

计算机辅助几何设计

2021秋学期

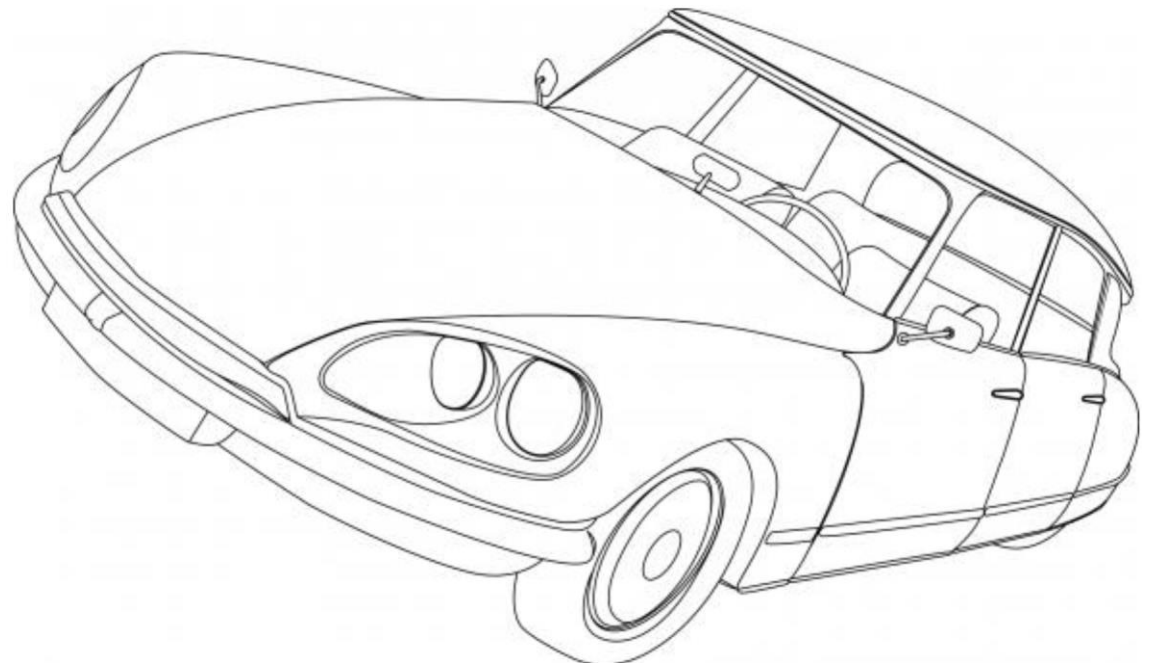
Bézier Curves

陈仁杰

中国科学技术大学

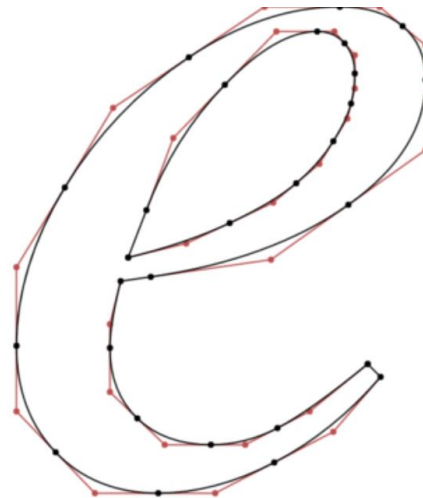
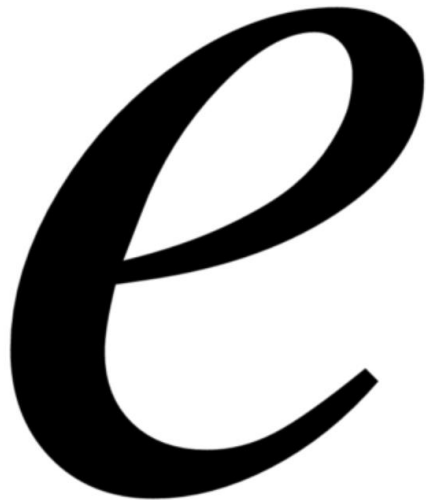
Bézier curves

- Bézier curves/splines developed by
 - Paul de Casteljau at Citroen (1959)
 - Pierre Bézier at Renault (1963)for free-form parts in automotive design



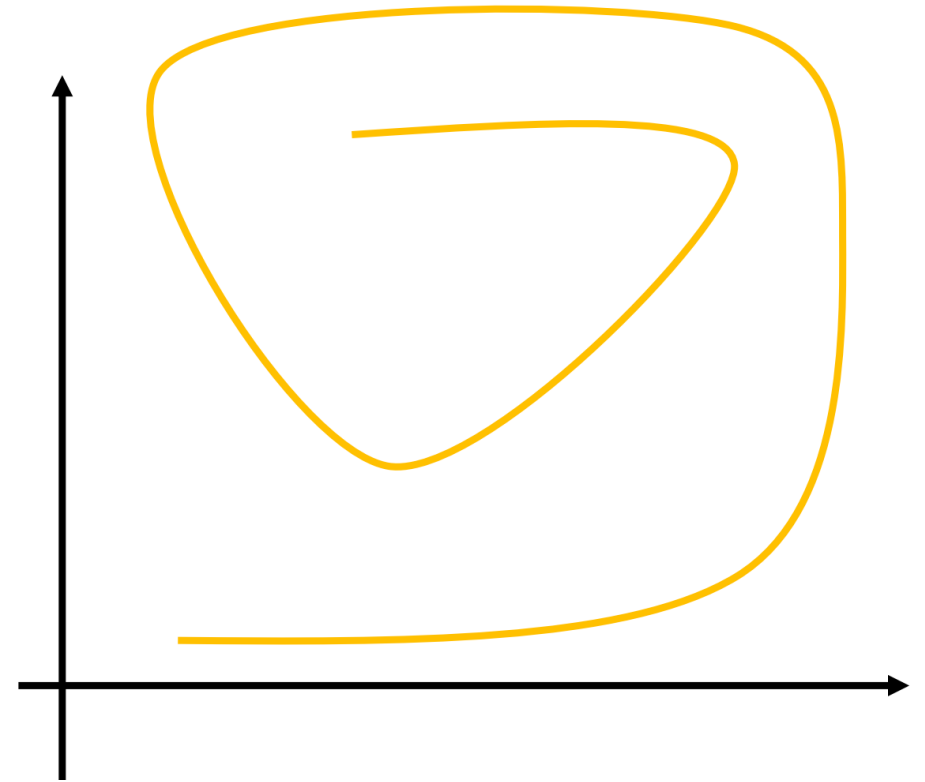
Bézier curves

- Today: Standard tool for 2D curve editing
- Cubic 2D Bézier curves are everywhere:
 - Inkscape, Corel Draw, Adobe Illustrator, Powerpoint, ...
 - PDF, Truetype (quadratic curves), Windows GDI, ...
- Widely used in 3D curve & surface modeling as well



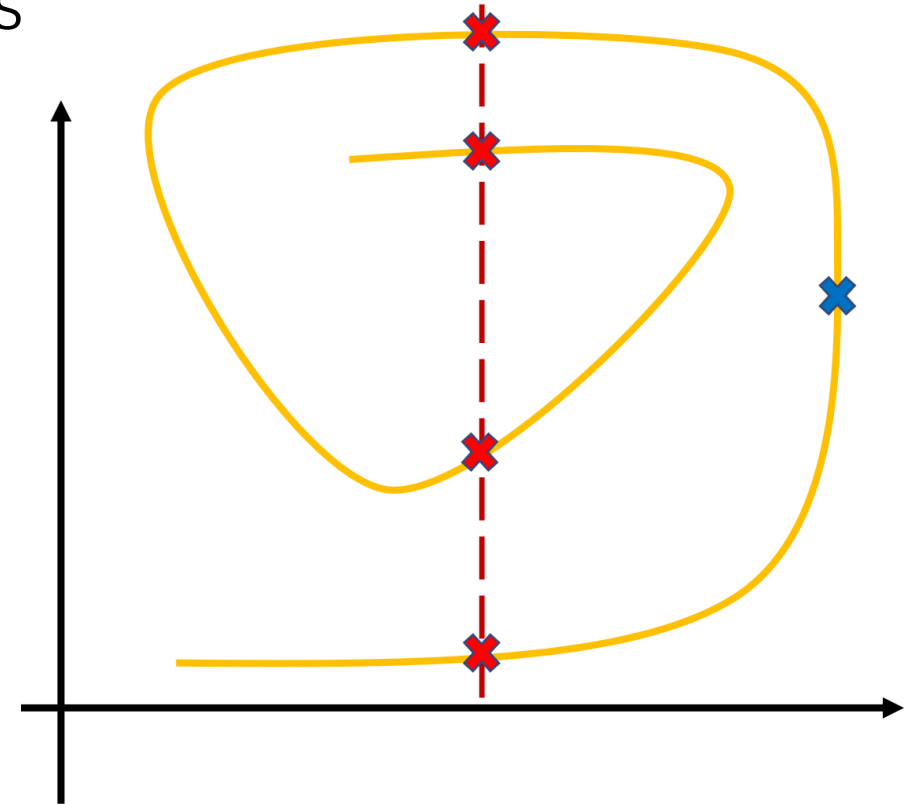
Curve representation

- The implicit curve form $f(x, y) = 0$ suffers from several limitations:



Curve representation

- The implicit curve form $f(x, y) = 0$ suffers from several limitations:
 - Multiple values for the same x -coordinates
 - Undefined derivative $\frac{dy}{dx}$ (see blue cross)
 - Not invariant w.r.t axes transformations

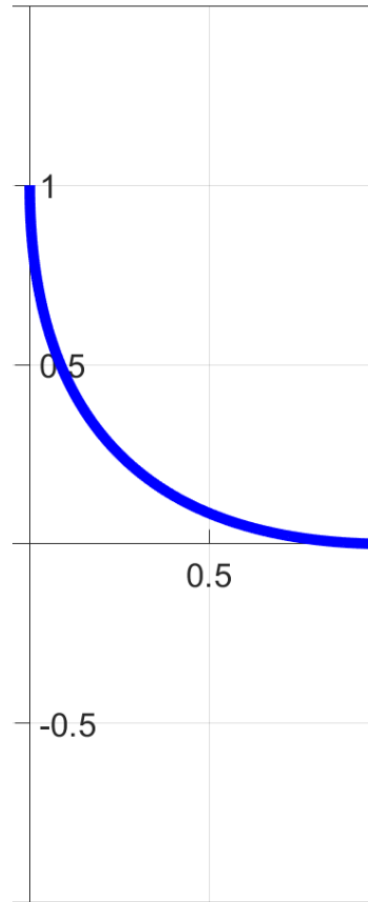


Parametric representation

- Remedy: parametric representation $c(t) = (x(t), y(t))$
 - Easy evaluations
 - The parameter t can be interpreted as time
 - The curve can be interpreted as the path traced by a moving particle

Modeling with the power basis, ...

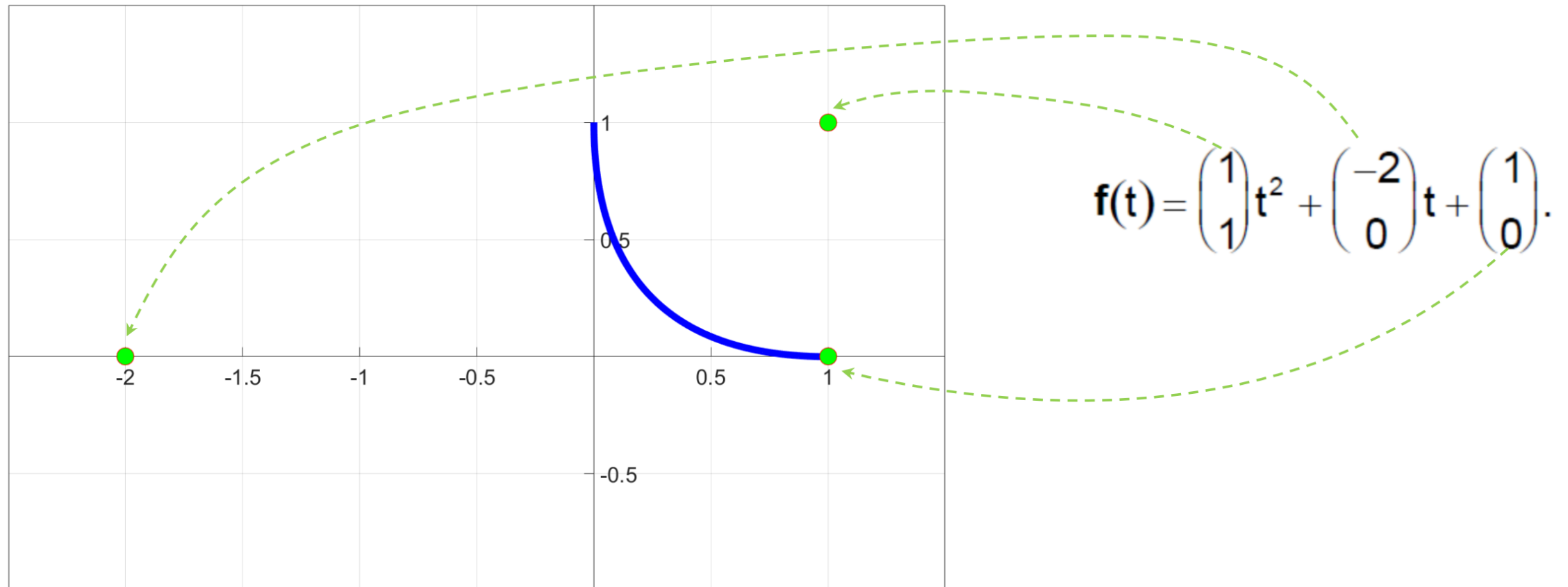
- Example of a parabola: $f(t) = at^2 + bt + c$



$$f(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} t^2 + \begin{pmatrix} -2 \\ 0 \end{pmatrix} t + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

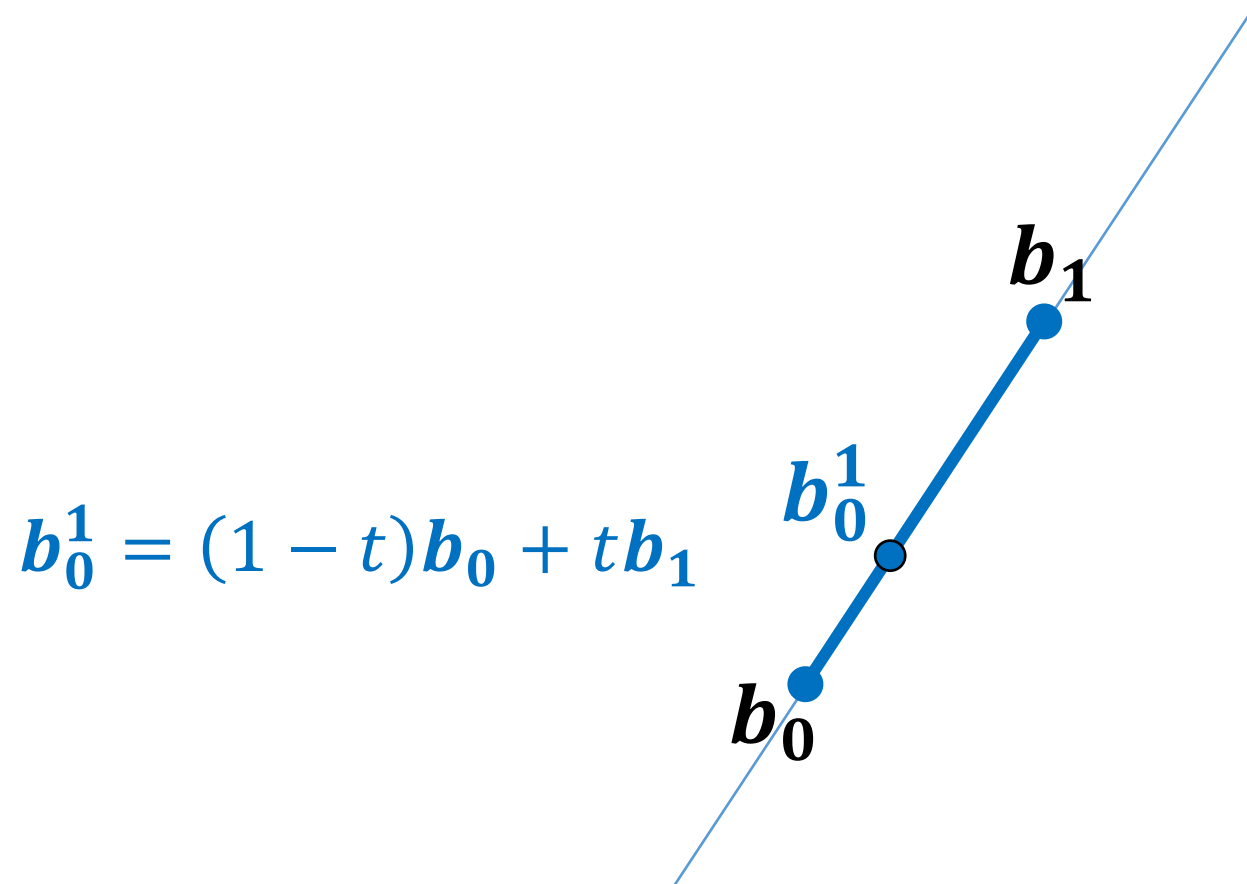
Modeling with the power basis, ... no thanks!

