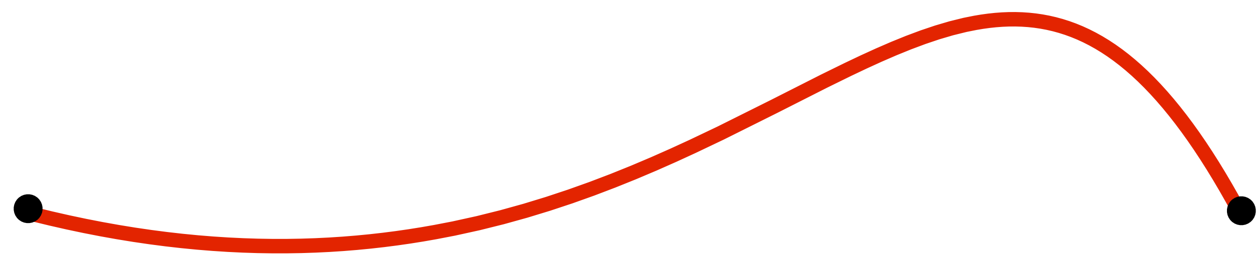
Piecewise Polynomial Curves

Cubics



$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3$$

- Allow up to C^2 continuity at knots
- need 4 control points
  - may be 4 points on the curve, combination of points and derivatives, ...
- good smoothness and computational properties

Advantages of Cubics

- allow for C2 continuity (C1 often not enough, more than C2 unnecessary)
- n piecewise cubics for n+3 points give minimum curvature curve
- symmetry: position and derivatives can be specified at beginning and end
- good tradeoff between numerical issues and smoothness

We can get any 3 of 4 properties

1. piecewise cubic
2. curve interpolates control points
3. curve has local control
4. curves has C2 continuity at knots

Cubics

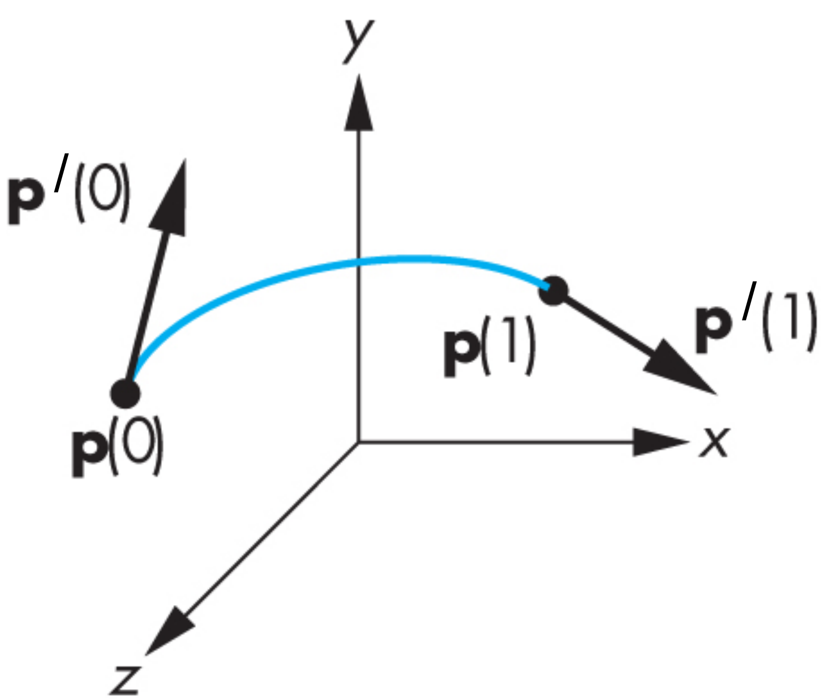
- Natural cubics
  - C2 continuity
  - n points -> n-1 cubic segments
- control is non-local :(
- ill-conditioned x(
- (properties 1, 2, 4)

Cubic Hermite Curves

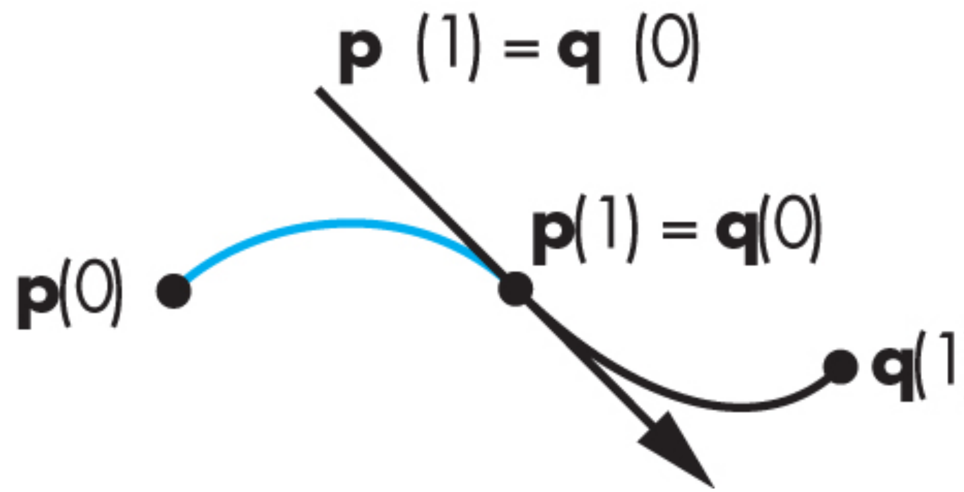
- C1 continuity
- specify both positions and derivatives
- (properties 1, 2, 3)

Cubic Hermite Curves

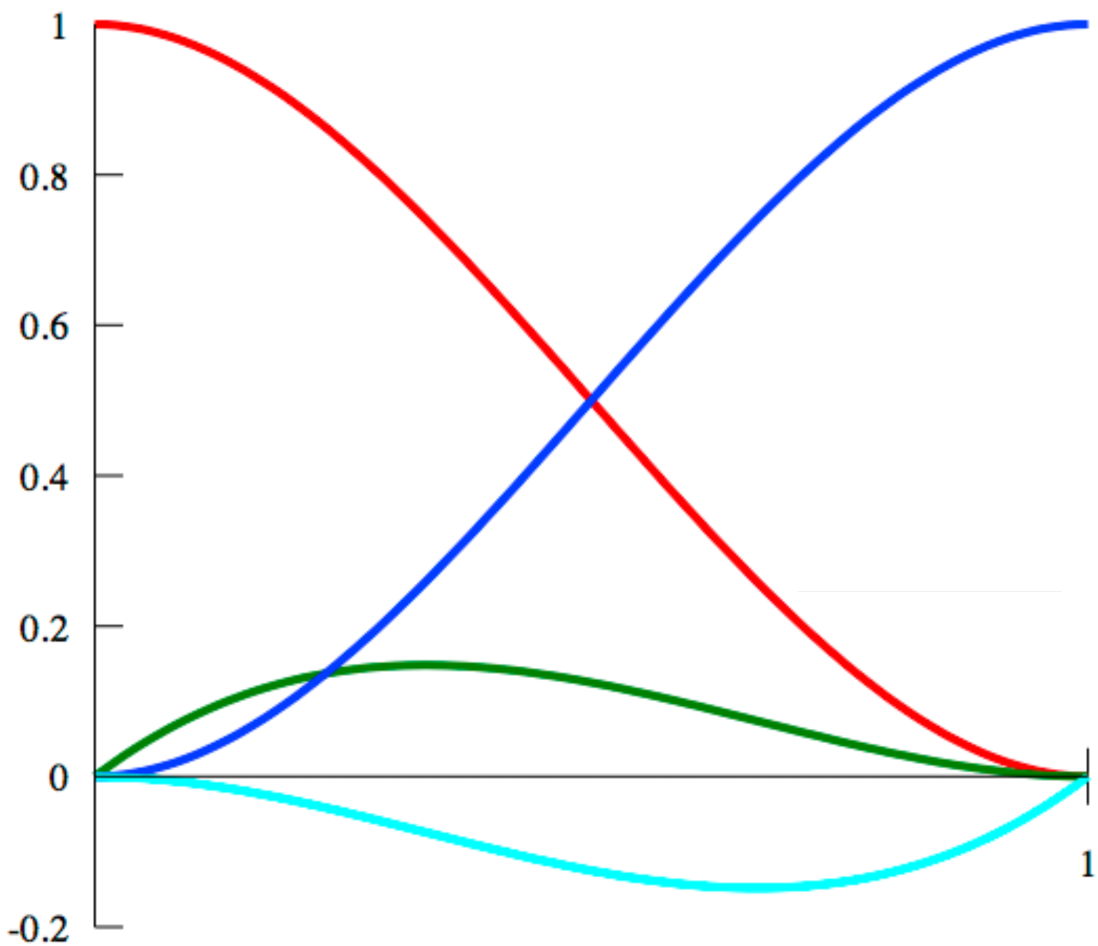
Specify endpoints and derivatives



construct curve with C^1 continuity



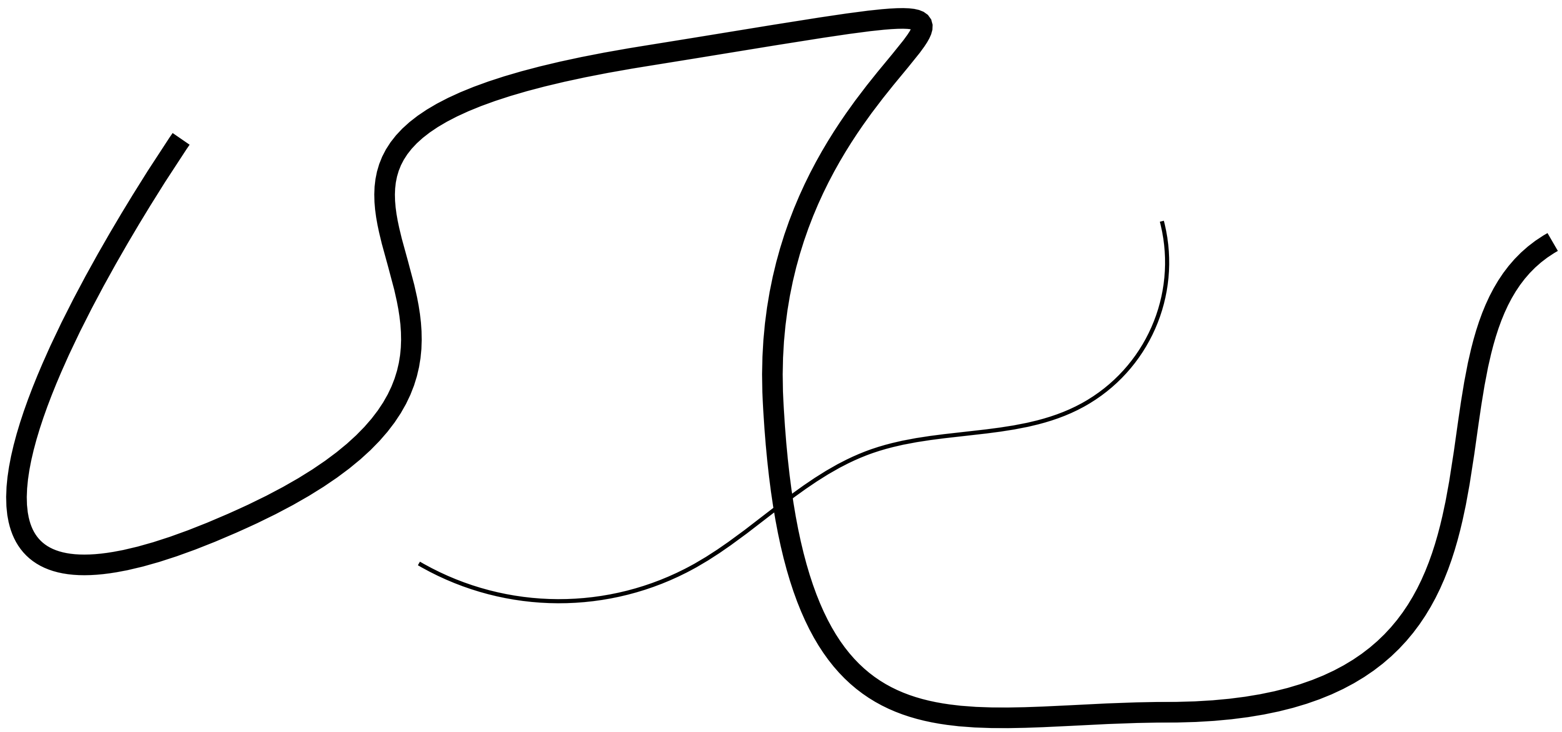
Hermite blending functions



[Wikimedia Commons]

