QUADRIC SURFACES

QUADRIC SURFACES

- A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids, and hyperboloids.

- Quadric surfaces, particularly spheres and ellipsoids, are common elements of graphics scenes

Sphere

In Cartesian coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the set of points (x, y, z) that satisfy the equation

$$x^2 + y^2 + z^2 = r^2$$

Quadric Surfaces

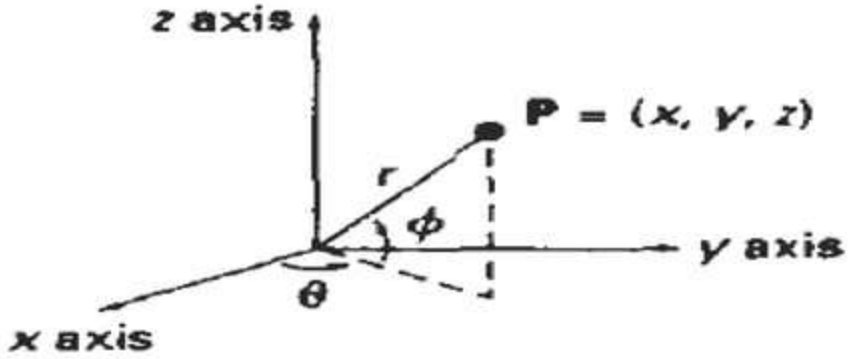


Figure 10-8
Parametric coordinate position (r, θ, ϕ) on the surface of a sphere with radius r.

We can also describe the spherical surface in parametric form, using latitude and longitude angles

$$x = r\cos\phi\cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$
$$y = r\cos\phi\sin\theta, \qquad -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$

- we could write the parametric equations using standard spherical coordinates, where angle Ø is specified as the

colatitude (Fig. 10-9). Then, ϕ is defined over the range $0 \leq \phi \leq \pi$, and θ is often taken in the range $0 \leq \theta \leq 2\pi$. We could also set up the representation using parameters u and v defined over the range from 0 to 1 by substituting $\phi = \pi u$ and $\theta = 2\pi v$.

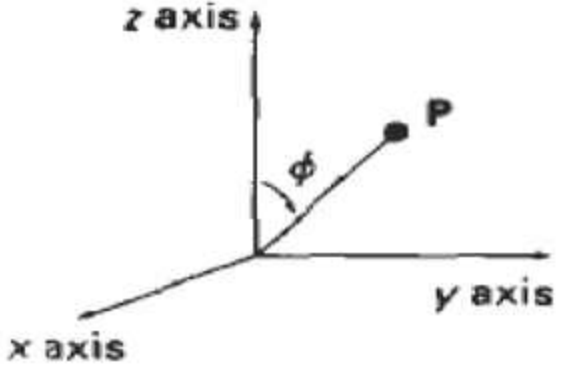


Figure 10-9
Spherical coordinate parameters (r, θ, ϕ), using colatitude for angle ϕ.

z ↓

Ellipsoid

- An ellipsoidal surface can be described as an extension of a spherical surface, where the radii in three mutually perpendicular directions can have different values(Fig. 10-10).

