) Line onto a plane

(*b*) Curve onto a plane

(*c*) Curve onto a general surface

FIGURE 6-50
Projecting a curve onto a surface.

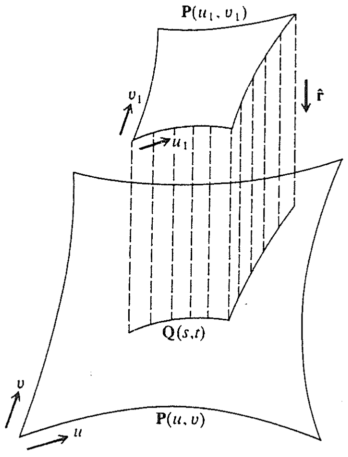


FIGURE 6-51
Projecting a surface onto a surface.

is to generate a set of points on $\mathbf{P}(s)$, project them onto the given plane or surface, and then connect them by a B-spline curve to obtain the projection curve $Q(t)$.

The projection of a surface $\mathbf{P}(u_1, v_1)$ onto a surface $\mathbf{P}(u, v)$ (Fig. 6-51) can be seen as an extension of the above strategy. A set of points is first generated on the surface $\mathbf{P}(u_1, v_1)$. Utilizing Eq. (6.112), this set is projected onto the surface $\mathbf{P}(u, v)$. The projection points are then connected by a B-spline surface to produce the projection surface $\mathbf{Q}(s, t)$. Note here that the greater the number of projection points, the closer the surface $\mathbf{Q}(s, t)$ to the surface $\mathbf{P}(u, v)$.

6.7.7 Transformation

The homogeneous transformation of surfaces is an extension of those of curves. Transformations offer very useful tools to designers while creating surface models. Functions such as translation, rotation, mirror, and scaling are offered by most CAD/CAM systems and are based on transformation concepts. To transform a surface, points on the surface may be evaluated first, transformed, and then the surface is redisplayed in the new transformed position and/or orientation. Other methods may exist. More on transformation is covered in Chap. 9.

6.8 DESIGN AND ENGINEERING APPLICATIONS

The following examples show how surface functions offered by CAD/CAM systems and their related theory covered in this chapter are used for engineering and design applications. The reader can work these examples on a CAD/CAM system. The reader is also advised to expand the line of thinking presented here into other applications of interest.

Example 6.11. Figure 6-52 shows a set of data points whose coordinates are measured directly from a given closed surface. Reconstruct the surface from the data points.

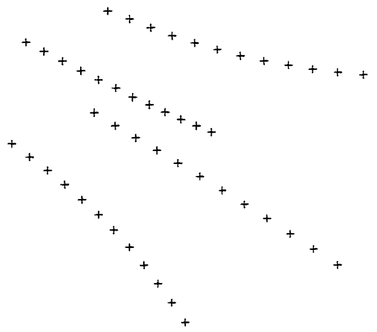
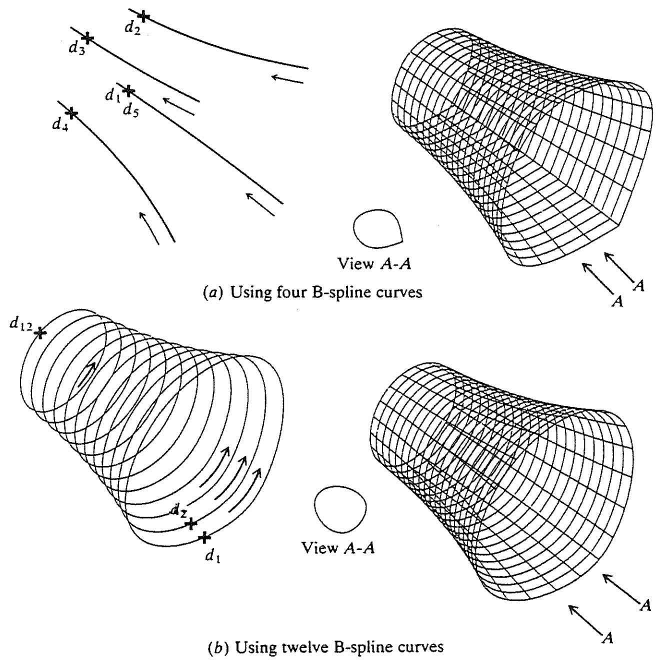


FIGURE 6-52
Data points.

