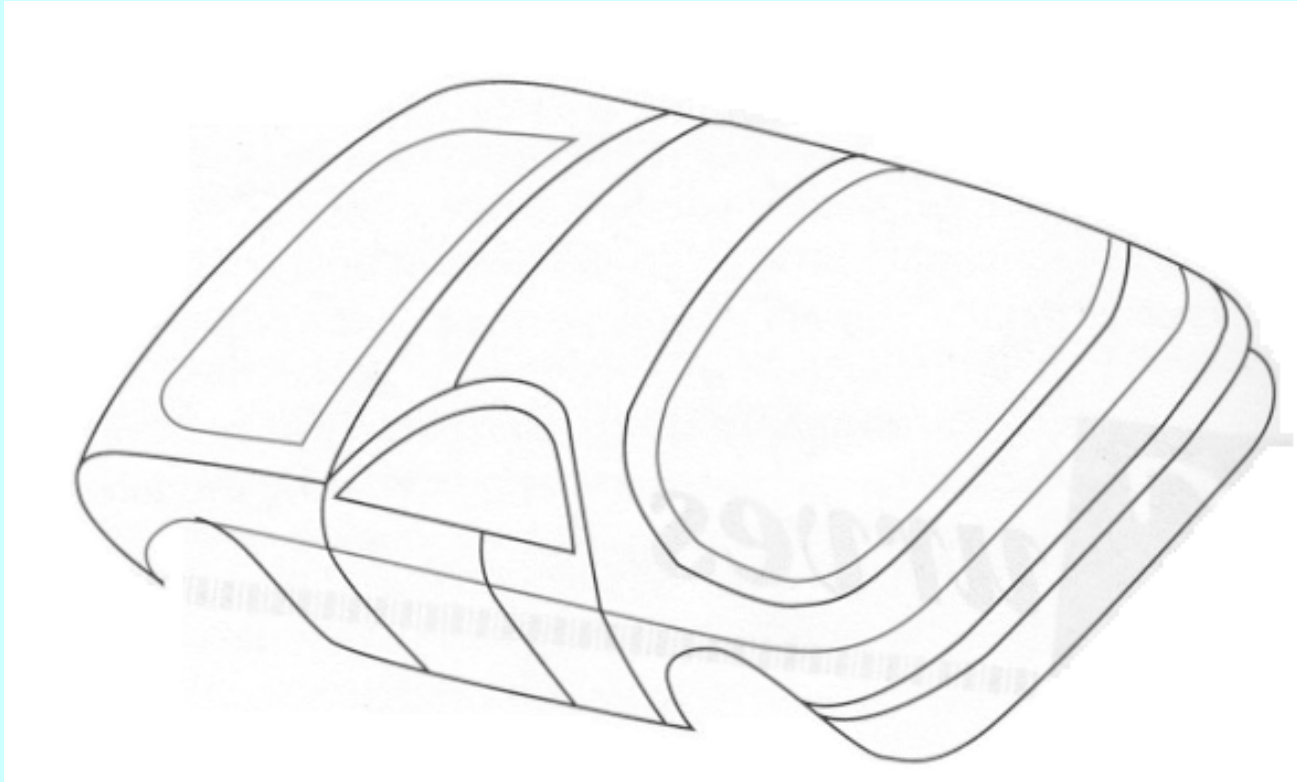


Curve and Surface Design



Examples of free form curves in the design of automobile bodies

Modeling

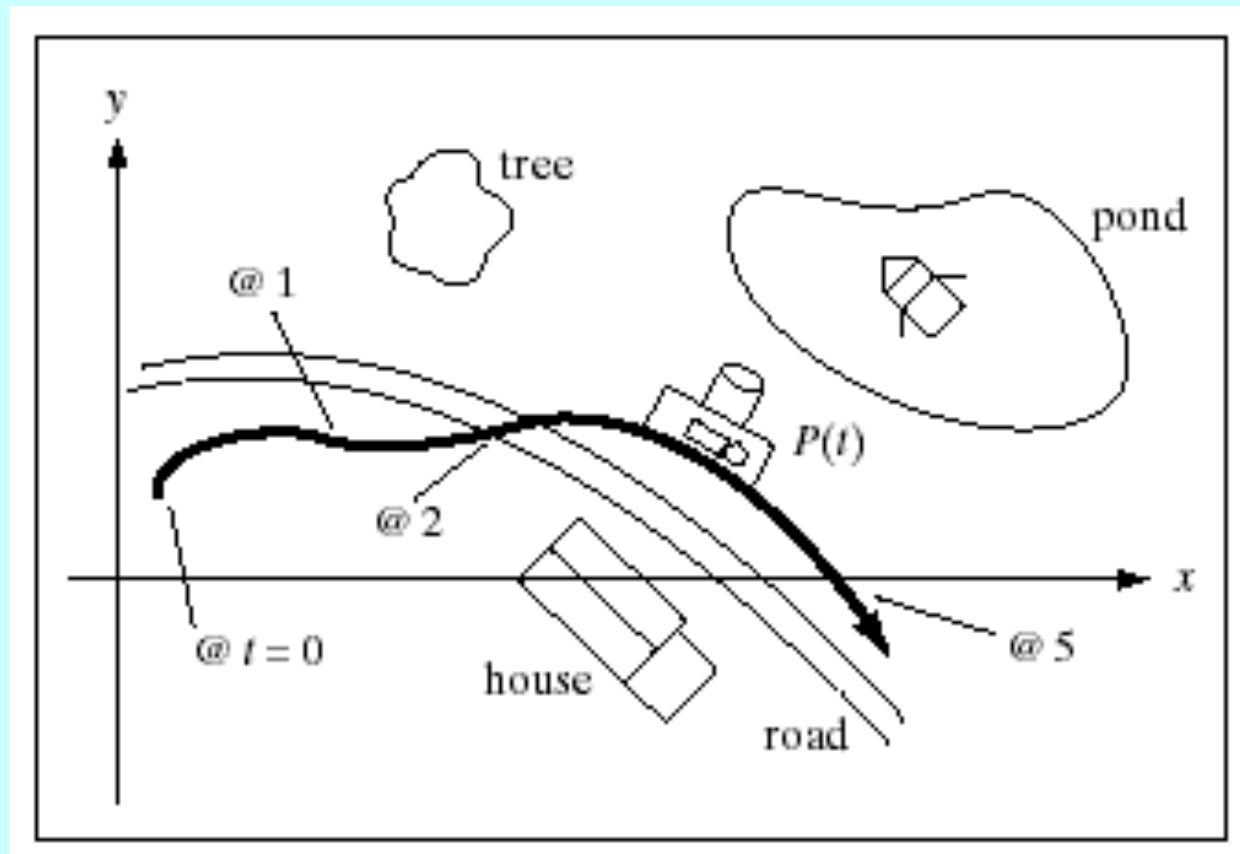
- polygons
- constructive solid geometry
- parametric surfaces
- implicit surfaces
- subdivision surfaces
- particle systems
- volumes

Curves: Motivation

- Key for modeling in computer graphics
- Interpolation of data
- Later in course, surfaces are (simple) extension
- Modeling of shape

Parametric Curves as Trajectories

An important application of representing curves parametrically:
describing the path that an object takes as it moves in time.

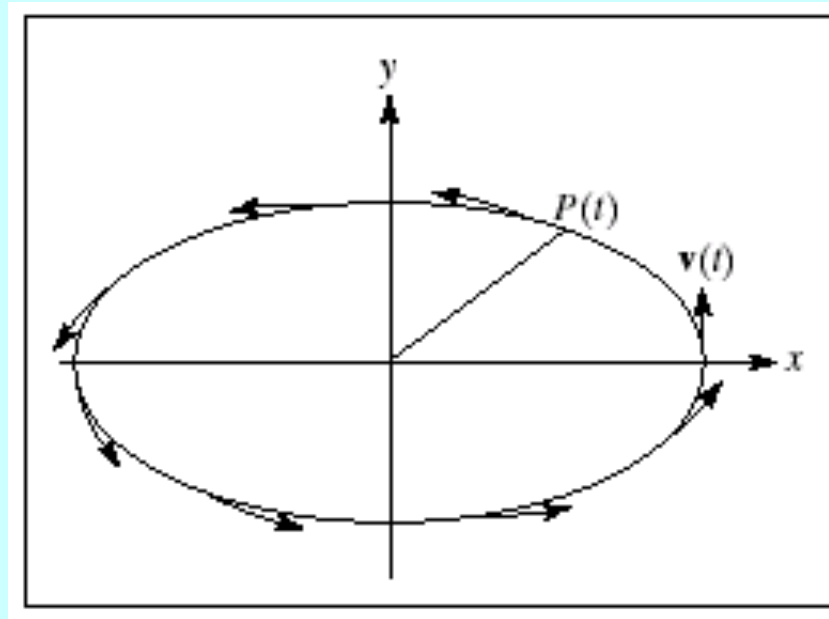


One important feature of curves is *independence of the coordinate axes*.

We don't want the curve to change shape when the coordinate axes
(or the points defining the curve) are rigidly moved or rotated.

Smoothness of motion

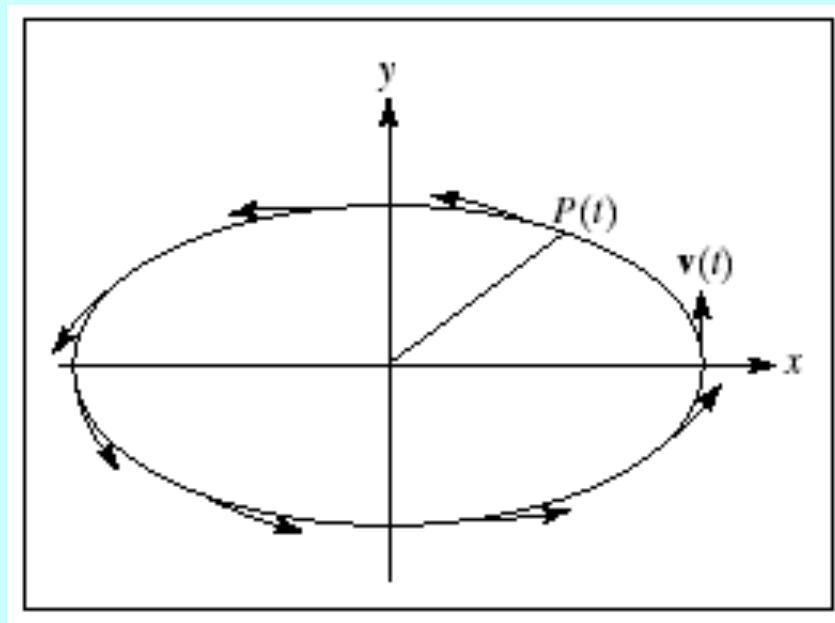
What is the velocity of $P(t)$ along the curve as t increases?



Does the object move jerkily or smoothly along its trajectory?

The **velocity** $\mathbf{v}(t)$ is a vector that describes the speed and direction of $P(t)$ as it traverses the curve.

$$\mathbf{v}(t) = \frac{dP(t)}{dt} = \left(\frac{dx(t)}{dt}, \frac{dy(t)}{dt} \right)$$



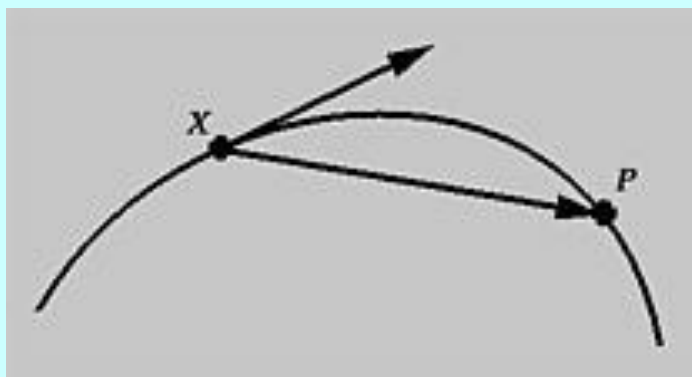
$$P(t) = (W \cos(t), H \sin(t))$$

$$\mathbf{v}(t) = (-W \sin(t), H \cos(t))$$

The magnitude of $\mathbf{v}(t)$ is the **speed** at t .

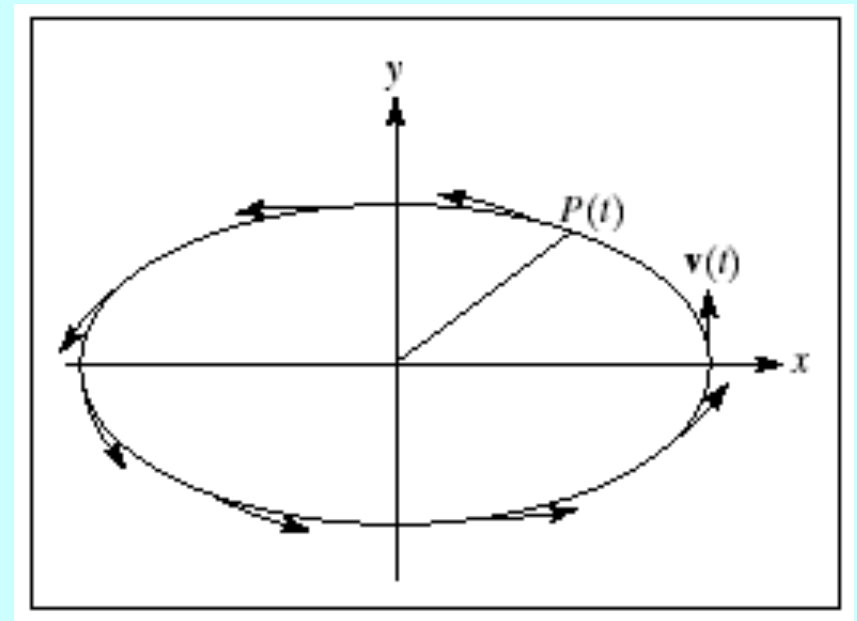
Consider a fixed point X and a moving point P on a curve.

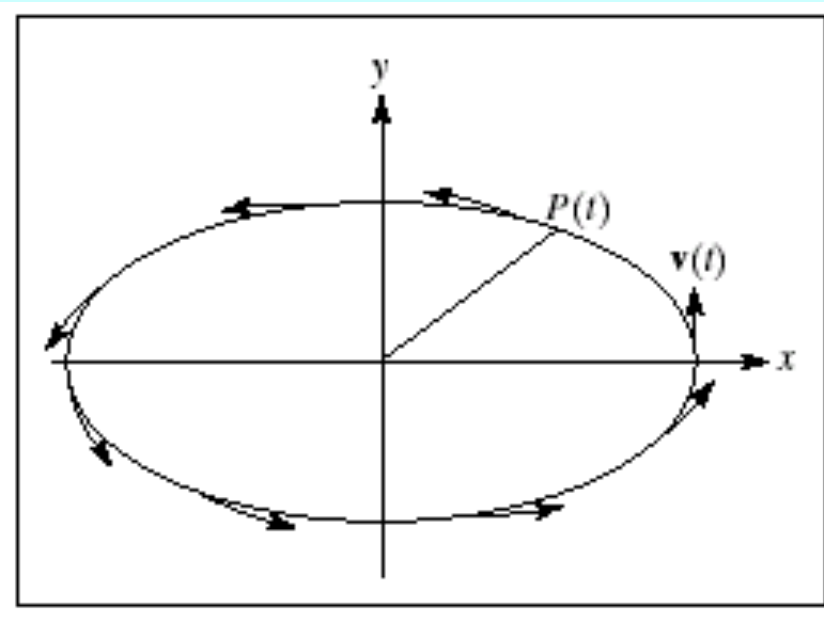
As point P moves toward X , the vector from X to P approaches the tangent vector at X . The line that contains the tangent vector is the **tangent line**.



The **tangent line** to the curve $P(t)$ at $t=t_0$:

$$L(u) = P(t_0) + \mathbf{v}(t_0)u$$

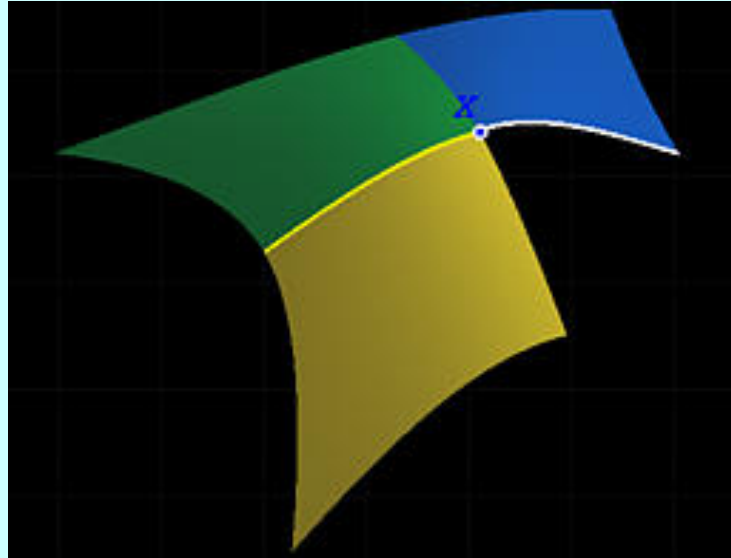




The **normal direction** to the curve $P(t)$ at $t=t_0$:

$$\mathbf{n}(t_0) = \mathbf{v}_\perp(t_0) = \left(-\frac{dy}{dt}, \frac{dx}{dt} \right) \Big|_{t=t_0}$$

Curve continuity



How we can make sure that these curves join together in a "**smooth**" way?

Curve continuity

- Parametric continuity
- Geometric continuity

Parametric continuity (C^k continuity or “ k -smoothness”)

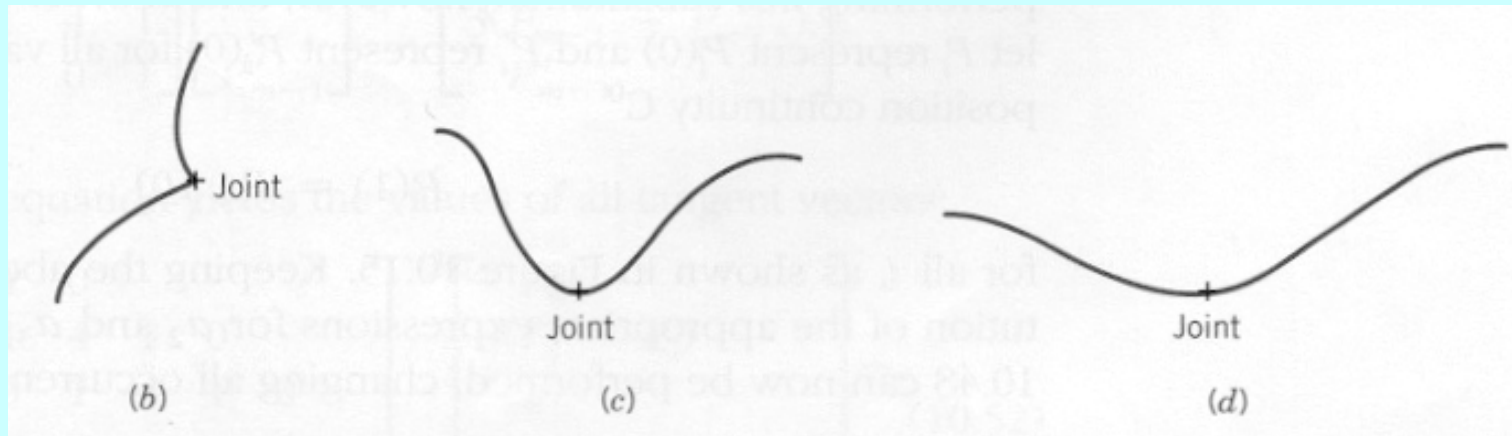
We say that a curve $P(t)$ has k th-order parametric continuity **everywhere** in the t -interval $[a,b]$ if **all derivatives of the curve up through the k th, exist and are continuous at all points inside $[a,b]$.**

Two curves are C^1 continuous at the joining point if the **speed** (*i.e.*, first derivative) does not change when crossing one curve to the other.

Similarly, two curves are C^2 continuous at the joining point if, in addition to the speed, the **acceleration** (*i.e.*, second derivative) is also the same when crossing one curve to the other.

C^1 continuous is "smoother" than C^0 continuous at the joining point,
 C^2 continuous is "smoother" than C^1 continuous at the joining point, ...

Curve continuity



C^0 continuity

C^1 continuity

C^2 continuity

Curve continuity

- **Positional continuity (C0)**
holds whenever the end positions of two curves or surfaces are coincidental. The curves or surfaces may still meet at an angle, giving rise to a sharp corner or edge and causing broken highlights.
- **Tangential continuity (C1)**
requires the end vectors of the curves or surfaces to be parallel, ruling out sharp edges. Because highlights falling on a tangentially continuous edge are always continuous and thus look natural, this level of continuity can often be sufficient.
- **Curvature continuity (C2)**
further requires the end vectors to be of the same magnitude. Highlights falling on a curvature-continuous edge do not display any change, causing the two surfaces to appear as one. This can be visually recognized as “perfectly smooth”. This level of continuity is very useful in the creation of models that require many bi-cubic patches comprising one continuous surface.

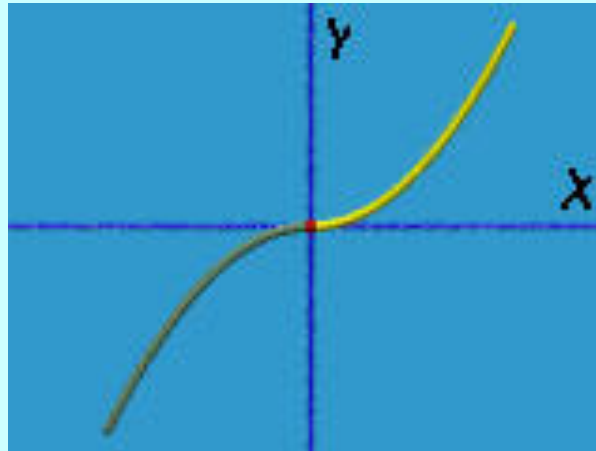
Example: The following curve consists of two parabolas:

$$\mathbf{f}(u) = (u, -u^2, 0)$$

$$\mathbf{g}(v) = (v, v^2, 0)$$

where $\mathbf{f}()$, the left curve, and $\mathbf{g}()$, the right curve, have domains $[-1,0]$ and $[0,1]$, respectively.

Are these two curves C^2 continuous there?



$$\mathbf{f}'(u) = (1, -2u, 0)$$

$$\mathbf{f}''(u) = (0, -2, 0)$$

$$\mathbf{g}'(v) = (1, 2v, 0)$$

$$\mathbf{g}''(v) = (0, 2, 0)$$

Since $\mathbf{f}'(0) = \mathbf{g}'(0) = (1, 0, 0)$, these two curves are C^1 continuous at the origin. However, since $\mathbf{f}''(0) = (0, -2, 0) \neq \mathbf{g}''(0) = (0, 2, 0)$, they are not C^2 continuous at the origin.

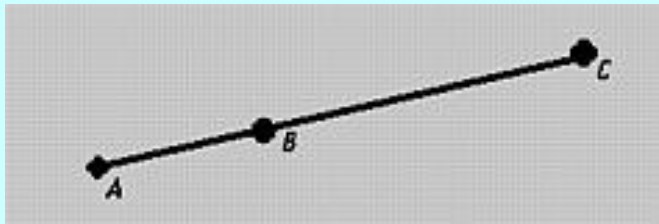
Problems with Parametric Representations

C^k continuity looks a promising way of specifying how smooth two curves are joint together. However, it does have a problem.

Consider the following two line segments:

$$\mathbf{f}(u) = \mathbf{A} + u(\mathbf{B} - \mathbf{A})$$

$$\mathbf{g}(v) = \mathbf{B} + v(\mathbf{C} - \mathbf{B})$$

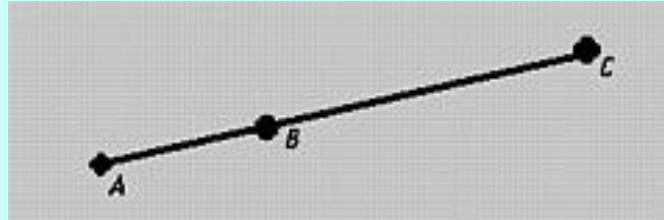


Line segments $\mathbf{f}(u)$ and $\mathbf{g}(v)$ are obviously C^0 continuous at the joining point \mathbf{B} . Is it C^1 continuous?

