Unit 4

Curves and Surfaces

Curves and Surfaces

- Curve Representation
- Non-parametric and parametric curves
- Representation of space curves
- Cubic Spline
- Bezier curves
- Z- buffer algorithm
- Warnock algorithm

Curve Representation

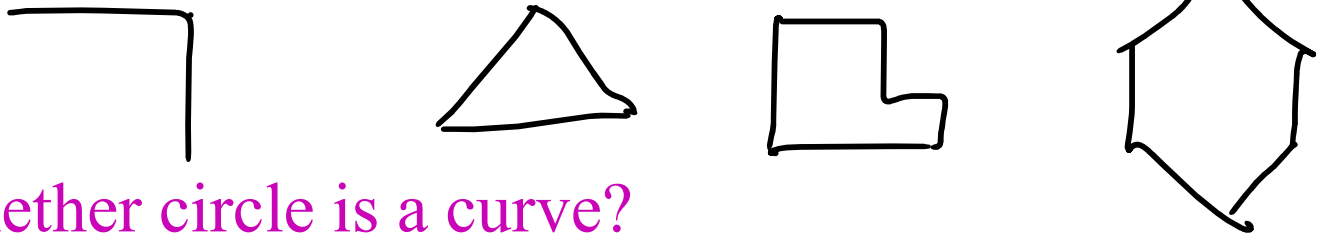
- A curve may be represented as a collection of points.

Def 1: When set of point infinite or finite are joined continuous then it is called a curve.

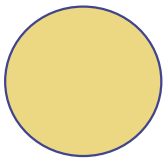
Def 2: When we start from a point for drawing some geometrical figure and end at some point without any gap, we call it as a curve.

Whether line is a curve?

- If line is a curve then all geometrical figures generated by lines are also curves.



Whether circle is a curve?

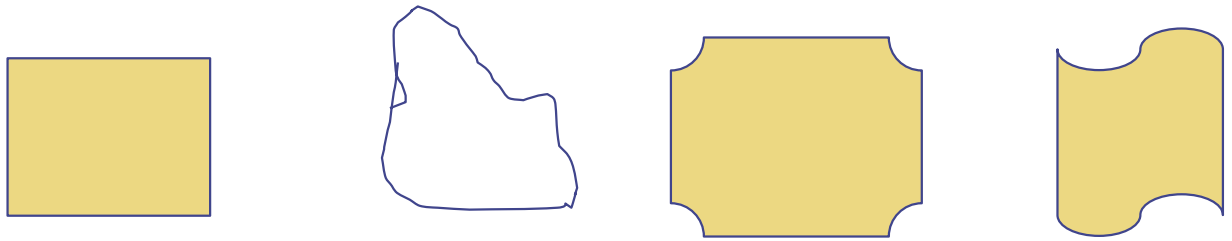
-  This circle is a curve but  this is not a curve (gap).

Curve Representation

- **Types of curves:**

Open Curve:

Closed Curve:



Crossing Curve:

Curve Representation

- Point representation of curves :
  - Points along the curve as shown in fig 4.1(a) are equally spaced of the same plane curve. Connection of the points by short straight line segments generates a poor representation of curve.
  - Increasing the point density in these regions, improves the representation as shown in fig 4.1(b).

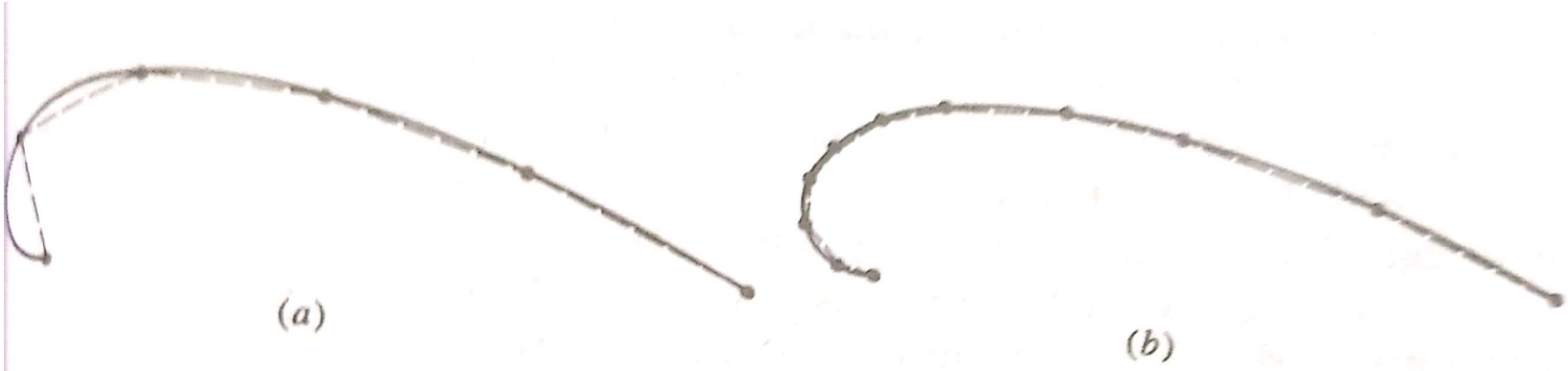
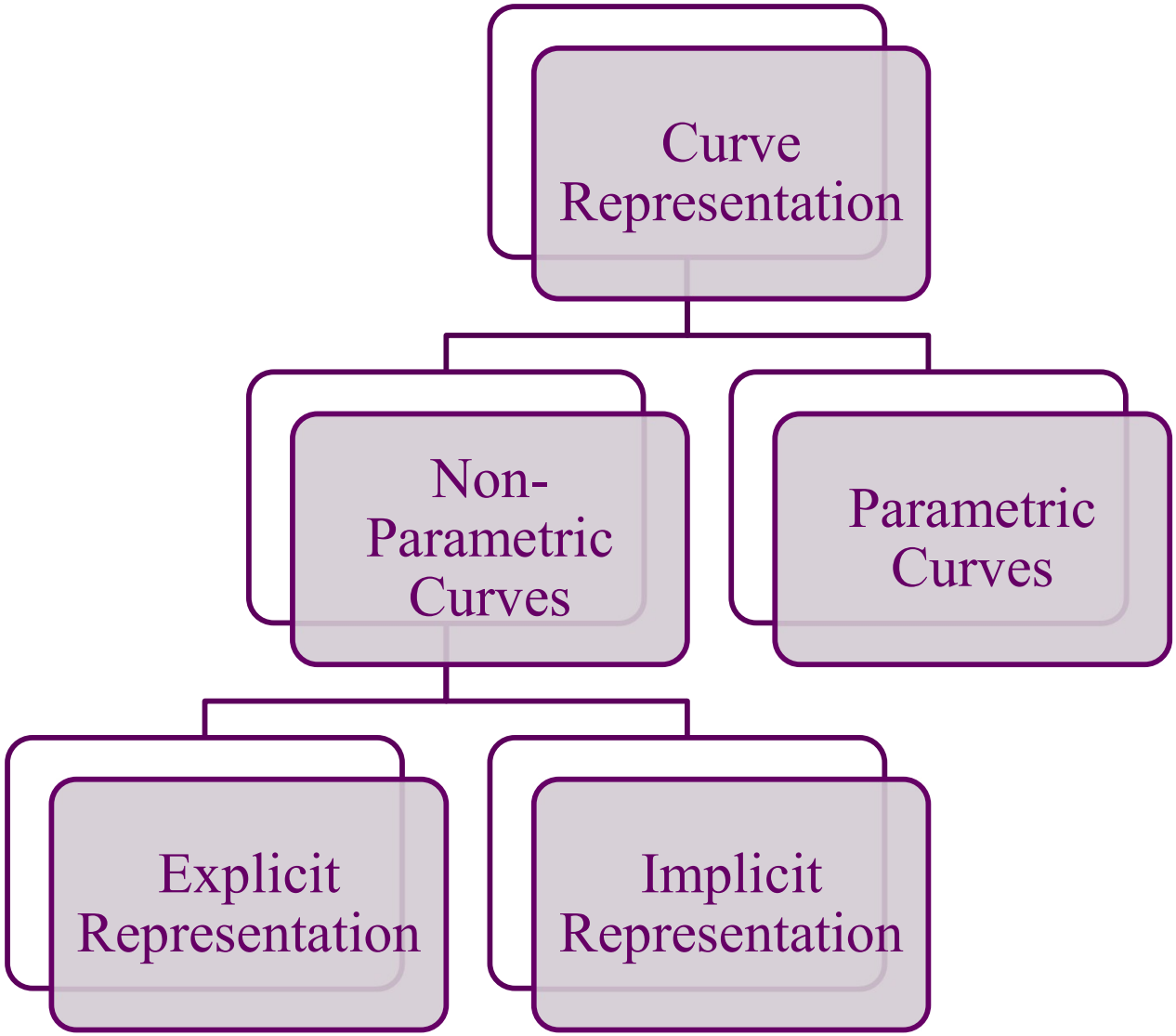


Figure 4–1 Point representations of curves. (a) Equal point density along the curve length; (b) point density increases with decreasing radius of curvature.

Curve Representation

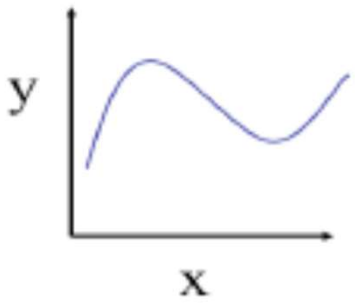


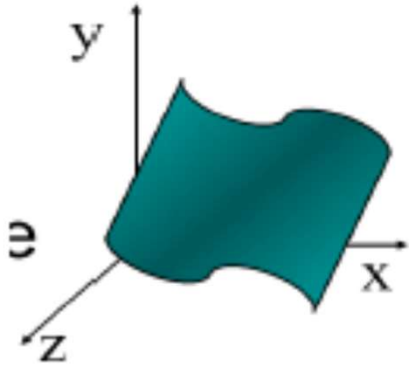
Non-Parametric Curves

- Non-Parametric Curves:
  - Explicit
  - Implicit

Explicit Representation

- The explicit form of a curve in two dimensions gives the value of one variable, the dependent variable, in terms of the other, the independent variable.
- Ex: In x, y space, we might write y = f (x) – the equation of a straight line is y = mx + b, means for each x-value only one y-value is obtained.





- Open curves can be represented explicitly.

Implicit Representation

- Closed or multi-value curves (multiple y values for single value of x) like circle, cannot be represented explicitly.
- In this, dependent variable is not expressed in terms of some independent variables.
- In two dimensions, an implicit curve can be represented by the equation ***f (x, y) = 0. ex: $x^2 + y^2 - R^2 = 0$ – equation of circle***
  - The implicit form is less coordinate-system dependent than is the explicit form.
- A general 2nd degree implicit representation:

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

- This form of the expression, with the coefficients, provide a wide variety of 2D curve forms called CONIC SECTIONS.

Implicit Representation – Conic Section

- Three types of conic sections:
  - Parabola
  - Hyperbola
  - Ellipse

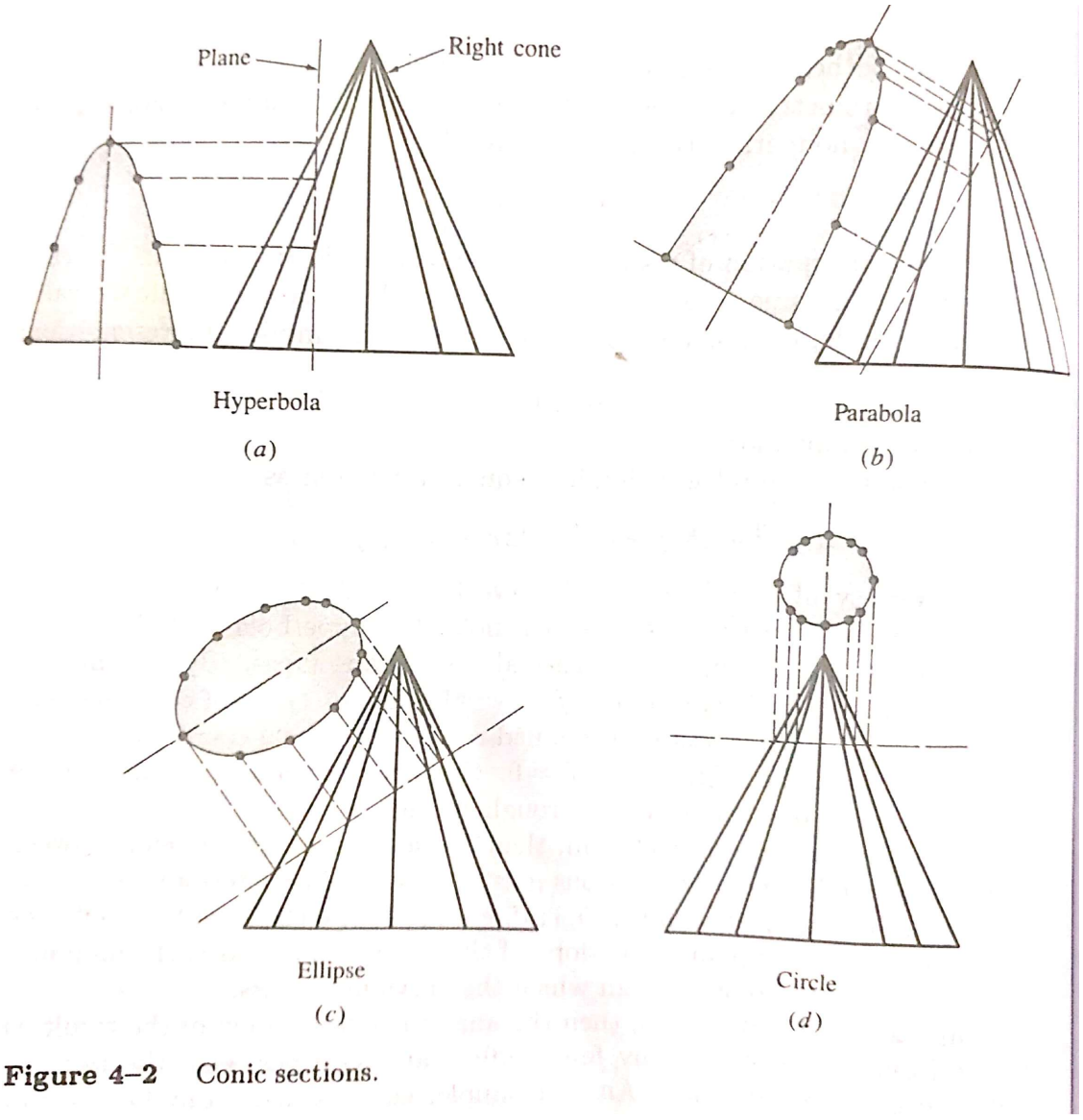


Figure 4–2 Conic sections.

Generalized CONIC

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

Re-organize:

$$\Rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Implicit Representation - Example

- If the conic passes through the origin: f = 0.
- If b = 0 and c = 1.0, then curve is fixed by specifying 4 additional conditions:
  - Two end points and
  - Two end slopes
- If a =1.0, b = 0 and c = 1.0, then the form of the curve is:

$$x^2 + y^2 + 2dx + 2ey + f = 0$$

Three conditions required to fix d, e and f are:

- two end points and

- either the slope at the beginning or at the end of curve segment.

- A straight line is obtained by using a = b = c = 0.

dx + ey + f = 0

y = -(d/e)x – f/e = mx + b'

Where m is slope of line b' is y intercept.

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

Implicit Representation

Special Conditions:

- If $b^2 = ac$, the equation represents a PARABOLA;
- If $b^2 < ac$, the equation represents an ELLIPSE;
- If $b^2 > ac$, the equation represents a HYPERBOLA.

