13 - Splines

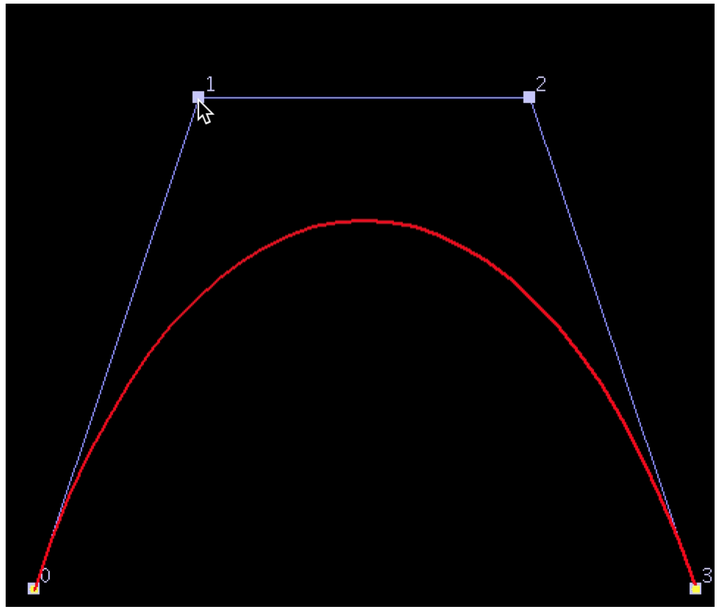
In this lecture

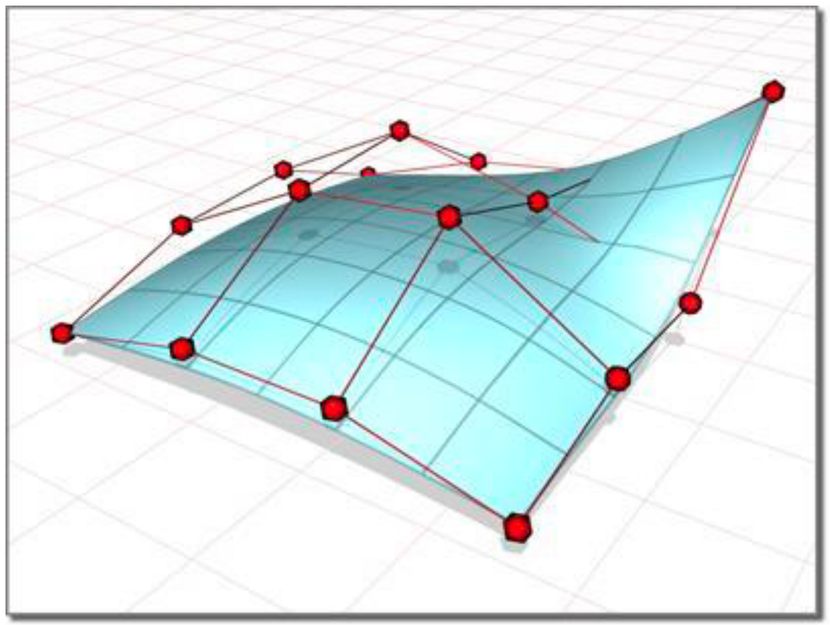
- Introduction to spline curves and surfaces

Splines

General idea:

- Smooth curve/surface in parametric form
- Defined by combining two ingredients:
  - Control polygon/grid built upon a small set of **control points**
  - **Basis functions** to compute *affine averages* of point positions
- The curve/surface can be edited by
  - Moving the control points
  - Reshaping the basis functions



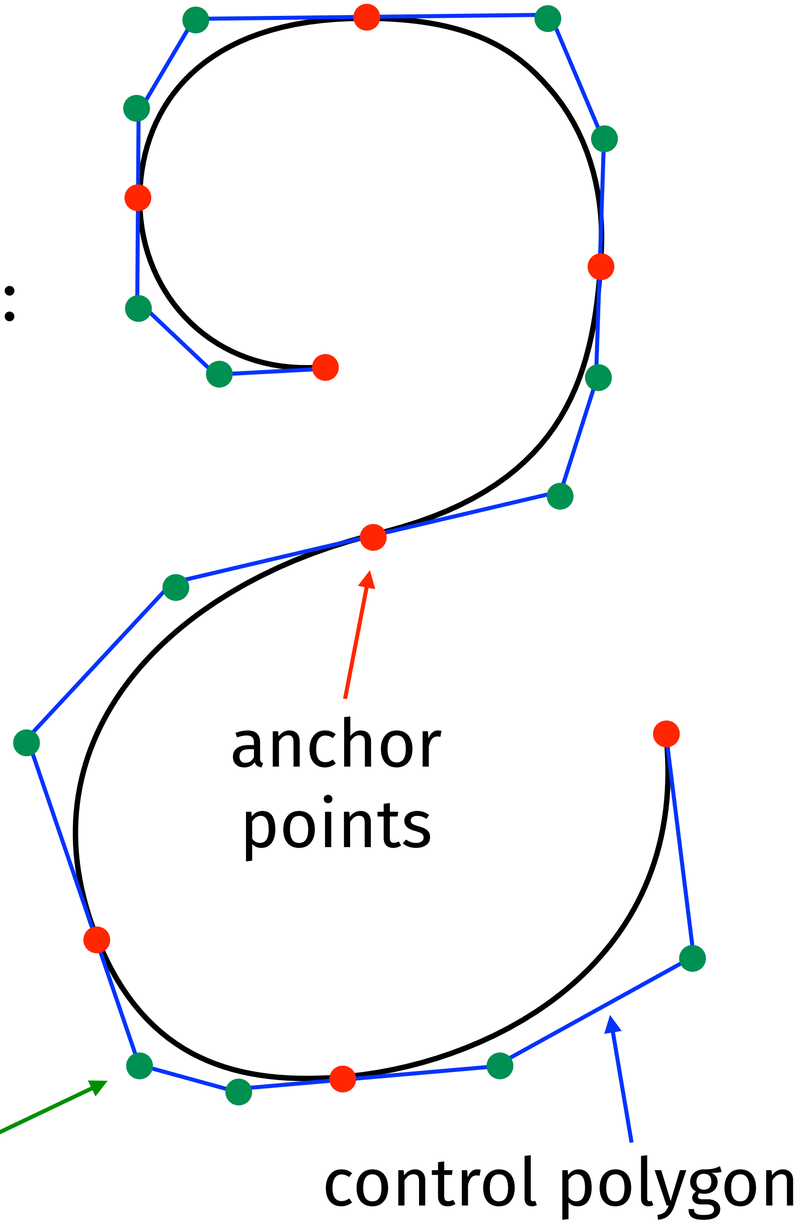


Curves

Spline curves

Informal idea:

- Control points determine the shape of the curve:
  - Anchor points are on the curve
  - Handle points “pull” the curve
- Control polygon joining the control points approximates the curve
- A spline may consist of several *segments* of curve



Spline curves

In math terms:

affine sum
blends the contributions of
the various control points

$$s(t) = \sum_i \mathbf{p}_i B_i(t)$$

the enumeration
of indices define
the control polygon

control points

base functions
weigh the influence
of control points at
each parameter value

Spline curves

- Basis functions characterize the type of spline:
  - Interpolating vs approximating
  - Smoothness (especially at junctions between segments)
  - Ease of editing (algorithms)
  - Expressive power (how many types of curve can be represented)

Bézier curves

- Bases are polynomials of given degree *n*
- A Bézier segment of degree *n* has *n+1* control points

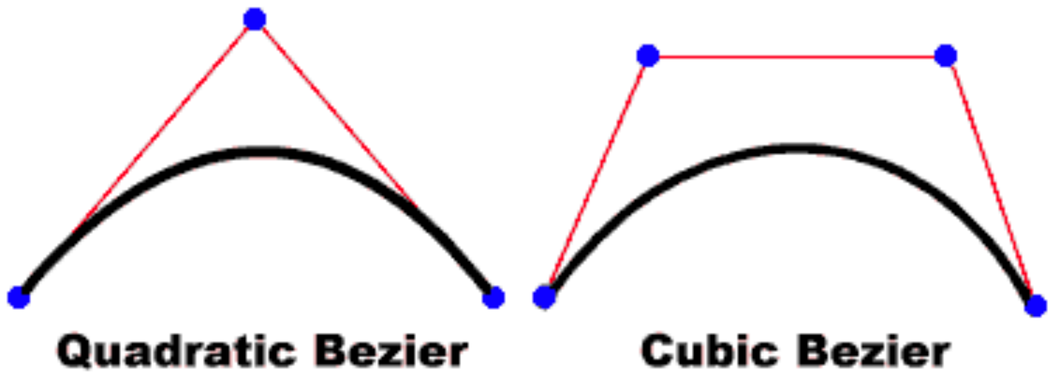
$$s(t) = \sum_{i=0}^{n} B_{i,n}(t)\mathbf{p}_i$$

