计算机辅助几何设计 2023秋学期

Bézier Curves

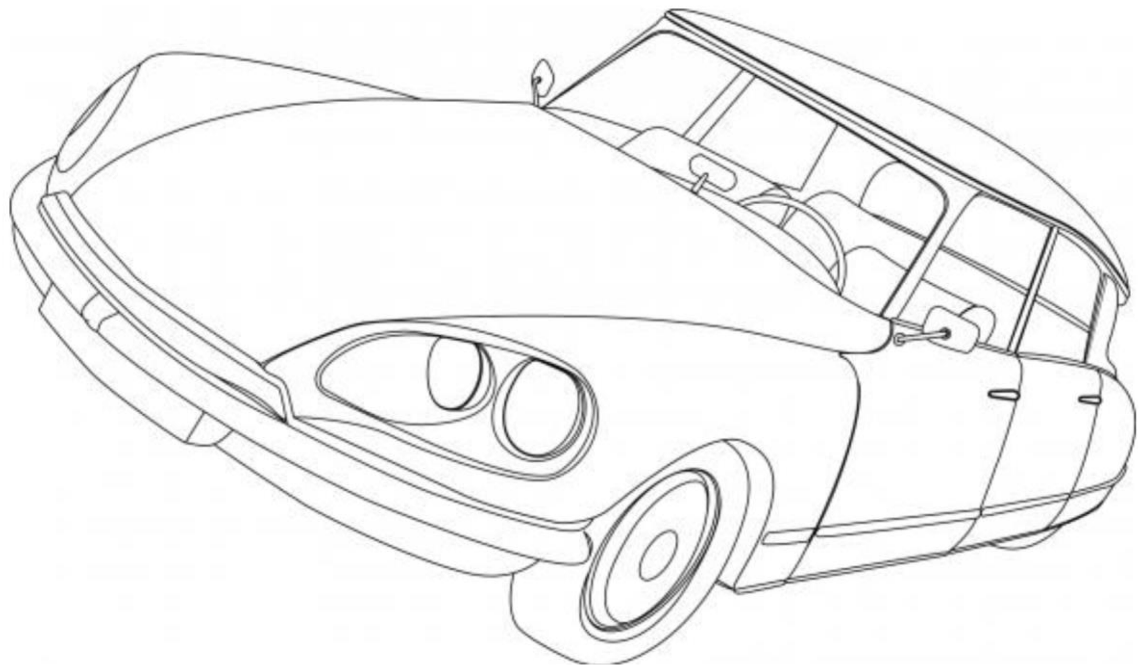
陈仁杰

中国科学技术大学

Bézier curves

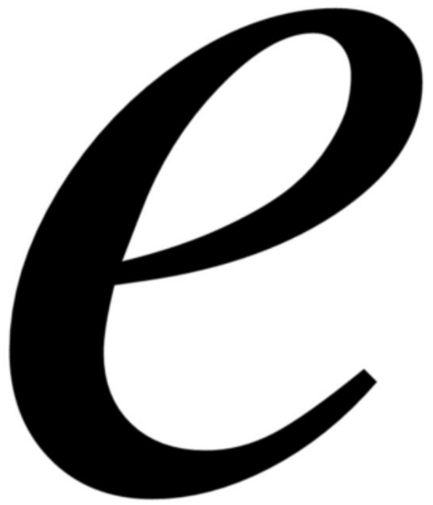
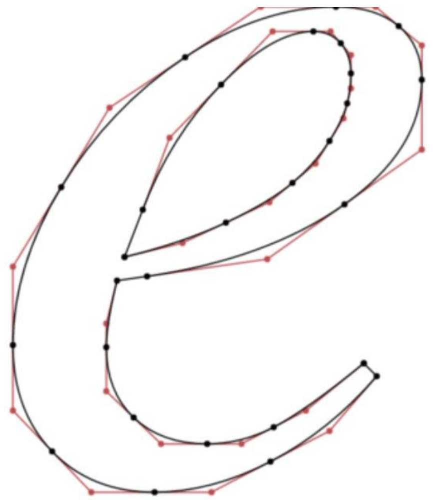
- Bézier curves/splines developed by
  - Paul de Casteljau at Citroen (1959)
  - Pierre Bézier at Renault (1963)

  for free-form parts in automotive design



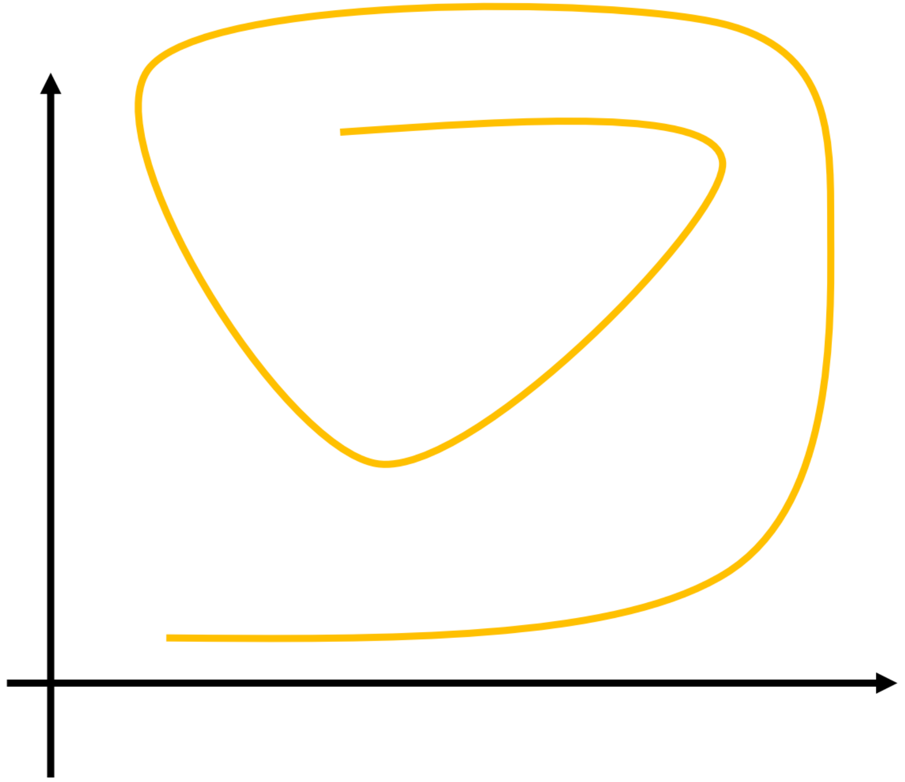
Bézier curves

- 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

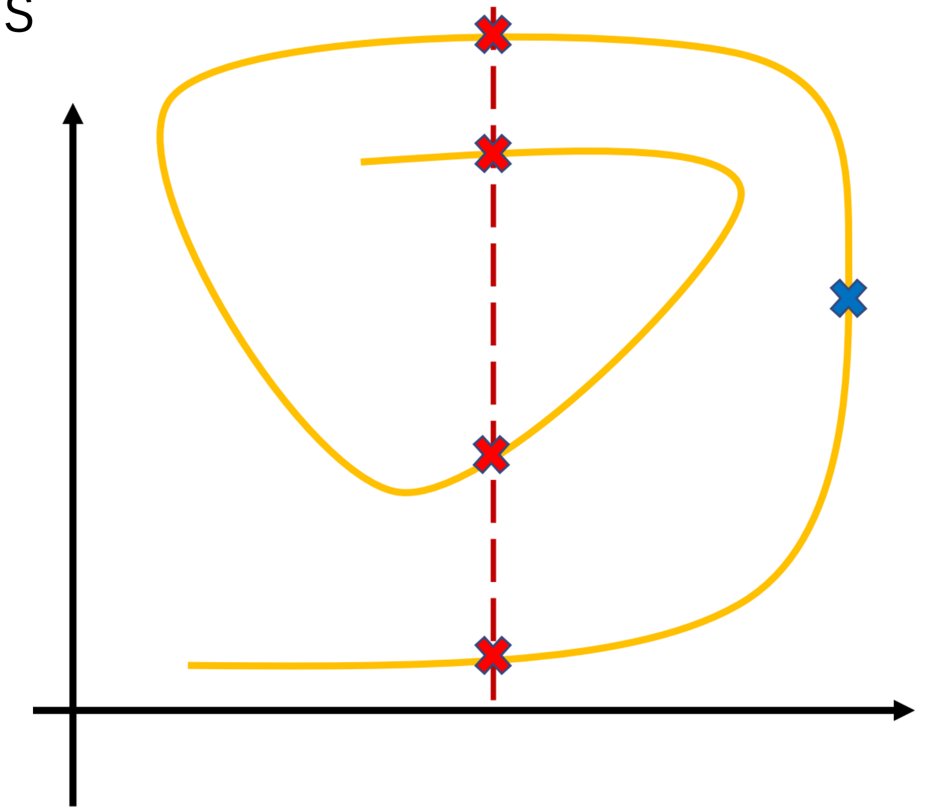
Curve representation

- 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

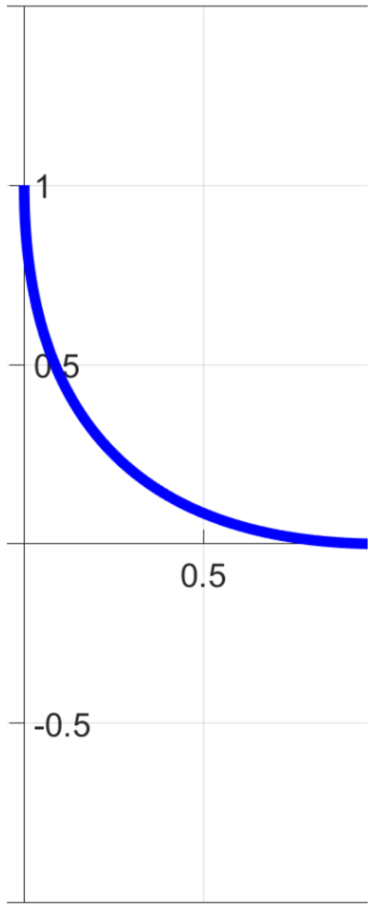


Parametric representation

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

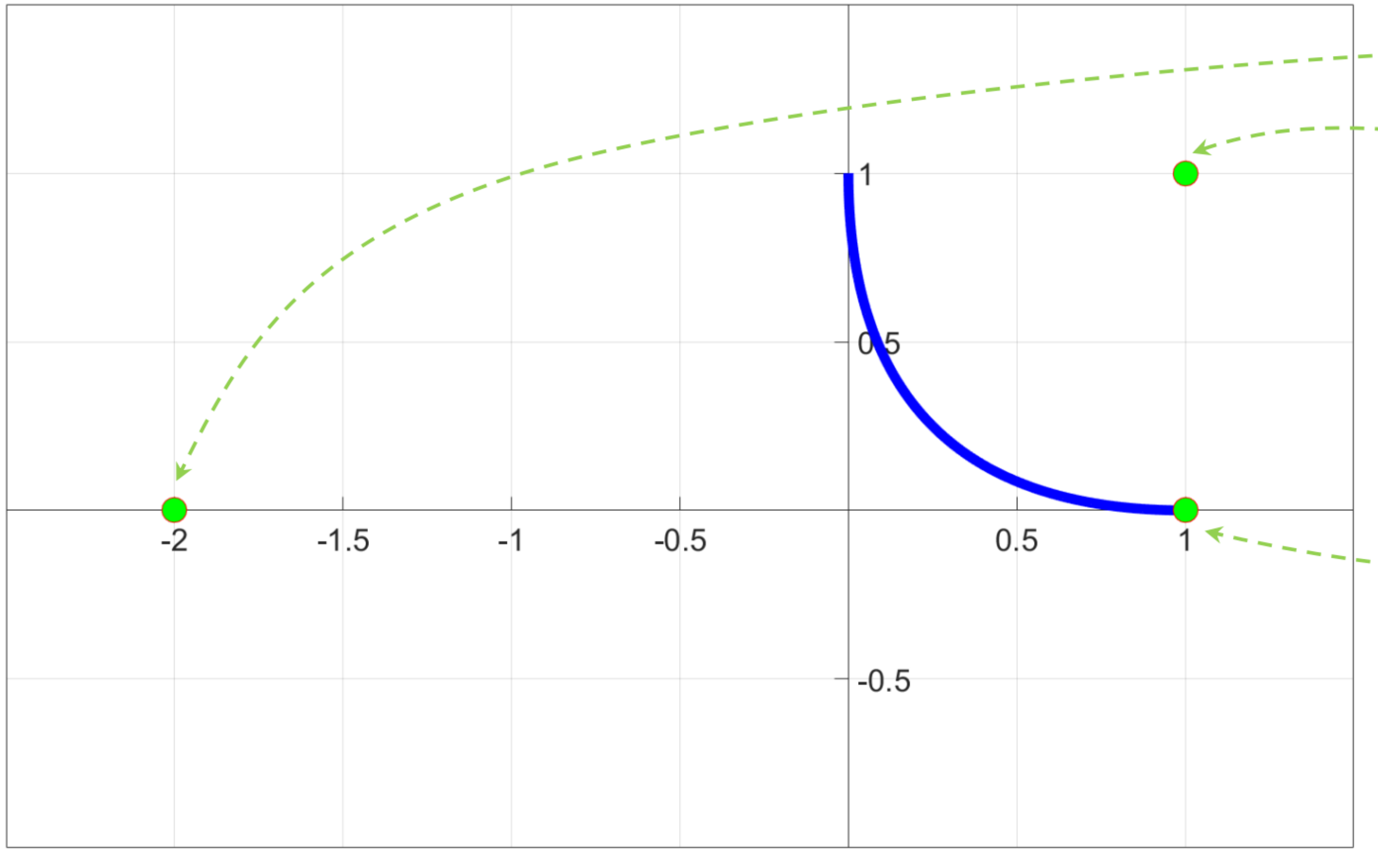
- Example of a parabola: $\boldsymbol{f}(t) = \boldsymbol{a}t^2 + \boldsymbol{b}t + \boldsymbol{c}$



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

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

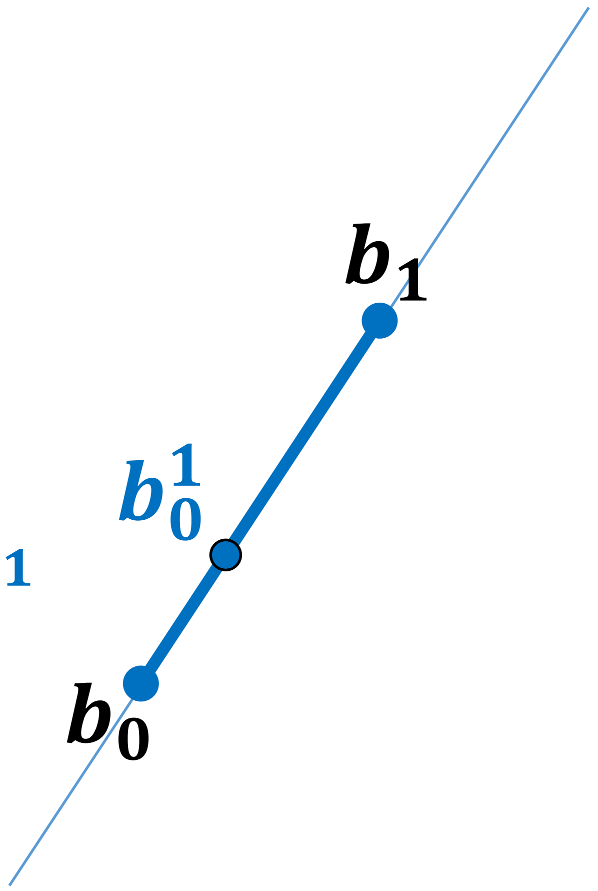


$$\mathbf{f}(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} t^2 + \begin{pmatrix} -2 \\ 0 \end{pmatrix} t + \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Back to the drawing board

