Topic 15:

Interpolating Curves

- Intro to curve interpolation & approximation
- Polynomial interpolation
- Bézier curves
- Cardinal splines

Interactive Design of 2D Curves

Goal: expand vocabulary of modeling primitives beyond global analytic shapes and 3D meshes

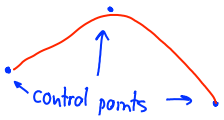
Criteria:

- Natural & intuitive interaction
- Controllable smoothness
- Adjustable resolution
- Analytic derivatives that are easy to compute
- Compact representation

Interactive Curve Design: Three Basic Tasks

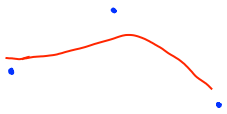
- Interpolation

  Curve goes through control points



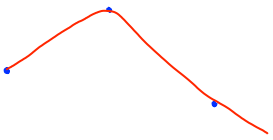
- Approximation

  Curve approximates but does not go through the control points



- Extrapolation

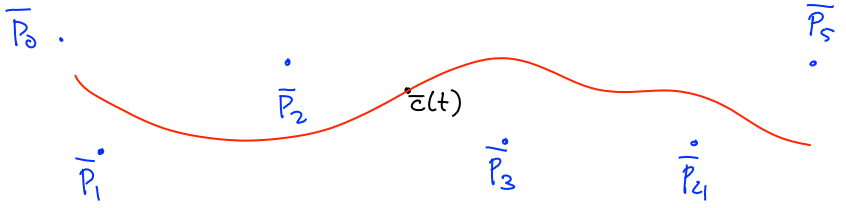
  Extend curve beyond the domain of the control points



2D Curve Design: General Problem Statement

Given N control points $\bar{P}_i$, $i = 0, \ldots, N-1$

① define a curve $\bar{c}(t)$, $t \in [0, N-1]$ that interpolates/approximates them



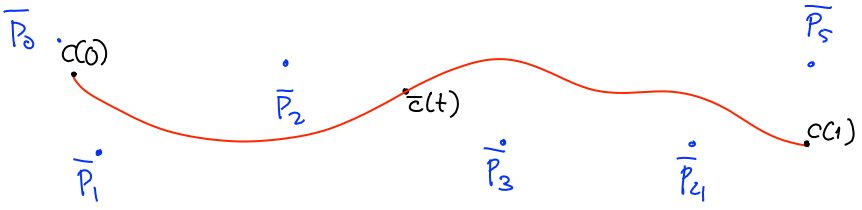
② compute its derivatives (and tangents, normals, etc)

2D Curve Design: General Problem Statement

Given N control points $\bar{P}_i$, $i = 0, \ldots, N-1$

① define a curve $\bar{c}(t)$, $t \in [0,1]$ that interpolates/approximates them

Convention: we assume that t ranges from 0 to 1

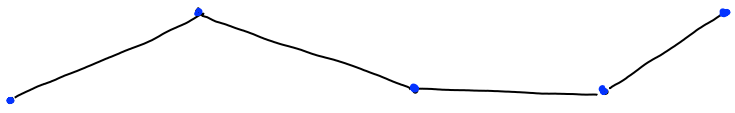


② compute its derivatives (and tangents, normals, etc)

Linear Interpolation

The simplest possible interpolation technique

Create a piecewise linear curve that connects the control points



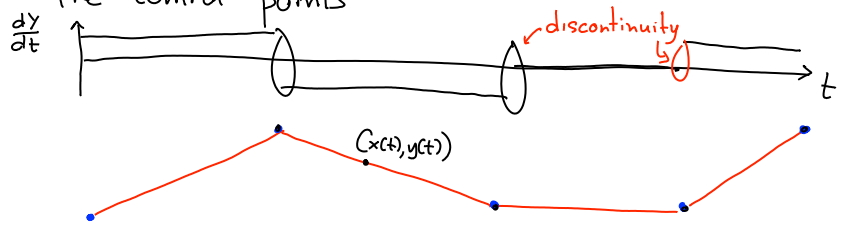
Q: What is the disadvantage of this technique?

Ans: The curve may be continuous, but its derivatives are not!

Linear Interpolation

The simplest possible interpolation technique

Create a piecewise linear curve that connects the control points

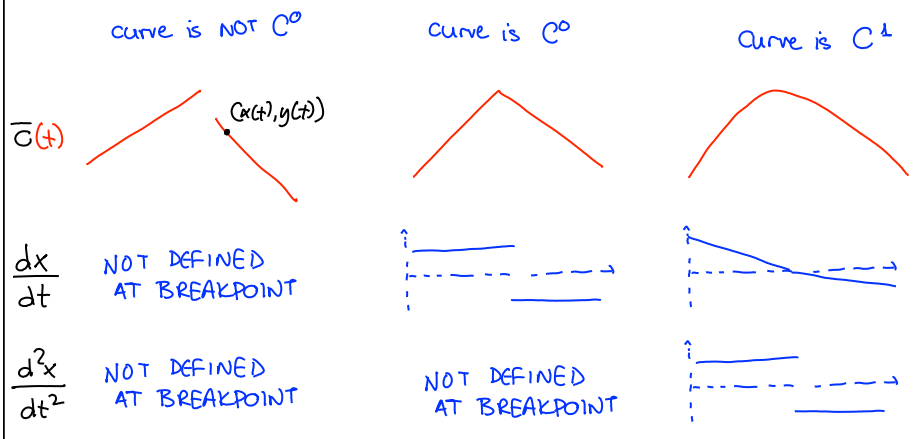


Q: What is the disadvantage of this technique?

Ans: The curve may be continuous, but its derivatives are not!

C^n Continuity

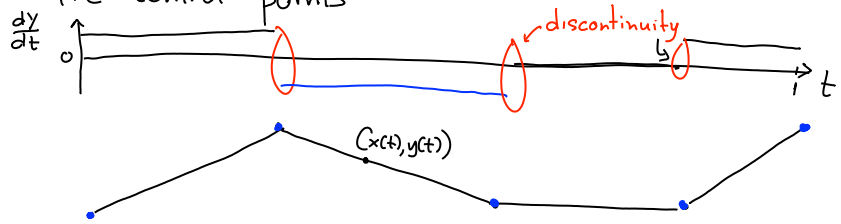
Definition A function is called C^n if its n-th order derivative is continuous everywhere



Linear Interpolation

The simplest possible interpolation technique

Create a piecewise linear curve that connects the control points



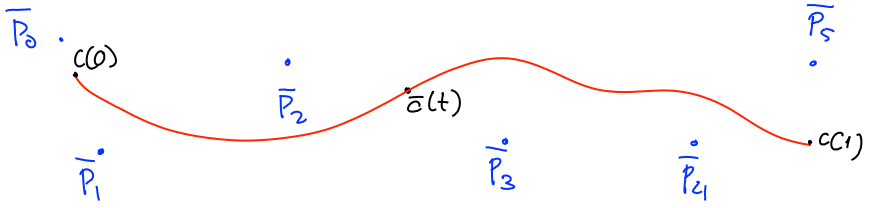
Q: What is the disadvantage of this technique?

