CSE528 Computer Graphics: 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

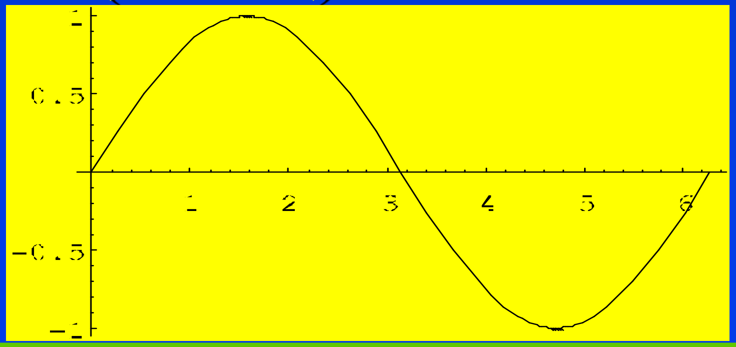
Parametric Curves

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

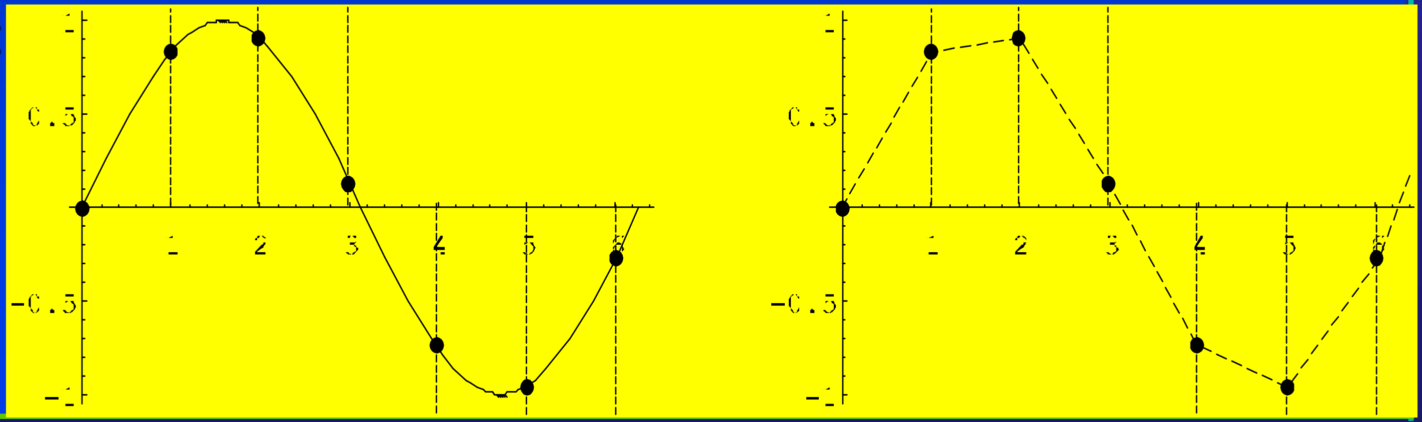
Explicit Representation

- 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:



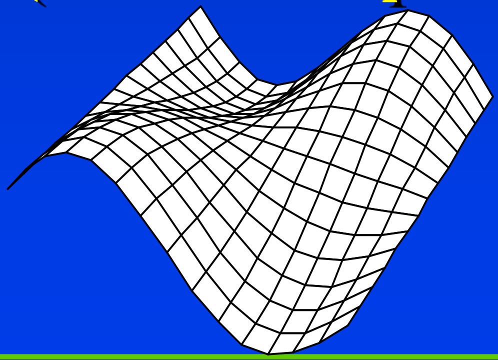
Modeling vs. Rendering

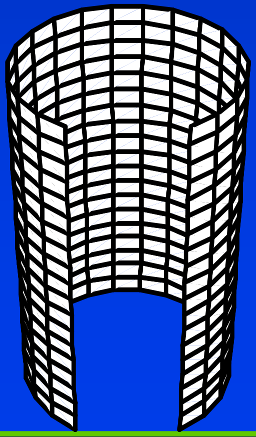
- Now we must determine how fine or coarse a representation we need to use in order to display this curve.
- We will sample the curve at regular intervals of θ along the length of the curve. In this example, the curve will be sampled at regular points a unit distance apart (i.e. at $\theta = 0, 1, 2 \ldots$).
- This yields the following sample points which we will join by straight lines which is the way the curve will be finally displayed on the raster:



Surfaces

- Note that the final representation is not very smooth. If the intervals are chosen carefully, however (for example, by relating the interval distance to the size of a pixel of the raster), then the curve representation will appear continuous and smooth.
- This technique may be extended to surfaces in the same manner (surfaces require 2 parameters):





Parametric Curves

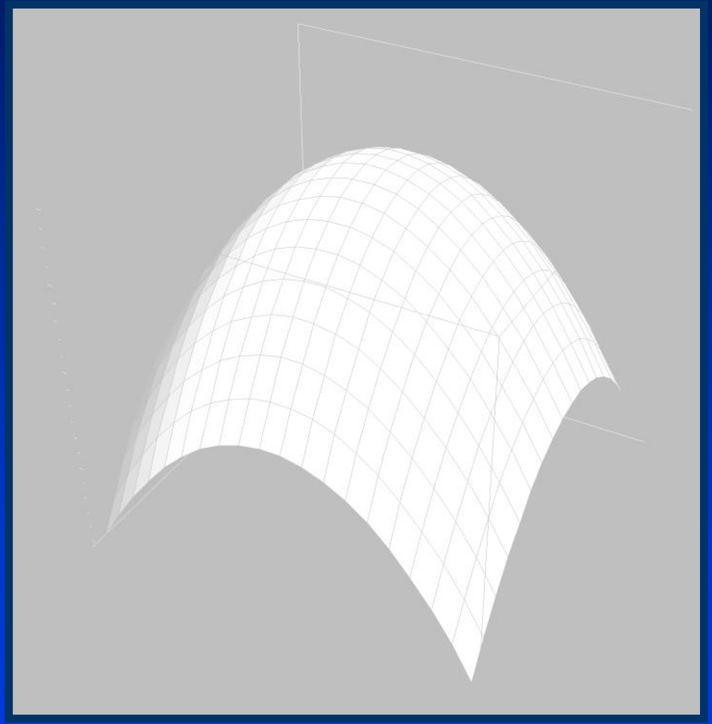
- Please remember to make comparisons between parametric representations and the following equations:
  - – Explicit representation:
    - $y = f(x)$
  - – Implicit representation:
    - $f(x,y) = 0$

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

Explicit Representation

- Explicit, non-parametric representation will naturally lead to the concept of parametric curves and surfaces
  - Bézier curves (de Casteljau ’59, P. Bézier ’62).
  - Spline curves/surfaces (de Boor ’72, Gordon *et al*. ’74, Böhm ’83).
  - Bernstein-Bézier solids (Lasser ’85), tensor product trivariate B-spline solid (Greissmair *et al*. ’89).



$$f(t,s) = (t,\ s,\ 1-(t^2+s^2))$$

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: $\mathbf{v} = \mathbf{v}_0(1-t) + \mathbf{v}_1(t)$
- Local coordinates: $\mathbf{v} \in [\mathbf{v}_0, \mathbf{v}_1], t \in [0,1]$
- Re-parameterization: $f(u), u = g(v), f(g(v)) = h(v)$
- Affine transformation:

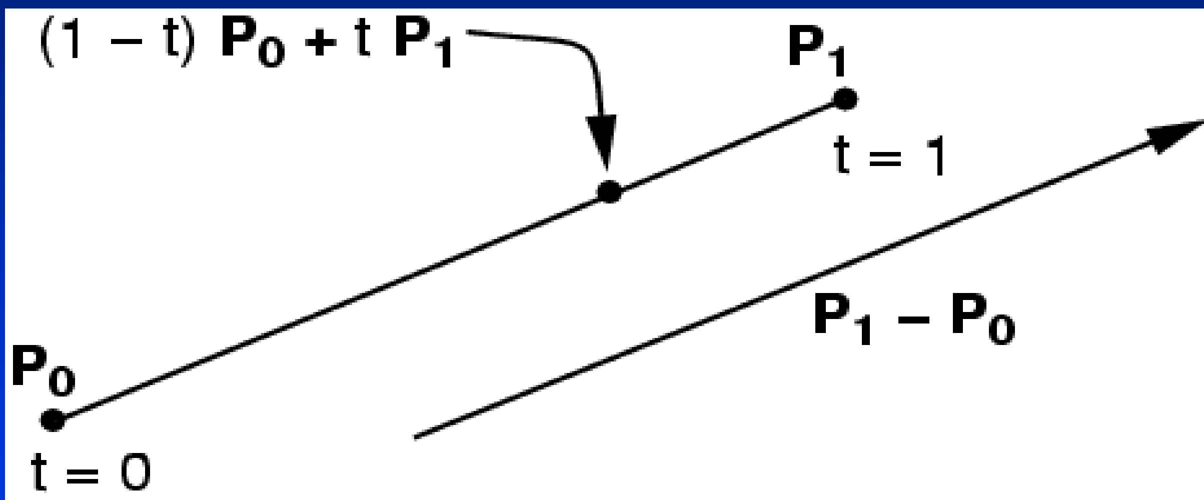
$$f(ax+by) = af(x) + bf(y)$$

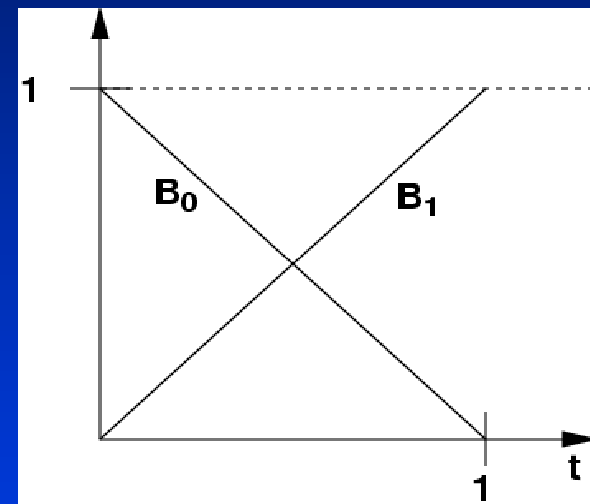
$$a+b=1$$

- Polynomials
- Continuity

Linear Interpolation

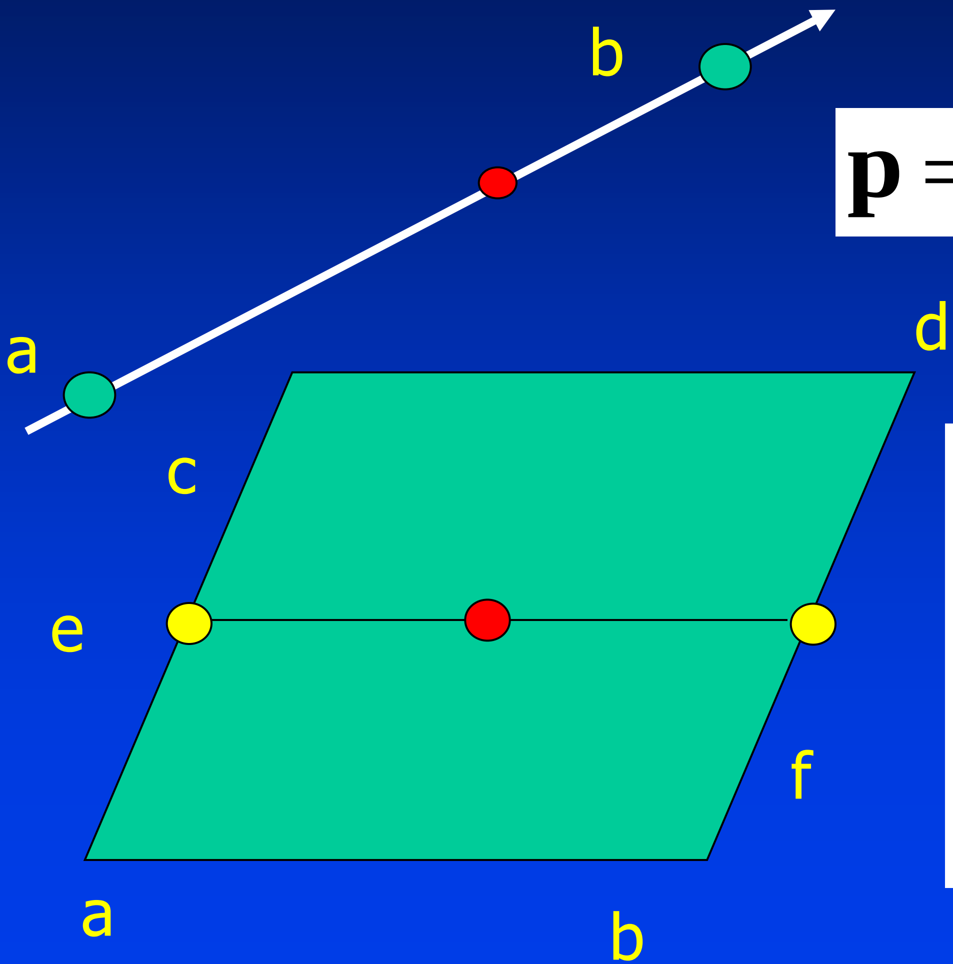
- Simplest "curve" between two points





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

Linear and Bilinear Interpolation



$$\mathbf{p} = (1-u)\mathbf{a} + u\mathbf{b}$$

$$\mathbf{e} = (1-u)\mathbf{a} + u\mathbf{c}$$

$$\mathbf{f} = (1-u)\mathbf{b} + u\mathbf{d}$$

$$\mathbf{p} = (1-v)\mathbf{e} + v\mathbf{f}$$

Fundamental Features

- Geometry
  - Position, direction, length, area, normal, tangent, etc.
- Interaction
  - Size, continuity, collision, intersection
- Topology
- Differential
  - Curvature, arc-length
- Physical
- Computer representation & data structure
- Others!

Mathematical Formulations

- Point:

$$\mathbf{p} = \begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix}$$

- Line:

$$\mathbf{l}(u) = [\mathbf{a} \quad \mathbf{a} \quad \mathbf{a}]^T u + [\mathbf{b} \quad \mathbf{b} \quad \mathbf{b}]^T$$

- Quadratic curve:

$$\mathbf{q}(u) = [\mathbf{a}_x \quad \mathbf{a}_y \quad \mathbf{a}_z]^T u^2 + [\mathbf{b}_x \quad \mathbf{b}_y \quad \mathbf{b}_z]^T u + [\mathbf{c}_x \quad \mathbf{c}_y \quad \mathbf{c}_z]^T$$

- Parametric domain and reparameterization:

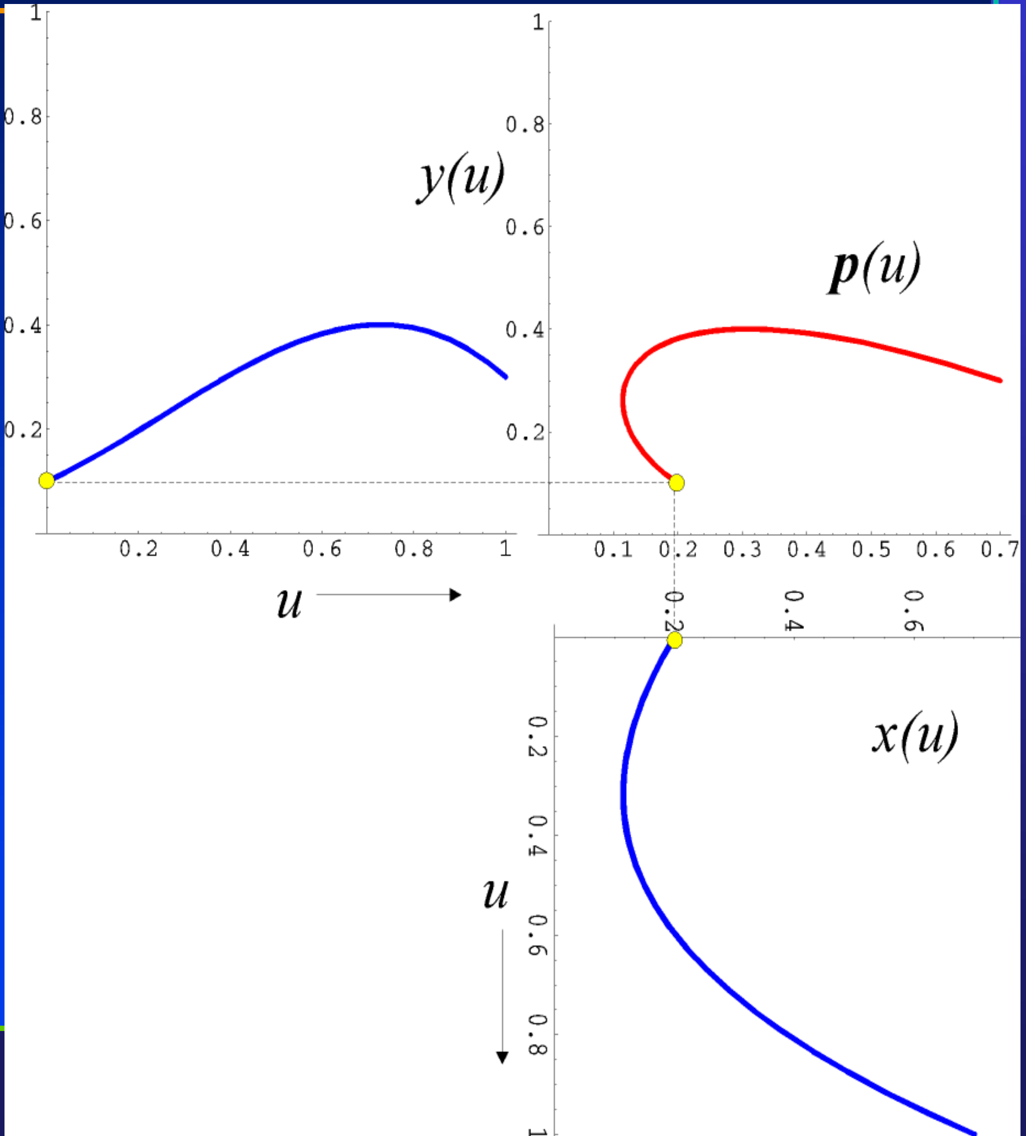
$$u \in [u_s, u_e]; v \in [0,1]; v = (u - u_s)/(u_e - u_s)$$

Parametric Cubic Curves

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

Parameterization: The Basic Concept

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



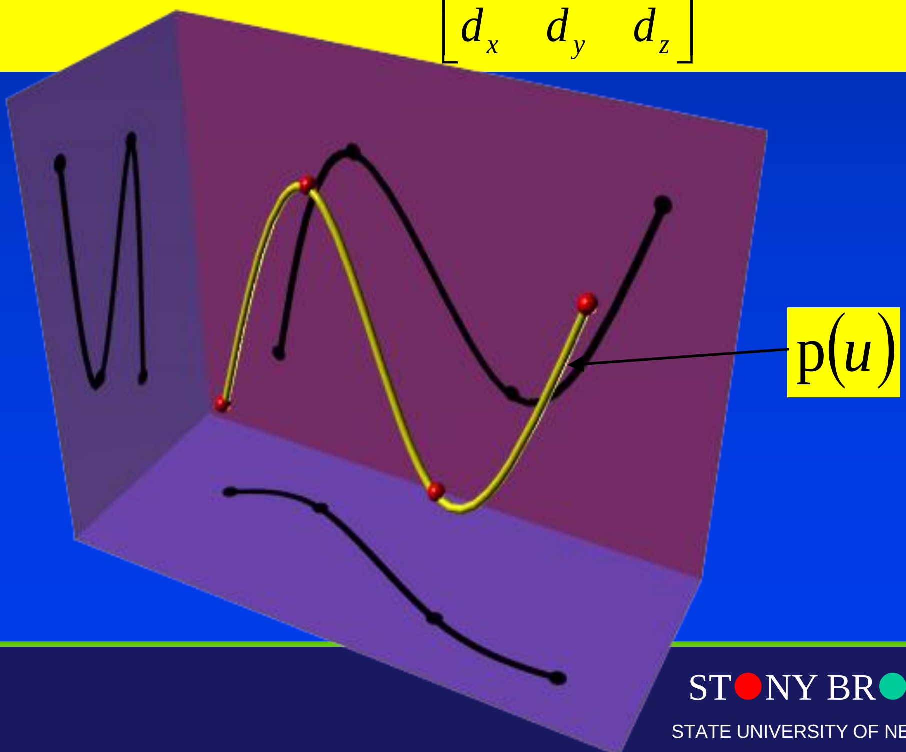
Splines

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

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

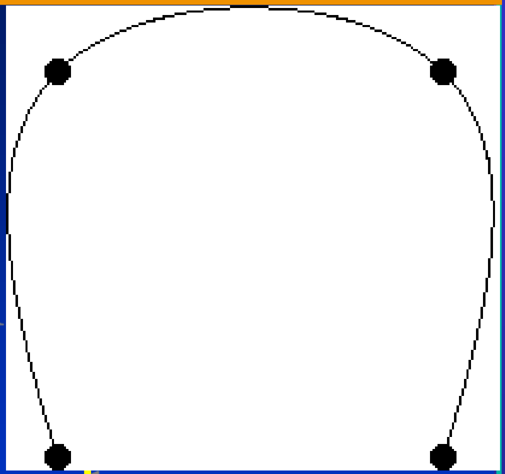
12 unknowns

$\therefore$ 4 3D points required



Defines the variation in x with distance u along the curve

Interpolation Curves



- Curve is constrained to pass through all control points
- Given points P_0, P_1, ... P_n, find lowest degree polynomial which passes through the points

$$x(t) = a_{n-1}t^{n-1} + ... + a_2t^2 + a_1t + a_0$$
$$y(t) = b_{n-1}t^{n-1} + ... + b_2t^2 + b_1t + b_0$$

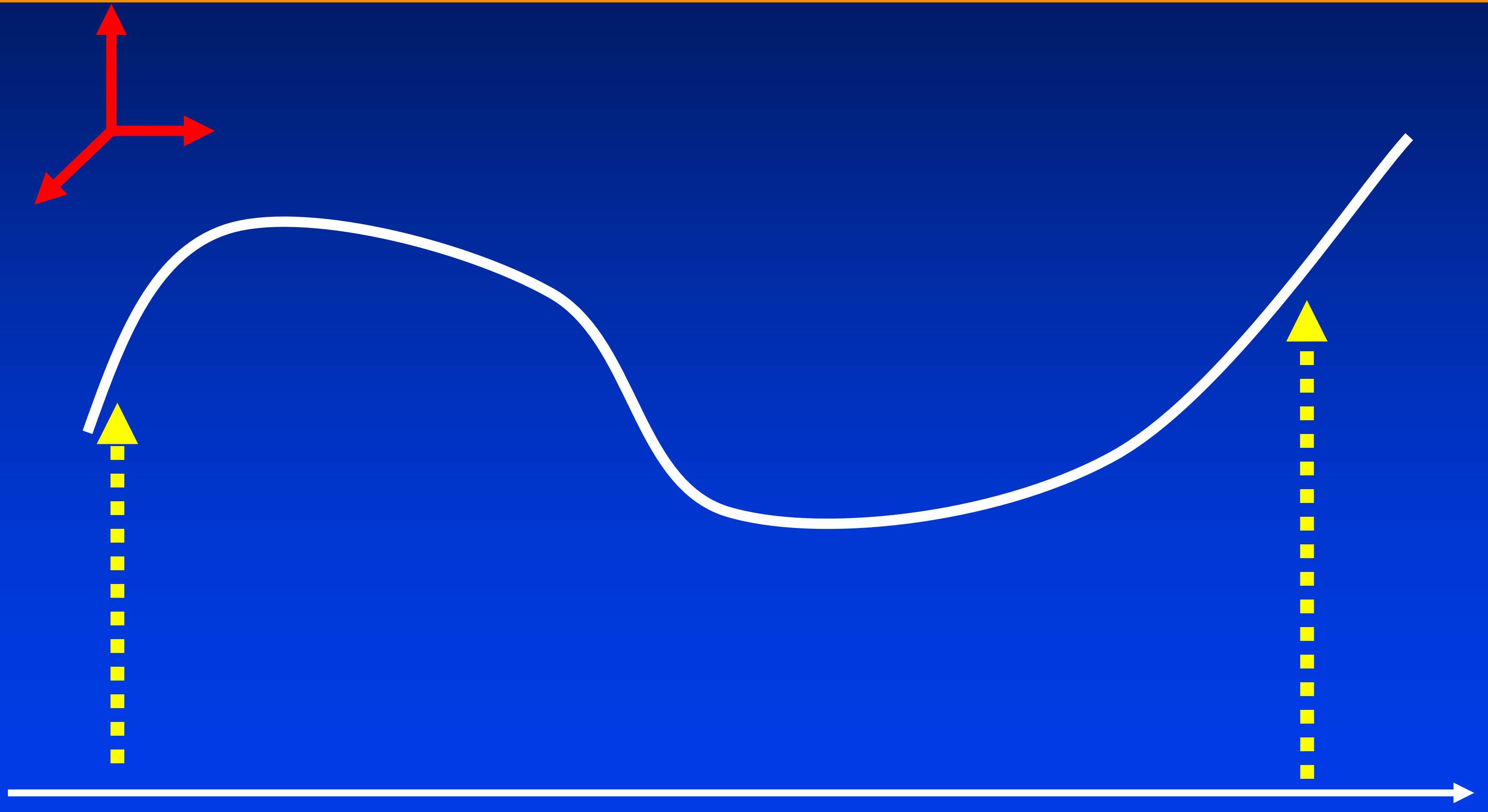
Parametric Polynomials

- High-order polynomials

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

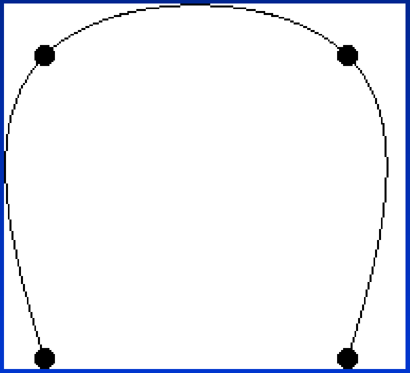
- No intuitive insight for the curved shape
- Difficult for piecewise smooth curves

Parametric Polynomials

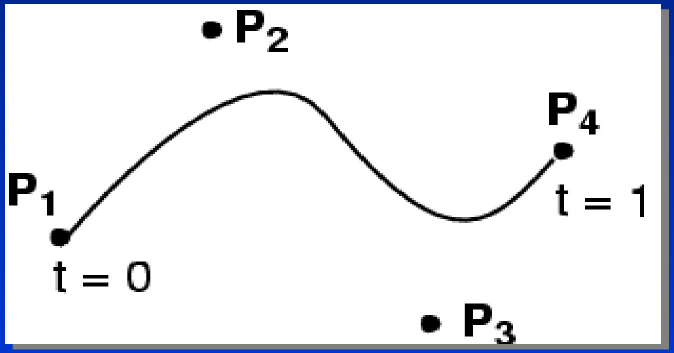


Definition: What's a Spline?

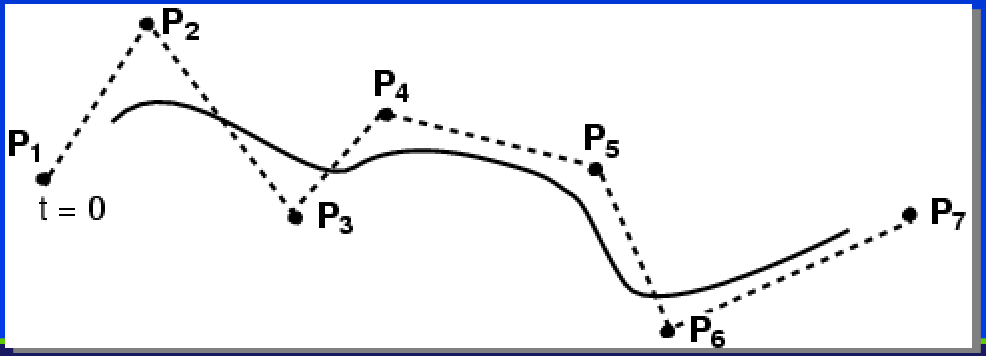
- Smooth curve defined by some control points
- Moving the control points changes the curve



Interpolation

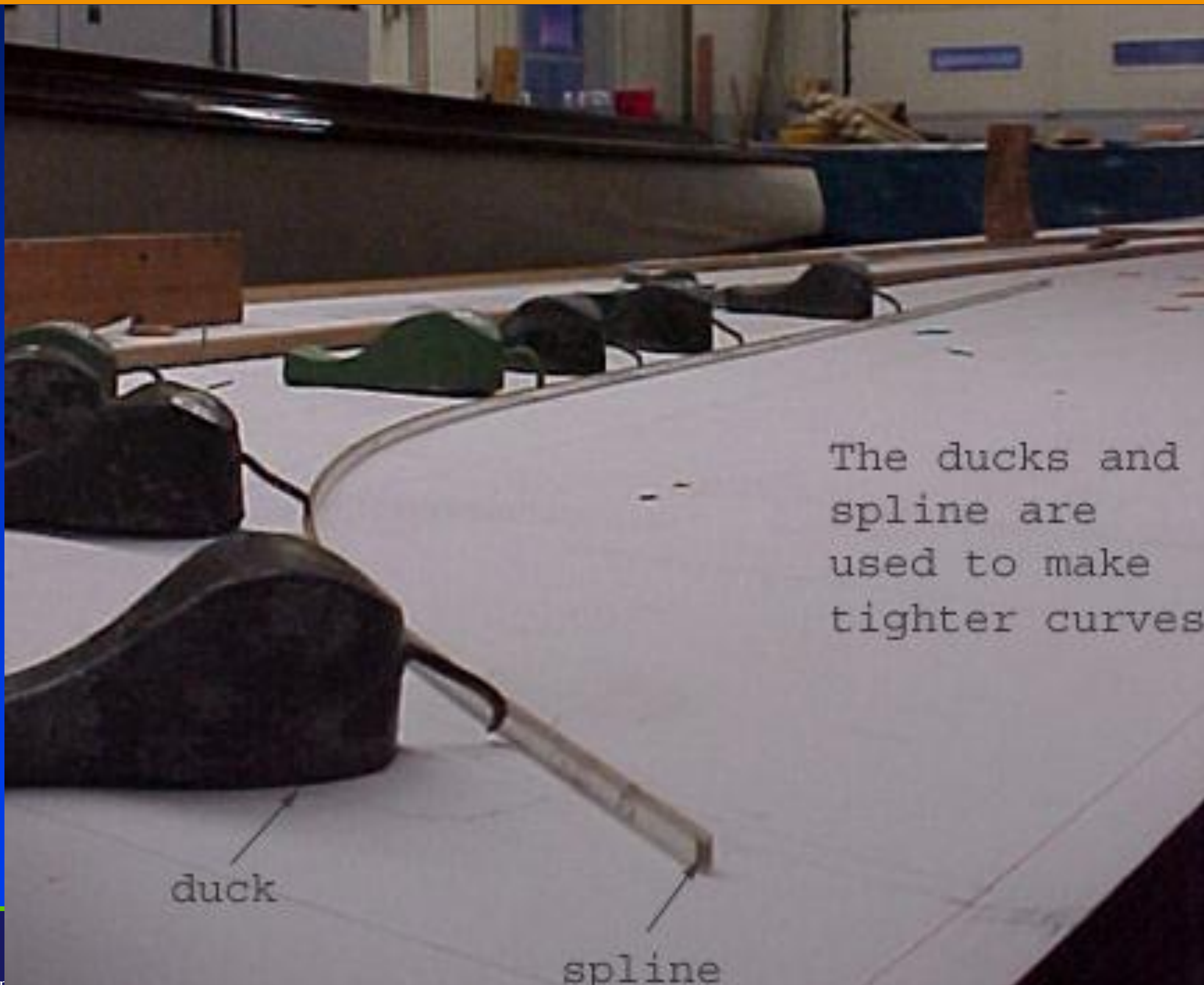


Bézier (approximation)

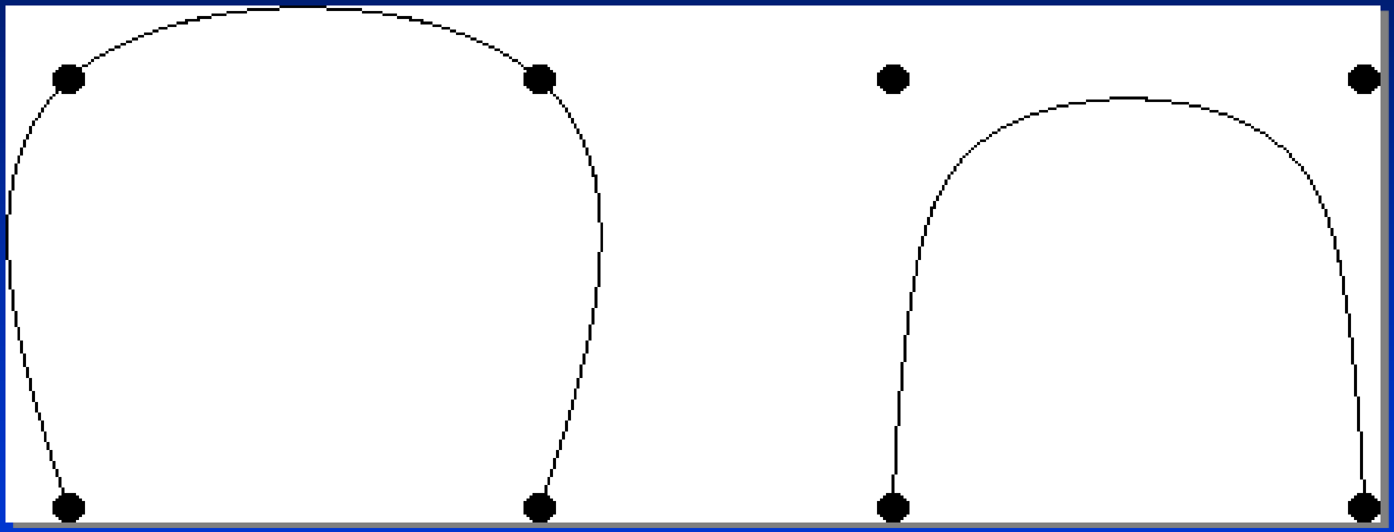


BSpline (approximation)

Interpolation Curves / Splines (Prior to the Digital Representation)



Interpolation vs. Approximation Curves



Interpolation

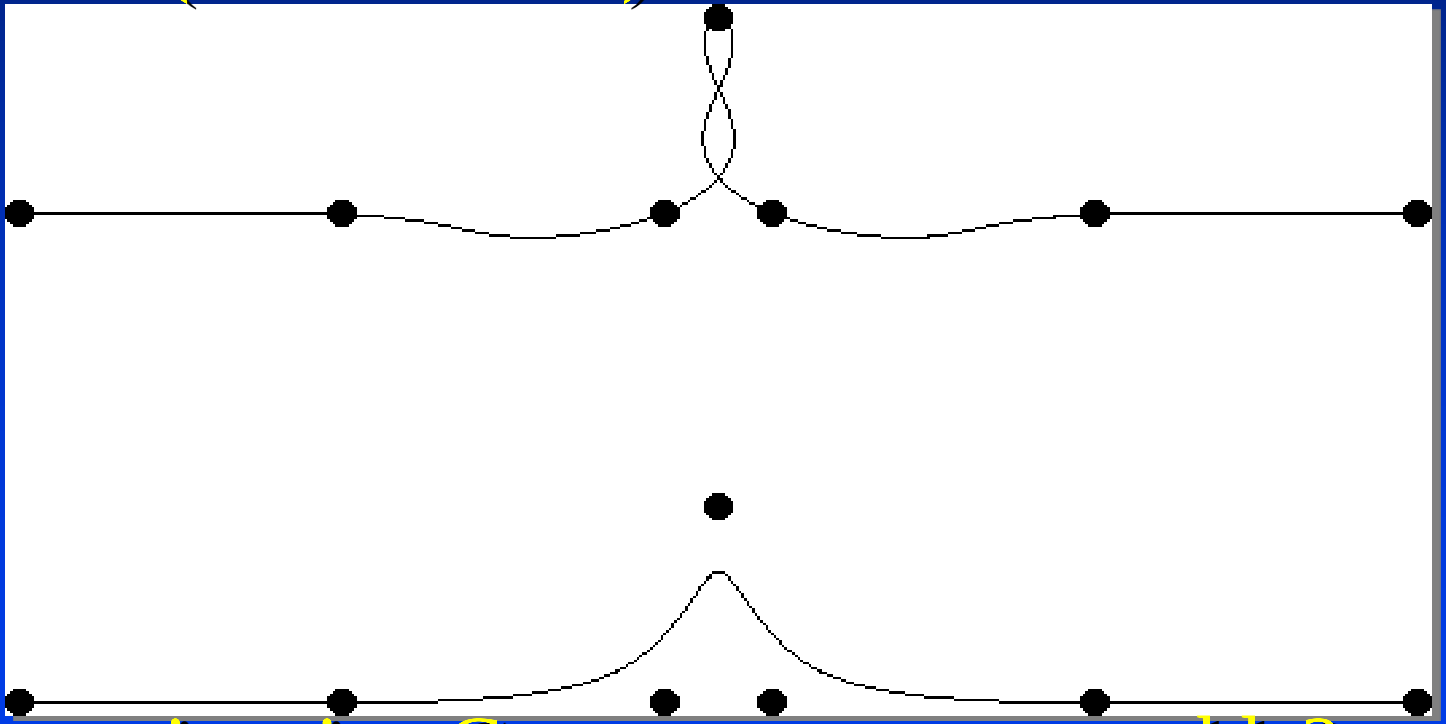
curve must pass
through control points

Approximation

curve is influenced
by control points

Interpolation vs. Approximation Curves

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



- Approximation Curve – more reasonable?

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?

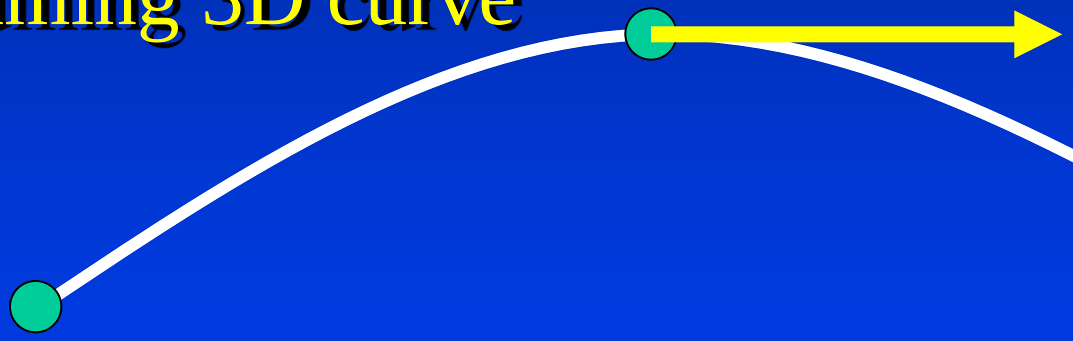
- Specify a set of points for interpolation and/or approximation with fixed or unfixed parameterization

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

- 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

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

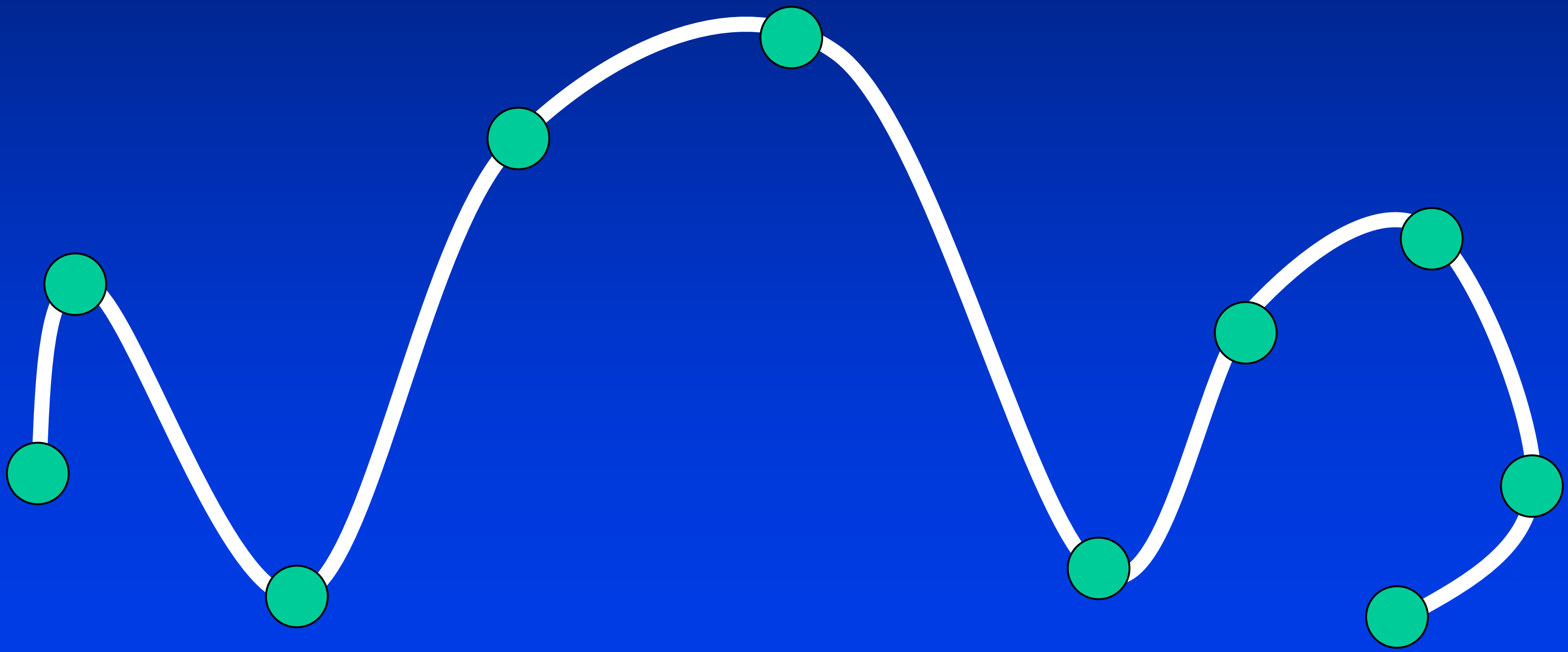
- What is the image of these basis functions?
- Polynomial curve can be defined by

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

- Observations
  - More intuitive, easy to control, polynomials

Lagrange Curve

- Point interpolation



Lagrange Curves

- Curve

$$\mathbf{c}(u) = \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \mathbf{a} \end{bmatrix} L_0^n(u) + \ldots + \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \mathbf{a} \end{bmatrix} L_n^n(u)$$

- Lagrange polynomials of degree n: $L_i^n(u)$
- Knot sequence: $u_0, \ldots, u_n$
- Kronecker delta: $L_i^n(u_j) = \delta_{ij}$
- The curve interpolate all the data point, but unwanted oscillation

Lagrange Basis Functions

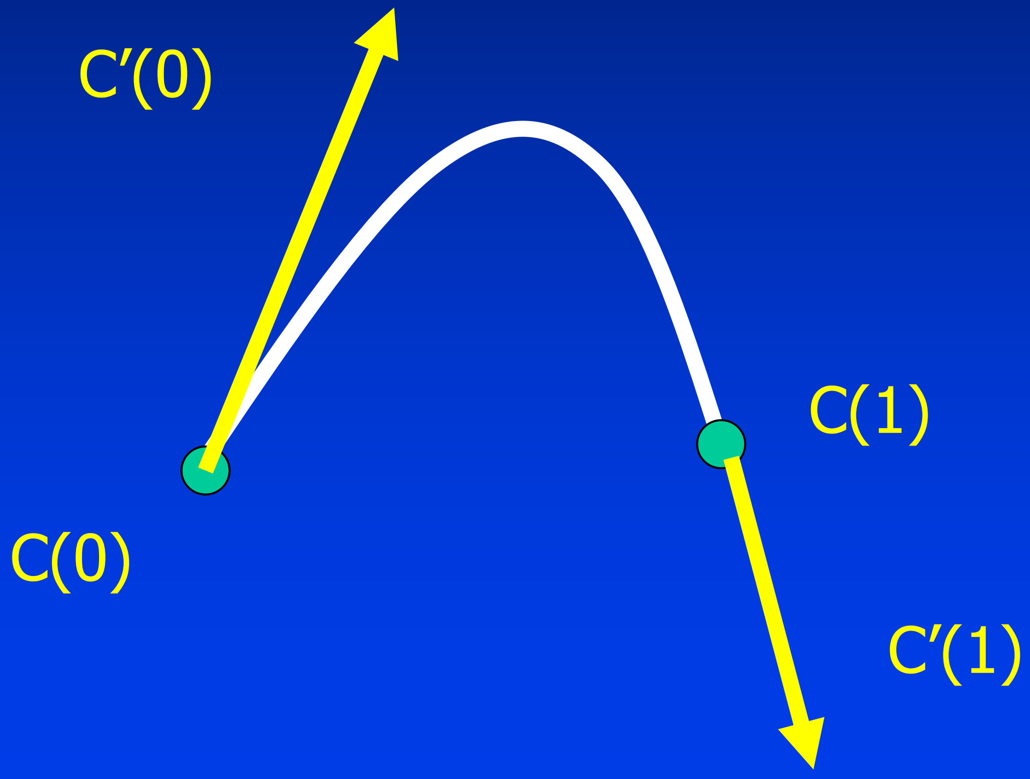
$$L_i^n(u_j) = \begin{cases} 1 & i = j (i, j = 0,1,\ldots,n) \\ 0 & Otherwise \end{cases}$$

$$L_0^n(u) = \frac{(u-u_1)(u-u_2)\ldots(u-u_n)}{(u_0-u_1)(u_0-u_2)\ldots(u_0-u_n)}$$

$$L_i^n(u) = \frac{(u-u_0)\ldots(u-u_{i-1})(u-u_{i+1})\ldots(u-u_n)}{(u_i-u_0)\ldots(u_i-u_{i-1})(u_i-u_{i+1})\ldots(u_i-u_n)}$$

$$L_n^n(u) = \frac{(u-u_0)\ldots(u-u_{n-2})(u-u_{n-1})}{(u_n-u_0)\ldots(u_n-u_{n-2})(u_n-u_{n-1})}$$

Cubic Hermite Splines



Cubic Hermite Curve

- Hermite curve

$$\mathbf{c}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$

- Two end-points and two tangents at end-points

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

- Matrix inversion

$$x(u) = U \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix}$$

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

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

Hermite Curve

- Basis functions

$$f_1(u) = 2u^3 - 3u^2 + 1$$
$$f_2(u) = -2u^3 + 3u^2$$
$$f_3(u) = u^3 - 2u^2 + u$$
$$f_4(u) = u^3 - u^2$$

- Display the image of these basis functions and the Hermite curve itself

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

Cubic Hermite Splines

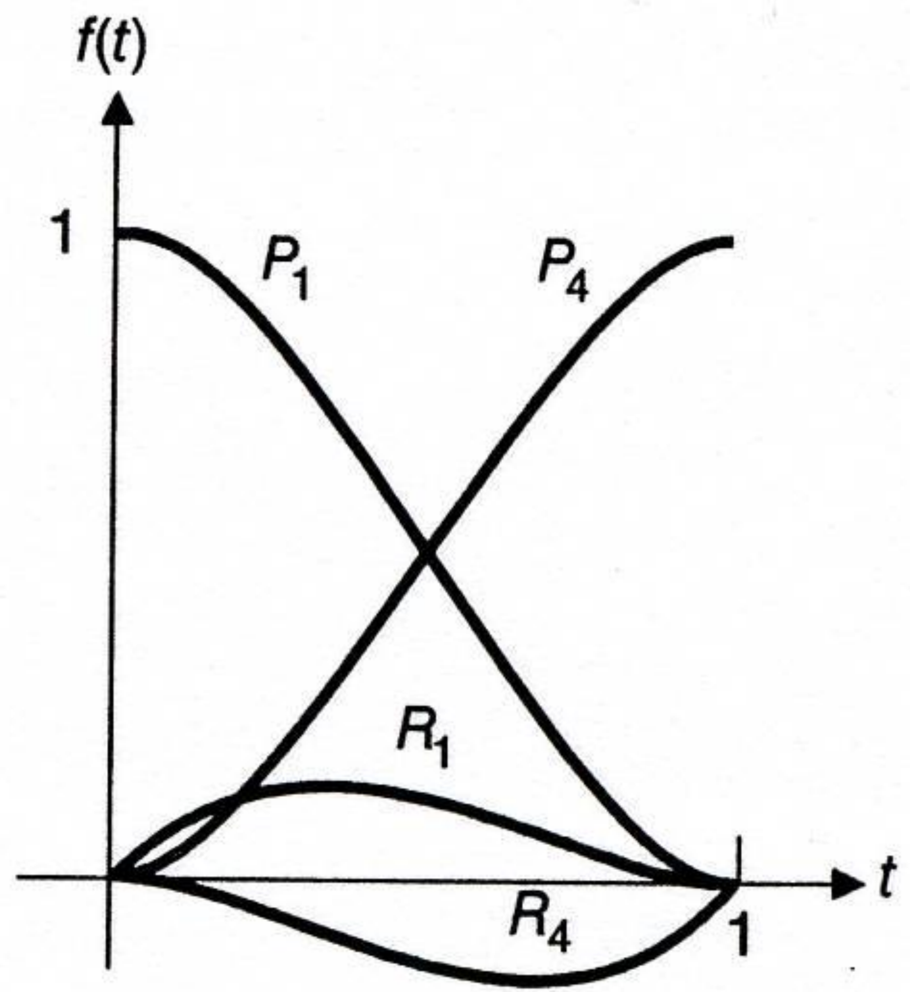
- Two vertices and two tangent vectors:

$$\mathbf{c}(0) = \mathbf{v}_0, \mathbf{c}(1) = \mathbf{v}_1;$$
$$\mathbf{c}^{(1)}(0) = \mathbf{d}_0, \mathbf{c}^{(1)}(1) = \mathbf{d}_1;$$

