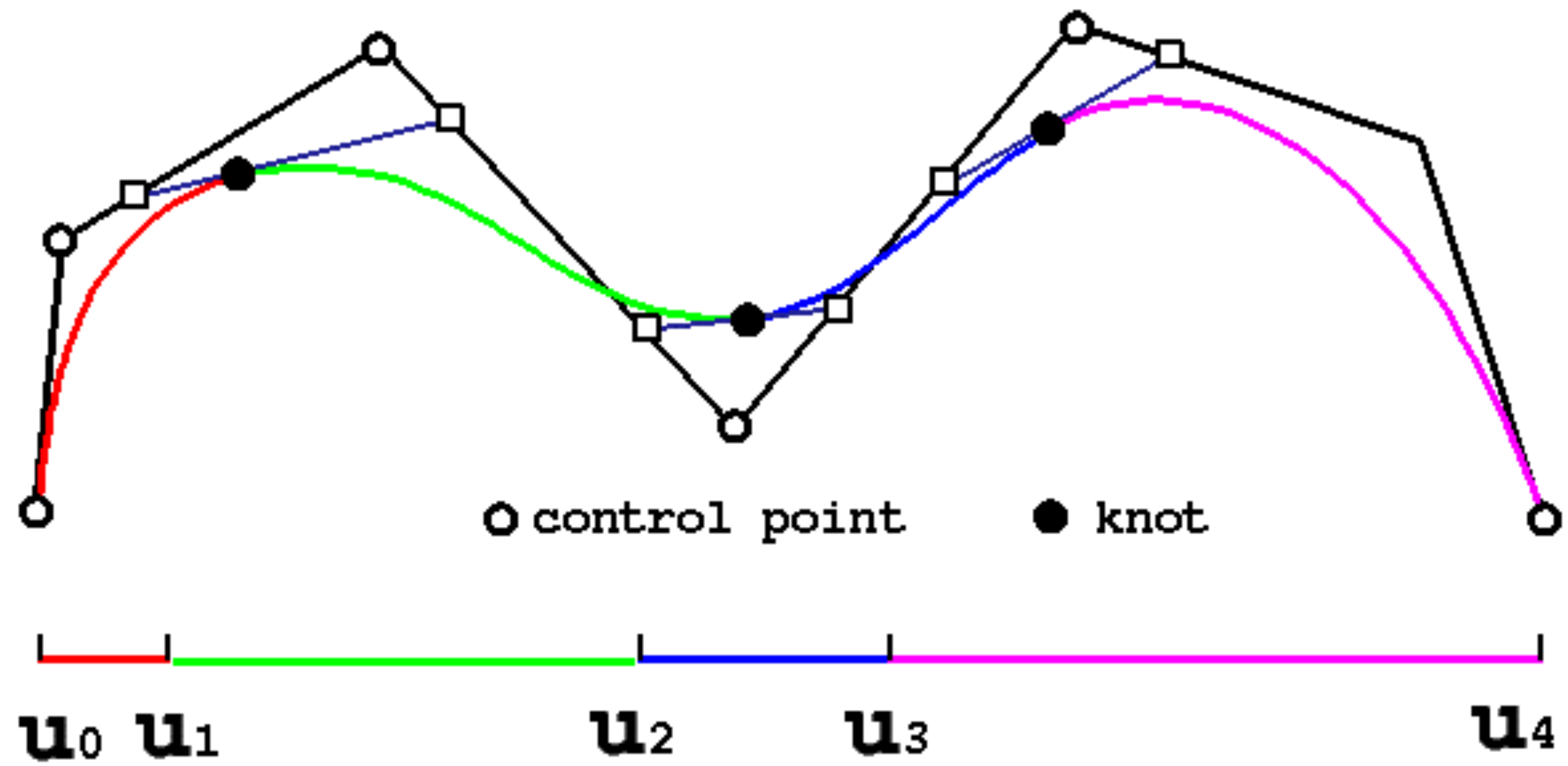
Topic 12: Interpolating Curves

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

Some slides and figures courtesy of Kyros Kutulakos
Some figures from Peter Shirley, “Fundamentals of Computer Graphics”, 3rd Ed.

What are Splines?

- Numeric function that is piecewise-defined by polynomial functions
- Possesses a high degree of smoothness where pieces connect
- These are intuitively called “knots”



History

- Used by engineers in ship building and airplane design before computers were around
- Used to create smoothly varying curves
- Variations in curve achieved by the use of weights (like control points)

Applications

- Specify smooth camera path in scene along spline curve
- Rollercoaster tracks
- Curved smooth bodies and shells (planes, boats, etc)

Motivation and Goal

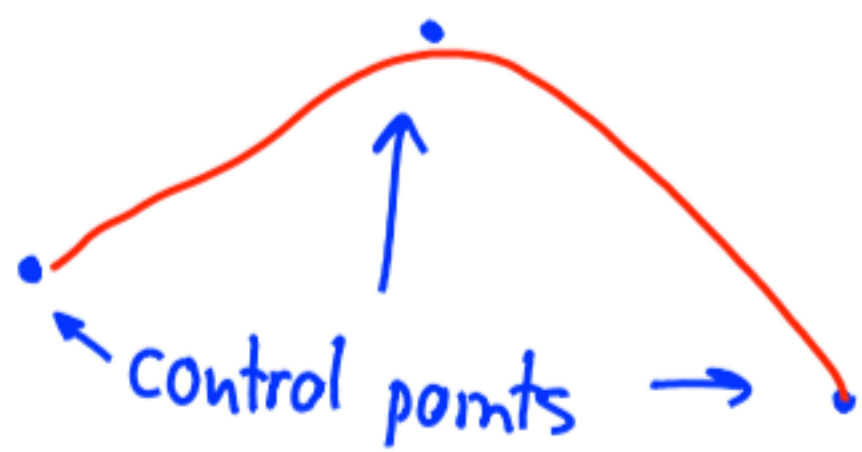
- Expand the capabilities of shapes beyond lines and conics, simple analytic functions and to allow design constraints.

