CS 174C Week 3

Aidan Jan

January 24, 2024

Motion Curves

- The most basic capability of an animation package is to let the user set animation variables in each frame
  - Not so easy - major HCI challenges in designing an effective user interface
  - We will not consider HCI issues
- The next is to support keyframing: Computer automatically interpolates in-between frames
- A motion curve is what you get when you plot an animation variable against time
  - The computer must come up with motion curves that interpolate your keyframe values

Different Forms of Curve Functions

- Explicit: $y = f(x)$
  - Cannot get multiple values for single x or infinite slopes
- Implicit: $f(x, y) = 0$
  - Cannot easily compare tangent vectors at joints
  - In/Out test, normals from gradient
- Parametric: $x = f_x(t), y = f_y(t), z = f_z(t)$
  - Most convenient for motion representation

Describing Curves by Means of Polynomials

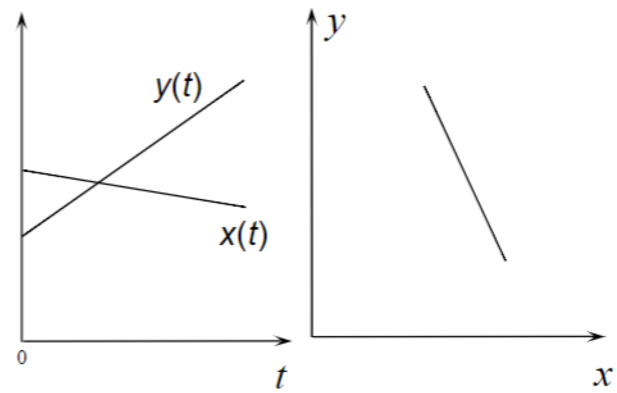
Reminder:

- L^{th} degree polynomial
- $p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_L t^L$
- $a_0, \ldots, a_L$ are the coefficients
- L is the degree
- $(L + 1)$ is the "order" of the polynomial

Polynomial Curves of Degree 1

Parametric and implicit forms are linear

$$x(t) = at + b$$
$$y(t) = ct + d$$



Polynomial Curves of Degree 2

Parametric

- $x(t) = at^2 + 2bt + c$
- $y(t) = dt^2 + 2et + f$
- For any choice of constants
  - $a, b, c, d, e, f \rightarrow$ parabola

Rational Parametric

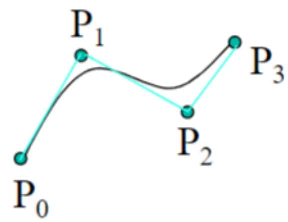
- $P(t) = \frac{P_0(1-t)^2 + 2wP_1t(1-t) + P_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}$
- $w < 1$: ellipse
- $w = 1$: parabola
- $w > 1$: hyperbola

Curves From Geometric Constraints

Geometric Approach

- Constraints $\rightarrow$ Polynomial $\rightarrow$ Curve
- $P_0, \ldots, P_L \rightarrow$ (Curve Generation) $\rightarrow P(t)$
  - P_i: control points
  - $P_0, \ldots, P_L$: control polygon

Interpolation vs. Approximation



Bezier Curves and the De Casteljau Algorithm

Tweening

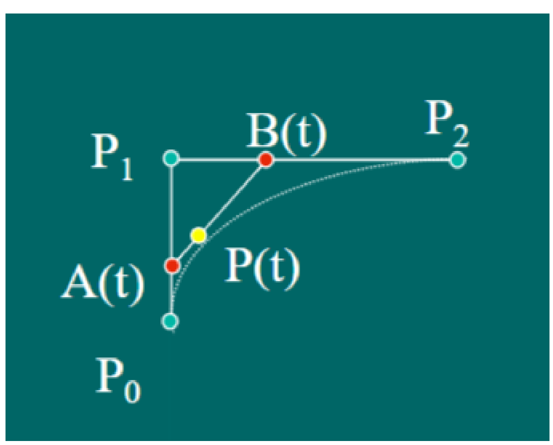
When there are two points:

- $A(t) = (1-t)P_0 + tP_1$
- $P(t) = A(t)$
- Essentially, A is a point on the line between P_0 and P_1

When there are three points:

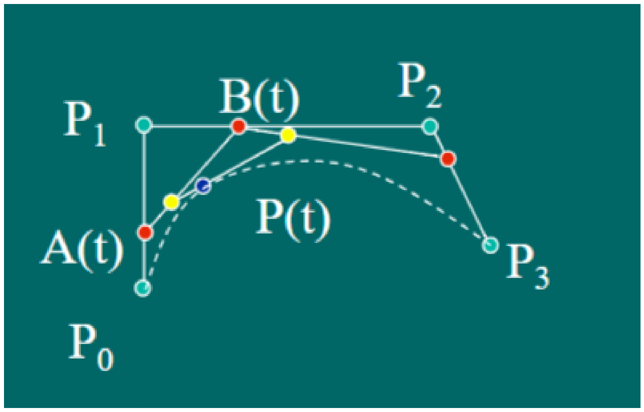
- $A(t) = (1-t)P_0 + tP_1$
- $B(t) = (1-t)P_1 + tP_2$
- $A(t)$ is a point between P_0 and P_1 and $B(t)$ is a point between P_1 and P_2.
- Now, place another point, $P(t)$ on the line between $A(t)$ and $B(t)$.
  - $P(t) = (1-t)A + tB = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$

When we move the value of t from 0 to 1, $P(t)$ would move from P_0 to P_2 along a curved path, defined by the quadratic equation.



If we repeat the same process for $P(t)$ but instead of four points, then

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$



Cubic Berstein Polynomials

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

$$\begin{aligned}
B_0^3(t) &= (1-t)^3 \\
B_1^3(t) &= 3(1-t)^2 t \\
B_2^3(t) &= 3(1-t)t^2 \\
B_3^3(t) &= t^3
\end{aligned}$$

Expansion of $[(1-t)+t]^3 = (1-t)^3 + 3(1-t)^2 t + 3(1-t)t^2 + t^3 \rightarrow \sum_k B_k^3(t) = 1, k = 0, 1, 2, 3$
An affine combination of points

Berstein Polynomials of Degree L

Degree L implies $L+1$ control points.

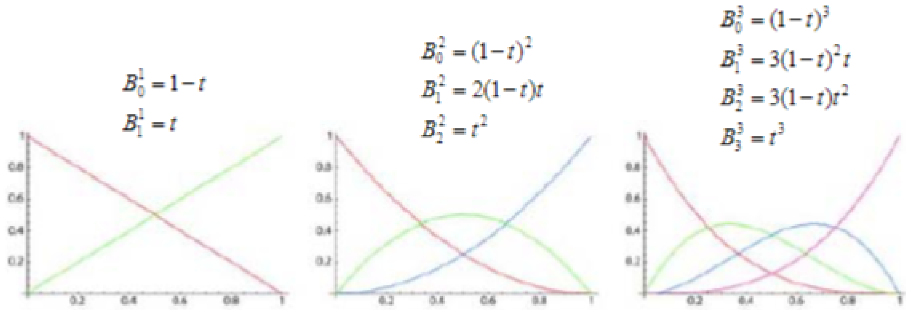
$$P(t) = \sum_{k=0}^{L} B_k^L(t) P_k$$

where:

- $B_k^L(t) = \binom{L}{k}(1-t)^{L-k} t^k$
- $\binom{L}{k} = \frac{L!}{k!(L-k)!}$, for $L > k$
- $\sum_{k=0}^{L} B_k^L(t) = 1$, for all t

Expansion of $[(1-t)+t]^L$

Common Berstein Polynomials



- Always positive
- Zero only at $t = 0$ or $t = 1$
- Berstein polynomials can also be defined recursively