- Examples of a parabola: $f(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}$: the coefficients of the power basis lack intuitive geometric meaning



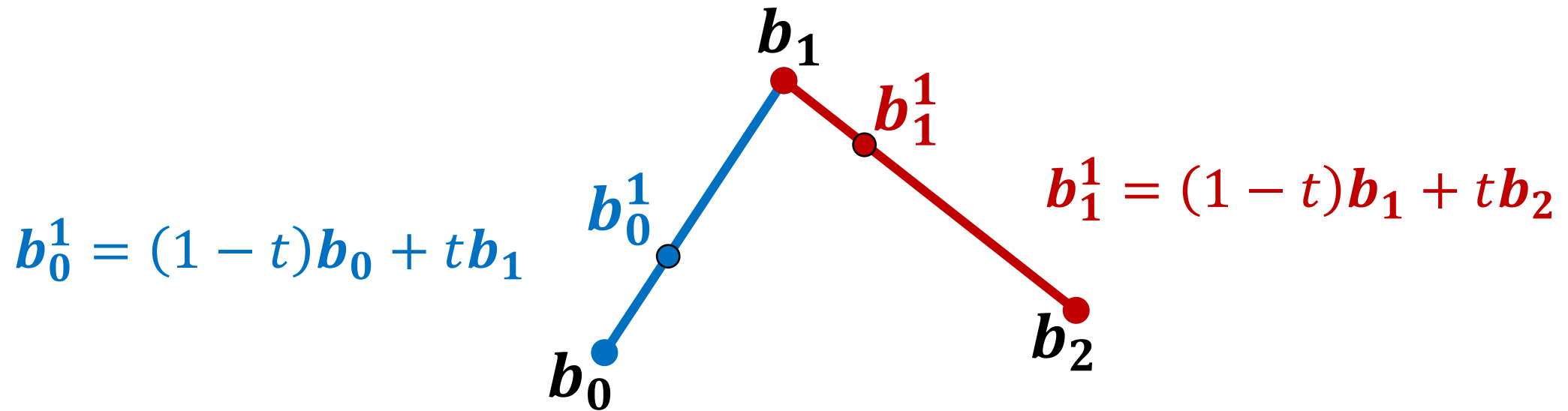
Back to the drawing board

- A point on a parametric line



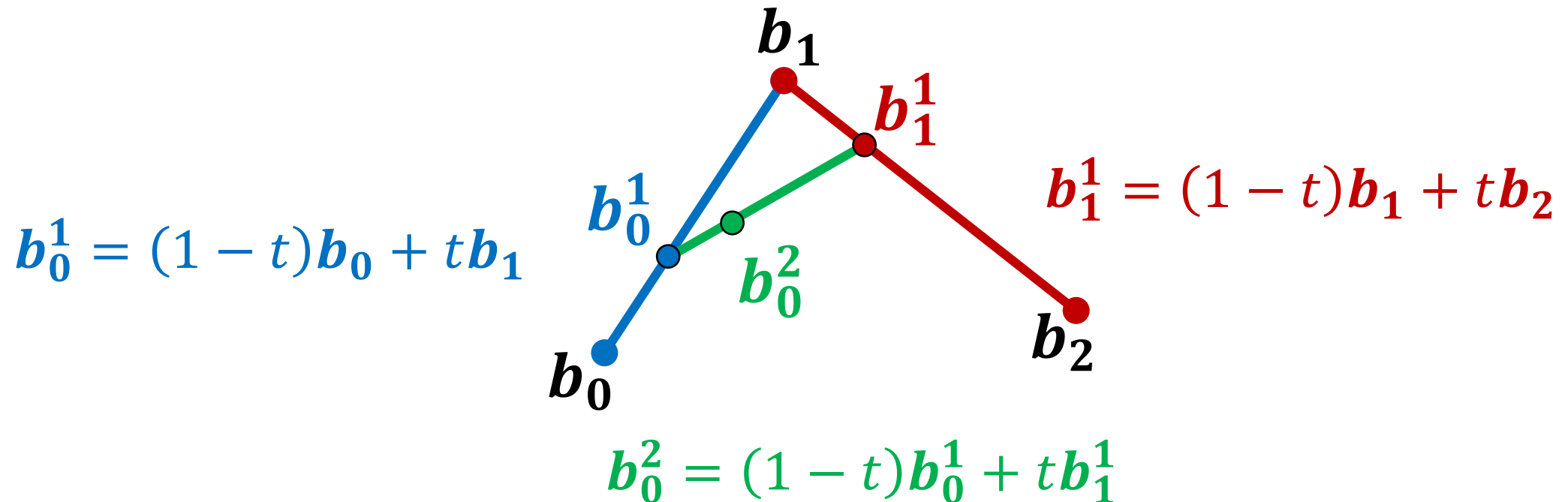
Back to the drawing board

- Another point on a second parametric line



Back to the drawing board

- A third point on the line defined by the first two points



Back to the drawing board

- And then simplify...

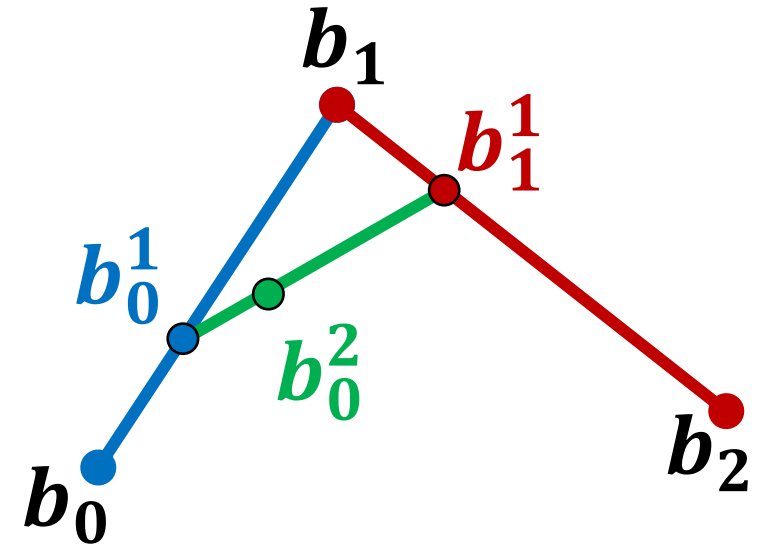
$$b_0^1 = (1 - t)b_0 + tb_1$$

$$b_0^2 = (1 - t)b_0^1 + tb_1^1$$

$$b_1^1 = (1 - t)b_1 + tb_2$$

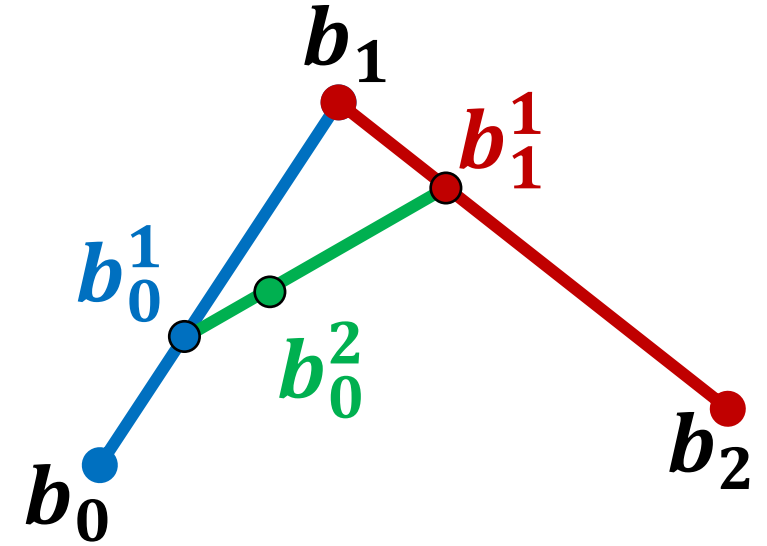
$$b_0^2 = (1 - t)[(1 - t)b_0 + tb_1] + t[(1 - t)b_1 + tb_2]$$

$$b_0^2 = (1 - t)^2 b_0 + 2t(1 - t)b_1 + t^2 b_2$$



Back to the drawing board

- We obtained another description of parabolic curves
- The coefficients b_0, b_1, b_2 have a geometric meaning



$$b_0^2 = (1 - t)^2 b_0 + 2t(1 - t)b_1 + t^2 b_2$$

Example re-visited

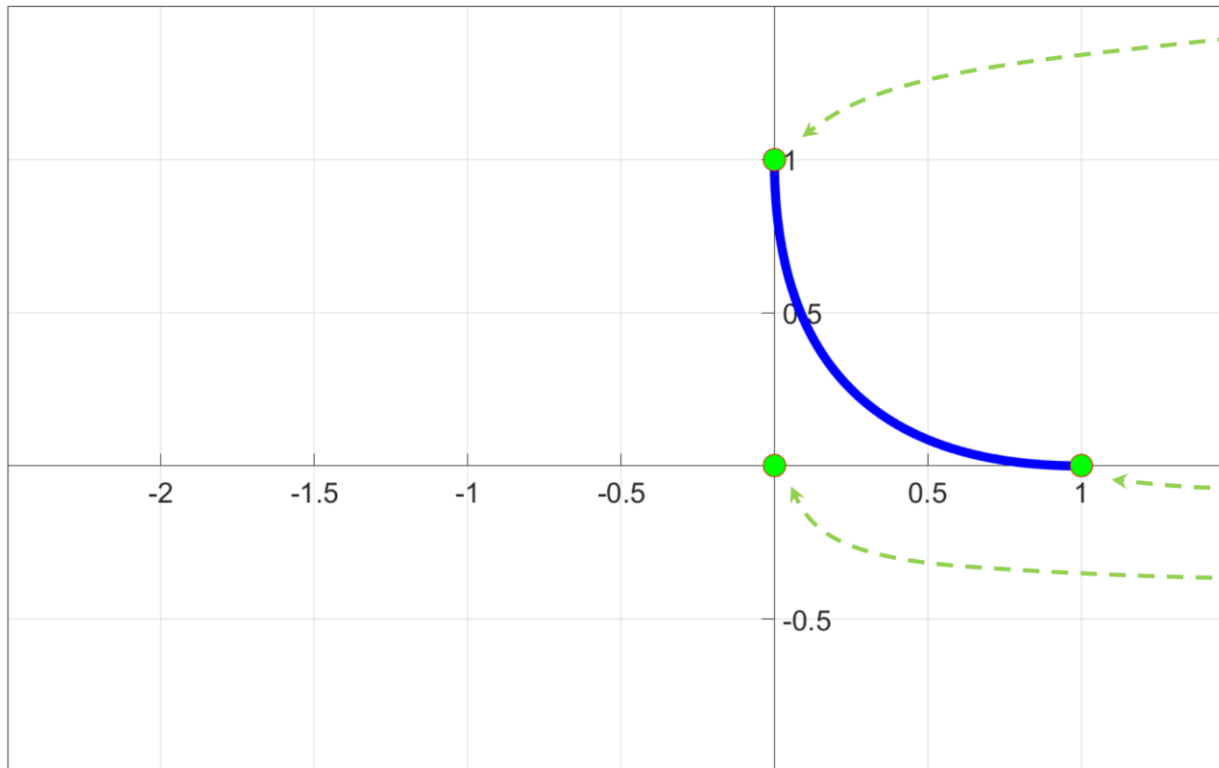
- Let's rewrite our initial parabolic curve example in the new basis

$$\mathbf{f}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} t^2 + \begin{pmatrix} -2 \\ 0 \end{pmatrix} t + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\mathbf{f}(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1-t)^2 + \begin{pmatrix} 0 \\ 0 \end{pmatrix} 2t(1-t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} t^2$$

Example re-visited

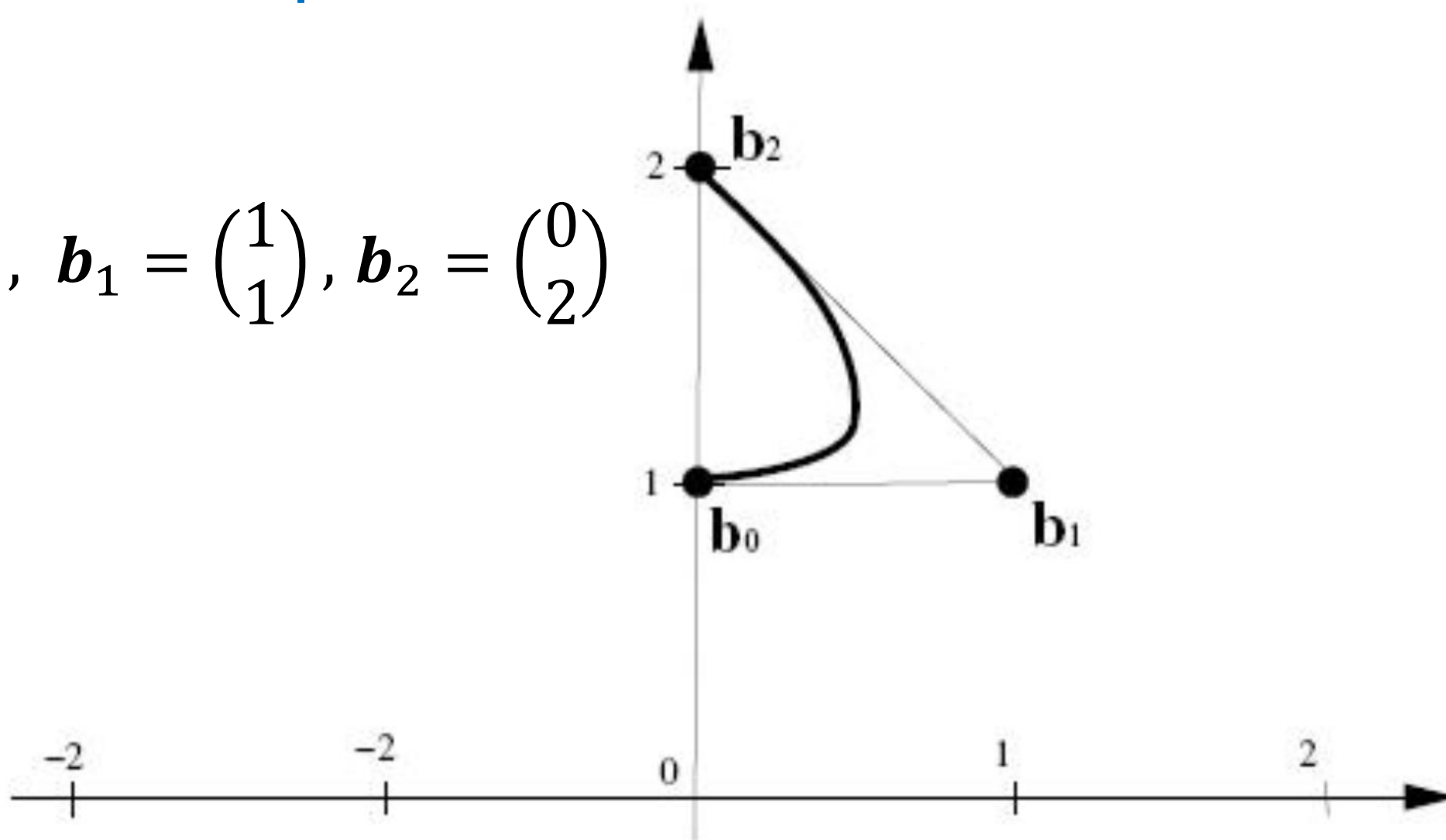
- The coefficients have a geometric meaning
- More intuitive for curve manipulation



$$\mathbf{f}(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1-t)^2 + \begin{pmatrix} 0 \\ 0 \end{pmatrix} 2t(1-t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} t^2$$

