

Parametric Curves and Surfaces

Parametric Polynomial Curves

多项式的

- The functions are all **polynomials** in the parameter.

$$x(u) = a_0 + a_1 u^1 + a_2 u^2 + \cdots + a_n u^n = \sum_{k=0}^n a_k u^k$$

$$y(u) = b_0 + b_1 u^1 + b_2 u^2 + \cdots + b_n u^n = \sum_{k=0}^n b_k u^k$$

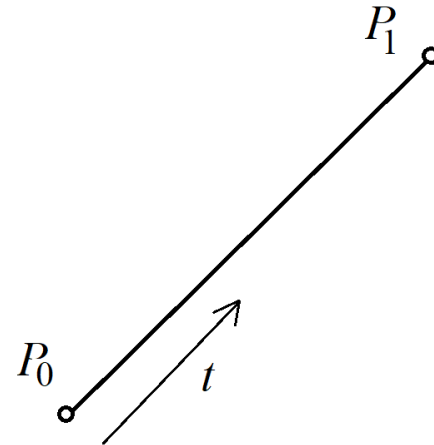
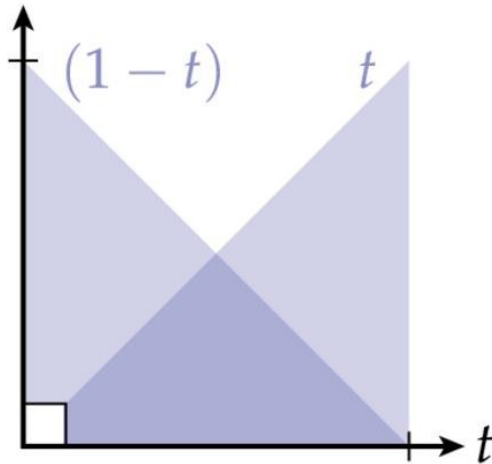
- We'll assume that u varies from 0 to 1
- Pros:
 - efficient to compute
 - infinitely differentiable
 - Generalization to 3D curves is completely straightforward: add $z(u)$
- **Power basis** $(1, u, u^2, \dots)$ form does not reveal geometry of curves.

Linear Interpolation (1D)

- Interpolate values using linear interpolation; in 1D:

$$\hat{f}(t) = (1 - t)f_i + tf_j$$

- Can think of this as a linear combination of two functions:



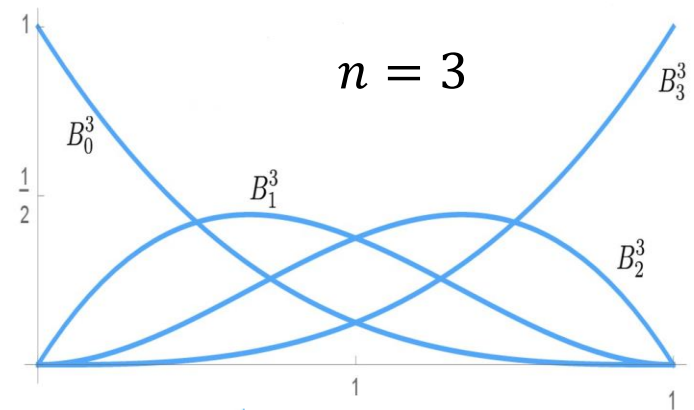
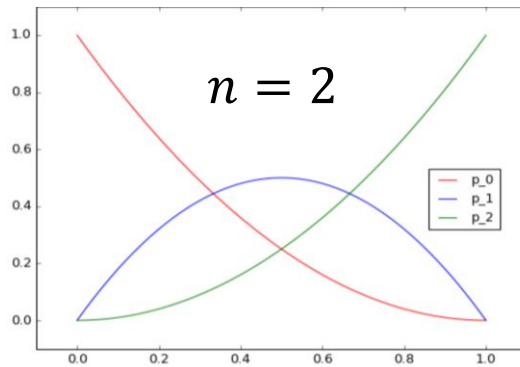
- $(1-t)$ and t are two linear basis functions
 - Each gives the contribution of one point while t varies
- Can we use higher-order bases to get more interesting geometry?

Bernstein Basis

- Provide more flexibility by using higher-order polynomials

$$B_k^n(x) := \binom{n}{k} x^k (1-x)^{n-k}$$

degree \swarrow
 $0 \leq x \leq 1$ \swarrow
 $k=0, \dots, n$ \nwarrow
 "n choose k" \nwarrow



$$B_0^3(u) = (1-u)^3$$

$$B_1^3(u) = 3u(1-u)^2$$

$$B_2^3(u) = 3u^2(1-u)$$

$$B_3^3(u) = u^3$$

Bezier Curves

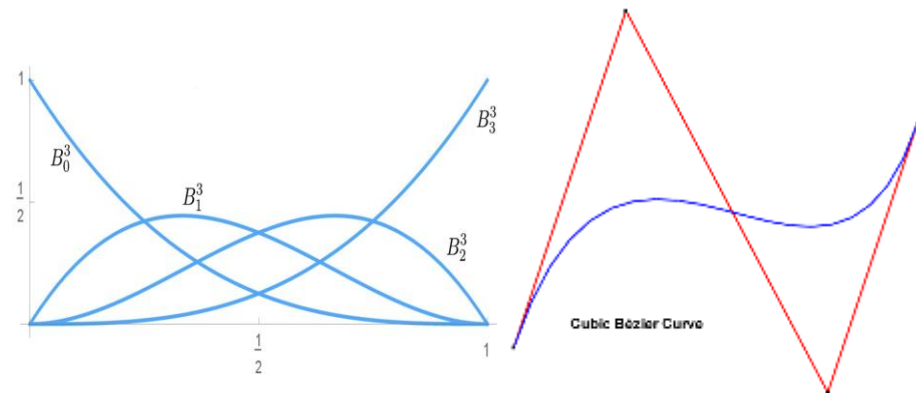
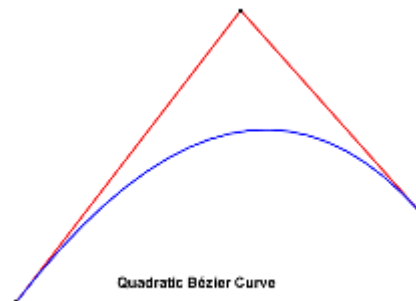
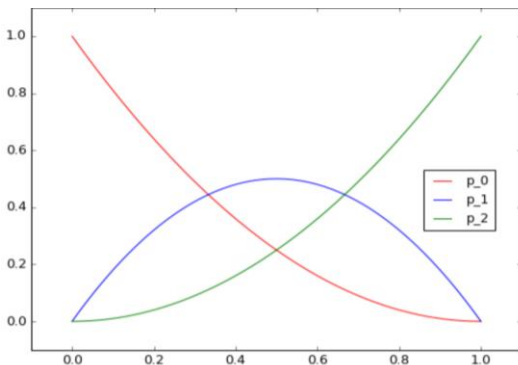
- A Bezier curve is a curve expressed in the Bernstein basis:

$$C(u) := \sum_{k=0}^n B_{n,k}(u) p_k$$

control points

Blending control points using weights computed from basis functions

- For $n=1$, just get a line segment!
- For $n=2$, get quadratic Bezier (parabola)
- For $n=3$, get cubic Bezier



Bezier Curves

- Important features (see next few slides):
 - Interpolates endpoints
 - Tangent to end segments
 - Contained in convex hull
 - Symmetry
 - Affine invariant
 - Variational diminishing
 - ...

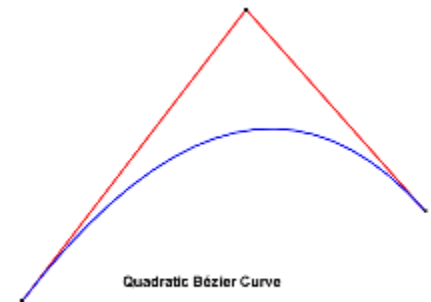
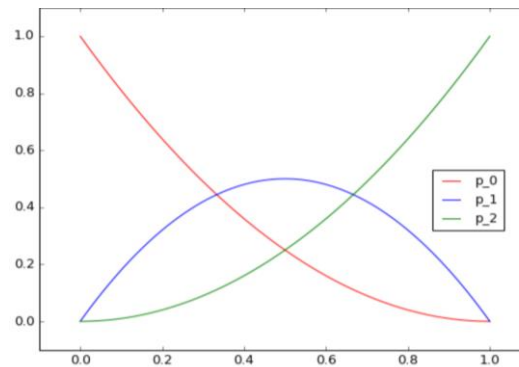
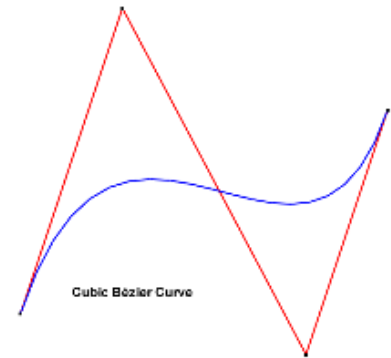
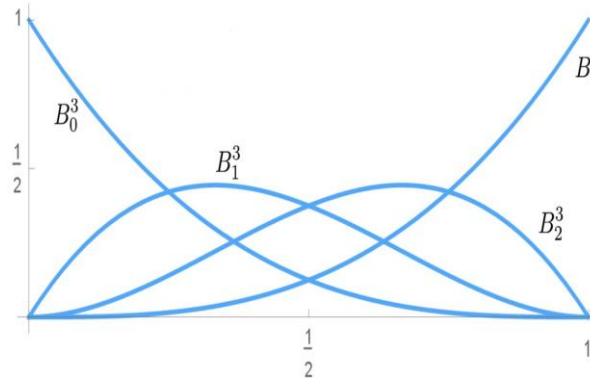
Endpoint interpolations

$$B_0^3(u) = (1-u)^3$$

$$B_1^3(u) = 3u(1-u)^2$$

$$B_2^3(u) = 3u^2(1-u)$$

$$B_3^3(u) = u^3$$



At $u=0$, $B_0(0)=1$, the other functions = 0

At $u=1$, $B_3(1)=1$, the other functions = 0

Tangent to end segments

- For cubic Bezier curves, we have

$$Q(u) = (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u)V_2 + u^3 V_3$$

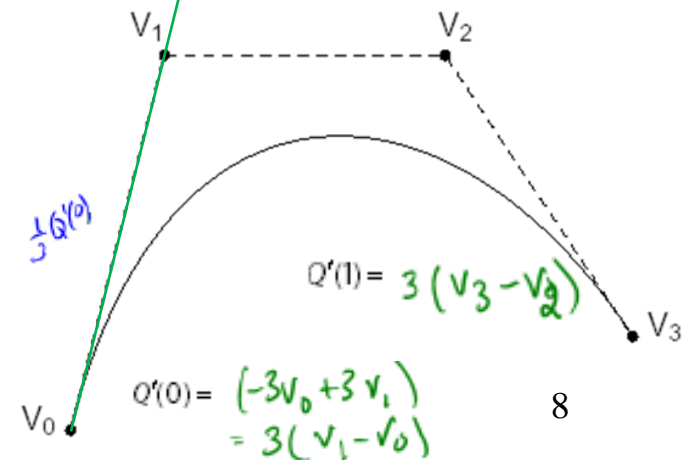
- Expanding the terms in u and rearranging:

$$Q(u) = (-V_0 + 3V_1 - 3V_2 + V_3)u^3 + (3V_0 - 6V_1 + 3V_2)u^2 + (-3V_0 + 3V_1)u + V_0$$

- Differentiating:

$$Q'(u) = 3(-V_0 + 3V_1 - 3V_2 + V_3)u^2 + 2(3V_0 - 6V_1 + 3V_2)u + (-3V_0 + 3V_1)$$

- What are the tangents at endpoints?



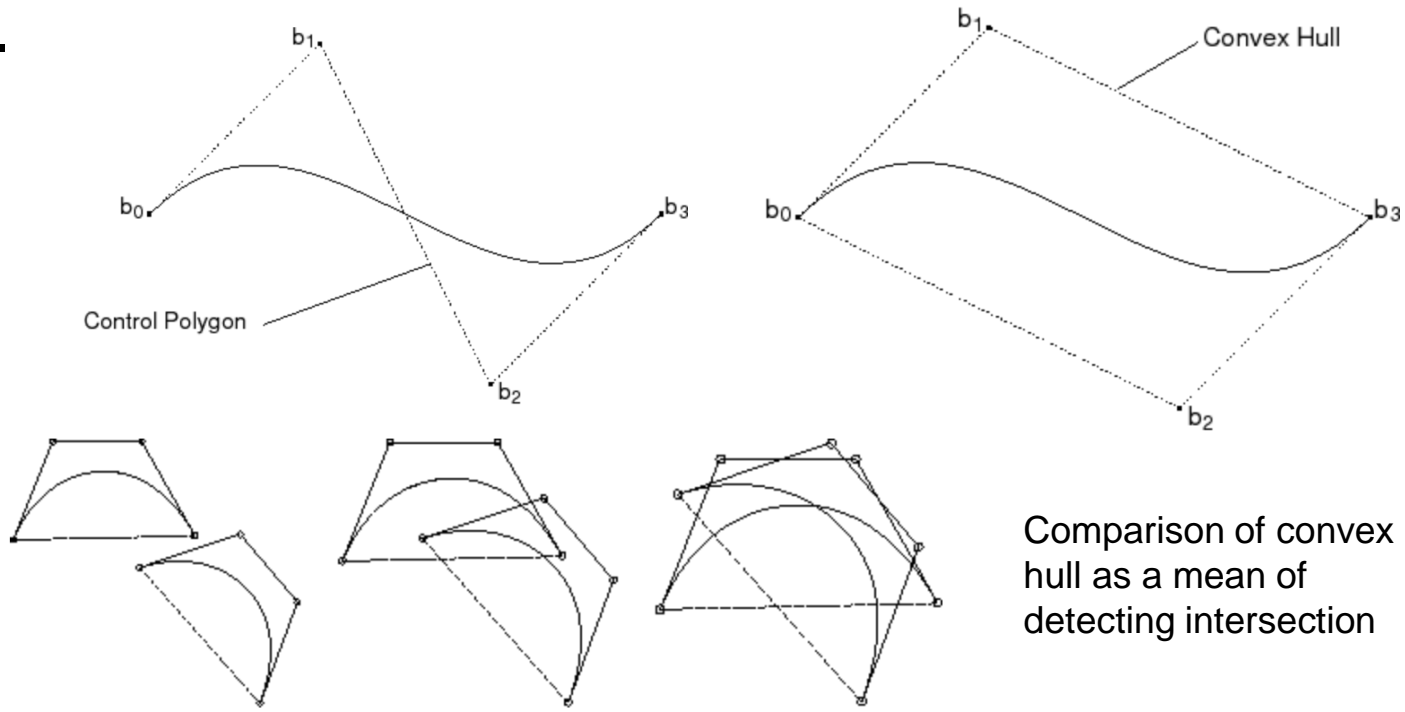
Contained in Convex hull

- Desirable properties:

Affine combination – **sum** of all coefficients is always **exactly 1** (a.k.a. , a “partition of unity”)

Convex combination – each Bernstein coefficient is **positive**

The curve is generated by **convex combinations** of the control points and therefore lies within the **convex hull** of the control points.