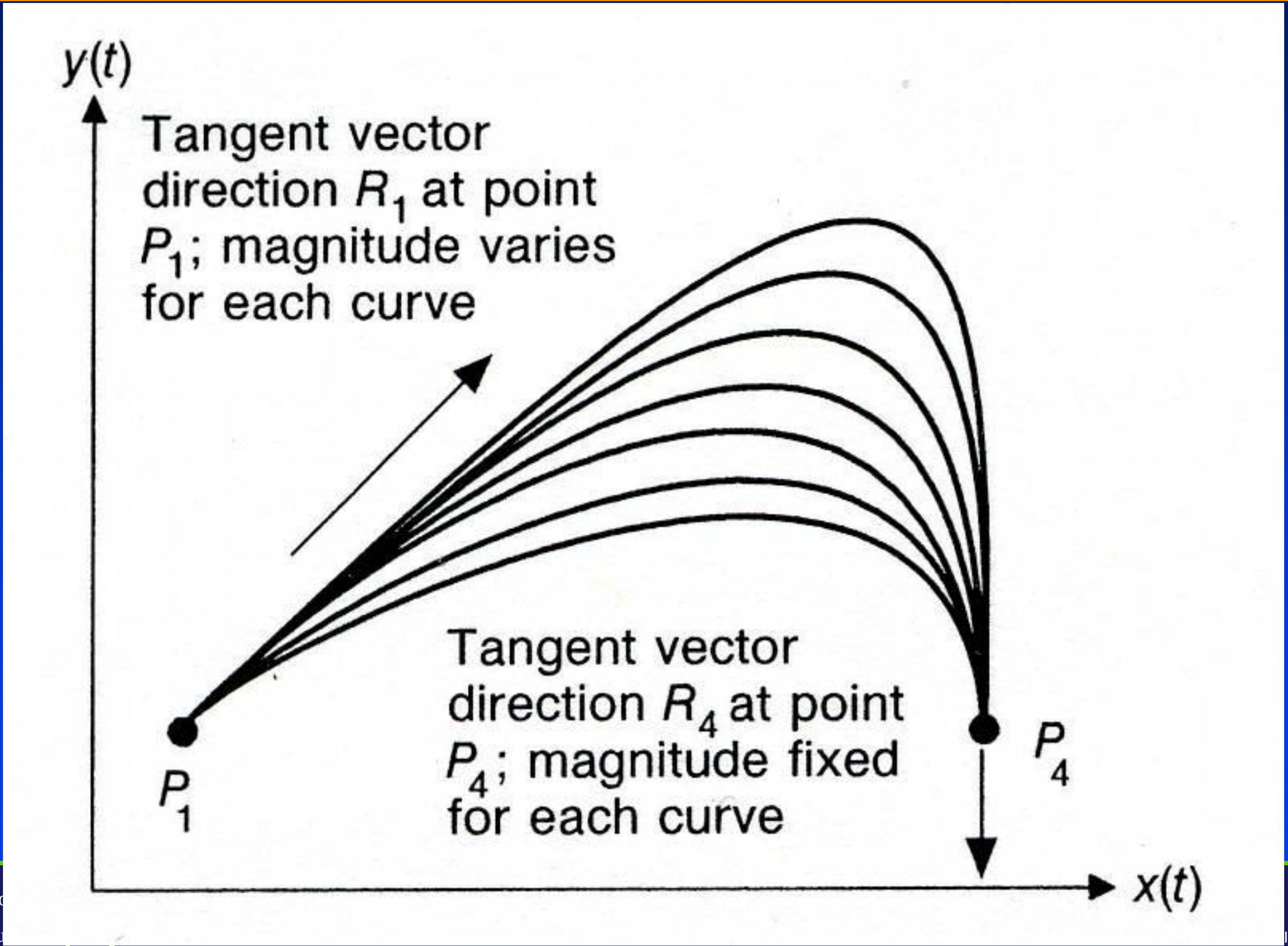
- Hermite curve

$$\mathbf{c}(u) = \mathbf{v}_0 H_0^3(u) + \mathbf{v}_1 H_1^3(u) + \mathbf{d}_0 H_2^3(u) + \mathbf{d}_1 H_3^3(u);$$
$$H_0^3(u) = f_1(u), H_1^3(u) = f_2(u), H_2^3(u) = f_3(u), H_3^3(u) = f_4(u)$$

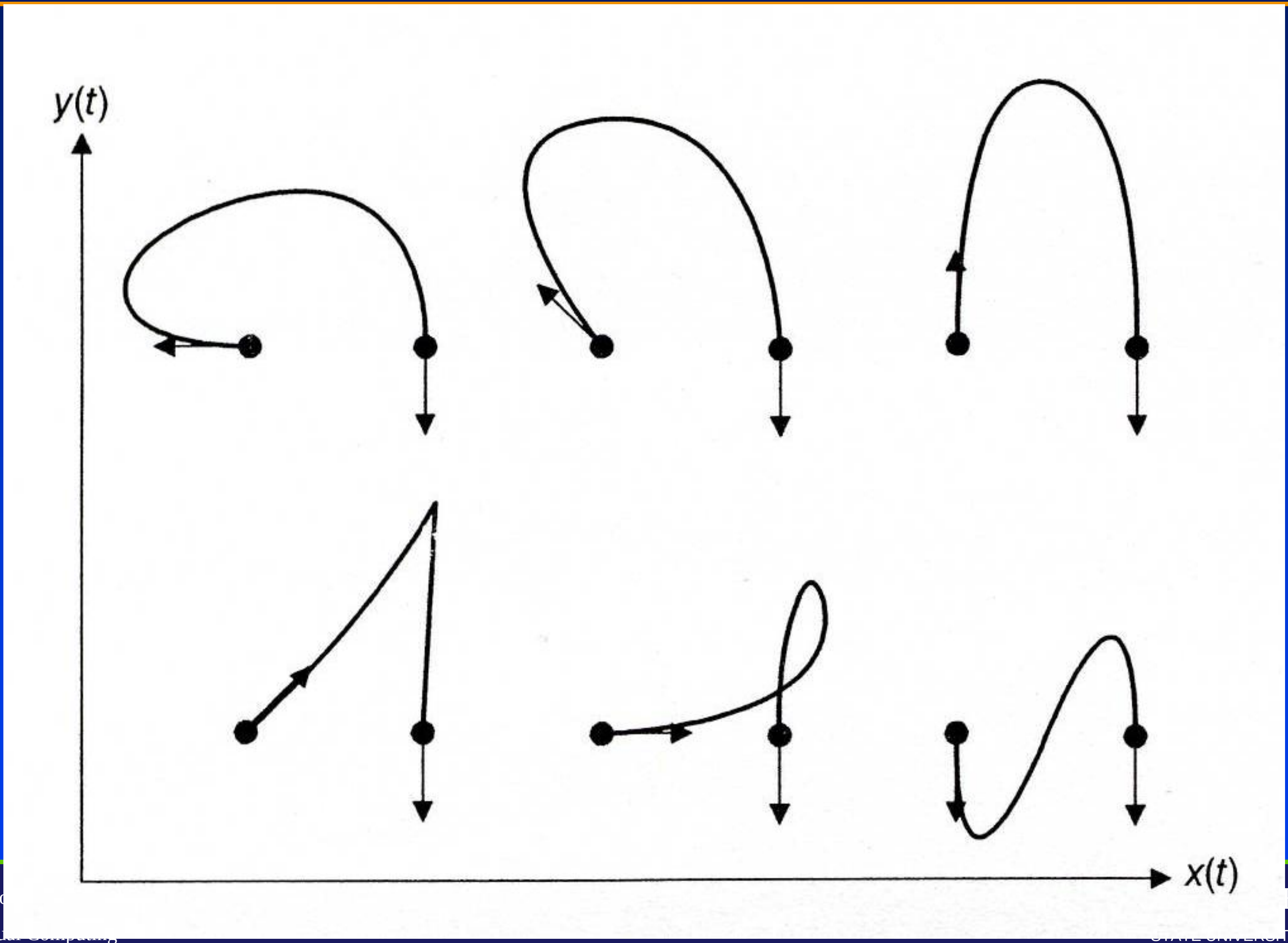
Hermite Basis Functions



Varying the Magnitude of the Tangent Vector



Varying the Direction of the Tangent Vector



Hermite Splines

- Higher-order polynomials

$$\mathbf{c}(u) = \mathbf{v}_0^0 H_0^n(u) + \mathbf{v}_0^1 H_1^n(u) + \ldots + \mathbf{v}_0^{(n-1)/2} H_{(n-1)/2}^n(u)$$
$$+ \mathbf{v}_1^{(n-1)/2} H_{(n+1)/2}^n(u) + \ldots + \mathbf{v}_1^1 H_{(n-1)}^n(u) + \mathbf{v}_1^0 H_n^n(u);$$
$$\mathbf{v}_0^i = \mathbf{c}^{(i)}(0), \mathbf{v}_1^i = \mathbf{c}^{(i)}(1), i = 0, \ldots (n-1)/2;$$

- Note that, n is odd!
- Geometric intuition
- Higher-order derivatives are required

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

Variations of Hermite Curve

- Variations of Hermite curves

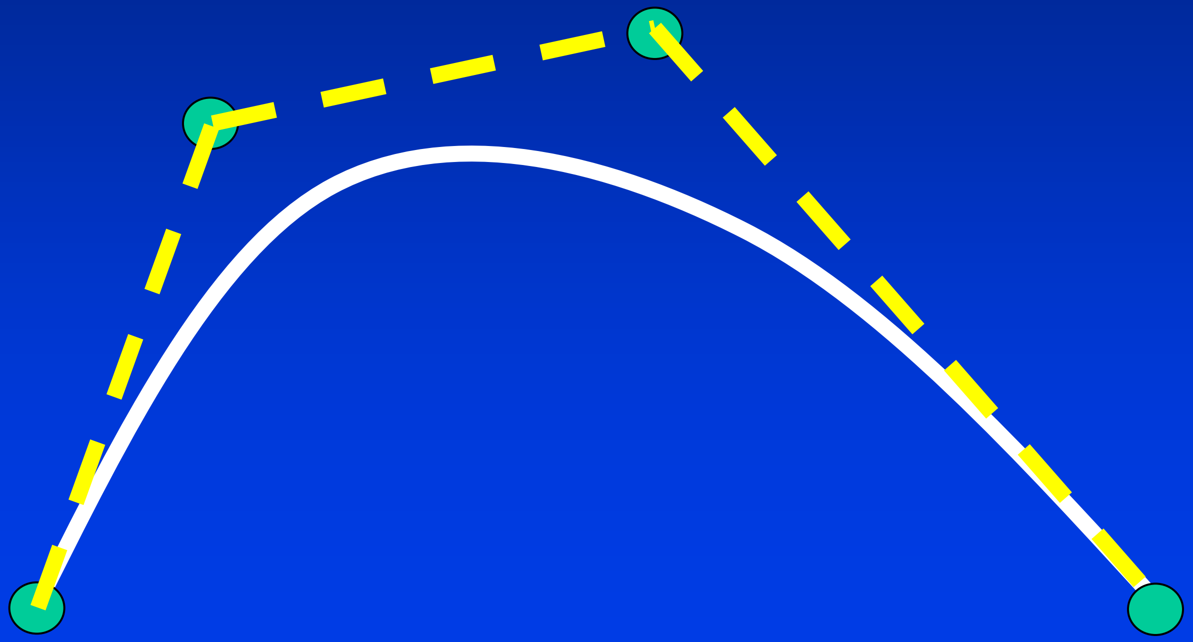
$$\mathbf{p}_0 = \mathbf{c}(0)$$
$$\mathbf{p}_3 = \mathbf{c}(1)$$
$$\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0), \mathbf{p}_1 = \mathbf{p}_0 + \mathbf{c}'(0)/3$$
$$\mathbf{c}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2), \mathbf{p}_2 = \mathbf{p}_3 - \mathbf{c}'(1)/3$$

- In matrix form (x-component only)

$$\begin{bmatrix} \mathbf{c}(0)_x \\ \mathbf{c}(1)_x \\ \mathbf{c}'(0)_x \\ \mathbf{c}'(1)_x \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \end{bmatrix}$$

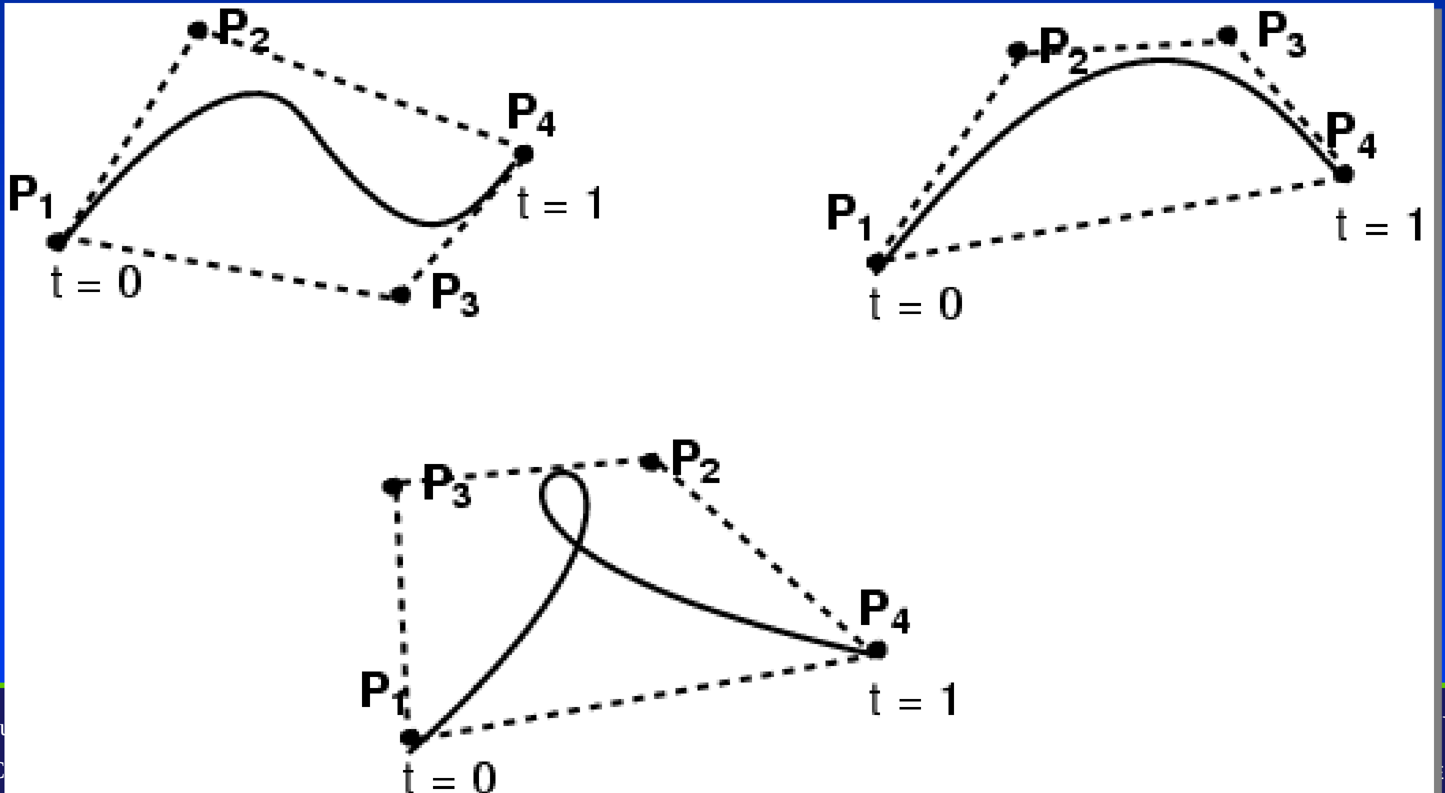
Cubic Bezier Curves

- 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 $(\mathbf{P_0}\text{-}\mathbf{P_1})$ and at $\mathbf{P_4}$ to $(\mathbf{P_4}\text{-}\mathbf{P_3})$



Curve Mathematics (Cubic)

- Bezier curve

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

- Control points and basis functions

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

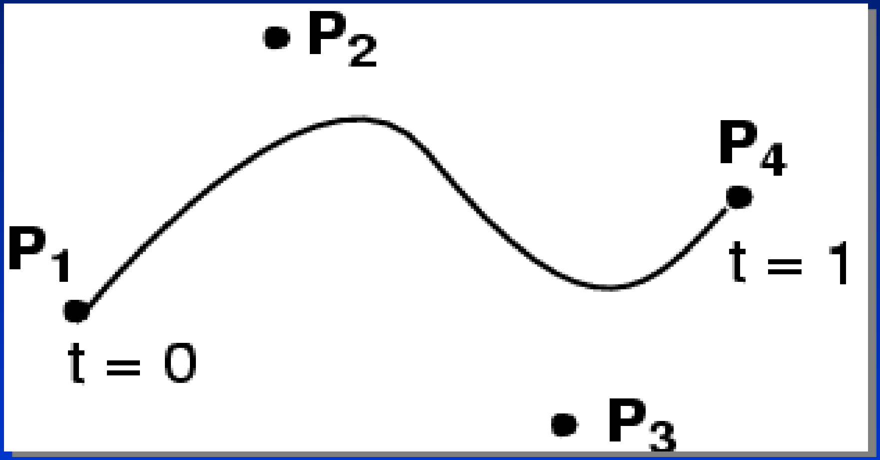
$$B_1^3(u) = 3u(1-u)^2$$

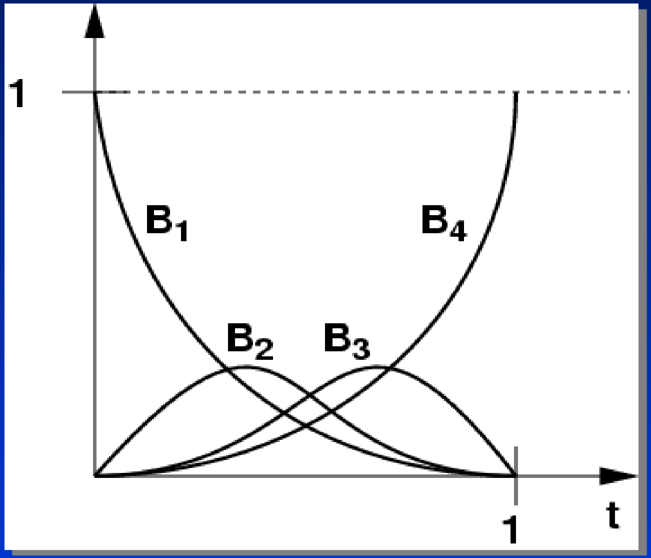
$$B_2^3(u) = 3u^2(1-u)$$

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

- Image and properties of basis functions

Cubic Bézier Basis Functions

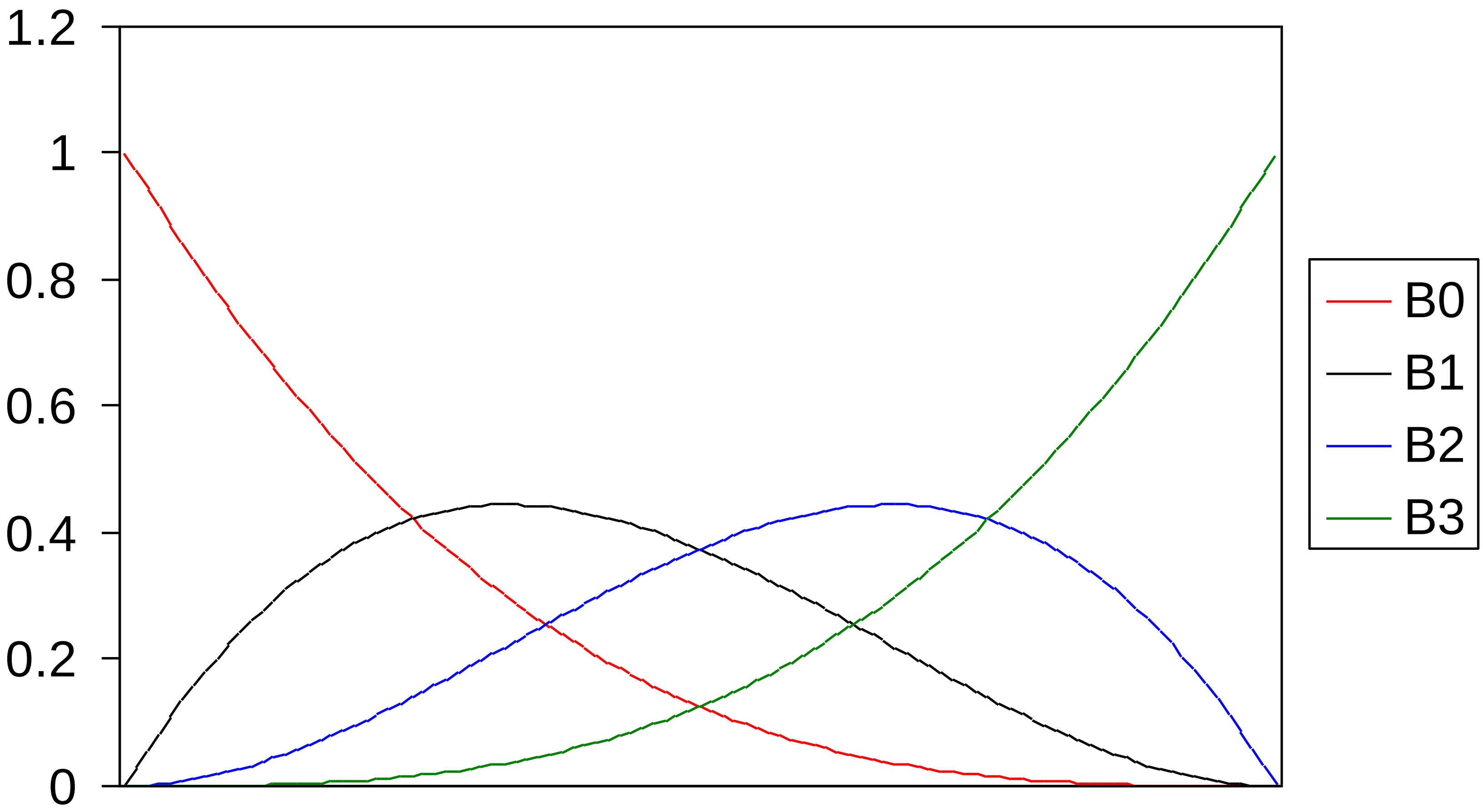




$$B_1(t) = (1-t)^3;\ B_2(t) = 3t(1-t)^2;\ B_3(t) = 3t^2(1-t);\ B_4(t) = t^3$$

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

The Bernstein Polynomials (n=3)



Recursive Evaluation

- Recursive linear interpolation

$$(1-u) \qquad (u)$$

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

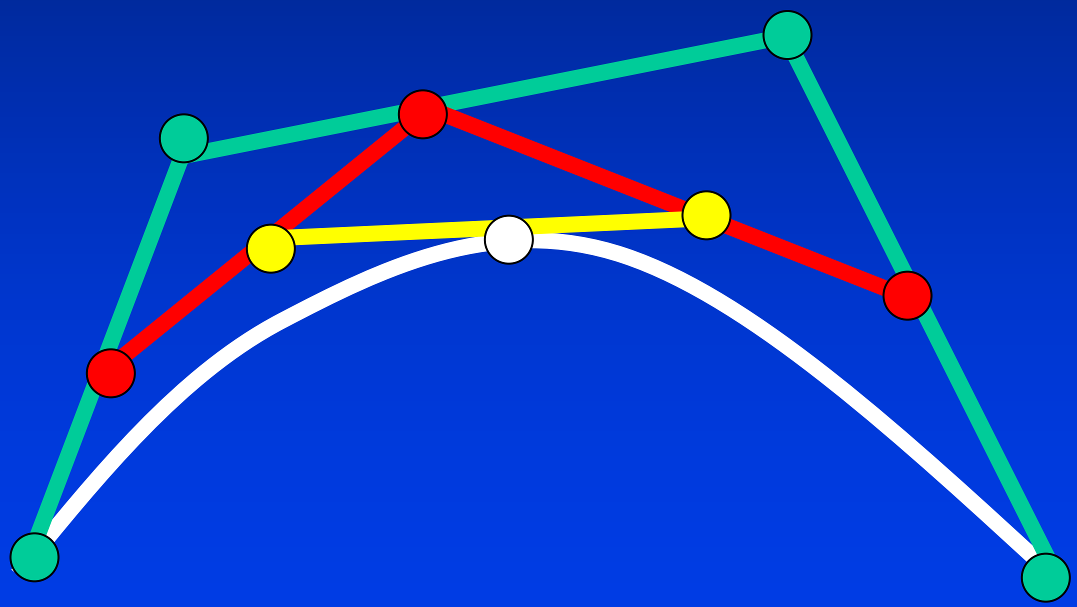
$$\mathbf{p}_0^1 \qquad \mathbf{p}_1^1 \qquad \mathbf{p}_2^1$$

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

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

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

Derivatives

- Tangent vectors can easily evaluated at the end-points $\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0); \mathbf{c}'(1) = (\mathbf{p}_3 - \mathbf{p}_2)$
- Second derivatives at end-points can also be easily computed:

$$\mathbf{c}^{(2)}(0) = 2 \times 3((\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)) = 6(\mathbf{p}_2 - 2\mathbf{p}_1 + \mathbf{p}_0)$$

$$\mathbf{c}^{(2)}(1) = 2 \times 3((\mathbf{p}_3 - \mathbf{p}_2) - (\mathbf{p}_2 - \mathbf{p}_1)) = 6(\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1)$$

Derivative Curve

- The derivative of a cubic Bezier curve is a quadratic Bezier curve

$$\mathbf{c}'(u) = -3(1-u)^2\mathbf{p}_0 + 3((1-u)^2 - 2u(1-u))\mathbf{p}_1 + 3(2u(1-u) - u^2)\mathbf{p}_2 + 3u^2\mathbf{p}_3 =$$

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

More Properties (Cubic)

- Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

$$\mathbf{c}(v), v \in [0, u]$$
$$\mathbf{c}(v), v \in [u, 1]$$
$$\mathbf{c}_l(u), u \in [0,1]$$
$$\mathbf{c}_r(u), u \in [0,1]$$

- The control points for the left and the right are

$$\mathbf{P}_0^0, \mathbf{P}_0^1, \mathbf{P}_0^2, \mathbf{P}_0^3$$
$$\mathbf{P}_0^3, \mathbf{P}_1^2, \mathbf{P}_2^1, \mathbf{P}_3^0$$

High-Degree Curves

- Generalizing to high-degree curves

$$\begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^{n} \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} u^i$$

- Advantages:
  - Easy to compute, Infinitely differentiable
- Disadvantages:
  - Computationally complex, undulation, undesired wiggles
- How about high-order Hermite? Not natural!!!

Higher-Order Bézier Curves

- > 4 control points
- Bernstein Polynomials as the basis functions

$$B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, \qquad 0 \le i \le n$$

- Every control point affects the entire curve
  - Not simply a local effect
  - More difficult to control for modeling

The Bernstein Polynomials

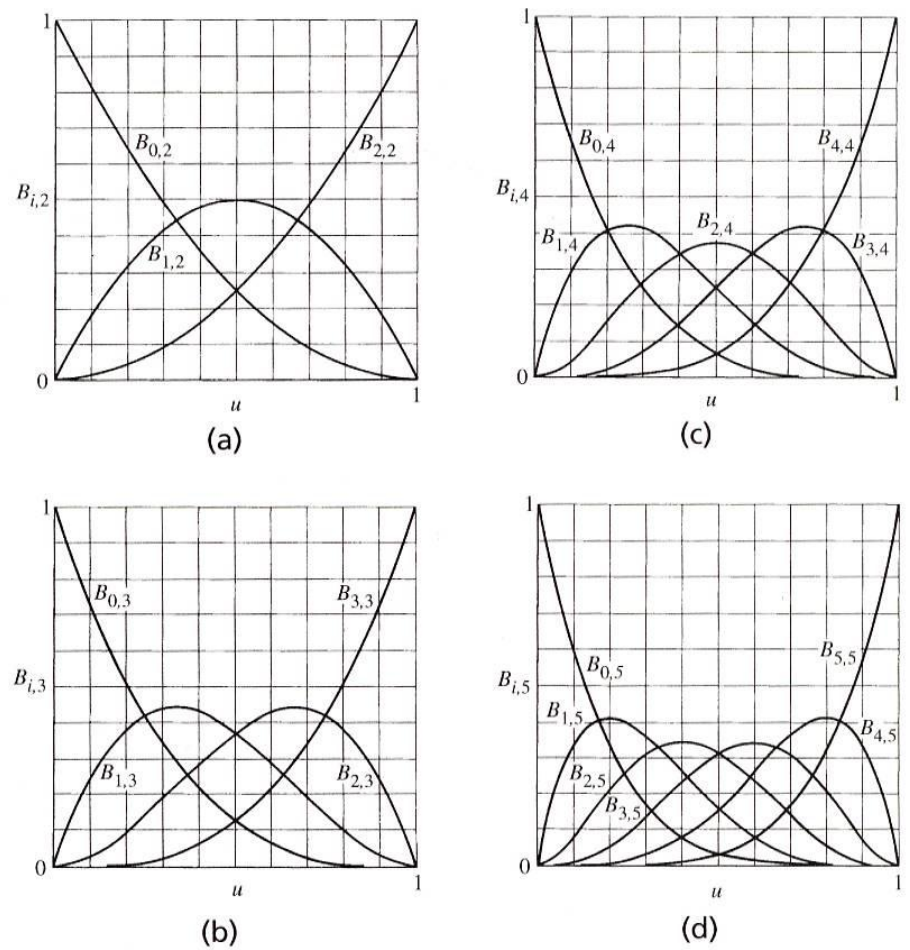


Figure 4.6 Bézier basis functions: (*a*) Three points, $n = 2$; (*b*) Four points, $n = 3$; (*c*) Five points, $n = 4$; (*d*) Six points, $n = 5$.

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:

$$B_i^n(u) = \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i}$$

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{(n-i)!\,i!}$$

Recursive Computation: The De Casteljau Algorithm

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

$$\begin{aligned}
B_i^n(u) &= \binom{n}{i} u^i (1-u)^{n-i} \\
&= \binom{n-1}{/\,i} u^i (1-u)^{n-i} + \binom{n-1}{i-1} u^i (1-u)^{n-i} \\
&= (1-u)B_i^{n-1}(u) + uB_{i-1}^{n-1}(u)
\end{aligned}$$

Recursive Computation

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

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

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

Recursive Computation

- N+1 levels

$$
\begin{array}{ccccccc}
 & (1-u) & & & (u) & & \\
\mathbf{P}_0^0 & & \cdots & & \cdots & & \mathbf{P}_n^0 \\
 & \mathbf{P}_0^1 & & \cdots & & \mathbf{P}_{n-1}^1 & \\
 & \cdots & & \cdots & & \cdots & \\
 & & \mathbf{P}_0^{n-1} & & \mathbf{P}_1^{n-1} & & \\
 & & & \mathbf{P}_0^n = \mathbf{c}(u) & & &
\end{array}
$$

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

More properties

- Recursive subdivision and evaluation
- Symmetry: c(u) and c(1-u) are defined by the same set of point points, but different ordering

$$\mathbf{p}_0, \ldots, \mathbf{p}_n;$$

$$\mathbf{p}_n, \ldots, \mathbf{p}_0$$

Tangents and Derivatives

- End-point tangents: $$\mathbf{c}'(0) = n(\mathbf{p}_1 - \mathbf{p}_0)$$ $$\mathbf{c}'(1) = n(\mathbf{p}_n - \mathbf{p}_{n-1})$$
- I-th derivatives at two end-points depend on $$\mathbf{p}_0, \ldots, \mathbf{p}_i;$$ $$\mathbf{p}_n, \ldots, \mathbf{p}_{n-i}$$
- Derivatives at non-end-points involve all control points

Tangents and Derivatives

End-point tangents:

$$c'(0) = n(p_1 - p_0)$$

$$c'(1) = n(p_n - p_{n-1})$$

i-th derivatives:

$c^{(i)}(0)$ **depends only on** $p_0, \ldots, p_i$

$c^{(i)}(1)$ **depends only on** $p_n, \ldots, p_{n-i}$

Derivatives at non-end-points:

$c^{(i)}(u)$ **involve all control points**

Other Advanced Topics

- Efficient evaluation algorithm
- Differentiation and integration
- Degree elevation
  - Use a polynomial of degree (n+1) to express that of degree (n)
- Composite curves
- Geometric continuity
- Display of curve

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

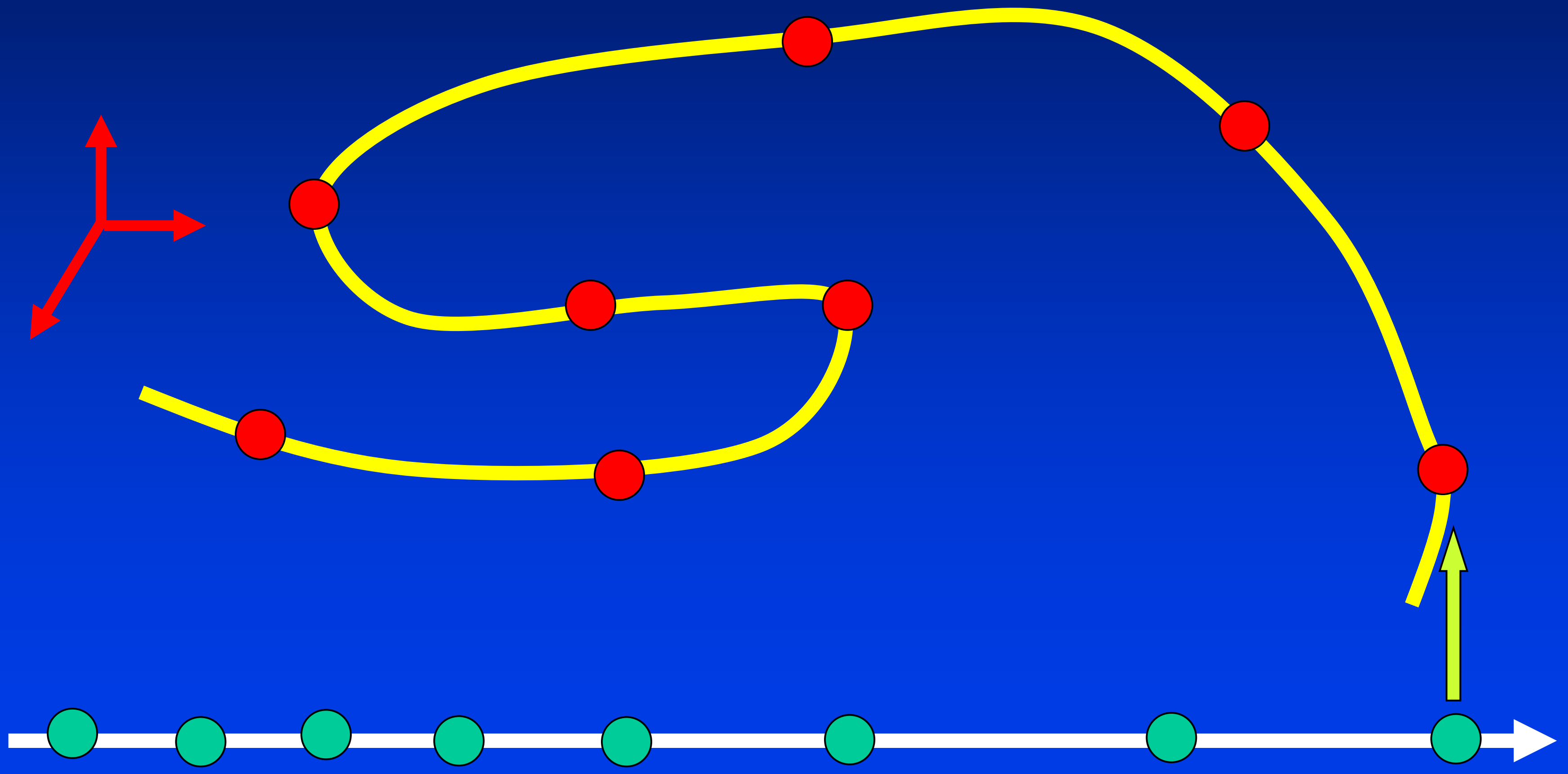
High-Degree polynomials

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

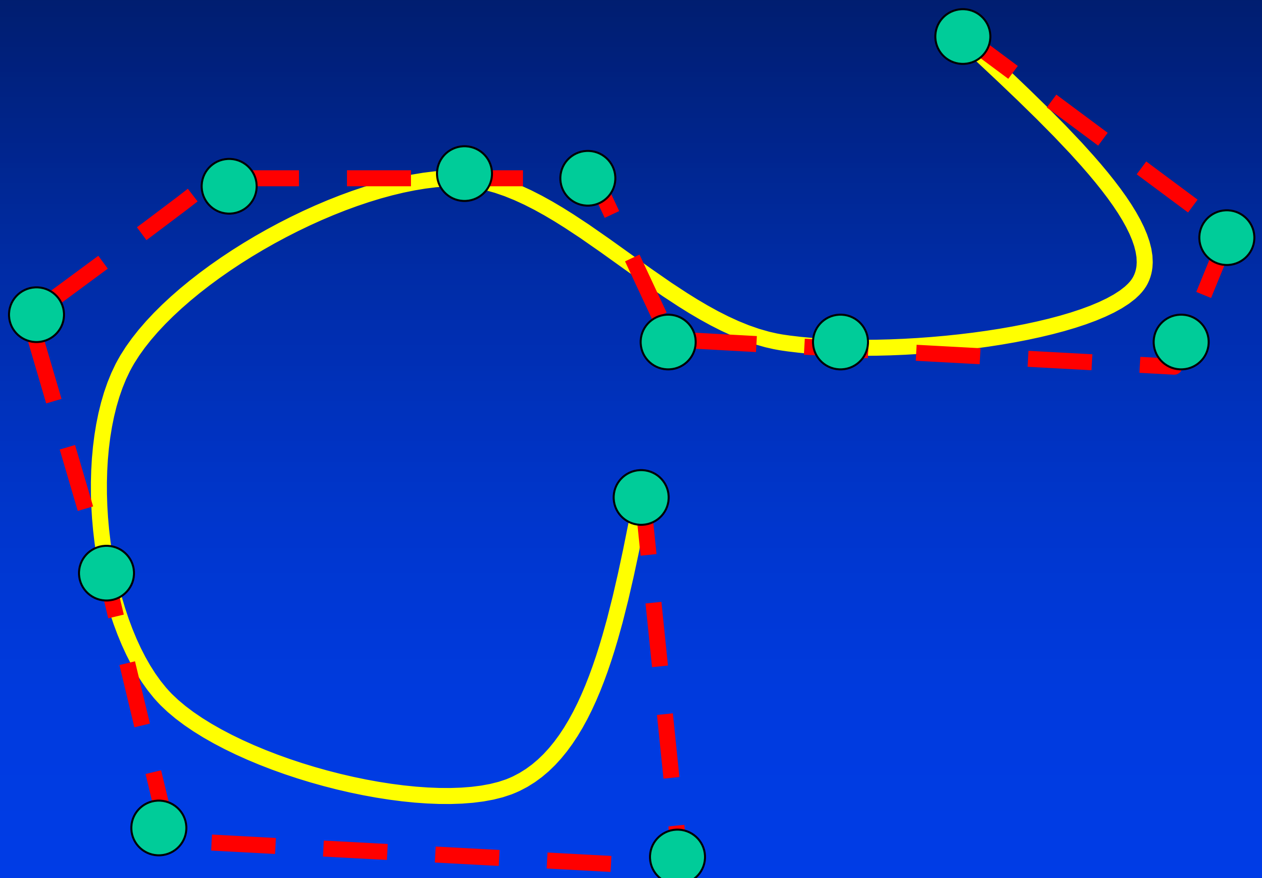
Piecewise Polynomials

- 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

- One of the fundamental concepts.
- Commonly used cases: C^0 ,C^1 ,C^2 , etc.
- C^0 Continuity: Position.
- C^1 Continuity: Velocity.
- C^2 Continuity: Acceleration.

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

Continuity Definitions:

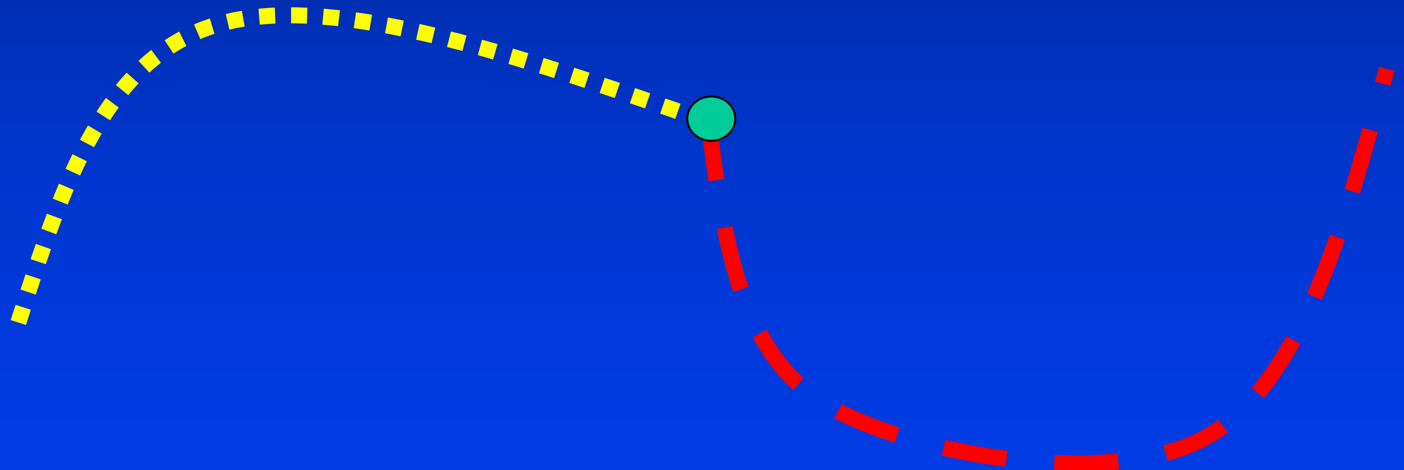
- C^0 continuous
  - curve/surface has no breaks/gaps/holes
  - "watertight"

- C^1 continuous
  - curve/surface derivative is continuous
  - "looks smooth, no facets"

- C^2 continuous
  - curve/surface 2nd derivative is continuous
  - Actually important for shading

Positional Continuity

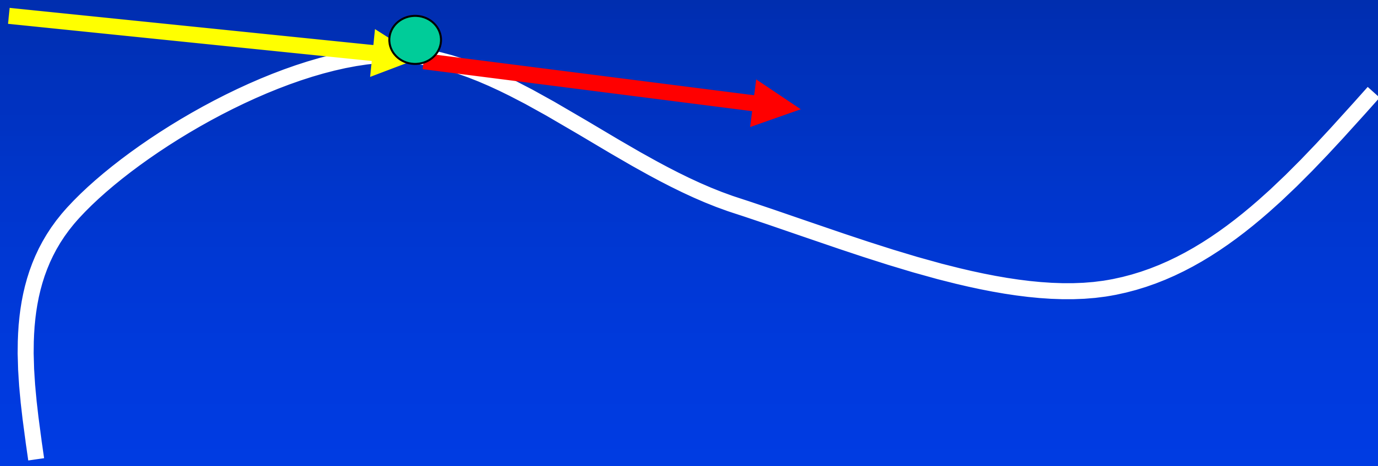
$$\mathbf{a}(1) = \mathbf{b}(0)$$



Derivative Continuity

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

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



General Continuity

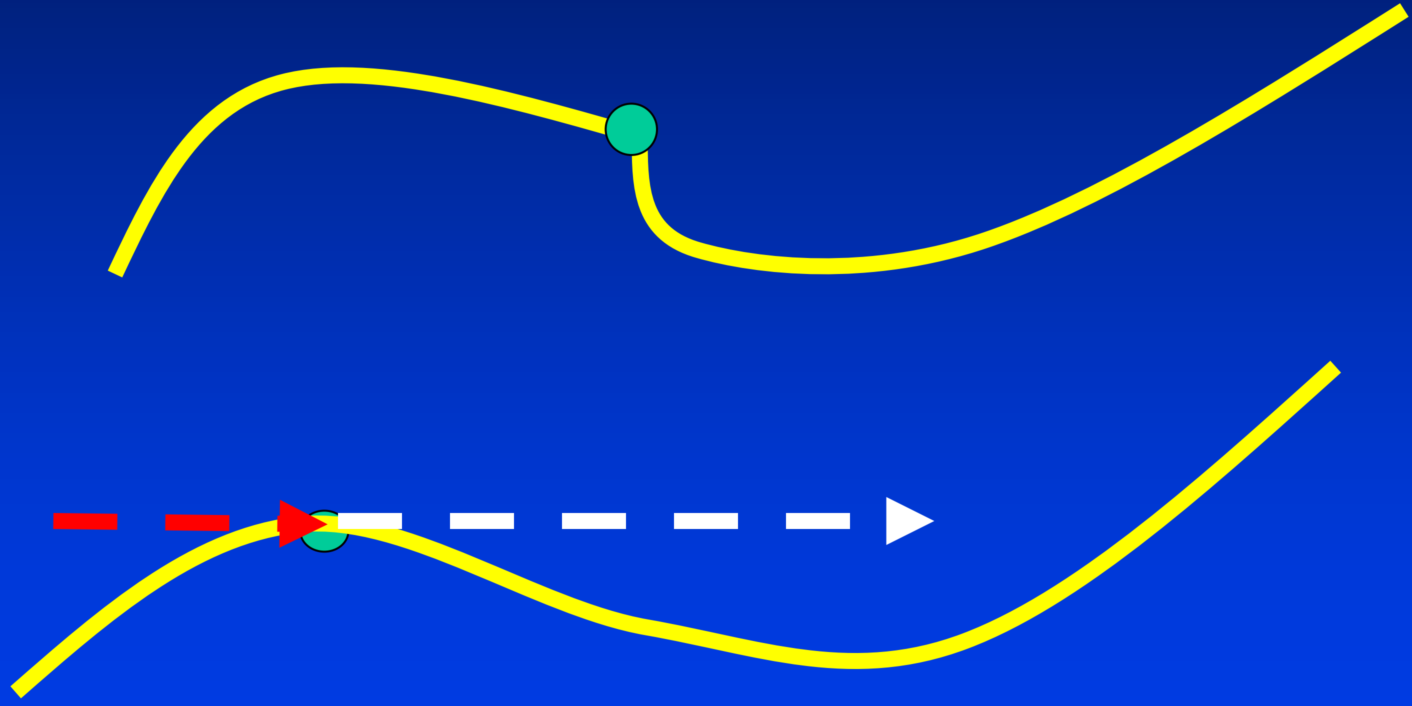
- Cn continuity: derivatives (up to n-th) are the same at the joining point

$$\mathbf{a}^{(i)}(1) = \mathbf{b}^{(i)}(0)$$
$$i = 0,1,2,\ldots,n$$

- The prior definition is for parametric continuity
- Parametric continuity depends of parameterization! But, parameterization is not unique!
- Different parametric representations may express the same geometry
- Re-parameterization can be easily implemented
- Another type of continuity: geometric continuity, or Gn

