# 计算机辅助几何设计
# 2019秋学期

# Bezier Curves

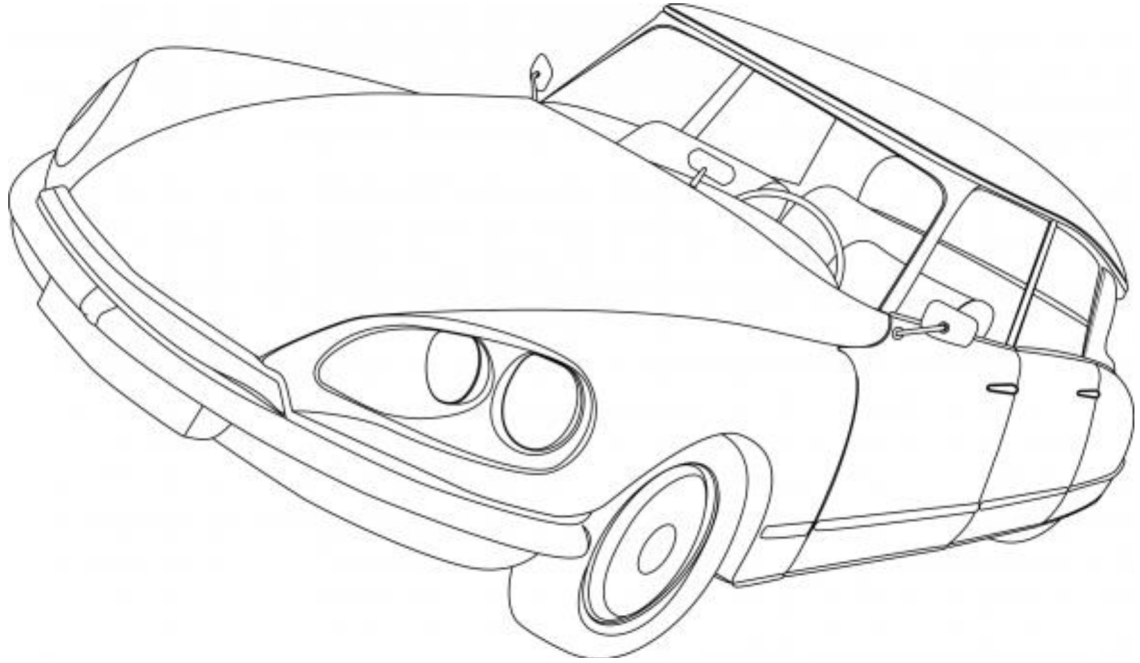
陈仁杰

中国科学技术大学

# Bezier curves

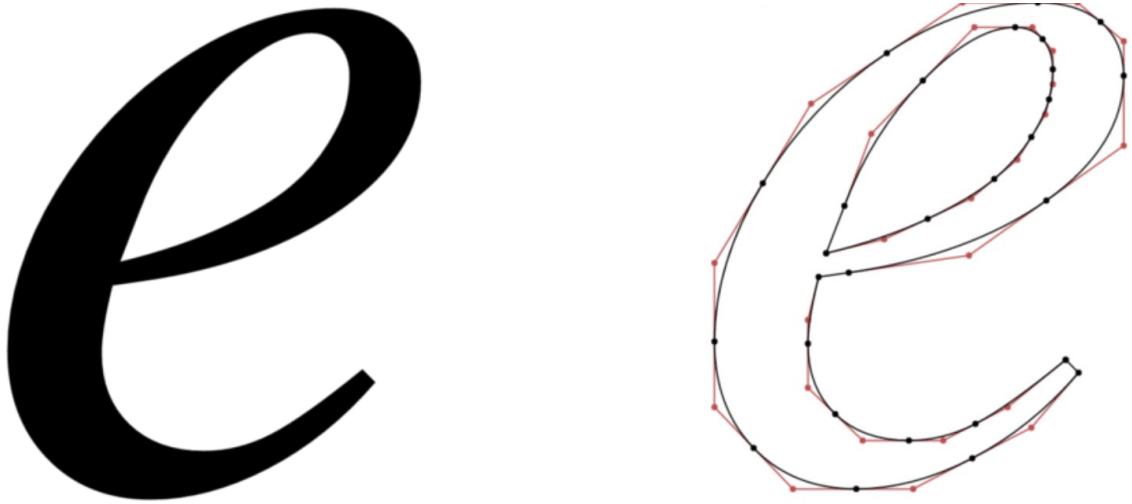
- Bezier curves/splines developed by
  - Paul de Casteljau at Citroen (1959)
  - Pierre Bezier at Renault (1963)

  for free-form parts in automotive design



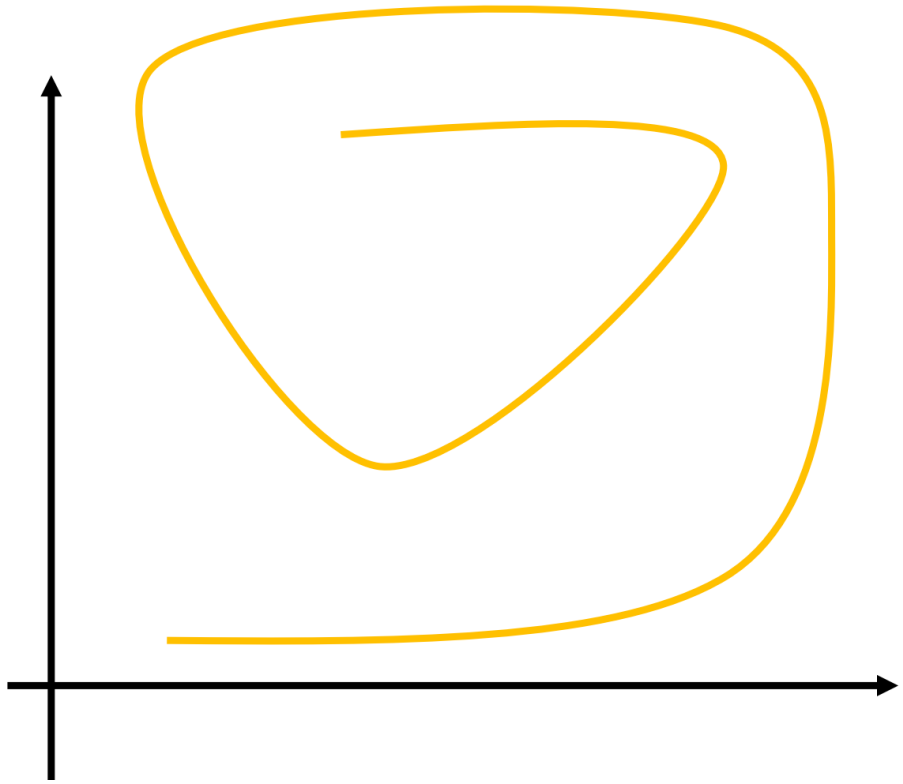
# Bezier curves

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



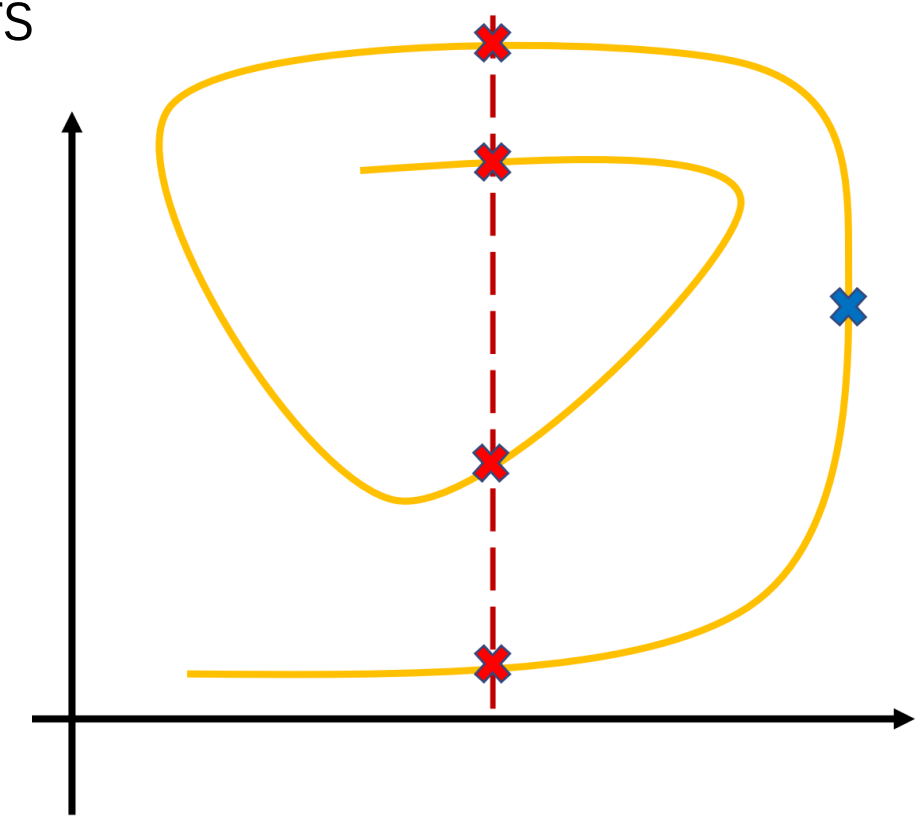
# Curve representation

- 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

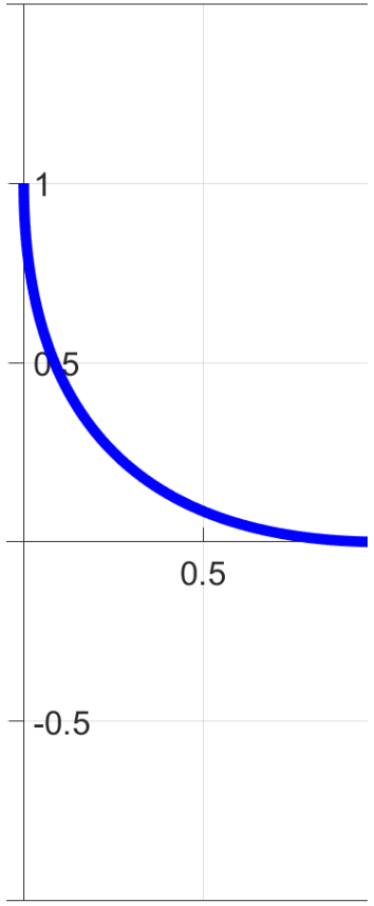


# Parametric representation

- Remedy: parametric representation $c(t) = \big(x(t), y(t)\big)$
  - 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, …

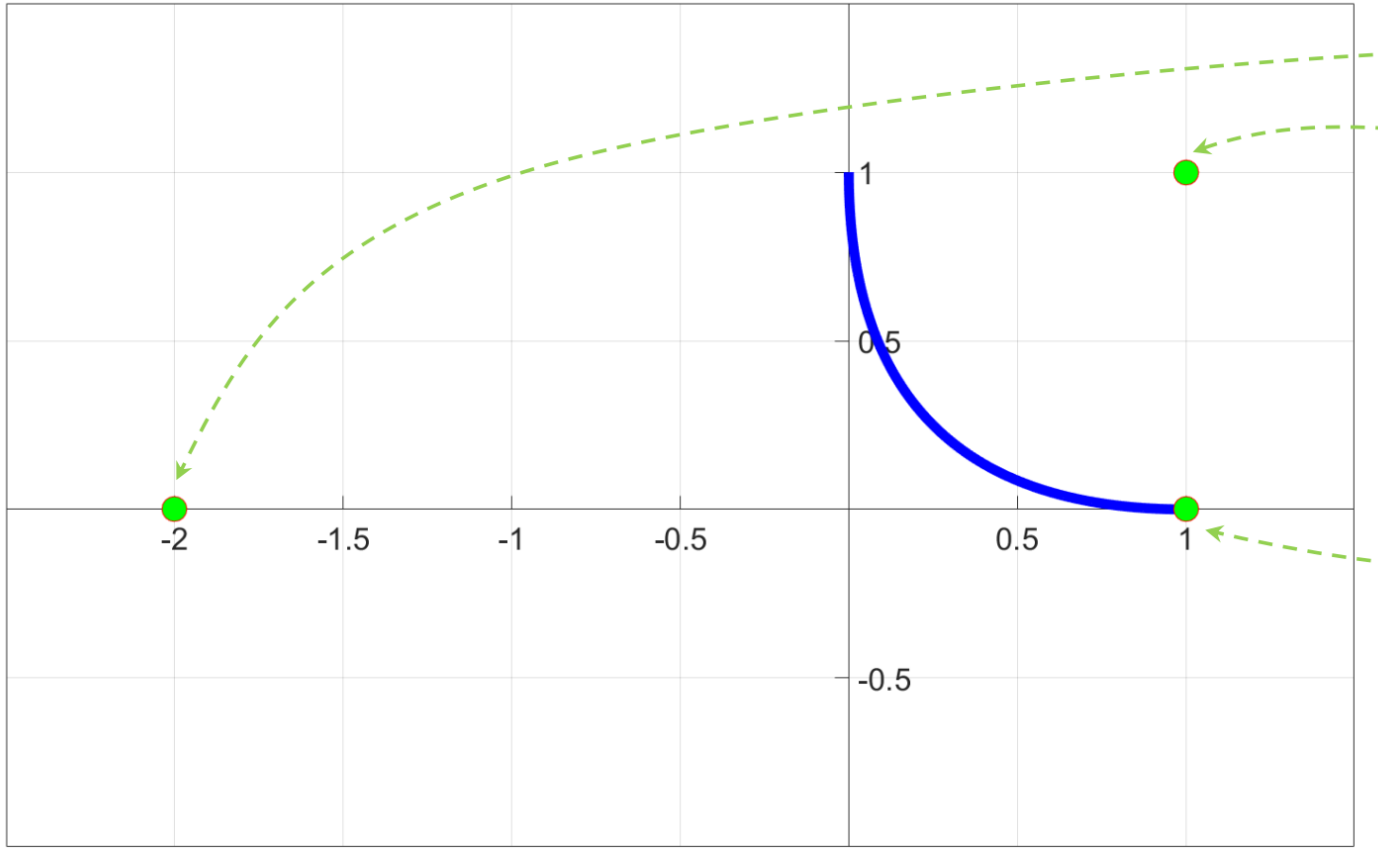
- Example of a parabola: $\boldsymbol{f}(t) = \boldsymbol{a}t^2 + \boldsymbol{b}t + \boldsymbol{c}$




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

# Modeling with the power basis, … no thanks!

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

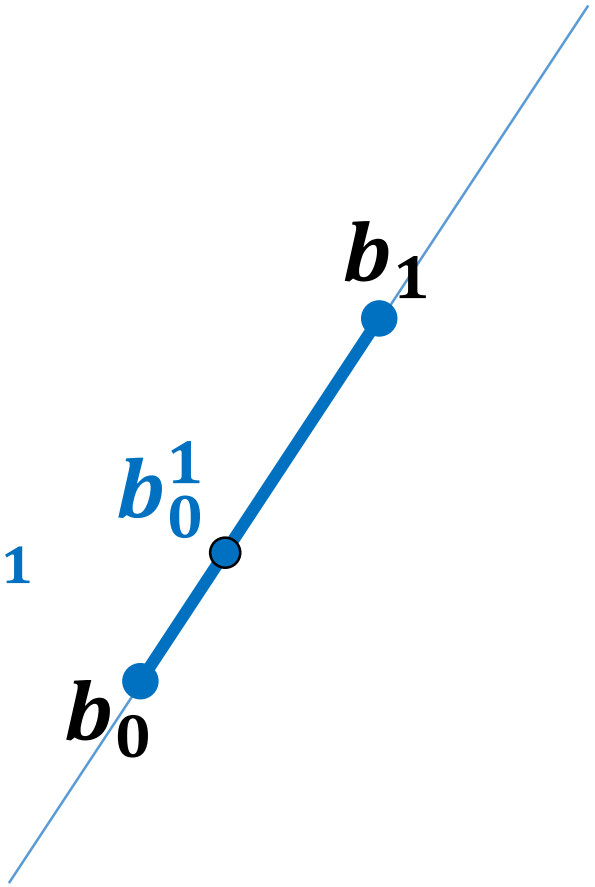



$$\mathbf{f}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} t^2 + \begin{pmatrix} -2 \\ 0 \end{pmatrix} t + \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

# Back to the drawing board

- A point on a parametric line

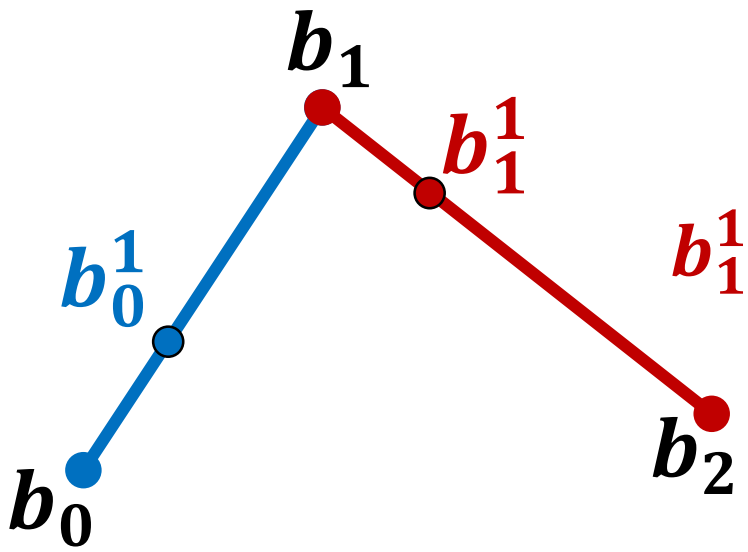
$$\boldsymbol{b_0^1} = (1 - t)\boldsymbol{b_0} + t\boldsymbol{b_1}$$



