Outline

- Interpolation and polynomial approximation
- Interpolation
  - Lagrange
  - Cubic Splines

Approximation

  - Bézier curves
  - B-Splines

Outline

- Approximation
  - Bézier curves
  - B-Splines
  - We still focus on curves for the moment.

Bézier curves

- Bézier curves
  - Pierre Bézier (1910-1999)
  - Develops UNISURF –
    first surface modelling software
    at Renault's (1971)

  - Publicizes the theory under his name in 1962... however, the principle was discovered in 1959 by Paul de Casteljau (at Citroën's) ! Because of the “culture of secret” at Citroën, De Casteljau will have his works recognized only in 1975.

Bézier curves

- Use of Bézier curves :
  - Postscript fonts (cubic Bézier) & TrueType (quadratic Bézier)

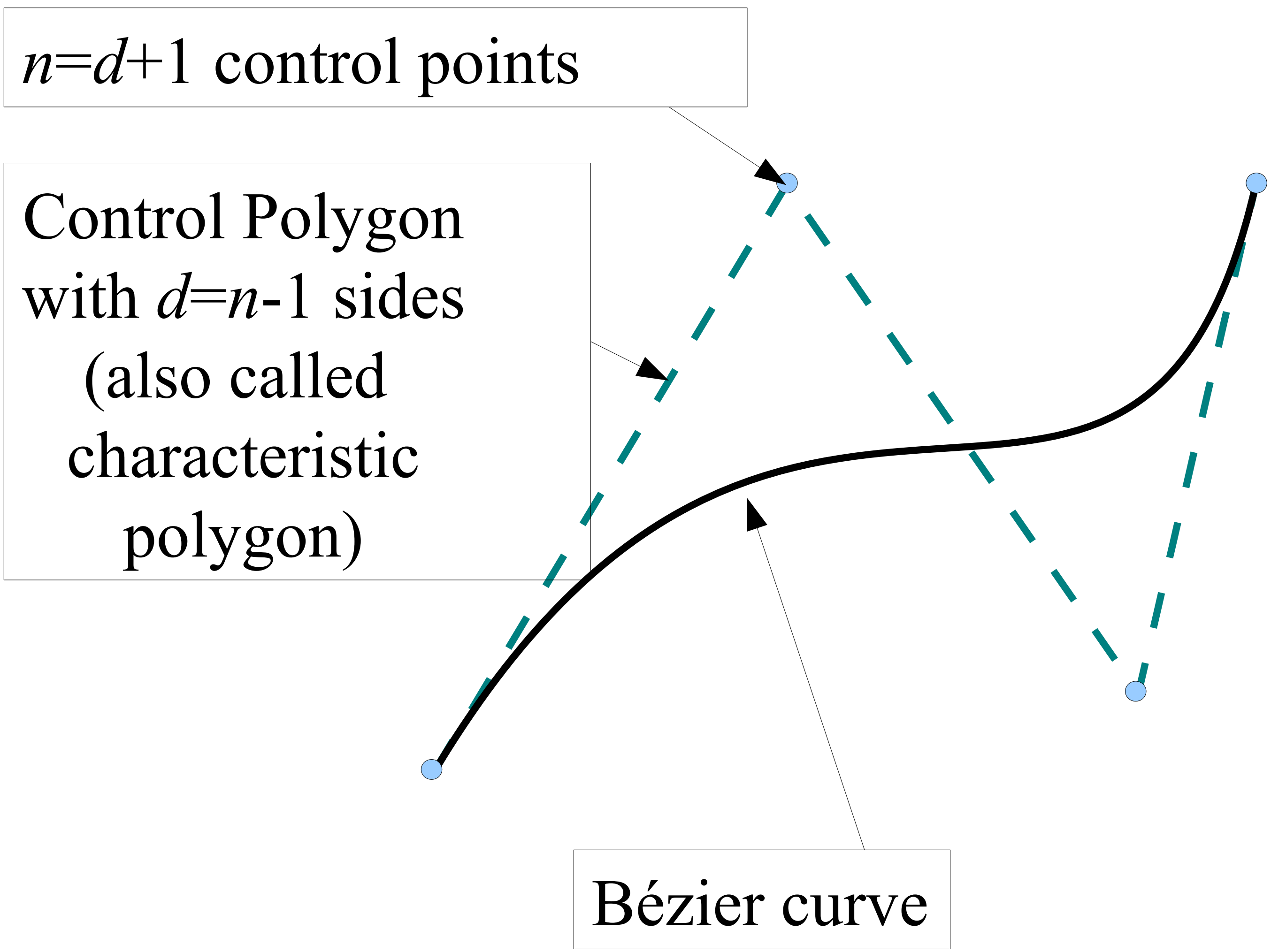
    AaBbCc

  - Computer graphics
  - In geometrical modeling and CAD, they tend to be replaced by more general techniques (NURBS, a.k.a B-Splines in homogeneous coordinates)

Bézier curves

- Modelling by interpolation is not very practical
  - We seldom have interpolation points at our hand
    - Instead, we hope to define these points as the result of a modeling process instead of as an input data
  - Approximation gives more freedom in the design of the curve

- Elements of a Bézier curve :



For Bézier curves, the notion of knot is trivial :

$$u_0 = 0 \qquad u_1 = 1$$

Bézier curves

- Characteristics of Bézier curves
  - More freedom than interpolation
    - Any degree
    - Precise control of the curve's shape
    - Numerical stability even with high degree (not as Lagrange !)

$$P(u)=\sum_{i=0}^{d} P_i B_i^d(u)$$

    - The $B_i^d(u)$ are Bernstein polynomials (Sergei N. Bernstein, 1880-1968 - don't mistake for Leonard Bernstein...:) :
    - They form a complete polynomial basis
    - They are a partition of the unity
    - Sometimes called ‘blending functions’
    - The curve is described as one polynomial (unlike splines)

- Bernstein polynomials

$$B_i^d(u) = \binom{d}{i} u^i (1-u)^{d-i}$$

Binomial coefficients

Bézier curves

- Binomial coefficients : computed with Pascal's triangle

$$\binom{d}{i}$$

d=0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1
	i=0	1	2	3	4	5

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

$$\binom{d}{i} = \frac{d!}{(d-i)!\,i!}$$

- Bernstein polynomials

$$B_i^d(u) = \binom{d}{i} u^i (1-u)^{d-i}$$

Binomial coefficients

$$1 = \left[(1-u)+u\right]^d = [A+B]^d = \sum_{i=0}^{d} \binom{d}{i} A^i B^{d-i} = \sum_{i=0}^{d} \binom{d}{i} u^i (1-u)^{d-i}$$

$$= \sum_{i=0}^{d} B_i^d(u)$$

  - By design, they form a partition of unity...

- Some characteristics of the B. polynomials.
  - $B_i^d(u)=0$ if $i<0$ or $i>d$
  - $B_i^d(0)=\delta_{i0}$ and $B_i^d(1)=\delta_{id}$
  - $B_i^d(u)$ has a root of multiplicity i for $u=0$
  - $B_i^d(u)$ has a root of multiplicity $d-i$ for $u=1$
  - $B_i^d(u)\geq 0$ for $u\in[0,1]$
  - $B_i^d(1-u)=B_{d-i}^d(u)$ (symmetry of the basis)
  - $B_i^{'d}=d\left(B_{i-1}^{d-1}(u)-B_i^{d-1}(u)\right)$
  - If $i\neq 0$, $B_i^d(u)$ has a <u>unique</u> maximum at $u=i/d$

$$B_i^d(i/d)=i^i d^{-d}(d-i)^{(d-i)}\binom{d}{i}$$

- Recurrence relations of Bernstein's basis

$$B_i^d(u) = (1-u)\,B_i^{d-1}(u) + u\,B_{i-1}^{d-1}(u)$$

$$B_d^d(u) = u\,B_{d-1}^{d-1}(u) \qquad B_0^d(u) = (1-u)\,B_0^{d-1}(u)$$

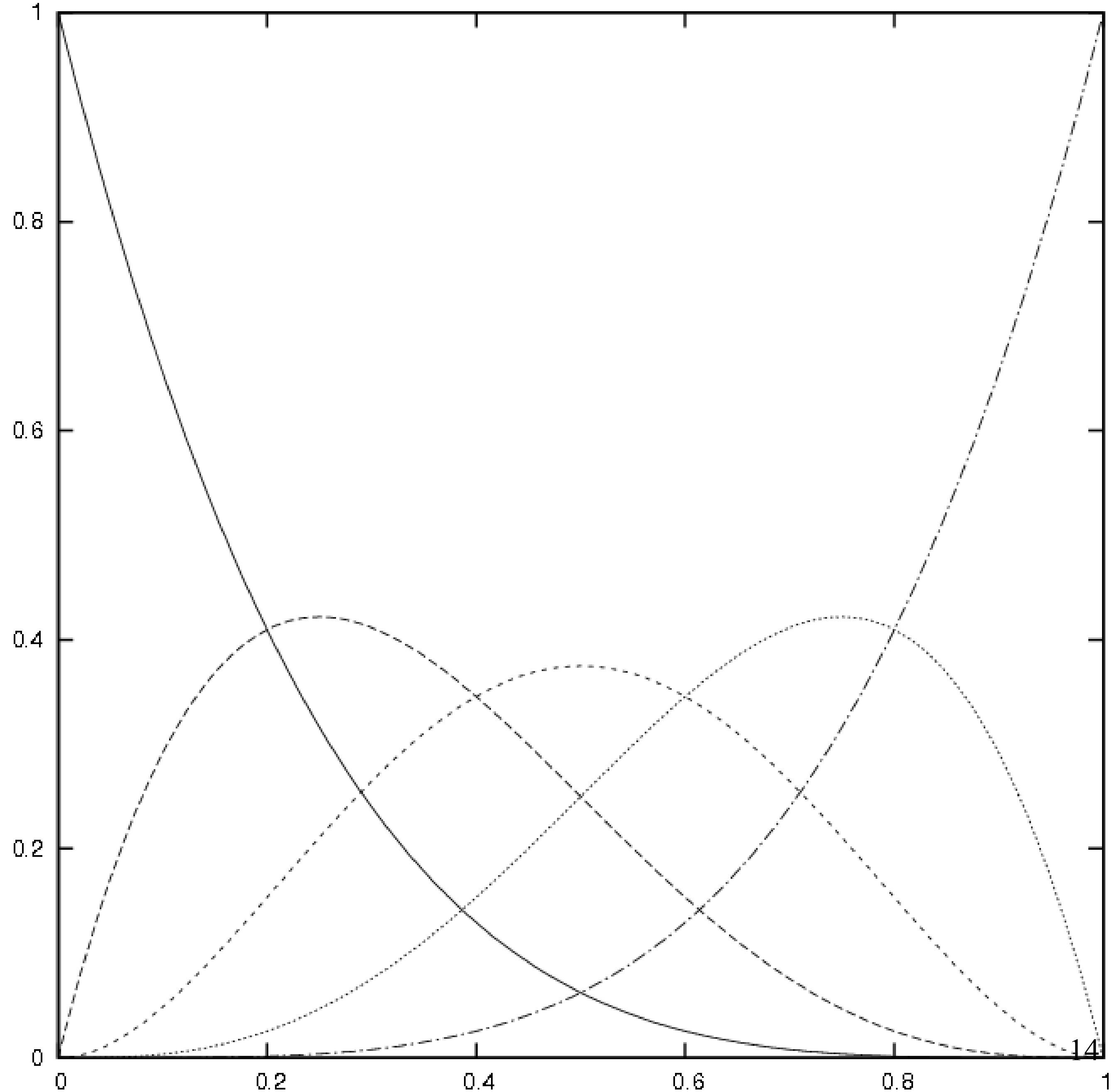
... but no practical interest other than demonstrating algebraic relations (cf. following)