Ans: Curve only has C^0 continuity

2D Curve Design: General Problem Statement

Given N control points $\bar{P}_i$, $i = 0, \ldots, N-1$

① define a curve $\bar{c}(t)$, $t \in [0,1]$ that interpolates/approximates them



② compute its derivatives (and tangents, normals, etc)

* We will seek functions that are at least C^1

Topic 15:

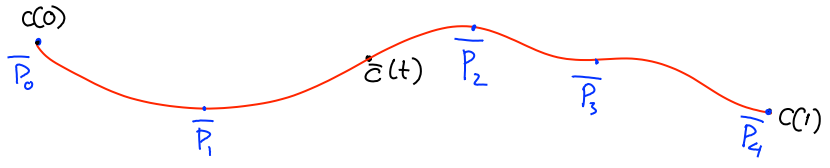
Interpolating Curves

- Intro to curve interpolation & approximation
- Polynomial interpolation
- Bézier curves
- Cardinal splines

General Polynomial Interpolation

Given N control points $\bar{P}_i = (x_i, y_i)$ $i = 0, \cdots, N-1$

① define $(N-1)$ – order polynomials $x(t), y(t)$
such that $x(\frac{i}{N-1}) = x_i$, $y(\frac{i}{N-1}) = y_i$ for $i = 0, \cdots, N-1$

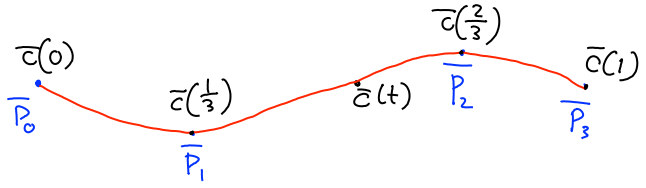


② compute its derivatives (and tangents, normals, etc)

Cubic Interpolation

Given 4 control points $\bar{P}_i = (x_i, y_i)$ $i = 0, \dots, N-1$

① define 3rd-order polynomials $x(t), y(t)$
such that $x(\frac{i}{3}) = x_i$, $y(\frac{i}{3}) = y_i$ for $i = 0, \dots, 3$



② compute its derivatives (and tangents, normals, etc)

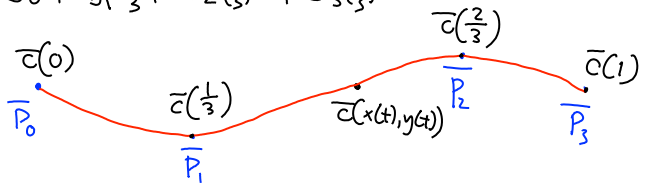
Cubic Interpolation: Basic Equations

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$
compute a_i, b_i

Equations for one control point

$$x_1 = a_0 + a_1 \cdot \tfrac{1}{3} + a_2 (\tfrac{1}{3})^2 + a_3 (\tfrac{1}{3})^3$$
$$y_1 = b_0 + b_1 \cdot \tfrac{1}{3} + b_2 (\tfrac{1}{3})^2 + b_3 (\tfrac{1}{3})^3$$



Equations in matrix form:

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & (\frac{1}{3})^2 & (\frac{1}{3})^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

Cubic Interpolation: Computing the Coeffs

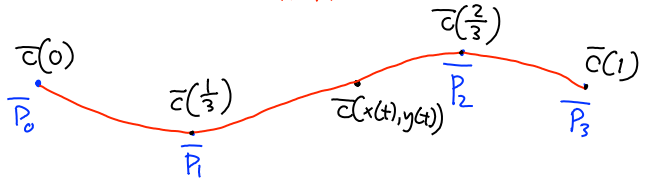
$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

$$[x_i \; y_i] = \begin{bmatrix} 1 & t_i & (t_i)^2 & (t_i)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

$[x_i \; y_i]$: known; $\begin{bmatrix} 1 & t_i & (t_i)^2 & (t_i)^3 \end{bmatrix}$: known $(t_i = i/N-1)$; matrix: unknown



Equations in matrix form:

$$[x_1 \; y_1] = \begin{bmatrix} 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

Cubic Interpolation: Computing the Coeffs

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

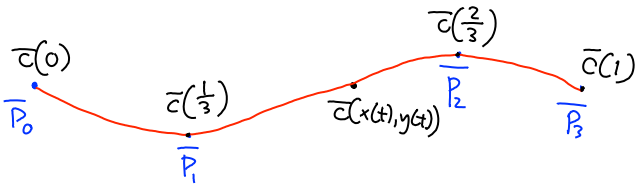
Eqs for 4 control points:

$$\underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{C} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & (1/3)^2 & (1/3)^3 \\ 1 & 2/3 & (2/3)^2 & (2/3)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix}}_{X}$$

C: known; A: known; X: unknown

$\Rightarrow$ solve system in terms of unknown matrix

$$X = A^{-1} C$$



Cubic Interpolation: Computing the Coeffs

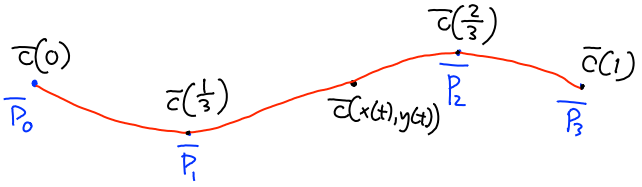
$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

Coefficients of interpolating polynomial computed by

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & (1/3)^2 & (1/3)^3 \\ 1 & 2/3 & (2/3)^2 & (2/3)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$



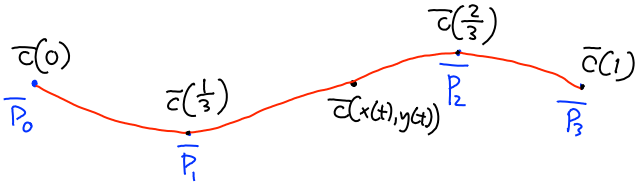
Cubic Interpolation: Evaluating the Polynomial

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

$$\begin{bmatrix} x(t) & y(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cubic Interpolation: What If < 4 Control Points?