# Back to the drawing board

- Another point on a second parametric line

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

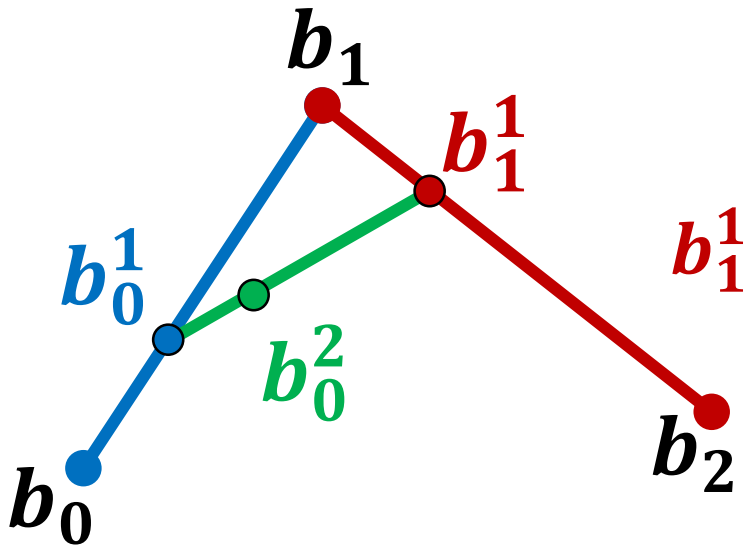



$$b_1^1 = (1-t)b_1 + tb_2$$

# Back to the drawing board

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

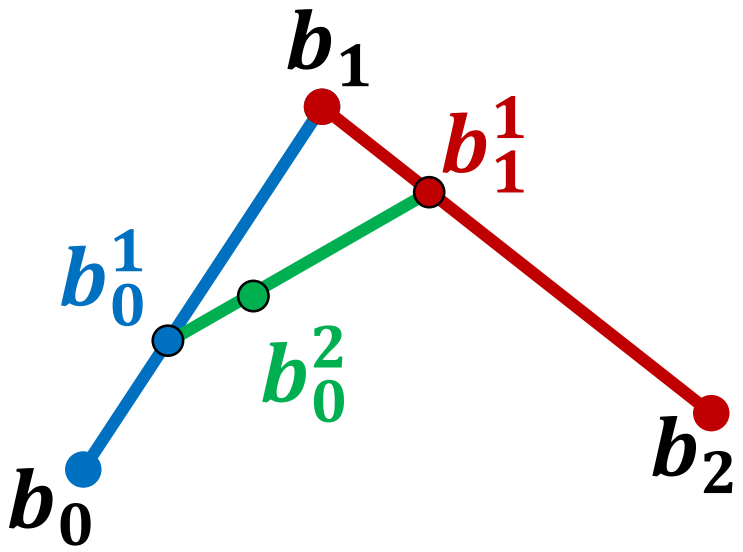
$$b_0^1 = (1-t)b_0 + tb_1$$




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

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

# Back to the drawing board




- And then simplify…

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

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

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

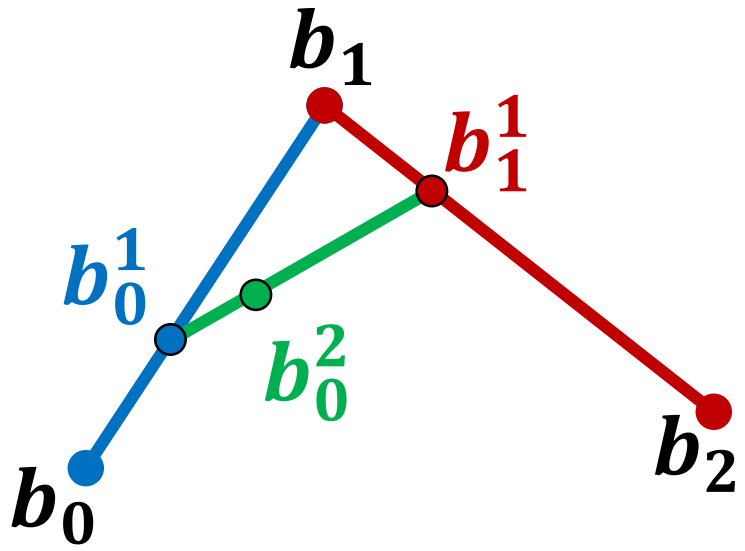
$$b_0^2 = (1-t)[(1-t)b_0 + tb_1] + t[(1-t)b_1 + tb_2]$$

$$\boxed{b_0^2 = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2}$$

# Back to the drawing board

- We obtained another description of parabolic curves

- The coefficients $\boldsymbol{b_0, b_1, b_2}$ have a geometric meaning




$$\boldsymbol{b_0^2} = (1-t)^2\boldsymbol{b_0} + 2t(1-t)\boldsymbol{b_1} + t^2\boldsymbol{b_2}$$

# Example re-visited

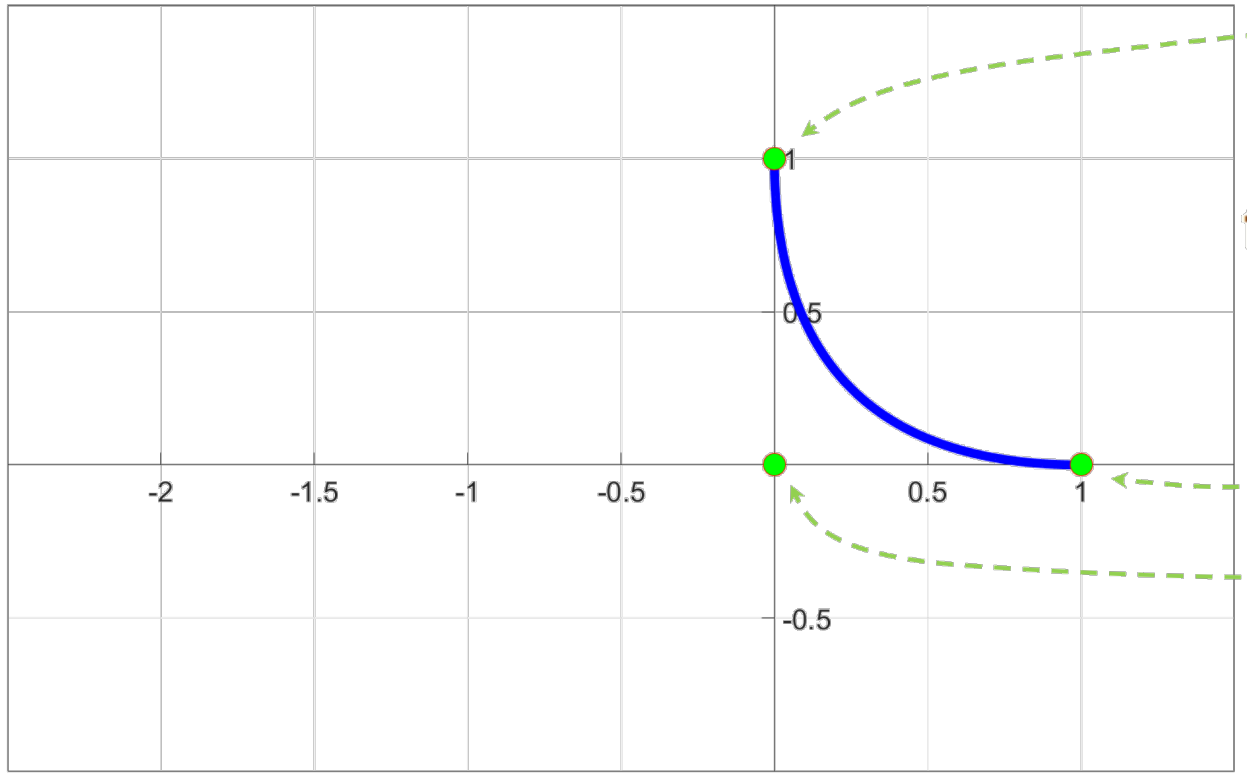
- Let's rewrite our initial parabolic curve example in the new basis

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

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

# Example re-visited

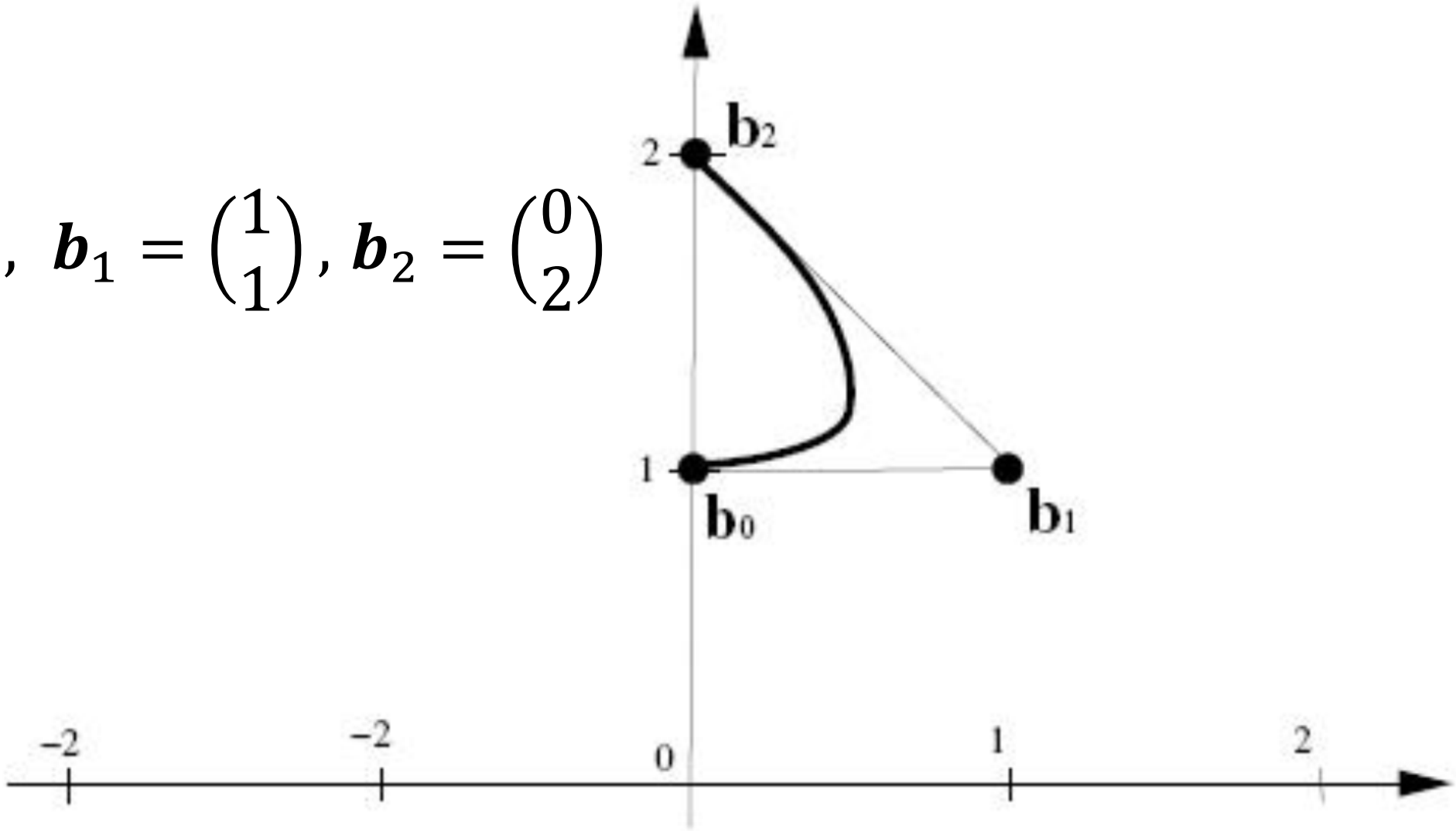
- The coefficient have a geometric meaning
- More intuitive for curve manipulation




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

# Another example

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



# Going further

- Cubic approximation
- Given 4 points: $\boldsymbol{p}_0^0(t) = \boldsymbol{p}_0, \;\; \boldsymbol{p}_1^0(t) = \boldsymbol{p}_1, \;\; \boldsymbol{p}_2^0(t) = \boldsymbol{p}_2, \;\; \boldsymbol{p}_3^0(t) = \boldsymbol{p}_3$
- First iteration

$$\boldsymbol{p}_0^1 = (1-t)\boldsymbol{p}_0 + t\boldsymbol{p}_1$$

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

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

- 2nd iteration

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

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

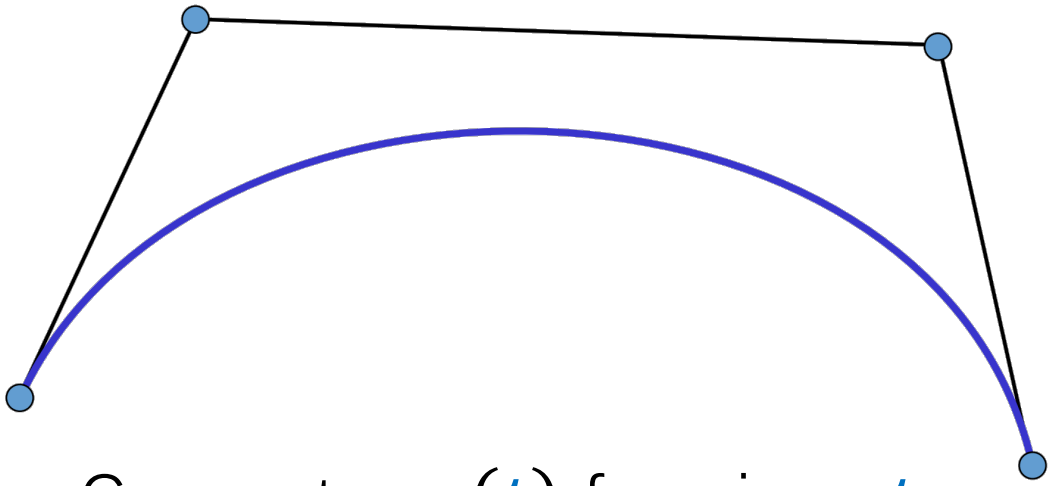
- Curve

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

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

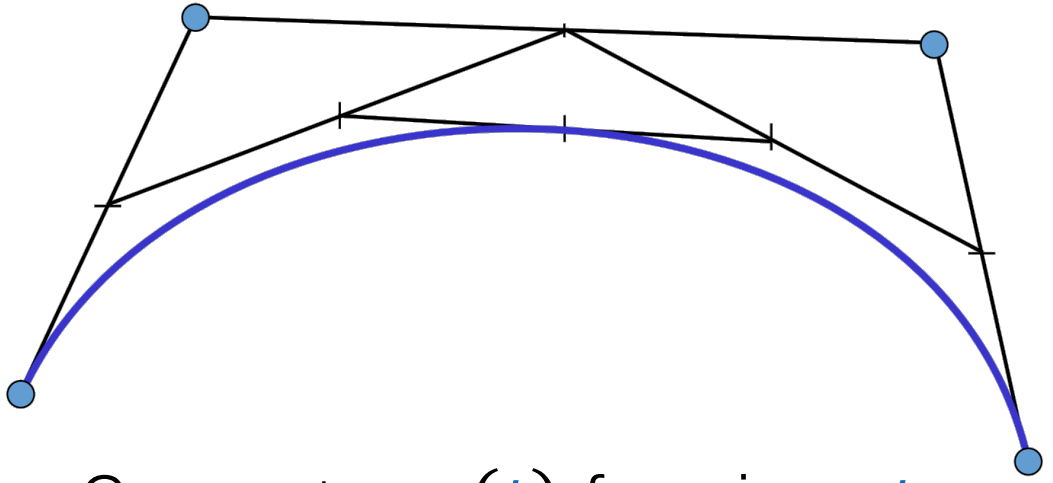
Now let's examine it more closely …

# De Casteljau algorithm



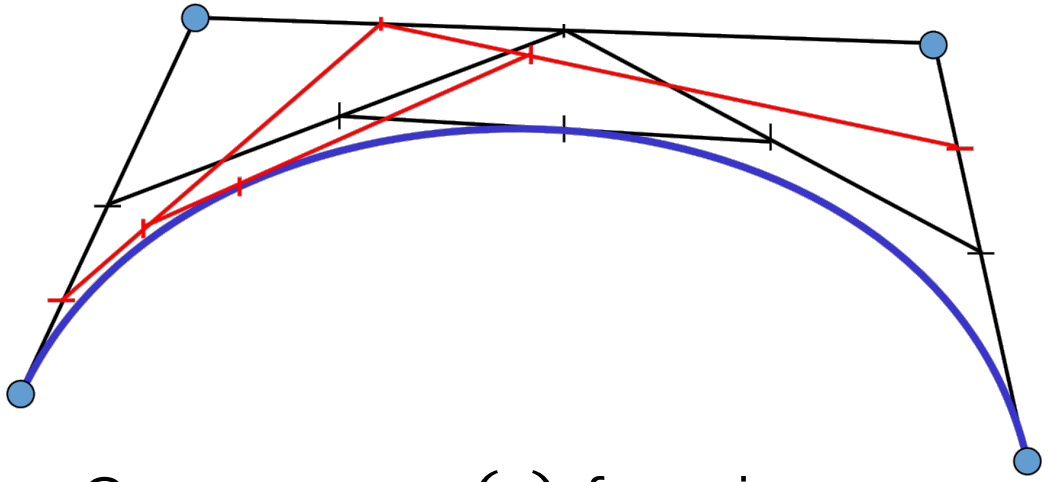
- De Casteljau Algorithm: Computes $x(t)$ for given $t$
  - Bisect control polygon in ratio $t:(1-t)$
  - Connect the new dots with lines (adjacent segments)
  - Interpolate again with the same ratio
  - Iterate, until only one points is left