Geometric Continuity

- G0 and G1



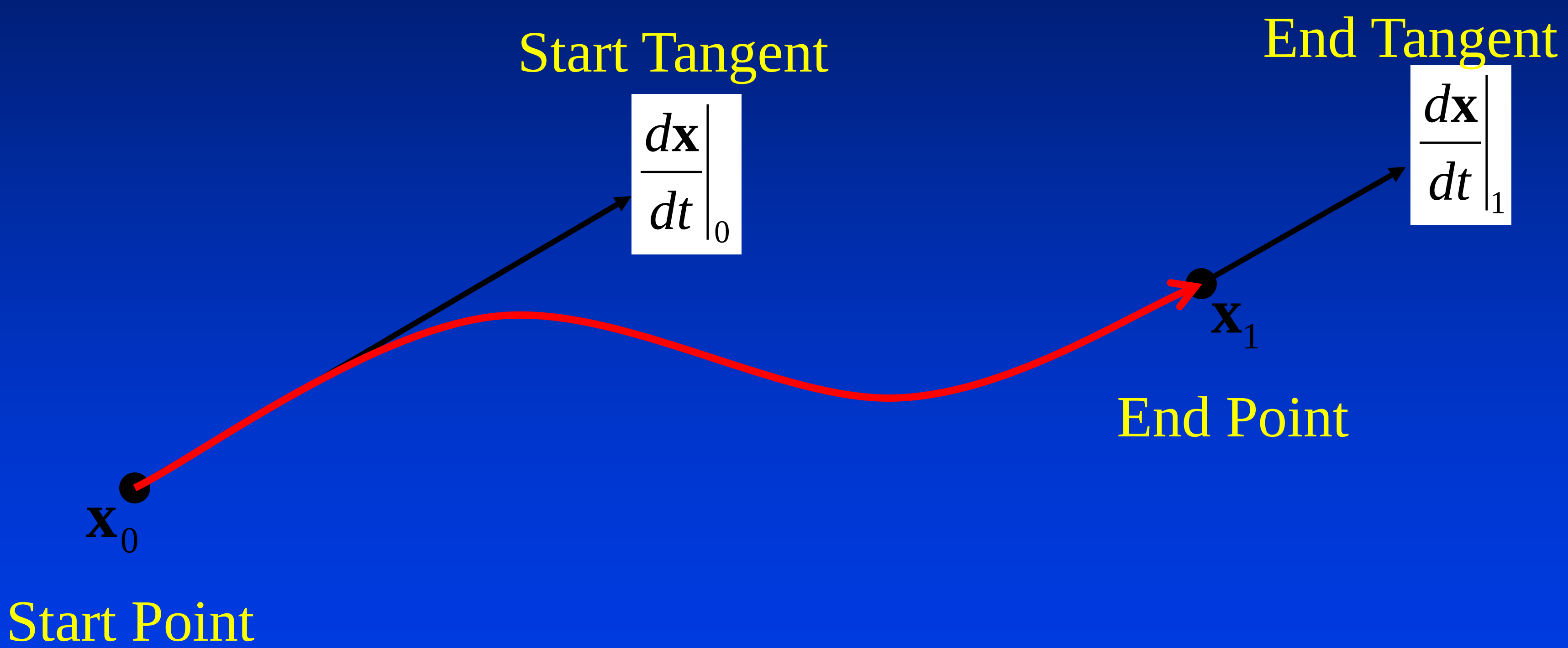
Geometric Continuity

- Depend on the curve geometry
- DO NOT depend on the underlying parameterization
- G0: the same joint
- G1: two curve tangents at the joint align, but may (or may not) have the same magnitude
- G1: it is C1 after the reparameterization
- Which condition is stronger???
- Examples

Hermite Spline

- A *Hermite spline* is a curve for which the user provides:
  - The endpoints of the curve
  - The parametric derivatives of the curve at the endpoints (tangent directions with magnitude)
    - The parametric derivatives are *dx/dt, dy/dt, dz/dt*
  - That is enough to define a *cubic* Hermite spline

Control Point Interpretation



Piecewise Hermite Curves

- How to build an interactive system to satisfy various constraints.
- C^0 continuity:

  $a(1) = b(0)$

- C^1 continuity:

  $a(1) = b(0)$

  $a'(1) = b'(0)$

- G^1 continuity:

  $a(1) = b(0)$

  $a'(1) = \alpha b'(0)$

Piecewise Hermite Curves

- How to build an interactive system to satisfy various constraints
- C0 continuity

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

- C1 continuity

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

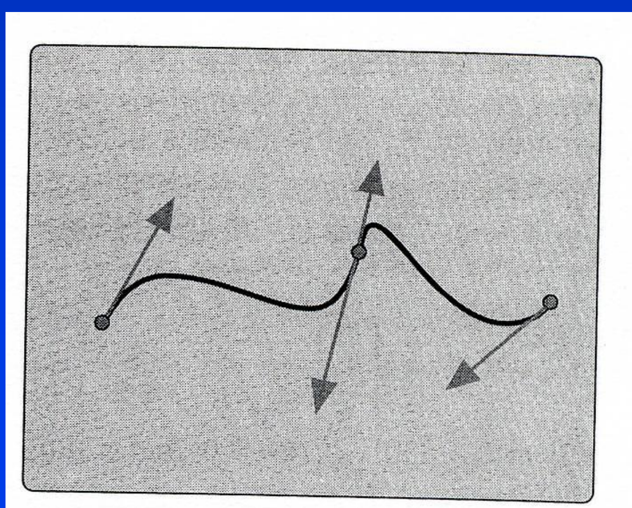
- G1 continuity

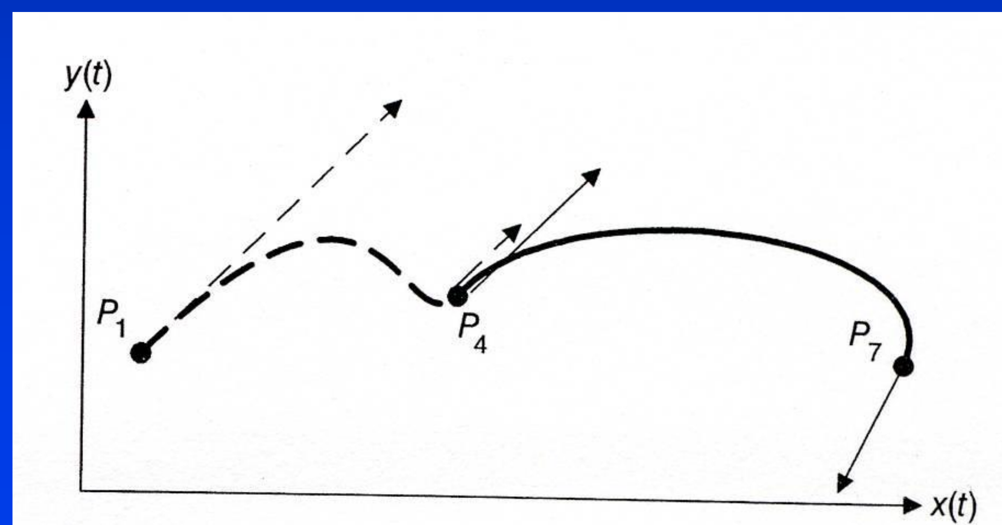
$$\mathbf{a}(1) = \mathbf{b}(0)$$
$$\mathbf{a}'(1) = \alpha\mathbf{b}'(0)$$

Obtaining Geometric Continuity G^1

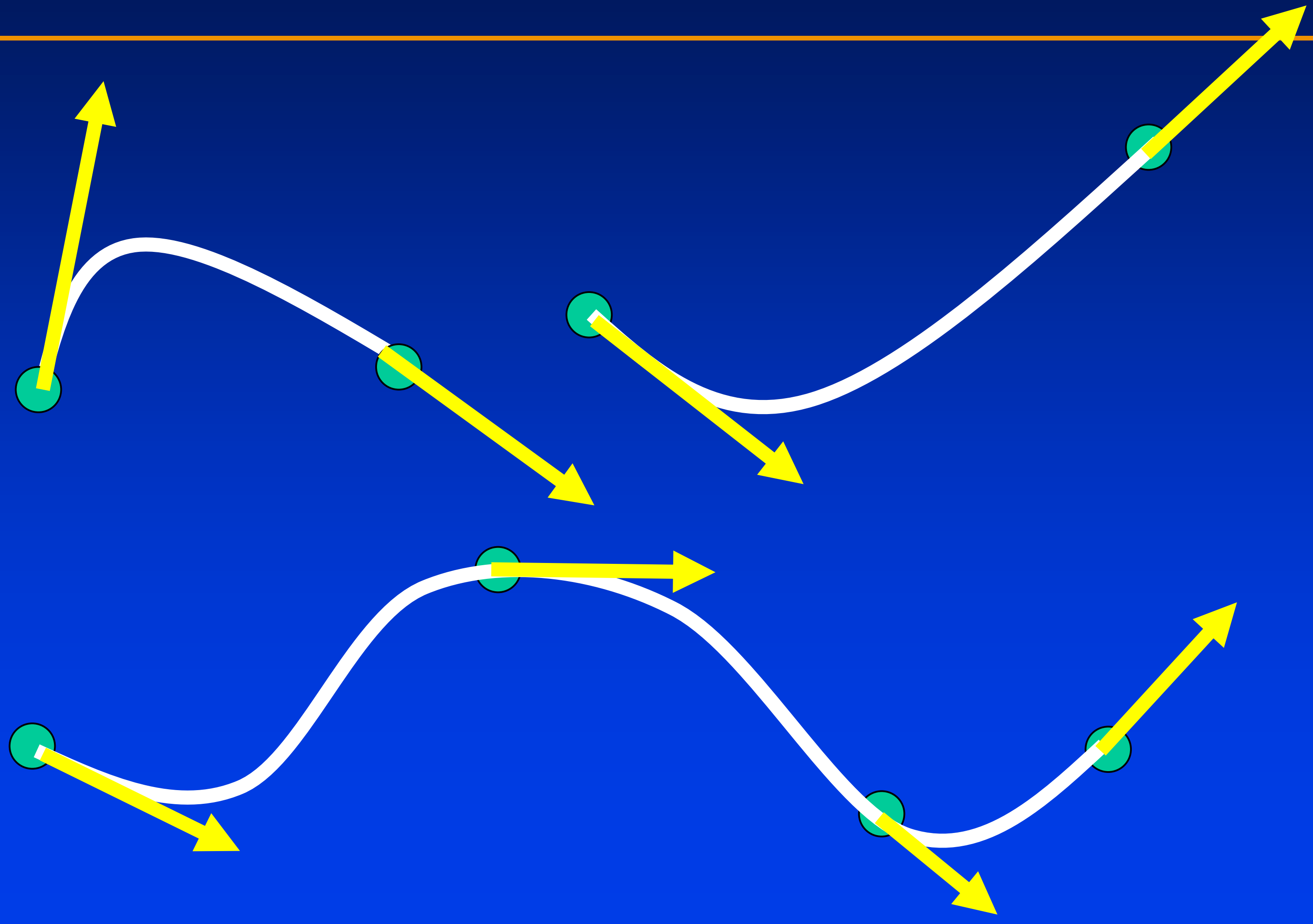
$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} \text{ and } \begin{bmatrix} P_4 \\ P_7 \\ kR_4 \\ R_7 \end{bmatrix}, \text{ with } k > 0.$$

for parametric continuity C^1, $k = 1$





Piecewise Hermite Curves



Hermite Spline

- Say the user provides $$\mathbf{x}_0, \mathbf{x}_1, \left.\frac{d\mathbf{x}_0}{dt}\right|_0, \left.\frac{d\mathbf{x}_1}{dt}\right|_1$$
- A cubic spline has degree 3, and is of the form:
$$x = at^3 + bt^2 + ct + d$$
  - For some constants a, b, c and d derived from the control points, but how?
- We have constraints:
  - The curve must pass through x_0 when $t=0$
  - The derivative must be x'_0 when $t=0$
  - The curve must pass through x_1 when $t=1$
  - The derivative must be x'_1 when $t=1$

Hermite Spline

- Solving for the unknowns gives:

$$
\begin{aligned}
a &= -2x_1 + 2x_0 + x_1' + x_0' \\
b &= 3x_1 - 3x_0 - x_1' - 2x_0' \\
c &= x_0' \\
d &= x_0
\end{aligned}
$$

- Rearranging gives:

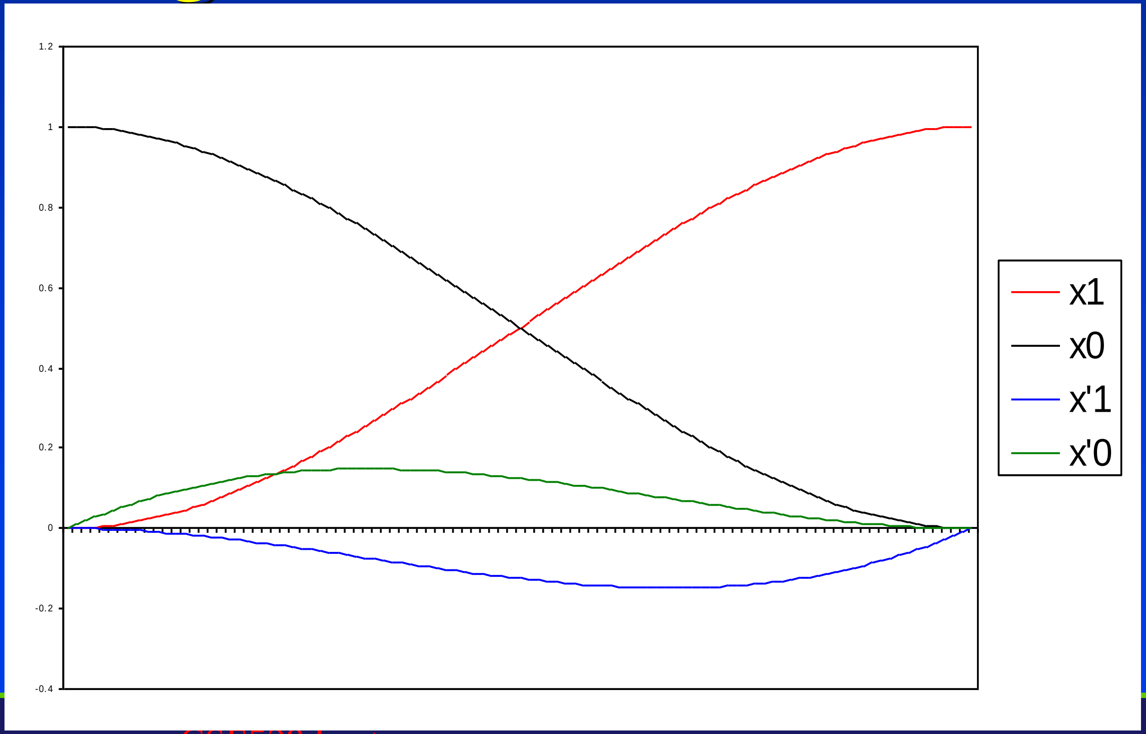
$$
\begin{aligned}
\mathbf{x} = \mathbf{x}_1&(-2t^3 + 3t^2) \\
+ \mathbf{x}_0&(2t^3 - 3t^2 + 1) \\
+ \mathbf{x}_1'&(t^3 - t^2) \\
+ \mathbf{x}_0'&(t^3 - 2t^2 + t)
\end{aligned}
$$

or

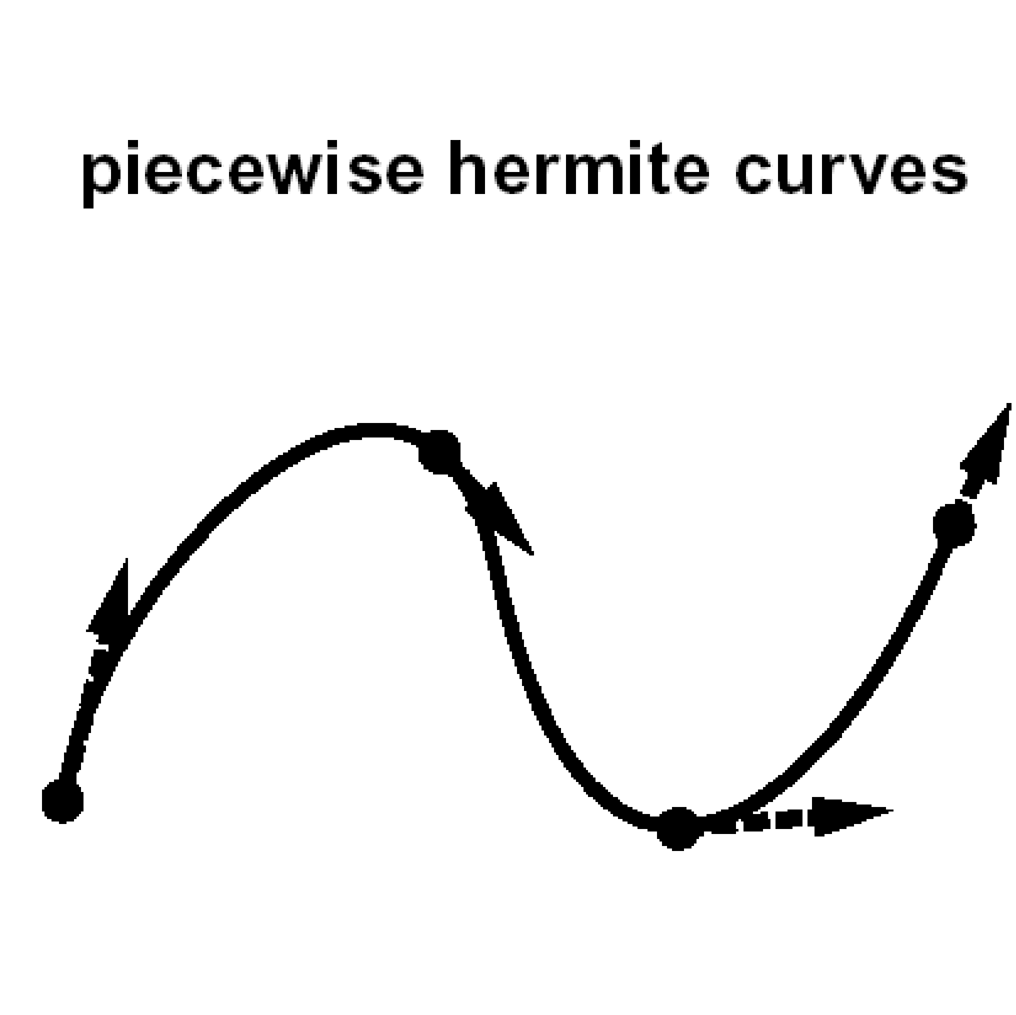
$$
x = \begin{bmatrix} x_1 & x_0 & x_1' & x_0' \end{bmatrix}
\begin{bmatrix} -2 & 3 & 0 & 0 \\ 2 & -3 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \end{bmatrix}
\begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}
$$

Basis Functions

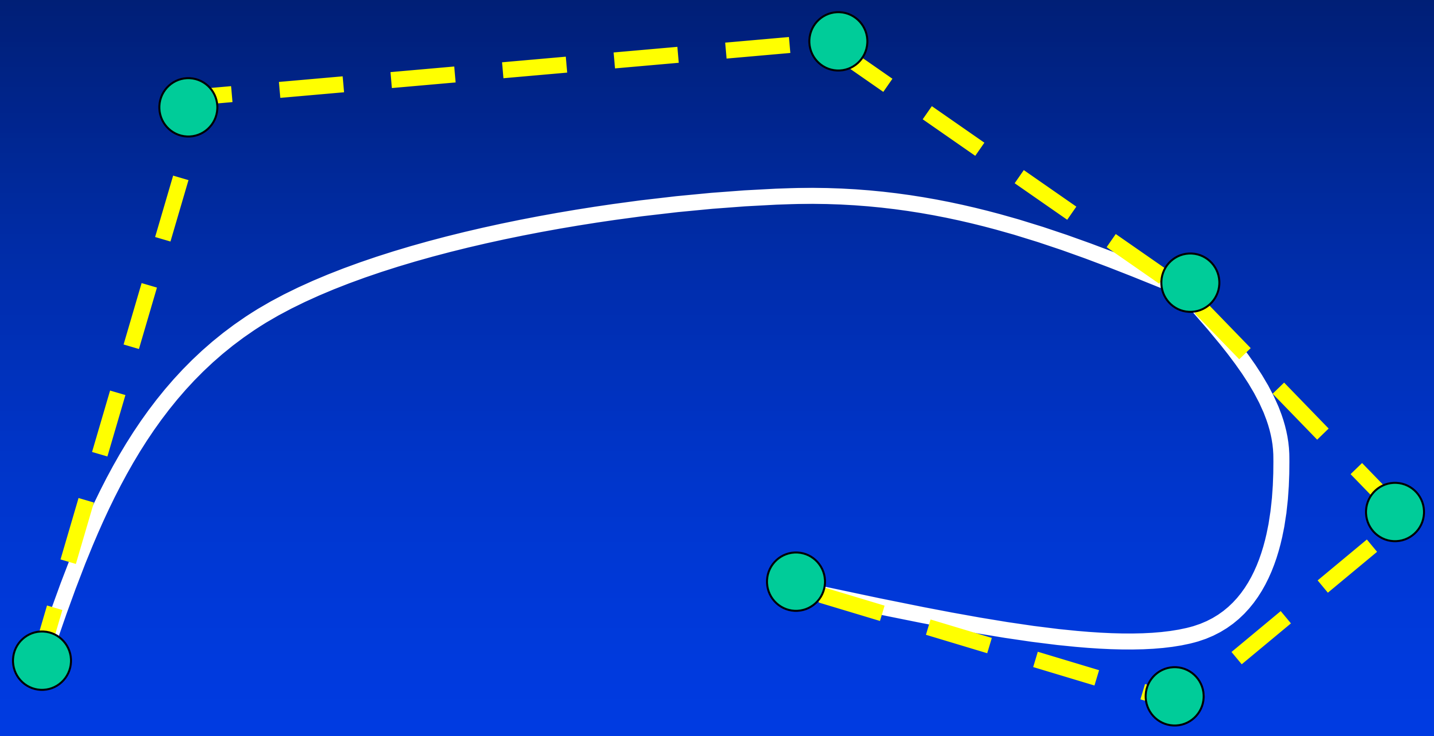
- A point on a Hermite curve is obtained by multiplying each control point by some function and summing



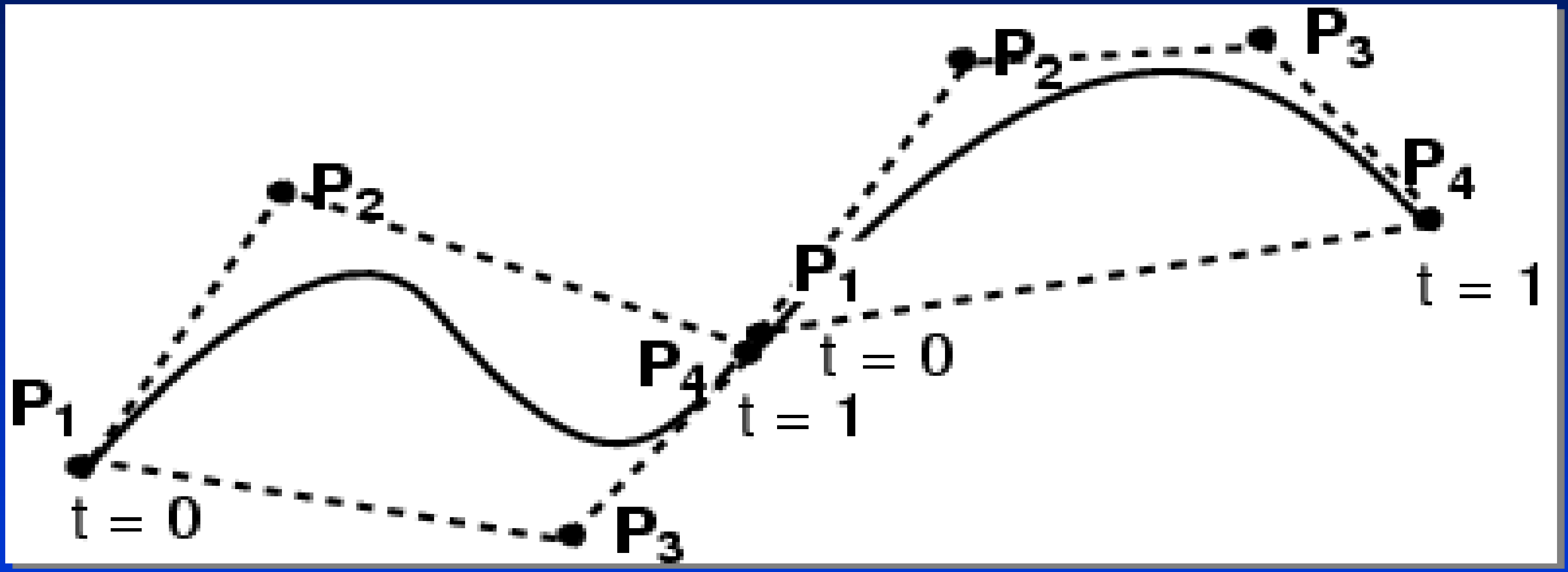
Piecewise Hermite Curves



Piecewise Bezier Curves

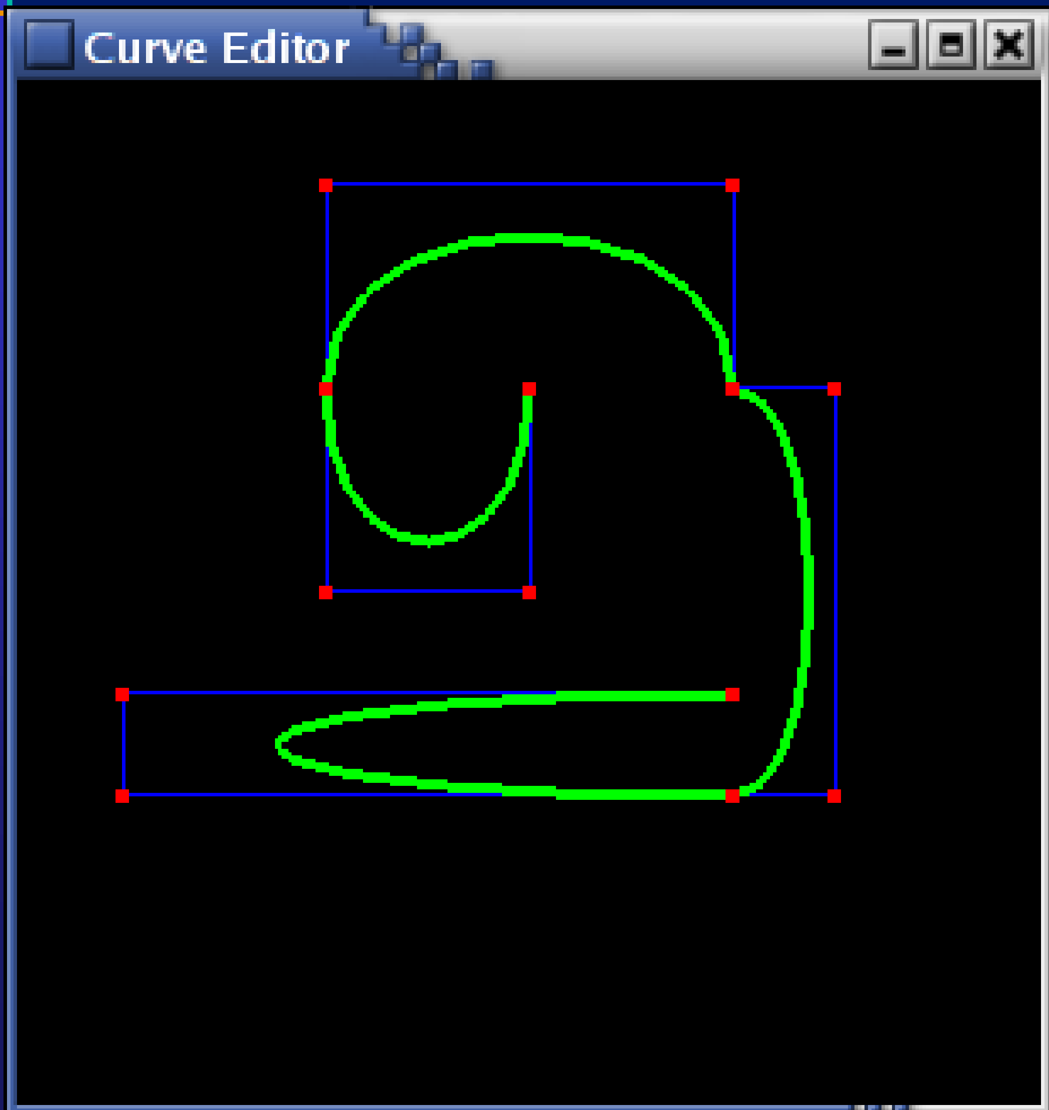


Connecting Cubic Bézier Curves



- 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

Connecting Cubic Bézier Curves



- Where is this curve
  - C^0 continuous?
  - G^1 continuous?
  - C^1 continuous?
- What's the relationship between:
  - the # of control points, and
  - the # of cubic Bézier sub-curves?

Piecewise Bezier Curves

- C0 continuity

$$\mathbf{p}_3 = \mathbf{q}_0$$

- C1 continuity

$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = (\mathbf{q}_1 - \mathbf{q}_0)$$

- G1 continuity

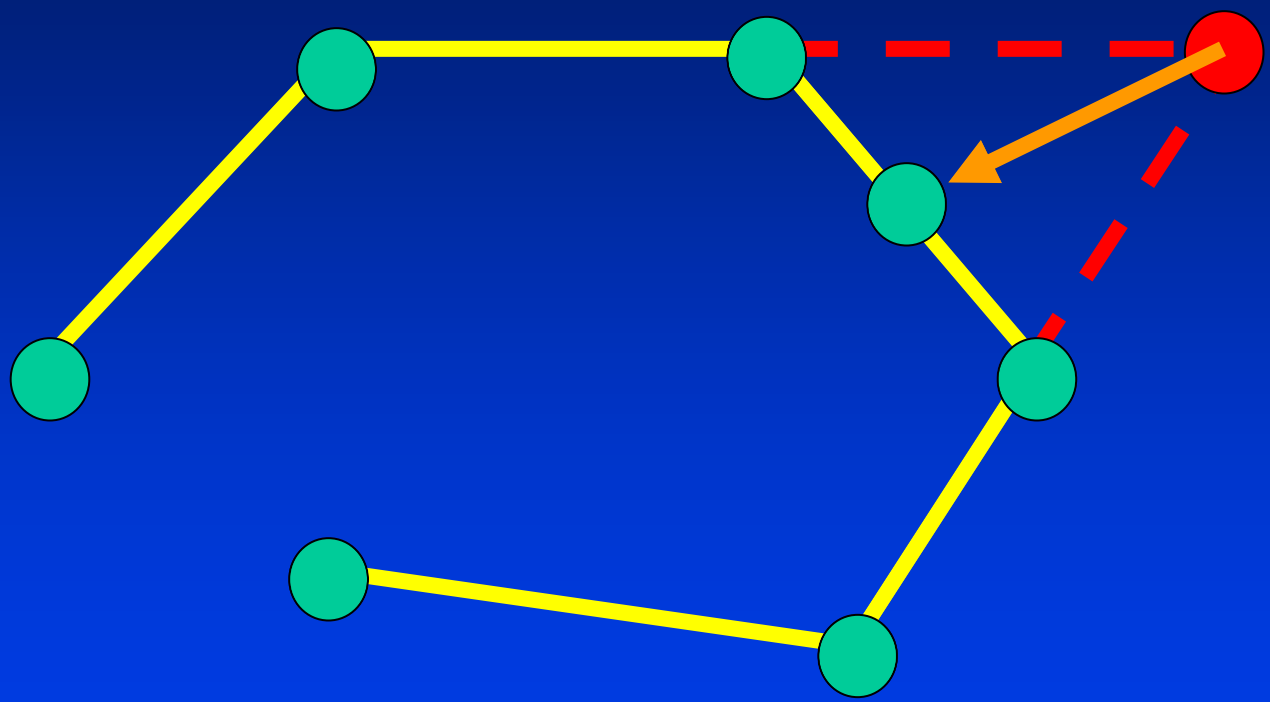
$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = \alpha(\mathbf{q}_1 - \mathbf{q}_0)$$

- C2 continuity

$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = (\mathbf{q}_1 - \mathbf{q}_0)$$
$$\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1 = \mathbf{q}_2 - 2\mathbf{q}_1 + \mathbf{q}_0$$

- Geometric interpretation
- G2 continuity

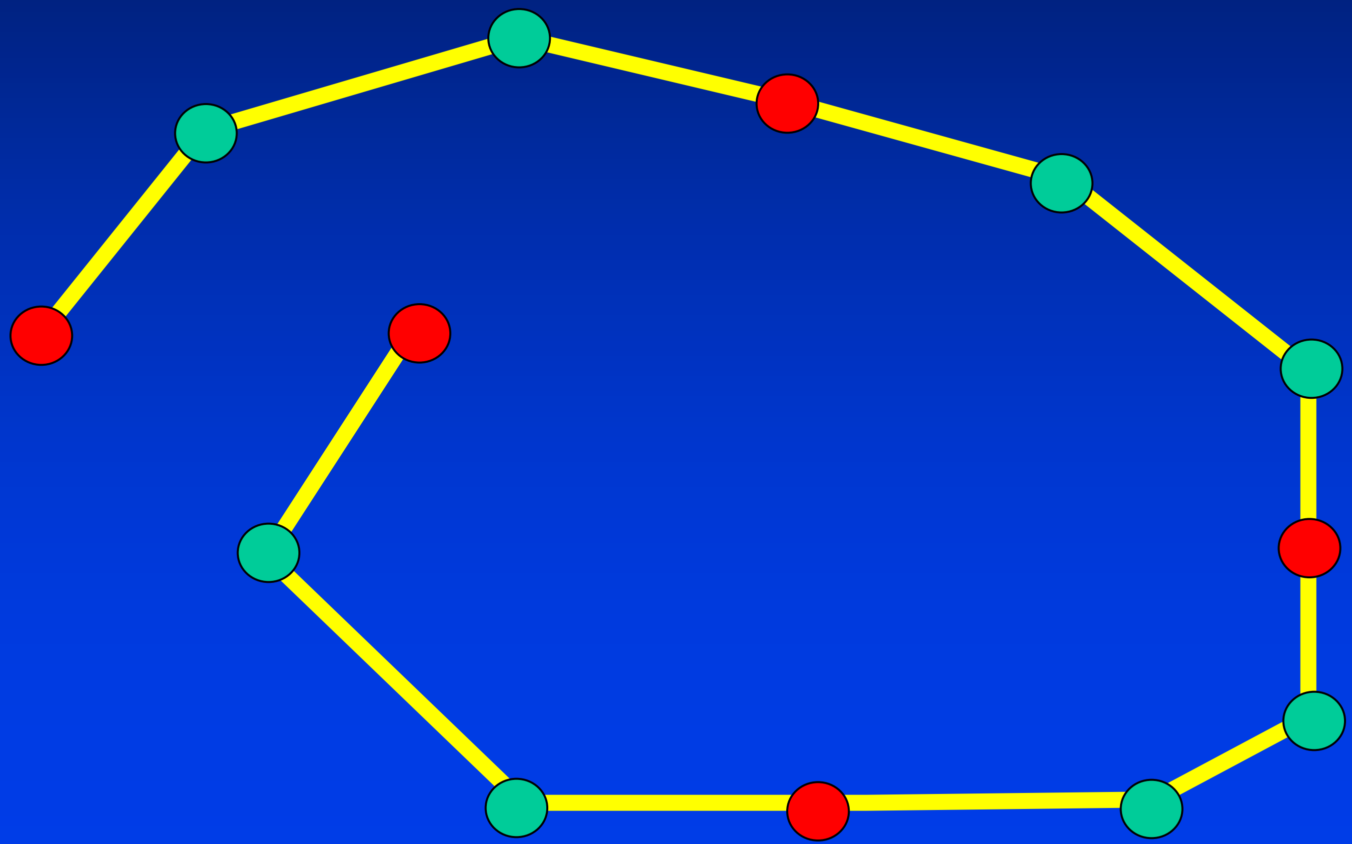
Piecewise C2 Bezier Curves



Continuity Summary

- C0: straightforward, but not enough
- C3: too constrained
- Piecewise curves with Hermite and Bezier representations satisfying various continuity conditions
- Interactive system for C2 interpolating splines using piecewise Bezier curves
- Advantages and disadvantages

C2 Interpolating Splines



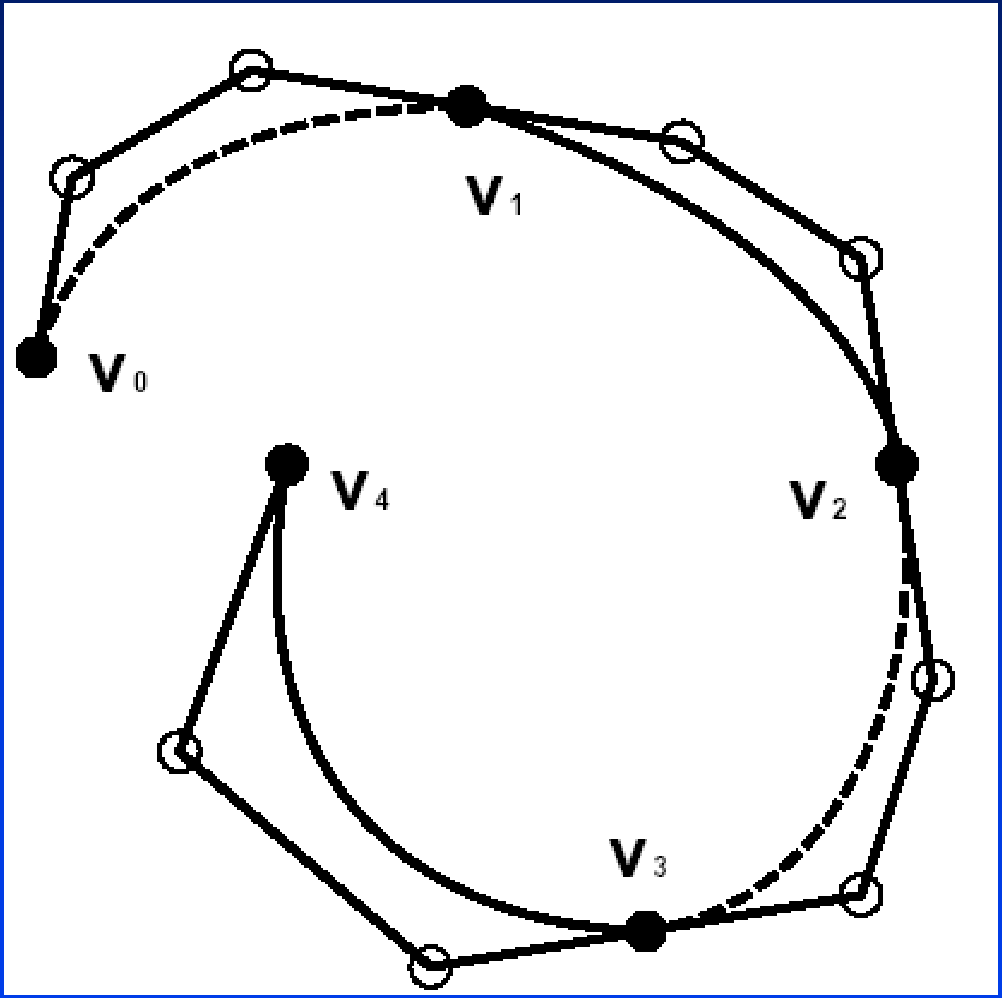
Natural C2 Cubic Splines

- A set of piecewise cubic polynomials

$$\mathbf{c}_i(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$

- C2 continuity at each vertex

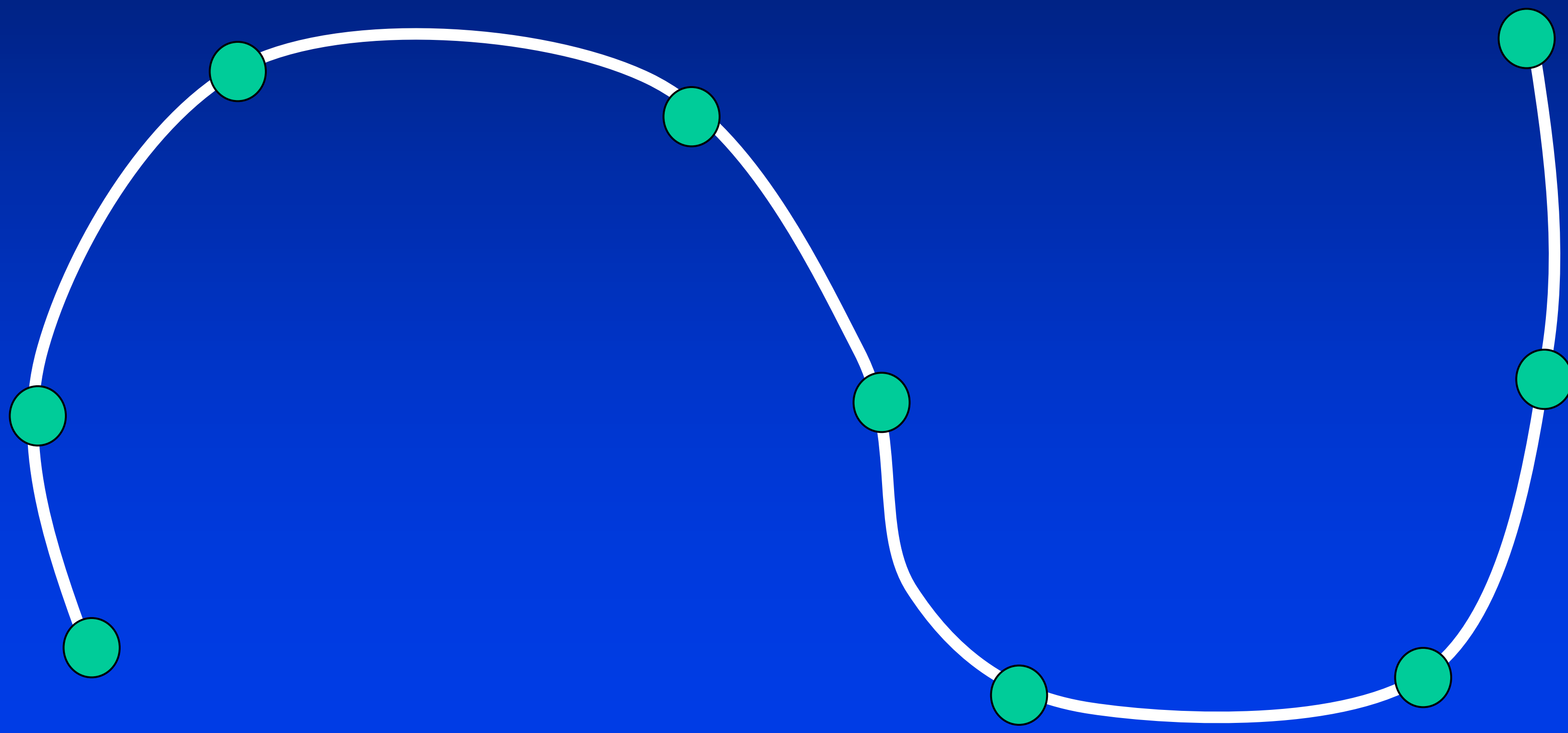
C^2 Interpolating Splines



C^2 Interpolating Splines

- Interpolate all control points
- Equivalent to a thin strip of metal in a physical sense.
- Forced to pass through a set of desired points.
- Advantages:
  - interpolation,
  - C^2
- Disadvantages:
  - No local control (if one point is changes, the entire curve will move)
- How to overcome the drawbacks: B-splines.

Natural C2 Cubic Splines



Natural Splines

- Interpolate all control points
- Equivalent to a thin strip of metal in a physical sense
- Forced to pass through a set of desired points
- No local control (global control)
- N+1 control points
- N pieces
- 2(n-1) conditions
- We need two additional conditions

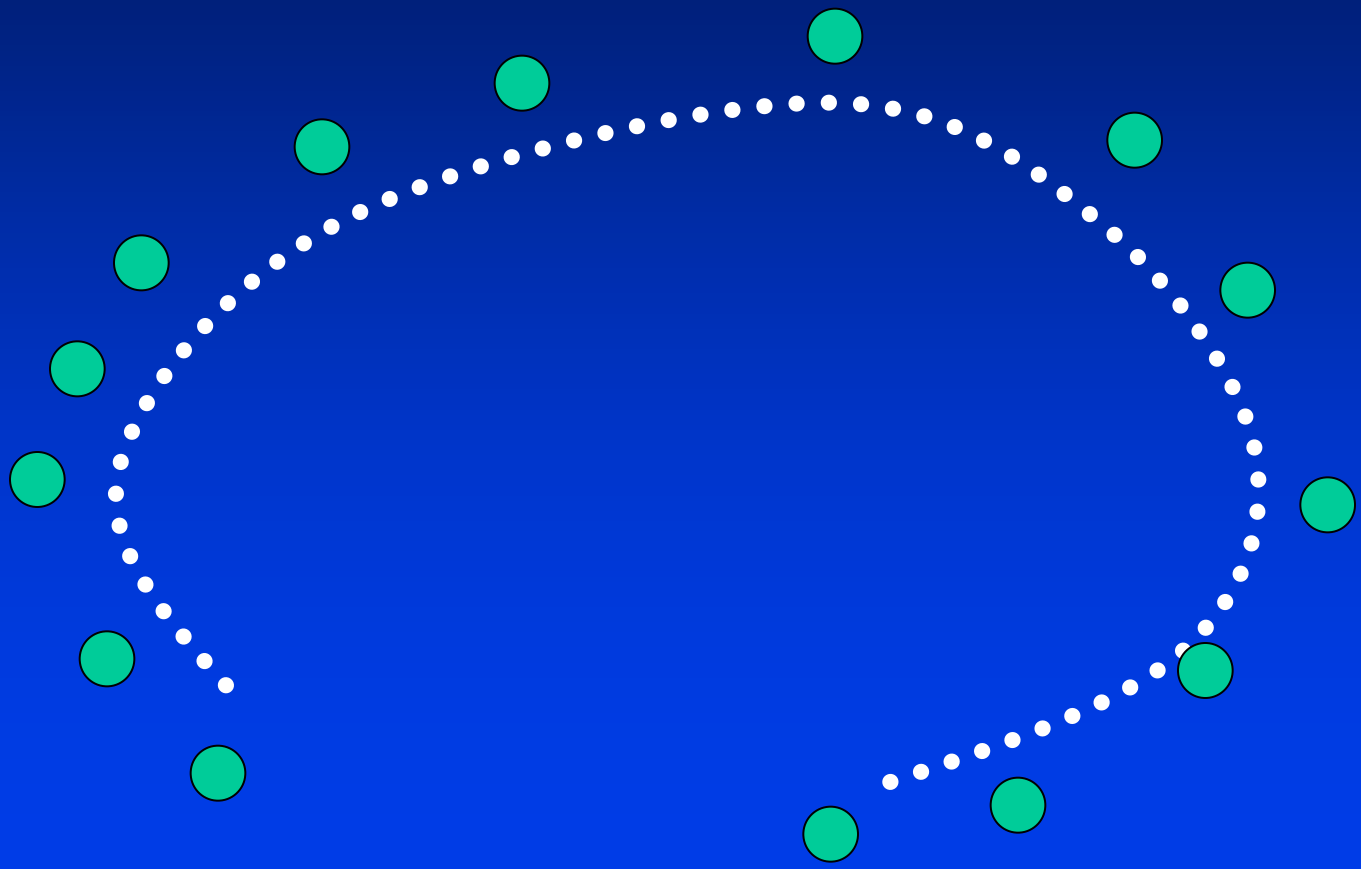
Natural Splines

- Interactive design system
  - Specify derivatives at two end-points
  - Specify the two internal control points that define the first curve span
  - Natural end conditions: second-order derivatives at two end points are defined to be zero
- Advantages: interpolation, C2
- Disadvantages: no local control (if one point is changed, the entire curve will move)
- How to overcome this drawback: B-Splines

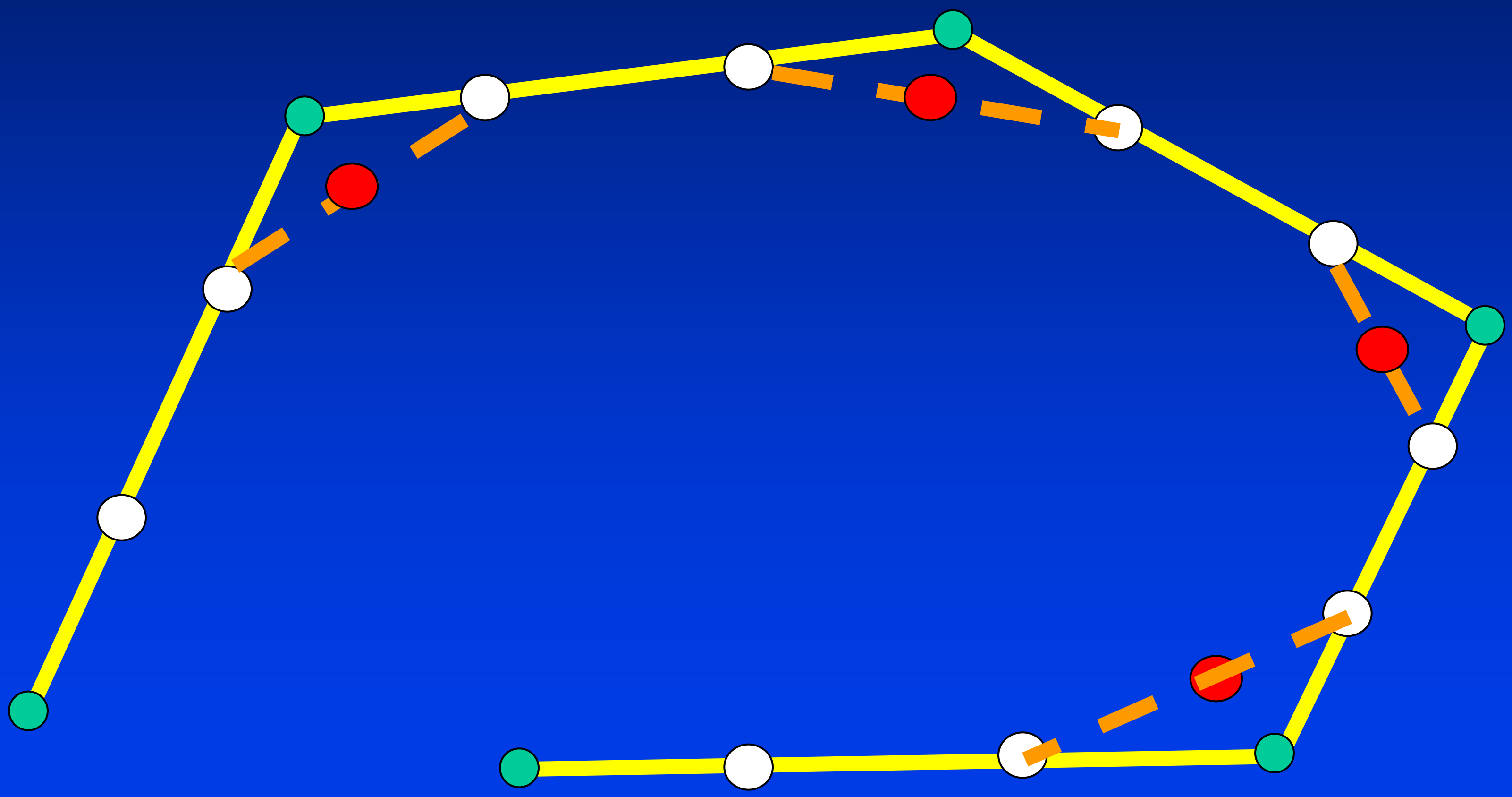
B-Splines Motivation

- The goal is local control!!!
- B-splines provide local control
- Do not interpolate control points
- C2 continuity
- Alternatively
  - Catmull-Rom Splines
  - Keep interpolations
  - Give up C2 continuity (only C1 is achieved)
  - Will be discussed later!!!