Comparison of convex
hull as a mean of
detecting intersection

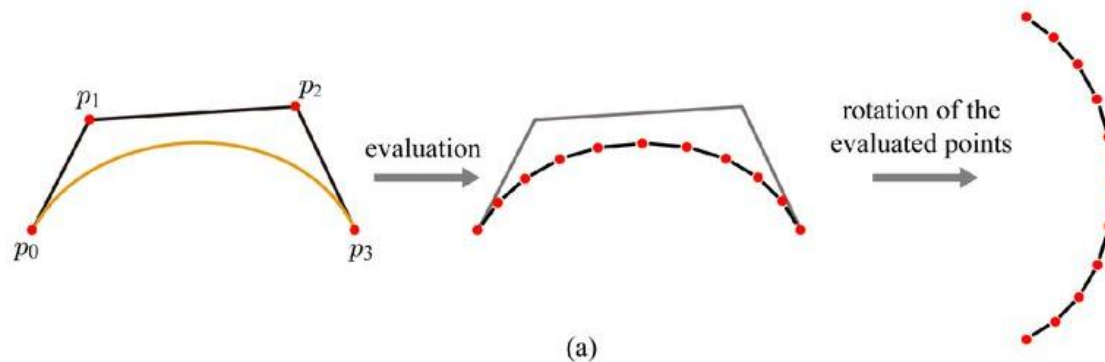
Symmetry

- Relabeling the control points from P_0, P_1, \dots, P_n to P_n, P_{n-1}, \dots, P_0 and using the symmetry property of Bernstein polynomials, we get the same Bezier curve

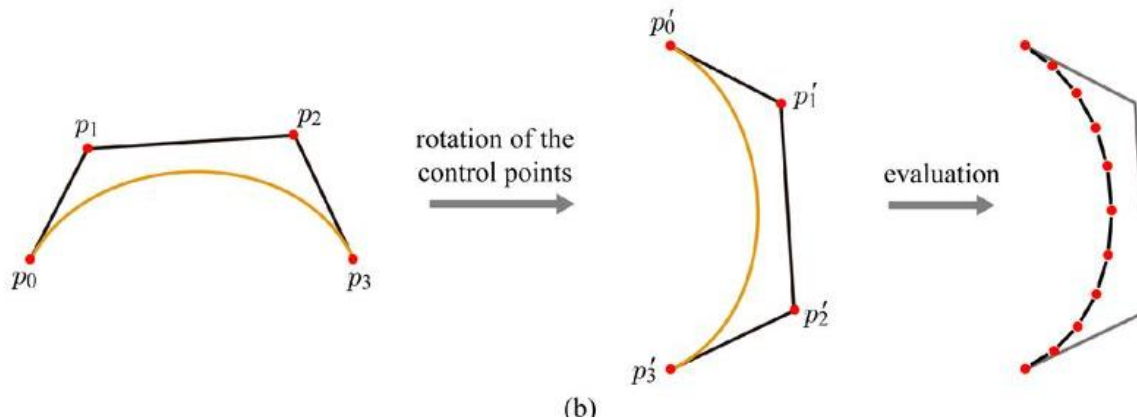
仿射(几何学)的

Affine invariance

- Partition of unity and non-negativity also imply **affine invariance**



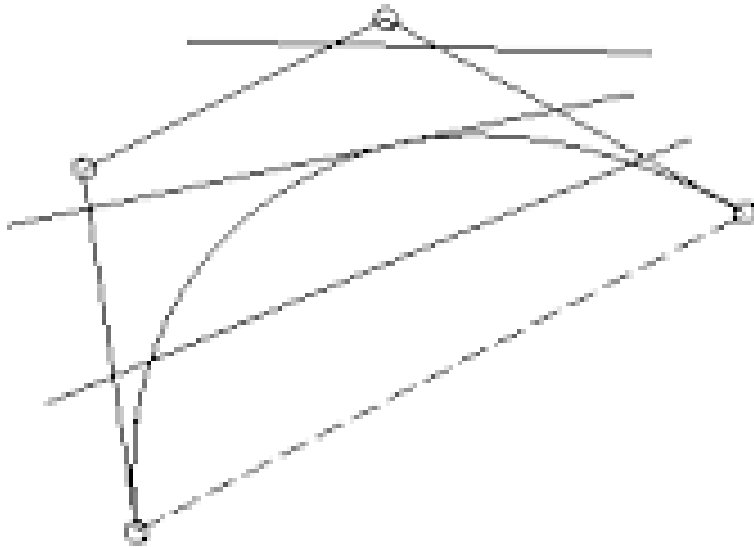
(a)



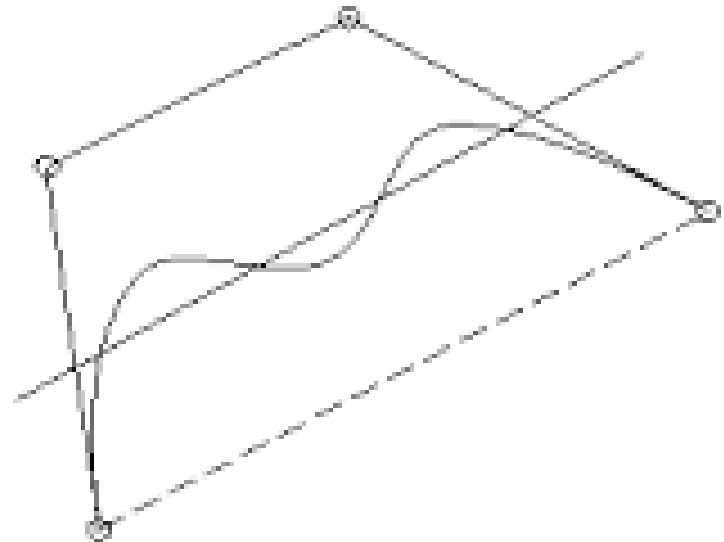
(b)

Variational Diminishing

- Bezier curves are smoother than the polygon formed by their control points.
- Any line drawn through the curve has equal or fewer intersections with the curve than with the control polygon.



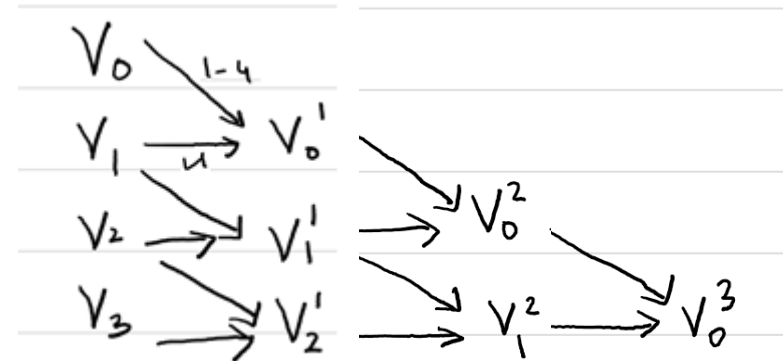
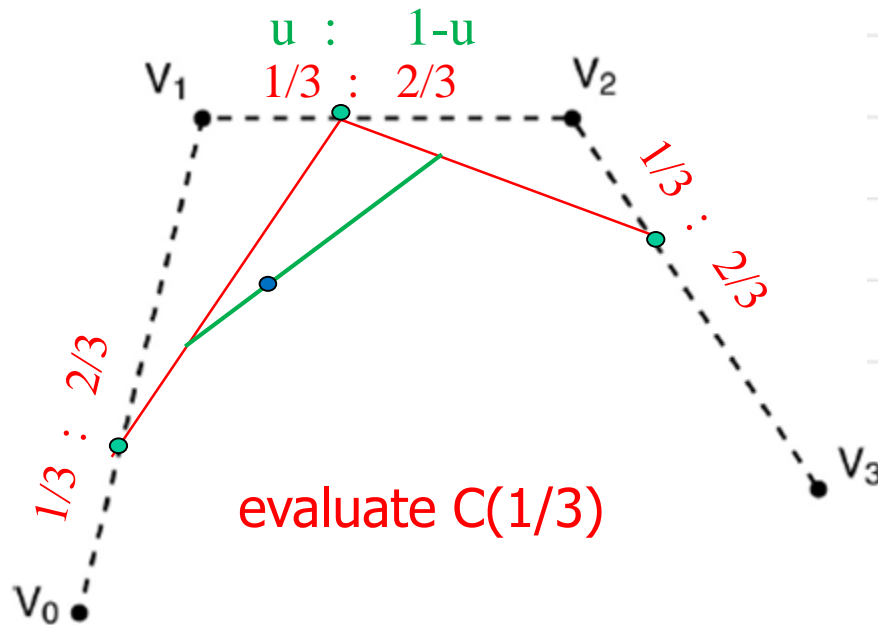
Possible



Impossible

Evaluation by recursive interpolations

- A Bezier curve at a particular u can be evaluated using **deCasteljau's algorithm** (recursive linear interpolation)



- How many linear interpolations for a degree n Bezier curve?

deCasteljau's algorithm

There are several ways to define Bezier curves.

One way is using Bernstein polynomials.

Here, we show another way: via repeated linear interpolations

$$V_0^1 = (1-u)V_0 + uV_1$$

$$V_1^1 = (1-u)V_1 + uV_2$$

$$V_2^1 = (1-u)V_2 + uV_3$$

$$V_0^2 = (1-u)V_0^1 + uV_1^1$$

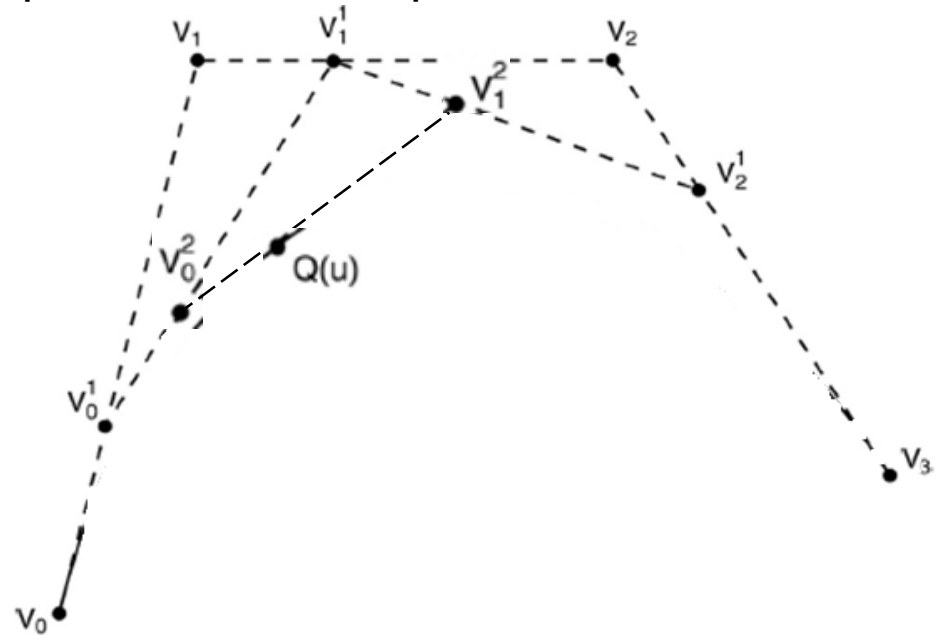
$$V_1^2 = (1-u)V_1^1 + uV_2^1$$

$$Q(u) = (1-u)V_0^2 + uV_1^2$$

$$= (1-u)[(1-u)V_0^1 + uV_1^1] + u[(1-u)V_1^1 + uV_2^1]$$

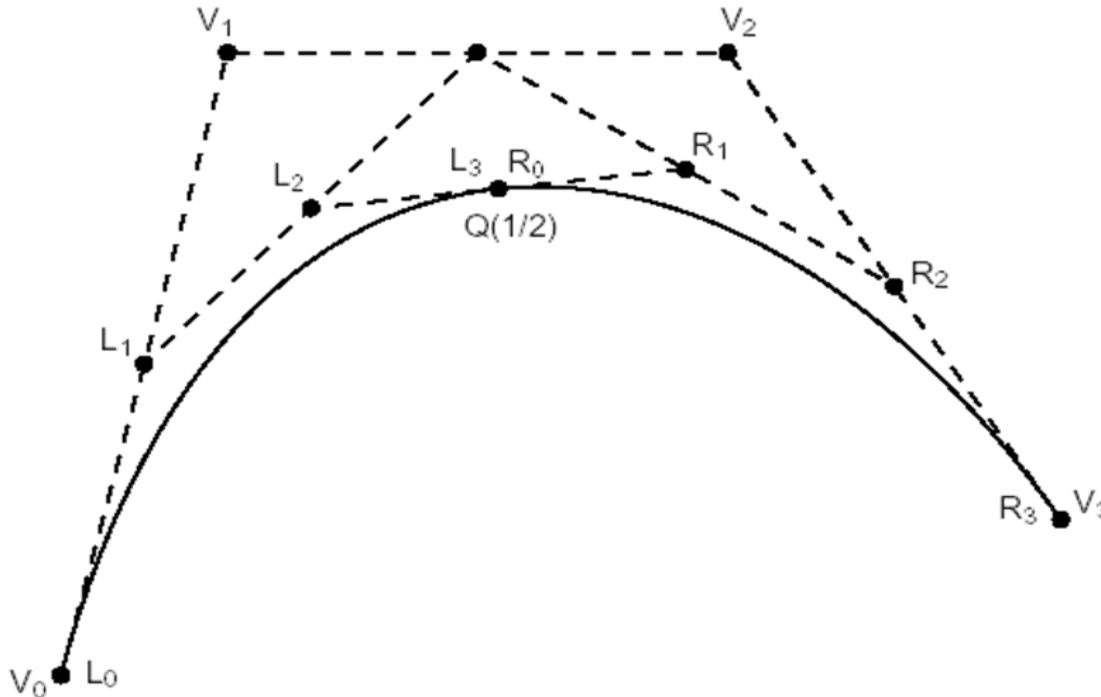
$$= (1-u)[(1-u)\{(1-u)V_0 + uV_1\} + u\{(1-u)V_1 + uV_2\}] + \dots$$

