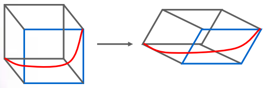
Bézier curves

History

- Renault car design, ’60s
- First practical mathematical model for curves used in design
- Replaced point sets, physical models, and French curves in design
- Could be used for numerical milling machines

Bézier curves

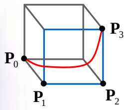
- Based on the idea of cube deformation (to parallelepiped)



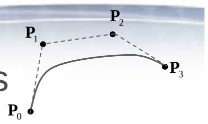
- Embedded cubic curve to opposite corners, tangential to edges, is unique

Bézier curves

- Control points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ determine deformations and hence determine the curve

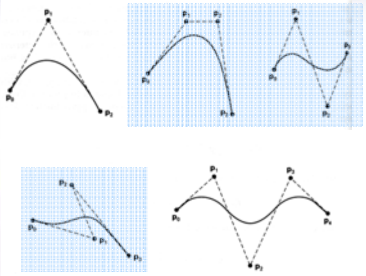


Cubic Bézier curves



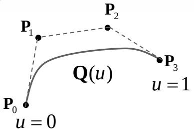
- Determined by 4 control points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$
- Contained in convex hull of control polygon
- End points of curve interpolate (touch) $\mathbf{P}_0$ and $\mathbf{P}_3$
- End of curves tangent to $\mathbf{P}_1 - \mathbf{P}_0$ and $\mathbf{P}_3 - \mathbf{P}_2$
- Curve ’mimicks’ control polygon

Some Bézier curves



Bézier curves

- Formal definition:



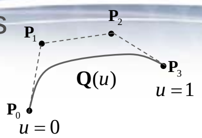
$$\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i B_i(u)$$

$$0 \le u \le 1$$

- Weighted average of control points
- Note: $\mathbf{Q},\mathbf{P}$ are vectors, B is a scalar

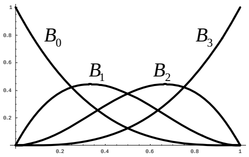
Bézier curves

$$\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i B_i(u)$$



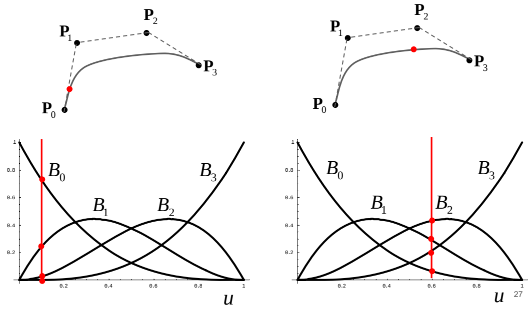
- Weight functions are called blending or basis functions. Sum of weights is always 1
- With Bézier curves, the weights are cubic Bernstein functions:

$$\begin{aligned} B_0(u) &= (1-u)^3 \\ B_1(u) &= 3u(1-u)^2 \\ B_2(u) &= 3u^2(1-u) \\ B_3(u) &= u^3 \end{aligned}$$



Bézier curves $\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i B_i(u)$

Example weights for control points for u=0.1 and u=0.6



Exercise

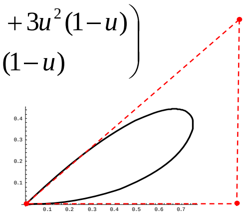
- Draw the 2D Bézier curve with control points

$$\mathbf{P}_0 = \mathbf{P}_3 = (0,0)$$
$$\mathbf{P}_1 = (1,0)$$
$$\mathbf{P}_2 = (1,1)$$

Solution

$$\mathbf{Q}(u) = 3u(1-u)^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 3u^2(1-u) \begin{pmatrix} 1 \\ 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 3u(1-u)^2 + 3u^2(1-u) \\ 3u^2(1-u) \end{pmatrix}$$



Bézier curves

- Because each point on the curve is a weighted average of the control points, the curve cannot leave the convex hull of the control polygon
- Except at the end points, all weights are non-zero → moving a control point affects the entire curve. (No local control)