- The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

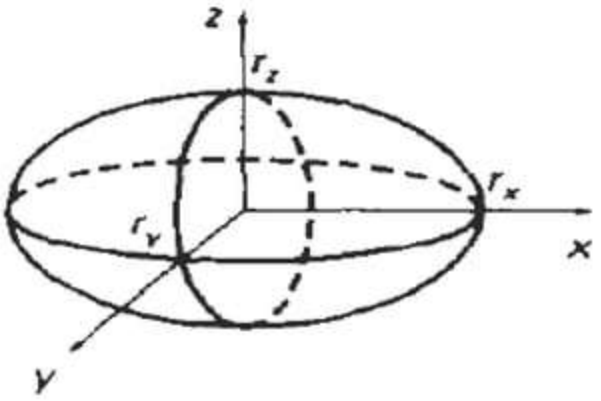


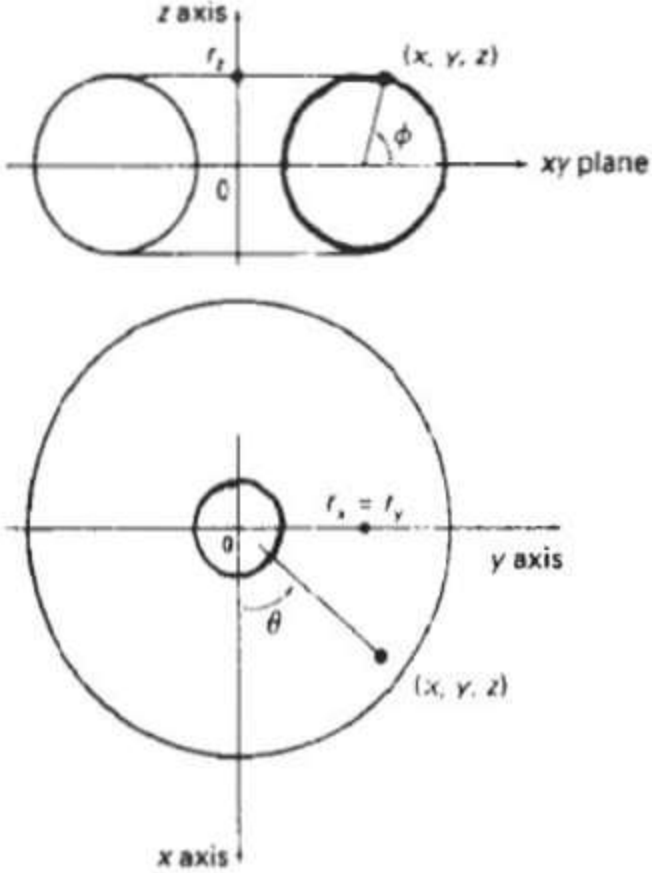
Figure 10-10
An ellipsoid with radii r_x, r_y, and r_z centered on the coordinate origin.

And a parametric representation for the ellipsoid in terms of the latitude angle ϕ and the longitude angle θ in Fig. 10-8 is

$$
\begin{aligned}
x &= r_x \cos\phi \cos\theta, \qquad -\pi/2 \le \phi \le \pi/2 \\
y &= r_y \cos\phi \sin\theta, \qquad -\pi \le \theta \le \pi \\
z &= r_z \sin\phi
\end{aligned}
\tag{10-10}
$$

Torus

- **A torus is a doughnut-shaped object, as shown in Fig. 10-11.**
- **It can be generated** by rotating a circle or other conic about a specified axis.



A torus with a circular cross section centered on the coordinate origin.

The Cartesian representation for points over the surface of a torus can be written in the form

$$\left[r - \sqrt{\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2}\right]^2 + \left(\frac{z}{r_z}\right)^2 = 1 \qquad (10\text{-}11)$$

where r is any given offset value. Parametric representations for a torus are similar to those for an ellipse, except that angle ϕ extends over 360°. Using latitude and longitude angles ϕ and θ, we can describe the torus surface as the set of points that satisfy

$$\begin{aligned} x &= r_x(r + \cos\phi)\cos\theta, \qquad -\pi \le \phi \le \pi \\ y &= r_y(r + \cos\phi)\sin\theta, \qquad -\pi \le \theta \le \pi \\ z &= r_z \sin\phi \end{aligned} \qquad (10\text{-}12)$$

BLOBBY OBJECTS

- Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other objects.
- Examples in this class of objects include molecular structures, water droplets and other liquid effects, melting objects, and muscle shapes in the human body.
- **These objects can be** described as exhibiting "blobbiness" and are often simply referred to as blobby objects, since their shapes show a certain degree of fluidity.