- These polynomials are usually not computed explicitly

Bézier curves

- Degree 4
- No negative values

  Therefore, no value above 1!

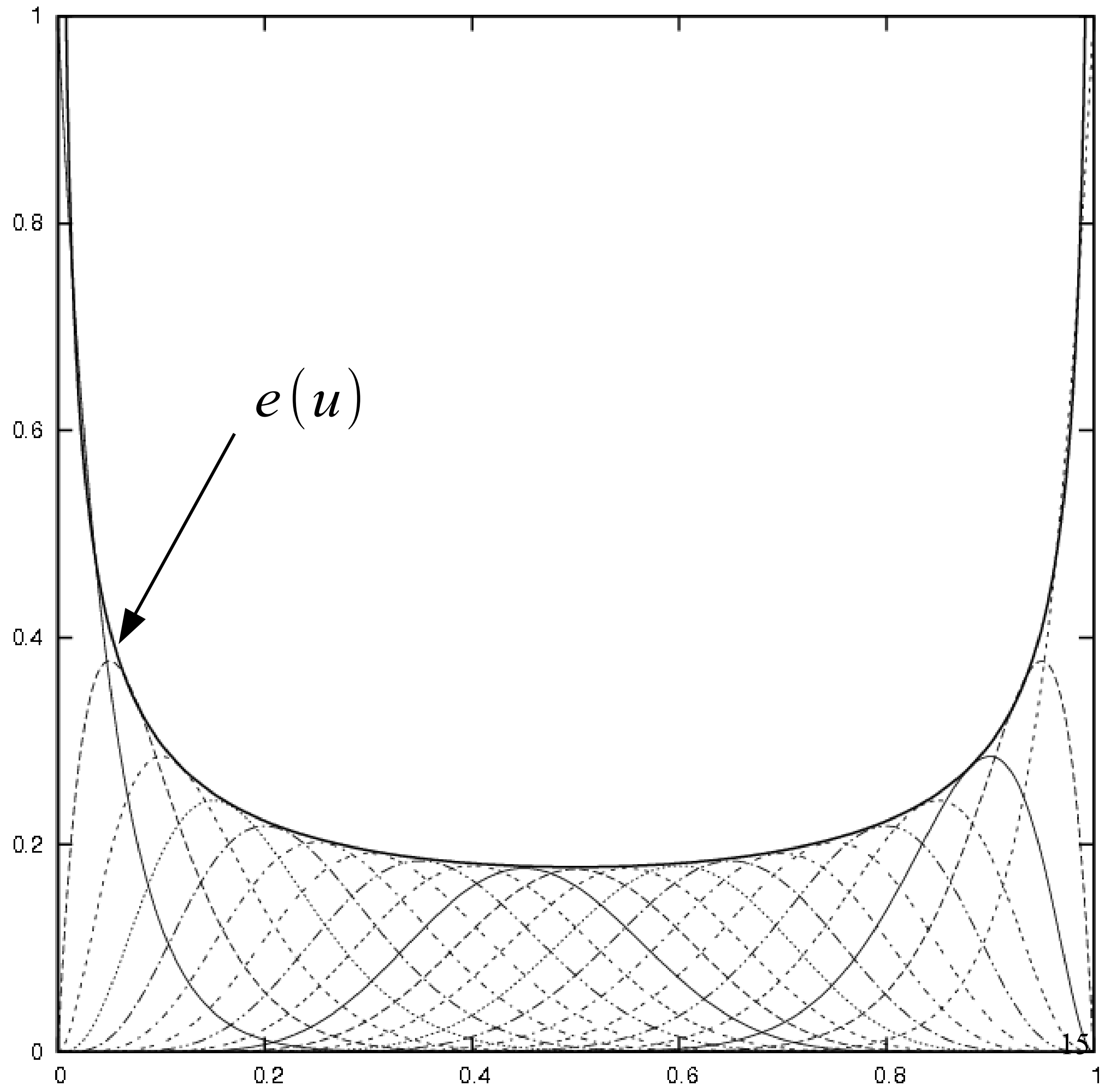


Bézier curves

- Degree 20
- No extreme values

- Existence of a limit envelope

$$e(u)=\frac{1}{\sqrt{2d\pi u(1-u)}}$$

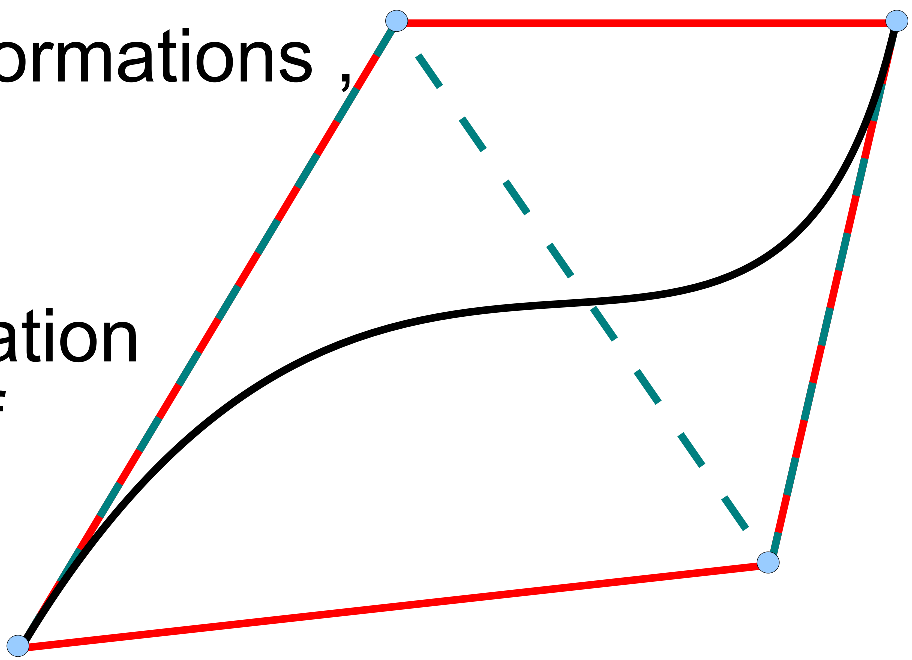


Bézier curves

- The characteristics of Bernstein polynomials involve that the Bézier curve

$$P(u)=\sum_{i=0}^{d} P_i B_i^d(u):$$

  - interpolates P_0 and P_d ,
  - is invariant by affine transformations ,
  - is contained in the convex hull of its control points (because $P(u)$ is a combination with positive coefficients of control points – also called convex combination) ,



Bézier curves

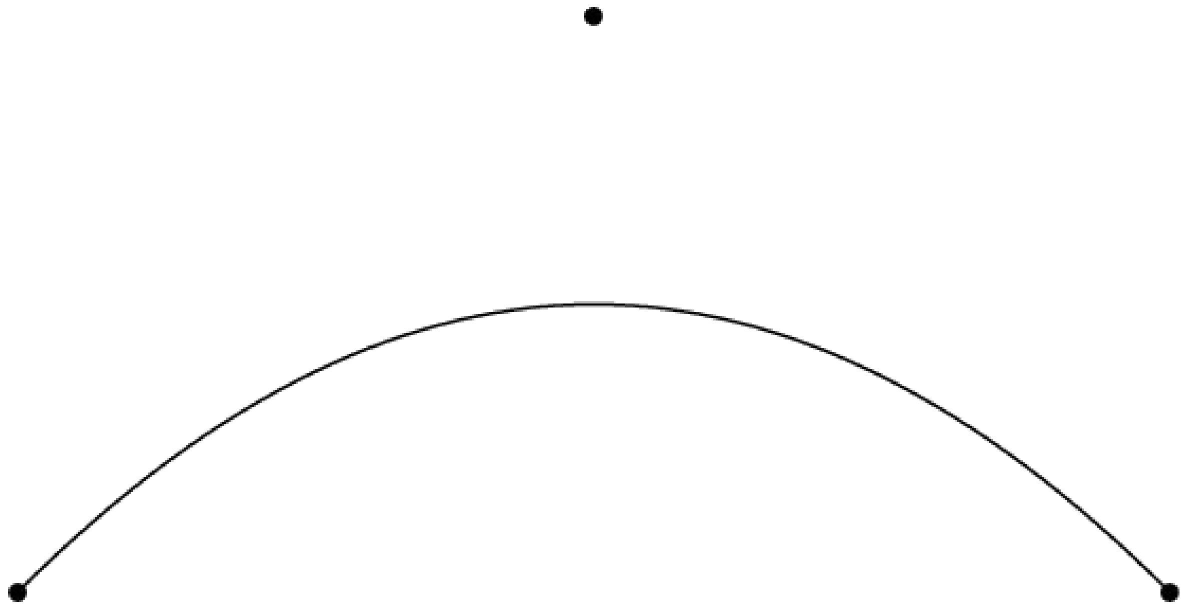
(following)

- is *variation diminishing* : the curve has less inflexion points (wiggles) than there are undulations of the characteristic polynomial (proof by the fact that a Bézier curve is obtained by recursive subdivision, see further) ,
- delimits a closed convex domain if the control polygon itself is convex and closed... ,
- Its length is smaller than that of the control polygon.

