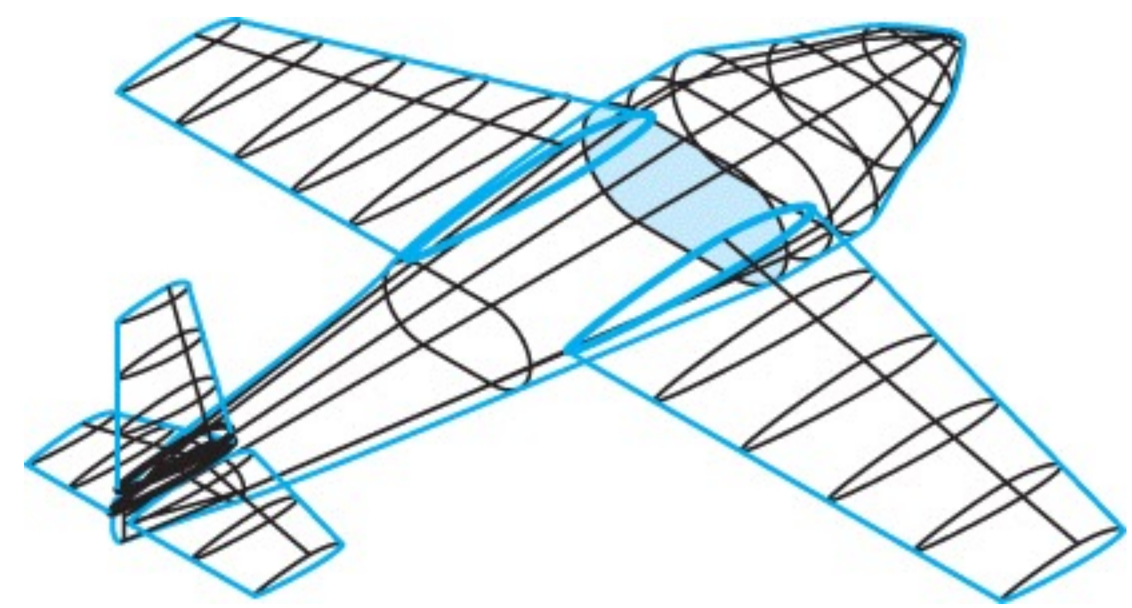
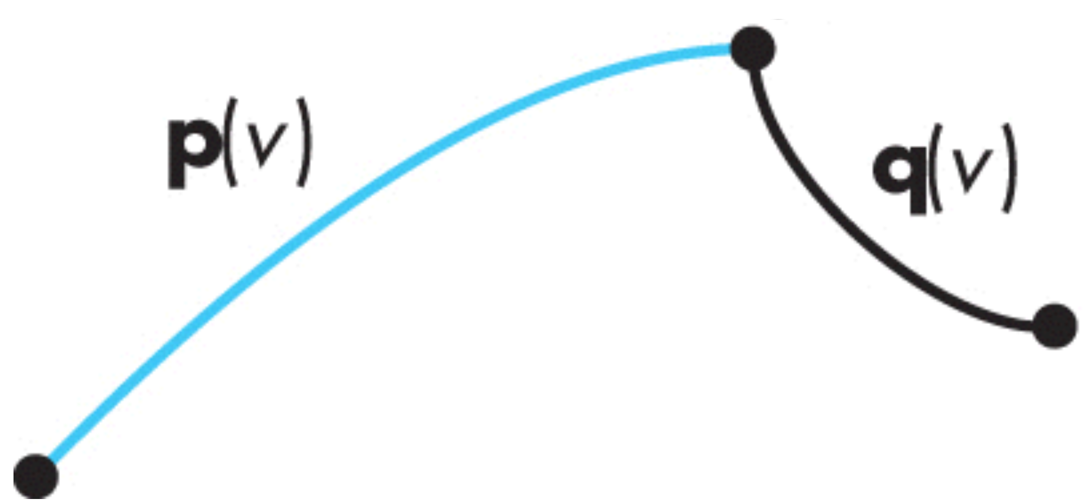
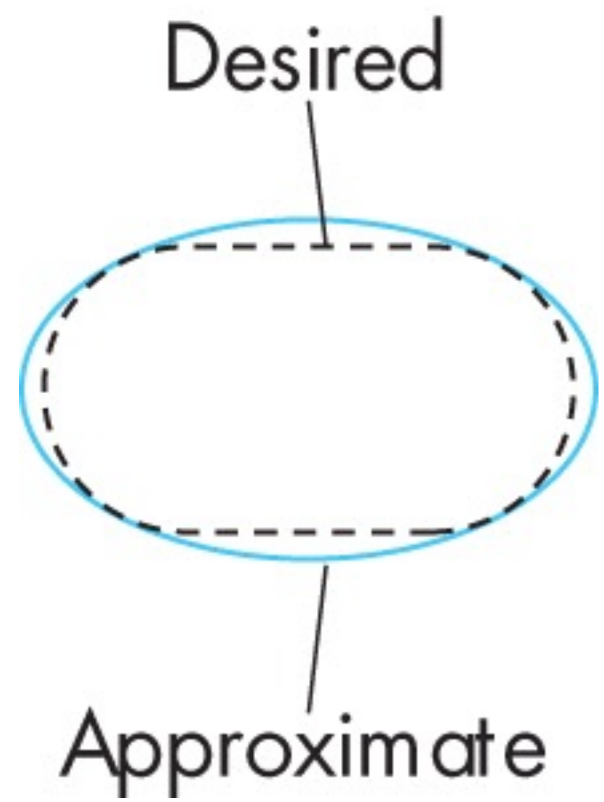
Curves

Design considerations

- local control of shape
  - design each segment independently
- smoothness and continuity
- ability to evaluate derivatives
- stability
  - small change in input leads to small change in output
- ease of rendering

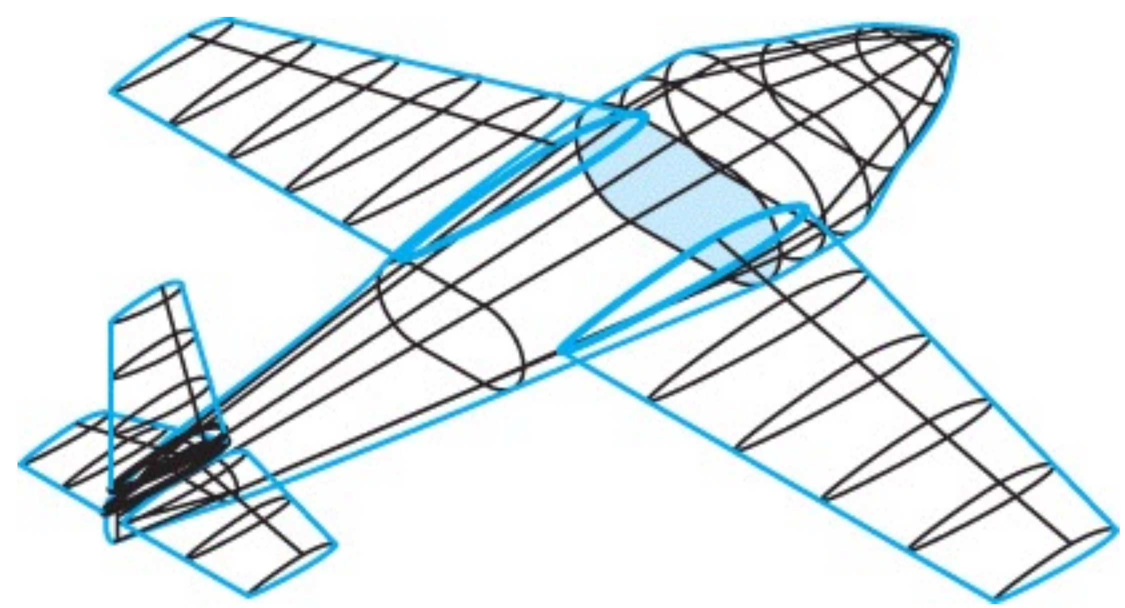


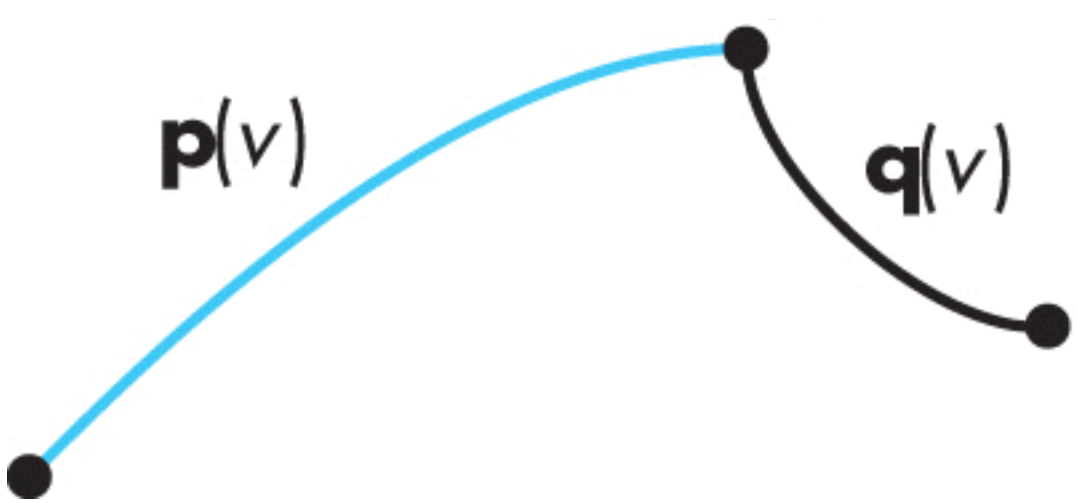


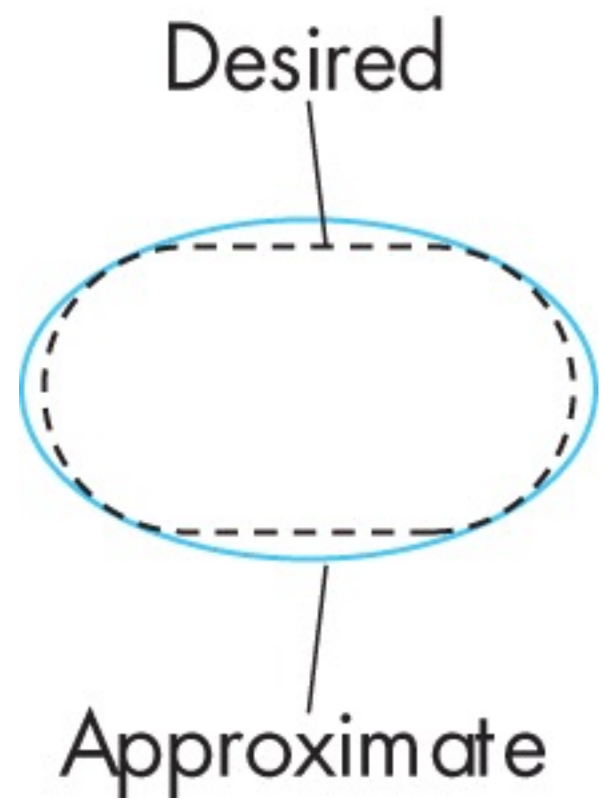


Design considerations

- local control of shape
  - design each segment independently
- smoothness and continuity
- ability to evaluate derivatives
- stability
  - small change in input leads to small change in output
- ease of rendering



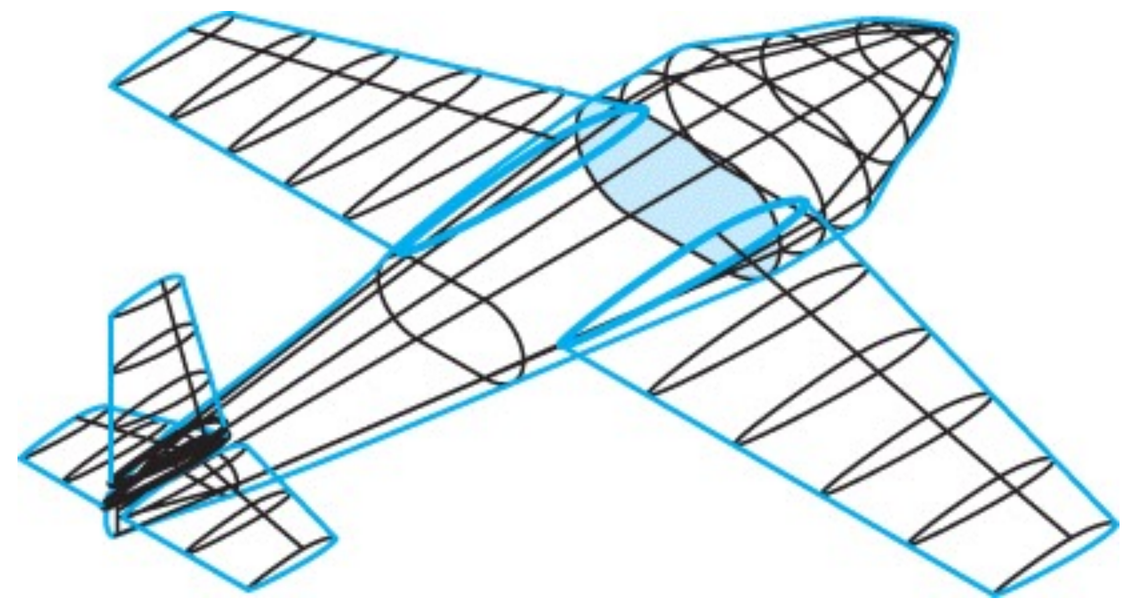


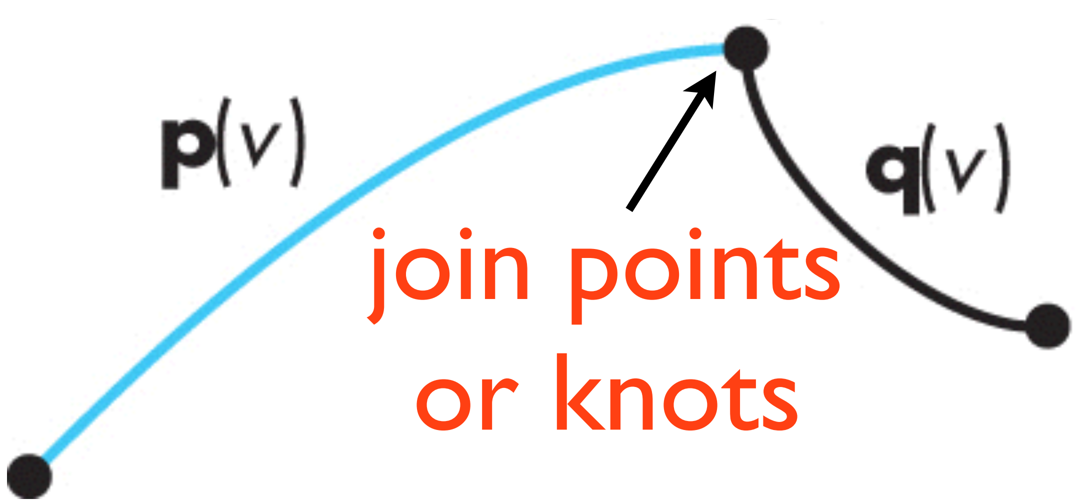


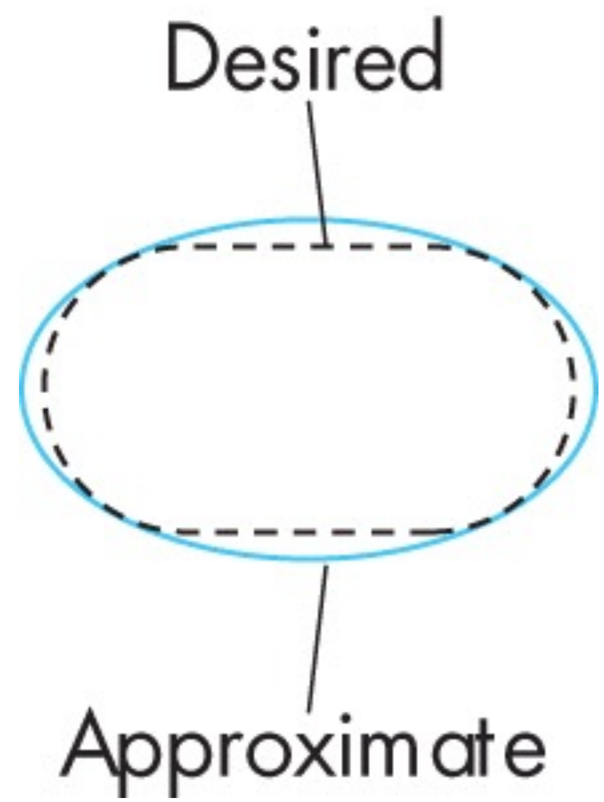
approximate out of a number of wood strips

Design considerations

- local control of shape
  - design each segment independently
- smoothness and continuity
- ability to evaluate derivatives
- stability
  - small change in input leads to small change in output
- ease of rendering





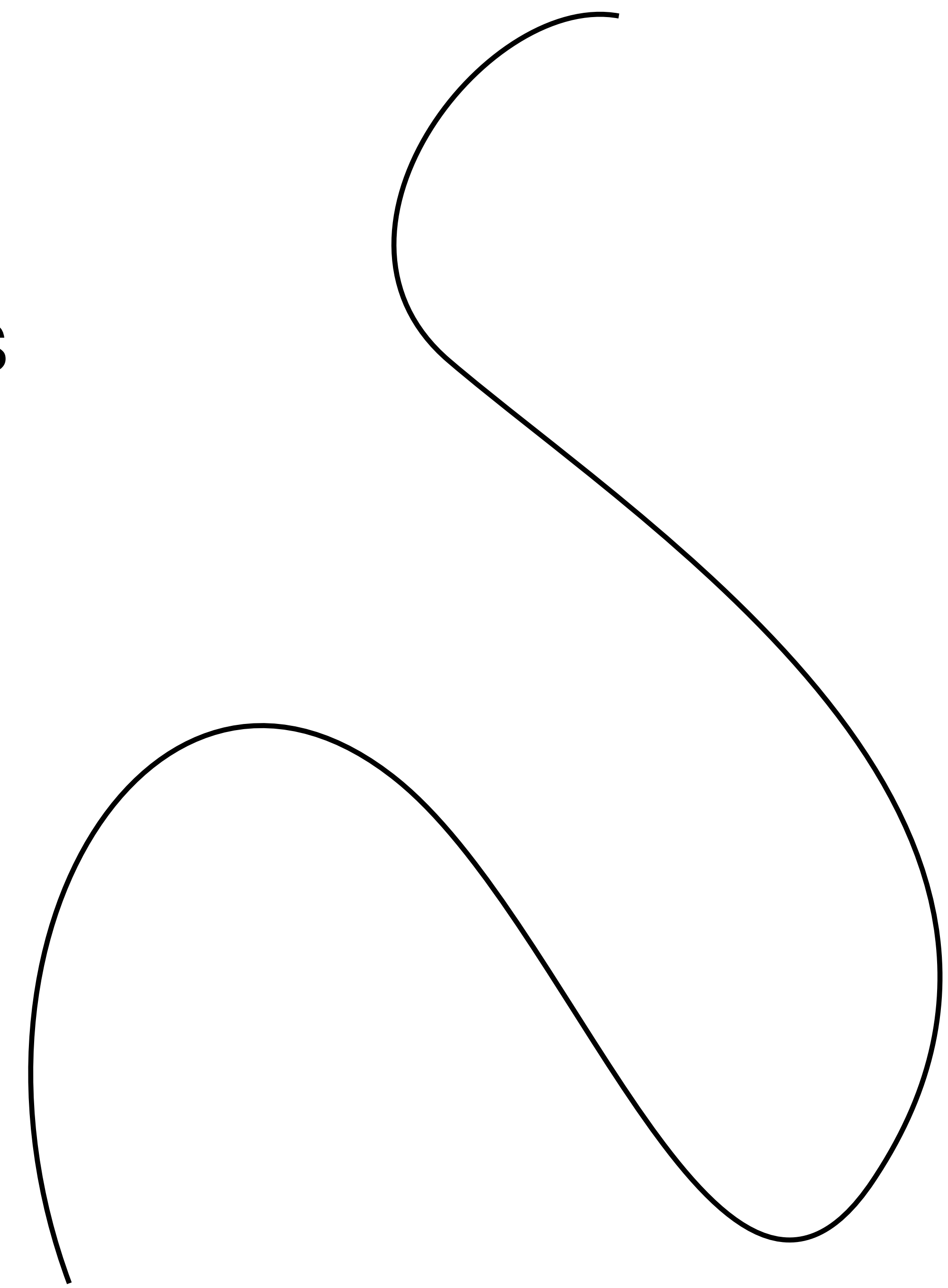


approximate out of a number of wood strips

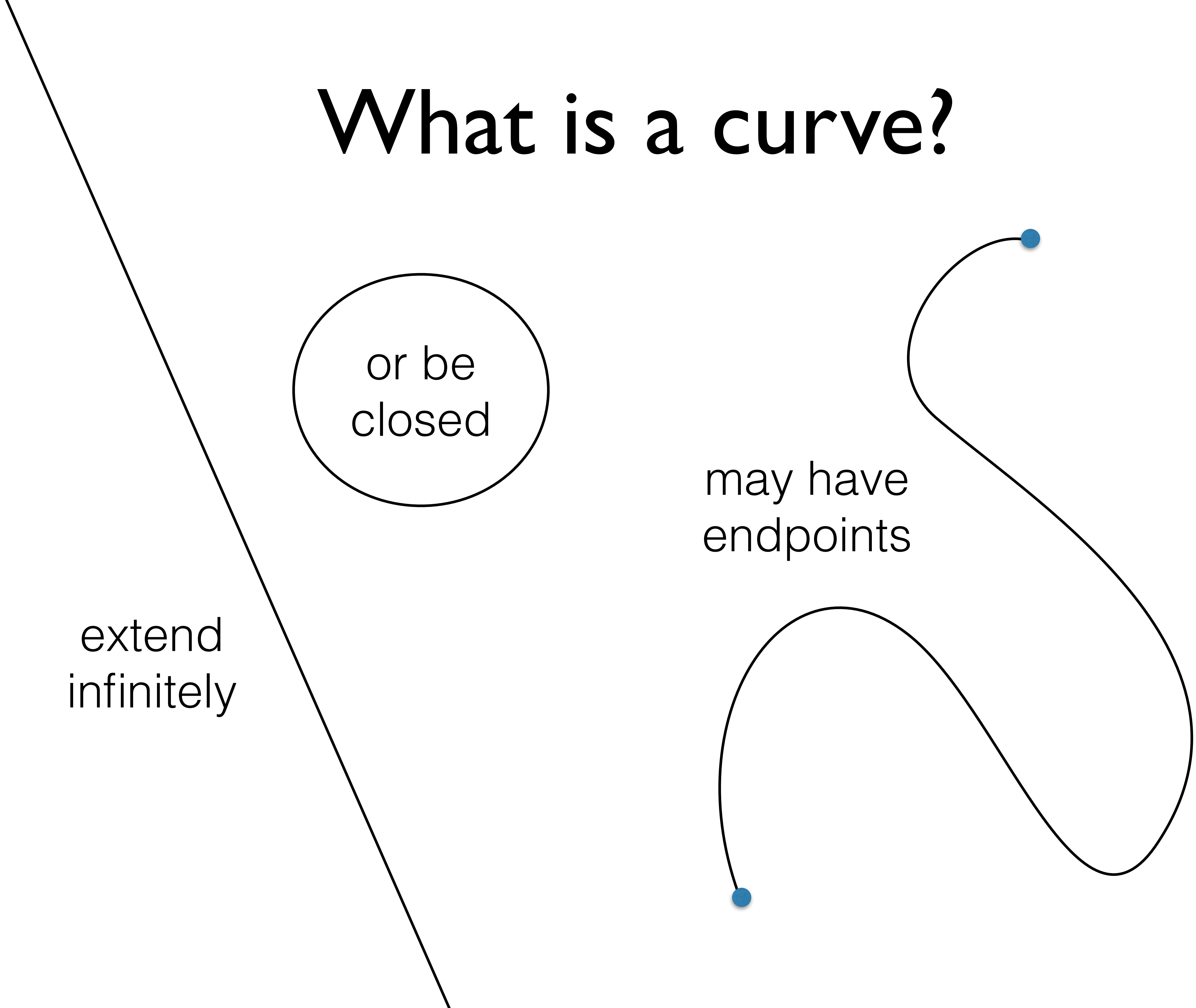
What is a curve?

intuitive idea:
  draw with a pen
  set of points the pen traces

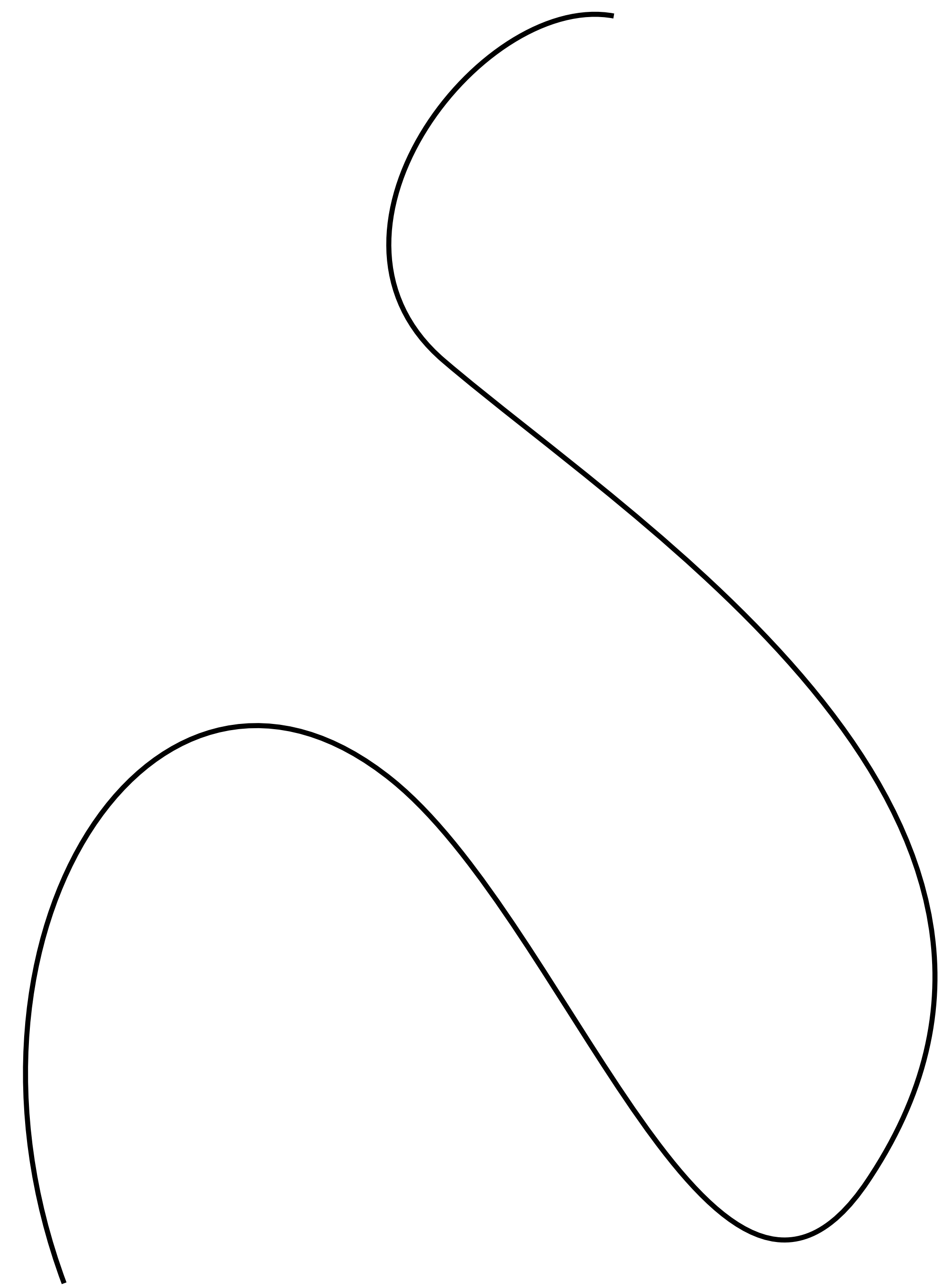
may be 2D, like on paper
or 3D, *space curve*



What is a curve?