Bézier curves: matrix form

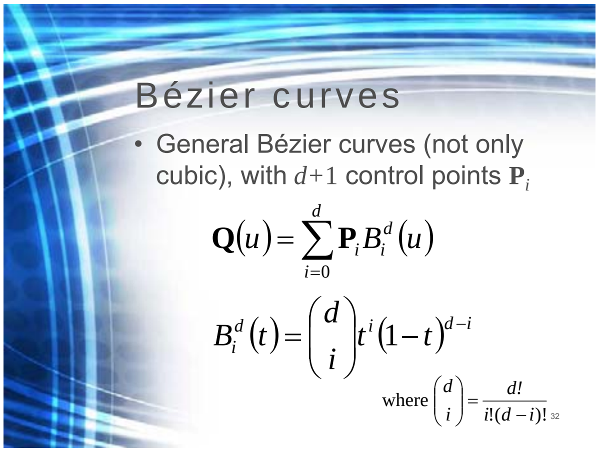
$$\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i B_i(u) \quad \text{Expands to}$$

$$\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$$

Which is often found in the matrix form

$$\begin{aligned} \mathbf{Q}(u) &= \mathbf{U}\mathbf{B}_z\mathbf{P} \\ &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \end{aligned}$$

Note the 'incorrect' notation, treating the P as scalars



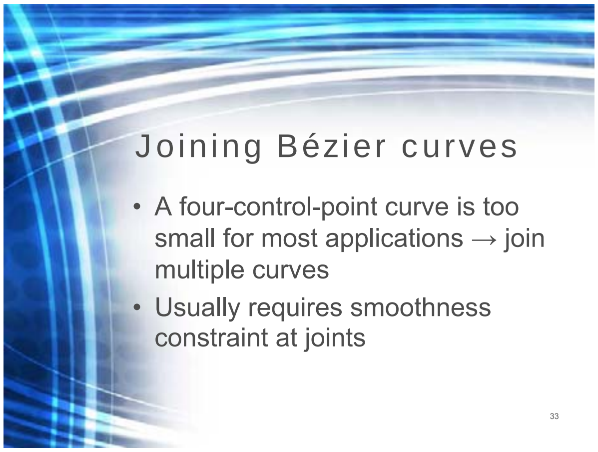
Bézier curves

- General Bézier curves (not only cubic), with $d+1$ control points $\mathbf{P}_i$

$$\mathbf{Q}(u) = \sum_{i=0}^{d} \mathbf{P}_i B_i^d(u)$$

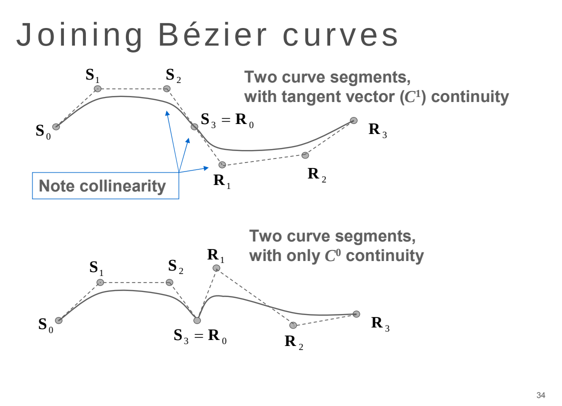
$$B_i^d(t) = \binom{d}{i} t^i (1-t)^{d-i}$$

where $\binom{d}{i} = \frac{d!}{i!(d-i)!}$



Joining Bézier curves

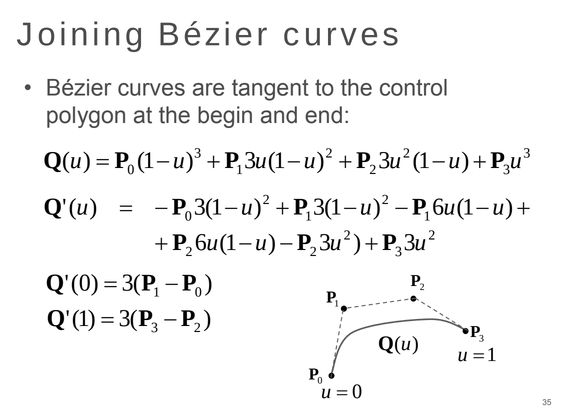
- A four-control-point curve is too small for most applications → join multiple curves
- Usually requires smoothness constraint at joints



