2D Spline Curves

CS 4620 Lecture 27

Administration

- PPA2 due today
- A5 out today

Plan

1. Spline segments
   - how to define a polynomial on [0,1]
   - …that has the properties you want
   - …and is easy to control
2. Spline curves
   - how to chain together lots of segments
   - …so that the whole curve has the properties you want
   - …and is easy to control
3. Refinement and evaluation
   - how to add detail to splines
   - how to approximate them with line segments

Matrix form of spline

$$\mathbf{f}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

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

How to find the matrix?

- Given constraints
  - Points that you must go through or nearby: 2 or 4
  - Derivatives you must match
  - Acceleration

Hermite splines: 2 points, 2 derivatives

- Solve constraints to find coefficients

$$x(t) = at^3 + bt^2 + ct + d$$
$$x'(t) = 3at^2 + 2bt + c$$
$$x(0) = x_0 = d$$
$$x(1) = x_1 = a + b + c + d$$
$$x'(0) = x'_0 = c$$
$$x'(1) = x'_1 = 3a + 2b + c$$

$$d = x_0$$
$$c = x'_0$$
$$a = 2x_0 - 2x_1 + x'_0 + x'_1$$
$$b = -3x_0 + 3x_1 - 2x'_0 - x'_1$$

Hermite splines

- Matrix form is much simpler

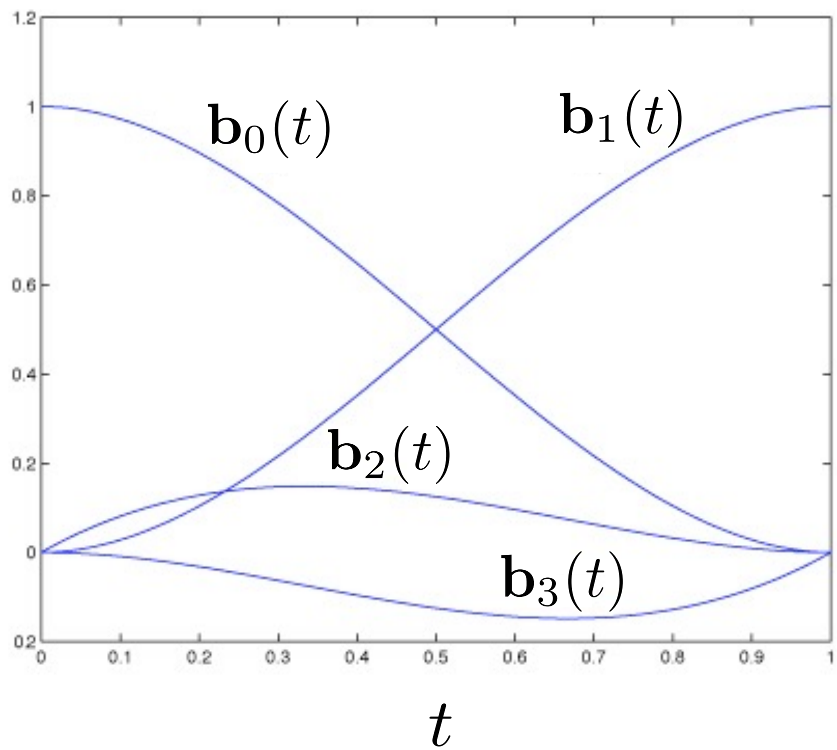
$$\mathbf{f}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{t}_0 \\ \mathbf{t}_1 \end{bmatrix}$$

  – coefficients = rows

  – basis functions = columns

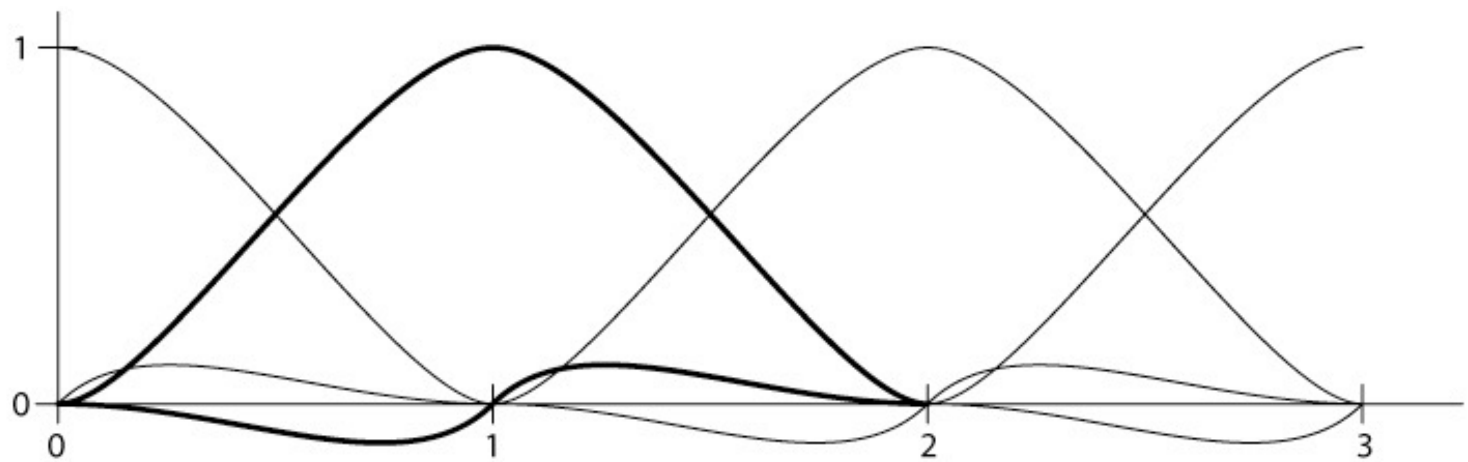
Hermite splines

- Hermite blending functions



Hermite splines

- Hermite basis functions



Bézier matrix

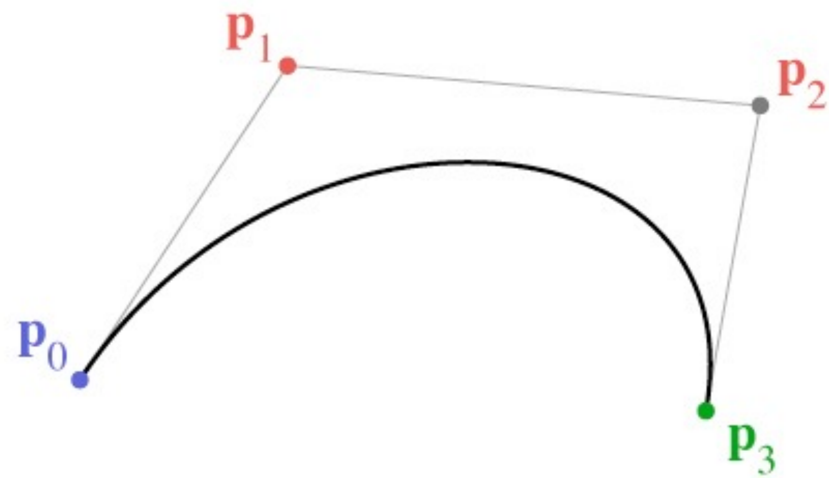
$$\mathbf{f}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