Figure 10-14
Molecular bonding. As two molecules move away from each other, the surface shapes stretch, snap, and finally contract into spheres.

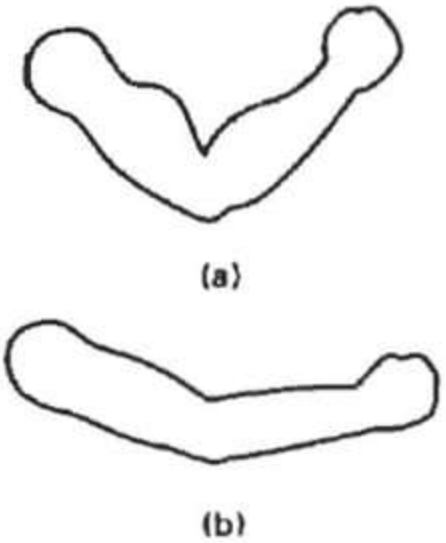


Figure 10-15
Blobby muscle shapes in a human arm.

- Several models have been developed for representing blobby objects as distribution

functions over a region of space. One way to do this is to model objects as combinations of Gaussian density functions, or "bumps".

A surface function is then defined as

$$f(x, y, z) = \sum_{k} b_k e^{-a_k r_k^2} - T = 0$$

where $r_k^2 = \sqrt{x_k^2 + y_k^2 + z_k^2}$, parameter T is some specified threshold, and parameters a and b are used to adjust the amount of blobbiness of the individual objects.

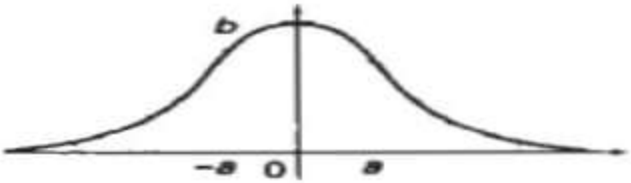


Figure 10-16
A three-dimensional Gaussian bump centered at position 0, with height b and standard deviation a.

Figure 10-17
A composite blobby object formed with four Gaussian bumps.

Other methods for generating blobby objects use density functions that fall off to 0 in a finite interval, rather than exponentially. The "metaball" model describes composite objects as combinations of quadratic density functions of the form

$$f(r) = \begin{cases} b(1 - 3r^2/d^2), & \text{if } 0 < r \le d/3 \\ \frac{3}{2}b(1 - r/d)^2, & \text{if } d/3 < r \le d \\ 0, & \text{if } r > d \end{cases} \tag{10-18}$$

And the "soft object" model uses the function

$$f(r) = \begin{cases} 1 - \frac{22r^2}{9d^2} + \frac{17r^4}{9d^4} - \frac{4r^6}{9d^6}, & \text{if } 0 < r \le d \\ 0, & \text{if } r > d \end{cases} \tag{10-19}$$

SPLINE REPRESENTATIONS

- a spline is a flexible strip **used to produce a smooth** curve through a designated **set of points.**
- **Several small weights are distributed** along the length of the strip to hold it in position on the drafting table as the curve is drawn.
- The term ***spline curve originally referred to a curve drawn in this*** manner.

- In computer graphics, the term spline curve now refers to any composite curve formed with polynomial sections satisfying specified

  continuity conditions at the boundary of the pieces.
- **A spline surface can be** described with two sets of orthogonal spline curves.
- Splines are used in graphics applications to design curve and surface shapes, to digitize drawings for computer storage, and to specify animation paths for the objects or the camera in a scene

Interpolation and Approximation Splines

- We specify a spline cuve by giving a set of coordinate positions, called control points, which indicates the general shape of the curve

- These control points are then fitted with piecewise continuous parametric polynomial functions in one of two ways.

- When polynomial sections are fitted so that the curve passes through each control point, as in Figure **the resulting curve is said to interpolate the** set of control points.