Problems with Parametric Representations

$$\mathbf{f}(u) = \mathbf{A} + u(\mathbf{B} - \mathbf{A})$$

$$\mathbf{g}(v) = \mathbf{B} + v(\mathbf{C} - \mathbf{B})$$



$$\mathbf{f}'(u) = \mathbf{B} - \mathbf{A}$$

$$\mathbf{g}'(v) = \mathbf{C} - \mathbf{B}$$

Thus, $\mathbf{f}'(u) = \mathbf{B} - \mathbf{A}$ is, in general, not equal to $\mathbf{g}'(v) = \mathbf{C} - \mathbf{B}$



these two line segments are not C^1 continuous at the joining point **B**!

If we replace the direction vectors $\mathbf{B} - \mathbf{A}$ and $\mathbf{C} - \mathbf{B}$ with unit-length vectors and change the domain of parameters u and v , this problem will disappear.

$$\mathbf{F}(u) = \mathbf{A} + u(\mathbf{B} - \mathbf{A}) / |\mathbf{B} - \mathbf{A}|$$

$$\mathbf{G}(v) = \mathbf{B} + v(\mathbf{C} - \mathbf{B}) / |\mathbf{C} - \mathbf{B}|$$

where u is in the range of $(0, |\mathbf{B} - \mathbf{A}|)$ and v is in the range of $(0, |\mathbf{C} - \mathbf{B}|)$.

Now, since we have $\mathbf{F}'(u) = \mathbf{G}'(v) =$ the unit-length vector in the direction of \mathbf{A} to \mathbf{C} , the line segments are C^1 continuous.

Example: Consider the parametric curve:

$$\mathbf{P}(t) = (1-t)^3 + \mathbf{P}_0 + t^3 \mathbf{P}_1 = \left((1-t)^3 x_0 + t^3 x_1, (1-t)^3 y_0 + t^3 y_1 \right)$$

$$\mathbf{P}(0) = \mathbf{P}_0$$

$$\mathbf{P}(1) = \mathbf{P}_1$$

$$\left(\frac{dx}{dt}, \frac{dy}{dt} \right) = \left(-3(1-t)^2 x_0 + 3t^2 x_1, -3(1-t)^2 y_0 + 3t^2 y_1 \right)$$

Start with 2 points: $\mathbf{P}_0 = (0,0)$ $\mathbf{P}_1 = (5,6)$

$$\frac{dy}{dx} = \frac{dy}{dt} / \frac{dx}{dt} = \frac{-3(1-t)^2 0 + 3t^2 * 6}{-3(1-t)^2 0 + 3t^2 * 5} = \frac{6}{5} = \text{constant}$$

So the curve is a straight line

Translate both points by the same amount (0,-1):

$$\mathbf{P}_0 = (0, -1) \quad \mathbf{P}_1 = (5, 5)$$

$$\frac{dy}{dx} = \frac{dy}{dt} / \frac{dx}{dt} = \frac{-3(1-t)^2 + 15t^2}{15t^2} = \frac{1}{5} \left(\frac{1}{t} - 1 \right) + 1$$

The curve is no longer a straight line!!
The curve has changed its shape just because its endpoints have
been moved!!

Reparametrization or change of variables have a dramatic impact on continuity!!

Geometric continuity (G^k continuity)

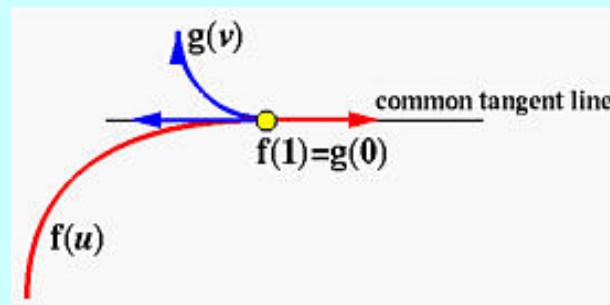
Geometric continuity describes the visual smoothness of a curve rather than the smoothness of motion along a curve.

G^k -continuity requires that the derivative vector have a continuous direction, even though it is allowed to have discontinuities in speed.

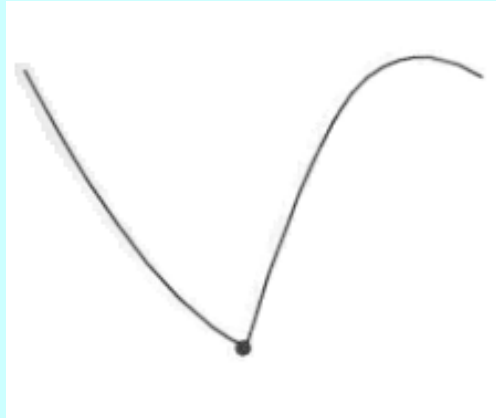
Two C^0 curve segments are said G^1 *geometric continuous* at the joining point **if and only if vectors $f'(u)$ and $g'(v)$ are in the same direction at the joining point**. Note that $f'(u)$ and $g'(v)$ are evaluated at the joining point.

Since the tangent vectors at the joining point have the same direction, both curves have the same tangent line at the joining point.

Two curve segments having the same tangent line **does not** imply they are G^1 at the joining point.



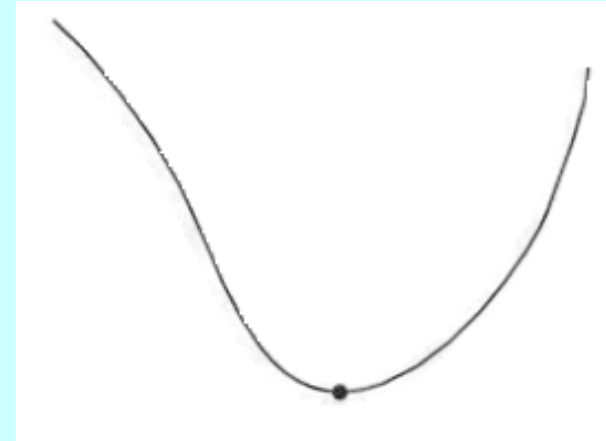
Curve continuity



G^0 continuity
Sharp angle



G^1 continuity
Smooth connection



G^2 continuity
A tight curve

The reason for having two types of continuities has to do with parameter substitution.

Given a curve segment $\mathbf{P}(t)$ where $0 \leq t \leq 1$, we can substitute $T = t^2$.

The new segment $\mathbf{Q}(T) = \mathbf{Q}(t^2)$ where $0 \leq T \leq 1$, is identical in shape to $\mathbf{P}(t)$.

The two identical curves must, of course, have the same tangents.

$$\frac{d\mathbf{Q}(t^2)}{dt} = 2t \frac{d\mathbf{Q}(t)}{dt} = 2t \frac{d\mathbf{P}(t)}{dt}$$

This is why we separate the direction and the magnitude of the tangent vectors when considering curve continuities.

Example: Consider the two straight segments $\mathbf{P}(t) = (8t, 6t)$ and $\mathbf{Q}(t) = (4(t + 2), 3(t+2))$.

The total curve has G^1 continuity at point $(8,6)$, but not C^1 continuity.

It turns out that a curve of the form

$$\mathbf{P}(t) = \sum_{i=0}^n w_i \mathbf{P}_i, \quad \text{where} \quad \sum_{i=0}^n w_i = 1$$

is independent of the particular coordinate axes used.

Describing Curves with Polynomials

An L-th-degree polynomial in t is:

$$a_0 + a_1t + a_2t^2 + \dots + a_Lt^L$$

degree

coefficients

The **order** of the polynomial: the number of coefficients (L + 1)

Polynomial Curves of degree 1:

$$x(t) = a_0 + a_1 t$$

$$y(t) = b_0 + b_1 t$$

straight line

Polynomial Curves of degree 2:

$$x(t) = a_0 + a_1 t + a_2 t^2$$

$$y(t) = b_0 + b_1 t + b_2 t^2 \longrightarrow$$

parabola

Implicit forms of degree 2:

$$F(x, y) = Ax^2 + 2Bxy + Cy^2 + Dx + Ey + F$$

If $AC - B^2 > 0$, the curve is an **ellipse**

If $AC - B^2 = 0$, the curve is a **parabola**

If $AC - B^2 < 0$, the curve is a **hyperbola**

Rational Parametric Forms

Parametric representations using polynomials are simply not powerful enough, because many curves (e.g., circles, ellipses and hyperbolas) can not be obtained this way.

Example:

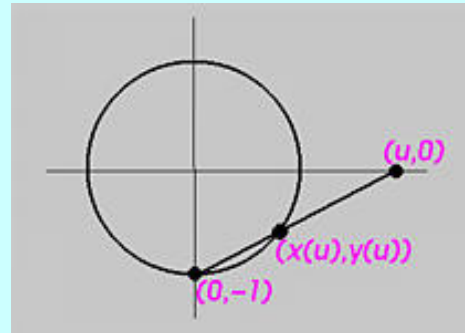
Consider a second degree parametric form:

$$\begin{aligned}x &= f(u) = au^2 + bu + c \\y &= g(u) = pu^2 + qu + r\end{aligned}$$

Let the domain of u be the real line. Now, let u run from negative infinity to positive infinity. The values of x and y will both go to infinity, positive or negative.

In other words, the curve described by the above parametric form contains at least one point at infinity. Since all points of a circle are in finite range, it is impossible to obtain a circle from the above parametric form.

Let us find the rational form of a circle.



The line joining the south pole and $(u, 0)$ is $x = uy + u$.
The circle equation is $x^2 + y^2 = 1$.

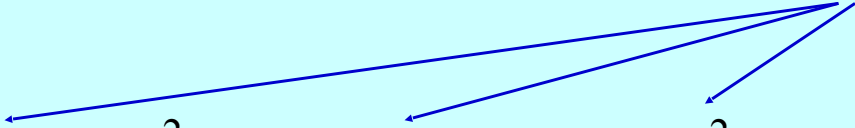
Two roots:
 $y = -1$
 $y = (1 - u^2) / (1 + u^2)$.

$$\begin{aligned} x &= \frac{2u}{1 + u^2} \\ y &= \frac{1 - u^2}{1 + u^2} \end{aligned}$$

Rational Parametric Forms

Quadratic Polynomial

Control points


$$P(t) = \frac{P_0(1-t)^2 + 2wP_1t(1-t) + P_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}$$

$x(\cdot)$ and $y(\cdot)$ are each defined as a **ratio** of two polynomials.

$$(x(t), y(t)) = \left(\frac{x_0(1-t)^2 + 2x_1wt(1-t) + x_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}, \frac{y_0(1-t)^2 + 2y_1wt(1-t) + y_2t^2}{(1-t)^2 + 2wt(1-t) + t^2} \right)$$

$$P(t) = \frac{P_0(1-t)^2 + 2wP_1t(1-t) + P_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}$$

$P(t)$ is a linear combination of control points. (In order to make sense, it must be an affine combination of points)

$P(t)$ interpolates P_0 and P_2 . For
 $t = 0$ and $t = 1$

Parametric curves discussed so far are not very *geometric*.

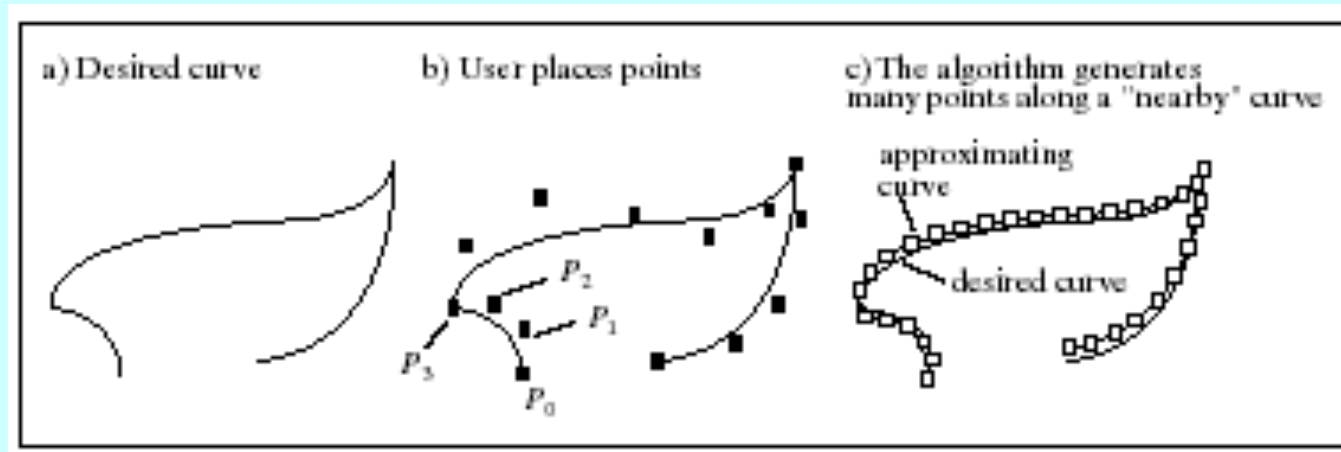
The coefficients of the equations do not have any geometric meaning, and it is almost impossible to predict the change of shape if one or more coefficients are modified.

As a result, designing a curve that follows certain outline is very difficult.

A system that supports users to design curves must be:

- Intuitive
- Flexible
- Unified Approach
- Invariant
- Efficiency and Numerically Stability

On Interactive Curve Design

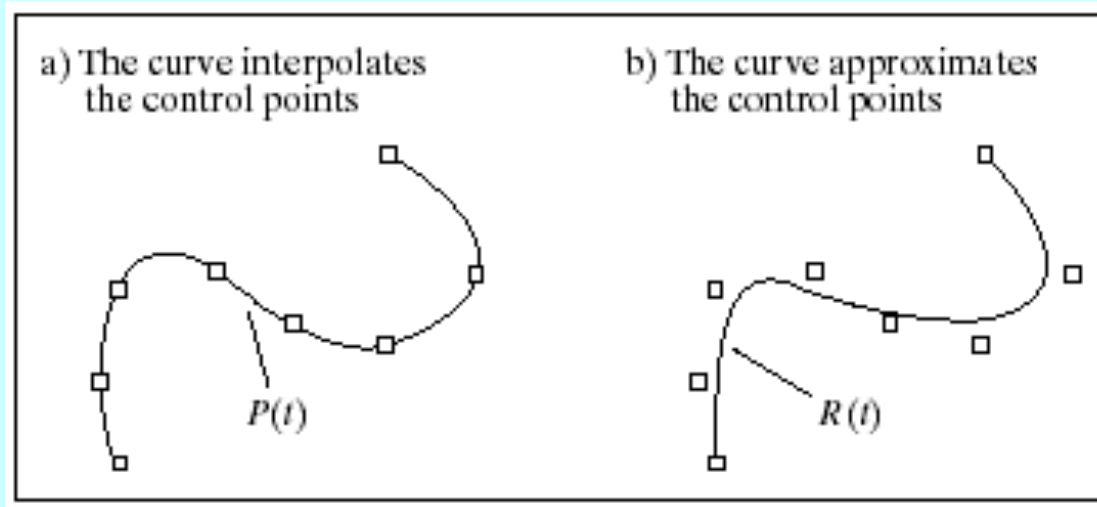


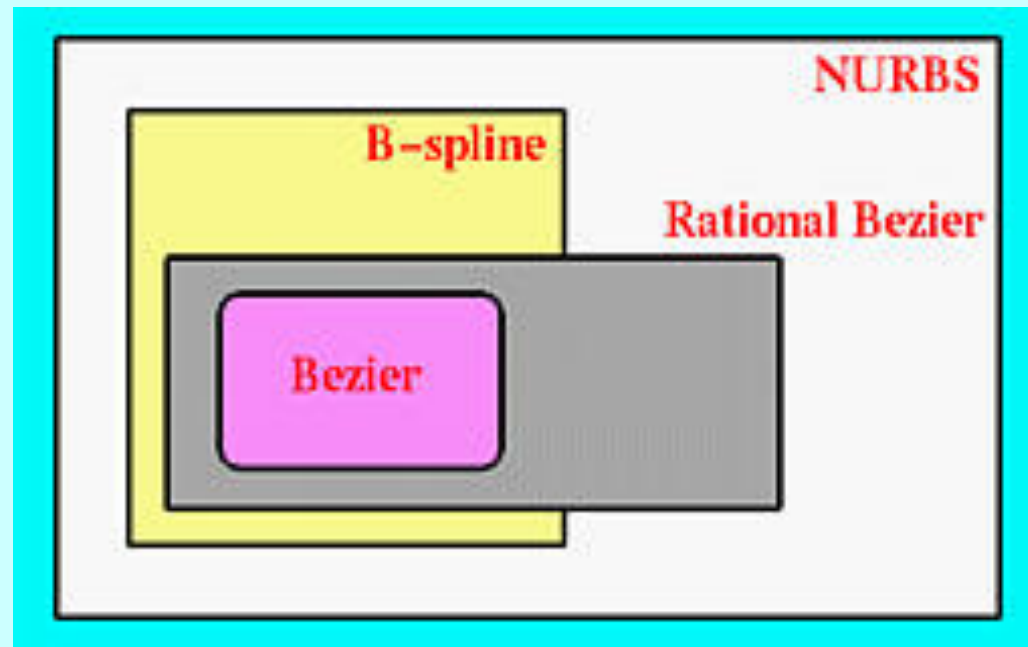
- Lay down the initial control points
- Use the algorithm to generate the curve
- If the curve is satisfactory, stop
- Adjust some control points
- Go to step 2.

Representation Issues

- Parametric/Implicit? [We use polynomials]
- Order/Degree vs continuity (number of pieces?)
We use low order polynomials (usually cubic)
- Interpolating vs approximating
- Level of continuity
- Convex Hull
- Number of parameters
- Invariance (Euclidean, affine, reparam,...)
- Local control
- Variational optimality

Interpolation versus Approximation





Bezier Curves for Curve Design

Historical Notes

Pierre Etienne Bezier was an applied mathematician with Renault. In the early 1960s, he began searching for ways to automate the process of designing cars. His methods have been the basis of modern CAGD.

Paul de Faget de Casteljau – an applied mathematician with Citroen- was the first, in 1959, to develop the various Bezier methods but because of the secretiveness of his employer, never published it.

Steven Anson Coons of MIT did most of his work on surfaces (around 1967) while a consultant for Ford.

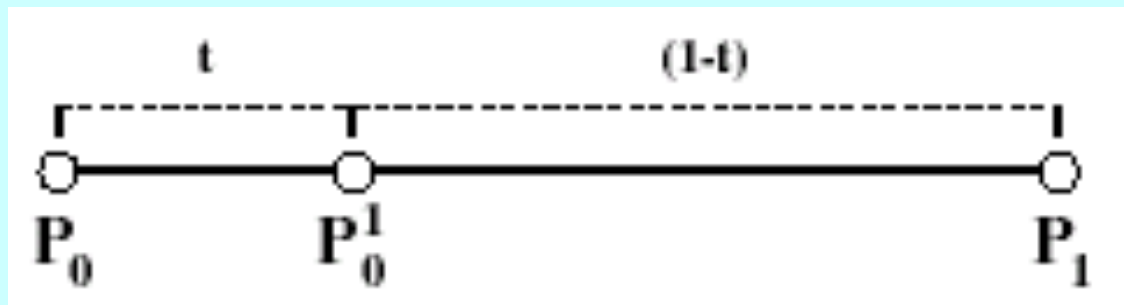
William J. Gordon, has generalized the Coons surfaces, in 1969, at General Motors.

Constructing Curve Segments

Linear blend:

Line segment from an affine combination of points

$$P_0^1(t) = (1 - t)P_0 + tP_1$$



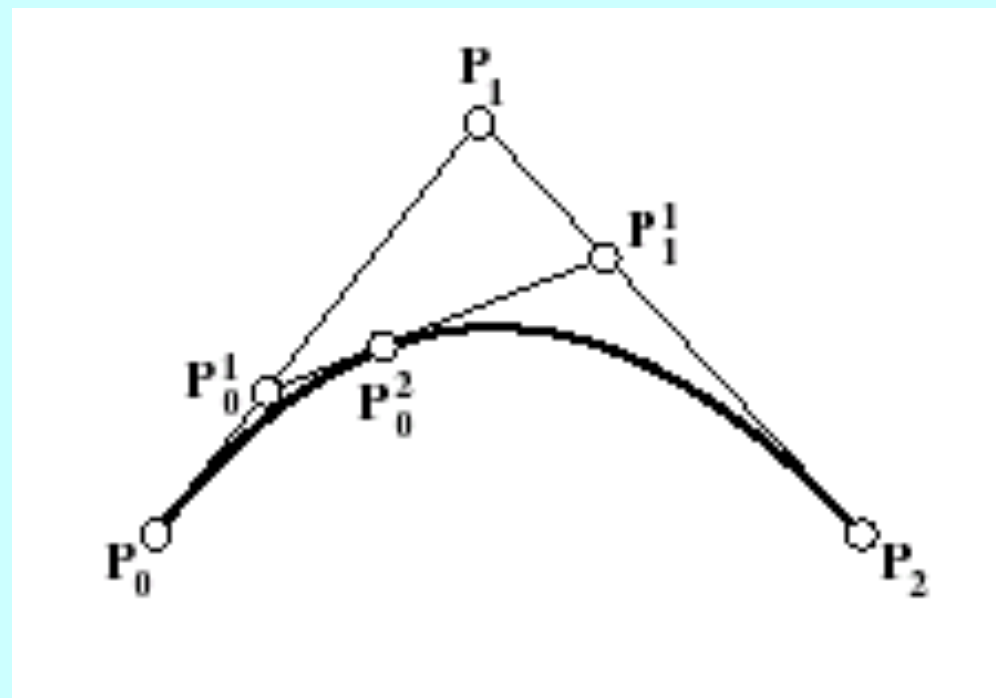
Quadratic blend:

Quadratic segment from an affine combination of line segments

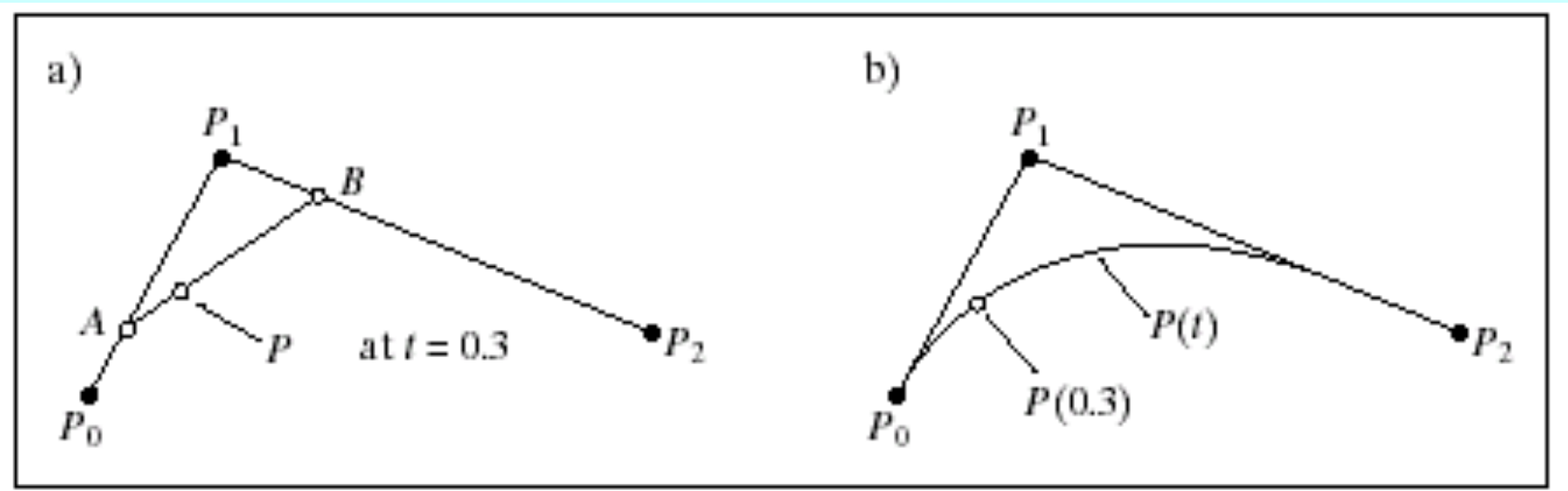
$$P_0^1(t) = (1 - t)P_0 + tP_1$$

$$P_1^1(t) = (1 - t)P_1 + tP_2$$

$$P_0^2(t) = (1 - t)P_0^1 + tP_1^1$$



deCasteljau Algorithm

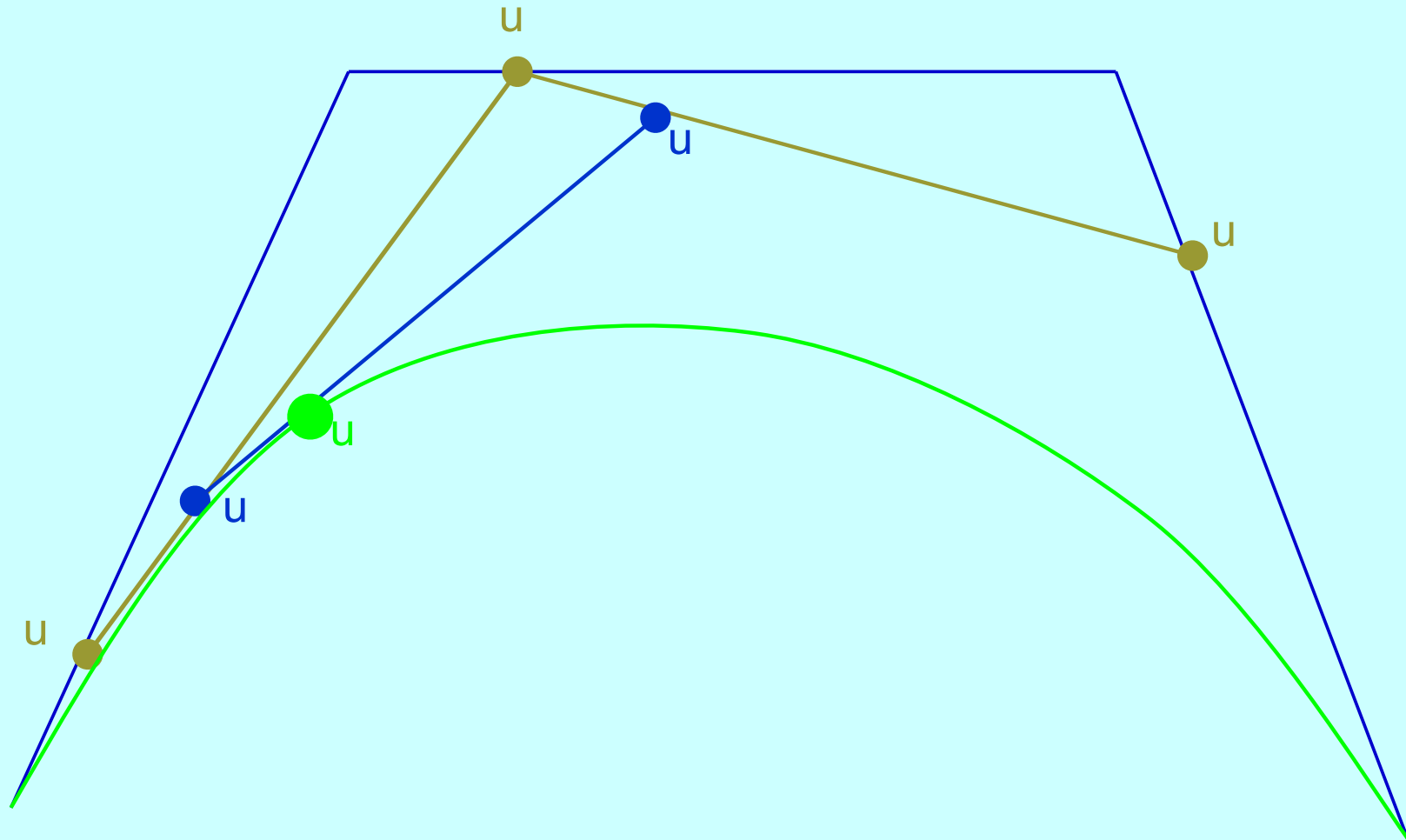


$$B(t) = (1-t) P_1 + tP_2$$

$$P(t) = (1-t) A + tB$$

$$P(t) = (1-t)^2 P_0 + 2t (1-t)P_1 + t^2P_2$$

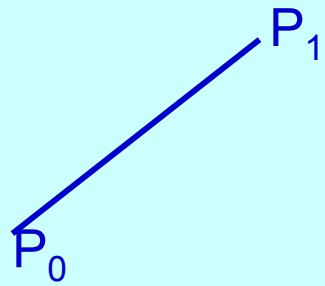
Geometrically



Geometric view (deCasteljau Algorithm):

- Join the points P_i by line segments
- Join the $t : (1 - t)$ points of those line segments by line segments
- Repeat as necessary
- The $t : (1 - t)$ point on the final line segment is a point on the curve
- The final line segment is tangent to the curve at t

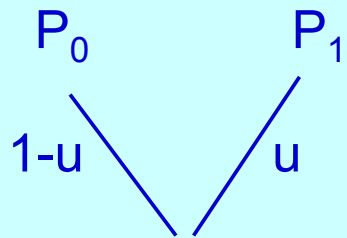
deCasteljau Algorithm



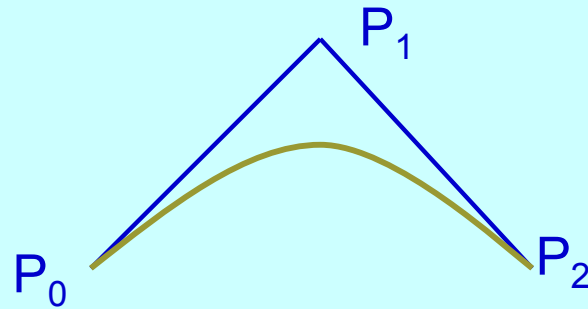
Linear

Degree 1, Order 2

$$F(0) = P_0, F(1) = P_1$$



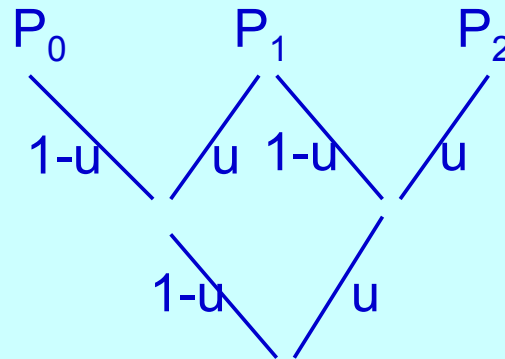
$$F(u) = (1-u) P_0 + u P_1$$



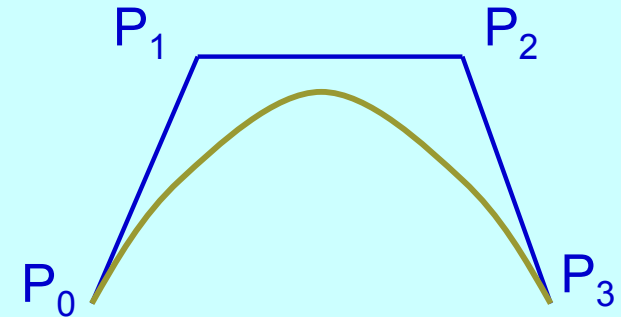
Quadratic

Degree 2, Order 3

$$F(0) = P_0, F(1) = P_2$$



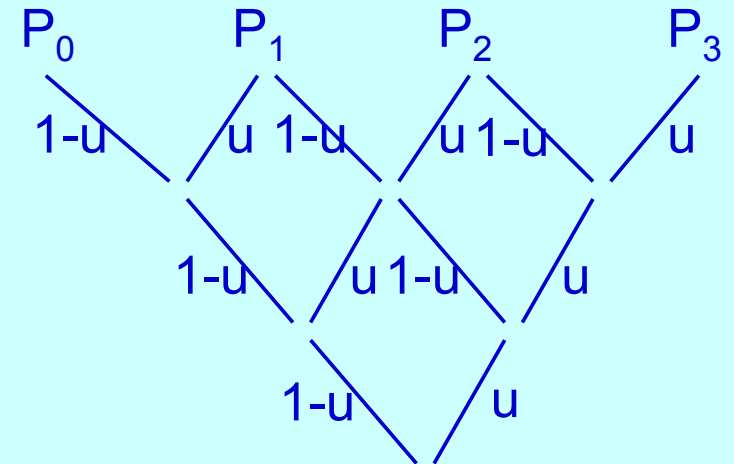
$$F(u) = (1-u)^2 P_0 + 2u(1-u) P_1 + u^2 P_2$$



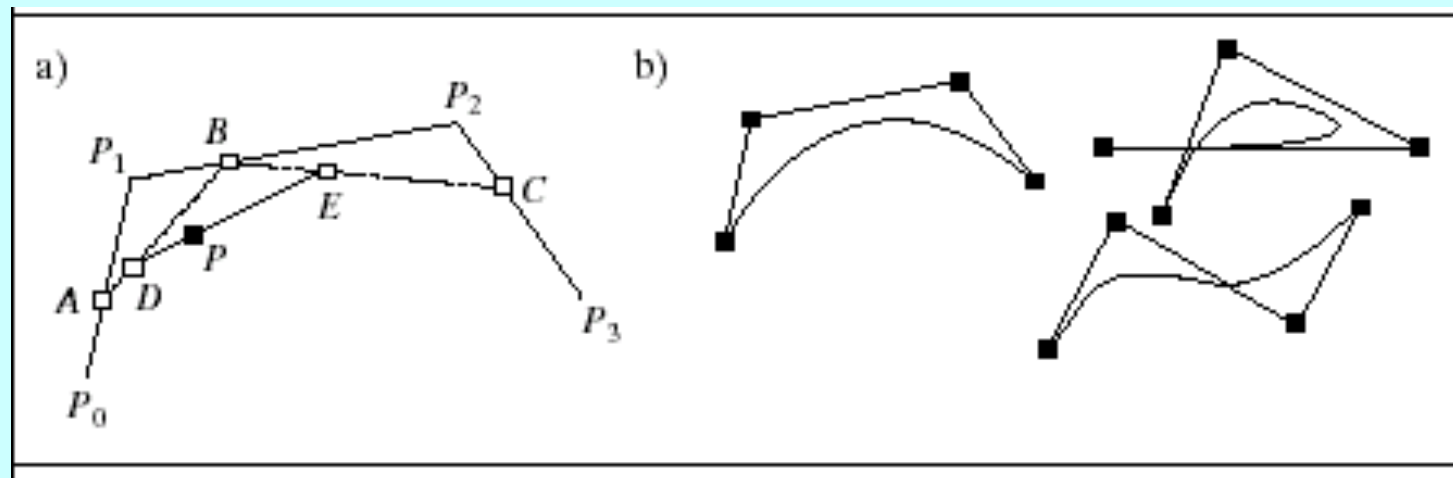
Cubic

Degree 3, Order 4

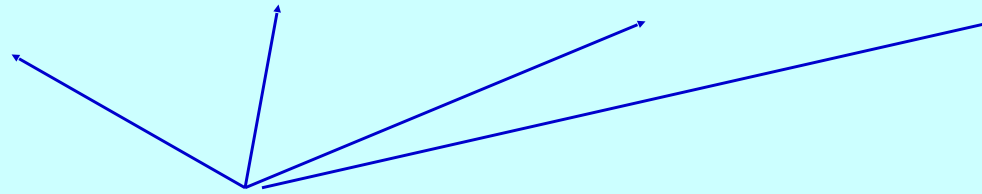
$$F(0) = P_0, F(1) = P_3$$



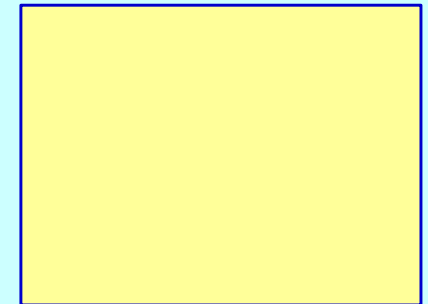
$$F(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u) P_2 + u^3 P_3$$



The Bezier curve based on four points:



Berstein polynomials



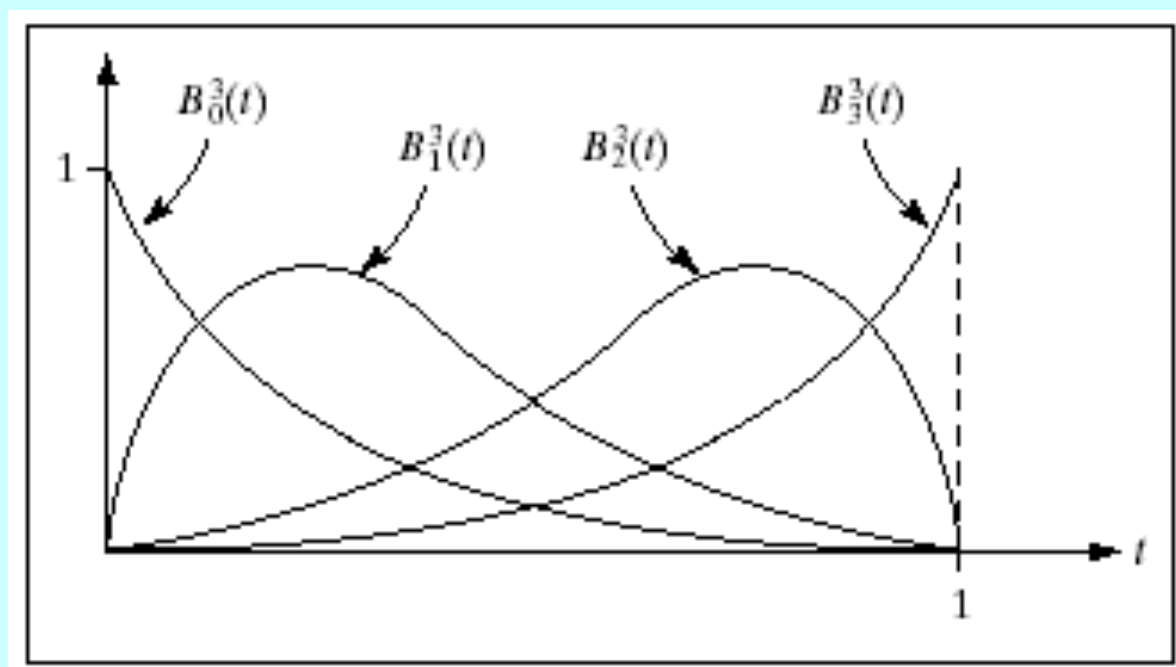
Given $n+1$ points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots$ and \mathbf{P}_n in space, the *control points*, the *Bézier curve* defined by these control points is

$$C(u) = \sum_{i=1}^n B_{n,i}(u) \mathbf{P}_i$$

where the coefficients are defined as follows:

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

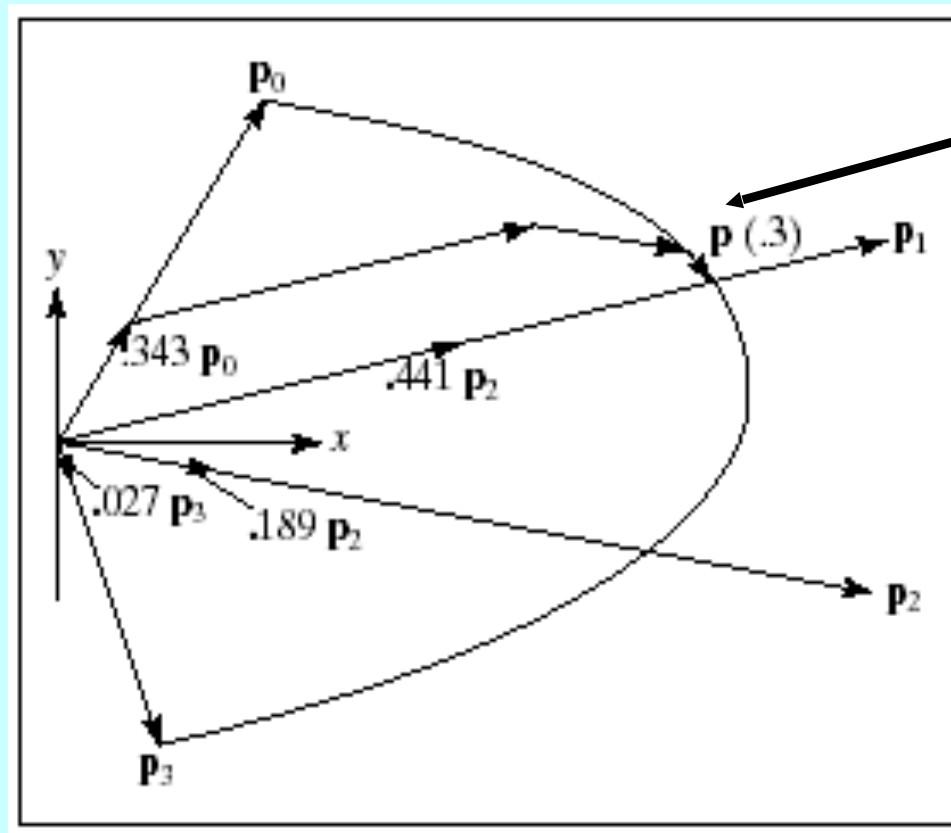
Therefore, the point that corresponds to u on the Bézier curve is the "weighted" average of all control points, where the weights are the coefficients $B_{n,i}(u)$.



If we view the points as vectors bound to the origin, and let $t = 0.3$

$$P(t) = P_0(1-t)^3 + P_13(1-t)^2t + P_23(1-t)t^2 + P_3t^3$$

$$\mathbf{p}(0.3) = 0.343\mathbf{p}_0 + 0.441\mathbf{p}_1 + 0.189\mathbf{p}_2 + 0.027\mathbf{p}_3$$



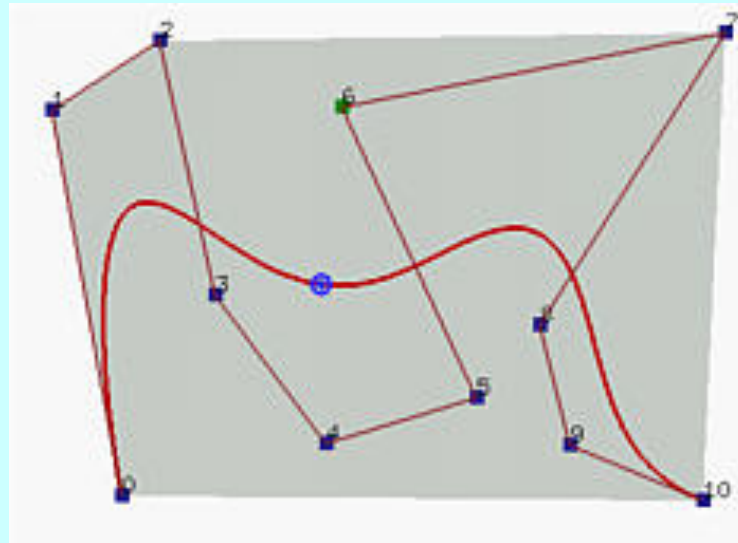
Extending the de Casteljau Algorithm to any number of points

$$P_i^4(t) = (1-t)P_i^3(t) + tP_{i+1}^3(t)$$

$$P_i^L(t) = (1-t)P_i^{L-1}(t) + tP_{i+1}^{L-1}(t)$$

$$P(t) = \sum_{k=0}^L P_k B_k^L(t) \quad B_k^L(t) = \binom{L}{k} (1-t)^{L-k} t^k$$

Note that the domain of u is $[0,1]$. As a result, all basis functions are non-negative. In the above, since u and i can both be zero and so do $1 - u$ and $n - i$, we adopt the convention that 0^0 is 1.

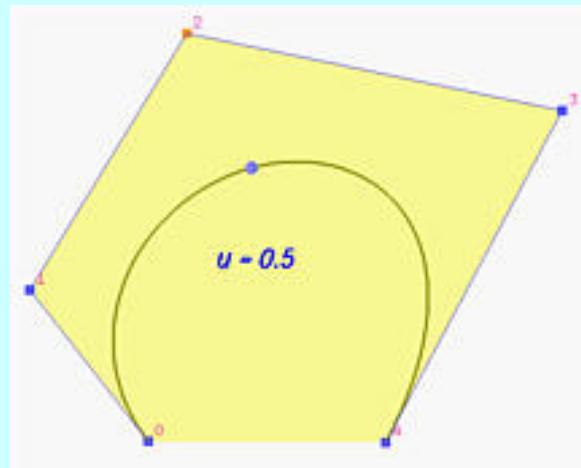


a Bézier curve defined by 11 control points, where the blue dot is a point on the curve that corresponds to $u=0.4$.

Properties of Bezier Curves

- **The degree of a Bézier curve defined by $n+1$ control points is n :**
In each basis function, the exponent of u is $i + (n - i) = n$.
Therefore, the degree of the curve is n .

$C(u)$ passes through P_0 and P_n :

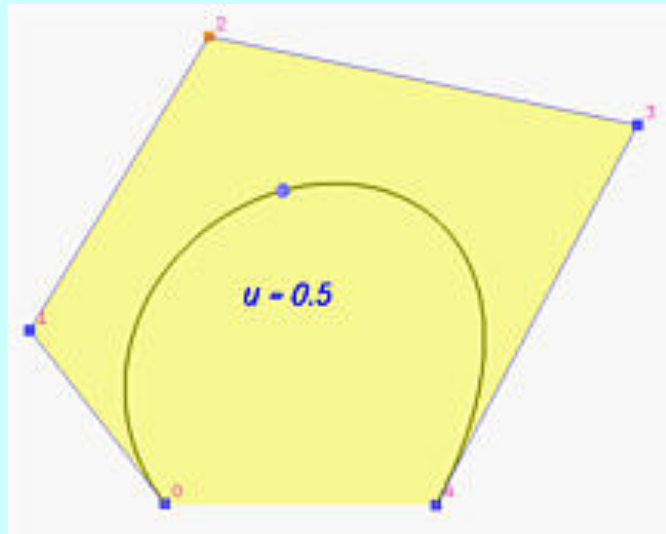


- **Non-negativity:**
All basis functions are non-negative.

Properties of Bezier Curves

- **Partition of Unity:**

The sum of the basis functions at a fixed u is 1.



The figures show $u=0.5$ and all five basis functions.

The right-most vertical bar shows the way of partitioning 1 into five intervals.

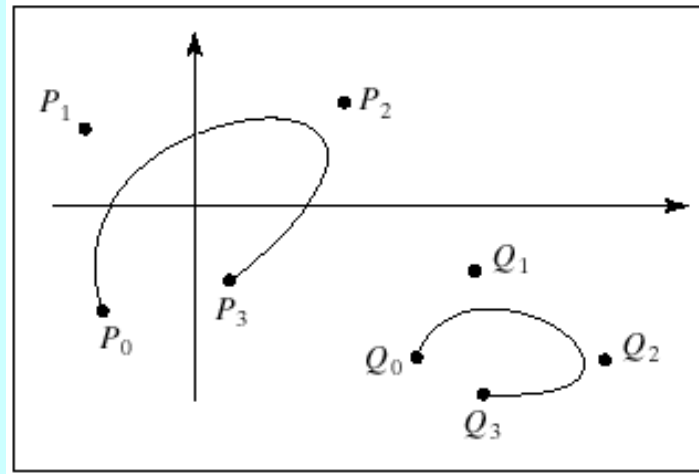
Note that the color of a partition is identical to the color used to draw its corresponding basis functions.

