Interpolating Curves

D.A. Forsyth UIUC

Central issues in modelling

- Construct families of curves, surfaces and volumes that
  - can represent common objects usefully;
  - are easy to interact with; interaction includes:
    - manual modelling;
    - fitting to measurements;
  - support geometric computations
    - intersection
    - collision
- Question: How much do you know about B-Splines?

Main topics

- Simple curves
- Interpolation
- Continuity and splines for interpolation

Parametric forms

- A parametric curve is
  - a mapping of one parameter into
    - 2D
    - 3D
  - Examples
    - circle as $(\cos t, \sin t)$
    - twisted cubic as $(t, t*t, t*t*t)$
    - circle as $(1-t^2, 2\, t, 0)/(1+t^2)$
  - domain of the parametrization MATTERS
    - $(\cos t, \sin t)$, $0 <= t <= pi$ is a semicircle

Curves - basic ideas

- Important cases on the plane
  - Monge (or explicit)
    - y(x)
    - Examples:
      - many lines, bits of circle, sines, etc
  - Implicit curve
    - F(x, y)=0
    - Examples:
      - all lines, circles, ellipses
      - any explicit curve; any parametric algebraic curve; lots of others
      - Important special case: F polynomial
  - Parametric curve
    - (x(s), y(s)) for s in some range
    - Examples
      - all lines, circles, ellipses
      - Important special cases: x, y polynomials, rational

Powerful view of a curve

- A set of points pasted together by blending functions
  - blending functions depend on parameter
  - points (control points; control vectors) don't
  - representation isn't unique (but that really doesn't matter very much)

$$\mathbf{p}_0\phi_0(t) + \mathbf{p}_1\phi_1(t) + \mathbf{v}_0\phi_2(t) + \mathbf{v}_1\phi_3(t)$$

- Advantage:
  - we don't need to worry much about dimension
    - that's carried by the points
    - we can do a variety of clever tricks with the blending functions
      - meet constraints
      - convert from form to form

Interpolation

- Construct a parametric curve that passes through (interpolates) a set of points.
- Lagrange interpolate:
  - give parameter values associated with each point
  - use Lagrange polynomials (one at the relevant point, zero at all others) to construct curve
  - curve is:

$$\boxed{\sum_{i \in \text{points}} p_i \phi_i^{(l)}(t)}$$

Lagrange interpolate

- Construct a parametric curve that passes through (interpolates) a set of points.
- Lagrange interpolate:
  - give parameter values associated with each point
  - use Lagrange polynomials (one at the relevant point, zero at all others) to construct curve
  - degree is (#pts-1)
    - e.g. line through two points
    - quadratic through three.
  -

Lagrange polynomials

- Interpolate points at s=s_i, i=1..n
- Blending functions

$$\phi_i(s) = \begin{cases} 1 & s = s_i \\ 0 & s = s_k, k \neq i \end{cases}$$

- Can do this with a polynomial

$$\frac{\prod_{j=1..i-1,i..n}(s - s_j)}{\prod_{j=1..i-1,i..n}(s_j - s_i)}$$

Fig 2.16a. Interpolation by a polynomial of degree 4.

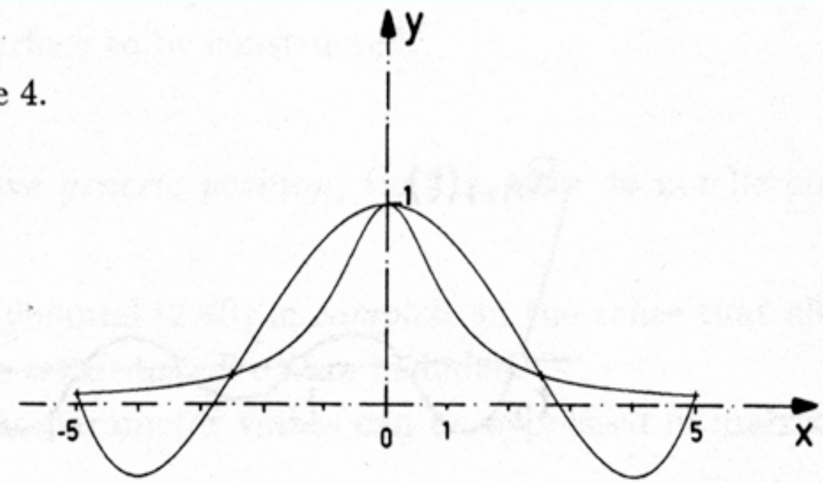
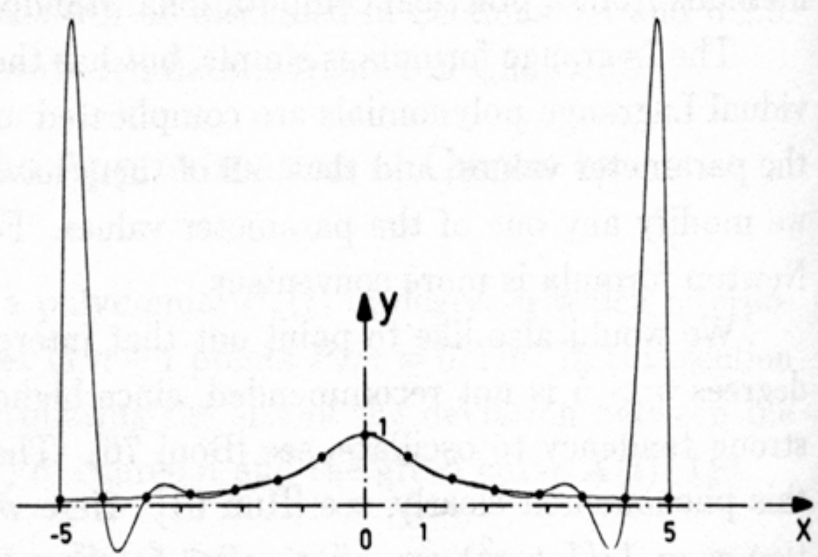


