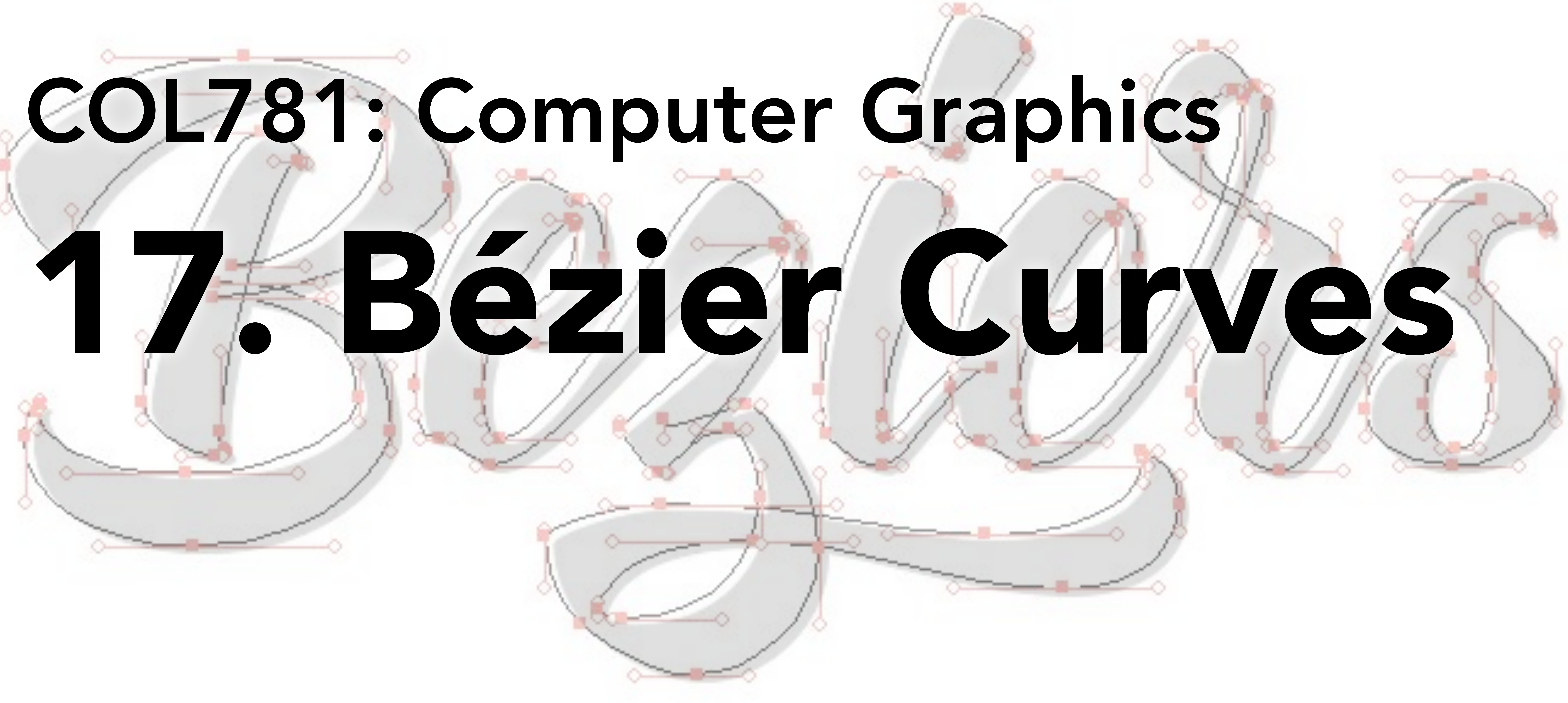
COL781: Computer Graphics

17. Bézier Curves

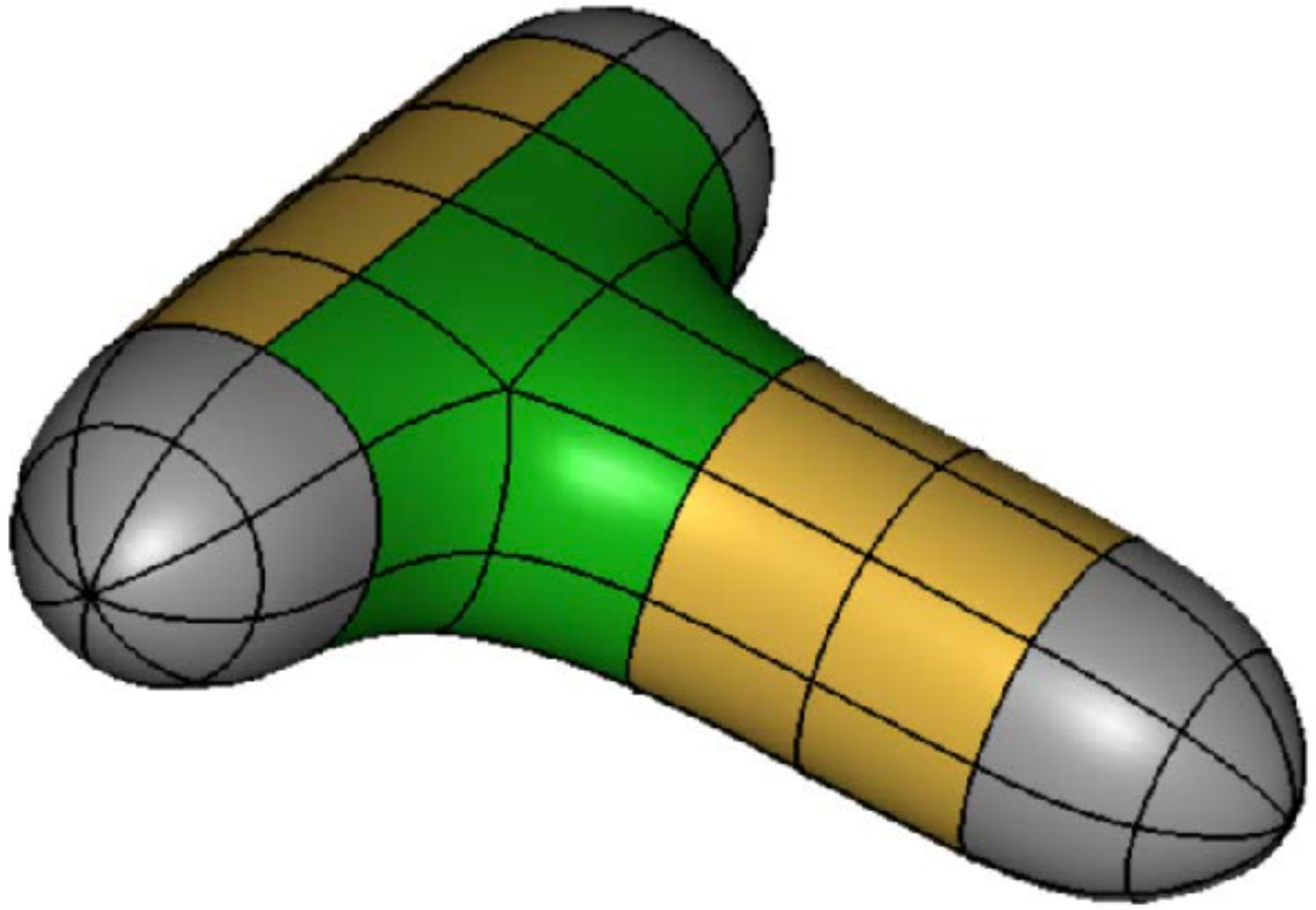


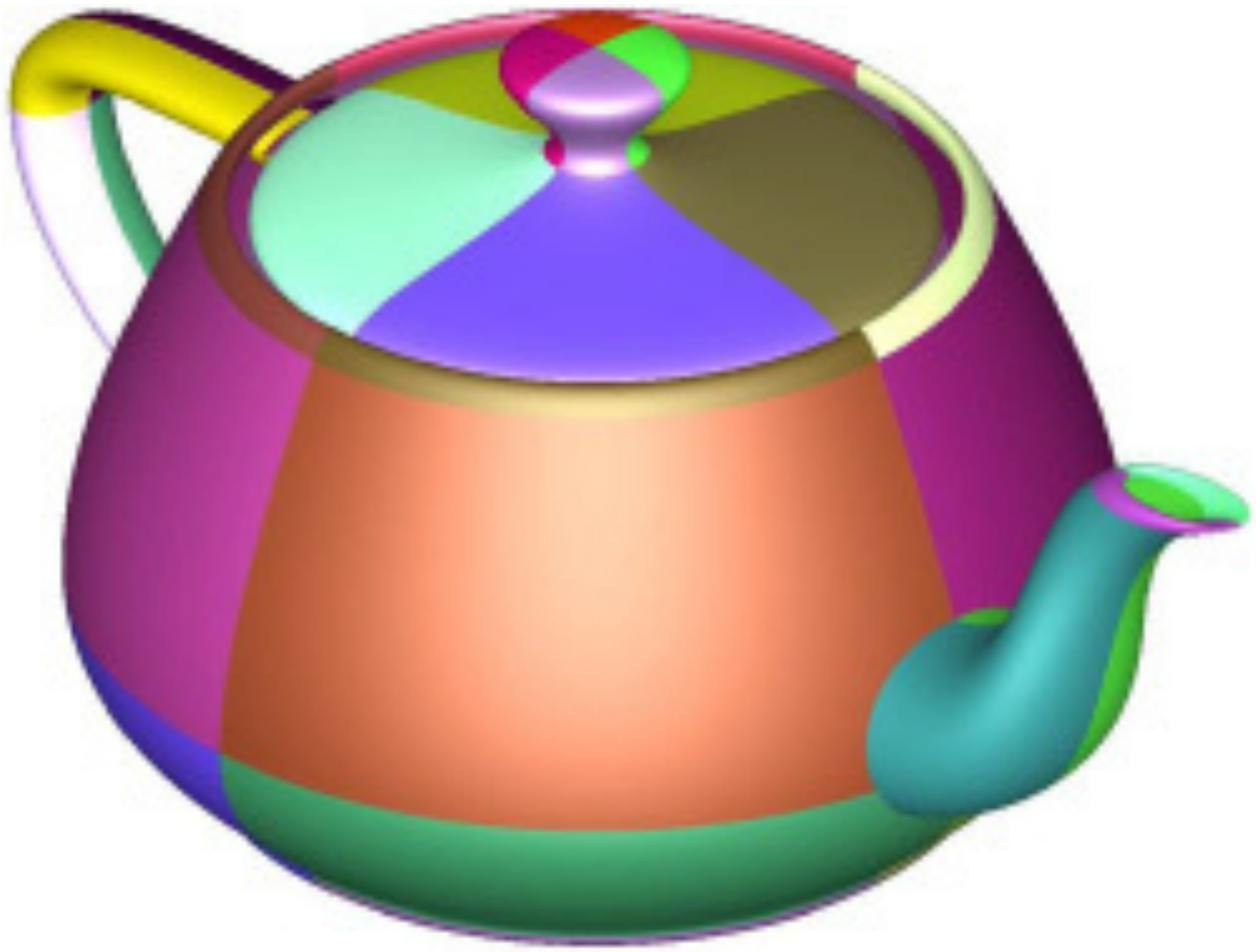
Dave Coleman

Modeling curved shapes

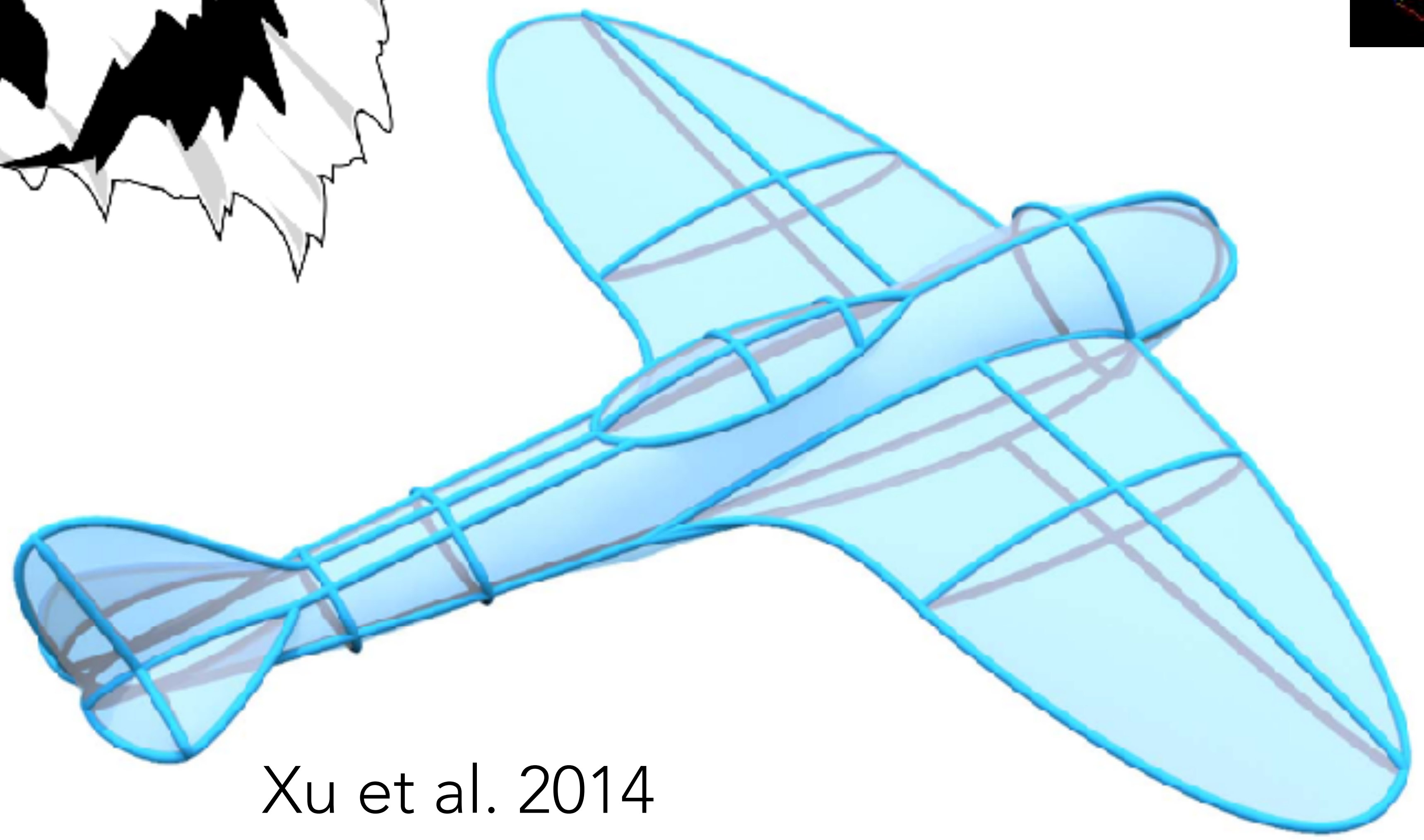
With meshes we can only approximate smooth shapes, by adding lots and lots of vertices and triangles.

