Computer Graphics

Curves & Surfaces. Part 2.

Konstantin Tretyakov
kt@ut.ee

Dec 4, 2013

Previous time

- Polynomial curve:

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \cdots + \mathbf{c}_n t^n := \mathbf{C}\mathbf{T}_n(t)$$

- Representation via geometry and basis matrices

$$\mathbf{p}(t) = \mathbf{G}\mathbf{M}\mathbf{T}(t)$$

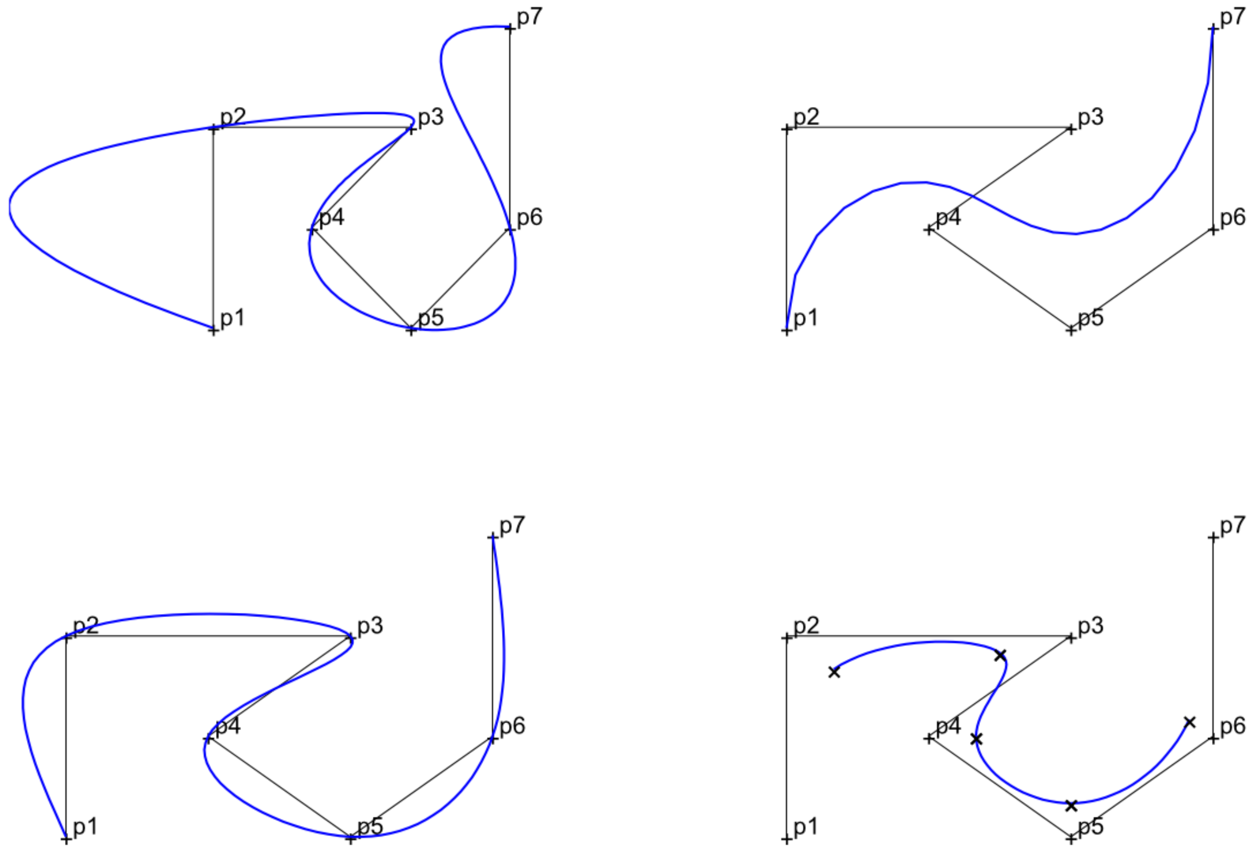
- Representation via blending functions

$$\mathbf{p}(t) = \sum_{i=0}^{n} b_i(t)\mathbf{p}_i, \qquad \sum_{i=0}^{n} b_i(t) = 1$$

Previous time

- Interpolating curves:
  - Lagrange' curve, [+Lagrange' spline]
  - Polynomial (natural) spline
- Approximating curves:
  - Bezier' curve, [+Bezier' spline]
- Specific basis matrices for some cubic curves: $\boldsymbol{M}_L, \boldsymbol{M}_B, \boldsymbol{M}_H$.

Guess a curve!



Next

- B-spline. Non-uniform B-spline.
- Rational B-spline. NURBS.
- Surfaces. Tensor product surfaces.
- Rendering curves and surfaces.
- Curves, surfaces & OpenGL.

Motivation for B-splines

- Suppose we have n control points and we wish to construct a C^2-smooth curve based on them.
- Quiz: What are our options so far?

Motivation for B-splines

- Suppose we have n control points and we wish to construct a C^2-smooth curve based on them.
- Quiz: What are our options so far?
  - *Degree* $n-1$ *polynomial curve (e.g. Lagrange' or Bezier')*
  - *Interpolating cubic spline.*

Motivation for B-splines

- Suppose we have n control points and we wish to construct a C^2-smooth curve based on them.
- Quiz: What are our options so far?
  - *Degree* $n-1$ *polynomial curve (e.g. Lagrange' or Bezier')*
    - Overly complex for large n
  - *Interpolating cubic spline.*
    - Does not allow incremental construction.

(Cubic) B-spline

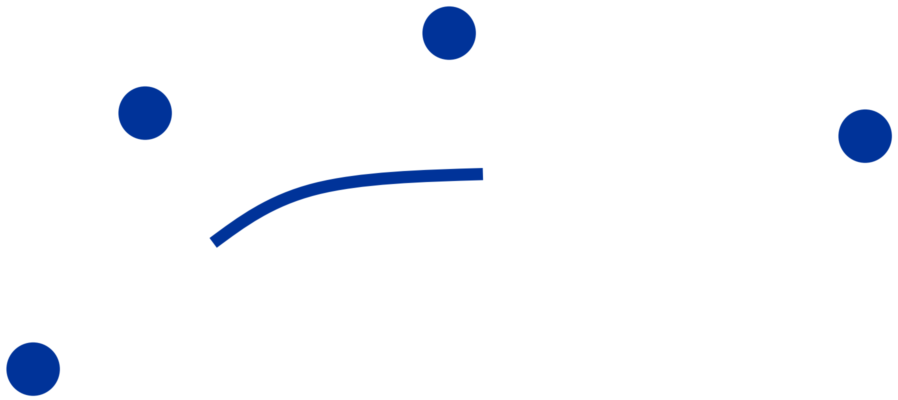
- A piecewise cubic curve
- C^2-smooth at connection points
- Can be constructed incrementally

(Cubic) B-spline

As we know, any cubic curve can be specified using   ?   control points.

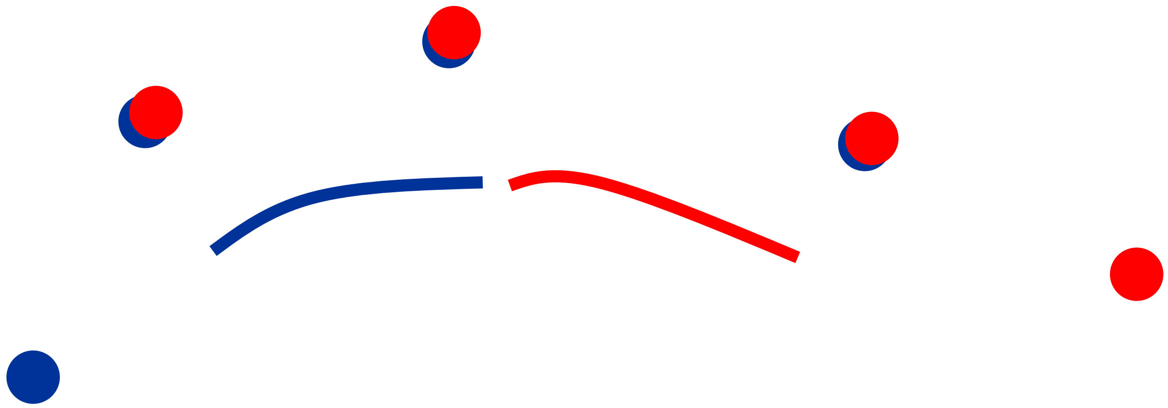
(Cubic) B-spline

As we know, any cubic curve can be specified using 4 control points.



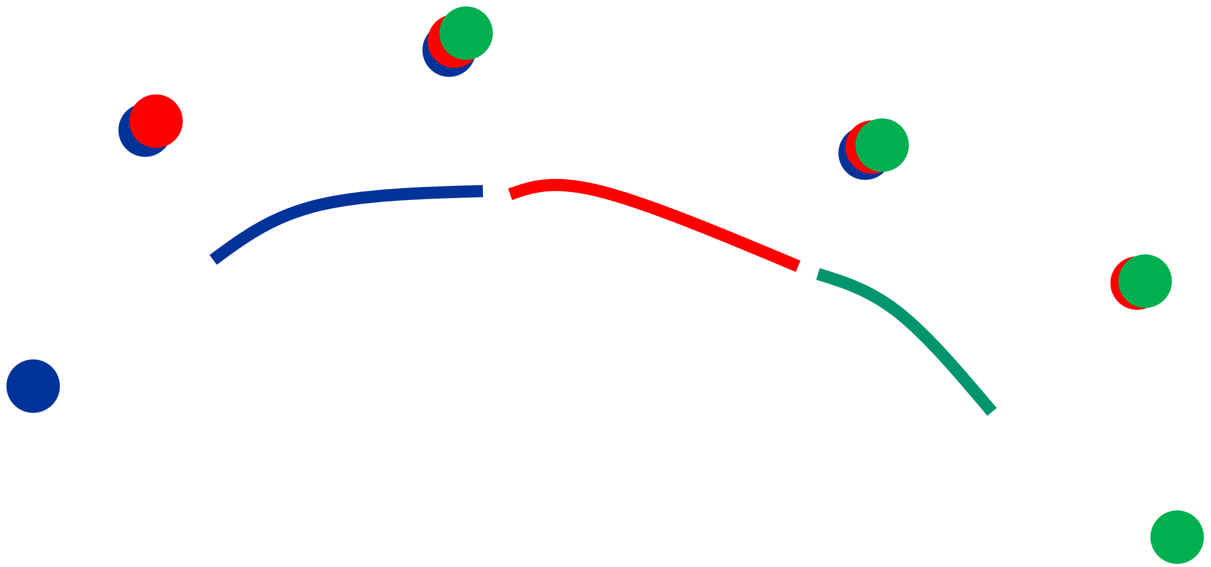
(Cubic) B-spline

The idea of a B spline: each consecutive 4 points specify a cubic piece of the whole curve…



