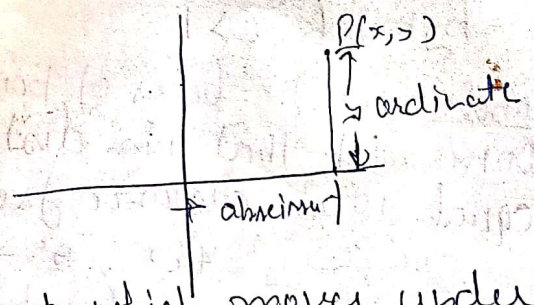
6 Sept

Co-ordinates of a point:- There is a one-one correspondence between set of points in a plane & the set of ordered pair of real number



Locus:-
The path described by a point which moves under a given condition is called its locus.

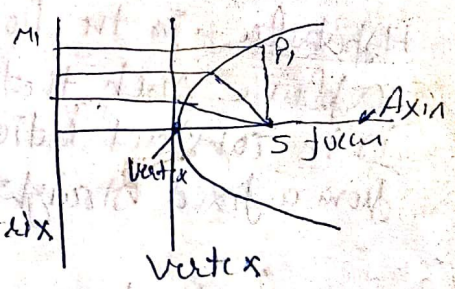
Circle:- A circle is the locus of a point which moves in a plane such that its distance from fixed point always remains constant.

$$(x-h)^2+(y-k)^2=r^2$$

Conic Section or Conic:- The locus of a point which moves in a plane such that its distance from a fixed point is in a constant ratio to its distance from a fixed straight line is called conic section or conic.

The fixed point is called the focus & is generally denoted by S

The fixed st. line is called the directrix.



The constant ratio is called the eccentricity & is denoted by e

The straight line passing through the focus & perpendicular to the directrix is called the axis of conic section.

The point where axis meet, the conic is called vertex of conic section

parabola if $e=1$

ellipse $e<1$

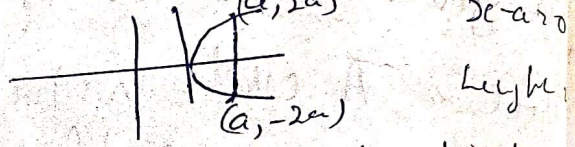
hyperbola $e>1$

Parabola: is the locus of point which moves in a plane such that its distance from fixed point is equal to its distance from a fixed straight line

$y^2 = 4ax$, $y^2 = -4ax$, $x^2 = 4ay$, $x^2 = -4ay$

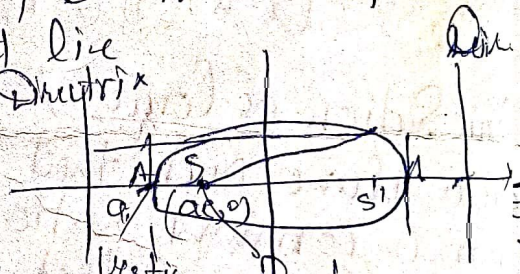
Latus Rectum: - of a parabola is chord passing through the focus & perpendicular to the axis of parabola.

$\frac{4a^2}{4a}$



Ellipse: An ellipse is the locus of a point which moves in a plane such that its distance from a fixed point is in a constant ratio, less than one, to its distance from a fixed straight line

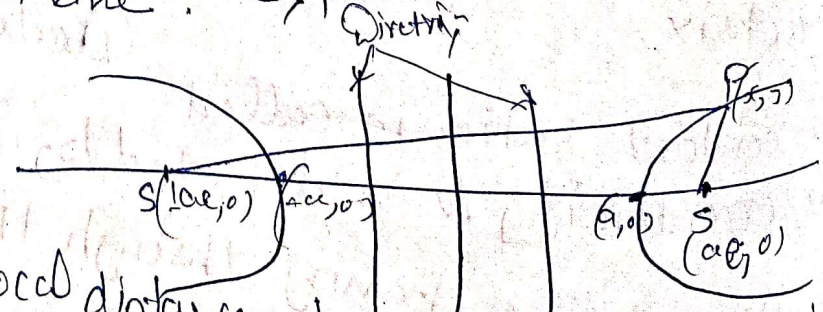
$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$



Sum of the focal distance of a point on an ellipse is constant & equal to major axis

Hyperbola: is the locus of a point which moves in a plane such that its distance from a fixed point is in constant ratio, greater than one, to its distance from a fixed straight line. $e>1$

$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$



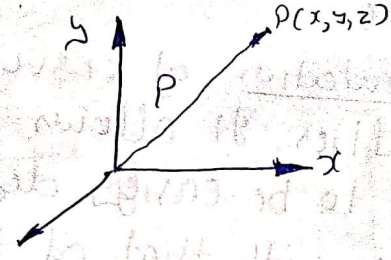
Difference of the focal distance is constant & equal to the length of transverse axis.

