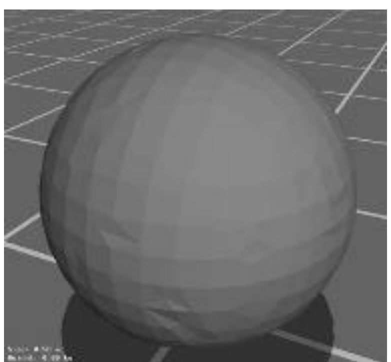
Computer Graphics (COMP0027) 2022/23

Spline Curves

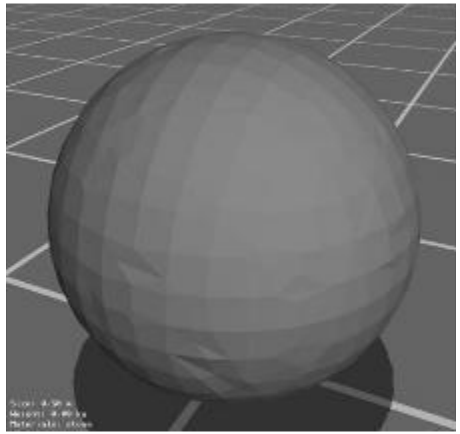
Tobias Ritschel

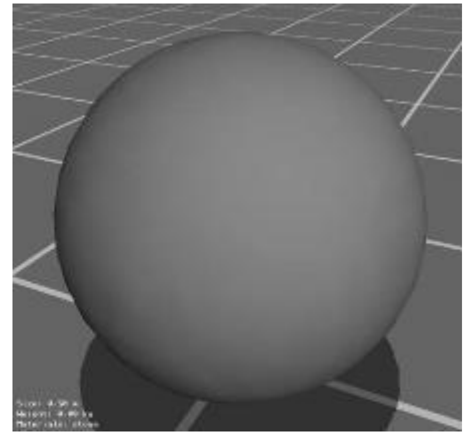
Limitations of Polygonal Meshes

- Planar facets (& silhouettes)
- Fixed resolution
- Deformation is difficult
- No natural parameterization (for texture mapping)



Need to Disguise the Facets

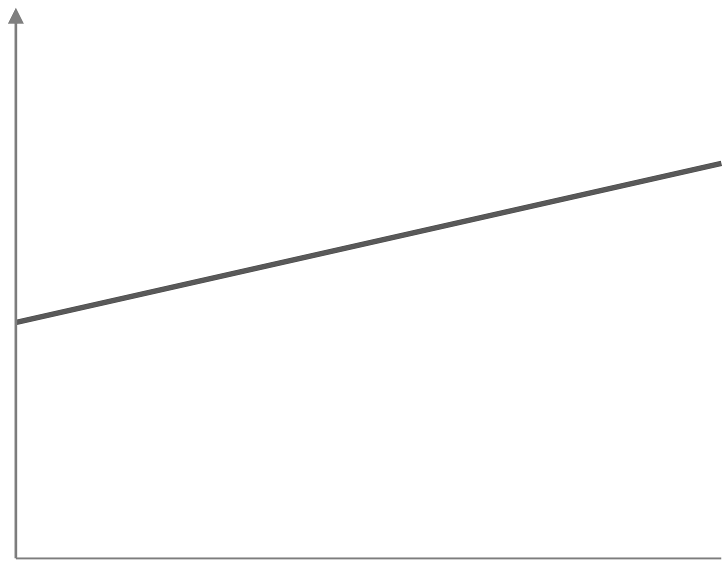




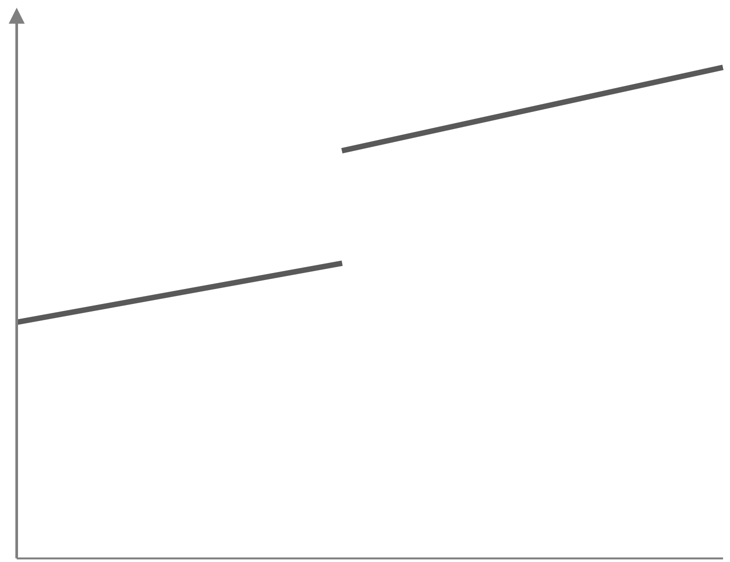
Name, from woodwork

Curves

C^0 Continuity

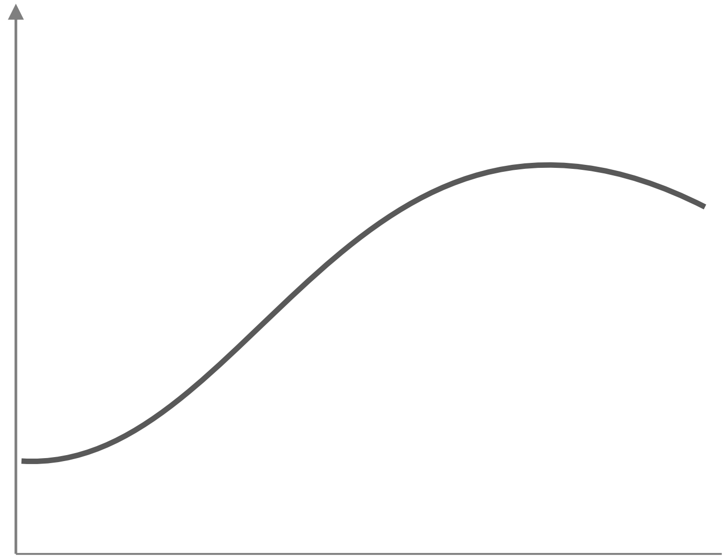


Yes

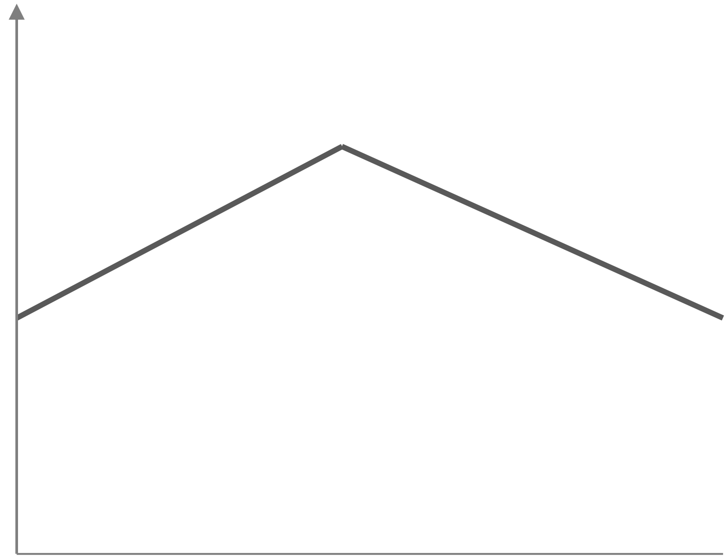


No

C^1 Continuity



Yes



No

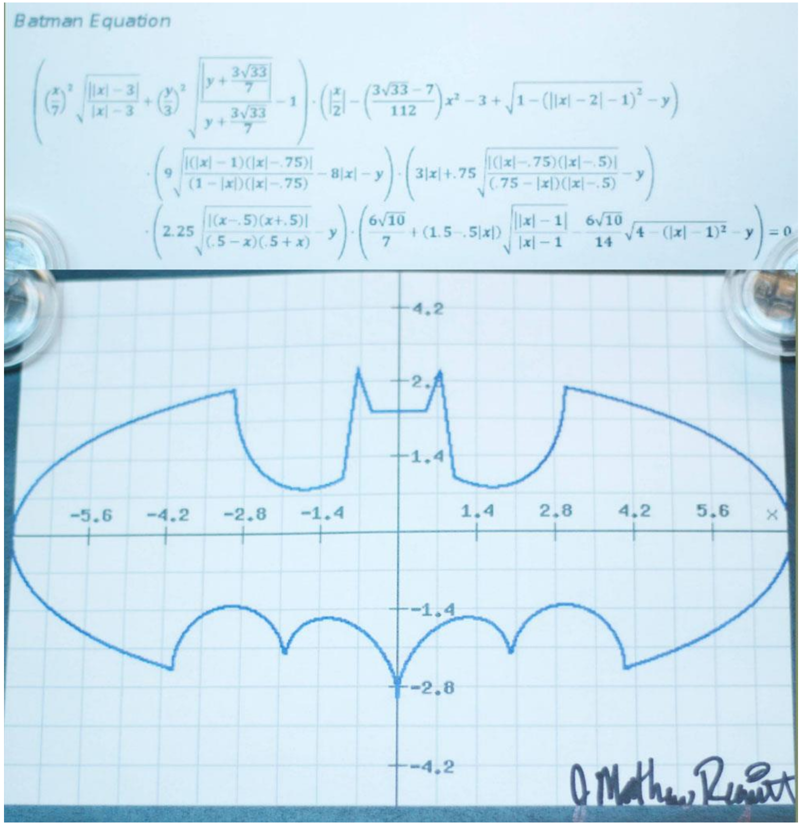
Continuity definitions

- C^0 continuous
  - curve/surface has no breaks/gaps/holes
- C^1 continuous
  - curve/surface derivative is continuous
  - tangent at join has same direction *and* magnitude
- C^n continuous
  - curve/surface through n^{th} derivative is continuous
  - important for shading

