Parametric surfaces

COM3404

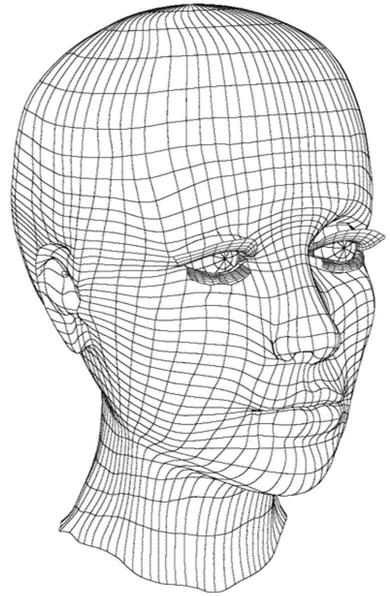
Richard Everson

School of Engineering, Computer Science and Mathematics
University of Exeter

R.M.Everson@exeter.ac.uk
http://www.secamlocal.ex.ac.uk/studyres/COM304

Outline

1. Parametric representation
2. Bézier curves
3. Bézier surfaces
4. B-splines
   - Uniform B-splines
   - Non-uniform B-splines
5. Non-uniform rational B-splines
6. Drawing splines



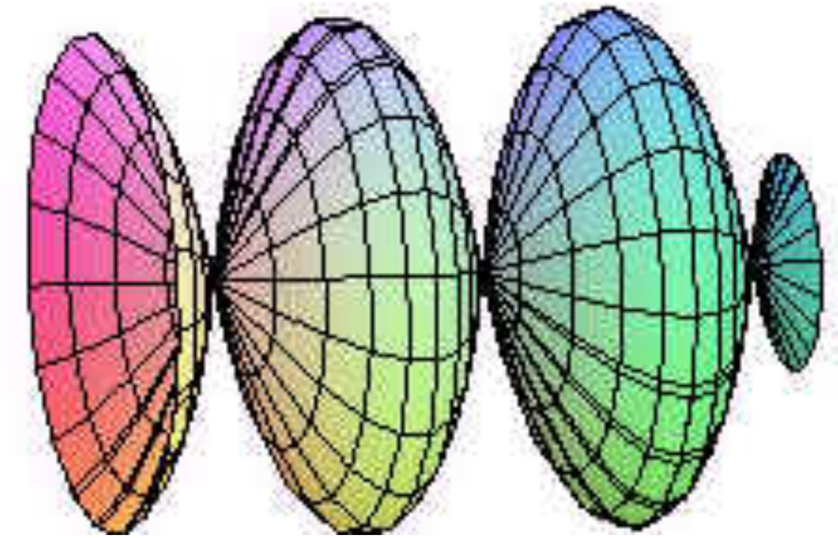
References

- Fundamentals of 3D Computer Graphics. Watt. Chapters 1 & 2
- Computer Graphics: Principles and Practice. Foley et al (1995).
- Mathematical Elements of Computer Graphics. Rogers & Adams (1976).
- `http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html`

Parametric curves and surfaces

Direct curves and surfaces

$$y(x) = f(x)$$
$$z(x, y) = F(x, y)$$



Parametric curves

$$x(t), \quad y(t), \quad z(t)$$

Parametric surfaces

$$x(u, v), \quad y(u, v), \quad z(u, v)$$

$$x(u, v) = u$$
$$y(u, v) = cos(u)cos(v)$$
$$z(u, v) = cos(u)sin(v)$$

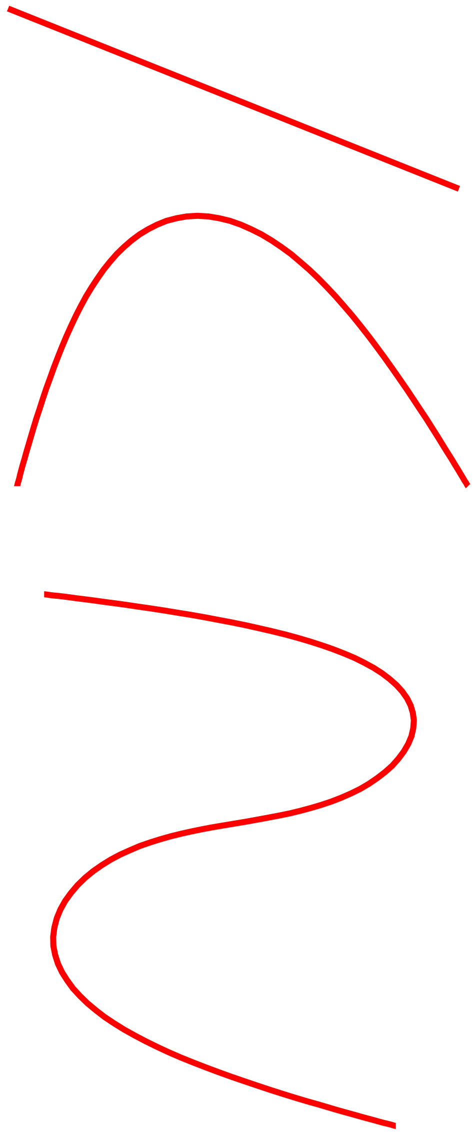
Parameters

t: distance along the curve

u, v : location on the surface

Parametric representation allows surfaces to be multi-valued

Polynomial curves



Linear

$$x(t) = a_{x0} + a_{x1}t$$
$$y(t) = a_{y0} + a_{y1}t$$
$$z(t) = a_{z0} + a_{z1}t$$

Quadratic

$$x(t) = a_{x0} + a_{x1}t + a_{x2}t^2$$
$$y(t) = a_{y0} + a_{y1}t + a_{y2}t^2$$
$$z(t) = a_{z0} + a_{z1}t + a_{z2}t^2$$