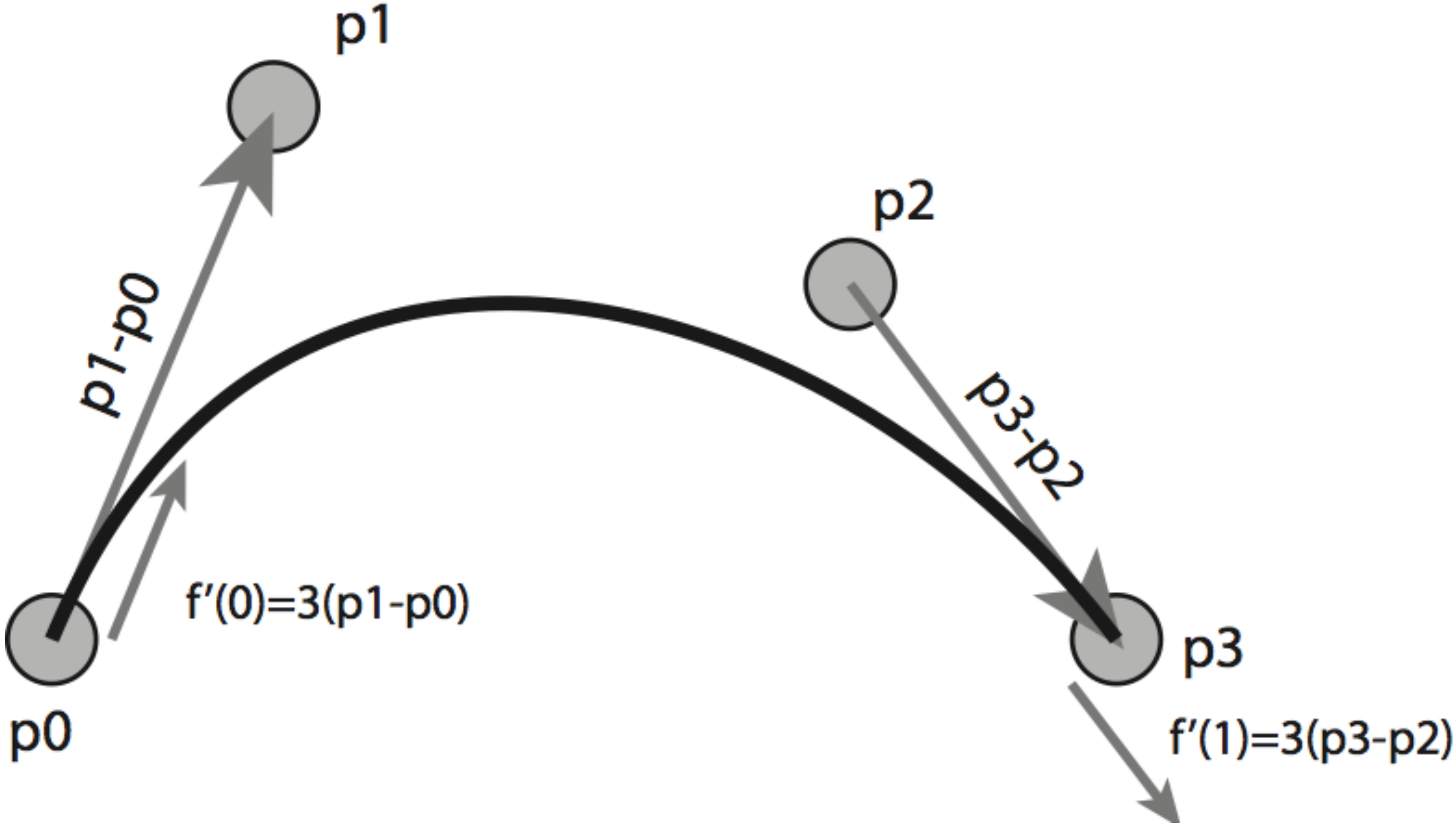
$$b_0(u) = 2u^3 - 3u^2 + 1$$
$$b_1(u) = -2u^3 + 3u^2$$
$$b_2(u) = u^3 - 2u^2 + u$$
$$b_3(u) = u^3 - u^2$$

