Computer Graphics

Course outline

- Introduction
- Images and display techniques
  - Bases
  - Gamma correction
  - Aliasing and techniques to remedy
  - Storage

Course outline

- 3D Perspective & 2D / 3D transformations
  - Go from a 3D space to a 2D display device
- Two paradigms for image synthesis
- Representation of curves and surfaces
  - Splines & co.
  - Meshes
- Realistic rendering by ray tracing
  - Concepts and theoretical bases

Representation of curves and surfaces

- Curves
  - Interpolation
    - Cubic Splines
  - Approximation
    - Bézier Curves
    - B-spline Curves
- Surfaces
  - Interpolation
    - Coons Patches
  - Approximation
    - B-splines Surfaces
    - Subdivision Surfaces

- Two useful references...

  JC.Léon, Modélisation et construction de surfaces pour la CFAO, Hermes, 1991

  L. Piegl, W. Tiller, The NURBS Book, Second Edition, Springer , 1996

Cubic Splines

Splines

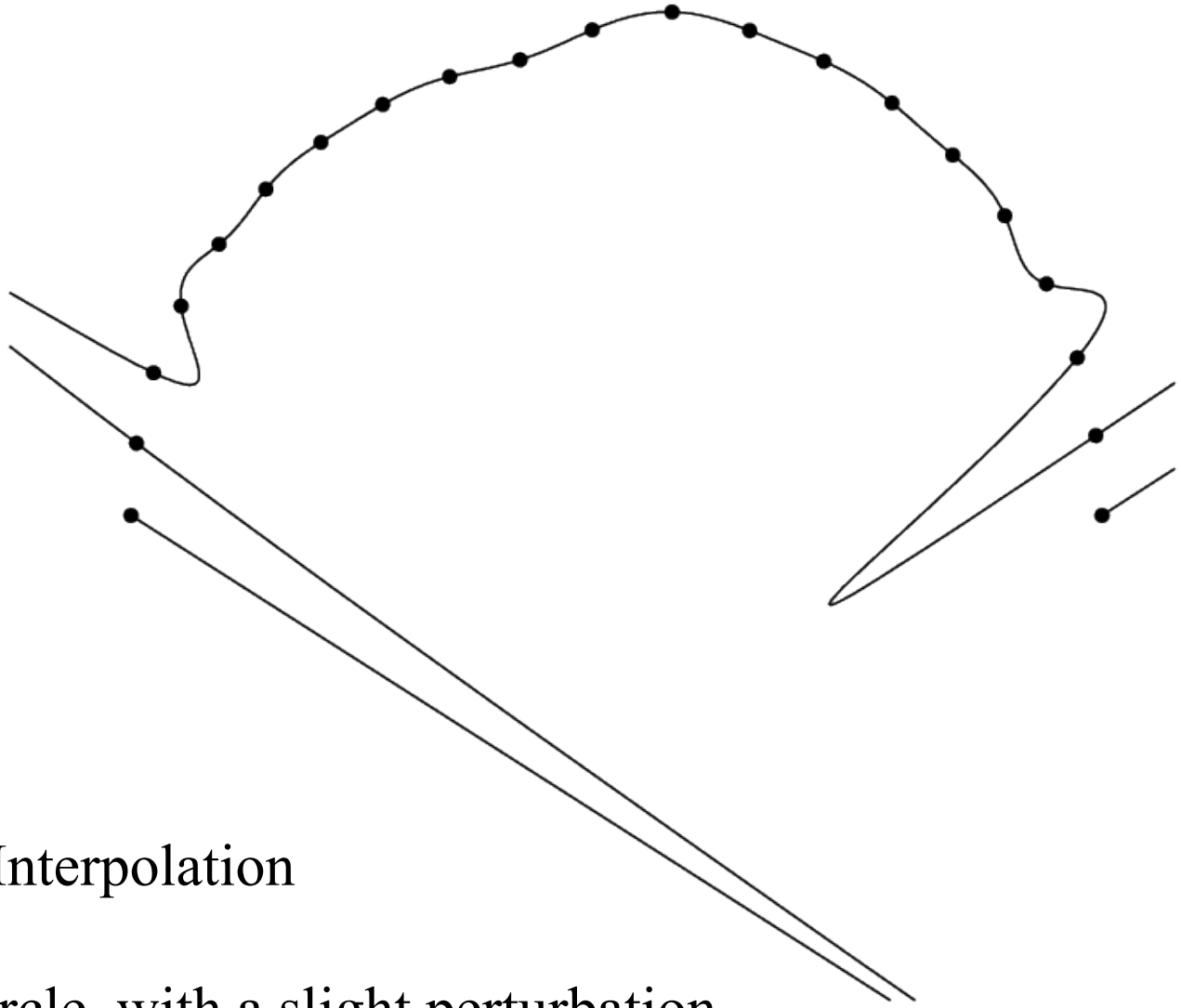
- Goal :
  - Interpolate points (control points)
    - Go through points (interpolation ≠ approximation)
  - Ensure a certain regularity to the curve
  - Simple (easy) to compute

Splines

- Let start with the idea that we have many control points
  - Lagrange interpolation ?
    - This corresponds to a unique polynomial interpolating every control points (d+1)
    - C_∞ Continuity

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

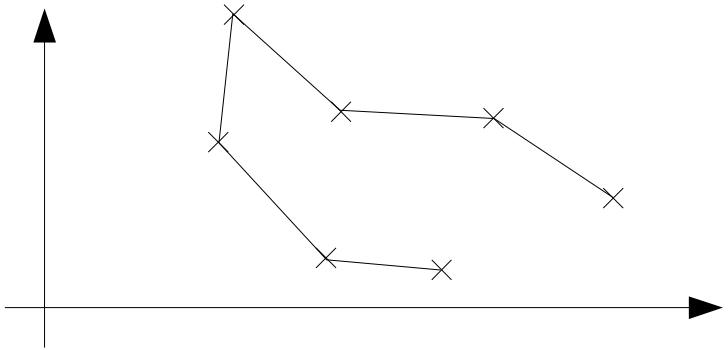
Splines



Lagrange Interpolation
21 CP
All on a circle, with a slight perturbation

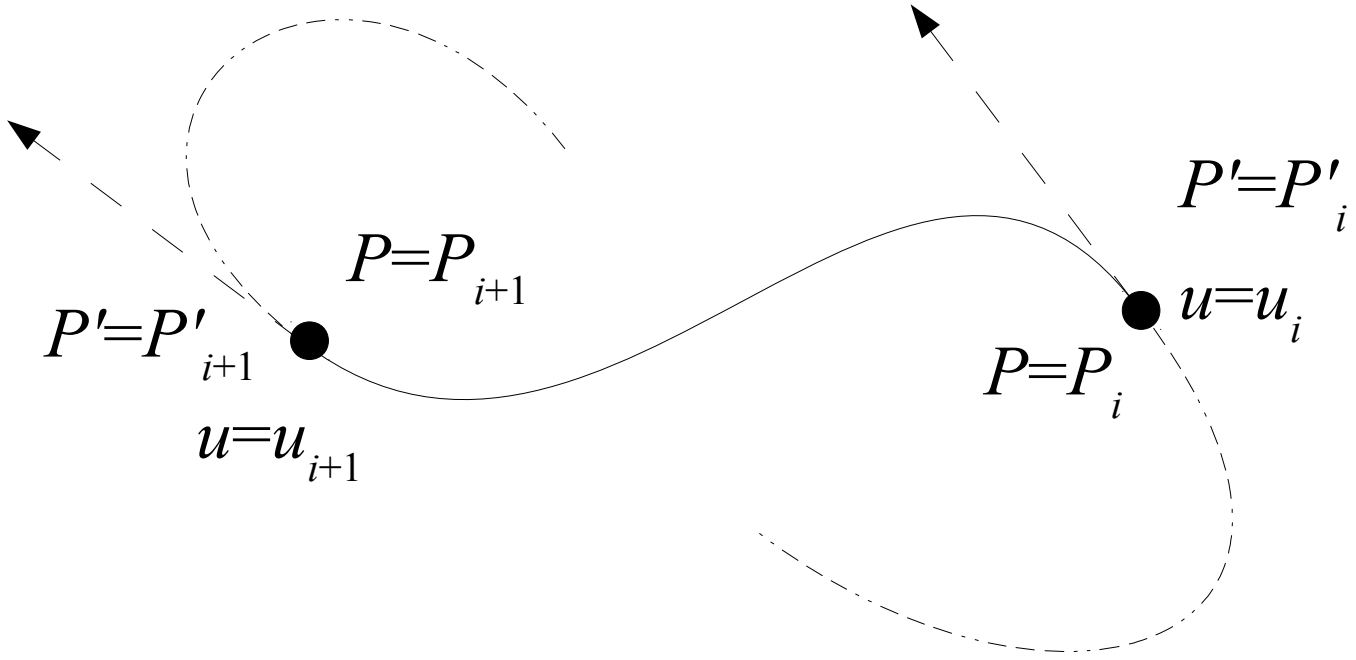
Splines

- Motivation
  Let us imagine that we have many (100's of) control points
  - But we don't want a Lagrange interpolation !
  - We should stay with a low order scheme but conserve enough freedom to pass through every point
    - Curve defined by pieces ... and of low order (1)



Splines

- We are going to build a low order interpolation for each knot interval, such that we can impose slopes at the knots.



Splines

- In each range $[i,i+1]$, we want to have an independent polynomial
- We have 4 parameters : position at each knot and associated tangents.
  - The basis must have 4 degrees of freedom, thus be of order 3 in the case of polynomials.

$$P(u_i)=P_i \equiv \begin{cases} x(u_i)=x_i \\ y(u_i)=y_i \end{cases}$$

$$x_{[i]}(u)=A_{[i]0}+A_{[i]1}u+A_{[i]2}u^2+A_{[i]3}u^3 \quad , u\in[u_i,u_{i+1}]$$

Splines

- First, every interval has a unit length i.e.

$$u_{i+1} - u_i = 1$$

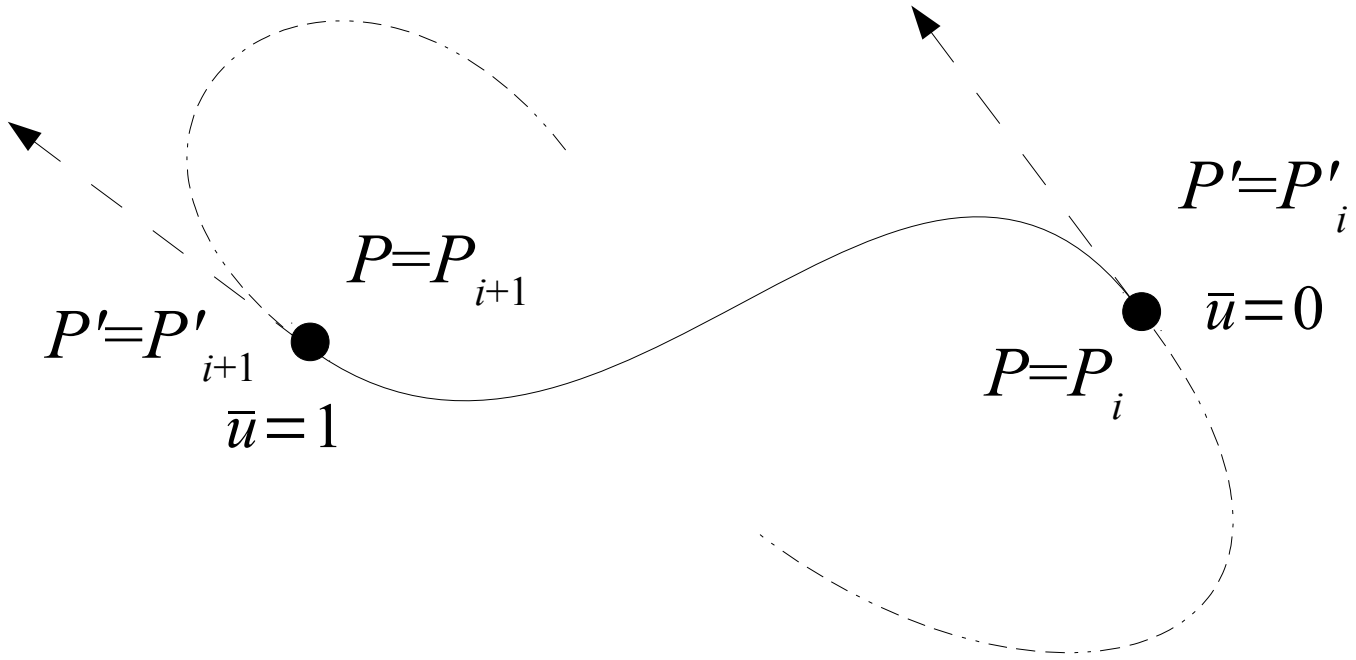
- Then we ensure identical intervals $[0...1]$ between each interpolation point :

$$\bar{u} = \frac{u - u_i}{u_{i+1} - u_i} = u - u_i \qquad \frac{d\bar{u}}{du} = 1$$

- On each interval i , we thus have the following relation:

$$x_{[i]}(\bar{u}) = a_{[i]0} + a_{[i]1}\bar{u} + a_{[i]2}\bar{u}^2 + a_{[i]3}\bar{u}^3 \quad , \ \bar{u} \in [0,1]$$

Splines



Splines

- We pass through both control points:

$$P(\bar{u}_0=0)=P_i \Leftrightarrow a_{[i]0}+a_{[i]1}\bar{u}_0+a_{[i]2}\bar{u}_0^2+a_{[i]3}\bar{u}_0^3=x_i$$

$$P(\bar{u}_1=1)=P_{i+1} \Leftrightarrow a_{[i]0}+a_{[i]1}\bar{u}_1+a_{[i]2}\bar{u}_1^2+a_{[i]3}\bar{u}_1^3=x_{i+1}$$

- We impose both slopes :

$$P'(\bar{u}_0=0)=P'_i \Leftrightarrow a_{[i]1}+2a_{[i]2}\bar{u}_0+3a_{[i]3}\bar{u}_0^2=x'_i$$

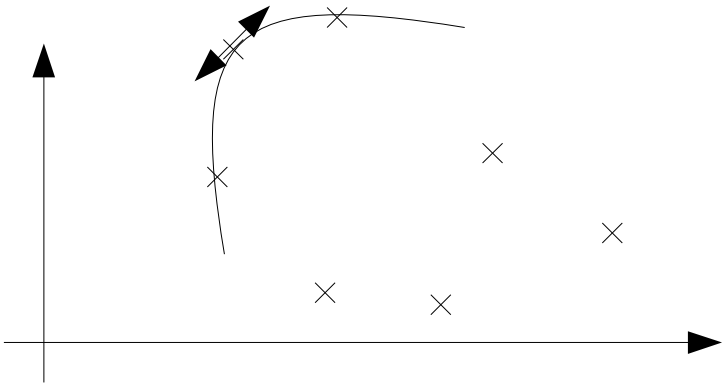
$$P'(\bar{u}_1=1)=P'_{i+1} \Leftrightarrow a_{[i]1}+2a_{[i]2}\bar{u}_1+3a_{[i]3}\bar{u}_1^2=x'_{i+1}$$

- At the end :

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

Splines

- We have continuity
- We have continuity of the derivatives
- But how to choose the slopes ?
  - Let the user choose (“artistic” freedom)
  - Automatically ...



Splines

- By finite differences with three points :

$$x_i' = \frac{x_{i+1} - x_i}{2(u_{i+1} - u_i)} + \frac{x_i - x_{i-1}}{2(u_i - u_{i-1})}$$

  - At the boundaries, we use finite differences (asymmetric)

$$x_0' = \frac{x_1 - x_0}{u_1 - u_0} \qquad x_{n-1}' = \frac{x_{n-1} - x_{n-2}}{u_{n-1} - u_{n-2}}$$

  - The result depends on the parametrization !

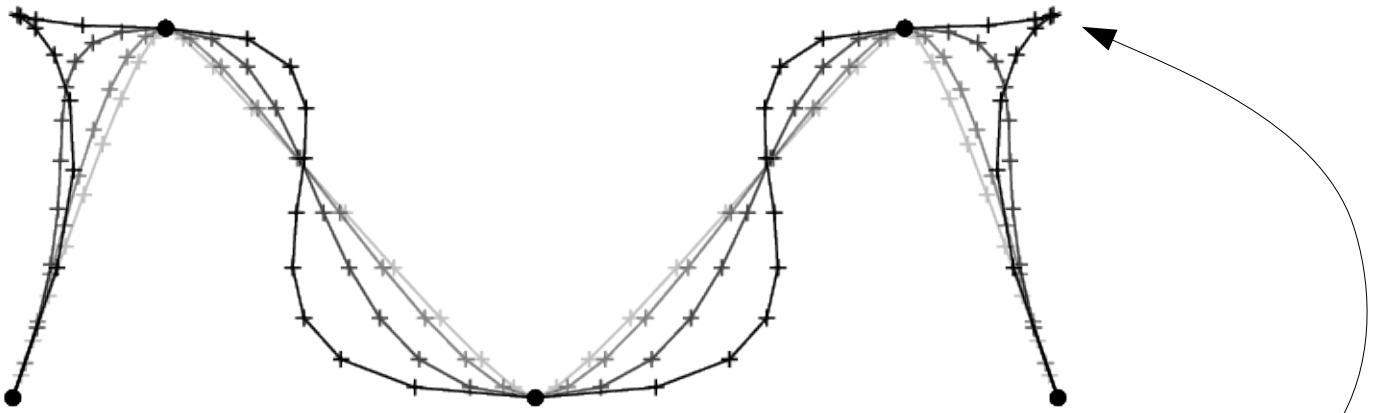
- Cardinal spline

$$x_i' = (1-c)\frac{x_{i+1} - x_{i-1}}{2}, \quad 0 \le c \le 1$$

$$x_0' = (1-c)(x_1 - x_0)$$
$$x_{n-1}' = (1-c)(x_{n-1} - x_{n-2})$$

  - c is a « tension » parameter. c=0 gives yields the so called “Catmull-Rom” spline, c=1 a zigzagging line.

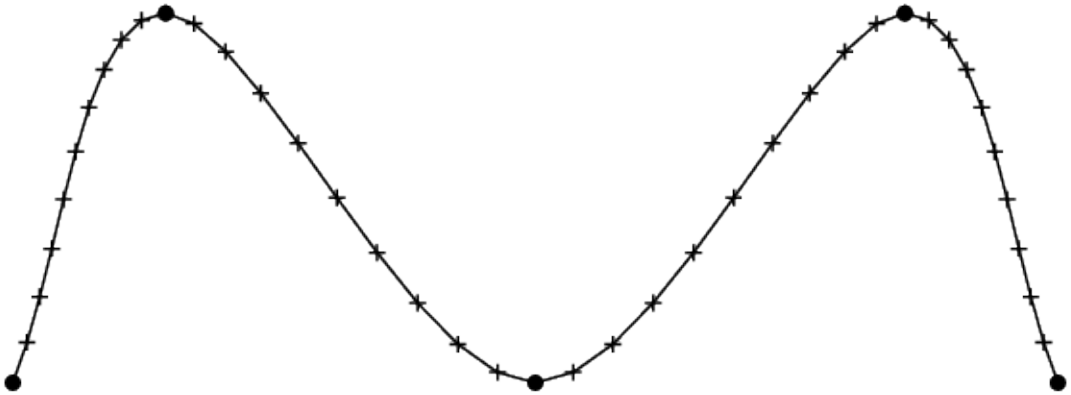
Splines



Continuity of the curve/parameter
but loss of regularity (and of geometric continuity in many cases)

5 points, finite differences by varying the parametrization
[0..1] , [0..2] , [0..5] , [0..10]

Splines

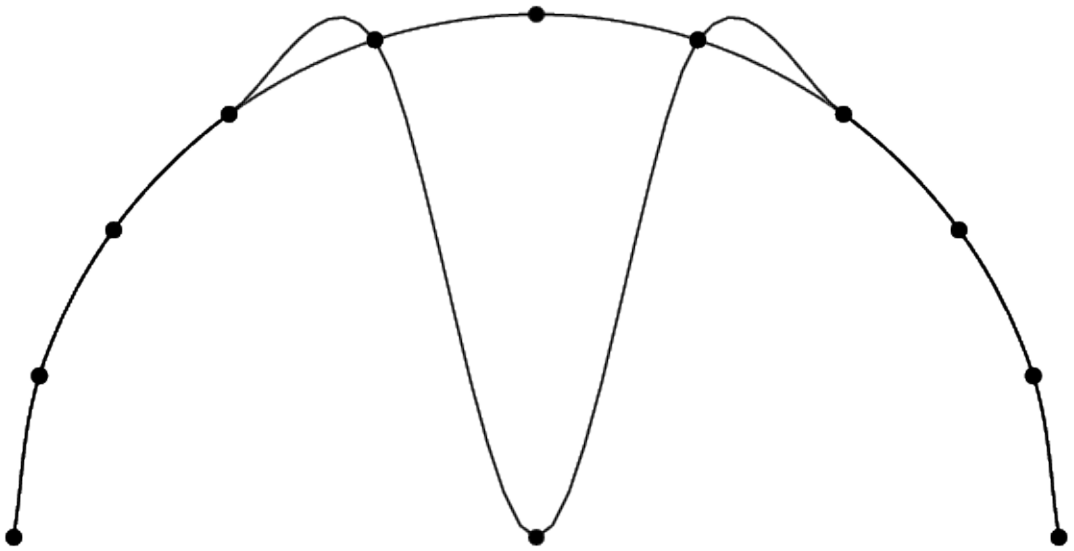


5 points, Cardinal Spline (Catmull-Rom) c=0

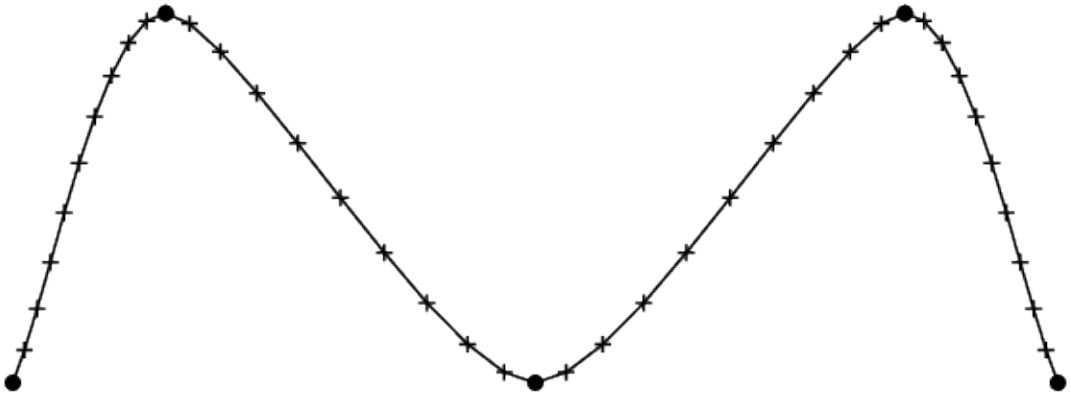
Splines

Catmull-Rom Splines are widely used in computer graphics

- Simple to compute, effective
- Local control (price to pay : discontinuous sec^{d} derivative)
- Animations with keyframing
  - Ensures a fluid motion because of the continuity of the slope

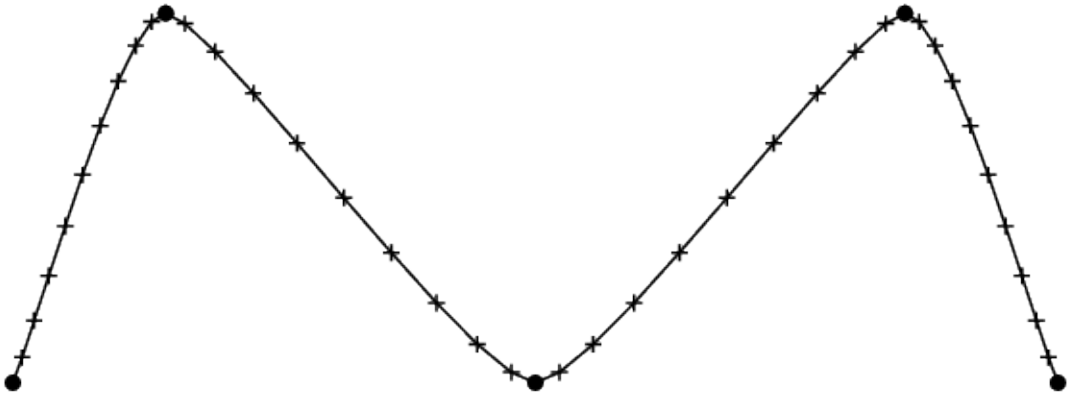


Splines



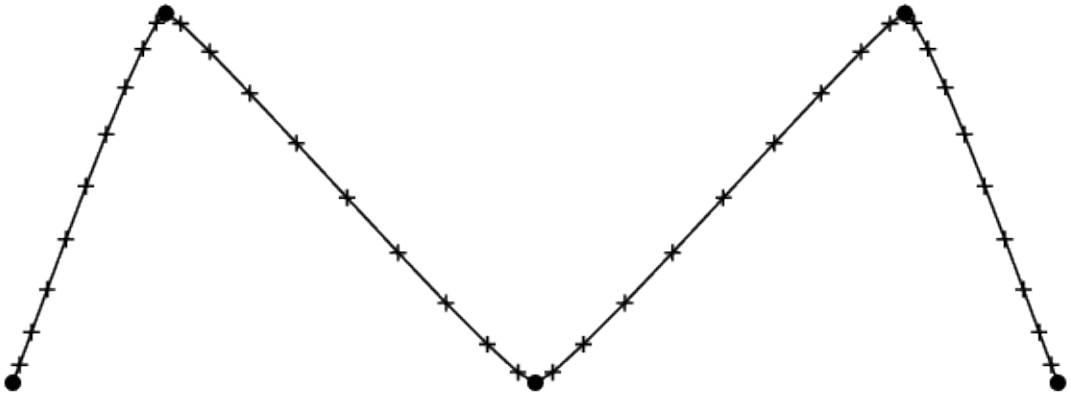
5 points, Cardinal Spline c=0.25

Splines



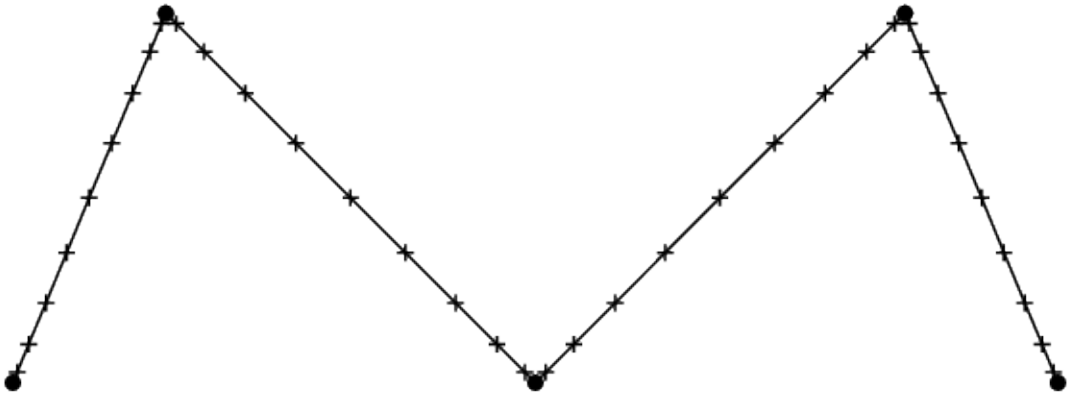
5 points, Cardinal Spline c=0.5

Splines



5 points, Cardinal Spline c=0.75

Splines



5 points, Cardinal Spline c=1.0

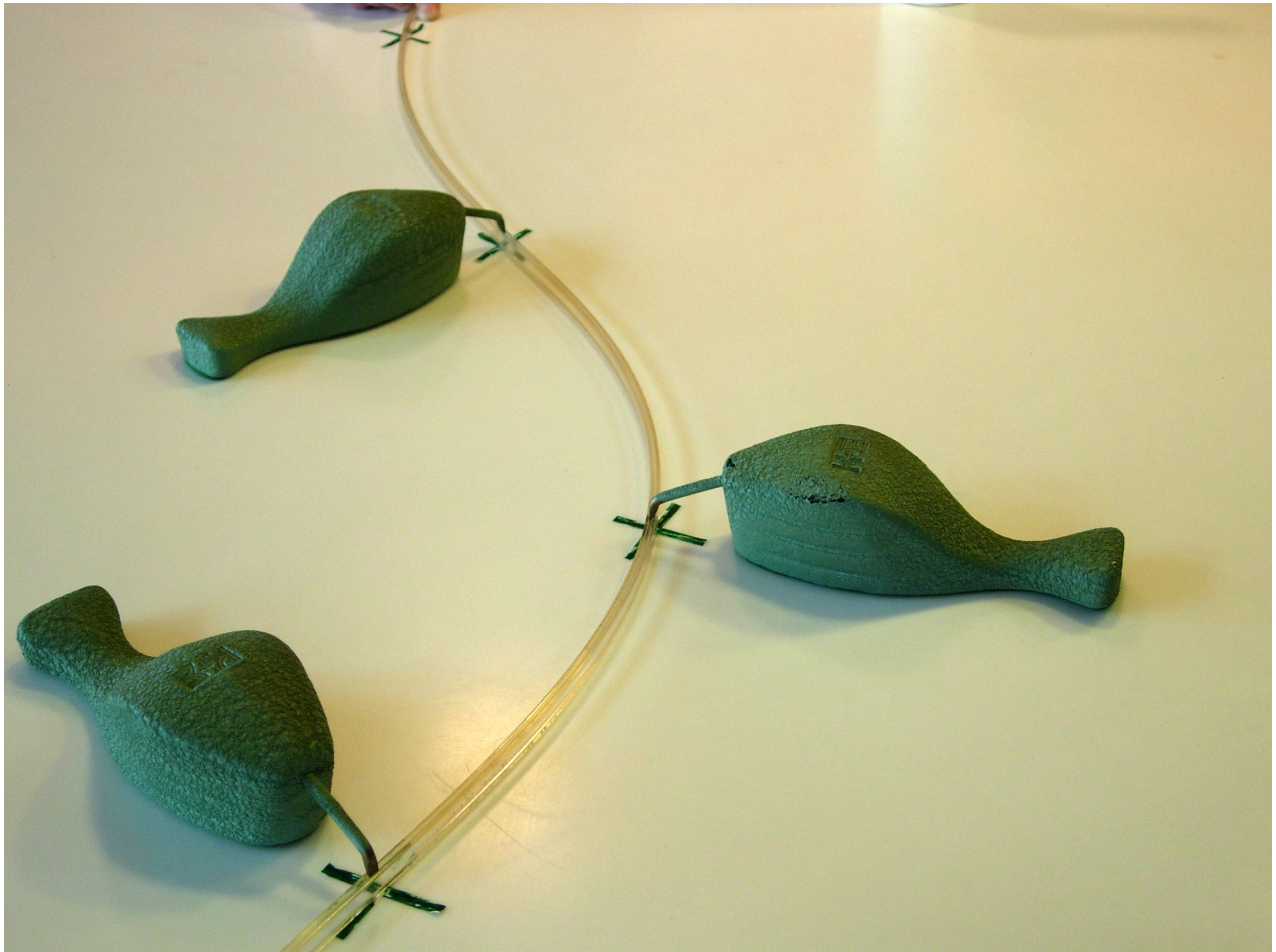
Splines

- We can impose the continuity of second derivatives...
  - On a curve with n points, we have n extra relations to impose
  - We may impose the continuity of the second derivative only on the n-2 interior knots

    What about the 2 points on the boundary ?
    - Impose a vanishing second derivative.
      We obtain what is called « **natural spline** »
    - We could also impose the slopes (i.e. only n-2 relations remaining)
    - Or , impose that the third derivative is zero on the points 1 and n-2
      - That means a single polynomial expression for the first two knot intervals, and the last two.

Splines

- Natural Spline : mathematical approximation of the spline historically used in naval construction.



C. DeBoor

Computer Graphics

Splines

Boeing

Splines

- We impose the continuity of the second derivatives

$$x''_{[i-1]}(1) = x''_{[i]}(0) \Leftrightarrow 2a_{[i-1]2} + 6a_{[i-1]3} = 2a_{[i]2}$$

- We substitute in the “internal” equations

$$2\left[3(x_i - x_{i-1}) - 2x'_{i-1} - x'_i\right] + 6\left[2(x_{i-1} - x_i) + x'_{i-1} + x'_i\right]$$
$$= 2\left[3(x_{i+1} - x_i) - 2x'_i - x'_{i+1}\right]$$

- Finally we obtain :

$$x'_{i-1} + 4x'_i + x'_{i+1} = 3(x_{i+1} - x_{i-1})$$

Splines

- At the boundaries we want

$$x''_{[0]}(0)=0 \Leftrightarrow 2a_{[0]2}=0 \qquad 2x'_0+x'_1=3(x_1-x_0)$$

$$x''_{[n-2]}(1)=0 \Leftrightarrow 2a_{[n-2]2}+6a_{[n-2]3}=0 \qquad x'_{n-2}+2x'_{n-1}=3(x_{n-1}-x_{n-2})$$

- We have then a linear system with n unknowns :

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

Splines

- By solving the system, we have :

$$\begin{pmatrix} x'_0 \\ x'_1 \\ x'_2 \\ \vdots \\ x'_{n-2} \\ x'_{n-1} \end{pmatrix}$$

, which is substituted in

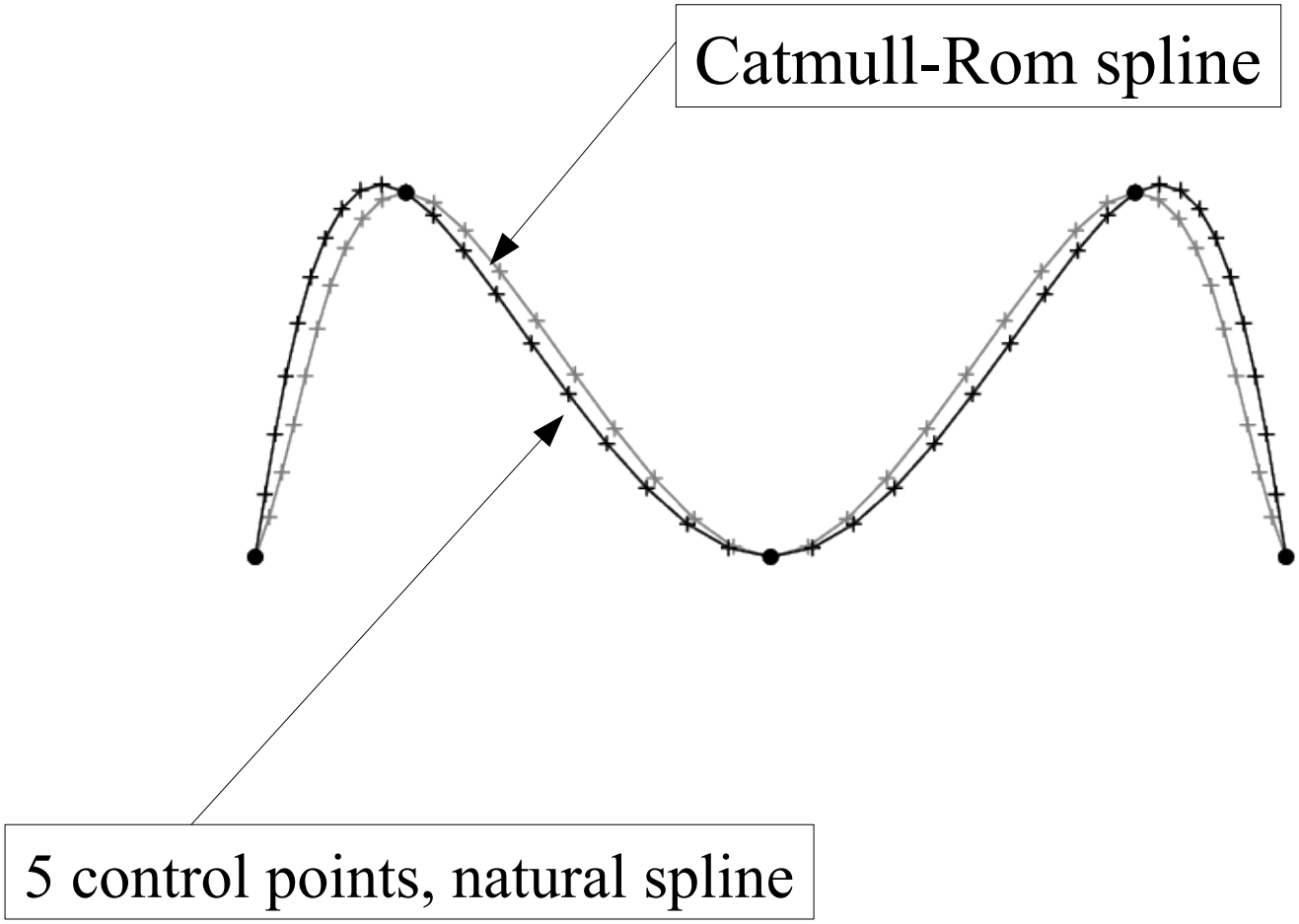
$$\begin{cases} a_{[i]0} = x_i \\ a_{[i]1} = x'_i \\ a_{[i]2} = 3(x_{i+1} - x_i) - 2x'_i - x'_{i+1} \\ a_{[i]3} = 2(x_i - x_{i+1}) + x'_i + x'_{i+1} \end{cases}$$

,to get the polynomial in each portion :

$$x_{[i]}(\bar{u}) = a_{[i]0} + a_{[i]1}\bar{u} + a_{[i]2}\bar{u}^2 + a_{[i]3}\bar{u}^3 \ , \ 0 \le \bar{u} < 1$$

From the global parameter u, we have to find in which portion we are (the value of i) , then compute right polynomial...

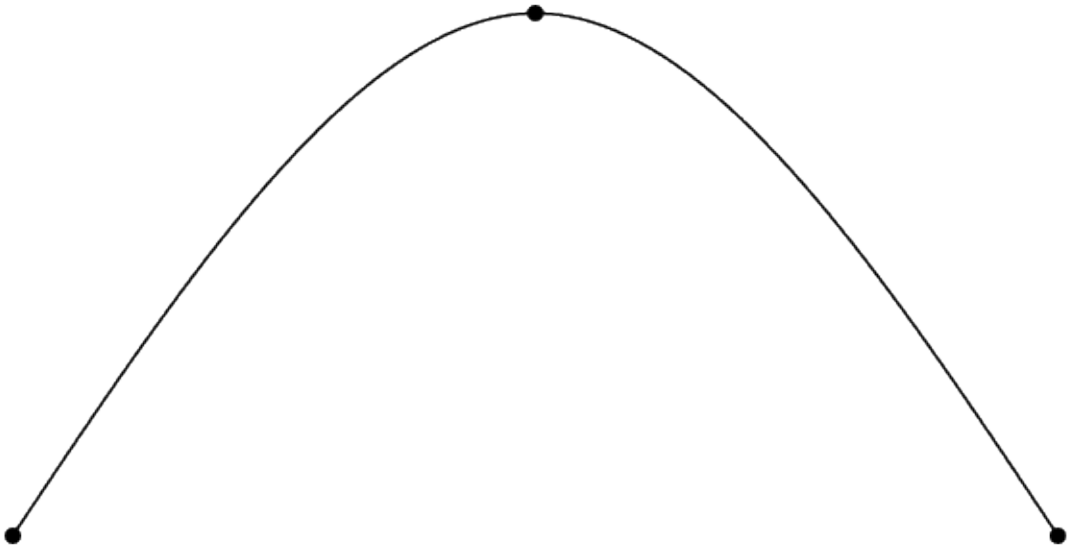
Splines



Splines

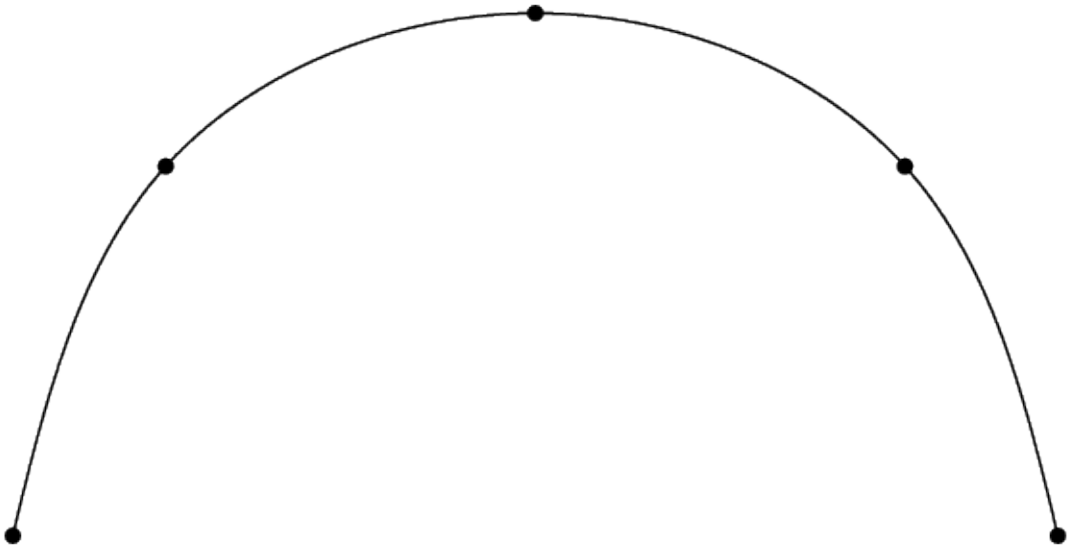
- An experiment
  - We approximate a circle by a number of increasing points
  - Simultaneously , ~~the order of the approximation~~ the number of pieces increases.
  - In all the cases, the curve is $\not\in_{\infty} C_2$

Splines



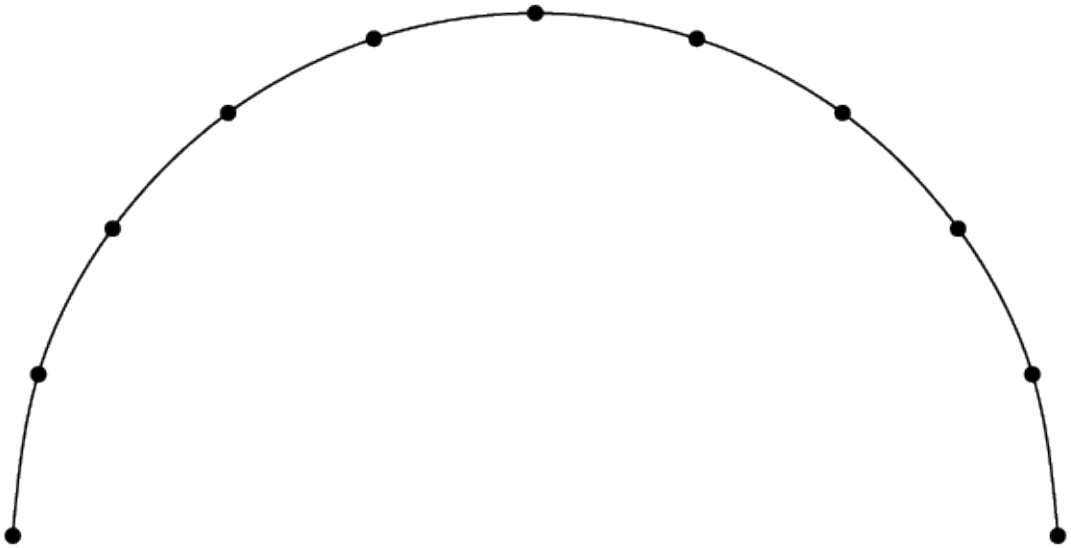
3 points, order 3!