- where $$B_{i,n}(t) = \frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}$$

  are the Bernstein polynomials

Bézier curves

- n=1 : straight-line segment
- n=2 : parabolic arc
- n=3 : cubic arc
- For all n the curve:
  - interpolates the endpoints of control polygon
  - Is tangent to first and last segment of the control polygon at the endpoints



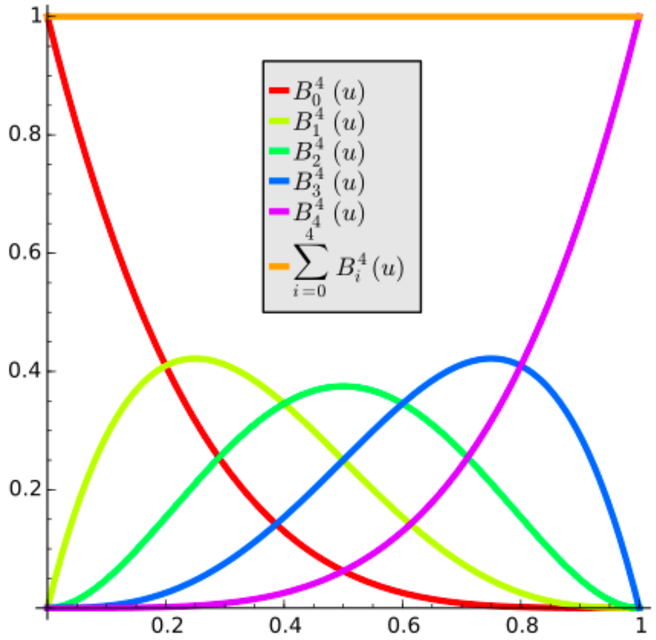
Bézier curves

Demo http://nurbscalculator.in/

Bézier curves

Properties:

- The $B_{i,n}(t)$ are non-negative in [0,1]
- Partition of unity: $\sum_{i=0}^{n} B_{i,n}(t) = 1$
- Interpolation at endpoints: $B_{0,n}(0) = B_{n,n}(1) = 1$
- All $B_{i,n}(t)$ have exactly one maximum in [0,1]
- Symmetry: $B_{i,n}(t) = B_{n-i,n}(1-t)$



Bernstein bases
of degree 5

Bézier curves

Bernstein bases

- Recursive definition: $B_{i,0}(t) = 1 \qquad i \geq 0$

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t)$$

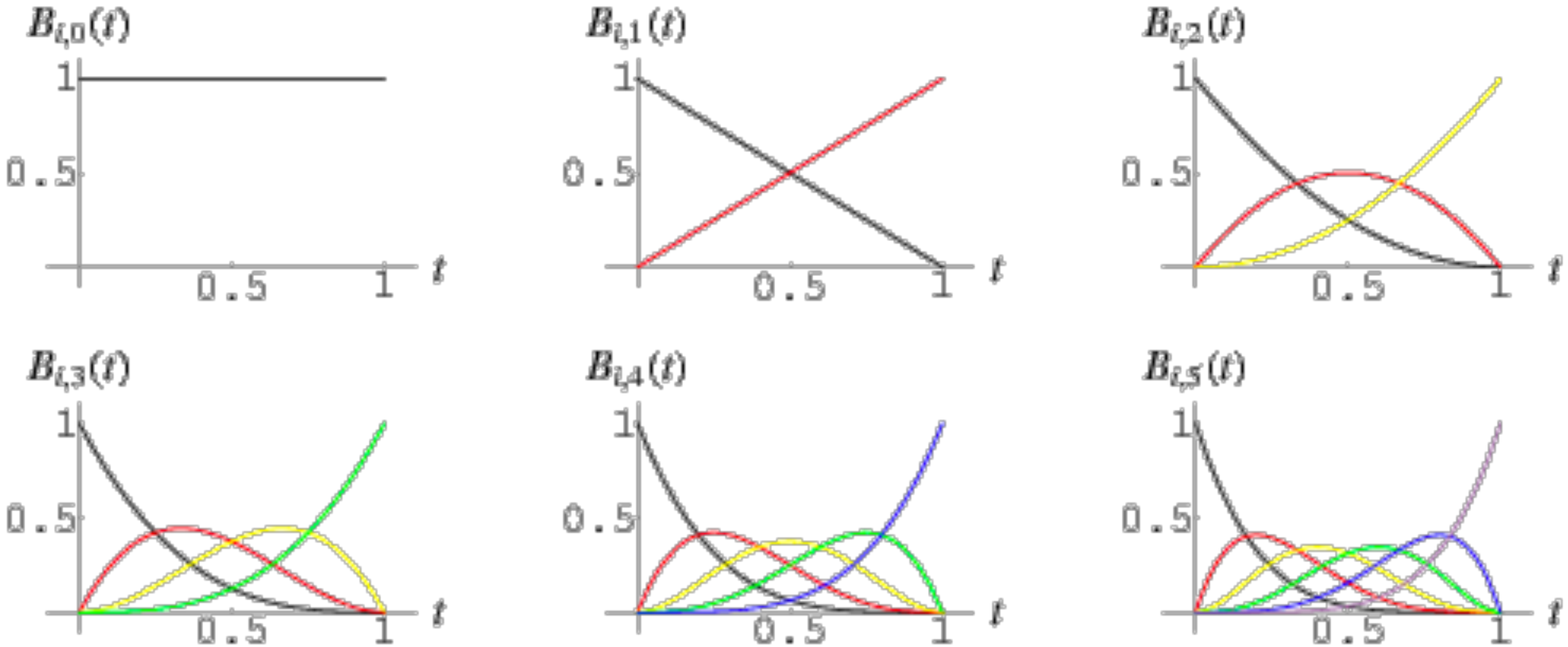
- Derivatives: $B_{i,n}^{'}(t) = t(B_{i-1,n-1}(t) - B_{i,n-1}(t))$

with $B_{-1,n}(t) = B_{n,n-1} = 0$

- Tangents at endpoints: derivatives at $\begin{matrix} t=0 \\ t=1 \end{matrix}$ depend only on $\begin{matrix} \mathbf{p}_0, \mathbf{p}_1 \\ \mathbf{p}_{n-1}, \mathbf{p}_n \end{matrix}$

Bézier curves

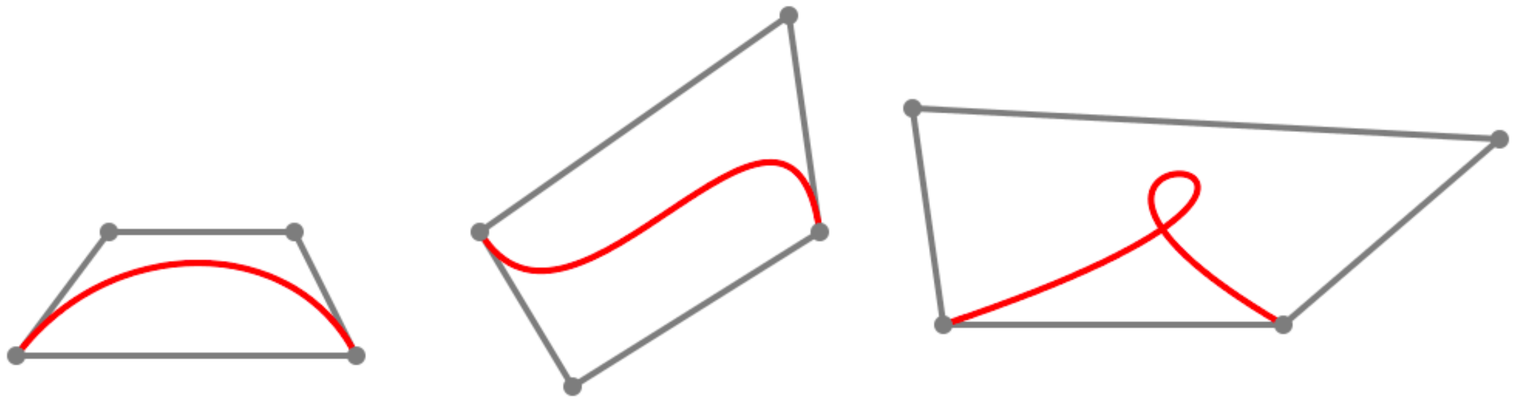
Bernstein bases



Bézier curves

- *Convex hull property*: the curve is contained in the convex hull of its control points