$$= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u)V_2 + u^3 V_3$$



Splitting Bezier curves

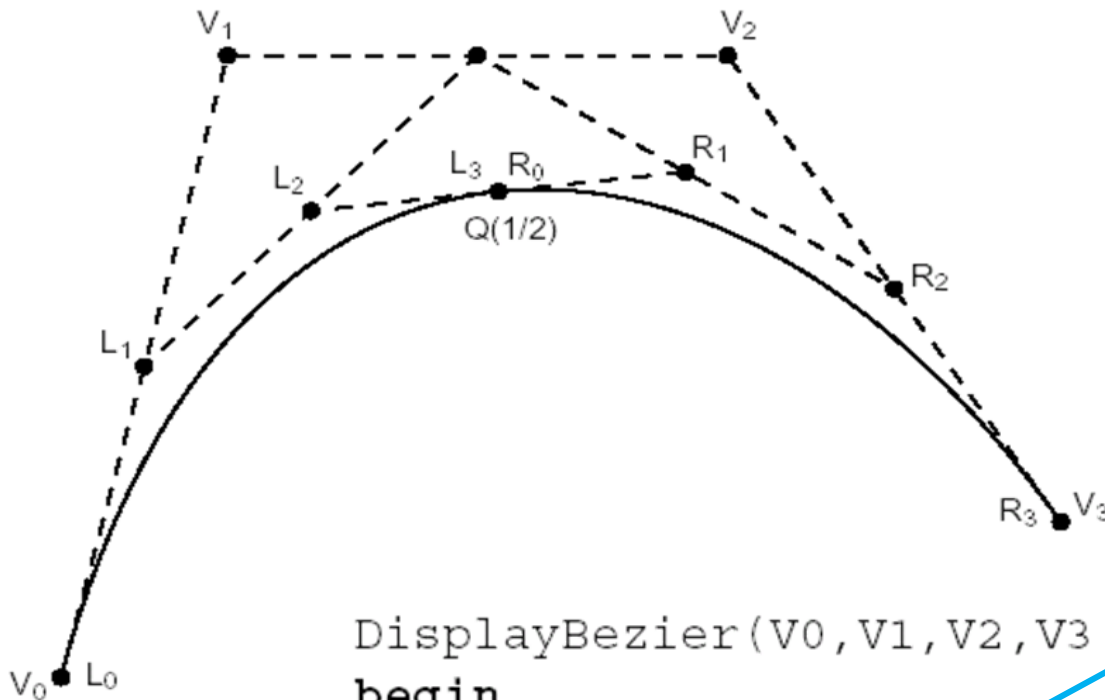
- **deCasteljau's algorithm** is useful for splitting a Bezier curve into two Bezier curves
- Original Bezier curve represented by V_0, V_1, V_2, V_3
- Split into two Bezier curves represented by L_0, L_1, L_2, L_3 and R_0, R_1, R_2, R_3



Useful for

- finding line-Bezier intersection or Bezier-Bezier intersections
- Adaptive display

Subdivide: display



$$\frac{|V_0 - V_1| + |V_1 - V_2| + |V_2 - V_3|}{|V_0 - V_3|} < 1 + \varepsilon$$

```

DisplayBezier(V0, V1, V2, V3)
begin
    if ( FlatEnough(V0, V1, V2, V3))
        Line(V0, V3);
    else
        Subdivide(V[]) ⇒ L[], R[]
        DisplayBezier(L0, L1, L2, L3);
        DisplayBezier(R0, R1, R2, R3);
    end
end
    
```

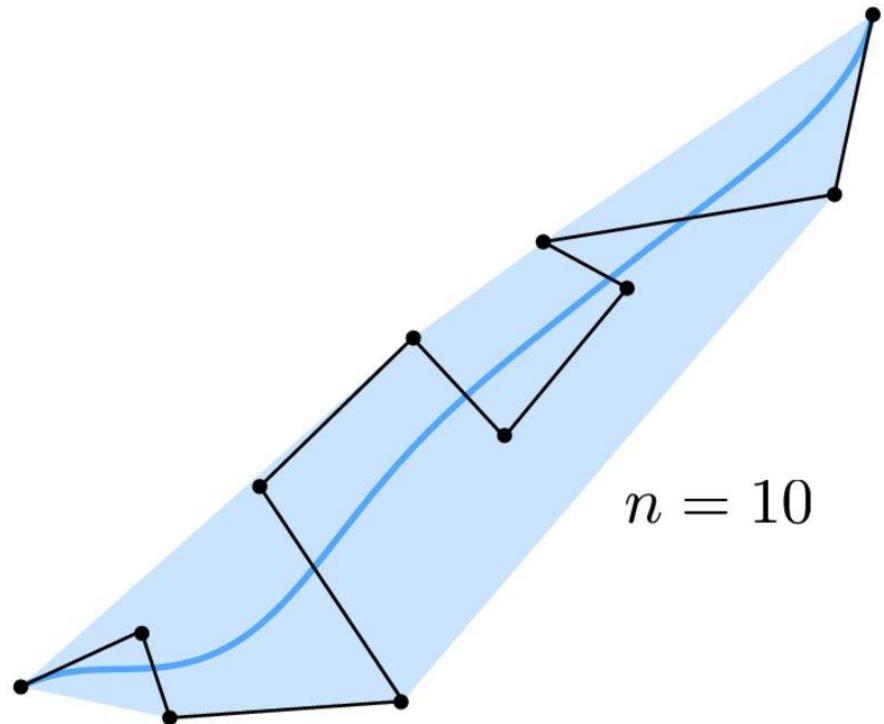

Power-basis form vs Bernstein-basis form

- Mathematically, Bezier curves are equivalent to polynomial curves expressed in power-basis form
- Bezier curves are expressed in terms of meaningful geometric elements (**control points**).

$$\begin{aligned}
 Q(u) &= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\
 &= V_0 + (-3V_0 + 3V_1)u + (3V_0 - 6V_1 + 3V_2)u^2 + (-V_0 + 3V_1 - 3V_2 + V_3)u^3 \\
 &= \begin{bmatrix} V_{0,x} + (-3V_{0,x} + 3V_{1,x})u + (3V_{0,x} - 6V_{1,x} + 3V_{2,x})u^2 + (-V_{0,x} + 3V_{1,x} - 3V_{2,x} + V_{3,x})u^3 \\ V_{0,y} + (-3V_{0,y} + 3V_{1,y})u + (3V_{0,y} - 6V_{1,y} + 3V_{2,y})u^2 + (-V_{0,y} + 3V_{1,y} - 3V_{2,y} + V_{3,y})u^3 \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} a_0 + a_1 u + a_2 u^2 + a_3 u^3 \\ b_0 + b_1 u + b_2 u^2 + b_3 u^3 \\ 1 \end{bmatrix} = \begin{bmatrix} x(u) \\ y(u) \\ 1 \end{bmatrix}
 \end{aligned}$$

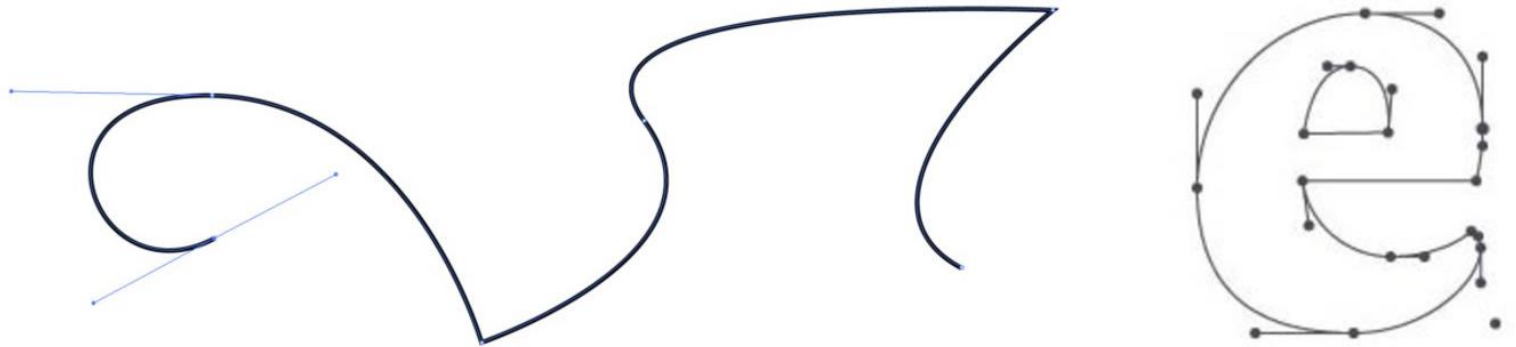
Higher-order polynomials...?

- What if we want more interesting curves than cubic polynomials?
- Use higher-degree polynomials?
 - have more wiggles
 - Very hard to control



Piecewise Bezier curves

- Piece together several Bezier curves to get a Bezier **spline**
- Widely-used (illustrator, fonts, SVG, etc.)



- Formally, **piecewise** Bezier curve:

piecewise Bézier



$$C(u) := C_i \left(\frac{u - u_i}{u_{i+1} - u_i} \right),$$

single Bézier



$$u_i \leq u < u_{i+1}$$

Bezier splines: continuity

- **First-order continuity** means continuous first derivative

$$Q'(u) = \frac{dQ(u)}{du}$$

- $Q'(u)$ is called the **tangent**
- If we think of u as "time" and $Q(u)$ as the path of a particle through space, $Q'(u)$ represents the **velocity** (direction and magnitude)



- Second-order continuity means continuous second derivative

$$Q''(u) = \frac{d^2Q(u)}{du^2}$$

- $Q''(u)$ represents **acceleration** if $Q(u)$ represents a motion curve

Parametric Continuity

- In general, C^n continuity is defined as follows:

$Q(u)$ is C^n continuous
iff
 $Q^{(i)}(u)$ is continuous for $0 \leq i \leq n$

Two curves $Q_1(u)$ and $Q_2(u)$
$$Q_1^{(i)}(1) = Q_2^{(i)}(0)$$

These conditions are *nested*

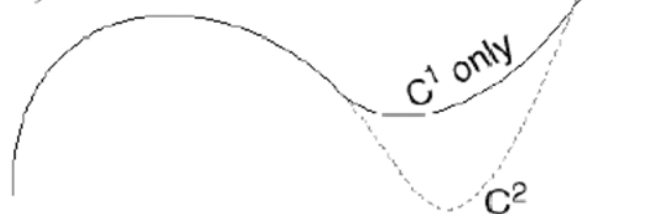
C^{-1} :



C^0 :



C^1, C^2 :



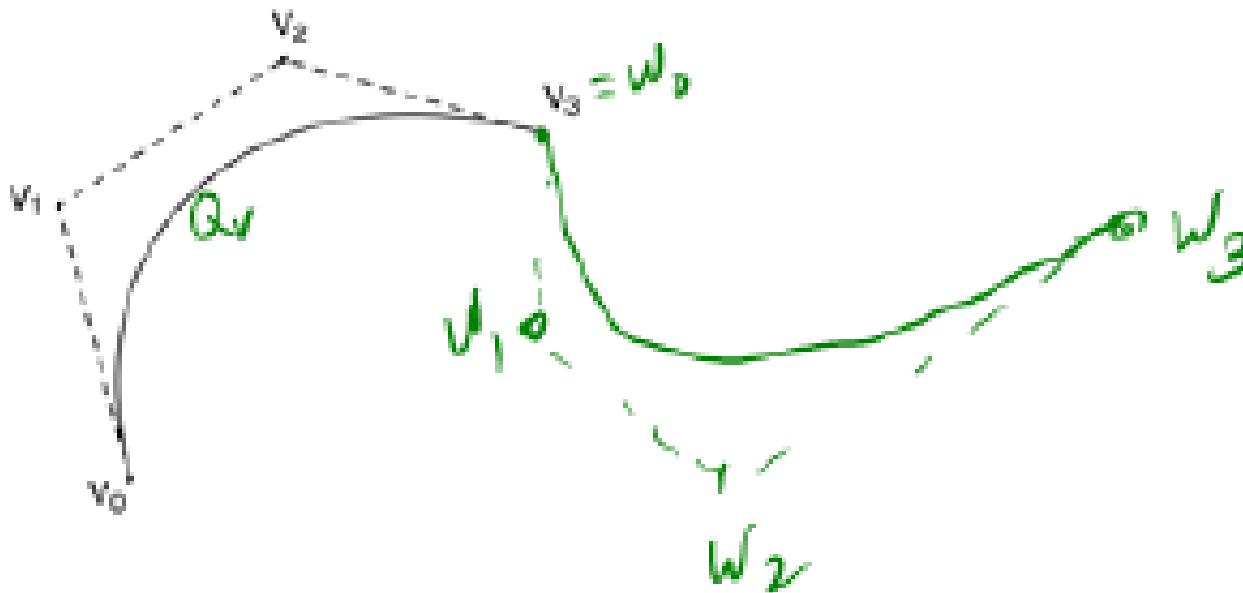
Ensuring C^0 continuity

- $Q_v(u)$ defined by V_0, V_1, V_2, V_3
- $Q_w(u)$ defined by W_0, W_1, W_2, W_3
- Joint is C^0 continuous if

$$C^0 : Q_v(1) = Q_w(0)$$

- What constraint does this place on W_0, W_1, W_2, W_3 ?

$$W_0 = V_3$$

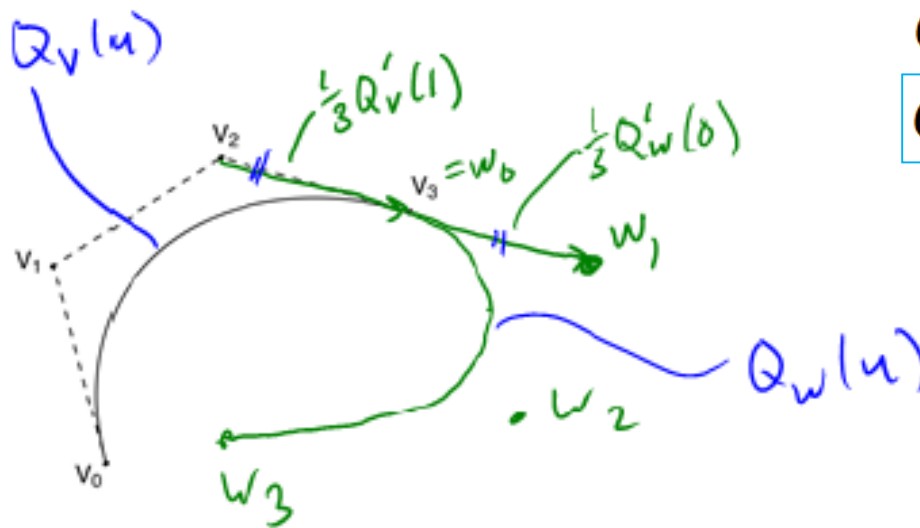


Ensuring C^1 continuity

- Joint is C^1 continuous if

$$C^0 : Q_V(1) = Q_W(0)$$

$$C^1 : Q'_V(1) = Q'_W(0)$$
- What additional constraint does this place on (W_0, W_1, W_2, W_3) ?

