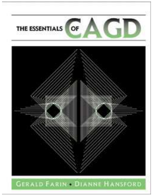
The Essentials of CAGD

Chapter 10: B-Spline Curves

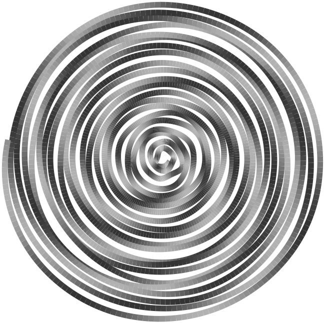
Gerald Farin & Dianne Hansford

CRC Press, Taylor & Francis Group, An A K Peters Book
www.farinhansford.com/books/essentials-cagd



Outline

Introduction to B-Spline Curves



B-spline curves provide a more complete theory of splines compared with composite Bézier curves

Sometimes called NURBS (Non-Uniform Rational B-Splines)

Basic Definitions

Bézier curve

$$\mathbf{x}(t) = \mathbf{b}_0 B_0^n(t) + \ldots + \mathbf{b}_n B_n^n(t)$$

– Properties determined by basis functions B_i^n

– Each Bernstein basis function is a polynomial function

B-spline curve

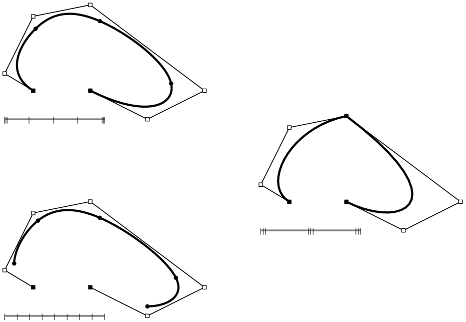
$$\mathbf{x}(u) = \mathbf{d}_0 N_0^n(u) + \ldots + \mathbf{d}_{D-1} N_{D-1}^n(u)$$

– Defined by piecewise polynomial basis functions

– $N_i^n(u)$ are the degree n B-splines

– de Boor points or *control points* $\mathbf{d}_i$

Basic Definitions



Three cubic B-spline curves

- Each has same number of de Boor points
- Number of polynomial segments?
- Continuity?

Basic Definitions

Degree n B-spline curve defined by control polygon

$$\mathbf{d}_0, \ldots, \mathbf{d}_{D-1}$$

Also defined by a knot sequence

$$u_0, \ldots, u_{K-1} \quad \text{where } u_{i+1} \geq u_i$$

Up to n consecutive knots may coincide

$$D = K - n + 1$$

D equal to number of consecutive n-tuples of knots

Basic Definitions

Domain knots

$$u_{n-1}, \ldots, u_{K-n}$$

Parameter values within this range used for evaluating a B-spline curve

u_{n-1} is the last knot in the first n-tuple
u_{K-n} is the first knot in the last n-tuple

Up to n knots may coincide
– Number of coincident values is the multiplicity

If the first and last n knots are multiplicity n
$\Rightarrow$ Curve passes through the first and last de Boor points

Knot with multiplicity one called a simple knot

Basic Definitions

If $u_i = u_{i+1}$ then the interval $[u_i, u_{i+1}]$ has length zero

Number of polynomial segments L equal to the number of nonzero length intervals within the domain knots

If all interior domain knots $u_n, \ldots, u_{K-n-1}$ are simple
$\Rightarrow L = K - 2n + 1$ or $L = D - n$

Span: interval $[u_i, u_{i+m}]$ for $m > 0$
– Span of length m
– Number of spans of length n equals number of legs of the control polygon

Basic Definitions

Example: Top cubic curve in previous Figure
Knot sequence – Number of knots $K = 9$

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
0	0	0	1	2	3	4	4	4

$u_0 = u_1 = u_2 \Rightarrow$ multiplicity 3
$u_6 = u_7 = u_8 \Rightarrow$ multiplicity 3
All other knots are simple knots

Number of de Boor points $D = 9 - 3 + 1 = 7$

Domain knots $u_2, \ldots, u_6$ (solid circle on the curve)
First and last circle correspond to the first and last de Boor point