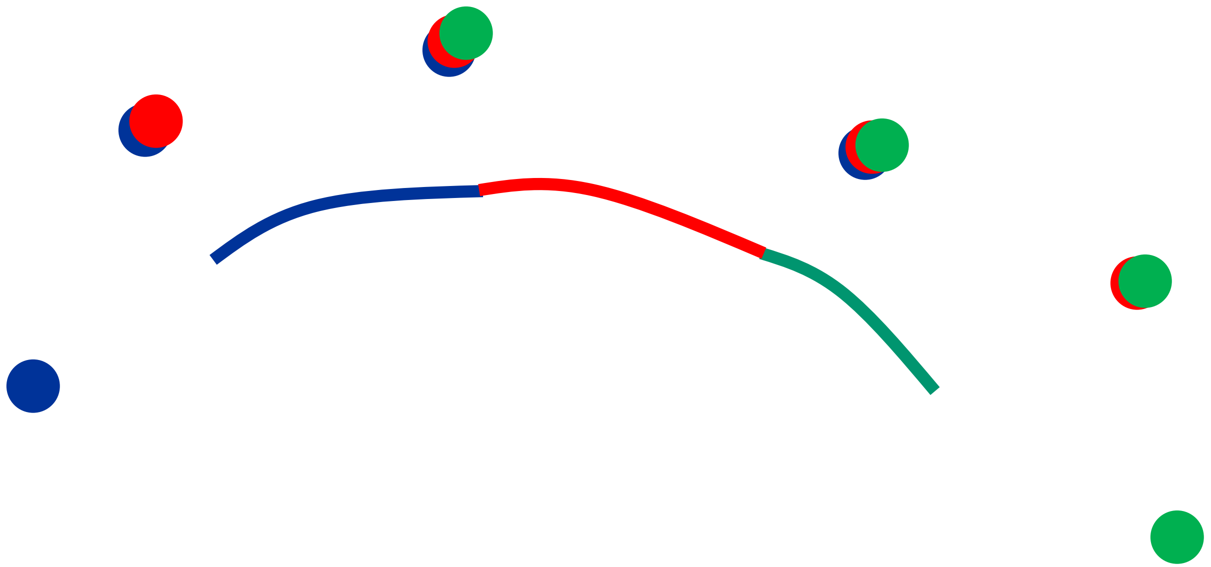
(Cubic) B-spline

The idea of a B spline: each consecutive 4 points specify a cubic piece of the whole curve…



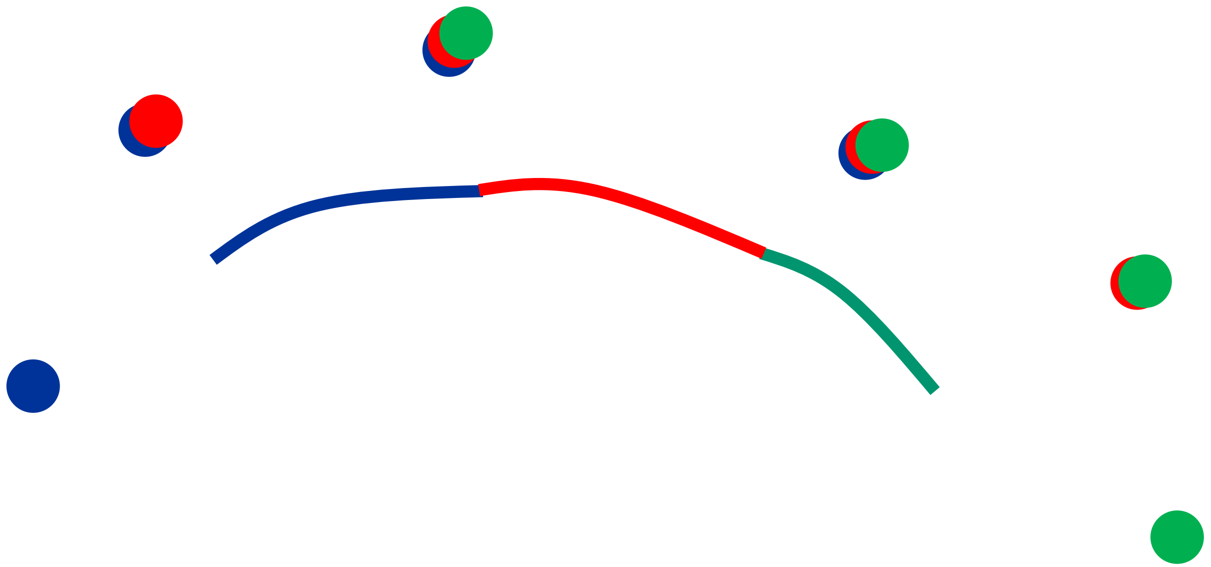
(Cubic) B-spline

… and the pieces connect with C^2-smoothness.



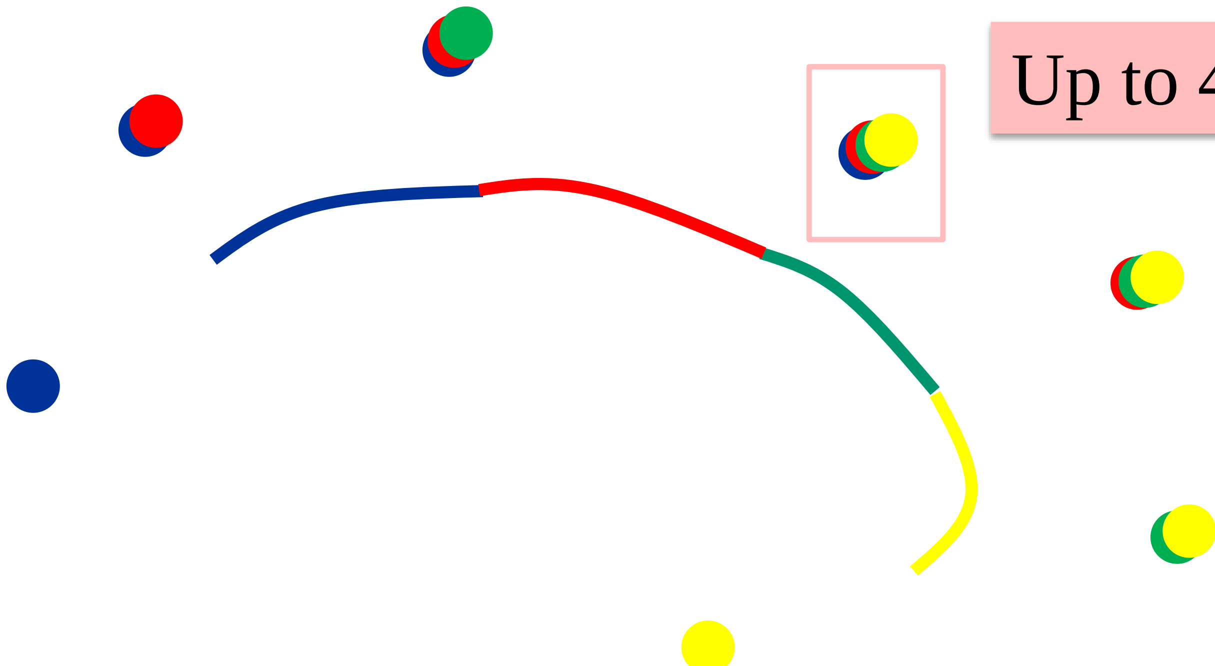
(Cubic) B-spline

Quiz: In such a construction, how many pieces (maximum) of the curve are affected by a single control point?



(Cubic) B-spline

Quiz: In such a construction, how many pieces (maximum) of the curve are affected by a single control point?



Up to 4

(Cubic) B-spline

- Is such a construction at all possible?

(Cubic) B-spline

Represent each piece of a B-spline using blending functions:

$$\boldsymbol{q}(t) = \sum_{i=0}^{3} b_i(t)\boldsymbol{p}_i$$

Let us look for the blending functions that satisfy our needs.

Pick 5 control points and consider two pieces:

$$\boldsymbol{q_1}(t) = b_0(t)\boldsymbol{p}_0 + b_1(t)\boldsymbol{p}_1 + b_2(t)\boldsymbol{p}_2 + b_3(t)\boldsymbol{p}_3$$

$$\boldsymbol{q_2}(t) = b_0(t)\boldsymbol{p}_1 + b_1(t)\boldsymbol{p}_2 + b_2(t)\boldsymbol{p}_3 + b_3(t)\boldsymbol{p}_4$$

(Cubic) B-spline

$$\boldsymbol{q}_{\mathbf{1}}(t) = b_0(t)\boldsymbol{p}_0 + b_1(t)\boldsymbol{p}_1 + b_2(t)\boldsymbol{p}_2 + b_3(t)\boldsymbol{p}_3$$
$$\boldsymbol{q}_{\mathbf{2}}(t) = b_0(t)\boldsymbol{p}_1 + b_1(t)\boldsymbol{p}_2 + b_2(t)\boldsymbol{p}_3 + b_3(t)\boldsymbol{p}_4$$

(Cubic) B-spline

$$\boldsymbol{q_1}(t) = b_0(t)\boldsymbol{p}_0 + b_1(t)\boldsymbol{p}_1 + b_2(t)\boldsymbol{p}_2 + b_3(t)\boldsymbol{p}_3$$
$$\boldsymbol{q_2}(t) = b_0(t)\boldsymbol{p}_1 + b_1(t)\boldsymbol{p}_2 + b_2(t)\boldsymbol{p}_3 + b_3(t)\boldsymbol{p}_4$$

Continuity: $\boldsymbol{q_1}(1) = \boldsymbol{q_2}(0)$

Can only be established if:

$$b_0(1) = 0, \qquad b_3(0) = 0$$
$$b_1(1) = b_0(0)$$
$$b_2(1) = b_1(0)$$
$$b_3(1) = b_2(0)$$

5 equations

(Cubic) B-spline

$$\boldsymbol{q_1}'(t) = b_0'(t)\boldsymbol{p}_0 + b_1'(t)\boldsymbol{p}_1 + b_2'(t)\boldsymbol{p}_2 + b_3'(t)\boldsymbol{p}_3$$
$$\boldsymbol{q_2}'(t) = b_0'(t)\boldsymbol{p}_1 + b_1'(t)\boldsymbol{p}_2 + b_2'(t)\boldsymbol{p}_3 + b_3'(t)\boldsymbol{p}_4$$

C^1-continuity: $\boldsymbol{q_1}'(1) = \boldsymbol{q_2}'(0)$

Can only be established if:

$$b_0'(1) = 0, \qquad b_3'(0) = 0$$
$$b_1'(1) = b_0'(0)$$
$$b_2'(1) = b_1'(0)$$
$$b_3'(1) = b_2'(0)$$

+5 equations

(Cubic) B-spline

$$\boldsymbol{q_1}''(t) = b_0''(t)\boldsymbol{p}_0 + b_1''(t)\boldsymbol{p}_1 + b_2''(t)\boldsymbol{p}_2 + b_3''(t)\boldsymbol{p}_3$$
$$\boldsymbol{q_2}''(t) = b_0''(t)\boldsymbol{p}_1 + b_1''(t)\boldsymbol{p}_2 + b_2''(t)\boldsymbol{p}_3 + b_3''(t)\boldsymbol{p}_4$$

C^2-continuity: $\boldsymbol{q_1}''(1) = \boldsymbol{q_2}''(0)$

Can only be established if:

$$b_0''(1) = 0, \qquad b_3''(0) = 0$$
$$b_1''(1) = b_0'(0)$$
$$b_2''(1) = b_1''(0)$$
$$b_3''(1) = b_2''(0)$$

+5 equations
=15 equations
total

Quiz

- How many parameters do we need to specify to provide the necessary four blending functions b_0, b_1, b_2, b_3?

Quiz

- How many parameters do we need to specify to provide the necessary four blending functions b_0, b_1, b_2, b_3?
  - Each function is a cubic polynomial.

Quiz

- How many parameters do we need to specify to provide the necessary four blending functions b_0, b_1, b_2, b_3?
  - Each function is a cubic polynomial.
  - I.e. 4 coefficients per function = 16 total.

Quiz

- How many parameters do we need to specify to provide the necessary four blending functions b_0, b_1, b_2, b_3?
  - Each function is a cubic polynomial.
  - I.e. 4 coefficients per function = 16 total.
- This leaves 1 degree of freedom. What could it be?

(Cubic) B-spline

Hence,

- 15 equations for ensuring C^2-continuity,
- plus an equation for ensuring *scale*:

$$b_0(0) + b_1(0) + b_2(0) + b_3(0) = 1$$

Result in a unique solution: the (cubic) B-spline curve.

The same logic applies for higher-order B-splines.

