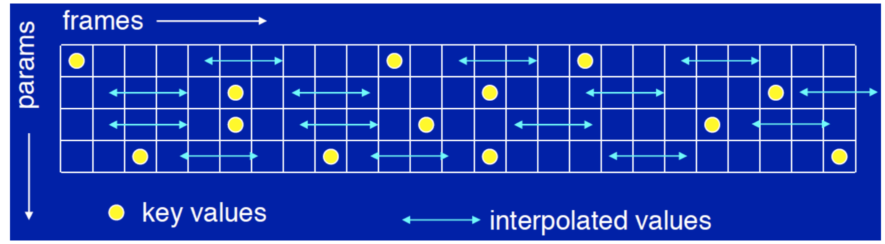
EECS 487: Interactive Computer Graphics

Lecture 33:

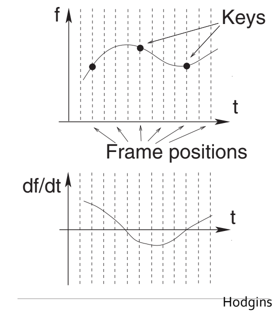
- Keyframe interpolation and splines
- Cubic splines

Interpolating Key Values



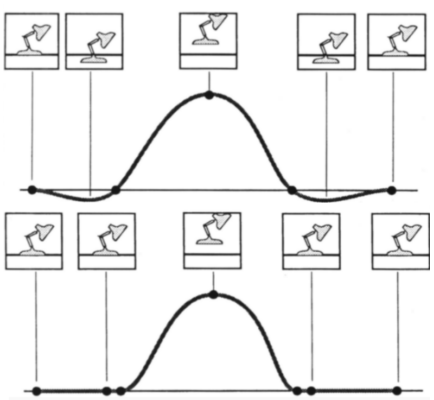
The key values of each variable may occur at different frames

The interpolation of key values of a variable defines a curve



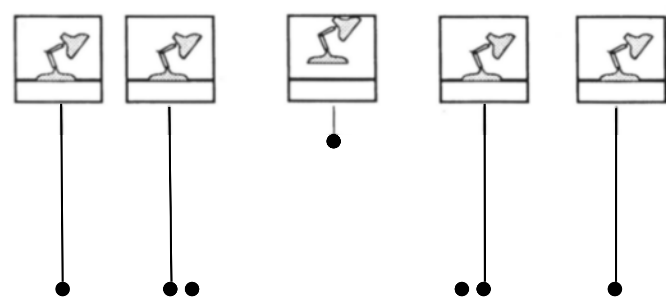
Potential Problem with Interpolation

The curve may undershoot and cause inter-penetration
Solution: add key frames (= control points)!



Motion Control Curve

Given the key frames, how would you mathematically represent a control curve that interpolates (passes through) the control points?

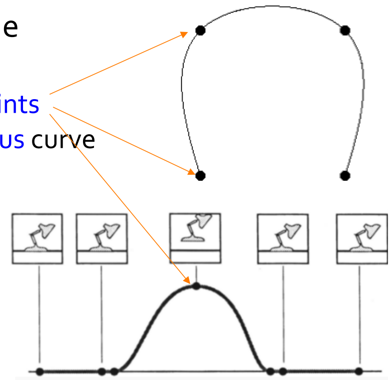


Motion Control Curve

We don't want motion with
- unnatural (painful) twists and bends
- jerkiness

Which representation of curves has these characteristics?

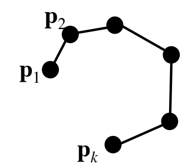
Desired characteristics of the motion control curve:
- user controlled with control points
- defines a smooth and continuous curve
  - ability to evaluate derivatives
- stable: doesn't cross over itself
- local control of curve shape
  - change to one part of the curve doesn't effect the entire curve



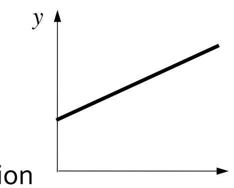
Hodgins

Representation of Curves

Polyline: piecewise linear curves
- given a sequence of vertices (control points)
- connect each pair of consecutive vertices with a line segment
- – a smooth curve will need a continuous set of points on the plane (or in space)
  - – hard to get precise, smooth results
  - – too much data, too hard to work with



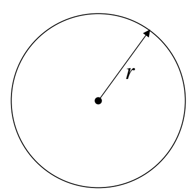
Explicit: $y = f(x)$
- \+ easy to generate points
- – single-valued for each x
- – must be a function: big limitation, e.g., vertical lines?
- – rotations completely change representation



Yu

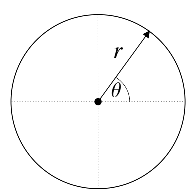
Representation of Curves

Implicit: $f(x, y) = 0$
- \+ supports multiple values for each x
- \+ easy to test if on, or to either side, of curve
- – hard to generate points



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

Parametric: $(x, y) = (f(u), g(u))$
- parameterization of a curve == how a change in u moves you along a given curve in xyz space
- \+ supports multiple values for each x
- \+ fairly easy to generate points
- \+ can describe trajectories, the speed at which we move on the curve



$(x, y) = (r \cos\theta, r \sin\theta)$

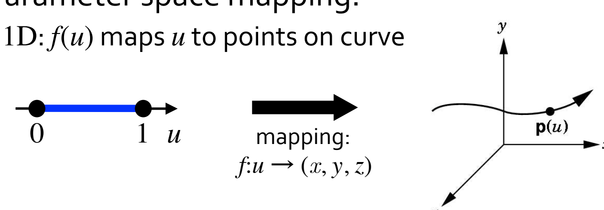
We want some kind of parametric curve to control motion!

Schulze,Yu

Parametric Curves

Define a mapping from parameter space (e.g., u, usually $\in [0,1]$), to points in 2D, 3D, etc.