# De Casteljau algorithm



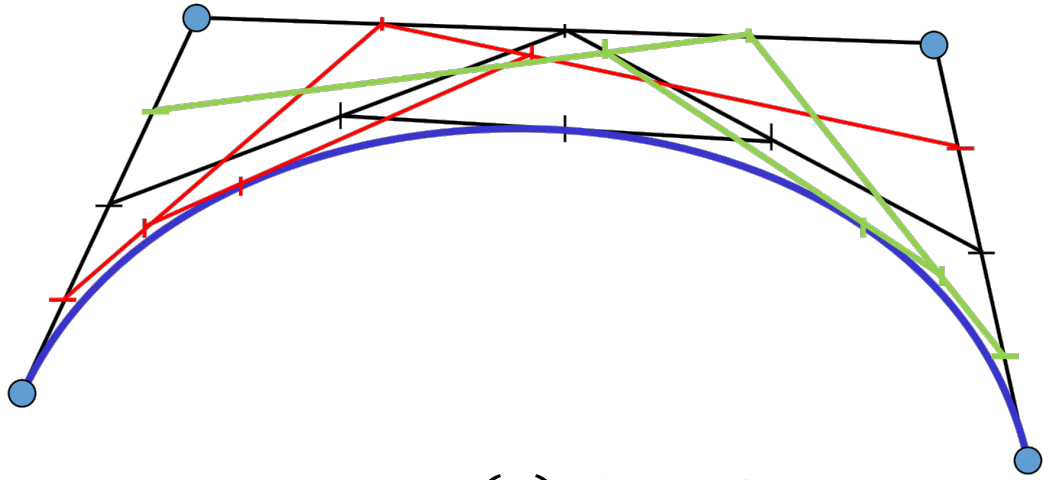
- De Casteljau Algorithm: Computes $x(t)$ for given $t$
  - Bisect control polygon in ratio $t:(1-t)$
  - Connect the new dots with lines (adjacent segments)
  - Interpolate again with the same ratio
  - Iterate, until only one points is left

# 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 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 points is left

# De Casteljau algorithm

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

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

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

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

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

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

# De Casteljau algorithm

- Repeated convex combination of control points

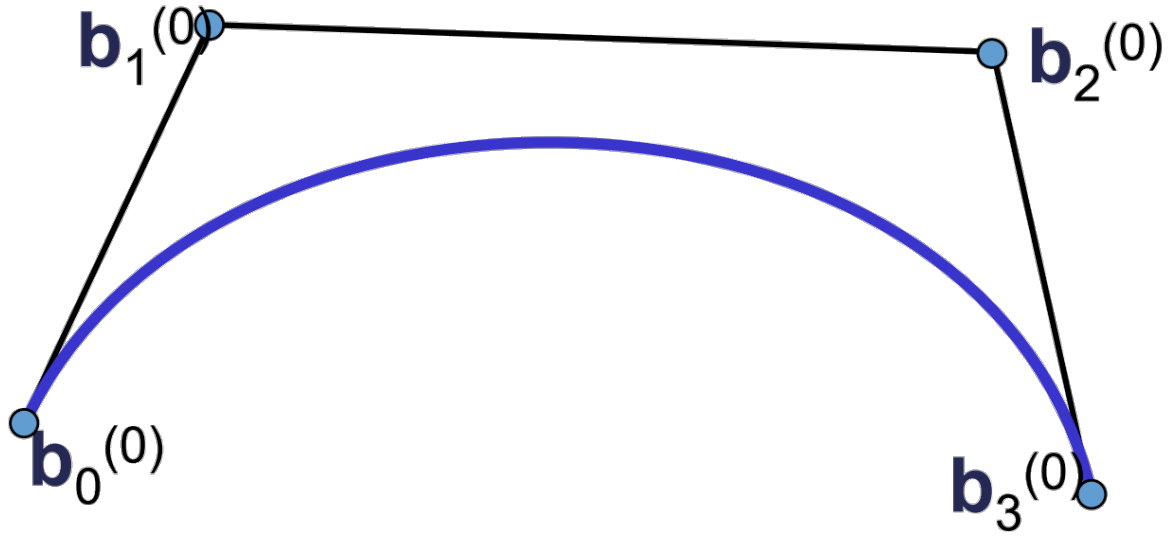
$$\boldsymbol{b}_i^{(r)} = (1-t)\boldsymbol{b}_i^{(r-1)} + t\boldsymbol{b}_{i+1}^{(r-1)}$$

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

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

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

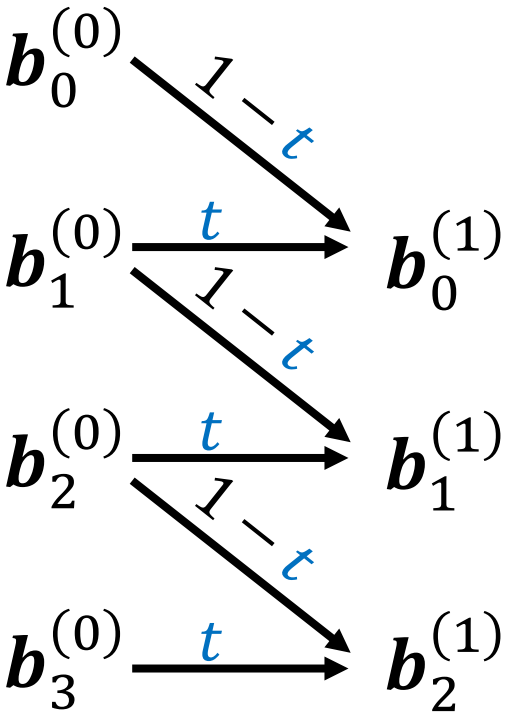
$\boldsymbol{b}_3^{(0)}$

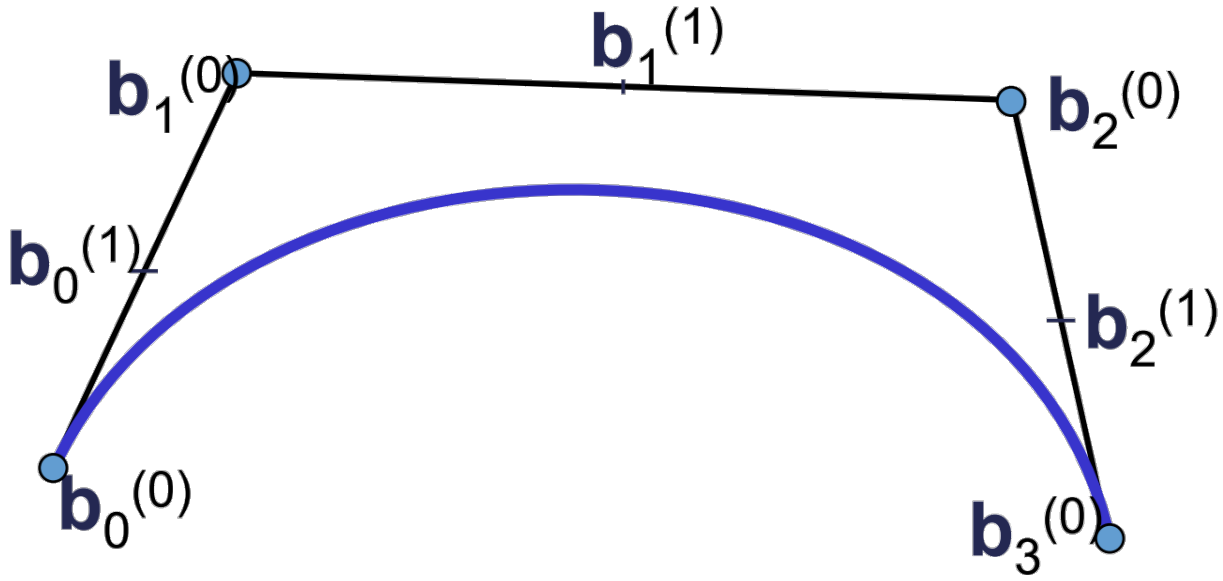


# De Casteljau algorithm

- Repeated convex combination of control points

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

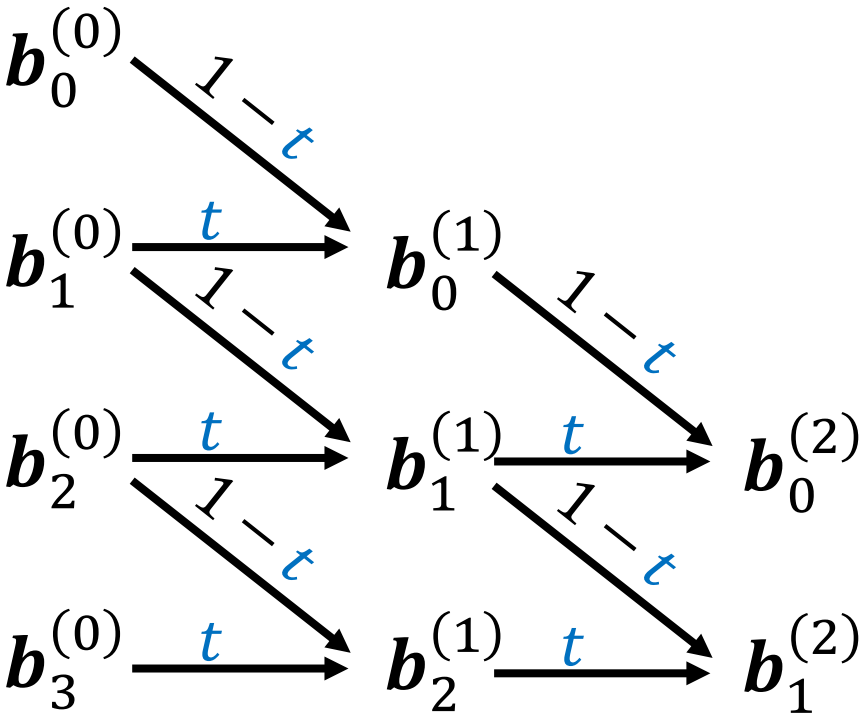


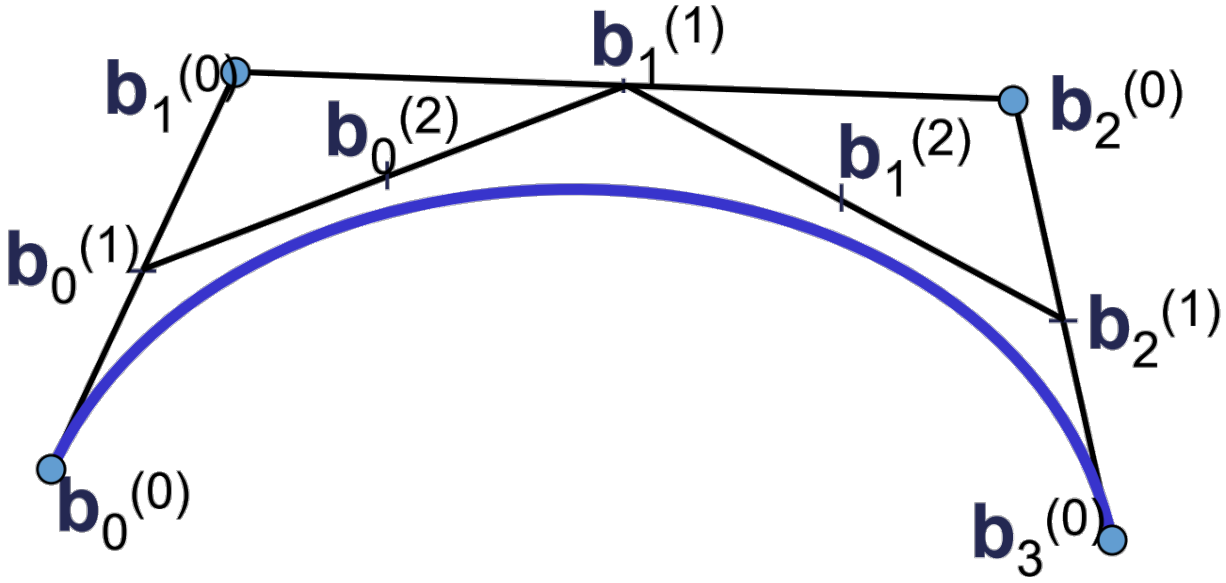



# De Casteljau algorithm

- Repeated convex combination of control points

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

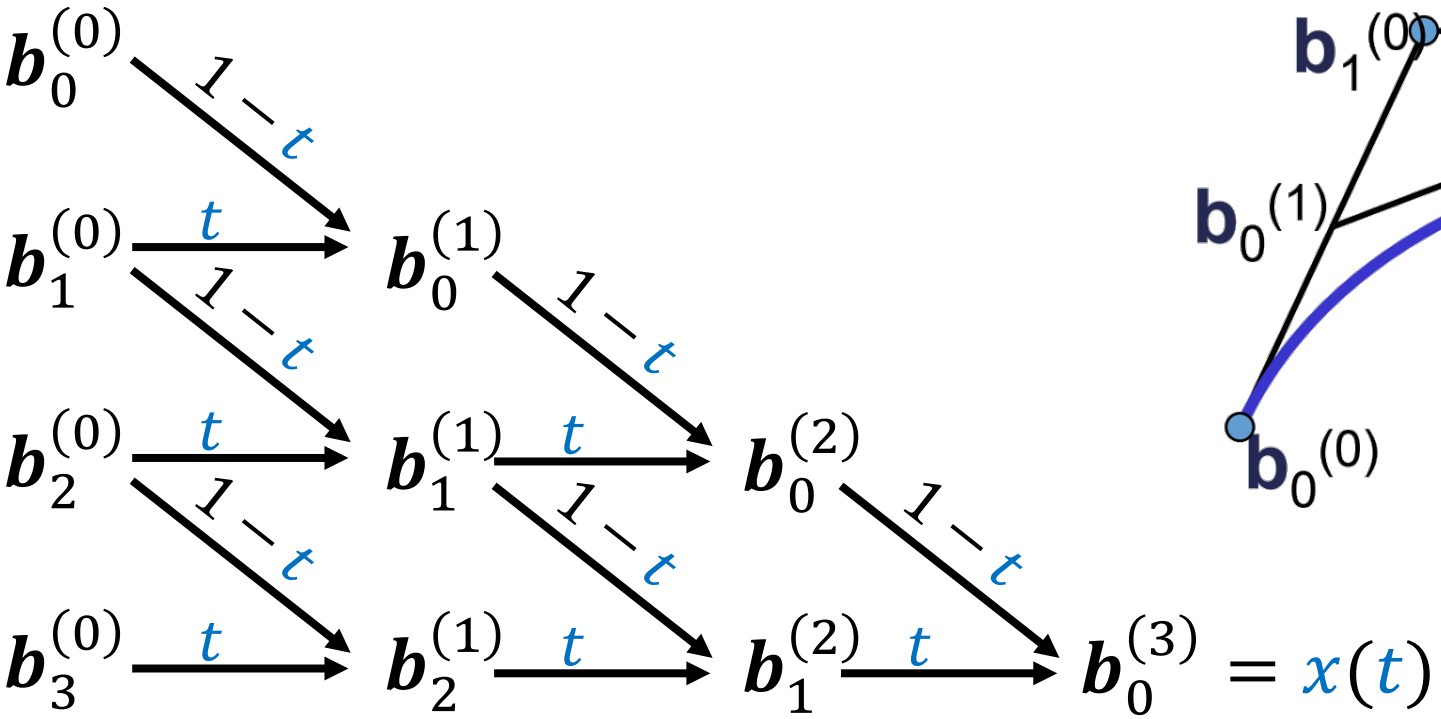


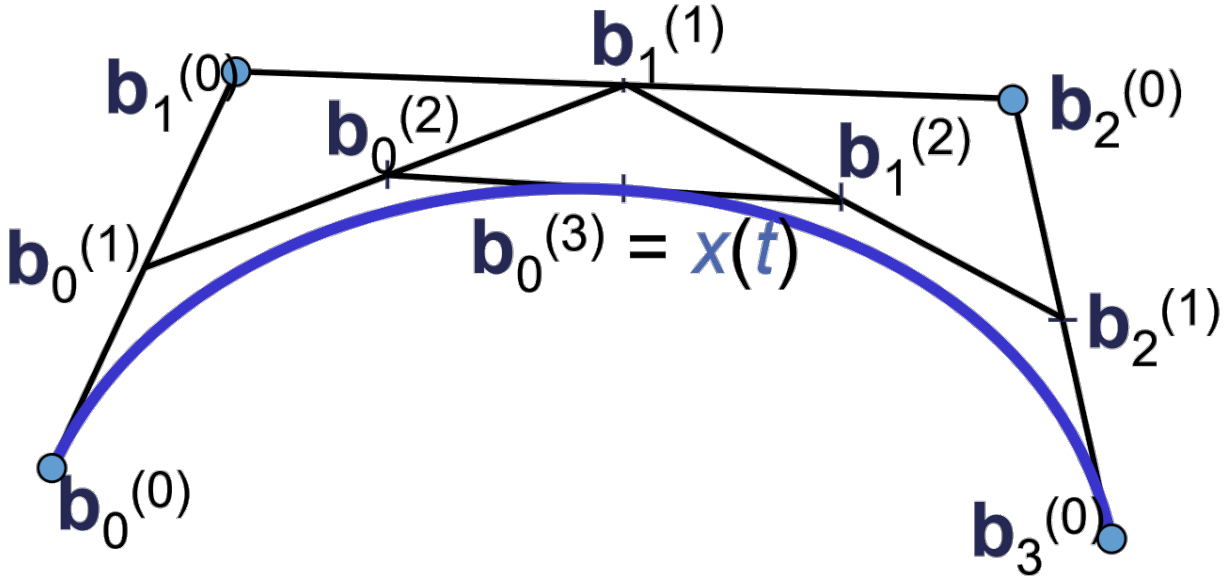
# De Casteljau algorithm

- Repeated convex combination of control points

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




De Casteljau scheme



# De Casteljau algorithm

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

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

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

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

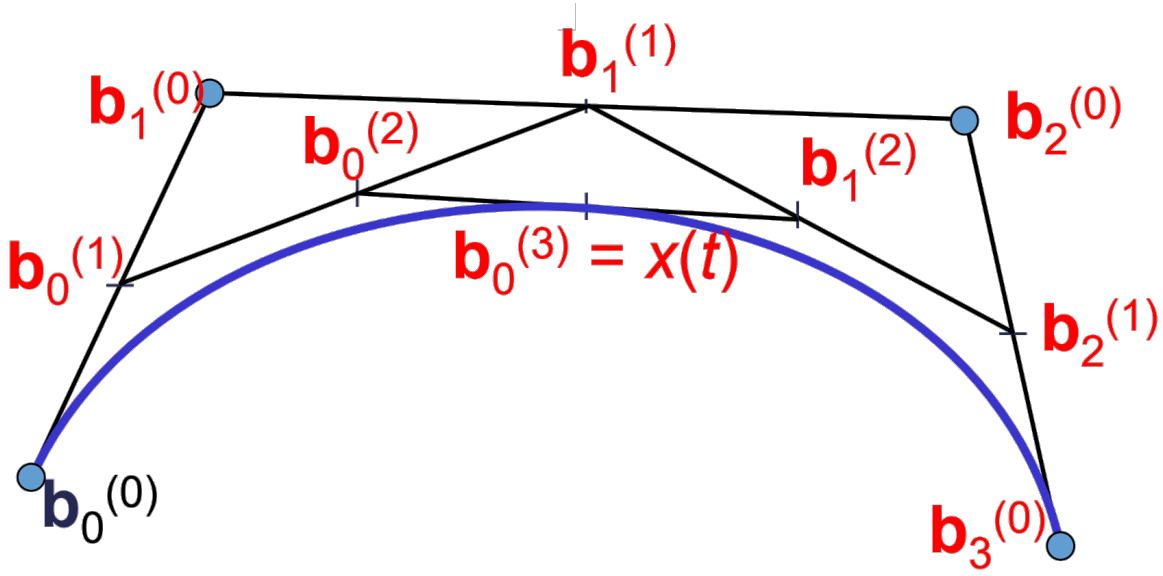
$$\boldsymbol{b}_3 = \boldsymbol{b}_3^0 \quad \boldsymbol{b}_2^1 \quad \boldsymbol{b}_1^2 \quad \boldsymbol{b}_0^3$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\boldsymbol{b}_{n-1} = \boldsymbol{b}_{n-1}^0 \quad \boldsymbol{b}_{n-2}^1 \quad \ldots \quad \boldsymbol{b}_0^{n-1}$$