Design issues

- Create curves that can have constraints specified
- Have natural and intuitive interaction
- Controllable smoothness
- Control (local vs global)
- Analytic derivatives that are easy to compute
- Compactly represented
- Other geometric properties (planarity, tangent/curvature control)

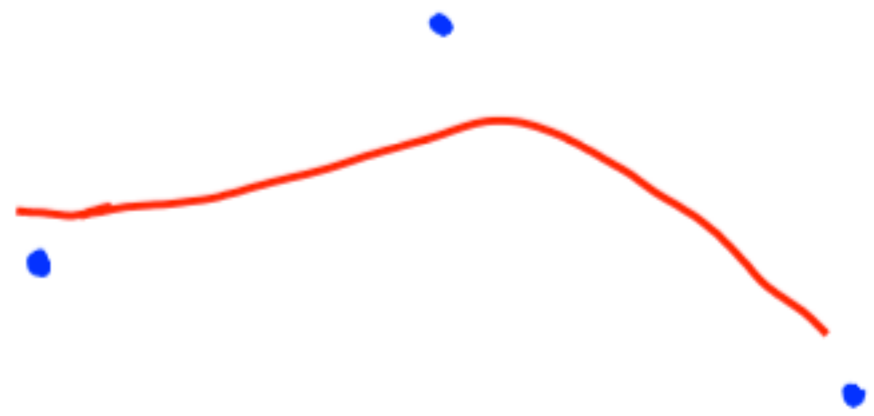
Interpolation

- Interpolating splines: pass through all the data points (control points). Example: Hermite splines



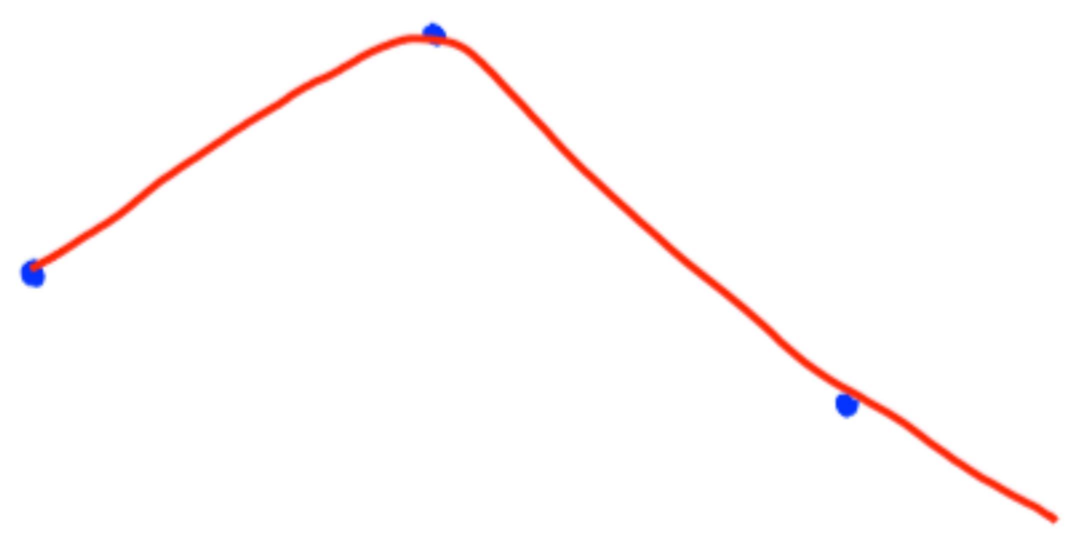
Approximation

- Curve approximates but does not go through all of the control points.
- Comes close to them.



Extrapolation

- Extend the curve beyond the domain of the control points



Local properties

- Continuity
- Position at a specific place on the curve
- Direction at a specific place on the curve
- Curvature

Global properties

- Closed or open curve
- Self intersection
- Length

Local vs Global Control

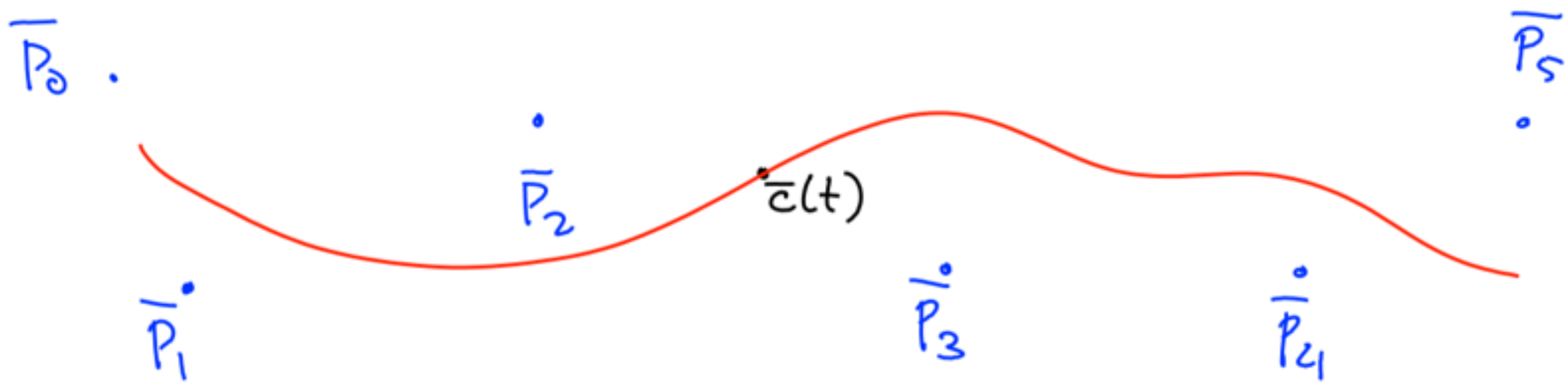
- Local control changes curve only locally while maintaining some constraints
- Modifying point on curve affects local part of curve or entire curve

Parametric and Geometric Continuity

- When piecing together smooth curves, consider the degrees of smoothness at the joints.
- Parametric Continuity: differentiability of the parametric representation (C^0, C^1, C^2, ...)
- Geometric Continuity: smoothness of the resulting displayed shape (G^0=C^0, G^1=tangent-cont., G^2=curvature-cont.)

2D Curve Design: General Problem Statement

- Given N control points, P_i, $i = 0 \ldots n - 1$, $t \in [0, 1]$ (by convention)
- Define a curve $c(t)$ that interpolates / approximates them
- 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 such a technique?

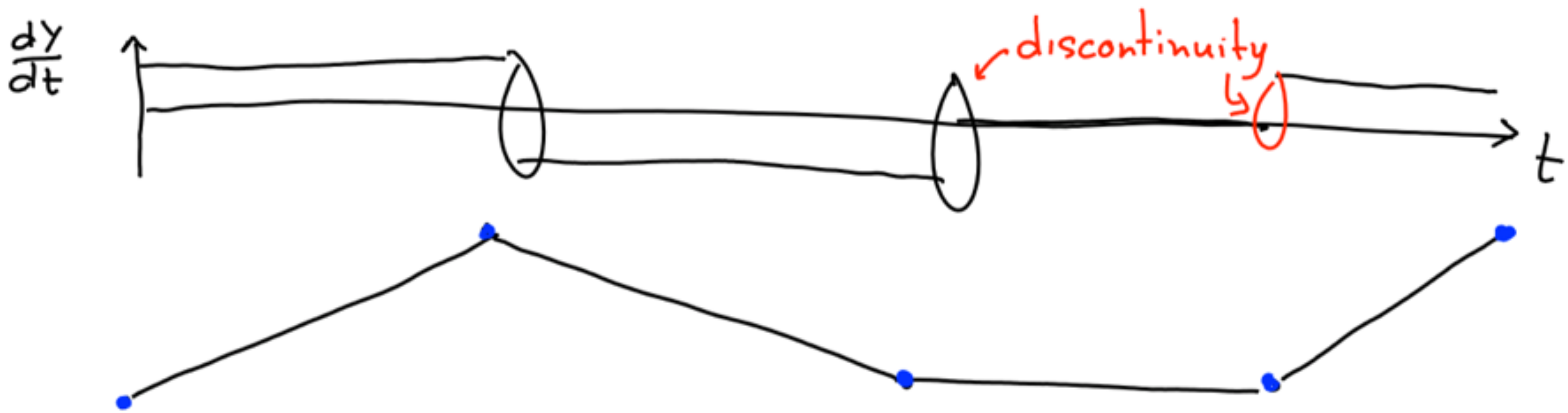
Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points

