B-splines

- B-splines automatically take care of continuity, with exactly one control vertex per curve segment
- Many types of B-splines: degree may be different (linear, quadratic, cubic,…) and they may be uniform or non-uniform
  - We will only look closely at uniform B-splines
- With uniform B-splines, continuity is always one degree lower than the degree of each curve piece
  - Linear B-splines have C^0 continuity, cubic have C^2, etc

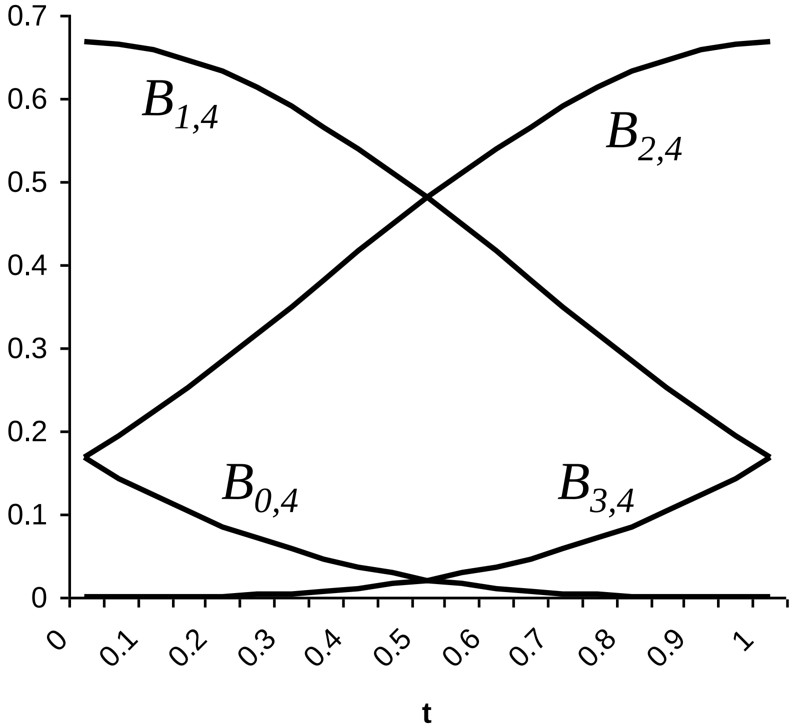
Uniform Cubic B-spline on [0,1)

- Four control points are required to define the curve for $0 \leq t < 1$ (t is the parameter)
  - Not surprising for a cubic curve with 4 degrees of freedom
- The equation looks just like a Bezier curve, but with different basis functions
  - Also called *blending functions* - they describe how to blend the control points to make the curve

$$x(t) = \sum_{i=0}^{3} P_i B_{i,4}(t)$$

$$= P_0 \frac{1}{6}\left(1 - 3t + 3t^2 - t^3\right) + P_1 \frac{1}{6}\left(4 - 6t^2 + 3t^3\right) + P_2 \frac{1}{6}\left(1 + 3t + 3t^2 - 3t^3\right) + P_3 \frac{1}{6}\left(t^3\right)$$

Basis Functions on [0,1)



- Does the curve interpolate its endpoints?
- Does it lie inside its convex hull?

$$x(t) = P_0 \frac{1}{6}\left(1 - 3t + 3t^2 - t^3\right) + P_1 \frac{1}{6}\left(4 - 6t^2 + 3t^3\right) + P_2 \frac{1}{6}\left(1 + 3t + 3t^2 - 3t^3\right) + P_3 \frac{1}{6}\left(t^3\right)$$

Uniform Cubic B-spline on [0,1)