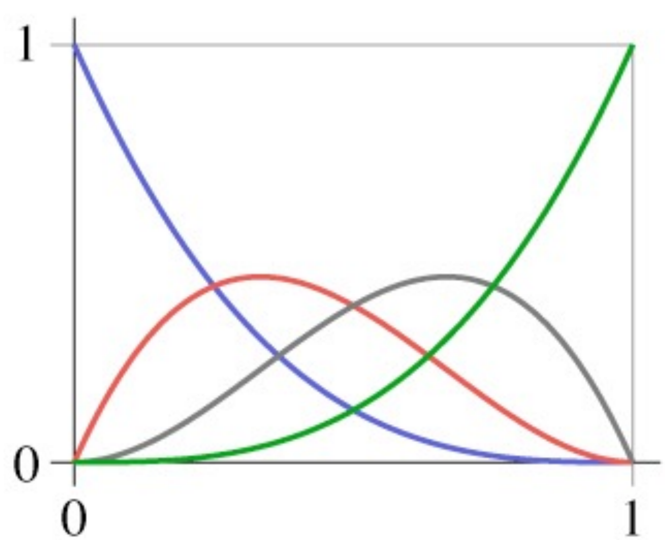
– note that these are the Bernstein polynomials

$$b_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

and that defines Bézier curves for any degree

Bézier basis



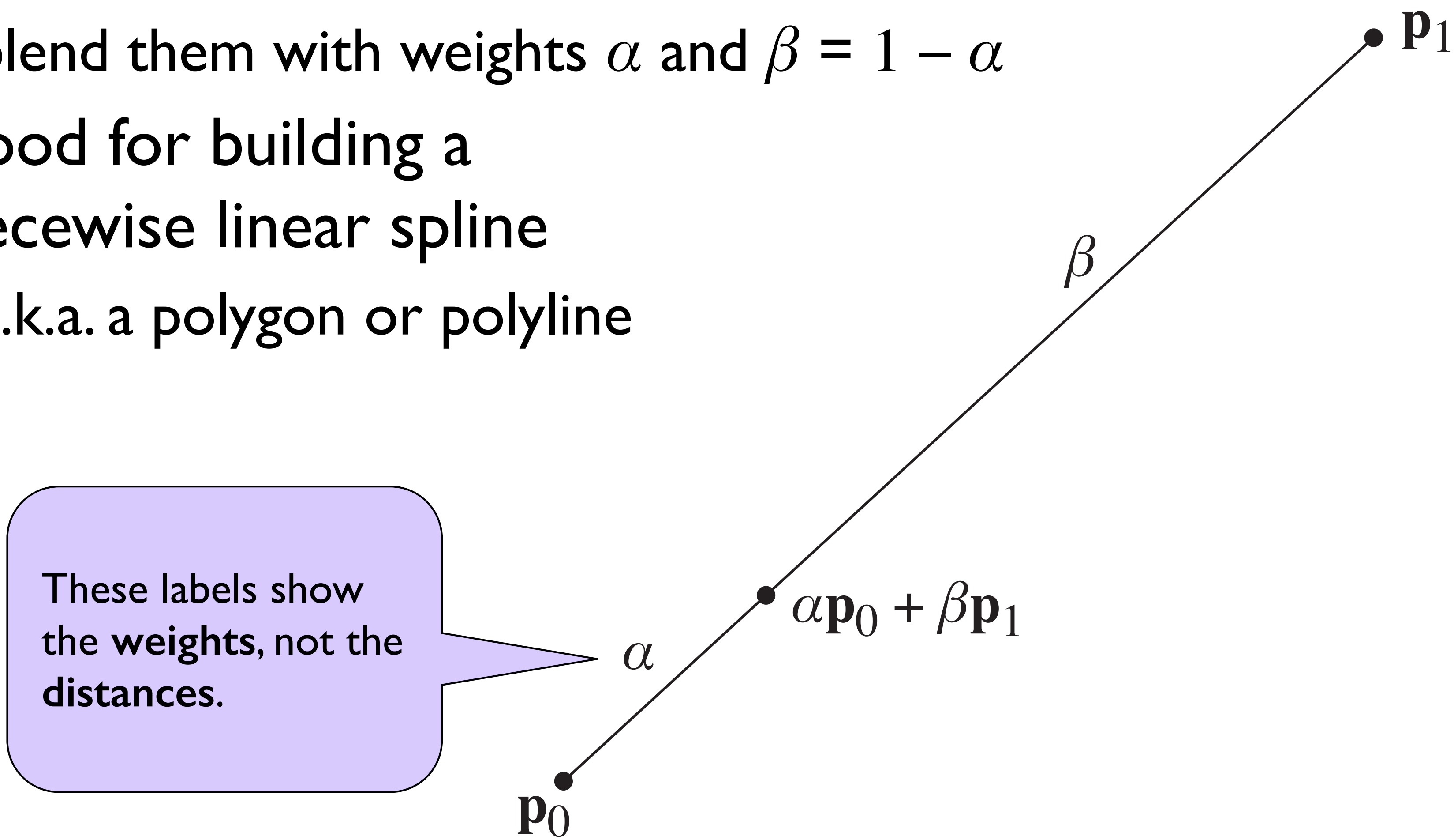


Another way to Bézier segments

- A really boring spline segment: f(t) = p0
  - it only has one control point
  - the curve stays at that point for the whole time
- Only good for building a *piecewise constant* spline
  - a.k.a. a set of points