Solution. A B-spline surface is the proper surface to connect the above given set of data points. "B-spline surface" commands provided by most CAD/CAM systems allow users to create B-spline surfaces by connecting a set of data points as described in Sec. 6.6.3 or by connecting a set of B-spline curves. In the latter case, the CAD/CAM system is actually using the points that define the spline curves for the surface definition. Therefore, the splines must have been created with the same number of points and in the same order, that is, connect points from left to right or right to left; otherwise the resulting surface will be twisted.



(a) Using four B-spline curves

(b) Using twelve B-spline curves

FIGURE 6-53
Constructing a B-spline surface from a given set of points.

The data points shown in Fig. 6-52 can be seen to form four B-spline curves which can then be used to reconstruct the surface (see Fig. 6-53*a*). They can also be interpreted as points on 12 cross sections of the surface, four points per cross section. In this case, the surface is shown in Fig. 6-53*b*. In both figures, the points are connected in the order shown by arrows to create the splines and the splines are digitized in the order shown to create the surface. In Fig. 6-53*b* we used the same tangents at the beginning and end points of each spline to force C^1 continuity. If we had not used it, we would have obtained an identical surface to that shown in Fig. 6-53*a*.

Example 6.12. Figure 6-54 shows a surface whose cross sections are circular. Construct the surface.

Solution. This is an axisymmetric surface. The most efficient way to construct it is to choose an MCS as shown in Fig. 6-54 and create points *a*, *b*, *c*, and *d*. These points are then copied using a mirror command to generate points *b*′, *c*′, and *d*′. The seven points are connected with a B-spline curve. Utilizing a surface of revolution command, this curve can be rotated 360° about the *Y* axis to generate the required surface (see Fig. 6-55).

Another way of constructing the surface is to replace the circular cross sections by B-spline curves which can be connected with a B-spline surface. Figure 6-56 shows the B-spline surface where 36 points are used to represent each circle.

Both ways of constructing the surface are compared by superposing them as shown in Fig. 6-57. As can be seen, while each surface has its unique database, both ways yield identical surfaces.

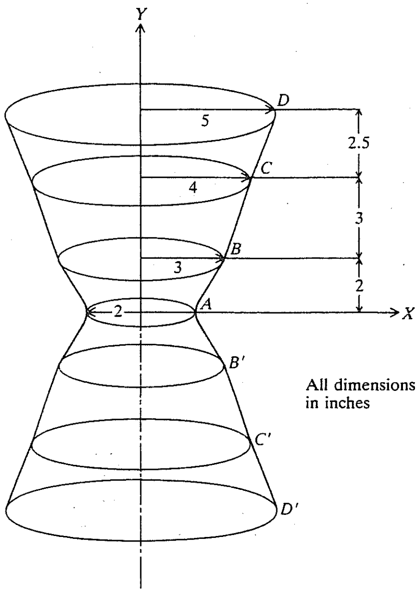


FIGURE 6-54
Circular cross sections of a surface.

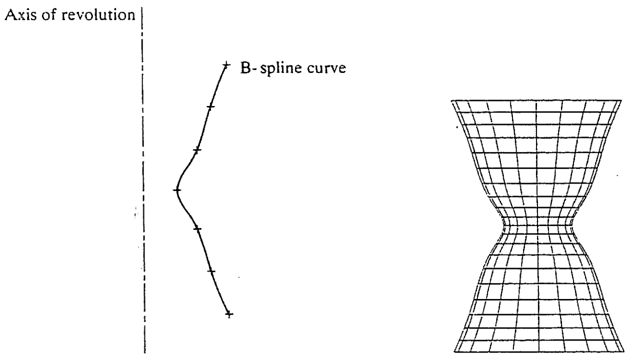


FIGURE 6-55
Surface of revolution of Example 6.12.

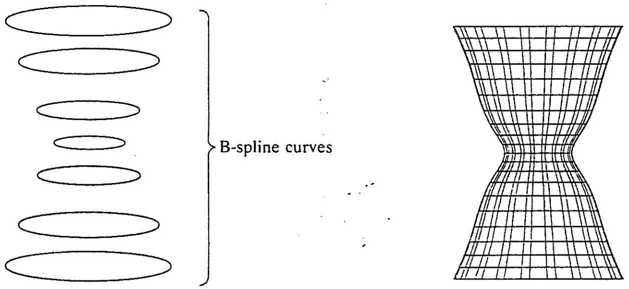


FIGURE 6-56
B-spline surface of Example 6.12.