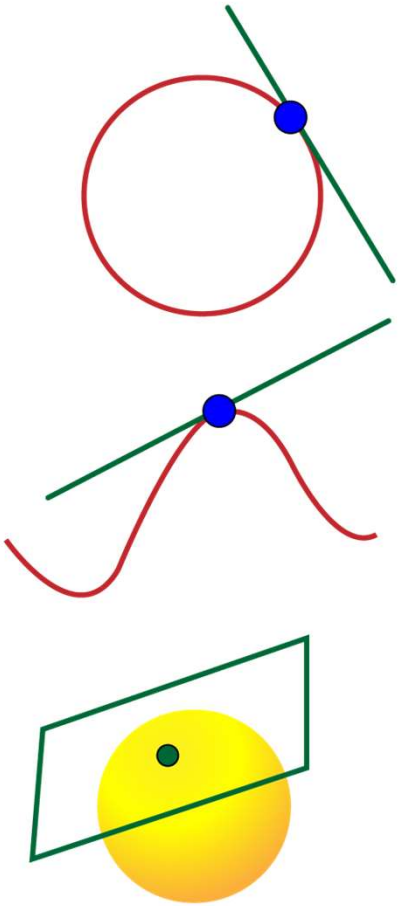
Parametric Curves

- The parametric form is suitable for representing closed and multivalued curves. Eg circle
- In parametric curves each coordinate of a point on a curve is represented as a function of a single parameter.
- The position vector of a point on the curve is fixed by the value of the parameter.
- For a 2D curve with t as the parameter, the Cartesian coordinates of a point on the curve are
  - $x = x(t)$
  - $y = y(t)$
- The position vector of a point on the curve is then:
  - $P(t) = [x(t) \quad y(t)]$

Parametric Curves

- The derivative or tangent vector on a parametric curve is given by
  - $P'(t) = [x'(t) \quad y'(t)]$

  Where the ‘ denotes differentiation with respect to the parameter.
- The slope of the curve, dy/dx, is
  - $dy/dx = (dy/dt)/(dx/dt) = y'(t)/x'(t)$
- Since a point on a parametric curve is specified by a single value of the parameter, the parametric form is axis dependent



Tangent to a circle

Tangent line to a curve

Tangent plane to a sphere

Parametric Curves - Example

Parametric Representation of a Straight Line:

- For two position vectors P1 and P2 a parametric representation of the straight line segment between them is
  - $P(t) = P1 + (P2 - P1)t \qquad 0 \le t \le 1$
- Since P(t) is a position vector, each of the components of P(t) has a parametric representation x(t) and y(t) between P1 and P2
  - $x(t) = x1 + (x2 - x1)t \qquad 0 \le t \le 1$
  - $y(t) = y1 + (y2 - y1)t$

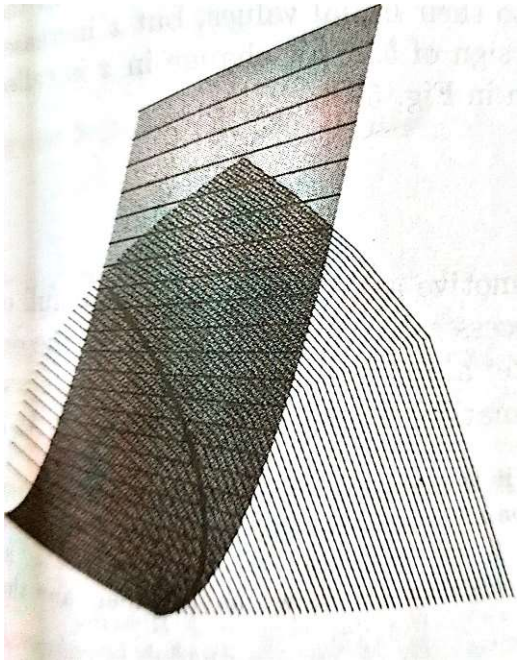
Parametric Curves - Example

- Ex: For position vectors P1[1 2] and P2[4 3] determine the parametric representation of the line segment between them. Also determine the slope and tangent vector of the line segment.
- A parametric representation is
  - $P(t) = P1 + (P2 - P1)t = [1\ \ 2] + ([4\ \ 3] - [1\ \ 2])t \qquad 0 \le t \le 1$
  - $P(t) = [1\ \ 2] + [3\ \ 1]t \qquad 0 \le t \le 1$
- Parametric representation of the x and y components are
  - $x(t) = x1 + (x2 - x1)t = 1 + 3t \qquad 0 \le t \le 1$
  - $y(t) = y1 + (y2 - y1)t = 2 + t$
- The tangent vector is obtained by differentiating P(t)
  - $P'(t) = [x'(t)\ \ y'(t)] = [3\ \ 1]$ or
  - $Vi = 3i + j$
- Where vi is the tangent vector and i, j are unit vectors in x, y direction, respectively
  - $dy/dx = (dy/dt)/(dx/dt) = y'(t)/x'(t) = 1/3$

Representation of space curves (3D)

- 3D space curves are represented non parametrically or parametrically.
- Explicit non-parametric representation:
  - x = x, y = f(x), z = g(x).
- Non-parametric implicit representation:
  - f(x, y, z) = 0, g(x, y, z) = 0.
- Intersection of two second degree surfaces yields a third degree curve.
- By letting z = t the parametric equation for that curve are

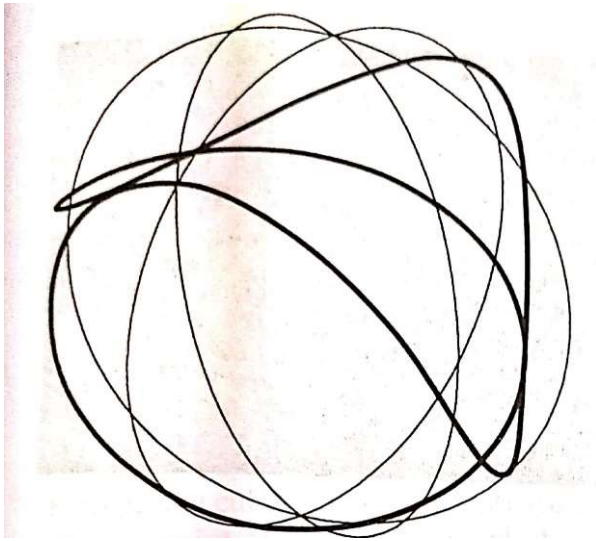
$$x = t^3, y = t^2, z = t$$

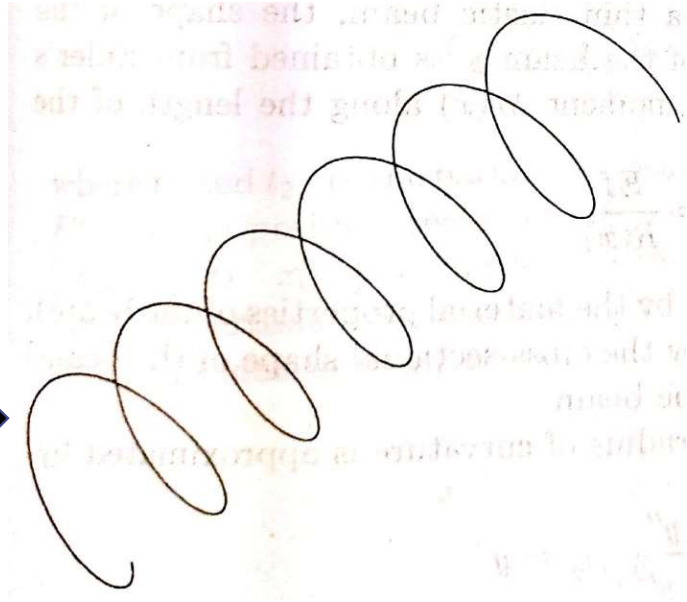


Representation of space curves (3D)

- A parametric space curve:
  - **x = x(t), y = f(t), z = g(t).**
- Ex: Curve on the seam of a baseball
  - $x = \lambda[a.\cos(\theta+\pi/4) - b.\cos 3(\theta+\pi/4)]$,
  - $y = \mu[a.\sin(\theta+\pi/4) - b.\sin 3(\theta+\pi/4)]$,
  - $z = c.\sin(2\theta)$.

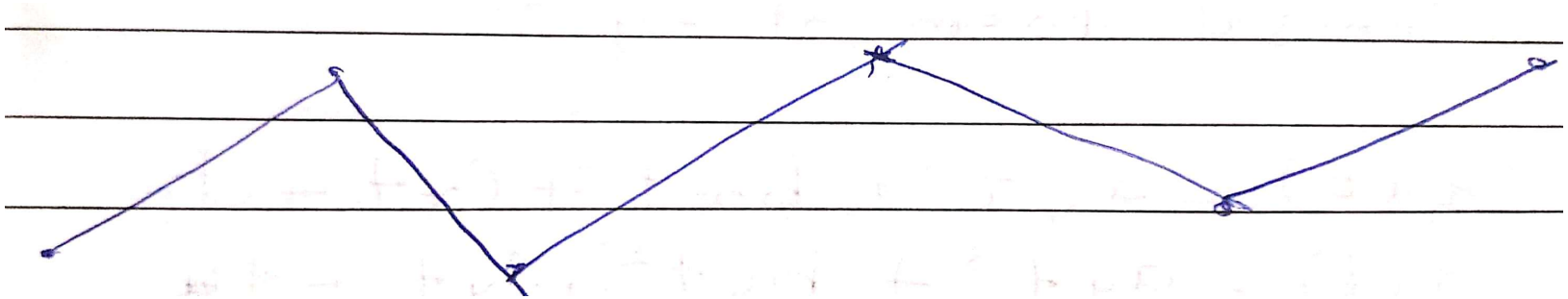
  Where
  - $\lambda = 1 + d.\sin(2\theta) = 1 + d(z/c)$,
  - $\mu = 1 - d.\sin(2\theta) = 1 - d(z/c)$;
  - $\theta = 2\pi t, 0 \le t \le 1.0$.
- Helix
  - $x = r.\cos(t), y = r.\sin(t), z = bt$;
  - $b \neq 0, -\infty < t < \infty$



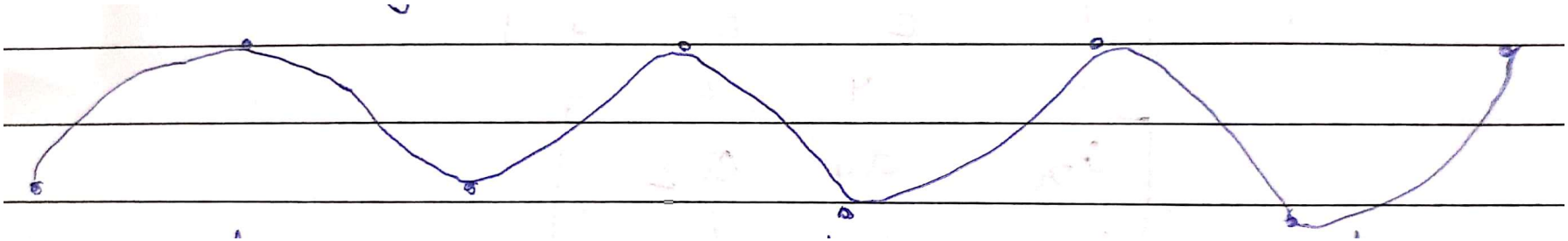


Representation of space curves (3D)

- Parametric space curves in 3D are of 2 types:
  - Piecewise linear curve
    - Represented with linear equation such as $y = mx + b$



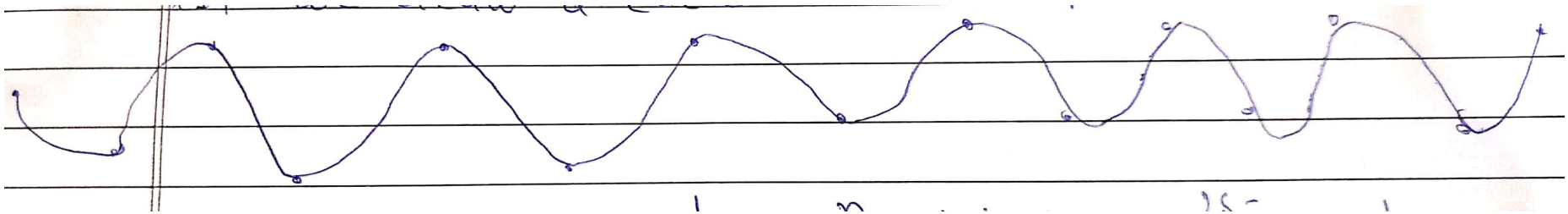
  - Piecewise polynomial curve
    - Represented with polynomial equation such as $x^3 + x^2 + 8$ with degree 3, means 4 points are used to draw a curve.



We are going to consider polynomial form of curve.

Representation of space curves (3D)

- If we draw a curve with 16 points such as



- Then polynomial equation will be x^{15} which becomes very complex for execution.
- So we here consider a curve of 4 points with polynomial of degree 3 (x^3) - **PARAMETRIC CUBIC CURVES**

PARAMETRIC CUBIC CURVES

- x = x(t), y = y(t), z = z(t).
- Now this parametric form we are going to represent as:

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

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$

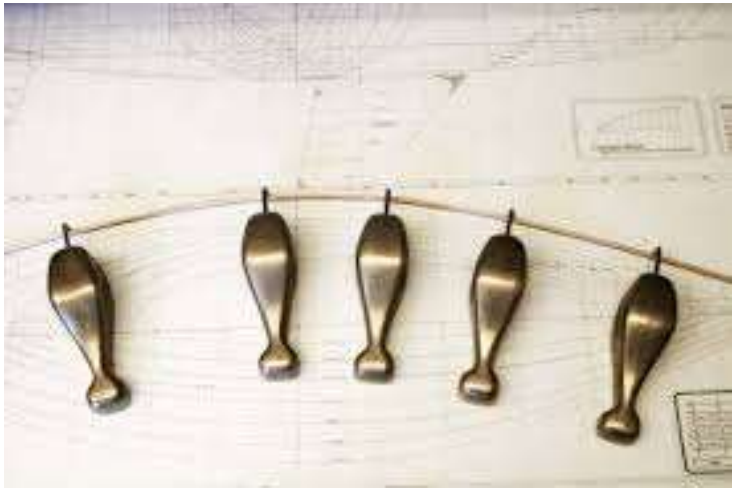
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z .$$

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.C,$$

$$\text{where, } T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \text{ and } C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Cubic Spline

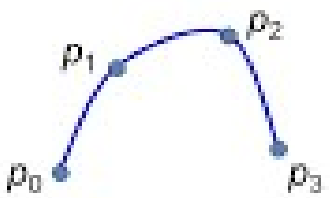
- In drafting terminology, mainly in shipbuilding a spline is a flexible strip used to produce a smooth curve through a designated set of points.
- Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.
- A curve drawn in this manner in known as spline curve.



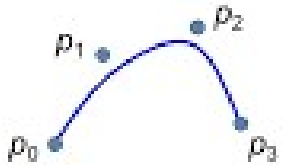
- We specify a spline curve by giving a set of coordinate positions, called **control points**, which indicates the general shape of the curve.

Cubic Spline – Interpolation and Approximation Spline

- These, control points are then fitted with piecewise continuous parametric polynomial functions in one of two ways:
  - **Interpolation:** When polynomial sections are fitted so that the curve passes through each control point.



  - interpolation curves are commonly used to digitize drawings or to specify animation paths
  - **Approximation:** when the polynomials are fitted to the general control-point path without necessarily passing through all control points.



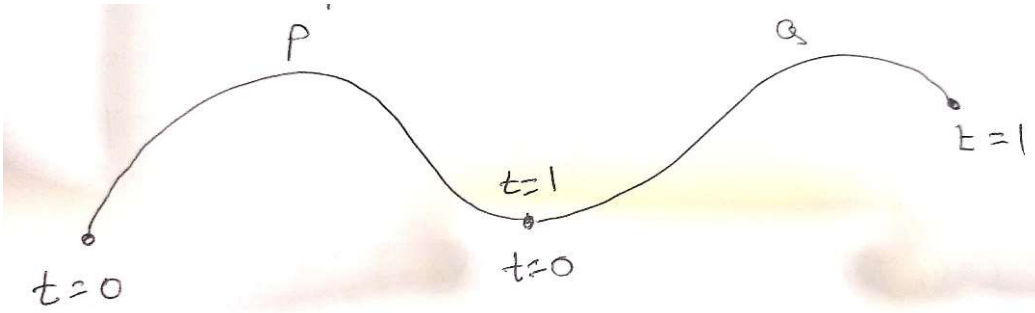
  - Approximation curves are primarily used as design tools to structure object surfaces such as table, chairs.

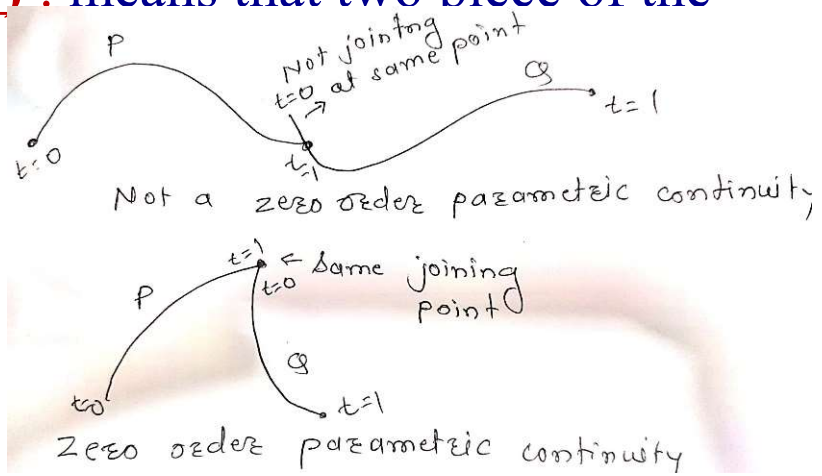
Cubic Spline - Interpolation

- To ensure a smooth transition from one section of a piecewise polynomial curve to the next, we can impose various continuity conditions at the connection points:
  - Parametric Continuity Conditions (C)
  - Geometric Continuity Conditions (G)

Cubic Spline - Interpolation

- **Parametric Continuity Conditions :**
  - **Zero-order parametric continuity (C^0)** : means that two piece of the curves are joined or meet at same point.