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

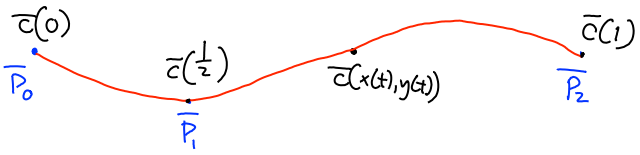
given $\overline{P_1}, \overline{P_2}, \overline{P_3}, \overline{P_4}$
compute a_i, b_i

Coefficients of interpolating polynomial computed by

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} \text{more unknowns than Eqs} \Rightarrow \text{cannot compute inverse} \end{bmatrix}^{-1} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

degree +1 ↑ ↓

← # control points →



Cubic Interpolation: What If > 4 Control Points?

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

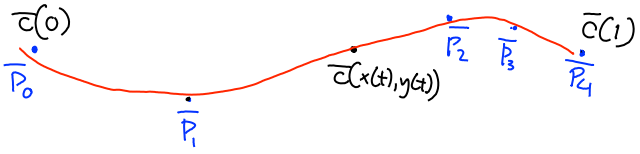
given $\overline{P_1}, \overline{P_2}, \overline{P_3}, \overline{P_4}$
compute a_i, b_i

Coefficients of interpolating polynomial computed by

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} \text{over-determined linear system} \Rightarrow \text{poly cannot pass through all pts} \end{bmatrix}^{-1} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

degree +1 ↑ ↓

← # control points →



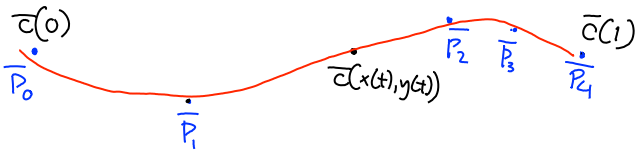
Exact Interpolation of N Points

⇒ To interpolate N points perfectly with a single polynomial we need a polynomial of degree N-1

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} \text{NxN matrix} \\ \#\text{constraints} = \\ \#\text{unknown coeffs} \end{bmatrix}^{-1} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

degree +1 ↑↓

← # control points →

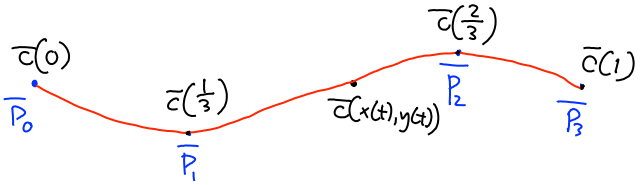


Cubic Interpolation: Evaluating Derivatives

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\frac{dx}{dt}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

$$\left[\frac{dx}{dt}(t) \quad \frac{dy}{dt}(t)\right] = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



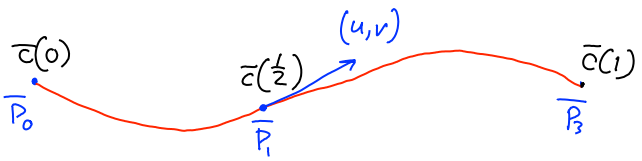
Specifying the Poly via Tangent Constraints

Instead of specifying 4 control points, we could specify 3 points and a derivative

$\Rightarrow$ replace 4th pair of eqs with

$$\begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3\left(\frac{1}{2}\right)^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

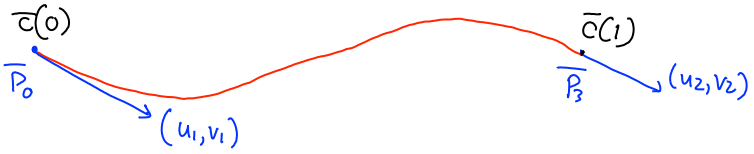
$$\begin{bmatrix} \frac{dx}{dt}(t) & \frac{dy}{dt}(t) \end{bmatrix} = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Specifying the Poly via Tangent Constraints

Instead of specifying 4 control points, we could specify 2 points and 2 derivatives (we will use this later today!)

$$\begin{bmatrix} \frac{dx}{dt}(t) & \frac{dy}{dt}(t) \end{bmatrix} = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

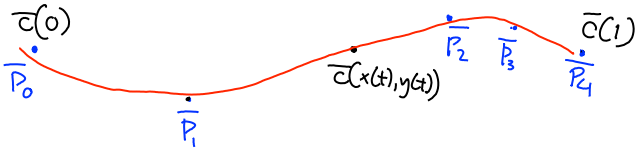


Degree-N Poly Interpolation: Major Drawback

⇒ To interpolate N points perfectly with a single polynomial we need a polynomial of degree N-1

Major drawback: It is a global interpolation scheme

i.e. moving one control point changes the interpolation of all points, often in unexpected/unintuitive ways

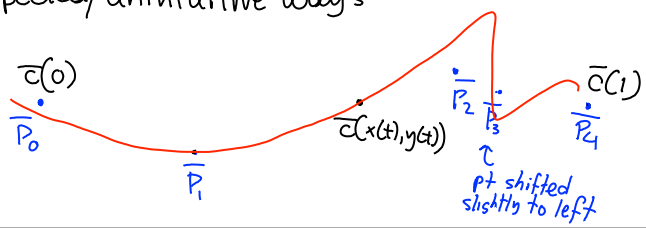


Degree-N Poly Interpolation: Major Drawback

⇒ To interpolate N points perfectly with a single polynomial we need a polynomial of degree N-1

Major drawback: It is a global interpolation scheme

i.e. moving one control point changes the interpolation of all points, often in unexpected/unintuitive ways



Topic 15:

Interpolating Curves

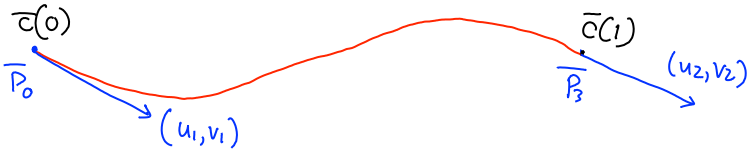
- Intro to curve interpolation & approximation
- Polynomial interpolation
- Bézier curves
- Cardinal splines