Another way to Bézier segments

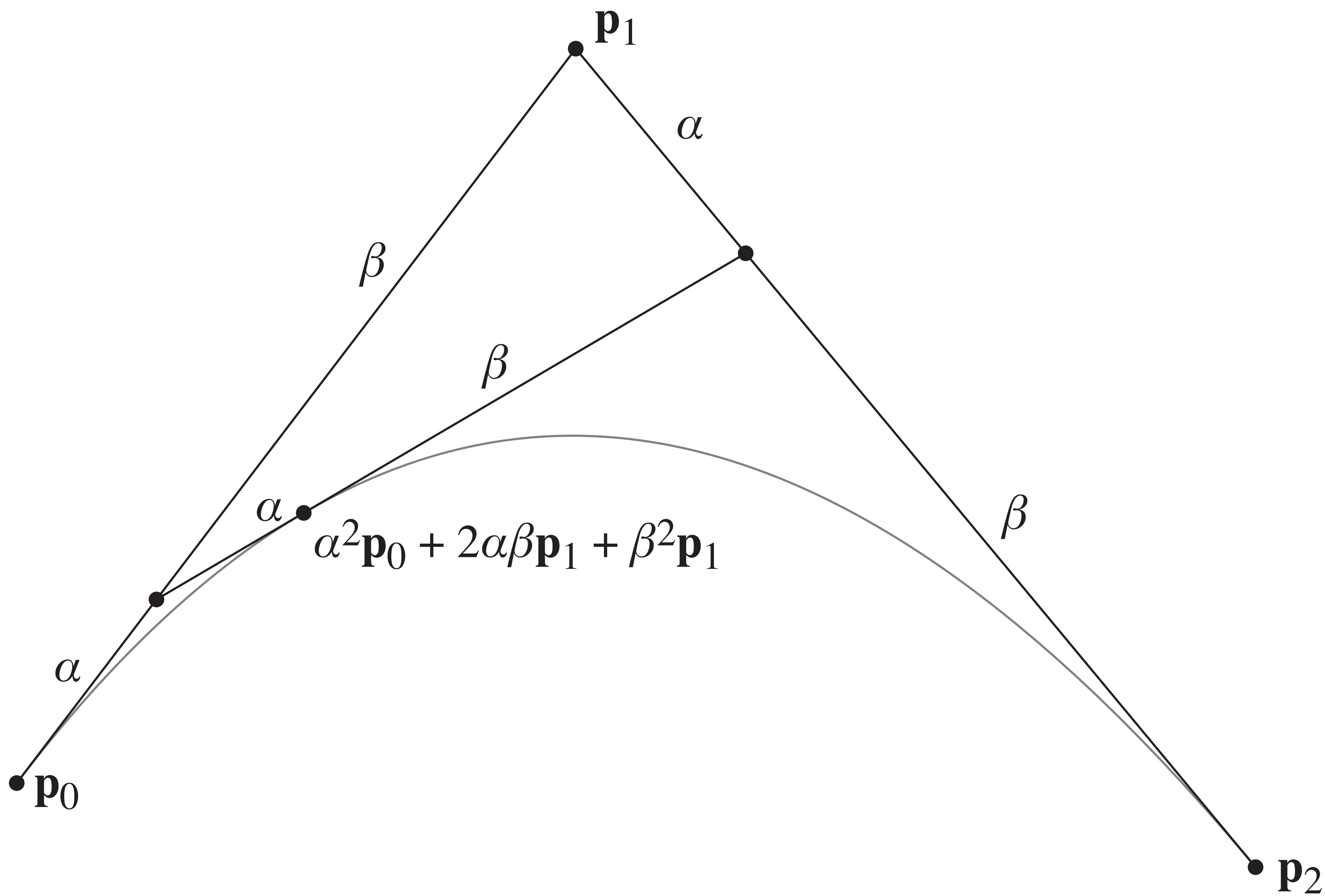
- A piecewise linear spline segment
  - two control points per segment
  - blend them with weights α and $\beta = 1 - \alpha$
- Good for building a piecewise linear spline
  - a.k.a. a polygon or polyline



Another way to Bézier segments

- A linear blend of two piecewise linear segments
  - three control points now
  - interpolate on both segments using α and β
  - blend the results with the same weights
- makes a quadratic spline segment
  - finally, a curve!

$$
\begin{aligned}
\mathbf{p}_{1,0} &= \alpha \mathbf{p}_0 + \beta \mathbf{p}_1 \\
\mathbf{p}_{1,1} &= \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 \\
\mathbf{p}_{2,0} &= \alpha \mathbf{p}_{1,0} + \beta \mathbf{p}_{1,1} \\
&= \alpha\alpha \mathbf{p}_0 + \alpha\beta \mathbf{p}_1 + \beta\alpha \mathbf{p}_1 + \beta\beta \mathbf{p}_2 \\
&= \alpha^2 \mathbf{p}_0 + 2\alpha\beta \mathbf{p}_1 + \beta^2 \mathbf{p}_2
\end{aligned}
$$