Fig 2.16c. Interpolation by a polynomial of degree 14.



Hermite interpolation

- Hermite interpolate
  - give parameter values and derivatives associated with each point
  - curve passes through given point and the given derivative at that parameter value
  - For two points (most important case) curve is:

$$\mathbf{p}_0\phi_0(t) + \mathbf{p}_1\phi_1(t) + \mathbf{v}_0\phi_2(t) + \mathbf{v}_1\phi_3(t)$$

- use Hermite polynomials to construct curve
  - one at some parameter value and zero at others or
  - derivative one at some parameter value, and zero at others

Hermite curves

- Natural matrix form:
  - solve linear system to get curve coefficients
- Easily “pasted” together

$$\mathbf{p}_0\phi_0(t) + \mathbf{p}_1\phi_1(t) + \mathbf{v}_0\phi_2(t) + \mathbf{v}_1\phi_3(t)$$

Blending functions are cubic polynomials, so we write as:

$$\left[\begin{array}{cccc} \phi_0(t) & \phi_1(t) & \phi_2(t) & \phi_3(t) \end{array}\right] = \left[\begin{array}{cccc} 1 & t & t^2 & t^3 \end{array}\right] \left\{\begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{array}\right\}$$

This allows us to write the curve as:

$$\left[\begin{array}{cccc} 1 & t & t^2 & t^3 \end{array}\right] \left\{\begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{array}\right\} \left\{\begin{array}{c} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{array}\right\}$$

Basis matrix Geometry matrix

Hermite polynomials

$$\left[\begin{array}{cccc} \phi_0(t) & \phi_1(t) & \phi_2(t) & \phi_3(t) \end{array}\right] = \left[\begin{array}{cccc} 1 & t & t^2 & t^3 \end{array}\right] \left\{\begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{array}\right\}$$

$$\frac{d}{dt}\left[\begin{array}{cccc} \phi_0(t) & \phi_1(t) & \phi_2(t) & \phi_3(t) \end{array}\right] = \left[\begin{array}{cccc} 0 & 1 & 2t & 3t^2 \end{array}\right] \left\{\begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{array}\right\}$$

Constraints

$$\begin{bmatrix} \phi_0(0) & \phi_1(0) & \phi_2(0) & \phi_3(0) \\ \phi_0(1) & \phi_1(1) & \phi_2(1) & \phi_3(1) \\ \frac{d\phi_0}{dt}(0) & \frac{d\phi_1}{dt}(0) & \frac{d\phi_2}{dt}(0) & \frac{d\phi_3}{dt}(0) \\ \frac{d\phi_0}{dt}(1) & \frac{d\phi_1}{dt}(1) & \frac{d\phi_2}{dt}(1) & \frac{d\phi_3}{dt}(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These constraints give:

Interpolate each endpoint
Have correct derivatives at each endpoint

We can write individual constraints like:

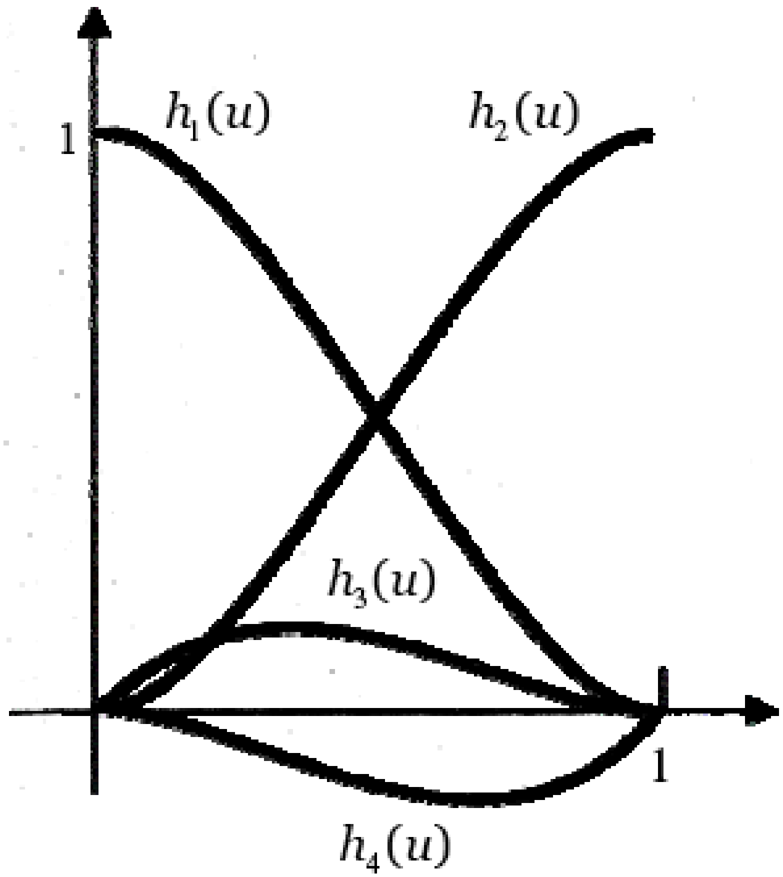
$$\left[\begin{array}{cccc}\phi_0(0) & \phi_1(0) & \phi_2(0) & \phi_3(0)\end{array}\right] = \left[\begin{array}{cccc}1 & 0 & 0^2 & 0^3\end{array}\right] \left\{\begin{array}{cccc}a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3\end{array}\right\}$$

To get:

$$\left[\begin{array}{cccc}1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3\end{array}\right] \left\{\begin{array}{cccc}a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3\end{array}\right\} = \left[\begin{array}{cccc}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{array}\right]$$

Hermite blending functions

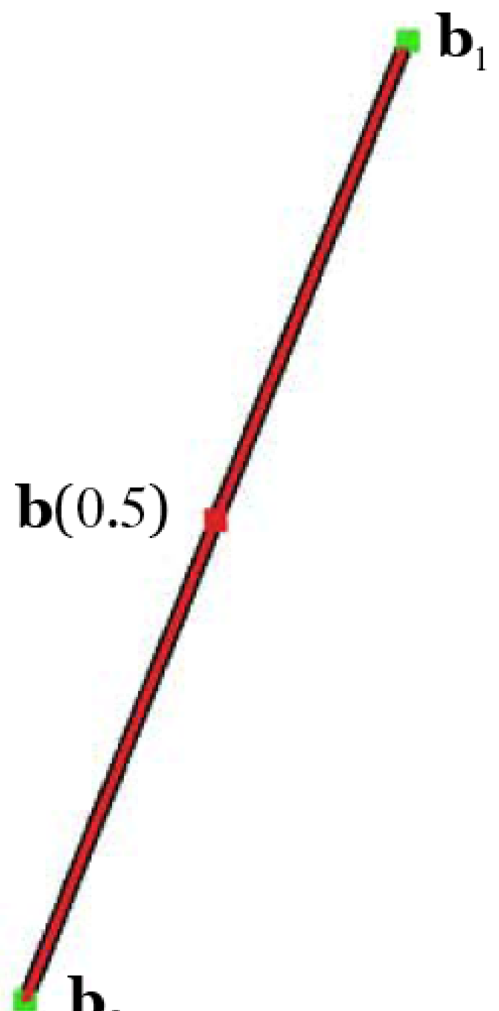
Hermite Blending Polynomials



$$h_1(u) = 2u^3 - 3u^2 + 1$$
$$h_2(u) = -2u^3 + 3u^2$$
$$h_3(u) = u^3 - 2u^2 + u$$
$$h_4(u) = u^3 - u^2$$