Cubic

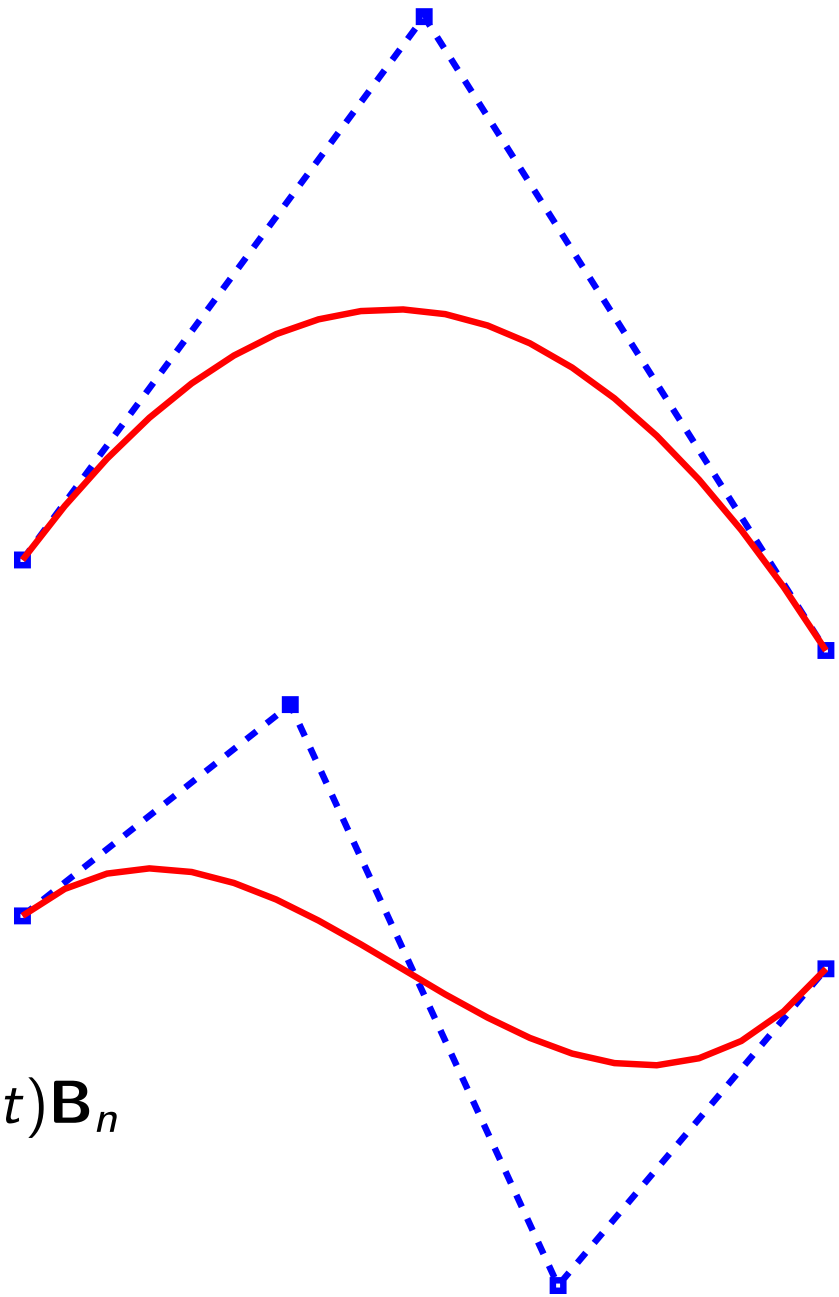
$$x(t) = a_{x0} + a_{x1}t + a_{x2}t^2 + a_{x3}t^3$$
$$y(t) = a_{y0} + a_{y1}t + a_{y2}t^2 + a_{z3}t^3$$
$$z(t) = a_{z0} + a_{z1}t + a_{z2}t^2 + a_{z3}t^3$$

Bézier curves

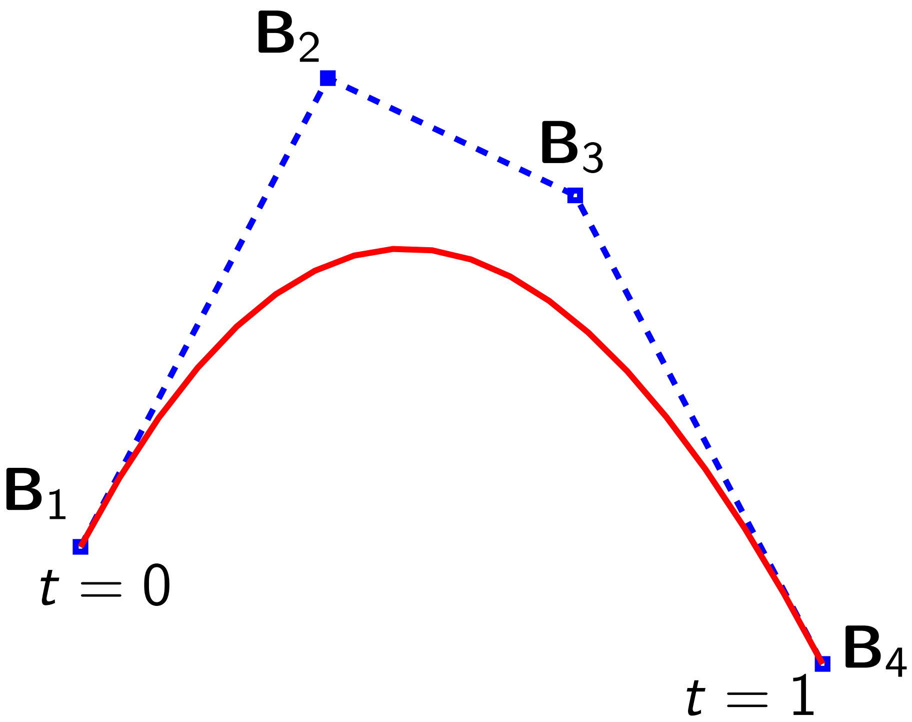
Polynomial curves defined by control vertices

- Pass through the end control vertices
- Usually cubic
- Curve lies within the convex hull of control vertices
- Curve $\mathbf{Q}(t)$ expressed as sum of *blending or basis functions*, N_n

$$(x(t), y(t), z(t)) = \mathbf{Q}(t) = \sum_{n=1}^{N} N_n(t)\mathbf{B}_n$$

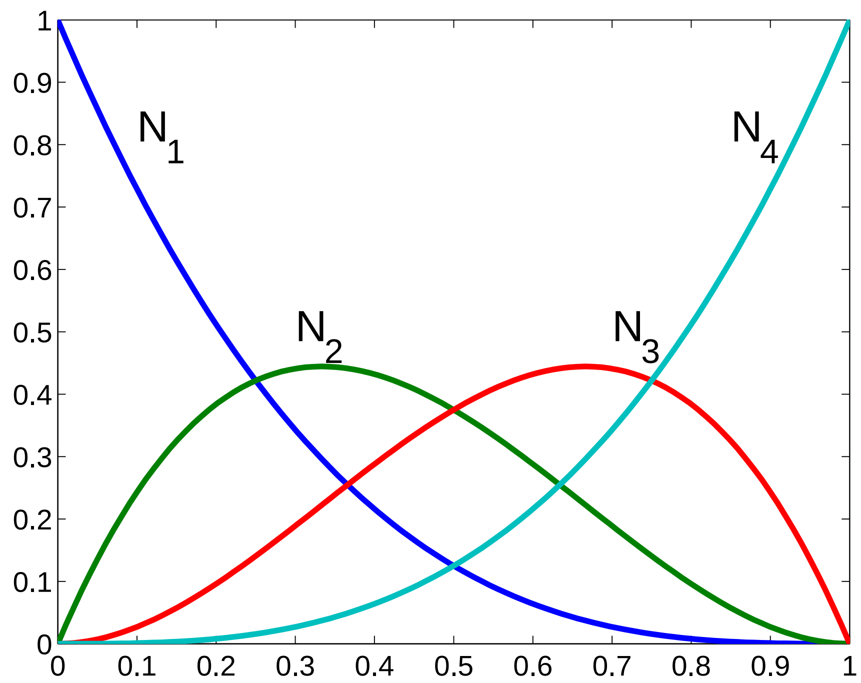


Basis functions



Control vertices $\mathbf{B}_n$ determine location of points along the curve according to blending functions

$$\mathbf{Q}(t) = \sum_{n=1}^{N} N_n(t)\mathbf{B}_n$$



- Basis functions sum to 1 for all $0 \le t \le 1$
- Basis functions are non-negative: $N_n(t) \ge 0$

Bézier curves

Cubic Bézier curves

$$\mathbf{Q}(t) = (1-t)^3\mathbf{B}_1 + 3t(1-t)^2\mathbf{B}_2 + 3t^2(1-t)\mathbf{B}_3 + t^3\mathbf{B}_4$$