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

# De Casteljau algorithm




**Algorithm:**

for r=1..n

  for i=0..n-r

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

  end

end

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

**The whole algorithm consists only of repeated linear interpolations.**

# De Casteljau algorithm: Properties

- The polygon consisting of the points $\boldsymbol{b_0, \ldots, b_n}$ is called Bezier polygon (control polygon)
- The points $\boldsymbol{b_i}$ are called Bezier points (control points)
- The curve defined by the Bezier points $\boldsymbol{b_0, \ldots, b_n}$ and the de Casteljau algorithm is called Bezier 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 Bezier points

# De Casteljau algorithm: Properties

- **Properties of Bezier curves:**
  - Given: Bezier points $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n$

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

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

# De Casteljau algorithm: Properties

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

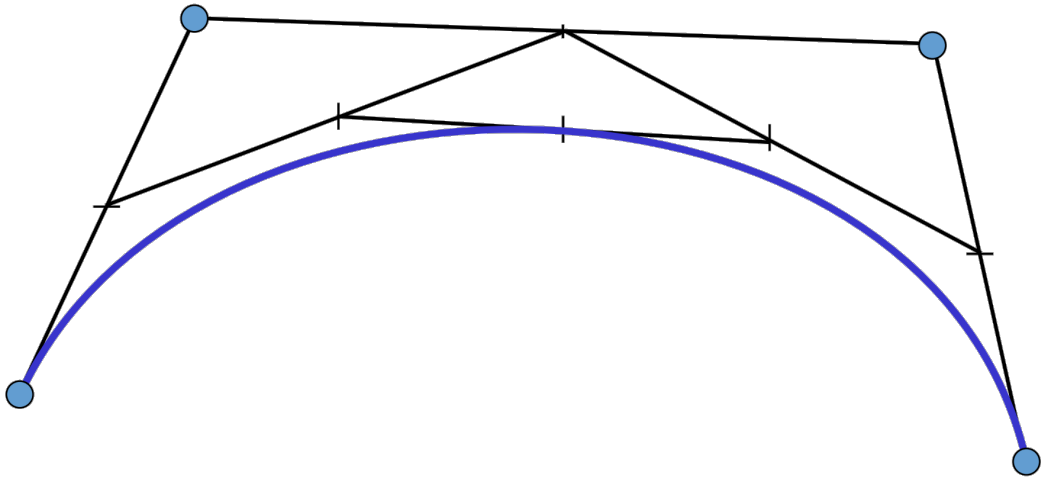
# De Casteljau algorithm: Properties

- **Symmetry**
  - The following two Bezier curves coincide, they are only traversed in opposite directions:

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

- **Linear Precision:**
  - Bezier curve is line segment, if $\boldsymbol{b}_0, \dots, \boldsymbol{b}_n$ are colinear
- Invariance under barycentric combinations

# Recap

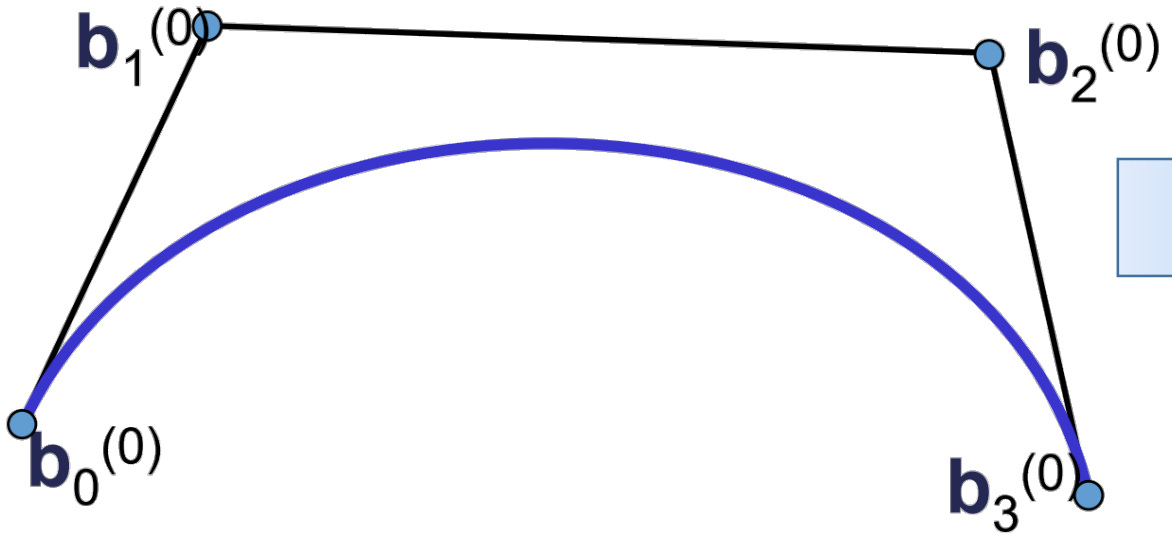


de Casteljau algorithm

# Bezier Curves

## Towards a polynomial description

# Bezier Curves
# Towards a polynomial description




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

# Polynomial description of Bezier curves

- The same problem as before:
  - Given: $(n+1)$ control points $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n$
  - Wanted: Bezier curve $\boldsymbol{x}(t)$ 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:
    - Appling an affine map $\boldsymbol{x} \rightarrow A\boldsymbol{x} + b$ 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

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

# Affine Invariance

- Affine map: $\boldsymbol{x} \rightarrow A\boldsymbol{x} + \boldsymbol{b}$
- **Part I:** Linear invariance – we get this automatically
  - Linear approach: $\boldsymbol{f}(t) = \sum_{i=1}^{n} b_i(t)\boldsymbol{p}_i = \sum_{i=1}^{n} b_i(t) \begin{pmatrix} p_i^{(x)} \\ p_i^{(y)} \\ p_i^{(z)} \end{pmatrix}$
  - Therefore: $A\big(\boldsymbol{f}(t)\big) = A(\sum_{i=1}^{n} b_i(t)\boldsymbol{p}_i) = \sum_{i=1}^{n} b_i(t)(A\boldsymbol{p}_i)$

# Affine Invariance

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

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

    - 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 $\boldsymbol{p}_i$
    - This is very important for modeling

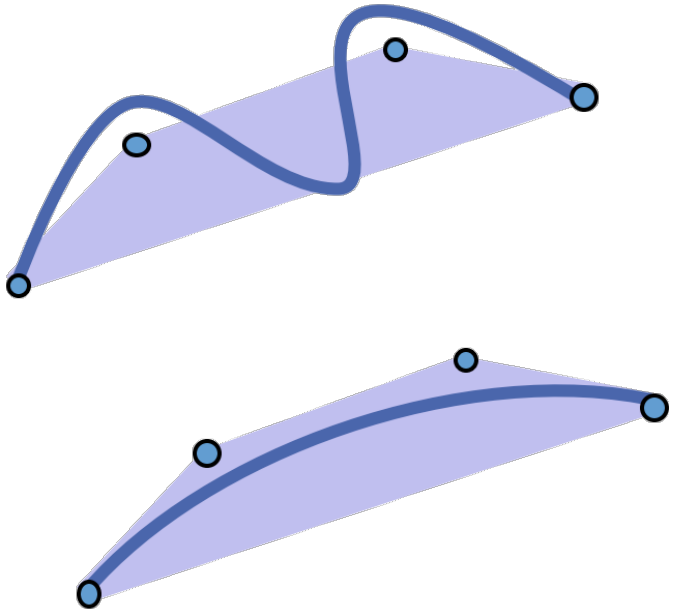
# Convex Hull Property

- Convex combinations:
  - A convex combination of a set of points $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ is any point of the form:

$$\sum_{i=1}^{n} \lambda_i \boldsymbol{p_i} \text{ with } \sum_{i=1}^{n} \lambda_i = 1 \text{ and } \forall i = 1 \ldots n: 0 \leq \lambda_i \leq 1$$

    - (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 $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ and every complete straight line between two elements of the set

# Convex Hull Property



- Accordingly:
  - If we have this property

    $\forall t \in \Omega: \sum_{i=1}^{n} b_i(t) = 1$ and $\forall t \in \Omega, \forall i: b_i(t) \geq 0$

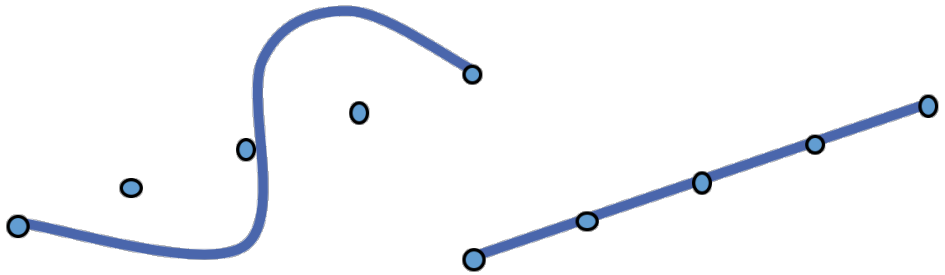
    the constructed curves / surfaces will be:
    - Affine invariant (translations, linear maps)
    - Be restricted to the convex hull of the control points

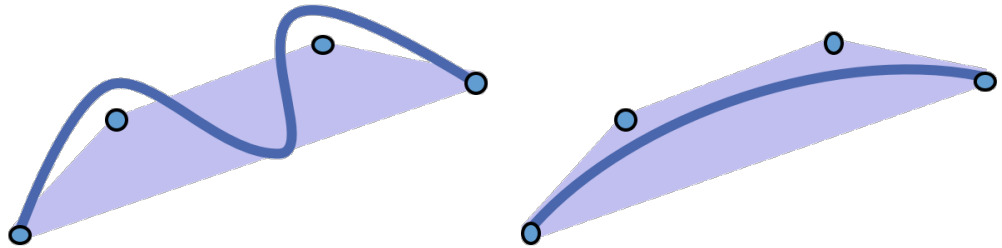
  - Corollary: Curves will have *linear precision*
    - All control points lie on a straight line

    ⇒ 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 Bezier curves

- The same problem as before:
  - Given: $(n+1)$ control points $\boldsymbol{b}_0, \dots, \boldsymbol{b}_n$
  - Wanted: Bezier curve $x(t)$ 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 Bezier curve:

$$B_0^n(t), \dots, B_n^n(t) \text{ over } [0,1]$$

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

# Bernstein Basis

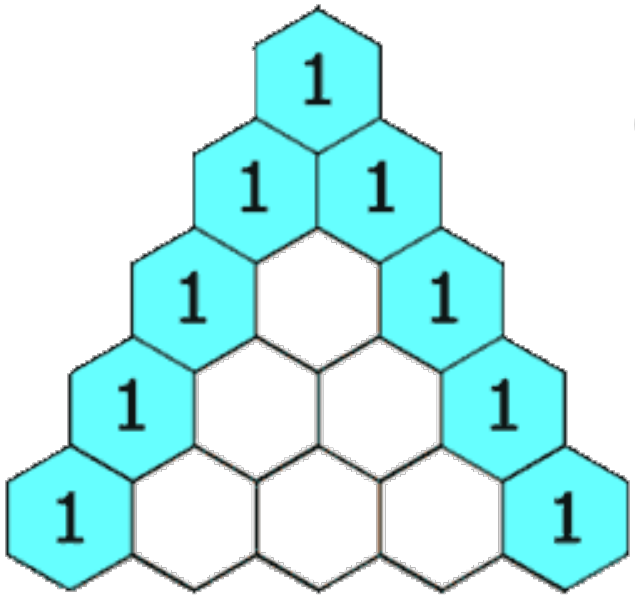
- Let's examine the Bernstein basis: $B = \{B_0^{(n)}, B_1^{(n)}, \ldots, B_n^{(n)}\}$
  - Bernstein basis of degree $n$:

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

where the binomial coefficients are given by:

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

# Binomial Coefficients and Theorem




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

$$\begin{array}{ccccccccccc}
 & & & & 1 & & & & & & \\
 & & & 1 & & 1 & & & & & \\
 & & 1 & & 2 & & 1 & & & & \\
 & 1 & & 3 & & 3 & & 1 & & & \\
1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 5 & & 10 & & 10 & & 5 & & 1 &
\end{array}$$

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

# Examples: The first few

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

- The first three Bernstein bases:

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

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

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

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

# Examples: The first few

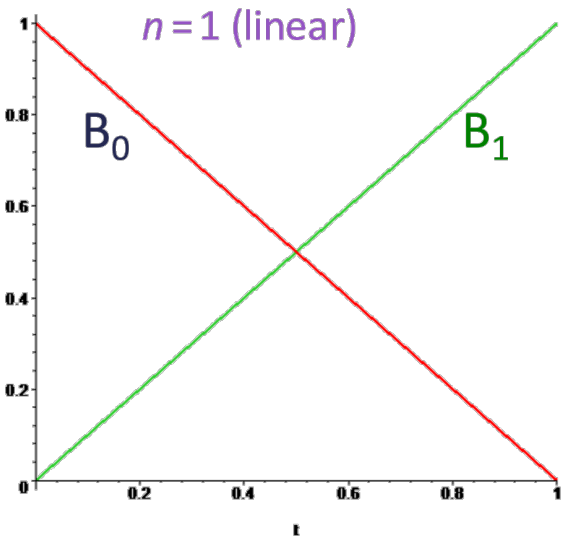
$$B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

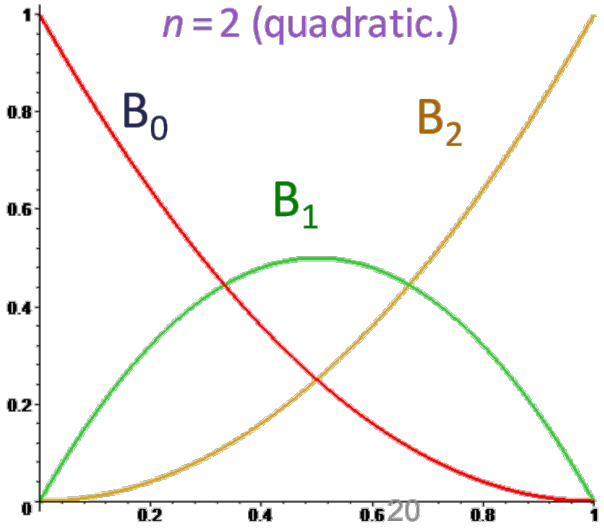
$$B_0^{(0)} := 1$$

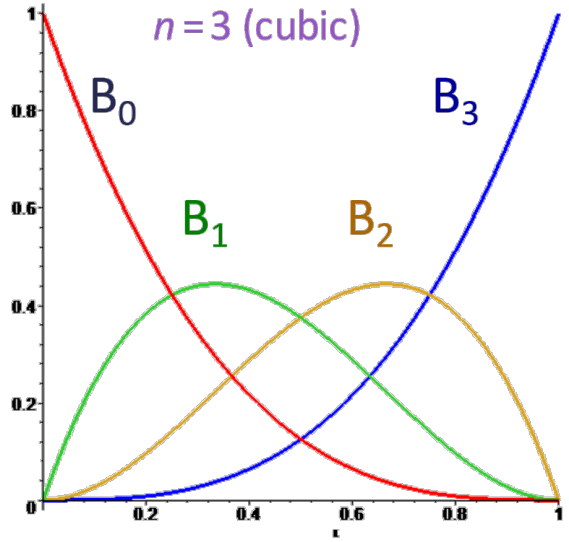
$$B_0^{(1)} := 1 - t$$
$$B_1^{(1)} := t$$