$S'P - SP = 2a$

Curve Representation :-

Curve can be described mathematically by non-parametric or parametric equations. Non-parametric equations can be explicit or implicit. For a non-parametric curve the coordinates y & z of a point on the curve as two separate functions of the third coordinate x as the independant variable (4.1). This curve representation is known as the nonparametric explicit form. Explicit non parametric representation of a general-3D curve takes the form:

$$P = [x \quad y \quad z]^T = [x \quad f(x) \quad g(x)]^T \qquad (4.1)$$



where P is the position vector of point P as shown above Eqn 4.1 is a one-to-one relationship. Thus, this form cannot be used to represent closed (eg:- circles) or multivalued curves (eg:- parabola).

→ If the co-ordinates x, y & z are related together by two ~~equations~~ functions (see eqn 4.2), a non parametric implicit form results. The implicit nonparametric representation can represent circles & parabola & is given by intersection of two surfaces as

$$F(x, y, z) = 0$$
$$G(x, y, z) = 0 \qquad 4.2$$

However, the equation must be solved to find its roots (y & z values) if a certain value of x is given. This may be inconvenient & lengthy. Other limitations of non parametric representation 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 & programming-wise, other ill-defined mathematical conditions may result.

2. Shapes of most eg. objects intrinsically independent of any co-ordinate system. What determine the shape of object is the relationship between its data point themselves & not between these points & some arbitrary coordinate system.

3. If the curves is to be displayed as a series of points or straight line segments the computation involved could be extensive.

Parametric representation of curves overcomes all the above difficulties. It allows closed & multiple valued functions to be easily defined & 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 & cubics, the equations are polynomial rather than equations involving roots. Hence, the parametric form is not only more general but it is also well suited to computations & display.

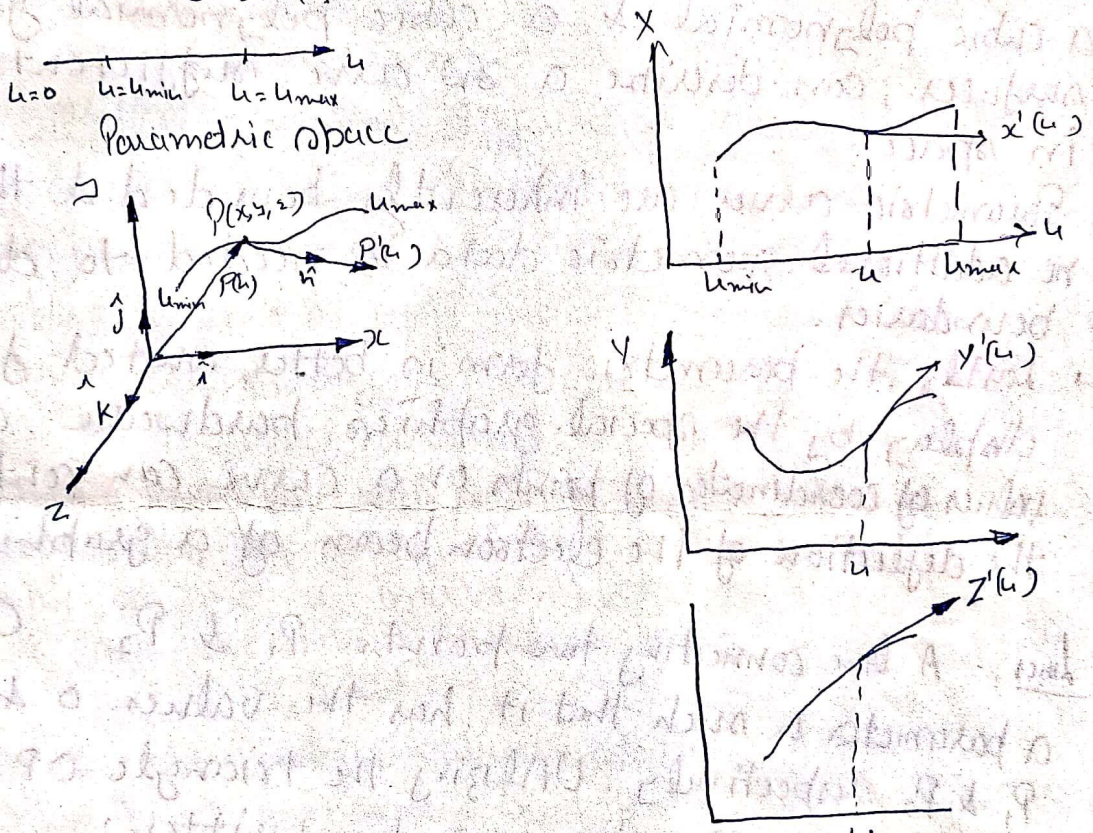
In parameteric form, each point on a curve is expressed as a function of a parameter u. The parameter acts as a local coordinate for locating on the curve. The parametric equation for a 3D curve in space takes the following vector form:

$$P(u) = [x \; y \; z]^T = [x(u) \; y(u) \; z(u)]^T \quad (4.3)$$

$$u_{min} \leq u \leq u_{max}$$

Eqn 4.3 implies that the coordinates of a point on the curve are the combin...

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). The parametric curve is bounded by two parametric values u_{min} & u_{max}. It is, however, convenient to normalize the parametric variable u to have the limits 0 & 1.



Curve Component in Parametric Space

Parametric Representation of 3D - Curve

→ The parametric forms facilitates many of the useful related ~~component~~ computations in geometric modeling.

→ To check whether a given point lies on the curve or not reduces to finding the corresponding u values & checking whether that values lies in the stated u range.

→ Geometrical Transformations can be performed

directly on parametric equations.

→ Parametric geometry can be easily expressed in terms of vectors & matrices which enables the use of simple computation techniques to solve complex analytical geometry problems.

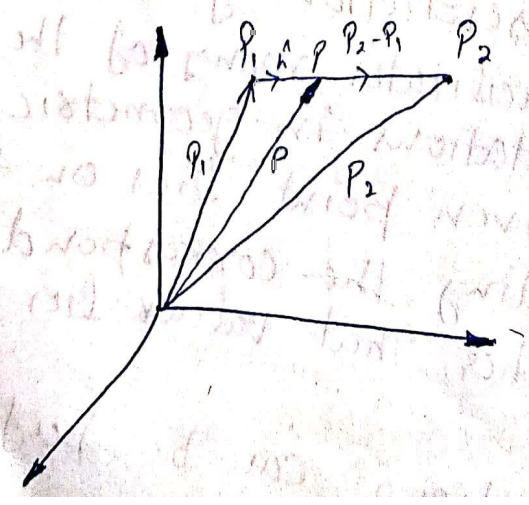
→ In addition, common forms for curves which are extendable to surfaces can be found. For example a cubic polynomial & a cubic polynomial for surfaces, can describe a 3D curve sufficiently in space.

→ Parametric curves are inherently bounded & 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 graphic display.

Lines :- A line connecting two points P_1 & P_2. Define a parameter u such that it has the values 0 & 1 at P_1 & P_2 respectively. Utilizing the triangle OPP_1, the following equation can be written:

$$P = P_1 + (P - P_1)$$



However, the vector $(P-P_1)$ is proportional to the vector P_2-P_1 such that

$$(P-P_1) = u(P_2-P_1)$$

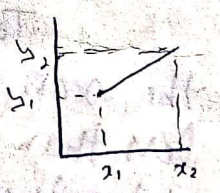
$$\Rightarrow P = P_1 + u(P_2-P_1) \qquad 0 \le u \le 1$$

In scalar form, this equation can be written as

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

DDA Algorithm: Digital differential analyzer (DDA) is scan conversion line algorithm based on calculating either Δy or Δx, using $\Delta y = m \Delta x$. We sample the line at unit intervals in one coordinate & determine corresponding integer values nearest the line path for the other coordinate.

Consider first a line with positive slope. If the slope is less than or equal to 1, we sample at unit x intervals ($\Delta x = 1$) & compute each successive y values as



$$y_{k+1} = y_k + m$$

Subscript k takes integer values starting from 1, for the first point & increases by 1 until the final endpoint is reached. Since m can be any real number between 0 & 1, the calculated y values must be rounded to the nearest integer.

For lines with a positive slope greater than 1, we reverse the roles of x & y. That is, we sample at unit y intervals ($\Delta y = 1$) & calculate each succeeding x value as

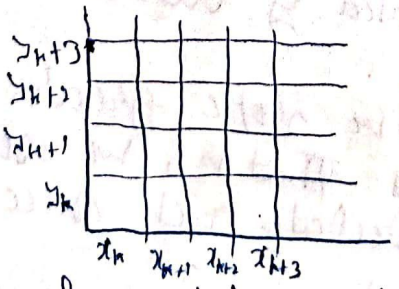
$$x_{k+1} = x_k + \frac{1}{m}$$

The DDA algorithm is a faster method for calculating pixel positions than direct use of $y = mx + b$

It eliminates the multiplication by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to step to pixel positions along the line path. The accumulation of round off error in successive additions of the floating-point increment, however, can cause the calculated pixel positions to drift away from the true line path for long line segments.