How do we specify a curve?

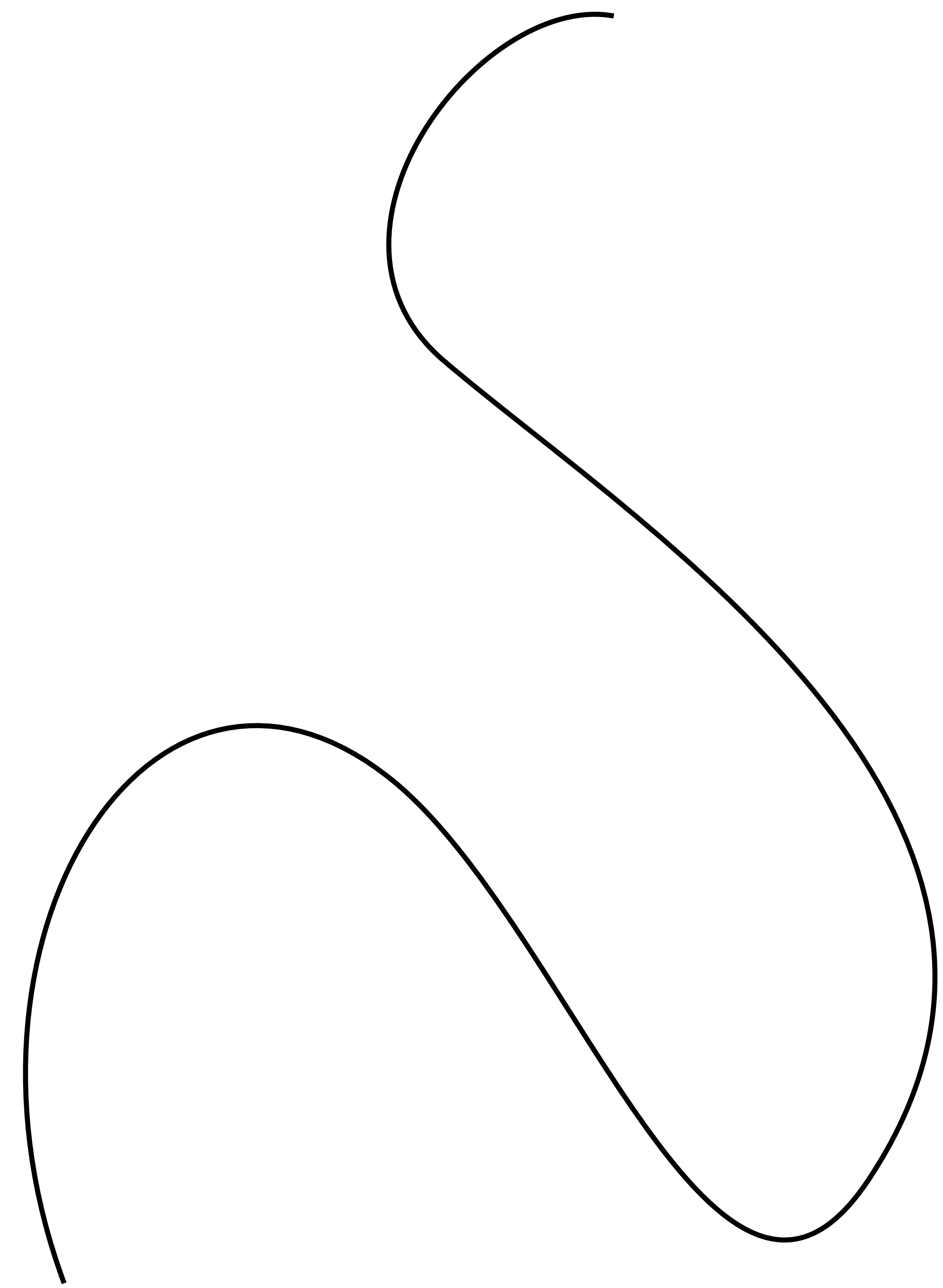


How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$

test if (x,y) is on the curve

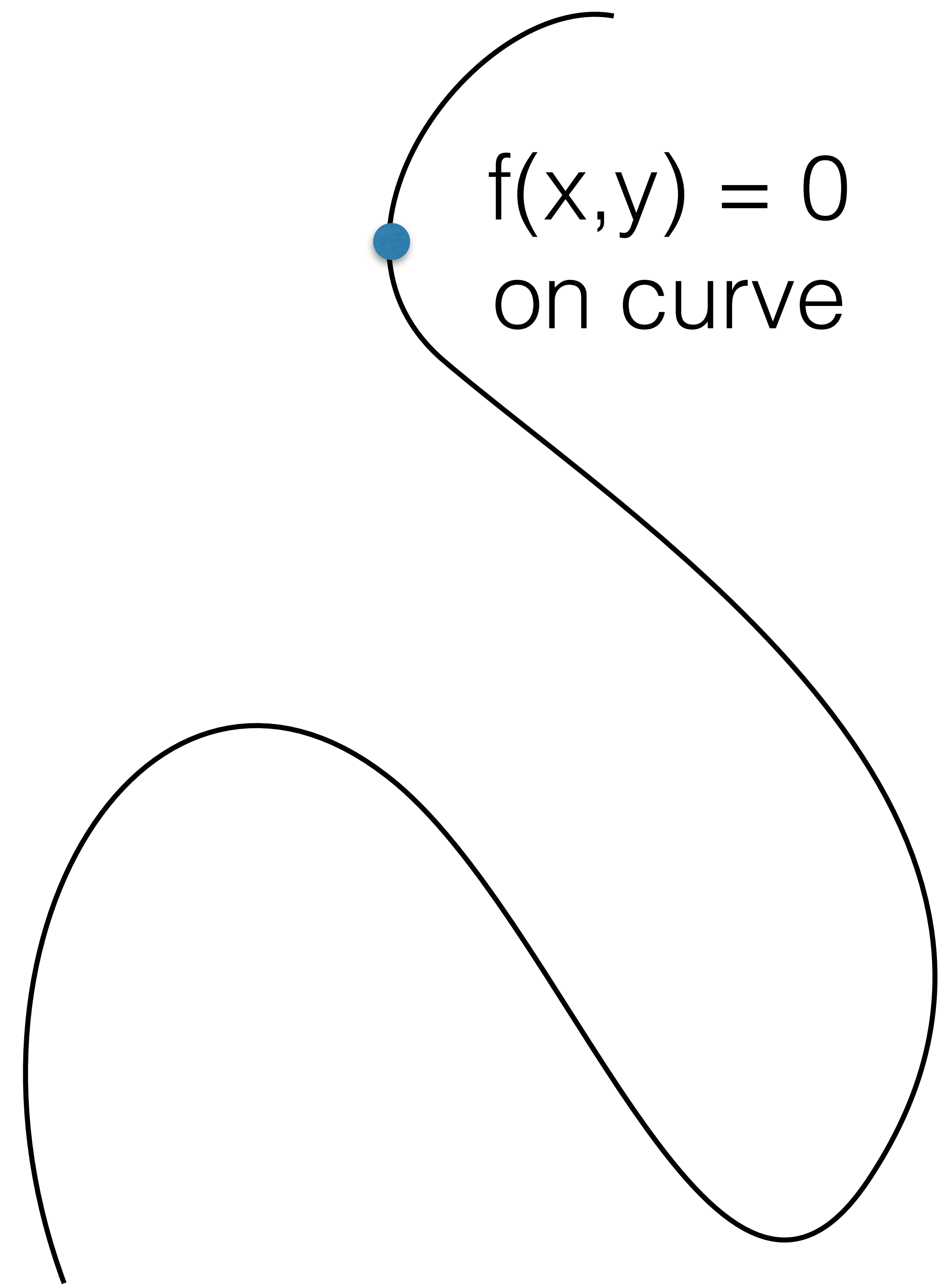


How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$

test if (x,y) is on the curve

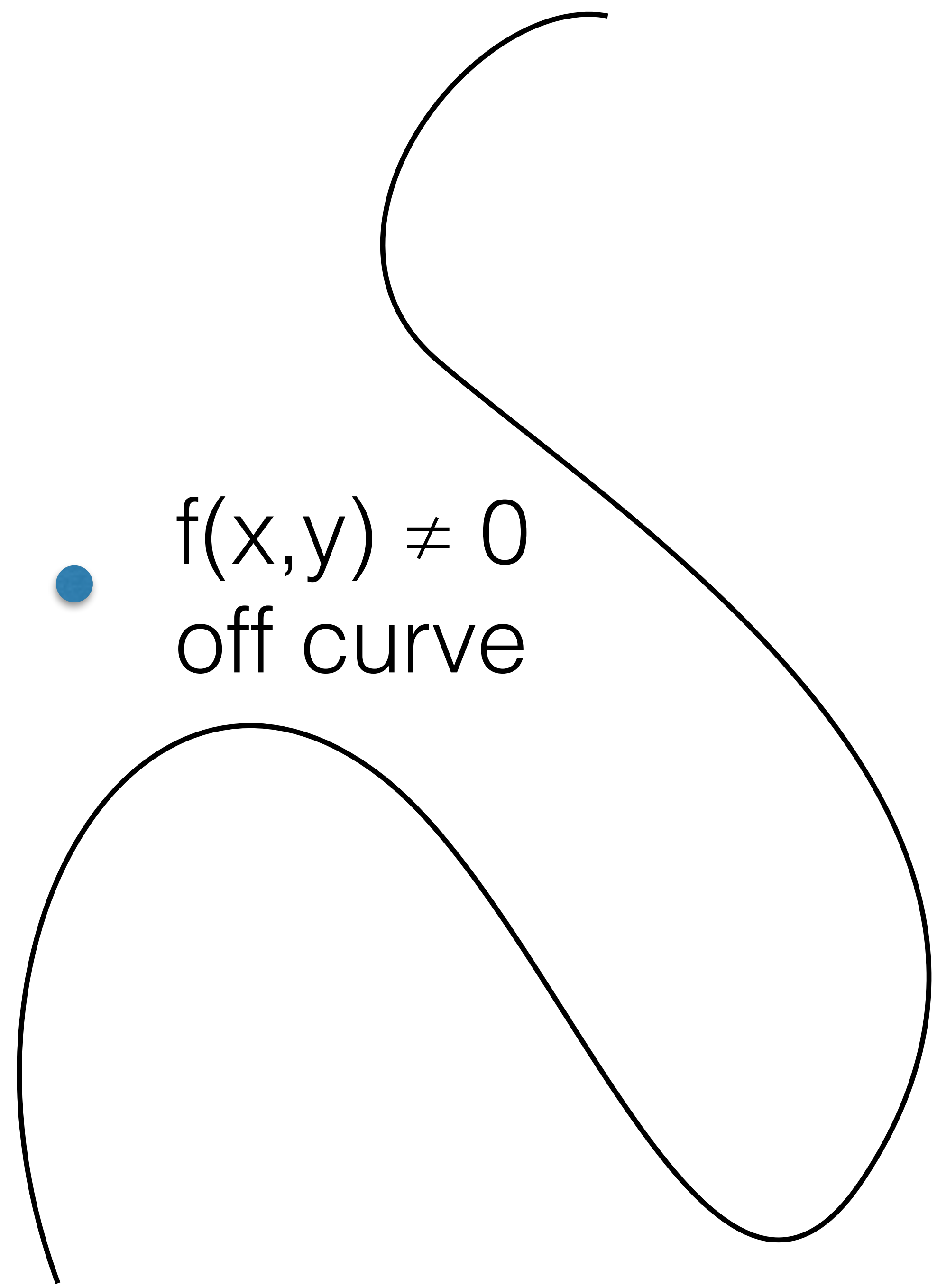


How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$

test if (x,y) is on the curve



How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$

test if (x,y) is on the curve

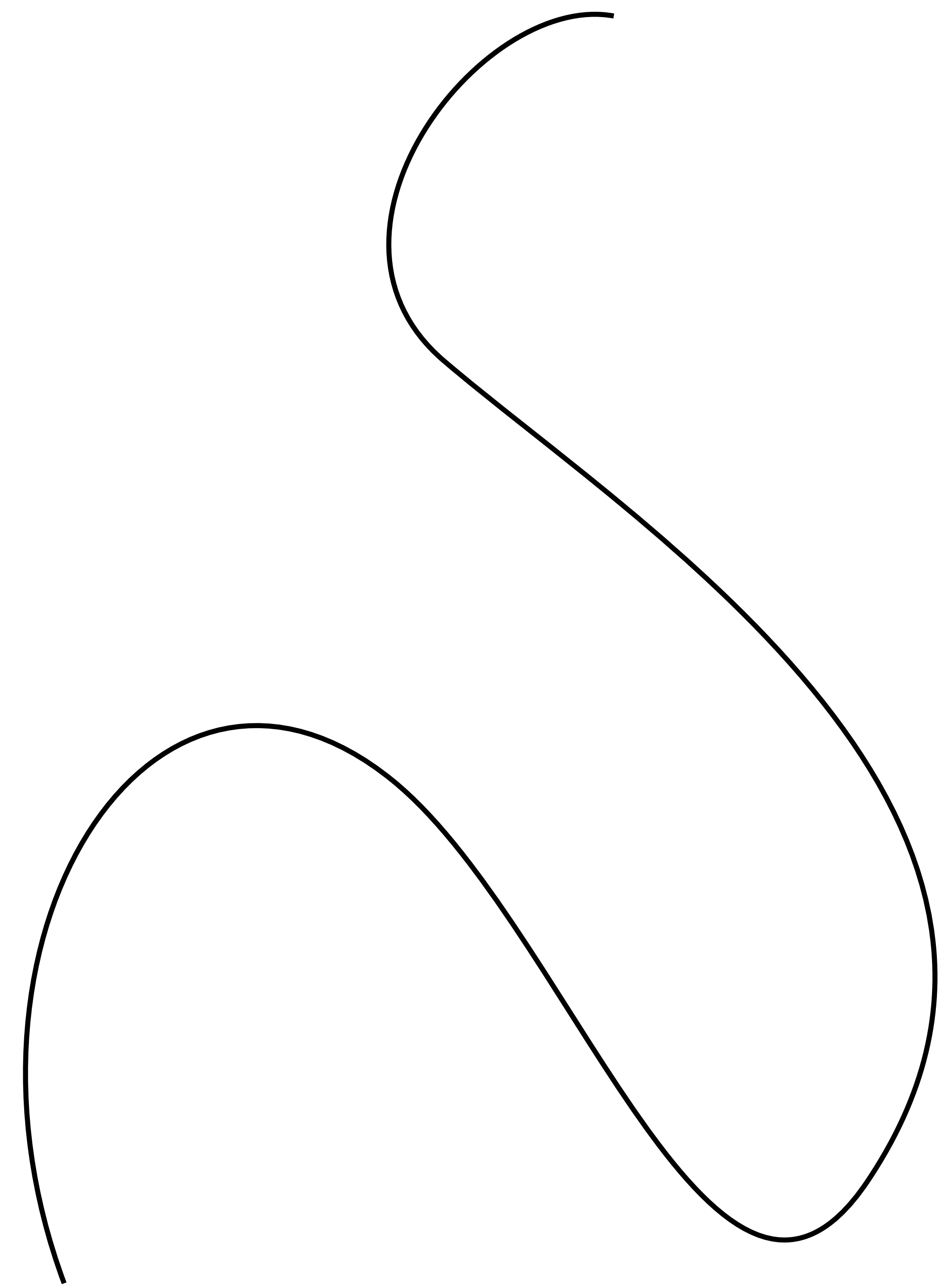
Parametric

(2D) $(x,y) = \mathbf{f}(t)$

(3D) $(x,y,z) = \mathbf{f}(t)$

map free *parameter* t

to points on the curve



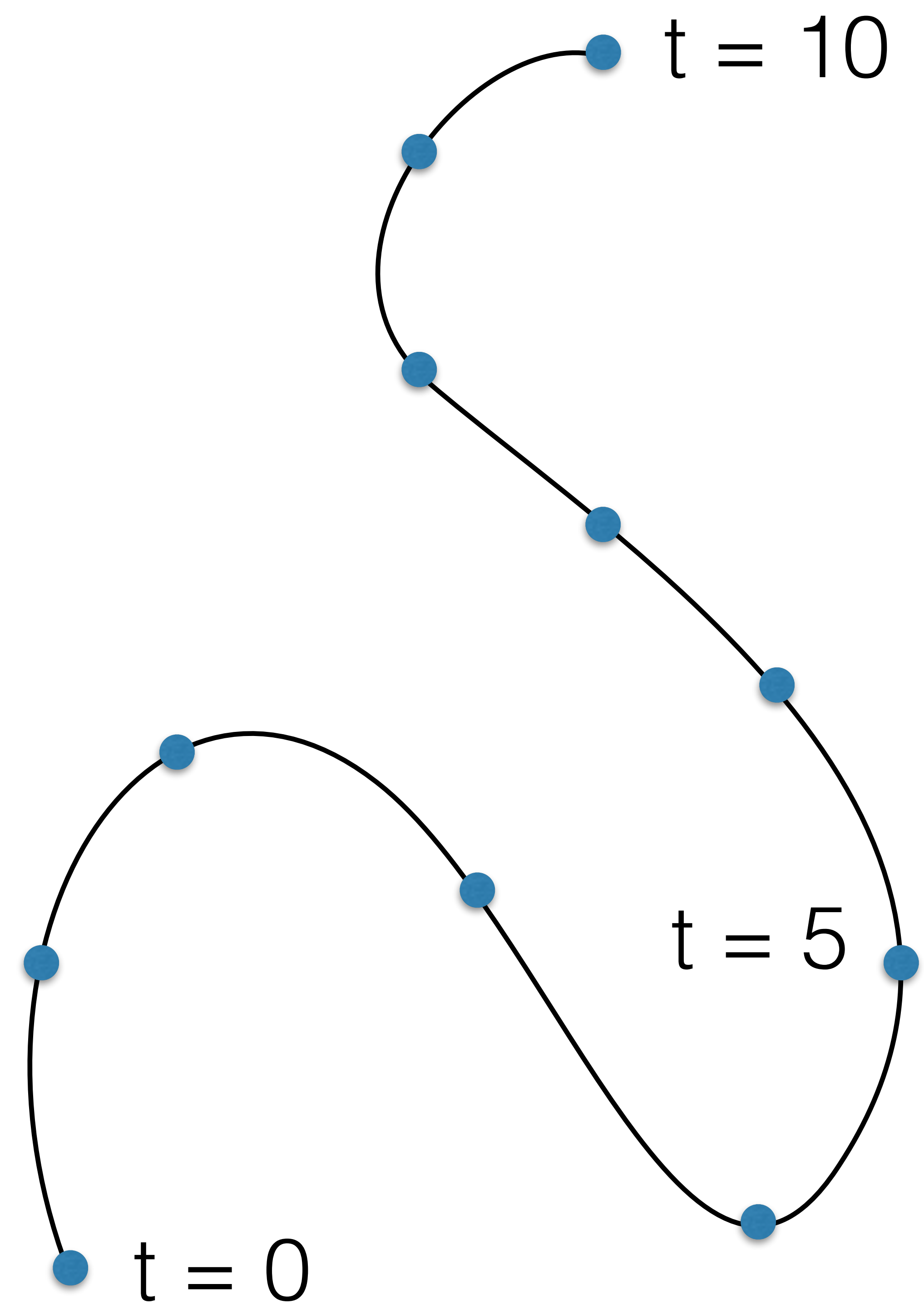
How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$
test if (x,y) is on the curve

Parametric

(2D) $(x,y) = \mathbf{f}(t)$
(3D) $(x,y,z) = \mathbf{f}(t)$
map free *parameter* t
to points on the curve



How do we specify a curve?

Implicit

(2D) f(x,y) = 0

test if (x,y) is on the curve

Parametric

(2D) (x,y) = **f**(t)

(3D) (x,y,z) = **f**(t)

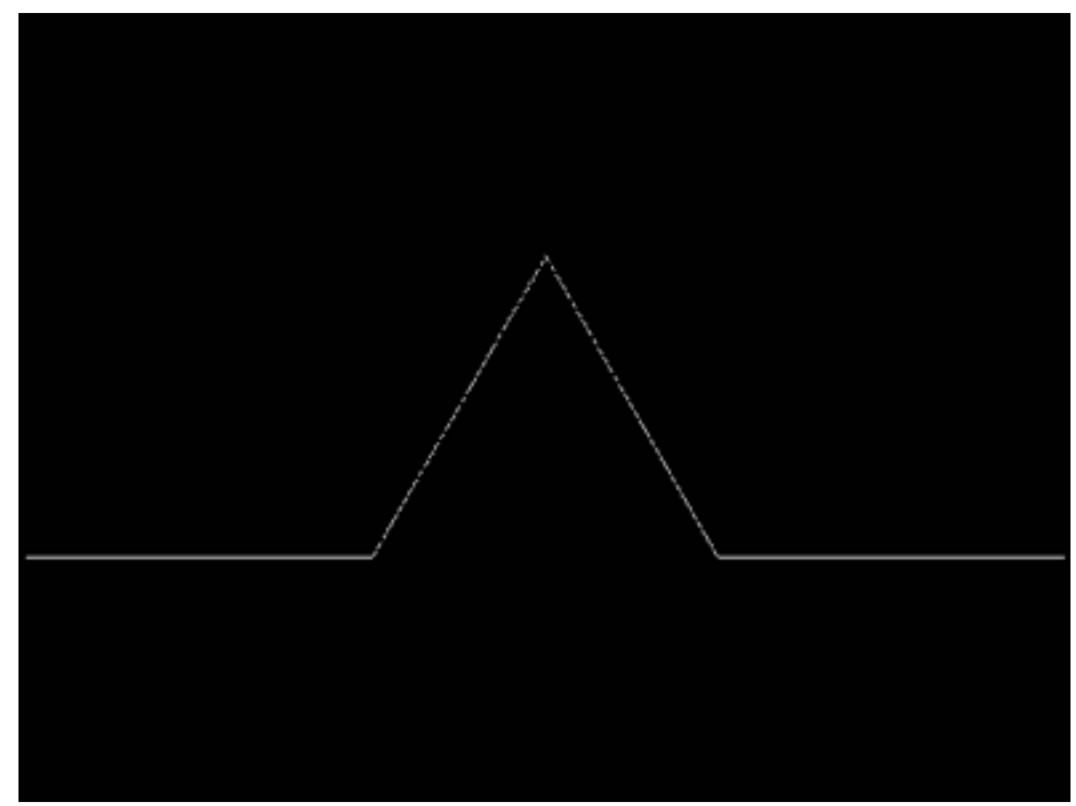
map free *parameter* t

to points on the curve

Procedural

e.g., fractals,

subdivision schemes



[George Reese]

Fractal: Koch Curve

How do we specify a curve?

Implicit

(2D) $f(x,y) = 0$

test if (x,y) is on the curve

Parametric

(2D) $(x,y) = \mathbf{f}(t)$

(3D) $(x,y,z) = \mathbf{f}(t)$

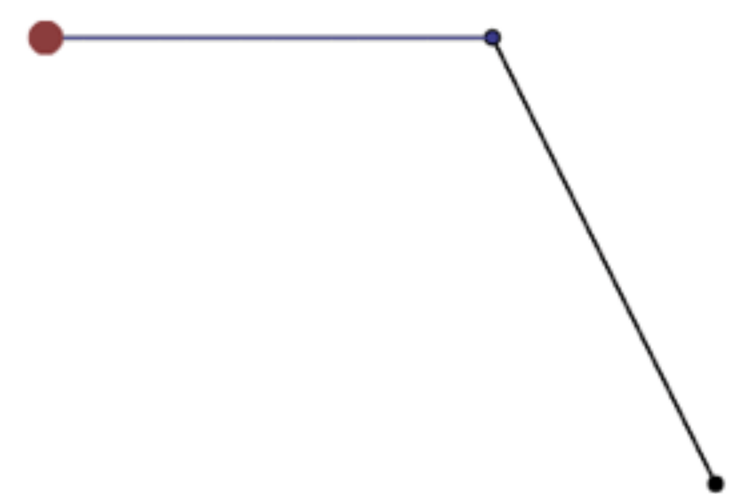
map free *parameter* t

to points on the curve

Procedural

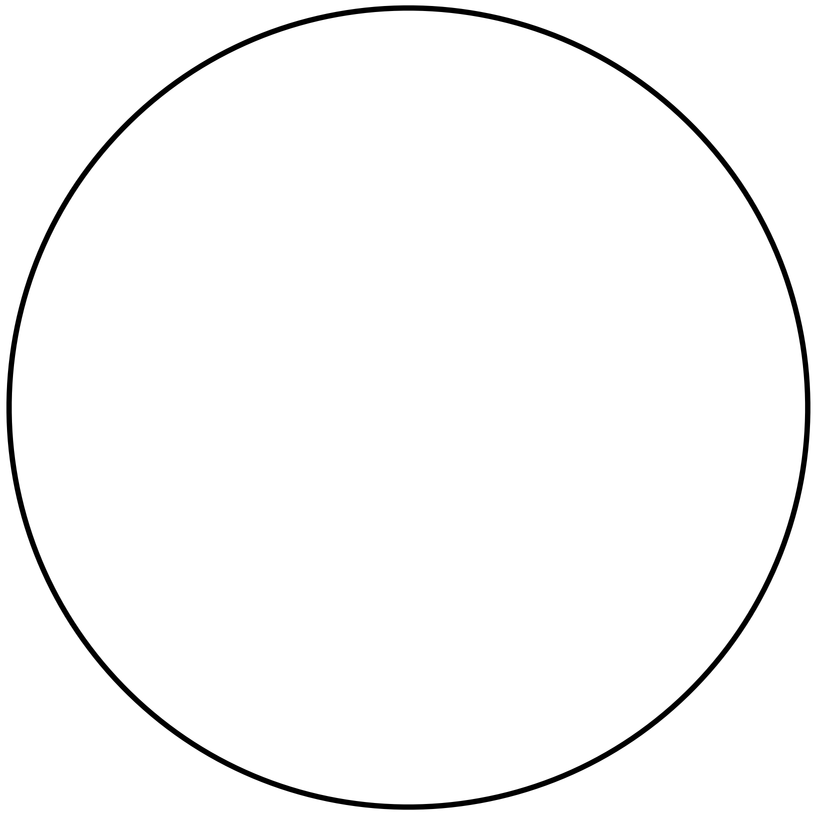
e.g., fractals,

subdivision schemes



Bezier Curve

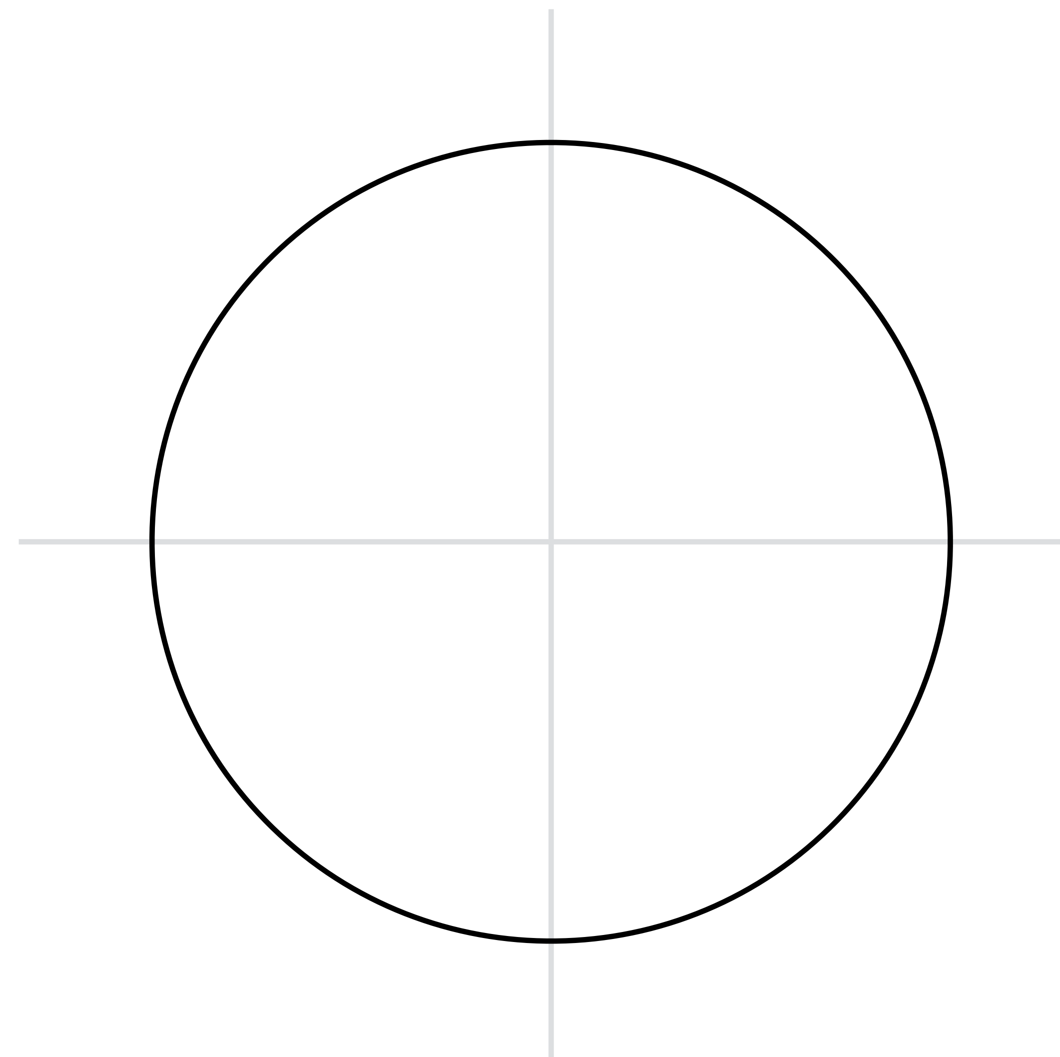
A curve may have multiple representations



