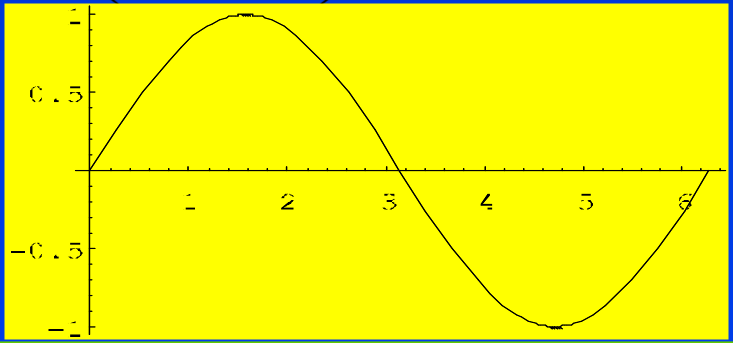
CSE328 Fundamentals of Computer Graphics: Concepts, Theory, Algorithms, and Applications

Hong Qin
Department of Computer Science
State University of New York at Stony Brook (Stony Brook University)
Stony Brook, New York 11794--4400
Tel: (631)632-8450; Fax: (631)632-8334
qin@cs.sunysb.edu
http://www.cs.sunysb.edu/~qin

Explicit Representation

- Consider one example: a function $f(\theta) = sin(\theta)$.
- This is the explicit description of a curve in 2 dimensions with parameter θ.
- This is an example of an unbounded curve (in that we can take values of θ from -∞…+∞. We'll limit our curve to the domain (0…2π). This gives the following curve:



Explicit Representation

- We are used to seeing an equation of a curve defined by expressing one variable as a function of the other

$$y = x^3$$

$$y = \sqrt{4 - x^2}$$

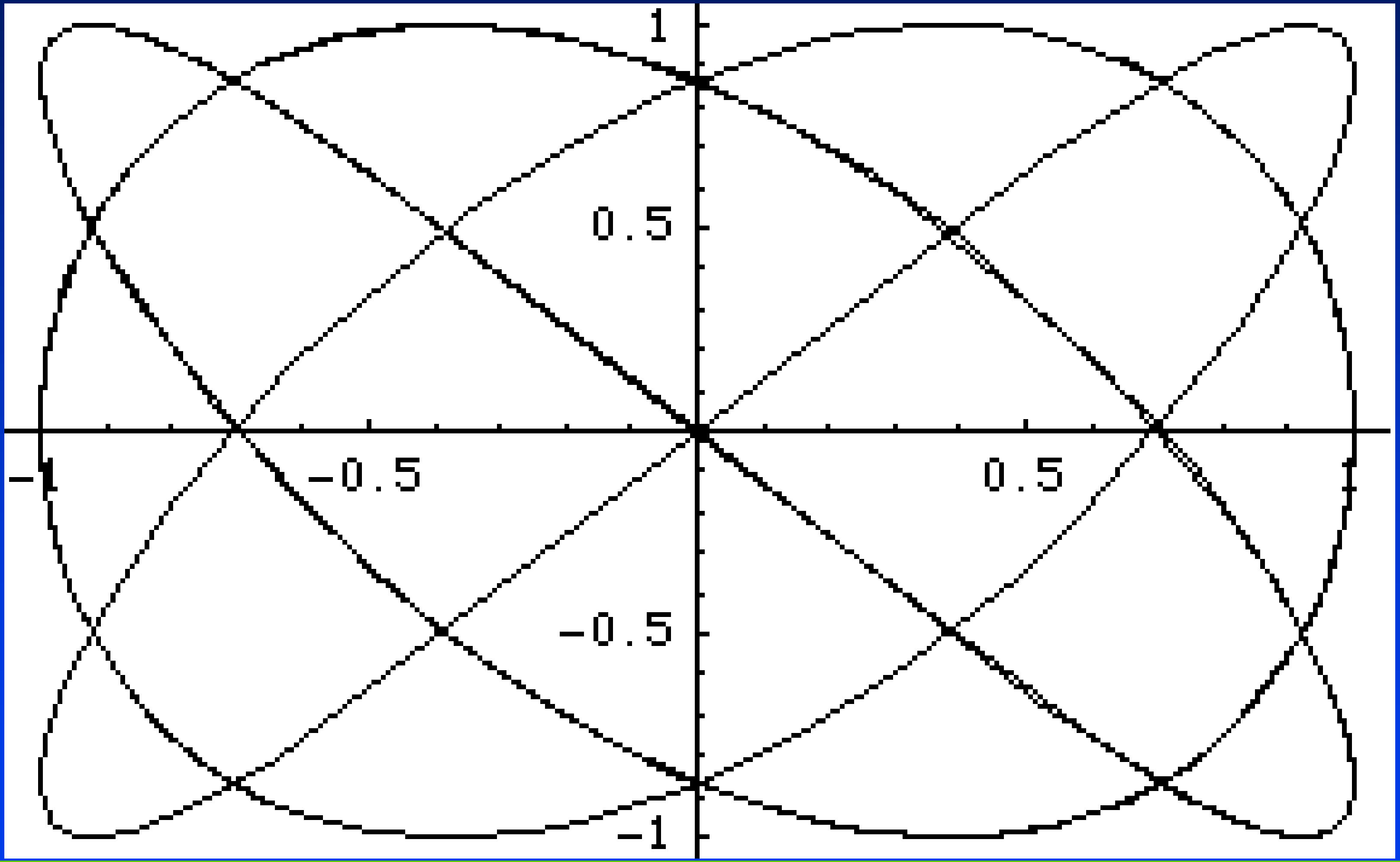
$$y = f(x)$$

Parametric Curves

Parametric Representations

- We are going to start the topic of parametric representation, especially for curves and surfaces
- But first, let us look at the concept of explicit, non-parametric representation

Parametric Representations



Parametric Representations

- The geometric and physical intuition: a *parameter* is a third, independent variable (for example, time).
- By introducing a parameter, x and y can be expressed as a function of the parameter, as opposed to functions of each other.
  - For example, **F**(t) = <f(t), g(t)>, where x= f(t) and y= g(t)

    F(t) = <cos(t), sin(t)> - what is this curve and why is this parameterization useful?

Parametric Representations

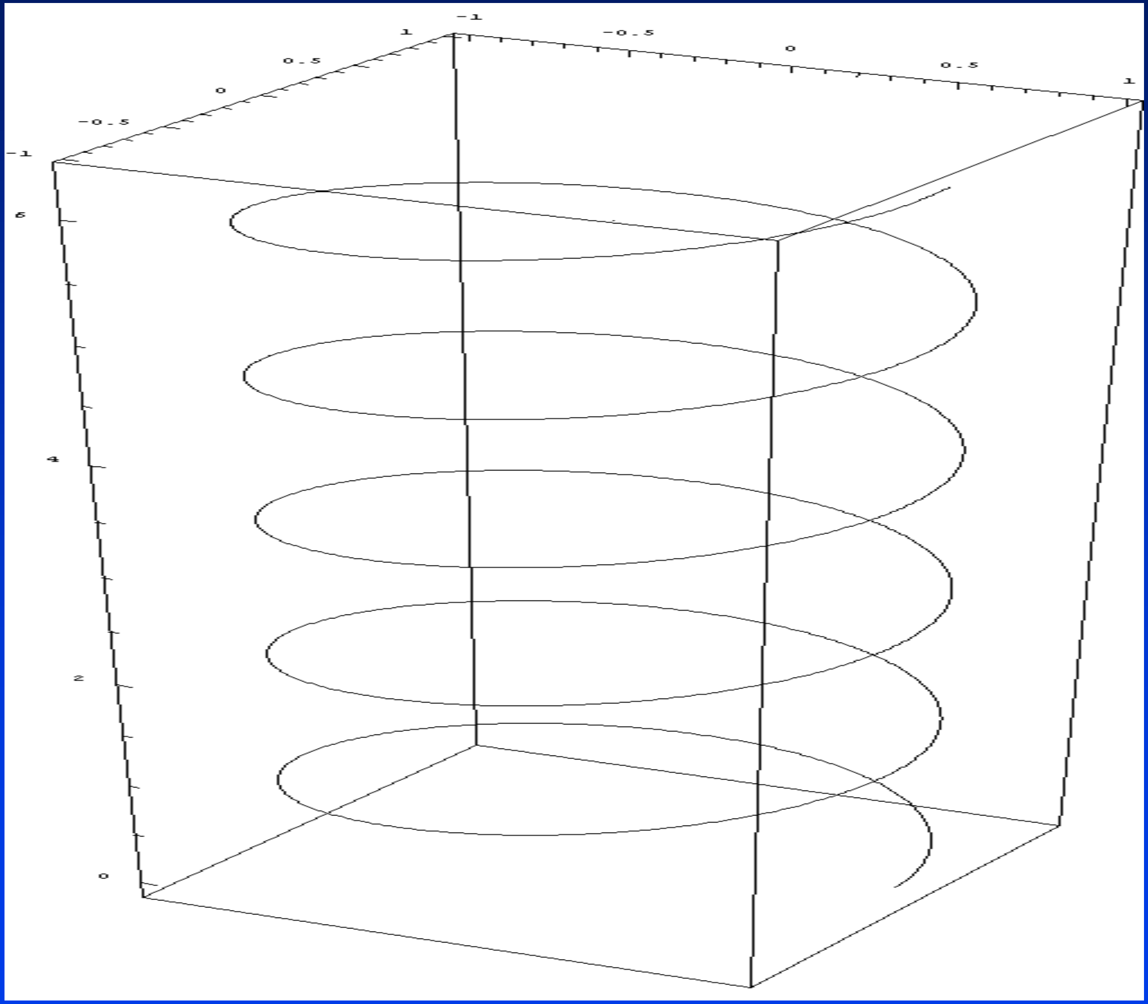
- Each value of the parameter t determines a point, (f(t), g(t)), and the set of all points comprises the graph of the curve.
- Complicated curves are easily dealt with since the components f(t) and g(t) each becomes a function.
  - For example, **F**(t)=<sin(3t), sin(4t)>
- From parametric representation to explicit representation --- Sometimes the parameter can be eliminated by solving one equation (say, x=f(t)) for the parameter t and substituting this expression into the other equation y=g(t). The result will be the parametric curve.

Properties and Visualization

- A conceptual example:
  - Picture the xy-plane to be on the table and the z-axis coming straight up out of the table
  - Picture the parameterized 2-D path (cos(t), sin(t)) which is a circle on the table
  - Add a simple z-component such that the circle climbs off the table to form a helix (or corkscrew), z=t
- Mathematically:
  - Add a simple linear term in the z-direction:

$$\mathbf{F}(t)=\langle \cos(t), \sin(t), t \rangle$$

Visualization



Parametric Curves

- Please remember to make comparisons between parametric representations and the following equations:
  - Explicit representation:
    - $y = f(x)$
  - Implicit representation:
    - $f(x,y) = 0$

Parametric Curves

- Please remember to make comparisons between parametric representations and the following equations:
  - Explicit representation:
    - $y = f(x)$
  - Implicit representation:
    - $f(x,y) = 0$

Parametric Curves

- Why use parametric curves?
  - Why curves (rather than polylines)?
    - reduce the number of points
    - interactive manipulation is easier
  - Why parametric (as opposed to y,z=f(x))?
    - arbitrary curves can be easily represented
    - rotational invariance
  - Why parametric (rather than implicit)?
    - simplicity and efficiency

Line (Geometric Line)

- Parametric representation

$$\mathbf{l}(\mathbf{p}_0, \mathbf{p}_1) = \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u$$
$$u \in [0,1]$$

- Parametric representation is not unique

$$\mathbf{l}(\mathbf{p}_0, \mathbf{p}_1) = \mathbf{0.5}(\mathbf{p}_1 + \mathbf{p}_0) + \mathbf{0.5}(\mathbf{p}_1 - \mathbf{p}_0)v$$
$$v \in [-1,1]$$

- In general

$$\mathbf{p}(u),$$
$$u \in [a,b]$$

- Re-parameterization (variable transformation)

$$v = (u-a)/(b-a)$$
$$u = (b-a)v + a$$
$$\mathbf{q}(v) = \mathbf{p}((b-a)v + a)$$
$$v \in [0,1]$$

Basic Concepts

- Linear interpolation: $\mathbf{v} = \mathbf{v}_0(1-t) + \mathbf{v}_1(t)$

$\mathbf{v} \in [\mathbf{v}_0, \mathbf{v}_1], t \in [0,1]$

- Local coordinates:
- Re-parameterization: $f(u), u = g(v), f(g(v)) = h(v)$
- Affine transformation:

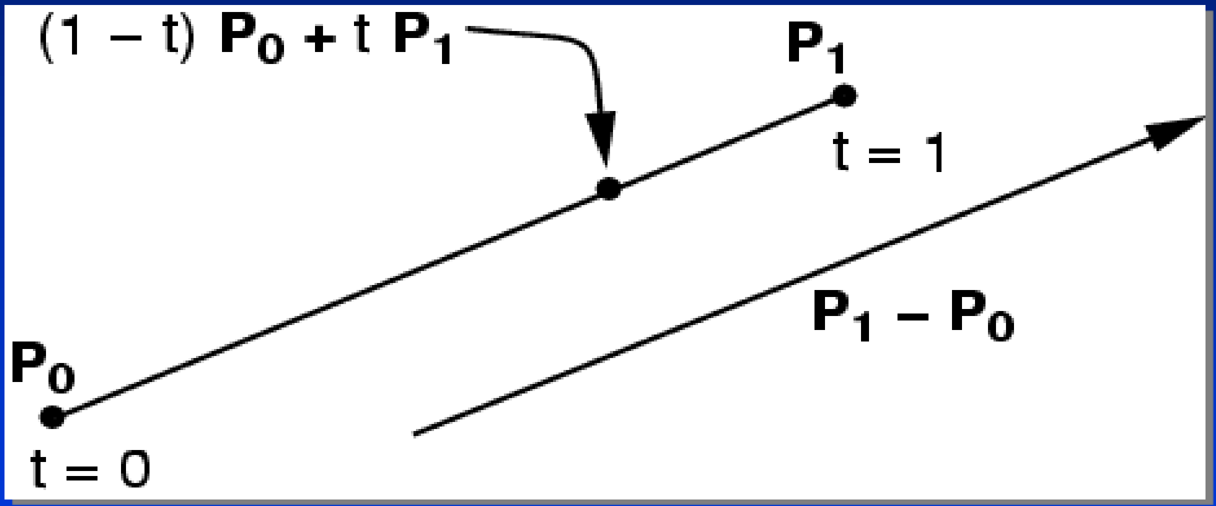
$$f(ax+by) = af(x) + bf(y)$$

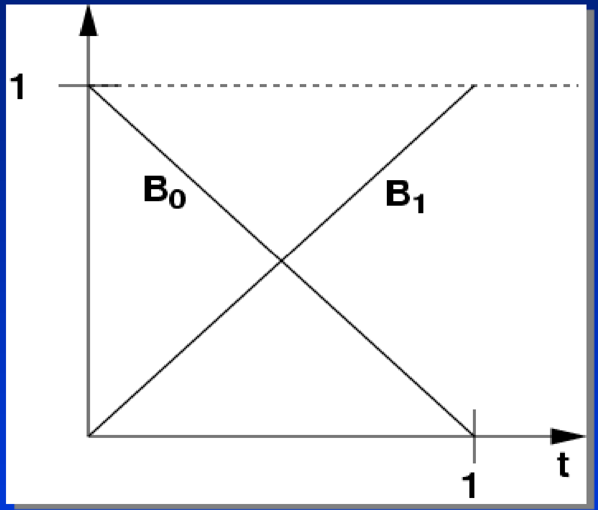
$$a + b = 1$$

- Polynomials
- Continuity

Linear Interpolation

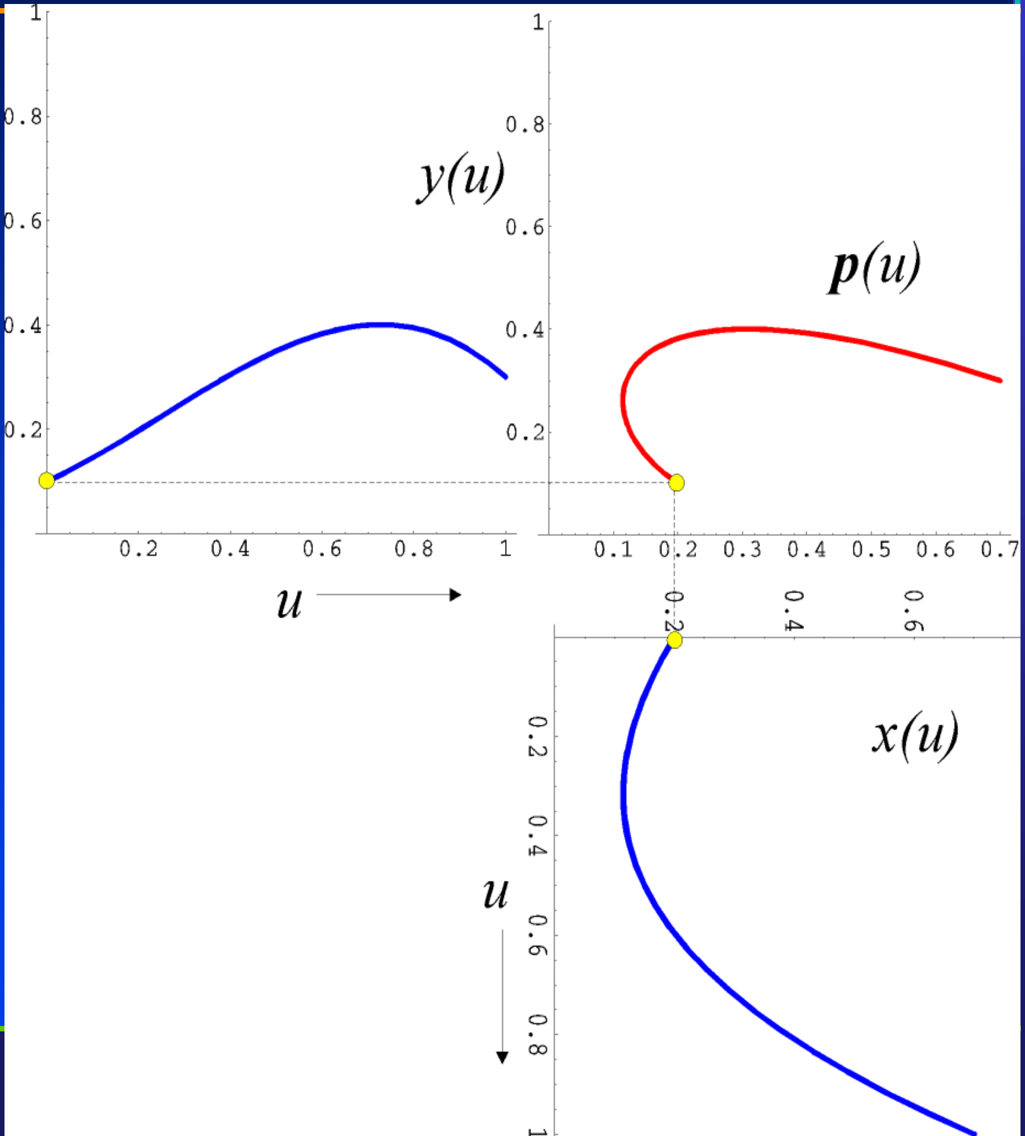
- Simplest "curve" between two points





$$x(t) = g_{1x}(1-t) + g_{2x}(t),$$
$$y(t) = g_{1y}(1-t) + g_{2y}(t),$$
$$z(t) = g_{1z}(1-t) + g_{2z}(t).$$

Parameterization: The Basic Concept



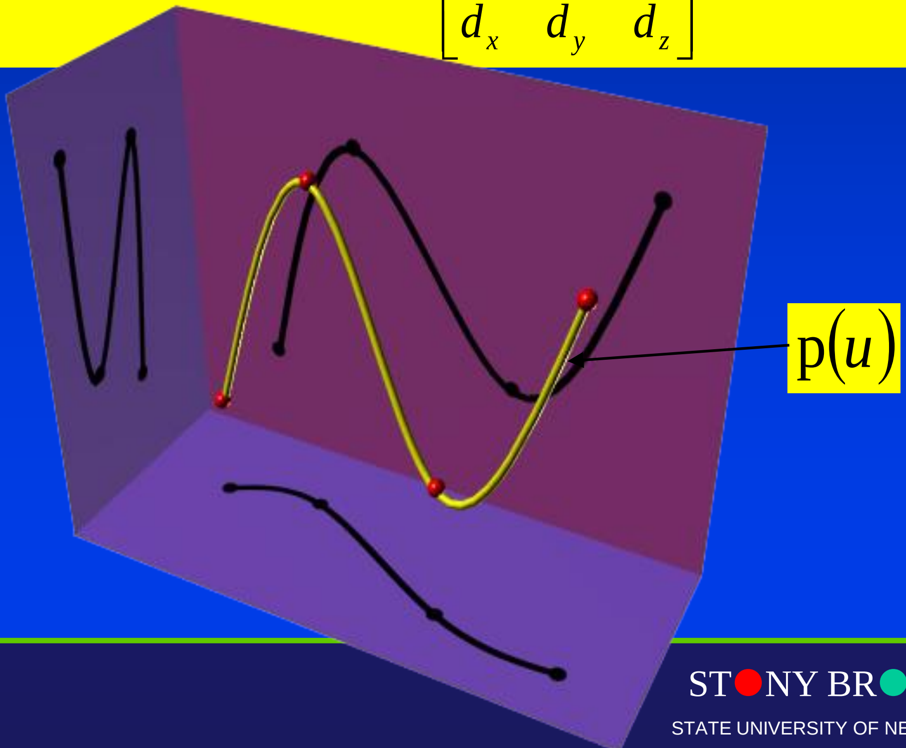
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$

Splines

- For a 3D spline, we have 3 polynomials:

$$\left.\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z \end{aligned}\right\} \rightarrow \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \rightarrow \mathbf{p}(u) = \mathbf{u}.\mathbf{C}$$

12 unknowns
4 3D points required



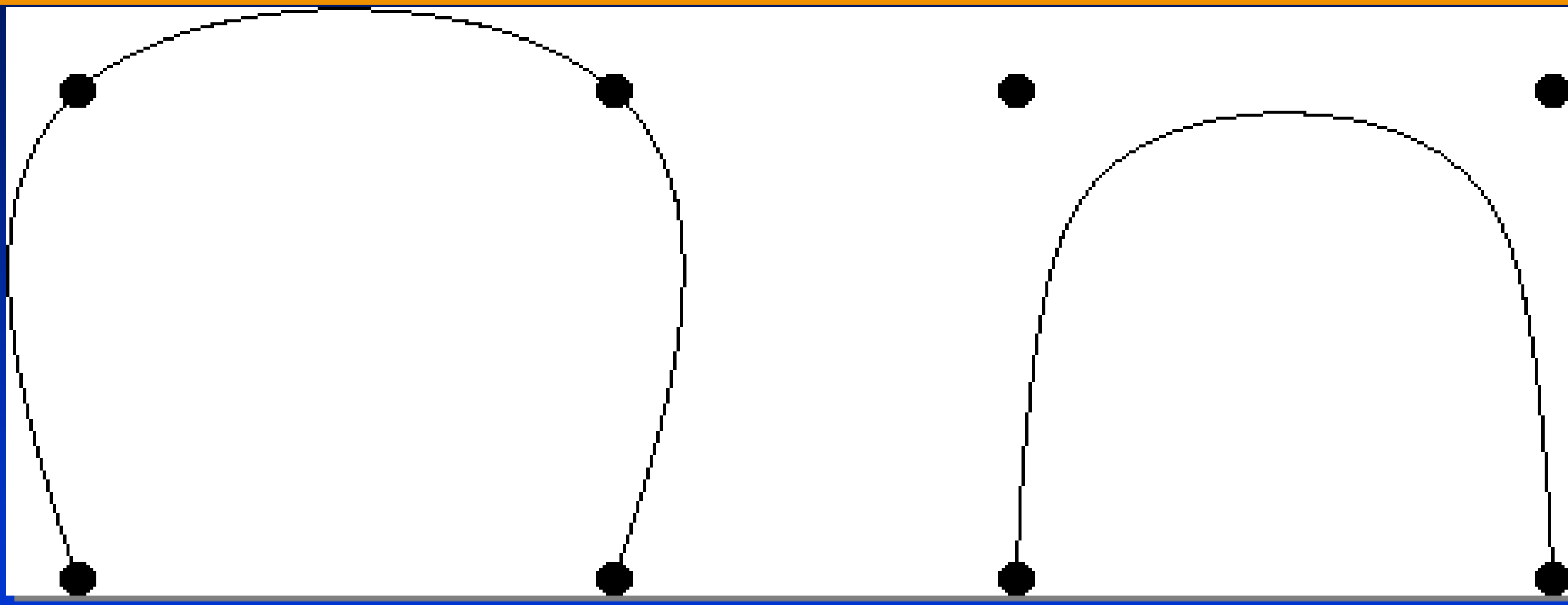
Defines the variation in x with distance u along the curve

Department of Computer Science
Center for Visual Computing
STONY BROOK
STATE UNIVERSITY OF NEW YORK

Parametric Cubic Curves

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z, \qquad 0 \le t \le 1.$$

Interpolation vs. Approximation Curves



Interpolation

curve must pass
through control points

Approximation

curve is influenced
by control points

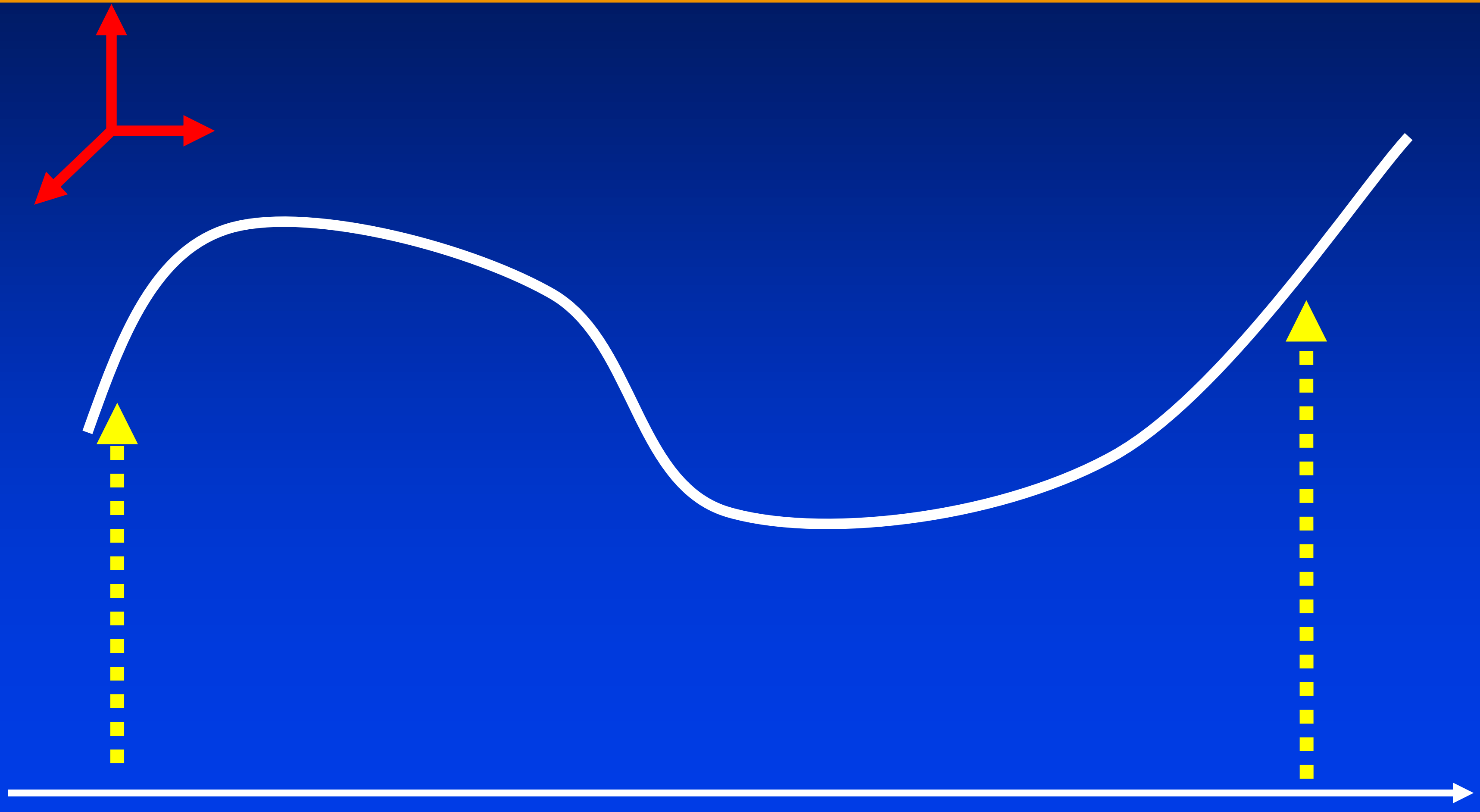
Parametric Polynomials

- High-order polynomials

$$\mathbf{c}(u) = \begin{bmatrix} \mathbf{a}_{0,x} \\ \mathbf{a}_{0,y} \\ \mathbf{a}_{0,z} \end{bmatrix} + \ldots + \begin{bmatrix} \mathbf{a}_{i,x} \\ \mathbf{a}_{i,y} \\ \mathbf{a}_{i,z} \end{bmatrix} u^i + \ldots + \begin{bmatrix} \mathbf{a}_{n,x} \\ \mathbf{a}_{n,y} \\ \mathbf{a}_{n,z} \end{bmatrix} u^n$$

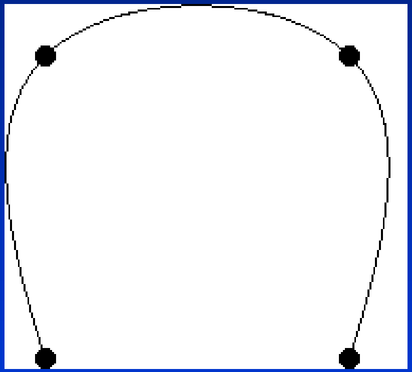
- No intuitive insight for the curved shape
- Difficult for piecewise smooth curves

Parametric Polynomials

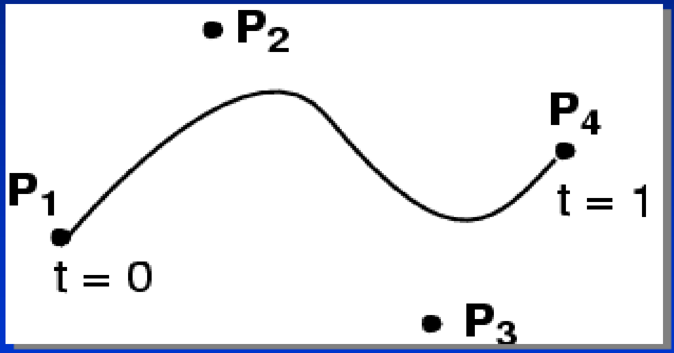


Definition: What's a Spline?

