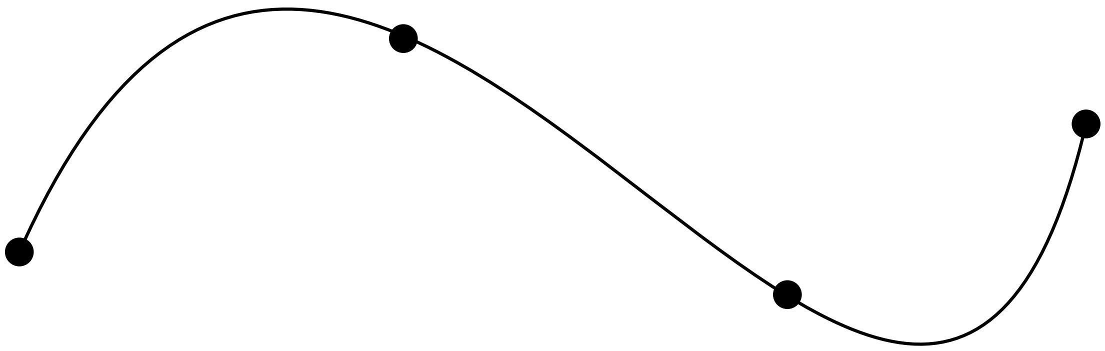
BÉZIER CURVES

Rodrigo Silveira

Curve and Surface Design
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

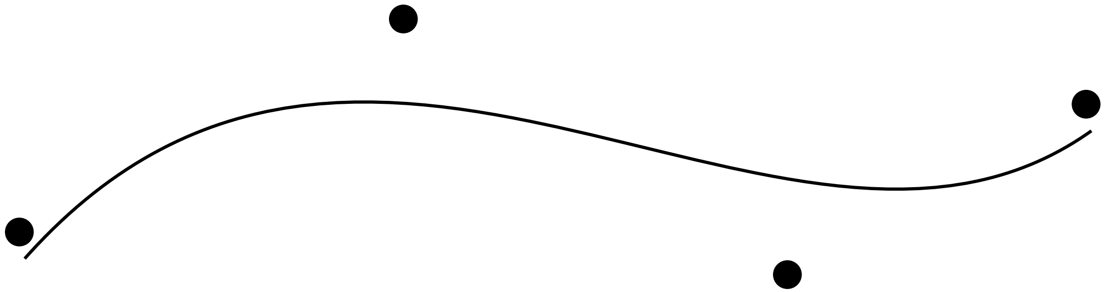
Interpolation or... approximation!

Previous curve design methods based on **interpolation**



Interpolating curve

Curve passes exactly through given points



Approximating curve

Curve passes near the given points

What's wrong with interpolation? Curve change when moving points is unpredictable

Approximating curves can provide better "shape control"

Bézier curves

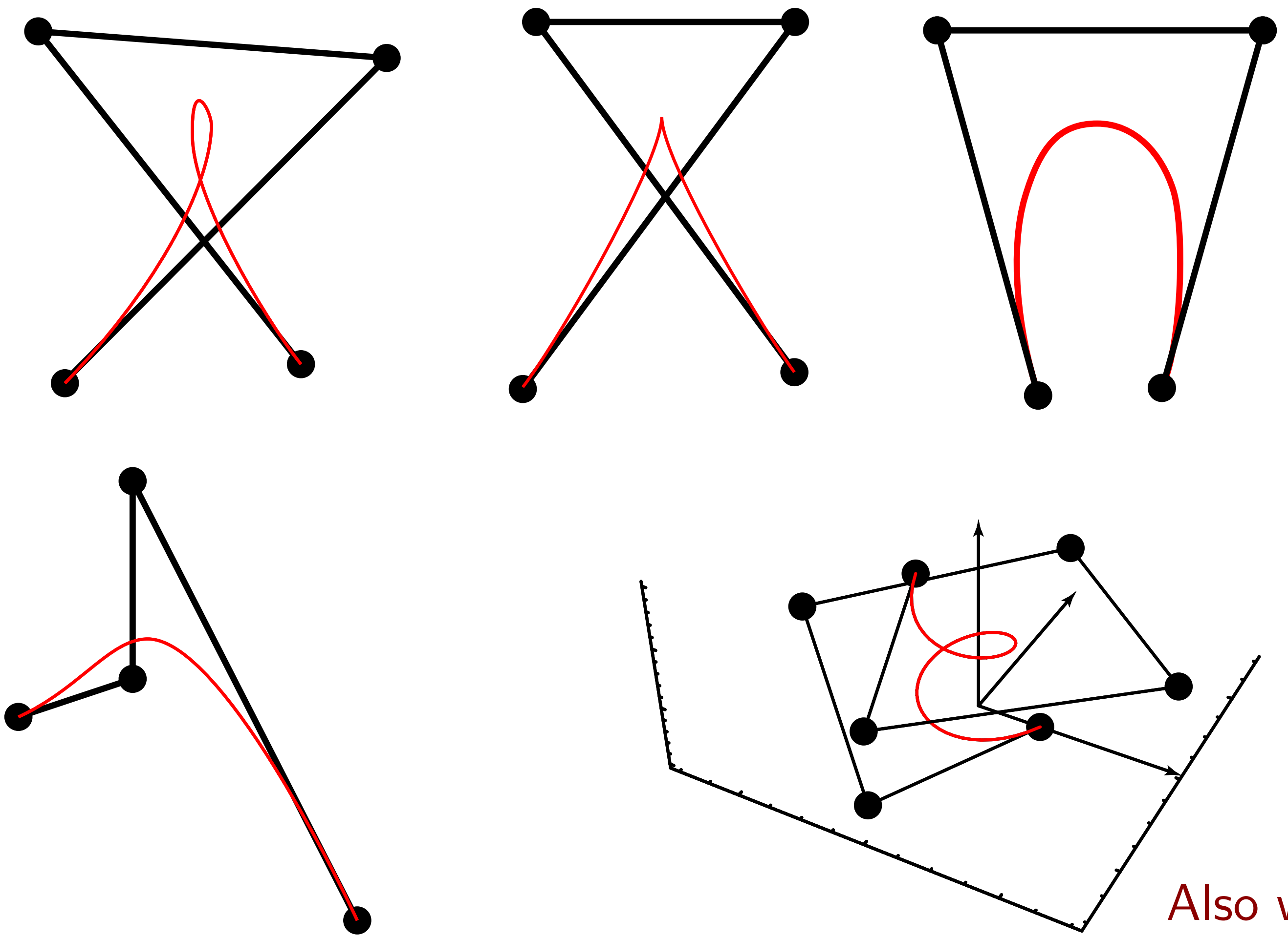
Named after Pierre Bézier (1910-1999)

- Worked on automizing the process of designing cars
- Paul de Casteljau (Citröen) developed similar methods, but were never published

Bézier curve

- Parametric ($P(t)$)
- Polynomial
- Based on *control points*

Some examples of Bézier curves



Also works in 3D!

What is a Bézier curve?

General form

$$P(t) = \sum_{i=0}^{n} \underbrace{P_i}_{\text{control point}} \underbrace{f_i(t)}_{\text{basis function}} \qquad t \in [0,1]$$

control point

basis function: gives weight of each point as function of t

Bézier looked for basis functions that gave the following properties:

- Interpolates the first and last point (to have control on first and last point)
- Tangent vectors: at P_0 must be $P_1 - P_0$, at P_n must be $P_n - P_{n-1}$ (to have control of directions at the beginning and end)
- Similar to 2), for higher order derivatives: $P^{(k)}(0)$ should depend on $P_0, \ldots, P_k$ only (e.g., $P''(0)$ should depend only on P_0, P_1, and P_2)
- The basis functions must be symmetric with respect to t and $(1-t)$ (so reversing the parameter and the order of control points gives the same curve)
- Control point weights are barycentric: shape independent from coordinate system. That is: $P(t)$ is an affine combination of control points, so curve is invariant under affinities

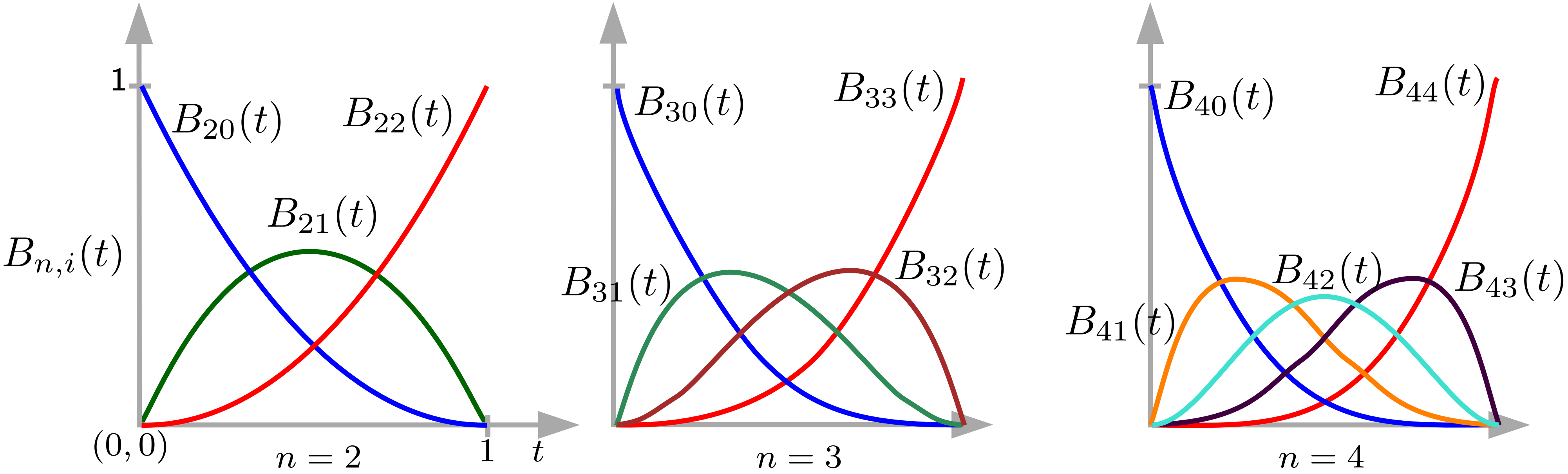
Basis functions

The family a functions used are **Bernstein polynomials**

$$f_i(t) = B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

recall that
$0 \leq i \leq n$,
$\binom{n}{i} = \frac{n!}{i!(n-i)!}$, $0! = 1$
and assume $0^0 = 1$

note the n: the basis depends on the number of control points



The Bézier curve becomes $P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t) \qquad t \in [0, 1]$