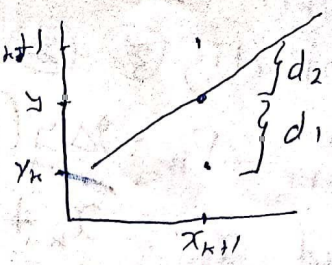
Bresenham's Line Algorithm :- An accurate & efficient raster line-generating algorithm, developed by Bresenham's, scan converts lines using only incremental integer calculations that can be adapted to display circles & other curves.

To illustrate Bresenham's approach, we first consider the scan conversion process for lines with positive slope less than 1. Pixel positions along a line path are then determined by sampling at unit x intervals. Starting from the left end point (x_0, y_0) of a given line, we step to each successive column (x position) & plot the pixel whose scan-line y value is closest to the line path. Fig 1 demonstrates the kth step in this process.



Assuming we have determined that the pixel at (x_k, y_k) is to be displayed, we next need to decide which pixel to plot in column x_{k+1}. Our choices are the pixels at positions (x_k+1, y_k) & (x_k+1, y_k+1).

At sampling position x_{k+1}, we label vertical pixel separations from the mathematical line path as d_1 & d_2. The y coordinate on mathematical line path at pixel column position x_k+1 is calculated as



$$y = m(x_k+1) + b$$

Then $d_1 = y - y_k$

$$= m(x_k+1) + b - y_k$$

& $d_2 = (y_k+1) - y$

$$= y_k + 1 - m(x_k+1) - b$$

The diff. b/w these two separation is

$$d_1 - d_2 = 2m(x_k+1) - 2y_k + 2b - 1 \quad (1)$$

A decision parameter P_k for the k^{th} step in the line algorithm can be obtained by rearranging eq (1) so that it involves only integer calculations.

We accomplish this by substituting $m = \Delta y/\Delta x$, where Δy & Δx are the vertical & horizontal separation of the endpoint positions & defining:

$$P_k = \Delta x(d_1 - d_2)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c \quad (2)$$

where $c = 2\Delta y + \Delta x(2b-1)$

Coordinate changes along the line occur in unit steps in either the x or y directions. Therefore, we can obtain the values of successive decision parameters using incremental integer calculations. At step $k+1$, the decision parameter

is evaluated from eqn (2) as

$$p_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c \quad (3)$$

Subtracting (3) - (2)

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

But $x_{k+1} = x_k + 1$, so that

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

where the term $y_{k+1} - y_k$ is either 0 or 1, depending upon the sign of parameter p_k.

This recursive calculations of decision para is performed at each integer position starting from the left co-ordinate endpoint of the line. The first parameter, p_0, is evaluated from eqn (2) at the starting pixel position (x_0, y_0) & with m evaluated as $\Delta y/\Delta x$

$$p_0 = 2\Delta y - \Delta x$$

irder Cir
most com
The basic
written

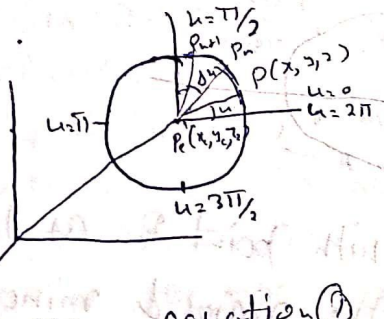
Assumin
for simp
u is the
P on t

For dis
nerate po
n from
with line
insufficie
unction
method i
Assumin
point P
circumfe
written

Circles: Circles & circular arcs are among the most common entities used in wireframe modeling. The basic parametric equation of a circle can be written as

$$\left.\begin{aligned} x &= x_c + R\cos u \\ y &= y_c + R\sin u \end{aligned}\right\} \quad 0 \le u \le 2\pi \qquad (1)$$
$$z = z_c$$

Assuming that the plane of 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.