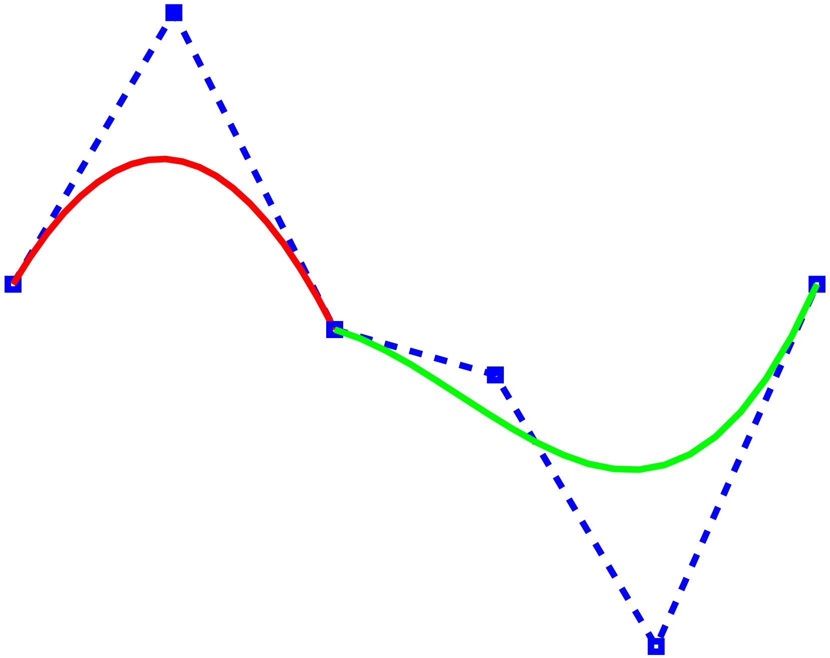
- Control vertices $\mathbf{B}_1 \ldots \mathbf{B}_4$
- Basis functions are the cubic $(n = 3)$ *Bernstein polynomials:*

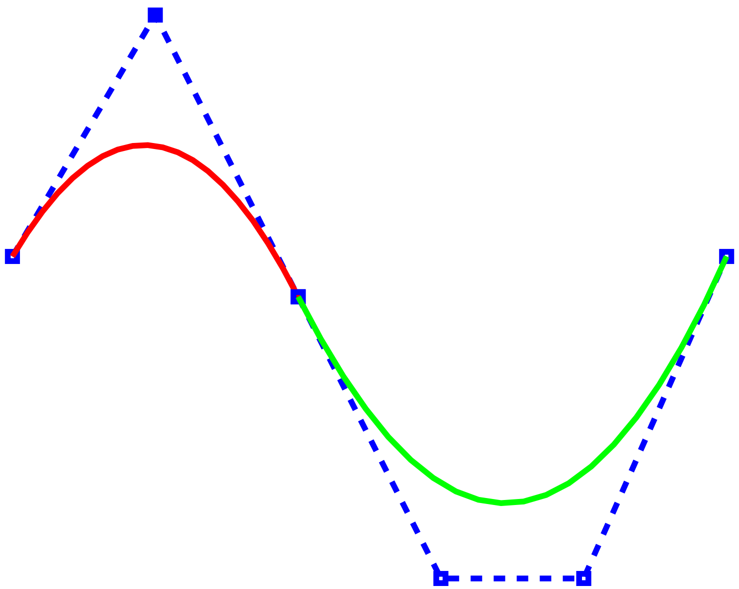
$$N_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{(n-i)}$$

- Basis functions are global, giving non-local control of the curve
- Complex curves constructed from multiple segments

Joining Bézier curves

- Join curves by co-locating end control points
- Smoothness maintained by keeping end pairs of control points co-linear



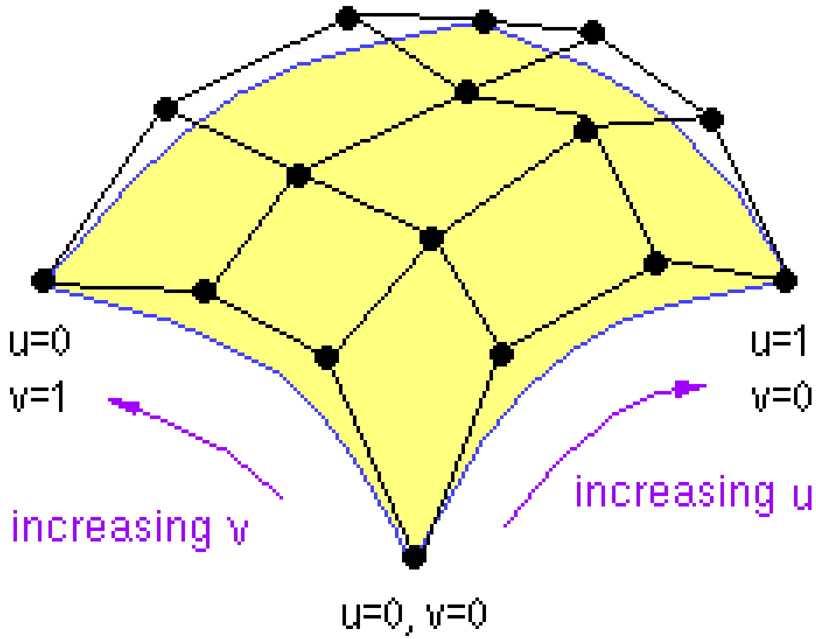


Bézier surfaces

- Surface parameterised by two coordinates: $0 \leq u, v \leq 1$
- Location of a point on the surface is

$$\mathbf{Q}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_i^n(u) N_j^m(v) \mathbf{B}_{ij}$$

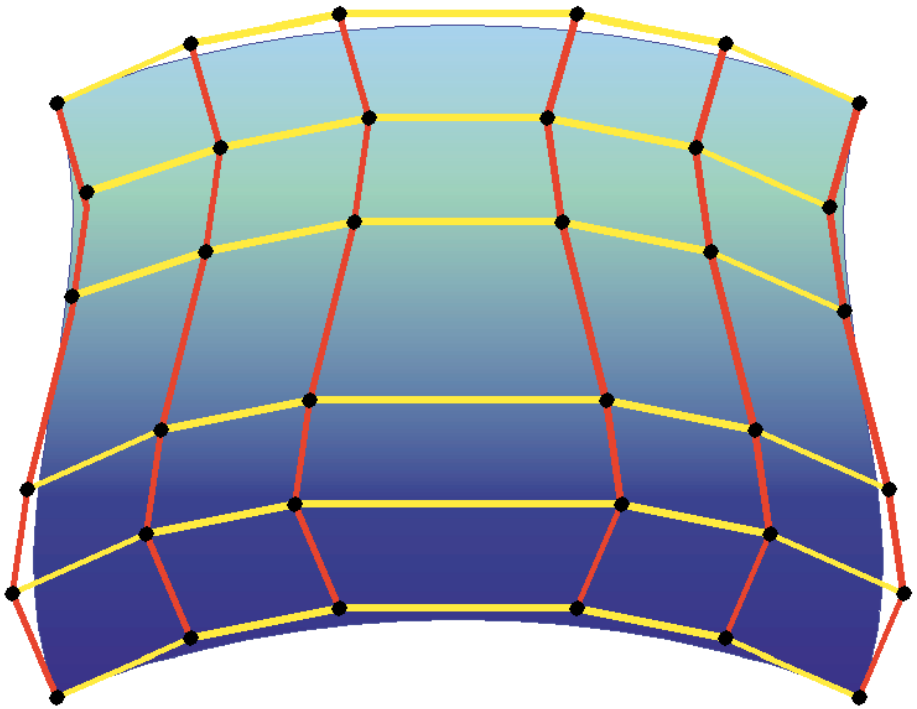
  with N_i^n the Bernstein polynomials.