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

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



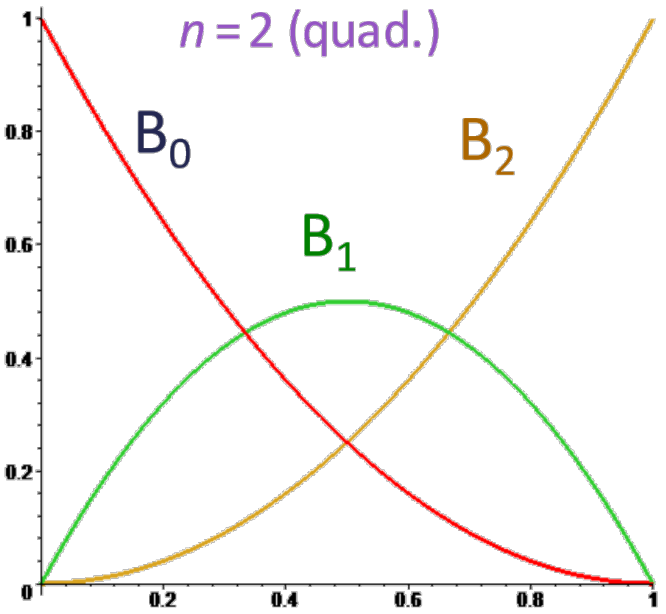



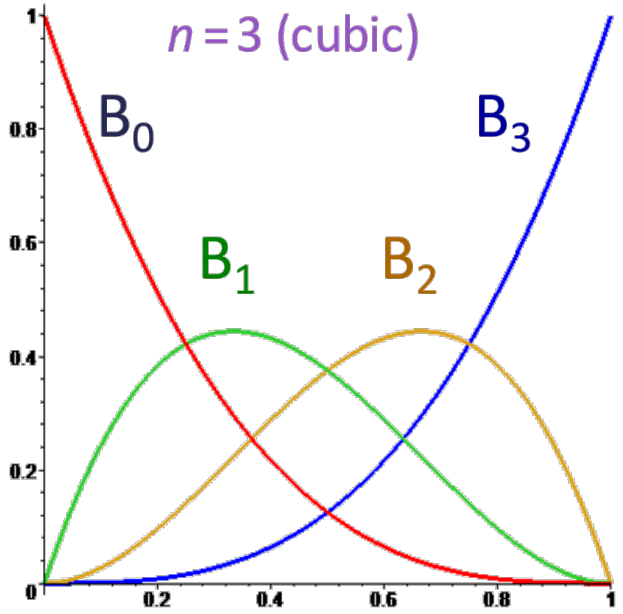



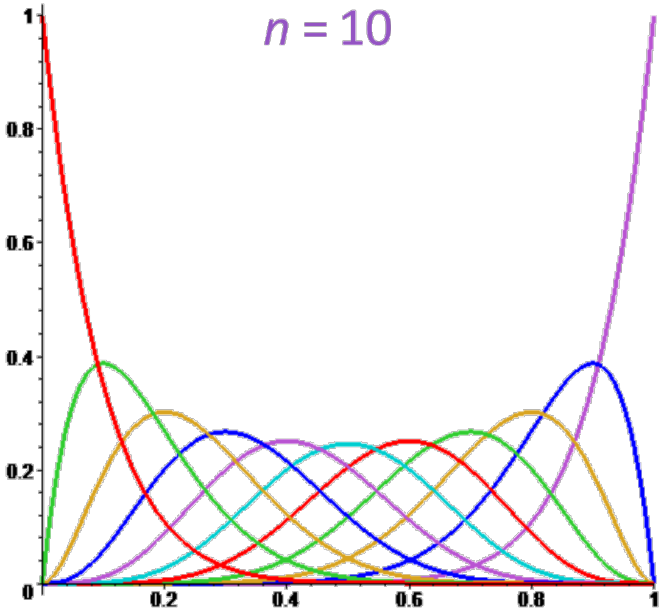
# Bernstein Basis

- Bezier curves use the Bernstein basis: $B = \left\{B_0^{(n)}, B_1^{(n)}, \ldots, B_n^{(n)}\right\}$
  - 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

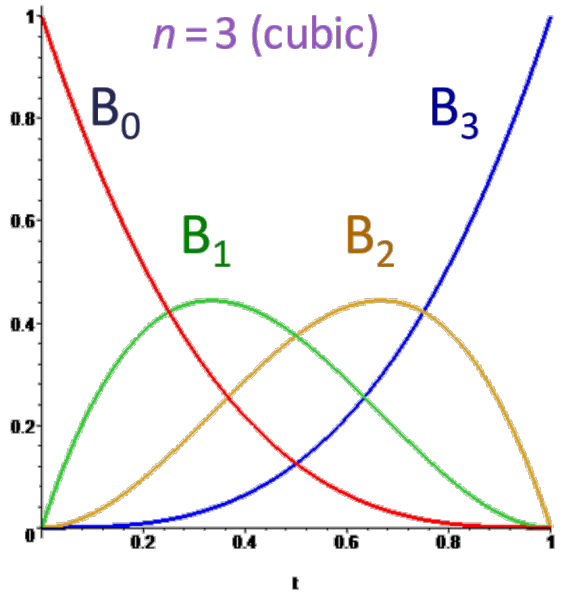
- $B = \left\{B_0^{(n)}, B_1^{(n)}, \ldots, B_n^{(n)}\right\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

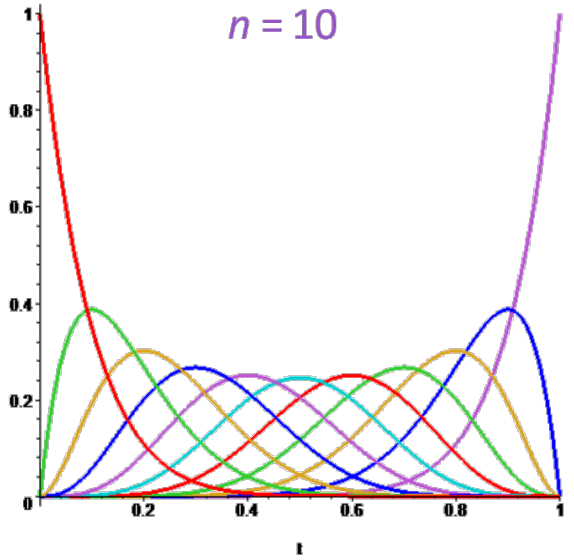
- Basis for polynomials of degree $n$

Smoothness

- Each basis function $B_i^{(n)}$ has its maximum at $t = \frac{i}{n}$

Local control (semi-local)



# Bernstein Basis: Properties

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

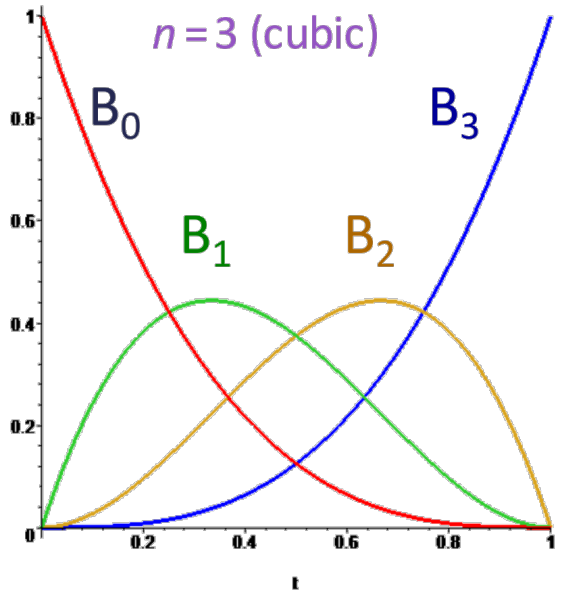
Affine invariance

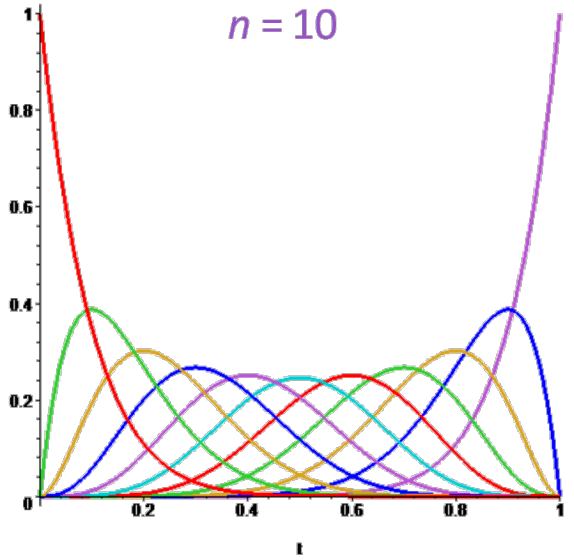
Convex hull property

- Partition of unity (binomial theorem)

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

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



# 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

- $B=\left\{B_0^{(n)},B_1^{(n)},\ldots,B_n^{(n)}\right\}$, $B_i^{(n)}(t)=\binom{n}{i}t^i(1-t)^{n-i}$
- Recursive computation

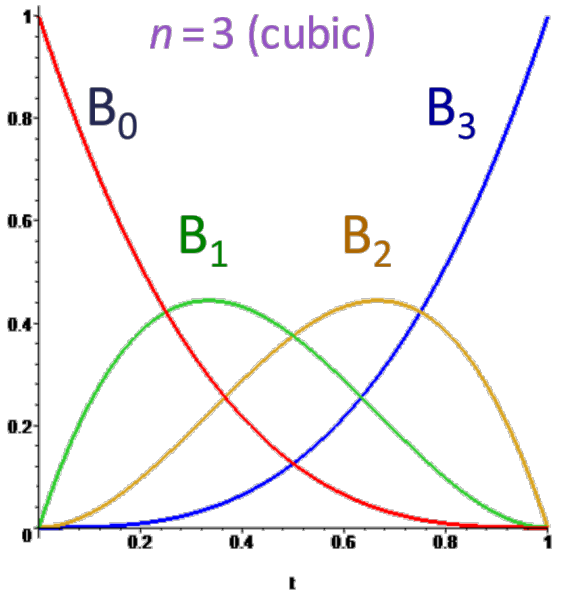
$$B_i^n(t):=(1-t)B_i^{(n-1)}(t)+tB_{i-1}^{(n-1)}(1-t)$$

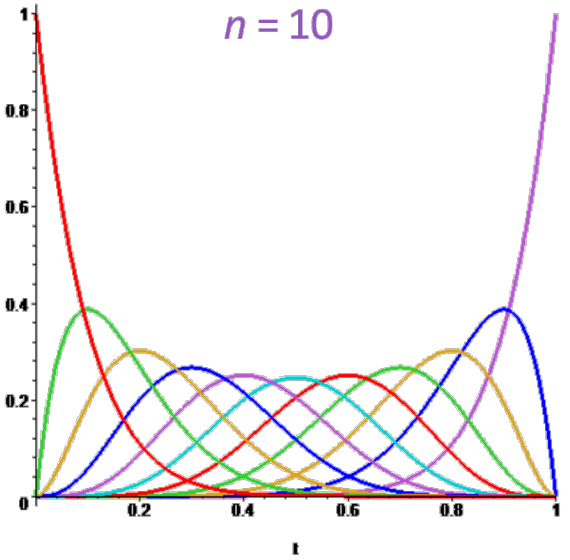
with $B_0^0(t)=1, B_i^n(t)=0$ for $i\notin\{0\ldots n\}$

- 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

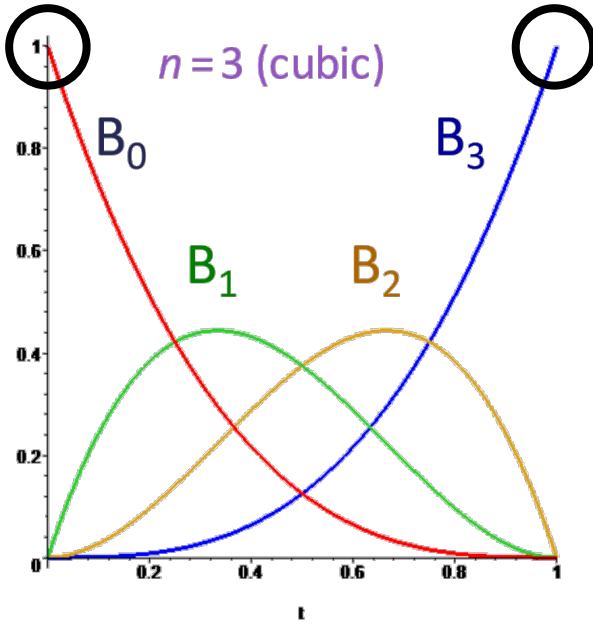
- $B = \left\{B_0^{(n)}, B_1^{(n)}, \ldots, B_n^{(n)}\right\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

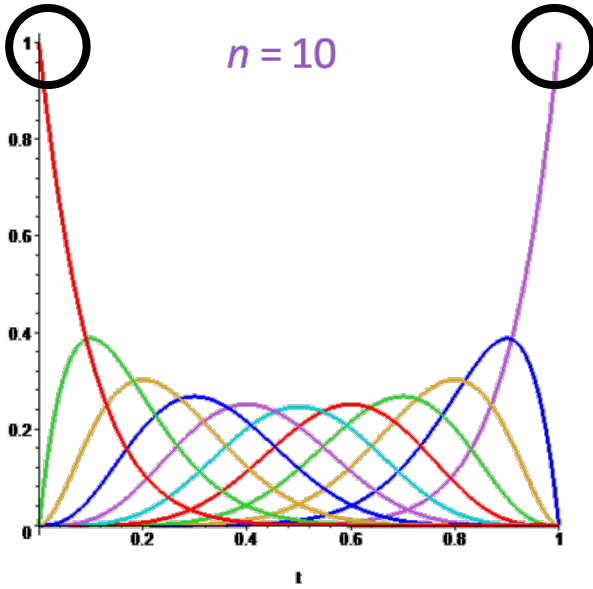
- Non-negativity II

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

$$B_0^n(0) = 1, \qquad B_1^n(0) = \cdots = B_n^n(0) = 0$$

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



# 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:

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

**(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:

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

$$= n \left[ (n-1) \left( B_{i-2}^{(n-2)}(t) - B_{i-1}^{(n-2)}(t) \right) - (n-1) \left( B_{i-1}^{(n-2)}(t) - B_i^{(n-2)}(t) \right) \right]$$

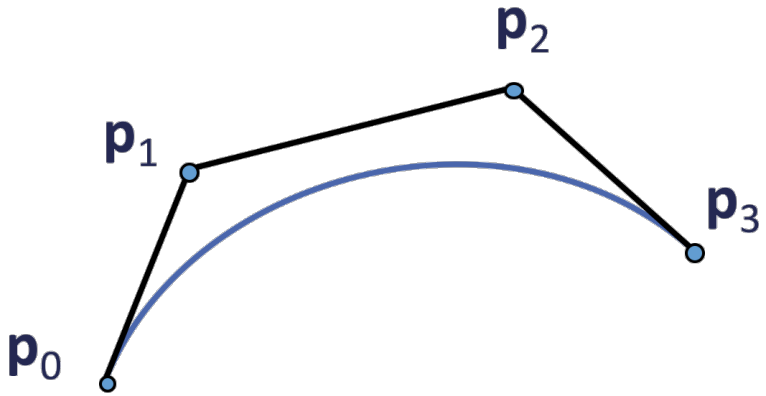
$$= n(n-1) \left[ B_{i-2}^{(n-2)}(t) - 2B_{i-1}^{(n-2)}(t) + B_i^{(n-2)}(t) \right]$$

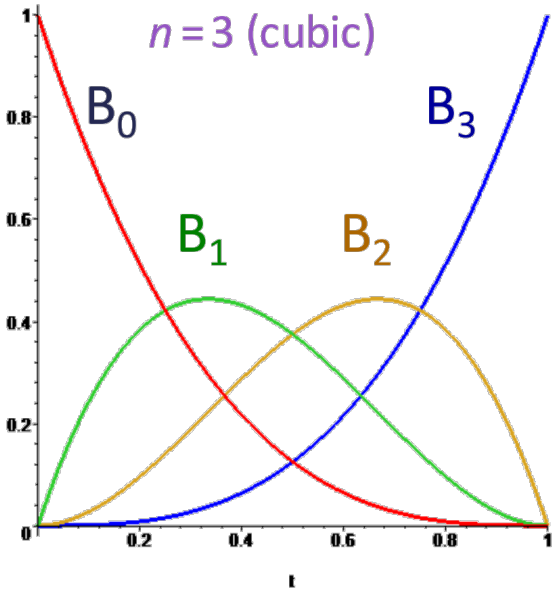
**(Notation: $\{k\} = k$ if $k > 0$, zero otherwise)**

