Curves Note

A curve is a set of points & can be needed to be drawn in vector graphics.

e.g. a circle where the area bounded by it is a region.

Types of curve reprentations:

- **Implicit representation**: Generalised Form

$f(x,y) = 0$ (Phenomenon introduced in MPL chapter where points on the function give value = zero)

or $f(x,y,z) = 0$

(2D line) | (3D plane)

$\rightarrow$ $ax + by + c = 0$ | $ax + by + cz + d = 0$

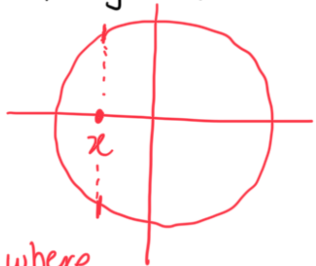
$\rightarrow$ $x^2 + y^2 - r^2 = 0$ (Circle) $\quad$ $x^2 + y^2 + z^2 - r^2 = 0$ (Sphere)

- **Explicit representation**: Defines a variable relation to another

$y = x^2$, $y = mx + c$

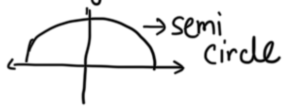
However, for circle, $y = \pm\sqrt{r^2 - x^2}$ Not a function as not a 1-1 relation

Multiple values of y for 1 x value



Hence this is where parametric equations work best.

* If needed, only 1 value of y taken, e.g. $y = \sqrt{r^2 - x^2}$ we get



- **Parametric Representation**:

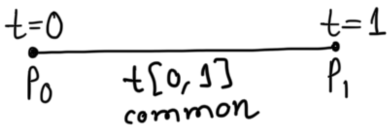
For a circle, $t[0, 2\pi]$

$f(t) = (r\cos t, r\sin t)$ $\rightarrow$ t a 3rd parameter used to define x & y

For every value of t, only 1 possible output.

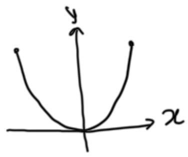
For a line segment,

$P(t) = P_0 + t(P_1 - P_0)$



For quadratic curves,

$f(t) = (\overset{x}{2t}, \overset{y}{t^2}) \rightarrow$



* Every explicit representation can be written implicitly however not every implicit representation can be written explicitly.

- Subdivision Representation:

  → Given set of points, add new points by removing old points (Repeat)

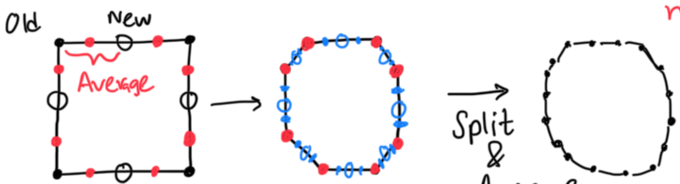
  → Each division makes the new points closer to the intended curve

  Chaikin's Algorithm:

(Split) Insert new points at midpoints of adjacent old points

(Average) Take the average of each midpoint with its neighbours.

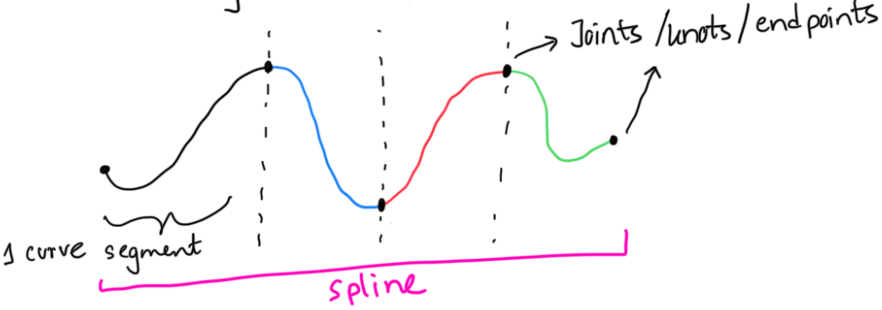
Loop back to splitting again. (Old vertices replaced with the averaged new ones)



After splitting new vertices produced → After averaging red vertices produced

- Procedural Representation:

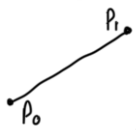
Gist: What if to render any complex shaped curve, we break it down to smaller pieces and then using known functions such as line segment, quadratic and cubic curves, we try to **morph** them into those shapes?



