Unit-5: Surface and Curve Modelling

3D object representation is divided into two categories:

a) Boundary Representations (B-reps): The boundary representations describe a three dimensional object as a set of surface that separates the object interior from the environment.

E.g. Polyhedron, ellipsoid, aircraft, medical images etc.

- B-reps for single polyhedron satisfy Euler's formula: V-E+F=2

b) Space Partitioning Representation: Space partitioning describes interior properties by partitioning the spatial region containing an object into a small, non- overlapping, contiguous solids.

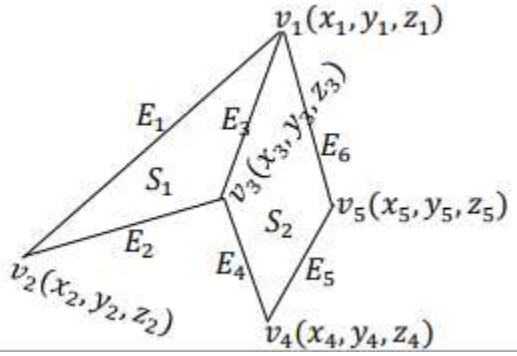
E.g. 3D object as Octree representation.

1. Polygon Surface

It is most common representation for 3D graphics object. In this representation, a 3D object is represented by a set of surfaces that enclose the object interior. This method simplifies and speeds up the surface rendering and display of the object.
The polygon surfaces are common in design and solid-modeling applications, since wire frame display can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.
Polygon surface can be represented by:

a) Polygon Table

In this method, a polygon surface is specified with a set of vertex co-ordinates and associated attributes. Polygon data tables can be organized into two groups: geometrical and attribute tables.
Geometric tables: It contain vertex coordinates and parameters to identify the spatial orientation of polygon surfaces
Attribute table: It gives attribute information for an object (Degree of transparency, surface reflectivity etc.).
Geometric data consists of three tables:
(i) Vertex table: It stores co-ordinate values for each vertex of the object.
(ii) Edge table: It stores the edge information of polygon.
(iii) Surface table: It stores the number of surfaces present in the polygon.



Vertex table

$v_1: x_1, y_1, z_1$
$v_2: x_2, y_2, z_2$
$v_3: x_3, y_3, z_3$
$v_4: x_4, y_4, z_4$
$v_5: x_5, y_5, z_5$

Edge table

$E_1: v_1, v_2$
$E_2: v_2, v_3$
$E_3: v_3, v_1$
$E_4: v_3, v_4$
$E_5: v_4, v_5$
$E_6: v_5, v_1$

Surface table

$S_1: E_1, E_2, E_3$
$S_2: E_3, E_4, E_5, E_6$

The object can be displayed efficiently by using data from tables and processing them for surface rendering and visible surface determination.

b) Polygon table using forward pointer in edge table

For above polygon,

Vertex table

$v_1: x_1, y_1, z_1$
$v_2: x_2, y_2, z_2$
$v_3: x_3, y_3, z_3$
$v_4: x_4, y_4, z_4$
$v_5: x_5, y_5, z_5$

Edge table

$E_1: v_1, v_2, S_1$
$E_2: v_2, v_3, S_1$
$E_3: v_3, v_1, S_1, S_2$
$E_4: v_3, v_4, S_2$
$E_5: v_4, v_5, S_2$
$E_6: v_5, v_1, S_2$

Surface table

$S_1: E_1, E_2, E_3$
$S_2: E_3, E_4, E_5, E_6$

c) Polygon Meshes

A polygon mesh is collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices and a polygon is a closed sequence of edges. An edge can be shared by two polygons and a vertex is shared by at least two edges.

This method can be used to represent a broad class of solids/surfaces in graphics. A polygon mesh can be rendered using hidden surface removal algorithms. The polygon mesh can be represented by three ways-

- Explicit representation
- Pointers to a vertex list
- Pointers to an edge list

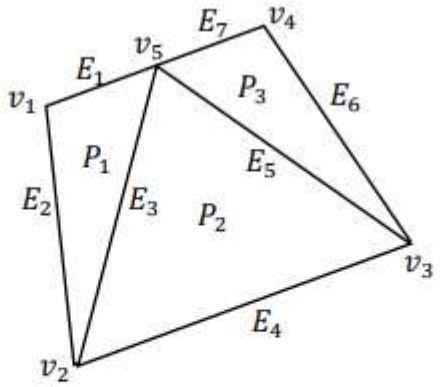
In **Explicit representation**, each polygon is represented by a list of vertex co-ordinates.