How can we directly model a smooth shape?

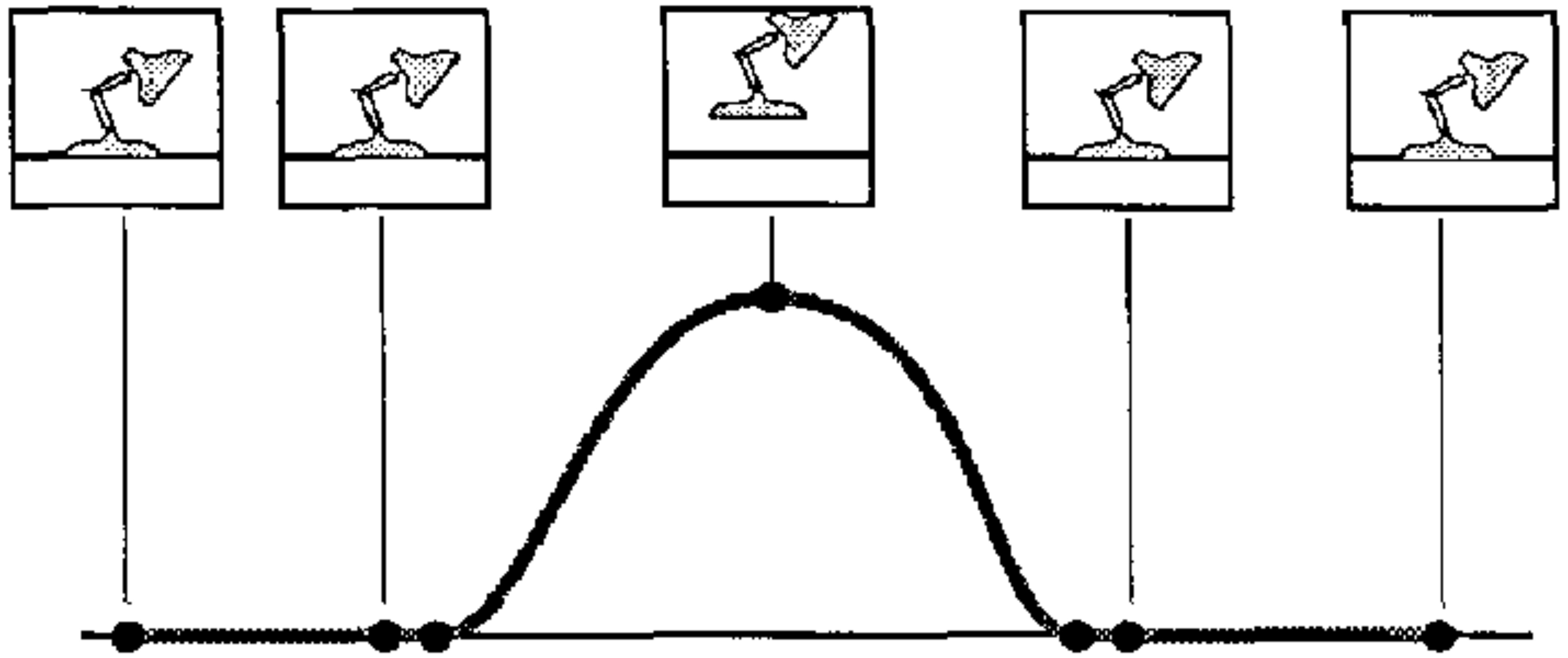




Curves

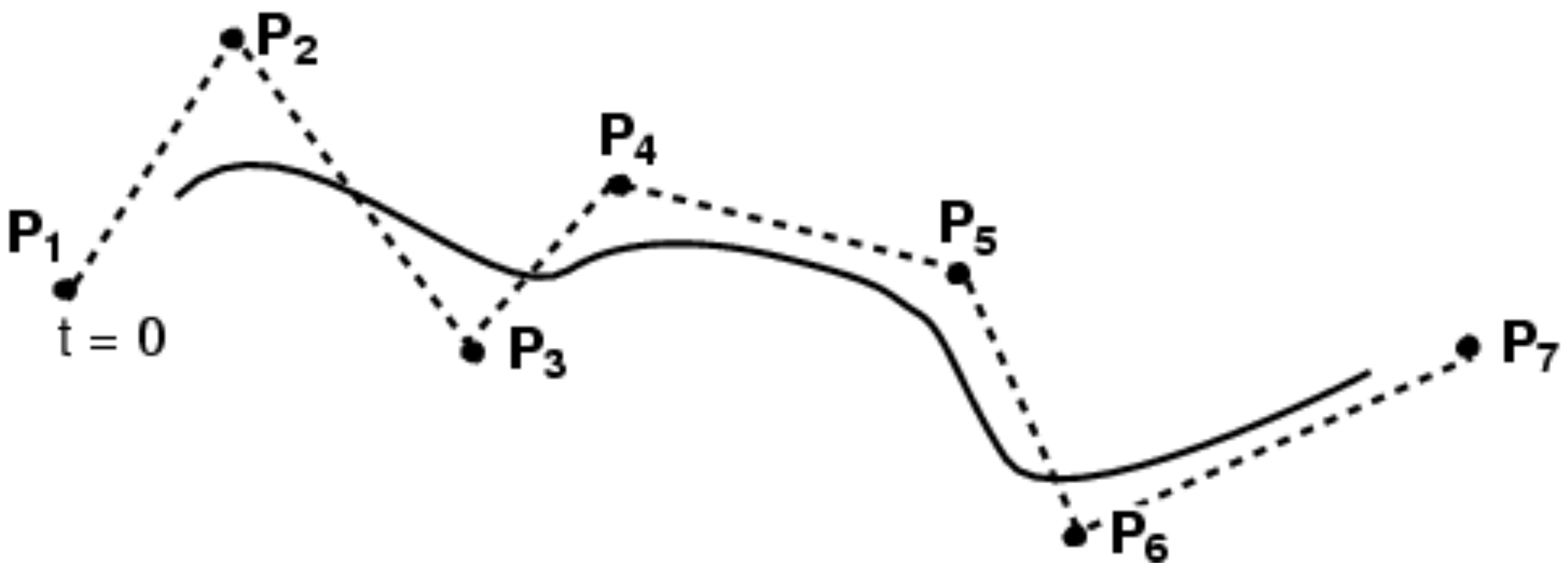


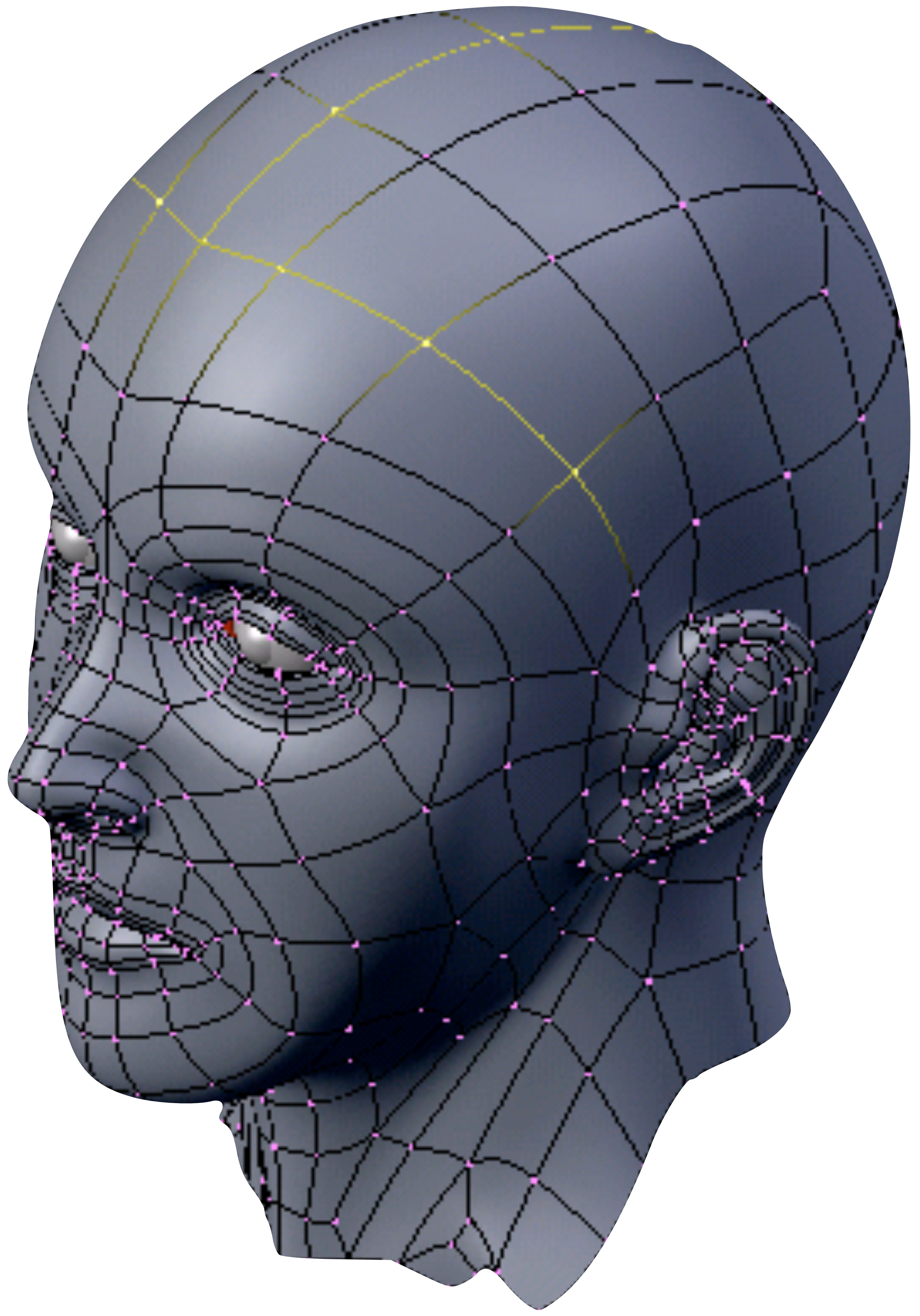
Xu et al. 2014



Lasseter 1987

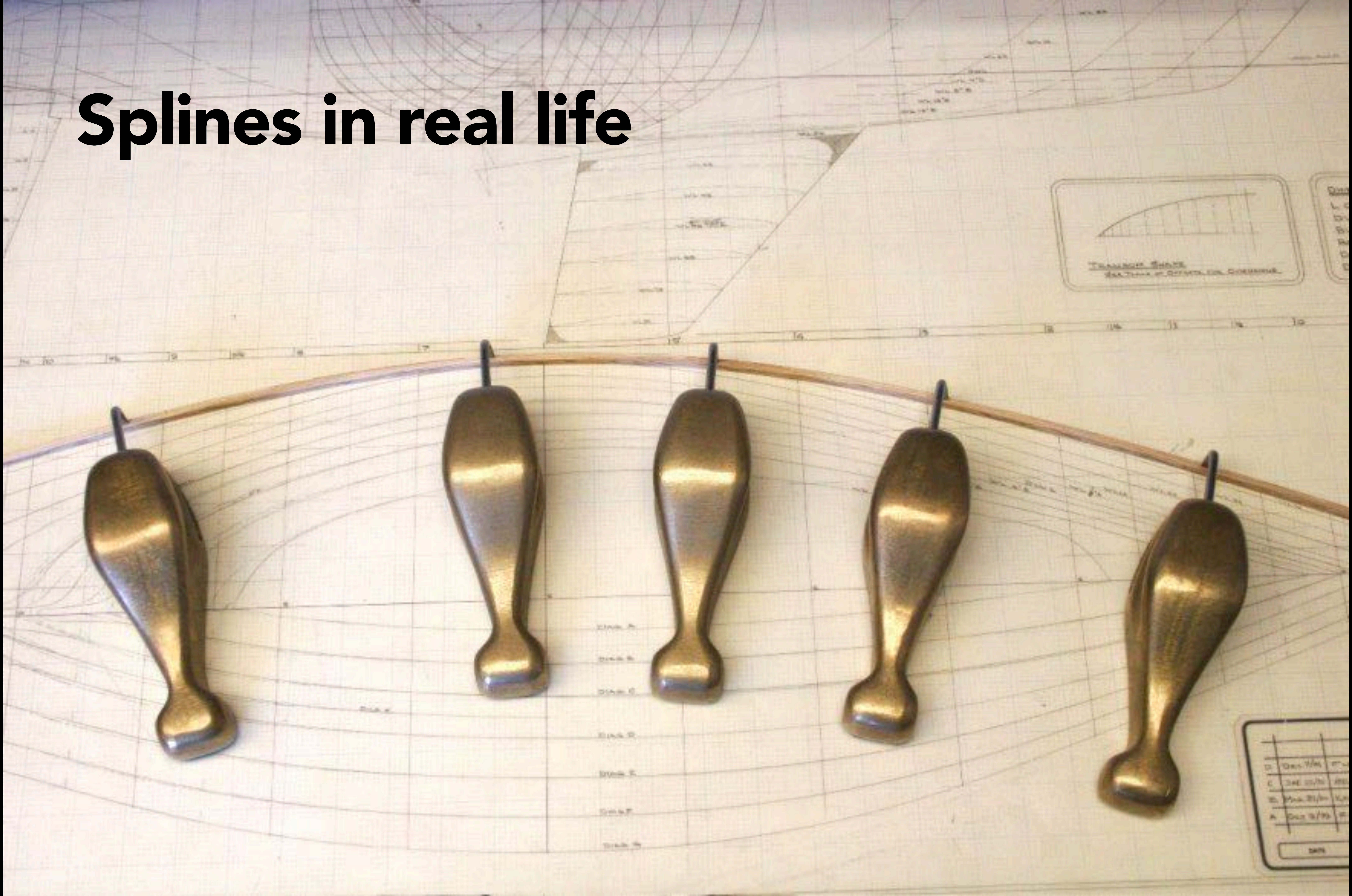
We will retain the same "interface" as polylines: user specifies a sequence of points. Now we want to define a smooth curve based on them.