Bezier curves

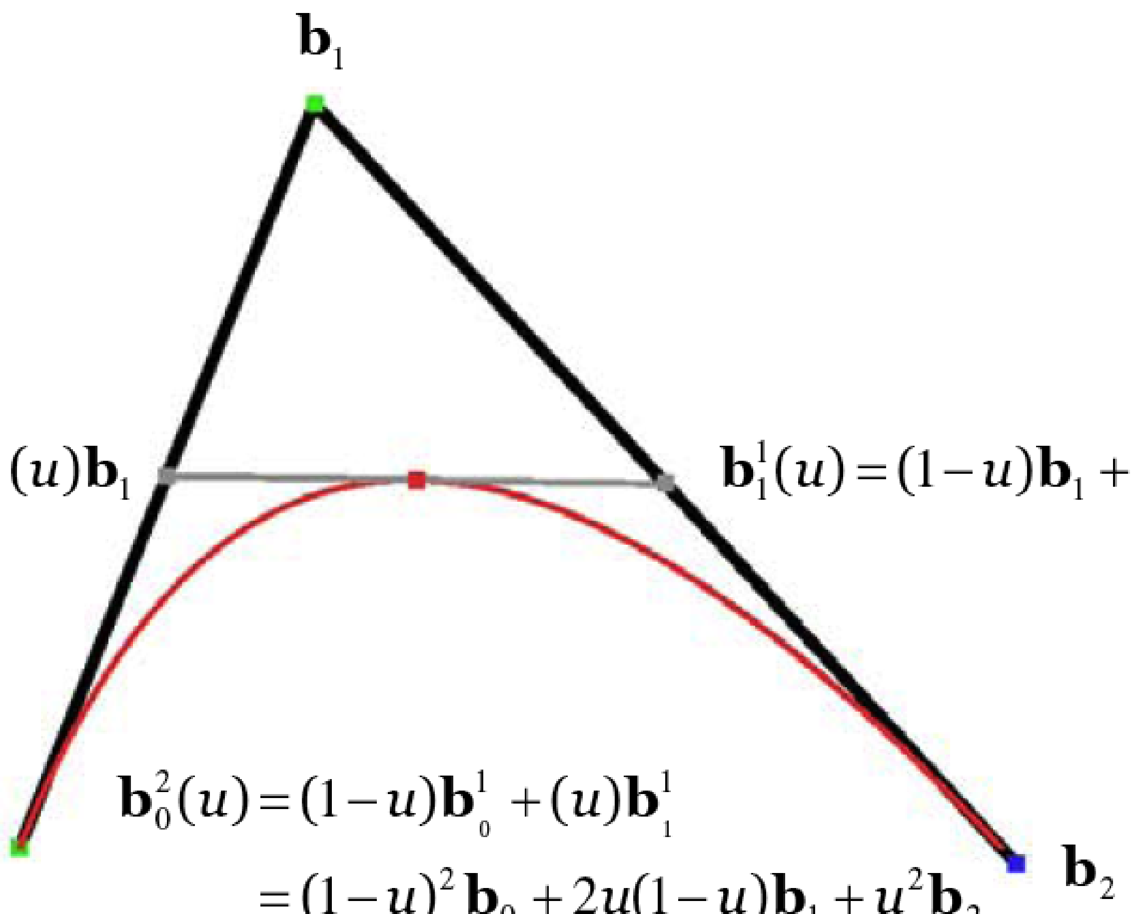
Linear Interpolation



$$\mathbf{b}(u) = (1-u)\mathbf{b}_0 + (u)\mathbf{b}_1 \qquad \text{where } 0 \le u \le 1$$