Splines



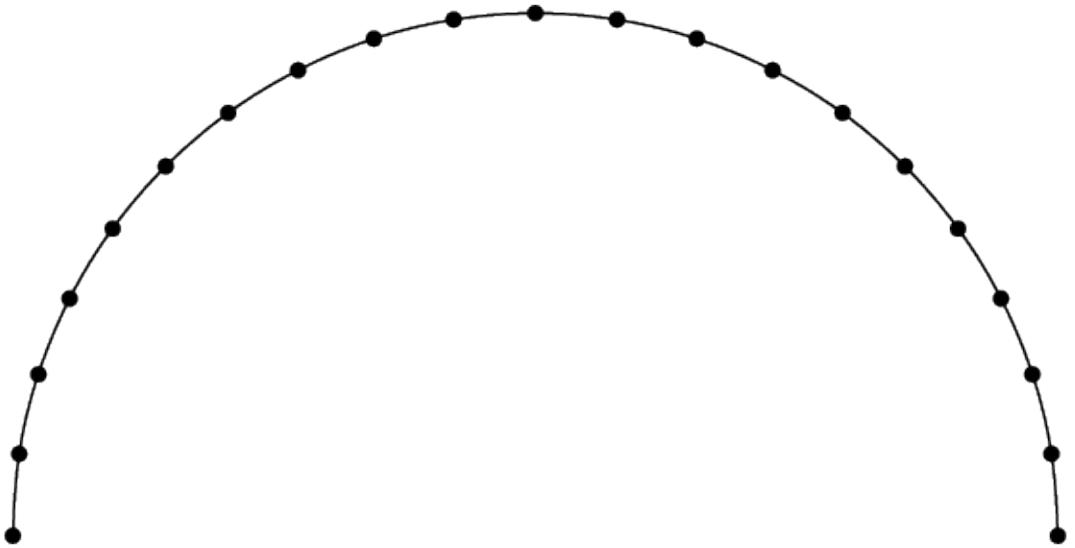
5 points

Splines



11 points

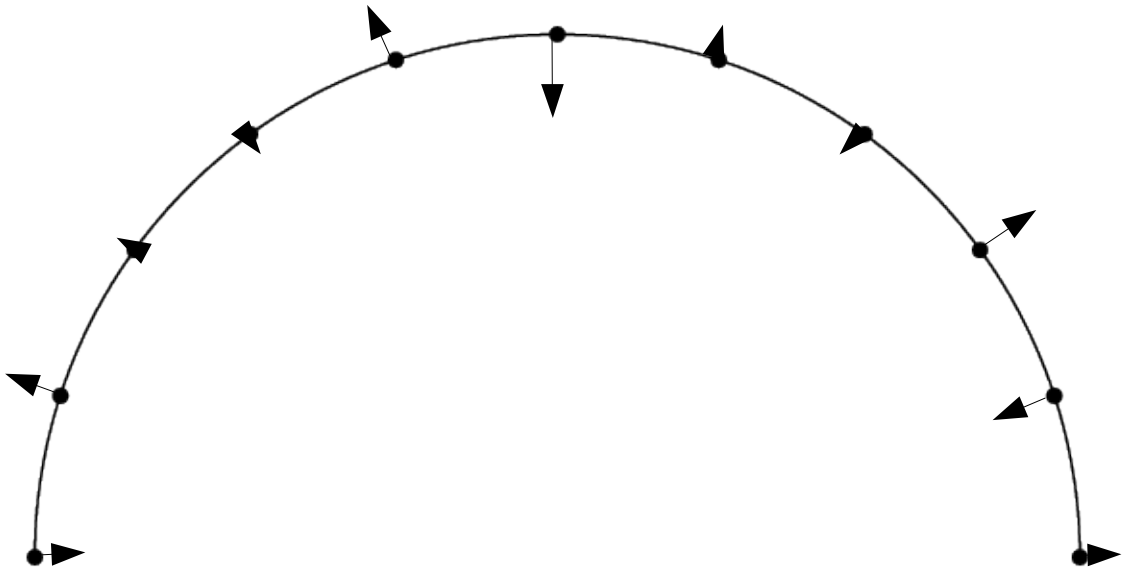
Splines



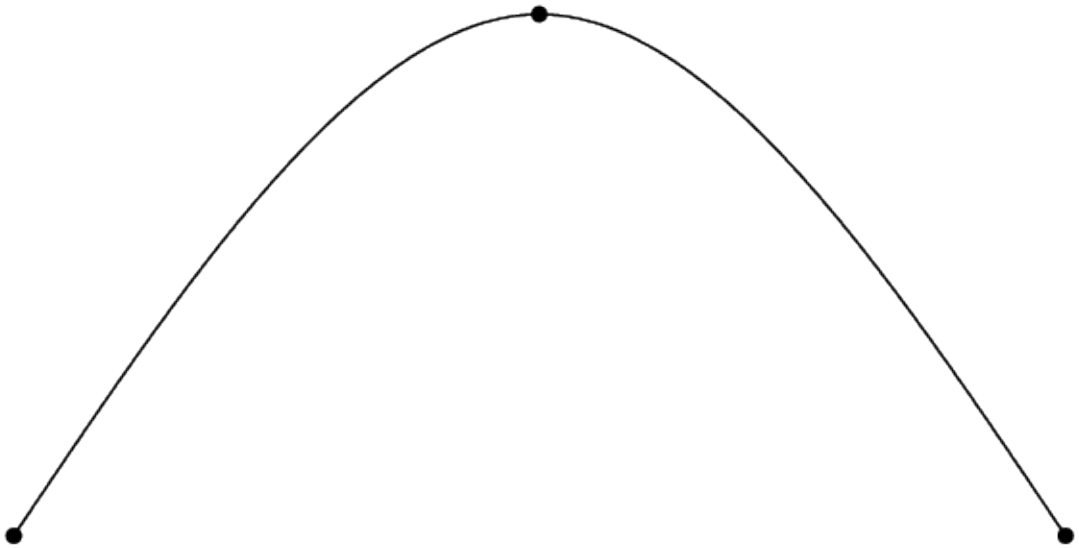
21 points

Splines

- Random perturbation
  - Each point is moved radially by a value between -0.5 and +0.5 % of the circle's radius

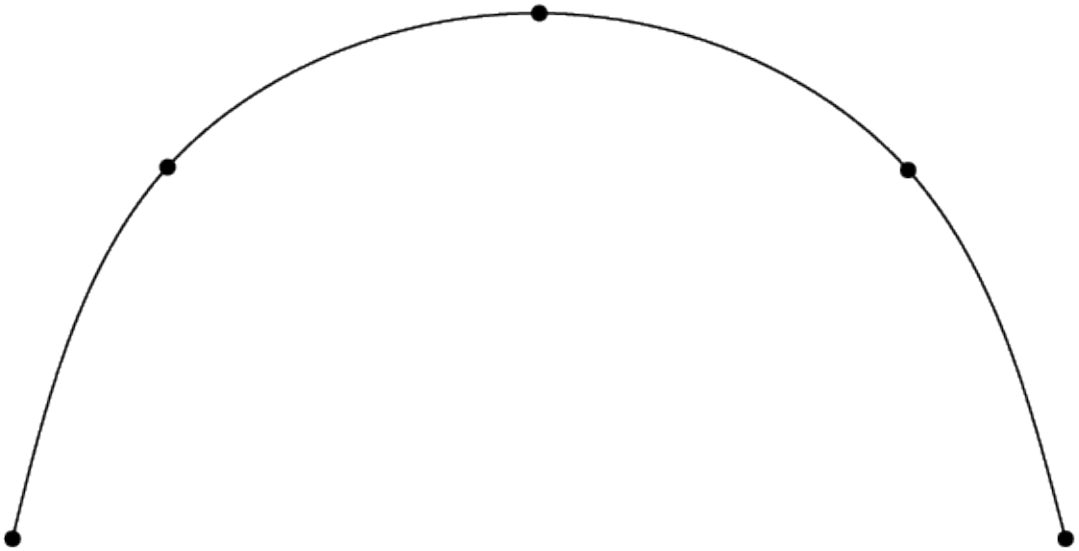


Splines



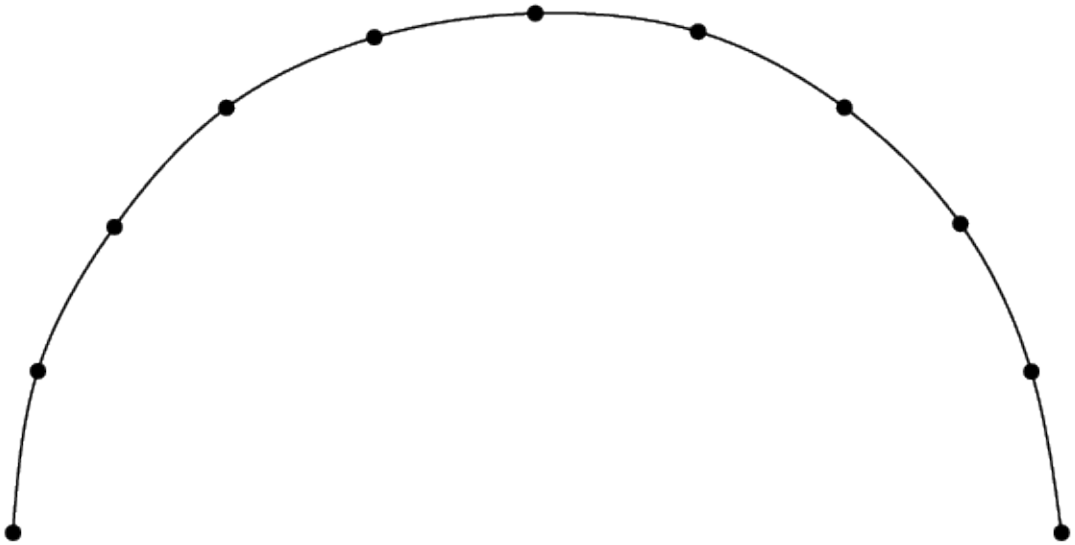
3 points, random perturbation 1%

Splines



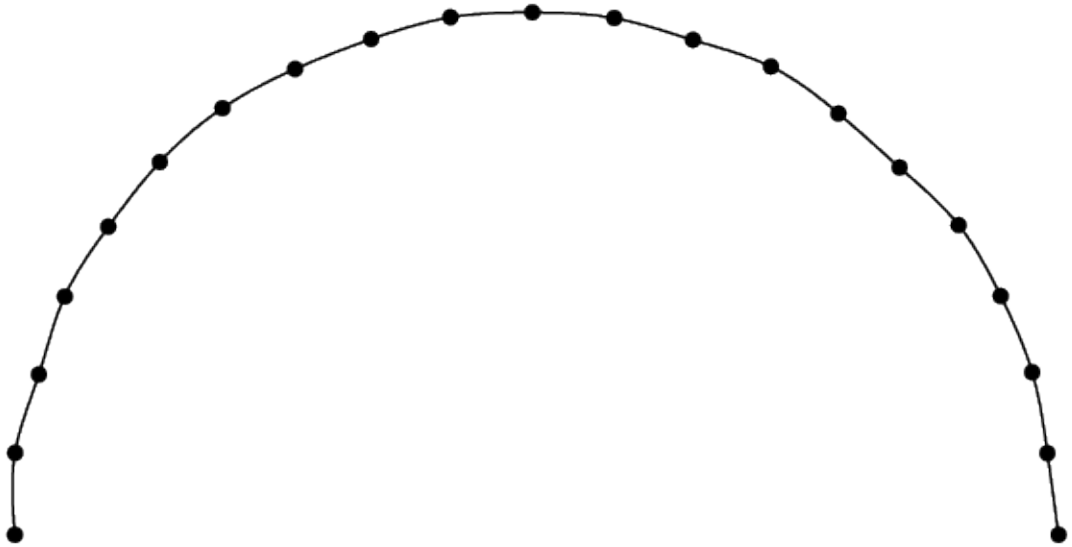
5 points, random perturbation 1%

Splines



11 points, random perturbation 1%

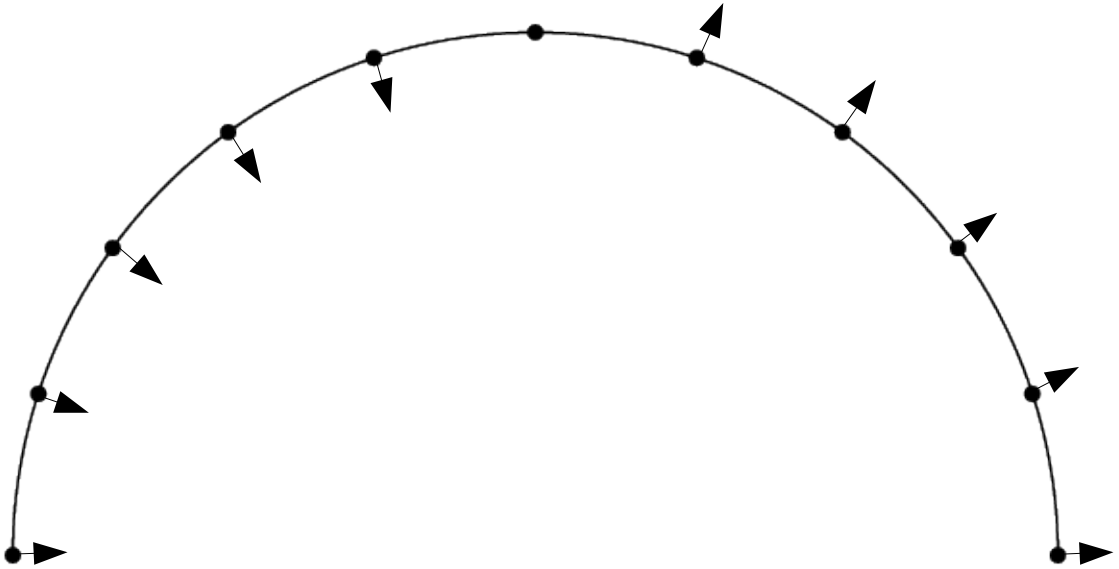
Splines



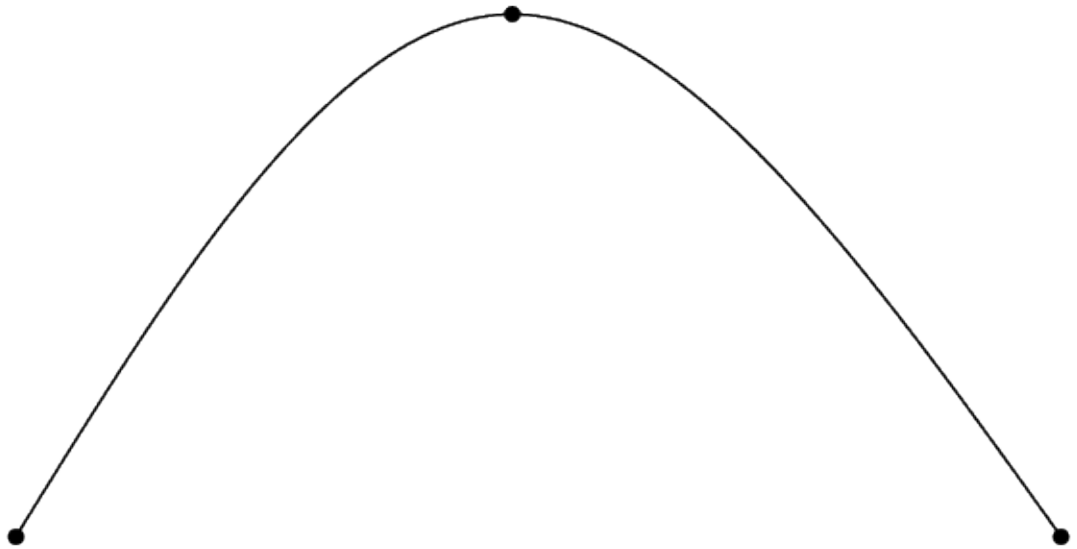
21 points, random perturbation 1%

Splines

- Deterministic perturbation
  - Each point is shifted radially depending on its position by -5 or +5 % of the circle's radius

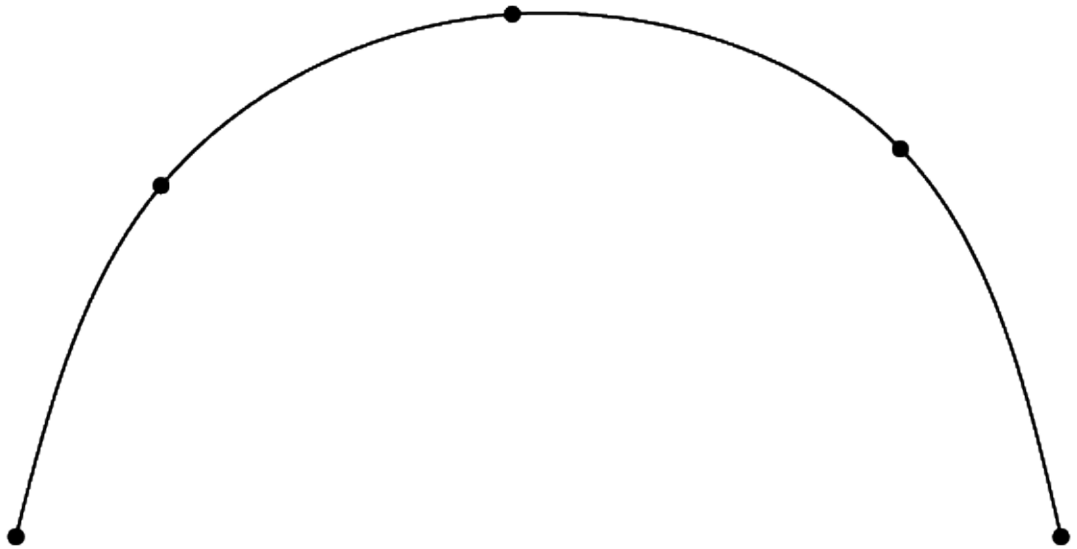


Splines



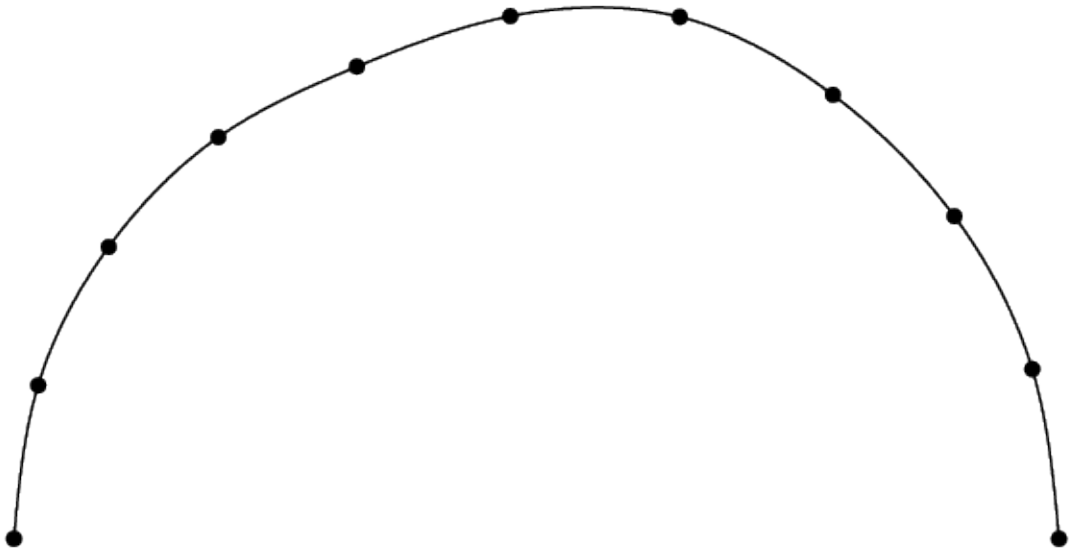
3 points, deterministic perturbation 5%

Splines



5 points, deterministic perturbation 5%

Splines

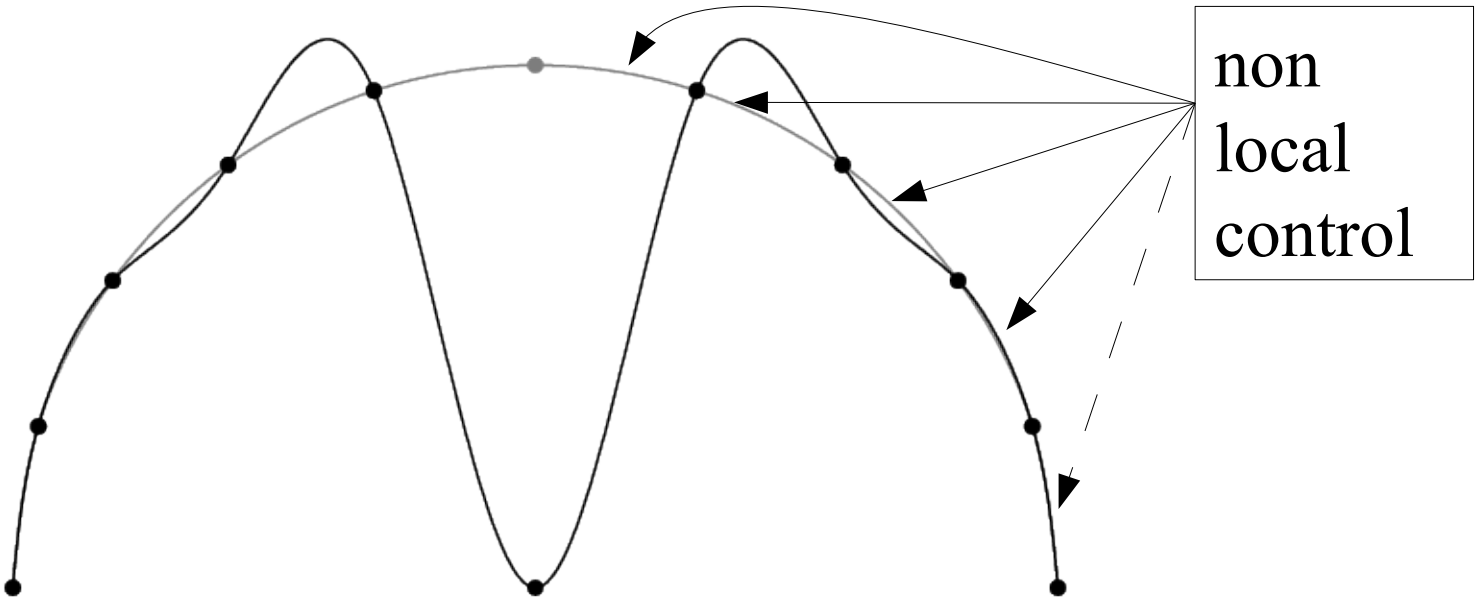


11 points, deterministic perturbation 5%

Splines

21 points, deterministic perturbation 5%

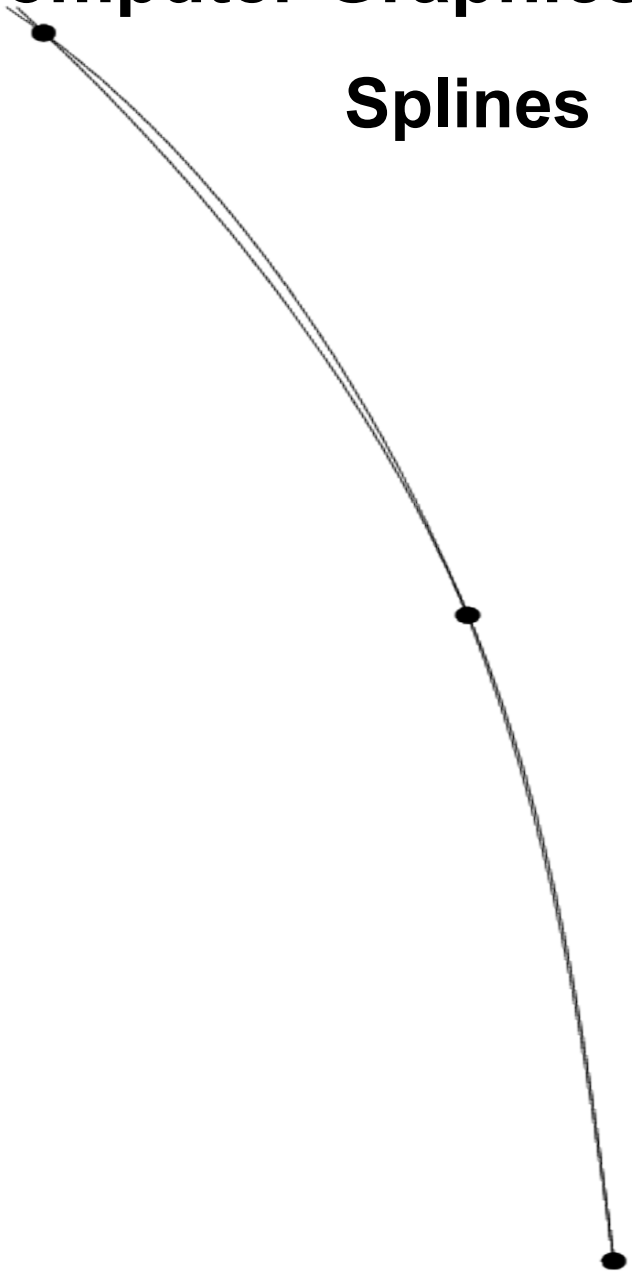
Splines



11 points

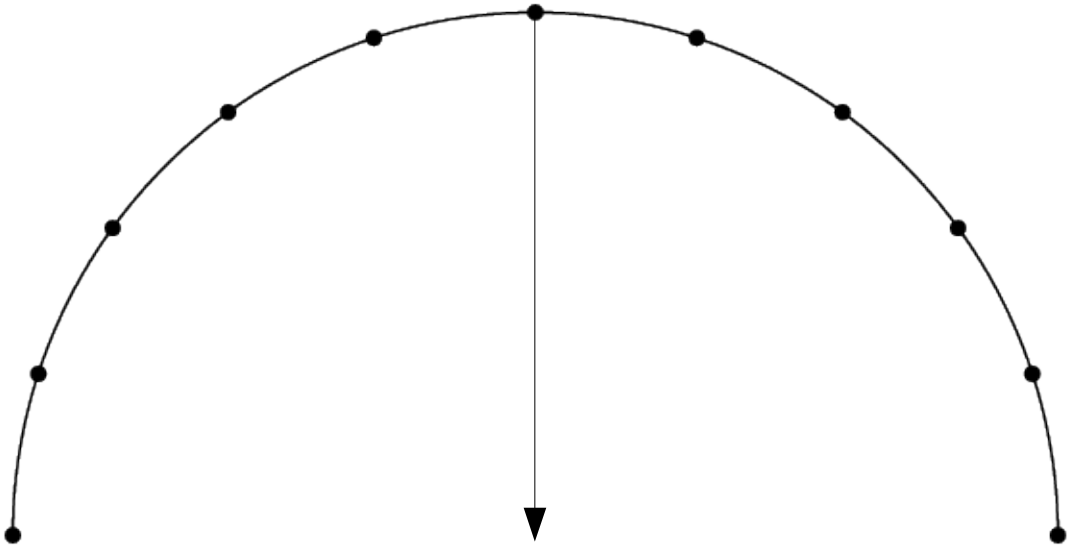
Computer Graphics

Splines

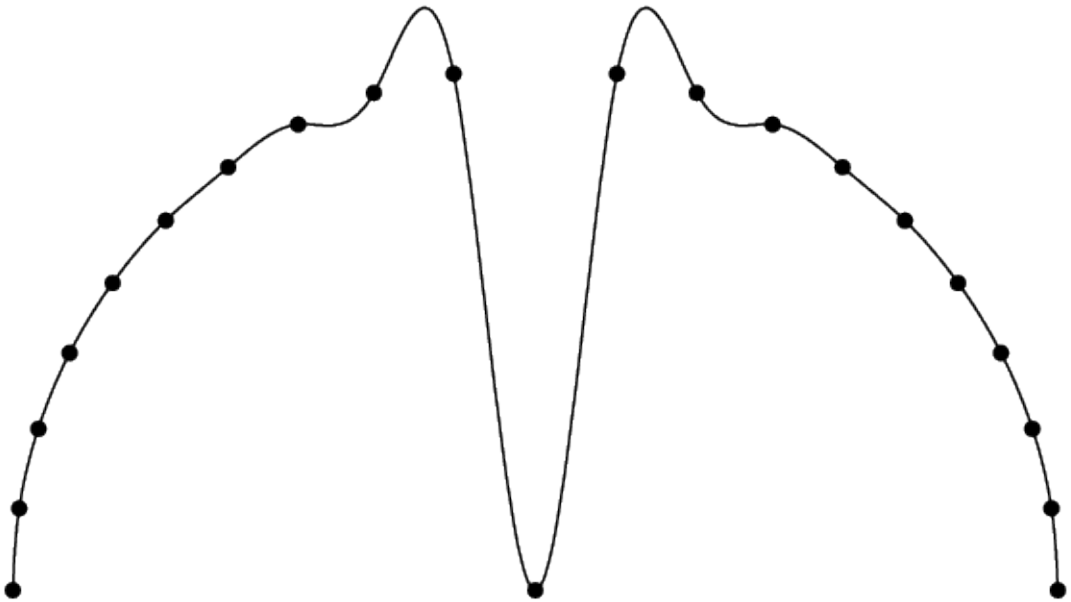


Splines

- Perturbation of a point
  - We shift one point by a significant amount

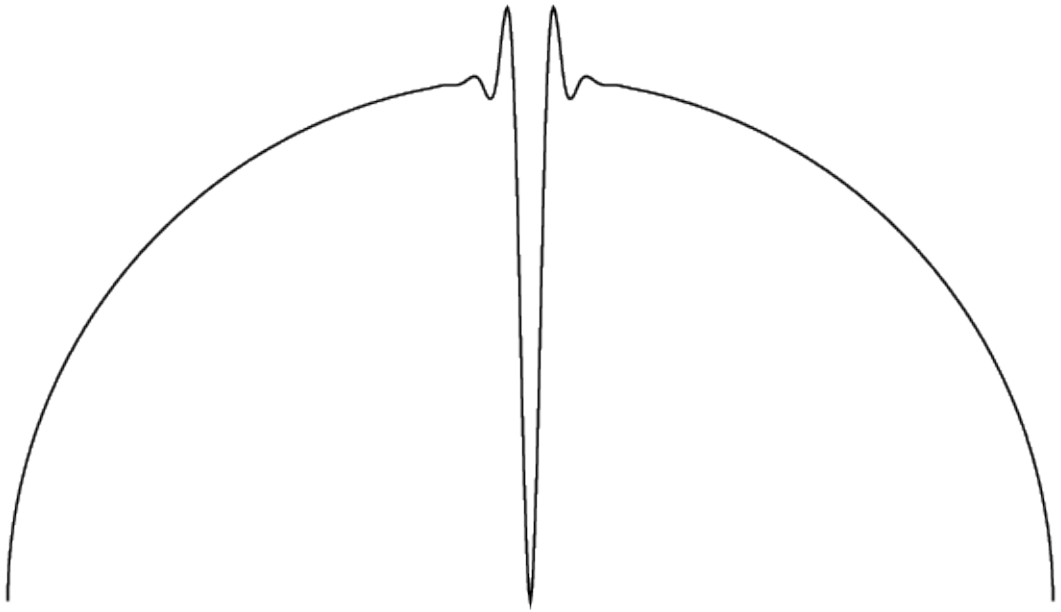


Splines



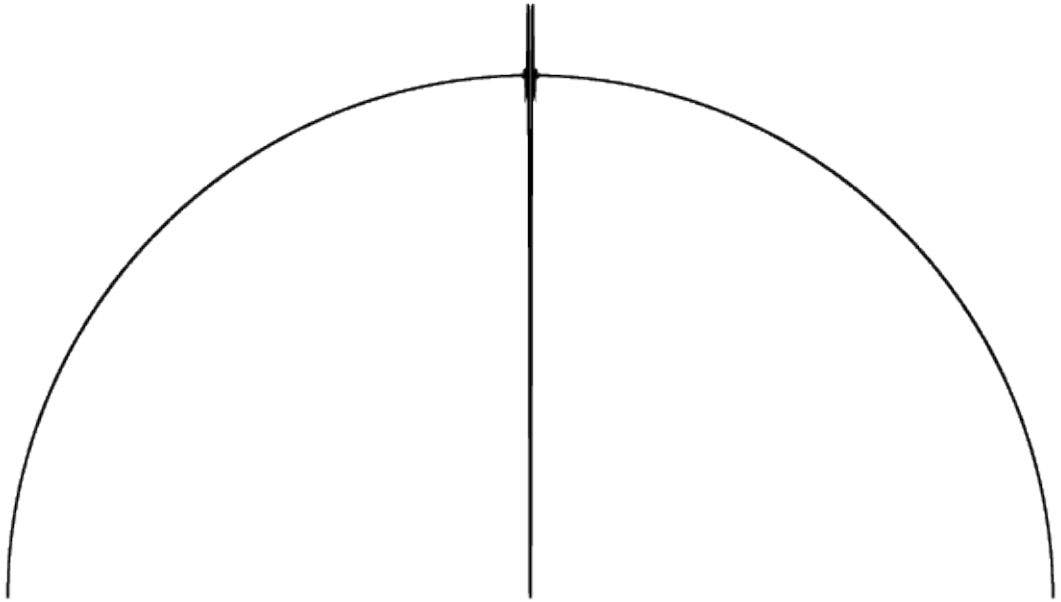
21 points

Splines



99 points

Splines



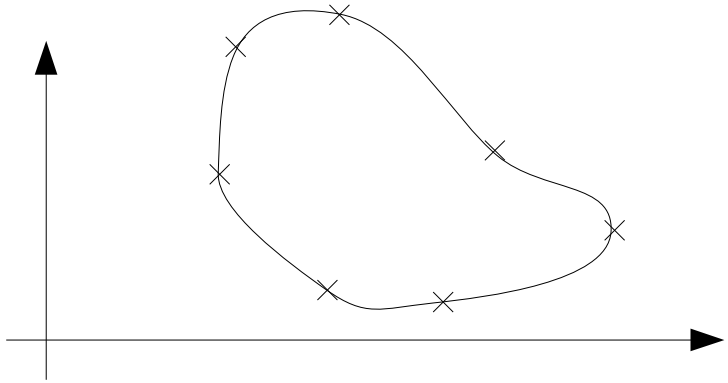
999 points

Splines

- Stable interpolation scheme
- Weak Runge phenomenon
- The displacement of a point yet affects all the curve
  - Nevertheless, the perturbation fades very quickly further away from the shifted point
  - « Overshoots » are limited.

Splines

- Closed curve ?
  - The curve can be closed, just impose everywhere that the second derivative is continuous.



Splines

- Instead of

$$x''_{[0]}(0)=0 \Leftrightarrow 2a_{[0]2}=0$$
$$x''_{[n-2]}(1)=0 \Leftrightarrow 2a_{[n-2]2}+6a_{[n-2]3}=0$$

... we have

$$x''_{[n-2]}(1)=x''_{[0]}(0) \Leftrightarrow 2a_{[n-2]2}+6a_{[n-2]3}=2a_{[0]2}$$

Circulant matrix

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

and $x_{n-1}=x_0$, $x'_{n-1}=x'_0$

Splines

- 3D Curves
  - Minimal order so that a curve can have a torsion (non planar curve)
    - Let's consider a Lagrange interpolation
    - 2 points → on a straight line (no curvature)
    - 3 points → in a plane (no torsion)
    - 4 points → torsion becomes possible

$$P(u) = \sum_{i=0}^{n-1} P_i \, l_i^p(u)$$

  - Minimal order to join smoothly two arbitrarily oriented curves = 3

Bézier curves

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)

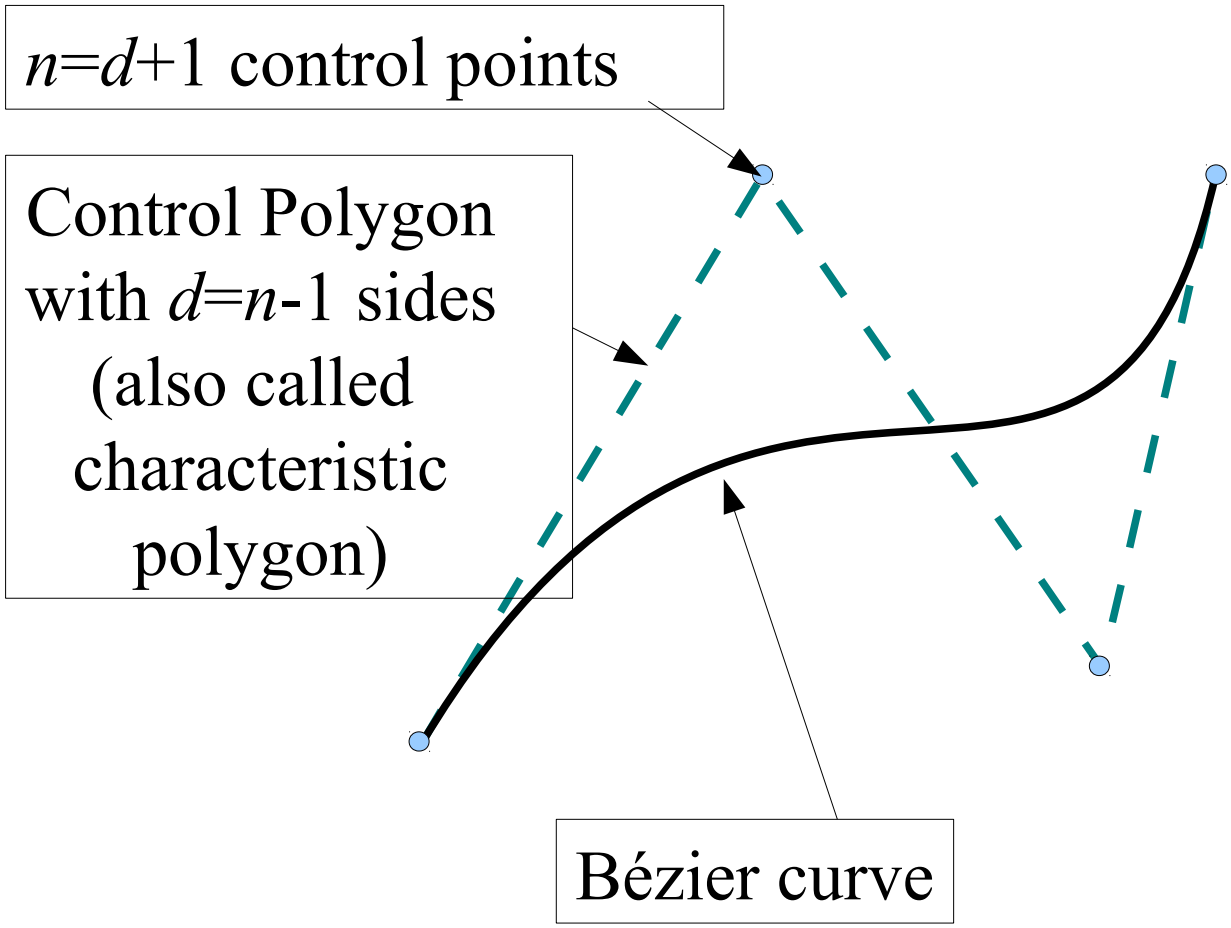
  - Computer graphics
  - In geometrical modelling and CAD, they tend to be replaced by more general techniques (NURBS)

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

Bézier curves

- 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
    - We sometimes call them blending functions
    - The curve is described as one polynomial (unlike splines)

Bézier curves

- 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

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

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

$\binom{d}{i}$	i=0	1	2	3	4	5
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

Bézier curves

- Bernstein polynomials

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

Binomial coefficients

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

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

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

Partition of unity

- Property of affine invariance
  - It is a useful property that the curves we define for a set of control points can undergo linear affine transformations.
    - Let P_i^* the affine transformation of the control points P_i
    - Let $P^*(P_i)$ the affine transformation of the points of the curve $P(P_i)$ defined from the original points P_i
    - Let $P(P_i^*)$ the new curve based on the modified control points P_i^*, with the same parametrization.
  - The affine invariance is verified iff $P^*(P_i) = P(P_i^*)$ for all u.

Partition of unity

- 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

Let P a parametric curve built this way :

$$P(u)=\sum_{0}^{n-1} P_i K_i^n(u)$$

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

$$P(P_i^*)=\sum_{0}^{n-1}(P_i+t)K_i^n(u)=\sum_{0}^{n-1}P_i K_i^n(u)+\sum_{0}^{n-1}t\,K_i^n(u)$$

$$=P(u)+\sum_{0}^{n-1}t\,K_i^n(u)$$

Partition of unity

$$=P(u)+t=P^*(P_i) \quad \text{iff} \quad \sum_{0}^{n-1}K_i^n(u)=1$$

Partition of unity

- For the other multiplicative transformations

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

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

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

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

Bézier curves

- 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 unique maximum at $u = i/d$

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

Bézier curves

- Demonstration of $B_i^{'d}(u) = d\left(B_{i-1}^{d-1}(u) - B_i^{d-1}(u)\right)$

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

$$\binom{d}{i} = \frac{d!}{(d-i)!\, i!} = \binom{d-1}{i-1} \frac{d}{i}$$

$$\binom{d}{i} = \frac{d!}{(d-i)!\, i!} = \binom{d-1}{i} \frac{d}{d-i}$$

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

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

QED

Bézier curves

- 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

- Demonstration of the recurrence relations

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

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

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

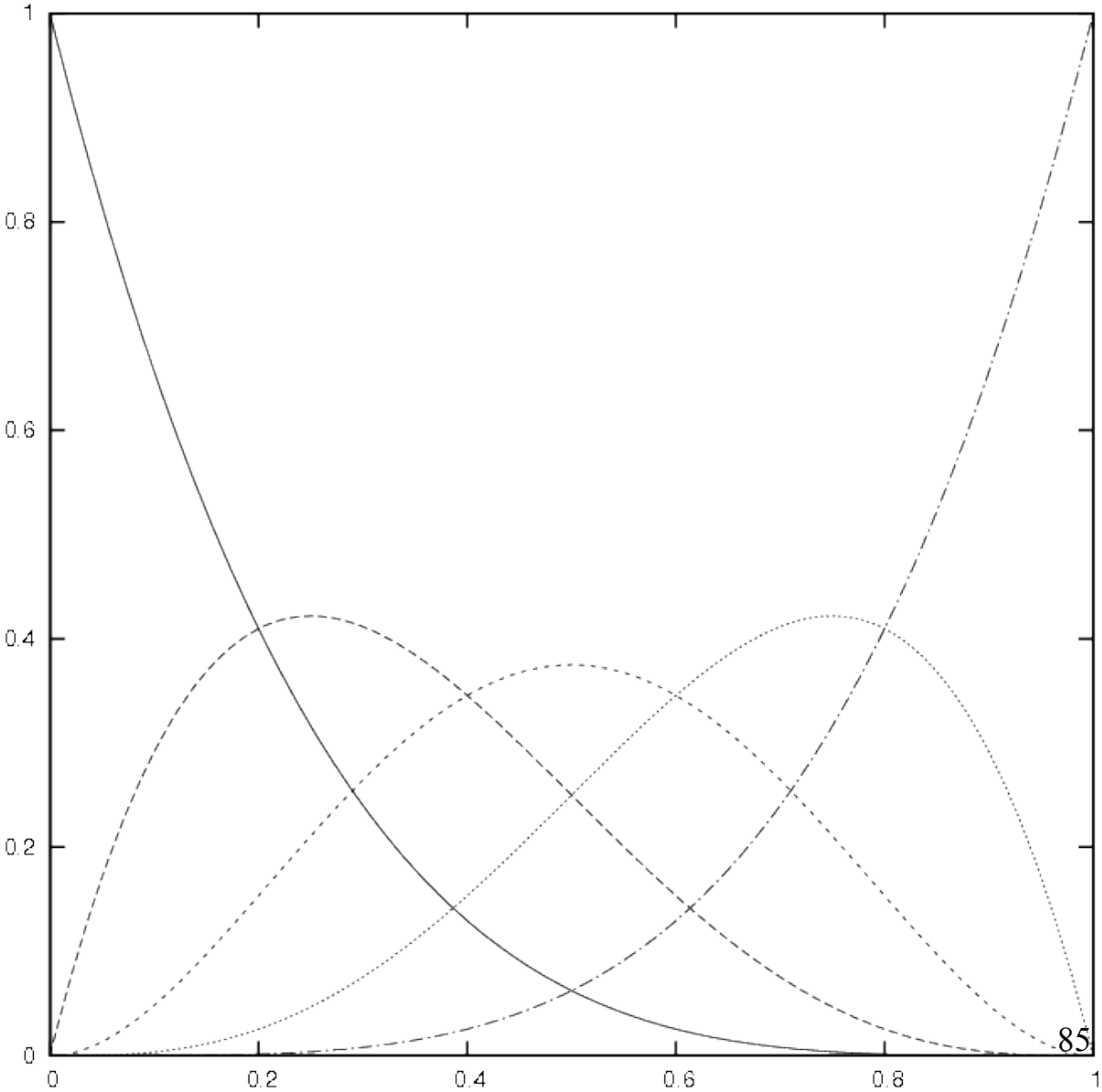
$$(1-u) \cdot B_i^{d-1}(u) \qquad u \cdot B_{i-1}^{d-1}(u)$$

QED

Bézier curves

- Degree 4
- No negative values

  Therefore, no value above 1!