Bicubic surface

- Surface lies within the convex hull of its control points
- Surface transforms with its control points
- Curves for constant u or v are themselves Bézier curves

Bézier surfaces



Biquintic patch

- Surfaces can be arbitrarily complex by using sufficiently many control points
- However, control is non-local
- Commonly surfaces are constructed from bicubic patches joined in the same manner as Bézier curves

B-splines

Basis splines

Piecewise cubic curves

- Basis functions located at intervals along t
- Basis functions are local
- Degree of spline polynomial is independent of number of control points

General equation

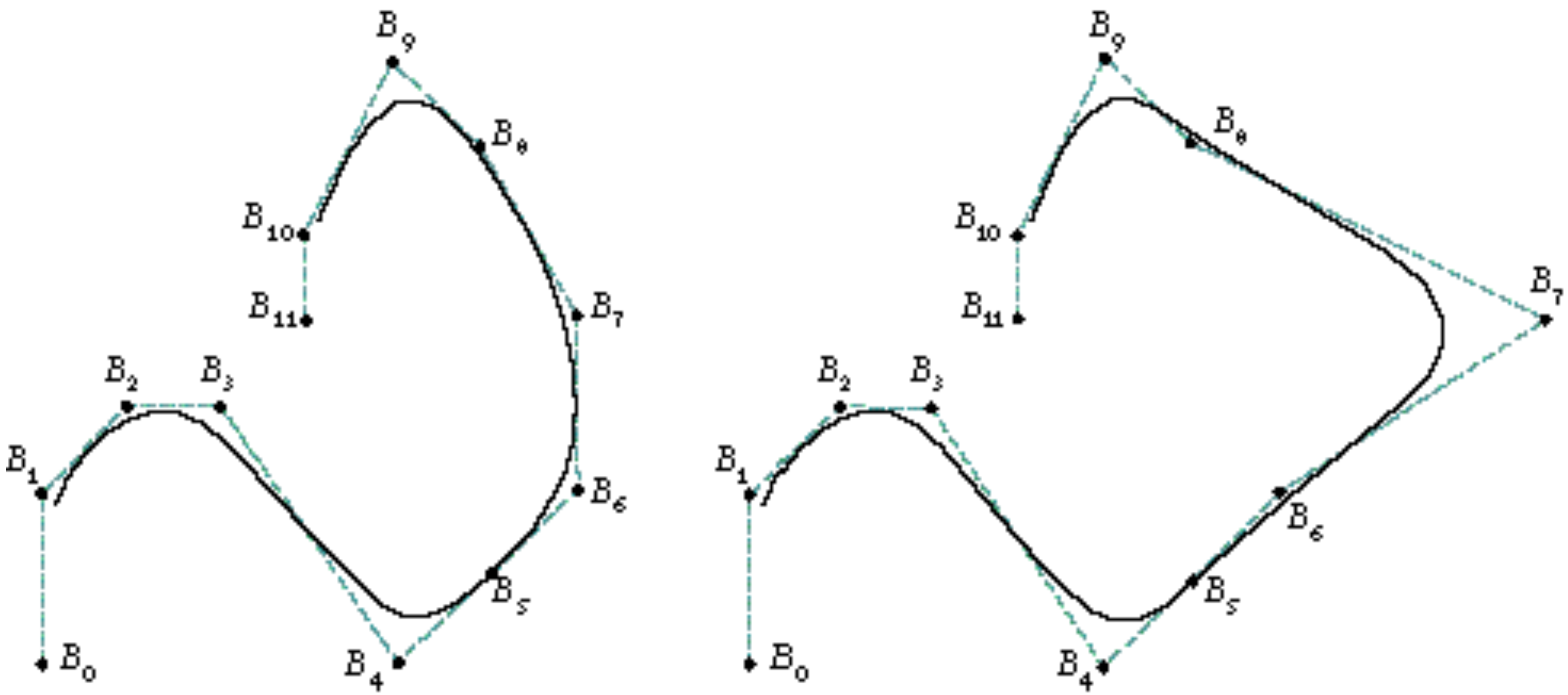
$$\mathbf{Q}(t) = \sum_{i=0}^{n} N_{i,k}(t)\mathbf{B}_i$$

with blending functions $N_{i,k}(t)$

- k defines the degree of the basis function
- n is the number of basis functions

B-splines

Local blending functions provide local control



Uniform B-splines

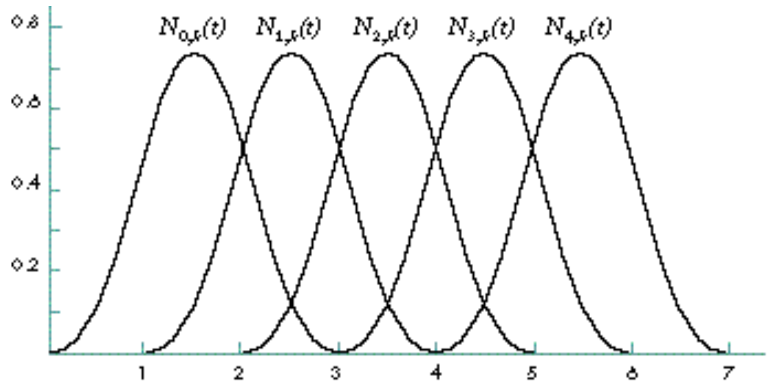
Knot vector Partitions t into intervals. Knots at: $\{t_0, t_1, \ldots, t_n\}$

Uniform B-splines Knots are at equal intervals

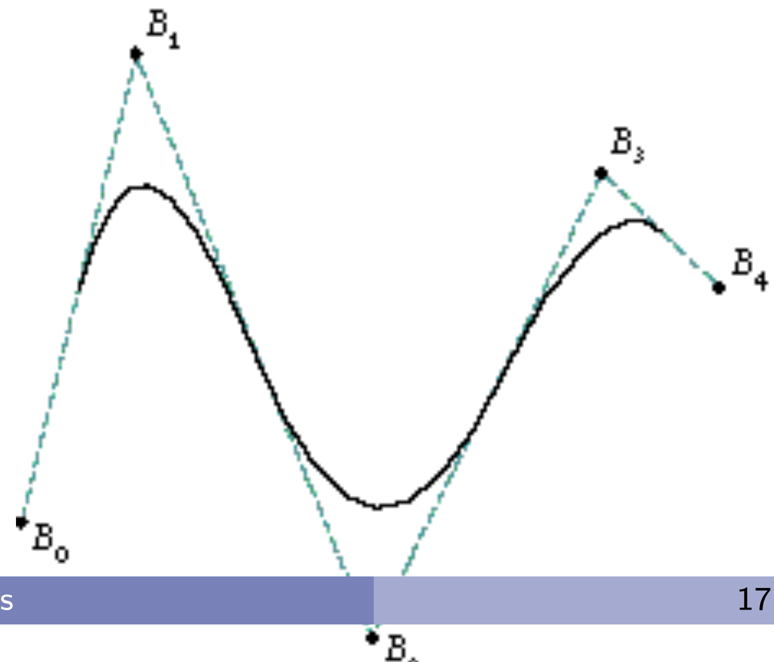
Number of knots $m+1$, number of control points $n+1$ and degree k of blending functions related by:

$$m = n + k + 1$$

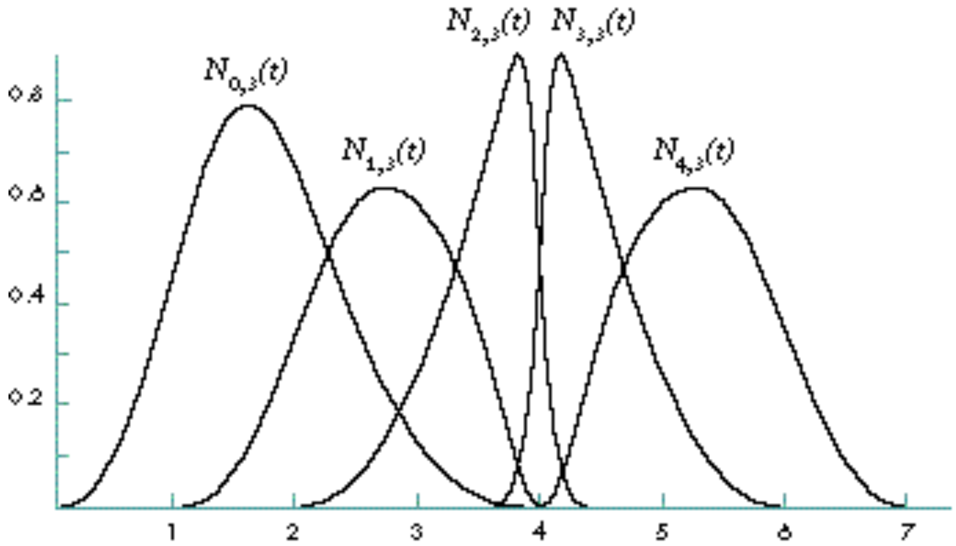
Local basis functions Basis function $N_i, k(t)$ is zero outside the interval $[t_i, t_{i+k+1})$

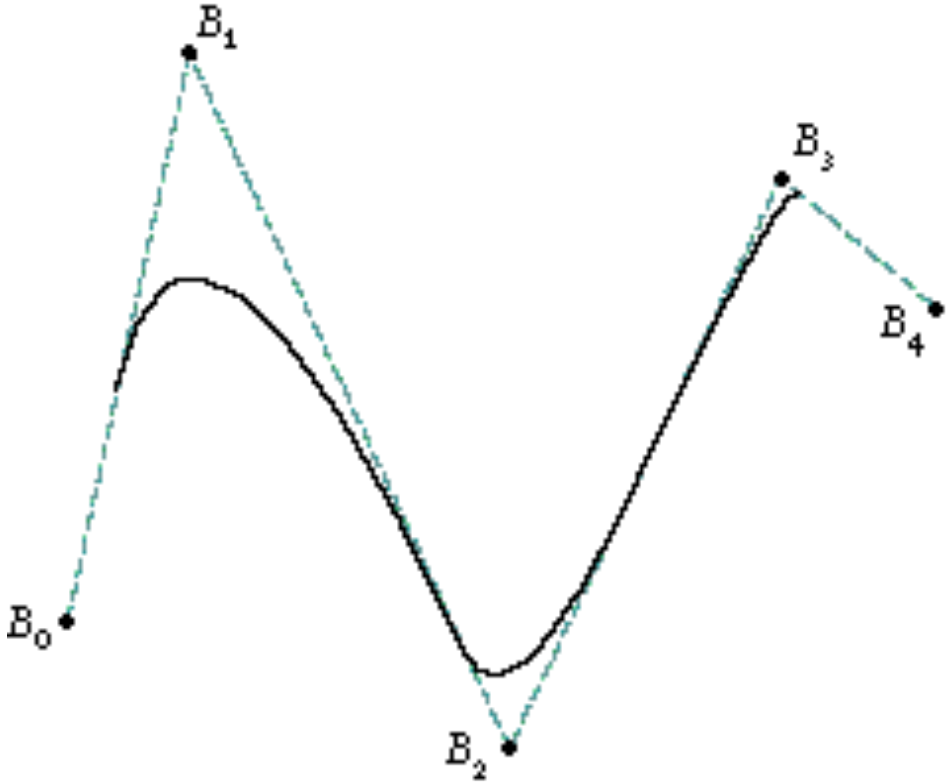


Uniformly spaced basis functions of degree $k=2$. Knot vector $\{0, 1, 2, 3, 4, 5, 6, 7\}$



Non-uniform B-splines





- Non-uniformly spaced knots: $\{0.0, 1.0, 2.0, 3.75, 4.0, 4.25, 6.0, 7.0\}$
- Curve pulled closer to $\mathbf{B}_2$ and $\mathbf{B}_3$ as neighbouring basis functions are larger and concentrated on smaller intervals.

Basis functions

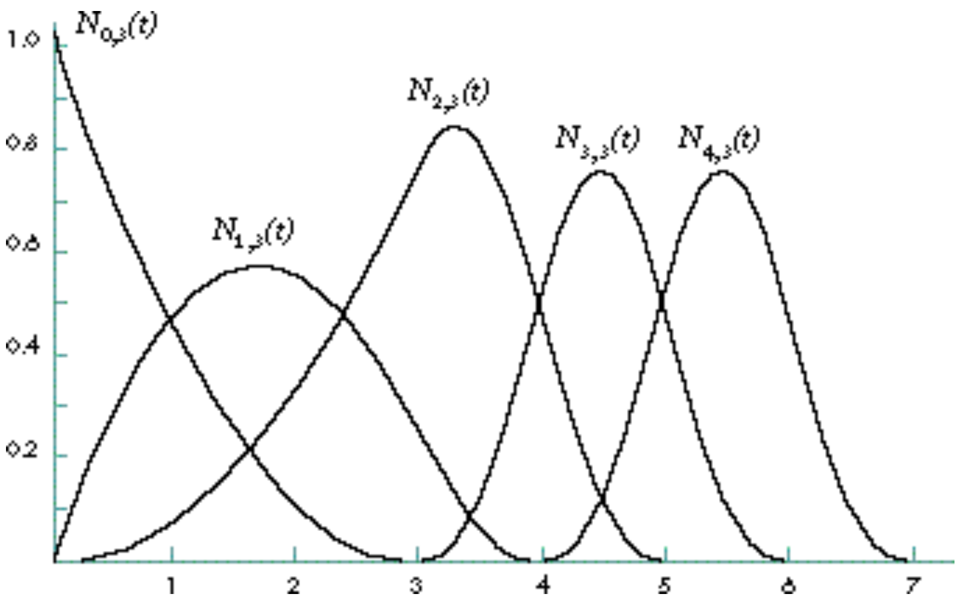
Basis functions defined as

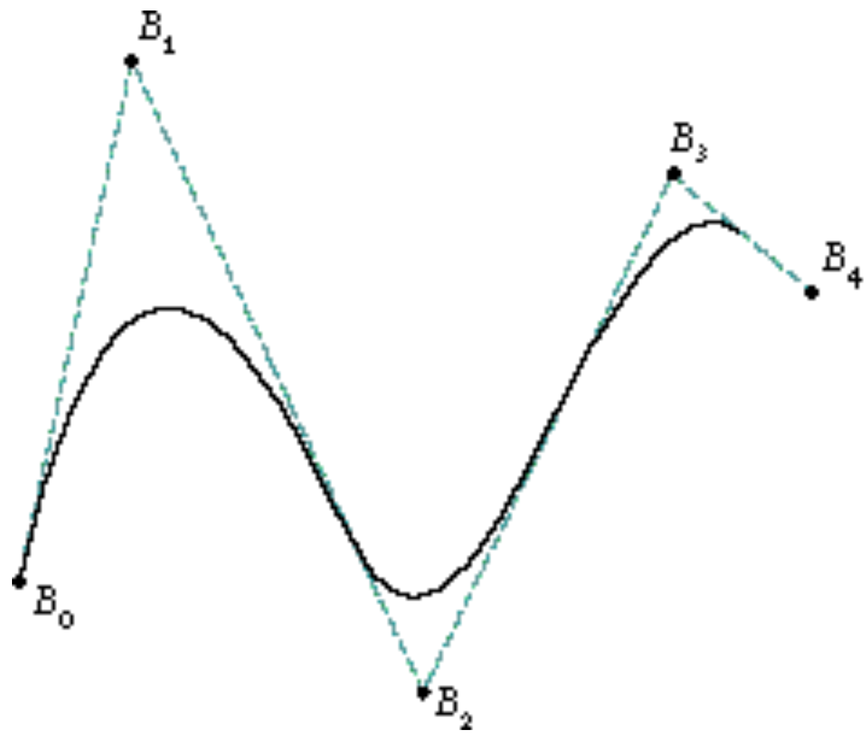
$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t)$$