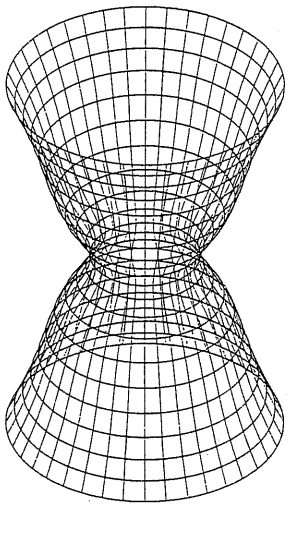


FIGURE 6-57
Comparison of both ways of constructing the surface of Example 6.12.

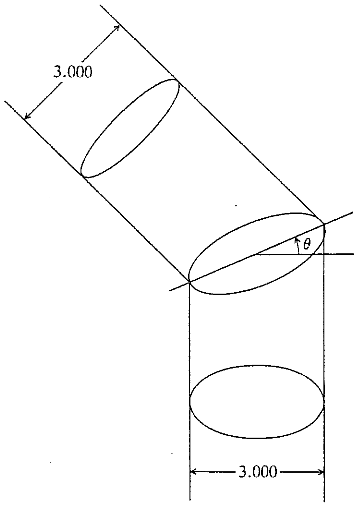
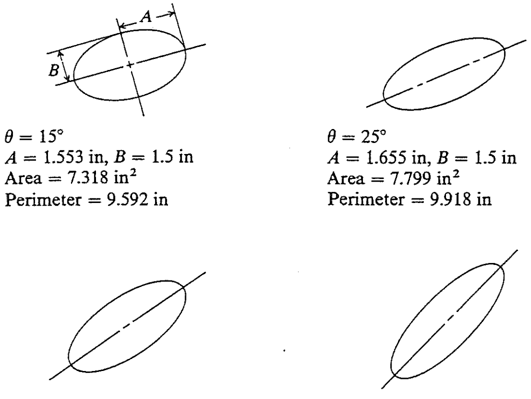


FIGURE 6-58
Pipeline design.

Example 6.13. Two pipes whose axes form an angle θ are shown in Fig. 6-58. Each pipe has a radius of 1.5 inch. The two pipes are to be butt welded along the common ellipse of intersection. The area of the ellipse must be 10 in^2 ± 1 percent for pressure drop considerations inside the pipes at the intersection cross section. Find the angle θ for which the resulting ellipse has the required area. Find also the lengths of the ellipse axes. For welding purposes, find also the perimeter of the ellipse. Ignore the thickness of the pipes for simplicity.



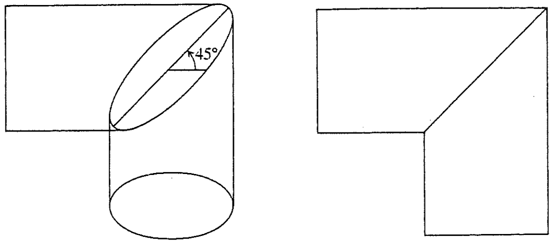
$\theta = 15°$
$A = 1.553$ in, $B = 1.5$ in
Area = 7.318 in^2
Perimeter = 9.592 in

$\theta = 25°$
$A = 1.655$ in, $B = 1.5$ in
Area = 7.799 in^2
Perimeter = 9.918 in

$\theta = 35°$
$A = 1.831$ in, $B = 1.5$ in
Area = 8.628 in^2
Perimeter = 10.490 in

$\theta = 45°$
$A = 2.121$ in, $B = 1.5$ in
Area = 9.995 in^2
Perimeter = 11.460 in

FIGURE 6-59
Characteristics of intersection ellipse of two intersecting pipes.



(*a*) Isometric view
(*b*) Front view

FIGURE 6-60
Final design of pipeline of Fig. 6-58.

Solution. The strategy to solve this problem is to construct the vertical pipe shown in Fig. 6-58 as a tabulated cylinder with a radius of 1.5 inch and an arbitrary length, say 6 inches, cut it with a plane at various values of θ ranging from 0 to 90°, verify the resulting ellipse entity to obtain the lengths of the ellipse axes, and calculate its area using a "calculate area" command provided by the particular CAD/CAM system. The ellipse that produces the given area is the desired one and the corresponding angle θ is the solution. The ellipse perimeter can be evaluated using the "measure length" command.