- A point on a parametric line

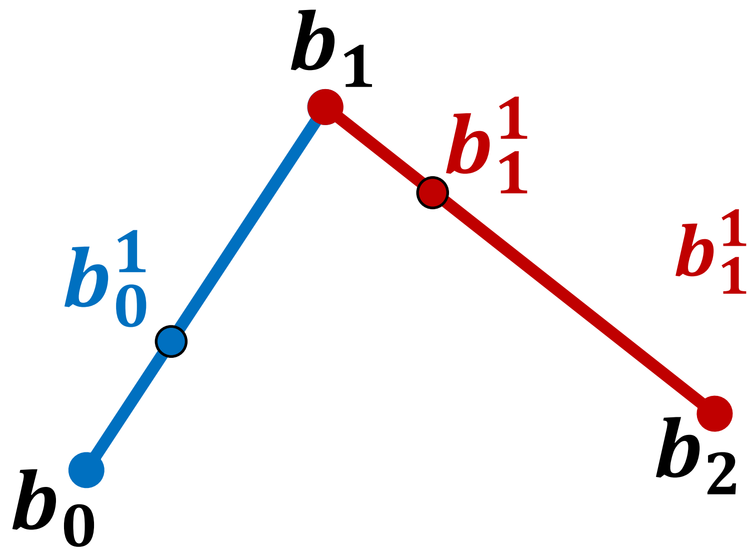
$$\boldsymbol{b}_0^1 = (1-t)\boldsymbol{b}_0 + t\boldsymbol{b}_1$$



Back to the drawing board

- Another point on a second parametric line

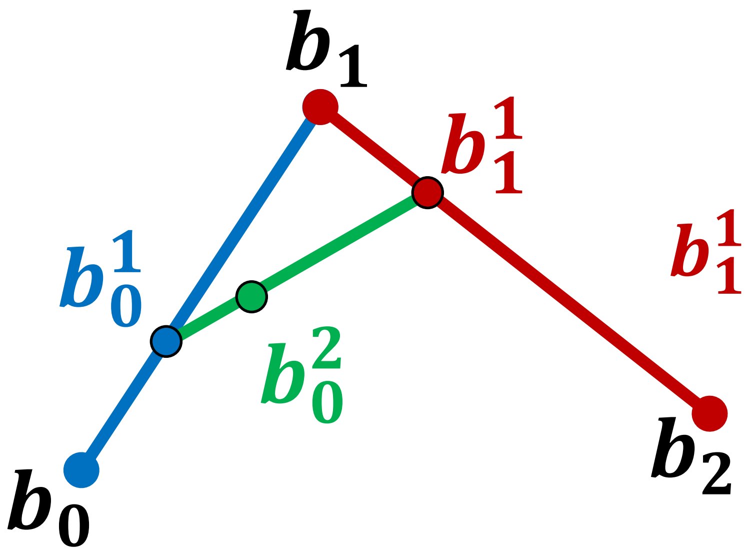
$\boldsymbol{b}_0^1 = (1-t)\boldsymbol{b}_0 + t\boldsymbol{b}_1$



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

Back to the drawing board

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

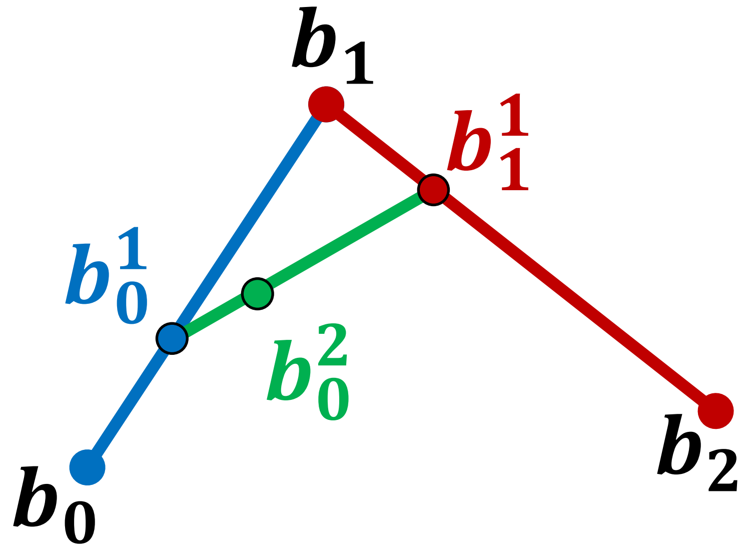


$$\boldsymbol{b}_0^1 = (1-t)\boldsymbol{b}_0 + t\boldsymbol{b}_1$$

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

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

Back to the drawing board



- And then simplify⋯

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

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

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

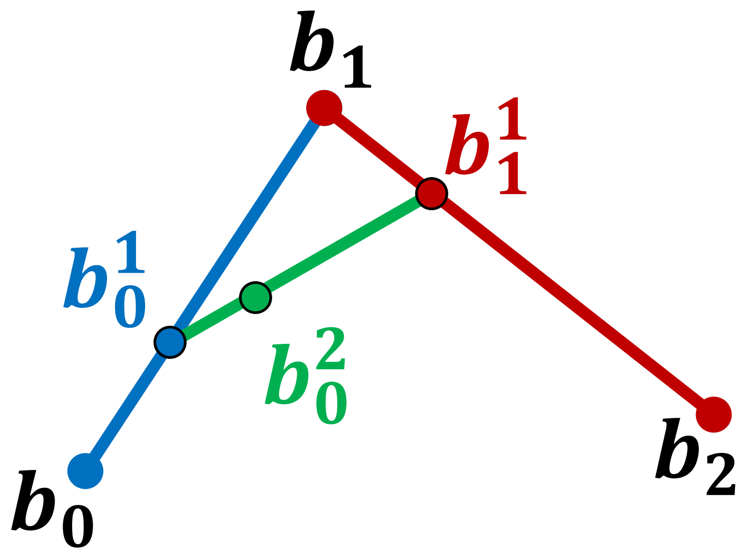
$$\boldsymbol{b}_0^2 = (1-t)[(1-t)\boldsymbol{b}_0 + t\boldsymbol{b}_1] + t[(1-t)\boldsymbol{b}_1 + t\boldsymbol{b}_2]$$

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

Back to the drawing board

- We obtained another description of parabolic curves

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



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

Example re-visited

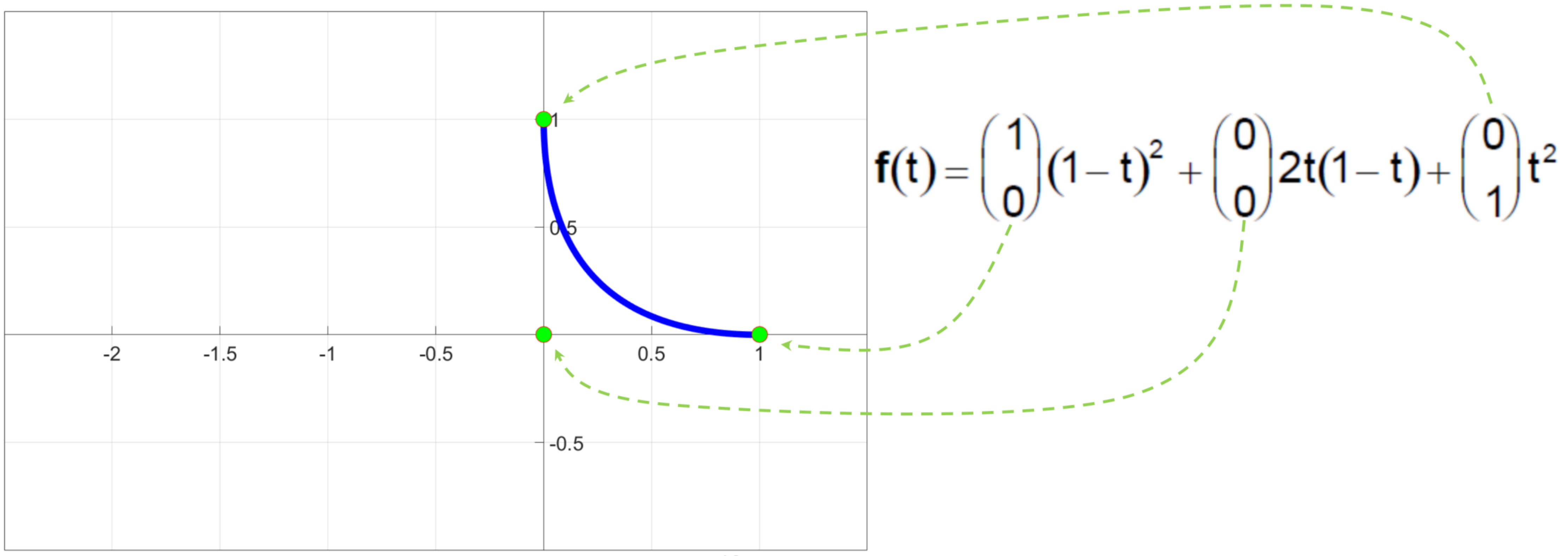
- Let’s rewrite our initial parabolic curve example in the new basis

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

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

Example re-visited

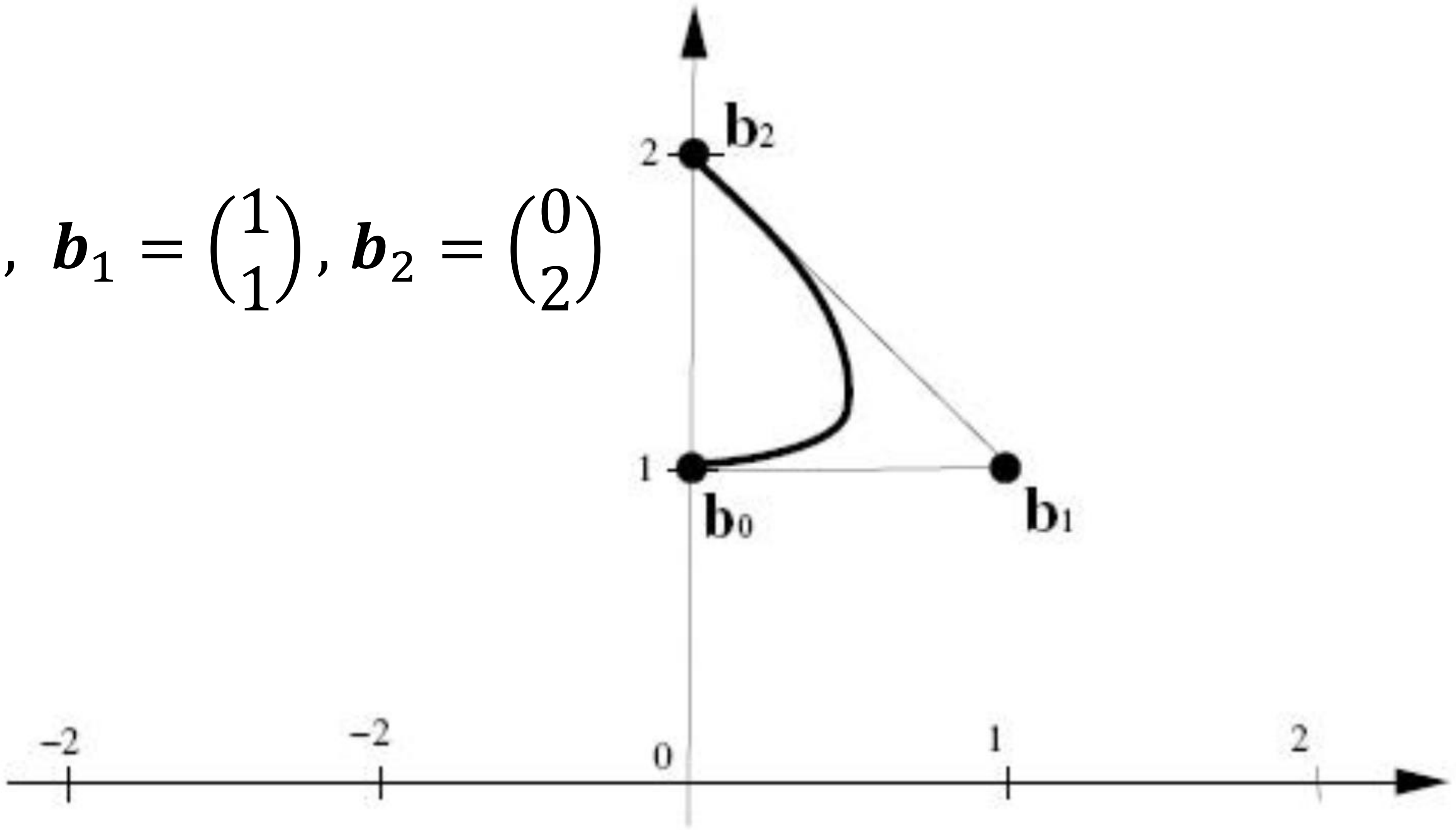
- The coefficient have a geometric meaning
- More intuitive for curve manipulation



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

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

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

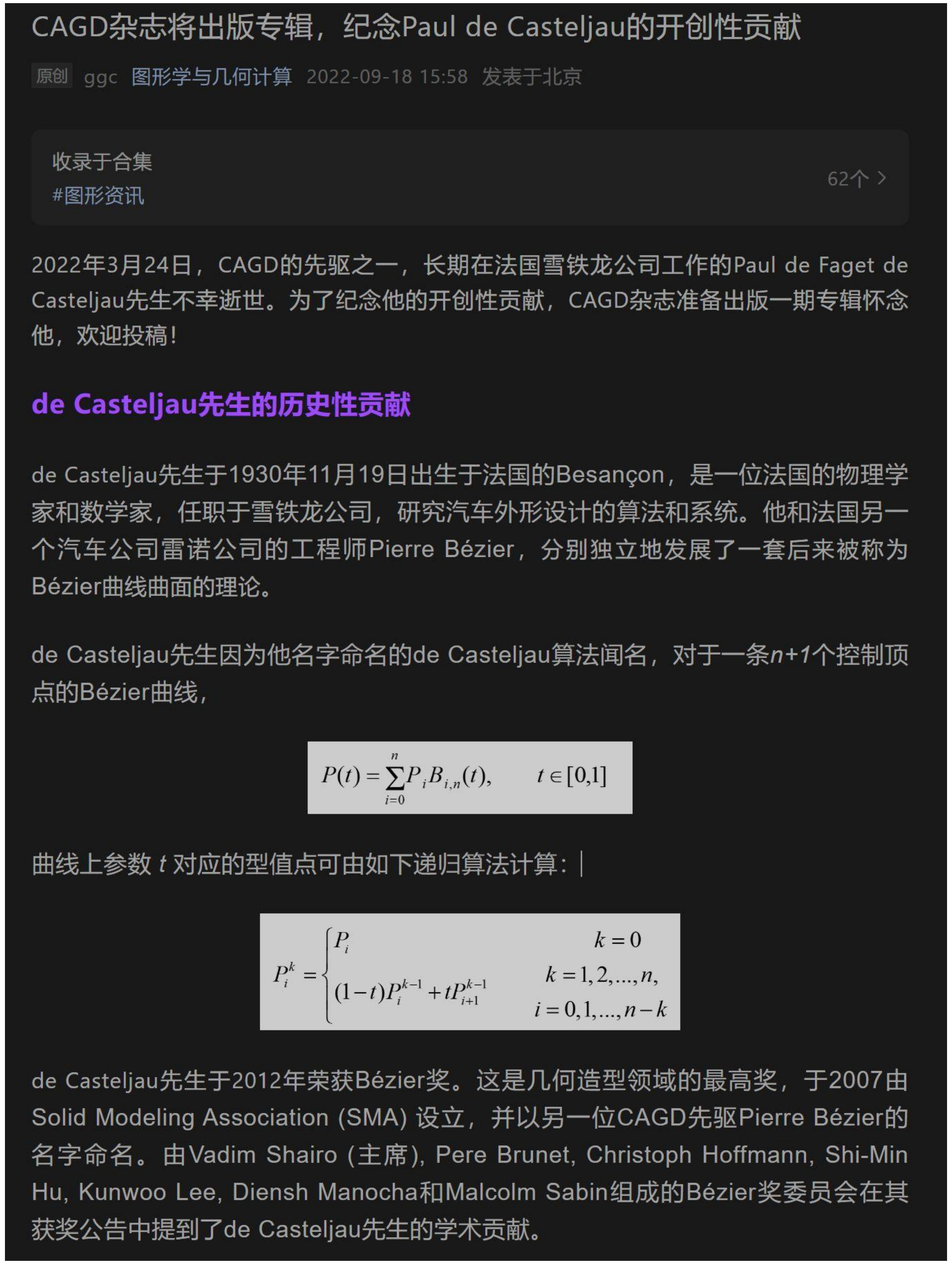
$$\boldsymbol{p}_1^2 = (1-t)^2\boldsymbol{p}_1 + 2t(1-t)\boldsymbol{p}_2 + t^2\boldsymbol{p}_3$$