(Cubic) B-spline

$$b_0(t) = \frac{(1-t)^3}{6}$$

$$b_1(t) = \frac{4 - 6t^2 + 3t^3}{6}$$

$$b_2(t) = \frac{1 + 3t + 3t^2 - 3t^3}{6}$$

$$b_3(t) = \frac{t^3}{6}$$

(Cubic) B-spline

$$b_0(t) = \frac{(1-t)^3}{6}$$

$$b_1(t) = \frac{4 - 6t^2 + 3t^3}{6}$$

$$b_2(t) = \frac{1 + 3t + 3t^2 - 3t^3}{6}$$

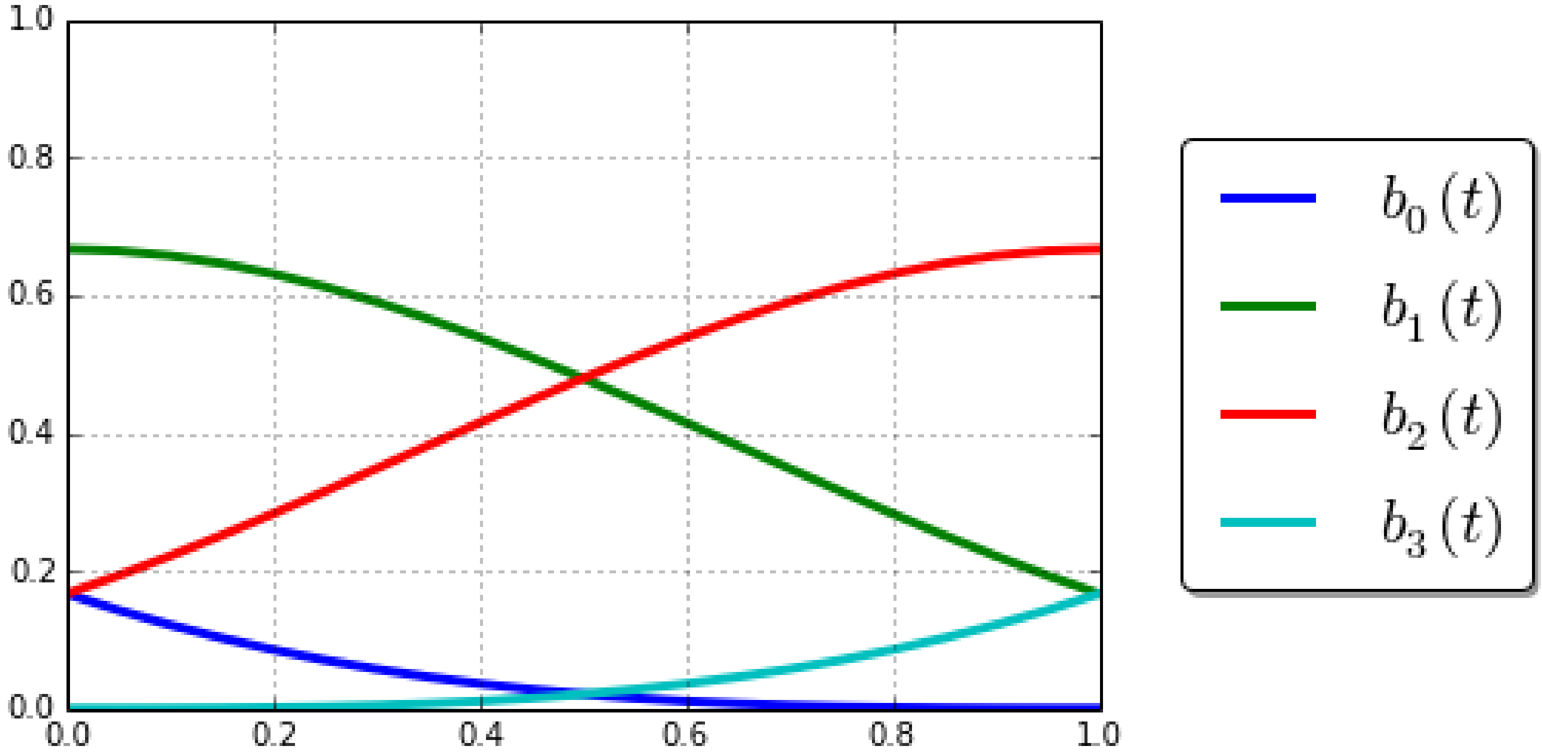
$$b_3(t) = \frac{t^3}{6}$$

Quiz: What is the corresponding basis matrix?

(Cubic) B-spline

$$M_{BS} = \frac{1}{6} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

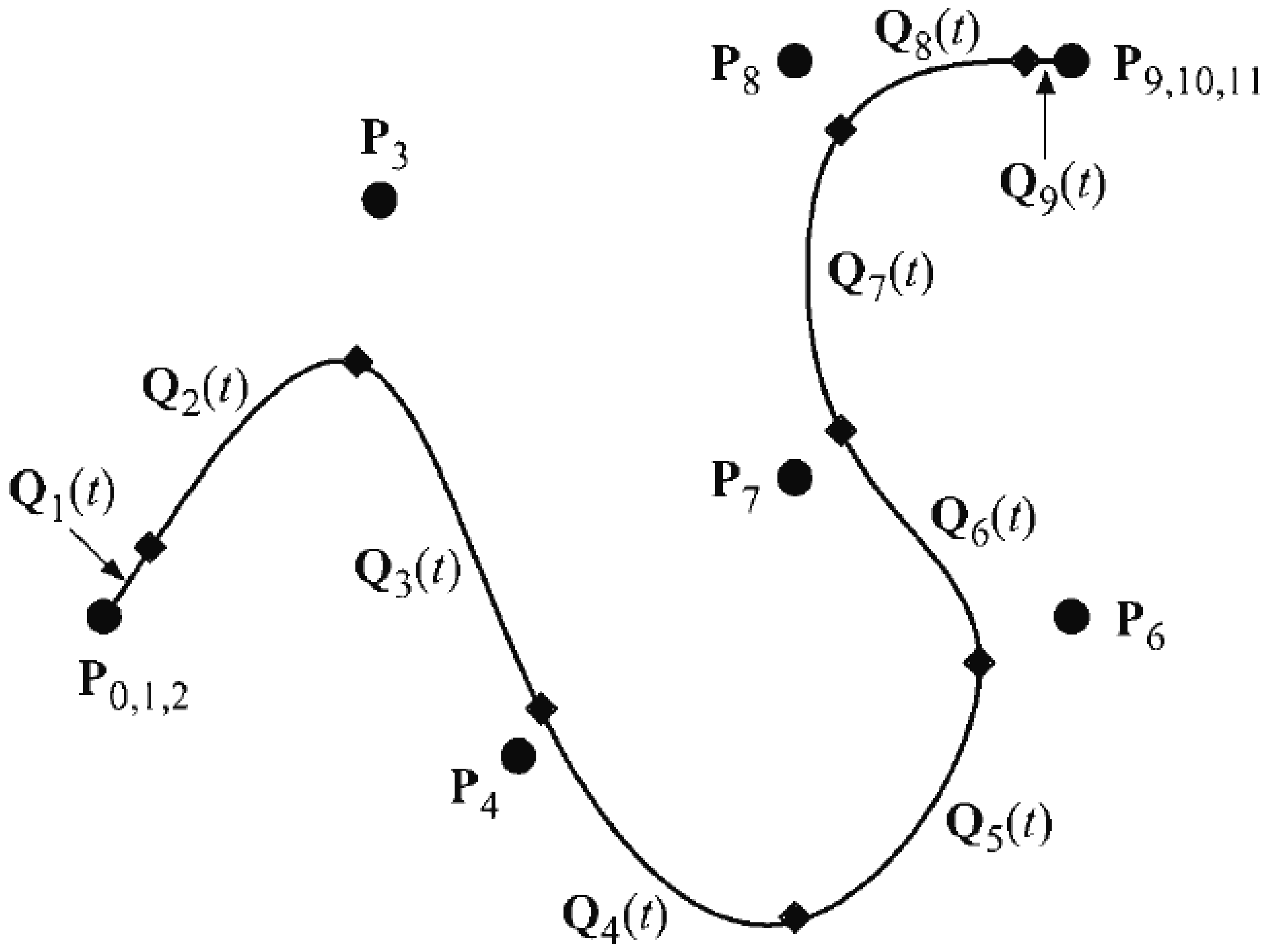
(Cubic) B-spline



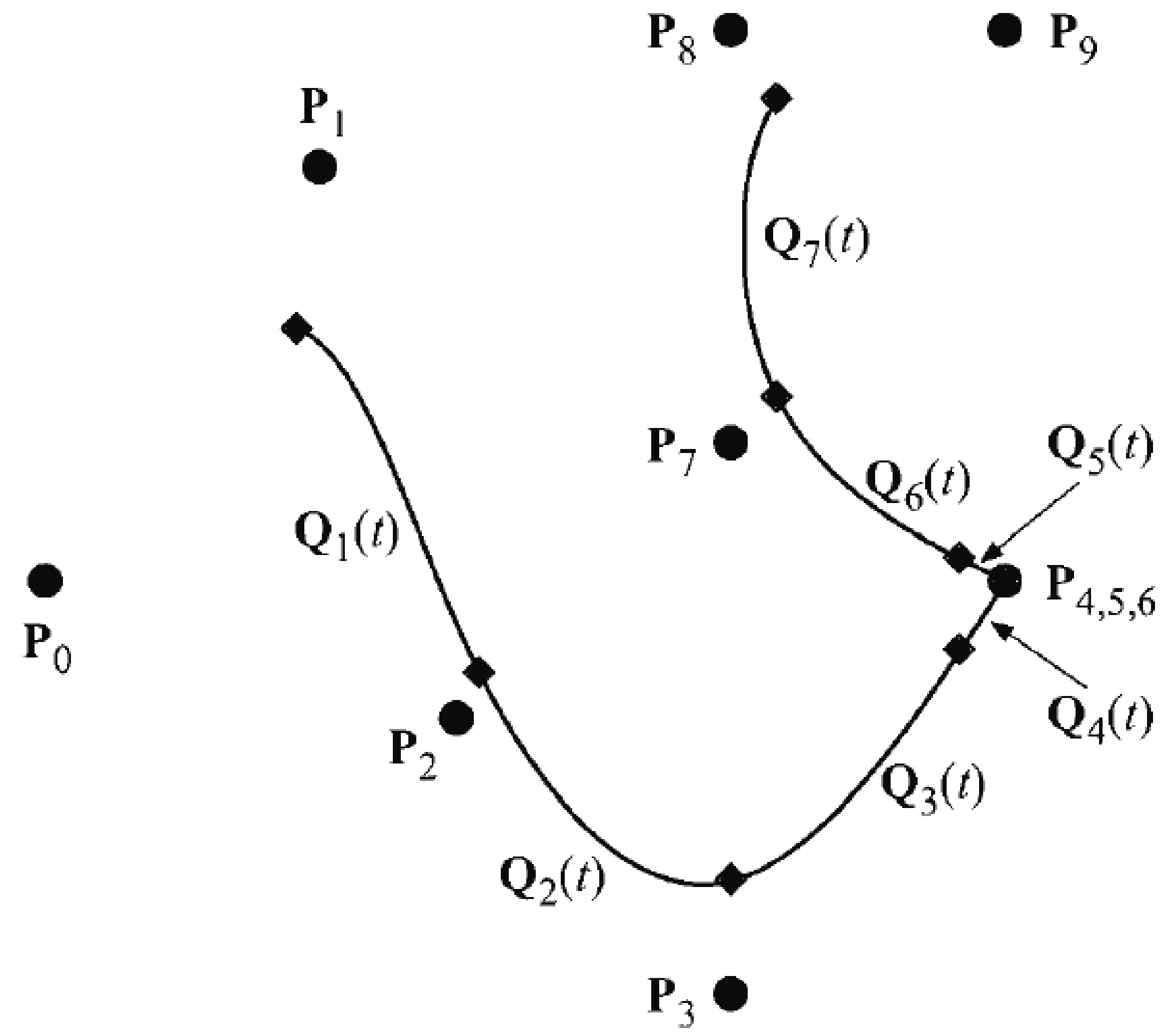
Repeating control points

- Control points can be repeated
  - Each repetition loses one degree of smoothness
  - Repeating a control point 3 times makes the curve pass this point
  - It is common to repeat the first and the last control points 3 times.