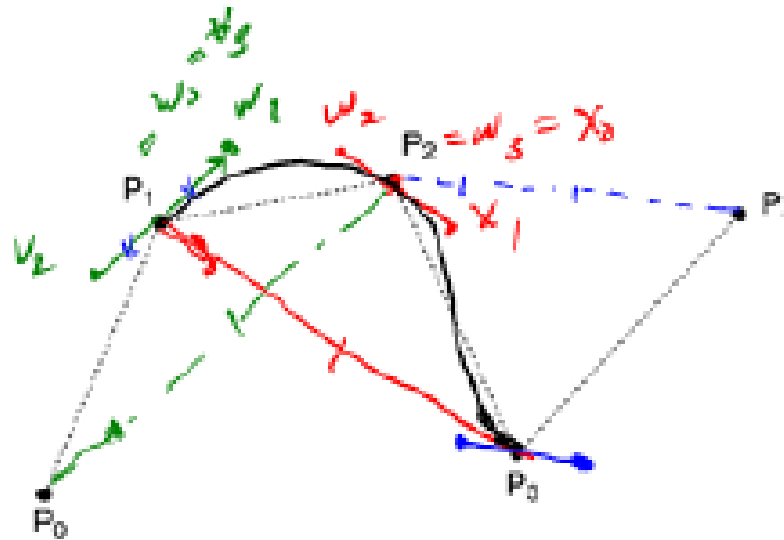


$$C^0 \Rightarrow W_0 = V_3$$

$$C^1 \Rightarrow W_1 - W_0 = V_3 - V_2$$

The C^1 Bezier spline

- How then could we construct a curve passing through a set of points P_0, \dots, P_n with C^1 continuity?
- We can devise a scheme to place the Bezier control points (consider the interior points; end points need special treatment)



$$\begin{aligned} V_0 &= P_1 \\ V_1 &= P_1 + \frac{1}{6}(P_2 - P_0) \\ V_2 &= P_2 - \frac{1}{6}(P_3 - P_1) \\ V_3 &= P_2 \end{aligned}$$

Second-order continuity

- To develop C^2 splines, we need second-order derivatives

$$Q'(u) = 3(-V_0 + 3V_1 - 3V_2 + V_3)u^2 + 2(3V_0 - 6V_1 + 3V_2)u + (-3V_0 + 3V_1)$$

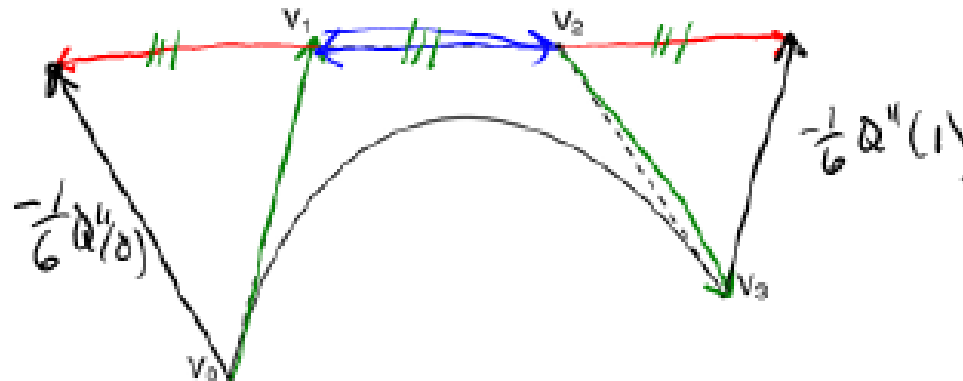
$$Q''(u) = 6(-V_0 + 3V_1 - 3V_2 + V_3)u + 2(3V_0 - 6V_1 + 3V_2)$$

At the two endpoints:

$$Q''(0) = 6(V_0 - 2V_1 + V_2) = -6[(V_1 - V_0) + (V_1 - V_2)]$$

$$-\frac{1}{6} Q''(0) = (V_1 - V_0) + (V_1 - V_2)$$

$$Q''(1) = 6(V_1 - 2V_2 + V_3) = -6[(V_2 - V_3) + (V_2 - V_1)]$$



Ensuring C^2 continuity

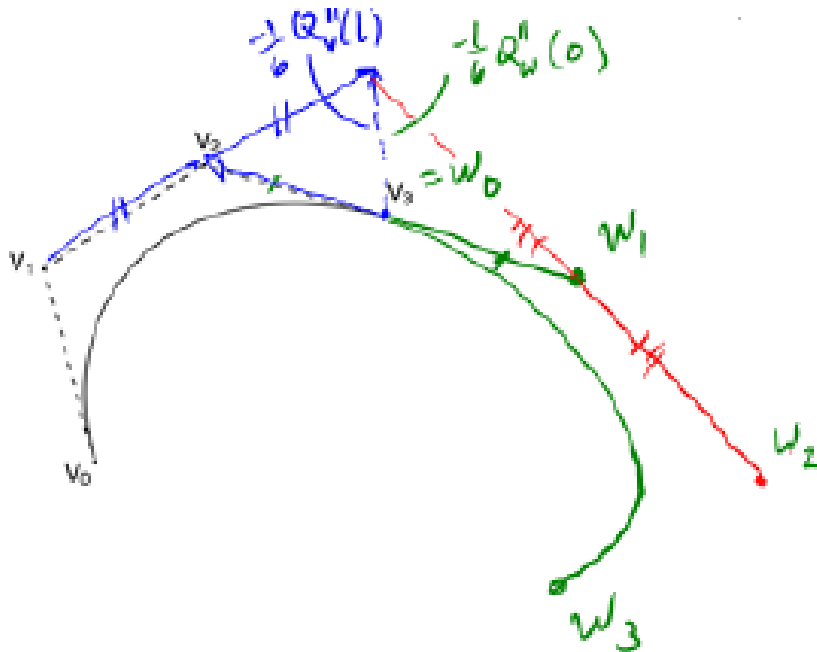
- The joint is C^2 continuous if

$$C^0 : Q_V(1) = Q_W(0)$$

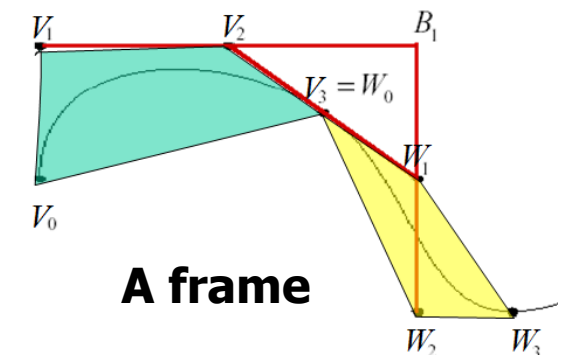
$$C^1 : Q'_V(1) = Q'_W(0)$$

$$C^2 : Q''_V(1) = Q''_W(0)$$

- What additional constraint does this place on (W_0, W_1, W_2, W_3) ?

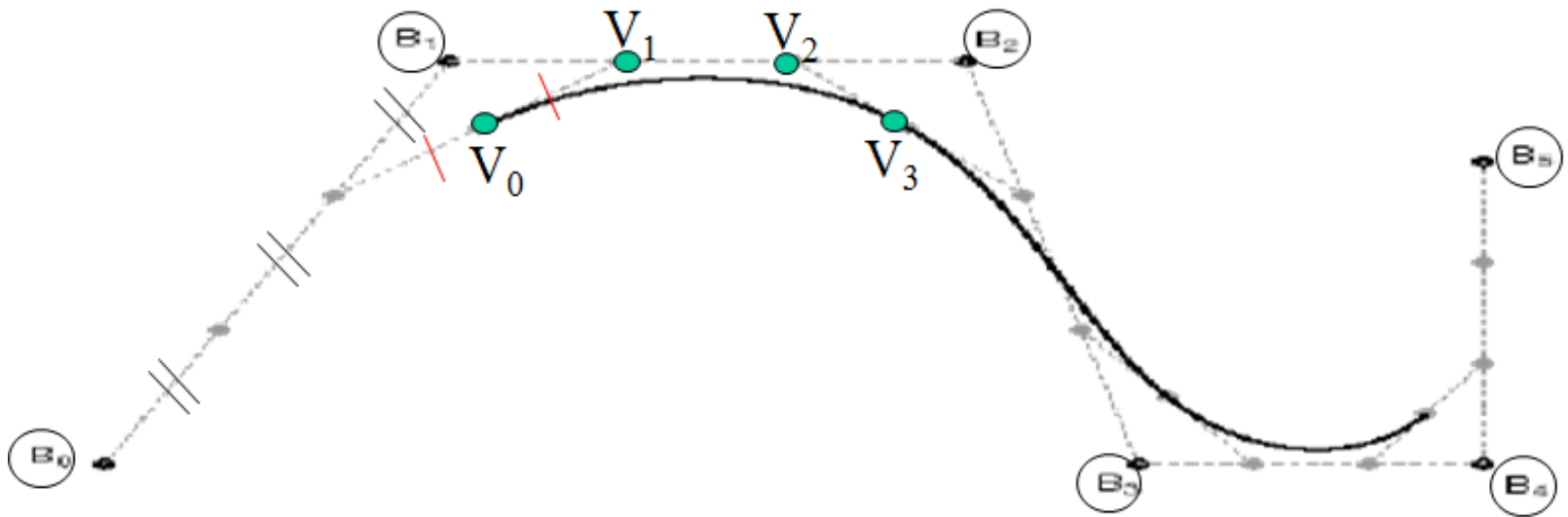


$$(W_1 - W_0) + (W_1 - W_2) = (V_2 - V_3) + (V_2 - V_1)$$



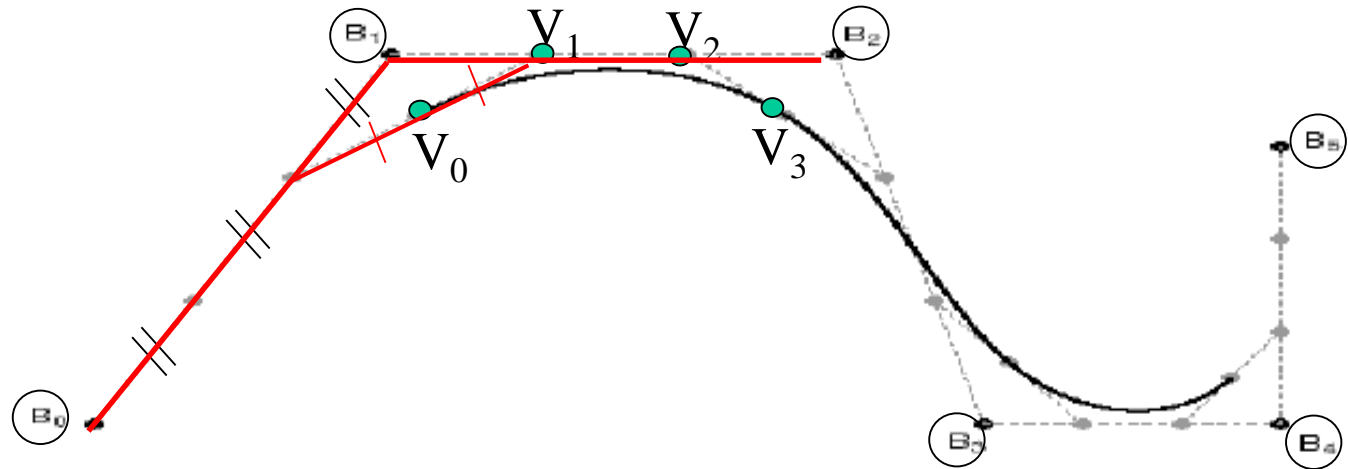
Constructing C^2 Bezier spline

- Given the **corner points** (i.e., \mathbf{B}_i) of the **A-frames**
- Let's build a C^2 continuous **approximating** spline



- The \mathbf{B}_i points are called **de Boor points**
- We will see how to define **B-splines** directly using these points later

