Prof. Supriya S

MODULE - 5

Curve Representation

- Non-parametric form
  - $y = f(x)$
  - Implicit form
    - $f(x, y) = 0$ → Bresenham's clipping
- Explicit form
  - $y = mx + b$
  - clipping
- Parametric form
  - $x = x(t)$
  - $y = y(t)$

Non-planar objects - spheres, ellipse, cones (primitive of OpenGL).

Some applications observe drawbacks with ~~planar~~ Polygon patches

1. Sharp bends in 2D/3D
2. Large changes of curvatures of surface
3. Gradient changes very fast

Intersection of 2 surfaces - yields a curve

Prof. Supriya S

Programming Event-Driven Input

① Using the pointing Device

2 types of events are associated with pointing devices.

* A mouse Event is generated when a mouse is moved with one of the buttons pressed (or released)
* If a mouse is moved without a button being held down, this event is called Passive move event.
* The information returned includes the button that generated the event, the state of button after the event (up or down) & the position of cursor tracking the mouse in window co-ordinates.
* Register the mouse callback function in main()

```
glutMouseFunc(myMouse);
```

* mouse callback must have the form

```
void myMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state ==
    GLUT_DOWN)
        exit(0);
}
```

* Reshape event is generated whenever the window is resized (by user interaction)

② Window Event.

* allows the user to resize the window interactively, usually by using the mouse to drag a corner of the window to a new location → Window Event.
* If the window size changes, 3 questions are to be considered
  1. Do we redraw all the objects that were in the window before it was resized?
  2. What if the aspect ratio of new window is different from the old window.
  3. Do the size or attributes of new primitives changes if the size of new window is different from that of old.

```
void MyReshape(int w, int h)
{

}
glutReshapeFunc(myReshape) ⟶ callback function.
```

* window movement without Resizing.

```
glutMotionFunc(drawsquare);
```

③ Keyboard Events.

Keyboard Events are generated when the mouse is in the window and one of the keys is pressed or released.

glutKeyBoardFunc → callback for events generated by pressing the key.

Prof. Supriya S

glutKeyboardUpFunc → event generated by releasing a key.

* glutKeyboardFunc (myKey);
* Void myKey (unsigned char key, int x, int y)
{ if (key == 'q' || key == 'Q') exit();
}

(4) The Display & Idle Callbacks

* glutDisplayFunc (myDisplay);
* glutPostRedisplay(); → avoids extra or unnecessary screen drawings by setting a flag inside GLUT's main loop indicating that the display needs to be redrawn.
* Idle callback → continue to generate graphical primitives through a display function while nothing else is happening. It is invoked when there are no other events.

(5) Window Management.

id = glutCreateWindow ("Window");
glutSetWindow (id);

Prof. Supriya S

Menus

GLUT provides one additional feature, pop-up menus, that would be used with the mouse to create sophisticated interactive applications.

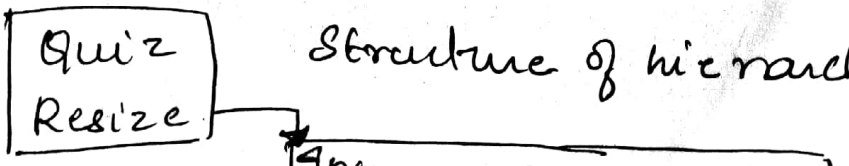
Menus involves

1) define the actions corresponding to each entry in the menu.
2) Link the menu to a particular mouse button
3) finally register a callback function for each menu.

Ex:

```
glutCreateMenu(demo_menu);          Register callback
glutAddMenuEntry("quit", 1);        function for each menu
glutAddMenuEntry("Increase square size", 2);
glutAddMenuEntry("decrease square size", 3);
glutAttachMenu(GLUT_RIGHT_BUTTON);  → Link menu to mouse button.

void demo_menu(int id)
{   switch(id)
    {  case 1: exit(0);
       break;                       action corresponding
       case 2: size = 2*size;       to each entry in
       break;                       the menu
       case 3: if(size>1) size = size/2;
       break;
    }glutPostRedisplay();
}
```