$L = 4$ polynomial segments

Basic Definitions

Example: Middle cubic curve in Figure
Knot sequence – Number of knots $K = 9$

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
0	0	0	1	1	1	2	2	2

Number of control points $D = 7$

Domain knots: $u_2, \ldots, u_6$

Multiplicity of the knots equal to the degree
$\Rightarrow$ curve passes through the de Boor points
– Influences the smoothness of the curve segments

$L = 2$ polynomial segments

Basic Definitions

Example: Bottom cubic curve in Figure

Knot sequence – Number of knots $K = 9$

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
0	1	2	3	4	5	6	7	8

All knots simple

Number of control points $D = 7$

Domain knots are $u_2, \ldots, u_6$

$L = 4$ polynomial segments

Some texts add one extra knot at either end of the knot sequence
– Not necessary
– Made popular by a flaw in the data exchange standard IGES

The de Boor Algorithm

B-spline curves evaluated using the de Boor algorithm
– Named after Carl de Boor who did pioneering work on B-splines
– Algorithm uses repeated linear interpolation

Let evaluation parameter u be within domain knots
Determine the index I such that

$$u_I \leq u < u_{I+1} \qquad \Rightarrow u \in [u_I, u_{I+1}) \subset [u_{n-1}, u_{K-n}]$$

Exception: $u = u_{K-n}$ then set $I = K - n - 1$ last domain interval

The de Boor Algorithm

The de Boor algorithm computes

$$\mathbf{d}_i^k(u) = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} \mathbf{d}_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} \mathbf{d}_i^{k-1}(u)$$

$$\text{for} \quad k = 1, \ldots, n, \quad \text{and}$$
$$i = l - n + k + 1, \ldots, l + 1$$

The point on the curve is

$$\mathbf{x}(u) = \mathbf{d}_{l+1}^n(u)$$

The de Boor Algorithm

Convenient schematic tool – triangular diagram:

$$
\begin{array}{cccc}
\mathbf{d}_{I-n+1} & & & \\
\vdots & \mathbf{d}^1_{I-n+2} & & \\
\vdots & \vdots & & \\
\mathbf{d}_{I+1} & \mathbf{d}^1_{I+1} & \vdots & \mathbf{d}^n_{I+1}
\end{array}
$$

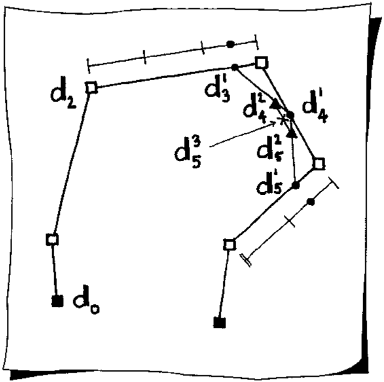
One evaluation involves $n+1$ de Boor points
$\Rightarrow$ B-splines known for local control

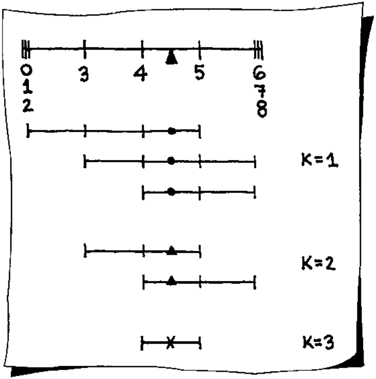
Geometric interpretation of the de Boor algorithm:
– Each step is simply linear interpolation
– May be viewed as an affine map

$$[u_{i+n-k}, u_{i-1}] \quad \Rightarrow \quad \mathbf{d}^{k-1}_{i-1}, \mathbf{d}^{k-1}_{i}$$

Point $\mathbf{d}^k_i$ is the image of u under this affine map

The de Boor Algorithm





The de Boor Algorithm

Example:

Linear ($n = 1$) B-spline curve given by the control polygon

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

and the knot sequence

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ u_0 & u_1 & u_2 & u_3 \end{array}$$