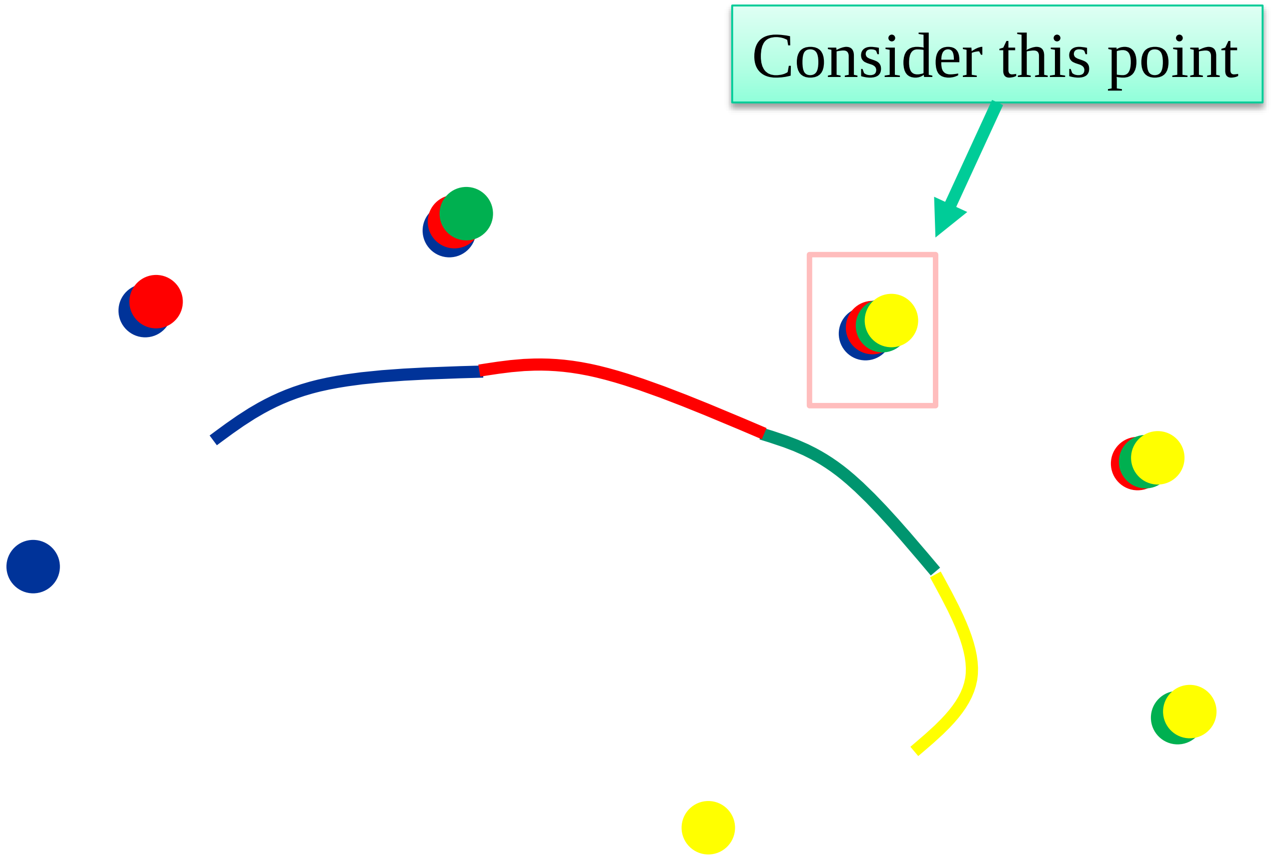
Repeating control points



Repeating control points



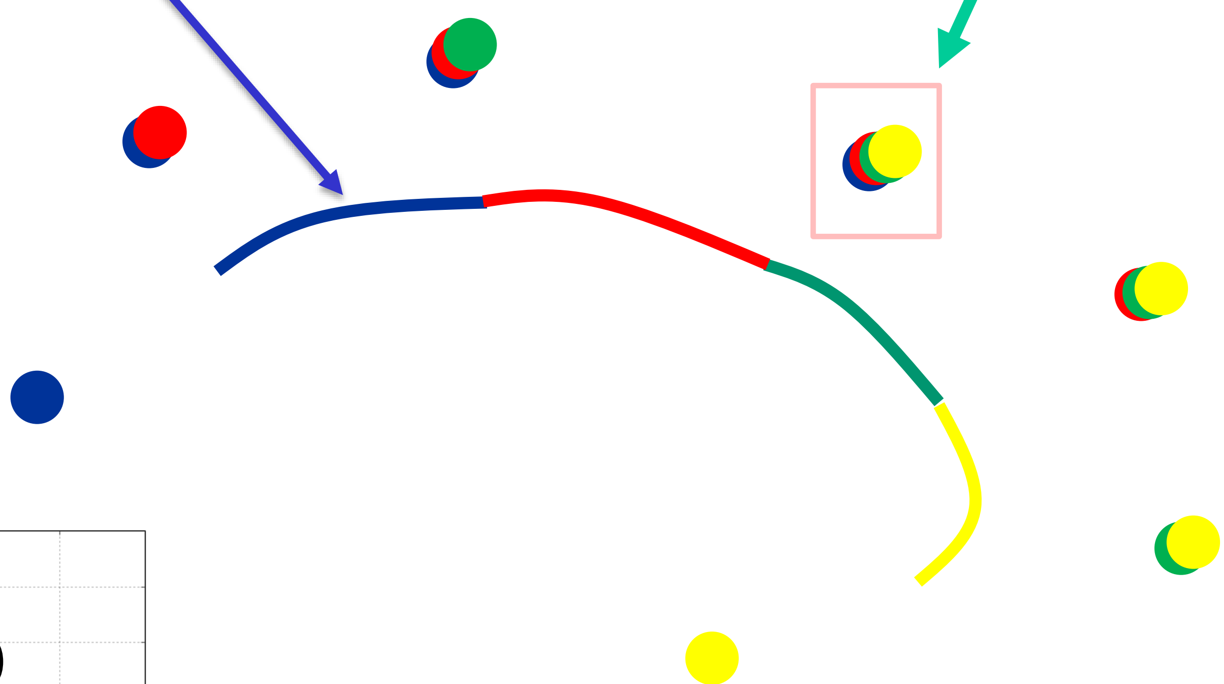
B-spline = Basis spline

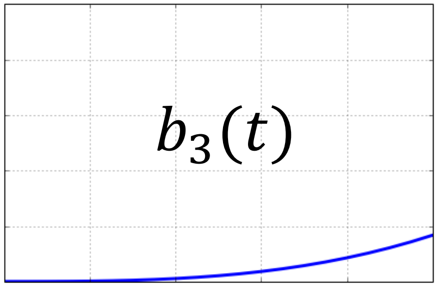


B-spline = Basis spline

In this segment of the curve it is weighed by b_3.

Consider this point

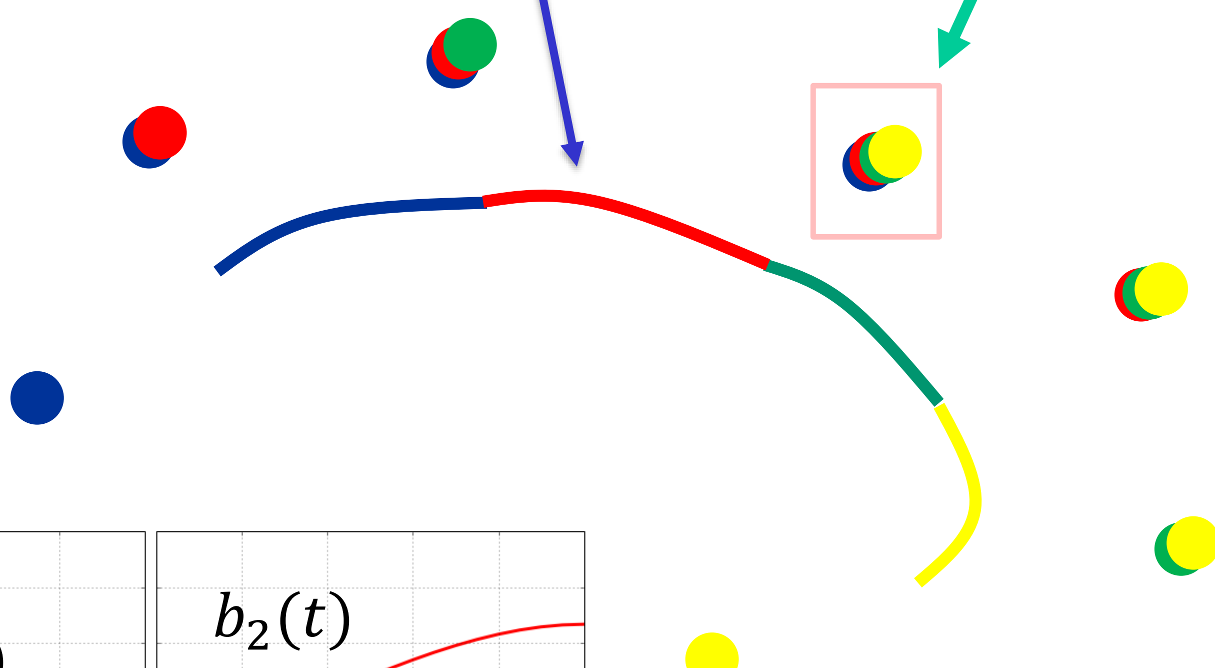


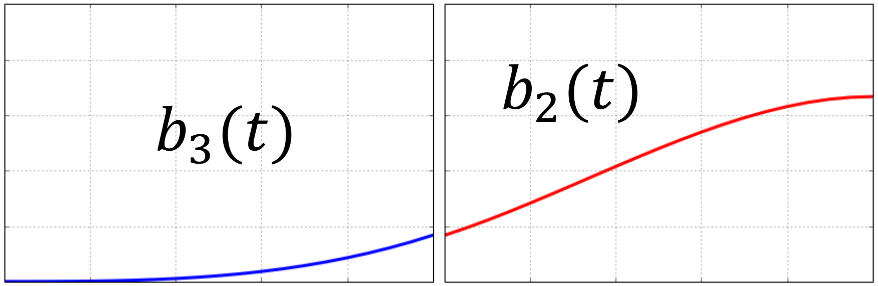


B-spline = Basis spline

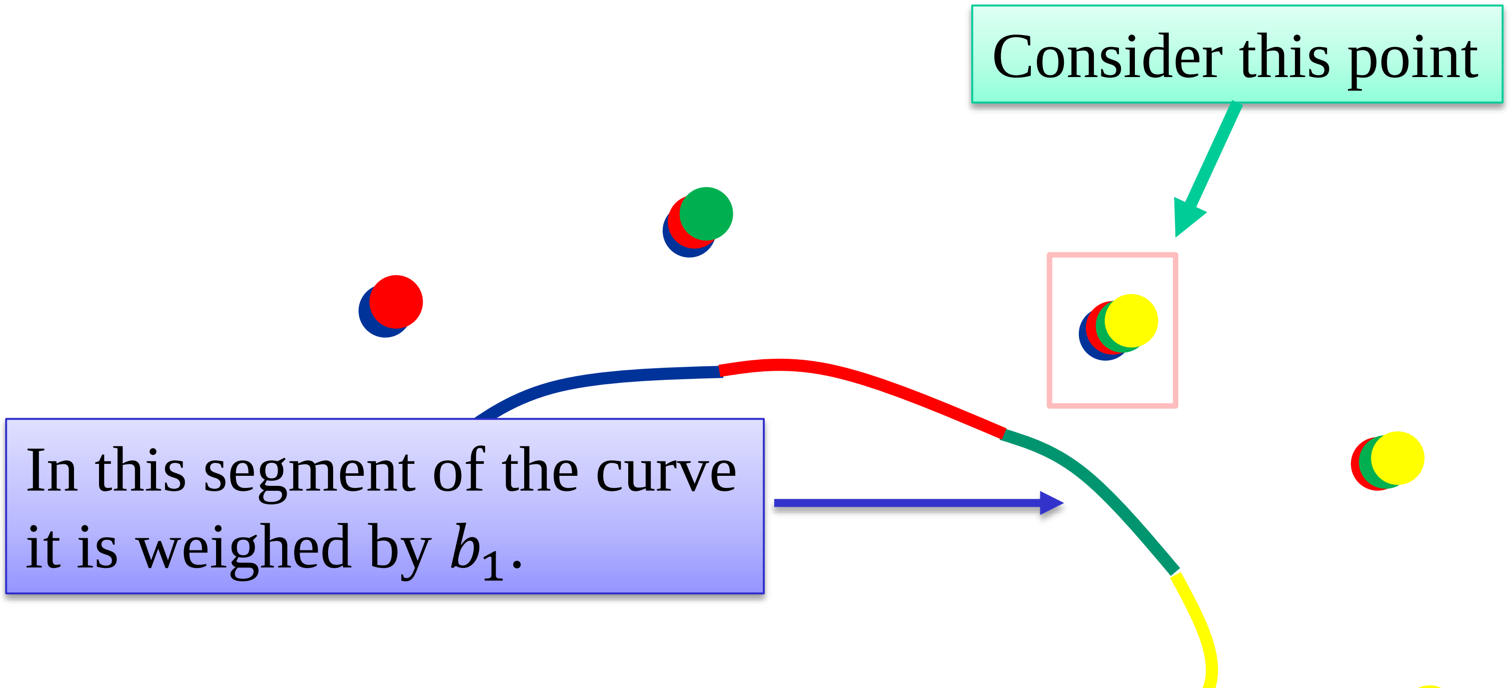
In this segment of the curve it is weighed by b_2.