Constructing C² Bezier spline



- Define the Bezier control points (V) in terms of the de Boor points (B):

$$V_1 = \frac{2}{3} B_1 + \frac{1}{3} B_2$$

$$V_2 = \frac{1}{3} B_1 + \frac{2}{3} B_2$$

$$\begin{aligned} V_0 &= \frac{1}{2} \left[\frac{1}{3} B_0 + \frac{2}{3} B_1 \right] + \frac{1}{2} \left[\frac{2}{3} B_1 + \frac{1}{3} B_2 \right] \\ &= \frac{1}{6} B_0 + \frac{4}{6} B_1 + \frac{1}{6} B_2 \end{aligned}$$

$$V_3 = \frac{1}{6} B_1 + \frac{4}{6} B_2 + \frac{1}{6} B_3$$

Constructing C² Bezier spline

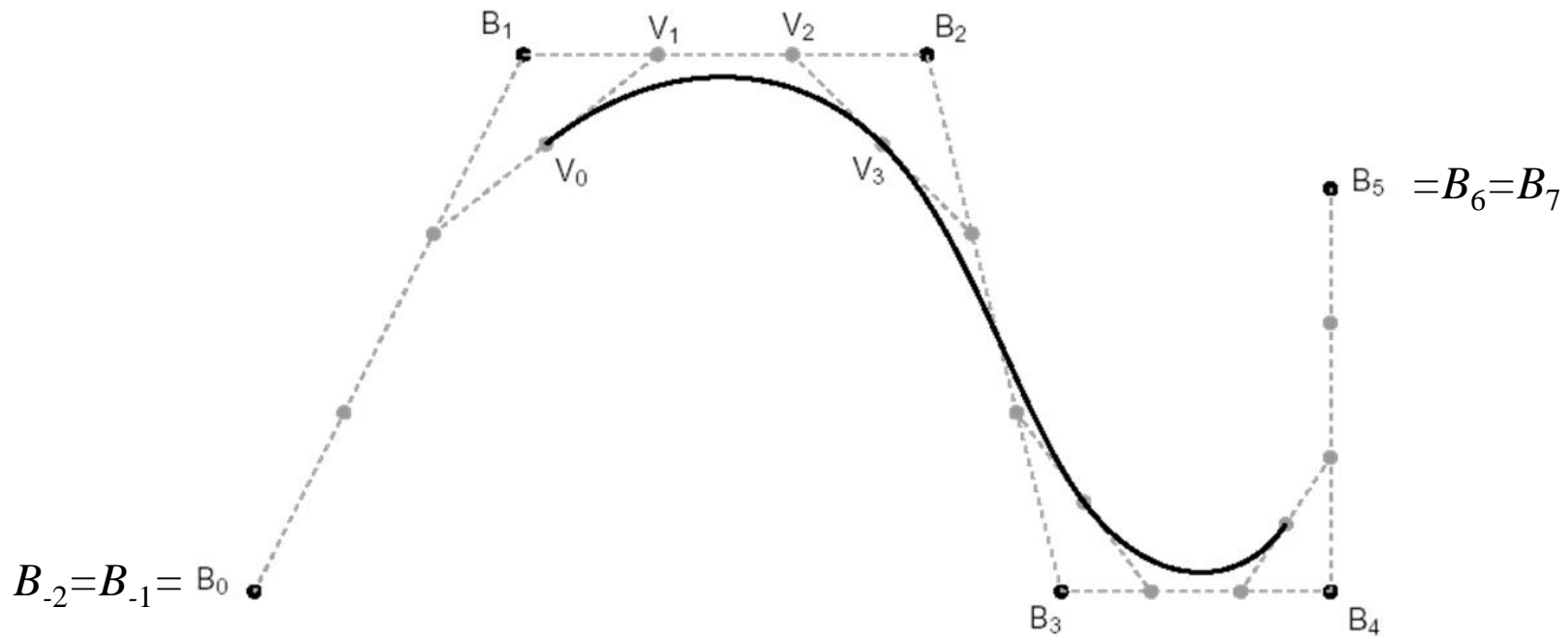
- Express the **Bezier points** in terms of the given **de Boor points**:

$$\begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 1/6 & 2/3 & 1/6 & 0 \\ 0 & 2/3 & 1/3 & 0 \\ 0 & 1/3 & 2/3 & 0 \\ 0 & 1/6 & 2/3 & 1/6 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

$$= \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

Endpoint interpolation

- How do we define the first and last curve segments?
- To get endpoint interpolation while still using the same matrix is to simply **repeat the endpoints** (*multiplicity = 3* for *cubics* to create *two curve segments* at each end)



Comparison

	Local control	interpolatory	C^2
C^1 interpolating Bezier spline	Y	Y	N
C^2 approximating Bezier spline	Y	N	Y
C^2-interpolating spline	N	Y	Y

Note that it is not possible to build a **C^2 continuous interpolating spline** using a *local* procedure. We will need to set up the C^2 continuous constraints for all the V_i points and solve the resulting system of linear equations.

<https://courses.cs.washington.edu/courses/csep557/10au/lectures/c2-interp.pdf>

Reparameterization

- We have so far consider **parametric continuity**, i.e. continuity of derivatives w.r.t. the parameter u
- This form of continuity makes sense particularly if we really are describing a particle moving over time and want its **motion** (e.g., velocity and acceleration) to be smooth.
- But, what if we're thinking only in terms of the shape of the curve? Is the parameterization actually **intrinsic to the shape**, i.e., is it the case that a shape has only one parameterization?



Piece
together 2
curves

$$\left\{ \begin{array}{l} Q_v(u) = \begin{bmatrix} 1-u \\ 0 \end{bmatrix} \\ Q_w(u) = \begin{bmatrix} 0 \\ u \end{bmatrix} \\ Q'_v(u) = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \\ Q'_w(u) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array} \right.$$

→
reparameterize

$$\left\{ \begin{array}{l} \tilde{Q}_v(u) = \begin{bmatrix} (1-u)^2 \\ 0 \end{bmatrix} \\ \tilde{Q}_w(u) = \begin{bmatrix} 0 \\ u^2 \end{bmatrix} \\ \tilde{Q}'_v(u) = \begin{bmatrix} 2(1-u) \\ 0 \end{bmatrix} \\ \tilde{Q}'_w(u) = \begin{bmatrix} 0 \\ 2u \end{bmatrix} \end{array} \right.$$

G^n (Geometric) Continuity

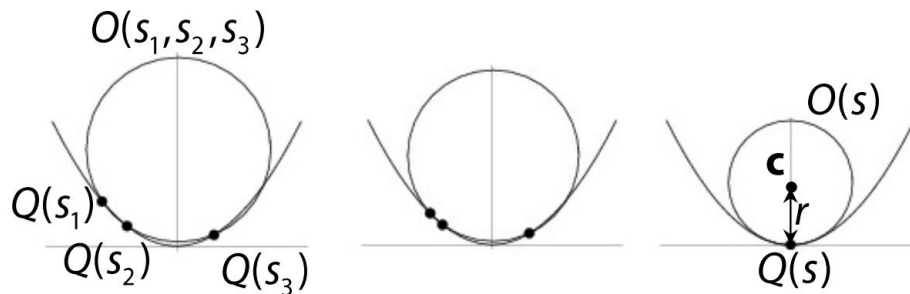
- Geometric continuity is defined in terms of **arc-length parametrization**:

$$Q(s) \text{ is } G^n \text{ continuous} \quad \text{iff}$$

$$Q^{(i)}(s) = \frac{d^i Q(s)}{ds^i} \text{ is continuous for } 0 \leq i \leq n$$

where $Q(s)$ is **parameterized by arc length** s .

- Then the **first derivative** (tangent) is of unit length
- And the **second derivative** points to center of the osculating circle



$$O(s) = \lim_{s_1, s_2, s_3 \rightarrow s} O(s_1, s_2, s_3)$$

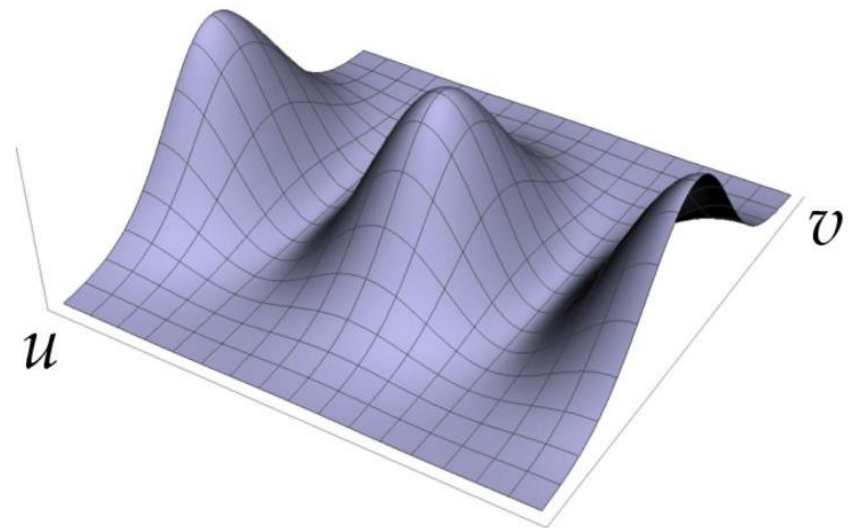
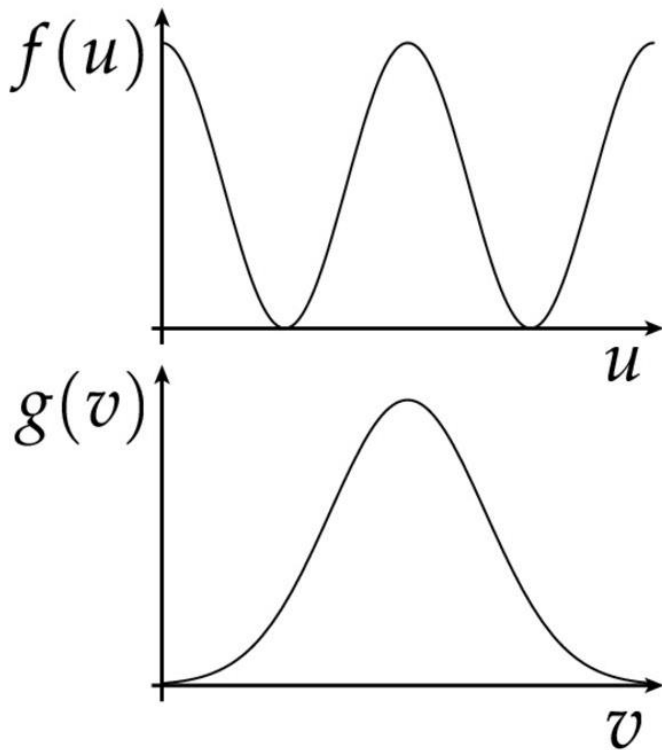
$$\|Q''(s)\| = \overset{\text{curvature}}{\kappa(s)} = \frac{1}{r(s)}$$

G^n continuity is a weaker constraint than C^n continuity

Parametric Surfaces

Tensor Product surfaces

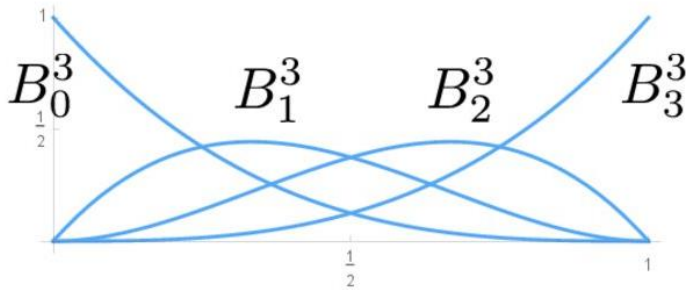
- Can use a pair of curves to get a surface
- Value at any point (u,v) is given by product of a curve f at u and a curve g at v (sometimes called the “**tensor product**”):



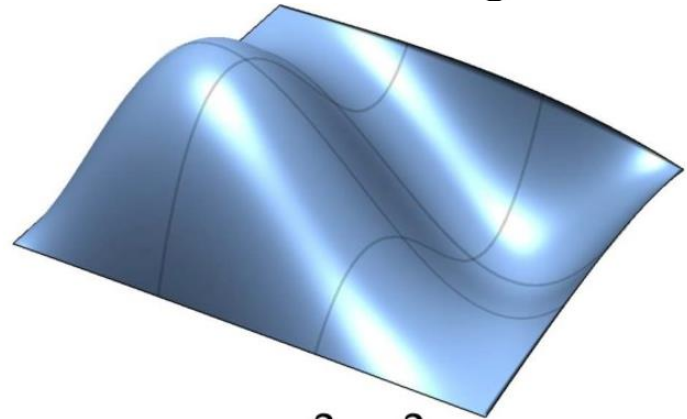
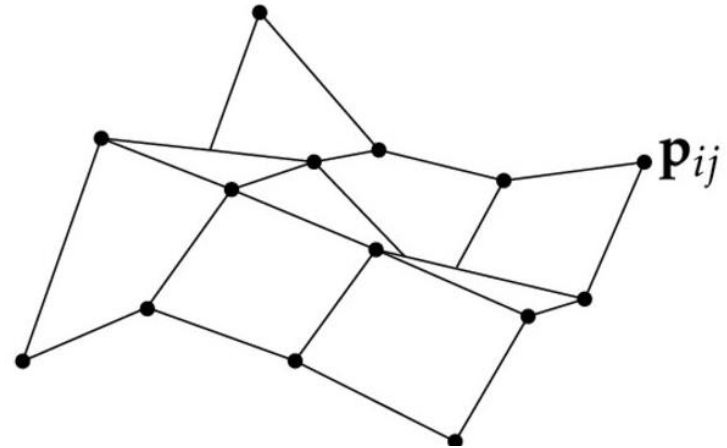
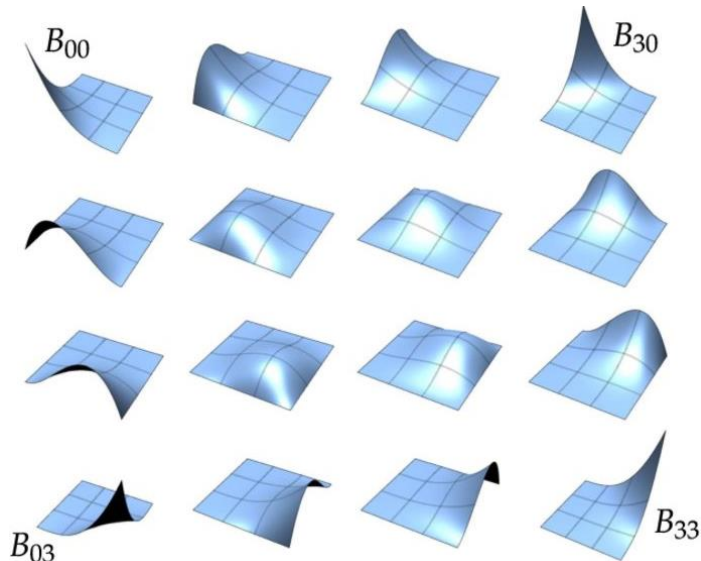
$$(f \otimes g)(u, v) := f(u)g(v)$$