Analytic functions

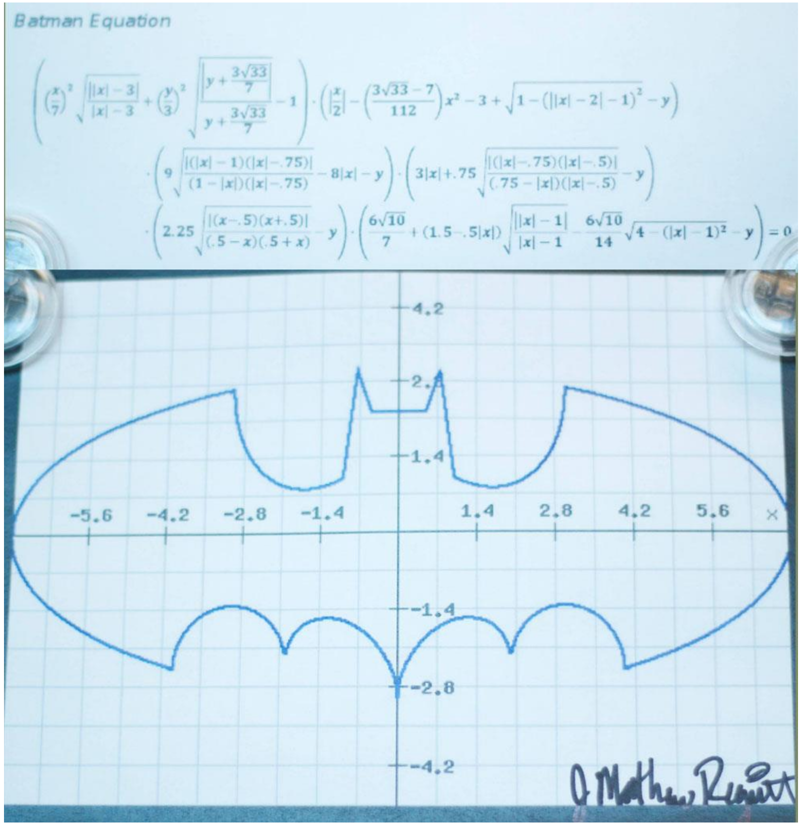
$$
\begin{aligned}
&\left(\left(\frac{x}{7}\right)^2 \sqrt{\frac{||x|-3|}{|x|-3}} + \left(\frac{y}{3}\right)^2 \sqrt{\frac{\left|y+\frac{3\sqrt{33}}{7}\right|}{y+\frac{3\sqrt{33}}{7}}} - 1\right) \cdot \left(\left|\frac{x}{2}\right| - \left(\frac{3\sqrt{33}-7}{112}\right)x^2 - 3 + \sqrt{1-(||x|-2|-1)^2} - y\right) \\
&\cdot \left(9\sqrt{\frac{|(|x|-1)(|x|-.75)|}{(1-|x|)(|x|-.75)}} - 8|x| - y\right) \cdot \left(3|x| + .75\sqrt{\frac{|(|x|-.75)(|x|-.5)|}{(.75-|x|)(|x|-.5)}} - y\right) \\
&\cdot \left(2.25\sqrt{\frac{|(x-.5)(x+.5)|}{(.5-x)(.5+x)}} - y\right) \cdot \left(\frac{6\sqrt{10}}{7} + (1.5-.5|x|)\sqrt{\frac{||x|-1|}{|x|-1}} - \frac{6\sqrt{10}}{14}\sqrt{4-(|x|-1)^2} - y\right) = 0
\end{aligned}
$$

Analytic functions



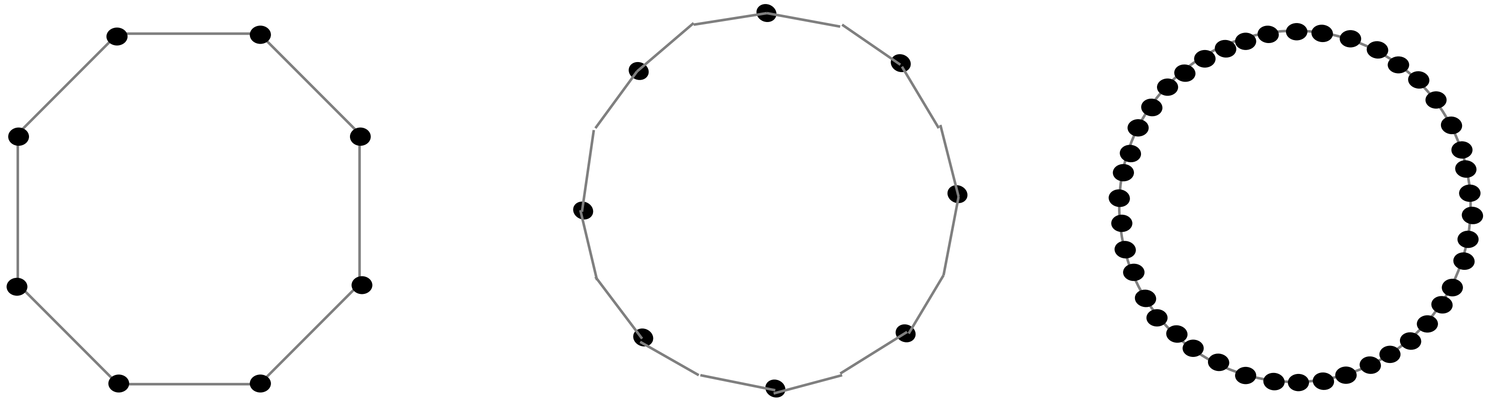
What if you want to have curves?

- Curves are often described with an analytic equation
- It's different from the discrete description of polygons
- How do you deal with it in Computer Graphics?



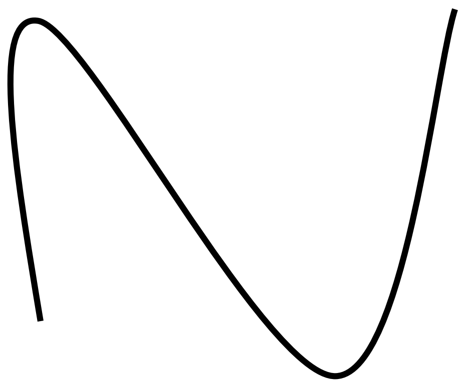
First Solution: Line Segments

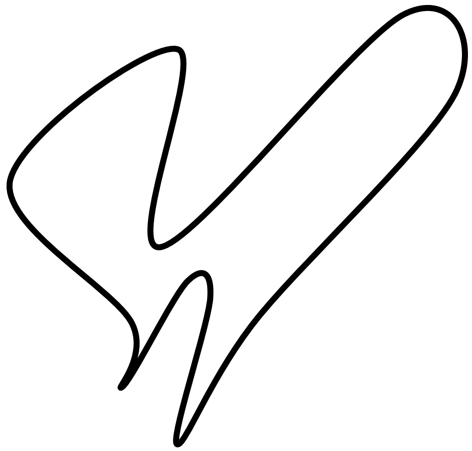
- At some point (e.g. magnification) any linear approximation will not be sufficient



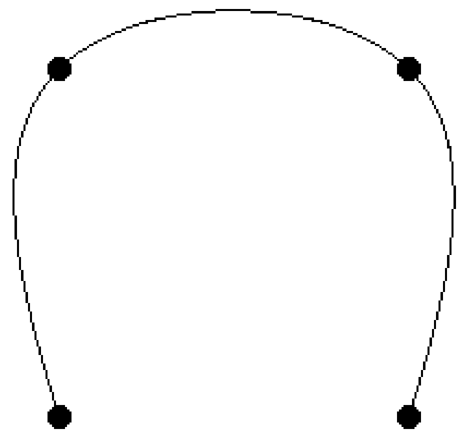
And for more complex curves?

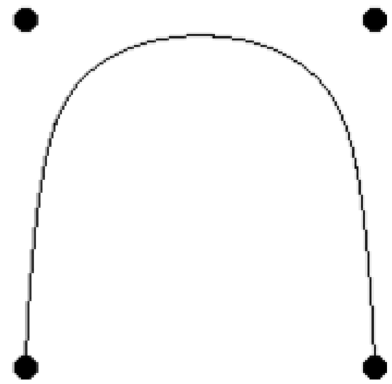
- Can I approximate this with line segments?





Interpolation vs. Approximation Curves





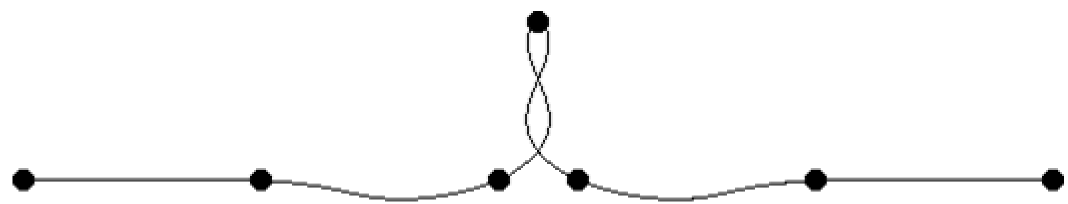
Interpolation

Curve must pass through control points

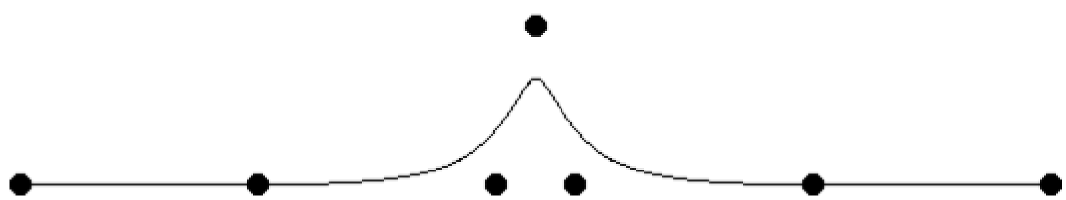
Approximation

Curve is influenced by control points

Interpolation vs. Approximation Curves



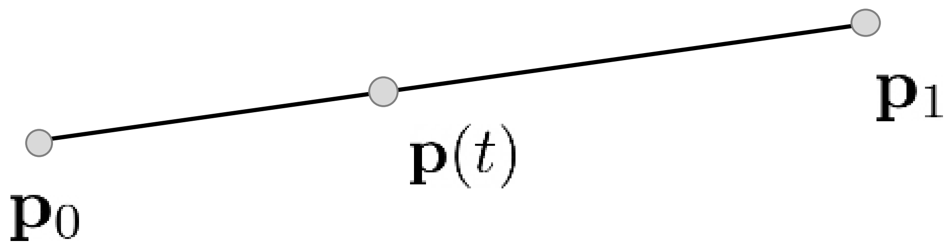
Interpolation – Over-constrained, lots of (undesirable?) oscillations



Approximation – More reasonable?

Parameterised line segment

$$\mathbf{p}(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$



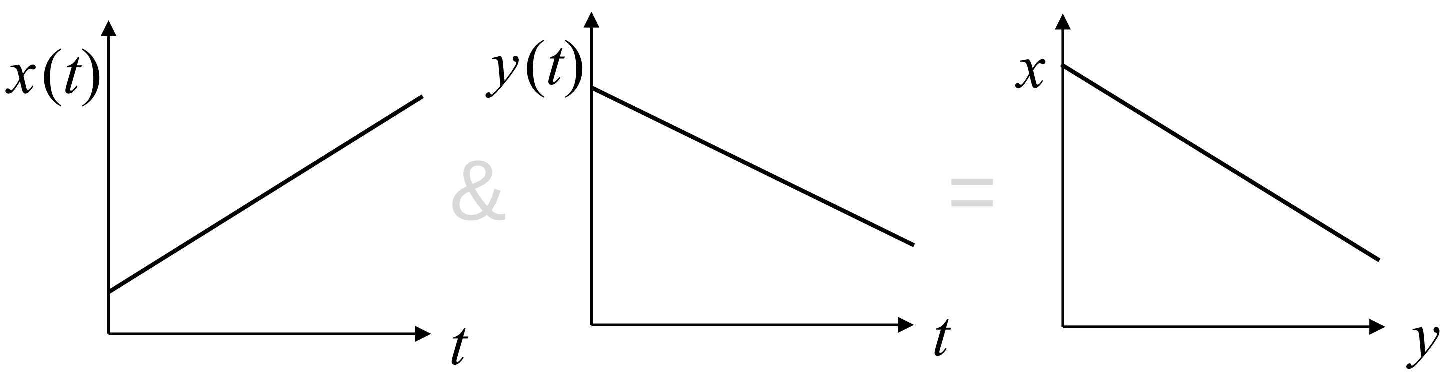
How to generalize to **non-linear** interpolation?