A curve may have multiple representations

Implicit

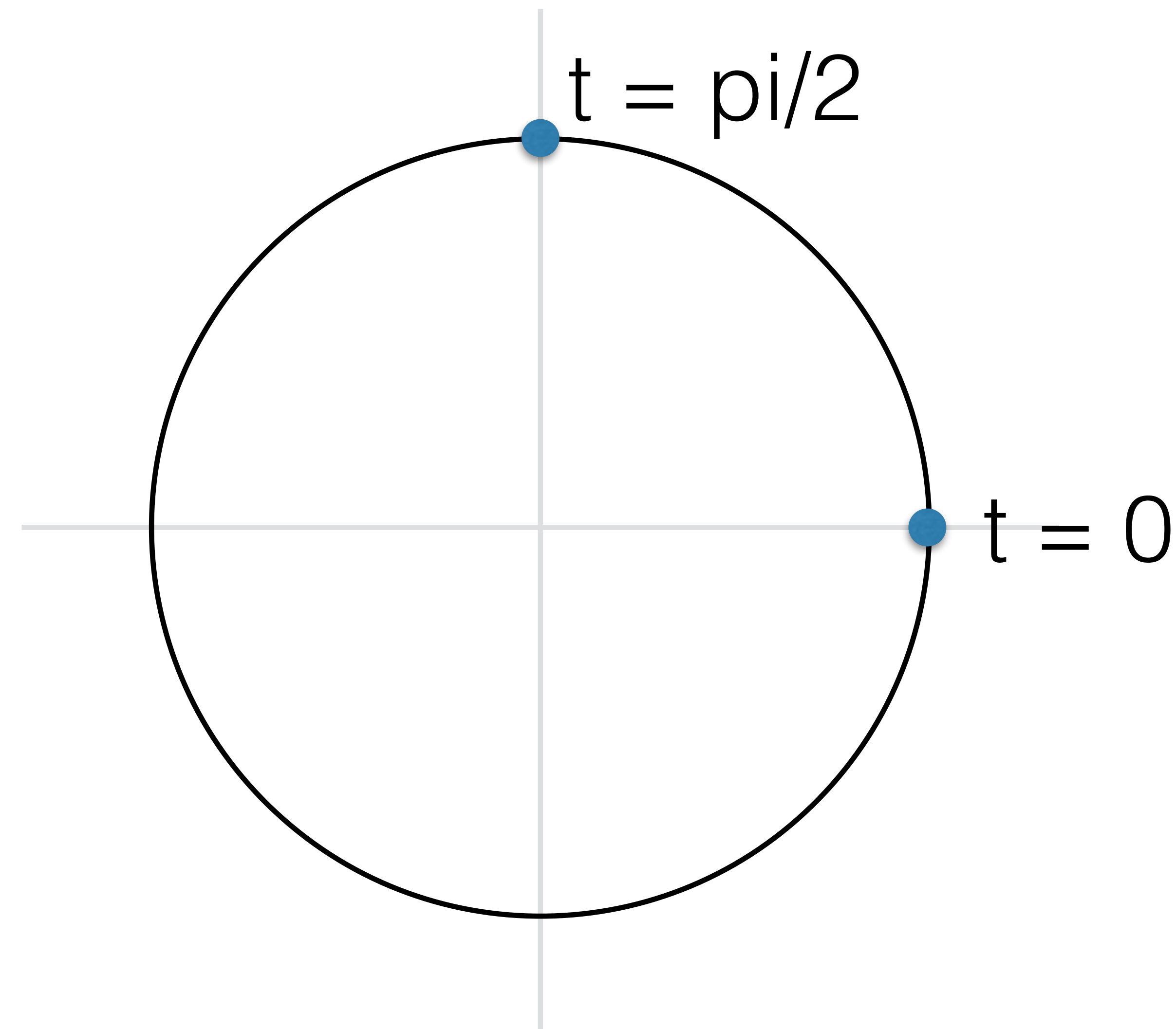
$f(x,y) = x^2 + y^2 - 1 = 0$



A curve may have multiple representations

Parametric

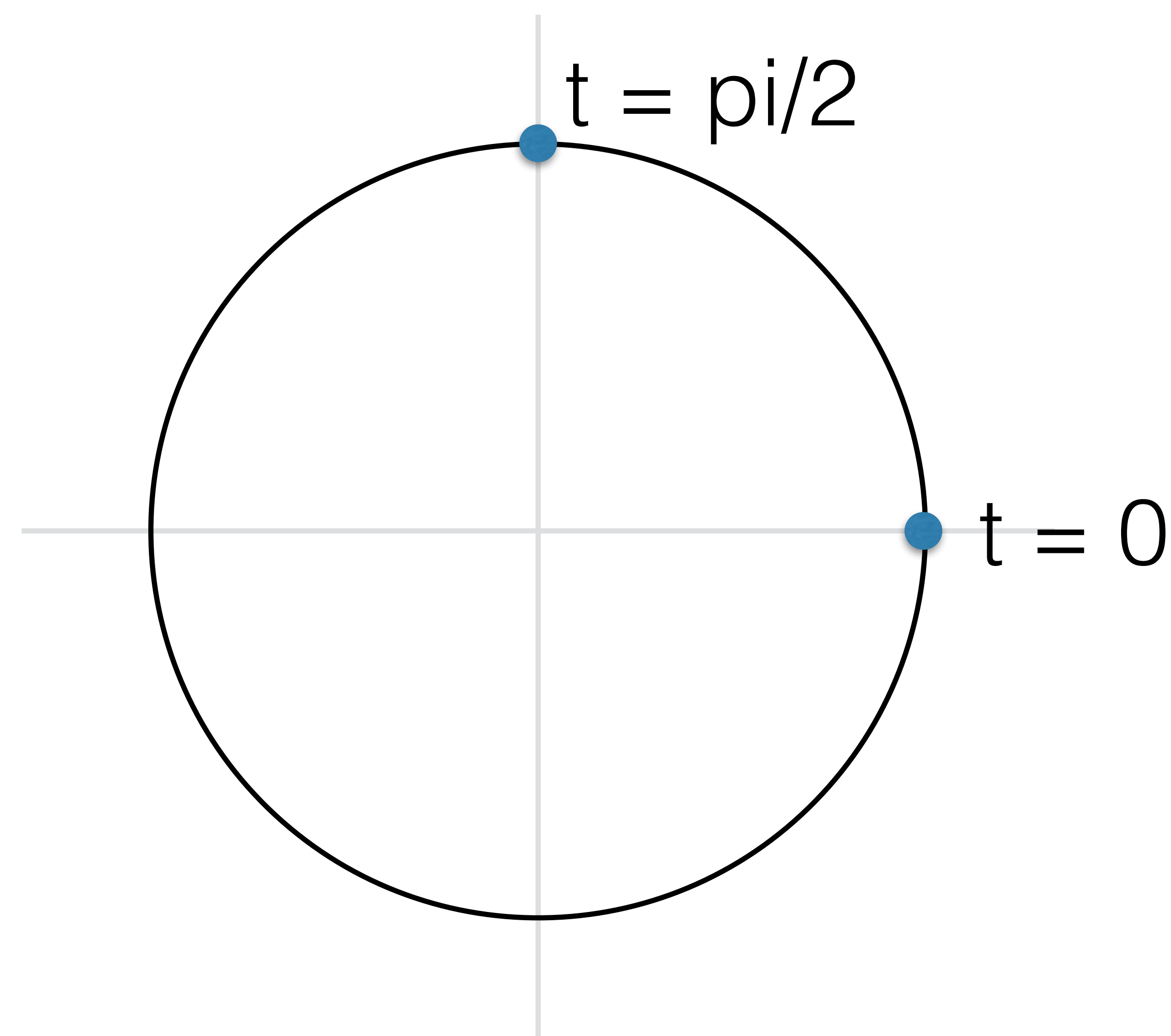
$(x,y) = \mathbf{f}(t) = (\cos t, \sin t)$



A curve may have multiple representations

Parametric

$(x,y) = \mathbf{f}(t) = (\cos t, \sin t)$,
t in $[0,2\pi)$

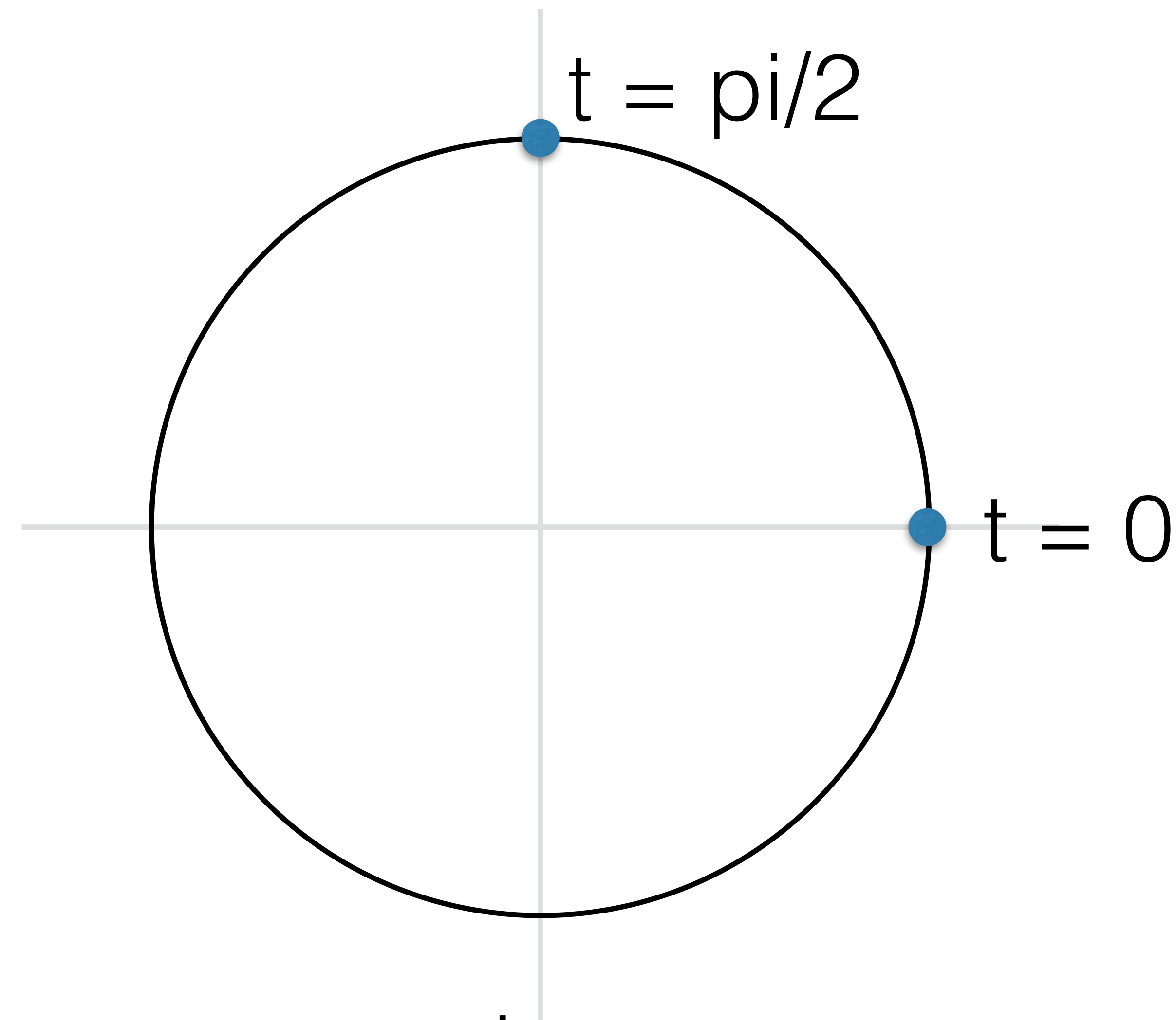


Same curve (set of points),
but different mathematical representation!

A curve may have multiple representations

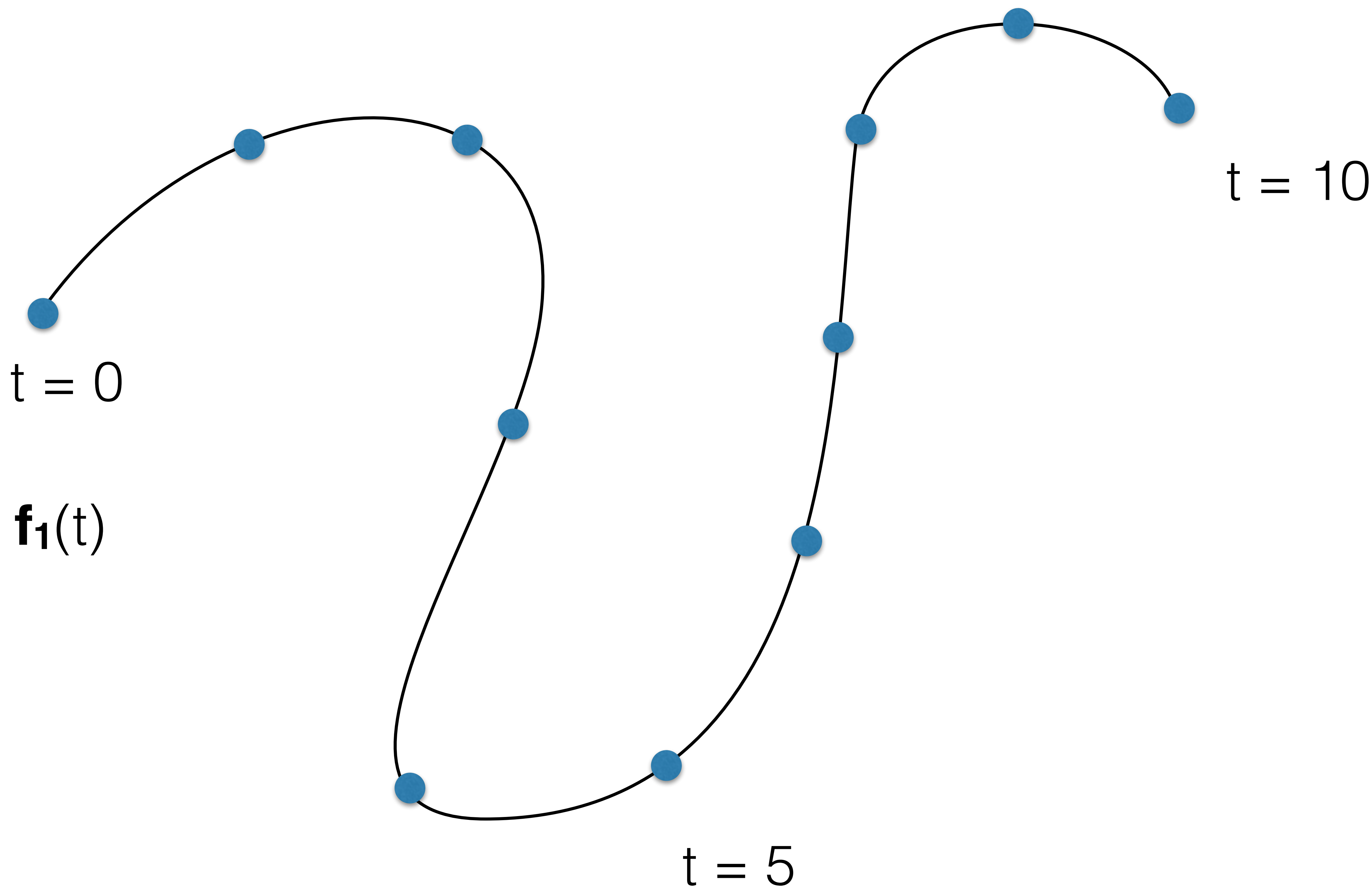
Parametric

$(x,y) = \mathbf{f}(t) = (\cos t, \sin t)$,
$t \in [0, 2\pi)$

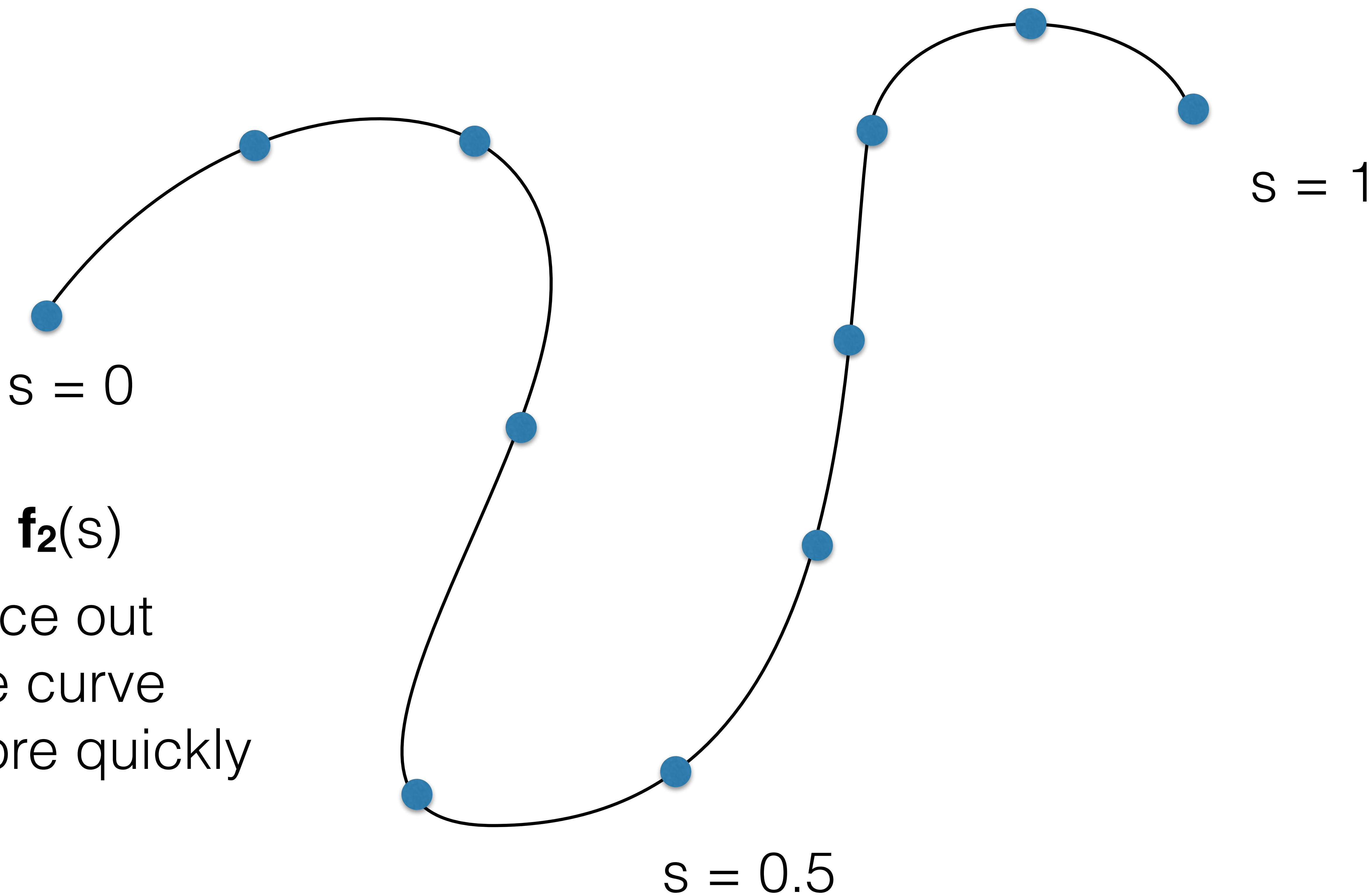


We will focus on parametric representations

Parameterization, re-parameterization



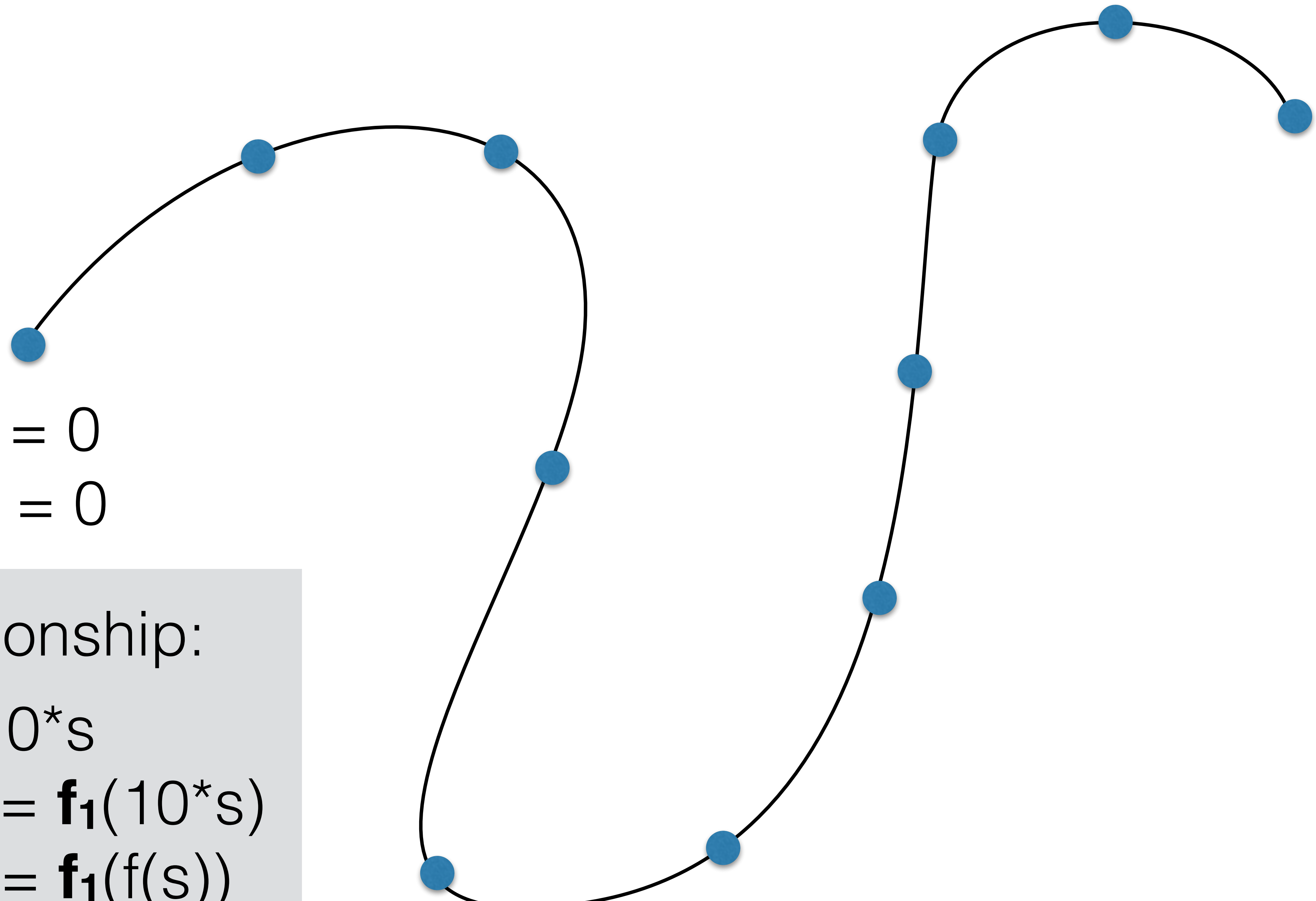
Parameterization, re-parameterization



$\mathbf{f_2}(s)$

trace out
the curve
more quickly

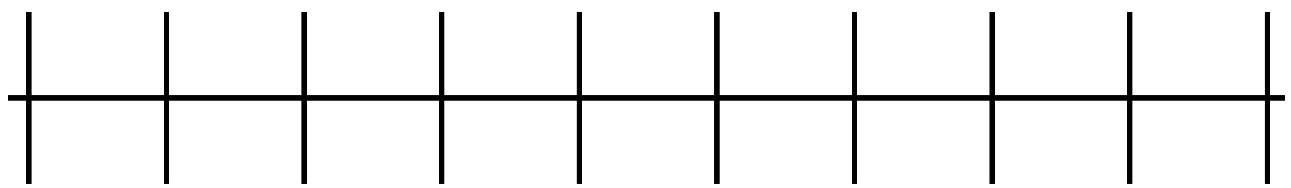
Parameterization, re-parameterization



relationship:

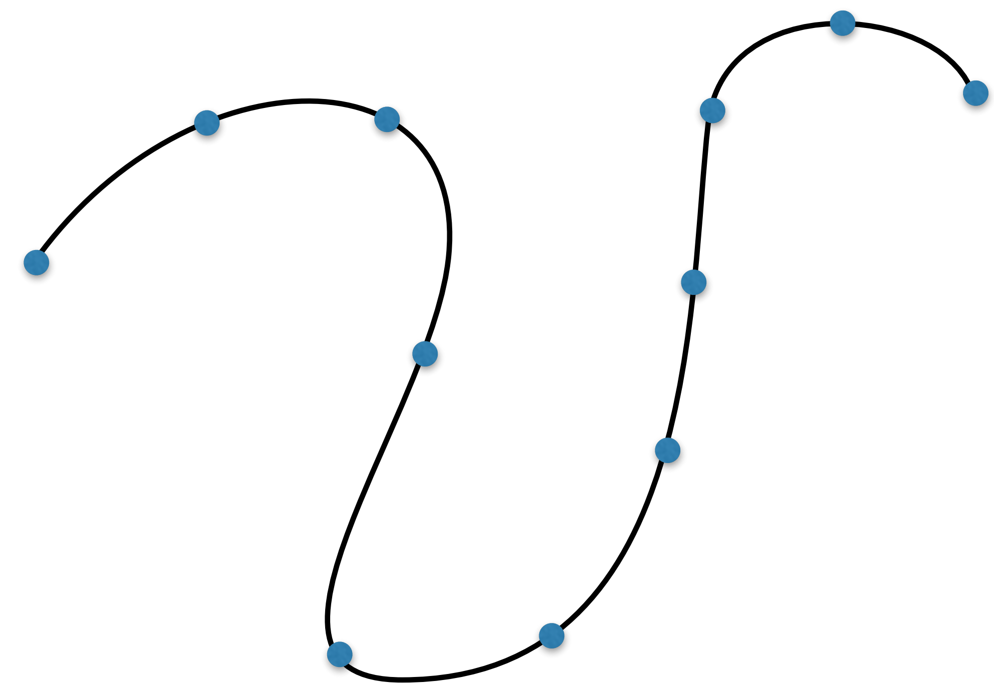
$$
\begin{aligned}
t &= 10*s \\
\mathbf{f_1}(t) &= \mathbf{f_1}(10*s) \\
&= \mathbf{f_1}(f(s)) \\
&= \mathbf{f_2}(s)
\end{aligned}
$$