- Smooth curve defined by some control points
- Moving the control points changes the curve

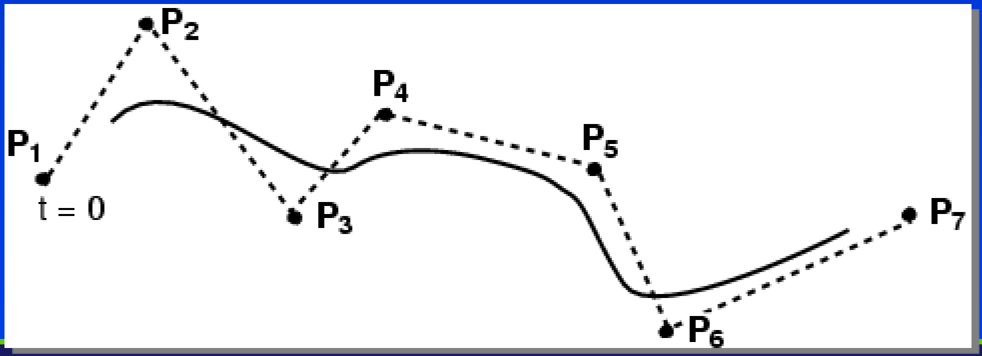


Interpolation

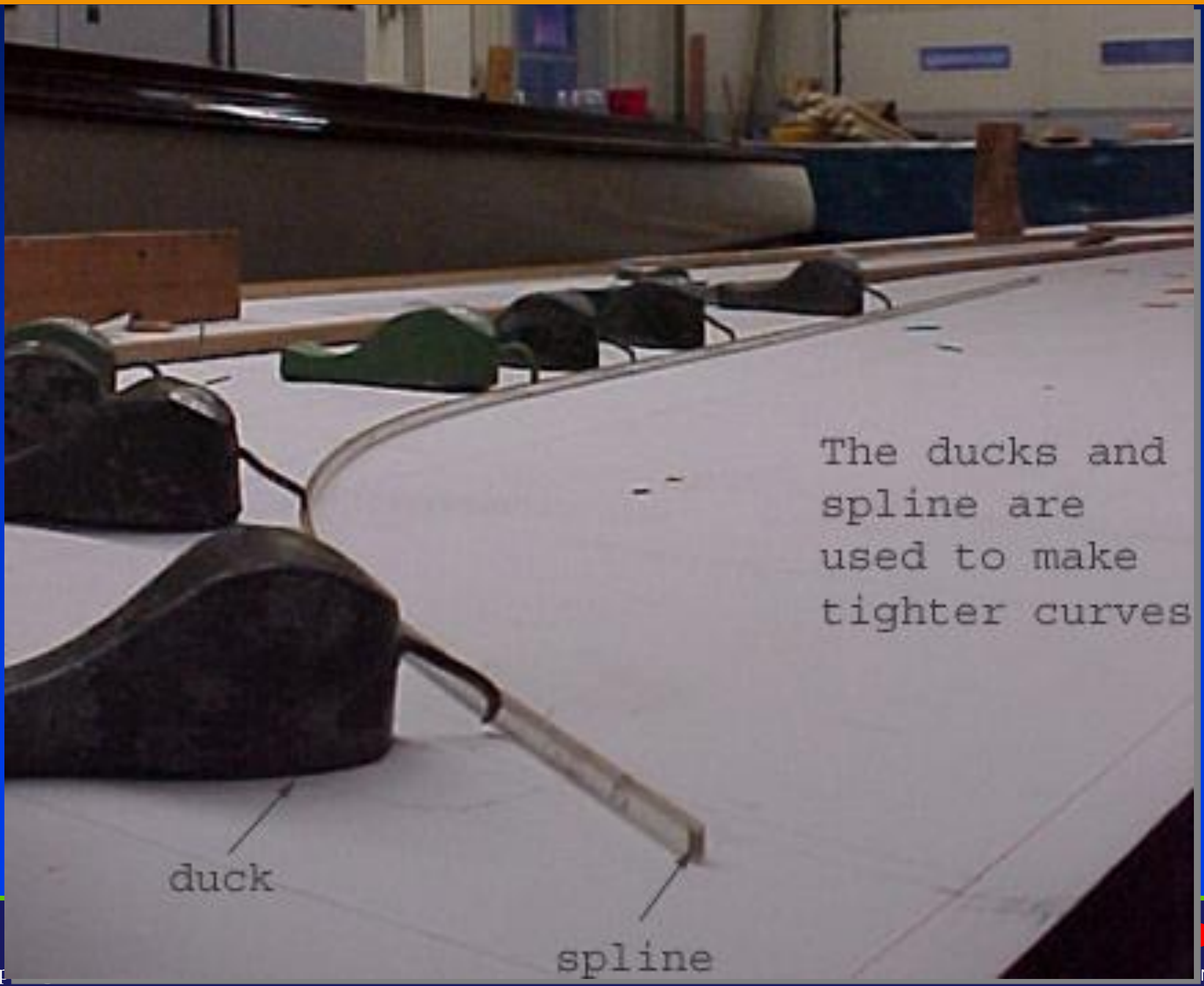


Bézier (approximation)

BSpline (approximation)

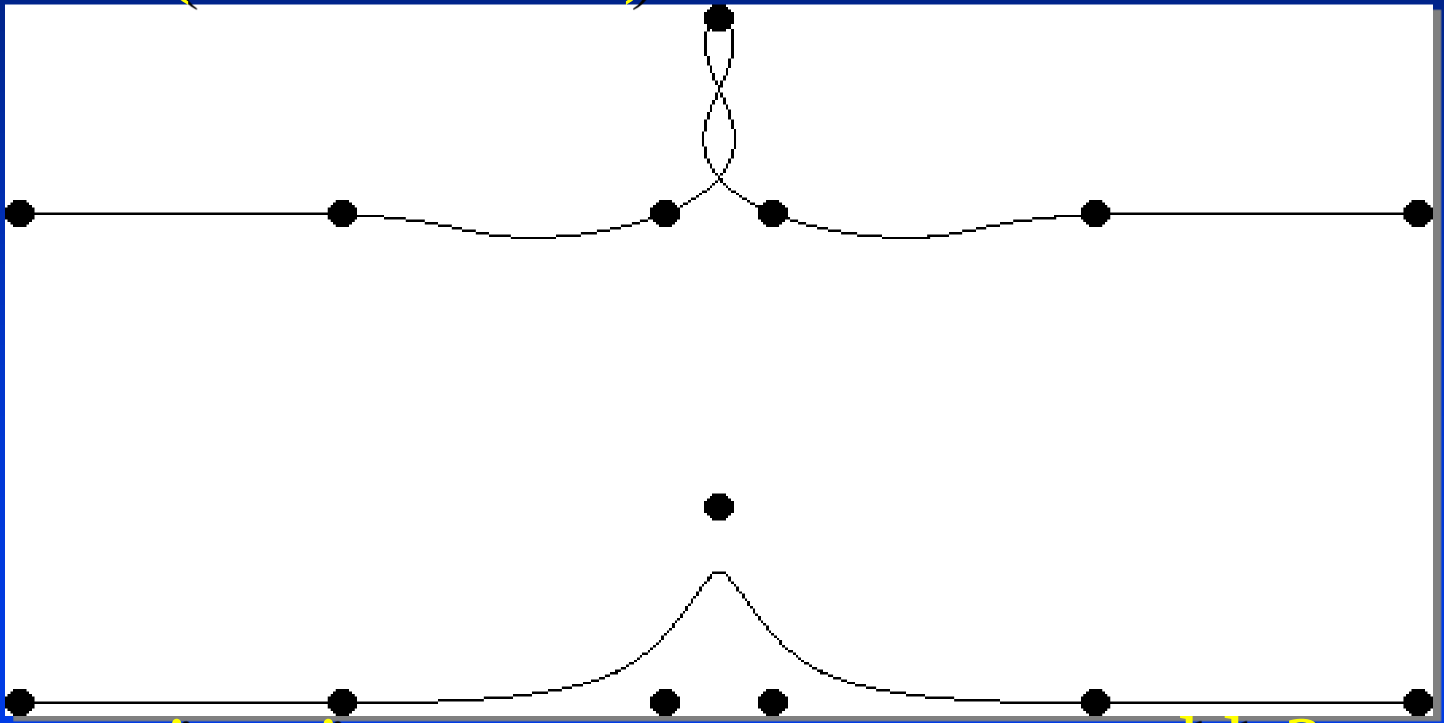


Interpolation Curves / Splines (Prior to the Digital Representation)



Interpolation vs. Approximation Curves

- Interpolation curve – over constrained → lots of (undesirable?) oscillations



- Approximation curve – more reasonable?

STONY BROOK
STATE UNIVERSITY OF NEW YORK

Interpolating Splines: Applications

- Idea: Use **key frames** to indicate a series of positions that must be “hit”
- For example:
  - Camera location
  - Path for character to follow
  - Animation of walking, gesturing, or facial expressions
    - Morphing
- Use splines for smooth interpolation

How to Define a Curve?

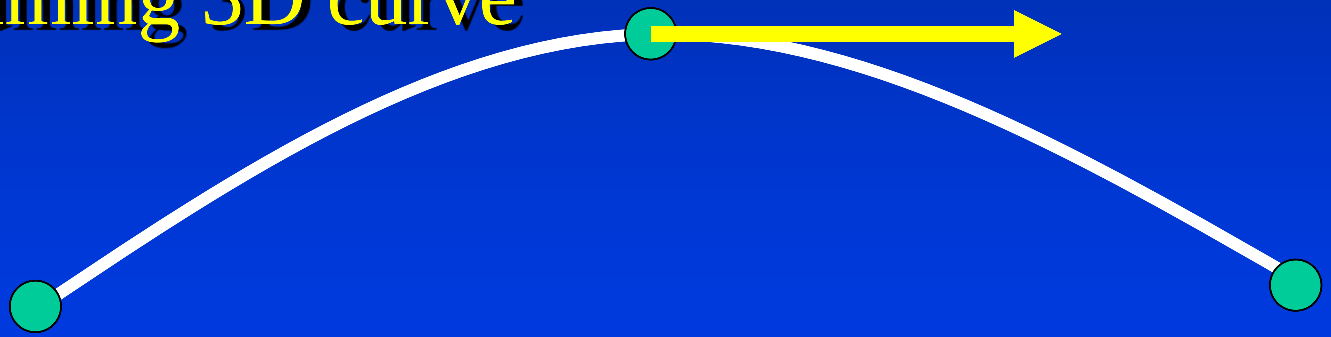
- Specify a set of points for interpolation and/or approximation with fixed or unfixed parameterization

$$\begin{bmatrix} x(u_i) \\ y(u_i) \\ z(u_i) \end{bmatrix} \qquad \begin{bmatrix} x'(u_i) \\ y'(u_i) \\ z'(u_i) \end{bmatrix}$$

- Specify the derivatives at some locations
- What is the geometric meaning to specify derivatives?
- A set of constraints
- Solve constraint equations

One Example

- Two end-vertices: c(0) and c(1)
- One mid-point: c(0.5)
- Tangent at the mid-point: c'(0.5)
- Assuming 3D curve



Cubic Polynomials

- Parametric representation (u is in [0,1])

$$\begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \begin{bmatrix} a_3 \\ b_3 \\ c_3 \end{bmatrix} u^3 + \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} u^2 + \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} u + \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$$

- Each components are treated independently
- High-dimension curves can be easily defined
- Alternatively

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \end{bmatrix}^T = UA$$

$$y(u) = UB$$

$$z(u) = UC$$

Cubic Polynomial Example

- Constraints: two end-points, one mid-point, and tangent at the mid-point

$$x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A$$
$$x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5^1 & 1 \end{bmatrix} A$$
$$x'(0.5) = \begin{bmatrix} 3(0.5)^2 & 2(0.5) & 1 & 0 \end{bmatrix} A$$
$$x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A$$

- In matrix form

$$\begin{bmatrix} x(0) \\ x(0.5) \\ x'(0.5) \\ x(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.125 & 0.25 & 0.5 & 1 \\ 0.75 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} A$$

Solve this Linear Equation

- Invert the Matrix

$$A = \begin{bmatrix} -4 & 0 & -4 & 4 \\ 8 & -4 & 6 & -4 \\ -5 & 4 & -2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(0.5) \\ x'(0.5) \\ x(1) \end{bmatrix}$$

- Rewrite the curve expression

$$x(u) = UM\begin{bmatrix} x(0) & x(0.5) & x'(0.5) & x(1) \end{bmatrix}^T$$

$$y(u) = UM\begin{bmatrix} y(0) & y(0.5) & y'(0.5) & y(1) \end{bmatrix}^T$$

$$z(u) = UM\begin{bmatrix} z(0) & z(0.5) & z'(0.5) & z(1) \end{bmatrix}^T$$

Basis Functions

- Special polynomials

$$f_1(u) = -4u^3 + 8u^2 - 5u + 1$$
$$f_2(u) = -4u^2 + 4u$$
$$f_3(u) = -4u^3 + 6u^2 - 2u$$
$$f_4(u) = 4u^3 - 4u^2 + 1$$

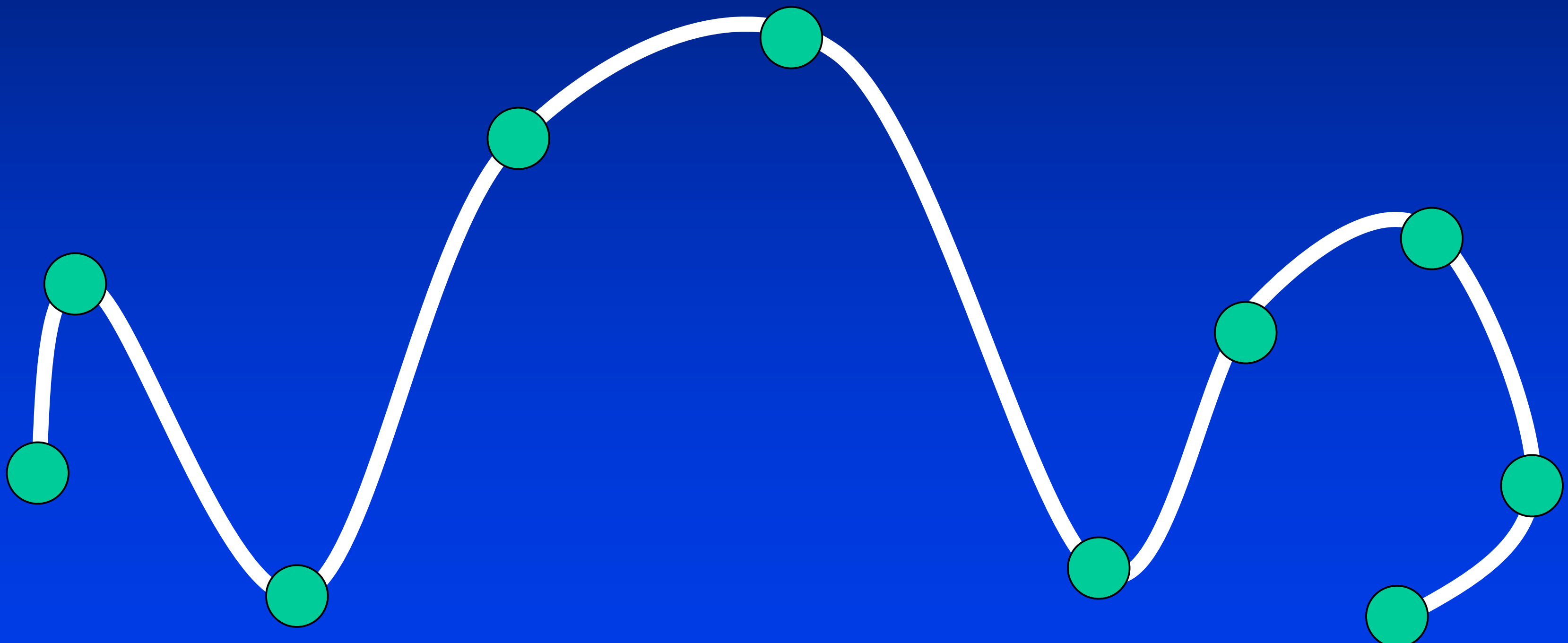
- What is the image of these basis functions?
- Polynomial curve can be defined by

$$\mathbf{c}(u) = \mathbf{c}(0) f_1(u) + \mathbf{c}(0.5) f_2(u) + \mathbf{c}'(0.5) f_3(u) + \mathbf{c}(1) f_4(u)$$