Another example

$$\mathbf{b}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{b}_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$



Going further

- Cubic approximation

- Given 4 points: $\mathbf{p}_0^0(t) = \mathbf{p}_0, \mathbf{p}_1^0(t) = \mathbf{p}_1, \mathbf{p}_2^0(t) = \mathbf{p}_2, \mathbf{p}_3^0(t) = \mathbf{p}_3$

- First iteration $\mathbf{p}_0^1 = (1 - t)\mathbf{p}_0 + t\mathbf{p}_1$

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

$$\mathbf{p}_2^1 = (1 - t)\mathbf{p}_2 + t\mathbf{p}_3$$

- 2nd iteration $\mathbf{p}_0^2 = (1 - t)^2\mathbf{p}_0 + 2t(1 - t)\mathbf{p}_1 + t^2\mathbf{p}_2$

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

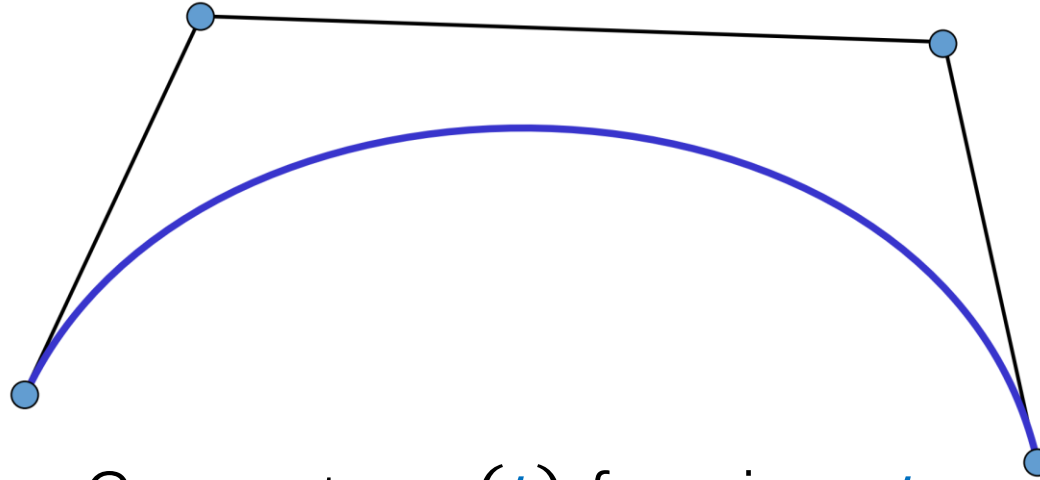
- Curve

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

Throughout these examples, we just re-invented a primitive version of the de Casteljau algorithm

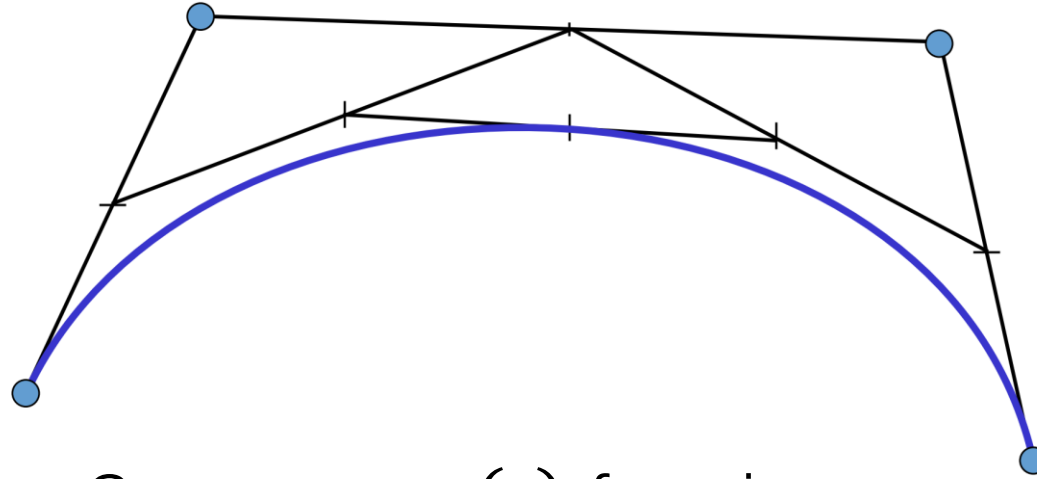
Now let's examine it more closely ...

De Casteljau algorithm



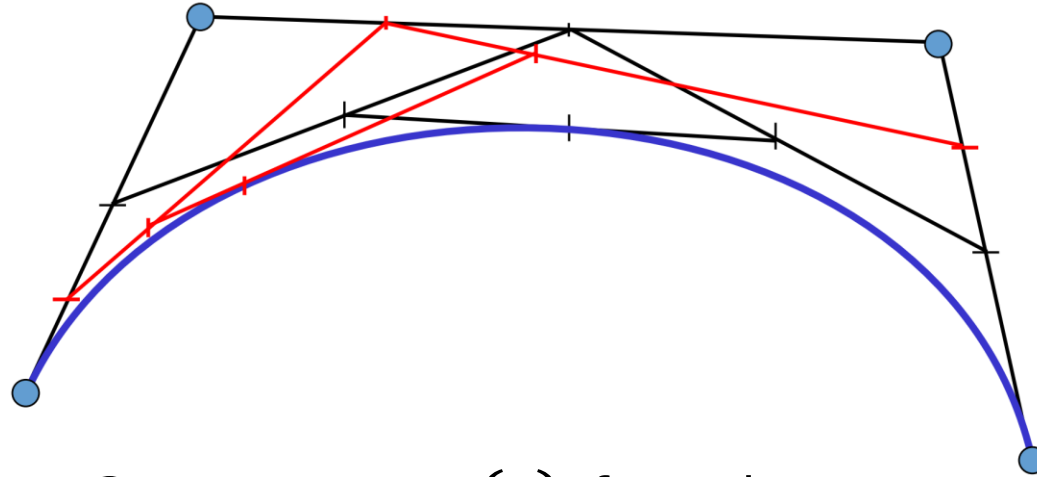
- De Casteljau Algorithm: Computes $x(t)$ for given t
 - Bisect control polygon in ratio $t: (1 - t)$
 - Connect the new dots with lines (adjacent segments)
 - Interpolate again with the same ratio
 - Iterate, until only one point is left

De Casteljau algorithm



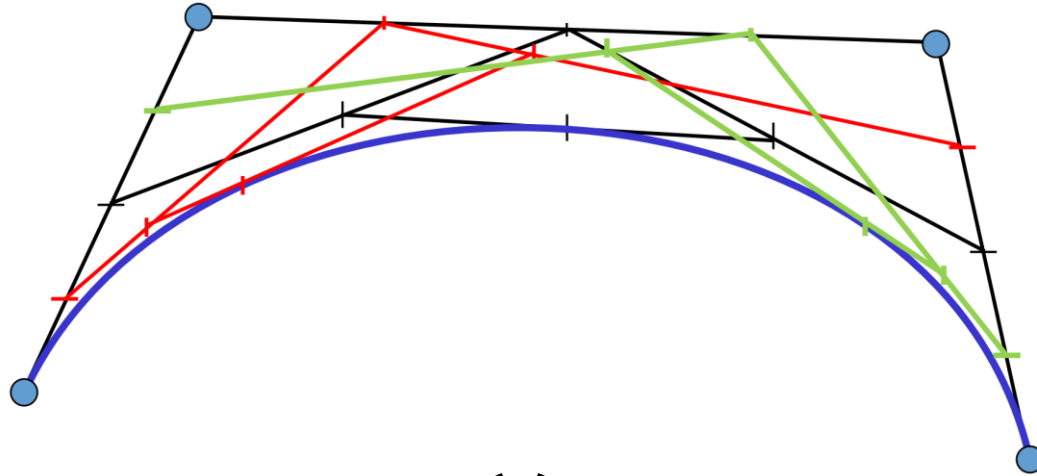
- De Casteljau Algorithm: Computes $x(t)$ for given t
 - Bisect control polygon in ratio t : $(1 - t)$
 - Connect the new dots with lines (adjacent segments)
 - Interpolate again with the same ratio
 - Iterate, until only one points is left

De Casteljau algorithm



- De Casteljau Algorithm: Computes $x(t)$ 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



- De Casteljau Algorithm: Computes $x(t)$ 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 points is left

De Casteljau algorithm

- Algorithm description

- Input: points $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{R}^3$
- Output: curve $\mathbf{x}(t), t \in [0,1]$

- Geometric construction of the points $\mathbf{x}(t)$ for given t :

$$\mathbf{b}_i^0(t) = \mathbf{b}_i, \quad i = 0, \dots, n$$

$$\mathbf{b}_i^r(t) = (1 - t)\mathbf{b}_i^{r-1}(t) + t \mathbf{b}_{i+1}^{r-1}(t)$$

$$r = 1, \dots, n \quad i = 0, \dots, n - r$$

- Then $\mathbf{b}_0^n(t)$ is the searched curve point $\mathbf{x}(t)$ at the parameter value t

De Casteljau algorithm

- Repeated convex combination of control points

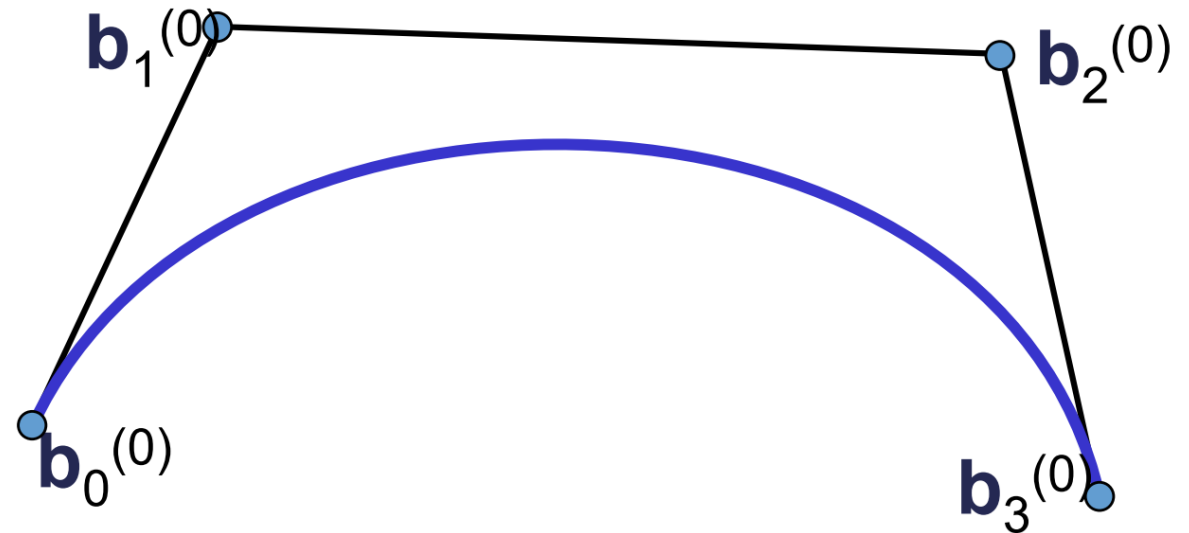
$$\mathbf{b}_i^{(r)} = (1 - t)\mathbf{b}_i^{(r-1)} + t\mathbf{b}_{i+1}^{(r-1)}$$

$\mathbf{b}_0^{(0)}$

$\mathbf{b}_1^{(0)}$

$\mathbf{b}_2^{(0)}$

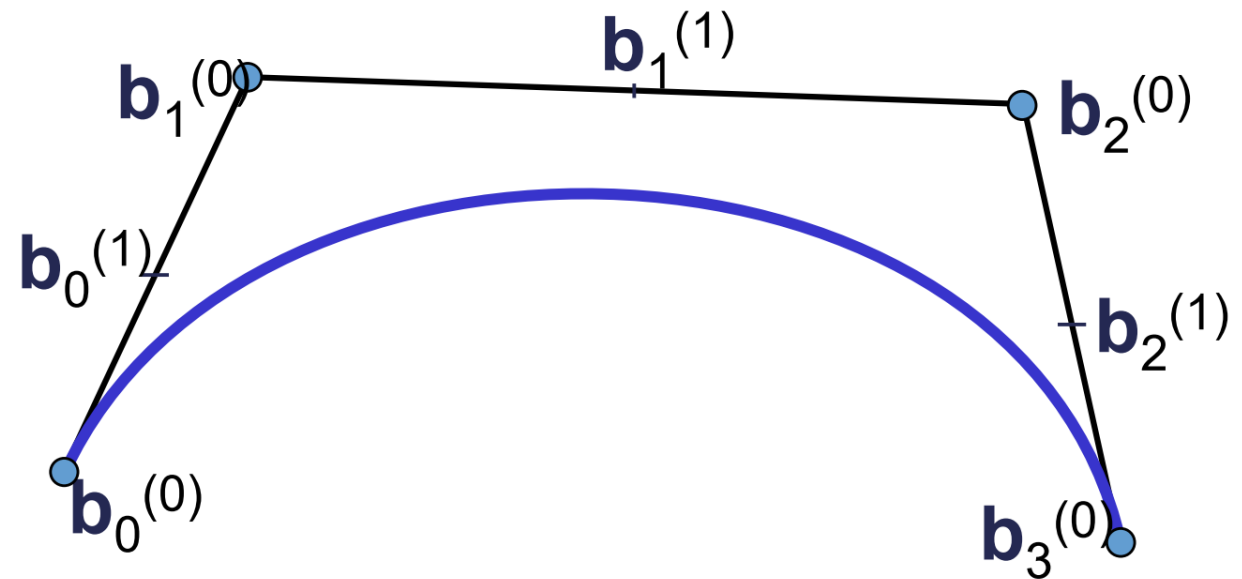
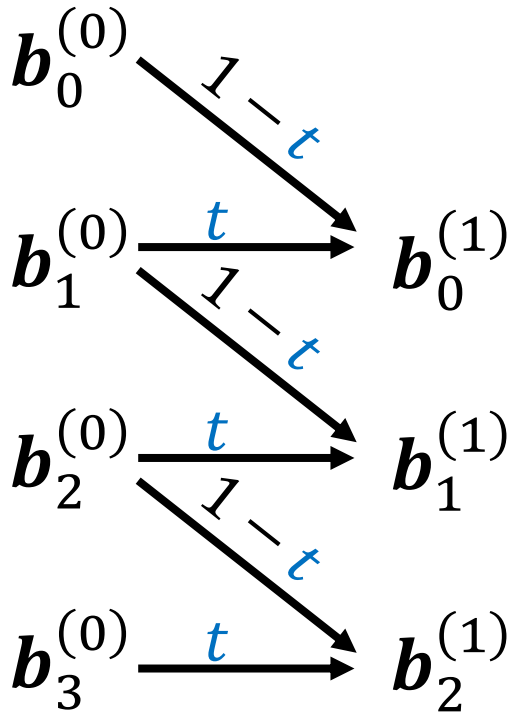
$\mathbf{b}_3^{(0)}$