Parameter space mapping:
- 1D: $f(u)$ maps u to points on curve



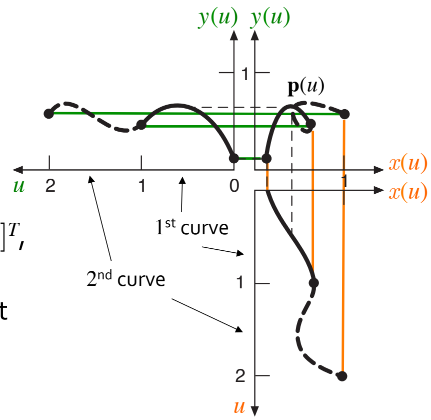
- 2D: $[f(u), g(u)]$ maps u to points on surface

Chenney

Parametric Curves

In general, described by a vector-valued function, i.e., n scalar functions, of 1D parameter space

$$\mathbf{p}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$



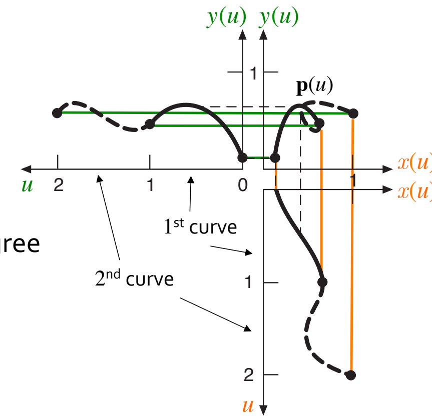
Tangent of the curve,
$\mathbf{t}(u) = \mathbf{p}'(u) = [x'(u)\ \ y'(u)\ \ z'(u)]^T$,
is the velocity of movement and
$||\mathbf{t}(u)||$ is the speed of movement

$\mathbf{T}(u) = \mathbf{t}(u)/||\mathbf{t}(u)||$
is the unit tangent of the curve

Foley, Van Dam, Schulze

Parameter Mapping

$$\mathbf{p}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$



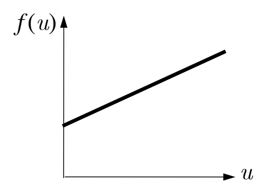
The coordinate functions can be any function:
- polynomials of arbitrary degree
- trigonometric functions
- exponentiations
- logarithms
- Fourier series, etc.

We'll only consider polynomials;
more complex functions are harder to deal with

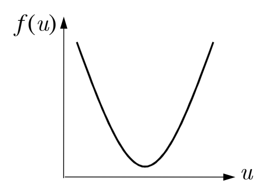
FvD,Schulze,Merrell

Polynomial Functions

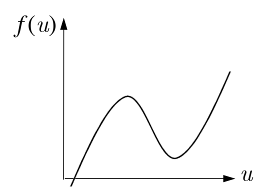
Linear (1st order):
$f(u) = a_0 + a_1 u$



Quadratic (2nd order):
$f(u) = a_0 + a_1 u + a_2 u^2$



Cubic (3rd order):
$f(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$

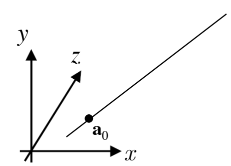


Schulze

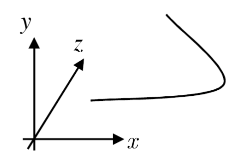
Parametric Polynomial Curves

Linear: $\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u,\ 0 \le u \le 1$
Evaluated as:

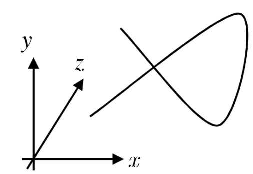
$$\mathbf{f}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \begin{bmatrix} a_{0_x} + a_{1_x} u \\ a_{0_y} + a_{1_y} u \\ a_{0_z} + a_{1_z} u \end{bmatrix}$$



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



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



Schulze

Canonical Form

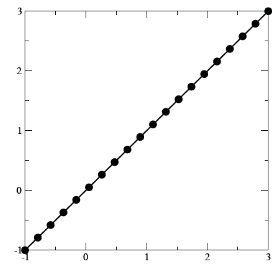
Splines have the canonical form:

$$\mathbf{f}(u) = \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 + \ldots + u^{n-1}\mathbf{a}_{n-1},$$
$$= \sum_{i=0}^{n-1} u^i \mathbf{a}_i,$$
$$= \mathbf{ua},$$
$$\mathbf{u} = \begin{bmatrix} 1 & u & u^2 & \ldots & u^k \end{bmatrix},$$
$$u \in [0,1]$$

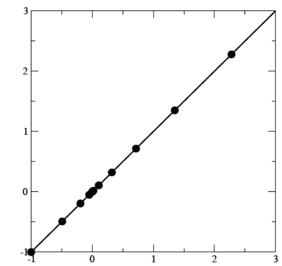
Schulze

Non-unique

Even restricted to polynomial functions, the same parametric curve may have multiple descriptions



$\mathbf{p}(u) = [u\ u]^T$



$\mathbf{p}(u) = [u^3\ u^3]^T$

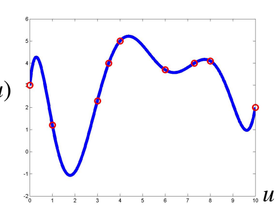
O'Brien

Lagrange Interpolation

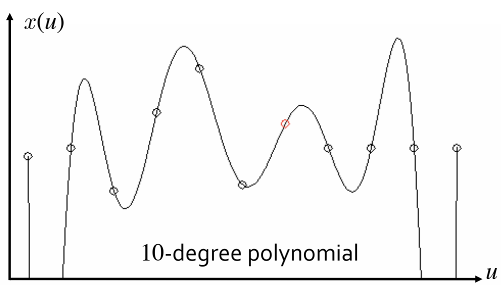
Problem: given $n+1$ control points, how do we define a parametric curve that interpolates all points?