- 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 …



CAGD杂志将出版专辑，纪念Paul de Casteljau的开创性贡献

原创 ggc 图形学与几何计算 2022-09-18 15:58 发表于北京

收录于合集
#图形资讯
62个 >

2022年3月24日，CAGD的先驱之一，长期在法国雪铁龙公司工作的Paul de Faget de Casteljau先生不幸逝世。为了纪念他的开创性贡献，CAGD杂志准备出版一期专辑怀念他，欢迎投稿！

de Casteljau先生的历史性贡献

de Casteljau先生于1930年11月19日出生于法国的Besançon，是一位法国的物理学家和数学家，任职于雪铁龙公司，研究汽车外形设计的算法和系统。他和法国另一个汽车公司雷诺公司的工程师Pierre Bézier，分别独立地发展了一套后来被称为Bézier曲线曲面的理论。

de Casteljau先生因为他名字命名的de Casteljau算法闻名，对于一条*n+1*个控制顶点的Bézier曲线，

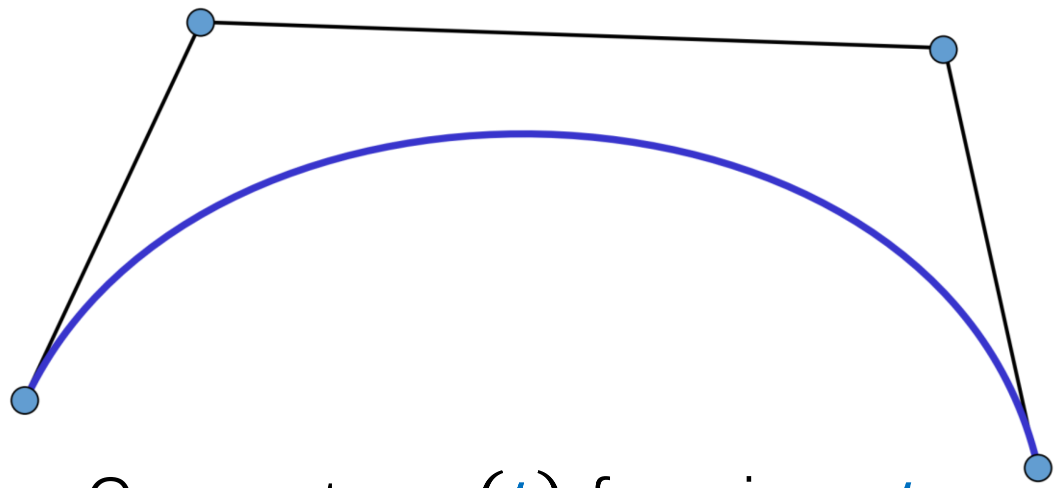
$$P(t)=\sum_{i=0}^{n} P_i B_{i,n}(t), \qquad t\in[0,1]$$

曲线上参数 *t* 对应的型值点可由如下递归算法计算：

$$P_i^k=\begin{cases} P_i & k=0 \\ (1-t)P_i^{k-1}+tP_{i+1}^{k-1} & k=1,2,\dots,n,\\ & i=0,1,\dots,n-k \end{cases}$$

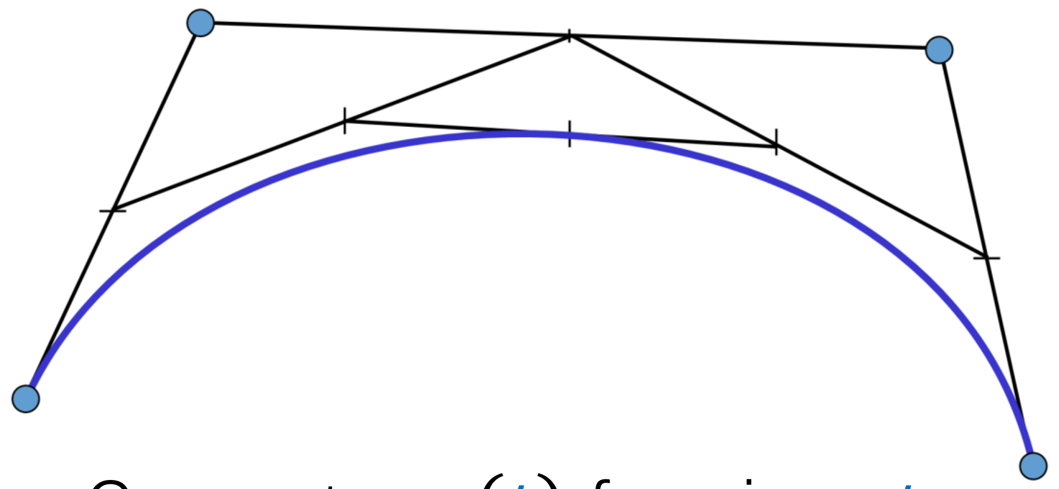
de Casteljau先生于2012年荣获Bézier奖。这是几何造型领域的最高奖，于2007由Solid Modeling Association (SMA) 设立，并以另一位CAGD先驱Pierre Bézier的名字命名。由Vadim Shairo (主席), Pere Brunet, Christoph Hoffmann, Shi-Min Hu, Kunwoo Lee, Diensh Manocha和Malcolm Sabin组成的Bézier奖委员会在其获奖公告中提到了de Casteljau先生的学术贡献。

De Casteljau algorithm



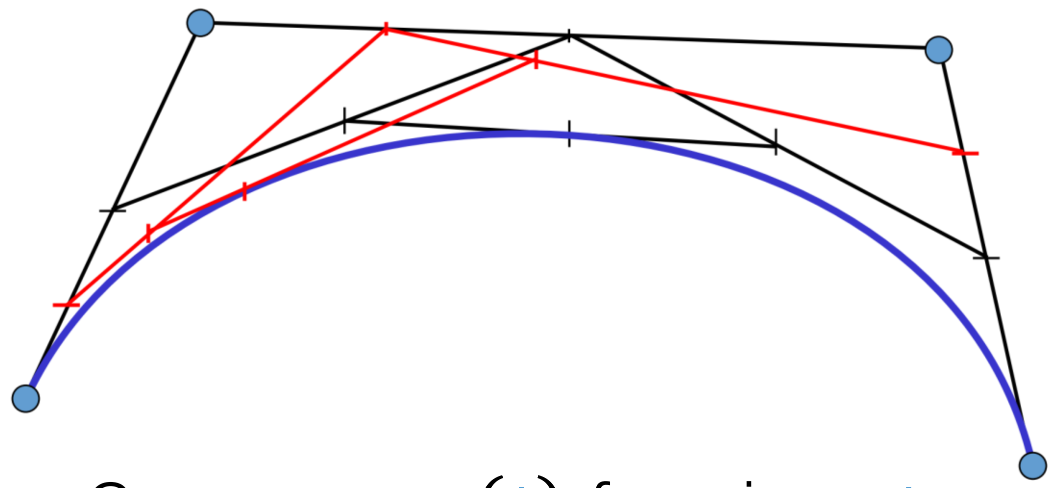
- De Casteljau Algorithm: Computes $x(t)$ for given t
  - Bisect control polygon in ratio $t: (1 - t)$
  - Connect the new dots with lines (adjacent segments)
  - Interpolate again with the same ratio
  - Iterate, until only one points is left

De Casteljau algorithm



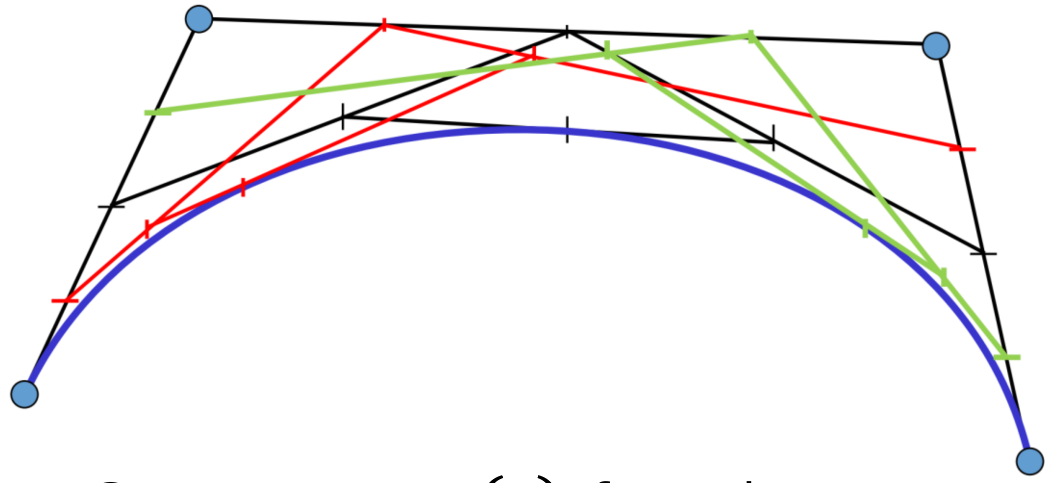
- De Casteljau Algorithm: Computes $x(t)$ for given t
  - Bisect control polygon in ratio $t : (1 - t)$
  - Connect the new dots with lines (adjacent segments)
  - Interpolate again with the same ratio
  - Iterate, until only one points is left

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

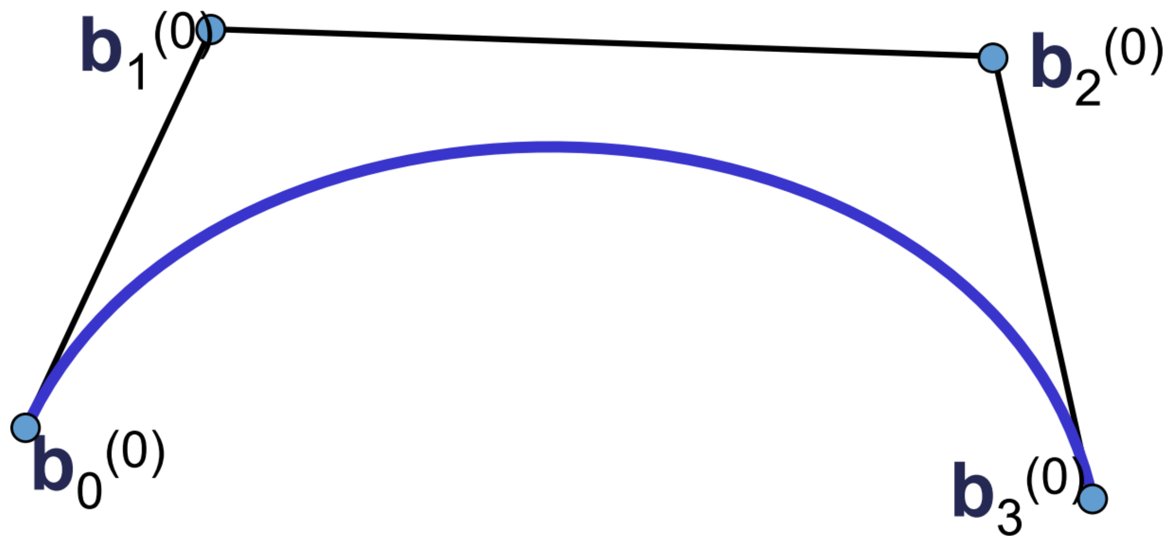
$$\boldsymbol{b}_i^{(r)} = (1 - t)\boldsymbol{b}_i^{(r-1)} + t\boldsymbol{b}_{i+1}^{(r-1)}$$

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

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

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

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

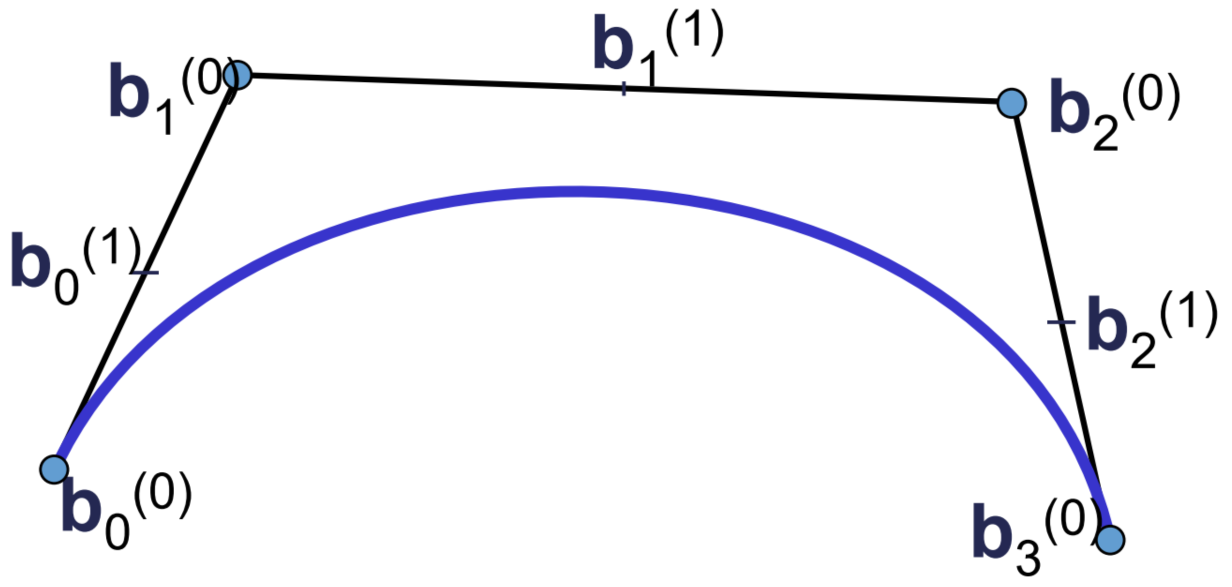


De Casteljau algorithm

- Repeated convex combination of control points

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

$$\begin{tikzcd}
\boldsymbol{b}_0^{(0)} \arrow[dr, "1-t"] & \\
\boldsymbol{b}_1^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_0^{(1)} \\
\boldsymbol{b}_2^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_1^{(1)} \\
\boldsymbol{b}_3^{(0)} \arrow[r, "t"] & \boldsymbol{b}_2^{(1)}
\end{tikzcd}$$