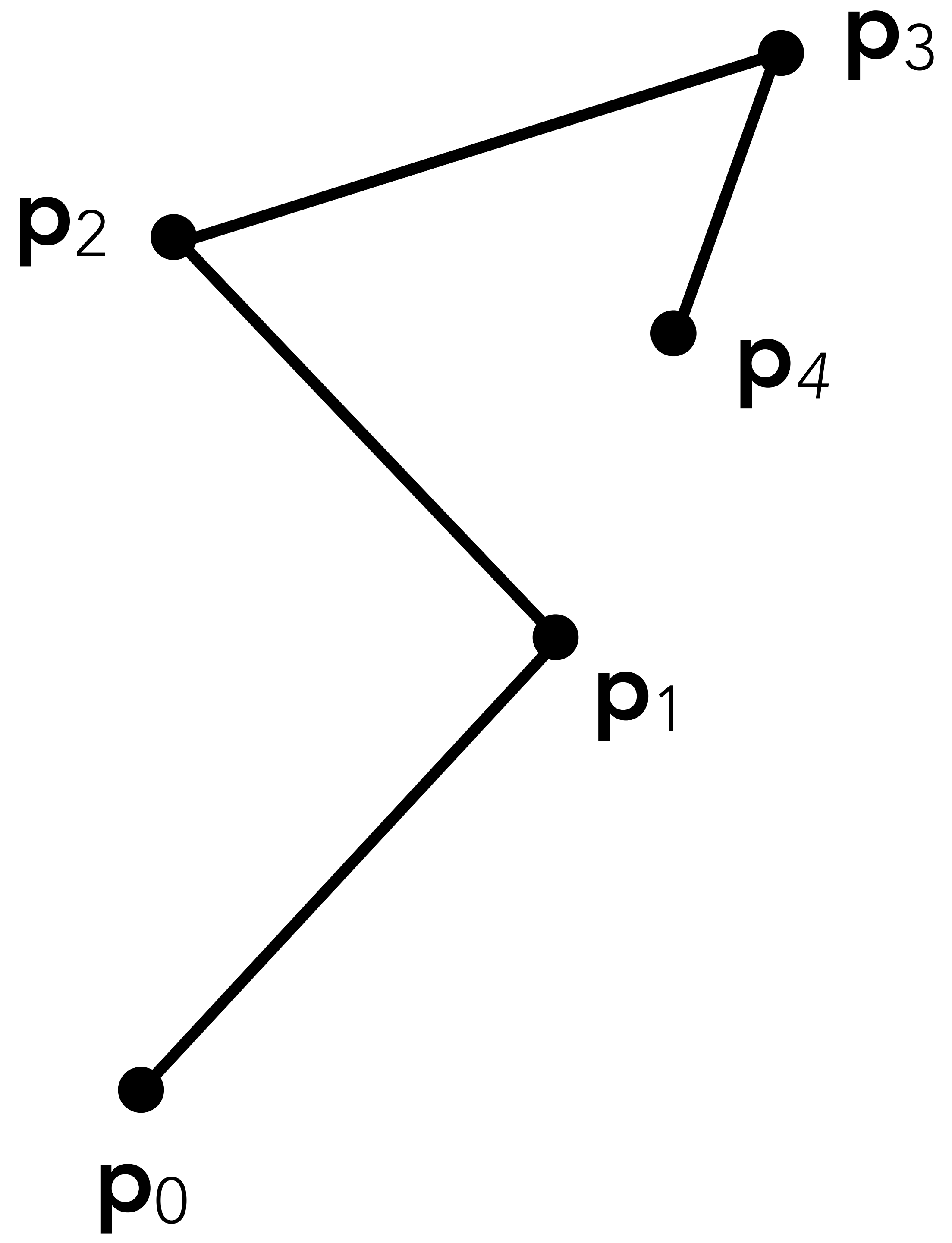


Usually define parametrically: $x(t)$, $y(t)$ where
x, y are piecewise polynomial functions a.k.a. **splines**

Splines in real life



Polylines as parametric curves



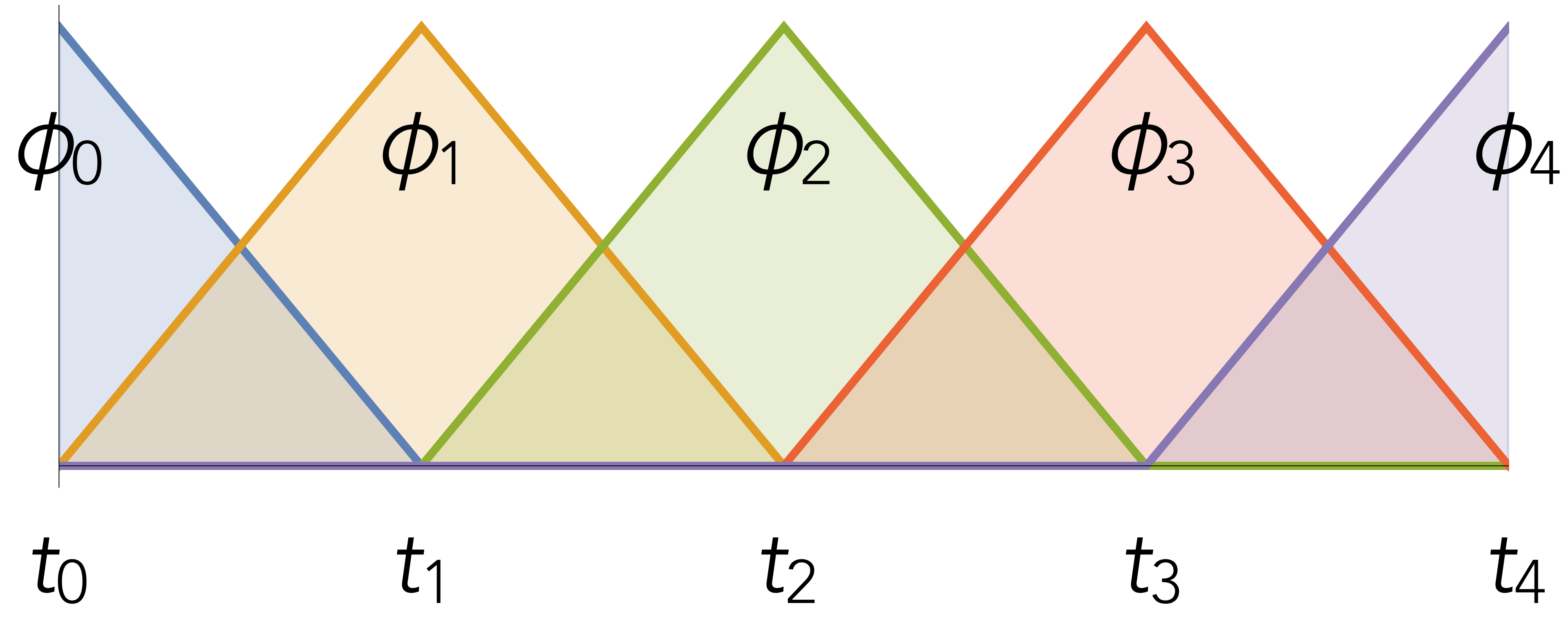
Given $n+1$ vertices $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$, define

$$\mathbf{p}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i}\,\mathbf{p}_i + \frac{t - t_i}{t_{i+1} - t_i}\,\mathbf{p}_{i+1}$$

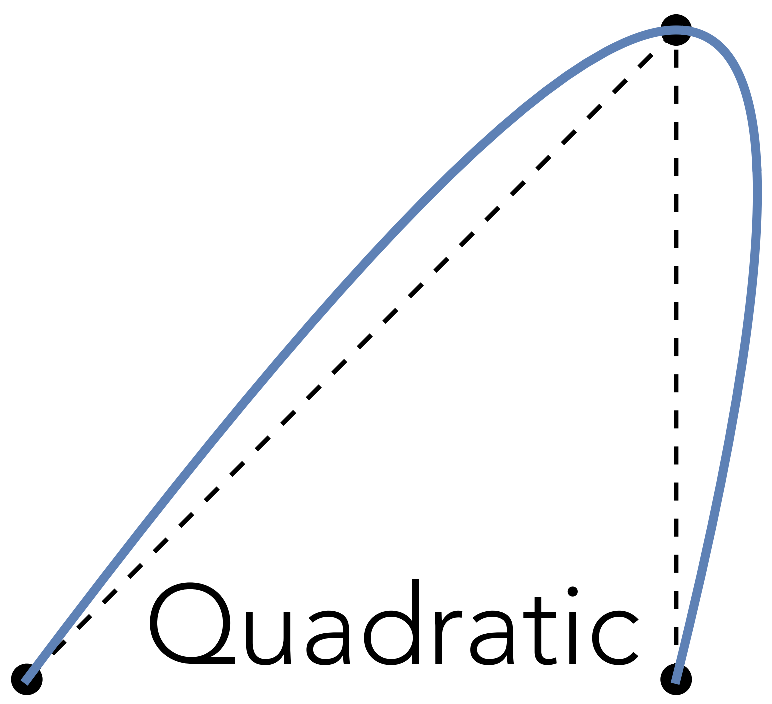
when $t_i \leq t < t_{i+1}$.

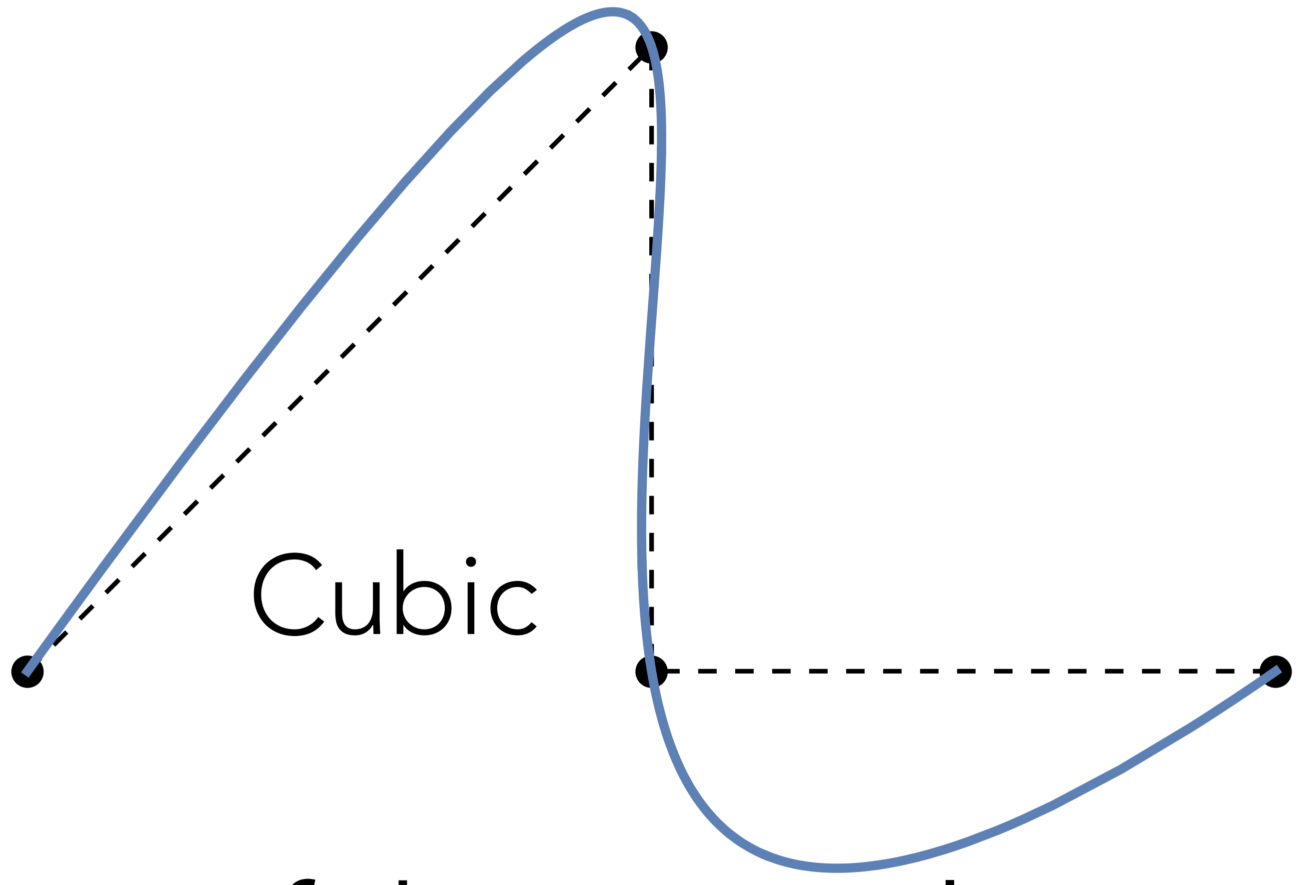
Equivalently, $\mathbf{p}(t) = \sum \phi_i(t)\,\mathbf{p}_i$

Influence of each vertex:



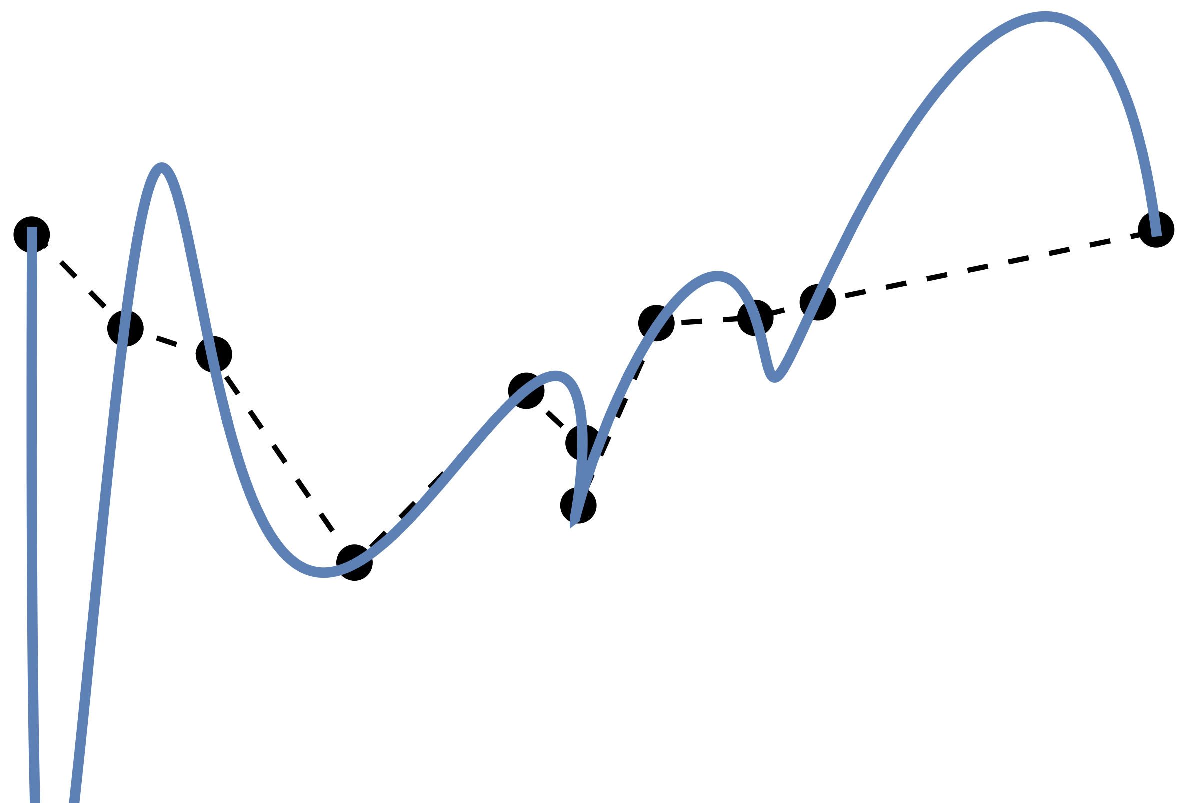
You all probably already know one way to fit a smooth function through multiple points: polynomial interpolation.





Hard to control: curve goes beyond the range of the control points

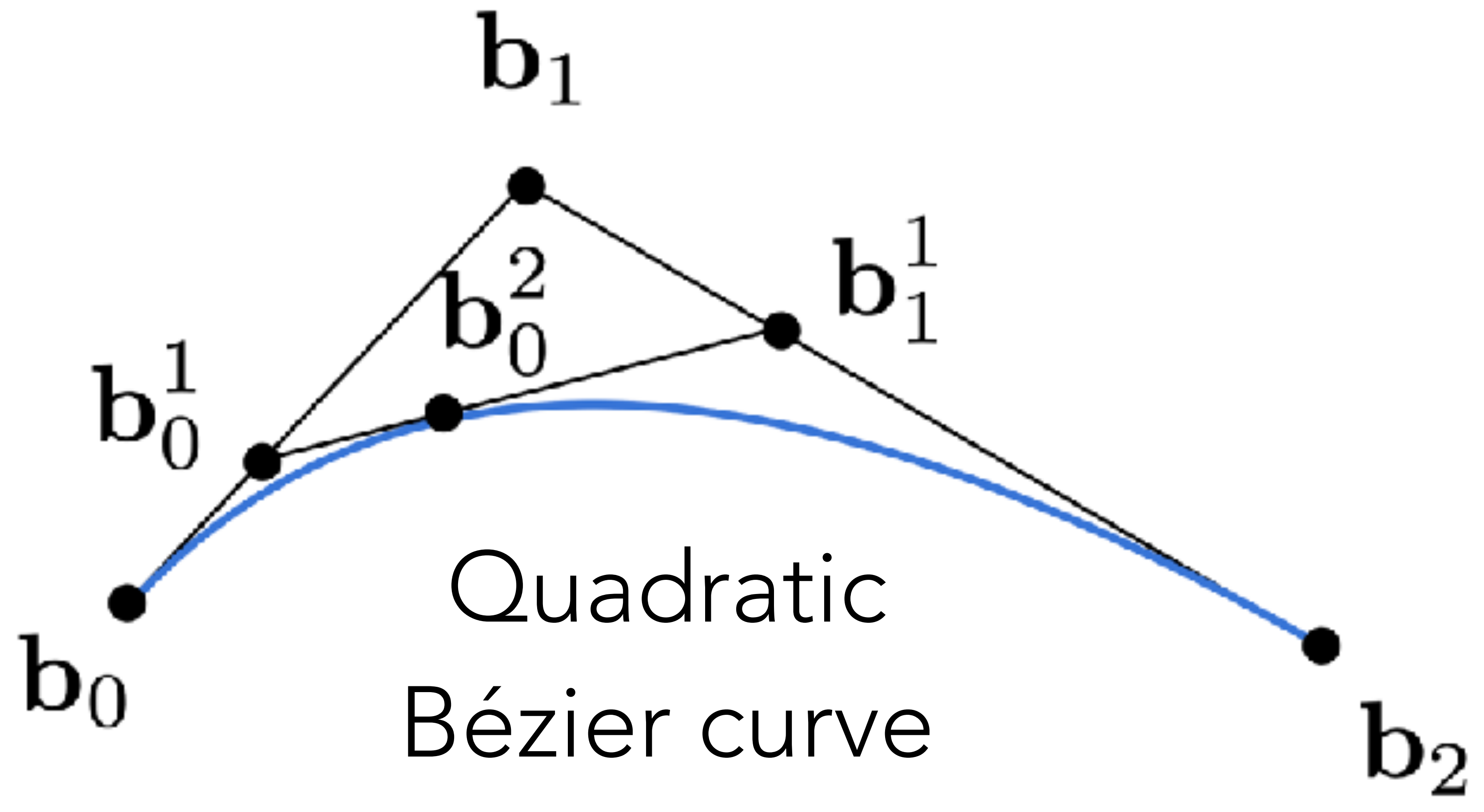
Very unstable for higher degrees!



Bézier curves

How can we guarantee the curve stays within the range of the control points?

Construct the curve by recursive interpolation: **de Casteljau's algorithm** a.k.a. "corner cutting"



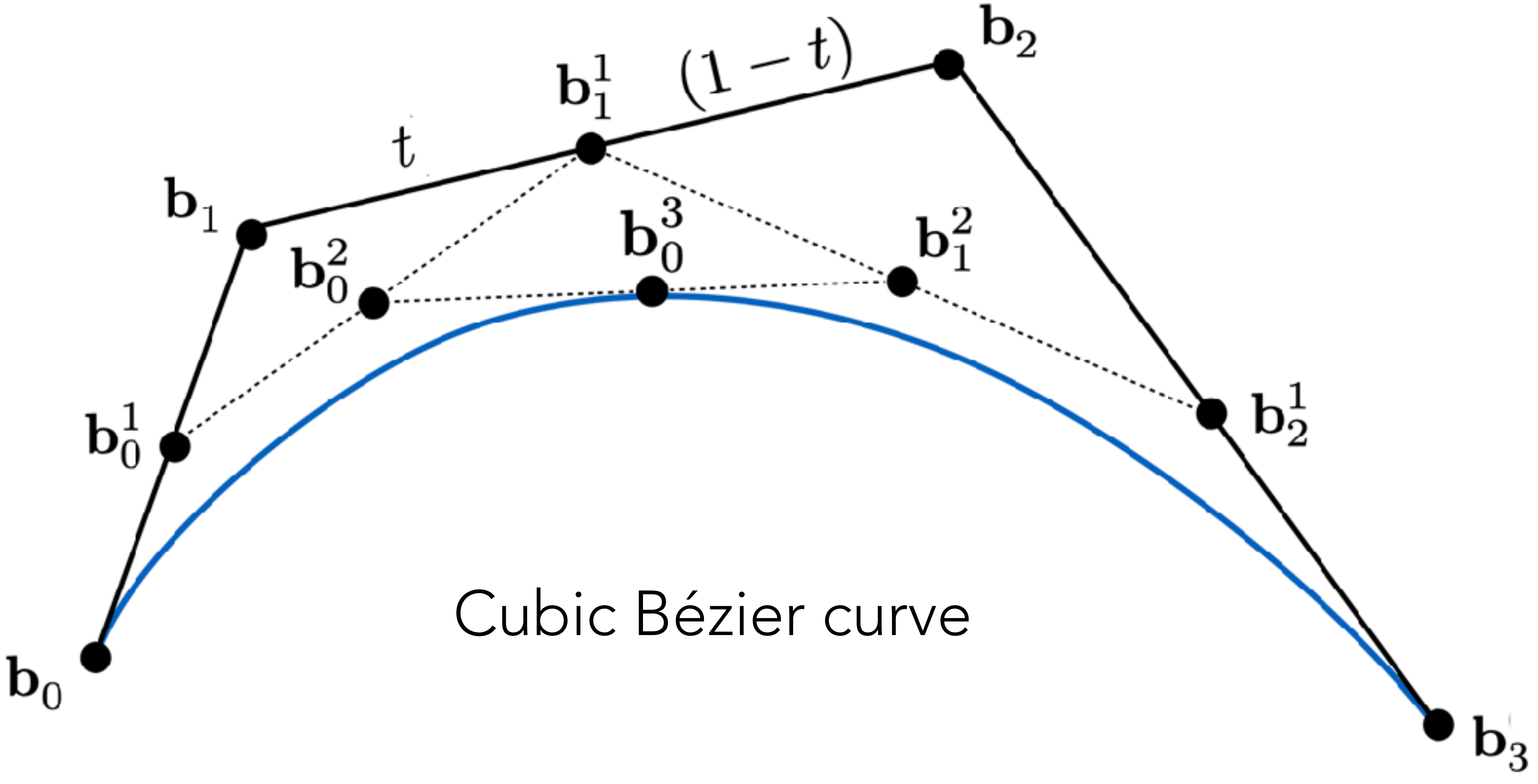
$$\mathbf{b}_0^1 = \text{lerp}(t, \mathbf{b}_0, \mathbf{b}_1)$$
$$\mathbf{b}_1^1 = \text{lerp}(t, \mathbf{b}_1, \mathbf{b}_2)$$