Bézier curves

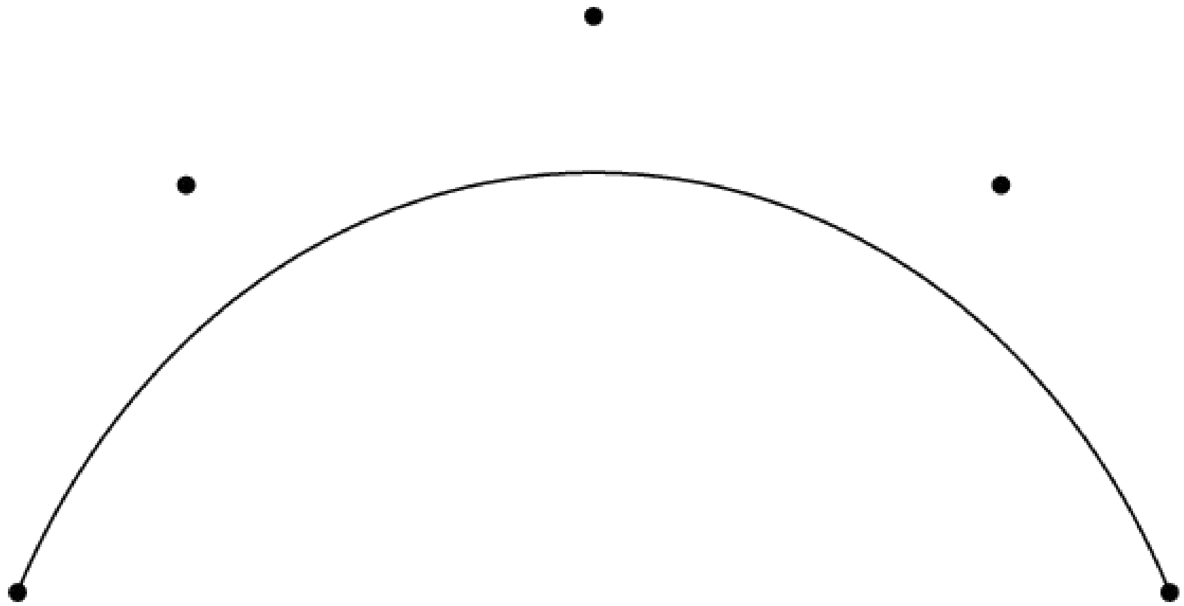
- Same examples as shown earlier on Lagrange interpolation
  - Circle with an increasing number of points
  - Perturbation of the control points