Figure 10-19
A set of six control points interpolated with piecewise continuous polynomial sections.

- On the other hand, when the polynomials are fitted to the general control-point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points

Figure 10-20
A set of six control points approximated with piecewise continuous polynomial sections

- A spline curve is defined, modified, and manipulated with operations on the control points.
- In addition, the curve can be translated, rotated, or scaled with transformations applied to the control points.
- The convex polygon boundary that encloses a set of control points is called the convex hull.

Parametric Continuity Conditions

- To ensure a smooth transition from one section of a piecewise parametric curve to the next, we can impose various continuity conditions at the connection points.
- If each section of a spline is described with a set of parametric coordinate functions of the form

$$x = x(u), \qquad y = y(u), \qquad z = z(u), \qquad u_1 \le u \le u_2$$

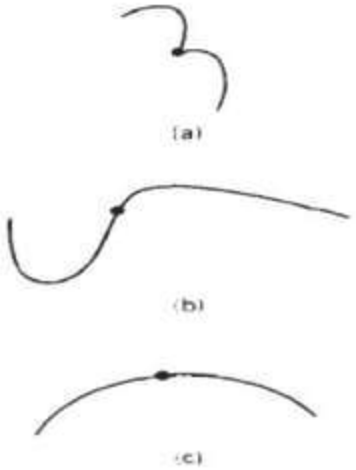
- **Zero-order** parametric continuity, described as C1 continuity, means simply that the ***curves meet.***
- That is, the values of x, y, and *z evaluated at u, for the first* curve section are equal, respectively, to the values of x, y, and *z evaluated at u,* for the next curve section.
- **First-order** parametric continuity, C1 continuity, means that the ***first parametric derivatives*** (tangent lines) of the coordinate functions for ***two successive curve*** sections are ***equal*** at their joining point.

- **Second-order** parametric continuity, or **C2 continuity, means that** both the first and second parametric derivatives of the two curve sections are the same at the intersection.
- The rates of change of the tangent vectors for

connecting sections are equal at their intersection. Thus, the tangent line transitions

smoothly from one section of the curve to the next

- But with first-order continuity, the rates of change of the tangent vectors for the two sections can be quite different , so that the general shapes of the two adjacent sections can change abruptly.



Piecewise construction of a curve by joining two curve segments using different orders of continuity: (a) zero-order continuity only, (b) first-order continuity, and (c) second-order continuity.

Geometric Continuity Conditions

- An alternate method for joining two successive curve sections is to specify conditions for geometric continuity.

- In this case, we only require parametric derivatives of the two sections to be proportional to each other at their common boundary instead of equal to each other.

- **Zero-order** geometric continuity, described as G0 continuity is the same as zero-order parametric continuity. That is, the two curves sections must have the same coordinate position at the boundary point.

- **First-order** geometric continuity or *G1 continuity, means that the parametric first derivatives are proportional* at the intersection of two successive sections.

- Second-order geometric continuity, or G2 continuity means that both the first and second parametric derivatives of the two curve sections are proportional at their boundary.
- Under ***G2 continuity, curvatures*** of two curve sections will match at the joining position.

- **A curve generated with geometric continuity conditions is similar to one** generated with parametric continuity, but with slight differences in curve shape.

Spline Specifications

- There are three equivalent methods for specifying a particular spline representation:

(1)We can state the set of boundary conditions that are imposed on the spline; or

(2) we can state the matrix that characterizes the spline; or

(3)we can state the set of blending functions (or basis functions) that determine how specified geometric constraints on the curve are combined to calculate positions along

the curve path.

- suppose **we have the following** parametric cubic polynomial representation for the **x coordinate along the** path of a spline section:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \qquad 0 \le u \le 1$$

Boundary conditions for this curve might be set, for example, on the endpoint coordinates $x(0)$ and $x(1)$ and on the parametric first derivatives at the endpoints $x'(0)$ and $x'(1)$. These four boundary conditions are sufficient to determine the values of the four coefficients a_x, b_x, c_x, and d_x.