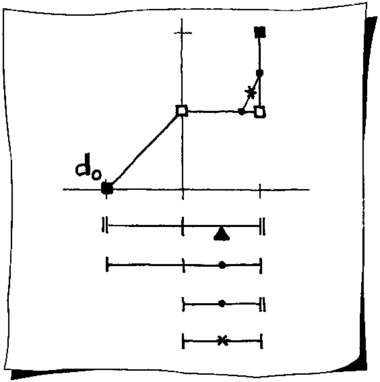
Number of segments $L = 3$
Evaluate at parameter value $u = 1.5$
Parameter value in knot interval $[u_1, u_2] \Rightarrow I = 1$
Only one stage with $i = 2$

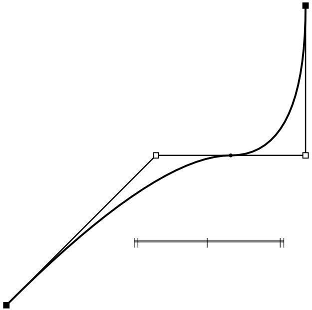
$$\mathbf{d}_2^1(u) = \frac{u_2 - u}{u_2 - u_1}\mathbf{d}_1^0(u) + \frac{u - u_1}{u_2 - u_1}\mathbf{d}_2^0(u)$$

$$\mathbf{x}(1.5) = \mathbf{d}_2^1(1.5) = 0.5\begin{bmatrix} 0 \\ 1 \end{bmatrix} + 0.5\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

The de Boor Algorithm

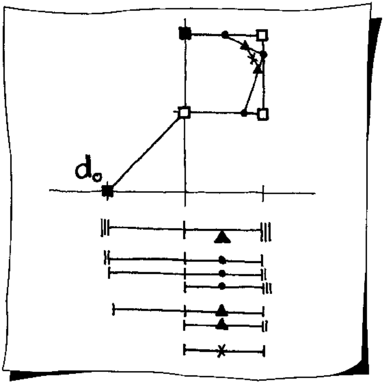
Example: quadratic ($n = 2$)

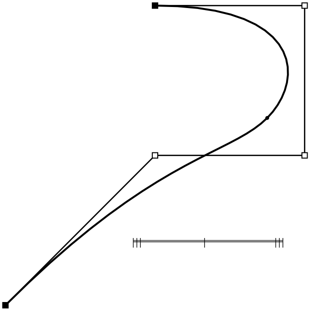




The de Boor Algorithm

Example: cubic ($n = 3$)





Practicalities of the de Boor Algorithm

Take a look at knot multiplicity and a data structure

Evaluation for display:
– Choose an increment to step along the curve
– For piecewise polynomials: specify increment for each segment
(Avoid missing a piece of the curve)
– Segments correspond to non-zero length knot intervals
– Want to avoid plotting zero-length segments

$\Rightarrow$ Label non-zero length segments as part of data structure

Practicalities of the de Boor Algorithm

Expanded knot sequence: Floating point array with every knot stored explicitly

Alternative approach:
– Store only the unique floating point values
– Create an integer array indicating knot multiplicity
$\Rightarrow$ knot multiplicity vector

Example:

0.0	0.0	0.0	1.0	2.0	3.0	3.0	4.0	5.0	5.0	5.0
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
3	0	0	1	1	2	0	1	3	0	0

Example:

5.0	6.0	10.0	11.0	12.5
u_0	u_1	u_2	u_3	u_4
1	1	1	1	1

Search only within the *domain knots* for non-zero length intervals

Practicalities of the de Boor Algorithm

Given parameter value u

$$u \in [u_I, u_{I+1}) \subset [u_{n-1}, u_{K-n}] \quad \text{and} \quad u_I \neq u_{I+1}$$

(Exception for $u = u_{K-n}$: set $I = K - n - 1$ – the last domain interval)

Determine interval it is in and multiplicity r
– If $u = u_I \Rightarrow r$ is multiplicity of u_I, otherwise $r = 0$

Simplify the de Boor algorithm

$$\mathbf{d}_i^k(u) = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} \mathbf{d}_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} \mathbf{d}_i^{k-1}(u)$$

$$\text{for} \quad k = 1, \ldots, n - r, \quad \text{and}$$
$$i = I - n + k + 1, \ldots, I + 1$$