Structure of hierarchical menu

The Structure of hierarchical menu is shown in the previous page. The code is as follows

```
sub_menu = glutCreateMenu(Size_menu);
glutAddMenuEntry("Increase square size", 2);
glutAddMenuEntry("Decrease Square Size", 3);
glutCreateMenu(top_menu);
glutAddMenuEntry("Quit", 1);
glutAddSubMenu("Resize", sub_menu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Picking

Picking is the logical operation that allows the user to identify an object on the display. Action of picking uses a pointing device.

3 ways

1) Selection : involves adjusting the clipping region and viewport, tracks which primitives in a small clipping region are rendered into the region near the cursor. These primitives are placed in the hit list to be examined later.

2) bounding boxes or extents (objects of interest) extent of an object is a smallest rectangle, aligned with co-ordinate axes, that contains the object.

3) using double buffer scan using extra rendering. Double buffering – 2 color buffers. ① A front buffer & back buffer

function glRenderMode( ) can be one of three modes.

① normal rendering to the color buffer (GL_RENDER)

② Selection mode (GL_SELECT)

③ feedback mode (GL_FEEDBACK) - used to obtain a list of primitives that were rendered

when selection mode or render mode is selected, each primitive within the clipping window/volume generates a message called a hit that is stored in a buffer called name stack.

glSelectBuffer – used to identify an array for selected data.

i.e 4 important functions for initialising the name stack, for pushing & poping info on it, for manipulating the top entry of stack.

void glSelectBuffer (GLsizei n , GLuint * buffer.

void glInitNames () → initialises name stack

void glPushName(GLuint name) – pushes name on the name stack.

void glPopName () – pops the top name from the name stack.

Prof. Supriya S

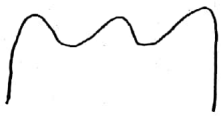
Void glLoadName(GLuint name) — replaces top of name stack with name.

** gluPickMatrix(x, y, w, h, *vp);

Prof. Supriya S

Curved Surfaces.

* Surface modeling free form curves and surfaces.


curves


Surfaces

* In Computer graphics, one would be interested to visually represent an object with mathematical representation (Geometric modeling)

Parametric Representation.

$x = r\cos(t)$
$y = r\sin(t)$
$z = h$

Planar eqn.

$x = a + bu + cw$
$y = d + eu + fw$
$z = g + hu + iw$

where every co-ordinate point on a plane (x, y, z) can be represented as a function of 2 variable (u, w) used for surfaces.

* Parametric equations completely seperate the roles dependent & independent variables
* case 1 : $r = 5, t = \pi, z = 20$ represents a "point"
* case 2 : $r = 5, -\pi \leq t \leq \pi, z = 20$ represents a "circle".
* case 3 : $r = 5, -\pi \leq t \leq \pi, 0 \leq z \leq 20$. represents cylindrical surface.
* case 4 : vary t & r - circular disk - planar entity
* case 5 : vary r & z - rectangle.

case 6 : $0 \le r \le 5$, $-\pi \le t \le \pi$, $0 \le z \le 20$

Solid cylinder.

* Offers more degree of freedom for controlling the shape of curves & surfaces.

