EECS 487: Interactive Computer Graphics

Lecture 37:

- B-splines curves
- Rational Bézier and NURBS

Cubic Splines

A representation of cubic spline consists of:

- four control points (why four?)
  - these are completely user specified
  - determine a set of blending functions

There is no single "best" representation of cubic spline:

Cubic	Interpolate?	Local?	Continuity	Affine?	Convex*?	VD*?
Hermite	✔	✔	C^1	✔	n/a	n/a
Cardinal (Catmull-Rom)	except endpoints	✔	C^1	✔	no	no
Bézier	endpoints	✗	C^1	✔	✔	✔
natural	✔	✗	C^2	✔	n/a	n/a
B-splines	✗	✔	C^2	✔	✔	✔

* n/a when some of the control "points" are tangents, not points

Natural Cubic Spline

A natural cubic spline's control points:

$$\mathbf{f}(u) = \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 + u^3\mathbf{a}_3$$

- position of start point $\quad \mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 + 0^1\mathbf{a}_1 + 0^2\mathbf{a}_2 + 0^3\mathbf{a}_3$
- 1st derivative of start point $\quad \mathbf{p}_1 = \mathbf{f}'(0) = \mathbf{a}_1 + 2*0^1\mathbf{a}_2 + 3*0^2\mathbf{a}_3$
- 2nd derivative of start point $\quad \mathbf{p}_2 = \mathbf{f}''(0) = 2*1^1\mathbf{a}_2 + 6*0^2\mathbf{a}_3$
- position of end point $\quad \mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a}_0 + 1^1\mathbf{a}_1 + 1^2\mathbf{a}_2 + 1^3\mathbf{a}_3$

- constraint and basis matrices:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \qquad \mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ -1 & -1 & -\frac{1}{2} & 1 \end{bmatrix}$$

- subsequent segments assume the position and 1st and 2nd derivatives of the end point of the preceding segment

Natural Cubic Spline

Given n control points, a natural cubic spline has $n-1$ segments

For segment i:

$$\mathbf{f}_i(0) = \mathbf{p}_{i-1}, \qquad i = 1,\ldots,n$$
$$\mathbf{f}_i(1) = \mathbf{p}_i, \qquad i = 1,\ldots,n$$

$$\mathbf{f}_i'(0) = \mathbf{f}_{i-1}'(1), \qquad i = 1,\ldots,n-1$$
$$\mathbf{f}_i''(0) = \mathbf{f}_{i-1}''(1), \qquad i = 1,\ldots,n-1$$

Set: $\mathbf{f}_1''(0) = \mathbf{f}_n''(1) = 0$

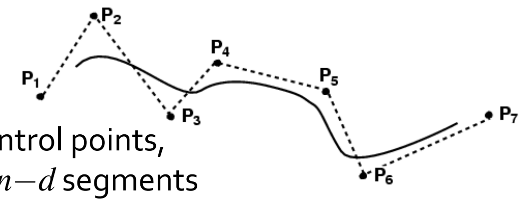
O'Brien

Natural Cubic Splines

Each curve segment (other than the first) receives three out of its four control points from the preceding segment, this gives the curve C^2 continuity

However the polynomial coefficients are dependent on all n control points
- control is not local: any change in any segment may change the whole curve
- curve tends to be ill-conditioned: a small change at the beginning can lead to large subsequent changes

B-splines



Given n $(\geq d + 1)$ control points, a B-spline curve has $n-d$ segments
- d is the degree of each B-spline segment
- the segments are numbered d to $n-1$, for ease of notation
- number of control points is independent of the degree
  - unlike a Bézier spline, where adding a control point necessarily increases degree by one,
  - and unlike multi-segmented Bézier curve where multiple control points supporting a new segment must be added at the same time
- segment degree (d) is also curve degree

B-splines of degree d are said to have order k $(= d + 1)$

TP3, Watt

Advantages of B-splines

Main advantages of B-splines:
- number of control points not limited by degree (d)
- automatic C^{d-1} continuity
- local control

To create a large model with C^2 continuity and local control, you pretty much want to use cubic B-splines

Aside from the first segment, each B-spline segment shares the first d control points with its preceding segment
- sounds like natural spline … how can B-splines have local control?

Local Control

Unlike natural splines and Bézier curves, B-splines' control points are not derivatives

Instead each segment is a weighted-sum of d basis functions (only), giving the control points local control

Hence Basis spline

Why is $\mathbf{B}$ called the Basis Matrix?

Polynomials as a Vector Space

Polynomials $f(u) = a_0 + a_1 u + a_2 u^2 + \ldots + a_n u^n$

- can be added: just add the coefficients
- can be multiplied by a scalar: multiply the coefficients
- are closed under addition and multiplication by scalar
  - i.e., the result is still a polynomial

$\Rightarrow$ It's a vector space!