Bézier curves



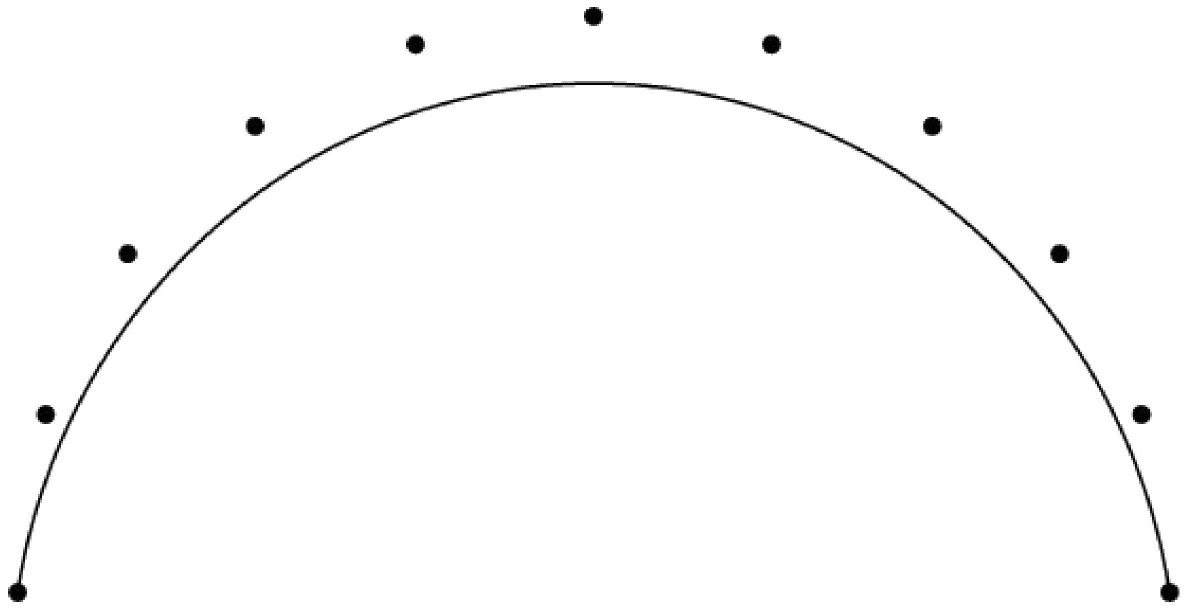
Degree 2

Bézier curves



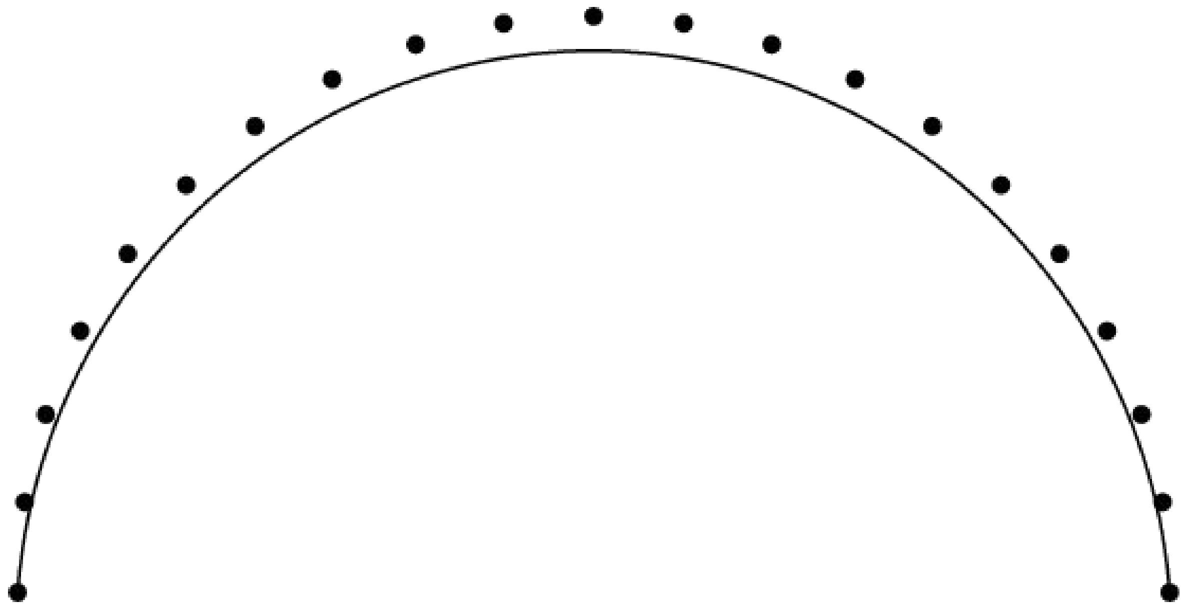
Degree 4

Bézier curves



Degree 10

Bézier curves

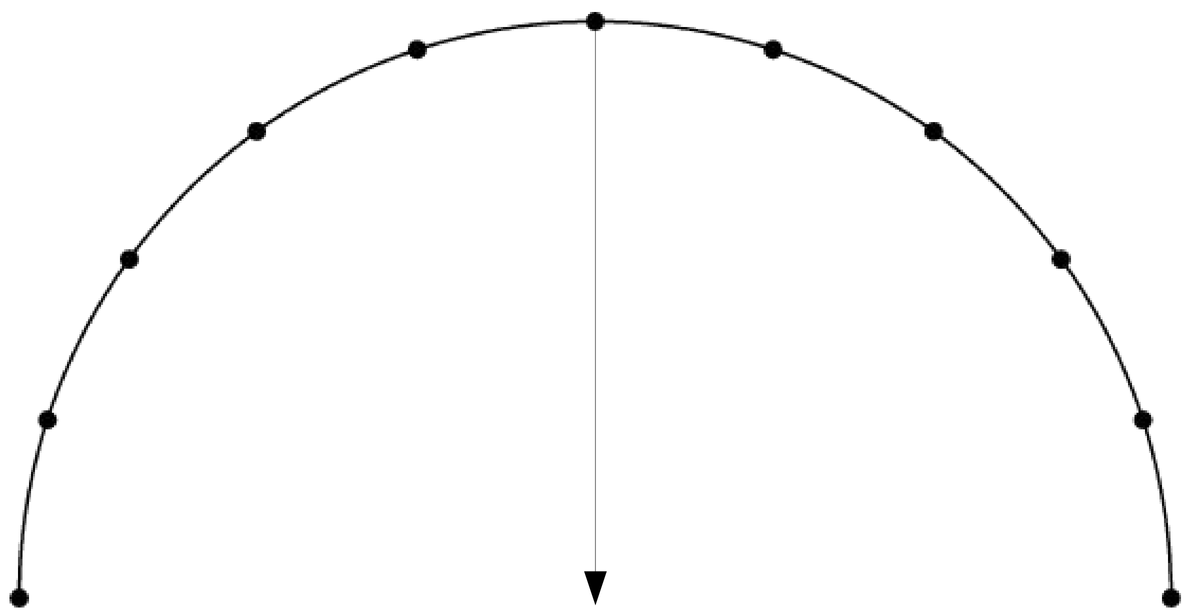


Degree 20

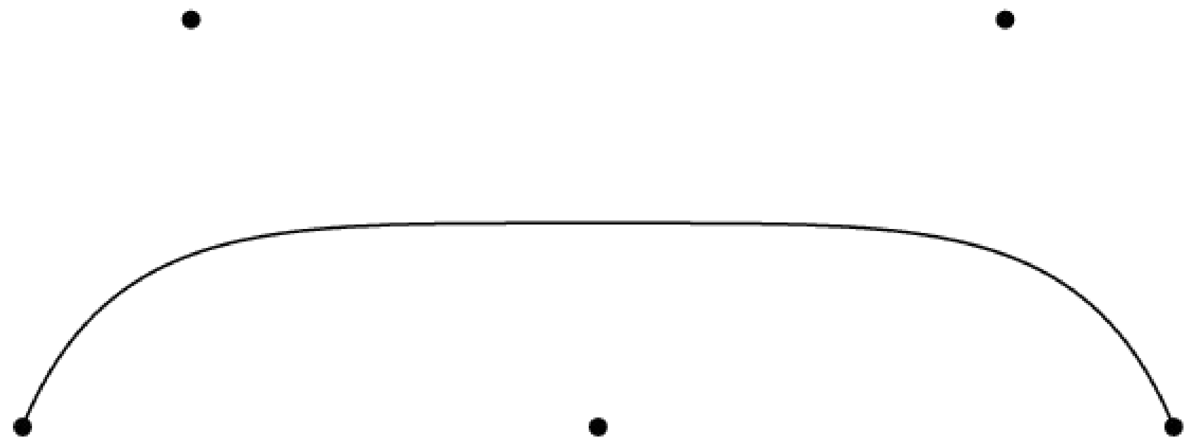
Bézier curves

- When the number of control points increases, the curve tends to the control polygon (under the assumption that the control polygon itself converges to a smooth curve ...)
- The approximation involves a substantial error between the curve and the control points
  - However, an interpolation is not the objective here...

- Perturbation of a point
  - We shift the indicated point



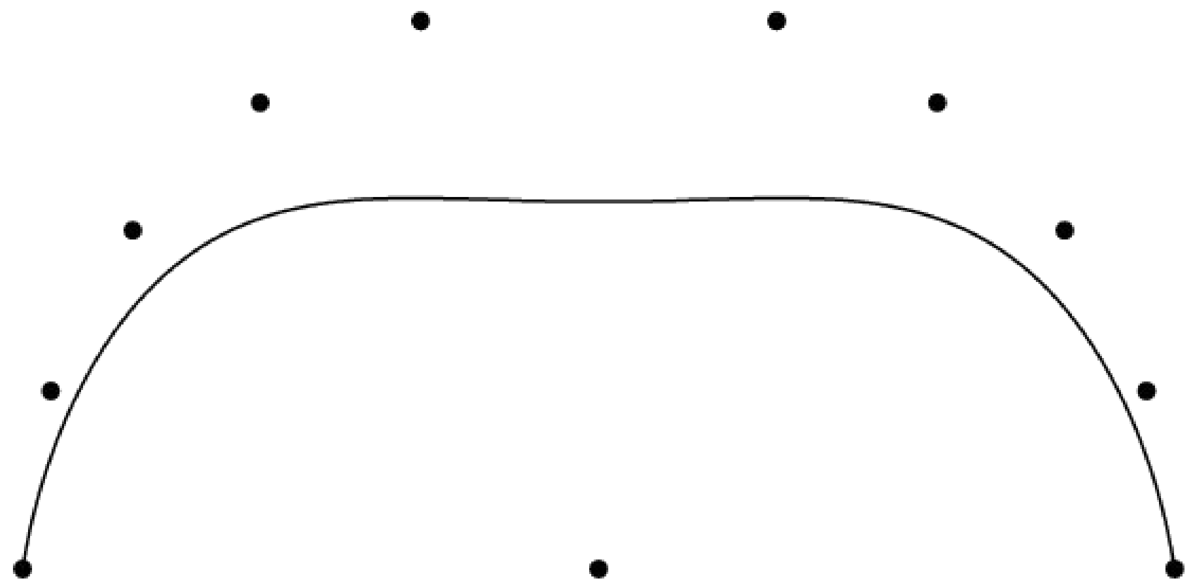
Bézier curves



Degree 4

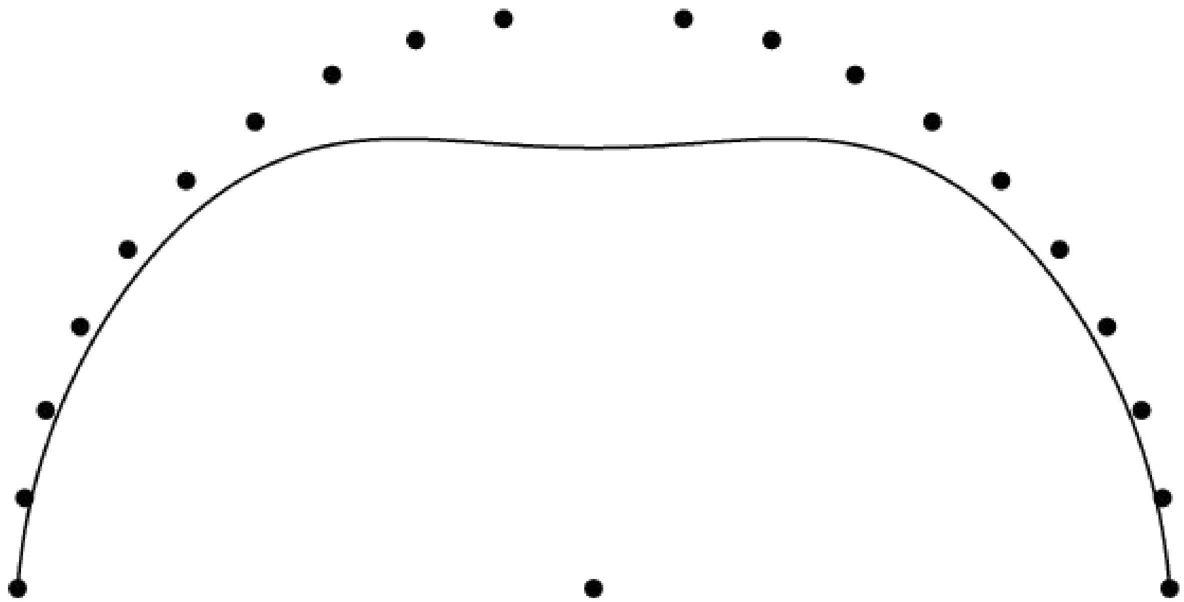
CAD & Computational Geometry

Bézier curves



Degree 10

Bézier curves



Degree 20

Bézier curves

- Editing Bézier curves
  - Degree elevation
  - Computation of points on the curve (De Casteljau's algorithm and others)
  - Changing the range of a curve
    - Cutting, extension
  - Curves defined by pieces and recursive subdivision

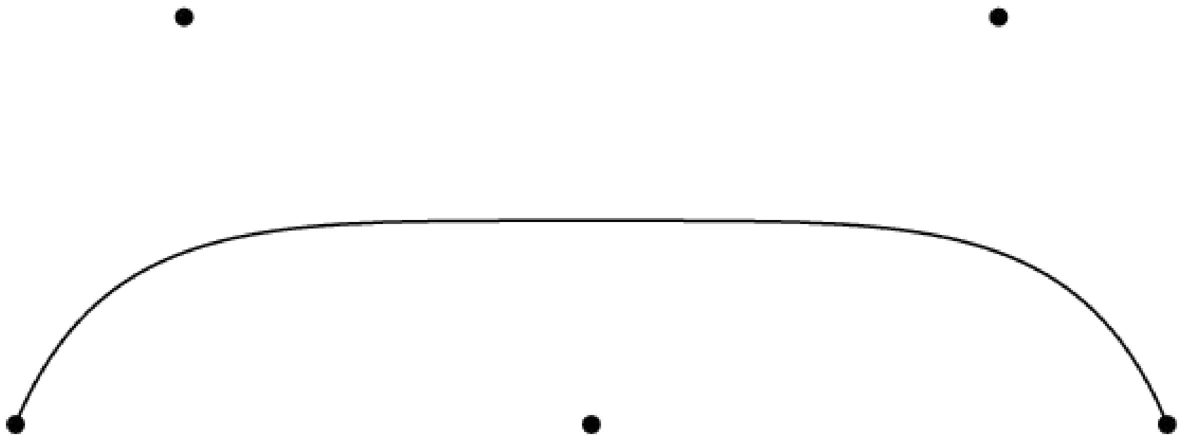
- Degree elevation
  - A curve of degree $d+1$ is able to represent any curve of degree d
  - If there aren't enough control points to design a given shape, the degree may be increased...
    - New control points must be determined (one more !)
    - Forrest's equations [1972]

$$Q_0 = P_0$$

$$Q_i = \frac{i}{d+1} P_{i-1} + \left(1 - \frac{i}{d+1}\right) P_i \text{ for } i = 1, \cdots, d$$

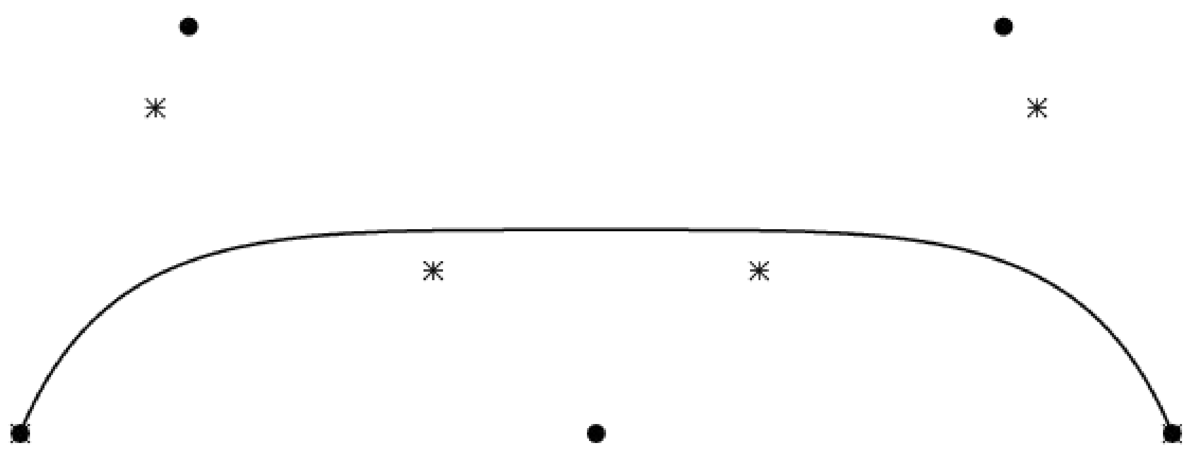
$$Q_{d+1} = P_d$$

- Degree elevation in practice ...