C2 Approximating Splines

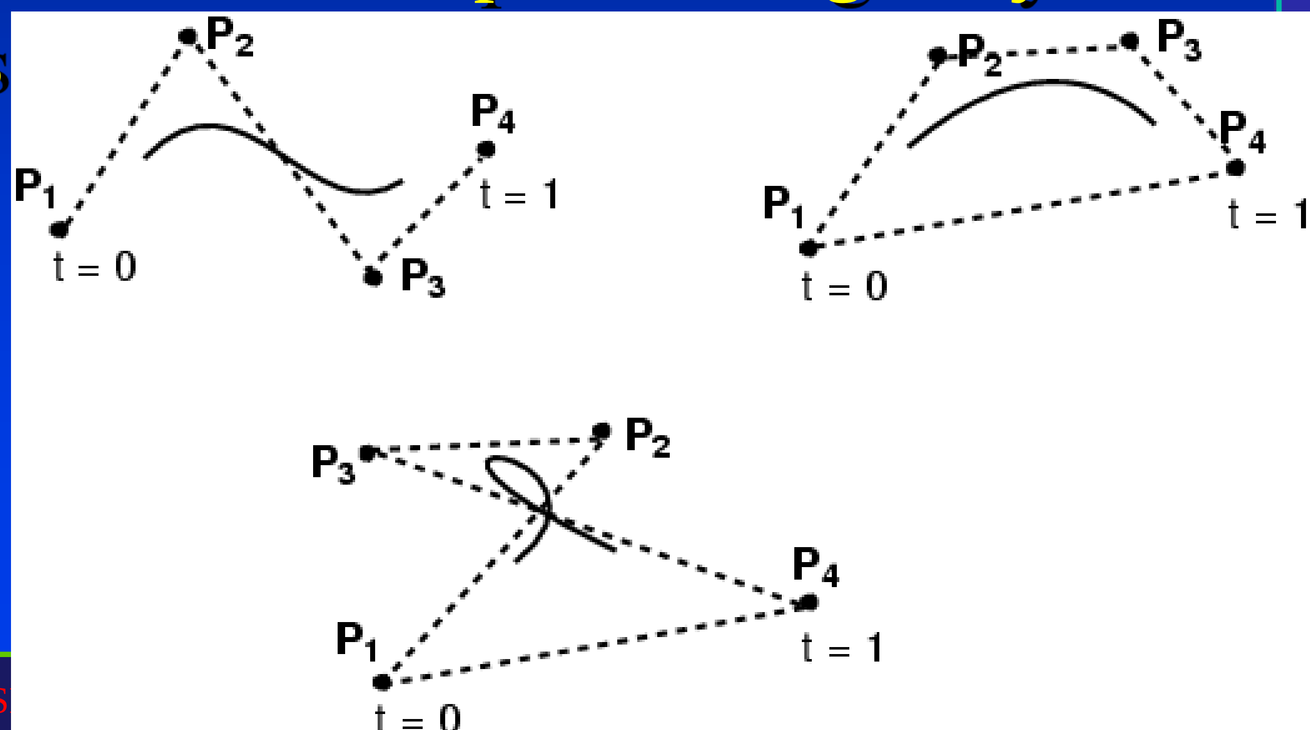


From B-Splines to Bezier

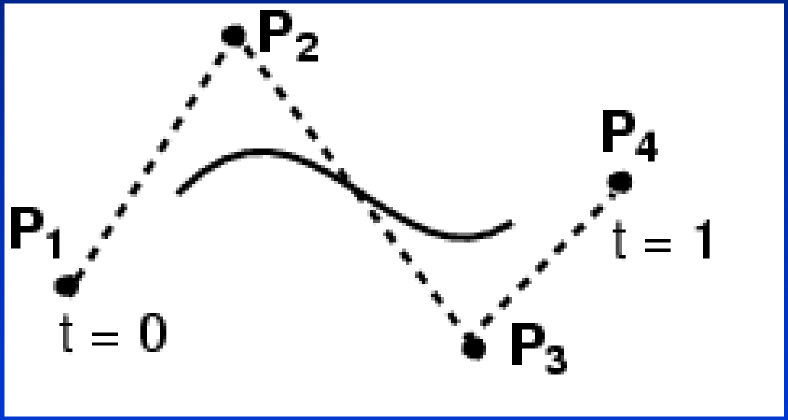


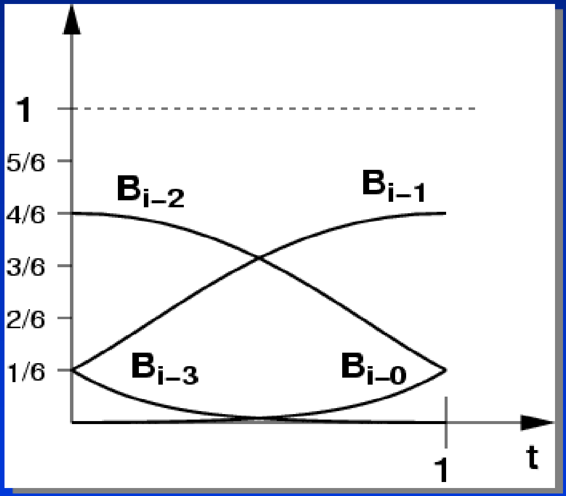
Cubic B-spline Curves (One Curve Span)

- $\geq$ 4 control points
- Locally cubic
- Curve is not constrained to pass through any control points



Cubic B-spline Curve (One Curve Span)

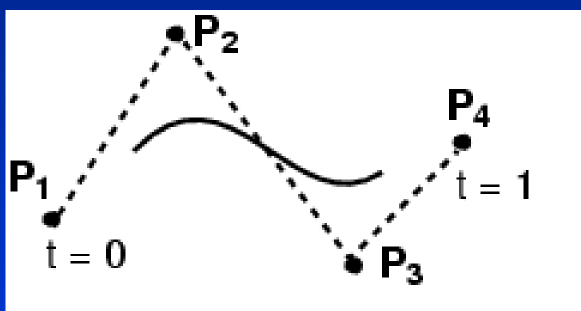


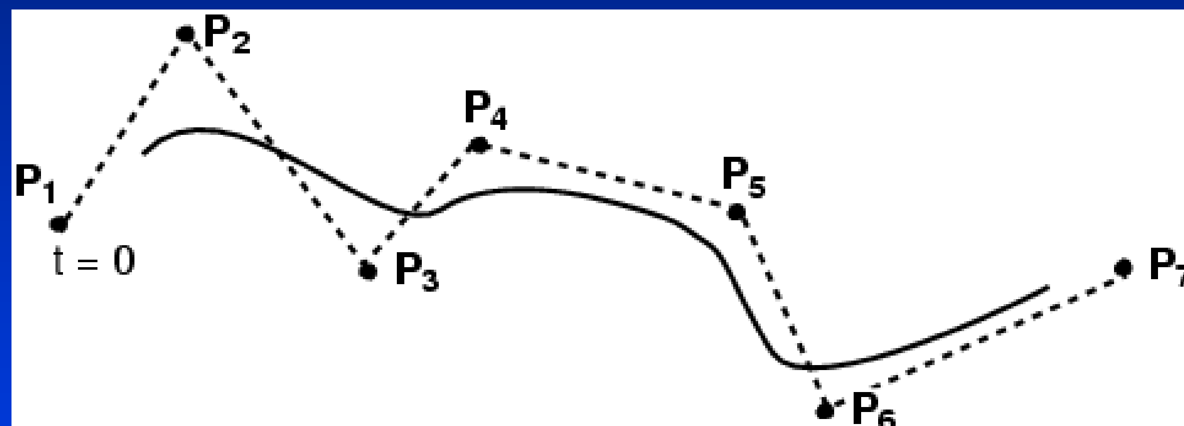


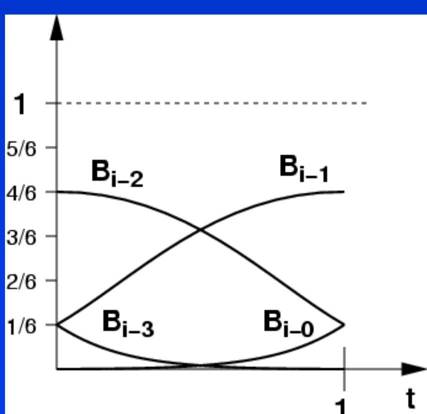
$$Q(t) = \frac{(1-t)^3}{6} P_{i-3} + \frac{3t^3 - 6t^2 + 4}{6} P_{i-2} + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_{i-1} + \frac{t^3}{6} P_i$$

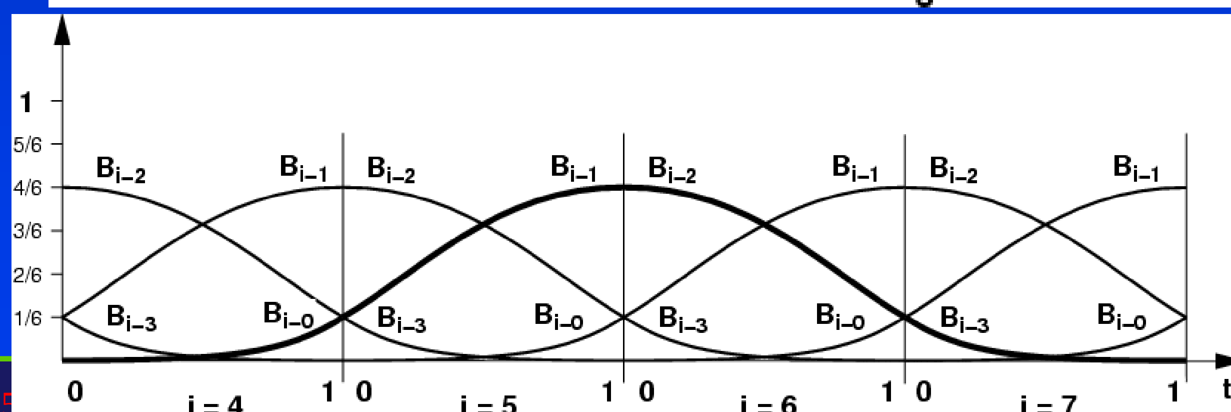
Cubic B-Spline Curve (Many Curve Spans)

- can be chained together with a higher-order continuity
- better control locally (windowing)





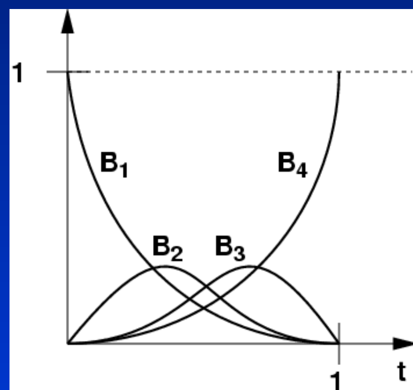


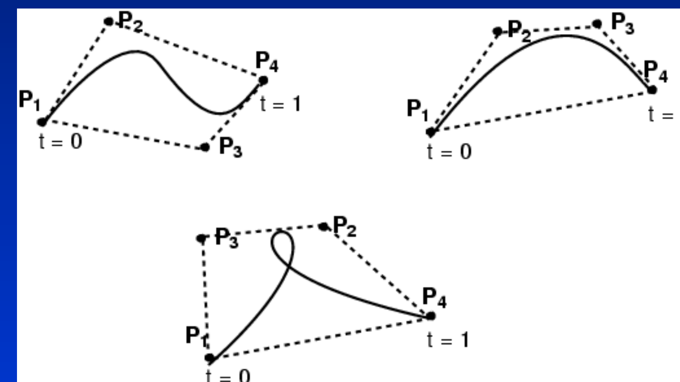


Bézier Curve vs. B-Spline Curve

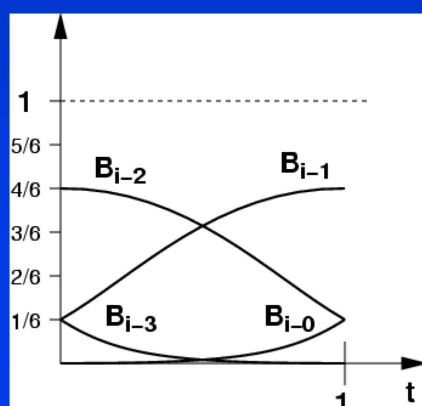
- Bezier curve is NOT the same as B-Spline curve!

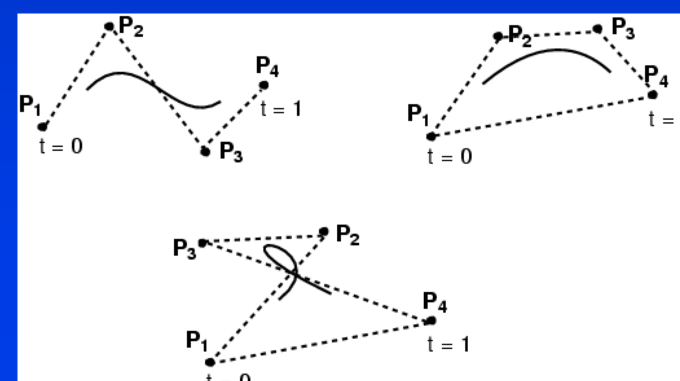
Bézier





B-Spline





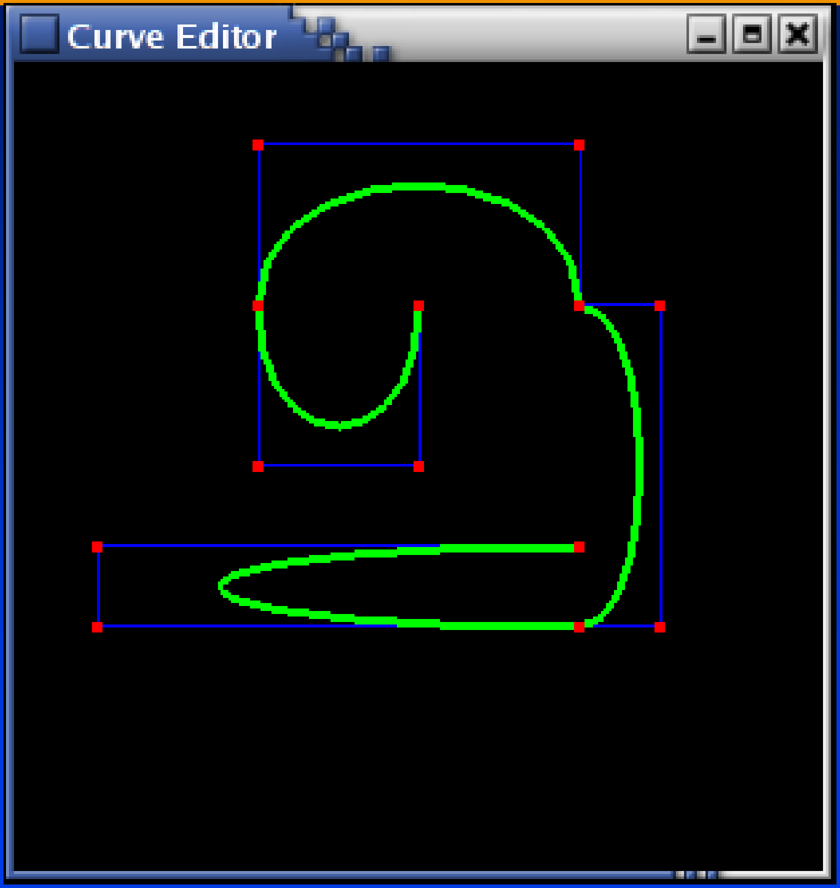
Bezier is not the same as B-spline

- But we can convert between the curves using the basis functions:

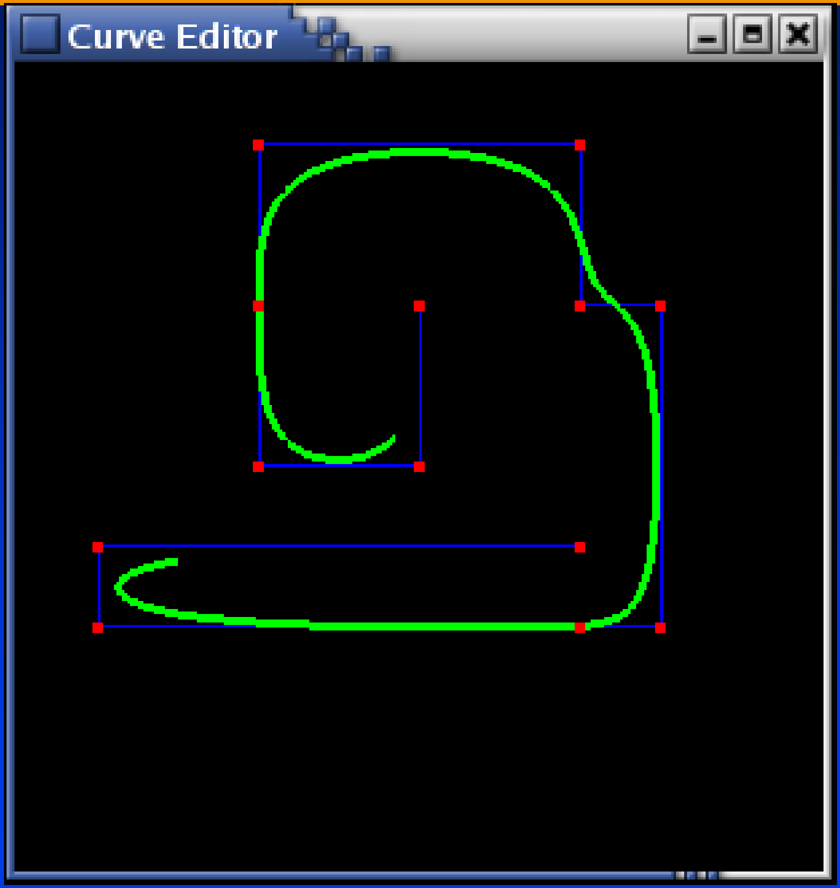
$$B_{Bezier} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

Bézier Curve vs. B-Spline Curve



Bézier

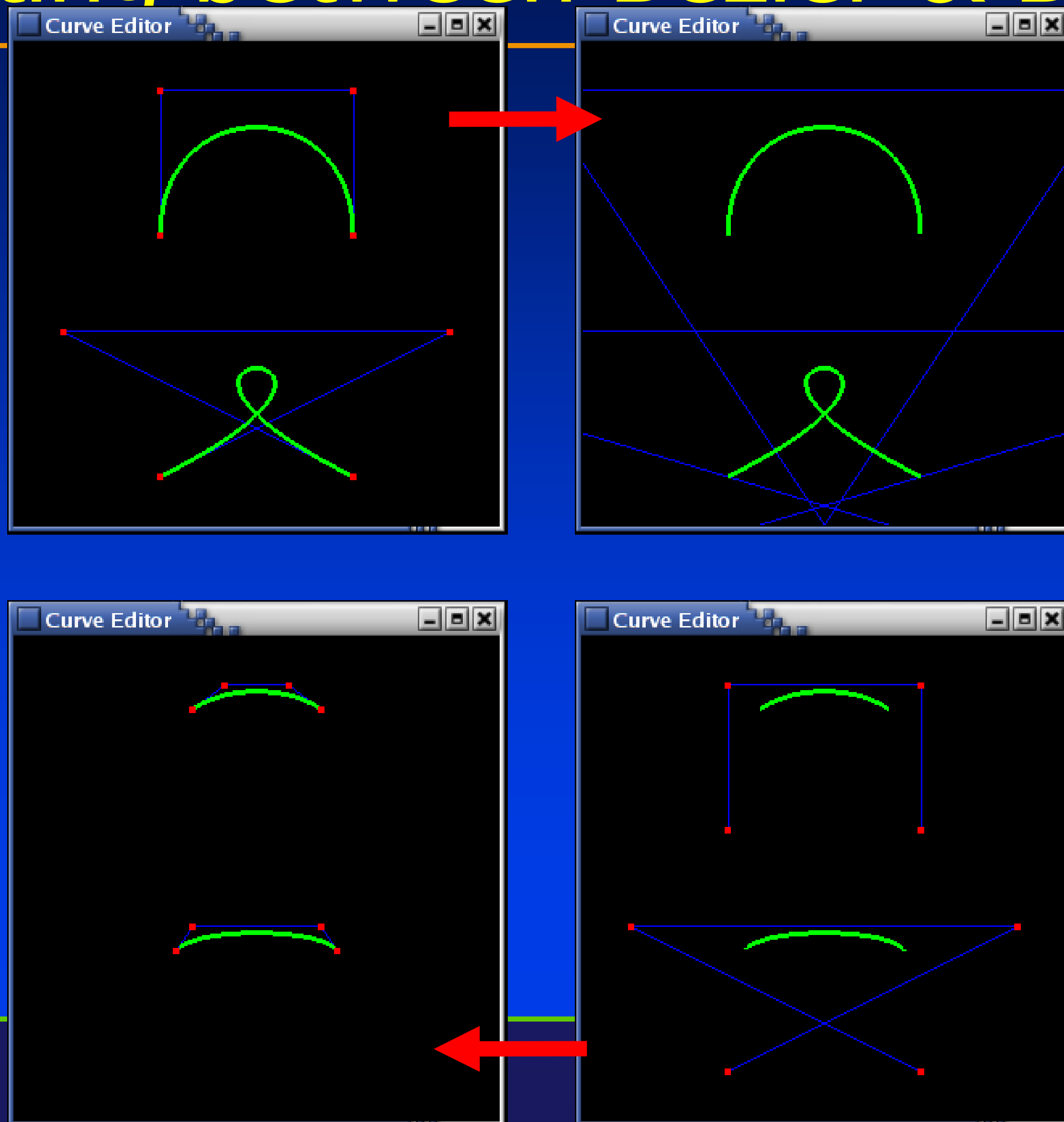


B-Spline

Converting between Bézier & B-Spline

original control points as Bézier

new BSpline control points to match Bézier



new Bézier control points to match BSpline

original control points as BSpline

STONY BROOK
STATE UNIVERSITY OF NEW YORK

Uniform B-Splines

- B-spline control points: $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$
- Piecewise Bezier curves with C2 continuity at joints
- Bezier control points:

$$\mathbf{v}_0 = \mathbf{p}_0$$
$$\mathbf{v}_1 = \frac{2\mathbf{p}_1 + \mathbf{p}_2}{3}$$
$$\mathbf{v}_2 = \frac{\mathbf{p}_1 + 2\mathbf{p}_2}{3}$$
$$\mathbf{v}_0 = \frac{1}{2}\left(\frac{\mathbf{p}_0 + 2\mathbf{p}_1}{3} + \frac{2\mathbf{p}_1 + \mathbf{p}_2}{3}\right) = \frac{1}{6}(\mathbf{p}_0 + 4\mathbf{p}_1 + \mathbf{p}_2)$$
$$\mathbf{v}_3 = \frac{1}{6}(\mathbf{p}_1 + 4\mathbf{p}_2 + \mathbf{p}_3)$$

Uniform B-Splines

- In general, I-th segment of B-splines is determined by four consecutive B-spline control points

$$\mathbf{v}_1 = \frac{2\mathbf{p}_{i+1} + \mathbf{p}_{i+2}}{3}$$

$$\mathbf{v}_2 = \frac{\mathbf{p}_{i+1} + 2\mathbf{p}_{i+2}}{3}$$

$$\mathbf{v}_0 = \frac{1}{6}(\mathbf{p}_i + 4\mathbf{p}_{i+1} + \mathbf{p}_{i+2})$$

$$\mathbf{v}_3 = \frac{1}{6}(\mathbf{p}_{i+1} + 4\mathbf{p}_{i+2} + \mathbf{p}_{i+3})$$

Uniform B-Splines

- In matrix form

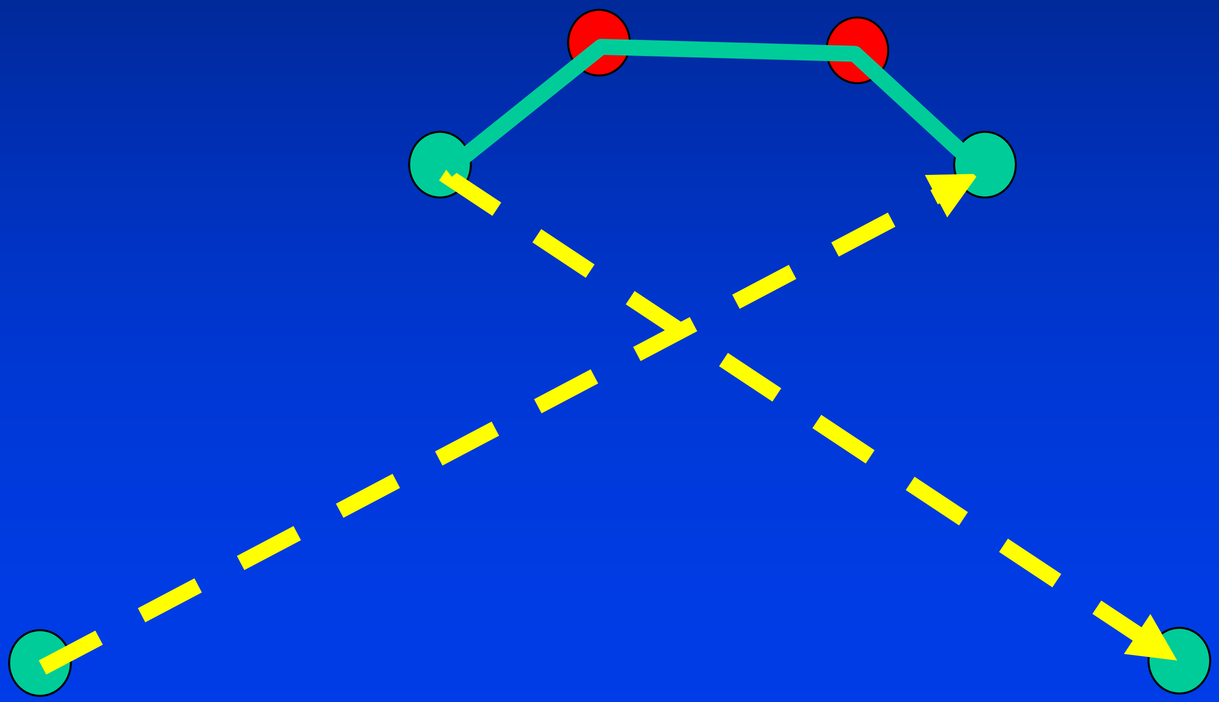
$$\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \\ \mathbf{p}_{i+3} \end{bmatrix}$$

- Question: how many Bezier segments???

B-Spline Properties

- C2 continuity, Approximation, Local control, convex hull
- Each segment is determined by four control points
- Questions: what happens if we put more than one control points in the same location???
  - Double vertices, triple vertices, collinear vertices
- End conditions
  - Double endpoints: curve will be tangent to line between first distinct points
  - Triple endpoint: curve interpolate endpoint, start with a line segment
- B-spline display: transform it to Bezier curves

Catmull-Rom Splines



Catmull-Rom Splines

- Keep interpolation
- Give up C2 continuity
- Control tangents locally
- Idea: Bezier curve between successive points
- How to determine two internal vertices

$$\mathbf{c}(0) = \mathbf{p}_i = \mathbf{v}_0, \mathbf{c}(1) = \mathbf{p}_{i+1} = \mathbf{v}_3$$

$$\mathbf{c}'(0) = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2} = 3(\mathbf{v}_1 - \mathbf{v}_0)$$

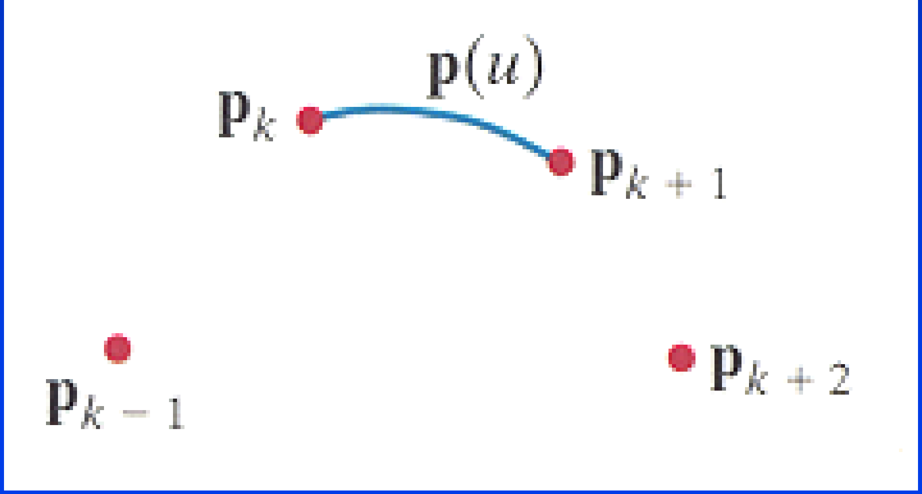
$$\mathbf{c}'(1) = \frac{\mathbf{p}_{i+2} - \mathbf{p}_i}{2} = 3(\mathbf{v}_3 - \mathbf{v}_2)$$

$$\mathbf{v}_1 = \frac{\mathbf{p}_{i+1} + 6\mathbf{p}_i - \mathbf{p}_{i-1}}{6}$$

$$\mathbf{v}_2 = \frac{-\mathbf{p}_{i+2} + 6\mathbf{p}_{i+1} + \mathbf{p}_i}{6}$$

Catmull-Rom Spline

- Different from Bezier curves in that we can have arbitrary number of control points, but only 4 of them influence each section of curve
  - And it is **interpolating** (goes through points) instead of **approximating** (goes "near" points)
- Four points define curve between 2nd and 3rd



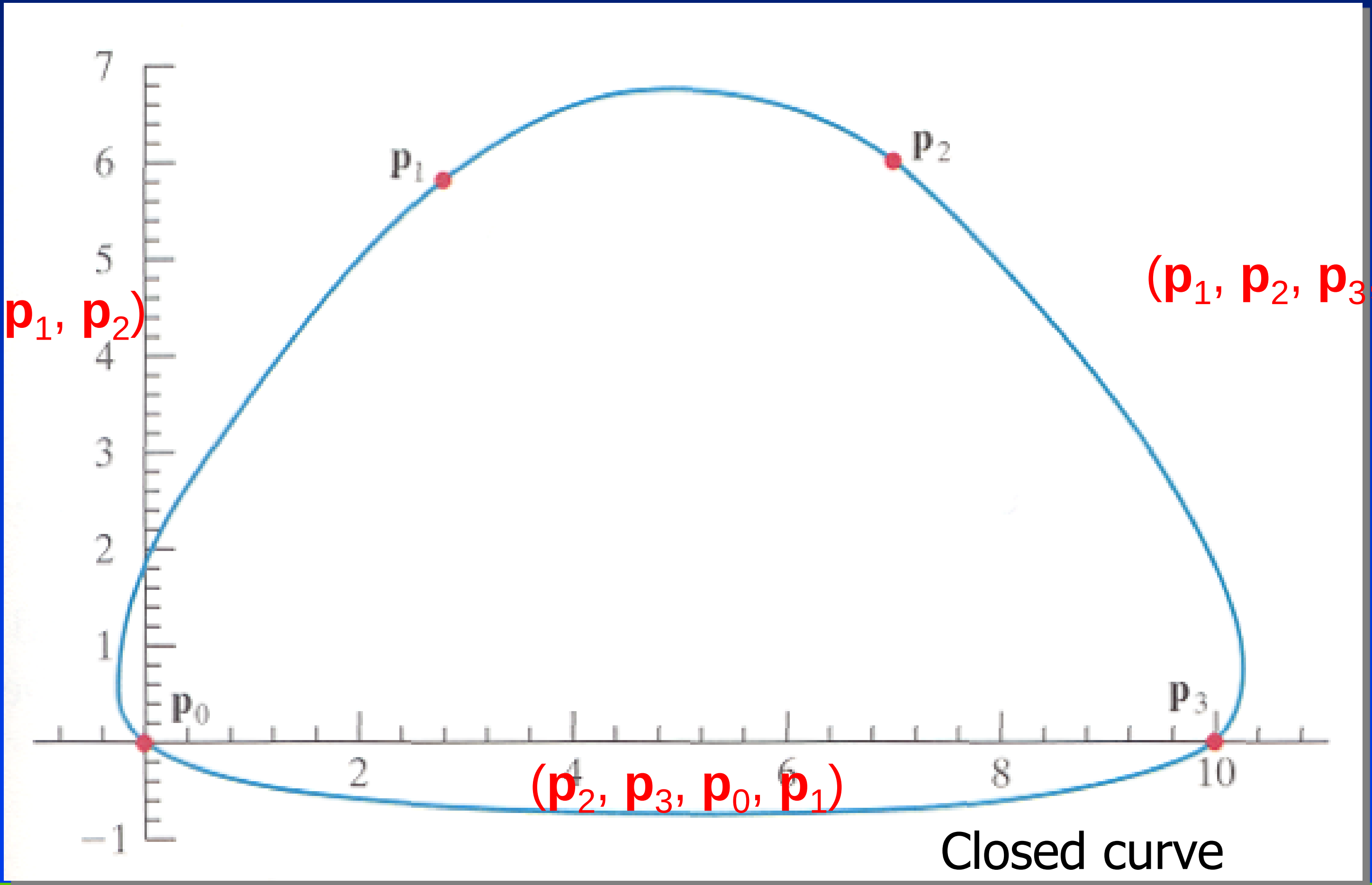
Catmull-Rom Spline: Example

$(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$

$(\mathbf{p}_3, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$

$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_0)$

$(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_0, \mathbf{p}_1)$



from Hearn & Baker

Catmull-Rom Splines

- In matrix form

$$\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & 6 & 0 & 0 \\ -1 & 6 & 1 & 0 \\ 0 & 1 & 6 & -1 \\ 0 & 0 & 6 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$

- Problem: boundary conditions
- Properties: C1, interpolation, local control, non-convex-hull

Cardinal Splines

- Four vertices define end-points and their associated tangents

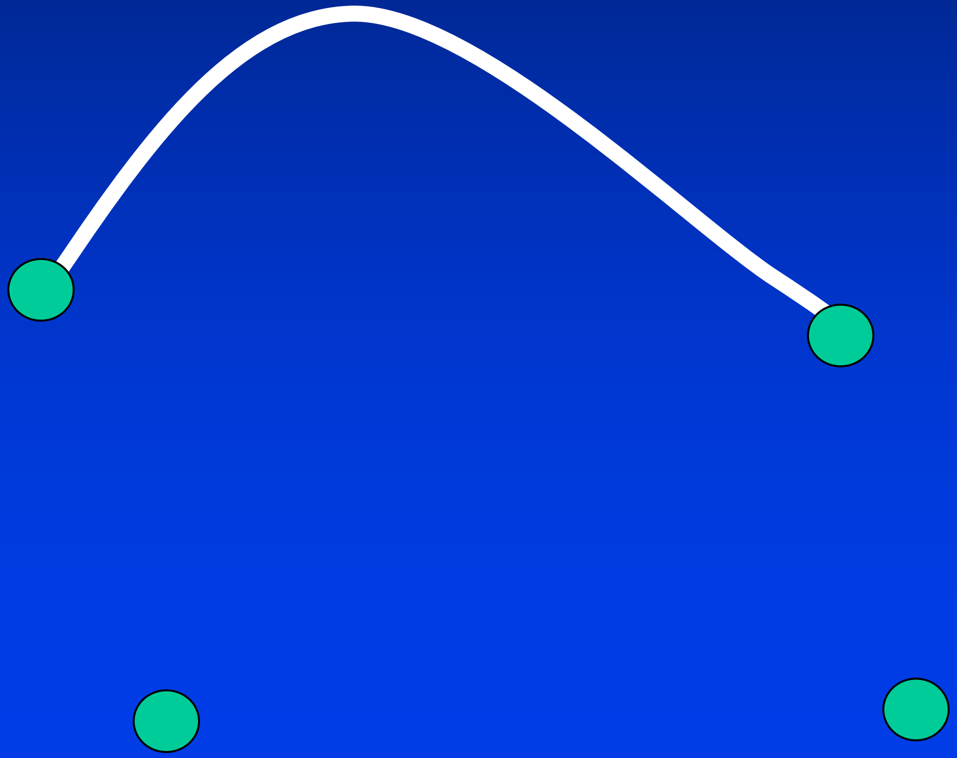
$$\mathbf{c}(0)=\mathbf{v}_1,\mathbf{c}(1)=\mathbf{v}_2$$

$$\mathbf{c}^{(1)}(0)=\frac{1}{2}(1-\alpha)(\mathbf{v}_2-\mathbf{v}_0)$$

$$\mathbf{c}^{(1)}(1)=\frac{1}{2}(1-\alpha)(\mathbf{v}_3-\mathbf{v}_1)$$

- Special case: Catmull-Rom splines when $\alpha=0$
- More general case: Kochanek-Bartels splines
  – Tension, bias, continuity parameters

Cardinal Splines



Kochanek-Bartels Splines

- Four vertices to define four conditions

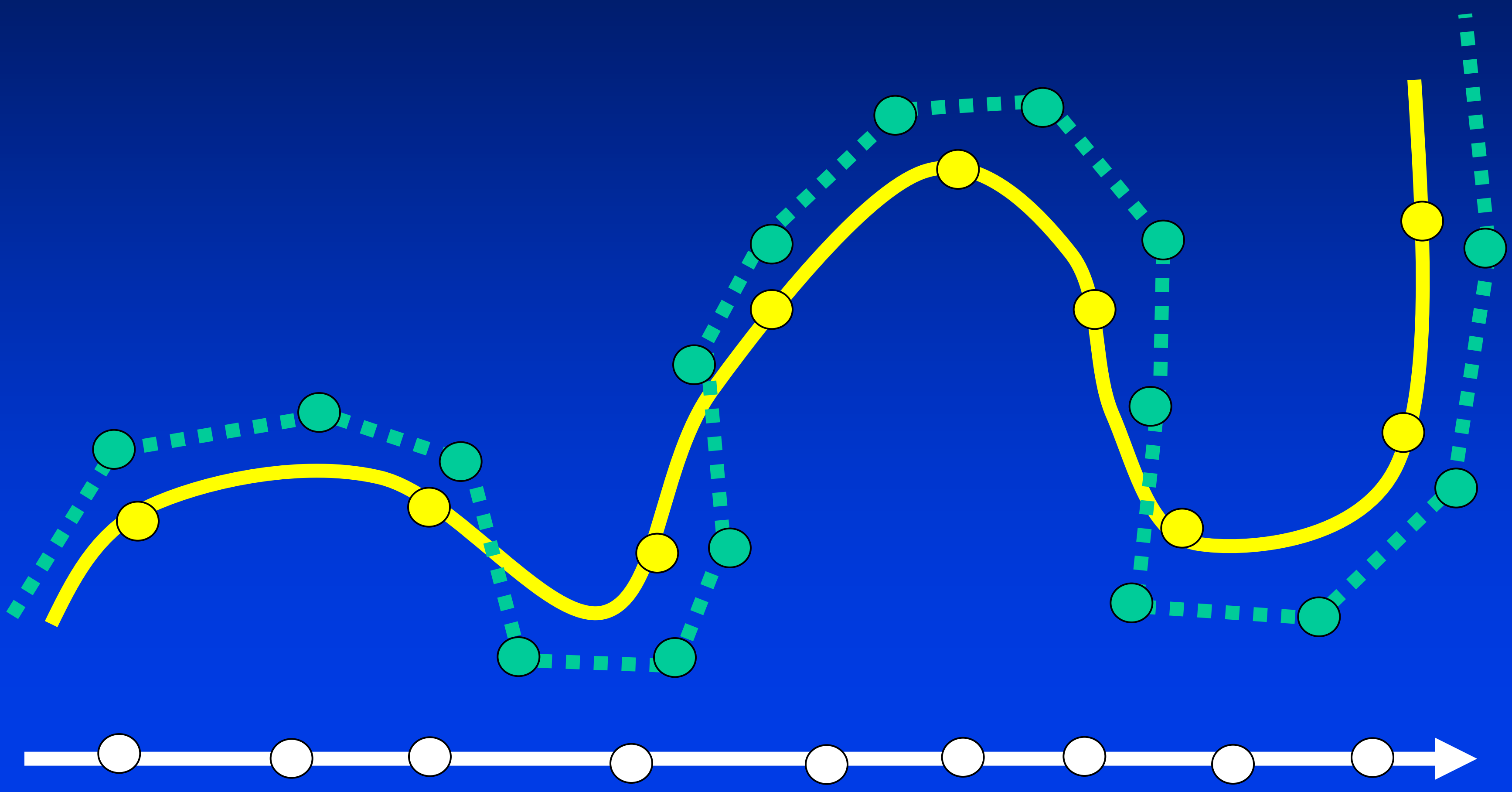
$$\mathbf{c}(0) = \mathbf{v}_1, \mathbf{c}(1) = \mathbf{v}_2$$

$$\mathbf{c}^{(1)}(0) = \frac{1}{2}(1-\alpha)((1+\beta)(1-\gamma)(\mathbf{v}_1 - \mathbf{v}_0) + (1-\beta)(1+\gamma)(\mathbf{v}_2 - \mathbf{v}_1))$$

$$\mathbf{c}^{(1)}(1) = \frac{1}{2}(1-\alpha)((1+\beta)(1+\gamma)(\mathbf{v}_2 - \mathbf{v}_1) + (1-\beta)(1-\gamma)(\mathbf{v}_3 - \mathbf{v}_2))$$

  – Tension parameter: α
  – Bias parameter: β
  – Continuity parameter: γ

Piecewise B-Splines

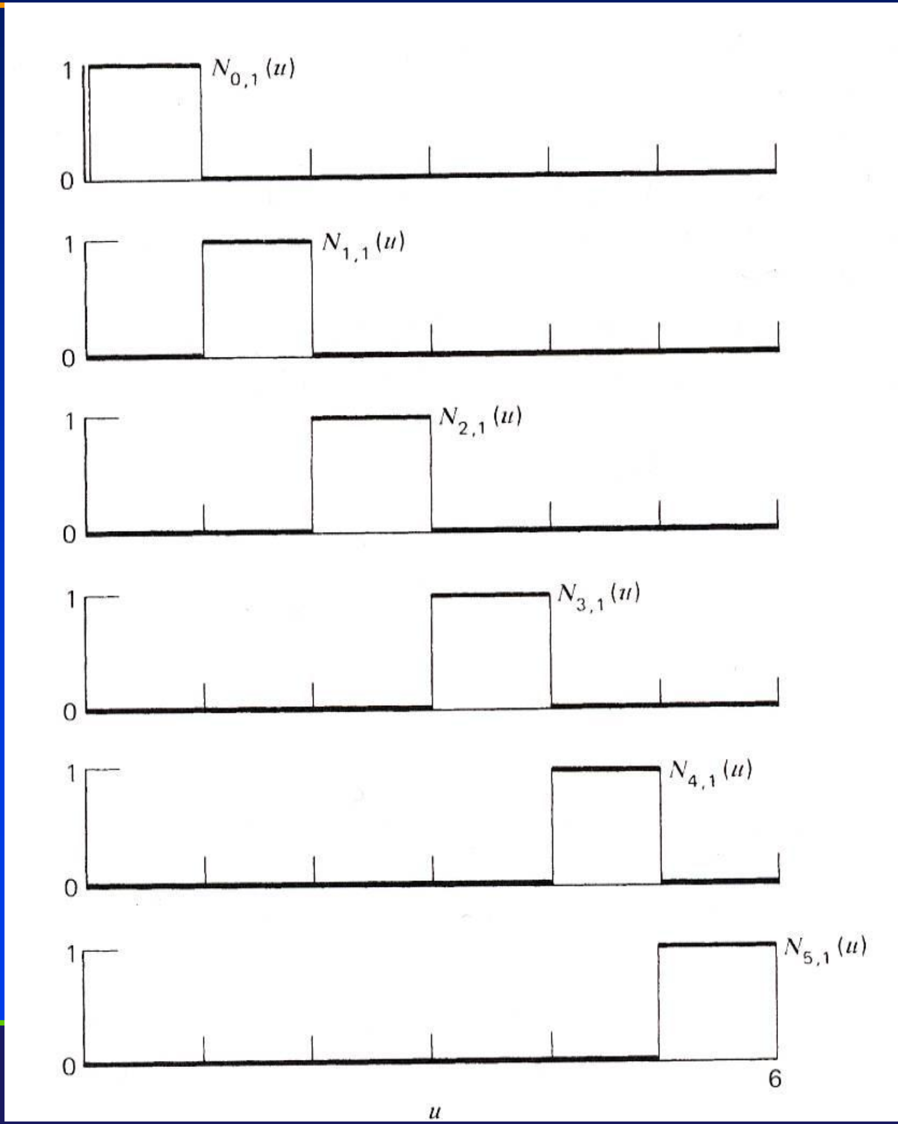


B-Spline Basis Functions

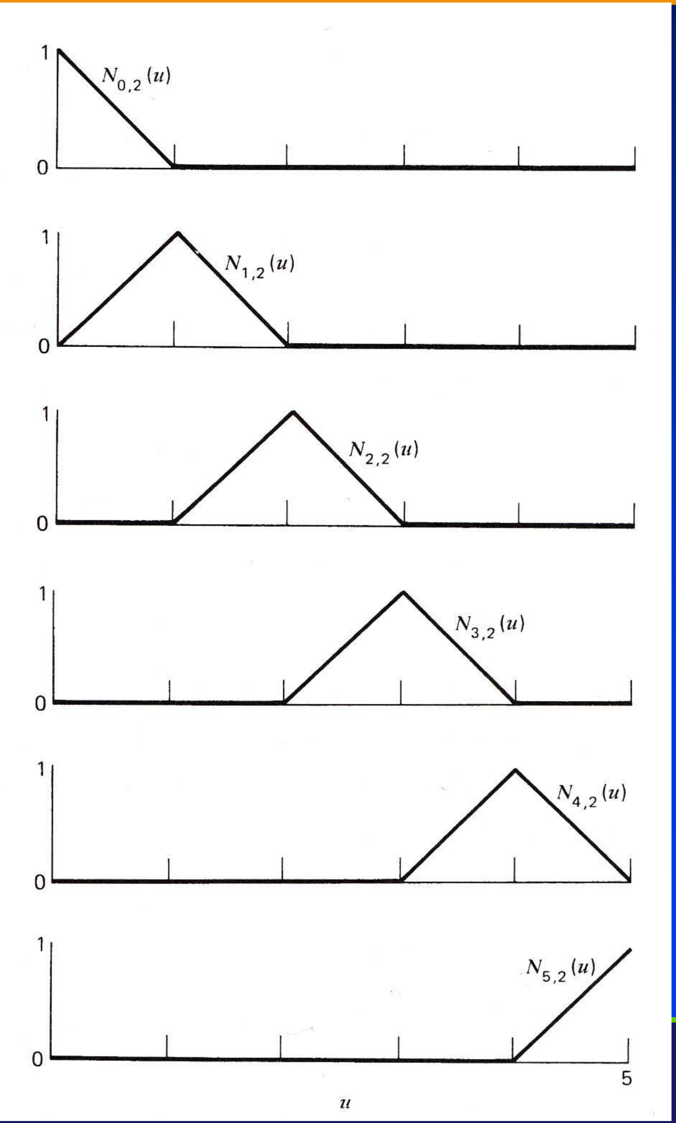
$$B_{i,1}(u) = \begin{cases} 1 & u_i <= u < u_{i+1} \\ 0 & otherwise \end{cases}$$

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

B-Spline Basis Functions



B-Spline Basis Functions



Basis Functions

- Linear examples

$$B_{0,2}(u) = \begin{cases} u & u \in [0,1] \\ 2-u & u \in [1,2] \end{cases}$$

$$B_{1,2}(u) = \begin{cases} u-1 & u \in [1,2] \\ 3-u & u \in [2,3] \end{cases}$$

$$B_{2,2}(u) = \begin{cases} u-2 & u \in [2,3] \\ 4-u & u \in [3,4] \end{cases}$$

- How does it look like???

