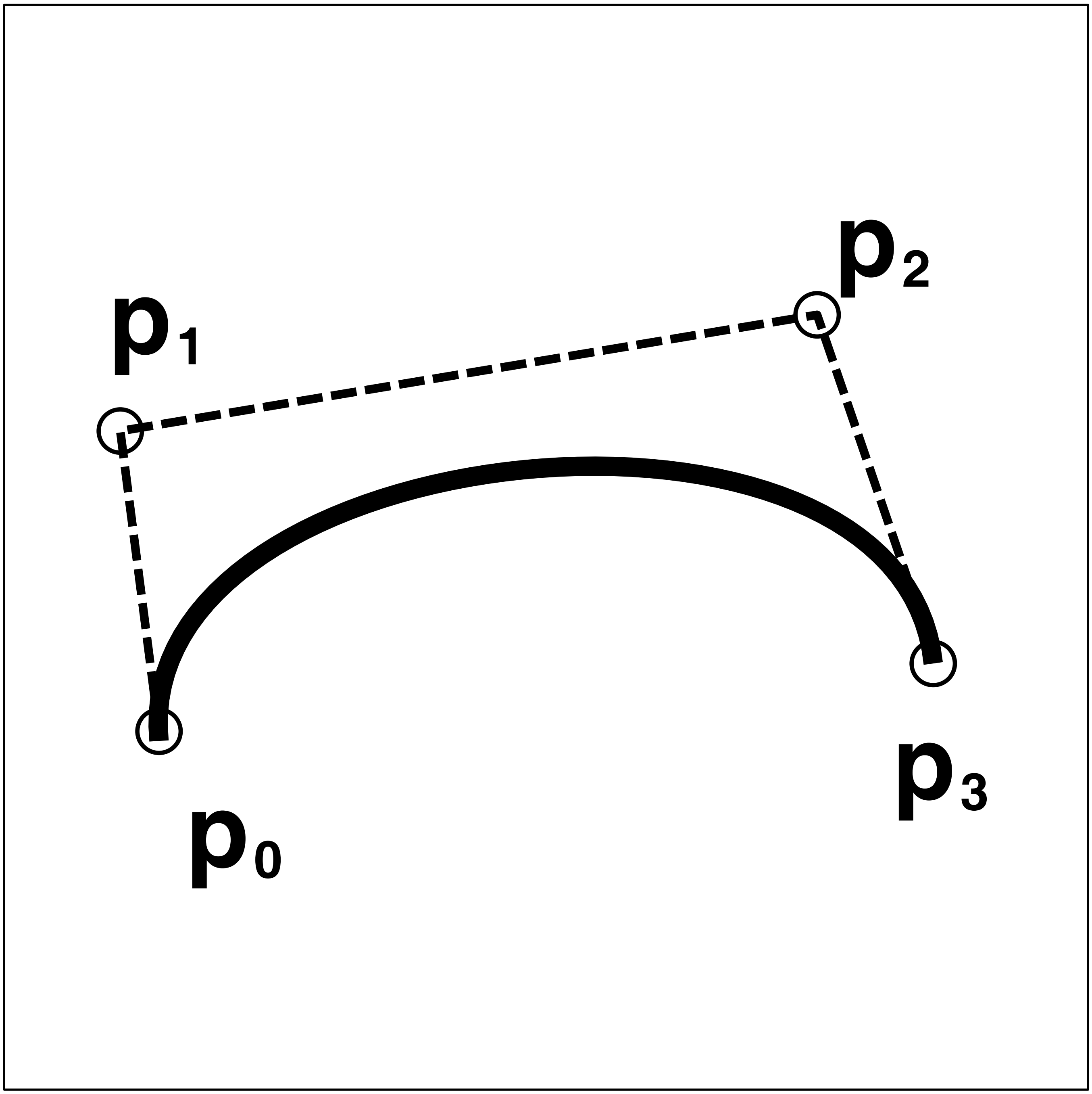
Bezier Curves

Curve Geometry