An n-degree polynomial (Lagrange polynomial) can interpolate any $n+1$ points

Problem: small-degree Lagrange polynomials are fine but high degree ones are too wiggly



8-degree polynomial



10-degree polynomial

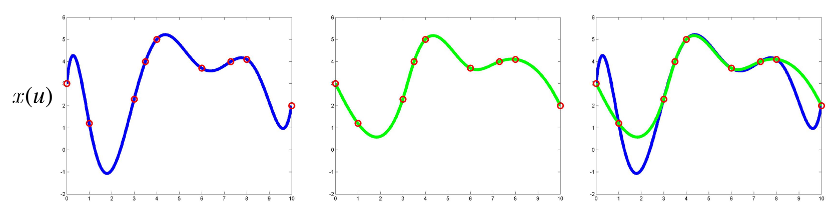
Durand,Hodgins

Introducing . . . Splines

A spline is a piecewise parametric polynomial function

Many low degree (mostly cubic) splines can be pieced together to interpolate a given set of control points, with guaranteed continuity

Piecewise definition gives local control of curve



8-degree polynomial | joined splines | splines vs. polynomial

Demo: http://www.math.ucla.edu/~baker/java/hoefer/TwoDemos.htm

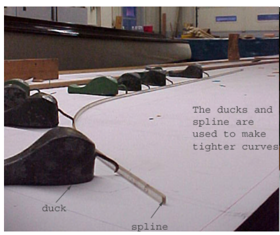
Durand

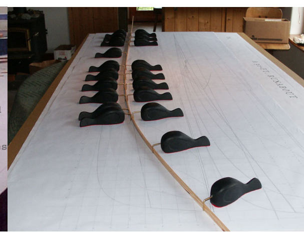
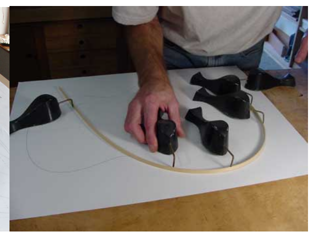
Splines

Originally used by draftsmen (draughtsmen in English, loftsmen in shipbuilding) to draw life-size curves

Physically, a stiff piece of metal that can be bent into desired shape for tracing, and held in place with "ducks"

- (the mathematical equivalent of these metal strips is the natural-cubic spline)

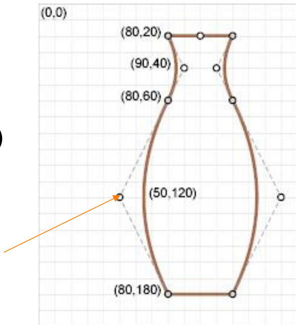


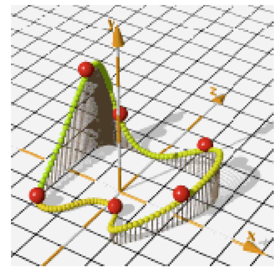
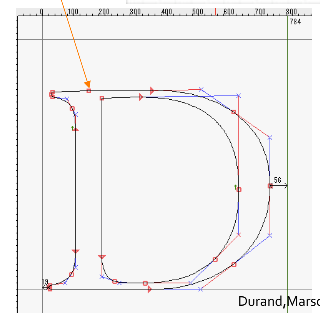
Splines

Many uses in CG:

- 2D illustration (e.g., Adobe Illustrators)
- font definition
- 3D modeling
- color ramps
- animation: trajectories
  - in general, interpolate keyframes



control points

Durand,Marschner,Hanrahan

Advantages of Splines

Specified by a few control points

- efficient for UI
- efficient for storage

Gives a smooth parametric polynomial curve $\mathbf{p}(u)$

- defined in Cartesian coordinates by $x(u)$ and $y(u)$
- convenient for animation where u is time
- convenient for tessellation in modeling as u can be discretized and the curve approximated with small linear segments

Durand

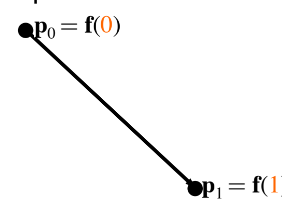
Linear Splines

The two coefficients of a first-order, linear polynomial can be determined from its two end-points:

$$\mathbf{f}(u) = \mathbf{a}_0 + u\mathbf{a}_1$$

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 + 0\mathbf{a}_1 = \mathbf{a}_0$$

$$\mathbf{p}_1 = \mathbf{f}(1) = \mathbf{a}_0 + 1\mathbf{a}_1 = \mathbf{a}_0 + \mathbf{a}_1$$



in matrix form:

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix}, \ \mathbf{p} = \mathbf{C}\mathbf{a},$$

Remember that the $\mathbf{p}_i$'s and $\mathbf{a}_i$'s are themselves vectors

where $\mathbf{C}$ is called the constraint matrix

[Schulze, wikipedia]

Basis Matrix

Then $\mathbf{a} = \mathbf{C}^{-1}\mathbf{p}$, $\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$,

$$\mathbf{a}_0 = \mathbf{p}_0$$
$$\mathbf{a}_1 = \mathbf{p}_1 - \mathbf{p}_0$$

Let's call $\mathbf{B} = \mathbf{C}^{-1}$ the blending/basis matrix (for reasons to be explained later), then:

$$\mathbf{f}(u) = \mathbf{ua} = \mathbf{uBp}$$

Schulze

Two Views of Splines

$$\mathbf{f}(u) = \begin{bmatrix} 1 & u \end{bmatrix} \mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}, \text{ where } \mathbf{B} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

$$\mathbf{f}(u) = \begin{bmatrix} 1 & u \end{bmatrix} \left(\mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} \right)$$

$$\mathbf{f}(u) = \begin{bmatrix} 1 & u \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix}$$

$$\mathbf{f}(u) = \mathbf{a}_0 + u\mathbf{a}_1$$

Coefficients ($\mathbf{a}_i$'s) can be computed from control points $\mathbf{p}_i$'s

$$\mathbf{f}(u) = \left(\begin{bmatrix} 1 & u \end{bmatrix} \mathbf{B} \right) \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

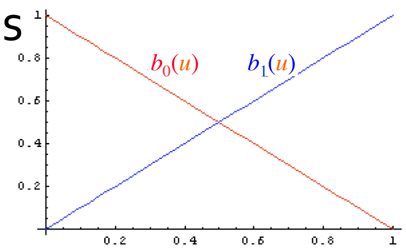
$$\mathbf{f}(u) = \begin{bmatrix} (1-u) & u \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

$$\mathbf{f}(u) = (1-u)\mathbf{p}_0 + u\mathbf{p}_1$$

Each point on the curve is a linear blending of the control points $\mathbf{p}_i$'s