Basis Functions

- Quadratic cases (knot vector is [0,1,2,3,4,5,6])

$$B_{0,3}(u) = \begin{cases} \frac{1}{2}u^2, & 0 <= u < 1 \\ \frac{1}{2}u(2-u) + \frac{1}{2}(u-1)(3-u), & 1 <= u < 2 \\ \frac{1}{2}(3-u)^2, & 2 <= u < 3 \end{cases}$$

$$B_{1,3}(u) = \begin{cases} \frac{1}{2}(u-1)^2, & 1 <= u < 2 \\ \frac{1}{2}(u-1)(3-u) + \frac{1}{2}(u-2)(4-u), & 2 <= u < 3 \\ \frac{1}{2}(4-u)^2, & 3 <= u < 4 \end{cases}$$

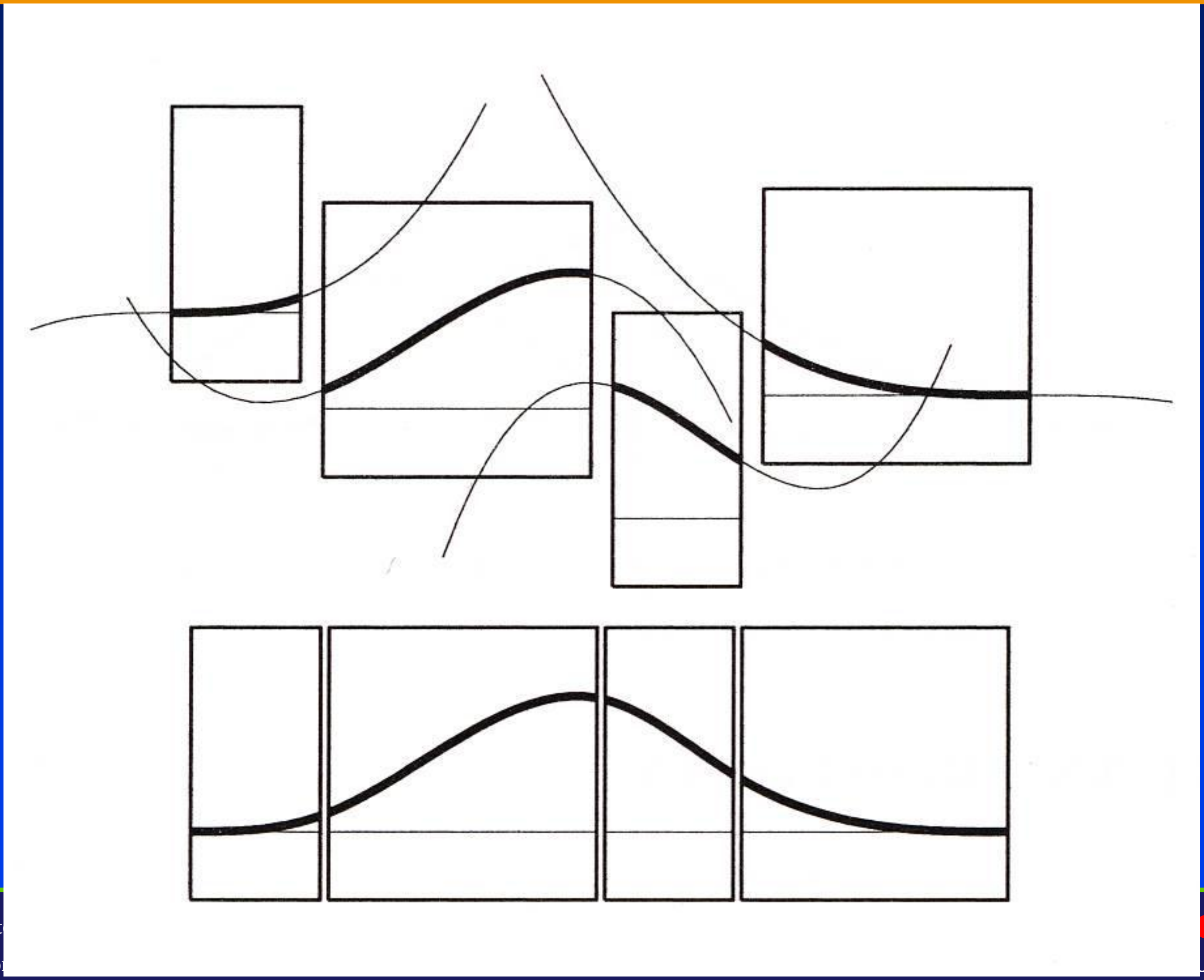
$$B_{2,3}(u) = \ldots\ldots$$

$$B_{3,3}(u) = \ldots\ldots$$

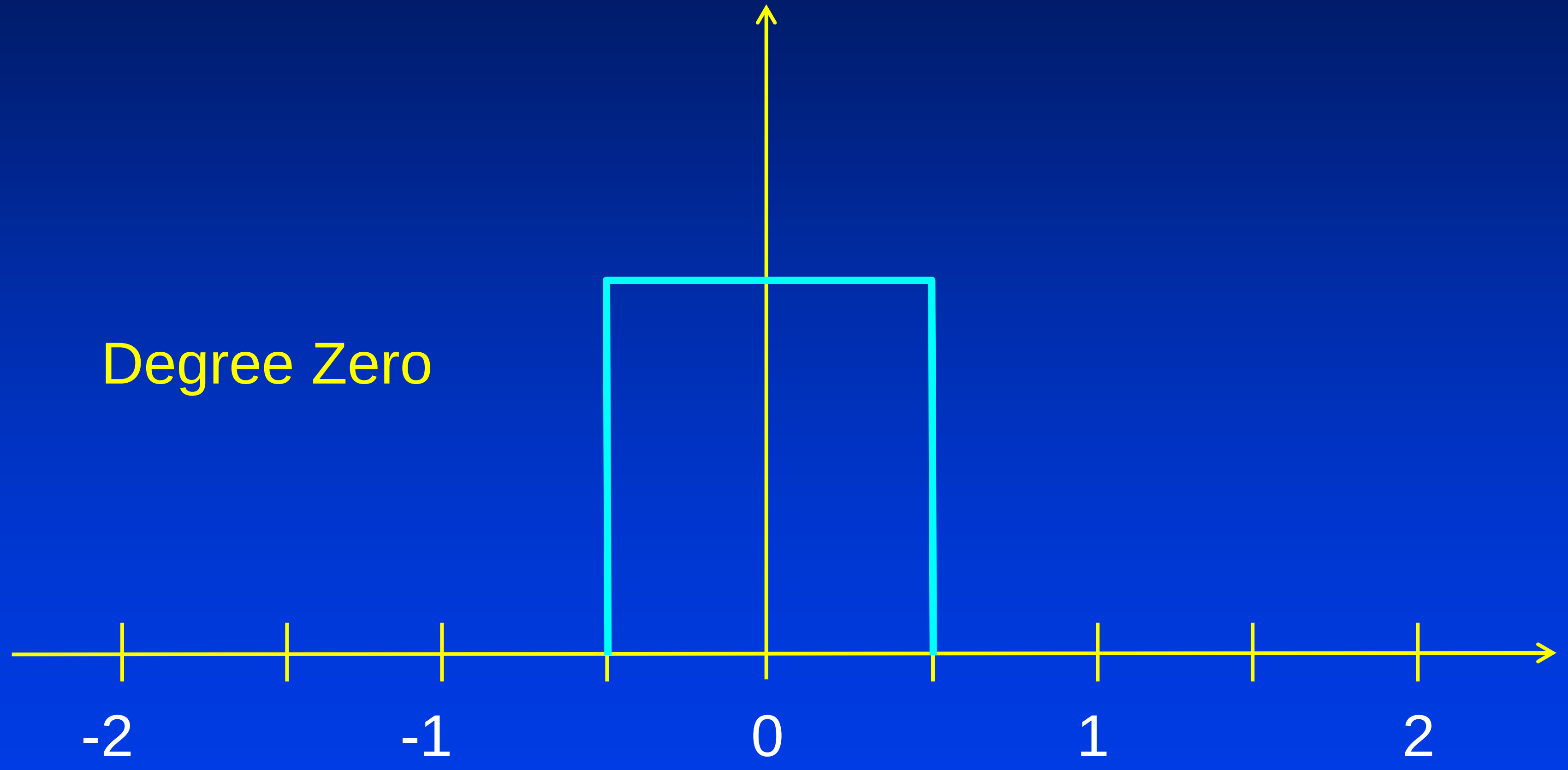
- Cubic example

B-Spline Basis Function Image

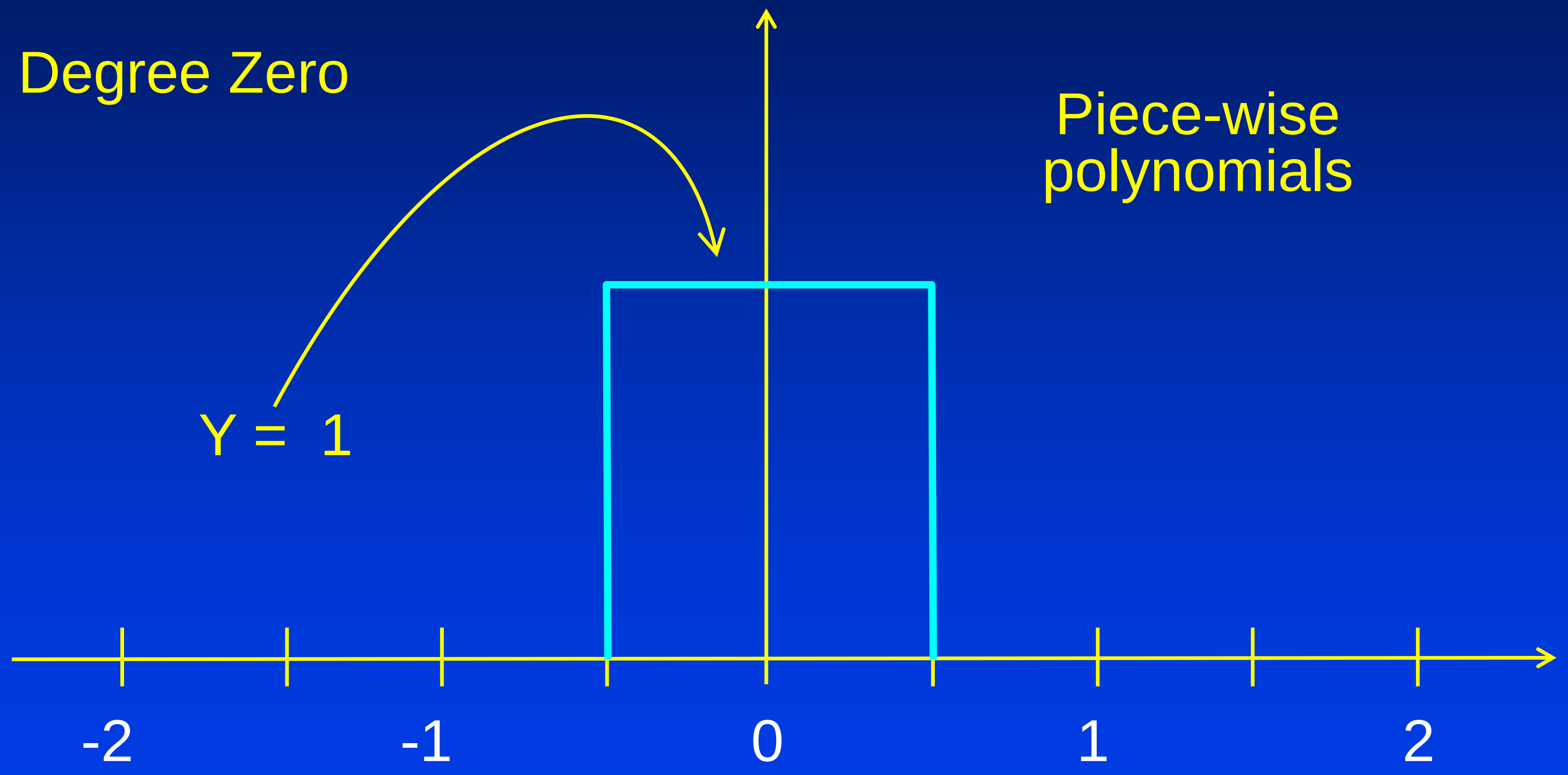
B-Spline Basis Functions



B-Spline Basis Function

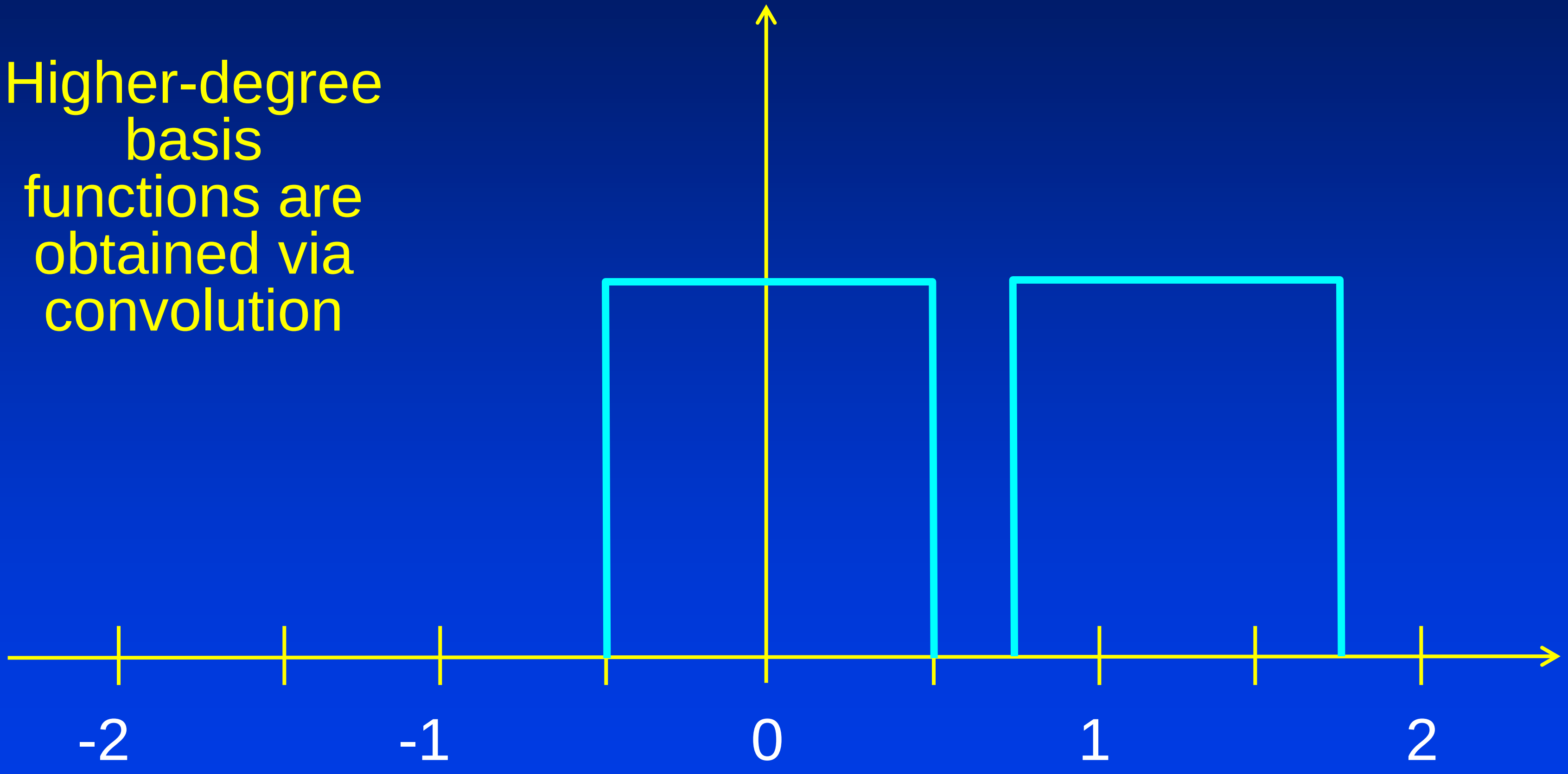


B-Spline Basis Function



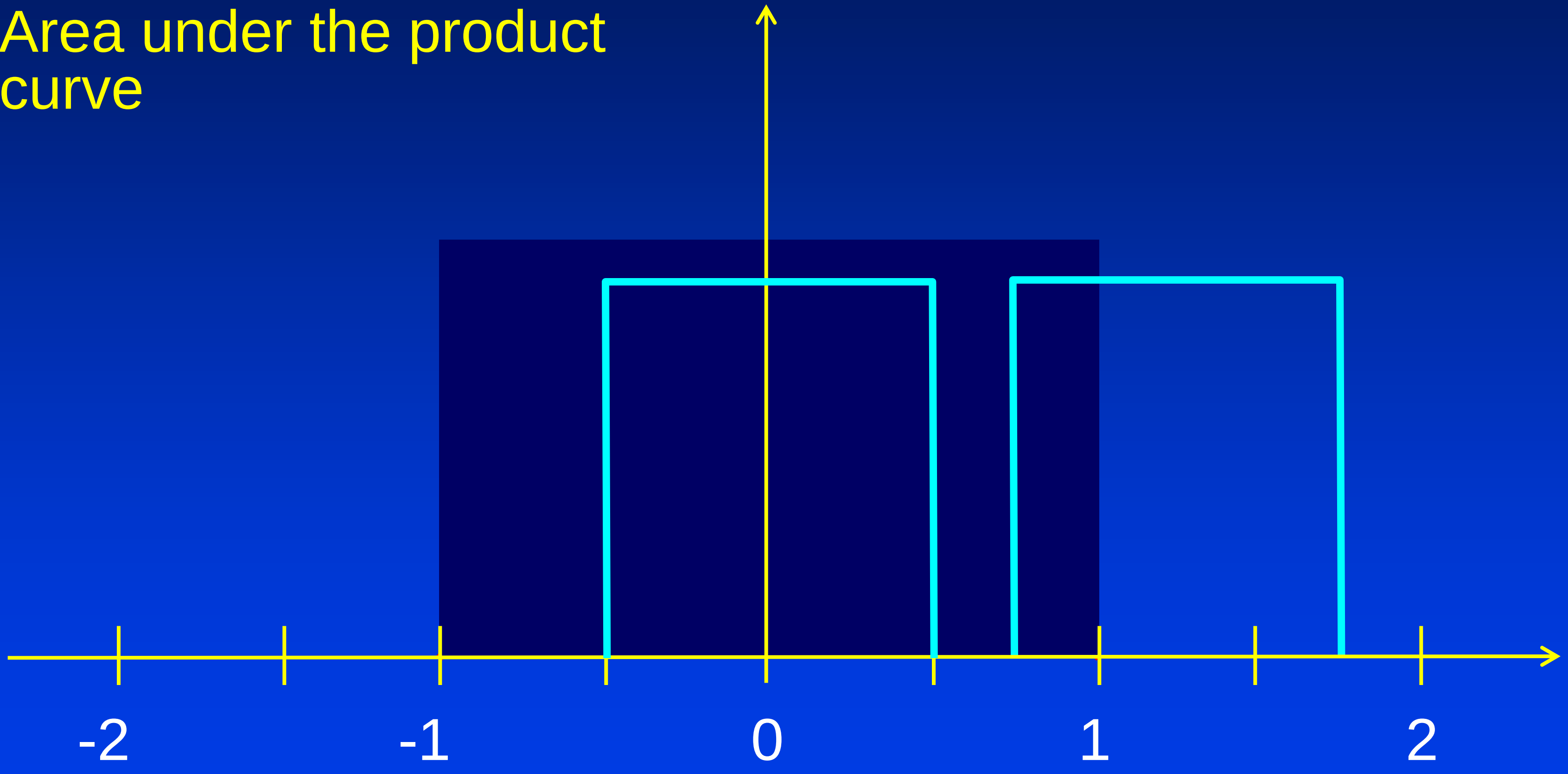
B-Spline Basis Function

Higher-degree basis functions are obtained via convolution

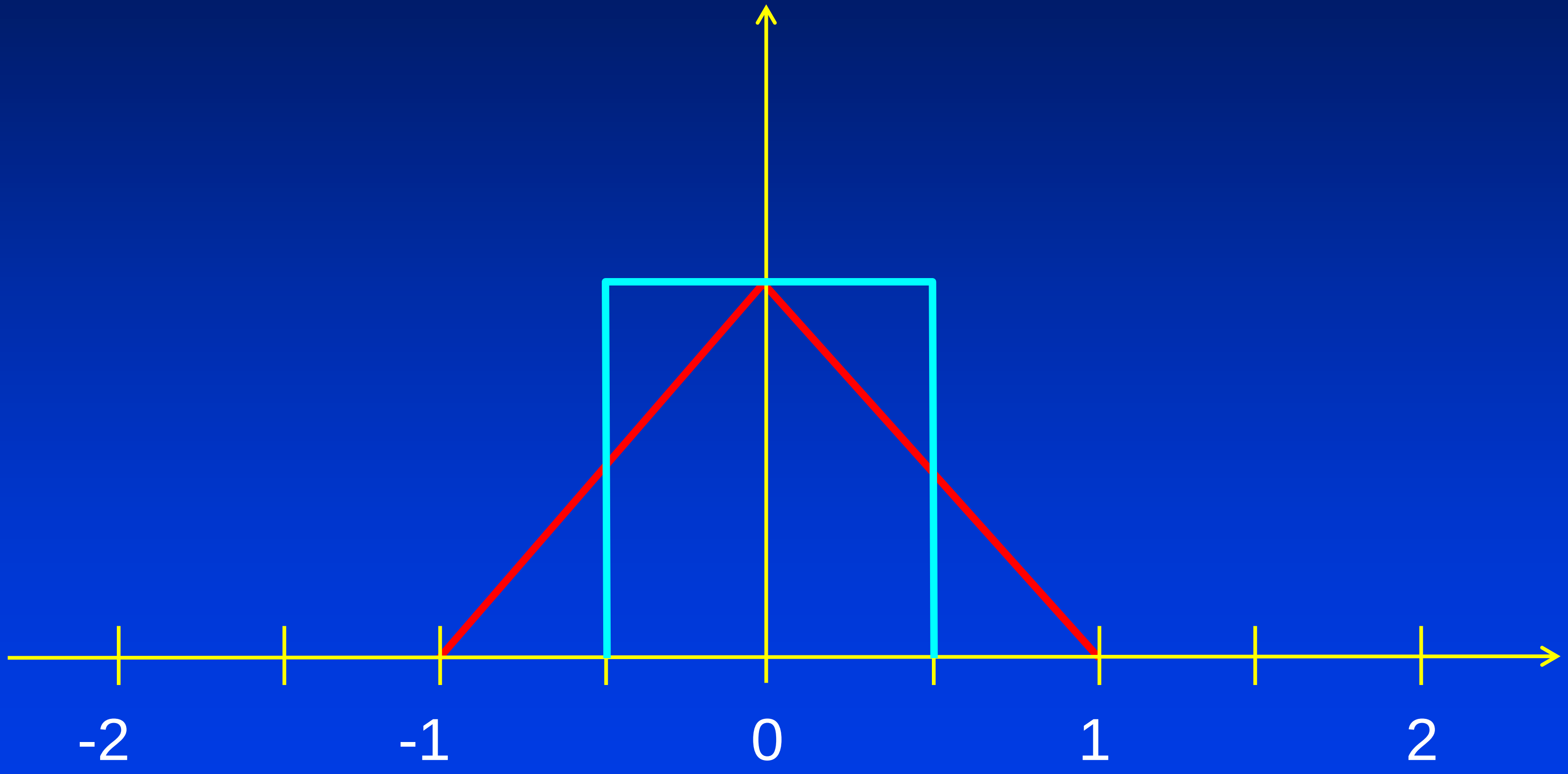


B-Spline Basis Function

Area under the product curve

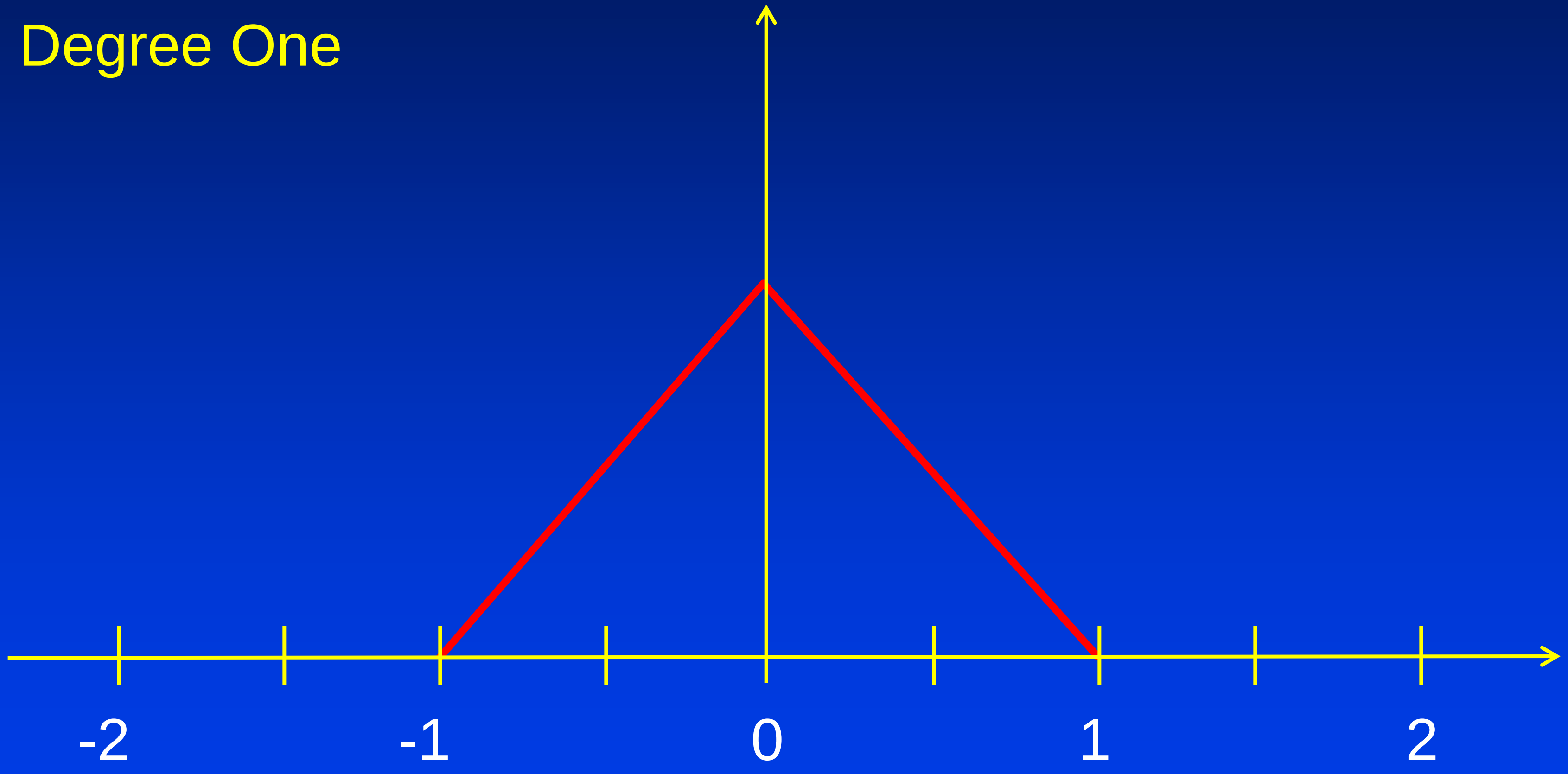


B-Spline Basis Function

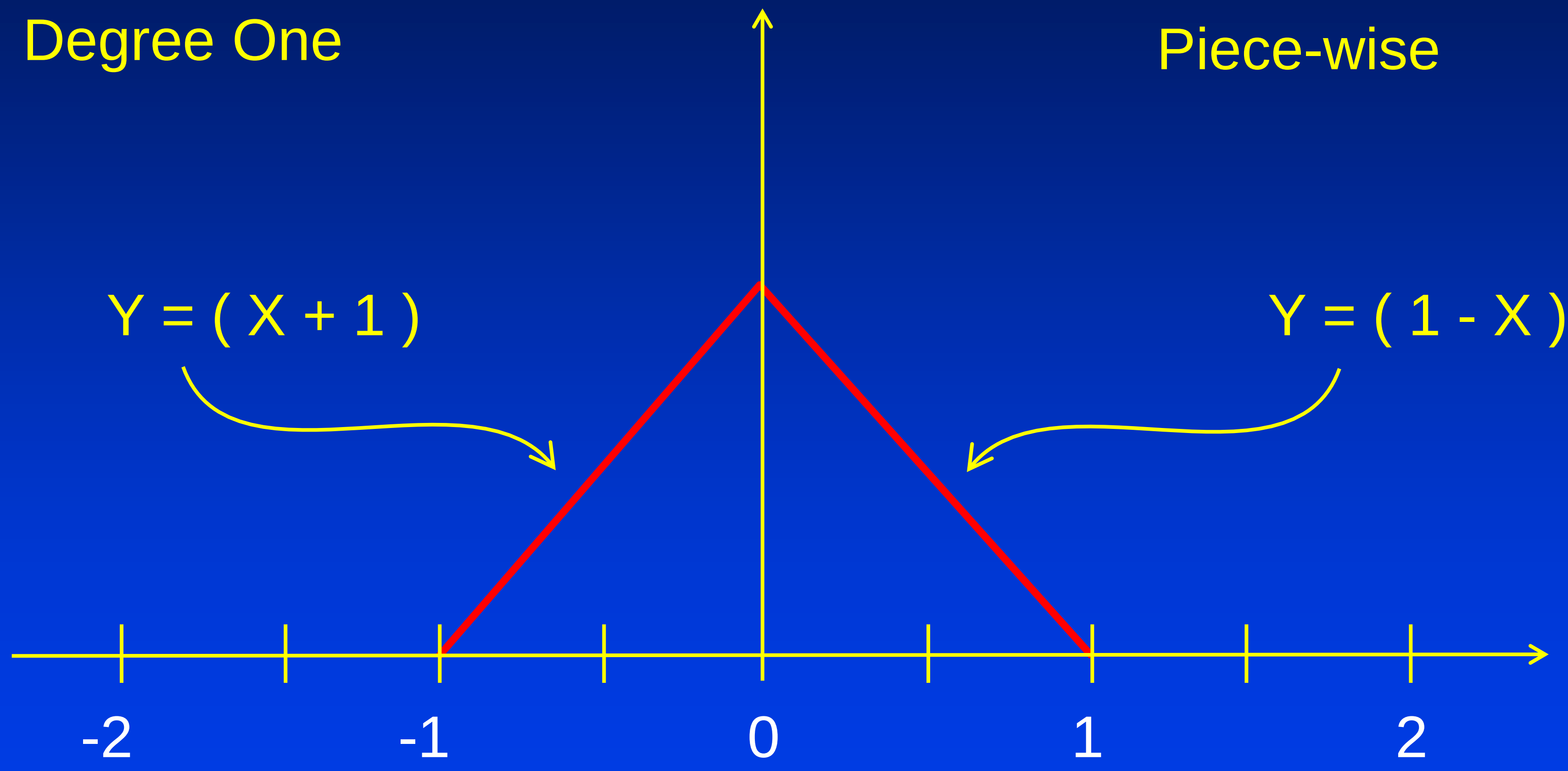


B-Spline Basis Function

Degree One

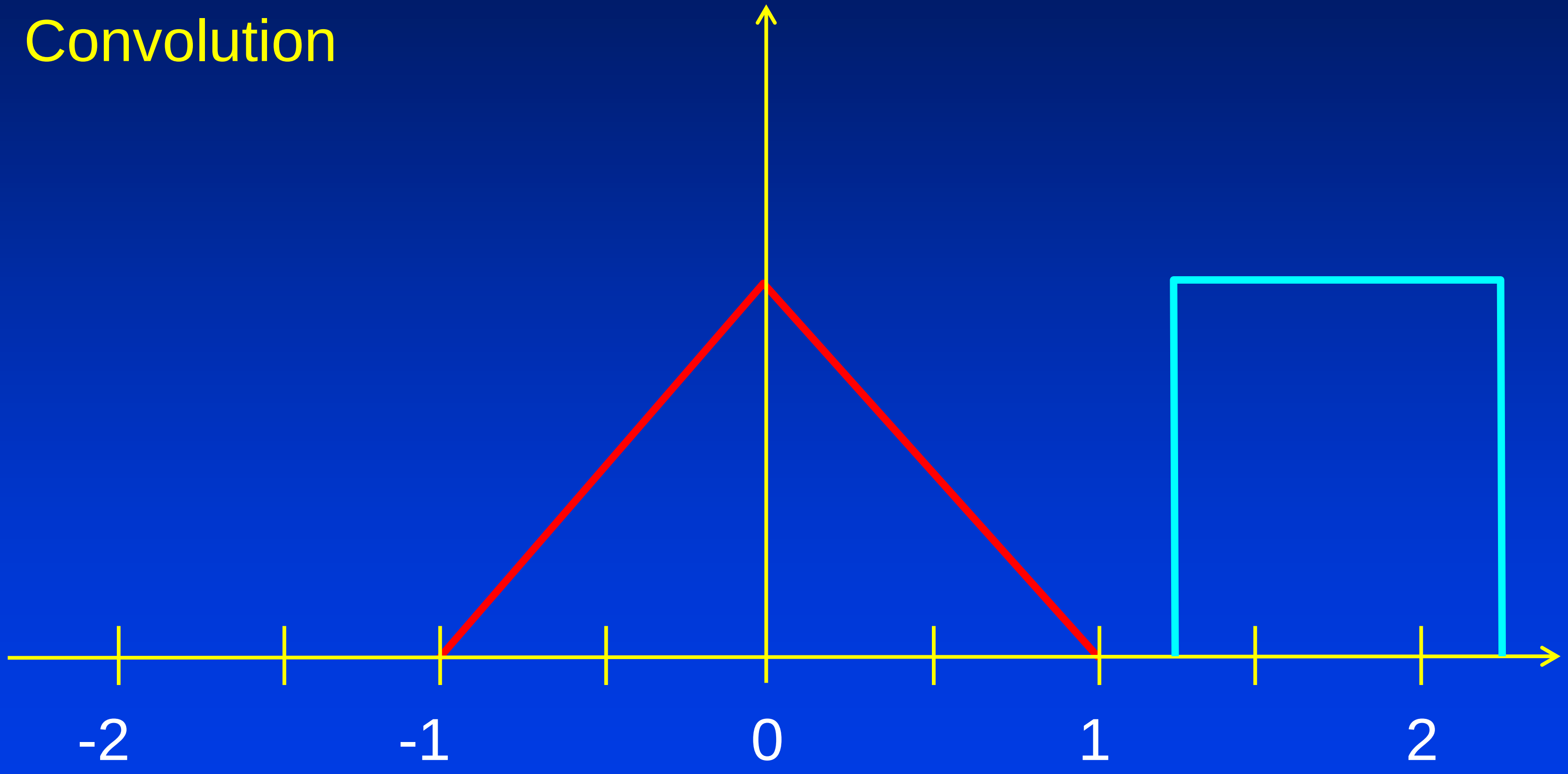


B-Spline Basis Function



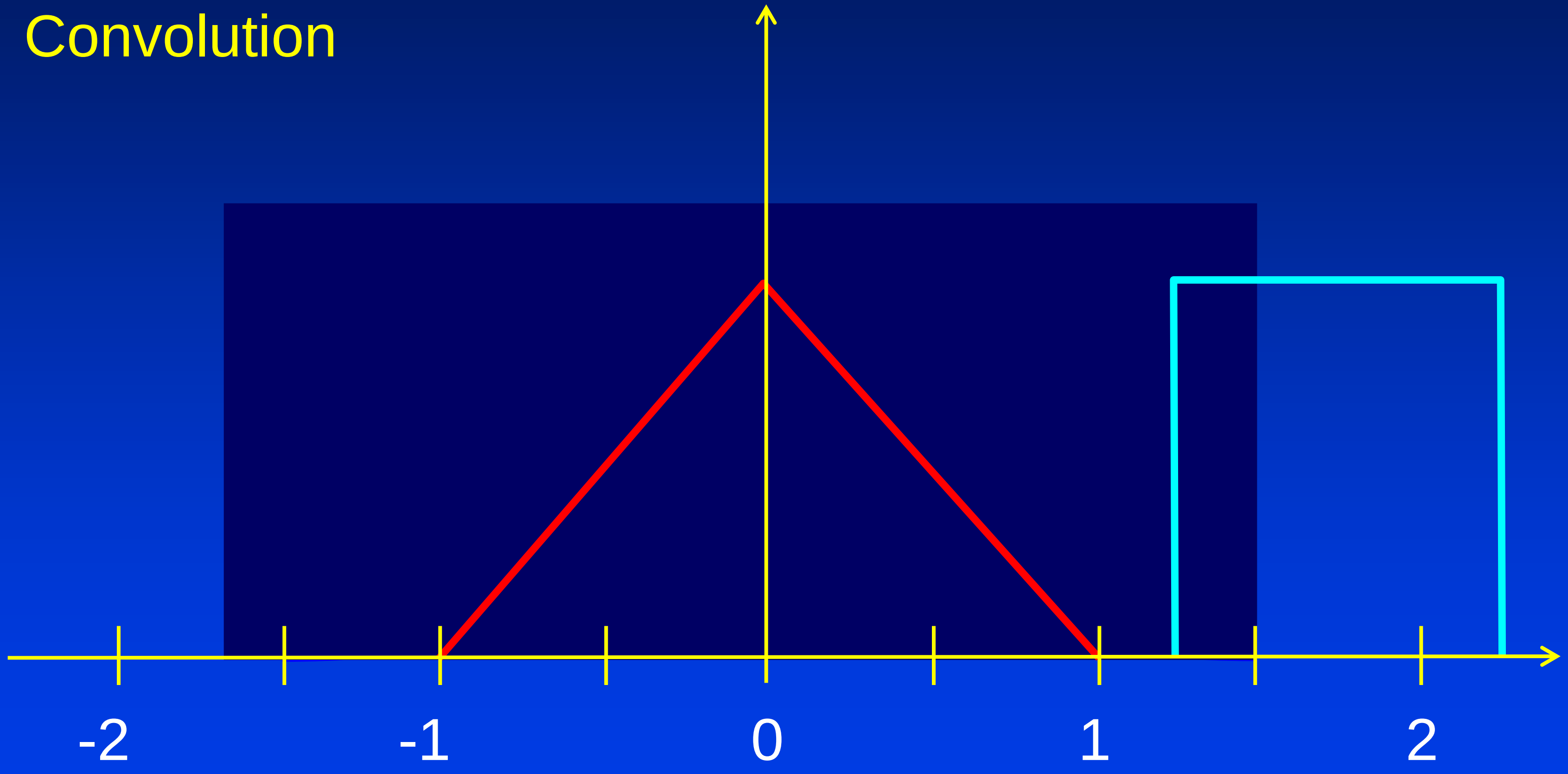
B-Spline Basis Function

Convolution



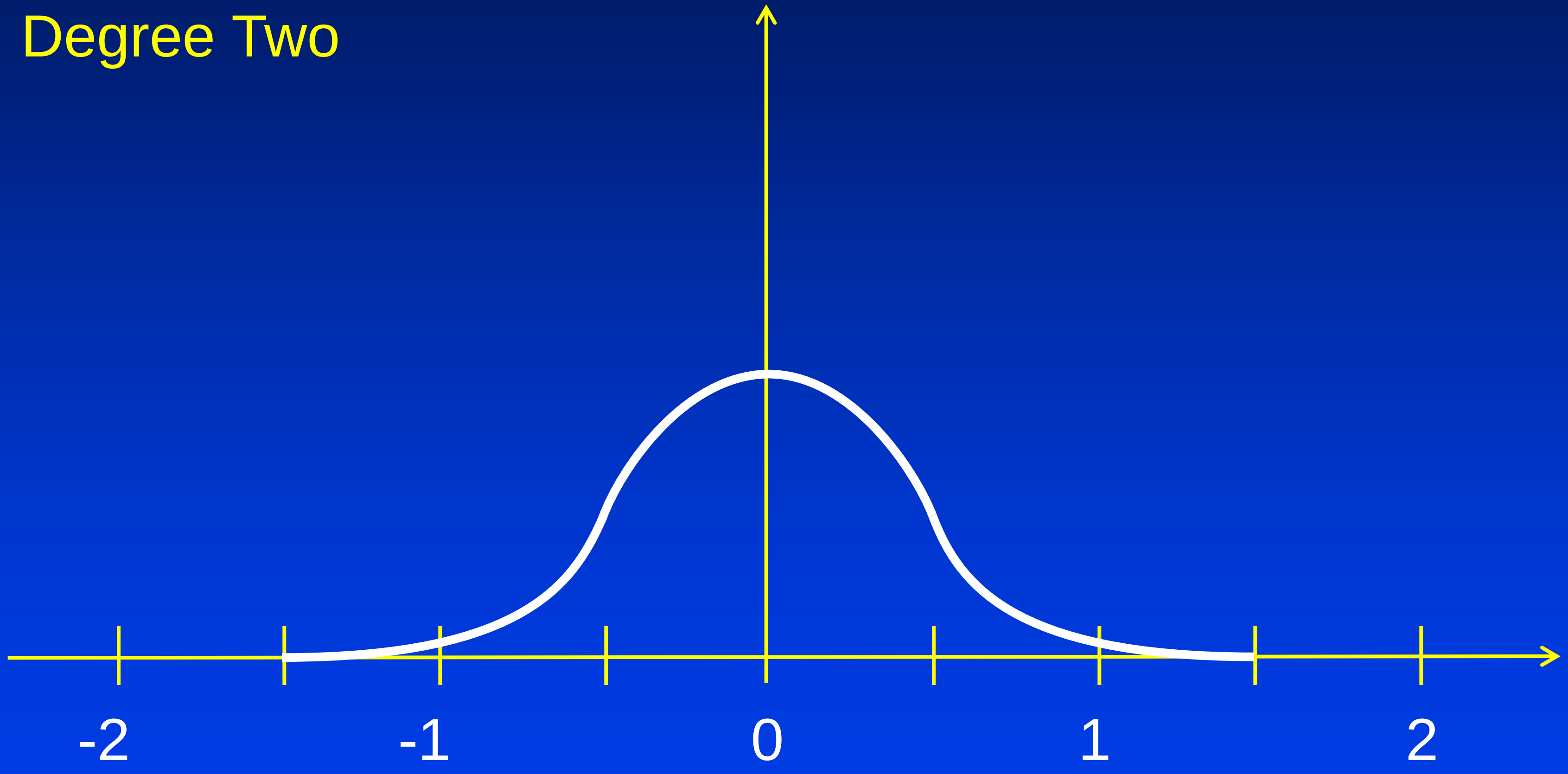
B-Spline Basis Function

Convolution

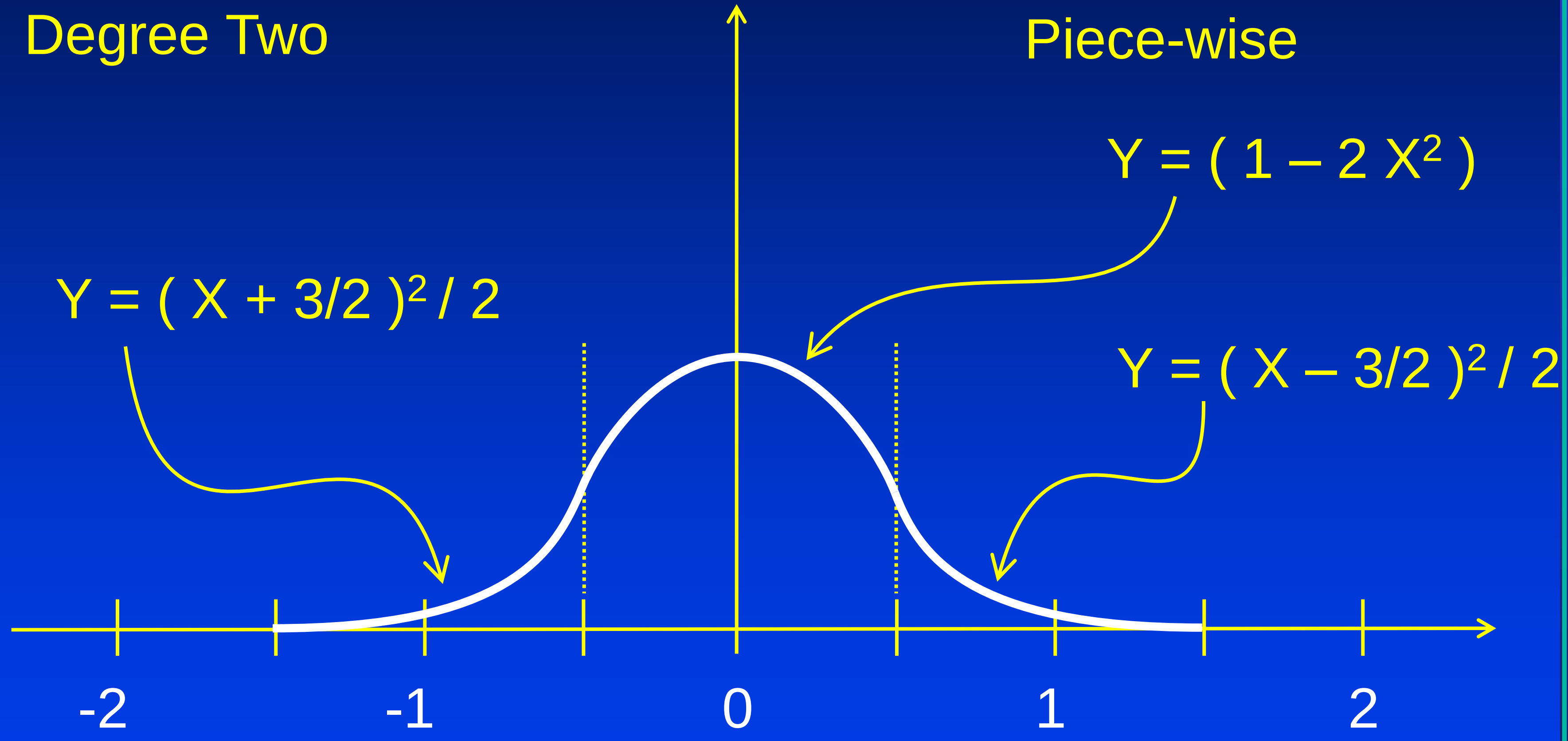


B-Spline Basis Function

Degree Two

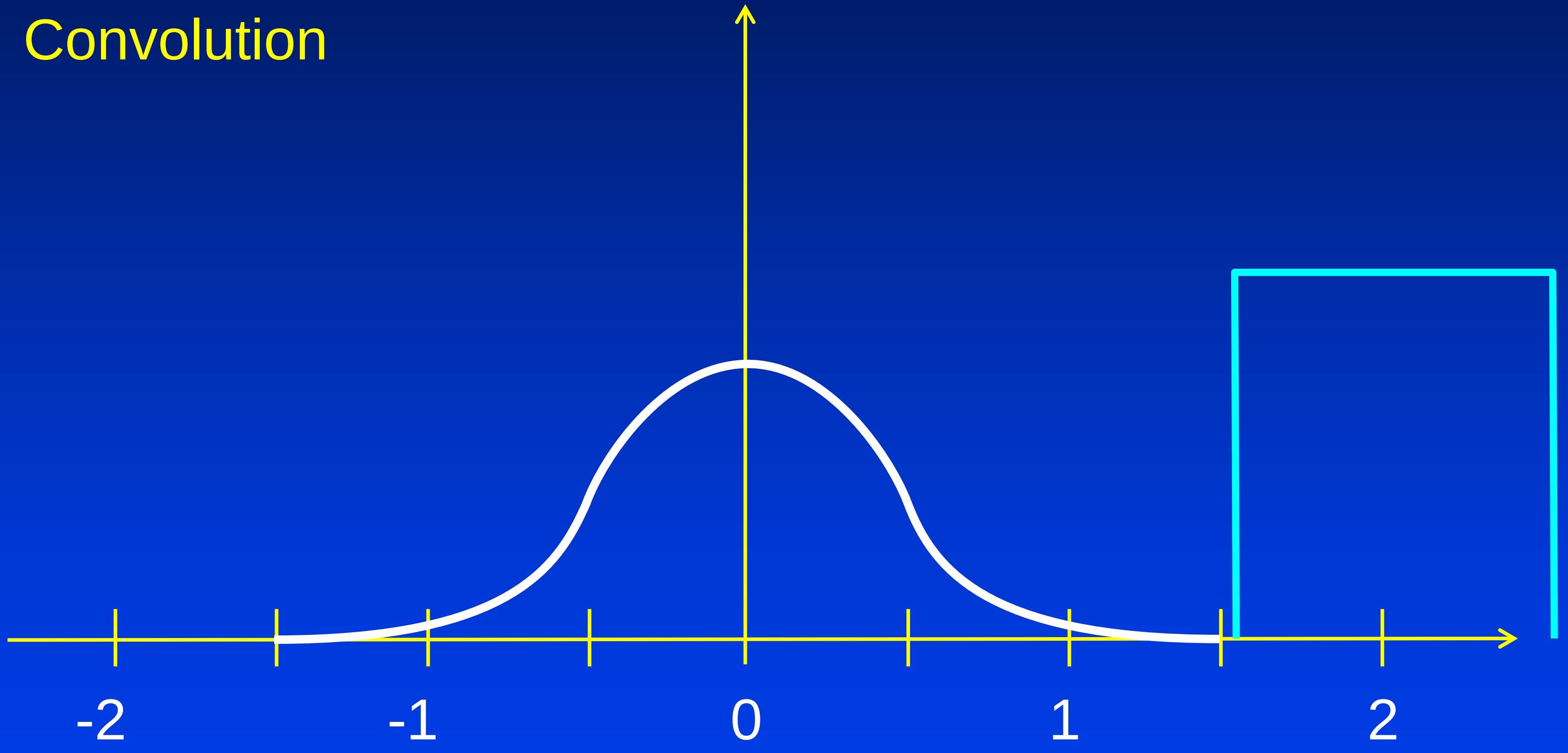


B-Spline Basis Function



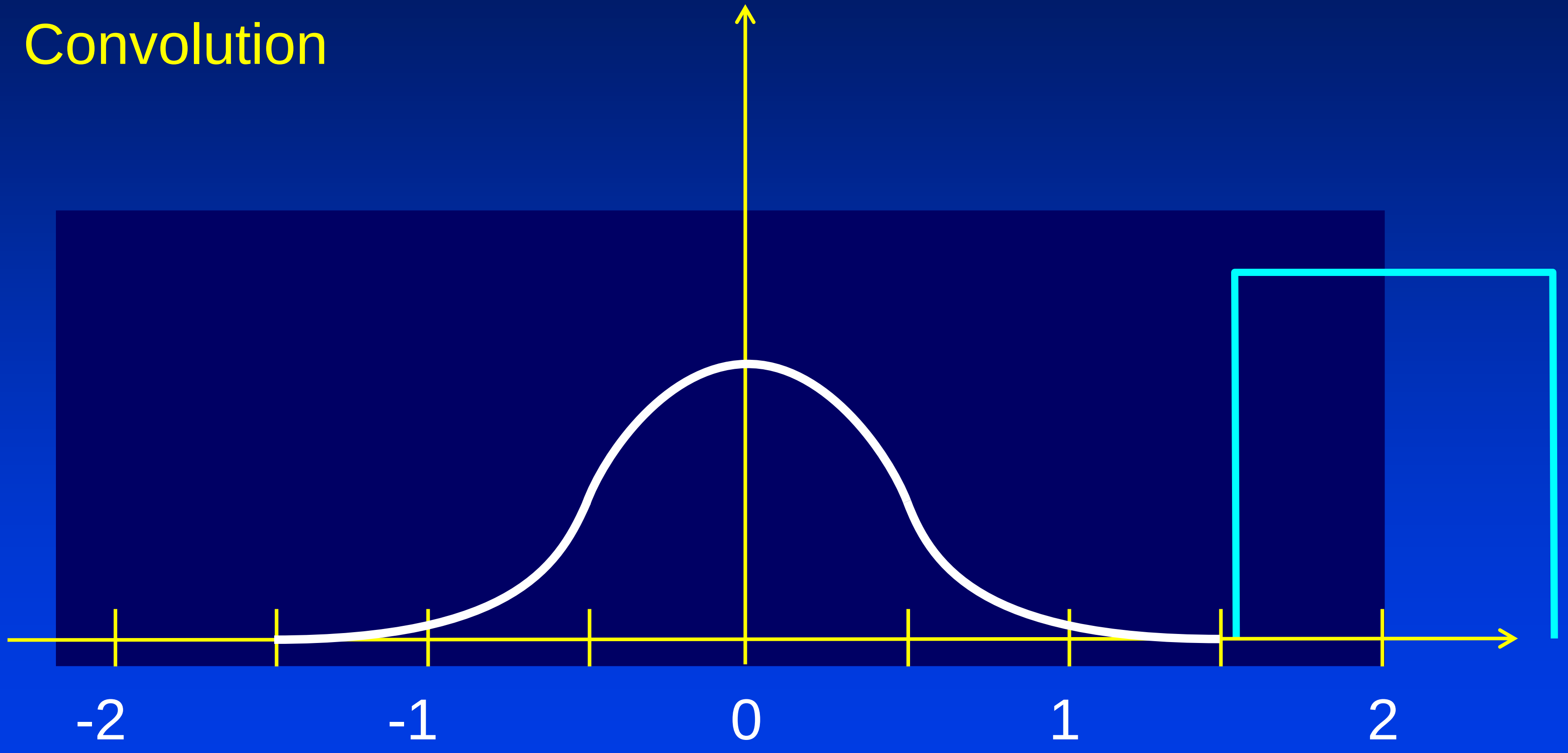
B-Spline Basis Function

Convolution



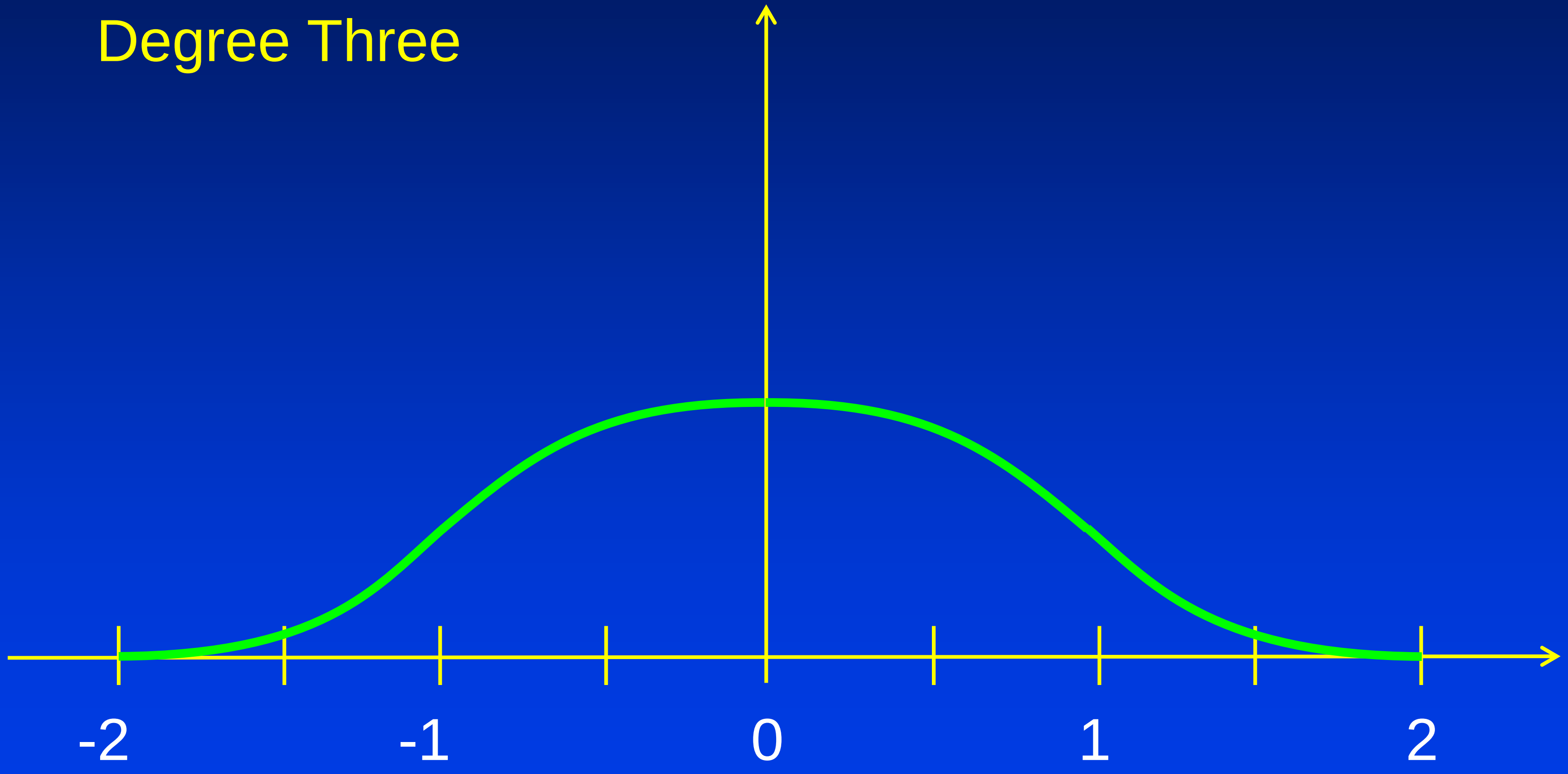
B-Spline Basis Function

Convolution



B-Spline Basis Function

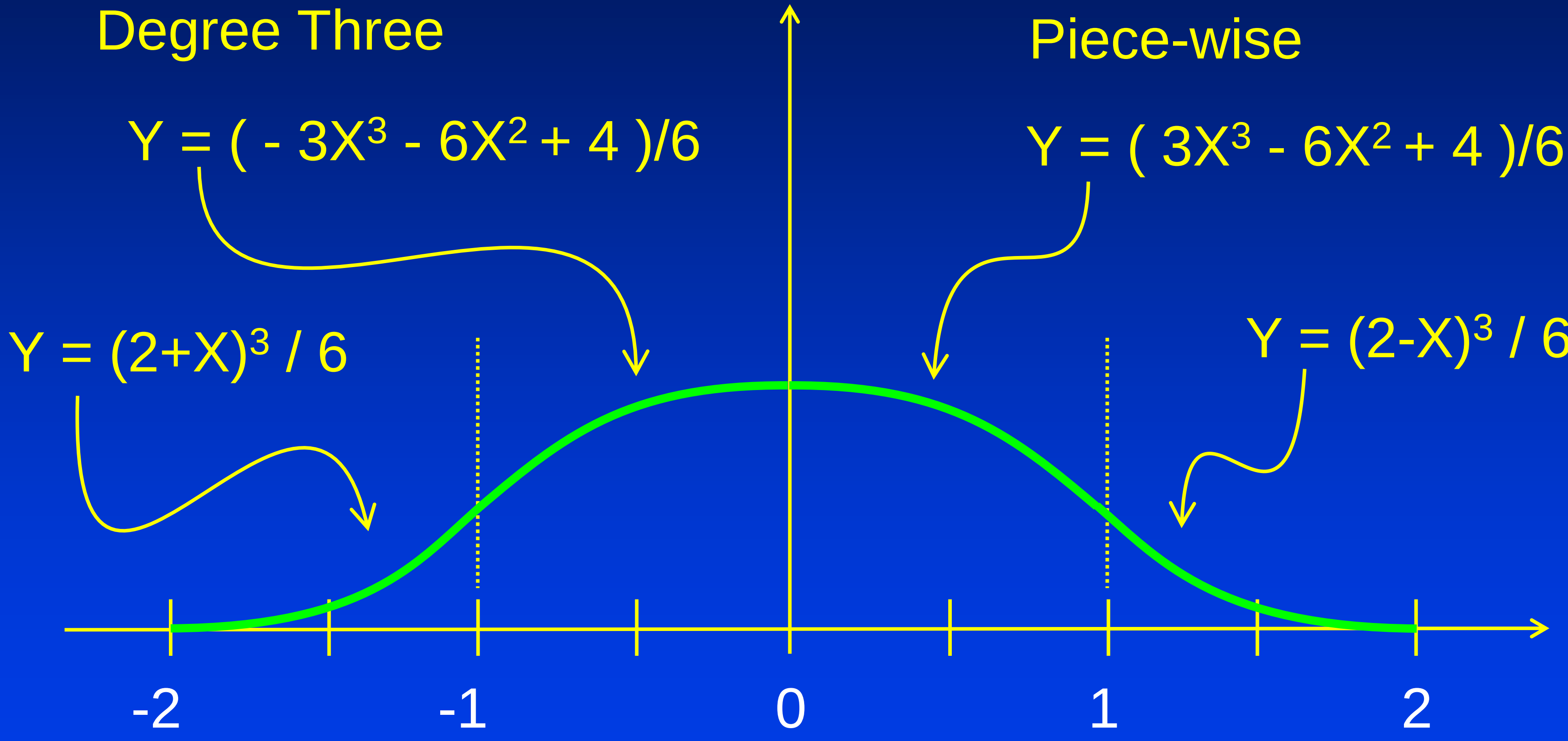
Degree Three



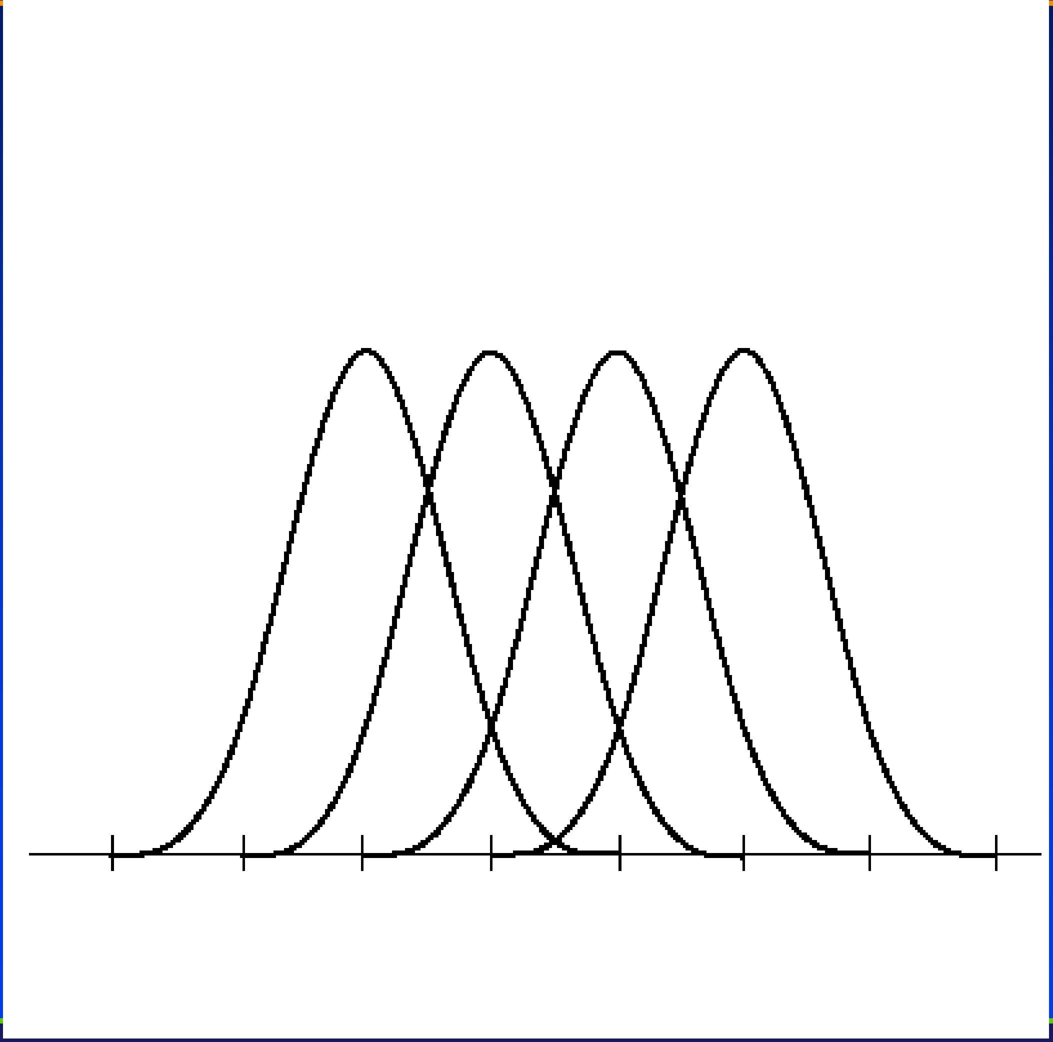
B-Spline Basis Function

Degree Three

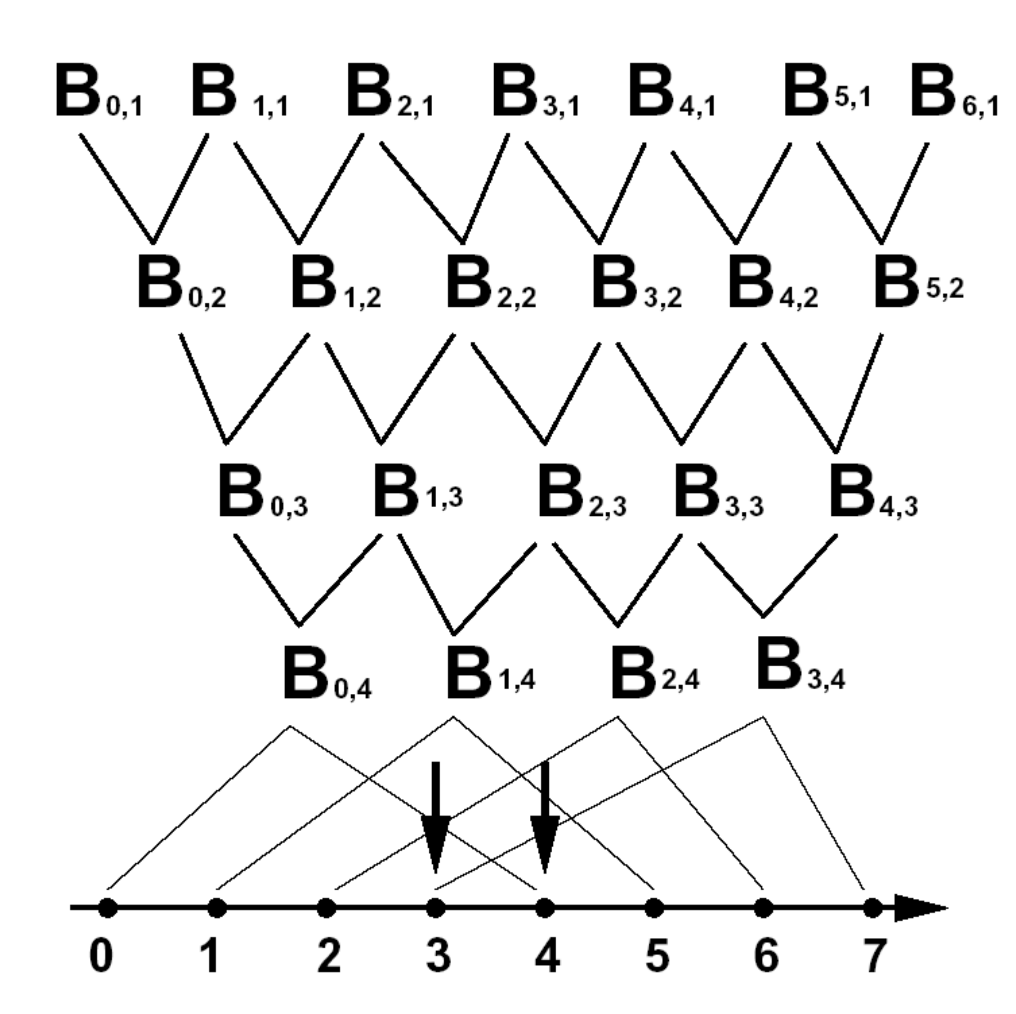
Piece-wise



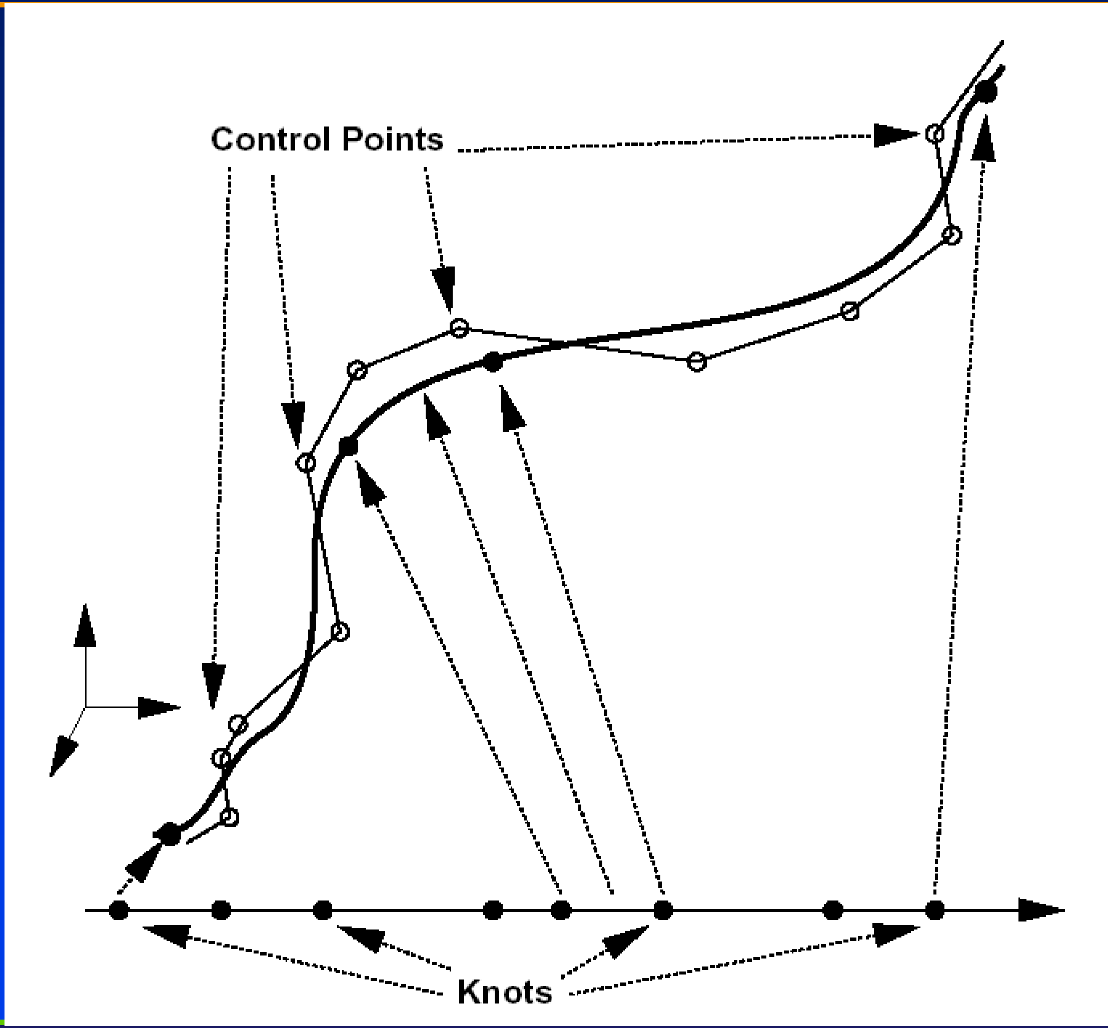
B-Spline Basis Functions



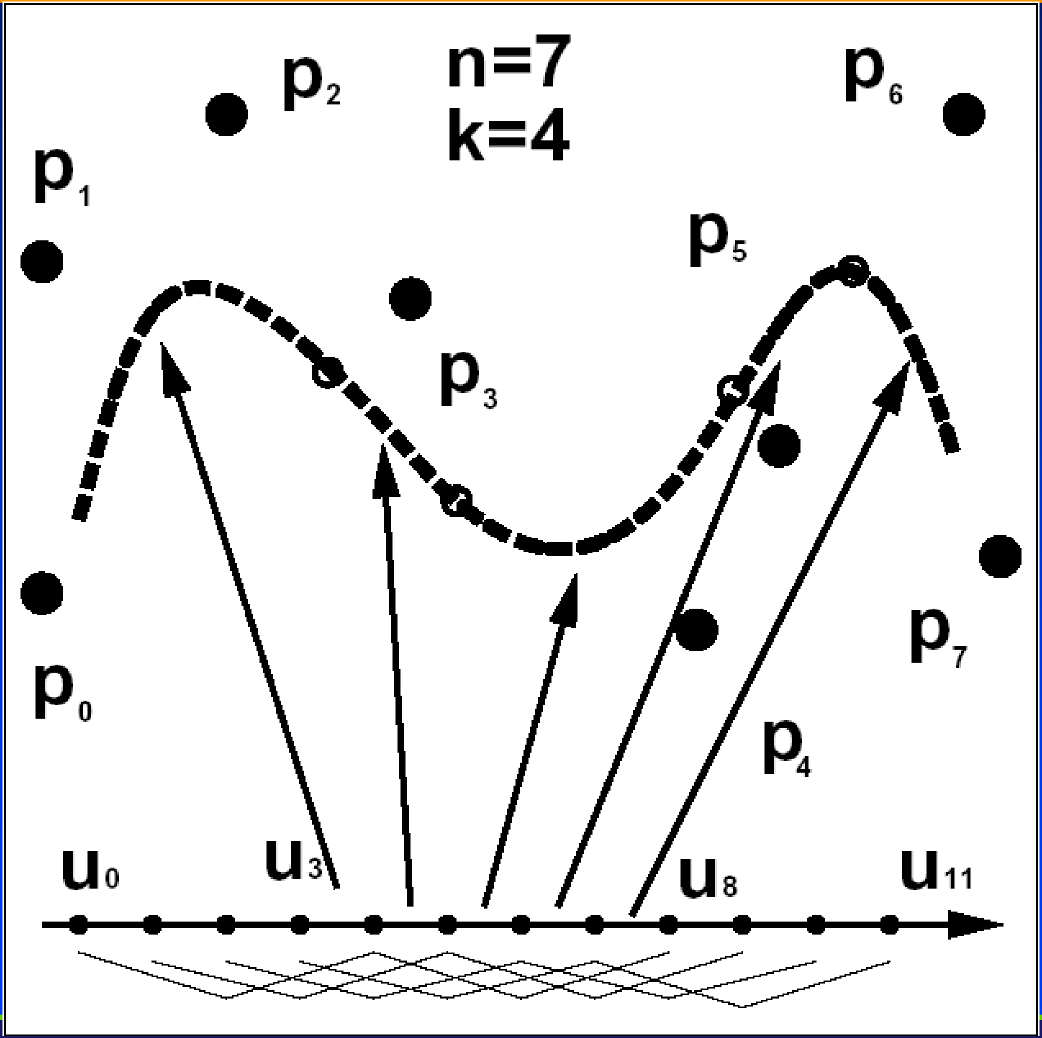
B-Spline Basis Function



B-Splines



B-Splines



B-Spline Applications

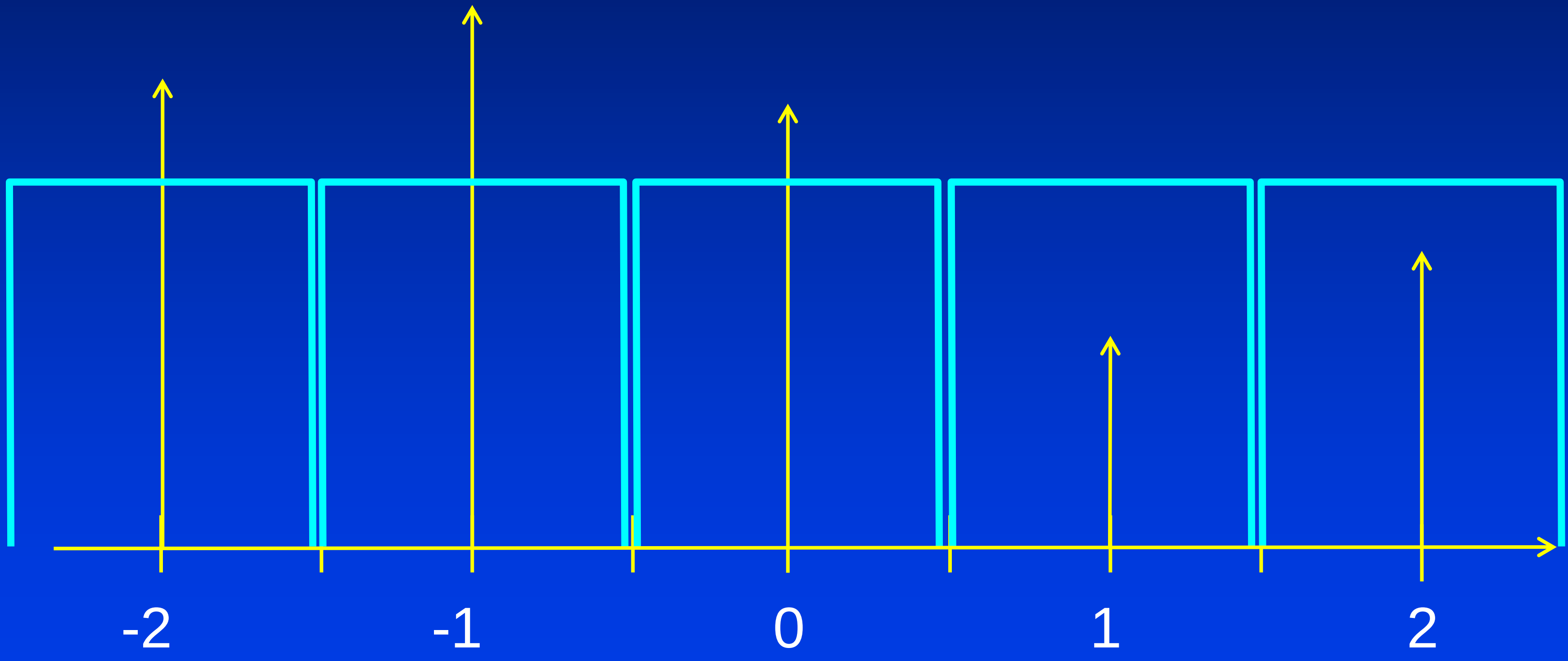
Data Interpolation

with

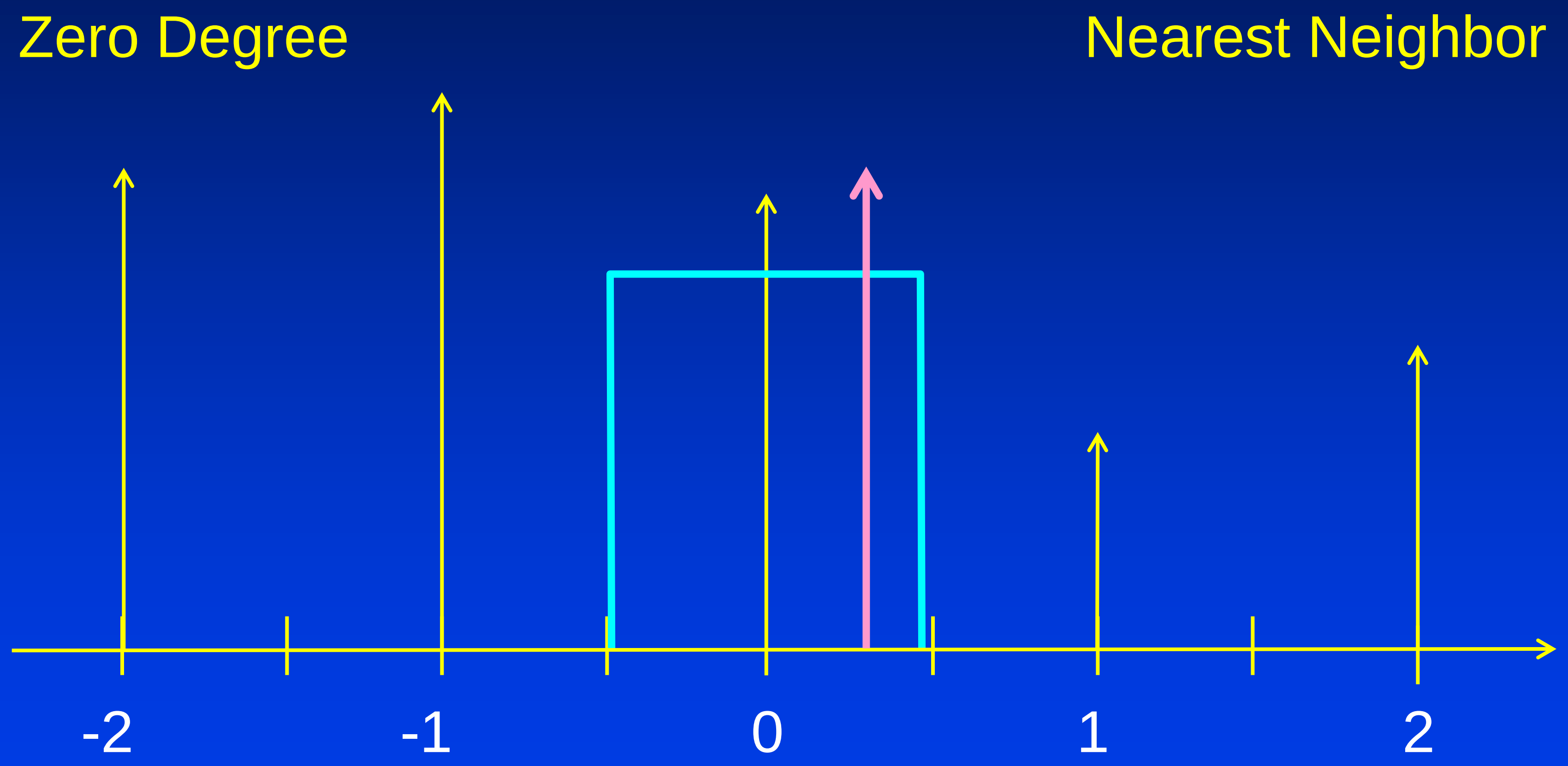
B-Splines

B-Spline Data Interpolation

Zero Degree Nearest Neighbor



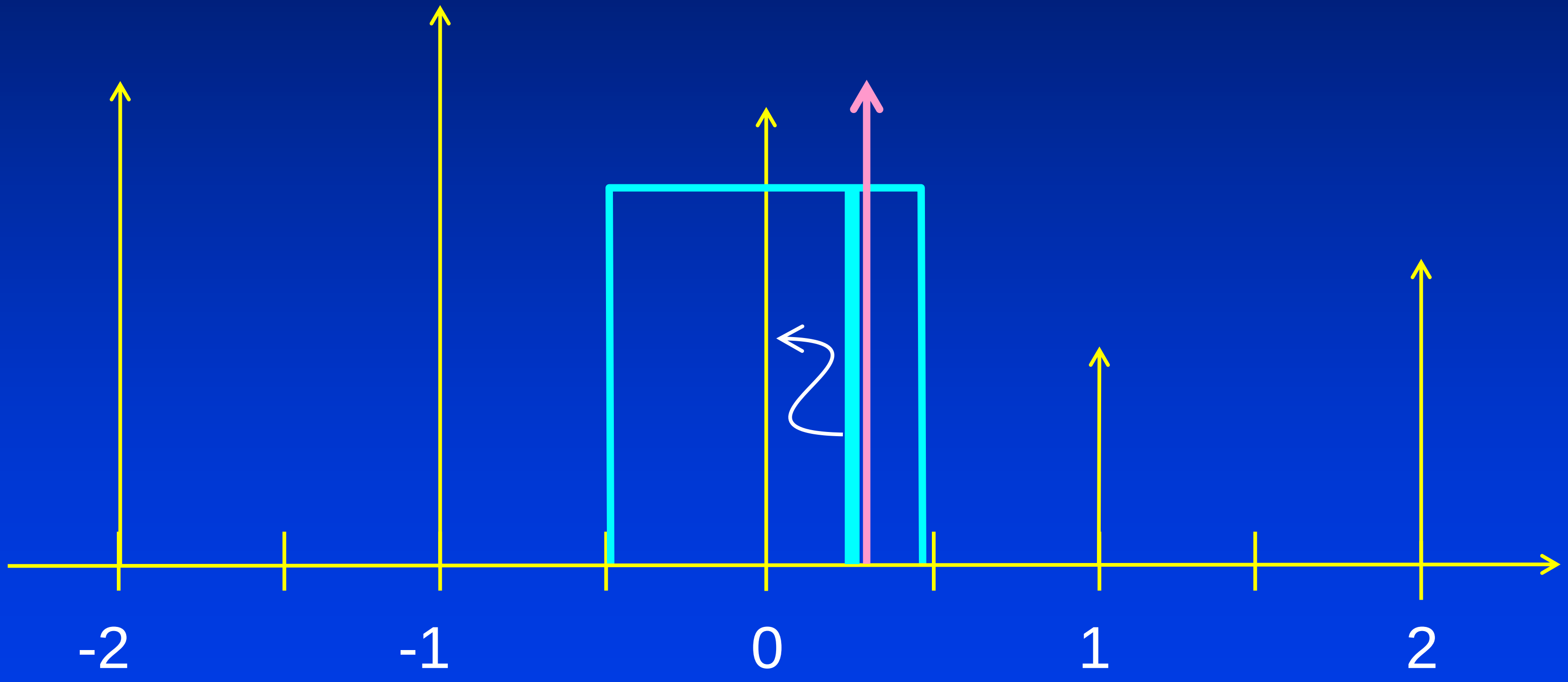
B-Spline Data Interpolation



B-Spline Data Interpolation

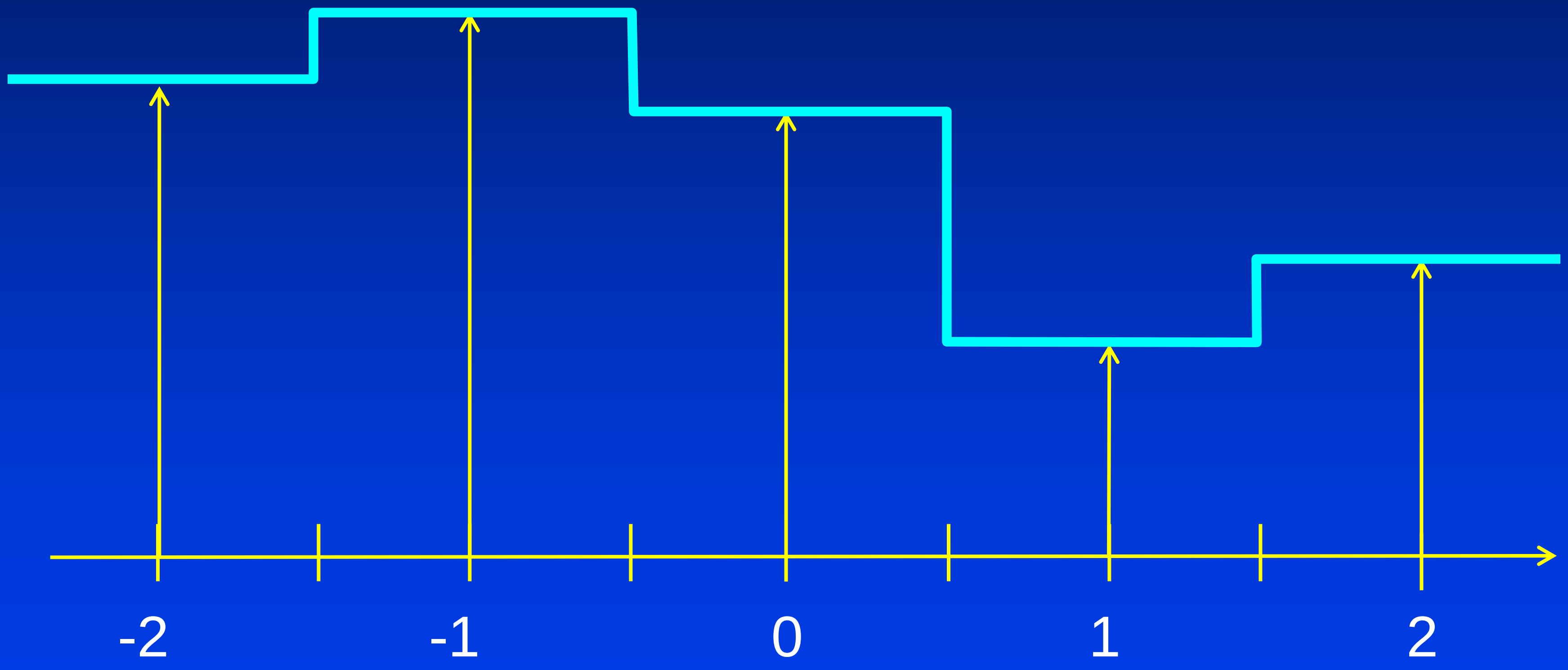
Zero Degree

Nearest Neighbor