Bézier Curves

Properties

- Polynomial curves defined via endpoints and derivative constraints
- Derivative constraints defined implicitly through extra control points (that are not interpolated)

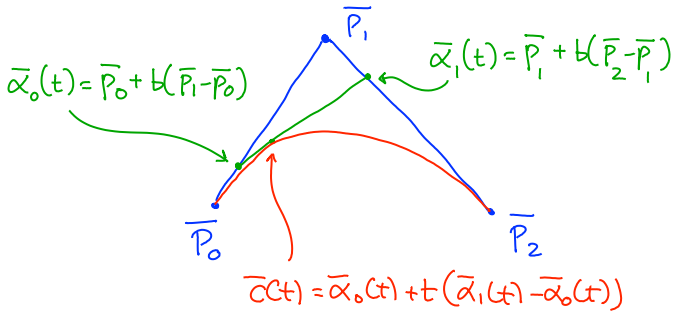
⇒ they are approximating, not interpolating curves



Bézier Curves: Main Idea

Polynomial & its derivatives expressed as a cascade of linear interpolations

Example: a double cascade



algorithm:
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t
1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{C}(t)$

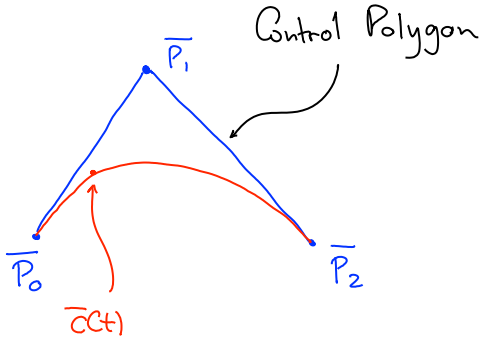
Q: Where have we seen such a cascade before?

Bézier Curves: The Control Polygon

A Bézier curve is completely determined by its control polygon

$\Rightarrow$

We manipulate the curve by manipulating its polygon

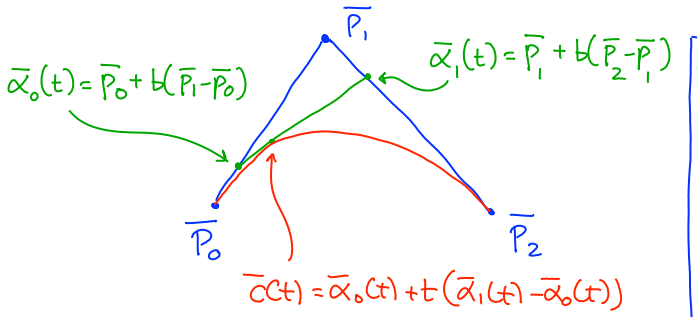


algorithm:
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t
1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{C}(t)$

Expressing the Bézier Curve as a Polynomial

Computing the polynomial

$$\begin{aligned} c(t) &= \left[\bar{P}_0 + t(\bar{P}_1 - \bar{P}_0)\right] + t\left[\bar{P}_1 + t(\bar{P}_2 - \bar{P}_1) - \bar{P}_0 - t(\bar{P}_1 - \bar{P}_0)\right] \\ &= \bar{P}_0(1 - t - t + t^2) + \bar{P}_1(t + t - t^2 - t^2) + \bar{P}_2 t^2 \\ &= \bar{P}_0(1-t)^2 + 2\bar{P}_1 t(1-t) + \bar{P}_2 t^2 \end{aligned}$$



algorithm:
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t
1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{c}(t)$

Derivatives of the Bézier Curve

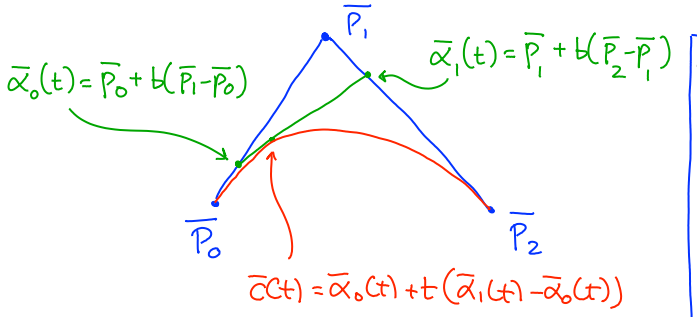
Computing the polynomial's derivatives:

$$\frac{d}{dt}c(t) = -2(1-t)\bar{P}_0 + 2\bar{P}_1(1-2t) + \bar{P}_2 2t$$

$= 2(\bar{P}_1 - \bar{P}_0)$ at $t=0$

$= 2(\bar{P}_2 - \bar{P}_1)$ at $t=1$

$$c(t) = \bar{P}_0(1-t)^2 + 2\bar{P}_1 t(1-t) + \bar{P}_2 t^2$$

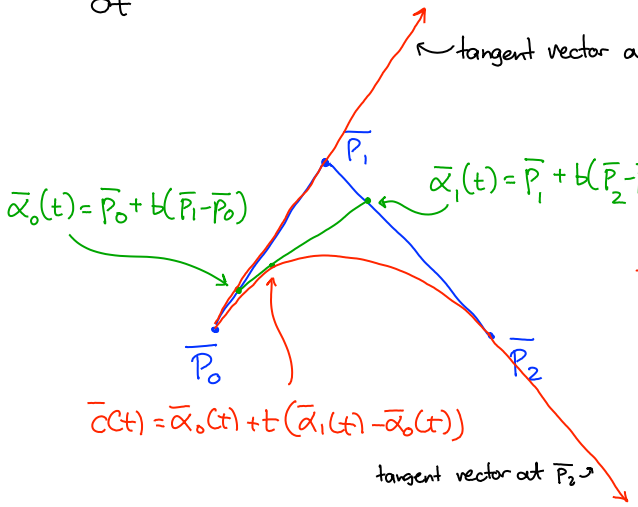


algorithm:
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t
1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{c}(t)$

Bézier Curves: Endpoint & Tangent Constraints

Computing the polynomial's derivatives:

$$\frac{d}{dt}c(t) = -2(1-t)P_0 + 2P_1(1-2t) + P_2 2t \quad \begin{array}{l} = 2(P_1-P_0) \text{ at } t=0 \\ = 2(P_2-P_1) \text{ at } t=1 \end{array}$$