Example: degree-2 Bézier curve

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

For $n = 2$, we have $B_{n,i}(t) = \binom{2}{i} t^i (1-t)^{2-i}$, for $0 \leq i \leq 2$

So, for $n = 2$, these are the three Bernstein polynomials:

- $B_{2,0}(t) = \binom{2}{0} t^0 (1-t)^{2-0} = (1-t)^2$
- $B_{2,1}(t) = \binom{2}{1} t^1 (1-t)^{2-1} = 2t(1-t)$
- $B_{2,2}(t) = \binom{2}{2} t^2 (1-t)^{2-2} = t^2$

So the quadratic Bézier curve is

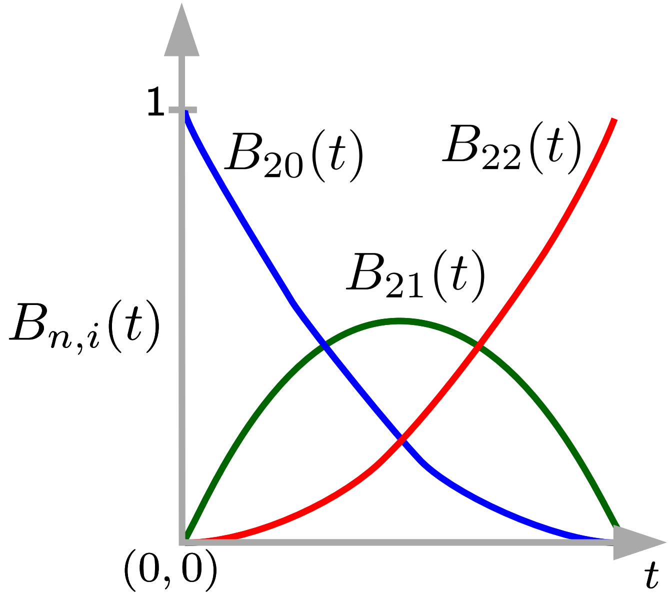
$$P(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 P_2$$

Example?

Question: Does this curve satisfy the properties in the previous slide?

Properties of Bézier curves

1. Endpoint interpolation ✓
2. Symmetry ✓
3. Affine invariance
4. Invariance under affine parameter transformations
5. Convex hull property
6. Pseudolocal control
7. Variation-diminishing property



$$P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t)$$

Properties of Bézier curves

3. Affine invariance

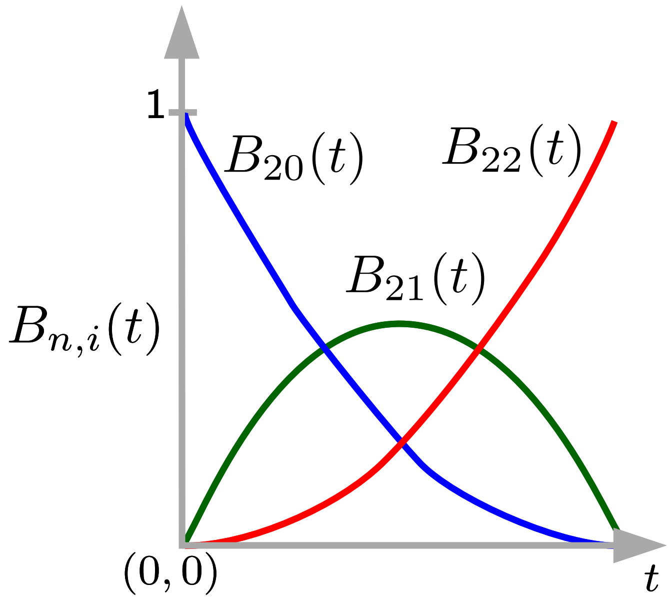
Applying an affine transformation to the curve is the same as applying the transformation to the control points

More precisely: $f(P(t)) = \sum_{i=0}^{n} f(P_i) B_{n,i}(t)$, for any affine map f, i.e., $f(v) = Av + W$

Why is that? Observe that $\sum_{i=0}^{n} B_{n,i}(t) = 1$

This follows from binomial theorem:
$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$, with $a = 1$ and $b = (1-t)$

Affine maps are precisely the maps that leave affine combinations invariant, so same applies to Bézier curves!



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

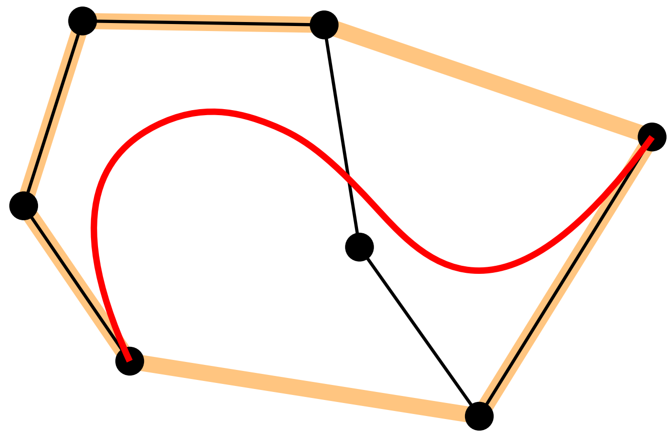
$$P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t)$$

4. Invariance under affine parameter transformations

That is: $\sum_{i=0}^{n} P_i B_{n,i}(t) = \sum_{i=0}^{n} P_i B_{n,i}(\frac{u-a}{b-a})$

Practical consecuence: it is easy to have a curve defined over $[a, b]$ instead of $[0, 1]$

Properties of Bézier curves



5. Convex hull property

The curve lies inside the convex hull of the control points

Why is this important? Gives local control (remember Runge's phenomenon), and helps in checking if two curves intersect (**Question**: how?)

Why is this property true?

$P(t) = \sum_{i=0}^{n} B_{n,i} P_i(t)$ where $\sum_{i=0}^{n} B_{n,i}(t) = 1$ and $B_{n,i}(t) \geq 0 \; \forall n, i$

$P(t)$ is a **convex combination** of the control points.

The convex hull of the a set of points S is **exactly** the set of all convex combinations of points in S, thus all points in the curve belong to the convex hull.

Question: What does this say about collinear control points?

Properties of Bézier curves

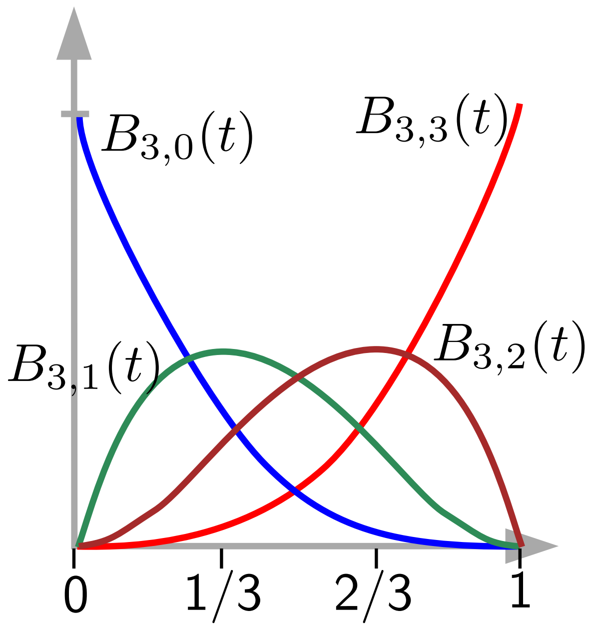
6. "Pseudolocal" control

Question: When does a control point influence the curve most?

The Bernstein polynomials have only one maximum at $t = i/n$.

Consequence: if we move only one control point, P_i, the curve is mostly affected around $t = i/n$. This makes the effect of the change more or less predictable.

However, note that the change still affects the whole curve (so it is **global control**).

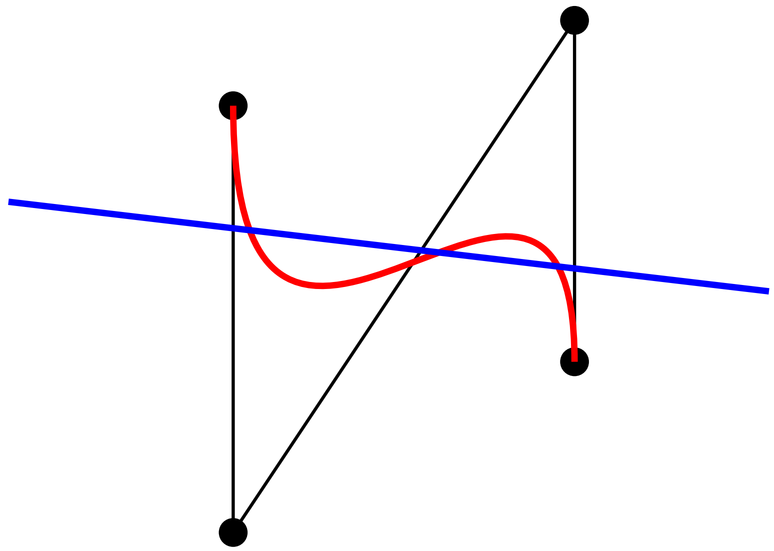


local maxima of $B_{n,i}$s

Question: What happens to $P(t)$ if P_k is moved by a vector (α, β)?