- We breakdown the spline into multiple smaller curves.

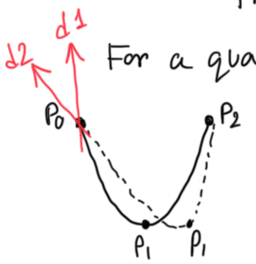
- Each curve segment shown using a linear, quadratic or a higher degree polynomials, however there are certain drawbacks related to them.

For a linear polynomial (a line segment); (2 points)



the slope of a line segment is fixed which cannot be changed. Hence there is no flexibility here. $(\overrightarrow{P_1 - P_0})$ Cannot change the direction the curve leaves from P_0 & P_1.

For a quadratic polynomial, (3 points)



more freedom here to control the slope however using P_1 point only 1 of the slopes i.e. at P_0 or P_2 can be controlled (the direction the curve leaves)

(if dragged here direction the curve leaves from P_0 changes)

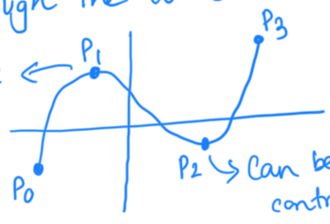
Hence, since both endpoints' direction cannot be controlled, quadratic is not idle.

For higher-degree polynomial, more control over the slope at endpoints however computationally expensive and wiggles around endpoints.

* Hence ideal polynomials are <u>cubic</u>! (4 points)

- Low degree polynomial
- Having 2 internal points other than 2 endpoints, help to control the direction everywhere through the curve

Can be used to control direction of



Can be used to control direction of P_3

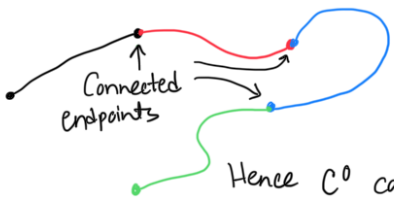
Continuity

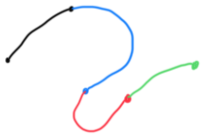
Now, after representing each segment, we will need to join them to render the final spline.

- The endpoints of the segments need to be the same for them to connect and hence in order to ensure this, we look at the <u>smoothness/continuity</u> of the curve.

- **<u>Parametric Continuity</u>** $\boxed{C^n}$

$C^0 \to$ if all the endpoints are connected (<u>same position</u>)

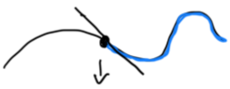




Hence C^0 continuous

$C^1 \to$ curves at endpoints have the same tangent (<u>velocity</u>)

$\to$ if C^1 continuous then C^0 continuous as well.



Same derivative at joint (direction + magnitude)

✓ C^1

Different direction of tangent vectors here

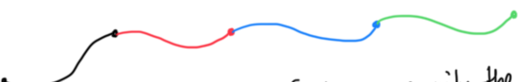
✗ C^1

$C^2 \to$ curves at endpoints have same 2^{nd} order derivative (acceleration)

* $C(n) \to$ all derivatives upto n match (direction + size)

- **<u>Geometric Continuity</u>** $\boxed{G^n}$

$G^0 \to$ if 2 curve segments join together
↳ same as C^0



$G^1 \to$ If the directions (not necessarily the magnitude) of the curve segments' tangent vectors are equal at endpoints.

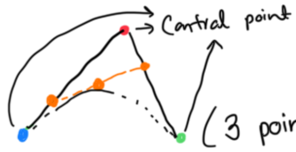
* $G(n) \rightarrow$ directions of all derivatives match upto n.

<u>Note:</u> By increasing parametric continuity, $C(n)$, the smoothness of the curve increases.

