2D Spline Curves

CS 4620 Lecture 28

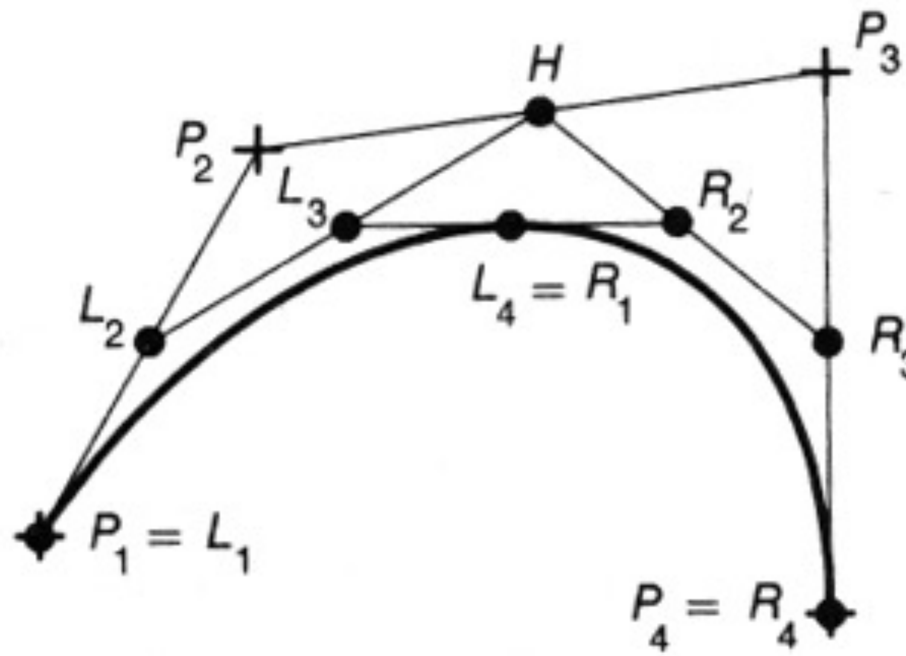
Administration

- A4 and PPA2 demos
  - Together on Monday
  - Please sign up

de Casteljau's algorithm

- A recurrence for computing points on Bézier spline segments:

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



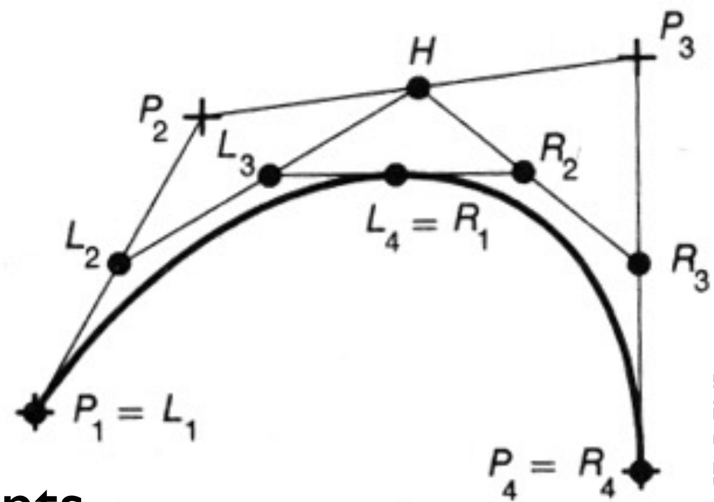
- Cool additional feature: also subdivides the segment into two shorter ones

[FvDFH]

Recursive algorithm

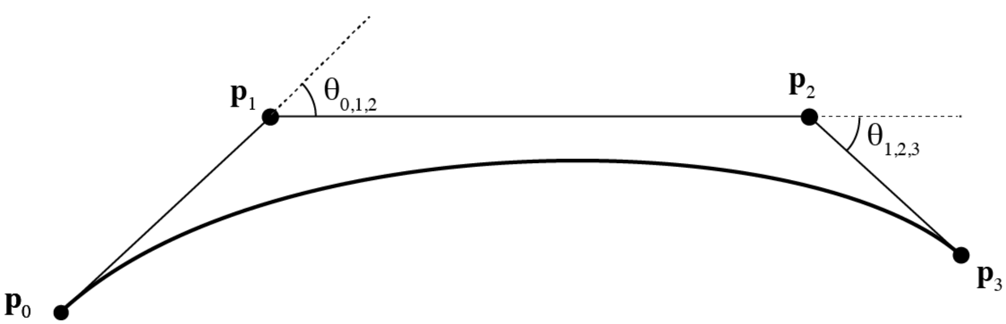
```
void DrawRecBezier (float eps) {
  if Linear (curve, eps)
    DrawLine (curve);
  else
    SubdivideCurve (curve, leftC, rightC);
    DrawRecBezier (leftC, eps);
    DrawRecBezier (rightC, eps);
}
```

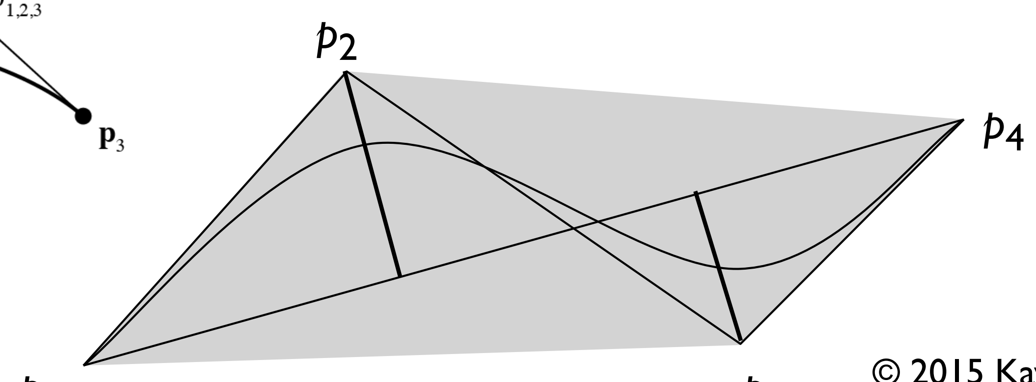
Evaluating by subdivision



[FvDFH]

