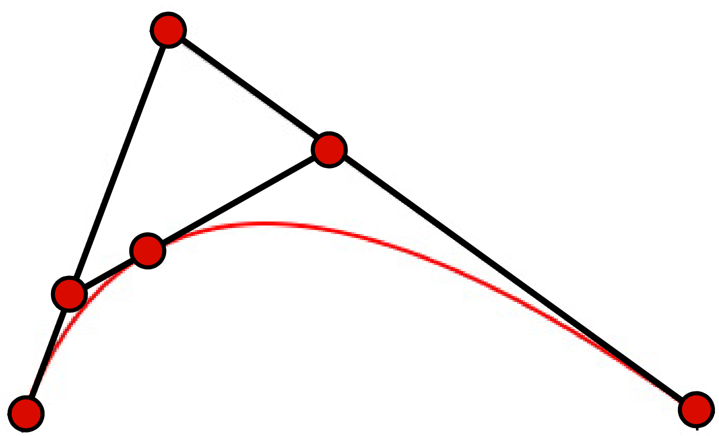
G22.3033-008, Spring 2010

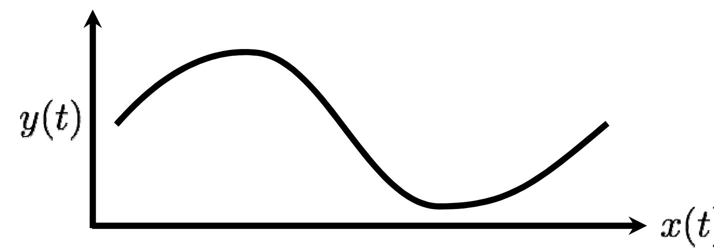
Geometric Modeling

Parametric Curves



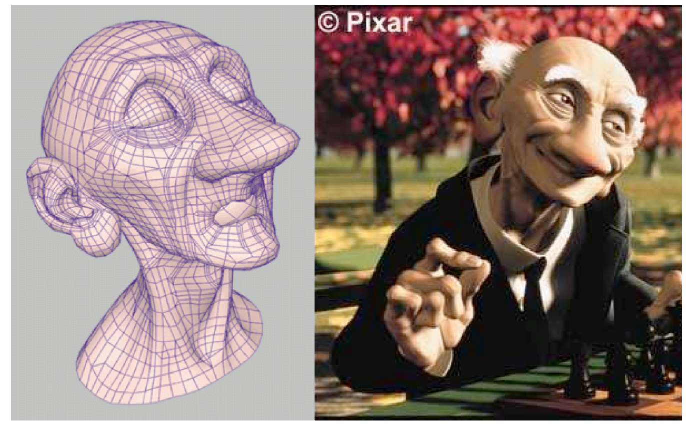
Representation

- Explicit parametric
  - Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$



Representation

- Explicit parametric
  - Range of a function $f : X \to Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$
  - Surface in 3D: $m = 2, n = 3$



Representation

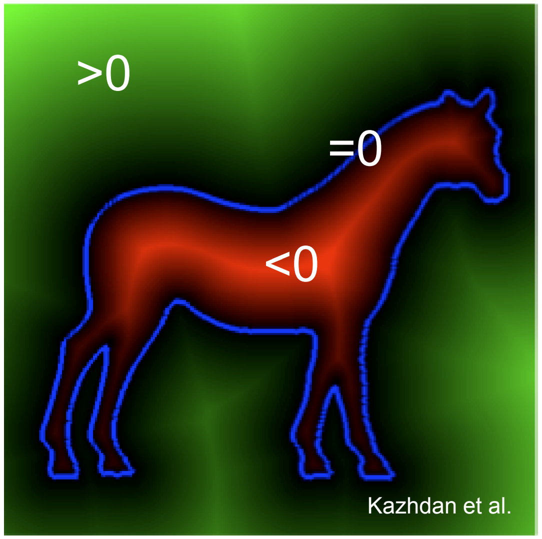
- Explicit parametric
  - Range of a function $f : X \to Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$
  - Surface in 3D: $m = 2, n = 3$
  - Volumetric density: $m = 3, n = 1$

Representation

- Explicit parametric
  - Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$
  - Surface in 3D: $m = 2, n = 3$
  - Volumetric density: $m = 3, n = 1$
- Implicit
  - Kernel of a function $f : \mathbb{R}^m \rightarrow \mathbb{R}$
  - Zero level set

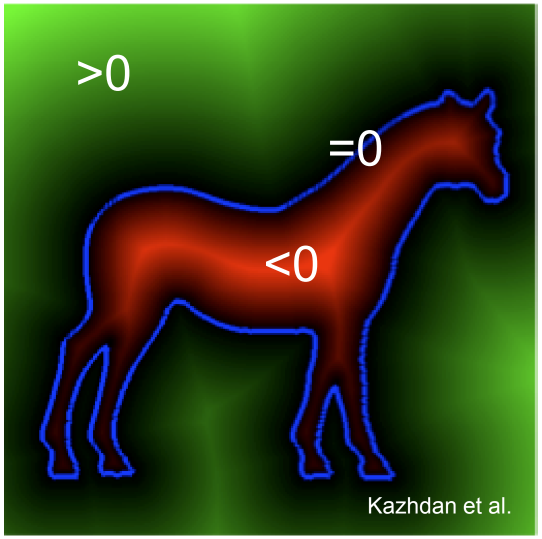
Representation

- Explicit parametric
  - Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$
  - Surface in 3D: $m = 2, n = 3$
  - Volumetric density: $m = 3, n = 1$
- Implicit
  - Kernel of a function $f : \mathbb{R}^m \rightarrow \mathbb{R}$
  - Zero level set
  - Curve in 2D: $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$



Representation

- Explicit parametric
  - Range of a function $f : X \rightarrow Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$
  - Curve in 2D: $m = 1, n = 2$
  - Surface in 3D: $m = 2, n = 3$
  - Volumetric density: $m = 3, n = 1$
- Implicit
  - Kernel of a function $f : \mathbb{R}^m \rightarrow \mathbb{R}$
  - Zero level set
  - Curve in 2D: $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$
  - Surface in 3D: $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$



Representation

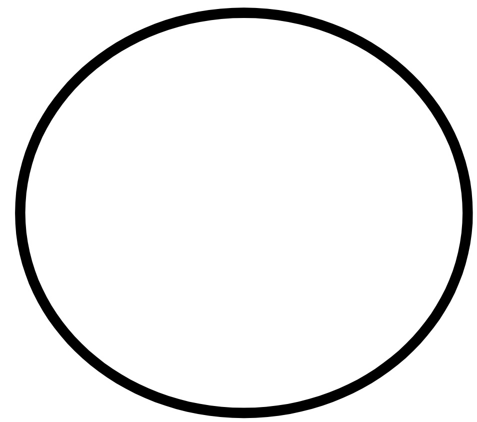
- Parametric:

$$f(t) = \begin{pmatrix} r\cos t \\ r\sin t \end{pmatrix}, S = f([0, 2\pi])$$

- Implicit:

$$f(x, y) = x^2 + y^2 - r^2$$

$$S = \{(x, y) \in \mathbb{R}^2 | f(x, y) = 0\}$$

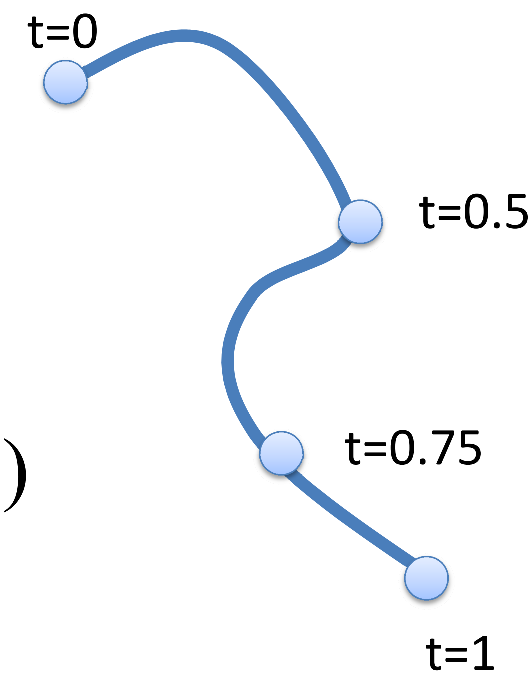


Parametric curves

- Curves are 1-dimensional parameterizations

- Planar curve: $f(t) = (x(t), y(t))$

- Space curve: $f(t) = (x(t), y(t), z(t))$



Parametric Curves

Continuity and regularity

- Line segment $f:[a,b]\to R^d, d=1,2,3,\ldots$
- The same segment can be parameterized differently

$$\mathbf{p}_1:[0,1]\to R^3,\quad f(t)=t\,\mathbf{p}_1+(1-t)\mathbf{p}_2$$

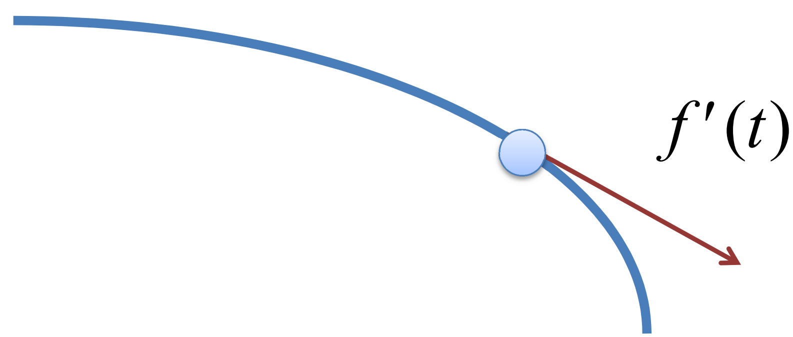
$$\mathbf{p}_2:[0,1]\to R^3,\quad f(t)=t^2\mathbf{p}_1+(1-t^2)\mathbf{p}_2$$

Parametric Curves

Continuity and regularity

- A parametric curve is n-times continuously differentiable if the image f is n-times continuously differentiable (C^n)
- The derivative $f'(t)$ at position t is a tangent vector
- A curve is regular when f is differentiable and $f'(t) \neq 0$

$$f'(t) = (x'(t), y'(t), \ldots)$$



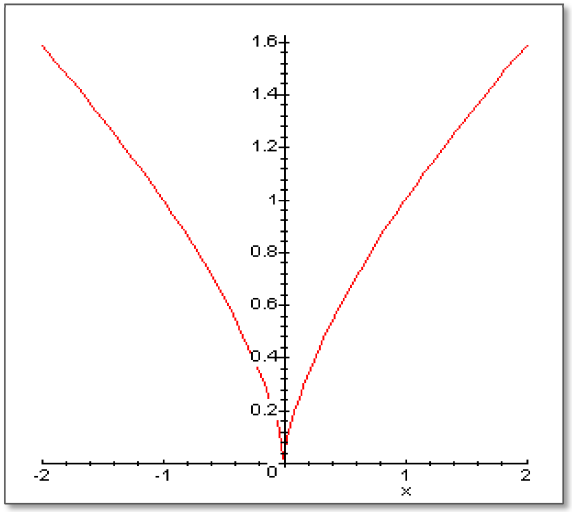
Parametric Curves

Continuity and regularity

- Example

$$f:[-2,2]\rightarrow R^3,\ \ f(t)=(t^3,t^2,0)$$

$$f'(t)=(3t^2,2t,0)\ \Rightarrow\ f'(0)=0$$



- f is continuously differentiable, but not regular at $t=0$

- The regularity of a curve can be interpreted as its visual smoothness

Parametric Curves

Arc length parameterization

- A curve is parameterized by arc length when

$$\left\| f'(t) \right\| = 1, \;\; t \in [a,b]$$

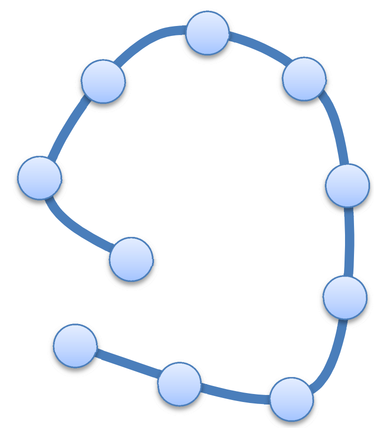
- Any regular curve can be parameterized by arc length