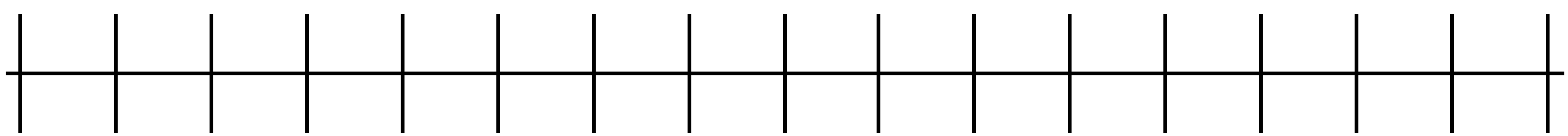
Parameterization, re-parameterization



$t = 0$ $t = 10$

$\mathbf{f_1}(t)$

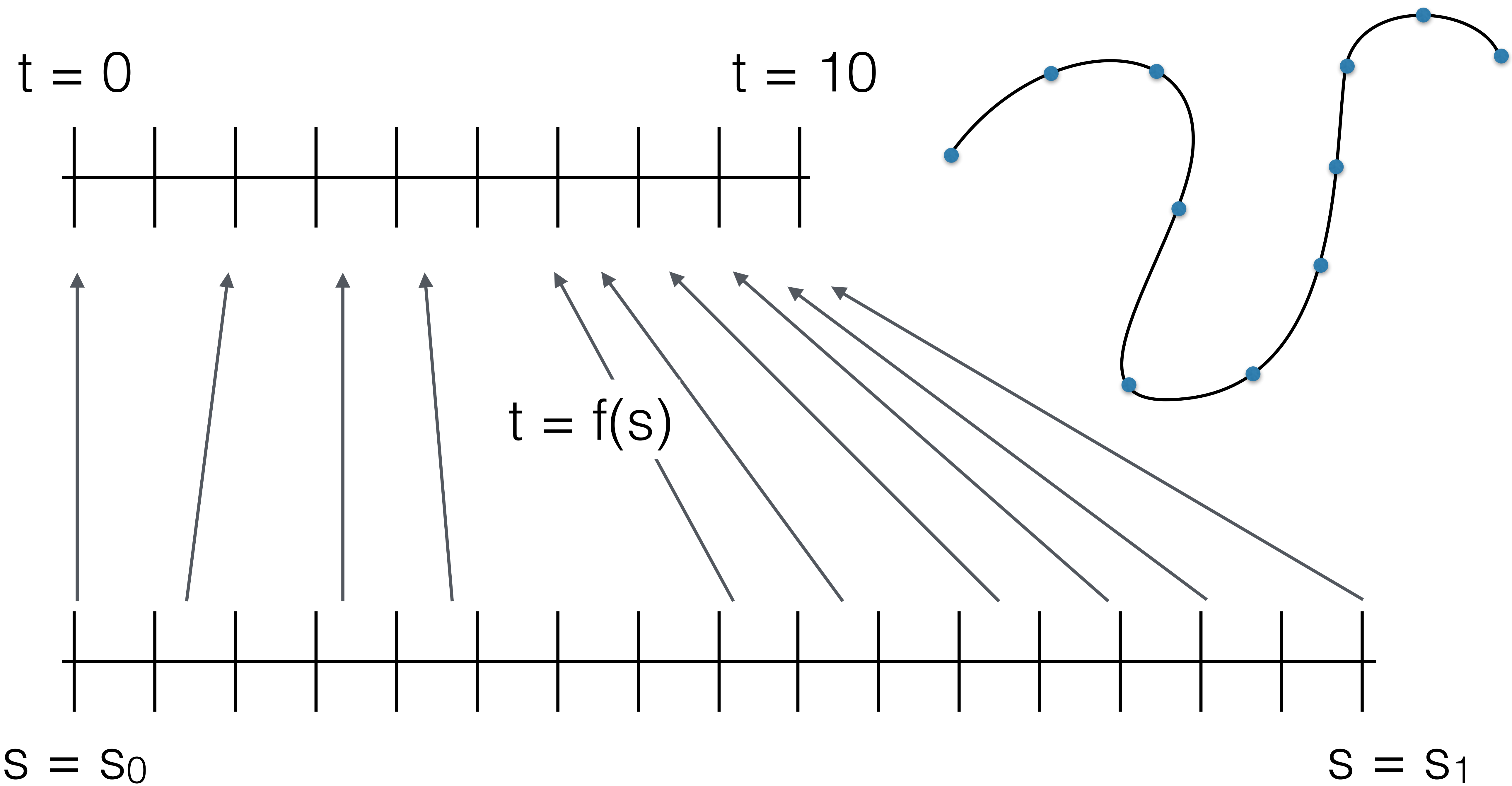




$s = s_0$ $s = s_1$

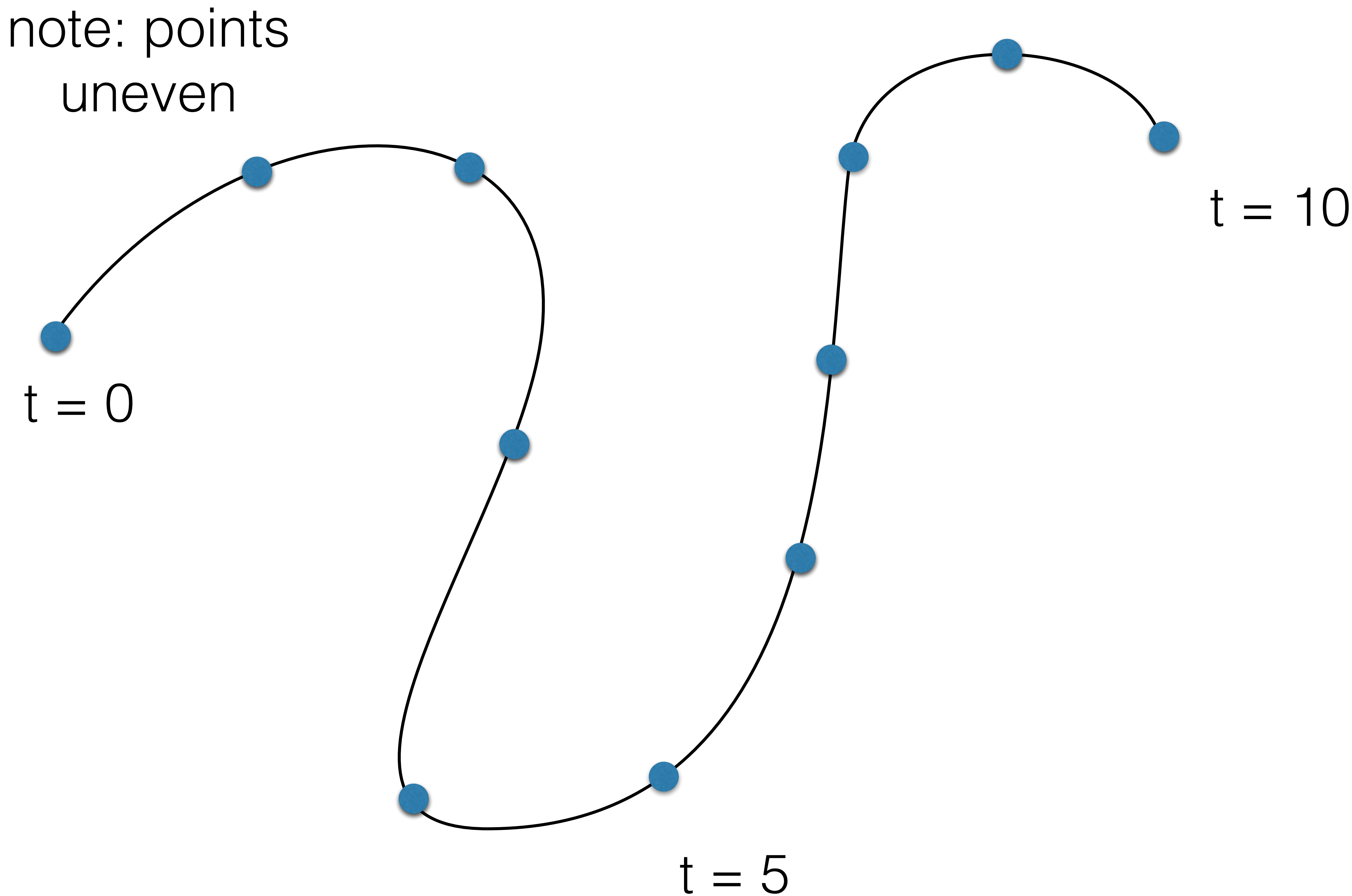
$\mathbf{f_2}(s) = \mathbf{f_1}(f(s))$

Parameterization, re-parameterization



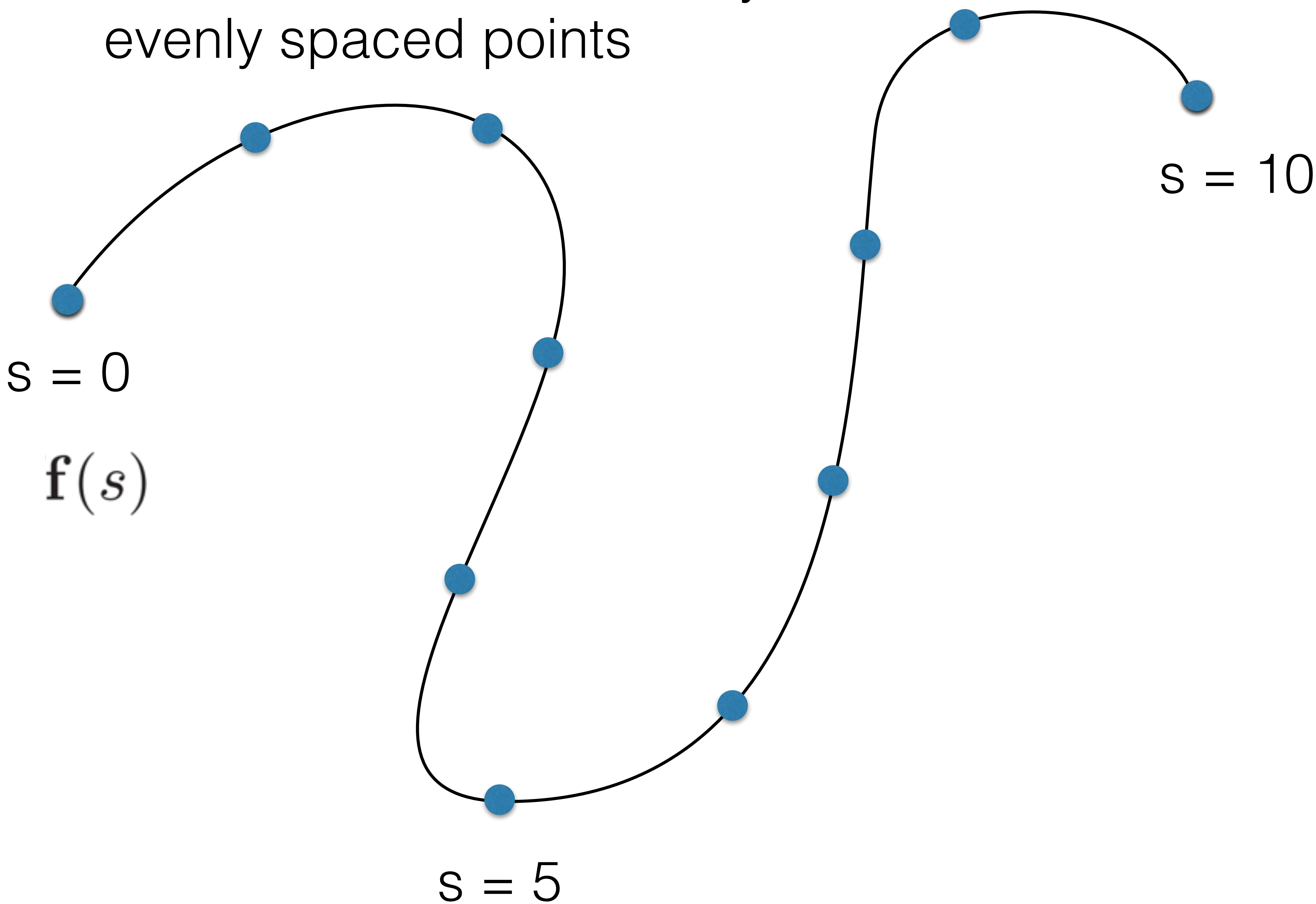
$\mathbf{f_2}(s) = \mathbf{f_1}(f(s))$

Natural parameterization



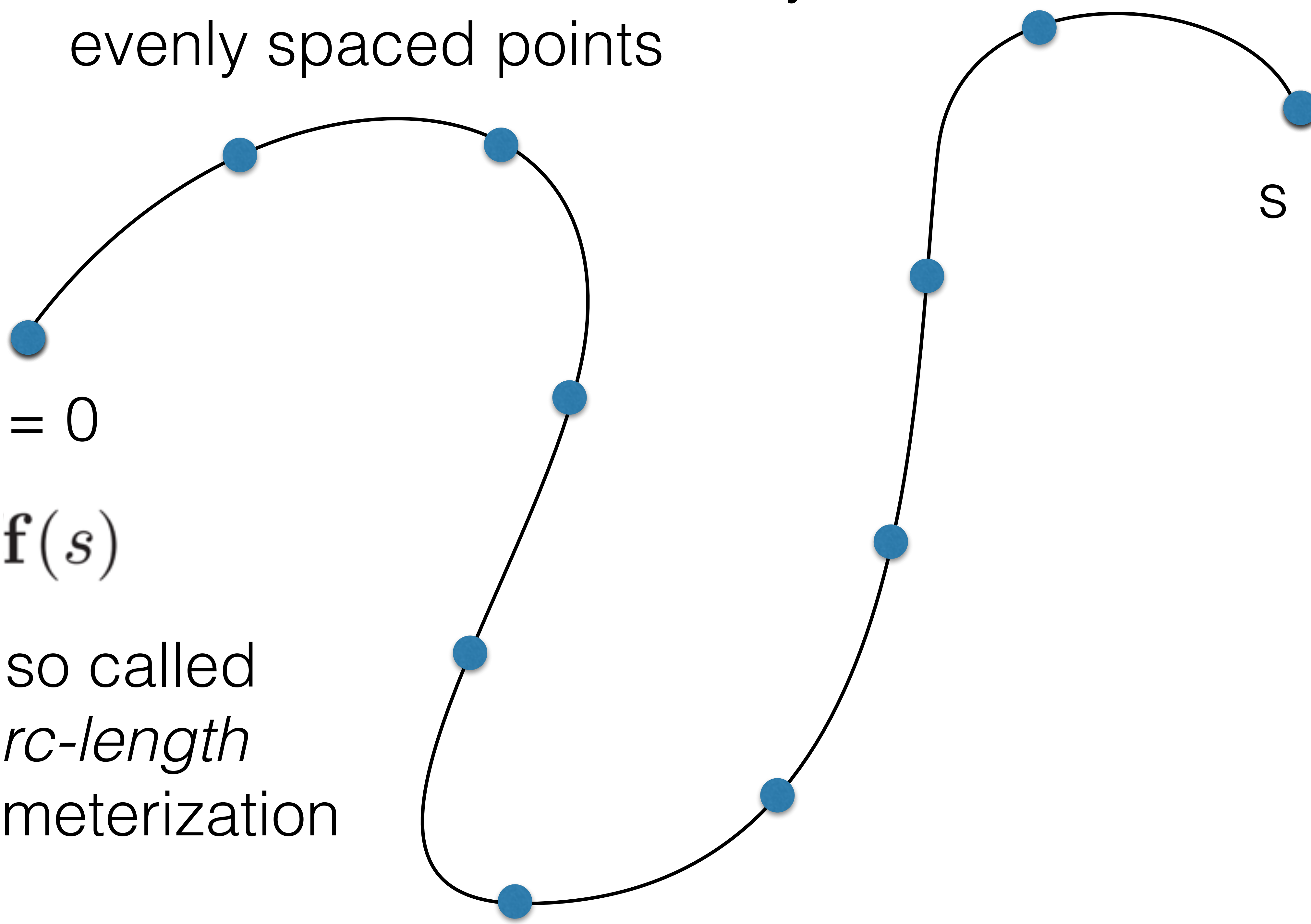
Natural parameterization

pen moves at a constant velocity:
evenly spaced points



Natural parameterization

pen moves at a constant velocity:
evenly spaced points

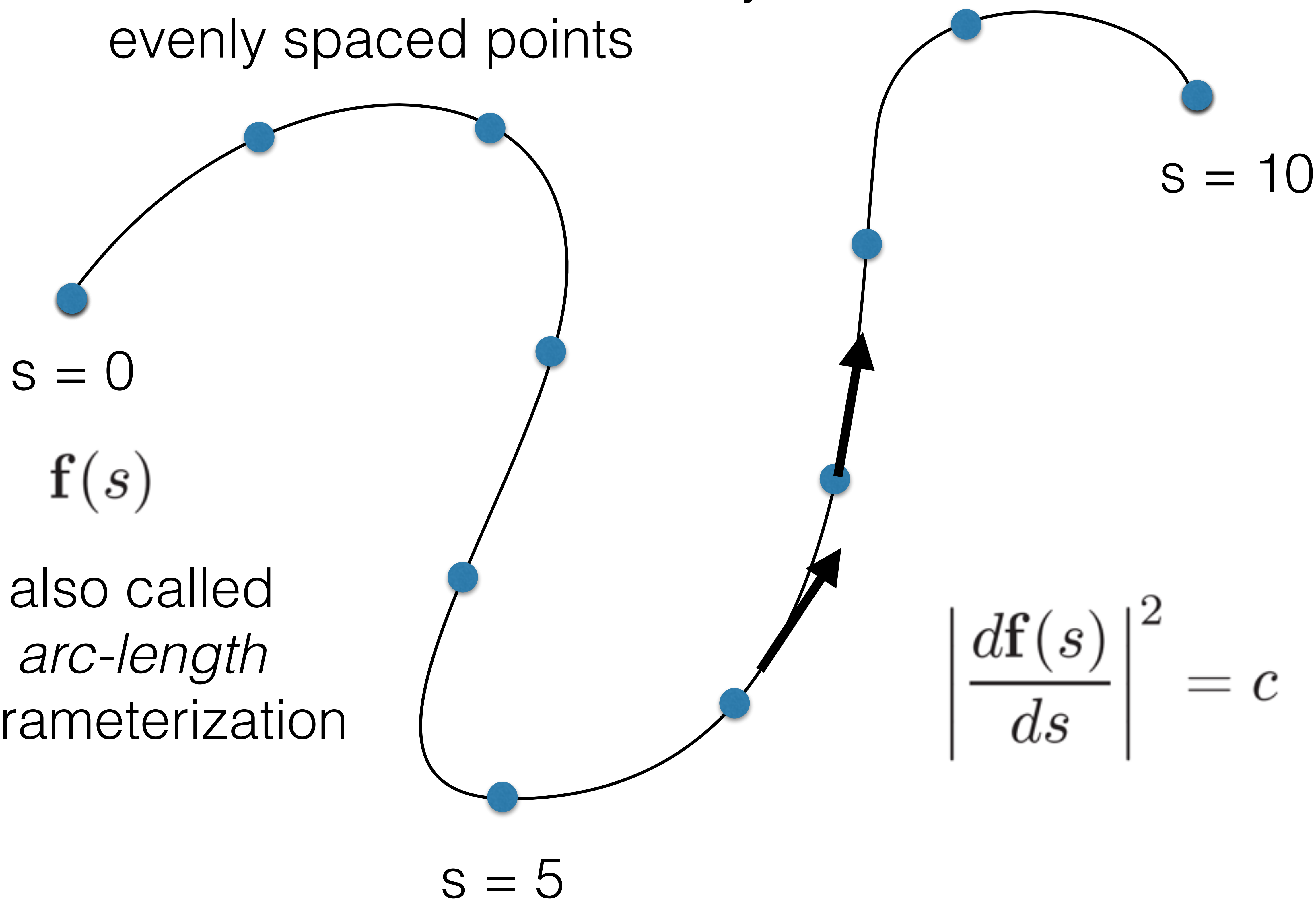


$\mathbf{f}(s)$

also called
arc-length
parameterization

Natural parameterization

pen moves at a constant velocity:
evenly spaced points



$\mathbf{f}(s)$

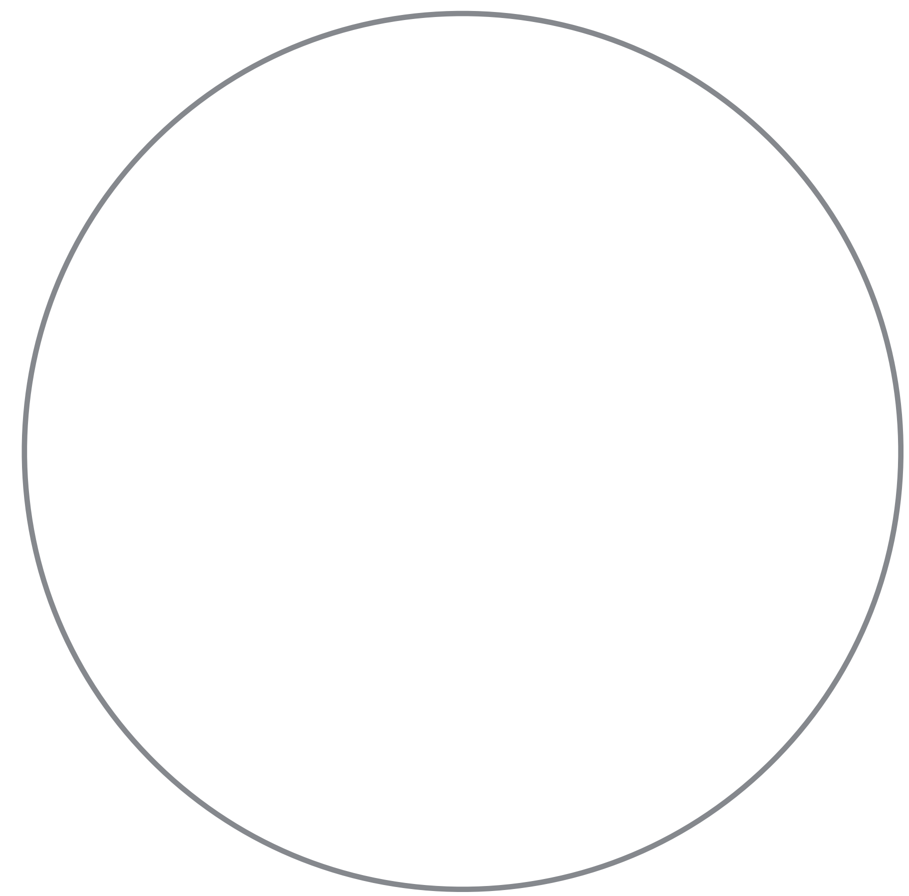
also called
arc-length
parameterization

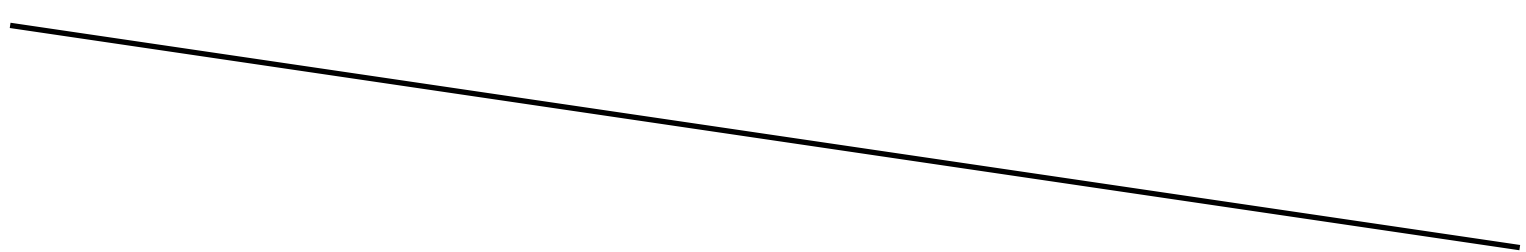
$$\left|\frac{d\mathbf{f}(s)}{ds}\right|^2 = c$$