A vector space is defined by a set of basis

- linearly independent vectors
- linear combination of the basis vectors spans the space
- here vector = polynomial

Canonical Power Basis

$1, u, u^2, u^3, \ldots, u^n$

- are independent
- any polynomial is a linear combination of these, $a_0 + a_1 u + a_2 u^2 + \ldots + a_n u^n$
- often called the canonical basis functions

Just as with Euclidean space,
there are infinite number of possible basis

For cubic, the basis functions could be, for example:

- $1, 1+u, 1+u+u^2, 1+u-u^2+u^3$
- $u^3, u^3-u^2, u^3+u, u^3+1$

Basis Matrix and Basis Functions

A basis matrix ($\mathbf{B}$) transforms the canonical basis ($\mathbf{u}$) to another basis:

$$\mathbf{f}(u) = \mathbf{ua} = \mathbf{uBp} = (1-u)\mathbf{p}_0 + u\mathbf{p}_1 = \sum_{i=0}^{n} b_i(u)\mathbf{p}_i$$

$$\mathbf{uB} = \sum_{i=0}^{n} b_i(u)$$

The $b_i(u)$'s are the basis functions of the other basis (we've known them as the blending functions)

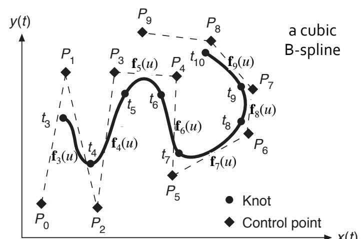
B-splines

Given n control points, there are $n-d$ segments

- we call the segments $\mathbf{f}_i(u)$, $d \le i < n$
  - each segment has a unit range, $0 \le u \le 1$
- we call the entire B-spline curve with n control points $\mathbf{f}(t)$

The parameters t_i's where two segments join are called knots

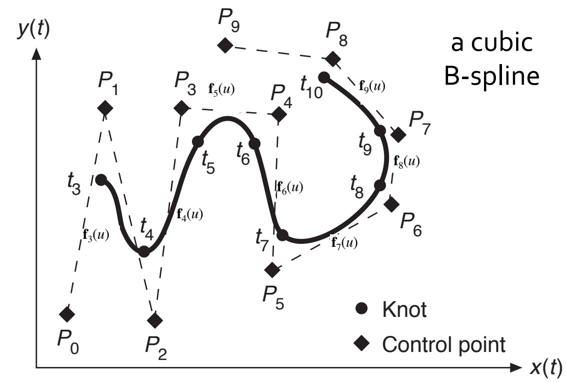
- the start and end points (t_d and t_n) are also called knots
- the range $[t_d, t_n]$ is the domain of a B-spline curve
- the parameter u of segment i is scaled to $t_i \le u < t_{i+1}$



Foley, van Dam90

Uniform B-splines

The knots of a uniform B-splines are spaced at equal intervals



Foley, van Dam90

What Degree is Sufficient?

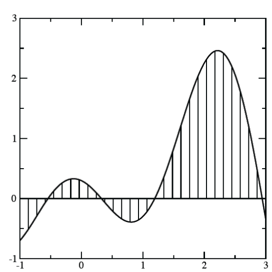
Arbitrary curves have an uncountable number of parameters

Real-number function value expanded into an infinite set of basis functions:

$$\mathbf{f}(u) = \sum_{i=0}^{\infty} b_i(u)\mathbf{p}_i$$

Approximate by truncating set at some reasonable point, e.g., 3:

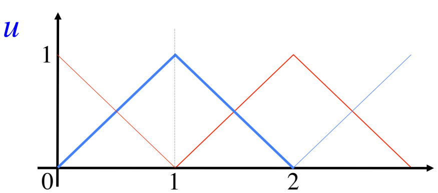
$$\mathbf{f}(u) = \sum_{i=0}^{3} b_i(u)\mathbf{p}_i$$



O'Brien

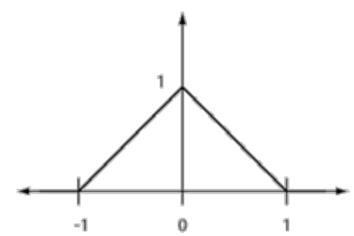
Linear B-spline Segment

In the linear case, the basis functions are $b_0(u) = (1-u)$ and $b_1(u) = u$



(a.k.a. tent/triangle basis, the i'th functions are shifted versions of the 0'th)

$$b(u) = \begin{cases} 0 & u < -1 \\ 1+u & -1 < u < 0 \\ 1-u & 0 < u < 1 \\ 0 & u > 1 \end{cases}$$

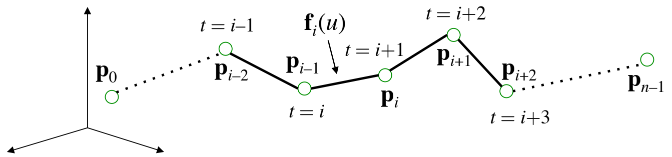


James,Hanrahan

Linear B-spline Curve

Consider using linear B-splines (d=1, k=2) to draw a piecewise linear curve (a polyline)