Joining Bézier curves

Two curve segments, with tangent vector (C^1) continuity

Two curve segments, with only C^0 continuity



Joining Bézier curves

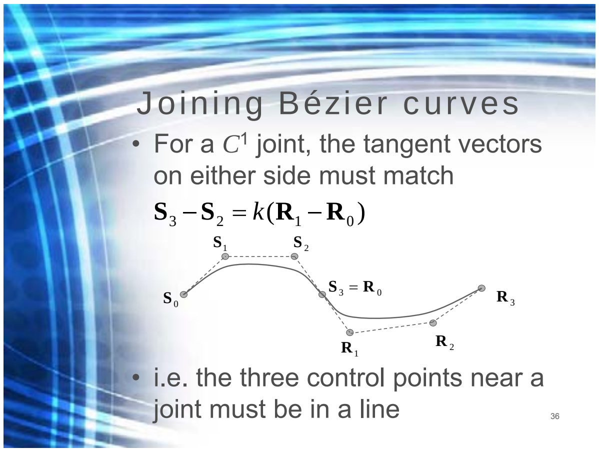
- Bézier curves are tangent to the control polygon at the begin and end:

$$\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$$

$$\begin{aligned}\mathbf{Q}'(u) &= -\mathbf{P}_0 3(1-u)^2 + \mathbf{P}_1 3(1-u)^2 - \mathbf{P}_1 6u(1-u) + \\ &\quad + \mathbf{P}_2 6u(1-u) - \mathbf{P}_2 3u^2) + \mathbf{P}_3 3u^2\end{aligned}$$

$$\mathbf{Q}'(0) = 3(\mathbf{P}_1 - \mathbf{P}_0)$$

$$\mathbf{Q}'(1) = 3(\mathbf{P}_3 - \mathbf{P}_2)$$

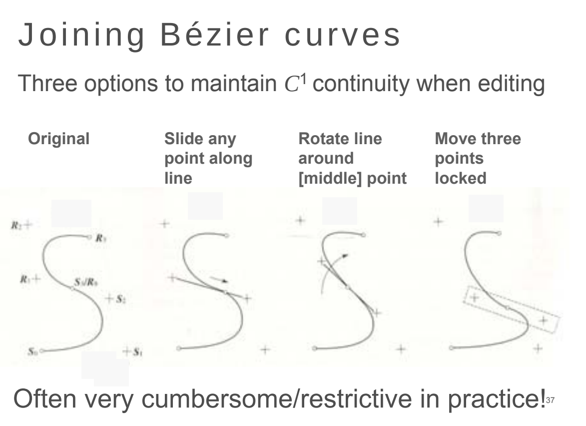


Joining Bézier curves

- For a C^1 joint, the tangent vectors on either side must match

$$\mathbf{S}_3 - \mathbf{S}_2 = k(\mathbf{R}_1 - \mathbf{R}_0)$$

- i.e. the three control points near a joint must be in a line



Joining Bézier curves

Three options to maintain C^1 continuity when editing

Original

Slide any point along line

Rotate line around [middle] point

Move three points locked

Often very cumbersome/restrictive in practice!

Bézier curves: summary

- Polynomial curve, often of degree 3
- Shape ’mimicks’ control polygon
- No local control
- First and last control point touched
- Tangent to first and last control edge
- Curve is invariant (retains shape) under affine transformation of control points

Bézier curves: summary

- Some drawbacks of Bézier curves:
  - No local control
  - Maintaining C^1 continuity of joint segments requires constraints
  - Number of control points (d) determines the degree of the curve (d-1)

These can be overcome by using B-spline curves

B-spline curves

- A *spline* is
  1. a thin wood or metal strip (used in building construction)
  2. a piecewise polynomial curve
- A B-spline curve
  - does not (in general) pass through its control points
  - is a piecewise polynomial (mostly cubic)
  - can have any number of segments
  - depends only on local control points

B-spline curves

- General formulation: weighted combination of control points ($\mathbf{P}$):

$$\mathbf{Q}(u) = \sum_{k=0}^{n} \mathbf{P}_k B_{k,d}(u) \qquad u_{\min} \le u \le u_{\max}$$

$$2 \le d \le n+1$$

n+1	# control points
d-1	degree of polynomials (d=4 for cubic)
B	blending functions