piecewise parametric representation

sometimes easy
to find a parametric
representation

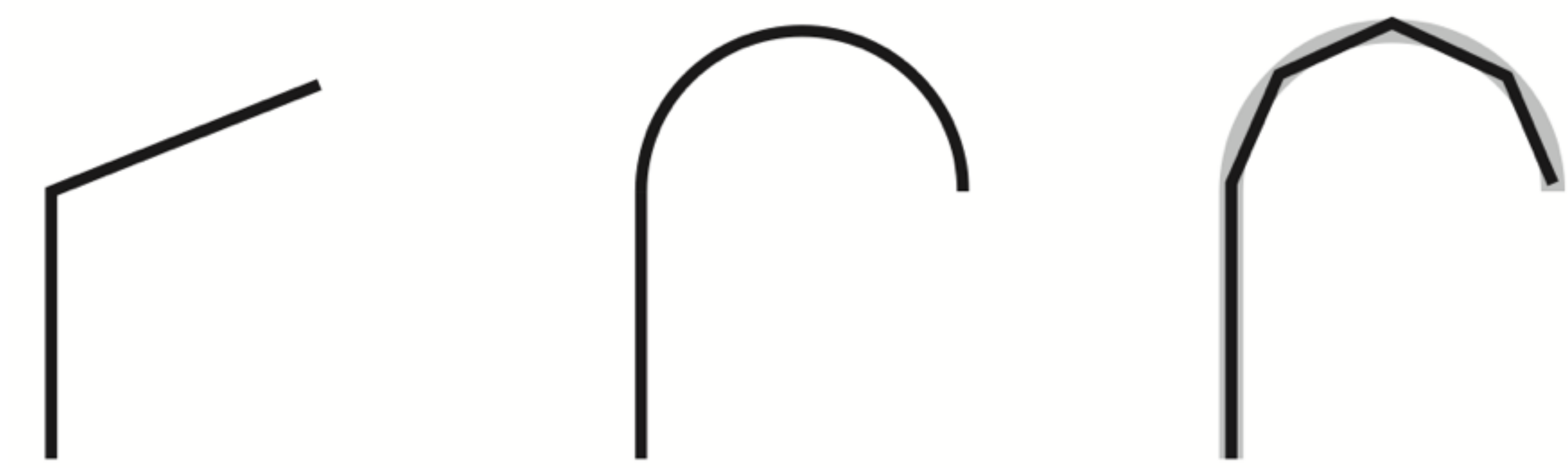
e.g., circle, line segment





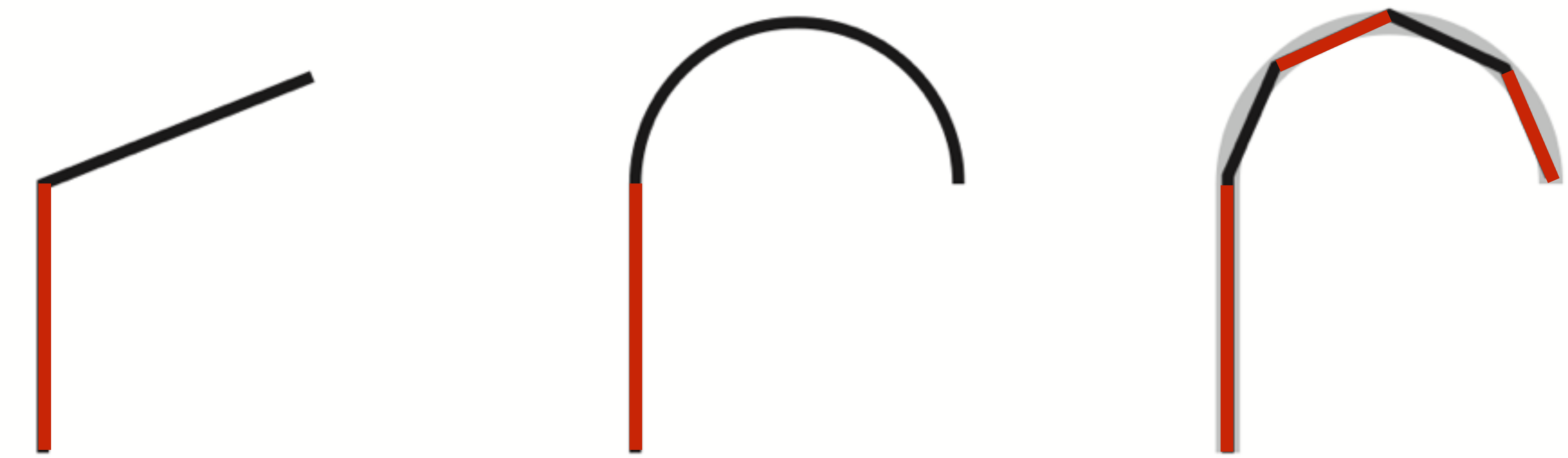
piecewise parametric representation

in other cases, not obvious



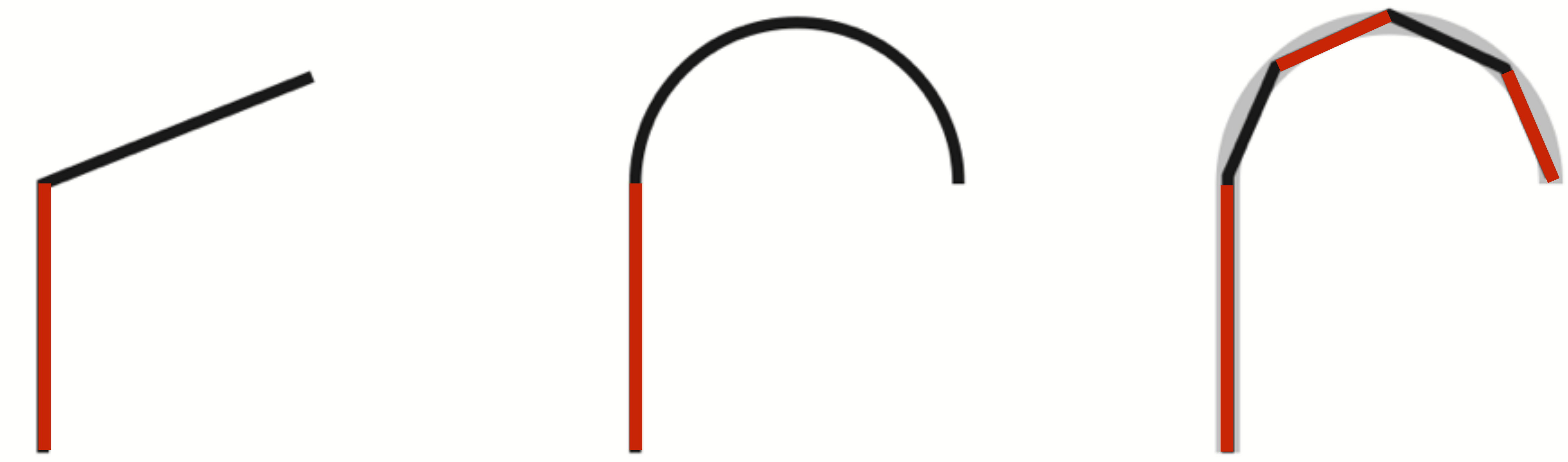
piecewise parametric representation

strategy: break into simpler pieces



piecewise parametric representation

strategy: break into simpler pieces

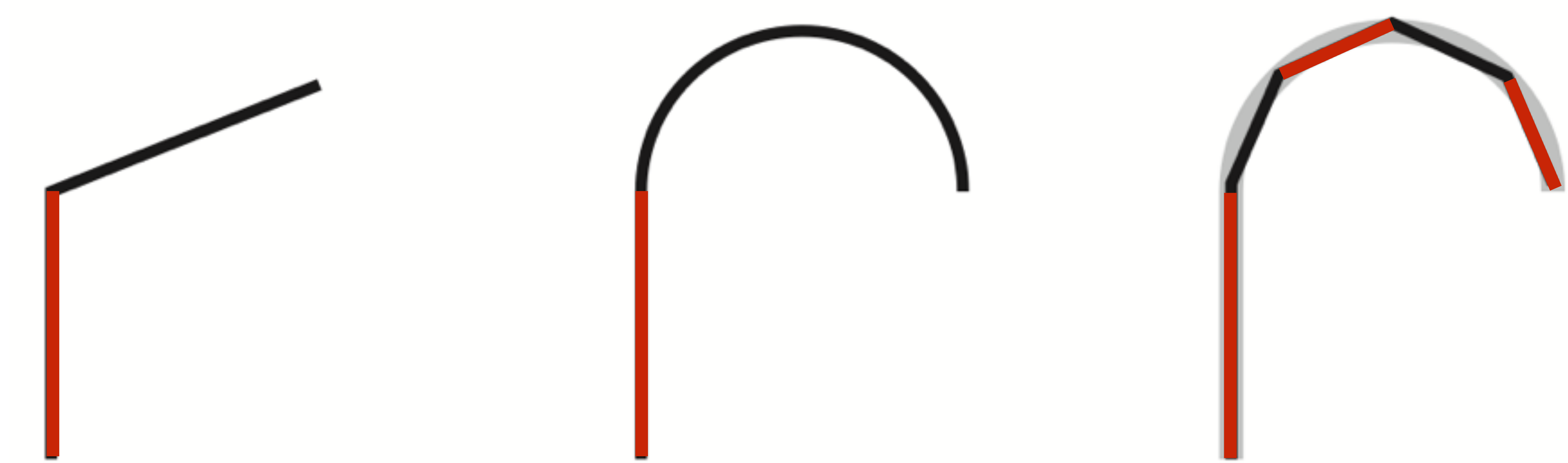


switch between functions that represent pieces:

$$\mathbf{f}(u) = \begin{cases} \mathbf{f}_1(2u) & u \le 0.5 \\ \mathbf{f}_2(2u-1) & u > 0.5 \end{cases}$$

piecewise parametric representation

strategy: break into simpler pieces



switch between functions that represent pieces:

$$\mathbf{f}(u) = \begin{cases} \mathbf{f}_1(2u) & u \leq 0.5 \\ \mathbf{f}_2(2u-1) & u > 0.5 \end{cases}$$

map the inputs to $\mathbf{f}_1$ and $\mathbf{f}_2$ to be from 0 to 1

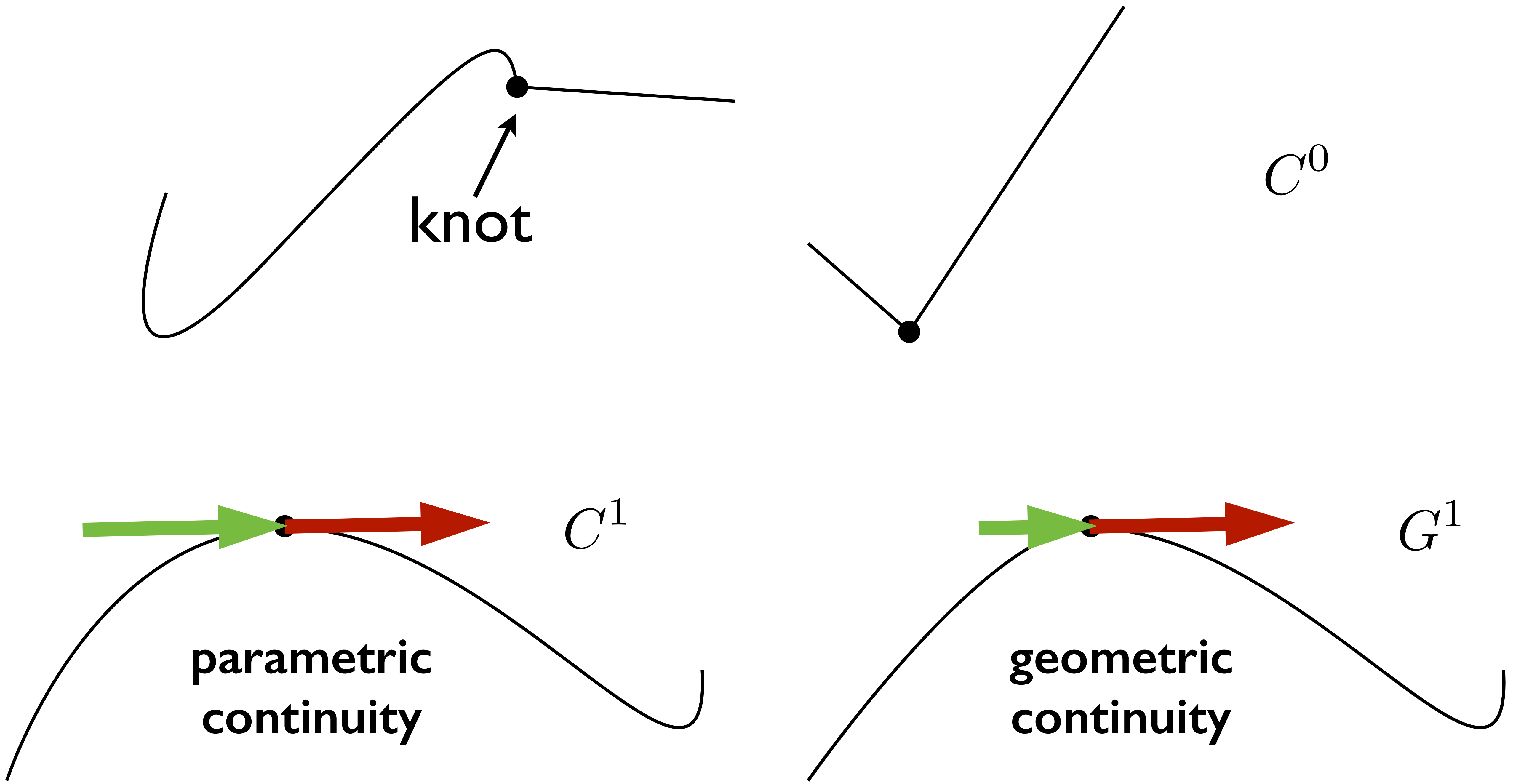
Curve Properties

Local properties:
- continuity
- position
- direction
- curvature

Global properties (examples):
- closed curve
- curve crosses itself

Interpolating vs. non-interpolating

Continuity: stitching curve segments together



Finding a Parametric Representation

Polynomial Pieces

<whiteboard>

Blending Functions

Blending functions are more convenient basis than monomial basis

- "canonical form" (monomial basis)

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

- "geometric form" (blending functions)

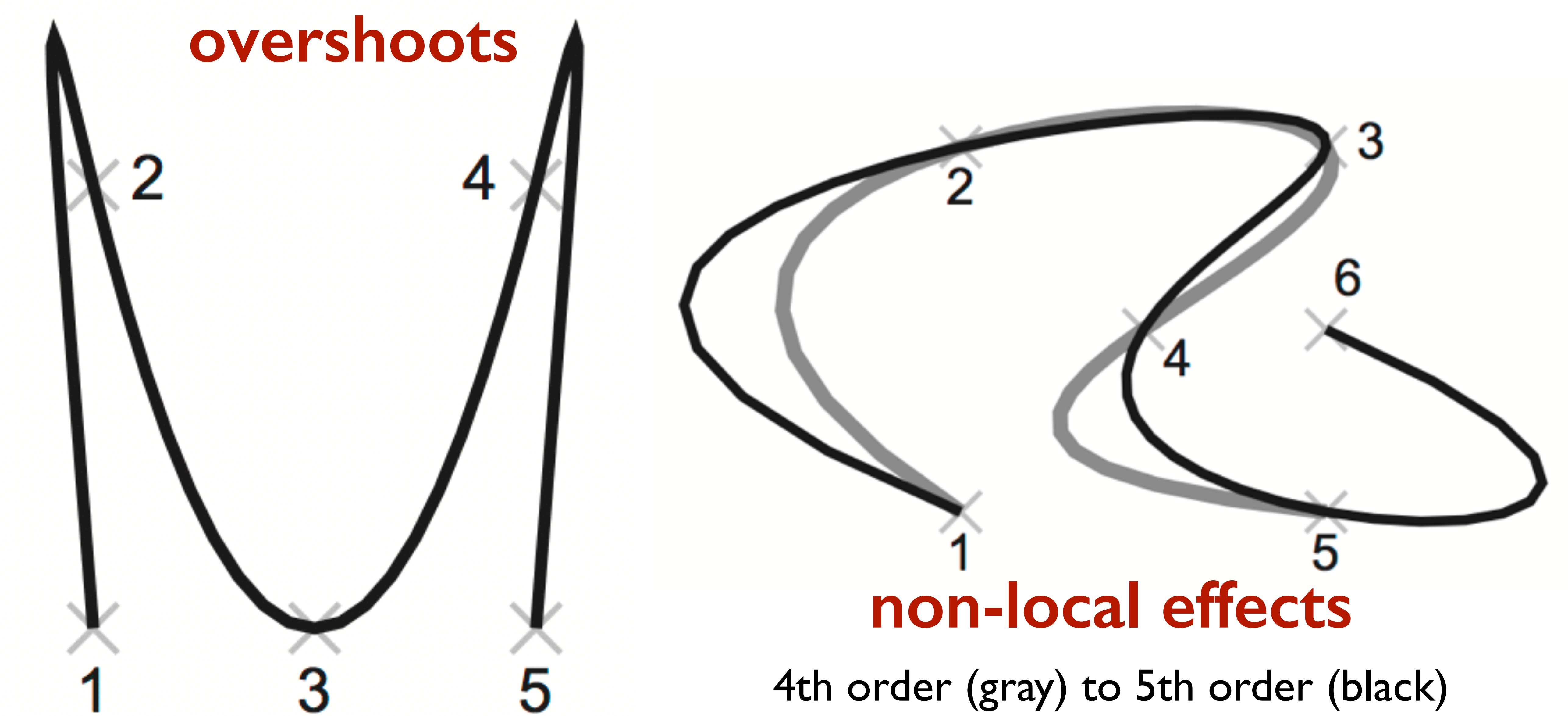
$$\mathbf{f}(u) = b_0(u)\mathbf{p}_0 + b_1(u)\mathbf{p}_1 + b_2(u)\mathbf{p}_2 + b_3(u)\mathbf{p}_3$$

Interpolating Polynomials

Interpolating polynomials

- Given n+1 data points, can find a unique interpolating polynomial of degree n
- Different methods:
  - Vandermonde matrix
  - Lagrange interpolation
  - Newton interpolation

higher order interpolating polynomials are rarely used

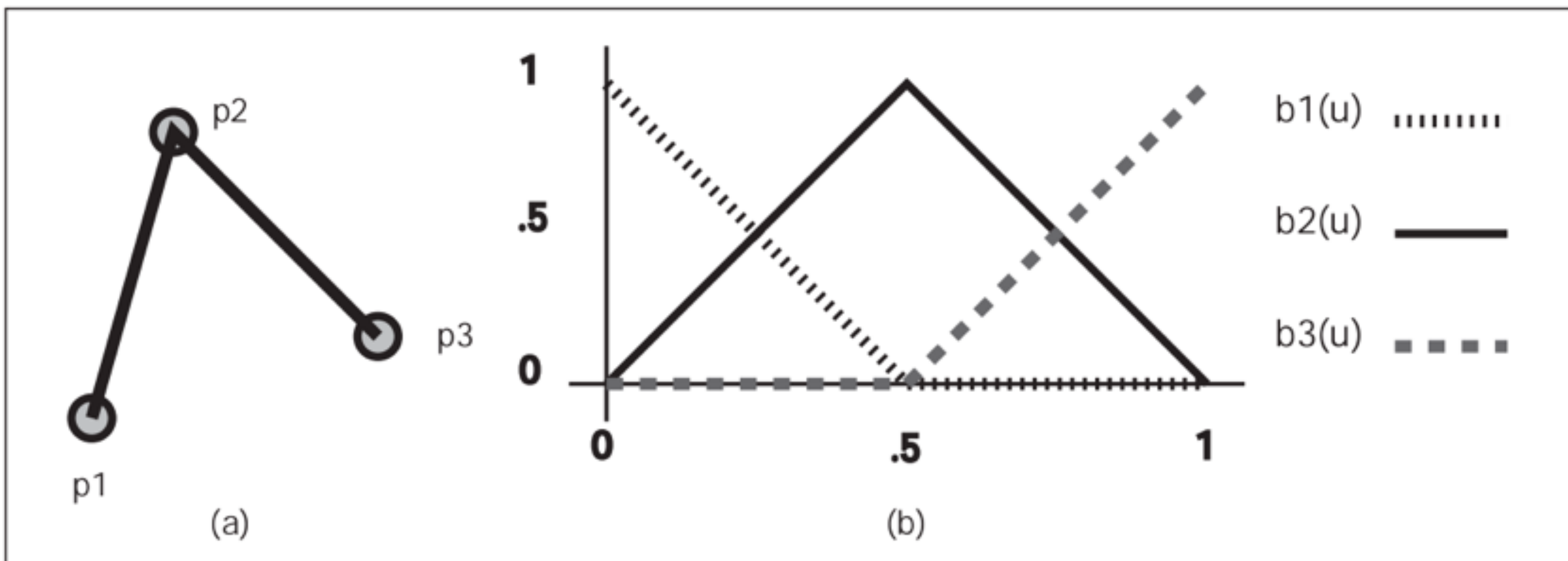


4th order (gray) to 5th order (black)

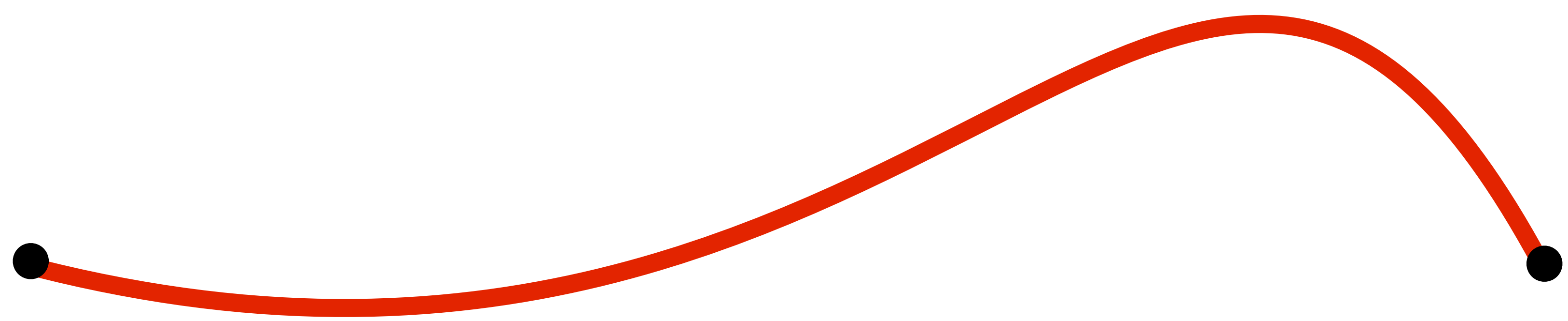
Piecewise Polynomial Curves

Example: blending functions for two line segments

$$\mathbf{f}(u) = \begin{cases} \mathbf{f}_1(2u) & u \leq 0.5 \\ \mathbf{f}_2(2u-1) & u > 0.5 \end{cases}$$



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

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

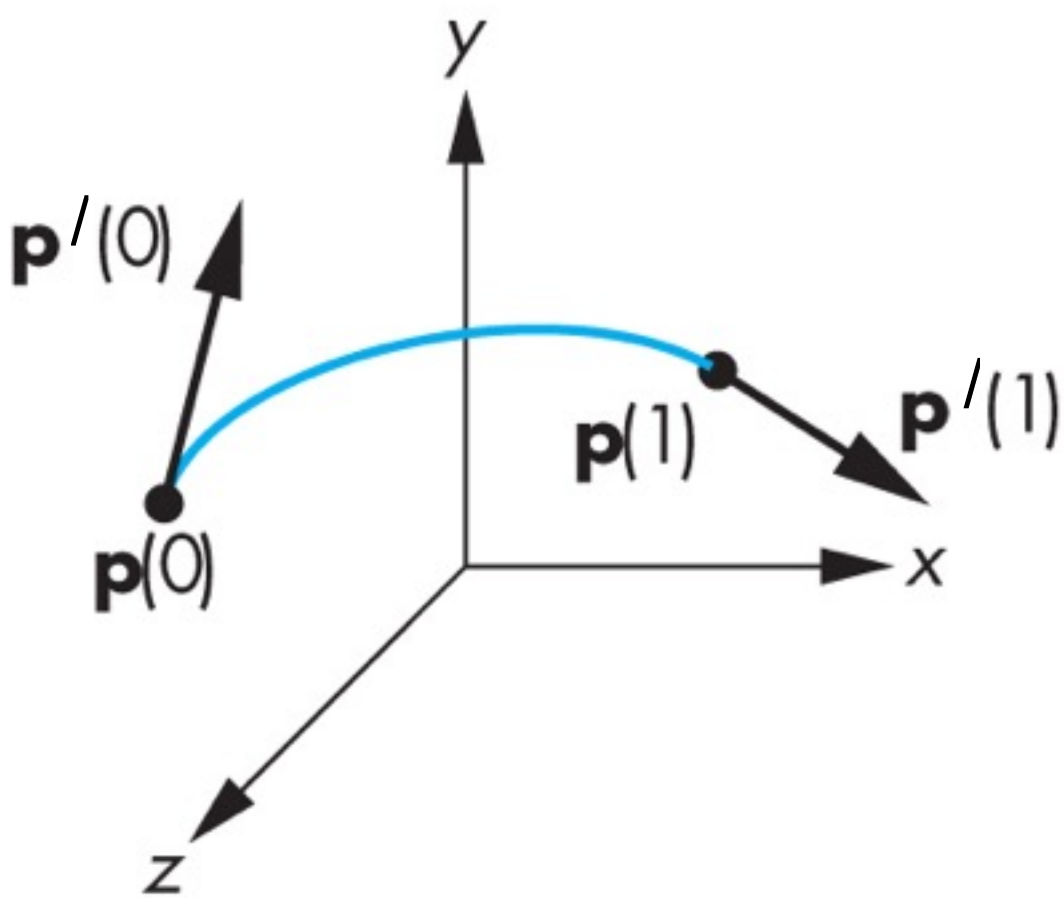
- Natural cubics
  - C2 continuity
  - n points -> n-1 cubic segments
- control is non-local :(
- ill-conditioned x(

Cubic Hermite Curves

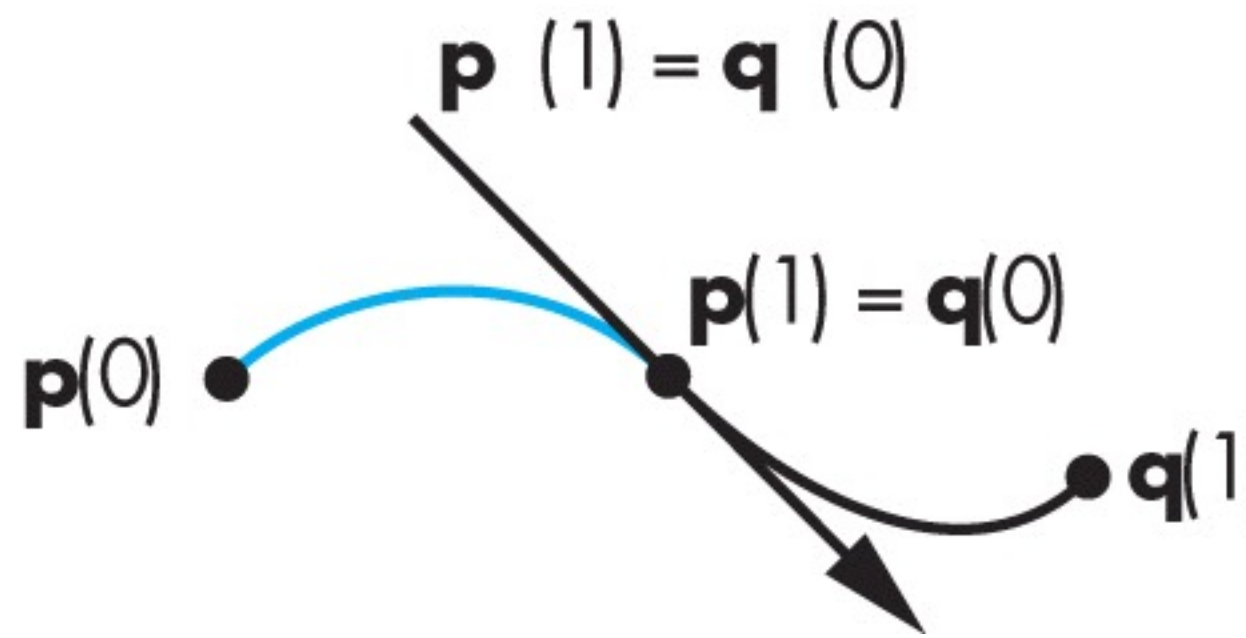
- C1 continuity
- specify both positions and derivatives

Cubic Hermite Curves

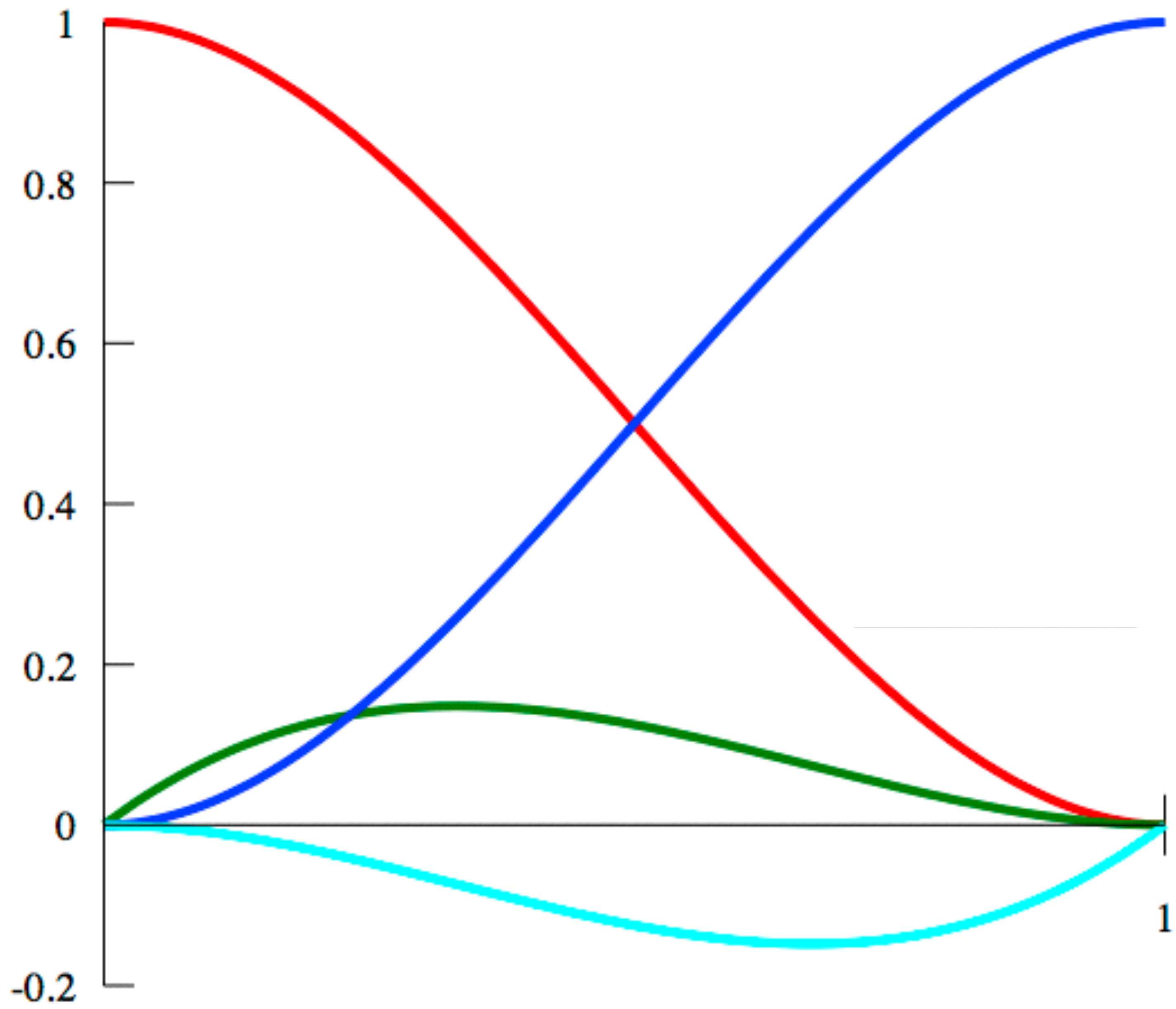
Specify endpoints and derivatives



construct curve with C^1 continuity



Hermite blending functions



[Wikimedia Commons]