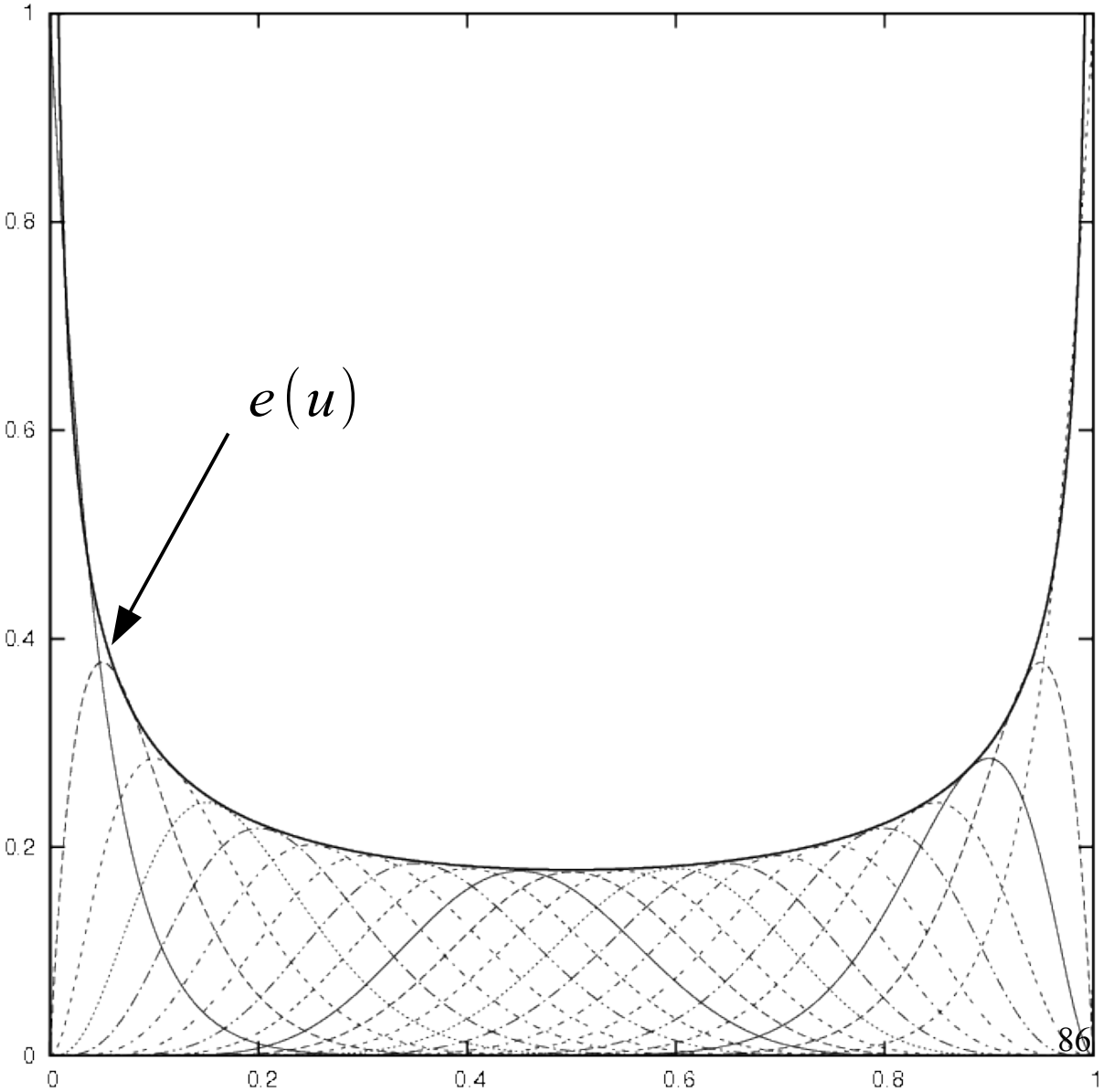
Computer Graphics

Bézier curves

- Degree 20
- No extreme values

  Therefore, no value above 1!
- Existence of a limiting envelope

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

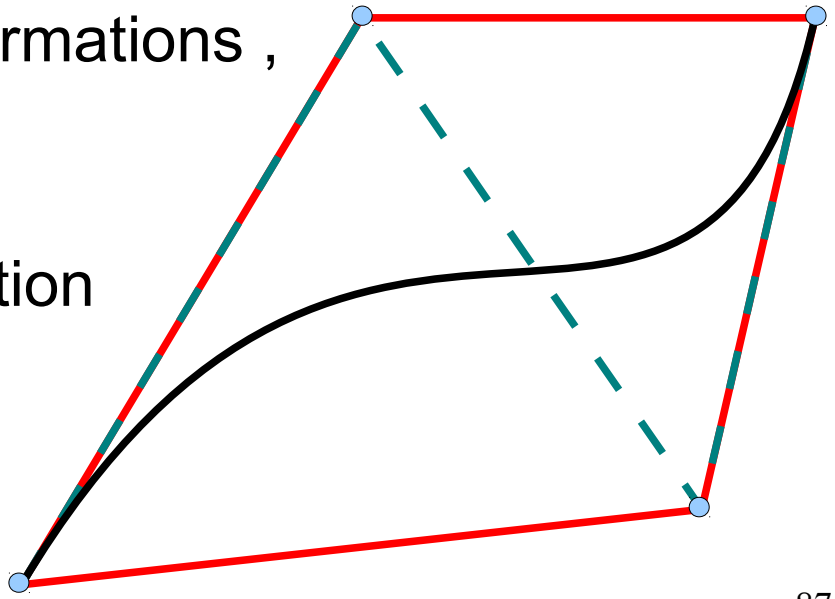


Bézier curves

- The characteristics of Bernstein polynomials involve that the Bézier curve

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

  - interpolates P_0 et 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

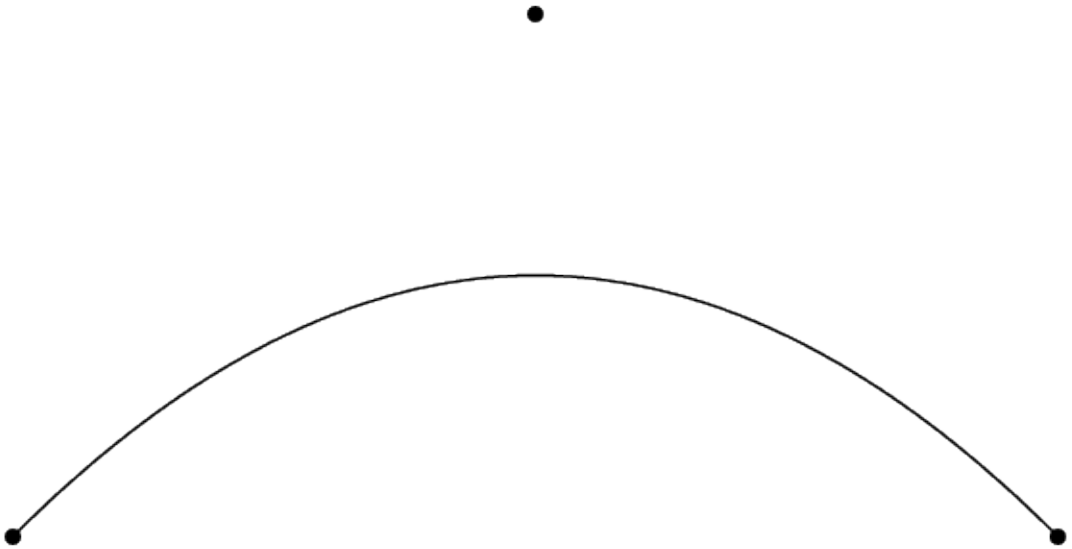
(following)

- 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

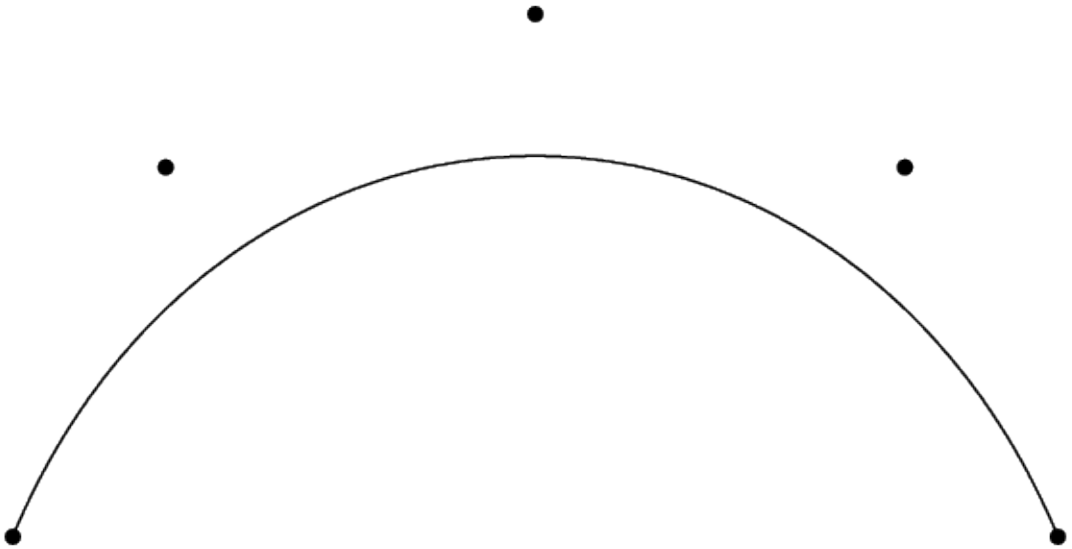
- Same examples as shown earlier on Lagrange interpolation
  - Circle with an increasing number of points
  - Perturbation of the control points

Bézier curves



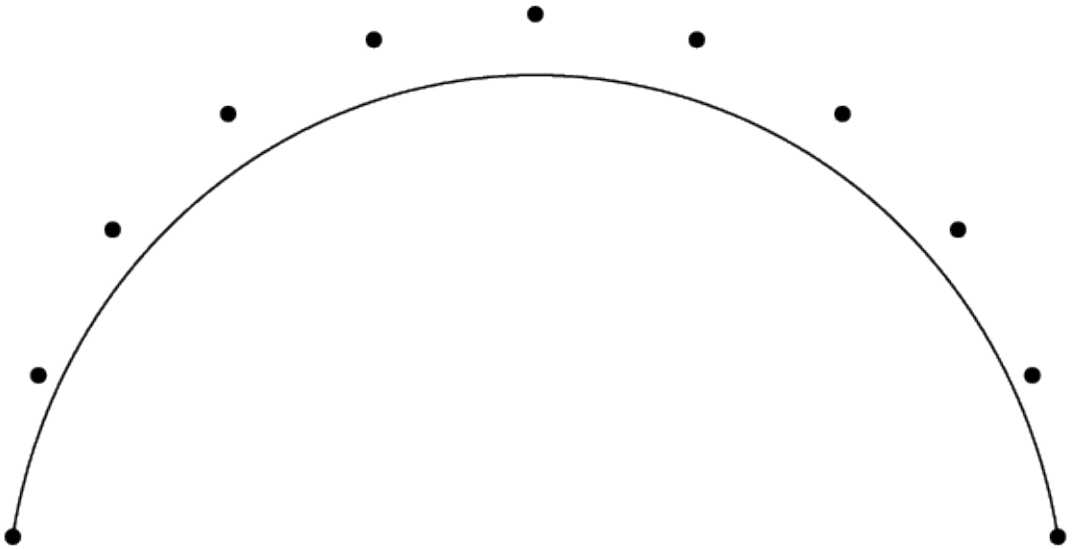
Degree 2

Bézier curves



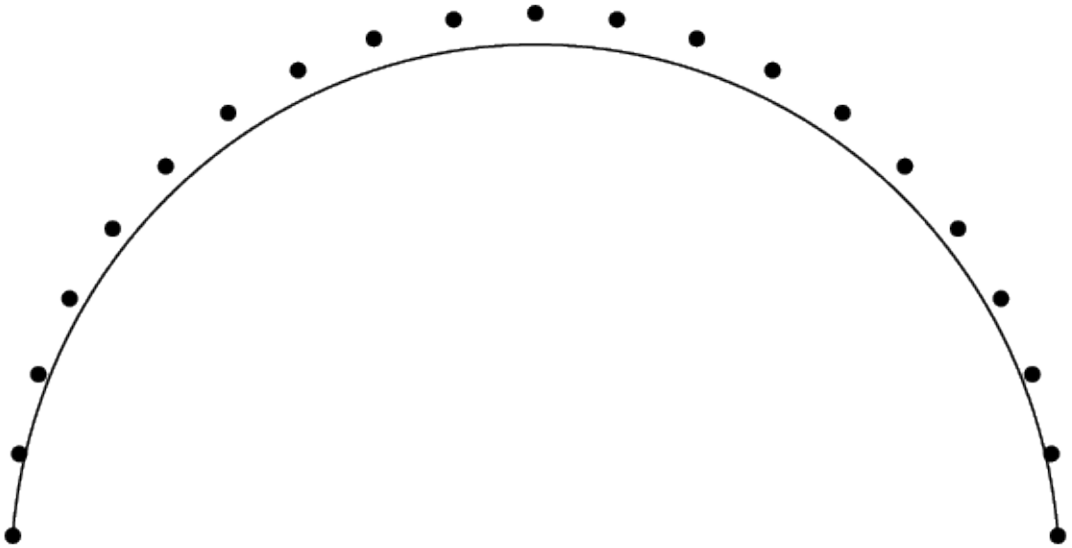
Degree 4

Bézier curves



Degree 10

Bézier curves



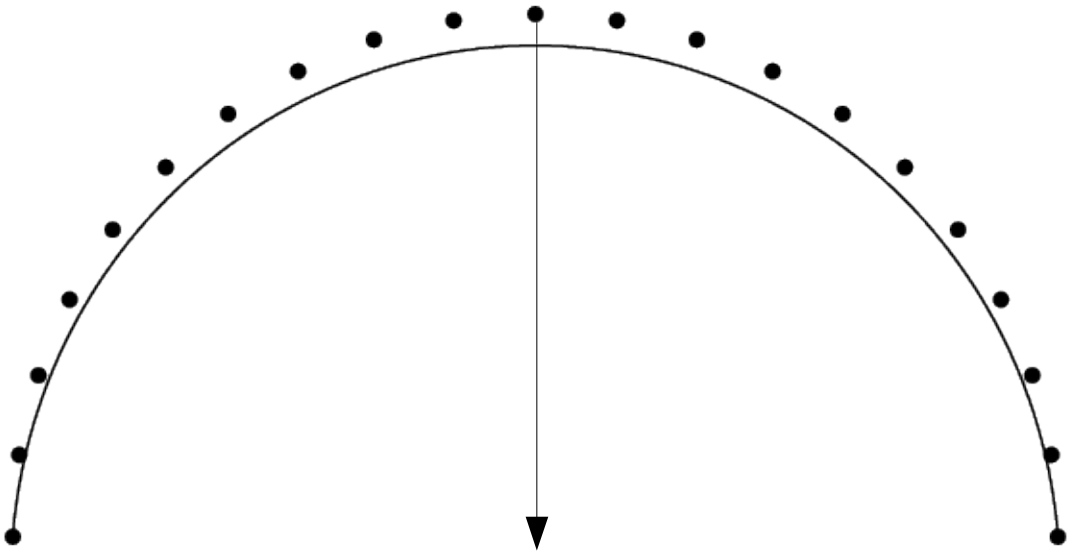
Degree 20

Bézier curves

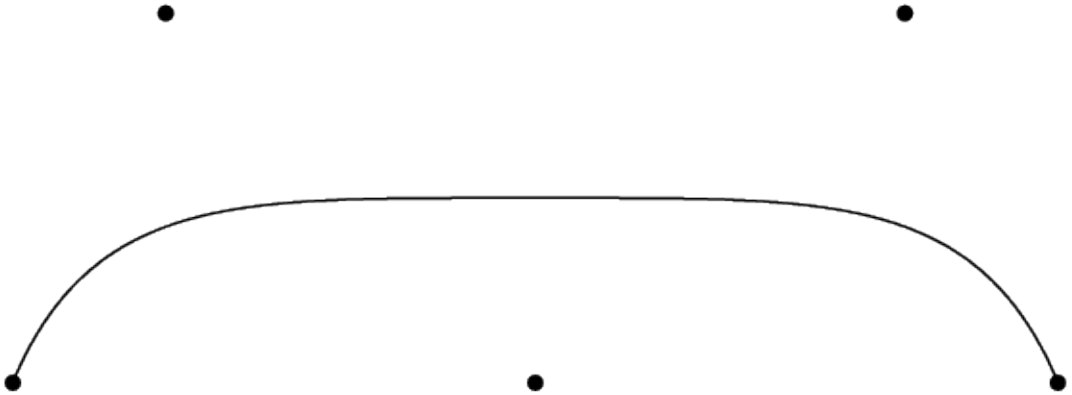
- When the number of control points increases, the curve tends to the control polygon (under the assumption that the control polygon itself converges to a smooth curve ...)
- The approximation involves a substantial error between the curve and the control points
  - However, an interpolation is not the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

Bézier curves

- Perturbation of a point
  - We shift the indicated point

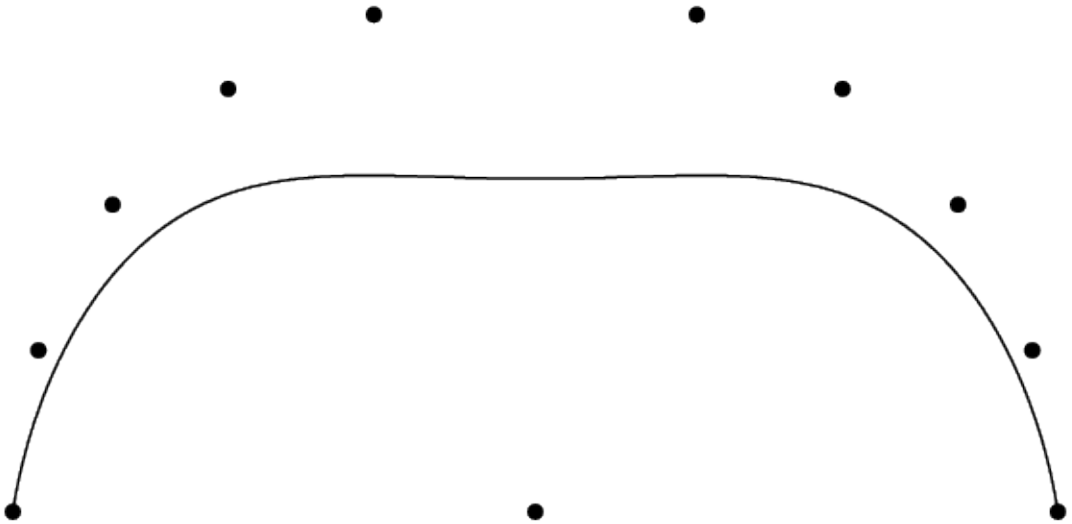


Bézier curves



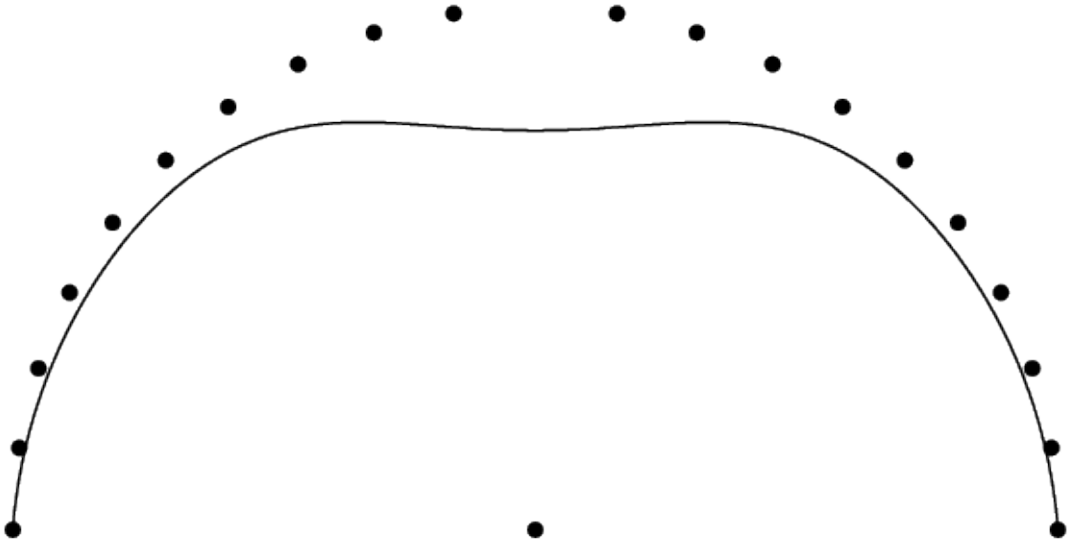
Degree 4

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

Bézier curves

- 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]

$$Q_0 = P_0$$

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

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

Bézier curves

- Demonstration of Forrest's equations :

$$P(u)=\sum_{i=0}^{d} P_i B_i^d(u) \qquad Q(u)=\sum_{i=0}^{d+1} Q_i B_i^{d+1}(u)$$

$$Q(u)=P(u) \quad \forall u\in[0,1] \qquad Q_i=f(P_0 \dots P_d)$$

  - Let's express $B_i^d(u)$ in function of $B_i^{d+1}(u)$

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

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

Bézier curves

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

- We replace the terms of the binomial $\binom{d}{i}=\frac{d!}{(d-i)!\,i!}$

$$\binom{d+1}{i}=\frac{(d+1)!}{(d+1-i)!\,i!}=\binom{d}{i}\frac{d+1}{d+1-i}$$

$$\binom{d+1}{i+1}=\frac{(d+1)!}{(d-i)!\,(i+1)!}=\binom{d}{i}\frac{d+1}{i+1}$$

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

Bézier curves

- We split up $B_i^d(u) \longrightarrow B_i^d(u) = (1-u)\,B_i^d(u) + u\,B_i^d(u)$

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

$$Q(u) = P(u) \Rightarrow \sum_{i=0}^{d+1} Q_i B_i^{d+1}(u) = \sum_{i=0}^{d} P_i \left(\frac{d+1-i}{d+1} B_i^{d+1}(u) + \frac{i+1}{d+1} B_{i+1}^{d+1}(u) \right)$$

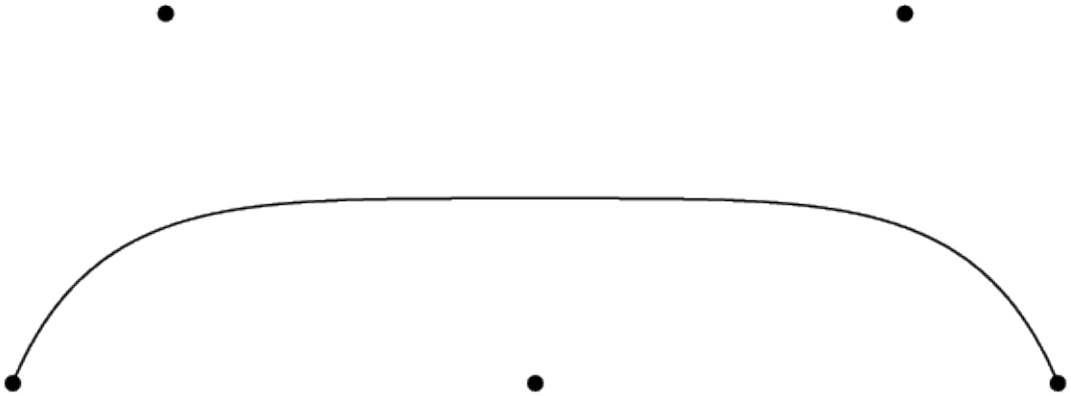
$$= \sum_{i=0}^{d} P_i \left(1 - \frac{i}{d+1} \right) B_i^{d+1}(u) + \sum_{i=1}^{d+1} P_{i-1} \frac{i}{d+1} B_i^{d+1}(u)$$

$$= \sum_{i=1}^{d} \left(P_i \left(1 - \frac{i}{d+1} \right) + P_{i-1} \frac{i}{d+1} \right) B_i^{d+1}(u) + P_0 B_0^{d+1}(u) + P_d B_{d+1}^{d+1}(u)$$

Q_i QED Q_0 Q_{d+1}

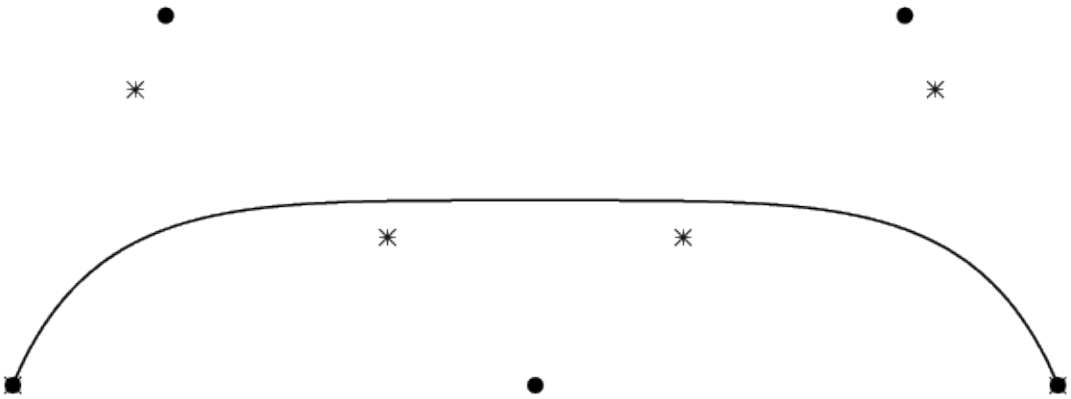
Bézier curves

- Degree elevation in practice ...



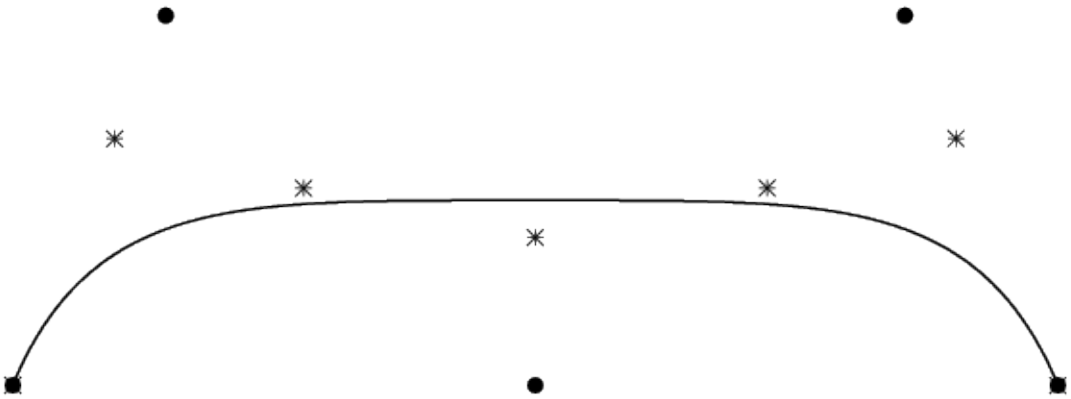
Degree 4

Bézier curves



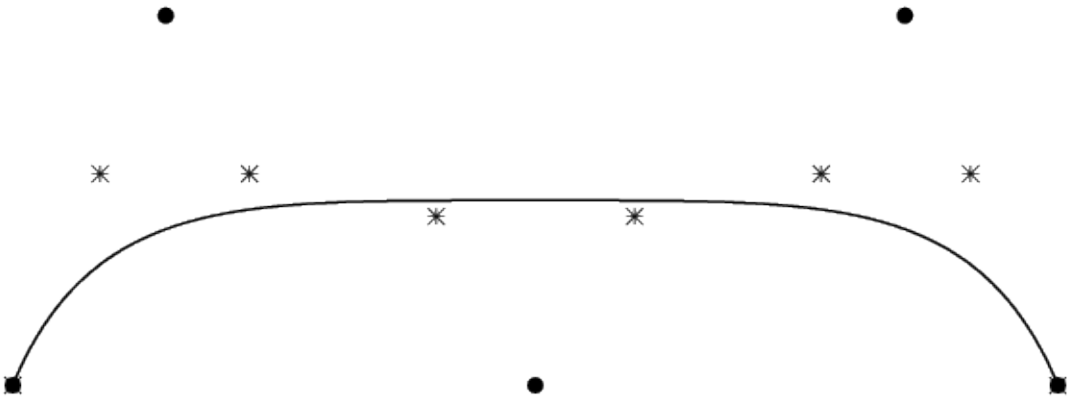
Degree 5

Bézier curves



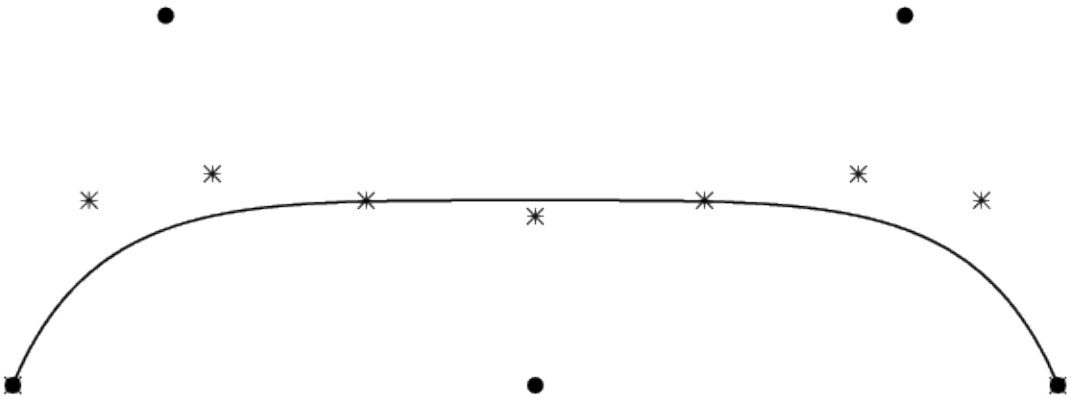
Degree 6

Bézier curves



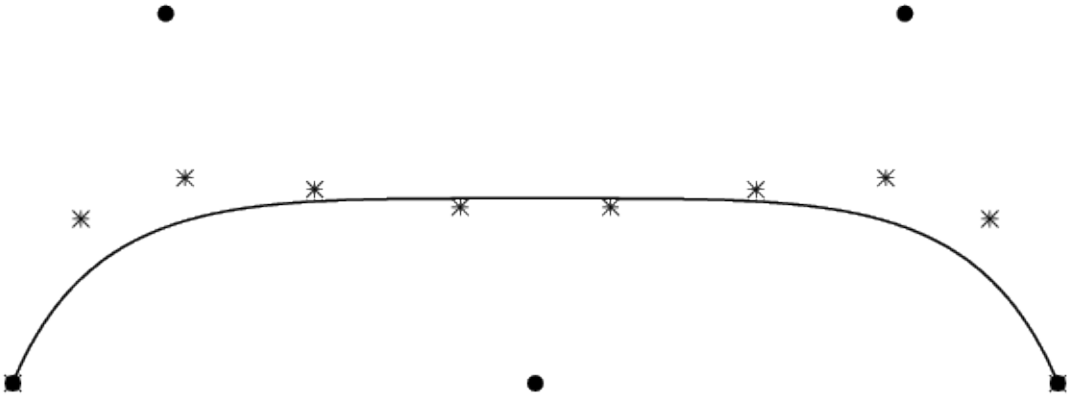
Degree 7

Bézier curves



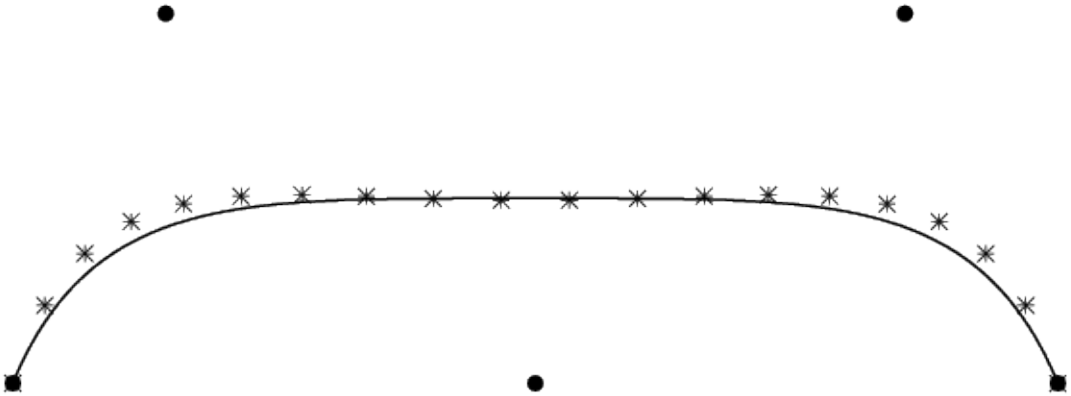
Degree 8

Bézier curves



Degree 9

Bézier curves



Degree 21

Bézier curves

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

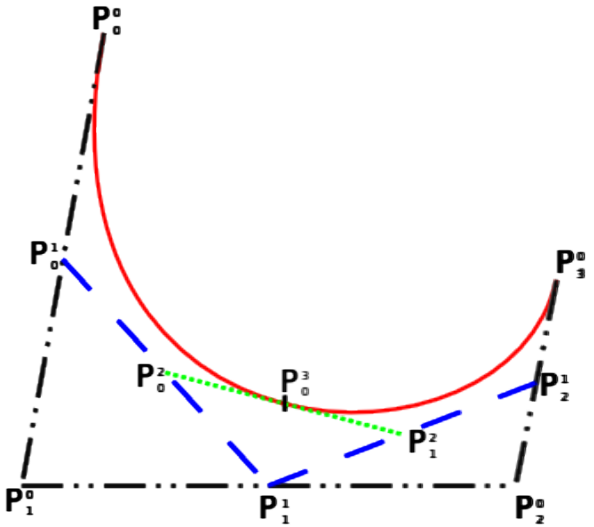
Bézier curves

- Principle of De Casteljau's algorithm
  - Construction of the centroids P_i^1 of the control points P_i^0 :

$$P_i^1 = (1-u)\,P_i^0 + u\,P_{i+1}^0$$

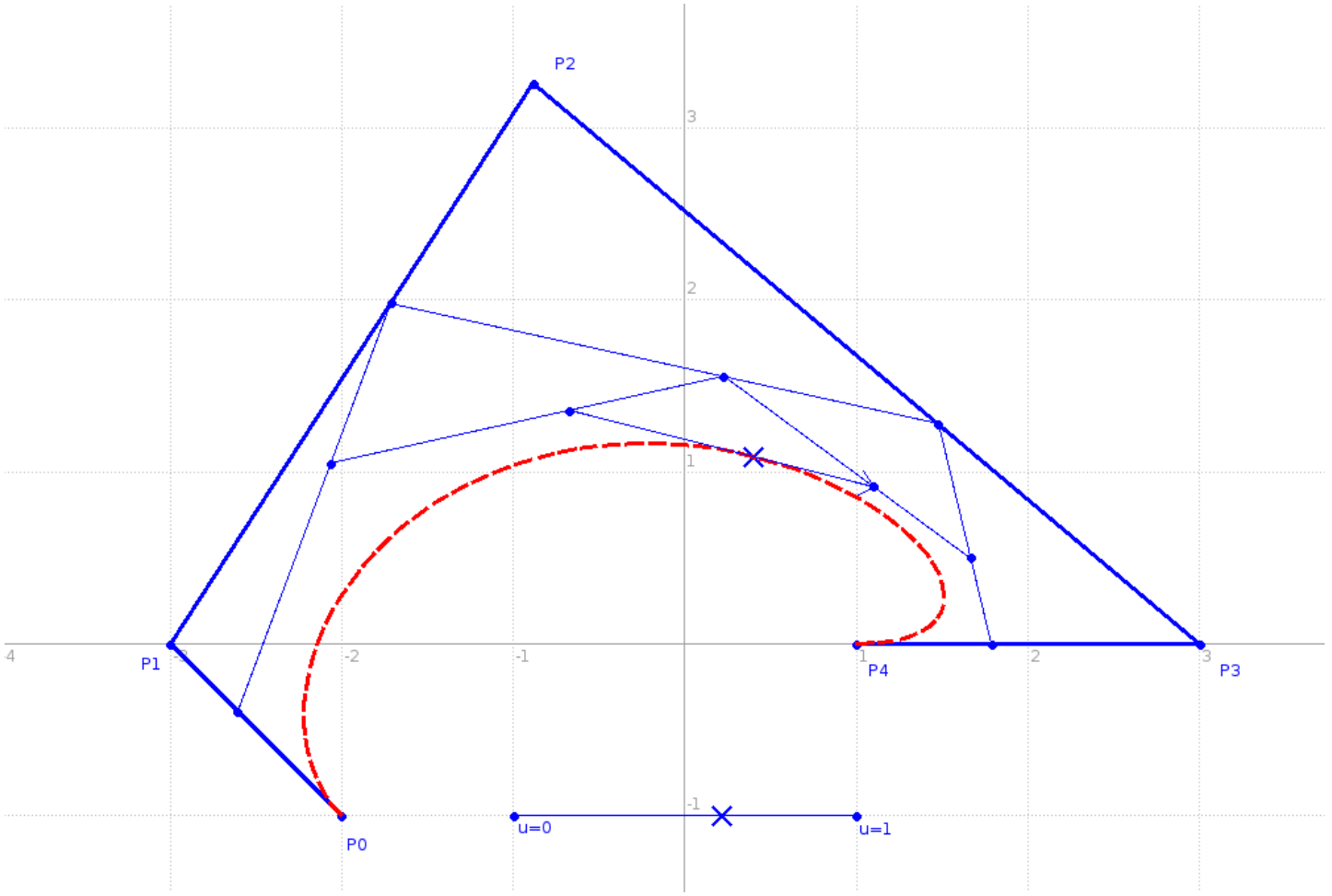
  - We continue with P_i^2 ....
    - As far as possible, until only one control point remains, That control point is $P(u)$.

P_0^d



Bézier curves

- 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.

Bézier curves

- Demonstration with the help of recurrence relations

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

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

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

$$\rightarrow \quad P(u)=\sum_{i=1}^{d-1} P_i\left((1-u)B_i^{d-1}(u)+u\,B_{i-1}^{d-1}(u)\right)$$
$$+P_0(1-u)B_0^{d-1}(u)+P_d\,uB_{d-1}^{d-1}(u)$$

By gathering the terms $B_i^{d-1}(u)$:

$$P(u)=\sum_{i=0}^{d-1}\left[(1-u)P_i+uP_{i+1}\right]B_i^{d-1}(u)$$

by setting $P_i^1=(1-u)P_i^0+uP_{i+1}^0 \longrightarrow P(u)=\sum_{i=0}^{d-1} P_i^1 B_i^{d-1}(u)$

Bézier curves

- We do it again with $P(u)=\sum_{i=0}^{d-1} P_i^1 B_i^{d-1}(u)$

$$P(u)=\sum_{i=1}^{d-2} P_i^1\left((1-u)B_i^{d-2}(u)+u\,B_{i-1}^{d-2}(u)\right)$$

$$+\,P_0^1(1-u)B_0^{d-2}(u)+P_{d-1}^1 u\,B_{d-2}^{d-2}(u)$$

$$P(u)=\sum_{i=0}^{d-2}\left[(1-u)P_i^1+u\,P_{i+1}^1\right]B_i^{d-2}(u)$$

$$P(u)=\sum_{i=0}^{d-2} P_i^2 B_i^{d-2}(u)$$

.....

→ $P(u)=P_0^d B_0^0(u)\ ,\ B_0^0(u)\equiv 1$ QED

Bézier curves

- Evaluation with Horner's method
  - We rewrite the polynomials to a monomial form

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

$$a_i=\binom{d}{i}\sum_{j=0}^{i}(-1)^{i-j}\binom{i}{j}P_j$$

$$(a_i)=M_h(P_i)$$

$$M_h=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -d & d & 0 & \cdots & 0 \\ \frac{d(d-1)}{2} & -d(d-1) & \frac{d(d-1)}{2} & \cdots & 0 \\ \vdots & & & & \vdots \\ -1^{d-1}d & d(d-1) & \cdots & \cdots & 0 \\ -1^{d} & -1^{d-1}d & -1^{d-2}\frac{d(d-1)}{2} & \cdots & 1 \end{pmatrix}$$

Bézier curves

- Complexity of Horner's method :
  - $3d$ multiplications + $3d$ additions ... without taking into account the change of polynomial basis (to be done only once) $(a_i) = M_h (P_i)$
  - But : change of the internal representation
    - Some authors have shown that numerical errors introduced during the change of internal representation are substantial.

Bézier curves

- Method based on Bernstein polynomials
  - We factor the terms $(1-u)^{d-i}$ in $B_i^d(u) = \binom{d}{i} u^i (1-u)^{d-i}$

$$P(u) = \sum_{i=0}^{d} P_i B_i^d(u) = (1-u)^d \sum_{i=0}^{d} P_i \binom{d}{i} \left(\frac{u}{1-u}\right)^i$$

  - This is a monomial form that can be evaluated with Horner's method – without any change of the internal representation (points P_i)
  - But beware when $u \to 1$!!!!
    - We'd rather factor rather the terms u^i, That gives :

$$P(u) = \sum_{i=0}^{d} P_i B_i^d(u) = u^d \sum_{i=0}^{d} P_i \binom{d}{i} \left(\frac{1-u}{u}\right)^{d-i}$$

Bézier curves

- To summarize :

$$P(u)=\begin{cases}(1-u)^d \sum_{i=0}^{d} P_i \binom{d}{i} \left(\frac{u}{1-u}\right)^i & , 0<u\leq\frac{1}{2} \\ u^d \sum_{i=0}^{d} P_i \binom{d}{i} \left(\frac{1-u}{u}\right)^{d-i} & , \frac{1}{2}<u\leq 1\end{cases}$$

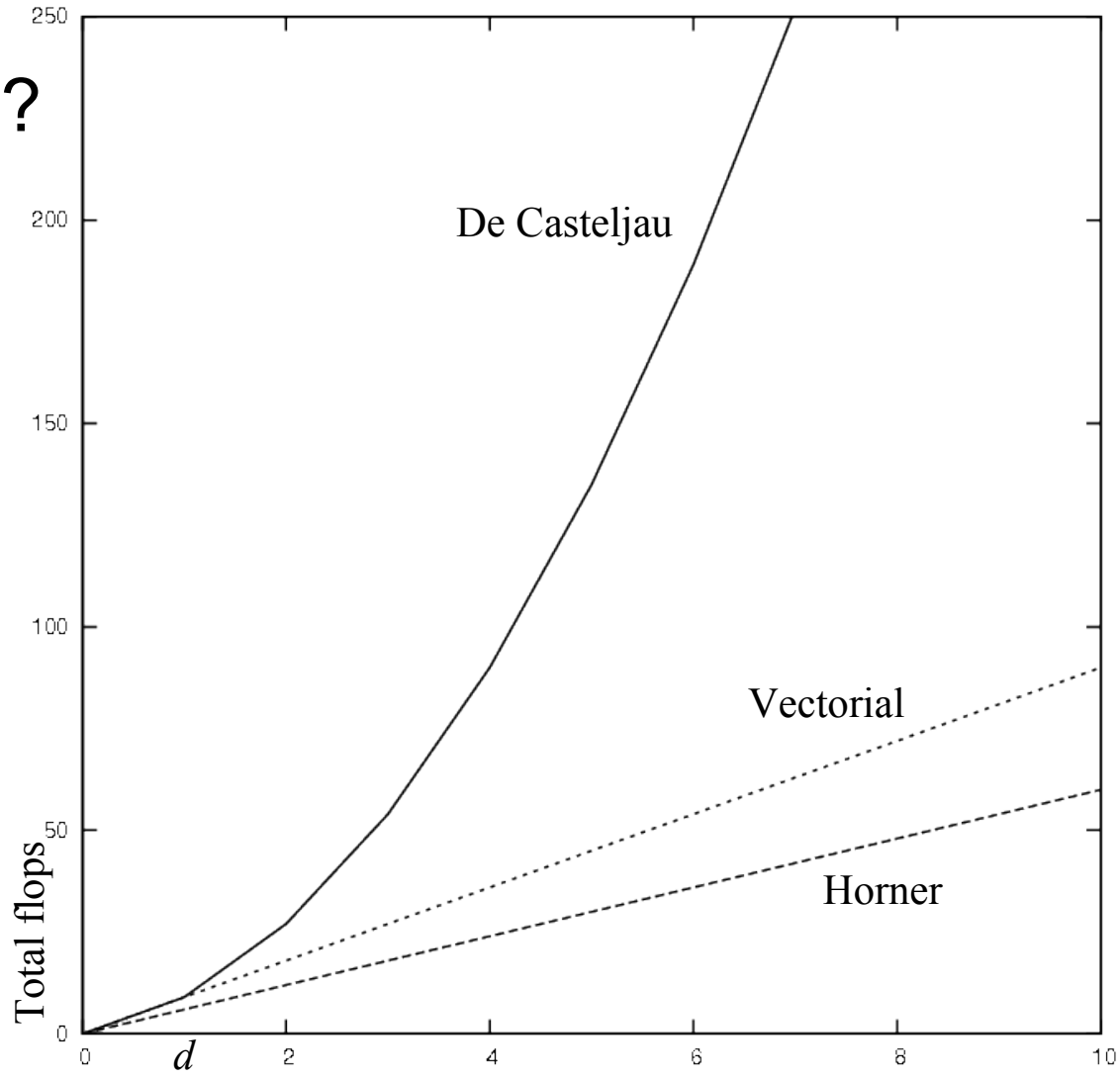
$$P(u)=(1-u)^d \begin{pmatrix} P_0 & P_1 & \cdots & P_d \end{pmatrix} \begin{pmatrix} \binom{d}{0} \\ \binom{d}{1}\frac{u}{1-u} \\ \binom{d}{2}\left(\frac{u}{1-u}\right)^2 \\ \vdots \\ \binom{d}{d}\left(\frac{u}{1-u}\right)^d \end{pmatrix}$$

Bézier curves

- Algorithmic complexity of the vectorial method
  - Requires $6d$ multiplications and $3d$ additions.
  - No change in the internal representation

Bézier curves

- Choice of algorithm ?
  - Robustness
    - 1 De Casteljau
    - 2 Vectorial
    - 3 Horner
  - Speed
    - 1 Horner
    - 2 Vectorial
    - 3 De Casteljau *

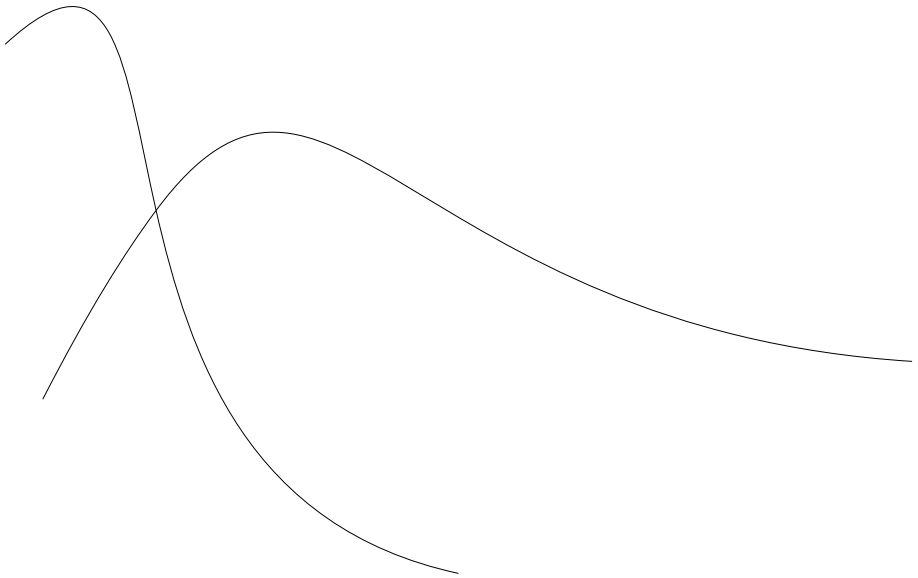


Bézier curves

- In practice
  - The vectorial algorithm is often used to quickly compute points for display purposes
  - De Casteljau's algorithm is used for increased robustness and ...
    - It allows to obtain derivatives of curve (see later...)
    - It allows interesting geometrical operations (see later …)
    - If many points are to be computed, it may actually perform well (see later !)