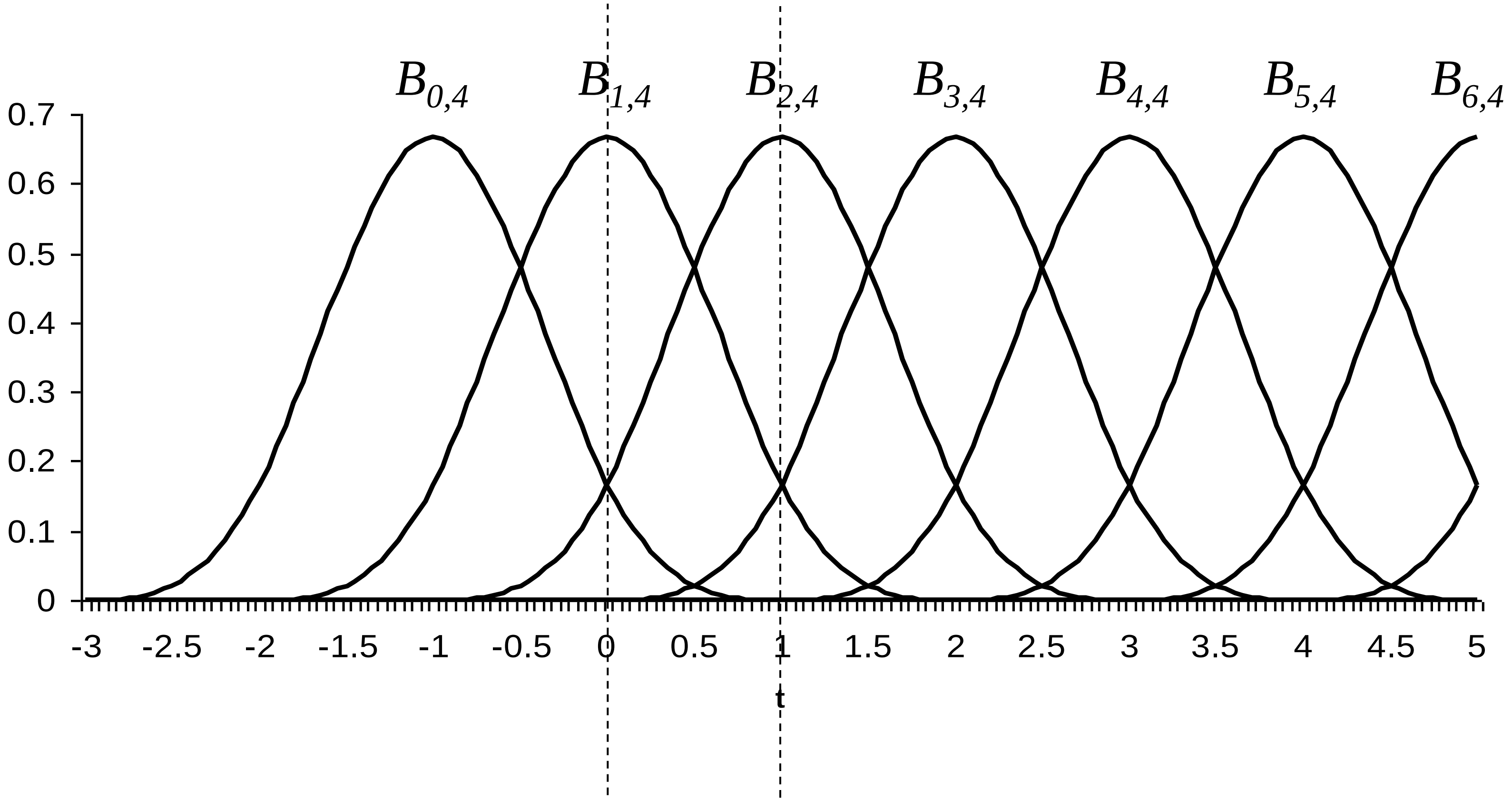
- The blending functions sum to one, and are positive everywhere
  - The curve lies inside its convex hull
- The curve does not interpolate its endpoints
  - Requires hacks or non-uniform B-splines
- There is also a matrix form for the curve:

$$x(t) = \frac{1}{6}\begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Uniform Cubic B-splines on [0,m)