Example: keynote curve tool

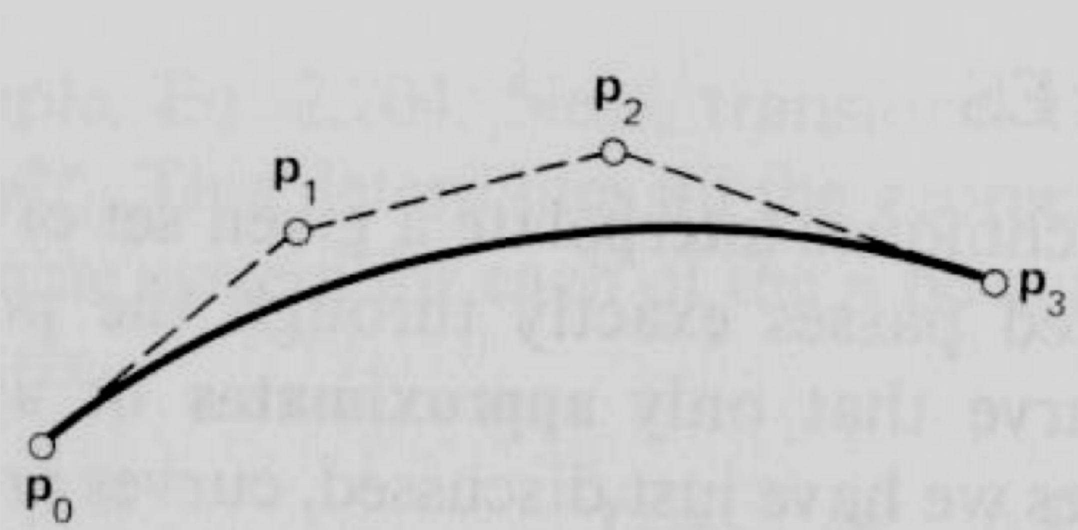


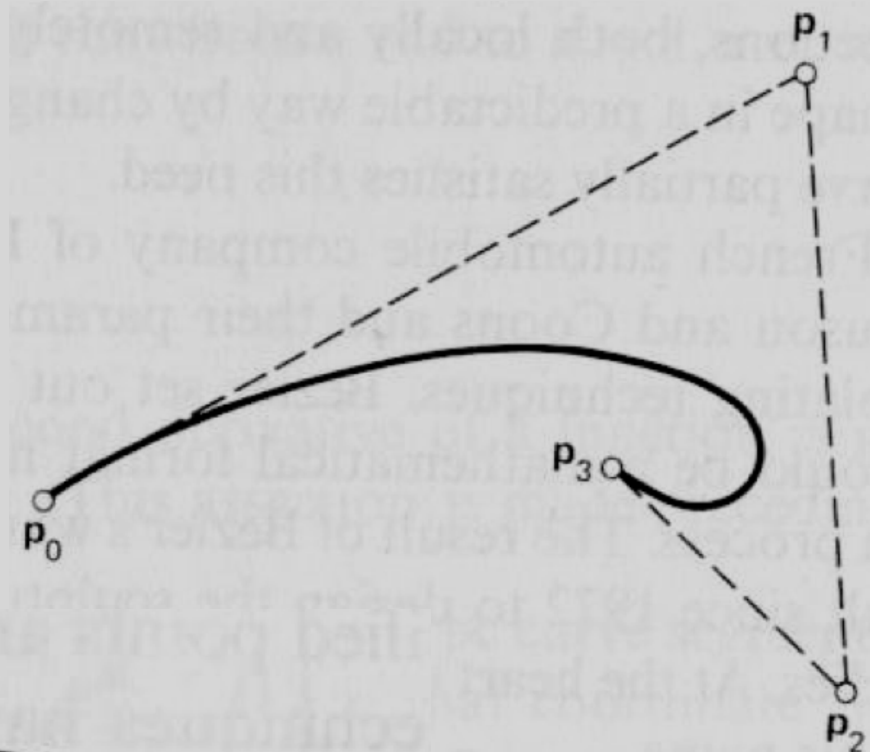
Cubic Bezier Curves

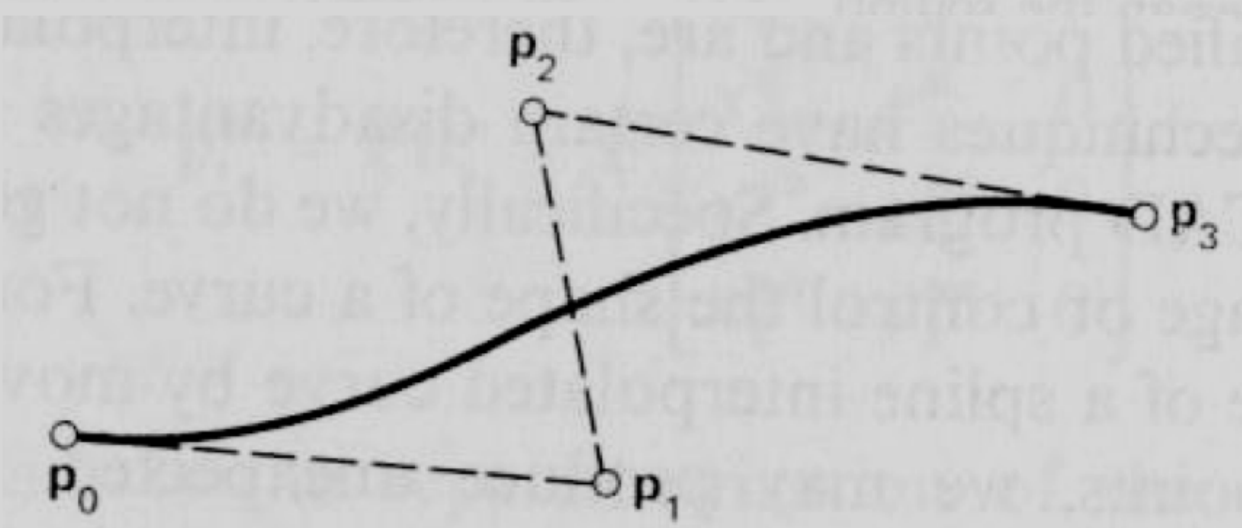
Cubic Bezier Curves



Cubic Bezier Curve Examples



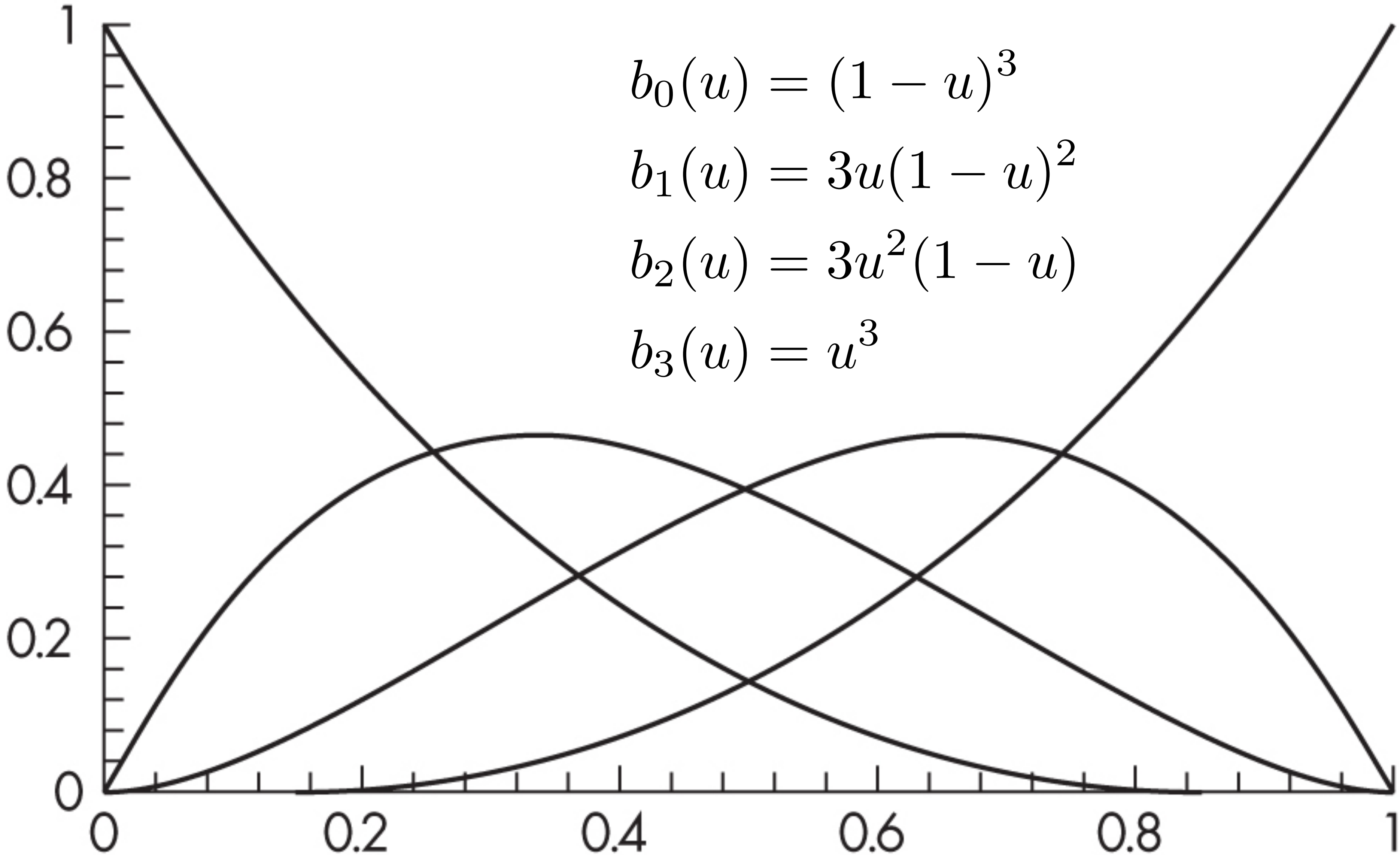




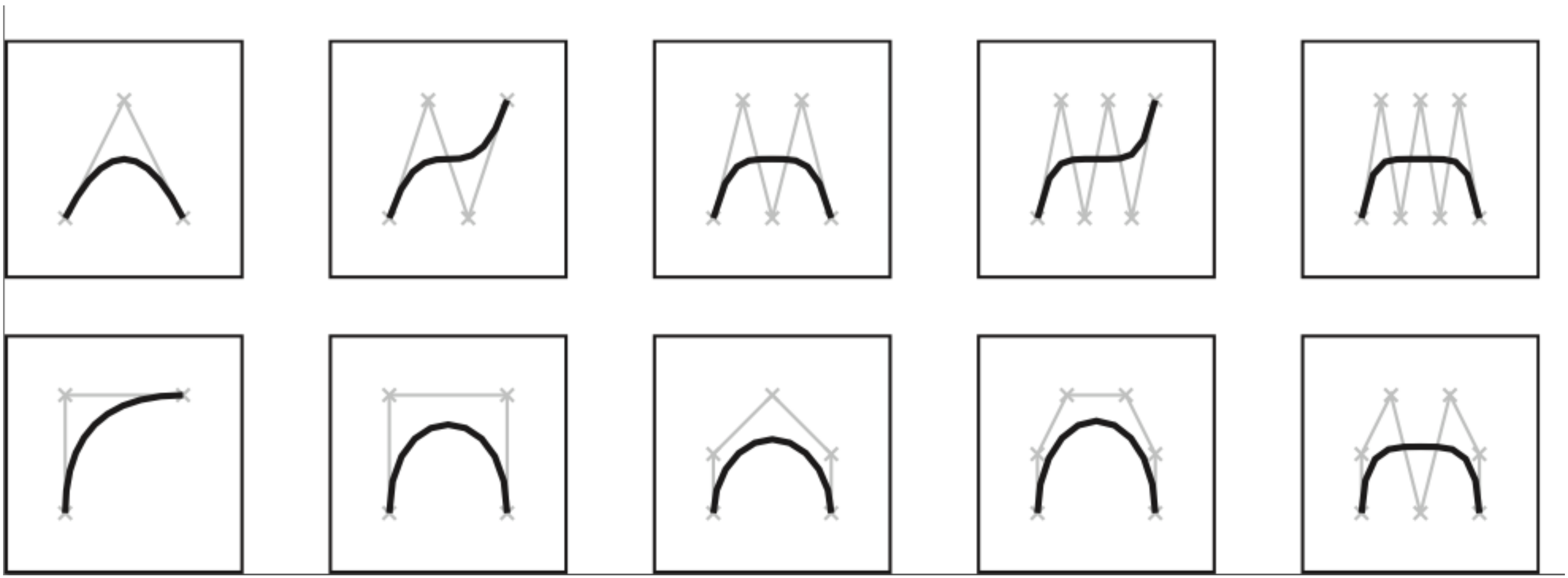
Cubic Bezier blending functions

<whiteboard>

Cubic Bezier blending functions



Bezier Curves Degrees 2-6

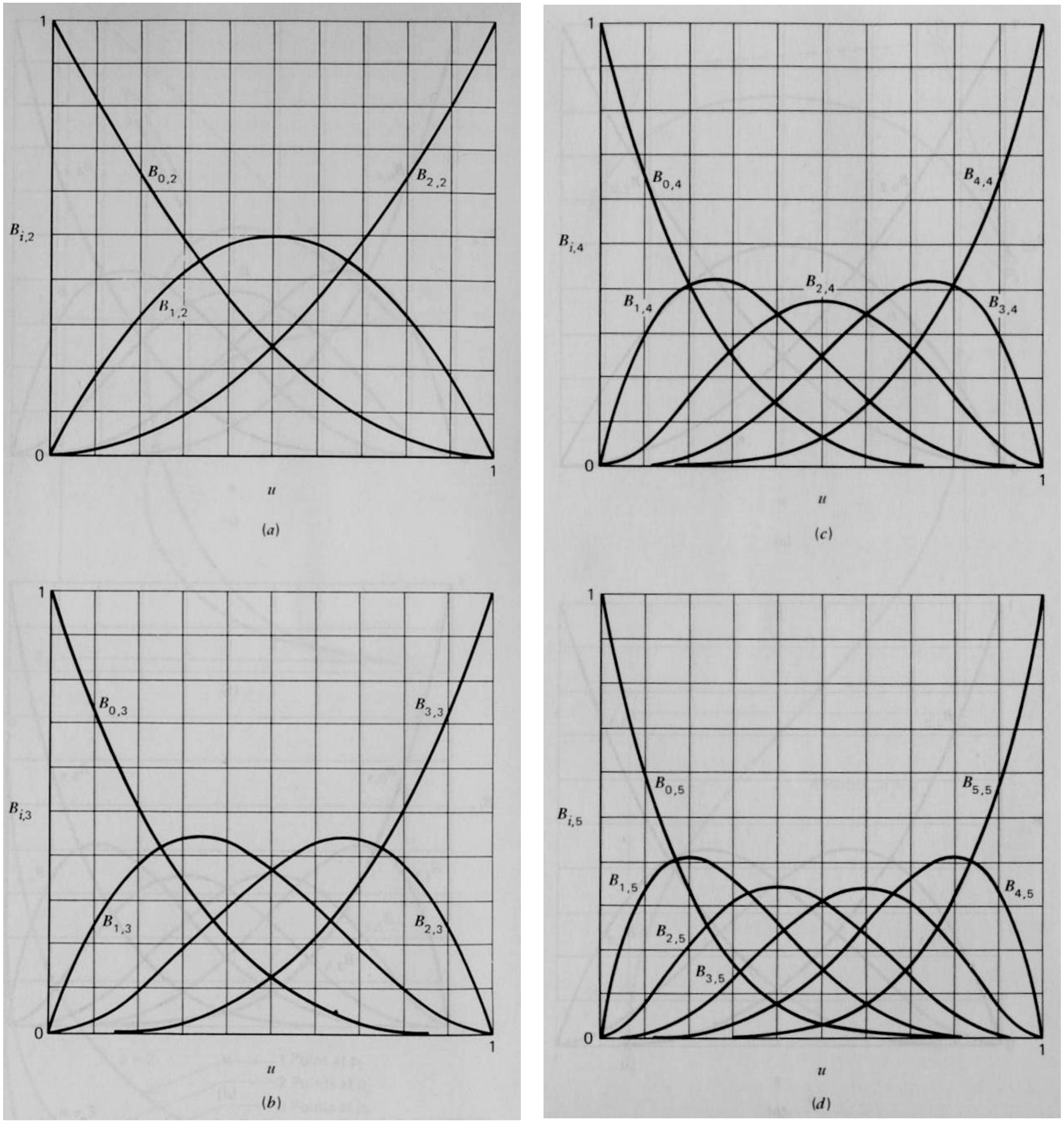


Bernstein Polynomials

- The blending functions are a special case of the Bernstein polynomials

$$b_{\mathrm{kd}}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k}$$

- These polynomials give the blending polynomials for any degree Bezier form
  - All roots at 0 and 1
  - For any degree they all sum to 1
  - They are all between 0 and 1 inside (0,1)