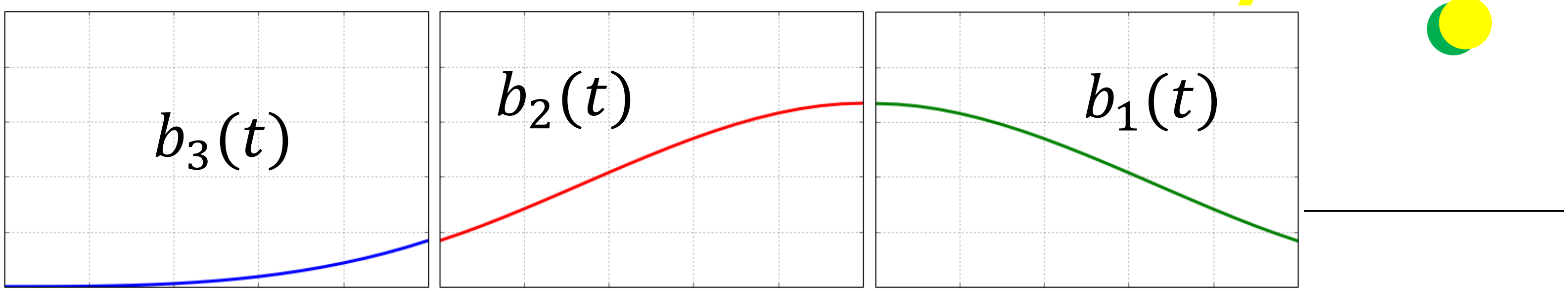
Consider this point



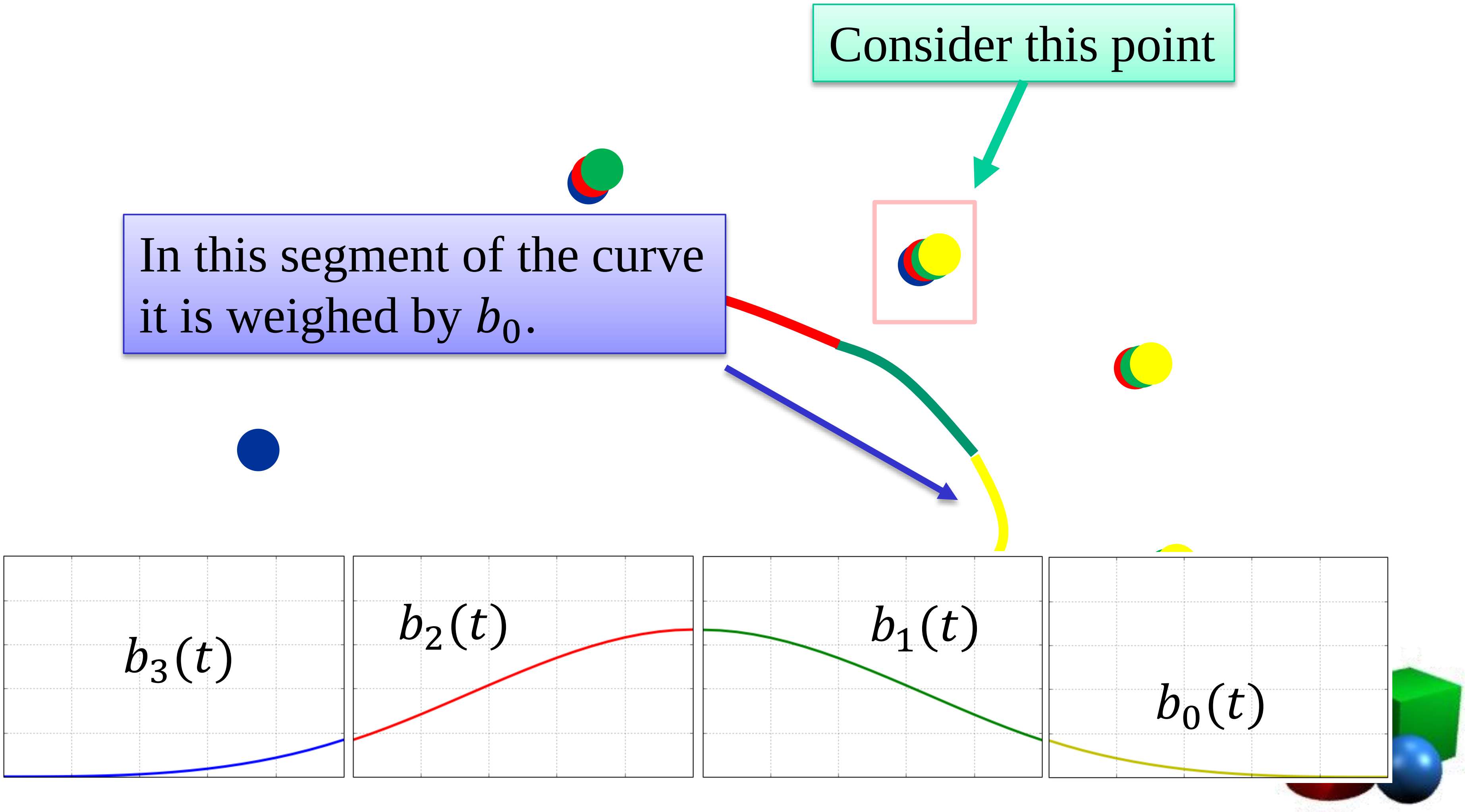


B-spline = Basis spline





B-spline = Basis spline



Basis function

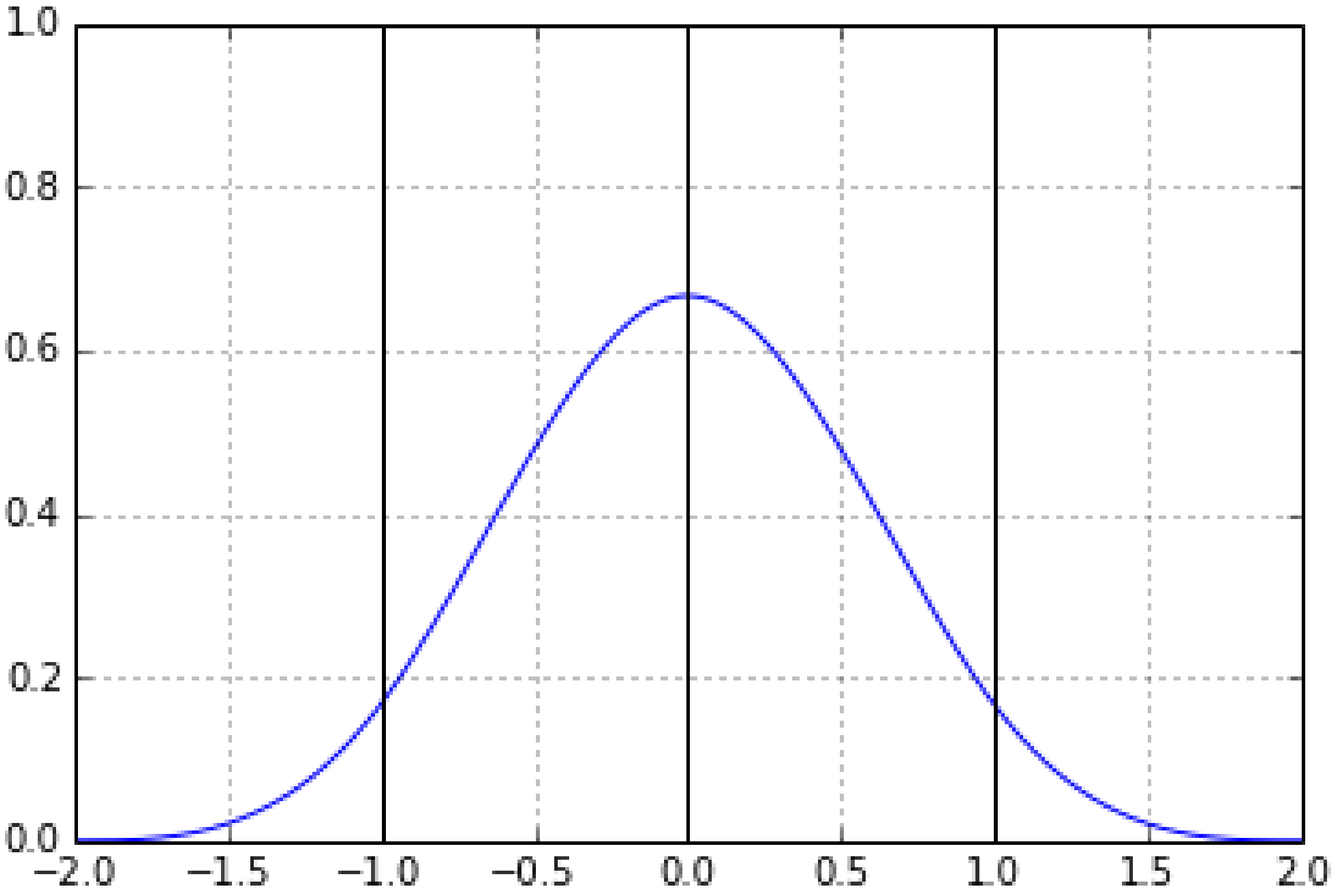
- Let us parameterize the whole curve as a single function, defined over $\mathrm{t} \in [1, n-1]$:

$$\boldsymbol{q}(t) = \boldsymbol{q_i}(t-i), \text{ for } i \leq t \leq i+1$$

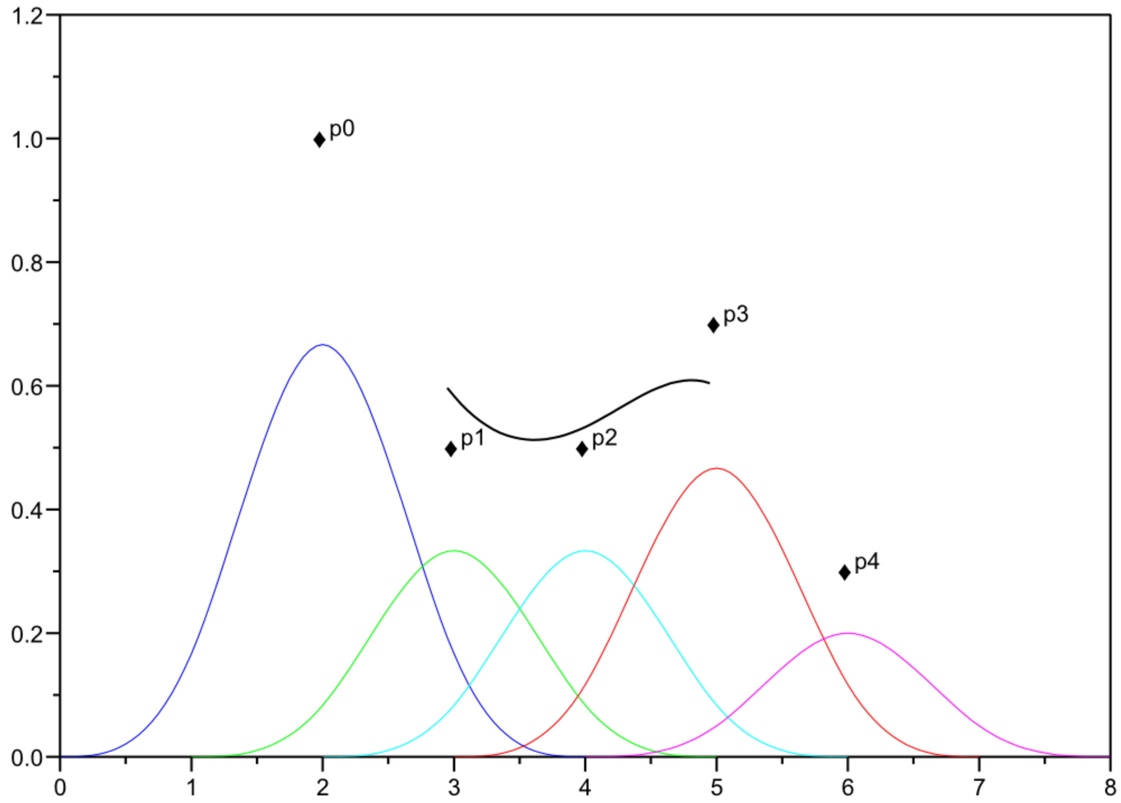
- In this function, each point $\boldsymbol{p}_i$ spreads its influence over the region $[i-2, i+2]$.

- In this region it's contribution is defined by the *basis function* $N(t)$.

Basis function



Basis functions



$$\boldsymbol{q}(t) = \sum_{i} N(t - i)\boldsymbol{p}_i$$

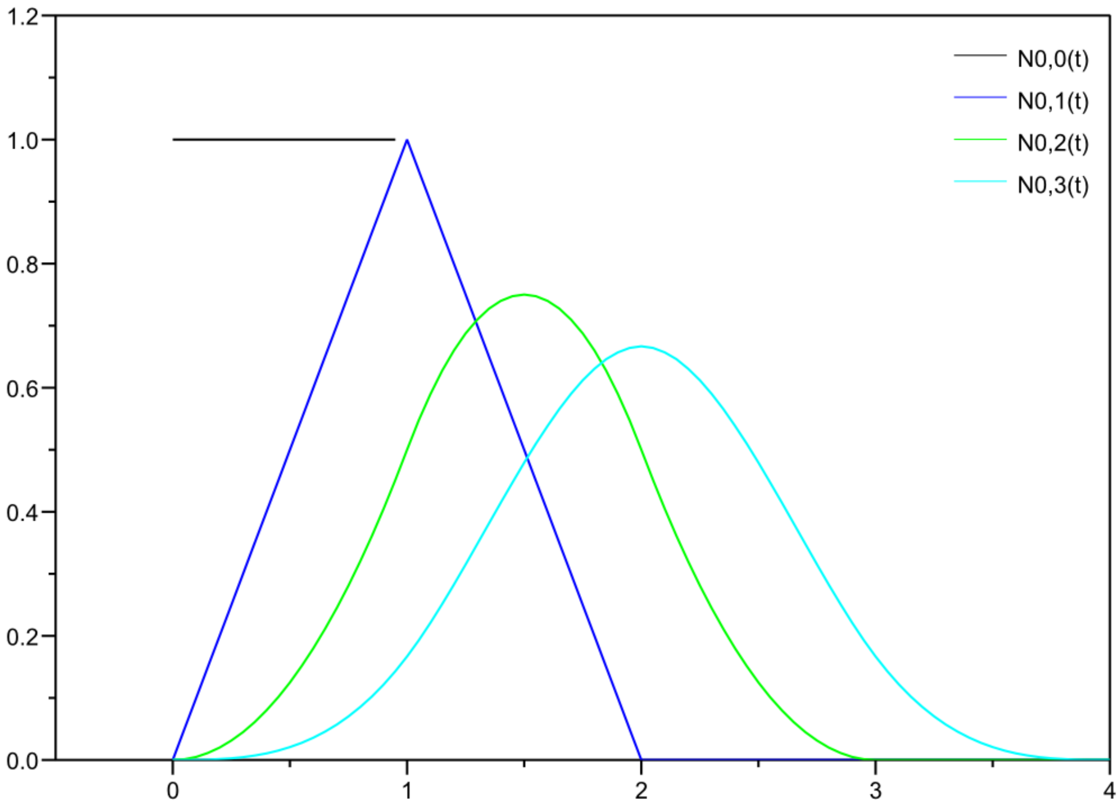
General B-splines

- We only considered cubic B-splines, but the same logic applies to B-splines of arbitrary degree.

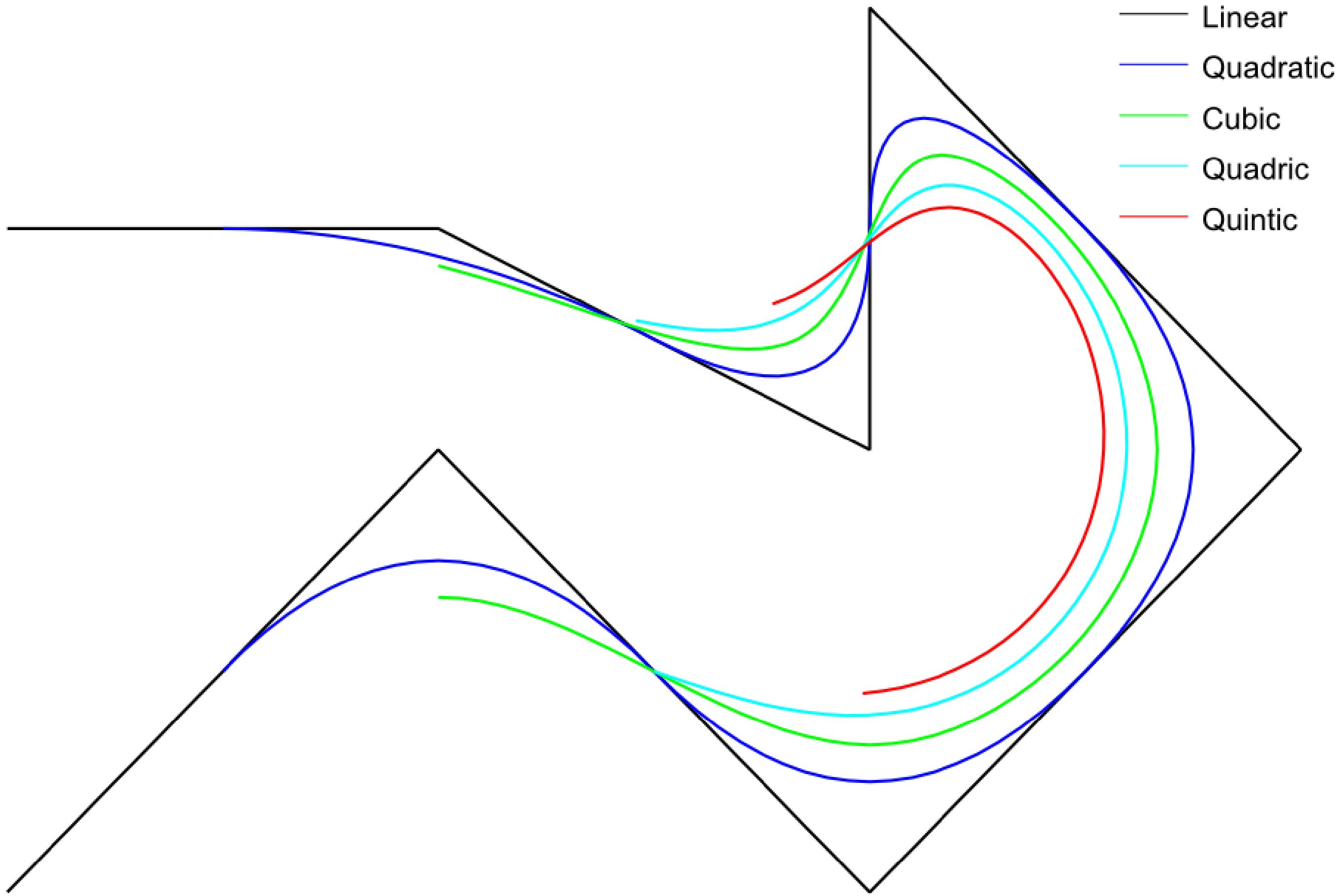
- A B-spline of degree k is C^{k-1}-smooth, and is expressed using the basis functions $N_{i,k}(t)$:

$$\boldsymbol{q}(t) = \sum_{i} N_{i,k}(t)\boldsymbol{p}_i$$

General basis functions



General B-splines



Cox-De Boor' equations

- B-spline basis functions, like the Bernstein' polynomials, can be constructed recursively:

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

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

- This leads to an efficient B-spline evaluation algorithm *(De Boor algorithm)* similar to that for the Bezier' curve *(De Casteljau's algorithm)*.

Non-uniform B-splines

- In B-splines that we defined so far:
  - The parameter region [1,2] is affected by control points $\boldsymbol{p_0}, \boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$
  - The parameter region [2,3] is affected by control points $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}, \boldsymbol{p_4},$
  - etc
- Sometimes we would like to change at which parameter regions each control point has effect.
  - E.g. we might want the curve to “reach” some control points faster and “stay there longer”.
  - … or we might need more control points in a particularly curvy region.
  - … or we might want to “insert” control points

Non-uniform B-splines

- For that let us just specify a list of *knot values* along with the control points:

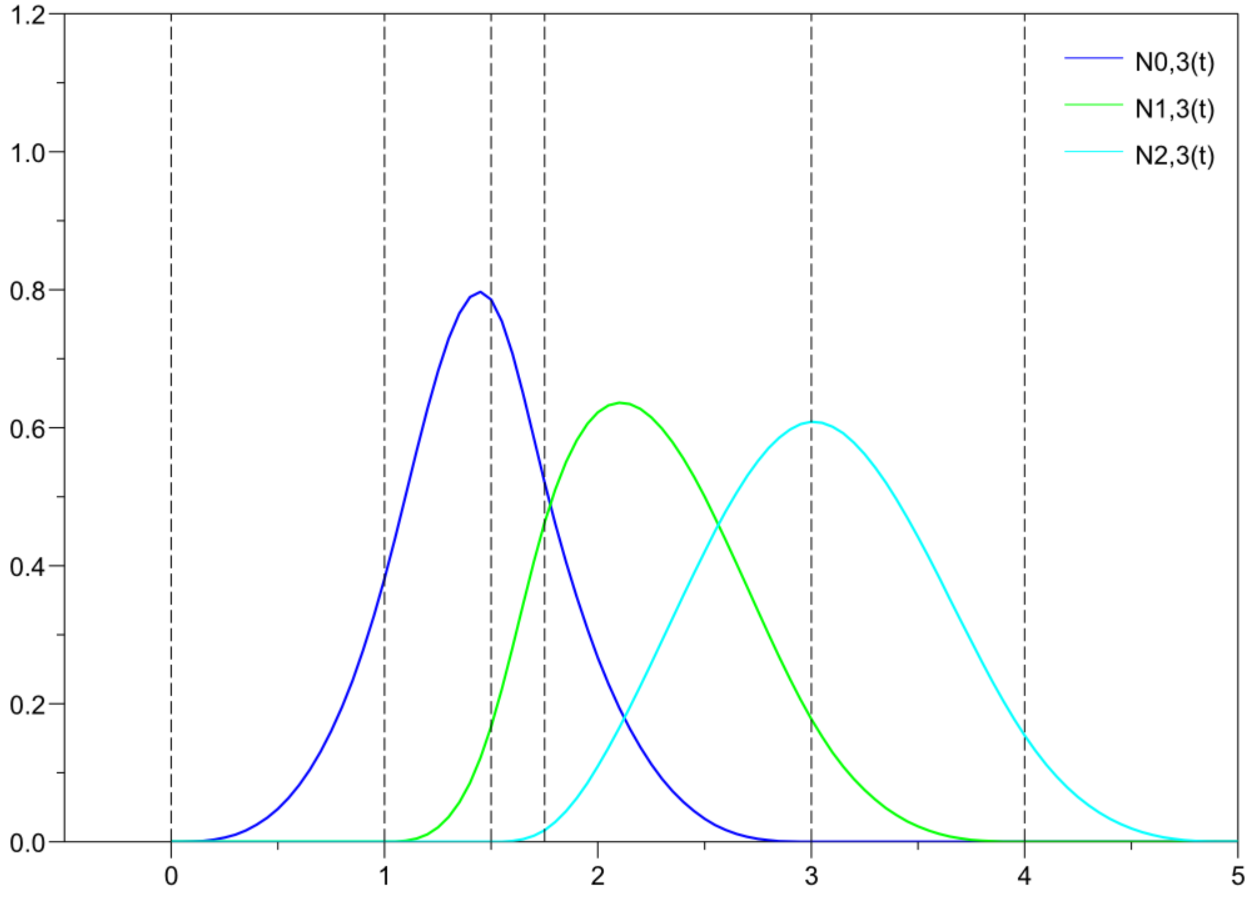
$$t_0, t_1, \dots, t_{n+k+1}$$

- The Cox-De Boor recurrence becomes:

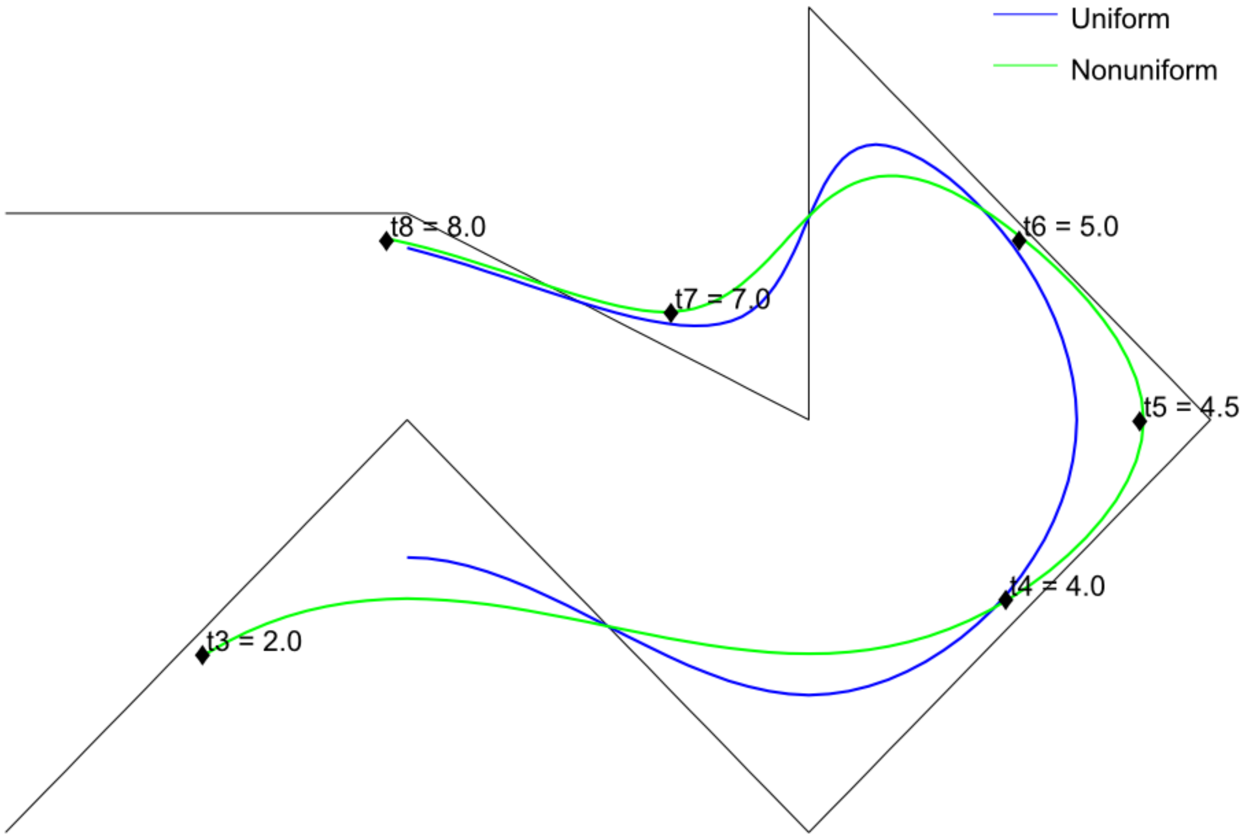
$$N_{i,0}(t) = \begin{cases} 1 & t \in [t_i, t_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

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

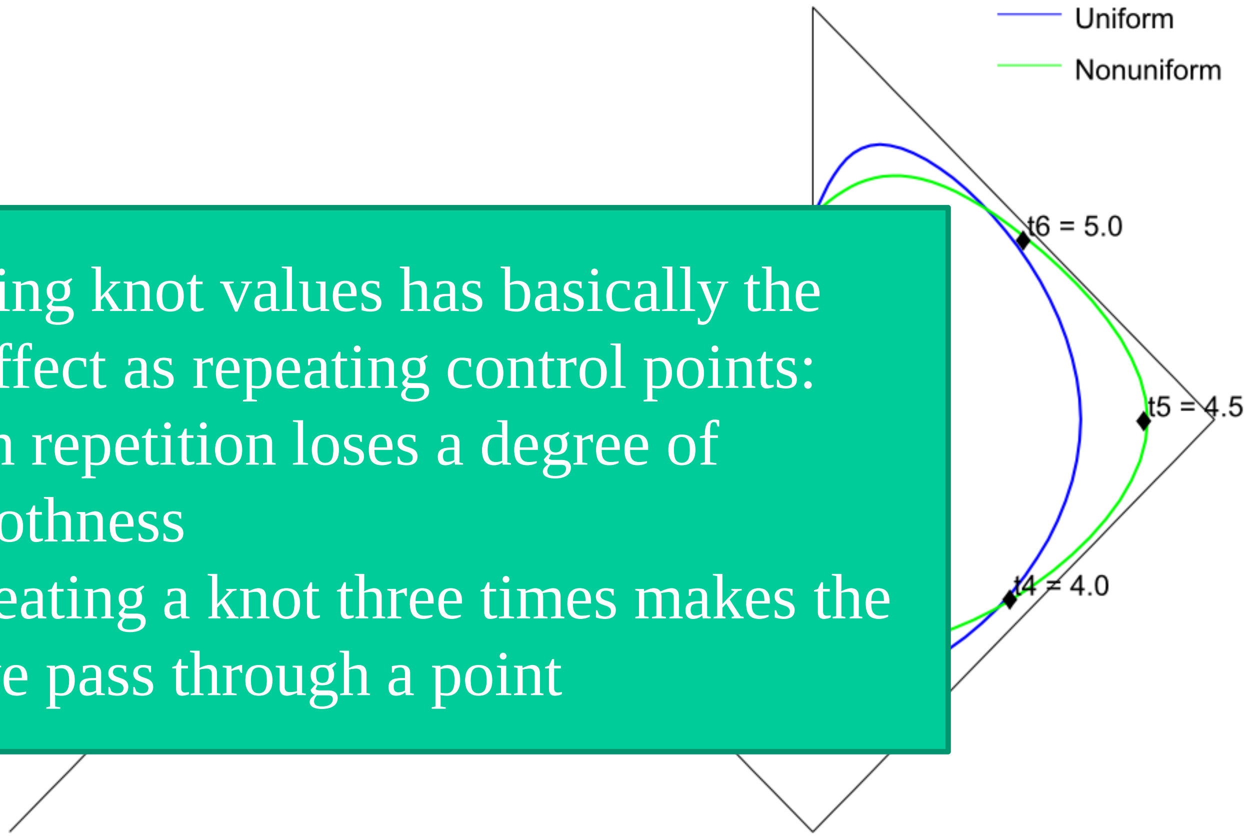
Non-uniform B-spline basis functions



Non-uniform B-splines



Non-uniform B-splines



Repeating knot values has basically the same effect as repeating control points:

- Each repetition loses a degree of smoothness
- Repeating a knot three times makes the curve pass through a point

http://www.saccade.com/writing/graphics/KnotVectors.pdf

Rational B-spline

- In 3D graphics we like to work with homogeneous coordinates

$$(xw, yw, zw, w)^T$$

- It is therefore natural to construct curves as a 4 dimensional curves in homogeneous coordinates with 4-dimensional control points.

$$\boldsymbol{p}_i = (x_i w_i, y_i w_i, z_i w_i, w_i)$$

Rational B-spline

- Curves, constructed in homogeneous space will respect perspective transformations (because those are linear in homogeneous space).

$$\begin{pmatrix} xw \\ yw \\ zw \\ w \end{pmatrix} = \sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

Rational B-spline

$$\begin{pmatrix} xw \\ yw \\ zw \\ w \end{pmatrix} = \sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

which can be rewritten as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{w}$$

Rational B-spline

hence

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

hence

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

Rational B-spline

hence

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

This is exactly like the usual B-spline, but each point now has a *weight* w_i.

hence

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

Rational B-spline

hence

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

.. and once we added weights, the weighted basis functions won’t necessarily add up to 1, so we have to renormalize.

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

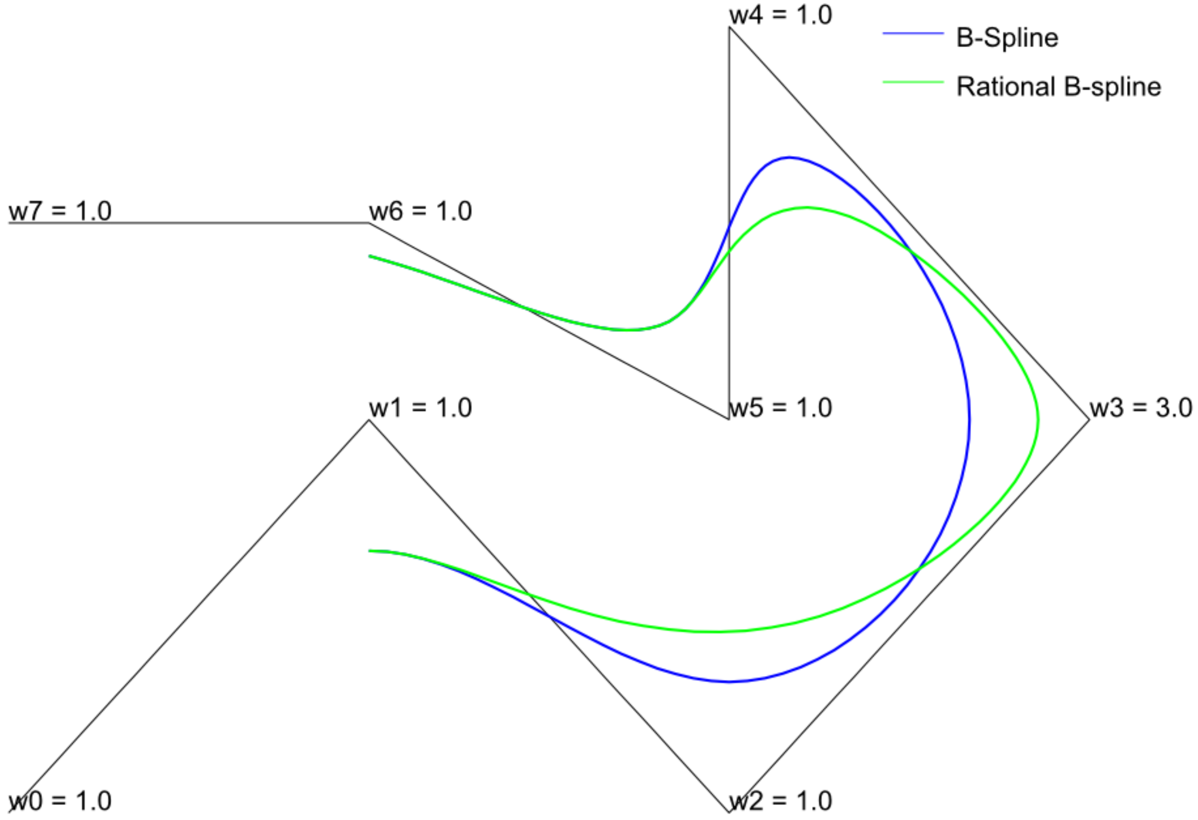
Rational B-spline

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

- A rational B-spline is invariant wrt affine and perspective transformations.