Properties of Bezier Curves

Affine Invariance

If an affine transformation is applied to a Bézier curve, the result can be constructed from the affine images of its control points.

$$Q(t) = \sum_{k=0}^L T(P_k) B_k^L(t)$$



The transformed curve is identical, point by point, to the the curve that is based on the transformed control points. (Bezier curves are formed as an affine combination of points)

Properties of Bezier Curves

Invariance under Affine transformation of the parameter

Bezier curves are defined for $t \in [0, 1]$. It may be convenient to use a different interval

$$t = \frac{u - a}{b - a}$$

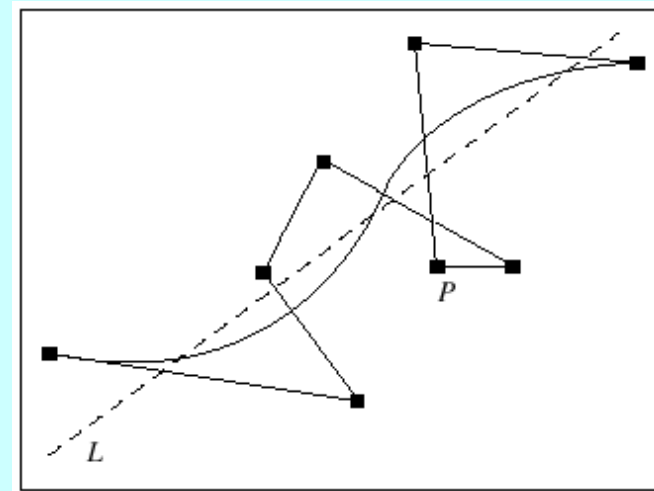
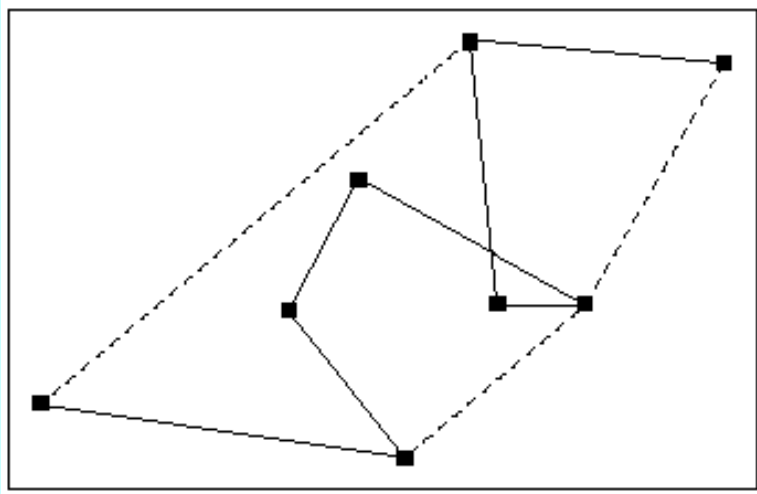
The Bezier curve is now given by

$$R(u) = \sum_{k=0}^L P_k B_k^L \left(\frac{u - a}{b - a} \right)$$

Properties of Bezier Curves

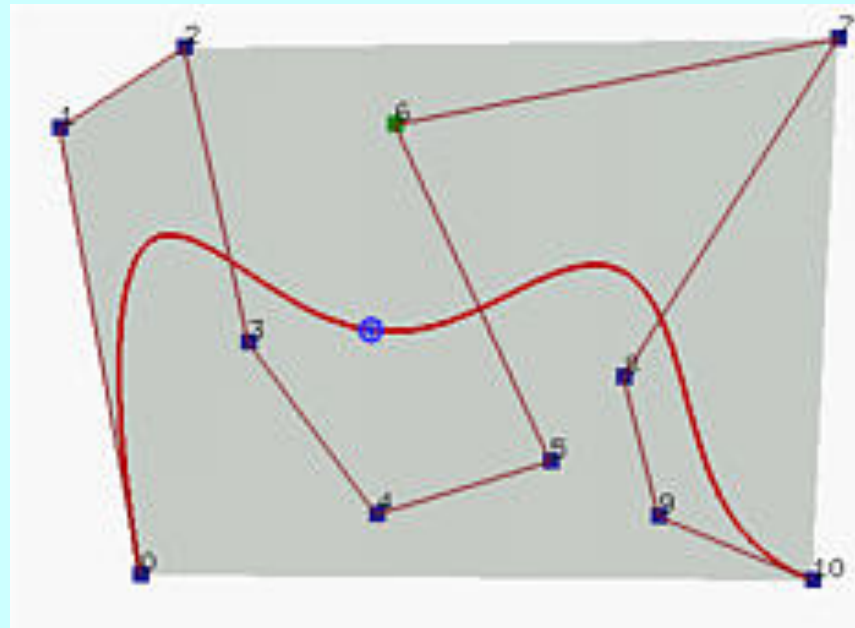
Convex-hull property

Bezier curve $P(t)$ never wanders outside the convex hull of the control points. ($P(t)$ is a convex combination of points)



Properties of Bezier Curves

Convex-hull property

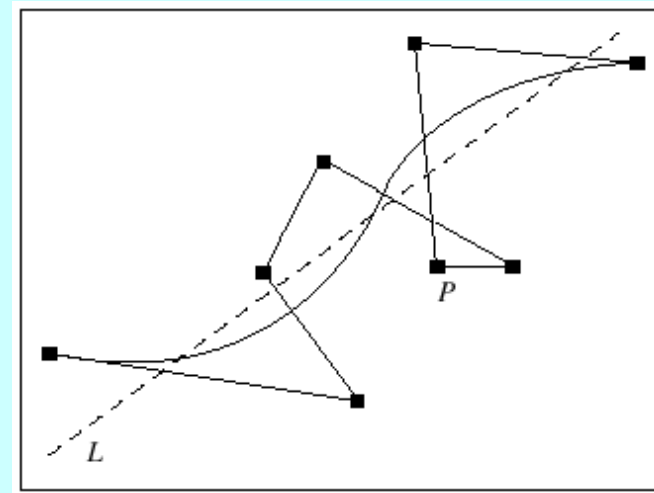


This property is important because we are guaranteed that the generated curve will be in an understood and computable region and will not go outside of it.

Properties of Bezier Curves

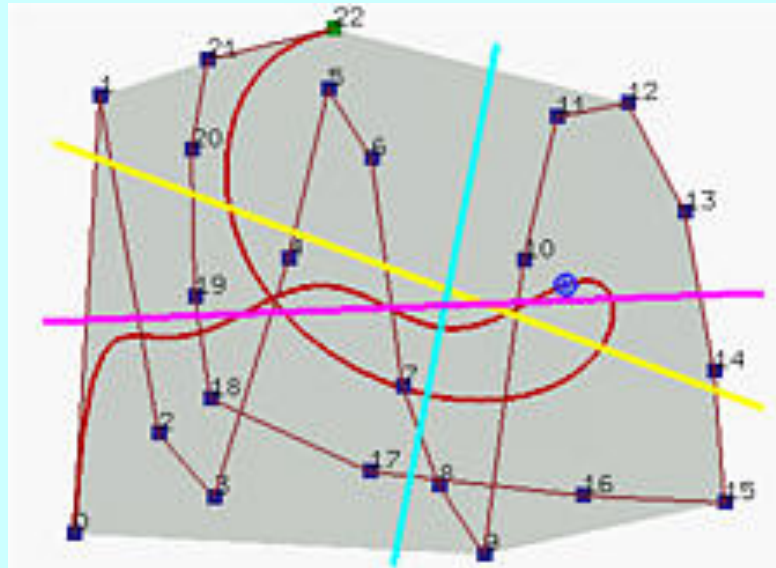
Variation-diminishing property

Bezier curves cannot “fluctuate” more than their control polygon does. (No straight line can have more intersections with a Bezier curve than it has with the curve’s control polygon.



Properties of Bezier Curves

Variation-diminishing property



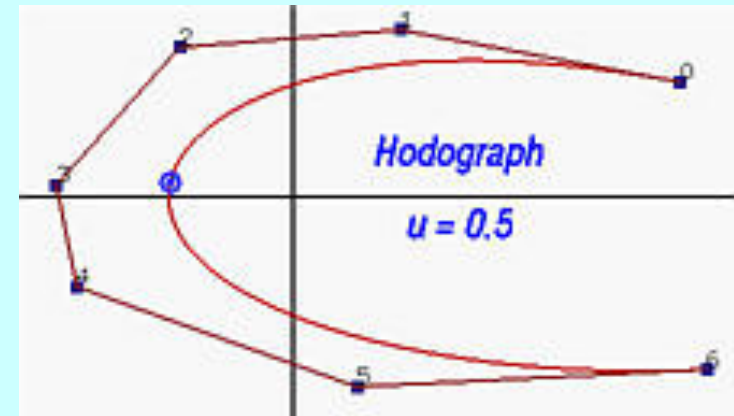
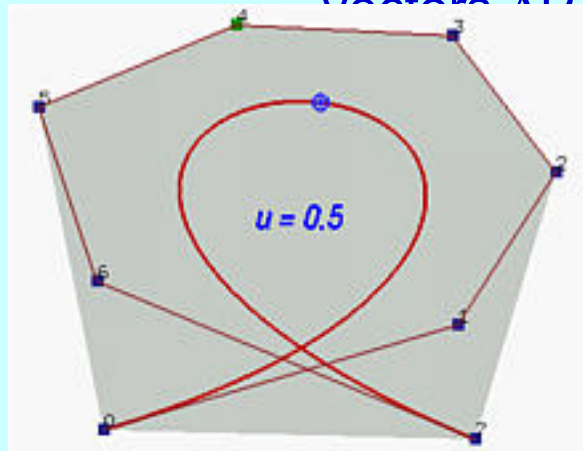
The yellow line intersects the curve 3 times and the polyline 7 times; the magenta line intersects the curve 5 times and the polyline 7 times; and the cyan line intersects the curve and its polyline twice.

Derivatives of Bezier Curves

To compute tangent and normal vectors at a point on a Bézier curve, we must compute the first and second derivatives at that point.

$$\mathbf{p}'(t) = L \sum_{k=0}^{L-1} \Delta \mathbf{P}_k B_k^{L-1}(t) \quad \text{where} \quad \Delta \mathbf{P}_k = \mathbf{P}_{k+1} - \mathbf{P}_k$$

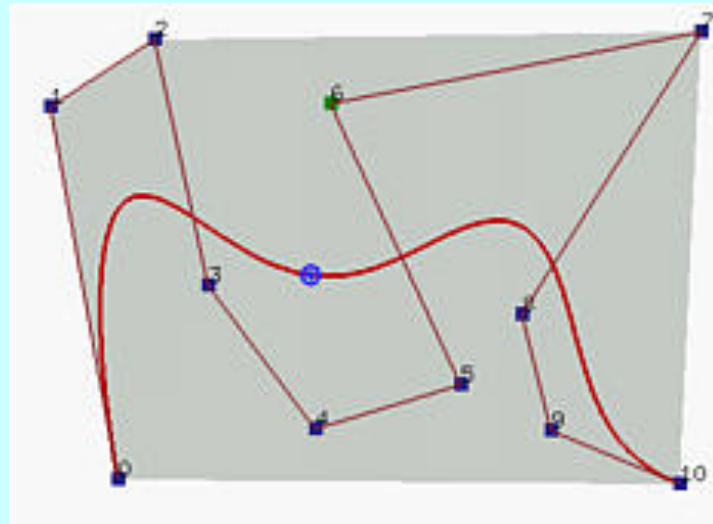
So the velocity is another Bezier curve, built on a new set of control vectors $\Delta \mathbf{P}_k$.



Bézier Curves Are Tangent to Their First and Last Legs

Letting $t = 0$ and $t = 1$ gives $\mathbf{C}'(0) = n (\mathbf{P}_1 - \mathbf{P}_0)$ and $\mathbf{C}'(1) = n (\mathbf{P}_n - \mathbf{P}_{n-1})$

Therefore, the first leg in the indicated direction is tangent to the Bézier curve



Joining Two Bézier Curves with C^1 -Continuity

That a Bézier curve being tangent to its first and last legs provides us with a technique for joining two or more Bézier curves together for designing a desired shape.

Let the first curve $\mathbf{C}(u)$ be defined by $m + 1$ **control points**

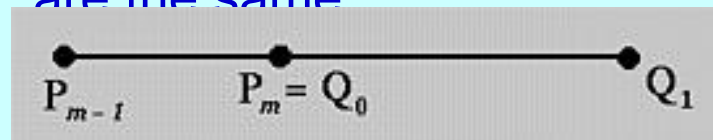
$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_m$.

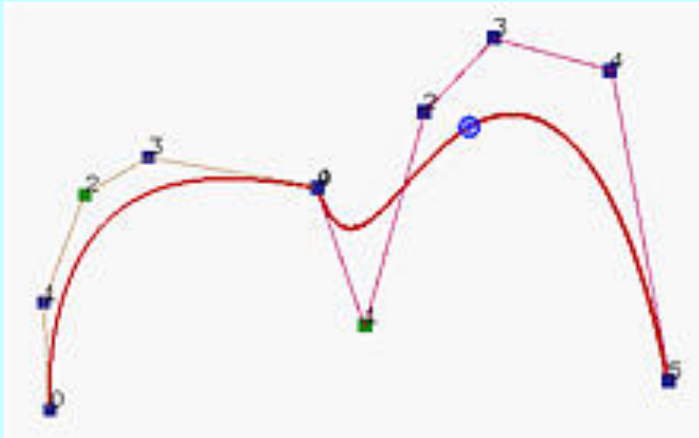
Let the second curve $\mathbf{D}(u)$ be defined by $n + 1$ **control points**

$\mathbf{Q}_0, \mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_n$.

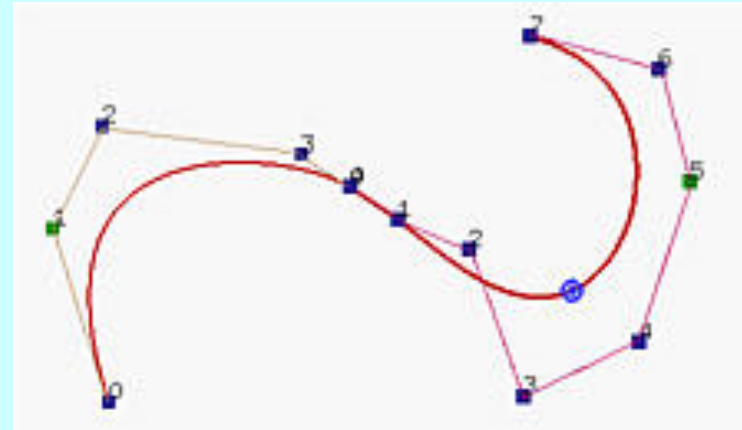
\mathbf{P}_m must be equal to \mathbf{Q}_0 for C^0 continuity.

For a smooth transition, \mathbf{P}_{m-1} , \mathbf{P}_m and \mathbf{Q}_0 , \mathbf{Q}_1 must be on the same line such that the directions from \mathbf{P}_{m-1} to \mathbf{P}_m and the direction from \mathbf{Q}_0 to \mathbf{Q}_1 are the same



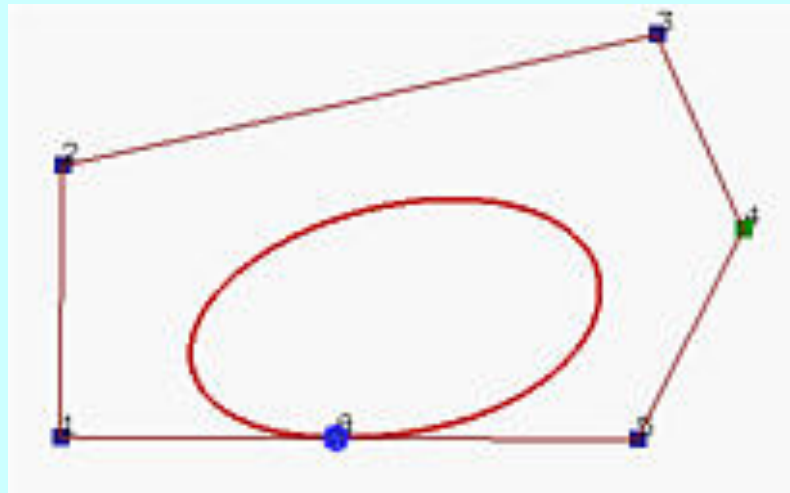


Since the last leg of the first curve and the first leg of the second are not on the same line, the two curves are not joint smoothly.



If we let the first and last control points be identical (*i.e.*, $\mathbf{P}_0 = \mathbf{P}_n$) and \mathbf{P}_1 , \mathbf{P}_0 and \mathbf{P}_{n-1} collinear, the generated Bézier curve will be a closed one and is G^1 continuous at the joining point as shown below.

To achieve C^1 continuous at \mathbf{P}_0 , we must have $\mathbf{P}_1 - \mathbf{P}_0 = \mathbf{P}_n - \mathbf{P}_{n-1}$ (*i.e.*, the first and last legs have the same length and $\mathbf{P}_1, \mathbf{P}_0 = \mathbf{P}_n, \mathbf{P}_{n-1}$ are collinear)



A Short Summary

To define a Bézier curve of degree n , we need to choose $n + 1$ control points in space so that they roughly indicate the shape of the desired curve.

Then, if it is not up to our expectation, we can move the control points around. As one or more control points are moved, the shape of the Bézier curve changes accordingly. But, the curve always lies in the convex hull defined by the control points (the convex hull property), and the shape of generated curve is less complex than the control polyline (variation diminishing property).

Bezier (de Casteljau) Curves

Properties:

Endpoint interpolation.

Invariant under any affine mapping.

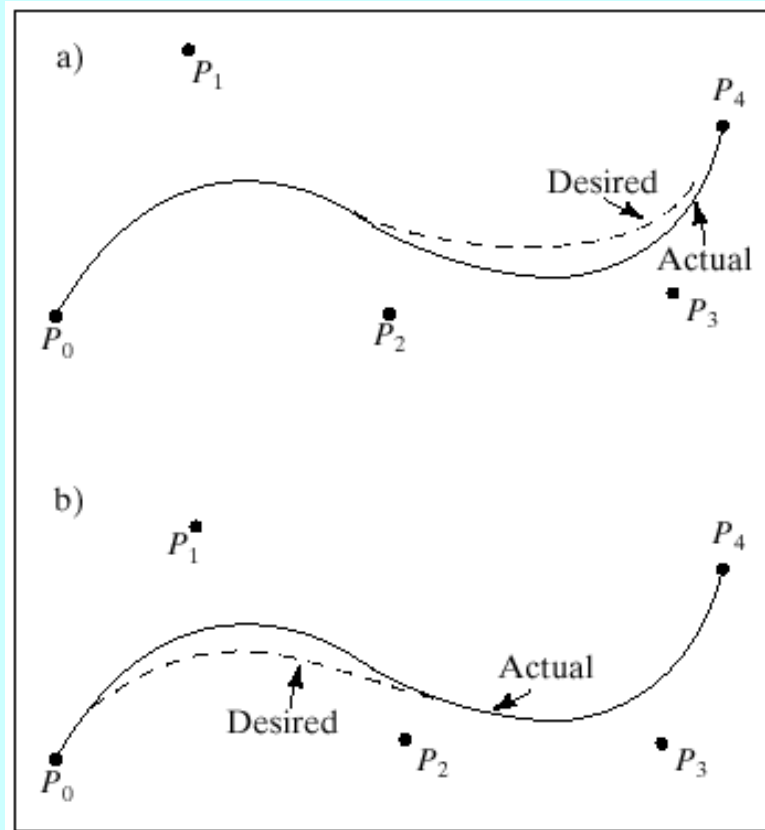
Even a change to the axis system

Convex hull property

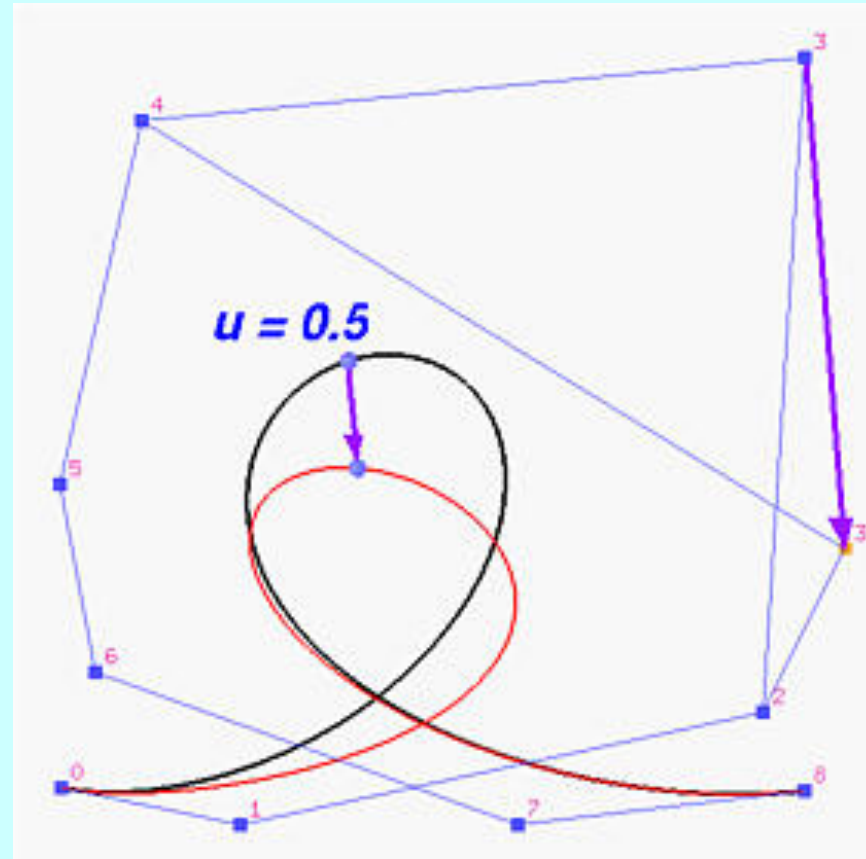
Smoothness and versatility

Great for CAD!

Global but no local control?

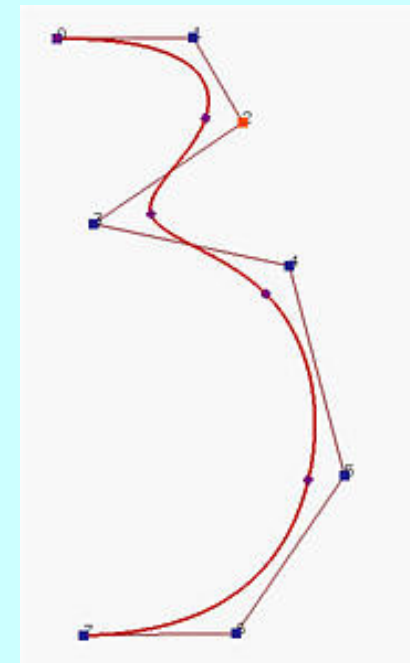
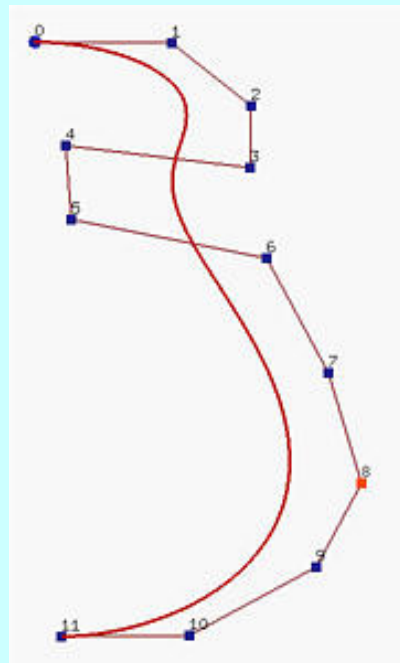


A change to any point alters the *entire* curve.