The point on the curve is

$$\mathbf{x}(u) = \mathbf{d}_{I+1-r}^{n-r}(u)$$

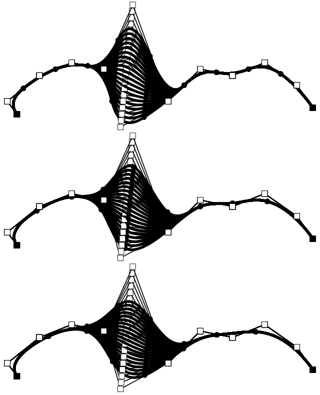
Properties of B-spline Curves

Affine invariance

Differentiability:
At a simple knot u_i curve is C^{n-1}
At knot with multiplicity r curve is C^{n-r}

Endpoint interpolation:
Full multiplicity at end knots $\Rightarrow$ curve will pass through end control points
If $u_0 = \ldots = u_{n-1} \quad \Rightarrow \quad \mathbf{x}(u_{n-1}) = \mathbf{d}_0$
If $u_{K-n} = \ldots = u_{K-1} \quad \Rightarrow \quad \mathbf{x}(u_{K-n}) = \mathbf{d}_{D-1}$

Properties of B-spline Curves



Local control:

Change a control point $\mathbf{d}_i$

$\Rightarrow$ Only the closest $n+1$ curve segments change

Curve degrees (from top): $n = 2, 3, 4$

– Affected curve areas become larger as the degree increases

Property clear from de Boor algorithm

Makes B-spline curves flexible

– Can modify only part of curve

Properties of B-spline Curves

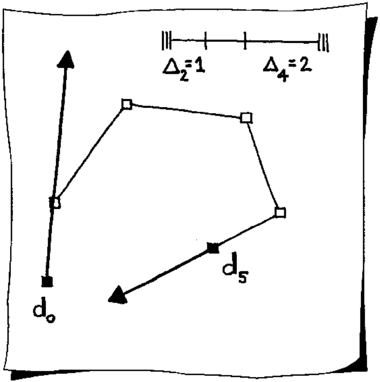
Bézier curves:
For some very special knot sequence configurations
B-spline curves are actually Bézier curves
Conditions:

$$K = 2n - 1$$
$$u_0 = \ldots = u_{n-1}$$
$$u_n = \ldots = u_{2n-1}$$

Example: cubic with knot sequence $0, 0, 0, 1, 1, 1$

de Boor algorithm "collapses" to the de Casteljau algorithm
$\Rightarrow$ B-spline curves are a true superset of Bézier curves

Properties of B-spline Curves



Endpoint derivatives:
If the knot sequence has end knots of multiplicity n

$$\dot{\mathbf{x}}(u_{n-1}) = \frac{n}{u_n - u_{n-1}}[\mathbf{d}_1 - \mathbf{d}_0]$$

$$\dot{\mathbf{x}}(u_{K-n}) = \frac{n}{u_{K-n} - u_{K-n-1}}[\mathbf{d}_{D-1} - \mathbf{d}_{D-2}]$$

Properties of B-spline Curves

Convex hull:
Each point on the curve lies within the convex hull of the control polygon

Each point on the curve lies within the convex hull of no more than $n + 1$ nearby control points

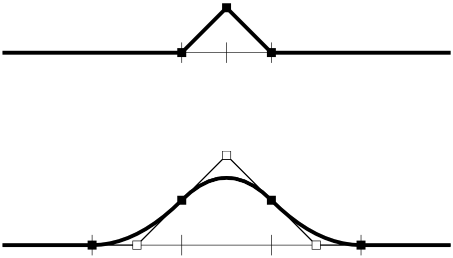
B-splines: The Building Block

B-splines: the basis functions for B-spline curves

– Generalization of Bernstein polynomials

– Composed of several polynomial pieces

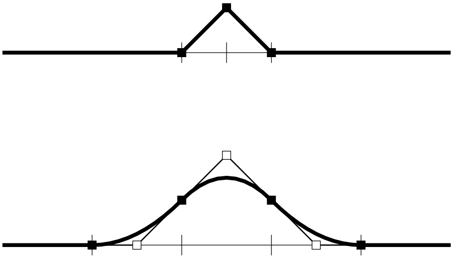
– Pieces fit together with certain smoothness