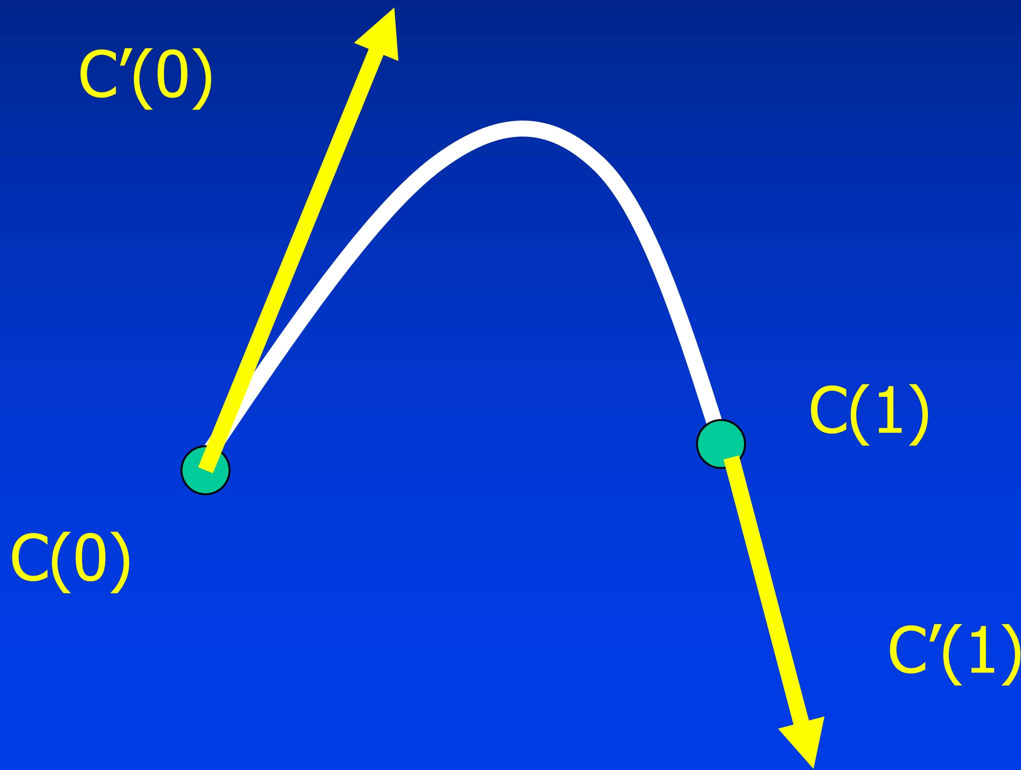
- Observations
  - More intuitive, easy to control, polynomials

Lagrange Curve

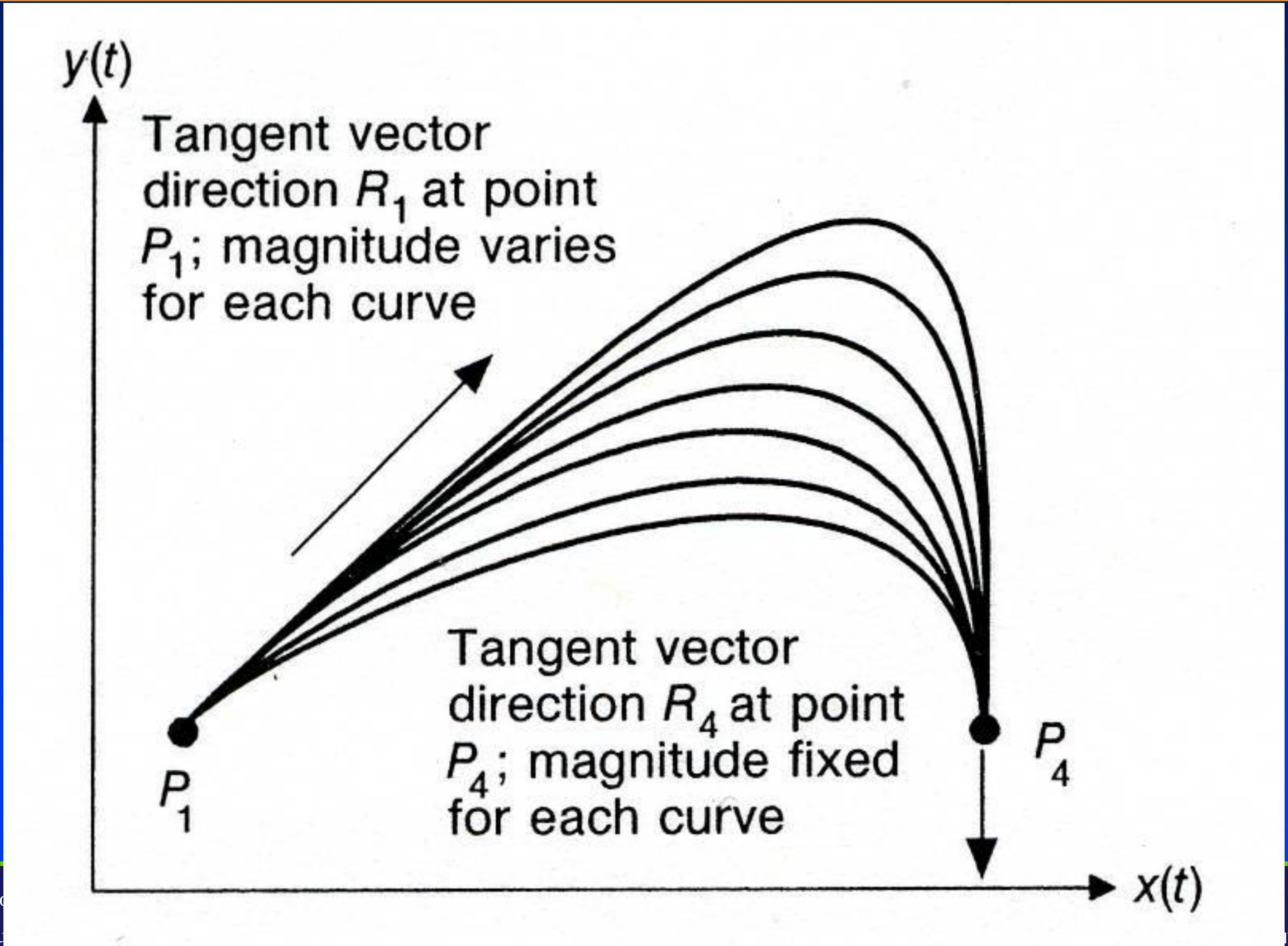
- Point interpolation



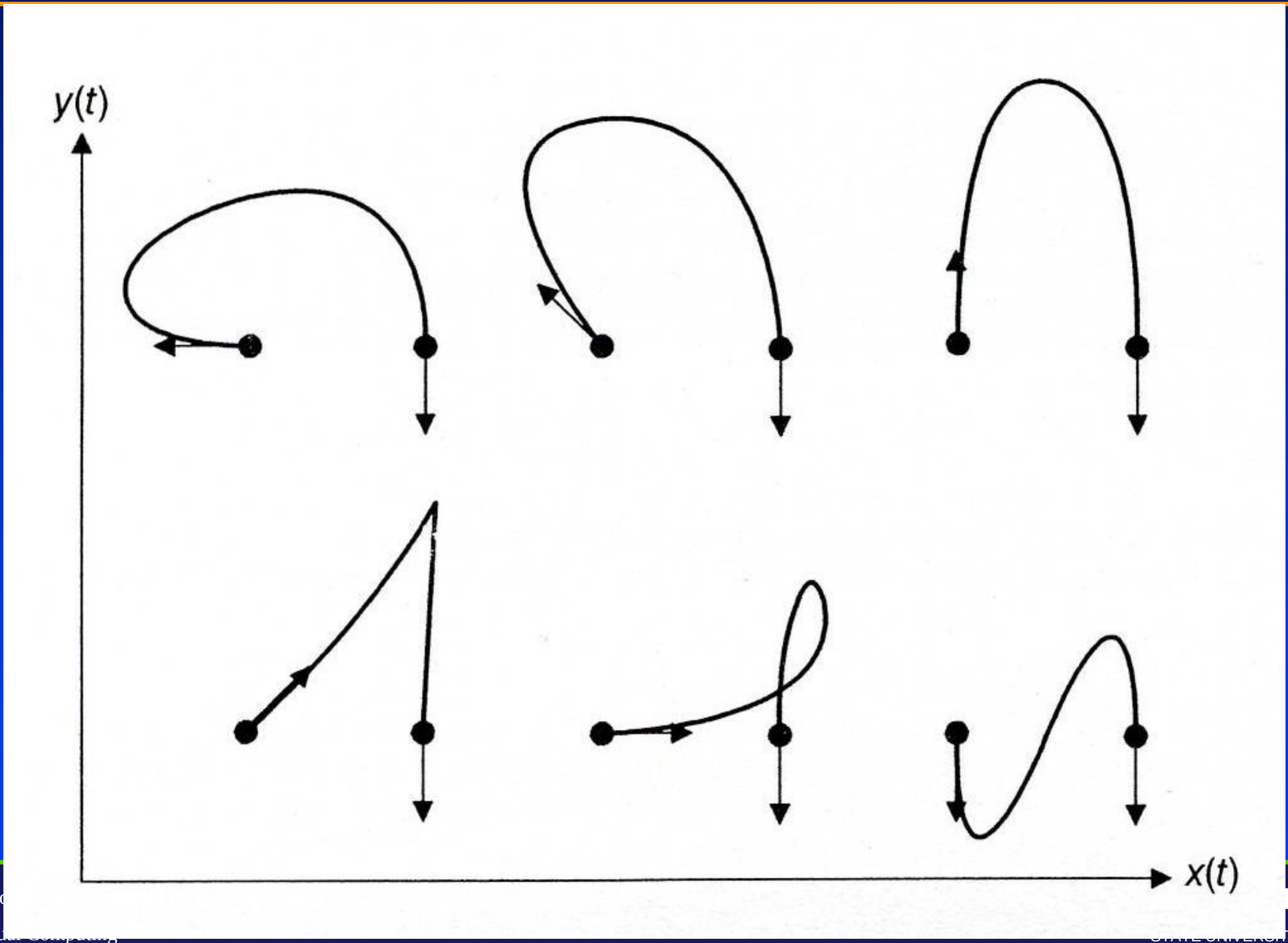
Cubic Hermite Splines



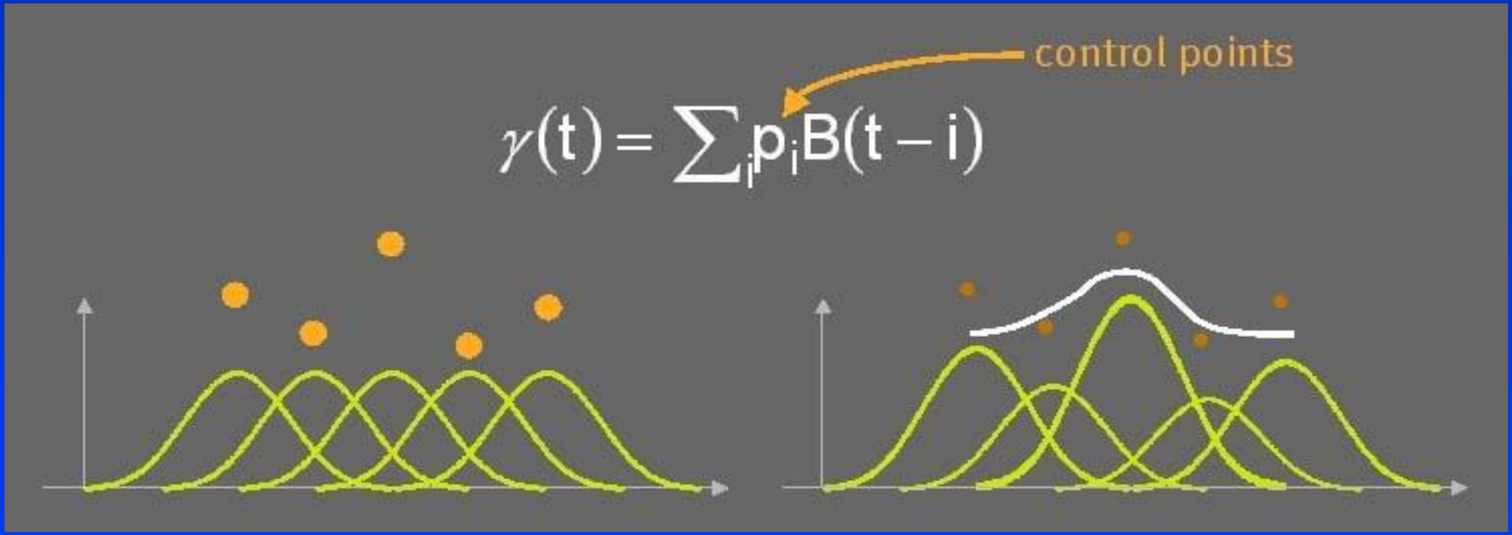
Varying the Magnitude of the Tangent Vector



Varying the Direction of the Tangent Vector



Piecewise Polynomial Blending

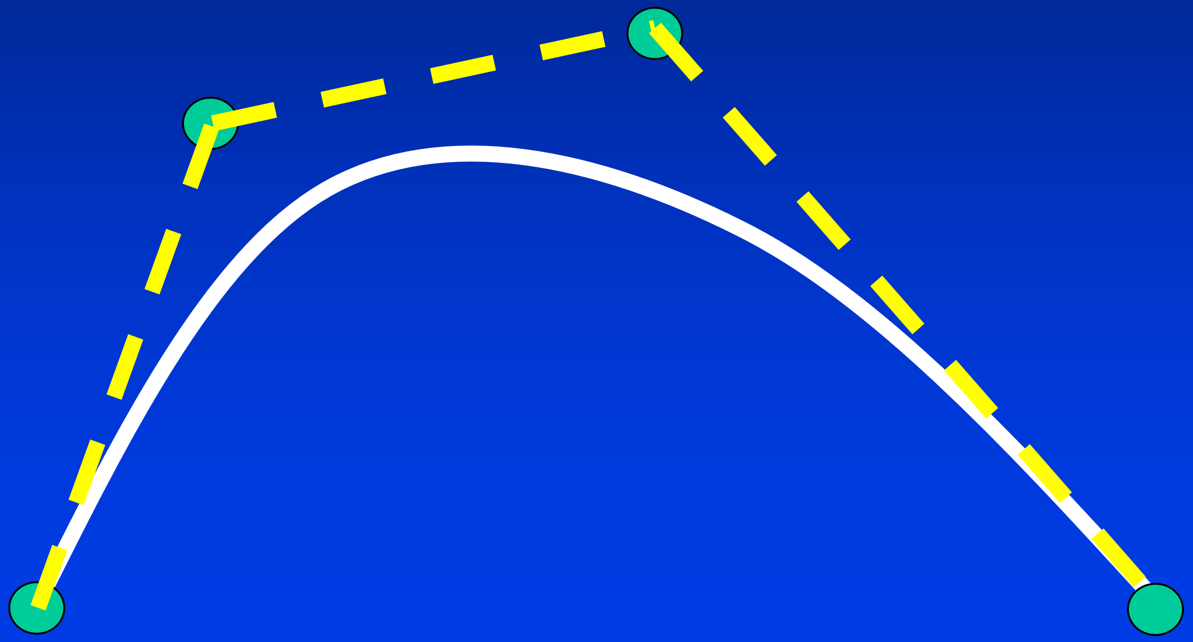


Why Cubic Polynomials

- Lowest degree for specifying curve in space
- Lowest degree for specifying points to interpolate and tangents to interpolate
- Commonly used in computer graphics
- Lower degree has too little flexibility
- Higher degree is unnecessarily complex, exhibit undesired wiggles