Zhang

Blending Functions



$$\mathbf{f}(u) = \underbrace{(1-u)}_{b_0(u)}\mathbf{p}_0 + \underbrace{u}_{b_1(u)}\mathbf{p}_1$$

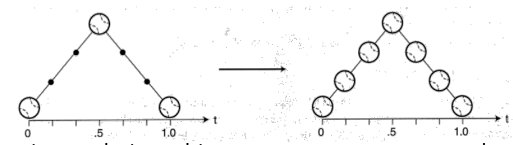
The weights $b_0(u) = (1-u)$ and $b_1(u) = u$ are called the blending functions

$$\mathbf{f}(u) = \mathbf{ua} = \mathbf{uBp} = (1-u)\mathbf{p}_0 + u\mathbf{p}_1 = \sum_{i=0}^{1} b_i(u)\mathbf{p}_i$$

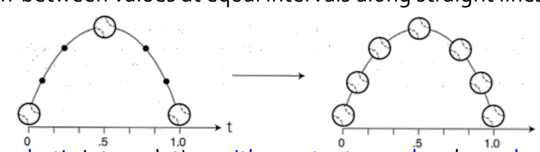
expresses the polynomial as a weighted sum (combination) of the control points: contribution of each point as u changes (and we don't have to solve for the $\mathbf{a}$'s (coefficient vectors)!)

Schulze

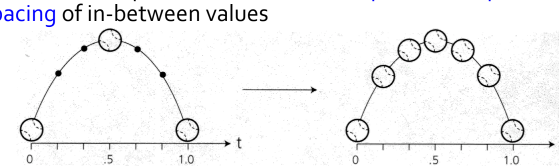
Non-linear Interpolations



Linear interpolation: object moves at constant speed
- in-between values at equal intervals along straight lines



Quadratic interpolation with constant speed and equal spacing of in-between values

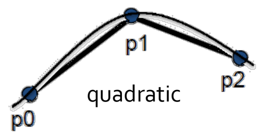


Cubic interpolation with acceleration

Hodgins

Quadratic Splines

Quadratic (2nd degree) spline $\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2$ are specified by three coefficients and can be solved by, for example,



- three points:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{f}(0) &&= \mathbf{a}_0 + 0^1\,\mathbf{a}_1 + 0^2\,\mathbf{a}_2 \\
\mathbf{p}_1 &= \mathbf{f}(0.5) &&= \mathbf{a}_0 + 0.5^1\,\mathbf{a}_1 + 0.5^2\,\mathbf{a}_2 \\
\mathbf{p}_2 &= \mathbf{f}(1) &&= \mathbf{a}_0 + 1^1\,\mathbf{a}_1 + 1^2\,\mathbf{a}_2
\end{aligned}$$

The control points are the geometric constraints or boundary conditions and are completely user specified

- or by a point and its first and second derivatives:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{f}(0.5) &&= \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 &&= \mathbf{a}_0 + 0.5^1\,\mathbf{a}_1 + 0.5^2\,\mathbf{a}_2 \\
\mathbf{p}_1 &= \mathbf{f}'(0.5) &&= \mathbf{a}_1 + 2u\mathbf{a}_2 &&= \mathbf{a}_1 + 2*0.5\mathbf{a}_2 \\
\mathbf{p}_2 &= \mathbf{f}''(0.5) &&= 2\mathbf{a}_2 &&= 2\mathbf{a}_2
\end{aligned}$$

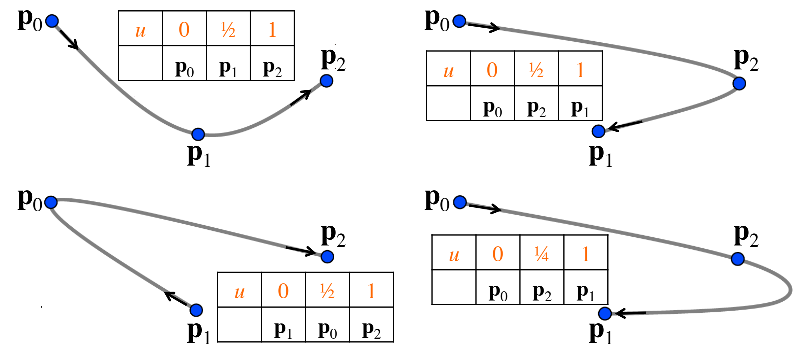
Zhang

User-Specified Control Points

Given control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, the quadratic spline:

$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u^1 + \mathbf{a}_2 u^2,\ u \in [0,1]$$

can describe any of the following motion:



Gilles

Blending Functions

The quadratic spline specified as:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{f}(0.5) &&= \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 &&= \mathbf{a}_0 + 0.5^1\,\mathbf{a}_1 + 0.5^2\,\mathbf{a}_2 \\
\mathbf{p}_1 &= \mathbf{f}'(0.5) &&= \mathbf{a}_1 + 2u\mathbf{a}_2 &&= \mathbf{a}_1 + 2*0.5\mathbf{a}_2 \\
\mathbf{p}_2 &= \mathbf{f}''(0.5) &&= 2\mathbf{a}_2 &&= 2\mathbf{a}_2
\end{aligned}$$

has

$$\mathbf{C} = \begin{bmatrix} 1 & .5 & .25 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} \text{ and } \mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & -.5 & .125 \\ 0 & 1 & -.5 \\ 0 & 0 & .5 \end{bmatrix}$$