Bezier Patches

- Bezier patch is sum of (tensor) products of Bernstein bases



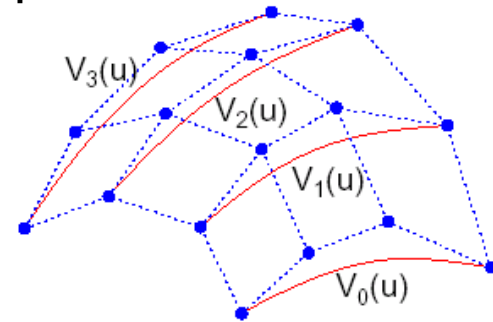
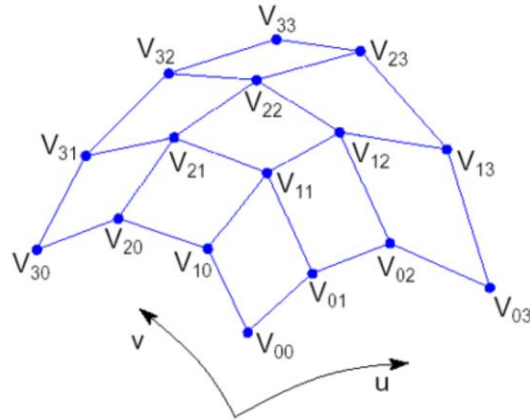
$$B_{i,j}^3(u, v) := B_i^3(u) B_j^3(v)$$



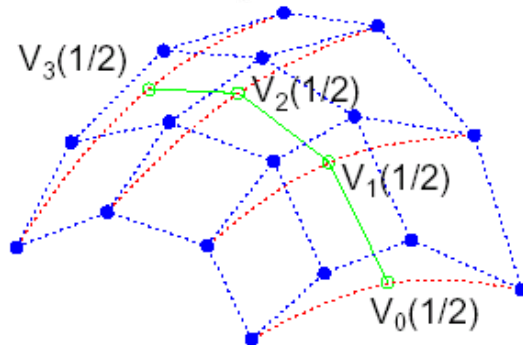
$$S(u, v) := \sum_{i=0}^3 \sum_{j=0}^3 B_{i,j}^3(u, v) \mathbf{p}_{ij}$$

Tensor product Bezier surfaces

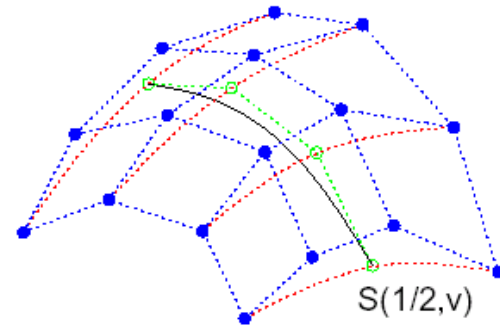
- Let's walk through the construction steps:



Control curves in u



Control polygon at $u=1/2$



Curve at $S(1/2, v)$

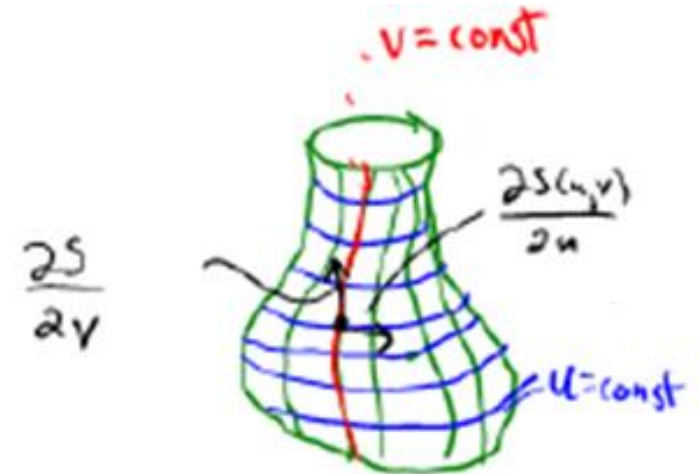
- The surface interpolates the four corner control points
- The boundary curves of a Bézier patch are themselves Bézier curves. E.g.,

$$S(0, v) = \sum_{j=0}^3 p_{0j} B_{j,3}(v)$$

Tangents and normal

- We can compute tangents to the surface at any point by looking at (infinitesimally) nearby points.
 - Holding one parameter constant and finding two nearby points by varying the other parameter. Thus, we can get **two tangents**:

$$\mathbf{t}_u = \frac{\partial S(u,v)}{\partial u} \quad \mathbf{t}_v = \frac{\partial S(u,v)}{\partial v}$$



- How do we compute the **normal**?
 - Take cross product of the two tangents

Bézier Patch

- Differentiation at the corners:

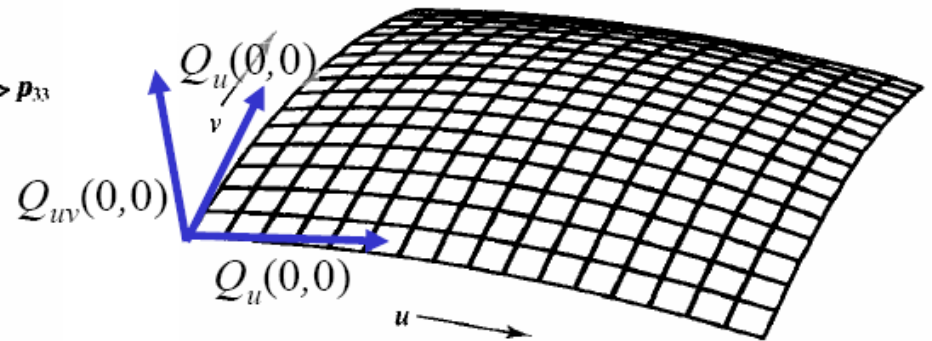
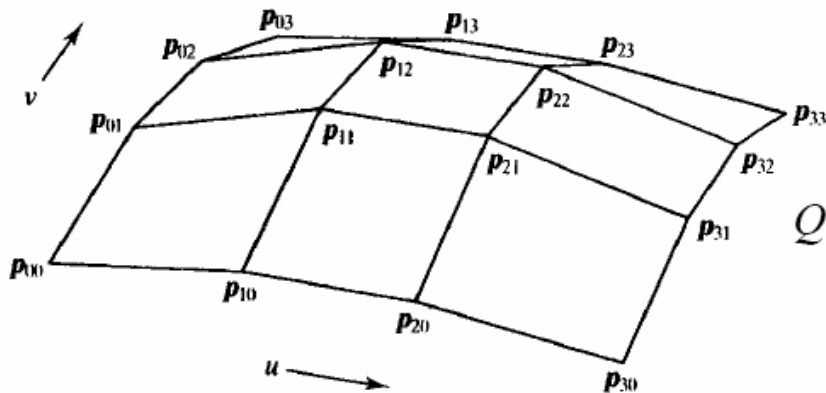
$$Q_u(0,0) = 3(p_{10} - p_{00})$$

$$Q_v(0,0) = 3(p_{01} - p_{00})$$

$$Q_{uv}(0,0) = 9(p_{00} - p_{01} - p_{10} + p_{11})$$

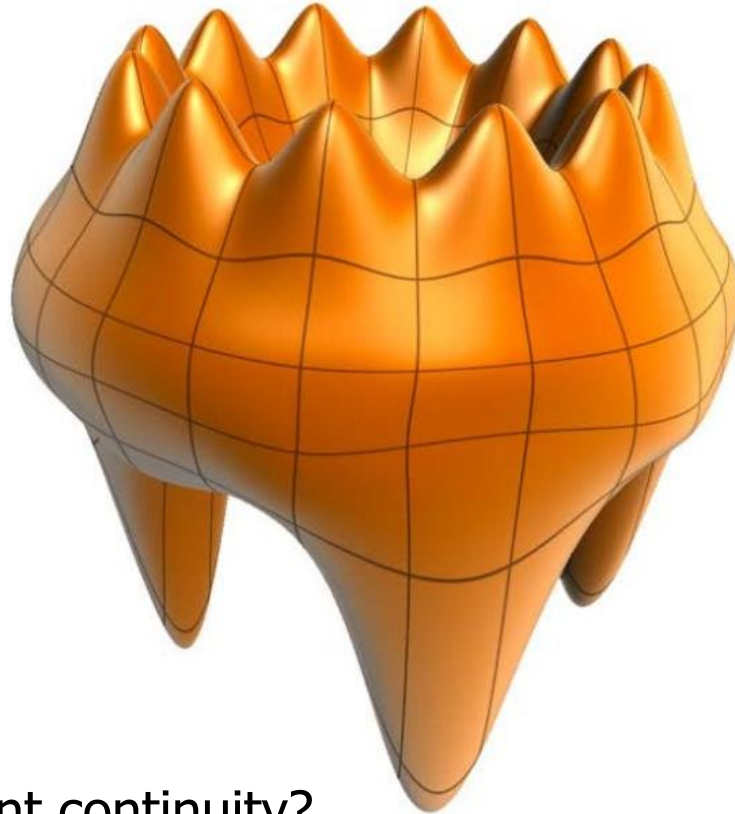
← tangent vectors

← twist vector



Bezier surface

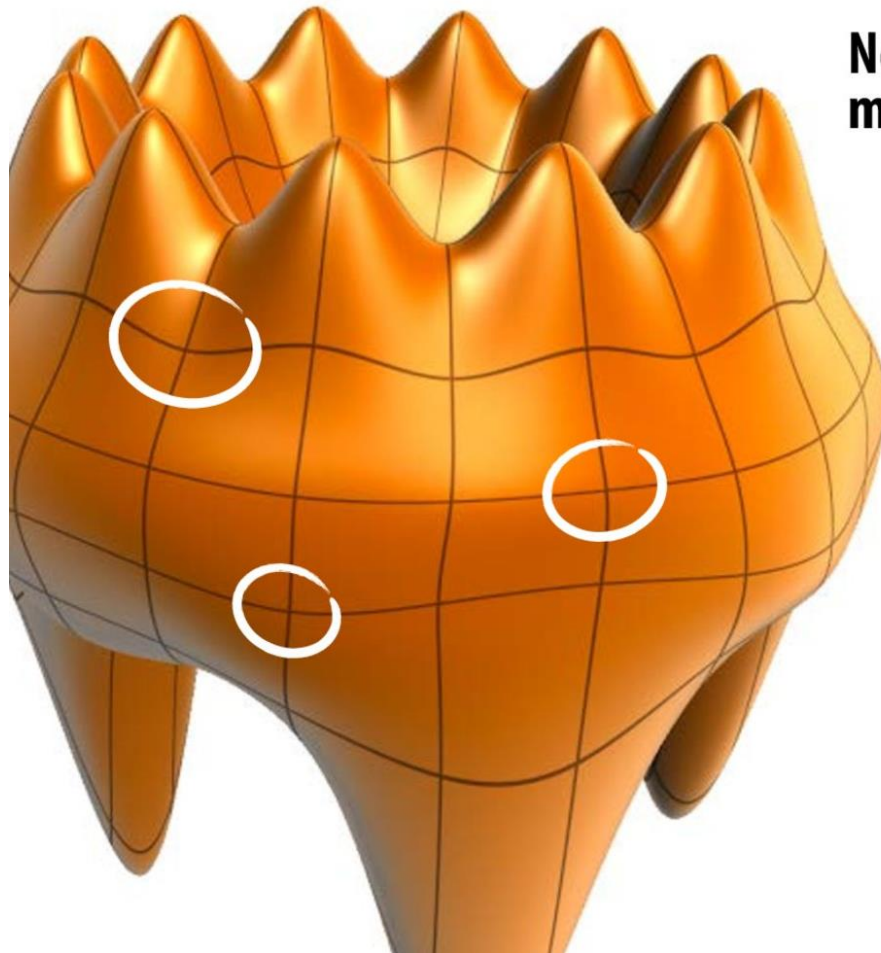
- Just as we connected Bezier *curves*, we can connect Bezier *patches* to get a surface:



- Can we always get tangent continuity?

no

Bezier Patches: limited connectivity



Notice that exactly four patches meet around *every* vertex!

In practice, far too constrained.

To make interesting shapes (with good continuity), we need patches that allow more interesting connectivity...