2D curves

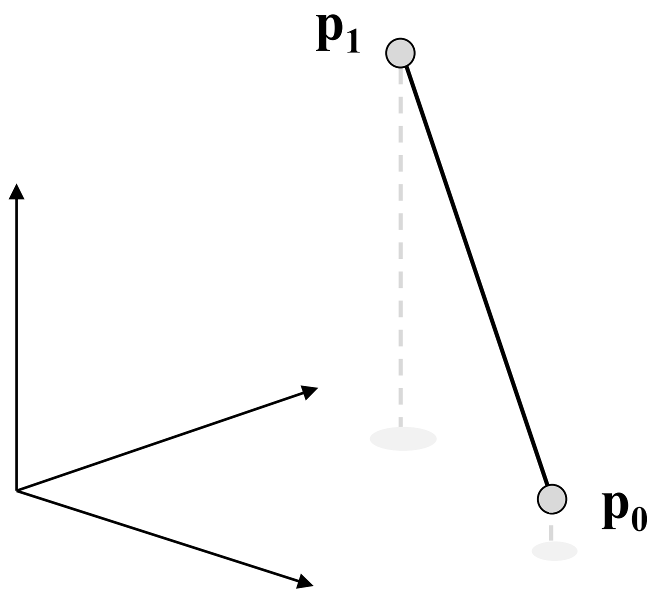
- $\mathbf{p}(t)$ maps from a 1D scalar t to a 2D vector $\mathrm{p}(t)$

$$x(t) = (1-t)x_0 + tx_1$$

$$y(t) = (1-t)y_0 + ty_1$$

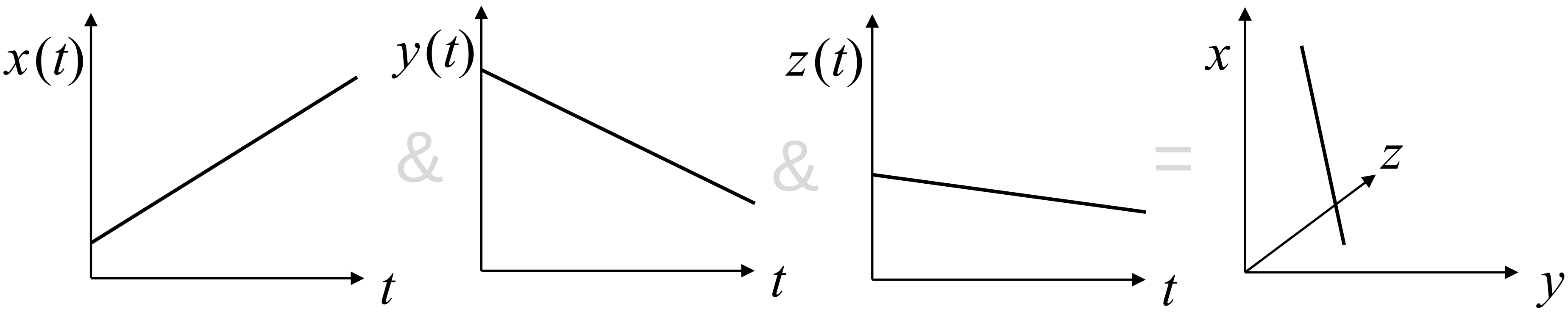


3D curve



3D curves

- Three coordinate functions

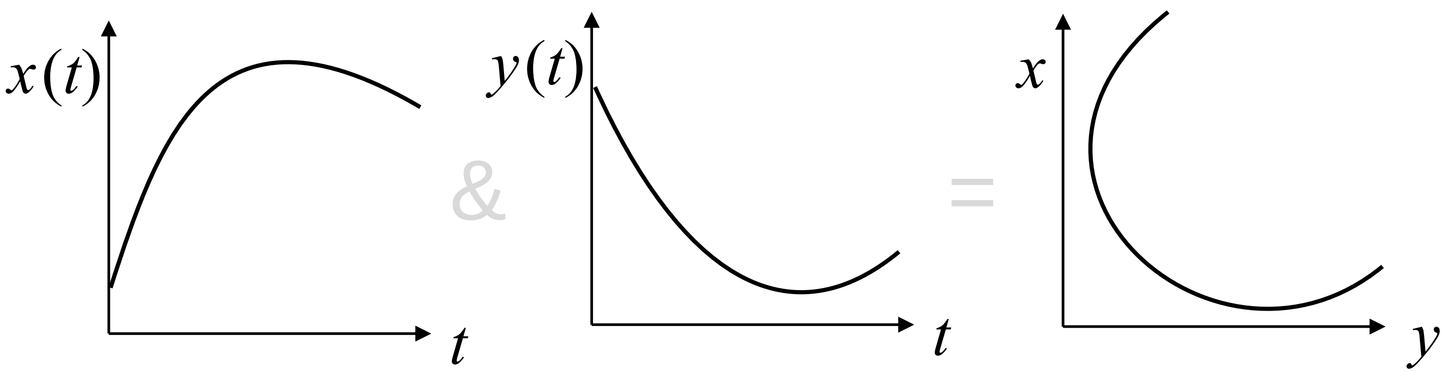


Bézier Curves

2D Bézier curves

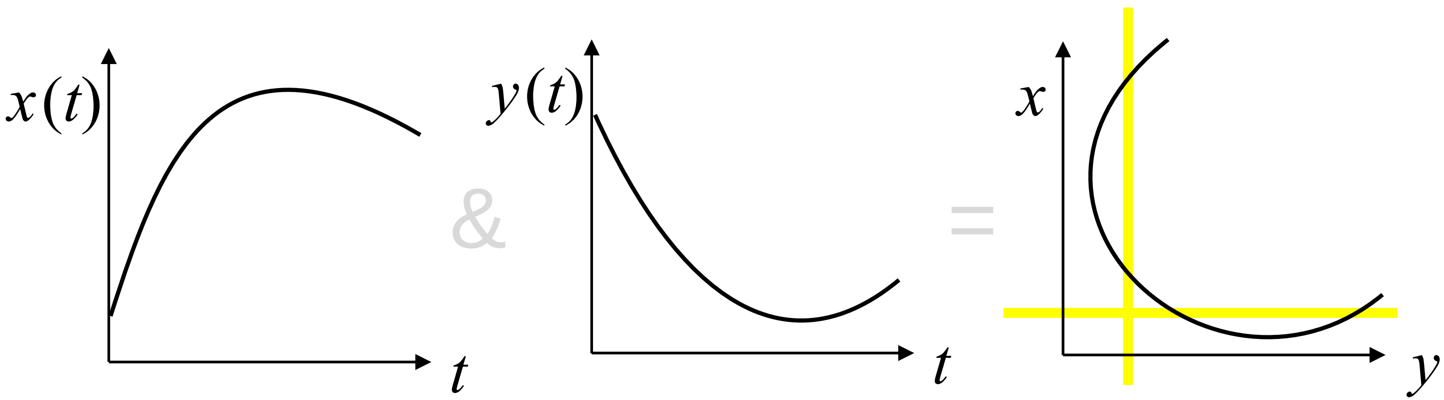
- Start with 2nd degree (quadratic)

$$\mathbf{p}(t) = (1-t)^2\mathbf{p}_0 + 2t(1-t)\mathbf{p}_1 + t^2\mathbf{p}_2$$

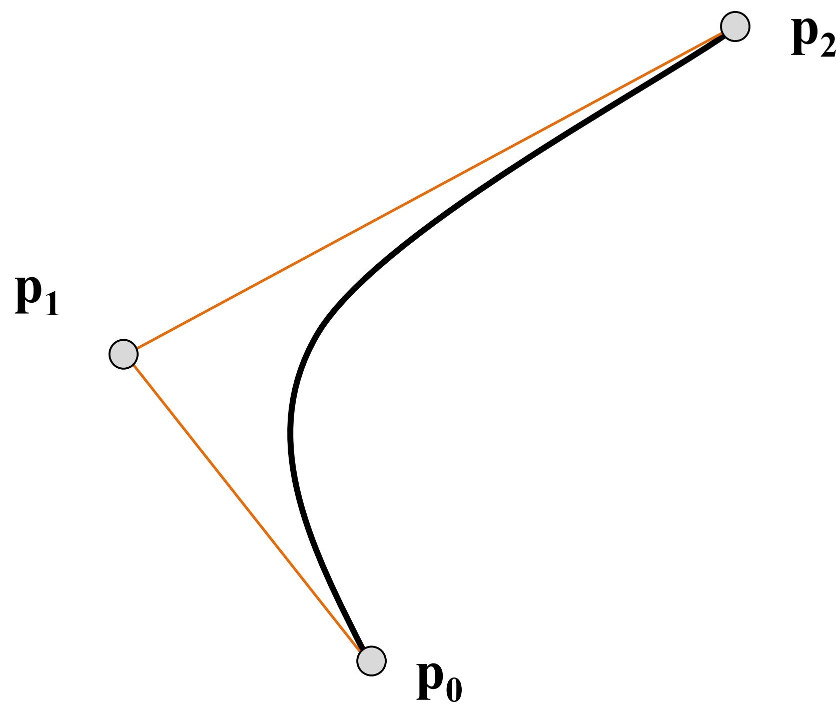


Bézier curves

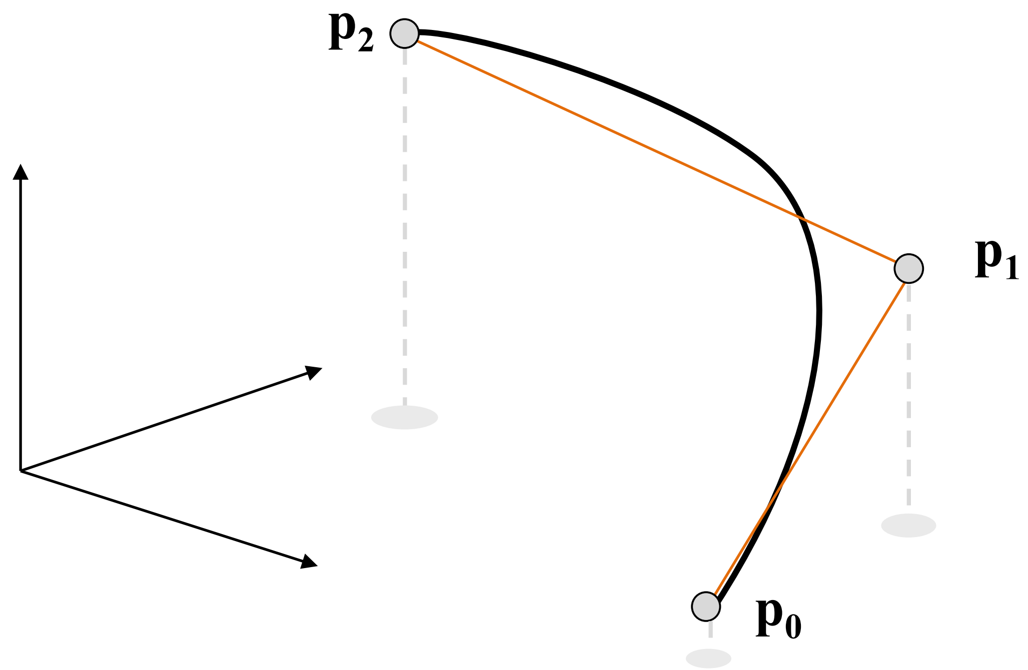
- Why this complication?
- I can write a line much easier?
- Not every curve C in $\mathbb{R}^2$ is a function $\mathbb{R}$ to $\mathbb{R}^2$