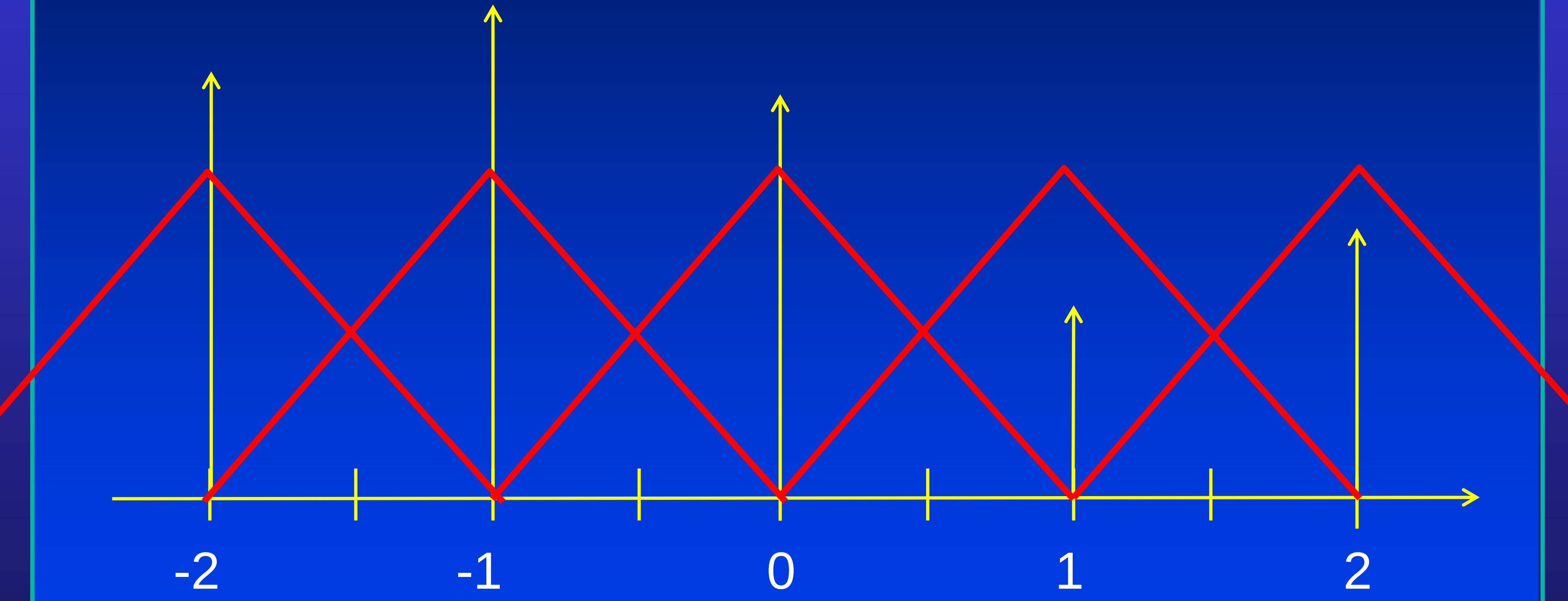
B-Spline Data Interpolation

Zero Degree Nearest Neighbor



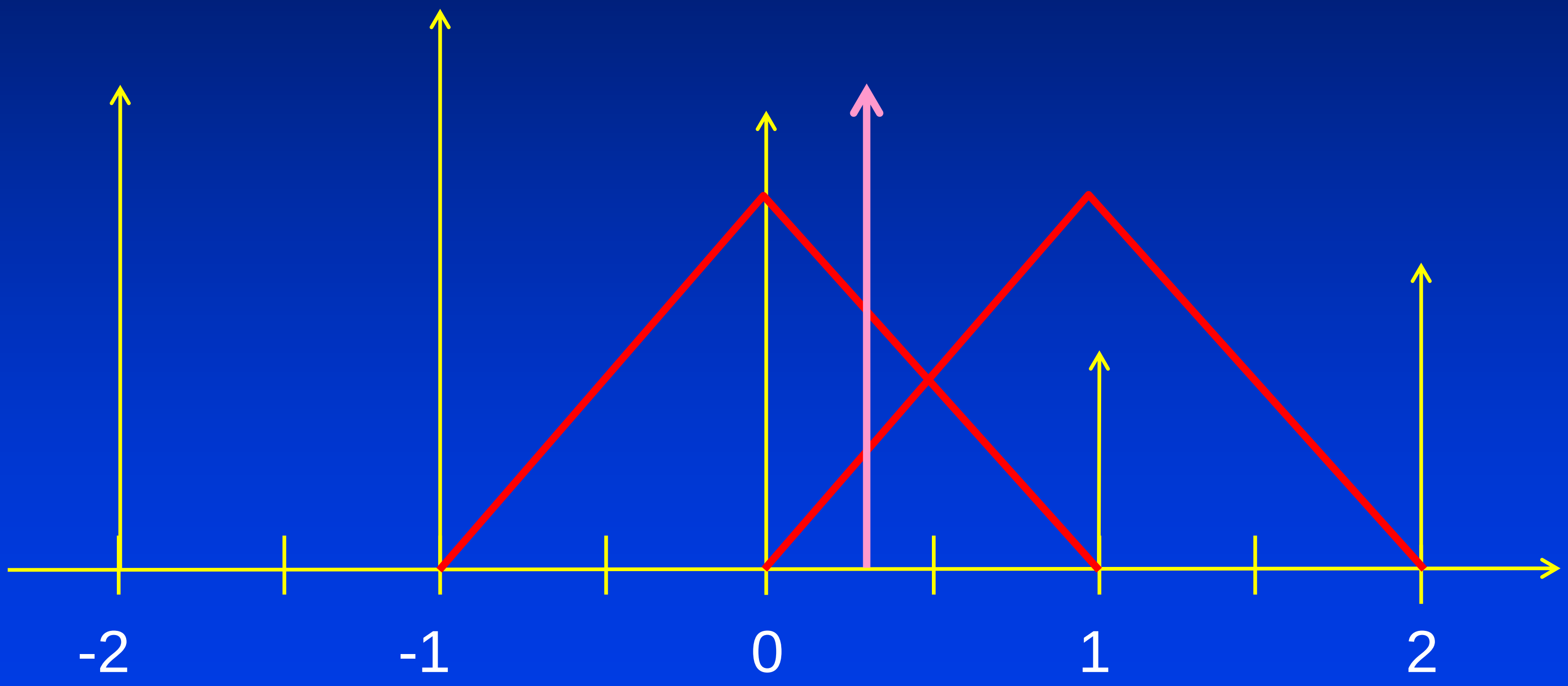
BSplines Interpolation

First Order | Linear Interpolation



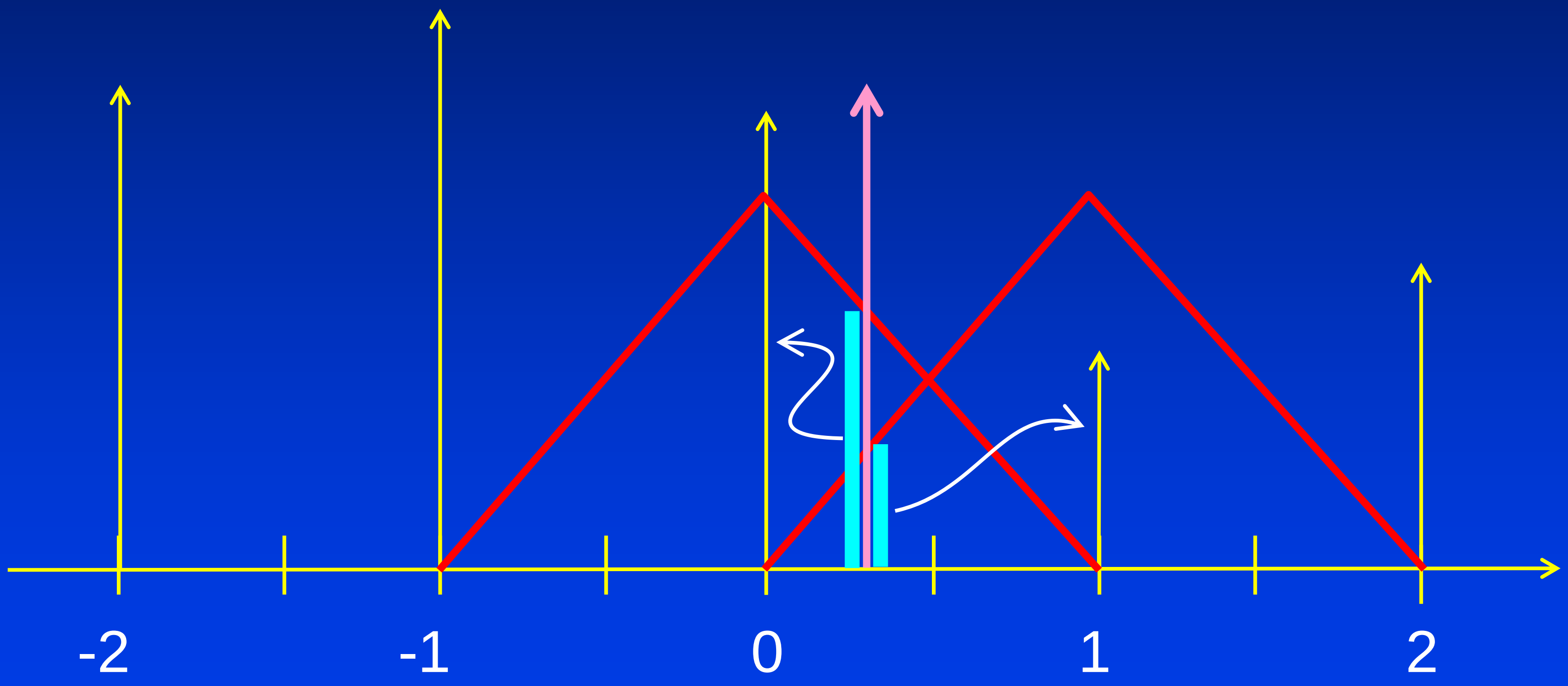
BSplines Interpolation

First Order | Linear Interpolation



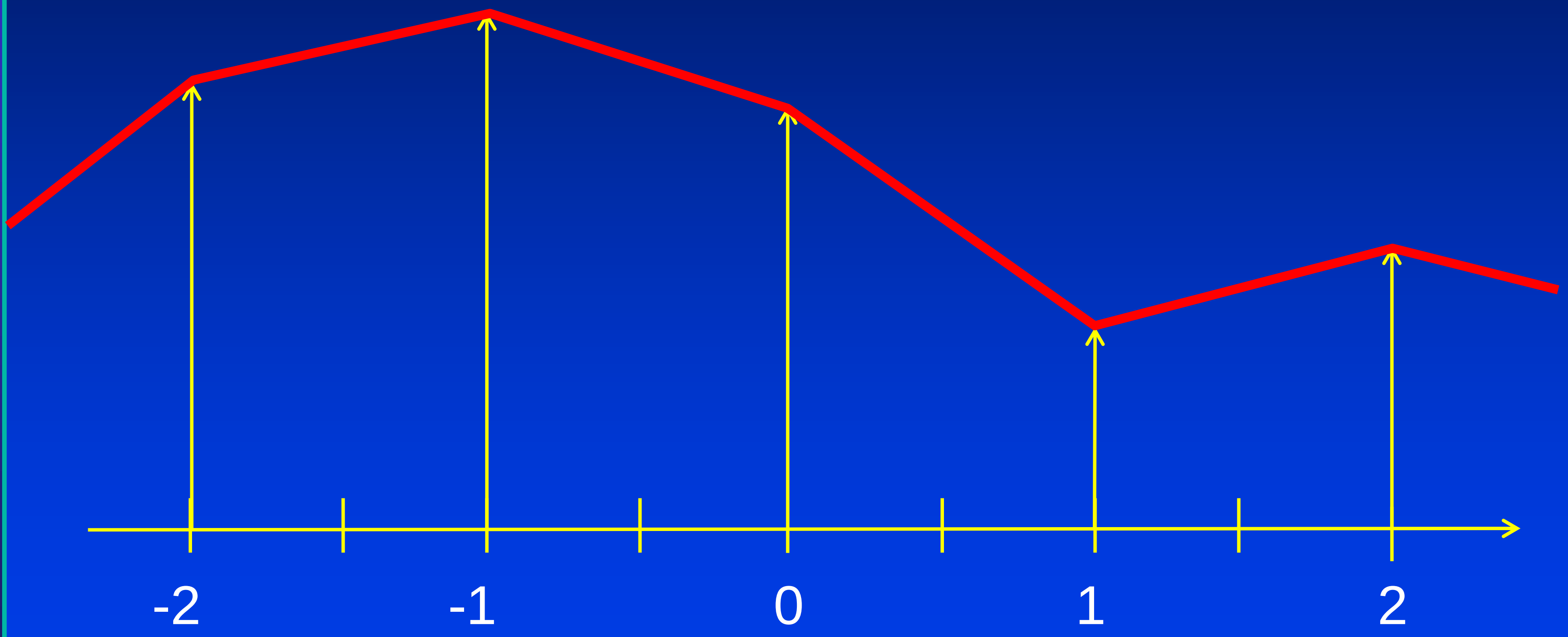
BSplines Interpolation

First Order Linear Interpolation



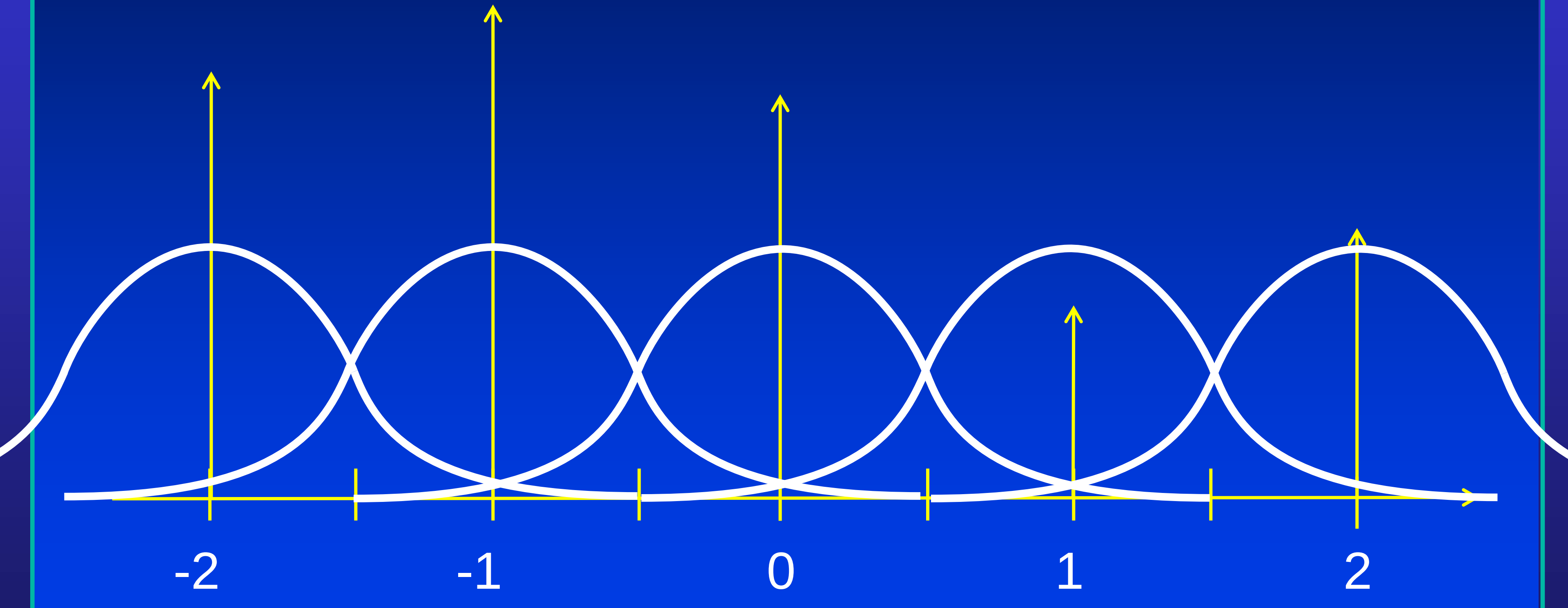
BSplines Interpolation

First Order Linear Interpolator



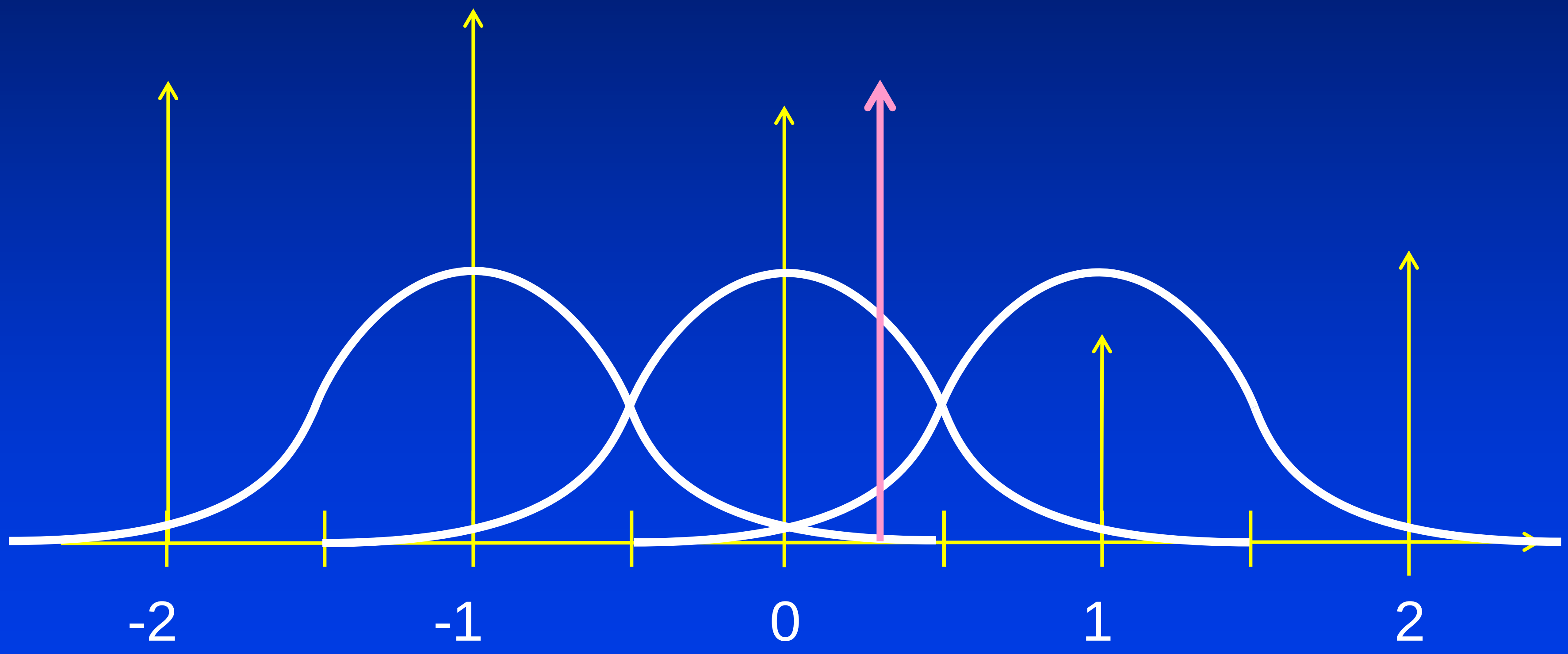
BSplines Interpolation

Second Order Quadratic Interpolation



BSplines Interpolation

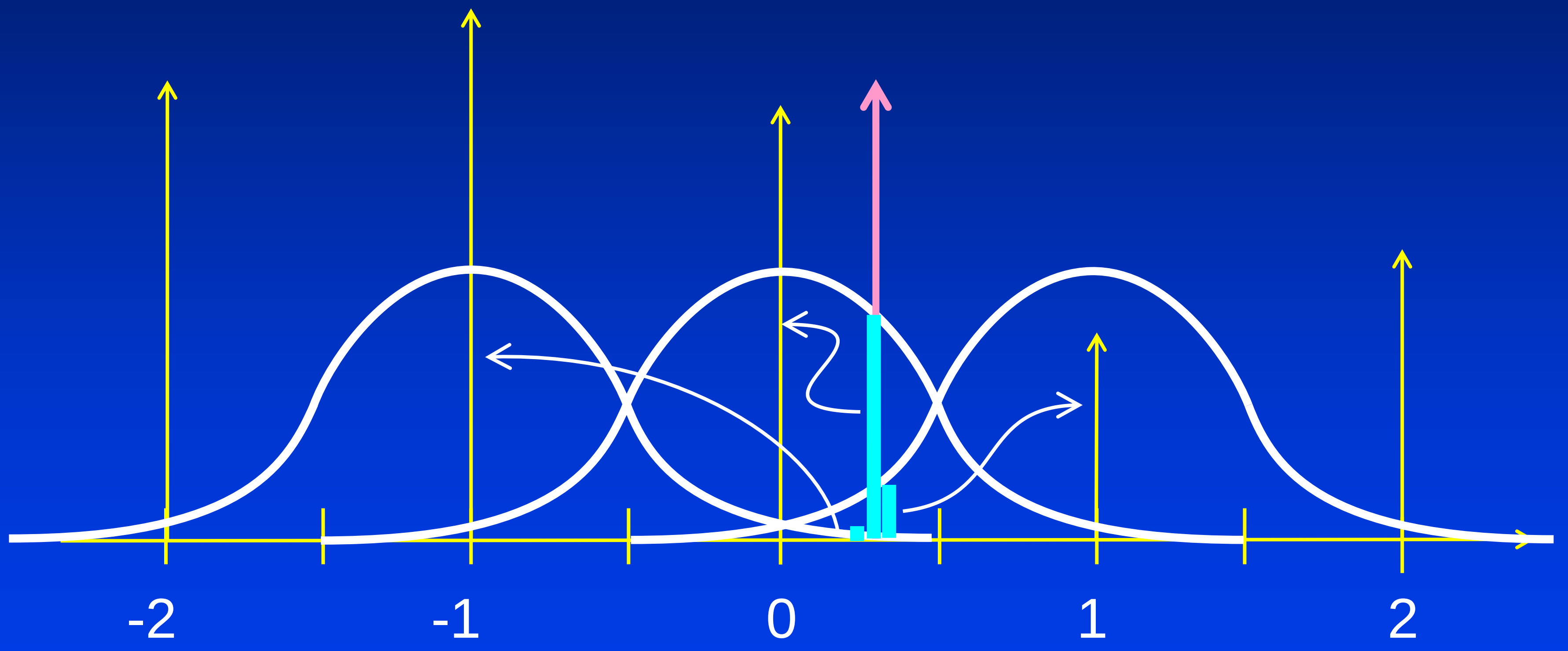
Second Order Quadratic Interpolation



BSplines Interpolation

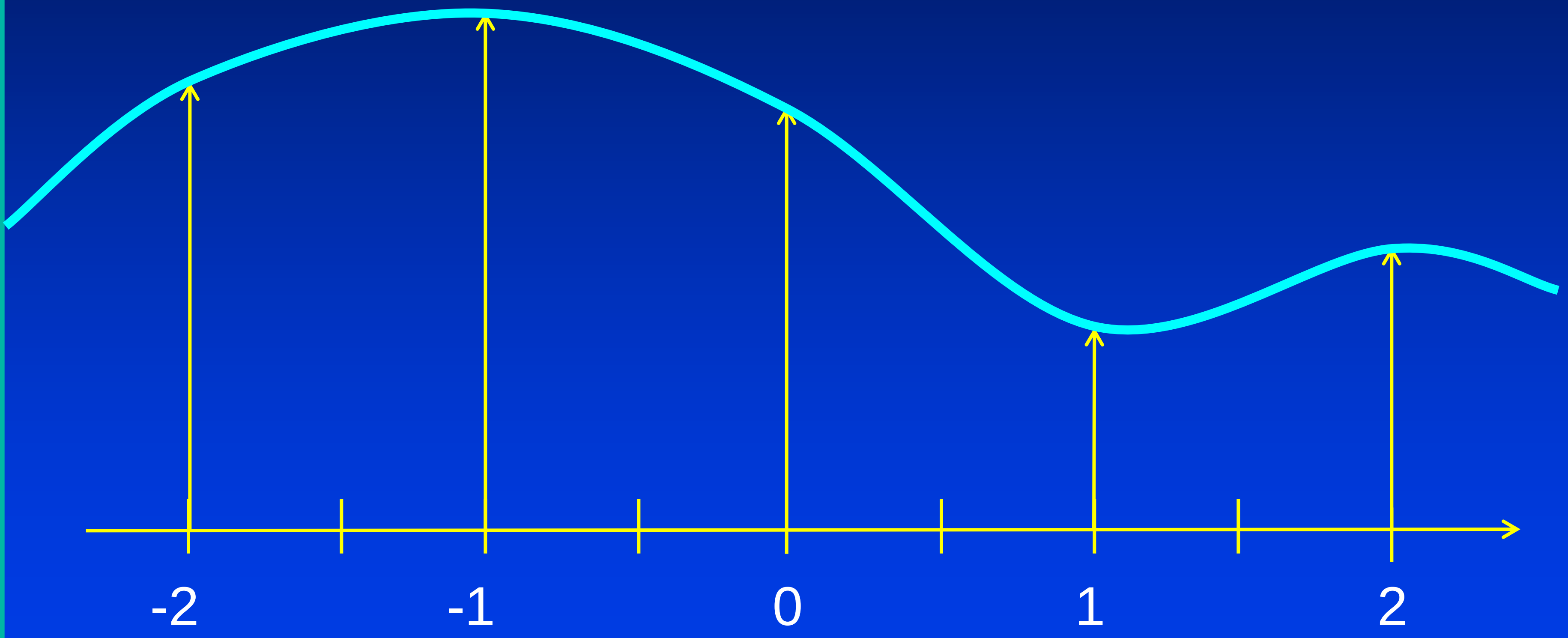
Second Order

Quadratic Interpolation



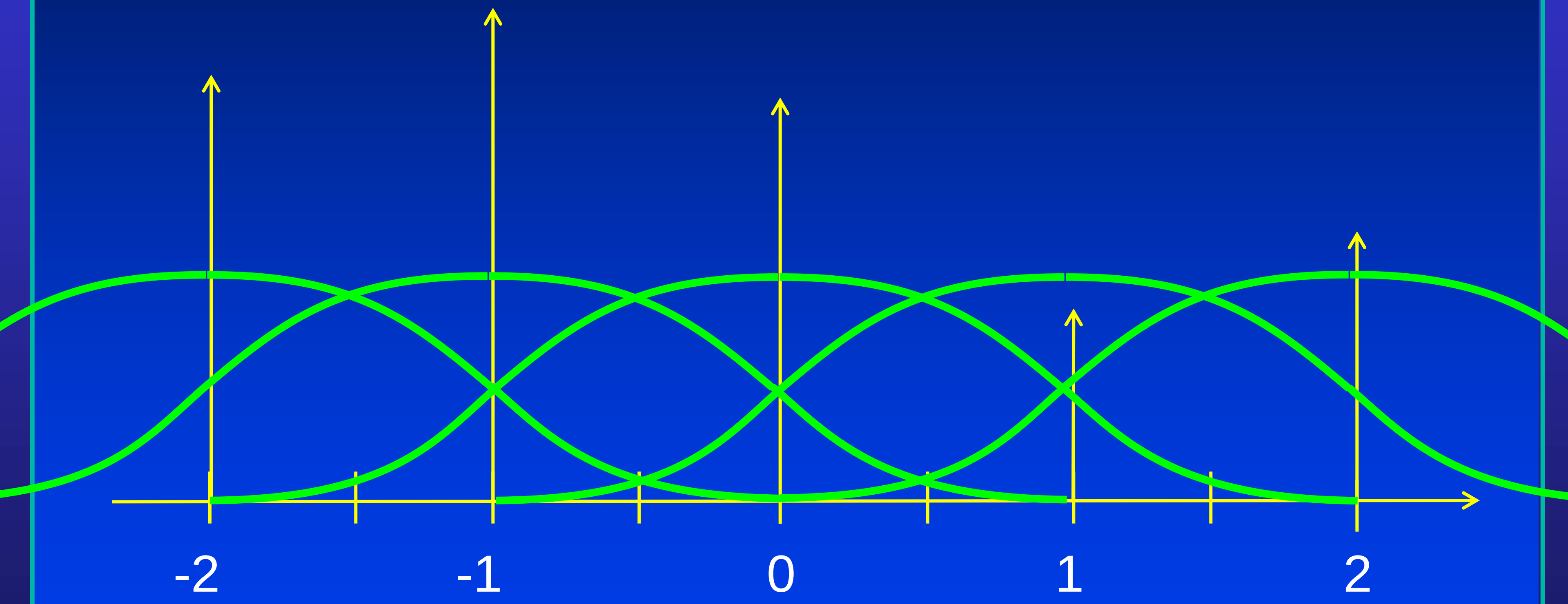
BSplines Interpolation

Second Order Quadratic Interpolator

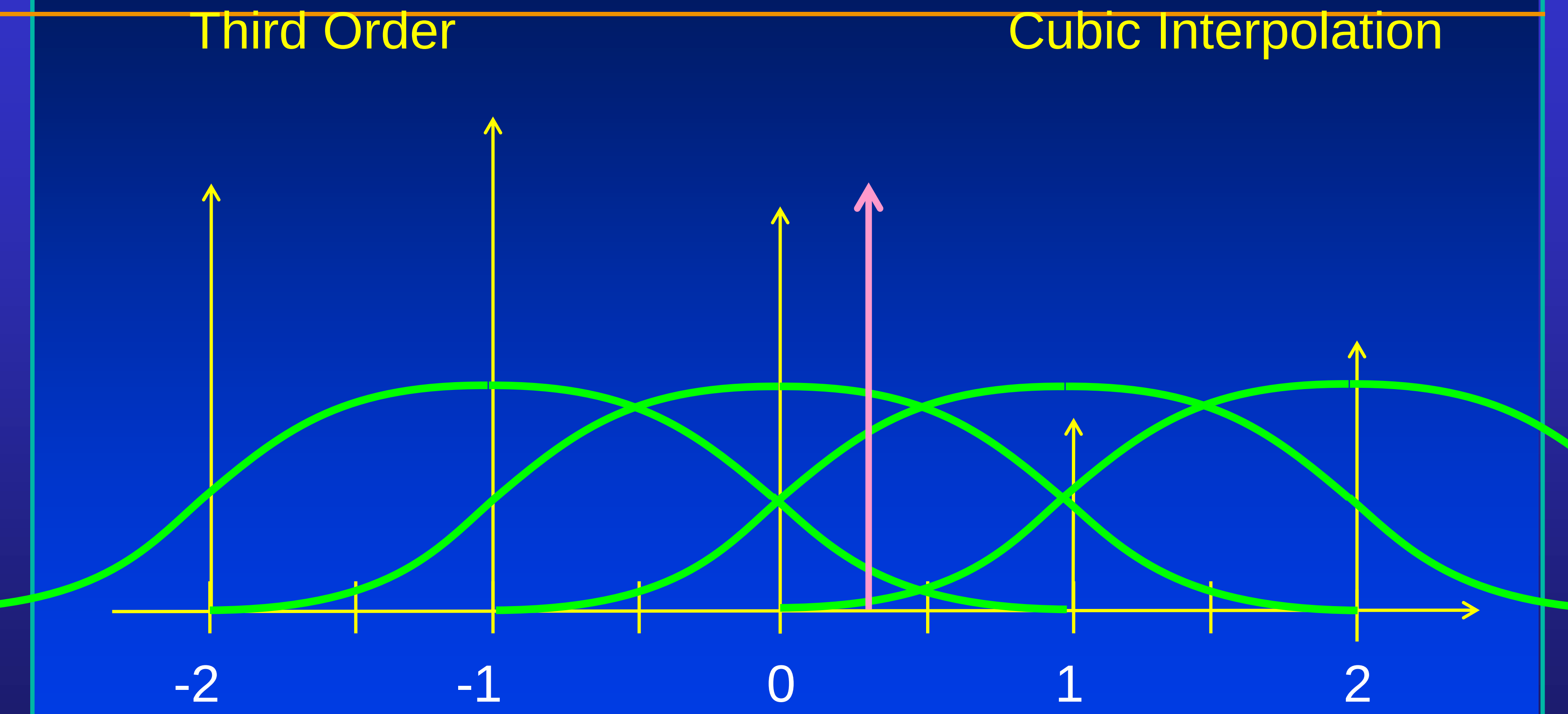


BSplines Interpolation

Third Order — Cubic Interpolation



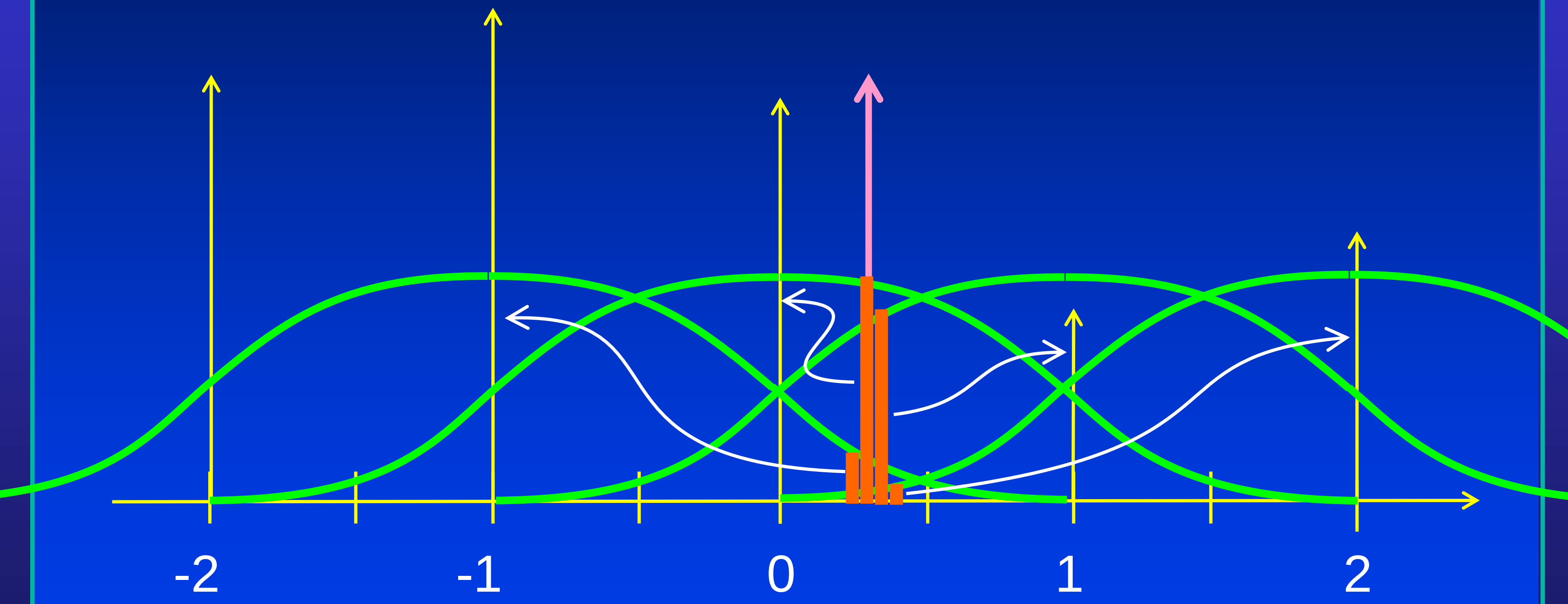
BSplines Interpolation



BSplines Interpolation

Third Order

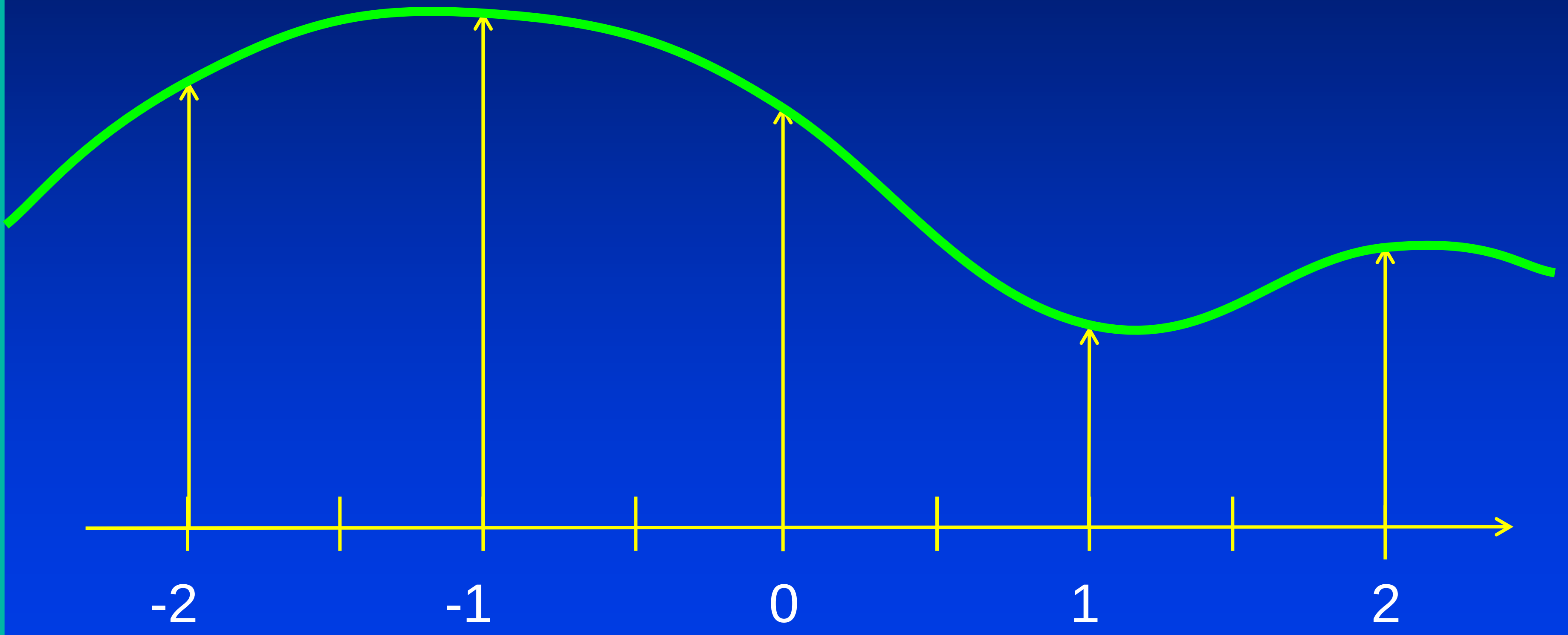
Cubic Interpolation



BSplines Interpolation

Third Order

Cubic Interpolator



B-Splines

- Mathematics $$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(u)$$
- Control points and basis functions of degree (k-1)
- Piecewise polynomials
- Basis functions are defined recursively
- We also have to introduce a knot sequence (n+k+1) in a non-decreasing order
$$u_0, u_1, u_2, u_3, \ldots\ldots, u_{n+k}$$
- Note that, the parametric domain: $u \in [u_{k-1}, u_{n+1}]$

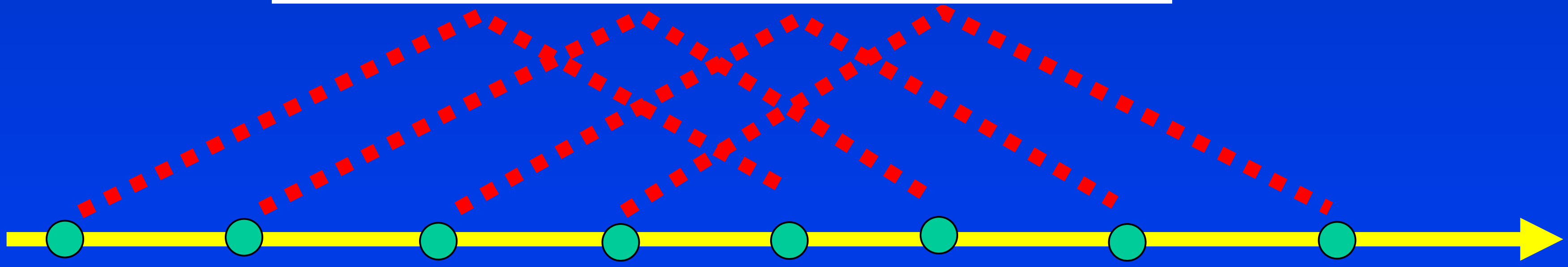
Basis Functions

$$B_{0,1} \quad B_{1,1} \quad B_{2,1} \quad B_{3,1} \quad B_{4,1} \quad B_{5,1} \quad B_{6,1}$$

$$B_{0,2} \quad B_{1,2} \quad B_{2,2} \quad B_{3,2} \quad B_{4,2} \quad B_{5,2}$$

$$B_{0,3} \quad B_{1,3} \quad B_{2,3} \quad B_{3,3} \quad B_{4,3}$$

$$B_{0,4} \quad B_{1,4} \quad B_{2,4} \quad B_{3,4}$$



B-Spline Facts

- The curve is a linear combination of control points and their associated basis functions ((n+1) control points and basis functions, respectively)
- Basis functions are piecewise polynomials defined (recursively) over a set of non-decreasing knots

$$\{u_0, \ldots\ldots, u_{k-1}, \ldots\ldots, u_{n+1}, \ldots\ldots, u_{n+k}\}$$

- The degree of basis functions is independent of the number of control points (note that, I is index, k is the order, k-1 is the degree)