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(u)$$
$$B_0^0(t) = 1$$

Properties of Bezier Curves

- End point Interpolation
- Affine Invariance: $T(P(t)) = \sum_{k=0}^{L} B_k^L(t)T(P)_k$
- Invariance under affine transformation of the parameter
- Convex Hull property for t in [0, 1]: $P = \sum_{k=0}^{L} a_k P_k$ where $\sum_{k=0}^{L} a_k = 1$ and $a_k > 0$
- Linear precision by collapsing convex hull
- Variation Diminishing property: No straight line cuts the curve more times than it cuts the control polygon

Derivatives of Bezier Curves

It can be shown that:

- Velocity is also a Bezier curve of lower degree

$$P'(t) = L\sum_{k=0}^{L-1} B_k^{L-1}(t)\Delta P_k, \text{ where } \Delta P_k = P_{k+1} - P_k$$

- Acceleration

$$P''(t) = L(L-1)\sum_{k=0}^{L-2} B_k^{L-2}(t)\Delta^2 P_k, \text{ where } \Delta^2 P_k = \Delta P_{k+1} - \Delta P_k$$

Cubic Parametric Curves

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$$
$$z(t) = c_3t^3 + c + 2t^2 + c_1t + c_0$$
$$t \in [0, 1]$$

As a matrix,

$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$\begin{aligned} x(t) &= TA \\ y(t) &= TB \\ z(t) &= TC \end{aligned}$$

Derivative of Cubic Parameter Curves

Simply take the derivative of the matrix in terms of t.

$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

How does the magnitude of the tangent affect the curve?

- Same lower tangent direction but different magnitude.
- The magnitude defines how fast the curve assumes the tangent direction
- (remember: tangent $\rightarrow$ velocity in parametric space)

Parametric Cubic Curves From Constraints

Example: Endpoints and a tangent at midpoint
Constraints:

- $x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} A$
- $x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} A$
- $x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A$
- $x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5 & 1 \end{bmatrix} A$
- $x'(0.5) = \begin{bmatrix} 3(0.5^2) & 2(0.5) & 1 & 0 \end{bmatrix} A$
- $x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A$
- $G_x = BA$

This can be written as:

$$
\begin{aligned}
x(t) &= \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} A = TA \\
x'(t) &= \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} A = T'A \\
\begin{bmatrix} x_0 \\ x_{0.5} \\ x'_{0.5} \\ x_1 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.5^3 & 0.5^2 & 0.5 & 1 \\ 3(0.5)^2 & 2(0.5) & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} A \\
G_x &= BA \Rightarrow A = B^{-1}G_x
\end{aligned}
$$

$$x(t) = TA \Rightarrow \boxed{x(t) = TB^{-1}G_x}$$

Final form:

- Basis matrix

$$
\begin{aligned}
&x(t) = TB^{-1}G_x \\
&\text{Set } M = B^{-1} \\
&\quad x(t) = TMG_x \\
&\quad y(t) = TMG_y \\
&\quad z(t) = TMG_z \\
&P(t) = TMG
\end{aligned}
$$

- For the example

$$M = \begin{bmatrix} -4 & 0 & -4 & 4 \\ 8 & -4 & 6 & -4 \\ -5 & 5 & -2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Blending Functions

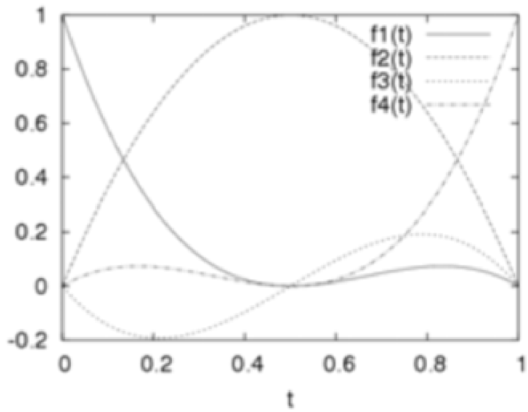
Given by TM:

- $x(t) = TMG_x \Rightarrow x(t) = \begin{bmatrix} f_1(t) & f_2(t) & f_3(t) & f_4(t) \end{bmatrix} G_x$

For the example:

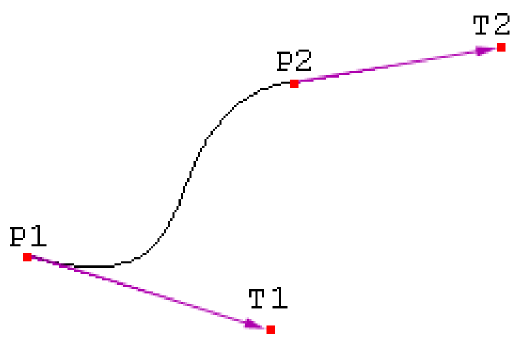
- $f_1(t) = -4t^3 + 8t^2 - 5t + 1$
- $f_2(t) = -4t^2 + 4t$
- $f_3(t) = -4t^3 + 6t^2 - 2t$
- $f_4(t) = 4t^3 - 4t^2 + t$

Each blending function weights the contribution of one of the constraints.



Hermite Curves

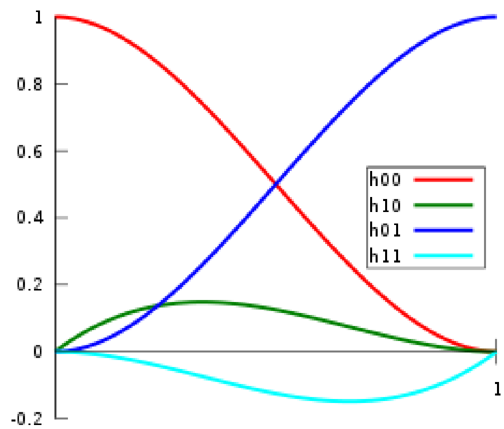
Constraints: Two endpoints and two tangents



$$G_h = \begin{bmatrix} P_1 & P_4 & R_1 & R_4 \end{bmatrix}$$
$$x(t) = TA_h = TM_hG_h$$

$$x(0) = P_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A_h$$
$$x(1) = P_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A_h$$
$$x'(0) = R_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} A_h$$
$$x'(1) = R_4 = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} A_h$$
$$G_h = B_hA_h$$
$$A_h = B_h^{-1}G_h$$
$$x(t) = TA_h$$

Blending Functions



$$M_h = B_h^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$x(t) = TM_hG_h \Rightarrow x(t) = \begin{bmatrix} f_1(t) & f_2(t) & f_3(t) & f_4(t) \end{bmatrix} G_h$$

$$f_1(t) = 2t^3 - 3t^2 + 1$$
$$f_2(t) = -2t^3 + 3t^2$$
$$f_3(t) = t^3 - 2t^2 + t$$
$$f_4(t) = t^3 - t^2$$

Bezier Curves

Bezier curves are a special case of Hermite curves.

$$P_{1,h} = P_1$$
$$P_{4,h} = P_4$$
$$R_{1,h} = 3(P_2 - P_1)$$
$$R_{4,h} = 3(P_4 - P_3)$$

As a matrix:

$$\begin{bmatrix} P_{1,h} \\ P_{4,h} \\ R_{1,h} \\ R_{4,h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$G_h = M_{bh}G_b$$

$$P(t) = TM_hG_h \Rightarrow P(t) = TM_hM_{bh}G_b \Rightarrow P(t) = TM_bG_b$$

We can verify that TM_b are the Berstein Polynomials.
Recall that:

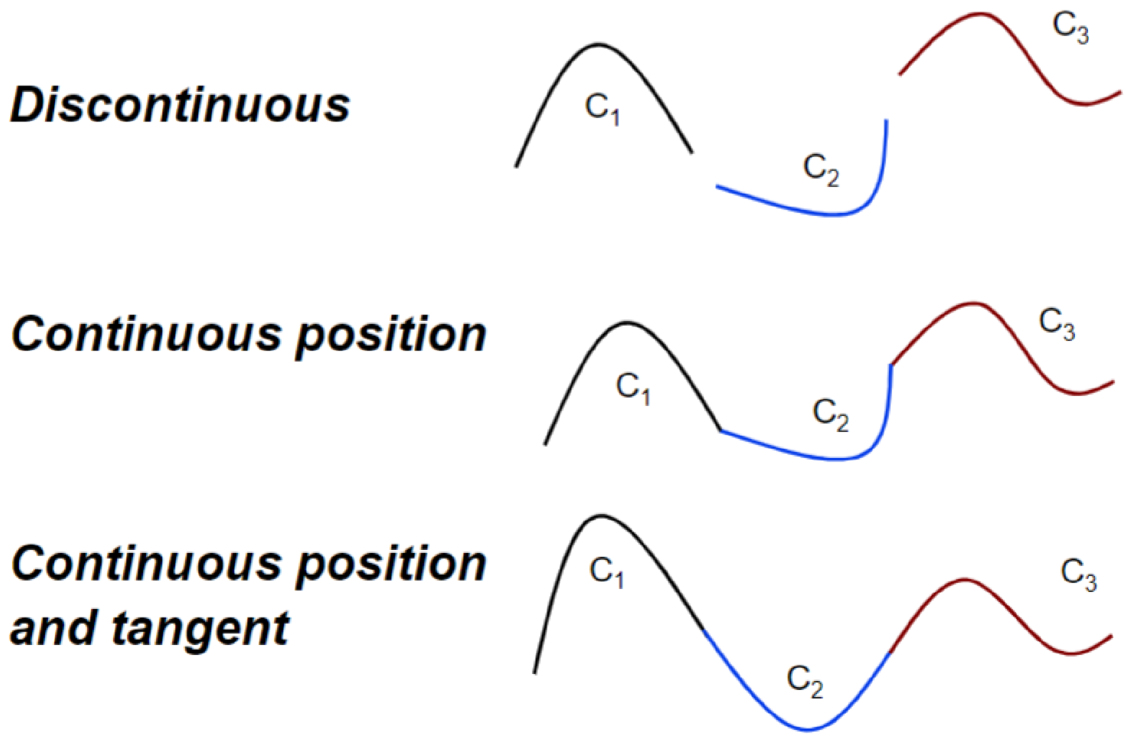
- $f_1(t) = (1-t)^3$
- $f_2(t) = 3t(1-t)^2$

- $f_3(t) = 3t^2(1-t)$
- $f_4(t) = t^3$

Splines

- Splines are the standard way to generate a smooth curve which interpolates given values
- A spline function is just a piecewise polynomial function
  - Split up the real line into intervals
  - Over each interval, pick a different polynomial
- If the polynomials are small degree (Typically at most cubics) the curve is easy and fast to compute

Connecting piecewise curves to splines:



Continuity

- Parametric C^k Continuity
  - Derivatives $P^{(i)}$ for $i = 0, \ldots, k$ exist and are continuous for t in $[a, b]$
  - Terminology:
    * P is k-smooth
    * P has k^{th}-order continuity
- Geometric G^k Continuity
  - $P^{(i)}(t-) = c_i P^{(i)}(t+)$

  * for i = 0, . . . , k and
  * for some constants c_i
  * for t in $[a, b]$

Knots and Control Points

- The ends of the intervals, where one polynomial ends and another one starts, are called "knots"
- A control point is a knot together with associated shape control variables
- The spline either interpolates (goes through) or approximates (goes near) the control points

Spline Curve

Different definitions exist.
Ours:

- A spline curve is an affine blend of points weighted by piecewise polynomial functions. It must be continuous at the knots, but may have discontinuous derivatives.

Hermite Splines

- Hermite splines have even richer control points:
- In addition to a function value, a slope (derivative) is specified.
  - So the Hermite spline interpolates the control values and must match the control slopes at the knots
- Particularly useful for animation - more control over slow in/out, etc.