- For arc length parameterized curves:

$T(s) := f'(s)$ Tangent vector

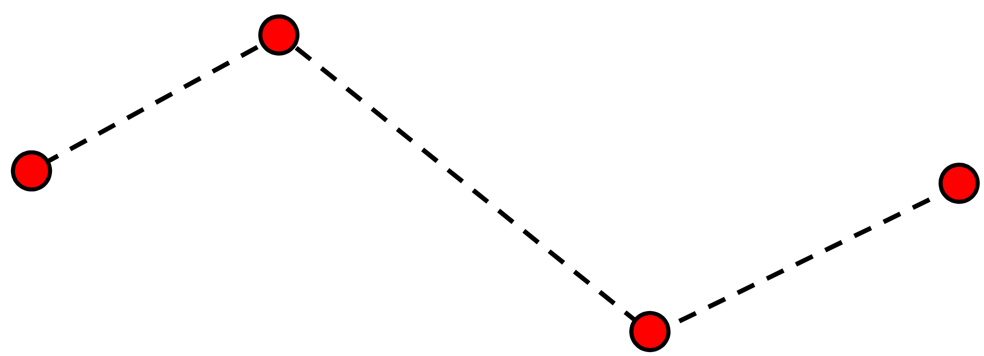
$K(s) := f''(s)$ Curvature vector

$\kappa(s) := \left\| f''(s) \right\|$ Curvature (scalar)



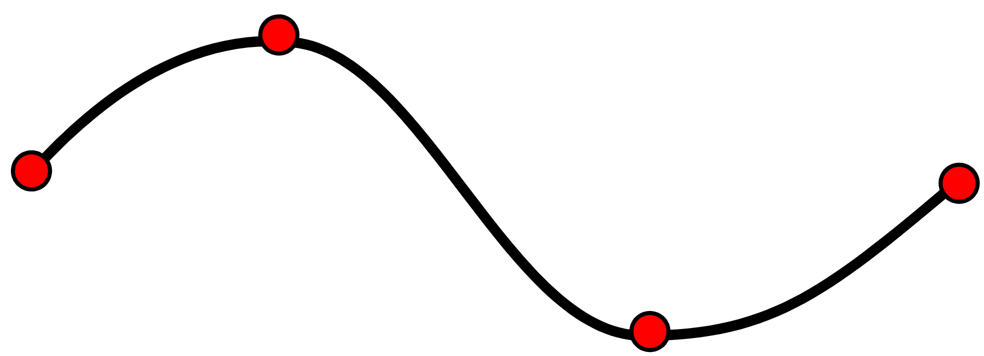
Smooth Curves (2D)

- Goal: intuitive modeling tool for curves
- User inputs points

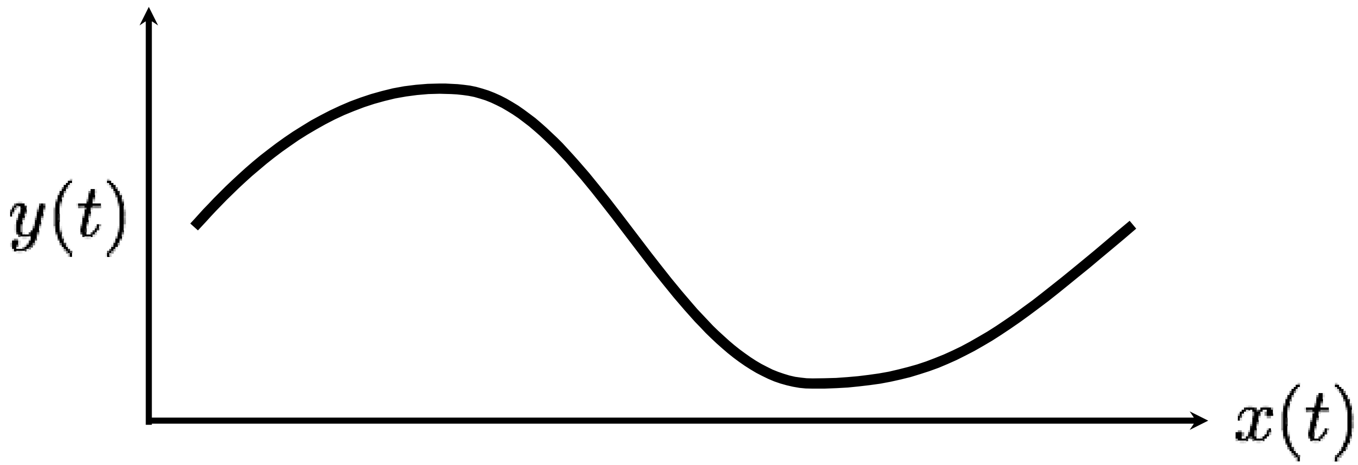


Smooth Curves (2D)

- Goal: intuitive modeling tool for curves
- User inputs points
- Find a curve that interpolates/approximates the points
- Allow user to change the curve (how?)



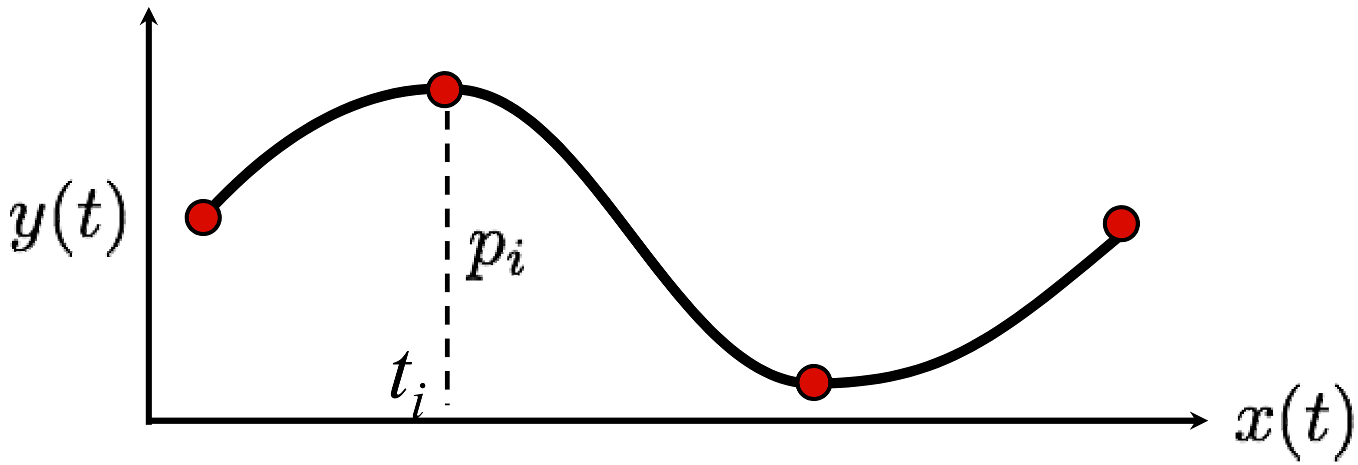
Polynomial Curves



- Parametric form with polynomials

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \ldots \end{pmatrix}$$

Polynomial Curves



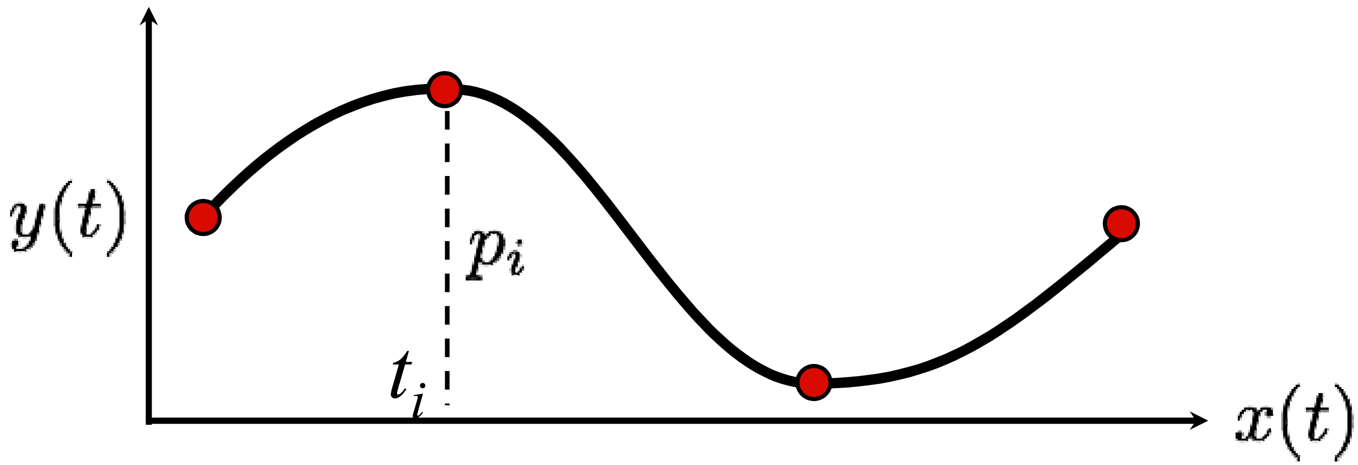
- Parametric form with polynomials

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \dots \end{pmatrix}$$

- Interpolated control points p_i with $y(t_i) = p_i$

$$\underbrace{\begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix}}_{\text{Basis}} \underbrace{\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \end{pmatrix}^T}_{\text{Coefficients}} = y(t)$$