Explicit form

$$y = px^3 + qx^2 + rx + s$$

Parametric form

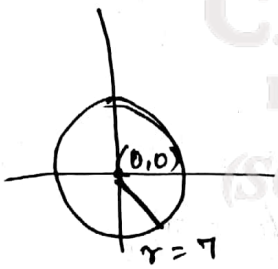
$$x = au^3 + bu^2 + cu + d \quad \text{(cubic eqn)}$$
$$y = eu^2 + fu^2 + gu + h$$

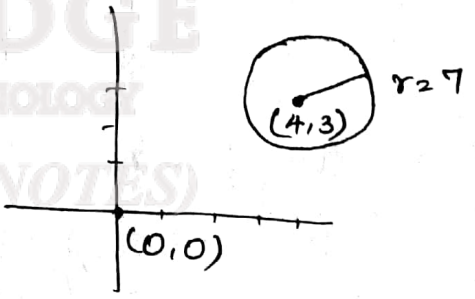
* Relative positions with the other [Transformations] are easy to apply

En: Circle with center (0,0) & radius - 7 $\xrightarrow{\text{transformed}}$ circle with center (4, 3) & radius - 7

$x = 7\cos(t)$
$y = 7\sin(t)$

$x = 4 + 7\cos(t)$
$y = 3 + 7\sin(t)$





* From the programmer view (Geometric entities) is represented are
A point in vector form, transformations in matrices.

Vectors & matrices

↕ ↑

Geometric entity — transformation representations

Prof. Supriya S

* Has an advantage in representing the bounds of Geometric entities.

circle:

$x = r\cos(t)$
$y = r\sin(t)$
$r = 8,\ -\pi \le t \le \pi$

circular arc

$x = r\cos(t)$
$y = r\sin(t)$
$r = 8,\ -\pi \le t \le 0$

Quadric Surfaces - frequently used class of objects described with second-degree equations. Includes spheres, ellipsoids, tori, paraboloids, hyperboloids.

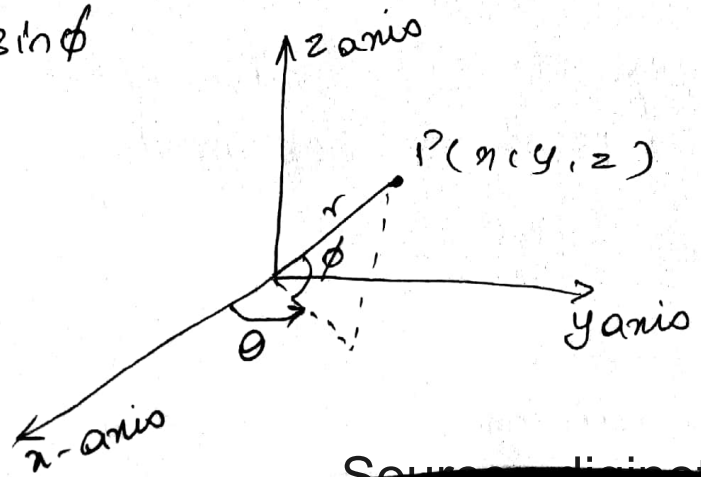
Sphere

In Implicit/Cartesian co-ordinates, a spherical surface with radius r centered at co-ordinate origin is defined as set of point (x, y, z) that satisfy the eqn.

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

Parametric form.

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

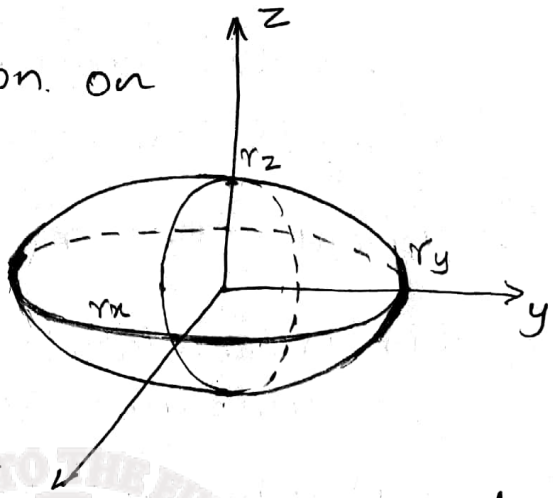


Prof. Supriya S

Ellipsoid

Cartesian representation on the origin

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



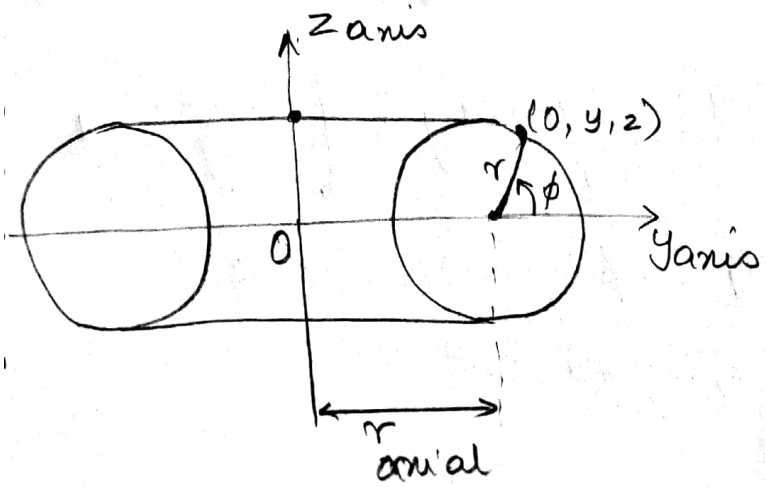
Parametric representation for the ellipsoid in terms of latitude angle ϕ & longitude angle θ

$$x = r_x \cos\phi \cos\theta \qquad -\pi/2 \le \phi \le \pi/2$$
$$y = r_y \cos\phi \sin\theta \qquad -\pi \le \theta \le \pi$$
$$z = r_z \sin\phi$$

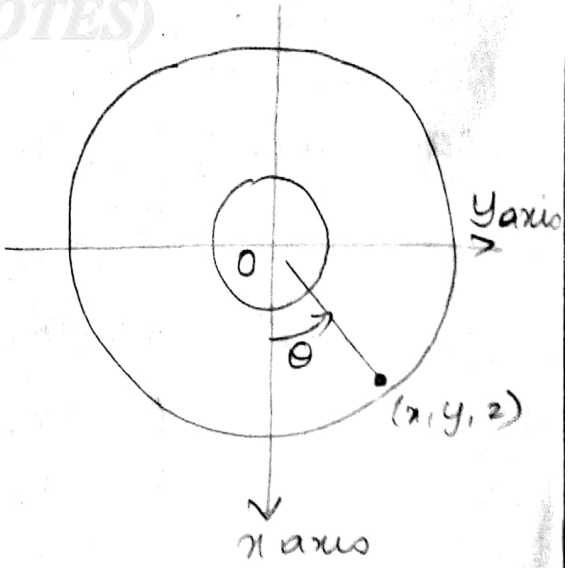
Torus

A doughnut shaped object is called a torus or anchor ring.

Parameters for a torus – the distance of conic center from the rotation axis & the dimensions of the conic.



Side View



Prof. Supriya S

The equation for the cross-sectional circle in side view figure is

$$(y - r_{axial})^2 + z^2 = r^2$$

Rotating this circle about z-axis produces the torus with cartesian equation.

$$(\sqrt{x^2+y^2} - r_{axial})^2 + z^2 = r^2$$

Parametric eqn for torus with circular cross section are

$$x = (r_{axial} + r\cos\phi)\cos\theta \qquad -\pi \le \phi \le \pi$$
$$y = (r_{axial} + r\cos\phi)\sin\theta \qquad -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$

we can generate a torus by rotating an ellipse instead of a circle. about z-axis

For an ellipse in yz plane with semimajor & semiminor axis denoted by r_y & r_z.

$$\left(\frac{y - r_{axial}}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

cartesian eqn $\Rightarrow$ $$\left(\frac{\sqrt{x^2+y^2} - r_{axial}}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1.$$

Parametric eqn

$$x = (r_{axial} + r_y\cos\phi)\cos\theta \qquad -\pi \le \phi \le \pi$$