Properties of Bézier curves

7. Variation-diminishing property



The number of intersections of any line with a Bézier curve is at most the number of intersections of the line with the control polygonal line

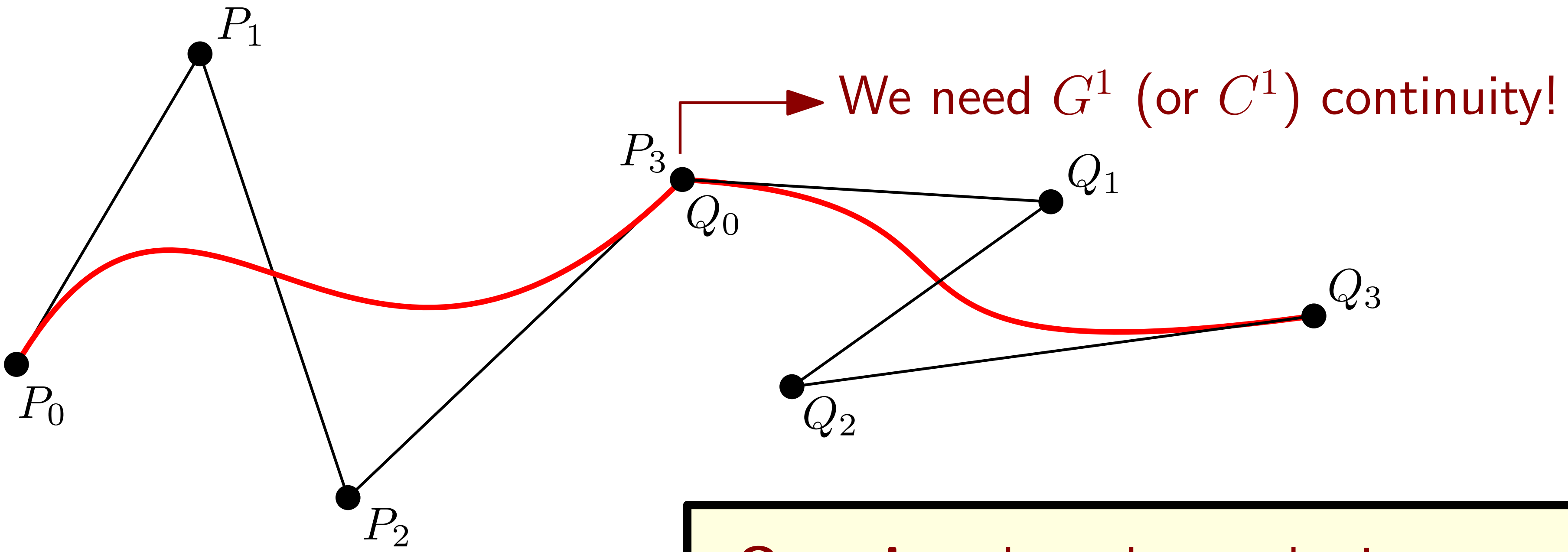
This means that, to some extent, the curve imitates the shape and is not “rougher” than the corresponding control polygon,

One consequence: if the control polygon is convex, then the Bézier curve is also convex

Proof? Later, after looking at **degree elevation**

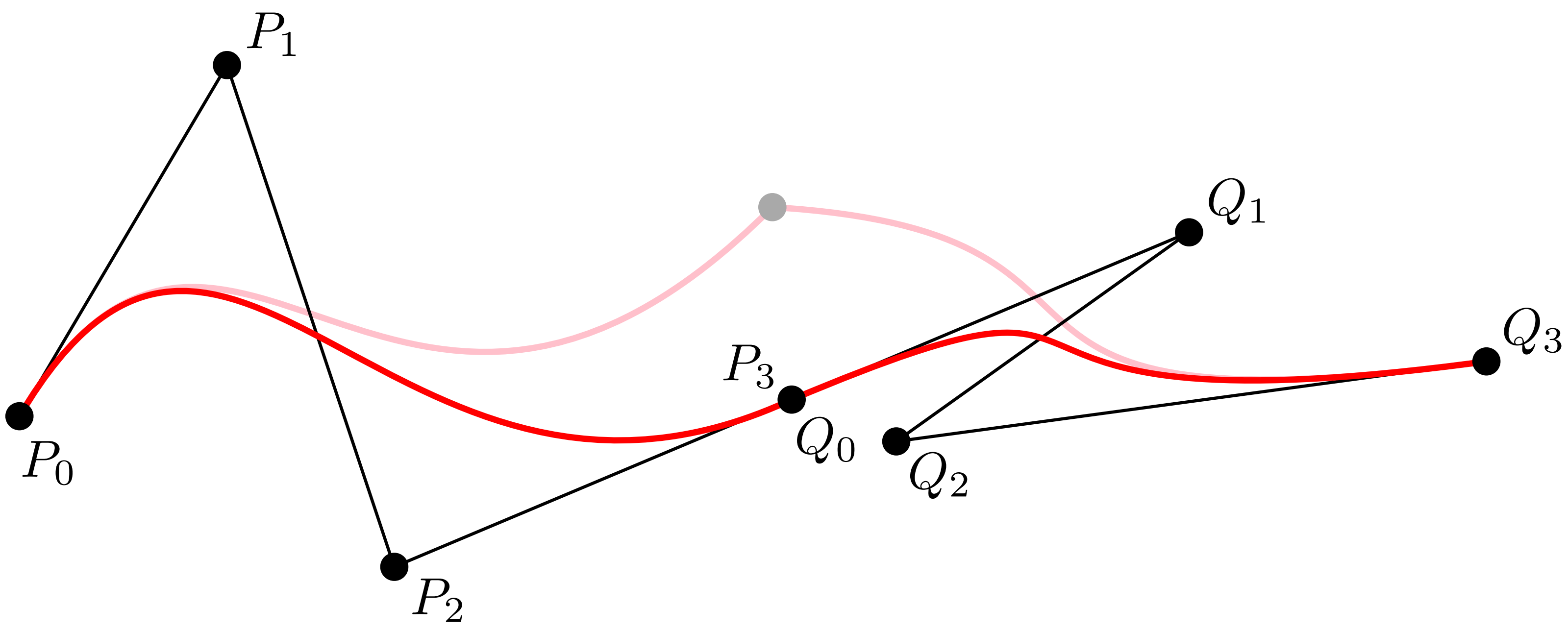
Connecting two curves

- In practice, one should avoid high-degree Bézier curves
- Better use many low-degree curves (they give local control)
- Requires smooth connection between consecutive curves



Question: how do we obtain a smooth joint?

Connecting two curves



In general, for a curve P with $(n+1)$ control points and Q with $(m+1)$, the C^1-continuity condition is

$$Q_0 = P_n = \frac{m}{m+n} Q_1 + \frac{n}{m+n} P_{n-1}$$

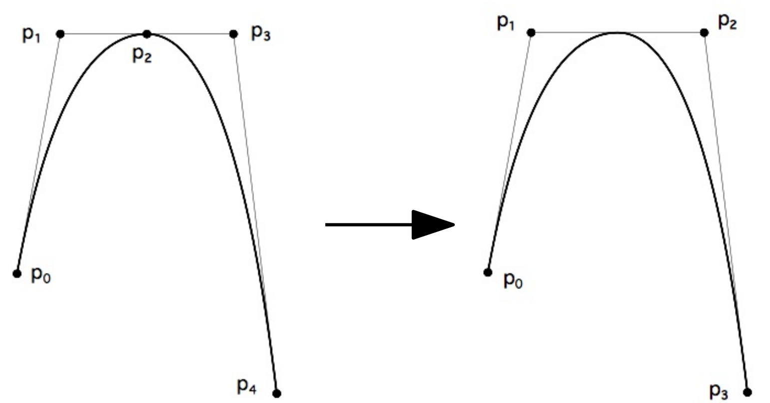
Question: how can you obtain higher-degree continuity?

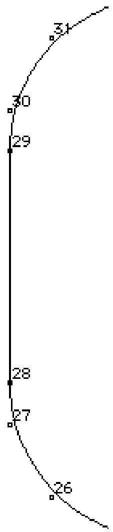
Guess how are the fonts you use designed?

- True-type fonts (Apple, Microsoft): quad
- PostScript (Adobe, printers,...) and Oper

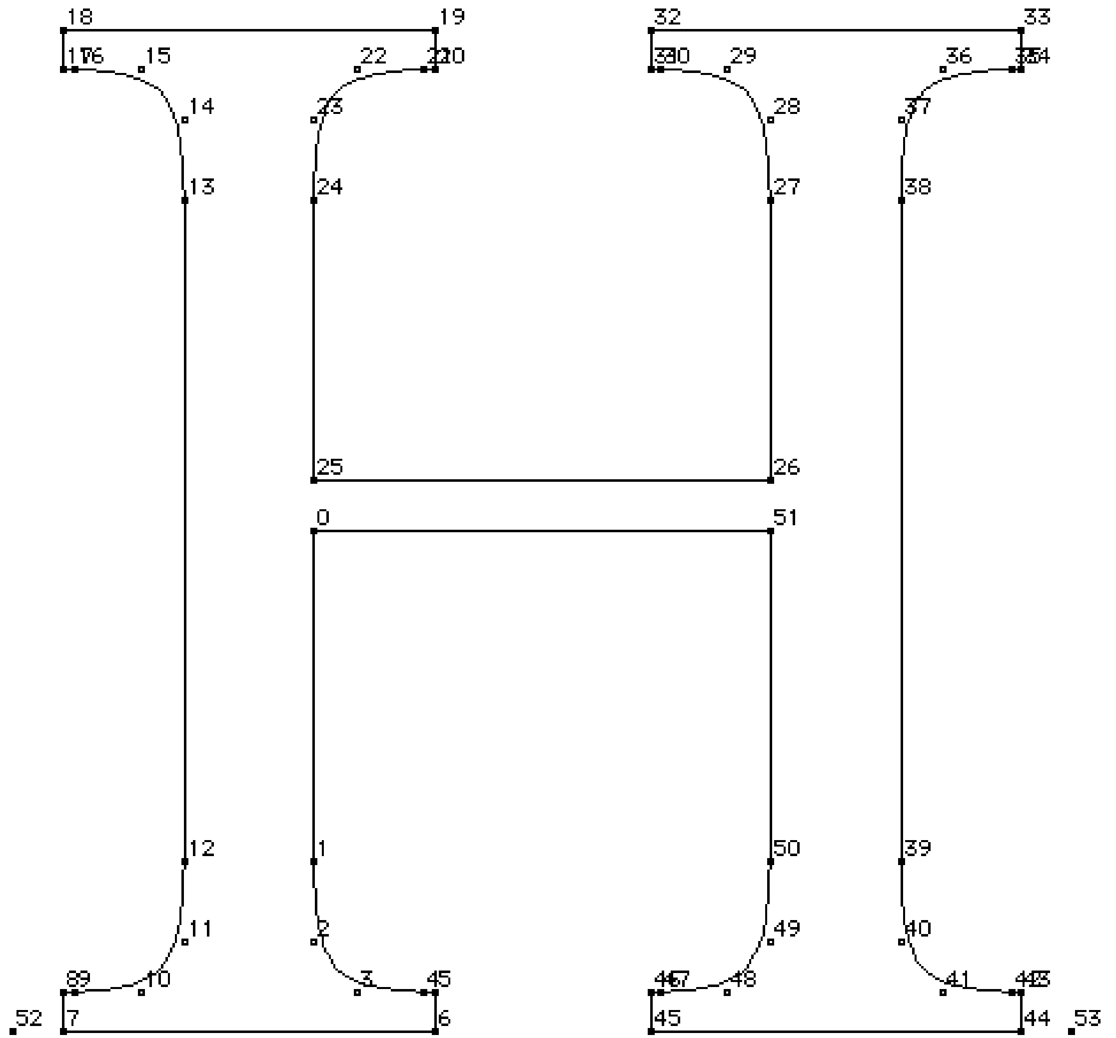
Question: Can you convert between these types of fonts?