Blending Functions

For the quadratic spline specified as:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{f}(0.5) &&= \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 &&= \mathbf{a}_0 + 0.5^1\,\mathbf{a}_1 + 0.5^2\,\mathbf{a}_2 \\
\mathbf{p}_1 &= \mathbf{f}'(0.5) &&= \mathbf{a}_1 + 2u\mathbf{a}_2 &&= \mathbf{a}_1 + 2*0.5\mathbf{a}_2 \\
\mathbf{p}_2 &= \mathbf{f}''(0.5) &&= 2\mathbf{a}_2 &&= 2\mathbf{a}_2
\end{aligned}$$

its

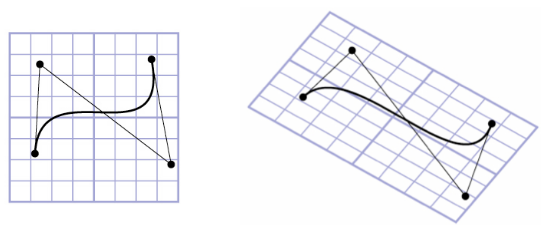
$$\mathbf{C} = \begin{bmatrix} 1 & .5 & .25 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} \text{ and } \mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & -.5 & .125 \\ 0 & 1 & -.5 \\ 0 & 0 & .5 \end{bmatrix}$$

$$\mathbf{f}(u) = \sum_{i=0}^{2} b_i(u)\mathbf{p}_i = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} \begin{bmatrix} 1 & -.5 & .125 \\ 0 & 1 & -.5 \\ 0 & 0 & .5 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

Desired Properties of Blending Functions

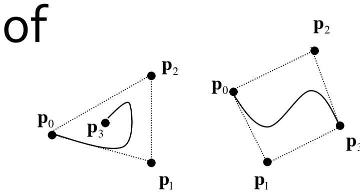
Affine invariance:

- affine combination: $\sum_{i=0}^{k} b_i(u) = 1, \ 0 \le u \le 1$
- to transform a curve, we can transform the control points and then regenerate the curve
  - perspective transformations are non-affine ⇒ only rational basis can be perspective transformed (see NURBS)



Marschner

Desired Properties of Blending Functions

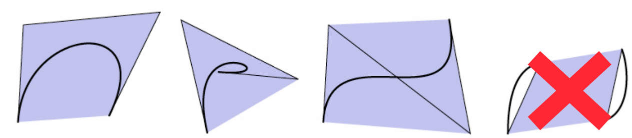


Convex hull property:

- convex combination of control points:: $\sum_{i=0}^{k} b_i(u) = 1, \ b_i(u) \ge 0, \ 0 \le u \le 1$

⇒ any point on the curve is a convex combination of its control points
  - ⇒ the curve is a weighted average of the control points
  - ⇒ no point on the curve lies outside the convex hull

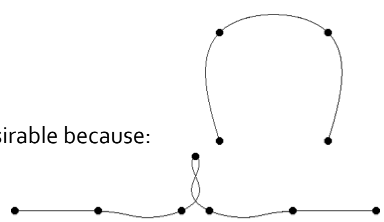
⇒ makes clipping, culling, picking, etc. simpler



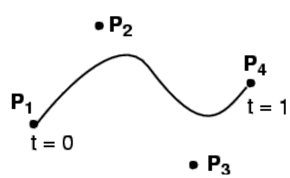
Interpolate or Approximate?

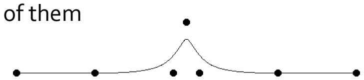
A spline can:

- interpolate some or all control points
- sounds more useful, but can be undesirable because:
  - less continuity
  - more unstable, ringing
  - lack of control between points



- approximate the control points: control points not on the spline
- control points influence the shape of spline, but does not specify it exactly
- gain local control and better behavior of spline
- when needed, control points can be computed such that the spline interpolates some of them





Durand

Splines

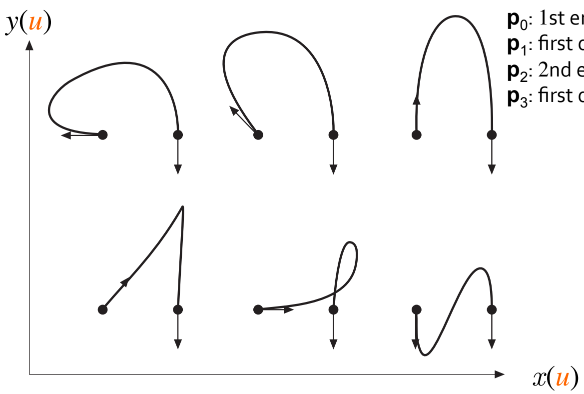
To interpolate/approximate $n+1$ control points requires a spline of order n:
- 3 control points: quadratic spline
- 4 control points: cubic spline

For cubic with four coefficients, we'd need four knowns to solve the polynomials
- for example, the two endpoints and their first derivatives
  - Recall: control points are the geometric constraints or boundary conditions and are completely user specified

Gilles,Merrell

Cubic Splines

Examples of (Hermite) cubic splines:



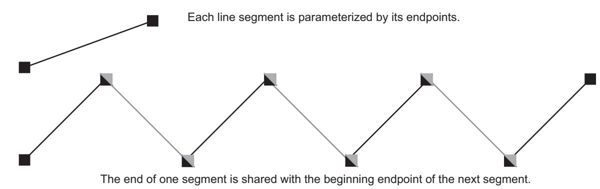
$\mathbf{p}_0$: 1st end point
$\mathbf{p}_1$: first derivative of 1st end point
$\mathbf{p}_2$: 2nd end point
$\mathbf{p}_3$: first derivative of 2nd end point

Foley, van Dam 92

Joining Splines

To interpolate a large number of control points, we can join together a number of splines

Where the splines meet are called knots/joints

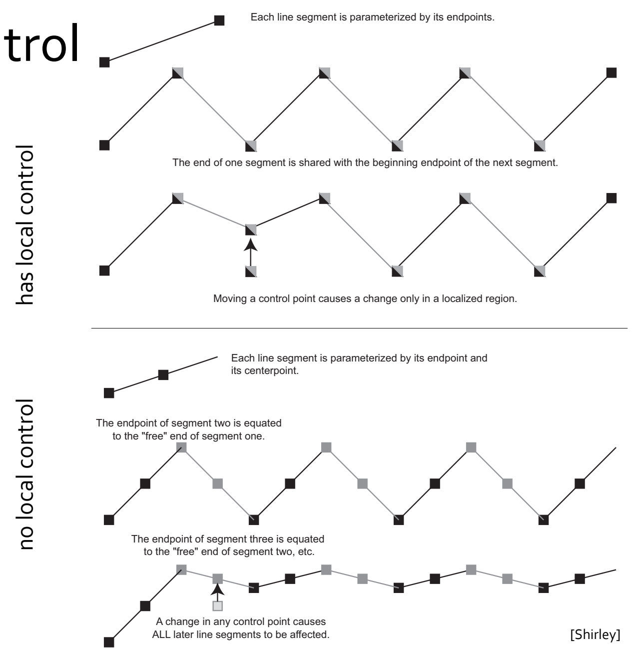


Two issues:

1. whether the combined curve has local control
2. smoothness of overall combined curve

Gilles,Merrell,Shirley

Local Control



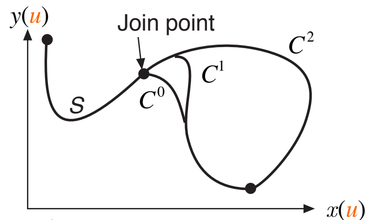
Joint Smoothness

Smoothness of overall combined curve, useful for:
- computing normals across joints in shading
- parameter interpolation between keyframes in animation

Two types of smoothness measures:

1. Parametric continuity:
   - continuity of coordinate functions $(x(u), y(u))$
   - useful for trajectories
2. Geometric continuity:
   - continuity of the curve/surface itself
   - useful in defining shapes (see modeling lectures)

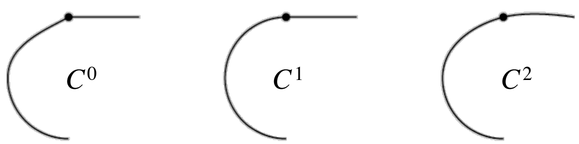
Measures of Joint Smoothness



Foley, van Dam 92

Parametric continuity:

- 0^{th} order, C^0
  curve segments meet at joint: $\mathbf{f}_2(0) = \mathbf{f}_1(1)$
- 1^{st} order, C^1
  1^{st} derivatives, velocities, equal at joint: $\mathbf{f}_2'(0) = \mathbf{f}_1'(1)$
- 2^{nd} order, C^2
  2^{nd} derivatives, accelerations, equal at joint



Marschner

Cubic Splines

Reasons we prefer to work with cubic splines in CG:

- cubics allow for C^2 continuity, quadratics offer only C^1
  - lower degree polynomials are not flexible enough
  - the three points specifying a second-order polynomial define a plane in which the polynomial lies
  - cubics are the lowest order polynomials that can be non-planar in 3D
- the greater smoothness offered by quartic* and other higher-order polynomials are rarely important
  - higher degree polynomials can introduce "wiggles" (oscillations) and are more expensive to compute
  - used mainly in designing aerodynamic curves/surfaces

*don't confuse *quartic* (4^{th} order) polynomial with *quadric* (implicit quadratic surfaces formed from conic sections)

Cubic Splines

A representation of cubic spline consists of:

- four control points (why four?)
  - these are completely user specified
  - determine a set of blending functions

There is no single "best" representation of cubic spline:

Cubic	Interpolate?	Local?	Continuity	Affine?	Convex*?	VD*?
Hermite	✔	✔	C^1	✔	n/a	n/a
Cardinal (Catmull-Rom)	except endpoints	✔	C^1	✔	no	no
Bézier	endpoints	✗	C^1	✔	✔	✔
natural	✔	✗	C^2	✔	n/a	n/a
B-Splines	✗	✔	C^2	✔	✔	✔

* n/a when some of the control "points" are tangents, not points

Hermite Cubic

Control "points": position and 1^{st} derivative of endpoints:

$$\begin{aligned}
\mathbf{f}(u) &= \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 + u^3\mathbf{a}_3 \\
\mathbf{p}_0 = \mathbf{f}(0) &= \mathbf{a}_0 + 0^1\mathbf{a}_1 + 0^2\mathbf{a}_2 + 0^3\mathbf{a}_3 \\
\mathbf{p}_1 = \mathbf{f}'(0) &= \mathbf{a}_1 + 2*0^1\mathbf{a}_2 + 3*0^2\mathbf{a}_3 \\
\mathbf{p}_2 = \mathbf{f}(1) &= \mathbf{a}_0 + 1^1\mathbf{a}_1 + 1^2\mathbf{a}_2 + 1^3\mathbf{a}_3 \\
\mathbf{p}_3 = \mathbf{f}'(1) &= \mathbf{a}_1 + 2*1^1\mathbf{a}_2 + 3*1^2\mathbf{a}_3
\end{aligned}$$

Constraint matrix

Basis matrix:

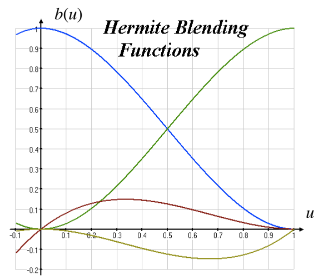
Hermite Cubic Blending Functions

$$\mathbf{f}(u) = \sum_{i=0}^{3} b_i(u)\mathbf{p}_i = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{f}(u) = (2u^3 - 3u^2 + 1)\mathbf{p}_0 + (u^3 - 2u^2 + u)\mathbf{p}_1 + (-2u^3 + 3u^2)\mathbf{p}_2 + (u^3 - u^2)\mathbf{p}_3$$

Which graph is $b_0(u)$, $b_1(u)$, $b_2(u)$, and $b_3(u)$?

How can you tell?