$$y = (r_{axial} + r_y\cos\phi)\sin\theta \qquad -\pi \le \theta \le \pi$$
$$z = r_z\sin\phi$$

Curves can be classified in number of ways

* Plane curves & space curves

Ex: Circle & Helix (curve is a line (straightness), is a circle (having equal radius))

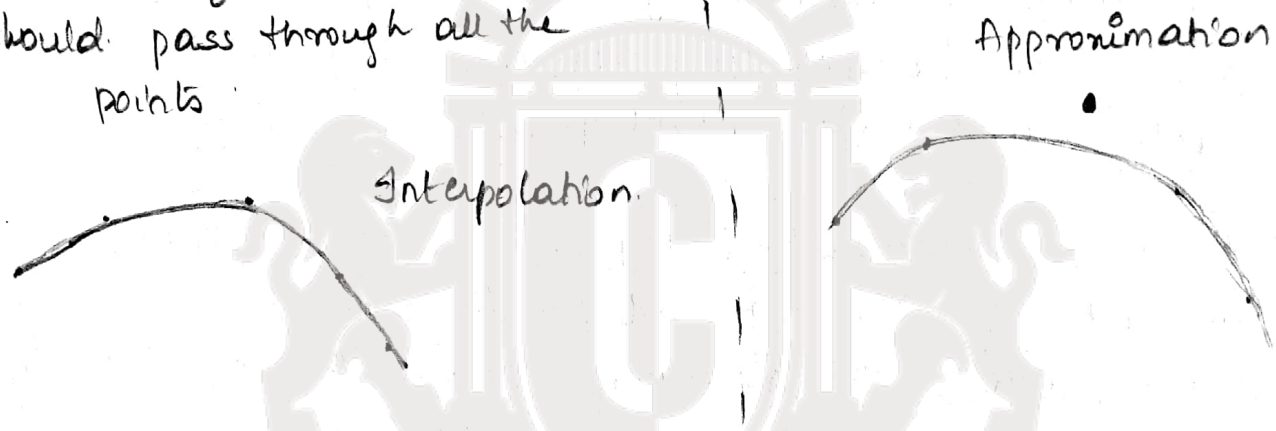
* Curves of known forms & free form curves

Ex: Circle Vr Bezier curve.

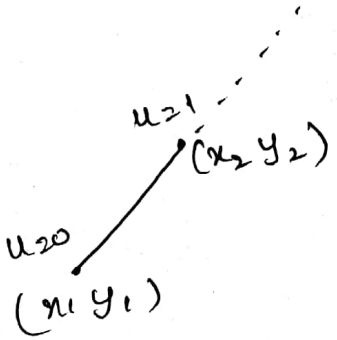
* In order to define a curve, certain conditions are to be satisfied.
  - Interpolation curves & Approximation curves (If conditions may not be satisfied)

Ex: Hermite Curve Vr Bezier curve.

Ex: Set of points, curves should pass through all the points



A Straight line is a subset of a curve.



Line in xy plane

$$x = x_1 + (x_2 - x_1)u$$
$$y = y_1 + (y_2 - y_1)u$$
$$0 \le u \le 1.0$$

} Parametric representation

To find any point (x, y) at any time interval

$$x = x_1 + t_1 \Delta x \qquad \text{here } \Delta x = x_2 - x_1$$
$$y = y_1 + t_1 \Delta y \qquad \Delta y = y_2 - y_1$$

Hence
$$x = x_1 + (x_2 - x_1)u$$
$$y = y_1 + (y_2 - y_1)u$$

Prof. Supriya S

Spline Representations

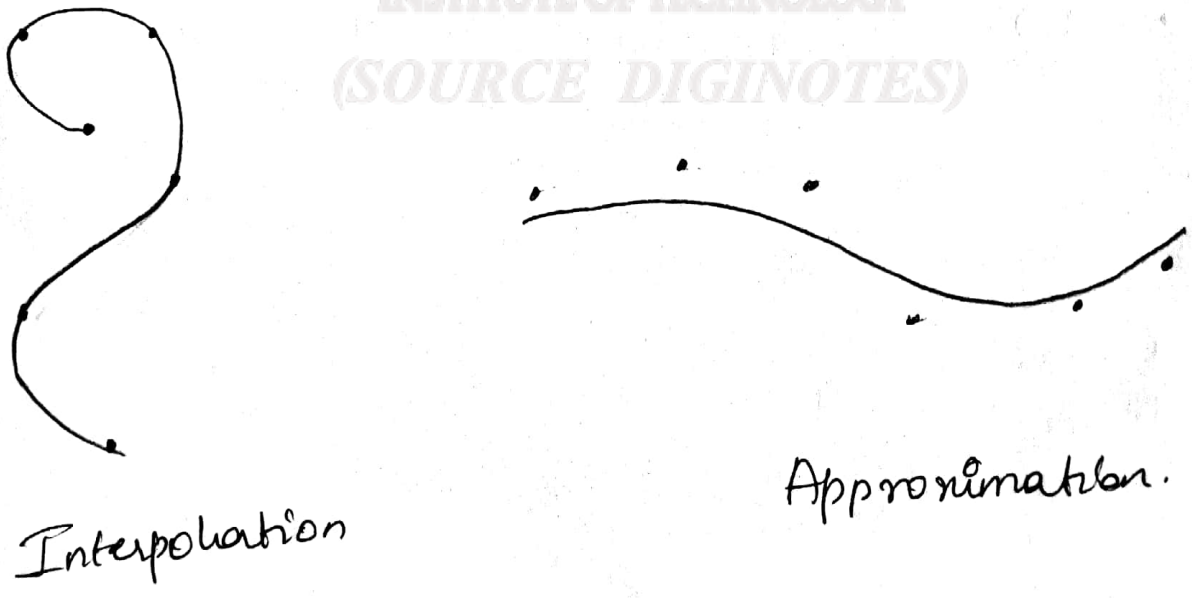
Spline is a flexible strip used to produce a smooth curve through a designated set of points

* Interpolation and Approximation splines

A spline curve can be specified by giving a set of co-ordinate points, called control points, which indicate shape of the curve.

A curve generated by connecting all the control points → the resulting curve is said to interpolate the set of control points.

A curve generated by connecting / plotting some or all, of the control points are not on the curve path, the resulting curve is said to approximate set of control points

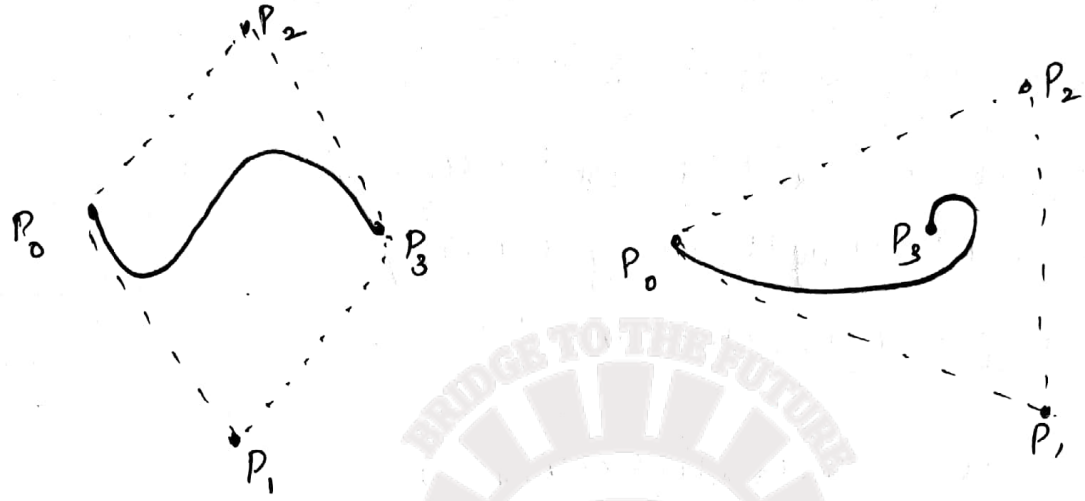


Interpolation

Approximation.

Prof. Supriya S

A set of control points forms a boundary for a region of space that is called a convex hull



Spline Specifications

It has 3 different forms.

* Algebraic form (12 algebraic co-efficients)
* Geometeric form (end points & tangent vectors)
* 4 point form (four points)

(1) Algebraic form – Parametric cubic polynomial representation

$$x = a_{3x}u^3 + a_{2x}u^2 + a_{1x}u + a_{0x}$$
$$y = a_{3y}u^3 + a_{2y}u^2 + a_{1y}u + a_{0y}$$
$$z = a_{3z}u^3 + a_{2z}u^2 + a_{1z}u + a_{0z}$$

} 12 unknowns

where $0 \le u \le 1$

(2) $P(u)$ is a point vector for a point (x, y, z)

Vector form.

$$P(u) = a_3u^3 + a_2u^2 + a_1u + a_0$$

Prof. Supriya S

Boundary for this curve can be set for the endpoint co-ordinate positions (x_0, y_0) (x_1, y_1)

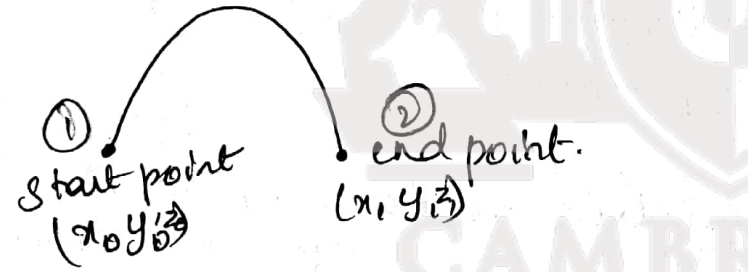
i.e Algebraic to Geometric form

$$x = a_{3x}u^3 + a_{2x}u^2 + a_{1x}u + a_{0x}$$
$$y = a_{3y}u^3 + a_{2y}u^2 + a_{1y}u + a_{0y}$$
$$z = a_{3z}u^3 + a_{2z}u^2 + a_{1z}u + a_{0z}.$$

$$x' = 3a_{3x}u^2 + 2a_{2x}u + a_{1x}$$
$$y' = 3a_{3y}u^2 + 2a_{2y}u + a_{1y}$$
$$z' = 3a_{3z}u^2 + 2a_{2z}u + a_{1z}$$

} tangent vectors.

start & end points with its tangent vectors.



total 4 points.

3 starting tangent vector.
$(x_0' \; y_0' \; z_0')$

4 ending tangent vector
