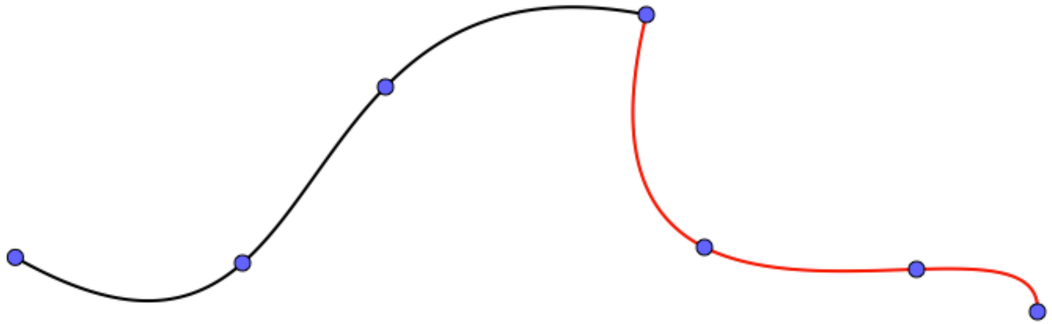
13 - Designing Approximating Curves

Acknowledgement: Daniele Panozzo, Olga Sorkine-Hornung, Alexander Sorkine-Hornung, Ilya Baran

Last time

- Interpolating curves
  - Monomials
  - Lagrange
  - Hermite
- Different control types
- Polynomials as a vector space
  - Basis transformation
- Connecting curves to splines



Disadvantages

- Monomials
  - unintuitive control, inefficient
- Lagrange
  - non-local control, oscillation for higher order polynomials, continuity for splines
- Hermite
  - requires setting tangent
- Catmull-Rom
  - unintuitive bending

Bezier Basis

Smooth Curves

- General parametric form
  - Weighted sum of coefficients and basis functions

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i F_i^n(t)$$

Coefficients

$\mathbf{c}_i \in \mathbb{R}^k$

Basis functions

$F_i^n(t) \in \Pi^n$

What might be “good” basis functions?

- Intuitive editing
  - Control points are coefficients
  - Predictable behavior
  - No oscillation
  - Local control

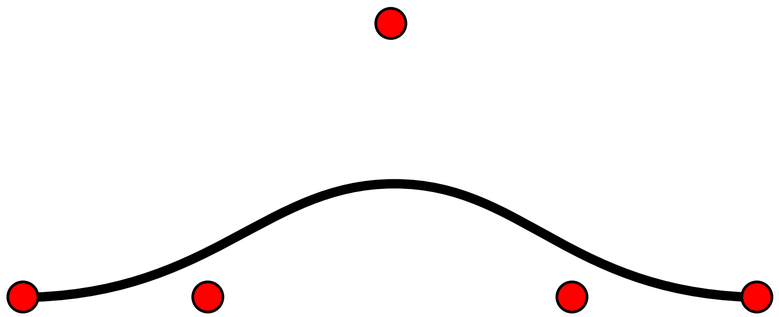
$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i F_i^n(t)$$

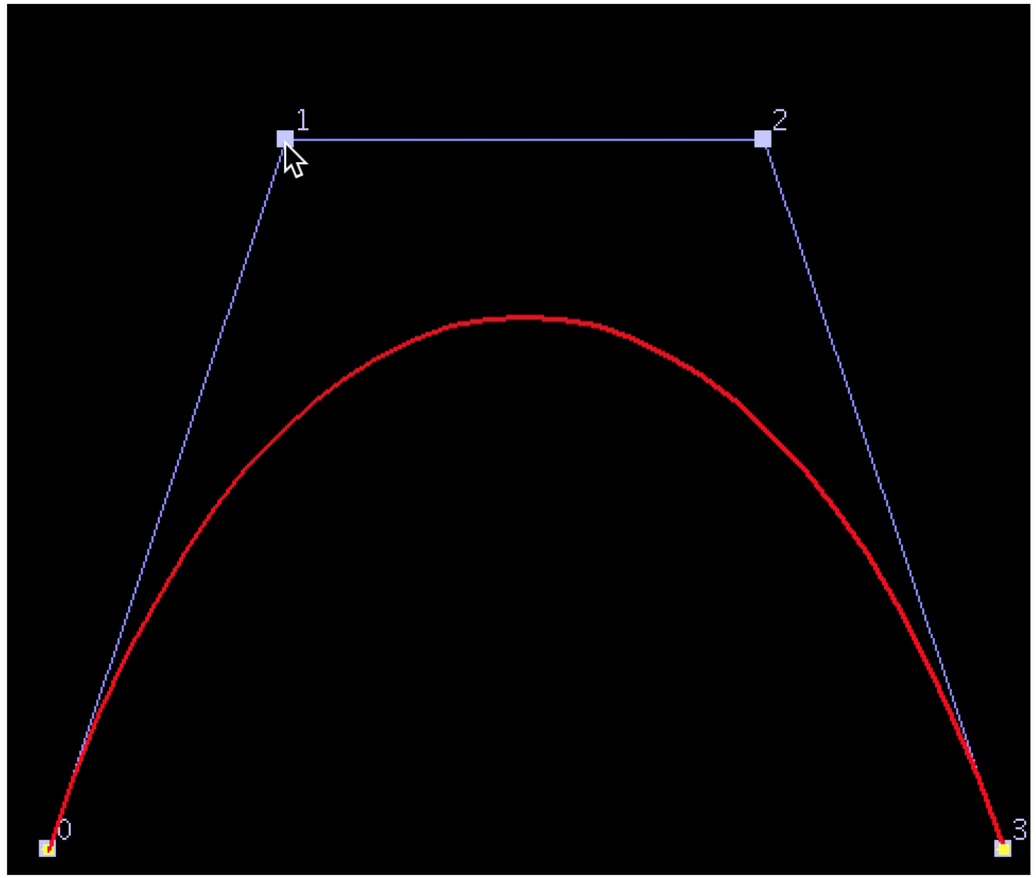
- Mathematical guarantees
  - Smoothness, affine invariance, linear precision, …

- Efficient processing and rendering

What might be “good” basis functions?

- Approximation instead of interpolation
- Bézier- and B-Spline curves





Bezier Curves

- Curve based on Bernstein polynomials

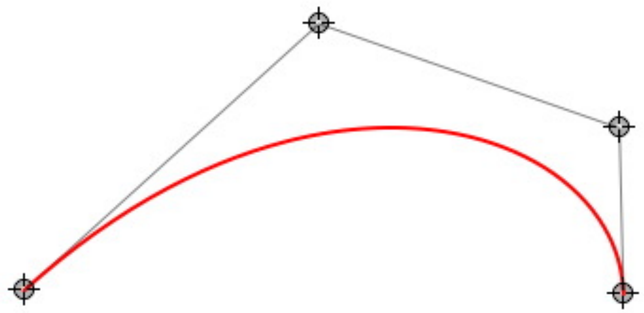
$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(t)$$

Control points

$\mathbf{c}_i \in \mathbb{R}^k$

Bernstein polynomials

$B_i^n(t) \in \Pi^n$



Control polygon

Bernstein Polynomials

- Bernstein polynomials $$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$
- Binomial coefficients $$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(t)$$

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

Properties

- Partition of Unity

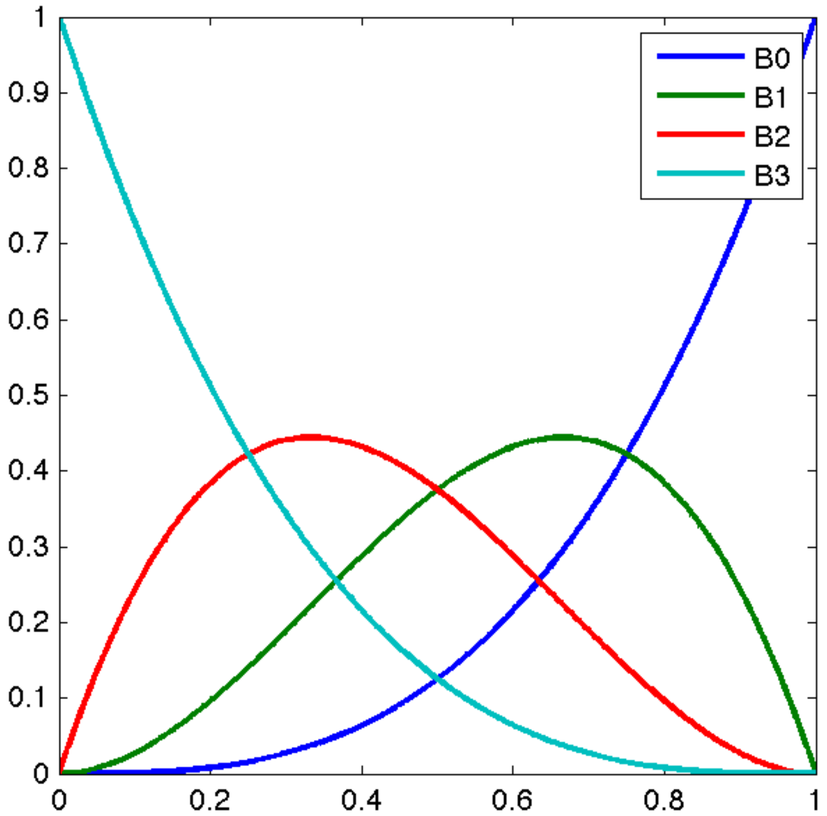
$$\sum_{i=0}^{n} B_i^n(t) = 1$$

- Non-negativity

$$B_i^n(t) \geq 0, t \in [0,1]$$

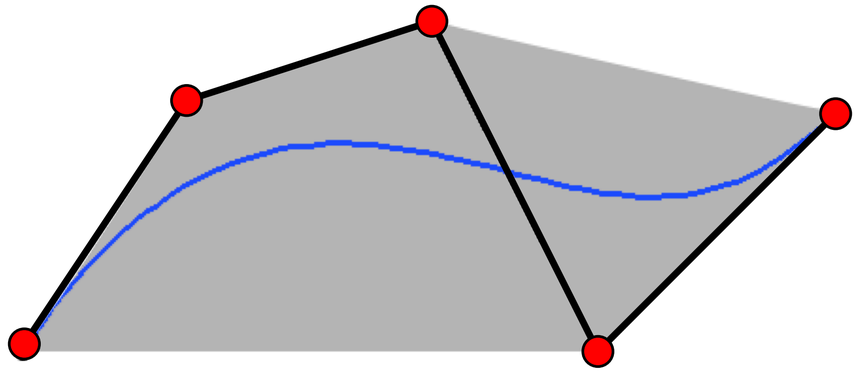
- Maximum $\max_{t \in [0,1]} B_i^n(t) : t = \frac{i}{n}$

