

Geometric Modeling


Summer Semester 2010

Blossoming and Polar Forms

Piecewise Polynomials Splines Revisited...

Overview...

Topics:

- Interpolation & Approximation
- Polynomial Spline Curves
- Blossoming and Polars 
 - The De Casteljau Algorithm
 - Polar Forms: Idea & Definition
 - Polynomial Splines Revisited
 - Bezier Splines
 - B-Splines

Geometric View:

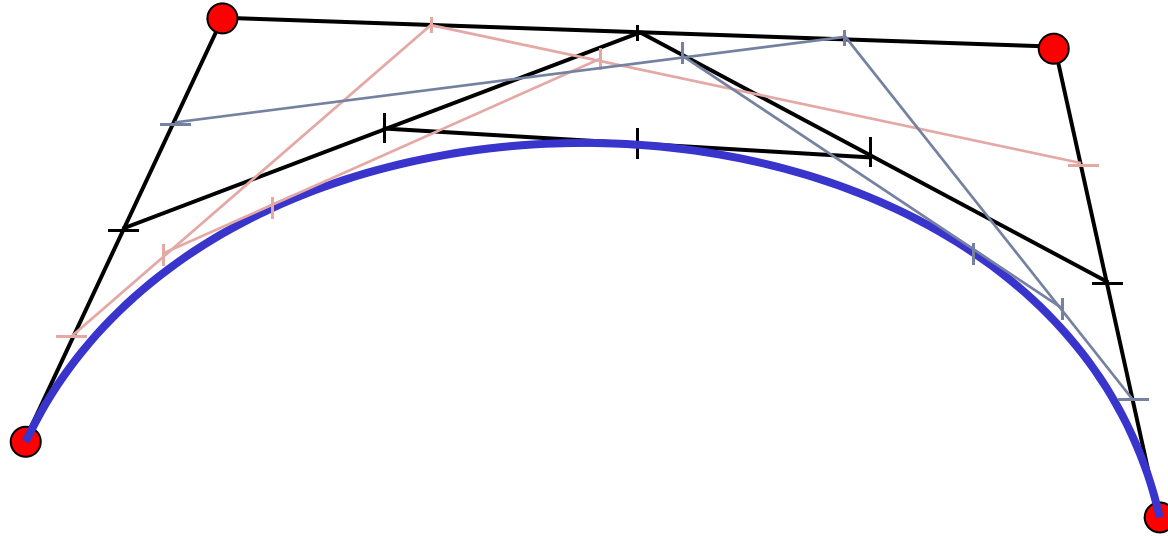
The De Casteljau Algorithm

De Casteljau Algorithm

Idea of Bezier splines can be formulated differently:

- Geometric view
 - Repeated linear interpolation with common parameter t
 - Implicitly creates polynomial in t
 - Degree depends on number of cascaded interpolations
- The geometric interpretation is usually more intuitive
 - Properties of the Bezier spline can be interpreted geometrically
 - Derivatives
 - Operations (e.g. subdivision)
 - ...
- We will now look at the corresponding algorithm...

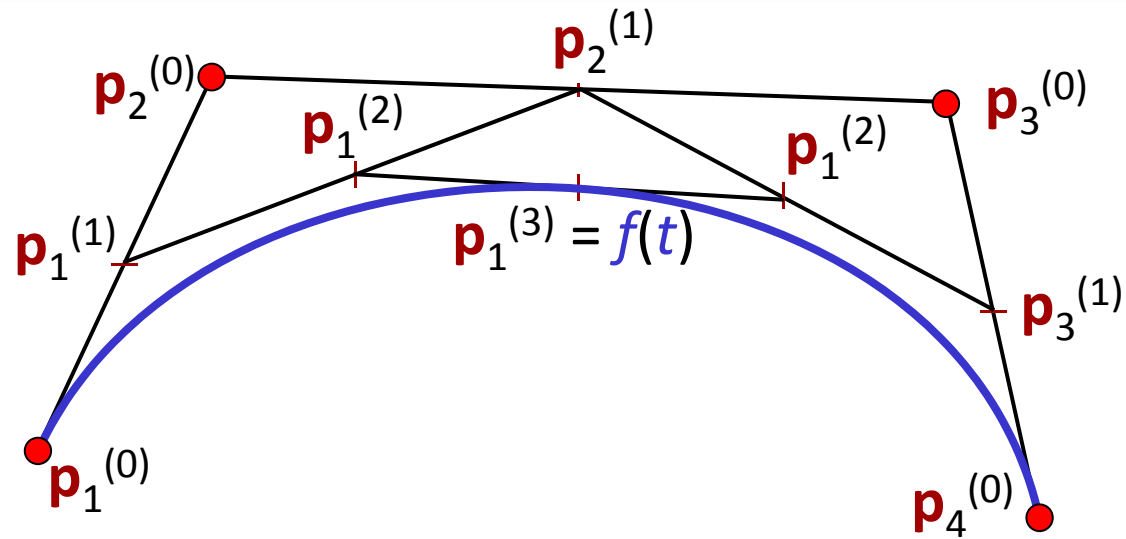
De Casteljau



De Casteljau Algorithm: Computes $f(t)$ for given t

- Bisect control polygon in ratio $t : (1 - t)$
- Connect the new dots with lines (adjacent segments)
- Interpolate again with the same ratio
- Iterate, until only one point is left

De Casteljau



Algorithm:

for $j = 0 \dots d-1$ **do**

for $i = 1 \dots d-j$ **do**

$$p_i^{(j)} = (1-t) \cdot p_i^{(j-1)} + t \cdot p_{i+1}^{(j-1)}$$

end for

end for

return $p_1^{(d)}$

Properties

Properties:

- The algorithm yields the same result as evaluating the spline using the Bernstein basis
- The algorithm uses only iterated convex combinations of control points
 - Numerically more stable than monomial evaluation
 - Easy to see:
 - Affine invariant
 - Convex hull property
- Open questions: How to geometrically interpret
 - Derivatives
 - Operations (knot insertion, degree elevation etc.)
 - ...

Polar Forms & Blossoms:

Idea & Definition

Affine Combinations

First: A quick look at “linear interpolation”

- Actually, the right name should be “affine interpolation”

Definition:

- An *affine combination* of n points $\in \mathbb{R}^d$ is given by:

$$\mathbf{p}_\alpha = \sum_{i=1}^n \alpha_i \mathbf{p}_i \quad \text{with} \quad \sum_{i=1}^n \alpha_i = 1$$

- A function f is set to be *affine* in its parameter \mathbf{x}_i , if:

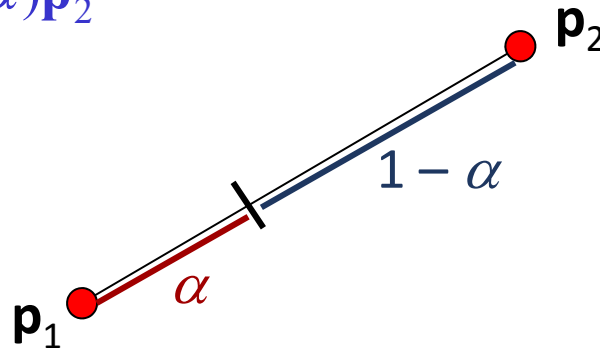
$$f\left(x_1, \dots, \sum_{k=1}^n \alpha_i x_i^{(k)}, \dots, x_m\right) = \sum_{k=1}^n \alpha_i f\left(x_1, \dots, x_i^{(k)}, \dots, x_m\right) \quad \text{for} \quad \sum_{i=1}^n \alpha_i = 1$$

Affine Combinations

Examples:

- Linear (affine) interpolation of 2 points:

$$\mathbf{p}_\alpha = \alpha \mathbf{p}_1 + (1 - \alpha) \mathbf{p}_2$$



Affine Combinations

Examples:

- Barycentric combinations of 3 points (“barycentric coordinates”)

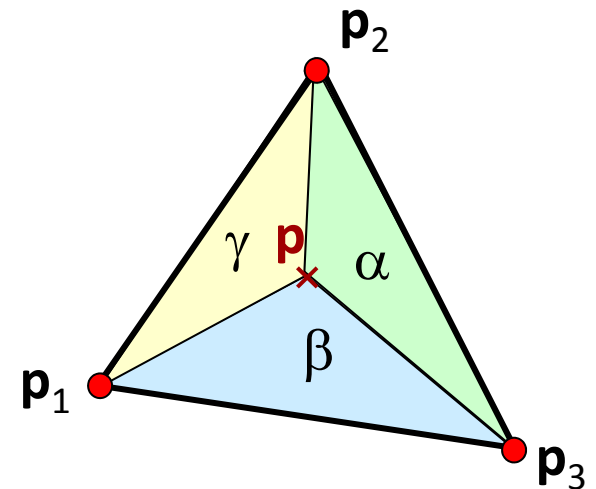
$$\mathbf{p} = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3, \text{ with } \alpha + \beta + \gamma = 1$$

Properties:

$$\gamma = 1 - \alpha - \beta$$

$$\alpha = \frac{\text{area}(\Delta(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \beta = \frac{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \gamma = \frac{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}$$

Transformation to barycentric coordinates is a linear map (heights in triangles).



Formalizing the Idea

Idea: Express (piecewise) polynomial curves as iterated linear (affine) interpolations

First try:

- A polynomial: $p(t) = at^3 + bt^2 + ct + d$
- Can be written as: $p(t) = a \cdot t \cdot t \cdot t + b \cdot t \cdot t + c \cdot t + d$
- Interpret each variable t a separate parameter:
$$\mathbf{p}(t_1, t_2, t_3) = \mathbf{a} \cdot t_1 \cdot t_2 \cdot t_3 + \mathbf{b} \cdot t_1 \cdot t_2 + \mathbf{c} \cdot t_1 + \mathbf{d}$$
 - t_1 moves linearly in direction $(\mathbf{a} + \mathbf{b} + \mathbf{c})$
 - t_2 in direction $(\mathbf{a} + \mathbf{b})$
 - t_3 in direction \mathbf{a}
- Problems: fixed directions, many representations

Polar Forms

Improved solution: Polar Forms / Blossoms

A *polar form* or *blossom* f of a polynomial F of degree d is a function in d variables:

$$F: \mathbb{R} \rightarrow \mathbb{R}$$

$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

with the following properties:

- *Diagonality*: $f(t, t, \dots, t) = F(t)$
- *Symmetry*: $f(t_1, t_2, \dots, t_d) = f(t_{\pi(1)}, t_{\pi(2)}, \dots, t_{\pi(d)})$
for all permutations of indices π .
- *Multi-affine*: $\sum \alpha_k = 1$
 $\Rightarrow f(t_1, t_2, \dots, \sum \alpha_k t_i^{(k)}, \dots, t_d)$
 $= \alpha_1 f(t_1, t_2, \dots, t_i^{(1)}, \dots, t_d) + \dots + \alpha_n f(t_1, t_2, \dots, t_i^{(n)}, \dots, t_d)$

Polar Forms

Based on the same idea as on slide 9:

- Model polynomial as multi-affine function (multi-affinity property)
- Plugging in a common parameter to obtain the original polynomial
- *New*: Symmetry property – makes the solution unique
 - There is exactly one polar form for each polynomial
 - This standardization makes different polars “compatible”, we can compare them with each other
 - We will see how this works in a few slides...

Properties

Properties of polar forms:

- The mapping from polynomials to their polar forms is one-to-one:
 - For each polar form $f(t_1, t_2, \dots, t_n)$, a unique polynomial $F(t, t, \dots, t)$ exists
 - For each polynomial F , a unique polar form $f(t_1, t_2, \dots, t_n)$ exists

Properties

Properties of polar forms:

- Polar forms of monomials:

- Degree 0: $1 \rightarrow 1$

- Degree 1: $1 \rightarrow 1, t \rightarrow t$

- Degree 2: $1 \rightarrow 1, t \rightarrow \frac{t_1 + t_2}{2}, t^2 \rightarrow t_1 t_2$

- Degree 3: $1 \rightarrow 1, t^2 \rightarrow \frac{t_1 t_2 + t_2 t_3 + t_1 t_3}{3},$
 $t \rightarrow \frac{t_1 + t_2 + t_3}{3}, t^3 \rightarrow t_1 t_2 t_3$

Properties

Properties of polar forms:

- Polar forms of monomials:
 - Degree 0: $f = c_0$
 - Degree 1: $f(t_1) = c_0 + c_1 t_1$
 - Degree 2: $f(t_1, t_2) = c_0 + c_1 \frac{t_1 + t_2}{2} + c_2 t_1 t_2$
 - Degree 3: $f(t_1, t_2, t_3) = c_0 + c_1 \frac{t_1 + t_2 + t_3}{3} + c_2 \frac{t_1 t_2 + t_2 t_3 + t_1 t_3}{3} + c_3 t_1 t_2 t_3$

Properties

General Case:

- $$f(t_1, \dots, t_n) = \sum_{i=0}^n c_i \binom{n}{i}^{-1} \sum_{\substack{S \subseteq \{1..n\}, \\ |S|=i}} \prod_{j \in S} t_j$$

- The c_i are the monomial coefficients.
- Idea: Use all possible subsets of t_j to make it symmetric.
- This solution is unique.
- Without the symmetry property, there would be a large number of solutions.

Generalizations

Blossoms for polynomial curves (points as output):

- Polar form of a polynomial curve of degree d :

$$\begin{array}{l} \mathbf{F}: \mathbb{R} \rightarrow \mathbb{R}^n \\ \mathbf{f}: \mathbb{R}^d \rightarrow \mathbb{R}^n \end{array} \leftarrow \begin{array}{l} \text{new} \\ \text{new} \end{array}$$

- Required Properties:

- Diagonality: $\mathbf{f}(t, t, \dots, t) = \mathbf{F}(t)$

- Symmetry: $\mathbf{f}(t_1, t_2, \dots, t_d) = \mathbf{f}(t_{\pi(1)}, t_{\pi(2)}, \dots, t_{\pi(d)})$
for all permutations of indices π .