For display purposes, equation (1) 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. So, this is insufficient method due to computing the trigonometric functions in the equation for each point. A less computational method is to write eqn (1) in an incremental form. Assuming there is an increment Δu between two consecutive point $P_n(x_n, y_n, z_n)$ & $P(x_{n+1}, y_{n+1}, z_{n+1})$ on the circle circumference, the following recursive relationship can be written

$$x_n = x_c + R\cos u$$
$$y_n = y_c + R\sin u$$
$$x_{n+1} = x_c + R\cos(u+\Delta u)$$
$$y_{n+1} = y_c + R\sin(u+\Delta u)$$

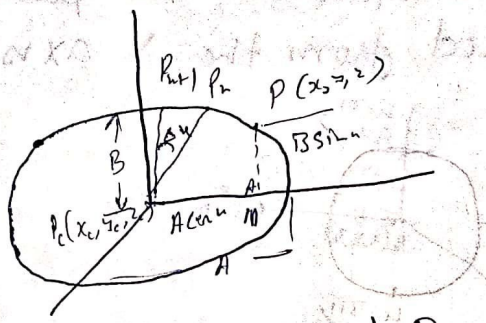
$$x_{n+1} = x_c + R\cos u\cos\Delta u - R\sin u\sin\Delta u$$
$$= x_c + (x_n - x_c)\cos\Delta u - (y_n - y_c)\sin\Delta u$$
$$y_{n+1} = y_c + (y_n - y_c)\cos\Delta u + (x_n - x_c)\sin\Delta u$$
$$z_{n+1} = z_n$$

Thus, the circle can start from an arbitrary point & successive points with equal spacing can be calculated by recursively. Cosδu & sinδu have to be calculated only once.

Ellipse :- $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

The parametric form of representation of ellipse is

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



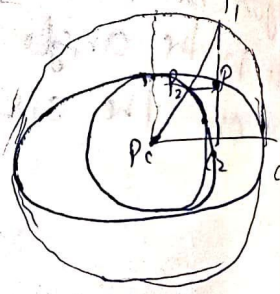
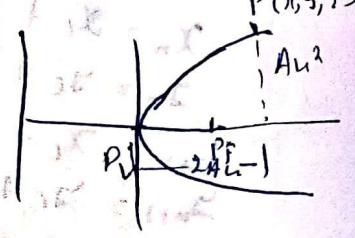


Fig shows an ellipse with point P_c as the centre & the length of half of the major & minor axes are A & B respectively. The parametric equation of an ellipse can be written as assuming the plane of ellipse in the xy plane. The parameter u is the angle as in the case of a circle.

$$x_{n+1} = x_c + A\cos u$$
$$x_{n+1} = x_c + A\cos(u+\delta u)$$
$$= x_c + A\cos u\cos\delta u - A\sin u\sin\delta u$$
$$x_{n+1} = x_c + (x_n - x_c)\cos\delta u - \frac{A}{B}(y_n - y_c)\sin\delta u$$
$$y_{n+1} = y_c + (y_n - y_c)\cos\delta u + \frac{A}{B}(x_n - x_c)\sin\delta u$$

Parabolas :-

$$x = x_v + Au^2$$
$$y = y_v + 2Au \qquad 0 \le u \le \infty$$
$$z = z_v$$



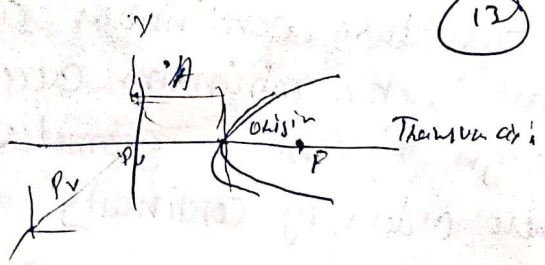
bc
calc

Hyperbola

$x = x_v + A \cosh u$

$y = y_v + B \sinh u$

$z = z_v$



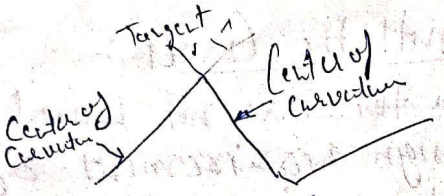
Parametric Representation of Synthetic Curves :-

Analytical curves described in the are not usually sufficient to meet geometric design requirement of mechanical parts. Products such as car bodies, ship hulls, airplane fuselage & wings, propeller blades shoe insoles & bottles are a few examples that require free-form, or synthetic, curves & surfaces.

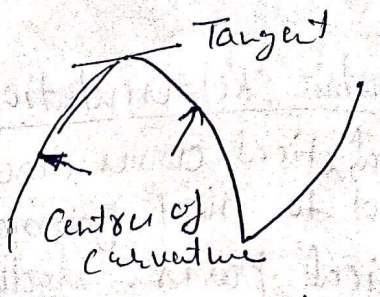
The need for synthetic curves in design arises on two occasions: when a curve is represented by a collection of measured data points, & when an existing curve must change to meet the new design requirements. In the latter occasion, the designer would need a curve representation that is directly related to the data points & is flexible enough to bend, twist or change the curve shape by changing one or more data points. Data points are usually called the control points & the curve itself is called an interpolant if it passes through all the data points.