To draw the curve, we perform linear interpolation of a set of control points $\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}$



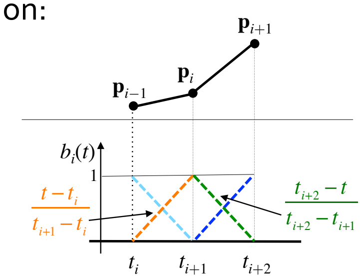
For segment i, we write the interpolating linear curve as $\mathbf{f}_i(u) = (1-u)\mathbf{p}_{i-1} + u\mathbf{p}_i$, where $u = \dfrac{t-t_i}{t_{i+1}-t_i} \in [0,1]$

[Craig]

Linear B-splines

The influence of control point $\mathbf{p}_i$ on the whole curve is thus the "tent/triangle" function:

$$b_i(t) = \begin{cases} \dfrac{t-t_i}{t_{i+1}-t_i}, & t_i \le t < t_{i+1}, \\ \dfrac{t_{i+2}-t}{t_{i+2}-t_{i+1}}, & t_{i+1} \le t < t_{i+2}, \\ 0, & \text{everywhere else} \end{cases}$$



The hardest part of working with B-splines is keeping track of the tedious notations!

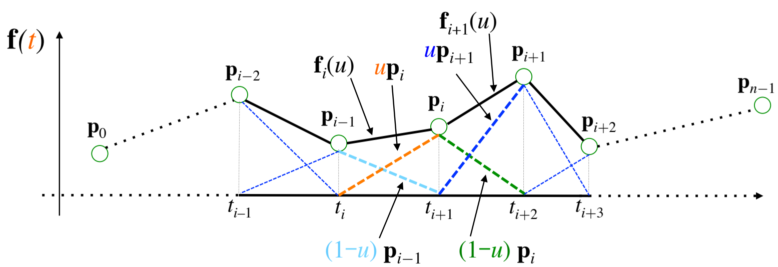
[Craig]

Linear B-splines

Linearly interpolating the set of control points to draw the curve:

$$\mathbf{f}_i(u) = (1-u)\mathbf{p}_{i-1} + u\mathbf{p}_i = \frac{t_{i+1}-t}{t_{i+1}-t_i}\mathbf{p}_{i-1} + \frac{t-t_i}{t_{i+1}-t_i}\mathbf{p}_i,\; u = \frac{t-t_i}{t_{i+1}-t_i},\; t_i \le t < t_{i+1}$$

$$\mathbf{f}_{i+1}(u) = (1-u)\mathbf{p}_i + u\mathbf{p}_{i+1} = \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}\mathbf{p}_i + \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}}\mathbf{p}_{i+1},\; u = \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}},\; t_{i+1} \le t < t_{i+2}$$



[Craig]

Linear B-splines

$$\mathbf{f}_i(u) = (1-u)\mathbf{p}_{i-1} + u\mathbf{p}_i = \frac{t_{i+1}-t}{t_{i+1}-t_i}\mathbf{p}_{i-1} + \frac{t-t_i}{t_{i+1}-t_i}\mathbf{p}_i,\; u = \frac{t-t_i}{t_{i+1}-t_i},\; t_i \le t < t_{i+1}$$

$$\mathbf{f}_{i+1}(u) = (1-u)\mathbf{p}_i + u\mathbf{p}_{i+1} = \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}\mathbf{p}_i + \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}}\mathbf{p}_{i+1},\; u = \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}},\; t_{i+1} \le t < t_{i+2}$$

We can rewrite the segment functions as:

$$\mathbf{f}_i(t) = b_{i-1}(t)\mathbf{p}_{i-1} + b_i(t)\mathbf{p}_i,\; t_i \le t < t_{i+1}$$

$$\mathbf{f}_{i+1}(t) = b_i(t)\mathbf{p}_i + b_{i+1}(t)\mathbf{p}_{i+1},\; t_{i+1} \le t < t_{i+2}$$

And for the whole curve:

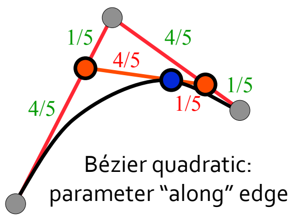
$$\mathbf{f}(t) = \sum_{i=1}^{n-1}\mathbf{f}_i(t) = \sum_{i=1}^{n-1} b_i(t)\mathbf{p}_i$$

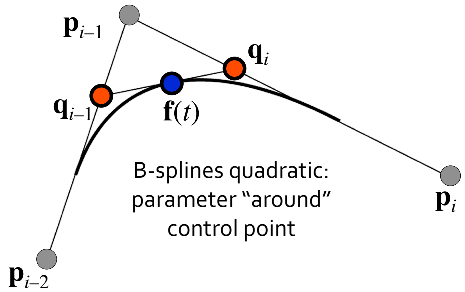
where $b_i(t)$'s are the basis functions (in this case, linear)