- Multi-affine: $\sum \alpha_k = 1$

$$\Rightarrow \mathbf{f}(t_1, t_2, \dots, \sum \alpha_k t_i^{(k)}, \dots, t_d)$$

$$= \alpha_1 \mathbf{f}(t_1, t_2, \dots, t_i^{(1)}, \dots, t_d) + \dots + \alpha_n \mathbf{f}(t_1, t_2, \dots, t_i^{(n)}, \dots, t_d)$$

Generalizations

Blossoms with points as arguments:

- Polar form degree d with points as input and output:

$$\begin{array}{l} \mathbf{F}: \mathbb{R}^m \rightarrow \mathbb{R}^n \\ \mathbf{f}: \mathbb{R}^{d \times m} \rightarrow \mathbb{R}^n \end{array} \quad \text{new}$$

- Required Properties:

- Diagonality: $\mathbf{f}(\mathbf{t}, \mathbf{t}, \dots, \mathbf{t}) = \mathbf{F}(\mathbf{t})$

- Symmetry: $\mathbf{f}(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_d) = \mathbf{f}(\mathbf{t}_{\pi(1)}, \mathbf{t}_{\pi(2)}, \dots, \mathbf{t}_{\pi(d)})$
for all permutations of indices π .

- Multi-affine: $\sum \alpha_k = 1$

$$\Rightarrow \mathbf{f}(\mathbf{t}_1, \mathbf{t}_2, \dots, \sum \alpha_k \mathbf{t}_i^{(k)}, \dots, \mathbf{t}_d)$$

$$= \alpha_1 \mathbf{f}(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_i^{(1)}, \dots, \mathbf{t}_d) + \dots + \alpha_n \mathbf{f}(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_i^{(n)}, \dots, \mathbf{t}_d)$$

Generalizations

Vector arguments

- We will have to distinguish between *points* and *vectors* (differences of points)
- Use “hat” notation $\hat{\mathbf{v}} = \mathbf{p} - \mathbf{q}$ to denote vectors (differences of points)
- Also defined in the one dimensional case (vectors in \mathbb{R})
- One vector: $\hat{1} = 1 - 0$, $\hat{\mathbf{1}} = [1, \dots, 1]^T - \mathbf{0}$
- Define shorthand notation (recursive):

$$f(\underbrace{t_1, \dots, t_{n-k}}_{n-k}, \underbrace{\hat{v}_1, \dots, \hat{v}_k}_k) := f(\underbrace{t_1, \dots, t_{n-k}}_{n-k}, p_1, \underbrace{\hat{v}_2, \dots, \hat{v}_k}_{k-1}) - f(\underbrace{t_1, \dots, t_{n-k}}_{n-k}, q_1, \underbrace{\hat{v}_2, \dots, \hat{v}_k}_{k-1})$$

Properties

Derivatives of blossoms:

- $$f(t_1, \dots, t_n) = \sum_{i=0}^n c_i \binom{n}{i}^{-1} \sum_{\substack{S \subseteq \{1..n\}, \\ |S|=i}} \prod_{j \in S} t_j$$

- The c_i are related to the derivatives at $t = 0$.

- Hence:
$$c_i = \frac{\frac{d^k}{dt^k} F(0)}{k!} = \binom{n}{k} f(\underbrace{0, \dots, 0}_{n-k}, \underbrace{\hat{1}, \dots, \hat{1}}_k)$$

- In general:
$$\frac{d^k}{dt^k} F(t) = \frac{n!}{(n-k)!} f(\underbrace{t, \dots, t}_{n-k}, \underbrace{\hat{1}, \dots, \hat{1}}_k)$$

Example

Example:

$$f(t_1, t_2, t_3) = c_0 + c_1 \frac{t_1 + t_2 + t_3}{3} + c_2 \frac{t_1 t_2 + t_2 t_3 + t_1 t_3}{3} + c_3 t_1 t_2 t_3$$

$$f'(t) = \frac{3!}{2!} \left[\left(c_0 + c_1 \frac{1+t+t}{3} + c_2 \frac{1t+tt+1t}{3} + c_3 1tt \right) - \left(c_0 + c_1 \frac{0+t+t}{3} + c_2 \frac{tt}{3} \right) \right]$$

$$= 3 \left(c_1 \frac{1}{3} + c_2 \frac{2t}{3} + c_3 tt \right)$$

$$= 3c_3 t^2 + 2tc_2 + c_1$$

Continuity Condition

Theorem: Continuity condition for polynomials

The following statements are equivalent:

- F and G are C^k -continuous at t
- $\forall t_1, \dots, t_k: f(t, \dots, t, t_1, \dots, t_k) = g(t, \dots, t, t_1, \dots, t_k)$
- $f(t, \dots, t, \underbrace{\hat{1}, \dots, \hat{1}}_{k\text{-times}}) = g(t, \dots, t, \underbrace{\hat{1}, \dots, \hat{1}}_{k\text{-times}})$ *)

$$\begin{aligned} *) \quad 2 \Leftrightarrow 3: f(t, \dots, t, t_1) &= f(t, \dots, t, (t_1 - 0)) \\ &= t_1 f(t, \dots, t, 1) - f(t, \dots, t, 0) \\ &= t_1 f(t, \dots, t, \hat{1}) \end{aligned}$$

Continuity Condition

Examples:


- $\forall t_1, t_2, t_3: f(t_1, t_2, t_3) = g(t_1, t_2, t_3) \Rightarrow$ same curve
- $\forall t_1, t_2: f(t_1, t_2, t) = g(t_1, t_2, t) \Rightarrow C^2$ at t
- $\forall t_1: f(t_1, t, t) = g(t_1, t, t) \Rightarrow C^1$ at t
- $f(t, t, t) = g(t, t, t) \Rightarrow C^0$ at t

Raising the Degree

Raising the degree of a blossom:

- Can we directly construct a polar form with degree elevated by one from a lower degree one, without changing the polynomial?
- [other than transforming to monomials, adding $0 \cdot t^{d+1}$, and transforming back?]

Solution:

- Given: $f(t_1, \dots, t_d)$
- We obtain: $f^{(+1)}(t_1, \dots, t_{d+1}) = \frac{1}{d+1} \sum_{i=1}^{d+1} f(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_{d+1})$


Raising the Degree

Proof:

$$\begin{aligned}\forall t: f^{(+1)}(t, \dots, t) &= \frac{1}{d+1} \sum_{i=1}^{d+1} f(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_{d+1}) \Big|_{t_1 = \dots = t_{d+1} = t} \\ &= \frac{1}{d+1} \sum_{i=1}^{d+1} f(t, \dots, t) \\ &= f(t, \dots, t)\end{aligned}$$

$$\Rightarrow F^{(+1)}(t) = F(t)$$

Polars and Control Points

Interpretation (Examples):

- Multi-variate function: $f(t_1, t_2, t_3)$
 - Describes a point depending on three parameters
 - Where $f(t_1, t_2, t_3)$ moves for changing (t_1, t_2, t_3) depends on f (think of storing monomial coefficients inside)
- Polynomial value: $f(1.5, 1.5, 1.5)$
 - One value of the polynomial curve: $F(1.5)$
- Off-curve points: $f(1, 2, 3)$
 - Evaluate points not necessarily on the polynomial curve
 - Question: What meaning do various off-curve points have?
 - We will use off-curve points as control points

Polars and Control Points

Interpretation (Examples):

- Specifying $\mathbf{f}(t_1, t_2, t_3)$:
 - Assume, f is not known yet
 - We want to determine a polar (i.e. a polynomial)
- On curve points:
 $\{\mathbf{f}(0,0,0) = \mathbf{x}_0, \mathbf{f}(1,1,1) = \mathbf{x}_1, \mathbf{f}(2,2,2) = \mathbf{x}_2, \mathbf{f}(3,3,3) = \mathbf{x}_3\}$
 - Degree d polynomial has $d+1$ degrees of freedom
 - We know already how to do polynomial interpolation
- Off-curve points:
 $\{\mathbf{f}(1,1,1) = \mathbf{x}_{111}, \mathbf{f}(1,2,3) = \mathbf{x}_{123}, \mathbf{f}(2,3,4) = \mathbf{x}_{234}, \mathbf{f}(3,3,3) = \mathbf{x}_{333}\}$
 - We can also use off-curve points to specify the polar/polynomial
 - This is the main motivation for the whole formalism

Polynomial Splines Revisited:

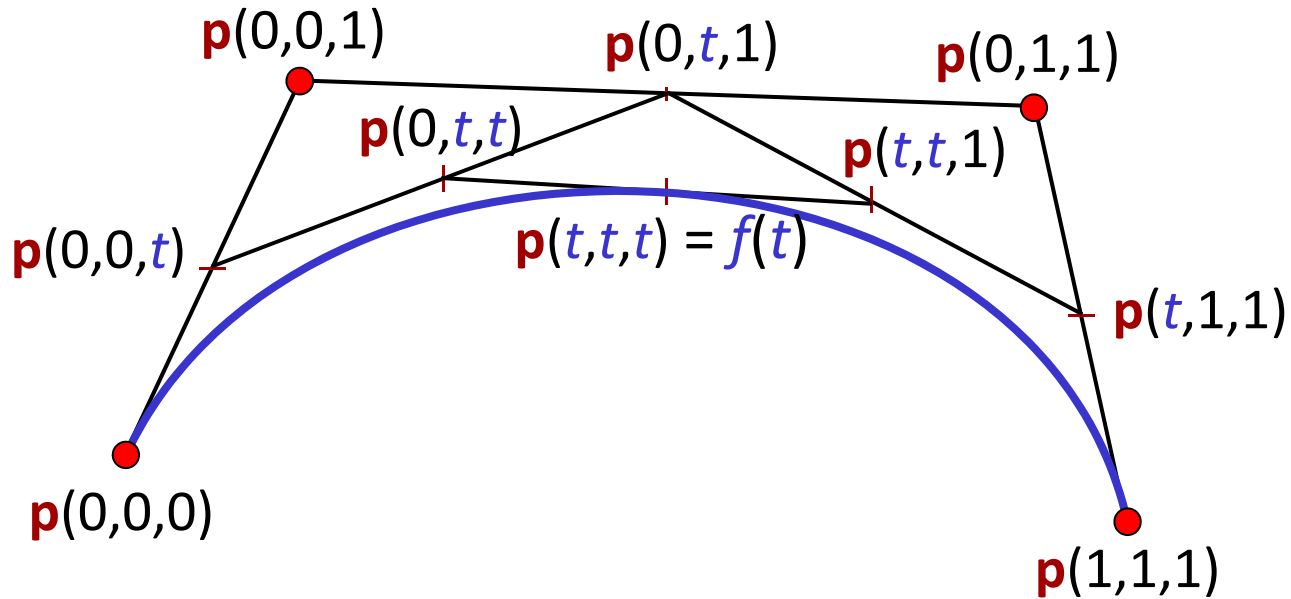
Bezier Splines

De Casteljau Algorithm

The de Casteljau algorithm is simple to state with blossoms:

- We just have to exchange the labels
- Then use the multi-affinity property in order to compute intermediate points
- With this view, we can easily show that the de Casteljau algorithm is equivalent to the formulation based on Bernstein polynomials

De Casteljau



Bezier control points: $p(0,0,0)$, $p(0,0,1)$, $p(0,1,1)$, $p(1,1,1)$

Analysis

Transforming a polar to the Bernstein basis:

$$\begin{aligned} \mathbf{f}(t, \dots, t) &= (1-t)\mathbf{f}(t, \dots, t, 0) + t\mathbf{f}(t, \dots, t, 1) \\ &= (1-t)[(1-t)\mathbf{f}(t, \dots, t, 0, 0) + t\mathbf{f}(t, \dots, t, 0, 1)] + t[(1-t)\mathbf{f}(t, \dots, t, 1, 0) + t\mathbf{f}(t, \dots, t, 1, 1)] \\ &= (1-t)^2\mathbf{f}(t, \dots, t, 0, 0) + 2t(1-t)\mathbf{f}(t, \dots, t, 0, 1) + t^2\mathbf{f}(t, \dots, t, 1, 1) \\ &= \dots \\ &= \sum_{i=0}^n \binom{n}{i} t^i (1-t)^{n-i} \mathbf{f}(\underbrace{0, \dots, 0}_{n-i}, \underbrace{1, \dots, 1}_i) \end{aligned}$$

Analysis

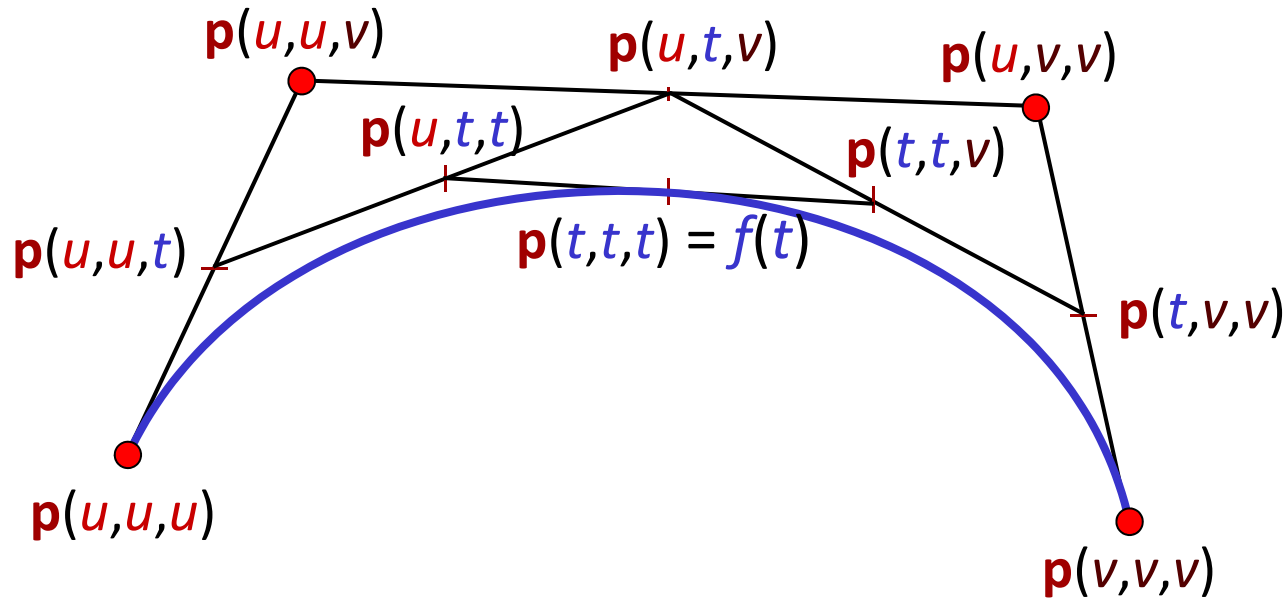
De Casteljau Algorithm: Performs this in reverse order