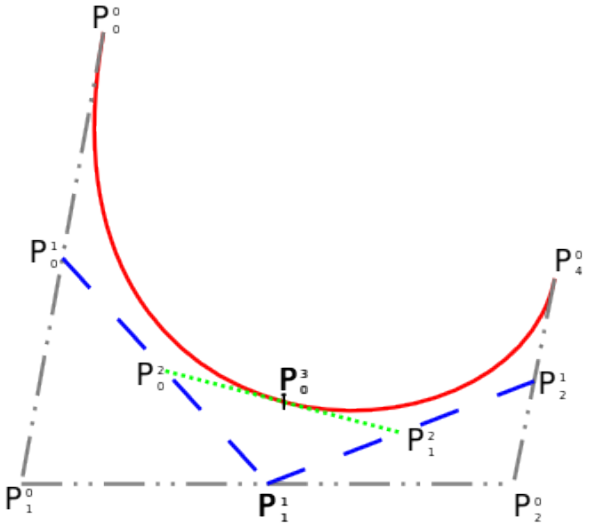
Bézier curves

- 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



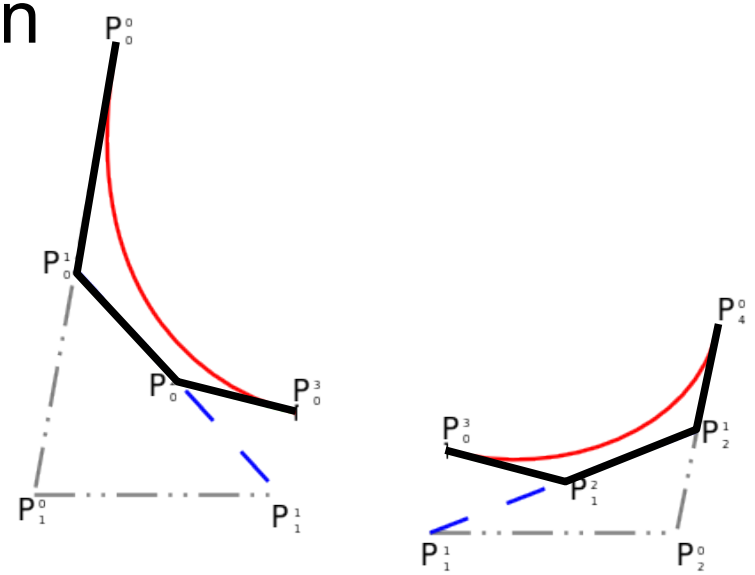
Bézier curves

- Let us start from De Casteljau's geometrical construction



Bézier curves

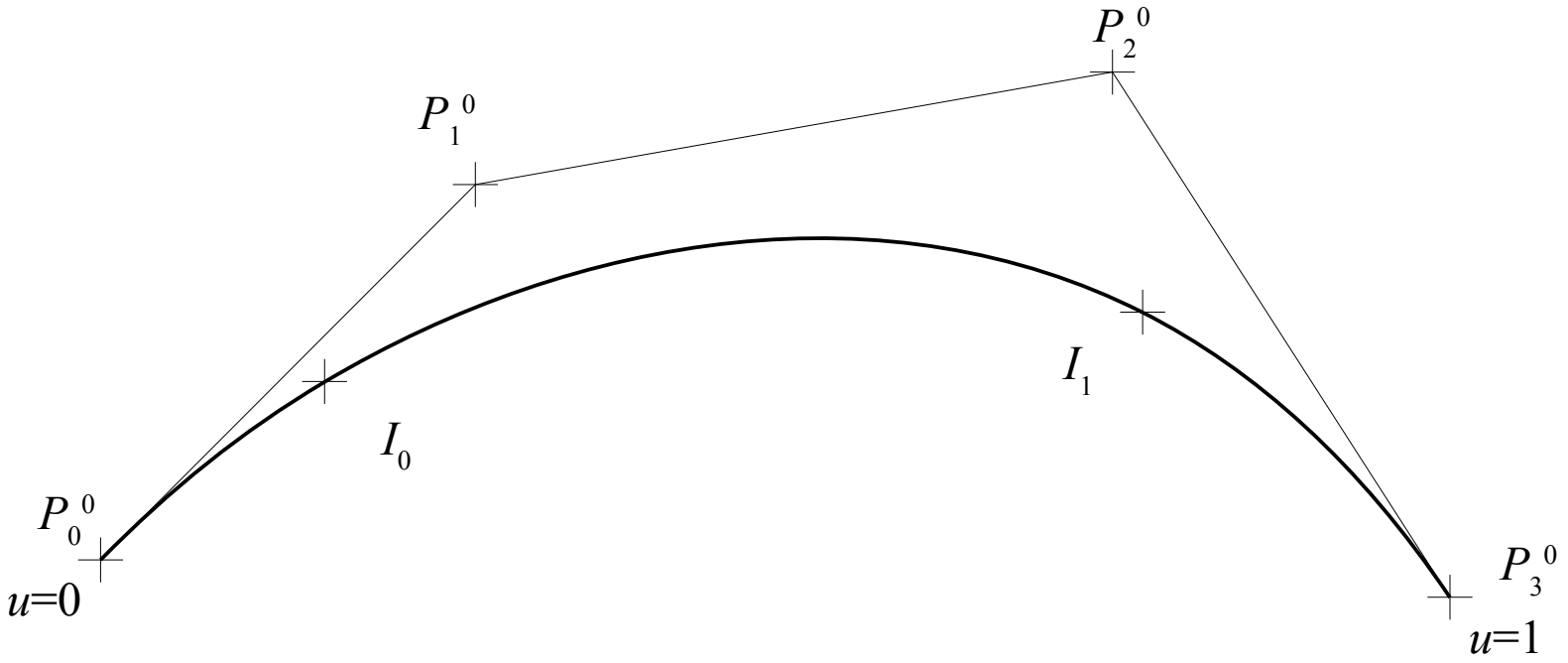
- Let us start from De Casteljau's geometrical construction



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

Bézier curves

- Curve to trim



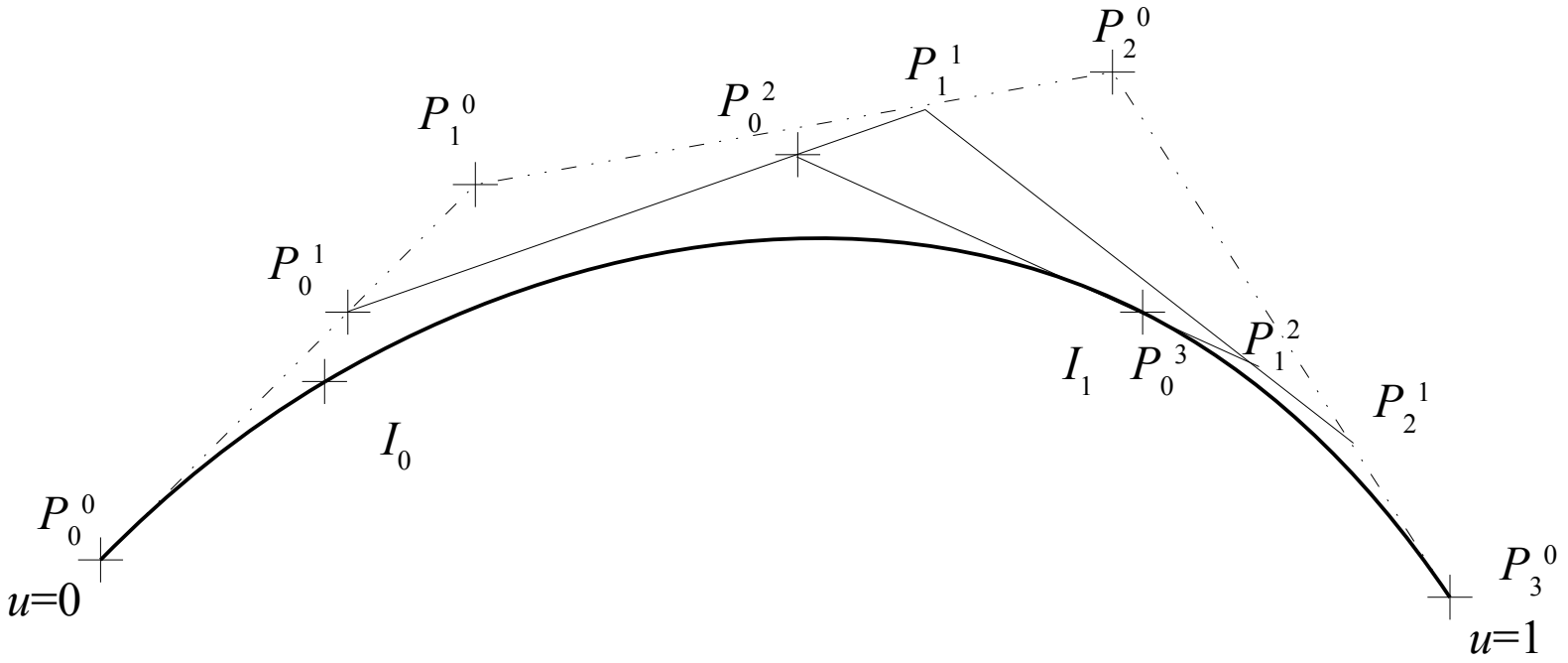
Bézier curves

- For a curve that we want to trim :

  1 – Compute the intersection point I_1 – at $u=u_1$ with help of De Casteljau's algorithm – this gives the points P_i^j

Bézier curves

1 – Compute the intersection point I_1 – at $u=u_1$ with help of De Casteljau's algorithm – this gives the points P_i^j



Bézier curves

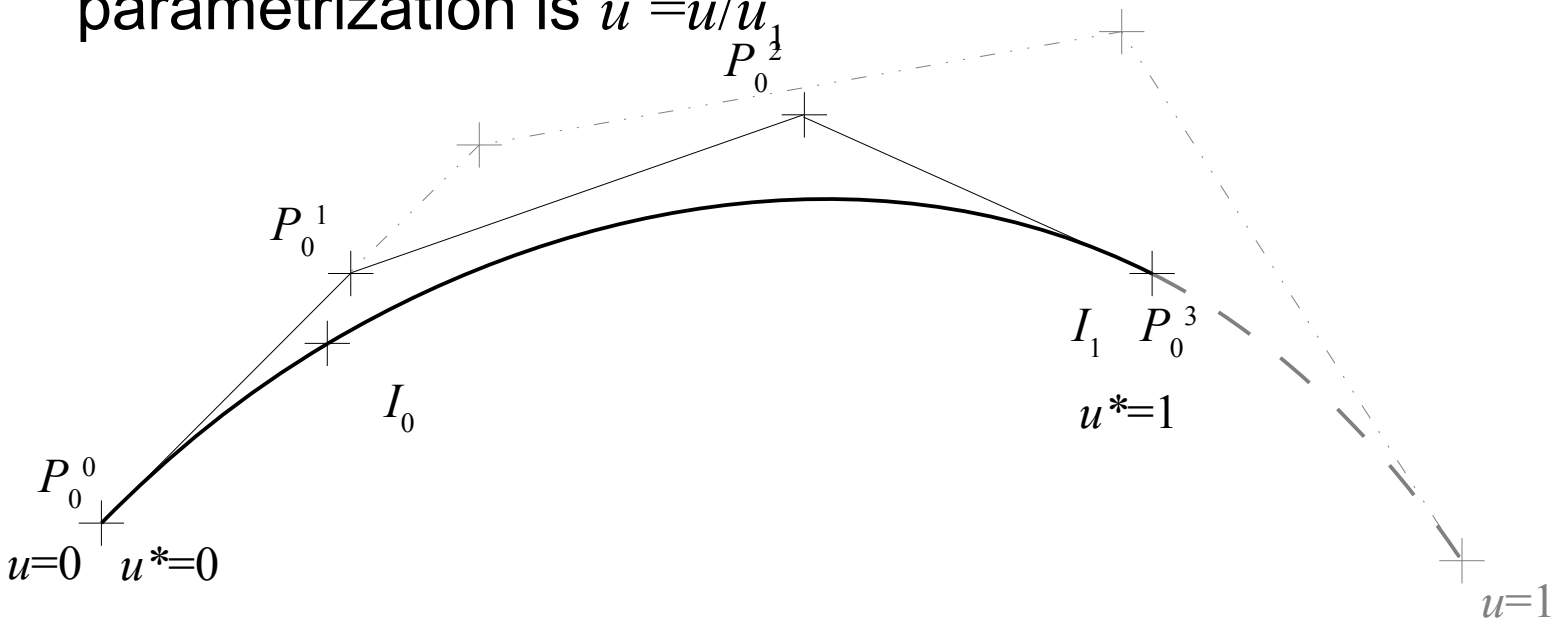
- For a curve that we want to limit :

  1 – Compute the intersection point I_1 – at $u=u_1$ with help of De Casteljau's algorithm – this gives the points P_i^j

  2 – Among these points, consider the points P_0^j : they are vertices of the characteristic polygon of the curve's restriction at the interval P_0-I_1 , and the new parametrization is $u^*=u/u_1$

Bézier curves

- 2 – Among these points, consider the points P_0^j : they are vertices of the characteristic polygon of the curve's restriction at the interval P_0-I_1 , and the new parametrization is $u^*=u/u_1$



Bézier curves

- For a curve that we want to limit :

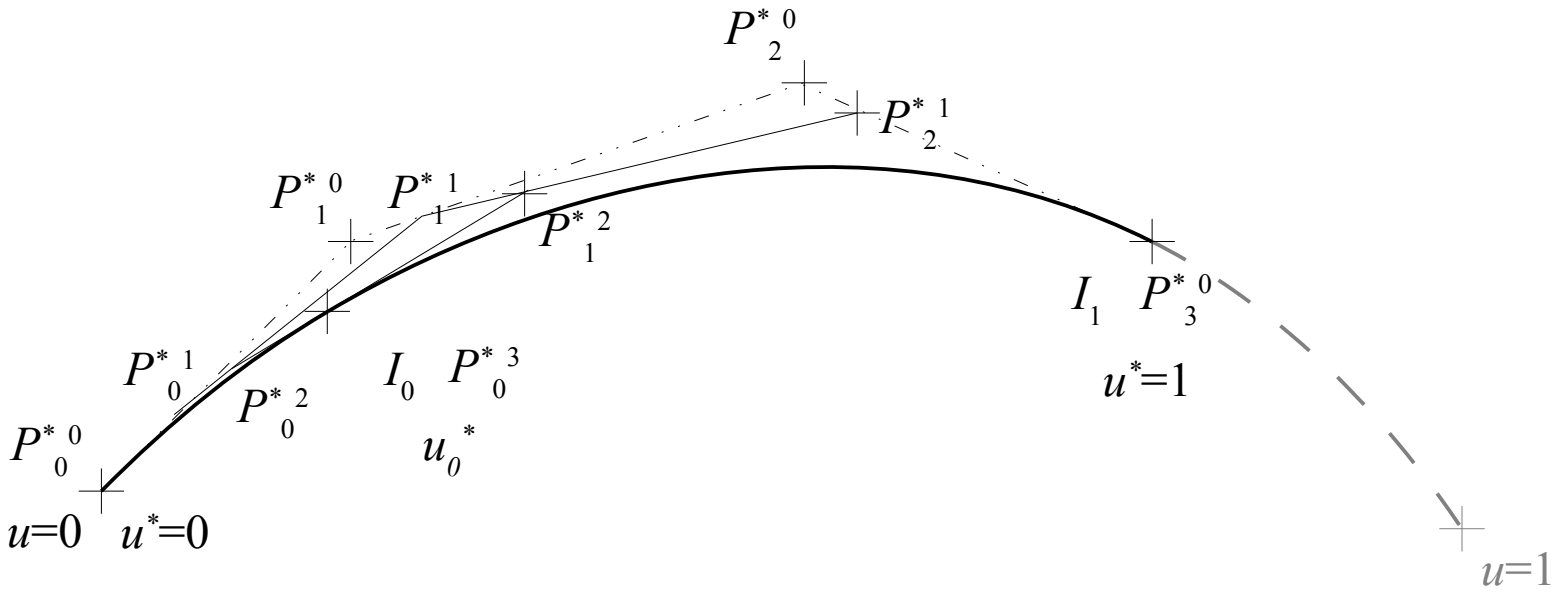
  1 – Compute the intersection point I_1 – at $u=u_1$ with help of De Casteljau's algorithm – this gives the points P_i^j

  2 – Among these points, consider the points P_0^j : they are vertices of the characteristic polygon of the curve's restriction at the interval P_0-I_1 , and the new parametrization is $u^*=u/u_1$

  3 – Calculate the intersection I_0 on the new curve – at $u^*=u^*_0$ - gives the points P^{*j}_i

Bézier curves

3 – Calculate the intersection I_0 on the new curve – at $u^*=u^*_0$ - gives the points $P^{*\,j}_{i}$



Bézier curves

- For a curve that we want to limit :

  1 – Compute the intersection point I_1 – at $u=u_1$ with help of De Casteljau's algorithm – this gives the points P_i^j

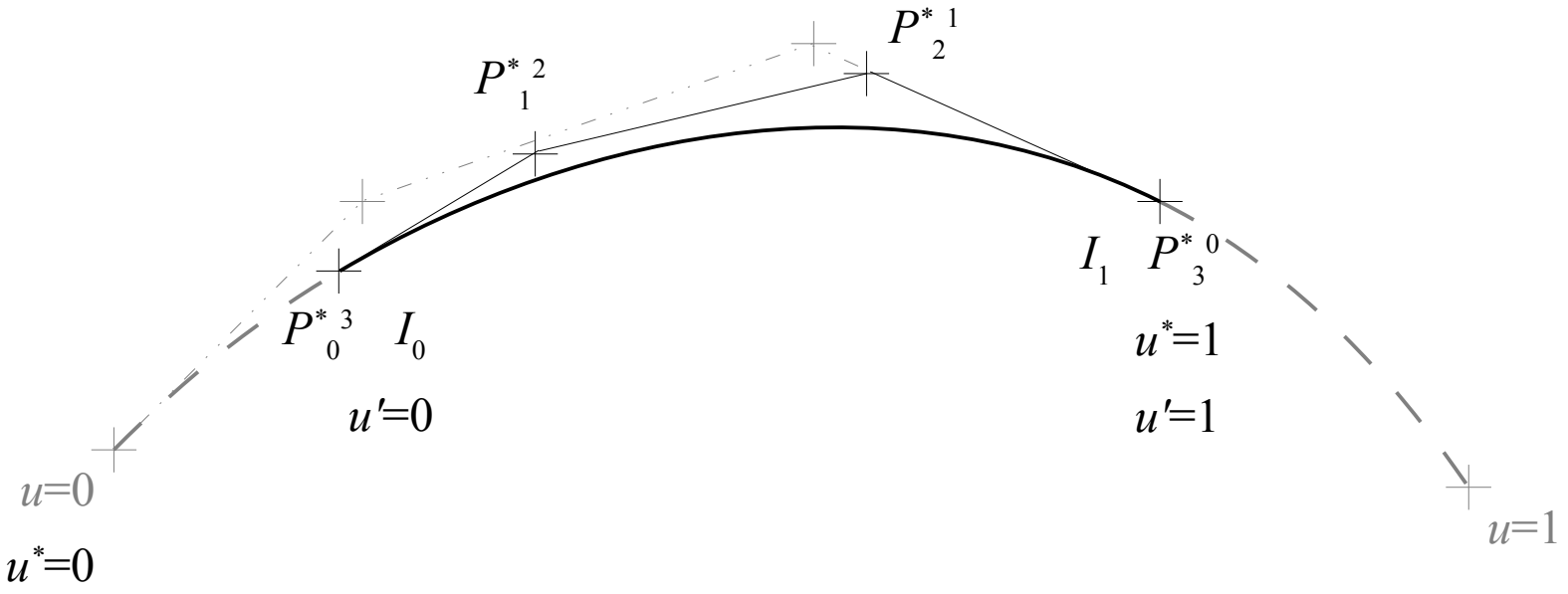
  2 – Among these points, consider the points P_0^j : they are vertices of the characteristic polygon of the curve's restriction at the interval P_0-I_1 , and the new parametrization is $u^*=u/u_1$

  3 – Calculate the intersection I_0 on the new curve – at $u^*=u^*_0$ - gives the points P^{*j}_i

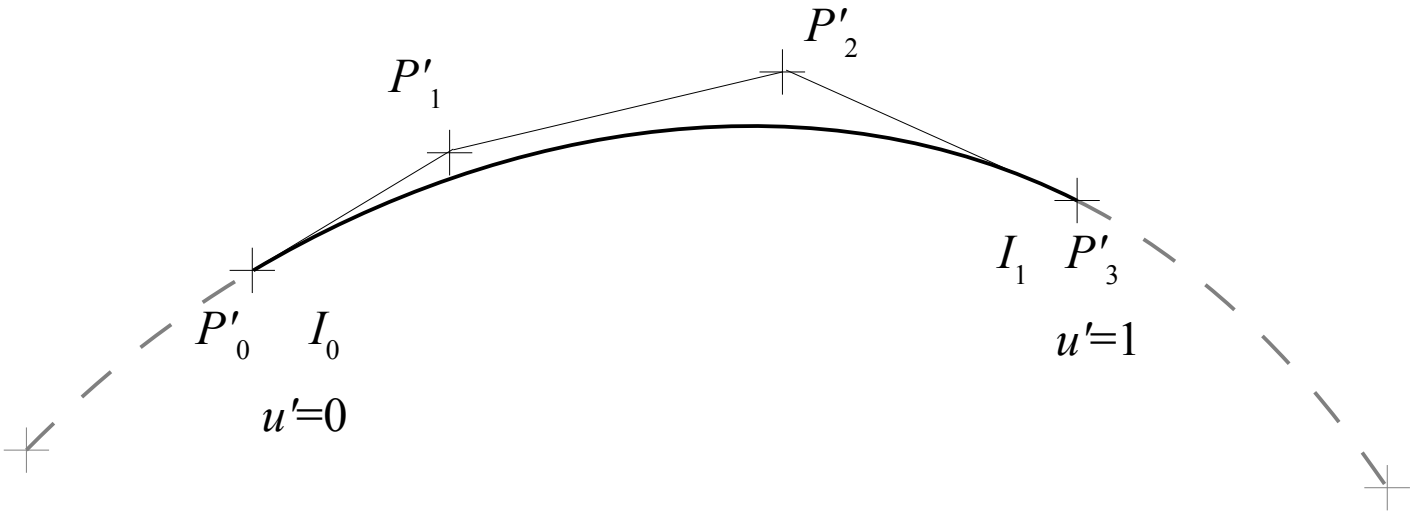
  4 – Consider the points P^{*d}_i : vertices of the characteristic polygon of the curve's restriction to the interval I_0-I_1 : new parametrization is $u'=(u^*-u^*_0)/(1-u^*_0)$

Bézier curves

4 – Consider the points $P^{*\,d-i}_{i}$: vertices of the characteristic polygon of the curve's restriction to the interval I_0-I_1 : new parametrization is $u'=(u^*-u^*_0)/(1-u^*_0)$



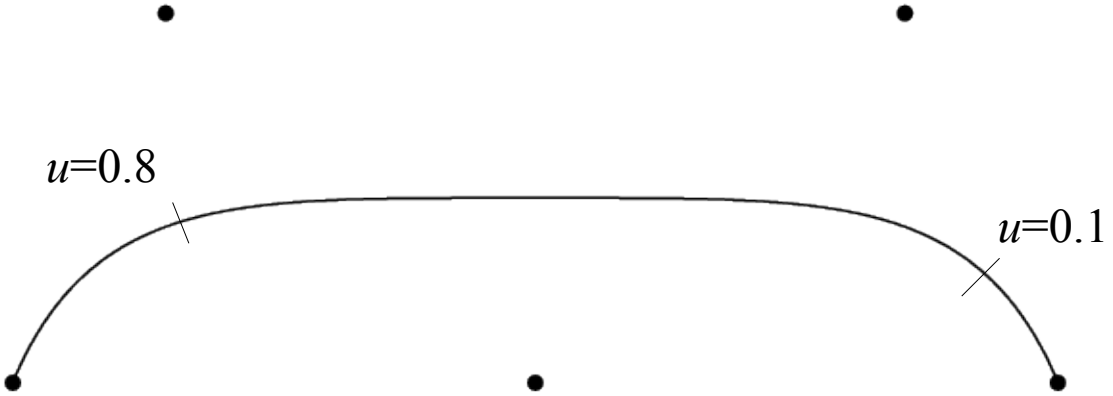
Bézier curves



Bézier curves

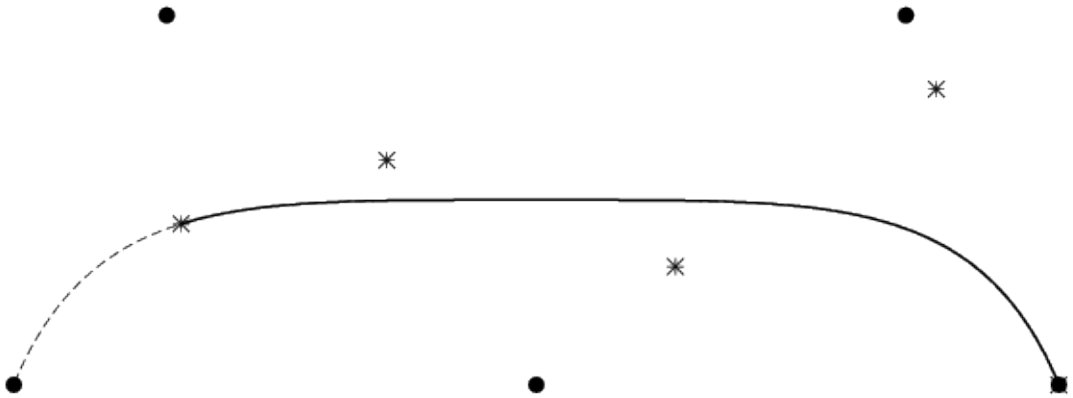
- With the same algorithm, we can increase the parametric domain of a curve
  - Intersection with objects close but not touching the curve's extremities
  - Beware : Bézier curves are variation diminishing and convex combinations only when $0 \leq u \leq 1$...

Bézier curves



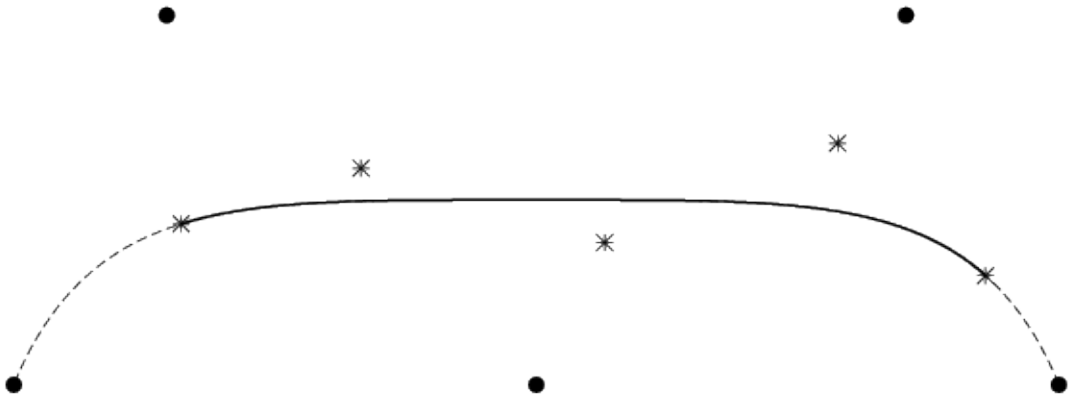
Desired cutting points

Bézier curves



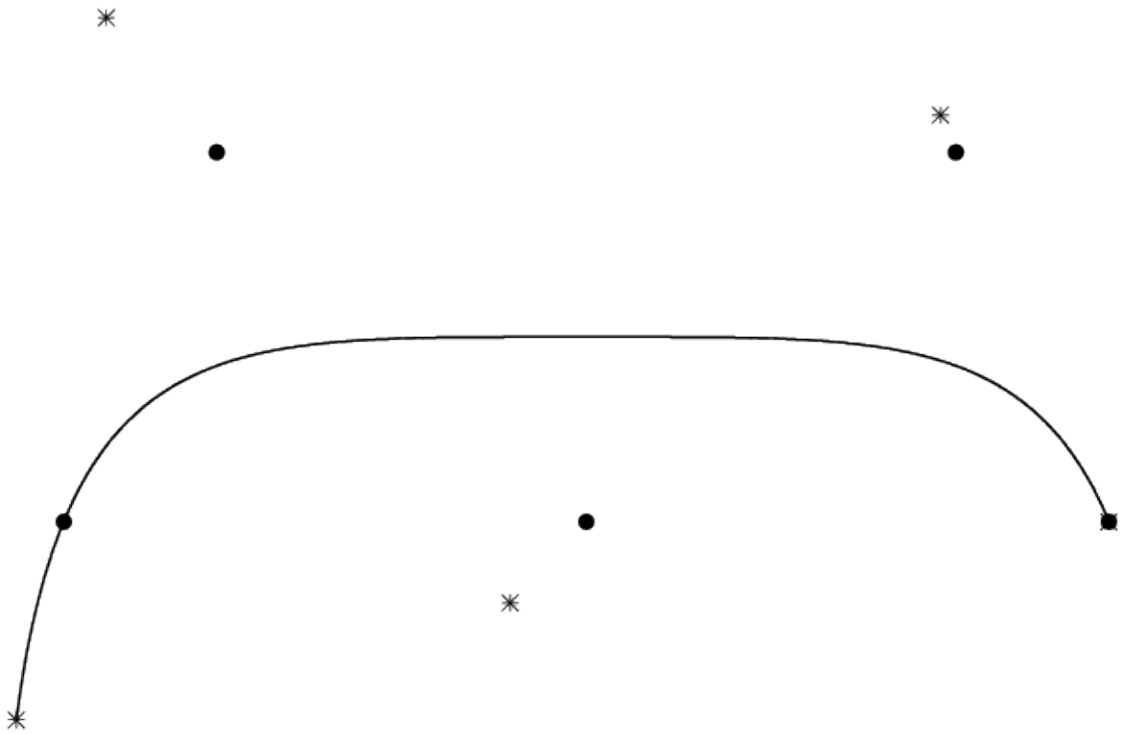
Cutting for u=0.8

Bézier curves



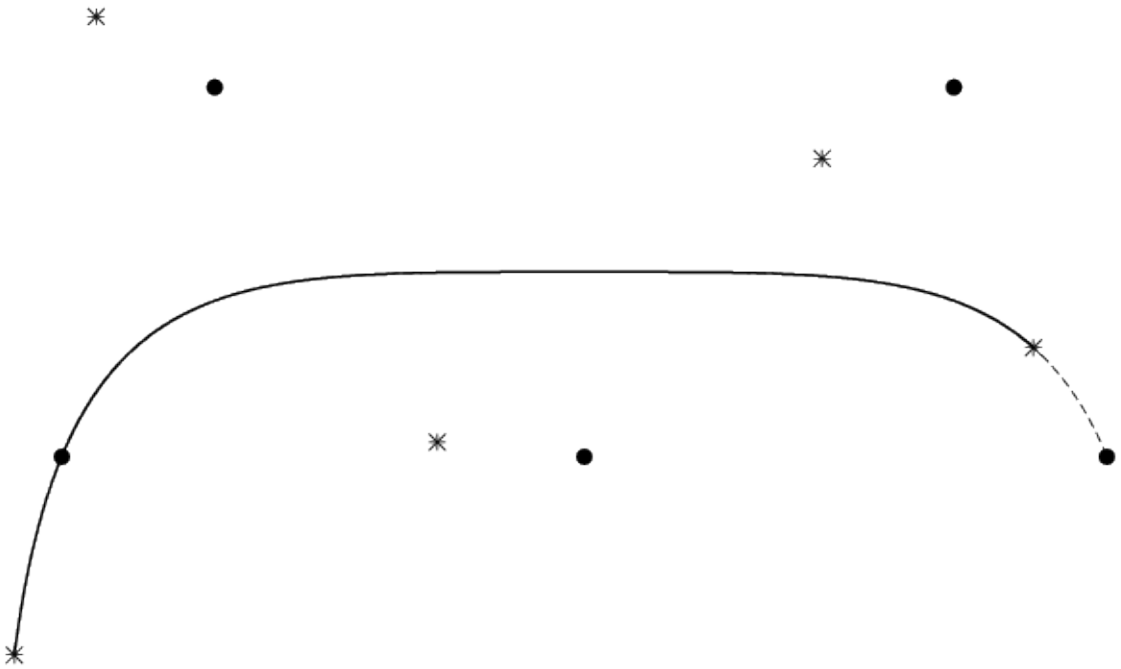
Cutting for u=0.8...and for u=0.1 (at u*=0.1/0.8)

Bézier curves



Extension for u=1.1

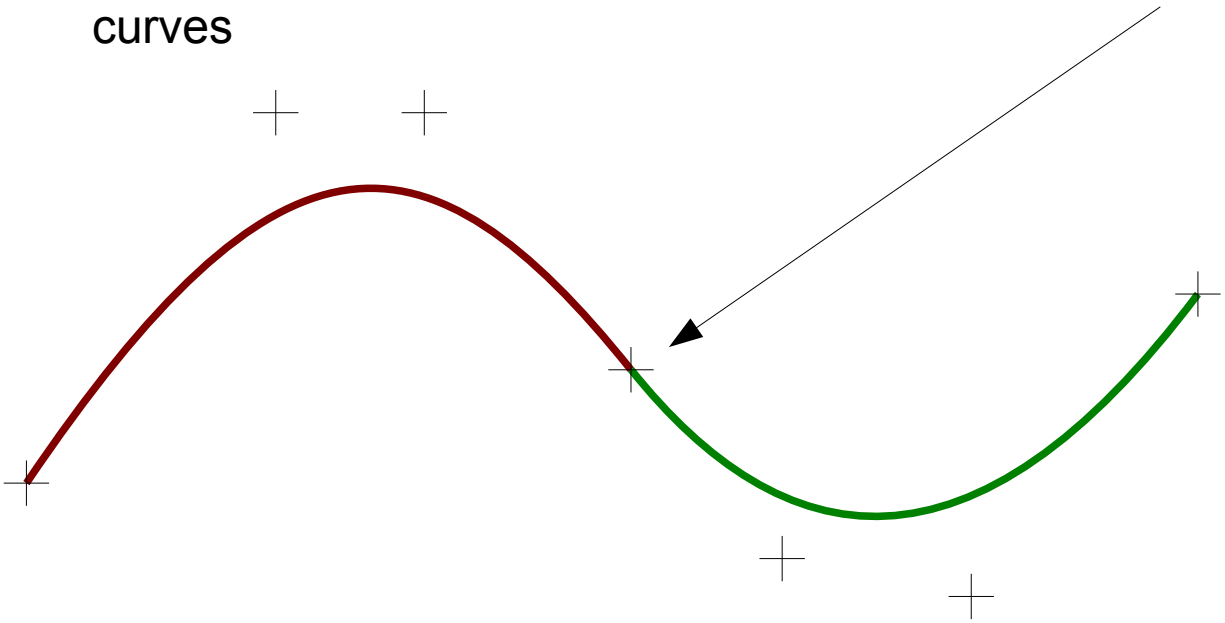
Bézier curves



Extension for u=1.1 and cutting for u=0.1 - at (new parameter) u^*=0.1/1.1 ...

Bézier curves

- Curves defined by pieces
  - Bézier curves do have a global control
  - If we need local control, we have to assemble several of them
    - We have to impose some continuity at the interface points between curves



Bézier curves

- Expression of the derivatives of a Bézier curve

$$\frac{dP}{du}(u)=\sum_{i=0}^{d} P_i B_i^{'d}(u) \quad \text{with} \quad B_i^{'d}=d\left(B_{i-1}^{d-1}(u)-B_i^{d-1}(u)\right)$$

$$\frac{dP}{du}(u)=d\sum_{i=0}^{d} P_i\left(B_{i-1}^{d-1}(u)-B_i^{d-1}(u)\right)=d\Big(P_0\underbrace{(B_{-1}^{d-1}-B_0^{d-1})}_{=0}$$
$$+P_1\underbrace{(B_0^{d-1}-B_1^{d-1})}$$
$$+P_2\underbrace{(B_1^{d-1}-B_2^{d-1})}$$
$$+\quad\ldots$$
$$+P_d\underbrace{(B_{d-1}^{d-1}-B_d^{d-1})}_{=0}\Big)$$

- By factoring $B_i^{d-1}(u)$:

$$\frac{dP}{du}(u)=d\sum_{i=0}^{d-1}\left(P_{i+1}-P_i\right)B_i^{d-1}(u)$$

$$P_i^{'}=P_{i+1}-P_i$$

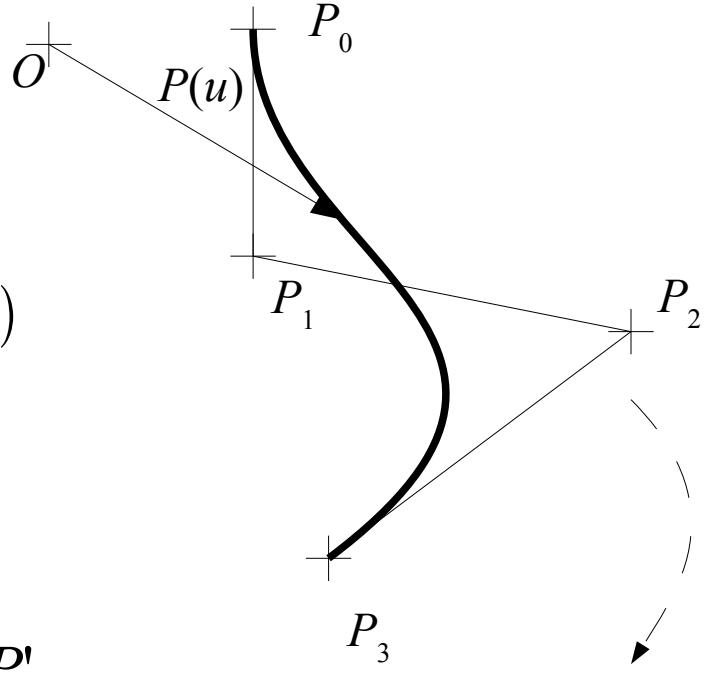
Bézier curves

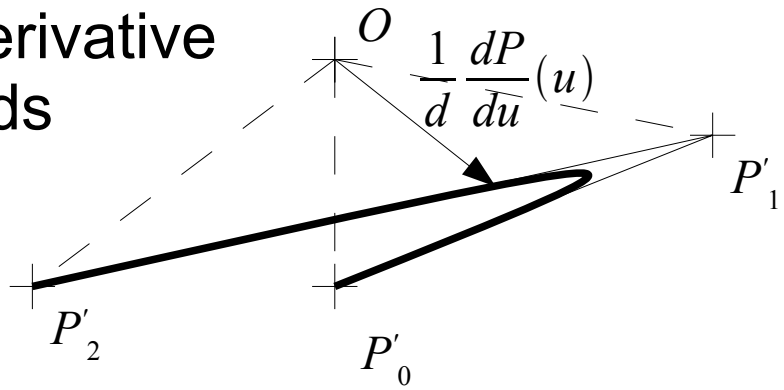
- First derivative

$$\frac{dP}{du}(u) = d \sum_{i=0}^{d-1} (P_{i+1} - P_i) B_i^{d-1}(u)$$

$$\frac{dP}{du}(u) = d \sum_{i=0}^{d-1} P'_i B_i^{d-1}(u)$$

  - The control points P'_0 and P'_{d-1} are interpolated so the first derivative at the extremities only depends on the **two first** (resp. last) control points





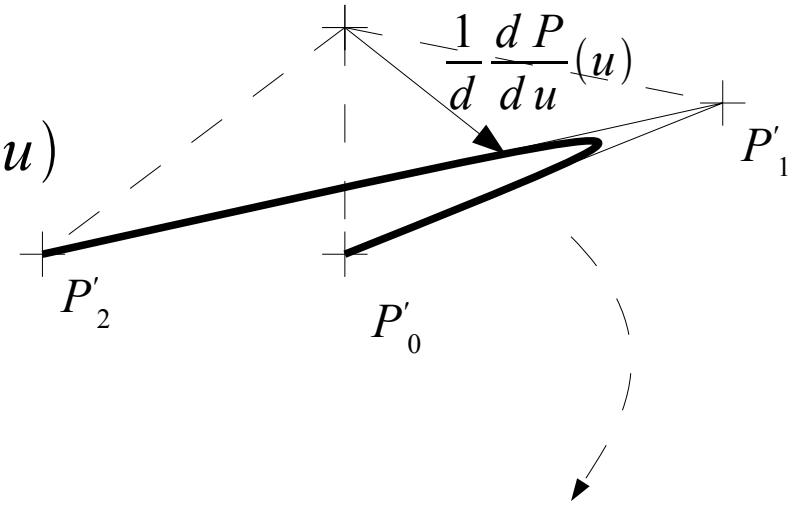
First hodograph

Bézier curves

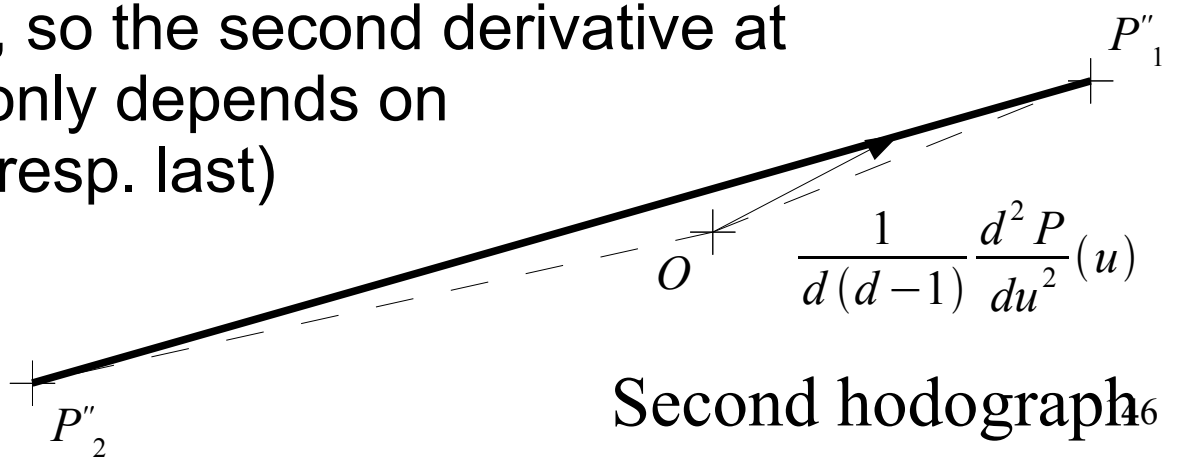
- Second derivative = derivative of the derivative

$$\frac{d^2P}{du^2}(u) = (d-1)\,d \sum_{i=0}^{d-2} (P'_{i+1} - P'_i)\, B_i^{d-2}(u)$$

$$\frac{d^2P}{du^2}(u) = (d-1)\,d \sum_{i=0}^{d-2} P''_i\, B_i^{d-2}(u)$$



- The control points P''_0 and P''_{d-2}...
  are interpolated, so the second derivative at the extremities only depends on **the three first** (resp. last) control points



Second hodograph

Bézier curves

- Derivative of order k

$$\frac{d^k P}{du^k}(u) = (d-k+1)\cdots(d-1)\,d \sum_{i=0}^{d-k} \left(P_{i+1}^{(\text{k-1})} - P_i^{(\text{k-1})}\right) B_i^{d-k}(u)$$

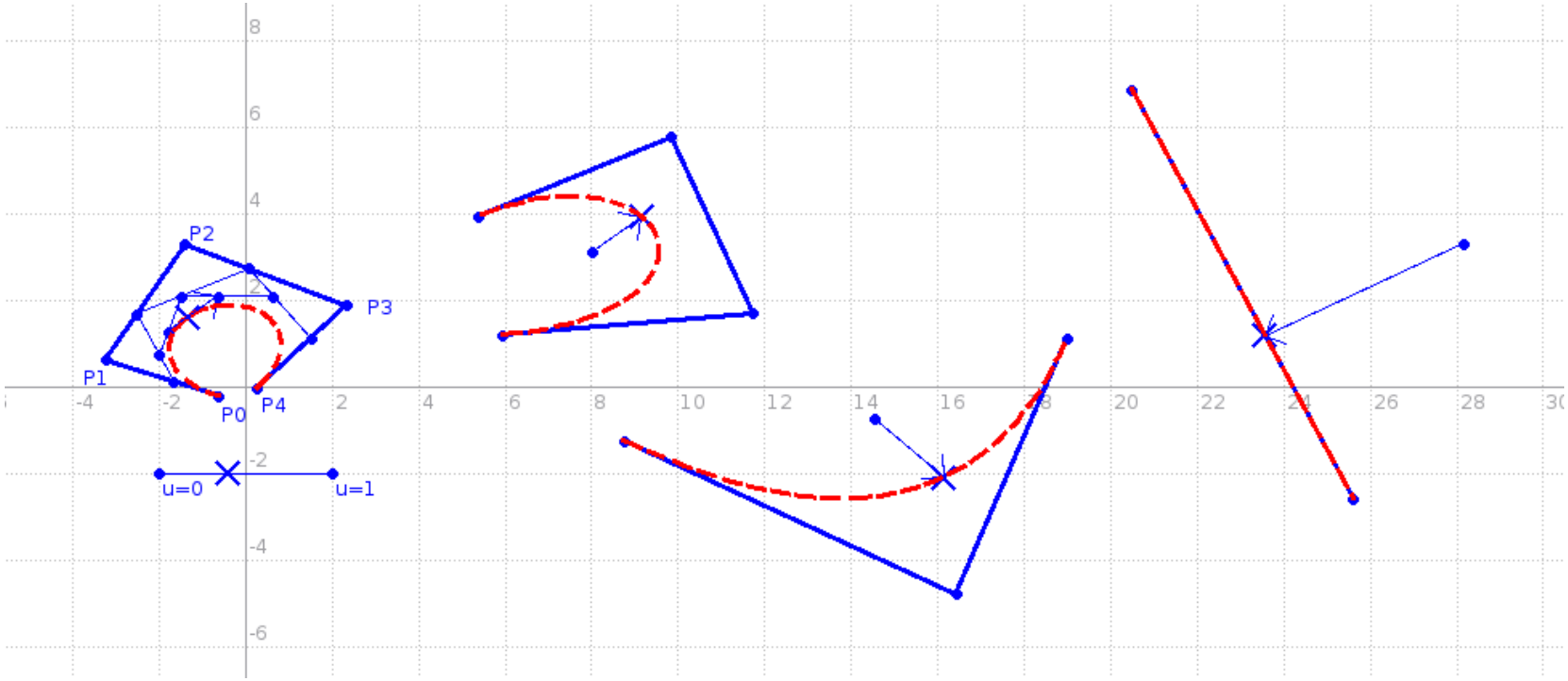
$$\frac{d^k P}{du^k}(u) = \prod_{l=1}^{k}(d-l+1) \sum_{i=0}^{d-k} P_i^{(\text{k})} B_i^{d-k}(u) \quad , \quad P_i^{(\text{k})} = \left(P_{i+1}^{(\text{k-1})} - P_i^{(\text{k-1})}\right)$$

$$= \left(P_{i+2}^{(\text{k-2})} - 2\,P_{i+1}^{(\text{k-2})} + P_i^{(\text{k-2})}\right)$$

$$= \sum_{j=0}^{k} (-1)^j \binom{d}{j} P_{j+i}^{(0)}$$

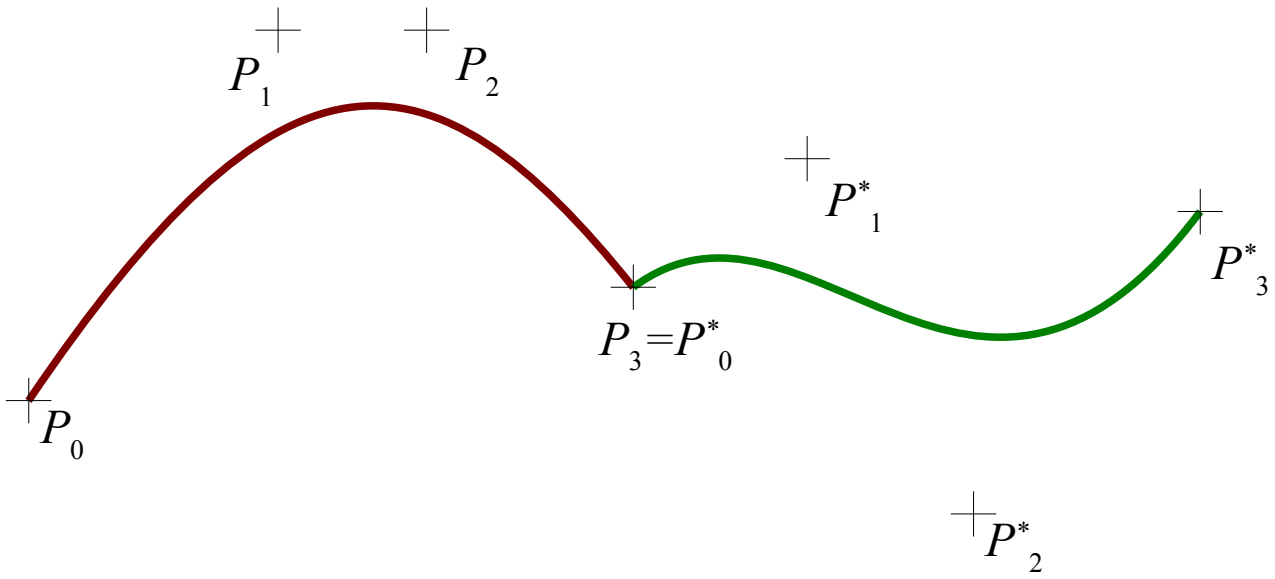
- The derivative of order k at the extremities only depends on the $k+1$ **first** (resp. last) control points.
- Of course, there must be enough control points ... ($k < d+1$)

Bézier curves



Bézier curves

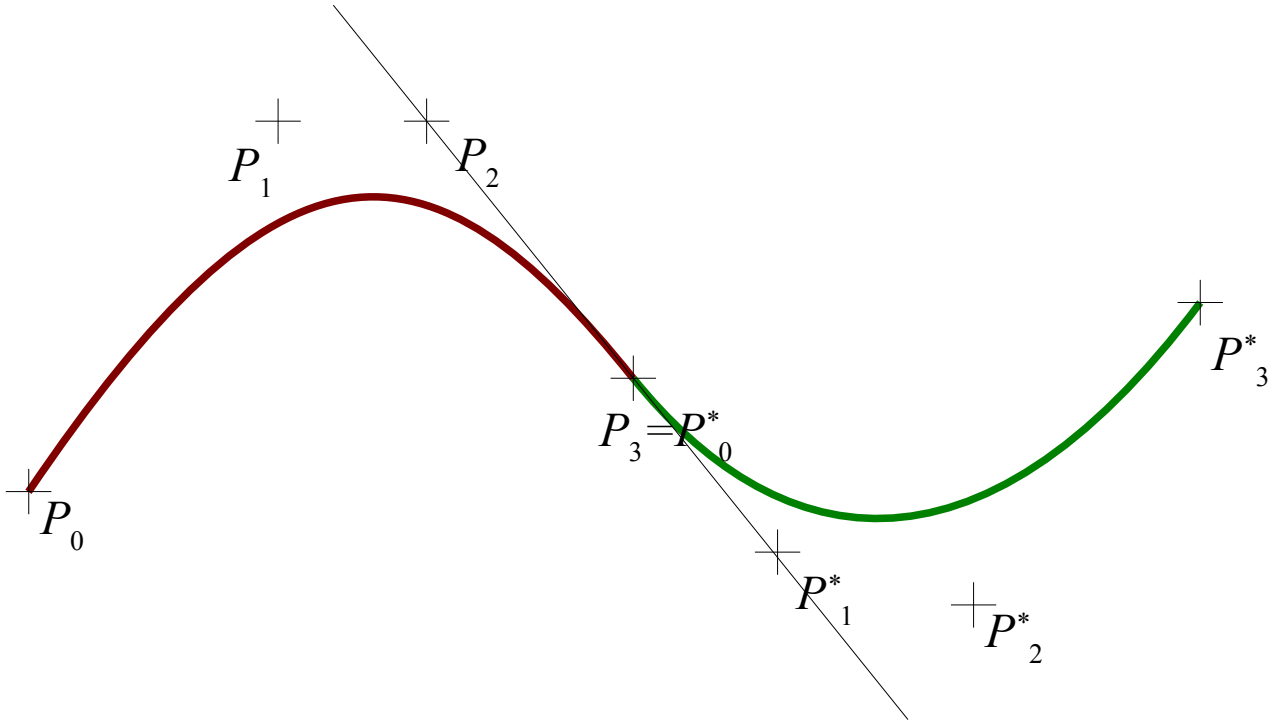
- Connecting two curves is the same as imposing constraints on the control points on both sides of the « sticking » point
  - We assume that the curves are regular
- G_0 continuity (positions) (same as C_0 continuity)