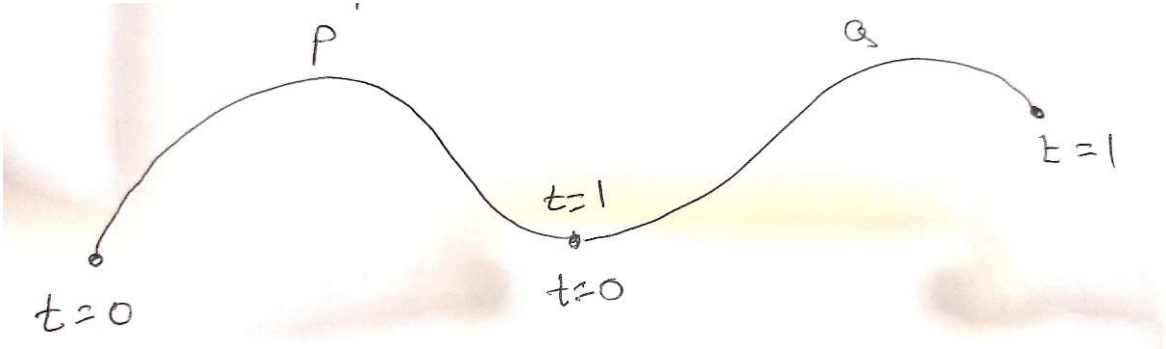




  - P (t = 1) = Q (t = 0)
  - **First-order parametric continuity (C^1)** : means that the first parametric derivatives (tangent lines) of the coordinate functions for two successive curve sections are equal at their joining point.
  - P' (t = 1) = Q' (t = 0)
  - **Second-order parametric continuity (C^2)** : means that both the first and second parametric derivatives of the two curve sections are equal at their joining point.
  - P'' (t = 1) = Q'' (t = 0)

Cubic Spline - Interpolation

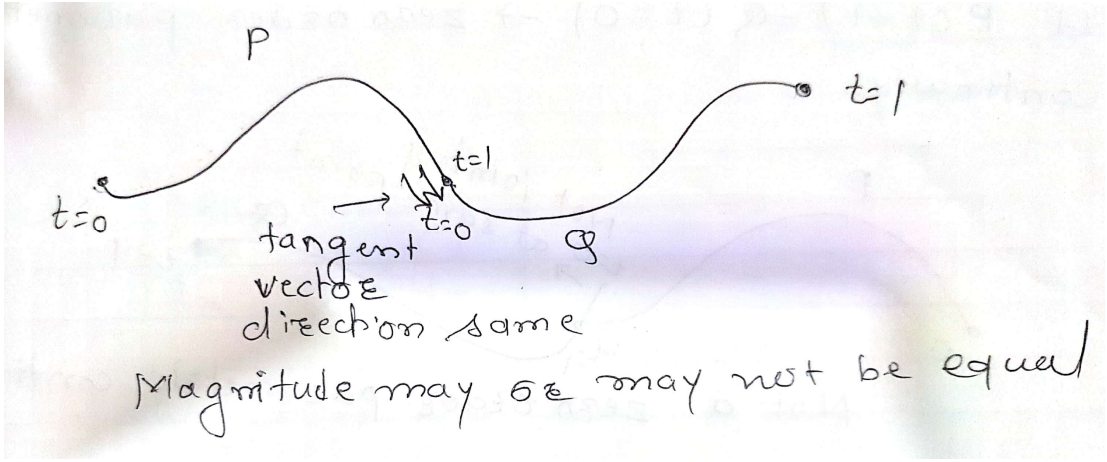
- **<u>Geometric Continuity Conditions</u>:** It deals with shape of curve. Here we require parametric derivatives of the two sections to be proportional to each other at their common boundary instead of equal to each other.
  - **<u>Zero-order geometric continuity (G^0) :</u>** is same as zero-order parametric continuity. That is, the two curves sections must have the same coordinate position at the boundary point.



  - P (t = 1) = Q (t = 0)

Cubic Spline - Interpolation

- **Geometric Continuity Conditions:**
  - **First-order geometric continuity (G^1) :** means that the parametric first derivatives are proportional at the intersection of two successive sections.
  - It may or may not be equal.
  - P' (x, y ,z) = Q' (k * x , k * y, k * z)
  - $C^1 \Rightarrow G^1$ but $G^1 \neq C^1$



  - Tangent vector direction is same (proportional), but not necessarily its magnitude, will be the same for two successive curve sections at their joining point under G^1 continuity.

Cubic Spline - Interpolation

- **Geometric Continuity Conditions:**
  - **Second-order geometric continuity (G^2) :** means that both the first and second parametric derivatives of the two curve sections are proportional at their boundary point and tangent vector direction is same.
  - Magnitude may or may not be same.
  - P'' (x, y ,z) = Q'' (k * x , k * y, k * z)
  - $C^2 \Rightarrow G^2$ but $G^2 \neq C^2$

Cubic Spline - Interpolation

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

- In above example, three control points fitted with two curve sections joined with (a) parametric continuity and (b) geometric continuity, where the tangent vector of **curve C_3** at point **p_1,** has a greater magnitude than the tangent vector of curve **C_1** at **p_1,.**
- With geometric continuity, the curve is pulled toward the section with the greater tangent vector.
- Video

Cubic Spline

- 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.

Cubic Spline

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

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \qquad (1)$$

- we can obtain the matrix that characterizes this spline curve by first rewriting Eq. (1) as the matrix product

  - $X(t) = [t^3 \quad t^2 \quad t^1 \quad 1] \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$

  - $= T \cdot C$

Cubic Spline

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

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$

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

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.C,$$

$$\text{where, } T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \text{ and } C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Cubic Spline

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.M.G,$$

$$\text{where, } T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix},$$

$$M = \begin{bmatrix} m_{ij} \end{bmatrix}_{4x4} \text{ and } G = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 \end{bmatrix}^T$$

M is a 4x4 *basis matrix* and G is a four element column vector of geometric constants, called the *geometric vector*.

The curve is a weighted sum of the elements of the geometry matrix.

The weights are each cubic polynomials of t, and are called the *blending functions*:

B = T.M.

Cubic Spline

- The equation for a single parametric cubic spline segment is: Use boundary conditions to evaluate the constant coefficients B_i.

$$P(t) = B_1 + B_2 t + B_3 t^2 + B_4 t^3 \qquad t_1 \le t \le t_2 \qquad (1)$$

  - Where t_1 and t_2 - beginning and end of segment

$$P(t) = \sum_{i=1}^{4} B_i t^{i-1}; \; t_1 \le t \le t_2.$$

P(t) is the position vector of any point on the cubic spline segment.

P(t) = [x(t), y(t), z(t)]	Cartesian
or [r(t), θ(t), z(t)]	Cylindrical
or [r(t), θ(t), ϕ(t)]	Spherical

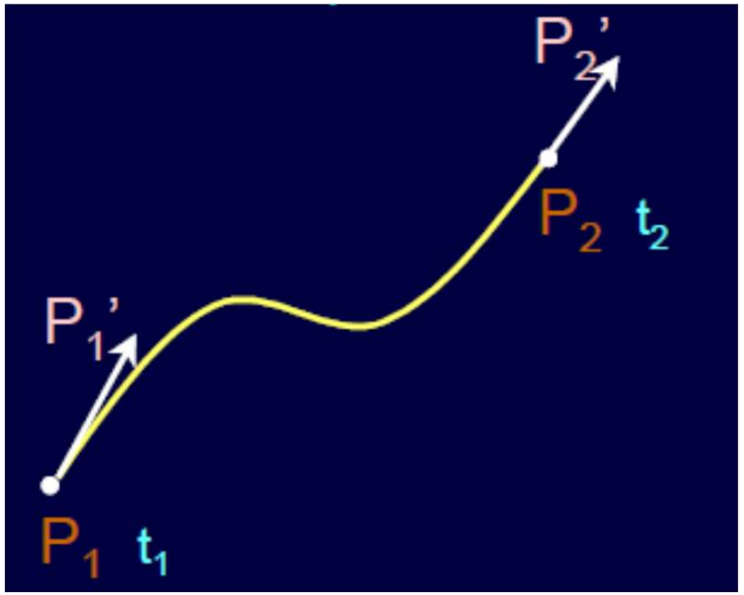
Cubic Spline

$$\left.\begin{aligned} x(t) &= \sum_{i=1}^{4} B_{ix} t^{i-1} \\ y(t) &= \sum_{i=1}^{4} B_{iy} t^{i-1} \\ z(t) &= \sum_{i=1}^{4} B_{iz} t^{i-1} \end{aligned}\right| \quad t_1 \le t \le t_2.$$

- The constant coefficients B_i are determined by specifying four boundary conditions for the spline segment.

Cubic Spline

- Let P_1 and P_2 be the position vectors at the ends of the spline segment.
- Also let P'_1 and P'_2, the derivatives with respect to t, be the tangent vectors at the ends of the spline segment.



- $P'(t) = [x'(t) \;\; y'(t) \;\; z'(t)]$

$$\mathbf{P'(t)} = \sum_{i=1}^{4} \boldsymbol{(i-1)B_i t^{i-2}} \qquad t_1 \le t \le t_2.$$

$$P'(t) = \mathbf{B_2 + 2B_3 t + 3B_4 t^2}$$

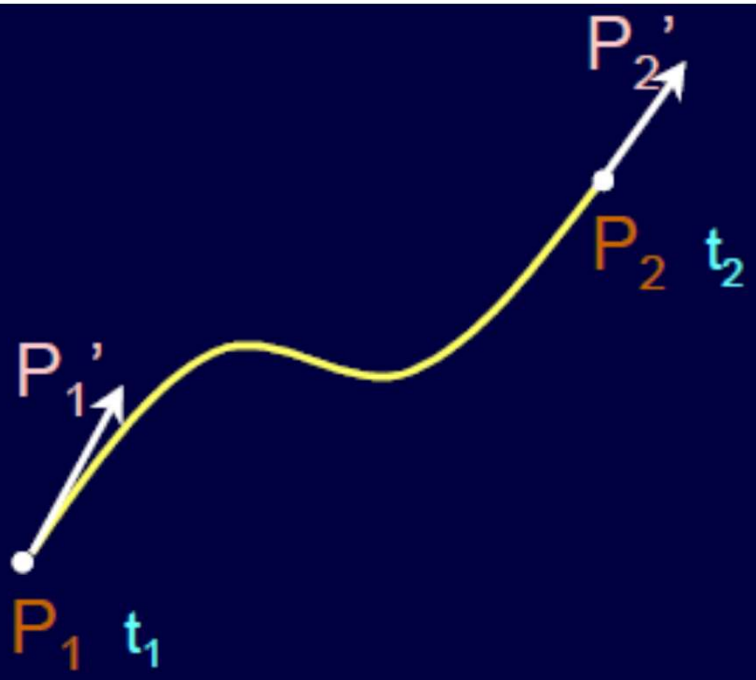
- Let, t_1=0: and applying the four boundary conditions.

$$P(0) = P_1; \qquad P(t_2) = P_2.$$

$$P'(0) = P'_1; \qquad P'(t_2) = P'_2.$$

Cubic Spline

- Then four equations for the unknown B_i's are:



$$P(0) = B_1 = P_1$$

$$P(t_2) = \sum_{i=1}^{4} B_i t^{i-1}\Big|_{t=t_2}$$

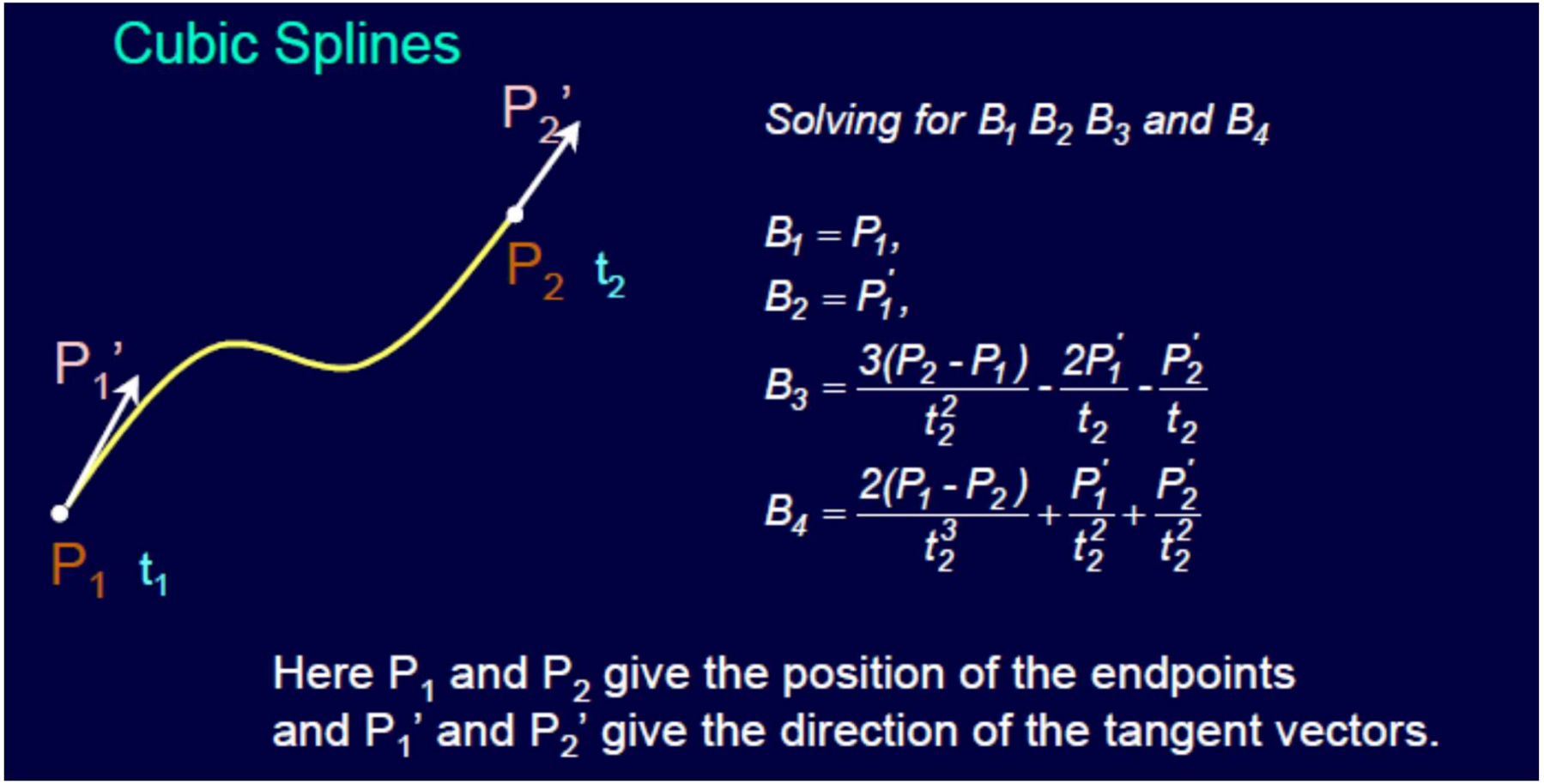
$$= B_1 + B_2 t_2^{\,1} + B_3 t_2^{\,2} + B_4 t_2^{\,3} = P_2$$

$$P'(0) = \sum_{i=1}^{4} B_i (i-1) t^{i-2}\Big|_{t=0} = B_2 = P_1'$$

$$P'(t_2) = \sum_{i=1}^{4} B_i (i-1) t^{i-2}\Big|_{t=t_2}$$

$$= B_2 + 2B_3 t_2^{\,1} + 3B_4 t_2^{\,2} = P_2'$$

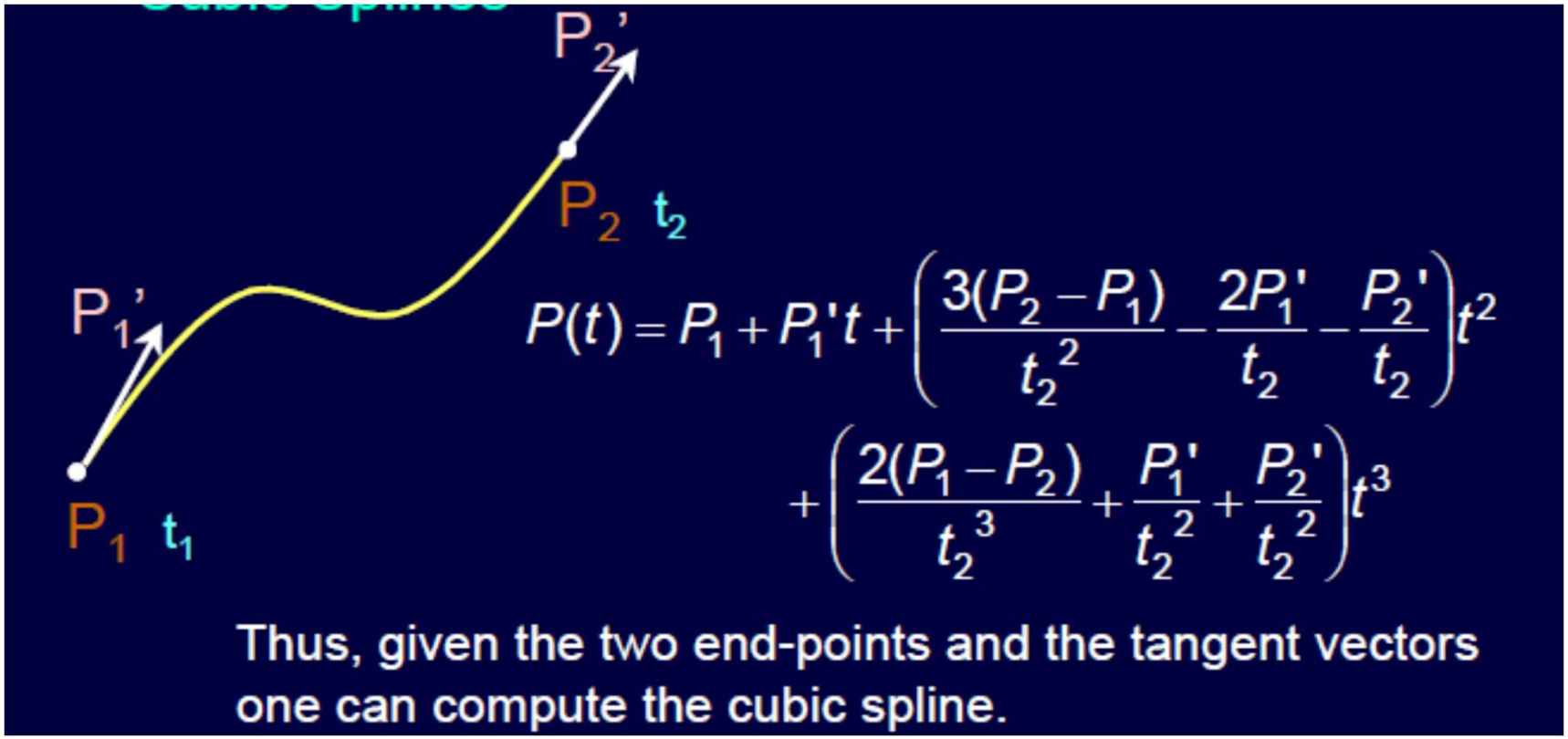
Cubic Spline



- These values of B1, B2, B3 and B4 determine the cubic spline segment.