- Storage of glyphs in TTF:





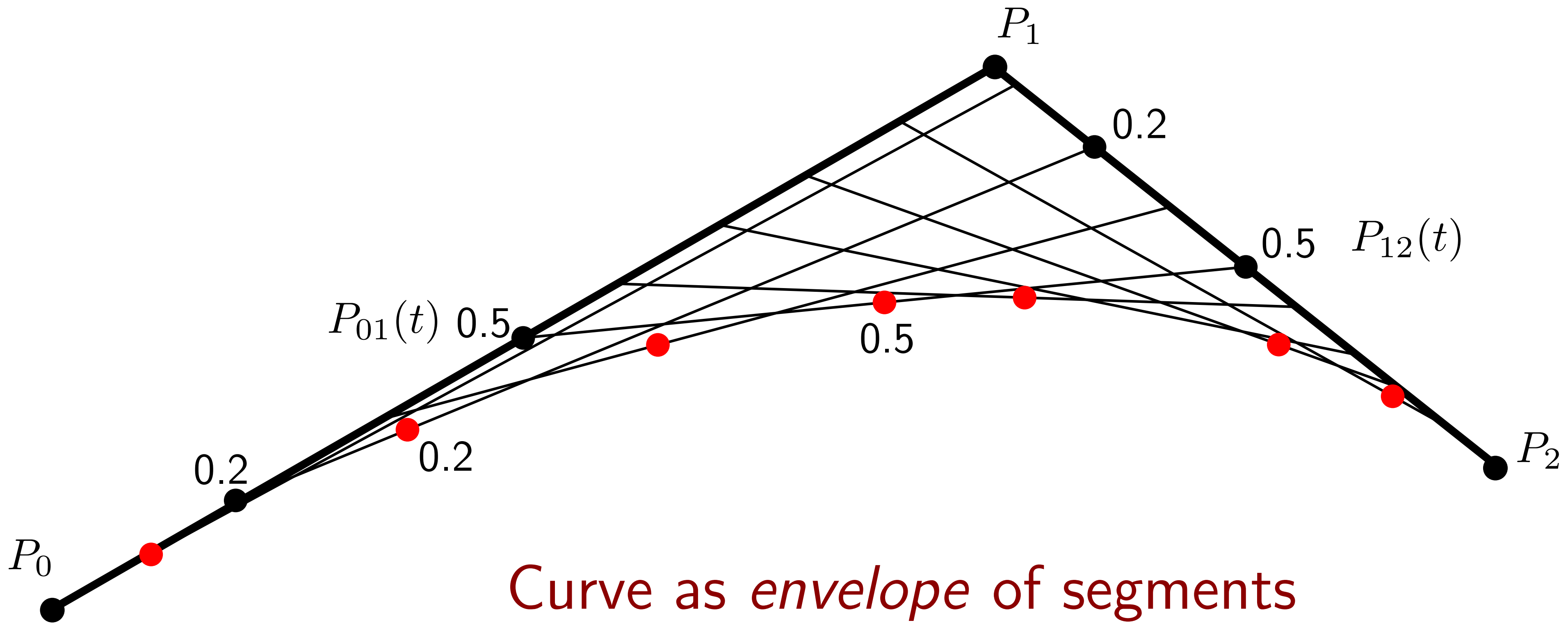
Gl



Difference between points on-curve and off-curve

An alternative approach to Bézier curves

De Casteljau (Citroën) followed a different approach based on **repeated linear interpolation**



Curve as *envelope* of segments

Question: What is the expression of this envelope, as a function of t?

De Casteljau's algorithm

For 3 points P_0, P_1, P_2, and $0 \leq t \leq 1$, we have:

$$P_{01}(t) = (1-t)P_0 + tP_1$$
$$P_{12}(t) = (1-t)P_1 + tP_2$$
$$P(t) = P_{012}(t) = (1-t)P_{01}(t) + tP_{12}(t)$$

This repeated linear interpolation process can be generalized to n points

For n points $P_0, \ldots, P_n$, $0 \leq t \leq 1$, and $0 \leq i \leq j \leq n$, we have:

$$\begin{aligned} P_i(t) &= P_i \\ P_{i(i+1)\ldots j}(t) &= (1-t)P_{i\ldots(j-1)}(t) + tP_{(i+1)\ldots j}(t) \end{aligned}$$

Recursive / geometric construction method

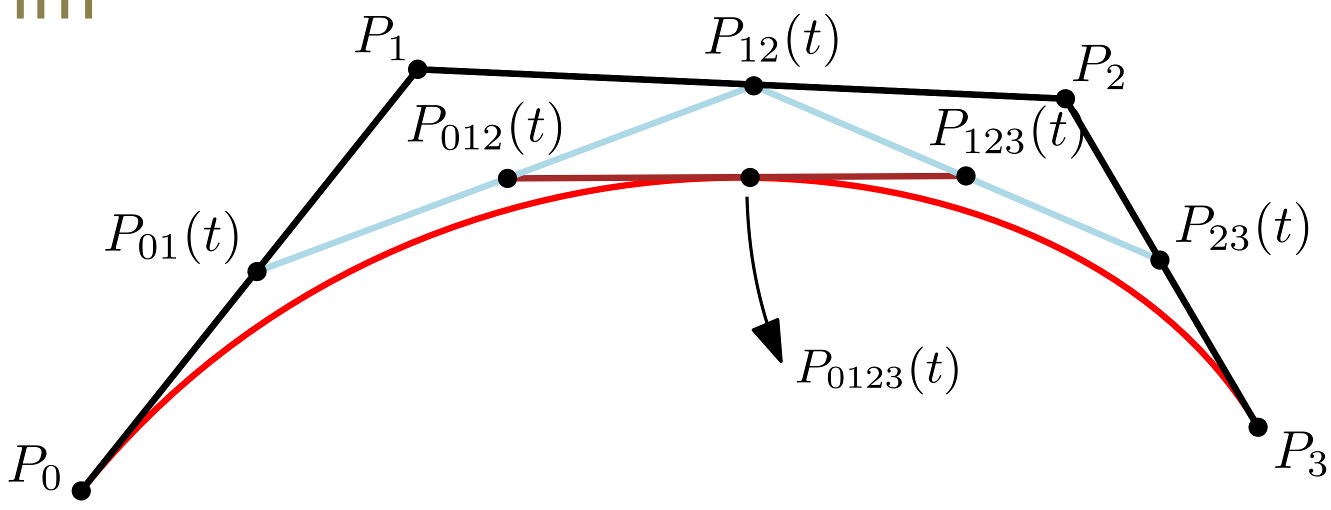
The final curve is given by $P(t) = P_{0\ldots n}(t)$

BÉZIER CURVES AS LINEAR INTERPOLATION

De Casteljau's algorithm

Example for $n = 3$ and $t = 1/2$

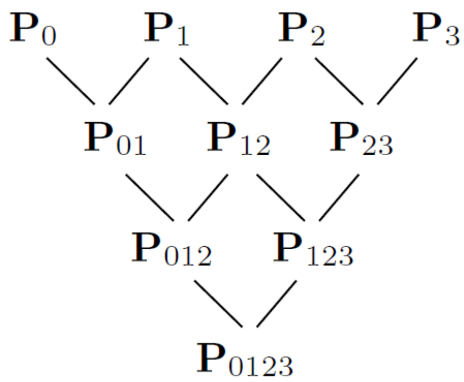
$$P(t) = P_{0123}(t)$$



Implementation of the algorithm

How to evaluate $P(1/2)$?

Step	Points constructed	#points
1	$\mathbf{P}_{01}\ \mathbf{P}_{12}\ \mathbf{P}_{23} \ldots \mathbf{P}_{n-1,n}$	n
2	$\mathbf{P}_{012}\ \mathbf{P}_{123}\ \mathbf{P}_{234} \ldots \mathbf{P}_{n-2,n-1,n}$	$n-1$
3	$\mathbf{P}_{0123}\ \mathbf{P}_{1234}\ \mathbf{P}_{2345} \ldots \mathbf{P}_{n-3,n-2,n-1,n}$	$n-2$
$\vdots$	$\vdots$	$\vdots$
n	$\mathbf{P}_{0123\ldots n}$	



How many points computated in total?

$$n + (n-1) + (n-2) + \cdots + 2 + 1 = n(n+1)/2$$

BÉZIER CURVES AS LINEAR INTERPOLATION

De Casteljau's algorithm

Note: to generate one point on the curve, $\approx n^2/2$ computations is quite a lot...