Hint:

$\mathbf{p}_0 = \mathbf{f}(0)$

$\mathbf{p}_1 = \mathbf{f}'(0)$

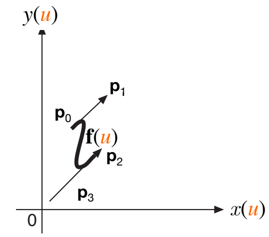
$\mathbf{p}_2 = \mathbf{f}(1)$

$\mathbf{p}_3 = \mathbf{f}'(1)$

Hodgins

Hermite Cubic Blending Functions

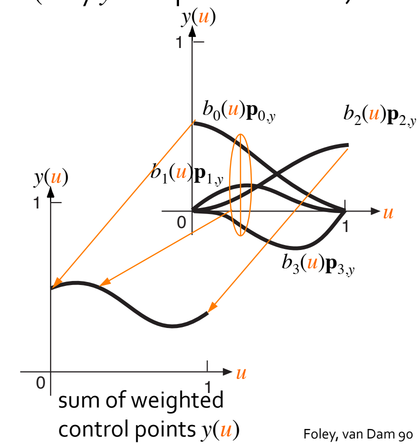
For the Hermite curve:



$\mathbf{p}_0$ is most influential at $u = 0$

Near $u = 0$, mainly b_0 and b_1 determine the curve, with b_2 and b_3 contributing

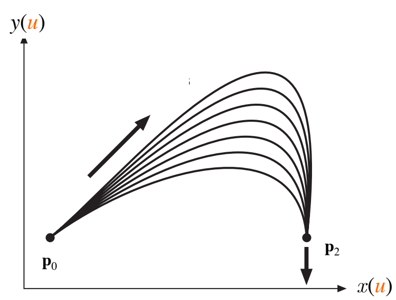
Control points weighted by the blending functions (only y-component shown):



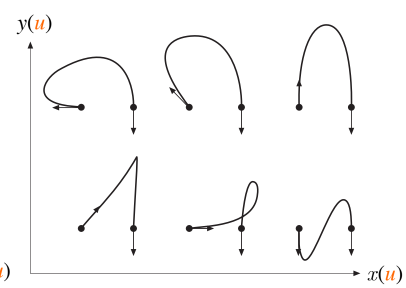
Foley, van Dam 90

Hermite Cubic Examples

Recall: the control points are the geometric constraints or boundary conditions and are completely user specified



only $\mathbf{p}_1$'s magnitude varies for each curve



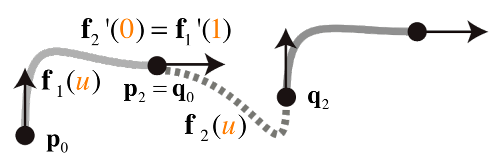
only $\mathbf{p}_1$'s direction varies for each curve

FvD 90

Hermite Cubic Chain

Can achieve C^1 continuity with: $\mathbf{q}_0 = \mathbf{f}_2(0) = \mathbf{f}_1(1) = \mathbf{p}_2$

$\mathbf{q}_1 = \mathbf{f}_2'(0) = \mathbf{f}_1'(1) = \mathbf{p}_3$



Given n control points, the chain contains $(n\text{–}2)/2$ cubic segments

The chain interpolates the control points, provides local control and is affine invariant

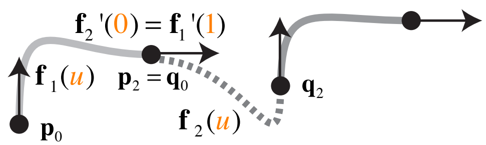
Shirley

Problem with Hermite Spline

Mixing points and tangents as control points is awkward

To get C^1, designer must explicitly specify derivative at each endpoint such that consecutive tangents are collinear

This gets tedious . . .



Hodgins,Shirley

Cubic Splines

A representation of cubic spline consists of:

- four control points (why four?)
  - these are completely user specified
  - determine a set of blending functions

There is no single "best" representation of cubic spline:

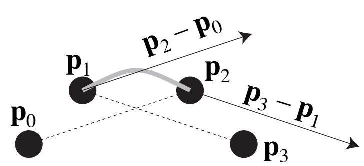
Cubic	Interpolate?	Local?	Continuity	Affine?	Convex*?	VD*?
Hermite	✔	✔	C^1	✔	n/a	n/a
Cardinal (Catmull-Rom)	except endpoints	✔	C^1	✔	no	no
Bézier	endpoints	✗	C^1	✔	✔	✔
natural	✔	✗	C^2	✔	n/a	n/a
B-Splines	✗	✔	C^2	✔	✔	✔

* n/a when some of the control "points" are tangents, not points

Cardinal Cubic Spline

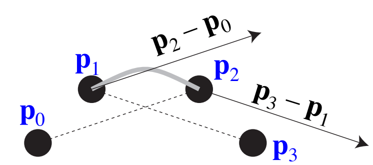
Given n control points, a cardinal cubic spline chain has n-3 segments, it interpolates all points except the endpoints

- each segment i uses as control points $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}$, so each segment shares control points with its 3 subsequent neighbors ⇒ local control
- each segment i spans only $\mathbf{p}_i, \mathbf{p}_{i+1}$
- the derivative at $\mathbf{p}_i$ is determined by the vector $(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$ the derivative at $\mathbf{p}_{i+1}$ is determined by the vector $(\mathbf{p}_{i+2} - \mathbf{p}_i)$
- since the third point of segment i is the second point of segment $i+1$, the curve is C^0
- further, the same vector determines the first derivative at the end of segment i and that at the start of segment $i+1$, hence Cardinal cubic spline (and therefore Catmull-Rom spline) has built-in C^1 continuity



Shirley

Cardinal Cubic Spline



Canonical form:

$$\mathbf{f}(u) = \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 + u^3\mathbf{a}_3$$

$$\mathbf{f}'(u) = 1\mathbf{a}_1 + 2u\mathbf{a}_2 + 3u^2\mathbf{a}_3$$

Control points:

$$\mathbf{p}_1 = \mathbf{f}(0) = \mathbf{a}_0$$

$$\mathbf{p}_2 = \mathbf{f}(1) = \mathbf{a}_0 + 1^1\mathbf{a}_1 + 1^2\mathbf{a}_2 + 1^3\mathbf{a}_3$$

$$\mathbf{f}'(0) = 1\mathbf{a}_1 + 2*0*\mathbf{a}_2 + 3*0^2*\mathbf{a}_3$$

$$\mathbf{f}'(1) = 1\mathbf{a}_1 + 2*1*\mathbf{a}_2 + 3*1^2*\mathbf{a}_3$$

$$(\mathbf{p}_2 - \mathbf{p}_0) = \mathbf{f}'(0) \Rightarrow \mathbf{p}_0 = \mathbf{p}_2 - \mathbf{f}'(0) = \mathbf{a}_0 + 0\mathbf{a}_1 + 1^2\mathbf{a}_2 + 1^3\mathbf{a}_3$$