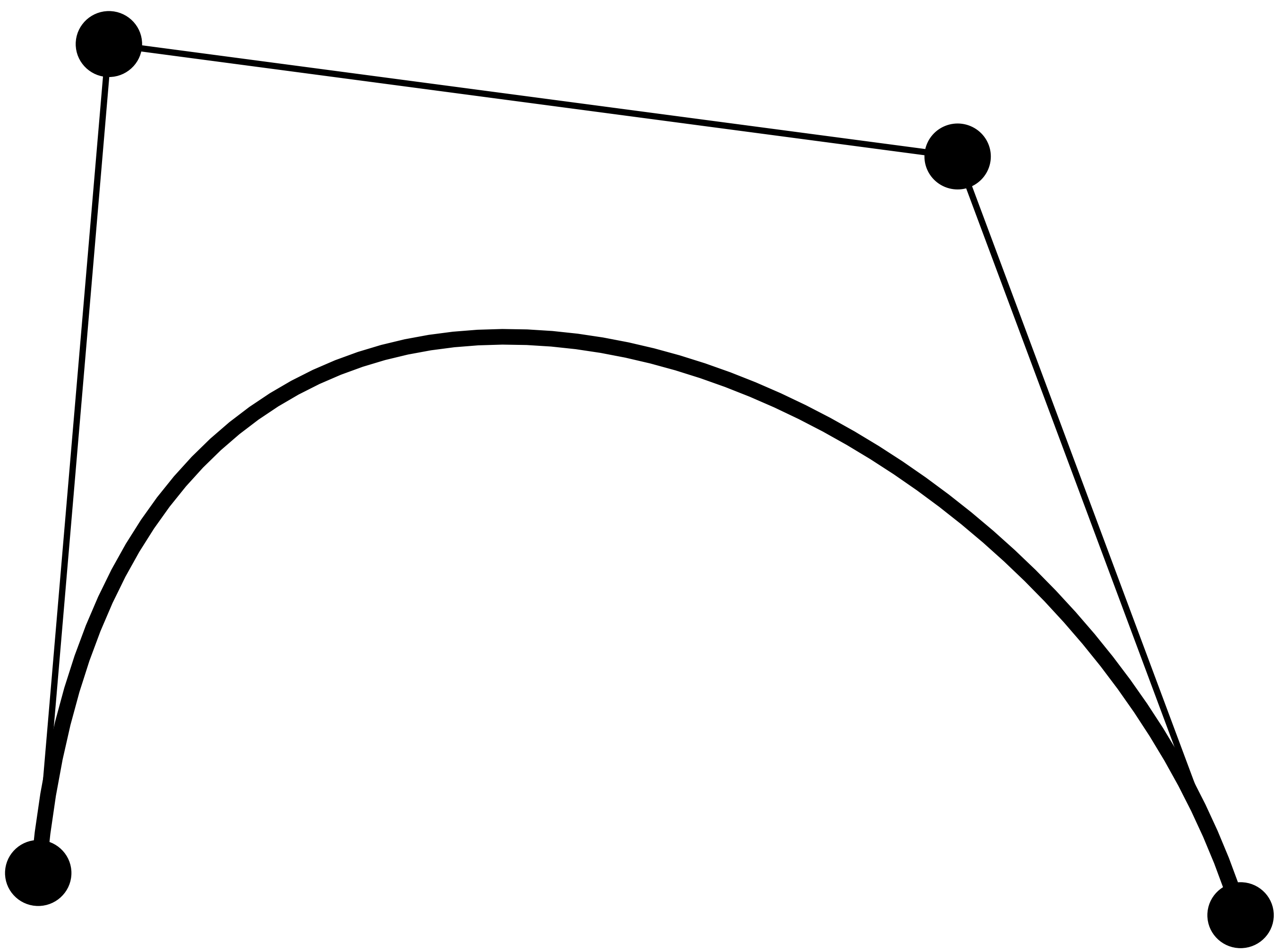
Curve Mathematics (Cubic)