(x_1', y_1', z_1')

These four boundary conditions are sufficient to determine the values of 4 coefficients a_x, b_x, c_x & d_x

(2)

To represent in matrix format : $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$

$$x(u) = [u^3 \; u^2 \; u \; 1] \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

$= U \cdot C$

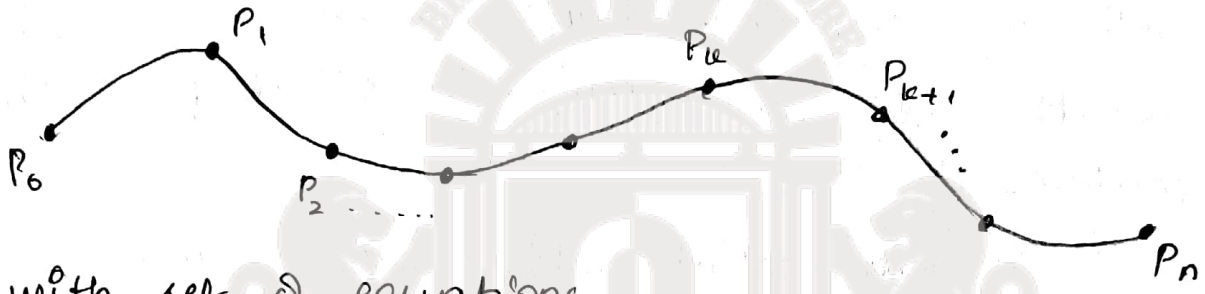
where c is co-efficient matrix for boundary condit^n.

Cubic - Spline Interpolation Methods.

Cubic - interpolation splines are obtained by connecting passing through every control point as shown in fig

Suppose n+1 control points are specified.

$$P_k = (x_k, y_k, z_k) \quad k = 0, 1, \ldots, n.$$



with set of equations.

$$\left.\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z \end{aligned}\right\} (0 \le u \le 1)$$

* Natural Cubic Splines

This interpolation curve is a mathematical representation of original drafting spline.

A natural spline is formulated by requiring the two adjacent curve sections to have same first and second parametric derivatives at their common boundary. Thus natural cubic splines have C^2 continuity. 0 ——— n

(0 to n = n+1) → n curve sections

If n+1 control points are considered, then n curve sections with a total of 4n polynomial (a, b, c, d) co-efficients are to be determined.

Prof. Supriya S

* The two curve sections on either side of a control point must have same first and second parametric derivations at that control point and each curve must pass through the control point. Thus gives 4n-4 eq^n to be satisfied by 4n polynomial co-efficients.

P_0 → first control point (beginning of the curve.

P_n → last control point

In order to determine values for all the co-efficients

(1) Method: set second derivatives at P_0 & P_n equal to 0.

(2) add 2 extra control points (called dummy points), one at the beginning of the curve labelled as P_{-1} and the other labelled P_{n+1} at the end.

Thus all the original control points are interior & has 4n boundary conditions.

Hermite Interpolation

Hermite spline named by French Mathematician Charles Hermite) is an interpolating piecewise cubic polynomial with a specified tangent at each control point.

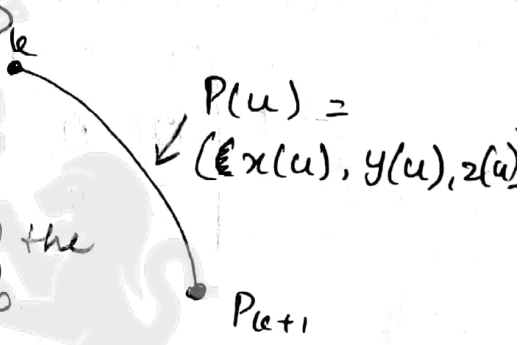
If P(u) represents parametric cubic point function for curve section b/w point P_k & P_{k+1}, then the boundary conditions of hermite curve section are.

$P(0) = P_k$

$P(1) = P_{k+1}$

$P'(0) = DP_k$ } Slope of the curves

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



$\therefore P(u) = au^3 + bu^2 + cu + d \quad 0 \le u \le 1$

equivalent matrix is for $x(u) = a_x u^3 + b_x u^2 + c_x u + d$

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

Similarly y & z component

& corresponding tangent vectors.

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

Substitute $u = 0$ & $u = 1$

$$\begin{bmatrix} P(0) \to P_k \\ P(1) \to P_{k+1} \\ P'(0) \to P'_k \\ P'(1) \to P'_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Prof. Supriya S

solving for polynomial co-efficients -