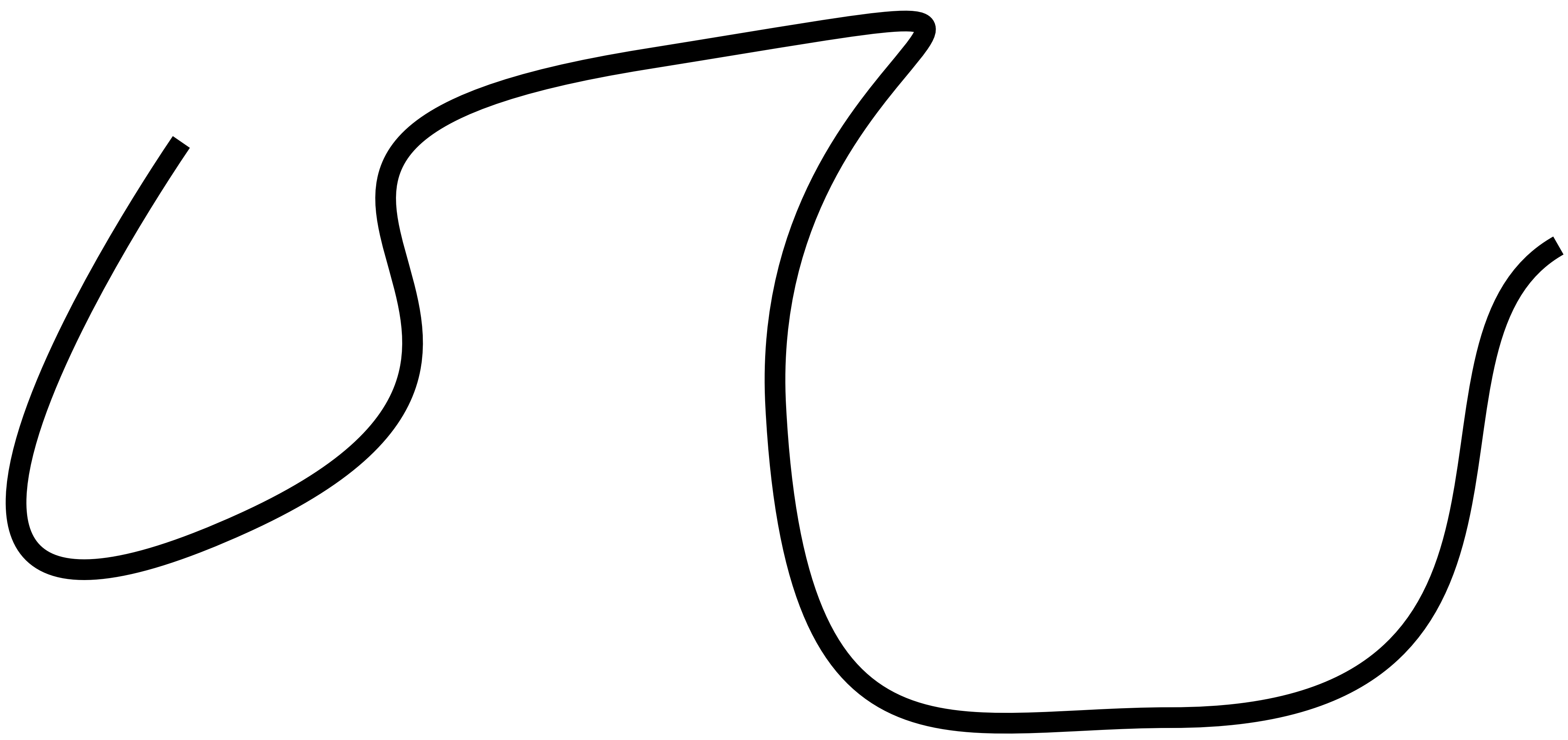
$$b_0(u) = 2u^3 - 3u^2 + 1$$

$$b_1(u) = -2u^3 + 3u^2$$

$$b_2(u) = u^3 - 2u^2 + u$$

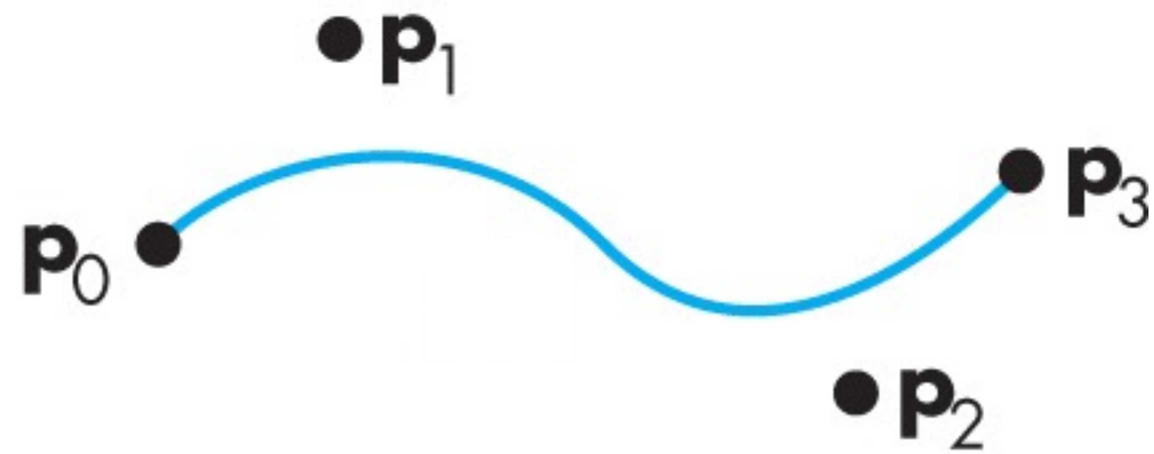
$$b_3(u) = u^3 - u^2$$

Example: keynote curve tool



Interpolating vs. Approximating Curves

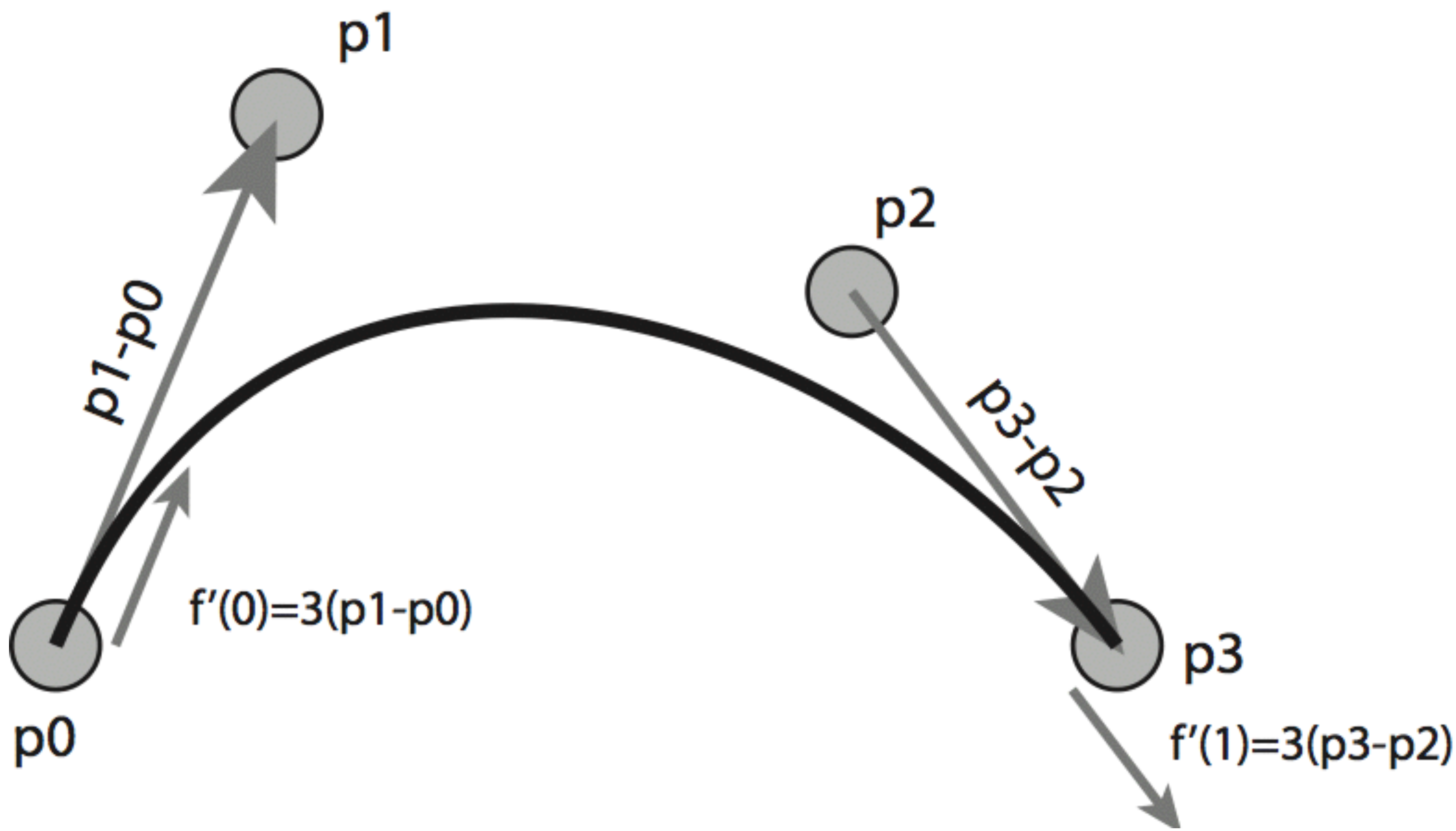
Interpolating



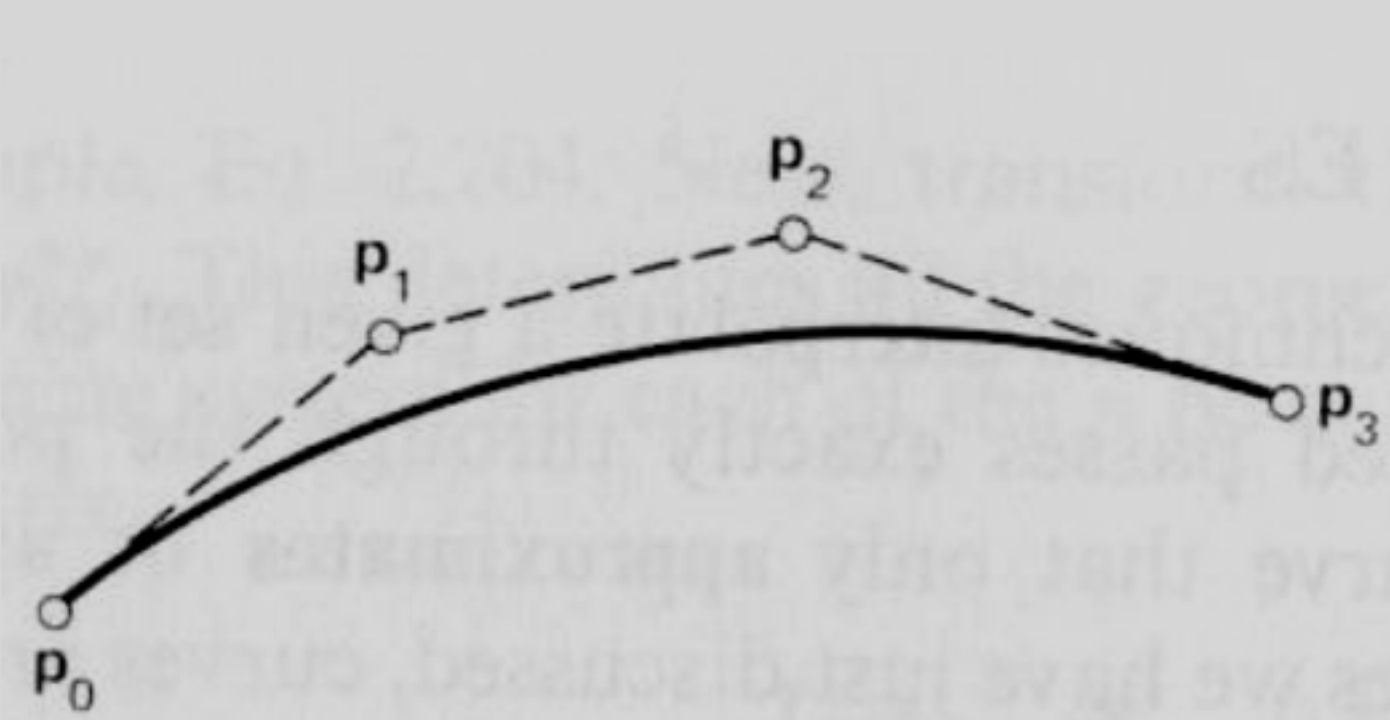
Approximating
(non-interpolating)

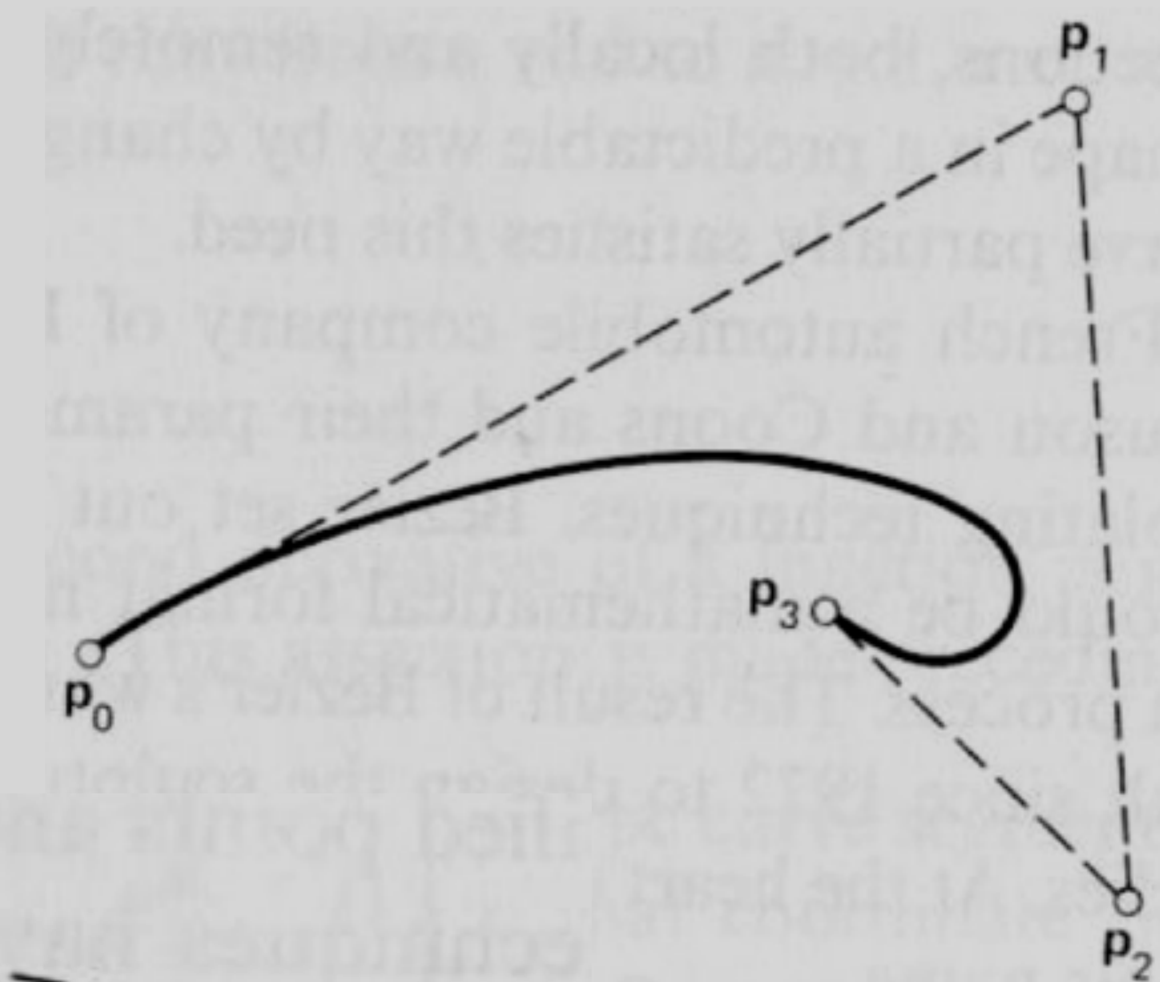
Cubic Bezier Curves

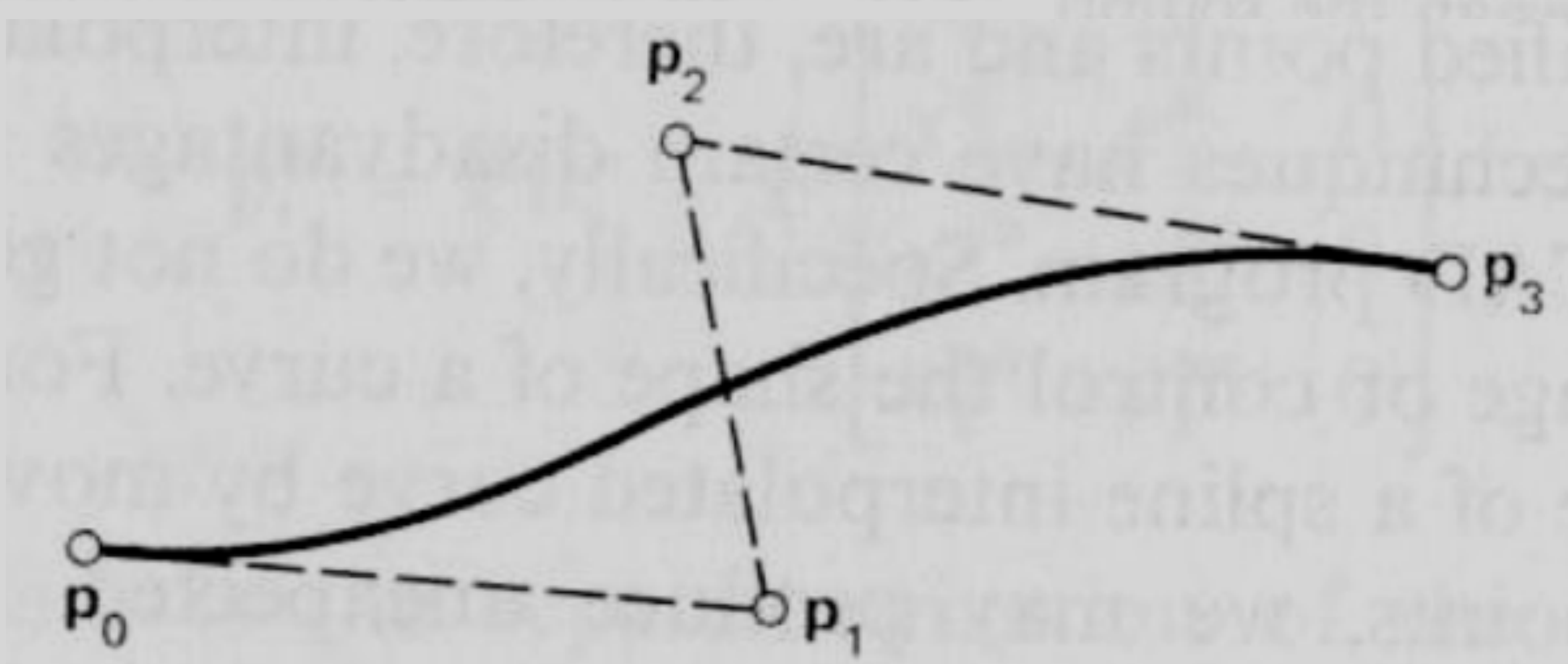
Cubic Bezier Curves



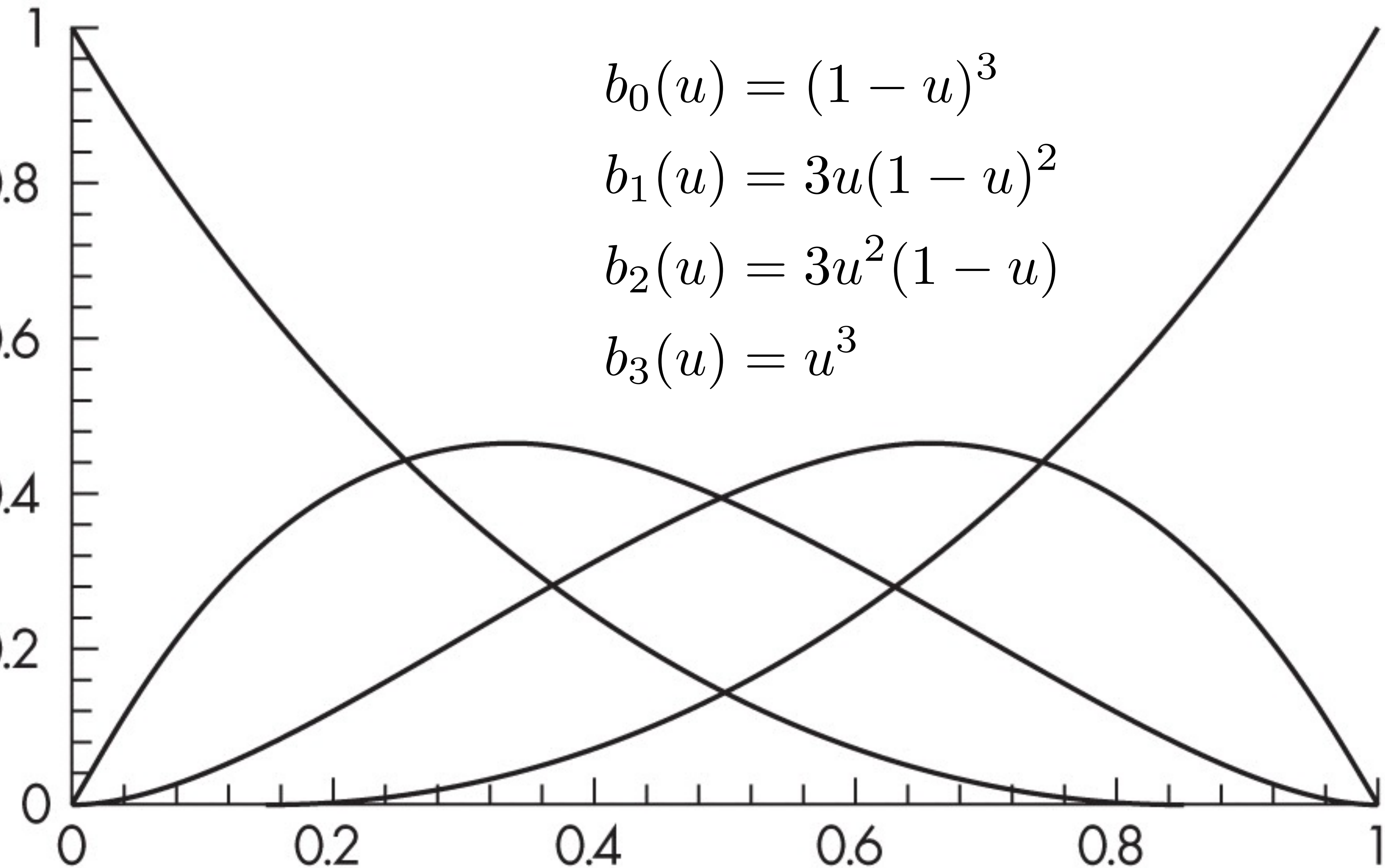
Cubic Bezier Curve Examples





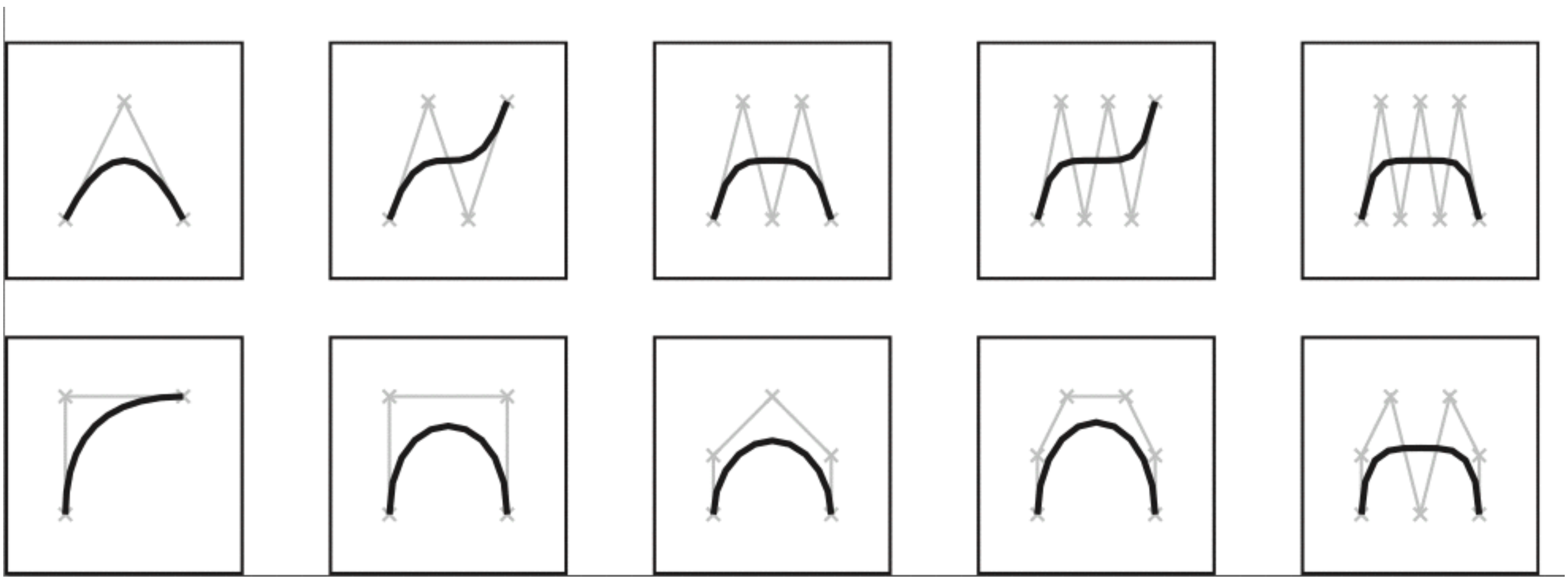


Cubic Bezier blending functions



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

Bezier Curves Degrees 2-6



Bernstein Polynomials

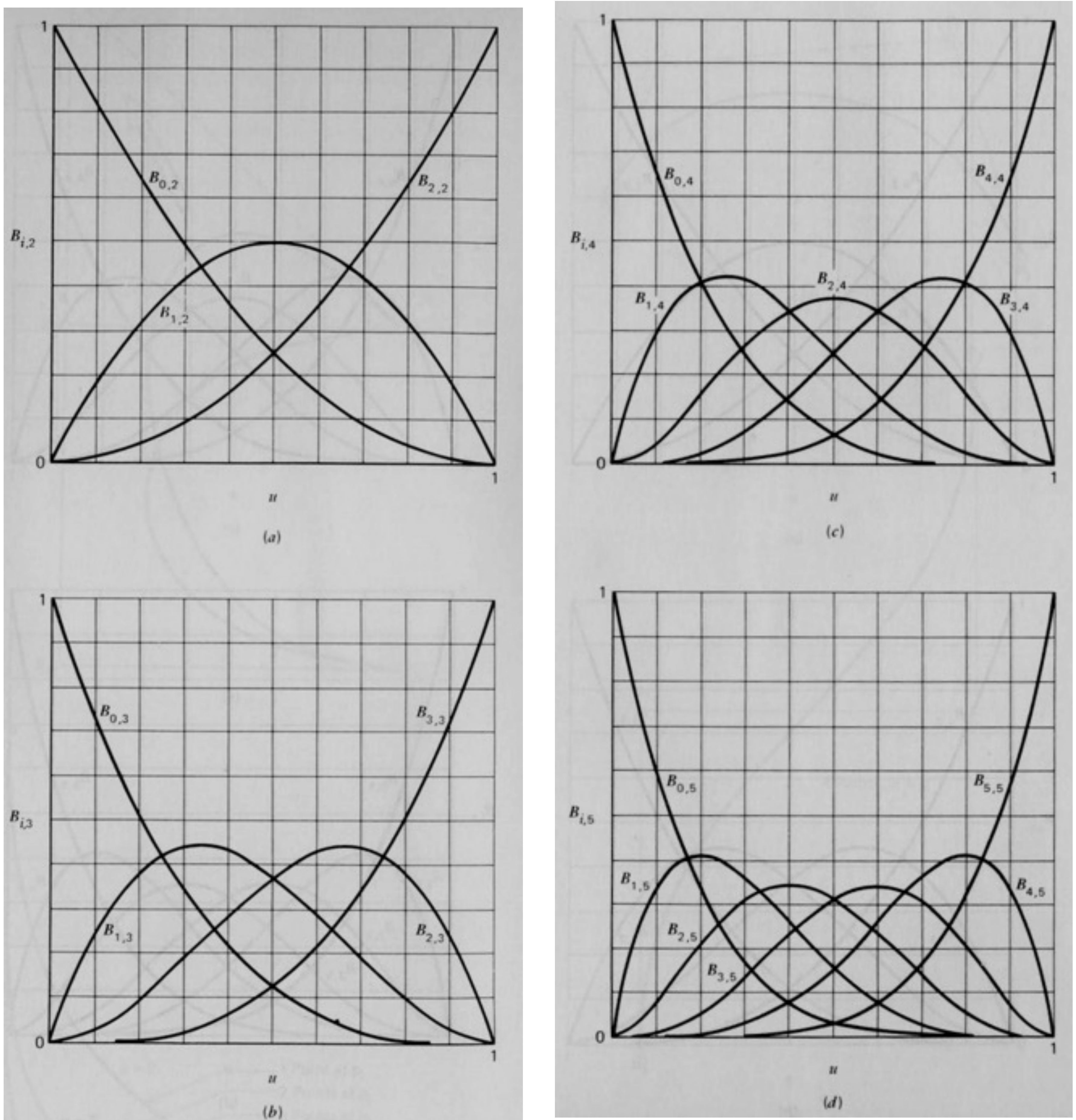
- The blending functions are a special case of the Bernstein polynomials