# Bezier Curves in Bernstein form

- **Bezier Curves:**

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

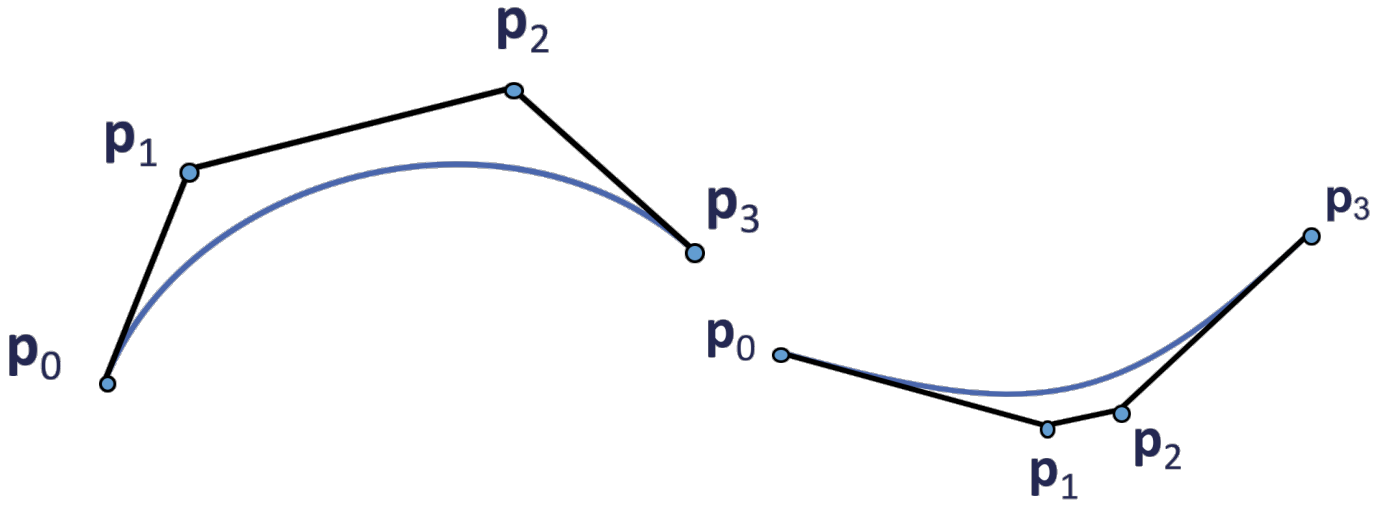


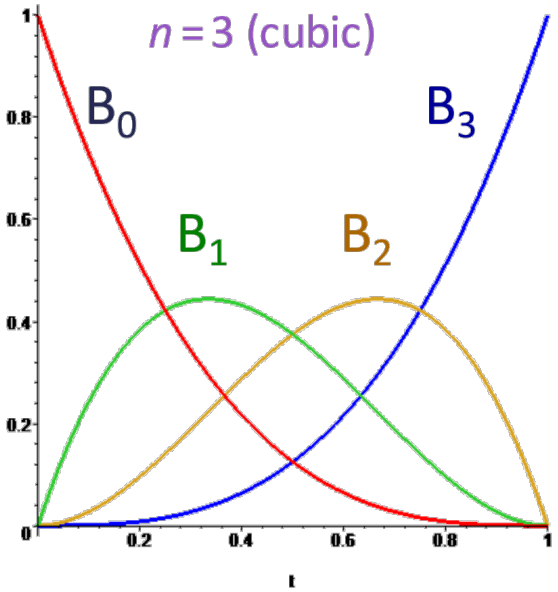



# Bezier Curves in Bernstein form

- **Bezier Curves:**

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

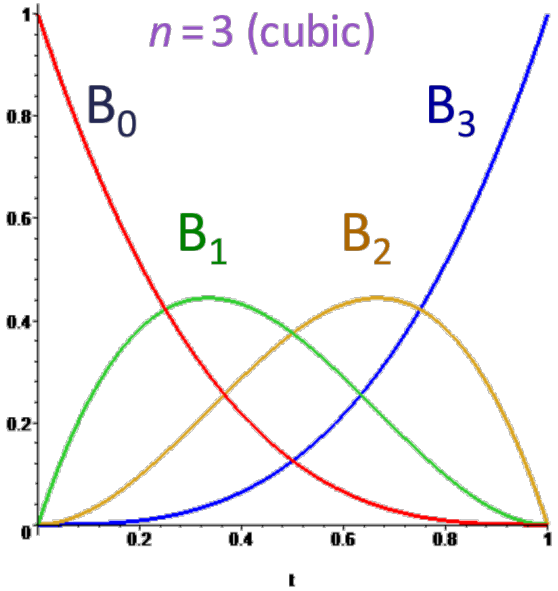


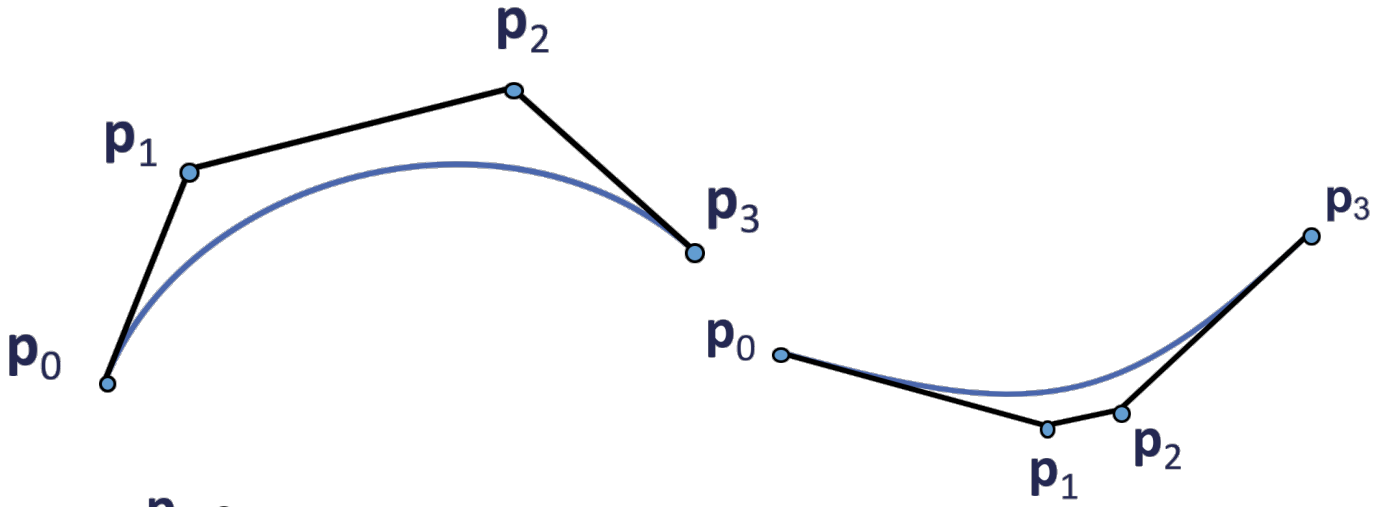



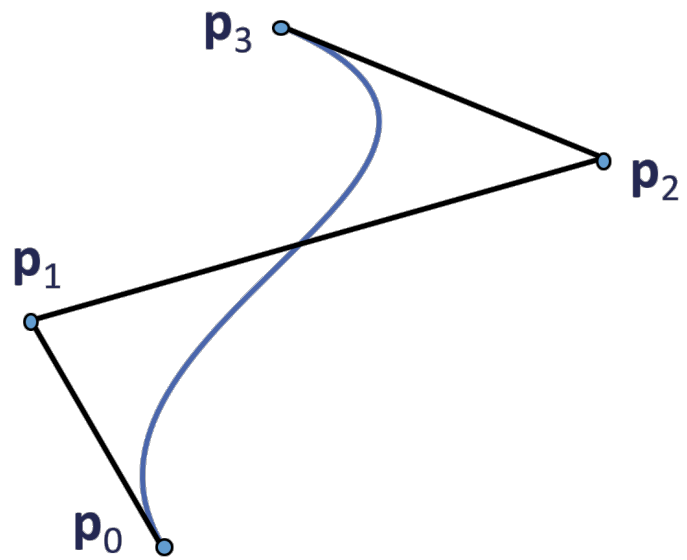
# Bezier Curves in Bernstein form

- **Bezier Curves:**

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



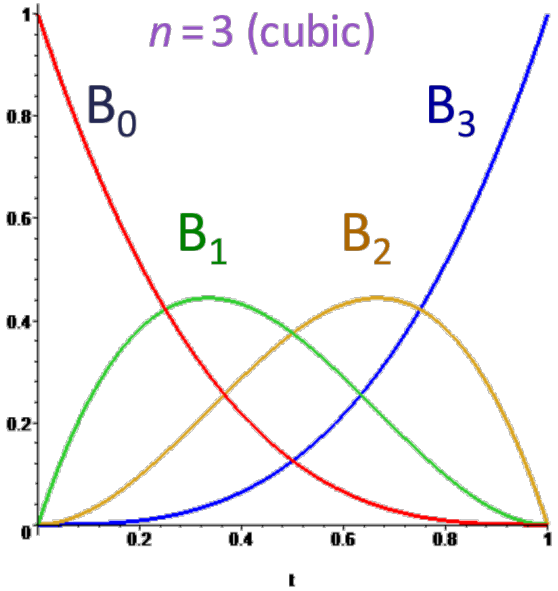



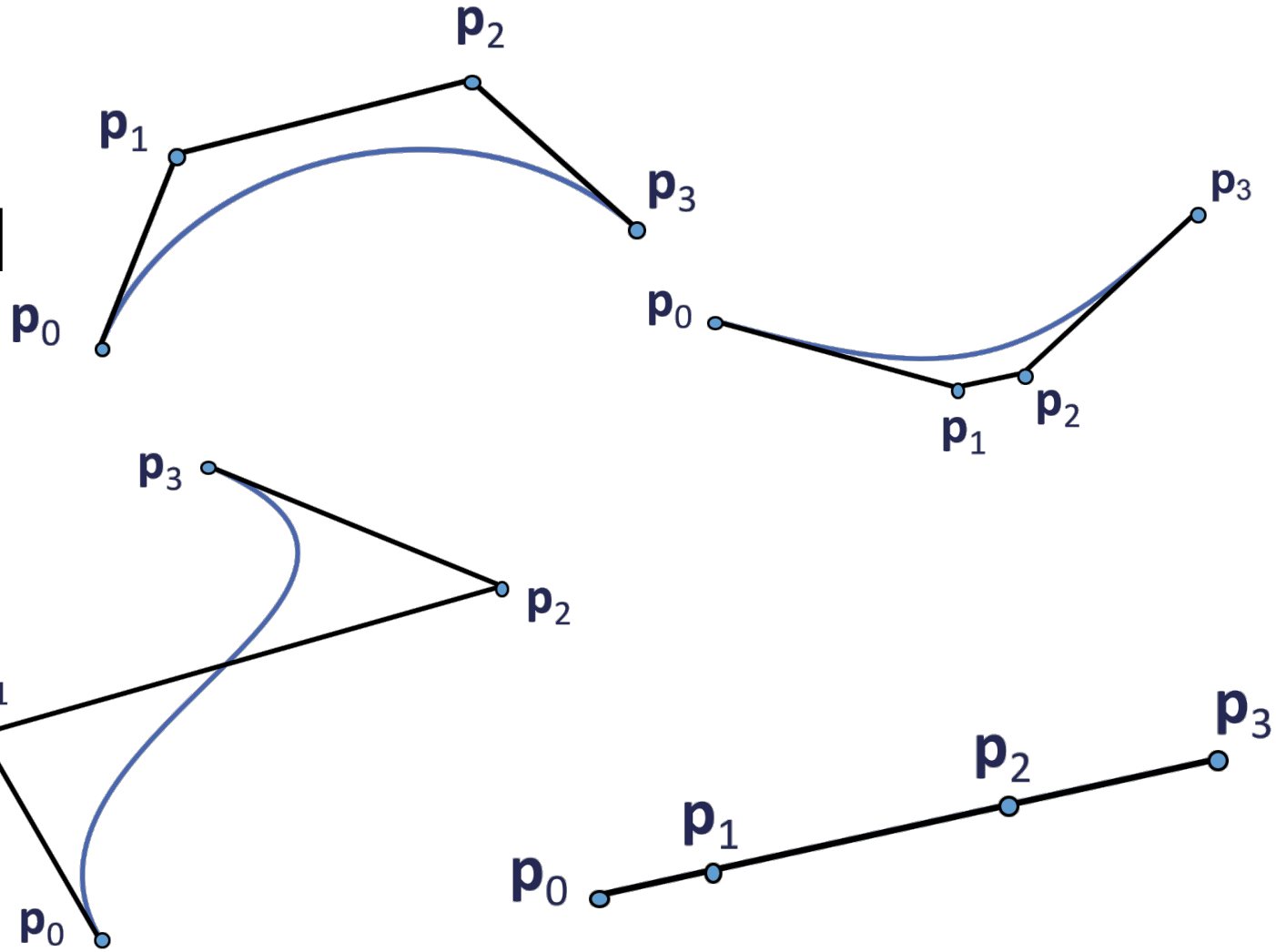



# Bezier Curves in Bernstein form

- **Bezier Curves:**

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

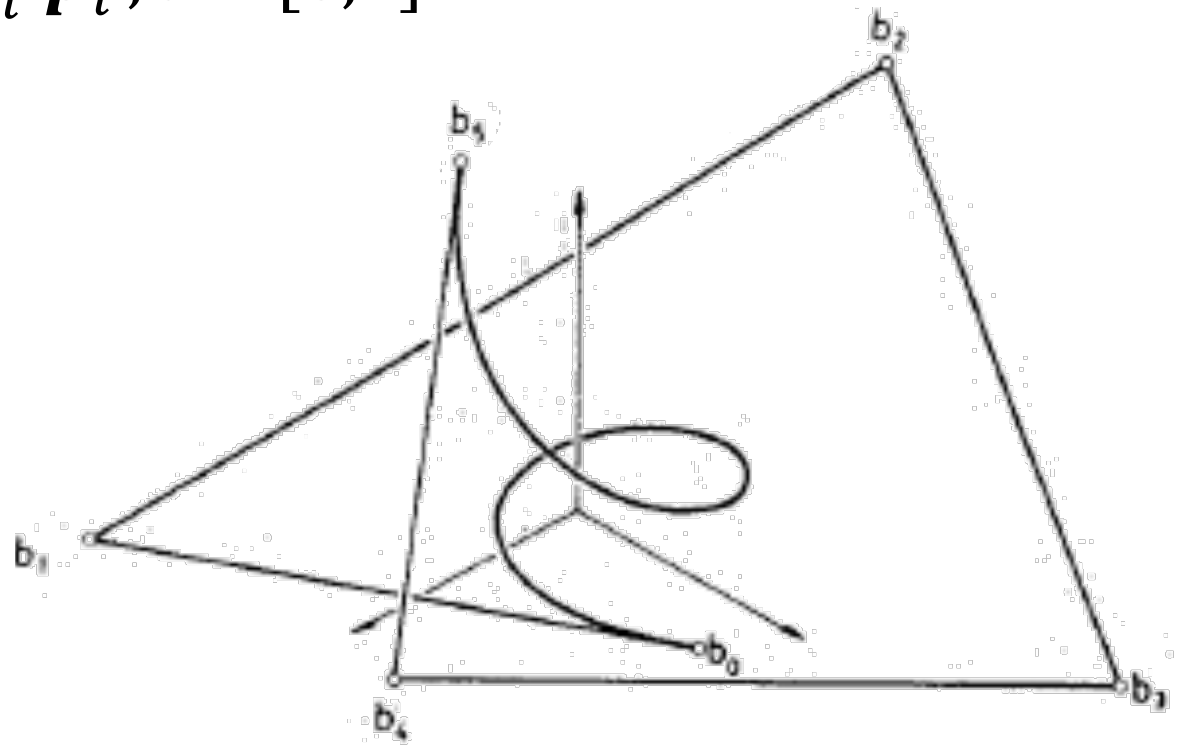


# Bezier Curves in Bernstein form

- **Bezier Curves, also in 3D**

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



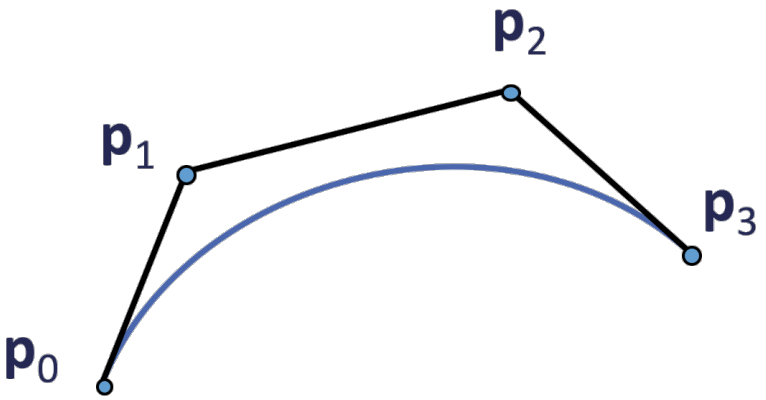
# Bezier Curves in Bernstein form

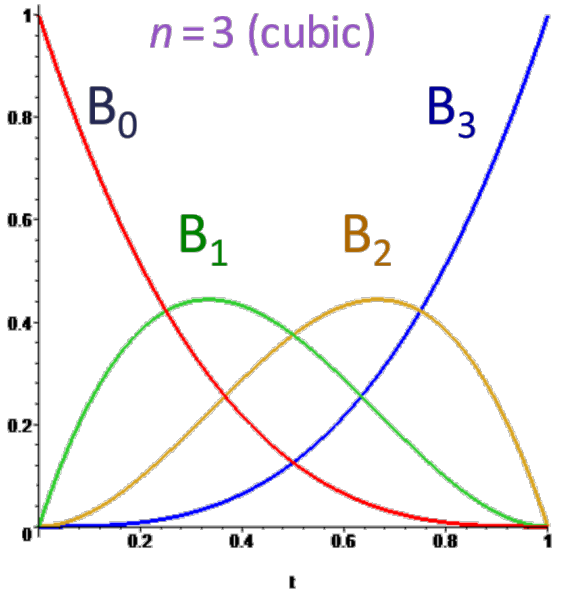
- Bezier curves:
  - Curves: $\boldsymbol{f}(t) = \sum_{i=1}^{n} B_i^n \boldsymbol{p}_i$
  - Considering the interval $t \in [0..1]$
  - Properties as discussed before:
    - Affine invariant
    - Curves contained in the convex hull
    - Influence of control points

    Moving along the curve with index $i$

    Largest influence at $t = \frac{i}{n}$

    Single curve segments: no full local control



# Bezier Curve Properties: another look at derivatives

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

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

Index change

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

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

# Bezier Curve Properties

- Higher order derivatives:

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

# Bezier Curve Properties

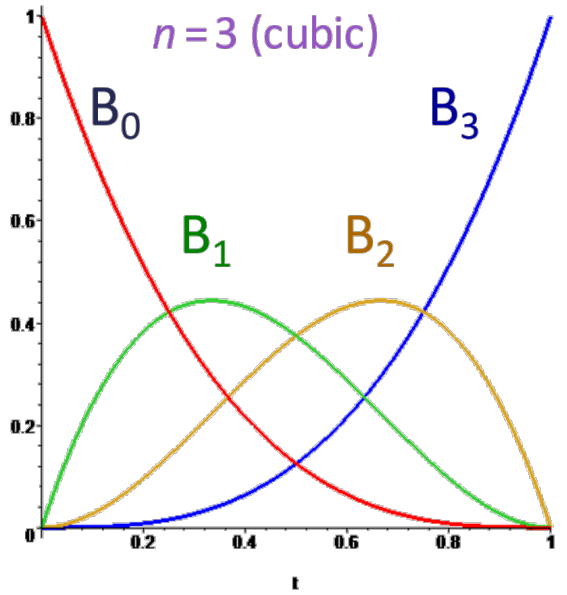
- Imporant for continuous concatenation:
  - Function value at $\{0,1\}$:

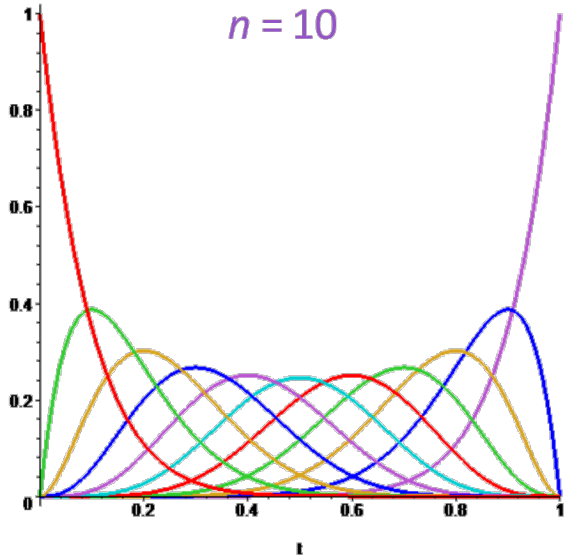
$$\boldsymbol{f}(t) = \sum_{i=0}^{n-1} \binom{n}{i} t^i (1-t)^{n-i} \boldsymbol{p}_i$$

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

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

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

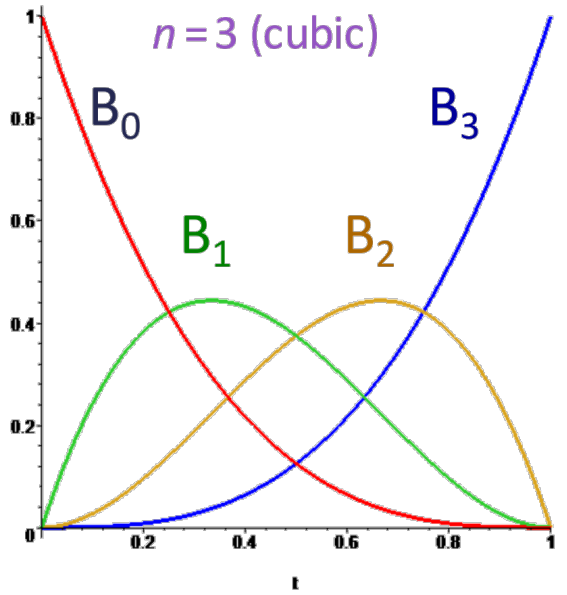


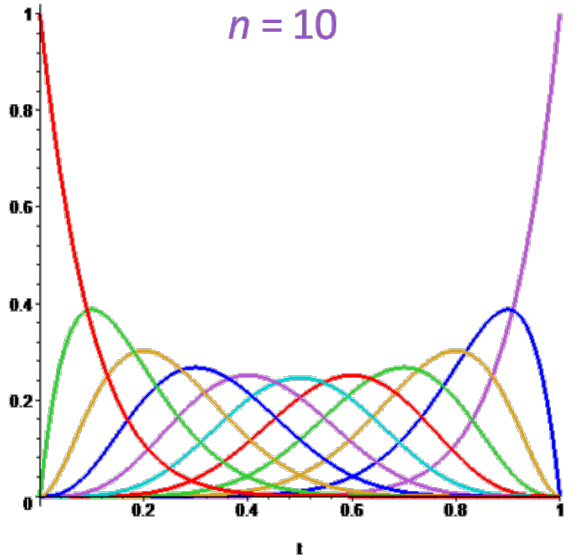



# Bezier Curve Properties

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

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

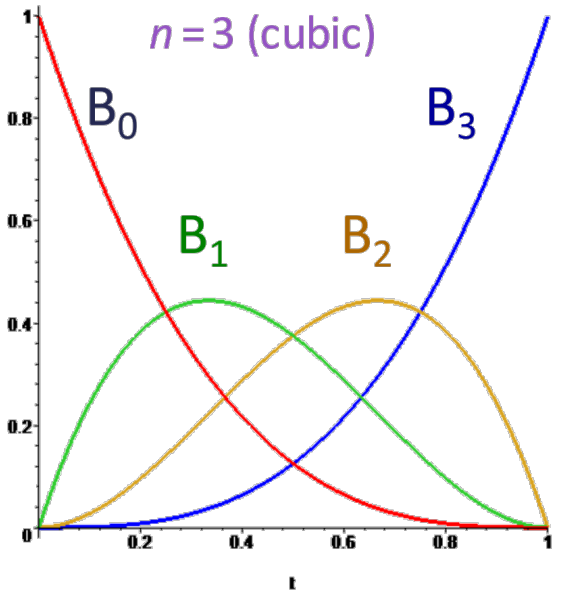


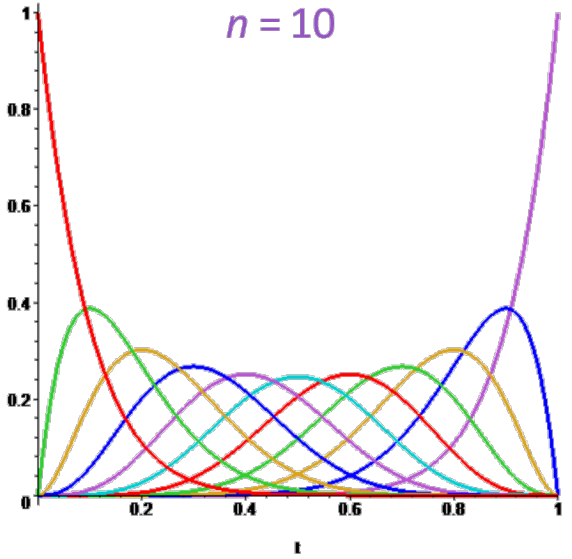



# Bezier Curve Properties

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

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



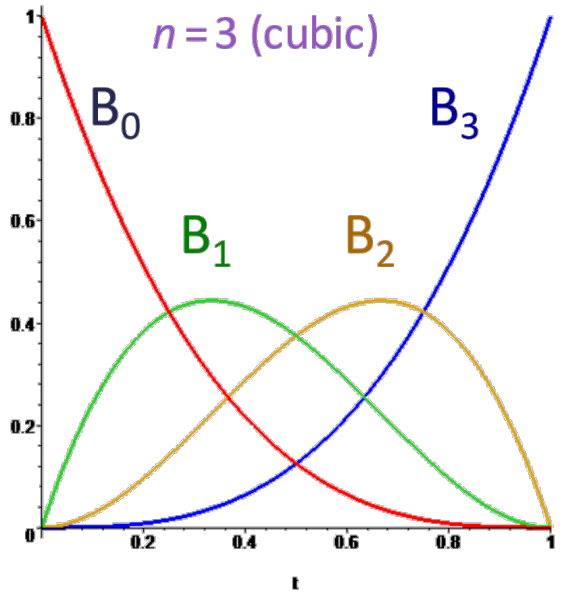
# Bezier Curve Properties

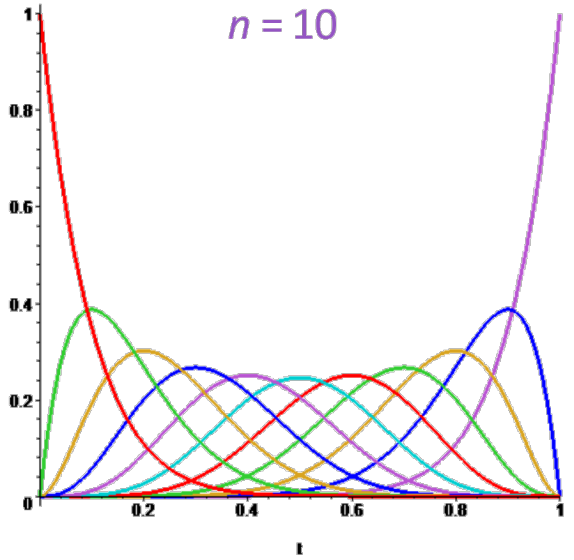
First derivative vector at $\{0,1\}$

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

$$= n\left(\left[-B_0^{(n-1)}(t)\right]\boldsymbol{p}_0 + \left[B_0^{(n-1)}(t) - B_1^{(n-1)}(t)\right]\boldsymbol{p}_1 + \cdots\right.$$

$$\frac{d}{dt}\boldsymbol{f}(0) = n(\boldsymbol{p}_1 - \boldsymbol{p}_0) \qquad \frac{d}{dt}\boldsymbol{f}(1) = n(\boldsymbol{p}_n - \boldsymbol{p}_{n-1})$$



# Bezier Curve Properties

- Imporant for continuous concatenation:
  - Function value at $\{0,1\}$:

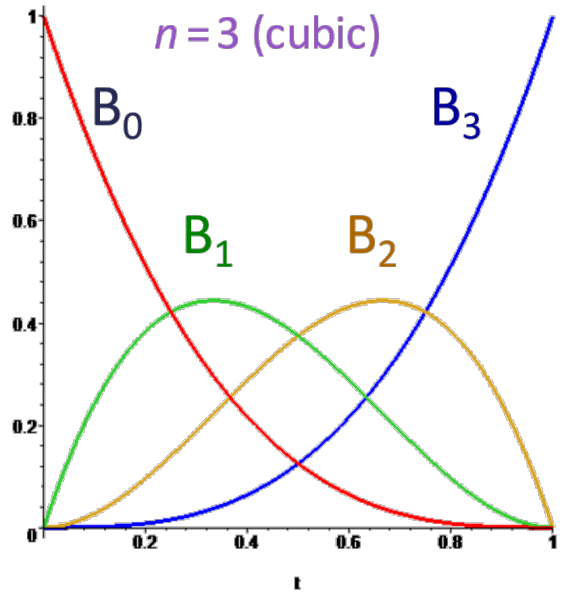
$$\boldsymbol{f}(0) = \boldsymbol{p}_0$$
$$\boldsymbol{f}(1) = \boldsymbol{p}_1$$

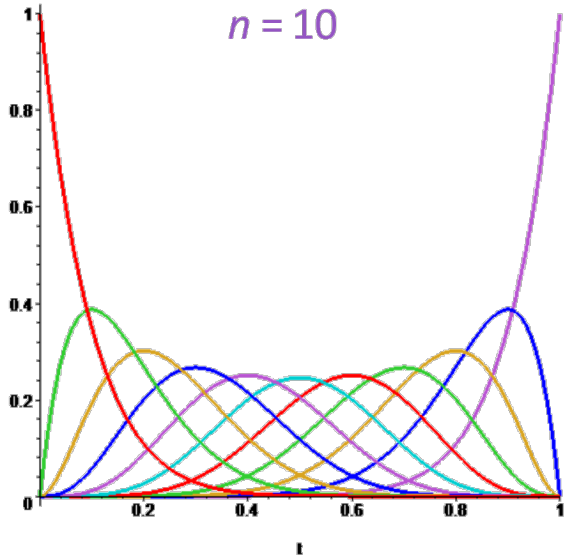
  - First derivative vector at $\{0,1\}$

$$\boldsymbol{f}'(0) = n[\boldsymbol{p}_1 - \boldsymbol{p}_0]$$
$$\boldsymbol{f}'(1) = n[\boldsymbol{p}_n - \boldsymbol{p}_{n-1}]$$

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

$$\boldsymbol{f}''(0) = n(n-1)[\boldsymbol{p}_2 - 2\boldsymbol{p}_1 + \boldsymbol{p}_0]$$
$$\boldsymbol{f}''(1) = n(n-1)[\boldsymbol{p}_n - 2\boldsymbol{p}_{n-1} + \boldsymbol{p}_{n-2}]$$



# Degree elevation

- Given: $\boldsymbol{b}_0,\cdots,\boldsymbol{b}_n \rightarrow \boldsymbol{x}(t)$
- Wanted: $\overline{\boldsymbol{b}}_0, \ldots, \overline{\boldsymbol{b}}_n, \overline{\boldsymbol{b}}_{n+1} \rightarrow \overline{\boldsymbol{x}}(t)$ with $\boldsymbol{x} = \overline{\boldsymbol{x}}$
- Solution:

# Degree elevation

- Given: $\boldsymbol{b}_0,\cdots,\boldsymbol{b}_n \rightarrow \boldsymbol{x}(t)$
- Wanted: $\overline{\boldsymbol{b}}_0, \ldots, \overline{\boldsymbol{b}}_n, \overline{\boldsymbol{b}}_{n+1} \rightarrow \overline{\boldsymbol{x}}(t)$ with $\boldsymbol{x} = \overline{\boldsymbol{x}}$
- Solution:

$$\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$$

$$\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \ \textit{for}\ j = 1, \ldots, n$$

# Proof

- Let's consider

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

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

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

Similarly

$$tB_i^n(t) = \frac{i+1}{n+1}B_i^{n+1}(t)$$

# Proof

$$\boldsymbol{f}(t) = [(1-t)+t]\boldsymbol{f}(t) = [(1-t)+t]\sum_{i=0}^{n} B_i^n(t)\boldsymbol{P}_i = \sum_{i=0}^{n}[(1-t)B_i^n(t) + tB_i^n(t)]\boldsymbol{P}_i$$

$$= \sum_{i=0}^{n}\left[\frac{n+1-i}{n+1}B_i^{n+1}(t) + \frac{i+1}{n+1}B_{i+1}^{n+1}(t)\right]\boldsymbol{P}_i = \sum_{i=0}^{n}\frac{n+1-i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_i + \sum_{i=0}^{n}\frac{i+1}{n+1}B_{i+1}^{n+1}(t)\boldsymbol{P}_i$$

Using results from last slide

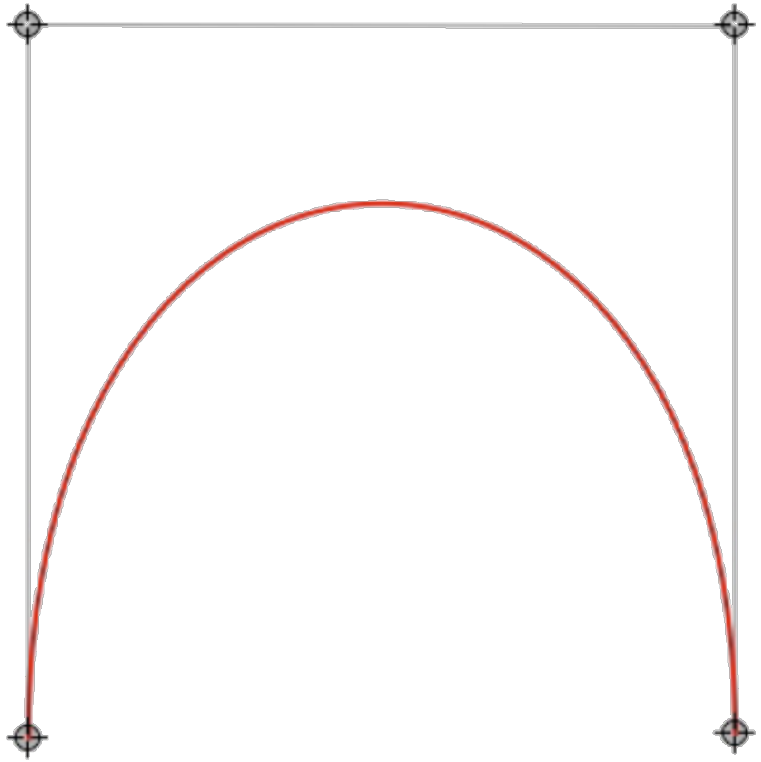
$$= \sum_{i=0}^{n}\frac{n+1-i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_i + \sum_{i=1}^{n+1}\frac{i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_{i-1}$$

$$= \sum_{i=0}^{n+1}\frac{n+1-i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_i + \sum_{i=0}^{n+1}\frac{i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_{i-1}$$

Adding null terms, $i = n+1$, $i = 0$

$$= \sum_{i=0}^{n+1} B_i^{n+1}(t)\left[\frac{n+1-i}{n+1}\boldsymbol{P}_i + \frac{i}{n+1}\boldsymbol{P}_{i-1}\right]$$