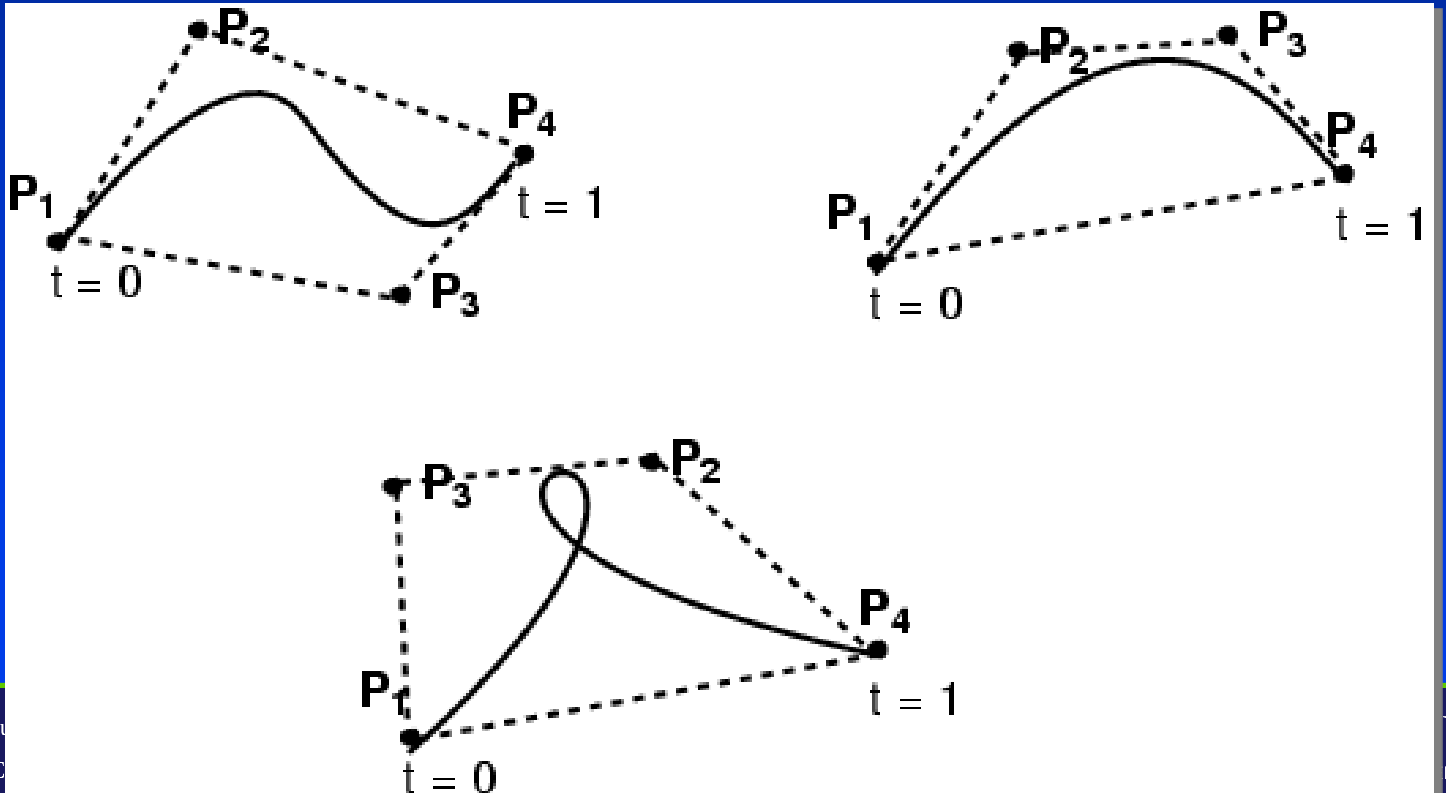
Cubic Bezier Curves

- Four control points to Bezier curve
- Curve geometry



Cubic Bézier Curve

- 4 control points
- Curve passes through the first & last control points
- Curve is tangent at $\mathbf{P_0}$ to $(\mathbf{P_0}-\mathbf{P_1})$ and at $\mathbf{P_4}$ to $(\mathbf{P_4}-\mathbf{P_3})$



Curve Mathematics (Cubic)

- Bezier curve

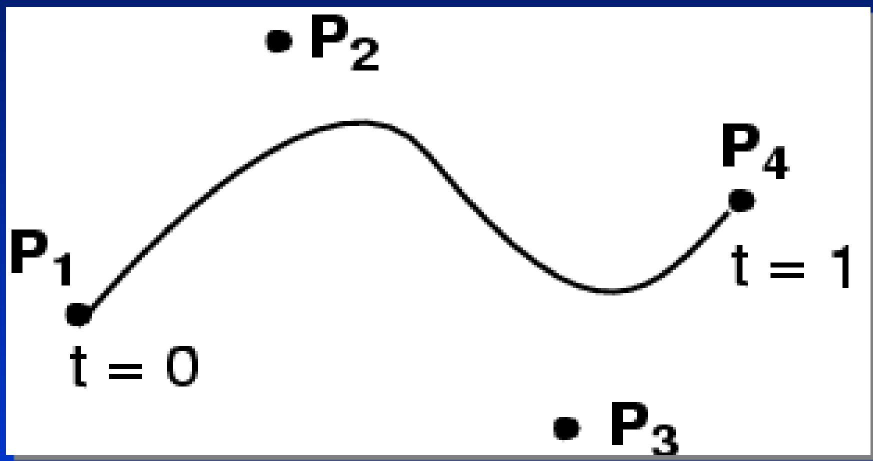
$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$$

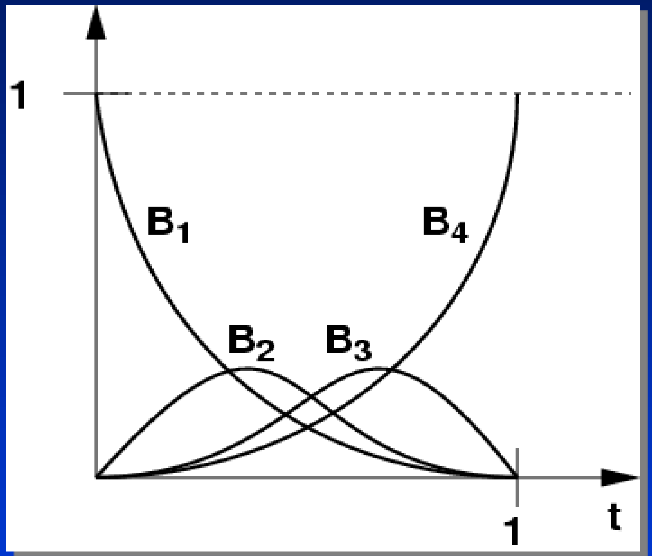
- Control points and basis functions

$$B_0^3(u) = (1-u)^3$$
$$B_1^3(u) = 3u(1-u)^2$$
$$B_2^3(u) = 3u^2(1-u)$$
$$B_3^3(u) = u^3$$

- Image and properties of basis functions

Cubic Bézier Basis Functions

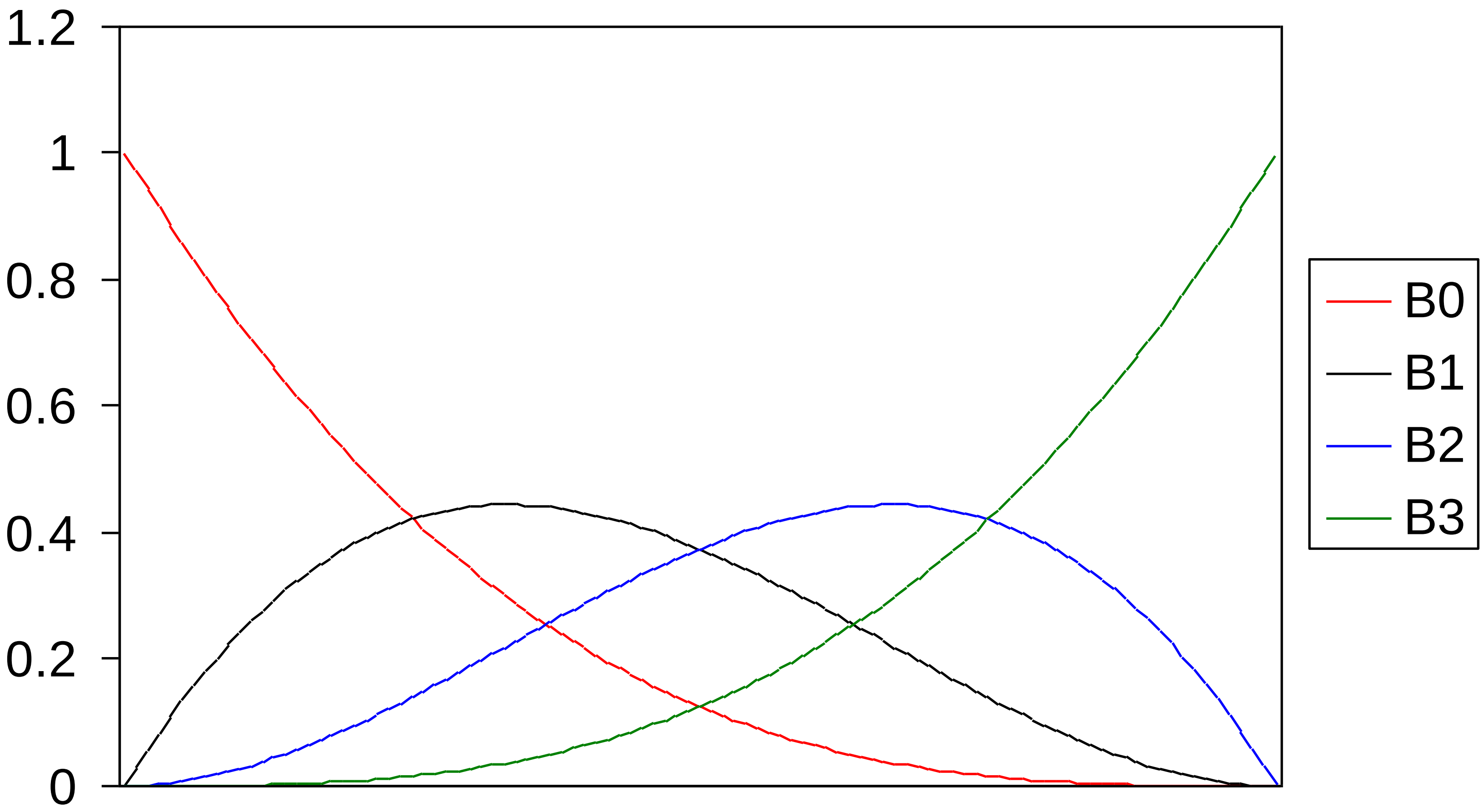




$$B_1(t) = (1-t)^3;\ B_2(t) = 3t(1-t)^2;\ B_3(t) = 3t^2(1-t);\ B_4(t) = t^3$$

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

The Bernstein Polynomials (n=3)



Recursive Evaluation

- Recursive linear interpolation

$$(1-u) \quad (u)$$

$$\mathbf{p}_0^0 \quad \mathbf{p}_1^0 \quad \mathbf{p}_2^0 \quad \mathbf{p}_3^0$$

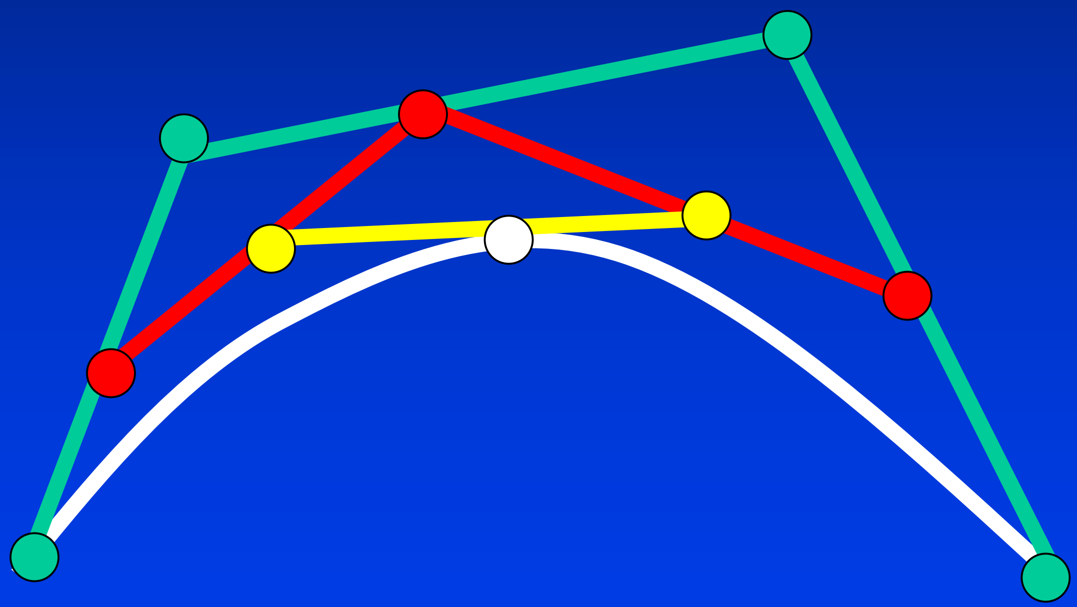
$$\mathbf{p}_0^1 \quad \mathbf{p}_1^1 \quad \mathbf{p}_2^1$$

$$\mathbf{p}_0^2 \quad \mathbf{p}_1^2$$

$$\mathbf{p}_0^3 = \mathbf{c}(u)$$

Recursive Subdivision Algorithm

- de Casteljau's algorithm for constructing Bézier curves



Basic Properties (Cubic)

- The curve passes through the first and the last points (end-point interpolation)
- Linear combination of control points and basis functions
- Basis functions are all polynomials
- Basis functions sum to one (partition of unity)
- All is functions are non-negative
- Convex hull (both necessary and sufficient)
- Predictability

Bezier Curves (Degree n)

- Curve: $c(u) = \sum_{i=0}^{n} p_i B_i^n(u)$
- Control points p_i
- Basis functions $B_i^n(u)$ are bernstein polynomials of degree n:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

$$\binom{n}{i} = \frac{n!}{(n-i)!\,i!}$$

STONY BROOK
STATE UNIVERSITY OF NEW YORK

Recursive Computation: The De Casteljau Algorithm

$$B_i^n(u) = (1-u)B_i^{n-1}(u) + uB_{i-1}^{n-1}(u)$$

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

$$= \binom{n-1}{/\,i} u^i (1-u)^{n-i} + \binom{n-1}{i-1} u^i (1-u)^{n-i}$$

$$= (1-u)B_i^{n-1}(u) + uB_{i-1}^{n-1}(u)$$

Recursive Computation

$$\mathbf{p}_i^0 = \mathbf{p}_i, i = 0,1,2,\ldots n$$

$$\mathbf{p}_i^j = (1-u)\mathbf{p}_i^{j-1} + u\mathbf{p}_{i+1}^{j-1}$$

$$\mathbf{c}(u) = \mathbf{p}_0^n(u)$$

Properties

- End point interpolation.
- Basis functions are non-negative.
- The summation of basis functions are unity
  – Binomial Expansion Theorem:

$$1 = [u + (1-u)]^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by the control points.