From the boundary conditions, we can obtain the matrix that characterizes this spline curve by first rewriting Eq. 10-21 as the matrix product.

$$x(u) = [u^3\ u^2\ u\ 1]\begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} \qquad (10\text{-}22)$$

$$= \mathbf{U} \cdot \mathbf{C}$$

where **U** is the row matrix of powers of parameter u, and **C** is the coefficient column matrix. Using Eq. 10-22, we can write the boundary conditions in matrix form and solve for the coefficient matrix **C** as

$$\mathbf{C} = \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}} \qquad (10\text{-}23)$$

where $\mathbf{M}_{geom}$ is a four-element column matrix containing the geometric constraint values (boundary conditions) on the spline, and $\mathbf{M}_{spline}$ is the 4-by-4 matrix that transforms the geometric constraint values to the polynomial coefficients and provides a characterization for the spline curve. Matrix $\mathbf{M}_{geom}$ contains control-point coordinate values and other geometric constraints that have been specified. Thus, we can substitute the matrix representation for C into Eq. 10-22 to obtain

$$x(u) = \mathbf{U} \cdot \mathbf{M}_{spline} \cdot \mathbf{M}_{geom}$$

CUBIC SPLINE INTERPOLATION METHODS

- This class of splines is most often used to set up paths for object motions or to provide a representation for an existing object or drawing.
- cubic splines require less calculations and

  memory and they are more stable. Compared to lower-order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.

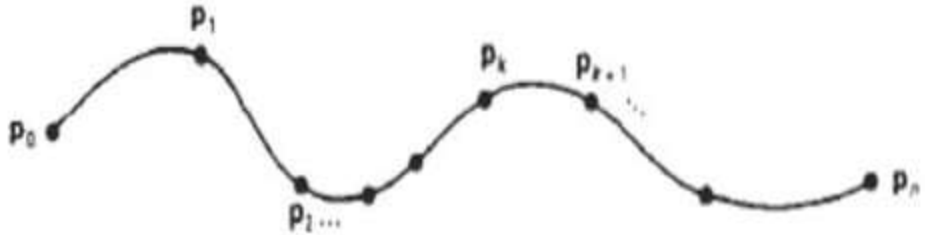


Figure 10-26
A piecewise continuous cubic-spline interpolation of $n + 1$ control points.

- A cubic interpolation fit of these points can be illustrated
- We can de**scribe** the parametric cubic polynomial that is to be fitted between each pair of control points with the following set of equations:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y, \quad (0 \le u \le 1) \qquad (10\text{-}26)$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

Hermite Interpolation

- Hermite spline is an interpolating piecewise cubic polynomial with a specified tangent at each control point.
- Unlike the natural cubic splines, Hermite splines can be adjusted locally because each curve section is only dependent on its endpoint constraints.

- If **P(u) represents a parametric cubic point function for the curve section between**

  control points p_k **and then the boundary conditions** that define this Hermite curve section are

$$\mathbf{P}(0) = \mathbf{p}_k$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$

$$\mathbf{P}'(0) = \mathbf{D}\mathbf{p}_k$$

$$\mathbf{P}'(1) = \mathbf{D}\mathbf{p}_{k+1}$$

with $\mathbf{Dp}_i$ and $\mathbf{Dp}_{i+1}$ specifying the values for the parametric derivatives (slope of the curve) at control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, respectively.