- $N_{i,k}(t) \geq 0$ for all i, k, t
- $N_{i,k}(t) = 0$ if t not in $[t_i, t_{i+k+1})$
- At any t no more than k basis functions affect the curve
- $\sum_{i=0}^{i} N_{i,k}(t) = 1$
- Curve lies within convex hull of control points

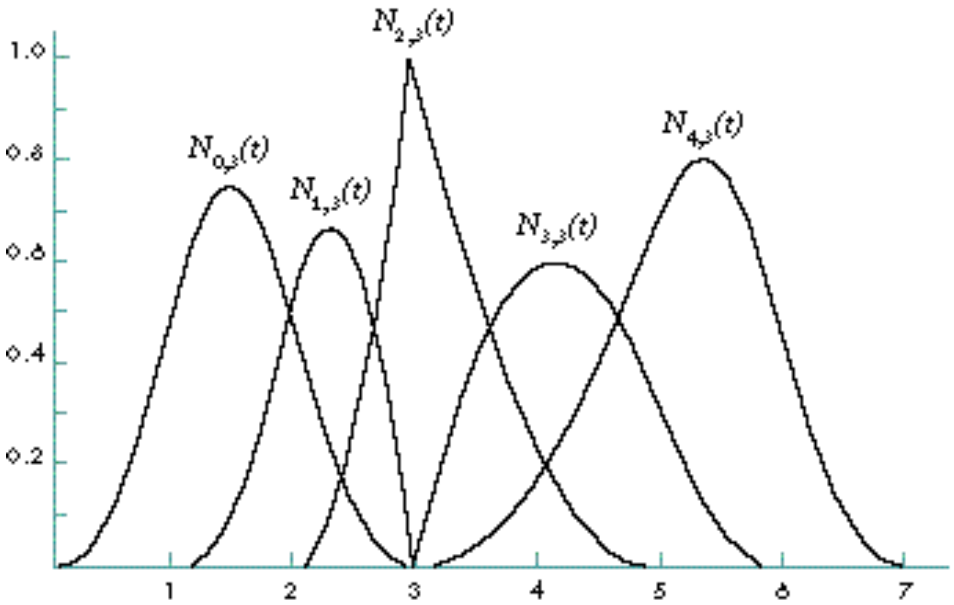
Multiple knots at ends

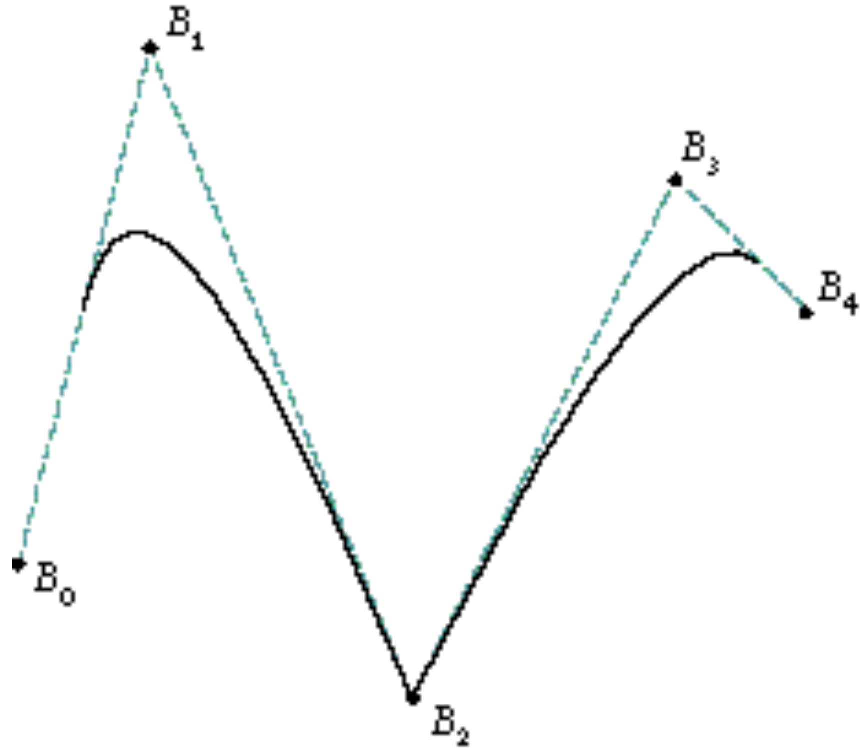




- Knots at $\{0.0, 0.0, 0.0, 3.0, 4.0, 5.0, 6.0, 7.0\}$
- All basis functions except $N_{0,3}(t)$ are zero at $t = 0$. Therefore curve coincides with $\mathbf{B}_0$

Multiple interior knots



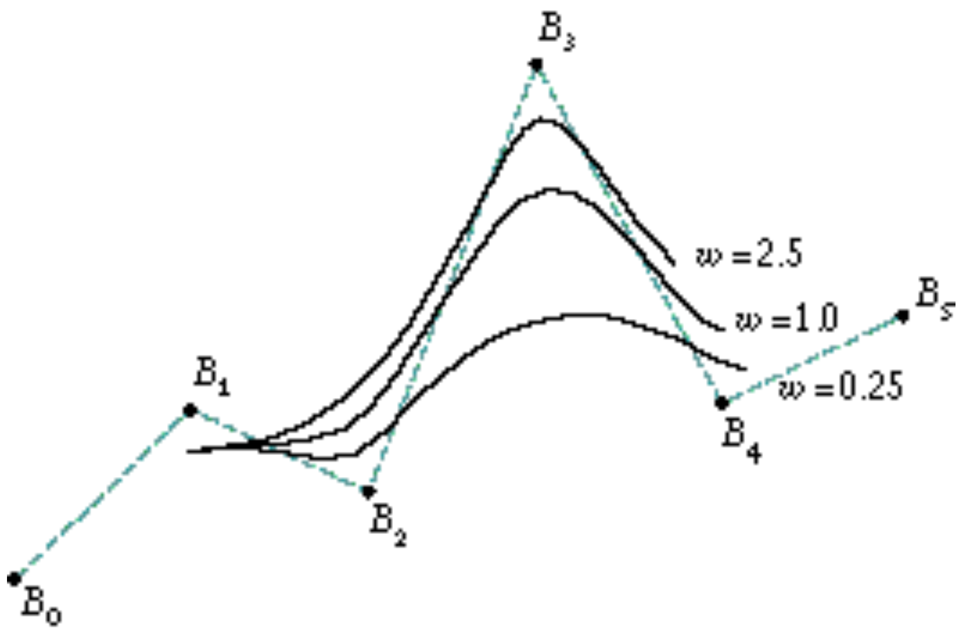


- Knots at $\{0.0, 1.0, 2.0, 3.0, 3.0, 5.0, 6.0, 7.0\}$
- All basis functions except $N_{2,3}(t)$ are zero at $t = 0$. Therefore curve coincides with $\mathbf{B}_2$
- Continuity at a knot is C^{n-p} where p is the multiplicity of the knot.
  - Produce kinks and gaps with sufficient knots

Non-uniform rational B-splines

Weights Weight the control points with weight w_i

$$\mathbf{Q}(t) = \frac{\sum_{i=0}^{n} w_i N_{i,k} \mathbf{B}_n}{\sum_{i=0}^{n} w_i N_{i,k}}$$



Homogeneous coordinates Regard w as an additional coordinate.