De Casteljau algorithm

- Repeated convex combination of control points

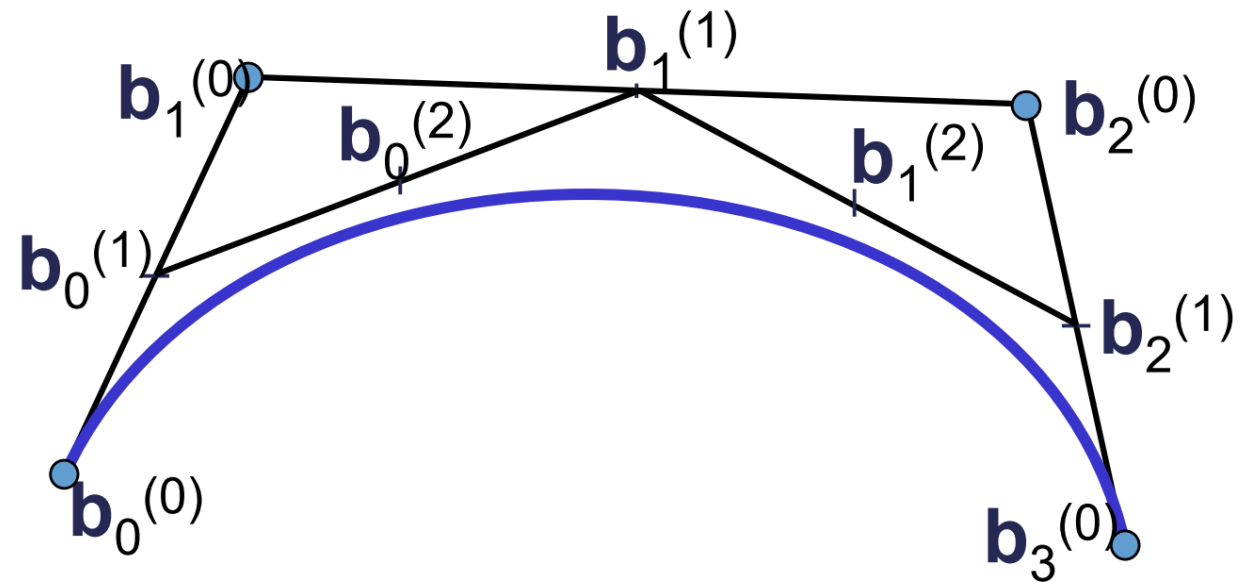
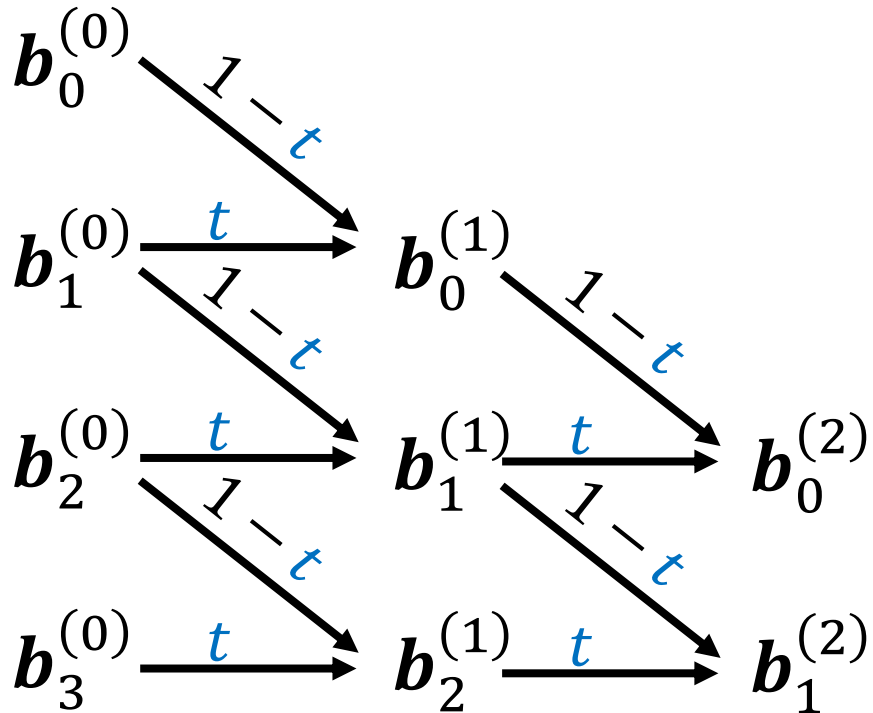
$$\mathbf{b}_i^{(r)} = (1 - t)\mathbf{b}_i^{(r-1)} + t\mathbf{b}_{i+1}^{(r-1)}$$



De Casteljau algorithm

- Repeated convex combination of control points

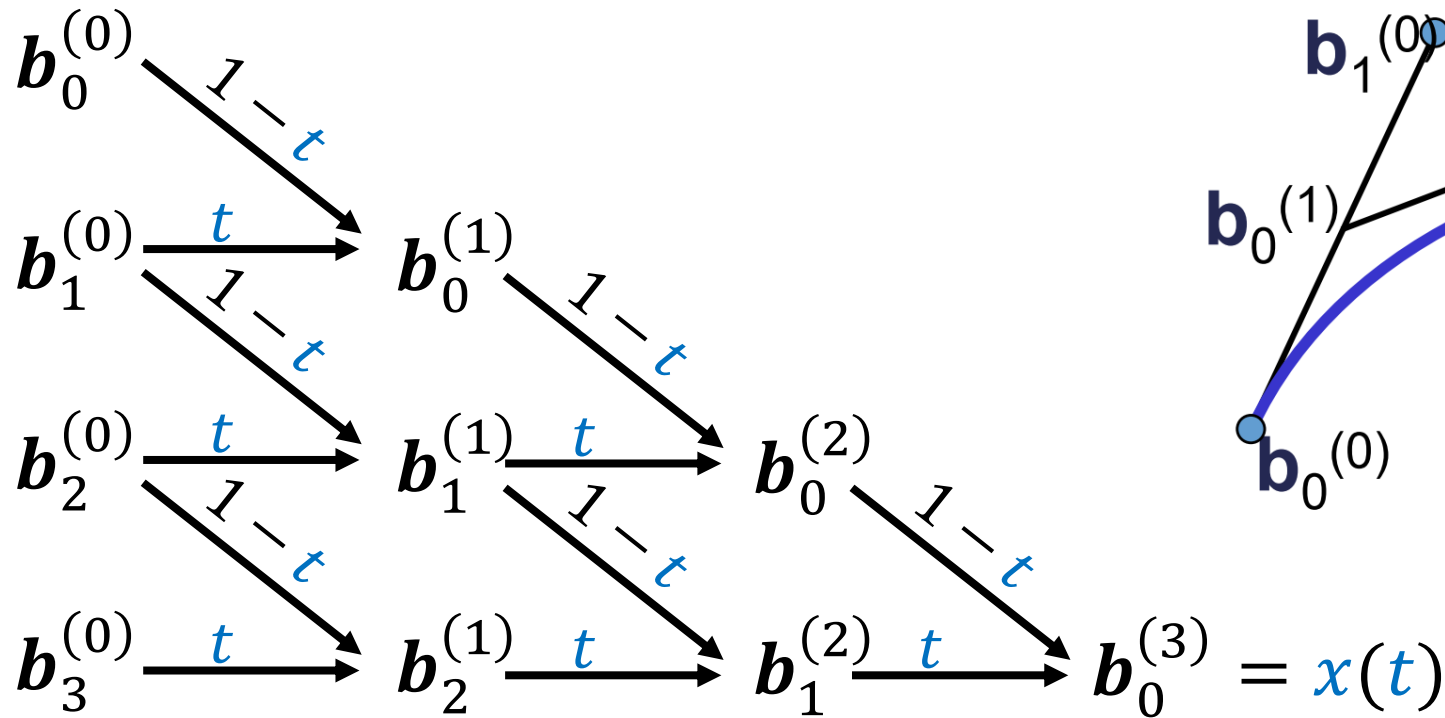
$$\mathbf{b}_i^{(r)} = (1 - t)\mathbf{b}_i^{(r-1)} + t\mathbf{b}_{i+1}^{(r-1)}$$



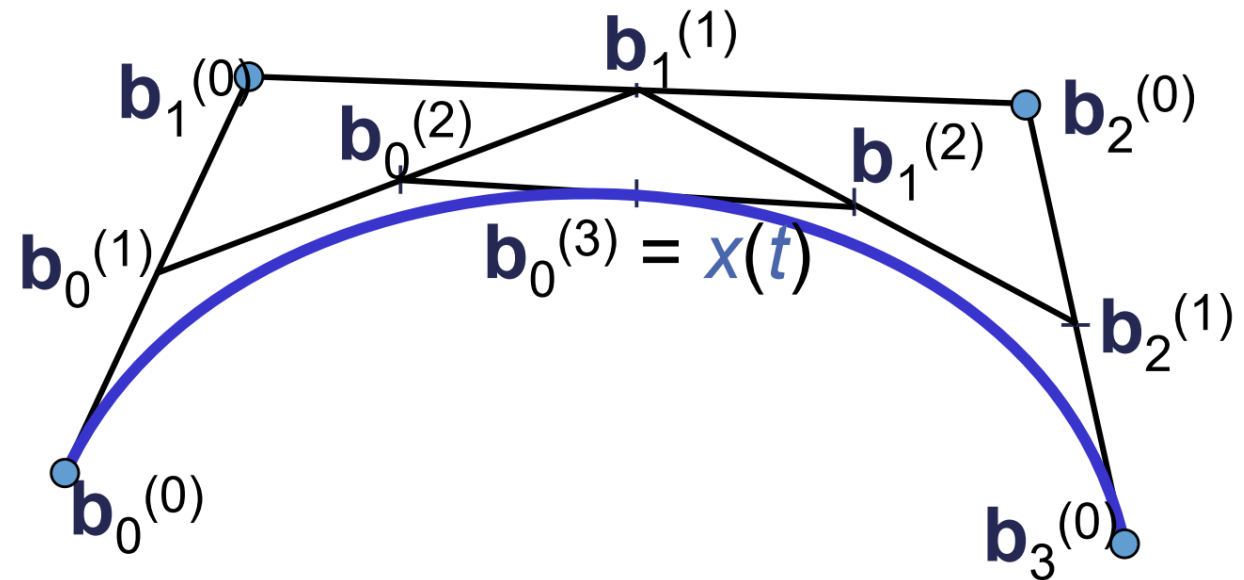
De Casteljau algorithm

- Repeated convex combination of control points

$$\mathbf{b}_i^{(r)} = (1 - t)\mathbf{b}_i^{(r-1)} + t\mathbf{b}_{i+1}^{(r-1)}$$



De Casteljau scheme



De Casteljau algorithm

- The intermediate coefficients $\mathbf{b}_i^r(t)$ can be written in a triangular matrix: the de Casteljau scheme:

$$\mathbf{b}_0 = \mathbf{b}_0^0$$

$$\mathbf{b}_1 = \mathbf{b}_1^0 \quad \mathbf{b}_0^1$$

$$\mathbf{b}_2 = \mathbf{b}_2^0 \quad \mathbf{b}_1^1 \quad \mathbf{b}_0^2$$

$$\mathbf{b}_3 = \mathbf{b}_3^0 \quad \mathbf{b}_2^1 \quad \mathbf{b}_1^2 \quad \mathbf{b}_0^3$$

.....

$$\mathbf{b}_{n-1} = \mathbf{b}_{n-1}^0 \quad \mathbf{b}_{n-2}^1 \quad \dots \quad \mathbf{b}_0^{n-1}$$

$$\mathbf{b}_n = \mathbf{b}_n^0 \quad \mathbf{b}_{n-1}^1 \quad \dots \quad \mathbf{b}_1^{n-1} \quad \mathbf{b}_0^n = x(t)$$

De Casteljau algorithm

Algorithm:

for $r=1..n$

for $i=0..n-r$

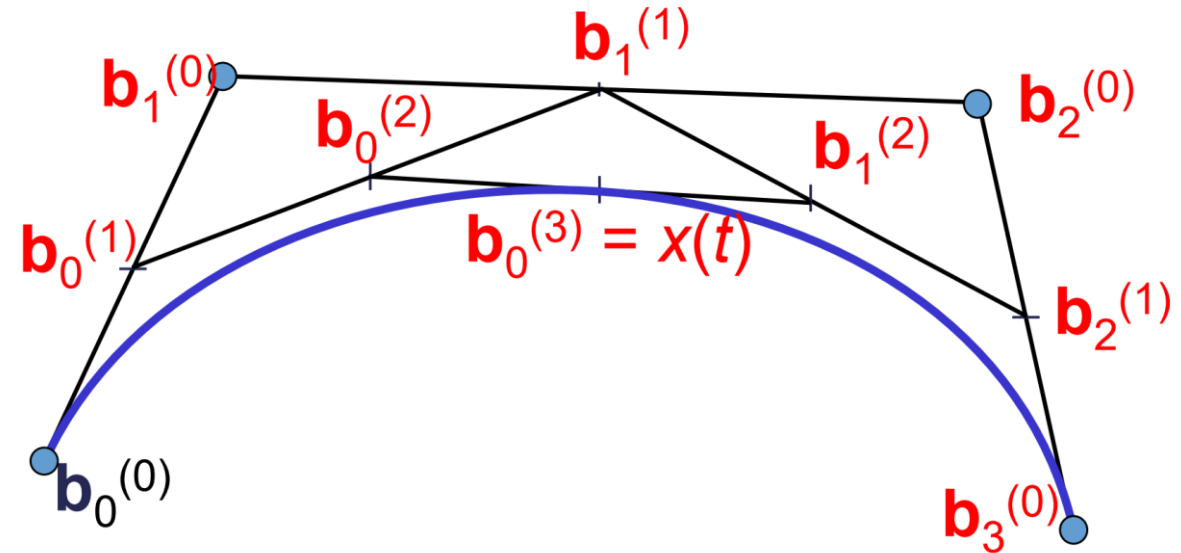
$$\mathbf{b}_i^{(r)} = (1 - t) \mathbf{b}_i^{(r-1)} + t \mathbf{b}_{i+1}^{(r-1)}$$

end

end

return $\mathbf{b}_0^{(n)}$

The whole algorithm consists only of repeated linear interpolations.



De Casteljau algorithm: Properties

- The polygon consisting of the points $\mathbf{b}_0, \dots, \mathbf{b}_n$ is called **Bézier polygon** (control polygon)
- The points \mathbf{b}_i are called **Bézier points** (control points)
- The curve defined by the Bézier points $\mathbf{b}_0, \dots, \mathbf{b}_n$ and the de Casteljau algorithm is called **Bézier curve**
- The de Casteljau algorithm is numerically stable, since only convex combinations are applied.
- Complexity of the de Casteljau algorithm
 - $O(n^2)$ time
 - $O(n)$ memory
 - with n being the number of Bézier points

De Casteljau algorithm: Properties

- **Properties of Bézier curves:**

- Given: Bézier points $\mathbf{b}_0, \dots, \mathbf{b}_n$

Bézier curve $\mathbf{x}(t)$

- Bézier curve is polynomial curve of degree n
 - End points interpolation: $\mathbf{x}(0) = \mathbf{b}_0$, $\mathbf{x}(1) = \mathbf{b}_n$. The remaining Bézier points are only approximated in general

- **Convex hull property:**

Bézier curve is completely inside the convex hull of its Bézier polygon

De Casteljau algorithm: Properties

- **Variation diminishing**
 - No line intersects the Bézier curve more often than its Bézier polygon
- **Influence of Bézier points:** global but pseudo-local
 - Global: moving a Bézier points changes the whole curve progression
 - Pseudo-local: \mathbf{b}_i has its maximal influence on $x(t)$ at $t = \frac{i}{n}$
- **Affine invariance:**
 - Bézier curve and Bézier polygon are invariant under affine transformations
- **Invariance under affine parameter transformations**

De Casteljau algorithm: Properties

- **Symmetry**

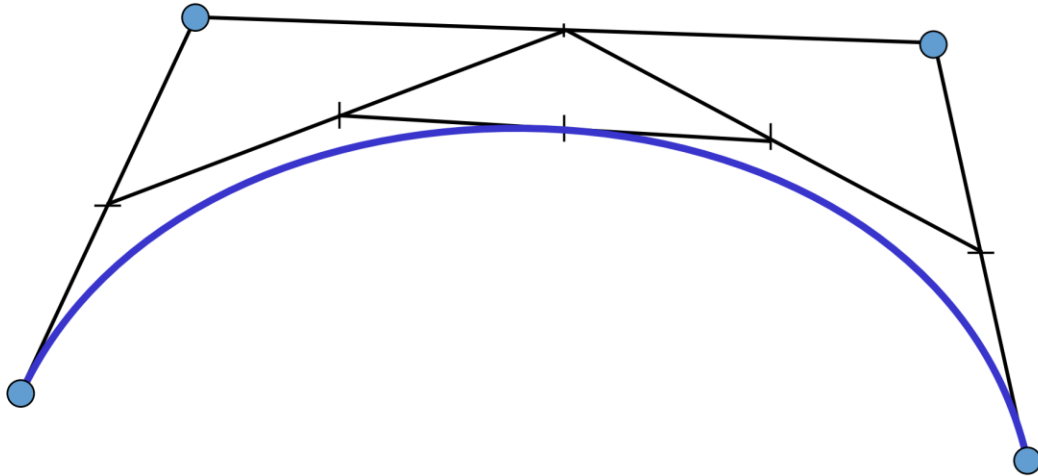
- The following two Bézier curves coincide, they are only traversed in opposite directions:

$$\mathbf{x}(t) = [\mathbf{b}_0, \dots, \mathbf{b}_n] \quad \mathbf{x}'(t) = [\mathbf{b}_n, \dots, \mathbf{b}_0]$$