- 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





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

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

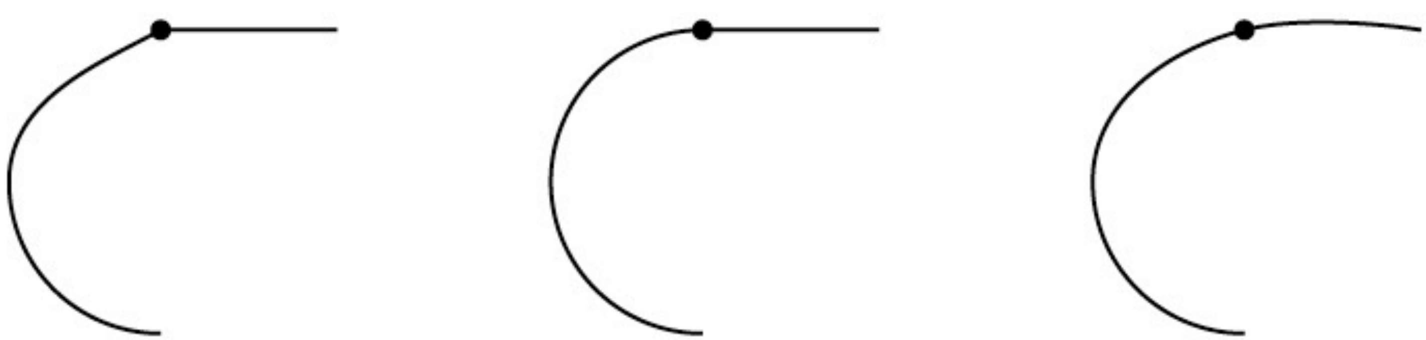
$$\mathbf{f}(t) = \mathbf{f}_i(t - i) \quad \text{for } i \leq t \leq i + 1$$

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

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

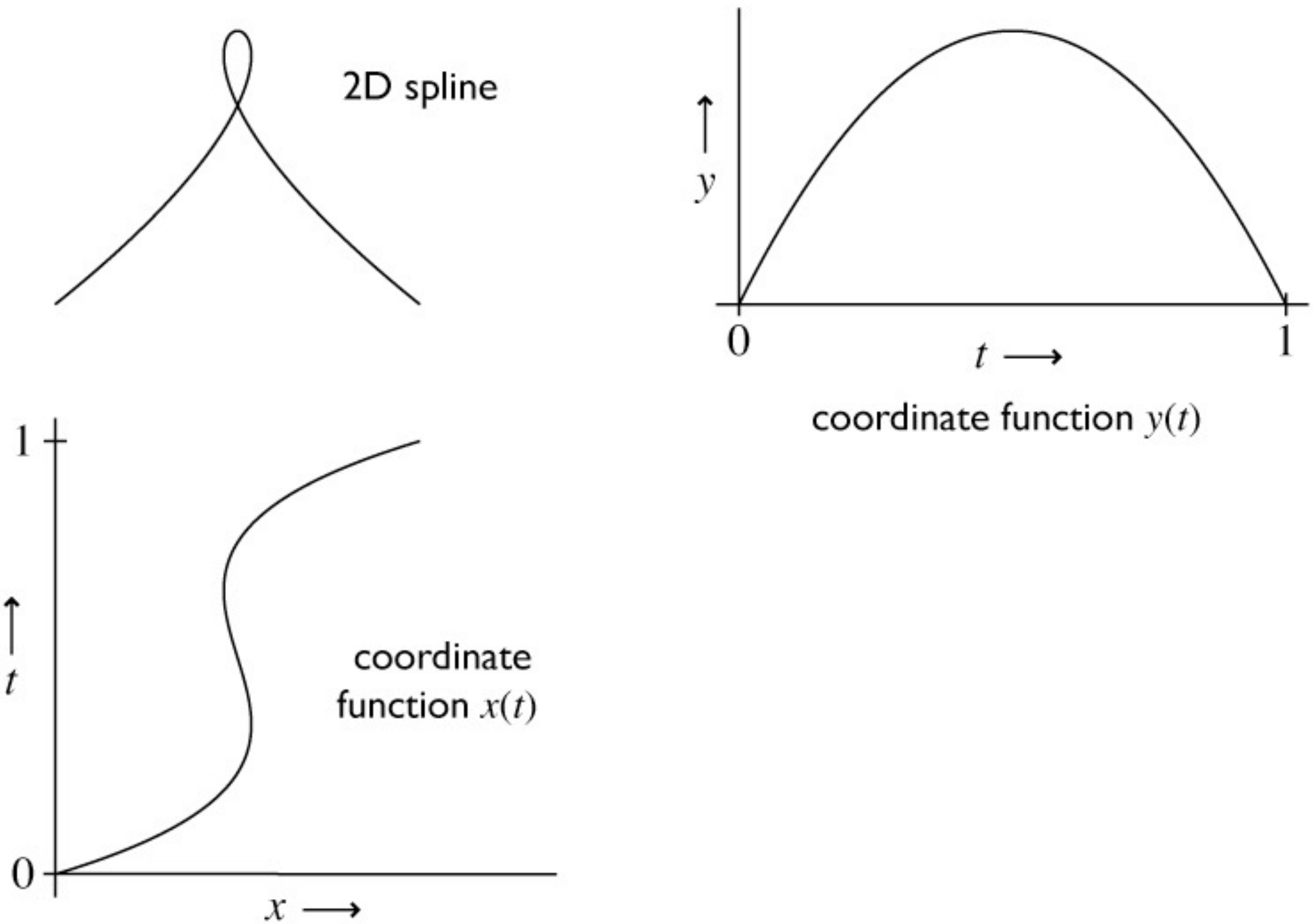
zero order first order second order



Continuity

- Parametric continuity (*C*) of spline is continuity of coordinate functions
  - f1'(1) = f2'(0)
- Geometric continuity (*G*) is continuity of the curve itself
  - f1'(1) = k f2'(0) for some k
  - Derivatives have same direction, but may have diff magnitude
  - – Generally G is less restrictive than C
  - – Can be G^1 but not C^1 when the tangent vector changes length
- 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)