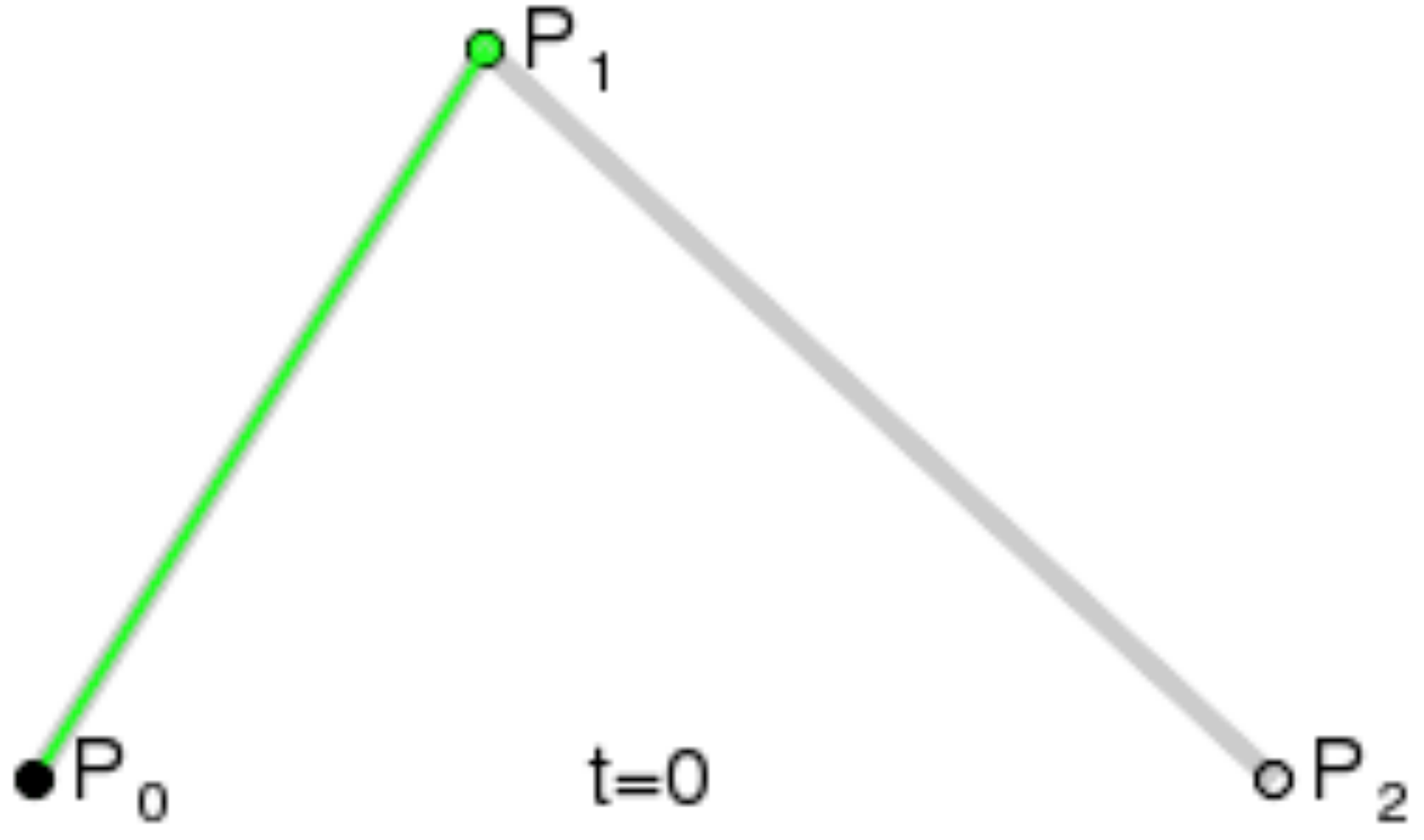
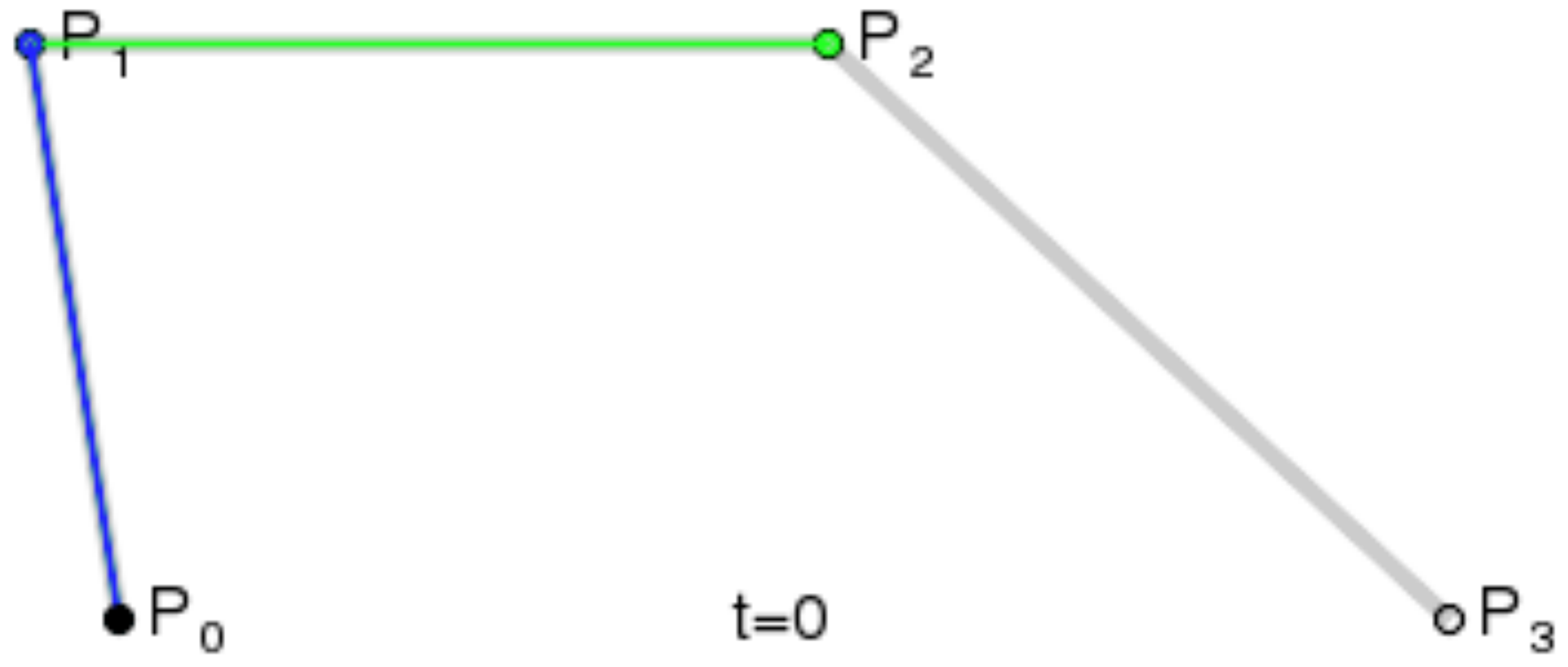
$$\mathbf{b}_0^2 = \text{lerp}(t, \mathbf{b}_0^1, \mathbf{b}_1^1)$$

$$\mathbf{b}_0^1 = \mathrm{lerp}(t, \mathbf{b}_0, \mathbf{b}_1)$$
$$\mathbf{b}_1^1 = \mathrm{lerp}(t, \mathbf{b}_1, \mathbf{b}_2)$$
$$\mathbf{b}_2^1 = \mathrm{lerp}(t, \mathbf{b}_2, \mathbf{b}_3)$$

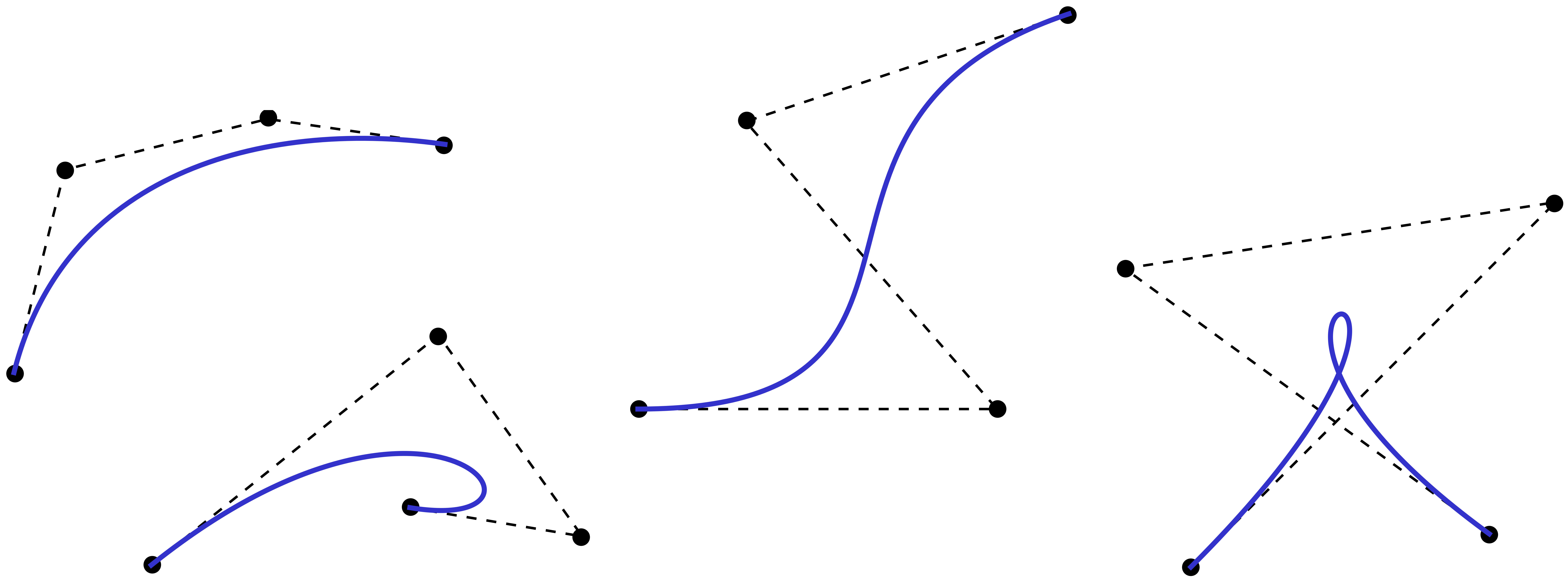
$$\mathbf{b}_0^2 = \mathrm{lerp}(t, \mathbf{b}_0^1, \mathbf{b}_1^1)$$
$$\mathbf{b}_1^2 = \mathrm{lerp}(t, \mathbf{b}_1^1, \mathbf{b}_2^1)$$

$$\mathbf{b}_0^3 = \mathrm{lerp}(t, \mathbf{b}_0^2, \mathbf{b}_1^2)$$




P1
P0
t=0
P2

P1
P2
P0
t=0
P3

No longer interpolation but approximation: Curve is influenced by the control points but does not pass through them

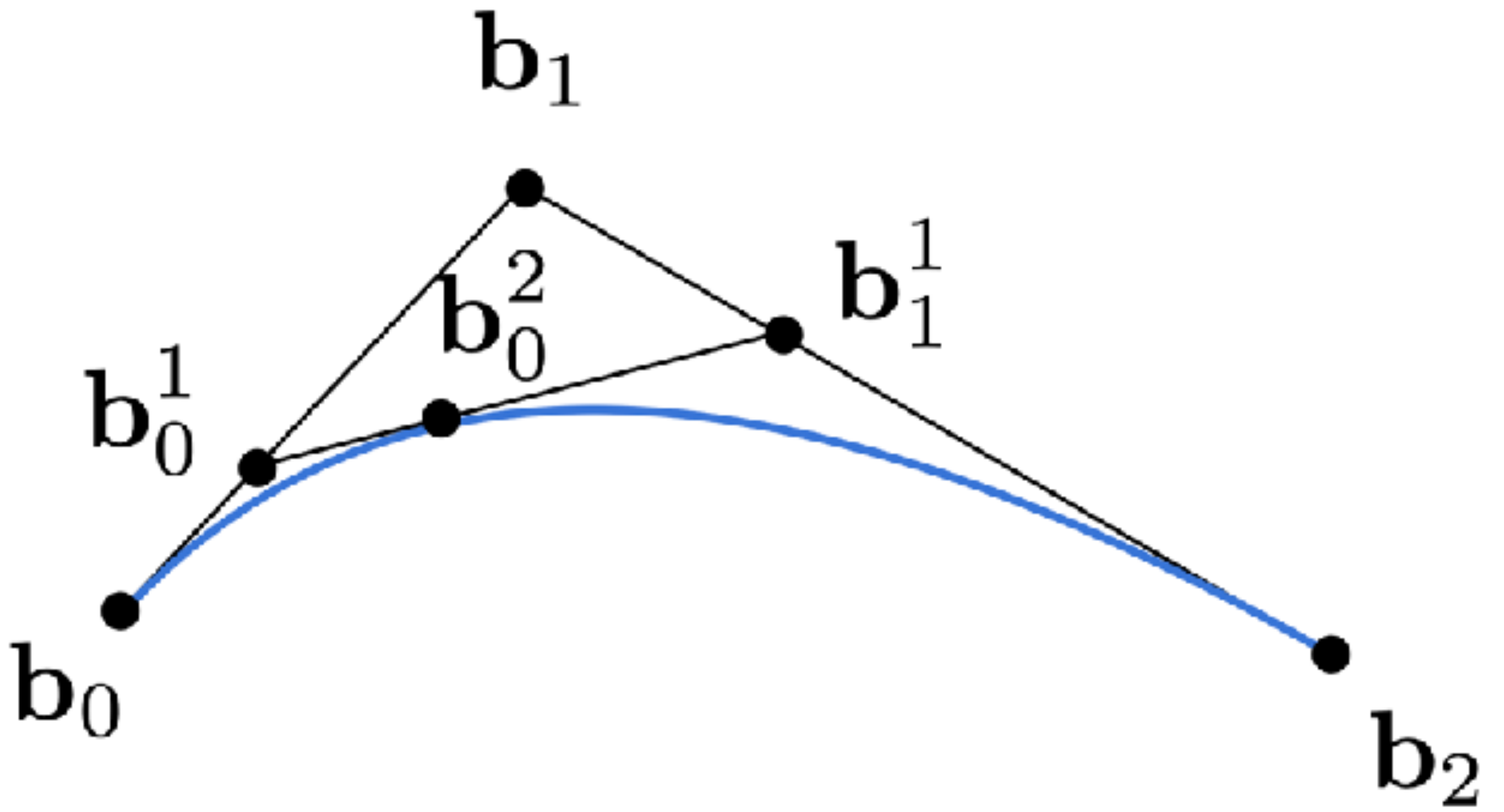


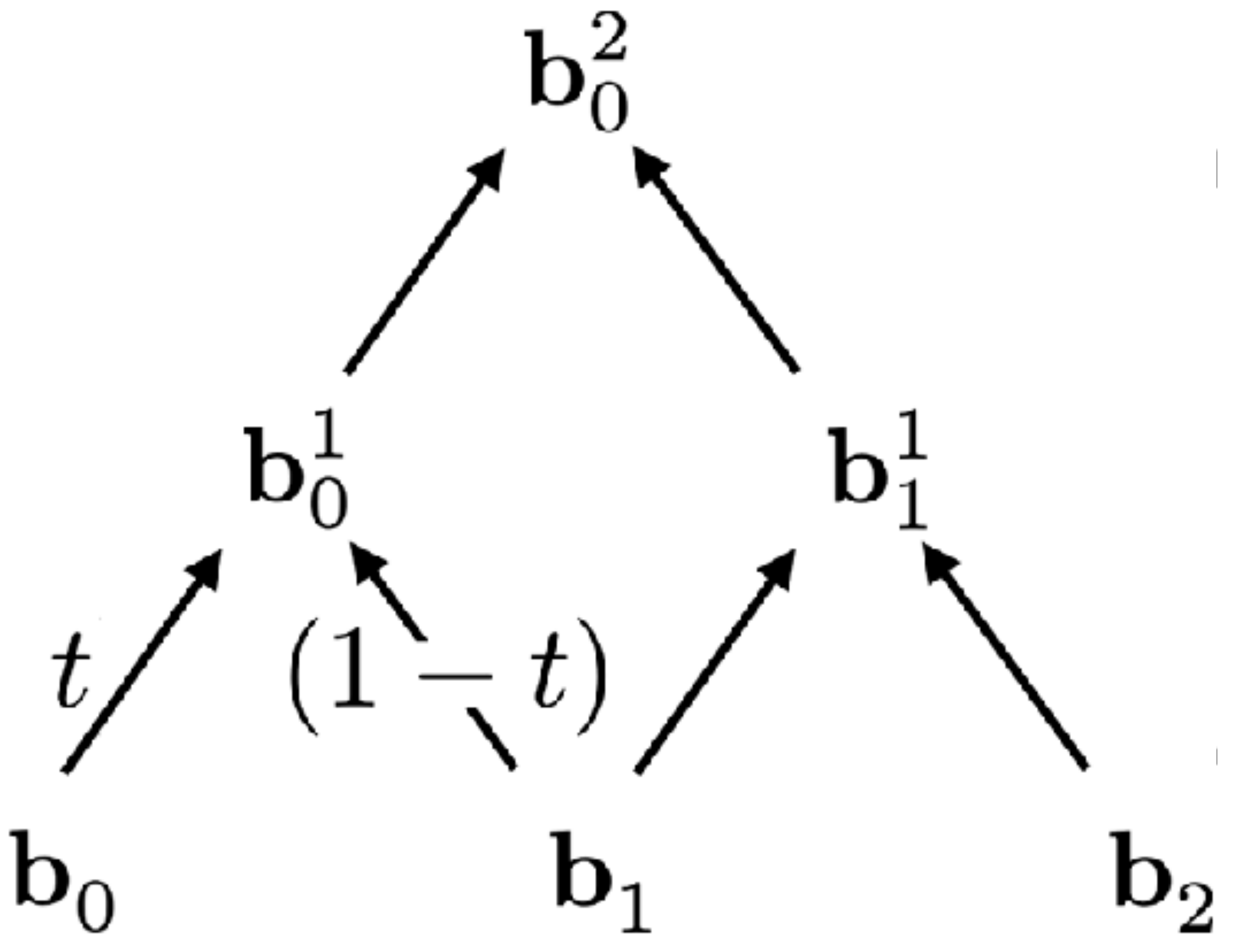
What’s the formula for this parametric curve?