Cubic B-spline curves

- cubic splines (d=4):

$$\mathbf{Q}(u) = \sum_{k=0}^{n} \mathbf{P}_k B_{k,4}(u) \qquad u_{\min} \le u \le u_{\max}$$

Entire curve

$$\mathbf{Q}_i(u) = \sum_{k=0}^{3} \mathbf{P}_{i-3+k} B_{i-3+k}(u) \qquad 0 \le u \le 1$$

Local variant (used by Watt) for segment *i*

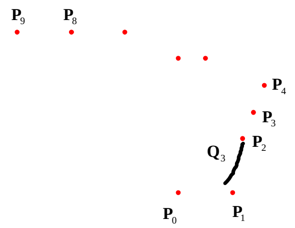
$$\begin{aligned}\mathbf{Q}_i(u) &= \mathbf{U}\mathbf{B}_s\mathbf{P} \\ &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-3} \\ \mathbf{P}_{i-2} \\ \mathbf{P}_{i-1} \\ \mathbf{P}_i \end{bmatrix}\end{aligned}$$

Cubic B-splines

$$\mathbf{Q}_i(u) = \sum_{k=0}^{3} \mathbf{P}_{i-3+k} B_{i-3+k}(u)$$

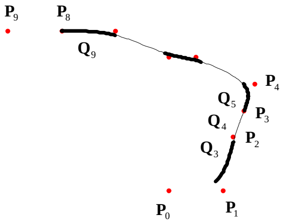
First segment is $\mathbf{Q}_3$.

$\mathbf{Q}_3$ depends on $\mathbf{P}_0 \ldots \mathbf{P}_3$



Cubic B-splines

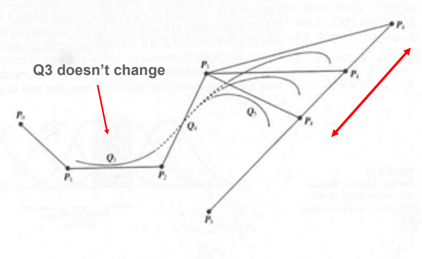
$$\mathbf{Q}_i(u)=\sum_{k=0}^{3}\mathbf{P}_{i-3+k}B_{i-3+k}(u)$$



- **Each segment $\mathbf{Q}_i$ depends on four control points**
- **# segments = (# control points) – 2**
- **In general no end point interpolation**
- **Blending functions are C^2 continuous → so is $\mathbf{Q}$ (linear combination of Bs)**

Cubic B-splines

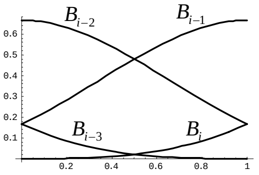
- Since each point of the curve depends on four control points only, we have local control



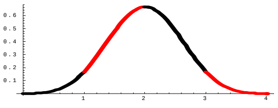
Uniform Cubic B-splines

$$\mathbf{Q}_i(u)=\sum_{k=0}^{3}\mathbf{P}_{i-3+k}B_{i-3+k}(u)$$

- **Uniform** B spline: the blending functions are all identical



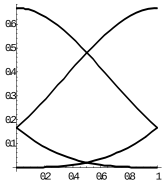
Blending functions on one segment $\mathbf{Q}_i$



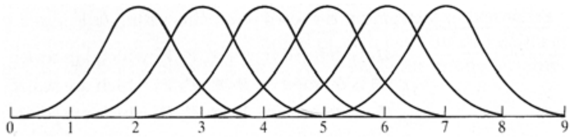
The four cubic segments form into C^2 curve

Uniform Cubic B-splines

$$\mathbf{Q}_i(u)=\sum_{k=0}^{3}\mathbf{P}_{i-3+k}B_{i-3+k}(u)$$



- Repetition of the blending function gives all the weights for the entire curve
- Now note that why the '-3' is in the formula for $\mathbf{Q}_i$: only from $u=3$ are there four functions active, and the sum of weights$=1$