Polynomial Curves



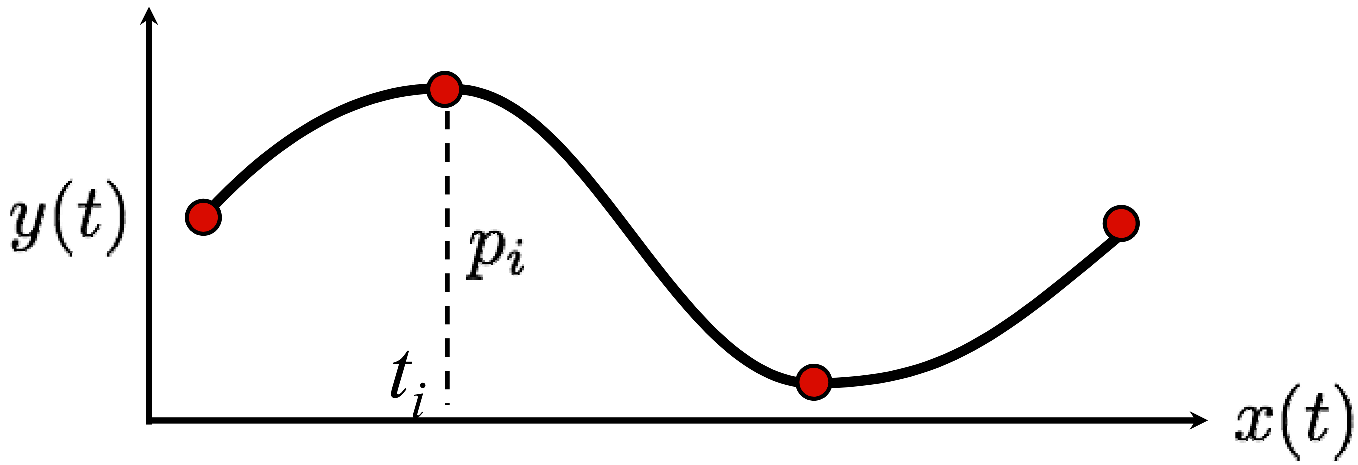
- Parametric form with polynomials

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \ldots \end{pmatrix}$$

- Interpolated control points p_i with $y(t_i) = p_i$
- Solve

$$\begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Polynomial Curves



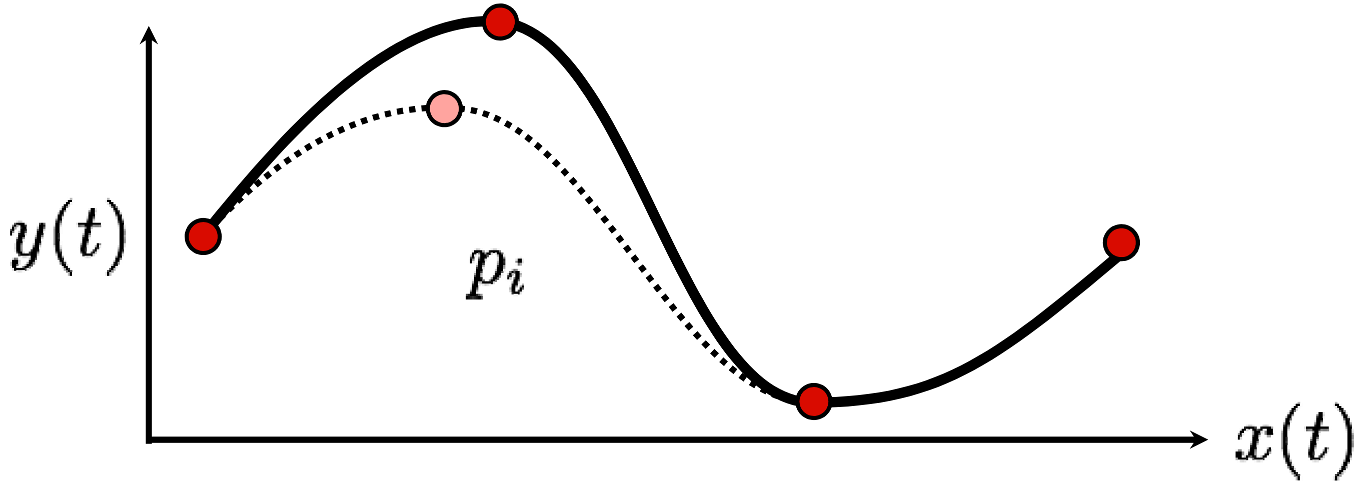
- Parametric form with polynomials

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \ldots \end{pmatrix}$$

- Interpolated control points p_i with $y(t_i) = p_i$
- Solve

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \end{pmatrix}^{-1} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Polynomial Curves



- Parametric form with polynomials

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \ldots \end{pmatrix}$$

- Interpolated control points p_i with $y(t_i) = p_i$
- Solve

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \end{pmatrix}^{-1} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

Polynomial Curves

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

$$f(t) = \sum_{i=0}^{n} c_i F_i^n(t)$$

Coefficients

$$c_i \in \mathbb{R}^k$$

Basis functions

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

$$F_i^n(t) = t^i$$

Problems

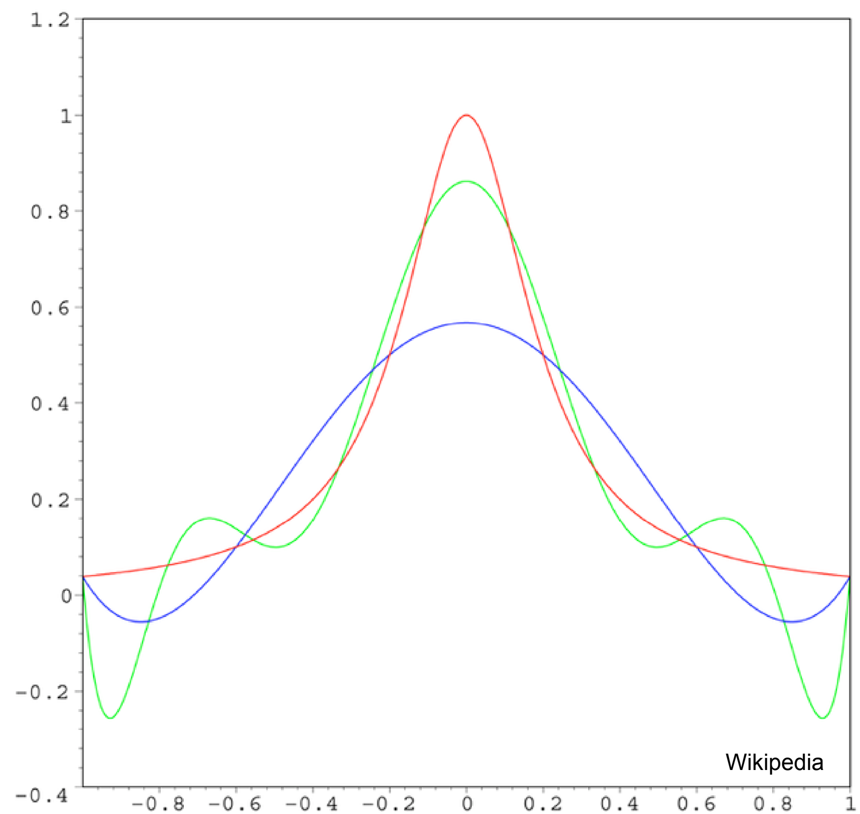
- Sum of monomials $f(t) = \sum_{i=0}^{n} c_i t^i$

Problems

- Sum of monomials $f(t) = \sum_{i=0}^{n} c_i t^i$
  - Not an affine combination (don't sum up to 1)
  - Coefficients $c_i \in \mathbb{R}^k$ don't have geometric meaning

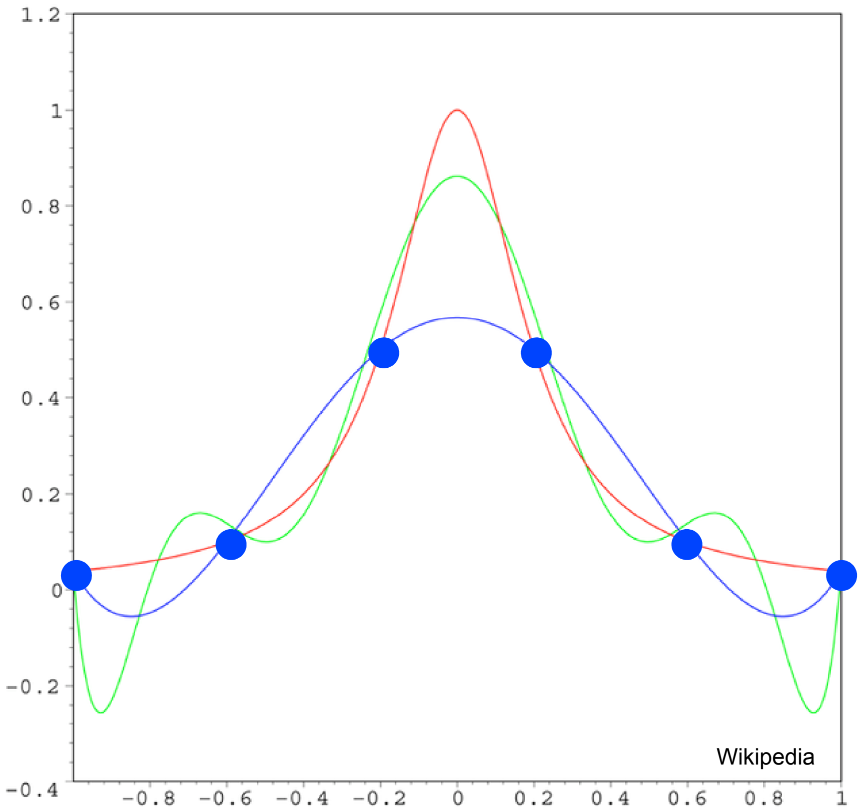
Problems

- Sum of monomials $f(t) = \sum_{i=0}^{n} c_i t^i$
  - Not an affine combination (don't sum to 1)
  - Coefficients $c_i \in \mathbb{R}^k$ don't have geometric meaning
- More control points?
  - More coefficients
  - Higher degree
  - Interpolation oscillates
  - Runge phenomenon



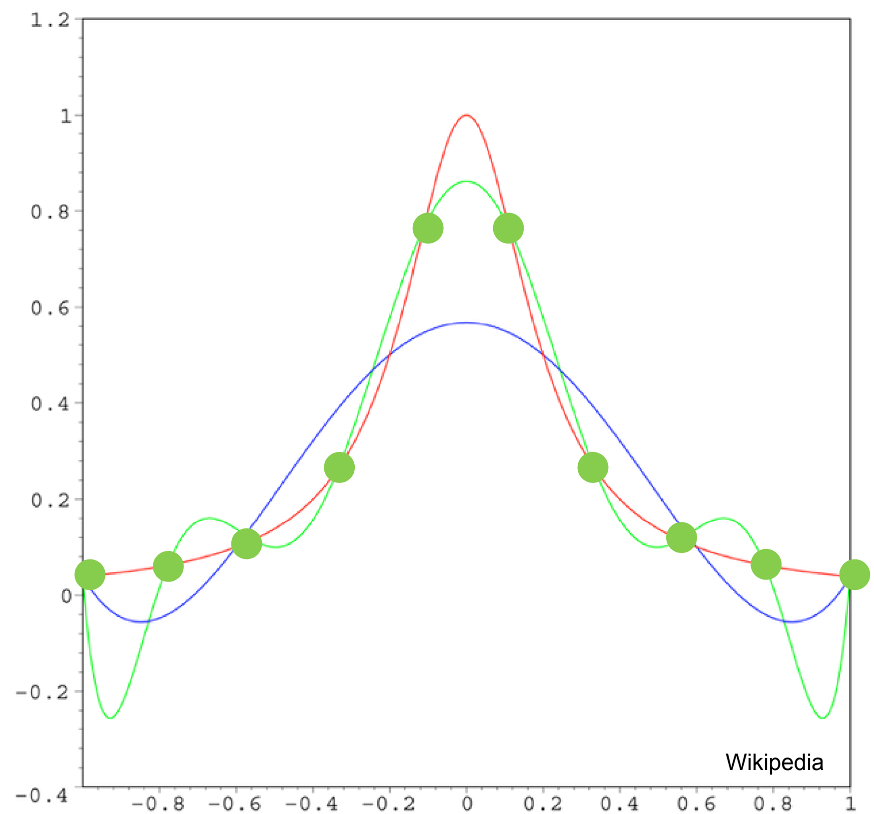
Problems

- Sum of monomials $f(t) = \sum_{i=0}^{n} c_i t^i$
  - Not an affine combination (don't sum to 1)
  - Coefficients $c_i \in \mathbb{R}^k$ don't have geometric meaning
- More control points?
  - More coefficients
  - Higher degree
  - Interpolation oscillates
  - Runge phenomenon



Problems

- Sum of monomials $f(t) = \sum_{i=0}^{n} c_i t^i$
  - Not an affine combination (don't sum to 1)
  - Coefficients $c_i \in \mathbb{R}^k$ don't have geometric meaning
- More control points?
  - More coefficients
  - Higher degree
  - Interpolation oscillates
  - Runge phenomenon



What might be good basis functions?

- Intuitive editing
  - **Control points** are coefficients
  - Predictable behavior
  - No oscillation
  - Local control
- Mathematical guarantees
  - Smoothness, affine invariance, linear precision, ...
- Efficient processing and rendering

What might be good basis functions?

- Lagrange polynomials

$$F_i^n(t) = \prod_{j=0, j\neq i}^{n} \frac{(t - t_j)}{(t_i - t_j)}$$

  – Oscillation, accuracy, shape preservation

What might be good basis functions?

- Lagrange polynomials

$$F_i^n(t) = \prod_{j=0, j \neq i}^{n} \frac{(t - t_j)}{(t_i - t_j)}$$

  - Oscillation, accuracy, shape preservation

- Hermite interpolation
  - Points and derivatives
  - Domain-dependent, e.g., (affine) transformations

What might be good basis functions?