General behavior:

- 1st & 3rd control pts define the endpoints
- 2nd control point define the tangent vector at endpoints

Bézier Curve Polynomial: Basic Expression

Expression in compact form

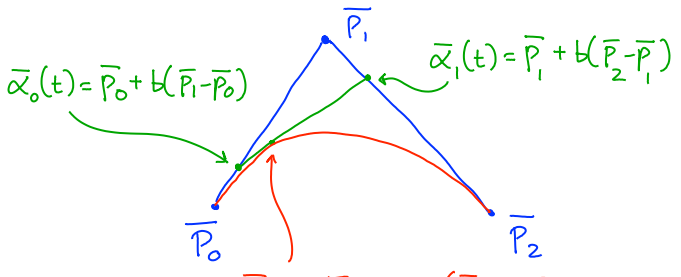
$$c(t) = \sum \bar{P}_i B_i^2(t) \quad \text{where}$$

curve ↑ control pt ↑

called the Bernstein Polynomials of degree 2

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

$$c(t) = \bar{P}_0(1-t)^2 + 2\bar{P}_1 t(1-t) + \bar{P}_2 t^2$$



algorithm:
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t

1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{c}(t)$

Bézier Curves: Generalization to N+1 points

Expression in compact form

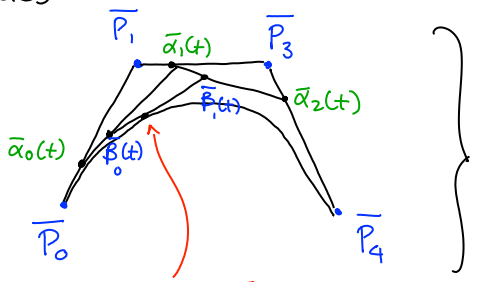
$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t) \quad \text{where}$$

($\bar{C}(t)$: curve; $\bar{P}_i$: control pt)

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

Curve defined by N linear interpolation cascades



Example for 4 points and 3 cascades

$$\bar{C}(t) = \bar{\beta}_0(t) + t(\bar{\beta}_1(t) - \bar{\beta}_0(t))$$

Bézier Curves: A Different Perspective

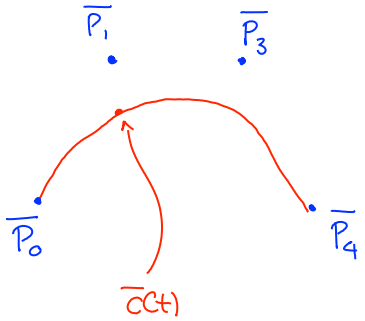
Expression in compact form

$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t) \quad \text{where}$$

($\bar{C}(t)$: curve; $\bar{P}_i$: control pt)

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$



- Each curve point $C(t)$ is a "blend" of the 4 control pts
- The blend coeffs depend on t
- They are Bernstein polynomials.

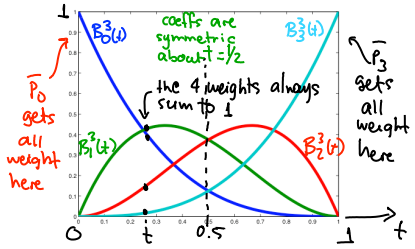
Bézier Curves as "Blends" of the Control Points

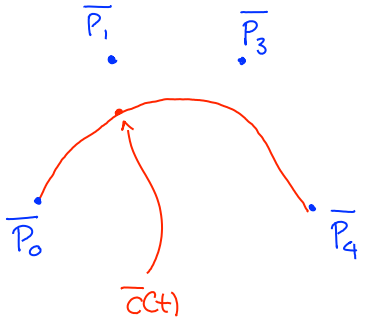
Expression in compact form

$$\overline{C}(t) = \sum_{i=0}^{N} \overline{P}_i B_i^N(t)$$

($\overline{C}(t)$: curve; $\overline{P}_i$: control pt)

with $\sum_{i=0}^{N} B_i^N(t) = 1$ for all t





- Each curve point $C(t)$ is a "blend" of the 4 control pts
- The "blending weights" depend on t
- They are Bernstein polynomials.

Bézier Curves: Useful Properties

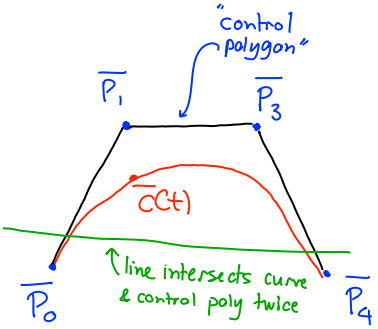
Expression in compact form

$$\overline{C}(t) = \sum_{i=0}^{N} \overline{P}_i B_i^N(t) \quad \text{where}$$

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

1. **Affine invariance**
   Transforming a Bézier curve by affine transform T $\Leftrightarrow$ transforming its control pts by T
2. **Diminishing variations**
   No line will intersect the curve at more pts than the control polygon $\Rightarrow$ curve cannot exhibit "excessive fluctuations"
3. **Linear precision**
   If control poly approximates a line, so will the curve



Bézier Curves: Useful Properties

Expression in compact form

$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t) \quad \text{where}$$

called the Bernstein Polynomials of degree N

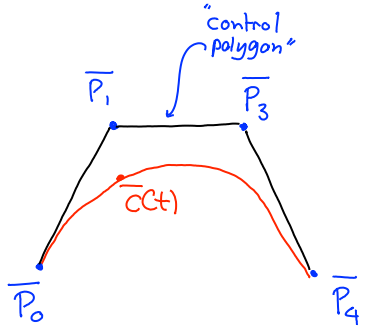
$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

④ Tangents at endpoints are along the 1st & last edges of control polygon:

$$\frac{d}{dt}\bar{C}(t) = \sum_{i=1}^{N} \bar{P}_i \frac{d}{dt} B_i^N(t)$$

w/ some work $= N \sum_{i=0}^{N-1} (\bar{P}_{i+1} - \bar{P}_i) B_i^{N-1}(t)$

$N(\bar{P}_1 - \bar{P}_0)$ for $t=0$ $\quad$ $N(\bar{P}_N - \bar{P}_{N-1})$ for $t=1$