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

In **Pointers to a vertex list**, each vertex is stored just once, in vertex list

$$V = (v_1, v_2, \ldots\ldots\ldots, v_n)$$

E.g. A polygon made up of vertices 3, 5, 7, 10 in vertex list be represented as $P_1 = \{3,5,7,10\}$



Representing polygon mesh with each polygon as vertex list.

$P_1 = \{v_1, v_2, v_5\}$

$P_2 = \{v_2, v_3, v_5\}$ $P_3 = \{v_3, v_4, v_5\}$

In **Pointers to an edge list**, we have vertex list V, represent the polygon as a list of pointers to an edge list. Each edge in edge list points to the two vertices in the vertex list. Also to one or two polygon, the edge belongs. Hence, we describe polygon as $P = (E_1, E_2, \ldots\ldots\ldots, E_n)$ and an edge as $E = (v_1, v_2, P_1, P_2)$ Here if edge belongs to only one polygon, either then P_1 or P_2 is null.

For the mesh given above,

$V = \{v_1, v_2, v_3, v_4, v_5\} = \{((x_1, y_1, z_1), \ldots\ldots\ldots, (x_5, y_5, z_5)\}$

$E_1 = (v_1, v_5, P_1, N)$ $E_6 = (v_3, v_4, P_3, N)$

$E_2 = (v_1, v_2, P_1, N)$ $E_7 = (v_4, v_5, P_3, N)$

$E_3 = (v_2, v_5, P_1, P_2)$ $P_1 = (E_1, E_2, E_3)$

$E_4 = (v_2, v_3, P_2, N)$ $P_2 = (E_3, E_4, E_5)$

$E_5 = (v_3, v_5, P_1, P_3)$ $P_3 = (E_5, E_6, E_7)$

Here, N represents Null.

d) Plane equation

It this method polygon surface is represented by the equation of plane in the coordinate system. The 3D object is represented through the set of equations. The equation for plane surface can be expressed as

$$Ax + By + Cz + D = 0$$

Where x, y, z is any point on the plane and A, B, C & D are coefficient of plane equation and represents the spatial orientation of the polygon surface in space coordinate system. Hence, the value of coefficient must be known to represent the 3D object.

The value of A, B, C & D can be obtained by solving a set of three plane equation using coordinate of three non-collinear point on plane. Let us assume that three vertices of plane are (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3). Then,

$$Ax_1 + By_1 + Cz_1 + D = 0$$

$$Ax_2 + By_2 + Cz_2 + D = 0$$

$$Ax_3 + By_3 + Cz_3 + D = 0$$

By Cramer's rule

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

For any points (x, y, z)
If $Ax + By + Cz + D \neq 0$, then (x, y, z) is not on the plane.
If $Ax + By + Cz + D < 0$, then (x, y, z) is inside the plane i. e. invisible side
If $Ax + By + Cz + D > 0$, then (x, y, z) is lies outside the surface.

Normal and Spatial Orientation of Surface

In computer graphics, the normal and spatial orientation of a surface play crucial roles in rendering realistic and visually appealing 3D scenes. Let's delve into these concepts:

1. **Surface Normal:**
    - **Definition:** The surface normal is a vector that is perpendicular to the surface at a specific point. It indicates the direction in which the surface is facing.
    - **Vertex Normals:** In a 3D model, each vertex often has an associated normal vector. Vertex normals are used in smooth shading techniques to interpolate lighting across the surface of polygons.
    - **Face Normals:** In flat shading, each polygon (face) has a single normal vector representing its orientation. This is used in flat shading calculations.
    - **Importance in Lighting:** Surface normals are crucial for lighting calculations, including diffuse and specular reflections. They help determine how light interacts with the surface, affecting its appearance.
    - **Normalization:** To ensure accurate lighting calculations, normal vectors are typically normalized, meaning they are scaled to have a length of 1 while maintaining their direction.
2. **Spatial Orientation:**
    - **Definition:** Spatial orientation refers to how a surface or object is positioned and oriented in three-dimensional space.
    - **Euler Angles:** Commonly used to describe spatial orientation, Euler angles represent rotations around the object's intrinsic axes. These rotations are often expressed as pitch, yaw, and roll.
    - **Quaternion Representation:** Quaternions are another way to represent spatial orientation. They provide a compact and efficient representation for rotations in 3D space.