- Lagrange polynomials

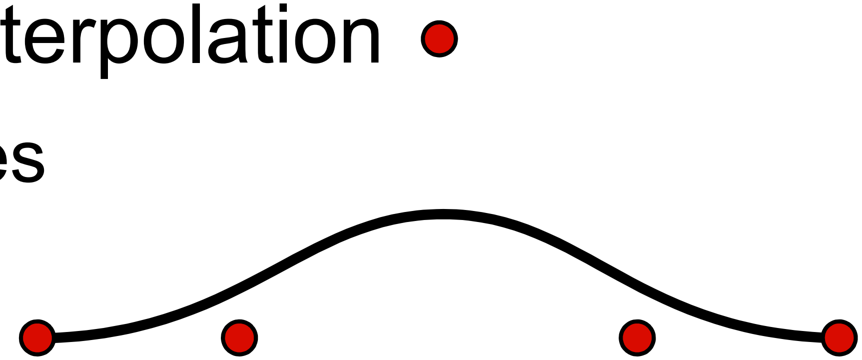
$$F_i^n(t) = \prod_{j=0, j\neq i}^{n} \frac{(t - t_j)}{(t_i - t_j)}$$

  - Oscillation, accuracy, shape preservation

- Hermite interpolation

  - Points and derivatives
  - Domain-dependent, e.g., (affine) transformations

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



Bernstein Polynomials

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

– Binomial coefficients $\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{otherwise} \end{cases}$

Bernstein Polynomials

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

– Binomial coefficients $\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}$

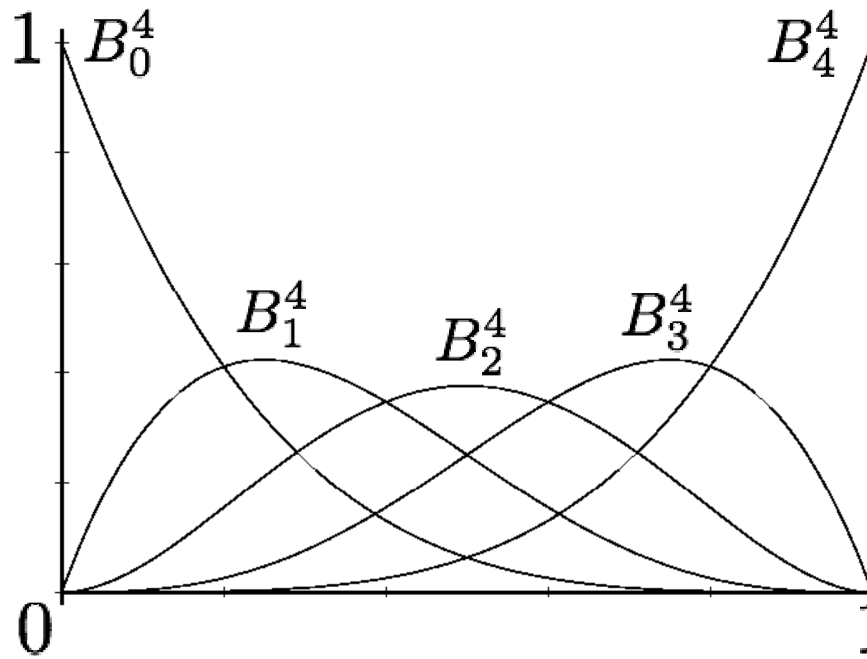
– $n = 1 : t, (1-t)$
– $n = 2 : t^2, 2t(1-t), (1-t)^2$
– $n = 3 : t^3, 3t^2(1-t), 3t(1-t)^2, (1-t)^3$

Bernstein Polynomials

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

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

$$B_0^0(t) = 1, B_j^n(t) = 0, j \notin \{0,\ldots,n\}$$



Bezier Curves

- General parametric form

$$f(t) = \sum_{i=0}^{n} c_i F_i^n(t)$$

Coefficients $c_i \in \mathbb{R}^k$

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

Bezier Curves

- Curve based on Bernstein polynomials

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

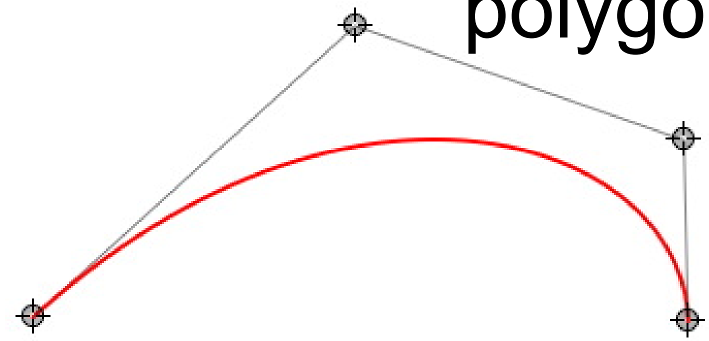
Coefficients $c_i \in \mathbb{R}^k$

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

Bezier Curves

- Curve based on Bernstein polynomials

Control polygon



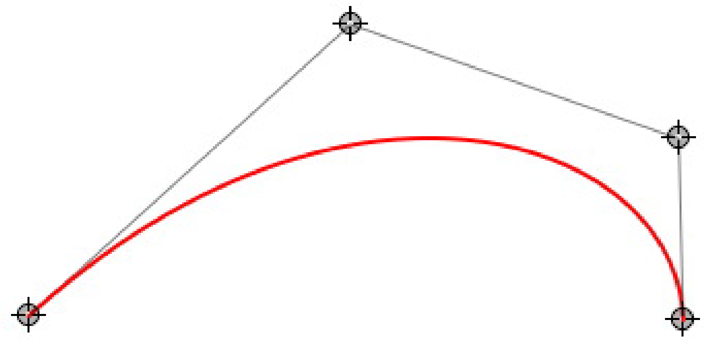
$$f(t) = \sum_{i=0}^{n} c_i B_i^n(t)$$

Control points $c_i \in \mathbb{R}^k$

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

Properties of Bezier Curves

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



Efficient Computation?

De Casteljeau Algorithm

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

$$c_i^k = (1-t)c_i^{k-1} + tc_{i+1}^{k-1} \qquad c_i^0 := c_i$$

De Casteljeau Algorithm

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

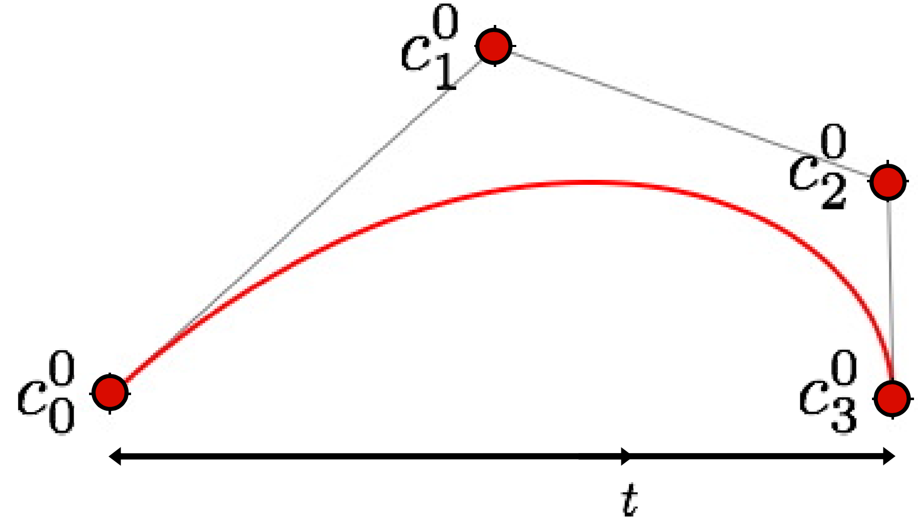
$$c_i^k = (1-t)c_i^{k-1} + tc_{i+1}^{k-1} \qquad c_i^0 := c_i$$

c_0^0

c_1^0

c_2^0

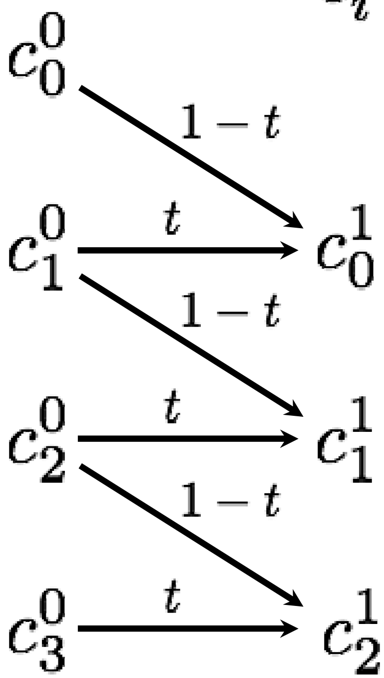
c_3^0

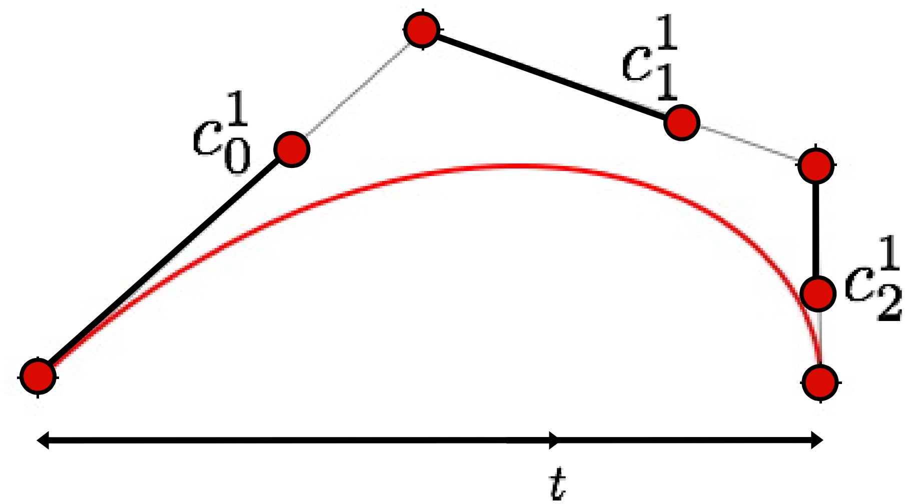


De Casteljeau Algorithm

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

$$c_i^k = (1-t)c_i^{k-1} + t c_{i+1}^{k-1} \qquad c_i^0 := c_i$$



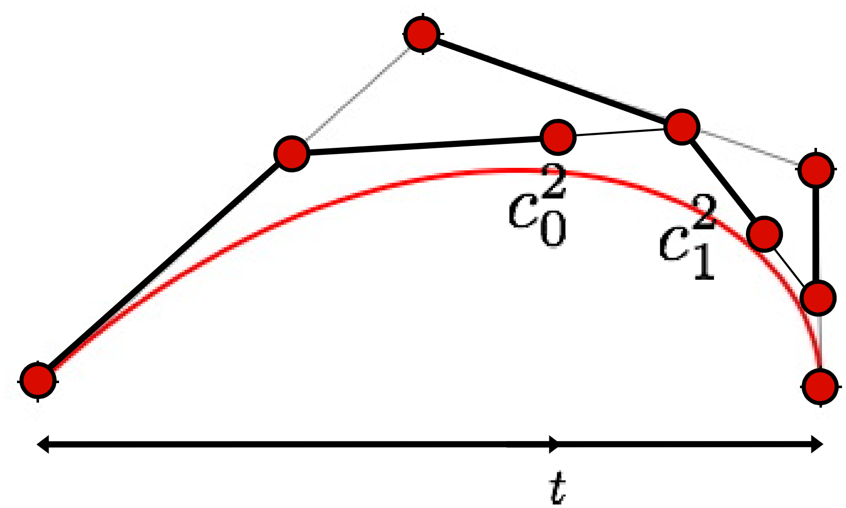


De Casteljeau Algorithm

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

$$c_i^k = (1-t)c_i^{k-1} + tc_{i+1}^{k-1} \qquad c_i^0 := c_i$$

$$\begin{tikzcd}
c_0^0 \arrow[dr, "1-t"] & & \\
c_1^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^1 \arrow[dr, "1-t"] & \\
c_2^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_1^1 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^2 \\
c_3^0 \arrow[r, "t"] & c_2^1 \arrow[r, "t"] & c_1^2
\end{tikzcd}$$