- Bezier points: $\mathbf{p}_i^{(0)}(t) = \mathbf{f}(\underbrace{0, \dots, 0}_{d-i}, \underbrace{1, \dots, 1}_i)$
- Intermediate points: $\mathbf{p}_i^{(j)}(t) = \mathbf{f}(\underbrace{0, \dots, 0}_{d-i-j}, \underbrace{1, \dots, 1}_i, \underbrace{t, \dots, t}_j)$
- Recursive computation:

$$\begin{aligned}\mathbf{p}_i^{(j)}(t) &= \mathbf{f}(\underbrace{0, \dots, 0}_{d-i-j}, \underbrace{t, \dots, t}_j, \underbrace{1, \dots, 1}_i) \\ &= (1-t)\mathbf{f}(\underbrace{0, \dots, 0}_{d-i-j+1}, \underbrace{t, \dots, t}_{j-1}, \underbrace{1, \dots, 1}_i) + t\mathbf{f}(\underbrace{0, \dots, 0}_{d-i-j}, \underbrace{t, \dots, t}_{j-1}, \underbrace{1, \dots, 1}_{i+1}) \\ &= (1-t)\mathbf{p}_i^{(j-1)}(t) + t\mathbf{p}_{i+1}^{(j-1)}(t)\end{aligned}$$

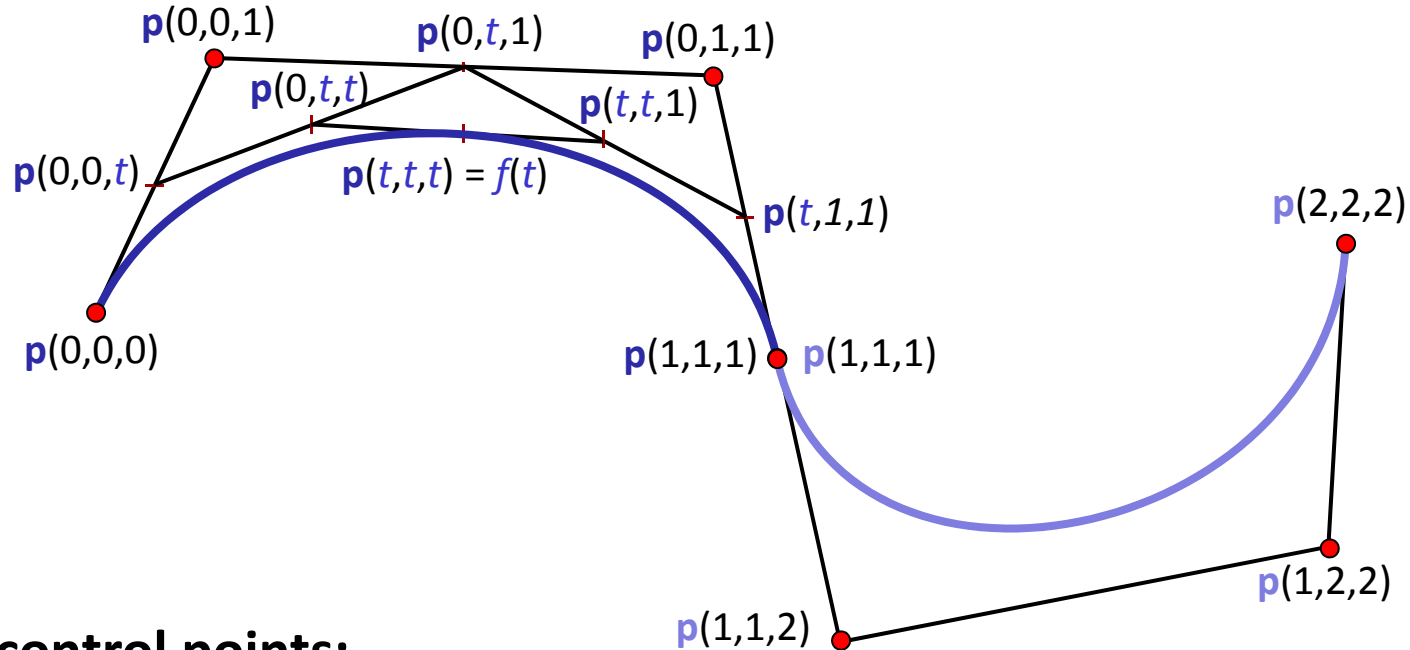
Consequence: Bernstein / de Casteljau lead to the same result

Generalized Parameter Intervals



Bezier control points: $p(u, u, u)$, $p(u, u, v)$, $p(u, v, v)$, $p(v, v, v)$

Multiple Segments



Bezier control points:

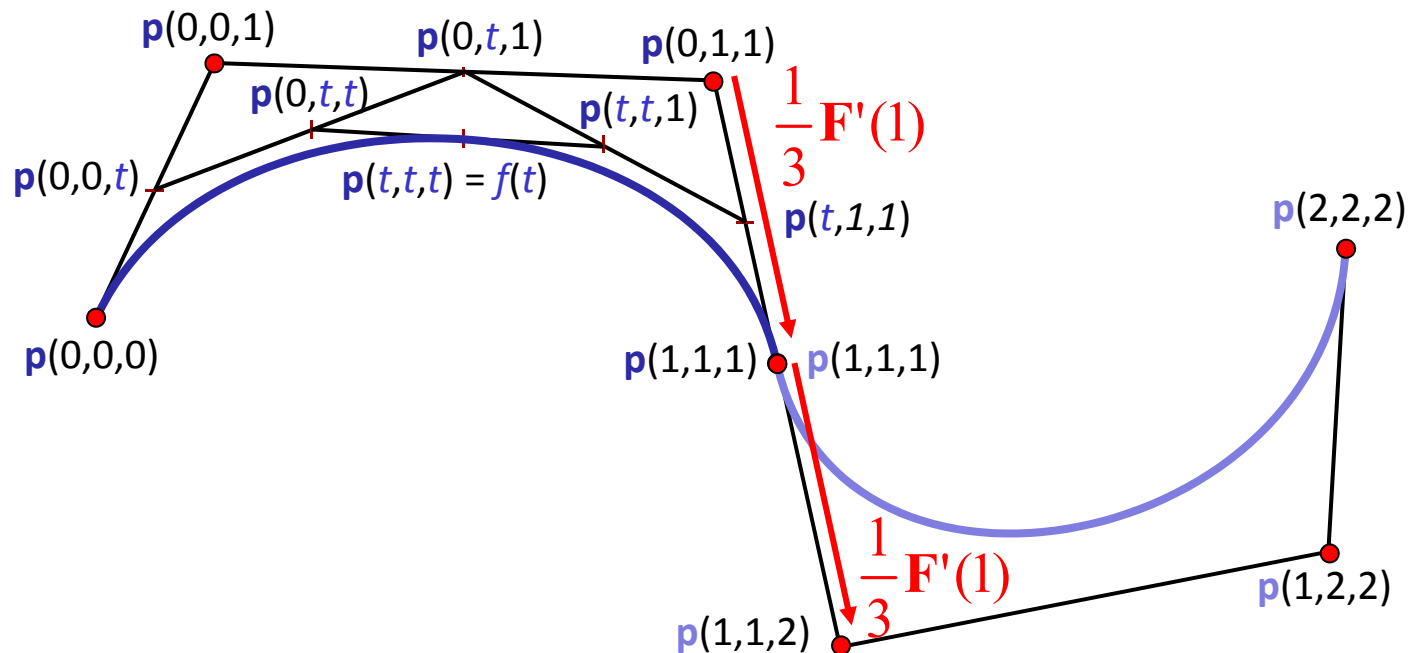
$p(0,0,0)$, $p(0,0,1)$, $p(0,1,1)$, $p(1,1,1) = p(1,1,1)$, $p(1,1,2)$, $p(1,2,2)$, $p(2,2,2)$

Two Curve Segments:

$\{p(0,0,0), p(0,0,1), p(0,1,1), p(1,1,1)\}$, $\{p(1,1,1), p(1,1,2), p(1,2,2), p(2,2,2)\}$

Remark: no interpolation between different segments
(e.g.: combination of $p(0,1,1)$ and $p(2,1,1)$ is not defined)

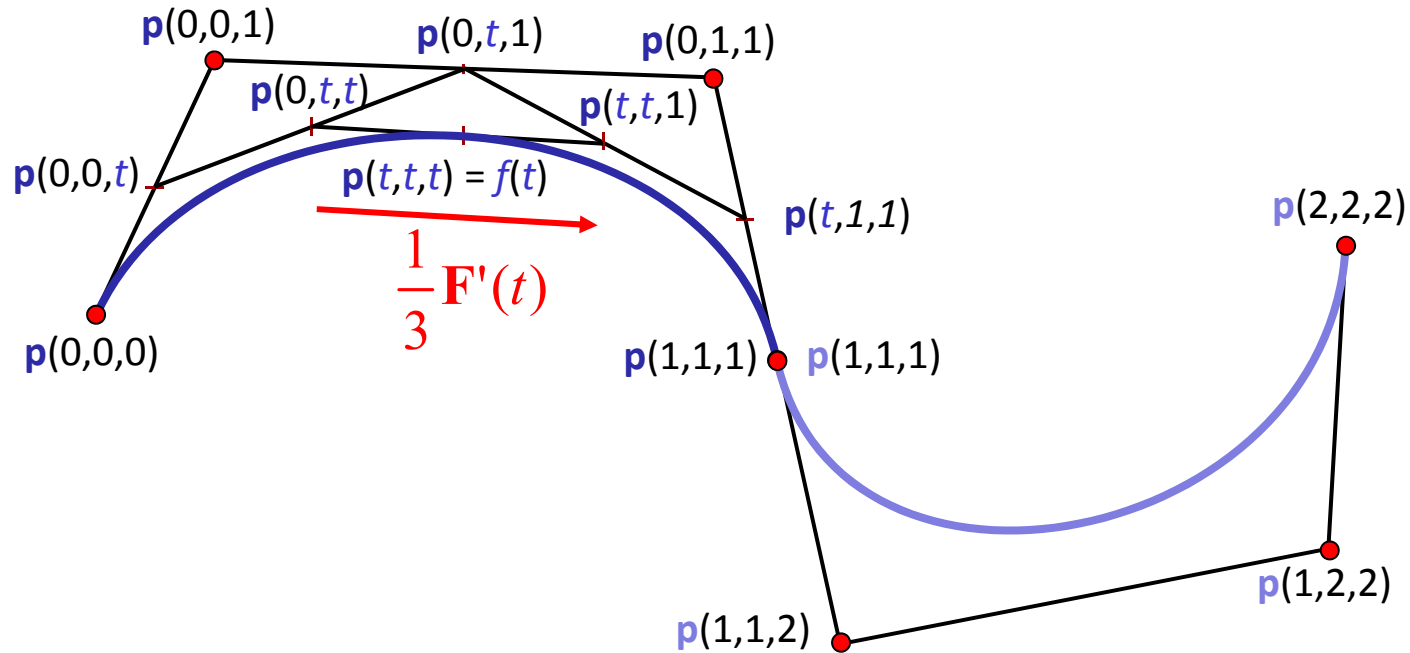
More Observations



Derivatives:

- $\frac{d}{dt} F(t) = d f(\underbrace{t, \dots, t}_{d-1}, \hat{1}) = d(f(t, \dots, t, t+1) - f(t, \dots, t, t))$ (degree d)
- C^1 Continuity condition follows

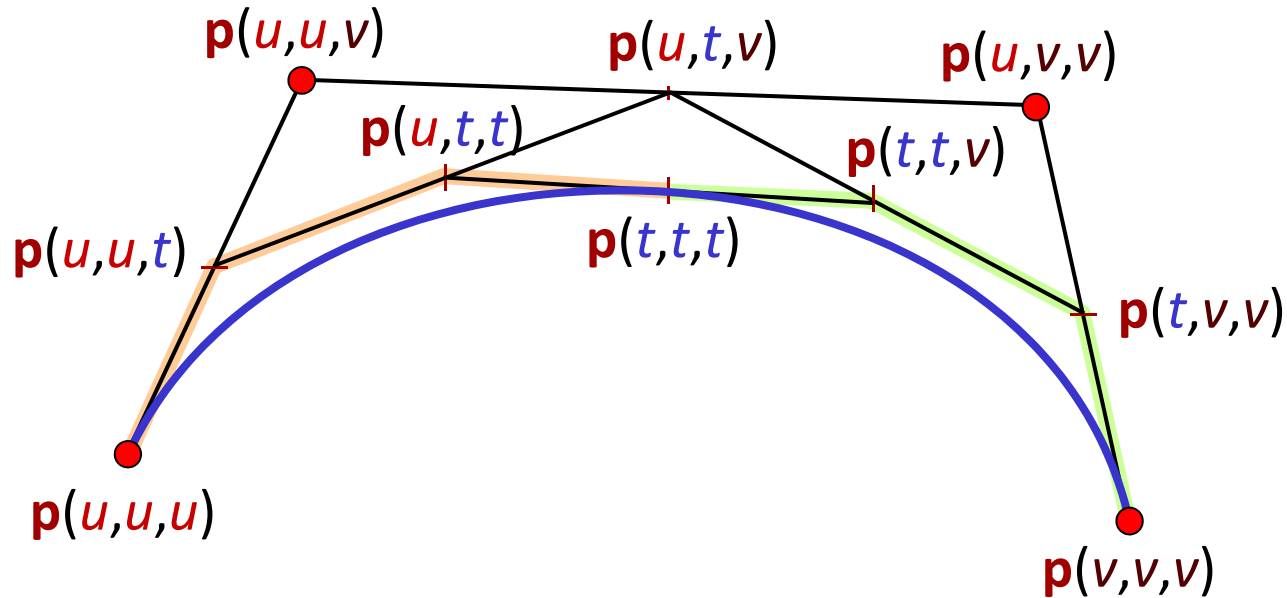
More Observations



Derivatives:

- De Casteljau Algorithm computes tangent vectors at any point as a byproduct
- Proportional to last line segment that is bisected

More Observations



Subdivision:

- After each de Casteljau step, we obtain two new control polygons left and right of $f(t)$ describing the same curve.
- We can divide a segment into two.
- Recursive subdivision can be used for rendering

Observations

Remark: The de Casteljau algorithm for computing

- Derivatives
 - at endpoints
 - at inner points t
- Subdivisions

hold for Bezier curves of arbitrary degree $d \geq 1$.