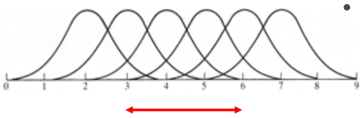


Uniform Cubic B-splines

- The entire curve:

$$\mathbf{Q}(u)=\sum_{k=0}^{n}\mathbf{P}_k B_{k,4}(u) \qquad u_{\min}\le u\le u_{\max}$$

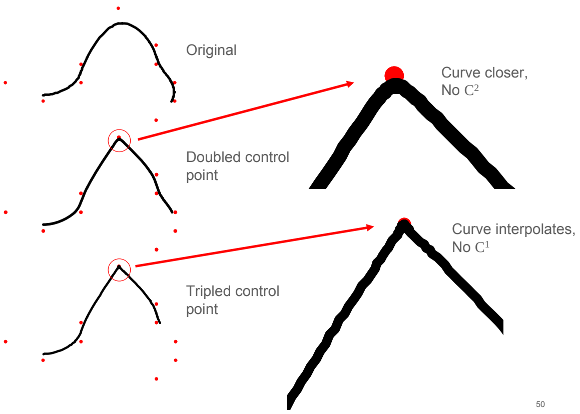
- $u_{\min}$ and $u_{\max}$ are determined by the range of u where there are four blending functions active.
- Values for B come from shifting the standard B



- e.g. $3\le u\le 6$ for 5 control points

Uniform Cubic B-splines

- In general, B-splines do not interpolate (touch) points
- Doubling a control point draws the curve nearer
- Tripling a control point forces interpolation
- Theoretically, the curve is still C^2, but the geometric continuity is reduced.



Original

Doubled control point

Tripled control point

Curve closer, No C^2

Curve interpolates, No C^1

Non-uniform B-splines

- A better way than control point duplication is to specify where blending functions (bf) have to start
- **Knot =** A start or end point of a bf
- **Knot vector =** Collection of all knots
- By using knots, we can enforce there are always 4 bf active → end and mid point interpolation possible
- bf need to be rewritten (to retain sum=1 and other properties). Each (cubic) bf is determined by a group of 5 knots

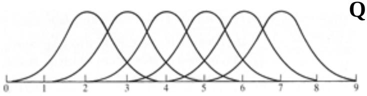
---

Example: end point interpolation

- With 6 control points, the normal knot vector u_i with uniform B-splines is [0,1,2,3,4,5,6,7,8,9] and

$$3 \le u \le 6$$

$$\mathbf{Q}(u) = \sum_{k=0}^{5} \mathbf{P}_k B_{k,4}(u)$$



(Because six control points means $B_0 \ldots B_5$ start at u=0,1,2,3,4,5, and at u=6,7,8,9 bf end)

[0,1,2,3,4,5,6,7,8,9]

Determines shape of B_0

Determines shape of B_1

Etc.

Since 5 points are needed for each bf, we need a knot vector of length 9 for 6 control points

---

Example (cont.)

$$\mathbf{Q}(u) = \sum_{k=0}^{5} \mathbf{P}_k B_{k,4}(u)$$

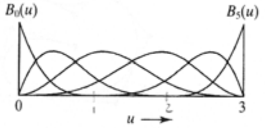
- We now change the knot vector (of 9 entries) such that 4 bf start at the first control point, and 4 end at the last, so the end points are interpolated

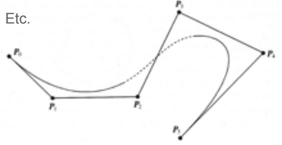
[0,0,0,0,1,2,3,3,3,3]

Determines shape of B_0

Determines shape of B_1

Etc.





Non-uniform B-splines: Generating the Blending functions

$$\mathbf{Q}(u) = \sum_{k=0}^{n} \mathbf{P}_k B_{k,d}(u)$$

- Cox-De Boor recursive algorithm

$$B_{i,1}(u) = \begin{cases} 1 & u_i \le u \le u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,d}(u) = \frac{u - u_i}{u_{i+d-1} - u_i} B_{i,d-1}(u) + \frac{u_{i+d} - u}{u_{i+d} - u_{i+1}} B_{i+1,d-1}(u)$$

- With knot vector $[u_i]$, degree d-1
- $x/0$ is defined 0