Polynomials are the typical form of these curves. Various continuity requirements can be specified at the data points to impose various degree 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 several curve segments pieced together end to end, a position continuous curve. First C^1 & second C^2 order continuities imply slope

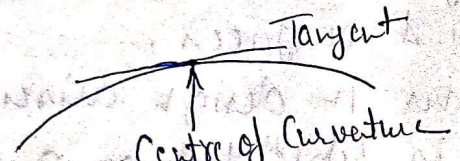
& curvature continuous curves respectively. A C¹ curve is the minimum acceptable curve for engineering design. Fig 1 shows geometrical interpretation of these orders of continuity.



Zero-order Continuity



First order continuity



Second order of Continuity

Fig 1

A cubic polynomial is the minimum order polynomial that can guarantee the generation of C^0, C^1 or C^2 curves. Hermite cubic spline, Bezier & B-spline are generally used in CAD/CAM systems.

Hermite Cubic Spline :-

Parametric splines curves are defined as piecewise polynomials 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 & B-splines curves do.

The parametric cubic spline curves connect two data (end) points & utilizes a cubic equation. Therefore, four conditions are required to determine the co-efficient of the equation. When there are

the positions of two end points & 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.

$$P(u) = \sum_{i=0}^{3} C_i u^3 \qquad 0 \le u \le 1 \qquad (1)$$

where u is the parameter & C_i are polynomial co-efficients. In scalar form this equation is written as

$$x(u) = C_{3x}u^3 + C_{2x}u^2 + C_{1x}u + C_{0x} \qquad (2)$$

$$y(u) = C_{3y}u^3 + C_{2y}u^2 + C_{1y}u + C_{0y} \qquad (3)$$

$$z(u) = C_{3z}u^3 + C_{2z}u^2 + C_{1z}u + C_{0z} \qquad (4)$$

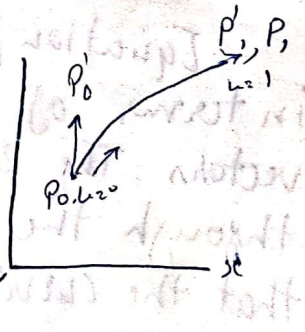
$$\therefore P(u) = C_3u^3 + C_2u^2 + C_1u + C_0 \qquad (5)$$

This equation can also be written in a matrix form as $P(u) = U^T C$ (6)

where $U = [u^3\ u^2\ u\ 1]^T$ & $C = [C_3\ C_2\ C_1\ C_0]^T$. C is called the co-efficient vector.

The tangent vector to the curve at any point is given by differentiating eqn (1)

$$P'(u) = \sum_{i=0}^{3} C_i\, i\, u^{i-1} \qquad 0 \le u \le 1 \qquad (7)$$



At $u=0$, $P(u) = P_0$, $P'(u) = P_0'$

& At $u=1$ $P(u) = P_1$, $P'(u) = P_1'$

By putting in eqn (6) & (7)

$P_0 = C_0$

$P_0' = C_1$ [eqn (7) $P'(u) = 3u^2C_3 + 2uC_2 + C_1$]

$$C_2 = 3(P_1 - P_0) - (2P_0' - P_1')$$

& $C_3 = 2(P_0 - P_1) + P_0' + P_1'$

By putting the value of co-efficient C_0, C_1, C_2 & C_3 in eqn 5

$$\therefore P(u) = [2(P_0 - P_1) + P_0' + P_1']u^3 + [3(P_1 - P_0) - (2P_0' - P_1')]u^2 + P_0' u + P_0$$

By rearranging

$$P(u) = (2u^3 - 3u^2 + 1)P_0 + (2u^3 + 3u^2)P_1 + (u^3 - 2u^2 + u)P_0' + (u^3 - u^2)P_1' \quad 0 \le u \le 1 \quad (8)$$

$$\therefore P'(u) = (6u^2 - 6u)P_0 + (-6u^2 + 6u)P_1 + (3u^2 - 4u + 1)P_0' + (3u^2 - 2u)P_1' \quad 0 \le u \le 1 \quad (9)$$

The functions of u in equation (8) & (9) are called blending functions.

Eqn (8) can be written in a matrix form as

$$P(u) = U^T [M_H] V \quad 0 \le u \le 1$$

where M_H is the Hermite matrix & V is the geometry vector.