- Q: What is the disadvantage of such a technique?
- A: The curves may be continuous but its derivatives are not…

Linear Interpolation

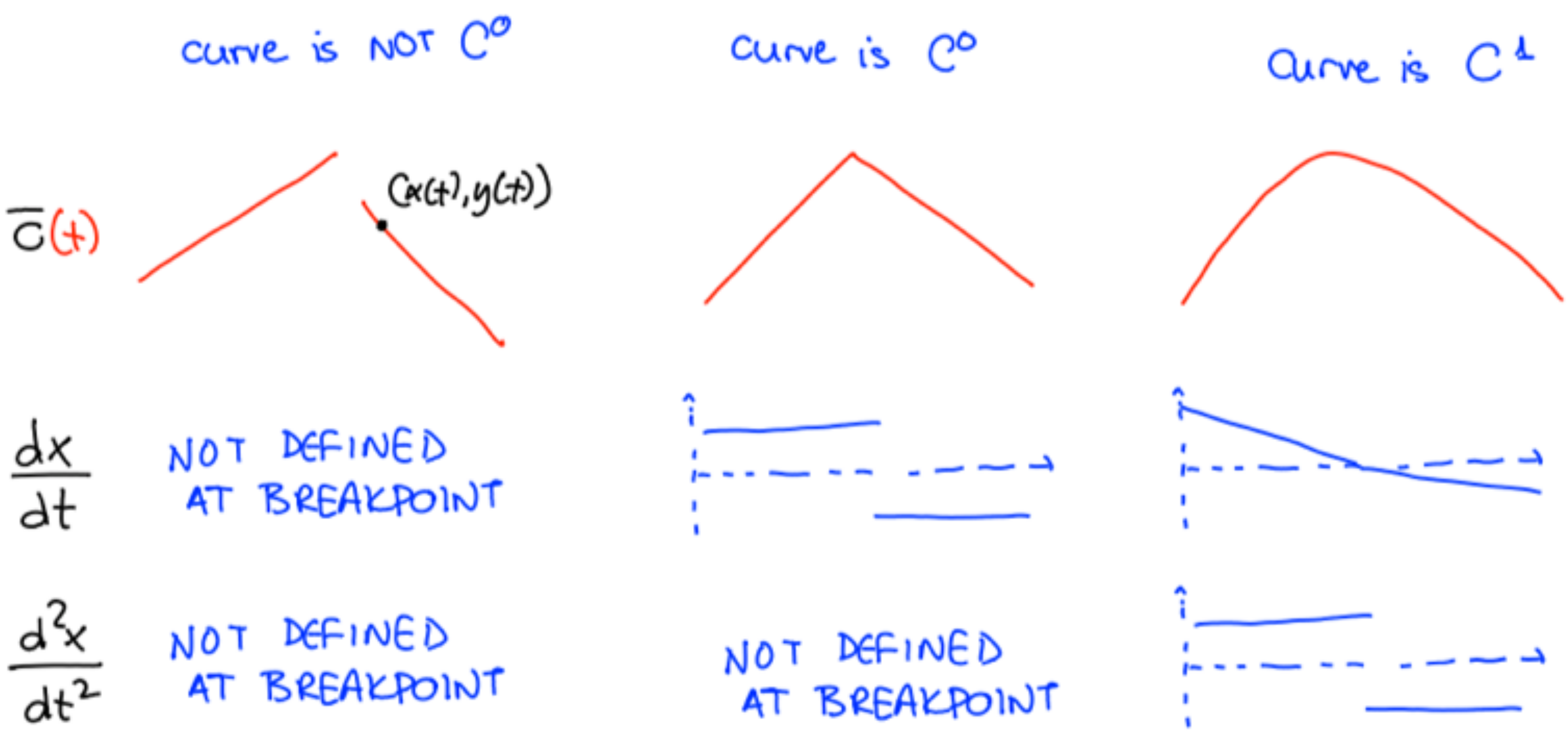
- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points



- Q: What is the disadvantage of such a technique?
- A: The curves may be continuous but its derivatives are not…

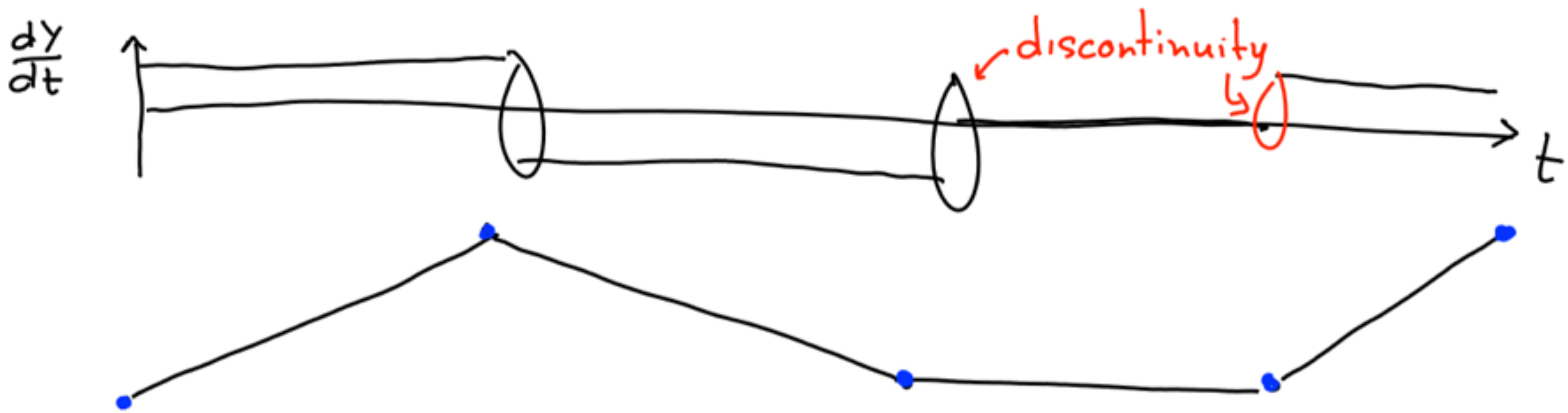
C^n continuity

- Definition: a function is called C^n if it's n^{th} order derivative is continuous everywhere