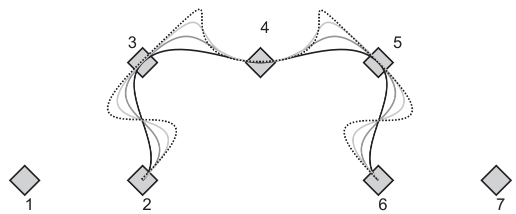
$$(\mathbf{p}_3 - \mathbf{p}_1) = \mathbf{f}'(1) \Rightarrow \mathbf{p}_3 = \mathbf{p}_1 + \mathbf{f}'(1) = \mathbf{a}_0 + 1\mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3$$

Constraint matrix:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 3 \end{bmatrix}$$

Cardinal Cubic Spline

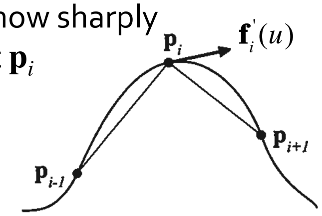
Cardinal splines have additional control parameters (beyond continuity): tension (t) and bias (b), allowing better control of the curve between control points

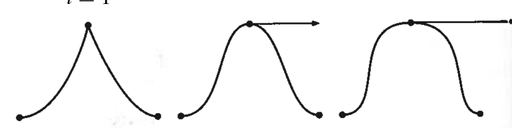


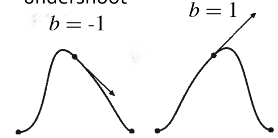
Shirley

Cardinal Cubic Spline

The tension parameter (t) controls how sharply the curve bends at the control point $\mathbf{p}_i$ by controlling the magnitude of the tangent ($\mathbf{f}_i'(u)$):







The bias parameter (b) puts weight on the left or right neighboring control point

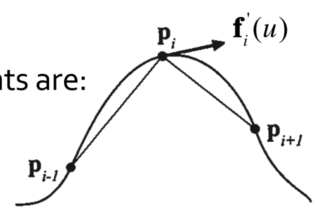
Akenine-Möller & Haines 02

Cardinal Cubic Spline with Tension

$$\mathbf{f}_i'(u) = \frac{(1-t)(1-b)}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{(1-t)(1+b)}{2}(\mathbf{p}_i - \mathbf{p}_{i-1})$$

For $b = 0$, $\mathbf{f}_i'(u) = \left(\frac{1-t}{2}\right)(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$

Let $s = \left(\frac{1-t}{2}\right)$, then the control points are:



$$\mathbf{p}_1 = \mathbf{f}(0) \;= \mathbf{a}_0$$
$$\mathbf{p}_2 = \mathbf{f}(1) \;= \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3$$
$$\mathbf{f}'(0) = 1\mathbf{a}_1 + 2*0*\mathbf{a}_2 + 3*0^2*\mathbf{a}_3$$
$$\mathbf{f}'(1) = 1\mathbf{a}_1 + 2*1*\mathbf{a}_2 + 3*1^2*\mathbf{a}_3$$
$$\mathbf{f}'(0) = s(\mathbf{p}_2 - \mathbf{p}_0) \Rightarrow \mathbf{p}_0 = \mathbf{p}_2 - s^{-1}\mathbf{f}'(0) = \mathbf{a}_0 + (1 - s^{-1})\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3$$
$$\mathbf{f}'(1) = s(\mathbf{p}_3 - \mathbf{p}_1) \Rightarrow \mathbf{p}_3 = \mathbf{p}_1 + s^{-1}\mathbf{f}'(1) = \mathbf{a}_0 + s^{-1}\mathbf{a}_1 + 2s^{-1}\mathbf{a}_2 + 3s^{-1}\mathbf{a}_3$$

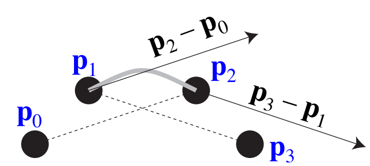
Akenine-Möller & Haines 02

Cardinal Cubic Spline

Cardinal cubic spline with bias (b) and tension (t) $= 0$ is also known as the Catmull-Rom spline

Control points ($s = ½$):

$$\mathbf{p}_1 = \mathbf{f}(0) \;= \mathbf{a}_0$$
$$\mathbf{p}_2 = \mathbf{f}(1) \;= \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3$$
$$\mathbf{p}_2 - \mathbf{p}_0 \Rightarrow \mathbf{p}_0 = \mathbf{p}_2 - 2\mathbf{f}'(0) = \mathbf{a}_0 - \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3$$
$$\mathbf{p}_3 - \mathbf{p}_1 \Rightarrow \mathbf{p}_3 = \mathbf{p}_1 + 2\mathbf{f}'(1) = \mathbf{a}_0 + 2\mathbf{a}_1 + 4\mathbf{a}_2 + 6\mathbf{a}_3$$

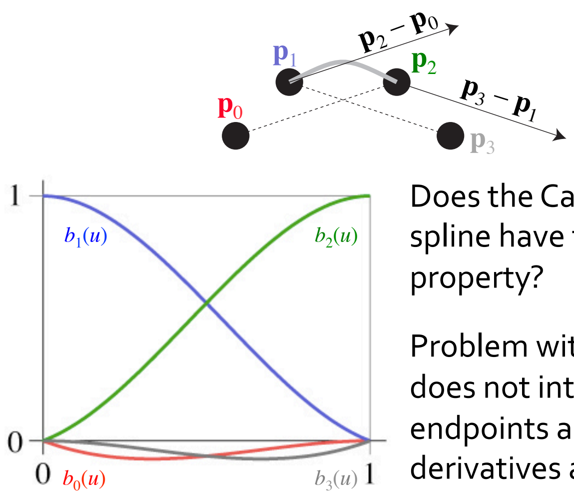


Constraint and Basis matrices:

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 6 \end{bmatrix}, \mathbf{B} = \mathbf{C}^{-1} = \frac{1}{2}\begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Shirley

Catmull-Rom Blending Functions



Does the Catmull-Rom spline have the convex hull property?

Problem with Catmull-Rom: does not interpolate endpoints and no control of derivatives at endpoints

Shirley,Marschner