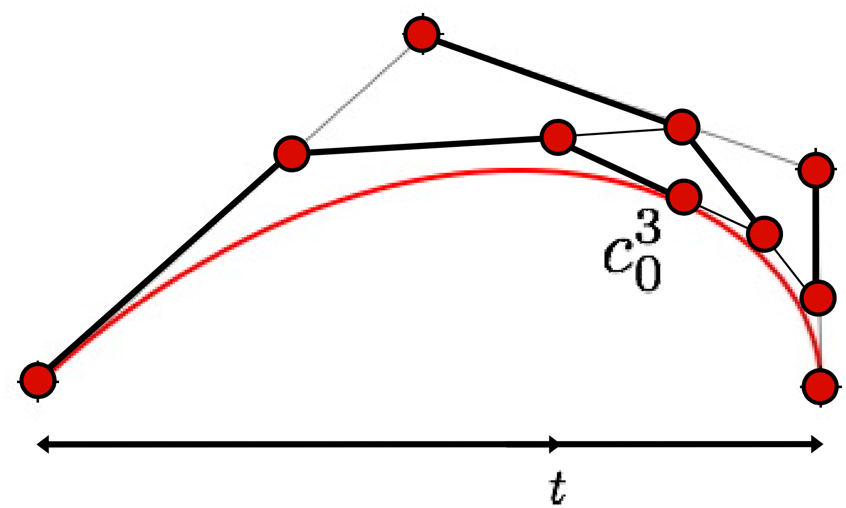


De Casteljeau Algorithm

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

$$c_i^k = (1-t)c_i^{k-1} + tc_{i+1}^{k-1} \qquad c_i^0 := c_i$$

$$\begin{tikzcd}
c_0^0 \arrow[dr, "1-t"] & & & \\
c_1^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^1 \arrow[dr, "1-t"] & & \\
c_2^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_1^1 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^2 \arrow[dr, "1-t"] & \\
c_3^0 \arrow[r, "t"] & c_2^1 \arrow[r, "t"] & c_1^2 \arrow[r, "t"] & c_0^3
\end{tikzcd}$$

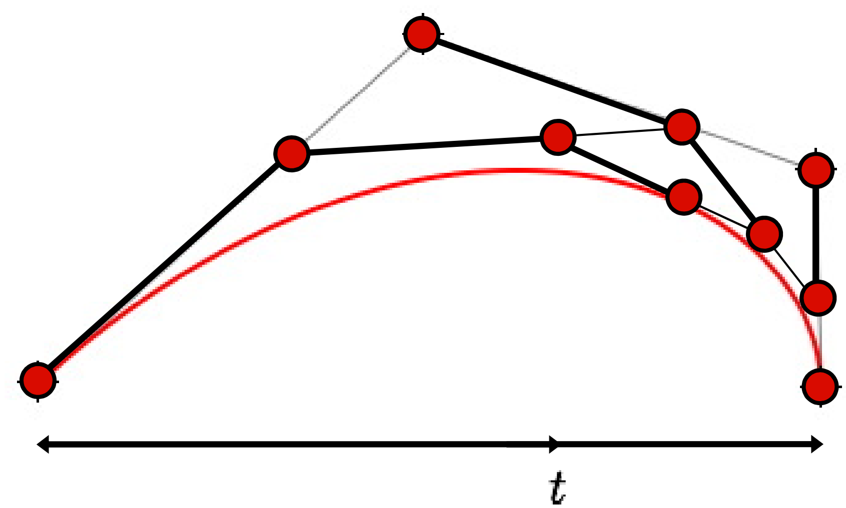


De Casteljeau Algorithm

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

$$c_i^k = (1-t)c_i^{k-1} + tc_{i+1}^{k-1} \qquad c_i^0 := c_i$$

$$\begin{tikzcd}
c_0^0 \arrow[dr, "1-t"] & & & \\
c_1^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^1 \arrow[dr, "1-t"] & & \\
c_2^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_1^1 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^2 \arrow[dr, "1-t"] & \\
c_3^0 \arrow[r, "t"] & c_2^1 \arrow[r, "t"] & c_1^2 \arrow[r, "t"] & c_0^3
\end{tikzcd}$$



- Numerically robust and efficient

Derivatives

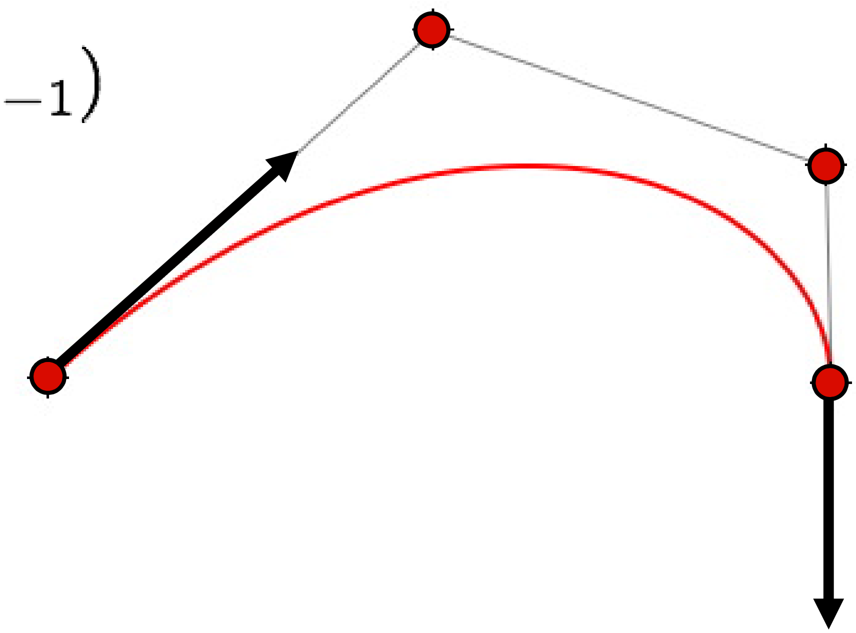
$$f'(t) = n \sum_{i=0}^{n-1} (c_{i+1} - c_i) B_i^{n-1}(t)$$

Derivatives

$$f'(t) = n \sum_{i=0}^{n-1} (c_{i+1} - c_i) B_i^{n-1}(t)$$

- Endpoints $f'(0) = n(c_1 - c_0)$

$$f'(1) = n(c_n - c_{n-1})$$

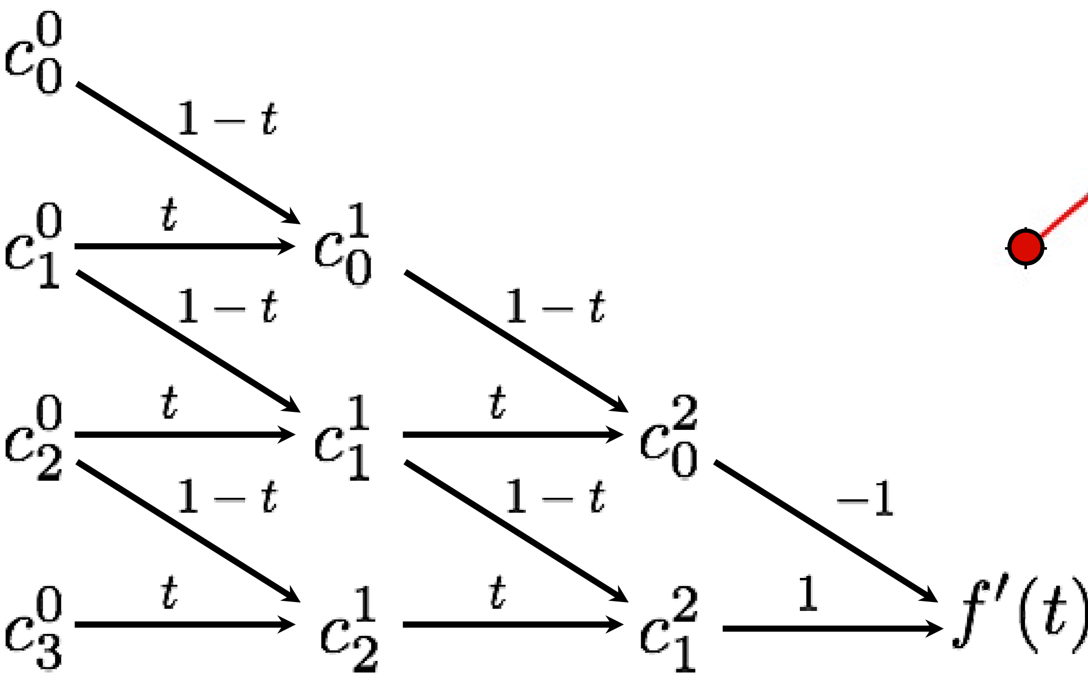


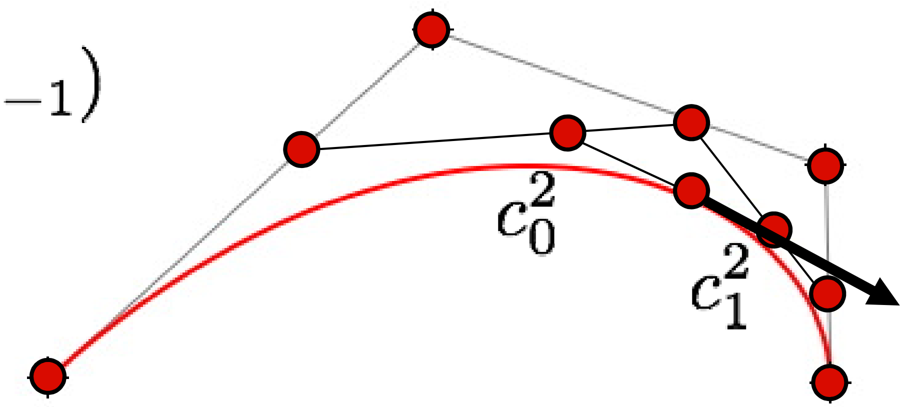
Derivatives

$$f'(t) = n \sum_{i=0}^{n-1} (c_{i+1} - c_i) B_i^{n-1}(t)$$

- Endpoints $f'(0) = n(c_1 - c_0)$

$$f'(1) = n(c_n - c_{n-1})$$





Degree Elevation

- Add control point without changing the shape
  - More degrees of freedom for editing

$$\sum_{i=0}^{n} c_i B_i^n(t) = \sum_{i=0}^{n+1} c_i' B_i^{n+1}(t)$$

Degree Elevation

- Add control point without changing the shape
  - More degrees of freedom for editing

$$\sum_{i=0}^{n} c_i B_i^n(t) \quad = \quad \sum_{i=0}^{n}((1-t)c_i + tc_i)B_i^n(t)$$

Degree Elevation

- Add control point without changing the shape
  - More degrees of freedom for editing

$$\begin{aligned}\sum_{i=0}^{n} c_i B_i^n(t) &= \sum_{i=0}^{n} ((1-t)c_i + tc_i) B_i^n(t) \\ &= \sum_{i=0}^{n} c_i \frac{n+1-i}{n+1} B_i^{n+1}(t) + \sum_{i=0}^{n} c_i \frac{i+1}{n+1} B_{i+1}^{n+1}(t)\end{aligned}$$

Degree Elevation

- Add control point without changing the shape
  - More degrees of freedom for editing

$$\begin{aligned}
\sum_{i=0}^{n} c_i B_i^n(t) &= \sum_{i=0}^{n} ((1-t)c_i + tc_i) B_i^n(t) \\
&= \sum_{i=0}^{n} c_i \frac{n+1-i}{n+1} B_i^{n+1}(t) + \sum_{i=0}^{n} c_i \frac{i+1}{n+1} B_{i+1}^{n+1}(t) \\
&= \sum_{i=0}^{n+1} c_i \frac{n+1-i}{n+1} B_i^{n+1}(t) + \sum_{i=0}^{n+1} c_{i-1} \frac{i}{n+1} B_{i+1}^{n+1}(t)
\end{aligned}$$