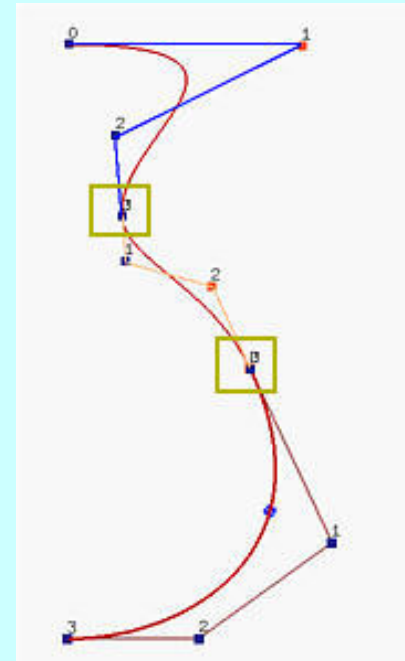
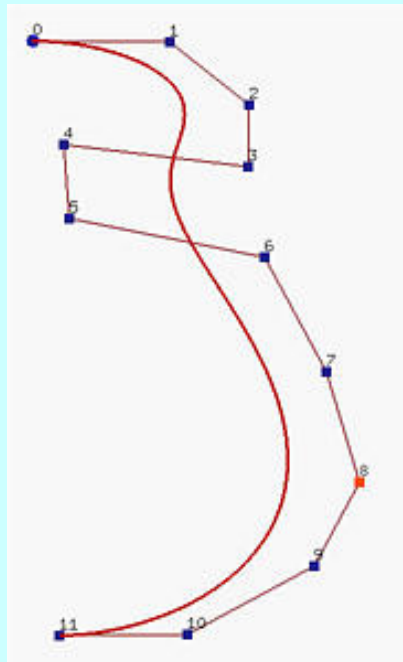


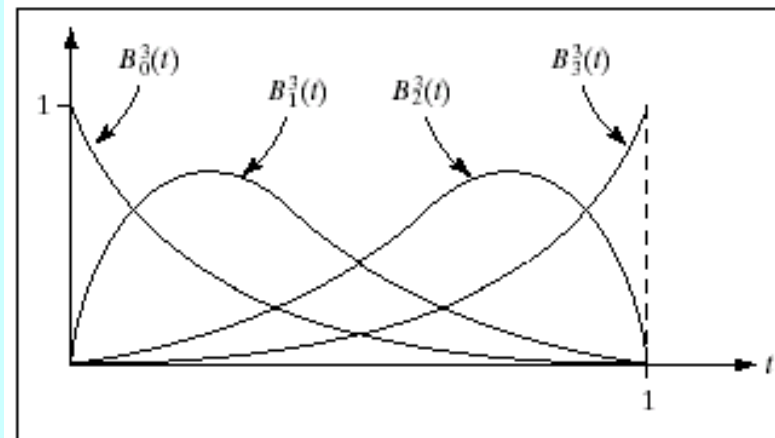
Consider designing the profile of a vase. The left figure below is a Bézier curve of degree 11; but, it is difficult to bend the "neck" toward the line segment P_4P_5 .

Of course, we can add more control points near this segment to increase the weight to that region. However, this will increase the degree of the curve.



We can join two Bézier curves together. As long as the last leg of the first curve and the first leg of the second have the same direction, we can at least achieve G^1 continuity.

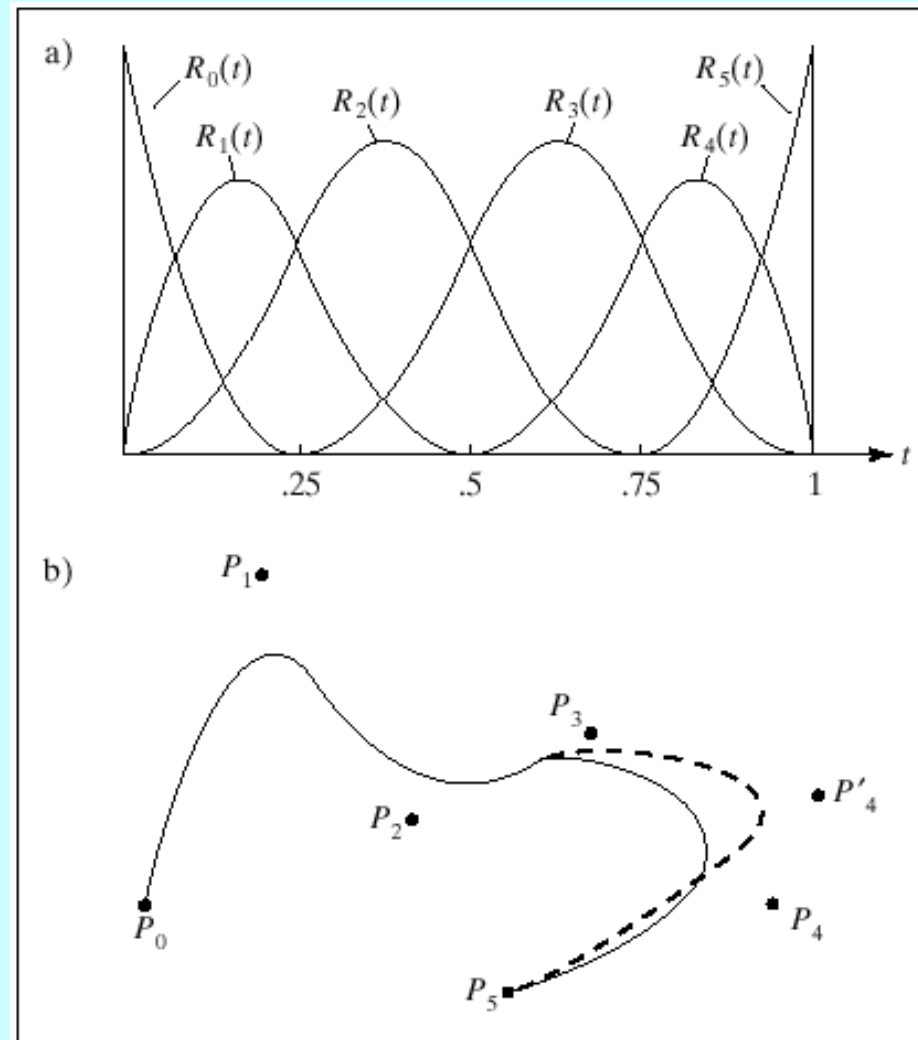




Each Bernstein polynomial is active over the entire interval $[0,1]$
Has support $[0,1]$

Finding better blending functions

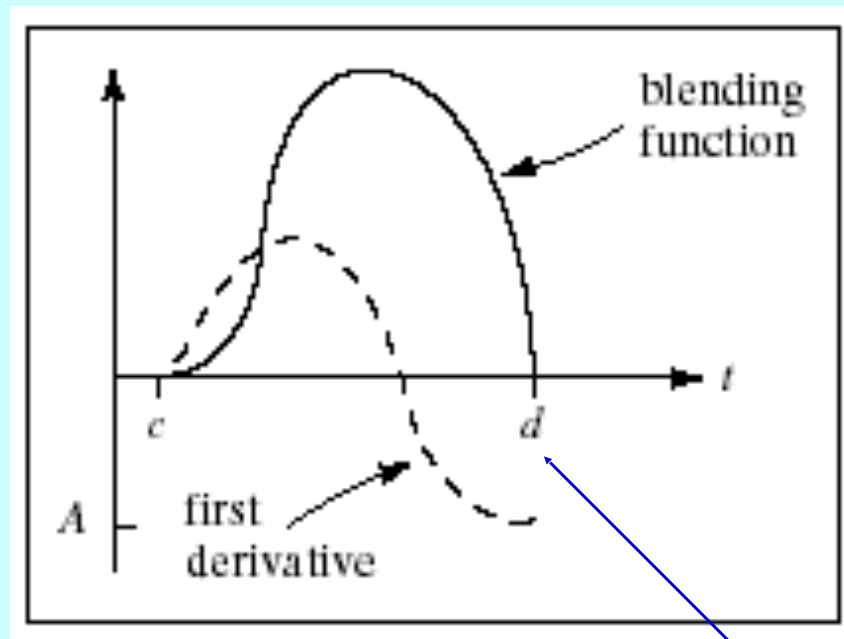
Partial support



Wish list for a set of blending functions

- be easy to compute and numerically stable
- sum to unity at every t in $[a,b]$
- have support only over a small portion of $[a,b]$, to offer local control
- interpolate certain control points, chosen by the designer
- be smooth enough to produce a desirable shape

- be smooth enough to produce a desirable shape
 - Typically $V(t)$ should be at least 1-smooth, maybe even 2-smooth
 - The smoothness of $V(t)$ depends on the smoothness of the blending function.



Derivative is discontinuous
One smooth everywhere except at $t=c$

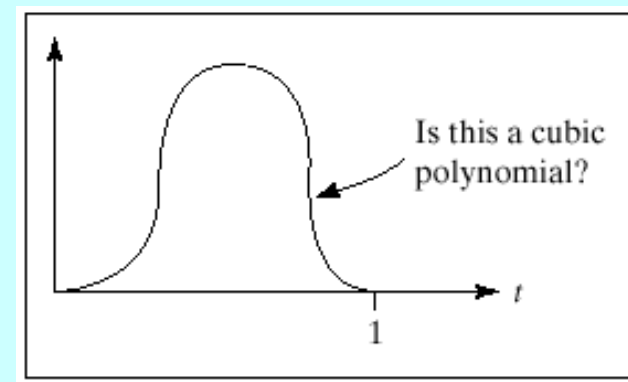
Jump to 0

Piecewise polynomial curves and splines

Let's start looking for a good candidate blending function, say, polynomials of low degree.

Is there a cubic polynomial that will satisfy all our needs?

$$R(t) = at^3 + bt^2 + ct + d$$



The first derivatives must be both zero at $t=0$, $t=1$

Unfortunately, conditions force $a = b = c = d = 0$, so there is no such shape.

There just isn't enough flexibility in a cubic to do the "bending" we need.

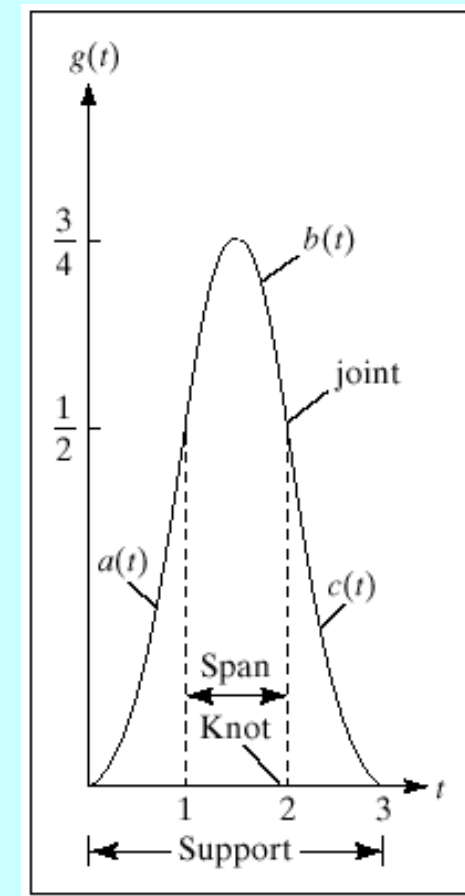
To attain more flexibility, we can try “piecing” together several low-degree polynomials. (Piecewise polynomials)

$$a(t) = \frac{1}{2}t^2 \quad \text{Span } [0,1]$$

$$b(t) = \frac{3}{4}\left(t - \frac{3}{2}\right)^2 \quad \text{Span } [1,2]$$

$$c(t) = \frac{1}{2}(3-t)^2 \quad \text{Span } [2,3]$$

The points at which a pair of the individual segments meet are called **joints** and the values of t at which this happens are called **knots**



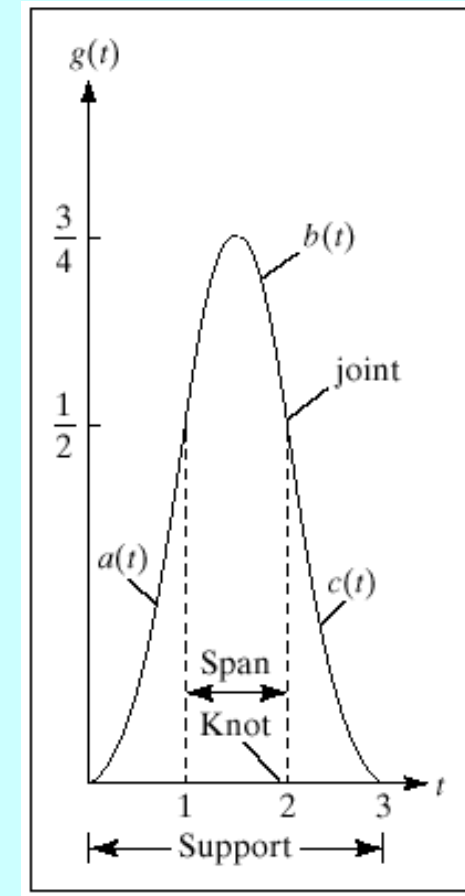
Is $g(t)$ continuous everywhere?

Is $g(t)$ 1-smooth?

$$a(t) = \frac{1}{2}t^2$$

$$b(t) = \frac{3}{4}\left(t - \frac{3}{2}\right)^2$$

$$c(t) = \frac{1}{2}(3-t)^2$$



$g(t)$ is an example of a **spline function**, a piecewise continuous function that possesses “enough” smoothness.

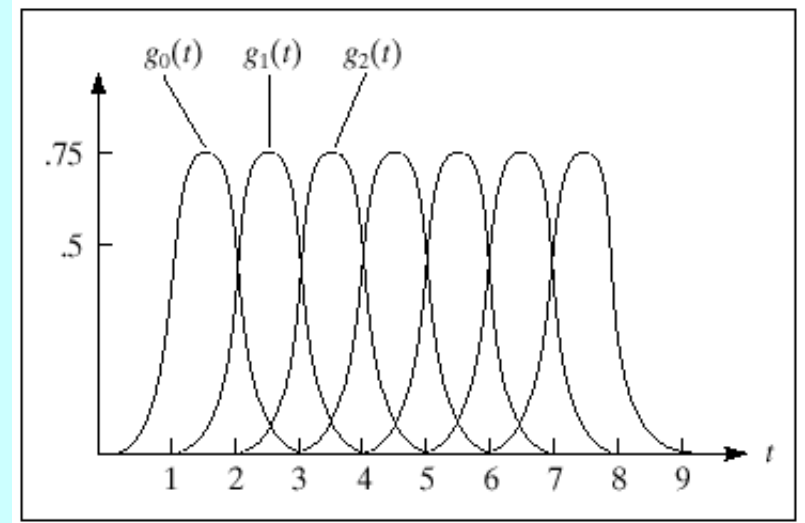
An M th-degree spline function is a piecewise polynomial of degree M that is $(M-1)$ smooth at each knot

Building a set of blending functions out of $g(t)$

Let's start looking for a good candidate blending function, say, polynomials of low degree.

$$g(t) = g(t - k), \quad \text{for } k = 0, 1, \dots$$

$$\sum_{k=0}^6 g(t - k) = 1 \quad \text{for } t \in [2, 7]$$

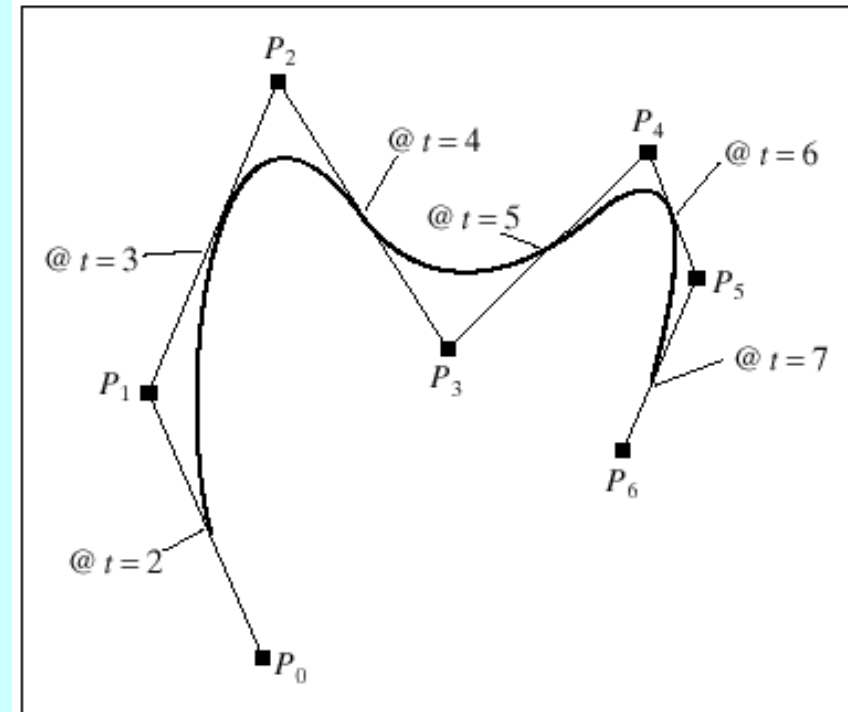


We have 7 blending Functions g_0 to g_6 , formed by shifting $g(t)$ by Integer amounts

So the designer chooses seen control points and generates the curve:

$$V(t) = \sum_{k=0}^6 P_k g(t-k)$$

- Note that exactly three of the blending functions are active at any value of t (between 2 and 7), so there is good local control of the curve's shape.
- Note that at times $t = 2, 3, \dots, 7$ only two of the functions are active and they both have the value 0.5

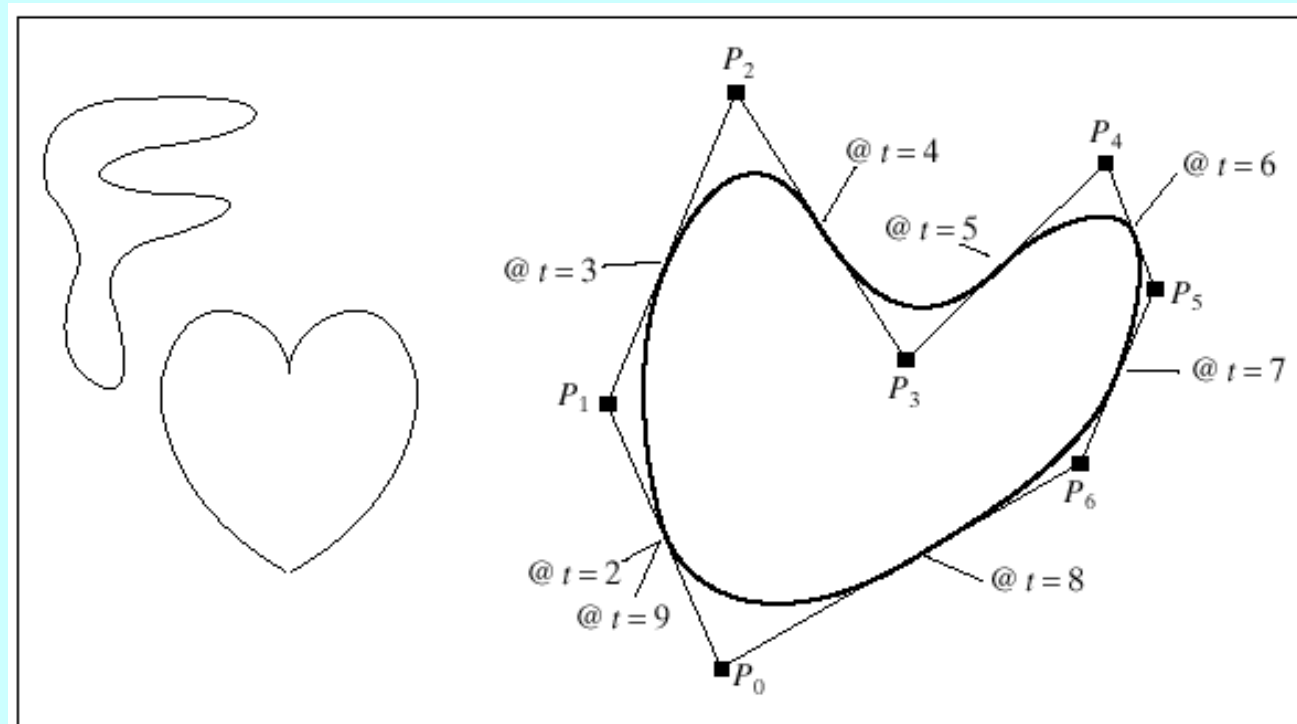


Some properties

- The designer has some local control over shape since the support interval for each blending function is limited to 3.
- The designer will lay down points exploiting the knowledge that the curve must pass through midpoints of the edges.
- Since each blending function is 1-smooth, the whole curve is 1-smooth
- No points on the curve are interpolated
- All polynomials are of degree 2, so they are fast and stable to compute

Polynomials are curve of choice

- PostScript (Adobe)
- TrueFont (MicroSoft)



$$P_7 = P_0$$

$$P_8 = P_1$$

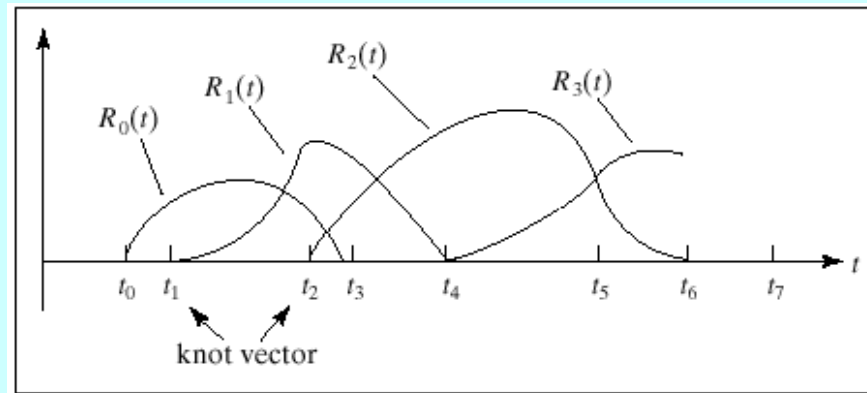
Wish list for a set of blending functions

- be easy to compute and numerically stable
 - sum to unity at every t in $[a,b]$
 - have support only over a small portion of $[a,b]$, to offer local control
- interpolate certain control points, chosen by the designer
- be smooth enough to produce a desirable shape

Thus we seek more general families of blending functions

$$P(t) = \sum_{k=0}^L P_k R_k(t)$$

based on $L + 1$ control points and $L + 1$ blending functions



Is there some family of blending functions that can be used to generate every possible spline curve that can be defined on that knot vector?

Such a family is called a **basis** for the splines.

There are many such families, but there is one basis in particular whose blending functions have the smallest support and therefore offer the greatest local control: the **B-splines**


The B-spline basis functions

Each B-spline function is based on polynomials of a certain order m .

$m = 3$, quadratic B-spline

$m = 4$, cubic B-spline

The B-spline basis functions

$$P(t) = \sum_{k=0}^L P_k N_{k,m}(t)$$



k^{th} B-spline blending function of order m .

The ingredients:

- a knot vector $T = (t_0, t_1, t_2, \dots)$
- $(L+1)$ control points P_k ; and
- the order m of the B-spline functions

The whole curve is a sum of piecewise polynomials weighted by the control points.

The B-spline basis functions

$$P(t) = \sum_{k=0}^L P_k N_{k,m}(t)$$


k^{th} B-spline blending function of order m .

Unlike a Bézier curve, a B-spline curve involves more information, namely: a set of $L+1$ control points, a knot vector of $m+1$ knots, and a degree p . Note that n , m and p must satisfy $m = L + p + 1$.

More precisely, if we want to define a B-spline curve of degree p with $L + 1$ control points, we have to supply $L + p + 2$ knots

$$t_0, t_1, \dots, t_{L+p+1}.$$

On the other hand, if a knot vector of $m + 1$ knots and $L + 1$ control points are given, the degree of the B-spline curve is $p = m - L - 1$.

Although $N_{i,p}(u)$ looks like $B_{n,i}(u)$, the degree of a B-spline basis function is an input, while the degree of a Bézier basis function depends on the number of control points.