1
0
1
$B_{i,2}$
$B_{0,2}$
$B_{1,2}$
$B_{2,2}$
u
(a)
1
0
1
$B_{i,4}$
$B_{0,4}$
$B_{1,4}$
$B_{2,4}$
$B_{3,4}$
$B_{4,4}$
u
(c)
1
0
1
$B_{i,3}$
$B_{0,3}$
$B_{1,3}$
$B_{2,3}$
$B_{3,3}$
u
(b)
1
0
1
$B_{i,5}$
$B_{0,5}$
$B_{1,5}$
$B_{2,5}$
$B_{3,5}$
$B_{4,5}$
$B_{5,5}$
u
(d)

d = 2

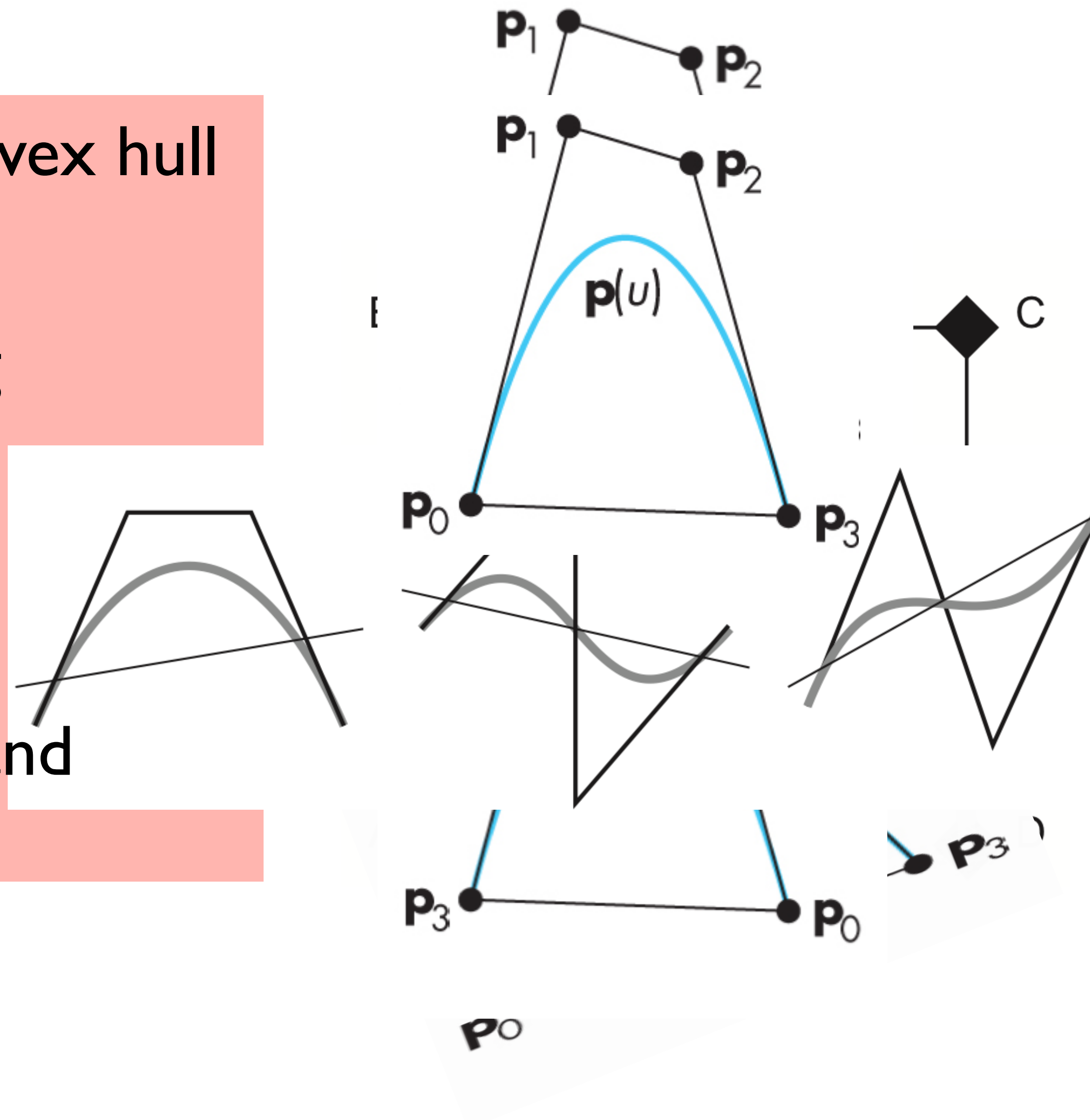
d = 4

d = 3

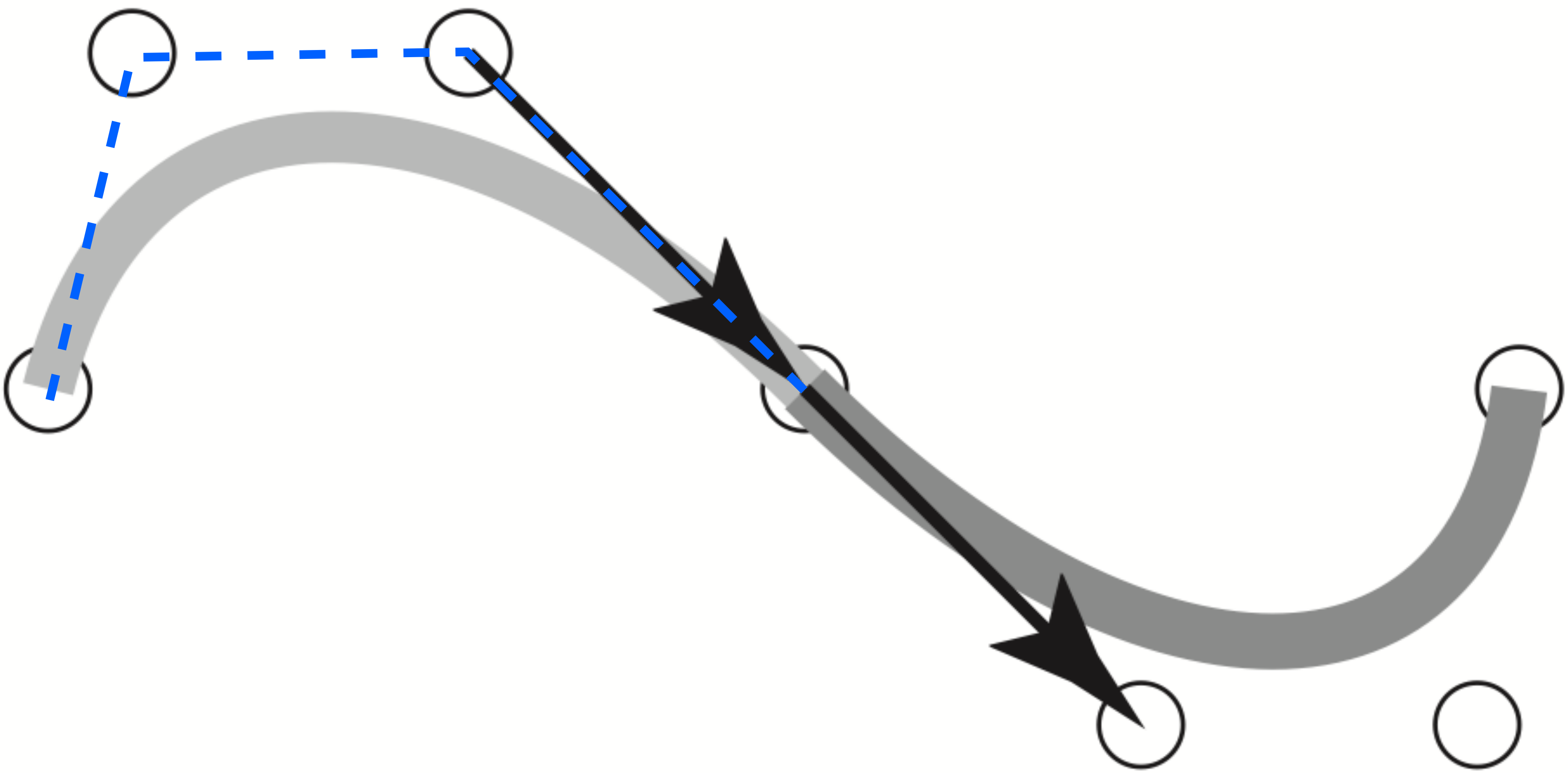
d = 5

Bezier Curve Properties

- curve lies in the convex hull of the data
- variation diminishing
- symmetry
- affine invariant
- efficient evaluation and subdivision