- **Linear Precision:**

- Bézier curve is line segment, if $\mathbf{b}_0, \dots, \mathbf{b}_n$ are colinear
- Invariance under barycentric combinations

Recap



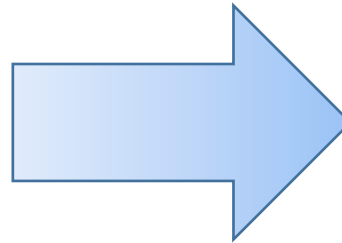
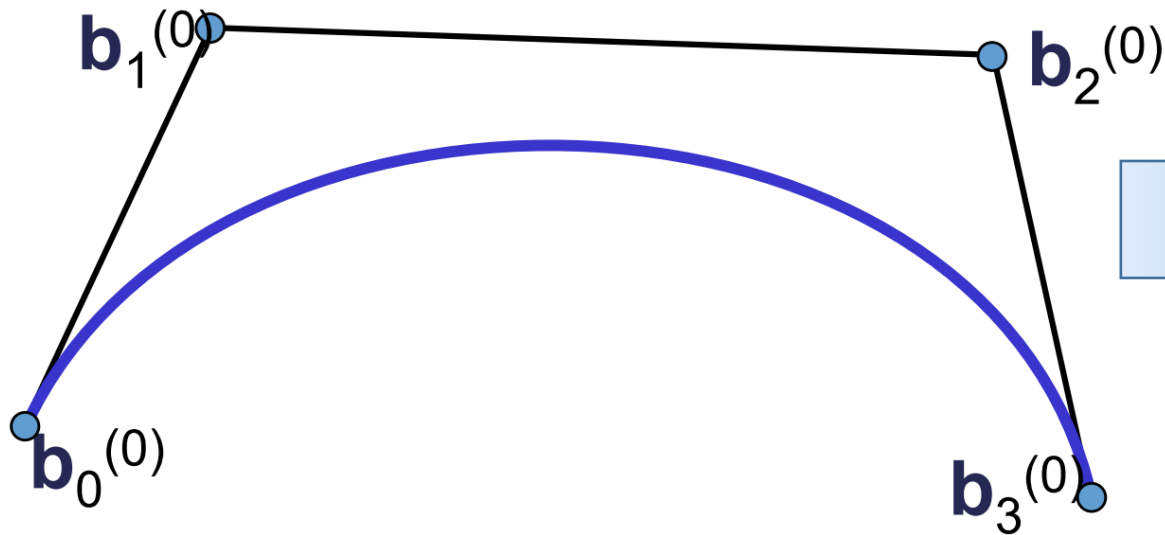
de Casteljau algorithm

Bézier Curves

Towards a polynomial description

Bézier Curves

Towards a polynomial description



$$x(t) = \sum_{i=0}^n B_i^n(t) \cdot b_i$$

Polynomial description of Bézier curves

- The same problem as before:
 - Given: $(n + 1)$ control points $\mathbf{b}_0, \dots, \mathbf{b}_n$
 - Wanted: Bézier curve $\mathbf{x}(t)$ with $t \in [0,1]$
- Now with an algebraic approach using basis functions

Desirable Properties

- Useful requirements for a basis:
 - Well behaved curve
 - Smooth basis functions

Desirable Properties

- Useful requirements for a basis:
 - Well behaved curve
 - Smooth basis functions
 - Local control (or at least semi-local)
 - Basis functions with compact support

Desirable Properties

- Useful requirements for a basis:
 - Well behaved curve
 - Smooth basis functions
 - Local control (or at least semi-local)
 - Basis functions with compact support
 - Affine invariance:
 - Applying an affine map $\mathbf{x} \rightarrow A\mathbf{x} + \mathbf{b}$ on
 - Control points
 - Curve
- Should have the same effect**
- In particular: rotation, translation
 - Otherwise: interactive curve editing very difficult

Desirable Properties

- Useful requirements for a basis:
 - **Convex hull property:**
 - The curve lays within the convex hull of its control points
 - Avoids at least too weird oscillations
- Advantages
 - Computational advantages (recursive intersection tests)
 - More predictable behavior

Summary

- Useful properties
 - Smoothness
 - Local control / support
 - **Affine invariance**
 - **Convex hull property**

Notations

Curve basis function control points

$$f(t) = \sum_{i=1}^n b_i(t) \mathbf{p}_i$$

Affine Invariance

- Affine map: $\mathbf{x} \rightarrow A\mathbf{x} + \mathbf{b}$
- **Part I:** Linear invariance – we get this automatically

- Linear approach: $\mathbf{f}(t) = \sum_{i=1}^n b_i(t) \mathbf{p}_i = \sum_{i=1}^n b_i(t) \begin{pmatrix} p_i^{(x)} \\ p_i^{(y)} \\ p_i^{(z)} \end{pmatrix}$

- Therefore: $A(\mathbf{f}(t)) = A(\sum_{i=1}^n b_i(t) \mathbf{p}_i) = \sum_{i=1}^n b_i(t) (A\mathbf{p}_i)$

Affine Invariance

- Affine Invariance:
 - Affine map: $\mathbf{x} \rightarrow A\mathbf{x} + \mathbf{b}$
 - **Part II:** Translational invariance

$$\sum_{i=1}^n b_i(t)(\mathbf{p}_i + \mathbf{b}) = \sum_{i=1}^n b_i(t)\mathbf{p}_i + \sum_{i=1}^n b_i(t)\mathbf{b} = \mathbf{f}(t) + \left(\sum_{i=1}^n b_i(t) \right) \mathbf{b}$$

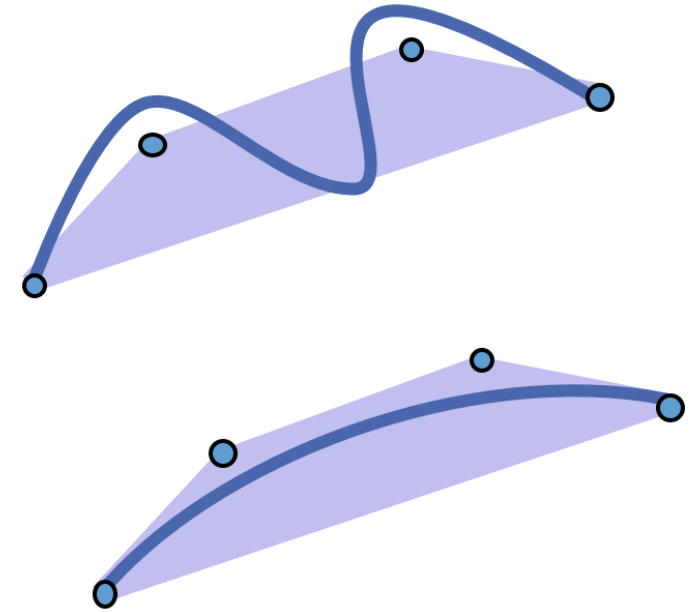
- For translational invariance, the sum of the basis functions must be one *everywhere* (for all parameter values t that are used).
- This is called “**partition of unity** property”
- The b_i ’s form an “**affine combination**” of the control points \mathbf{p}_i
- This is very important for modeling

Convex Hull Property

- Convex combinations:
 - A convex combination of a set of points $\{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ is any point of the form:
$$\sum_{i=1}^n \lambda_i \mathbf{p}_i \text{ with } \sum_{i=1}^n \lambda_i = 1 \text{ and } \forall i = 1 \dots n: 0 \leq \lambda_i \leq 1$$
 - (Remark: $\lambda_i \leq 1$ is redundant)
 - The set of all admissible convex combinations forms the convex hull of the point set
 - Easy to see (exercise): The convex hull is the smallest set that contains all points $\{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ and every complete straight line between two elements of the set

Convex Hull Property

- Accordingly:
 - If we have this property
$$\forall t \in \Omega: \sum_{i=1}^n b_i(t) = 1 \text{ and } \forall t \in \Omega, \forall i: b_i(t) \geq 0$$
the constructed curves / surfaces will be:
 - Affine invariant (translations, linear maps)
 - Be restricted to the convex hull of the control points
 - Corollary: Curves will have *linear precision*
 - All control points lie on a straight line
 \Rightarrow Curve is a straight line segment
 - Surfaces with planar control points will be flat, too



Convex Hull Property

- Very useful property in practice
 - Avoids at least the worst oscillations
 - no escape from convex hull, unlike polynomial interpolation
 - Linear precision property is intuitive (people expect this)
 - Can be used for fast range checks
 - Test for intersection with convex hull first, then the object
 - Recursive intersection algorithms in conjunctions with subdivision rules (more on this later)



Polynomial description of Bézier curves

- The same problem as before:
 - Given: $(n + 1)$ control points $\mathbf{b}_0, \dots, \mathbf{b}_n$
 - Wanted: Bézier curve $\mathbf{x}(t)$ with $t \in [0,1]$
- Now with an algebraic approach using basis functions
- Need to define $n + 1$ basis functions
 - Such that this describes a Bézier curve:

$$\mathbf{x}(t) = \sum_{i=0}^n B_i^n(t) \cdot \mathbf{b}_i$$

Bernstein Basis

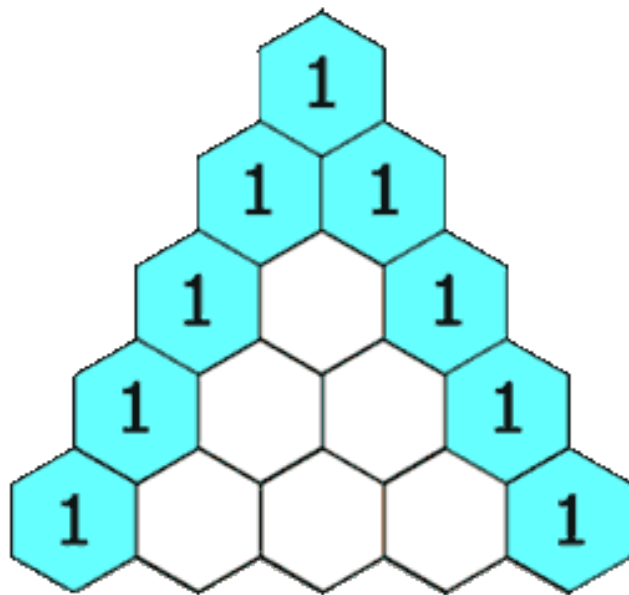
- Let's examine the Bernstein basis: $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$
 - Bernstein basis of degree n :

$$B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i} = B_{i\text{-th basis function}}^{(\text{degree})}$$

where the binomial coefficients are given by:

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

Binomial Coefficients and Theorem



$$\binom{n}{i} + \binom{n}{i+1} = \binom{n+1}{i+1}$$

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5	10		10		5		1	

$$(x + y)^n = \sum_{i=0}^n \binom{n}{i} x^i y^{n-i}$$

Examples: The first few

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

- The first three Bernstein bases:

$$B_0^{(0)} := 1$$

$$B_0^{(1)} := 1 - t \quad B_1^{(1)} := t$$

$$B_0^{(2)} := (1 - t)^2 \quad B_1^{(2)} := 2t(1 - t) \quad B_2^{(2)} := t^2$$

$$B_0^{(3)} := (1 - t)^3 \quad B_1^{(3)} := 3t(1 - t)^2 \quad B_2^{(3)} := 3t^2(1 - t) \quad B_3^{(3)} := t^3$$

Examples: The first few

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

$$B_0^{(0)} := 1$$

$$B_0^{(1)} := 1 - t$$

$$B_1^{(1)} := t$$

$$B_0^{(2)} := (1 - t)^2$$

$$B_1^{(2)} := 2t(1 - t)$$

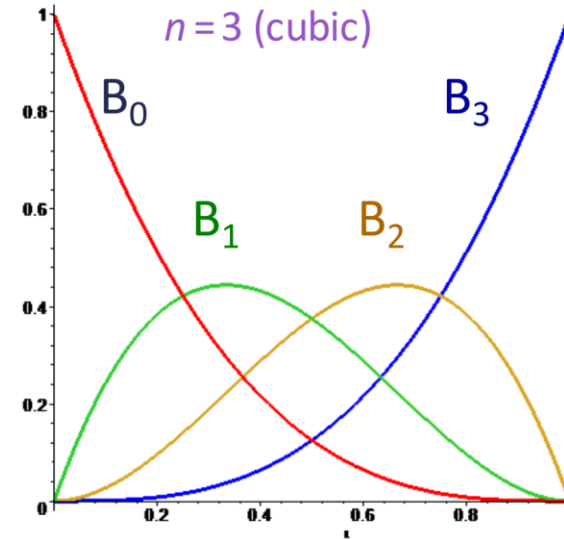
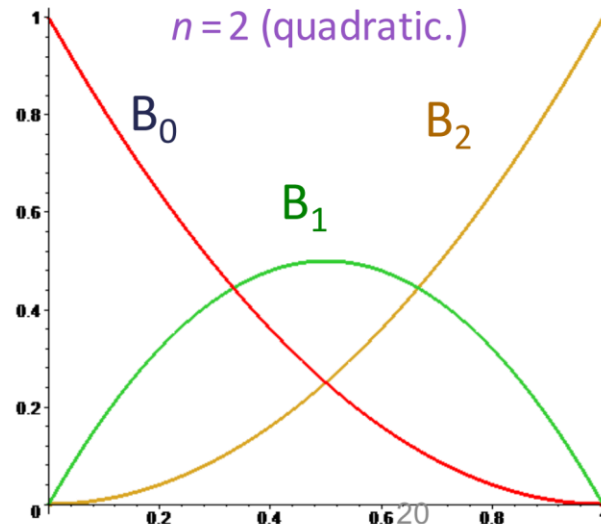
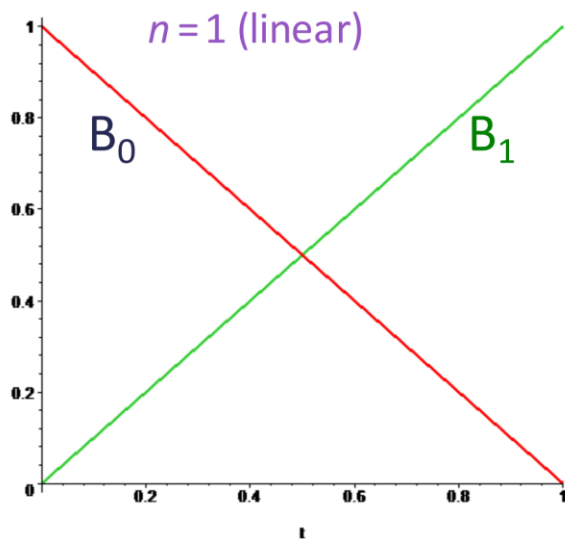
$$B_2^{(2)} := t^2$$