- *Variation diminishing property*: no straight line can have more intersection with the curve than with the control polygon



Bézier curves

Drawbacks:

- *Global support*: moving each single control point modifies the whole curve (and the change is not intuitive)
- *Numerical instability* at high degree / large number of control points
- *Low expressive power*: conics (e.g., the circle) cannot be represented

de Casteljau algorithm

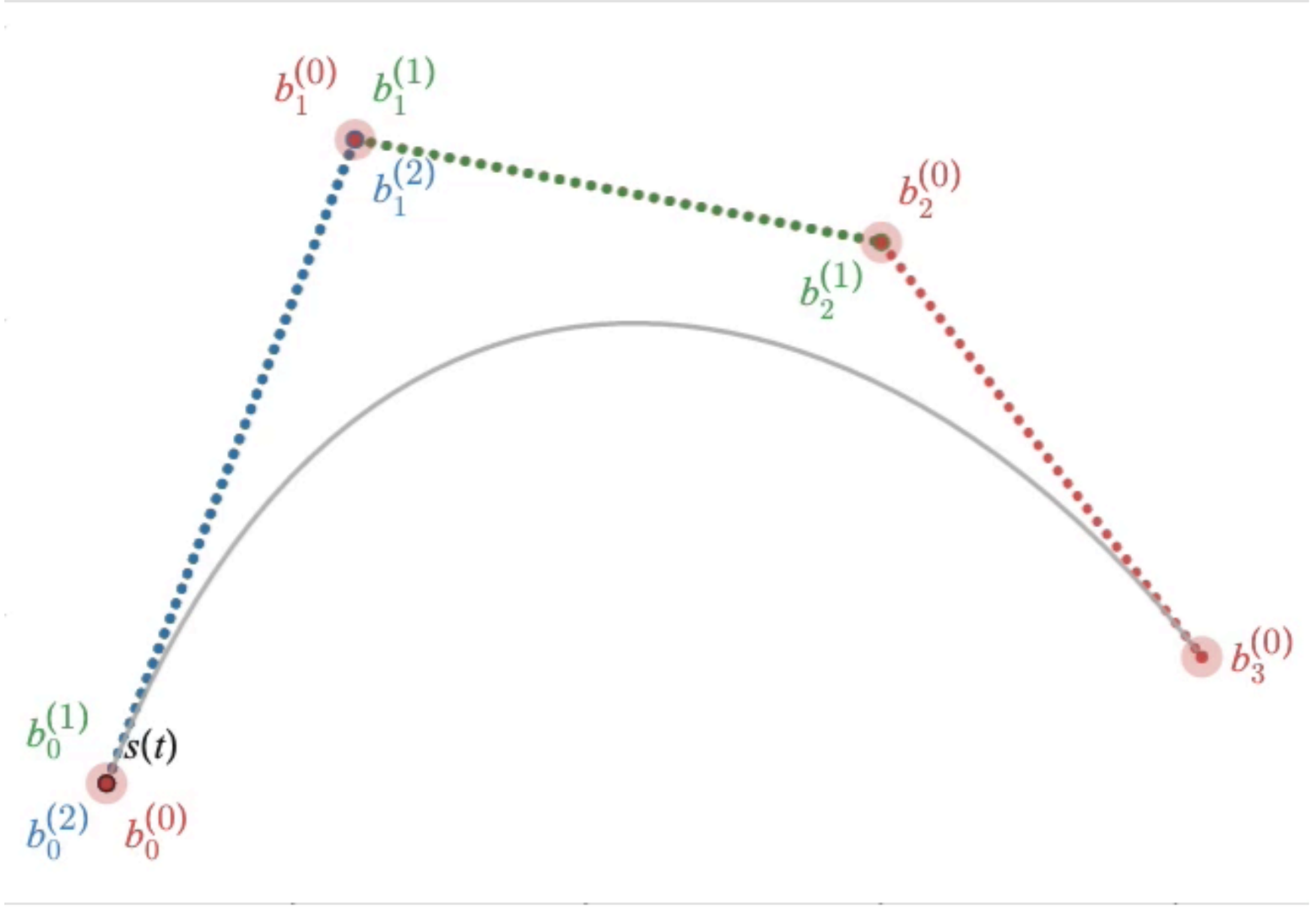
- Bézier curves admit a recursive definition:

$$\mathbf{b}_i^0(t) = \mathbf{p}_i$$
$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)$$

$$s(t) = \mathbf{b}_0^n(t)$$

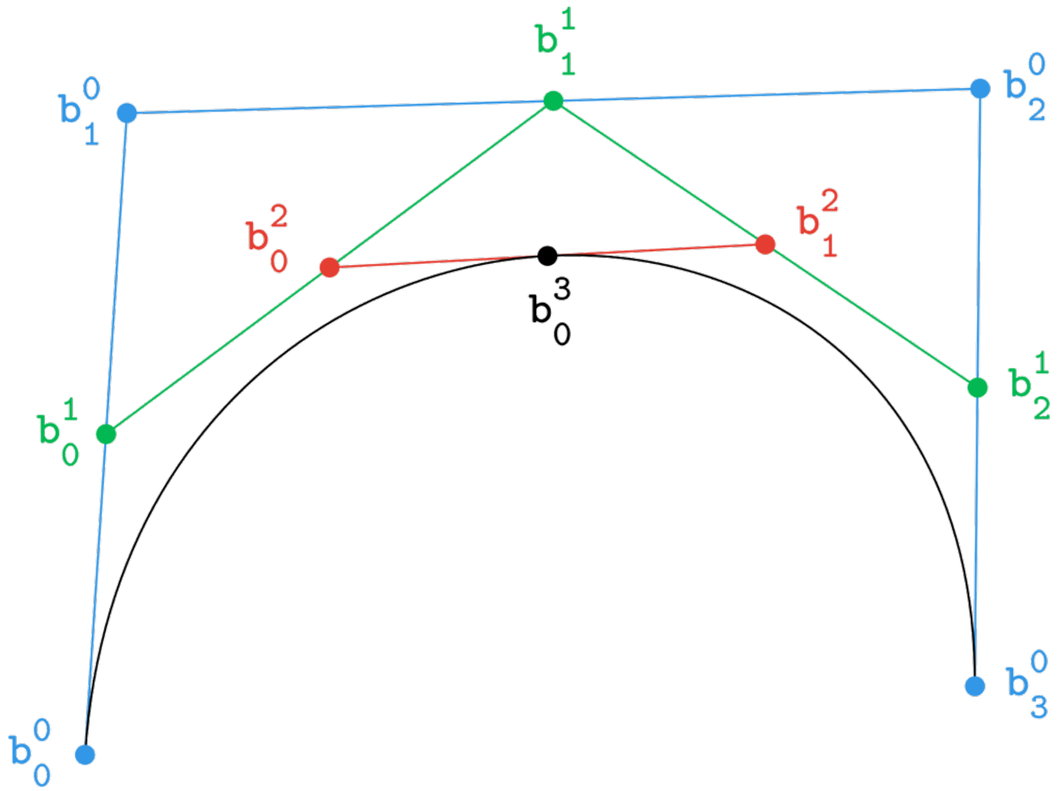
- This definition immediately provides a geometric construction to evaluate the curve by means of repeated affine averages