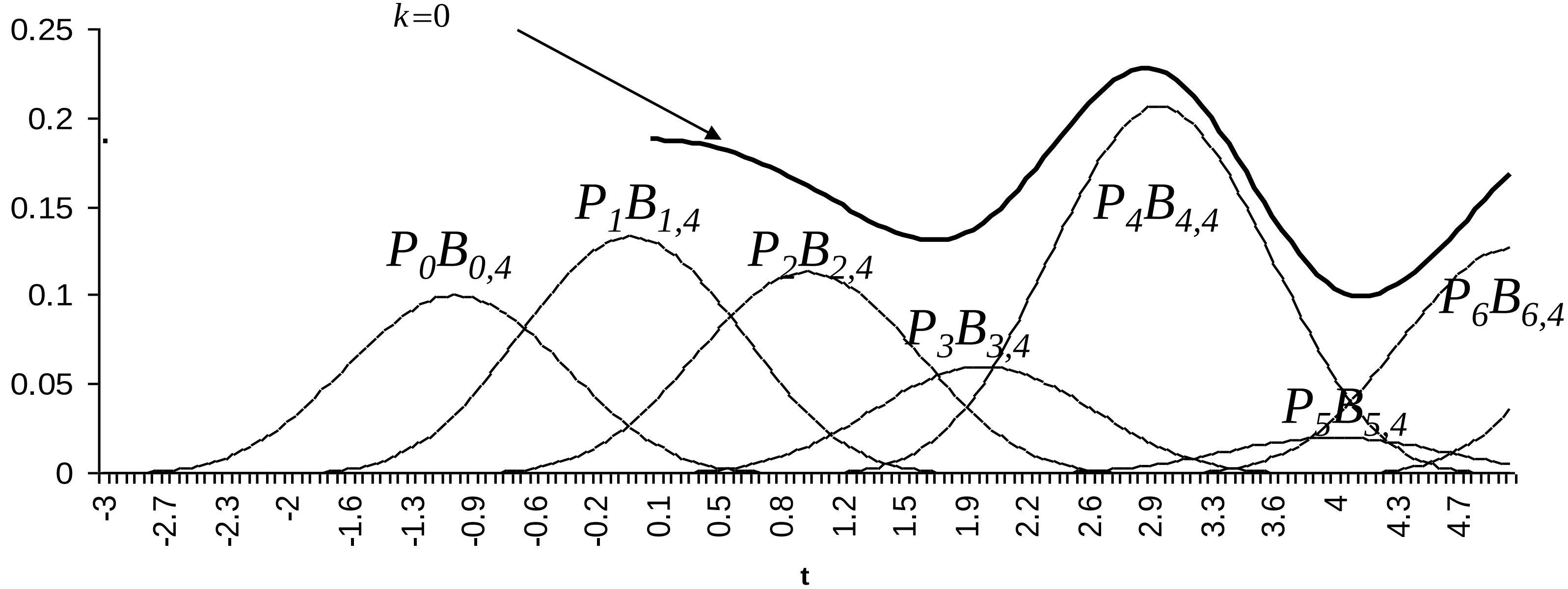
- Curve: $$X(t) = \sum_{k=0}^{n} P_k B_{k,d}(t)$$
  - n is the total number of control points
  - d is the order of the curves, $2 \le d \le n+1$, d typically 3 or 4
  - $B_{k,d}$ are the uniform B-spline blending functions of degree d-1
  - P_k are the control points
  - Each $B_{k,d}$ is only non-zero for a small range of t values, so the curve has local control

Uniform Cubic B-spline Blending Functions



Computing the Curve

$$X(t) = \sum_{k=0}^{n} P_k B_{k,4}(t)$$



The curve can't start until there are 4 basis functions active

Using Uniform B-splines

- At any point t along a piecewise uniform cubic B-spline, there are four non-zero blending functions
- Each of these blending functions is a translation of $B_{0,4}$
- Consider the interval $0 \leq t < 1$
  - We pick up the 4th section of $B_{0,4}$
  - We pick up the 3rd section of $B_{1,4}$
  - We pick up the 2nd section of $B_{2,4}$
  - We pick up the 1st section of $B_{3,4}$

Demo

Bspline-open.exe

Uniform B-spline at Arbitrary t

- The interval from an integer parameter value *i* to *i+1* is essentially the same as the interval from 0 to 1
  - The parameter value is offset by *i*
  - A different set of control points is needed
- To evaluate a uniform cubic B-spline at an arbitrary parameter value t:
  - Find the greatest integer less than or equal to *t*: *i* = floor(*t*)
  - Evaluate:

$$X(t) = \sum_{k=0}^{3} P_{i+k} B_{k,4}(t-i)$$

- Valid parameter range: $0 \le t < n-3$, where n is the number of control points