(General degree derivatives: $1/d \mathbf{F}'(t)$)

More Bezier Curve Properties...

General degree elevation:

- Increase the degree of a Bezier curve segment by one.
- What are the new control points?

Polar forms:

- Old curve: $\mathbf{b}(t_1, \dots, t_d)$
- New curve: $\mathbf{b}^{(+1)}(t_1, \dots, t_{d+1}) = \frac{1}{d+1} \sum_{i=1}^{d+1} \mathbf{b}(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_{d+1})$
 leave out t_i

Degree Elevation

$$\mathbf{b}^{(+1)}(0,\dots,0) = \frac{1}{d+1} \sum_{i=1}^{d+1} \mathbf{b}(0,\dots,0) = \mathbf{b}(0,\dots,0)$$

$$\mathbf{b}^{(+1)}(1,0,\dots,0) = \frac{1}{d+1} \mathbf{b}(0,\dots,0) + \frac{d}{d+1} \mathbf{b}(1,0,\dots,0)$$

$$\mathbf{b}^{(+1)}(1,1,0,\dots,0) = \frac{2}{d+1} \mathbf{b}(1,0,\dots,0) + \frac{d-1}{d+1} \mathbf{b}(1,1,0,\dots,0)$$

$$\mathbf{b}^{(+1)}(1,1,1,\dots,1,0) = \frac{d}{d+1} \mathbf{b}(1,1,1,\dots,1) + \frac{1}{d+1} \mathbf{b}(1,\dots,1,0)$$

$$\mathbf{b}^{(+1)}(1,\dots,1) = \mathbf{b}(1,\dots,1)$$

Degree Elevation

Result: new control points

$$\mathbf{p}_i^{(+1)} = \frac{i}{n+1} \mathbf{p}_{i-1} + \left(1 - \frac{i}{n+1}\right) \mathbf{p}_i, \quad i = 0, \dots, n+1 \quad (\text{zero points if out of range})$$

Repeated degree elevation:

$$\mathbf{p}_i^{(+k)} = \sum_{j=1}^d \mathbf{p}_j \binom{d}{j} \frac{\binom{k}{i-j}}{\binom{d+k}{j}} \quad (\text{proof by induction})$$

Repeating degree elevation lets the control point converge to the Bezier curve in the limit (proof using Stirling's formula).

Variation Diminishing Property

Settings:

- Given an original curve $C^{(original)} \subseteq \mathbb{R}^3$
- Given a second curve $\mathbb{R}^3 \supseteq C^{(derived)} = f(C^{(original)})$ that is derived from the original by some mapping (algorithm) f .
- If for any arbitrary plane P , $C^{(derived)}$ does not have more intersections with P than $C^{(original)}$, the mapping f is called *variation diminishing*.
- Formally: \forall planes $P: \#(C^{(derived)} \cap P) \leq \#(C^{(original)} \cap P)$

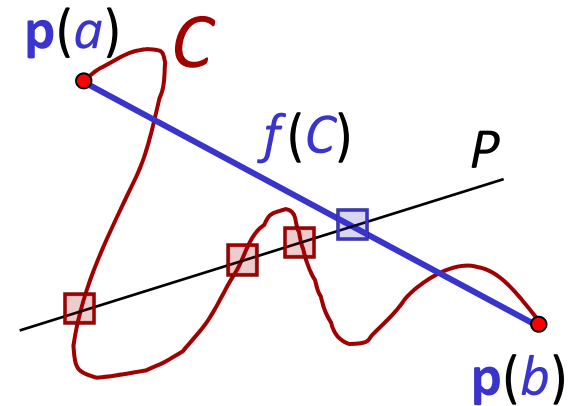
Mappings of interest:

- Mapping from a control polygon to the curve segment

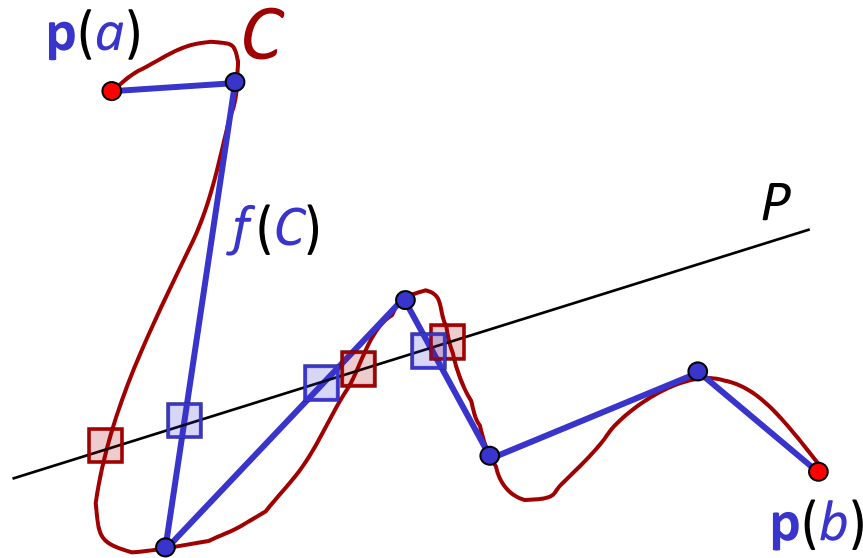
Easy example

Very simple mapping:

- Given a continuous curve $C = \mathbf{p}([a, b])$
- The mapping f creates a linear approximation: the line through $\mathbf{p}(a)$ and $\mathbf{p}(b)$.
- This mapping is variation diminishing
 - The line segment intersects the plane at most once
 - If this is the case, the original curve must have at least one intersection, too, because of continuity



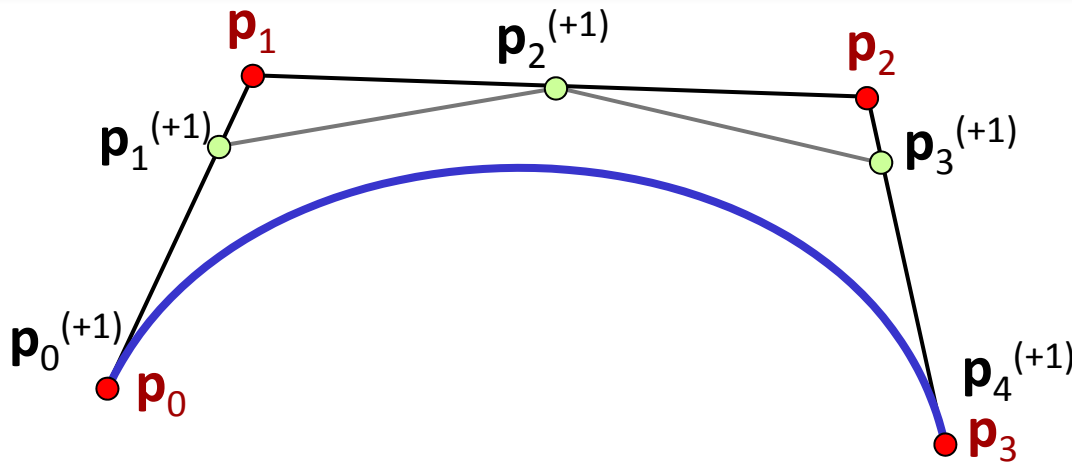
Another Example



Piecewise linear approximation:

- Obviously, a piecewise linear approximation of a curve (approximating with a polygon) is variation diminishing, too.

Bezier Curves



Degree Elevation:

$$\mathbf{p}_i^{(+1)} = \frac{i}{n+1} \mathbf{p}_{i-1} + \left(1 - \frac{i}{n+1}\right) \mathbf{p}_i$$

Bezier curves are variation diminishing:

- The Bezier curve is the limit of the control point sequence for infinite degree elevation.
- Every degree elevation step produces a piecewise linear approximation of the original control polygon, which is variation diminishing.

Bezier Curves

Consequence:

- A Bezier curve does not intersect any plane more frequently than its control polygon.
- Therefore, it cannot have too weird oscillations.
- However
 - The convergence of degree elevation is very slow
 - Not useful as evaluation technique in practice
 - One can approximate arbitrary smooth functions in a convergent way using the Bernstein basis (Weierstraß approximation theorem). But:
 - The convergence rate is not satisfactory in practice.
 - We do not have local control.

Polynomial Splines Revisited:

B-Splines

B-Spline Curves

(General) B-Spline Curves:

- Given:
 - A degree d (constant for the whole curve)
 - A knot sequence (t_0, t_1, \dots, t_n) with $t_0 \leq t_1 \leq \dots \leq t_n$
 - Control points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m$ ($m = n - d + 1$)
- With this information, we want to construct the spline curve within $t = [t_d, \dots, t_{n-d}]$
- Each polynomial segment $\mathbf{f}_i(t)$ is defined by the $d + 1$ control points $\mathbf{p}_i, \dots, \mathbf{p}_{i+d}$.
- Within each such segment / control point set, we can apply linear interpolation based on blossoms to compute points on the curve.

De Boor Algorithm

Blossoming version of the de Boor algorithm:

- The de Boor algorithm will appear as a generalization of the de Casteljau algorithm
 - similar structure
 - de Casteljau as a special case (special knot sequence)

De Boor Algorithm

Blossoming definition of B-Splines: very simple

- The control point sequence is given as:

$$\mathbf{p}_0 = \mathbf{p}(t_0, \dots, t_{d-1})$$

...

$$\mathbf{p}_i = \mathbf{p}(t_i, \dots, t_{i+d-1})$$

...

$$\mathbf{p}_m = \mathbf{p}(t_{n-d+1}, \dots, t_n)$$

- This means: we just use consecutive knot values as blossom arguments
- Then, we proceed as before
- Same computational scheme as de Casteljau, but different weights (because of the different knot values)

De Boor Algorithm

For simplicity, we will first look at a single B-spline segment:

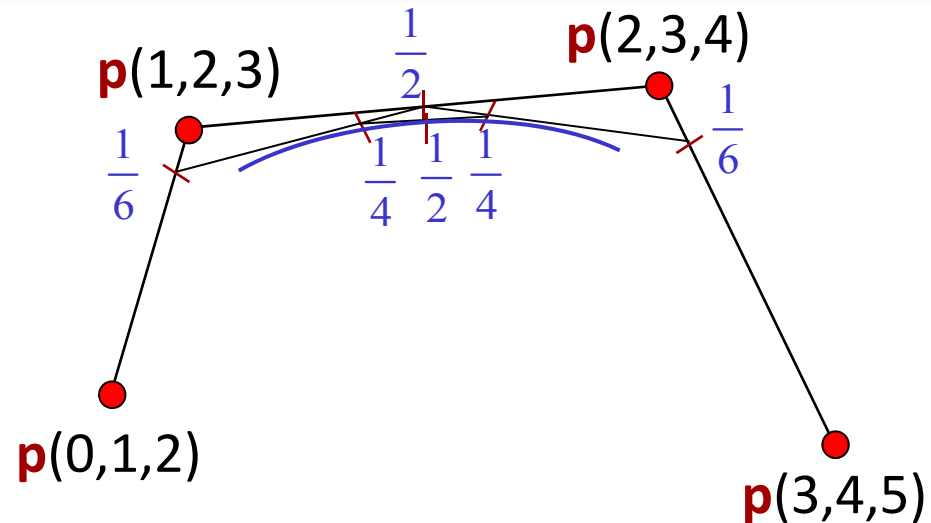
- Knots $(t_0, t_1, \dots, t_{2d-1})$
- Control points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{d-1}$
- B-spline segment within $[t_d \dots t_{d+1}]$

Interpolation rule:

- Given blossoms $f(\mathbf{a}, t_1, \dots, t_{d-1}), f(\mathbf{b}, t_1, \dots, t_{d-1})$
- We obtain $f(\mathbf{t}, t_1, \dots, t_{d-1})$ as:

$$f(t, t_1, \dots, t_{d-1}) = \frac{b-t}{b-a} f(a, t_1, \dots, t_{d-1}) + \left[1 - \frac{b-t}{b-a} \right] f(b, t_1, \dots, t_{d-1})$$