We can write the vector equivalent of Eqs. 10-26 for this Hermite-curve section as

$$\mathbf{P}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}, \qquad 0 \le u \le 1 \tag{10-28}$$

where the x component of $\mathbf{P}$ is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$, and similarly for the y and z components. The matrix equivalent of Eq. 10-28 is

$$\mathbf{P}(u) = [u^3 \; u^2 \; u \; 1] \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \tag{10-29}$$

and the derivative of the point function can be expressed as

$$P'(u) = [3u^2 \; 2u \; 1 \; 0] \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad (10\text{-}30)$$

Substituting endpoint values 0 and 1 for parameter u into the previous two equations, we can express the Hermite boundary conditions 10-27 in the matrix form:

$$\begin{bmatrix} p_i \\ p_{i-1} \\ Dp_i \\ Dp_{i-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad (10\text{-}31)$$

Solving this equation for the polynomial coefficients, we have

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} \qquad (10\text{-}32)$$

$$= \mathbf{M}_H \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

where $\mathbf{M}_H$, the Hermite matrix, is the inverse of the boundary constraint matrix. Equation 10-29 can thus be written in terms of the boundary conditions as

Equation 10-29 can thus be written in terms of the boundary conditions as

$$\mathbf{P}(u) = [u^3\ u^2\ u\ 1] \cdot \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\mathbf{p}_k \\ \mathbf{D}\mathbf{p}_{k+1} \end{bmatrix} \qquad (10\text{-}33)$$

Bezier Curves

- Bezier curve section can be fitted to any number of control points.
- The number of control points to be approximated and their relative position determine the degree of the Bezier polynomial.
- Bezier curve can **be specified with boundary conditions**

- Suppose we are given ***n + 1 control-point positions:*** p_k = ***(*** x_k, y_k, z_k***), with k*** varying from **0 to n.**
- **These coordinate points can be blended to produce the following** position vector ***P(u), which describes the path of an approximating Bezier*** polynomial function between $\mathbf{p}_0$ **and** $\mathbf{p}_n$

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k BEZ_{k,n}(u), \qquad 0 \leq u \leq 1$$

The Bézier blending functions $BEZ_{k,n}(u)$ are the *Bernstein polynomials*:

$$BEZ_{k,n}(u) = C(n, k)u^k(1 - u)^{n-k} \qquad (10\text{-}41)$$

where the $C(n, k)$ are the binomial coefficients:

$$C(n, k) = \frac{n!}{k!(n - k)!} \qquad (10\text{-}42)$$

Equivalently, we can define Bézier blending functions with the recursive calculation

$$BEZ_{k,n}(u) = (1 - u)\,BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u), \qquad n > k \geq 1 \quad (10\text{-}43)$$

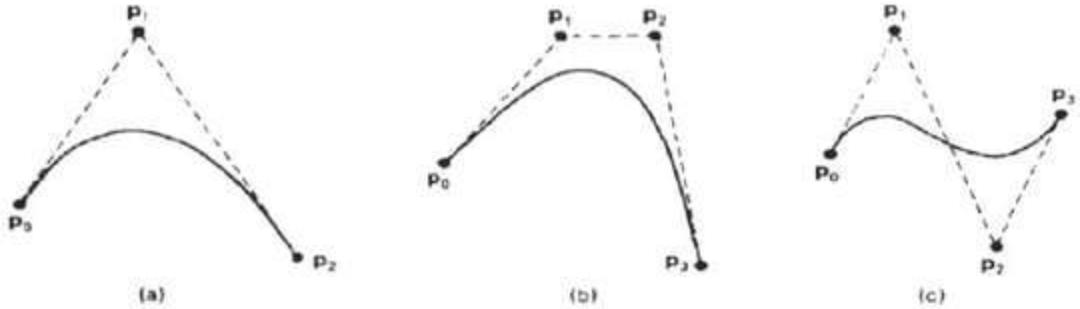
with $BEZ_{k,k} = u^k$, and $BEZ_{0,k} = (1 - u)^k$. Vector equation 10-40 represents a set of three parametric equations for the individual curve coordinates:

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(u)$$

- Bezier curve is a polynomial of degree one less than the number of control points used: Three points generate a parabola, four points a cubic curve, and so forth.