Figure 6-59 shows some of the intermediate results with the solution and Fig. 6-60 shows the final design of the pipeline. To check the system accuracy, compare the obtained results with those obtained using the following equations:

$$\text{Ellipse area } a = \pi AB$$

$$\text{Ellipse perimeter } p = 2\pi\sqrt{\frac{A^2 + B^2}{2}} = 4aE \qquad \text{(approximate)}$$

where E is the complete elliptic integral at $\mathscr{K} = \sqrt{A^2 - B^2}/A$. E can be obtained from the table of elliptic integrals available in CRC standard mathematical tables. For this problem E is found to be 1.3506.

Some CAD/CAM systems may recognize the intersection curve between the plane and the cylinder as a closed B-spline curve. In this case, the area and the perimeter can be calculated using the "calculate area" and "measure length" commands as above. To check the accuracy, the B-spline intersection curve can be replaced by an ellipse using $B = 1.5$ and $A = B/\cos\theta$. The reader is encouraged to re-solve this example on a CAD/CAM system. To find out whether your system treats the intersection curve as an ellipse or a B-spline curve, verify it using the "verify entity" command. *Note:* instead of using a plane to intersect the vertical cylinder, the two cylinders can be created and intersected.

Example 6.14. Figure 6-61 shows a duct of an air-conditioning system. The 4-inch diameter pipe is connected to a 4-inch diameter elbow. The elbow is joined to a truncated cone having a 4-inch diameter and 6-inch diameter ends. The 6-inch diameter end is increased to a 10 × 10 inch square end. If the thickness of the duct is ignored:

(*a*) Develop a surface model of the duct.
(*b*) Find the duct cross section at the valve location shown.

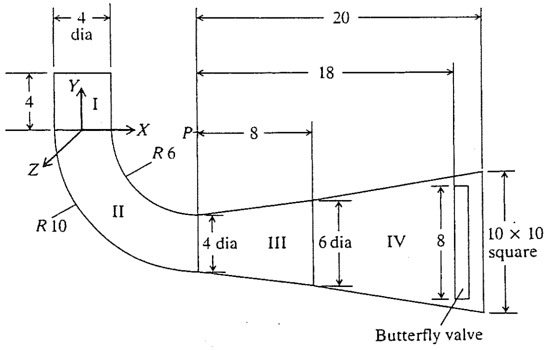


FIGURE 6-61
Geometry of an air-conditioning duct.

Solution

(*a*) Figure 6-61 also shows the four different surfaces required to create the duct. Surface I is a tabulated cylinder with a 4-inch diameter and 4-inch length. Surface II is a surface of revolution. It is created by rotating a horizontal circle located at the end of surface I about an axis passing through point *P* an angle of 90°. Surfaces III and IV can be created as ruled surfaces. Figure 6-62 shows the top, front, right, and isometric views of the duct.

(*b*) If the duct geometric model is cut by a vertical plane at the valve location, the resulting duct cross section is shown in Fig. 6-63. The area of the cross section is 85.229 in^2. The figure also shows cross sections 2 inches apart for surface IV. Notice that the right (first) cross section is square and the left (last) cross section is circular as expected and gradual change in shape occurs in between. The areas, centroids, and perimeters of these cross sections can easily be calculated using the "calculate area" and "measure length" commands.

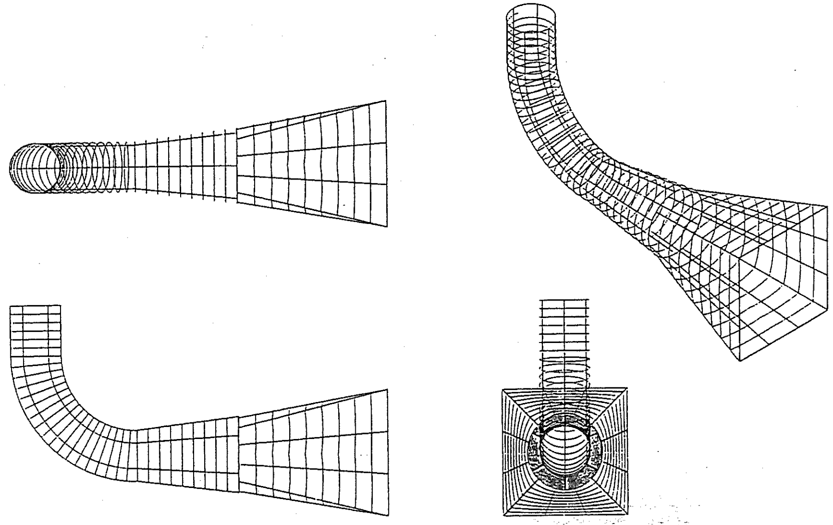


FIGURE 6-62
Duct geometric model.