- **Matrix Representation:** Spatial orientation can also be represented using transformation matrices. These matrices encapsulate translation, rotation, and scaling transformations.
- **Importance in Rendering:** Proper spatial orientation is essential for accurate rendering. It affects how an object is viewed from different angles and influences its appearance during animation or interaction.
- **Transformations:** Spatial orientation involves transformations like translation, rotation, and scaling. These transformations are applied to vertices or objects to place them correctly within the 3D scene.

In summary, the normal and spatial orientation of surfaces are fundamental concepts in computer graphics, particularly in 3D rendering. They contribute to realistic lighting, shading, and overall scene appearance. Efficiently handling and manipulating these vectors and transformations are essential for creating immersive and visually compelling virtual environments.

Octree Representation

An Octree is a tree data structure used in computer science, particularly in 3D computer graphics and computational geometry.

It is a hierarchical subdivision of 3D space, often employed for spatial partitioning and efficient representation of volumetric data.

Octree Representation (Solid-object representation)

This is the space-partitioning method for 3D solid object representation. Octrees are hierarchical tree structures that describes each region space as nodes. They are used to represent solid objects in some graphics system.

- Medical imaging and other applications that require displays of object cross sections commonly use octree representation. E.g. CT-scan.

Octrees are used to partition a 3D space by recursively subdividing it into eight octants. Octant subdivisions continue until the region of space contains only homogeneous octants.

- Octrees are often used in 3D graphics and 3D game engines.

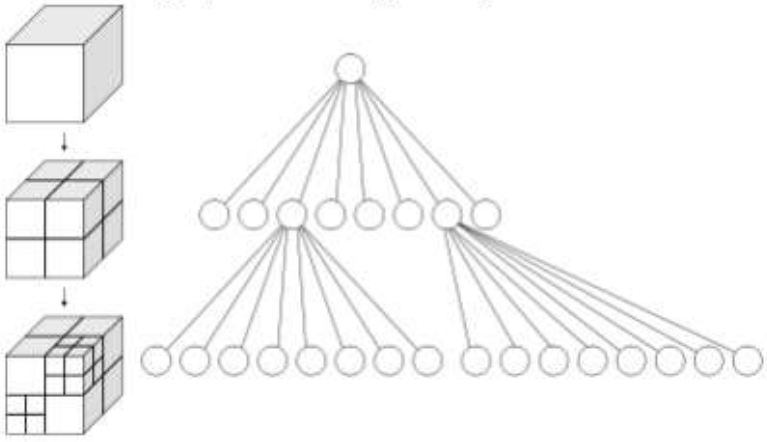


Fig: Left: Recursive subdivision of a cube into octants & Right: The corresponding octree

BSP tree

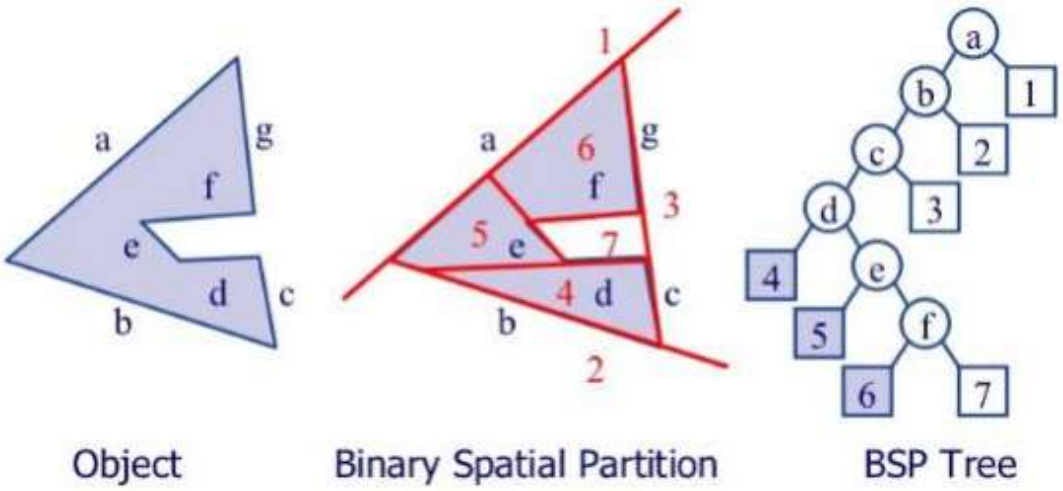
Binary Space Partitioning is implemented for recursively subdividing a space into two convex sets by using hyper-planes as partitions.