Other parametric schemes

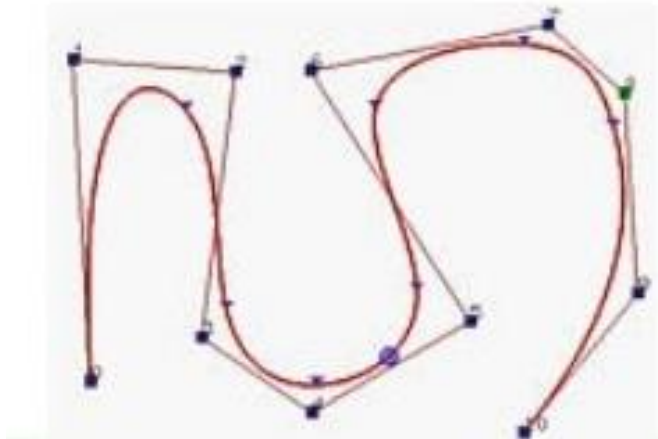
- There are many other parametric curve schemes
 - B-spline
 - Rational Bezier
 - NURBS
 - Hermite
 - Gregory
 - ...
- Can form corresponding tensor product surface schemes

Uniform Cubic B-spline

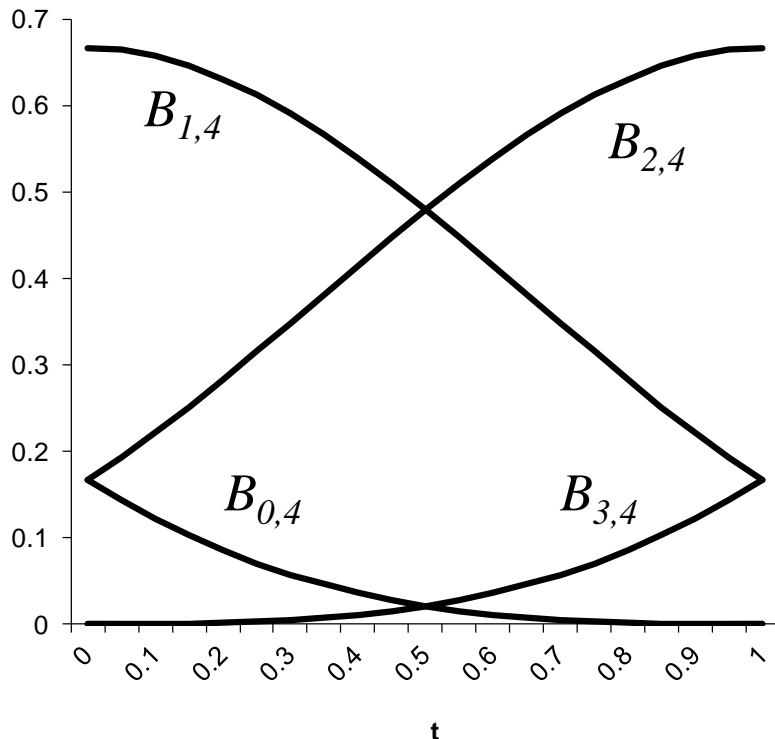
$$C(t) = \sum_{i=0}^3 P_i B_{i,4}(t)$$

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

uniform B-spline basis functions



Uniform cubic B-spline



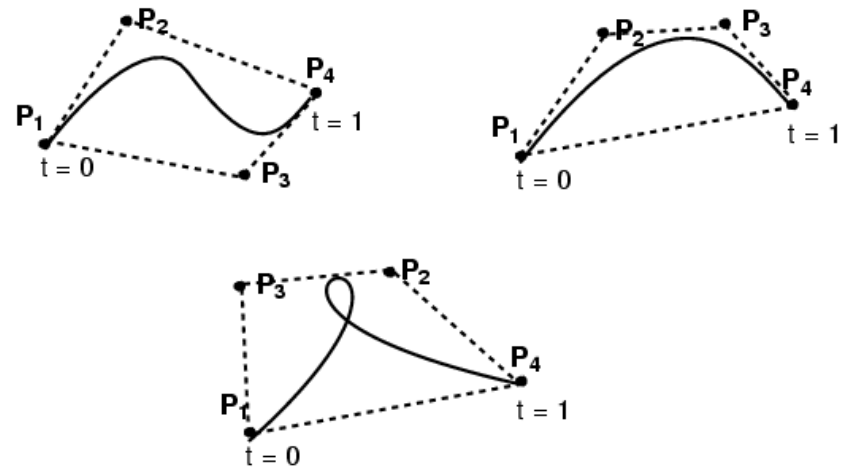
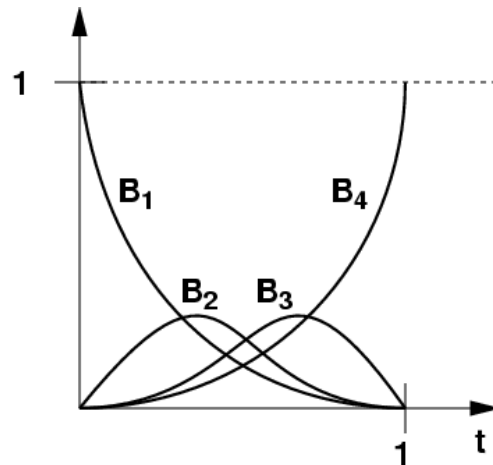
$$\begin{aligned}
 x(t) = & P_0 \frac{1}{6} (1 - 3t + 3t^2 - t^3) \\
 & + P_1 \frac{1}{6} (4 - 6t^2 + 3t^3) \\
 & + P_2 \frac{1}{6} (1 + 3t + 3t^2 - 3t^3) \\
 & + P_3 \frac{1}{6} (t^3)
 \end{aligned}$$

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

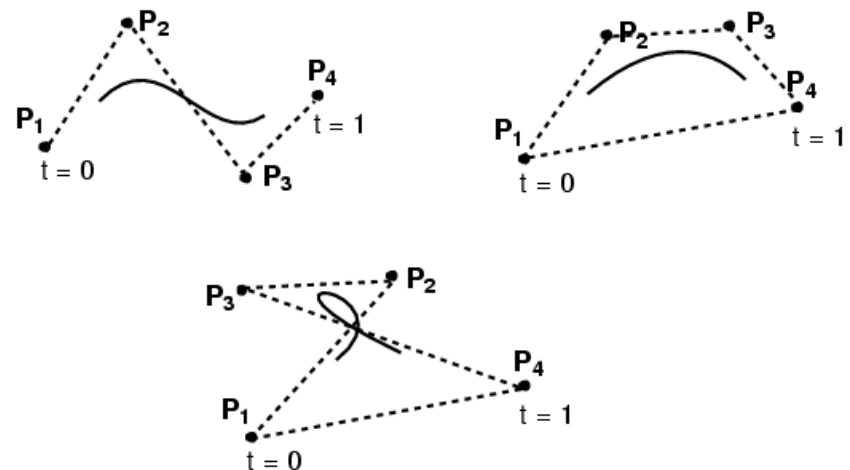
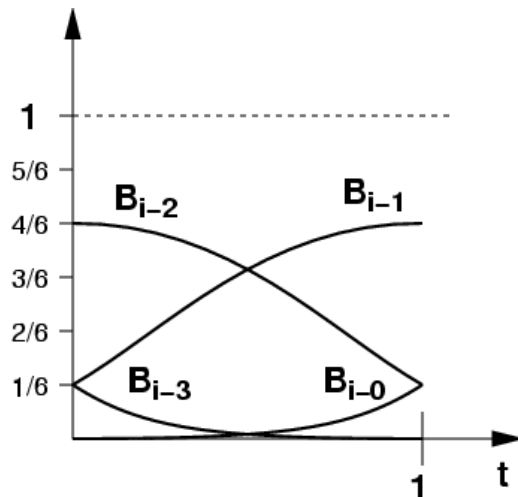
Bézier versus BSpline

- Relationship to the control points is different

Bézier



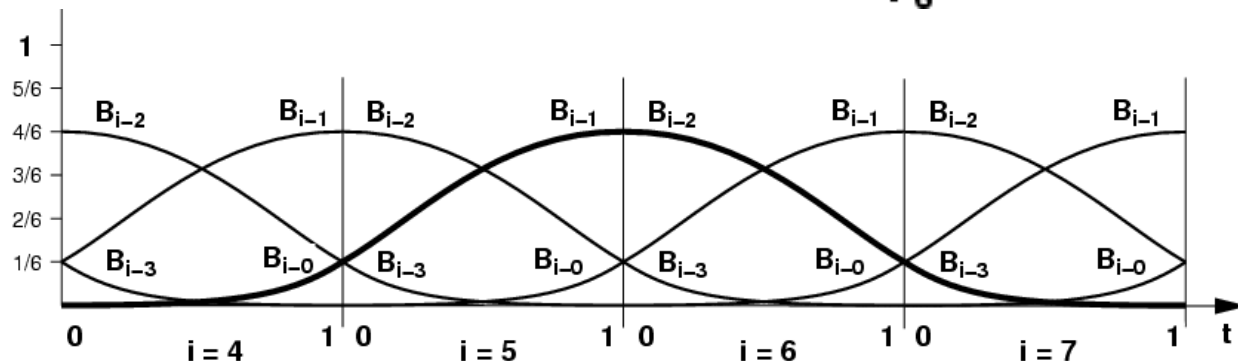
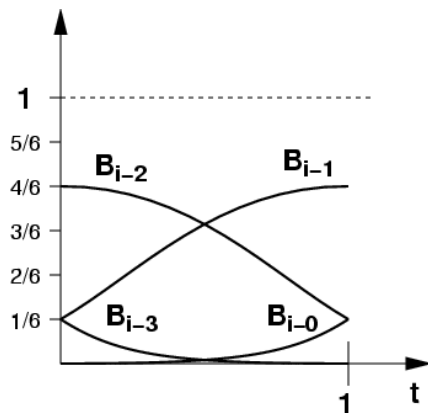
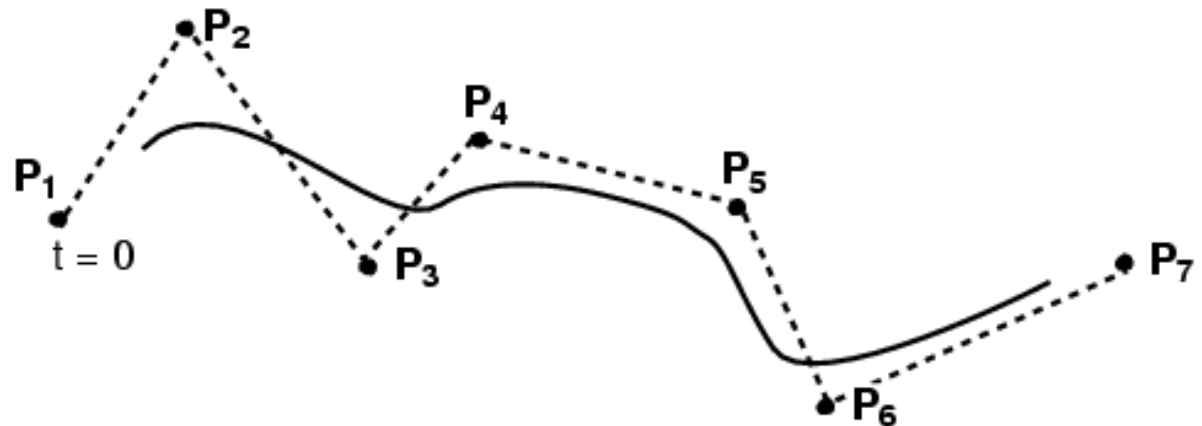
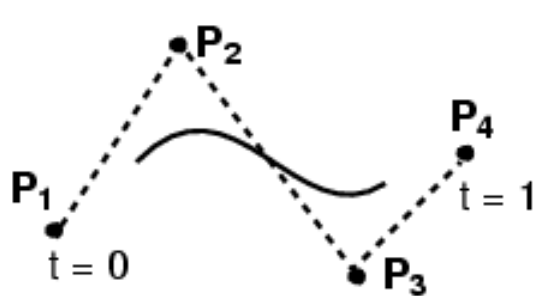
BSpline



Uniform Cubic B-Splines

- Basis functions can be chained together
- Local control

Applet



Basis functions are themselves C^2 continuous at knots

Rational Curves

- It is known from classical mathematics that all the **conic curves** (circle, ellipse, hyperbola, parabola) can be represented using **rational functions**, which is defined as **ratio of two polynomials**.
有理函数

$$x(u) = \frac{X(u)}{W(u)} \quad y(u) = \frac{Y(u)}{W(u)}$$

where $X(u)$, $Y(u)$ and $W(u)$ are polynomials.

Rational Curves

- Example 1:

Circle of radius 1, centered at the origin

$$x(u) = \frac{1 - u^2}{1 + u^2} \quad y(u) = \frac{2u}{1 + u^2}$$

- Easy to verify

$$\begin{aligned} (x(u))^2 + (y(u))^2 &= \left(\frac{1 - u^2}{1 + u^2} \right)^2 + \left(\frac{2u}{1 + u^2} \right)^2 \\ &= \frac{1 - 2u^2 + u^4 + 4u^2}{(1 + u^2)^2} = \frac{(1 + u^2)^2}{(1 + u^2)^2} = 1 \end{aligned}$$

- Example 2:

Ellipse, centered at the origin; the y -axis is the major axis, the x -axis is the minor axis, and the major and minor radii are 2 and 1, respectively

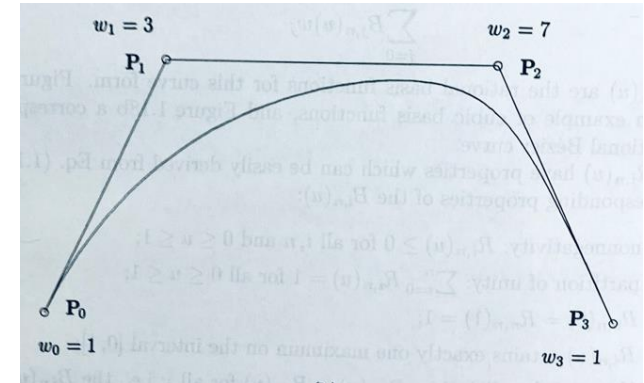
$$x(u) = \frac{1 - u^2}{1 + u^2} \quad y(u) = \frac{4u}{1 + u^2}$$

Rational Bezier

- An n-degree rational Bezier curve is defined as:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^n B_{i,n}(u) w_i \mathbf{P}_i}{\sum_{i=0}^n B_{i,n}(u) w_i} \quad 0 \leq u \leq 1$$

$w_i > 0$ are the weights controlling the 'strength' of control points. When $w_i = 1$ for all i , we get polynomial Bezier curves.

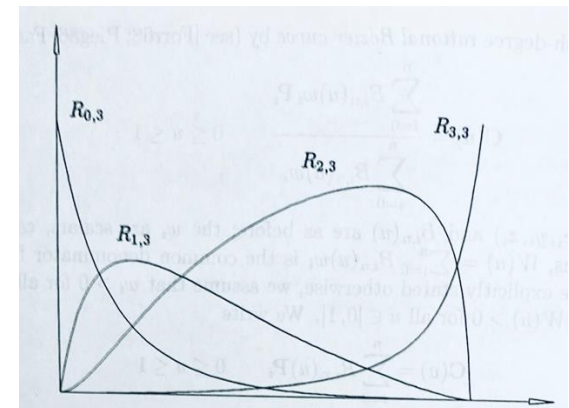


- Re-writing

$$\mathbf{C}(u) = \sum_{i=0}^n R_{i,n}(u) \mathbf{P}_i \quad 0 \leq u \leq 1$$

where

$$R_{i,n}(u) = \frac{B_{i,n}(u) w_i}{\sum_{j=0}^n B_{j,n}(u) w_j}$$



Rational Bezier

- Rational curves in n-dimensional space can be represented as a polynomial curve in (n+1)-dimensional space using homogeneous coordinates.