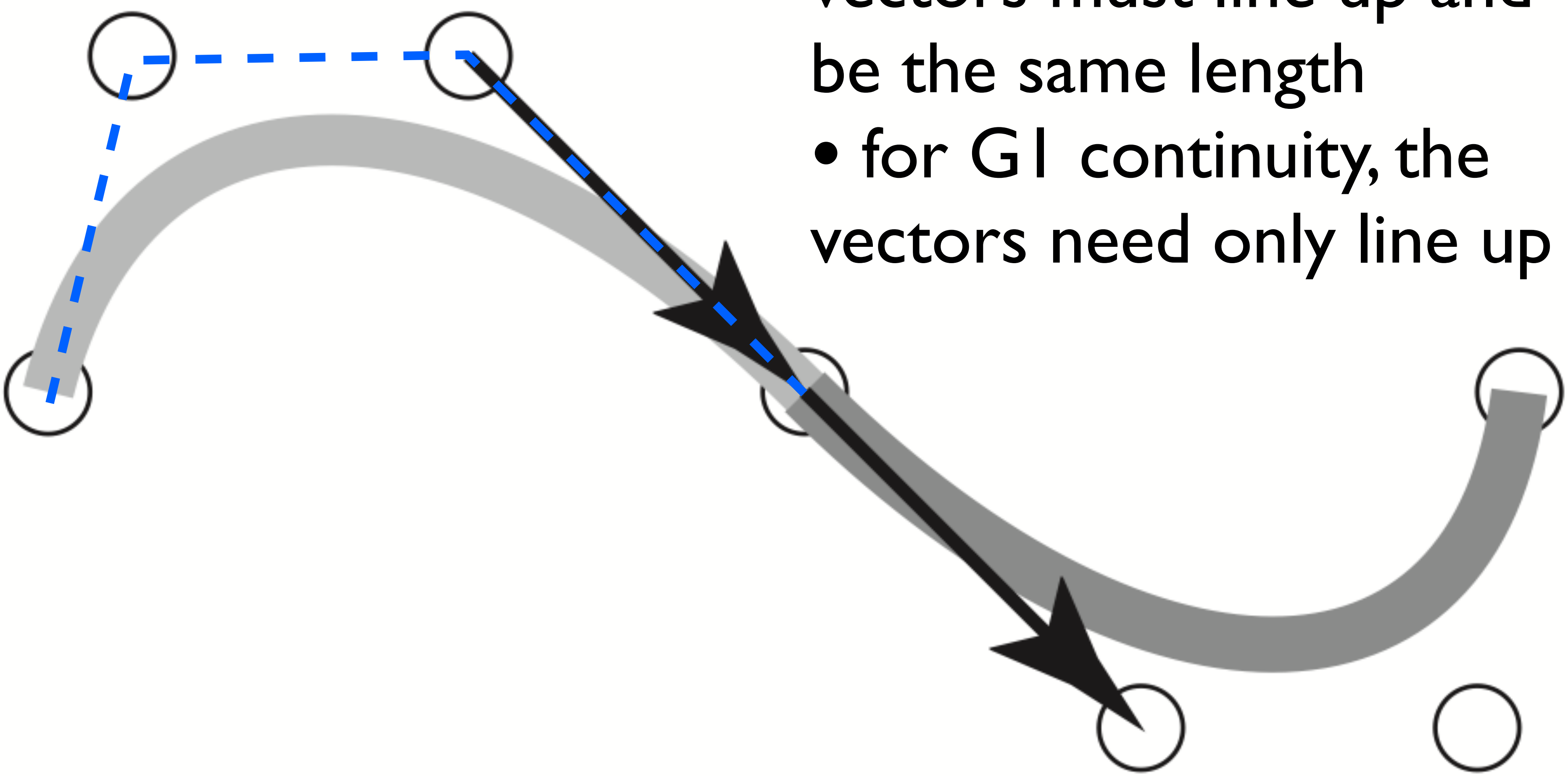


Joining Cubic Bezier Curves

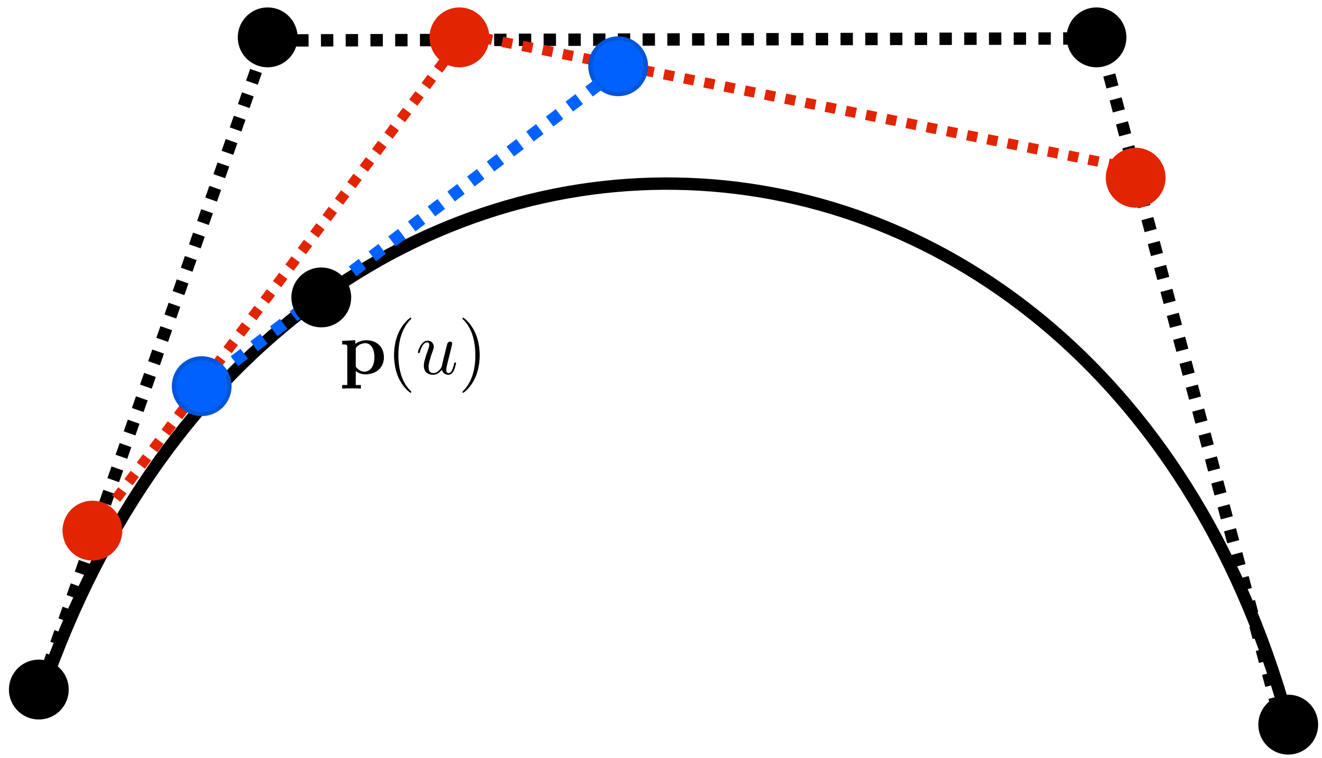


Joining Cubic Bezier Curves

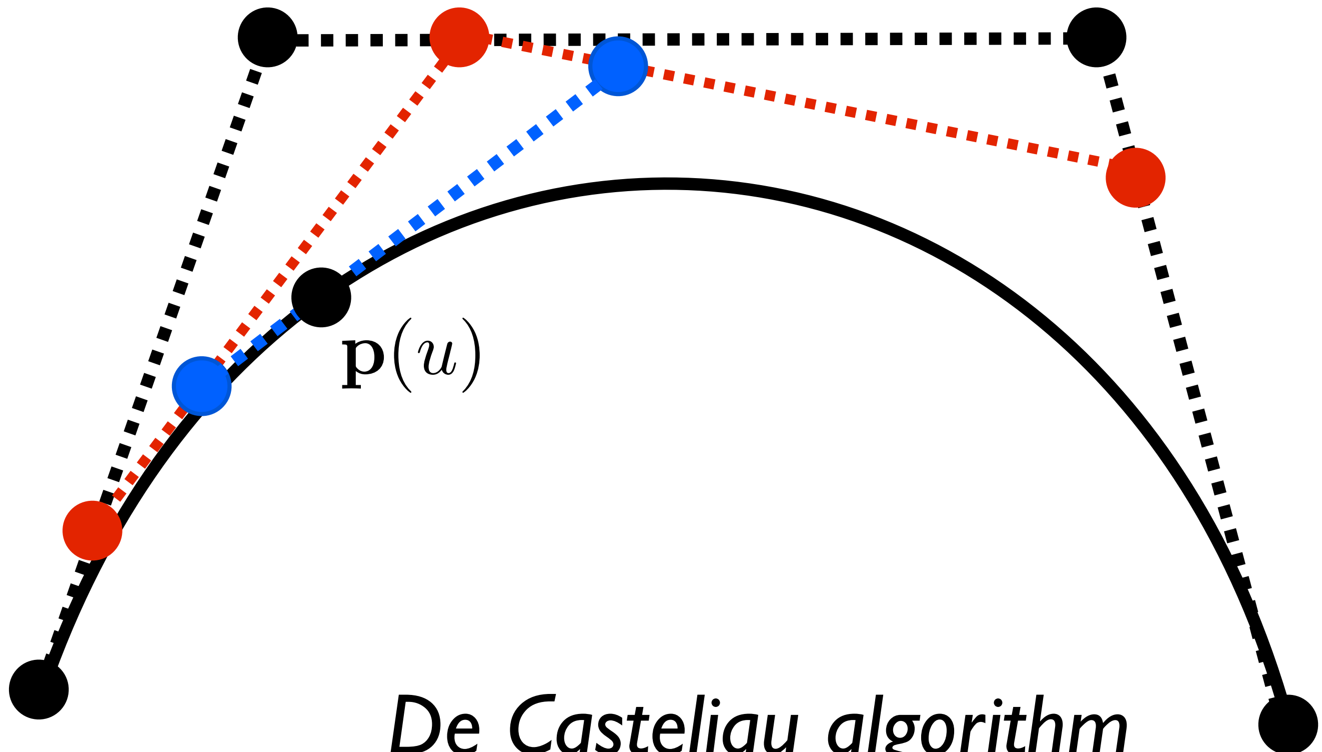
- for C1 continuity, the vectors must line up and be the same length
- for G1 continuity, the vectors need only line up