- Symmetry $B_i^n(t) = B_{n-i}^n(1-t)$



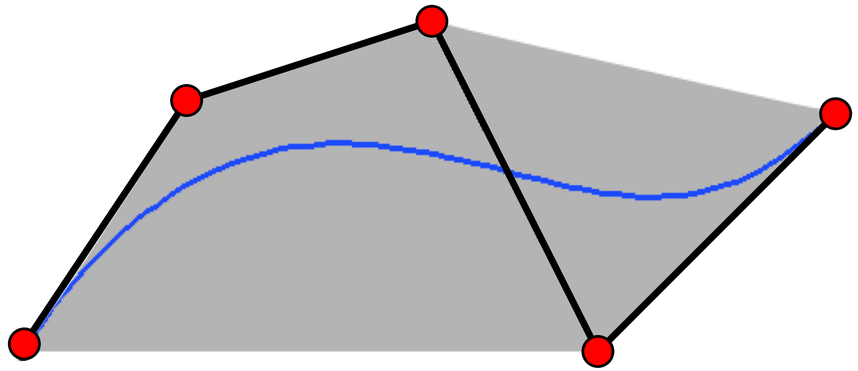
Properties of Bezier Curves

- Geometric interpretation of control points



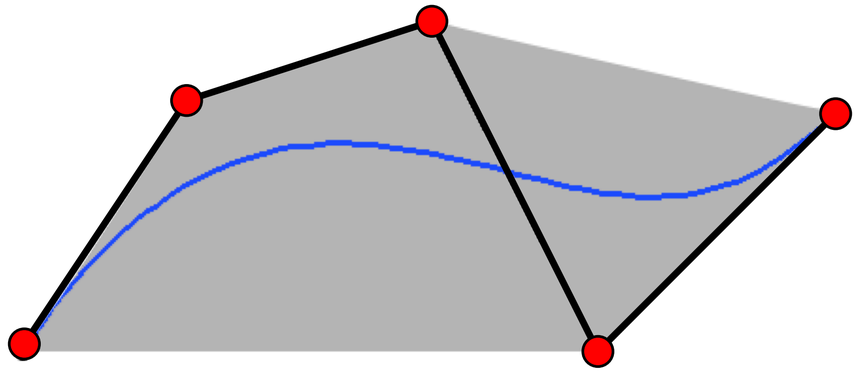
Properties of Bezier Curves

- Geometric interpretation of control points
- Convex hull
  - Polynomials positive
  - No oscillation



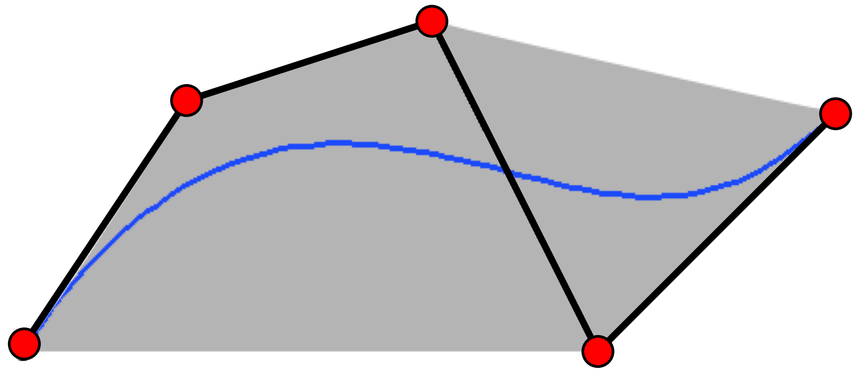
Properties of Bezier Curves

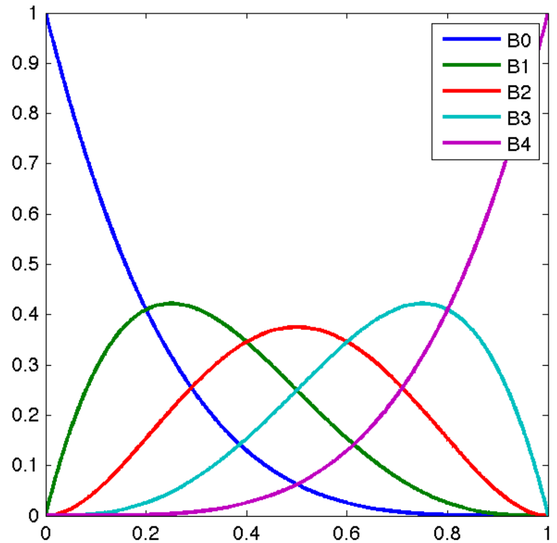
- Geometric interpretation of control points
- Convex hull
- Affine invariance
  - Barycentric combinations



Properties of Bezier Curves

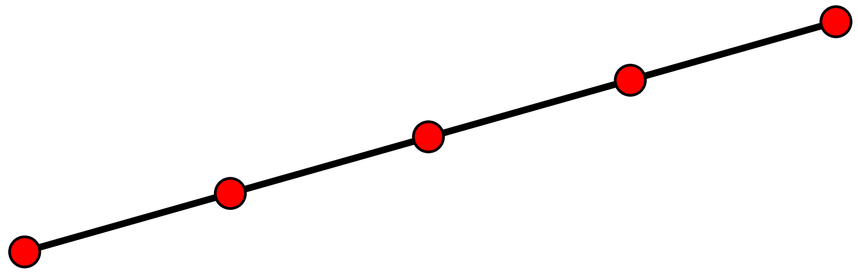
- Geometric interpretation of control points
- Convex hull
- Affine invariance
- Endpoint interpolation
- Symmetry





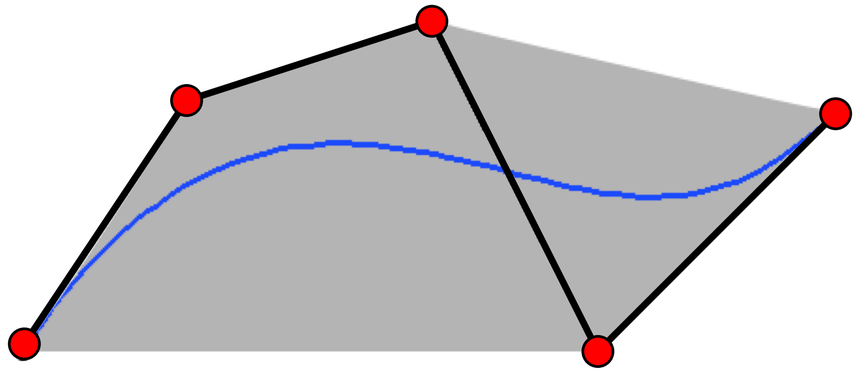
Properties of Bezier Curves

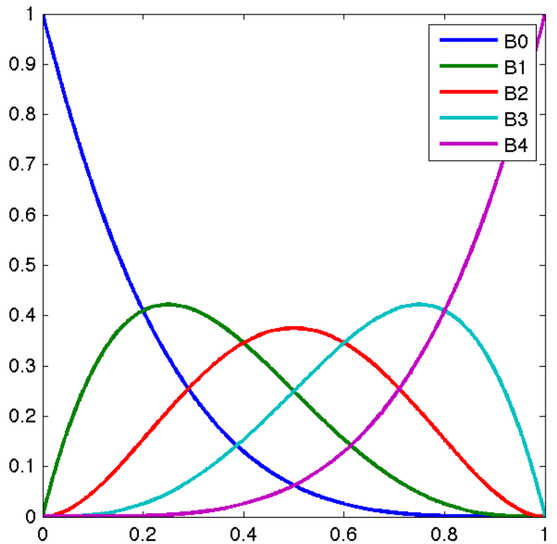
- Geometric interpretation of control points
- Convex hull
- Affine invariance
- Endpoint interpolation
- Symmetry
- Linear precision



Properties of Bezier Curves

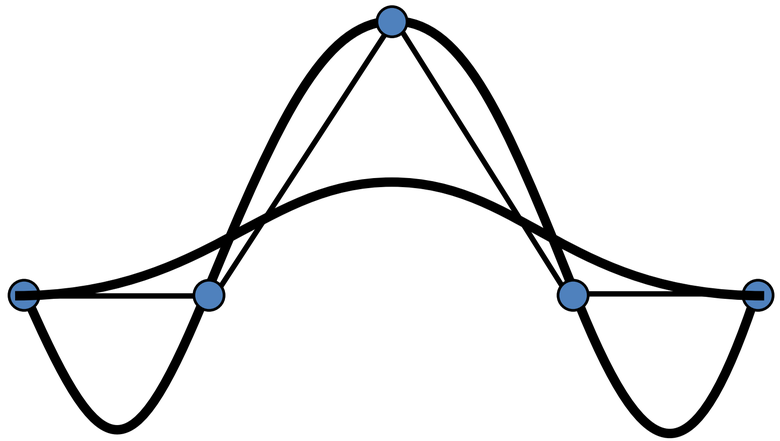
- Geometric interpretation of control points
- Convex hull
- Affine invariance
- Endpoint interpolation
- Symmetry
- Linear precision
- Quasi-local control





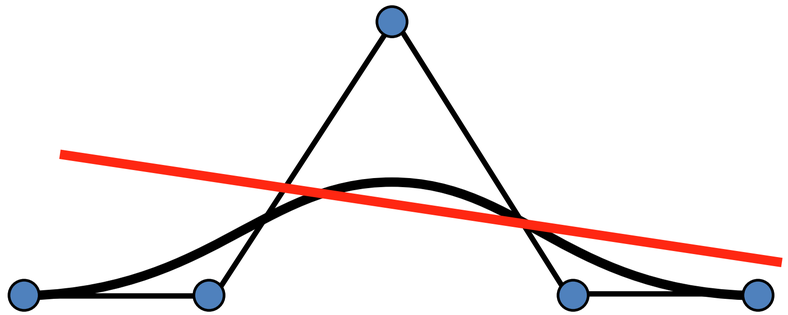
Variation Diminishing

- Curve “wiggles” no more than control polygon



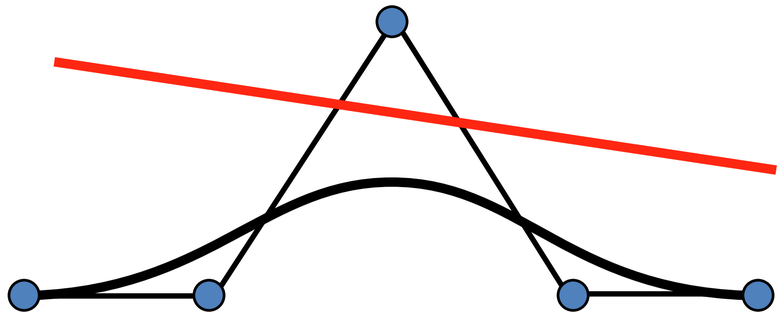
Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon >= intersection with curve