[Craig]

Quadratic B-splines

Quadratic B-splines (d=2, k=3) are drawn by two interpolation steps, similar but different to quadratic Bézier





Whereas de Casteljau algorithm performs the iterative interpolations for Bézier curves, de Boor algorithm does so for B-splines

[TP3,Buss]

de Boor Algorithm

De Boor algorithm is an iterative interpolation algorithm that generalizes de Casteljau's algorithm

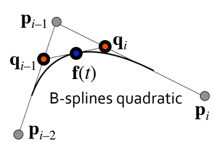
To evaluate a B-spline curve $\mathbf{f}(t)$ at parameter value t:

1. determine the $[t_i, t_{i+1})$ in which t belongs; $d \le i < n$, the domain of the curve is $[t_d, t_n]$
2. to compute $\mathbf{f}(t)$ of degree d, first interpolate between control points $\mathbf{p}$'s
3. then, in a bottom up fashion, continue to perform r rounds of pairwise linear interpolations, until $r = d$, using:

$$\mathbf{f}^r_{j,d}(t) = \frac{t_{j+k-r} - t}{t_{j+k-r} - t_j}\mathbf{f}^{r-1}_{j-1}(t) + \frac{t - t_j}{t_{j+k-r} - t_j}\mathbf{f}^{r-1}_{j}(t),\ t_j \le t < t_{j+k-r},$$
$$1 \le r \le d,$$
$$j = i-d+r,\ i-d+r+1,\ \ldots, i$$

[TP3,Buss]

Quadratic B-splines



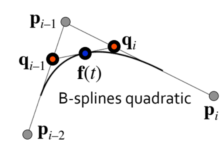
Using the de Boor algorithm we first compute $\mathbf{q}_{i-1}$ and $\mathbf{q}_i$ (note: over two knot intervals):

$$\mathbf{q}_{i-1} = \mathbf{f}^1_{i-1,2}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_{i-1}}\mathbf{p}_{i-2} + \frac{t - t_{i-1}}{t_{i+1} - t_{i-1}}\mathbf{p}_{i-1},\ t_{i-1} \le t < t_{i+1}$$

$$\mathbf{q}_{i} = \mathbf{f}^1_{i,2}(t) = \frac{t_{i+2} - t}{t_{i+2} - t_{i}}\mathbf{p}_{i-1} + \frac{t - t_{i}}{t_{i+2} - t_{i}}\mathbf{p}_{i},\ t_{i} \le t < t_{i+2}$$

[TP3,Buss]

Quadratic B-splines



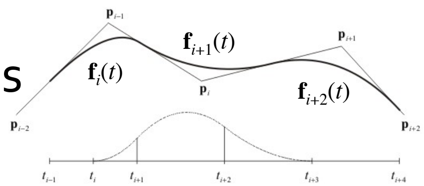
Then we linearly interpolate between $\mathbf{q}_{i-1}$ and $\mathbf{q}_i$ in a second round (r = 2) of interpolation:

$$\mathbf{f}^2_{i,2}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i}\mathbf{q}_{i-1} + \frac{t - t_i}{t_{i+1} - t_i}\mathbf{q}_i,\ t_i \le t < t_{i+1}$$

$$\begin{aligned}\mathbf{f}(t) = &\frac{t_{i+1} - t}{t_{i+1} - t_i}\frac{t_{i+1} - t}{t_{i+1} - t_{i-1}}\mathbf{p}_{i-2} \\ &+\left(\frac{t_{i+1} - t}{t_{i+1} - t_i}\frac{t - t_{i-1}}{t_{i+1} - t_{i-1}} + \frac{t - t_i}{t_{i+1} - t_i}\frac{t_{i+2} - t}{t_{i+2} - t_i}\right)\mathbf{p}_{i-1} \\ &+\frac{t - t_i}{t_{i+1} - t_i}\frac{t - t_i}{t_{i+2} - t_i}\mathbf{p}_i\end{aligned}$$

[TP3,Buss]

Quadratic B-splines



The control point $\mathbf{p}_i$ influences $\mathbf{f}_{i,2}(t)$, $\mathbf{f}_{i+1,2}(t)$, and $\mathbf{f}_{i+2,2}(t)$, from which we can assemble its blending function:

$$b_i(t) = \begin{cases} \dfrac{t_{i+1}-t}{t_{i+1}-t_i}\dfrac{t_{i+1}-t}{t_{i+1}-t_{i-1}}, & t_i \le t < t_{i+1}, \\ \dfrac{t_{i+1}-t}{t_{i+1}-t_i}\dfrac{t-t_{i-1}}{t_{i+1}-t_{i-1}} + \dfrac{t-t_i}{t_{i+1}-t_i}\dfrac{t_{i+2}-t}{t_{i+2}-t_i}, & t_{i+1} \le t < t_{i+2}, \\ \dfrac{t-t_i}{t_{i+1}-t_i}\dfrac{t-t_i}{t_{i+2}-t_i}, & t_{i+2} \le t < t_{i+3}, \\ 0, & \text{everywhere else} \end{cases}$$