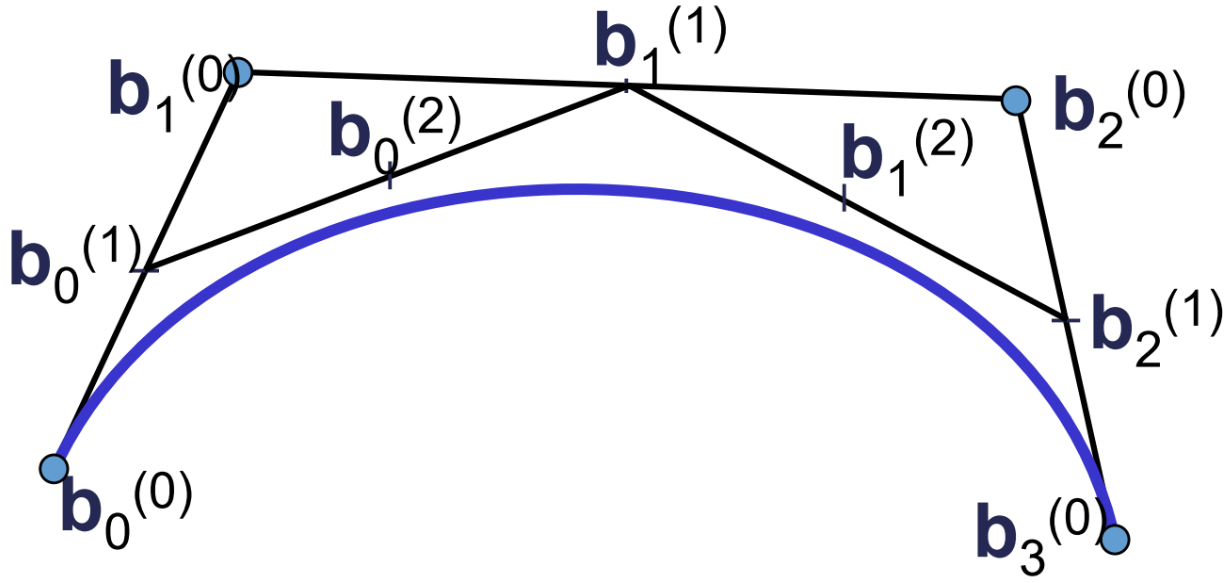


De Casteljau algorithm

- Repeated convex combination of control points

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

$$\begin{tikzcd}
\boldsymbol{b}_0^{(0)} \arrow[dr, "1-t"] & & \\
\boldsymbol{b}_1^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_0^{(1)} \arrow[dr, "1-t"] & \\
\boldsymbol{b}_2^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_1^{(1)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_0^{(2)} \\
\boldsymbol{b}_3^{(0)} \arrow[r, "t"] & \boldsymbol{b}_2^{(1)} \arrow[r, "t"] & \boldsymbol{b}_1^{(2)}
\end{tikzcd}$$



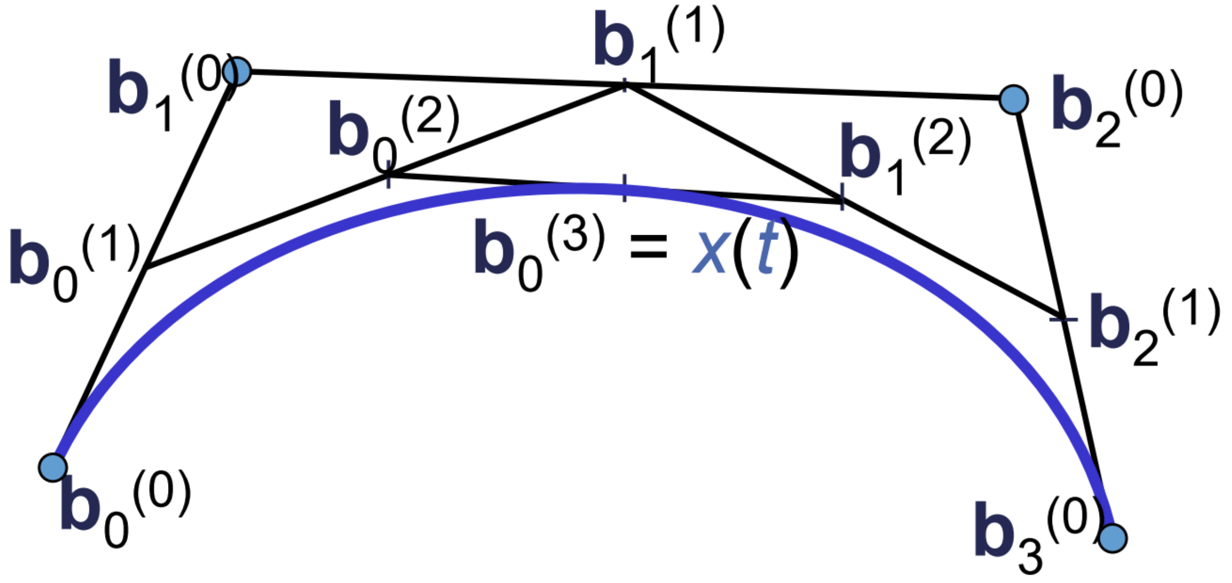
De Casteljau algorithm

- Repeated convex combination of control points

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

$$\begin{tikzcd}
\boldsymbol{b}_0^{(0)} \arrow[dr, "1-t"] & & & \\
\boldsymbol{b}_1^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_0^{(1)} \arrow[dr, "1-t"] & & \\
\boldsymbol{b}_2^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_1^{(1)} \arrow[r, "t"] \arrow[dr, "1-t"] & \boldsymbol{b}_0^{(2)} \arrow[dr, "1-t"] & \\
\boldsymbol{b}_3^{(0)} \arrow[r, "t"] & \boldsymbol{b}_2^{(1)} \arrow[r, "t"] & \boldsymbol{b}_1^{(2)} \arrow[r, "t"] & \boldsymbol{b}_0^{(3)} = x(t)
\end{tikzcd}$$

De Casteljau scheme

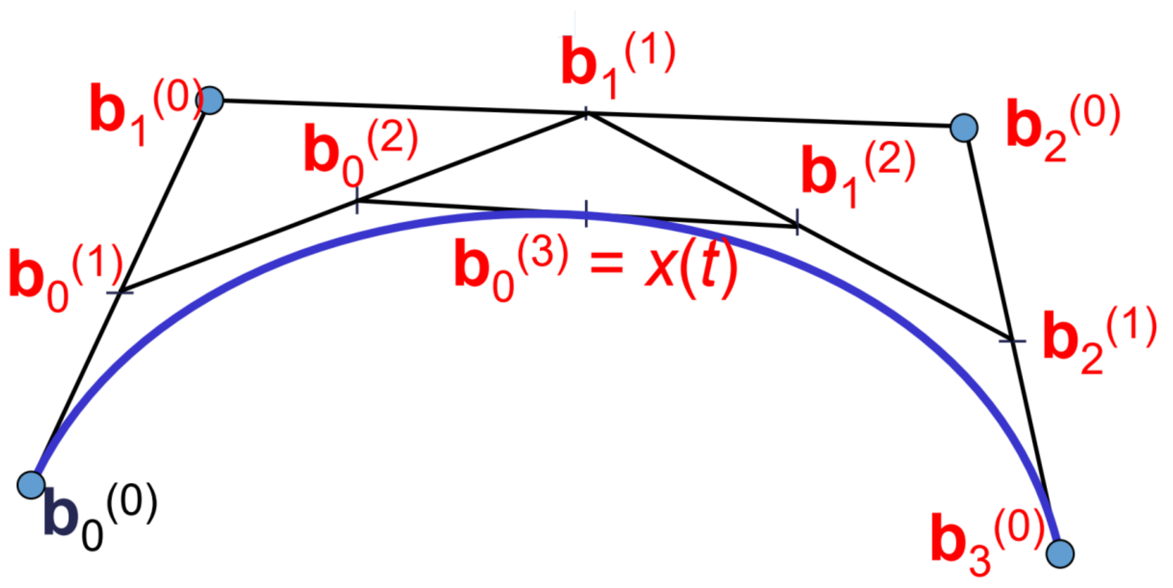


De Casteljau algorithm

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

$$
\begin{array}{llllll}
\boldsymbol{b}_0 = \boldsymbol{b}_0^0 & & & & & \\
\boldsymbol{b}_1 = \boldsymbol{b}_1^0 & \boldsymbol{b}_0^1 & & & & \\
\boldsymbol{b}_2 = \boldsymbol{b}_2^0 & \boldsymbol{b}_1^1 & \boldsymbol{b}_0^2 & & & \\
\boldsymbol{b}_3 = \boldsymbol{b}_3^0 & \boldsymbol{b}_2^1 & \boldsymbol{b}_1^2 & \boldsymbol{b}_0^3 & & \\
 & \ldots\ldots\ldots\ldots\ldots\ldots & & & & \\
\boldsymbol{b}_{n-1} = \boldsymbol{b}_{n-1}^0 & \boldsymbol{b}_{n-2}^1 & \ldots & \boldsymbol{b}_0^{n-1} & & \\
\boldsymbol{b}_n \quad = \boldsymbol{b}_n^0 & \boldsymbol{b}_{n-1}^1 & \ldots & \boldsymbol{b}_1^{n-1} & \boldsymbol{b}_0^n = x(t) &
\end{array}
$$

De Casteljau algorithm



Algorithm:

```
for r=1..n
    for i=0..n-r
        $b_i^{(r)} = (1-t)\, b_i^{(r-1)} + t\, b_{i+1}^{(r-1)}$
    end
end
return $b_0^{(n)}$
```

The whole algorithm consists only of repeated linear interpolations.

De Casteljau algorithm: Properties

- The polygon consisting of the points $\boldsymbol{b_0}, \dots, \boldsymbol{b_n}$ is called Bézier polygon (control polygon)
- The points $\boldsymbol{b_i}$ are called Bézier points (control points)
- The curve defined by the Bézier points $\boldsymbol{b_0}, \dots, \boldsymbol{b_n}$ 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 $\boldsymbol{b}_0, \dots, \boldsymbol{b}_n$

    Bézier curve $\boldsymbol{x}(t)$
  - Bézier curve is polynomial curve of degree n
  - End points interpolation: $\boldsymbol{x}(0) = \boldsymbol{b}_0,\ \boldsymbol{x}(1) = \boldsymbol{b}_n$. 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: $\boldsymbol{b}_i$ 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**

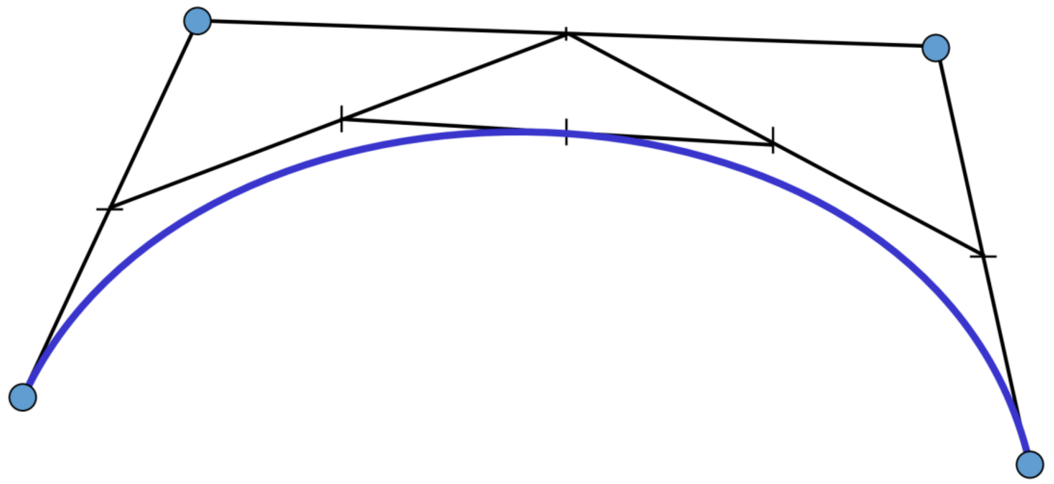
De Casteljau algorithm: Properties

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

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

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

Recap

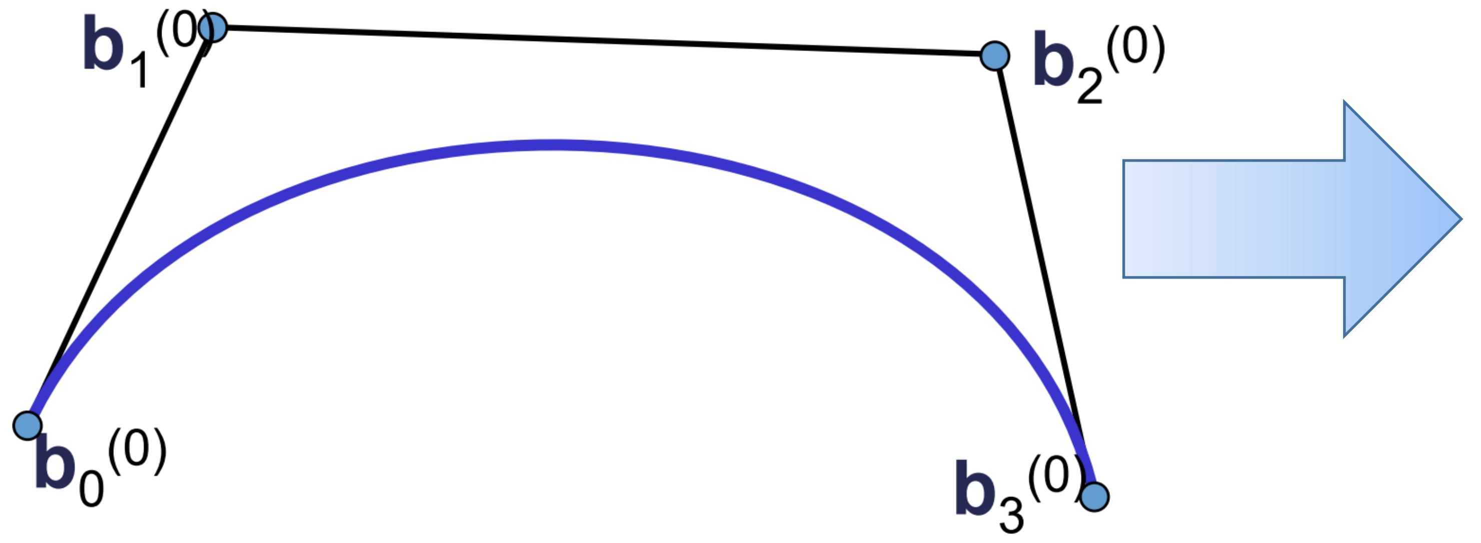


de Casteljau algorithm

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 $\boldsymbol{b}_0, \dots, \boldsymbol{b}_n$
  - Wanted: Bézier 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 $\boldsymbol{x} \rightarrow A\boldsymbol{x} + \boldsymbol{b}$ 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

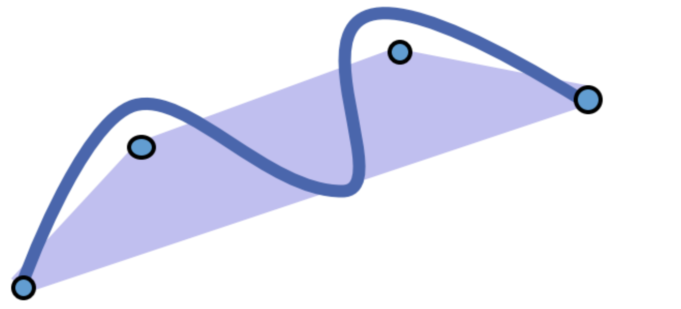
Convex Hull Property

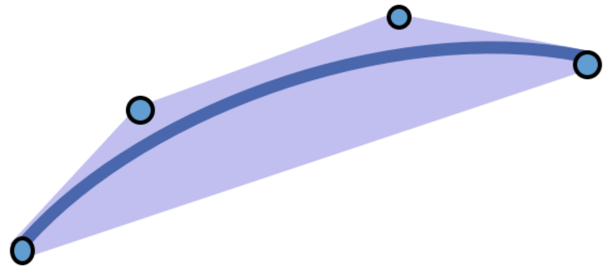
- Convex combinations:
  - A convex combination of a set of points $\{\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\}$ is any point of the form:

$$\sum_{i=1}^{n} \lambda_i \boldsymbol{p}_i \text{ with } \sum_{i=1}^{n} \lambda_i = 1 \text{ and } \forall i = 1 \dots n: 0 \leq \lambda_i \leq 1$$