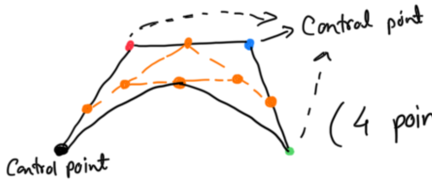
Bèzier curve

$$lerp(P_0, P_1, t) = P_0 + t(P_1 - P_0)$$ → Parametric Equation

- Using a fixed set of control points, we move around the control points to change shape of the curve. e.g. Quadratic Bèzier Curve



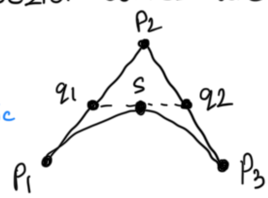
e.g. Cubic Bèzier Curve



- The Bèzier curves are build using linear interpolations

e.g.

Quadratic Bèzier curves



$$q_1 = lerp(P_1, P_2, t)$$
$$q_2 = lerp(P_2, P_3, t)$$
$$s = lerp(q_1, q_2, t)$$

$$\therefore\ s = lerp(q_1, q_2, t)$$
$$= q_1 + t(q_2 - q_1)$$

We know, $q_1 = lerp(P_1, P_2, t)$
$$= P_1 + t(P_2 - P_1)$$

similarly, $q_2 = P_2 + t(P_3 - P_2)$

$$\therefore\ s = \underbrace{P_1 + t(P_2 - P_1)}_{q_1} + t\left(\underbrace{[P_2 + t(P_3 - P_2)]}_{q_2} - \underbrace{[P_1 + t(P_2 - P_1)]}_{q_1}\right)$$

$$= P_1 + tP_2 - tP_1 + t\left(P_2 + tP_3 - tP_2 - P_1 - tP_2 + tP_1\right)$$
$$= P_1 + tP_2 - tP_1 + tP_2 + t^2P_3 - t^2P_2 - tP_1 - t^2P_2 + t^2P_1$$
$$= P_1(1 - t - t + t^2) + P_2(t + t - t^2 - t^2) + t^2P_3$$

$S = (1-2t+t^2)P_1 + (2t-2t^2)P_2 + t^2P_3$

(General Formula for Quadratic Bèzier curve)

(De Casteljau's Algorithm)

- General Formula for Cubic Bèzier curve:

$S = (1-3t+3t^2-t^3)P_1 + (3t-6t^2+3t^3)P_2 + (3t^2-3t^3)P_3 + t^3P_4$

- Matrix notation for s:

(Cubic)

$$S = \underbrace{\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{M_B:\ \text{Matrix Basis}} \underbrace{\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}}_{G_B:\ \text{Geometric properties (Convex Hull)}}$$

$$\boxed{\therefore S = T.M_B.G_B}$$

$$S = \underbrace{\begin{bmatrix} t^2 & t & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{M_B:\ \text{Matrix Basis}} \underbrace{\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}}_{G_B}$$

$$\boxed{\therefore S = T.M_B.G_B}$$

Bernstein Polynomials:

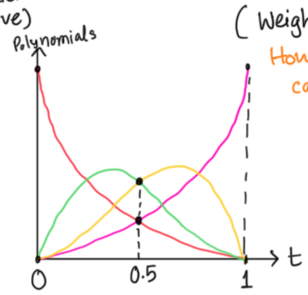
$Q(t) = \boxed{(1-3t+3t^2-t^3)}P_1 + \boxed{(3t-6t^2+3t^3)}P_2 + \boxed{(3t^2-3t^3)}P_3 + \boxed{t^3}P_4$

(Cubic Bèzier curve)

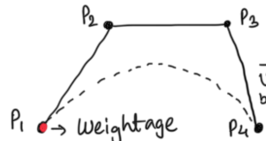
Bernstein basis/polynomial

(Weightage of each control points)

How much each control point contribute to a point on the Bèzier curve



– At t=0, the weightage of P1 (the starting coordinate is the maximum while others are 0)



$\xrightarrow{t=0}$ Using bernstein polynomials

$1-3(0)+3(0)^2-(0)^3 = \underline{1}.(P_1)$

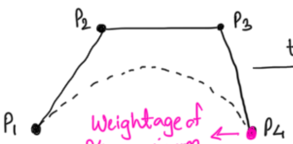
$3(0)-6(0)^2+3(0)^3 = \underline{0}.(P_2)$