de Casteljau algorithm



de Casteljau algorithm

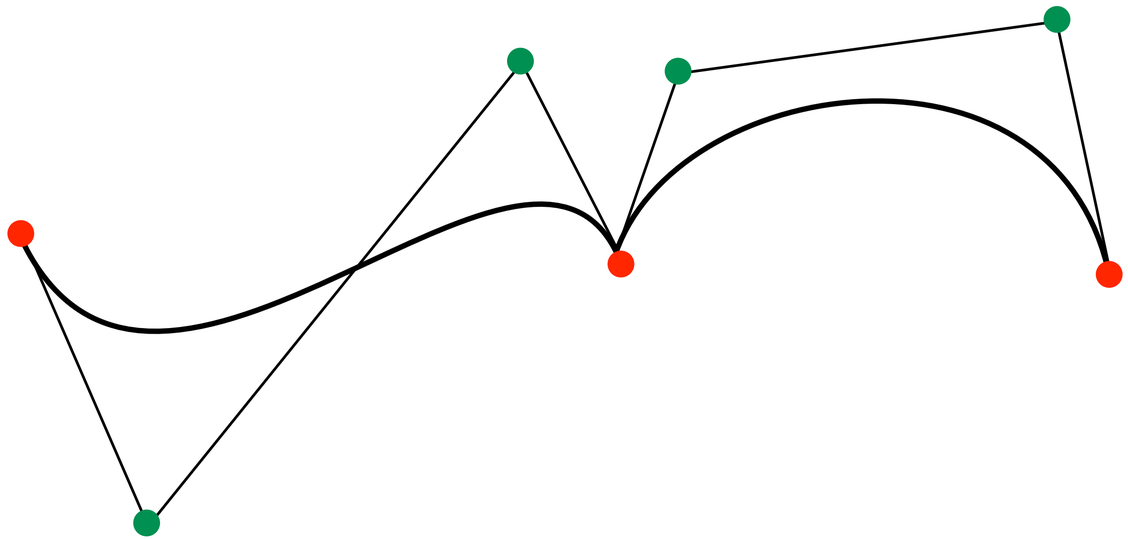
- The de Casteljau algorithm computed at any value t splits the curve into two Bézier segments, providing their control polygons $\mathbf{b}_0^0, \mathbf{b}_0^1, \mathbf{b}_0^2, \mathbf{b}_0^3$ $\mathbf{b}_0^3, \mathbf{b}_1^2, \mathbf{b}_2^1, \mathbf{b}_3^0$
- Recursive evaluation at $t = \frac{1}{2}$ provides a sequence of control polygons that collapses onto the curve



Bézier splines

Bézier curves can be joined easily with C^0 and C^1 continuity:

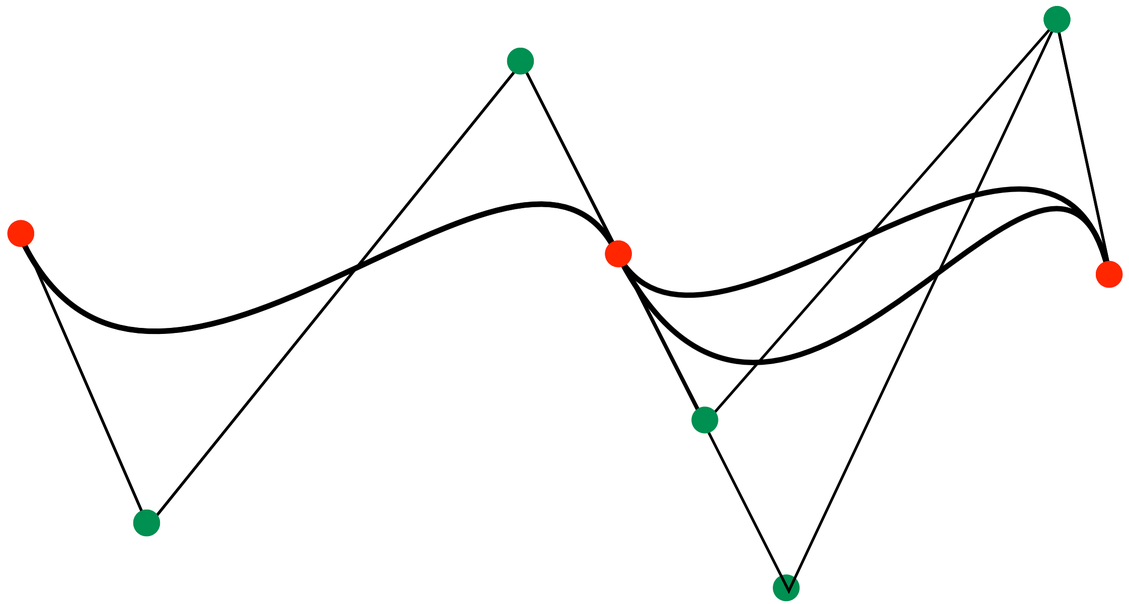
- C^0: the last control point of a curve and the first control point of the next curve coincide



Bézier splines

- G^1: the last edge of a control polygon and the first edge of the next control polygon are collinear

- C^1: the last edge of a control polygon and the first edge of the next control polygon are collinear and have the same length



Bézier splines

- C^2 continuity involves the mutual positions of the last three control points of a curve and the first three of the next

- This may cause a loss of local control (cascade effect on successive curve segments)

Can we do better than Bézier?

Wish list:

- Local support: each control point should influence only one small portion of curve
- degree of the curve should not depend on the number of control points
- Smooth splines: it should be easy to join curve segments smoothly
- Expressive power:
  - use a more flexible set of bases (Bézier bases are fixed)
  - what about representing conics?

B-splines

- basis functions with compact support and flexible
- defined upon a set of *knots* that subdivide an interval on $\mathbb{R}$
- knots control the amplitude and the shape of each basis function
- basis functions determine the influence of each control point
- each segment of spline is determined by a subset of control points
- consecutive segments have overlapping sets of control points

B-splines

B-spline basis functions:

- $U = \{u_0, \cdots, u_m\}$ uniform partition of interval $[u_0, u_m] \subset \mathbb{R}$
- The i-th basis function of degree p (order p+1) on U is defined:

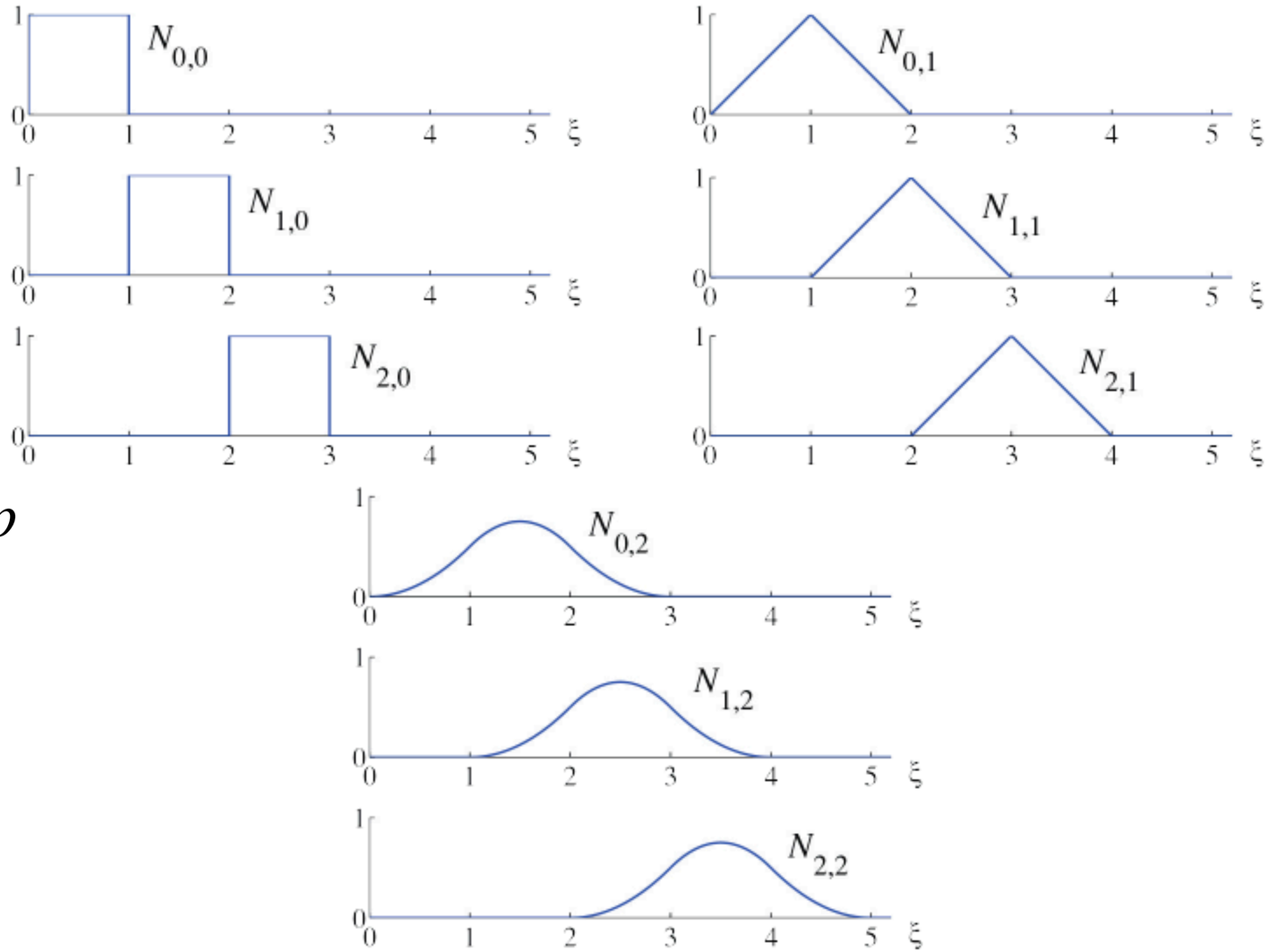
$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t \in [u_i, u_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(t) = \frac{t - u_i}{u_{i+p} - u_i} N_{i,p-1}(t) + \frac{u_{i+p+1} - t}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(t)$$