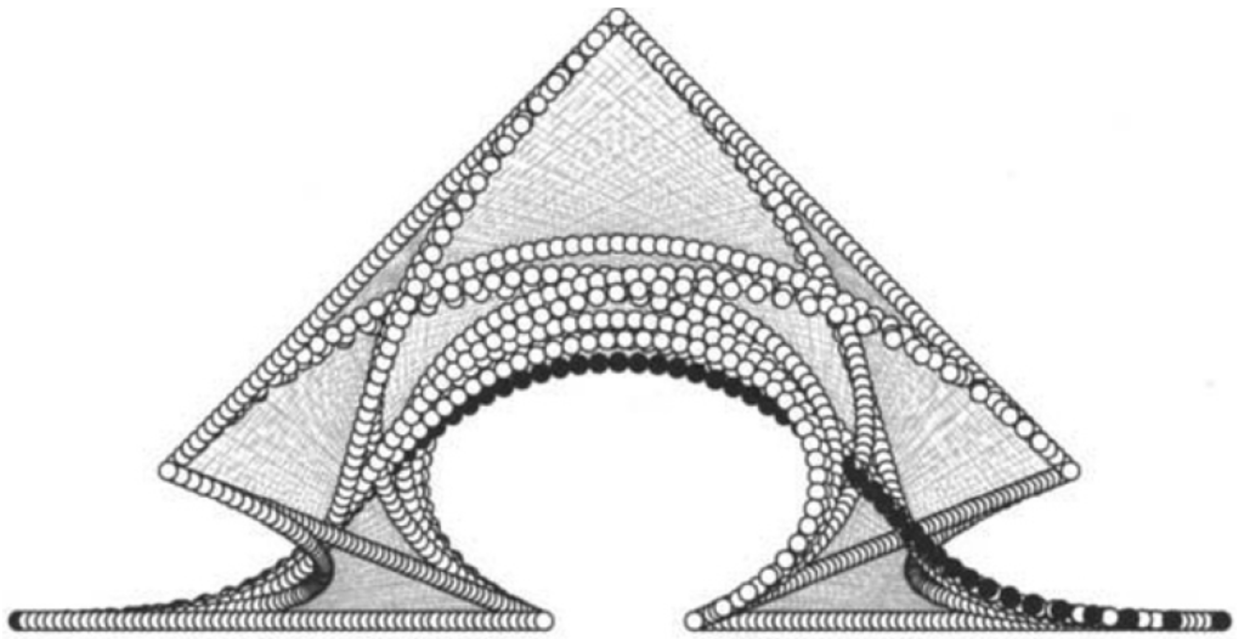


Figure 4.3 The de Casteljau algorithm: 60 points are computed on a degree six curve; all intermedediate points $\mathbf{b}_i^r$ are shown.

Figure from book by Farin (page 47)

Question for later: Is the computation based on Bernstein polynomials faster?

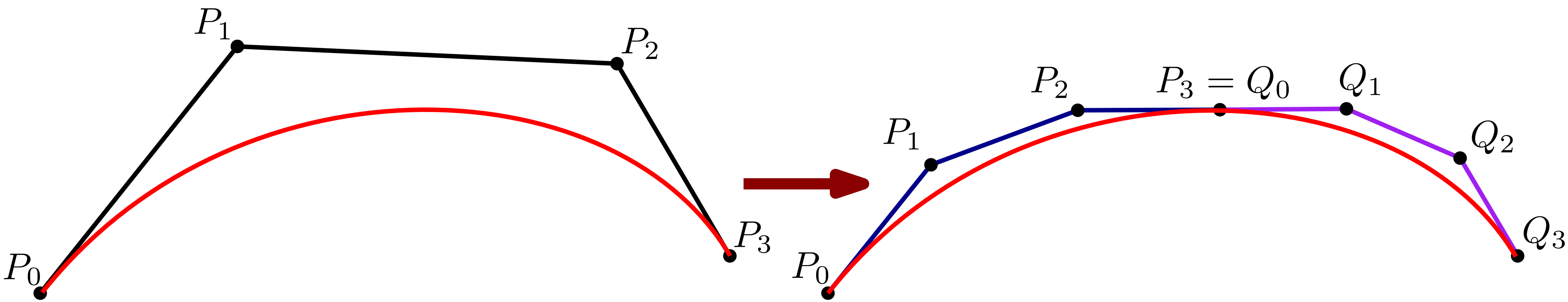
Using De Casteljau's to subdivide a curve

What if you want to add more points to a curve?
(We need this when we need more flexibility to design the curve)

Goal: increase number of points, but preserve shape of curve

One way to achieve this: subdivision

Subdivide degree-n curve into two curves, each of degree n

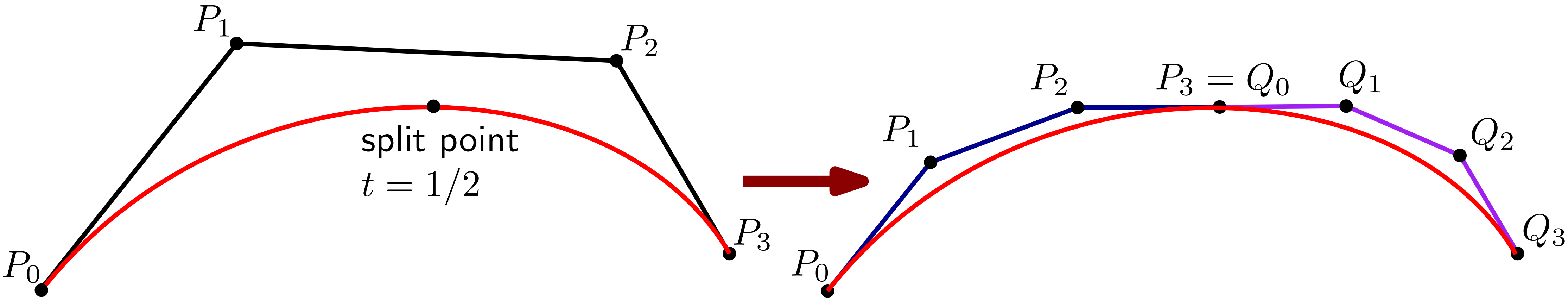


The new points come from the intermediate points of De Casteljau's algorithm!

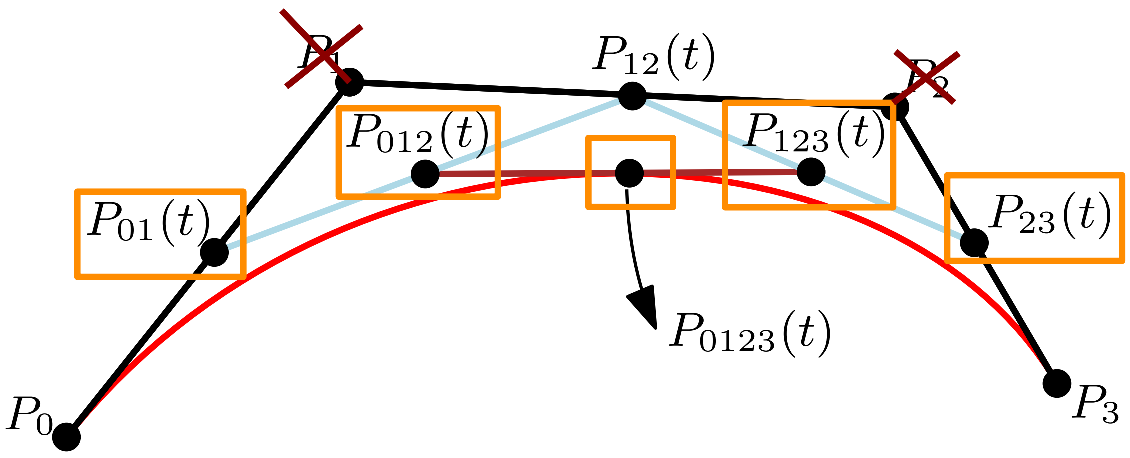
BÉZIER CURVES AS LINEAR INTERPOLATION

Using De Casteljau's to subdivide a curve

The new points come from the intermediate points of De Casteljau's algorithm!



Recall the De Casteljau algorithm ($t = 1/2$):



In general, the subdivision is done by:

- Discarding interior control points $P_1, \ldots, P_{n-1}$
- Adding $2n - 1$ points: the first and last points in each step of De Casteljau's

This method is also useful for **clipping**

Computation of a Bézier curve

Recall definition

$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$ recall that $0 \le i \le n$, $\binom{n}{i} = \frac{n!}{i!(n-i)!}$, and $0! = 1$

$P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t)$

Question: how many operations (say, products) needed to compute $P(1/2)$?

Simple speed-up: reuse computed values for different values of t

- Precalculate factorials, store them in table
- Then do the same with binomials
- Precalculate and store values t^i for $i = 0, \ldots, n$ and the necessary t values

The computation of $P(t)$ becomes

$\sum_{i=0}^{n} \boxed{\text{Table}_1[i,n] \times \text{Table}_2[t,i] \times \text{Table}_2[1-t, n-i]} \times P_i$

This can also be stored in a table, and reused for other points

Even faster: forward differences

step size

Idea: find a method to "jump" from one point in $P(t)$ to the next one ,$P(t+\boxed{\Delta})$, using only a few computations for each jump

Goal: find quantity dP such that $P(t+\Delta) = P(t) + dP$

If dP would exist, then we could do:

$$\begin{aligned} P(0) &= P_0 \\ P(0+\Delta) &= P(0) + dP = P_0 + dP \\ P(2\Delta) &= P(\Delta) + dP = P_0 + 2dP \\ P(i \cdot \Delta) &= P((i-1)\Delta) + dP = P_0 + i \cdot dP \end{aligned}$$

This would be very efficient!

Consider the *Taylor series* representation of $P(t)$:

$$P(t+\Delta) = P(t) + P'(t)\Delta + P''(t)\frac{\Delta^2}{2!} + P'''(t)\frac{\Delta^3}{3!} + \ldots$$

Infinite series
But becomes finite if $P(t)$ has constant degree!

Forward differences for cubic Bézier curve

$$P(t+\Delta) = P(t) + P'(t)\Delta + P''(t)\frac{\Delta^2}{2} + P'''(t)\frac{\Delta^3}{6}$$

For a cubic Bézier curve, we have

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

or, equivalently,

$$P(t) = at^3 + bt^2 + ct + d$$

where:

$a = 3(P_1 - P_2) - P_0 + P_3$, $b = 3(P_0 + P_2) - 6P_1$, $c = 3(P_1 - P_0)$, $d = P_0$

Now it is easy to differentiate:

$P'(t) = 3at^2 + 2bt + c \qquad P''(t) = 6at + 2b \qquad P'''(t) = 6a$

Recall, goal: find quantity dP such that $P(t+\Delta) = P(t) + dP \iff dP = \boxed{P(t+\Delta)} - P(t)$

$$\boxed{dP(t) = P(t+\Delta t) - P(t) = 3at^2\Delta + 2bt\Delta + c\Delta + 3at\Delta^2 + b\Delta^2 + a\Delta^3}$$

Problem: dP depends on t!

Forward differences for cubic Bézier curve

$dP(t) = 3at^2\Delta + 2bt\Delta + c\Delta + 3at\Delta^2 + b\Delta^2 + a\Delta^3$ degree-2 polynomial on t

Idea: use the same technique again, to get polynomial of degree 1...