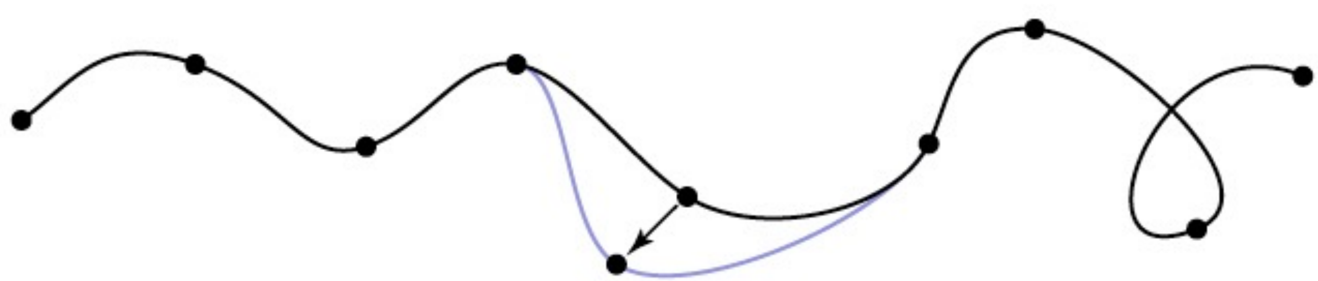
Geometric vs. parametric continuity



Properties

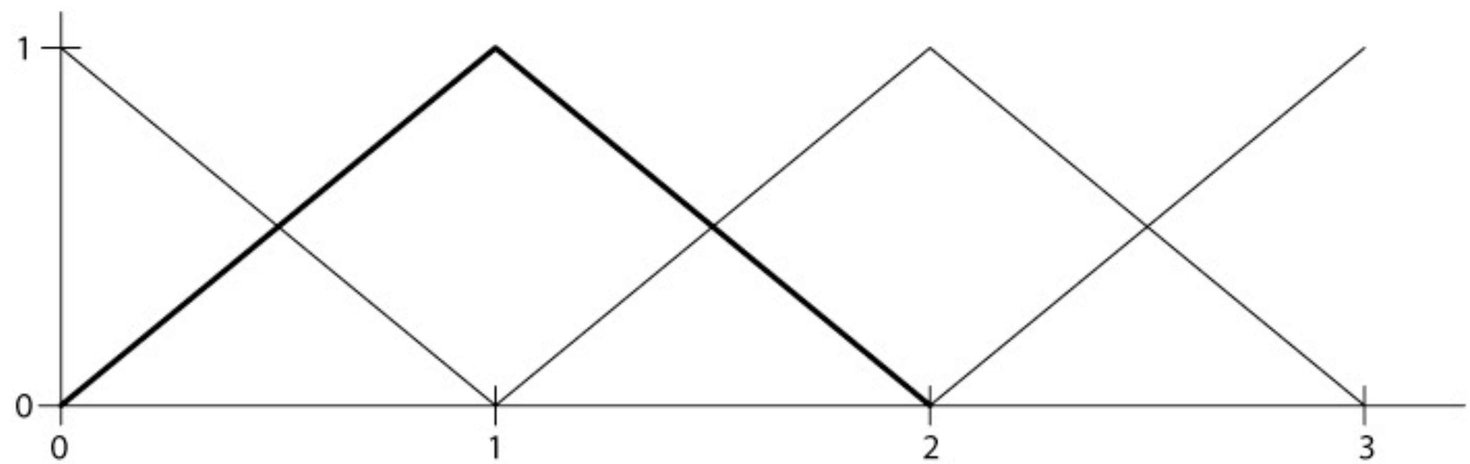
Control

- Local control
  – changing control point only affects a limited part of spline
  – without this, splines are very difficult to use
  – many likely formulations lack this
    - polynomial fits



Trivial example: piecewise linear

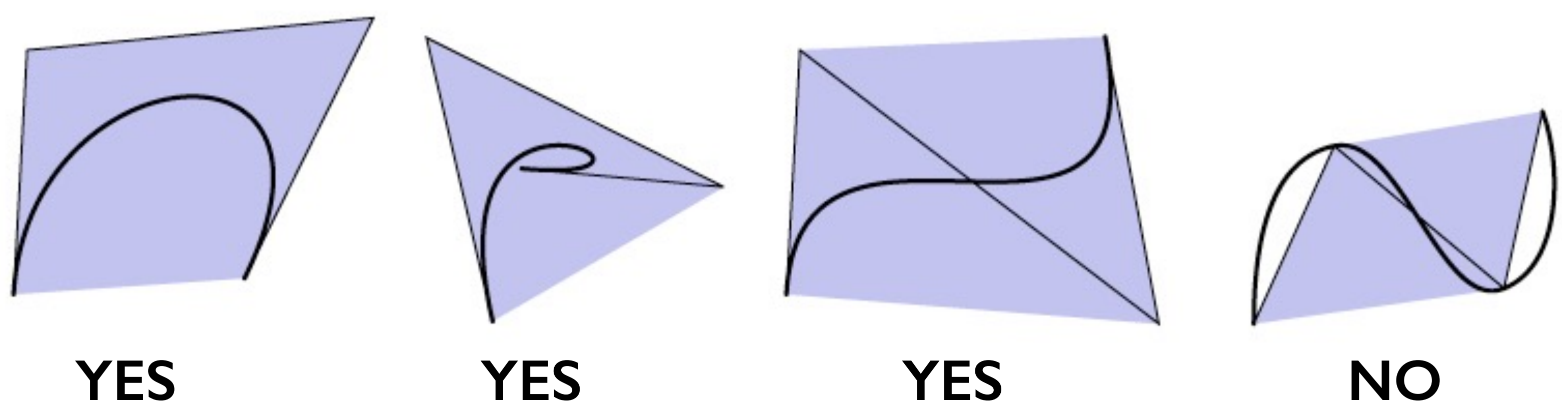
- Basis function formulation: "function times point"
  - basis functions: contribution of each point as *t* changes



  - can think of them as blending functions glued together

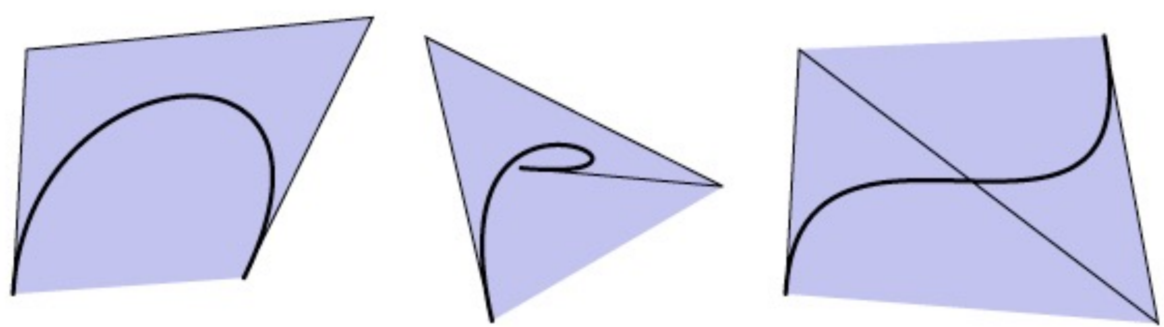
Control

- Convex hull property
  - convex hull = smallest convex region containing points
    - think of a rubber band around some pins
  - some splines stay inside convex hull of control points
    - make clipping, culling, picking, etc. simpler