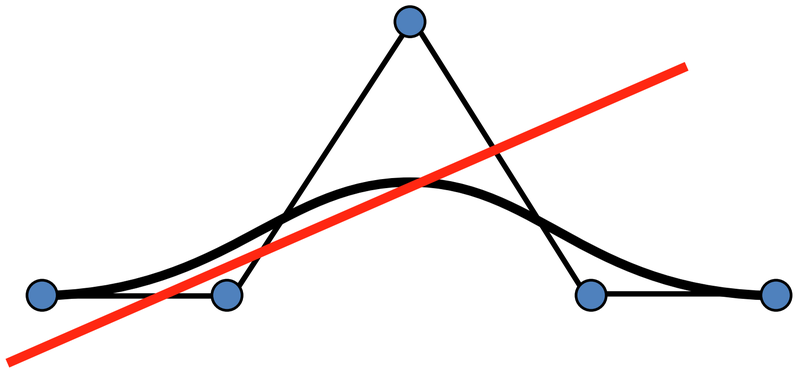
Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon >= intersection with curve



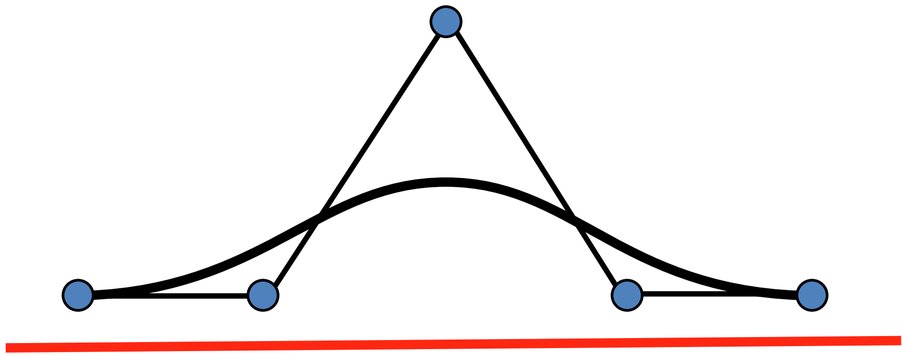
Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon >= intersection with curve



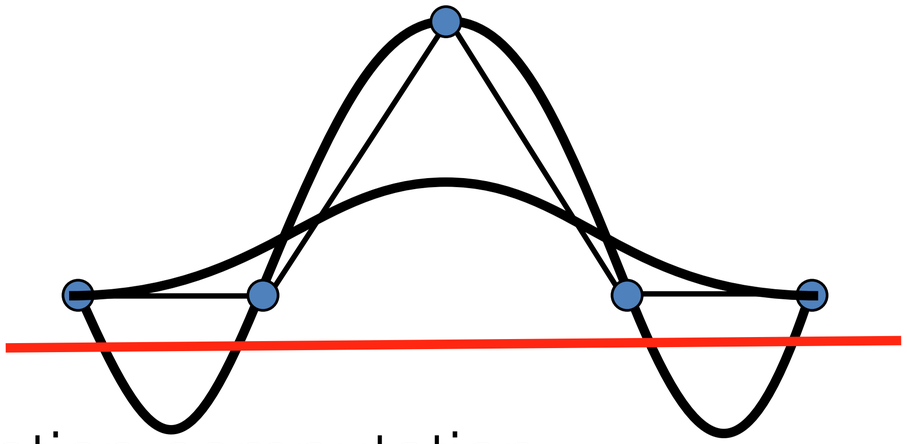
Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon >= intersection with curve



Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon >= intersection with curve

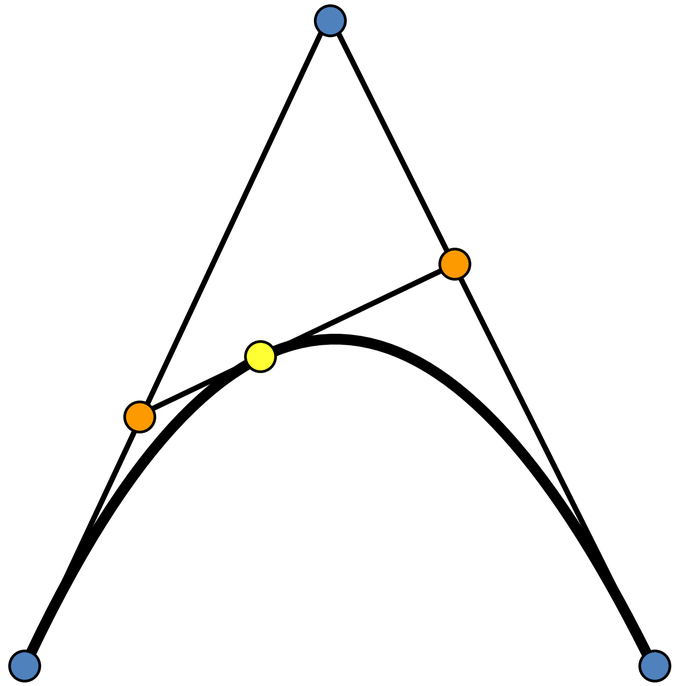


- Application: intersection computation

De Casteljau Algorithm

- Developed independently
- Successive linear interpolation
- Example: parabola

$$\begin{aligned}\mathbf{p}(t) = \quad & (1-t) \quad (\mathbf{c}_0(1-t) + \mathbf{c}_1 t) \\ + \quad & t \quad\quad (\mathbf{c}_1(1-t) + \mathbf{c}_2 t)\end{aligned}$$



- Recursive scheme behind Bernstein Polynomials

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

- Curves as series of linear interpolations

De Casteljau Algorithm

- Exploit recursive definition of Bernstein polynomials
  Repeated convex combination of control points

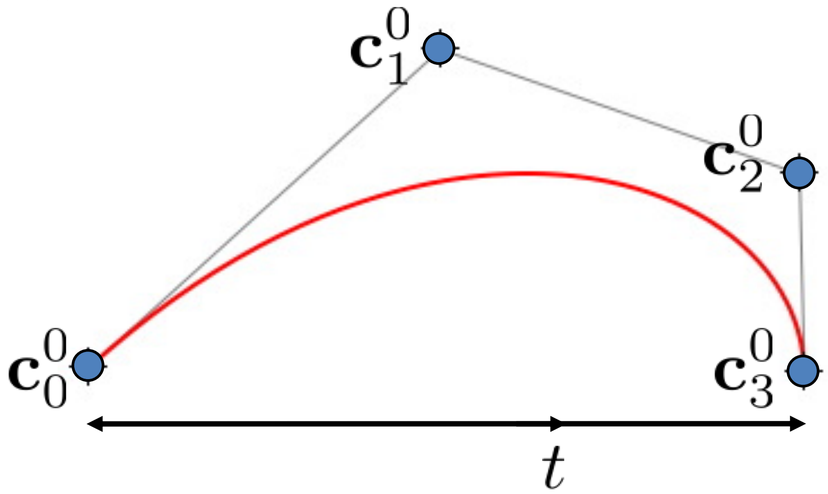
$$\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\mathbf{c}_{i+1}^{k-1} \qquad \mathbf{c}_i^0 := \mathbf{c}_i$$

$\mathbf{c}_0^0$

$\mathbf{c}_1^0$

$\mathbf{c}_2^0$

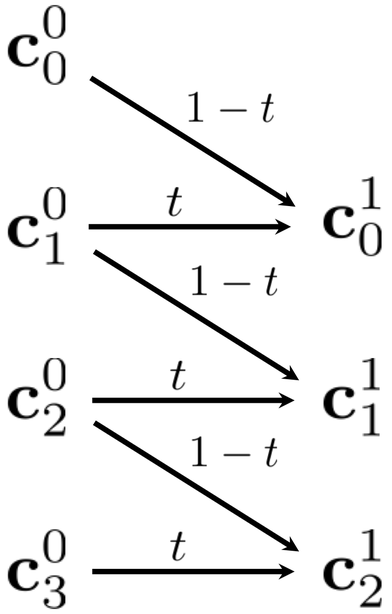
$\mathbf{c}_3^0$

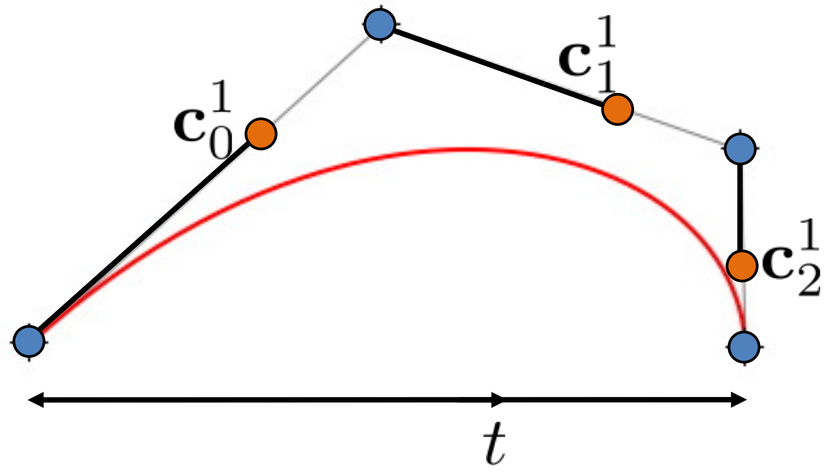


De Casteljau Algorithm

- Exploit recursive definition of Bernstein polynomials
  Repeated convex combination of control points

$$\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\mathbf{c}_{i+1}^{k-1} \qquad \mathbf{c}_i^0 := \mathbf{c}_i$$

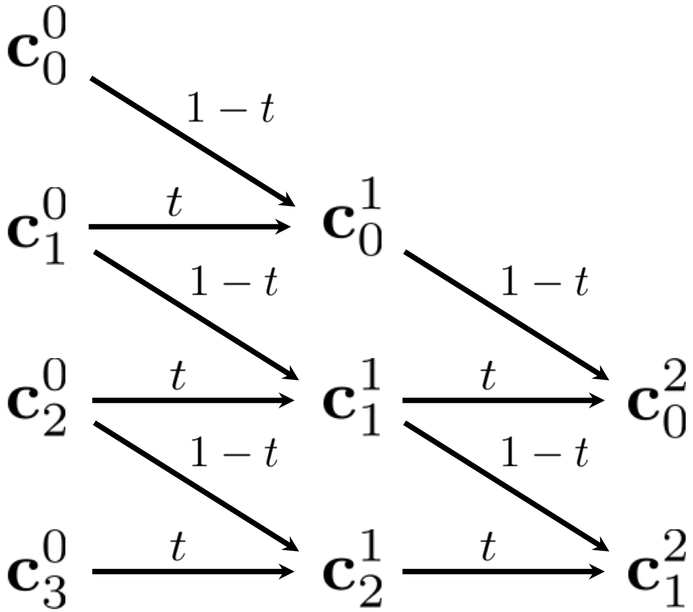


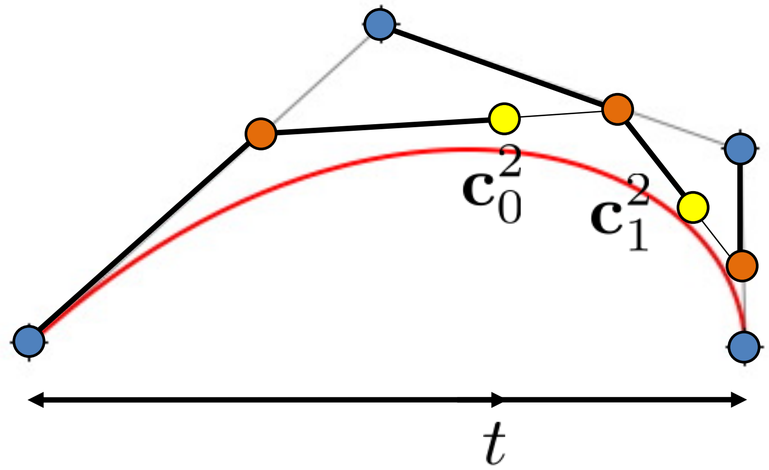


De Casteljau Algorithm

- Exploit recursive definition of Bernstein polynomials
  Repeated convex combination of control points

$$\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\mathbf{c}_{i+1}^{k-1} \qquad \mathbf{c}_i^0 := \mathbf{c}_i$$