Bézier Curves: Pros & Cons

Expression in compact form

$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t) \quad \text{where}$$

called the Bernstein Polynomials of degree N

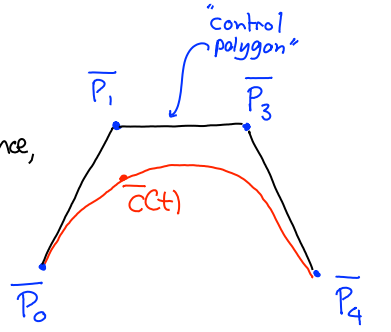
$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

Advantages

- Intuitive control for $N \leq 3$
- Derivatives easy to compute
- Nice properties (affine invariance, diminishing variations)

Disadvantages

- Scheme is still global (curve function of all control pts)



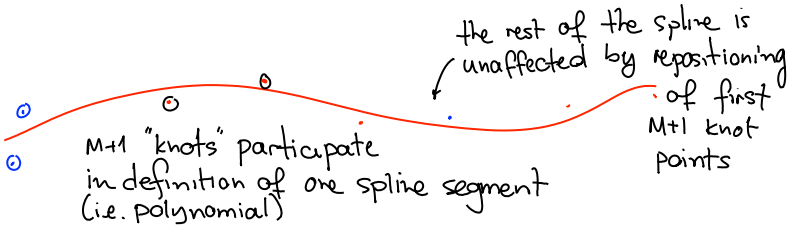
Topic 15:

Interpolating Curves

- Intro to curve interpolation & approximation
- Polynomial interpolation
- Bézier curves
- Cardinal splines

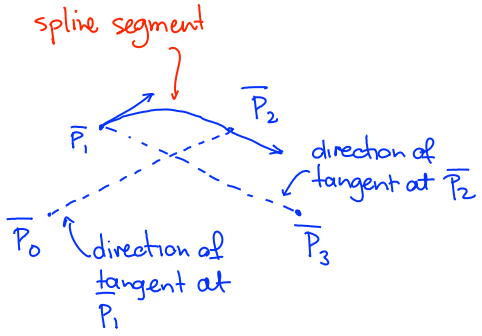
Splines: Local Control + Reduced Continuity

- Idea: provide local control of curve approximating N points by sacrificing C^N continuity
- M-degree spline: a piecewise-polynomial curve of degree M
- Splines often defined to have C^{M-2} continuity at the knot points (aka. control points)



Cubic Cardinal Splines

- Idea: provide local control of curve interpolating N-2 points, with C^1 continuity
- Cubic polynomials used as basic building block



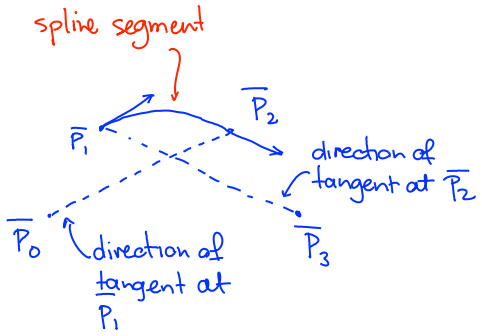
$\bar{P}_1, \bar{P}_2$ act as endpoint constraints

$\bar{P}_0, \bar{P}_3$ define derivative constraints

Cubic Cardinal Splines: Specifying the Curve

Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$



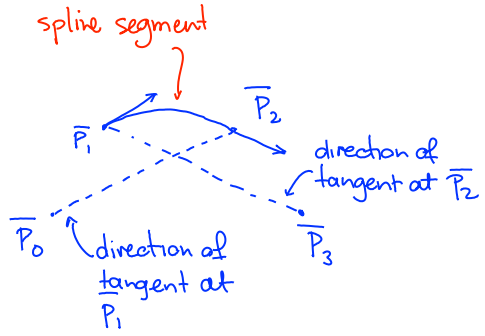
$\bar{P}_1, \bar{P}_2$ act as endpoint constraints

$\bar{P}_0, \bar{P}_3$ define derivative constraints

Cubic Cardinal Splines: Defining 1st Segment

Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$



$\bar{P}_1, \bar{P}_2$ act as endpoint constraints

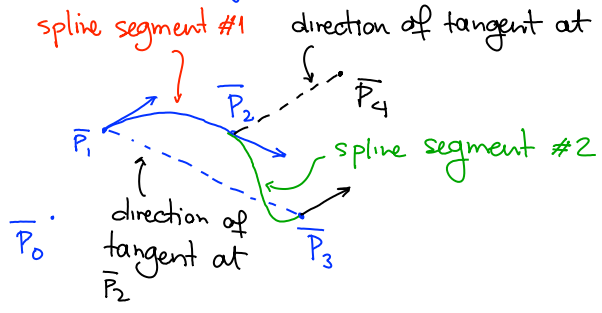
$\bar{P}_0, \bar{P}_3$ define derivative constraints

Cubic Cardinal Splines: Defining 2nd Segment

Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$

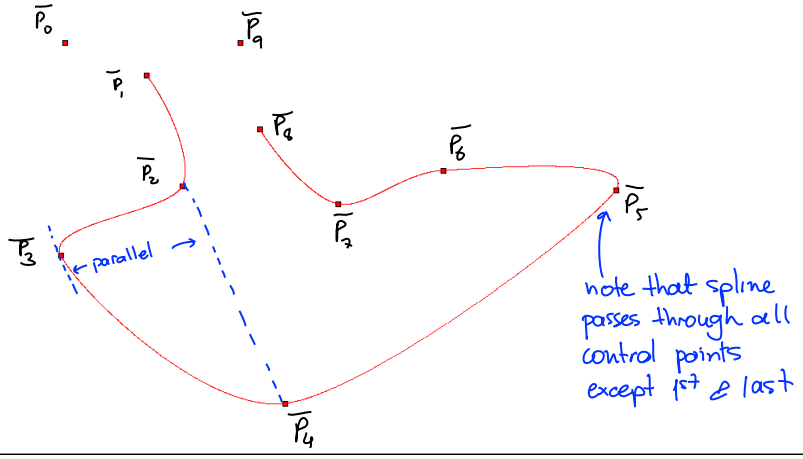
Example: Adding a fifth point adds a new segment