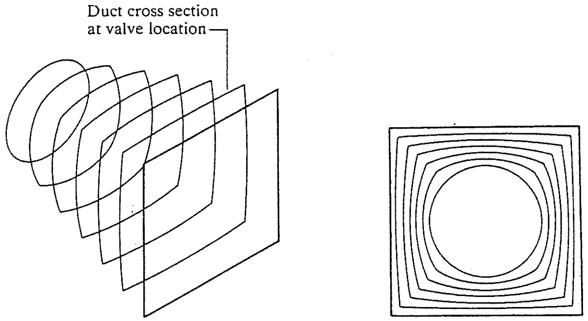


FIGURE 6-63
Cross sections of duct surface IV.

PROBLEMS

Part 1: Theory

6.1. Equation (6.36) gives the equation of a ruled surface joining two space curves. Find the equation of a ruled surface that passes through one space curve along a given set of direction vectors defining the surface rulings.

6.2. How can you implement Eq. (6.37) into a surface package?

6.3. Derive the parametric equations of an ellipsoid, a torus, and a circular cone.

6.4. Find the tangent and normal vectors to a surface of revolution in terms of its profile equation.

6.5. Repeat Prob. 6.4 but for a tabulated cylinder in terms of its directrix.

6.6. Prove that the curvature of a circular cylinder is zero. What is the radius of curvature at any point on its surface?

6.7. Derive Eq. (6.65) of Example 6.6 which states the planar condition of a bicubic surface patch.

6.8. Develop the relationships between the position, tangent, and twist vectors at the corner points of a bicubic surface patch and C_{ij} coefficients used in Eq. (6.39).

6.9. Derive the conditions for C^0 and C^1 continuity of a cubic Bezier composite surface of two patches.

6.10. Show that a bilinear Coons patch cannot provide C^1 continuity between adjacent patches even if their boundary curves form a C^1 continuity network.

6.11. Derive Eq. (6.103) for a bicubic Coons patch whose boundary curves and cross-boundary derivatives are given.

6.12. Given a point Q and a parametric surface in cartesian space (Fig. P6-12), find the closest point P on the surface to Q.

Hint: Find P such that $(\mathbf{Q} - \mathbf{P})$ is perpendicular to the tangent vectors $\mathbf{P}_u$ and $\mathbf{P}_v$ at P.

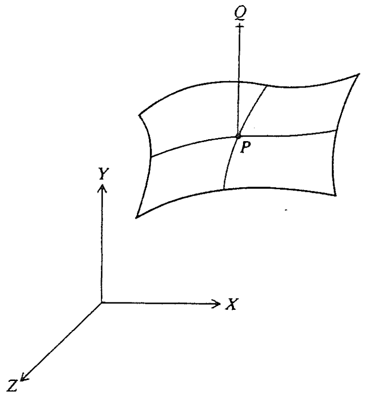


FIGURE P6-12

6.13. Find the minimum distance between:
(*a*) A point and a surface
(*b*) A curve and a surface
(*c*) Two surfaces

6.14. Refer to the published literature and find algorithms for the intersection between:
(*a*) A curve and a surface
(*b*) Two surfaces

Part 2: Laboratory

6.15. Create the surface models of the wireframe models of Prob. 5.20 in Chap. 5. Make a copy of each model database for surface construction.
Hint: Use layers and colors provided by your CAD/CAM system in order to manage the amount of graphics displayed on the screen and to facilitate identifying each surface-generating curve.

6.16. Create the surface models of the additional geometric models shown in Fig. P6-16. Follow the part setup procedure on your CAD/CAM system. All dimensions are in inches.

6.17. Create the quadric (quadratic) surfaces shown in Fig. P6-17 on your CAD/CAM system.

Part 3: Programming

Write a computer program to generate a:

6.18. Plane surface that passes through three points.
6.19. Ruled surface that connects two given rails.
6.20. Surface of revolution.
6.21. Tabulated cylinder.
6.22. Bicubic surface.
6.23. Cubic rectangular Bezier patch.
6.24. Open cubic B-spline surface patch.