Loops

- To create a loop, use control points from the start of the curve when computing values at the end of the curve:

$$X(t) = \sum_{k=0}^{3} P_{(i+k) \bmod n} B_{k,4}(t-i)$$

- Any parameter value is now valid
  - Although for numerical reasons it is sensible to keep it within a small multiple of n

Demo

Bspline-closed.exe

B-splines and Interpolation, Continuity

- Uniform B-splines do not interpolate control points, unless:
  - You repeat a control point three times
  - But then all derivatives also vanish (=0) at that point
  - To do interpolation with non-zero derivatives you must use non-uniform B-splines with repeated knots
- To align tangents, use double control vertices
  - Then tangent aligns similar to Bezier curve
- Uniform B-splines are automatically C^2
  - All the blending functions are C^2, so sum of blending functions is C^2
  - Provides an alternate way to define blending functions
  - To reduce continuity, must use non-uniform B-splines with repeated knots

Rendering B-splines

- Same basic options as for Bezier curves
  - Evaluate at a set of parameter values and join with lines
    - Hard to know where to evaluate, and how pts to use
  - Use a subdivision rule to break the curve into small pieces, and then join control points
    - What is the subdivision rule for B-splines?
- Instead of subdivision, view splitting as *refinement*:
  - Inserting additional control points, and knots, between the existing points
  - Useful not just for rendering - also a user interface tool
  - Defined for uniform and non-uniform B-splines by the Oslo algorithm

Refining Uniform Cubic B-splines

- Basic idea: Generate 2*n*-3 new control points:
  - Add a new control point in the middle of each curve segment: $P'_{0,1}$, $P'_{1,2}$, $P'_{2,3}$, …, $P'_{n-2,n-1}$
  - Modify existing control points: P'_1, P'_2, …, P'_{n-2}
    - Throw away the first and last control
- Rules: $$P'_{i,j} = \frac{1}{2}\left(P_i + P_j\right), \qquad P'_i = \frac{1}{8}\left(P_{i-1} + 6P_i + P_{i+1}\right)$$
- If the curve is a loop, generate 2*n* new control points by averaging across the loop
- When drawing, don't draw the control polygon, join the ***x****(i)* points

From B-spline to Bezier

- Both B-spline and Bezier curves represent cubic curves, so either can be used to go from one to the other
- Recall, a point on the curve can be represented by a matrix equation:

$$x(t) = P^T M T$$

  - P is the column vector of control points
  - M depends on the representation: $M_{B\text{-}spline}$ and M_{Bezier}
  - T is the column vector containing: t^3, t^2, t, 1
- By equating points generated by each representation, we can find a matrix $M_{B\text{-}spline\text{->}Bezier}$ that converts B-spline control points into Bezier control points

B-spline to Bezier Matrix

$$M_{B-spline \rightarrow Bezier} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} P_{0,Bezier} \\ P_{1,Bezier} \\ P_{2,Bezier} \\ P_{3,Bezier} \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}\begin{bmatrix} P_{0,B-spline} \\ P_{1,B-spline} \\ P_{2,B-spline} \\ P_{3,B-spline} \end{bmatrix}$$

Rational Curves

- Each point is the ratio of two curves
  - Just like homogeneous coordinates:

$$[x(t), y(t), z(t), w(t)] \rightarrow \left[\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)}, \frac{z(t)}{w(t)}\right]$$