This process of subdividing gives rise to the representation of objects within the space in the form of tree data structure known as BSP Tree.

Binary space partitioning arose in the context of 3D computer graphics where the structure of a BSP tree allows for spatial information about the objects in a scene that is useful in rendering, such as objects being ordered from front-to-back with respect to a viewer at a given location, to be accessed rapidly.

Binary space partitioning is a generic process of recursively dividing a scene into two until the partitioning satisfies one or more requirements.

- At each step, the space is divided by a plane of arbitrary position and orientation.
- Each internal node of the tree is associated with a plane and has two child pointers (one for inside the polygon and the other to the outside).
- If the subspace is homogenous (fully indoors and outdoors), cease to be divided.



Bezier Curve and Properties

Bezier curve is developed by the French engineer Pierre Bezier for the design of Renault automobile bodies.

- It is an approximating spline widely used in various CAD system.
- Bezier curve is generated under the control of points known as control points.

General Bezier curve for (n+1) control point, denoted as $p_k = (x_k, y_k, z_k)$ with 'k' varying from 0 to n is given by

$$P(u) = \sum_{k=0}^{n} p_k BEZ_{k,n}(u), \qquad 0 \le u \le 1 \quad \ldots\ldots\ldots\ldots (i)$$

Where, $P(u)$ is a point on Bezier Curve.

p_k is a control point.

$BEZ_{k,n}(u)$ is a Bezier blending function also known as *Bernstein Polynomial*.

Bezier blending function is defined as

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

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

Individual x, y, z coordinates an a Bezier curve is given by,

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

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

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

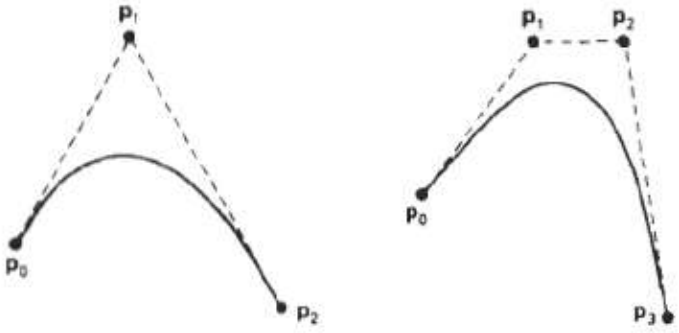


Fig: Bezier curve generated by three and four control point.

Properties of Bezier curve:

a) The basis functions are real.
b) The Bezier curve always passes through first and last control point i.e. $p(0) = p_0$ & $p(1) = p_n$.
c) The degree of polynomial representing Bezier curve is one less than the number of control points.
d) The Bezier curve always follows convex hull formed by control points.
e) The Bezier curve always lies inside the polygon formed by control points.
f) Bezier blending functions are positive and sum is equal to 1.
$\sum_{k=0}^{n} BEZ_{k,n}(u) = 1$
g) The direction of the tangent vector at the end points is same like vector determined by first and last segment.