Linear Interpolation

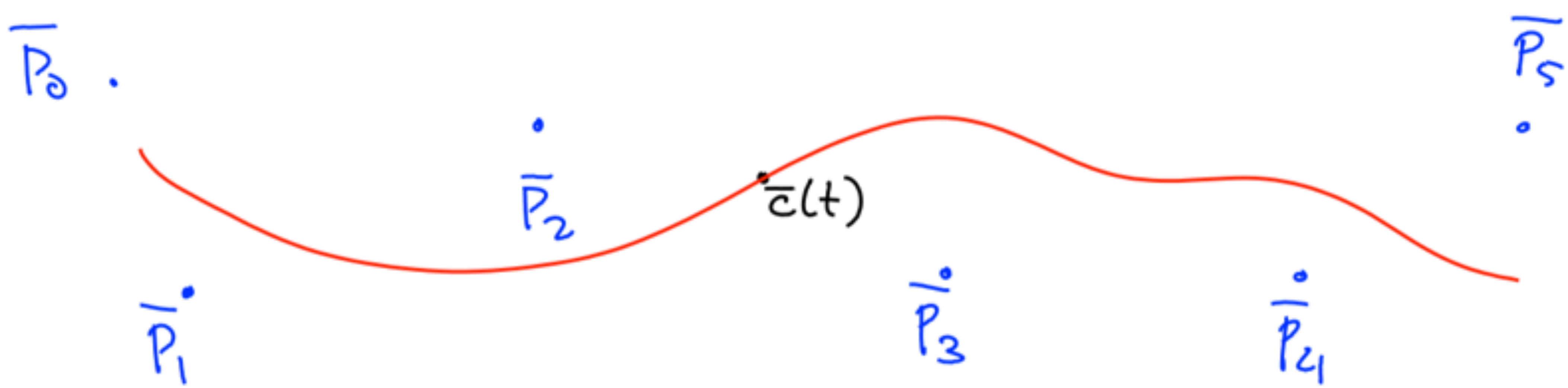
- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points



- Q: What is the disadvantage of such a technique?
- Curve has only C^0 continuity

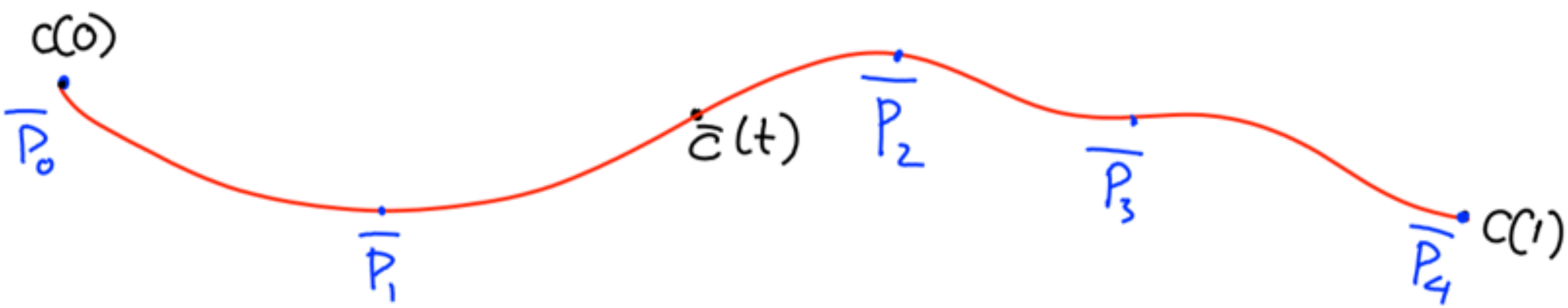
2D Curve Design: General Problem Statement

- Given N control points, P_i, i = 0…n-1, t ∈ [0, 1] (by convention)
- Define a curve c(t) that interpolates / approximates them
- Compute its derivatives (and tangents, normals etc)
- We will seek functions that are at least C^1



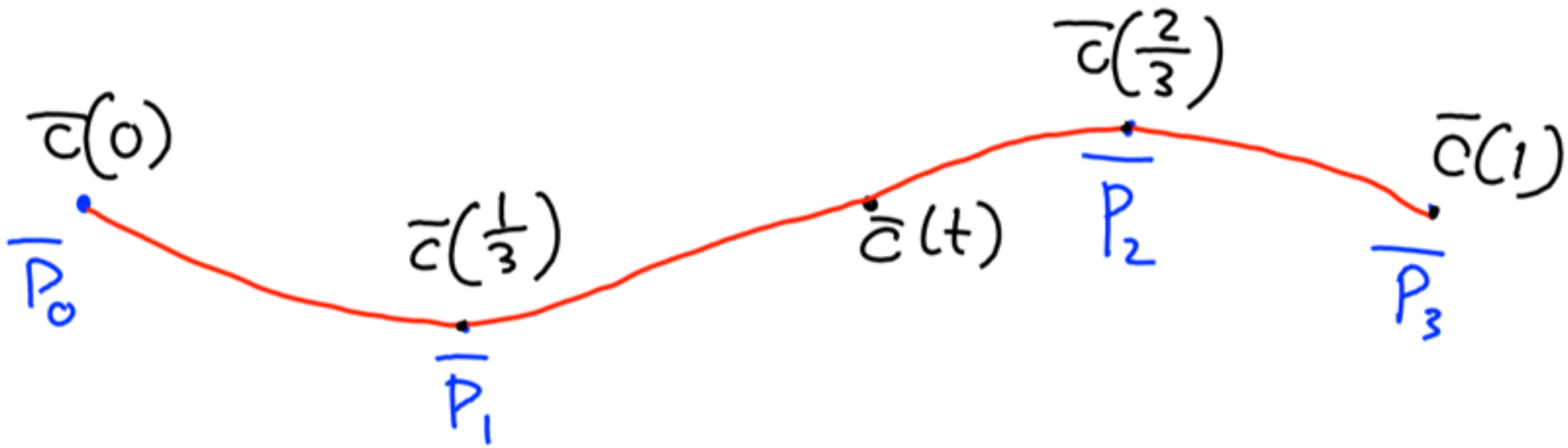
Polynomial Interpolation

- Given N control points, P_i, i = 0…n-1, t ∈ [0, 1] (by convention)
  - Define (N-1)-order polynomial x(t), y(t) such that x(i/(N-1) = x_i, y(i/(N-1) = y_i for i = 0, …, N-1
- Compute its derivatives (and tangents, normals etc)



Cubic Interpolation

- Given 4 control points, P_i, $i = (x_i, y_i)$, for $i = 0, \ldots, 3$
- Define 3rd-order polynomial $x(t)$, $y(t)$ such that $x(i/3) = x_i$, $y(i/3) = y_i$
- Compute its derivatives (and tangents, normals etc)