B-splines

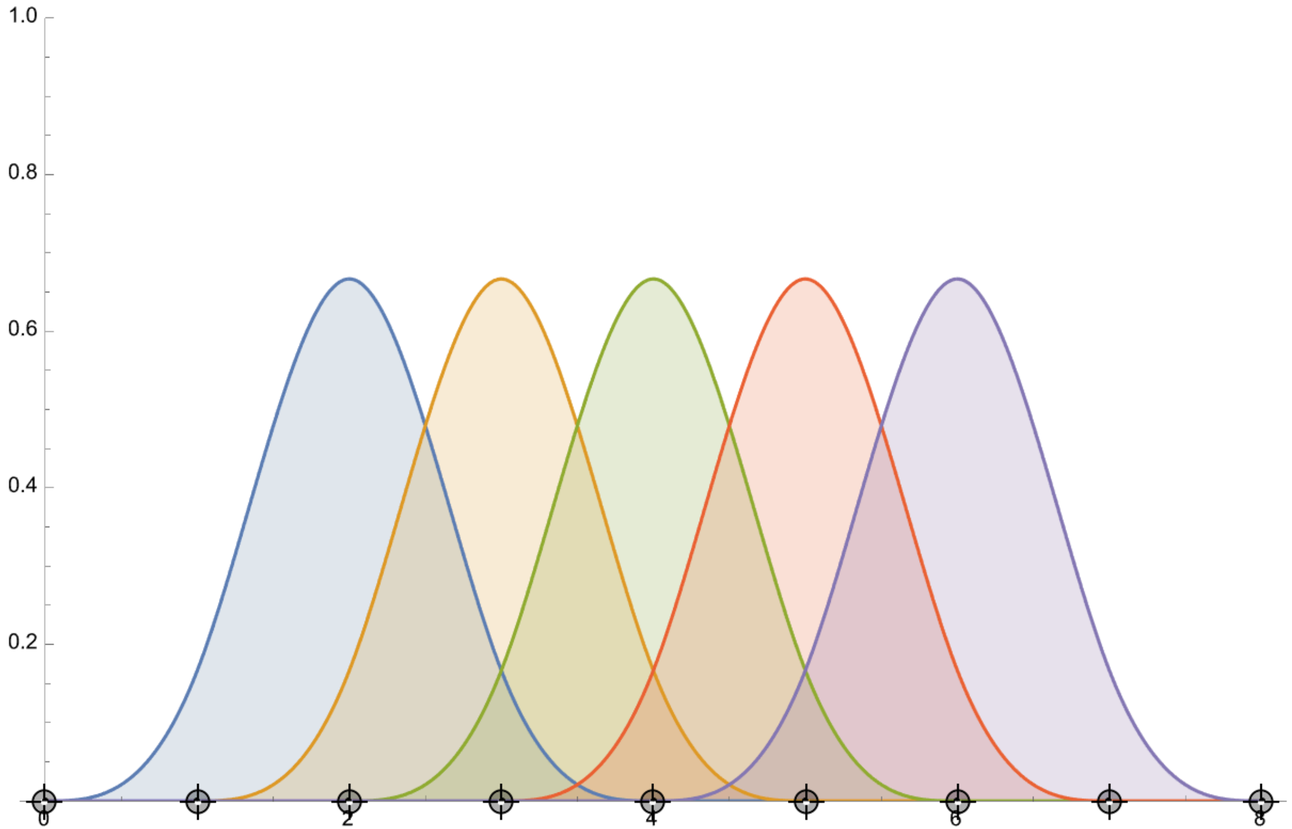
B-spline basis functions:

- recursive construction
- by linear combination
- smoothness increases with p



B-splines

B-spline basis functions with m=8, p=3



B-splines

B-spline basis functions - properties:

- there are $n = m\text{-}p$ bases of degree p on m intervals
- all non-negative
- each basis spans p+1 intervals
- consecutive bases overlap
- partition of unity in $[u_p, u_{m-p}]$: $\sum_{i=0}^{n-1} N_{i,p}(t) = 1$

B-splines

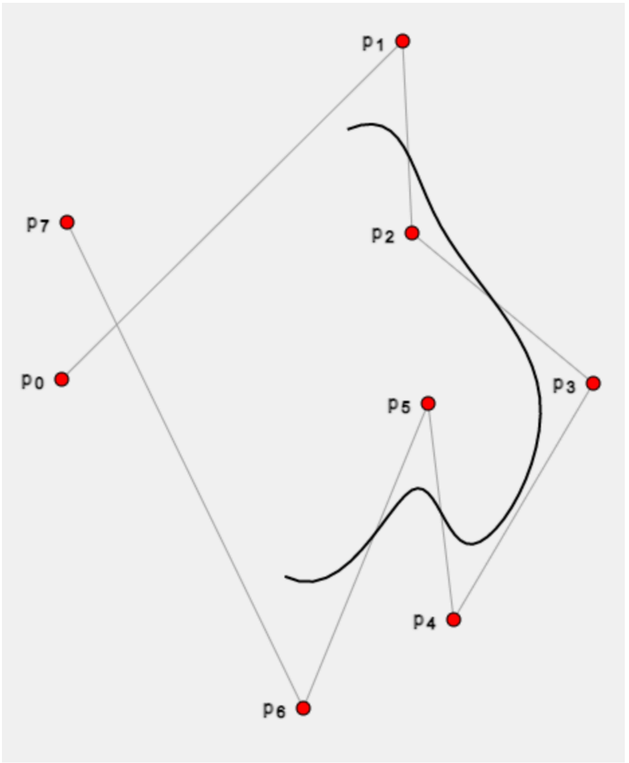
B-spline curve:

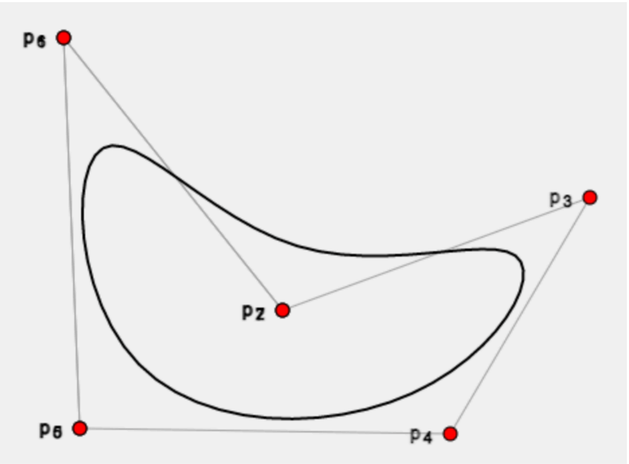
$$s(t) = \sum_{i=0}^{n-1} N_{i,p}(t)\mathbf{p}_i$$