Convex hull

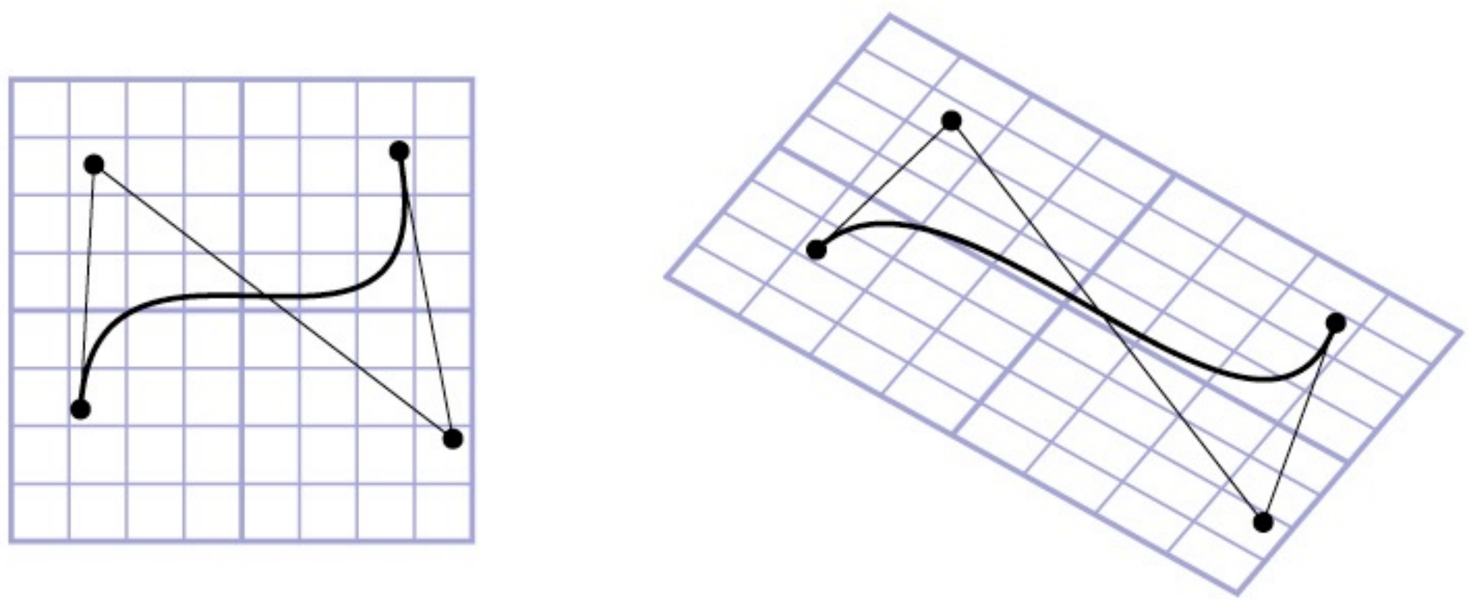
- If basis functions are all positive, the spline has the convex hull property
  – we require them to sum to 1



  – if any basis function is ever negative, no convex hull prop.

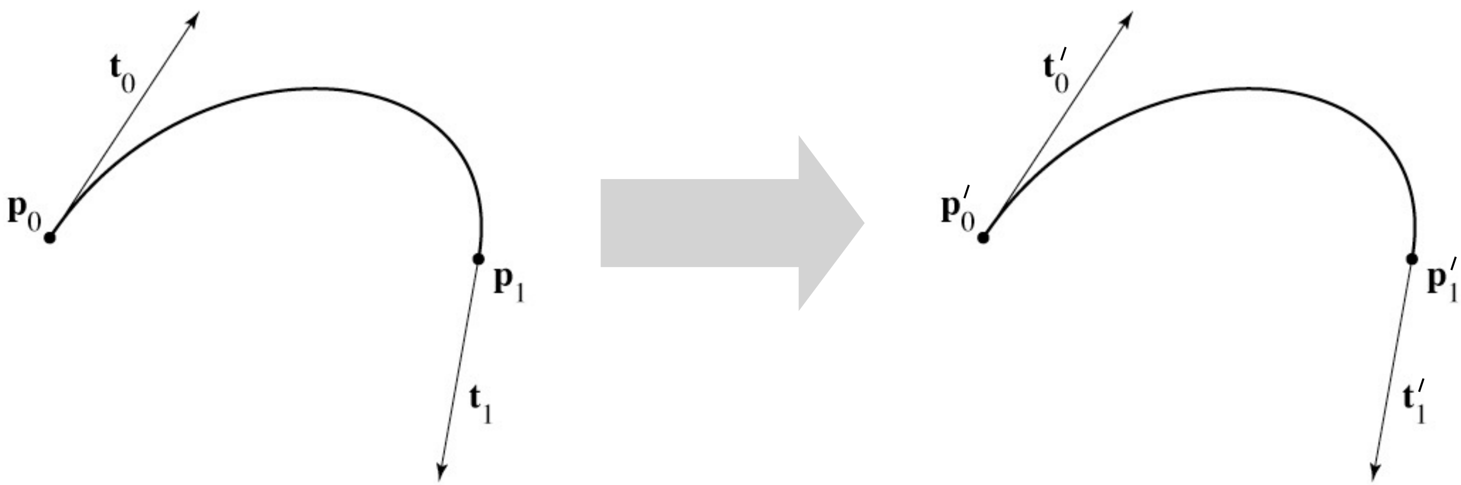
Affine invariance

- Transforming the control points is the same as transforming the curve
  - true for all commonly used splines
  - extremely convenient in practice…