Bezier curves

"Doubled" Linear Interpolation



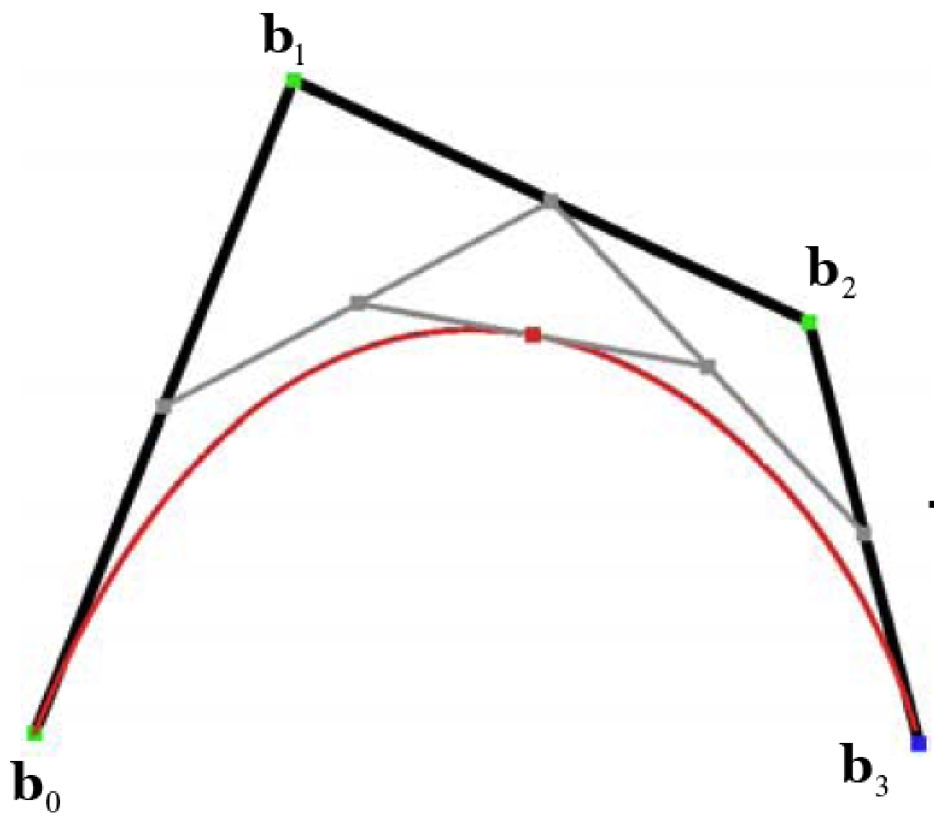
$$\mathbf{b}_0^1(u) = (1-u)\mathbf{b}_0 + (u)\mathbf{b}_1$$

$$\mathbf{b}_1^1(u) = (1-u)\mathbf{b}_1 + (u)\mathbf{b}_2$$

$$\begin{aligned}\mathbf{b}_0^2(u) &= (1-u)\mathbf{b}_0^1 + (u)\mathbf{b}_1^1 \\ &= (1-u)^2\mathbf{b}_0 + 2u(1-u)\mathbf{b}_1 + u^2\mathbf{b}_2\end{aligned}$$

Bezier curves

“Tripled” Linear Interpolation



Get a cubic polynomial curve

$$
\begin{aligned}
\mathbf{b}_0^3(u) = \; & (1-u)^3\,\mathbf{b}_0 \\
& +3(1-u)^2(u)\mathbf{b}_1 \\
& +3(1-u)(u)^2\,\mathbf{b}_2 \\
& +(u)^3\,\mathbf{b}_3
\end{aligned}
$$

This is a cubic Bézier curve

Bezier curves as a tableau

"Tripled" Linear Interpolation

Repeated averaging in tableau form:

$$
\begin{array}{llll}
\overbrace{\mathbf{b}_0}^{\text{Input points}} & & & \\
\mathbf{b}_1 & \mathbf{b}_0^1 & & \\
\mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\
\mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \underbrace{\mathbf{b}_0^3}_{\text{Point on curve}}
\end{array}
$$

This clearly suggests a recursive procedure …

de Casteljau (formal version)

General Bézier Curves

Given $n+1$ control points

$$\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in R^3$$

We can define a Bézier curve

$$\mathbf{b}(u) = \mathbf{b}^n(u) = \mathbf{b}_0^n(u)$$

via the recursive construction

$$\mathbf{b}_i^r(u) = (1-u)\,\mathbf{b}_i^{r-1}(u) + (u)\,\mathbf{b}_{i+1}^{r-1}(u)$$
$$\mathbf{b}_i^0(u) = \mathbf{b}_i$$

This is the de Casteljau Algorithm

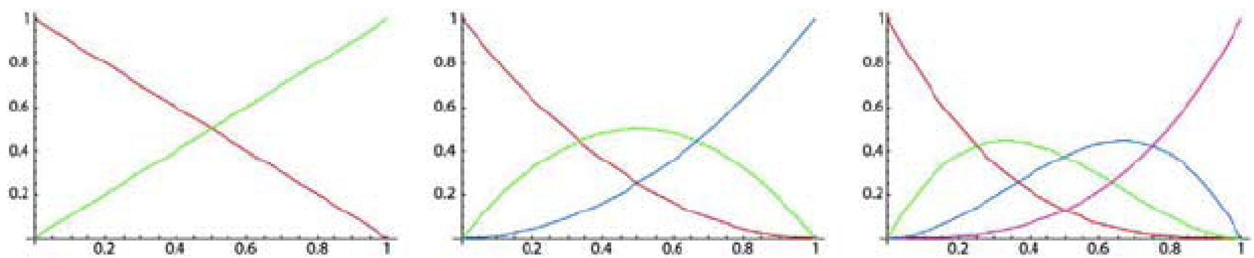
Bezier curve blending functions

Common Bernstein Polynomials

$$B_0^1 = 1 - u$$
$$B_1^1 = u$$

$$B_0^2 = (1-u)^2$$
$$B_1^2 = 2(1-u)(u)$$
$$B_2^2 = u^2$$

$$B_0^3 = (1-u)^3$$
$$B_1^3 = 3(1-u)^2(u)$$
$$B_2^3 = 3(1-u)(u)^2$$
$$B_3^3 = u^3$$