 # Cubic Bezier Curve

- It is a Bezier curve generated by four control points.
- General equation for cubic Bezier curve is

$$P(u) = \sum_{k=0}^{3} p_k BEZ_{k,n}(u), \qquad 0 \le u \le 1 \quad \ldots\ldots\ldots\ldots (i)$$

$$P(u) = p_0 BEZ_{0,3}(u) + p_1 BEZ_{1,3}(u) + p_2 BEZ_{2,3}(u) + p_3 BEZ_{3,3}(u)$$

Where,

$$BEZ_{0,3}(u) = C(3,0)u^0(1-u)^{3-0} = \frac{3!}{0!\,(3-0)!} \times (1-u)^3 = (1-u)^3$$

Similarly,

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

$$\therefore \; P(u) = p_0(1-u)^3 + p_1 3u(1-u)^2 + p_2 3u^2(1-u) + p_3 u^3$$

In matrix form,

$$P(u) = [u^3 \quad u^2 \quad u \quad 1].M_{BEZ}.\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Where,

$$M_{BEZ} = Bezier\ matrix = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bezier surfaces

- Generalizations of Bezier curves to higher dimensions are called Bezier surfaces.
- The parametric vector function for the Bezier surface is formed as the Cartesian product of Bezier blending function:

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

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

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

Q. Construct Bezier curve for control points (4, 2), (8, 8) and (16, 4).

Solution:

Given control points

$p_0 = (x_0, y_0) = (4,2)$

$p_1 = (x_1, y_1) = (8,8)$

$p_2 = (x_2, y_2) = (16,4)$

Here, degree (or order) n= 2

We have basis function as

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

$$= \sum_{k=0}^{2} p_k BEZ_{k,2}(u), \qquad 0 \le u \le 1$$

$$\therefore\ P(u) = p_0 BEZ_{0,2}(u) + p_1 BEZ_{1,2}(u) + p_2 BEZ_{2,2}(u)$$

Parametric equations are;

$$x(u) = x_0 BEZ_{0,2}(u) + x_1 BEZ_{1,2}(u) + x_2 BEZ_{2,2}(u) \ \ldots\ldots\ldots\ldots \text{(i)}$$
$$y(u) = y_0 BEZ_{0,2}(u) + y_1 BEZ_{1,2}(u) + y_2 BEZ_{2,2}(u) \ \ldots\ldots\ldots\ldots \text{(ii)}$$

Now,

$$BEZ_{0,2}(u) = C(2,0)u^0(1-u)^{2-0} = \frac{2!}{0!(2-0)!} \times (1-u)^2 = (1-u)^2$$
$$BEZ_{1,2}(u) = C(2,1)u^1(1-u)^{2-1} = \frac{2!}{1!(2-1)!} \times u(1-u)^1 = 2u(1-u)$$
$$BEZ_{2,2}(u) = C(2,2)u^2(1-u)^{2-2} = \frac{2!}{2!(2-2)!} \times u^2(1-u)^0 = u^2$$

Putting these values in eq. (i) & (ii) we get;

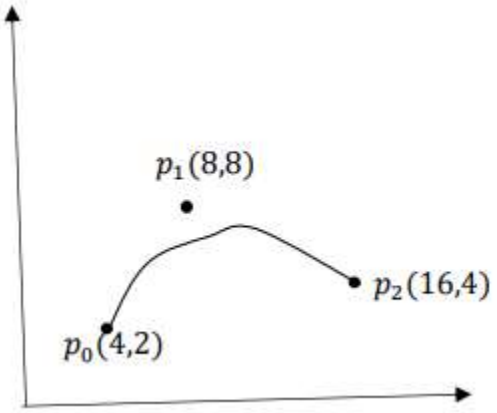
$$x(u) = x_0(1-u)^2 + x_1 2u(1-u) + x_2u^2 = 4u^2 + 8u + 4$$

$$y(u) = y_0(1-u)^2 + y_1 2u(1-u) + y_2u^2 = -10u^2 + 12u + 2$$

Now,

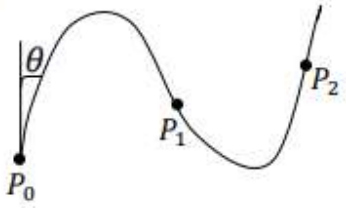
	$x(u)$	$y(u)$
u =0	4	2
u =0.2	5.76	4.0
u =0.4	7.84	5.20
u =0.6	10.24	5.6
u =0.8	12.96	5.2
u =1	16	4

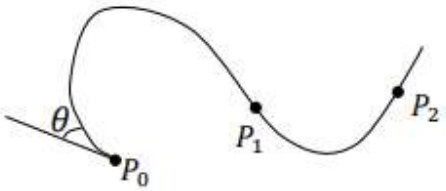
Drawing these points we get:



Hermite Interpolation (Hermite curve)

It is an interpolating piecewise cubic polynomial with a specified tangent at each control point.