De Boor Algorithm



$$\frac{3-t}{3} p(0,1,2) + \left[1 - \frac{3-t}{3}\right] p(3,1,2)$$

$$\frac{3-t}{2} p(t,1,2) + \left[1 - \frac{3-t}{2}\right] p(t,3,2)$$

$$\frac{3-t}{1} p(t,t,2) + \left[1 - \frac{3-t}{1}\right] p(t,t,3)$$

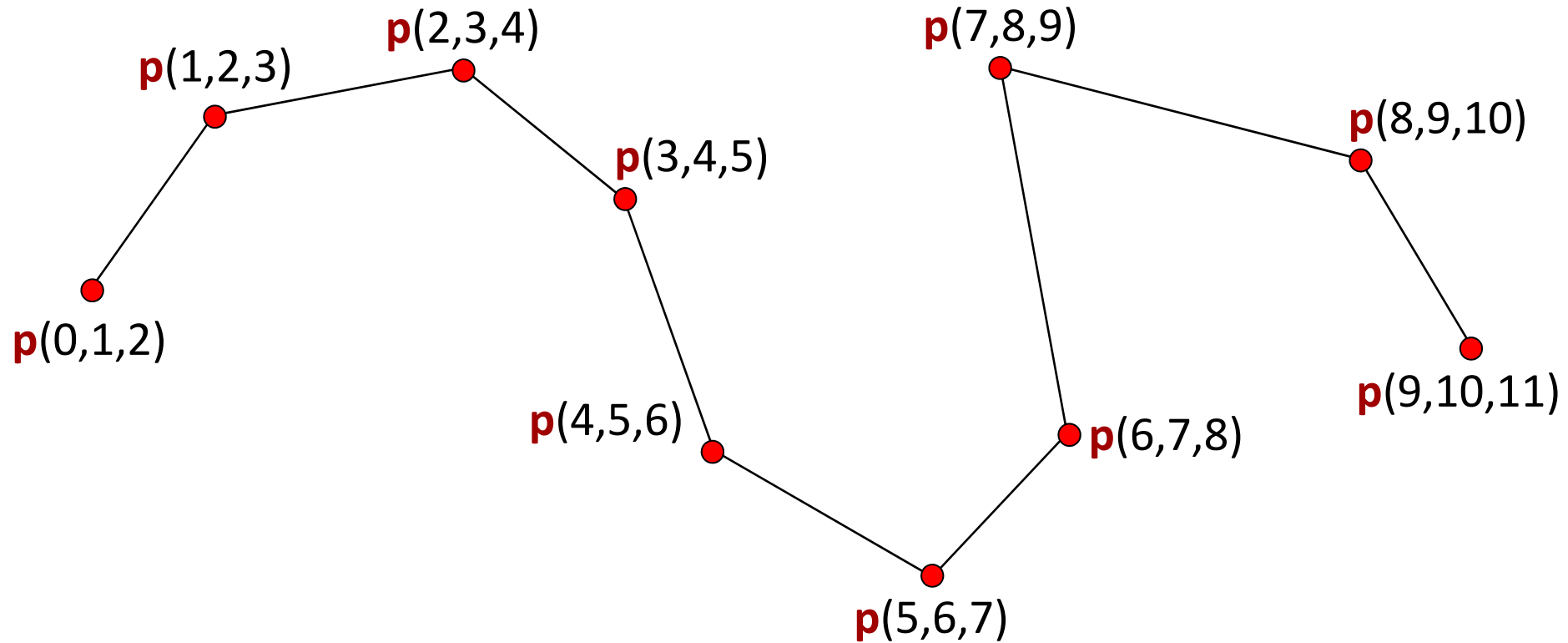
$p(t,t,t)$

$$\frac{4-t}{3} p(1,2,3) + \left[1 - \frac{4-t}{3}\right] p(4,2,3)$$

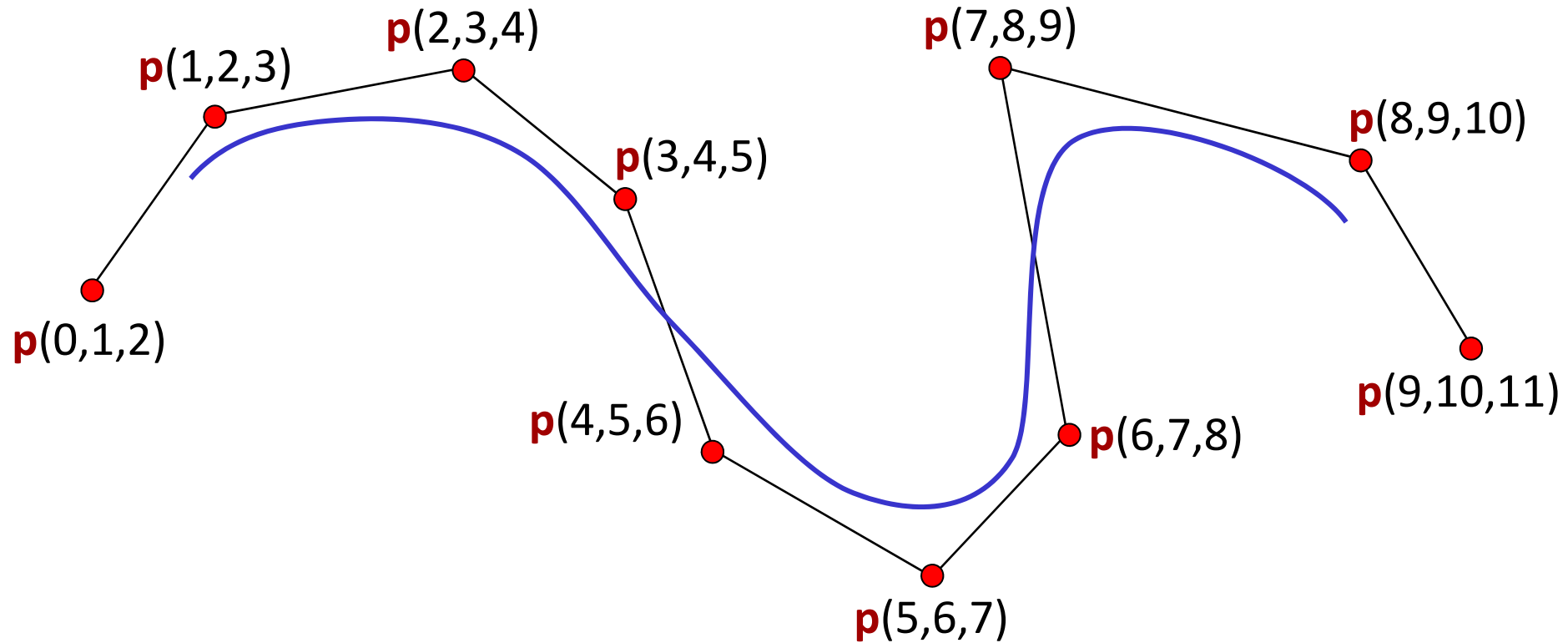
$$\frac{4-t}{2} p(t,3,2) + \left[1 - \frac{4-t}{2}\right] p(t,3,4)$$

$$\frac{5-t}{3} p(2,3,4) + \left[1 - \frac{5-t}{3}\right] p(5,3,4)$$

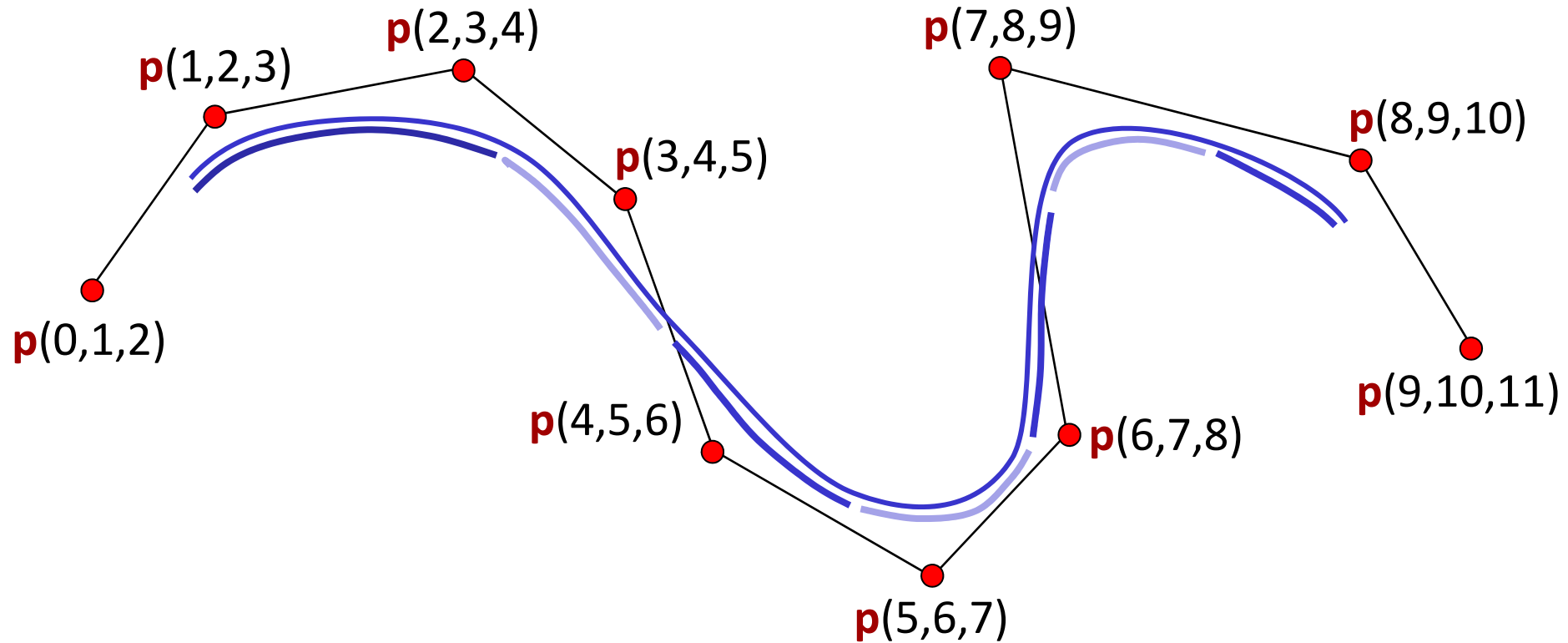
Example: General Case



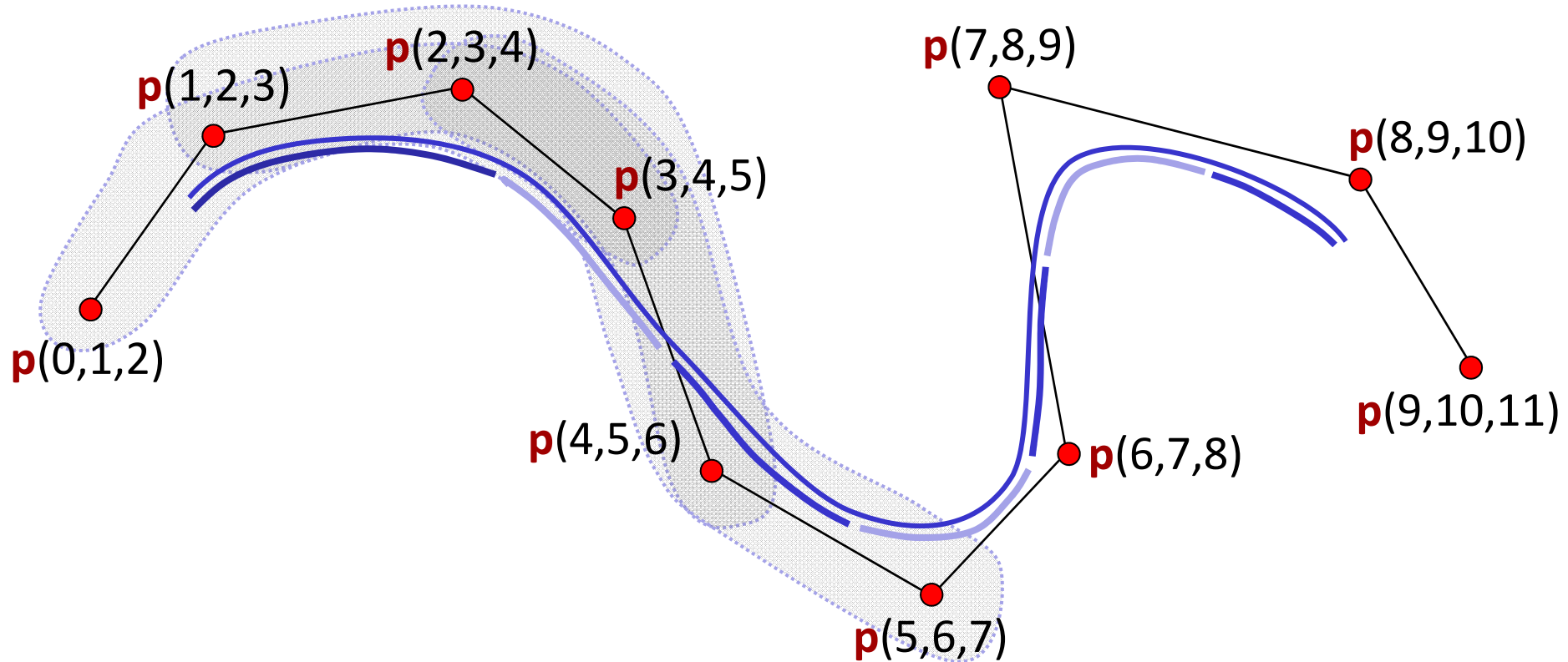
Example: General Case



Example: General Case



Example: General Case



De Boor Algorithm

De Boor Algorithm:

- In order to evaluate $f(t)$ for a $t \in [t_{i-1}, \dots, t_i)$
- Compute:

$$\mathbf{p}_i^{(0)} = \mathbf{p}_i$$

For increasing j :