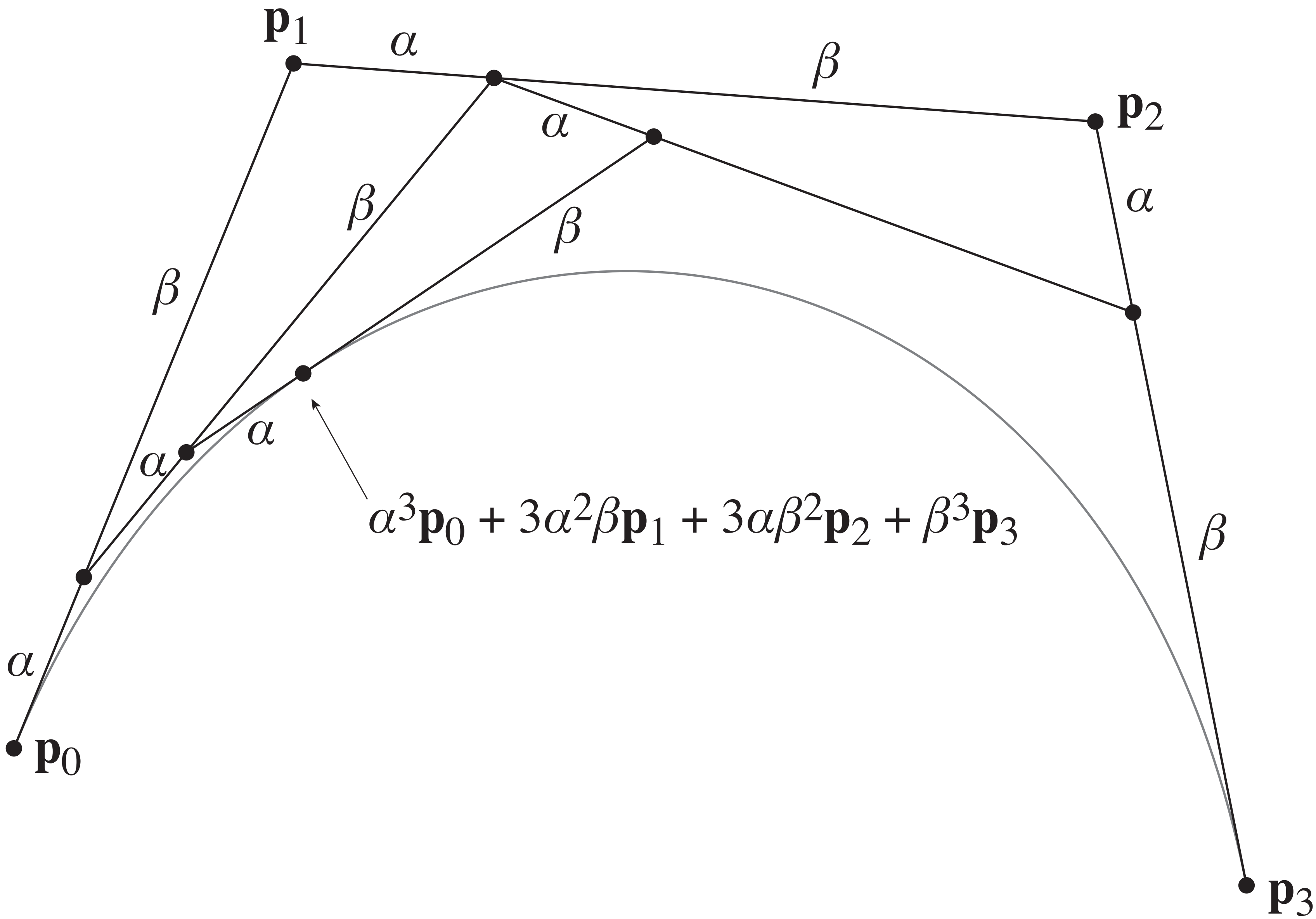


$\mathbf{p}_1$
α
β
β
α
$\alpha^2\mathbf{p}_0 + 2\alpha\beta\mathbf{p}_1 + \beta^2\mathbf{p}_1$
β
α
$\mathbf{p}_0$
$\mathbf{p}_2$

Another way to Bézier segments

- Cubic segment: blend of two quadratic segments
  – four control points now (overlapping sets of 3)
  – interpolate on each quadratic using α and β
  – blend the results with the same weights
- makes a cubic spline segment
  – this is the familiar one for graphics—but you can keep going

$$\begin{aligned}\mathbf{p}_{3,0} =& \alpha\mathbf{p}_{2,0} + \beta\mathbf{p}_{2,1} \\ =& \alpha\alpha\alpha\mathbf{p}_0 + \alpha\alpha\beta\mathbf{p}_1 + \alpha\beta\alpha\mathbf{p}_1 + \alpha\beta\beta\mathbf{p}_2 \\ & \beta\alpha\alpha\mathbf{p}_1 + \beta\alpha\beta\mathbf{p}_2 + \beta\beta\alpha\mathbf{p}_2 + \beta\beta\beta\mathbf{p}_3 \\ =& \alpha^3\mathbf{p}_0 + 3\alpha^2\beta\mathbf{p}_1 + 3\alpha\beta^2\mathbf{p}_2 + \beta^3\mathbf{p}_3\end{aligned}$$



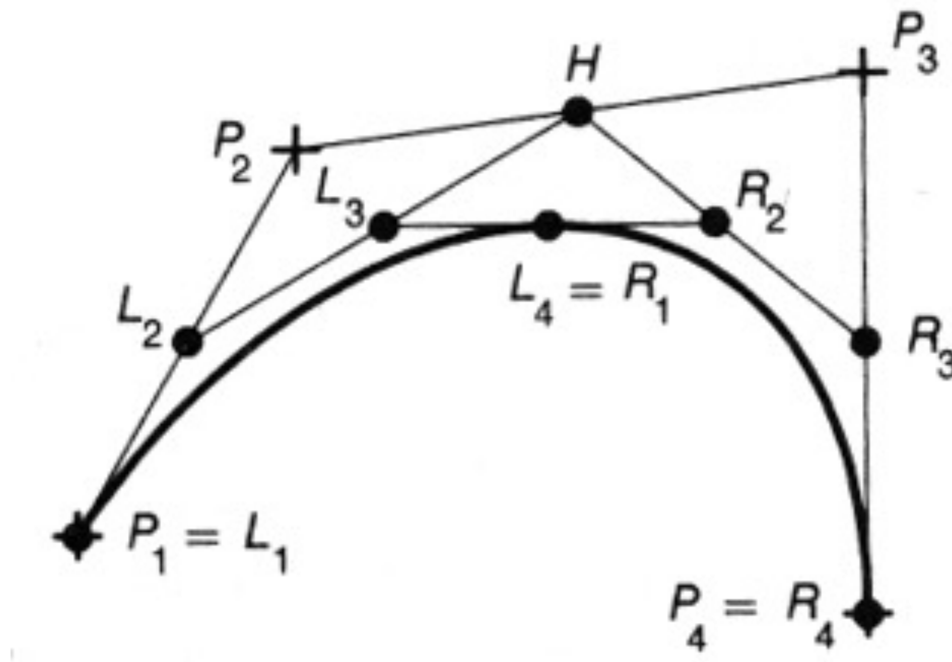
$\mathbf{p}_1$
α
β
$\mathbf{p}_2$
α
β
β
α
β
β
α
α
$\alpha^3\mathbf{p}_0 + 3\alpha^2\beta\mathbf{p}_1 + 3\alpha\beta^2\mathbf{p}_2 + \beta^3\mathbf{p}_3$
β
α
$\mathbf{p}_0$
$\mathbf{p}_3$

de Casteljau's algorithm

- A recurrence for computing points on Bézier spline segments:

$$\mathbf{p}_{0,i} = \mathbf{p}_i$$
$$\mathbf{p}_{n,i} = \alpha \mathbf{p}_{n-1,i} + \beta \mathbf{p}_{n-1,i+1}$$

- Cool additional feature:
also subdivides
the segment into two
shorter ones



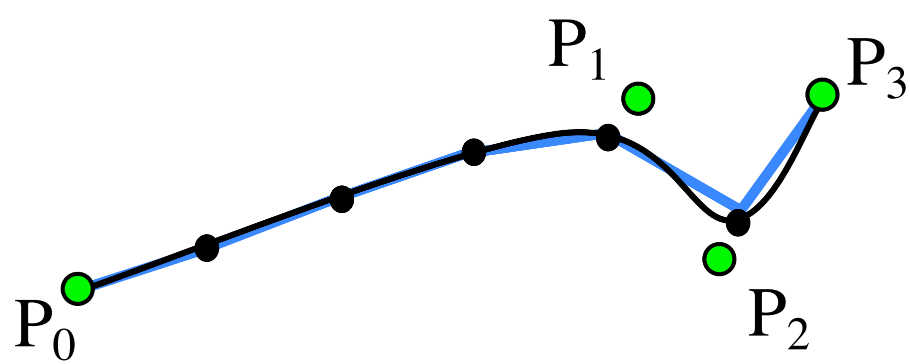
[FvDFH]

Evaluating splines for display

- Need to generate a list of line segments to draw
  - generate efficiently
  - use as few as possible
  - guarantee approximation accuracy
- Approaches
  - recursive subdivision (easy to do adaptively)
  - uniform sampling (easy to do efficiently)