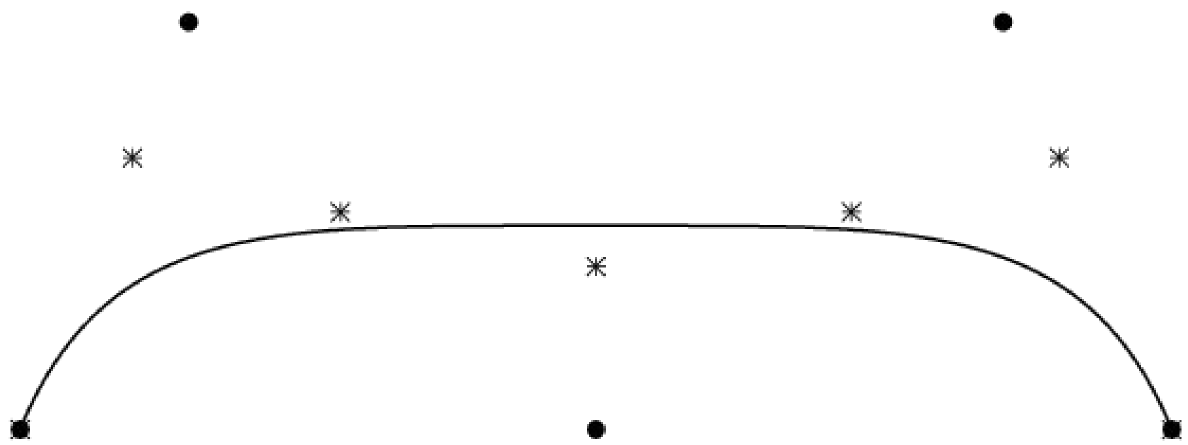
Degree 4

Bézier curves



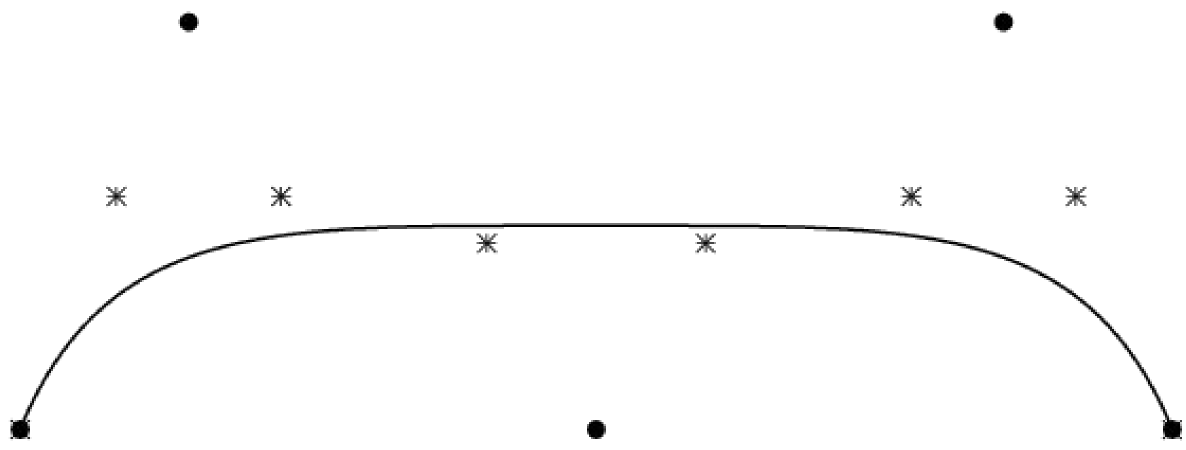
Degree 5

Bézier curves



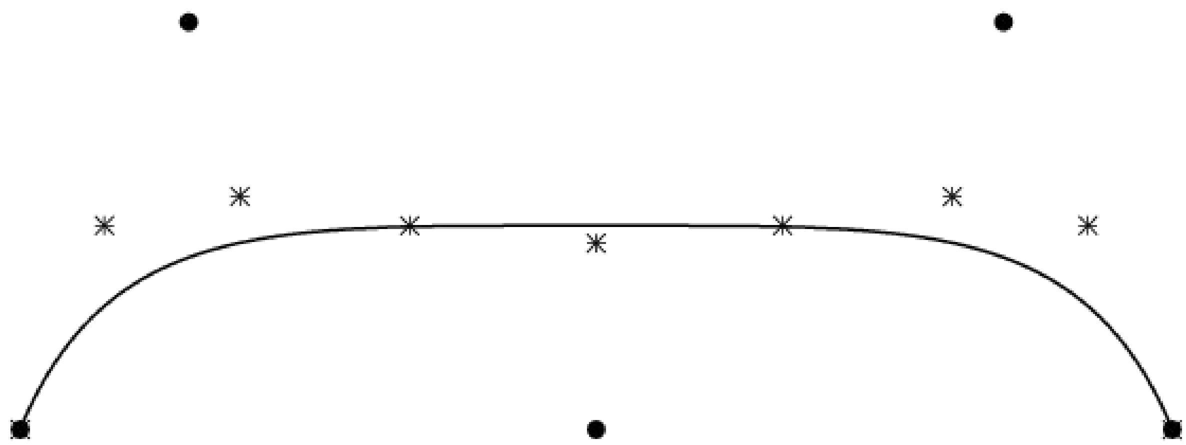
Degree 6

Bézier curves



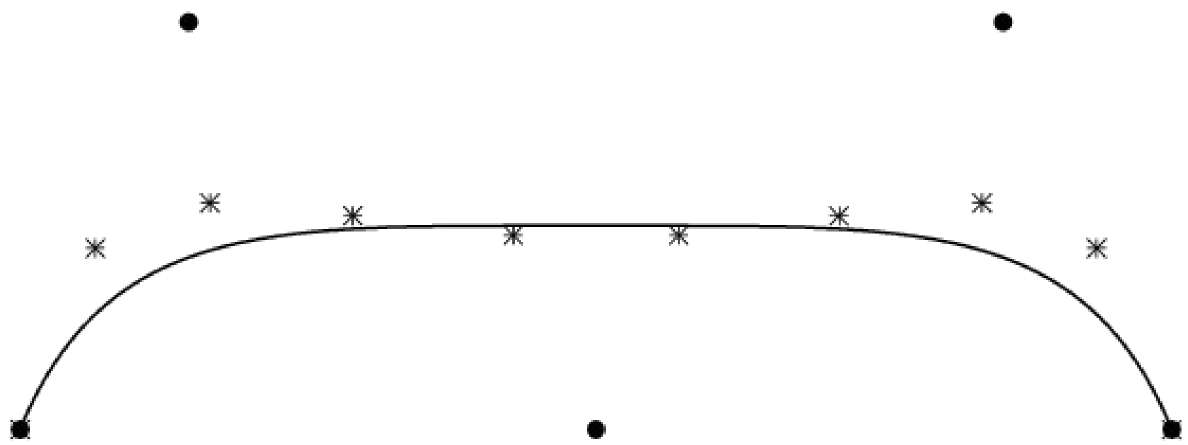
Degree 7

Bézier curves



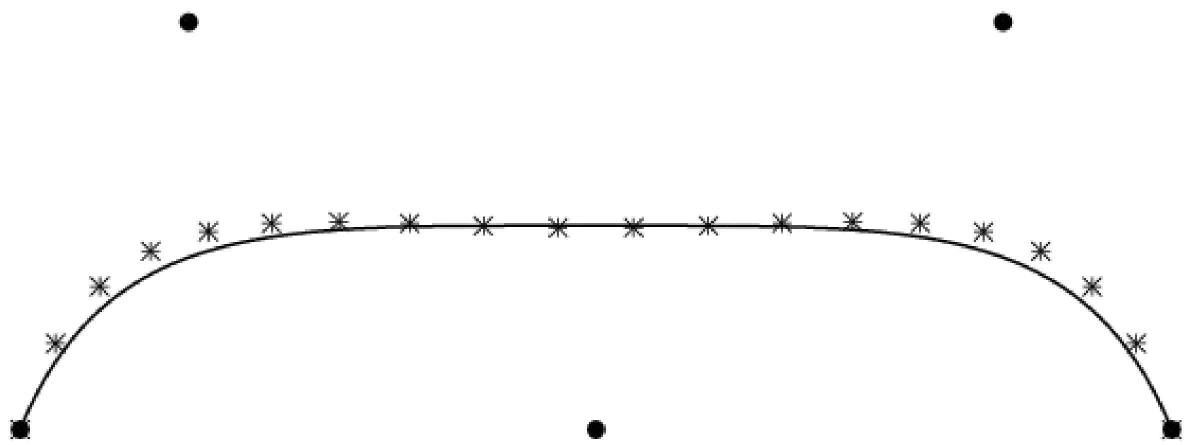
Degree 8

Bézier curves



Degree 9

Bézier curves

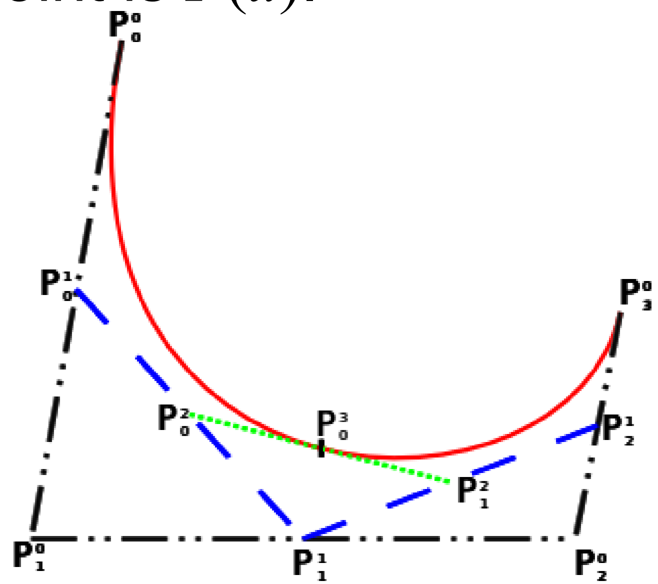


Degree 21

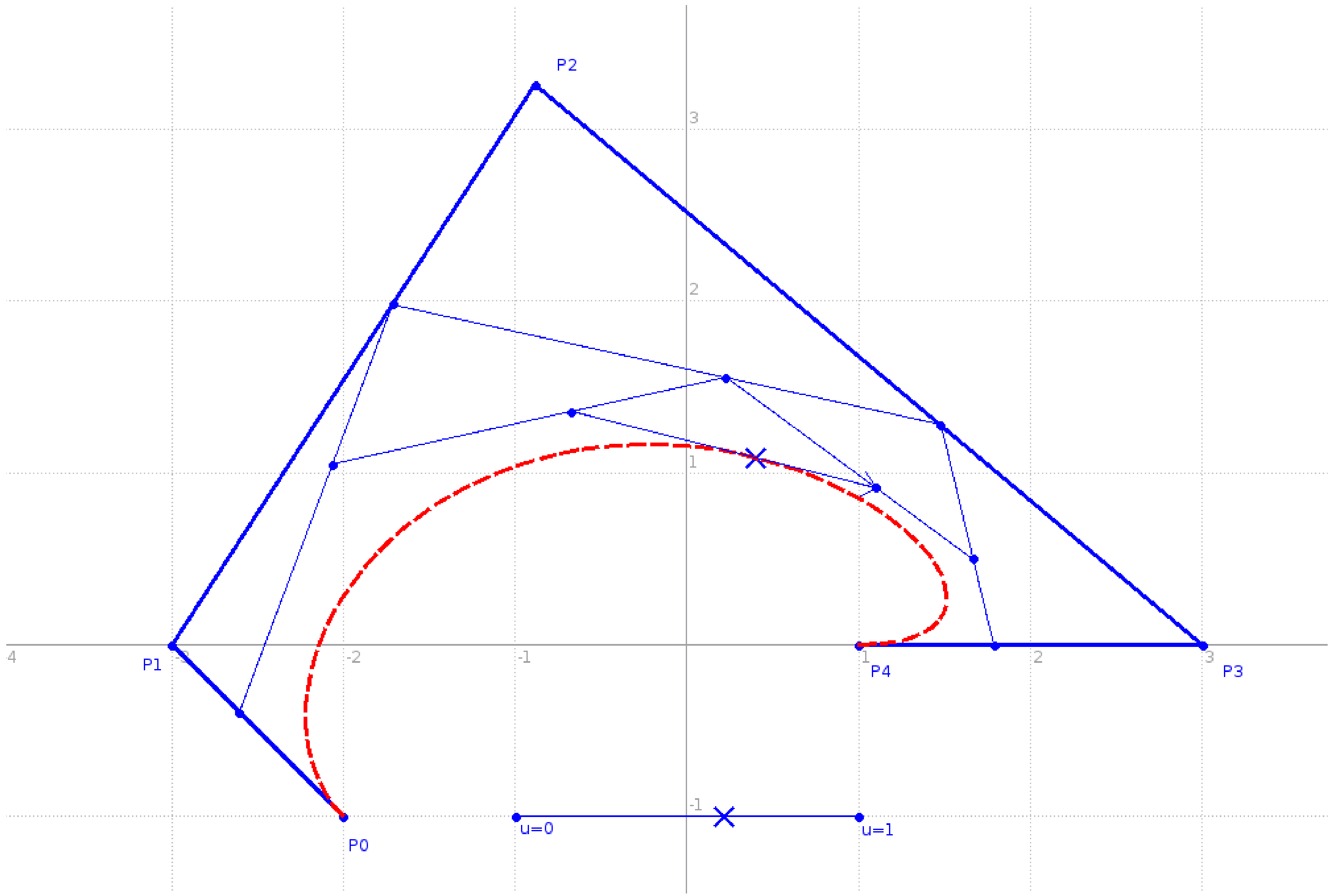
- De Casteljau's algorithm
  - Allows the robust construction of points on the curve
  - Very simple geometrical interpretation

Bézier curves

- Principle of De Casteljau's algorithm
  - Construction of the centroids P_i^1 of the control points P_i^0 : $P_i^1 = (1-u) P_i^0 + u P_{i+1}^0$