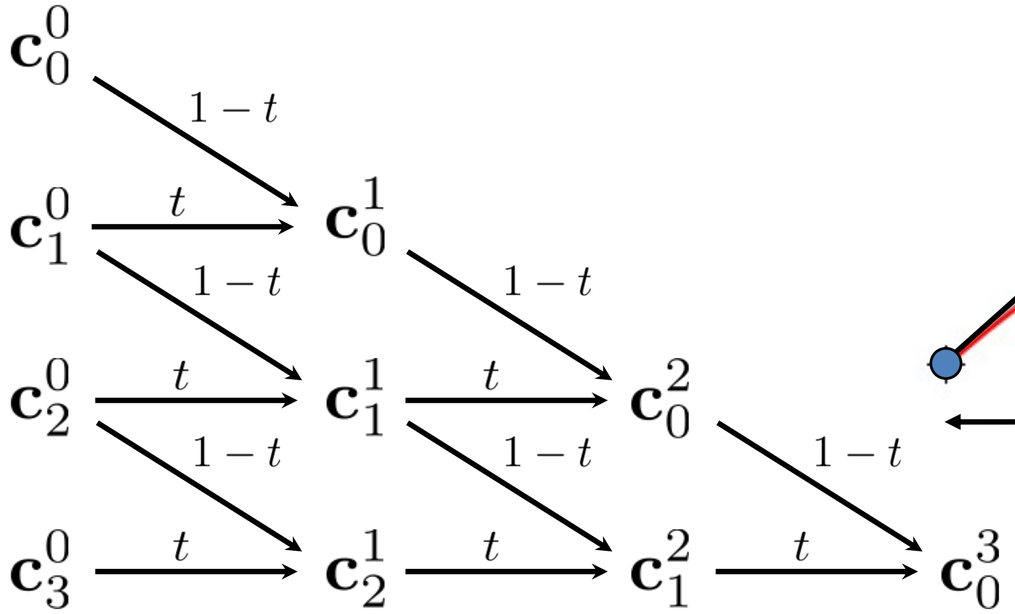


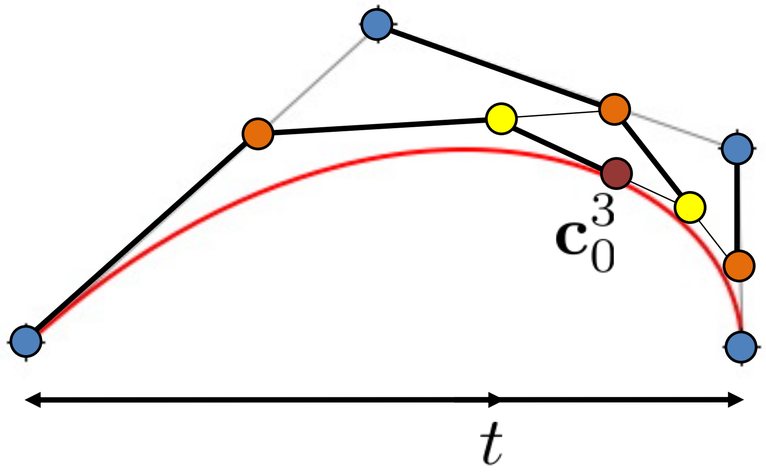


De Casteljau Algorithm

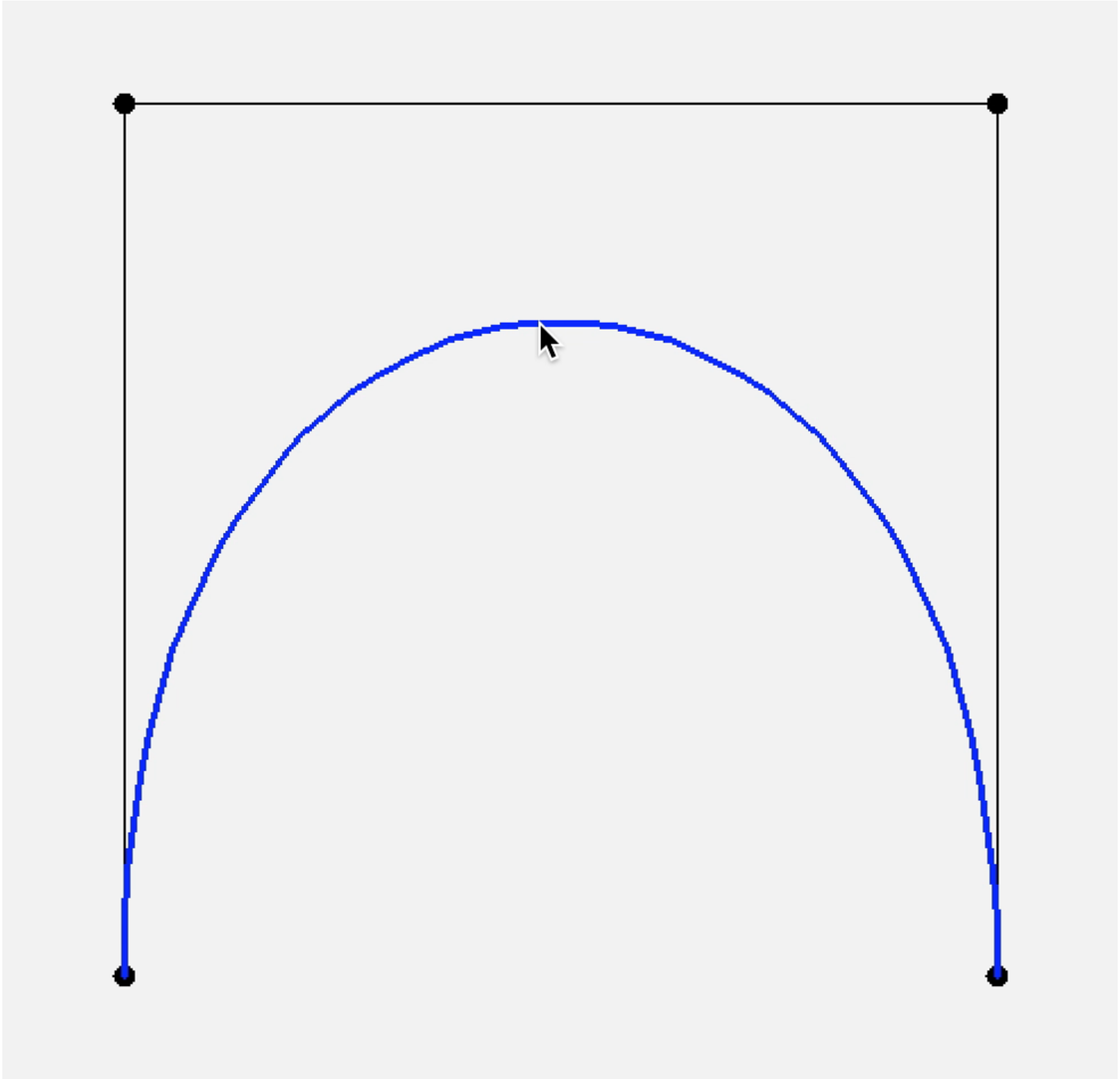
- Exploit recursive definition of Bernstein polynomials
  Repeated convex combination of control points

$$\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\mathbf{c}_{i+1}^{k-1} \qquad \mathbf{c}_i^0 := \mathbf{c}_i$$





De Casteljau Algorithm



i33www.ira.uka.de/applets/mocca/html/noplugin/BezierCurve/AppDeCasteljau/index.html

Disadvantages

- Still global support of basis functions for each curve segment
- Insertion of new control points?
- Continuity conditions restrict control polygon
  - No “inherent” smoothness control

Basis splines (B-splines)

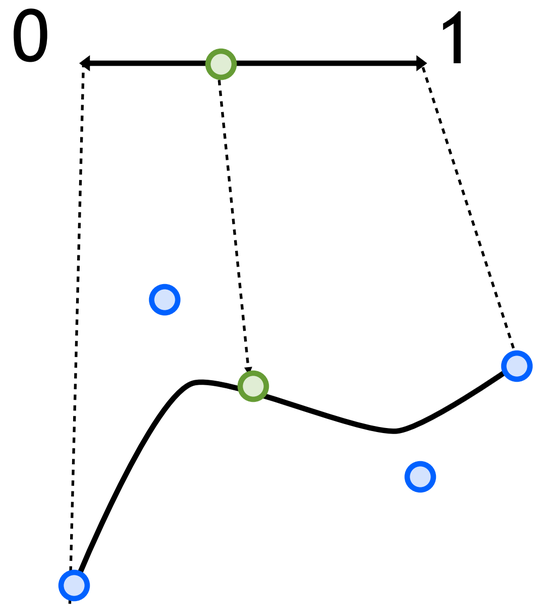
Why?