$$B_0^{(3)} := (1 - t)^3$$

$$B_1^{(3)} := 3t(1 - t)^2$$

$$B_2^{(3)} := 3t^2(1 - t)$$

$$B_3^{(3)} := t^3$$

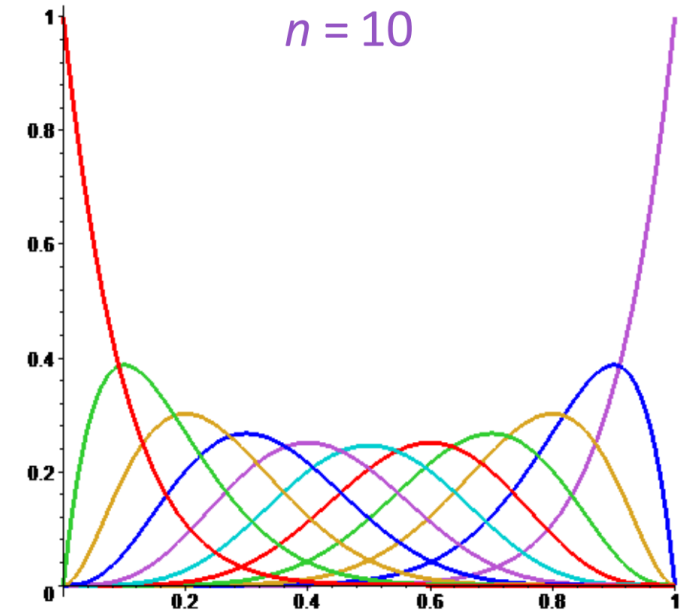
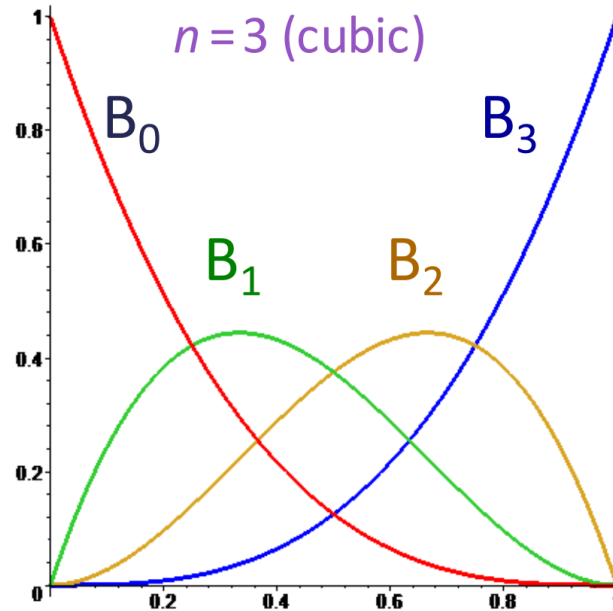
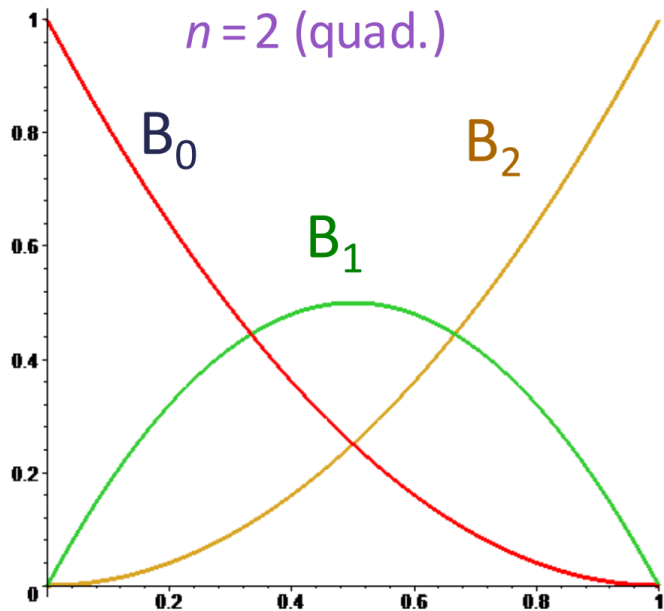


Bernstein Basis

- Bézier curves use the Bernstein basis: $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$

- Bernstein basis of degree n :

$$B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i} = B_{i\text{-th basis function}}^{(\text{degree})}$$



Bernstein Basis

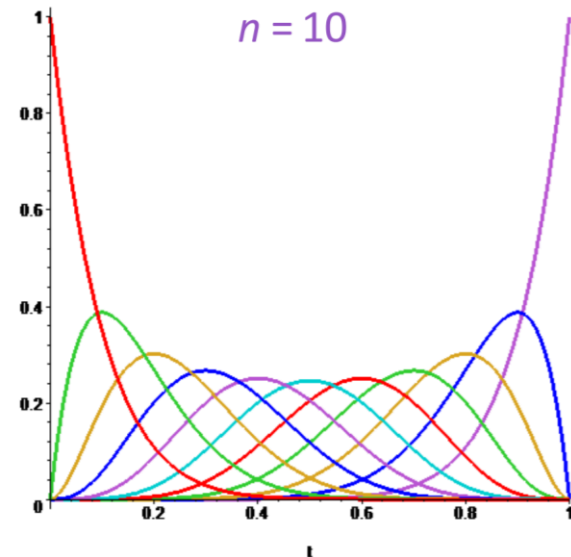
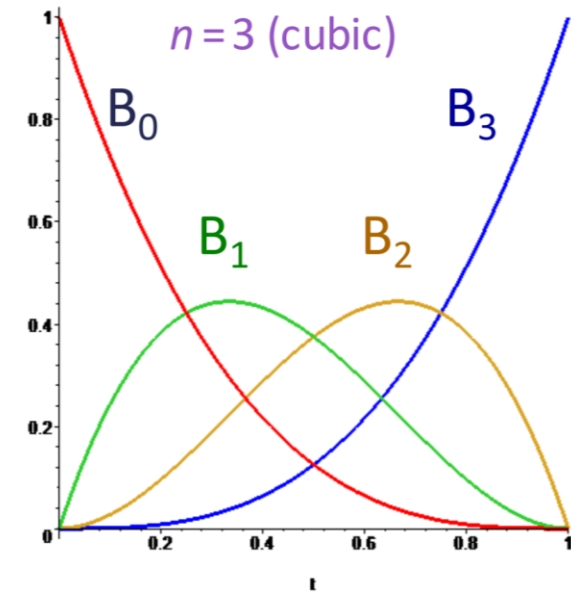
- What about the desired properties?
 - Smoothness
 - Local control / support
 - Affine invariance
 - Convex hull property

Bernstein Basis: Properties

- $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$
- Basis for polynomials of degree n
- Each basis function $B_i^{(n)}$ has its maximum at $t = \frac{i}{n}$

Smoothness

Local control (semi-local)



Bernstein Basis: Properties

- $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

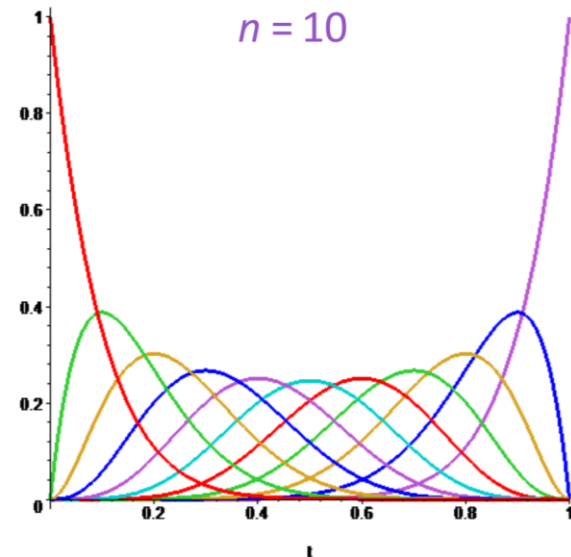
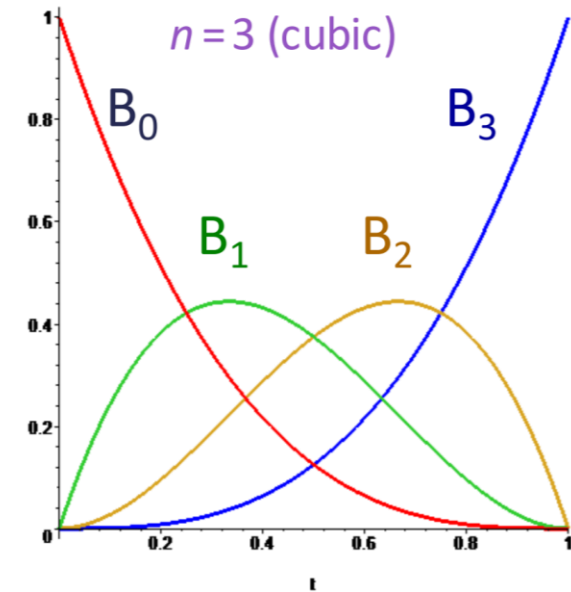
Affine invariance

Convex hull property

- Partition of unity (binomial theorem)

$$1 = (1 - t + t)$$

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



What about the desired properties?

- Smoothness Yes
- Local control / support To some extent
- Affine invariance Yes
- Convex hull property Yes

$$\binom{n-1}{i} + \binom{n-1}{i-1} = \binom{n}{i}$$

Bernstein Basis: Properties

- $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

- Recursive computation

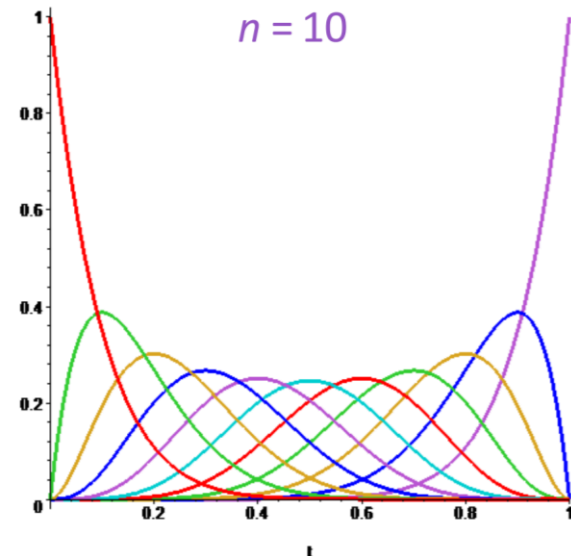
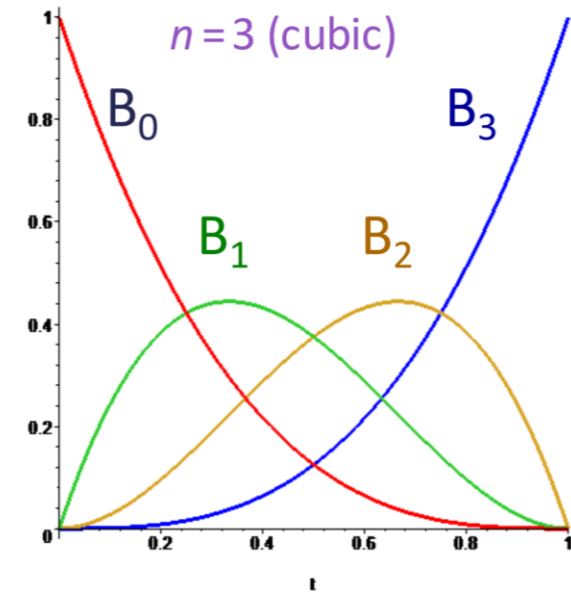
$$B_i^n(t) := (1-t)B_i^{(n-1)}(t) + tB_{i-1}^{(n-1)}(1-t)$$

with $B_0^0(t) = 1$, $B_i^n(t) = 0$ for $i \notin \{0 \dots n\}$

- Symmetry

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

- Non-negativity: $B_i^{(n)}(t) \geq 0$ for $t \in [0..1]$



Bernstein Basis: Properties

- $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

- Non-negativity II

$$B_i^n(t) > 0 \text{ for } 0 < t < 1$$

$$B_0^n(0) = 1, \quad B_1^n(0) = \dots = B_n^n(0) = 0$$

$$B_0^n(1) = \dots = B_{n-1}^n(1) = 0, \quad B_n^n(1) = 1$$

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

$$B_i^{(n)}(t) > 0 \text{ for } 0 < t < 1$$

$$B_1^n(0) = \dots = B_n^n(0) = 0 \\ B_{n-1}^n(1) = 0,$$

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

$$B_n^n(0) = 0 \\ B_n^n(1) = 1$$

Derivatives

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

- Bernstein basis properties
 - Derivatives:

$$\frac{d}{dt} B_i^{(n)}(t) =$$

Derivatives

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

- Bernstein basis properties
 - Derivatives:

$$\begin{aligned} \frac{d}{dt} B_i^{(n)}(t) &= \binom{n}{i} (i t^{i-1} (1-t)^{n-i} - (n-i) t^i (1-t)^{n-i-1}) \\ &= \frac{n!}{(n-i)! i!} i t^{i-1} (1-t)^{n-i} - \frac{n!}{(n-i)! i!} (n-i) t^i (1-t)^{n-i-1} \\ &= n \left[\binom{n-1}{i-1} t^{i-1} (1-t)^{n-i} - \binom{n-1}{i} t^i (1-t)^{n-i-1} \right] \\ &= n \left[B_{i-1}^{(n-1)}(t) - B_i^{(n-1)}(t) \right] \end{aligned}$$

(Notation: $\{k\} = k$ if $k > 0$, zero otherwise)

Derivatives

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

- Bernstein basis properties
 - Derivatives:

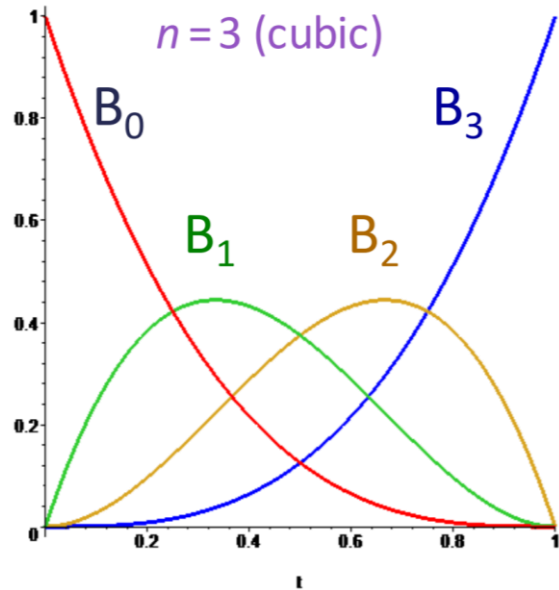
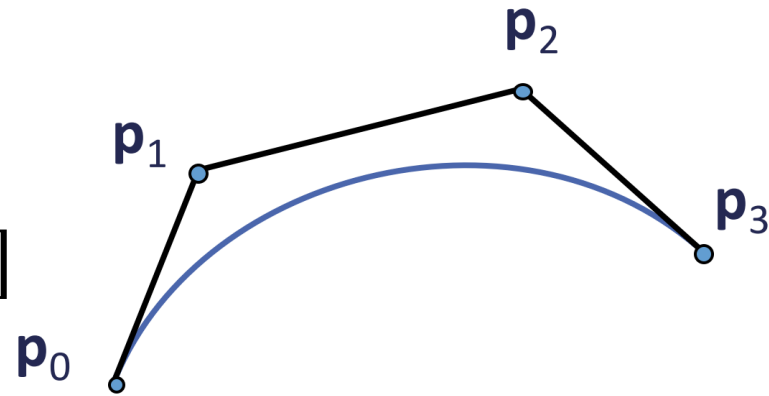
$$\begin{aligned} \frac{d^2}{dt^2} B_i^{(n)}(t) &= \frac{d}{dt} n \left[B_{i-1}^{(n-1)}(t) - B_i^{(n-1)}(t) \right] \\ &= n \left[(n-1) \left(B_{i-2}^{(n-2)}(t) - B_{i-1}^{(n-2)}(t) \right) - (n-1) \left(B_{i-1}^{(n-2)}(t) - B_i^{(n-2)}(t) \right) \right] \\ &= n(n-1) \left[B_{i-2}^{(n-2)}(t) - 2B_{i-1}^{(n-2)}(t) + B_i^{(n-2)}(t) \right] \end{aligned}$$

(Notation: $\{k\} = k$ if $k > 0$, zero otherwise)

Bézier Curves in Bernstein form

- Bézier Curves:

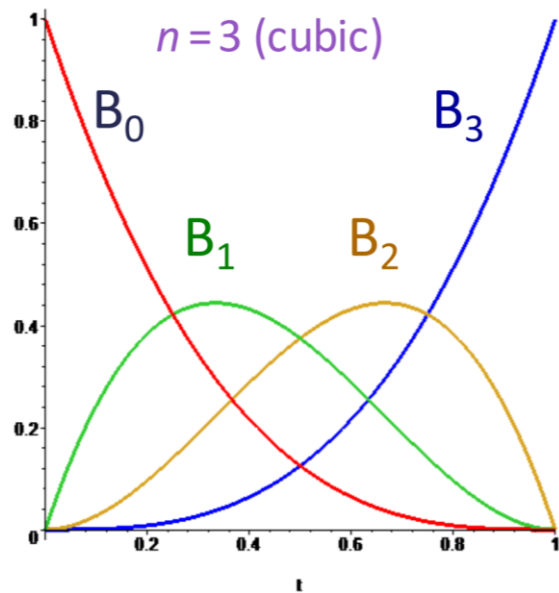
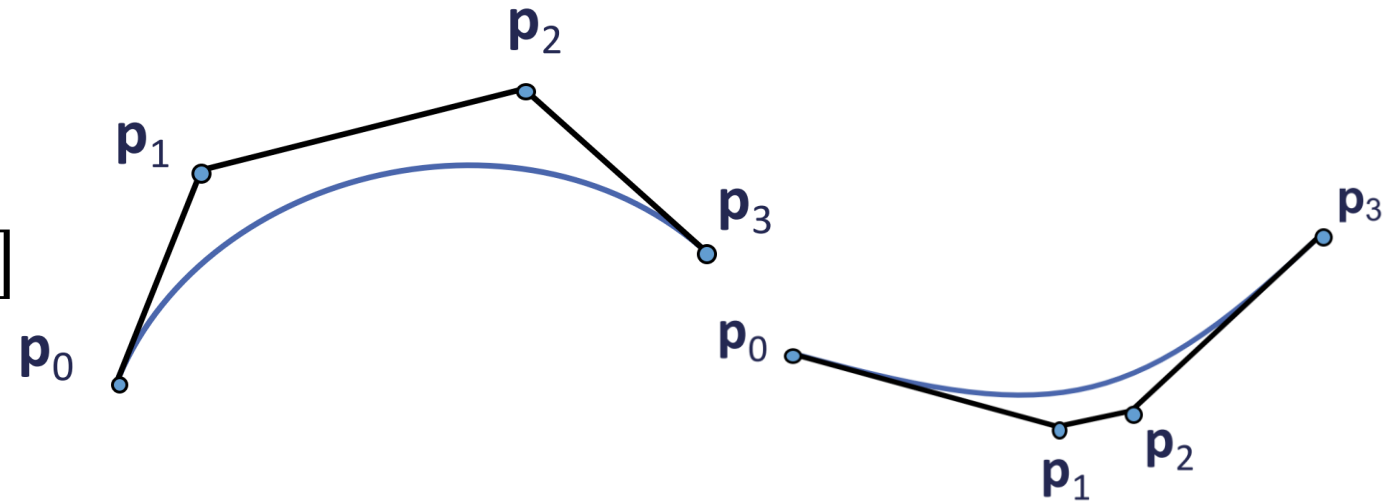
$$f(t) = \sum_{i=0}^n B_i^n \mathbf{p}_i, t \in [0,1]$$



Bézier Curves in Bernstein form

- Bézier Curves:

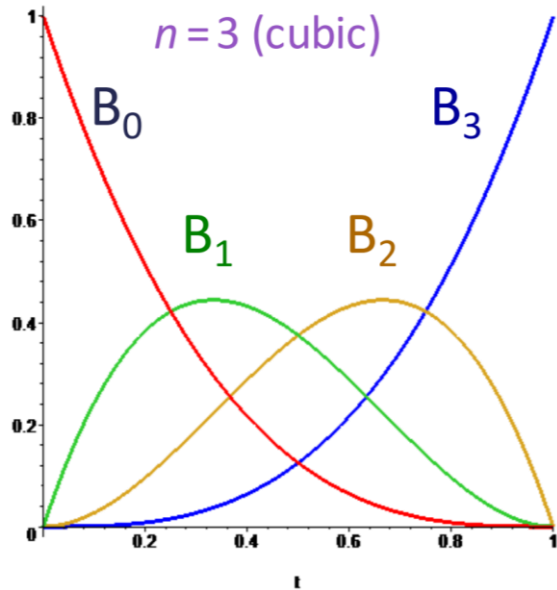
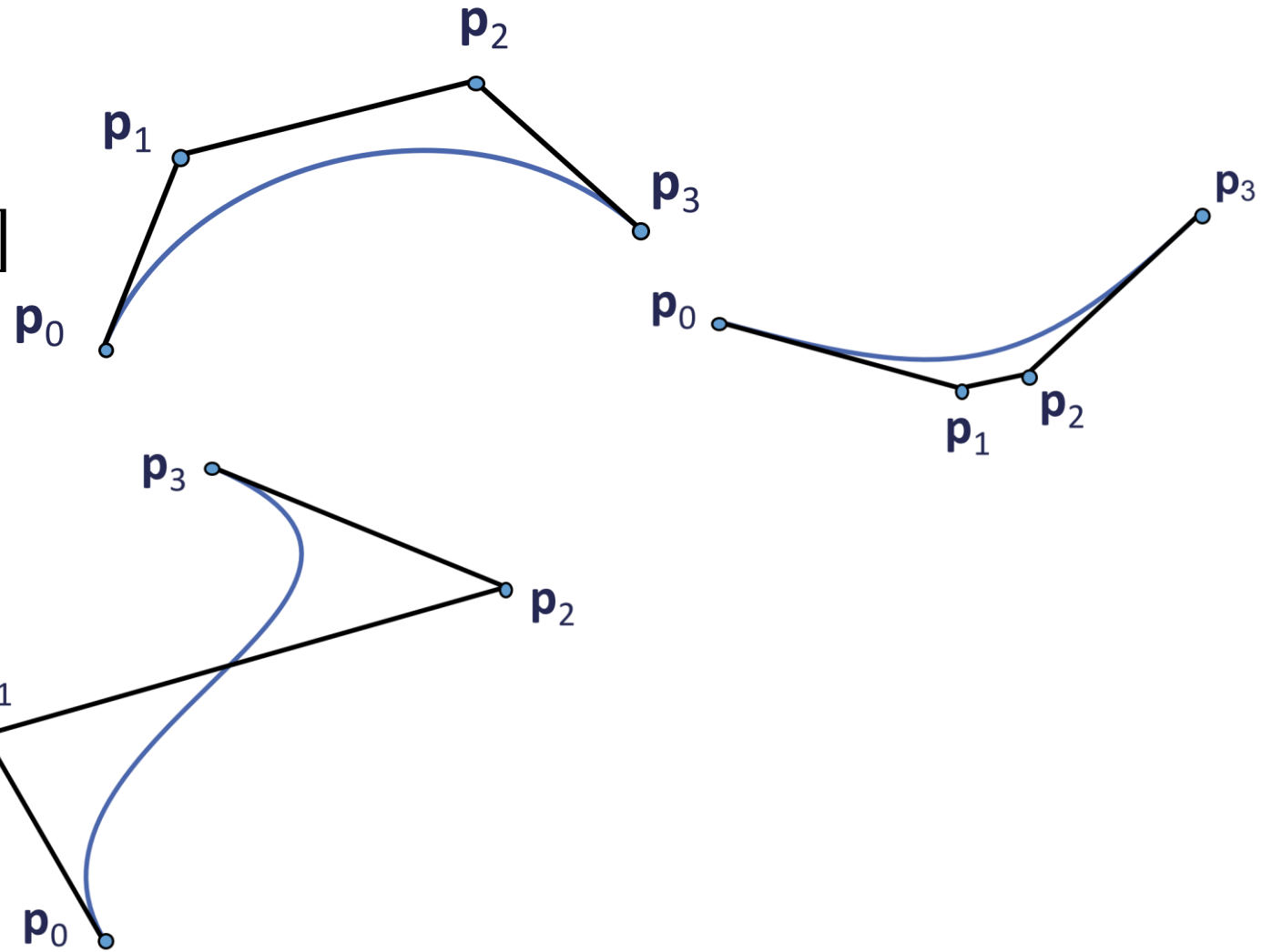
$$f(t) = \sum_{i=0}^n B_i^n \mathbf{p}_i, t \in [0,1]$$



Bézier Curves in Bernstein form

- Bézier Curves:

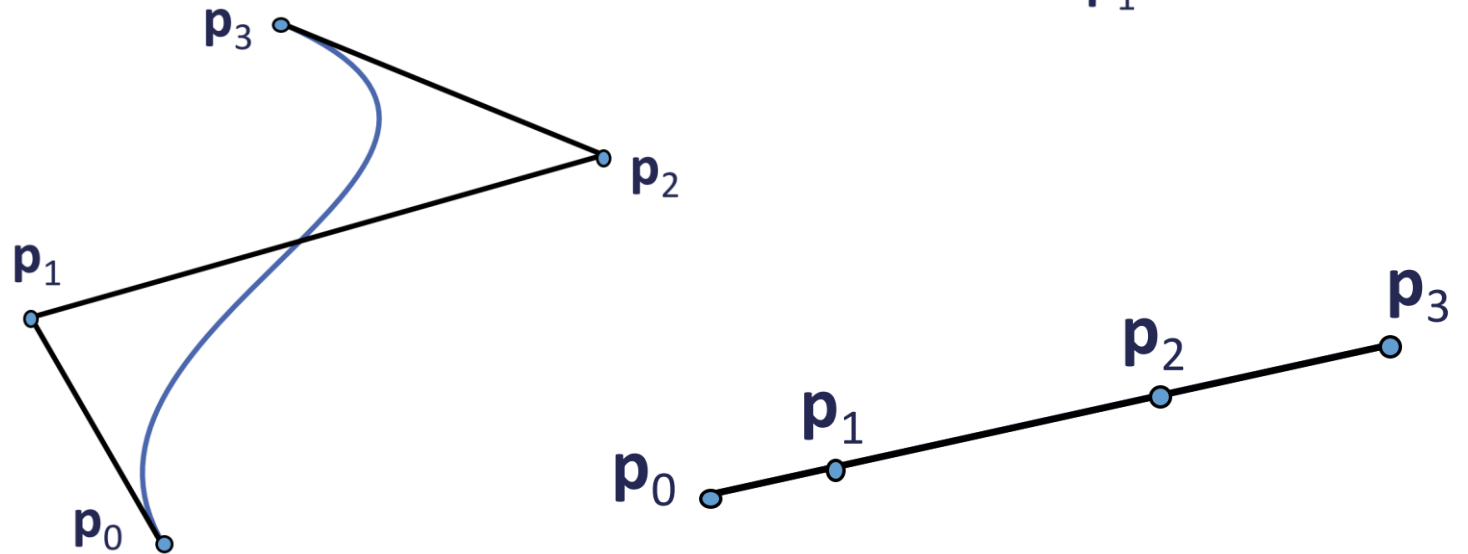
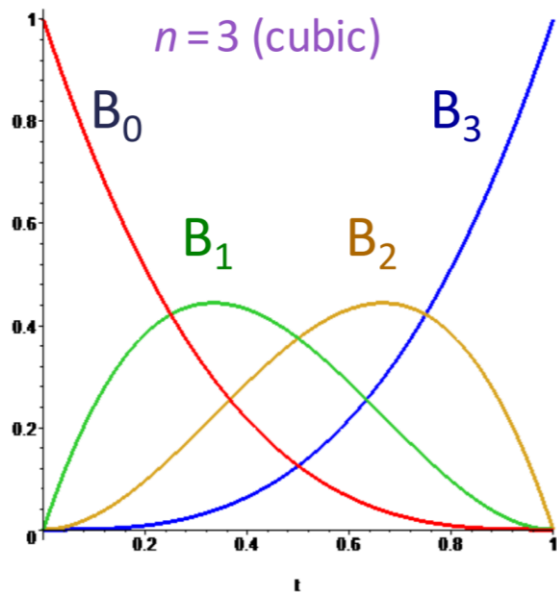
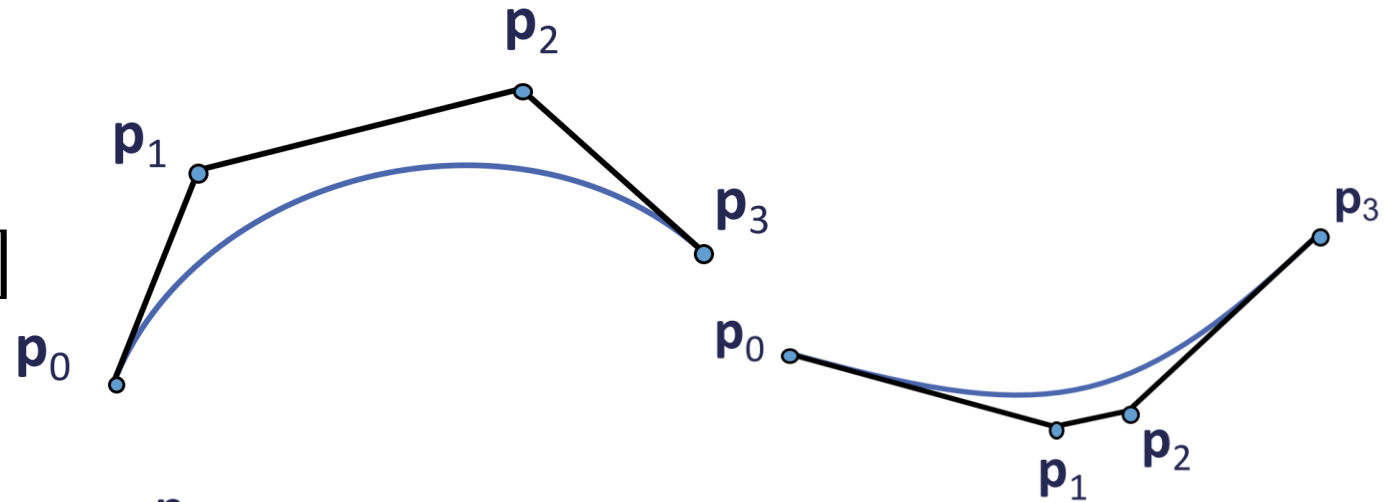
$$f(t) = \sum_{i=0}^n B_i^n \mathbf{p}_i, t \in [0,1]$$



Bézier Curves in Bernstein form

- Bézier Curves:

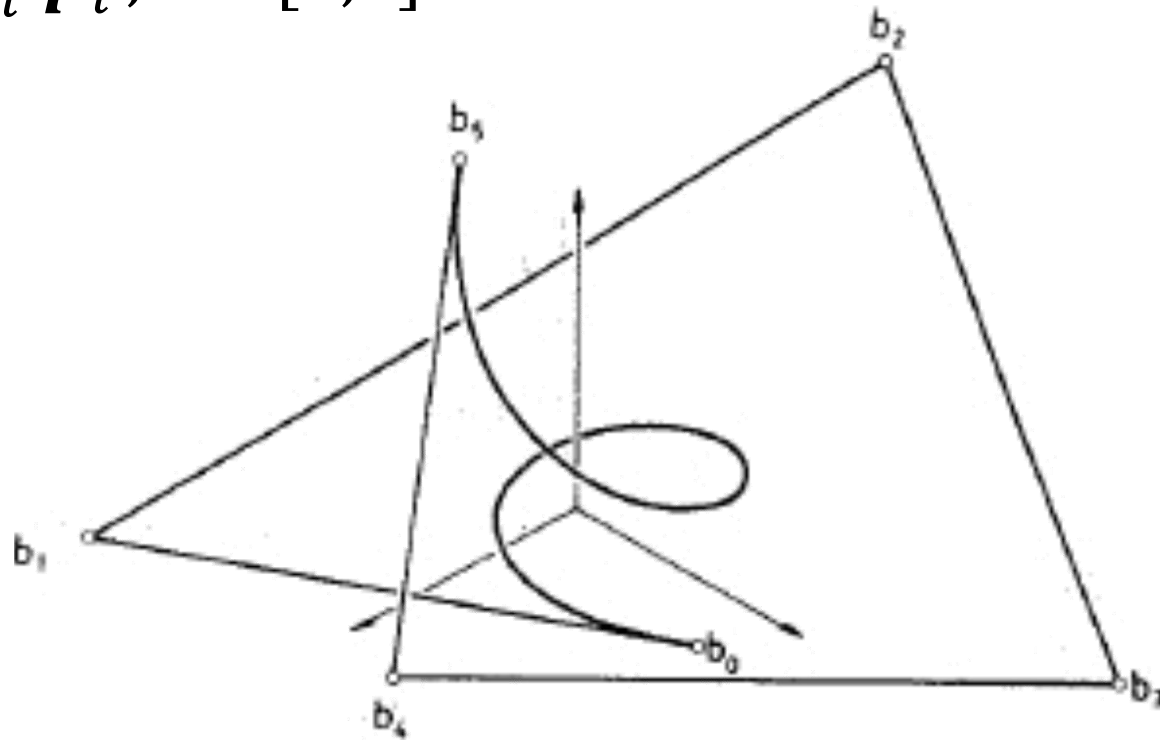
$$f(t) = \sum_{i=0}^n B_i^n \mathbf{p}_i, t \in [0,1]$$