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

- 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)

n = 3

n = 5



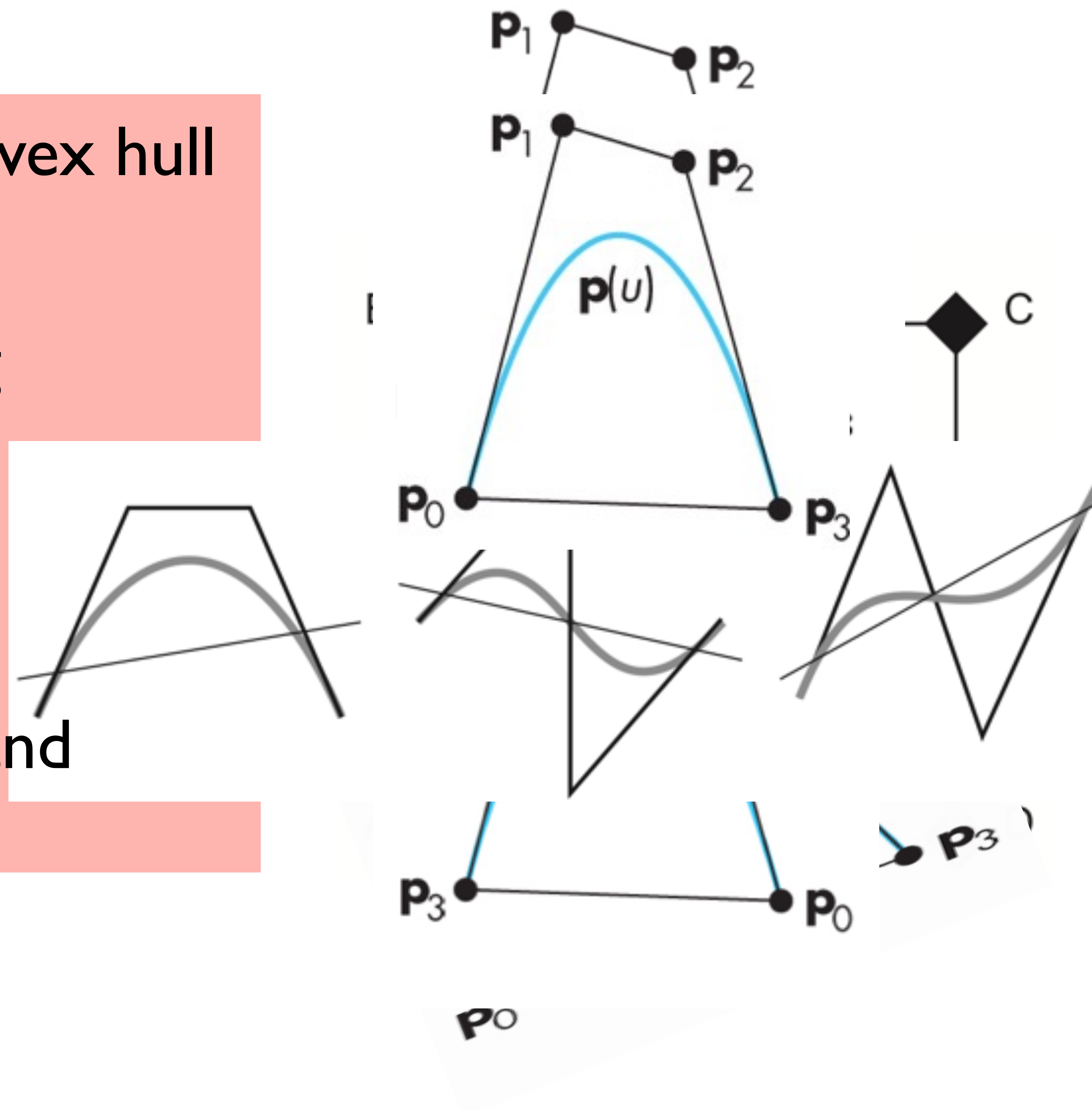
$B_{i,2}$
$B_{0,2}$
$B_{1,2}$
$B_{2,2}$
u
(a)
$B_{i,4}$
$B_{0,4}$
$B_{1,4}$
$B_{2,4}$
$B_{3,4}$
$B_{4,4}$
u
(c)
$B_{i,3}$
$B_{0,3}$
$B_{1,3}$
$B_{2,3}$
$B_{3,3}$
u
(b)
$B_{i,5}$
$B_{0,5}$
$B_{1,5}$
$B_{2,5}$
$B_{3,5}$
$B_{4,5}$
$B_{5,5}$
u
(d)

n = 4

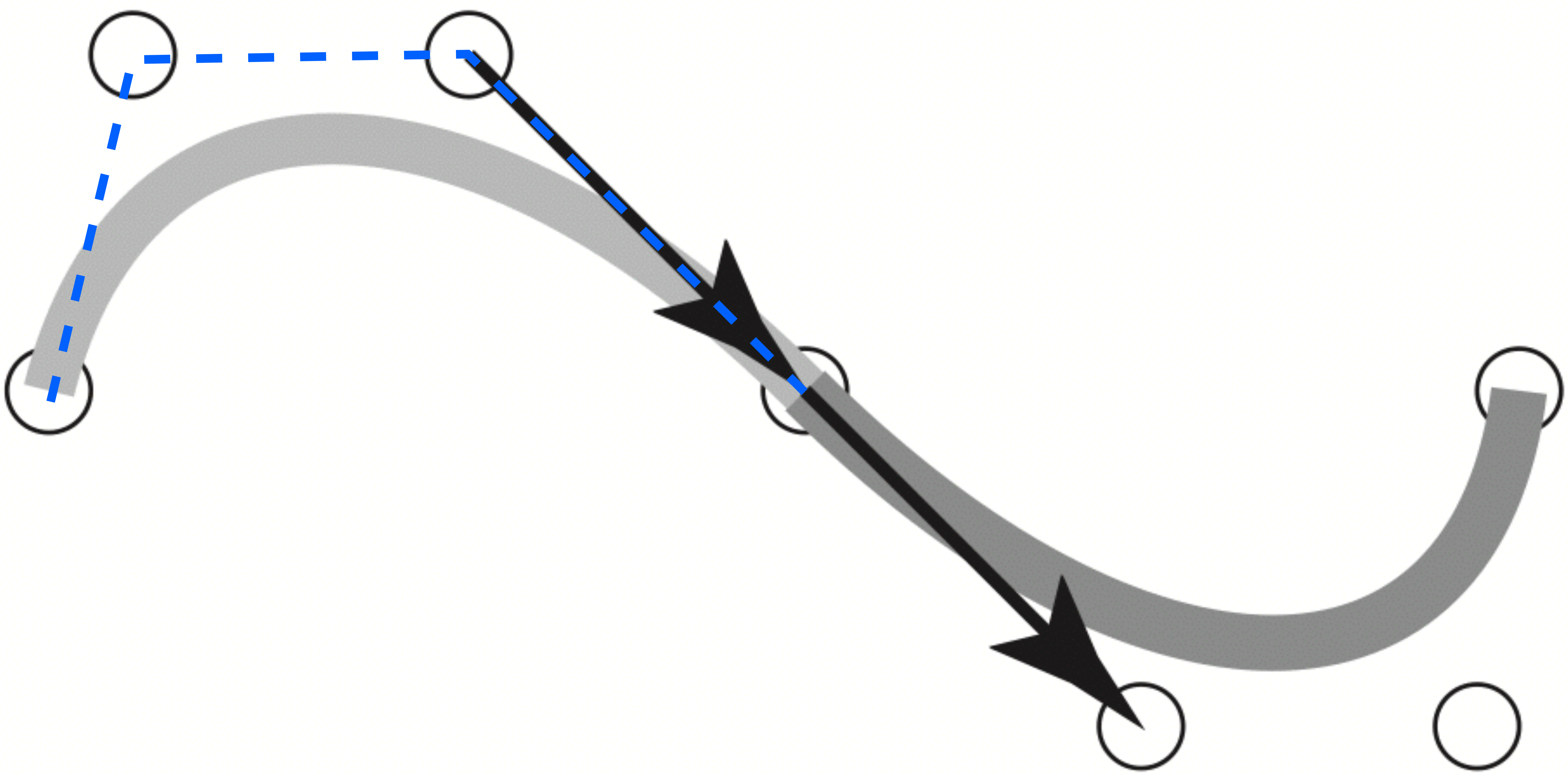
n = 6

Bezier Curve Properties

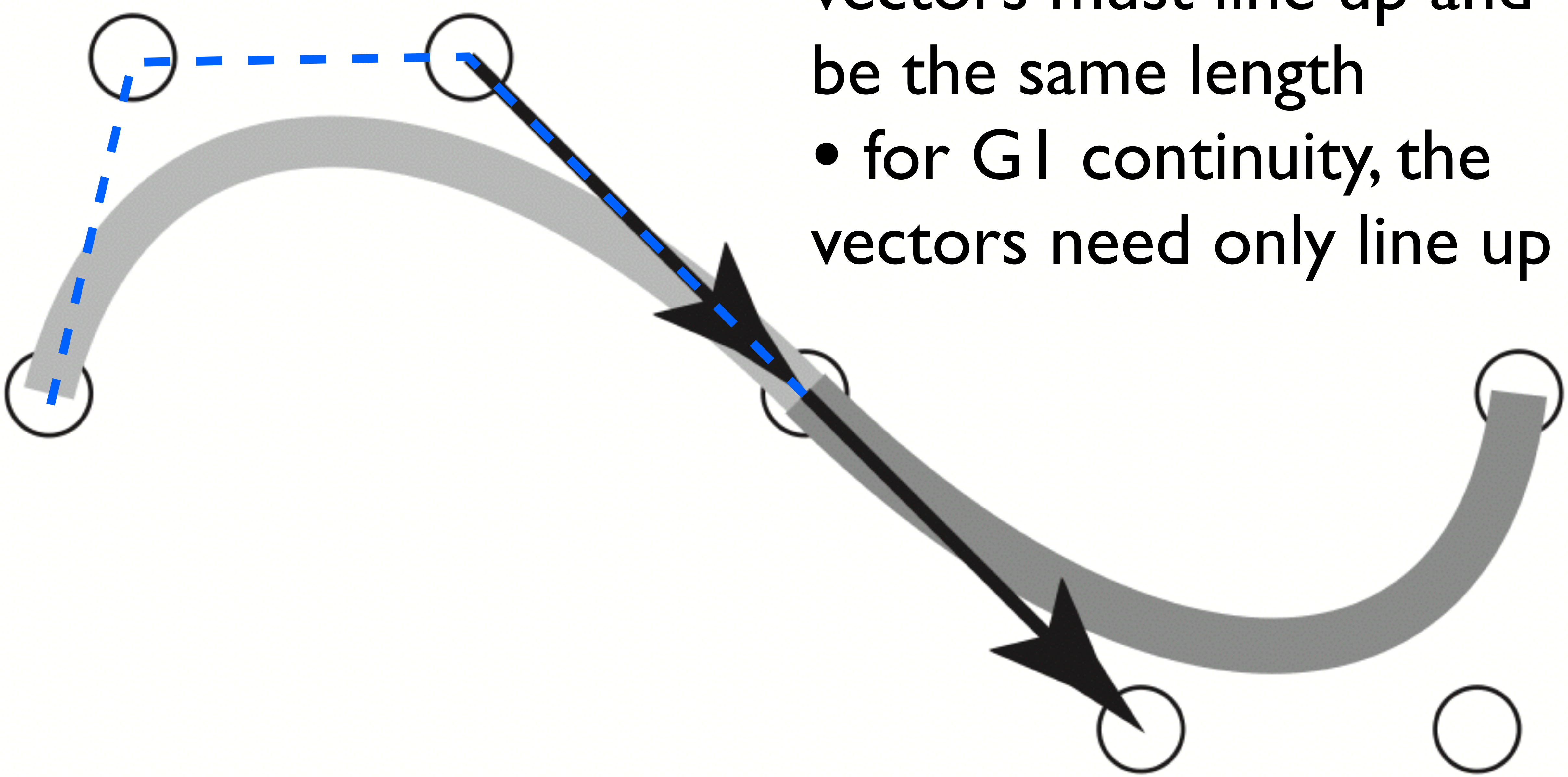
- curve lies in the convex hull of the data
- variation diminishing
- symmetry
- affine invariant
- efficient evaluation and subdivision



Joining Cubic Bezier Curves

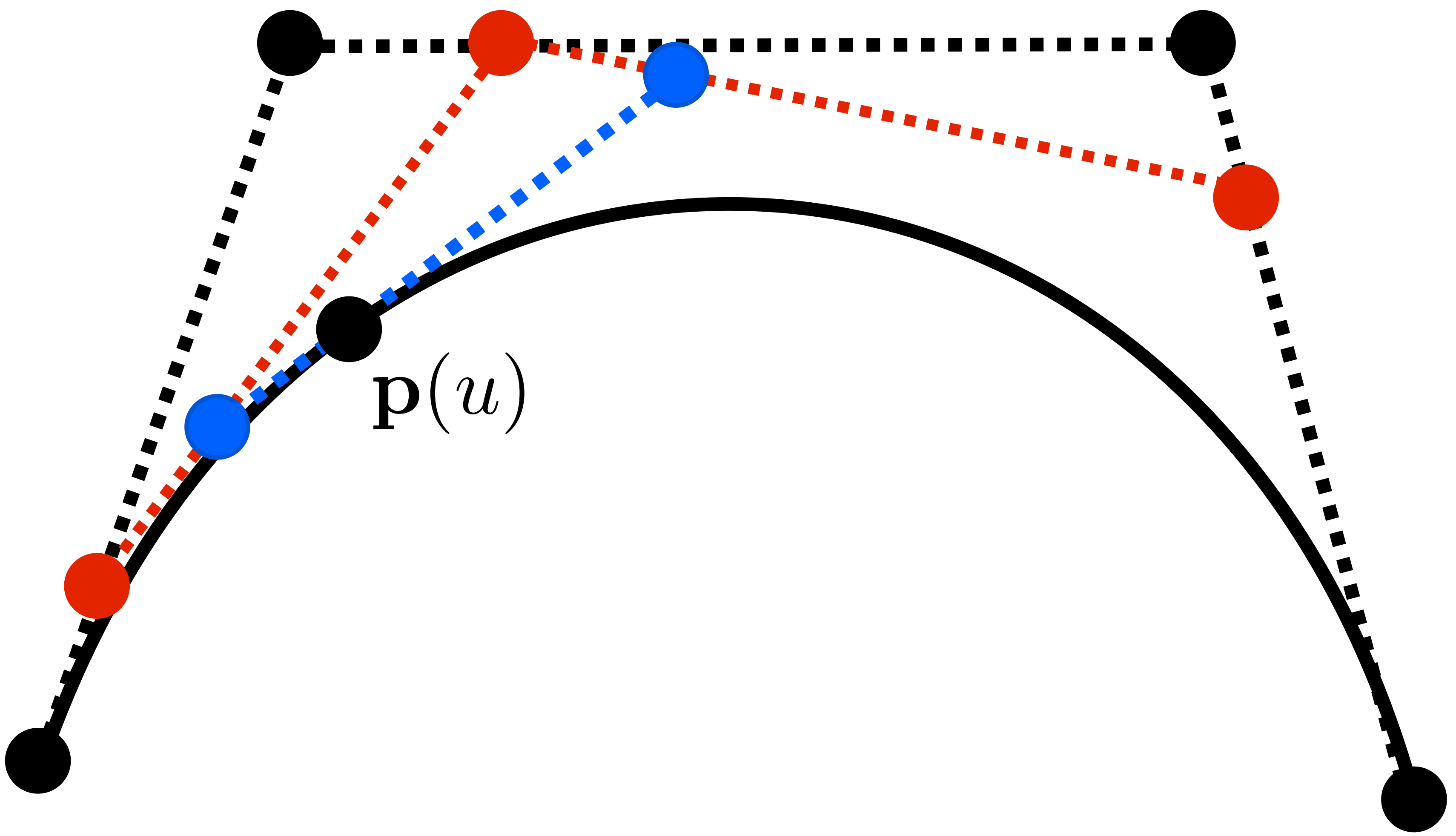


Joining Cubic Bezier Curves

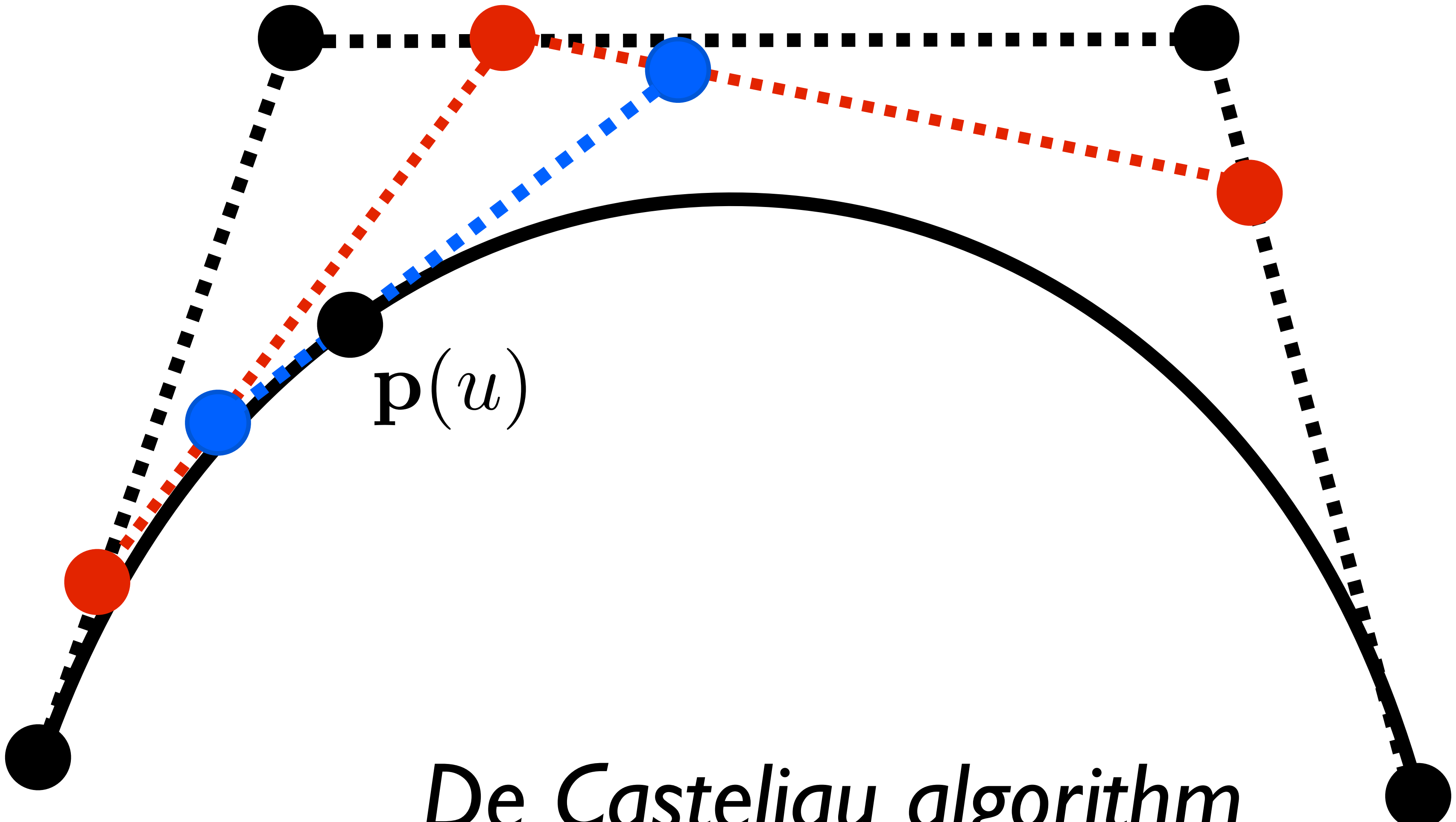


- for C1 continuity, the vectors must line up and be the same length
- for G1 continuity, the vectors need only line up