Two piecewise polynomials

Top: piecewise linear and C^0 Bottom: piecewise quadratic and C^1

B-splines: The Building Block



In Figure: Bézier points of each polynomial segment
– Endpoints of each polynomial marked by solid squares

A B-spline is zero almost everywhere
It assumes nonzero values only for a finite interval
– This region called the function's support

B-splines: The Building Block

Degree n B-spline functions:

$$\mathbf{d}_i = \begin{bmatrix} \xi_i \\ d_i \end{bmatrix} \qquad \text{where} \quad \xi_i = \frac{1}{n}(u_i + \ldots + u_{i+n-1})$$

d_i are called the control ordinates of the function

The ξ_i are the Greville abscissae
– They are moving averages of the knots
– Number of $n-$tuples of consecutive knots equals number of ξ_i
$\Rightarrow$ As many Greville abscissae as there are control points

B-splines: The Building Block

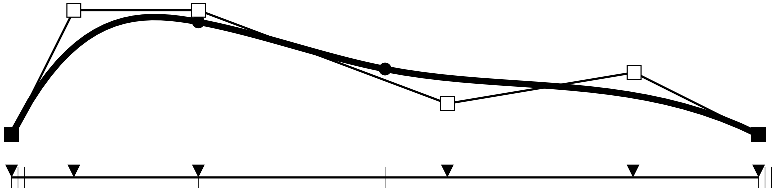
Example: A cubic B-spline function with knot sequence

0	0	0	3	6	12	12	12
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7

Greville abscissae:

0	1	3	7	10	12
ξ_0	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5

Depicted by solid triangular marks



B-splines: The Building Block

Application of B-spline functions: plot B-splines
– For some k: $d_k = 1$ and $d_i = 0$ for all other control ordinates
– Corresponding B-spline function called $N_k^n(u)$

Every piecewise polynomial function $f(u)$ may be written as a combination of these B-splines:

$$f(u) = d_0 N_0^n(u) + \ldots + d_{D-1} N_{D-1}^n(u)$$

Every parametric B-spline curve may be written as

$$\mathbf{x}(u) = \mathbf{d}_0 N_0^n(u) + \ldots + \mathbf{d}_{D-1} N_{D-1}^n(u)$$

N_i^n, also called basis splines (or B-splines for short)

B-splines: The Building Block

B-splines satisfy the recursion

$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u)$$

Recursion is anchored by the definition

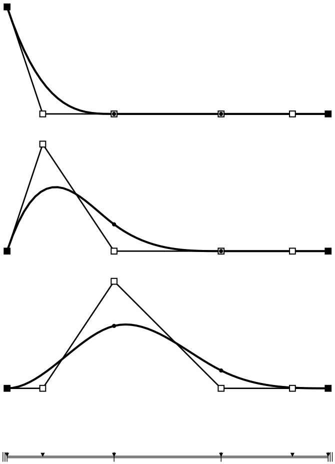
$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i, \\ 0 & \text{else} \end{cases}$$

Describes each degree n basis function as a linear blend of two degree $n-1$ basis functions

– Starts with the piecewise constant basis function

– Recall: similar concept with the de Casteljau algorithm

B-splines: The Building Block



The cubic B-splines N_0^3,N_1^3, and N_2^3 over the given knot sequence

B-splines: The Building Block

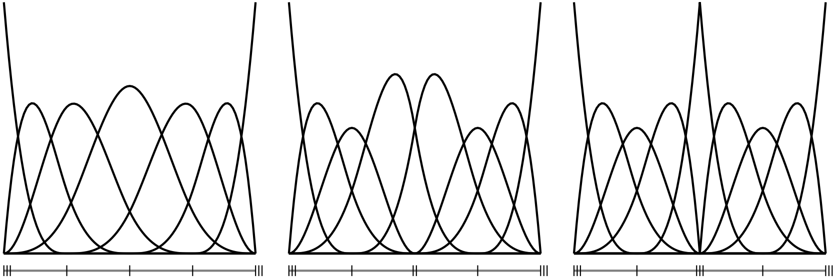
Properties of B-splines:

1. Partition of unity:

$$N_0^n(u) + \ldots + N_{D-1}^n(u) \equiv 1$$

2. Linear precision: If the d_i are sampled at the ξ_i from a linear function: $d_i = a\xi_i + b$
   $\Rightarrow$ corresponding B-spline function is that linear function
3. Local support: Every B-spline is nonzero only over $n+1$ intervals: $N_i^n(u) > 0$ only if $u \in [u_{i-1}, u_{i+n})$

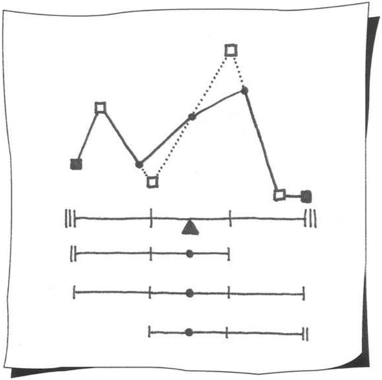
B-splines: The Building Block



All cubic B-splines over the three given knot sequences
Notice the multiplicity

Knot Insertion

A tool for adding a knot $\Rightarrow$ creating a refined control polygon
– Trace of the curve same as the original curve



0	0	0	1	2	3	3	3
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7

Note ξ_i and $\mathbf{d}_i$ positions

Add $u = 1.5$
$\Rightarrow$ New knot sequence and Greville abscissae:

0	0	0	1	1.5	2	3	3	3
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8

$\Rightarrow$ Refined control polygon

Process of refinement known as corner cutting

Knot Insertion

Application of knot insertion: The de Boor algorithm

First stage: parameter u is inserted into the polygon
$\Rightarrow$ Results in a refined polygon

When the knot is inserted n times $\Rightarrow$ point on the curve

de Boor algorithm does not modify the knot sequence or the polygon
– Leaves it in original form for the next evaluation

Knot Insertion

Application of knot insertion:
Converting from B-spline to piecewise Bézier form

B-spline curves are piecewise polynomials
$\Rightarrow$ Must exists a Bézier polygon for each piece
All knots multiplicity n then B-spline polygon is a Bézier polygon
$\Rightarrow$ Insert every knot in the knot sequence to full multiplicity n

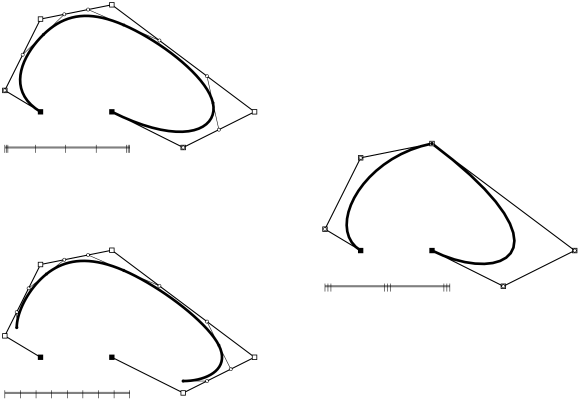
Example:
Given cubic B-spline curve with knot sequence

$$\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 2 & 3 & 3 & 3 \\ u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \end{array}$$

Insert knots

$$\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & u_9 & u_{10} & u_{11} \end{array}$$

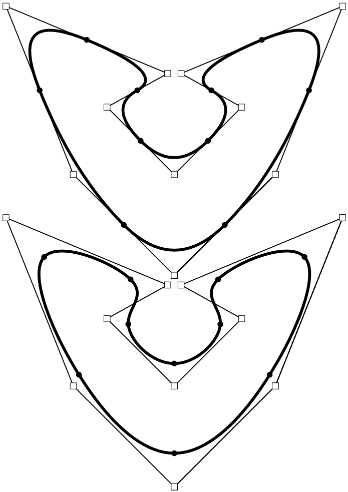
Knot Insertion



Piecewise Bézier polygons for the three curves

- The Bézier polygon approximates the curve more closely
- Many calculations are easier for Bézier curves than B-splines
- The order in which the knots are inserted doesn't matter