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

Convex Hull Property



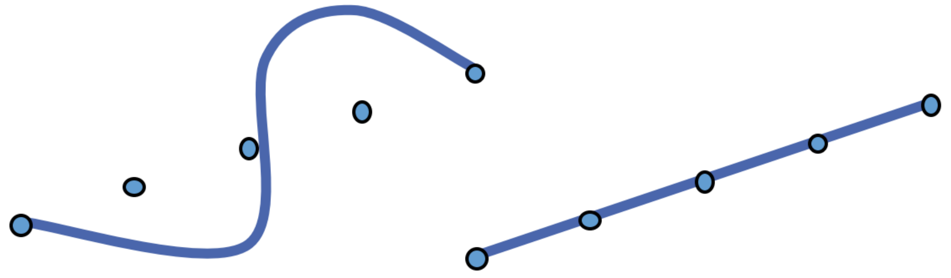


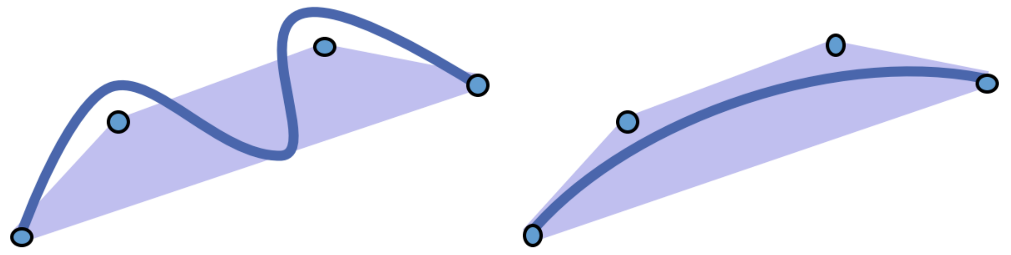
- Accordingly:
  - If we have this property
  $\forall t \in \Omega: \sum_{i=1}^{n} b_i(t) = 1$ and $\forall t \in \Omega, \forall i: b_i(t) \geq 0$
  the constructed curves / surfaces will be:
    - Affine invariant (translations, linear maps)
    - Be restricted to the convex hull of the control points

  - Corollary: Curves will have *linear precision*
    - All control points lie on a straight line
    $\Rightarrow$ 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 $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n$
  - Wanted: Bézier 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 Bézier curve:

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

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

Bernstein Basis

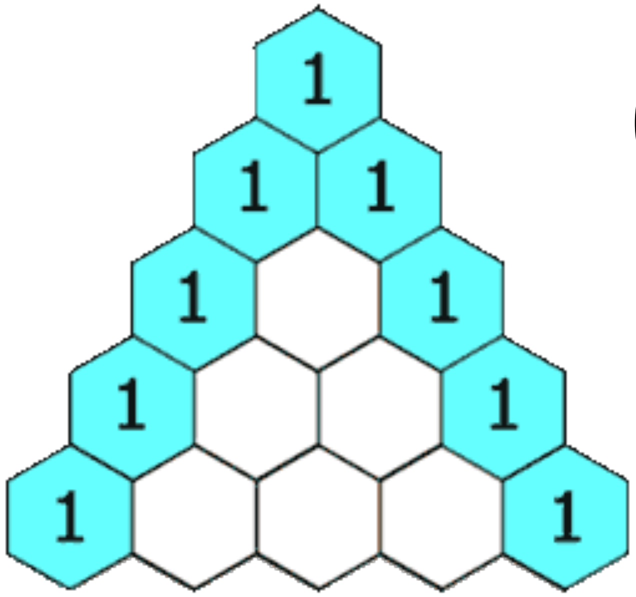
- Let's examine the Bernstein basis: $B = \{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\}$
  - Bernstein basis of degree n:

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

where the binomial coefficients are given by:

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

Binomial Coefficients and Theorem



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

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

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

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

Examples: The first few

- The first three Bernstein bases:

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

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

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

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

Examples: The first few

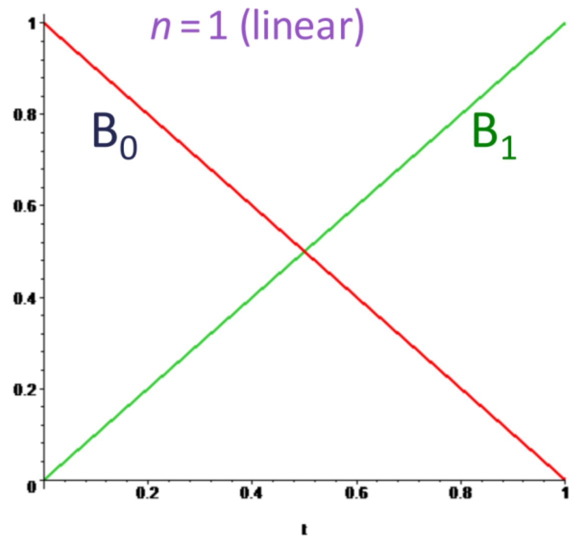
$$B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

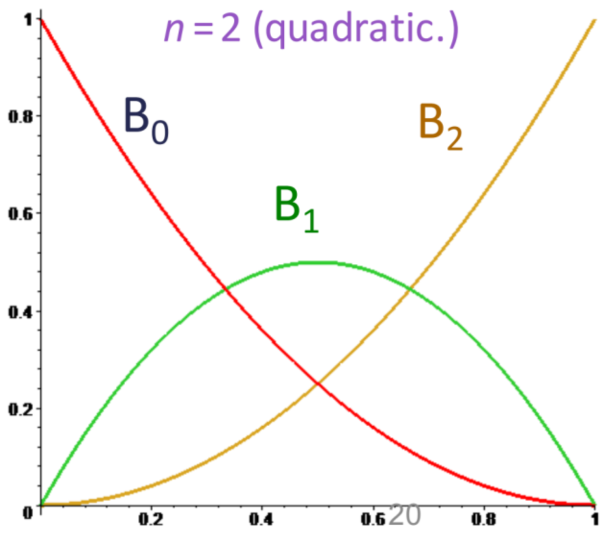
$$B_0^{(0)} := 1$$

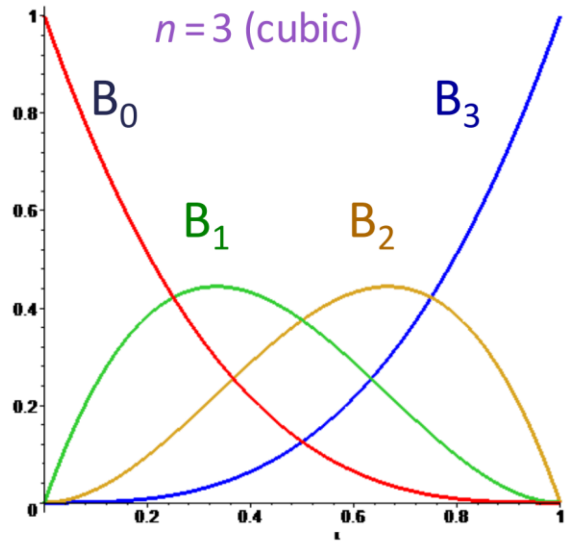
$$B_0^{(1)} := 1 - t$$
$$B_1^{(1)} := t$$

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

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



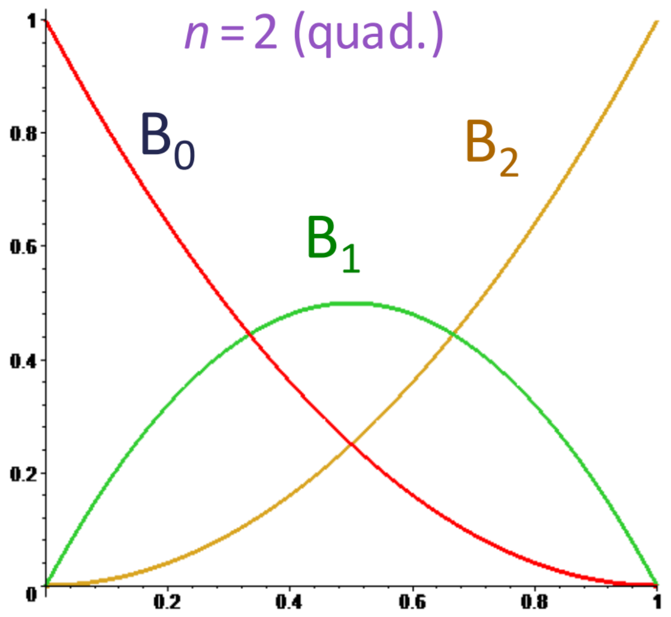


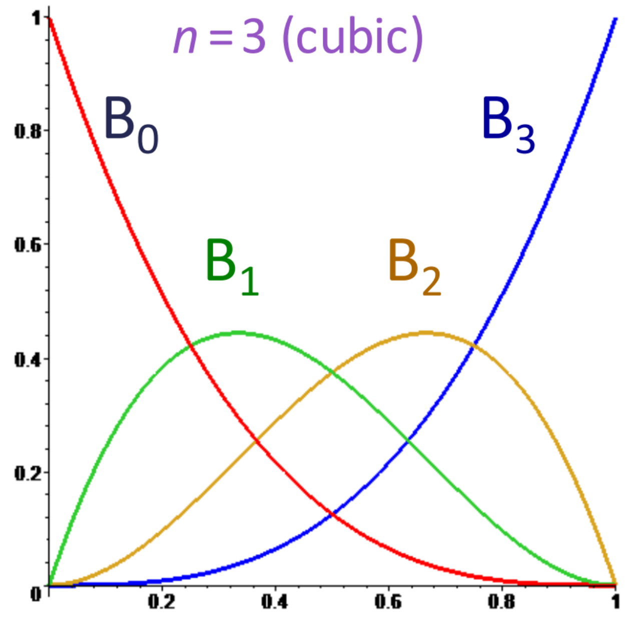


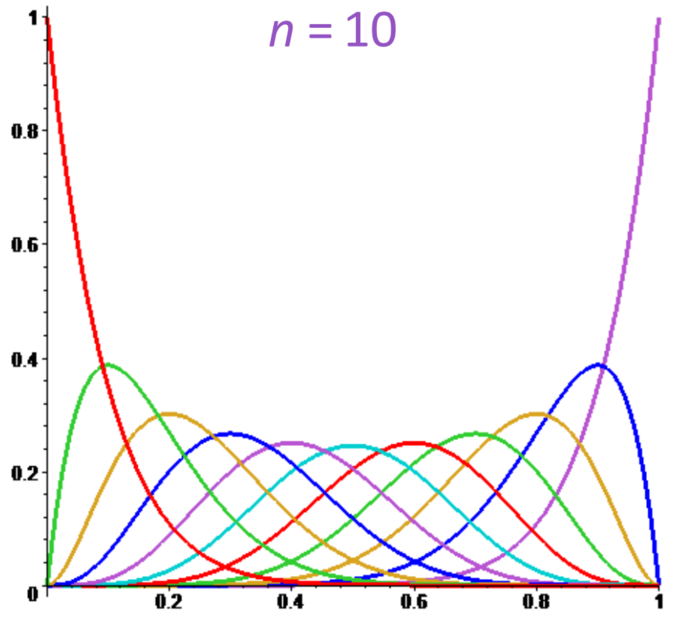
Bernstein Basis

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

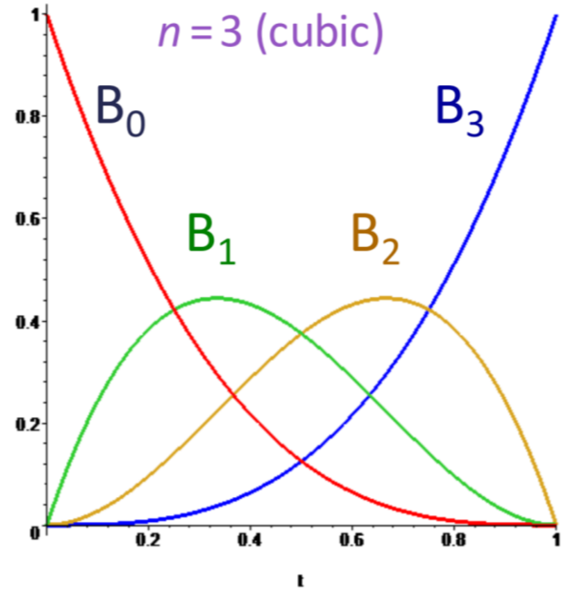
- What about the desired properties?
  - Smoothness
  - Local control / support
  - Affine invariance
  - Convex hull property

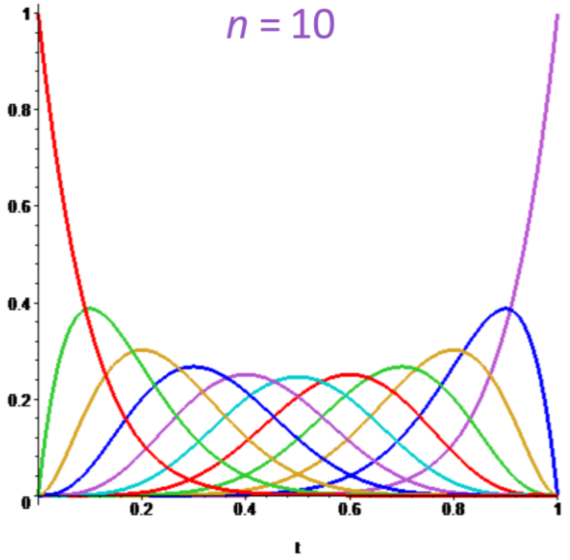
Bernstein Basis: Properties

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

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

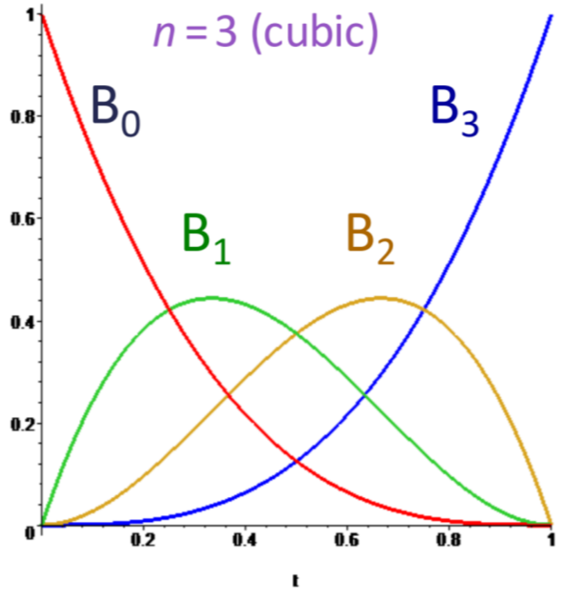
Affine invariance

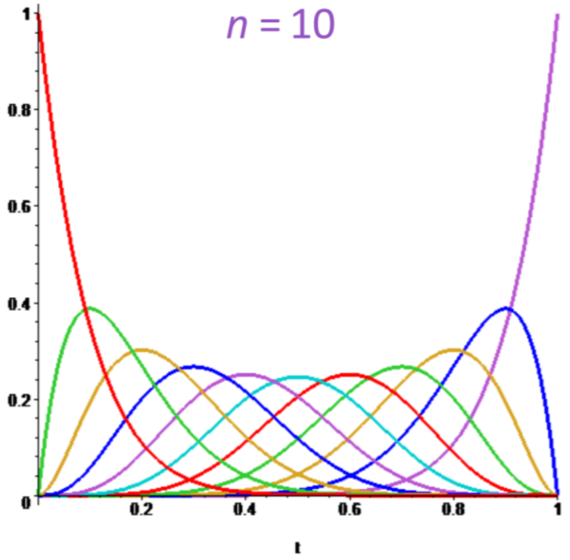
Convex hull property

- 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

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

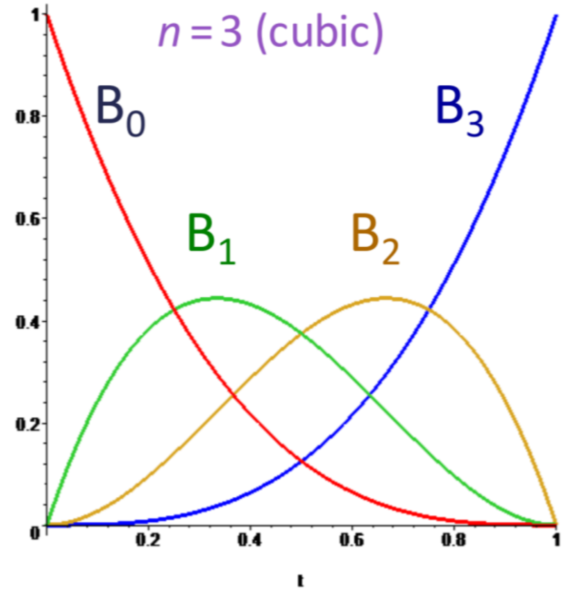
$$B_i^n(t) := (1-t)B_i^{(n-1)}(t) + tB_{i-1}^{(n-1)}(1-t)$$