...then do it again, to obtain polynomial of degree 0, i.e., constant!

```
procedure FASTCUBICBÉZIERSKETCH
    Precalculate certain quantities
    P ← P_0
    for t = 0 to 1 step Δ do
        Draw point P
        P ← P + dP
        dP ← dP + ddP
        ddP ← ddP + dddP
```

We need to figure out values for dP, ddP and $dddP$

$$ddP(t) = dP'(t)\Delta + \frac{dP''(t)\Delta^2}{2}$$

$$ddP(t) = (6at\Delta + 2b\Delta + 3a\Delta^2)\Delta + \frac{6a\Delta\Delta^2}{2} = 6at\Delta^2 + 2b\Delta^2 + 6a\Delta^3$$

degree-1 polynomial on t

One more time: let's compute $dddP(t)$

$dddP(t) = ddP'(t)\Delta = 6a\Delta^3$ $dddP$ is a constant! (does not depend on t)

BÉZIER CURVE COMPUTATION

Forward differences for cubic Bézier curve

Final code, trying to reuse computations as much as possible

procedure FastCubicBézier
    $Q_1 \leftarrow 3\Delta$
    $Q_2 \leftarrow Q_1 \cdot \Delta$ ▷ $3\Delta^2$
    $Q_3 \leftarrow \Delta^3$
    $Q_4 \leftarrow 2Q_2$ ▷ $6\Delta^2$
    $Q_5 \leftarrow 6Q_3$ ▷ $6\Delta^3$
    $Q_6 \leftarrow P_0 - 2P_1 + P_2$
    $Q_7 \leftarrow 3(P_1 - P_2) - P_0 + P_3$ ▷ a
    $P \leftarrow P_0$
    $dP \leftarrow (P_1 - P_0)Q_1 + Q_6 \cdot Q_2 + Q_7 \cdot Q_3$
    $ddP \leftarrow Q_6 \cdot Q_4 + Q_7 \cdot Q_5$
    $dddP \leftarrow Q_7 \cdot Q_5$
    for $t = 0$ to 1 step Δ **do**
        Draw point P
        $P \leftarrow P + dP$
        $dP \leftarrow dP + ddP$
        $ddP \leftarrow ddP + dddP$

The reduction in # of operations is huge: Ignoring the initialization, 3 sums for each evaluation of t

Matrix formulation

Bézier curves are often expressed in matrix form

For $n = 2$, we had $P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$

$$= ((1-t)^2, 2t(1-t), t^2)(P_0, P_1, P_2)^t$$

$$= (t^2, t, 1) \begin{pmatrix} & & \\ & ? & \\ & & \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \end{pmatrix}$$

Question: what is the matrix?

For n=3, we had $P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$

$$= (t^3, t^2, t, 1) \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Note: matrix notation is very practical, but not necessarily very efficient, or numerically stable

Question: how many sums/products to evaluate $P(t)$?

DEGREE ELEVATION

Another way to increase number of points

- Recall: curve subdivision took a degree-n curve and produced two curves of degree-n ($2n + 1$ control points in total)
- Alternative: add points (increase degree) while preserving curve

Why? More control points make it easier to edit, so its shape can be better adjusted

Degree elevation: given a degree-n curve $P_n(t)$, **add one control point** to produce an **identical** curve $P_{n+1}(t)$

original curve

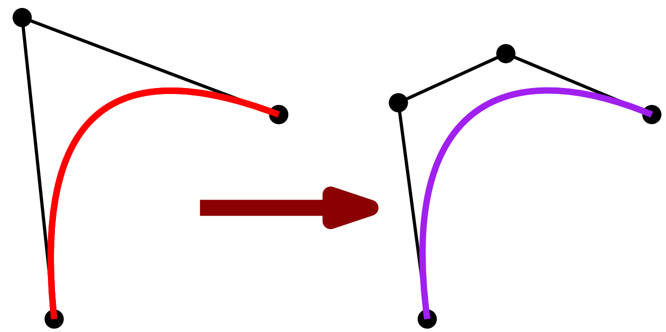
$P_n(t)$

P: $n+1$ control points

new curve

$P_{n+1}(t)$

Q: $n+2$ new control points



Adding one more point

original curve

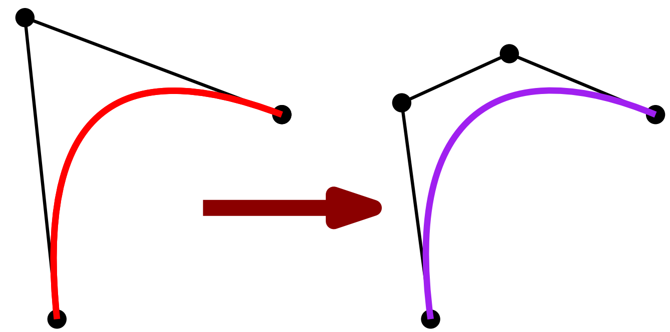
$P_n(t)$

P: $n+1$ control points

new curve

$P_{n+1}(t)$

Q: $n+2$ new control points



Producing control points for $P_{n+1}(t)$

With some basic algebraic tricks one can write $P(t)$ as an $(n+1)$-degree Bézier curve

Start from trivial identity $P(t) = (t + (1-t))P(t) = tP(t) + (1-t)P(t)$

Use that $P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i$, and extract coefficientes of new $(n+2)$ control points

Result:

$$P(t) = tP(t) + (1-t)P(t) = \sum_{i=0}^{n+1} \binom{n+1}{i} t^i (1-t)^{n+1-i} \boxed{\left(\frac{i}{n+1} P_{i-1} + \left(1 - \frac{i}{n+1}\right) P_i\right)}$$

Bézier curve of degree $(n+1)$!

$Q_i = \alpha_i P_{i-1} + (1-\alpha_i) P_i$

Note: here we assume $P_{-1} = 0$ and $P_{n+1} = 0$

new control points

Summary

The expression obtained for $P_{n+1}(t)$ is:

$$P_{n+1}(t) = \sum_{i=0}^{n+1} \binom{n+1}{i} t^i (1-t)^{n+1-i} \left(\frac{i}{n+1} P_{i-1} + \left(1 - \frac{i}{n+1}\right) P_i \right)$$

$$P_{n+1}(t) = \sum_{i=0}^{n+1} B_{n+1,i}(t) Q_i$$

where $Q_i = \alpha_i P_{i-1} + (1-\alpha_i) P_i$, $\alpha_i = \frac{i}{n+1}$
and $Q_0 = P_0$, $Q_{n+1} = P_n$

Example

degree-4 curve (5 control points)

degree-5 curve (6 control points)

degree-6 curve (7 control points)

degree-n curve ($n+1$ control points)

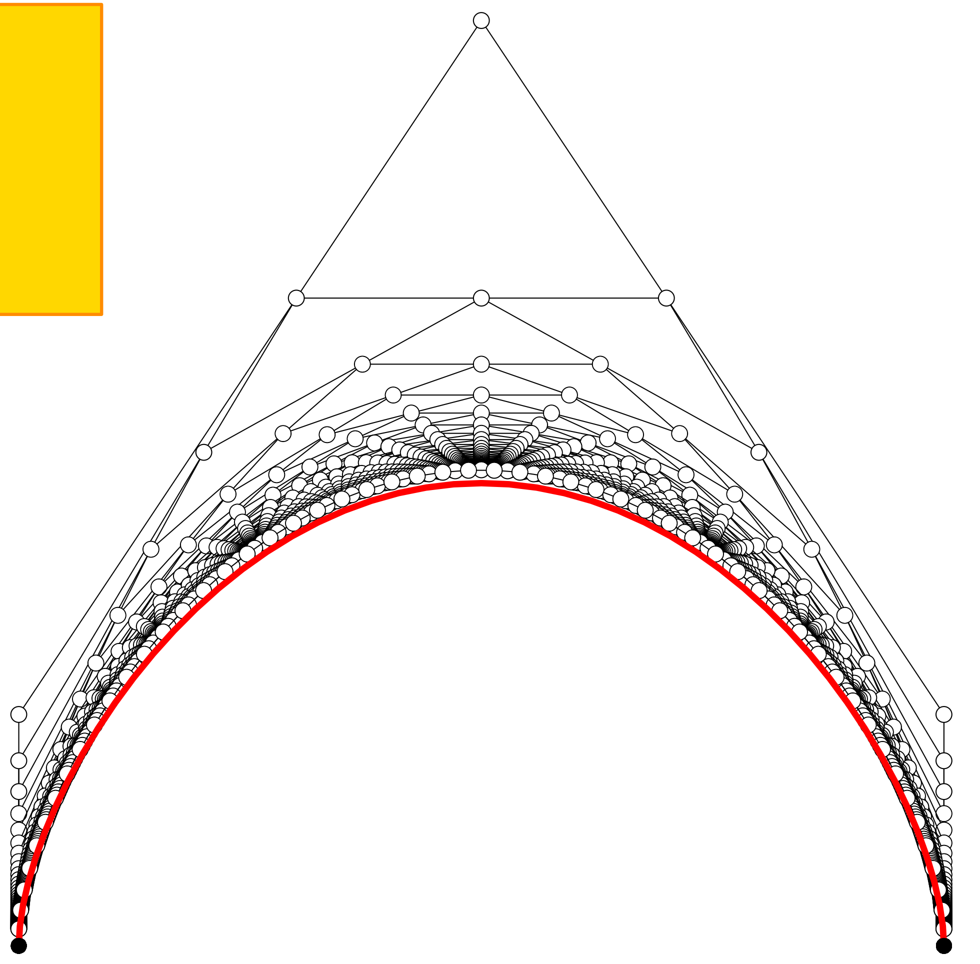
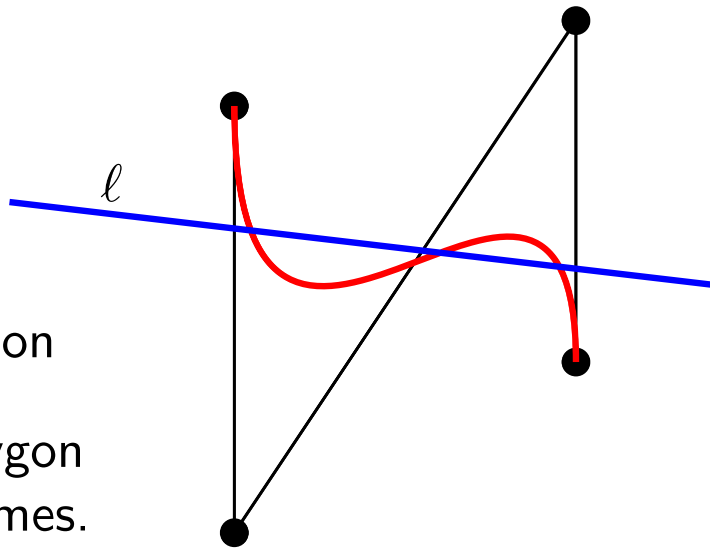


Figure from [G. Farin and D. Hansford, *The Essentials of CAGD*, AK Peters, 2000].