- The control points of Bezier curves have global support
- We can obtain local support if we split the curve into smaller Bezier segments, but we have to be careful with the borders to guarantee smoothness
- B-spline curves are a generalization of this construction

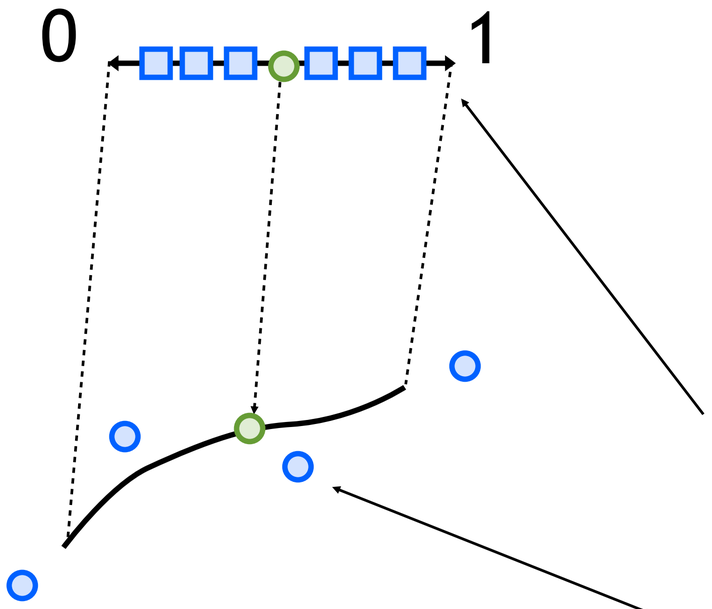
- http://i33www.ira.uka.de/applets/mocca/html/noplugin/inhalt.html

Ingredients

Bezier



B-spline



$$m = n + p + 1$$

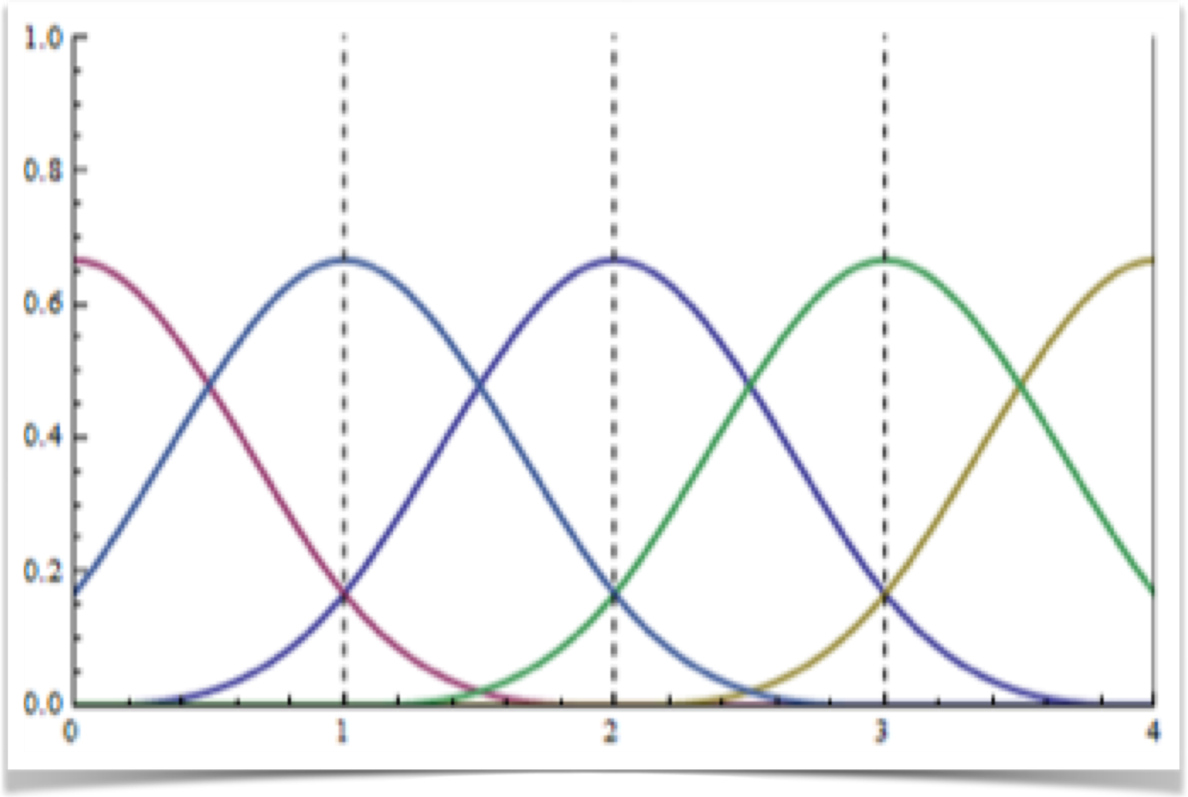
Basis degree p

Knots Points $m + 1$

Control Points $n + 1$

Basis functions

- Piecewise-polynomial
- C^p
- Symmetric
- Shifted
- Nonnegative
- Partition of unity
- Local support



Recursive definition

Degree

$$N_i^0(t) = \begin{cases} 1 & \text{if } i \le t < i+1 \\ 0 & \text{otherwise} \end{cases}$$

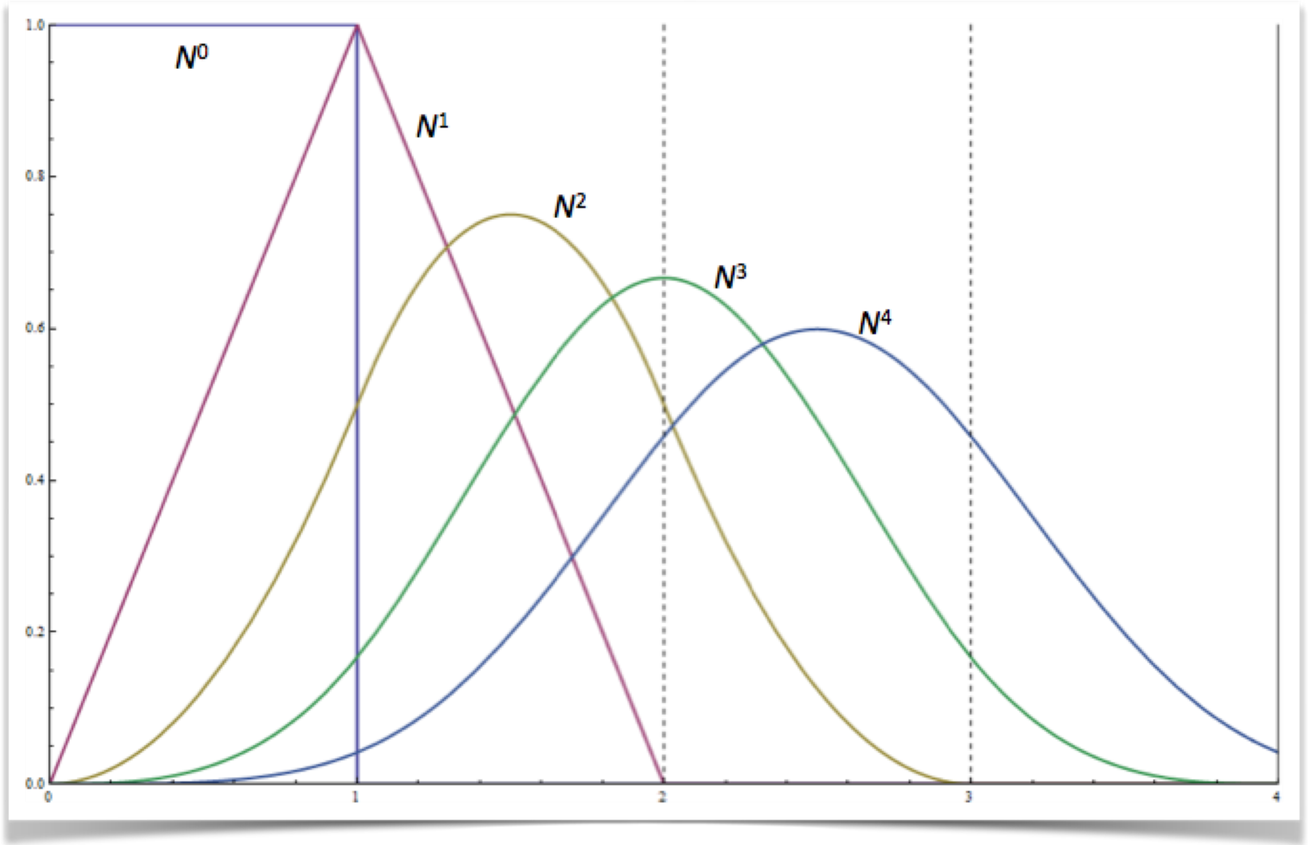
Knot Span

$$N_i^d = \left(\frac{t-i}{d}\right) N_i^{d-1} + \left(\frac{i+d+1-t}{d}\right) N_{i+1}^{d-1}$$

Non-uniform knots

$$N_{i,0}(u) = \begin{cases} 1 & if\ u_i \le u < u_{i+1} \\ 1 & otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$



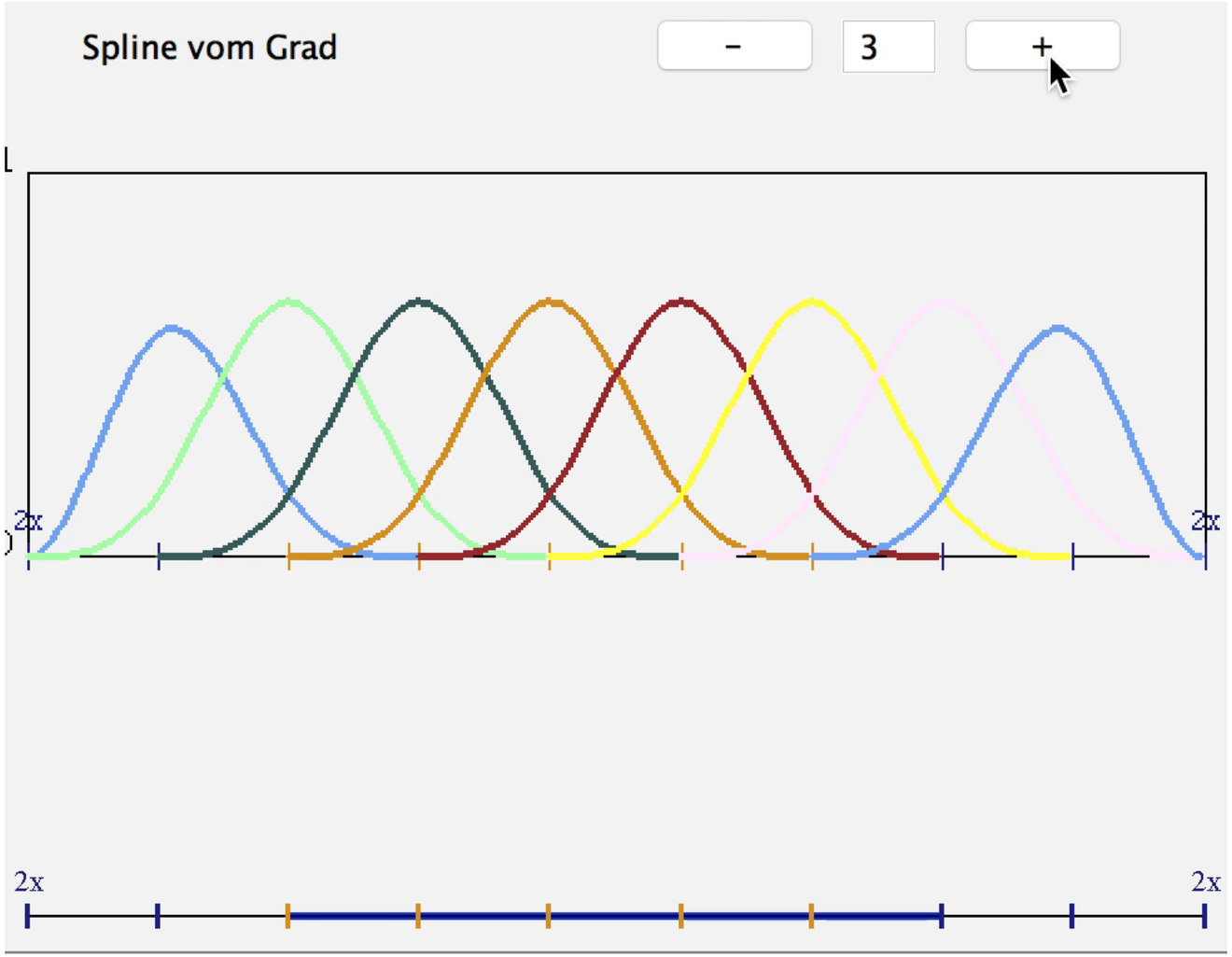
The support is always:

(1 + degree) knot spans

Basis functions

http://www.cs.technion.ac.il/~cs234325/Applets/applets/bspline/GermanApplet.html

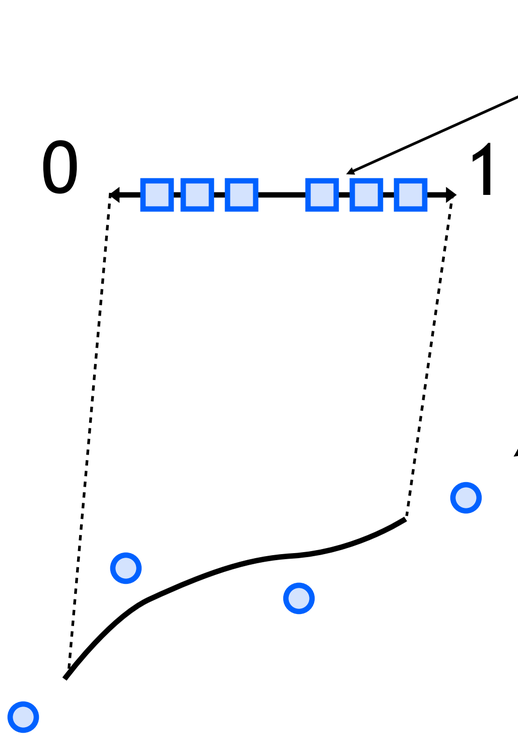
Basis functions

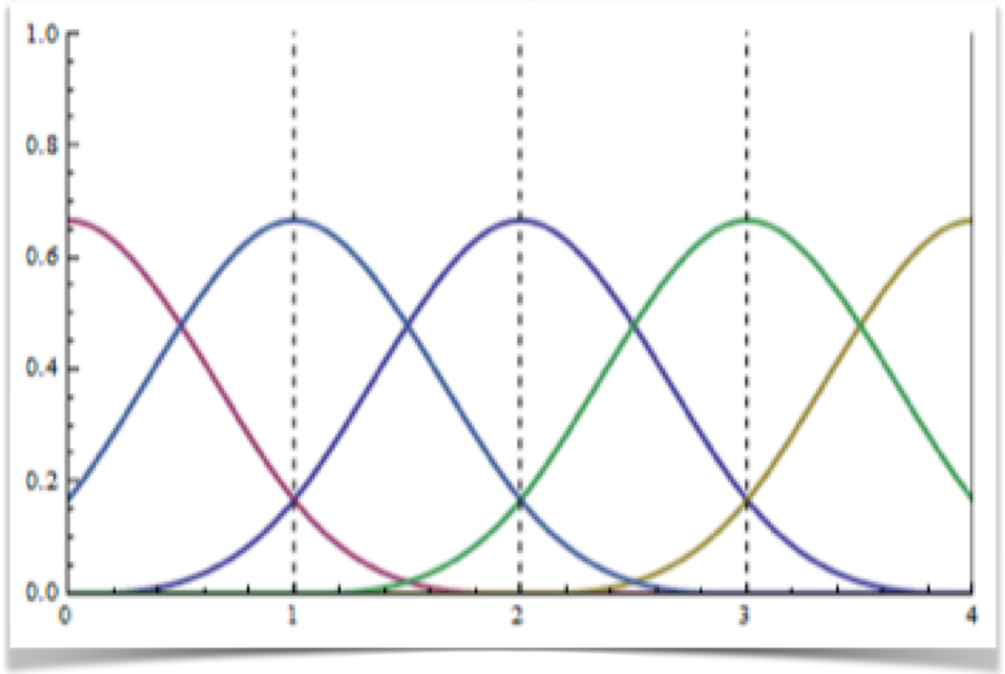


http://www.cs.technion.ac.il/~cs234325/Applets/applets/bspline/GermanApplet.html

Definition

$$\mathbf{p}(t) = \sum_i \mathbf{c}_i N_i^d(t)$$

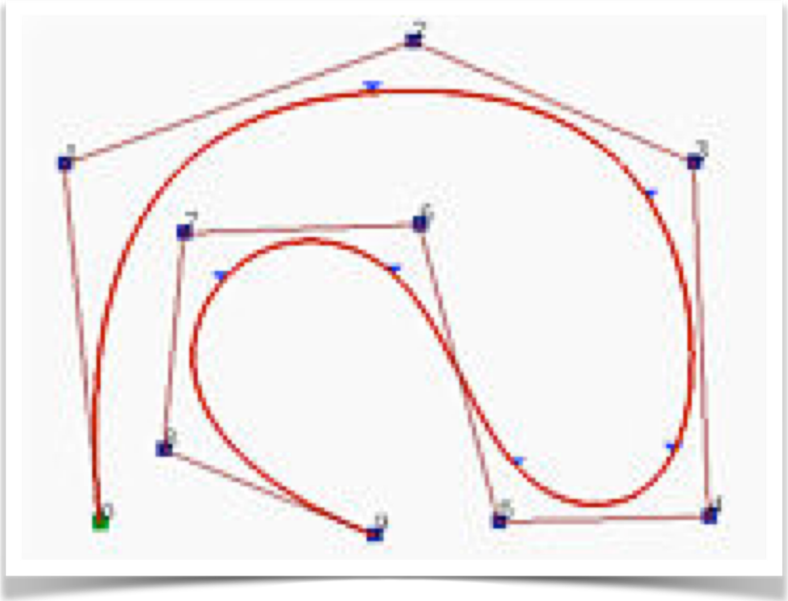




Special curves

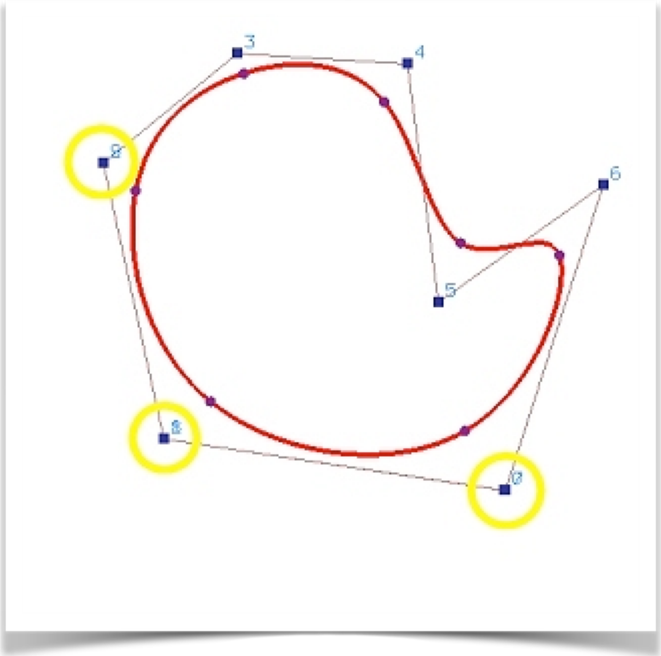
Interpolating curve

Increase the multiplicity of the first and last knots



Closed curve

Overlap the first and last control points, and align the corresponding knots



From http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-closed.html

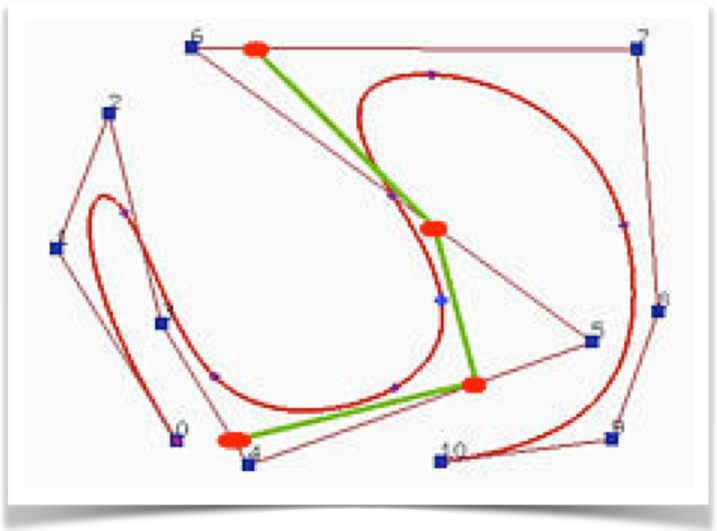
Properties

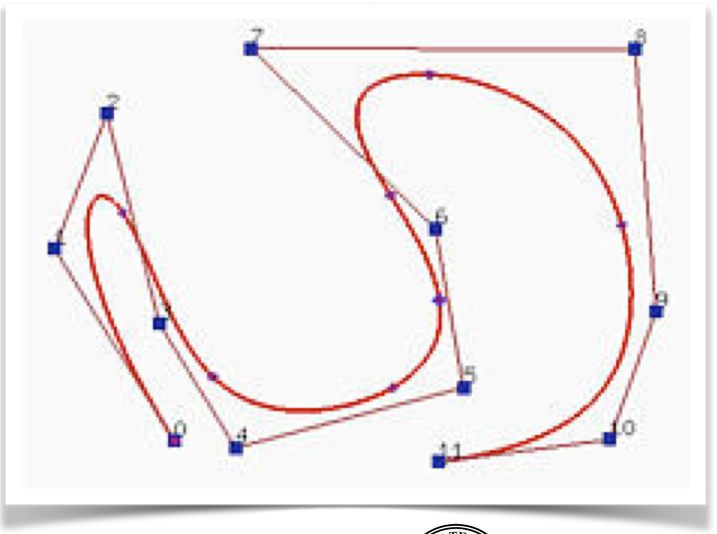
Notation: **m + 1** knots, **n+1** control points, **p** degree

1. B-spline curve is a piecewise curve with each component a curve of degree p
2. Equality m = n + p + 1 must be satisfied
3. Strong Convex Hull Property: A B-spline curve is contained in the convex hull of its control polyline
4. Local Modification Scheme: changing the position of control point P_i only affects the curve C(u) on interval $[u_i, u_{i+p+1})$
5. B-spline curve is C^{p-k} continuous at a knot of multiplicity k
6. Variation Diminishing Property
7. Bézier Curves Are Special Cases of B-spline Curves
8. Affine Invariance

Knot Insertion

- We want to insert a new **knot** without **changing the shape of the curve**
- Since m = n + p + 1, adding a knot must be compensated:
  - by changing the degree of the curve (global)
  - by adding a new control point (local)