Bézier curves

- G_1 continuity (C_1 continuity is more strict)
- Minimum degree : 2

$$P_{d-1}P_d = \alpha P_0^* P_1^* \text{ , } \alpha > 0 \quad (\alpha = 1 \text{ for a } C_1 \text{ continuity})$$



Bézier curves

- G_2 continuity (C_2 continuity is more strict)
  - Continuity of the osculatory plane's orientation

$$(P_d - P_{d-1}) \times (P_{d-1} - P_{d-2}) = \gamma (P_2^* - P_1^*) \times (P_1^* - P_0^*)$$

  - Continuity of the curvature radius

$$R = \frac{\left\| \frac{dP}{du} \right\|^3}{\left\| \frac{d^2 P}{du^2} \times \frac{dP}{du} \right\|}$$

$$\frac{d \left\| P_{d-1} - P_{d-2} \right\|^3}{(d-1) \left\| (P_d - P_{d-1}) \times (P_{d-1} - P_{d-2}) \right\|} = \frac{d^* \left\| P_1^* - P_0^* \right\|^3}{(d^*-1) \left\| (P_2^* - P_1^*) \times (P_1^* - P_0^*) \right\|}$$

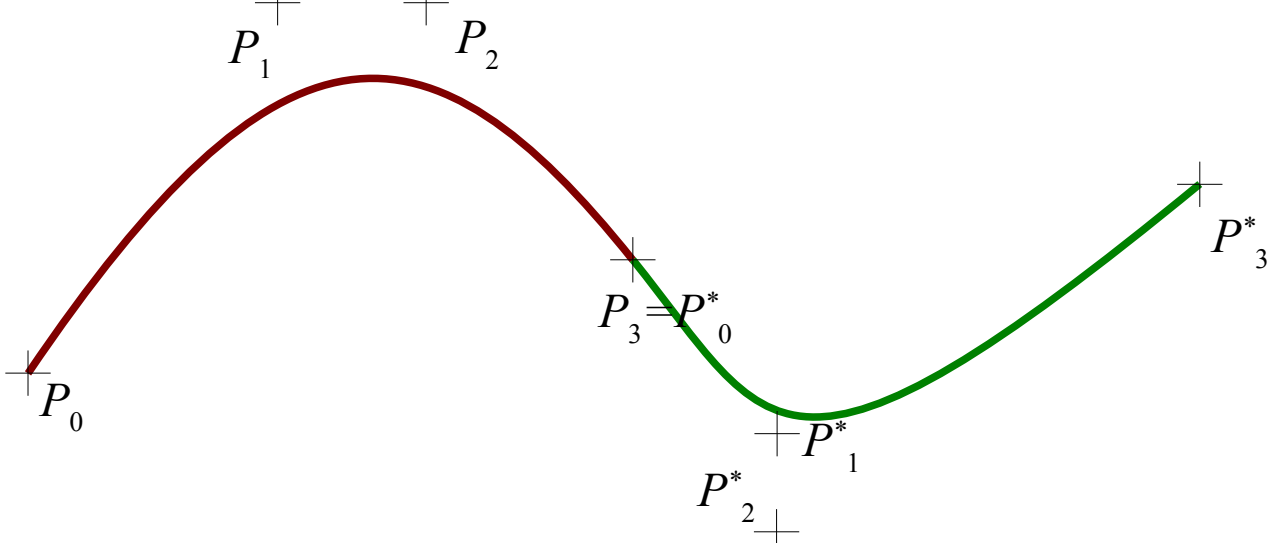
Bézier curves

- G_2 continuity (C_2 continuity is more strict)
  - Continuity of the osculatory plane's orientation

$$(P_d - P_{d-1}) \times (P_{d-1} - P_{d-2}) = \gamma (P_2^* - P_1^*) \times (P_1^* - P_0^*)$$

  - Continuity of the curvature radius

$$\frac{d \left\| P_{d-1} - P_{d-2} \right\|^3}{(d-1) \left\| (P_d - P_{d-1}) \times (P_{d-1} - P_{d-2}) \right\|} = \frac{d^* \left\| P_1^* - P_0^* \right\|^3}{(d^*-1) \left\| (P_2^* - P_1^*) \times (P_1^* - P_0^*) \right\|}$$



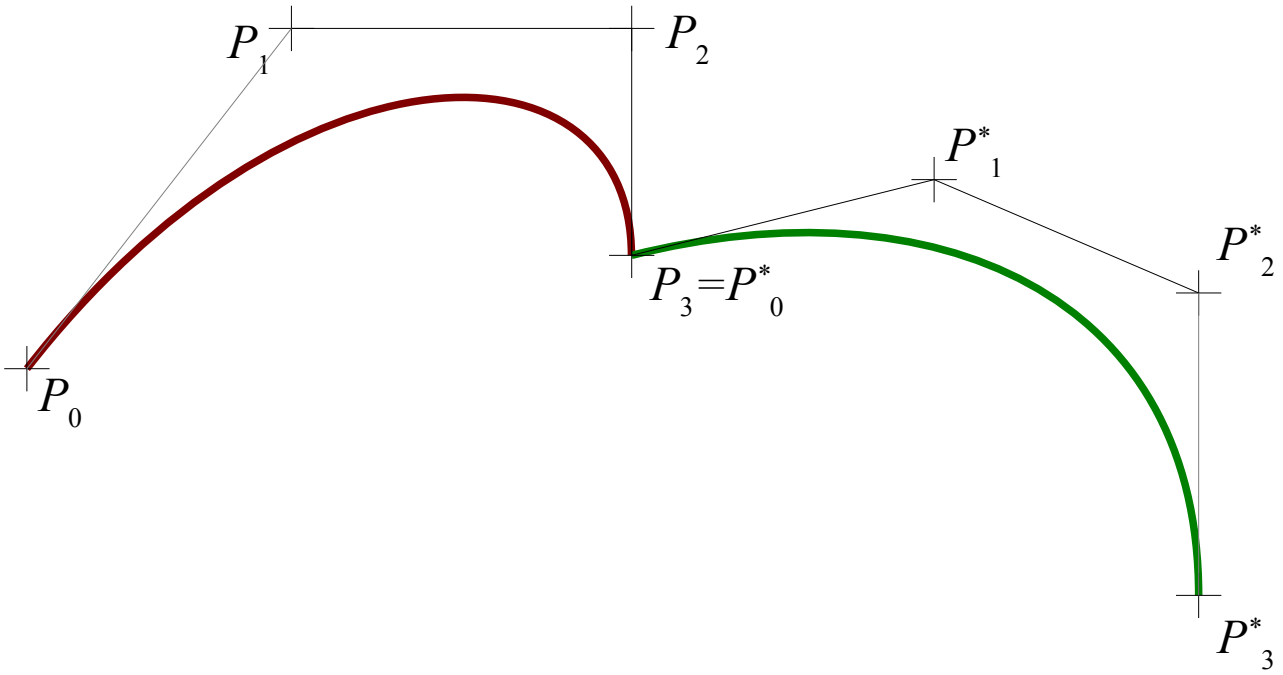
Bézier curves

- The G_k continuity with $k > 2$ in the general case is complex to impose
- The C_k continuity is easier to impose (simple expression of higher order derivatives)
  - Curve should be regular !
  - Same as imposing the continuity of functions $x(u)$, $y(u)$ and $z(u)$, independently of each other.

$$\frac{d^k P}{du^k}(u) = \prod_{l=1}^{k} (d-l+1) \sum_{i=0}^{d-k} P_i^{(k)} B_i^{d-k}(u)$$

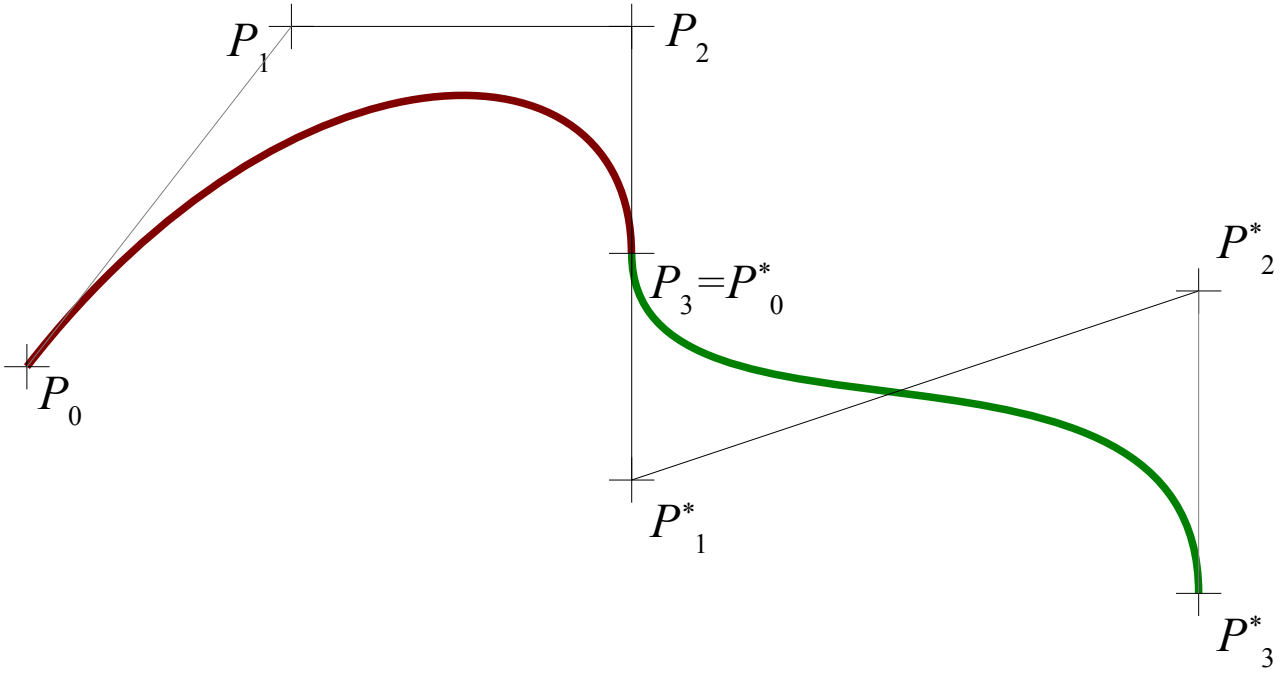
Bézier curves

- C_0 continuity $P_d = P^*_0$



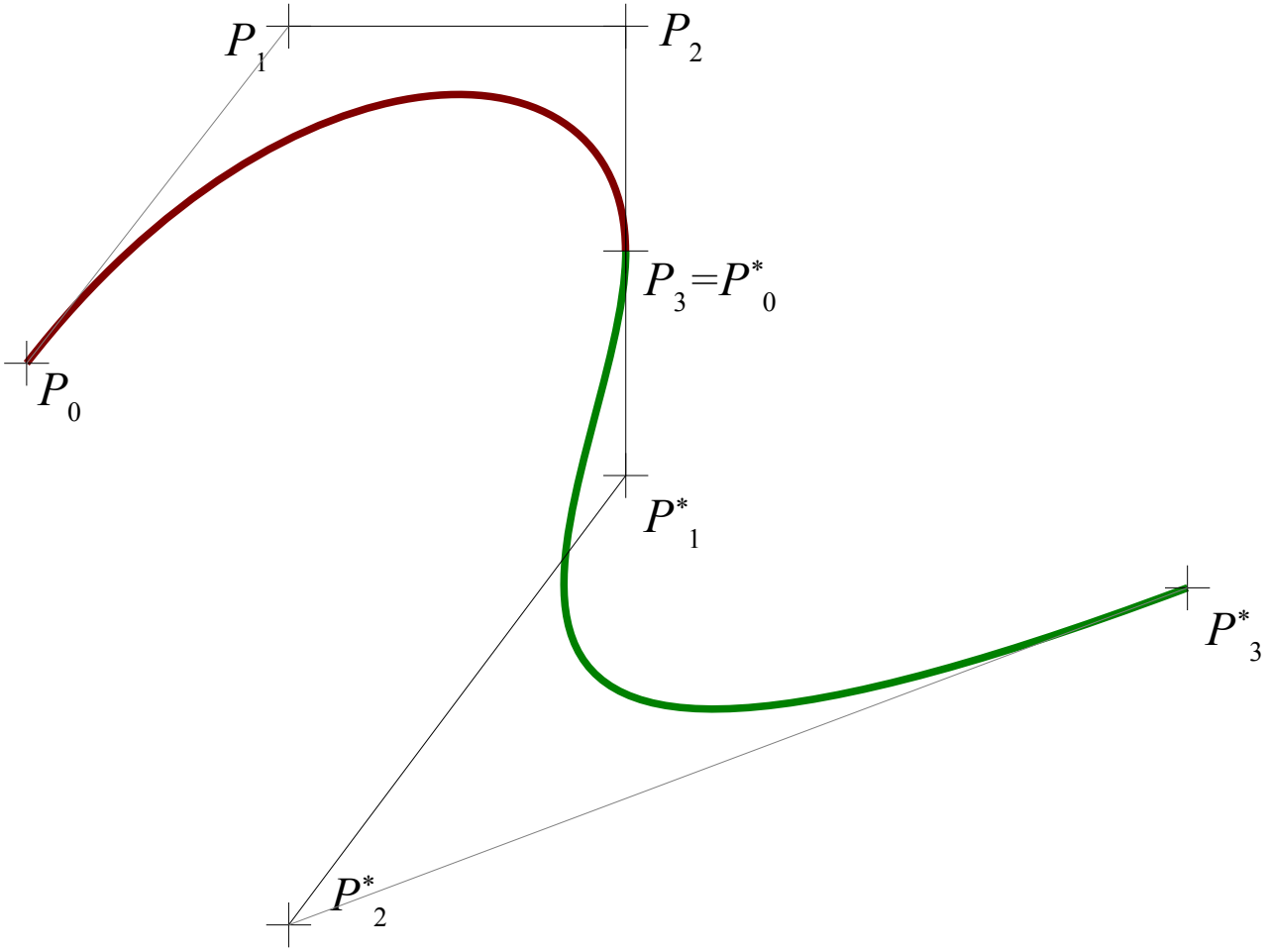
Bézier curves

- C_1 continuity $\quad P_1^* = P_d + (P_d - P_{d-1})$



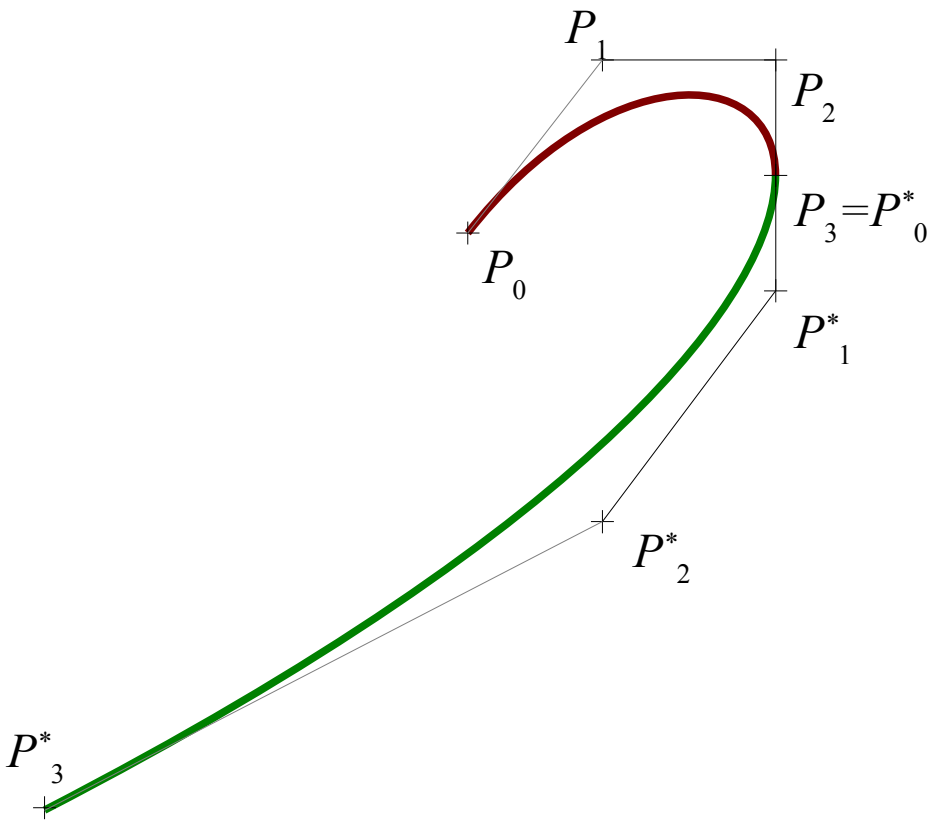
Bézier curves

- C_2 continuity $\quad P_2^* = P_{d-2} + 4(P_d - P_{d-1})$



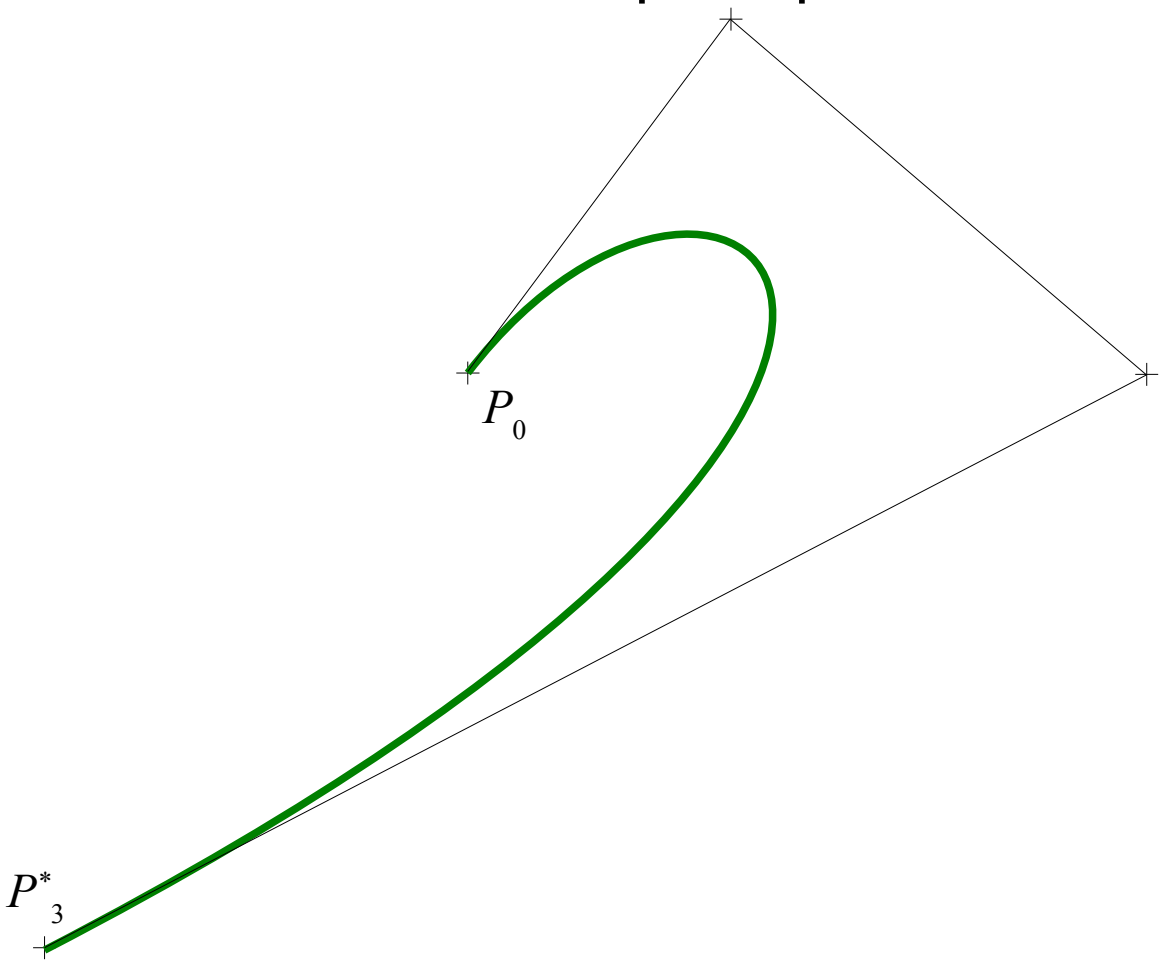
Bézier curves

- C_3 continuity $\quad P_3^* = 8P_d - 12P_{d-1} + 6P_{d-2} - P_{d-3}$



Bézier curves

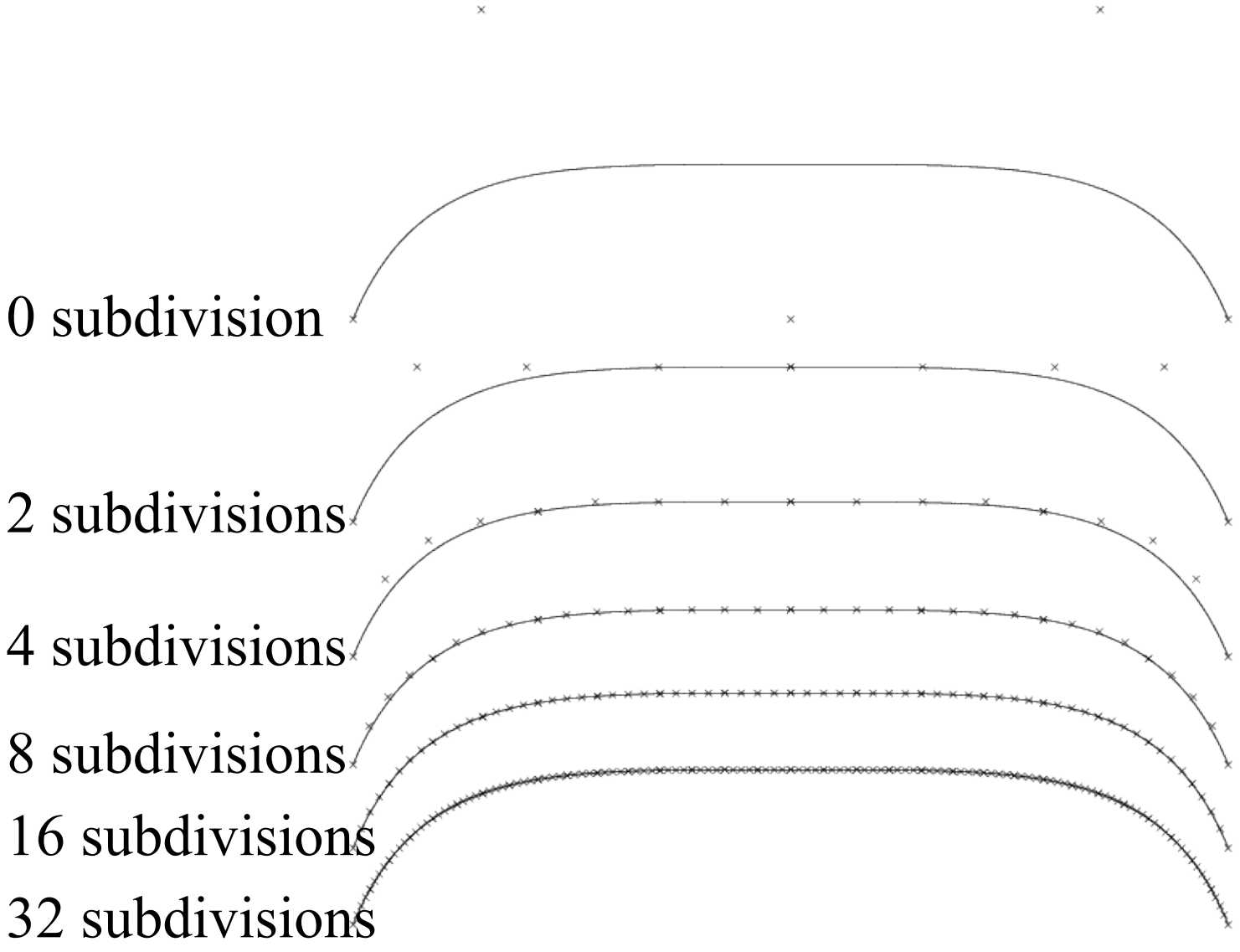
- The curve has now a unique representation of degree 3.



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.

Bézier curves



Bézier curves

- Cost of the recursive subdivision algorithm
  - $\mathrm{In}O(d^2 \cdot 2^m)$ for m levels of subdivision
  - Number of generated points: $d \cdot 2^m$
  - For each point that is generated, the algorithm becomes linear...
    - Competitive in comparison with Horner
    - It is not very accurate, nevertheless very robust.

B-Splines

B-Splines

- Three useful references :

  R. Bartels, J.C. Beatty, B. A. Barsky, An introduction to Splines for use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publications,1987

  JC.Léon, Modélisation et construction de surfaces pour la CFAO, Hermes, 1991

  L. Piegl, W. Tiller, The NURBS Book, Second Edition, Springer , 1996

B-Splines

- Isaac J. Schoenberg (1946)
- Carl De Boor (1972-76)
- Maurice G. Cox (1972)
- Richard Riesenfeld (1973)
- Wolfgang Boehm (1980)

B-Splines

- For Bézier curves, the polynomial degree is directly related to the number of control points.
  - The control of the continuity between Bézier curves is not trivial
- B-Splines are a generalization in the sense that the degree doesn't depend on the number of control points
  - One can impose every continuity at any point of the curve (we will see later how to do that)
  - They are polynomial curves, by pieces (Bézier curves have a unique polynomial representation along the interval of definition)
  - They may provide local control
  - The parametrization can be freely chosen (with Bézier, it is fixed , usually $0<u<1.$)

B-Splines

- Basis of Bézier curves :

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

  - The support of the basis functions is the interval $[0..1]$
  - Continuity is C_∞, and between different Bézier curves it is enforced by a wise choice of the P_i's

- B-splines basis

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

  - The basis functions N_i^d are piecewise polynomials
  - Have a *compact support* + satisfy partition of the unity
  - The continuity is defined at the basis function's level.

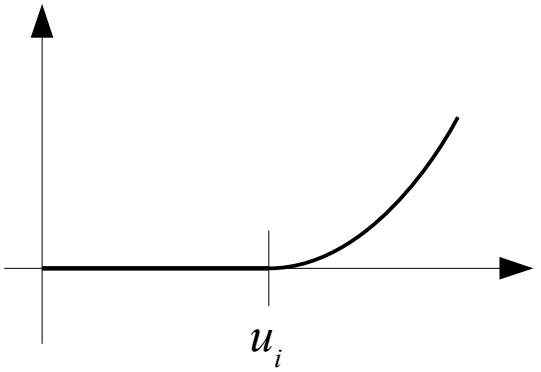
B-Splines

- The basis functions B-spline are defined
  - by the nodal sequence and by the polynomials degree of the curve
- Nodal sequence:
  - It is a series of values u_i (knots) of the parameter u of the curve, not strictly increasing – there can be equal values.
  - ex. $U=\{0,0,0,1,2,3,4,4,4\}$

B-Splines

- Constuction of B-Spline basis functions
  - Truncated Power Function

$$(u-u_i)_+^d = \begin{cases} (u-u_i)^d & \text{if} \quad u \geq u_i \\ 0 & \text{otherwise} \end{cases}$$



  - It is a function of C^{d-1} continuity

B-Splines

- Divided differences
  - order one (similar to a simple derivative)

$$[u_i, u_{i+1}]_U f(U) = \frac{f(u_{i+1}) - f(u_i)}{u_{i+1} - u_i}$$

U is a hidden parameter
(variable used to differentiate)

  - order 2 : application of the above formula twice...

$$[u_i, u_{i+1}, u_{i+2}]_U f(U) = \frac{[u_{i+1}, u_{i+2}]_U f(U) - [u_i, u_{i+1}]_U f(U)}{u_{i+2} - u_i}$$

$$[u_i, u_{i+1}, u_{i+2}]_U f(U) = \frac{\dfrac{f(u_{i+2}) - f(u_{i+1})}{u_{i+2} - u_{i+1}} - \dfrac{f(u_{i+1}) - f(u_i)}{u_{i+1} - u_i}}{u_{i+2} - u_i}$$

B-Splines

- At the order k

$$[u_i, \cdots, u_{i+k}] f = \frac{[u_{i+1}, \cdots, u_{i+k}] f - [u_i, \cdots, u_{i+k-1}] f}{u_{i+k} - u_i}$$

- One assumed that $u_i \neq u_{i+1} \neq u_{i+2} \cdots$

- Properties (see Bartels, 1987)

1- In the case where $u_i = u_{i+1} = u_{i+2} \cdots$

$$[u_i, \cdots, u_{i+k}] f = \frac{1}{k!} \frac{d^k f}{d u^k}\bigg|_{u=u_i}$$

2- if $u_i \neq u_{i+1} \neq u_{i+2} \cdots$ and $u_i < u_{i+1} < u_{i+2} \cdots$

$$[u_i, \cdots, u_{i+k}] f = \frac{1}{k!} \frac{d^k f}{d u^k}\bigg|_{u=u^*}, u_i < u^* < u_{i+k}$$

B-Splines

3- $[u_i, \cdots, u_{i+k}]f$ is symmetric with respect to the knot vector

4- If $f(u)$ is a polynomial of degree at the most equal to k , then $[u_i, \cdots, u_{i+k}]f$

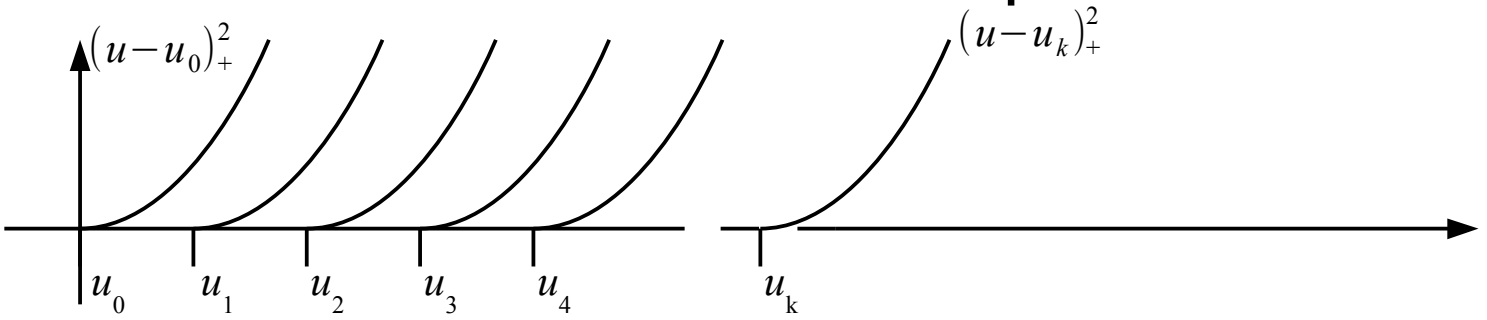
is a constant with respect to the u_i.

5- The divided difference of $f=g(u).h(u)$ is :

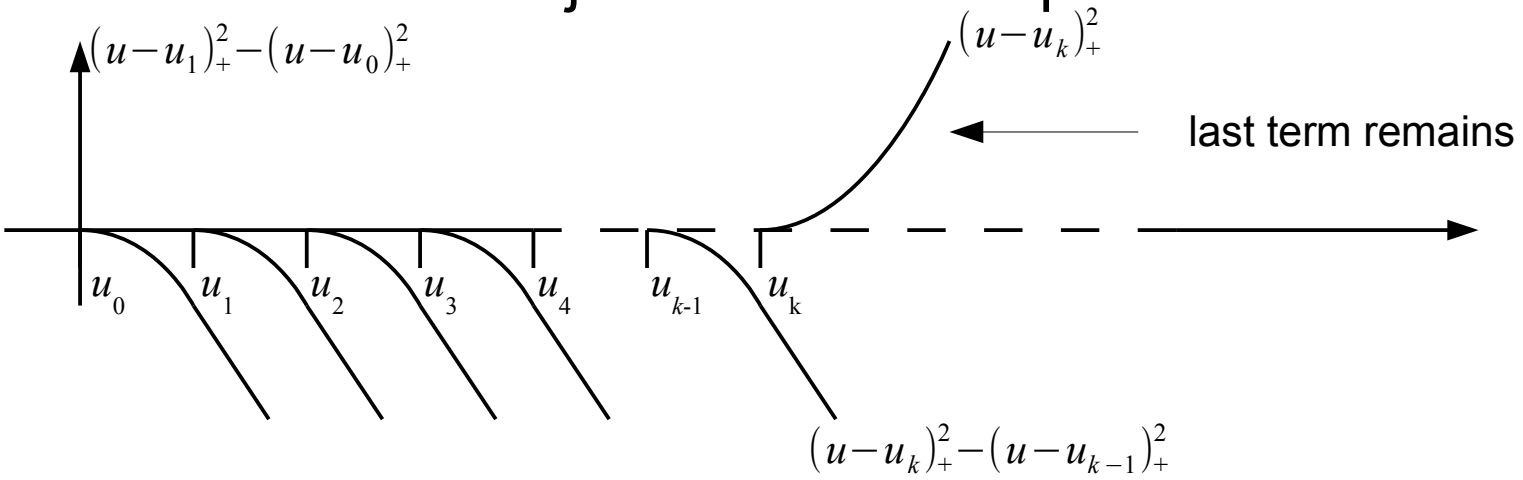
$$[u_i, \cdots, u_{i+k}]f = \sum_{j=i}^{j=i+k} ([u_i, \cdots, u_j]g)\cdot([u_j, \cdots, u_{i+k}]h)$$

B-Splines

- Divided differences and B-Splines



  - How to cancel quadratic terms ?
    → subtract adjacent truncated power functions.



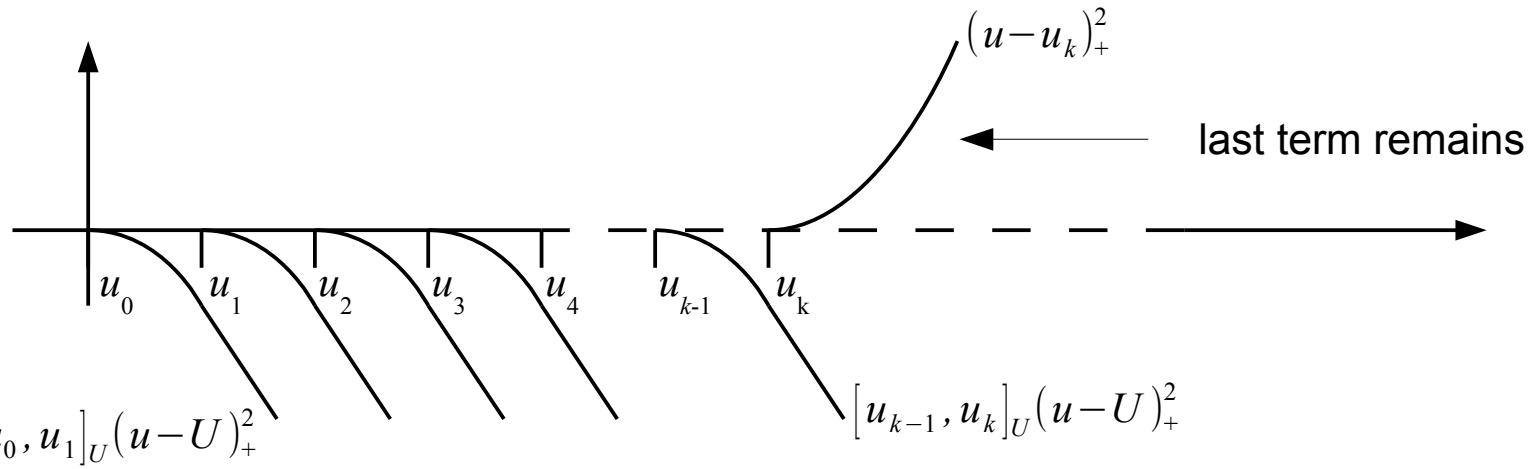
B-Splines

- Problem, lower order terms are dependent on k

$$(u-u_k)_+^2-(u-u_{k-1})_+^2\Big|_{u>u_k}=0\cdot u^2+(u_k-u_{k-1})\cdot u+(u_k-u_{k-1})(u_k+u_{k-1})\cdot 1$$

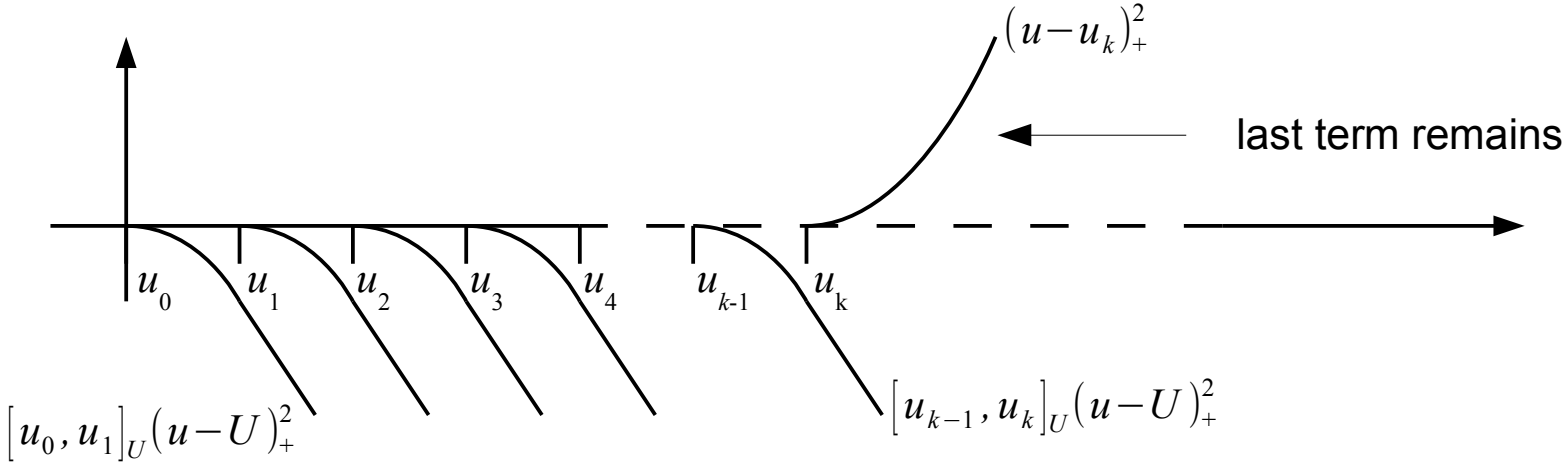
But, dividing by (u_k-u_{k-1}) yields a divided difference :

$$\frac{(u-u_k)_+^2-(u-u_{k-1})_+^2}{u_k-u_{k-1}}=[u_{k-1},u_k]_U(u-U)_+^2$$

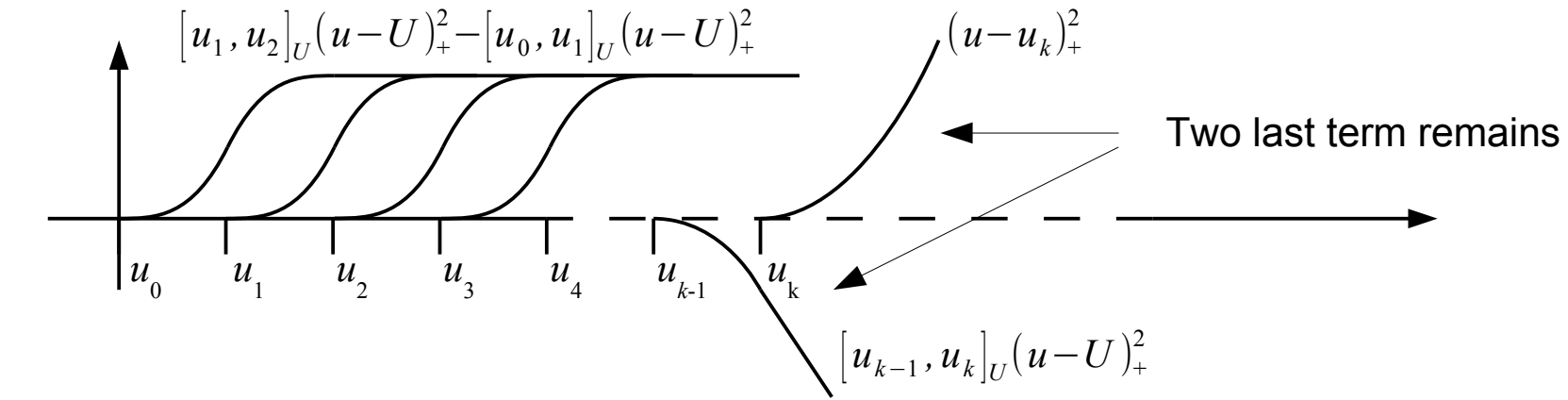


B-Splines

- Now, cancel linear terms …



- Same procedure : subtract adjacent terms.



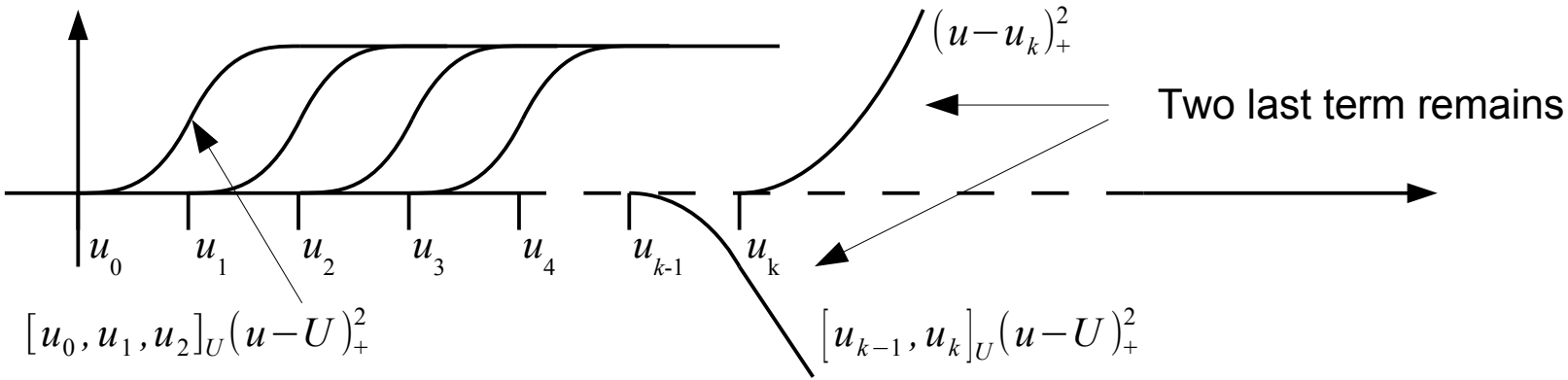
B-Splines

- Again, lower order terms are dependent on k

$$\left[u_{k-1},u_{k-2}\right]_U\left(u-U\right)_+^2-\left[u_k,u_{k-1}\right]_U\left(u-U\right)_+^2\Big|_{u>u_k}$$
$$=0\cdot u+\left(\left(u_k+u_{k-1}\right)-\left(u_{k-1}+u_{k-2}\right)\right)\cdot 1=\left(u_k-u_{k-2}\right)\cdot 1$$

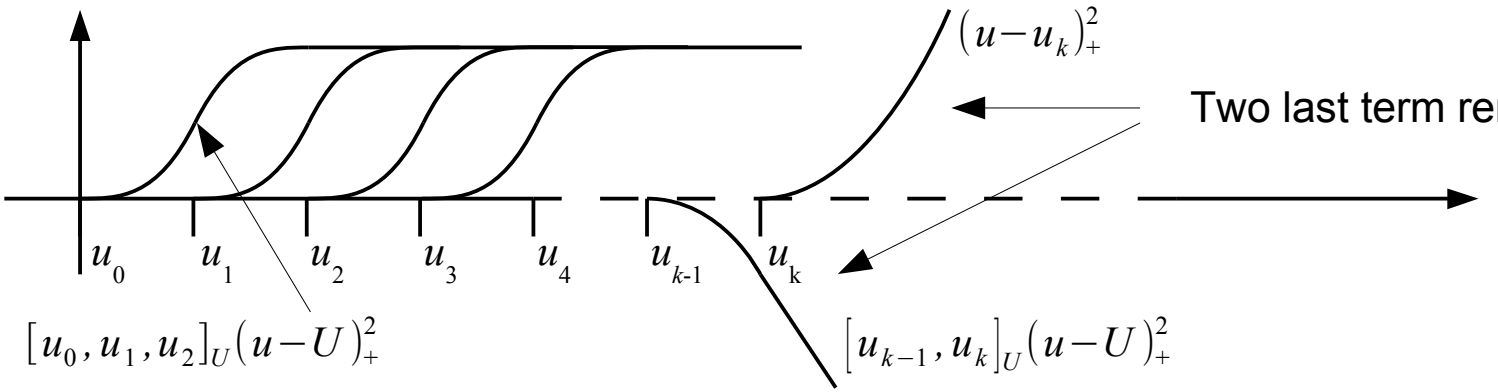
Dividing by $\left(u_k-u_{k-2}\right)$ yields again a divided difference :

$$\frac{\left[u_{k-1},u_{k-2}\right]_U\left(u-U\right)_+^2-\left[u_k,u_{k-1}\right]_U\left(u-U\right)_+^2}{u_k-u_{k-2}}$$
$$=\left[u_{k-2},u_{k-1},u_k\right]_U\left(u-U\right)_+^2$$

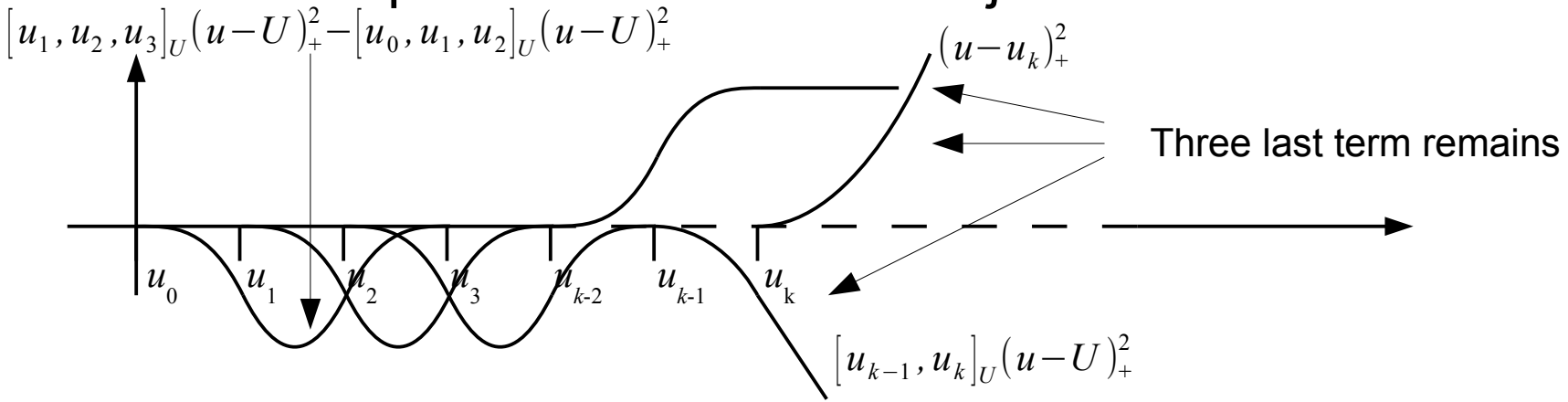


B-Splines

- Now, cancel constant terms …



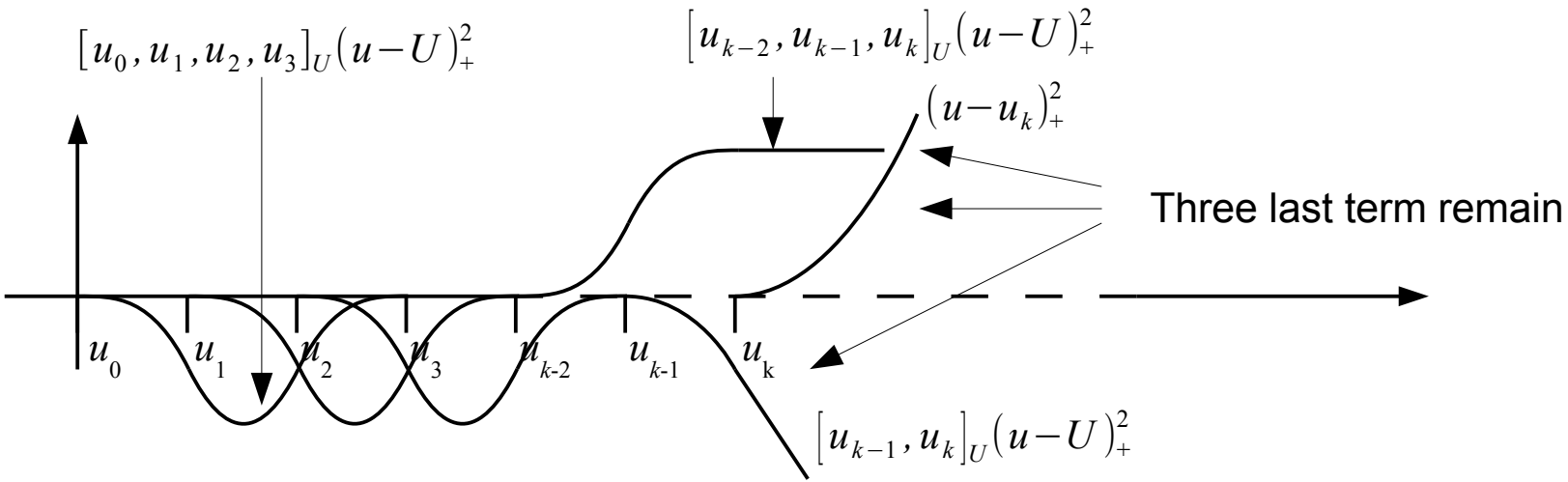
- Same procedure : subtract adjacent terms.