Cubic Spline

- Substituting these values in equation (1) yields the equation for the single cubic spline segment.

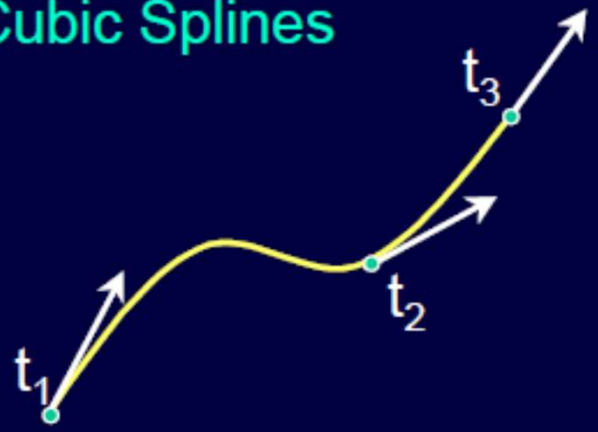


- For piece-wise continuity - for complex curves, two curves, two or more curve segments are joined together.
- In that case, use second derivative $P_2''(t)$ at end-points (joints).

Curves

Parametric Curves

Cubic Splines



Joining of Segments

2 SEGMENTS: P_1 P_2 P_3 (Points)

P_1' P_2' P_3' (Tangents)

P_2' is determined through the continuity condition

Curves

Parametric Curves

Cubic Splines

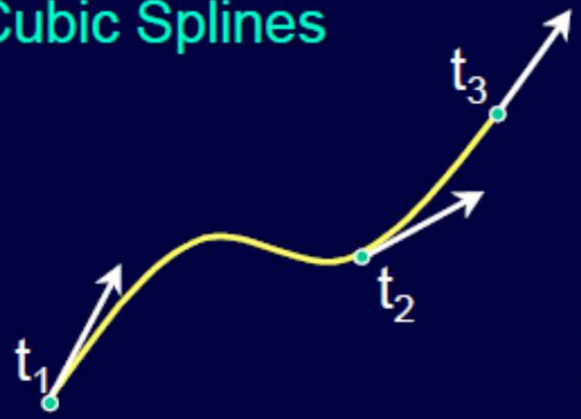
Piecewise Spline of degree ***k*** has continuity of order ***(k-1)*** at the internal joints.

Thus Cubic Splines have second order continuity i.e. P_2"(t) is continuous over the joint

Curves

Parametric Curves

Cubic Splines



$$P''(t) = \sum_{i=1}^{4} (i-1)(i-2)B_i t^{i-3} \qquad t_1 \le t \le t_2$$

at $t = t_2$

First segment

$$P'' = 6B_4 t_2 + 2B_3$$

Second segment

$$P'' = 2B_3$$

$$So, (6B_4 t_2 + 2B_3)_{seg1} = (2B_3)_{seg2}$$

Substitute the expressions for B_4 and B_3

Curves

Parametric Curves

Cubic Splines

$$t_3P_1'+2(t_3+t_2)P_2'+t_2P_3'=\frac{3}{t_2t_3}\left(t_2^2(P_3-P_2)+t_3^2(P_2-P_1)\right)$$

$$\begin{bmatrix} t_3 & 2(t_3+t_2) & t_2 \end{bmatrix}\begin{bmatrix} P_1' \\ P_2' \\ P_3' \end{bmatrix}=\frac{3}{t_2t_3}\left(t_2^2(P_3-P_2)+t_3^2(P_2-P_1)\right)$$

Curves

Parametric Curves

Cubic Splines

In general, for the ***k***th and ***(k+1)***th segment (1 ***k*** n-2)

$$\begin{bmatrix} t_{k+2} & 2(t_{k+1}+t_{k+2}) & t_{k+1} \end{bmatrix} \begin{bmatrix} P_k' \\ P_{k+1}' \\ P_{k+2}' \end{bmatrix}$$

$$= \frac{3}{t_{k+1}t_{k+2}} \left(t_{k+1}^2 (P_{k+2} - P_{k+1}) + t_{k+2}^2 (P_{k+1} - P_k) \right)$$

Set of n-2 equations form a linear system for the tangent vectors P_k'

Curves

Parametric Curves

Cubic Splines

$$\begin{bmatrix} t_3 & 2(t_2+t_3) & t_2 & 0 & \cdots \\ 0 & t_4 & 2(t_3+t_4) & t_3 & \\ \vdots & & \ddots & & \vdots \\ & \cdots\cdots & t_n & 2(t_n+t_{n-1}) & t_{n-1} \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ P_n' \end{bmatrix}$$

$$= \begin{bmatrix} \frac{3}{t_2 t_3}\left(t_2^2(P_3-P_2)+t_3^2(P_2-P_1)\right) \\ \frac{3}{t_3 t_4}\left(t_3^2(P_4-P_3)+t_4^2(P_3-P_2)\right) \\ \vdots \\ \frac{3}{t_{n-1} t_n}\left(t_{n-1}^2(P_n-P_{n-1})+t_n^2(P_{n-1}-P_{n-2})\right) \end{bmatrix}$$

Curves

Parametric Curves

Cubic Splines

$$\begin{bmatrix} 1 & 0 & \cdots & & \\ t_3 & 2(t_2+t_3) & t_2 & & \\ \vdots & t_4 & 2(t_3+t_4) & t_3 & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ & & t_n & 2(t_n+t_{n-1}) & t_{n-1} \\ & & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \\ \\ P_{n-1}' \\ P_n' \end{bmatrix} = \begin{bmatrix} P_1' \\ \frac{3}{t_2 t_3}\left(t_2^2(P_3-P_2)+t_3^2(P_2-P_1)\right) \\ \vdots \\ \frac{3}{t_{n-1}t_n}\left(t_{n-1}^2(P_n-P_{n-1})+t_n^2(P_{n-1}-P_{n-2})\right) \\ P_n' \end{bmatrix}$$

Curves

Parametric Curves

Cubic Splines

Solving for B_1 B_2 B_3 and B_4

$$B_{1k} = P_k,$$

$$B_{2k} = P_k',$$

$$B_{3k} = \frac{3(P_{k+1} - P_k)}{t_{k+1}^2} - \frac{2P_k'}{t_{k+1}} - \frac{P_{k+1}'}{t_{k+1}}$$

$$B_{4k} = \frac{2(P_k - P_{k+1})}{t_{k+1}^3} + \frac{P_k'}{t_{k+1}^2} + \frac{P_{k+1}'}{t_{k+1}^2}$$

Curves

Parametric Curves

Cubic Splines

$$\begin{bmatrix} B_{1k} \\ B_{2k} \\ B_{3k} \\ B_{4k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/t_{k+1}^2 & -2/t_{k+1} & 3/t_{k+1}^2 & -1/t_{k+1} \\ 2/t_{k+1}^3 & 1/t_{k+1}^2 & -2/t_{k+1}^3 & 1/t_{k+1}^2 \end{bmatrix} \begin{bmatrix} P_k \\ P_k' \\ P_{k+1} \\ P_{k+1}' \end{bmatrix}$$

$$P_k(t) = \sum_{i=1}^{4} B_{ik} t^{i-1} \qquad \begin{matrix} 0 \le t \le t_{k+1} \\ 1 \le k \le n-1 \end{matrix}$$

$$= \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} B_{1k} & B_{2k} & B_{3k} & B_{4k} \end{bmatrix}^T$$

Curves

Parametric Curves

Cubic Splines

$$P_k(t)$$

$$= \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/t_{k+1}^2 & -2/t_{k+1} & 3/t_{k+1}^2 & -1/t_{k+1} \\ 2/t_{k+1}^3 & 1/t_{k+1}^2 & -2/t_{k+1}^3 & 1/t_{k+1}^2 \end{bmatrix} \begin{bmatrix} P_k \\ P_k' \\ P_{k+1} \\ P_{k+1}' \end{bmatrix}$$

$$= \begin{bmatrix} (1 - 3t^2/t_{k+1}^2 + 2t^3/t_{k+1}^3) & (t - 2t^2/t_{k+1} + t^3/t_{k+1}^3) \\ (3t^2/t_{k+1}^2 - 2t^3/t_{k+1}^3) & (t^3/t_{k+1}^3 - t^2/t_{k+1}) \end{bmatrix} \begin{bmatrix} P_k & P_k' & P_{k+1} & P_{k+1}' \end{bmatrix}^T$$

Curves

Parametric Curves

Cubic Splines

Substituting $u=t/t_{k+1}$ rearranging

$$P_k(u) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)] \; [P_k \quad P_{k+1} \quad P_k' \quad P_{k+1}']^T$$

$$0 \le u \le 1$$

$$1 \le k \le n-1$$

$$F_1(u) = 2u^3 - 3u^2 + 1$$
$$F_2(u) = -2u^3 + 3u^2$$
$$F_3(u) = u(u^2 - 2u + 1)t_{k+1}$$
$$F_4(u) = u(u^2 - u)t_{k+1}$$

F_1, F_2, F_3, F_4 are called the ***Blending Functions***

Curves

Parametric Curves

Cubic Splines

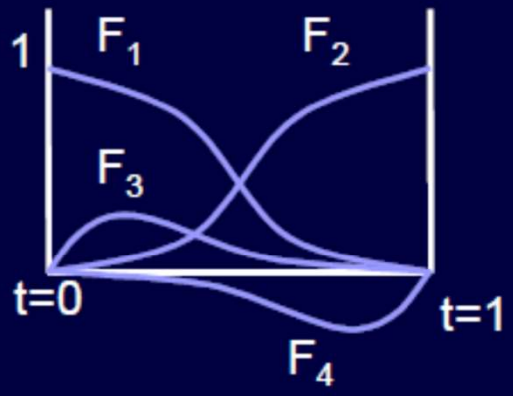
$$P_k(u)=[F][G]$$

Where F is the Blending function matrix and G Is the geometric information.

Curves

Parametric Curves

Cubic Splines



- $F_1(0)=1, F_2(0)=0, F_3(0)=0, F_4(0)=0$
  curve passes P_1
- $F_1(1)=0, F_2(1)=1, F_3(1)=0, F_4(1)=0$
  curve passes P_2
- $F_2=1-F_1, F_4=1-F_3$
- Relative magnitudes of $F_1, F_2 > F_3, F_4$

Curves

Parametric Curves

Cubic Splines

Piecewise Cubic Splines are determined by position vectors, tangent vectors and parameter value t_k .

The value of t_k can be chosen using either Chord Length parameterization or Uniform Parameterization.

If t_k=1 for all ***k*** then the Spline is called Normalized Spline.

Curves

Parametric Curves

Cubic Splines

Normalized Cubic Splines: $t_k=1$ for all segments

The blending functions thus become

$$F_1(t) = 2t^3-3t^2+1,\ F_2(t) = -2t^3+3t^2$$

$$F_3(t) = t^3-2t^2+t\ ,\ F_4(t) = t^3-t^2$$

These are called *Hermite Polynomial Blending* functions

$$\begin{bmatrix} F_1(t) \\ F_2(t) \\ F_3(t) \\ F_4(t) \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Curves

Parametric Curves

Cubic Splines

The tridiagonal system for getting P' becomes

$$\begin{bmatrix} 1 & 0 & \cdots & & \\ 1 & 4 & 1 & & \\ \vdots & 1 & 4 & 1 & \vdots \\ \vdots & & \ddots & & \vdots \\ & & 1 & 4 & 1 \\ & & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ \vdots \\ P_{n-1}' \\ P_n' \end{bmatrix} = \begin{bmatrix} P_1' \\ 3((P_3 - P_2) + (P_2 - P_1)) \\ \vdots \\ \\ 3((P_n - P_{n-1}) + (P_{n-1} - P_{n-2})) \\ P_n' \end{bmatrix}$$

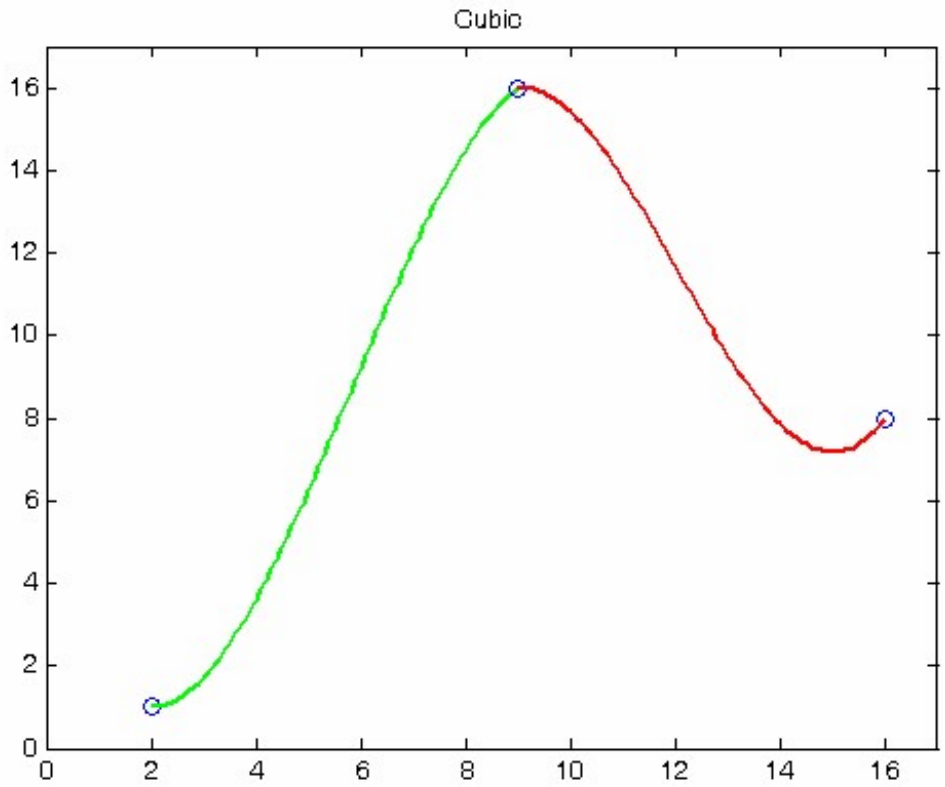
Curves

Parametric Curves

Cubic Splines

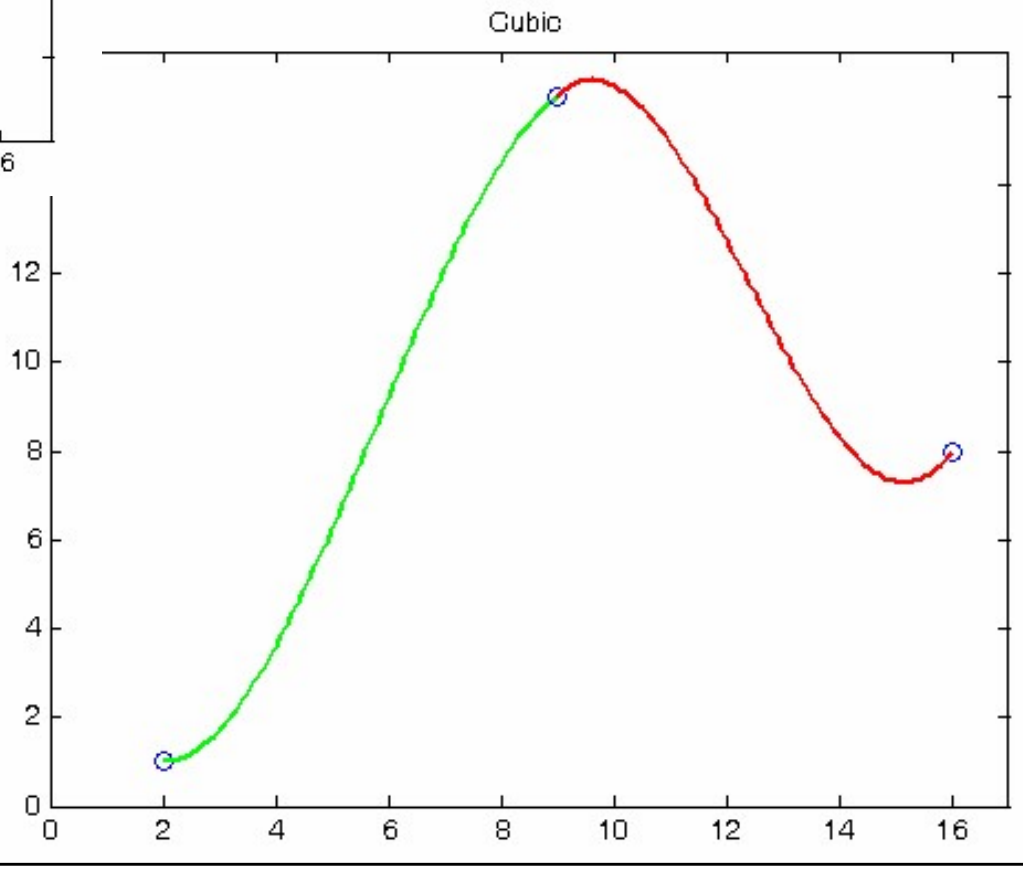
End Conditions as:

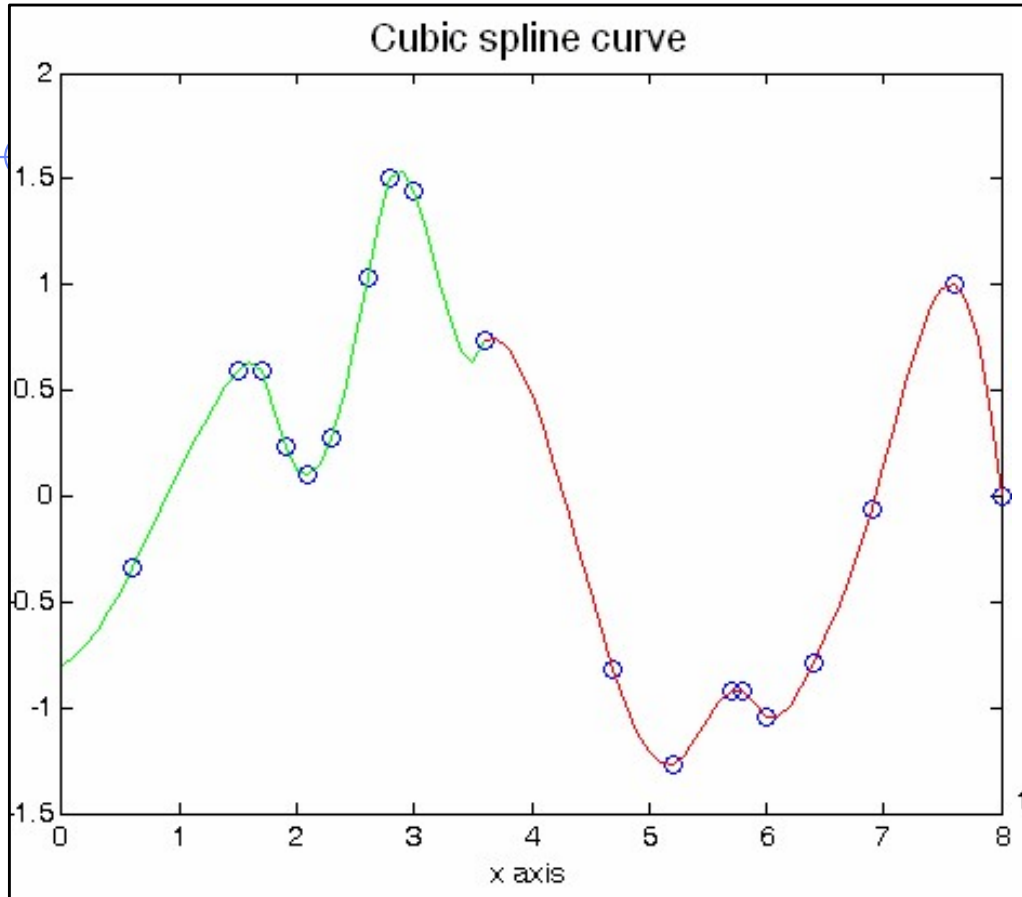
- Clamped: P_1',P_n' known
- Relaxed/Natural: $P''(0) = 0$
  $P''(t_n)=0$
- Cyclic: $P_1'(0)=P_n'(t_n)$
  $P_1''(0)=P_n''(t_n)$
- Anticyclic: $P_1'(0) = -P_n'(t_n)$
  $P_1''(0) = -P_n''(t_n)$



No use of 2nd derivative smoothing

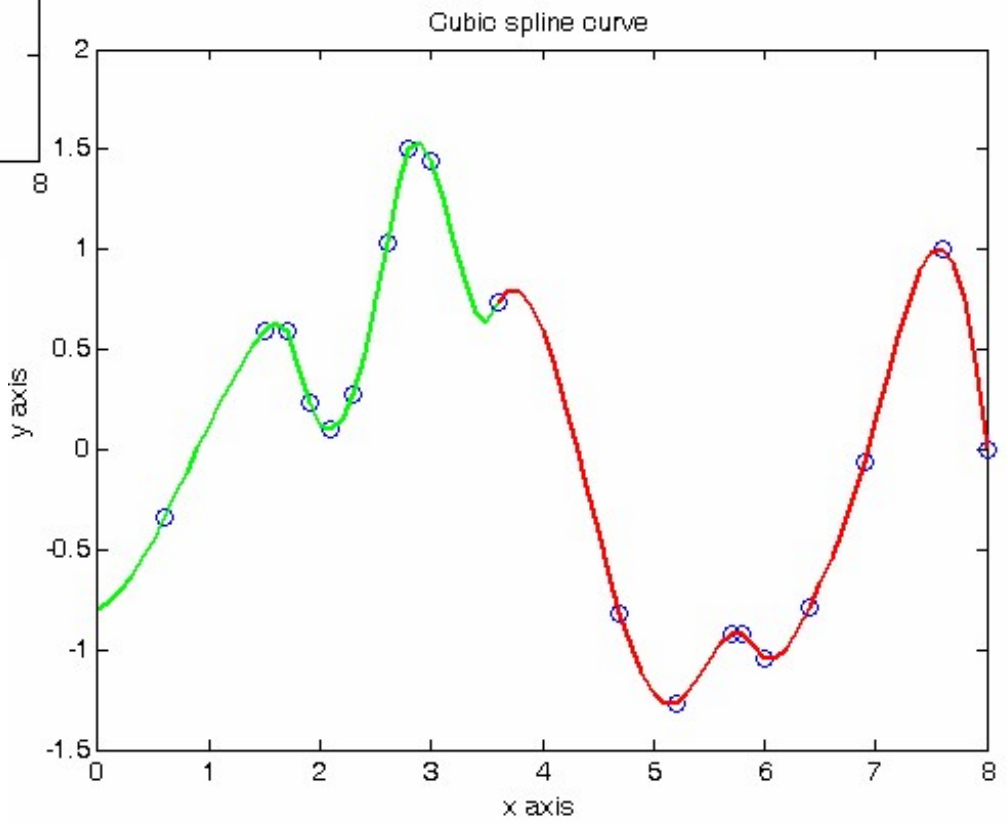
Two piecewise cubic spline segments using 2nd derivative smoothing





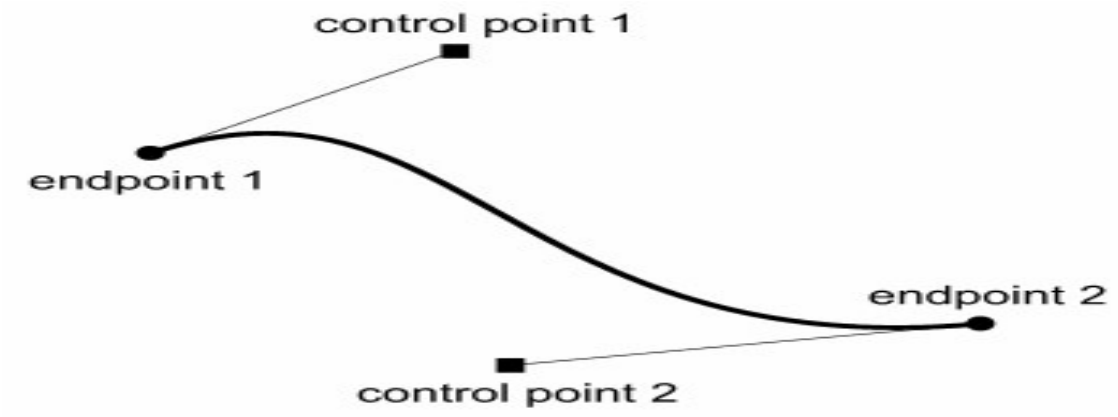
No use of 2nd derivative smoothing

Examples of spline interpolation Using 2nd derivative smoothing



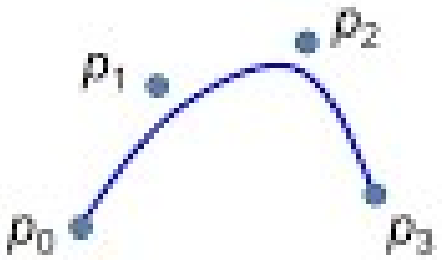
BEZIER CURVES

- **Definition:** A Bezier curve is a mathematically defined curve used in two dimensional graphic applications.
- The curve is defined by four points: the initial position and the terminating position (which are called "anchors") and two separate middle points (which are called "handles").
- The shape of a Bezier curve can be altered by moving the handles.
- The mathematical method for drawing curves was created by Pierre Bezier in the late 1960's for the manufacturing of automobiles at Renault.

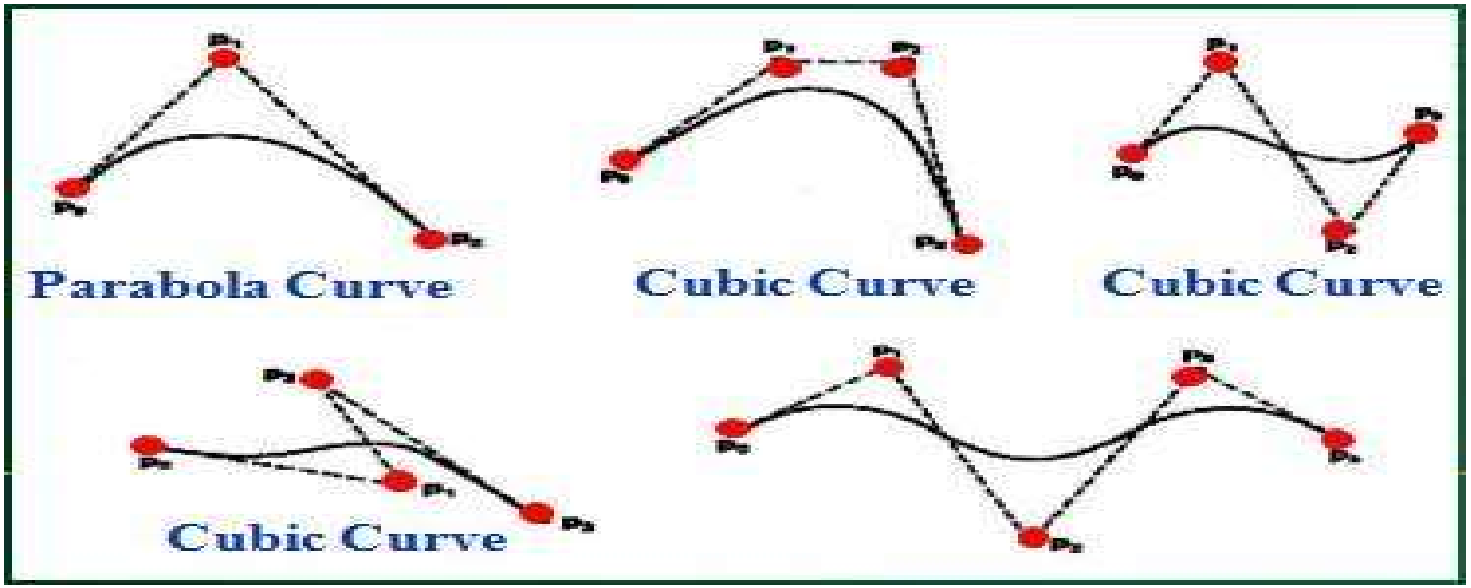


BEZIER CURVES - Properties

- Properties of Bezier Curve
  - Basis functions are real.
  - Bezier curve is an approximation spline curve.



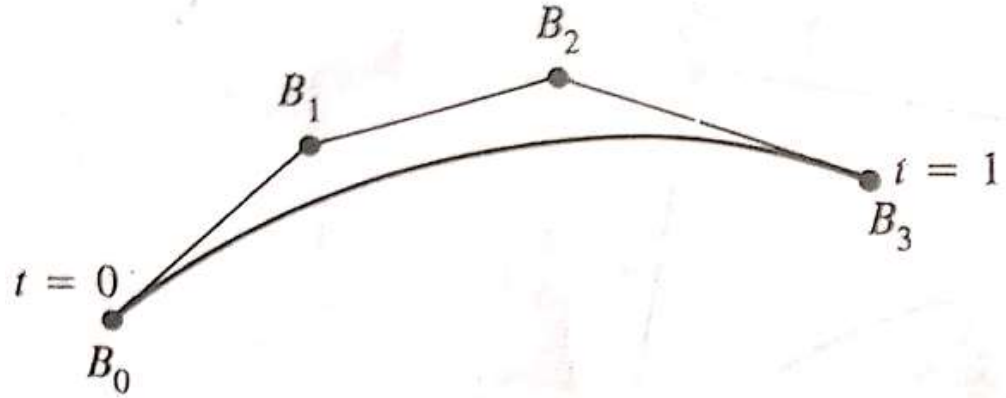
  - Degree of polynomial is one less than the number of control points, means degree is n if there are n + 1 control points.



BEZIER CURVES - Properties

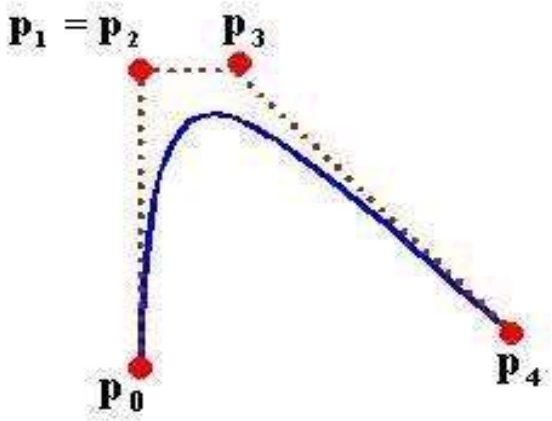
- Curve generally follows the shape of the defining polygon
- First and last points on the curve are coincident with the first and last points of the polygon
- The curve passes though the first and the last control point.

- The curve is contained within the convex hull of the defining polygon

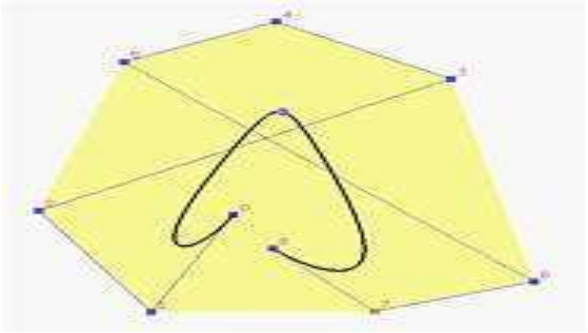


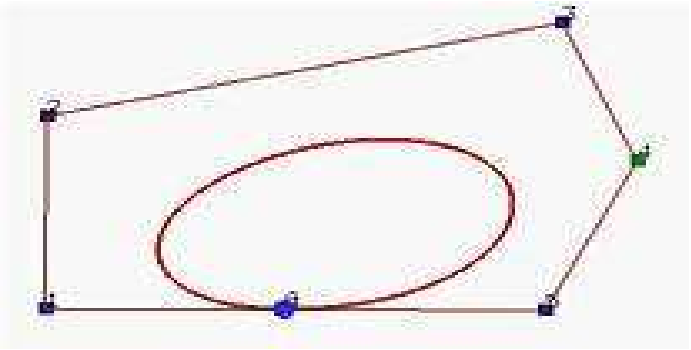
BEZIER CURVES - Properties

- Multiple control points at a single coordinate position gives more weight to that position.



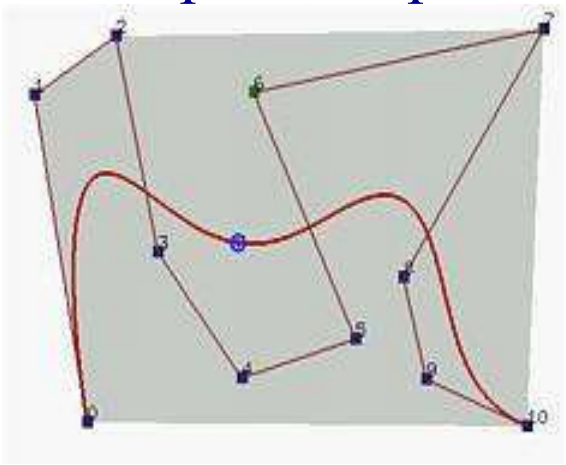
- Closed Bezier curves are generated by specifying the first and the last control points at the same position.

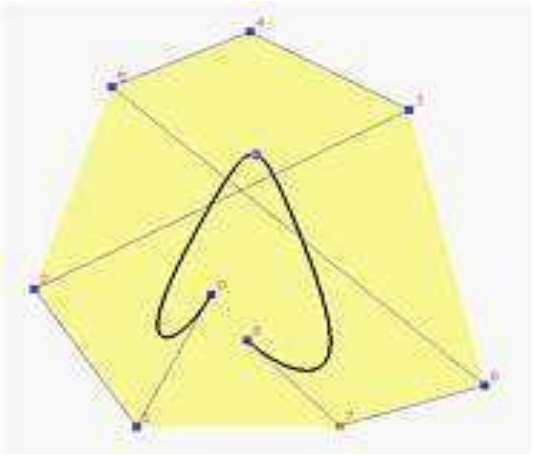




BEZIER CURVES - Properties

- Bezier curves are tangent to their first and last edges of control polyline. Tangent vectors at the ends of the curve have the same directions as the respective spans.