If we change the control point at P_0, then the curve will also change, so that angle θ between P_0 and tangent at P_0 will remain constant.

- It has local control over the curve i.e. each curve section depend on its end point only.
- The vector equivalence of Hermite curve is
  $P(u) = au^3 + bu^2 + cu + d$ ..... (i)

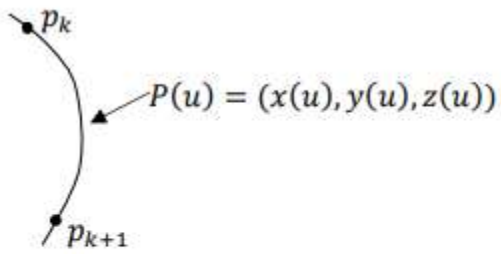
Where, x component of $P(u)$ is

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

Similarly, y and z component

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

Let $P(u)$ denotes the parametric cubic point function for the curve section between control point p_k & p_{k+1}.



At p_k, u=0

$\therefore P(0) = p_k$

At p_{k+1}, u=1

$\therefore P(1) = p_{k+1}$

Also let Dp_k & Dp_{k+1} denote the slope at p_k & p_{k+1}.

$\therefore\ P'(0) = Dp_k$

$P'(1) = Dp_{k+1}$

Hence boundary condition for Hermite curve

$P(0) = p_k$

$P(1) = p_{k+1}$ ..... (ii)

$P'(0) = Dp_k$

$P'(1) = Dp_{k+1}$

Matrix equivalent of eq. (i) is

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

Similarly derivative of point function can be represented as,

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

In matrix form, the Hermite boundary condition from eq. (ii) can be represented as

$$\begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \text{..... (iv)}$$

Solving eq. (iv) for polynomial coefficient, we have

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

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

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

Where, M_H is the Hermite matrix.

Hence eq. (iii) can be represented as

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] . M_H . \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots \text{(v)}$$

Expanding (v)

$$P(u) = p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2)$$

In terms of Hermite blending function, 'H', the Hermite curve can be represented as:

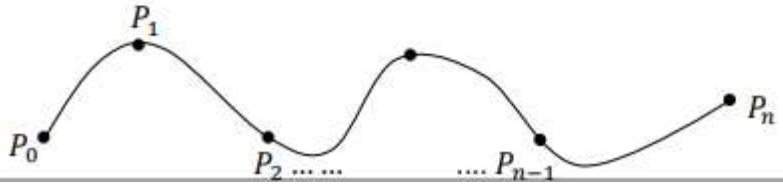
$$\boxed{P(u) = p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + Dp_{k+1} H_3(u)}$$

Parametric Cubic Spline

- It is used to set up path for object motions or to provide a representation for an existing object or drawings.
- Compared to higher-order polynomials, cubic splines requires less calculation and memory and they are more stable. Compared to lower-order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.
- Cubic interpolation spline is obtained by fitting the input points with a piecewise cubic polynomial curve that passes through every control points.

Suppose we have n+1 control points having co-ordinates

$$P_k = (x_k, y_k, z_k) \qquad K = 0, 1, 2, 3, \ldots\ldots\ldots, n$$



A parametric cubic polynomial that is to be fitted between each pair of control points have following equations:

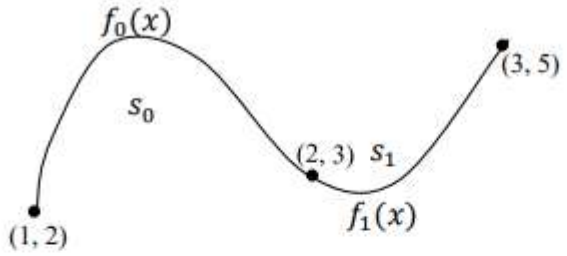
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \qquad (0 \le u \le 1)$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

We need to determine the values of the four coefficients a, b, c, and d in the polynomial representation for each of the n curve section. We do this by setting enough boundary conditions at the "joints" between curve sections we can obtain numerical values for all the coefficients.