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

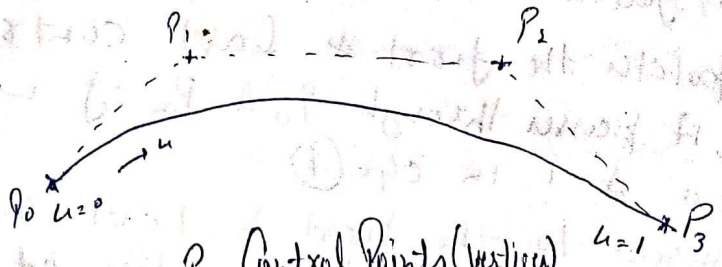
$$V = [P_0 \; P_1 \; P_0' \; P_1']^T$$

Equation (8) describe the cubic spline curve in terms of its two end-points & their tangent vectors. The equation shows that the curve passes through the endpoints ($u = 0$ & 1). It also shows that the curve's shape can be controlled by changing its endpoint or its tangent vectors.

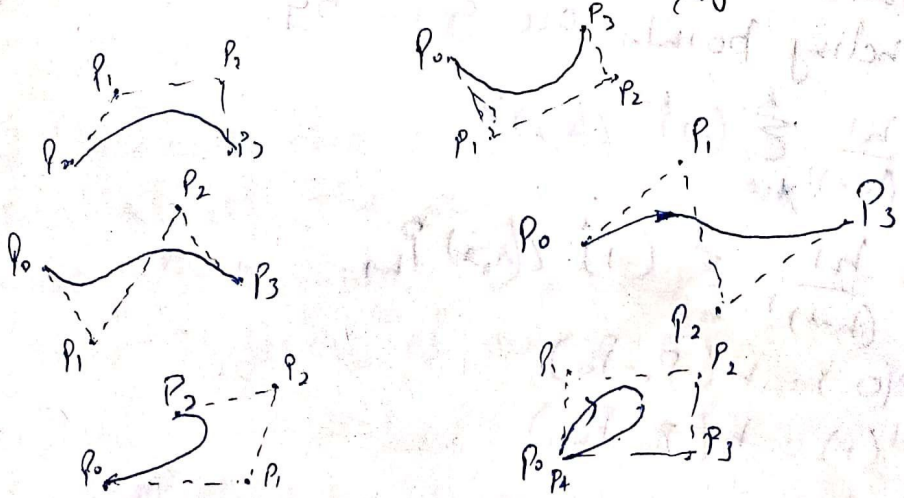
Bezier Curves:

Bezier curves are based upon approximation techniques which produces (curves) that do not pass through the given data points. Instead, these points are used to control the shape of resulting curves.

The Bezier curve is defined in terms of the location 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. Only first & last control points or vertices of the polygon actually lie on the curve. The other vertices define the order, derivatives & shape of the curve. The curve is also always tangent to the first & last polygon segments. In addition, the curve shape tends to follow the polygon shape. There three observations should enable the user



P_k Control Points (vertices)
--- Characteristic Polygon



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

$$P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u), \quad 0 \le u \le 1 \qquad (1)$$

where $P(u)$ is any point on the curve & P_i is a control point, $B_{i,n}$ are the Bernstein polynomial. The Bernstein polynomial serves as the blending or basis functions for the Bezier curve & is given by

$$B_{i,n}(u) = C(n,i)\, u^i (1-u)^{n-i} \qquad (2)$$

where $C(n,i)$ is the binomial co-efficient

$$C(n,i) = \frac{n!}{i!(n-i)!} \qquad (3)$$

∴ (1) becomes

$$P(u) = P_0 C(n,0) u^0 (1-u)^n + P_1 C(n,1) u (1-u)^{n-1} + P_2 {}^nC_2 u^2 (1-u)^{n-2} + \dots + P_{n-1} C(n,n-1) u^{n-1}(1-u) + P_n u^n \quad 0 \le u \le 1$$

Properties are as follows.

1. The curve interpolates the first & last control points: that is, it passes through P_0 & P_n if we substitute $u = 0$ & 1 in eqn (1)
2. The curve is tangent to the first & last segments of the characteristic polygon. n^{th} derivatives at the starting & ending points are given by

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

$$P'(1) = \frac{n!}{(n-1)!} \sum_{i=0}^{1} (-1)^i C(1,i) P_{n-1}$$

$$\therefore P'(0) = n(P_1 - P_0)$$

$$P'(1) = n(P_n - P_{n-1})$$

where $(P_1 - P_0)$ & $(P_n - P_{n-1})$ define the first & last segments of the curve polygon. Similarly, it can be shown that the second derivative at P_0 is determined by P_0, P_1 & P_2; or in general, the kth derivative at an endpoint is determined by its k neighbouring vertices.