- **Curve:** $\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$
- **Control points:** $\mathbf{p}_i$
- **Basis functions:**
  - $B_0^3(u) = (1-u)^3$
  - $B_1^3(u) = 3u(1-u)^2$
  - $B_2^3(u) = 3u^2(1-u)$
  - $B_3^3(u) = u^3$

Basis Functions (Cubic)

- Refer to P332 of “Computer Graphics” by H&B

Cubic Bezier Curves



Basic Properties (Cubic)

- The curve passes through the first and the last point
  (end-point interpolation)
- Linear combination of control points and basis functio
- Basis functions are all polynomials
- Basis functions sum to one
- All basis functions are non-negative
- Convex hull (both necessary and sufficient)
- Tangent vectors can be easily evaluated at end-point

$$\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{c}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

- Second derivatives at end-points can also be easily
  computed:

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

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

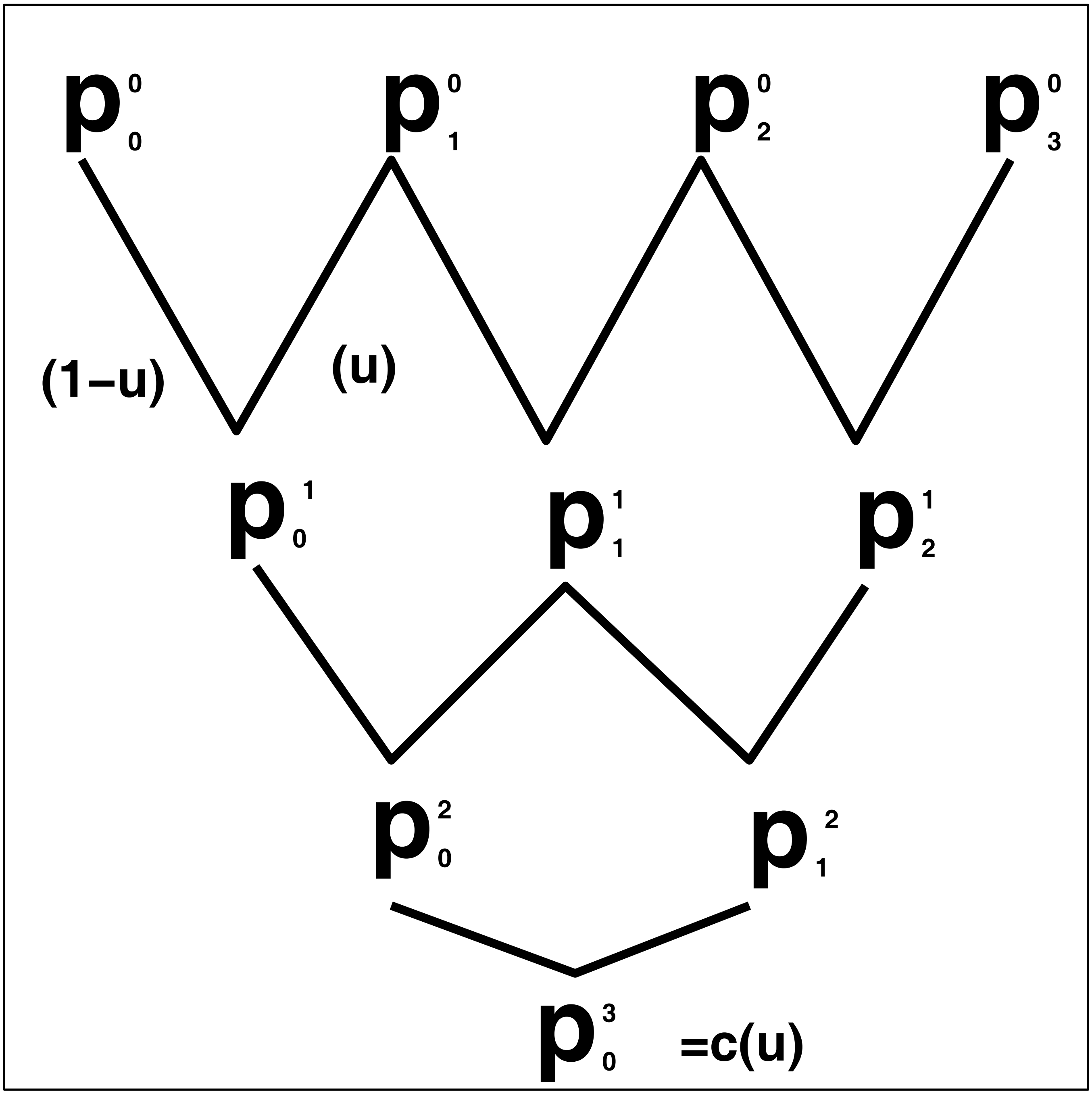
- **Predictability**
- **The derivative of a cubic Bezier curve is a quadratic Bezier curve**