To change the shape of a B-spline curve, one can modify one or more of these control parameters: the positions of control points, the positions of knots, and the degree of the curve.

B-spline Curves: Computing the Coefficients

The fundamental formula for the B-spline function $N_{k,m}(t)$ is:

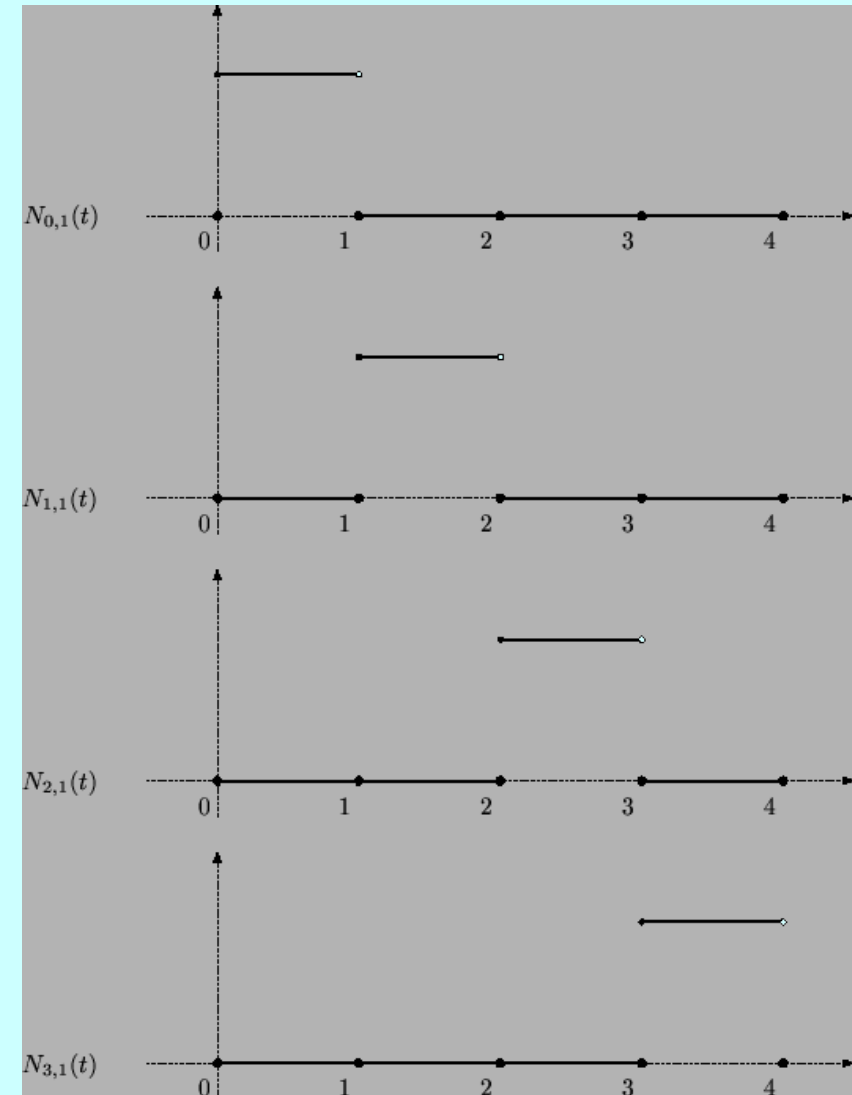
$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} - t_k} \right) N_{k,m-1}(t) + \left(\frac{t_{k+1} - t}{t_{k+m} - t_{k+1}} \right) N_{k+1,m-1}(t)$$

This is a recursive definition, specifying how to construct the m th order function from two B-spline functions of order $(m-1)$

for $k = 0, 1, \dots, L$.

Blending Functions for $k = 1$

$$N_{k,1}(t) = \begin{cases} 1 & \text{if } t_k < t \leq t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

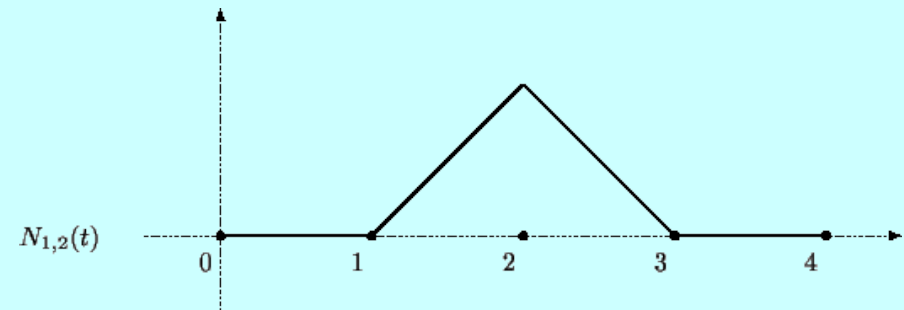
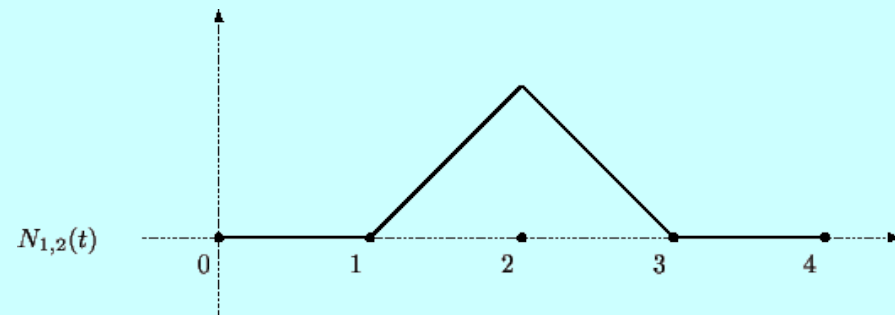
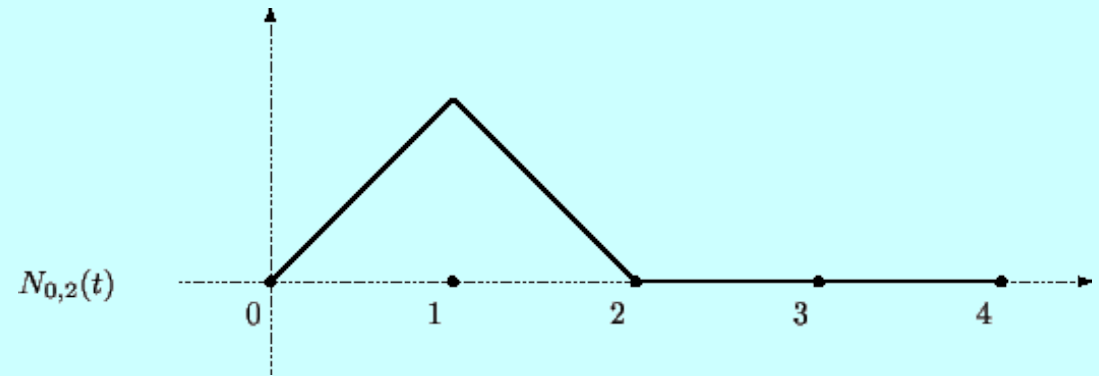


Blending Functions for $k = 2$

Linear B-splines

$$N_{0,2}(t) = \frac{t}{1} N_{0,1}(t) + \frac{2-t}{1} N_{1,1}(t)$$

$$N_{0,1}(t) = \begin{cases} 1 & \text{if } t_0 < t \leq t_1 \\ 0 & \text{otherwise} \end{cases}$$

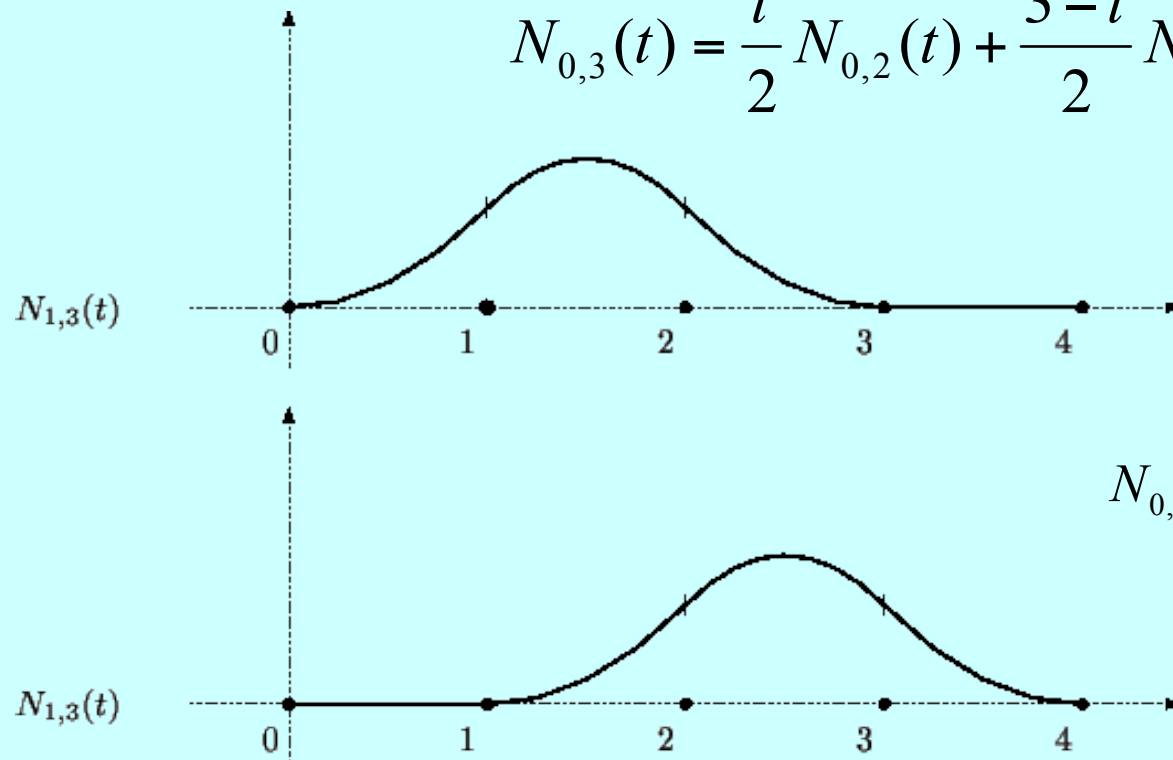


Blending Functions for $k = 3$ Quadratic B-splines

$$N_{0,1}(t) = \begin{cases} 1 & \text{if } t_0 < t \leq t_1 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{0,2}(t) = \frac{t}{1} N_{0,1}(t) + \frac{2-t}{1} N_{1,1}(t)$$

$$N_{0,3}(t) = \frac{t}{2} N_{0,2}(t) + \frac{3-t}{2} N_{1,2}(t)$$

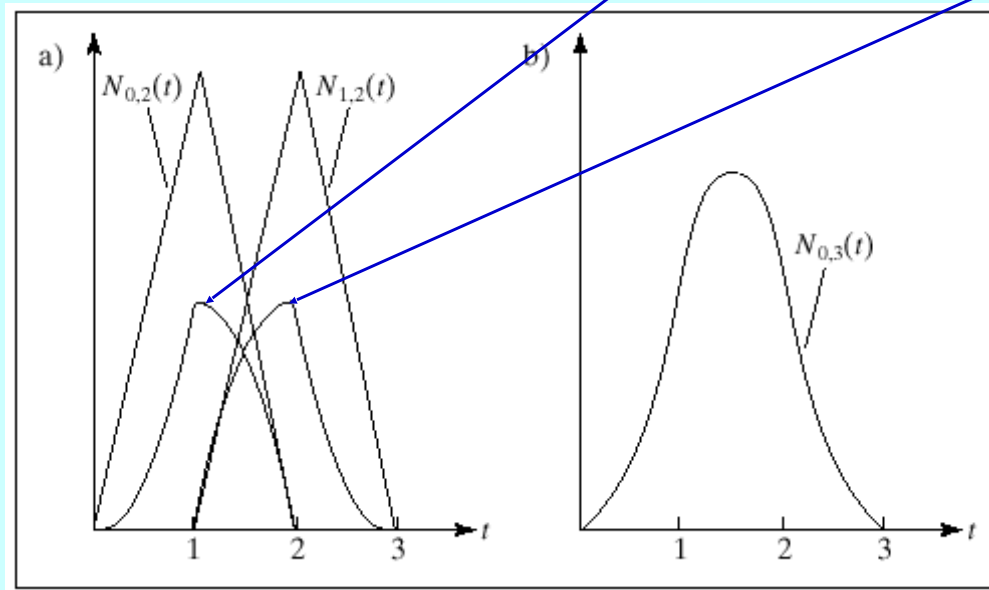


$$N_{0,3}(t) = \begin{cases} \frac{1}{2}t^2 & \text{for } 0 \leq t \leq 1 \\ \frac{3}{4} - \left(t - \frac{3}{2}\right)^2 & \text{for } 1 \leq t \leq 2 \\ \frac{1}{2}(3-t)^2 & \text{for } 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{0,1}(t) = \begin{cases} 1 & \text{if } t_0 < t \leq t_1 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{0,2}(t) = \frac{t}{1} N_{0,1}(t) + \frac{2-t}{1} N_{1,1}(t)$$

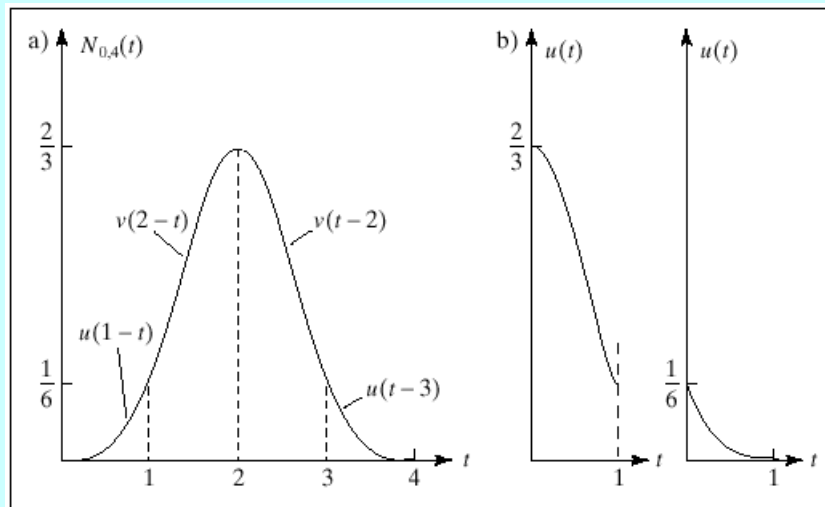
$$N_{0,3}(t) = \frac{t}{2} N_{0,2}(t) + \frac{3-t}{2} N_{1,2}(t)$$



$$N_{0,3}(t) = \begin{cases} \frac{1}{2}t^2 & \text{for } 0 \leq t \leq 1 \\ \frac{3}{4} - \left(t - \frac{3}{2}\right)^2 & \text{for } 1 \leq t \leq 2 \\ \frac{1}{2}(3-t)^2 & \text{for } 2 \leq t \leq 3 \\ 0, & \text{otherwise} \end{cases}$$

Cubic B-splines

$$N_{0,4}(t) = \begin{cases} u(1-t) & \text{for } 0 \leq t \leq 1 \\ v(2-t) & \text{for } 1 \leq t \leq 2 \\ v(t-2) & \text{for } 2 \leq t \leq 3 \\ u(t-3) & \text{for } 3 \leq t \leq 4 \\ 0 & \text{otherwise} \end{cases}$$



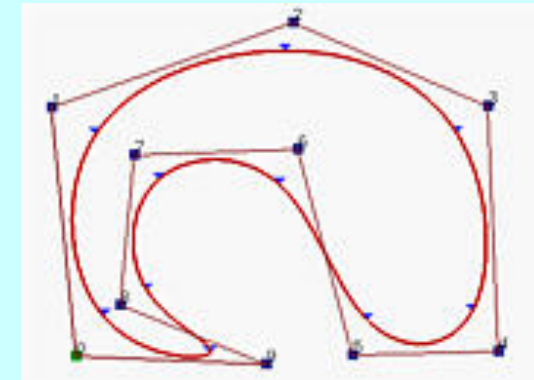
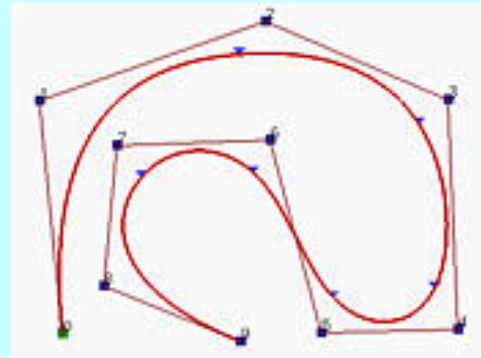
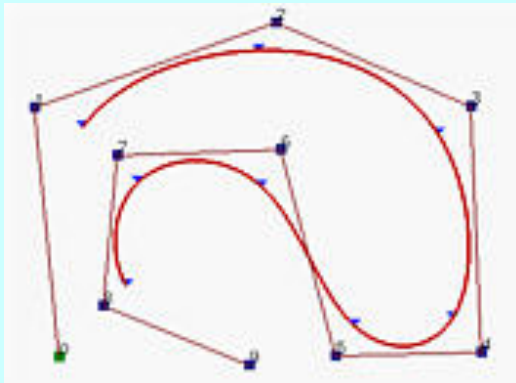
$$u(t) = \frac{1}{6}(1-t)^3$$

$$v(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

The first and second derivatives of the cubic spline are everywhere continuous, so **cubic spline curves are 2-smooth**, at least on equispaced knots.

If the knot vector does not have any particular structure, the generated curve **will not touch the first and last legs** of the control polyline as shown in the figure below.

This type of B-spline curves is called *open* B-spline curves.



To clamp the curve so that it is tangent to the first and the last legs at the first and last control points, respectively, as a Bézier curve does, the first knot and the last knot **must be of multiplicity $p+1$** .

This will generate the so-called *clamped* B-spline curves.

By repeating some knots and control points, the generated curve can be a *closed* one.

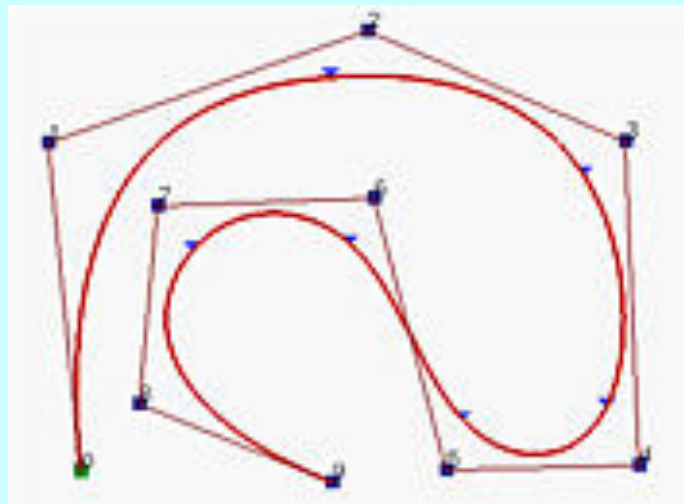
The figures have $n+1$ control points ($n=9$) and $p = 3$.

Then, m must be 13 so that the knot vector has 14 knots.

To have the clamped effect, the first $p+1 = 4$ and the last 4 knots must be identical.

The remaining $14 - (4 + 4) = 6$ knots can be anywhere in the domain. In fact, the curve is generated with knot vector

$$U = \{ 0, 0, 0, 0, 0.14, 0.28, 0.42, 0.57, 0.71, 0.85, 1, 1, 1, 1 \}.$$



B-spline Curves: Closed Curves

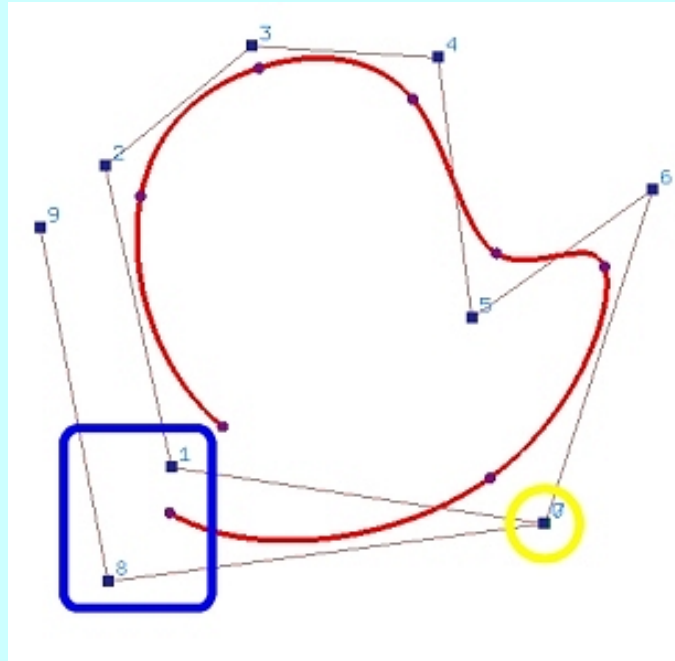
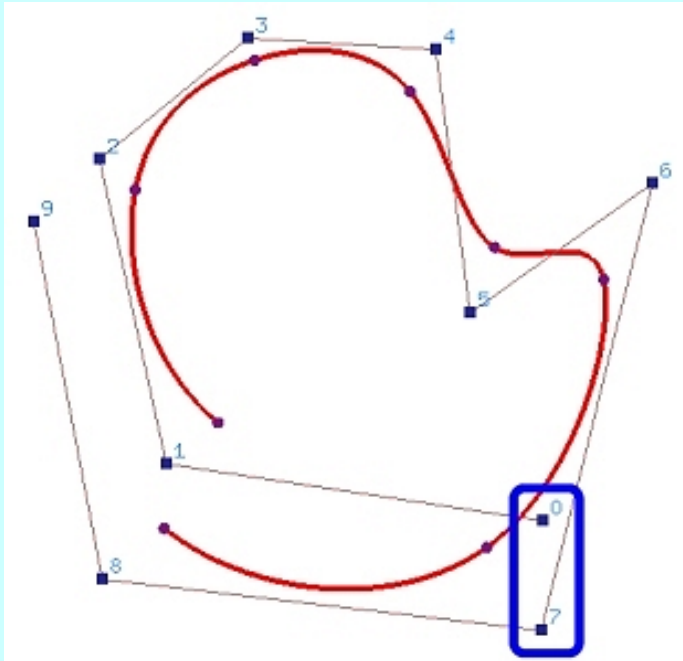
There are many ways to generate closed curves.

The simple ones are

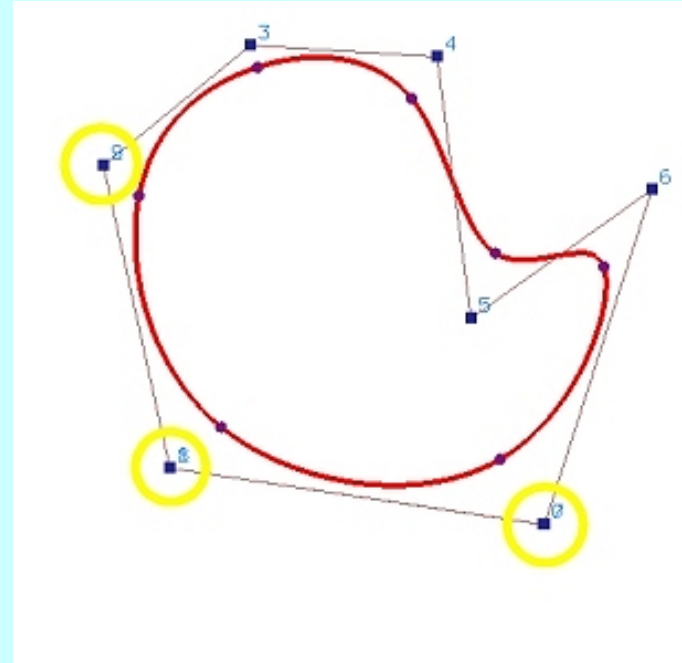
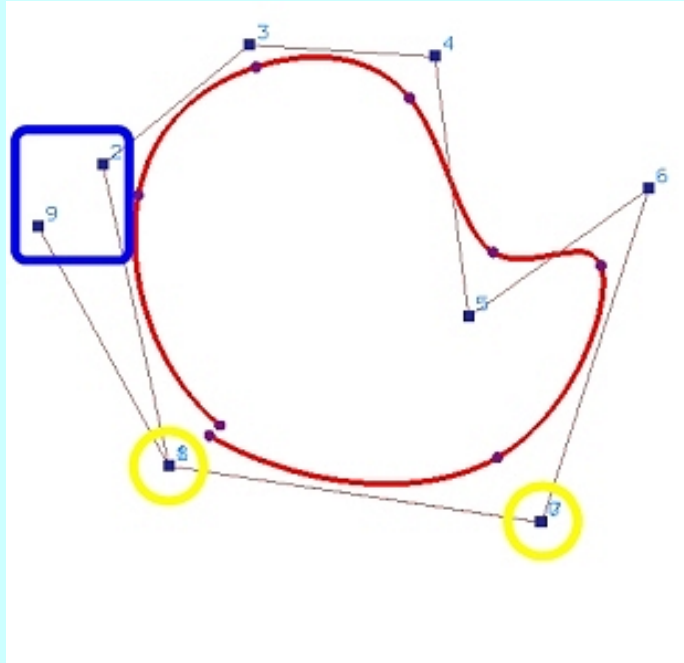
- wrapping control points
- wrapping knot vectors.