3 Control Points (quadratic 2D Bézier)



3 Control Points (quadratic 3D Bézier)

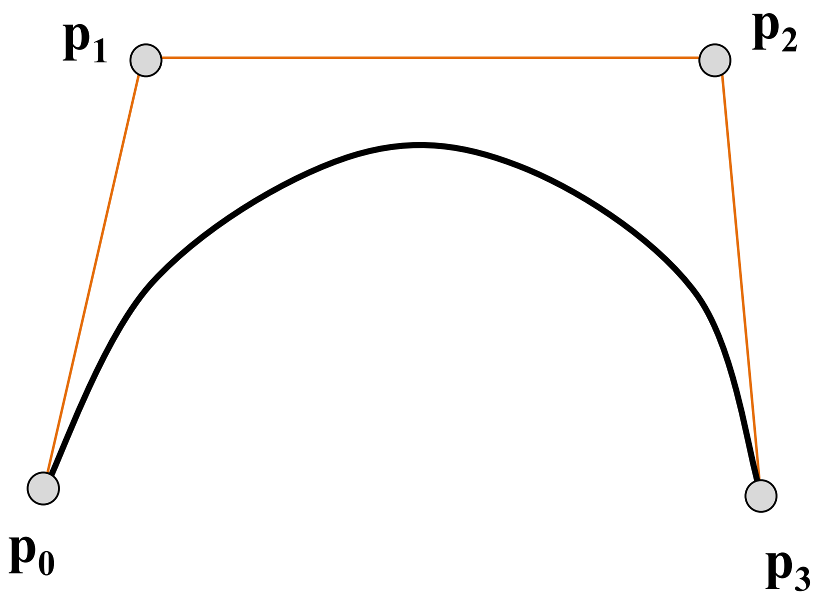


Bézier curves

- Next with 3rd degree

$$\mathbf{p}(t) = (1-t)^3\mathbf{p}_0 + 3t(1-t)^2\mathbf{p}_1 + 3(1-t)t^2\mathbf{p}_2 + t^3\mathbf{p}_3$$

4 control points (cubic Bézier)

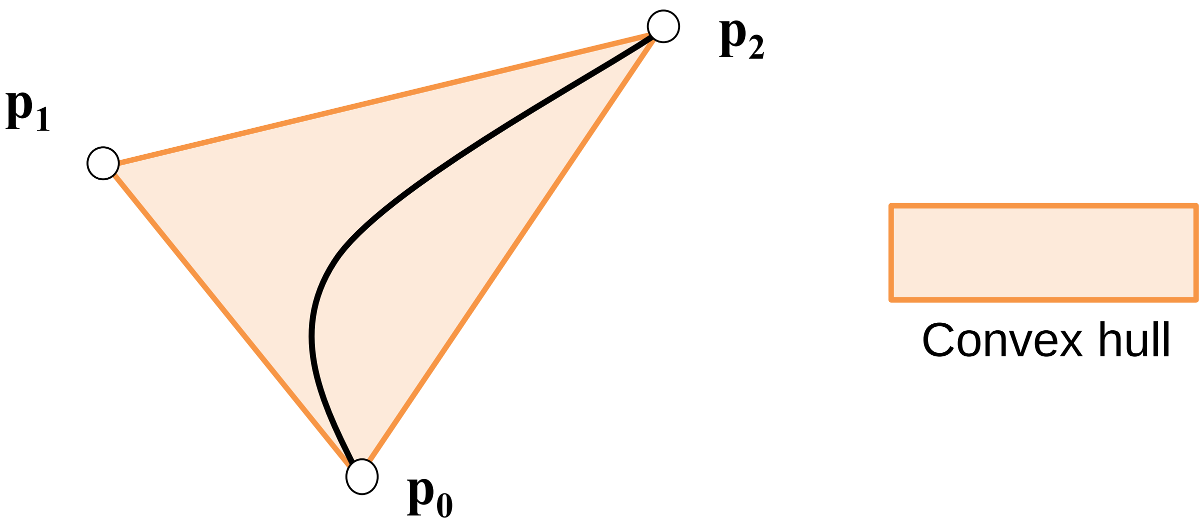


Properties of Bézier spline

1. The curve is bounded by the Convex hull given by the control points
2. An affine transformation of the control points is the same as an affine transformation of any points of the curve

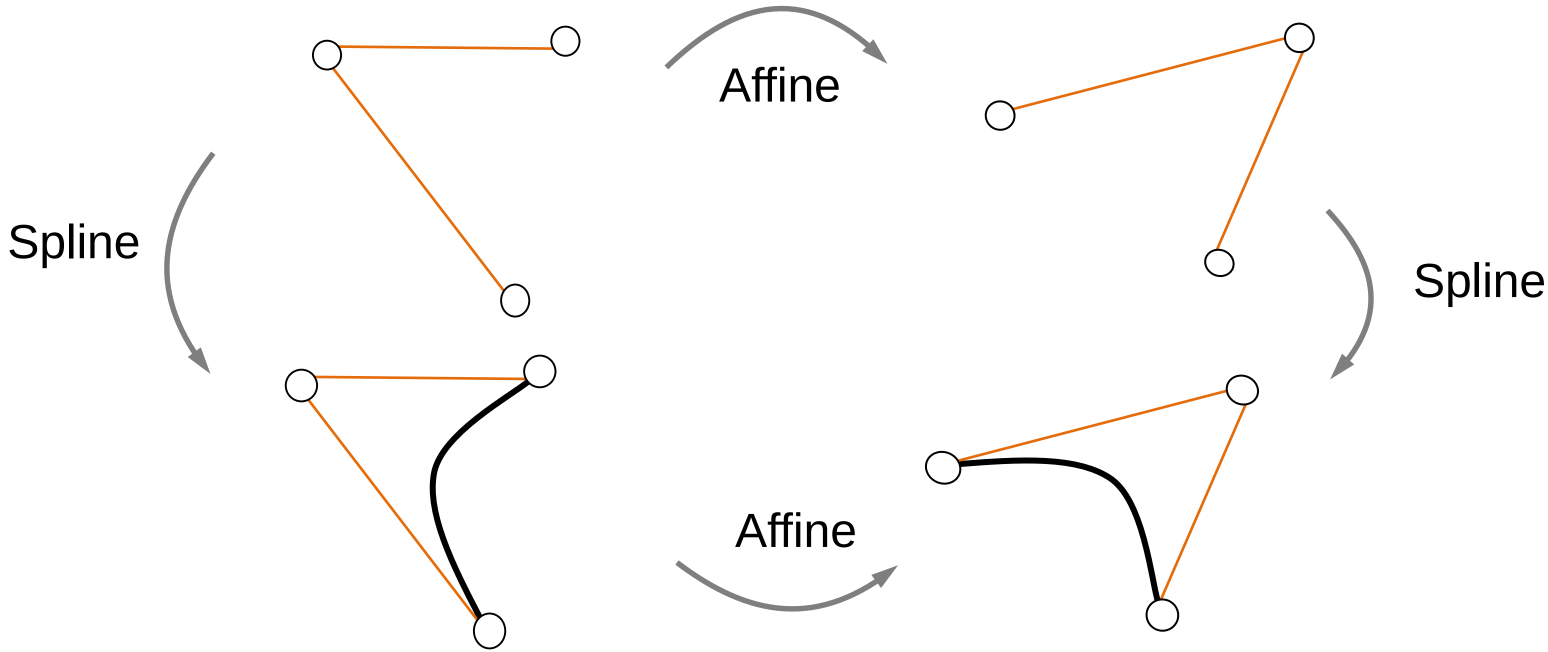
Property 1: Convex hull property

- A Bézier spline is completely contained in the convex hull of its control points



Property 2: Affine invariance

- An affine transformation of the control points is the same as an affine transformation of any points of the curve

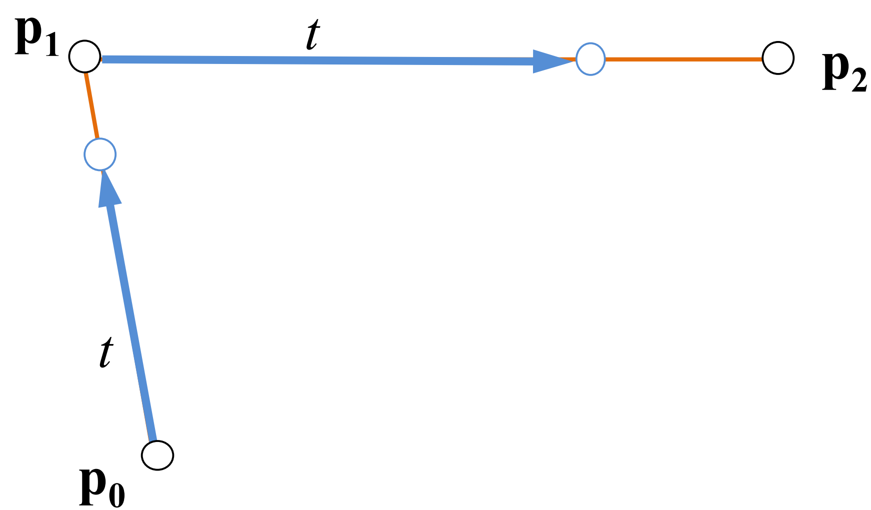


De Casteljau's Construction

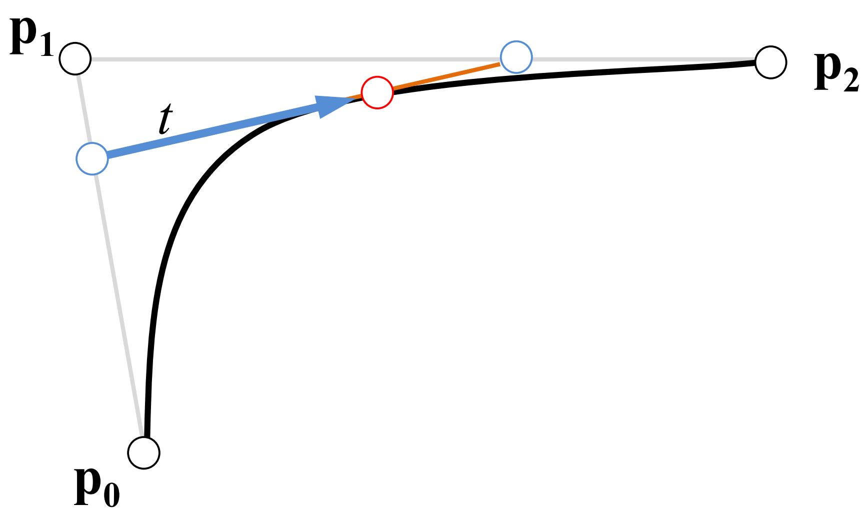
Multi-linear Interpolation

- Perform repeated linear interpolation:
  - Linearly interpolate each "edge" of the polyline from the n original points to get n-1 points
  - Keep doing this until you get a single point

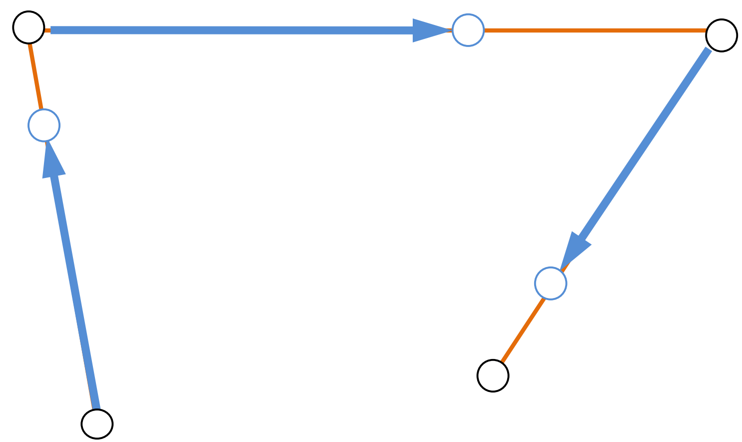
3 Control Points (quadratic Bézier)