Back to the variation dimishing property

Recall the property: The number of intersections of any line with a Bézier curve is at most the number of intersections of the line with the control polygon



Proof sketch using degree elevation

Recall: linear interpolation is variation diminishing

Observation 2: degree elevation is an instance of linear interpolation

Let R_0 be the control polygon of $P(t)$, let R_1 be the control polygon after increasing the degree by one, and R_k after increasing it k times.

Let ℓ be a given line. Then R_k has no more intersections with ℓ than R_0.

Lemma: in the limit, as $k \to \infty$, R_k approaches the Bézier curve $P(t)$

Corollary: the Bézier curve $P(t)$ has no more intersections with ℓ than its control polygon

INTERPOLATION WITH BÉZIER CURVES

Goal: find Bézier curve that interpolates given points

- In general, the Bézier curve does not interpolate its control points
- There are situations in which the user may want to force the curve through some points

One way to interpolate

Goal: given **data points** $Q_0, \ldots, Q_n$, find **control points** $P_0, \ldots, P_n$ and values $t_0, \ldots, t_n$ such that for each i, $P(t_i) = Q_i$

This can be expressed as a system of equations, for example for $n = 3$:

$$P(t_i) = (1-t_i)^3 P_0 + 3t_i(1-t_i)^2 P_1 + 3t_i^2(1-t_i) P_2 + t_i^3 P_3 = Q_i \quad i = 0, 1, 2, 3$$

$$\begin{pmatrix} (1-t_0)^3 & 3t_0(1-t_0)^2 & 3t_0^2(1-t_0) & t_0^3 \\ (1-t_1)^3 & 3t_1(1-t_1)^2 & 3t_1^2(1-t_1) & t_1^3 \\ (1-t_2)^3 & 3t_2(1-t_2)^2 & 3t_2^2(1-t_2) & t_2^3 \\ (1-t_3)^3 & 3t_3(1-t_3)^2 & 3t_3^2(1-t_3) & t_3^3 \end{pmatrix} \underbrace{\boxed{\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}}}_{\text{unknowns}} = \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix}$$

We are free to choose the values of the t_is!

Example for $n = 3$

For example, take $t_i = i/n = i/3$

unknowns

$$\begin{pmatrix} (1-0)^3 & 3\cdot 0(1-0)^2 & 3\cdot 0^2(1-0) & 0^3 \\ (1-1/3)^3 & 3\cdot 1/3(1-1/3)^2 & 3\cdot 1/3^2(1-1/3) & 1/3^3 \\ (1-2/3)^3 & 3\cdot 2/3(1-2/3)^2 & 3\cdot (2/3)^2(1-2/3) & (2/3)^3 \\ (1-1)^3 & 3\cdot 1(1-1)^2 & 3\cdot 1^2(1-1) & 1^3 \end{pmatrix} \boxed{\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}} = \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{8}{27} & \frac{4}{9} & \frac{2}{9} & \frac{1}{27} \\ \frac{1}{27} & \frac{2}{9} & \frac{4}{9} & \frac{8}{27} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix} \longrightarrow \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix}$$

Concrete example, take $Q_0 = (0,0)$, $Q_1 = (1,1)$, $Q_2 = (2,1)$, $Q_3 = (3,0)$

$$\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 1 \\ 3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 3/2 \\ 2 & 3/2 \\ 3 & 0 \end{pmatrix}$$

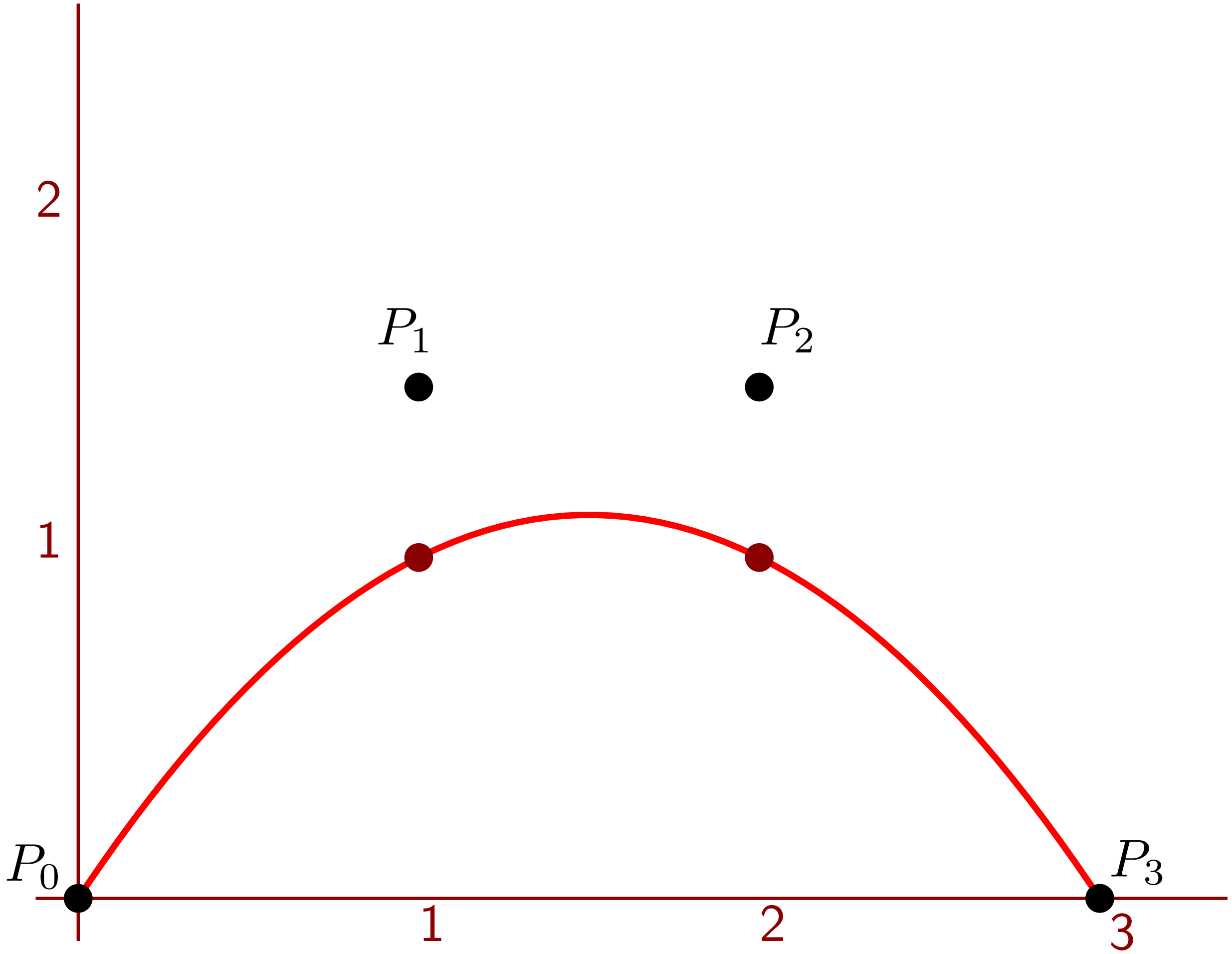
Example for $n = 3$

Concrete example, take $Q_0 = (0,0)$, $Q_1 = (1,1)$, $Q_2 = (2,1)$, $Q_3 = (3,0)$

Result:

$$\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 3/2 \\ 2 & 3/2 \\ 3 & 0 \end{pmatrix}$$

This is only one way to interpolate with Bézier curves, others are possible



Rational Bézier curves

Each control point has a weight, giving more flexibility to shape the curve

rational weights

$$P(t) = \sum_{i=0}^{n} P_i B_{n,i}(t) \qquad P(t) = \frac{\sum_{i=0}^{n} w_i P_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)} = \sum_{i=0}^{n} P_i \boxed{\left(\frac{w_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)}\right)}$$

Bézier curve Rational Bézier curve

- Weights are usually non-negative (otherwise denominator could be zero)

Advantages: why complicate things so much?

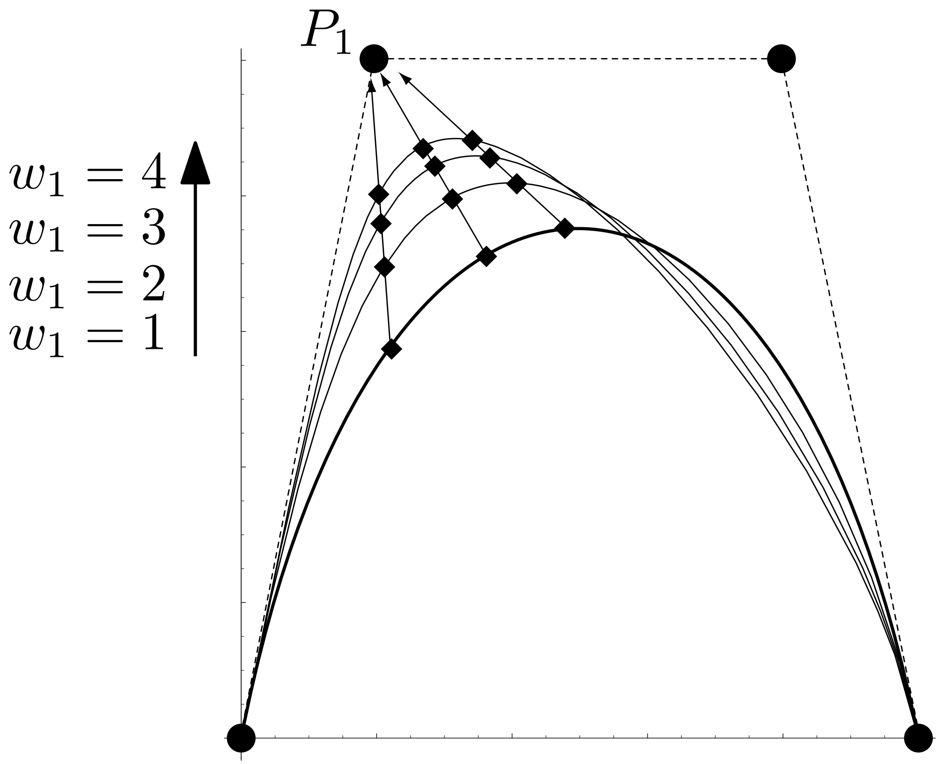
- Invariant under projections
- It can represent conic curves (impossible with Hermite or Bézier curves) (e.g., segments of circles, ellipses, hyperbolas and parabolas)