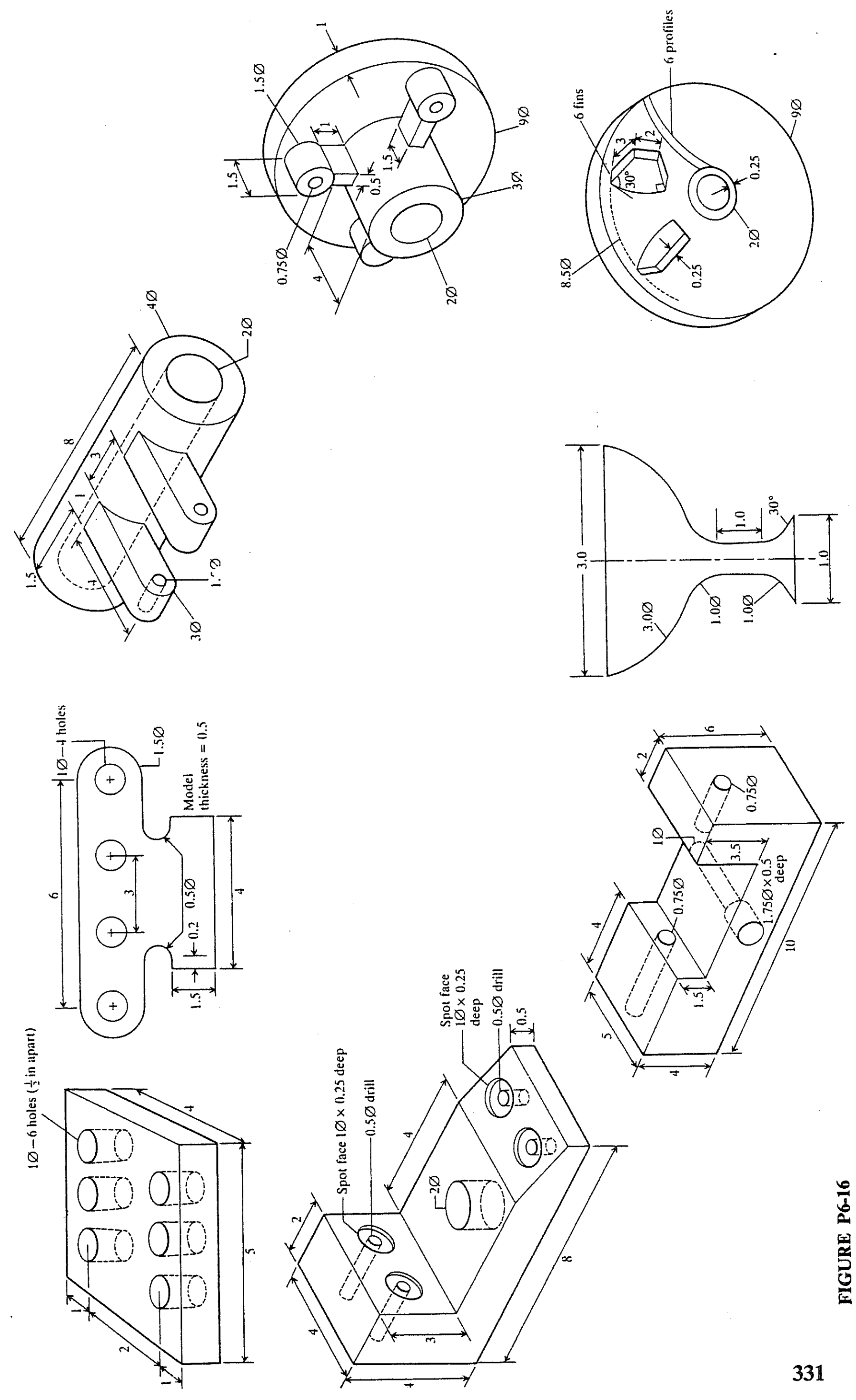


FIGURE P6-16

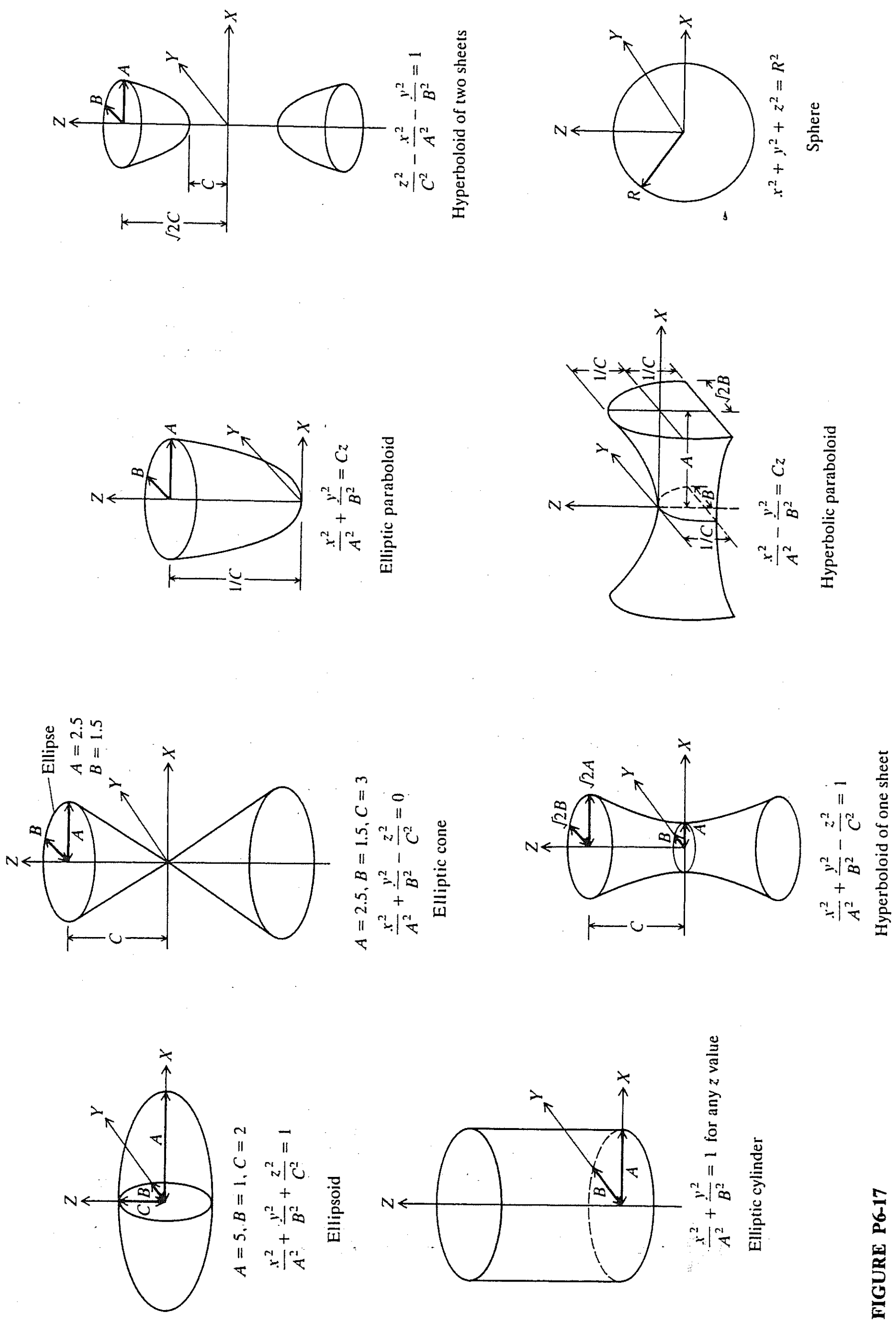

Hyperboloid of two sheets
$\frac{z^2}{C^2} - \frac{x^2}{A^2} - \frac{y^2}{B^2} = 1$
$\sqrt{2}C$
C
Sphere
$x^2 + y^2 + z^2 = R^2$
R
Elliptic paraboloid
$\frac{x^2}{A^2} + \frac{y^2}{B^2} = Cz$
1/C
Hyperbolic paraboloid
$\frac{x^2}{A^2} - \frac{y^2}{B^2} = Cz$
$\sqrt{2}B$
Elliptic cone
$A = 2.5, B = 1.5, C = 3$
$\frac{x^2}{A^2} + \frac{y^2}{B^2} - \frac{z^2}{C^2} = 0$
Ellipse
$A = 2.5$
$B = 1.5$
Hyperboloid of one sheet
$\frac{x^2}{A^2} + \frac{y^2}{B^2} - \frac{z^2}{C^2} = 1$
$\sqrt{2}A$
$\sqrt{2}B$
Ellipsoid
$A = 5, B = 1, C = 2$
$\frac{x^2}{A^2} + \frac{y^2}{B^2} + \frac{z^2}{C^2} = 1$
Elliptic cylinder
$\frac{x^2}{A^2} + \frac{y^2}{B^2} = 1$ for any z value
X
Y
Z
A
B
C

FIGURE P6-17

6.25. Closed cubic B-spline surface patch.

6.26. Bilinear Coons patch.

6.27. Cubic Coons patch.

6.28. Quadratic triangular Bezier patch.

6.29. Cubic triangular Bezier patch.

BIBLIOGRAPHY

Barnhill, R. E.: "A Survey of the Representation and Design of Surfaces," *IEEE Computer Graphics and Applic. Mag.*, pp. 9–16, October 1983.