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

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

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

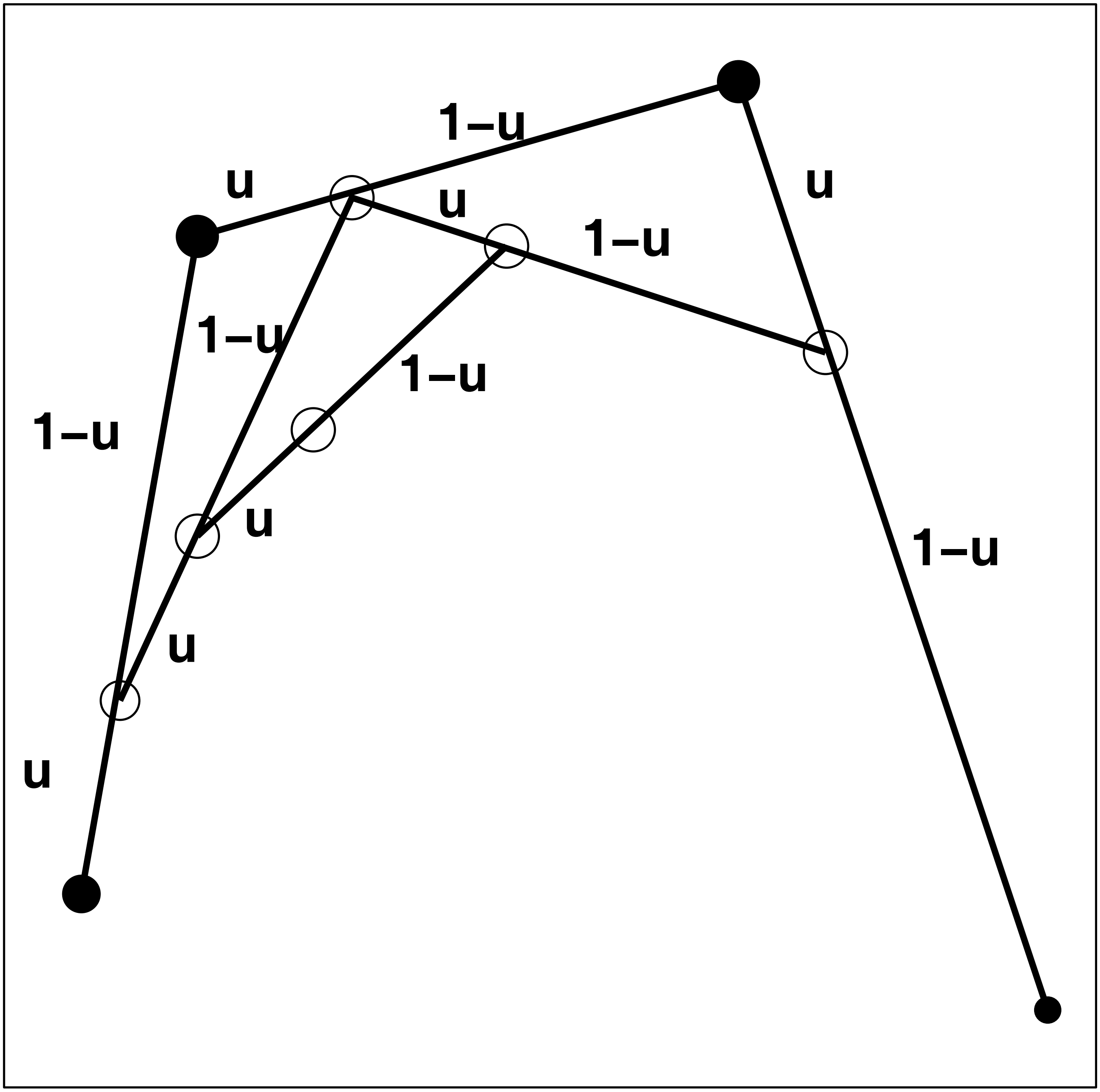