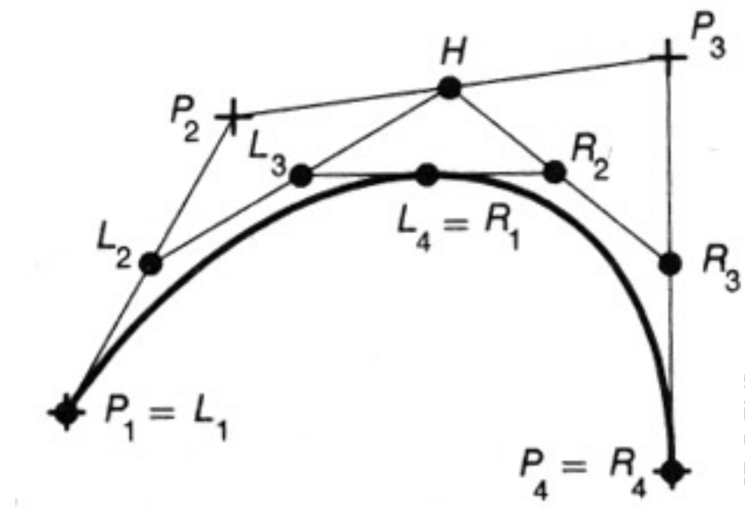
Rendering the curve

- Option 1: uniformly sample in t
- Problem
  - may oversample smooth regions: slow
  - may undersample highly curved regions: faceted rendering



Evaluating by subdivision

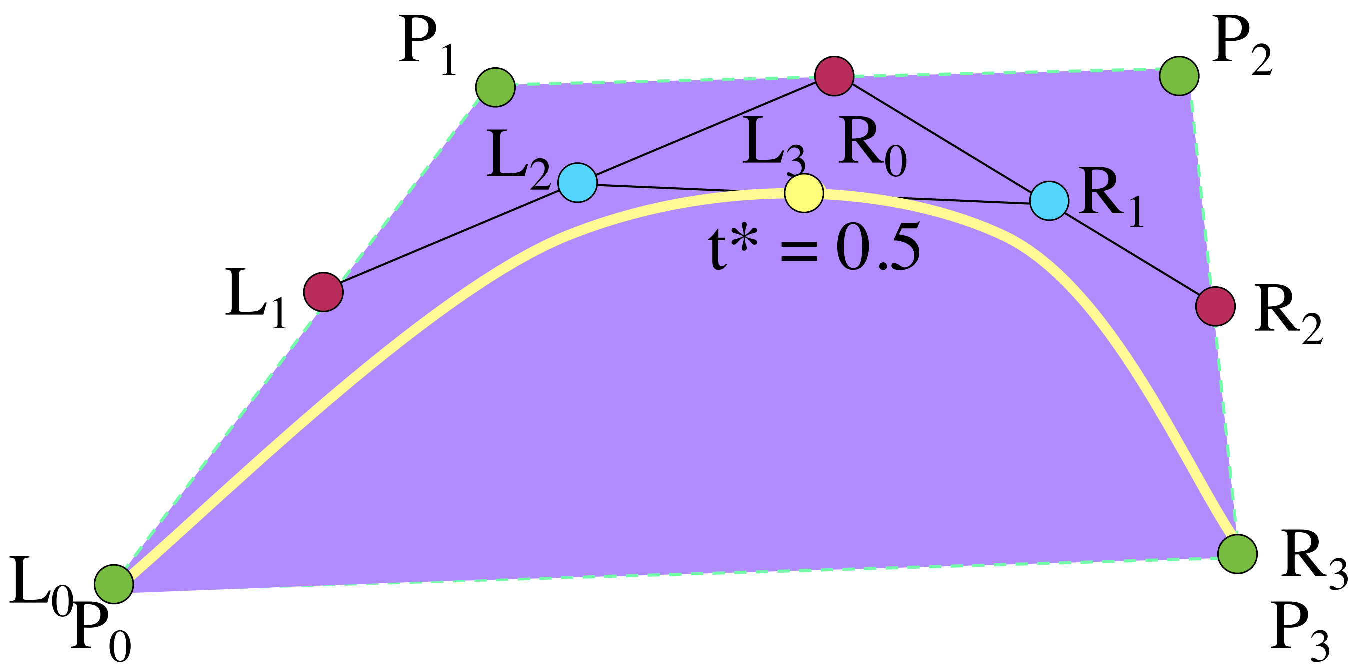
– Recursively split spline
  - stop when polygon is within epsilon of curve



[FvDFH]

De Casteljau algorithm

- Adaptive subdivision!

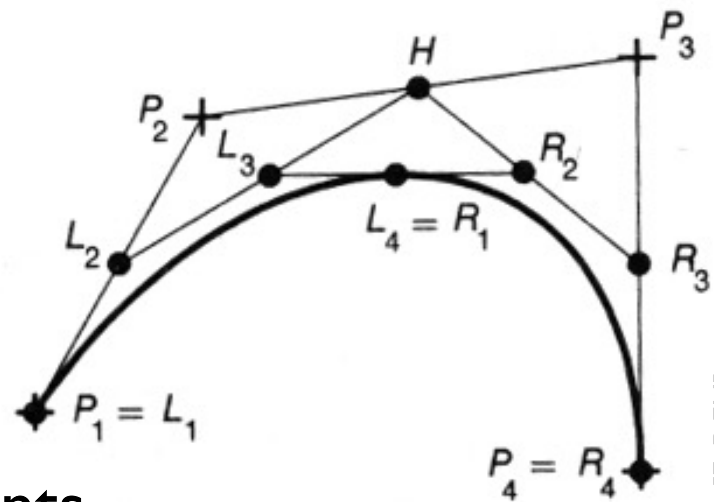


Recursive algorithm

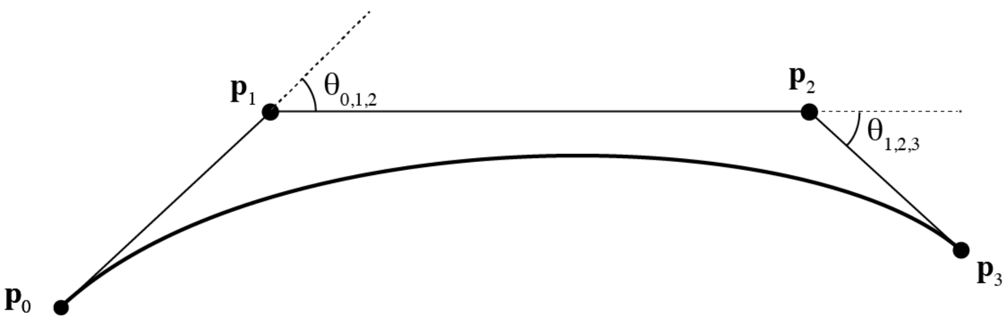
```
void DrawRecBezier (float eps) {
  if Linear (curve, eps)
    DrawLine (curve);
  else
    SubdivideCurve (curve, leftC, rightC);
    DrawRecBezier (leftC, eps);
    DrawRecBezier (rightC, eps);
}
```