Example:

Consider an open B-spline curve of degree 3 defined by 10 ($n = 9$) control points and a uniform knot vector.



Make control points 0 and 7 identical

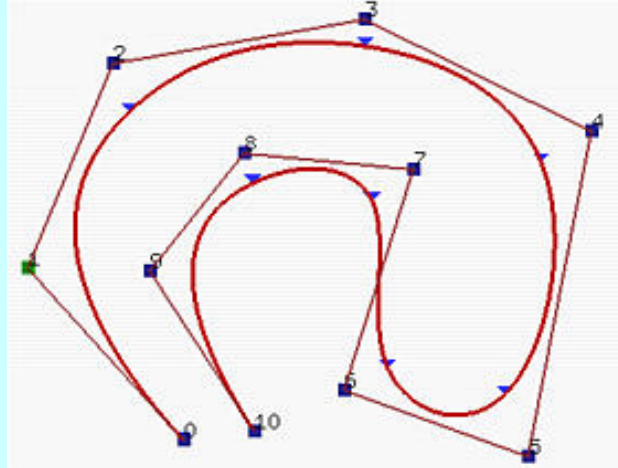


Make control points 1 and 8 identical

Make control points 2 and 9 identical

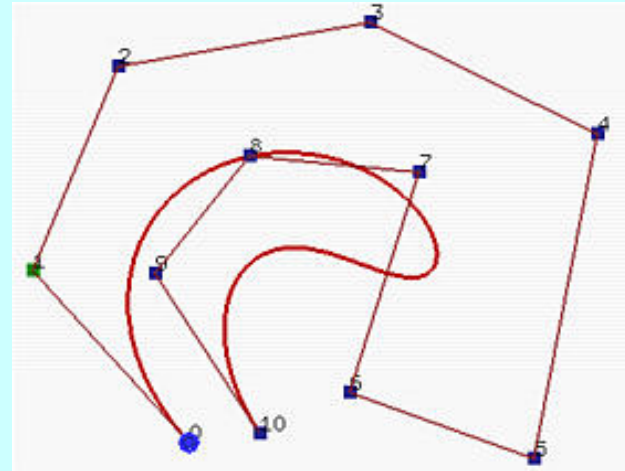
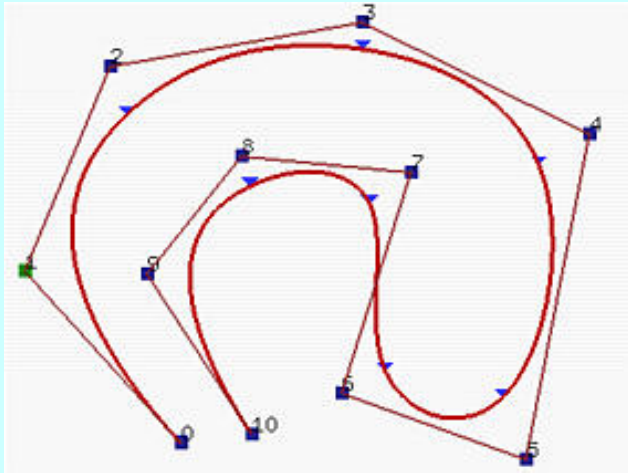
B-spline Curves: Important Properties

- B-spline curve $C(u)$ is a piecewise curve with each component a curve of degree p .



$C(u)$ can be viewed as the union of curve segments defined on each knot span.

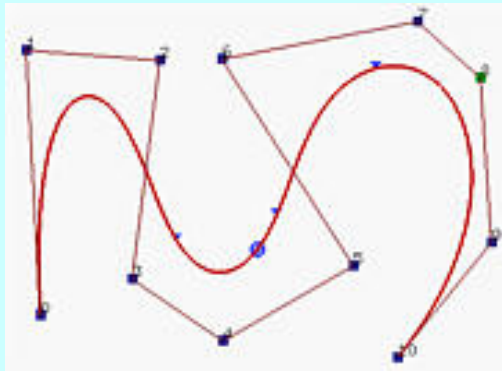
In the figure, where $n = 10$, $m = 14$ and $p = 3$, the first four knots and last four knots are clamped and the 7 internal knots are uniformly spaced. There are eight knot spans, each of which corresponds to a curve segment.



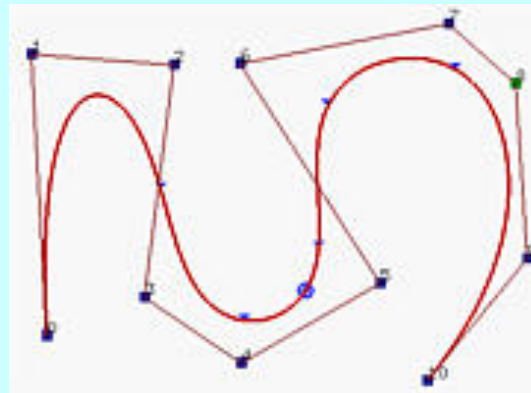
a Bézier curve with the same set of control points.

It still cannot follow the control polyline nicely even though its degree is 10!

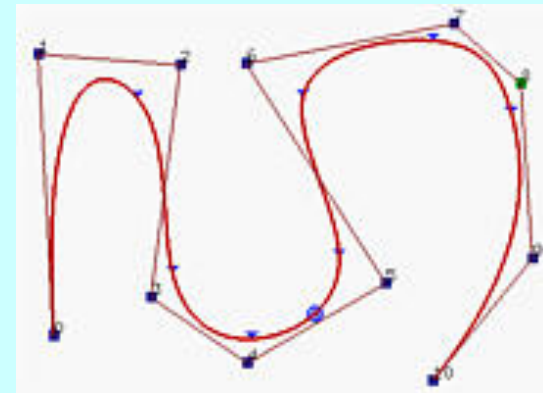
In general, the lower the degree, the closer a B-spline curve follows its control polyline.



degree 7,



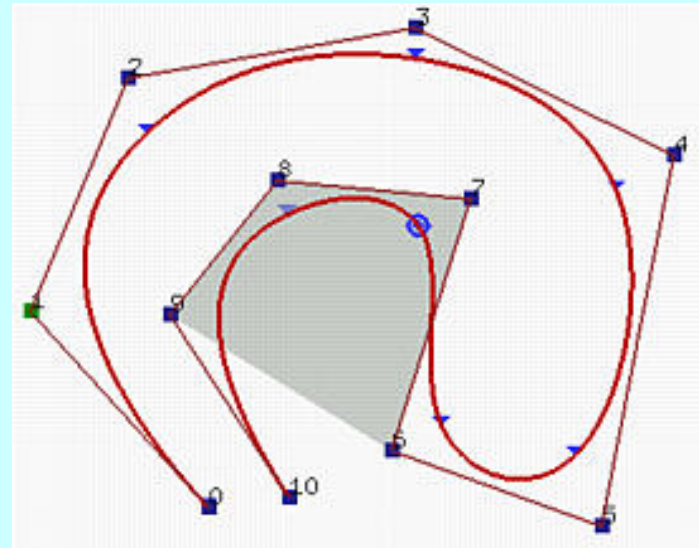
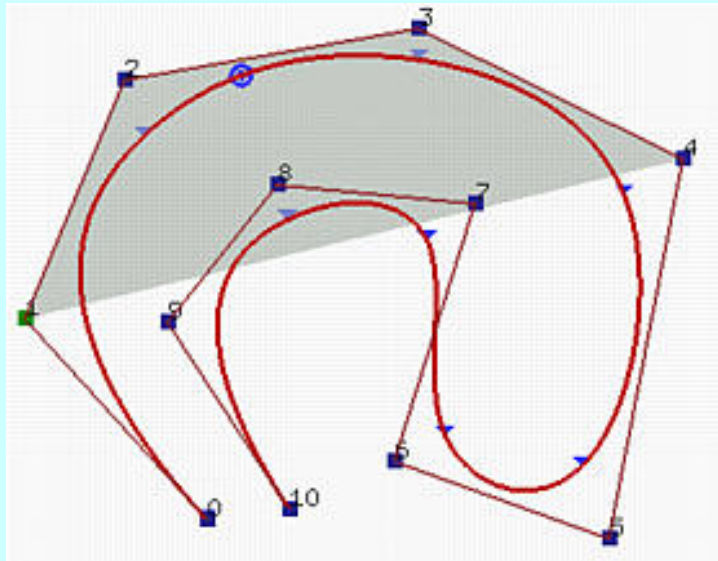
degree 5



degree 3.

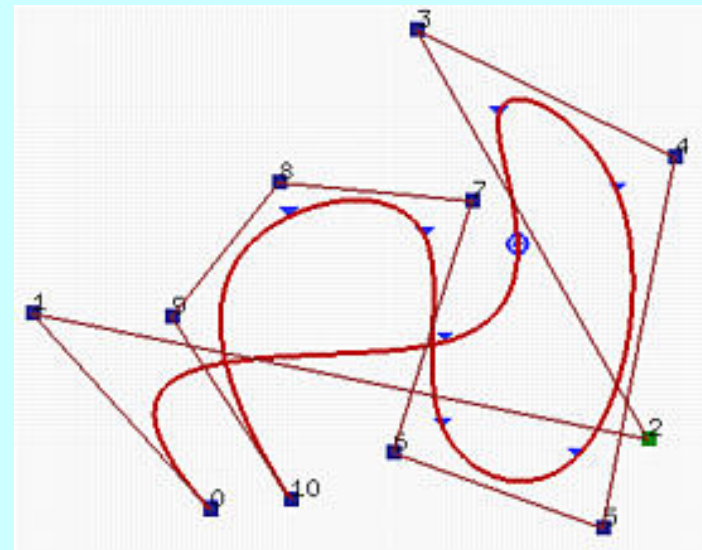
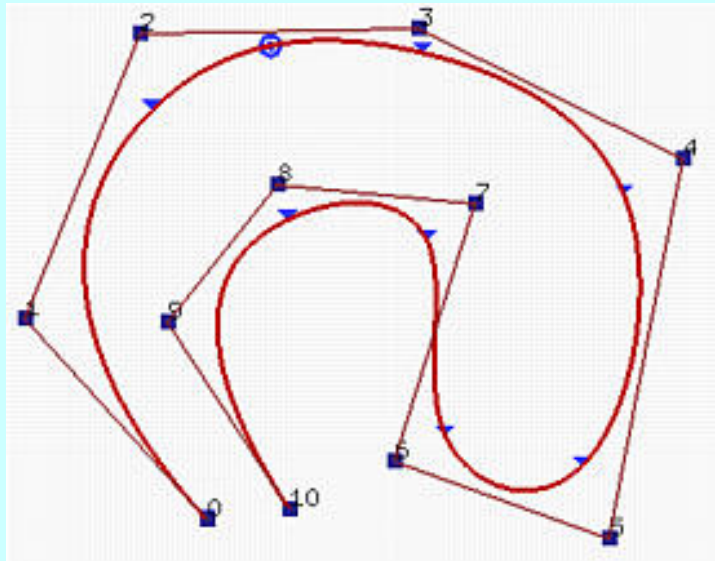
B-spline Curves: Important Properties

- Strong Convex Hull Property: A B-spline curve is contained in the convex hull of its control polyline. More specifically, if u is in knot span $[u_i, u_{i+1})$, then $C(u)$ is in the convex hull of control points $P_{i-p}, P_{i-p+1}, \dots, P_i$.



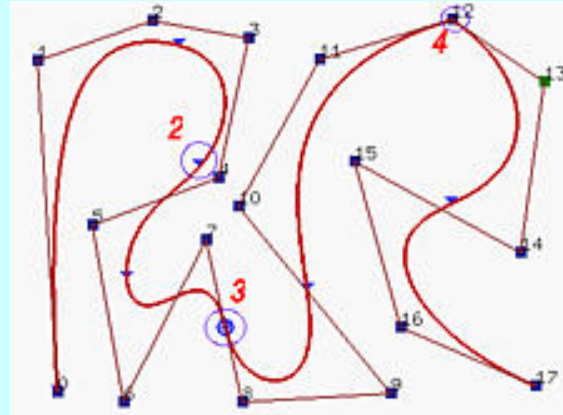
B-spline Curves: Important Properties

- Local Modification Scheme: changing the position of control point P_i only affects the curve $C(u)$ on interval $[u_i, u_{i+p+1})$.



B-spline Curves: Important Properties

- $\mathbf{C}(u)$ is C^{p-k} continuous at a knot of multiplicity k



18 control points (*i.e.*, $n = 17$), degree 4, and the clamped knot vector:

u_0 to u_4	u_5	u_6 and u_7	u_8	u_9 to u_{11}	u_{12}	u_{13} to u_{16}	u_{17}	u_{18} to u_{22}
0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1

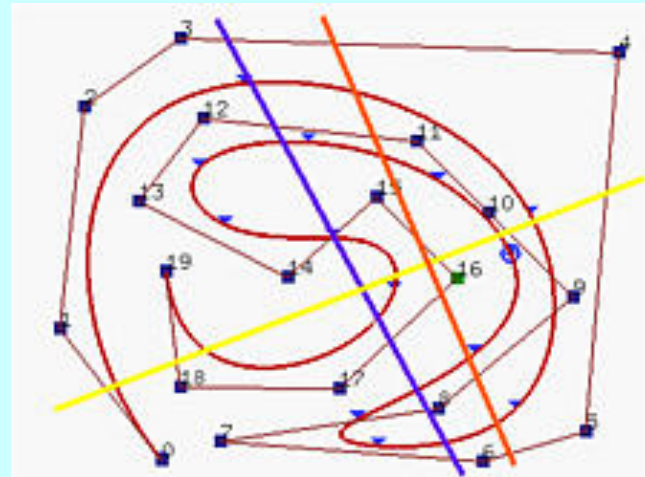
u_6 is a double knot, u_9 is a triple knot and u_{13} is a quadruple knot.

Consequently, $\mathbf{C}(u)$ is of C^4 continuous at any point that is not a knot, C^3 continuous at all simple knots, C^2 continuous at u_6 , C^1 continuous at u_9 , C^0 continuous at u_{13} .

B-spline Curves: Important Properties

- Variation Diminishing Property

No straight line intersects a B-spline curve more times than it intersects the curve's control polyline



The blue line intersects both the control polyline and the B-spline curve 6 times
the yellow line intersects the control polyline and the B-spline curve 5 times.
the orange line intersects the control polyline 6 times and the curve 4 times

B-spline Curves: Important Properties

- **Bézier Curves Are Special Cases of B-spline Curves**

If $n = p$ (*i.e.*, the degree of a B-spline curve is equal to n , the number of control points minus 1), and there are $2(p + 1) = 2(n + 1)$ knots with $p + 1$ of them clamped at each end, this B-spline curve reduces to a Bézier curve

B-spline Curves: Important Properties

- **Affine Invariance**

If an affine transformation is applied to a B-spline curve, the result can be constructed from the affine images of its control points.

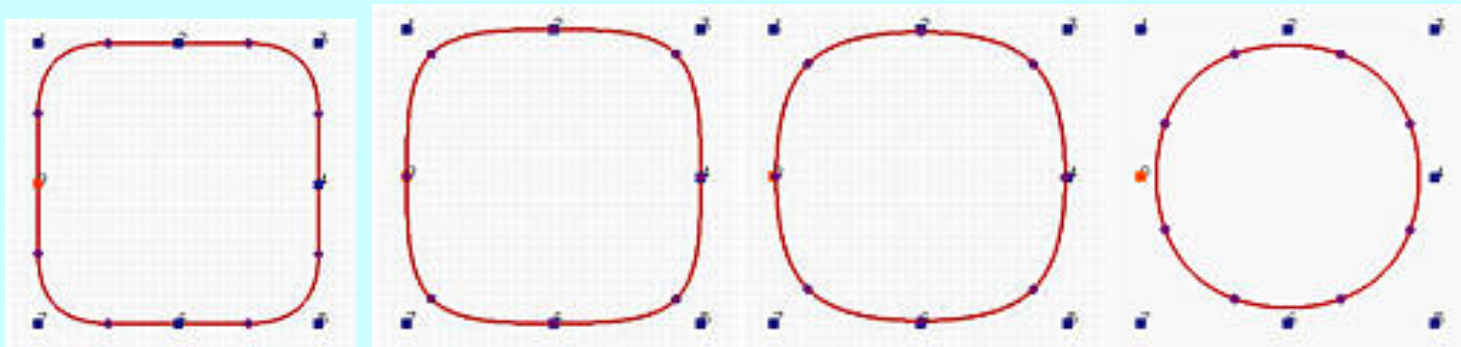
<http://www.doc.ic.ac.uk/~dfg/AndysSplineTutorial/BSplines.html>

In the case of the uniform knot sequence, the blending functions are fairly easy to calculate, are shifted versions of each other, and have support over a simple interval determined by the knots.

These characteristics are unique to the uniform blending functions.

B-spline curves are polynomial curves.

While they are flexible and have many nice properties for curve design, they are not able to represent the simplest curve: the circle.



Four closed B-spline curves with 8 control points.
The degrees, from left to right, are 2, 3, 5 and 10.

As discussed earlier, circles can only be represented with rational functions (*i.e.*, functions that are quotients of two polynomials).

To address this problem, we shall generalize B-splines to rational curves using homogeneous coordinates.

Therefore, we have the name:
NURBS- **N**on-**U**niform **R**ational **B**-**S**plines.

Given $n+1$ control points $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$ and knot vector $U = \{ u_0, u_1, \dots, u_m \}$ of $m+1$ knots, the B-spline curve of degree p defined by these parameters is:

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

Let control point \mathbf{P}_i be rewritten in homogeneous coordinates:

$$\mathbf{P}_i^w = \begin{pmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{pmatrix}$$

Plugging this new homogeneous form into the equation of the B-spline curve, we obtain :

$$C^w(u) = \sum_{i=0}^n \mathbf{P}_i^w N_{i,p}(u) = \sum_{i=0}^n N_{i,p}(u) \begin{pmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^n N_{i,p}(u)(w_i x_i) \\ \sum_{i=0}^n N_{i,p}(u)(w_i y_i) \\ \sum_{i=0}^n N_{i,p}(u)(w_i z_i) \\ \sum_{i=0}^n N_{i,p}(u)w_i \end{pmatrix}$$

Therefore, point $\mathbf{C}^w(u)$ is the original B-spline curve in homogeneous coordinate form.

Let's convert it back to Cartesian coordinate by dividing $\mathbf{C}^w(u)$ with the fourth coordinate:

$$C(u) = \begin{pmatrix} \frac{\sum_{i=0}^n N_{i,p}(u)(w_i x_i)}{\sum_{i=0}^n N_{i,p}(u)w_i} \\ \frac{\sum_{i=0}^n N_{i,p}(u)(w_i y_i)}{\sum_{i=0}^n N_{i,p}(u)w_i} \\ \frac{\sum_{i=0}^n N_{i,p}(u)(w_i z_i)}{\sum_{i=0}^n N_{i,p}(u)w_i} \\ 1 \end{pmatrix} = \sum_{i=0}^n \frac{N_{i,p}(u)w_i}{\sum_{j=0}^n N_{j,p}(u)w_j} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

This is the NURBS curve of degree p defined by control points $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$, knot vector $U = \{ u_0, u_1, \dots, u_m \}$, and weights w_0, w_1, \dots, w_n .

Two Immediate Results

- If all weights are equal to 1, a NURBS curve reduces to a B-spline curve.
- NURBS Curves are Rational

$\mathbf{C}(u)$ is a B-spline curve in the four-dimensional space.

Dividing the the first three coordinate components by the fourth one is equivalent to projecting a four-dimensional point to the plane $w = 1$.

A NURBS curve in the three-dimensional space is merely the projection of a B-spline curve in four-dimensional space.

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

where

$$R_i^p(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^n N_{j,p}(u)w_j}$$

$R_{i,p}(u)$'s are NURBS basis functions

Important Properties of NURBS Basis Functions

Since NURBS is a generalization of B-spline, it should have all properties of B-splines.

The following are some of the most important ones for NURBS basis functions.

- $R_{i,p}(u)$ is a degree p rational function in u
- Nonnegativity -- For all i and p , $R_{i,p}(u)$ is nonnegative
- Local Support -- $R_{i,p}(u)$ is a non-zero on $[u_i, u_{i+p+1})$
- On any knot span $[u_i, u_{i+1})$, at most $p+1$ degree p basis functions are non-zero, namely: $R_{i-p,p}(u)$, $R_{i-p+1,p}(u)$, $R_{i-p+2,p}(u)$, ..., and $R_{i,p}(u)$

Important Properties of NURBS Basis Functions

- Partition of Unity -- The sum of all non-zero degree p basis functions on span $[u_i, u_{i+p+1})$ is 1
- If the number of knots is $m+1$, the degree of the basis functions is p , and the number of degree p basis functions is $n+1$, then $m = n + p + 1$
- Basis function $R_{i,p}(u)$ is a composite curve of degree p rational functions with joining points at knots in $[u_i, u_{i+p+1})$
- At a knot of multiplicity k , basis function $R_{i,p}(u)$ is C^{p-k} continuous (increasing multiplicity decreases the level of continuity, and increasing degree increases continuity)
- If $w_i = c$ for all i , where c is a non-zero constant, $R_{i,p}(u) = N_{i,p}(u)$ (B-spline basis functions are special cases of NURBS basis functions when all weights become a non-zero constant)

Important Properties of NURBS Curves

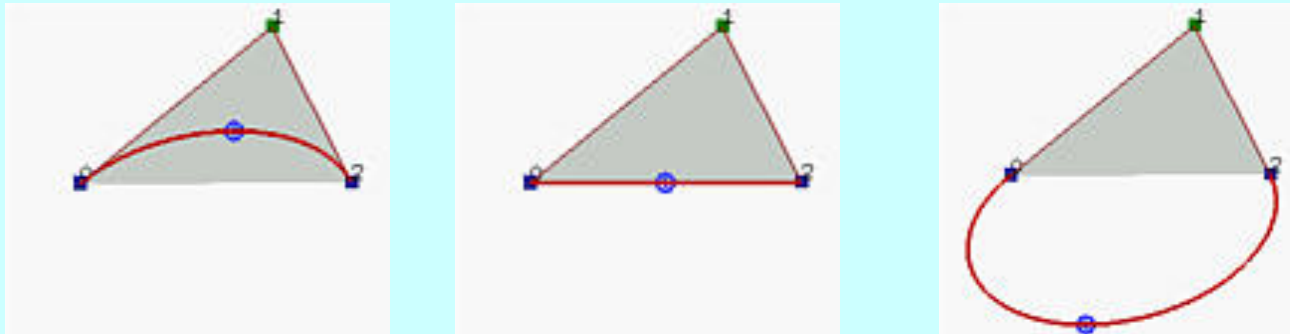
- NURBS curve $C(u)$ is a piecewise curve with each component a degree p rational curve

Actually, each component is a rational Bézier curve.