Recursive Evaluation



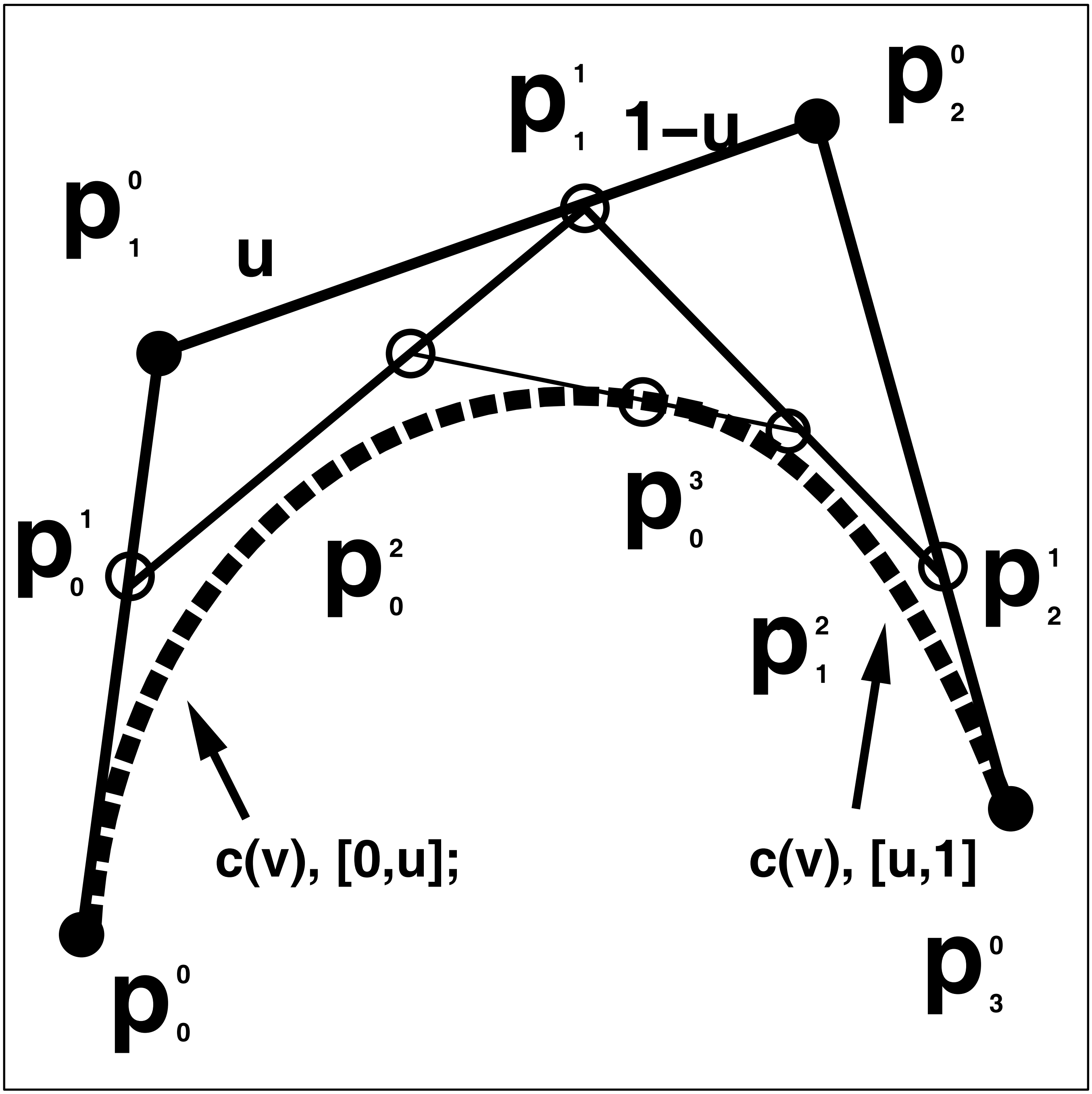
Why Cubic Polynomials

- 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 Algorithm



Recursive Algorithm (Cubic)