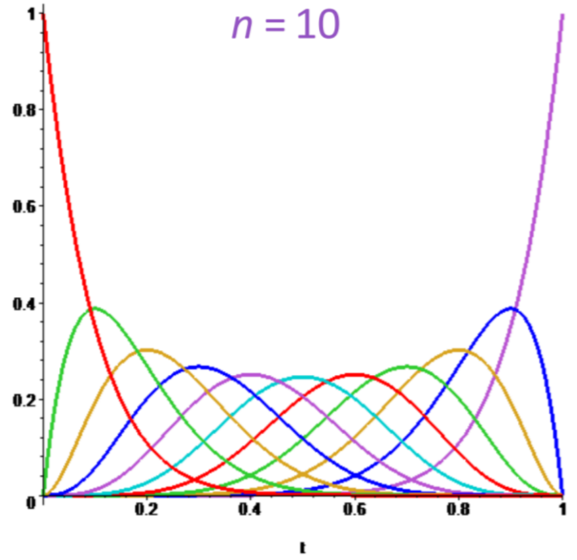
with $B_0^0(t) = 1, B_i^n(t) = 0$ for $i \notin \{0 \dots n\}$

- 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

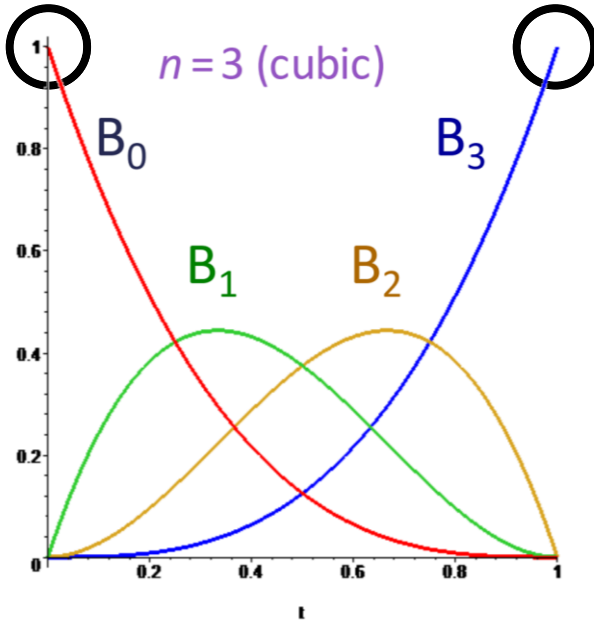
- $B = \left\{B_0^{(n)}, B_1^{(n)}, \dots, B_n^{(n)}\right\}$, $B_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

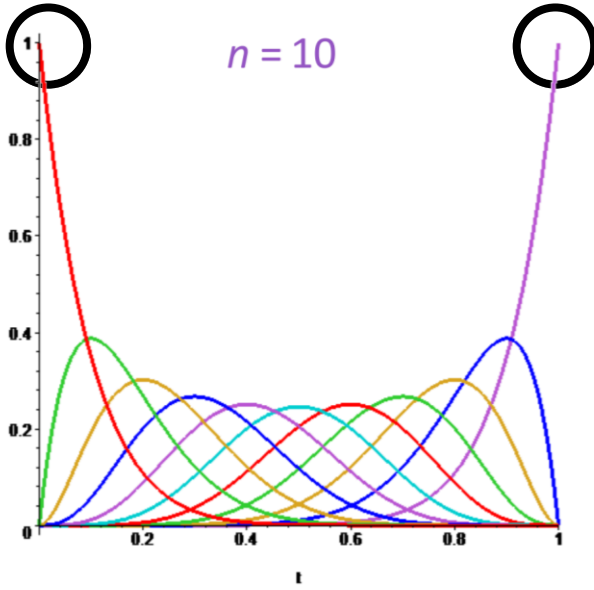
- Non-negativity II

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

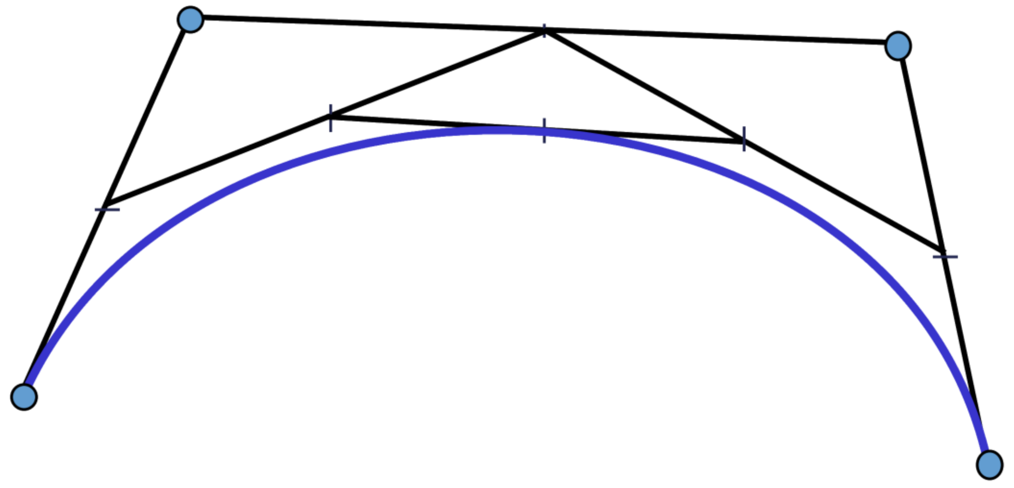
$$B_0^n(0) = 1, \qquad B_1^n(0) = \cdots = B_n^n(0) = 0$$

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

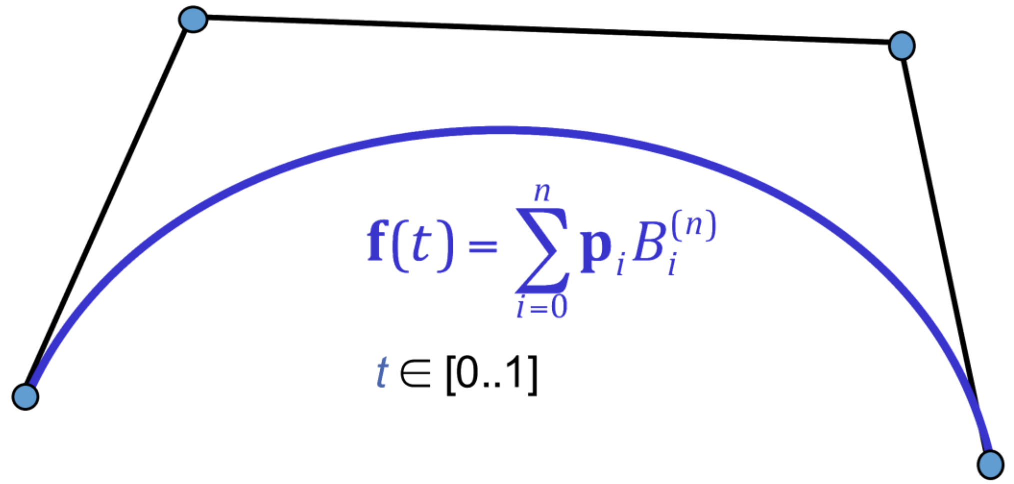




Recap



de Casteljau algorithm

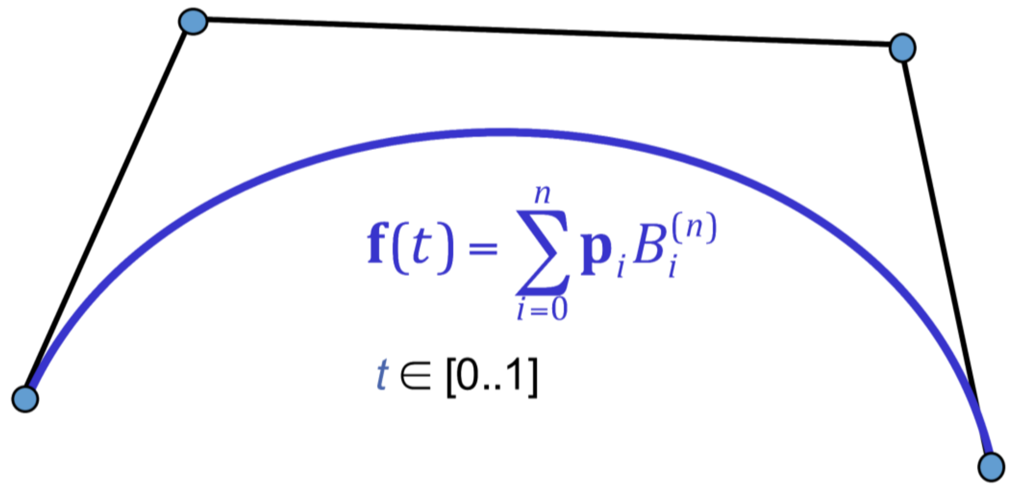


$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_i^{(n)}$$

$t \in [0..1]$

Bernstein form

Recap



Bernstein form

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

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: $\overline{\boldsymbol{b}}_0, \dots, \overline{\boldsymbol{b}}_n, \overline{\boldsymbol{b}}_{n+1} \rightarrow \overline{\boldsymbol{x}}(t)$ with $\boldsymbol{x} = \overline{\boldsymbol{x}}$

- Solution:

$$\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$$
$$\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$$
$$\overline{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j \quad \text{for } j = 1, \dots, n$$

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

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

Proof

Using results from last slide

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

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

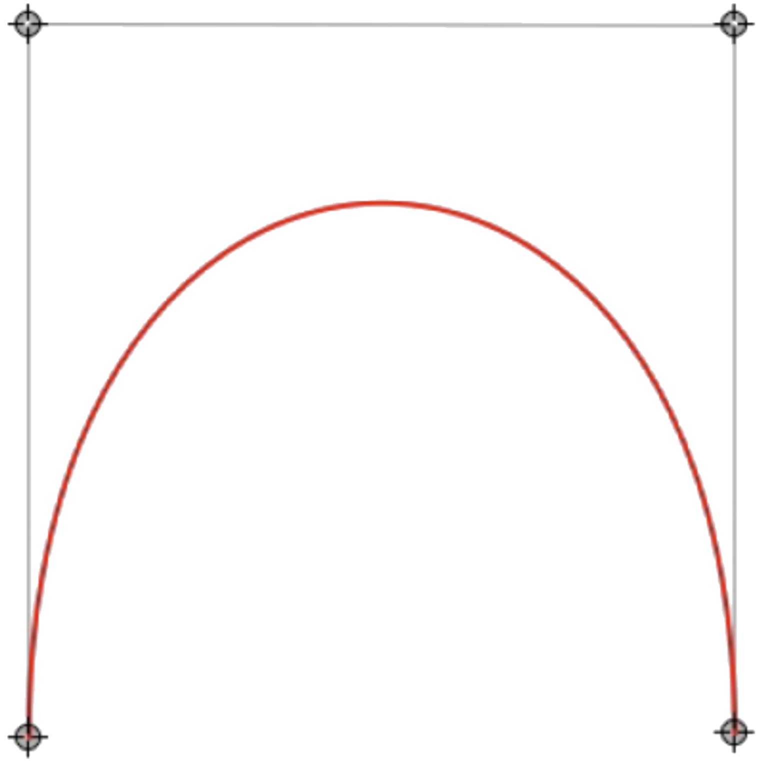
$$= \sum_{i=0}^{n}\frac{n+1-i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_i + \sum_{i=1}^{n+1}\frac{i}{n+1}B_i^{n+1}(t)\boldsymbol{P}_{i-1}$$

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

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

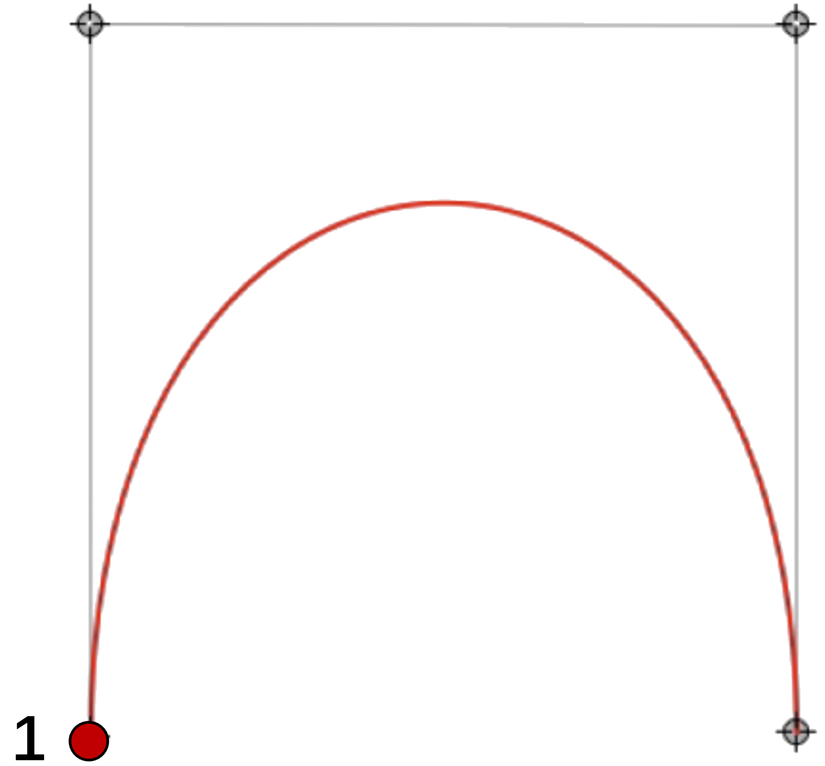


- $\bar{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\bar{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\bar{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j$$

$$j = 1, \ldots, n$$

Degree elevation: Example

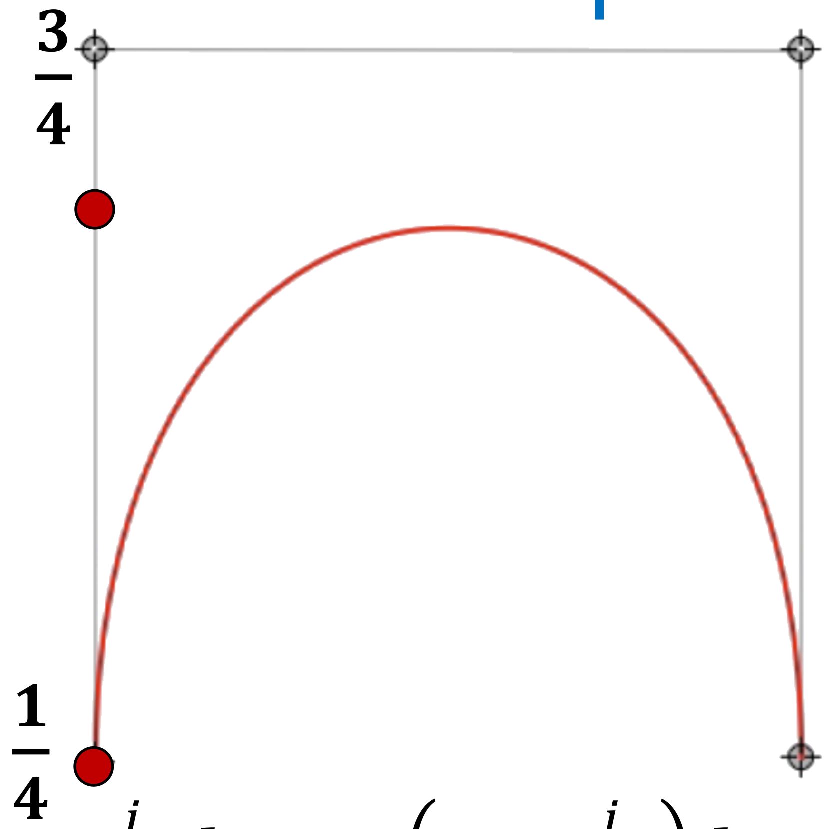


- $\bar{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\bar{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\bar{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j$$