Degree Elevation

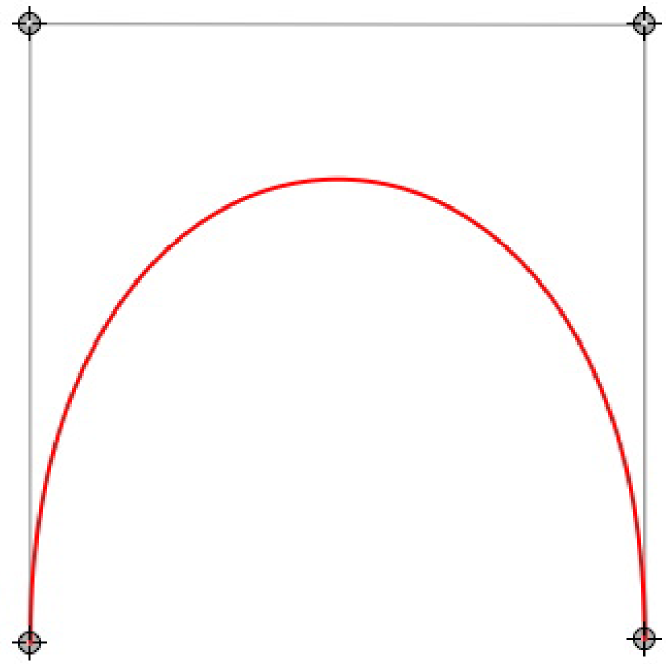
- Add control point without changing the shape
  - More degrees of freedom for editing

$$\begin{aligned}
\sum_{i=0}^{n} c_i B_i^n(t) &= \sum_{i=0}^{n} ((1-t)c_i + tc_i) B_i^n(t) \\
&= \sum_{i=0}^{n} c_i \frac{n+1-i}{n+1} B_i^{n+1}(t) + \sum_{i=0}^{n} c_i \frac{i+1}{n+1} B_{i+1}^{n+1}(t) \\
&= \sum_{i=0}^{n+1} c_i \frac{n+1-i}{n+1} B_i^{n+1}(t) + \sum_{i=0}^{n+1} c_{i-1} \frac{i}{n+1} B_{i+1}^{n+1}(t) \\
&= \sum_{i=0}^{n+1} ((1-\frac{i}{n+1})c_i + \frac{i}{n+1} c_{i-1}) B_i^{n+1}(t)
\end{aligned}$$

New control points

Degree Elevation

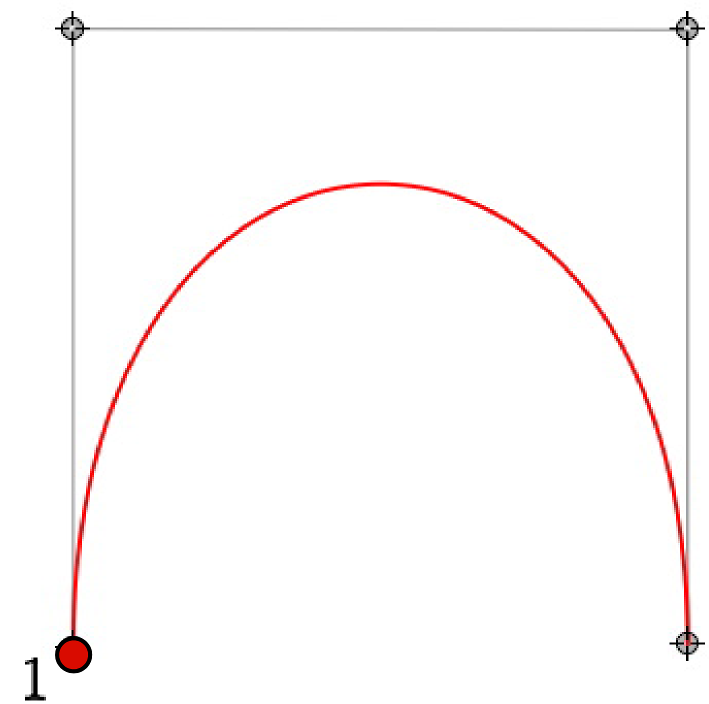
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c_i' = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

Degree Elevation

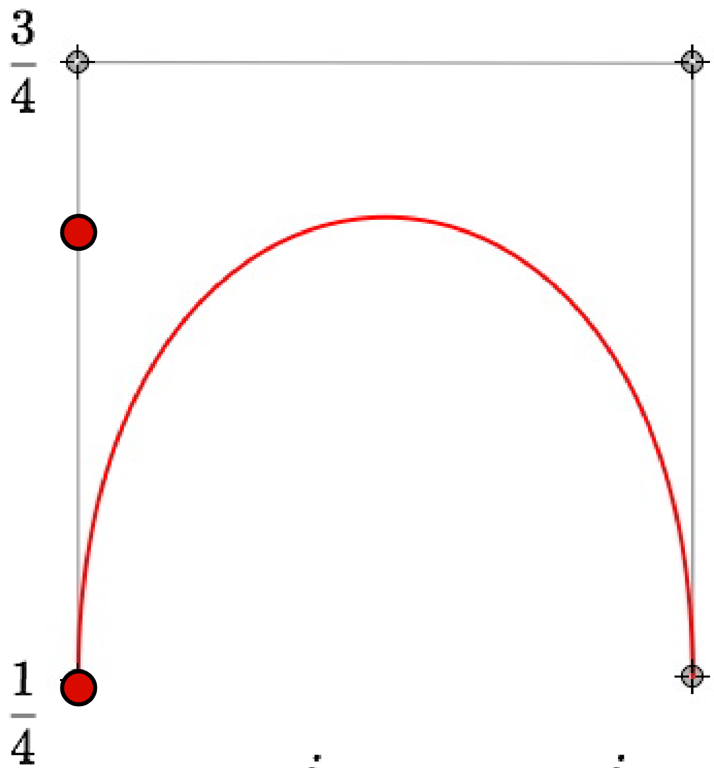
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c_i' = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

Degree Elevation

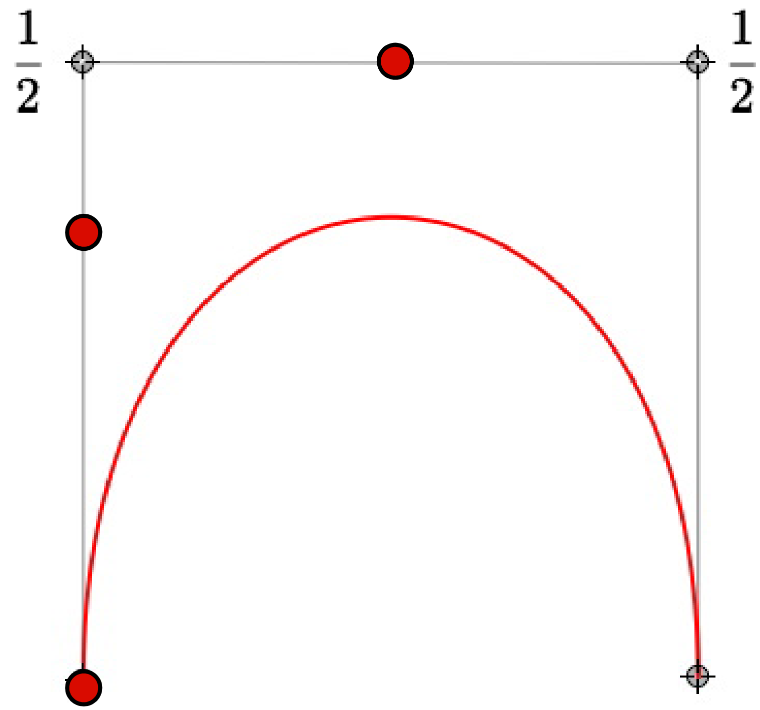
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c'_i = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

Degree Elevation

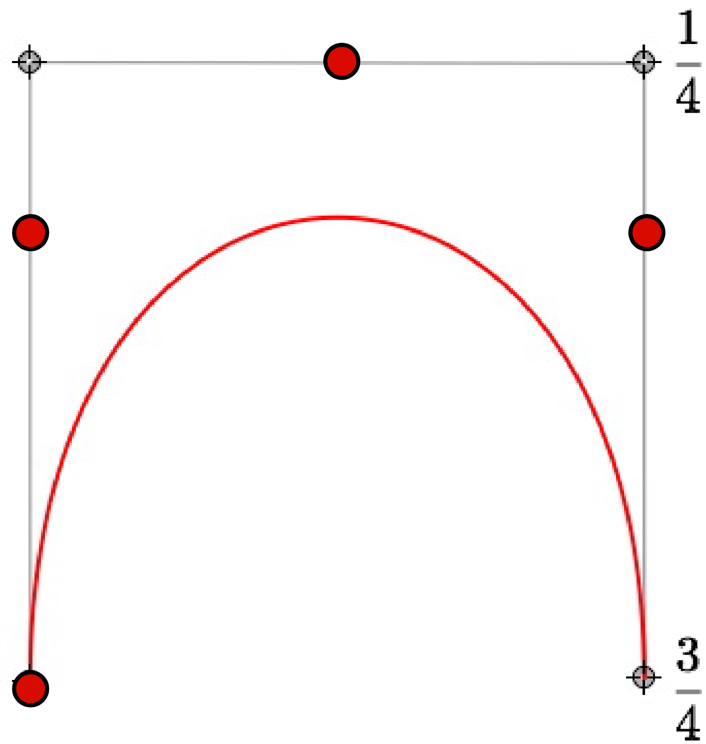
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c_i' = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

Degree Elevation

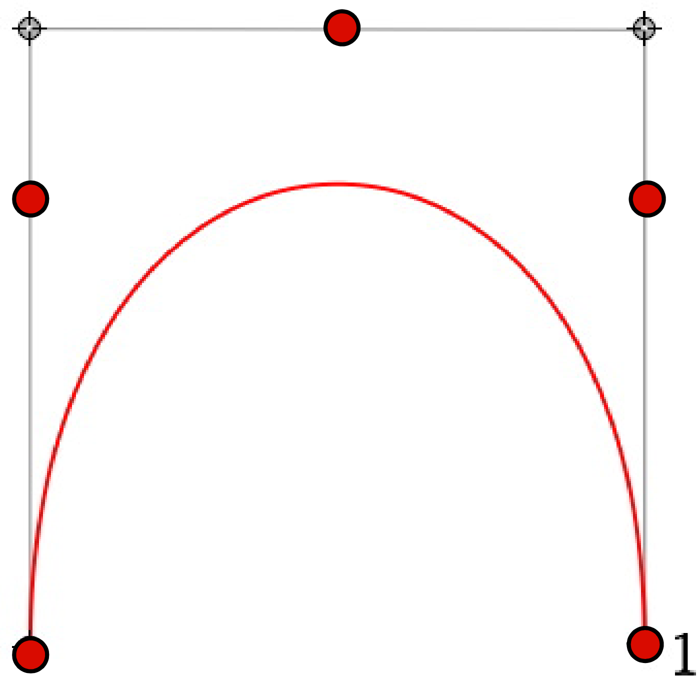
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c'_i = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

Degree Elevation

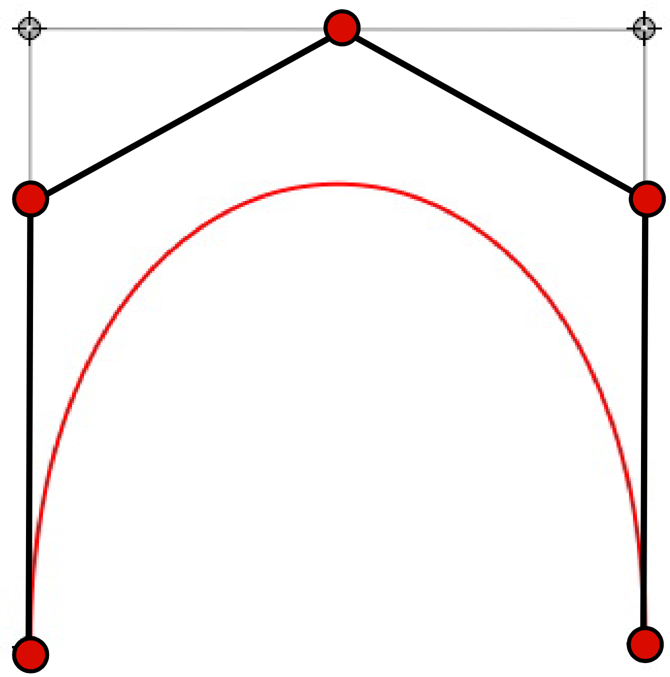
- Add control point without changing the shape
  - More degrees of freedom for editing



$$c'_i = (1 - \frac{i}{n+1})c_i + \frac{i}{n+1}c_{i-1}$$

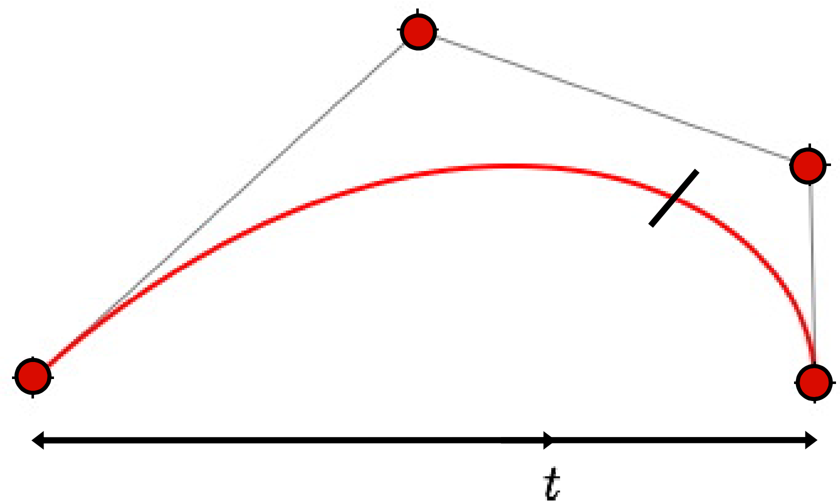
Degree Elevation

- Add control point without changing the shape
  - More degrees of freedom for editing