BEZIER CURVES

- A few typical examples of cubic polynomials (four point) for Bezier

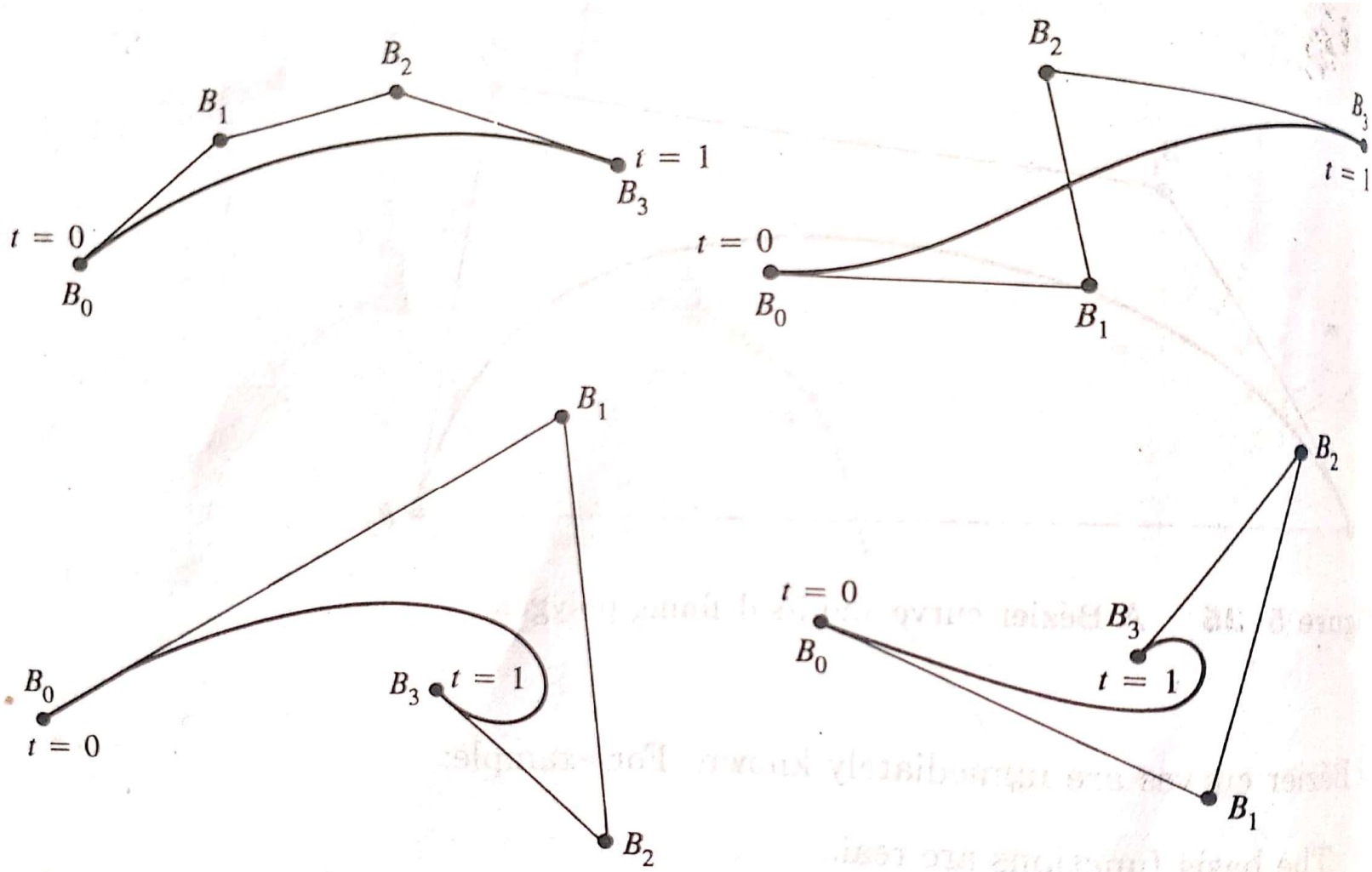


Figure 5–26 Bézier polygons for cubics.

BEZIER CURVES

- Mathematically a parametric Bezier curve is defined by

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t);\ 0 \le t \le 1 \qquad (1)$$

- Where B_i is position vector, $0 \le i \le n$
- And the Bezier or Bernstein basis or blending function is

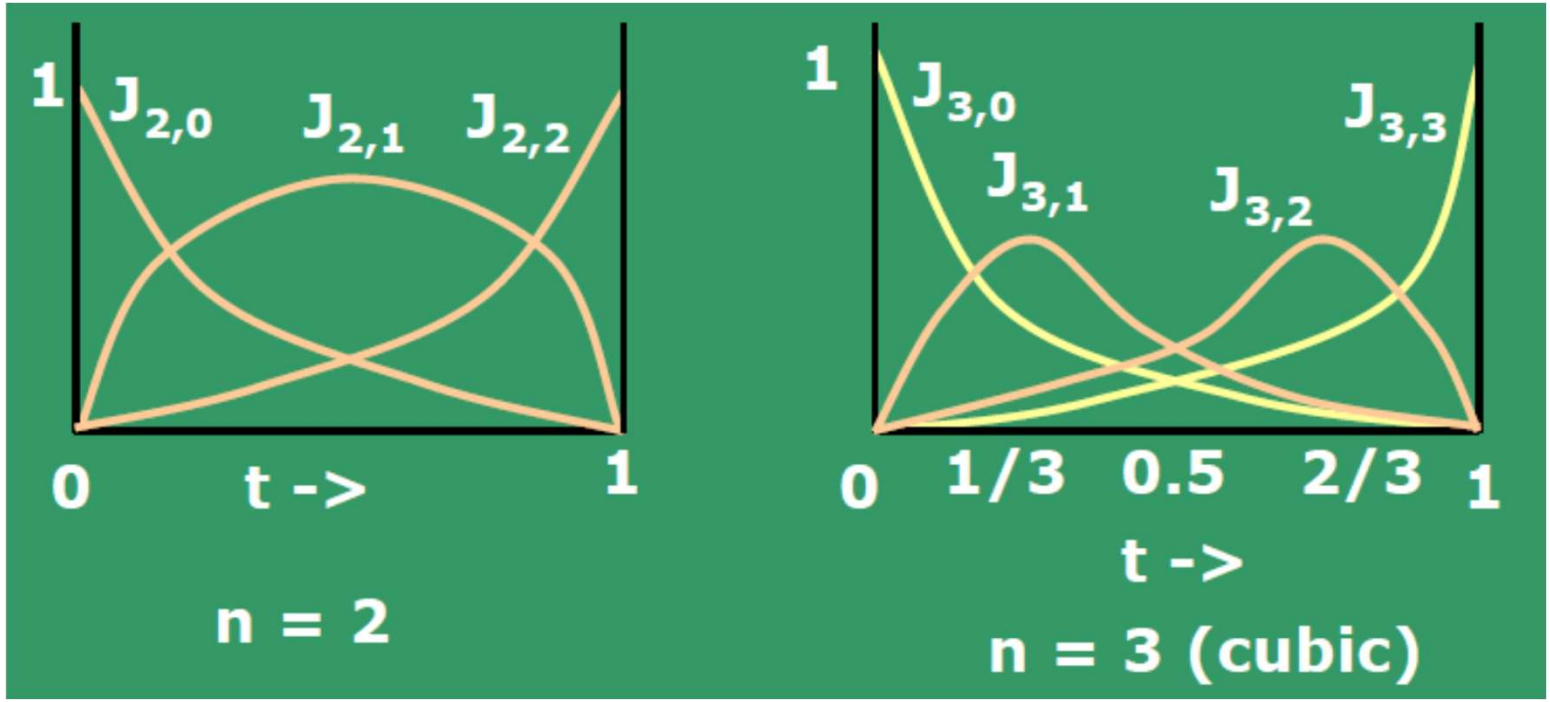
$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}; \qquad (2)$$

$$n_{ci} = \binom{n}{i} = \frac{n!}{i!(n-i)!} \quad \text{Binomial Coefficient} \qquad (3)$$

- $J_{n,\,i}\ (t)$ is the ith, nth-order Bernstein basis function. n is the degree of the defining Bernstein basis function (polynomial curve segment).
- This is one less than the number of points used in defining Bezier polygons.

BEZIER CURVES

- Below are some examples of BBF (Bezier /Bernstein blending functions:



BEZIER CURVES

Example :- Blending function for three polygon points, $n=2$ & $n+1=3$

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

$$J_{n,i}(t) = \frac{n!}{i!*(n-i)!} t^i (1-t)^{n-i}$$

$$P(t) = B_0 * J_{2,0}(t) + B_1 * J_{2,1}(t) + B_2 * J_{2,2}(t)$$

$i=0$ $\quad i=1$ $\quad i=2$

BEZIER CURVES

First we calculate Bezier or Bernstein basis or blending function

$$J_{2,0}(t) = \frac{2!}{0! * 2!} \cdot t^0 \cdot (1-t)^2 \qquad n=2,\ i=0$$

$$= 1 \cdot 1 \cdot (1-t)^2 = (1-t)^2$$

$$J_{2,1}(t) = \frac{2!}{1! * 1!} \cdot t^1 \cdot (1-t)^1 \qquad n=2,\ i=1$$

$$= 2 \cdot t \cdot (1-t)$$

$$= 2t(1-t) = 2t - 2t^2$$

$$J_{2,2}(t) = \frac{2!}{2! \cdot 0!} \cdot t^2 \cdot (1-t)^0 \qquad n=2,\ i=2$$

$$= 1 \cdot t^2 \cdot 1 = t^2$$

BEZIER CURVES

Now, addition of basis function -

$$\sum_{i=0}^{n} J_{n,i}(t) = J_{2,0}(t) + J_{2,1}(t) + J_{2,2}(t)$$

$$= (1-t)^2 + 2t - 2t^2 + t^2$$

$$= 1 - 2t + t^2 + 2t - 2t^2 + t^2$$

$$= 1 - 2t + 2t + 2t^2 - 2t^2$$

$$= 1$$

From this further, it can be shown that for any given value of the parameter t, the summation of the basis function is precisely one, ie.

$$\sum_{i=0}^{n} J_{n,i}(t) = 1 \qquad \text{---(4)}$$

BEZIER CURVES

Example to draw a Bezier curve:

Given $B_0[1\ 1]$, $B_1[2\ 3]$, $B_2[4\ 3]$ & $B_3[3\ 1]$ are the vertices of a Bezier polygon, determine 7 points on the Bezier curve.

-> The eqn for Bezier curve is

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

where $J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

and $\binom{n}{i} = \frac{n!}{i!(n-i)!}$

Here $n=3$, since there are 4 vertices.

$$P(t) = B_0 J_{3,0}(t) + B_1 J_{3,1}(t) + B_2 J_{3,2}(t) + B_3 J_{3,3}$$

$i=0$ $i=1$ $i=2$ $i=3$

BEZIER CURVES

First we calculate Bezier or Bernstein basis or blending function -

$$J_{3,0}(t) = \frac{3!}{0! \cdot 3!} \cdot t^0 \cdot (1-t)^3 \qquad n=3,\ i=0$$

$$= 1 \cdot 1 \cdot (1-t)^3 = (1-t)^3$$

$$J_{3,1}(t) = \frac{3!}{1!\ 2!} t^1 \cdot (1-t)^2 \qquad n=3,\ i=1$$

$$= \frac{6}{2} \cdot t \cdot (1-t)^2 = 3t(1-t)^2$$

$$J_{3,2}(t) = \frac{3!}{2! \cdot 1!} \cdot t^2 \cdot (1-t)^1 \qquad n=3,\ i=2$$

$$= \frac{6}{2} t^2 (1-t) = 3t^2(1-t)$$

$$J_{3,3}(t) = \frac{3!}{3!\ 0!} t^3 \cdot (1-t)^0$$

$$= t^3$$

BEZIER CURVES

Thus;

$$P(t) = B_0 J_{3,0}(t) + B_1 J_{3,1}(t) + B_2 J_{3,2}(t) + B_3 J_{3,3}(t)$$

$$= (1-t)^3 B_0 + 3t(1-t)^2 B_1 + 3t^2(1-t) B_2 + t^3 \cdot B_3$$

A table for 7 values of t is as below

Table 5-4 Coefficients for a Bézier curve

t	$J_{3,0}$	$J_{3,1}$	$J_{3,2}$	$J_{3,3}$
0	1	0	0	0
0.15	0.614	0.325	0.058	0.003
0.35	0.275	0.444	0.239	0.042
0.5	0.125	0.375	0.375	0.125
0.65	0.042	0.239	0.444	0.275
0.85	0.003	0.058	0.325	0.614
1	0	0	0	1

BEZIER CURVES

t	$J_{3,0}$	$J_{3,1}$	$J_{3,2}$	$J_{3,3}$
0	1	0	0	0
0.15				
0.35				
0.5				
0.65				
0.85				
1				

BEZIER CURVES

- $(x_0,y_0) = (1,1)$, $(x_1,y_1) = (2,3)$, $(x_2,y_2) = (4,3)$, $(x_3,y_3) = (3,1)$
- $x(0) = 1*1+2*0+4*0+3*0 = 1$
- $x(0.15) = 1*0.614 + 2*0.325 + 4*0.058 + 3*0.003 = 1.5$
- $x(0.35) = 1*0.275 + 2*0.444 + 4*0.239 + 3*0.042 = 2.245$
- $x(0.5) = 2.75$
- $x(0.65) = 3.121$
- $x(0.85) = 3.248$
- $x(1) = 3$

BEZIER CURVES

- $(x_0,y_0) = (1,1)$, $(x_1,y_1) = (2,3)$, $(x_2,y_2) = (4,3)$, $(x_3,y_3) = (3,1)$
- $y(0) = 1*1 + 3*0 + 3*0 + 1*0 = 1$
- $y(0.15) = 1*0.614 + 3*0.325 + 3*0.058 + 1*0.003 = 1.766$
- $y(0.35) = 2.366$
- $y(0.5) = 2.5$
- $y(0.65) = 2.366$
- $y(0.85) = 1.766$
- $y(1) = 1$

BEZIER CURVES

- P[0] = P[t = 0] = [1,1]
- P[1] = P[t = 0.15] = [1.5, 1.766]
- P[2] = P[t = 0.35] = [2.245, 2.366]
- P[3] = P[t = 0.5] = [2.75, 2.5]
- P[4] = P[t = 0.65] = [3.121, 2.366]
- P[5] = P[t = 0.85] = [3.248, 1.766]
- P[6] = P[t = 1] = [3, 1]