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ P'_k \\ P'_{k+1} \end{bmatrix}$$

$$\frac{Adj\ A}{|d|}$$

Explanation : Since the first row has 3 column values 0, will only consider the 4^{th} value - 1

$$-1 \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 3 & 2 & 1 \end{vmatrix} = -1 \left[+1 \begin{vmatrix} 1 & 1 \\ 3 & 2 \end{vmatrix} \right]$$

$$= -1 \left(+ (-2+3) \right)$$

$$= -1 \left(+ (+1) \right)$$

$$-1 \rightarrow \text{determinant}$$

$$\begin{vmatrix}
+\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 2 & 1 & 0 \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 3 & 1 & 0 \end{vmatrix} & +\begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 3 & 2 & 0 \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 3 & 2 & 1 \end{vmatrix} \\
-\begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 2 & 1 & 0 \end{vmatrix} & +\begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 3 & 1 & 0 \end{vmatrix} & -\begin{vmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 3 & 2 & 0 \end{vmatrix} & +\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 3 & 2 & 1 \end{vmatrix} \\
+\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{vmatrix} & -\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 3 & 1 & 0 \end{vmatrix} & +\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 3 & 2 & 0 \end{vmatrix} & -\begin{vmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 3 & 2 & 1 \end{vmatrix} \\
-\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix} & +\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix} & -\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{vmatrix} & +\begin{vmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}
\end{vmatrix}^T$$

Prof. Supriya S

$$= \begin{bmatrix} +(0-2) & -(-(0+3)) & +(0+0) & -(-2+3) \\ -(0-2) & +(-(0+3)) & -(0-0) & +(0) \\ +(-(-1+2)) & -(1-3) & +(-(-2+3)) & -(0) \\ -(-(1+0)) & +(-(-1+0)) & -(0) & +(0) \end{bmatrix}^T$$

$$= \begin{bmatrix} -2 & +3 & 0 & -1 \\ 2 & -3 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 1 & +1 & 0 & 0 \end{bmatrix}^T$$

divide by determinant -1

$$= \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ -1 & -1 & 0 & 0 \end{bmatrix}^T$$

$$= \begin{bmatrix} 2 & -2 & 1 & -1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

i.e $$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & -1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ P'_k \\ P'_{k+1} \end{bmatrix}$$

$$= M_H \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ P'_k \\ P'_{k+1} \end{bmatrix}$$

Prof. Supriya S

$$\therefore P(u) = [u^3\ u^2\ u\ 1] \cdot M_H \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ P'_k \\ P'_{k+1} \end{bmatrix}$$

when solving would produce.

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

(P_k, P'_{k+1})

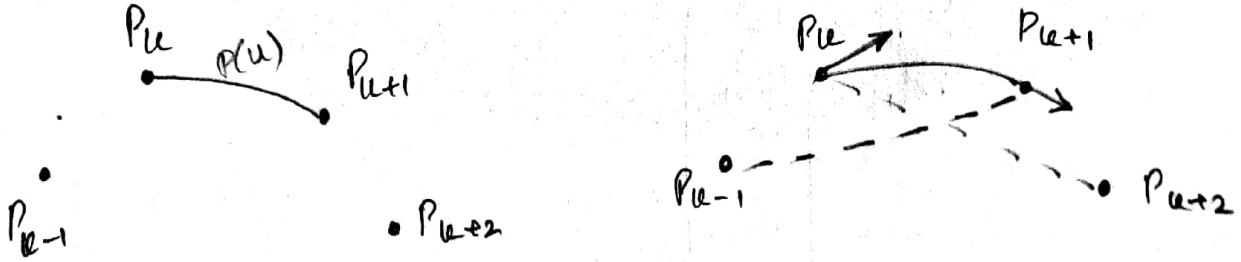
$$= P_k H_0(u) + P_{k+1} H_1 + DP_k H_2 + DP_{k+1} H_3$$

Cardinal Splines

The difference is that we do not input the values for the endpoint tangents.

For a cardinal ~~points~~ spline, the slope at a control point is calculated from the co-ordinates of two adjacent control points.