Now for a given set of control points, $\{\mathbf{P}_i\}$, and weights, $\{w_i\}$, construct the weighted control points, $\mathbf{P}_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$. Then define the *nonrational* (polynomial) Bézier curve in four-dimensional space

$$\mathbf{C}^w(u) = \sum_{i=0}^n B_{i,n}(u) \mathbf{P}_i^w$$

Writing out the coordinate functions:

$$\begin{aligned} X(u) &= \sum_{i=0}^n B_{i,n}(u) w_i x_i & Y(u) &= \sum_{i=0}^n B_{i,n}(u) w_i y_i \\ Z(u) &= \sum_{i=0}^n B_{i,n}(u) w_i z_i & W(u) &= \sum_{i=0}^n B_{i,n}(u) w_i \end{aligned}$$

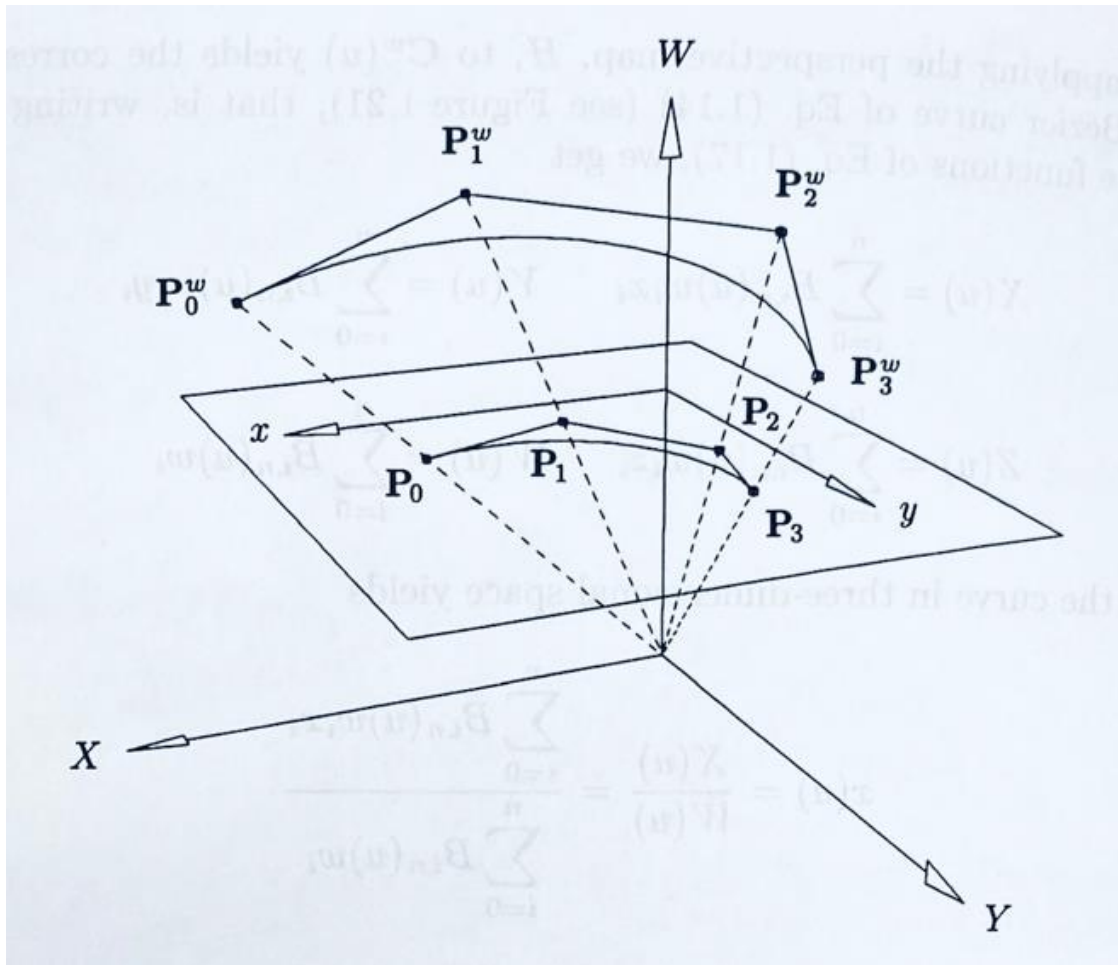
Locating the curve in three-dimensional space yields

$$x(u) = \frac{X(u)}{W(u)} = \frac{\sum_{i=0}^n B_{i,n}(u) w_i x_i}{\sum_{i=0}^n B_{i,n}(u) w_i}$$

$$y(u) = \frac{Y(u)}{W(u)} = \frac{\sum_{i=0}^n B_{i,n}(u) w_i y_i}{\sum_{i=0}^n B_{i,n}(u) w_i}$$

$$z(u) = \frac{Z(u)}{W(u)} = \frac{\sum_{i=0}^n B_{i,n}(u) w_i z_i}{\sum_{i=0}^n B_{i,n}(u) w_i}$$

Rational Bezier



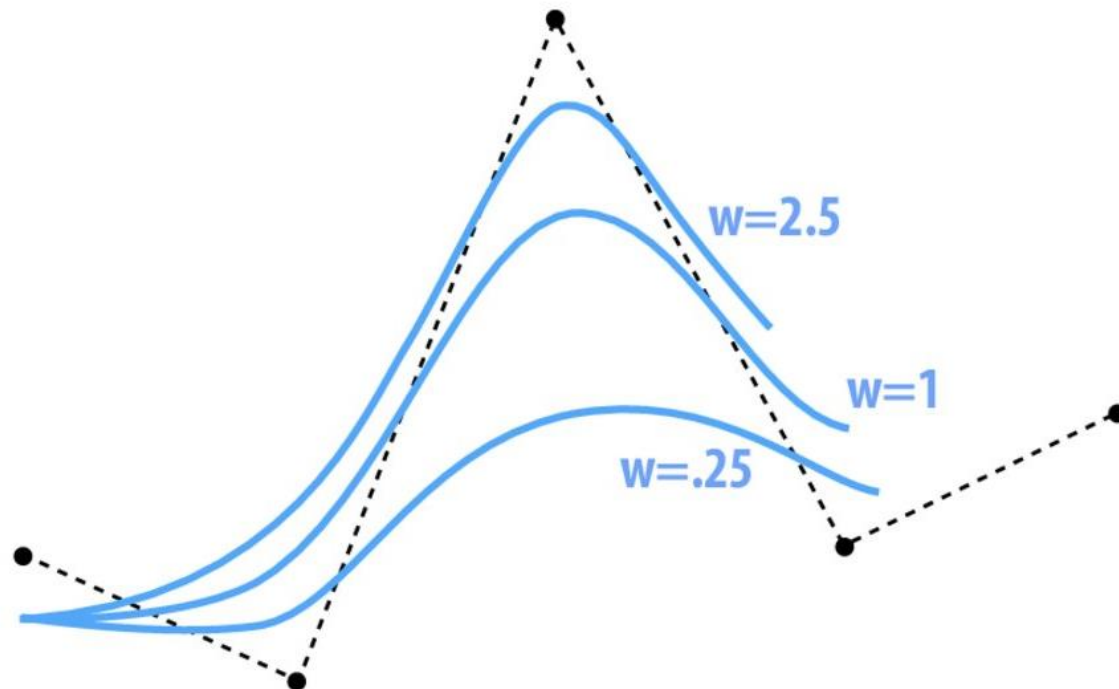
A geometric construction of a rational Bezier curve

Properties of Bezier rational curves

- The rational Bezier basis functions have all the following properties like the polynomial Bezier basis functions:
 - Nonnegativity
 - Partition of unity
 - $R_{0,n}(0) = R_{n,n}(1)=1,$
 - ...
- Therefore rational Bezier curves have all the following properties
 - Endpoint interpolation
 - Convex hull property
 - Transformation invariance
 - Variation diminishing
 - The k th derivative at $u=0$ ($u=1$) depends on the first (last) $k+1$ control points and weights

NURBS

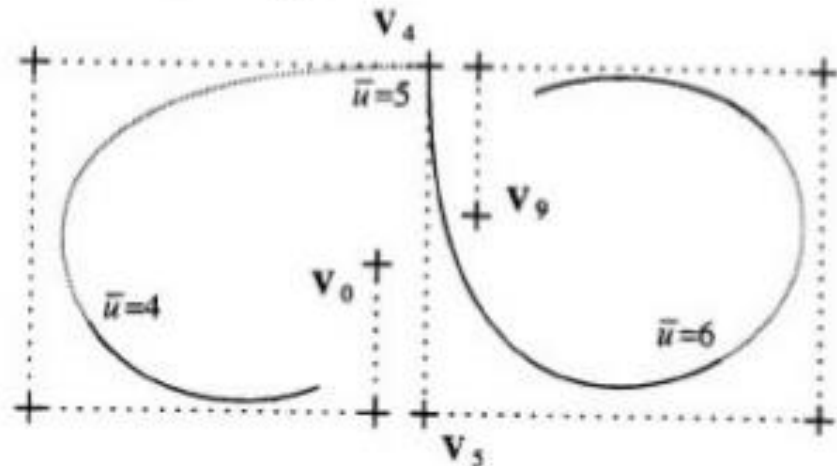
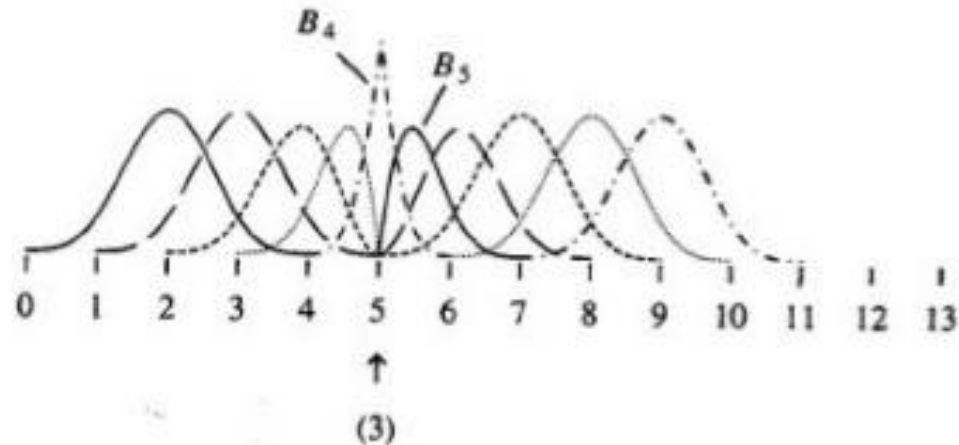
- (N)on-(U)niform (R)ational (B)-(S)pline
 - Knots at arbitrary locations (non-uniform)
 - Ratio of polynomials (rational)
 - Piecewise B-spline curve



NURBS

Non-uniform knots: different spacing between the basis functions

- Knot vector = $\{0, 1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9, 10, 11\}$
- Triple knots at 5
- 5 segments



Matrix representation

$$\begin{aligned}
 Q(u) &= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\
 &= (-V_0 + 3V_1 - 3V_2 + V_3)u^3 + (3V_0 - 6V_1 + 3V_2)u^2 + \\
 &\quad (-3V_0 + 3V_1)u + V_3
 \end{aligned}$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Bezier basis matrix

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

B-spline basis matrix

Matrix Representation

$$Q(u) = UMP$$

where $U = [u^3 \ u^2 \ u^1 \ 1]$

$$Q'(u) = [3u^2 \ 2u \ 1 \ 0]MP$$

Pros:

- Compact representation
- Convenient implementation in either hardware or software with available matrix facilities

Conversion between representations

$$UM_i P_i = UM_j P_j$$

$$M_i P_i = M_j P_j$$

$$P_i = M_i^{-1} M_j P_j$$

where M_* is the **characteristic matrix** of a particular curve scheme

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

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

E.g. Convert a cubic uniform B-spline curve to Bézier curve:

$$P_B = M_B^{-1} M_{BS} P_{BS}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} M_{BS} P_{BS} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} P_{BS}$$

双三次的 Bicubic Surfaces

- Matrix equation of a bicubic surface:

$$Q(u, v) = UMPM^T V^T$$

where

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$
$$V = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

P is a 4×4 matrix containing the 16 control points

Bézier Patch

$$Q(u, v) = UMPM^T V^T$$

- Differentiating

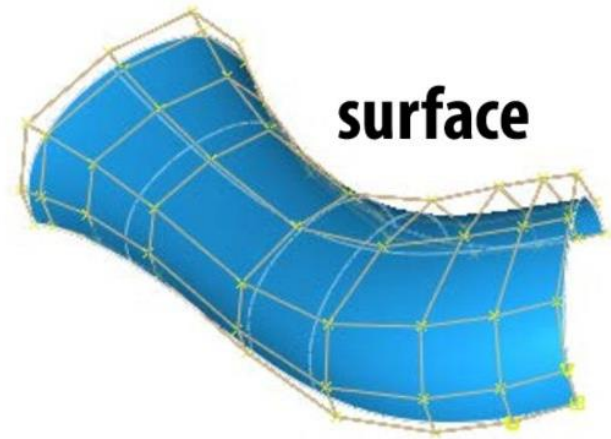
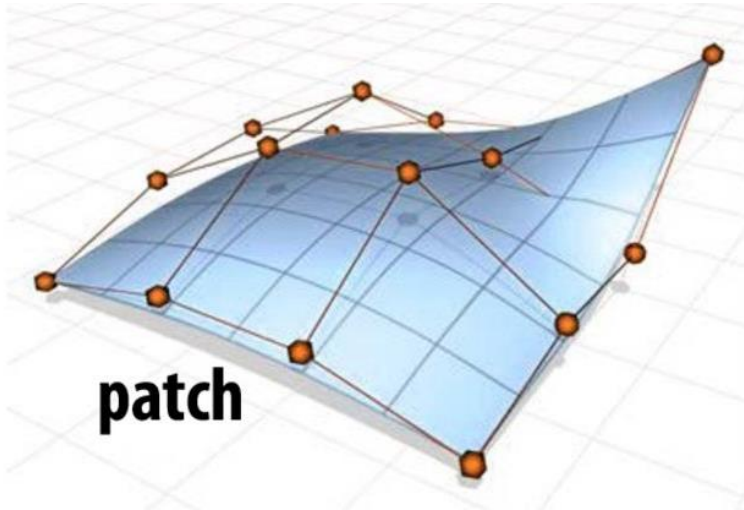
$$Q_u(u, v) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} M_B P M_B^T V^T$$

$$Q_v(u, v) = U M_B P M_B^T \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

$$Q_{uv}(u, v) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} M_B P M_B^T \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

NURBS surface

- Multiple NURBS patches form a surface



- Pros: easy to evaluate, exact conics, high degree of continuity
- Cons: rectangular underlying topology; difficult to maintain continuity for arbitrary topology