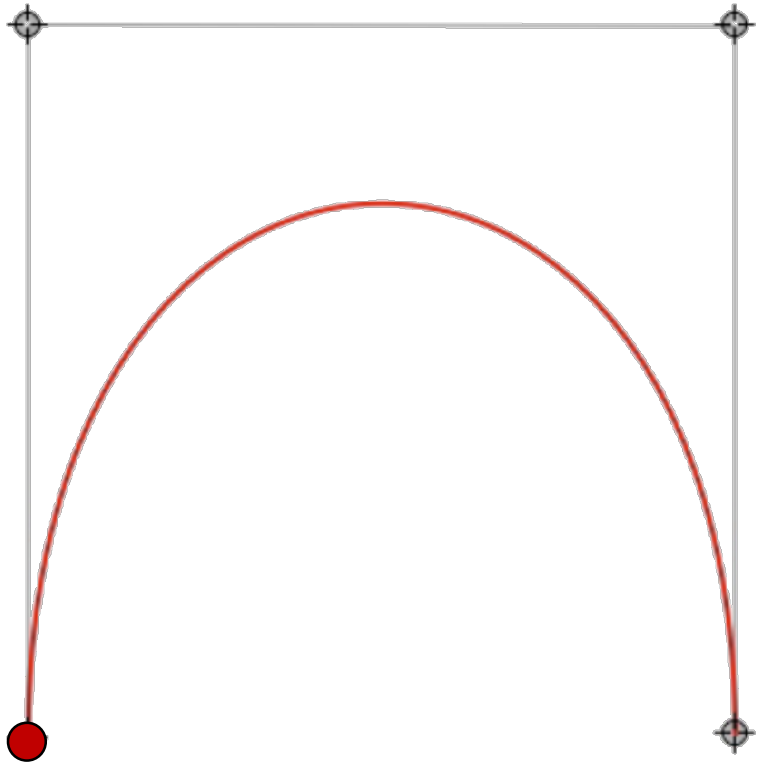
# Degree elevation: Example



- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

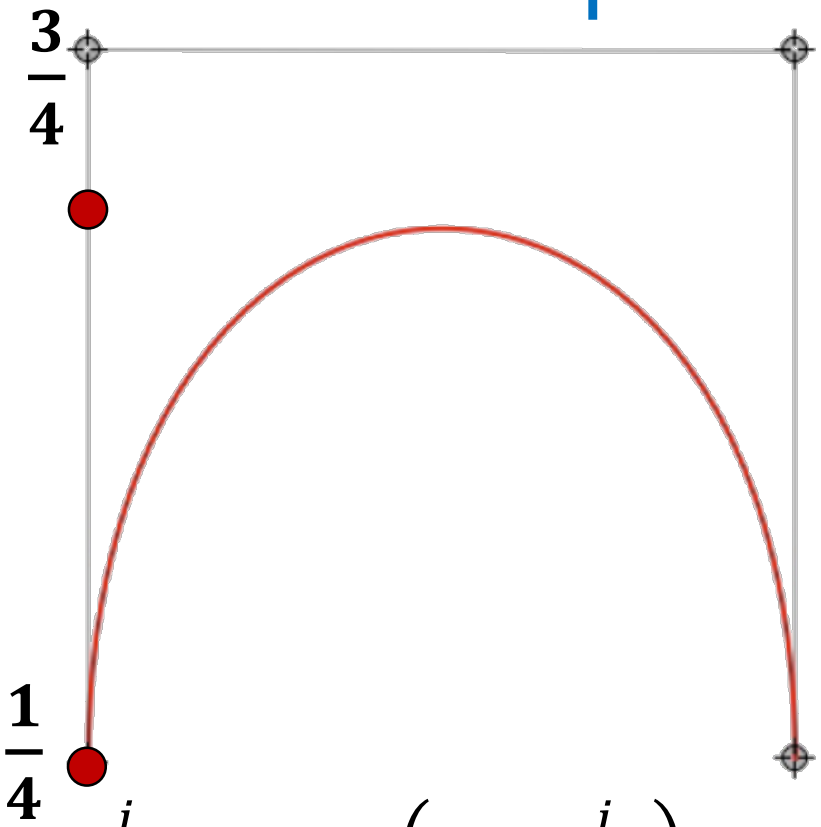
# Degree elevation: Example




- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

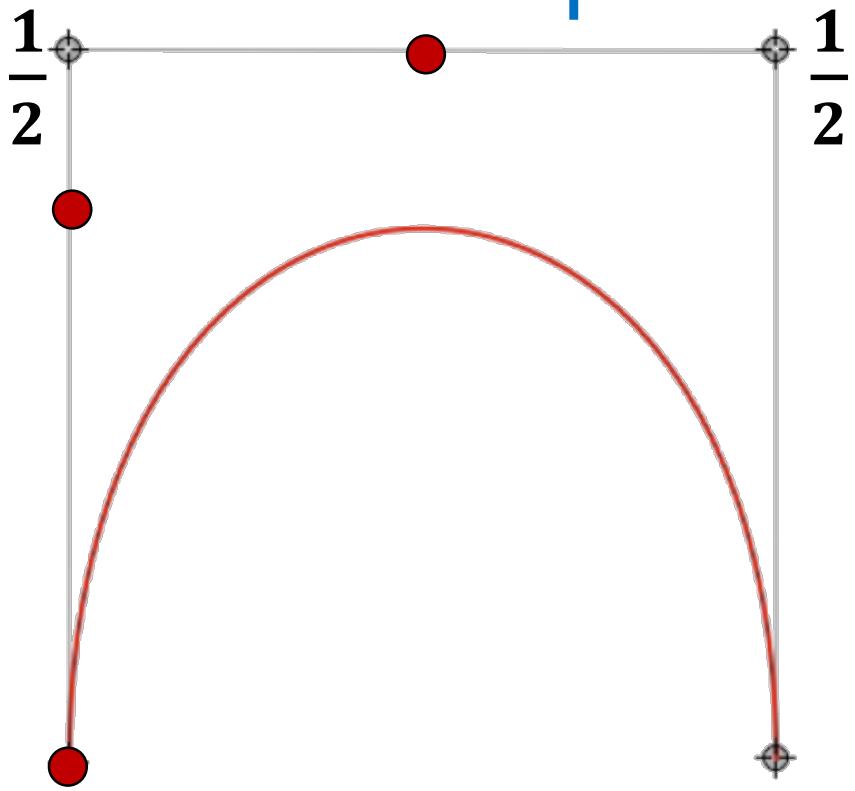
# Degree elevation: Example




- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

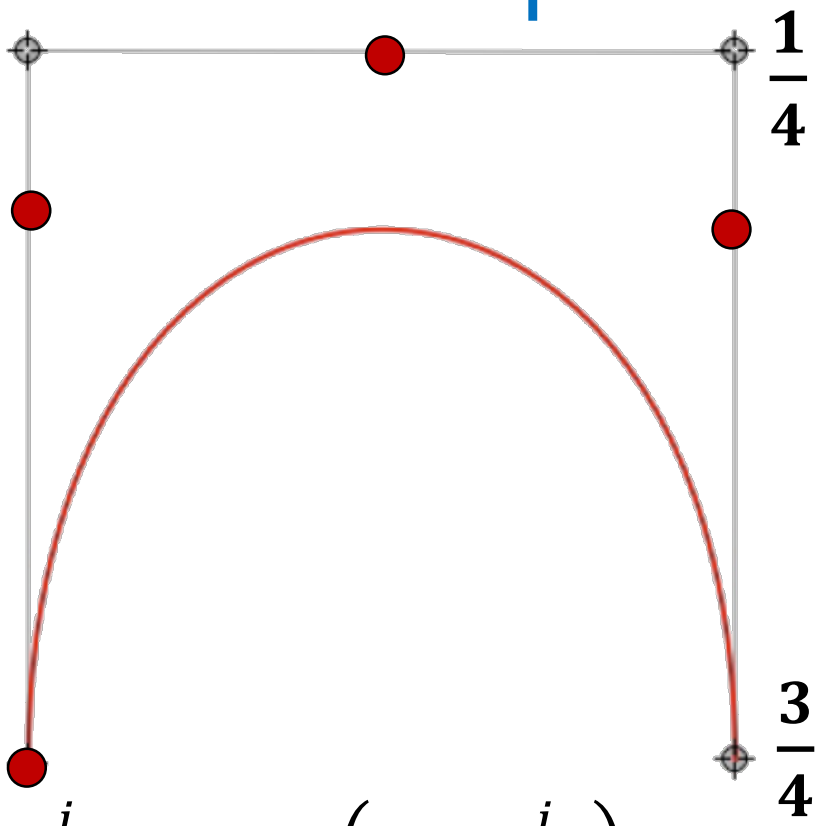
# Degree elevation: Example




- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

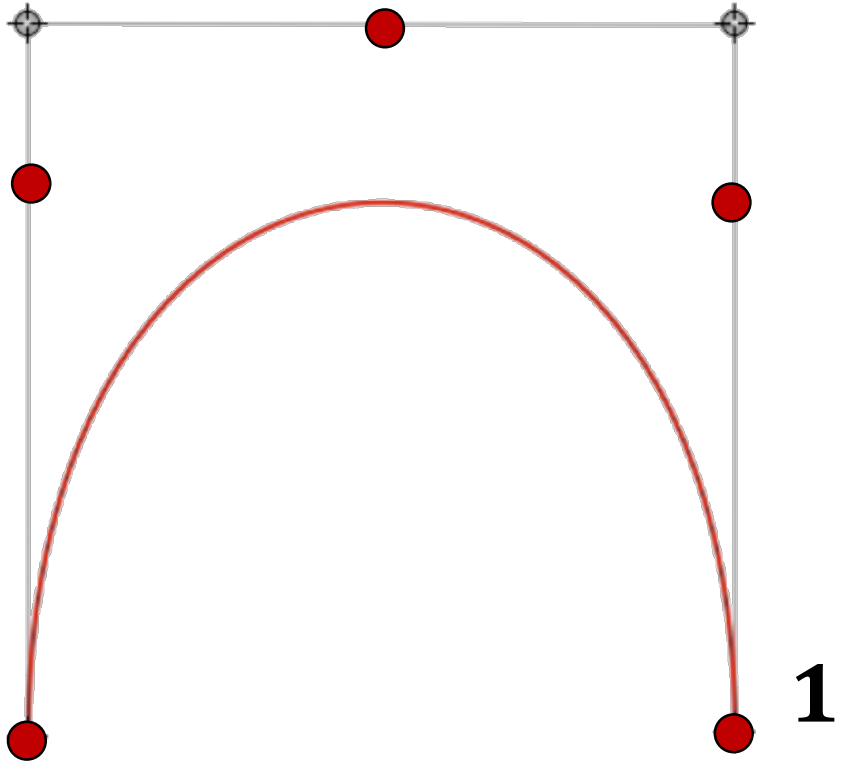
# Degree elevation: Example




- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

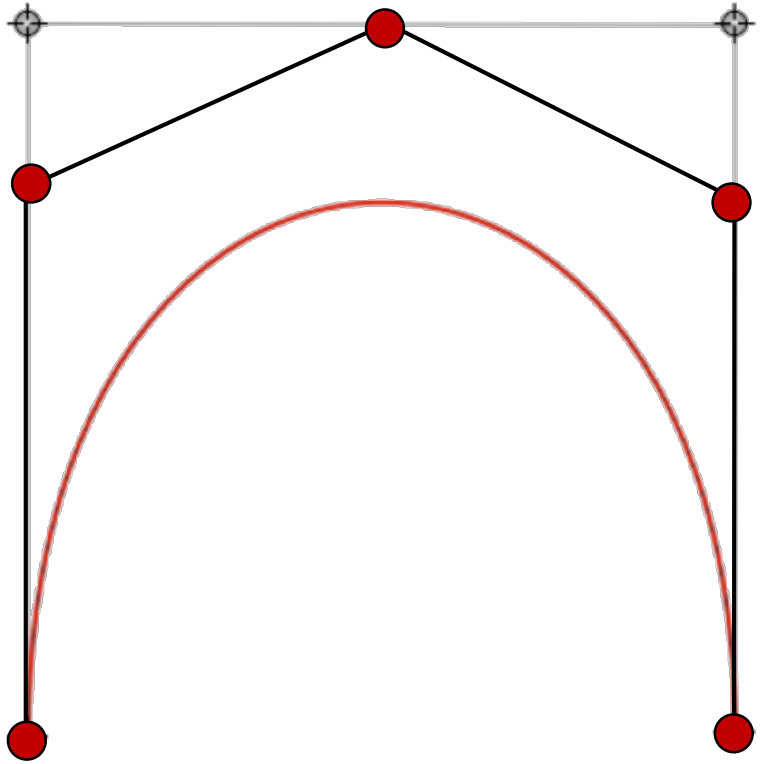
# Degree elevation: Example




- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

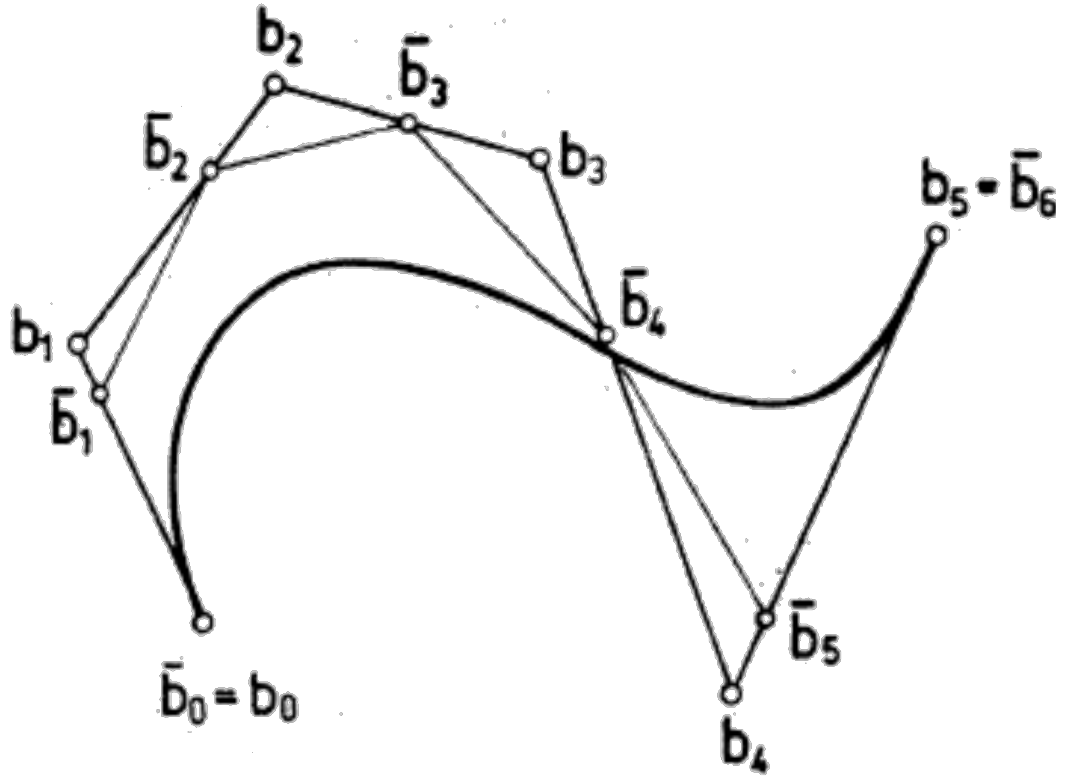
# Degree elevation: Example



- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

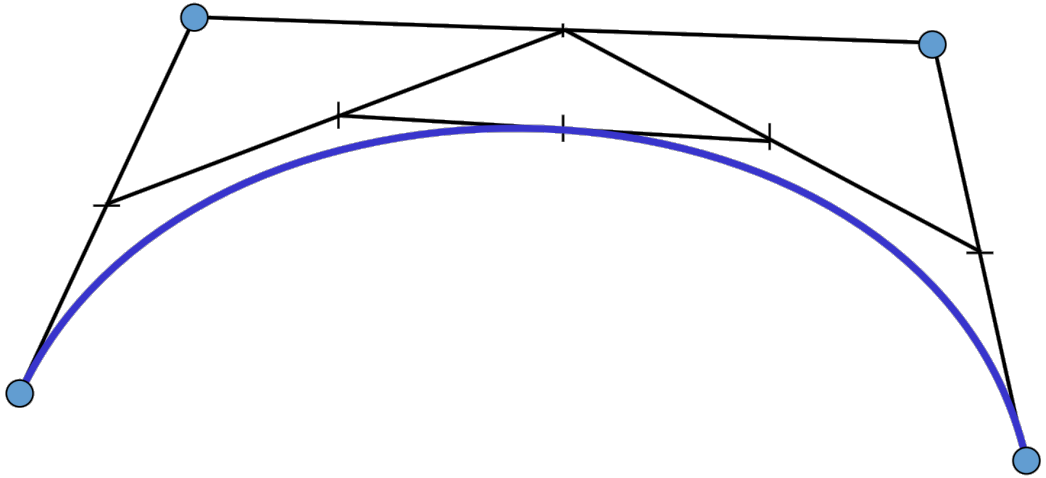
$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \qquad j = 1, \ldots, n$$

# Degree elevation




For repeated degree elevation, the Bezier polygon converges to the Bezier curve. (slow convergence)

# Recap



de Casteljau algorithm

Curve  basis function  control points

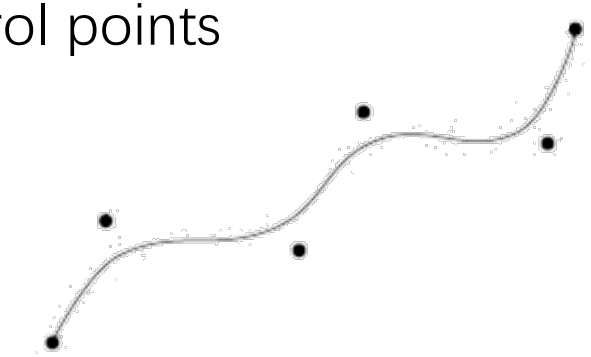
$$\boldsymbol{f}(t) = \sum_{i=1}^{n} b_i(t)\boldsymbol{p}_i$$

**Useful properties for basis functions**

- Smoothness
- Local control / support
- Affine invariance
- Convex hull property

# Summary and Outlook

- Bezier curves and curve design
  - The rough form is specified by the position of the control points
  - Results: smooth curve approximating the control points
  - Computation / Representation:
    - de Casteljau algorithm
    - Bernstein form



  - Problems:
    - High polynomial degree
    - Moving a control point can change the whole curve
    - Interpolation of points
    - →**Bezier splines**