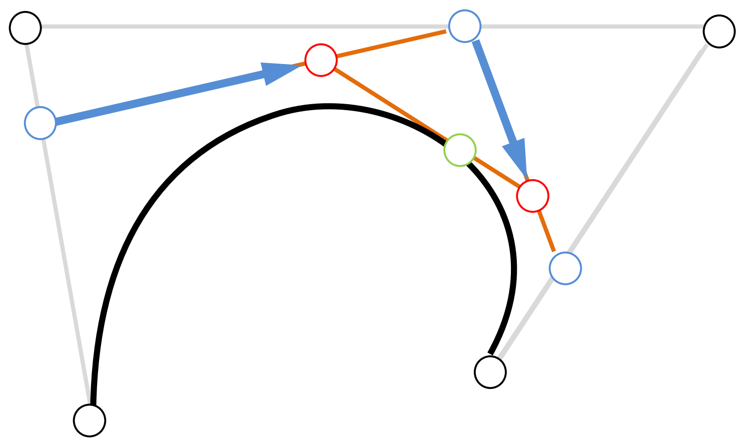
3 Control Points (quadratic Bézier)



4 Control Points (cubic Bézier)



4 Control Points (cubic Bézier)



Basis Functions

Basis form

- Consider the polynomial form:

$$\mathbf{p}(t) = \underbrace{(1-t)^3}_{B_0(t) = (1-t)^3}\mathbf{p}_0 + \underbrace{3t(1-t)^2}_{B_1(t) = 3t(1-t)^2}\mathbf{p}_1 + \underbrace{3(1-t)t^2}_{B_2(t) = 3t^2(1-t)}\mathbf{p}_2 + \underbrace{t^3}_{B_3(t) = t^3}\mathbf{p}_3$$

- We can re-written

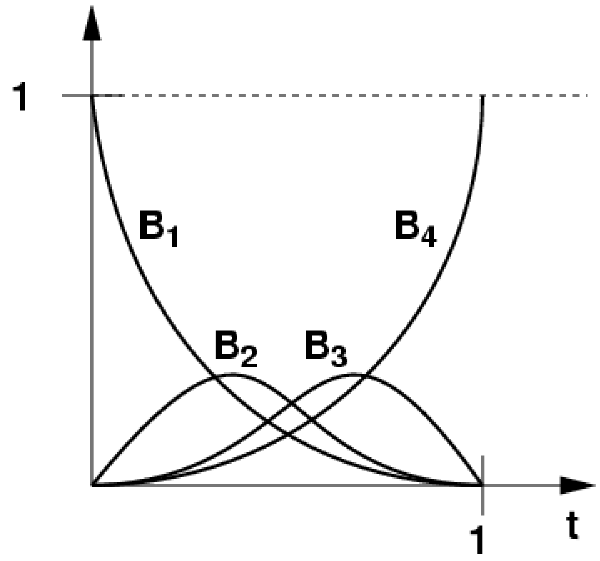
$$\mathbf{p}(t) = \sum \mathbf{p}_i B_i(t)$$

Bernstein basis

$$B_i(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

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

$$
\begin{aligned}
B_0(t) &= (1-t)^3 \\
B_1(t) &= 3t(1-t)^2 \\
B_2(t) &= 3t^2(1-t) \\
B_3(t) &= t^3
\end{aligned}
$$



Bernstein basis, matrix form

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

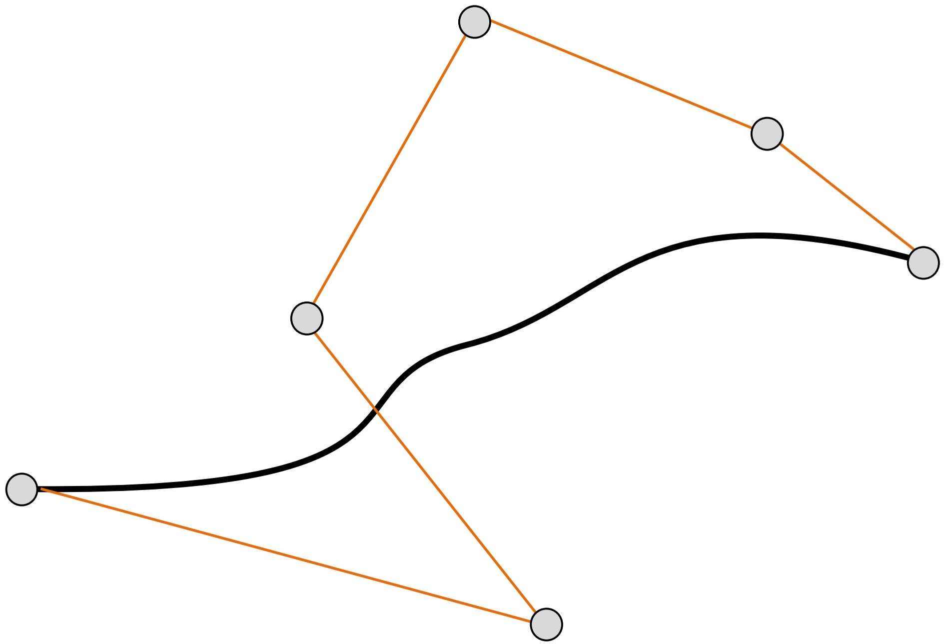
1×4×4×4×4×3=1×3

$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Longer curves

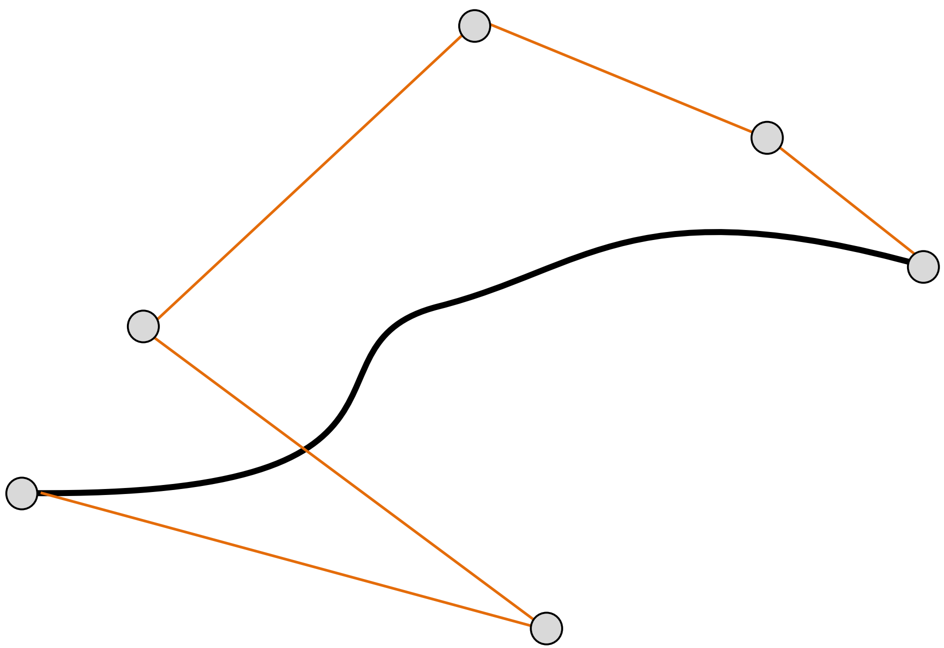
Lack of local control

- Becomes hard to control for many control points
- Changing one point, the entire curve changes



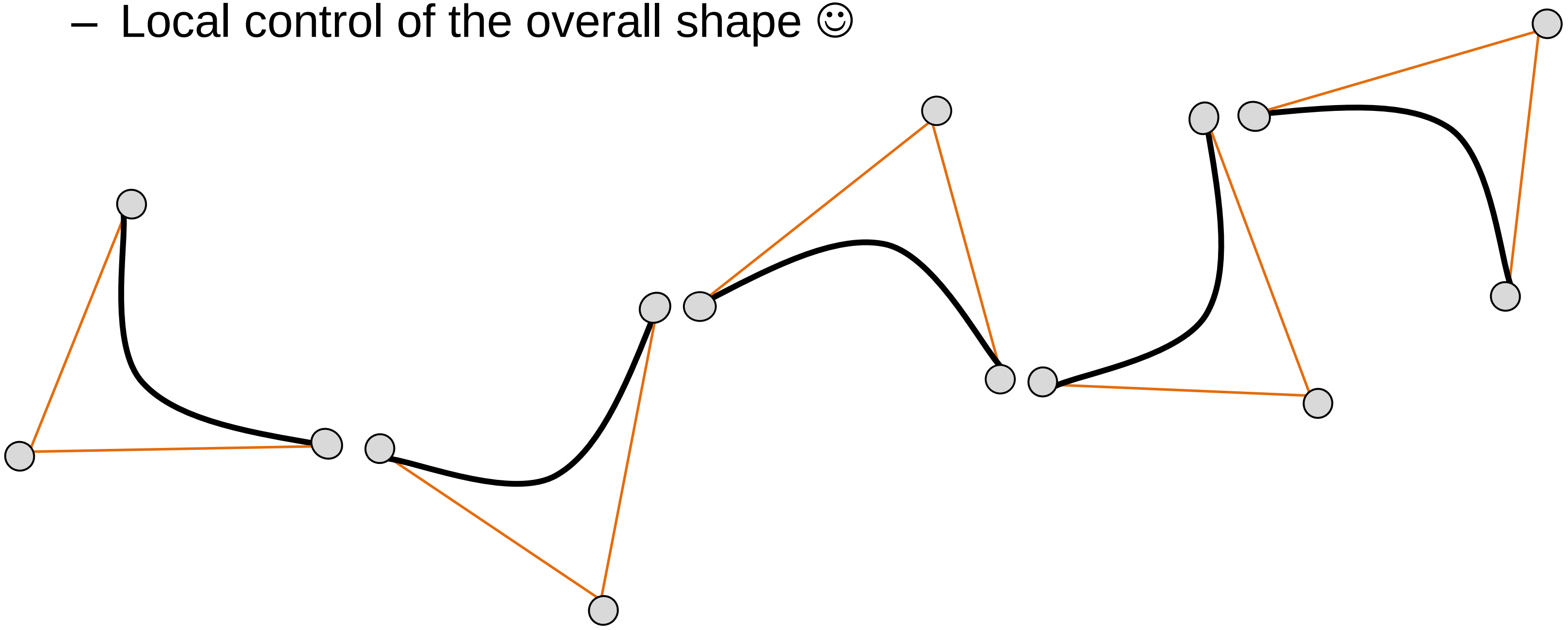
Lack of local control

- Becomes hard to control for many control points
- Changing one point, the entire curve changes