Affine invariance

- Basis functions associated with points should always sum to 1



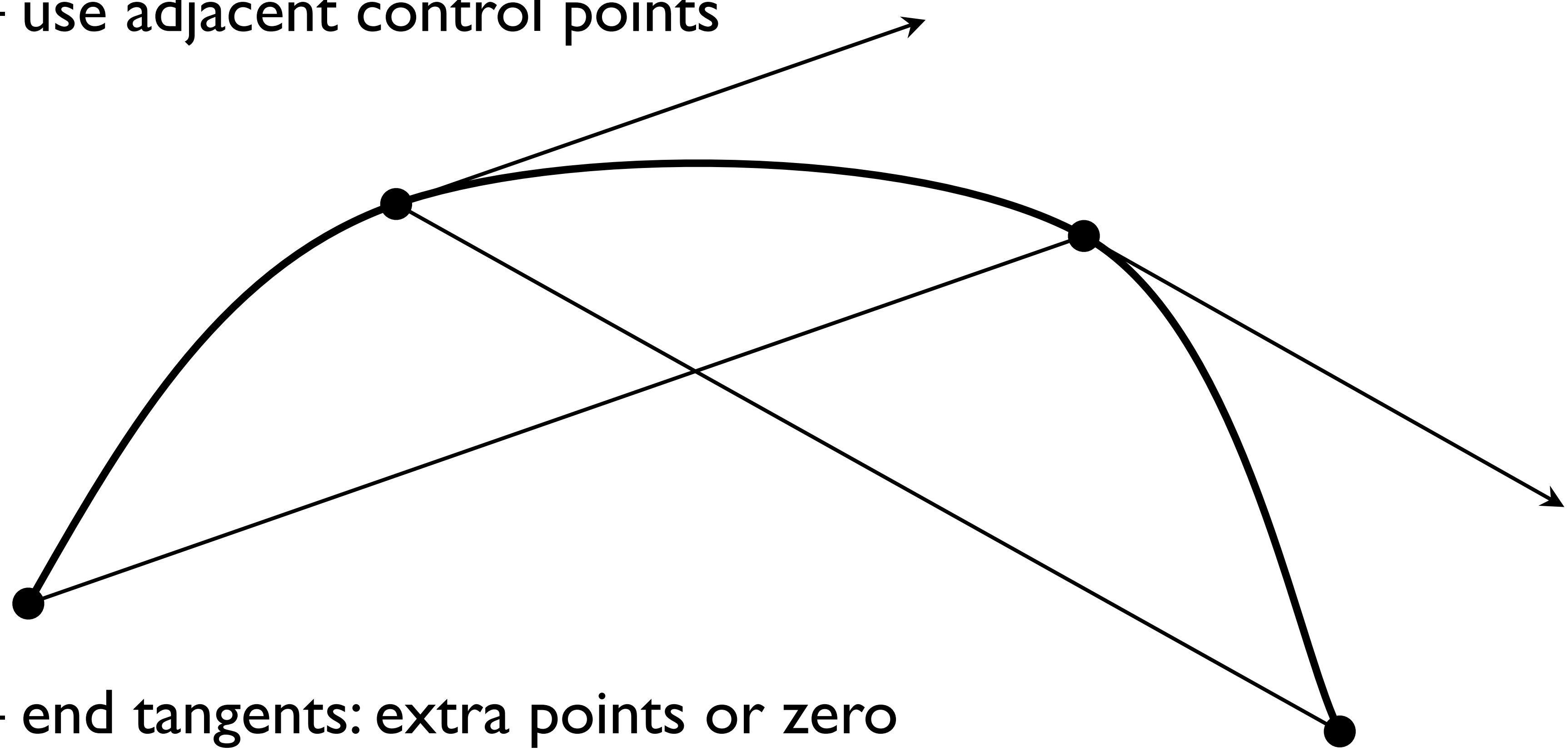
$$
\begin{aligned}
\mathbf{p}(t) &= b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 \\
\mathbf{p}'(t) &= b_0(\mathbf{p}_0 + \mathbf{u}) + b_1(\mathbf{p}_1 + \mathbf{u}) + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 \\
&= b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 + (b_0 + b_1)\mathbf{u} \\
&= \mathbf{p}(t) + \mathbf{u}
\end{aligned}
$$

Chaining spline segments

- Hermite curves are convenient because they can be made long easily
- Bézier curves are convenient because their controls are all points
  - but it is fussy to maintain continuity constraints
  - and they interpolate every 3rd point, which is a little odd
- We derived Bézier from Hermite by defining tangents from control points
  - a similar construction leads to the interpolating *Catmull-Rom* spline

Hermite to Catmull-Rom

- Have not yet seen any interpolating splines
- Would like to define tangents automatically
  – use adjacent control points



  – end tangents: extra points or zero

Hermite to Catmull-Rom

- Tangents are $(\mathbf{p}_{k+1} - \mathbf{p}_{k-1}) / 2$
  – scaling based on same argument about collinear case

$$\mathbf{p}_0 = \mathbf{q}_k$$
$$\mathbf{p}_1 = \mathbf{q}_k + 1$$
$$\mathbf{v}_0 = 0.5(\mathbf{q}_{k+1} - \mathbf{q}_{k-1})$$
$$\mathbf{v}_1 = 0.5(\mathbf{q}_{k+2} - \mathbf{q}_K)$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -.5 & 0 & .5 & 0 \\ 0 & -.5 & 0 & .5 \end{bmatrix} \begin{bmatrix} \mathbf{q}_{k-1} \\ \mathbf{q}_k \\ \mathbf{q}_{k+1} \\ \mathbf{q}_{k+2} \end{bmatrix}$$