Examples of two-dimensional Bézier curves generated from three, four, and five control points. Dashed lines connect the control-point positions.

Properties **of Bezier Curves**

- **A very useful property of a Bezier curve is that it always passes through the first and last control points. That is, the boundary conditions at the two ends of the curve are**

$$\mathbf{P}(0) = \mathbf{p}_0$$
$$\mathbf{P}(1) = \mathbf{p}_n$$

Values of the parametric first derivatives of a Bézier curve at the endpoints can be calculated from control-point coordinates as

$$\mathbf{P}'(0) = -n\mathbf{p}_0 + n\mathbf{p}_1$$
$$\mathbf{P}'(1) = -n\mathbf{p}_{n-1} + n\mathbf{p}_n \tag{10-47}$$

Thus, the slope at the beginning of the curve is along the line joining the first two control points, and the slope at the end of the curve is along the line joining the last two endpoints. Similarly, the parametric second derivatives of a Bézier curve at the endpoints are calculated as

$$\mathbf{P}''(0) = n(n-1)[(\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)]$$
$$\mathbf{P}''(1) = n(n-1)[(\mathbf{p}_{n-2} - \mathbf{p}_{n-1}) - (\mathbf{p}_{n-1} - \mathbf{p}_n)] \tag{10-48}$$

- Another important property of any Bezier curve is that it lies within the convex hull (convex polygon boundary) of the control points.
- This follows from the properties of Bezier blending functions: They are all positive and their sum is always 1 so that any curve position is simply the weighted sum of the control-point positions

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

- The convex-hull property for a Bezier curve ensures that the polynomial / smoothly follows the control points without erratic oscillations.

Cubic **Bezier Curves**

- Cubic Bezier **curves are generated with four control** points. The four blending functions for cubic Bezier curves, obtained by substituting

  n = 3 into Eq. 10-41 are

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

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

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

$$BEZ_{3,3}(u) = u^3$$

- The form of the blending functions determine how the control points influence the shape of the curve for values of parameter ***u over the range from 0 to 1***
- At the end positions of the cubic Bezier curve, the parametric first derivatives (slopes) are

$$P'(0) = 3(p_1 - p_0), \qquad P'(1) = 3(p_3 - p_2)$$

And the parametric second derivatives are

$$P''(0) = 6(p_0 - 2p_1 + p_2), \qquad P''(1) = 6(p_1 - 2p_2 + p_3)$$

- We can use these expressions for the parametric derivatives to construct piecewise

curves with C1 or **C2 continuity between sections.**

By expanding the polynomial expressions for the blending functions, we can write the cubic Bezier point function in the matrix form

$$\mathbf{P}(u) = [u^3\ u^2\ u\ 1] \cdot \mathbf{M}_{Bez} \cdot \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \qquad (10\text{-}51)$$

where the **Bézier matrix** is

$$\mathbf{M}_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad (10\text{-}52)$$

Bezier Surfaces

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

$$\mathbf{P}(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} \mathbf{p}_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

with $\mathbf{p}_{j,k}$ specifying the location of the $(m + 1)$ by $(n + 1)$ control points.