[TP3,Buss]

Interpolation and Basis Functions

	Bézier	B-spline
interpolation	de Casteljau	de Boor
basis functions	Bernstein polynomials	Cox-de Boor recurrence

Cox-de Boor Recurrence

de Boor algorithm constructs basis functions "bottom-up", whereas Cox-de Boor recurrence generates the basis functions "top-down"

Let $b_{i,k}(t)$ be a k-th order basis function for weighting control point $\mathbf{p}_i$,

$$b_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \text{ (both} \le \text{for last segment),} \\ 0, & \text{otherwise} \end{cases}$$

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

- if the denominator is 0 (non-uniform knots), the basis function is defined to be 0
- Cox-de Boor recurrence essentially takes a linear interpolation of linear interpolations of linear interpolations, similar to the de Casteljau algorithm

Cubic B-splines

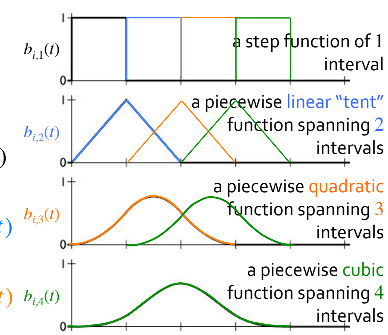
For 4th order (cubic) B-splines, the recursive definition starts at $b_{i,4}(t)$:

base: $b_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$

linear: $b_{i,2}(t) = \dfrac{t-t_i}{t_{i+1}-t_i} b_{i,1}(t) + \dfrac{t_{i+2}-t}{t_{i+2}-t_{i+1}} b_{i+1,1}(t)$

quadratic: $b_{i,3}(t) = \dfrac{t-t_i}{t_{i+2}-t_i} b_{i,2}(t) + \dfrac{t_{i+3}-t}{t_{i+3}-t_{i+1}} b_{i+1,2}(t)$

cubic: $b_{i,4}(t) = \dfrac{t-t_i}{t_{i+3}-t_i} b_{i,3}(t) + \dfrac{t_{i+4}-t}{t_{i+4}-t_{i+1}} b_{i+1,3}(t)$



Illustrated with uniformly spaced knots

Uniform Cubic Basis Function

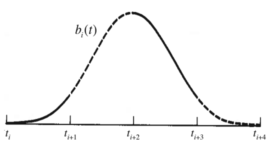
Constructed from the Cox-de Boor recurrence
- taking advantage of fixed interval between knots
- considering only intervals for which the basis function is non-zero

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

$$b_{i,2}(t) = (t - t_i)\, b_{i,1}(t) + (t_{i+2} - t)\, b_{i+1,1}(t)$$

$$b_{i,3}(t) = \frac{t - t_i}{2} b_{i,2}(t) + \frac{t_{i+3} - t}{2} b_{i+1,2}(t)$$

$$b_{i,4}(t) = \frac{t - t_i}{3} b_{i,3}(t) + \frac{t_{i+4} - t}{3} b_{i+1,3}(t)$$



Let $t_i = i$, specializing for $i = 0$:

$$b_{0,4}(t) = \begin{cases} \dfrac{t^3}{6}, & 0 \le t < 1, \\ \dfrac{-3t^3 + 12t^2 - 12t + 4}{6}, & 1 \le t < 2, \\ \dfrac{3t^3 - 24t^2 + 60t - 44}{6}, & 2 \le t < 3, \\ \dfrac{-t^3 + 12t^2 - 48t + 64}{6}, & 3 \le t < 4, \\ 0, & \text{everywhere else} \end{cases}$$

[Buss, Shirley, Gleicher]

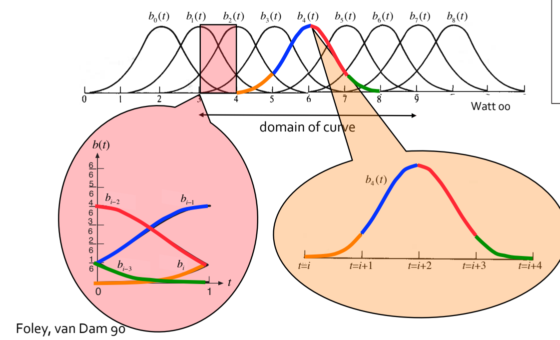
Local Control Property

For uniform, multi-segment B-spline curves, the knot values are equally spaced and each basis function is a copy and translate of each other