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

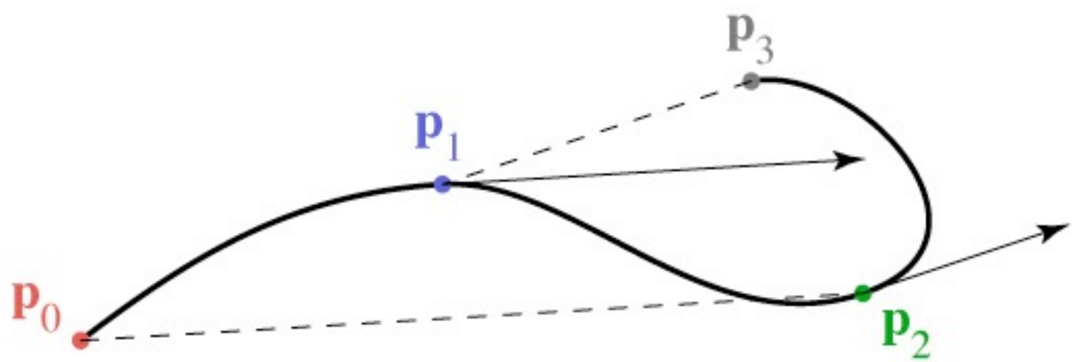
Hermite to Catmull-Rom

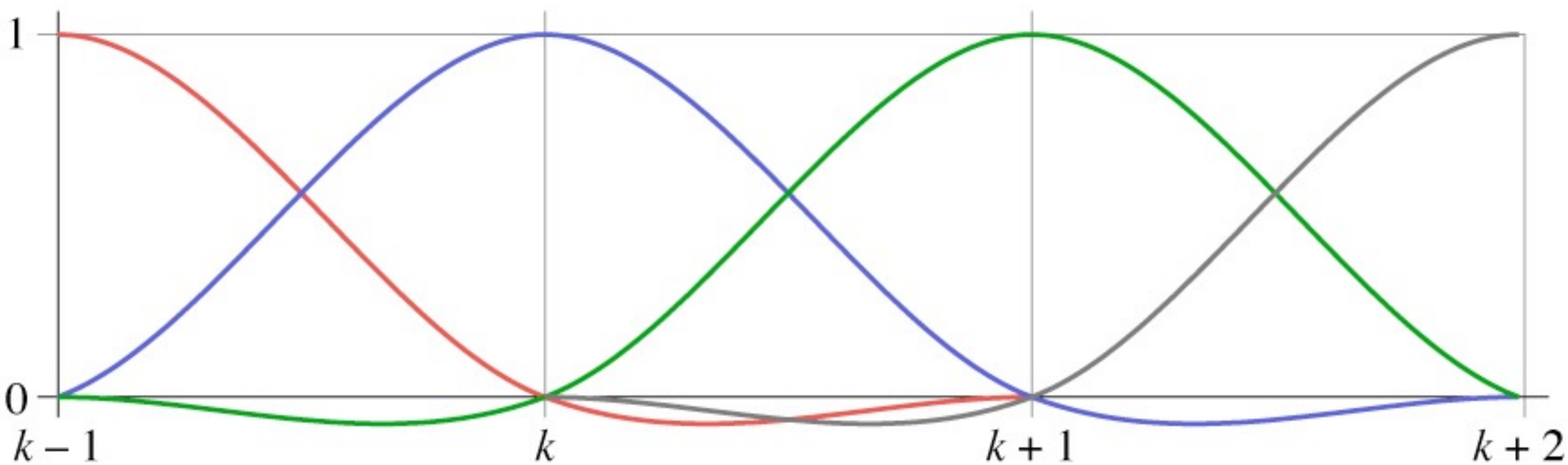
- Tangents are $(\mathbf{p}_{k+1} - \mathbf{p}_{k-1}) / 2$
  - scaling based on same argument about collinear case

$$\mathbf{p}_0 = \mathbf{q}_k$$
$$\mathbf{p}_1 = \mathbf{q}_k + 1$$
$$\mathbf{v}_0 = 0.5(\mathbf{q}_{k+1} - \mathbf{q}_{k-1})$$
$$\mathbf{v}_1 = 0.5(\mathbf{q}_{k+2} - \mathbf{q}_K)$$

$$\begin{bmatrix}\mathbf{a}\\\mathbf{b}\\\mathbf{c}\\\mathbf{d}\end{bmatrix} = \begin{bmatrix}2 & -2 & 1 & 1\\-3 & 3 & -2 & -1\\0 & 0 & 1 & 0\\1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\-.5 & 0 & .5 & 0\\0 & -.5 & 0 & .5\end{bmatrix}\begin{bmatrix}\mathbf{q}_{k-1}\\\mathbf{q}_k\\\mathbf{q}_{k+1}\\\mathbf{q}_{k+2}\end{bmatrix}$$

Catmull-Rom basis





Catmull-Rom splines

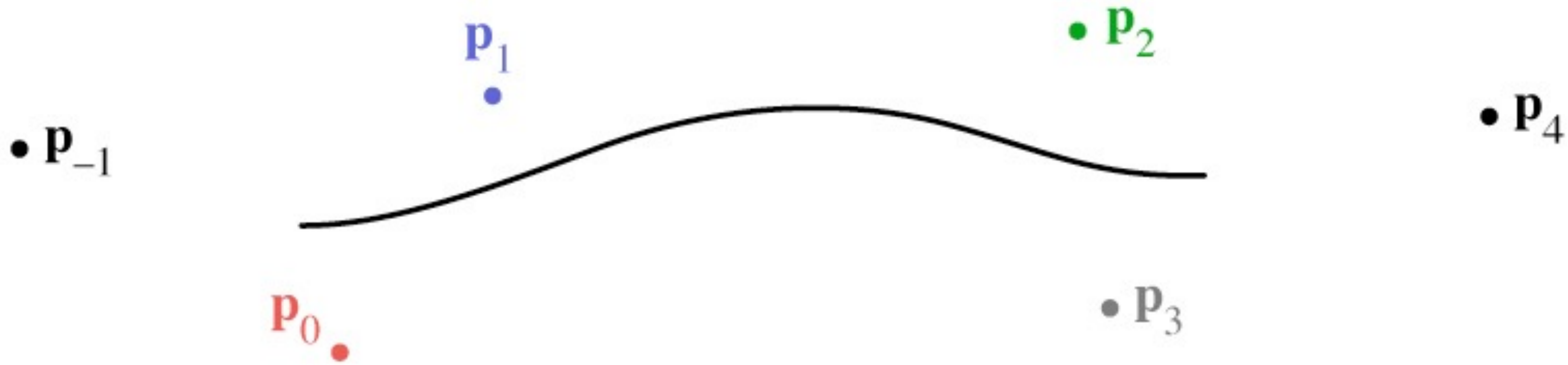
- Our first example of an interpolating spline
- Like Bézier, equivalent to Hermite

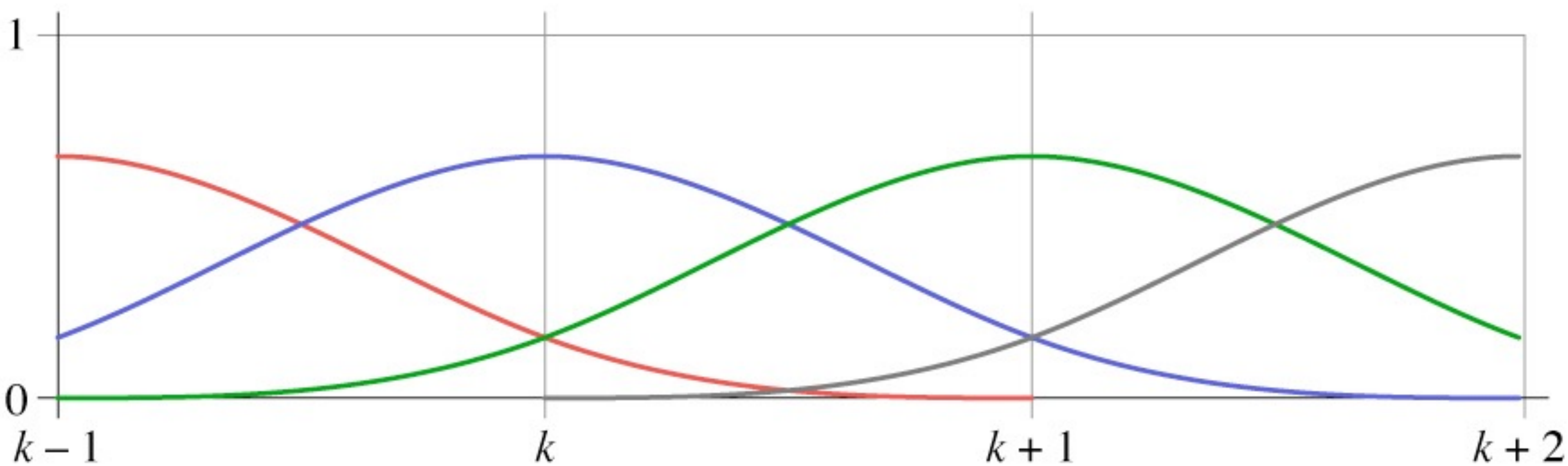
- First example of a spline based on just a control point sequence
- Does not have convex hull property