The figure shows a cardinal spline section with 4 consecutive control point positions. The middle 2 control points are the section end points, & other 2 points are used in the calculation of endpoint slopes.



The 4 control points P_{k-1} to P_{k+2} are used to set the boundary conditions for cardinal spline section

$$P(0) = P_k$$
$$P(1) = P_{k+1}$$
$$P'(0) = \frac{1}{2}(1-t)(P_{k+1} - P_{k-1})$$
$$P'(1) = \frac{1}{2}(1-t)(P_{k+2} - P_k)$$

Parameter t is called tension parameter (controls how loosely or tightly the spline fits the points).

When $t=0$, the class of curves is refered to as Catmull-Rom splines or Overhauser splines.

$$P(u) = [u^3 \; u^2 \; u \; 1] \cdot M_c \cdot \begin{bmatrix} P_{k-1} \\ P_k \\ P_{k+1} \\ P_{k+2} \end{bmatrix}$$

(looser curve) $t < 0$

(tighter curve) $t > 0$

Where cardinal matrix is

$$M_c = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

with $s = (1-t)/2$

Prof. Supriya S

Kochanek – Bartels splines.

Extension of cardinal splines.

* 2 additional parameters are introduced into constraint equations defining Kochanek – Bartels splines – to further provide flexibility in adjusting the shapes of curve sections.

$$P(0) = P_k$$

$$P(1) = P_{k+1}$$

$$P'(0)_{in} = \frac{1}{2}(1-t)[(1+b)(1-c)(P_k - P_{k-1}) + (1-b)(1+c)(P_{k+1} - P_k)]$$

$$P'(1)_{out} = \frac{1}{2}(1-t)[(1+b)(1+c)(P_{k+1} - P_k) + (1-b)(1-c)(P_{k+2} - P_{k+1})]$$

t → tension parameter
b → bias parameter
c → continuity parameter.

* these splines were designed ~~for~~ to model Animation paths. Eg: motion changes used in cartoon Animation

Prof. Supriya S

Bezier Spline curves

* Bezier curves are approximation curve.
* Proposed in 60's by P. Bezier (Pierre)
* used to define sculpured surfaces of automobile bodies (Renalt)
* Bezier curves can be fitted to any number of control points (some packages limit to 4 points)
* The degree of Bezier polynomial is determined by no. of control points to be approximated & their relative position.

* consider n+1 control points denoted as $P_k = (x_k, y_k, z_k)$ with k varying from 0 to n.
* These co-ordinate points are blended to produce position vector P(u) describing the paths between P_0 to P_n

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

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

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

where parameter C(n,k) are binomial co-efficients $C(n,k) = \dfrac{n!}{k!\,(n-k)!}$

To explain the above in detail: 4 control point Bezier

Def^n: $x(u) = (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u) u^2 x_2 + u^3 x_3$

Prof. Supriya S

This can be written as

$$x(u) = {}^3C_0 (1-u)^3 x_0 + {}^3C_1 (1-u)^2 u x_1 + {}^3C_2 (1-u) u^2 x_2 + {}^3C_3 u^3 x_3$$

$${}^3C_0 = 1, \quad {}^3C_1 = 3, \quad {}^3C_2 = 3, \quad {}^3C_3 = 1$$

$$\therefore x(u) = \sum_{i=0}^{3} \overbrace{{}^3C_i}^{\rightarrow n} (1-u)^{3-i} u^i x_i$$

$\underbrace{\qquad\qquad}_{BEZ_{k,n}}$

which can be represented w.r.t control points 0 to n. $-(n+1)$ control points.

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