  - We continue with P_i^2 ....
    - As far as possible, until only one control point remains, P_0^d That control point is $P(u)$.



- Kig



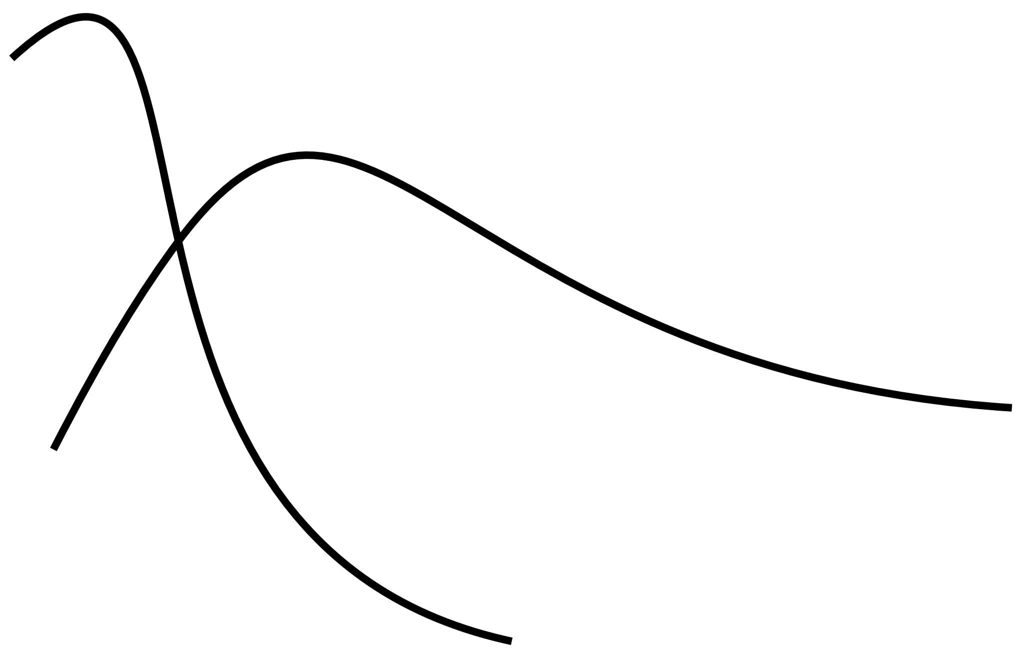
Bézier curves

- The algorithm is :

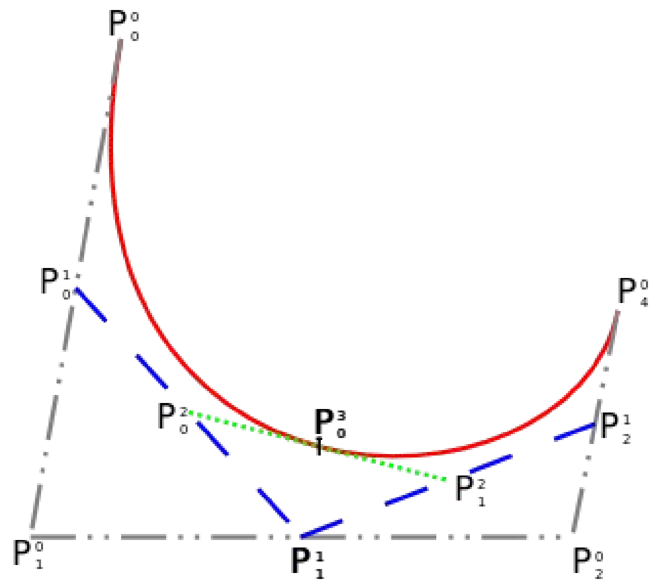
Initialization of P_i^0
For j from 1 to d
  For i from 0 to d-j
    $P_i^j = (1-u) P_i^{j-1} + u P_{i+1}^{j-1}$
  EndFor
EndFor
P_0^d is the point we want.

- What is its complexity ?
  - Consists of $3d(d+1)$ multiplications and $3d(d+1)/2$ additions , so quadratic with respect of the the degree d.

- Restriction of a curve (cutting)
  - Let us compute the intersection of two curves
    - We need a independent representation of each segment
    - One wants $0<u<1$ on each segment