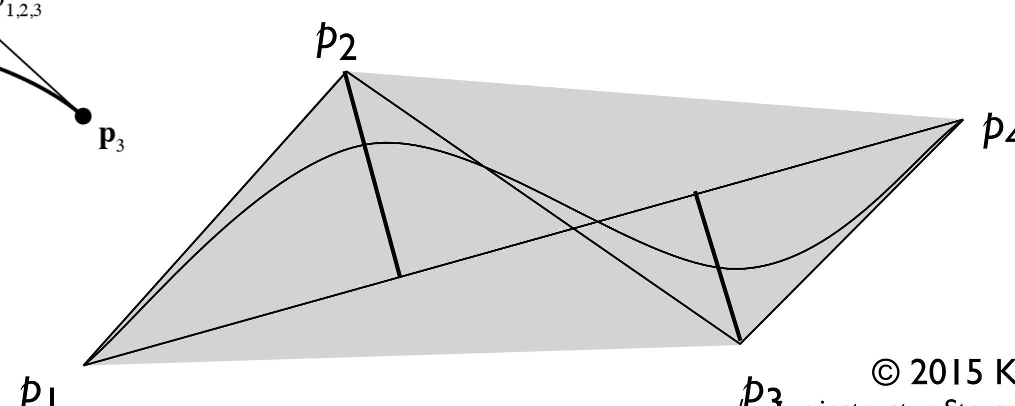
Evaluating by subdivision

– Recursively split spline
  - stop when polygon is within epsilon of curve
– Termination criteria
  - distance between control points
  - distance of control points from line
  - angles in control polygon



[FvDFH]





Cubic Bézier splines

- Very widely used type, especially in 2D
  – e.g. it is a primitive in PostScript/PDF
- Nice de Casteljau recurrence for evaluation

Spline Curves

Chaining spline segments

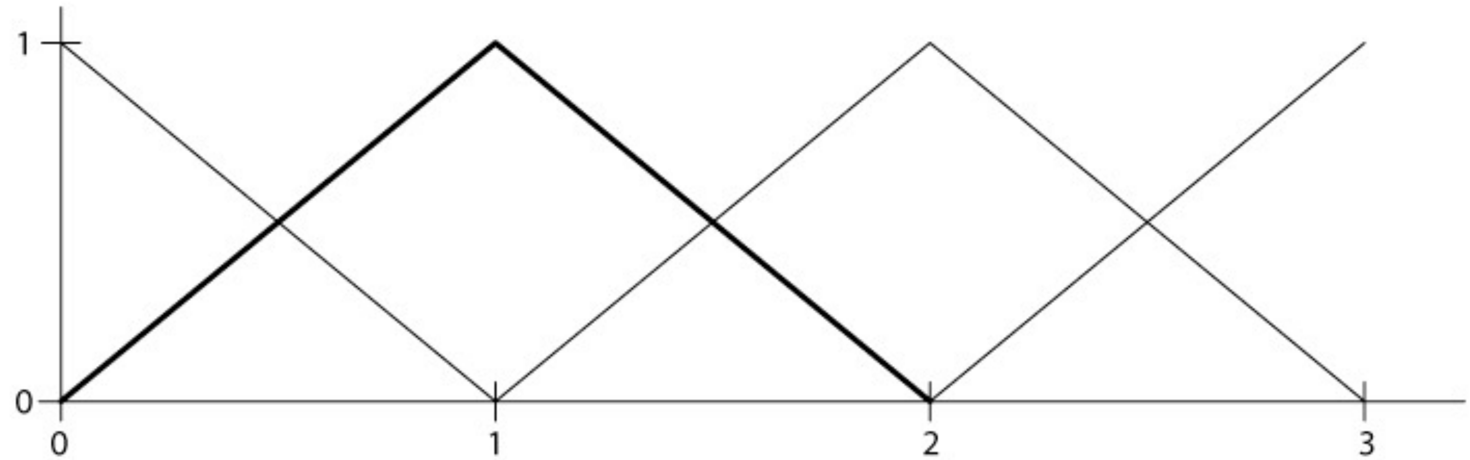
- Can only do so much with a single polynomial
- Can use these functions as segments of a longer curve
  – curve from t = 0 to t = 1 defined by first segment
  – curve from t = 1 to t = 2 defined by second segment

$$\mathbf{f}(t) = \mathbf{f}_i(t - i) \quad \text{for } i \leq t \leq i + 1$$

- To avoid discontinuity, match derivatives at junctions
  – this produces a C^1 curve

Trivial example: piecewise linear

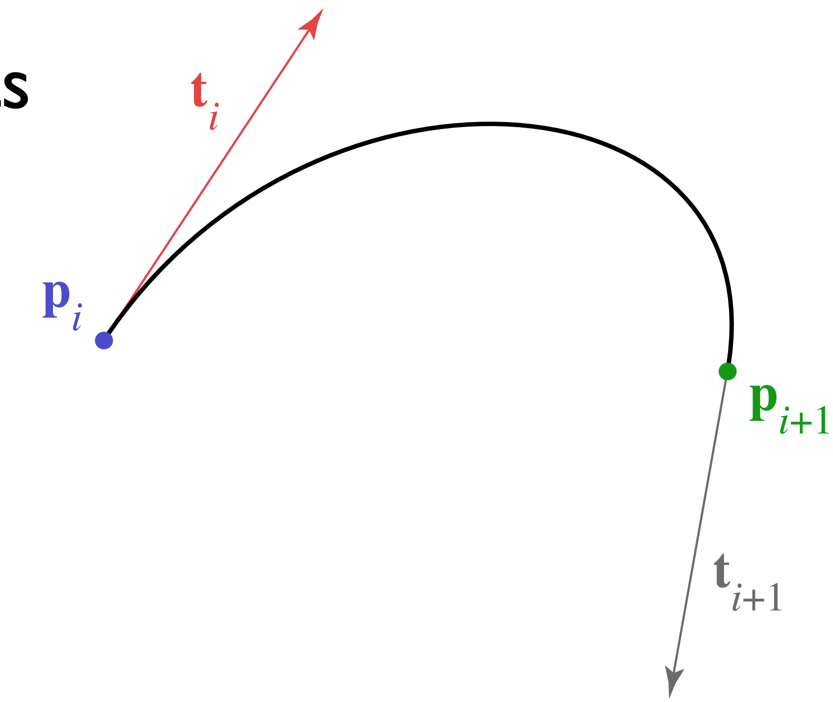
- Basis function formulation: "function times point"
  – basis functions: contribution of each point as *t* changes