Curve has the form:

Bezier blending functions

- Bezier-Bernstein polynomials $B_i^n(u) = C(n,i)(1-u)^i u^{n-1}$
    - here C(n, i) is the number of combinations of n items, taken i at a time
    -

$$C(n,i) = \frac{n!}{(n-i)!i!}$$

Bezier curve properties

- Pass through first, last points
- Tangent to initial, final segments of control polygon
- Lie within convex hull of control polygon
- Subdivide

Equivalences

- 4 control point Bezier curve is a cubic curve
- so is an Hermite curve
- so we can transform from one to the other
- Easy way:
  - notice that 4-point Bezier curve
    - interpolates endpoints
    - has tangents 3(b_1-b_0), 3(b_3-b_2)
    - this gives Hermite->Bezier, Bezier->Hermite
- Hard way:
  - do the linear algebra

4-point Bezier curve:

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \left\{ \begin{array}{cccc} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{array} \right\} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \mathcal{B}_b \mathcal{G}_b$$

Hermite curve:

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \left\{ \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{array} \right\} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \mathcal{B}_h \mathcal{G}_h$$

Converting

- Say we know G_b $\mathcal{B}_h\mathcal{G}_h = \mathcal{B}_b\mathcal{G}_b$
  - what G_h will give the same curve?

$$\mathcal{G}_h = \mathcal{B}_h^{-1}\mathcal{B}_b\mathcal{G}_b$$

  - known G_h works similarly

Joining up curves

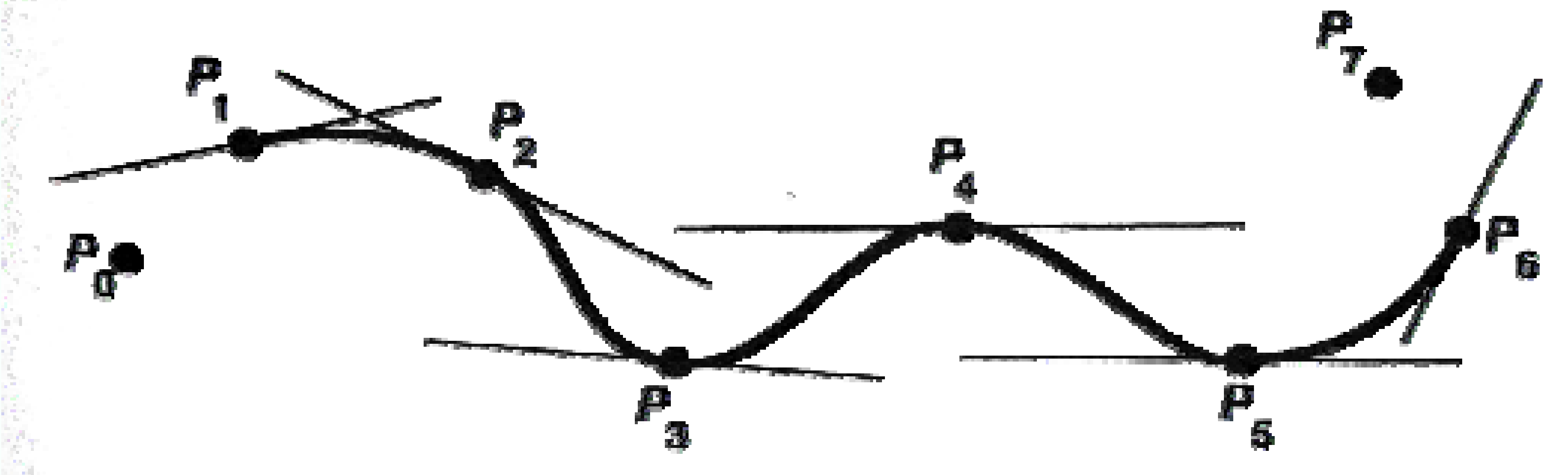
- Two kinds of join
  - Geometric continuity
    - G^0 - end points join up
    - G^1 - end points join up, tangents are parallel
    - Idea: the curves *could* be parametrized with a C^0 (C^1) parametrization, but currently are not
    - Very important in modelling
  - Parametric continuity, or continuity
    - C^0 - the parameter functions of the curve are continuous
    - C^1 - the parameter functions are continuous, have continuous deriv
    - C^2 - .. .. and continuous second deriv
    - Very important in animation (the parametrization is usually time)