More Properties (Cubic)

- **Two curve spans are obtained:**

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

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

- **Reparameterization:**

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

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

- **Both are Bezier curves**
- **The left is defined by** $\mathbf{p}_0^0, \mathbf{p}_0^1, \mathbf{p}_0^2, \mathbf{p}_0^3$
- **The right is defined by** $\mathbf{p}_0^3, \mathbf{p}_1^2, \mathbf{p}_2^1, \mathbf{p}_3^0$

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!!!

Bezier Curves (Splines)

- **Bezier curves of degree n**
- **Curve**

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

- **Control points: $\mathbf{p}_i$**
- **Basis functions**
- $B_i^n(u)$ **are Bernstein polynomials of degree n**

$$B_i^n(u) = \left(\begin{array}{c} n \\ i \end{array} \right) u^i (1-u)^{n-i}$$

$$\left(\begin{array}{c} n \\ i \end{array} \right) = \frac{n!}{(n-1)!i!}$$

- **More control points**
- **What do these basis functions look like?**

Properties

- **Basis functions are non-negative**
- **The summation of all basis functions is unity**
- **End point interpolation**

$$\mathbf{c}(0) = \mathbf{p}_0$$

$$\mathbf{c}(1) = \mathbf{p}_n$$

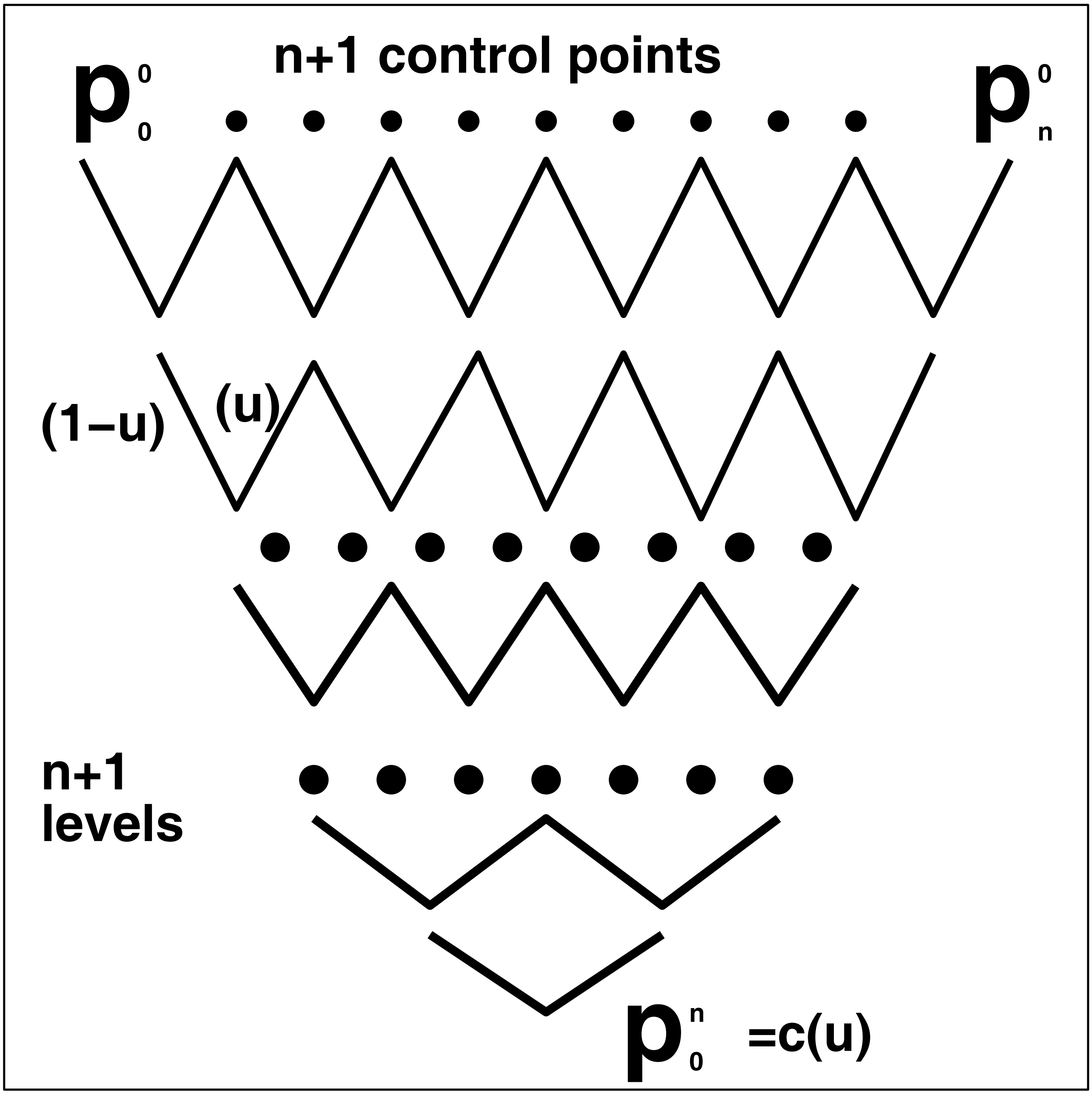
- **Binomial expansion theorem**

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

- **Convex hull: the curve is bounded by the convex hull defined by control points**
- **Recursive subdivision and evaluation**
- **Symmetry: $\mathbf{c}(u)$ is defined by $\mathbf{p}_0, \ldots, \mathbf{p}_n$; $\mathbf{c}(1-u)$ is defined by $\mathbf{p}_n, \ldots, \mathbf{p}_0$.**