  – can think of them as blending functions glued together

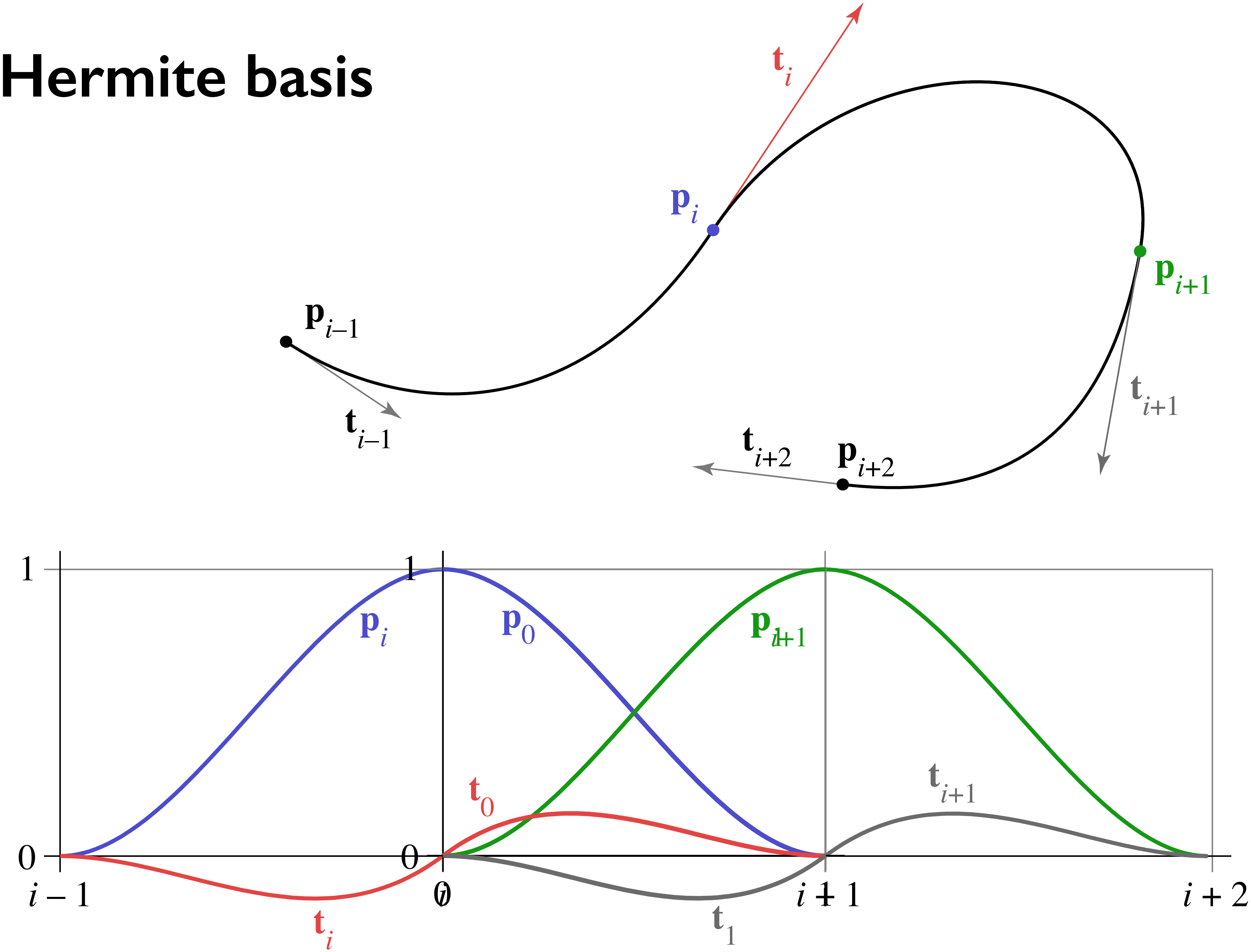
Hermite splines

- Constraints are endpoints and endpoint tangents



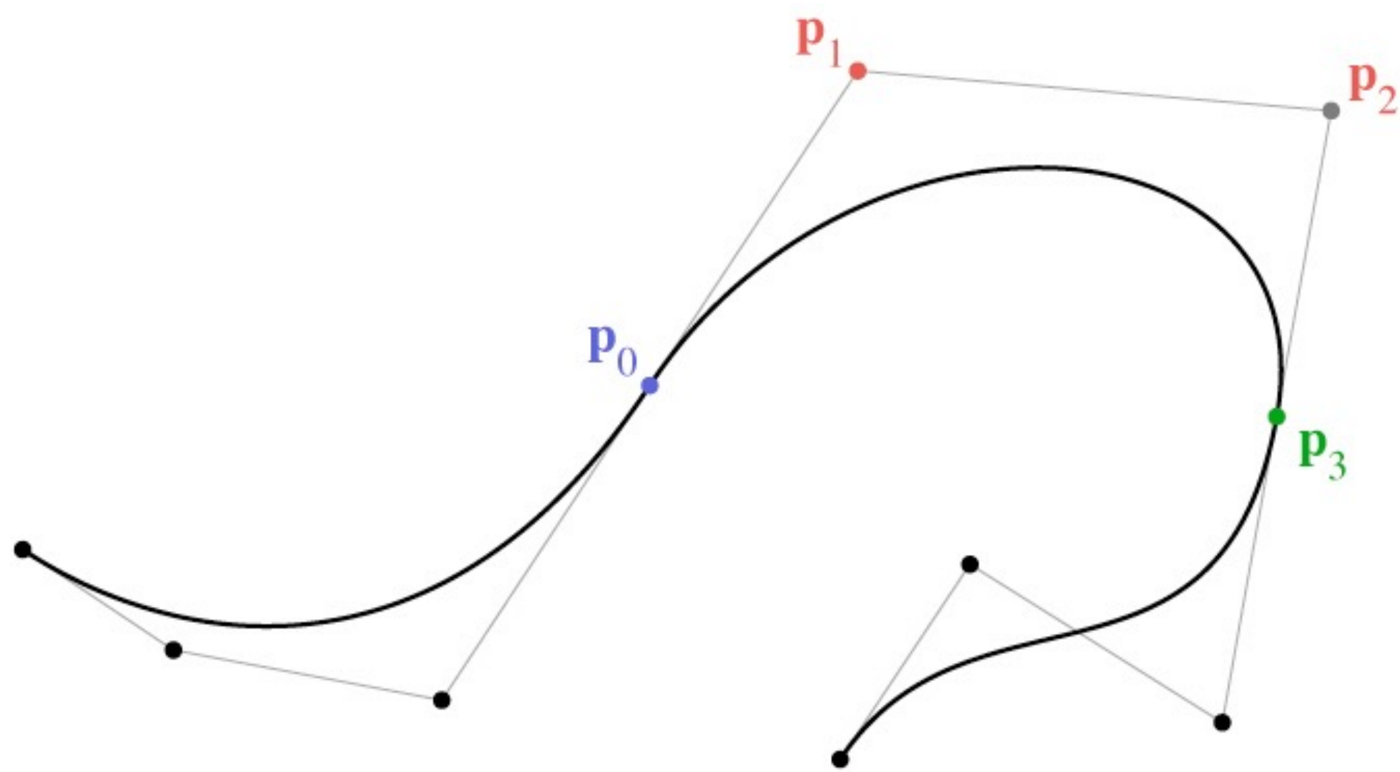
$$\mathbf{f}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 2 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix}$$