Periodic B-spline Curves



Two B-spline curves which are seemingly without beginning or end
Top: quadratic   Bottom: cubic

Periodic B-spline Curves

Periodic B-spline curve constructed as special case of a "normal" one

Goals:

– Seamless control polygon

– Evaluation at first and last domain knot produce the same point

Recall: de Boor algorithm involves only $n+1$ control points

$\Rightarrow$ Number of control points that must overlap

– First $2n-2$ knot intervals influence the position of the "first" point

– Last $2n-2$ intervals influence the position of the "last" point

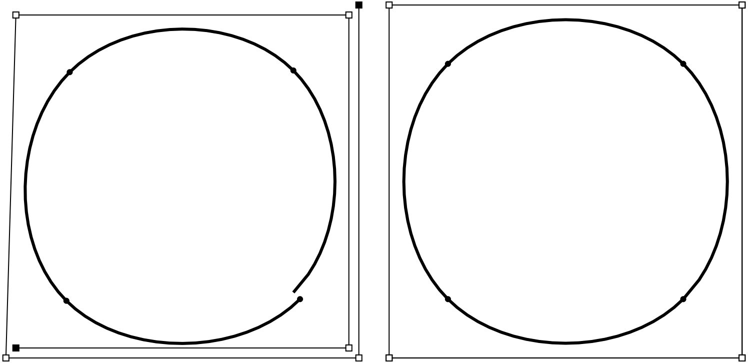
Let $\Delta_i = u_{i+1} - u_i \Rightarrow$ knot sequence constructed as

$$\Delta_0, \Delta_1, \ldots, \Delta_{2n-3}, \Delta_{2n-2}, \ldots \Delta_{K-2n}, \Delta_0, \Delta_1, \ldots \Delta_{2n-3},$$

and the de Boor points such that

$$\mathbf{d}_0 = \mathbf{d}_{D-n}, \quad \mathbf{d}_1 = \mathbf{d}_{D-(n-1)}, \quad \ldots, \quad \mathbf{d}_(n-1) = \mathbf{d}_{D-1}$$

Periodic B-spline Curves



Cubic example:

Left: not "quite" periodic

Knot sequence: $0, 1, 2, 3, 4, 5, 6, 7, 8$ (curve evaluated between $[2, 6]$)

First control point is solid square in the lower left corner

Right: Truly periodic

Derivatives

By differentiating the N_i^n and manipulating the indices
$\Rightarrow$ The first derivative for a B-spline curve:

$$\dot{\mathbf{x}}(u) = n[\mathbf{f}_0 N_1^{n-1} + \ldots + \mathbf{f}_{i-1} N_i^{n-1} + \ldots + \mathbf{f}_{D-2} N_{D-1}^{n-1}]$$

$$\mathbf{f}_{i-1} = \frac{\Delta \mathbf{d}_{i-1}}{u_{n+i-1} - u_{i-1}} \qquad i = 1, \ldots, D-1$$

The de Boor algorithm provides an easy way to implement this
Points $\mathbf{d}_I^{n-1}(u)$ and $\mathbf{d}_{I+1}^{n-1}(u)$ span the curve's tangent:

$$\dot{\mathbf{x}}(u) = \frac{n}{u_{I+1} - u_I}[\mathbf{d}_{I+1}^{n-1}(u) - \mathbf{d}_I^{n-1}(u)]$$

– Involves knot sequence spans of length n
– Similar to the first derivative of a Bézier curve computed via the de Casteljau algorithm

Derivatives

If B-spline curve has multiplicity n at the ends

$$\dot{\mathbf{x}}(u_{n-1}) = \frac{n}{u_n - u_{n-1}}[\mathbf{d}_1 - \mathbf{d}_0]$$

$$\dot{\mathbf{x}}(u_{K-n}) = \frac{n}{u_{K-n} - u_{K-n-1}}[\mathbf{d}_{D-1} - \mathbf{d}_{D-2}]$$