BEZIER CURVES

Now, we find the points on the curve x & y coordinates

$(x_0, y_0) = (1, 1)$ $(x_1, y_1) = (2, 3)$, $(x_2, y_2) = (4, 3)$,

$(x_4, y_4) = (3, 1)$

$x(0) = B_0$ i.e., $0 = 1$

$x(0.15) = 0.614 * 1 + 0.325 * 2 + 0.058 * 4 + 0.003 * 3$

$= 0.614 + 0.65 + 0.232 + 0.009 = 1.5$

$x(0.35) = 0.275 * 1 + 0.444 * 2 + 0.239 * 4 + 0.042 * 3$

$= 0.275 + 0.888 + 0.956 + 0.126 = 2.245$

$x(0.5) = 0.125 * 1 + 0.375 * 2 + 0.375 * 4 + 0.125 * 3$

$= 0.125 + 0.75 + 1.5 + 0.375 = 2.75$

$x(0.65) = 0.042 * 1 + 0.239 * 2 + 0.444 * 4 + 0.278 * 3$

$= 0.042 + 0.478 + 1.776 + 0.828 = 3.121$

$x(0.85) = 0.003 * 1 + 0.058 * 2 + 0.325 * 4 + 0.614 * 3$

$= 0.003 + 0.116 + 1.3 + 1.842 = 3.248$

$x(1) = B_3 = 3$

BEZIER CURVES

$y(0) = B_0 = 1$

$y(0.15) = 0.614 * 1 + 0.325 * 3 + 0.058 * 3 + 0.003 * 1$

$= 0.614 + 0.975 + 0.174 + 0.003 = 1.766$

$y(0.35) = 0.275 * 1 + 0.444 * 3 + 0.239 * 3 + 0.042 * 1$

$= 0.275 + 1.332 + 0.717 + 0.042 = 2.366$

$y(0.5) = 0.125 * 1 + 0.375 * 3 + 0.375 * 3 + 0.125 * 1$

$= 0.125 + 1.125 + 1.125 + 0.125 = 2.5$

$y(0.65) = 0.042 * 1 + 0.239 * 3 + 0.444 * 3 + 0.275 * 1$

$= 0.042 + 0.717 + 1.332 + 0.975 = 2.366$

$y(0.85) = 0.003 * 1 + 0.058 * 3 + 0.325 * 3 + 0.614 * 1$

$= 0.003 + 0.174 + 0.975 + 0.614 = 1.766$

$y(1) = B_3 = 1$

BEZIER CURVES

So

$P(0) = [1 \quad 1]$

$P(0.15) = [1.5 \quad 1.766]$

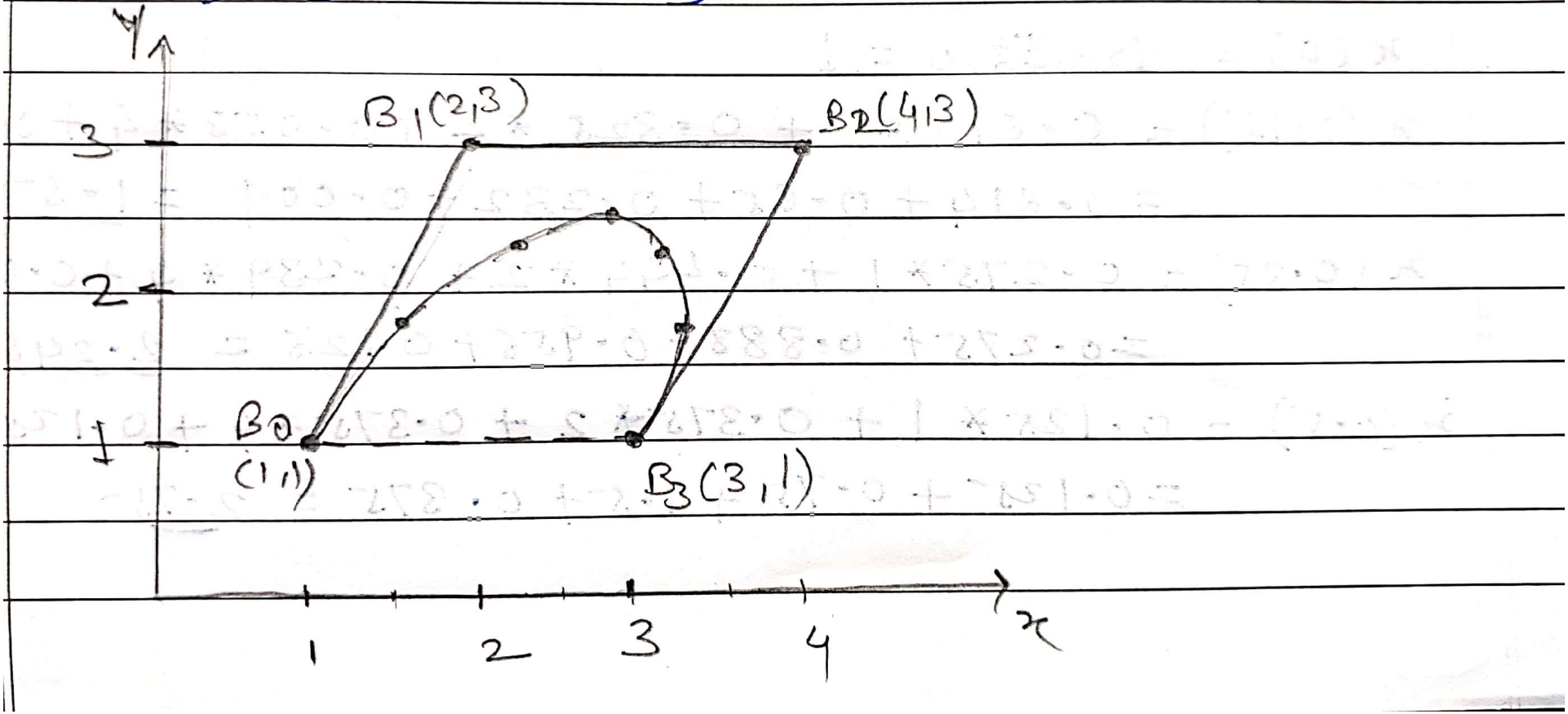
$P(0.35) = [2.245 \quad 2.366]$

$P(0.5) = [2.75 \quad 2.5]$

$P(0.65) = [3.121 \quad 2.366]$

$P(0.85) = [3.248 \quad 1.766]$

$P(1) = [3 \quad 1]$



BEZIER CURVES – Matrix Form

Take n = 3:

$$\binom{n}{i} = \binom{3}{i} = \frac{6}{i!(3-i)!}$$

$$J_{3,0}(t) = 1.t^0(1-t)^3 = (1-t)^3;$$

$$J_{3,1}(t) = 3.t.(1-t)^2;$$

$$J_{3,2}(t) = 3.t^2.(1-t);$$

$$J_{3,3}(t) = t^3.$$

Thus, for Bezier:

$$P(t) = (1-t)^3 B_0 + 3t(1-t)^2 B_1 + 3t^2(1-t)B_2 + t^3 B_3$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}; n = 3.$$

BEZIER CURVES - Application

- Computer graphics: Bezier curves are widely used in computer graphics to model smooth curves
- Animation: In animation application, such as Adobe Flash and synfig, Bezier curves are used to Outline, for example movement
- Font: TrueType fonts use Bezier splines composed of quadratic Bezier curves

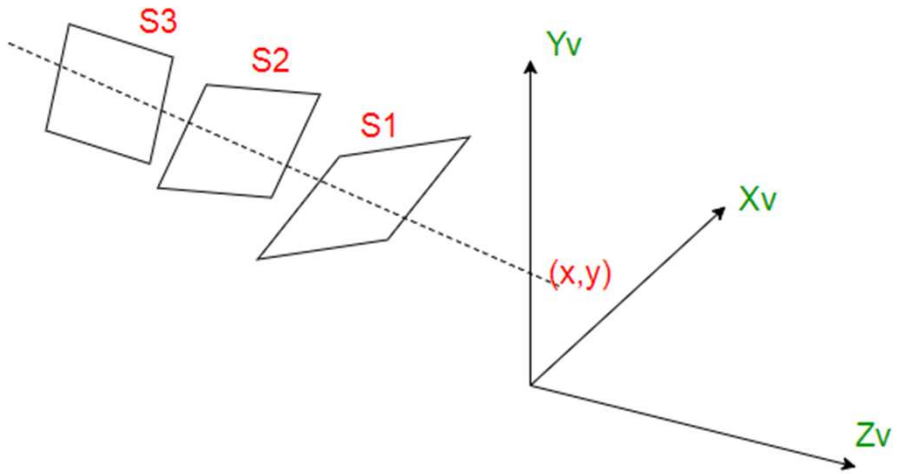
Z-buffer/Depth buffer algorithm

Z-Buffer or Depth-Buffer Method

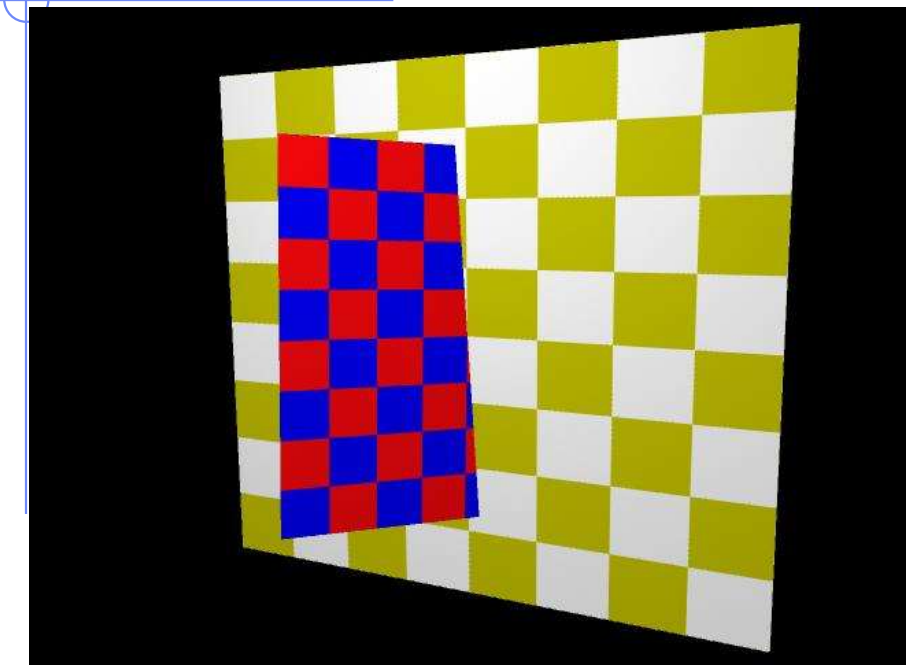
- When viewing a picture containing non transparent (opaque) objects and surfaces, it is not possible to see those objects from view which are behind from the objects closer to eye.
- To get the realistic screen image, removal of these hidden surfaces is must.
- The identification and removal of these surfaces is called as the **Hidden-surface problem.**
- Z-buffer, which is also known as the Depth-buffer method is one of the commonly used method for hidden surface detection. It is an **Image space method**.
- Normally z-axis is represented as the depth.
- In the Z buffer method, each surface is processed separately one position at a time across the surface.
- After that the depth values i.e, the z values for a pixel are compared and the closest surface determines the color to be displayed in frame buffer.

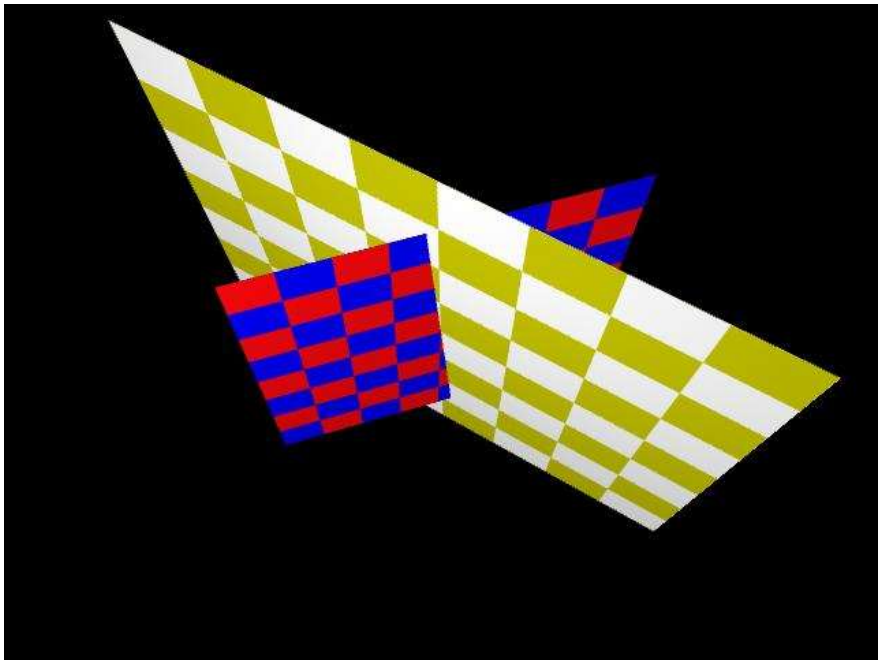
Z-Buffer or Depth-Buffer Method

- Ex: Let S1, S2, S3 are the surfaces.
- The surface closest to projection plane is called visible surface.
- The computer would start (arbitrarily) with surface 1 and put it's value into the buffer.
- It would do the same for the next surface. It would then check each overlapping pixel and check to see which one is closer to the viewer and then display the appropriate color.
- As at view-plane position (x, y), surface S1 has the smallest depth from the view plane, so it is visible at that position.



Example





Goal is to figure out which color to make the pixels based on what's in front of what.

Z-Buffering

- In this method, 2 buffers are used :
  - Frame/Refresh buffer: stores the attributes(intensity or shade) of each pixel in image.
  - Z-buffer/Depth Buffer: stores the z-coordinate or depth of every pixel.
- A z-buffer is a two dimensional array of real numbers of the same size as the frame buffer.
- A z-buffer algorithm scan converts an entire face at a time into both the frame buffer and z-buffer.

Z-Buffering

- Depth Values for a surface position(x , y) are calculated from the plane equation for each surface by:

$$z = \frac{(-Ax - By - D)}{C}$$

- Depth Values across the edge are calculated by:

$$z` = \frac{(-A(x+1) - By - D)}{C}$$

- Depth Values down the edge are recursively calculated by:

$$z` = z + \frac{(A/m) + B}{C}$$

  - m is slope of the edge.

Z-Buffering: Algorithm

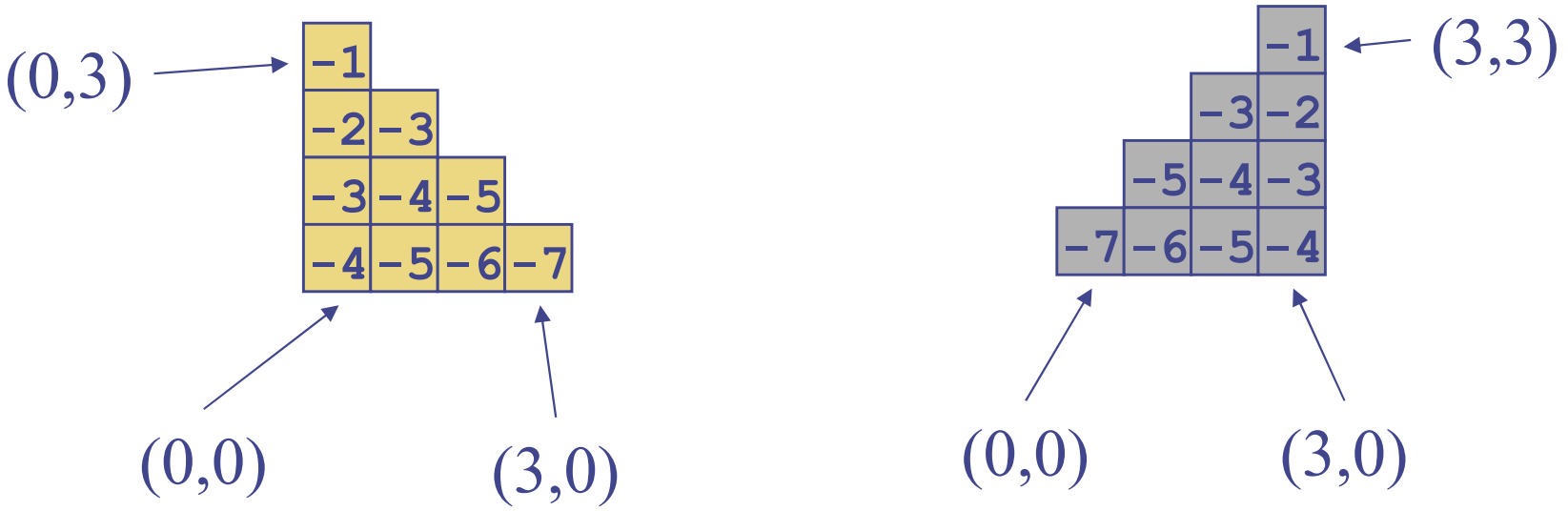
```
color array FRAMEBUFFER[XMAX][YMAX]
 real array ZBUFFER[XMAX][YMAX]

 begin
   Initialize FRAMEBUFFER to backgroundcolor;
  Initialize ZBUFFER to -∞
   for each polygon
     for each pixel (x,y) in polygon
       p_z = polygon's z-value at (x,y);
           if (p_z > ZBUFFER[x][y]))
             FRAMEBUFFER[x][y]=color of polygon
             ZBUFFER[x][y]= p_z

  end
```

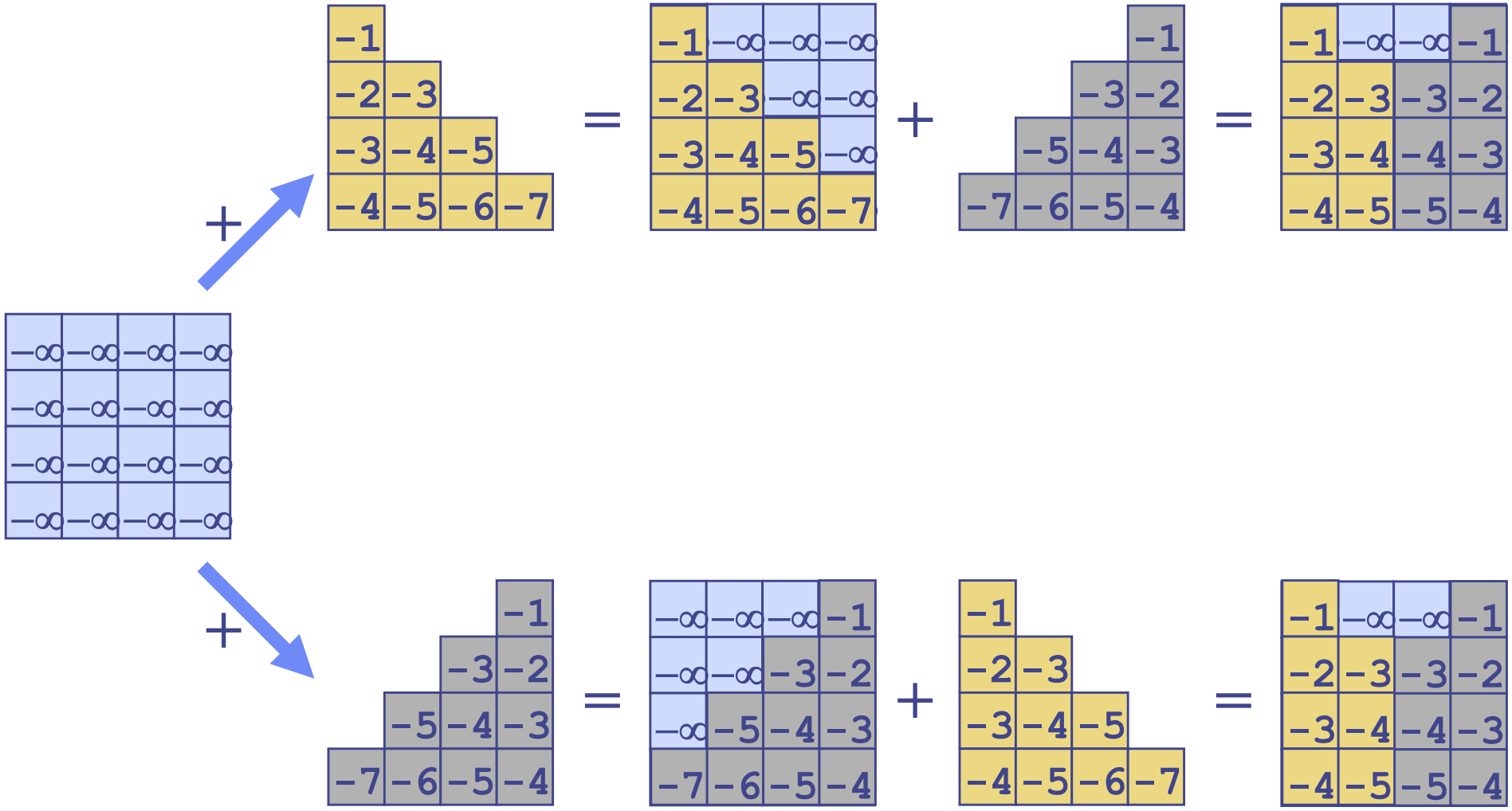
Z-Buffering: Example

Scan convert the following two polygons.
The number in the pixel represents the z- value for that pixel



Does order matter?