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

Bezier Curve Rendering

- Use its control polygon to approximate the curve
- Recursive subdivision till the tolerance is satisfied
- Algorithm go here
  - If the current control polygon is flat (with tolerance), then output the line segments, else subdivide the curve at u=0.5
  - Compute control points for the left half and the right half, respectively
  - Recursively call the same procedure for the left one and the right one

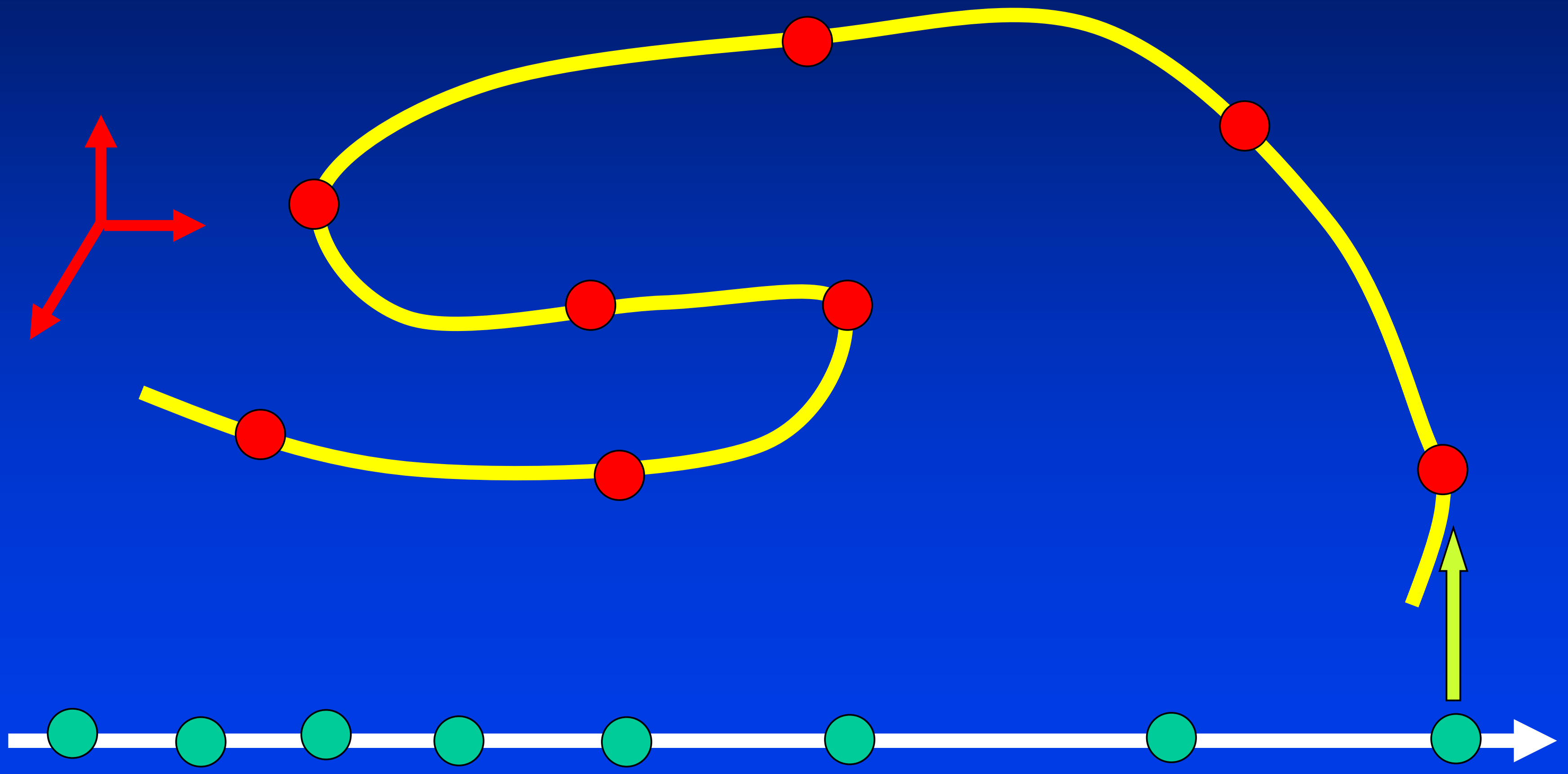
High-Degree polynomials

- More degrees of freedom
- Easy to compute
- Infinitely differentiable
- Drawbacks:
  – High-order
  – Global control
  – Expensive to compute, complex
  – undulation

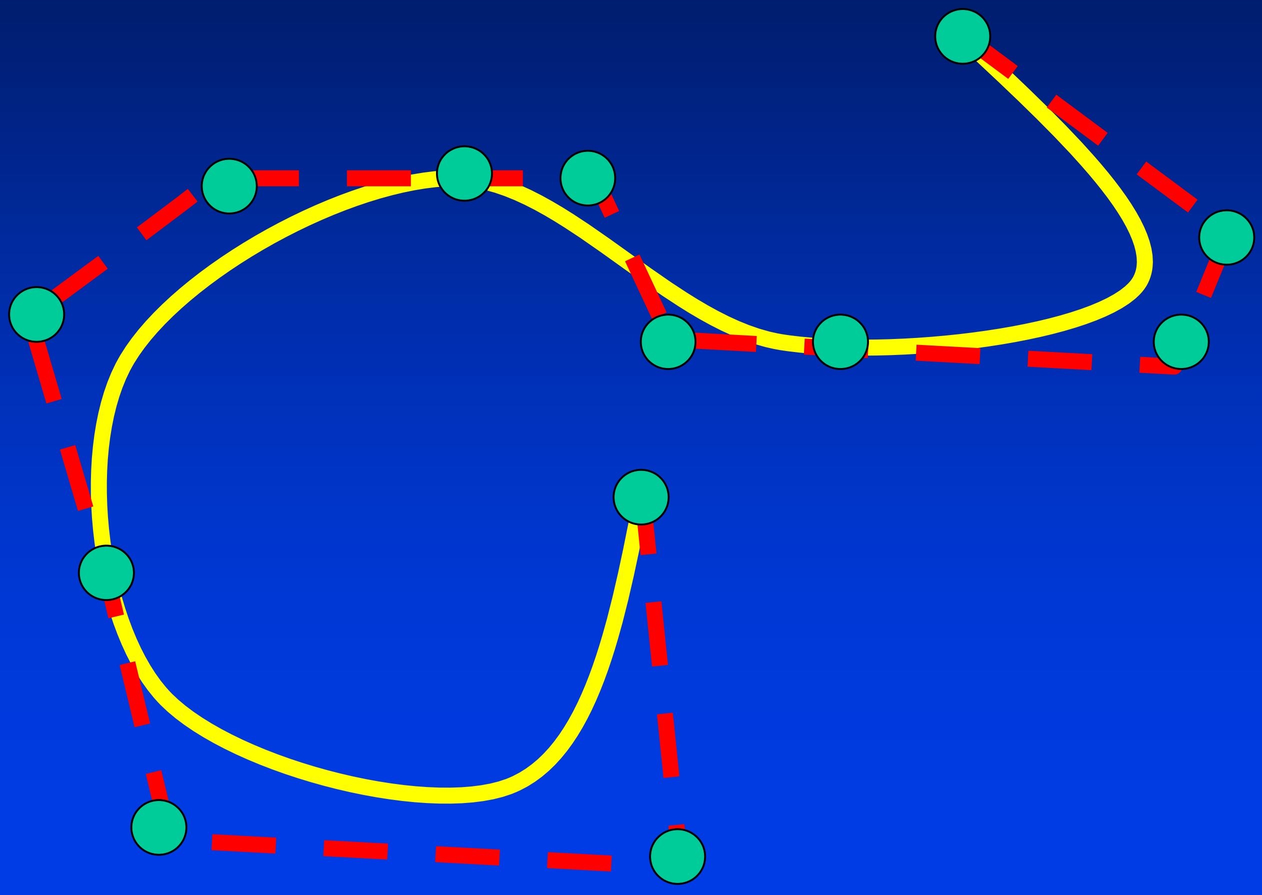
Piecewise Polynomials

- Piecewise --- different polynomials for different parts of the curve
- Advantages --- flexible, low-degree
- Disadvantages --- how to ensure smoothness at the joints (continuity)

Piecewise Curves



Piecewise Bezier Curves



Continuity

- One of the fundamental concepts
- Commonly used cases:

$$C^0, C^1, C^2$$

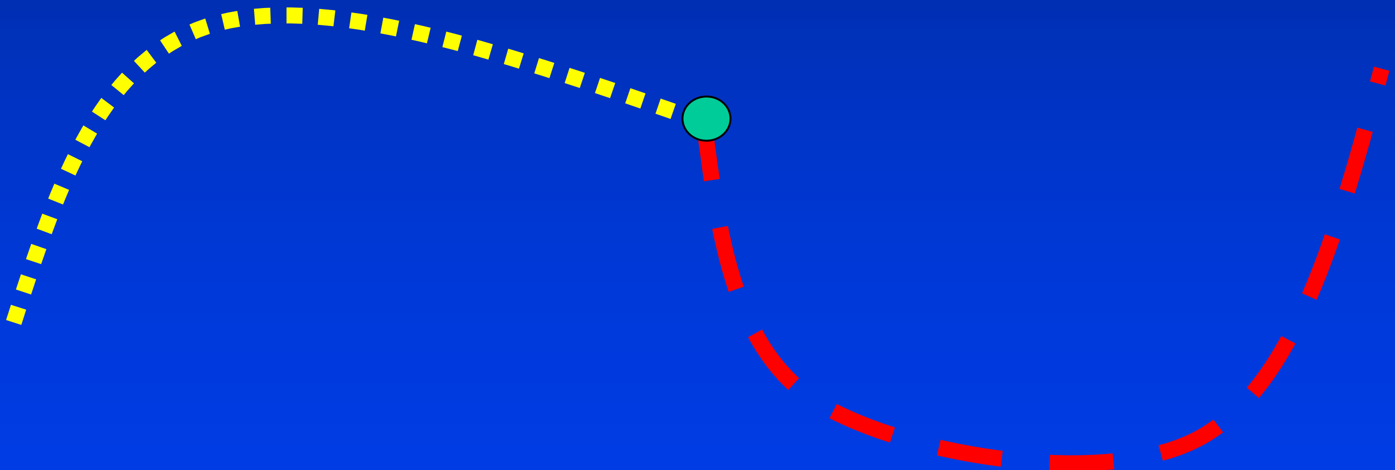
- Consider two curves: a(u) and b(u) (u is in [0,1])

Continuity

- Continuity between two parametric curves:
  - Geometric continuity
    - G^0: the two curves are connected
    - G^1: the two tangents have the same direction
  - Parametric continuity
    - C^0: the two curves are connected
    - C^1: the two tangents are equal