Evaluating p(u) geometrically

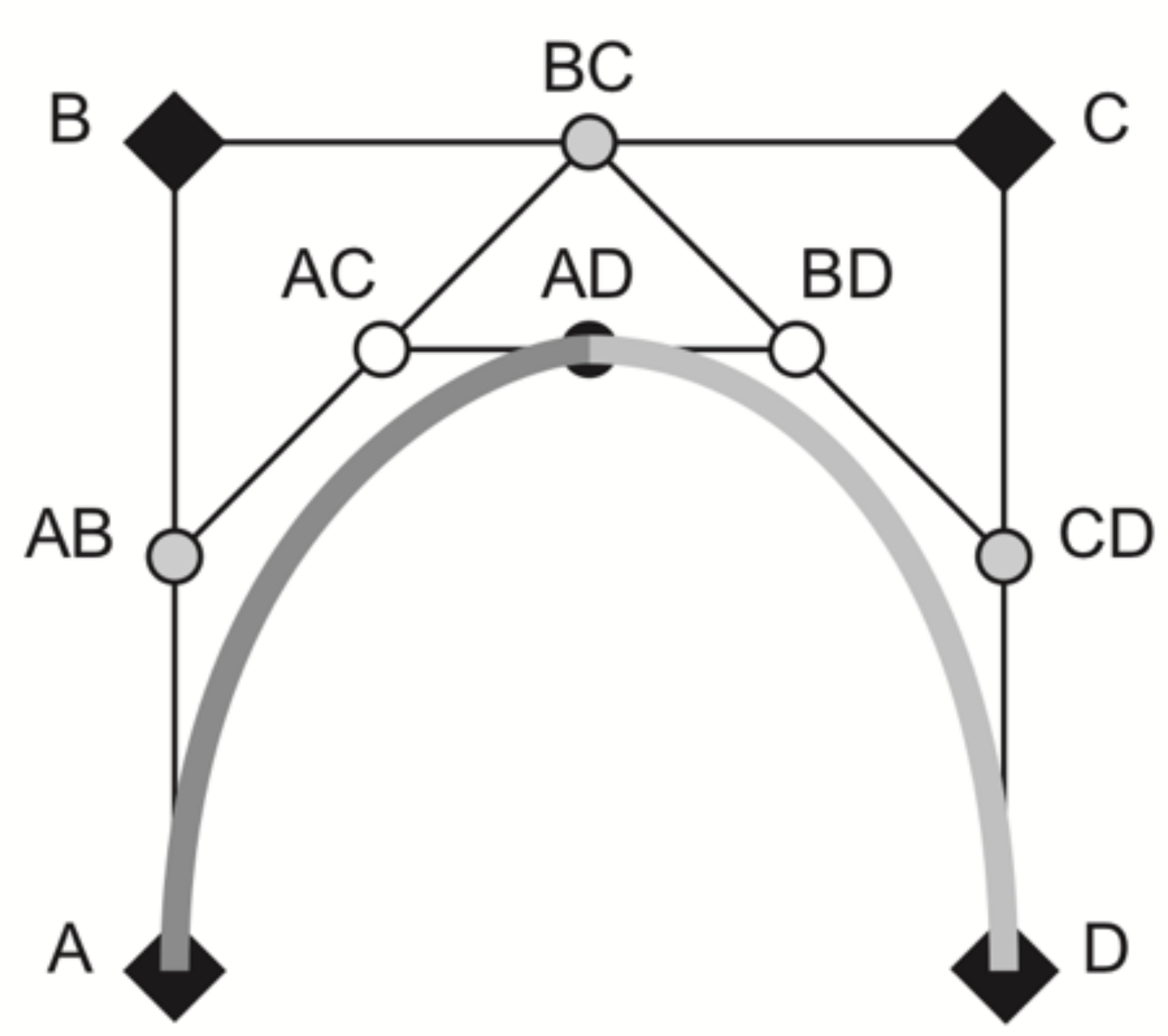


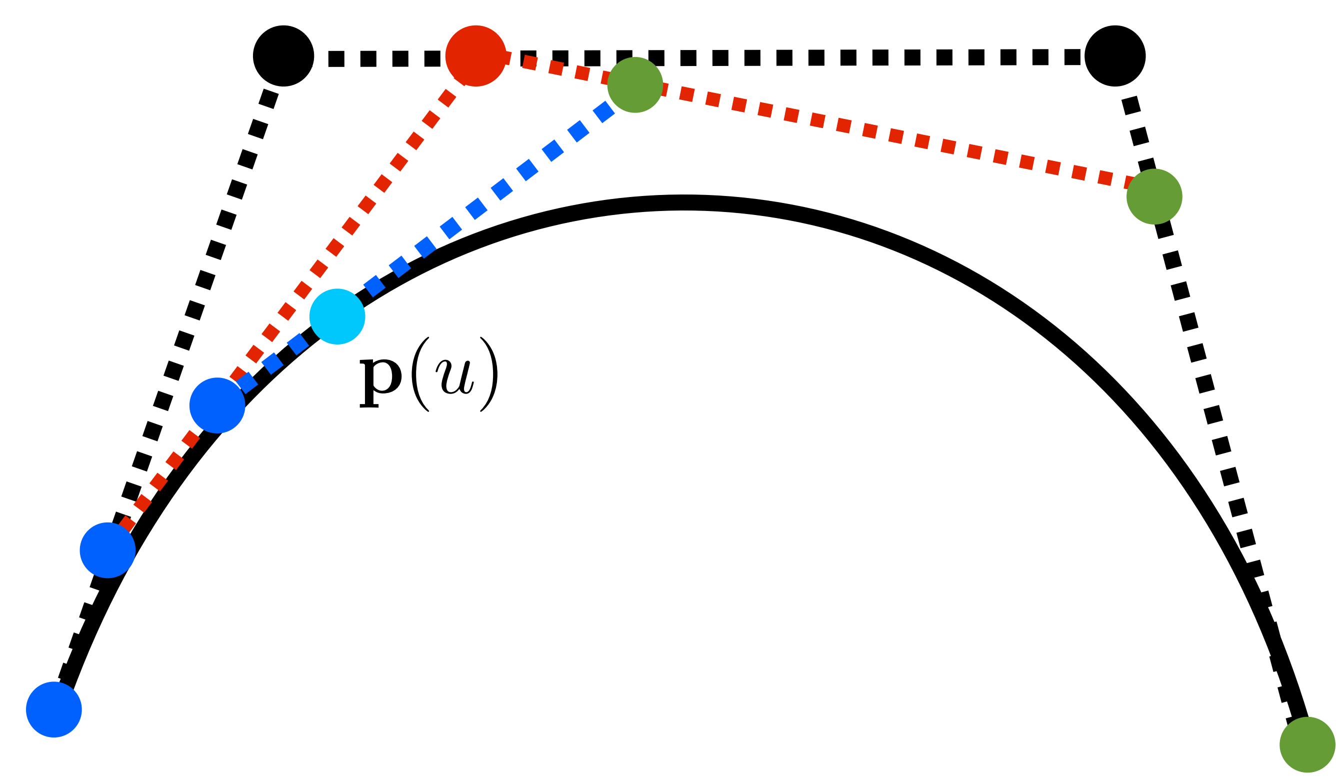
Evaluating p(u) geometrically



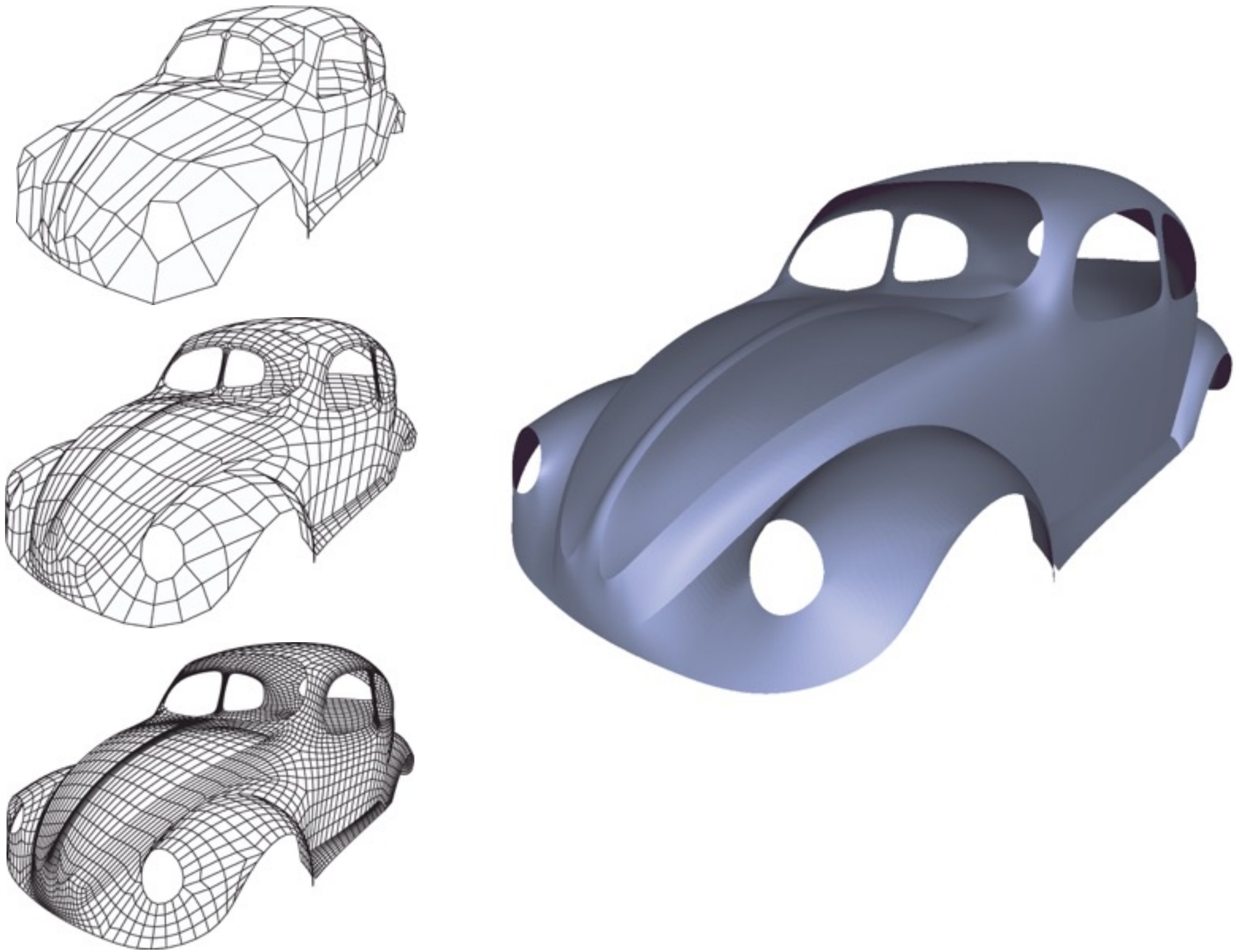
De Casteljau algorithm

Bezier subdivision



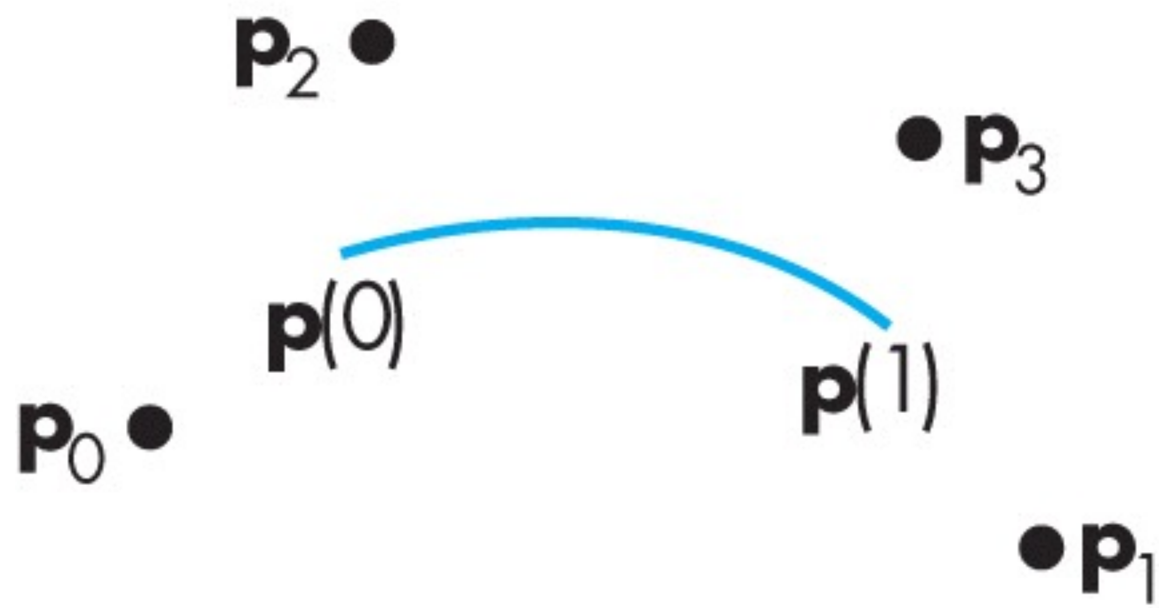


Recursive Subdivision for Rendering



Cubic B-Splines

Cubic B-Splines



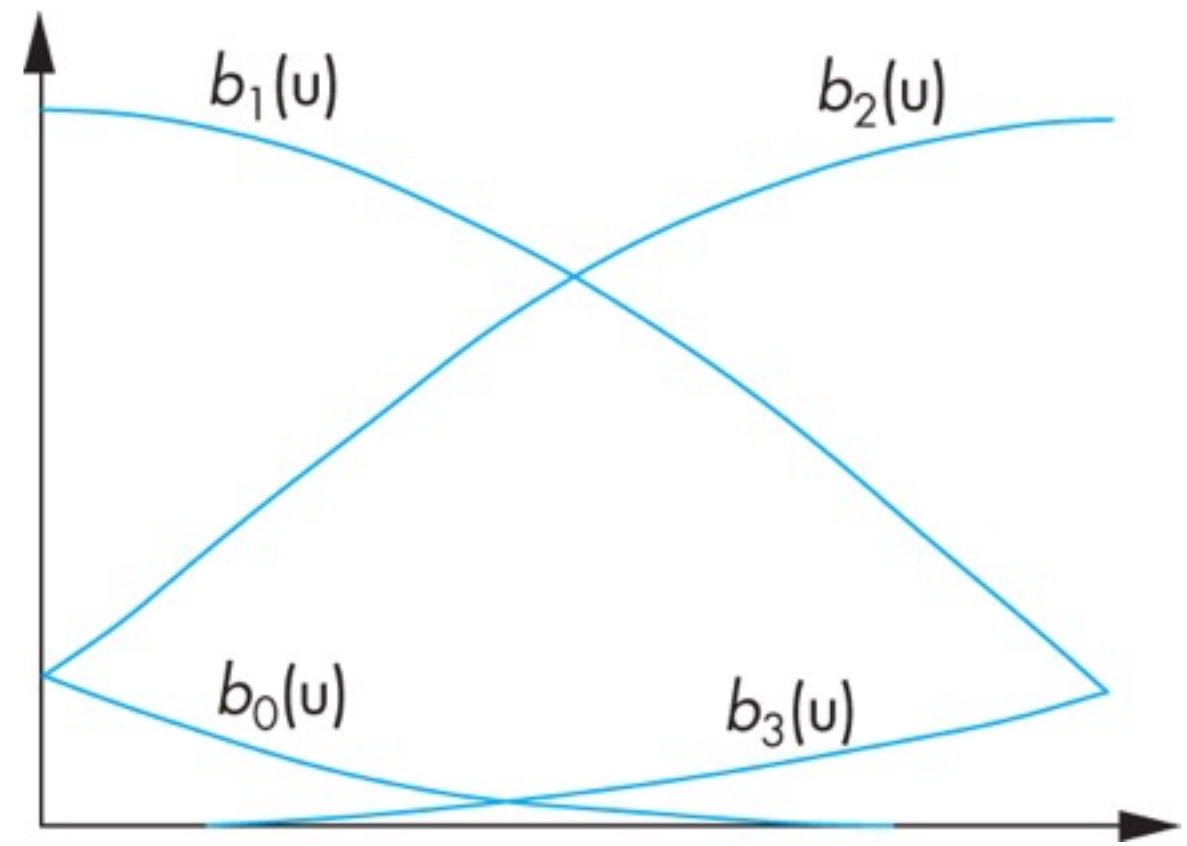
Spline blending functions

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

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

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

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

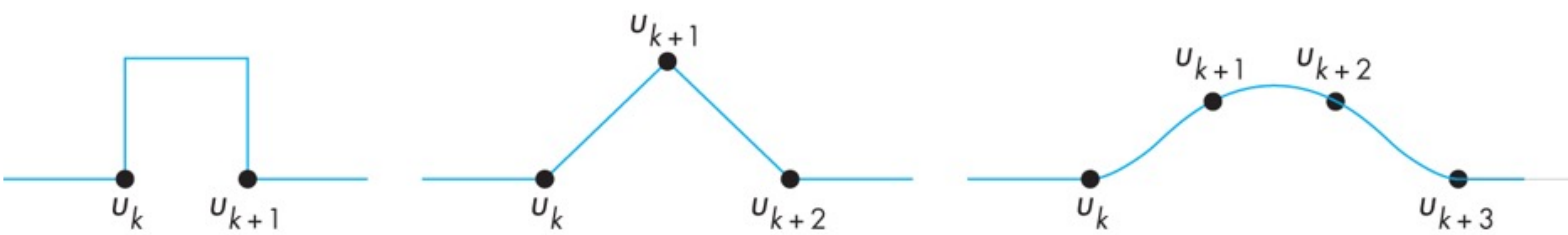


General Splines

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

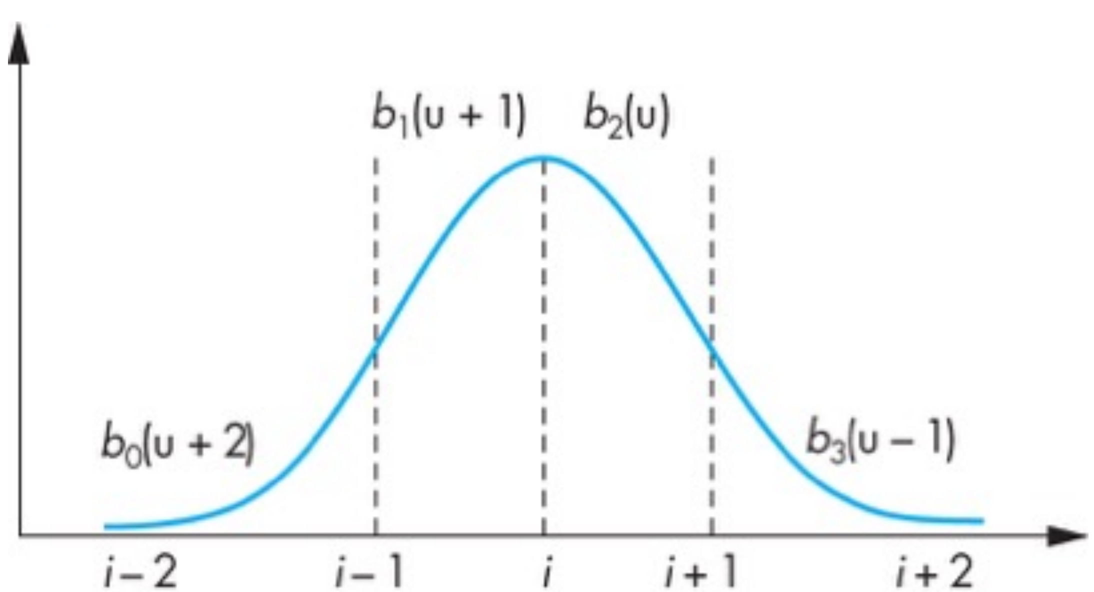
$$b_{j,0}(t) = \begin{cases} 1 & \text{if} \quad t_j \leq t \\ 0 & \text{otherwise} \end{cases}$$

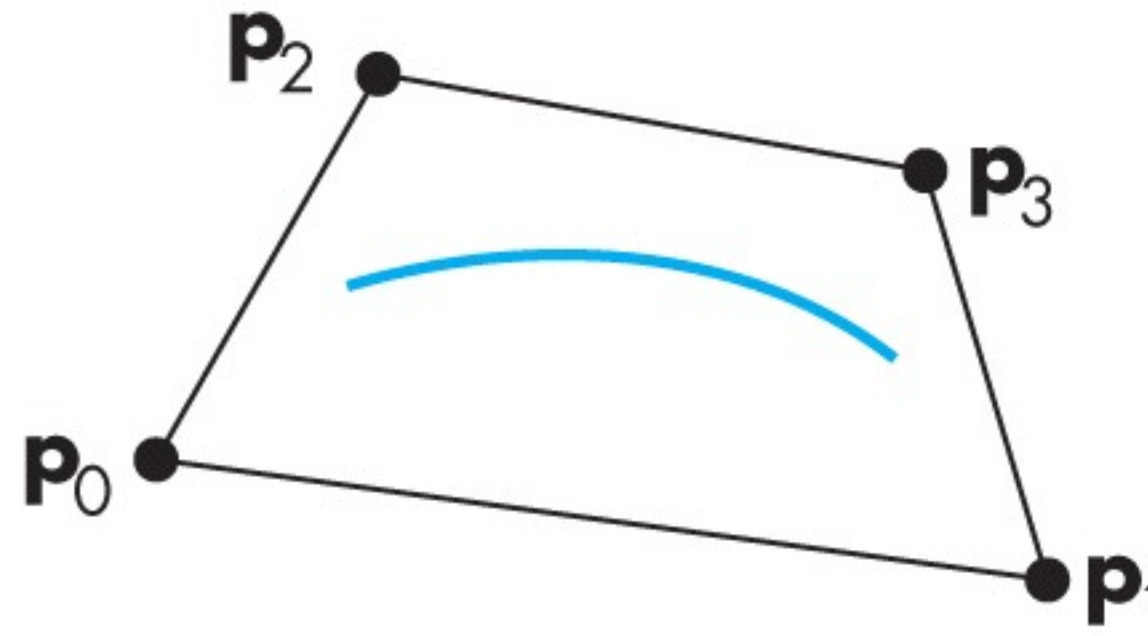
$$b_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t)$$



Spline properties

Basis functions



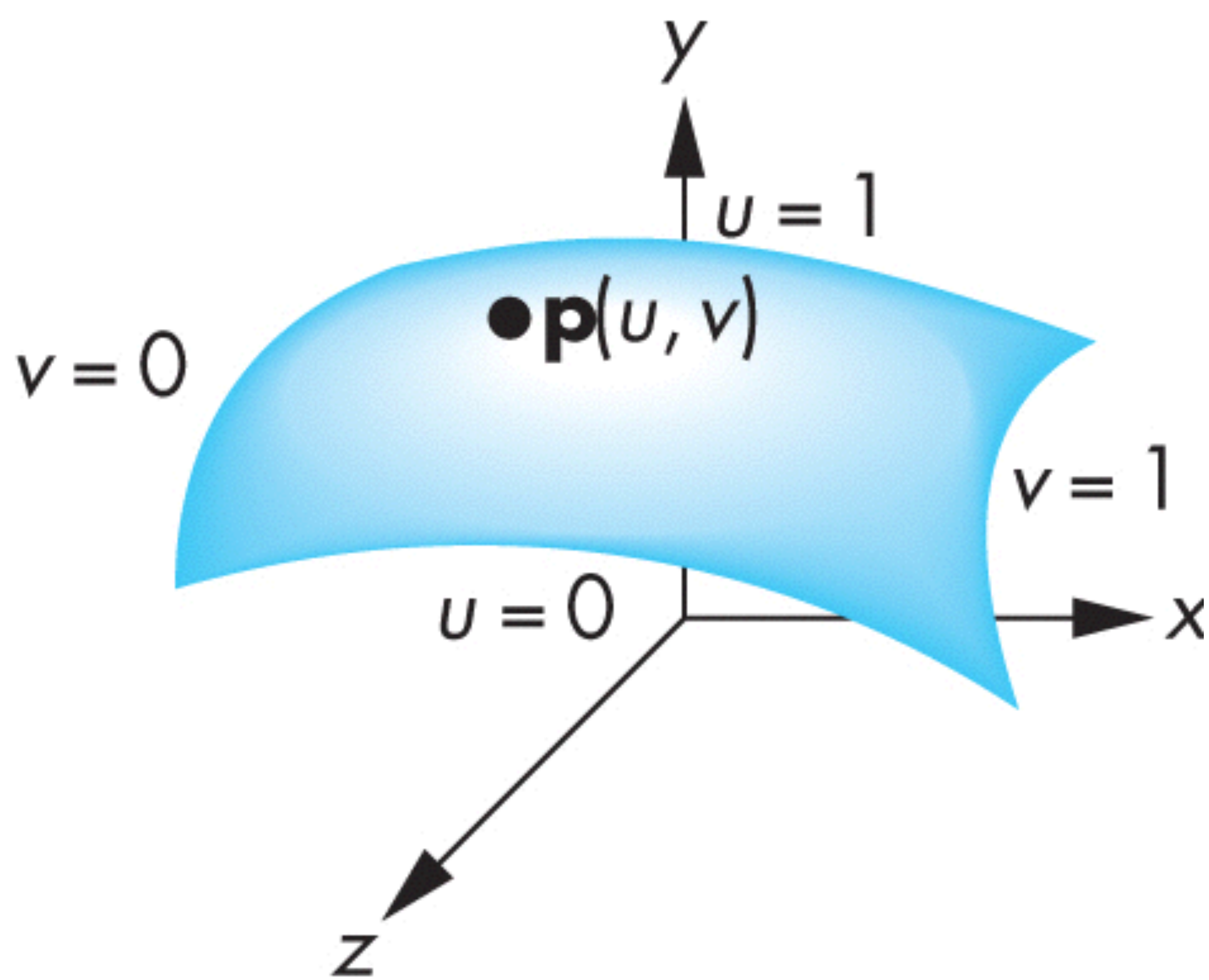


convexity

Surfaces

Parametric Surface

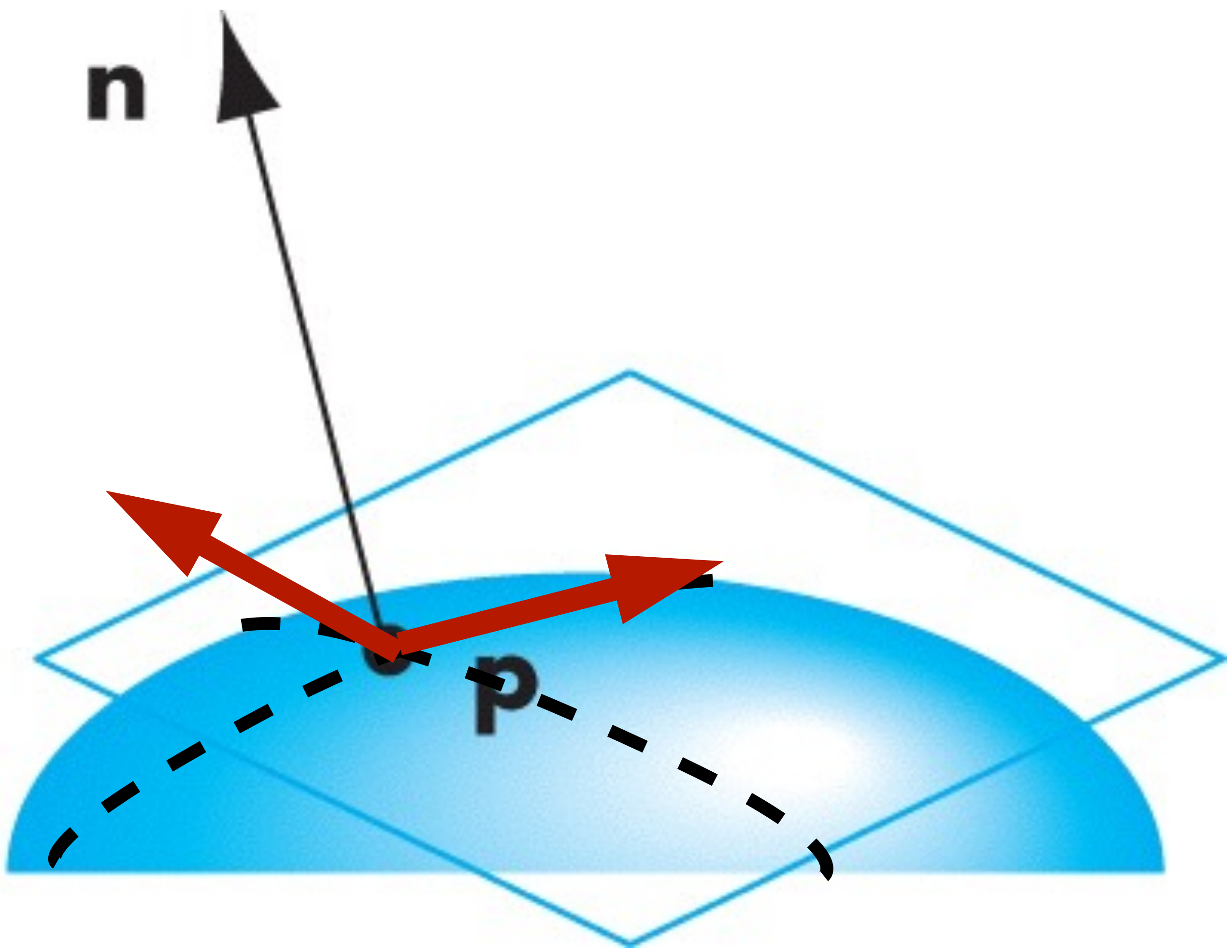
$$x = x(u, v)$$
$$y = y(u, v)$$
$$z = z(u, v)$$



Parametric Surface - tangent plane

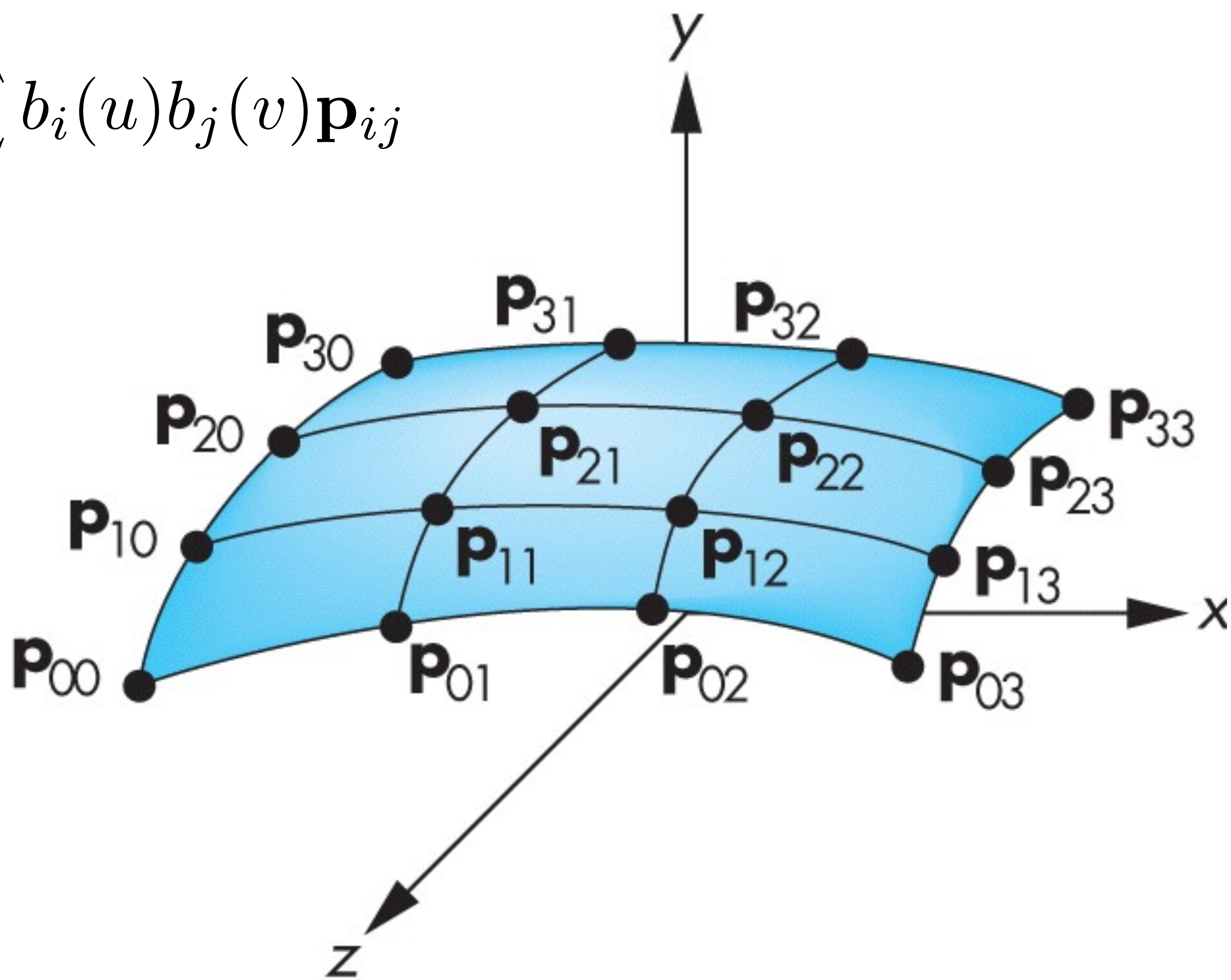
$$\mathbf{t}_u = \begin{pmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{pmatrix}$$

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



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