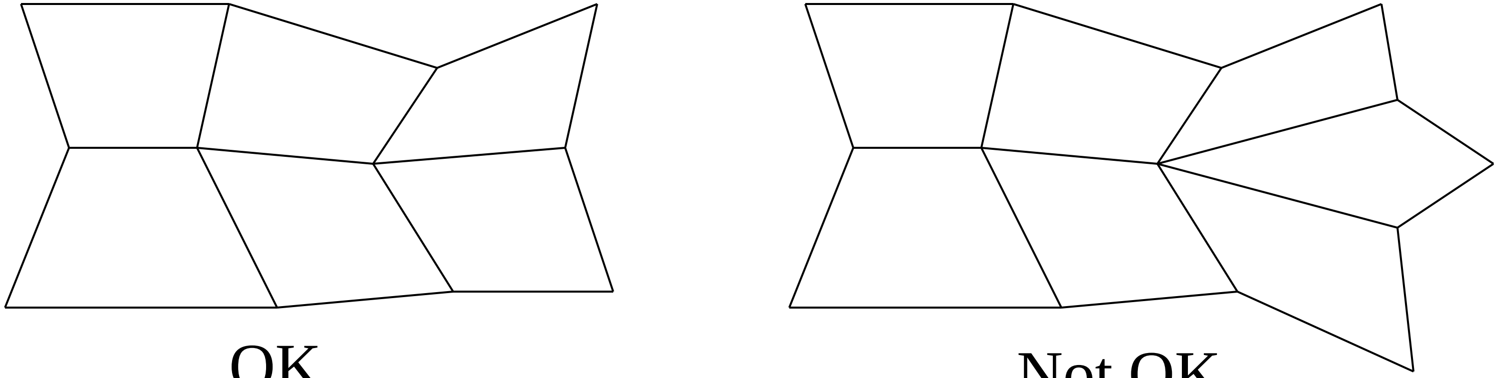
  - NURBS: *x(t)*, *y(t)*, *z(t)* and *w(t)* are non-uniform B-splines
- Advantages:
  - Perspective invariant, so can be evaluated in screen space
  - Can perfectly represent conic sections: circles, ellipses, etc
    - Piecewise cubic curves cannot do this

B-Spline Surfaces

- Defined just like Bezier surfaces:

$$X(s,t) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k} B_{j,d}(s) B_{k,d}(t)$$

- Continuity is automatically obtained everywhere
- BUT, the control points must be in a rectangular grid



OK

Not OK

Non-Uniform B-Splines

- Uniform B-splines are a special case of B-splines
- Each blending function is the same
- A blending functions starts at $t=-3$, $t=-2$, $t=-1$, …
- Each blending function is non-zero for 4 units of the parameter
- Non-uniform B-splines can have blending functions starting and stopping anywhere, and the blending functions are not all the same

B-Spline Knot Vectors

- *Knots*: Define a sequence of parameter values at which the blending functions will be switched on and off
- Knot values are increasing, and there are $n+d+1$ of them, forming a *knot vector*: $(t_0, t_1, \ldots, t_{n+d})$ with $t_0 \leq t_1 \leq \ldots \leq t_{n+d}$
- Curve only defined for parameter values between t_{d-1} and t_{n+1}
- These parameter values correspond to the places where the pieces of the curve meet
- There is one control point for each value in the knot vector
- The blending functions are recursively defined in terms of the knots and the curve degree

B-Spline Blending Functions

$$B_{k,1}(t) = \begin{cases} 1 & t_k \le t \le t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

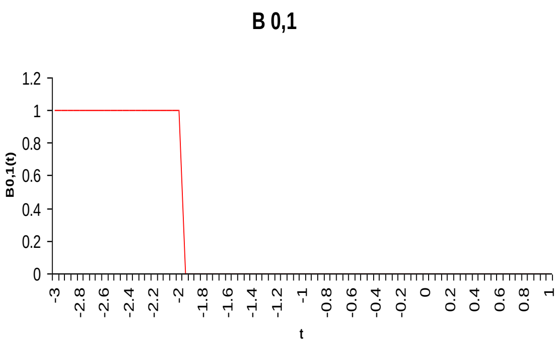
$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k} \right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}} \right) B_{k+1,d-1}(t)$$

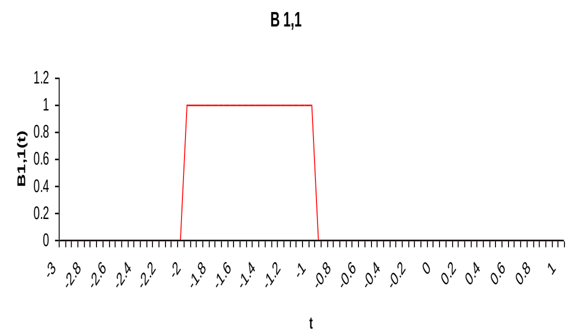
- The recurrence relation starts with the 1st order B-splines, just boxes, and builds up successively higher orders
- This algorithm is the Cox - de Boor algorithm
  - Carl de Boor was a professor here