3. The curve is symmetric with respect to u & $(1-u)$. This means that the sequence of control points defining the curve can be reversed without the change of the curve shape. This can be achieved by substituting $1-u = v$ in eqn (4) & noticing that $C(n,i) = C(n,n-i)$. This is result of the fact that $B_{i,n}(u)$ & $B_{n-i,n}(u)$ 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 $\frac{d(B_{i,n})}{du} = 0$

$$B_{i,n}(u) = {}^nC_i\, u^i (1-u)^{n-i}$$

$$\frac{dB}{du} = {}^nC_i\, i u^{i-1}(1-u)^{n-i} + {}^nC_i\, u^i (n-i)(1-u)^{n-i-1}(-1) = 0$$

$${}^nC_i\, u^i (1-u)^{n-i}\left[\frac{i}{u} + \frac{i-n}{1-u}\right] = 0$$

$$u = i/n$$

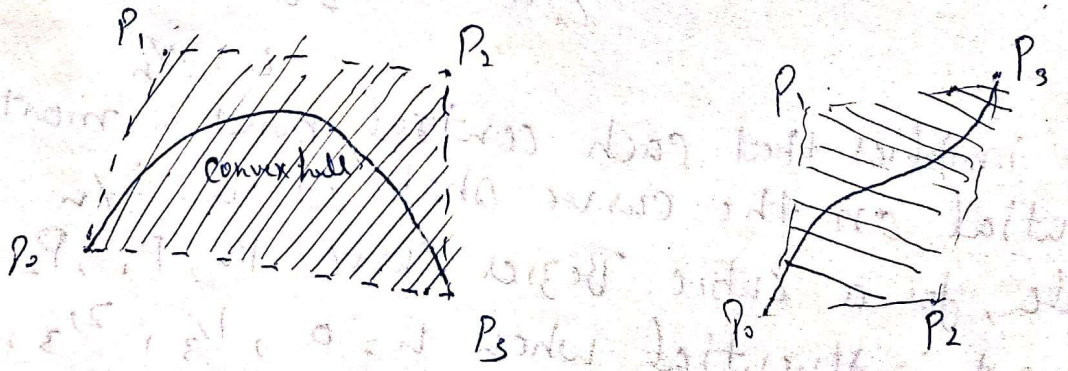
This implies that each control point is most influential on the curve shape at $u = i/n$. For example, for a cubic Bezier curve P_0, P_1, P_2, P_3 are most influential when $u = 0, 1/3, 2/3$ & 1 respectively. Therefore, each control point is weighted by its blending function for each u value.

5. The curve shape can be modified by either changing one or more vertices of its polygon or by keeping the polygon fixed & specifying multiple coincident point at a vertex.

6. A closed Bezier curve can simply be generated by closing its characteristics polygon or choosing P_0 & P_n to be coincident.

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.

8. 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. 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 closed polygon & in three dimensions it is a polyhedron.



Convex hull of Bezier Curve

B-spline Curves :-

B-spline curves provide another effective method of generating curves defined by polygons. In fact, B-spline curve are the proper & powerful generalizat of Bezier curves. In addition to sharing most of the characteristics of Bezier curves they enjoy some other unique advantages. They provide the 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.

In contrast to Bezier curve, the theory of B-spline curves separates the degree of the resulting curve from the number of the given control points, while four 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.

The B-spline curve defined by n+1 control points P_i is given by

$$P(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \qquad 0 \le u \le u_{max} \qquad (1)$$

$N_{i,k}(u)$ are the B-spline functions. Thus B-spline curve have a B-spline basis. The control points (sometime called the deboor points) form the vertices of the control

or deboor polygon.

The B-spline function have the following properties

Partition of unity $\sum_{i=0}^{n} N_{i,k}(u) = 1$ (2)

Positivity $N_{i,k}(u) \geq 0$ (3)

Local Support $N_{i,k}(u) = 0$ if $u \notin [u_i, u_{i+k+1}]$ (4)

Continuity $N_{i,k}(u)$ is $(k-2)$ times continuously differentiable (5)

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

$$N_{i,k}(u) = (u-u_i)\frac{N_{i,k-1}(u)}{u_{i+k-1}-u_i} + (u_{i+k}-u)\frac{N_{i+1,k-1}(u)}{u_{i+k}-u_{i+1}} \quad (6)$$

where $N_{i,1} = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases}$ (7)

Choose $0/0 = 0$ if the denominator of eqn (6) become zero.

Eqn (7) 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 $(k-1)$ (recall 6)

The u_i are called parametric knots or knot values. These values form a sequence of non decreasing integers called the knot vectors. The values of the u_i depend on whether the B-spline curve is an open (nonperiodic) or closed (periodic) curve. For an open curve, they are given by

$$u_j = \begin{cases} 0 & j < k \\ j-k+1 & k \leq j \leq n \\ n-k+2 & j > n \end{cases} \quad (8)$$