$$\boxed{\therefore P(u) = \sum_{k=0}^{n} BEZ_{k,n}(u) \cdot P_k}$$

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

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

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

for $n \geq k$, Recursive calculations.

$$C(n,k) = \frac{n-k+1}{k} C(n,k-1)$$

Bezier blending functions satisfy recursive relation ship

$$BEZ_{k,n}(u) = (1-u) BEZ_{k,n-1}(u) + u\, BEZ_{k-1,n-1}(u) \quad (4)$$

$n > k \geq 1$

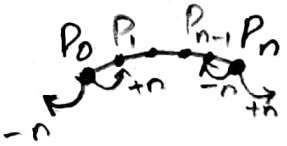
Properties of Bezier curve.

Useful property of Bezier curve is that the curve connects the first- and last control points

$$P(0) = P_0$$
$$P(1) = P_n$$

* First ~~dtgr~~ derivatives of Bezier curve at the endpoints

$$P'(0) = -nP_0 + nP_1$$
$$P'(1) = -nP_{n-1} + nP_n$$



from the above expressions,
Slope at the beginning of the curve is along the line joining first 2 control points
Slope at the end of the curve is along the line joining last 2 control points/endpoints.

* Second derivatives of Bezier curve - gives curvatures

$$P''(0) = n(n-1)[(P_2 - P_1) - (P_1 - P_0)]$$
$$P''(1) = n(n-1)[(P_{n-2} - P_{n-1}) - (P_{n-1} - P_n)]$$

* Important property of Bezier curve is that it lies within the convex hull (convex polygon boundary)

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

Cubic bezier curve

curve difinition in matrix form.

$$x(u) = (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u)u^2 x_2 + u^3 x_3$$

can be written as.

$$x(u) = (1-3u+3u^2-u^3)x_0 + (3u-6u^2+3u^3)x_1 + (3u^2-3u^3)x_2 + u^3 x_3.$$

$$= (-x_0+3x_1-3x_2+x_3)u^3 + (3x_0-6x_1+3x_2)u^2 + (-3x_0+3x_1)u + x_0$$

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$M_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$(a+b)^3 = a^3+b^3+3ab(a+b)$$

$$(a+b)^2 = a^2+b^2+2ab$$

$$(a-b)^2 = a^2+b^2-2ab$$

Bezier Surfaces

* 2 sets of orthogonal Bezier curves can be used to design an object surface.

* parametric vector function

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

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

* fig shows control points are connected by dashed lines, solid lines shows curves of constant u and constant v.

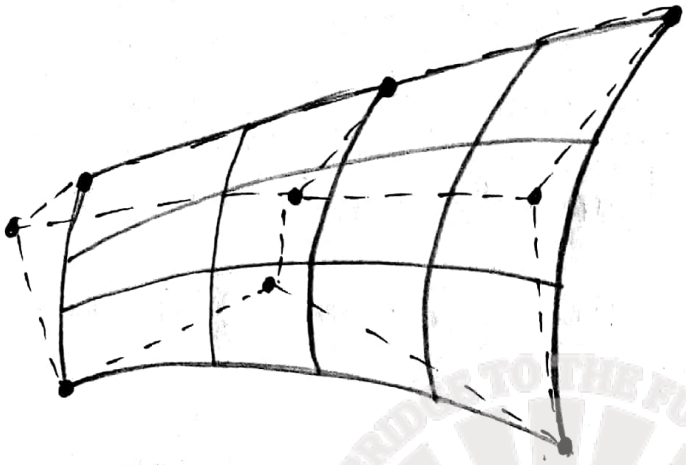
* Each curve of constant u is plotted by varying v over the interval from 0 to 1, with u fixed at one of the values in this unit interval.

* Curves of constant v are plotted similarly.

* 3D co-ordinate position for control point can be specified
  - construct a rectangular grid in 'xy' plane/ xy "ground" plane.
  - choose elevations above the ground plane at grid intersections as z-co-ordinate value for control point.

* figure illustrates a smooth surface formed with two Bezier sections.
  smooth transition from one section to other is assured by establishing both zero order and first order continuity at boundary line.
  - zero order continuity → matching control points at boundary.

first order continuity → control points along a straight line across boundary.



---- control points are connected

——— curve of constant u & v.