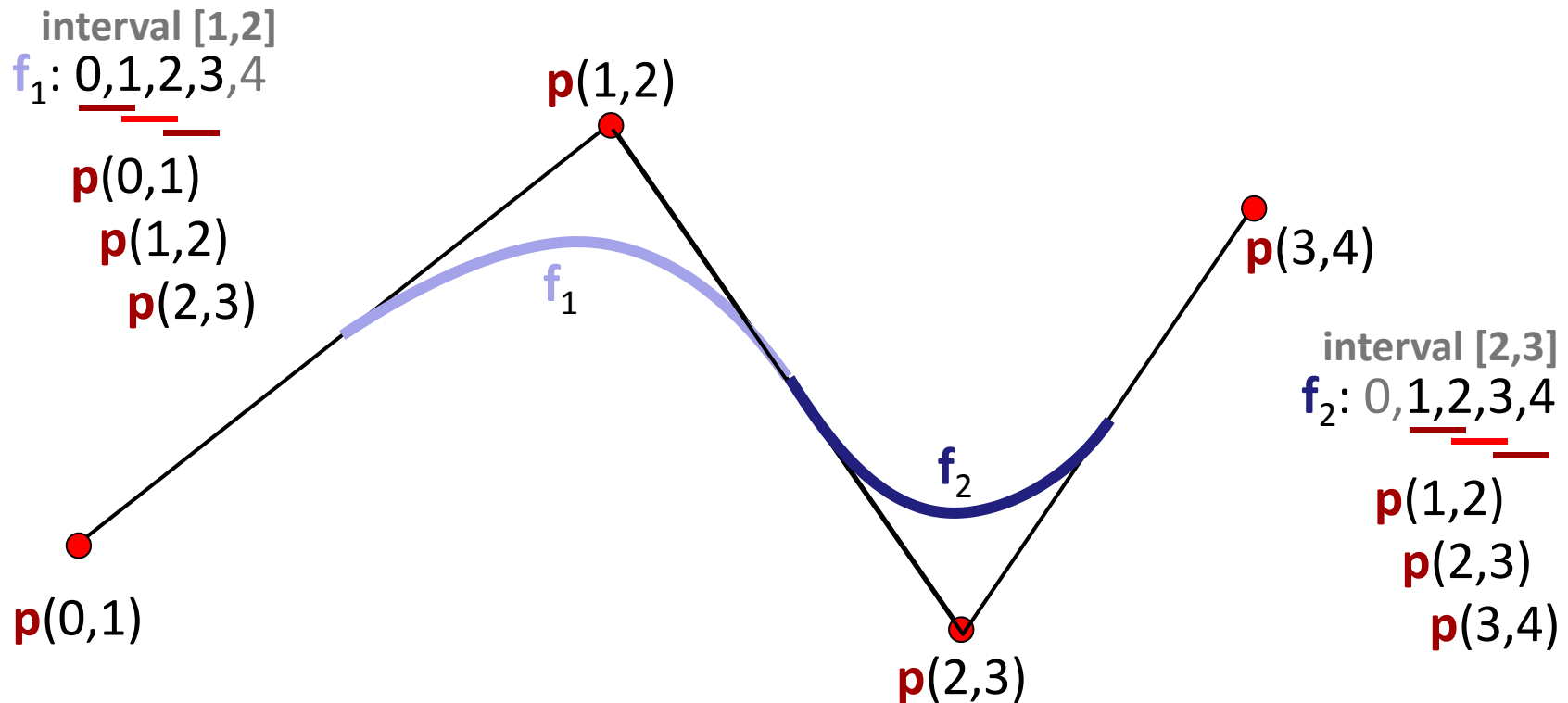
$$\mathbf{p}_i^j(t) = \alpha_i^{(j)} \mathbf{p}_i^{j-1}(t) + (1 - \alpha_i^{(j)}) \mathbf{p}_{i-1}^{j-1}(t), \quad \alpha_i^{(j)} = \frac{(t - t_{i-1})}{t_{i+d-j-1} - t_{i-1}}$$

Output $\mathbf{p}_i^{d-1}(t)$

Smoothness

Multiplied Control Points:

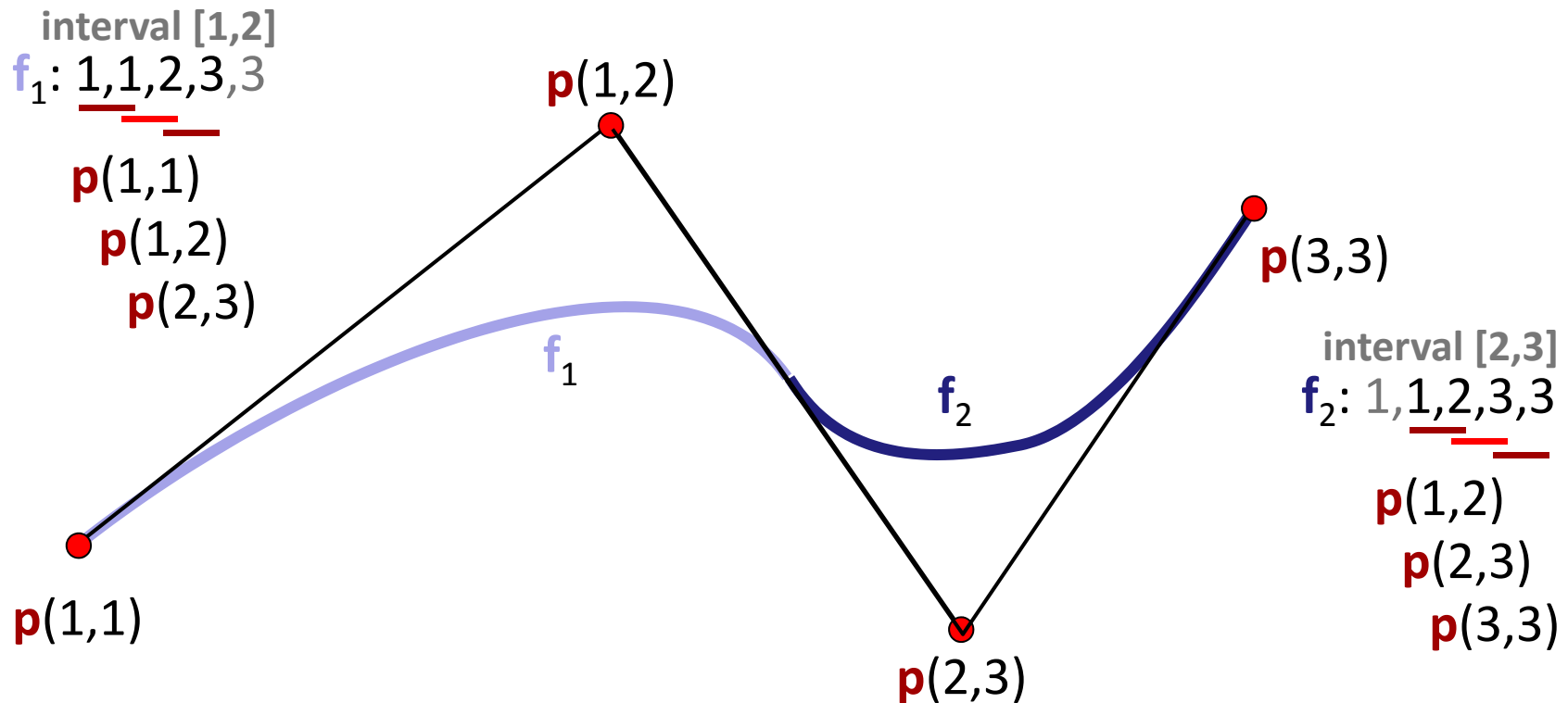
- Quadratic B-Spline Curve (two segments)



Smoothness

Multiplied Control Points:

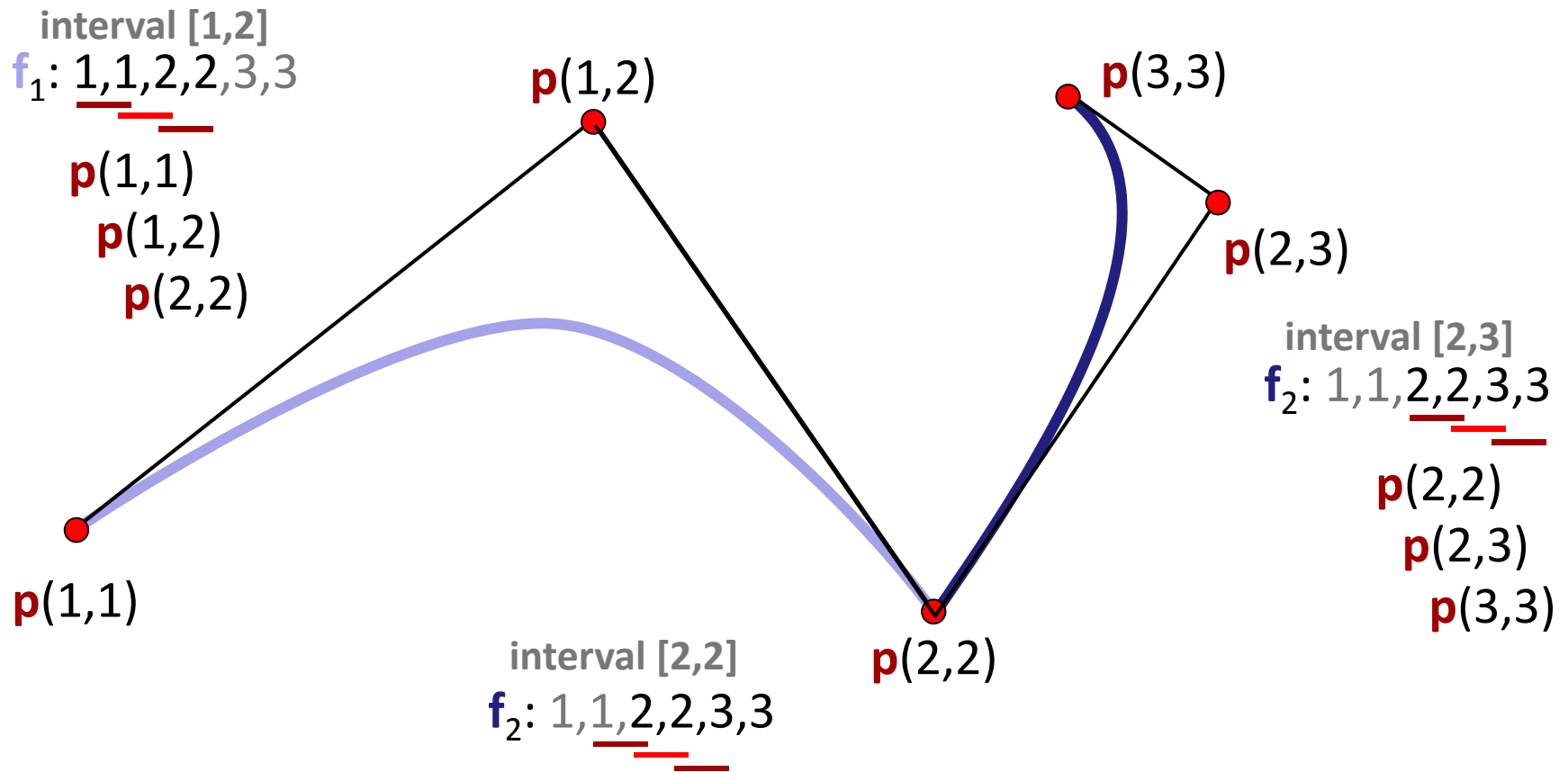
- Double end points \rightarrow interpolated (c.f. de Casteljau)



Smoothness

Multiplied Control Points:

- Double inner node: reduces continuity by one.



Smoothness

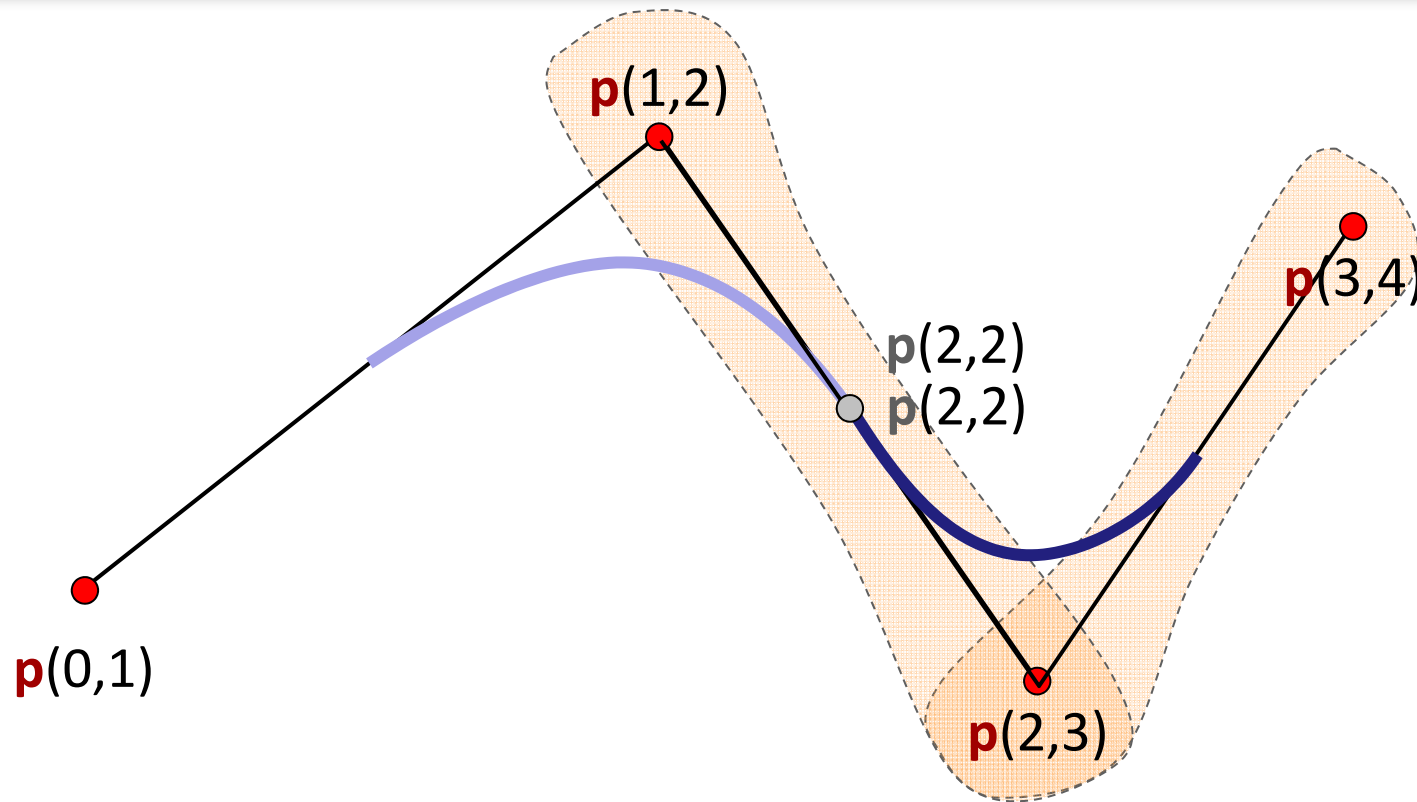
Multiplied Control Points: Blossoming

- Assume a control point contains one knot value v with multiplicity k .
- Through repeated linear interpolation, we can transform the blossoms for the left and right spline segment to:

$$\mathbf{p}(\underbrace{v, \dots, v}_k, v_1, \dots, v_{d-k}) \text{ and } \mathbf{p}(\underbrace{v, \dots, v}_k, v_1, \dots, v_{d-k}).$$

- This means, we have at least C^{d-k} continuity.