Cubic Cardinal Splines: General Case

Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$



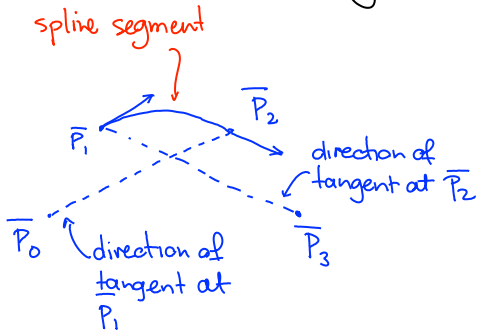
Cardinal Splines: The Strain Parameter

Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$

tangent at $\bar{P}_i = k(\bar{P}_{i+1} - \bar{P}_{i-1})$

called a strain parameter



Catmull-Rom Splines

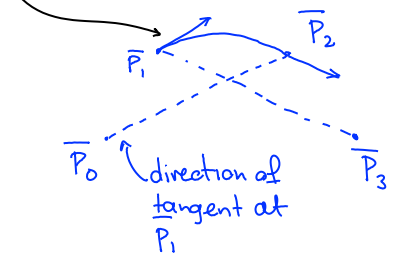
Approach: ① A user only specifies points $\bar{P}_0, \bar{P}_1, \ldots$

② tangent at $\bar{P}_i$ set to be parallel to vector connecting $\bar{P}_{i-1}$ and $\bar{P}_{i+1}$

tangent at $\bar{P}_i = k(\bar{P}_{i+1} - \bar{P}_{i-1})$

called a strain parameter

length of tangent = $\frac{1}{2}$ distance between $\bar{P}_0$ and $\bar{P}_2$

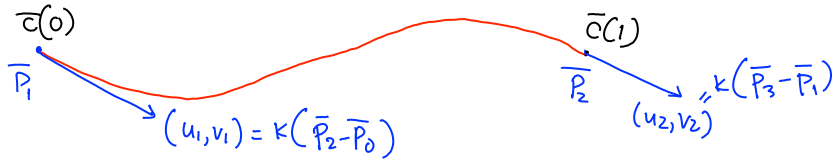


Note: if $k = \frac{1}{2}$ the spline is called a Catmull-Rom spline

Specifying the Poly via Tangent Constraints

Instead of specifying 4 control points, we could specify 2 points and 2 derivatives

$$\left[\frac{dx}{dt}(t) \quad \frac{dy}{dt}(t)\right] = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cardinal Splines: Solving for the Segment Coeffs

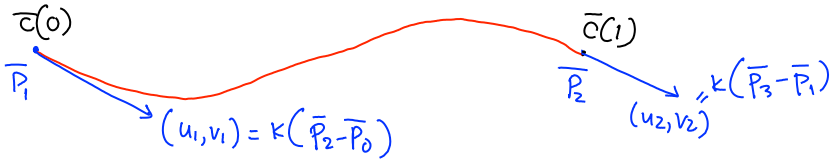
for $t=0$ $\quad [x_1\ y_1] = [1\ \cancel{t}_0\ \cancel{t^2}_0\ \cancel{t^3}_0] \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$

endpoint constraint

$$[k(x_2-x_0)\ k(y_2-y_0)] = [1\ \cancel{2t}_0\ \cancel{3t^2}_0] \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

derivative constraint

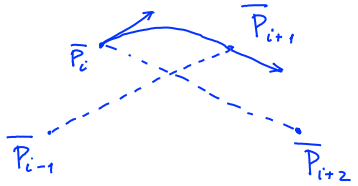
+ 2 more eqs for other endpoint ($t=1$)



Cubic Cardinal Spline Segment vs. Bézier Curve

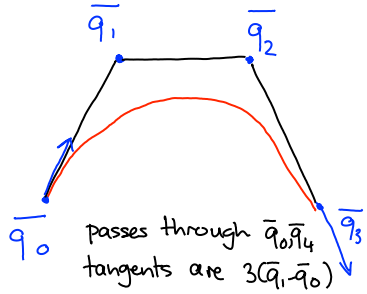
The two curves are actually equivalent:
given a cardinal spline, we can compute the control polygon of the equivalent Bézier curve

Cardinal Spline Segment



passes through $\bar{P}_i, \bar{P}_{i+1}$
tangents are $k(\bar{P}_{i+1} - \bar{P}_{i-1})$
$k(\bar{P}_{i+2} - \bar{P}_i)$

Bezier Curve



passes through $\bar{q}_0 \bar{q}_4$
tangents are $3(\bar{q}_1 - \bar{q}_0)$
$3(\bar{q}_3 - \bar{q}_2)$

Control Polygon of a Cardinal Spline Segment

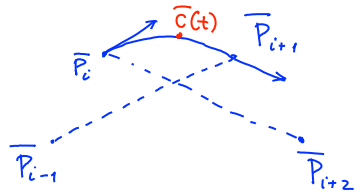
In order to have $\bar{c}(t) = \bar{r}(t)$ for all t, it must be

$$\bar{q}_0 = \bar{P}_i \quad , \quad \bar{q}_4 = \bar{P}_{i+1}$$

$$k(\bar{P}_{i+1} - \bar{P}_{i-1}) = 3(\bar{q}_1 - \bar{q}_0) \Rightarrow k(\bar{P}_{i+1} - \bar{P}_{i-1}) = 3\bar{q}_1 - 3\bar{q}_0$$

$$\Rightarrow \bar{q}_1 = \frac{k}{3}(\bar{P}_{i+1} - \bar{P}_{i-1}) + \bar{P}_i$$

Cardinal Spline Segment

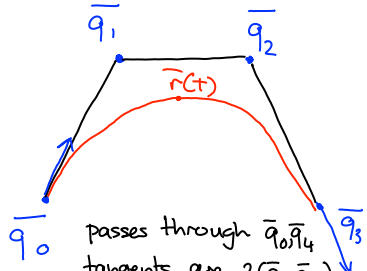


passes through $\bar{P}_i, \bar{P}_{i+1}$

tangents are $k(\bar{P}_{i+1} - \bar{P}_{i-1})$

$k(\bar{P}_{i+2} - \bar{P}_i)$

Bezier Curve



passes through $\bar{q}_0, \bar{q}_4$

tangents are $3(\bar{q}_1 - \bar{q}_0)$

$3(\bar{q}_3 - \bar{q}_2)$

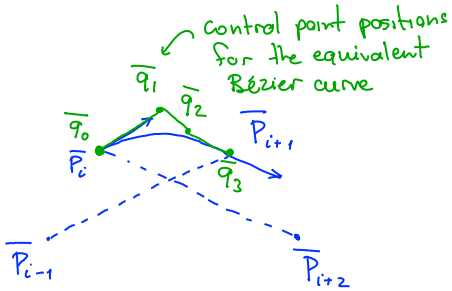
Control Polygon of a Cardinal Spline Segment