B-splines

- We may want more continuity than C^1
- We may not need an interpolating spline
- B-splines are a clean, flexible way of making long splines with arbitrary order of continuity

Cubic B-spline basis





Deriving the B-Spline

- Approached from a different tack than Hermite-style constraints
  - Want a cubic spline; therefore 4 active control points
  - Want C^2 continuity
  - Turns out that is enough to determine everything

Efficient construction of any B-spline

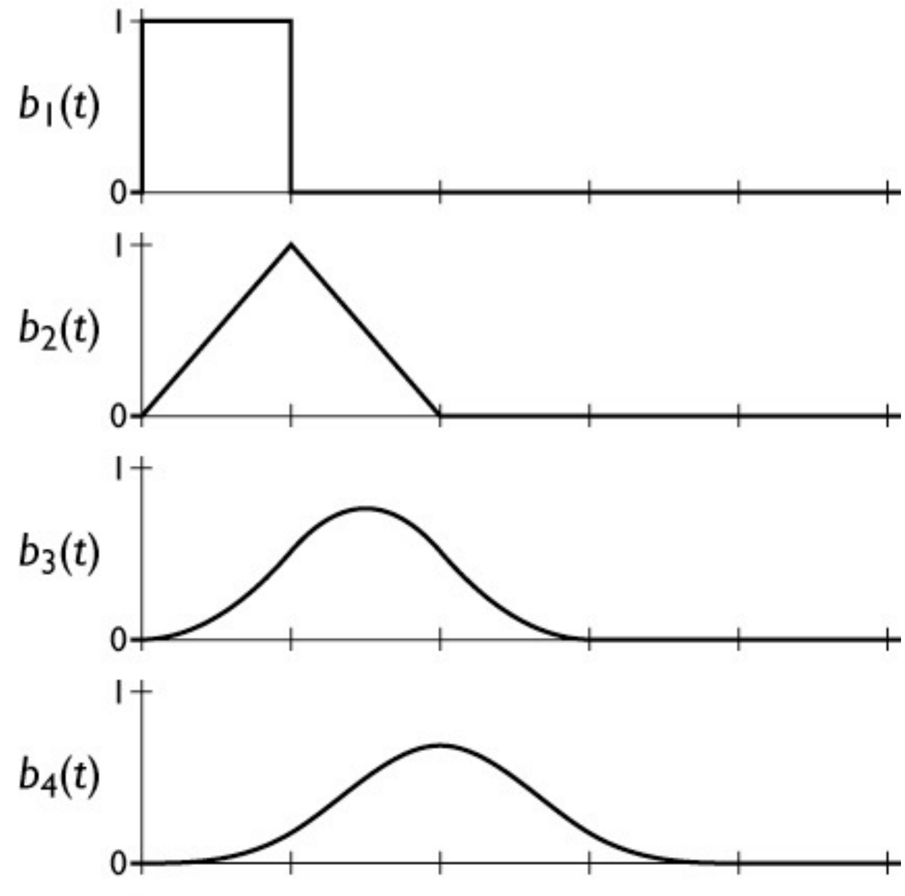
- B-splines defined for all orders
  - order *d*: degree *d* – 1
  - order *d*: *d* points contribute to value
- One definition: Cox-deBoor recurrence

$$b_1 = \begin{cases} 1 & 0 \le u < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$b_d = \frac{t}{d-1} b_{d-1}(t) + \frac{d-t}{d-1} b_{d-1}(t-1)$$

B-spline construction, alternate view

- Recurrence
  – ramp up/down
- Convolution
  – smoothing of basis fn
  – smoothing of curve



Cubic B-spline matrix

$$\mathbf{f}_i(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$

Bézier matrix

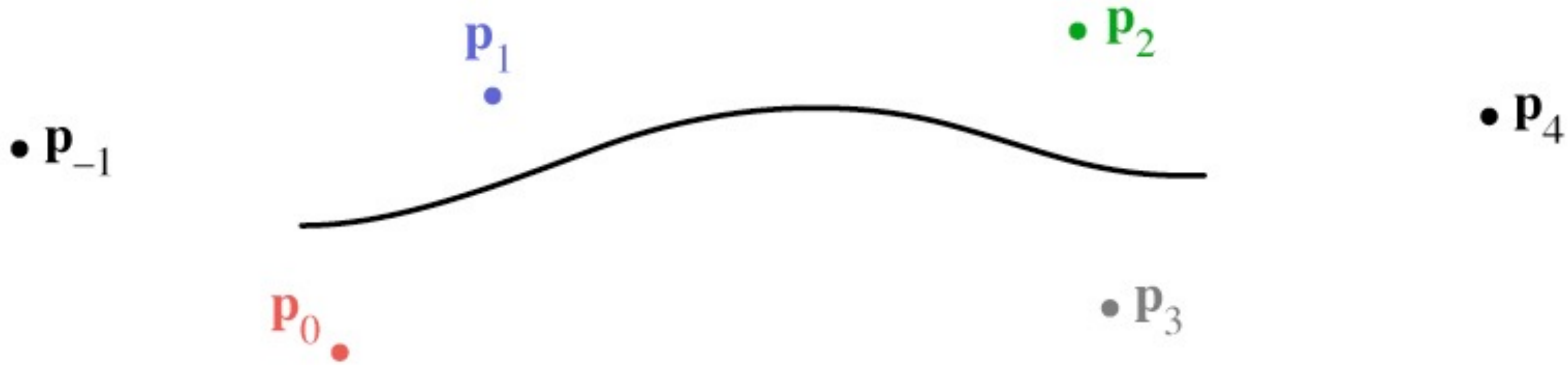
$$\mathbf{f}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