B-Splines

- There are no lower order terms. However we might divide anyway by $(u_k - u_{k-3})$ to remain consistent and get again an expression as a divided difference...

$$\frac{[u_{k-2}, u_{k-1}, u_k]_U (u-U)_+^2 - [u_{k-3}, u_{k-2}, u_{k-1}]_U (u-U)_+^2}{u_k - u_{k-3}}$$
$$= [u_{k-3}, u_{k-2}, u_{k-1}, u_k]_U (u-U)_+^2$$



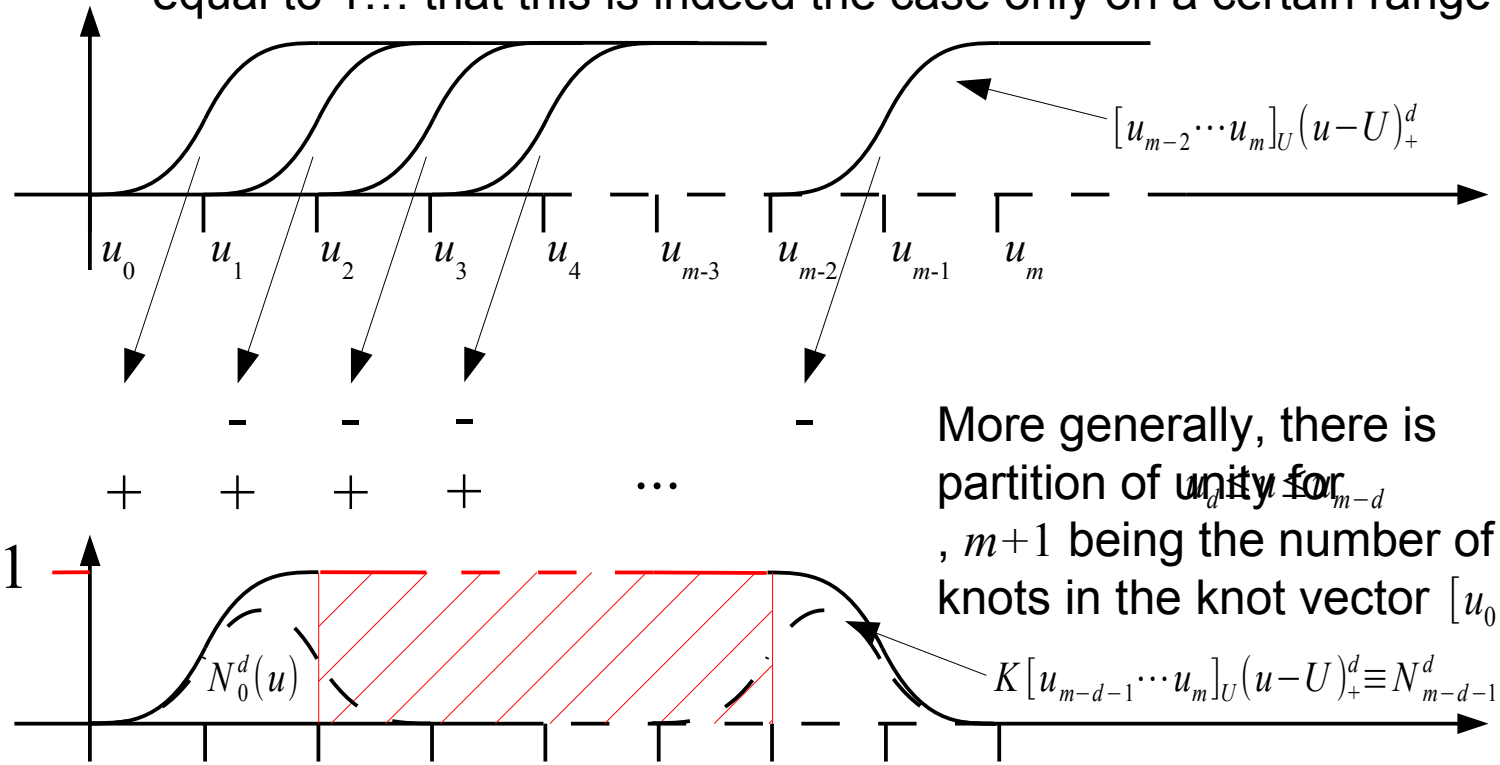
B-Splines

- The sign is alternating with the degree. Shape function of even degree are negative, while SF of uneven degree are negative.
- Multiplying by $(-1)^{d+1}$ makes every SF positive.
- To ensure that the SF form a partition of unity , we have to multiply again by $(u_{i+d+1}-u_i)$
- The compact representation of the B-Splines basis functions of degree d with the use of divided differences is therefore :

$$N_i^d=(-1)^{d+1}(u_{i+d+1}-u_i)[u_i,\cdots,u_{i+d+1}]_U(u-U)_+^d$$

B-Splines

- Proof of the partition of unity : consider the second last operation (the cancellation of constant terms)
  - We subtract consecutive terms to form the final shape functions
  - Partition of unity means the sum of all the final shape functions is equal to 1… that this is indeed the case only on a certain range of u.



More generally, there is partition of unity for $u_d \le u \le u_{m-d}$, $m+1$ being the number of knots in the knot vector $[u_0 \cdots u_m]$

B-Splines

- Recursive definition of basis functions
  - Setting $U=\{u_0,\cdots,u_m\}$, $u_i \le u_{i+1}$, $i=0\cdots m-1$ (nodal sequence)
  - The functions are such as : (recurrence formula of Cox – de Boor)

$$N_i^0(u)=\begin{cases}1 & \text{if} \quad u_i \le u < u_{i+1}\\ 0 & \text{otherwise}\end{cases}$$

$$N_i^d(u)=\frac{u-u_i}{u_{i+d}-u_i}N_i^{d-1}(u)+\frac{u_{i+d+1}-u}{u_{i+d+1}-u_{i+1}}N_{i+1}^{d-1}(u)$$

  - Where $u_{i+d} - u_i = 0$, necessarily $N_i^{d-1}(u)\equiv 0$

    By convention, we set in this case $\frac{0}{0}=0$ when the limit is undefined.

B-Splines

- Example : computation of basis functions of degree $d \leq 2$ for $U = \{u_0 = 0, u_1 = 0, u_2 = 0, u_3 = 1, u_4 = 1, u_5 = 1\}$

$$N_0^1(u) = \frac{u - u_0}{u_1 - u_0} N_0^0(u) + \frac{u_2 - u}{u_2 - u_1} N_1^0(u) \rightarrow \frac{0}{0} = 0 \text{ by convention}$$

$$N_0^0 = 0 \quad N_1^0 = 0 \quad N_2^0 = \begin{cases} 1 & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \quad N_3^0 = 0 \quad N_4^0 = 0$$

$$N_0^1 = 0 \quad N_1^1 = \begin{cases} 1-u & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \quad N_2^1 = \begin{cases} u & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \quad N_3^1 = 0$$

$$N_0^2 = \begin{cases} (1-u)^2 & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \quad N_1^2 = \begin{cases} 2u(1-u) & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases} \quad N_2^2 = \begin{cases} u^2 & 0 \leq u < 1 \\ 0 & \text{otherwise} \end{cases}$$

Bernstein polynomials of degree 2

B-Splines

- The Bernstein polynomials of degree d are a particular case of the B-splines basis
  - They correspond to a nodal sequence

$$U_B = \{u_0 = 0, \cdots, u_d = 0, u_{d+1} = 1, \cdots, u_{2d+1} = 1\}$$

  - Bézier curves are therefore a particular case of B-splines.
  - It is also possible to transform any B-spline into a sequence of Bézier curves – because the Bernstein polynomials form a complete basis of polynomials of degree d.

B-Splines

- Basis functions and control points
  - In contrary to Bézier curves, the number of control points is not imposed by the degree d
  - Let $m+1$ the number of knots. We have $n+1$ independent basis functions at our hands
    - For every basis function, we associate a control point

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

  - The number of control points is fixed by the relation $n+1=m-d$

B-Splines

- Types of nodal sequence...
  - Uniform – The gap between two successive knots is constant

$$U = \{u_0, u_1, \cdots, u_{m-d-1}\} \quad , \quad u_{i+1} - u_i = k$$

  - Periodic - The gap between the knots at the start of a nodal sequence is identical to the one at the end of the nodal sequence

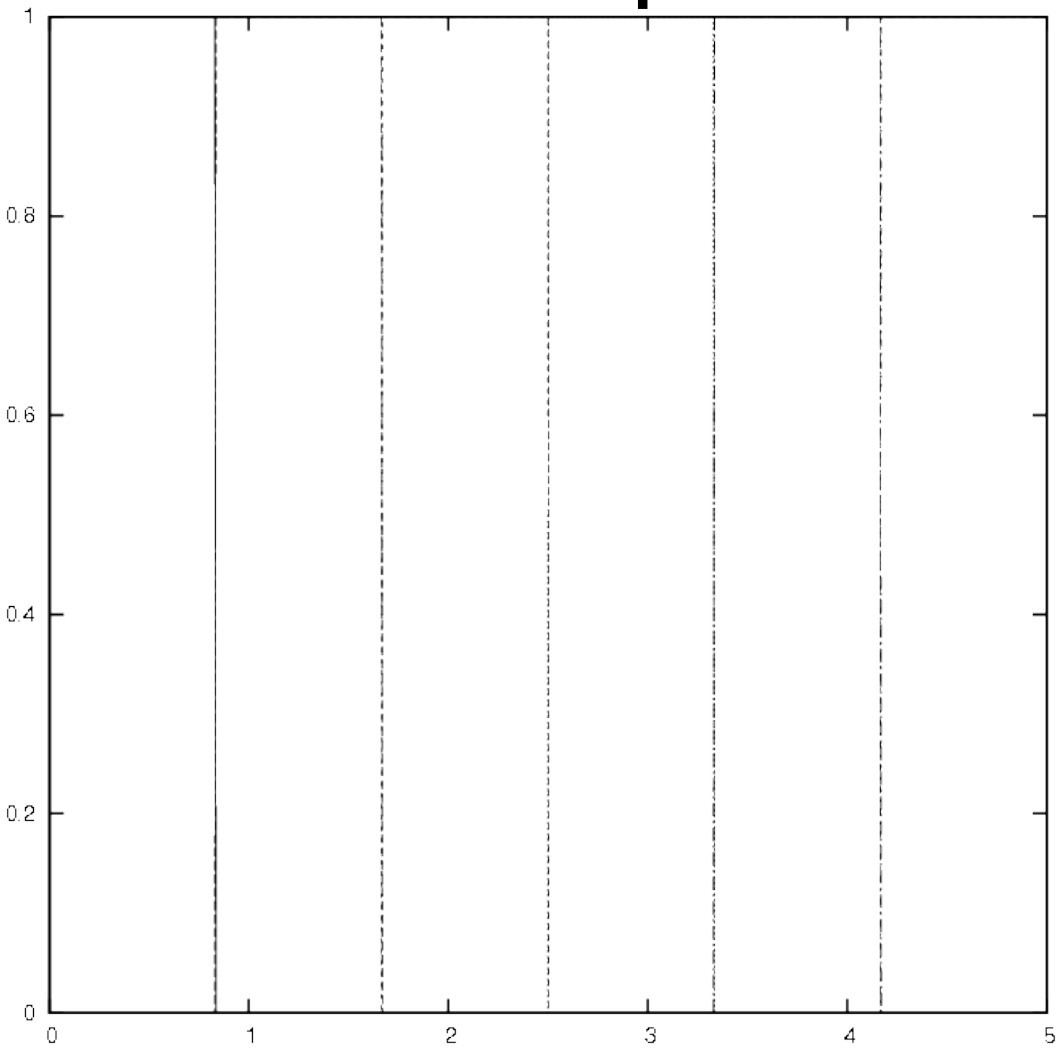
$$U = \{\underbrace{u_0, \cdots, u_d}_{d+1}, u_{d+1}, \cdots, u_{m-d-1}, \underbrace{u'_0, \cdots, u'_d}_{d+1}\} \quad , \quad u'_i - u_i = k$$

  - Non uniform, interpolating – first and last control point are interpolated

$$U = \{\underbrace{a, \cdots, a}_{d+1}, u_{d+1}, \cdots, u_{m-d-1}, \underbrace{b, \cdots, b}_{d+1}\}$$

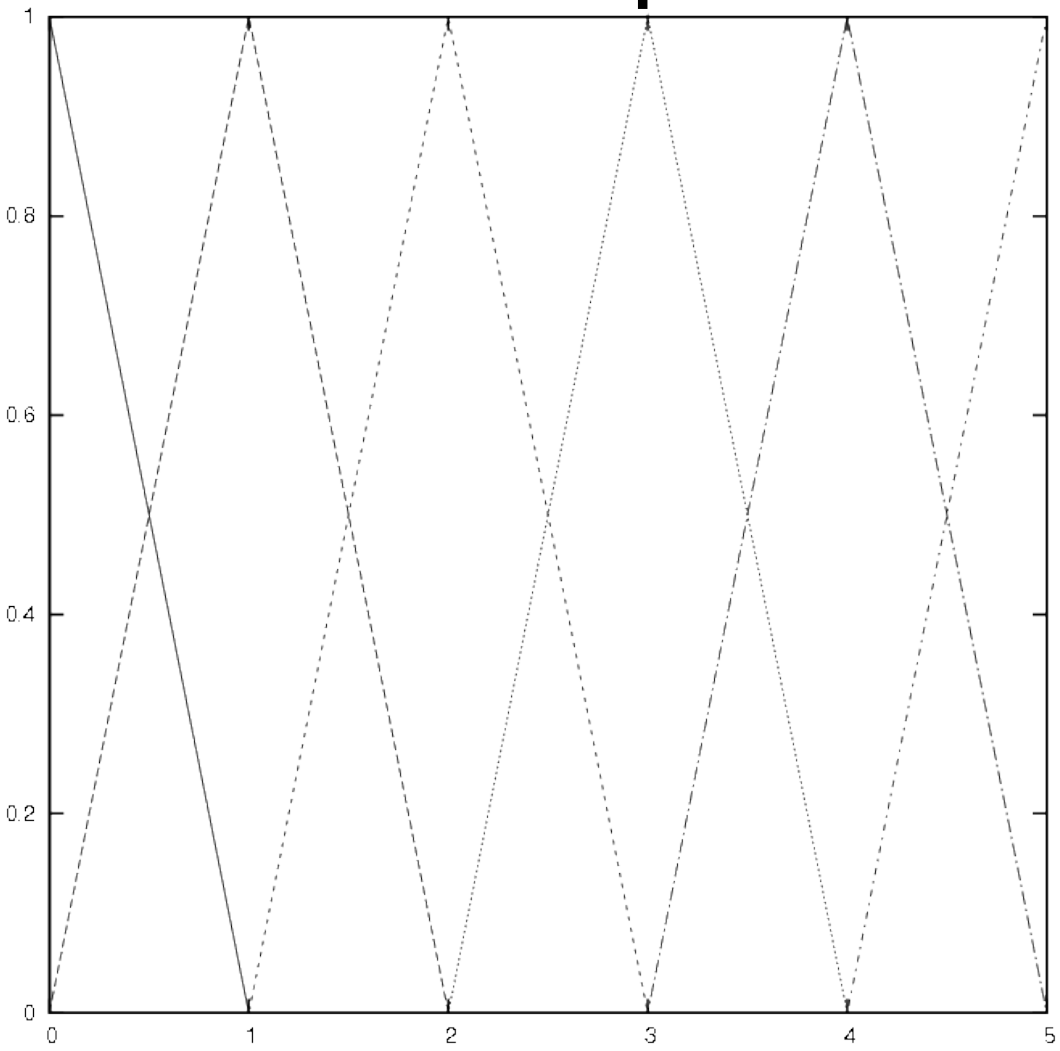
In the sequel, except where indicated, we consider non uniform nodal sequences interpolating the first and last control points.

B-Splines



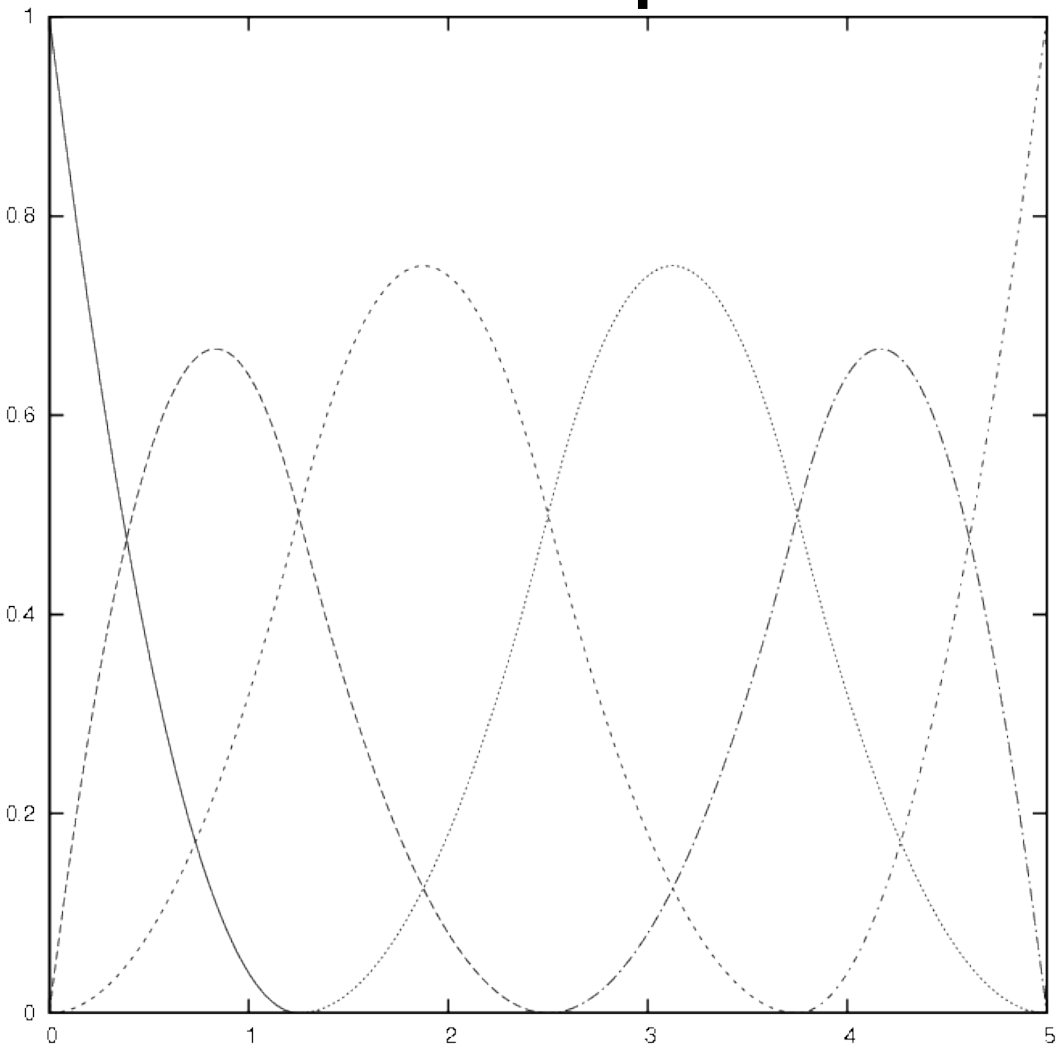
$$U=\{0,\frac{5}{6},\frac{10}{6},\frac{15}{6},\frac{20}{6},\frac{25}{6},5\}\quad d=0\quad m+1=7\quad n+1=6$$

B-Splines



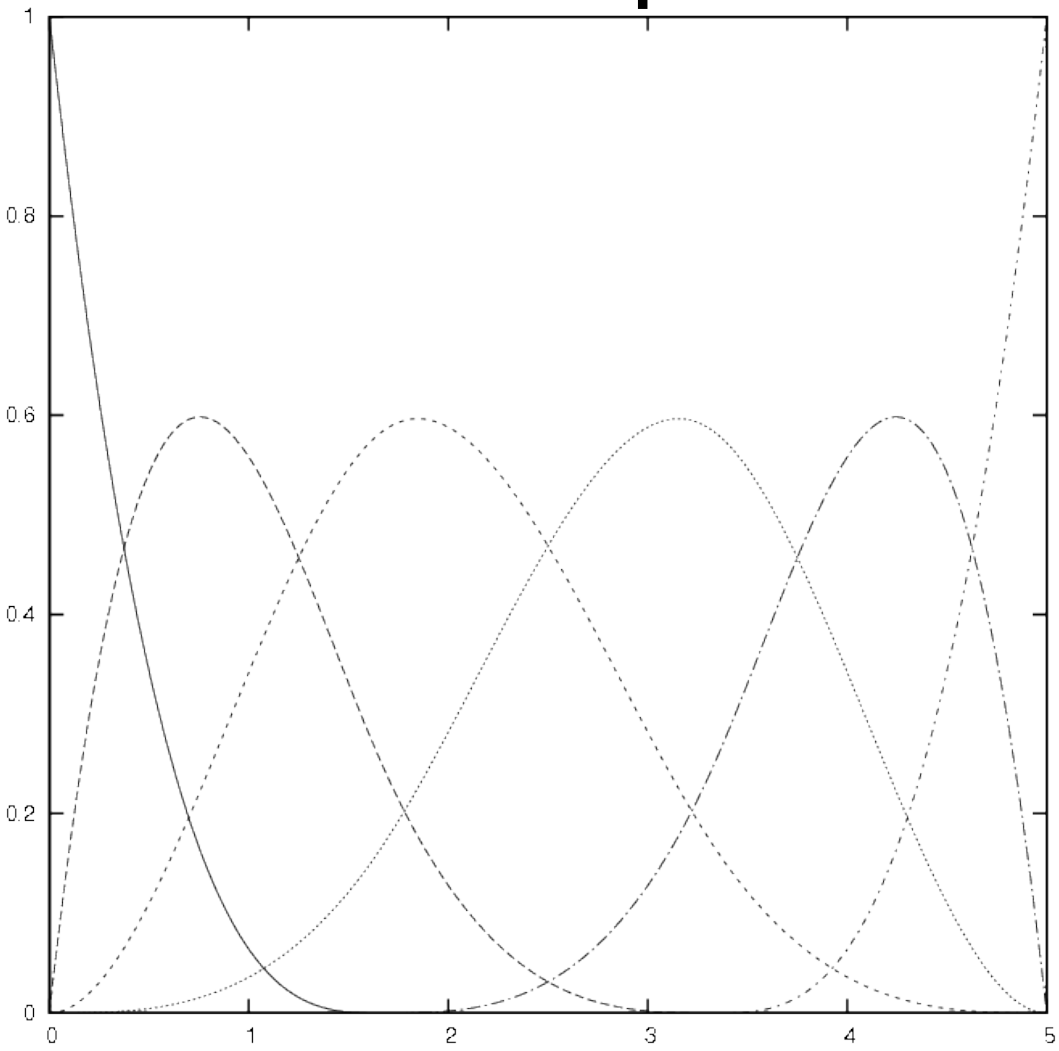
$$U=\{0,0,1,2,3,4,5,5\} \quad d=1 \quad m+1=8 \quad n+1=6$$

B-Splines



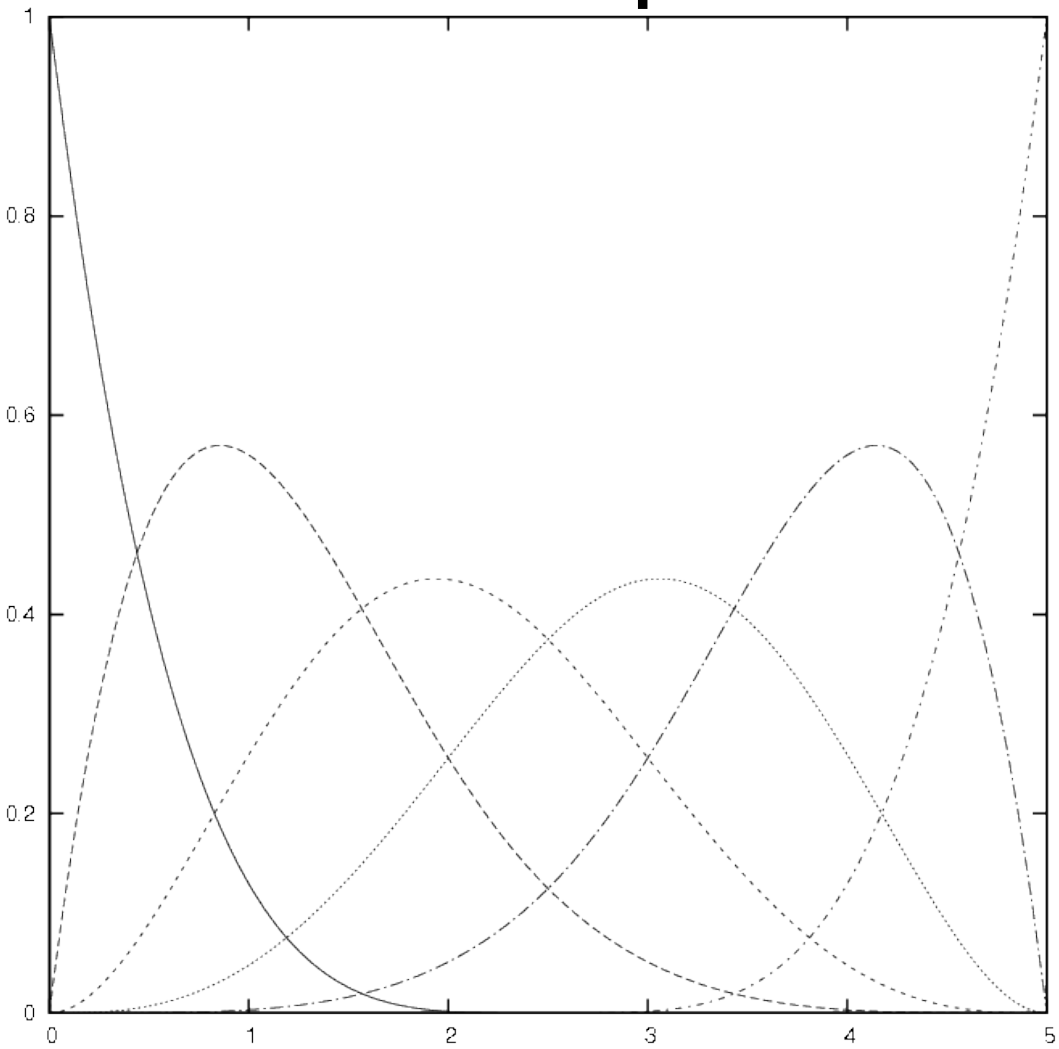
$$U=\{0,0,0,\frac{5}{4},\frac{10}{4},\frac{15}{4},5,5,5\} \quad d=2 \quad m+1=9 \quad n+1=6$$

B-Splines



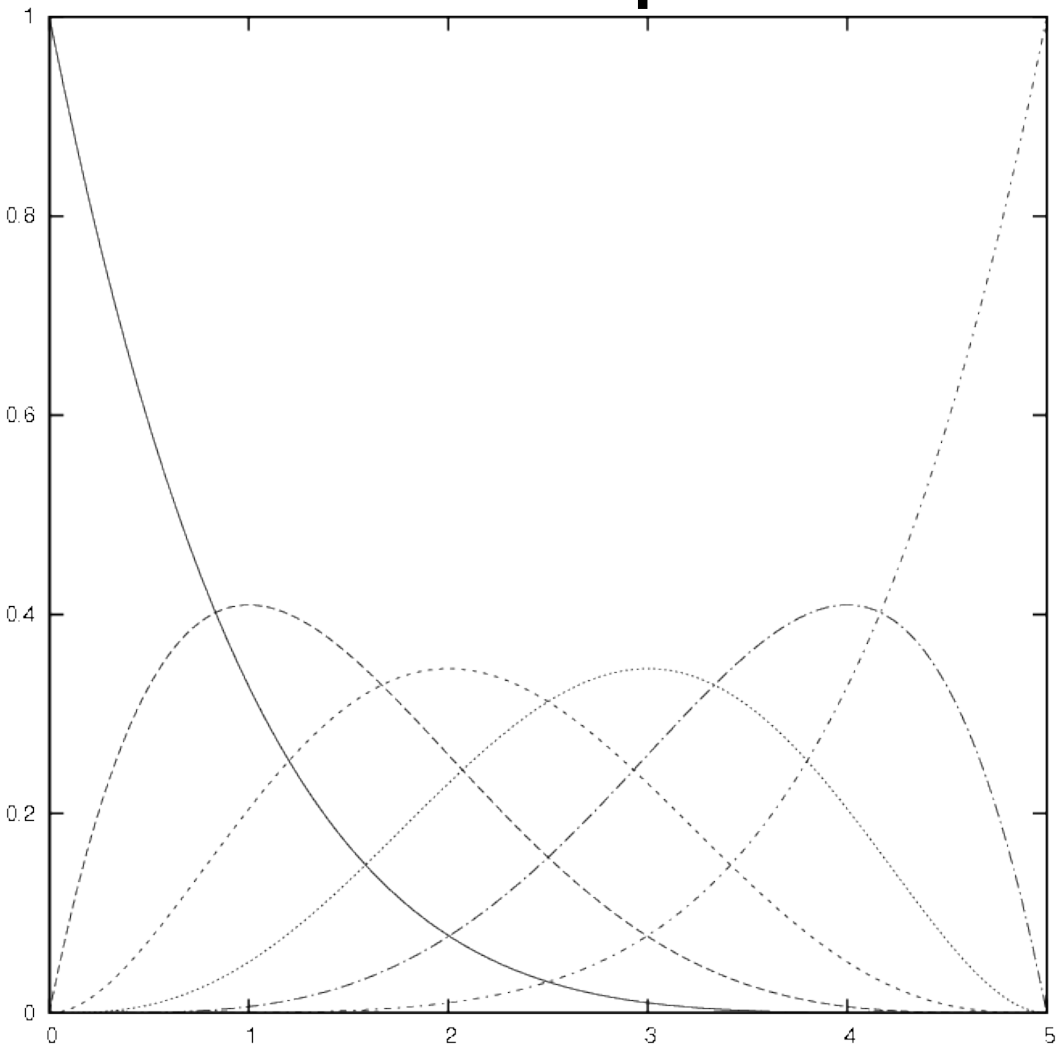
$$U=\{0,0,0,0,\frac{5}{3},\frac{10}{3},5,5,5,5\} \quad d=3 \quad m+1=10 \quad n+1=6$$

B-Splines



$$U=\{0,0,0,0,0,\frac{5}{2},5,5,5,5,5\} \quad d=4 \quad m+1=11 \quad n+1=6$$

B-Splines



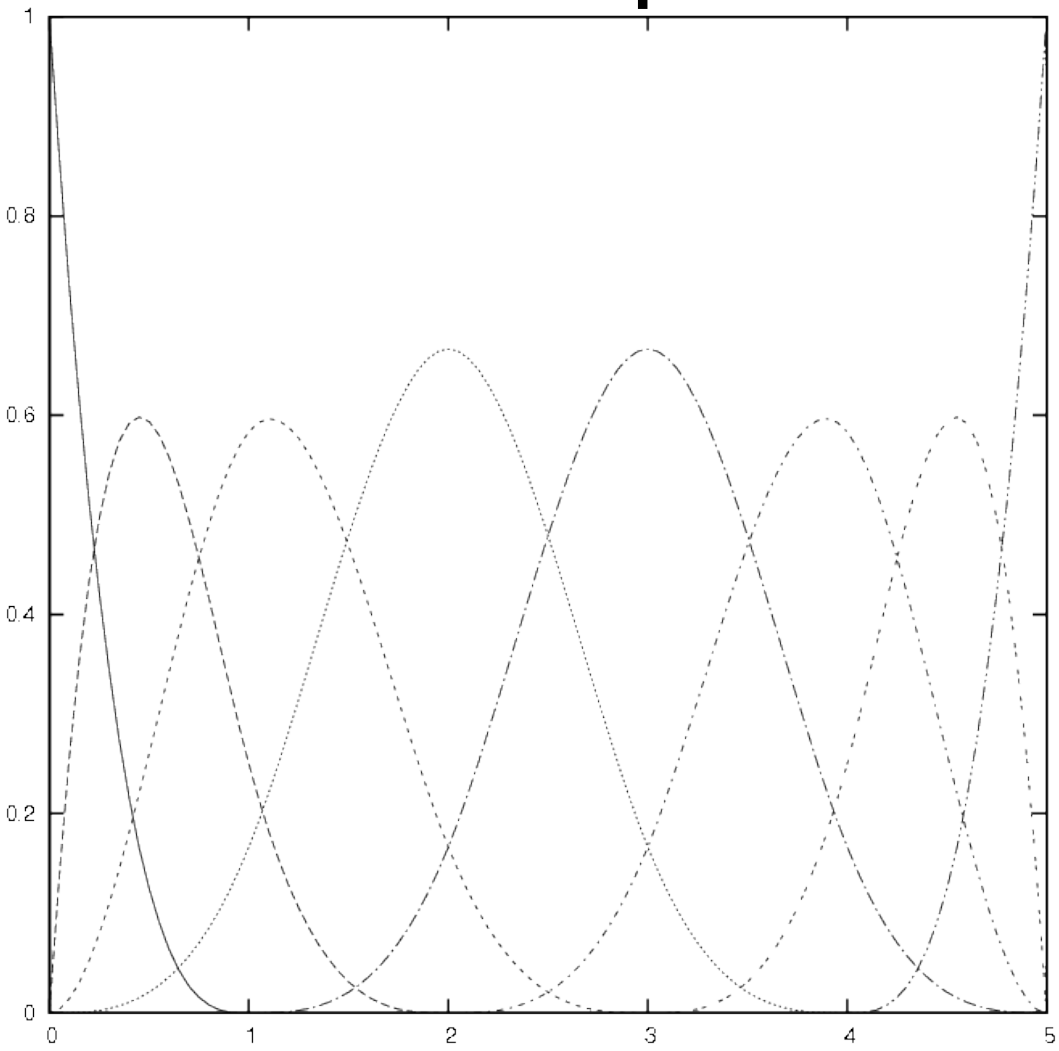
$U=\{0,0,0,0,0,0,5,5,5,5,5,5\}$ $d=5$ $m+1=12$ $n+1=6$

Bernstein polynomials (with a factor on u)

B-Splines

- Properties of B-spline basis functions
  - $N_i^d(u)=0$ outside the interval $[u_i, u_{i+d+1}[$
  - Inside the interval $[u_i, u_{i+1}[$, at most $d+1$ functions $N_*^d(u)$ are non zero : $N_{i-d}^d, \cdots, N_i^d$
  - $N_i^d(u) \geq 0 \quad \forall i, d$ and u (always positive)
  - For $u \in [u_i, u_{i+1}[$, $\sum_{j=i-d} N_j^d(u)=1$ (forms a partition of unity)
  - All derivatives of $N_i^d(u)$ exist inside the interval $[u_i, u_{i+1}[$. At a knot , $N_i^d(u)$ is d-k times differentiable, k being the node multiplicity.
  - Except for $d=0$, $N_i^d(u)$ reaches exactly one maximum

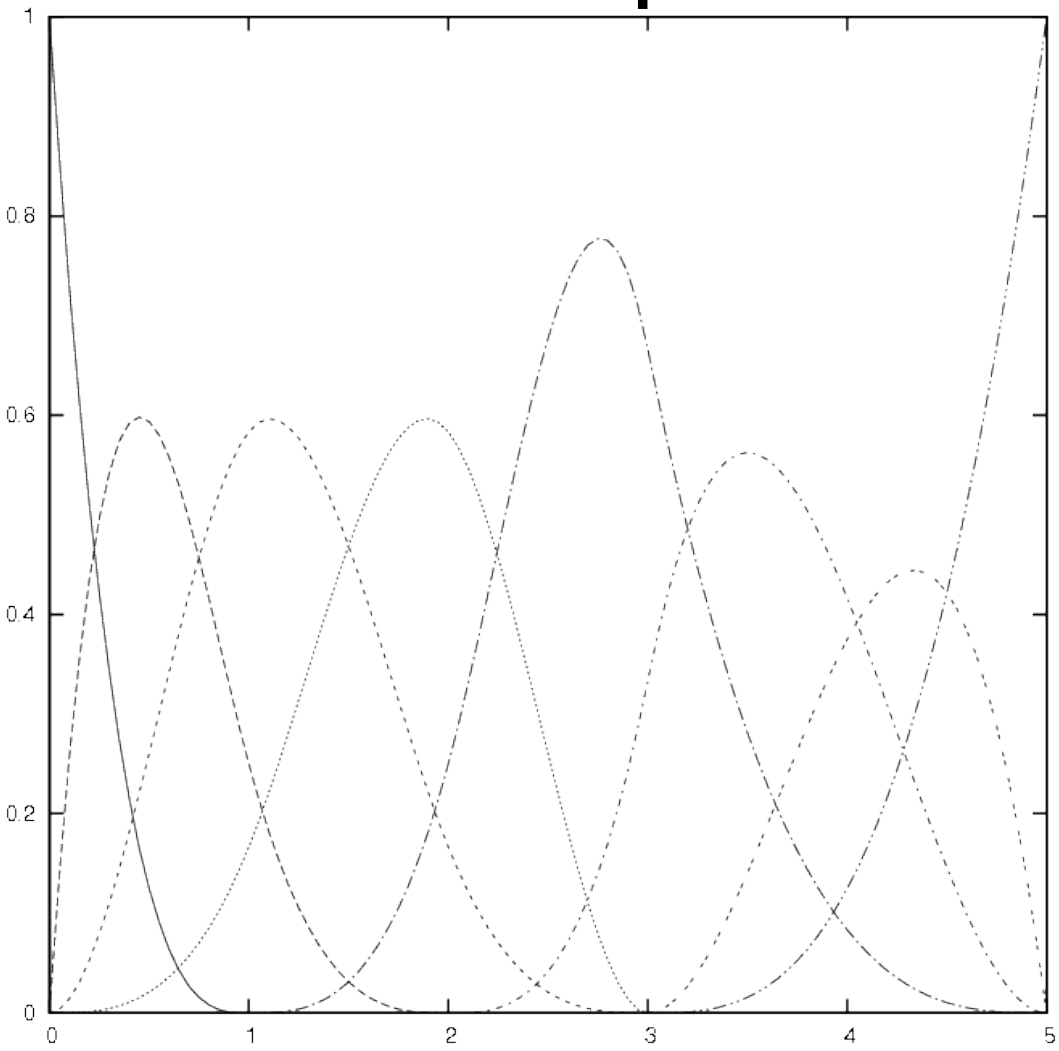
B-Splines



$$U=\{0,0,0,0,1,2,3,4,5,5,5,5\} \quad d=3 \quad m+1=12 \quad n+1=8$$

The knot u=3 is of multiplicity 1

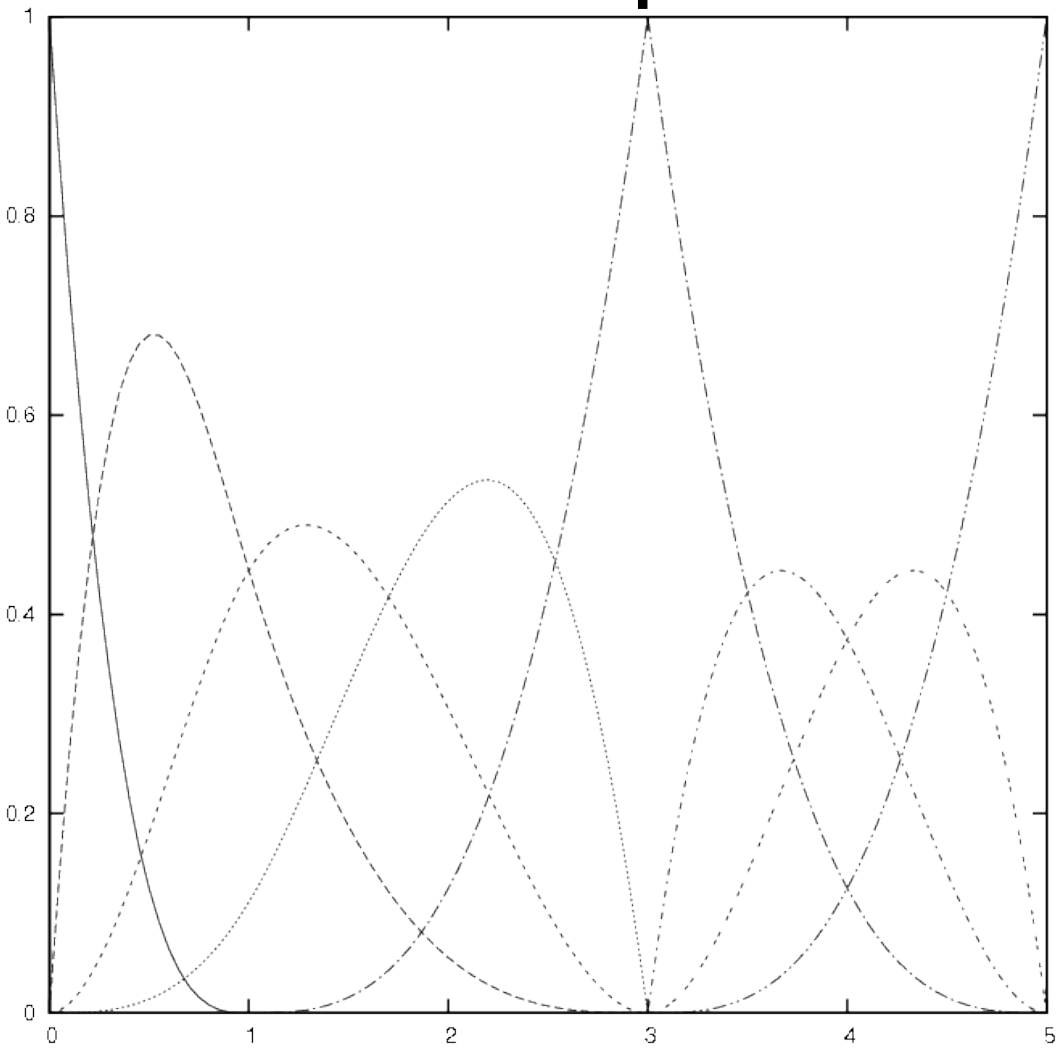
B-Splines



$U=\{0,0,0,0,1,2,3,3,5,5,5,5\}$ $d=3$ $m+1=12$ $n+1=8$

The node u=3 is of multiplicity 2

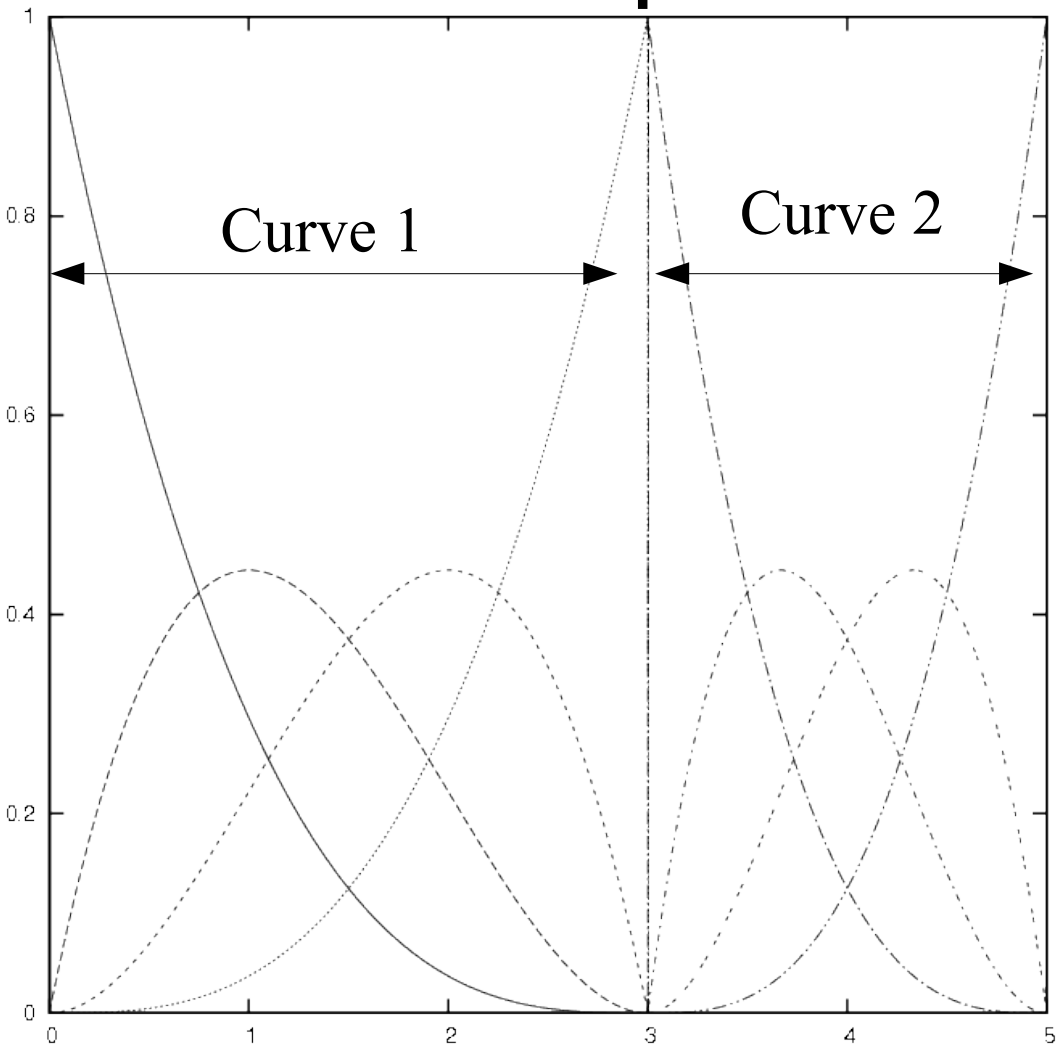
B-Splines



$$U=\{0,0,0,0,1,3,3,3,5,5,5,5\} \quad d=3 \quad m+1=12 \quad n+1=8$$

The node u=3 is of multiplicity 3

B-Splines



$U=\{0,0,0,0,3,3,3,3,5,5,5,5\}$ $d=3$ $m+1=12$ $n+1=8$

The node u=3 is of multiplicity 4

B-Splines

- The characteristics of basis functions involve that the B-Spline curve

$$P(u)=\sum_{i=0}^{n} P_i N_i^d(u) \quad U=\{u_0,\cdots,u_m\}\ ,\ u_i\le u_{i+1}\ ,\ i=0\cdots m-1$$

  - interpolates P_0 and P_n ,(only if the nodal sequence admits $d+1$ repetitions at the start and at the end !)
  - is invariant by affine transformation ,
  - is contained by the convex hull of the control points (because $P(u)$ is a linear combination of the control points with positive coefficients which sum to one)

B-Splines

(Following)

- Is *variation diminishing* : The number of inflexion points is lower than the number of wiggles of the characteristic polygon
- Is closed and convex if the characteristic polygon is closed and convex,
- Is of length shorter or equal than that of the control polygon.
- Is invariant by linear transformation of the nodal sequence $u'=au+b$, $a>0$

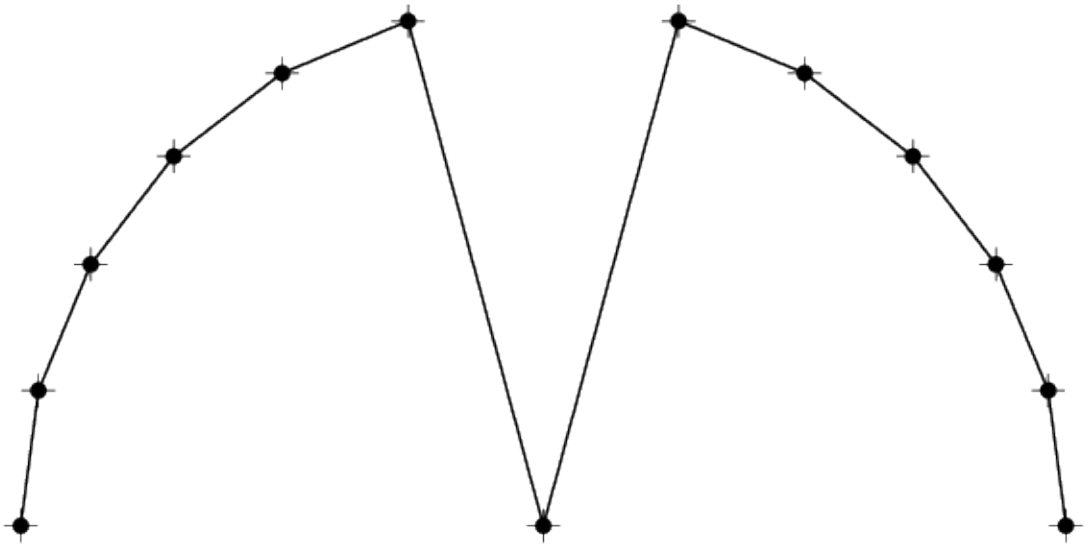
B-Splines

- Control points, degree and nodal sequence
  - We associate a control point for each basis function N_i^* . We have $n+1$ control points.
  - The degree d is chosen by the user.
  - The nodal sequence (that defines the intervals of the parameter on which the curve has a unique polynomial definition) is then built. We have $m+1=n+d+2$ knots (with $d+1$ repetitions at the start and at the end)
    - there remains $n-d$ values of the parameter to set (without taking into account the boundaries)

B-Splines

- Geometric examples
  - Constant number of control points
  - We increase the degree
  - Uniform repartition of knots (except at the boundaries)
  - For which degree do we have the best approximation of the control points ??