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

$$\begin{aligned}\mathbf{b}_0^2 &= (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1 \\ &= (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2\end{aligned}$$

$$\mathbf{b}_0^2(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

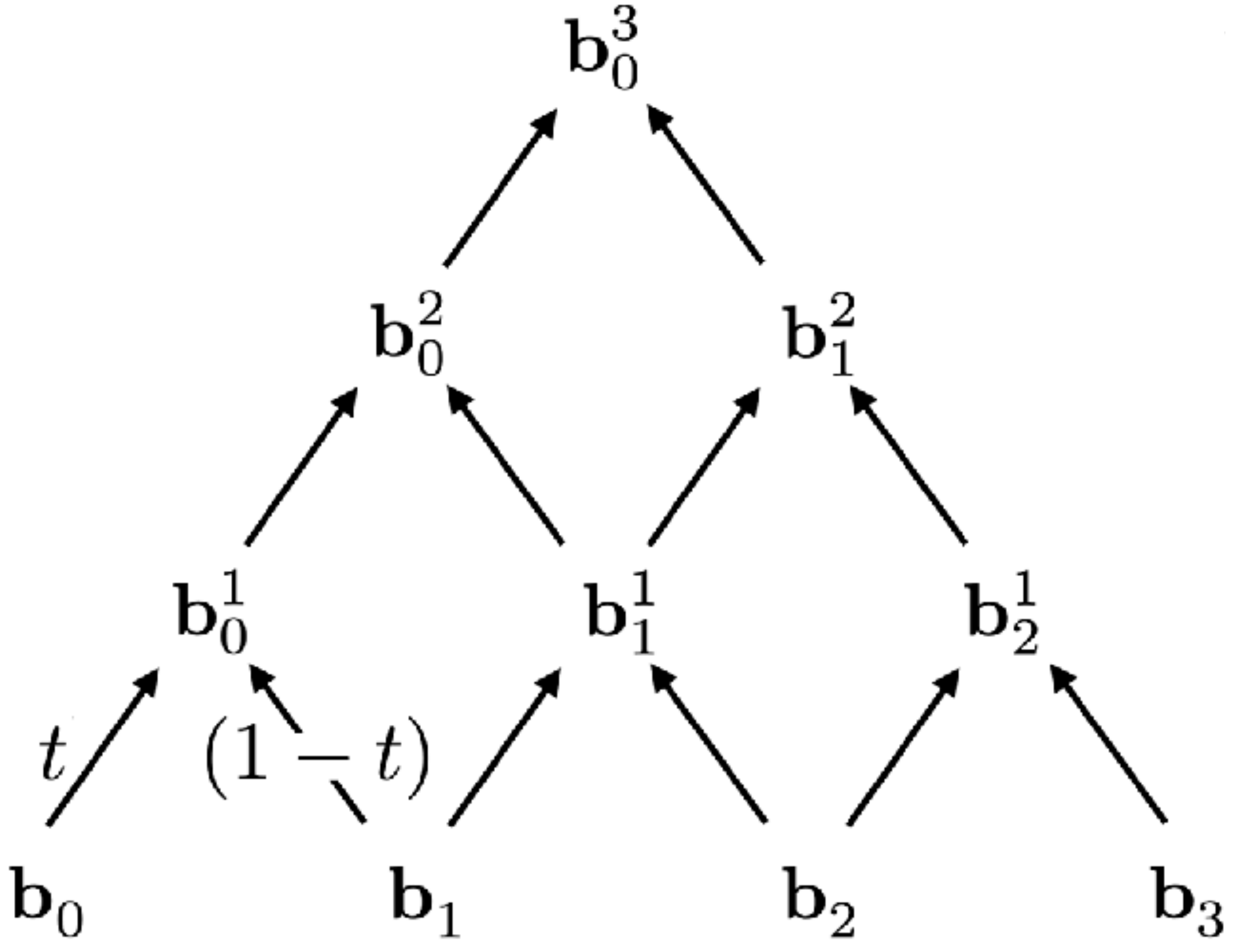




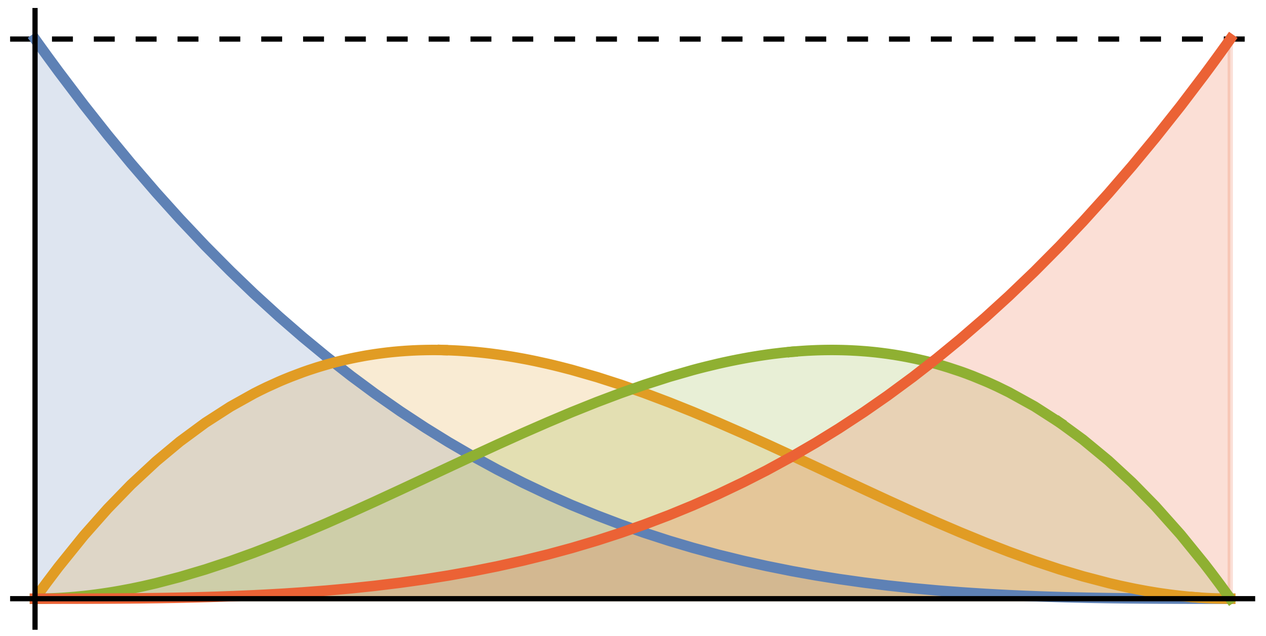
Cubic Bézier:

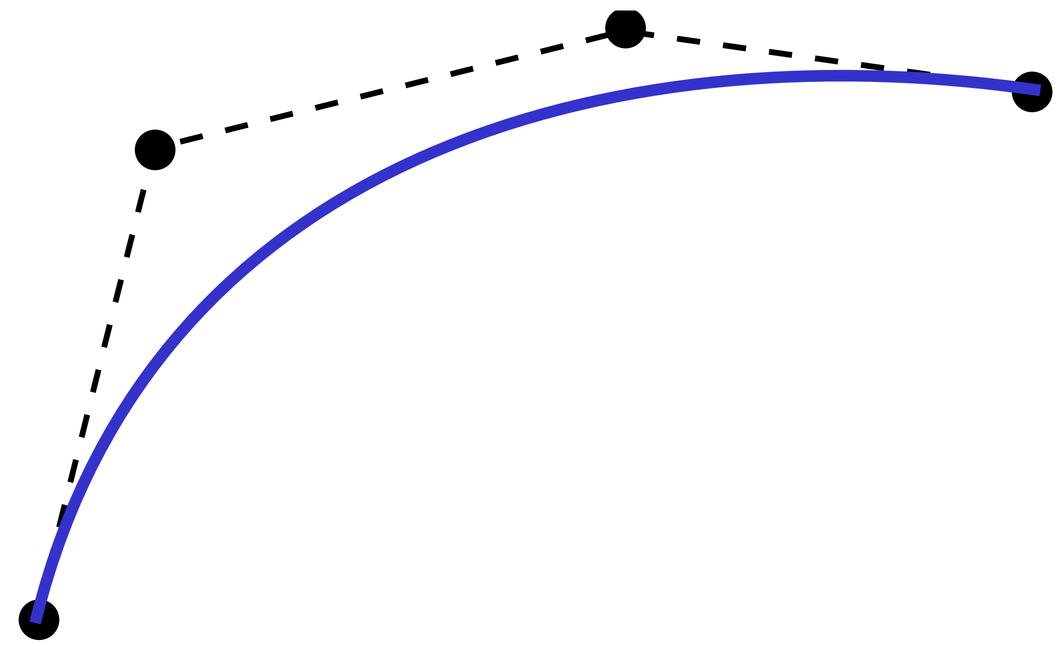
$$\mathbf{b}_0^3 = (1-t)^3\mathbf{b}_0 + 3t(1-t)^2\mathbf{b}_1 + 3t^2(1-t)\mathbf{b}_2 + t^3\mathbf{b}_3$$

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



Influence of each control point:

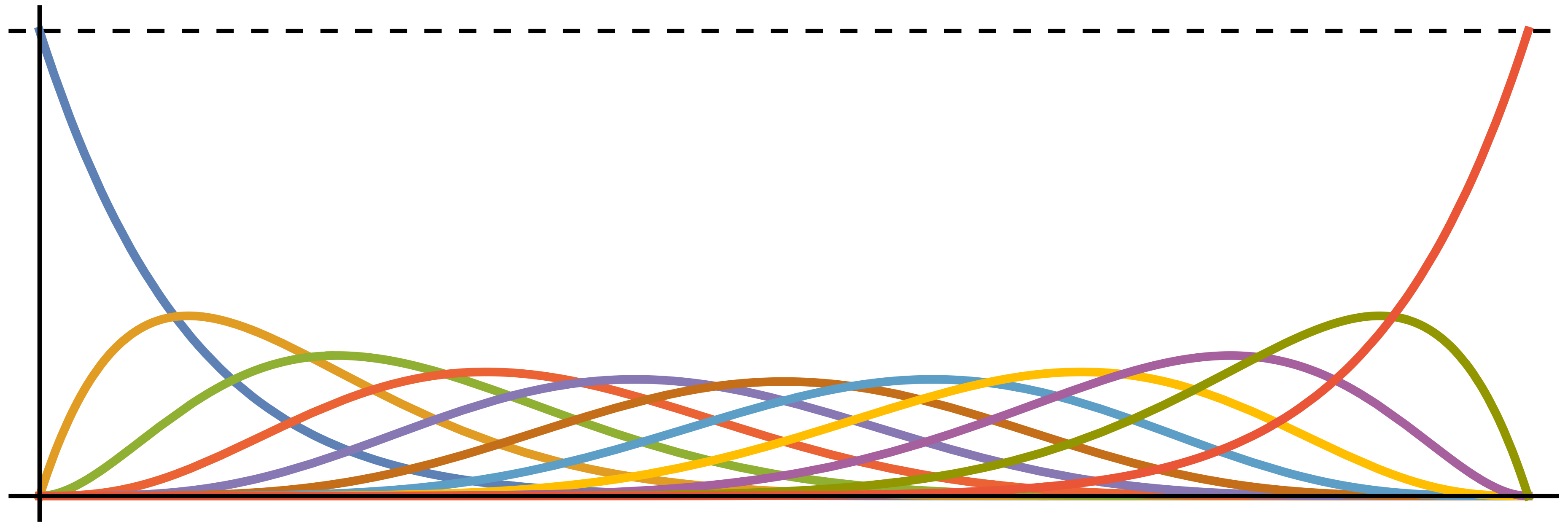




Bernstein polynomials

In general, $\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i$, where

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

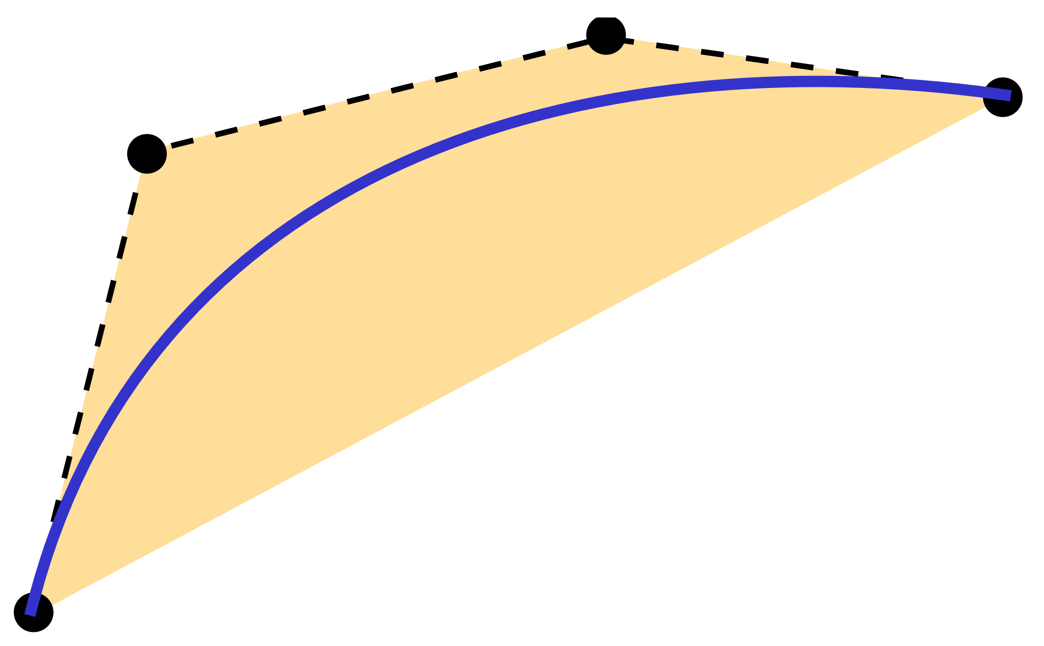


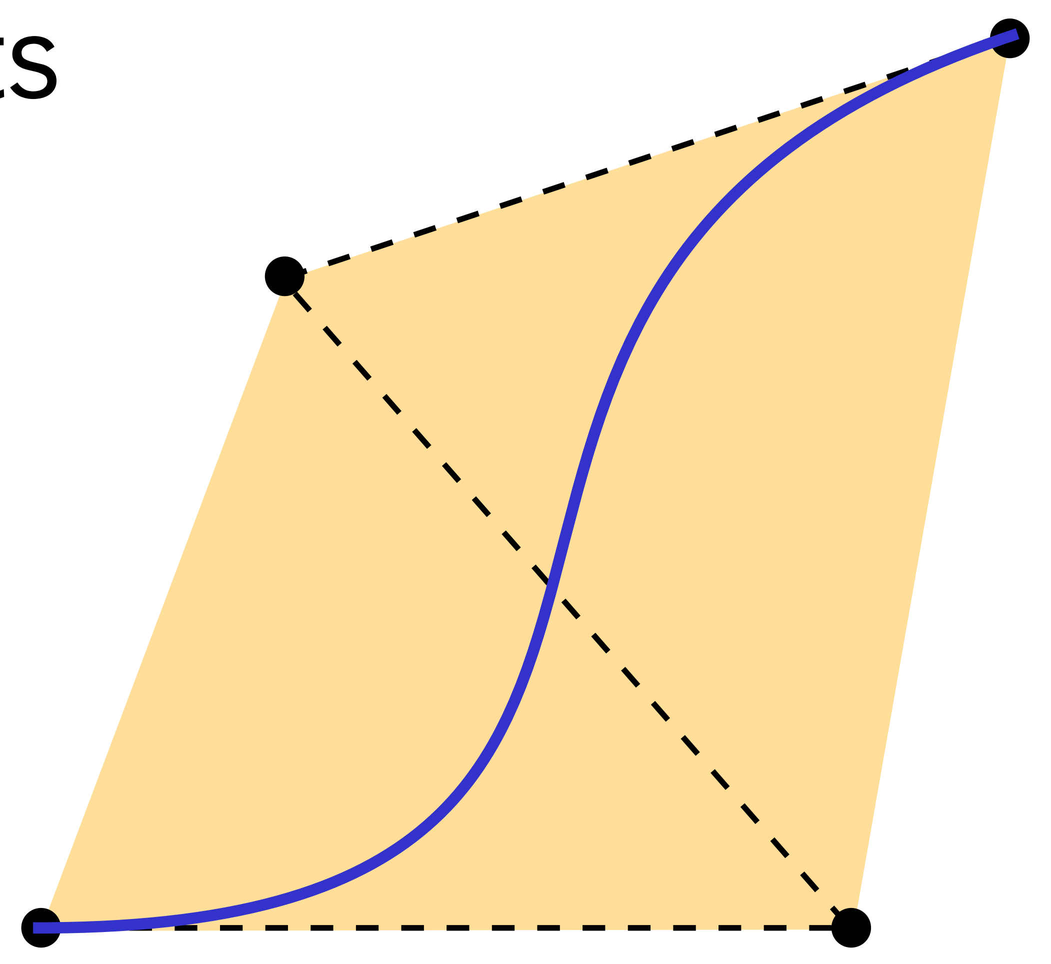
Bernstein polynomials for $n = 10$

- Nonnegative: $B_i^n(t) \geq 0$ for all $t \in [0, 1]$
- Partition of unity: $\sum_{i=0}^{n} B_i^n(t) = 1$ for all $t \in [0, 1]$
- Over $t \in [0, 1]$, $B_i^n(t)$ has a unique maximum at $t = i/n$

Desirable properties of Bézier curves:

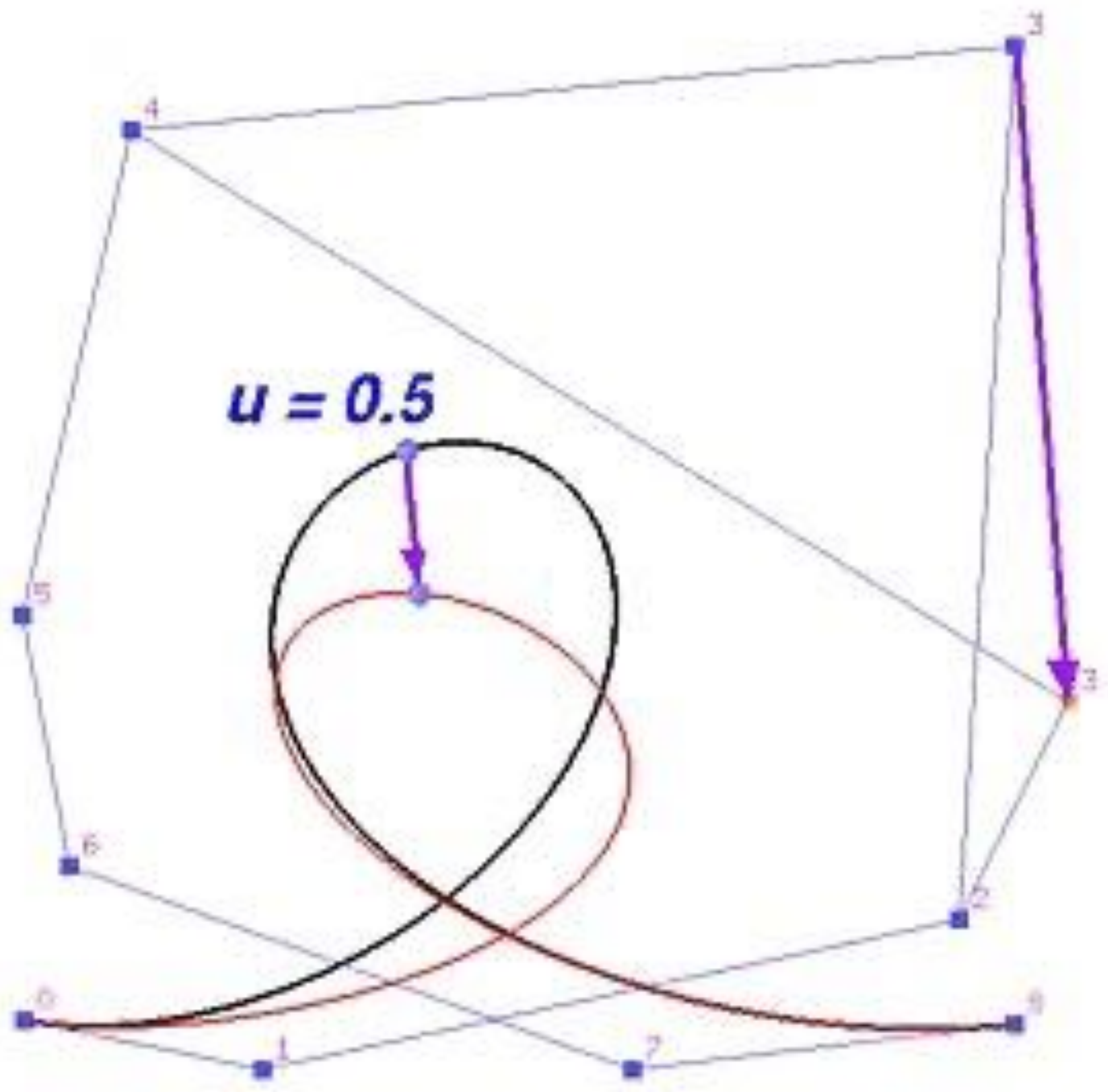
- Interpolates endpoints
- Tangent to end segments
- Affine invariance: Transform curve ⇔ transform control points
- Curve lies inside convex hull of control points

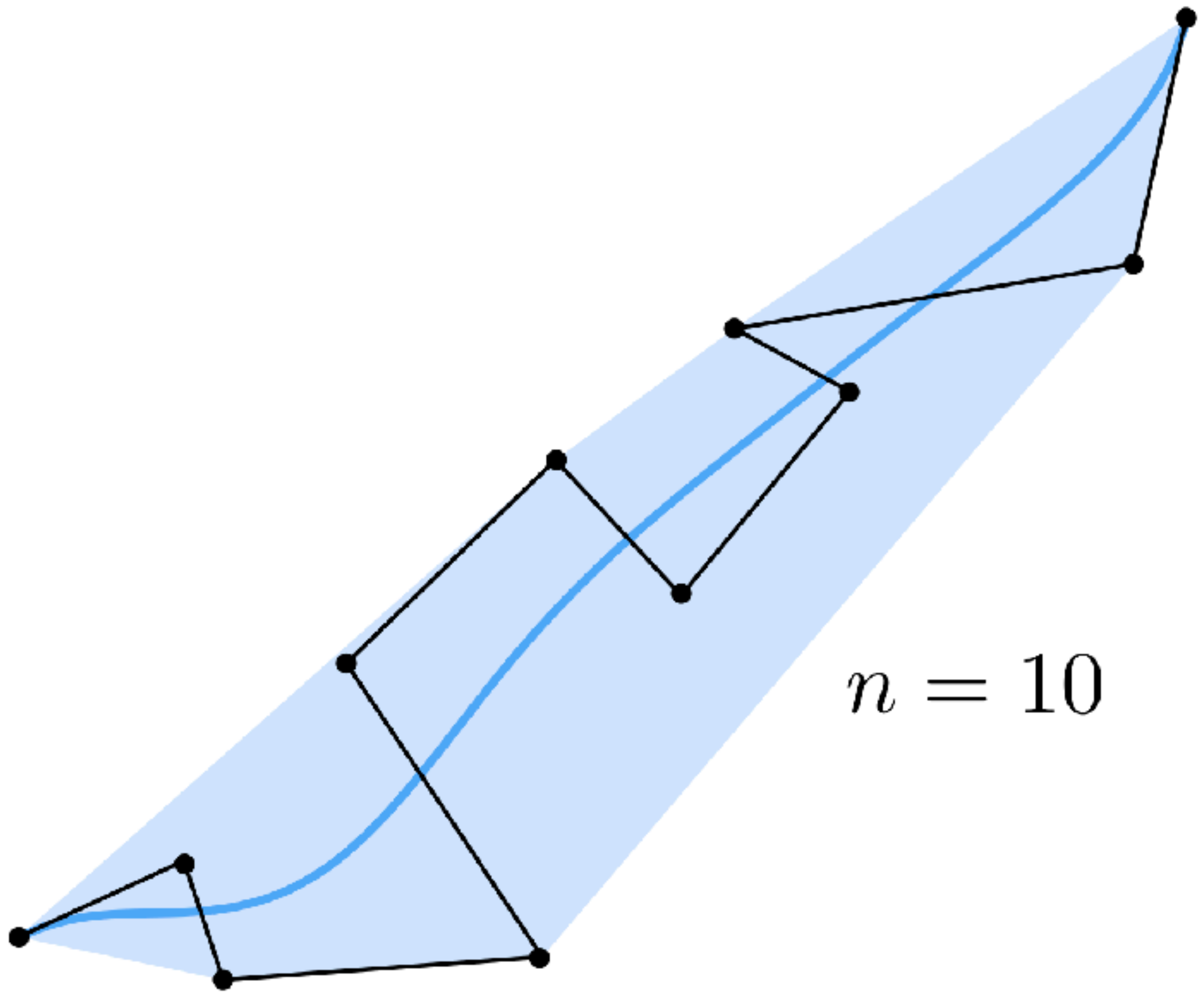




Undesirable properties of Bézier curves:

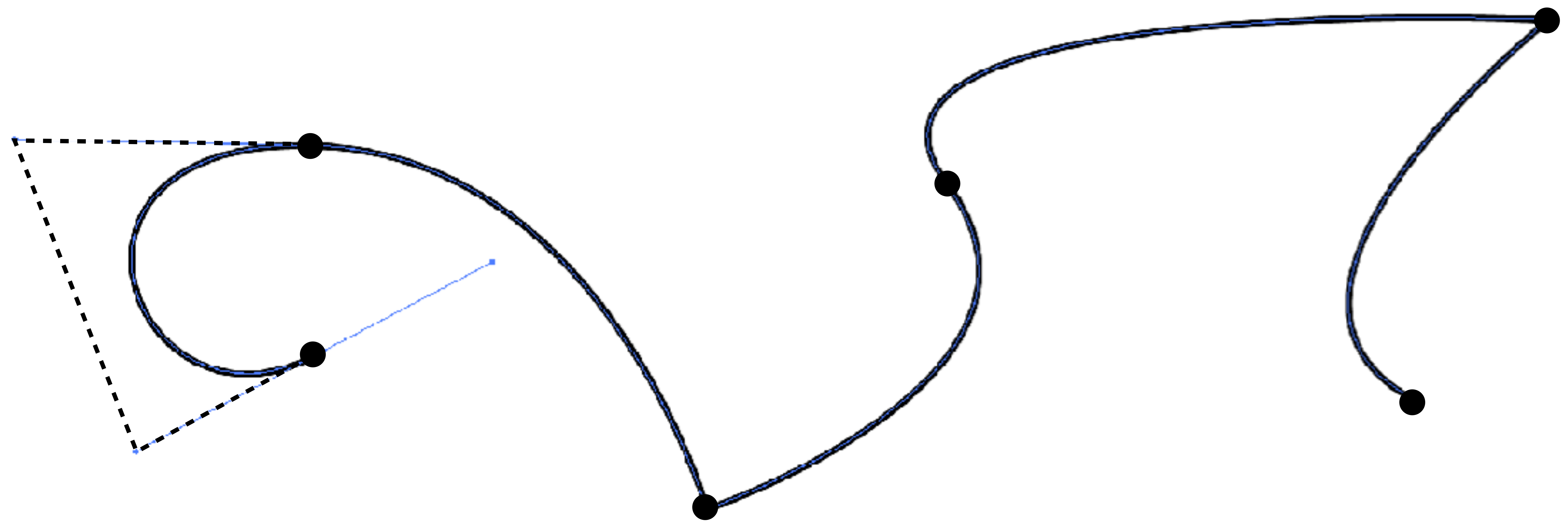
- Lack of local control: moving any one control point affects the whole curve
- High-degree Bézier curves are overly smooth





Piecewise Bézier curves (Bézier splines)

Chain together multiple Bézier curves of low degree (usually cubic)



Now we have local control: each control point only affects one or two segments

Used basically everywhere (fonts, paths, Illustrator, PowerPoint, …)

How to ensure that the pieces join up smoothly?