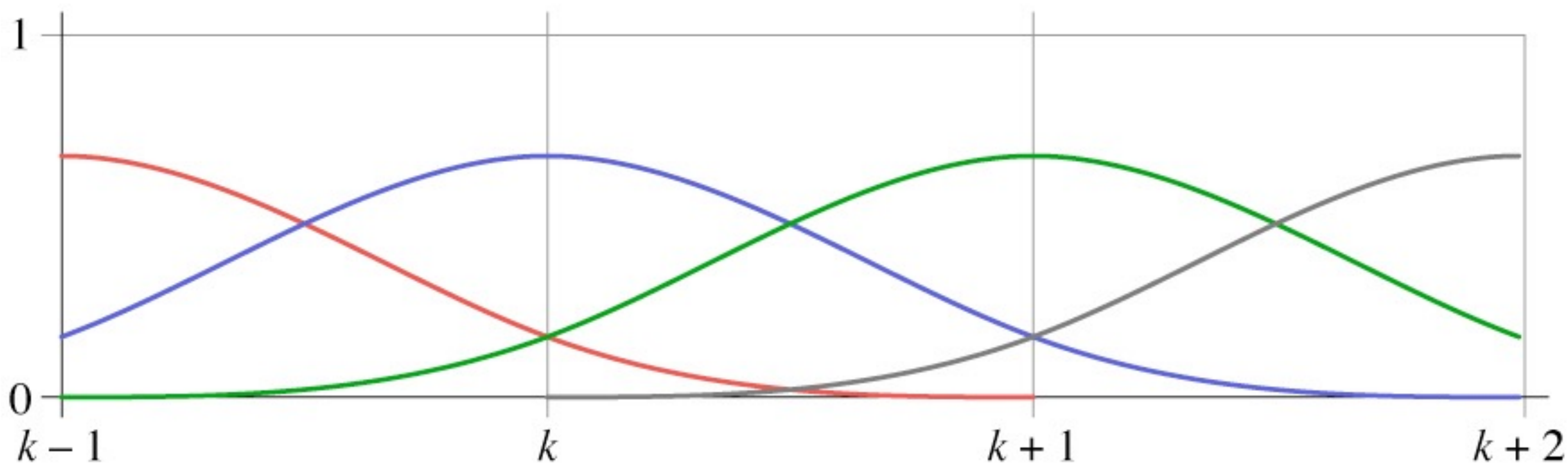
– note that these are the Bernstein polynomials

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

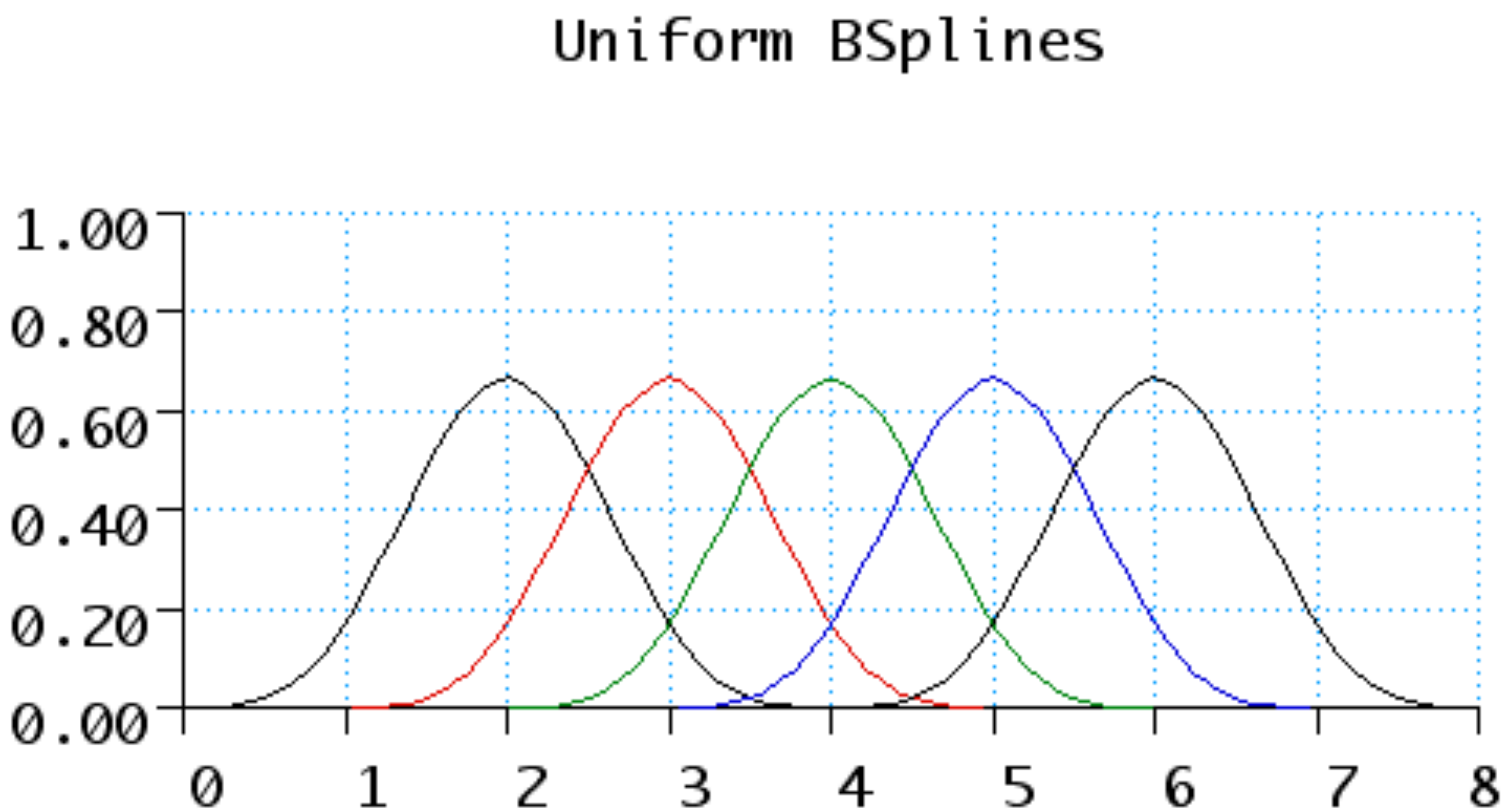
and that defines Bézier curves for any degree

Cubic B-spline basis





Over many segments



B-spline

- All points are same, no special points
- Basis functions are the same

Converting spline representations

- All the splines we have seen so far are equivalent
  - all represented by geometry matrices

$$\mathbf{p}_S(t) = T(t)M_S P_S$$

  - where *S* represents the type of spline
  - therefore the control points may be transformed from one type to another using matrix multiplication

$$P_1 = M_1^{-1} M_2 P_2$$

$$\begin{aligned}\mathbf{p}_1(t) &= T(t)M_1(M_1^{-1}M_2P_2) \\ &= T(t)M_2P_2 = \mathbf{p}_2(t)\end{aligned}$$

Other types of B-splines

- Nonuniform B-splines
  - discontinuities not evenly spaced
  - allows control over continuity or interpolation at certain points
  - e.g. interpolate endpoints (commonly used case)
- Nonuniform Rational B-splines (NURBS)
  - ratios of nonuniform B-splines: $x(t) / w(t); y(t) / w(t)$
  - key properties:
    - invariance under perspective as well as affine
    - ability to represent conic sections exactly