Cubic Interpolation: Basic Equations

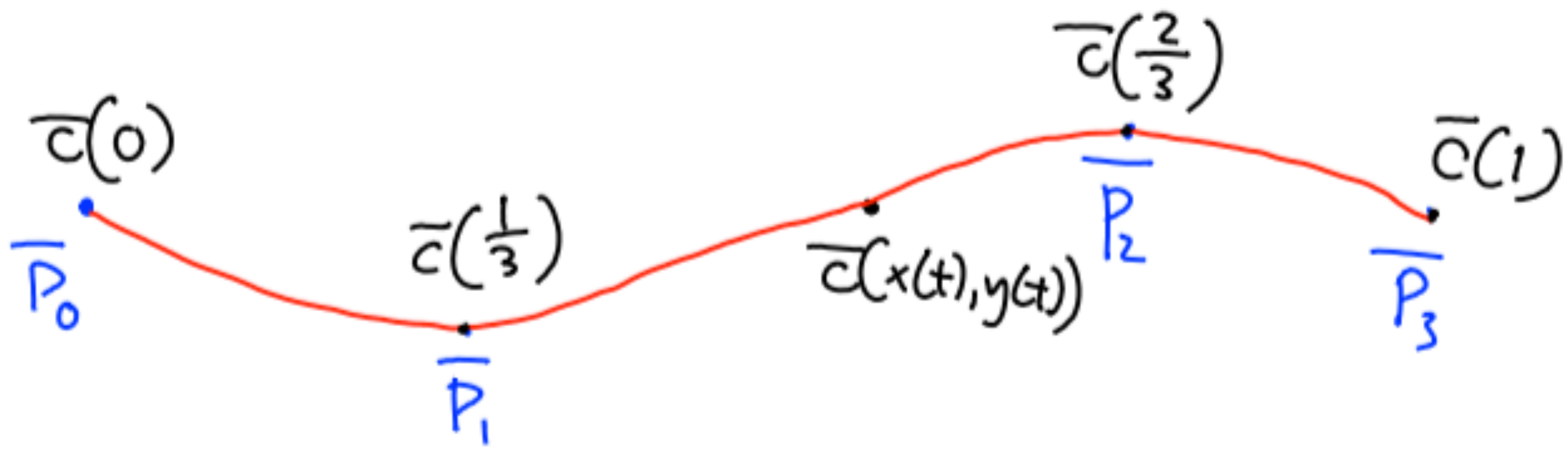
$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\} \quad \begin{aligned} &\underline{\text{given}}\ \bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4 \\ &\underline{\text{compute}}\ a_i, b_i \end{aligned}$$

Equations for one control point:

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

Equations in matrix form:

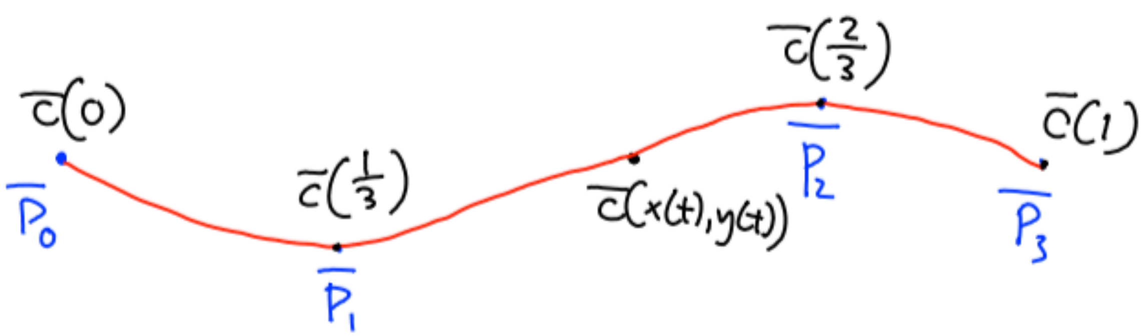
$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \tfrac{1}{3} & \left(\tfrac{1}{3}\right)^2 & \left(\tfrac{1}{3}\right)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cubic Interpolation: Computing Coeffs

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\} \quad \begin{aligned} &\underline{\text{given}}\ \bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4 \\ &\underline{\text{compute}}\ a_i, b_i \end{aligned}$$

$$\underbrace{\begin{bmatrix} x_i & y_i \end{bmatrix}}_{\text{known}} = \begin{bmatrix} 1 & \underbrace{t_i \quad (t_i)^2 \quad (t_i)^3}_{\text{known } \left(t_i = i/_{N-1}\right)} \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix} \leftarrow \underline{\text{unknown}}$$



Equations in matrix form:

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

Cubic Interpolation: Computing Coeffs

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\}$$

given $\overline{p}_1, \overline{p}_2, \overline{p}_3, \overline{p}_4$
compute a_i, b_i

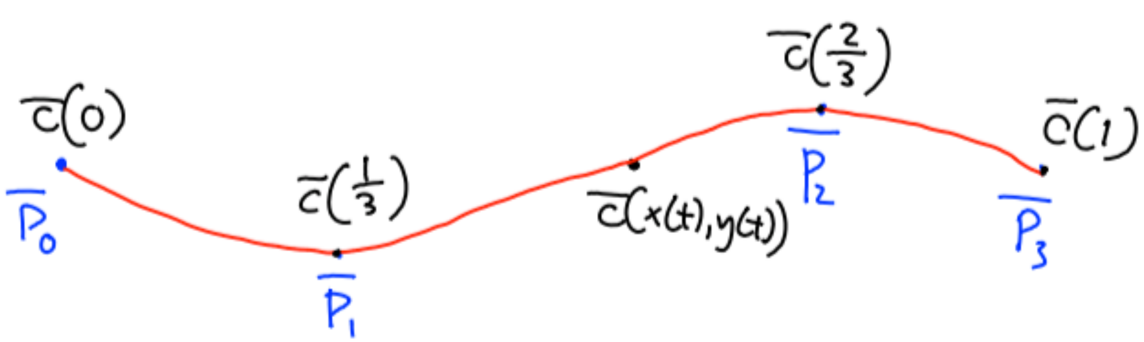
$$\underbrace{\overset{C}{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}}_{\text{known}} = \underbrace{\overset{A}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & (1/3)^2 & (1/3)^3 \\ 1 & 2/3 & (2/3)^2 & (2/3)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}}}_{\text{known}} \underbrace{\overset{X}{\begin{bmatrix}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

solve system in terms of unknown matrix

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

Equations in matrix form:

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



Cubic Interpolation: Computing Coeffs

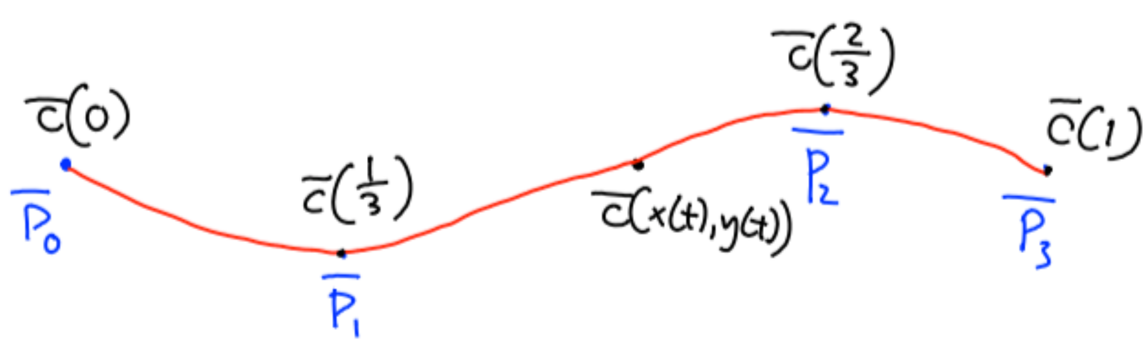
$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\}$$

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

compute a_i, b_i

Coefficients of interpolating polynomial computed by:

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



Equations in matrix form:

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

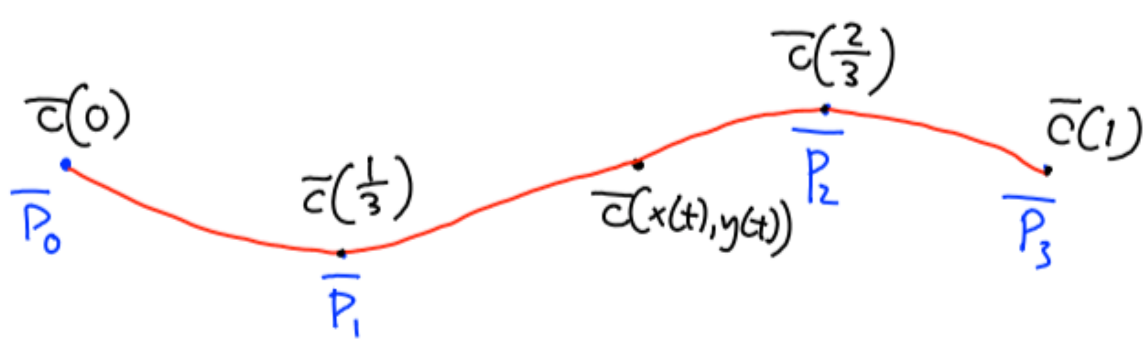
Cubic Interpolation: Evaluating the Polynomial

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\}$$

given $\overline{P_1}, \overline{P_2}, \overline{P_3}, \overline{P_4}$

compute a_i, b_i

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



Equations in matrix form:

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

Cubic Interpolation: What if < 4 Control Points?

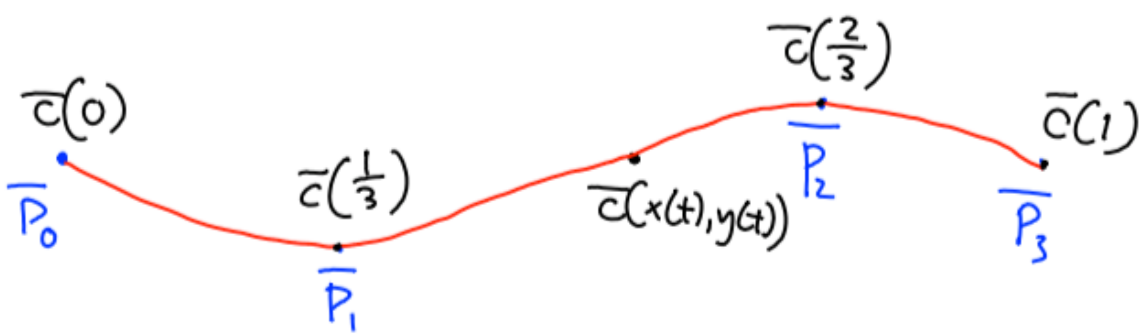
$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

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

degree +1

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

← # control points →



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: What if > 4 Control Points?

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\}$$

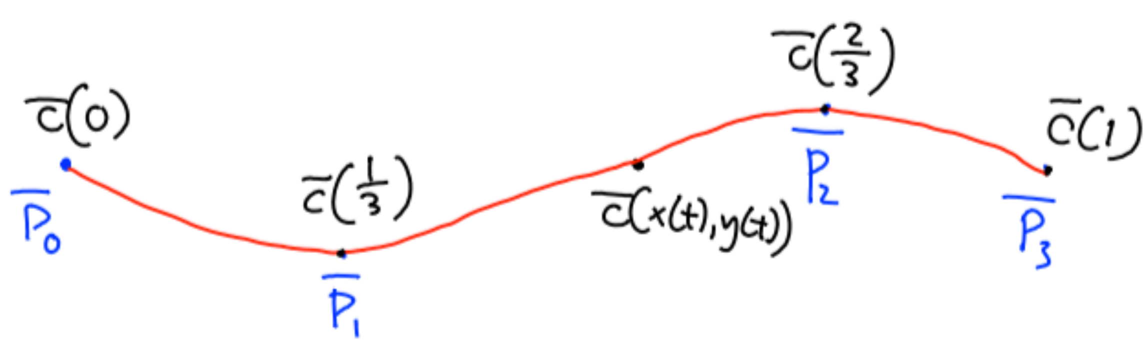
given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

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

degree +1

← #control points. →



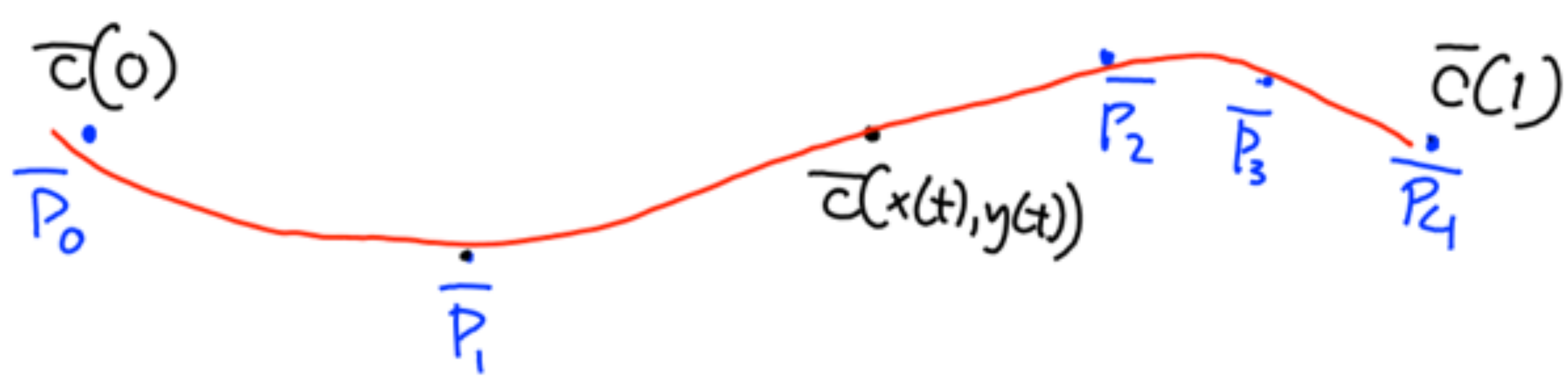
Equations in matrix form:

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

Exact Interpolation of N points

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

$$\text{degree}+1 \; \updownarrow \; \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \underbrace{\begin{bmatrix} \text{NxN matrix} \\ \text{\# constraints} = \\ \text{\# unknown coeffs} \end{bmatrix}^{-1}}_{\longleftarrow \text{\# control points.} \longrightarrow} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

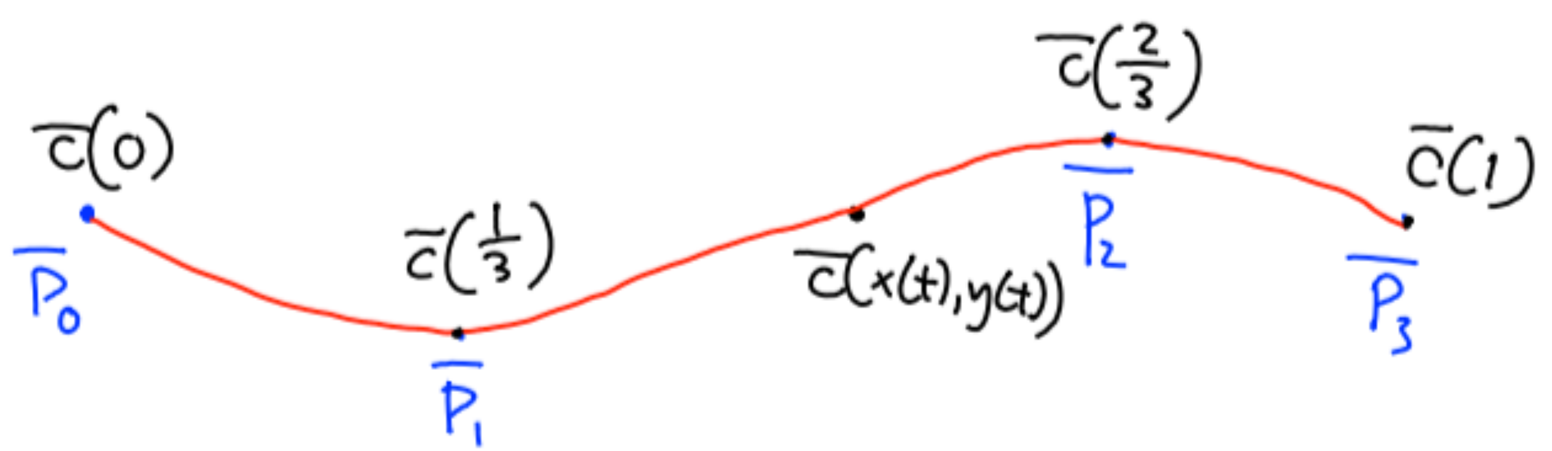


Cubic Interpolation: Evaluating Derivatives

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

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

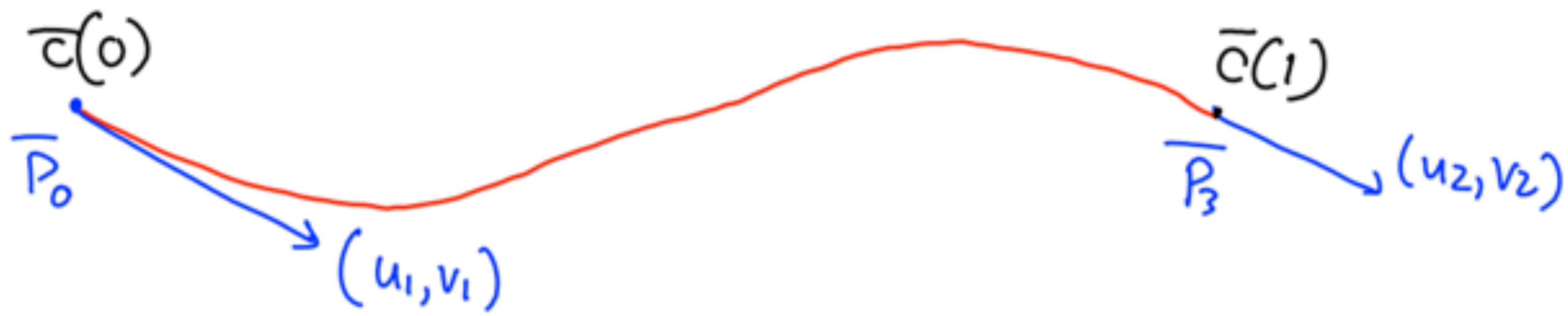
$$\boxed{\begin{bmatrix} \frac{dx}{dt}(t) & \frac{dy}{dt}(t) \end{bmatrix} = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}}$$



Specifying the Poly via Tangent Constraints

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

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

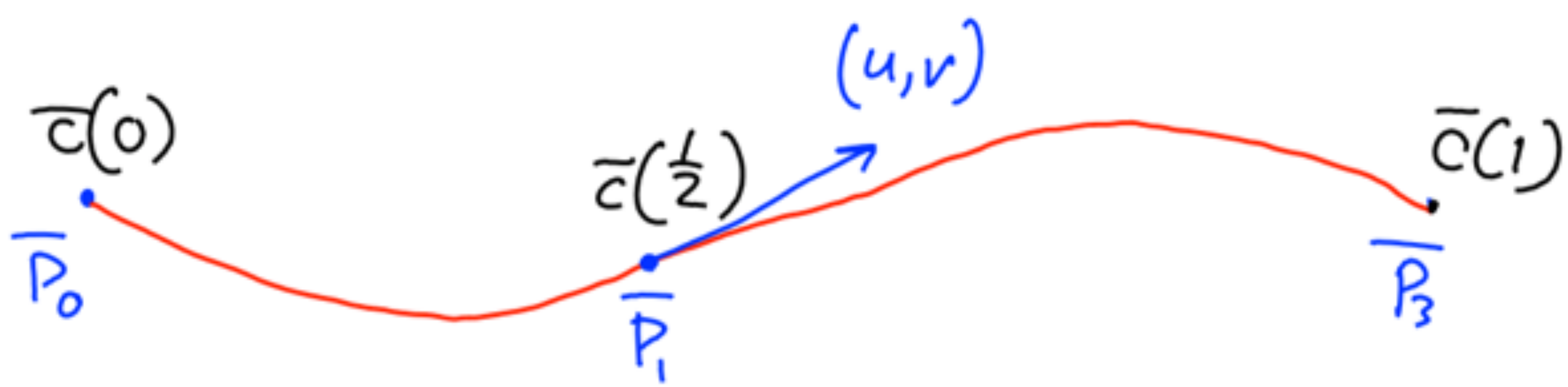


Specifying the Poly via Tangent Constraints

- Instead of specifying 4 control points, we could specify 3 points and a derivative.
- Replace the 4th pair of equations with