C^0 continuity: $\mathbf{p}(t)$ is continuous in t

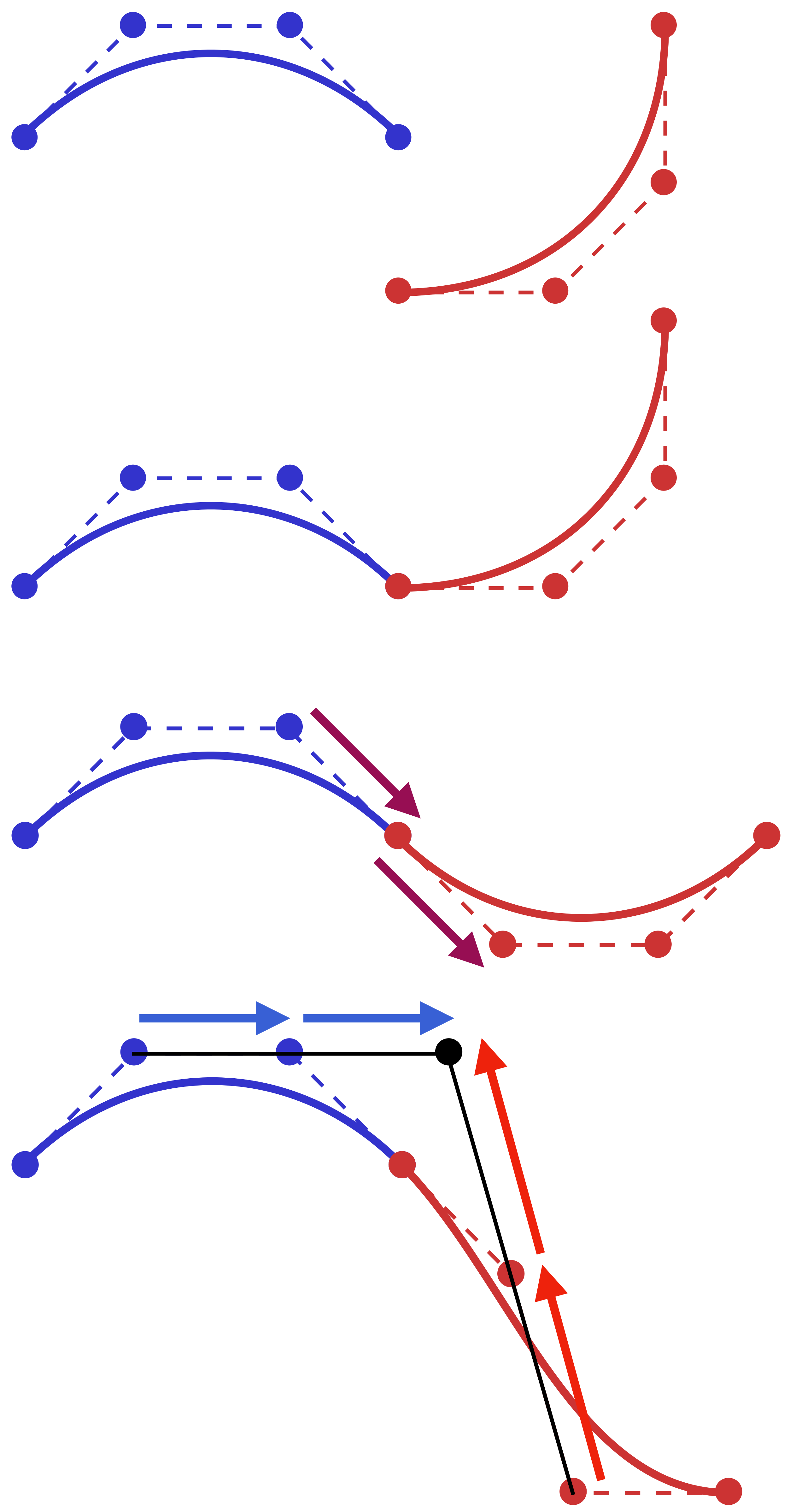
- Endpoints meet

C^1 continuity: $d\mathbf{p}/dt$ is also continuous

- Tangents (i.e. end segments) agree

C^2 continuity: $d^2\mathbf{p}/dt^2$ is also continuous

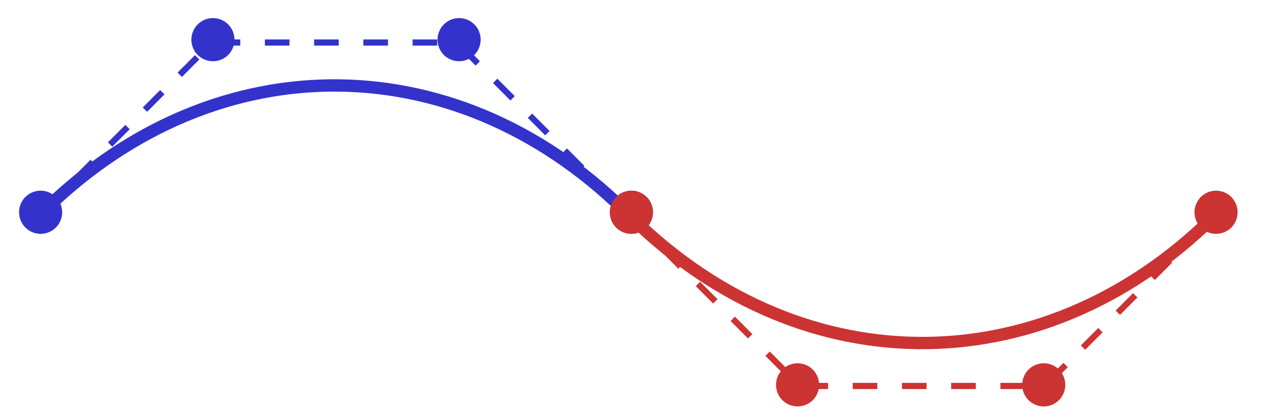
- "A-frame" construction: extrapolated segments should coincide

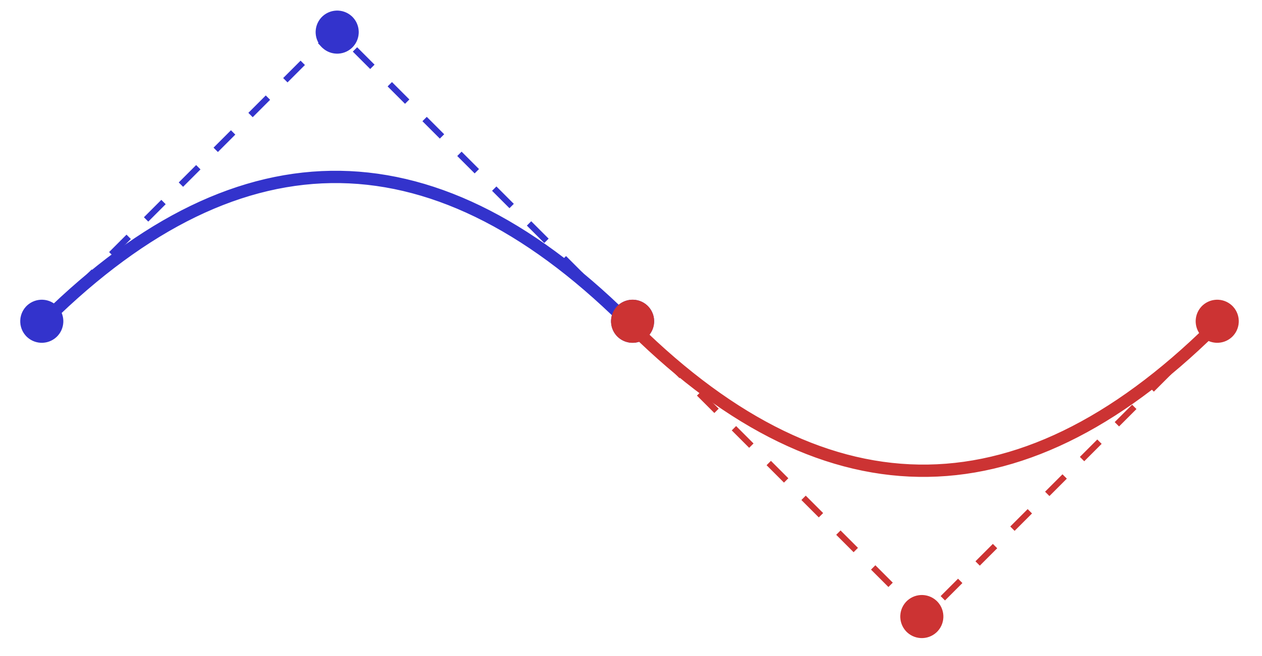


Puzzle:

Why does everyone use piecewise cubic Bézier curves?

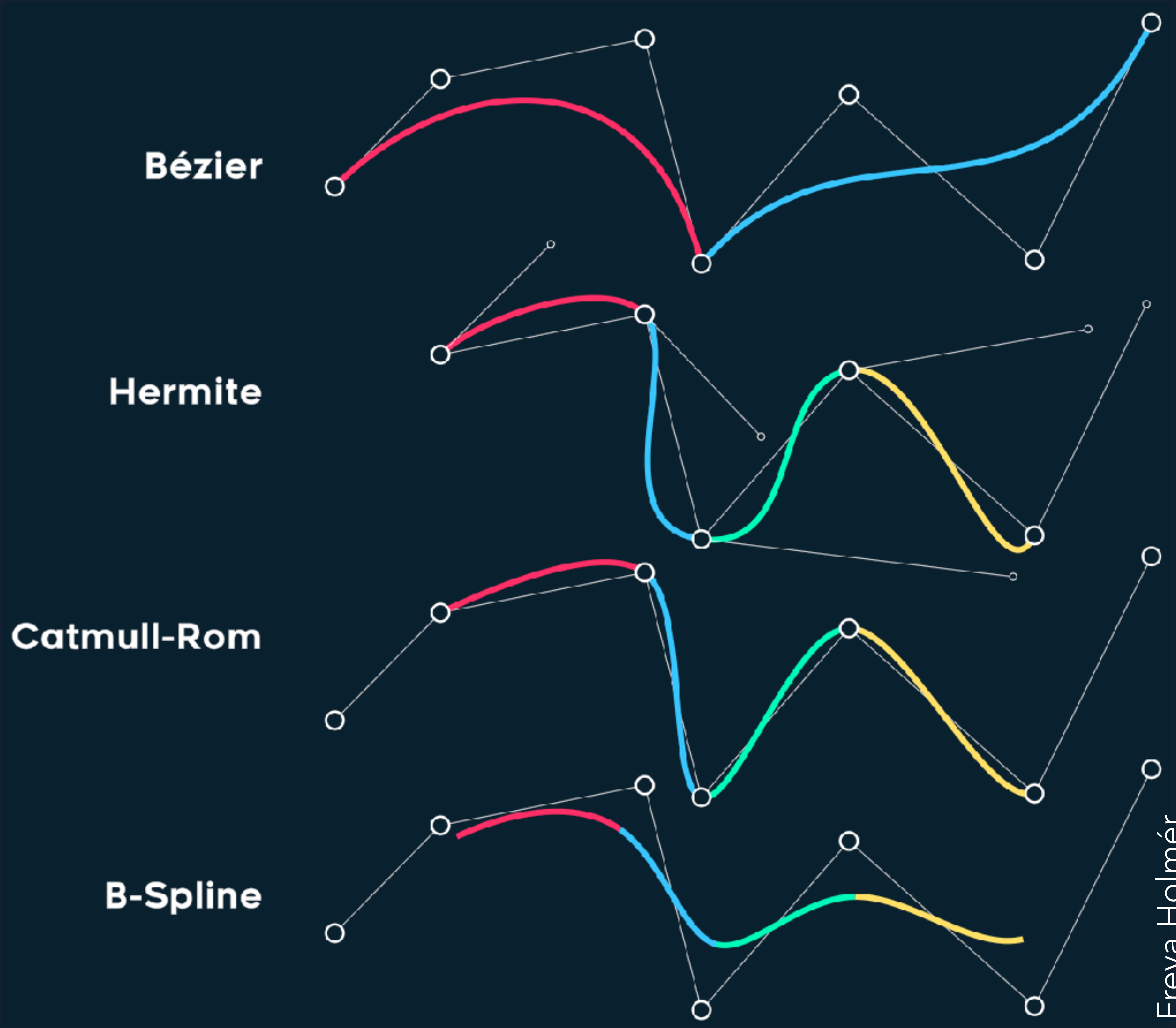
Couldn't we get C^1 continuity with just piecewise quadratic Bézier curves?





Lots of other types of splines we don't have time to cover:

- Hermite
- Catmull-Rom
- B-spline
- NURBS
- …



Freya Holmér