Positional Continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

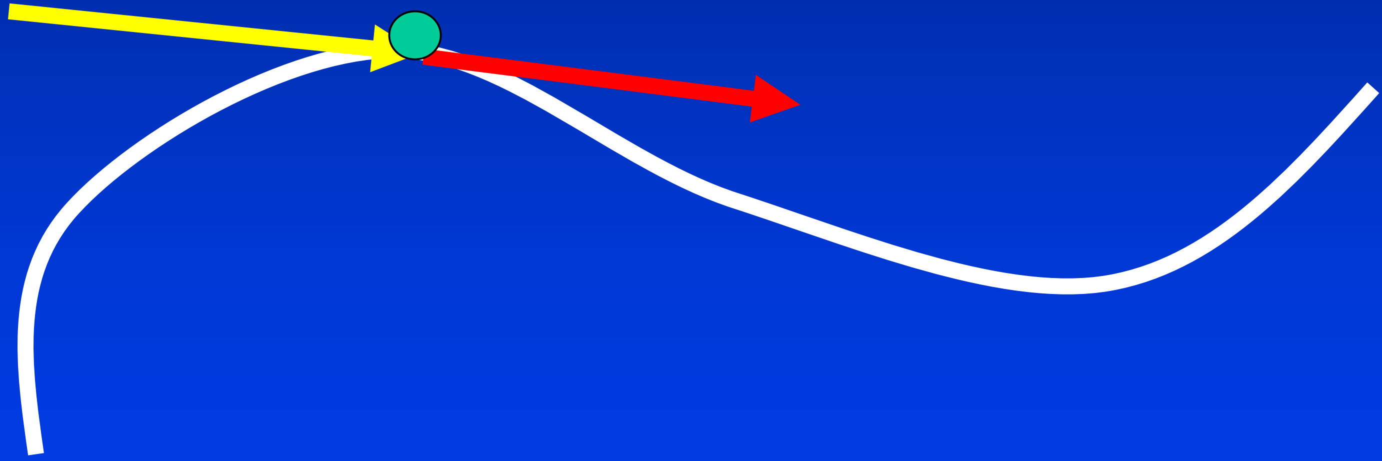


STONY BROOK
STATE UNIVERSITY OF NEW YORK

Derivative Continuity

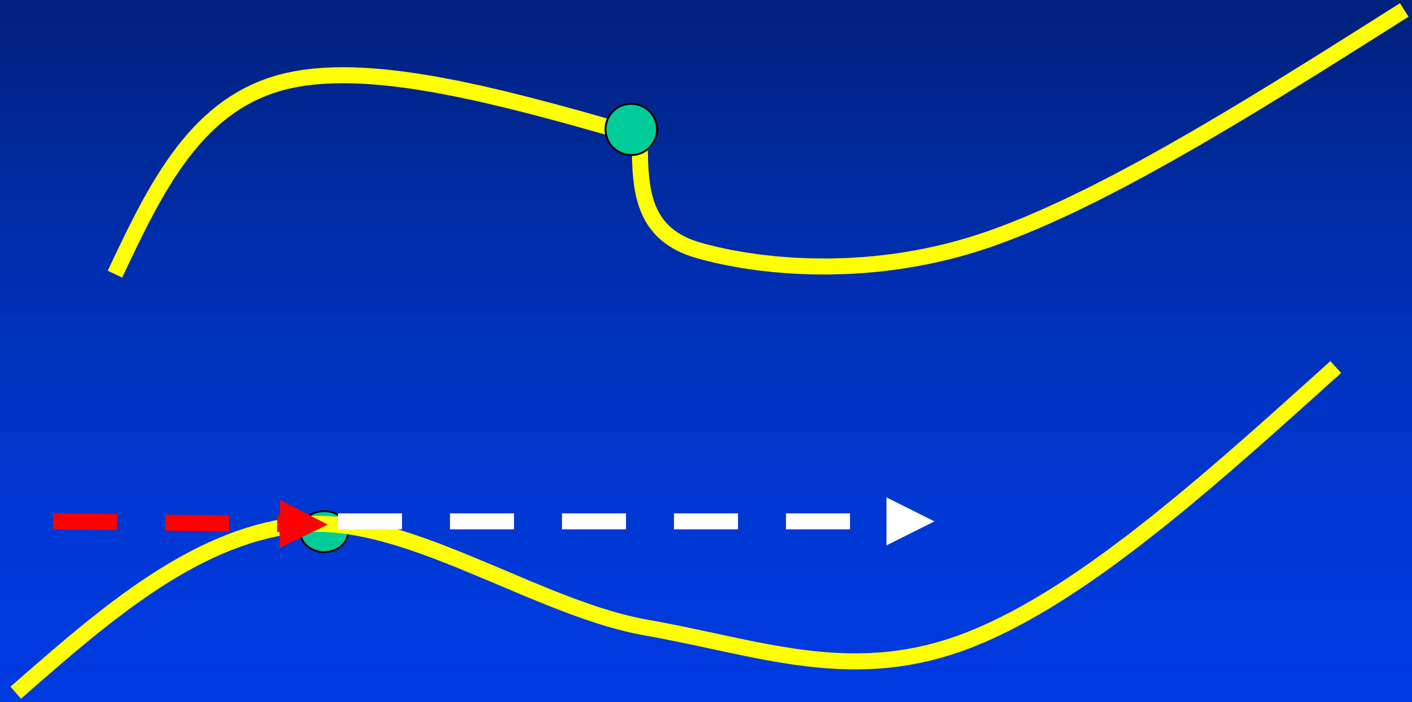
$$\mathbf{a}(1) = \mathbf{b}(0)$$

$$\mathbf{a}'(1) = \mathbf{b}'(0)$$



Geometric Continuity

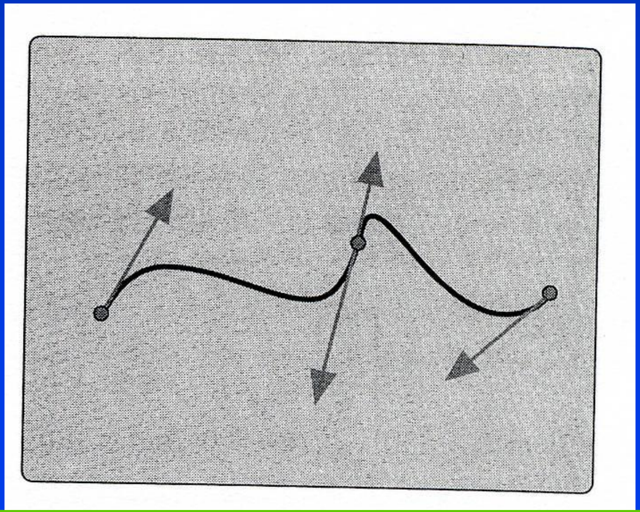
- G0 and G1

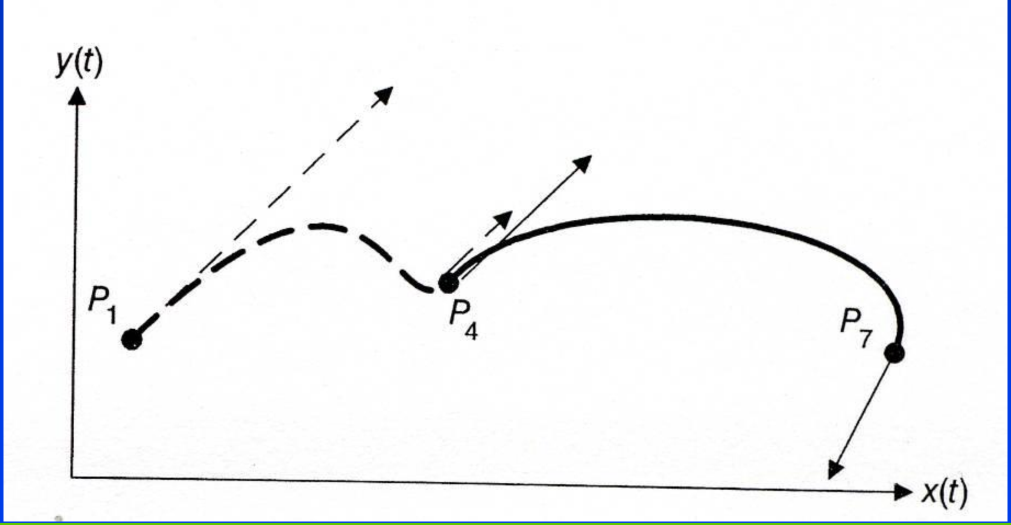


Obtaining Geometric Continuity G^1

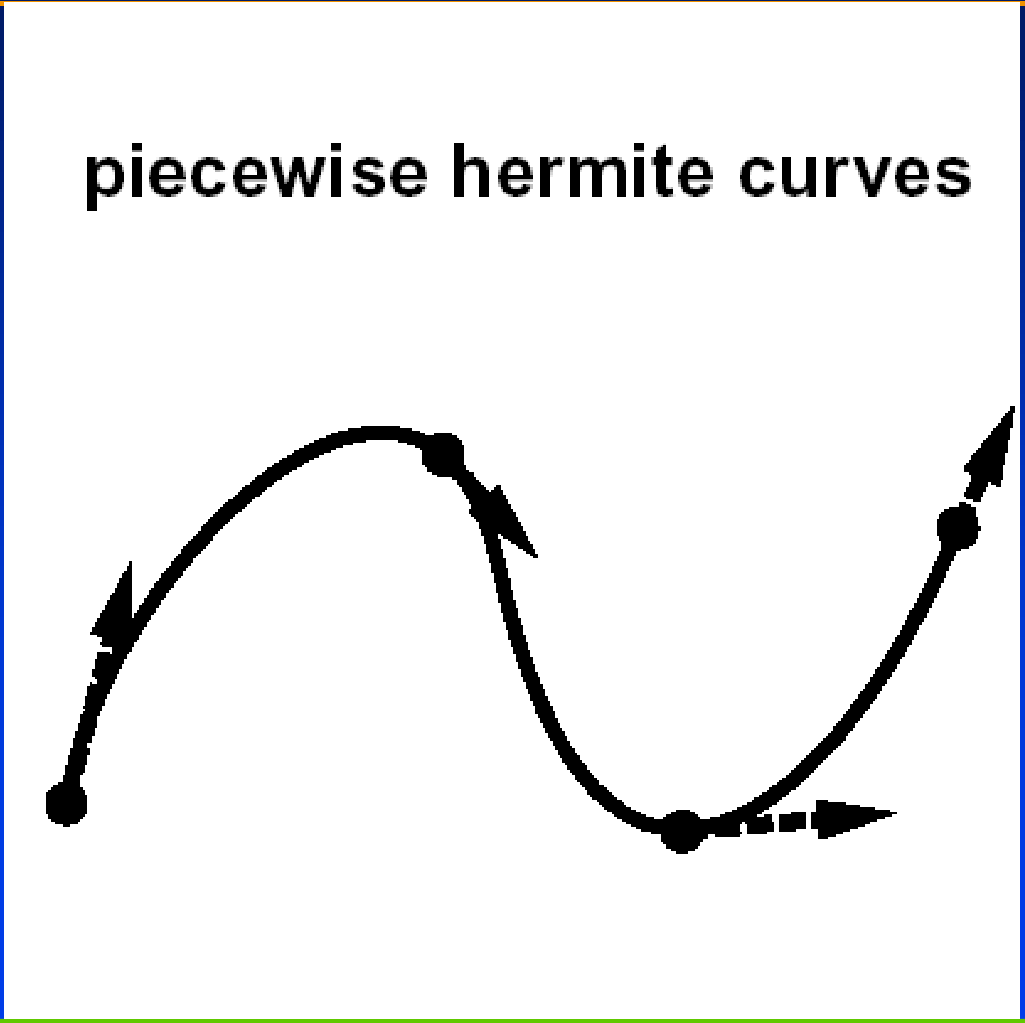
$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} \text{ and } \begin{bmatrix} P_4 \\ P_7 \\ kR_4 \\ R_7 \end{bmatrix}, \text{ with } k > 0.$$