- In addition, we can use w_i to provide a "weight" for each control point.

Rational B-spline



NURBS

- A rational B-spline with a non-uniform knot vector is called NURBS (Non-uniform Rational B-Spline).
- NURBS offer a lot of flexibility in defining curves and surfaces. You can define both standard shapes (spheres, cylinders, etc) as well as custom models.
- NURBS are a de-facto standard modeling tool in CAD as well as 3D art.

Summary: Curves

- Interpolating
  - Lagrange (not much used)
  - Natural spline (CAD/CAM, trajectories)
- Approximating
  - Bezier’ (Photoshop/GIMP/MSWord, …)
  - B-spline (trajectories)
  - NURBS (CAD/CAM, Blender/Maya, …)

Next

- B-spline. Non-uniform B-spline.
- Rational B-spline. NURBS.
- Surfaces. Tensor product surfaces.
- Rendering curves and surfaces.
- Curves, surfaces & OpenGL.

Surfaces

- The theory is largely similar to that of curves:
  - Parametric representation: $\boldsymbol{p}(u, v) = \boldsymbol{f}(u, v)$
  - Polynomial surface:
    - $x(u, v), y(u, v), z(u, v)$ are polynomial in u, v:

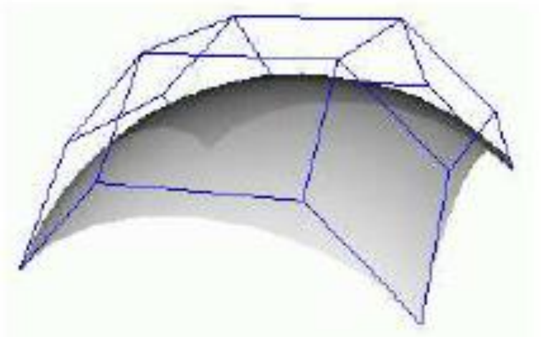
$$x(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{xij} u^i v^j = \mathbf{U}_n(u)^T \mathbf{C}_x \mathbf{V}_n(v)$$

$$\mathbf{U}_n(u) = (1, u, u^2, \ldots, u^n)^T$$

$$\mathbf{V}_n(v) = (1, v, v^2, \ldots, v^n)^T$$

Control points

To construct a degree d surface we need $(d+1)^2$ control points:



Like with curves, most widespread are cubic and piecewise-cubic surfaces.
A cubic surface patch requires 16 control points

Blending functions

Like curves, surfaces can be represented as a linear combination of control points via *blending functions*.

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

Tensor product surfaces

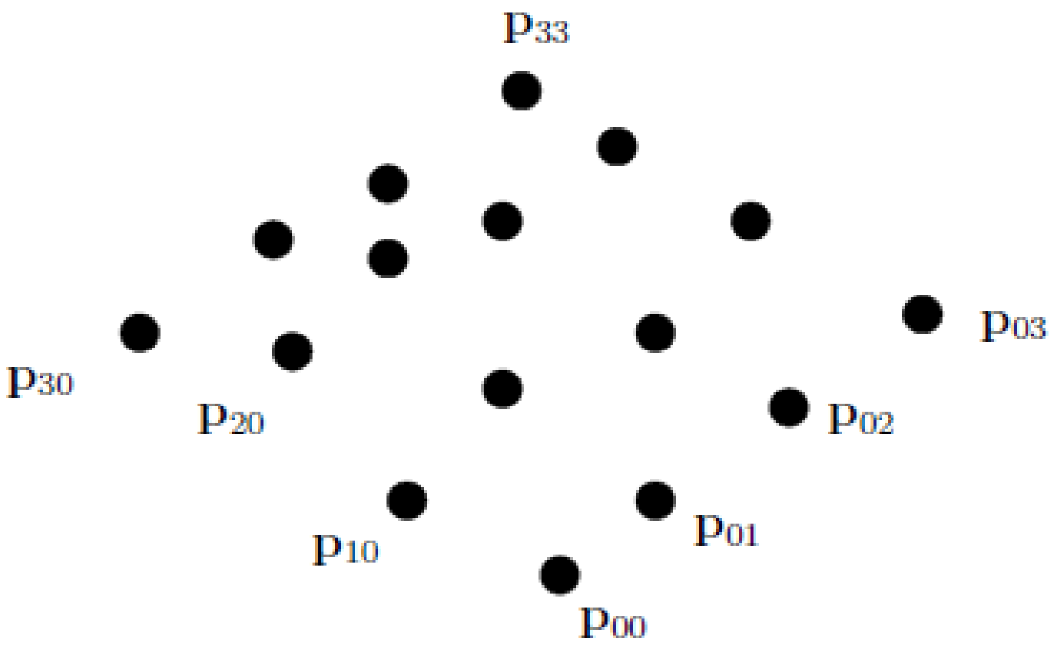
The easiest way to construct a blending function for a surface is to simply take a product of blending functions for some curve:

$$b_{ij}(u, v) = b_i(u)b_j(v)$$

The resulting surface is called *tensor product surface*.

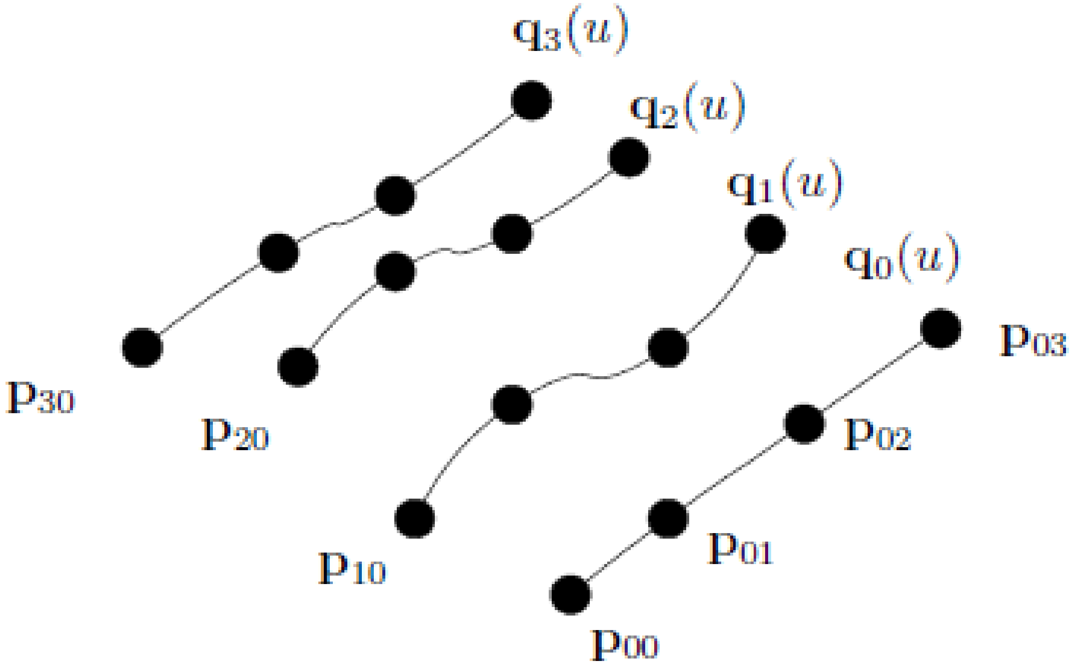
Tensor product surfaces

Consider 16 control points:



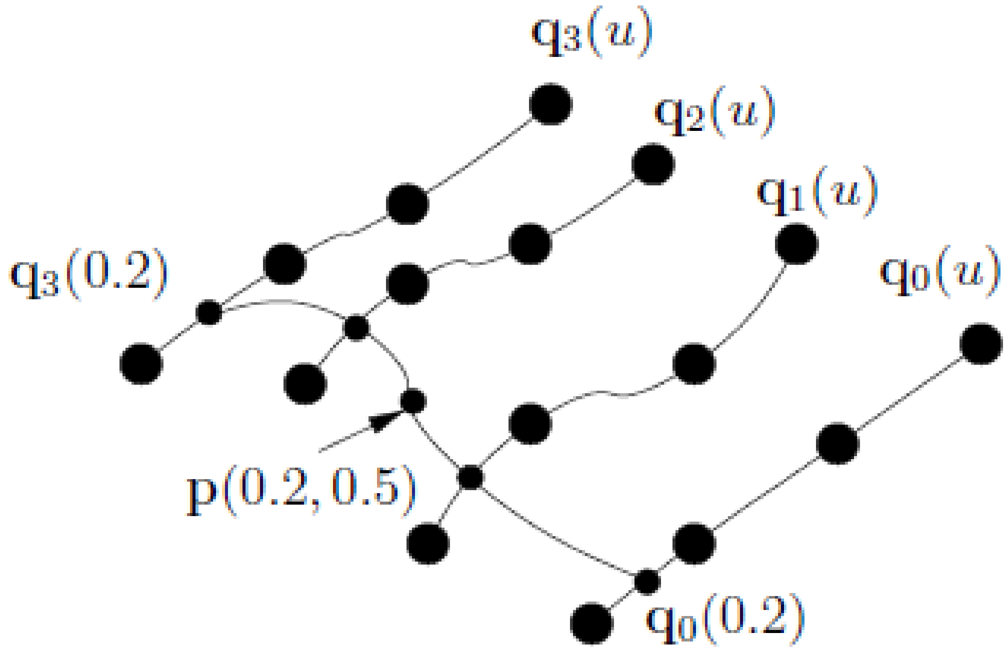
Tensor product surfaces

Start by constructing four curves $\boldsymbol{q_i}(u)$ as follows:



Tensor product surfaces

Now for each fixed u make a curve $\boldsymbol{p}(u, v)$, using $\boldsymbol{q_i}(u)$ as control points:



Tensor product surfaces

- Thus:

$$\mathbf{p}(u,v) = \sum_{j=0}^{3} b_j(v)\mathbf{q}_j(u) =$$

$$= \sum_{j=0}^{3} b_j(v) \left(\sum_{i=0}^{3} b_i(u)\mathbf{p}_{ij} \right) =$$

$$= \sum_{i} \sum_{j} b_i(u) b_j(v) \mathbf{p}_{ij}$$

Tensor product surfaces

- Obviously, this construction can be performed for any b_i, so this way we get:
  - Lagrange interpolating surface
  - Interpolating spline surface
  - Bezier surface
  - B-spline surface
  - NURBS surface

Next

- **B-spline. Non-uniform B-spline.**
- **Rational B-spline. NURBS.**
- **Surfaces. Tensor product surfaces.**
- **Rendering curves and surfaces.**
- **Curves, surfaces & OpenGL.**