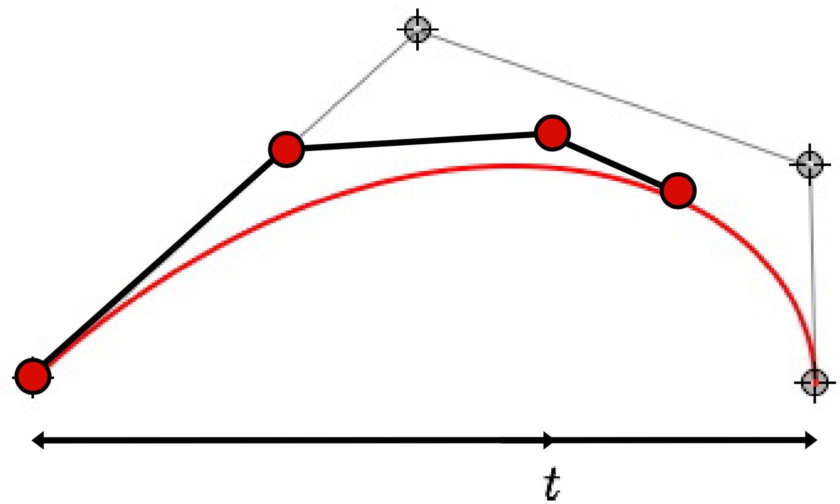
Subdivision

- Split curve at some parameter value
- Represent by two curve segments of same degree
- Control points?



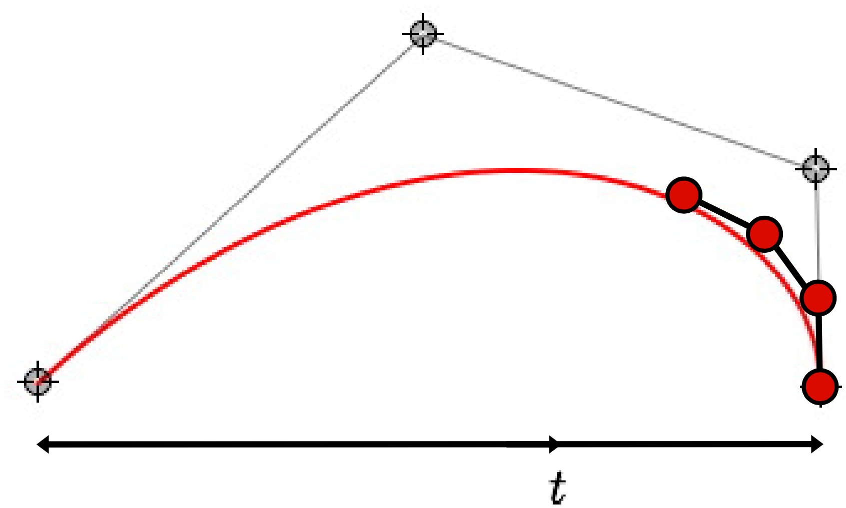
Subdivision

- Split curve at some parameter value
- Represent by two curve segments of same degree
- Control points?



Subdivision

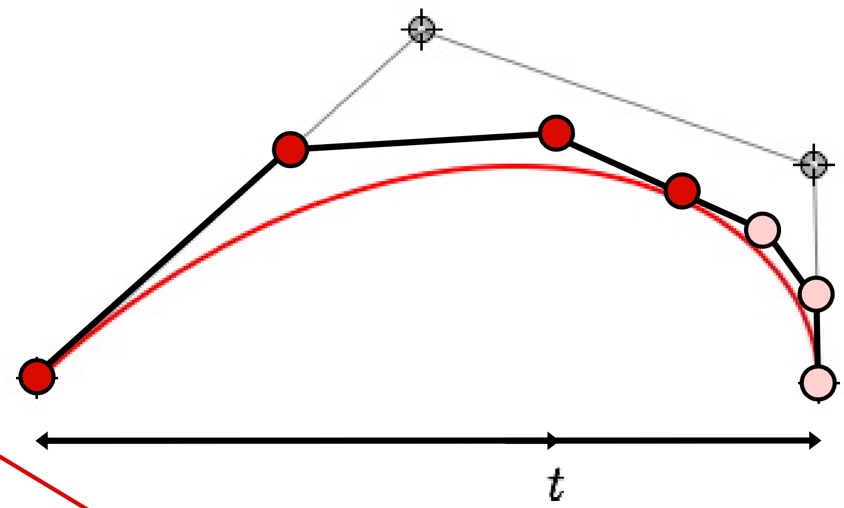
- Split curve at some parameter value
- Represent by two curve segments of same degree
- Control points?



Subdivision

- Split curve at some parameter value
- Represent by two curve segments of same degree

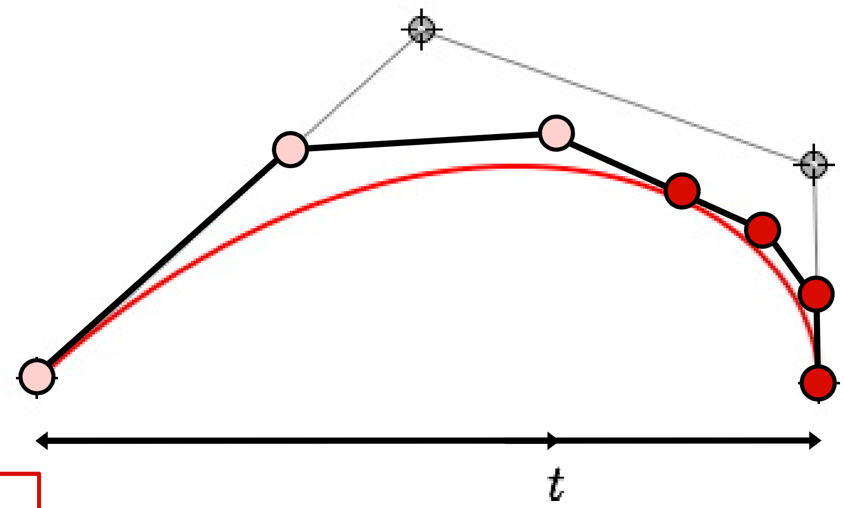
$$
\begin{tikzcd}
c_0^0 \arrow[dr, "1-t"] & & & \\
c_1^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^1 \arrow[dr, "1-t"] & & \\
c_2^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_1^1 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^2 \arrow[dr, "1-t"] & \\
c_3^0 \arrow[r, "t"] & c_2^1 \arrow[r, "t"] & c_1^2 \arrow[r, "t"] & c_0^3
\end{tikzcd}
$$



Subdivision

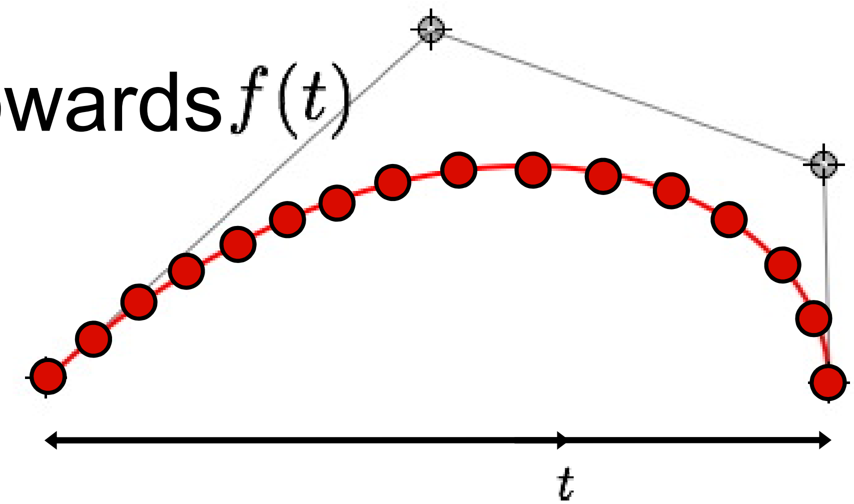
- Split curve at some parameter value
- Represent by two curve segments of same degree

$$\begin{tikzcd}
c_0^0 \arrow[dr, "1-t"] & & & \\
c_1^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^1 \arrow[dr, "1-t"] & & \\
c_2^0 \arrow[r, "t"] \arrow[dr, "1-t"] & c_1^1 \arrow[r, "t"] \arrow[dr, "1-t"] & c_0^2 \arrow[dr, "1-t"] & \\
c_3^0 \arrow[r, "t"] & c_2^1 \arrow[r, "t"] & c_1^2 \arrow[r, "t"] & c_0^3
\end{tikzcd}$$



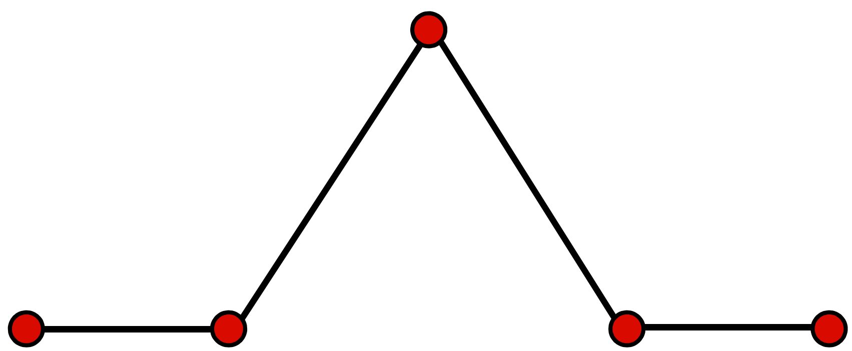
Subdivision

- Split curve at some parameter value
- Represent by two curve segments of same degree
- 2^k control polygons after k subdivisions
- Converges quadratically towards $f(t)$
  - Efficient rendering