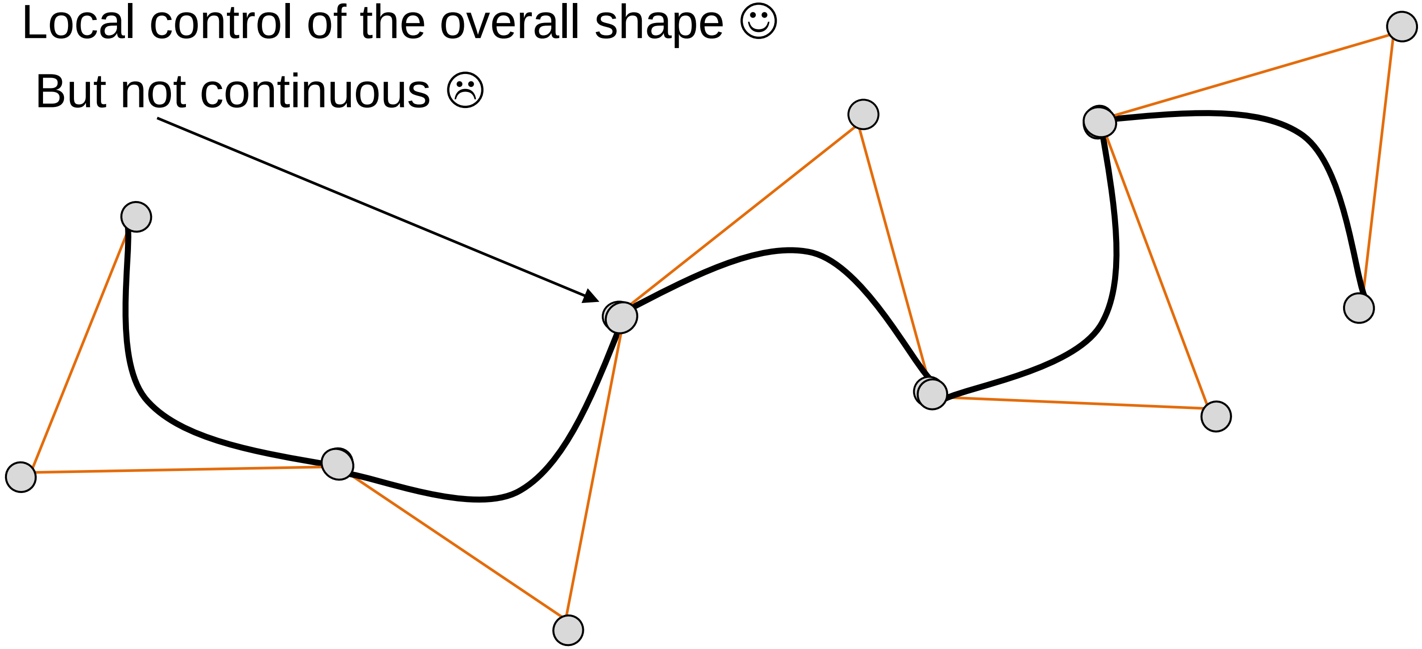
Joining Bézier curves

- Better to join curves than raise the number of controls points
  - Avoid numerical instability ☺
  - Local control of the overall shape ☺

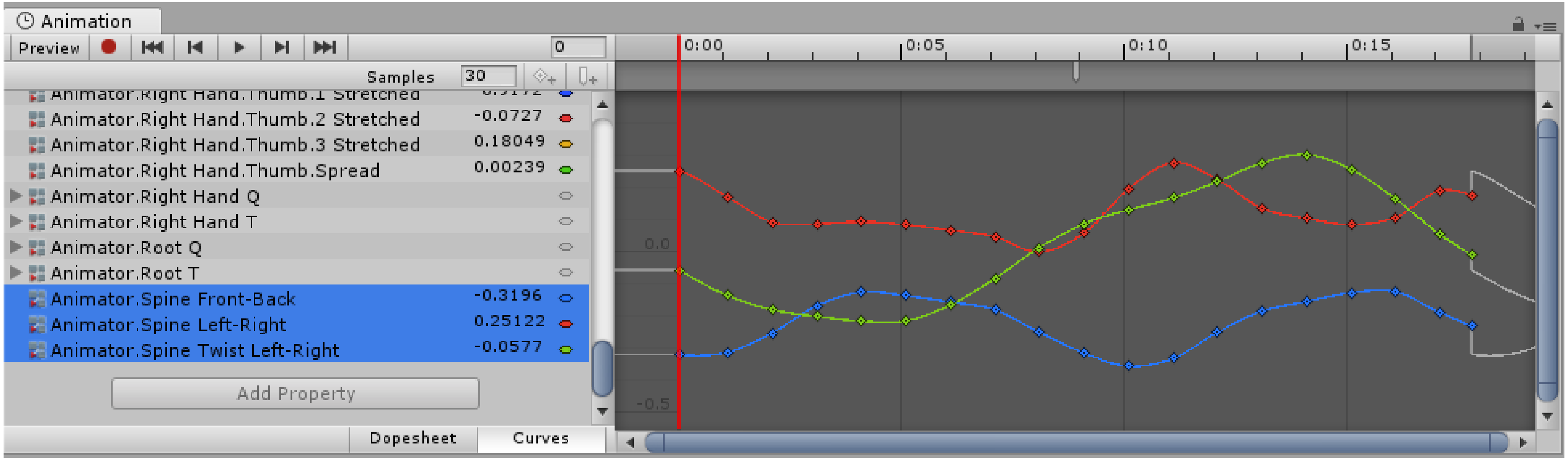


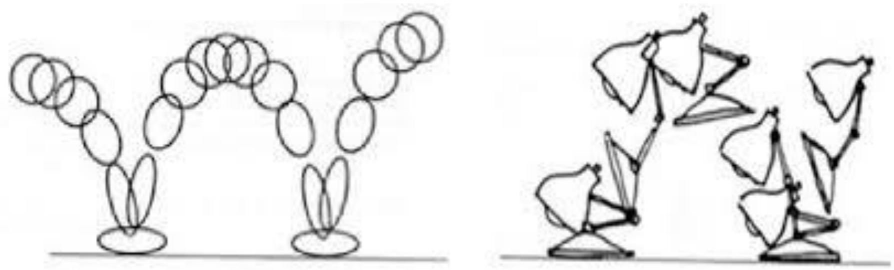
Joining Bézier curves

- Better to join curves than raise the number of controls points
  - Avoid numerical instability ☺
  - Local control of the overall shape ☺
  - But not continuous ☹



Application: Animation curves



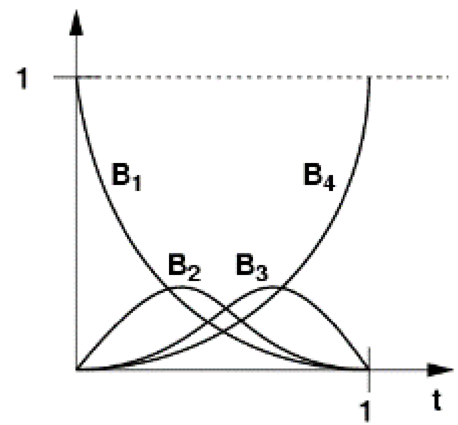


Conclusions

- It is possible to define and draw a curve with a discrete representation
- All is needed are control points and interpolation strategy
- We have seen Bézier curves
  - As polynomials
  - From the De Casteljau construction
  - From the Bernstein basis

B-Splines

Recall: Bernstein basis



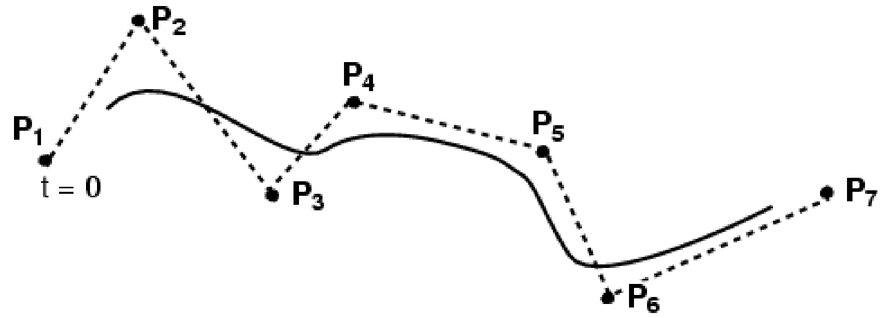
$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Other Basis Functions?

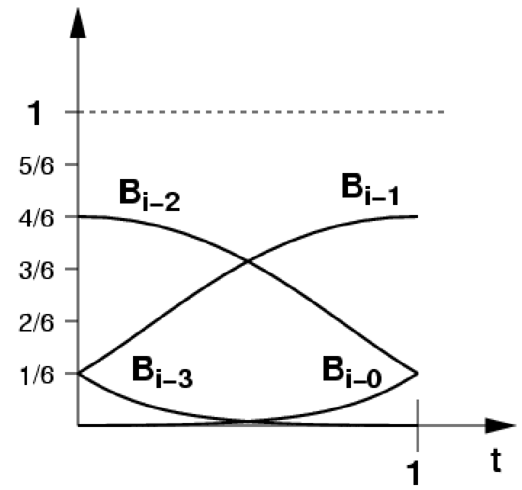
- Could have a different set of basis functions
- Desirable properties
  - Being able to set gradient at end points
  - Being able to set degree of control over each control point, etc.

B-Splines

- Polynomial curves
- $C^{k\text{-}1}$ continuity
  - Cubic B-spline: C^2 continuity
- Main properties:
  - Generalisation of Bézier curves
  - For Cubic B-spline, each four control points control a segment of the spline

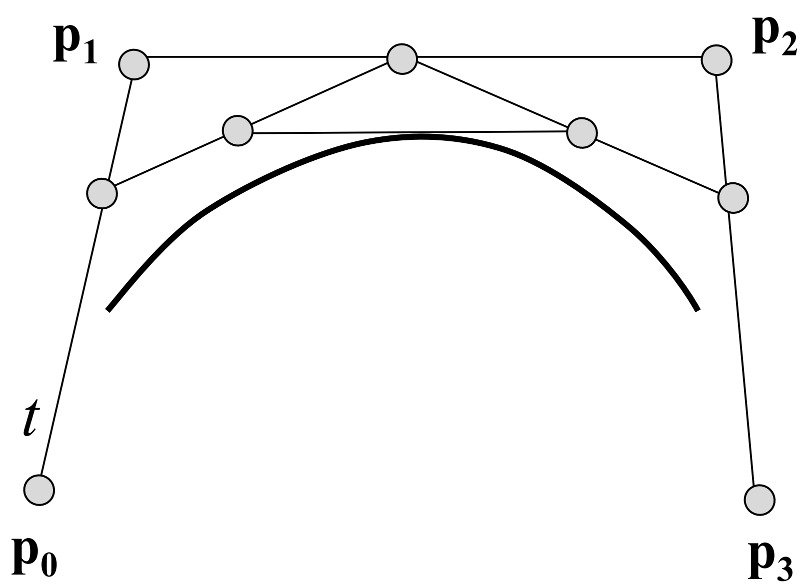


B-Spline Basis Functions



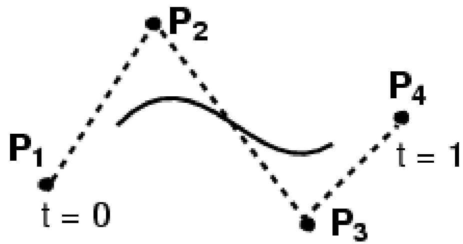
$$\mathbf{p}(t) = \mathsf{TM}\mathbf{p} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

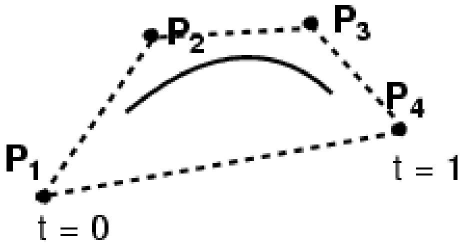
4 control points (cubic B-Spline)