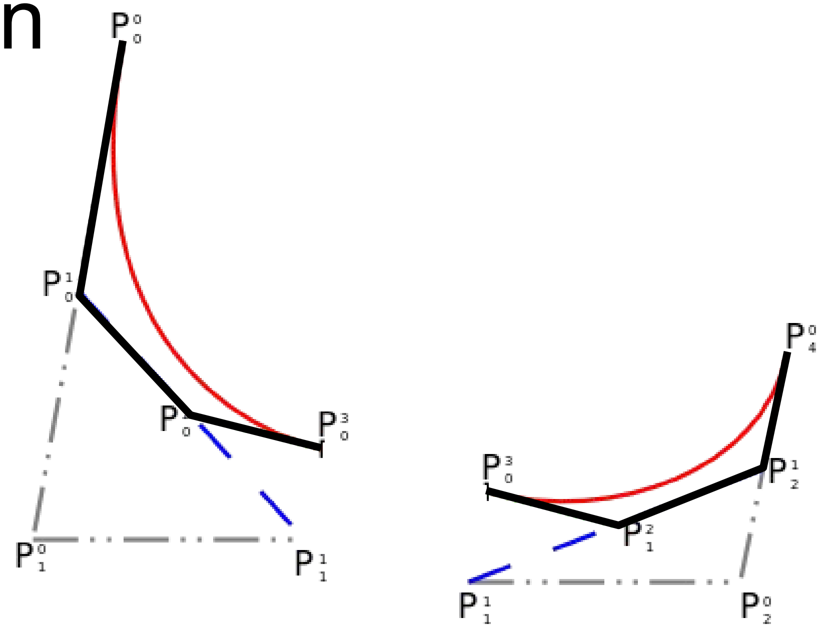


- Let us start from De Casteljau's geometrical construction



Bézier curves

- Let us start from De Casteljau's geometrical construction



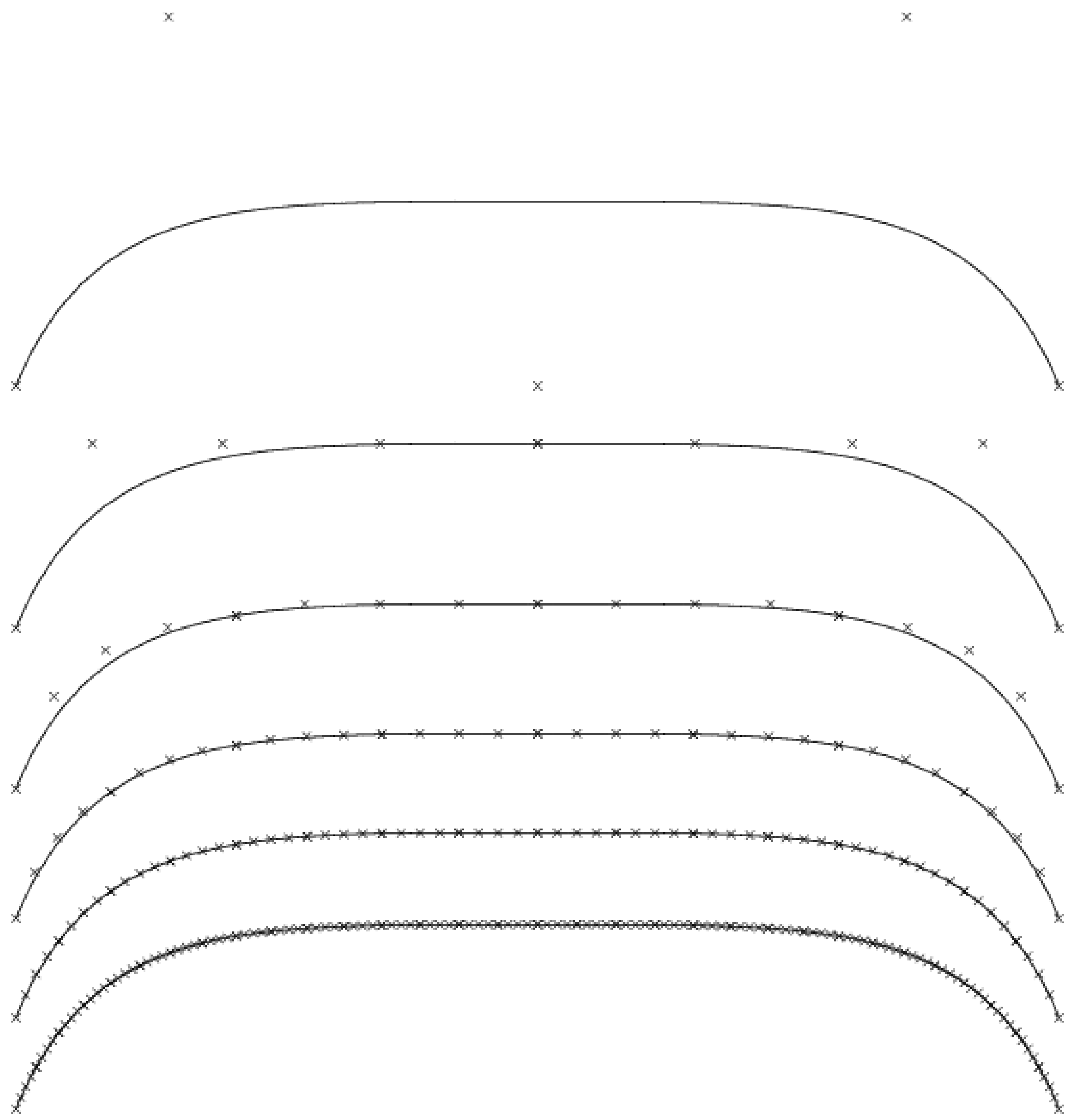
  - The control polygon of the both parts is obtained from points coming from De Casteljau's algorithm !

Bézier curves

- Recursive subdivision
  - Allows to draw the curve quickly with the help of De Casteljau's algorithm
    - Idea : splitting up the curve in two parts at $u=0.5$, then these sub-curves in four parts (still for $u^{*}=0.5$) and so on.
    - The control points of the sub-curves are obtained like a residual of the De Casteljau algorithm at each step
    - The control points quickly converge toward the curve
    - When the gap between the starting and ending points of each sub-curves is lower than a factor (depends on the resolution), we join simply the points of the characteristic polygon by straight line segments.
    - It's a « divide and conquer » approach – a famous paradigm in software engineering.

CAD & Computational Geometry

Bézier curves



0 subdivision

2 subdivisions

4 subdivisions

8 subdivisions

16 subdivisions

32 subdivisions

- Cost of the recursive subdivision algorithm
  - In $O\left(d^2 \cdot 2^m\right)$ for m levels of subdivision
  - Number of generated points: $d \cdot 2^m$
  - For each point that is generated, the algorithm becomes linear…
    - It is not very accurate, nevertheless very robust.

Partition of unity and affine invariance

- Property of affine invariance
  - It is a useful property such that the curves we define for a set of control points can undergo linear affine transformations without hassle.
    - Let P_i^* the affine transformation of the control points P_i
    - Let $P^*(u,P_i)$ the affine transformation of the points of the curve $P(u,P_i)$ defined from the original points P_i
    - Let $P(u,P_i^*)$ the new curve based on the modified control points P_i^* , with the same parametrization.
  - The affine invariance is verified iff $P^*(u,P_i) = P(u,P_i^*)$ for all u.

Partition of unity and affine invariance

- Affine transformations $\phi(\boldsymbol{P}) \equiv \mathbf{A}\cdot\boldsymbol{P}+\boldsymbol{u}$

  Translation

  $$\boldsymbol{u}=\begin{bmatrix} a \\ b \\ c \end{bmatrix} \;;\; \mathbf{A}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Scaling

  $$\boldsymbol{u}=0 \;;\; \mathbf{A}=\begin{bmatrix} d & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & f \end{bmatrix}$$

  3 rotations

  $$\boldsymbol{u}=0 \;;\; \mathbf{A}=\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdots$$

  Shear

  $$\boldsymbol{u}=0 \;;\; \mathbf{A}=\begin{bmatrix} 1 & g & h \\ 0 & 1 & i \\ 0 & 0 & 1 \end{bmatrix}$$

- 12 degrees of freedom

Partition of unity and affine invariance

Let P a parametric curve built this way :

$$P(u, P_i) = \sum_0^{n-1} P_i K_i^n(u)$$

- Let's verify the invariance by a translation t:

$$P(u, P_i^*) = \sum_0^{n-1} (P_i + t) K_i^n(u) = \sum_0^{n-1} P_i K_i^n(u) + \sum_0^{n-1} t K_i^n(u)$$

$$= P(u, P_i) + \sum_0^{n-1} t K_i^n(u)$$

Partition of unity

$$= P(u, P_i) + t = P^*(u, P_i) \quad \text{iff} \quad \sum_0^{n-1} K_i^n(u) = 1$$

Partition of unity and affine invariance

- For the other multiplicative transformations

$$P\left(u, P_i^*\right) = \sum_0^{n-1} \left(\mathbf{A} \cdot P_i\right) K_i^n(u) = \mathbf{A} \cdot \sum_0^{n-1} P_i K_i^n(u)$$

$$= \mathbf{A} \cdot P\left(u, P_i\right) = P^*\left(u, P_i\right)$$

(no particular conditions except linearity
with respect to the coordinates of the control points)

Consequently, iff the basis functions form a *partition of unity*, and the dependence with respect to the control points is *linear*, then the representation is invariant by any affine transformation.

Partition of unity and affine invariance

- Case of splines : we had on each interval :

$$\begin{cases} a_{[i]0} = x_i \\ a_{[i]1} = x_i' \\ a_{[i]2} = 3(x_{i+1} - x_i) - 2x_i' - x_{i+1}' \\ a_{[i]3} = 2(x_i - x_{i+1}) + x_i' + x_{i+1}' \end{cases}$$

$$x_{[i]}(\bar{u}) = x_i + x_i'\bar{u} + (3(x_{i+1} - x_i) - 2x_i' - x_{i+1}')\bar{u}^2 + (2(x_i - x_{i+1}) + x_i' + x_{i+1}')\bar{u}^3$$

- By rearranging equations

$$\begin{aligned} x_{[i]}(\bar{u}) = {} & x_i(1 - 3\bar{u}^2 + 2\bar{u}^3) + x_i'(\bar{u} - 2\bar{u}^2 + \bar{u}^3) \\ & + x_{i+1}(3\bar{u}^2 - 2\bar{u}^3) + x_{i+1}'(-\bar{u}^2 + \bar{u}^3) \end{aligned}$$

Partition of unity and affine invariance

- In fact, we use Hermite polynomials (for two points), on each interval

$$\begin{cases} h_{00}^{p} = 1 - 3\,\bar{u}^{2} + 2\,\bar{u}^{3} \\ h_{10}^{p} = 3\,\bar{u}^{2} - 2\,\bar{u}^{3} \\ h_{01}^{p} = \bar{u} - 2\,\bar{u}^{2} + \bar{u}^{3} \\ h_{11}^{p} = -\bar{u}^{2} + \bar{u}^{3} \end{cases}$$