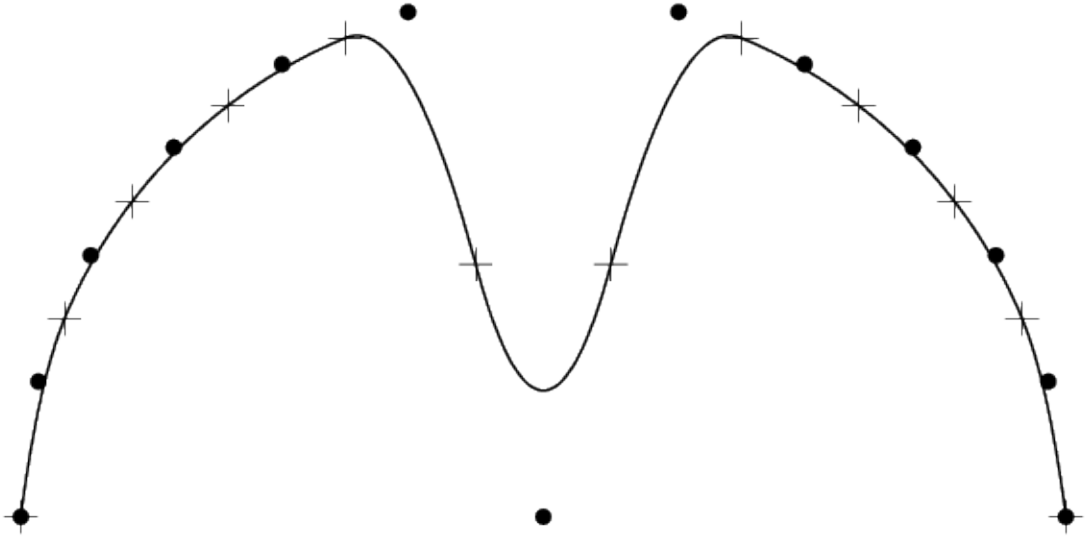
B-Splines



Degree 1

0 0 0.0833333 0.166667 0.25 0.333333 0.416667 0.5 0.583333 0.666667 0.75 0.833333 0.916667 1 1

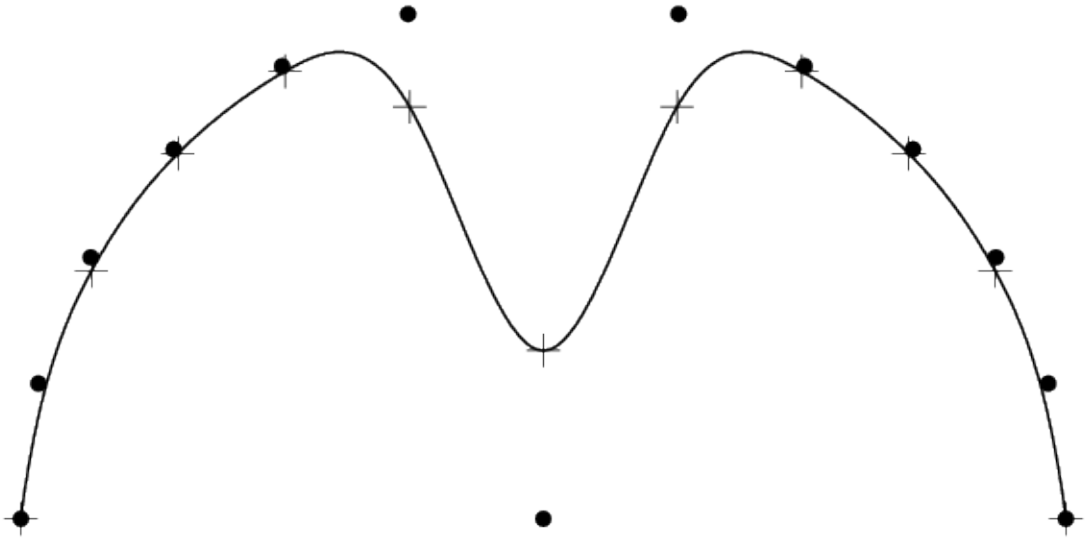
B-Splines



degree 2

0 0 0 0.0909091 0.181818 0.272727 0.363636 0.454545 0.545455 0.636364 0.727273 0.818182 0.909091 1 1 1

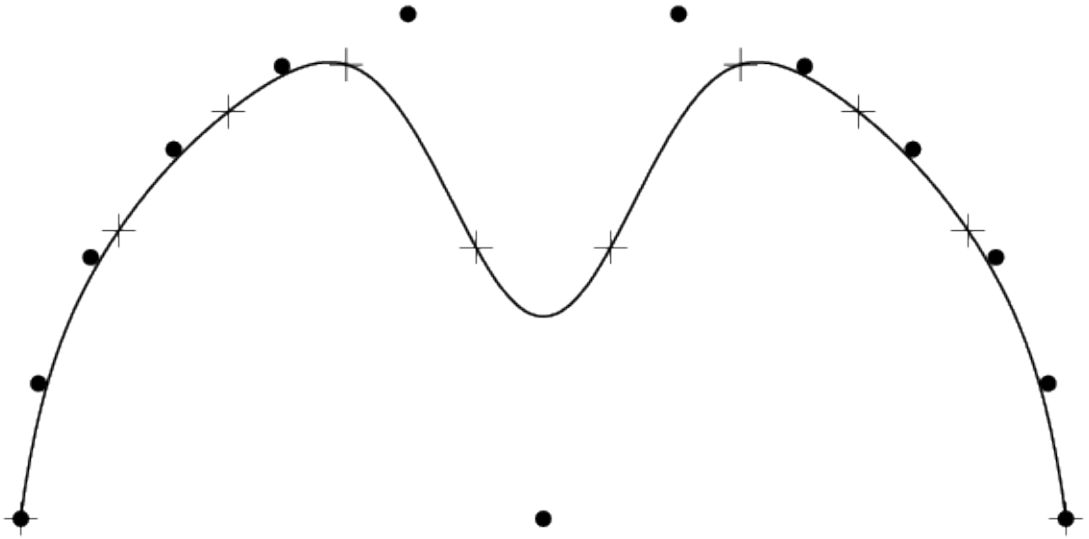
B-Splines



degree 3

0 0 0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1

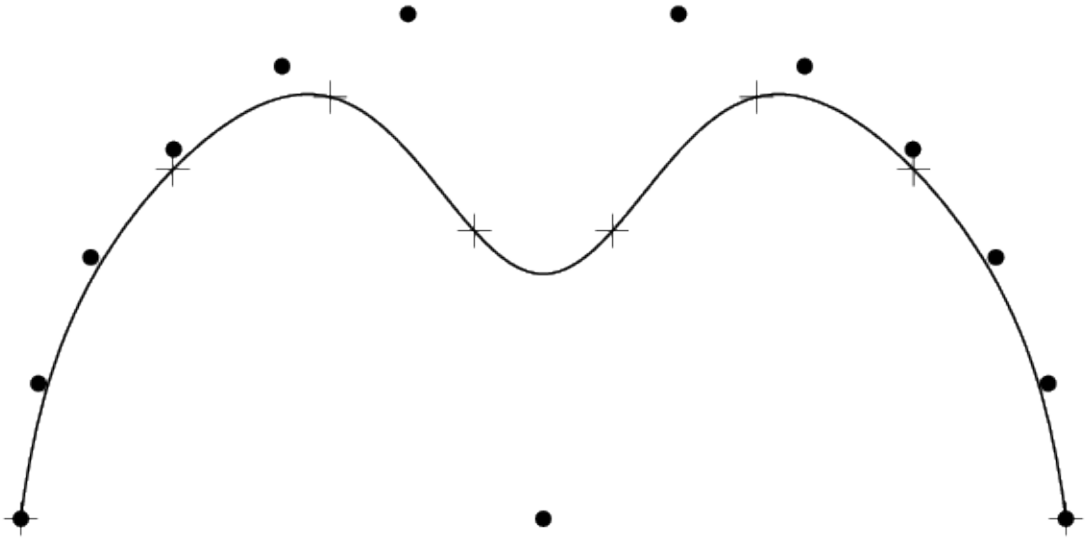
B-Splines



degree 4

0 0 0 0 0 0.111111 0.222222 0.333333 0.444444 0.555556 0.666667 0.777778 0.888889 1 1 1 1 1

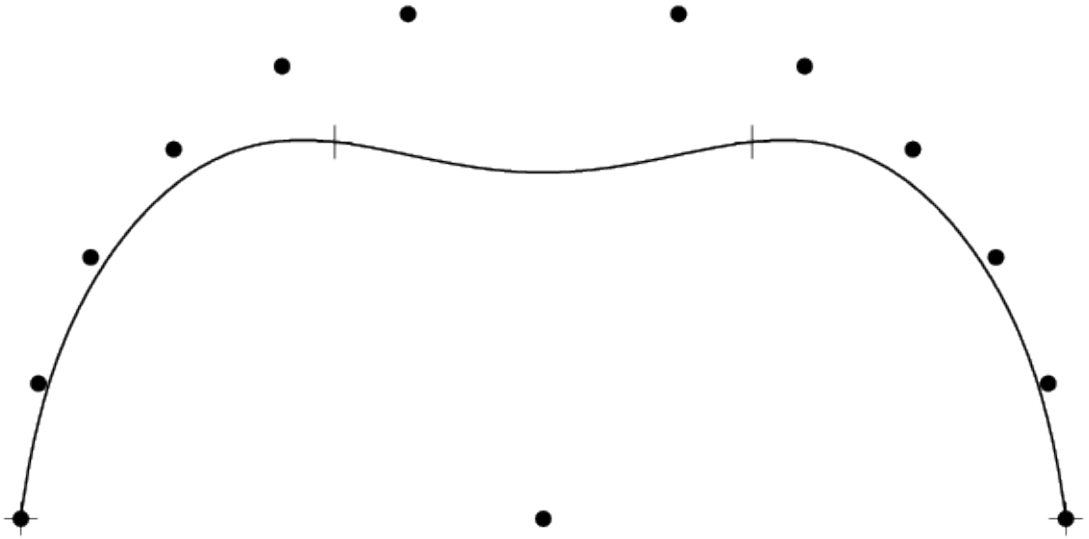
B-Splines



degree 6

0 0 0 0 0 0 0 0.142857 0.285714 0.428571 0.571429 0.714286 0.857143 1 1 1 1 1 1 1

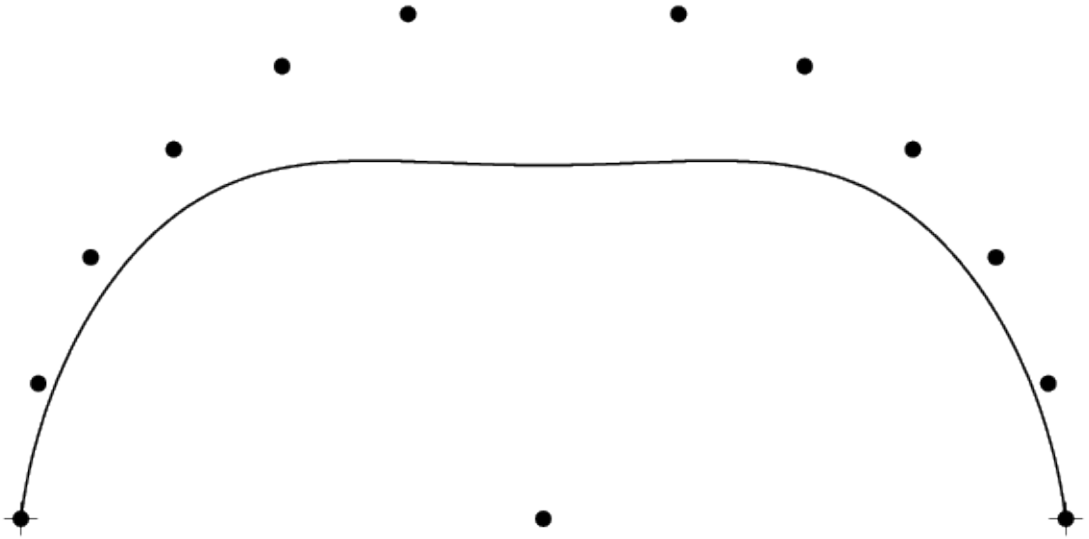
B-Splines



degree 10

0 0 0 0 0 0 0 0 0 0 0 0.333333 0.666667 1 1 1 1 1 1 1 1 1 1

B-Splines

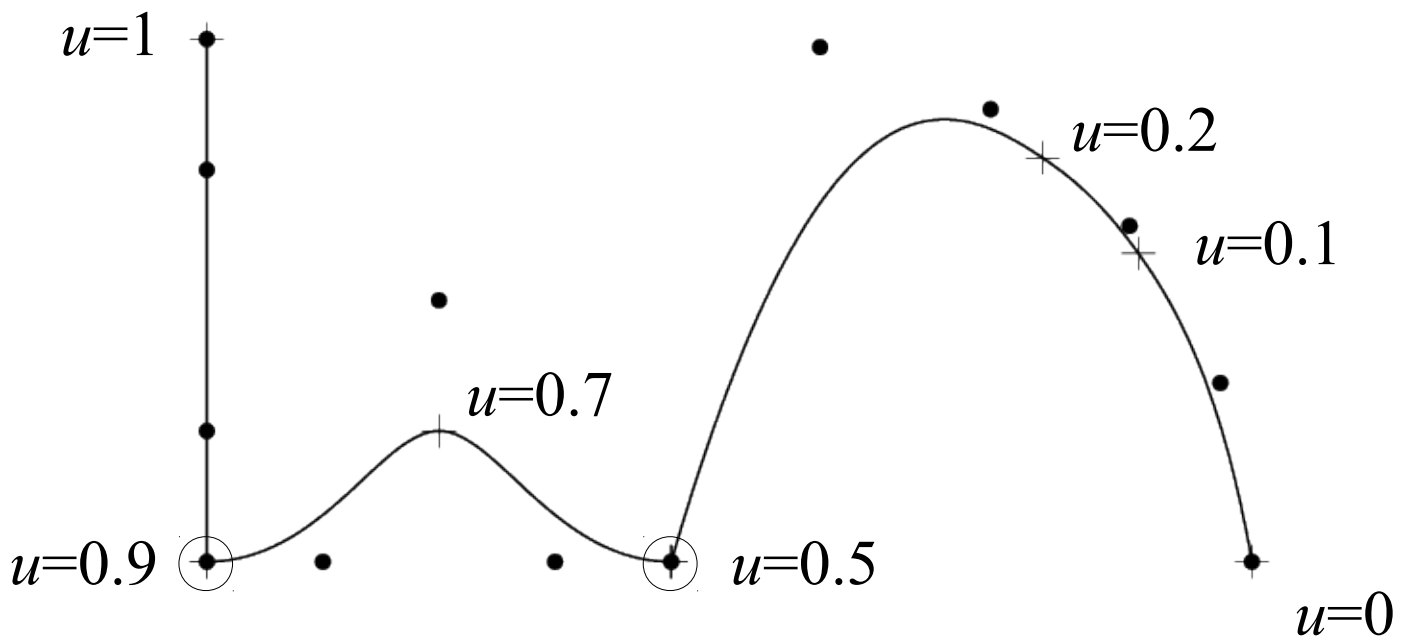


degree 12 (Bézier)
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1

B-Splines

- Impose interpolation points (and C_0 continuity)
  - It is the same as positioning knots of multiplicity d in the nodal sequence
  - One could also repeat d control points...(not shown here)

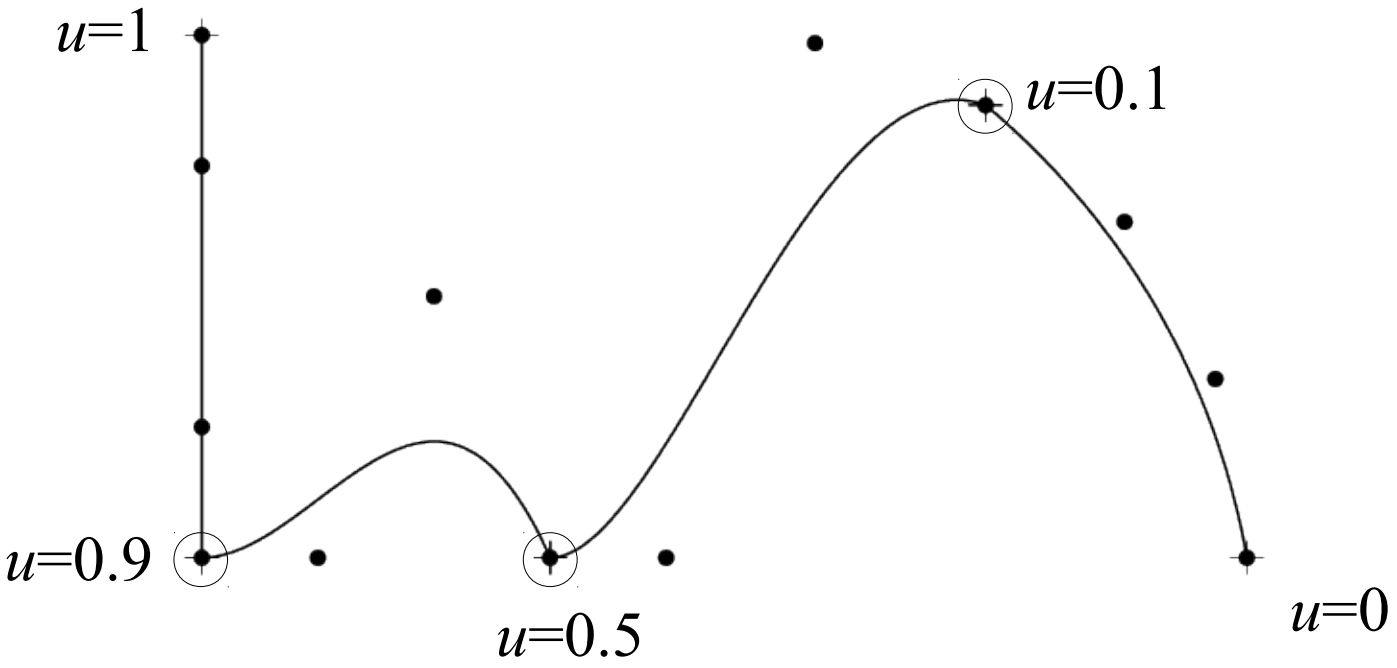
B-Splines



degree 3

0 0 0 0 0.1 0.2 0.5 0.5 0.5 0.7 0.9 0.9 0.9 1 1 1 1

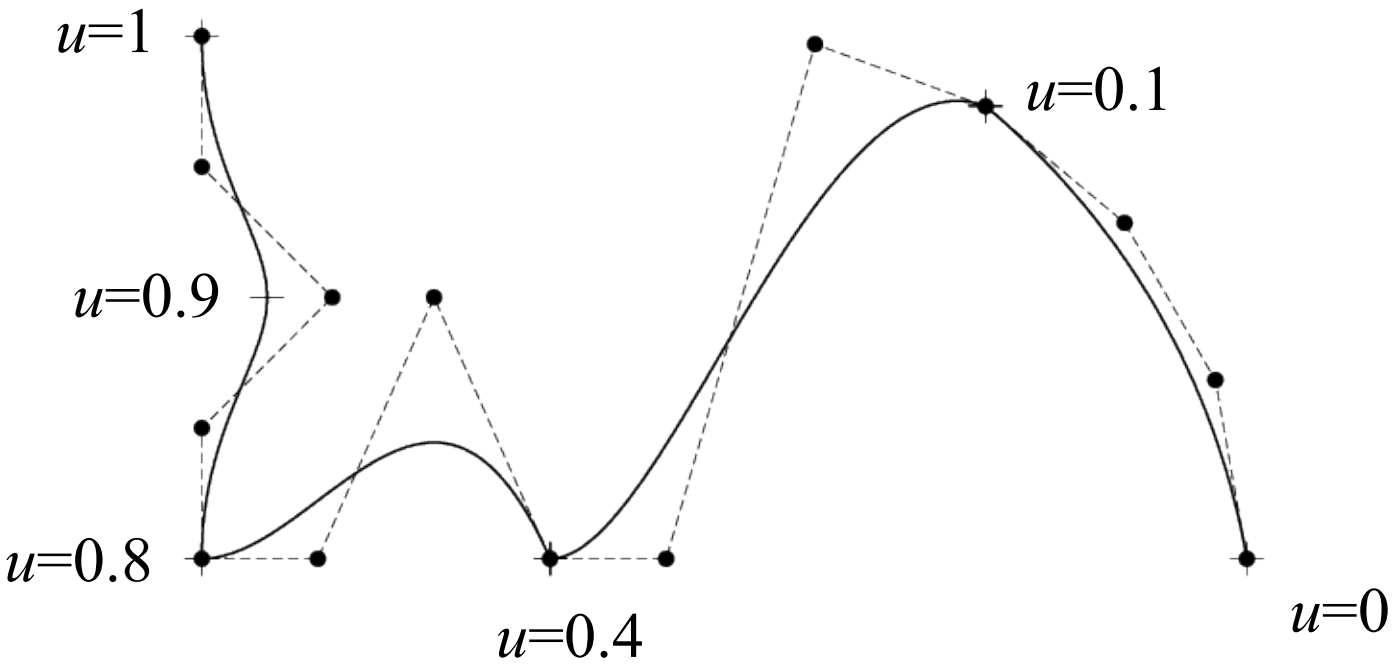
B-Splines



degree 3 (4 Bézier curves of continuity C_0)

0 0 0 0 0.1 0.1 0.1 0.5 0.5 0.5 0.9 0.9 0.9 1 1 1 1

B-Splines

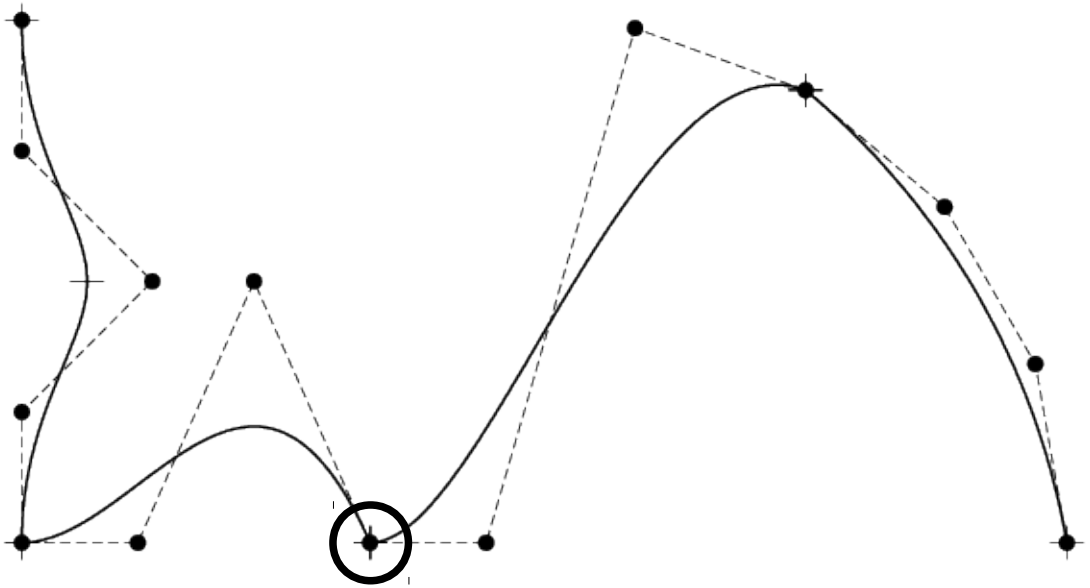


degree 3 (3 Bézier curves of continuity C_0 + 1 bspline deg 3 with 4control pts)

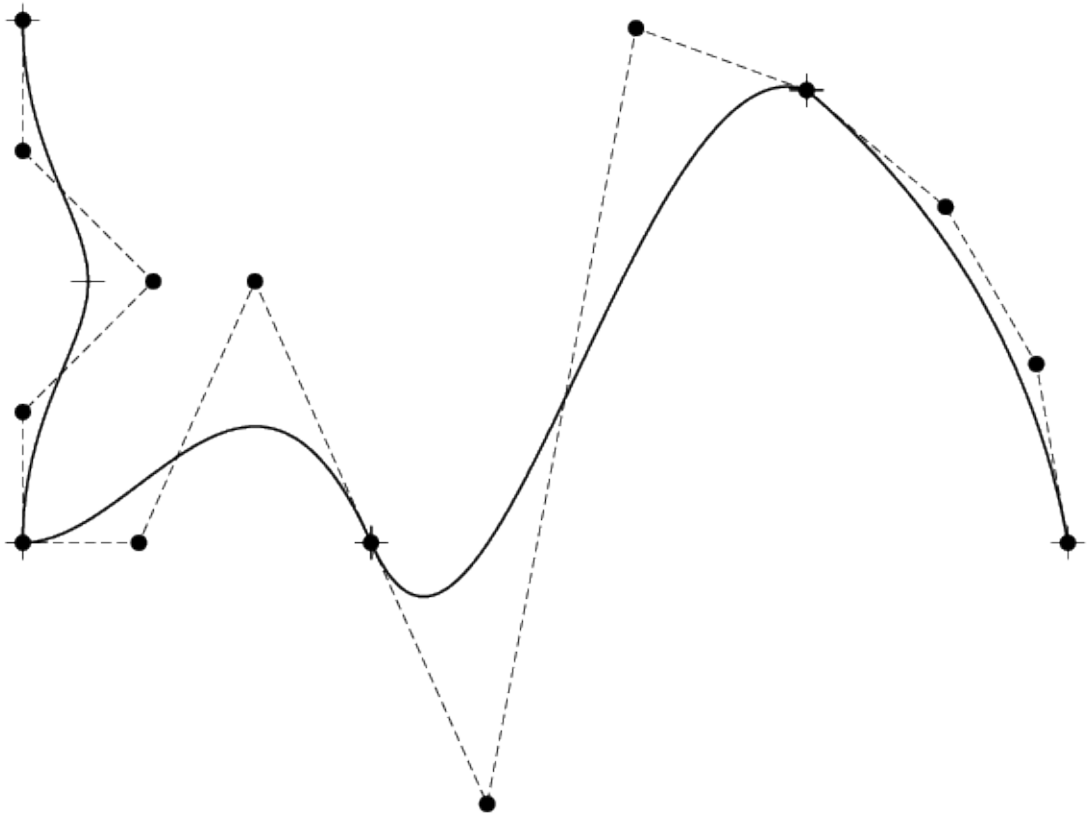
0 0 0 0 0.1 0.1 0.1 0.4 0.4 0.4 0.8 0.8 0.8 0.9 1 1 1 1

B-Splines

- And if we want to impose interpolation points ***and*** a certain continuity C_k ?
  - Add / align control points in a similar way than in the case of Bézier curves.

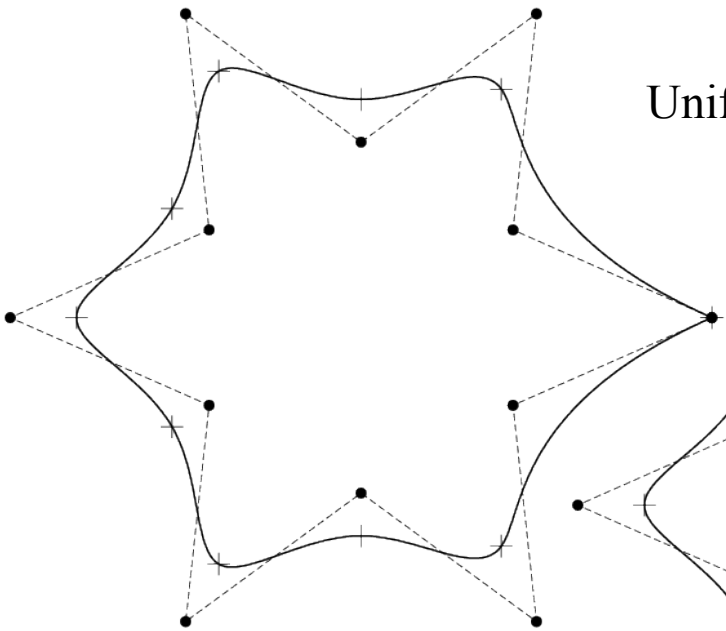


B-Splines



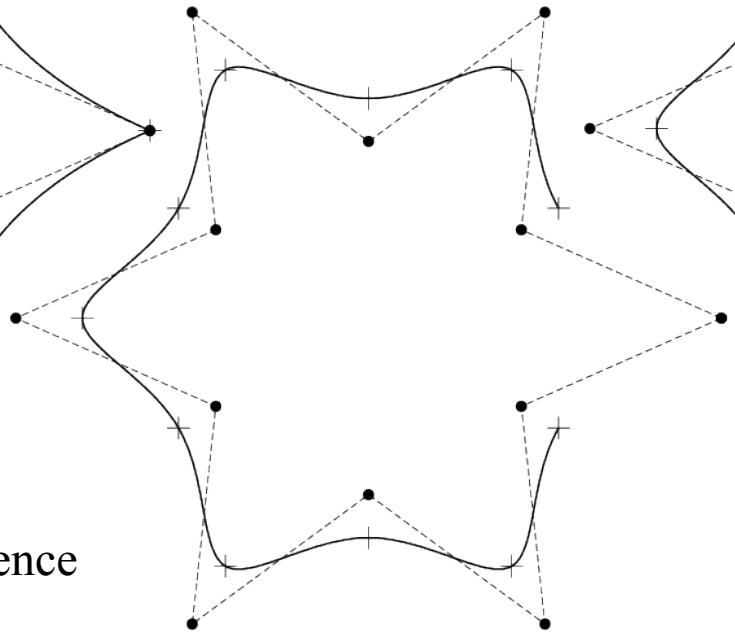
B-Splines

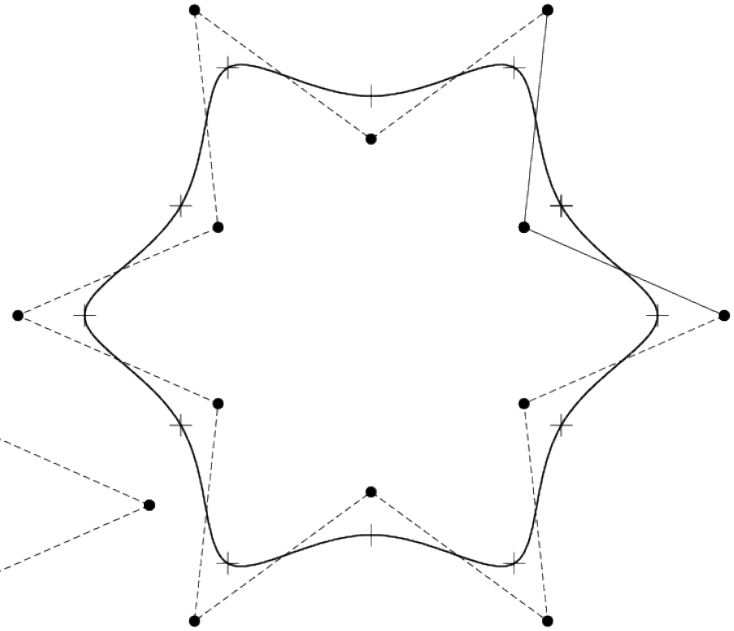
- Periodic curves
  - They may be represented by modifying the nodal sequence and by repeating some control points.



Non-uniform nodal sequence

Uniform nodal sequence

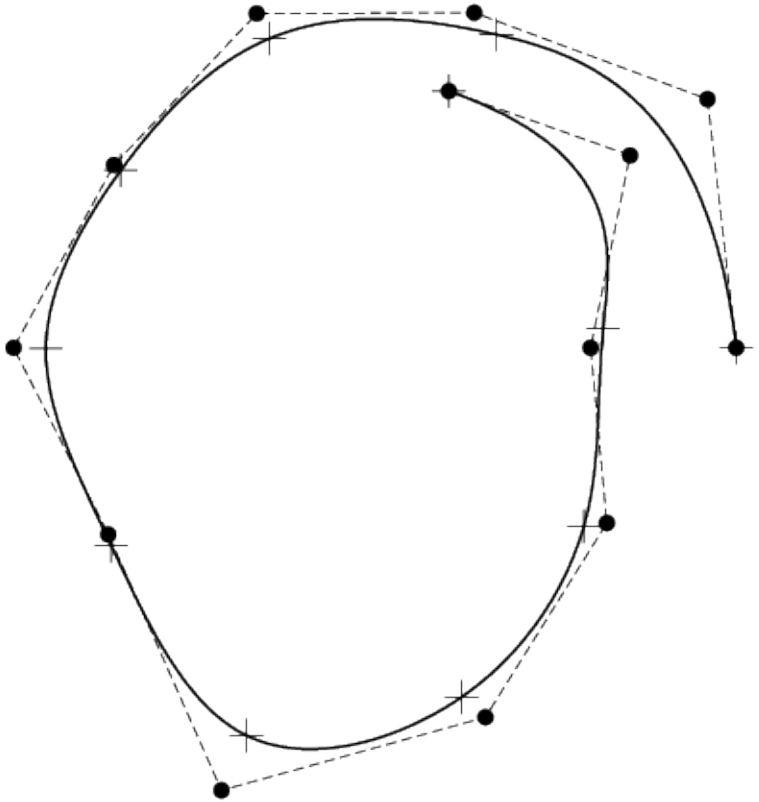




uniform nodal sequence and periodic curve

B-Splines

non uniform nodal sequence interpolating the first and last control points.

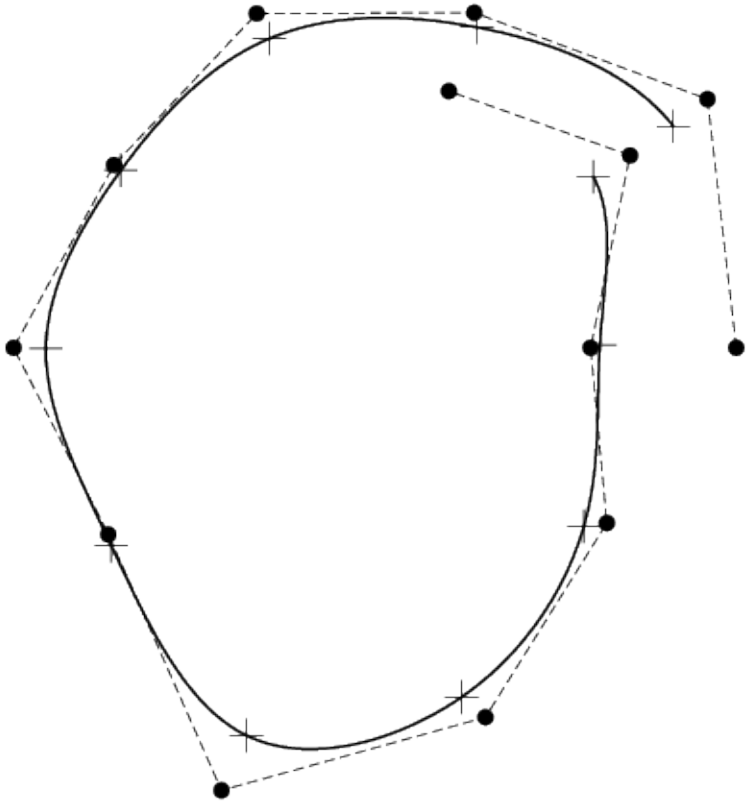


degree 3

0 0 0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1

B-Splines

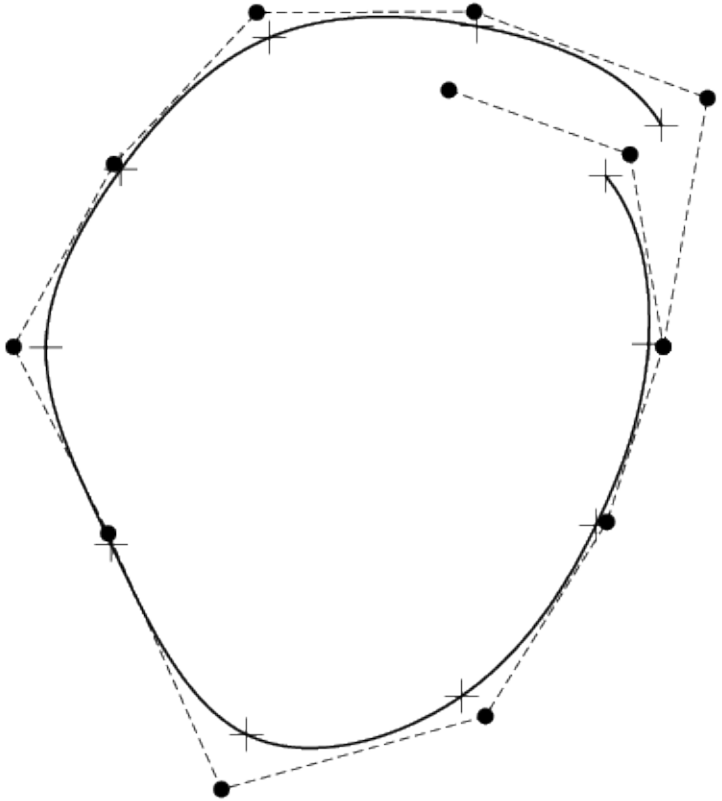
Periodic nodal sequence
(but control points located
in a non adequate way)



degree 3

-0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3

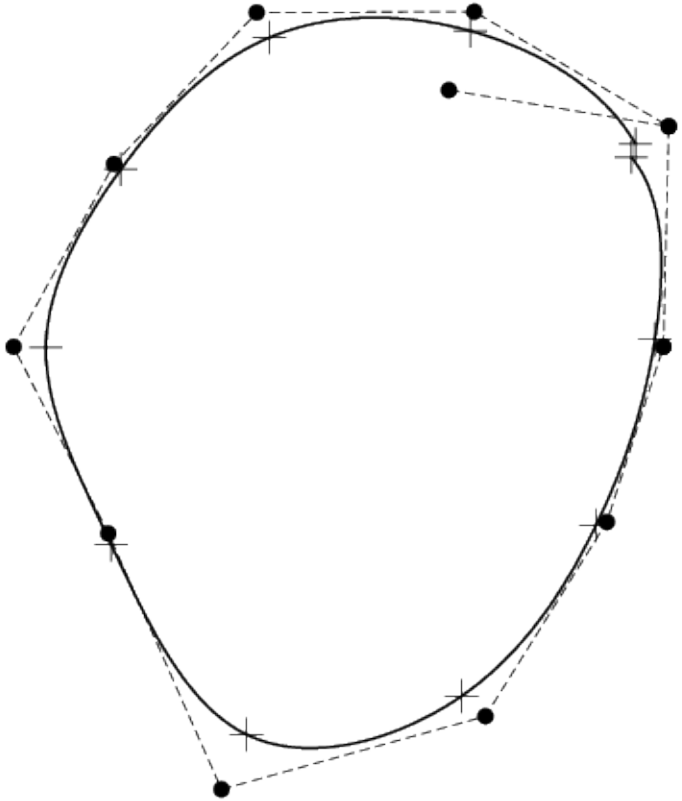
B-Splines



degree 3

-0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3

B-Splines

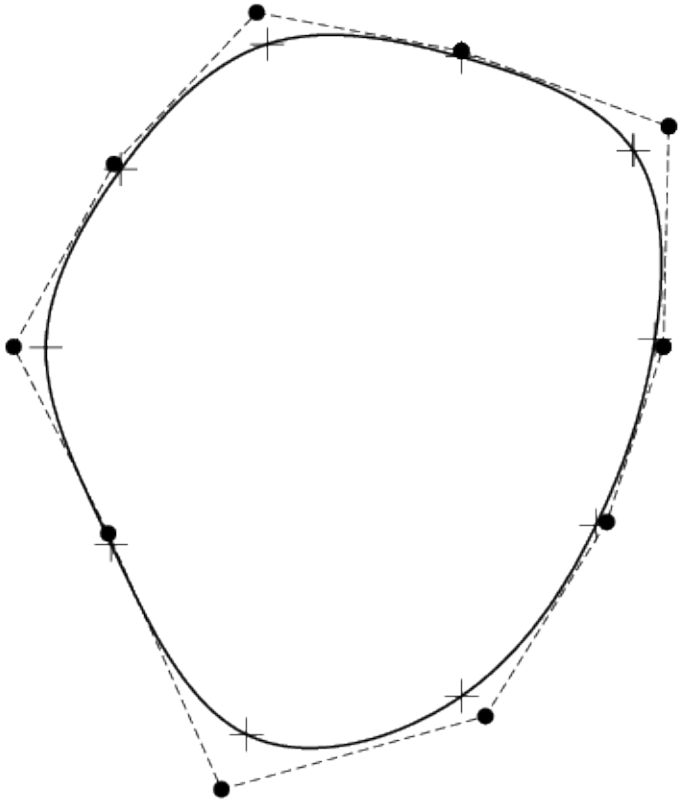


degree 3

-0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3

B-Splines

Periodic nodal sequence
+ control points placed in an adequate way (repeated)
= periodic curve



degree 3
-0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3

B-Splines

- Algorithms for the manipulation of B-Splines curves
  - Boehm's knot insertion algorithm
  - Evaluation of the curve (Cox-de Boor algorithm)
  - Derivatives and hodographs
  - Restriction/growth of the useful interval of a curve
  - Degree elevation
  - Recursive Subdivision

B-Splines

- Boehm's knot insertion algorithm

  The idea is to determine a new control polygon for the same curve after the insertion of one or several knots in the nodal sequence.