$3(0)^2-3(0)^3 = \underline{0}.(P_3)$

$(0)^3 = \underline{0}.(P_4)$

Weightage of P1 maximum

P2, P3, P4 minimum at the first point of curve



$\xrightarrow{t=1}$

$1-3(1)+3(1)^2-(1)^3 = \underline{0}.\ P_1$

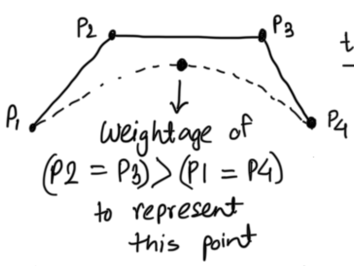
$3(1)-6(1)^2+3(1)^3 = \underline{0}.\ P_2$

$3(1)^2-3(1)^3 = \underline{0}.\ P_3$

$(1)^3 = \underline{1}.\ P_4$

Weightage of P4 maximum

P1, P2, P3 minimum



$t = 0.5 \longrightarrow$

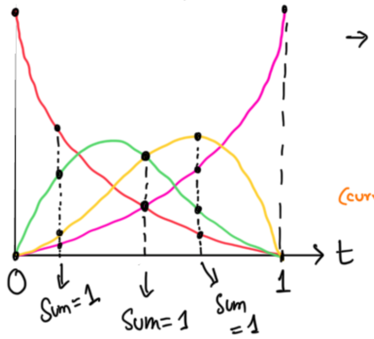
$1 - 3(0.5) + 3(0.5)^2 - (0.5)^3 = \underline{0.125} \cdot P1$

$3(0.5) - 6(0.5)^2 + 3(0.5)^3 = \underline{0.375} \cdot P2$

$3(0.5)^2 - 3(0.5)^3 = \underline{0.375} \cdot P3$

$(0.5)^3 = \underline{0.125} \cdot P4$

Sum of these polynomials $= 1$
(at any $t \rightarrow [0,1]$)



→ This is called a convex combination where the weighted sum of points / vectors $= 1$.

$$C = \lambda_1 P_1 + \lambda_2 P_2 + \ldots\ldots \lambda_n P_n$$

(curve point) → Bernstein polynomials

where, $\sum_{i=1}^{n} \lambda_i = 1$, $\lambda_i \geq 0$ (non-negative weights)

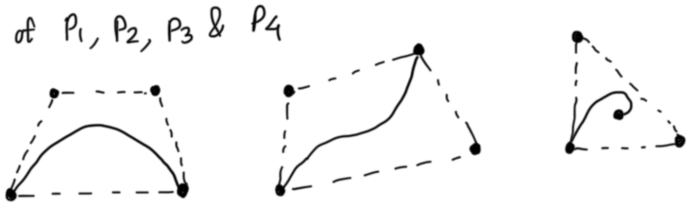
__Note__ → Only for t ranging from 0 to 1, will give a convex combination.

- __Convex Hull__

Any convex combination will lie inside the convex hull of its control points.

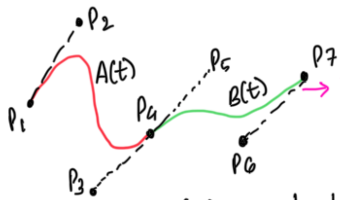
__convex hull__
(Geometric shape → triangle/polygon)

e.g. a cubic Bézier curve must be bound by the convex hull of P_1, P_2, P_3 & P_4



Continuity Condition Checking

- __For C^0 continuity__



→ Two cubic Bèzier curves

To be C^0 continuous, $A(t)$ & $B(t)$ needs to be connected.