Simple cases

- Join up two point Hermite curves
  - endpoints the same, vectors not - G^0
  - endpoints, vectors the same - G^1 (easy)
  - endpoints the same, vectors same direction - G^1 (harder)
  - Catmull Rom construction if we don't know tangents
- Subdivide a Bezier curve
  - result is G^infinity if we reparametrize each segment as we should
    - but not necessarily if we move the control points!
- Join up Bezier curves
  - endpoints join - G^0
  - endpoints join, end segments collinear - G^1

Catmull-Rom construction (partial)

$\mathbf{p}_0, \ldots, \mathbf{p}_n$ define tangent $\mathbf{r}_i = s(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$



Cubic interpolating splines

- n+1 points P_i
- X_i(t) is curve between P_i, P_i+1

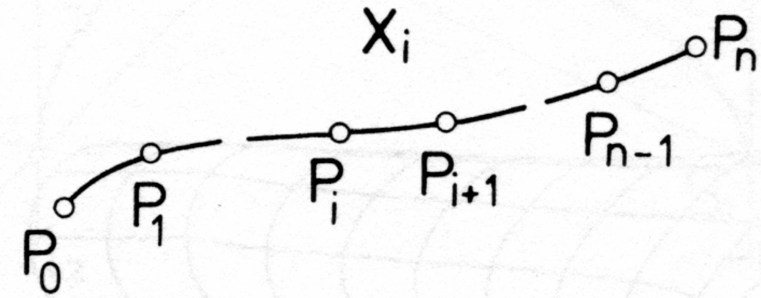


Fig. 3.11. The spline segment $\boldsymbol{X}_i$.

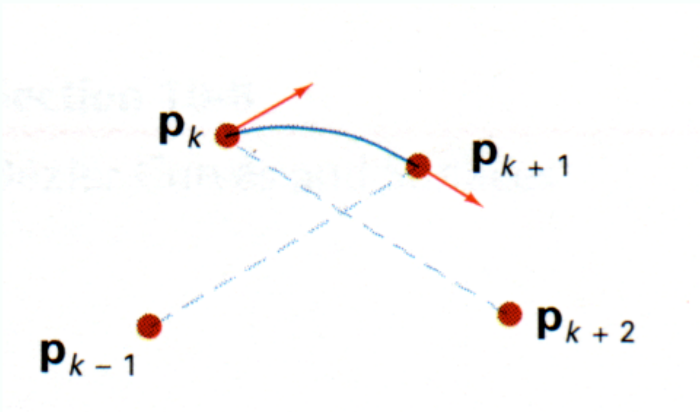
Interpolating Cubic splines, G^1

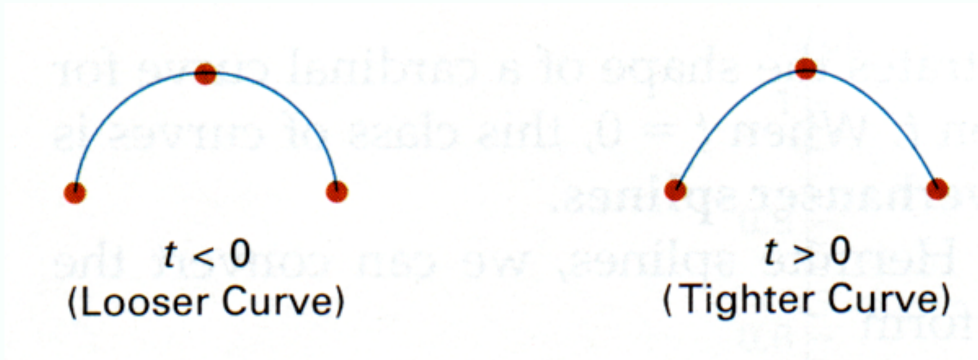
- join a series of Hermite curves with equal derivatives.
- But where are the derivative values to come from?
  - Measurements
  - Cardinal splines

$$\frac{d\mathbf{X}_i}{dt}(0) = \frac{1}{2}(1-t)(\mathbf{P}_{i+1} - \mathbf{P}_{i-1})$$

    - average points
    - t is “tension”
    - specify endpoint tangents
      - or use difference between first two, last two points