for parametric continuity C^1, k = 1

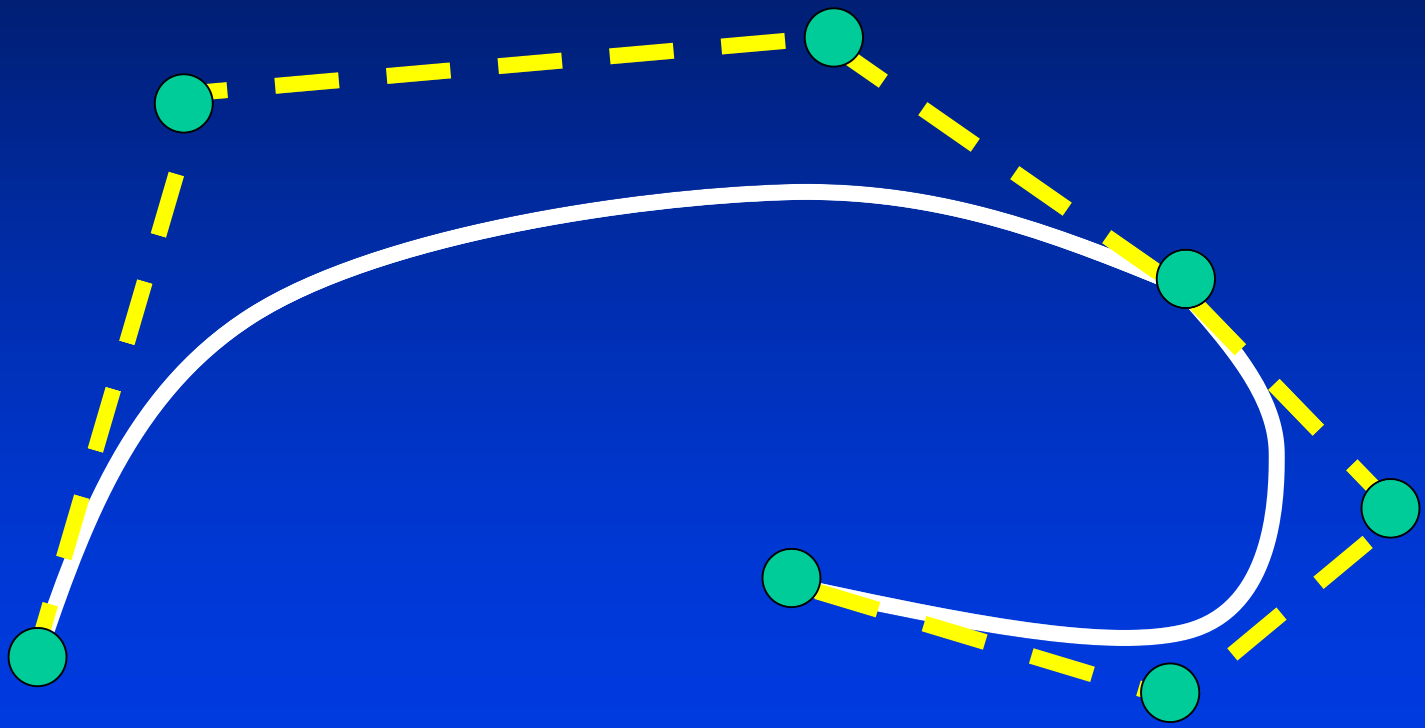




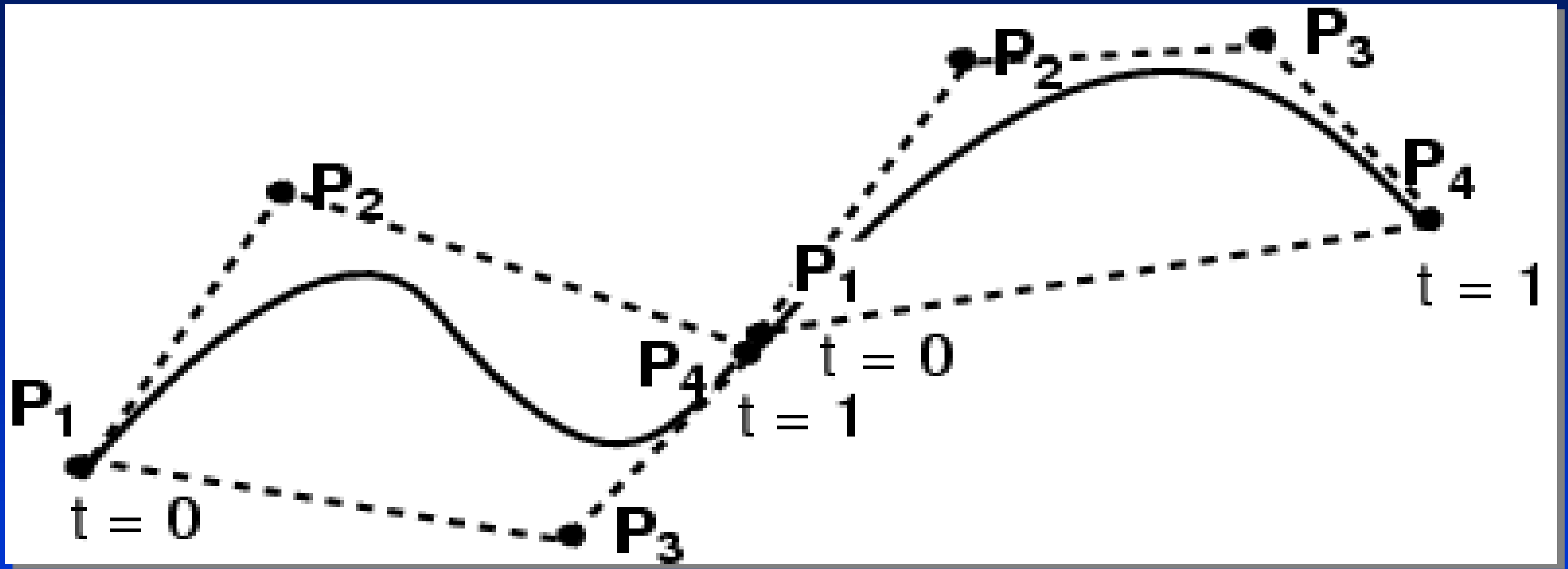
Piecewise Hermite Curves



Piecewise Bezier Curves



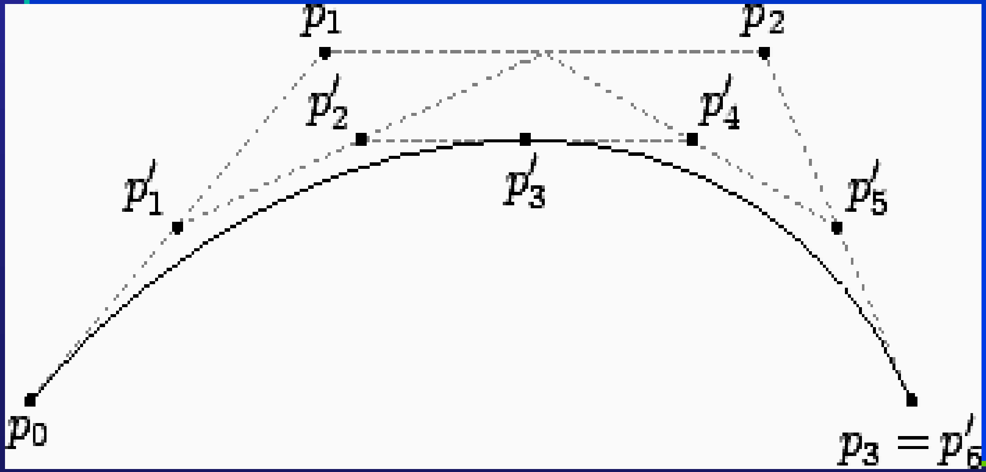
Connecting Cubic Bézier Curves



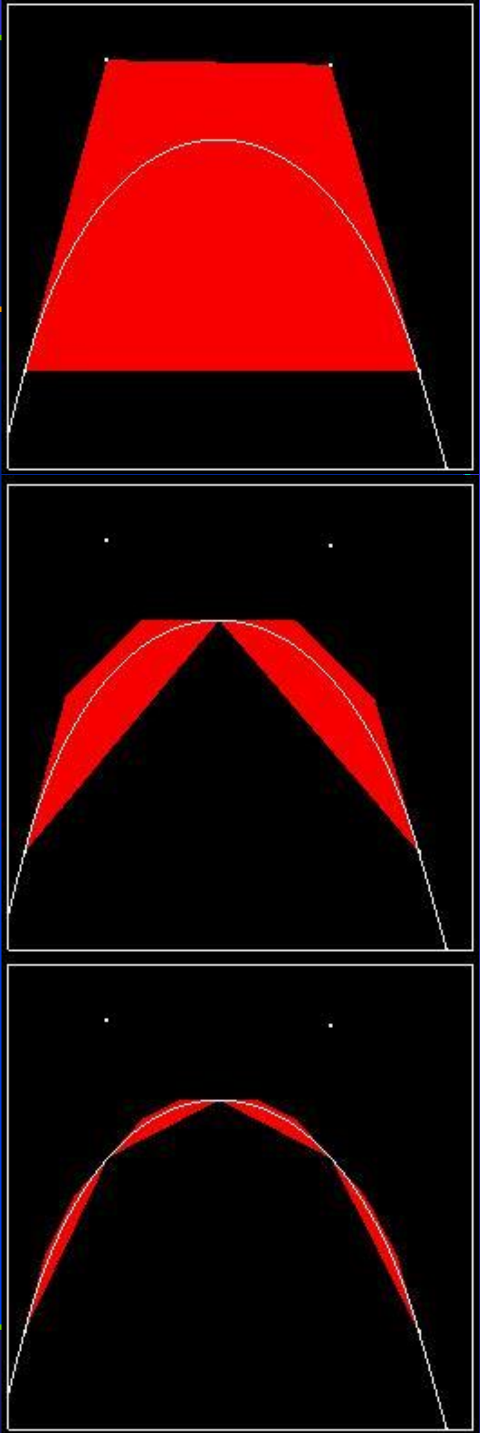
- How can we guarantee C0 continuity (no gaps between two curves)?
- How can we guarantee C1 continuity (tangent vectors match)?
- Asymmetric: Curve goes through some control points but misses others

Displaying Bezier Spline

- A Bezier curve with 4 control points:
  - $P_0 \quad P_1 \quad P_2 \quad P_3$
- Can be split into 2 new Bezier curves:
  - $P_0 \quad P'_1 \quad P'_2 \quad P'_3$
  - $P'_3 \quad P'_4 \quad P'_5 \quad P_3$



A Bézier curve is bounded by the convex hull of its control points.



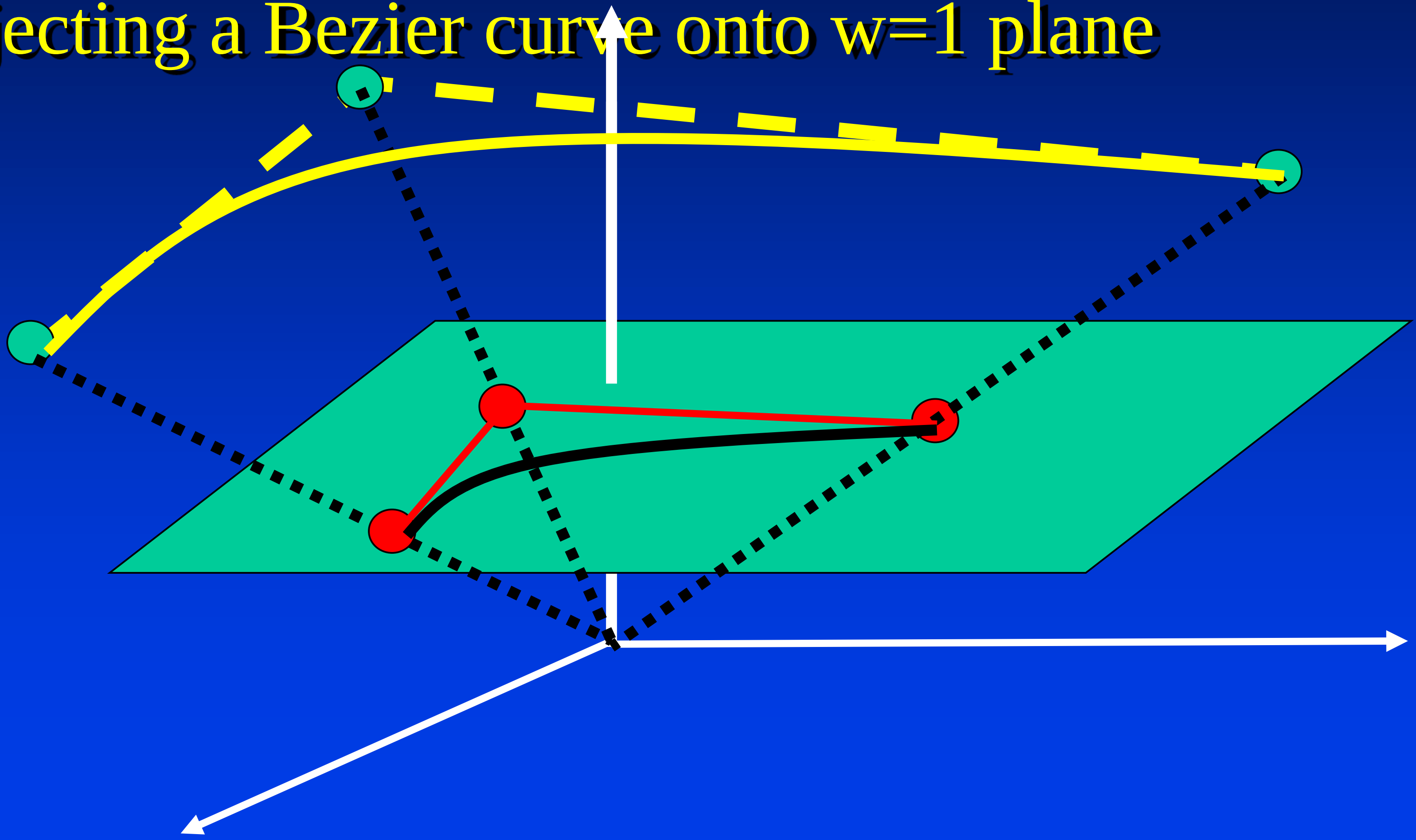
Department of Computer Science
Center for Visual Computing

Geometric NURBS

- Non-Uniform Rational B-Splines (NURBS)
- CAGD industry standard --- useful properties
- Degrees of freedom
  - Control points
  - Weights

Rational Bezier Curve

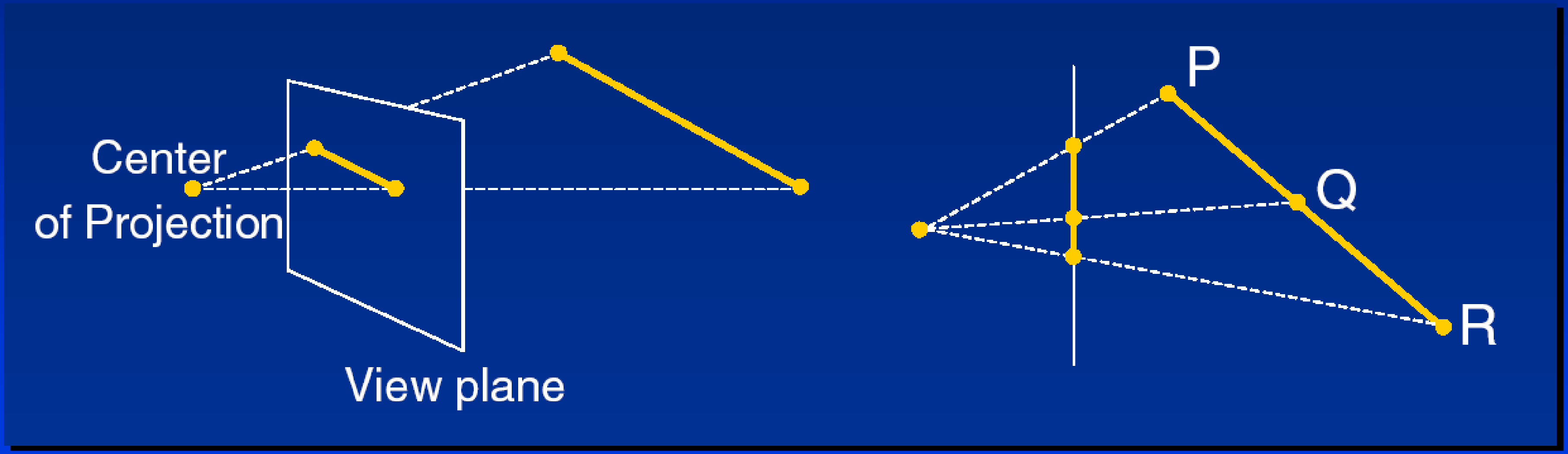
- Projecting a Bezier curve onto w=1 plane



Revisit Two Important Concepts

- Perspective projection
- Homogeneous coordinates

Perspective Projection



Consider Linear Case

$$\frac{\begin{bmatrix} x_0 w_0 \\ y_0 w_0 \end{bmatrix}(1-u) + \begin{bmatrix} x_1 w_1 \\ y_1 w_1 \end{bmatrix}(u)}{w_0(1-u) + w_1(u)}$$

or

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}(1-u) + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}(u)$$

From Bezier Spline to NURBS

- B-splines (Bezier Spline)

$$\mathbf{c}(u) = \sum_{i=0}^{n} \begin{bmatrix} \mathbf{p}_{i,x} \\ \mathbf{p}_{i,y} \\ \mathbf{p}_{i,z} \\ 1 \end{bmatrix} B_{i,k}(u)$$

- NURBS (curve)

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} \mathbf{p}_i w_i B_{i,k}(u)}{\sum_{i=0}^{n} w_i B_{i,k}(u)}$$

Two Examples

- B-splines (Bezier Spline)

$$\mathbf{c}(u) = \sum_{i=0}^{n} \begin{bmatrix} \mathbf{p}_{i,x} \\ \mathbf{p}_{i,y} \\ \mathbf{p}_{i,z} \\ 1 \end{bmatrix} B_{i,k}(u)$$

- NURBS (curve)

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} \mathbf{p}_i w_i B_{i,k}(u)}{\sum_{i=0}^{n} w_i B_{i,k}(u)}$$

$Linear:$

$(1-u)$

(u)

$Quadratic:$

$(1-u)^2$

$2(1-u)u$

$(u)^2$

Consider Quadratic Case

$$\frac{\begin{bmatrix} x_0 w_0 \\ y_0 w_0 \end{bmatrix}(1-u)^2 + \begin{bmatrix} x_1 w_1 \\ y_1 w_1 \end{bmatrix} 2(1-u)(u) + \begin{bmatrix} x_2 w_2 \\ y_2 w_2 \end{bmatrix}(u)^2}{w_0(1-u)^2 + w_1 2(1-u)(u) + w_2(u)^2}$$

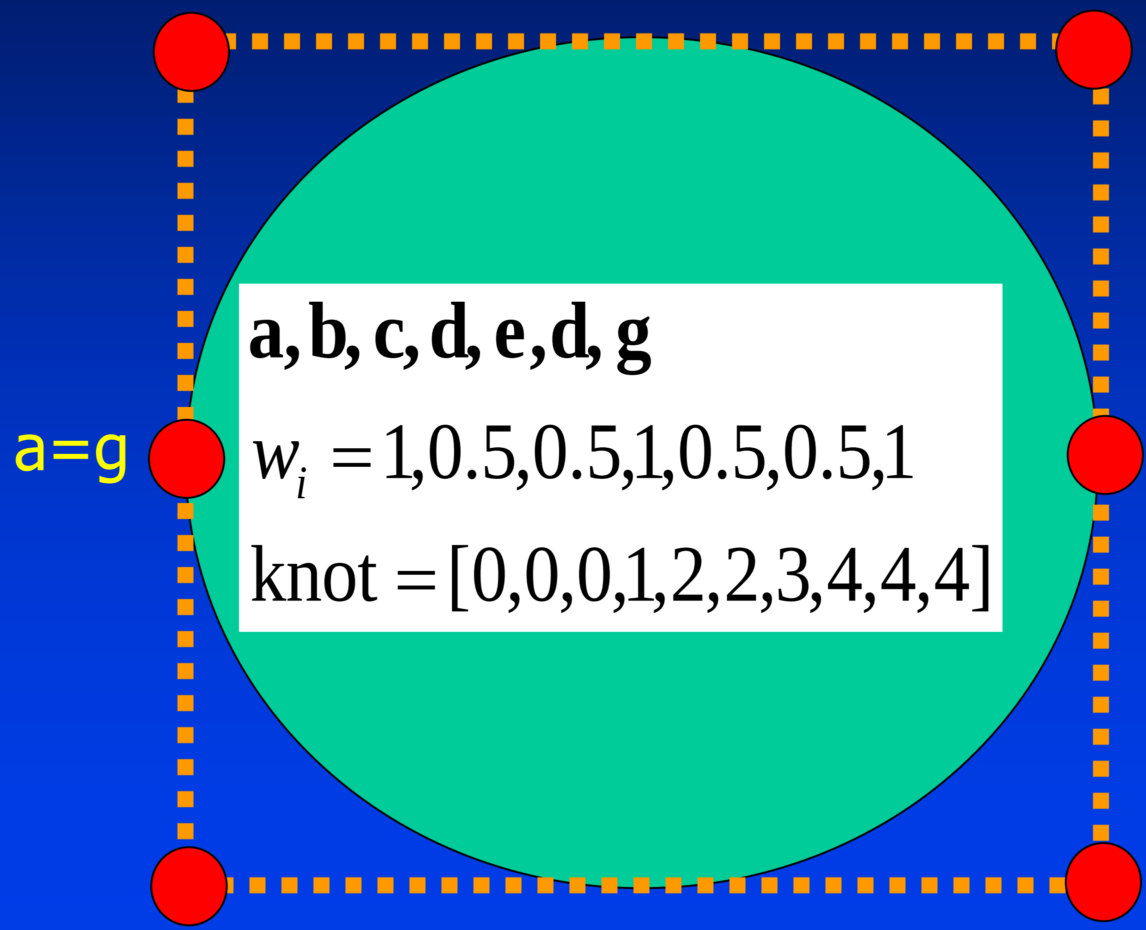
or

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}(1-u)^2 + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} 2(1-u)(u) + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}(u)^2$$

NURBS for Analytic Shapes

- Conic sections
- Natural quadrics
- Extruded surfaces
- Ruled surfaces
- Surfaces of revolution

NURBS Circle



$$\mathbf{a, b, c, d, e, d, g}$$

$$w_i = 1, 0.5, 0.5, 1, 0.5, 0.5, 1$$

$$\text{knot} = [0,0,0,1,2,2,3,4,4,4]$$

NURBS Curve

- Geometric components
  - Control points, parametric domain, weights, knots
- Homogeneous representation of B-splines
- Geometric meaning --- obtained from projection
- Properties of NURBS
  - Represent standard shapes, invariant under perspective projection, B-spline is a special case, weights as extra degrees of freedom, common analytic shapes such as circles, clear geometric meaning of weights