- Curves are defined in 4D and projected into 3D
- Control points have coordinates (x, y, z, w); projection in 3 dimensions is $(x/w, y/w, z/w)$

Permits representation of conic sections (circles, ellipses, parabolas, hyperbolas)

Invariant under *projective* as well as affine transformations.

Surfaces

Curves

$$\mathbf{Q}(t) = \frac{\sum_{i=0}^{n} w_i N_{i,k} \mathbf{B}_n}{\sum_{i=0}^{n} w_i N_{i,k}} = \sum_{i=0}^{n} R_{i,k}(t) \mathbf{B}_i$$

with

$$R_{i,k}(t) = \frac{w_i N_{i,k} \mathbf{B}_n}{\sum_{i=0}^{n} w_i N_{i,k}}$$

Surfaces

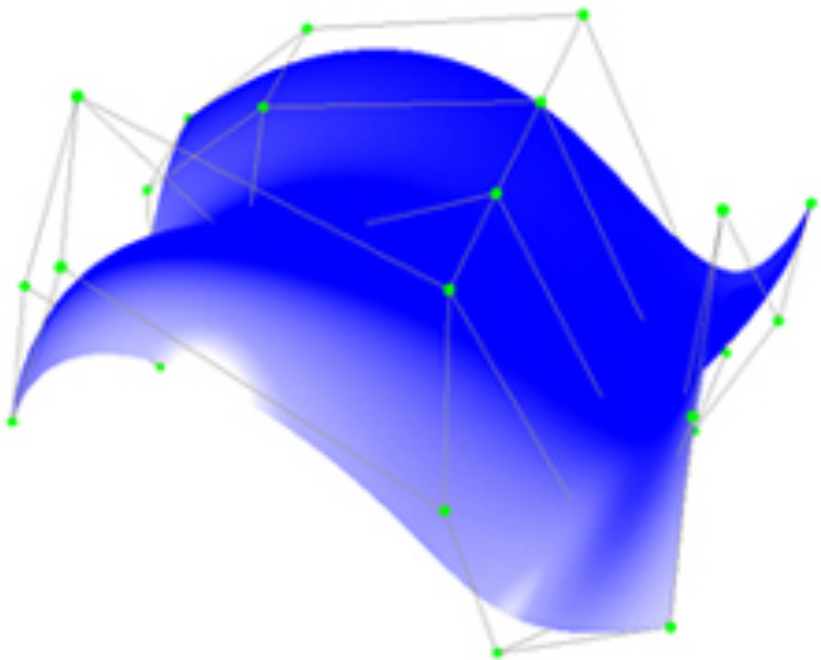
$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j,k,l}(u,v) \mathbf{B}_{i,j}$$

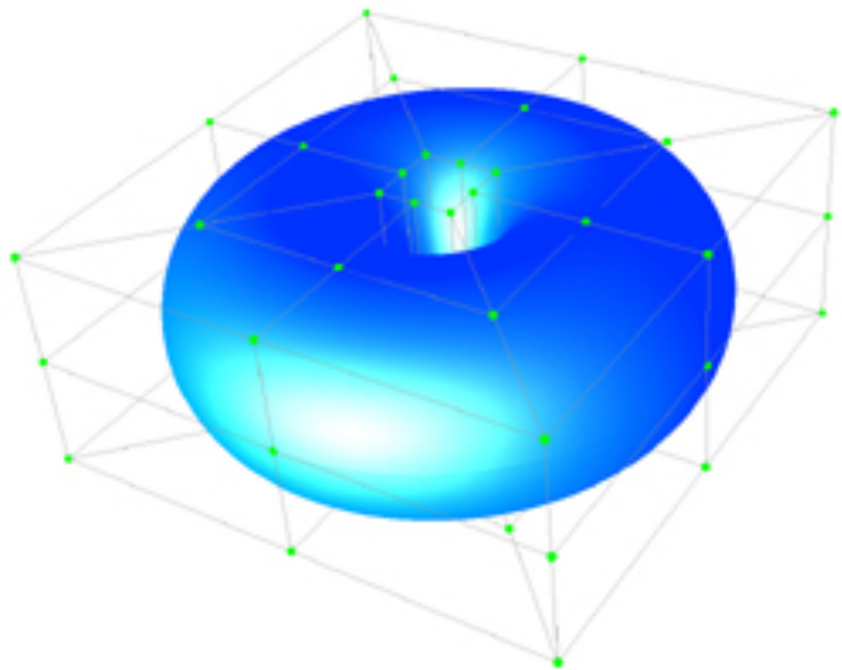
with

$$R_{i,j,k,l}(u,v) = \frac{w_{i,j} N_{i,k}(u) N_{j,l}(v)}{\sum_{r=0}^{n} \sum_{s=0}^{m} w_{r,s} N_{r,k}(u) N_{s,l}(v)}$$

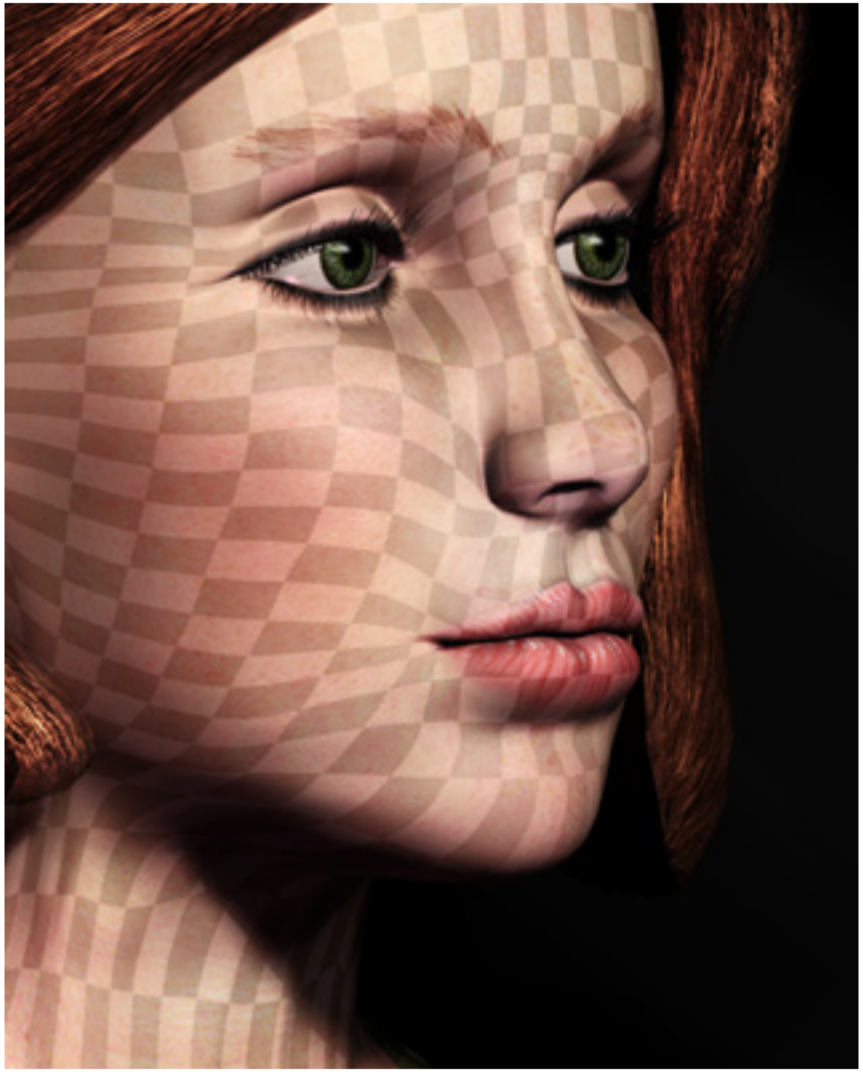
Transformation Curves and surfaces are invariant under affine and perspective transformations, so only control points need by transformed.