- **Efficient evaluation algorithm**
- **Differentiation & integration**
- **Degree elevation**
  - **use polynomial of degree $n+1$ to express that of degree n**
- **Composite curves**
- **Geometric continuity**
- **Display of curve**

Recursive Computation



Recursive Computation

- $\mathbf{p}_i^0 = \mathbf{p}_i$, $i = 0, \ldots, n$
- $\mathbf{p}_i^j = (1-u)\mathbf{p}_i^{j-1} + u\mathbf{p}_{i+1}^{j-1}$
- $\mathbf{c}(u) = \mathbf{p}_0^n(u)$

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:**
  $\mathbf{c}^{(i)}(0)$ **depends only on** $\mathbf{p}_0, \ldots, \mathbf{p}_i$
  $\mathbf{c}^{(i)}(1)$ **depends only on** $\mathbf{p}_n, \ldots, \mathbf{p}_{n-i}$

- **Derivatives at non-end-points:**
  $\mathbf{c}^{(i)}(u)$ **involve all control points**

Bezier Curve Rendering

- Use its control polygon to approximate the curve
- Recursive subdivision till the tolerance is satisfied
- Display $\mathbf{c}(u)$ defined by $\mathbf{p}_0, \ldots, \mathbf{p}_n$
  - if $\{\mathbf{p}_0, \ldots, \mathbf{p}_n\}$ is flat (with tolerance) then output line segments $\mathbf{p}_0, \ldots, \mathbf{p}_n$
  - else subdivide the curve at $u = 0.5$
  - let $\mathbf{l}_0, \ldots, \mathbf{l}_n$ define the first half
  - let $\mathbf{r}_0, \ldots, \mathbf{r}_n$ define the second half
  - display $(\mathbf{c}_l(u))$
  - display $(\mathbf{c}_r(u))$

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 the curve**
- **Advantages — flexible, low-degree**
- **Disadvantages — how to ensure smoothness at the joints (continuity)**