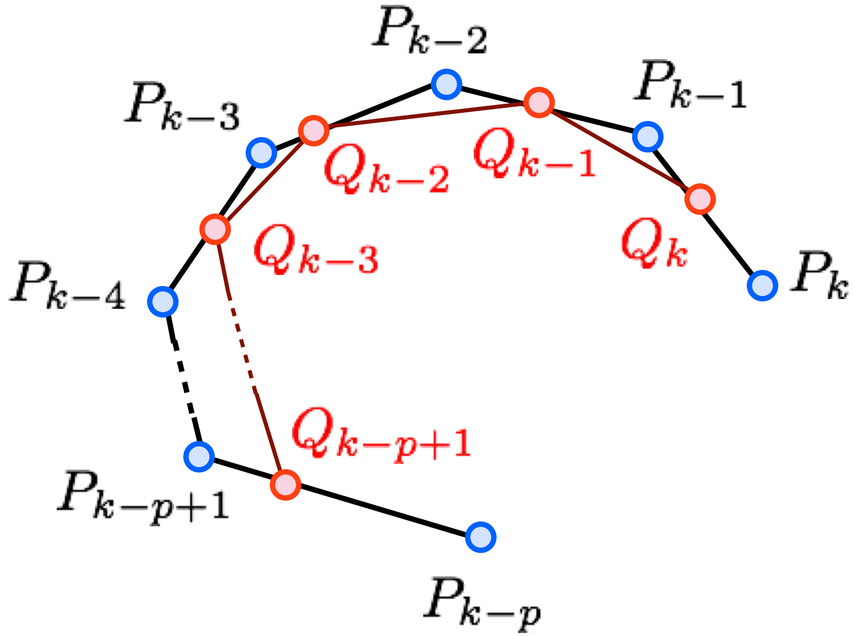
Inserting a knot

- Suppose the new knot t lies in knot span [uk, uk+1)
- Only the basis that corresponds to P_k, P_{k-1}, ... , P_{k-p} are non-zero
- Thus, the operation is **local**!
- To add the knot, we substitute the control points P_{k-p+1} to P_{k-1} with Q_{k-p+1} to Q_k using special corner cutting rules:

$$\mathbf{Q}_i = (1 - a_i)\mathbf{P}_{i-1} + a_i\mathbf{P}_i$$

$$a_i = \frac{t - u_i}{u_{i+p} - u_i} \qquad \text{for } k - p + 1 \leq i \leq k$$

- **Remember** to also add t to the knot vector!



Example

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}
0	0	0	0	0.2	0.4	0.6	0.8	1	1	1	1

- p = 3 (cubic)
- $P_2\,P_3\,P_4\,P_5$ are affected

0.5

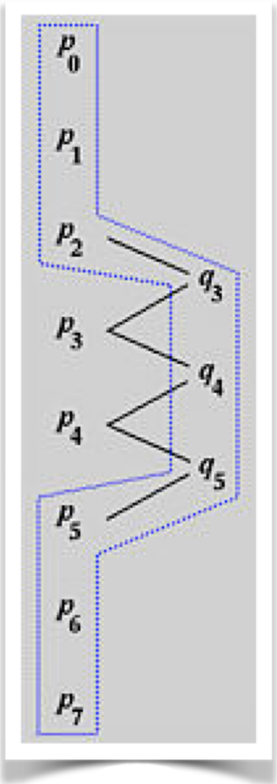
$$a_i = \frac{t - u_i}{u_{i+p} - u_i} \qquad \text{for } k - p + 1 \le i \le k$$

$$\mathbf{Q}_i = (1 - a_i)\mathbf{P}_{i-1} + a_i\mathbf{P}_i$$

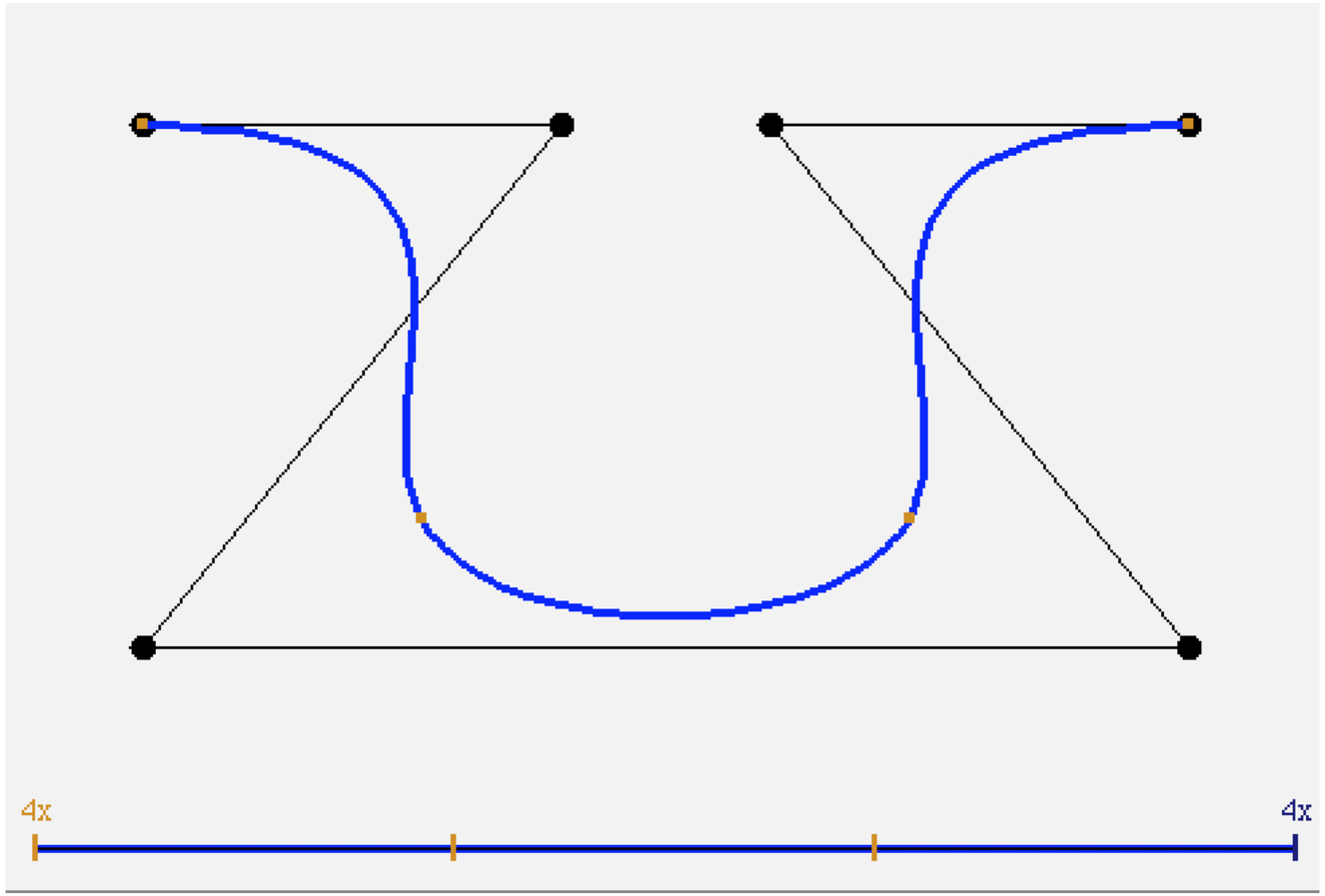
$$a_5 = \frac{t - u_5}{u_8 - u_5} = \frac{0.5 - 0.4}{1 - 0.4} = \frac{1}{6}$$
$$a_4 = \frac{t - u_4}{u_7 - u_4} = \frac{0.5 - 0.2}{0.8 - 0.2} = \frac{1}{2}$$
$$a_3 = \frac{t - u_3}{u_6 - u_3} = \frac{0.5 - 0}{0.6 - 0} = \frac{5}{6}$$

$$\mathbf{Q}_5 = \left(1 - \frac{1}{6}\mathbf{P}_4\right) + \frac{1}{6}\mathbf{P}_5$$
$$\mathbf{Q}_4 = \left(1 - \frac{1}{2}\mathbf{P}_3\right) + \frac{1}{2}\mathbf{P}_4$$
$$\mathbf{Q}_3 = \left(1 - \frac{5}{6}\mathbf{P}_2\right) + \frac{5}{6}\mathbf{P}_3$$

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}
0	0	0	0	0.2	0.4	0.5	0.6	0.8	1	1	1	1



Incremental rendering



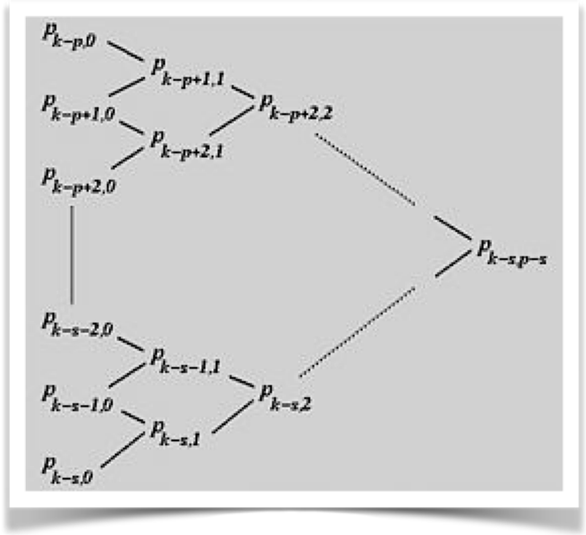
De Boor's Algorithm

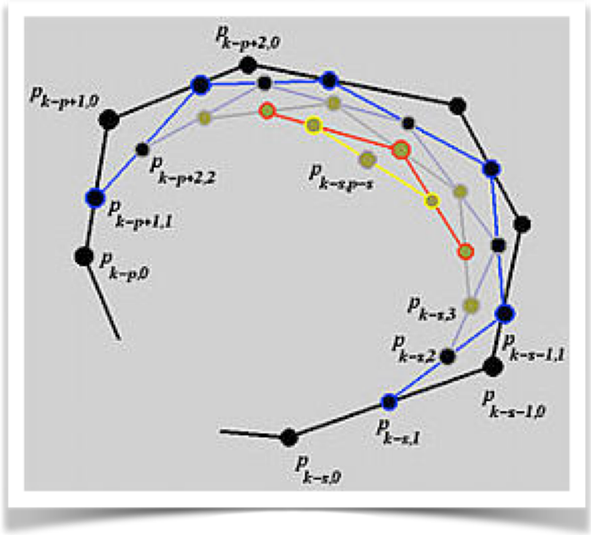
- It is a generalization of de Casteljau's algorithm
- It provides a numerically stable way to find a point on the B-spline
- The implementation requires to iteratively add new knots using the knot insertion algorithm
- The algorithm works because:
  - **If a knot u is inserted repeatedly so that its multiplicity is p, the last generated new control point is the point on the curve that corresponds to u**

De Boor's Algorithm

Algorithm

- Find the knot interval that corresponds to the point you want to evaluate
- Find the affected control points
- Start the corner cutting, until you have a single control point left