Example



$$f_1(0,1) = p(0,1)$$

$$f_1(1,2) = p(1,2)$$

$$f_1(2,3) = p(2,3)$$

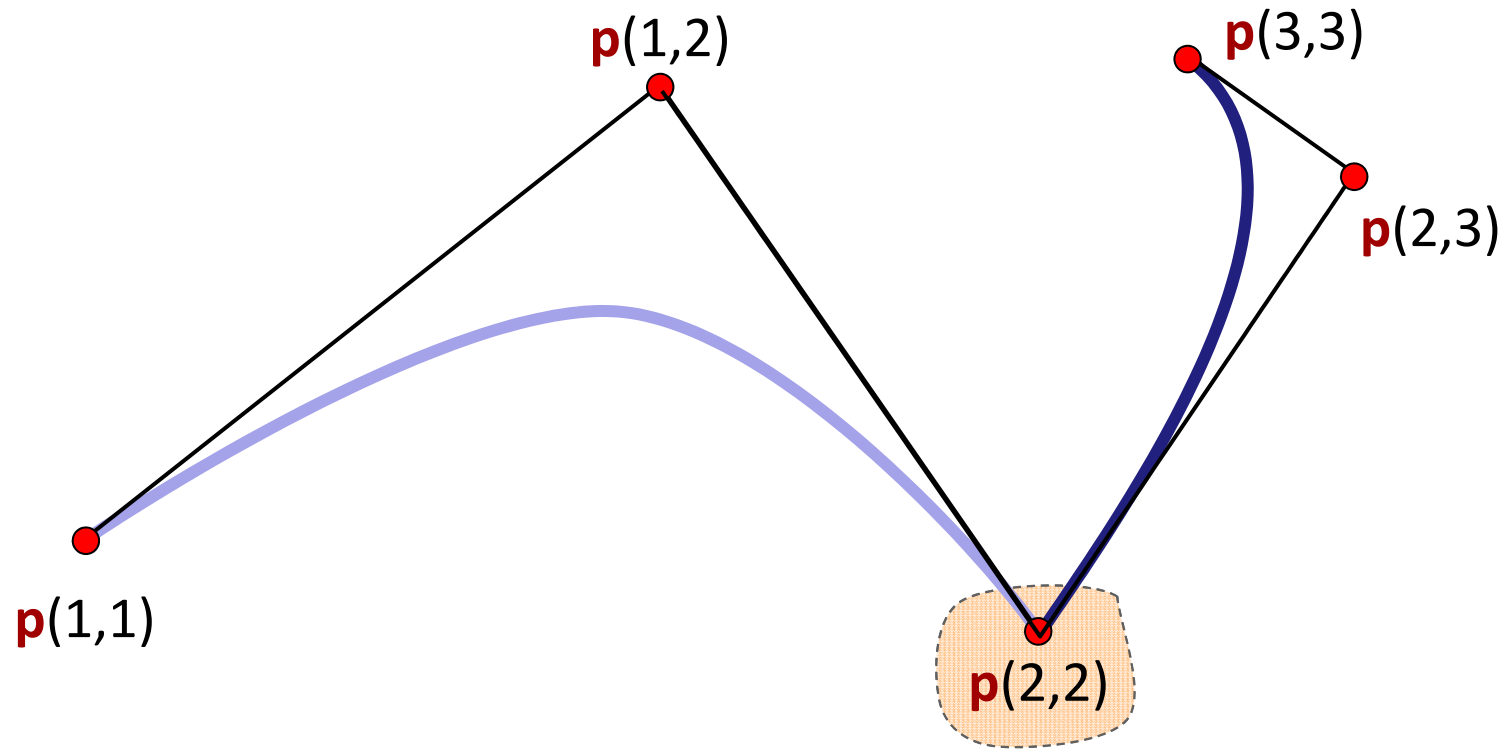
$$f_2(1,2) = p(1,2)$$

$$f_2(2,3) = p(2,3)$$

$$f_2(3,4) = p(3,4)$$

$$\Rightarrow f_1(t,2) = f_2(t,2)$$

Smoothness



$$f_1(1,1) = p(1,1)$$

$$f_1(1,2) = p(1,2)$$

$$f_1(2,2) = p(2,2)$$

$$f_2(2,2) = p(2,2)$$

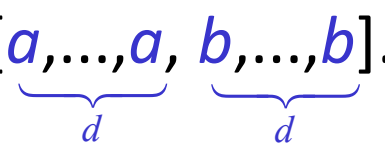
$$f_2(2,3) = p(2,3)$$

$$f_2(3,3) = p(3,3)$$

$$\Rightarrow f_1(2,2) = f_2(2,2)$$

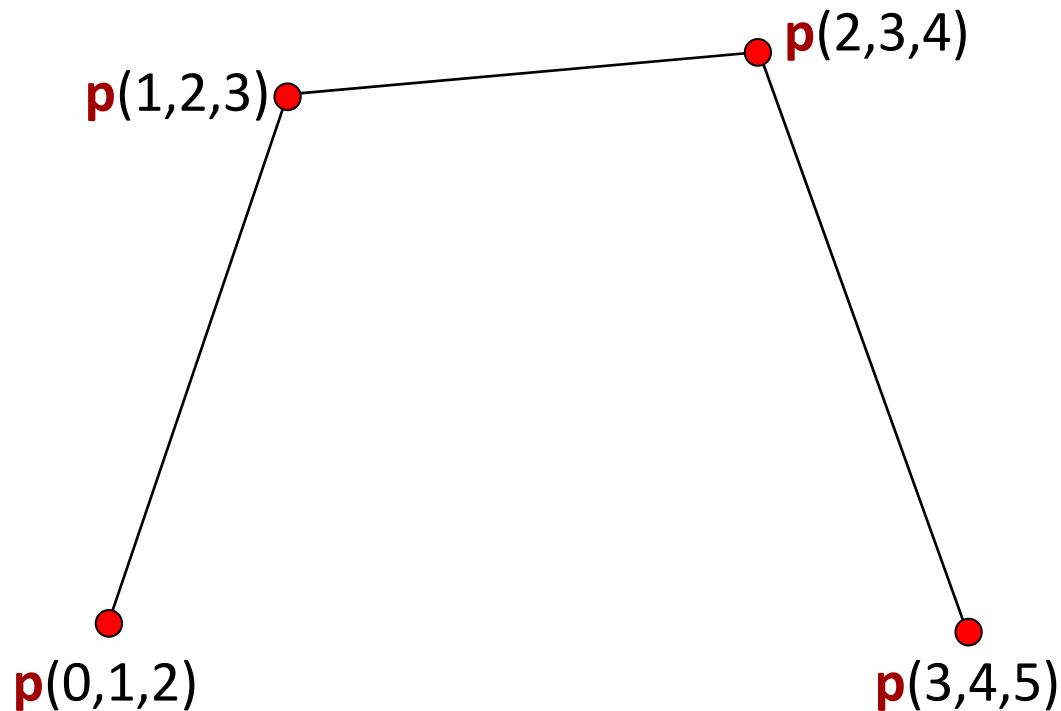
Connection to Bezier Splines

Special case:

- We can model Bezier splines through non-uniform B-Splines using a knot sequence $[a, \dots, a, b, \dots, b]$.

- For this knot sequence, the de Boor algorithm yields exactly the de Casteljau Algorithm.
- Therefore: the de Boor algorithm is a generalization of the de Casteljau algorithm.
- We can also insert Bezier segments somewhere into a B-spline.
- Interpolating end conditions are “half a Bezier segment”.

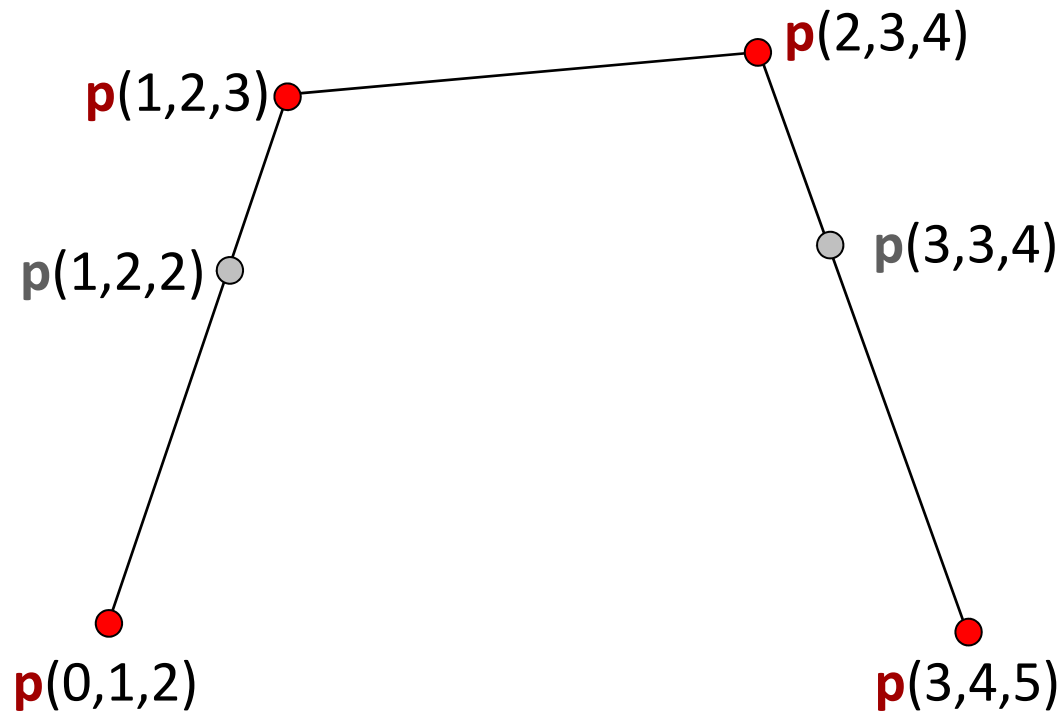
Conversion to Bezier Splines

Converting a B-spline segment to a Bezier segment:



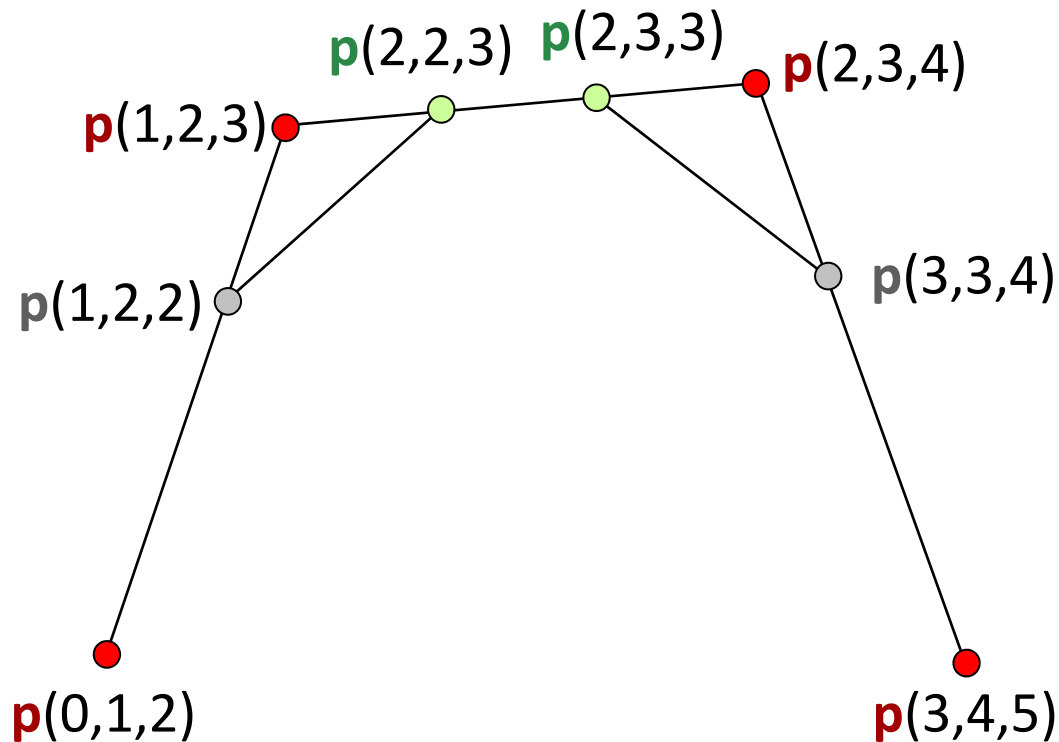
Conversion to Bezier Splines

Converting a B-spline segment to a Bezier segment:



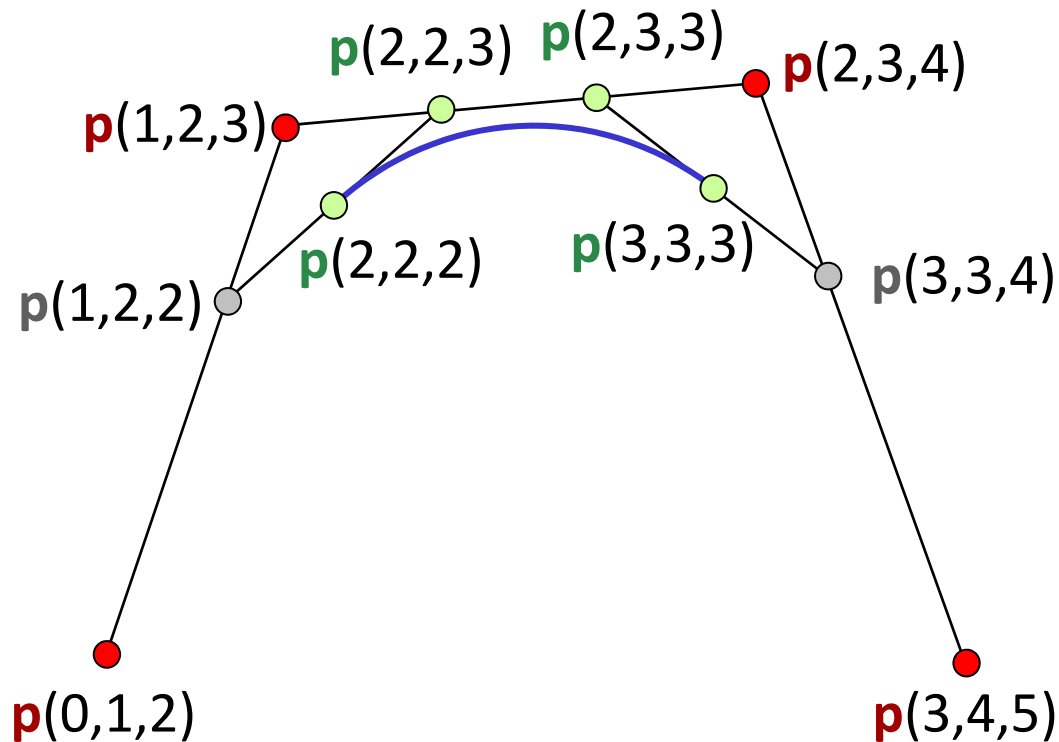
Conversion to Bezier Splines

Converting a B-spline segment to a Bezier segment:



Conversion to Bezier Splines

Converting a B-spline segment to a Bezier segment:



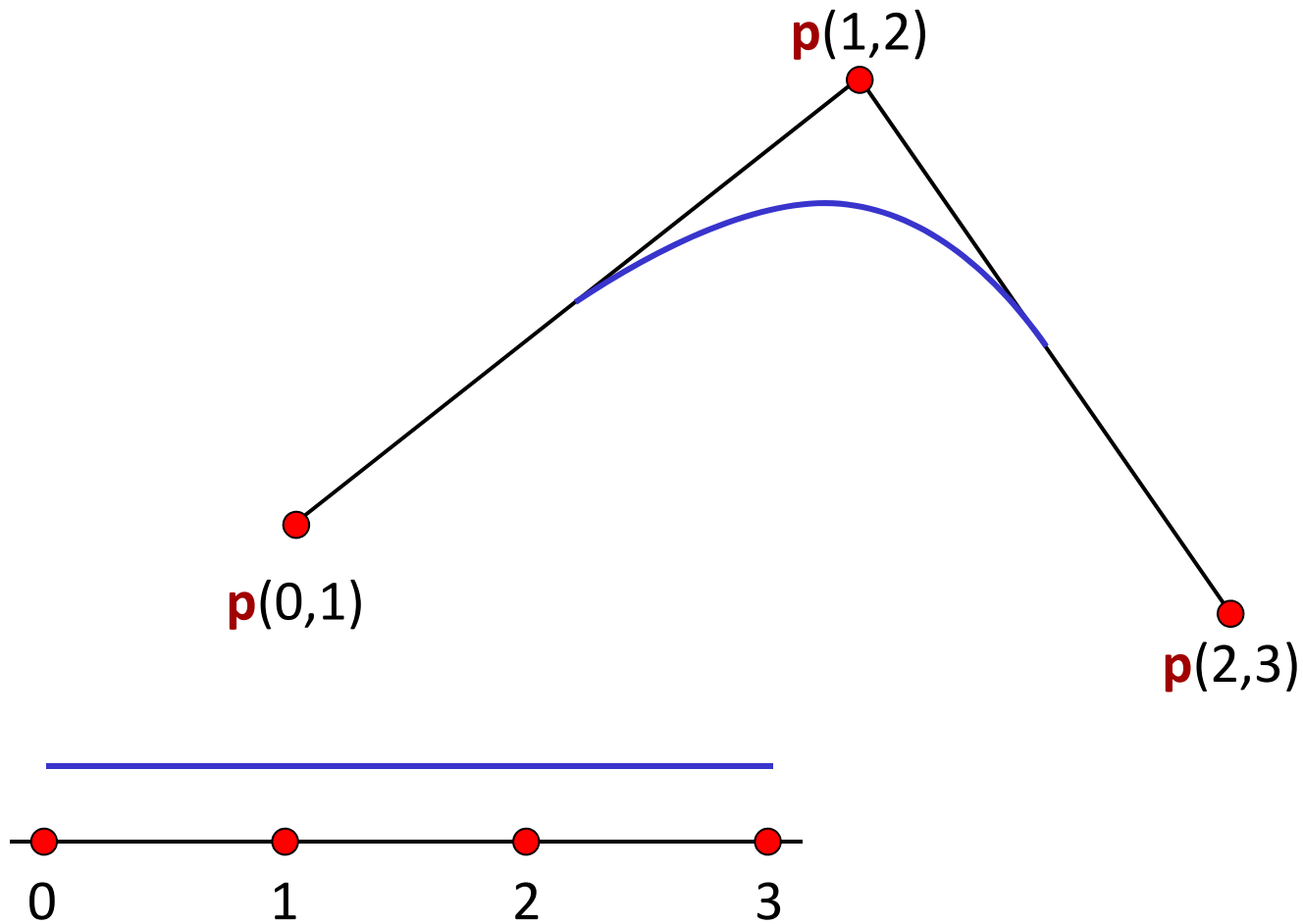
Knot Insertion

Problem:

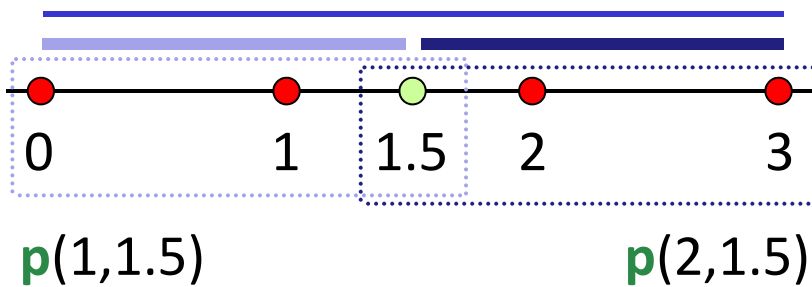
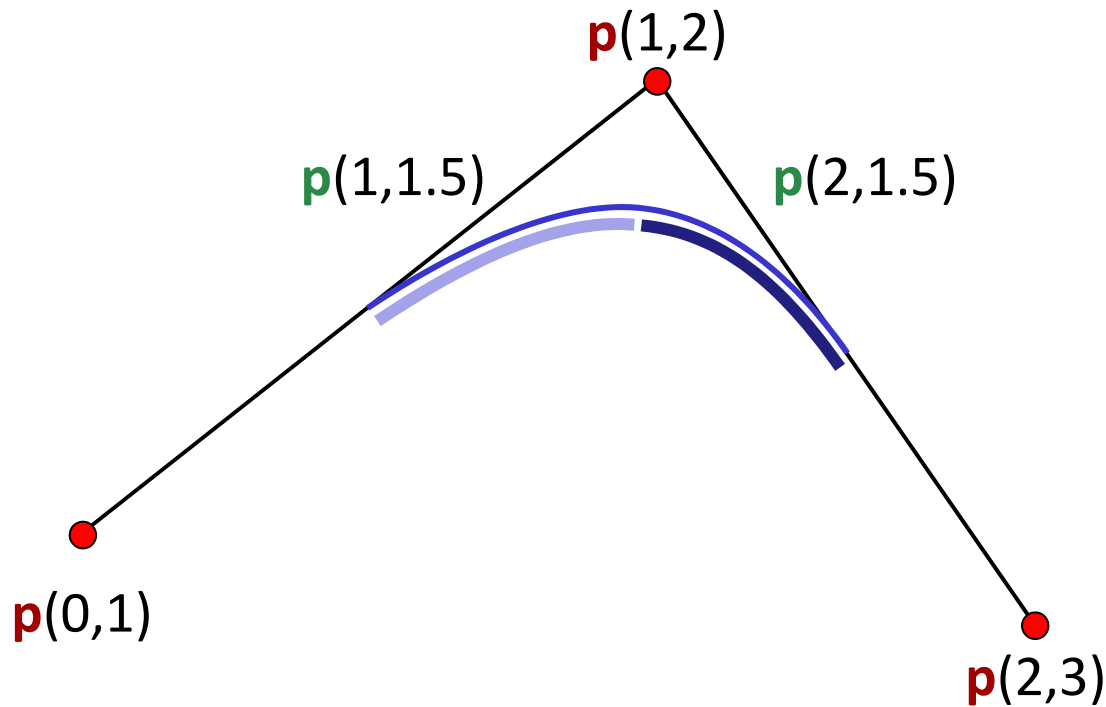
- Given a B-spline curve
- Insert a new knot value & a new control point in between two existing ones
- Without changing the curve

Very simple solution using blossoms

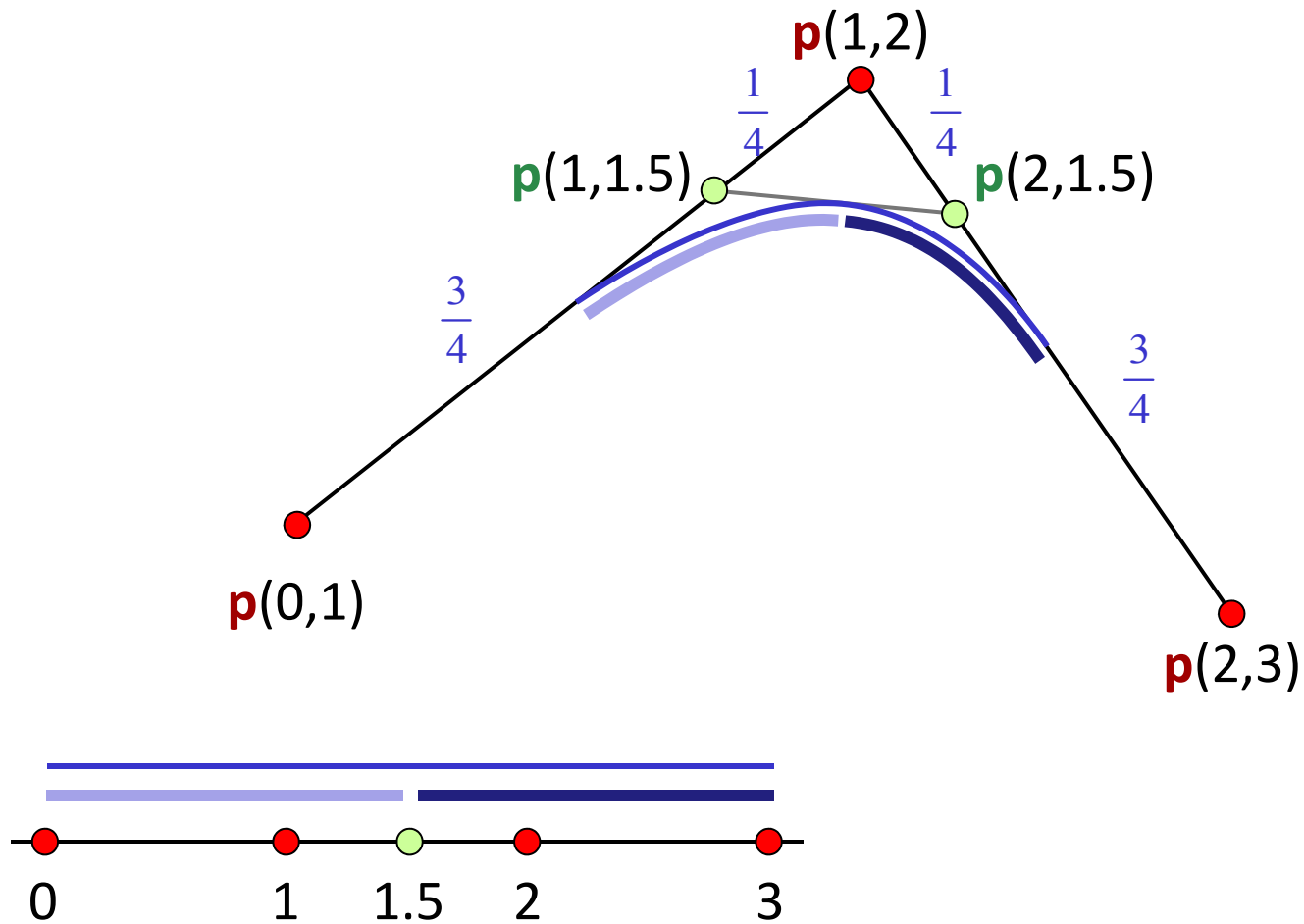
Quadratic Example



Quadratic Example



Quadratic Example



General Algorithm

The general case:

- Interval with knot to be added: (t_i, \dots, t_{i+2d-1})
- Execute one step of the de Boor algorithm
- This creates new control points
 - Keep the outer points $p(t_i, \dots, t_{i+d-1})$, $p(t_{i+1}, \dots, t_{i+2d-1})$
 - Replace the inner points $p(t_{i+1}, \dots, t_{i+d})$, $p(t_i, \dots, t_{i+d-2})$ with the newly created points from the de Boor step
- This will...
 - Insert one knot value
 - And insert one additional control point
 - Change the existing control points, except the outermost