Evaluating p(u) geometrically

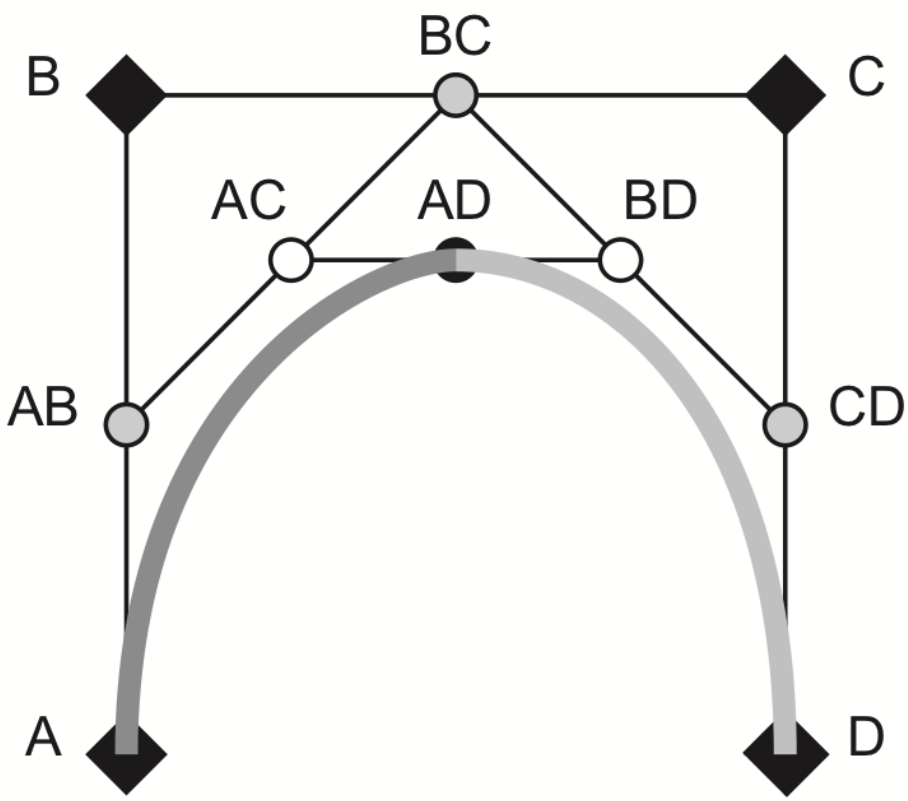


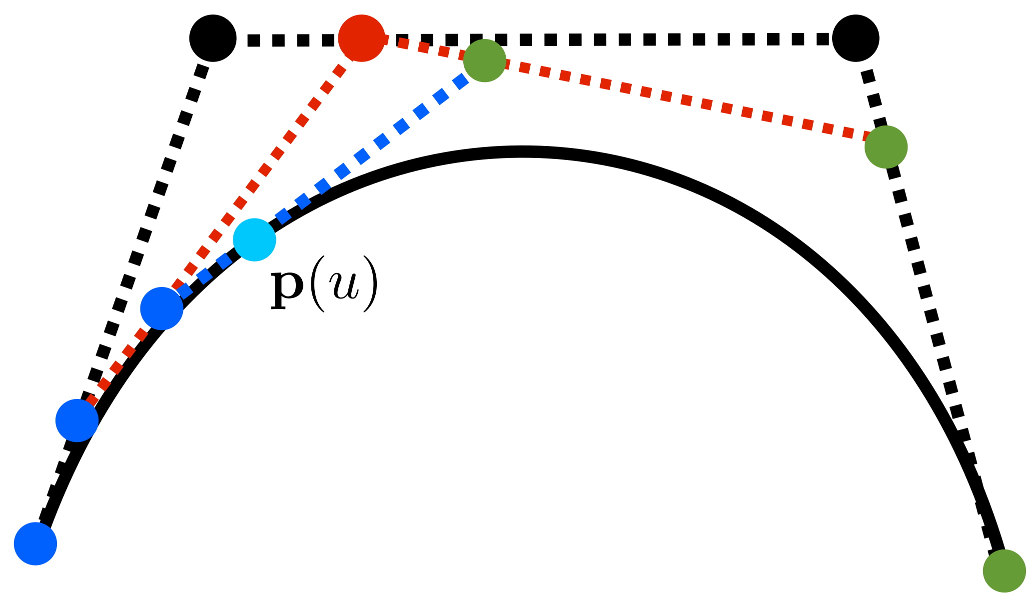
Evaluating p(u) geometrically



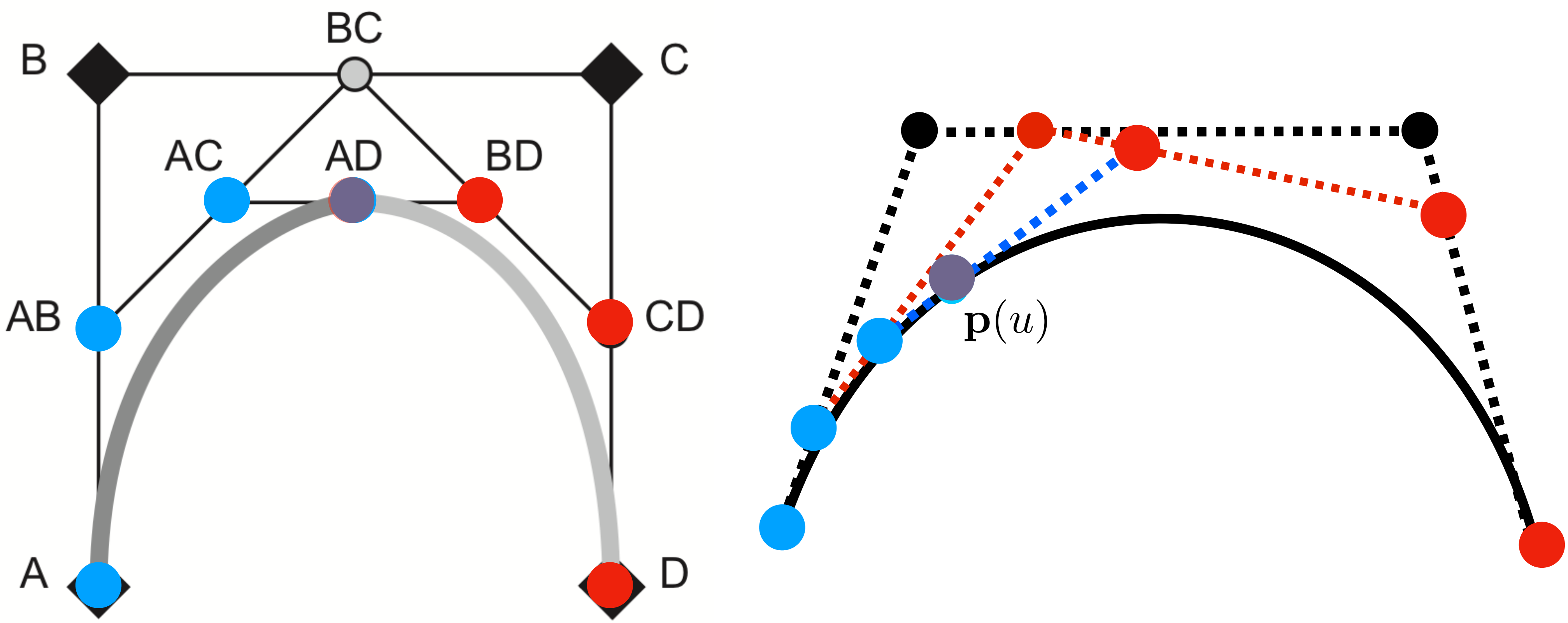
De Casteljau algorithm

Bezier subdivision

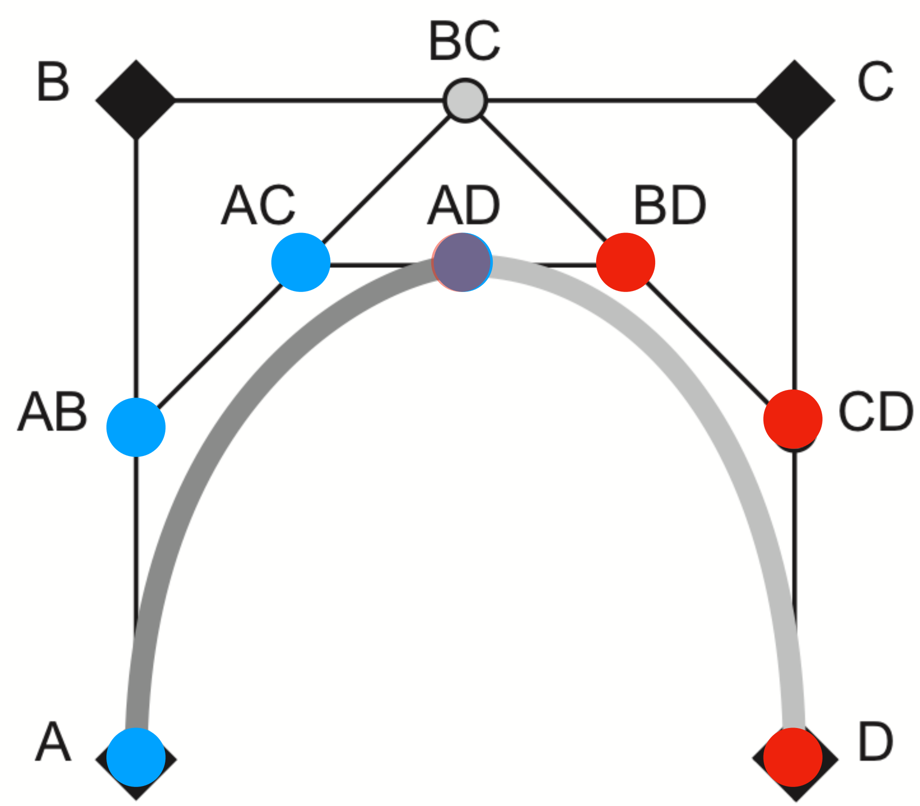


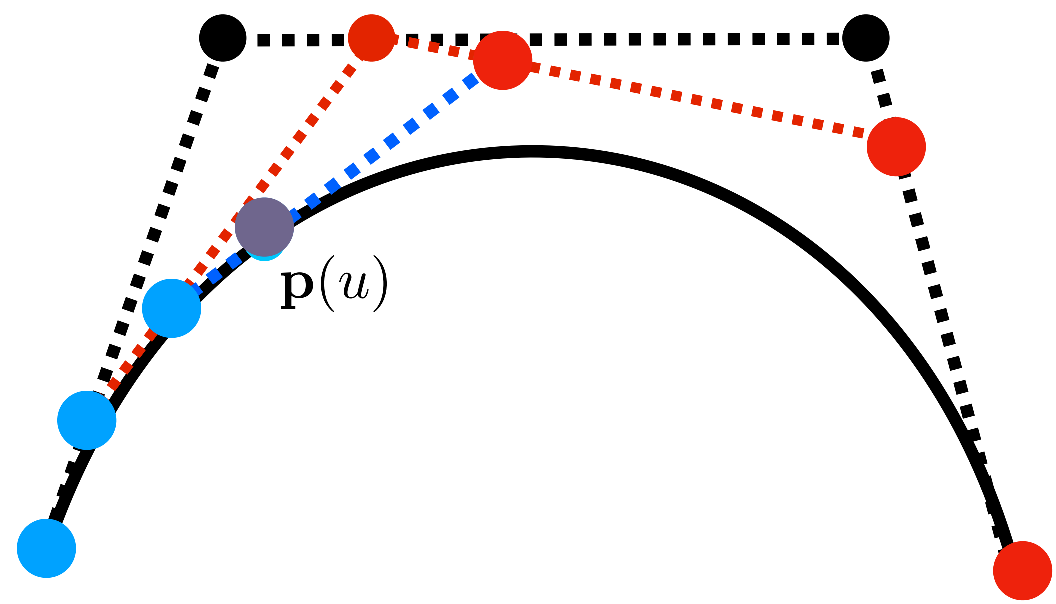


Bezier subdivision



Bezier subdivision

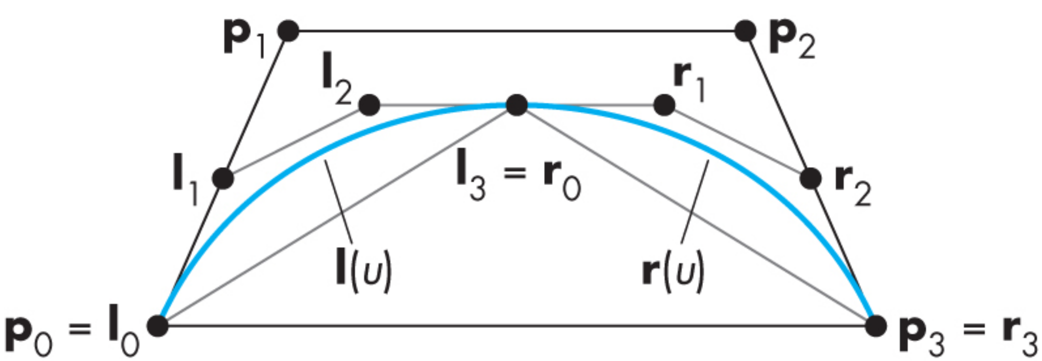


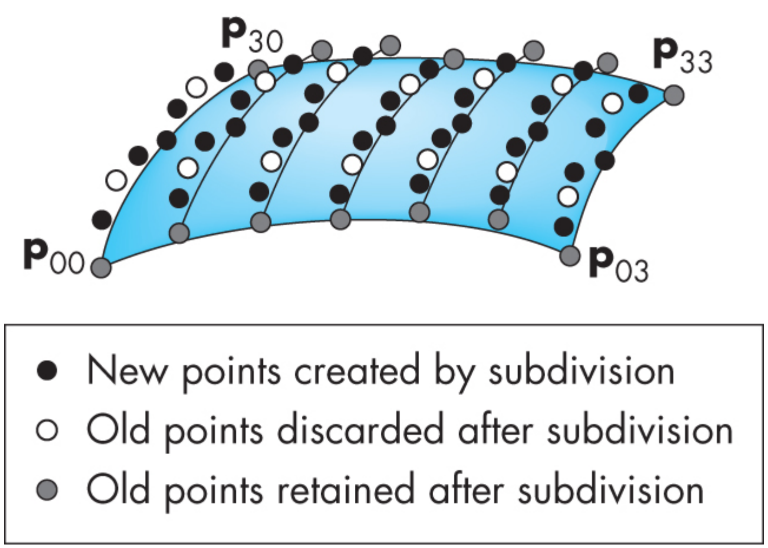


divid and conquer approach can be used for efficient rendering

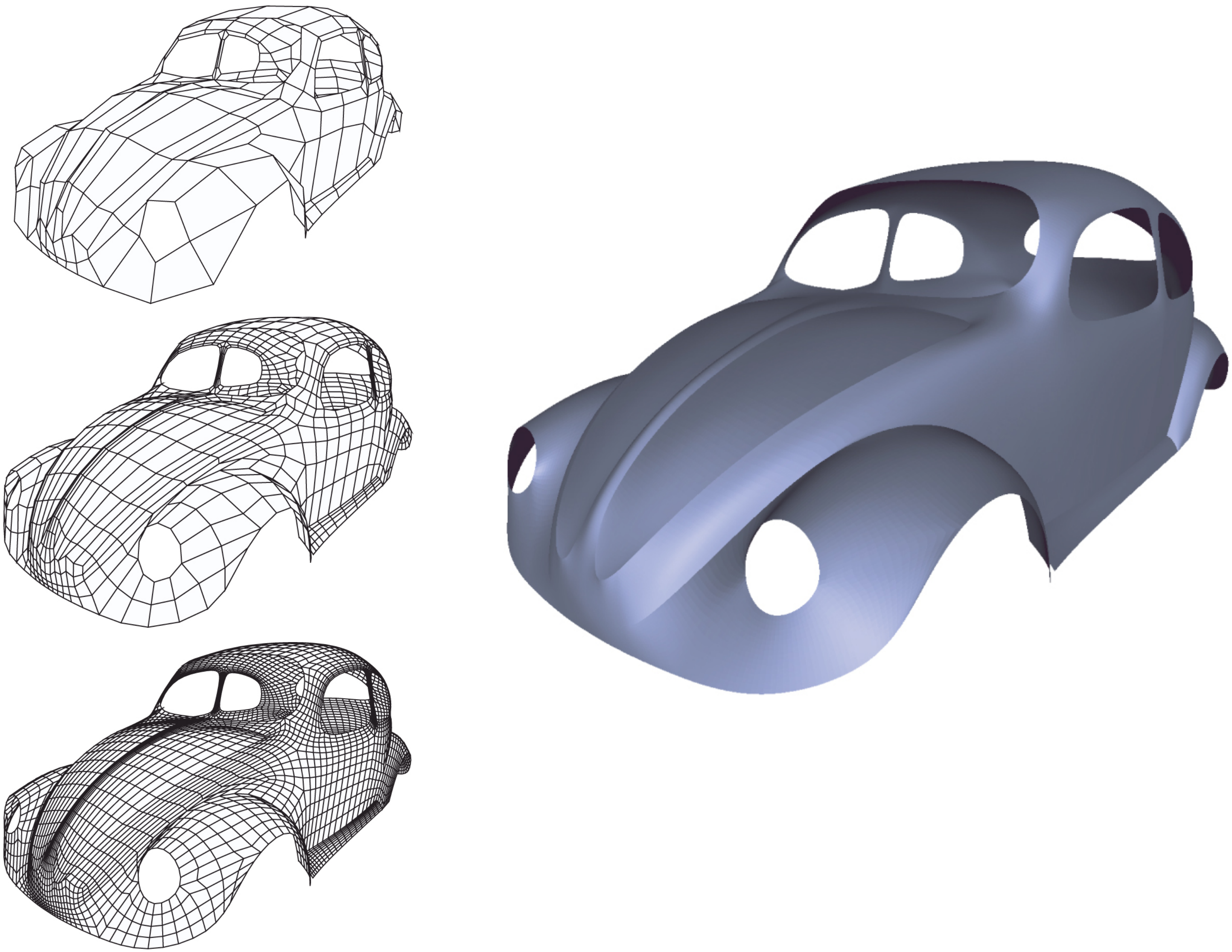
Recursive Subdivision

- work with convex hull, does not require evaluating the polynomial
- Bezier curves most convenient -- other curves can be transformed to Bezier
- same approach for surfaces





Recursive Subdivision for Rendering



Cubic B-Splines