- The first k and last k knots do NOT contribute to the parametric domain. Parametric domain is only defined by a subset of knots

B-Spline Properties

- C(u): piecewise polynomial of degree (k-1)
- Continuity at joints: C(k-2)
- The number of control points and basis functions: (n+1)
- One typical basis function is defined over k sub-intervals which are specified by k+1 knots ([u(k),u(I+k)])
- There are n+k+1 knots in total, knot sequence divides the parametric axis into n+k sub-intervals
- There are (n+1)-(k-1)=n-k+2 sub-intervals within the parametric domain ([u(k-1),u(n+1)])

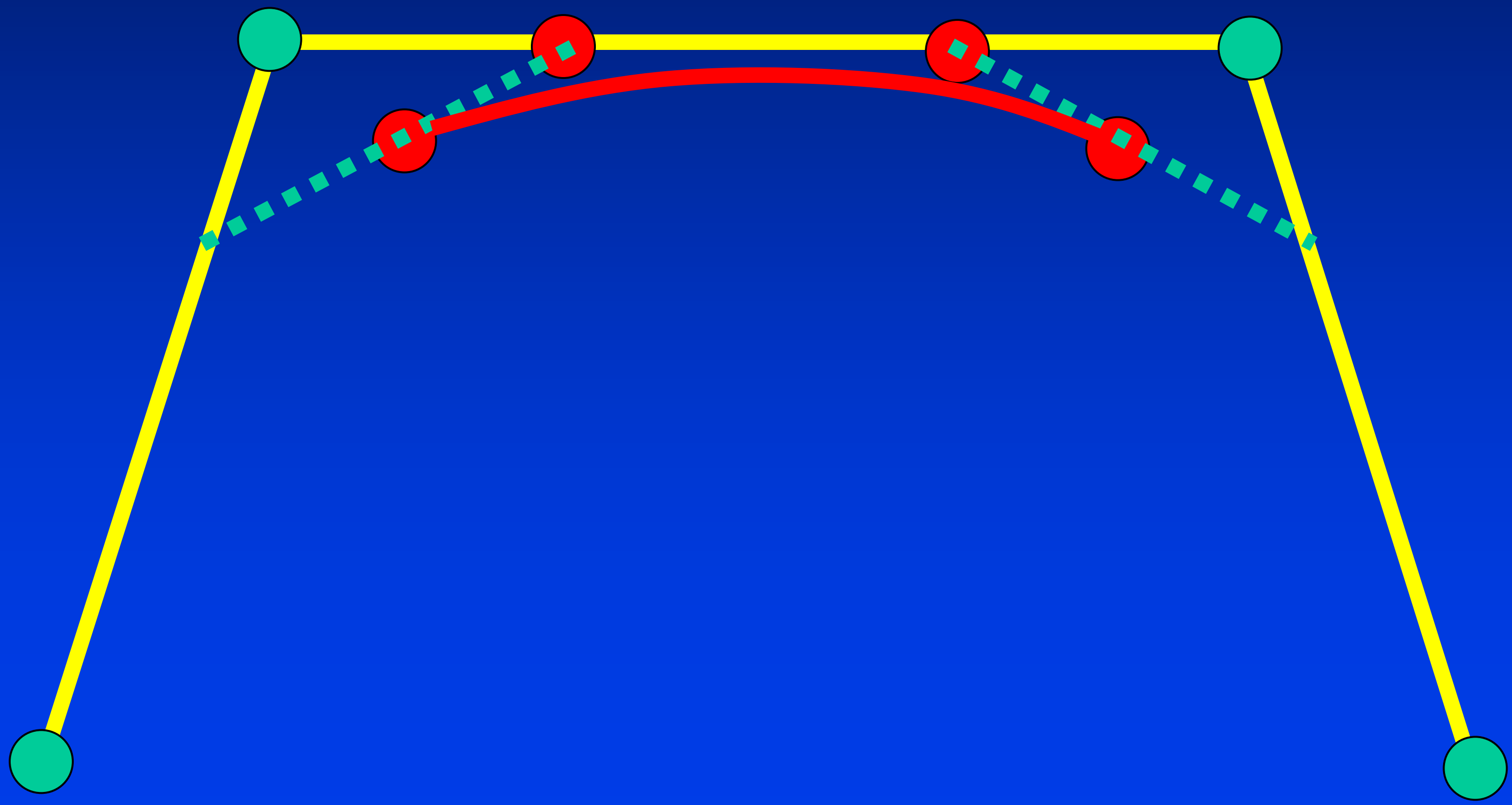
B-Spline Properties

- There are n-k+2 piecewise polynomials
- Each curve span is influenced by k control points
- Each control points at most affects k curve spans
- Local control!!!
- Convex hull
- The degree of B-spline polynomial can be independent from the number of control points
- Compare B-spline with Bezier!!!
- Key components: control points, basis functions, knots, parametric domain, local vs. global control, continuity

B-Spline Properties

- Partition of unity, positivity, and recursive evaluation of basis functions
- Special cases: Bezier splines
- Efficient algorithms and tools
  - Evaluation, knot insertion, degree elevation, derivative, integration, continuity
- Composite Bezier curves for B-splines

Uniform B-Spline



Another Formulation

- Uniform B-spline
- Parameter normalization (u is in [0,1])
- End-point positions and tangents

$$\mathbf{c}(0) = \frac{1}{6}(\mathbf{p}_0 + 4\mathbf{p}_1 + \mathbf{p}_2)$$

$$\mathbf{c}(1) = \frac{1}{6}(\mathbf{p}_1 + 4\mathbf{p}_2 + \mathbf{p}_3)$$

$$\mathbf{c}'(0) = \frac{1}{2}(\mathbf{p}_2 - \mathbf{p}_0)$$

$$\mathbf{c}'(1) = \frac{1}{2}(\mathbf{p}_3 - \mathbf{p}_1)$$

Another Formulation

- Matrix representation

$$\mathbf{c}(u) = UM_h \begin{bmatrix} \mathbf{c}(0) \\ \mathbf{c}(1) \\ \mathbf{c}'(0) \\ \mathbf{c}'(1) \end{bmatrix} = UM_h M' \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = UM\mathbf{p}$$

- Basis matrix

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

Basis Functions

- Note that, u is now in [0,1]

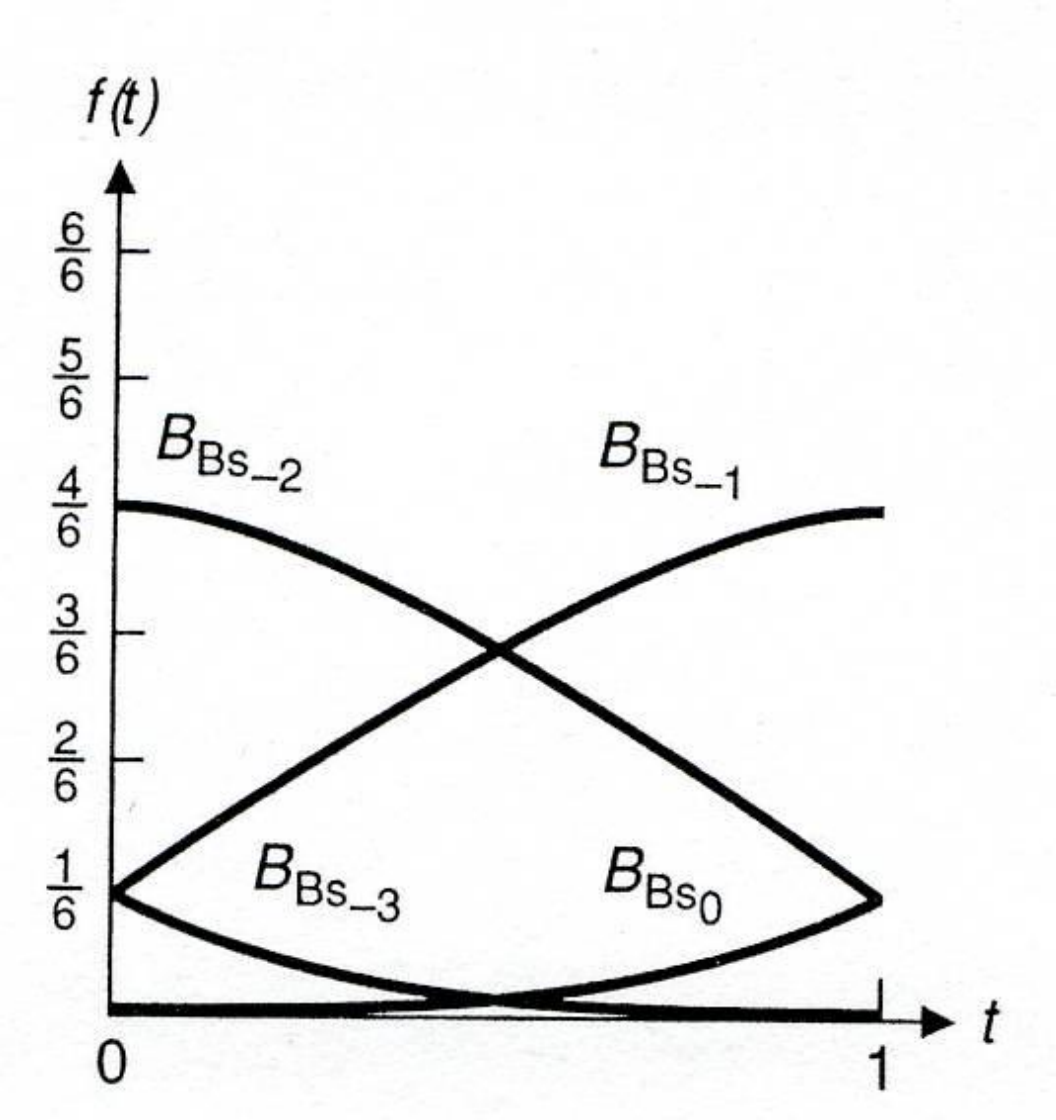
$$B_{0,4}(u) = \frac{1}{6}(1-u)^3$$

$$B_{1,4}(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

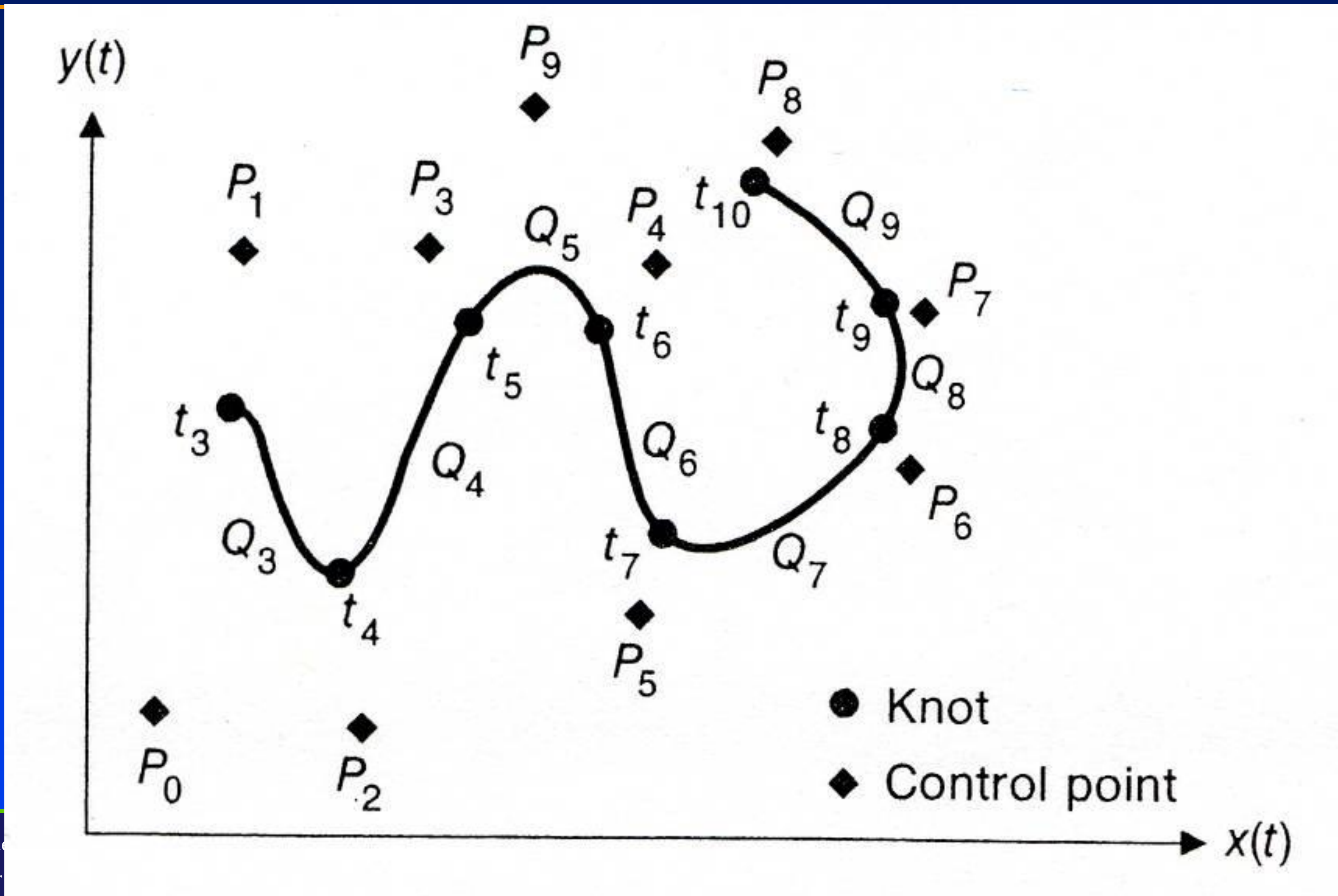
$$B_{2,4}(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$B_{3,4}(u) = \frac{1}{6}(u)^3$$