Partition of unity and affine invariance

- Properties of the Hermite basis

Interpolation of the positions

$$h^n_{i0}(u_j) = \delta_{ij}$$

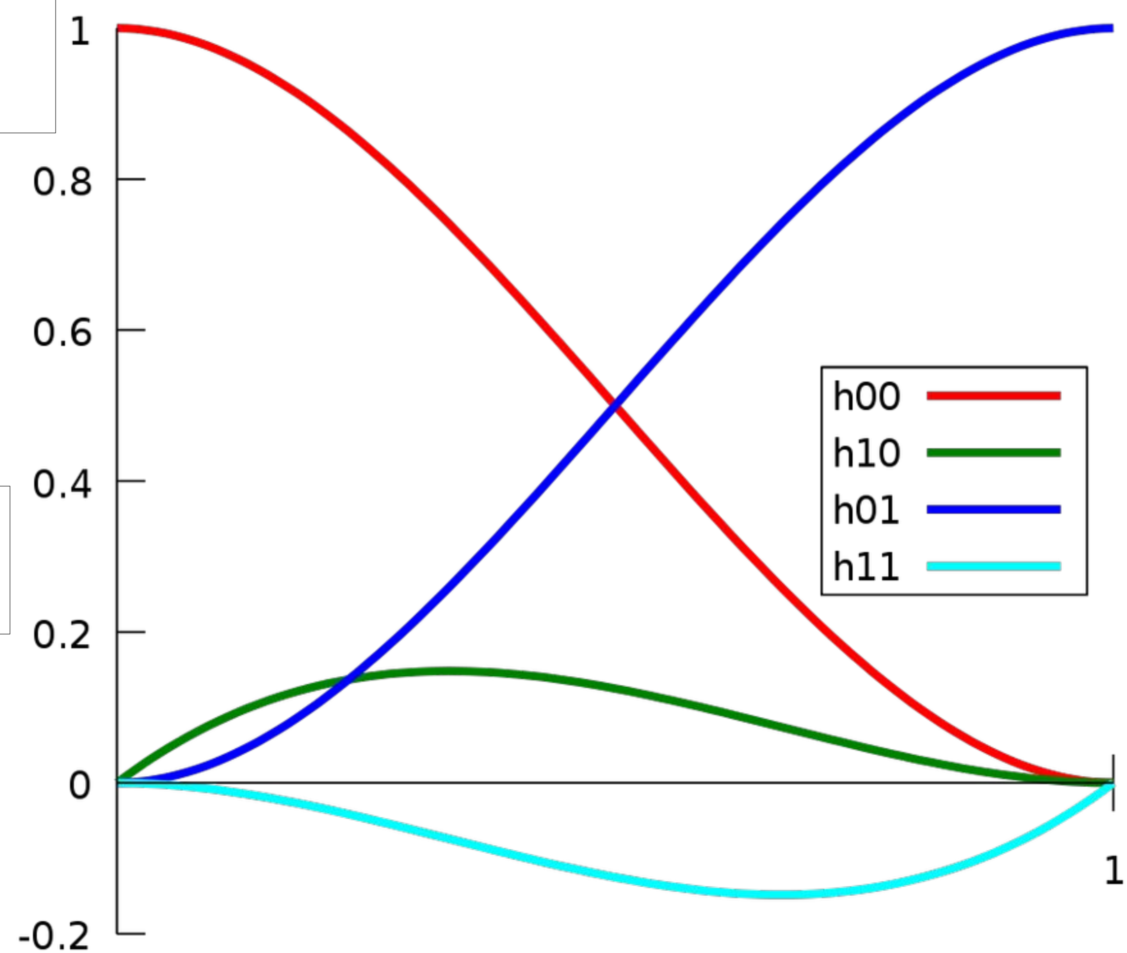
$$(h^n_{i0})'(u_j) = 0$$

$$h^n_{i1}(u_j) = 0$$

$$(h^n_{i1})'(u_j) = \delta_{ij}$$

Interpolation of the slopes

$$\sum_i h^n_{i0}(u) = 1$$



Partition of unity and affine invariance

- Do Hermite's basis form a partition of unity ?

$$\sum_{0}^{n-1} h^{n}_{??}(u) = 1$$

$$\begin{cases} h^{p}_{00} = 1 - 3\bar{u}^{2} + 2\bar{u}^{3} \\ h^{p}_{10} = 3\bar{u}^{2} - 2\bar{u}^{3} \\ h^{p}_{01} = \bar{u} - 2\bar{u}^{2} + \bar{u}^{3} \\ h^{p}_{11} = -\bar{u}^{2} + \bar{u}^{3} \end{cases}$$

$$F(u) = 1$$

$$F(0) = 1 \qquad F(1) = 1 \qquad F'(0) = 0 \qquad F'(1) = 0$$

Partition of unity and affine invariance

- Let's check the invariance
  - If we apply a translation to the control points P_i, the derivatives P'_i should not change ...

$$P_i^* = P_i + t \qquad P_i'^{\,*} = P_i'$$

$$P\left(P_i^*\right) = \sum_0^1 \left(P_i + t\right) h_{i0}^n(u) + \sum_0^1 P_i' h_{i1}^n(u)$$

$$= \sum_0^1 t h_{i0}^n(u) + \sum_0^1 P_i h_{i0}^n(u) + \sum_0^1 P_i' h_{i1}^n(u)$$

$$= t + P\left(P_i\right) = P^*\left(P_i\right)$$

Partition of unity and affine invariance

- We must also check the invariance for the other multiplicative transformations : those affect both the coordinates and the derivatives

$$P_i^* = A \cdot P_i \qquad P_i^{'*} = A \cdot P_i^{'}$$

$$P(P_i^*) = \sum_0^1 (A \cdot P_i) h_{i0}^n(u) + \sum_0^1 (A \cdot P_i^{'}) h_{i1}^n(u)$$

$$= A \cdot \left(\sum_0^1 P_i h_{i0}^n(u) + \sum_0^1 P_i^{'} h_{i1}^n(u) \right)$$

$$= A \cdot P(P_i) = P^*(P_i)$$

QED

Partition of unity and affine invariance

- Beware of the computation of the slopes x'_i ...
  - Natural Splines :

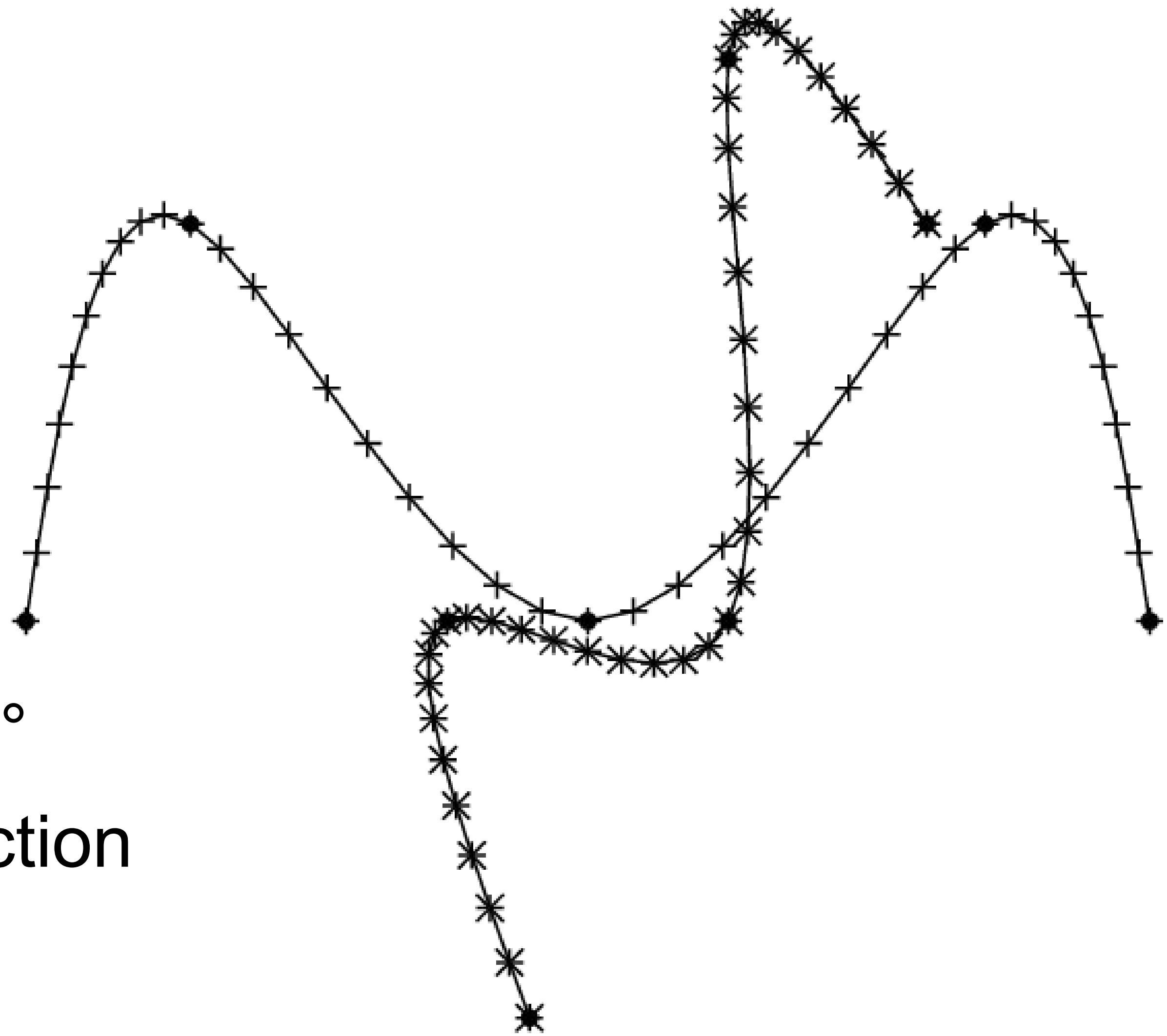
$$\begin{pmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} x'_0 \\ x'_1 \\ x'_2 \\ \vdots \\ x'_{n-2} \\ x'_{n-1} \end{pmatrix} = \begin{pmatrix} 3(x_1 - x_0) \\ 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_{n-1} - x_{n-2}) \end{pmatrix}$$

Linear operator

$$P'_i = L(P_i - P_j) \Rightarrow P'^{*}_i = L(P^*_i - P^*_j) = L((A \cdot P_i + t) - (A \cdot P_j + t))$$

$$= A \cdot L(P_i - P_j) = A \cdot P'_i \quad \text{It is OK in this case}$$

Partition of unity and affine invariance



- Rotation of 45°
- Scaling x direction (times 0.5)
- Followed by a translation