Hermite basis

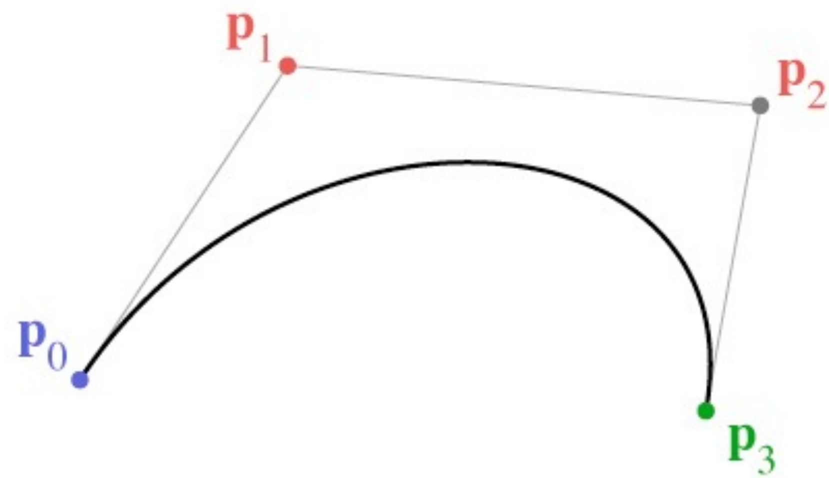


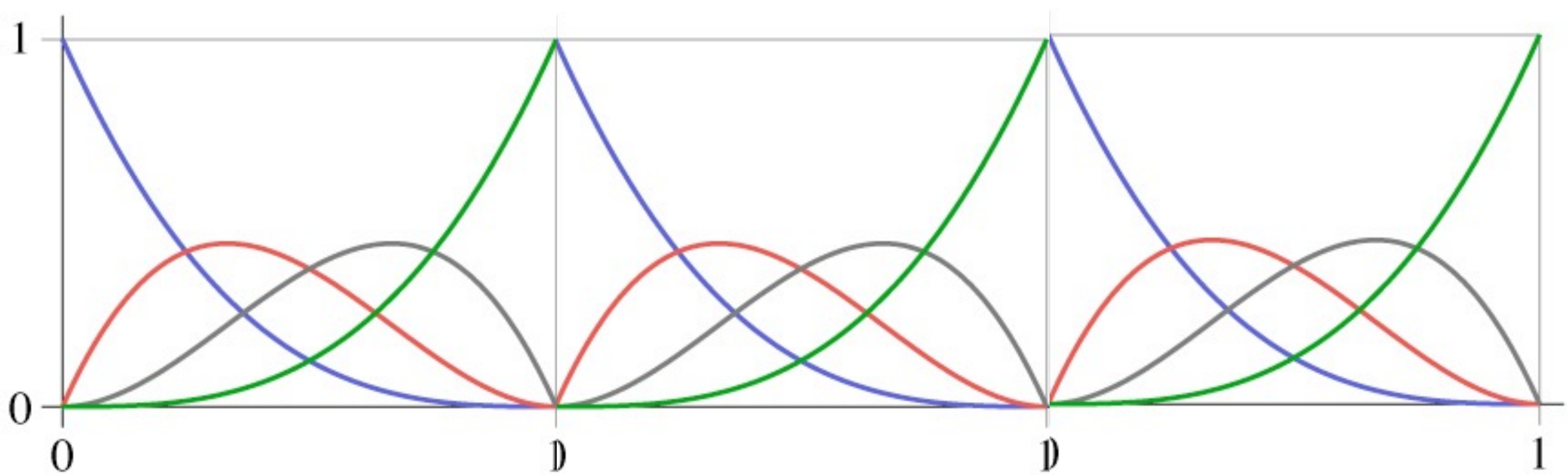
Chaining Bézier splines

- No continuity built in
- Achieve C^1 using collinear control points



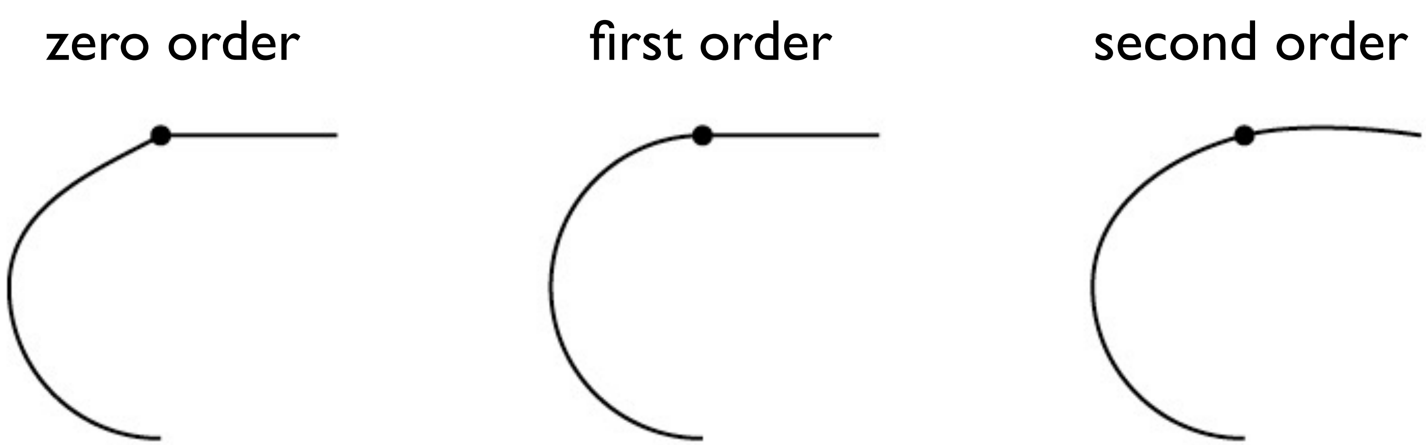
Bézier basis





Continuity

- Smoothness can be described by degree of continuity
  - zero-order (C^0): position matches from both sides
  - first-order (C^1): tangent matches from both sides
  - second-order (C^2): curvature matches from both sides
- G^n vs. C^n



Continuity

- Parametric continuity (*C*) of spline is continuity of coordinate functions
- Geometric continuity (*G*) is continuity of the curve itself
- Neither form of continuity is guaranteed by the other
  – Can be C^1 but not G^1 when $\mathbf{p}(t)$ comes to a halt (next slide)
  – Can be G^1 but not C^1 when the tangent vector changes length abruptly