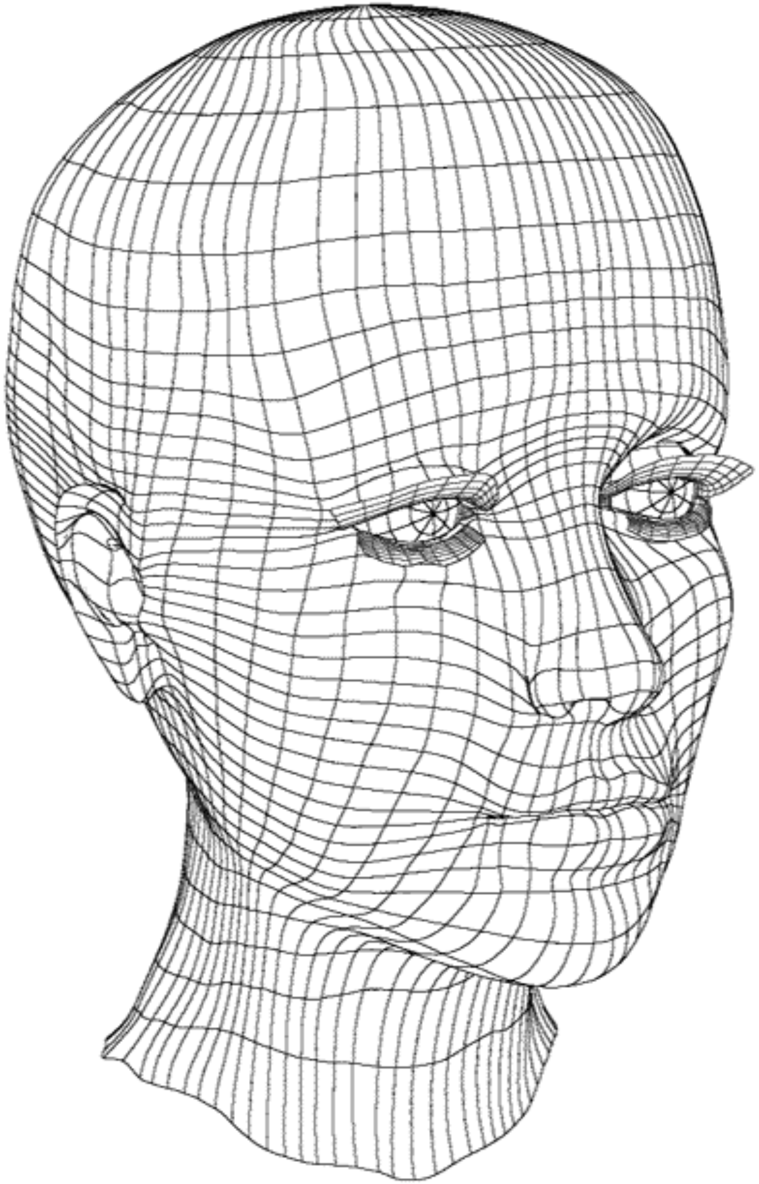
NURBS surfaces





NURBS surfaces





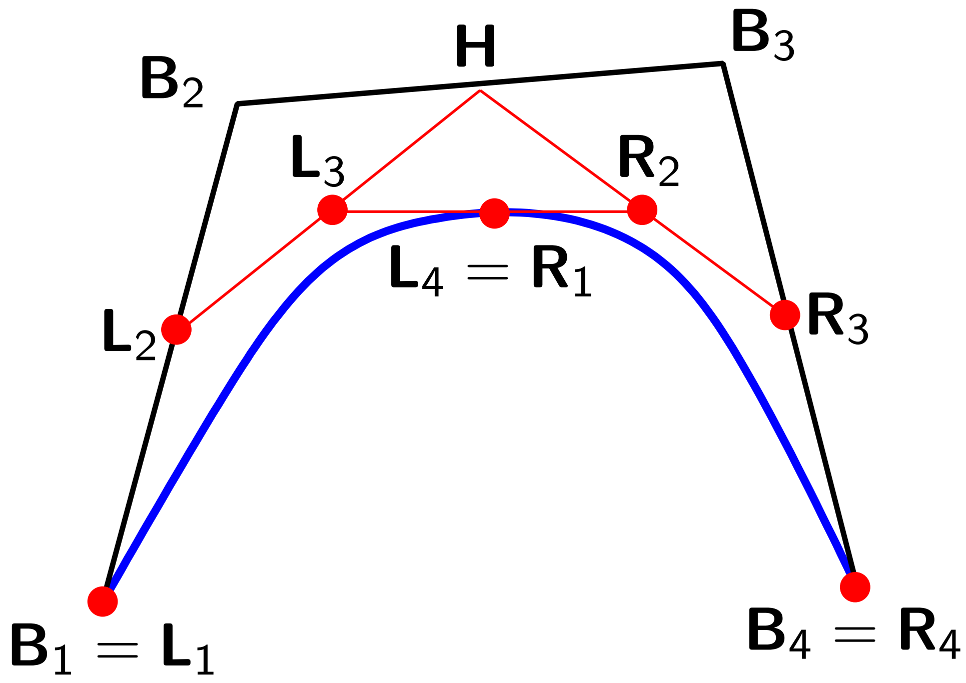
http://www.3drender.com/jbirn/ea/HeadModel.html

Drawing splines

Refine control points defining convex hull

For cubic Bézier curves:

$$\mathbf{L}_2 = (\mathbf{B}_1 + \mathbf{B}_2)/2$$
$$\mathbf{H} = (\mathbf{B}_2 + \mathbf{B}_3)/2$$
$$\mathbf{L}_3 = (\mathbf{L}_2 + \mathbf{H})/2$$
$$\mathbf{R}_3 = (\mathbf{B}_3 + \mathbf{B}_4)/2$$
$$\mathbf{R}_2 = (\mathbf{H} + \mathbf{R}_3)/2$$
$$\mathbf{L}_4 = \mathbf{R}_1 = (\mathbf{L}_3 + \mathbf{R}_2)/2$$



Divides curve into two at $t = 1/2$.
Stop when:

- Line segments are pixels
- Convex hull is sufficiently 'thin' – generally more efficient

Drawing splines

B-splines

- Similar recursive sub-division formulae
- Left and right segments are connected

NURBS

- Basis functions are defined implicitly by recursive formulae
- Sub-division achieved by adding knots (and therefore control points)
- Left and right segments are not connected

Surfaces

- Sub-division formulae can be written for surfaces
- Usually surface is divided until ‘segments’ are sufficiently planar and then drawn as polygons

Details in Foley *et al*, chapter 11.