- Cubic splines are more flexible for modeling arbitrary curve shapes.

Q. Construct a natural cubic spline that passes through (1, 2), (2, 3) & (3, 5) having two segments $f_0(x)$ & $f_1(x)$.

Solution:



For segment s_0

$$f_0(x) = a_0 + b_0 x + c_0 x^2 + d_0 x^3$$

a) Since it passes through (1, 2)
$$a_0 + b_0 + c_0 + d_0 = 2 \ldots\ldots\ldots \text{(i)}$$
b) Since it passes through (2, 3)
$$a_0 + 2b_0 + 4c_0 + 8d_0 = 3 \ldots\ldots\ldots\ldots \text{(ii)}$$
c) Slope of $f_0(x)$ & $f_1(x)$ must be same at (2, 3)
$$\frac{d\,f_0(x)}{dx} = \frac{df_1(x)}{dx}$$
or, $b_0 + 2c_0 x + 3d_0 x^2 = b_1 + 2c_1 x + 3d_1 x^2$
or, $b_0 + 4c_0 + 12d_0 - b_1 - 4c_1 - 12d_1 = 0 \ldots\ldots\ldots\ldots\ldots \text{(iii)}$
d) Curvature must be same for $f_0(x)$ & $f_1(x)$ at (2, 3)
$$\frac{d\,f_0'(x)}{dx} = \frac{df_1'(x)}{dx}$$
or, $2c_0 + 6d_0 x = 2c_1 + 6d_1 x$
or, $2c_0 - 2c_1 + 6d_0 x - 6d_1 x = 0$
or, $2c_0 - 2c_1 + 12d_0 - 12d_1 = 0 \ldots\ldots\ldots\ldots \text{(iv)}$
e) $f''(x) = 0$ at (1, 2)
or, $2c_0 + 6d_0 x = 0$
or, $2c_0 + 6d_0 = 0$
or, $c_0 + 3d_0 = 0 \ldots\ldots\ldots\ldots \text{(v)}$

For segment s_1

$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$$

a) Since it passes through (2, 3)

$$a_1 + 2b_1 + 4c_1 + 8d_1 = 3 \text{ ..... (vi)}$$

b) Since it passes through (3, 5)

$$a_1 + 3b_1 + 9c_1 + 27d_1 = 5 \text{ ..... (vii)}$$

c) $f''(x) = 0$ at (3, 5)

or, $2c_1 + 6d_1x = 0$

or, $2c_1 + 18d_1 = 0$

or, , $c_1 + 9d_1 = 0$ ..... (viii)

Solve these equation and plot the graph.

B-Spline Curve

B-spline curve is a set of piecewise polynomial segments that passes close to a set of control points.

It has two advantage over Bezier curve:

a) The degree of B-spline polynomial can be set independently of the number of control points.
b) It allows local control over the shape of a spline curve.

General equation of B-spline curve is given by

$$P(u) = \sum_{k=0}^{n} p_k B_{k,d}(u), \qquad 0 \le u \le n+d, \;\; 2 \le d \le n+1$$

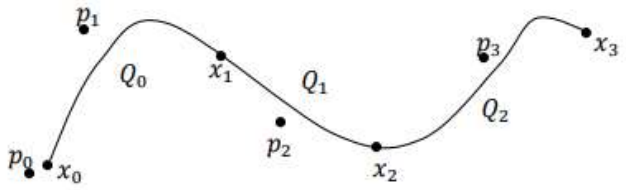
Where, p_k is a set of (n+1) control points.

$B_{k,d}(u)$ is the B-spline blending function.

Blending function for B-spline curves are defined by

$$B_{k,1}(u) = \begin{cases} 1, & if\ u_k \le u < u_{k+1} \\ 0, & Otherwise \end{cases}$$

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



p_0, p_1, p_2, p_3 → Control point

x_0, x_1, x_2, x_3 → Knot values

Q_0, Q_1, Q_2 → Curve segment

- The knots produce a vector that defines the domain of the curve.