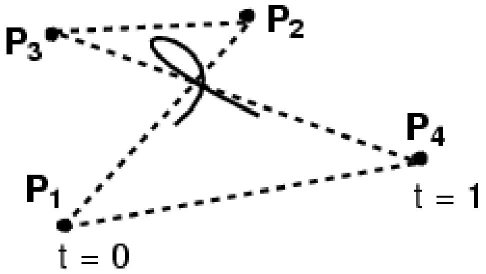


Cubic B-Spline Example

- Curve is **not** constrained to pass through any control points

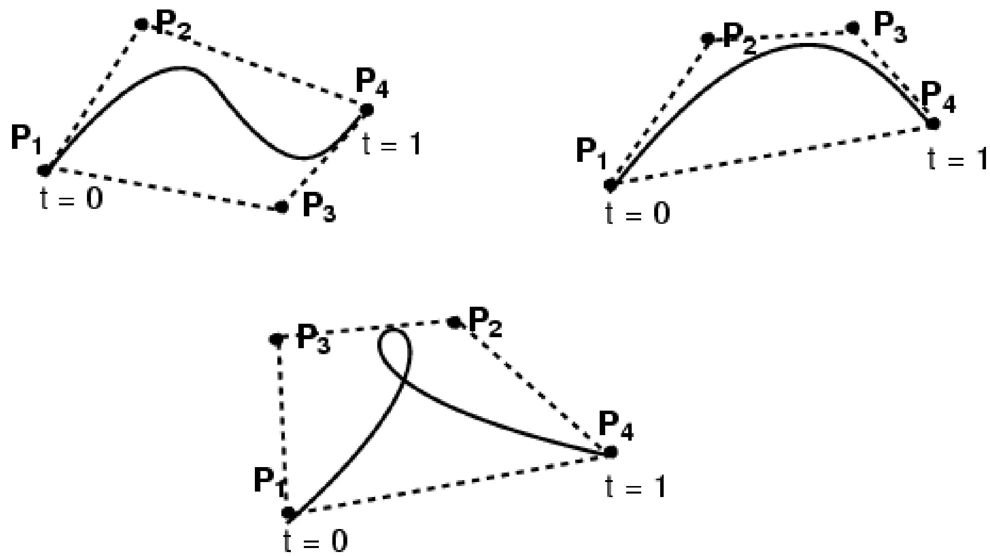






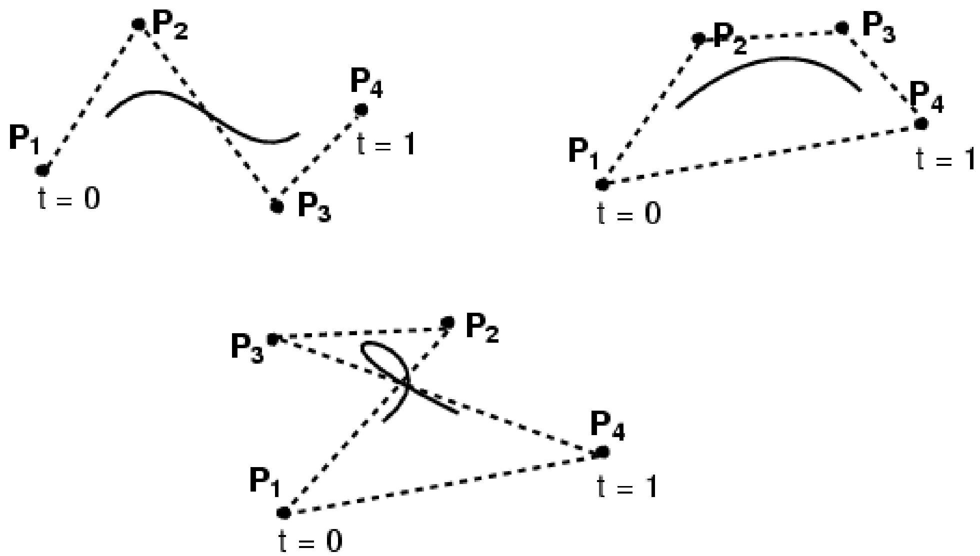
B-Spline curves are also bounded by the convex hull of their control points