Smoothness

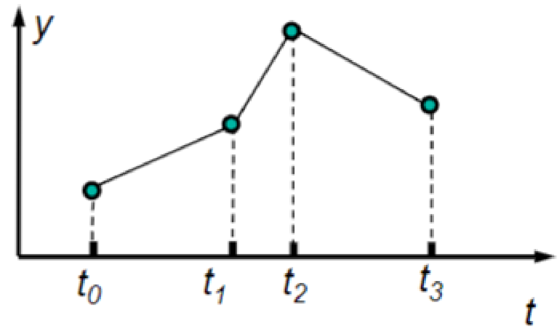
- Each polynomial in a spline is infinitely differentiable (very smooth)
- But at the junction between two polynomials, the spline isn't necessarily even continuous!
- We need to enforce constraints on the polynomials to get the degree of smoothness we want
  - Values match: C^0 continuous
  - Slopes (first derivatives) match: C^1 continuous
  - Second derivatives match: C^2 continuous
  - Etc.

Example: Piecewise Linear Spline

- Restrict all polynomials to be linear
  - $f(t) = a_i(t - t_i) + b_i$ in $[t_i, t_{i+1}]$
- Enforce continuity: make each line segment interpolate the control point on either specified
  - $a_i(t - t_i) + b_i = y_i$ and $a_i(t_{i+1} - t_i) + b_i = y_{i+1}$
- Solve to get
  - $a_i = \frac{y_{i+1} - y_i}{t_{i+1} - t_i}$; $b_i = y_i$
- End result:

  - straight line segments connecting the control points

- C^0 but not C^1



To do better than this, we need higher degree polynomials.
For motion curves, cubic splines are almost always used.
We now have three main choices:

- Hermite splines: Interpolating, up to C^1
- Catmull-Rom: Interpolating, C^1
- B-Splines: Approximating, C^2

Cubic Hermite Splines

- Our generic cubic in an interval $[t_i, t_{i+1}]$ is:
  - $q_i(t) = a_i(t - t_i)^3 + b_i(t - t_i)62 + c_i(t - t_i) + d_i$ with $t - t_i$ in $[0, 1]$
- Make it interpolate endpoints:
  - $q_i(t_i) = y_i$ and $q_i(t_{i+1}) = y_{i+1}$
- And make it match given slopes:
  - $q_i'(t_i) = s_i$ and $q_i'(t_{i+1}) = s_{i+1}$
- Work it out to get

$$a_i = \frac{-2(y_{i+1} - y_i)}{(t_{i+1} - t_i)^3} + \frac{s_i + s_{i+1}}{(t_{i+1} - t_i)^2} \qquad c_i = s_i$$
$$b_i = \frac{3(y_{i+1} - y_i)}{(t_{i+1} - t_i)^2} - \frac{2s_i + s_{i+1}}{t_{i+1} - t_i} \qquad d_i = y_i$$

Hermite Basis

Rearrange the solution to get:

$$\begin{aligned} y(t) = & y_i \left(\frac{2(t - t_i)^3}{(t_{i+1} - t_i)^3} - \frac{3(t - t_i)^2}{(t_{i+1} - t_i)^2} + 1 \right) + y_{i+1} \left(\frac{-2(t - t_i)^3}{(t_{i+1} - t_i)^3} - \frac{3(t - t_i)^2}{(t_{i+1} - t_i)^2} \right) \\ & + s_i \left(\frac{(t - t_i)^3}{(t_{i+1} - t_i)^2} - \frac{2(t - t_i)^2}{t_{i+1} - t_i} + (t - t_i) \right) + s_{i+1} \left(\frac{(t - t_i)^3}{(t_{i+1} - t_i)^2} - \frac{(t - t_i)^2}{t_{i+1} - t_i} \right) \end{aligned}$$

for $i \in [t_0, t_n]$
That is, we're taking a linear combination of four basis functions

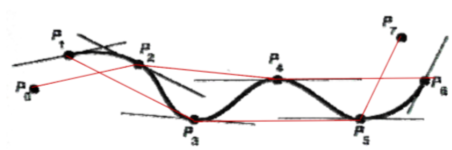
- Note the functions and their slopes are either 0 or 1 at the start and end of the interval

Breaking Hermite Splines

- Usually just specify one slope at each knot
- But a useful capability: use a different slope on each side of a knot
  - We break C^1 smoothness, but gain control
  - Can create motions that abruptly change, like collisions