Derivatives

The second derivative:

$$\ddot{\mathbf{x}}(u) = n(n-1)[\mathbf{s}_1 N_2^{n-2} + \ldots + \mathbf{s}_{i-1} N_i^{n-2} + \ldots + \mathbf{s}_{D-2} N_{D-1}^{n-2}]$$

$$\mathbf{s}_{i-1} = \frac{\Delta \mathbf{f}_{i-1}}{u_{n+1-2} - u_{i-1}} \qquad i = 2, \ldots, D-1$$

– Involves spans of length $n-1$ Implement via the de Boor algorithm

– Compute the intermediate de Boor points up to $\mathbf{d}_i^{n-2}$
– Remaining two steps of the algorithm are modified as follows:

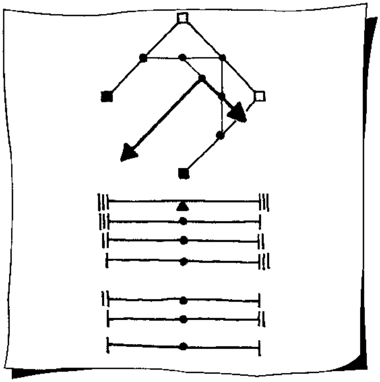
$$\mathbf{d}_i^k(u) = \frac{-k}{u_{i+n-k} - u_{i-1}} \mathbf{d}_{i-1}^{k-1}(u) + \frac{k}{u_{i+n-k} - u_{i-1}} \mathbf{d}_i^{k-1}(u)$$

$$\text{for} \quad k = n-1, n, \quad \text{and} \qquad i = I - n + k + 1, \ldots, I + 1$$

Then the second derivative is

$$\ddot{\mathbf{x}}(u) = \mathbf{d}_{I+1}^n(u)$$

Derivatives

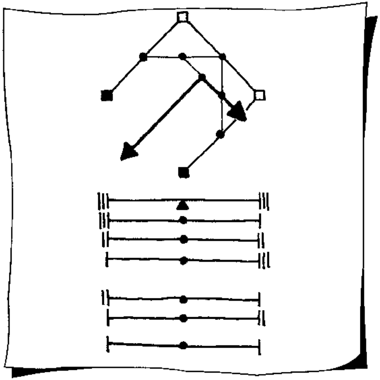


Example: cubic curve
Knot sequence $0, 0, 0, 2, 2, 2$
One cubic Bézier curve

Evaluate curve at $u = 1.0$
de Boor algorithm produces

$$
\begin{array}{cccc}
\mathbf{d}_i^0 & \mathbf{d}_i^1 & \mathbf{d}_i^2 & \mathbf{d}_i^3 \\
\begin{bmatrix} -1 \\ 0 \end{bmatrix} & & & \\
\begin{bmatrix} 0 \\ 1 \end{bmatrix} & \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix} & & \\
\begin{bmatrix} 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} & \begin{bmatrix} 0 \\ 1/2 \end{bmatrix} & \\
\begin{bmatrix} 0 \\ -1 \end{bmatrix} & \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix} & \begin{bmatrix} 1/2 \\ 0 \end{bmatrix} & \begin{bmatrix} 1/4 \\ 1/4 \end{bmatrix}
\end{array}
$$

Derivatives



Example: continued

First derivative: $\mathbf{d}_i^2$:

$$\dot{\mathbf{x}}(1.0) = \frac{3}{2}[\begin{bmatrix}1/2\\0\end{bmatrix} - \begin{bmatrix}0\\1/2\end{bmatrix}] = \begin{bmatrix}3/4\\-3/4\end{bmatrix}$$

Second derivative:

Begins with $\mathbf{d}_i^1$ and execute the modified de Boor algorithm:

$$\begin{matrix}\begin{bmatrix}-1/2\\1/2\end{bmatrix} & & \\ \begin{bmatrix}1/2\\1/2\end{bmatrix} & \begin{bmatrix}1\\0\end{bmatrix} & \\ \begin{bmatrix}1/2\\-1/2\end{bmatrix} & \begin{bmatrix}0\\-1\end{bmatrix} & \begin{bmatrix}-3/2\\-3/2\end{bmatrix}\end{matrix}$$

$$\ddot{\mathbf{x}}(1.0) = \begin{bmatrix}-3/2\\-3/2\end{bmatrix}$$