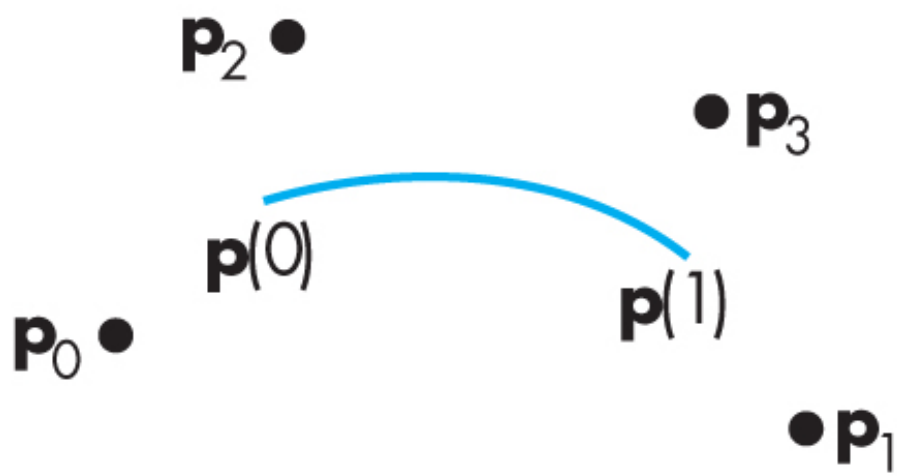
B-spline properties

- polynomials of degree d with (d-1) continuity
- preferred method for very smooth curves (C2 or higher)

B-spline properties

- C(d-1) continuity
- local control - any point on curve depends on d+1 control points
- bounded by convex hull
- variation diminishing property

Cubic B-Splines



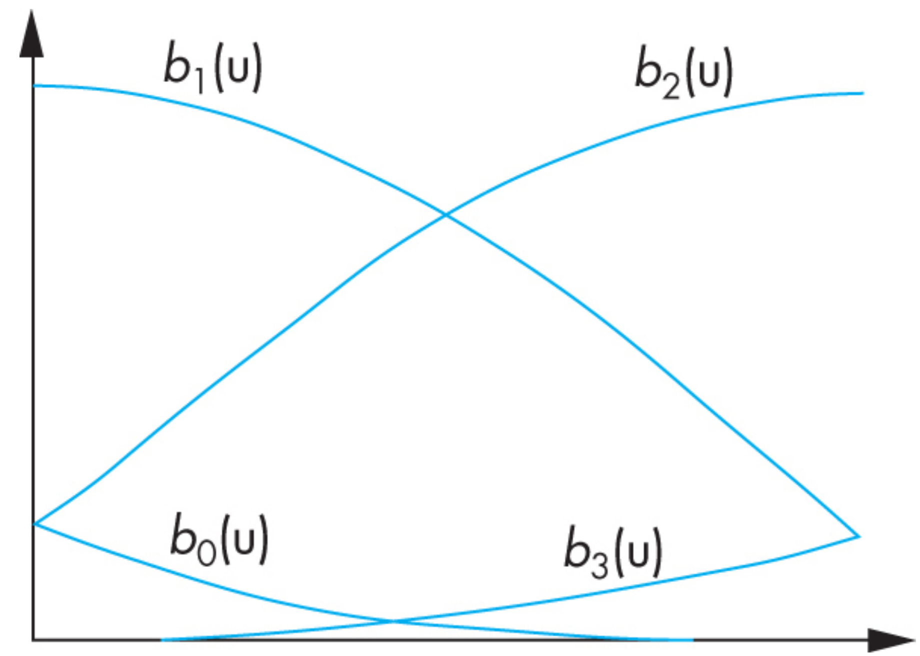
Spline blending functions

$$b_0(u) = \frac{1}{6}(1-u)^3$$

$$b_1(u) = \frac{1}{6}(4-6u^2+3u^3)$$

$$b_2(u) = \frac{1}{6}(1+3u+3u^2-3u^3)$$

$$b_3(u) = \frac{1}{6}u^3$$

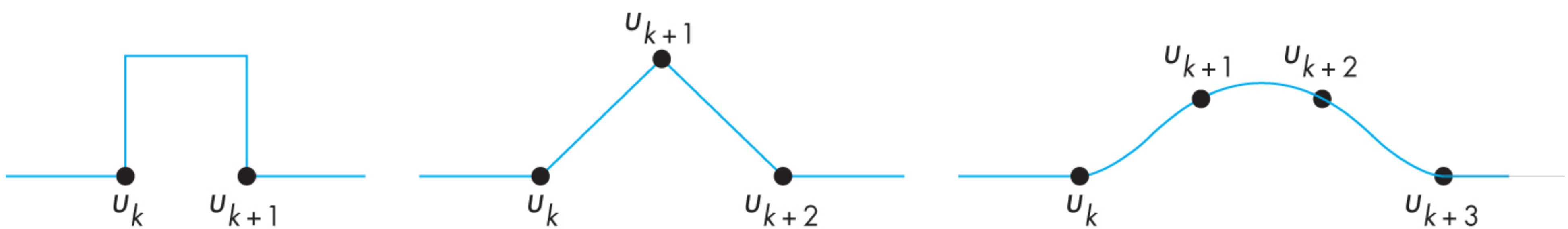


General Splines

- Defined recursively by *Cox-de Boor recursion formula*

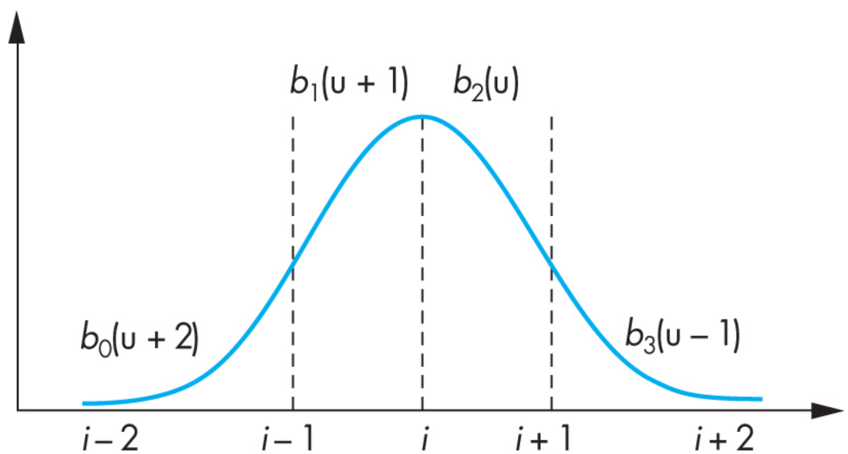
$$b_{j,0}(t) = \begin{cases} 1 & \text{if} \quad t_j \leq t \\ 0 & \text{otherwise} \end{cases}$$