Catmull-Rom Splines

- Given a set of points in space, suppose we want a spline that:
  - Interpolates the points (rules out Bezier)
  - With C^1 continuity (Hermite: Lots of tweaking)
- This is a common situation in animation
- We start with the given set of points $P_0, \ldots, P_n$
- Define tangents $r_i = s(P_{i+1} - P_{i-1})$



- This is really just a C^1 Hermite spline with an automati choice of slopes
  - Use a 2nd order finite difference formula to estimate slope from values

$$s_i = \left(\frac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}}\right) \div \frac{y_{i+1} - y_i}{t_{i+1} - t_i} - \left(\frac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}}\right) \div \frac{y_i - y_{i-1}}{t_i - t_{i-1}}$$

  - For equally spaced knots, it simplifies to:

$$s_i = \frac{y_{i+1} - y_{i-1}}{t_{i+1} - t_{i-1}}$$

Catmull-Rom Boundaries

- Need to use slightly different formulas for the Boundaries
- For example, 2nd order accurate finite difference at the start of the interval:

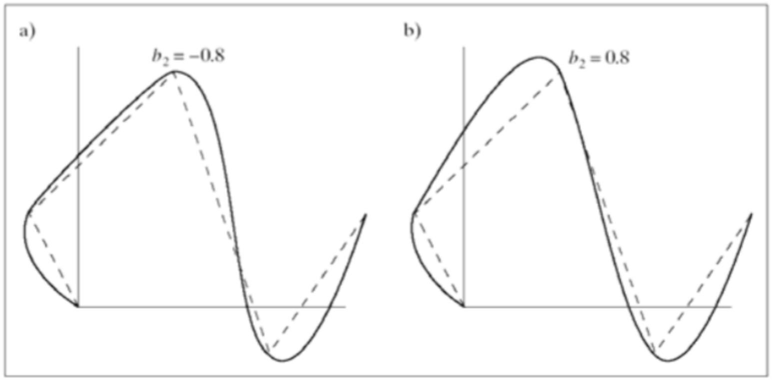
$$s_0 = \left(\frac{t_2 - t_0}{t_2 - t_1}\right) \div \frac{y_1 - y_0}{t_1 - t_0} - \left(\frac{t_1 - t_0}{t_2 - t_1}\right) \div \frac{y_2 - y_0}{t_2 - t_0}$$

  - Symmetric formula for end of interval
- Which simplifies for equally spaced knots:

$$s_0 = 2\frac{y_1 - y_0}{\Delta t} - \frac{y_2 - y_0}{2\Delta t}$$

Adding Tension Control

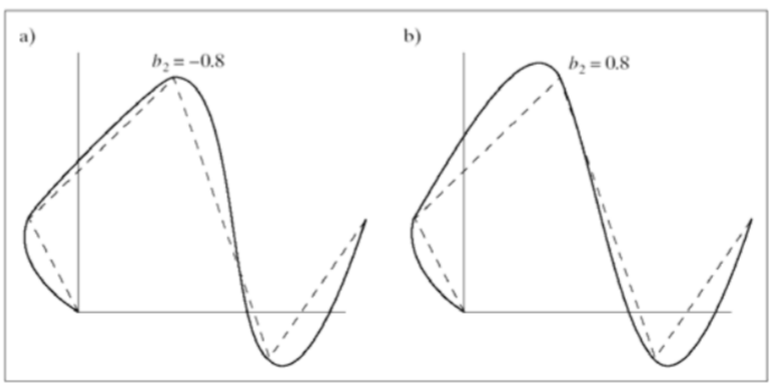
$$P'(t_k) = \frac{1}{2}(1 - v_k)(P_{k+1} - P_{k-1})$$