- defined in interval $[u_p, u_{m-p}]$
- consisting of *m-2p* segments
- the j-th segment is defined in $[u_{p+j}, u_{p+j+1}]$ and is controlled by points $\mathbf{p}_j, \ldots, \mathbf{p}_{j+p}$ for $j \in [0, m-2p-1]$

B-splines

- Uniform B-splines
  - completely defined by the n control points and the choice of p
  - have smoothness C^{p-1} (cubic B-splines are C^2)
  - do not interpolate any control point
  - closed curves can be defined by using the knot interval in a periodic way (modulo m)

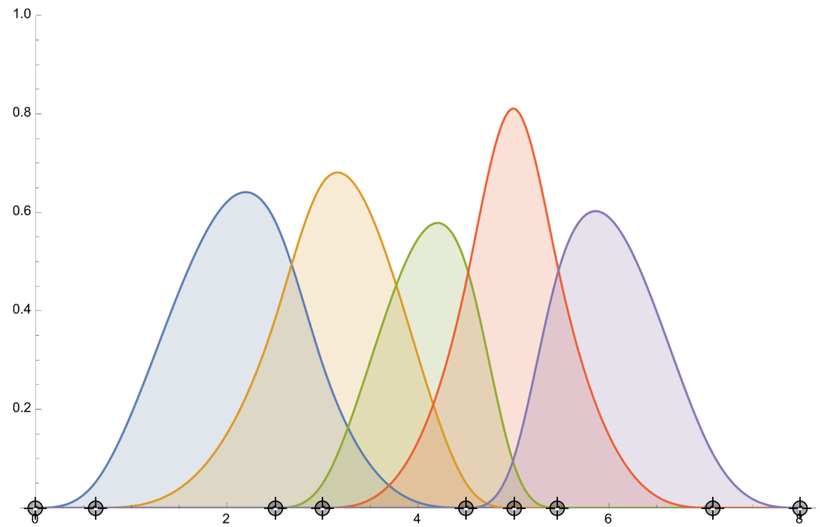




B-splines

Non-uniform B-splines:

- Knots ca be moved to create non-uniform intervals
- Non-uniform intervals modify the shape of the basis functions



B-splines

Non-uniform B-splines:

- the curve can be edited by:
  - moving the control points: they “pull” the curve
  - moving the knots: effect is not quite intuitive

demo: https://mathweb.ucsd.edu/~sbuss/MathCG2/NurbsDemoJS/NurbsDemo.html

B-splines

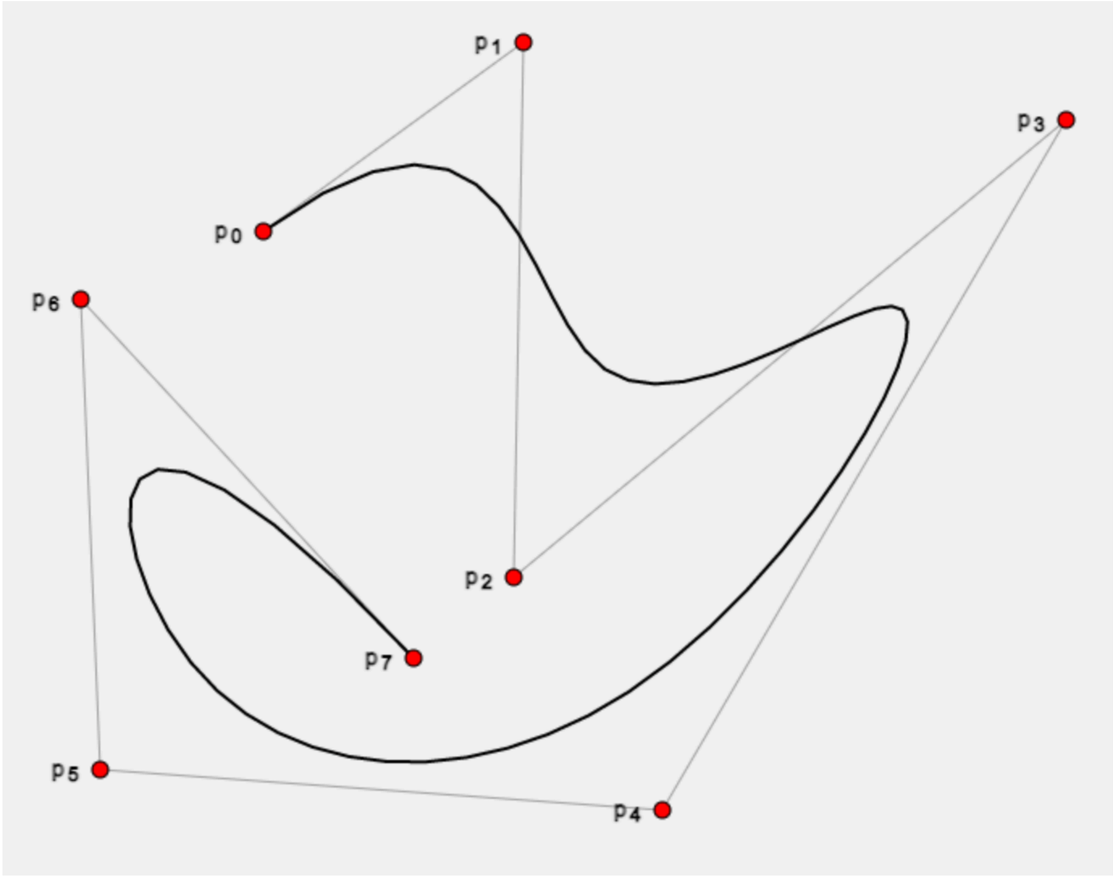
Non-uniform B-splines:

- different knots can be dragged to the same location to change the smoothness of the curve
- each time we increase the multiplicity of a knot, we decrease smoothness by one
- a knot with multiplicity p forces the curve to interpolate one control point (with C^0 continuity)

B-splines

Open-uniform B-splines:

- set multiplicity p at the endpoints of the interval and uniform intervals inside
- we obtain a curve interpolating the endpoints (similar to Bézier), while keeping the smoothness and control of B-splines



B-splines

Properties:

- Piecewise polynomial curves
- For n=p and U=$\{0,\ldots,0,1,\ldots,1\}$ we obtain a Bézier curve with the same control points
- Affine invariance
- Convex hull property
- Local modification
- Variation diminishing property

de Boor algorithm

- Generalization of de Casteljau algorithm for B-splines
- Exploits the recursive definition of B-spline bases

$$\mathbf{b}_i^0 = \mathbf{p}_i$$
$$\mathbf{b}_i^j = (1-\alpha_i^j)\mathbf{b}_{i-1}^{j-1} + \alpha_i^j \mathbf{b}_i^{j-1} \text{ where } \alpha_i^j = \frac{t-u_i}{u_{i+p-j+1}-u_i}$$
$$s(t) = \mathbf{b}_i^p$$

- Easy and efficient to implement

de Boor algorithm

Knot insertion

- Purpose: represent the same curve with more knots (and control points) to support more flexible editing
- Problem: given a B-spline with m knots, find another B-spline with $m+1$ knots that describes the same curve
- Boehm algorithm provides a simple formula to compute the positions of the control points upon insertion of one new node
- Oslo algorithms generalize to the simultaneous insertion of more nodes

Knot insertion

Boehm algorithm:

- given $s(t) = \sum_{i=0}^{n} N_{i,p}(t)\mathbf{p}_i$ defined on node vector $[u_0, \ldots, u_m]$
- insert node $\bar{u}$ in (u_k, u_{k+1})
- equivalent B-spline $\bar{s}(t) = \sum_{i=0}^{n+1} \bar{N}_{i,p}(t)\bar{\mathbf{p}}_i$ defined on $[u_0, \ldots, u_k, \bar{u}, u_{k+1}, u_m]$

$$\bar{\mathbf{p}}_i = (1-\alpha_i)\mathbf{p}_{i-1} + \alpha_i \mathbf{p}_i \quad \text{where} \quad \alpha_i = \begin{cases} 1 & \text{if } i \leq k-p \\ \frac{\bar{u}-u_i}{u_{k+p}-u_i} & \text{if } k-p+1 \leq i \leq k \\ 0 & \text{if } i \geq k+1 \end{cases}$$