- Equality $m = n + p + 1$ must be satisfied
- A clamped NURBS curve $C(u)$ passes through the two end control points P_0 and P_n

Important Properties of NURBS Curves

- **Strong Convex Hull Property:** the NURBS curve is contained in the convex hull of its control points. Moreover, if u is in knot span $[u_i, u_{i+1})$, then $C(u)$ is in the convex hull of control points $P_{i-p}, P_{i-p+1}, \dots, P_i$



The left figure is a NURBS curve of degree 2 with $n = 2$, $m = 5$ and the first three and last three knots clamped. The weights of the two control points at both ends are 1's and the weight

of the middle control point is 0.5. This is actually an elliptic arc. The curve segment lies in

the convex hull. The middle figure has the weight of the middle control point set to zero. Since this control point has no effect, the result is the line segment determined by the endpoints. It still lies in the convex hull.

If the weight is changed to -0.5, the curve segment is not contained in the convex hull and

hence the convex hull property fails.

Important Properties of NURBS Curves

- **Local Modification Scheme:** changing the position of control point P_i only affects the curve $C(u)$ on interval $[u_i, u_{i+p+1})$
- $C(u)$ is C^{p-k} continuous at a knot of multiplicity k
- **Variation Diminishing Property**
- **B-spline Curves and Bézier Curves Are Special Cases of NURBS Curves**
- **Projective Invariance** (If a projective transformation is applied to a NURBS curve, the result can be constructed from the projective images of its control points.)

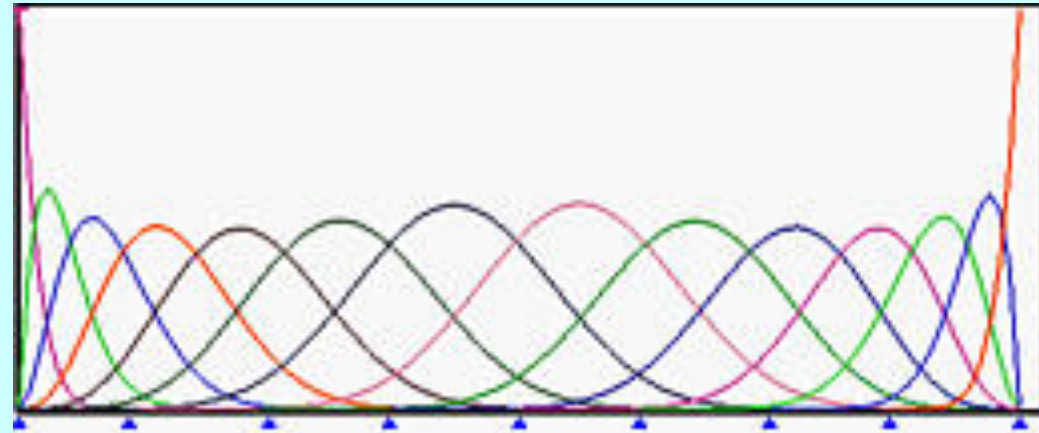
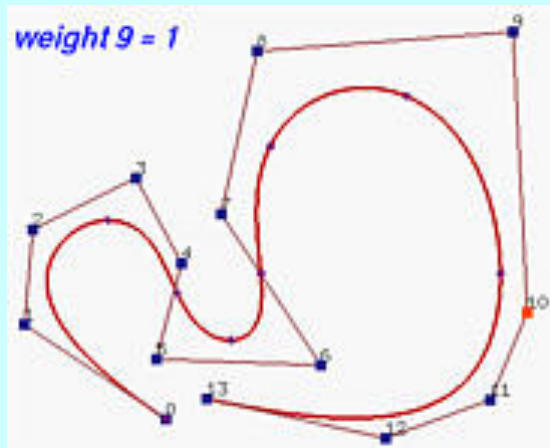
Since NURBS curves are defined by a set of control points, a knot vector, a degree and a set of weights, we have one more parameter for shape modification (*i.e.*, the weights).

Recall that the basis functions of a NURBS curve is:

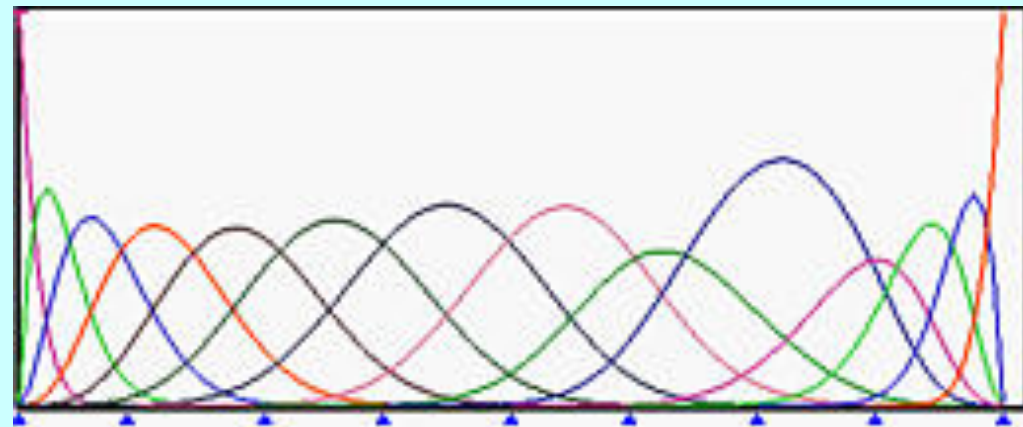
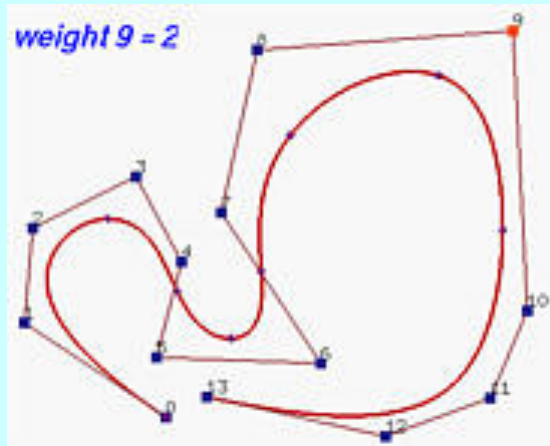
$$R_i^p(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^n N_{j,p}(u)w_j}$$

Therefore, increasing and decreasing the value of w_i will increase and decrease the value of $R_{i,p}(u)$, respectively. More precisely, increasing the value of w_i will pull the curve toward control point \mathbf{P}_i . In fact, all affected points on the curve will also be pulled in the direction to \mathbf{P}_i . When w_i approaches infinity, the curve will pass through control point \mathbf{P}_i .

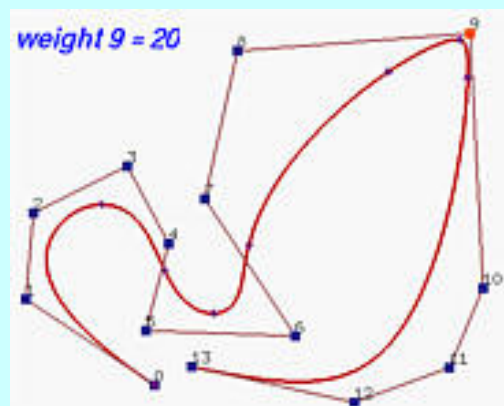
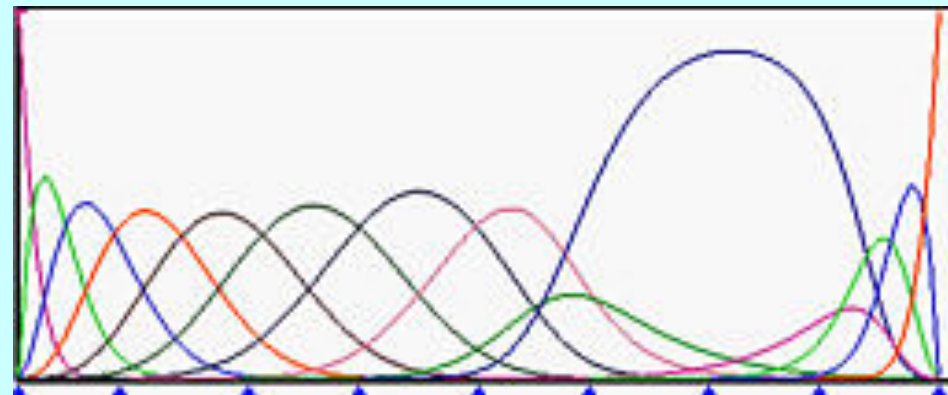
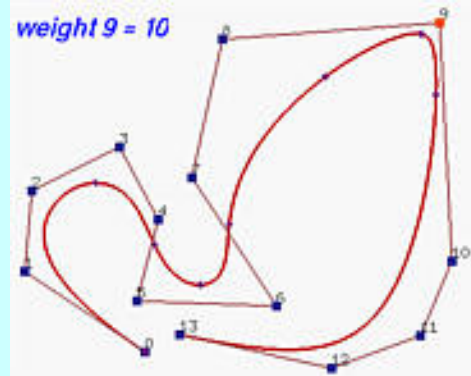
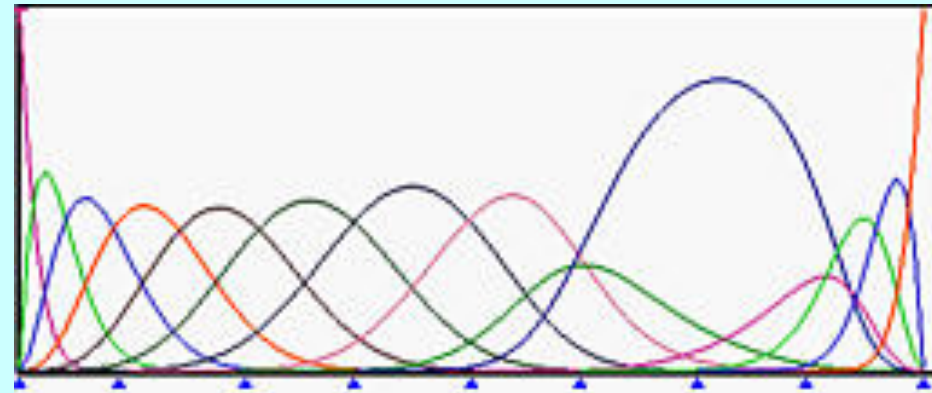
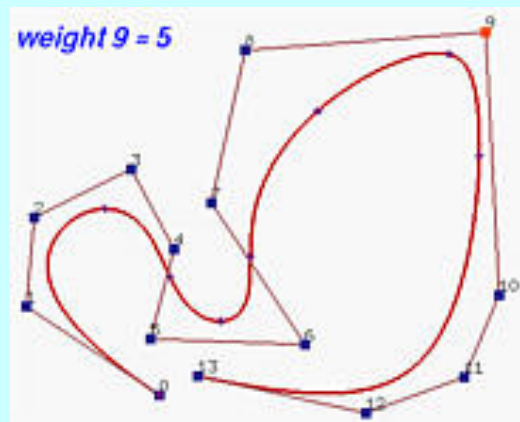
NURBS curve of degree 6 and its NURBS basis functions

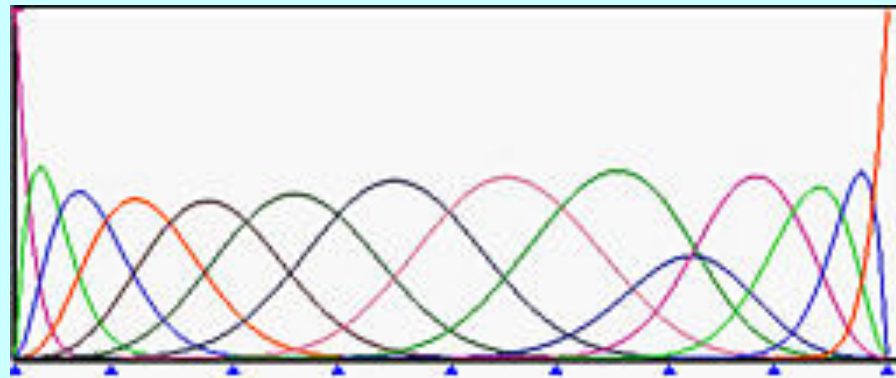
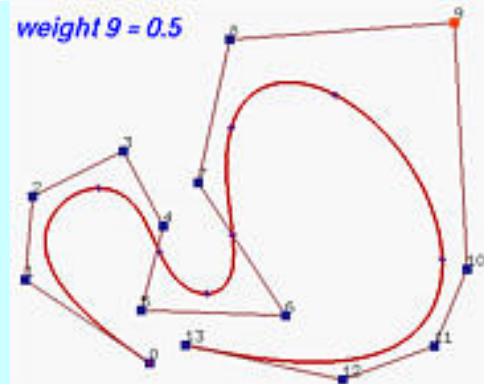
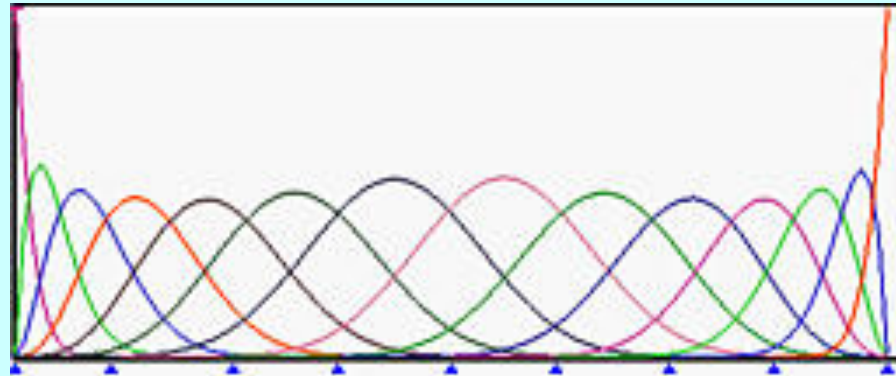
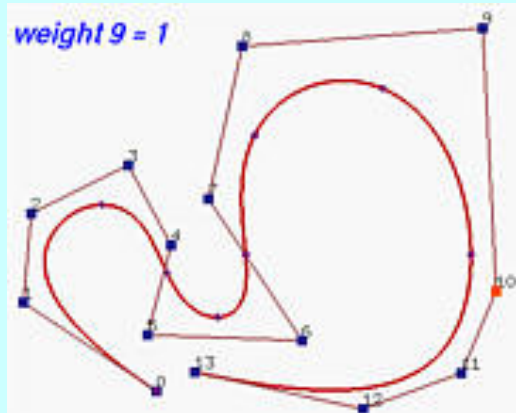


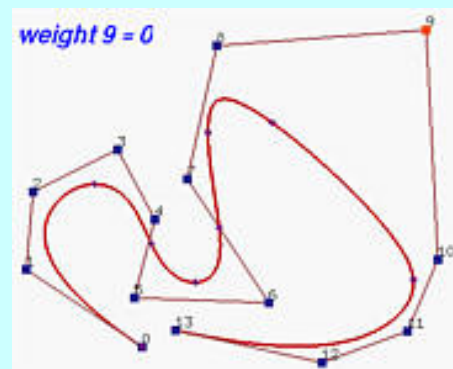
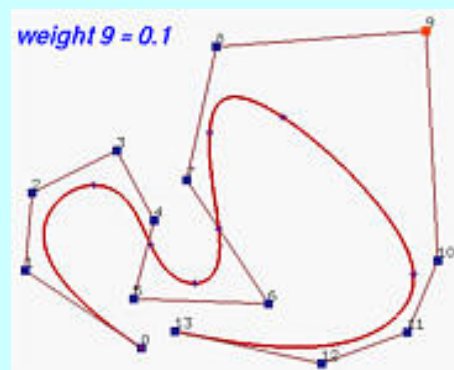
all weights are 1's and the curve is a B-spline curve



w_9 is increased to 2 and, a portion of the curve moves toward P_9





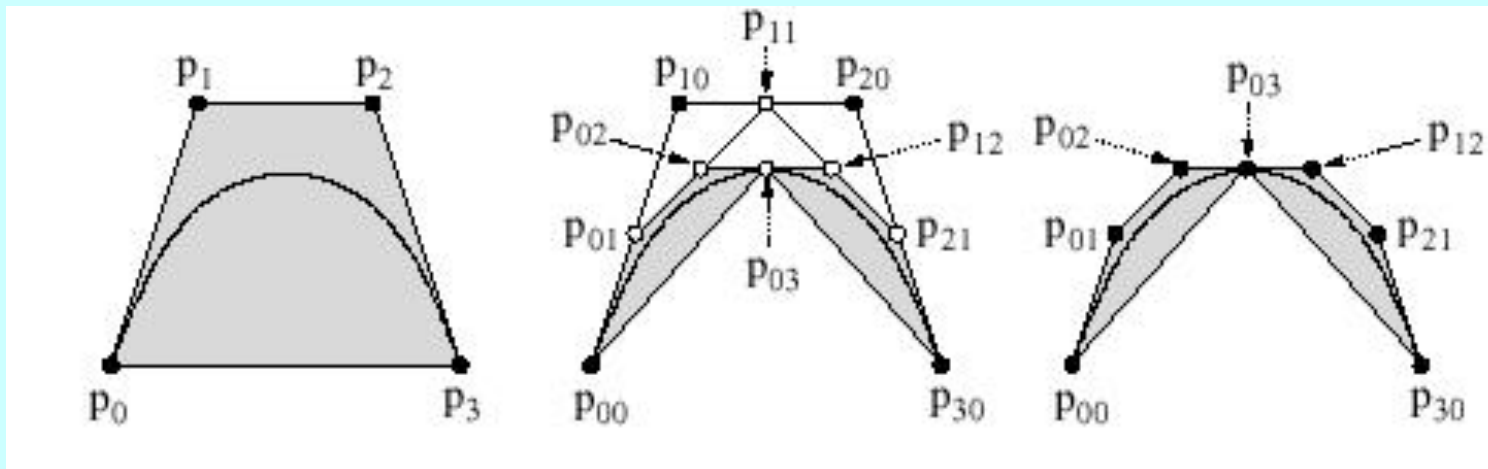


Subdividing Bezier Curves

- OpenGL renders flat objects
- To render curves, approximate by small linear segments
- Subdivide curved surface to polygonal patches
- Bezier curves useful for elegant, recursive subdivision
- May have different levels of recursion for different parts of curve or surface
- Example: may subdivide visible surfaces more than hidden surfaces

Subdividing Bezier Curves

- Let $(P_0 \dots P_3)$ denote original sequence of control points
- Relabel these points as $(P_{00} \dots P_{30})$
- Repeat interpolation ($u = \frac{1}{2}$) and label vertices as below
- Sequences $(P_{00}, P_{01}, P_{02}, P_{03})$ and $(P_{03}, P_{12}, P_{21}, P_{30})$ define Bezier curves also
- Bezier Curves can either be straightened or curved recursively in this way



Bezier Surfaces

- Bezier surfaces: interpolate in two dimensions
- This called Bilinear interpolation
- Example: 4 control points, P_{00} , P_{01} , P_{10} , P_{11} , 2 parameters u and v
- Interpolate between
 - P_{00} and P_{01} using u
 - P_{10} and P_{11} using u
 - Repeat two steps above using v

$$p(u, v) = (1 - v)((1 - u)p_{00} + up_{01}) + v((1 - u)p_{10} + up_{11})$$

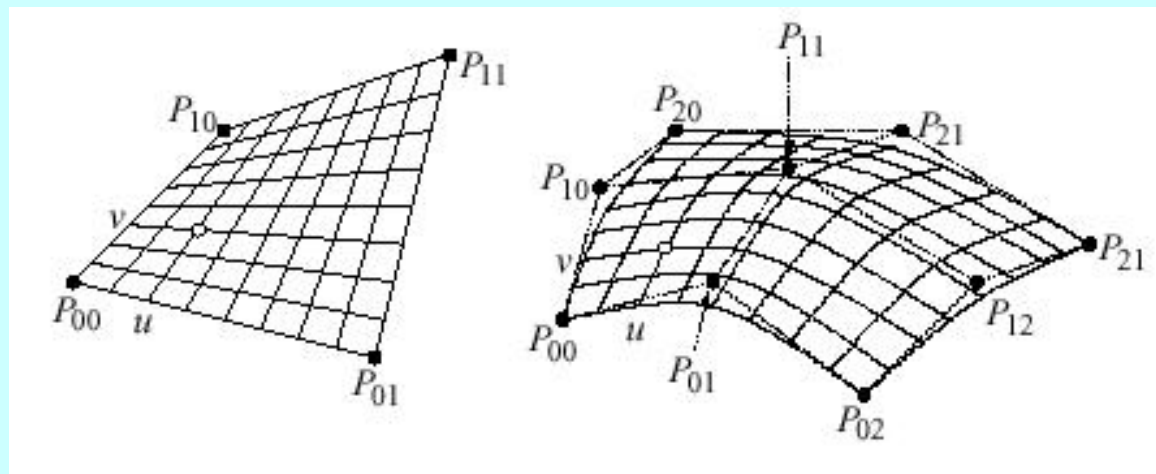
Bezier Surfaces

- Recalling, $(1-u)$ and u are first-degree Bezier blending functions $b_{0,1}(u)$ and $b_{1,1}(u)$

$$p(u, v) = b_{0,1}(v)b_{0,1}(u)p_{00} + b_{0,1}(v)b_{1,1}(u)p_{01} + b_{1,1}(v)b_{0,1}(u)p_{10} + b_{1,1}(v)b_{1,1}(u)p_{11}$$

Generalizing for cubic

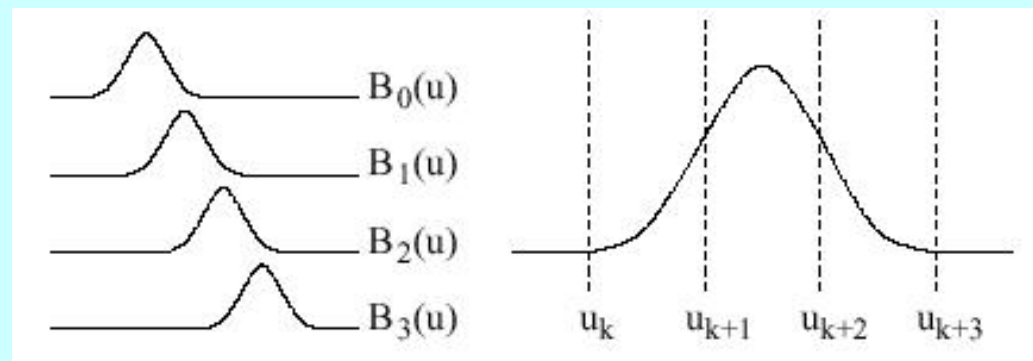
$$p(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 b_{i,3}(v)b_{j,3}(u)p_{i,j}$$



B-Splines

- Bezier curves are elegant but too many control points
- Smoother = more control points = higher order polynomial
- Undesirable: every control point contributes to all parts of curve
- B-splines designed to address Bezier shortcomings
- Smooth blending functions, each non-zero over small range
- Use different polynomial in each range, (**piecewise polynomial**)

$$p(u) = \sum_{i=0}^m B_i(u) p_i$$



B-spline blending functions, order 2

NURBS

- Encompasses both Bezier curves/surfaces and B-splines
- Non-uniform Rational B-splines (NURBS)
- Rational function is ratio of two polynomials
- NURBS use rational blending functions
- Some curves can be expressed as rational functions but not as simple polynomials
- No known exact polynomial for circle
- Rational parametrization of unit circle on xy-plane:

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

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

$$z(u) = 0$$

NURBS

- We can apply homogeneous coordinates to bring in w

$$x(u) = 1 - u^2$$

$$y(u) = 2u$$

$$z(u) = 0$$

$$w(u) = 1 + u^2$$

- Using w , we get we cleanly integrate rational parametrization
- Useful property of NURBS: preserved under transformation
- Thus, we can project control points and then render NURBS