Adding Bias Control

Kochanek-Bartels Splines

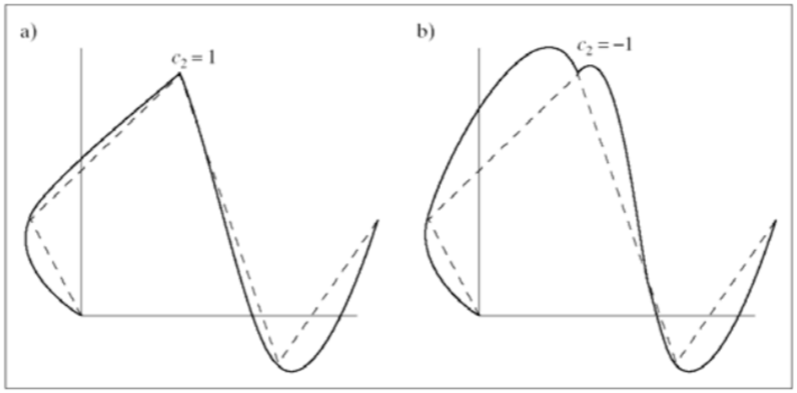
$$P'(t_k) = \frac{1}{2}(P_k - P_{k-1}) + \frac{1}{2}(P_{k+1} - P_k)$$
$$P'(t_k) = \frac{1}{2}(1 - b_k)(P_k - P_{k-1}) + \frac{1}{2}(1 + b_k)(P_{k+1} - P_k)$$



Adding Continuity Control

Kochanek-Bartels Splines

$$R'_{k-1}(1) = \frac{1}{2}(1 - c_k)(P_k - P_{k-1}) + \frac{1}{2}(1 + c_k)(P_{k+1} - P_k)$$
$$R'_k(0) = \frac{1}{2}(1 + c_k)(P_k - P_{k-1}) + \frac{1}{2}(1 - c_k)(P_{k+1} - P_k)$$



Splines Summary

- General form of the splines we have segment

$$P(t) = \sum_{k=0}^{L} P_k f_k(t),\ t \in \Re$$

- Often f is a translated version of a single function

$$P(t) = \sum_{k=0}^{L} P_k f(t - t_k)$$
$$\text{knots} : [t_0, \dots, t_L]$$

- For Hermite splines, the knots coincided with the control points

B-Splines

- Like Catmull-Rom splines, start with sequence of control points $P_0, \dots, P_n$
- Drop the interpolating condition and instead design a spline curve that is C^{m-2} smooth, where m is the order
  - Curve segments meet at knots
  - For Hermite and others, the knots were always at control points
- Now basis functions overlap more than one know interval

General Form of B-Splines

- Let's make the order m explicit

$$P(t) = \sum_{k=0}^{L} P_k N_{k,m}(t)$$

- Fundamental formula
  - generalization of successive linear Interpolation

$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} - t_k}\right) N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}}\right) N_{k+1,m-1}(t)$$
$$t \in [t_k, t_{k+m}]$$

- For equispaced knots $0, \dots, L$

$$P(t) = \sum_{k=0}^{L} P_k N_{k,m}(t) = \sum_{k=0}^{L} P_k N_{k,m}(t-k)$$

Constant B-Splines (Order m = 1)

- With knots $t_0 = 0$, $t_1 = 1$
- Constant (a.k.a. Haar) function,

$$N_1(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases}$$
$$N_{k,1}(t) = N_{0,1}(t-k)$$

- For non-equispaced knots

$$N_{k,1}(t) = \begin{cases} 1 & \text{if } t_k \le t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

- Each function has support over $[t_k, t_{k+1})$

Linear B-Splines (Order m = 2)

- With knots $t_0 = 0, t_1 = 1, \dots$

$$\begin{aligned} N_{0,2}(t) &= \frac{t}{1} N_{0,1} + \frac{2-t}{1} N_{1,1} \\ &= \begin{cases} t \times 1 + (2-t) \times 0 & \text{if } t \in [0,1] \\ (2-t) \times 0 + (2-t) \times 1 & \text{if } t \in [1,2] \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} t & \text{if } t \in [0,1) \\ 2-t & \text{if } t \in [1,2] \\ 0 & \text{otherwise} \end{cases} \\ N_{k,2}(t) &= N_{0,2}(t-t_k) \end{aligned}$$