Other algorithms

There exist many other algorithms to support curve editing:

- Knot removal
- Degree elevation
- Degree reduction
- Point inversion
- Reparametrization
- Data interpolation / approximation

NURBS

Non-Uniform Rational B-Splines:

- Add weights w_i to control points
  - a point with a larger weight “pulls” the curve stronger
- Weights violate the partition of unity: need for normalization
- Basis functions R_i become rational
- The most expressive class of splines

$$s(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t) w_i \mathbf{p}_i}{\sum_{i=0}^{n} N_{i,p}(t) w_i}$$

$$R_i(t) = \frac{N_{i,p}(t) w_i}{\sum_{i=0}^{n} N_{i,p}(t) w_i}$$

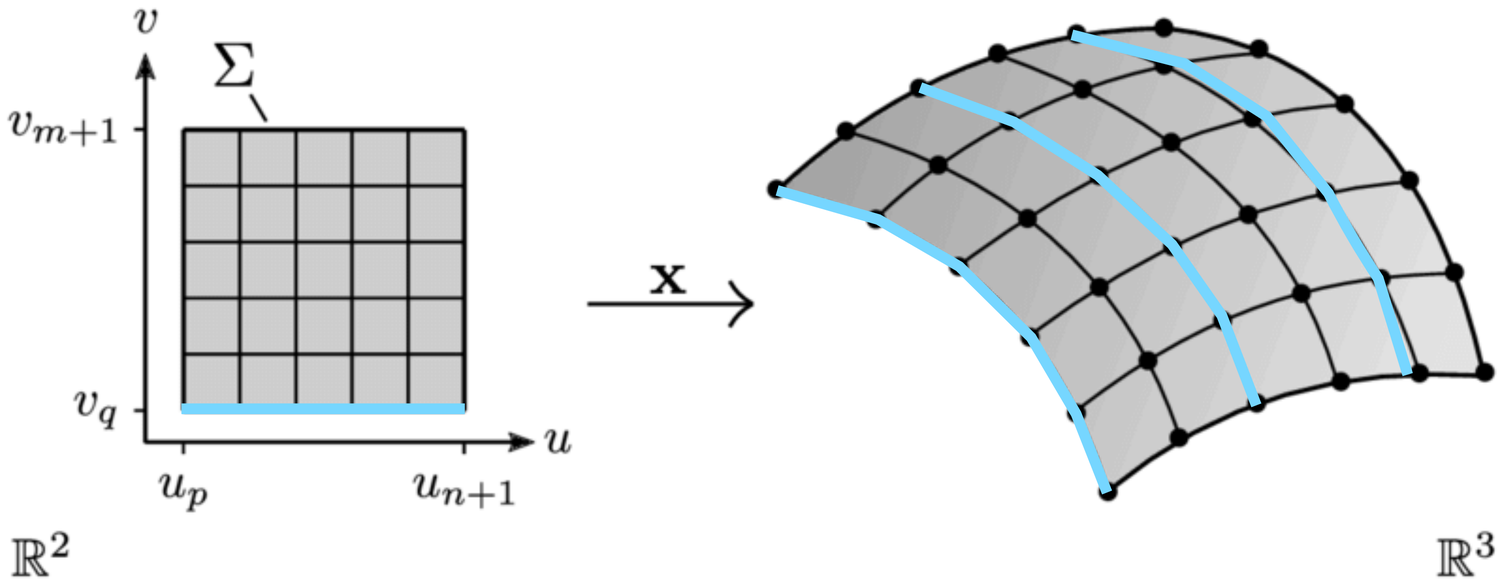
$$s(t) = R_i(t) \mathbf{p}_i$$

Surfaces

Tensor product surfaces

Idea:

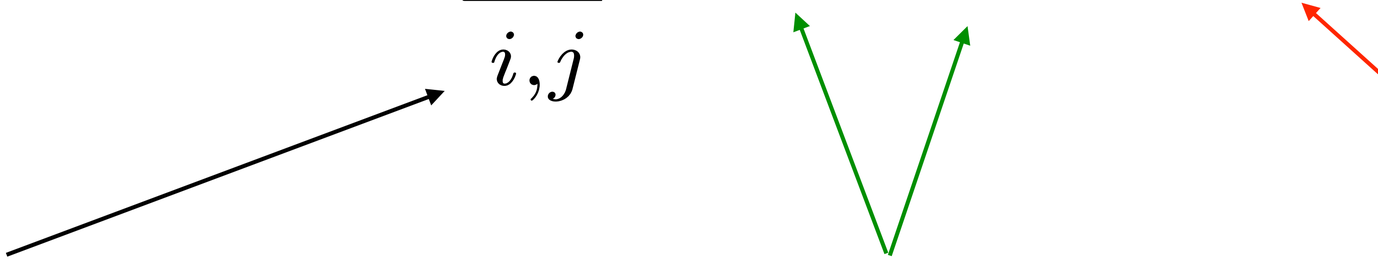
- Sweep a curve in space by dragging its control points along other curves



Tensor product surfaces

In math terms:

$$S(u,v) = \sum_{i,j} B_i(u) B_j(v) \mathbf{p}_{ij}$$



indices vary on a 2D grid

two 1D base functions

control points
arranged on a grid

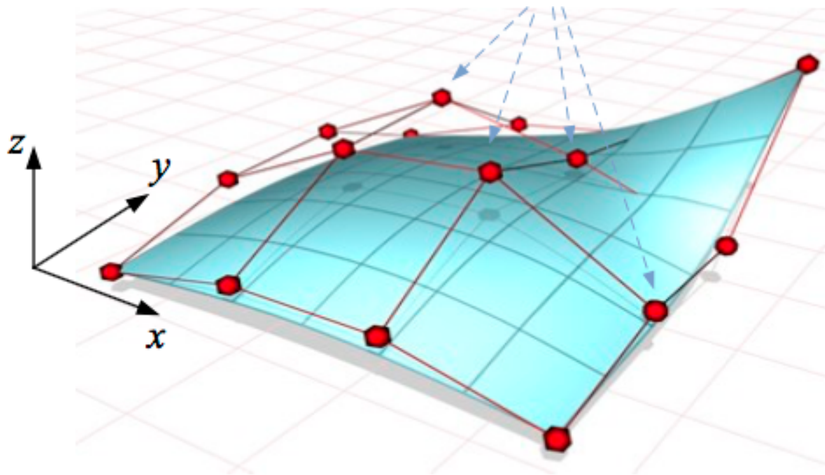
Tensor product surfaces

- This scheme can be applied to obtain surfaces from all piecewise-polynomial schemes of curves (Bézier, B-spline, ...)
- The properties - and the drawbacks - of the resulting schemes are analogous to the corresponding curve schemes

Bézier surfaces

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} B_{i,n}(u) B_{j,n}(v) \mathbf{p}_{ij}$$

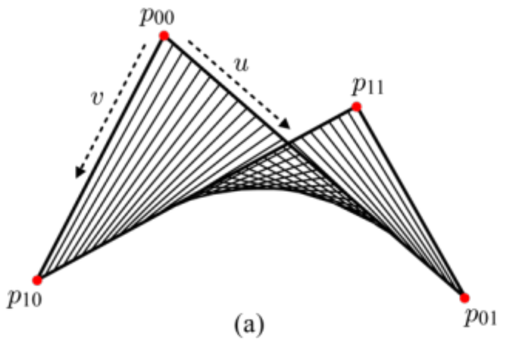
- the $B_{i,n}$ are Bernstein polynomals of degree n
- the control points form a square control net
- the control net is in fact a regular $n \times n$ quad mesh



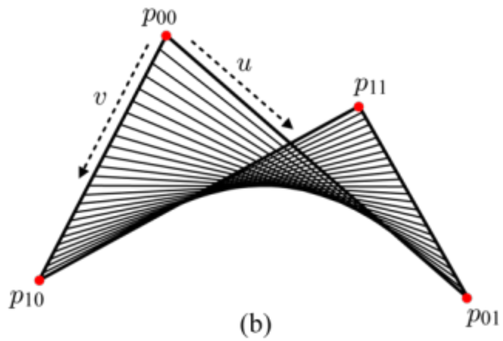
Bi-linear patches

- Each cell of the control net is a bilinear patch
- Linear tensor product
- Parametrized in [0,1]×[0,1]:

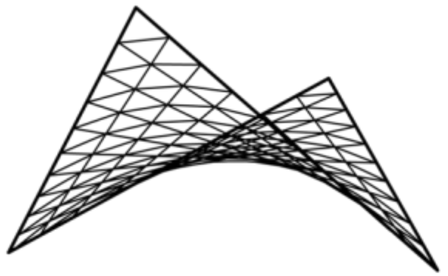
$$
\begin{aligned}
S_{i,j}(u,v) = \quad & \mathbf{p}_{i,j}(1-u)(1-v)+ \\
& \mathbf{p}_{i+1,j}u(1-v)+ \\
& \mathbf{p}_{i,j+1}(1-u)v+ \\
& \mathbf{p}_{i+1,j+1}uv
\end{aligned}
$$



(a)



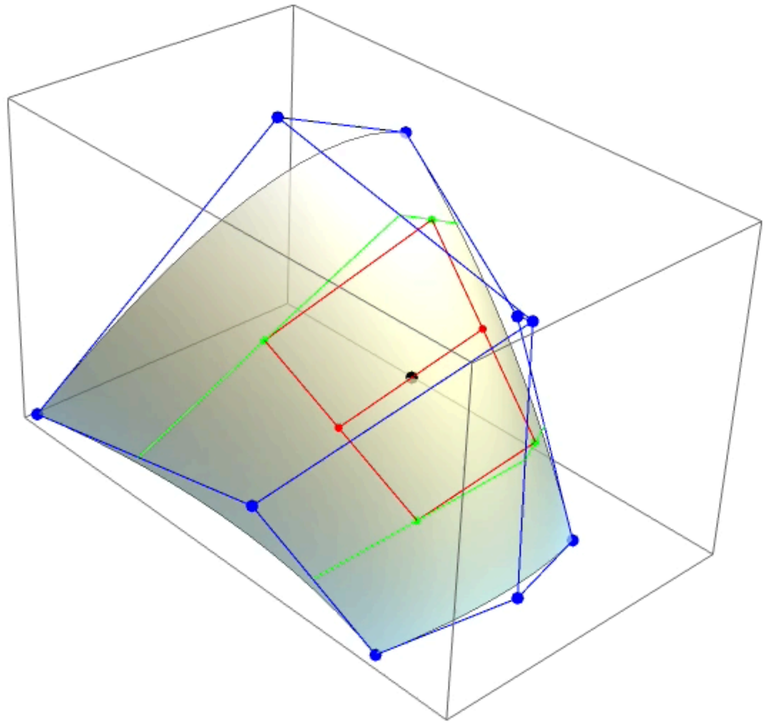
(b)

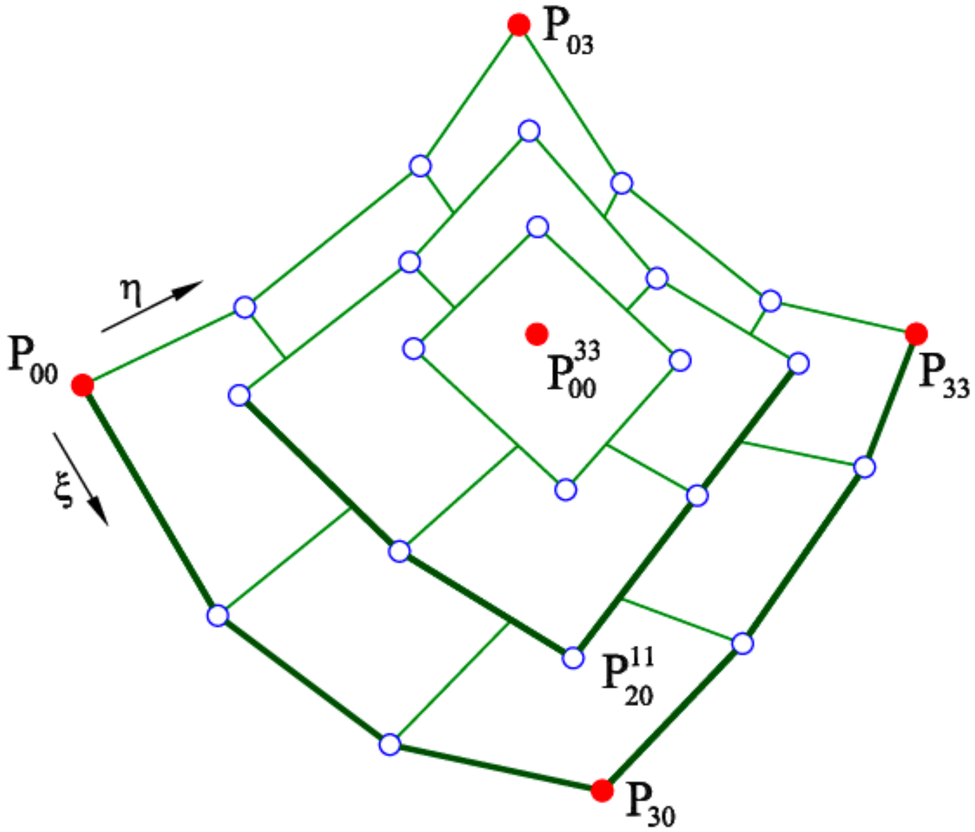

(c)


(d)

de Casteljau algorithm in 2D

- Analogous to the 1D algorithm, but intermediate points are computed with bi-linear interpolation





B-spline surfaces

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,p}(v) \mathbf{p}_{ij}$$

- the $N_{i,p}$ and $N_{j,p}$ are B-spline basis functions of degree p defined on knot vectors $[u_0,\ldots,u_{n+p+1}]$ and $[v_0,\ldots,v_{m+p+1}]$, respectively
- the control net is a regular $n \times m$ quad mesh

NURBS

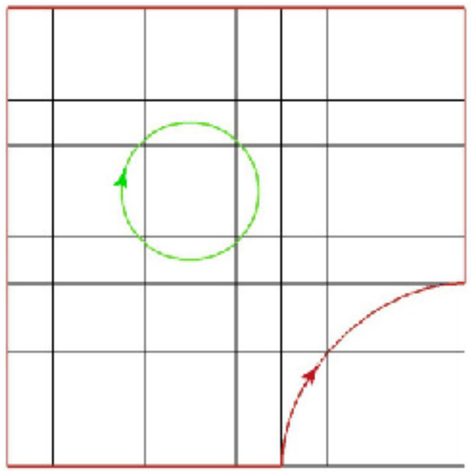
- As in 1D: add weights to control points and normalize:

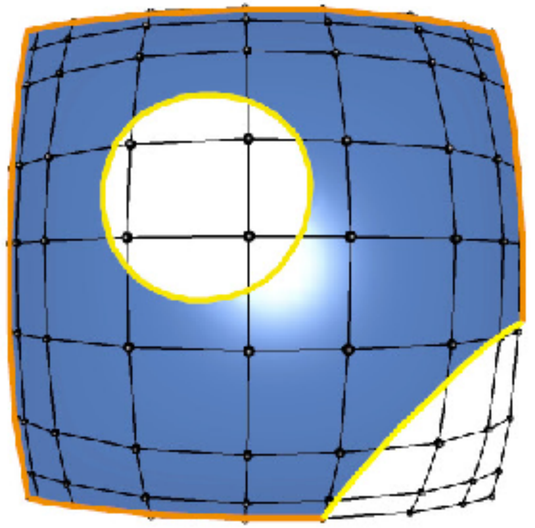
$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,p}(v)w_{i,j}\mathbf{p}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,p}(v)w_{i,j}}$$

- It is <u>not</u> a tensor product surface!

Trimmed surfaces

- Spline surfaces are defined only on regular control nets
- A spline patch is always “rectangular”
- Patches with more complex boundaries are obtained by *trimming* the parametric domain with parametric curves and excluding the portion of patch outside the trimming boundaries





Reference books

- L. Piegl, W. Tiller, “The NURBS Book”, 2nd ed., Springer Verlag, 1997
- “Handbook of Computer Aided Geometric Design”, Edited by G. Farin, J. Hoschek, M. Kim, North-Holland, Elsevier, 2002
- G. Farin, “Curves and Surfaces for CAGD - A practical guide” - Fifth edition, Morgan Kaufman, 2007

Thank you