$$j = 1, \dots, n$$

Degree elevation: Example

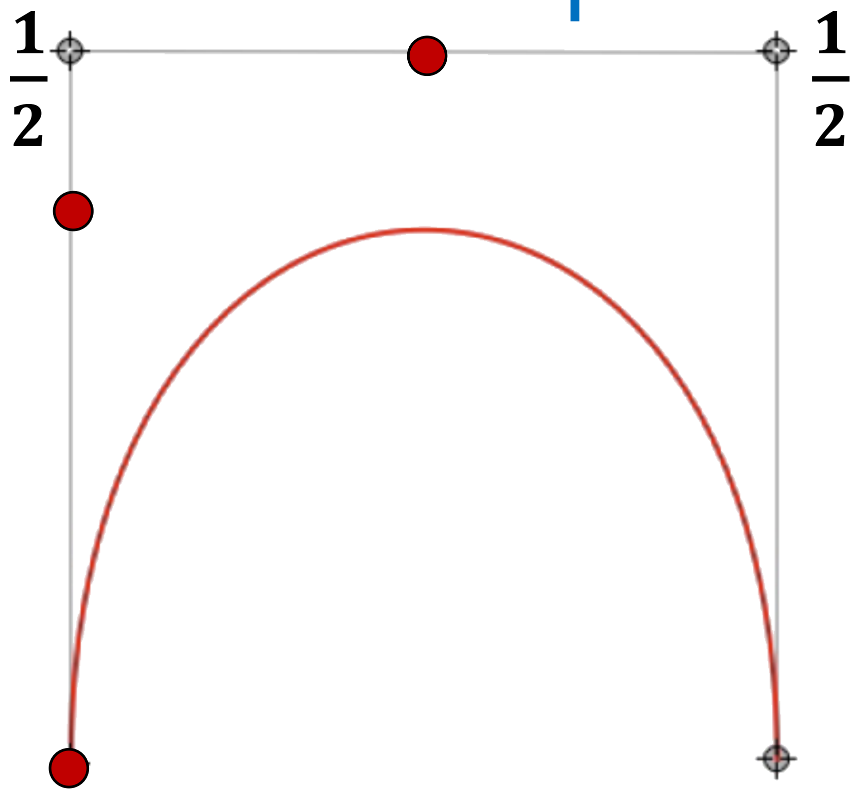


- $\bar{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\bar{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

$$\bar{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j$$

$$j = 1, \dots, n$$

Degree elevation: Example

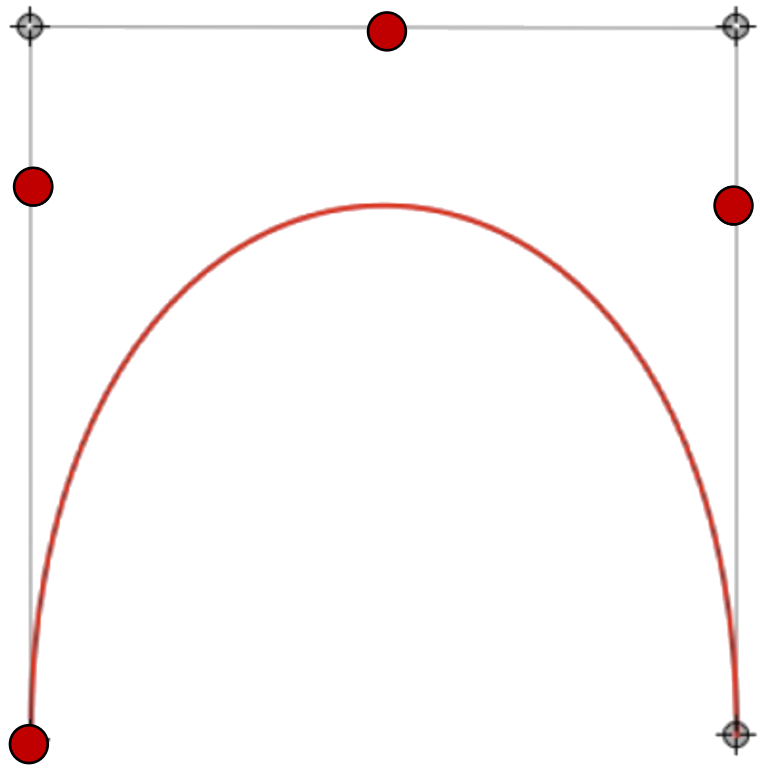


- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

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

$$j = 1, \ldots, n$$

Degree elevation: Example



$$\frac{1}{4}$$

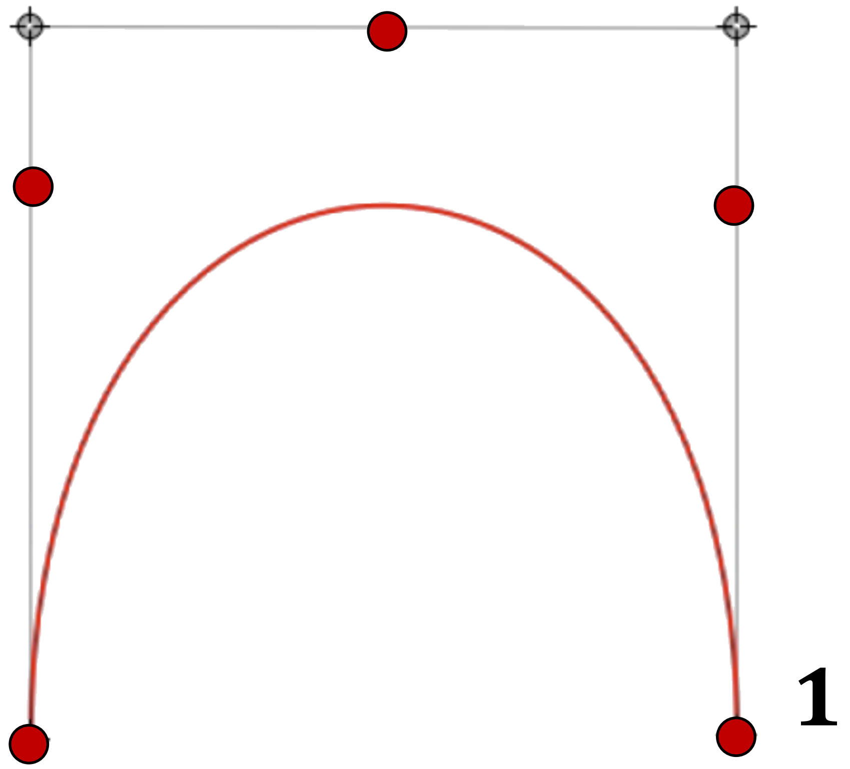
$$\frac{3}{4}$$

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

$$\bar{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j$$

$$j = 1, \dots, n$$

Degree elevation: Example

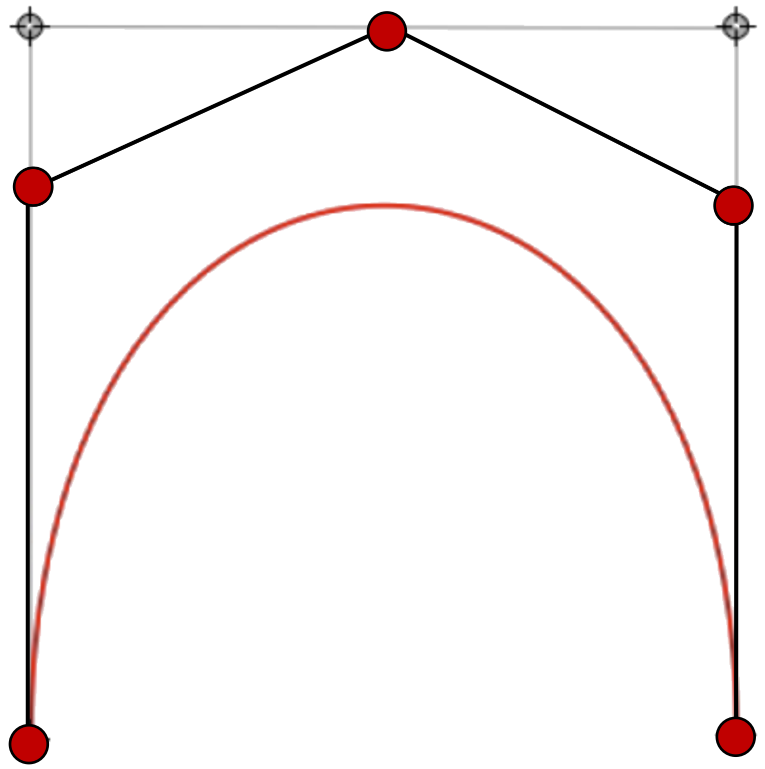


- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

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

$$j = 1, \ldots, n$$

Degree elevation: Example

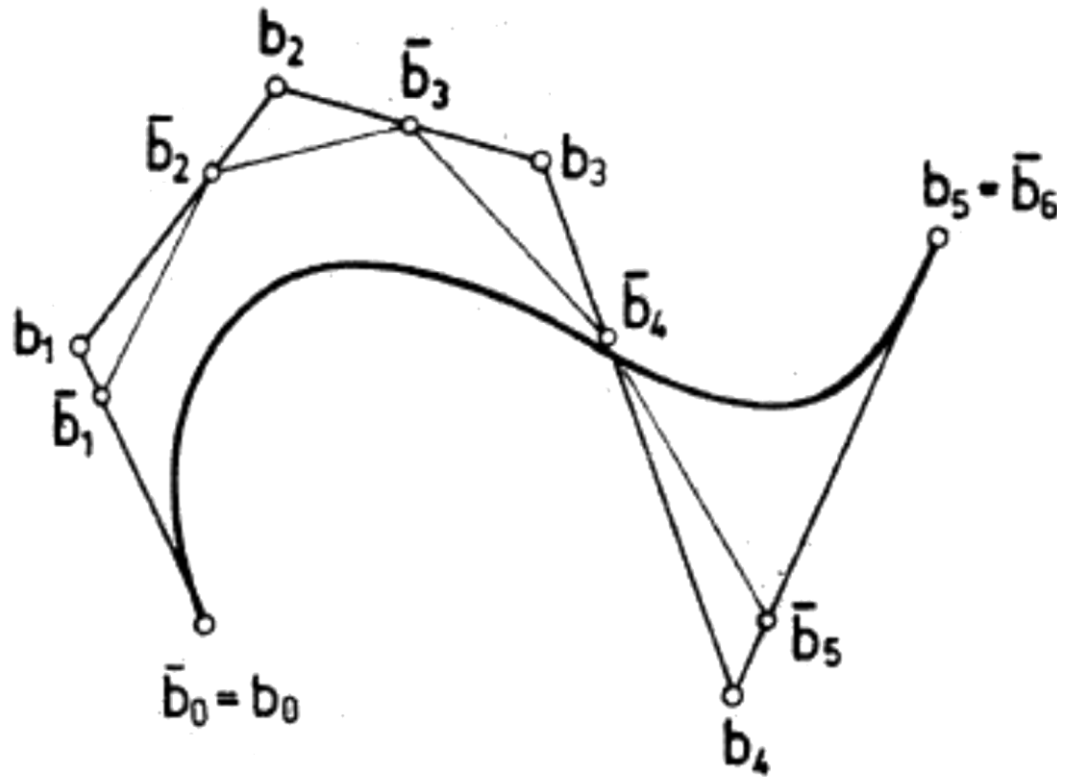


- $\overline{\boldsymbol{b}}_0 = \boldsymbol{b}_0$
- $\overline{\boldsymbol{b}}_{n+1} = \boldsymbol{b}_n$

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

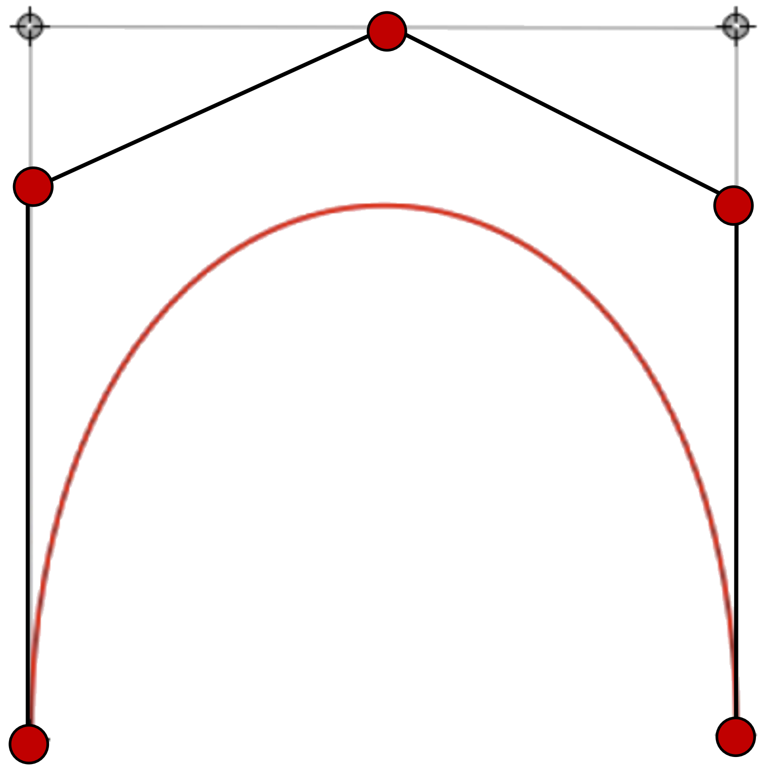
$$j = 1, \dots, n$$

Degree elevation



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

Degree elevation



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

$$\bar{\boldsymbol{b}}_j = \frac{j}{n+1}\boldsymbol{b}_{j-1} + \left(1 - \frac{j}{n+1}\right)\boldsymbol{b}_j$$

$$j = 1, \ldots, n$$

Bézier Curves

Subdivision

Subdivision

- Given: $b_0, \dots, b_n \rightarrow x(t), t \in [0,1]$

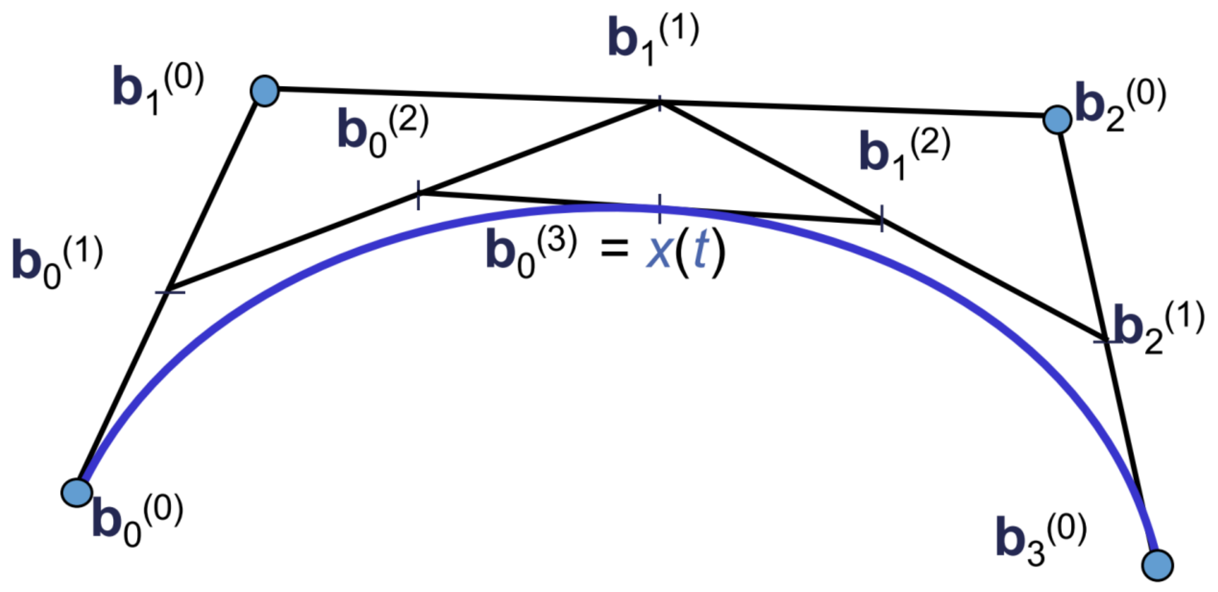
- Wanted: $b_0^{(1)}, \dots, b_n^{(1)} \rightarrow x^{(1)}(t),$
  $b_0^{(2)}, \dots, b_n^{(2)} \rightarrow x^{(2)}(t),$

  with $x = x^{(1)} \cup x^{(2)}$

Subdivision: Example

$$\begin{tikzcd}
\mathbf{b}_0^{(0)} \arrow[dr, "1-t"] & & & \\
\mathbf{b}_1^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_0^{(1)} \arrow[dr, "1-t"] & & \\
\mathbf{b}_2^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_1^{(1)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_0^{(2)} \arrow[dr, "1-t"] & \\
\mathbf{b}_3^{(0)} \arrow[r, "t"] & \mathbf{b}_2^{(1)} \arrow[r, "t"] & \mathbf{b}_1^{(2)} \arrow[r, "t"] & \mathbf{b}_0^{(3)} = x(t)
\end{tikzcd}$$