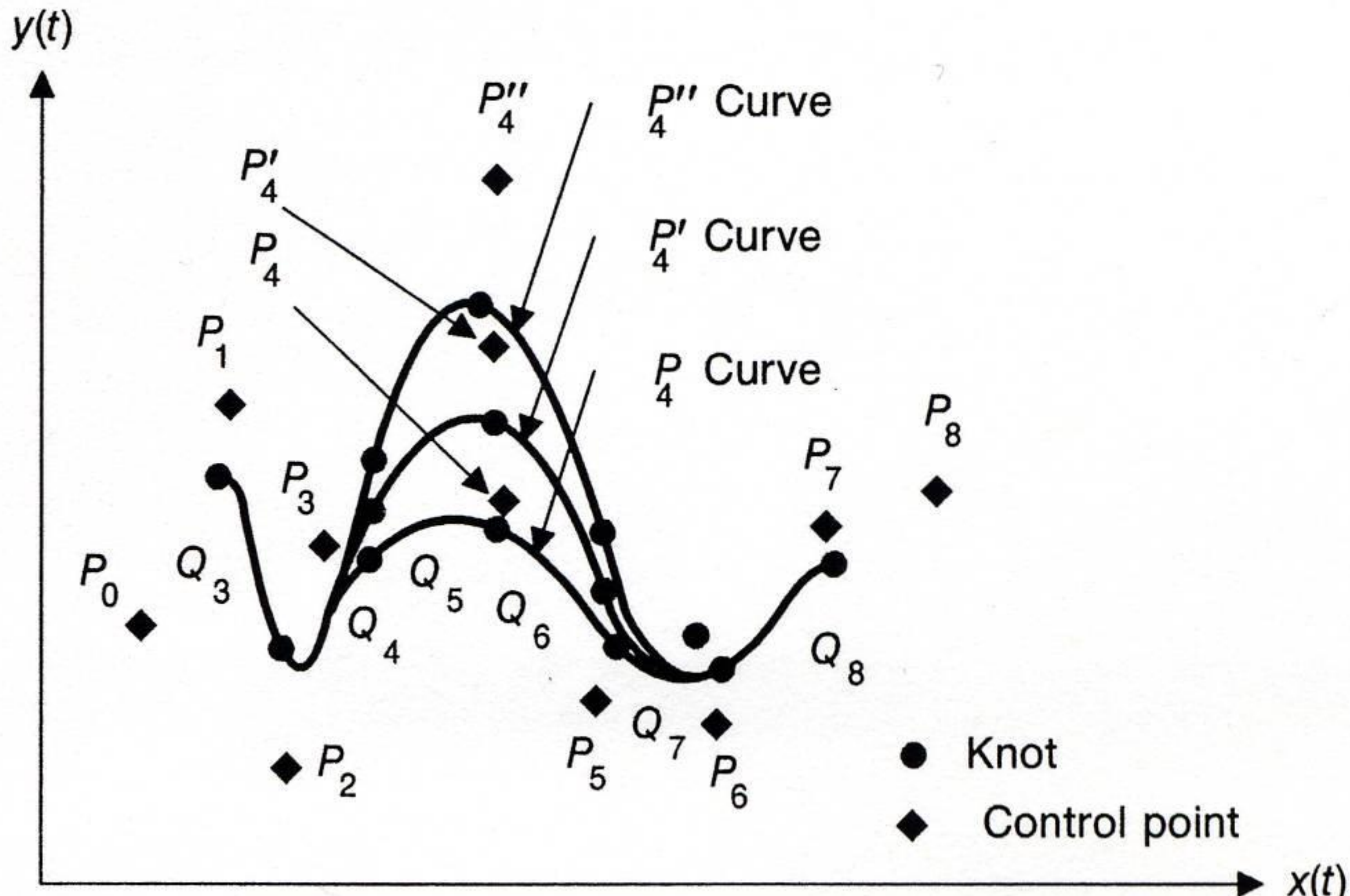
B-Spline Basis Functions



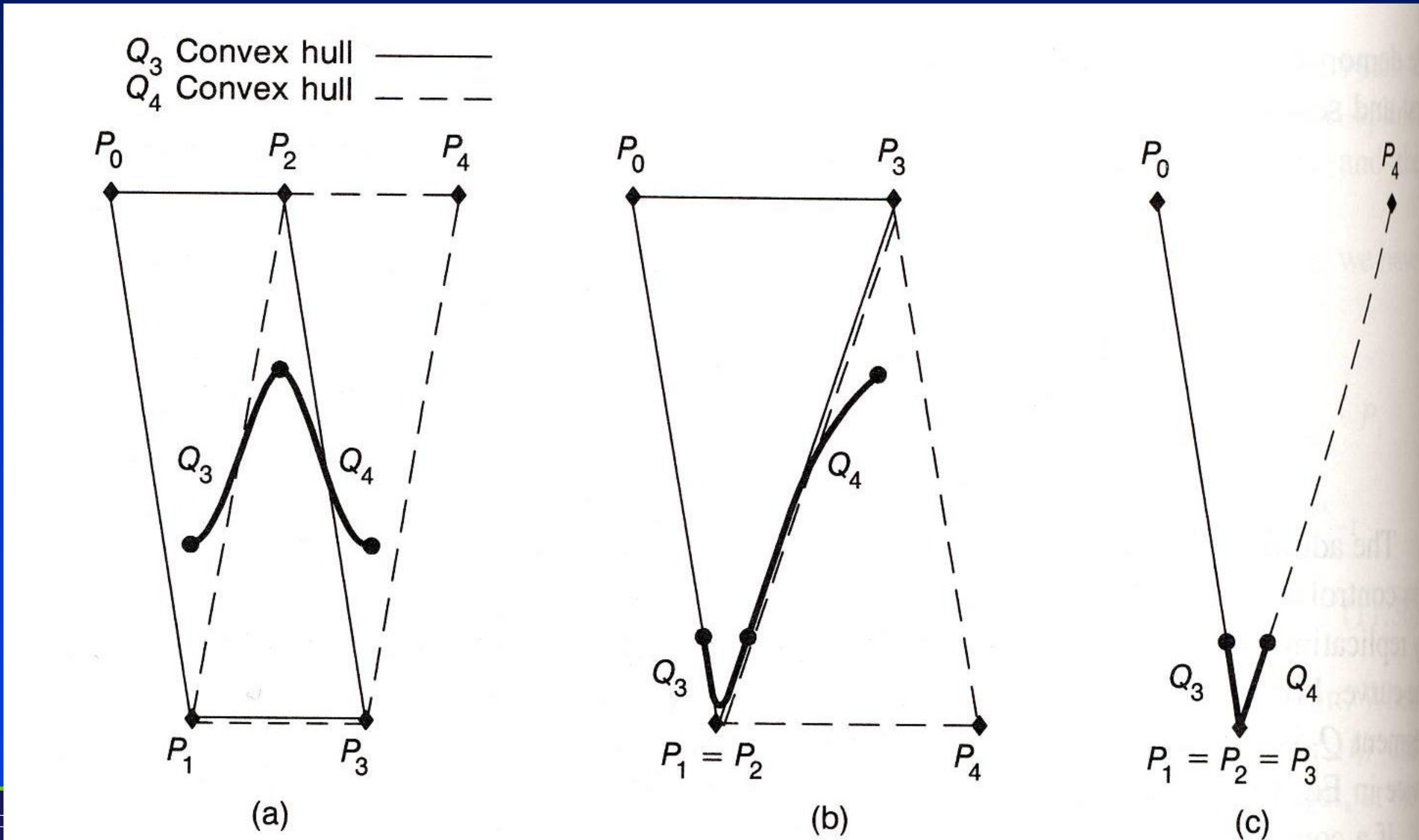
Uniform Non-rational B-Splines



Uniform Non-rational B-Splines



Uniform Non-rational B-Splines
multiple control points



B-Spline Rendering

- Transform it to a set of Bezier curves
- Convert the I-th span into a Bezier representation

$$\mathbf{p}_i, \mathbf{p}_{i+1}, \cdots\cdots, \mathbf{p}_{i+k-1}$$
$$\mathbf{v}_0, \mathbf{v}_1, \cdots\cdots, \mathbf{v}_{k-1}$$

- Consider the entire B-spline curve

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \cdots\cdots, \mathbf{p}_n$$
$$\mathbf{v}_0, \cdots\cdots, \mathbf{v}_3, \mathbf{v}_4, \cdots\cdots, \mathbf{v}_7, \cdots\cdots, \mathbf{v}_{4(n-3)}, \cdots\cdots, \mathbf{v}_{4(n-3)+3}$$

Matrix Expression

$$\begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{4(n-3)+3} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_n \end{bmatrix}$$

- The matrix structure and components of B?

$$\mathbf{q} = \mathbf{A}\mathbf{v} = \mathbf{A}\mathbf{B}\mathbf{p}$$

- The matrix structure and components of A?

B-Spline Discretization

- Parametric domain: [u(k-1),u(n+1)]
- There are n+2-k curve spans (pieces)
- Assuming m+1 points per span (uniform sampling)
- Total sampling points m(n+2-k)+1=l
- B-spline discretization with corresponding parametric values:

$$\mathbf{q}_0,\ldots\ldots,\mathbf{q}_{l-1}$$

$$\mathbf{v}_0,\ldots\ldots,\mathbf{v}_{l-1}$$

$$\mathbf{q}_i = \mathbf{c}(v_i) = \sum_{j=0}^{n} \mathbf{p}_j B_{j,k}(v_i)$$

B-Spline Discretization

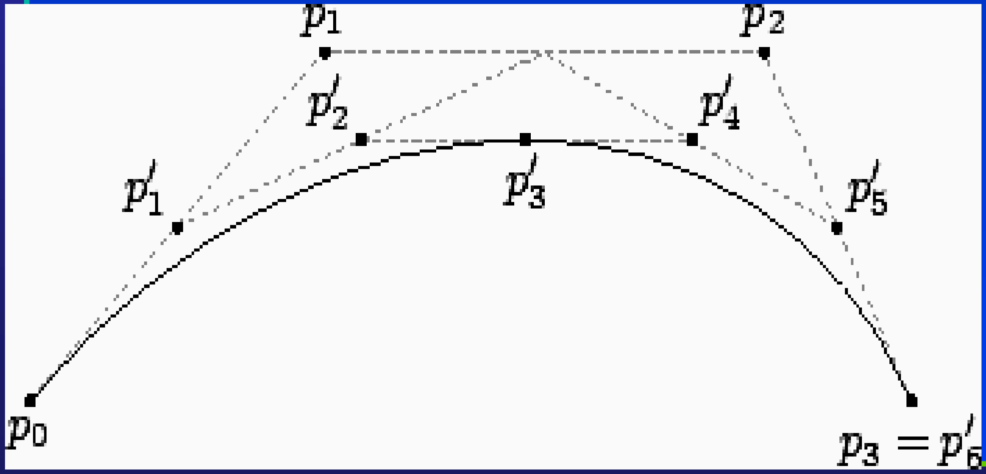
- Matrix equation

$$\begin{bmatrix} \mathbf{q}_0 \\ \vdots \\ \mathbf{q}_{l-1} \end{bmatrix} = \begin{bmatrix} B_{0,k}(v_0) & \cdots & B_{n,k}(v_0) \\ \vdots & \ddots & \vdots \\ B_{0,k}(v_{l-1}) & \cdots & B_{n,k}(v_{l-1}) \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_n \end{bmatrix}$$

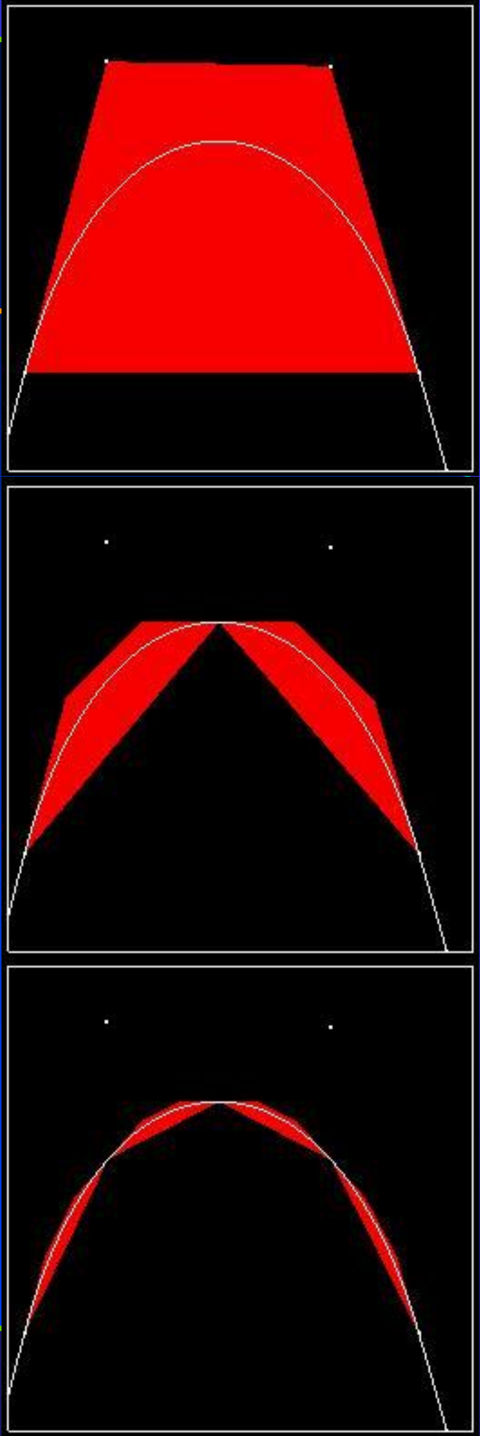
- A is (l)x(n+1) matrix, in general (l) is much larger than (n+1), so A is sparse
- The linear discretization for both modeling and rendering

Displaying Bezier Spline

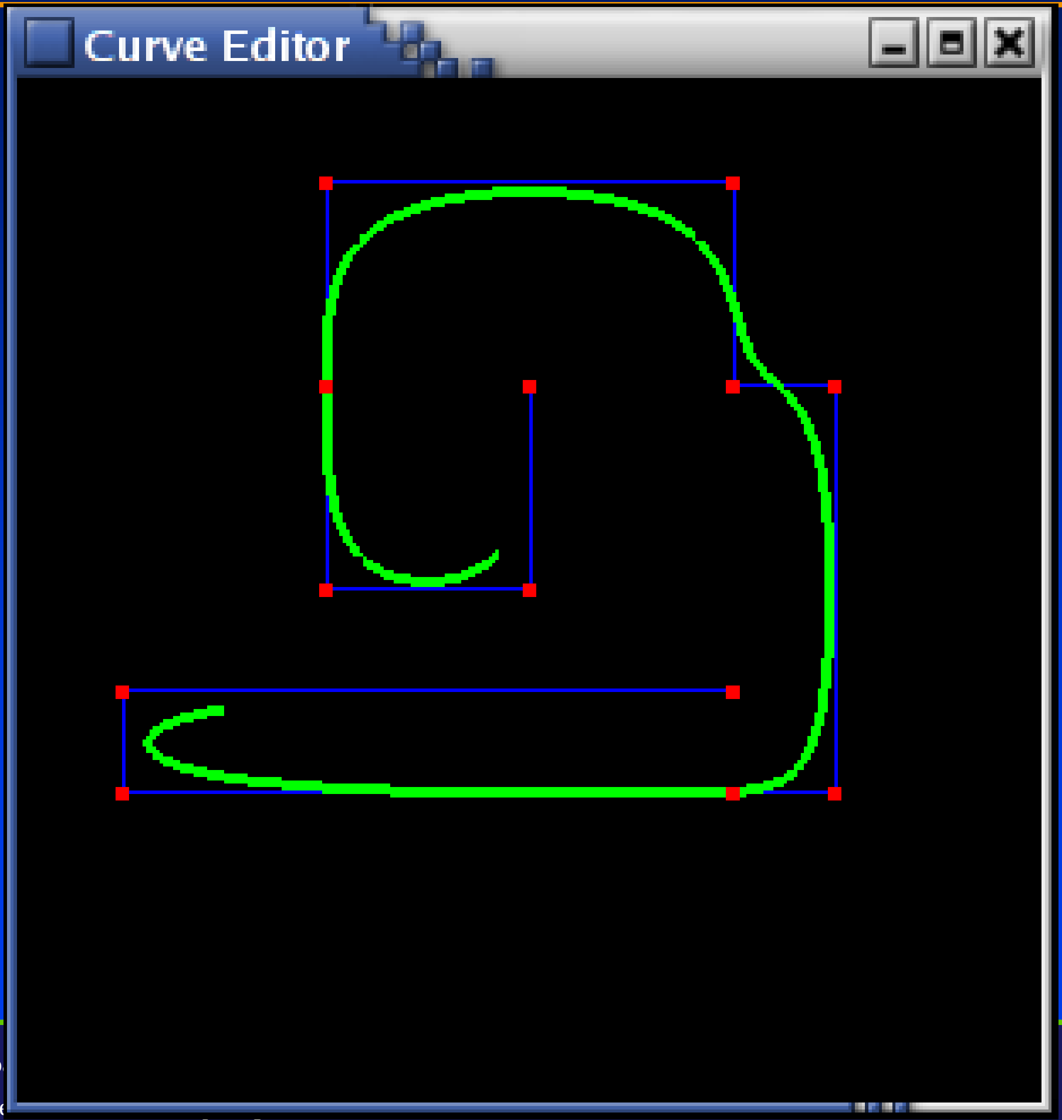
- A Bezier curve with 4 control points:
  - $P_0 \quad P_1 \quad P_2 \quad P_3$
- Can be split into 2 new Bezier curves:
  - $P_0 \quad P'_1 \quad P'_2 \quad P'_3$
  - $P'_3 \quad P'_4 \quad P'_5 \quad P_3$



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



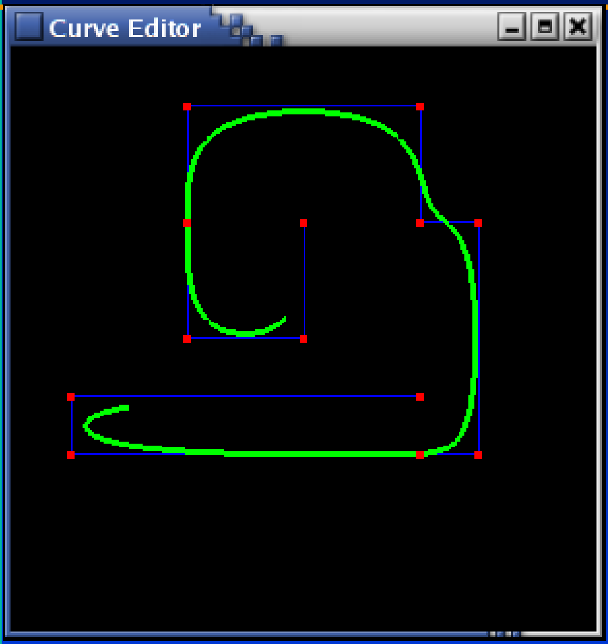
Connecting Cubic B-Spline Curves



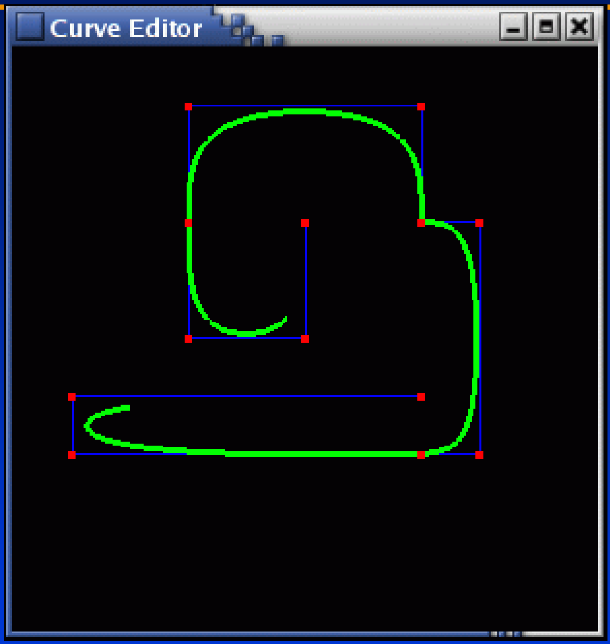
- What's the relationship between
  - the # of control points, and
  - the # of cubic BSpline subcurves?

STONY BROOK
STATE UNIVERSITY OF NEW YORK

B-Spline Curve Control Points

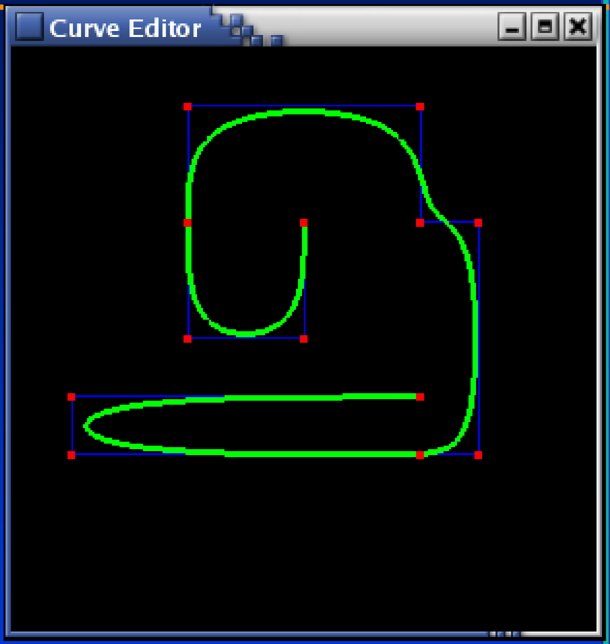


Default BSpline



BSpline with Discontinuity

Repeat interior control point



BSpline which passes through end points

Repeat end points

From B-Splines to NURBS

- What are NURBS???
- Non Uniform Rational B-Splines (NURBS)
- Rational curve motivation
- Polynomial-based splines can not represent commonly-used analytic shapes such as conic sections (e.g., circles, ellipses, parabolas)
- Rational splines can achieve this goal
- NURBS are a unified representation
  - Polynomial, conic section, etc.
  - Industry standard

NURBS (as Generalized B-Splines)

- B-Spline: uniform cubic B-Spline
- NURBS: Non-Uniform Rational B-Spline
  - non-uniform = different spacing between the blending functions, a.k.a. knots
  - rational = ratio of polynomials (instead of cubic)

From B-Splines to NURBS

- B-splines

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

- NURBS (curve)

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

NURBS

- NURBS mathematics:

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

- Geometric Meaning--- obtained from projection!
- B-splines in homogenous representation

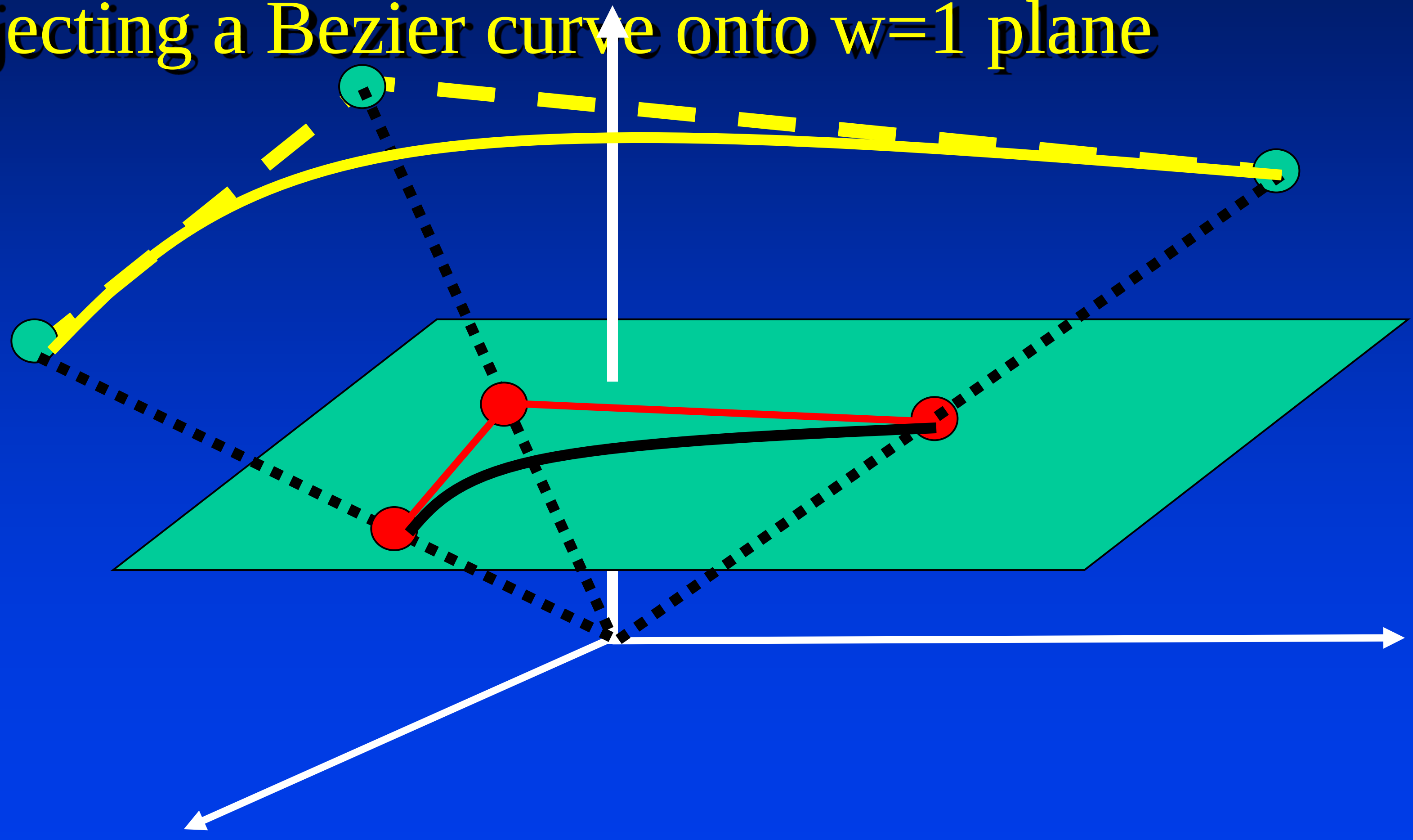
$$\mathbf{c}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \\ w(u) \end{bmatrix} = \sum_{i=0}^{n} \begin{bmatrix} \mathbf{p}_{i,x} w_i \\ \mathbf{p}_{i,y} w_i \\ \mathbf{p}_{i,z} w_i \\ w_i \end{bmatrix} B_{i,k}(u) = \sum_{i=0}^{n} \begin{bmatrix} \mathbf{p}_i w_i \\ \mathbf{w}_i \end{bmatrix} B_{i,k}(u)$$

Geometric NURBS

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

Rational Bezier Curve

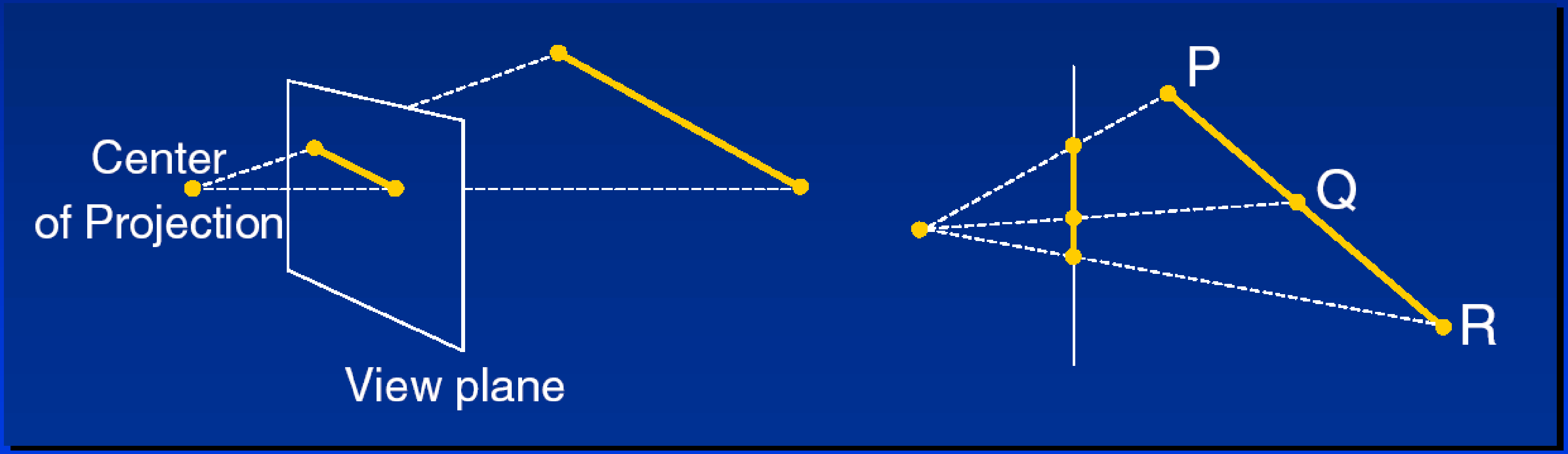
- Projecting a Bezier curve onto w=1 plane



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)

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

$$Linear:$$

$$(1-u)$$

$$(u)$$

$$Quadratic:$$

$$(1-u)^2$$

$$2(1-u)u$$

$$(u)^2$$

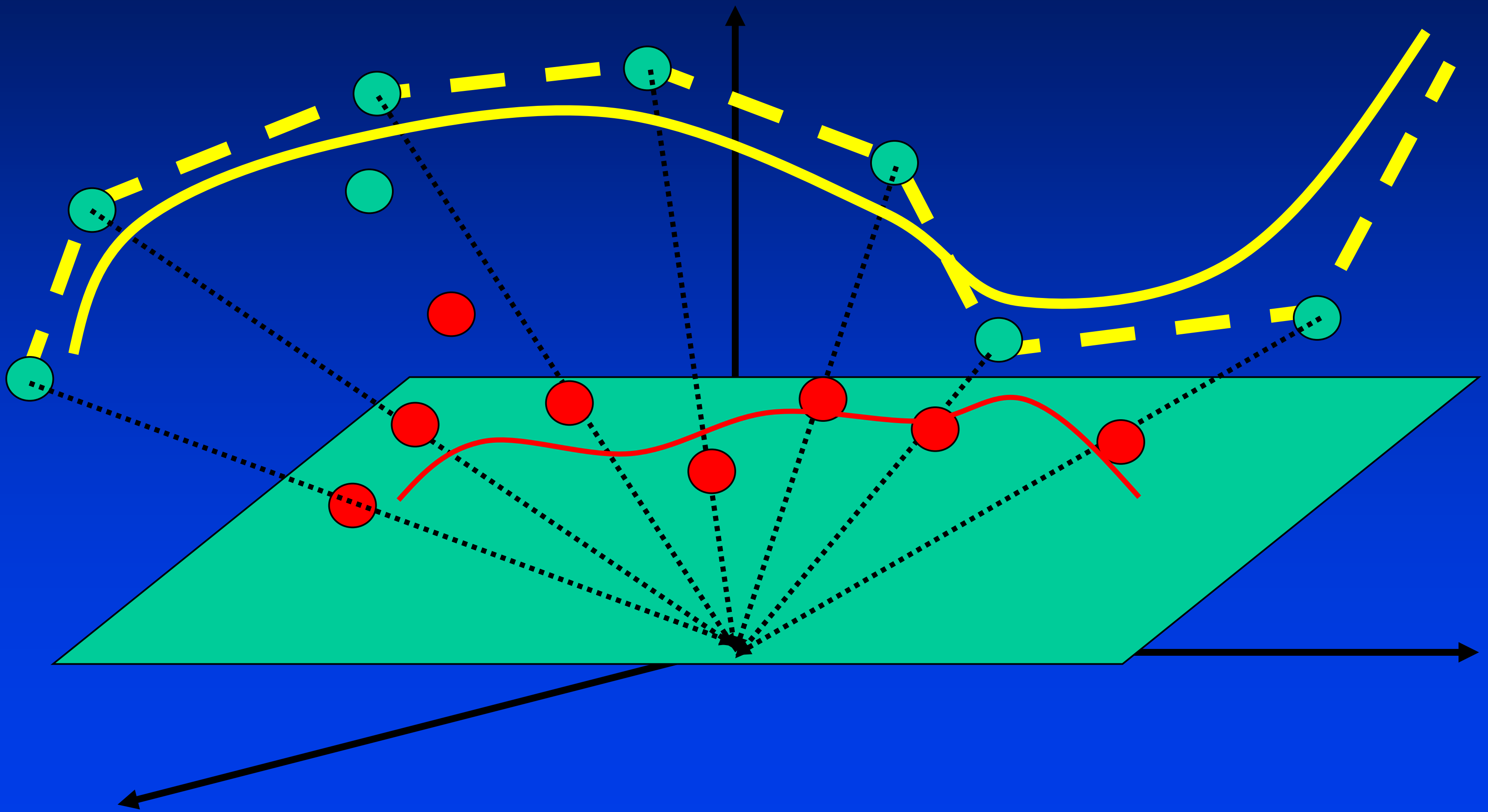
Consider Quadratic Case

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

or

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

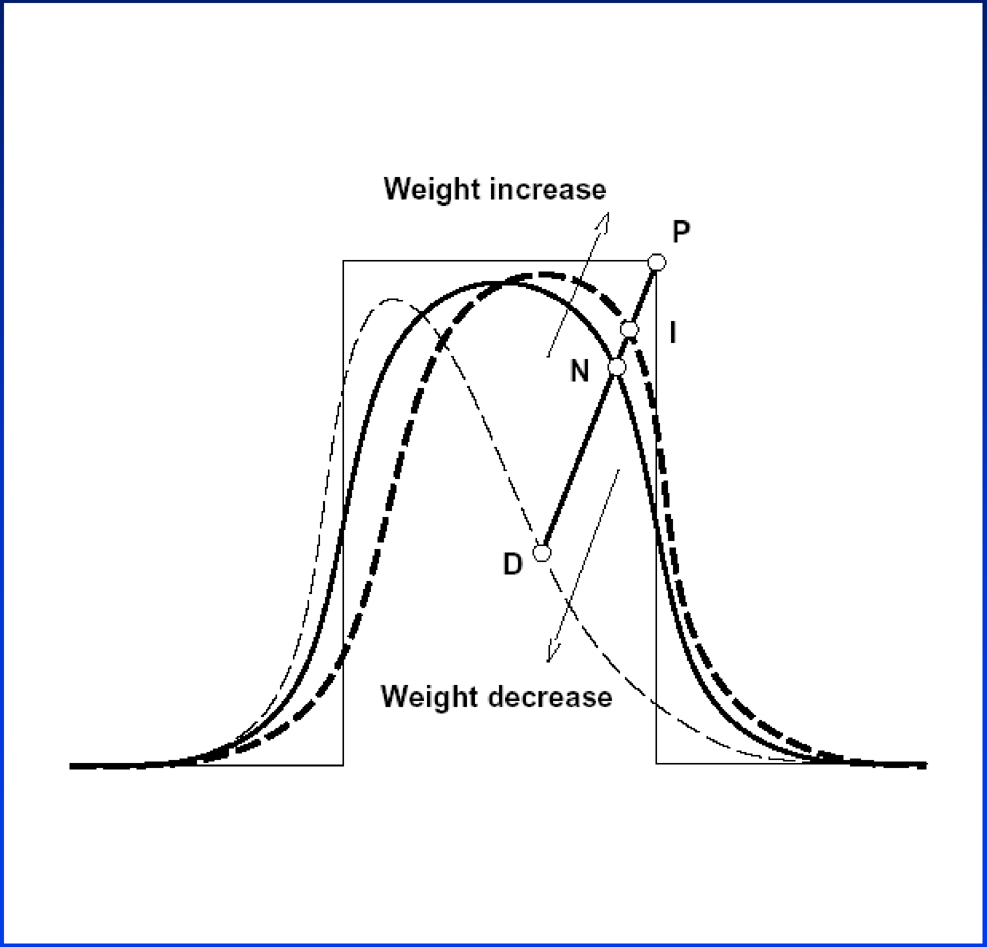
From B-Splines to NURBS



NURBS Weights

- Weight increase “attracts” the curve towards the associated control point
- Weight decrease “pushes away” the curve from the associated control point

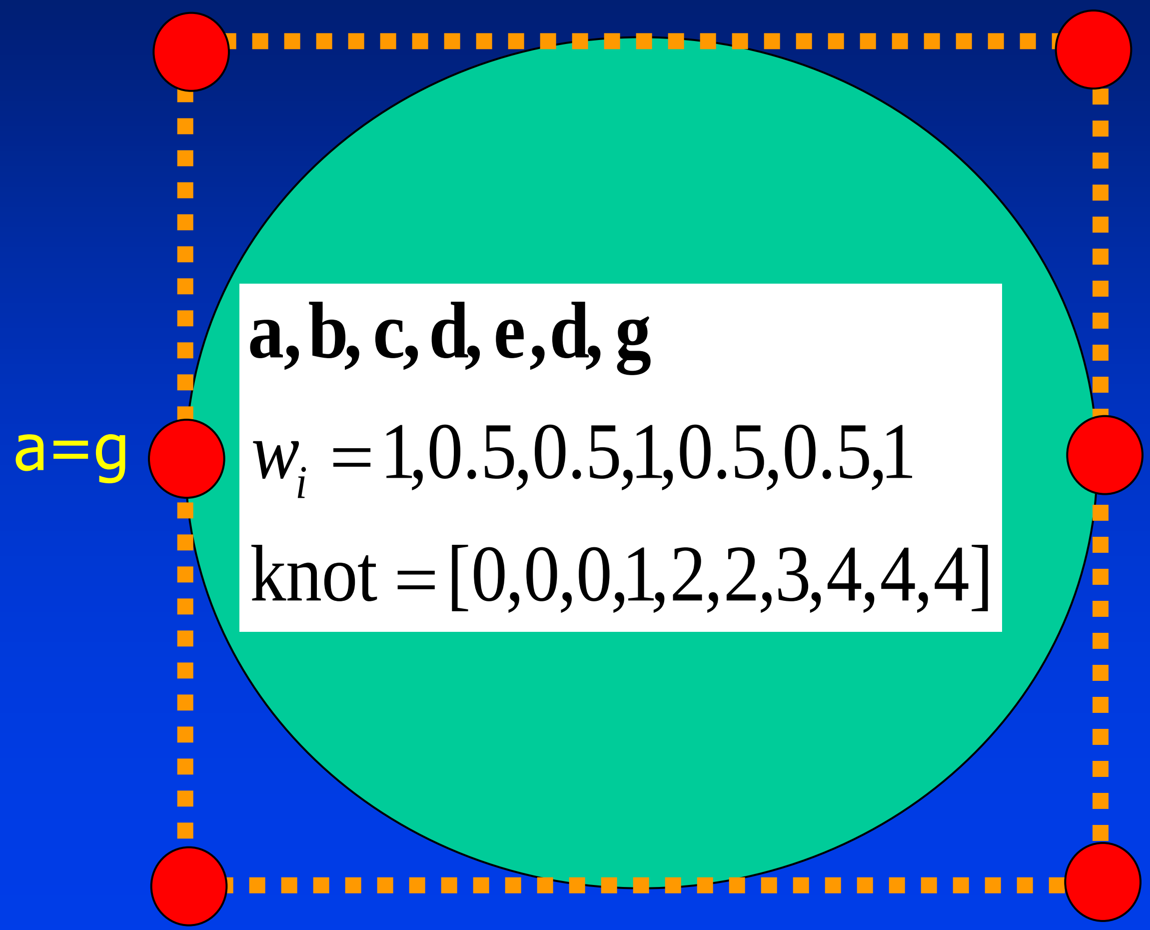
NURBS



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

NURBS Properties

- Generalization of B-splines and Bezier splines
- Unified formulation for free-form and analytic shape
- Weights as extra DOFs
- Various smoothness requirements
- Powerful geometric toolkits
- Efficient and fast evaluation algorithm
- Invariance under standard transformations
- Composite curves
- Continuity conditions

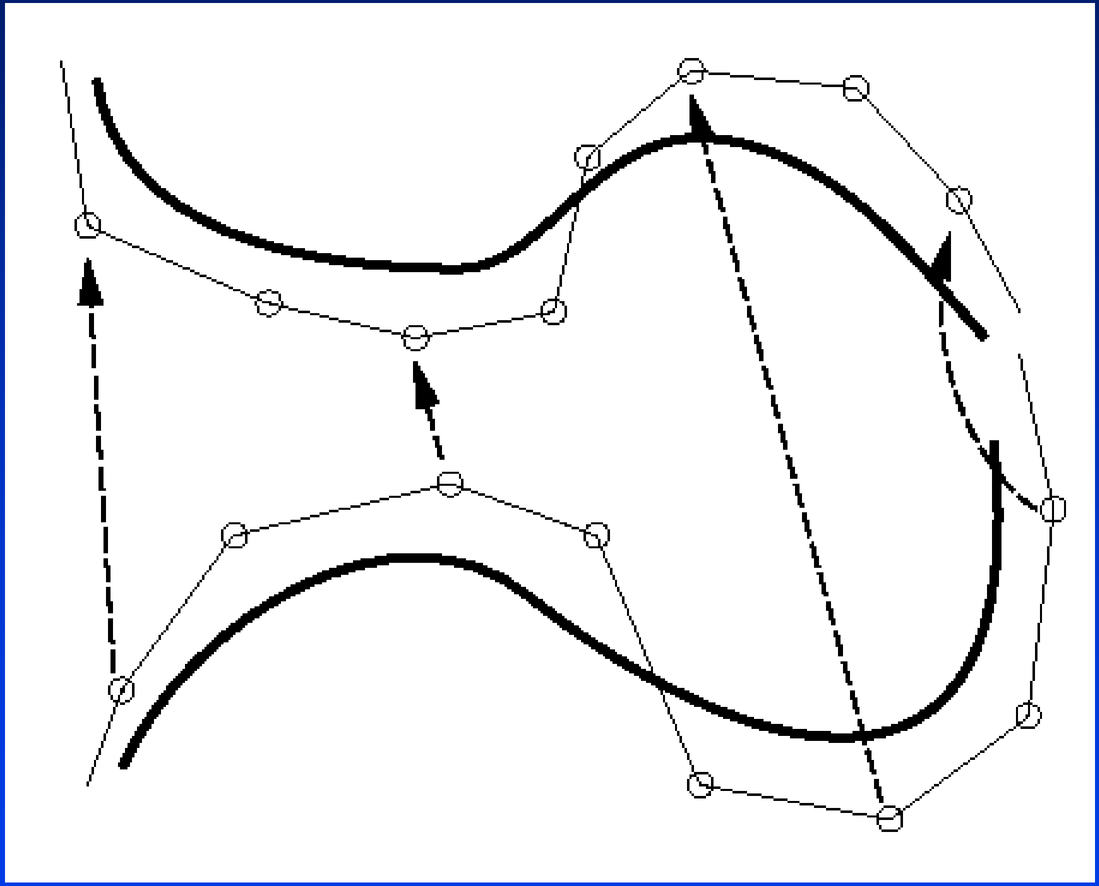
Properties of NURBS

- Represent standard shapes.
- Invariant under perspective projection.
- B-Spline is a special case.
- Weights as extra degrees of freedom.
- Can represent analytic shapes such as circles.

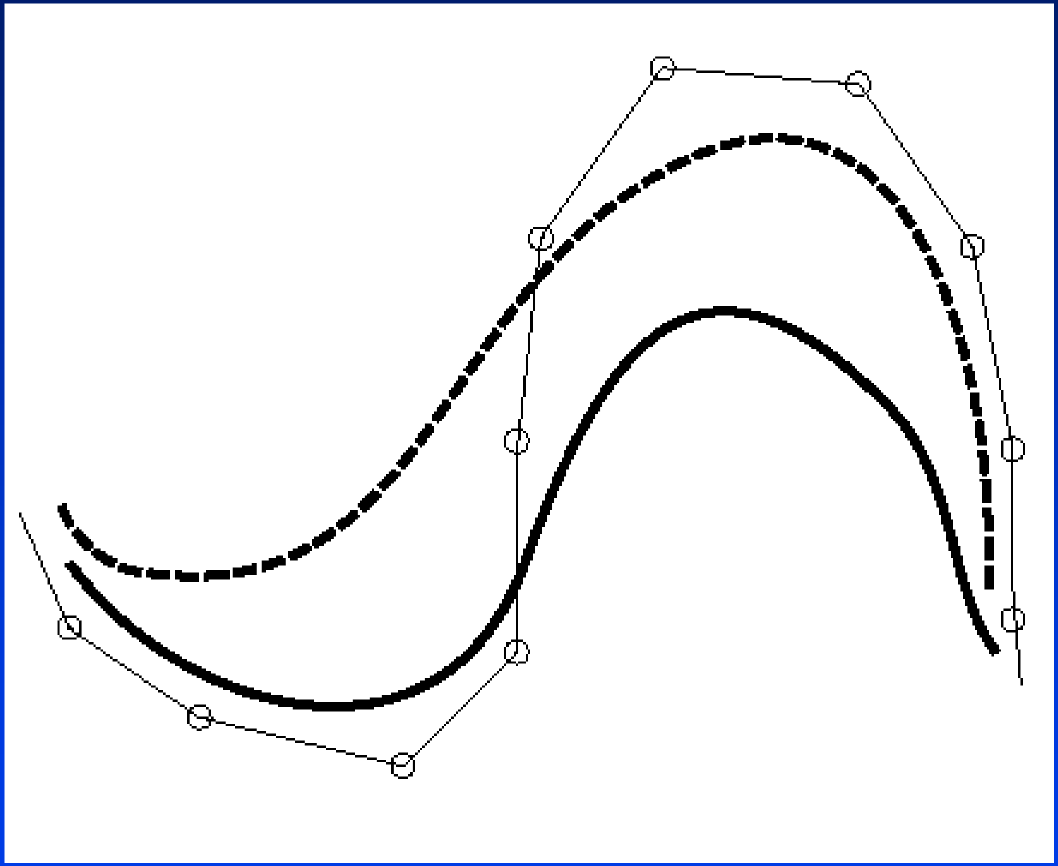
Geometric Modeling Techniques

- Control Point Manipulation.
- Weight Modification.
- Knot Vector Variation.
- Dynamic Modeling

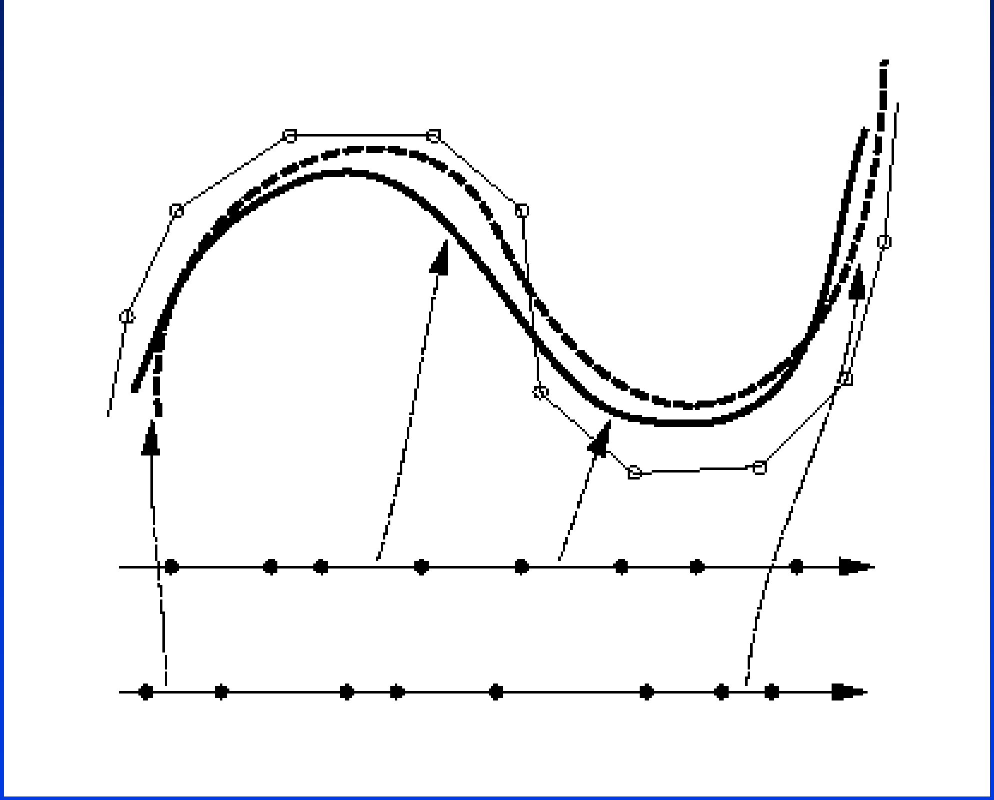
Control Point Manipulation



Weight Modification



Knot Vector Variation



Dynamic Modeling