∴ Where $A(t)$ ends, $B(t)$ needs to start

$$A(1) = B(0)$$ → Connected at $\underline{P4}$

$(t=1)$ since end point of curve

$(t=0)$ since start of curve

• <u>For C^1 continuity</u>, the velocity at P_4 must be equal for both curves. $A'(1) = B'(0)$

We know,

$Q(t) = (1 - 3t + 3t^2 - t^3)P_1 + (3t - 6t^2 + 3t^3)P_2 + (3t^2 - 3t^3)P_3 + t^3P_4$

(Provides positions of the curve)

$Q'(t) = (-3 + 6t - 3t^2)P_1 + (3 - 12t + 9t^2)P_2 + (6t - 9t^2)P_3 + 3t^2P_4$

$\therefore Q'(0) = 3P_2 - 3P_1 = 3(P_2 - P_1)$ → Same for every cubic Bèzier curve

$\therefore Q'(1) = 3P_4 - 3P_3 = 3(P_4 - P_3)$

$\therefore A'(1) = B'(0)$

$\Rightarrow 3(P_4 - P_3) = 3(P_2 - P_1)$

Now for $B(t)$ its first & second points are P_4 & P_5

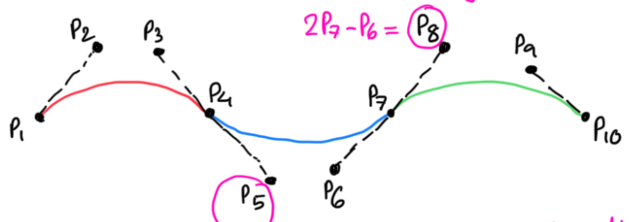
$\therefore 3(P_4 - P_3) = 3(P_5 - P_4)$

$\Rightarrow 3P_4 - 3P_3 = 3P_5 - 3P_4$

$\Rightarrow 3P_5 = 6P_4 - 3P_3$

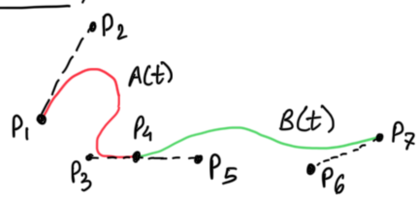
$\therefore \boxed{P_5} = 2P_4 - P_3$ // This condition needs to be fulfilled for the entire spline to be C^1 continuous. (Same for every connecting joint of 2 Bèzier curves)

Now controlled by P_3 & P_4, loss of local control



$= 2P_4 - P_3$ → <u>If yes</u> then C^1 continuous

• <u>For C^2 continuity</u>, same acceleration of curves at P_4.



$A''(1) = B''(0)$

We know

$Q'(t) = (-3 + 6t - 3t^2)P_1 + (3 - 12t + 9t^2)P_2 + (6t - 9t^2)P_3 + 3t^2P_4$

$Q''(t) = (6 - 6t)P_1 + (-12 + 18t)P_2 + (6 - 18t)P_3 + 6tP_4$

$\therefore Q''(0) = 6P_1 - 12P_2 + 6P_3 = 6(P_1 - 2P_2 + P_3)$

$Q''(1) = 6P_2 - 12P_3 + 6P_4 = 6(P_2 - 2P_3 + P_4)$

$\therefore A''(1) = B''(0)$

$6(P_2 - 2P_3 + P_4) = 6(P_1 - 2P_2 + P_3)$

Now for $B(t)$, its first, second & third points are P_4, P_5, P_6

$6(P_2 - 2P_3 + P_4) = 6(P_4 - 2P_5 + P_6)$

$\Rightarrow P_2 - 2P_3 + P_4 = P_4 - 2P_5 + P_6$

$\Rightarrow P_6 = P_2 - 2P_3 + P_4 - P_4 + 2P_5$

$= P_2 + 2P_5 - 2P_3$

$P_6 = P_2 + 2(P_5 - P_3)$

We know to maintain C^1 continuity before reaching C^2,

$P_5 = 2P_4 - P_3$

$\therefore P_6 = P_2 + 2(2P_4 - P_3 - P_3)$

$= P_2 + 2(2P_4 - 2P_3)$

$P_6 = P_2 + 4(P_4 - P_3)$ → Now P_6 is locked as it is controlled by P_2, P_4 & P_3.

Control over P_6 is lost

This condition needs to be fulfilled for C^2 continuity

Note: The more layers of continuity reached, the more control we lose as, changing one control point will affect other control points dependent on it.