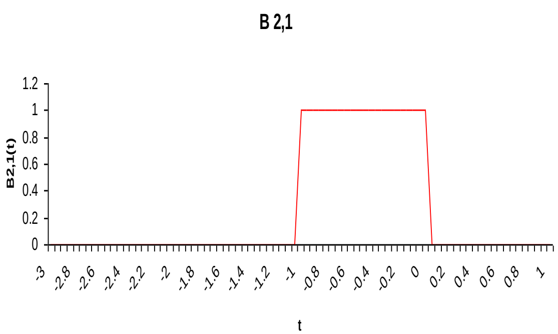
Uniform Cubic B-splines

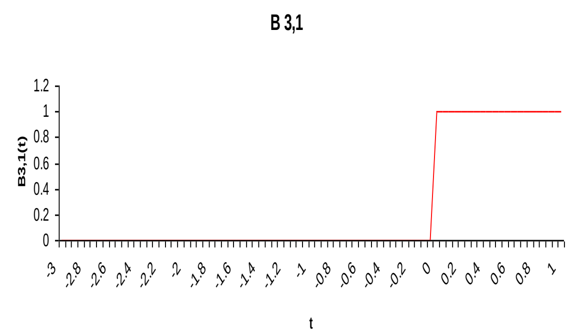
- Uniform cubic B-splines arise when the knot vector is of the form (-3,-2,-1,0,1,…,n+1)
- Each blending function is non-zero over a parameter interval of length 4
- All of the blending functions are translations of each other
  - Each is shifted one unit across from the previous one
  - $B_{k,d}(t)=B_{k+1,d}(t+1)$
- The blending functions are the result of convolving a box with itself d times, although we will not use this fact

$B_{k,1}$

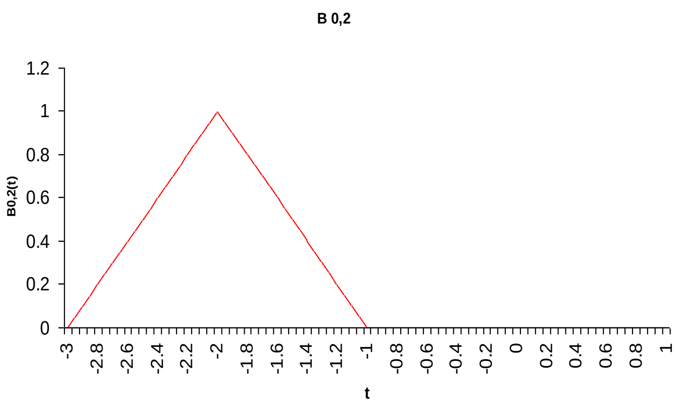

B 0,1
B0,1(t)
1.2
1
0.8
0.6
0.4
0.2
0
-3 -2.8 -2.6 -2.4 -2.2 -2 -1.8 -1.6 -1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
t

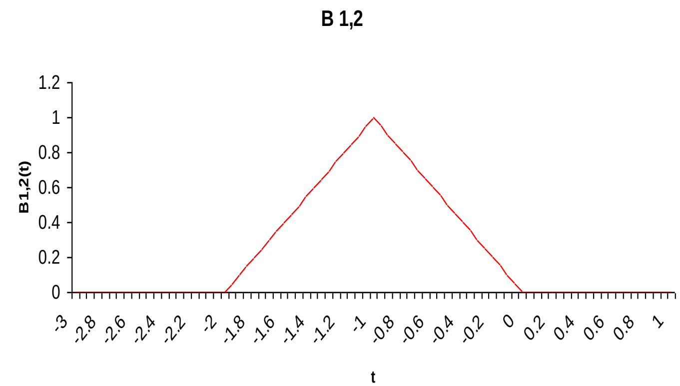

B 1,1
B1,1(t)
1.2
1
0.8
0.6
0.4
0.2
0
-3 -2.8 -2.6 -2.4 -2.2 -2 -1.8 -1.6 -1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
t