Understanding rational Bézier curves

Effect of the weights

$$P(t) = \frac{\sum_{i=0}^{n} w_i P_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)} = \sum_{i=0}^{n} P_i \left(\frac{w_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)} \right)$$

- If $w_i > 1$, the curve gets closer to P_i
- If $w_i < 1$, the curve moves away from P_i



changing weights

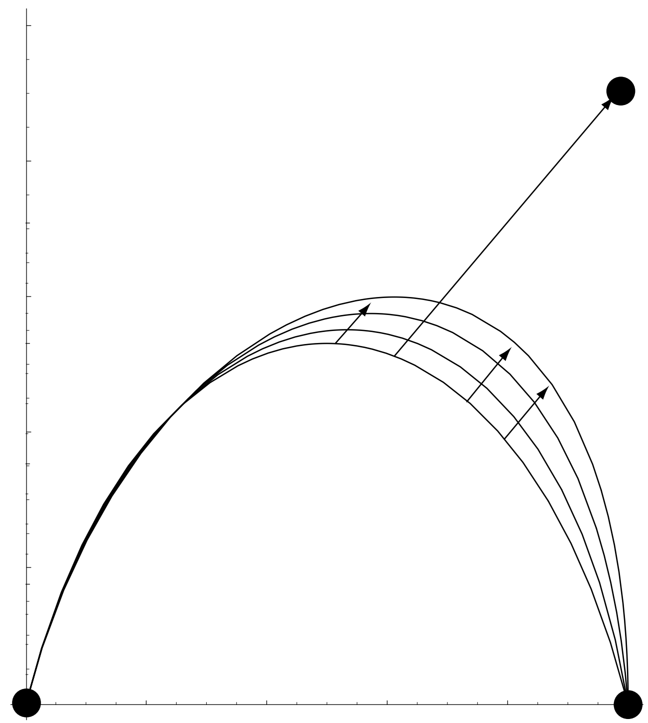
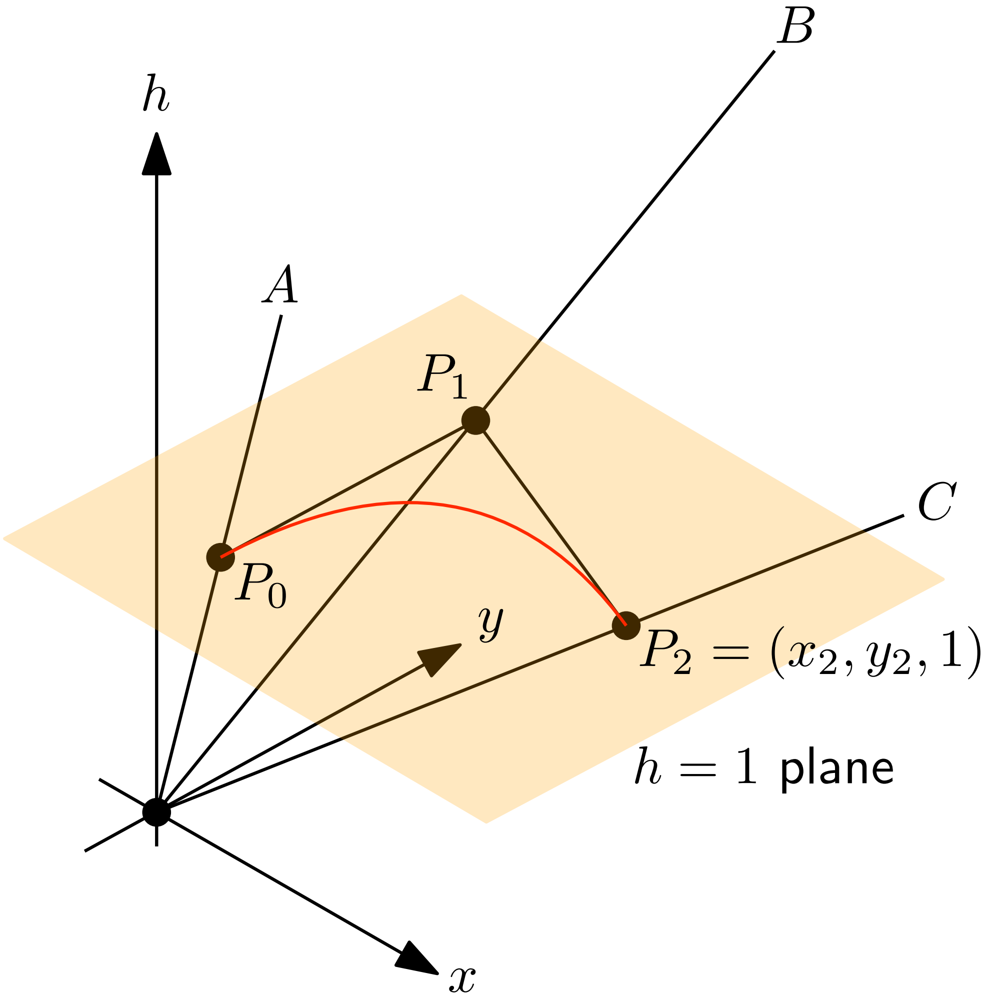

moving control point (weight fixed at 1)

Figure from book by Salomon (page 219)

Rational Bézier curves as curves in projective space



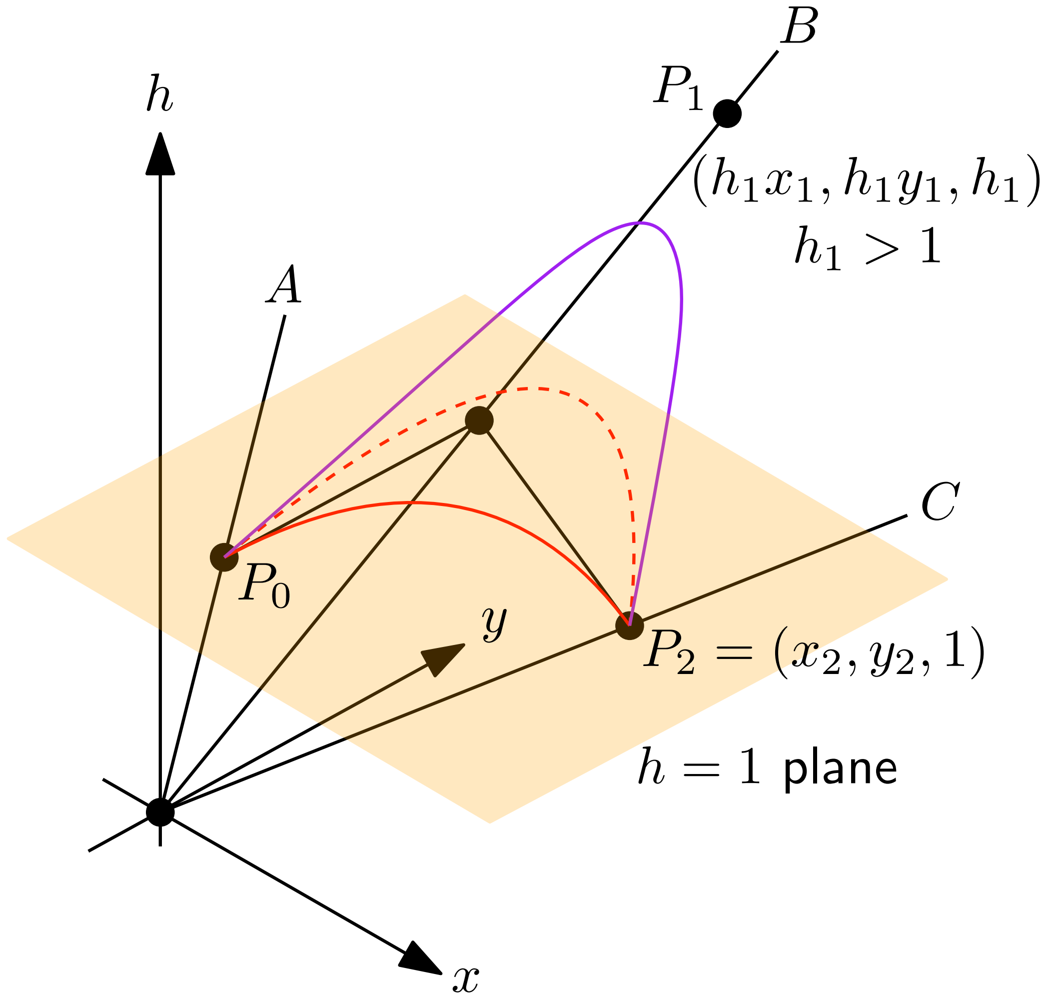


Figure adapted from book by Mortenson

Representing conics with rational Bézier curves

We can represent a conic curve *exactly* with a quadratic rational Bézier curve:

Theorem Consider a conic curve $C(t)$. Then there exist weights w_0, w_1, w_2 and control points P_0, P_1, P_2 such that

$$C(t) = \frac{w_0 P_0 B_{2,0}(t) + w_1 P_1 B_{2,1}(t) + w_2 P_2 B_{2,2}(t)}{w_0 B_{2,0}(t) + w_1 B_{2,1}(t) + w_2 B_{2,2}(t)}$$

Example

Take $w_0 = w_2 = 1$ and let $s = \frac{w_1}{1+w_1}$

- $s = 1/2$ produces a **parabolic** arc
- $s < 1/2$ produces an **elliptic** arc
- $s > 1/2$ produces a **hyperbolic** arc

for any three non-colinear control points