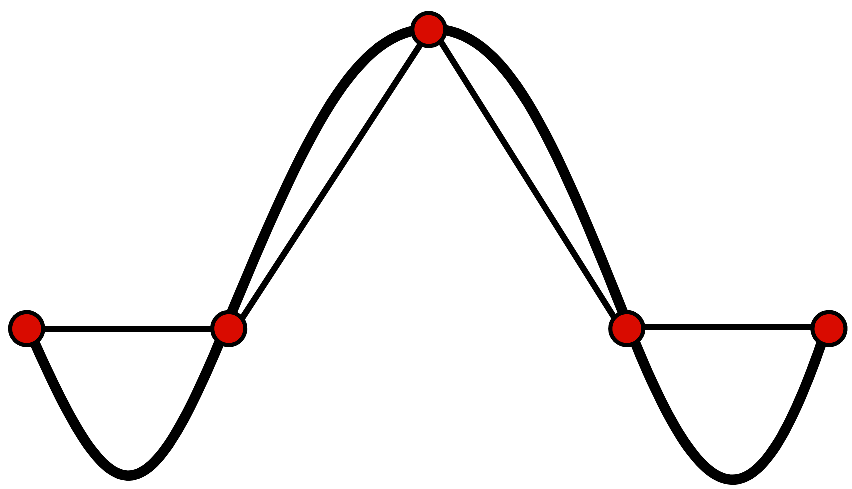
Variation Diminishing

- Curve “wiggles” no more than control polygon



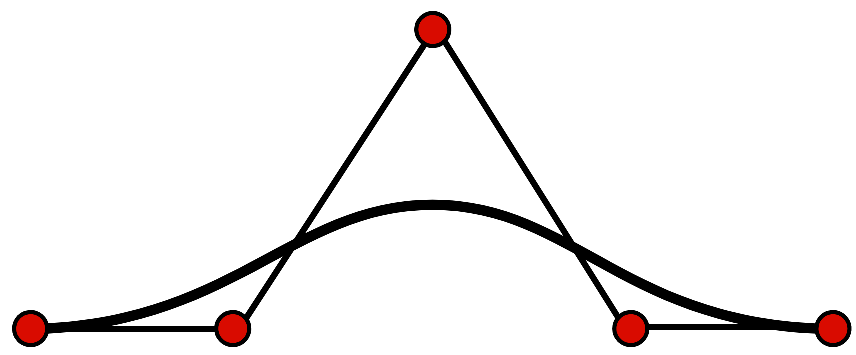
Variation Diminishing

- Curve "wiggles" no more than control polygon



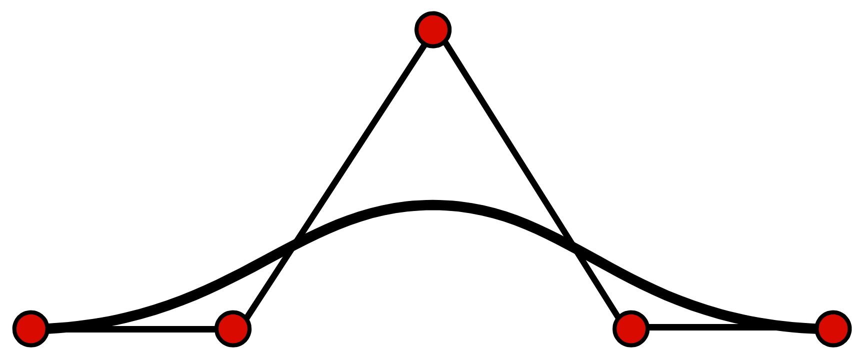
Variation Diminishing

- Curve "wiggles" no more than control polygon



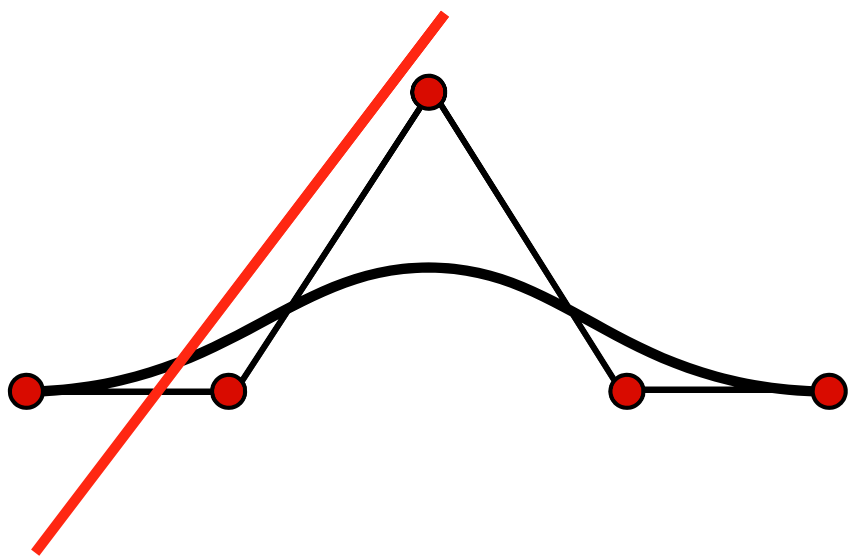
Variation Diminishing

- Curve "wiggles" no more than control polygon
- For any line, number of intersections with control polygon $\geq$ intersection with curve



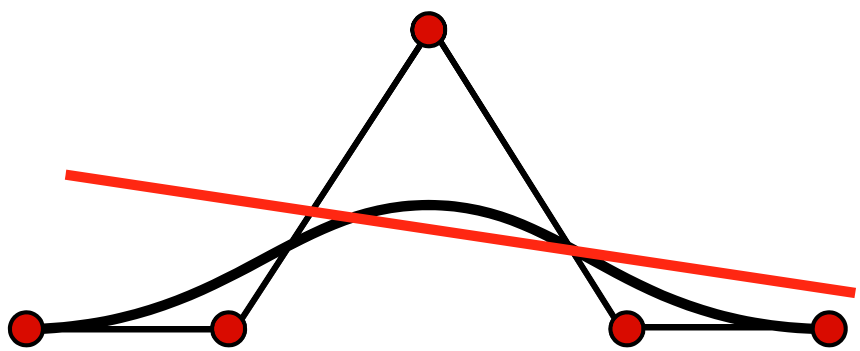
Variation Diminishing

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



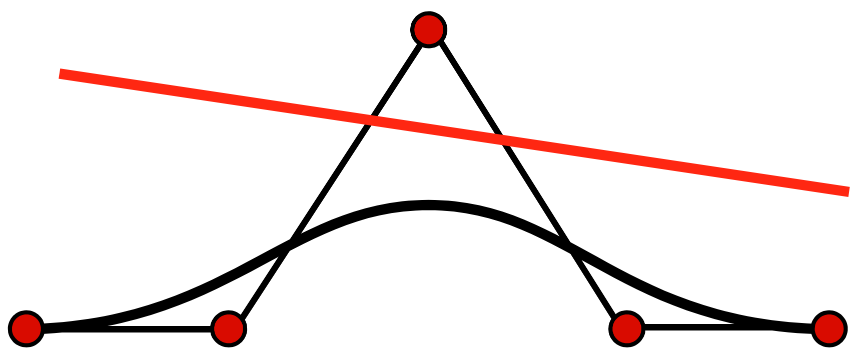
Variation Diminishing

- Curve "wiggles" no more than control polygon
- For any line, number of intersections with control polygon ≥ intersection with curve



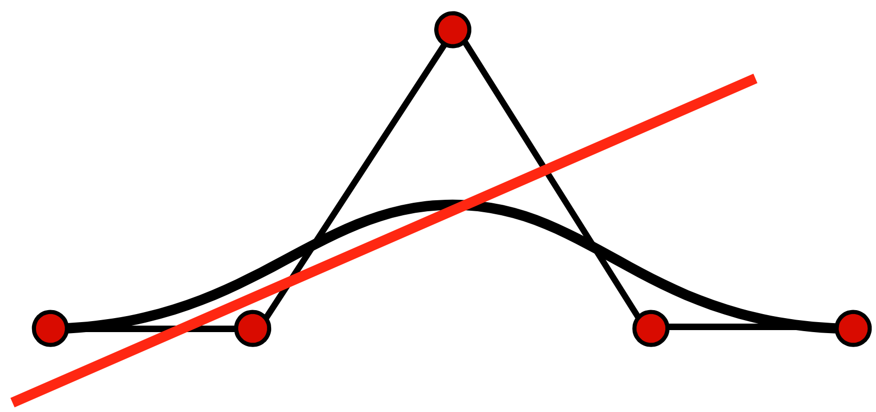
Variation Diminishing

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



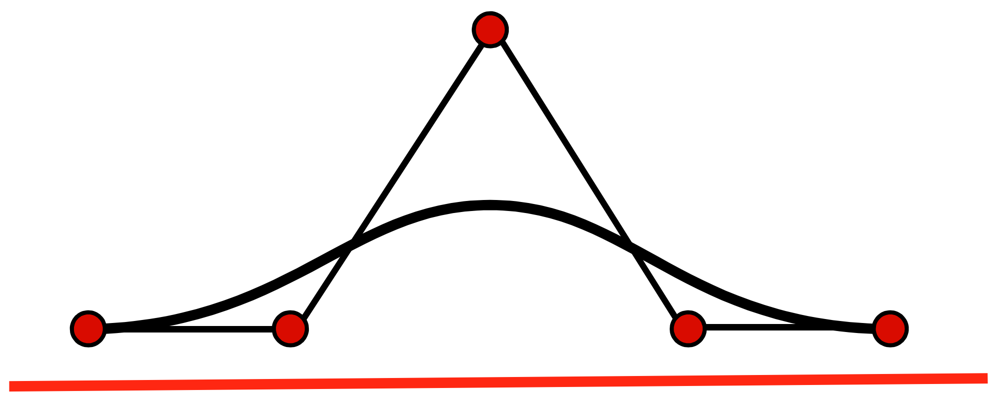
Variation Diminishing

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



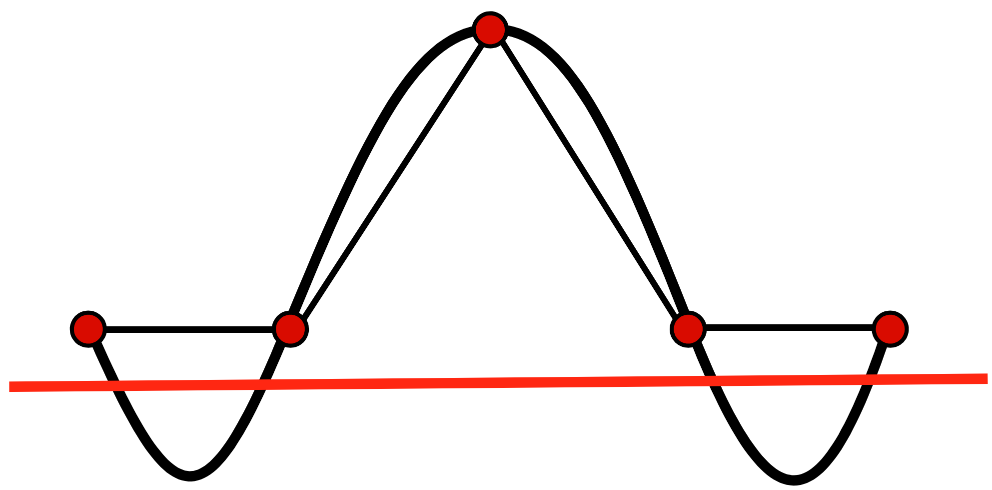
Variation Diminishing

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



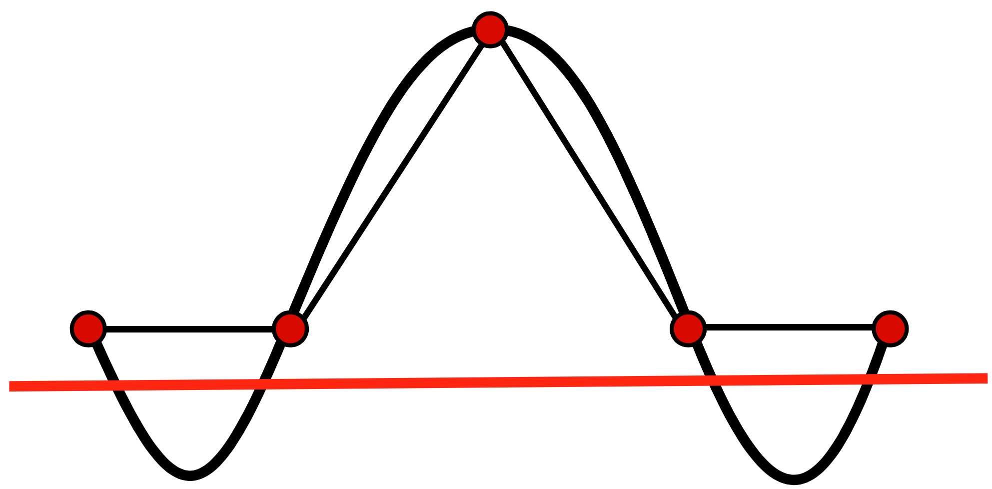
Variation Diminishing

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



Variation Diminishing

- Curve “wiggles” no more than control polygon
- For any line, number of intersections with control polygon $\geq$ intersection with curve
- Application: intersection computation



References

- Farin: Curves and Surfaces for CAGD, Morgan Kaufmann, 2002
- Demo applets:
  http://i33www.ira.uka.de/applets/mocca/html/noplugin/inhalt.html

Homework

- Practical part:
  - Allow user to input control points
  - Calculate the corresponding Bezier curve by using the De Casteljeau algorithm and display it
  - Bonus options (see the definition on the website)
- Theoretical part:
  - Prove the properties of Bezier curves we mentioned in class; think about modeling options

Homework handout is on the course website

Thank you!