In order to have $\bar{c}(t) = \bar{r}(t)$ for all t, it must be

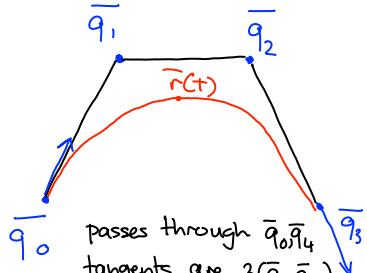
$$\bar{q}_0 = \bar{P}_i \quad , \quad \bar{q}_4 = \bar{P}_{i+1}$$

$$\bar{q}_1 = \frac{k}{3}(\bar{P}_{i+1} - \bar{P}_{i-1}) + \bar{P}_i \quad , \quad \bar{q}_2 = \bar{P}_{i+1} - \frac{k}{3}(\bar{P}_{i+2} - \bar{P}_i)$$

Cardinal Spline Segment



Bezier Curve



passes through $\bar{q}_0, \bar{q}_4$

tangents are $3(\bar{q}_1 - \bar{q}_0)$

$3(\bar{q}_3 - \bar{q}_2)$

Topic 15+:

Interpolating Surfaces

- Bézier surface patches

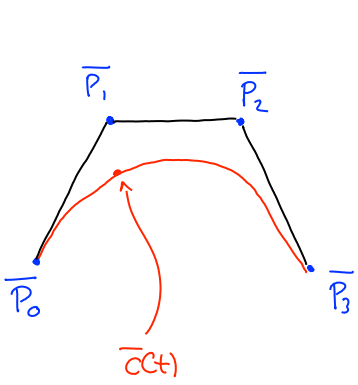
Bézier Surface Patches

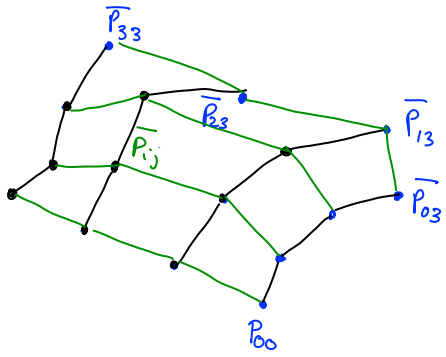
Basic idea: Define a 4×4 grid of 3D control points

$\Rightarrow$ Produces a cubic surface patch

$G(t,s)$ = 3rd-order poly in s and t

$s,t \in [0,1]$





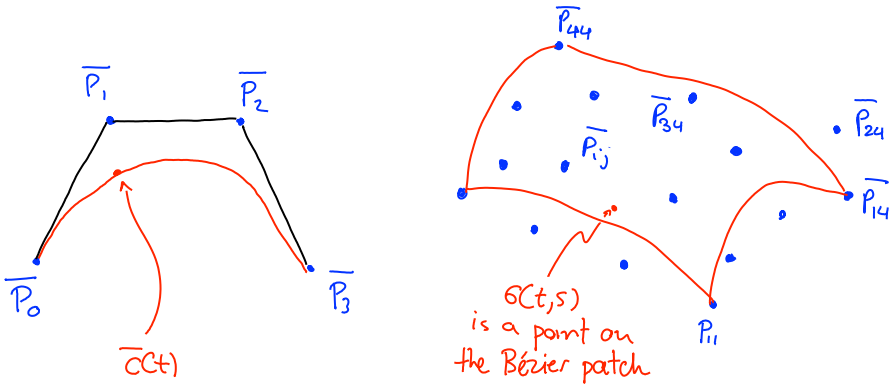
Bézier Surface Patches

Basic idea: Define a 4×4 grid of 3D control points

⇒ Produces a cubic surface patch

$G(t,s)$ = 3rd-order poly in s and t

$s,t \in [0,1]$



Bézier Surface Patches: Algorithm

Basic idea: ① Define a 4×4 grid of 3D control points

⇒ defines 4 Bézier curves

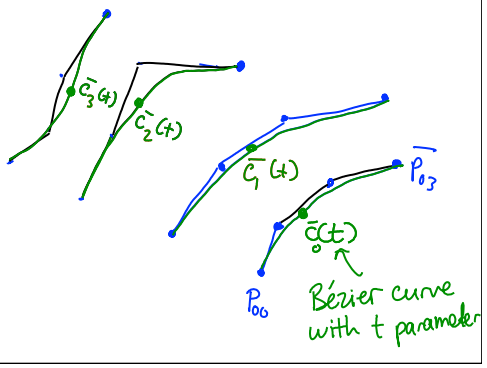
$C_0(t), C_1(t), C_2(t), C_3(t)$ parameterized by t

② Given a value for t

⇒ obtain 4 points

$\bar{C}_0(t), \bar{C}_1(t), \bar{C}_2(t), \bar{C}_3(t)$

③ These points become the control points of a Bézier curve parameterized by s



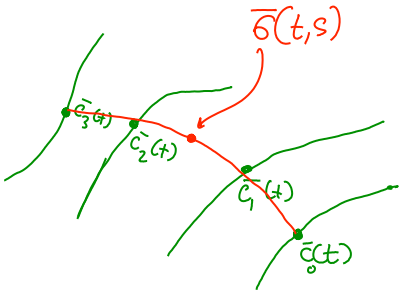
Bézier Surface Patches: Algorithm

Basic idea: ① Define a 4×4 grid of 3D control points
⇒ defines 4 Bézier curves
$C_0(t), C_1(t), C_2(t), C_3(t)$ parameterized by t

② Given a value for t
⇒ obtain 4 points
$\bar{C}_0(t), \bar{C}_1(t), \bar{C}_2(t), \bar{C}_3(t)$

③ These points become the control points of a Bézier curve parameterized by s

④ The patch is defined by points on that curve, for $s,t \in [0,1]$



Bézier Surface Patches: Polynomial Expression

Basic idea: Define a 4×4 grid of 3D control points
⇒ Produces a cubic surface patch

$\bar{G}(t,s)$ = 3rd-order poly in s and t
$s,t \in [0,1]$

General expression:

$$\bar{G}(t,s) = \sum_{i=0}^{3} \sum_{j=0}^{3} \bar{P}_{ij} \cdot \underbrace{B_i^3(t)}_{\text{Bernstein poly of degree 3}} \underbrace{B_j^3(s)}$$