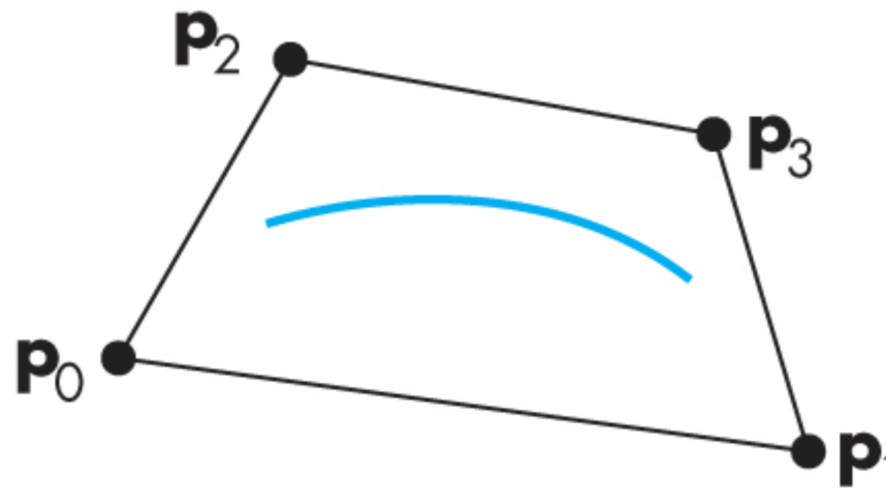
$$b_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t)$$



Spline properties

Basis functions



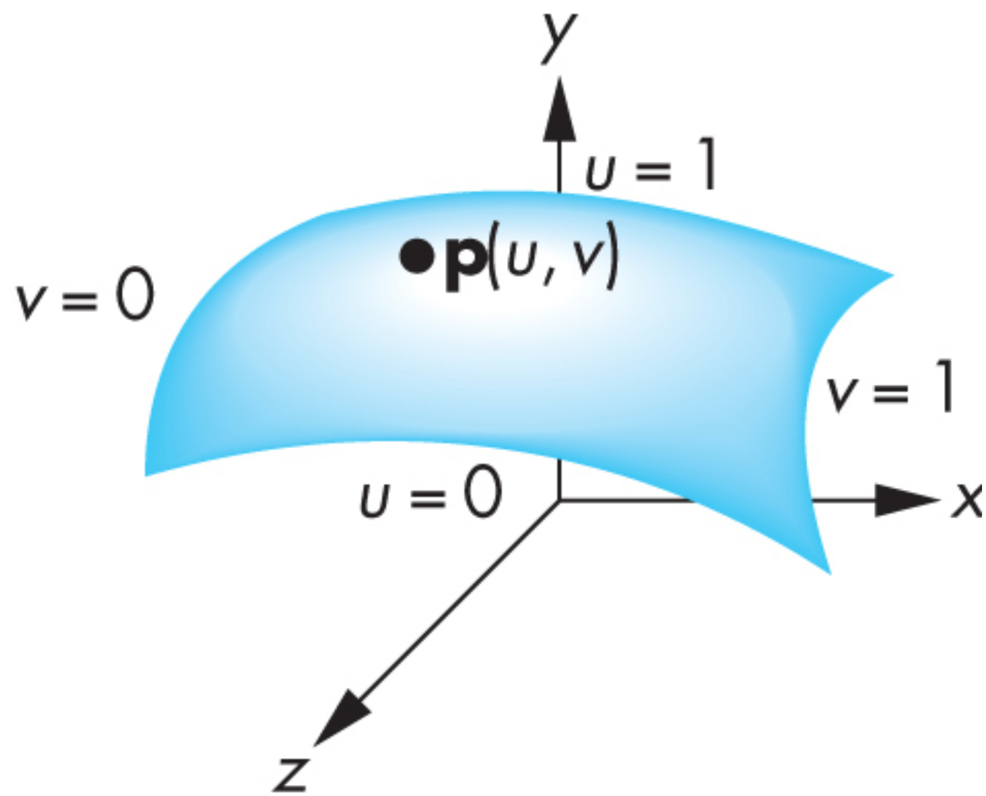


convexity

Surfaces

Parametric Surface

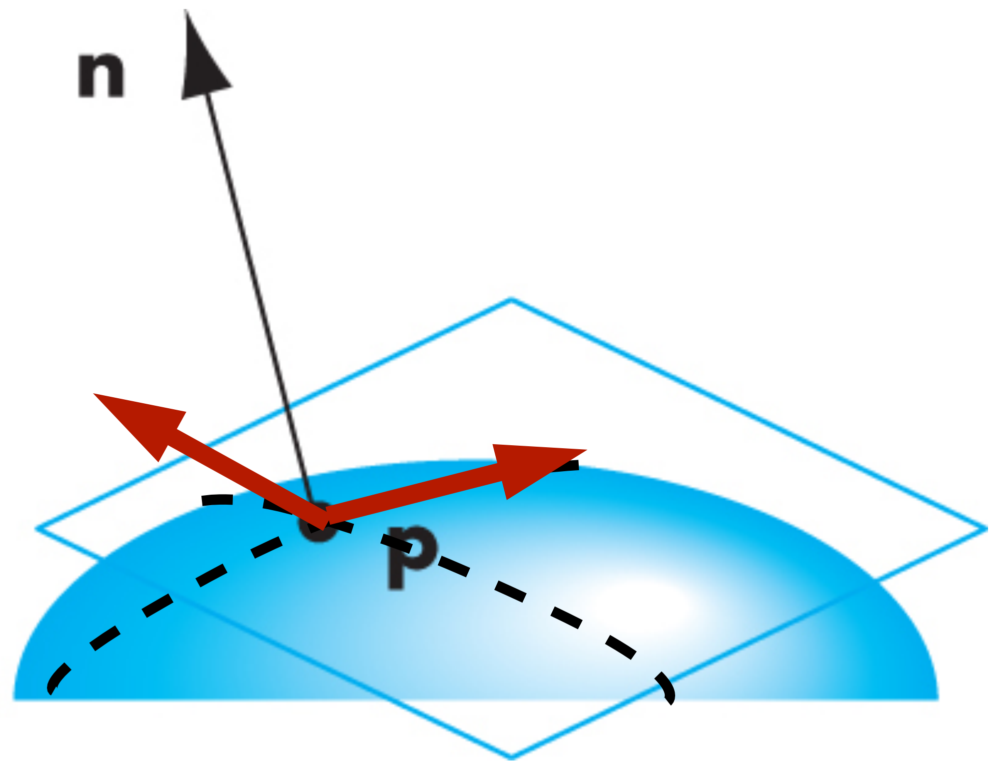
$$x = x(u, v)$$
$$y = y(u, v)$$
$$z = z(u, v)$$



Parametric Surface - tangent plane

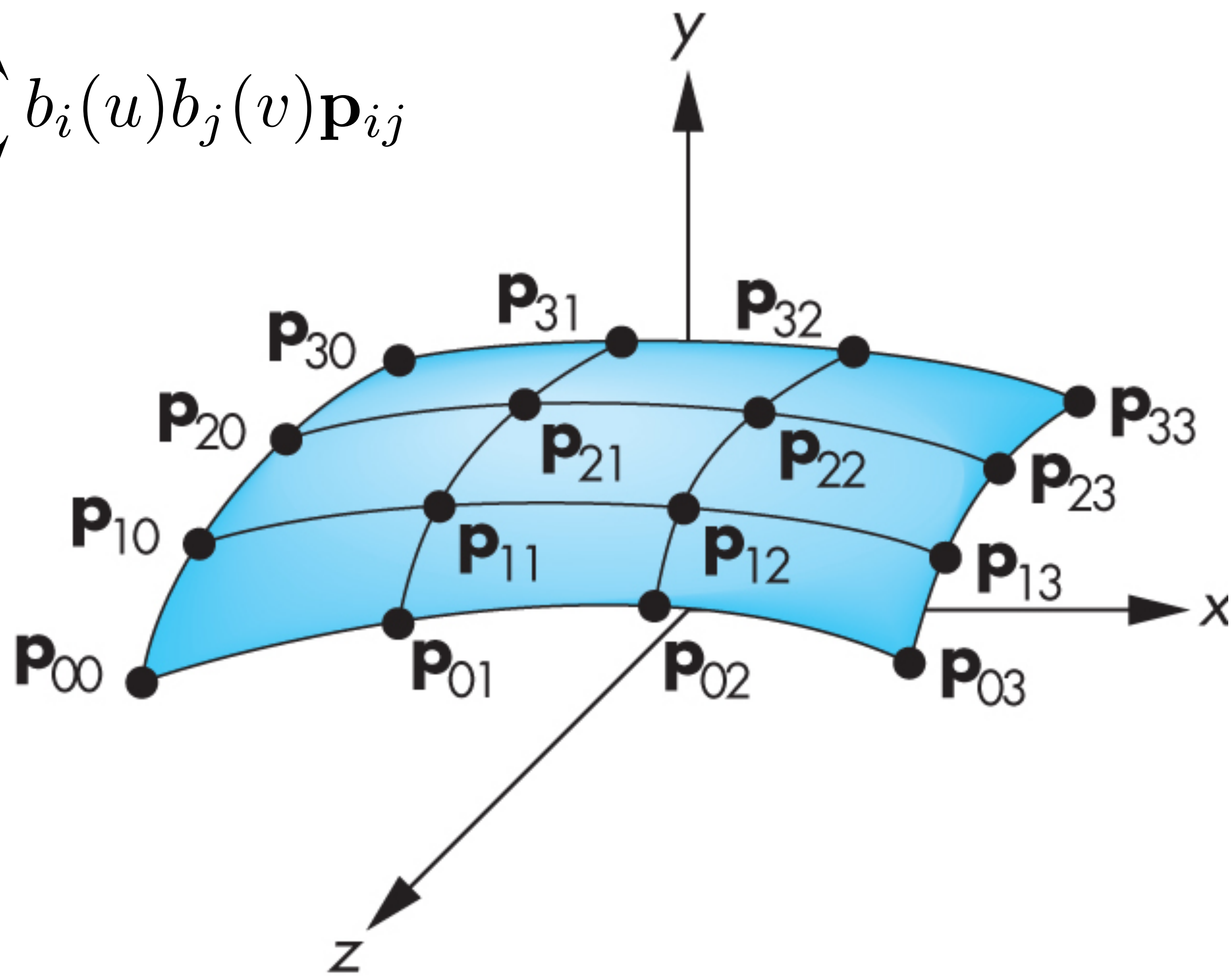
$$\mathbf{t}_u = \begin{pmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{pmatrix}$$

$$\mathbf{t}_v = \begin{pmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial v} \end{pmatrix}$$



Bicubic Surface Patch

$$\mathbf{f}(u,v) = \sum_i \sum_j b_i(u) b_j(v) \mathbf{p}_{ij}$$



Bezier Surface Patch

$$\mathbf{f}(u,v) = \sum_i \sum_j b_i(u) b_j(v) \mathbf{p}_{ij}$$

Patch lies in convex hull