Bézier Curves in Bernstein form

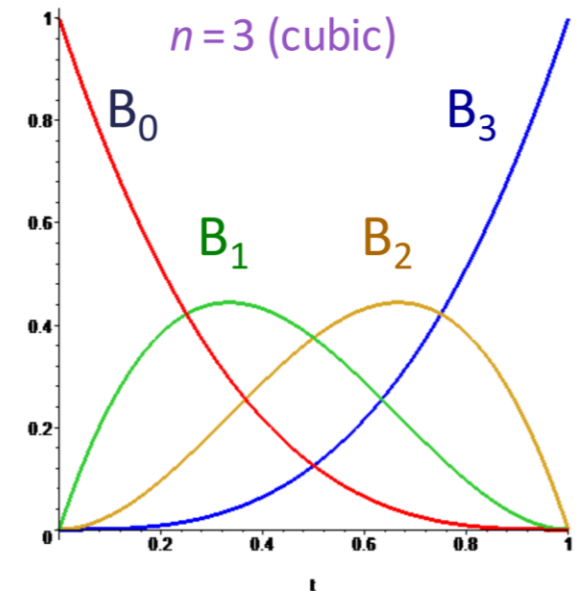
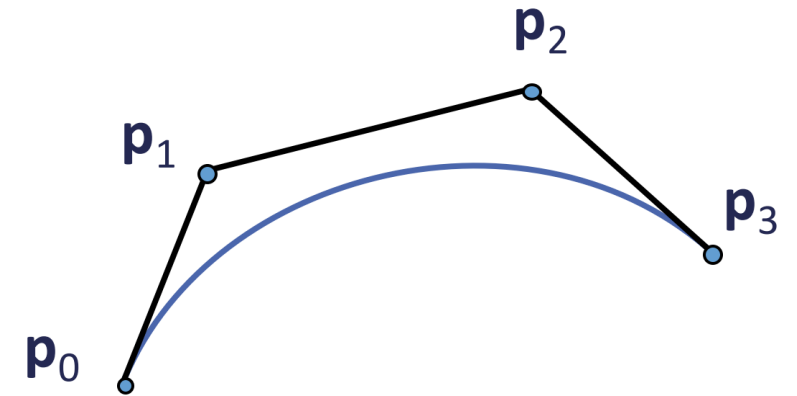
- Bézier Curves, also in 3D

$$\mathbf{f}(t) = \sum_{i=1}^n B_i^n \mathbf{p}_i, t \in [0,1]$$



Bézier Curves in Bernstein form

- Bézier curves:
 - Curves: $\mathbf{f}(t) = \sum_{i=1}^n B_i^n \mathbf{p}_i$
 - Considering the interval $t \in [0..1]$
 - Properties as discussed before:
 - Affine invariant
 - Curves contained in the convex hull
 - Influence of control points
- Moving along the curve with index i
- Largest influence at $t = \frac{i}{n}$
- Single curve segments: no full local control



Bézier Curve Properties: another look at derivatives

- Given: $\mathbf{p}_0, \dots, \mathbf{p}_n, \mathbf{f}(t) = \sum_{i=0}^n B_i^n(t) \mathbf{p}_i$
- Then: $\mathbf{f}'(t) = n \sum_{i=0}^{n-1} B_i^{n-1}(t) (\mathbf{p}_{i+1} - \mathbf{p}_i)$
- Proof: $\mathbf{f}'(t) = \sum_{i=0}^n \frac{d}{dt} B_i^n(t) \mathbf{p}_i = n \sum_{i=0}^n \left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right) \mathbf{p}_i$

$$= n \sum_{i=0}^n B_{i-1}^{n-1}(t) \mathbf{p}_i - n \sum_{i=0}^n B_i^{n-1}(t) \mathbf{p}_i$$

Index change

$$= n \sum_{i=-1}^{n-1} B_i^{n-1}(t) \mathbf{p}_{i+1} - n \sum_{i=0}^n B_i^{n-1}(t) \mathbf{p}_i = n \sum_{i=0}^{n-1} B_i^{n-1}(t) \mathbf{p}_{i+1} - n \sum_{i=0}^{n-1} B_i^{n-1}(t) \mathbf{p}_i$$

$$= n \sum_{i=0}^{n-1} B_i^{n-1}(t) (\mathbf{p}_{i+1} - \mathbf{p}_i)$$

Bézier Curve Properties

- Higher order derivatives:

$$f^{[r]}(t) = \frac{n!}{(n-r)!} \cdot \sum_{i=0}^{n-r} B_i^{n-r}(t) \cdot \Delta^r \mathbf{p}_i$$

Bézier Curve Properties

- Important for continuous concatenation:

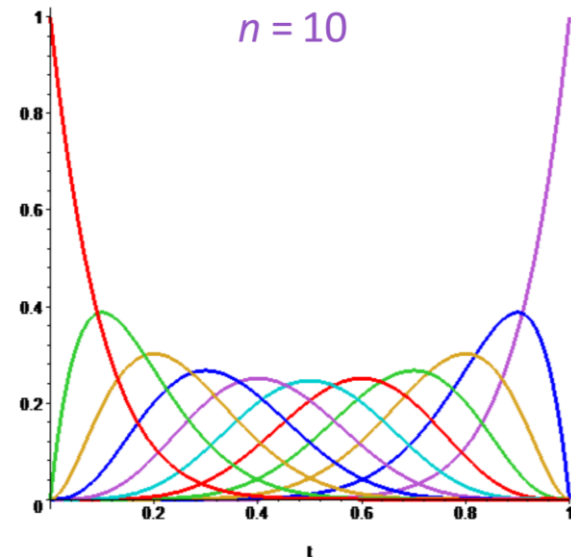
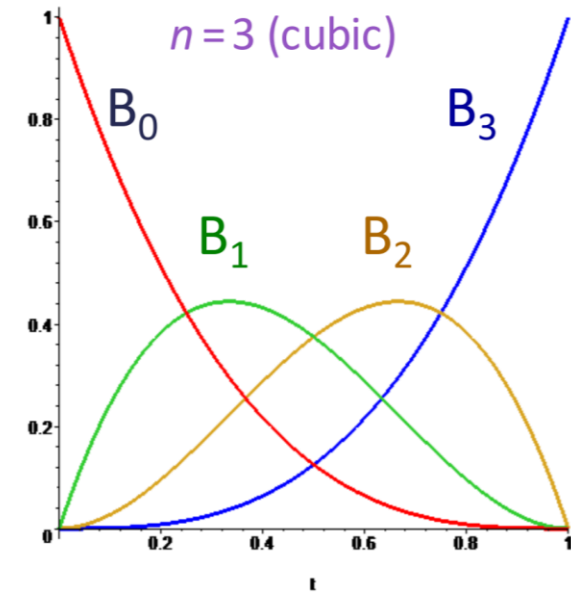
- Function value at $\{0,1\}$:

$$f(t) = \sum_{i=0}^{n-1} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{p}_i$$

$$\Rightarrow f(0) = \mathbf{p}_0$$

$$f(1) = \mathbf{p}_1$$

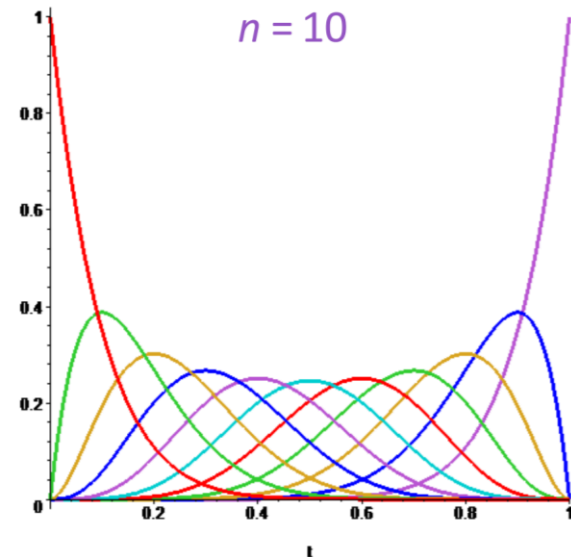
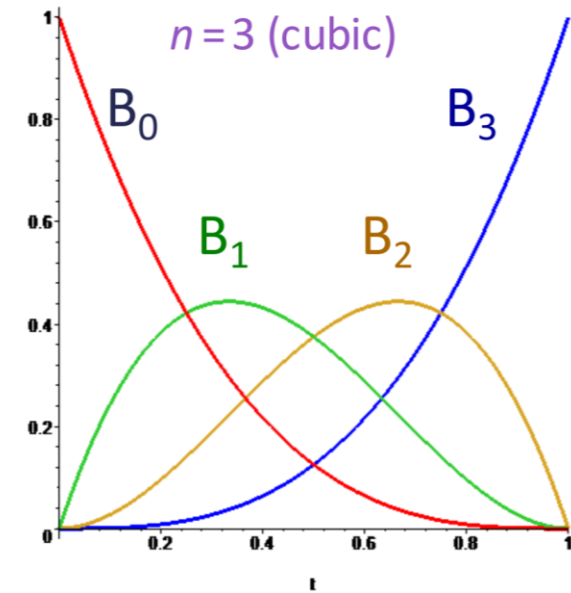
- First derivative vector at $\{0,1\}$
 - Second derivative vector at $\{0,1\}$



Bézier Curve Properties

First derivative vector at $\{0,1\}$

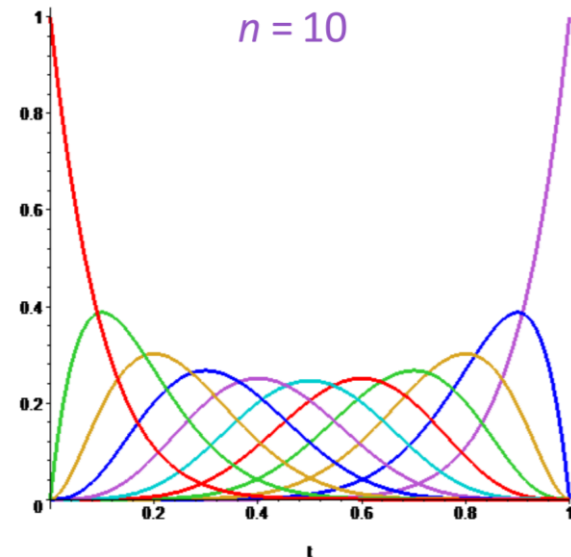
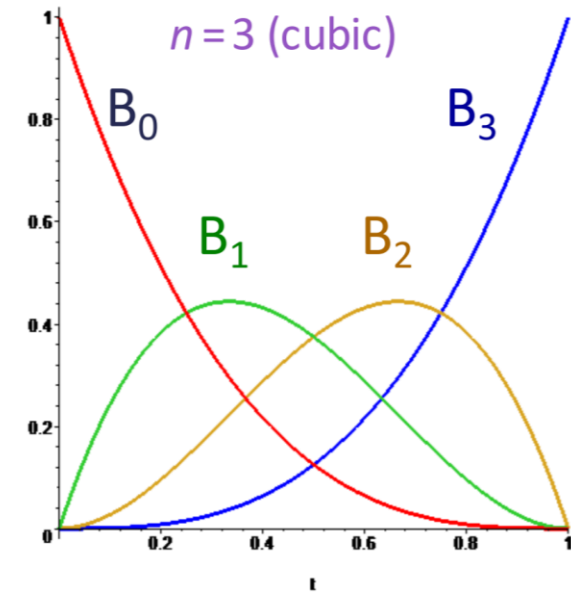
$$\frac{d}{dt} \mathbf{f}(t) =$$



Bézier Curve Properties

First derivative vector at $\{0,1\}$

$$\frac{d}{dt} \mathbf{f}(t) = n \sum_{i=0}^{n-1} \left[B_{i-1}^{(n-1)}(t) - B_i^{(n-1)}(t) \right] \mathbf{p}_i$$



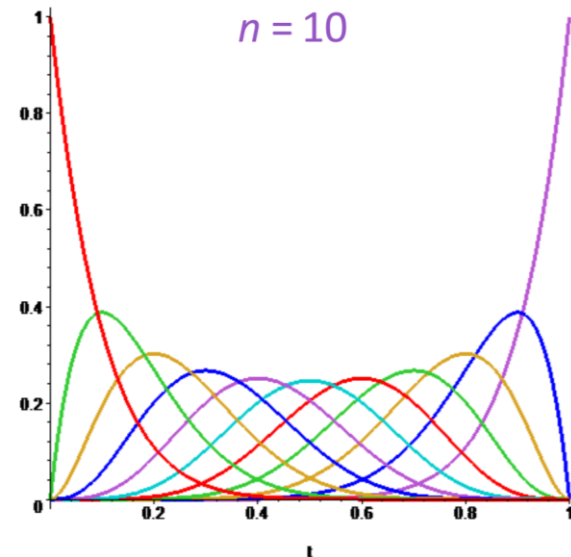
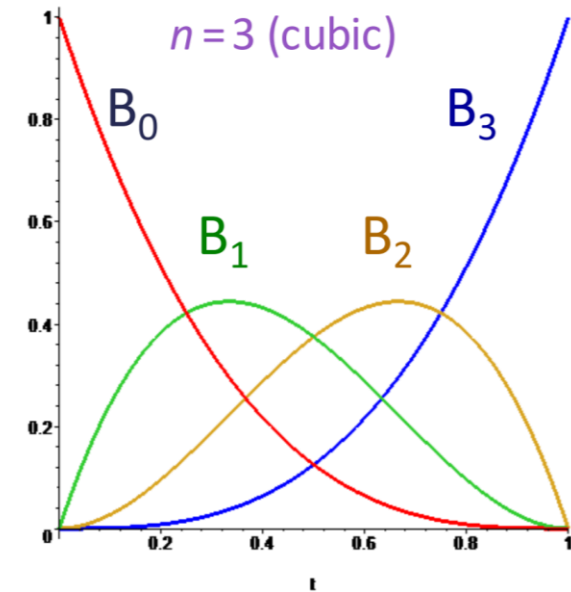
Bézier Curve Properties

First derivative vector at $\{0,1\}$

$$\begin{aligned}\frac{d}{dt}\mathbf{f}(t) &= n \sum_{i=0}^{n-1} \left[B_{i-1}^{(n-1)}(t) - B_i^{(n-1)}(t) \right] \mathbf{p}_i \\ &= n \left(\left[-B_0^{(n-1)}(t) \right] \mathbf{p}_0 + \left[B_0^{(n-1)}(t) - B_1^{(n-1)}(t) \right] \mathbf{p}_1 + \dots \right. \\ &\quad \left. + \left[B_{n-2}^{(n-1)}(t) - B_{n-1}^{(n-1)}(t) \right] \mathbf{p}_{n-1} + \left[B_{n-1}^{(n-1)}(t) \right] \mathbf{p}_n \right)\end{aligned}$$

$$\frac{d}{dt}\mathbf{f}(0) = n(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\frac{d}{dt}\mathbf{f}(1) = n(\mathbf{p}_n - \mathbf{p}_{n-1})$$



Bézier Curve Properties

- Important for continuous concatenation:

- Function value at $\{0,1\}$:

$$f(0) = p_0$$

$$f(1) = p_1$$

- First derivative vector at $\{0,1\}$

$$f'(0) = n[p_1 - p_0]$$

$$f'(1) = n[p_n - p_{n-1}]$$

- Second derivative vector at $\{0,1\}$

$$f''(0) = n(n-1)[p_2 - 2p_1 + p_0]$$

$$f''(1) = n(n-1)[p_n - 2p_{n-1} + p_{n-2}]$$