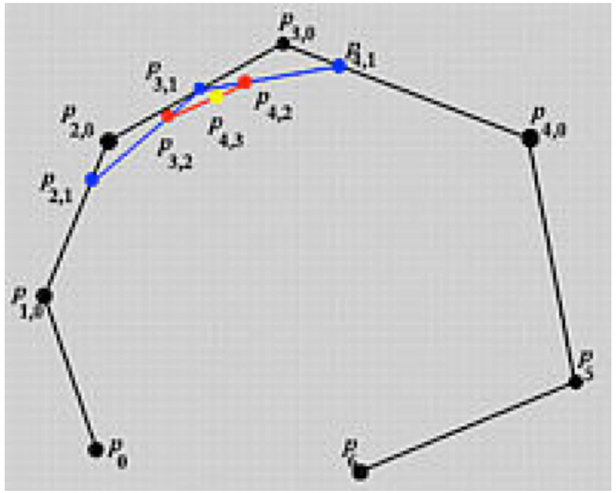


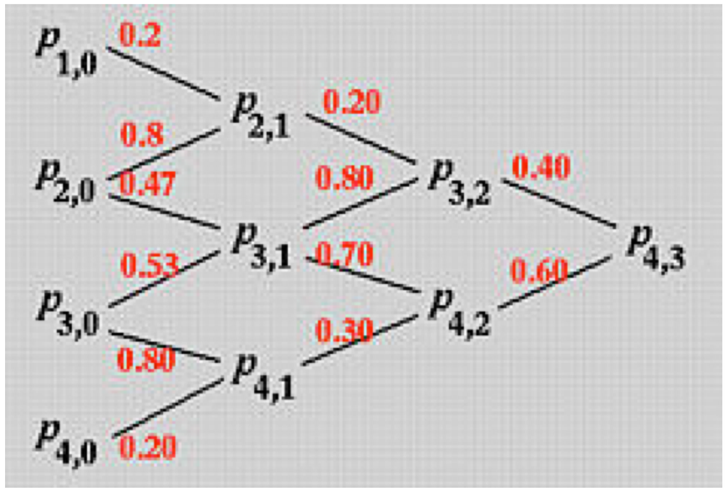
Example

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
0	0	0	0	0.25	0.5	0.75	1	1	1	1

0.4

- p = 3 (cubic)
- P_1 P_2 P_3 P_4 are affected





$$a_{4,3} = \frac{u - u_4}{u_{4+3-2} - u_4} = 0.6$$

$$a_{2,1} = \frac{u_{3+3-1} - u_3}{u_{2+3} - u_2} = 0.8$$

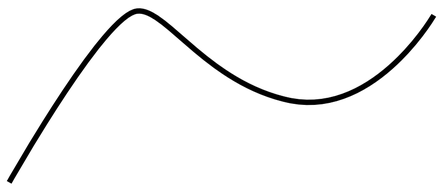
$$\mathbf{P}_{4,3} = (1 - a_{4,3})\mathbf{P}_{3,2} + a_{4,3}\mathbf{P}_{4,2} = 0.4\mathbf{P}_{3,2} + 0.6\mathbf{P}_{4,2}$$

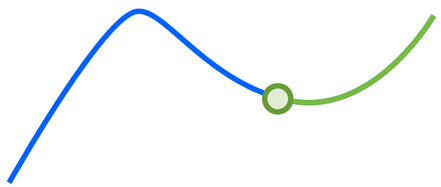
Relation with De Casteljau's

- The algorithm is similar to de Casteljau's, but with two important differences:
  - The weights used in the corner cutting **change** at every subdivision step
  - The effect of the corner cutting is **local**

Subdividing a B-spline

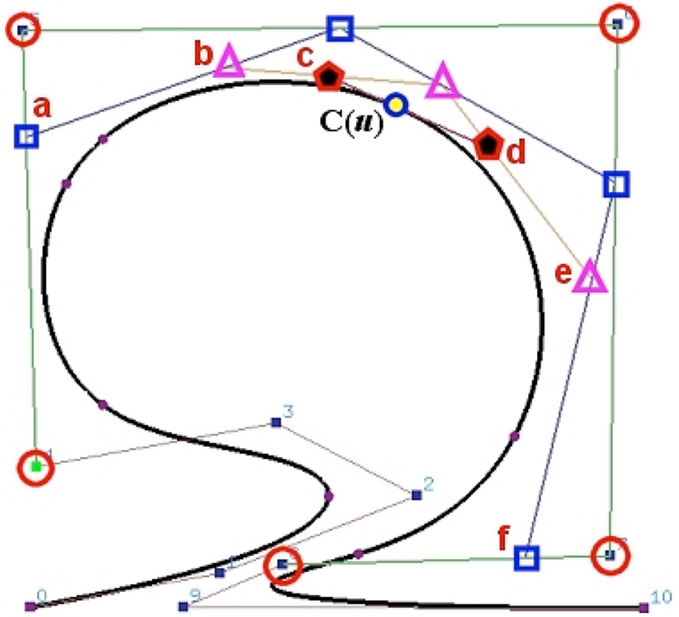
- We want to split a B-spline into two different curves:
  - let u be the splitting point
  - we want to define two B-splines defined on [0,u] and [u,1]

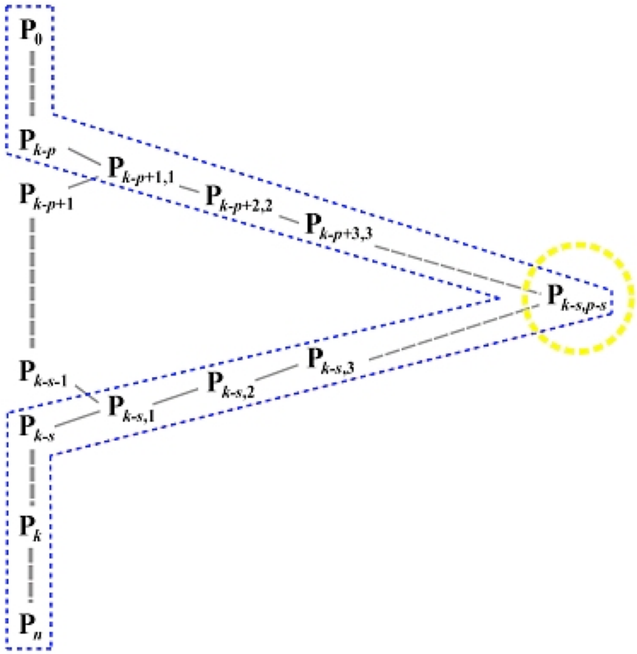




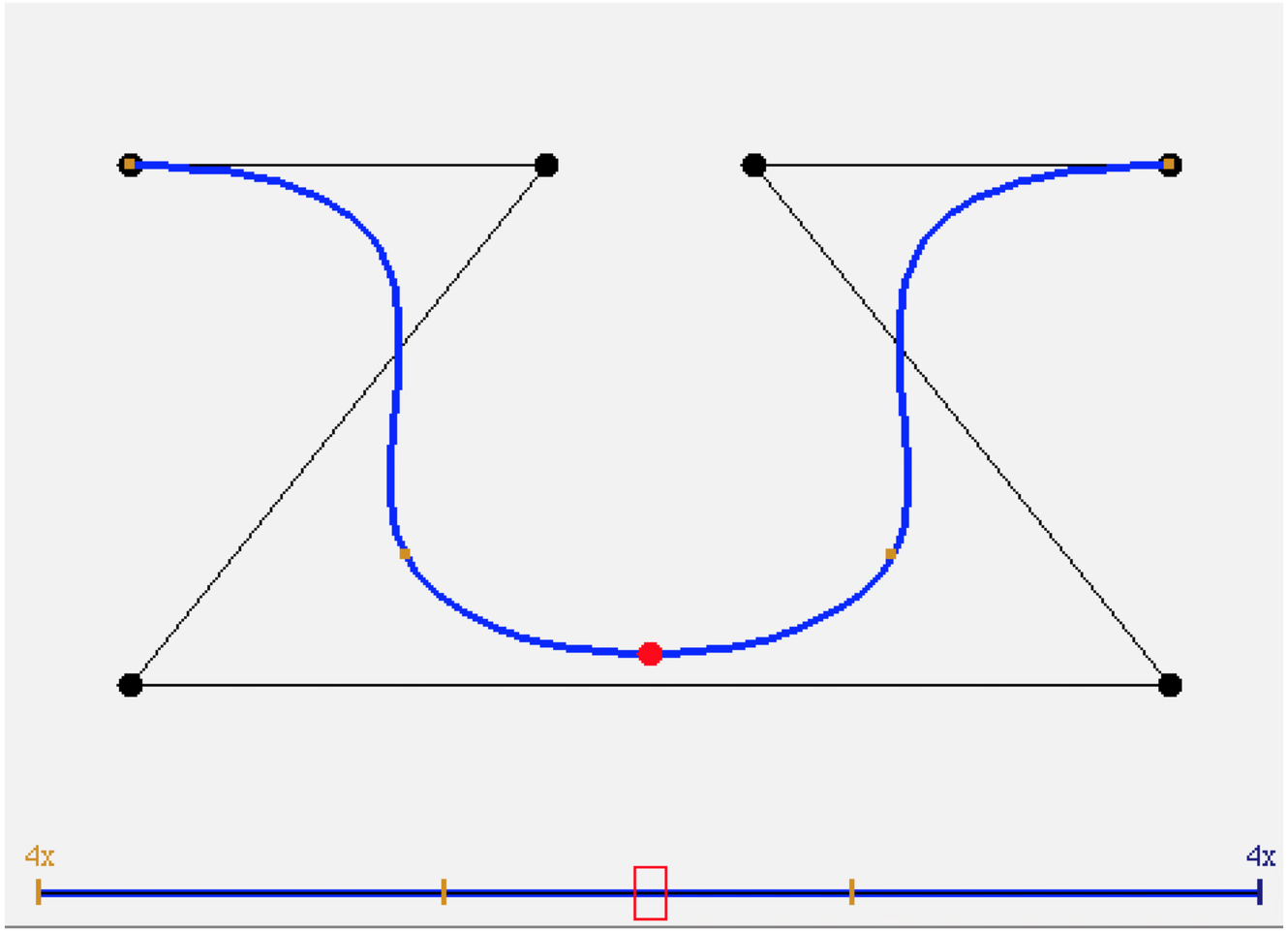
Subdivision algorithm

- Apply de Boor's at u
- Traverse the control points vector always turning right
- The knot vector for the first curve, contains all the knots in [0,u) followed by p+1 copies of u





Example



Subdividing a B-spline into Bézier segments

- If you subdivide a B-spline curve at every knot, then each curve segment becomes a Bézier curve of degree p
- This results follows from this lemma, that links the B-spline basis functions with the Bézier basis functions:
  - **If the first [last] n+1 knots are 0 [1], then the i-th B-spline basis function of degree n is identical to the i-th Bézier basis function for all i in the range of 0 and n**
- This means that the Bézier basis functions are special cases of B-spline basis, and that Bézier curves are special cases of B-spline curves

References

Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 15

Course notes: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/