  The curve is not modified by this change : neither the shape nor the parametrization are affected.

  Interest :

  - Evaluation of points on the curve
  - Subdivision of the curve
  - Addition of control points

B-Splines

- Let $P(u)=\sum_{i=0}^{n} P_i N_i^d(u)$ a B-Spline curve built on the nodal sequence : $U=\{u_0,\cdots,u_m\}$
- Let $\bar{u}\in[u_k,u_{k+1}[$ a knot to be inserted
- The new nodal sequence is :

$$\bar{U}=\{\bar{u}_0=u_0,\cdots,\bar{u}_k=u_k,\bar{u}_{k+1}=\bar{u},\cdots,\bar{u}_{m+1}=u_m\}$$

- The new representation of the curve is :

$$P(u)=\sum_{i=0}^{n+1} Q_i \bar{N}_i^d(u)$$

  - The $\bar{N}_i^d(u)$ are the basis functions defined on $\bar{U}$, the Q_i are the $n+2$ new control points.
  - How define the Q_i so that the shape is unchanged ?

B-Splines

After algebraic manipulations... we obtain

$$Q_{k-d} = P_{k-d}$$
$$Q_i = \alpha_i P_i + (1-\alpha_i) P_{i-1} \text{ for } i \in \{k-d+1, \cdots, k\}$$
$$Q_{k+1} = P_k$$

We had : $P_i = Q_i$ for $i \in \{0, \cdots, k-d-1\}$
$P_i = Q_{i+1}$ for $i \in \{k+1, \cdots, n\}$

so finally :

$$Q_i = \alpha_i P_i + (1-\alpha_i) P_{i-1} \quad \text{with } \alpha_i = \begin{cases} 1 & i \le k-d \\ \dfrac{\bar{u}-u_i}{u_{i+p}-u_i} & k-d+1 \le i \le k \\ 0 & i \ge k+1 \end{cases}$$

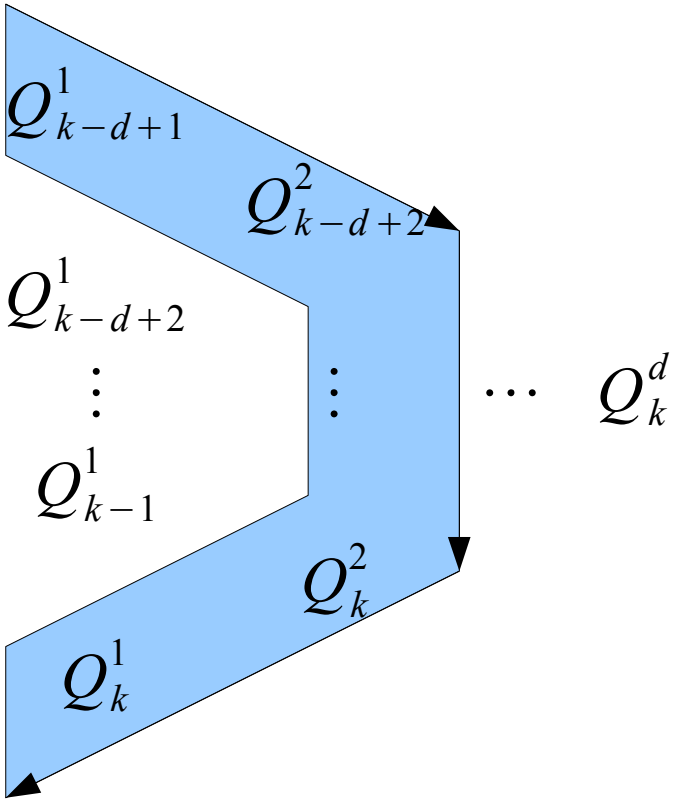
B-Splines

- Multiple knot insertions
  - Assume $\bar{u} \in [u_k, u_{k+1}[$ of multiplicity s $(0 \le s < d)$. We want to insert it r times with $r + s \le d$.
  - We note Q_i^r the control points of the r-th insertion step
  - We have then :

$$Q_i^r = \alpha_i^r Q_i^{r-1} + (1 - \alpha_i^r) Q_{i-1}^{r-1} \quad \text{with } \alpha_i^r = \begin{cases} 1 & i \le k - d + r - 1 \\ \dfrac{\bar{u} - u_i}{u_{i+d-r+1} - u_i} & k - d + r \le i \le k - s \\ 0 & i \ge k - s + 1 \end{cases}$$

B-Splines

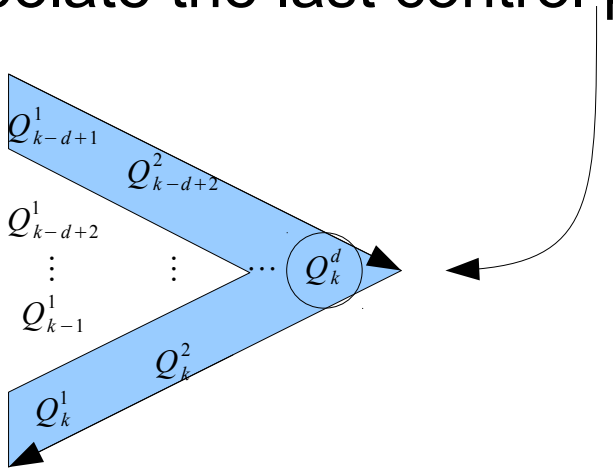
- The Q's can be put in a table:



- The total number of new control points is d-s+r-1 that replace d-s-1.

B-Splines

- The use of the algorithm of node insertion up to multiplicity of $d=r+s$ is such that the curve will interpolate the last control point that is computed.



- Therefore, one can use this algorithm to compute the position of a point of the curve knowing the parameter.
  - It's precisely Cox-de Boor's algorithm. The sequence of points P_i^j is not anything else than the Q_i^j indicated on the graph, cf following

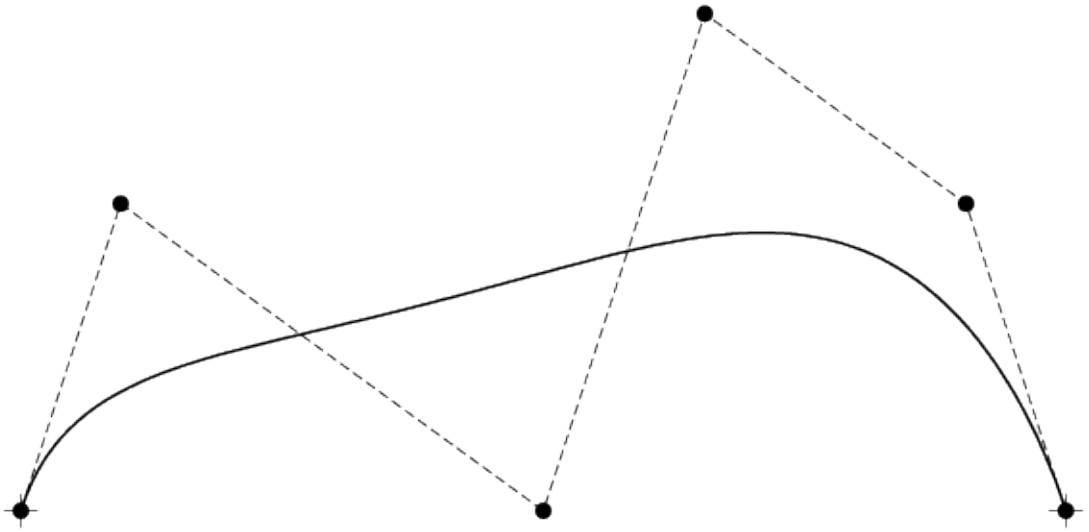
B-Splines

- Case $r+s=d+1$: We carry out the insertion of multiplicity r-1 then we insert one more knot to « cut » the B-spline curve in two independent parts.
- The last control point Q_k^d has to be duplicated.
- Allows to extract a portion of the B-spline.

- There exists an extension of this algorithm in the case of the simultaneous insertion of many knots: it is the somewhat more complex “Oslo” algorithm*

* E. Cohen, T. Lyche, R. Riesenfeld “Discrete B-splines ans subdivision techniques in computer-aided geometric design and computer graphics”, *Computer Graphics and Image Processing*, **14**(2):87-111, 1980.

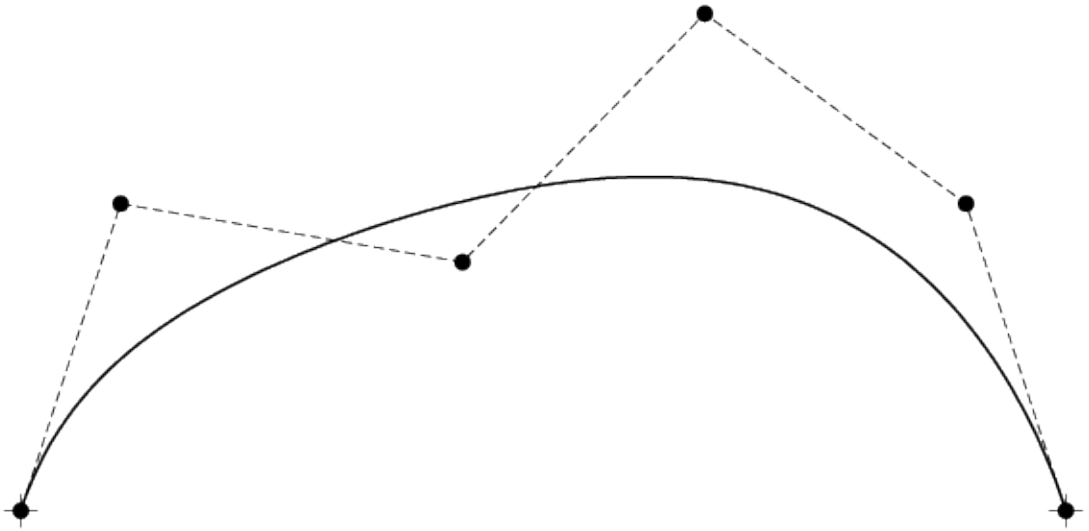
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1\}$$

degree 5

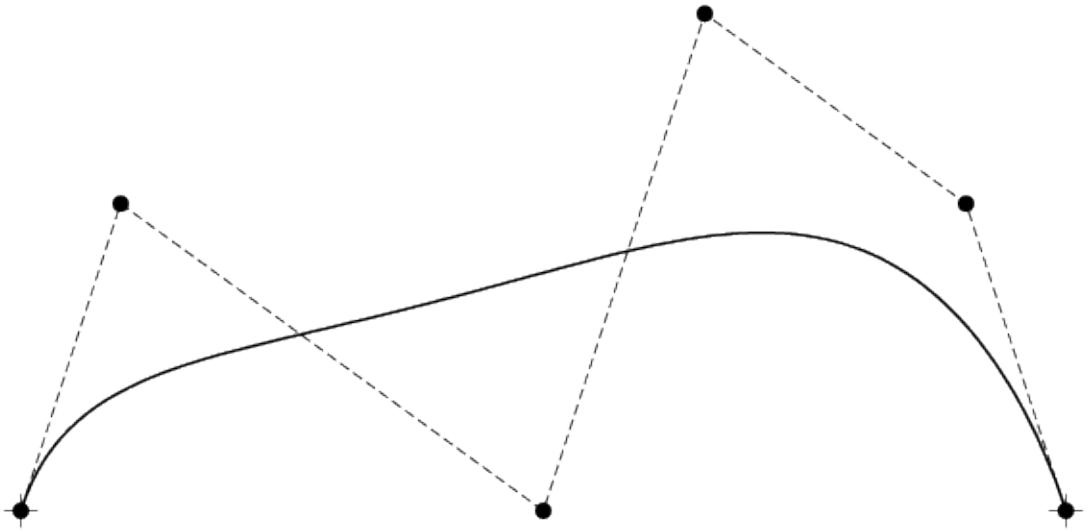
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1\}$$

degree 5

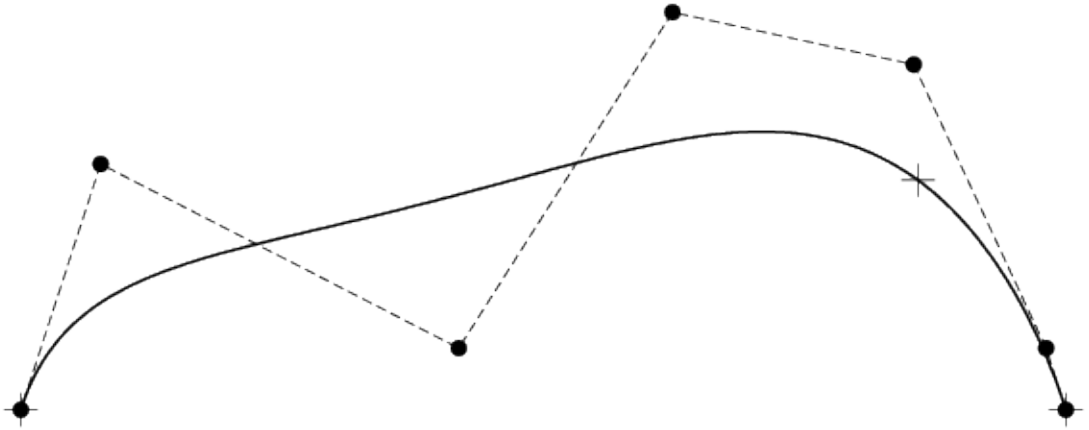
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1\}$$

degree 5

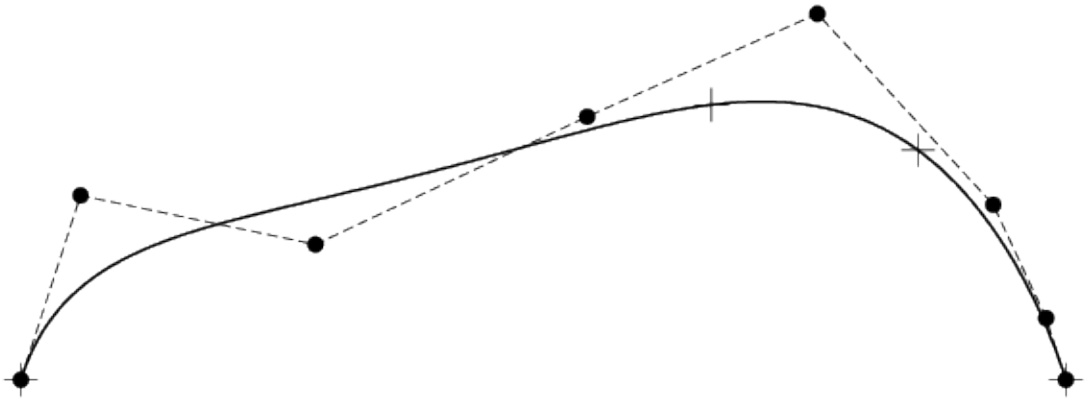
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.2, 1, 1, 1, 1, 1, 1\}$$

degree 5

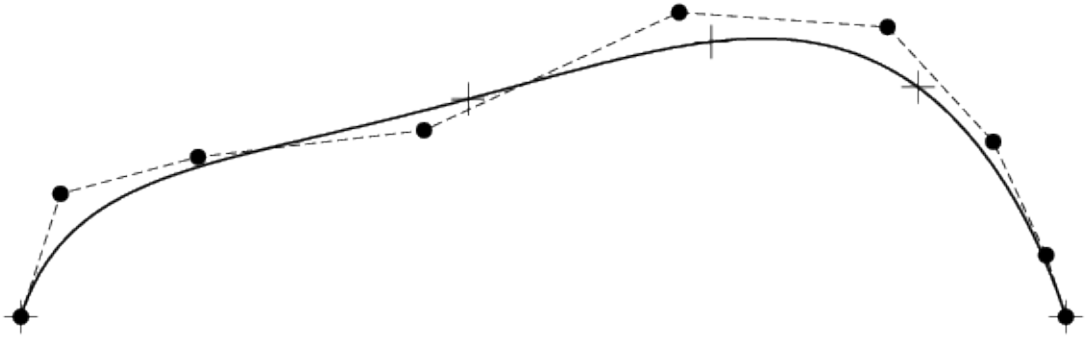
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.2, 0.4, 1, 1, 1, 1, 1, 1\}$$

degree 5

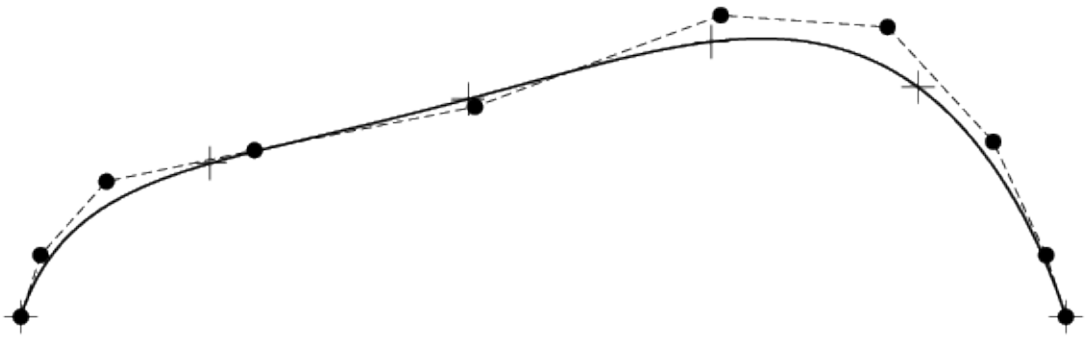
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.2, 0.4, 0.6, 1, 1, 1, 1, 1, 1\}$$

degree 5

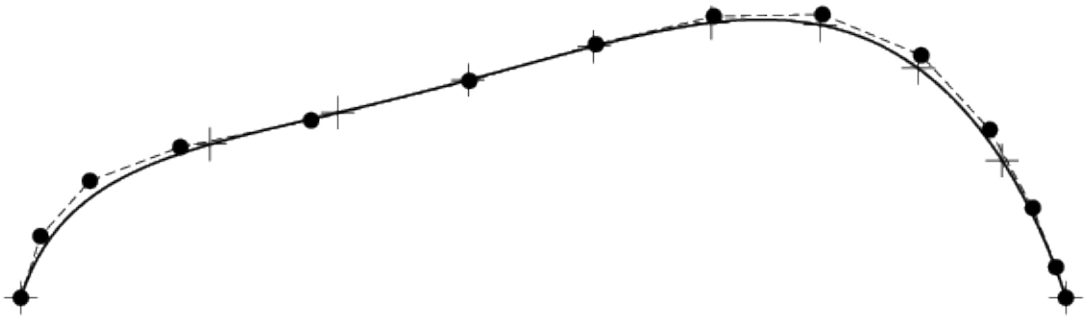
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1, 1, 1\}$$

degree 5

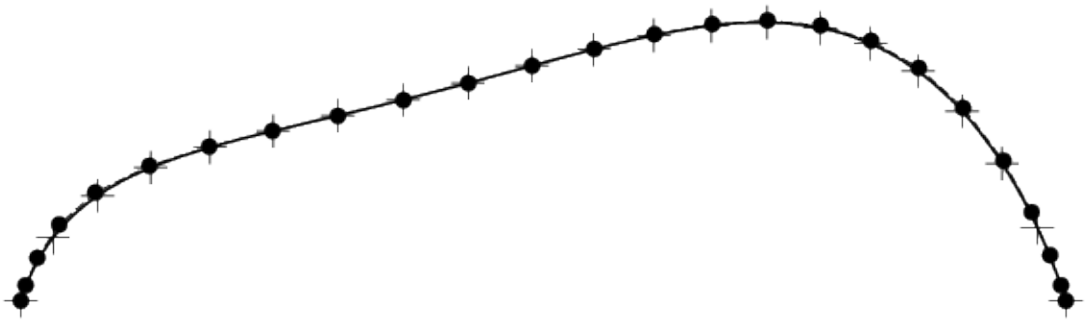
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.1, 0.2, \cdots, 0.9, 1, 1, 1, 1, 1, 1\}$$

degree 5

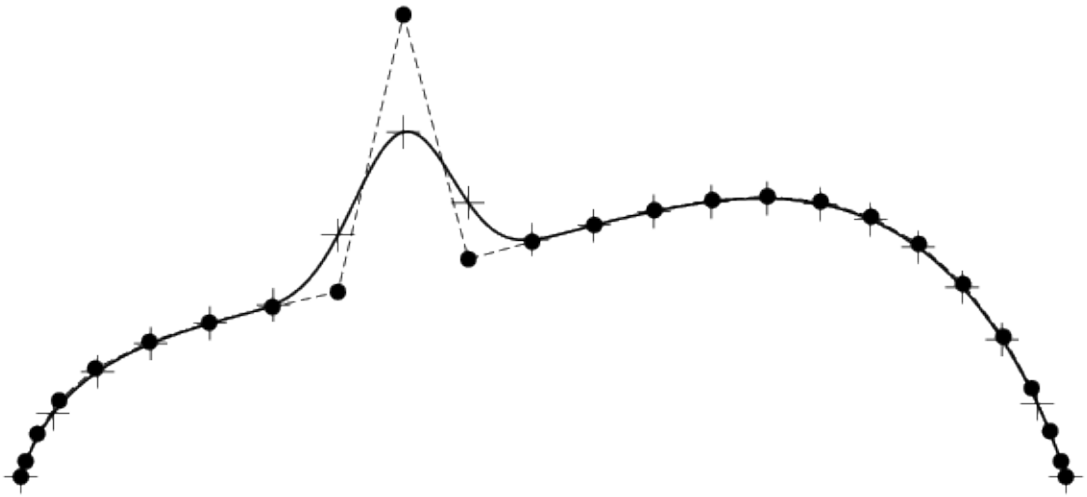
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.05, 0.1, \cdots, 0.95, 1, 1, 1, 1, 1, 1\}$$

degree 5

B-Splines

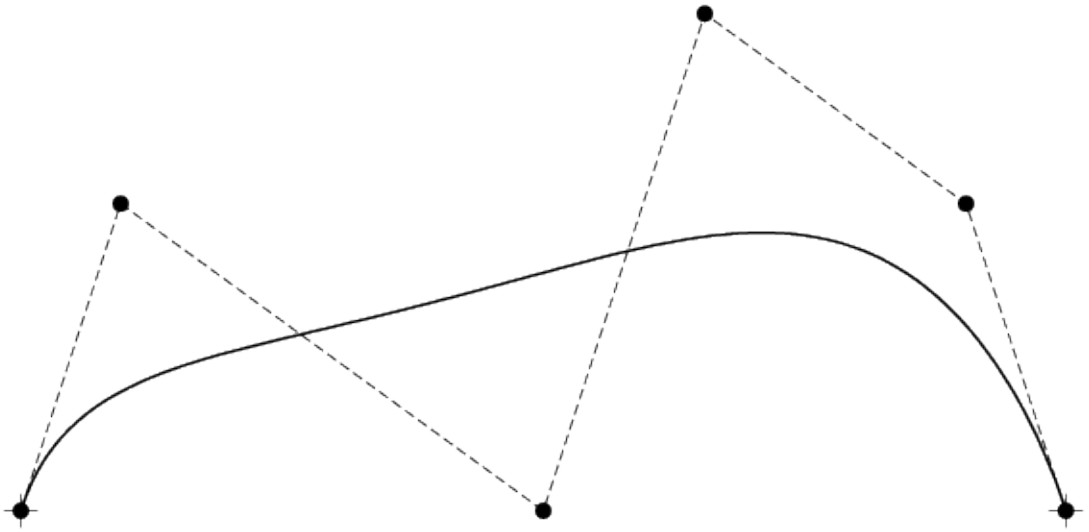


$$U = \{0, 0, 0, 0, 0, 0, 0.05, 0.1, \cdots, 0.95, 1, 1, 1, 1, 1, 1\}$$

degree 5

Local control...

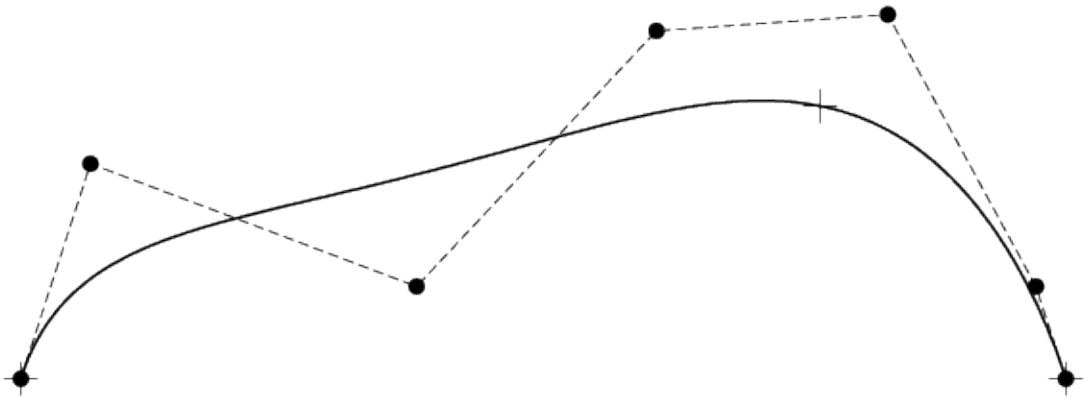
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1\}$$

degree 5

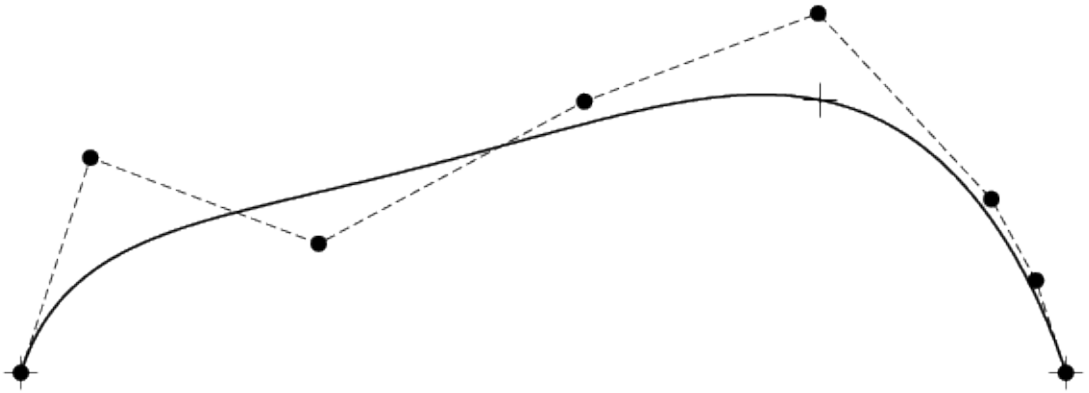
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

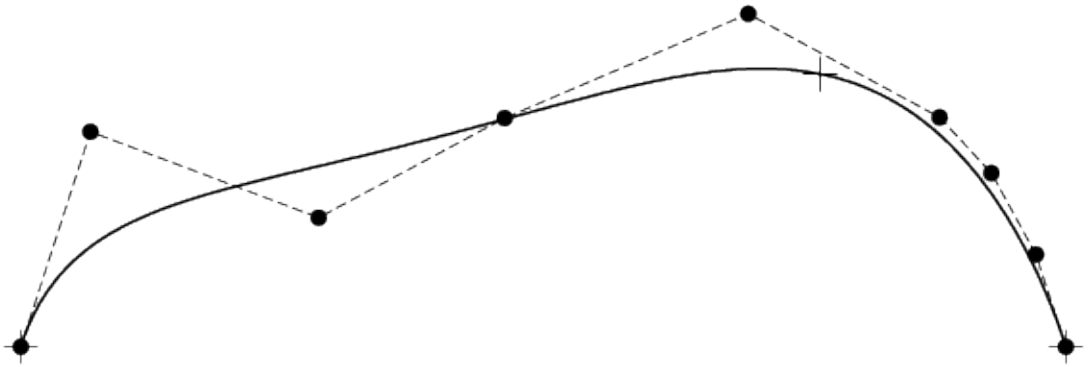
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

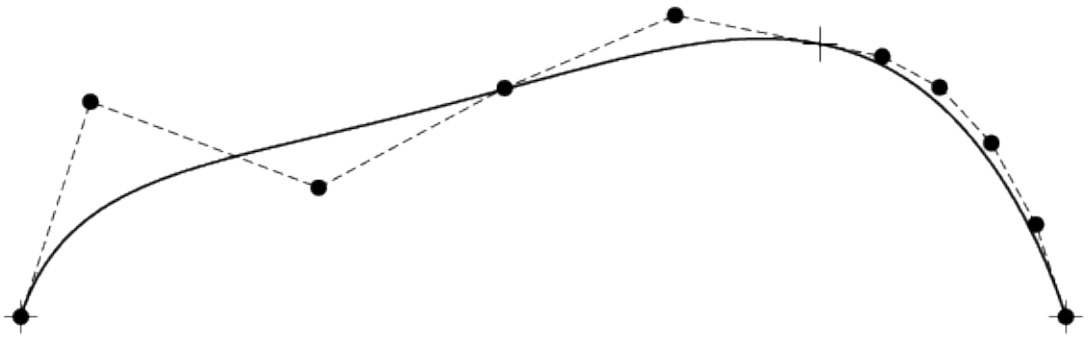
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 0.3, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

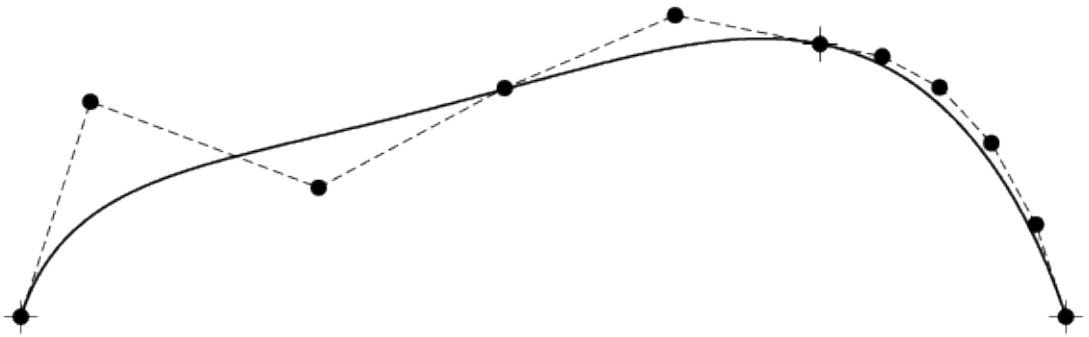
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 0.3, 0.3, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

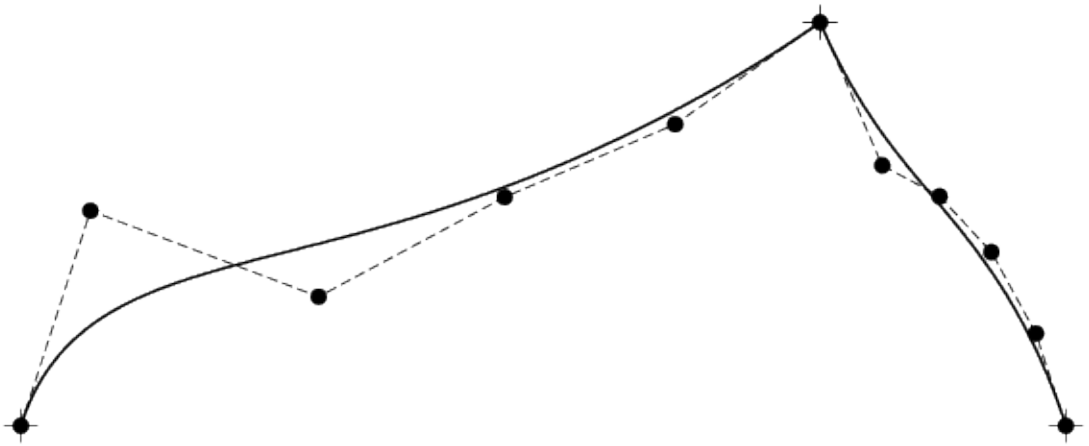
B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 0.3, 0.3, 0.3, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

B-Splines



$$U = \{0, 0, 0, 0, 0, 0, 0.3, 0.3, 0.3, 0.3, 0.3, 1, 1, 1, 1, 1, 1\}$$

degree 5

B-Splines

- Computation of a point on a B-Spline curve
  - By the use of basis functions

1 – Find the nodal interval in which u is located

$$u \in [u_i, u_{i+1}[$$

2 – Calculate the non vanishing basis functions

$$N_{i-d}^d(u), \cdots, N_i^d(u)$$

3 – Multiply the values of these basis functions with the right control points

$$P(u) = \sum_k N_k^d(u) P_k \qquad i-d \le k \le i$$

  - By Cox-de Boor's algorithm

B-Splines

- (simplified) Cox-de Boor's Algorithm :

Determine the interval of u : $u \in [u_i, u_{i+1}[$

Initialization of P_j^0 $i \in \{d, d+1, \cdots, m-d-1\}$

For k from 1 to d

  For j from i to i-d+k

$$P_j^k = \left(\frac{u - u_j}{u_{j+d+1-k} - u_j} \right) P_j^{k-1} + \left(\frac{u_{j+d+1-k} - u}{u_{j+d+1-k} - u_j} \right) P_{j-1}^{k-1}$$

  Endfor

Endfor

P_i^d is the point that is sought.

- What is its complexity ?
  - quadratic in function of the degree d.

B-Splines

- Example of computation

$$P_0^0=(0,1)\quad P_1^0=(2,3)\quad P_2^0=(5,4)\quad P_3^0=(7,1)\quad P^0{}_4=(6,-1)\quad P_5^0=(6,-2)$$

$$U=\{0,0,0,0,1,2,3,3,3,3\}\qquad d=3\qquad u=3/2$$

  - Determination of the interval

    $1\le 3/2<2\quad, u_4=1\ \rightarrow\ i=4$

$$P_j^k=\left(\frac{u-u_j}{u_{j+d+1-k}-u_j}\right)P_j^{k-1}+\left(\frac{u_{j+d+1-k}-u}{u_{j+d+1-k}-u_j}\right)P_{j-1}^{k-1}$$

  - Iteration 1

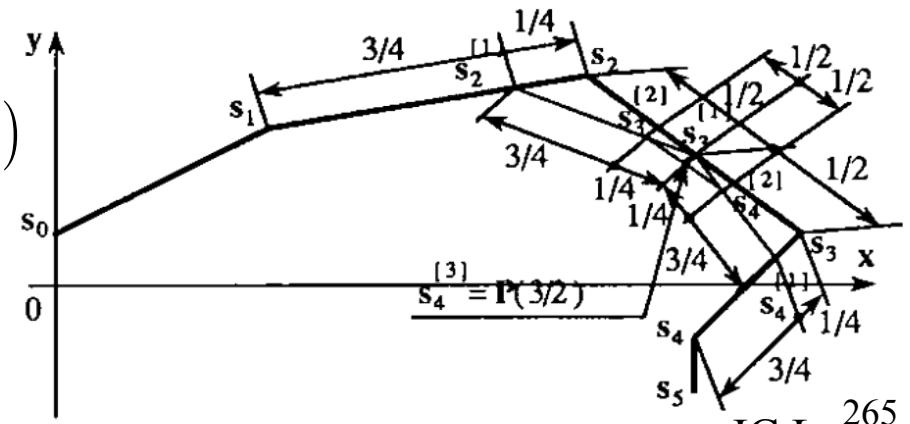
$$P_4^1=(27/4,1/2)\quad P_3^1=(6,5/2)\quad P_2^1=(17/4,15/5)$$

  - Iteration 2

$$P_4^2=(99/16,2)\quad P_3^2=(89/16,45/16)$$

  - Iteration 3

$$P_4^3=(47/8,77/32)=P(3/2)$$



B-Splines

- The algorithm is similar to De Casteljau's algorithm for Bézier curves
  - It is built on a restriction of the set of control points ($d+1$ points)
  - On this restriction, it is identical, except for the coefficients related to the nodal sequence (which is potentially non uniform)
  - The complete algorithm is somewhat longer than this one (possibility to have $0/0$: we set conventionally $0/0 = 0$!)

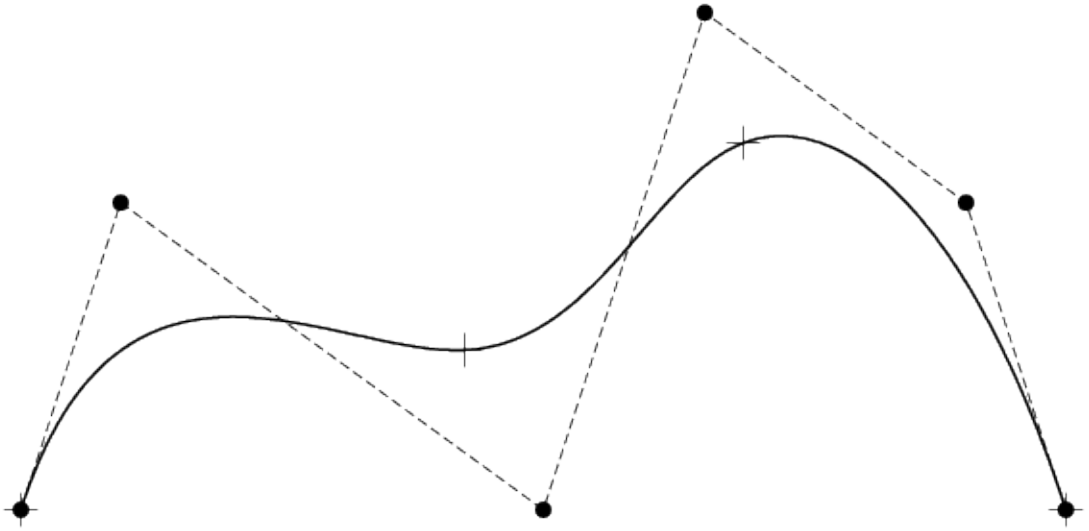
B-Splines

- Transformation of a B-Spline curve into a composite Bézier curve
  - We saturate each distinct knot until its multiplicity is equal to d.
  - This is made with the help of Boehm's algorithm of nodal insertion.
    - The curve is not modified !
  - We obtain a nodal sequence which has the following form :

$$U = \{\underbrace{a, a, a, a}_{d+1 \text{ times}}, \underbrace{b, b, b}_{d \text{ times}}, \underbrace{c, c, c}_{d \text{ times}}, \cdots, \underbrace{z, z, z, z}_{d+1 \text{ times}}\}$$

    - Each distinct value of u corresponds to one of the points of the curve.

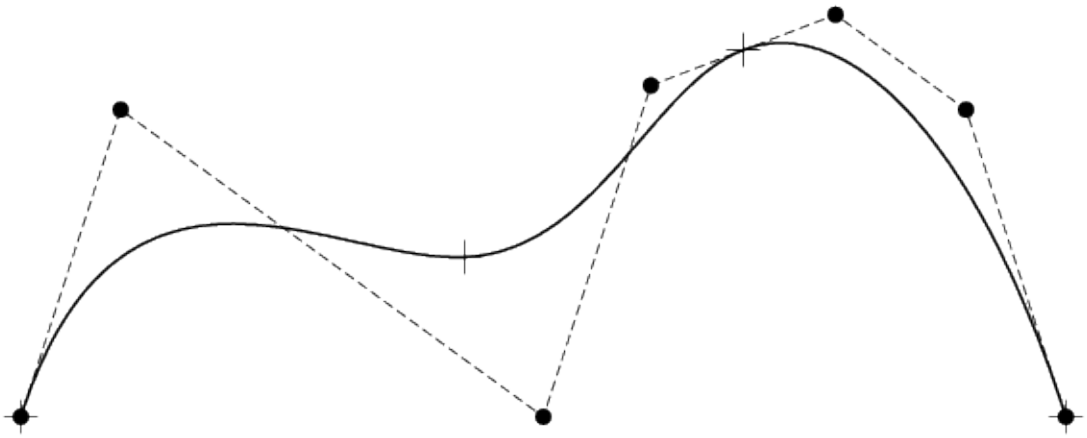
B-Splines



$$U = \{0, 0, 0, 0, 1, 2, 3, 3, 3, 3\}$$

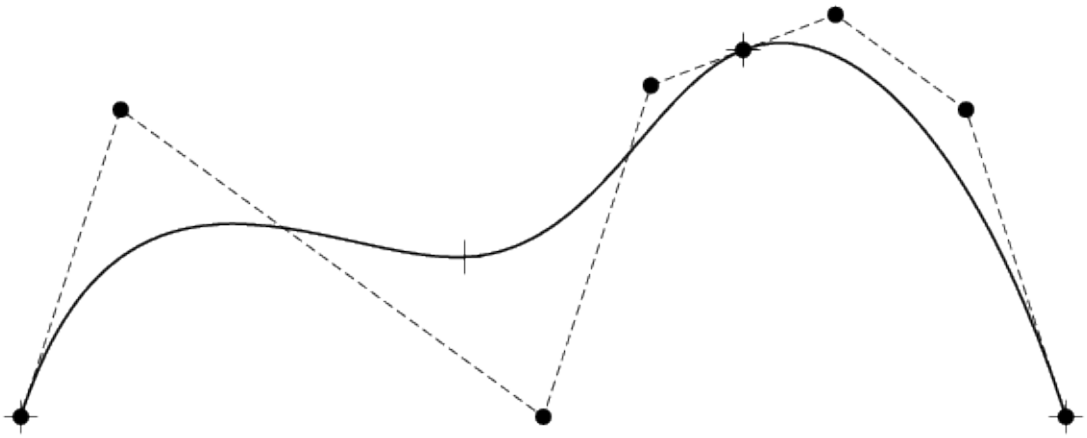
degree 3

B-Splines



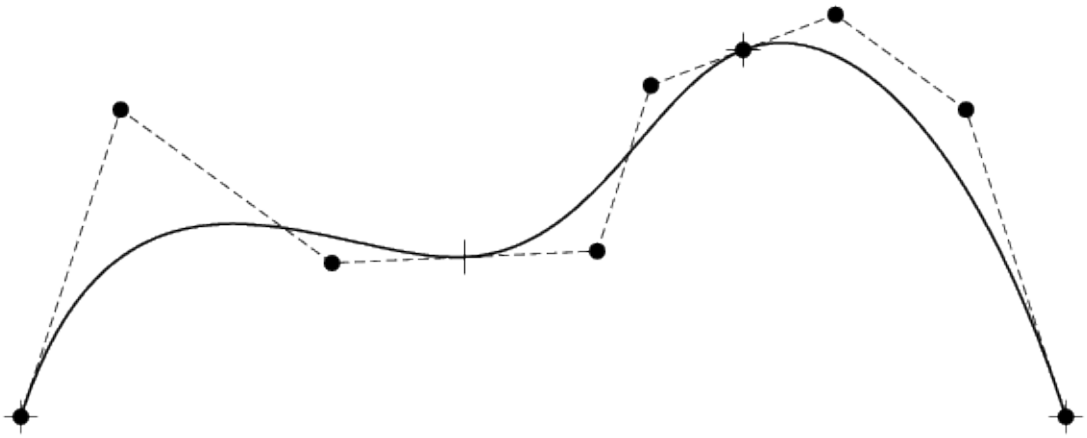
$$U = \{0, 0, 0, 0, 1, 1, 2, 3, 3, 3, 3\}$$

B-Splines



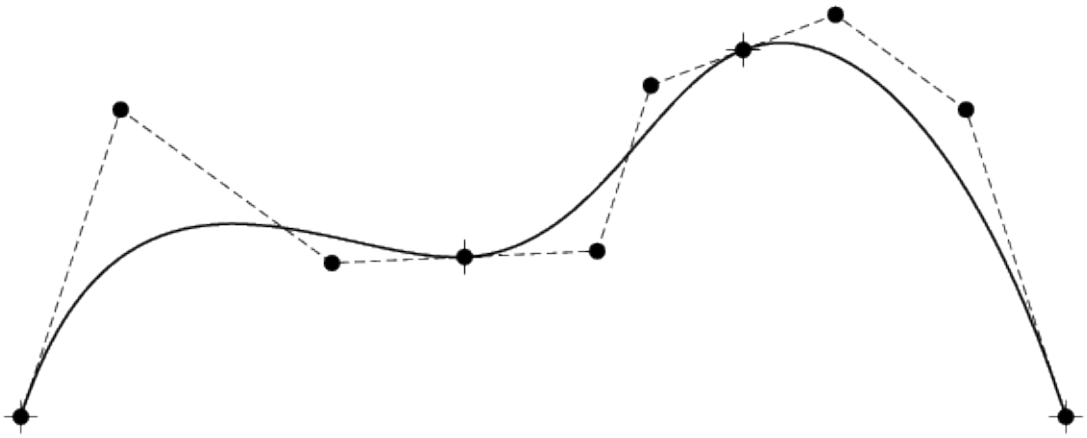
$$U = \{0, 0, 0, 0, 1, 1, 1, 2, 3, 3, 3, 3\}$$

B-Splines



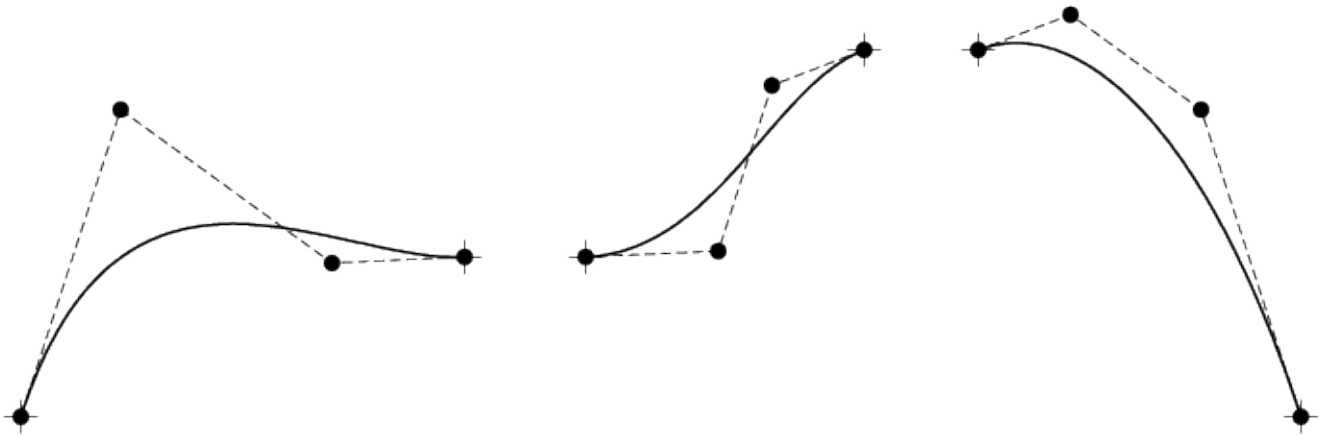
$$U = \{0, 0, 0, 0, 1, 1, 1, 2, 2, 3, 3, 3, 3\}$$

B-Splines



$$U = \{0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3\}$$

B-Splines



Bézier n°3
4 CP
degree 3

Bézier n°2
4 CP
degree 3

Bézier n°1
4 CP
degree 3

B-Splines

- Some conclusions
  - Flexibility
  - Low order
  - Continuity
  - Periodic curves
  - Conics ?