Z-Buffering: Example



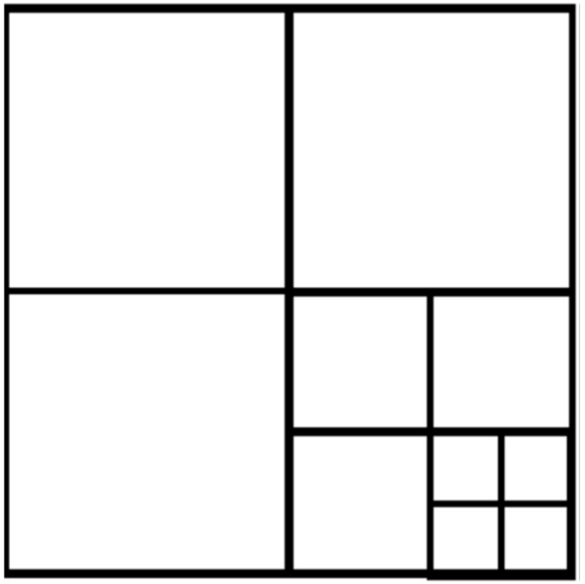
Z-Buffering : Summary

- Advantages:
  - Easy to implement
  - Fast with hardware support → Fast depth buffer memory
  - No sorting of objects required.
  - No object to object comparison is required.
  - The method is simple to use and does not require additional data structure.
  - This method can be executed quickly even with many polygons.
- Disadvantages:
  - Extra memory required for z-buffer
  - Wastage of time may occur because of drawing of hidden objects.
  - Only used for Opaque surface, not used for transparent object.

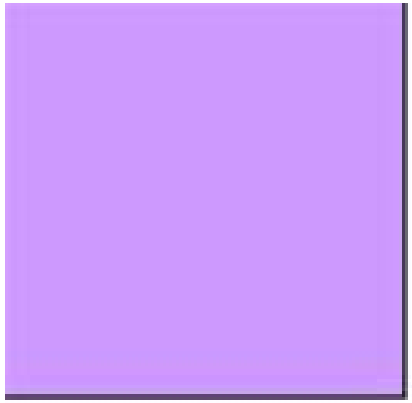
Warnock Algorithm / Area Subdivision Algorithm

Warnock or Area Subdivision Algorithm

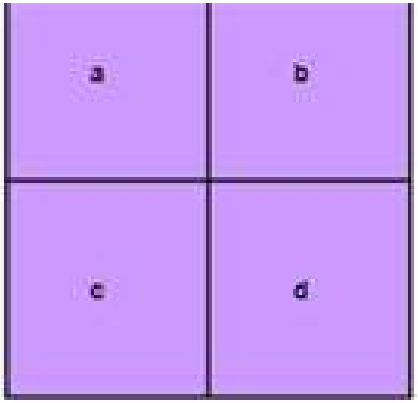
- It was invented by John Warnock and also called a Warnock Algorithm.
- It is based on a divide & conquer method.
- It uses fundamental of area coherence.
- It is used to resolve the visibility of algorithms.
- The total viewing area is successively divided into smaller and smaller rectangles until each small area is simple, ie. it is a single pixel, or is covered wholly by a part of a single visible surface or no surface at all.
- Warnock's algorithm is a recursive area-subdivision algorithm.



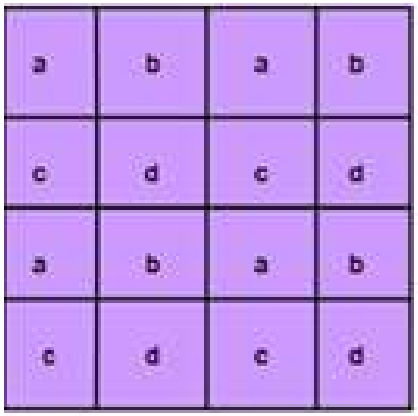
Warnock or Area Subdivision Algorithm



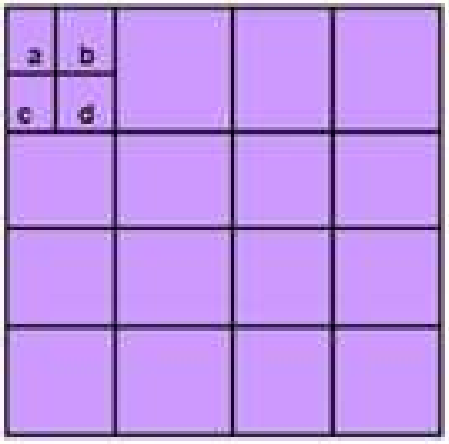
Original area
(a)



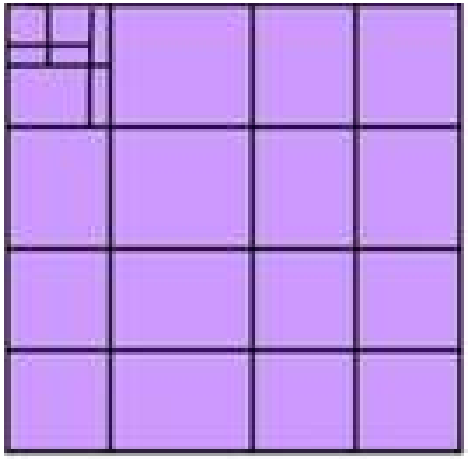
First division or subdivision of area
(b)



Second division
(c)



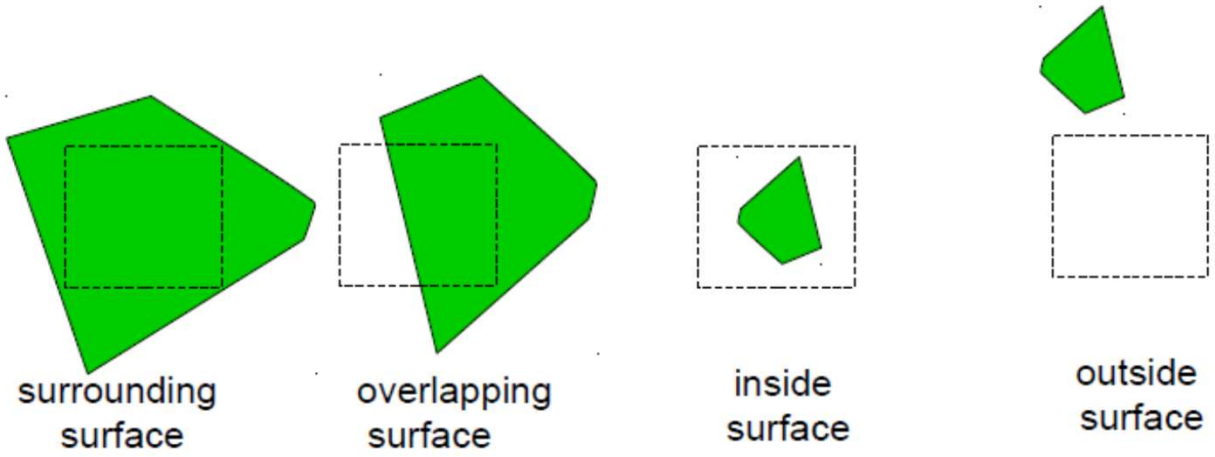
Third subdivision
(d)



Fourth subdivision
(e)

Warnock or Area Subdivision Algorithm

- The procedure to determine whether we should subdivide an area into smaller rectangle is:
  1. We first classify each of the surfaces, according to their relations with the area:
     a. Surrounding surface - a single surface completely encloses the area
     b. Overlapping surface - a single surface that is partly inside and partly outside the area
     c. Inside surface - a single surface that is completely inside the area
     d. Outside surface - a single surface that is completely outside the area.

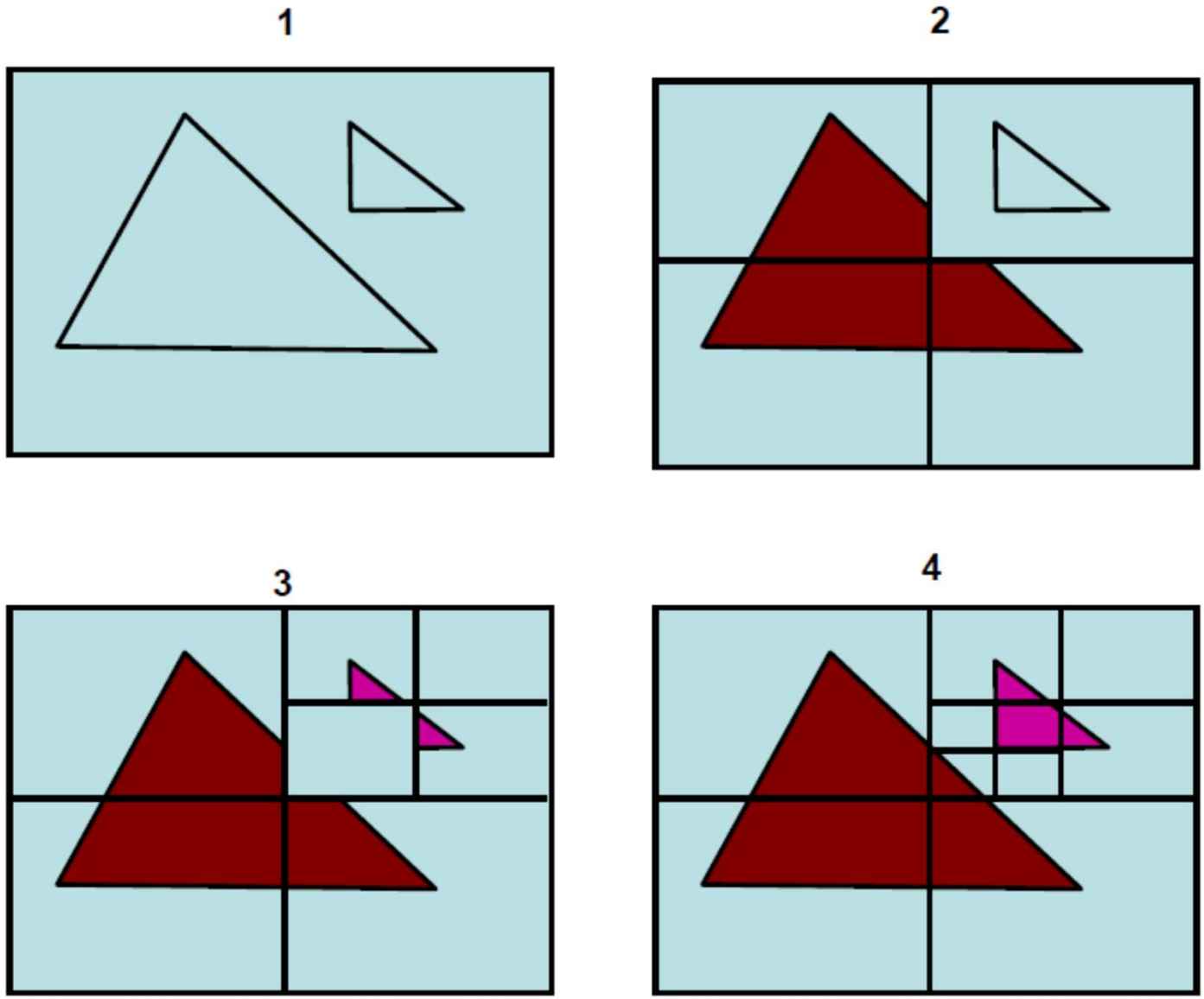


Warnock or Area Subdivision Algorithm

2. At each stage of the algorithm, examine the areas:
   - If no polygons lie within an area, the area is filled with the background colour
   - If only one polygon is in part of the area, the area is first filled with the background colour and then the polygon is scan converted within the area.
   - If one polygon surrounds the area and it is in front of any other polygons, the entire area is filled with the color of the surrounding polygon.
   - Otherwise, subdivide the area and repeat the above 4 tests
3. Subdivision continues until:
   - All areas meet one of the four criteria
   - An area is pixel size

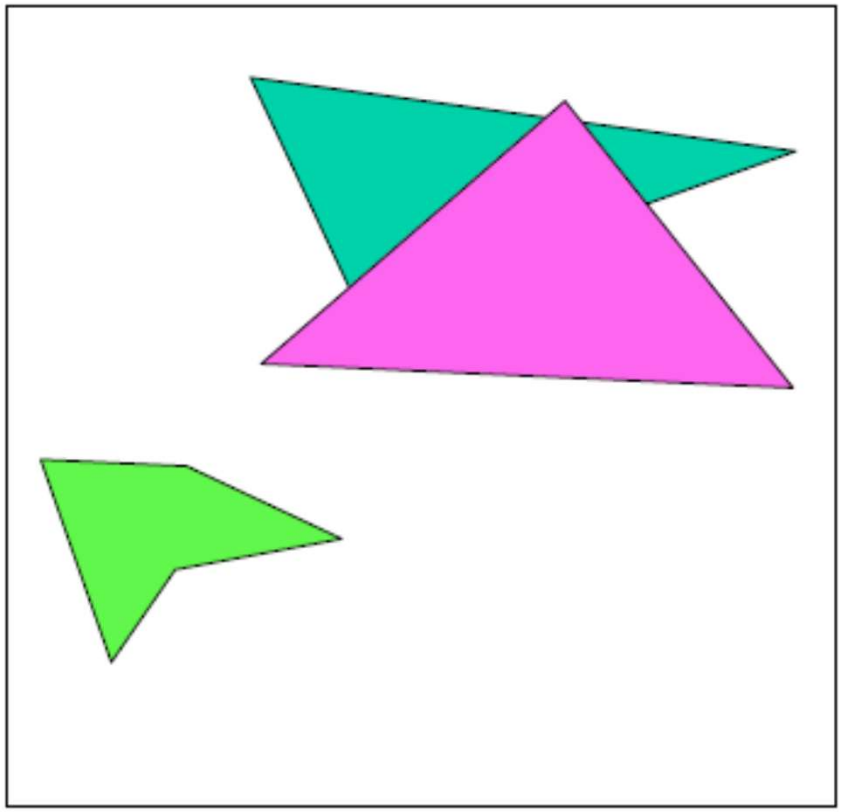
Warnock or Area Subdivision Algorithm

- Example:

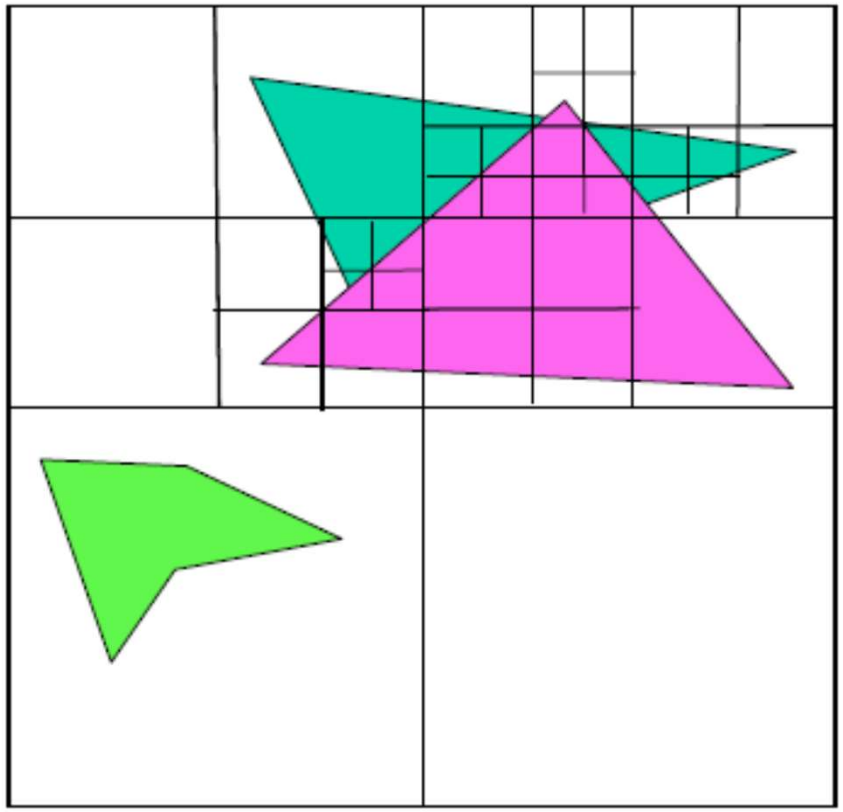


Warnock or Area Subdivision Algorithm

- Example:



Initial scene



Final scene

End of Unit 4