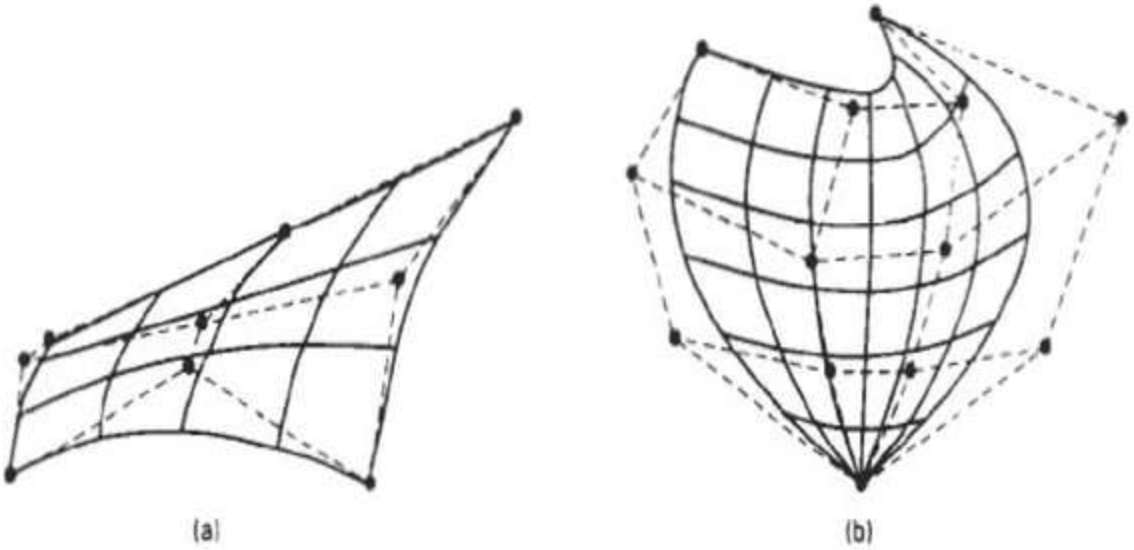


Figure 10-39
Bézier surfaces constructed for (a) $m = 3, n = 3$, and (b) $m = 4, n = 4$. Dashed lines connect the control points.

- Bezier surfaces have the same properties as Bezier curves, and they provide a convenient method for interactive design applications.
- For each surface patch, we can select a mesh of control points in the **xy "ground" plane, then we choose** elevations above the ground plane for the z-coordinate values of the control points.

B-SPLINE CURVES

- B-splines have two advantages over Bezier splines:

(1) the degree of a B-spline polynomial can

be set independently of the number of control points (with certain limitations)

(2) B-splines allow local control over the shape of a spline curve or surface

- We can write a general expression for the calculation of coordinate positions along a

  B-spline curve in a blending-function formulation as

- $$P(u) = \sum_{k=0}^{n} p_k B_{k,d}(u), \qquad u_{min} \le u \le u_{max}, \qquad 2 \le d \le n+1 \qquad (10\text{-}54)$$

formulation and that for Bezier splines. The range of parameter ***u now depends on how we choose the Bspline parameters.***

- Bspline blending functions $B_{k,d}$ ***are polynomials of degree d - 1, where parameter*** d can be chosen to be any integer value in the range from 2 up to the number of control points, **n + 1.**
- Local control for Bsplines is achieved by defining the blending functions over subintervals of the total range of u.
- Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas:

$$B_{k,1}(u) = \begin{cases} 1, & \text{if } u_k \le u < u_{k+1} \\ 0, & \text{otherwise} \end{cases} \qquad (10\text{-}55)$$

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

- where each blending function is defined over d subintervals of the total range of ***u.***
- ***The selected set of subinterval endpoints u, is referred to as a knot vector.***
- ***Values for u_{min} and u_{max} then depend on the number of control points*** we select, the value we choose for parameter d, and how we set up the subintervals (knot vector)

B-spline curves have the following properties

- The polynomial curve has degree d - 1 and C^{d-2} ***continuity over the range*** of ***u.***
- For n + 1 control points, the curve is described with ***n + 1 blending functions.***
- Each blending function $B_{k,d}$ ***is defined over d subintervals of the total range*** of u, starting at knot value u_k
- The range of parameter u is divided into ***n + d subintervals by the n + d + 1*** values specified in the knot vector.

- With knot values labelled as ***[u_0 , u_1 ,....., u_{n+d}], the resulting B-spline curve is*** defined only in the interval from knot value u_{d-1} , **up to knot value** u_{n+1}.
- Each section of the spline curve (between two successive knot values) is influenced by d control points.
- Any one control point can affect the shape of at most d curve sections.

Thank you