Barnhill, R. E.: "Surfaces in Computer Aided Geometric Design: A Survey with New Results," *Computer Aided Geometric Des. J.*, vol. 2, pp. 1–17, 1985.

Barnhill, R. E., and W. Boehm (Eds.): *Surfaces in CAGD'84*, Elsevier Science, 1985.

Beck, J. M., R. T. Farouki, and J. K. Hinds: "Surface Analysis Methods," *IEEE CG&A*, pp. 18–36, December 1986.

Bohm, W., G. Farin, and J. Kahmann: "A Survey of Curve and Surface Methods in CAGD," *CAGD J.*, vol. 1, pp. 1–60, 1984.

Brunet, P.: "Increasing the Smoothness of Bicubic Spline Surfaces," *CAGD J.*, vol. 2, pp. 157–164, 1985.

CAM-I, Inc.: "Common Normals for Arbitrary Parametric Surfaces," Document no. R-64-APT-03, Arlington, Tex., 1964.

CAM-I, Inc.: "Final Report on Parametric Surfaces," Document no. R-66-APT-02, Arlington, Tex., 1966.

CAM-I, Inc.: "Internal Data Structure Design for Sculptured Surfaces SS × 2 System," Document no. R-71-SS-02, Arlington, Tex., 1971.

CAM-I, Inc.: "Mathematical Specifications for Sculptured Surfaces Patches," Document no. R-71-SS-03, Arlington, Tex., 1971.

CAM-I, Inc.: "Sculptured Surfaces Phase IV," Document no. R-75-SS-01, Arlington, Tex., 1975.

CAM-I, Inc.: "Demonstration Test of Sculptured Surfaces," Document no. R-75-SS-03, Arlington, Tex., 1975.

CAM-I, Inc.: "Proceedings on Sculptured Surfaces," Document no. P-80-SS-01, Arlington, Tex., 1980.

CAM-I, Inc.: "Proceedings on Sculptured Surfaces," Document no. P-83-SS-01, Arlington, Tex., 1983.

CAM-I, Inc.: "Extended Geometric Facilities for Surface and Solid Modelers," Document no. R-83-GM-02.1, vols. 1, 2, 3, Arlington, Tex., 1983.

Chang, G., and Y. Feng: "An Improved Condition for the Convexity of Bernstein-Bezier Surfaces Over Triangles," *CAGD J.*, vol. 1, pp. 279–283, 1984.

Chang, G., and B. Su: "Families of Adjoint Patches for a Bezier Triangle Surface," *CAGD J.*, vol. 2, pp. 37–42, 1985.

Chasen, S. H.: *Geometric Principles and Procedures for Computer Graphics Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

Cohen, E., T. Lyche, and R. Riesenfeld: "Discrete Box Splines and Refinement Algorithms," *CAGD J.*, vol. 1, pp. 131–148, 1984.

Cottingham, M. S.: "A Compressed Data Structure for Surface Representation," *Computer Graphics Forum*, vol. 4, pp. 217–228, 1983.

Dahmen, W., and C. A. Micchelli: "Subdivision Algorithms for the Generation of Box Spline Surfaces," *CAGD J.*, vol. 1, pp. 115–129, 1984.

Dahmen, W., and C. A. Micchelli: "Line Average Algorithm: A Method for the Computer Generation of Smooth Surfaces," *CAGD J.*, vol. 2, pp. 77–85, 1985.

Dodd, S. L., D. F. McAllister, and J. A. Roulier: "Shape-Preserving Spline Interpolation for Specifying Bivariate Functions on Grids," *IEEE CG&A*, pp. 70–79, September 1983.

Encarnacao, J., and E. G. Schlechtendahl: *Computer Aided Design; Fundamentals and System Architectures*, Springer-Verlag, New York, 1983.

Faux, I. D., and J. J. Pratt: *Computational Geometry for Design and Manufacture*, Ellis Horwood (John Wiley), Chichester, West Sussex, 1979.

Foley, J. D., and A. van Dam: *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.