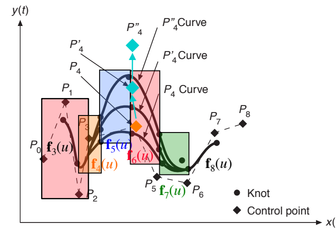
We define the entire set of curve segments as one B-spline curve in t:

$$\mathbf{f}(t) = \sum_{i=0}^{n-1} b_i(t)\mathbf{p}_i, \; t \in [3, n]$$

The curve is a linear combination of all the basis functions of the segments:



Foley, van Dam 90



- each segment is influenced by four (non-zero) basis functions
- each control point is scaled by a single basis function
- each basis function is non-zero over four successive intervals in t

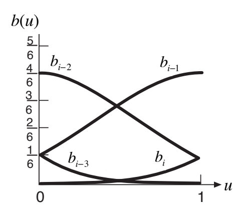
⇒ each control point influences four segments (only) ⇒ local control

Convex Hull Property

The basis function is ≥ 0 and sums to unity in the range t_i to t_{i+4}

⇒ all the control points form a convex hull

⇒ the whole curve is within the convex hull

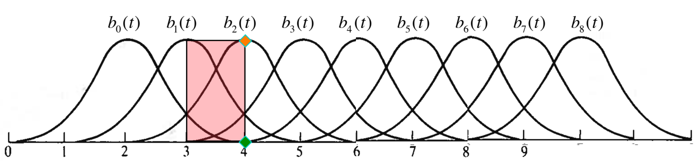


Between knot values, the four basis functions are non-zero and sum to unity

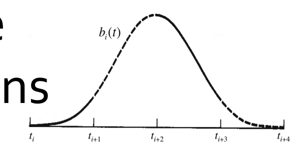
$$\sum_{k=0}^{3} b_{i-k}(t) = 1, \; t_i \le t < t_{i+1}$$

$$b_{i-k}(t) \ge 0, \; 3 \le i < n, \; 0 \le k \le 3$$

At each knot value, one basis function "switches off" and another "switches on", and three basis functions are non-zero and sum to unity



Uniform Cubic B-spline Segment Basis Functions



Basis functions for a single B-spline segment are
- shifted pieces of a single basis function to $u \in [0,1]$ range

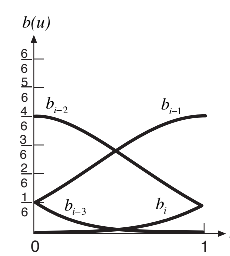
Specializing for $i = 0$:

$$b_{i,4}(u) = \frac{u^3}{6}, \qquad u = t, \; 0 \le t < 1$$

$$b_{i-1,4}(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}, \qquad u = t - 1, \; 1 \le t < 2$$

$$b_{i-2,4}(u) = \frac{3u^3 - 6u^2 + 4}{6}, \qquad u = t - 2, \; 2 \le t < 3$$

$$b_{i-3,4}(u) = \frac{(1-u)^3}{6}, \qquad u = t - 3, \; 3 \le t < 4$$



[Shirley, Gleicher]

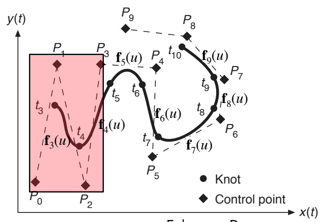
Uniform Cubic B-spline Segment

Control points for one segment $\mathbf{f}_i(u)$ are $\mathbf{p}_{i-3}$, $\mathbf{p}_{i-2}$, $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, $3 \le i < n$, recall: the control points can take on arbitrary values (geometric constraints)

A segment is described as:

$$\mathbf{f}_i(u) = \sum_{j=0}^{3} b_{i-3+j}(u)\mathbf{p}_{i-3+j},\ u \in [0,1]$$

$$= \frac{(1-u)^3}{6}\mathbf{p}_{i-3} + \frac{3u^3 - 6u^2 + 4}{6}\mathbf{p}_{i-2} + \frac{-3u^3 + 3u^2 + 3u + 1}{6}\mathbf{p}_{i-1} + \frac{u^3}{6}\mathbf{p}_i$$



Foley, van Dam 90

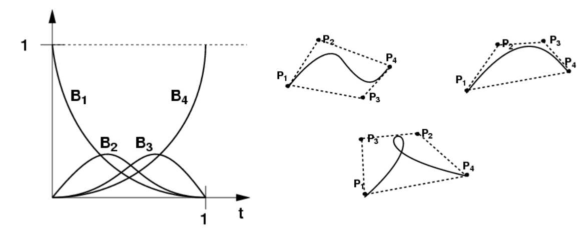
The cubic B-spline segment basis matrix is:

$$\mathbf{B} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

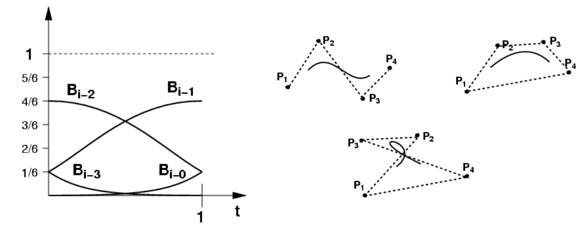
Bézier is Not B-spline

Relationship to the control points is different

Bézier



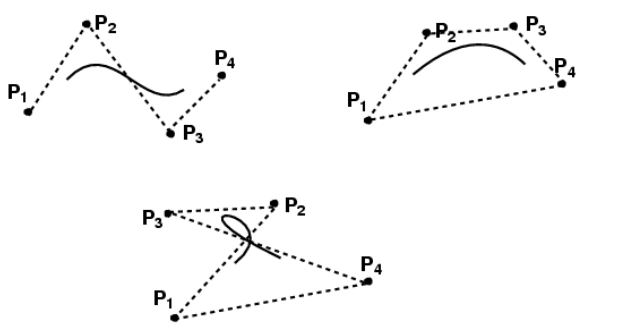
B-spline



Durand

Interpolation

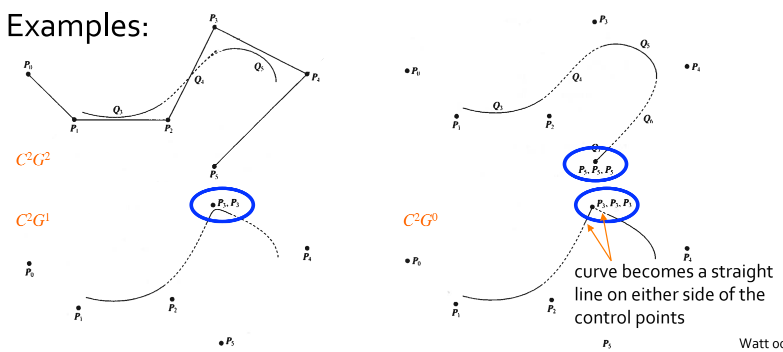
A B-spline curve doesn't have to interpolate any of its control points



Interpolation with Multiple Control Points

A B-spline curve can be made to interpolate one or more of its control points by adding multiple control points of the same value, at the loss of continuity

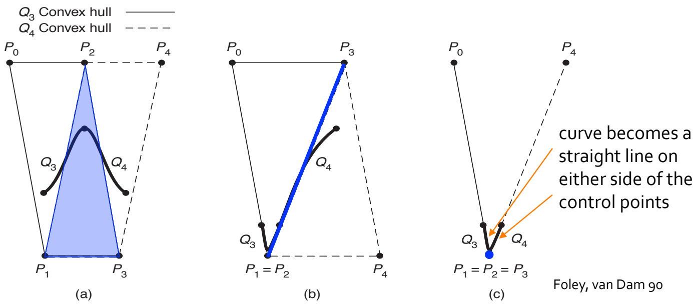
Examples:



Watt 00

Interpolation with Multiple Control Points

Multiple control points reduces continuity: the intersection between the two convex hulls shrinks from a region to a line to a point, and causes the adjacent segments to become linear

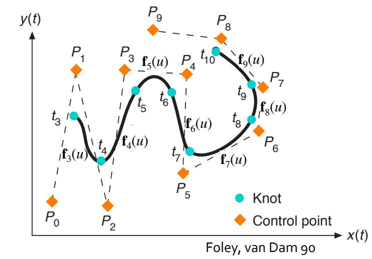


Foley, van Dam 90

Non-uniform B-splines

Non-uniform B-splines interpolate without causing adjacent segments to become linear by using multiple knots instead of multiple control points

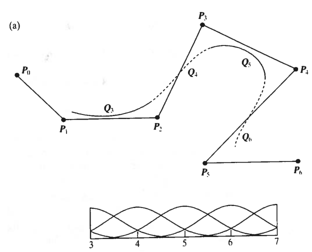
The interval between t_i and t_{i+1} may be non-uniform; when $t_i = t_{i+1}$, curve segment $\mathbf{f}_i$ is a single point



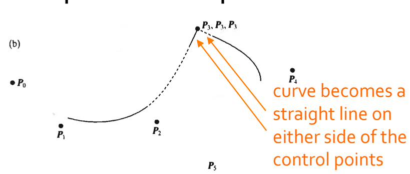
Foley, van Dam 90

Interpolation with Multiple Knots

Uniform B-splines:



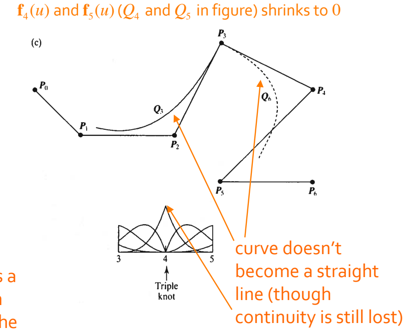
Uniform B-splines, multiple control points:



Non-uniform B-splines, multiple knots:

$t = [0,1,2,3,4,4,4,5,6,7,8]$

$\mathbf{f}_4(u)$ and $\mathbf{f}_5(u)$ (Q_4 and Q_5 in figure) shrinks to 0



Watt 00

Non-Uniform B-splines Basis Functions

Because the intervals between knots are not uniform, there is no single set of basis functions

Instead, the basis functions depend on the intervals between knot values and are defined recursively in terms of lower-order basis functions (using the Cox-de Boor recurrence)

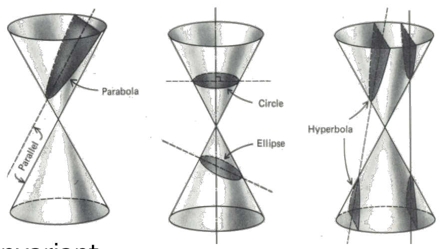
A Bézier curve is really a non-uniform B-splines with no (interior) knot between control points

- B-splines can be rendered as a Bézier curve, by inserting multiple knots at the control points, with no interior knot!

Rational Curves

Polynomial curves cannot represent conic sections/ quadrics exactly–for modeling machine parts, e.g.

Why not?
A conic section in $2D$ is the perspective projection of a parabola in $3D$ onto the plane $z = 1$, with the COP at the origin $\mathbf{o}$



Polynomial curves are affine invariant, but not perspective invariant

Instead, use a rational curve,
i.e., a ratio of polynomials: $\mathbf{f}(u) = \dfrac{p_1(u)}{p_2(u)}$

Merrell, Funkhouser, andrews.edu

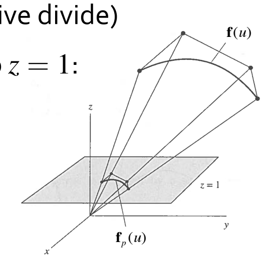
Rational Cubic Bézier

As with homogeneous coordinate, a rational curve is a nonrational curve that has been perspective projected

Cubic Bézier:

- add an extra weight coordinate: $w_i\mathbf{p}_i = (w_ix_i, w_iy_i, w_iz_i, w_i)$ (w_i is the homogeneous coordinate)
- rational due to division by final weight coordinate: ($=$ perspective divide)
- projected to $z = 1$:

$$\mathbf{f}_p(u) = \frac{\sum_{i=0}^{3} w_i b_i(u)\mathbf{p}_i}{\sum_{i=0}^{3} w_i b_i(u)}$$



If the w_i's are all equal, we recover the nonrational curve

Ramamoorthi, Wattoo

Advantages of Rational Curves

Both affine and perspective invariant

Can represent conics as rational quadratics

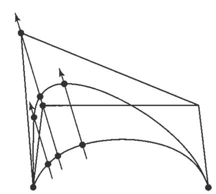
Weights (w_i's) provide extra control:
values affect "tension" near control points

- the w_i's cannot all be simultaneously zero
- if all the w_i's are ≥ 0, the curve is still contained in the convex hull of the control polygon

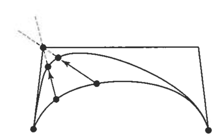
[Farin]

Role of the Weights (w_i's)

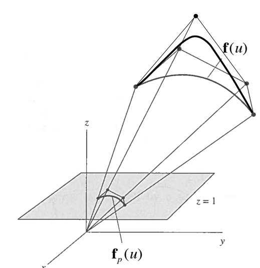
For example: larger w_1 means that the pre-image, nonrational curve near $\mathbf{p}_1$ is "further up" in z, and the projected image is "pulled" towards $\mathbf{p}_1$



moving control point



changing weight



[Farin,Watt]

Non-Uniform Rational B-Splines

$$\mathbf{f}(u) = \frac{\sum_{i=0}^{3} w_i b_{i,k}(u)\mathbf{p}_i}{\sum_{i=0}^{3} w_i b_{i,k}(u)}$$

with:

$w_i =$ scalar weight for each control point

$\mathbf{p}_i =$ control points

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

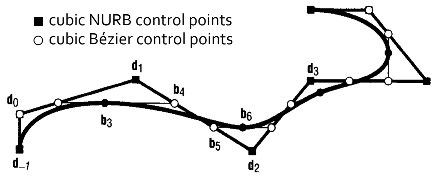
$b_{i,k}(u) =$ the B-splines basis functions

$k =$ B-splines order

Advantages of NURBS

Most general, can represent:

- B-splines
- Bézier and rational Bézier
- conic sections



Properties:

- local control
- convex hull (if $w_i \geq 0$)
- variation diminishing (if $w_i \geq 0$)
- invariant under both affine and projective transformations

Standard tool for representing
freeform curves in CAGD applications

[Farin]

How to Choose a Spline

Hermite curves are good for single segments when you know the parametric derivative or want easy control of it

Bézier 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)