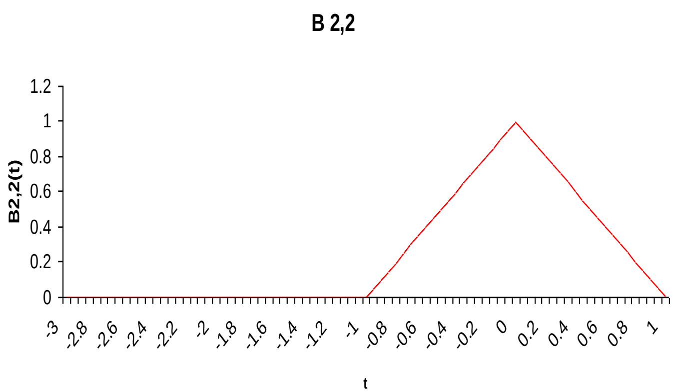

B 2,1
B2,1(t)
1.2
1
0.8
0.6
0.4
0.2
0
-3 -2.8 -2.6 -2.4 -2.2 -2 -1.8 -1.6 -1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
t


B 3,1
B3,1(t)
1.2
1
0.8
0.6
0.4
0.2
0
-3 -2.8 -2.6 -2.4 -2.2 -2 -1.8 -1.6 -1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
t

$B_{k,2}$

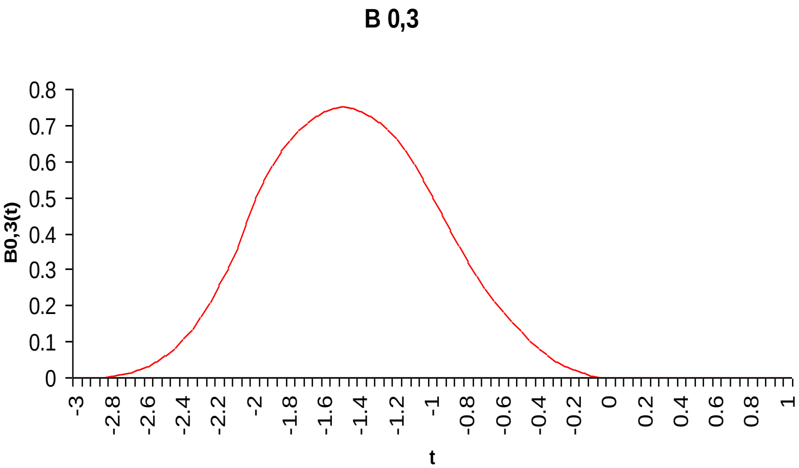


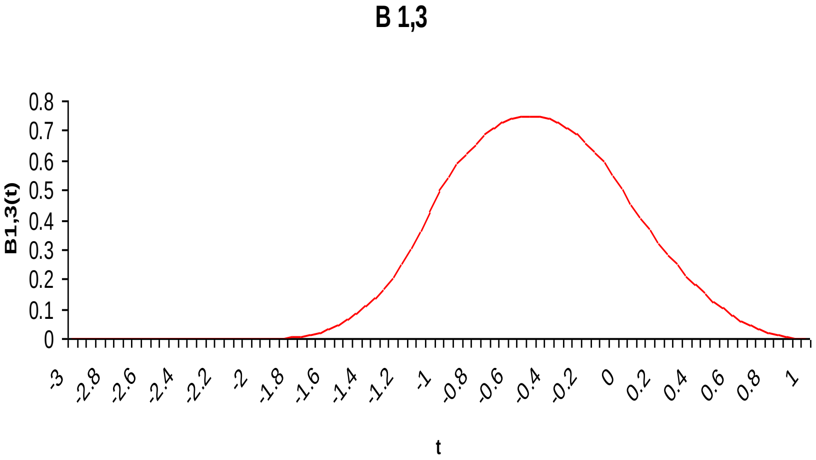




$$B_{0,2}(t) = \begin{cases} t+3 & -3 \le t < -2 \\ -1-t & -2 \le t < -1 \end{cases}$$