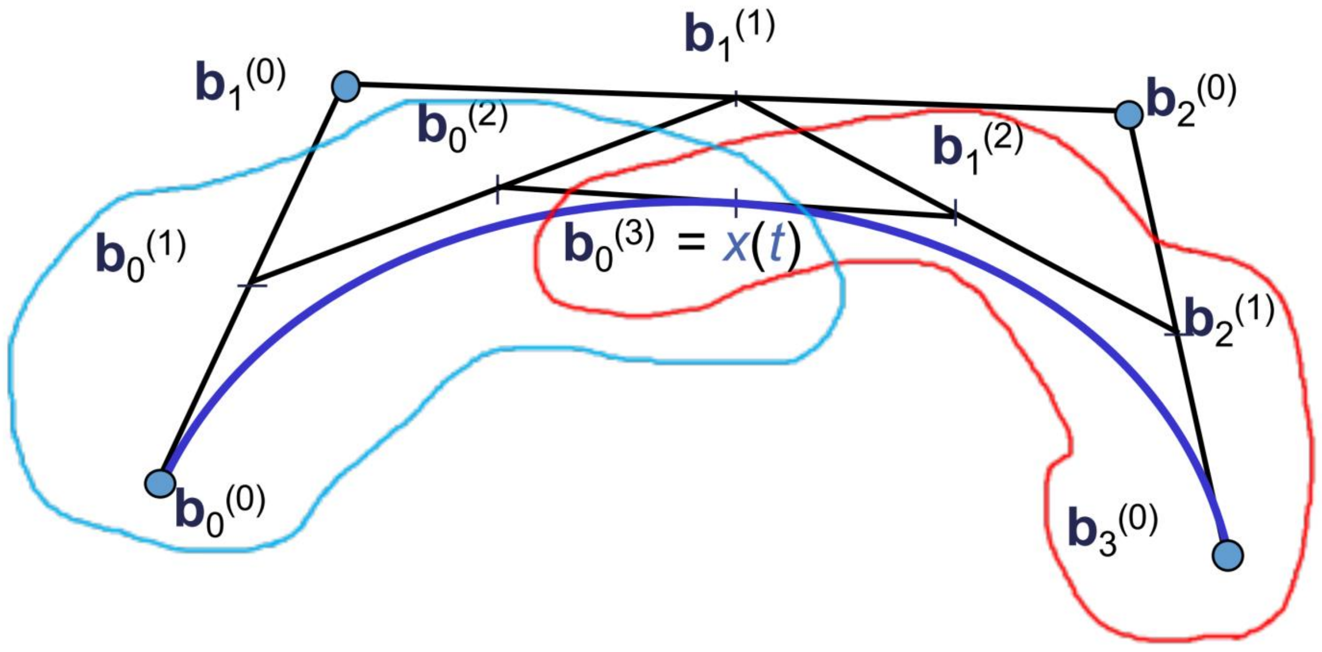
de Casteljau scheme



Subdivision: Example

$$\begin{tikzcd}
\mathbf{b}_0^{(0)} \arrow[dr, "1-t"] & & & \\
\mathbf{b}_1^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_0^{(1)} \arrow[dr, "1-t"] & & \\
\mathbf{b}_2^{(0)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_1^{(1)} \arrow[r, "t"] \arrow[dr, "1-t"] & \mathbf{b}_0^{(2)} \arrow[dr, "1-t"] & \\
\mathbf{b}_3^{(0)} \arrow[r, "t"] & \mathbf{b}_2^{(1)} \arrow[r, "t"] & \mathbf{b}_1^{(2)} \arrow[r, "t"] & \mathbf{b}_0^{(3)} = x(t)
\end{tikzcd}$$

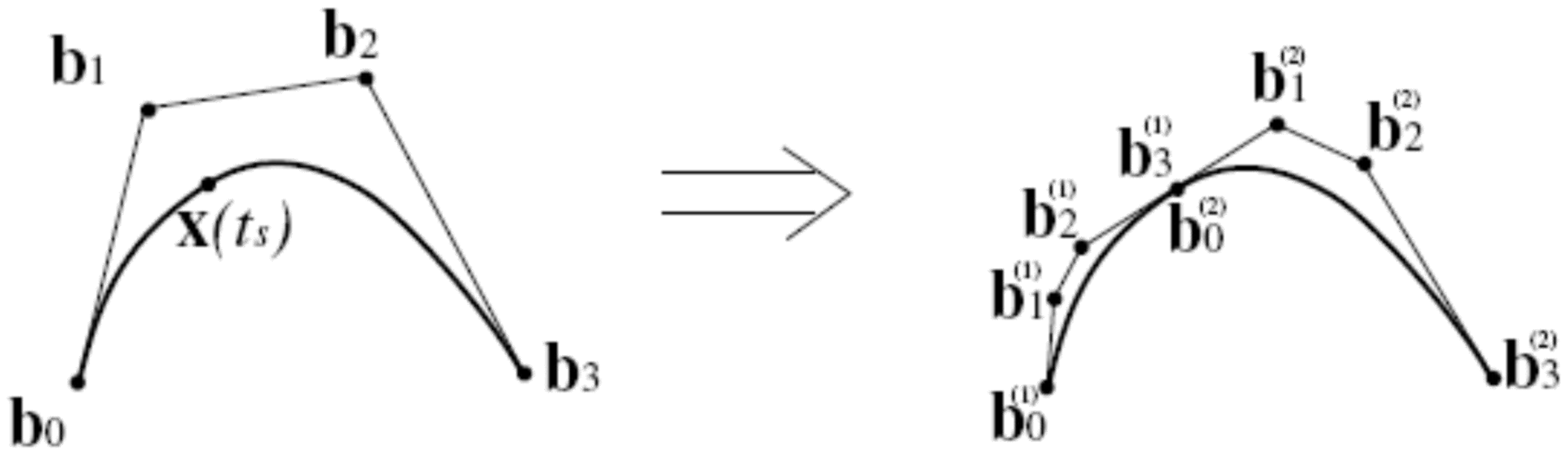
de Casteljau scheme



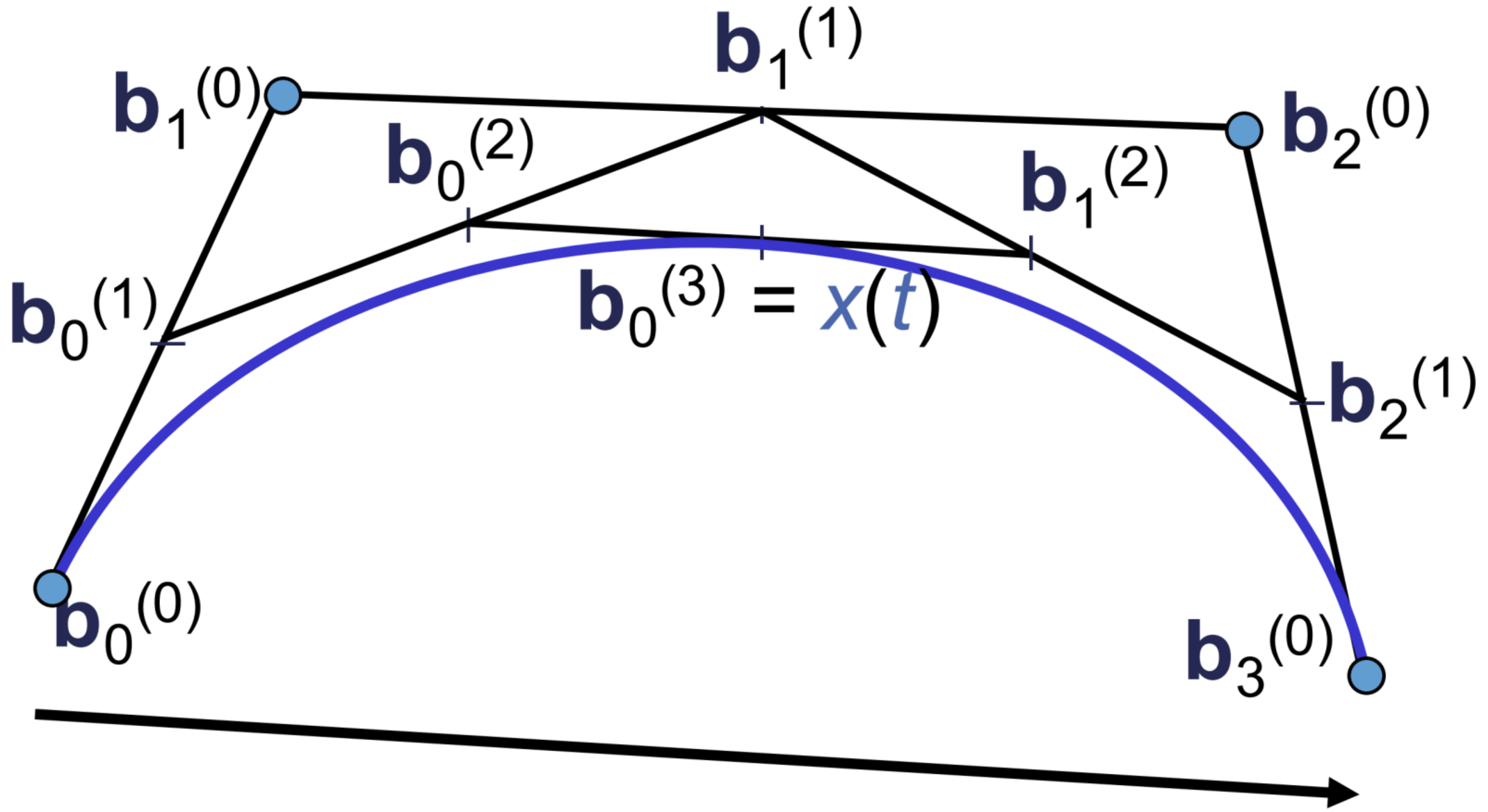
Subdivision

Solution: $b_i^{(1)} = b_0^i,\ b_i^{(2)} = b_0^{n-i}$ for $i = 0, \dots, n$

That means that the new points are intermediate points of the de Casteljau algorithm!

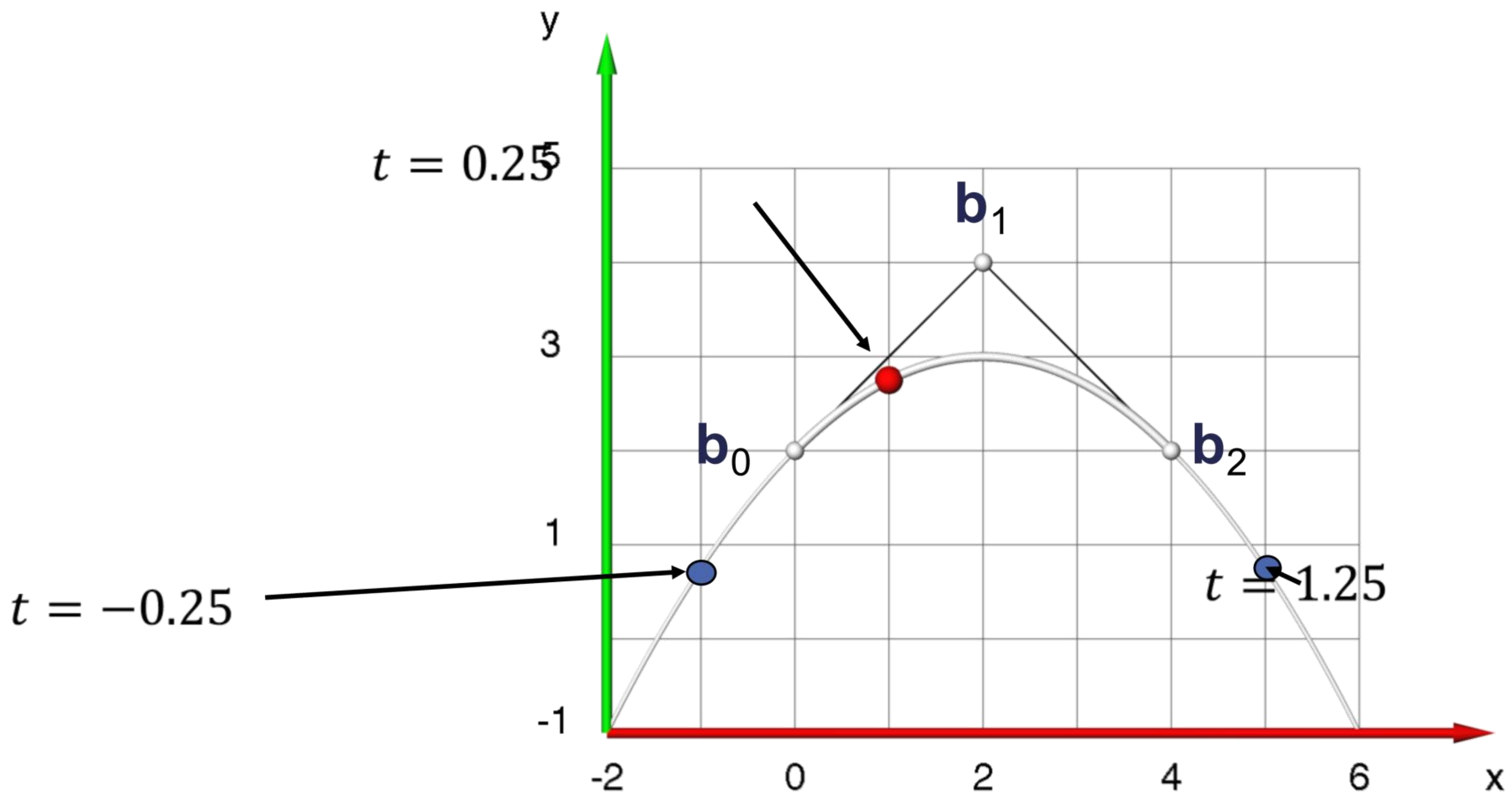


Curve range



parameterization: $t \in [0,1]$

Curve range



Summary & Outlook

- Bézier 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
    - →**Bézier splines**