Tension


$\mathbf{p}_k$
$\mathbf{p}_{k+1}$
$\mathbf{p}_{k-1}$
$\mathbf{p}_{k+2}$


$t < 0$
(Looser Curve)
$t > 0$
(Tighter Curve)

Interpolating Cubic splines: C^2

- One parametrization for the whole curve
  - divided up into intervals, called knots

$$a = t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = b.$$

$$\Delta t_i := t_{i+1} - t_i.$$

  - In each segment, there is a cubic curve FOR THAT SEGMENT

$$\mathbf{A}_i(t-t_i)^3 + \mathbf{B}_i(t-t_i)^2 + \mathbf{C}_i(t-t_i) + \mathbf{D}_i$$

  - And we must make this lot C^2

$$t_i \leq t < t_{i+1}$$

Continuity

- at interval endpoints, curves must be
  - Continuous

$$\mathbf{X}_i(t_i) = \mathbf{X}_{i-1}(t_i)$$

  - have continuous derivative

$$\frac{d\mathbf{X}_i}{dt}(t_i) = \frac{d\mathbf{X}_{i-1}}{dt}(t_i)$$

  - have continuous second derivative

$$\frac{d^2\mathbf{X}_i}{dt^2}(t_i) = \frac{d^2\mathbf{X}_{i-1}}{dt^2}(t_i)$$

Curves

- Assume we KNOW the derivative at each point
  - write derivatives with ‘

$$\mathbf{X}_i(t_i) = \mathbf{P}_i = \mathbf{D}_i$$

$$\frac{d\mathbf{X}_i}{dt}(t_i) = \mathbf{X}_i'(t_i) = \mathbf{P}_i' = \mathbf{C}_i$$

$$\mathbf{X}_i(t_{i+1}) = \mathbf{P}_{i+1} = \mathbf{A}_i \Delta t_i^3 + \mathbf{B}_i \Delta t_i^2 + \mathbf{C}_i \Delta t_i + \mathbf{D}_i$$

$$\mathbf{X}_i'(t_{i+1}) = \mathbf{P}_{i+1}' = 3\mathbf{A}_i \Delta t_i^2 + 2\mathbf{B}_i \Delta t_i + \mathbf{C}_i$$

Curves

$$
\begin{aligned}
\mathbf{X}_i(t) \quad = \quad & \mathbf{P}_i \left(2\frac{(t-t_i)^3}{(\Delta t_i)^3} - 3\frac{(t-t_i)^2}{(\Delta t_i)^2} + 1 \right) + \\
& \mathbf{P}_{i+1} \left(-2\frac{(t-t_i)^3}{(\Delta t_i)^3} + 3\frac{(t-t_i)^2}{(\Delta t_i)^2} \right) + \\
& \mathbf{P}'_i \left(\frac{(t-t_i)^3}{(\Delta t_i)^2} - 2\frac{(t-t_i)^2}{(\Delta t_i)} + (t-t_i) \right) + \\
& \mathbf{P}'_{i+1} \left(\frac{(t-t_i)^3}{(\Delta t_i)^2} - \frac{(t-t_i)^2}{(\Delta t_i)} \right)
\end{aligned}
$$

C^2 Continuity supplies derivatives

- Second derivative is continuous

$$\mathbf{X}''_{i-1}(t_i) = \mathbf{X}_i(t_i)$$

- Differentiate curves, rearrange to get

$$\Delta t_i \mathbf{P}'_{i-1} + 2(\Delta t_{i-1} + \Delta t_i)\mathbf{P}'_i + \Delta t_{i-1}\mathbf{P}'_{i+1} = 3\frac{\Delta t_{i-1}}{\Delta t_i}(\mathbf{P}_{i+1} - \mathbf{P}_i) + 3\frac{\Delta t_i}{\Delta t_{i-1}}(\mathbf{P}_i - \mathbf{P}_{i-1})$$

- This is a linear system in tridiagonal form
  - can see as recurrence relation - we need two tangents to solve

C^2 cubic splines

- Recurrence relations
  - d(n-1) equations in d(n+1) unknowns (d is dimension)
- Options:
  - give P'_0, P'_1 (easiest, unnatural)
  - second derivatives vanish at each end (natural spline)
  - give slopes at the boundary
    - vector from first to second, second last to last
  - parabola through first three, last three points
  - third derivative is the same at first, last knot

More general splines

- We would like to retain continuity, local control
  - but drop interpolation
- Mechanism
  - get clever with blending functions
  - continuity of curve=continuity of blending functions
  - we will need to “switch” on or off pieces of function
    - e.g. linear example
- This takes us to B-splines, which you know
  - so we’ll move on to surface constructions