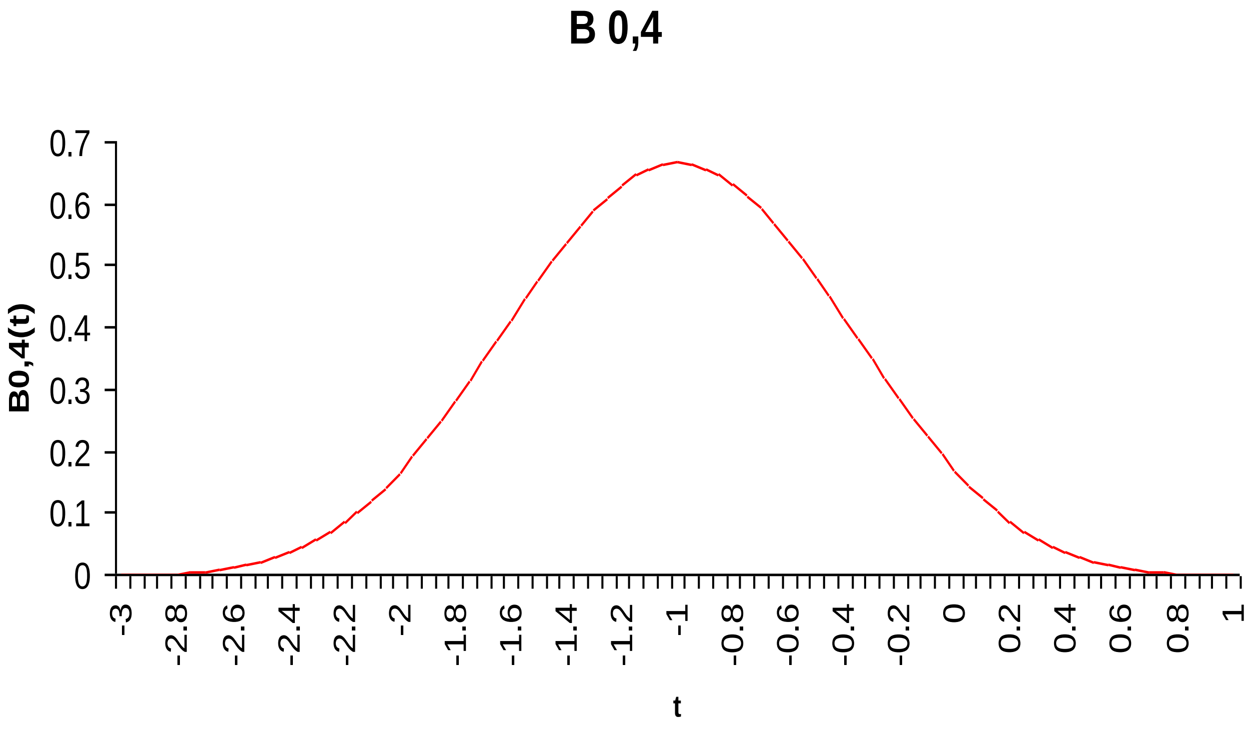
$B_{k,3}$





$$B_{0,3}(t) = \frac{1}{2}\begin{cases} (t+3)^2 & -3 \le t < -2 \\ -2t^2 - 6t - 3 & -2 \le t < -1 \\ t^2 & -1 \le t < 0 \end{cases}$$

$B_{0,4}$


B 0,4
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
B0,4(t)
-3
-2.8
-2.6
-2.4
-2.2
-2
-1.8
-1.6
-1.4
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
t

$B_{0,4}$

$$B_{0,4}(t) = \frac{1}{6}\begin{cases} (t+3)^3 & -3 \le t < -2 \\ -3t^3 - 15t^2 - 21t - 5 & -2 \le t < -1 \\ 3t^3 + 3t^2 - 3t + 1 & -1 \le t < 0 \\ (1-t)^3 & 0 \le t < 1 \end{cases}$$

Note that the functions given on slides 4 and 5 are translates of this function obtained by using *(t-1)*, *(t-2)* and *(t-3)* instead of just *t*, and then selecting only a sub-range of *t* values for each function

Interpolation and Continuity

- The knot vector gives a user control over interpolation and continuity
- If the first knot is repeated three times, the curve will interpolate the control point for that knot
  - Repeated knot example: (-3,-3,-3, -2, -1, 0, …)
  - If a knot is repeated, so is the corresponding control point
- If an interior knot is repeated, continuity at that point goes down by 1
- Interior points can be interpolated by repeating interior knots
- A deep investigation of B-splines is beyond the scope of this class

How to Choose a Spline

- Hermite curves are good for single segments where you know the parametric derivative or want easy control of it
- Bezier curves are good for single segments or patches where a user controls the points
- B-splines are good for large continuous curves and surfaces
- NURBS are the most general, and are good when that generality is useful, or when conic sections must be accurately represented (CAD)