Bézier compared to B-Spline



Bézier

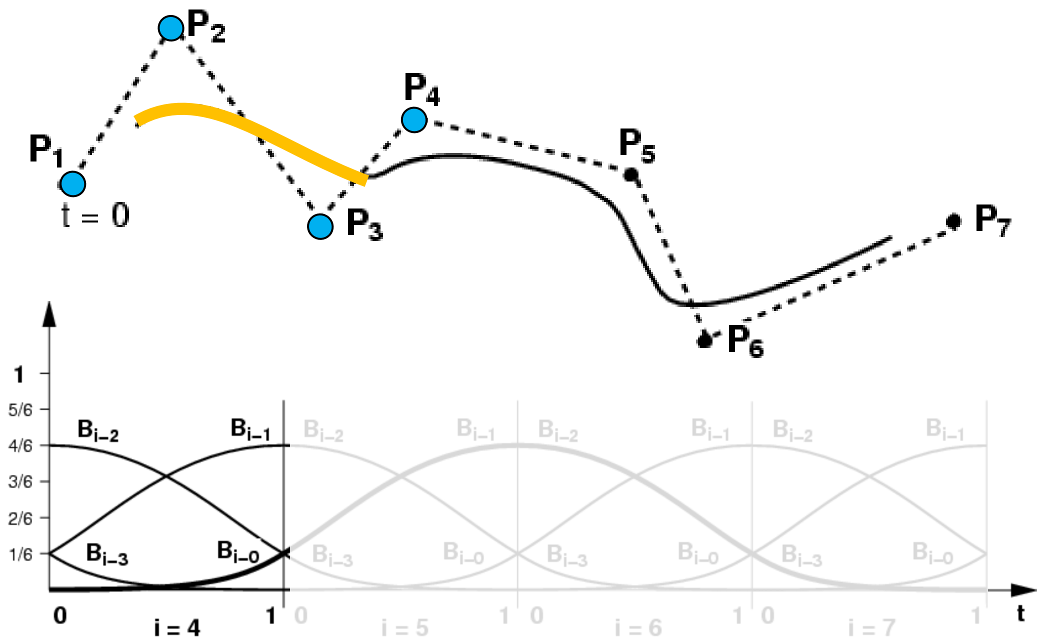
Bézier compared to B-Spline



B-spline

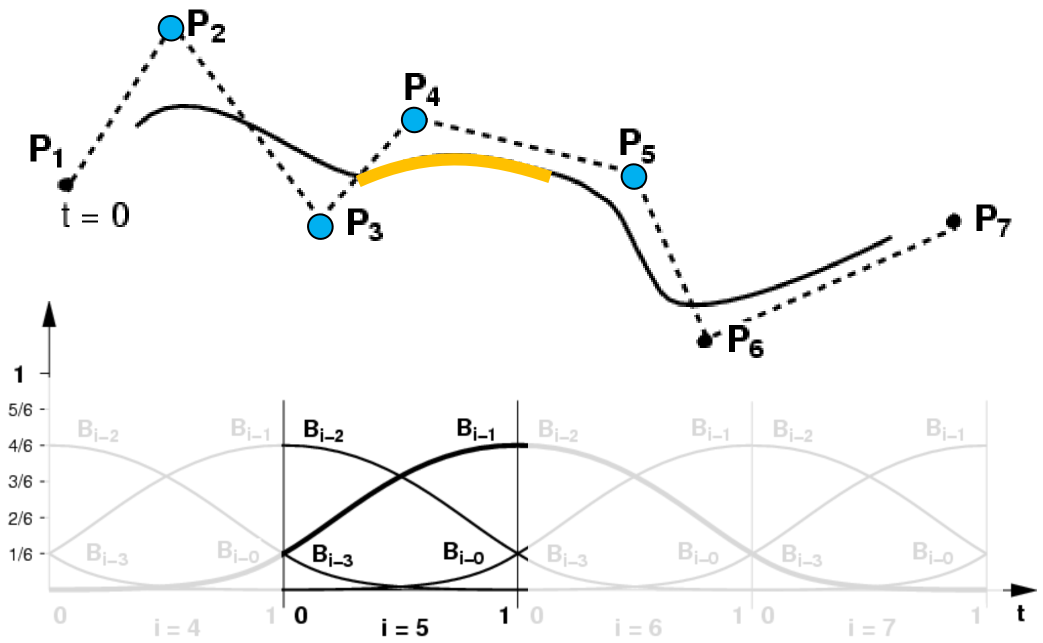
Knot vector

- Chained together using a **knot sequence**



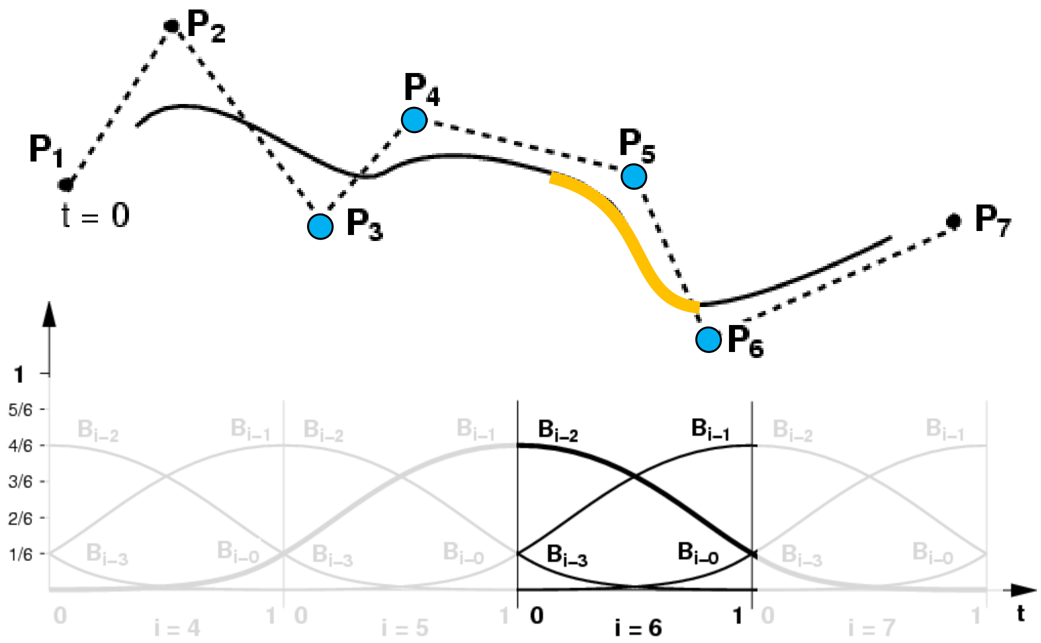
Knot vector

- Chained together using a **knot sequence**



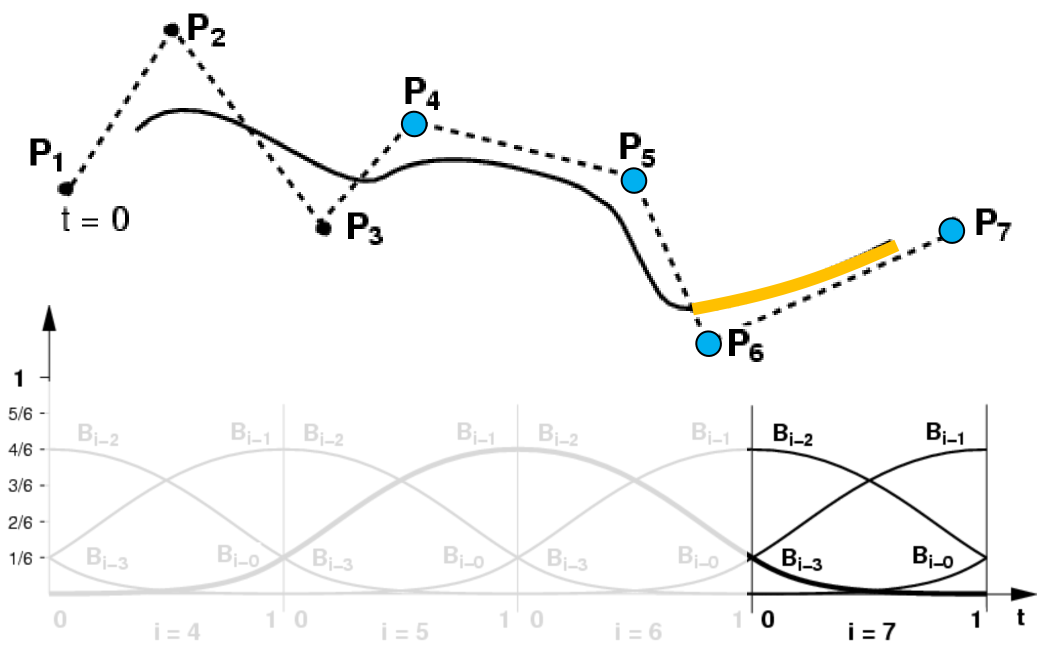
Knot vector

- Chained together using a **knot sequence**



Knot vector

- Chained together using a **knot sequence**



Cox-de Boor

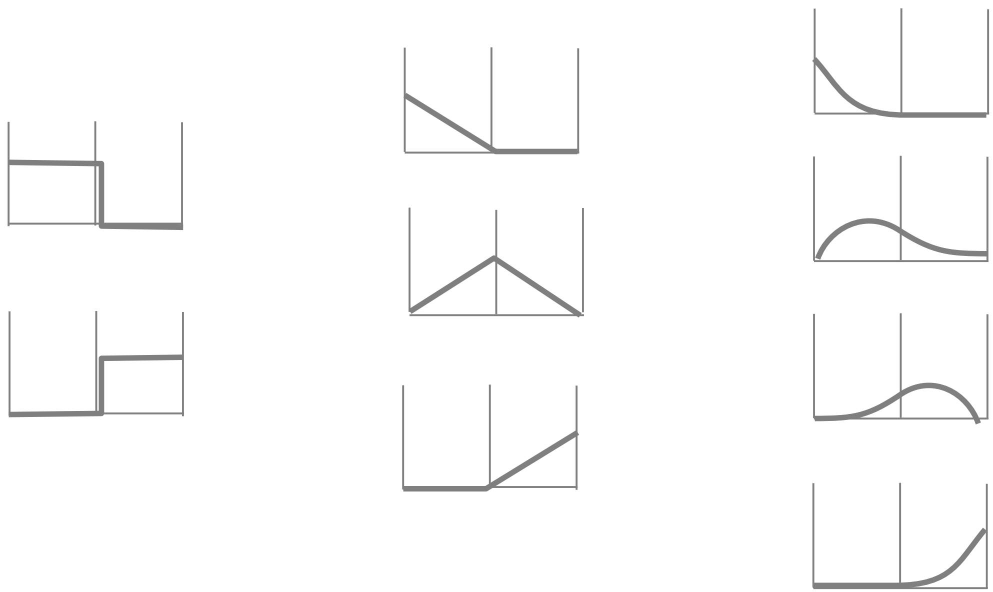
- Recursive definition of B-spline basis

$$B_{i,1}(t) = \begin{cases} 1 \text{ if } t_i < t < t_{i+1} \\ 0 \end{cases}$$

$$B_{i,k+1}(t) = w_{i,k} B_{i,k} + (1 - w_{i+1,k}) B_{i+1,k}$$

$$w_{i,k}(t) = \begin{cases} \frac{t_i - t}{t_{i+k} - t_i} \text{ if } t_{i+k} \neq t \\ 0 \end{cases}$$

Basis functions



$B_{i,\,1}$ $B_{i,\,2}$ $B_{i,\,3}$

Advantages of B-Splines

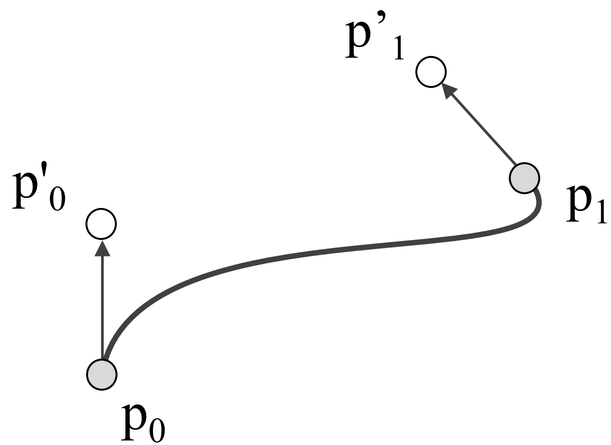
- Convex hull based on m control points is smaller than for Bézier curve
- Better local control
- The control points give a better idea of the shape of the curve

Other Splines

Hermite Splines

- Instead of
  - setting control points as positions
  - mix positions and directions
- Properties:
  - Very common in editing packages (PowerPoint)
  - Easily converted into Bezier curves

Hermite Splines

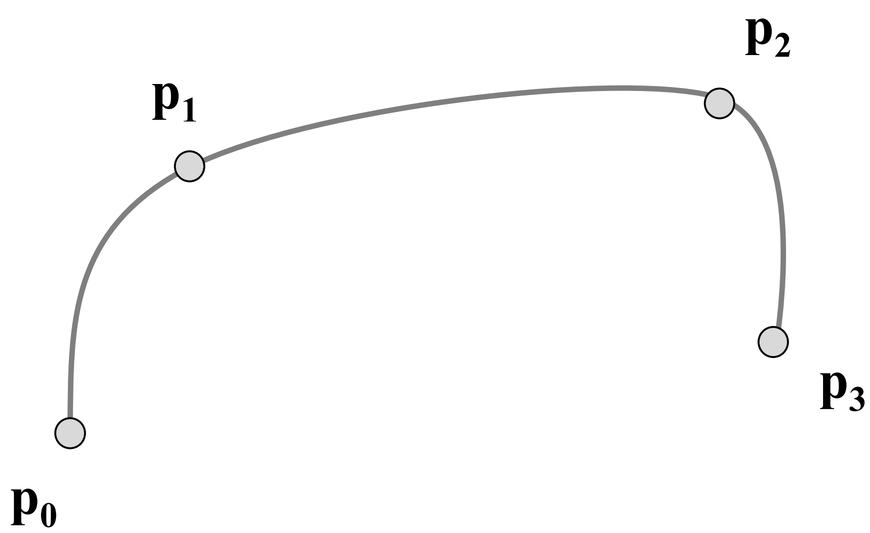


$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix}$$

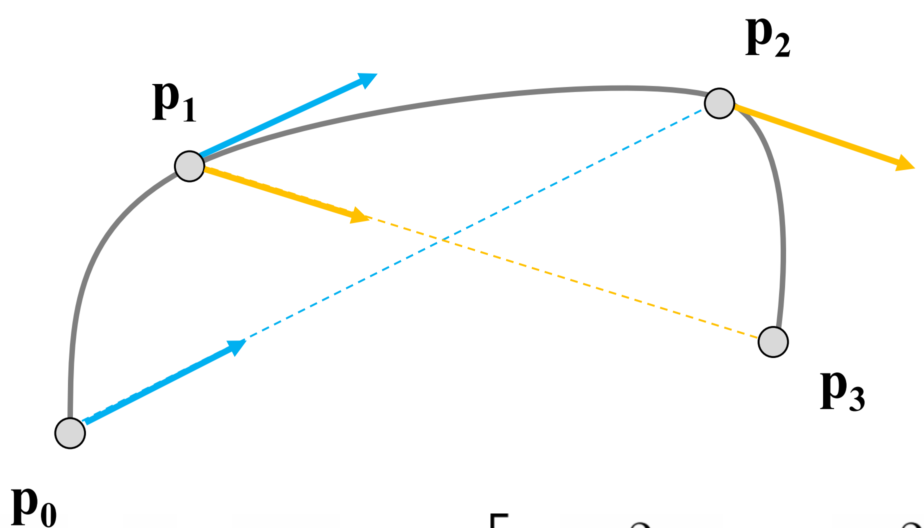
Catmull-Rom Splines

- Interpolates all the points
- However, gradient set only by the previous and next point
- Thus highly constrained, but again, cubic formulation

Catmull-Rom Splines



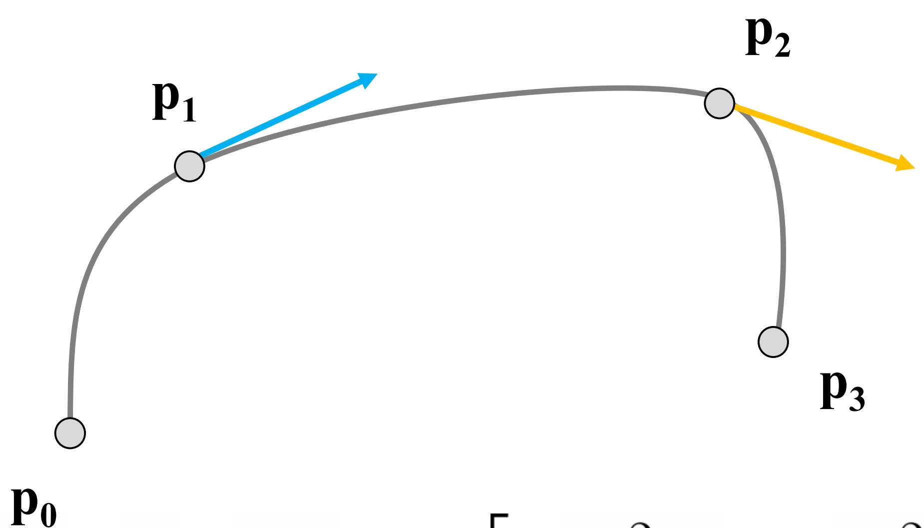
Catmull-Rom Splines



s typically 0.5

$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -3 \\ -s & 0 & s & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

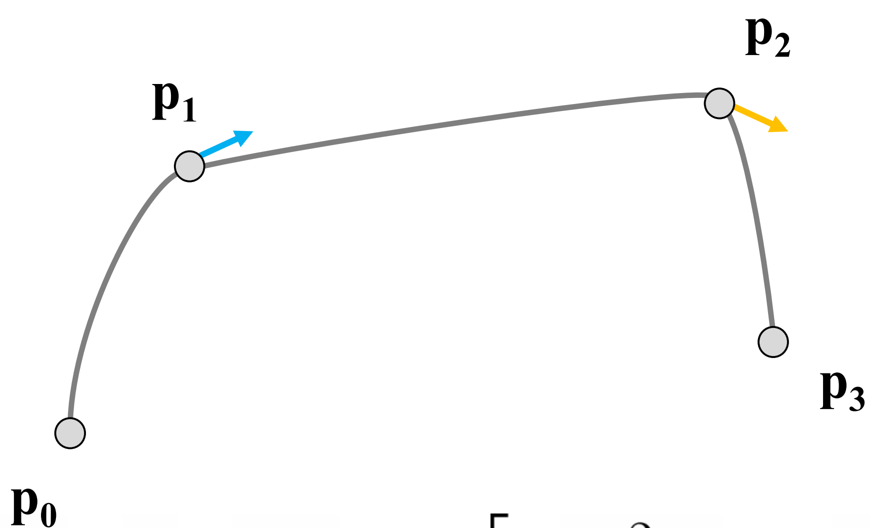
Catmull-Rom Splines



$s = 0.5$

$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -3 \\ -s & 0 & s & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Catmull-Rom Splines



$s = 0.1$

$$\mathbf{p}(t) = \mathsf{TMP} = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -3 \\ -s & 0 & s & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Summary

- B-Splines, Hermite splines and Catmull-Rom splines all have their uses with slightly different capabilities
- Relatively easy to convert between the forms (can be difficult to tell which your user interface uses!)