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

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

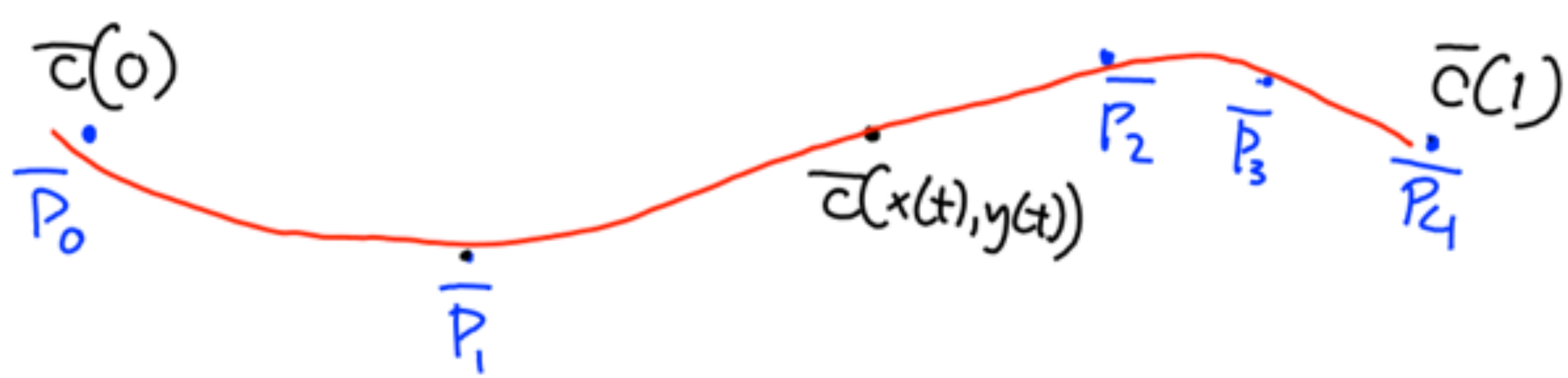


Degree-N Poly Interpolation: Major Drawback

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 and undesirable ways

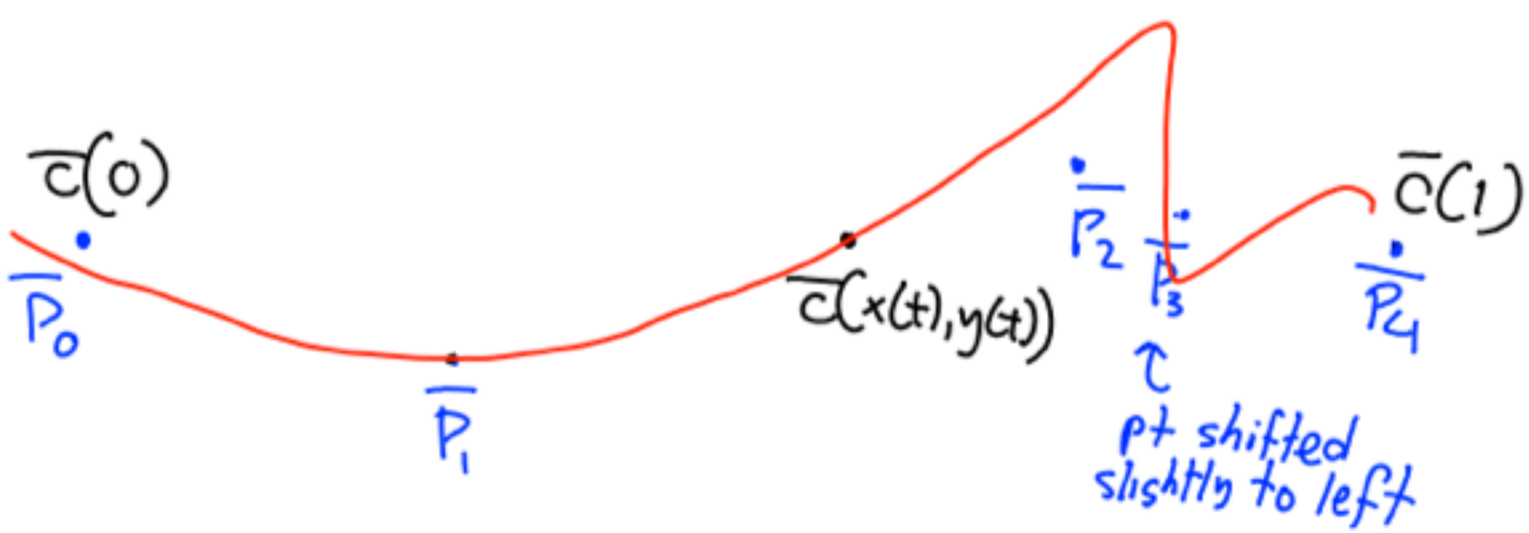


Degree-N Poly Interpolation: Major Drawback

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 and undesirable ways



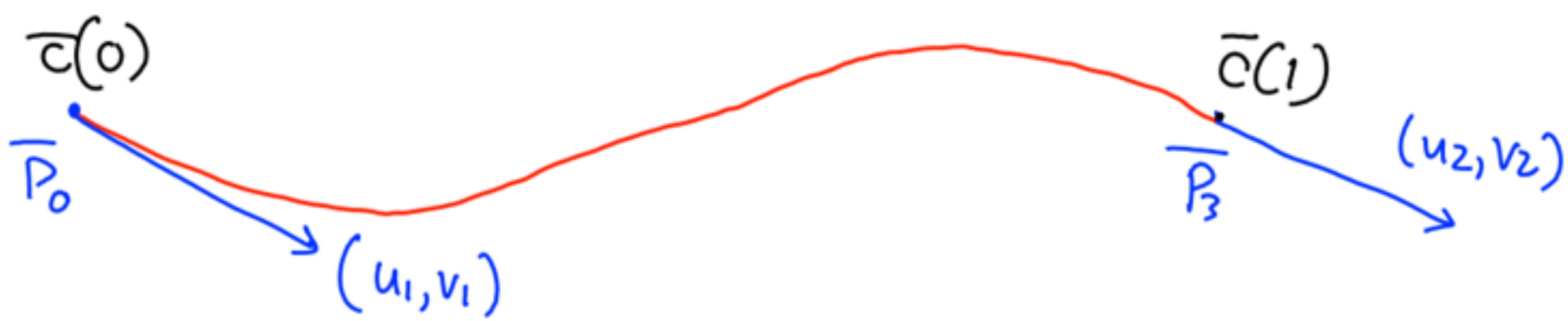
Topic 12: Interpolating Curves

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

Bézier Curves

Properties:

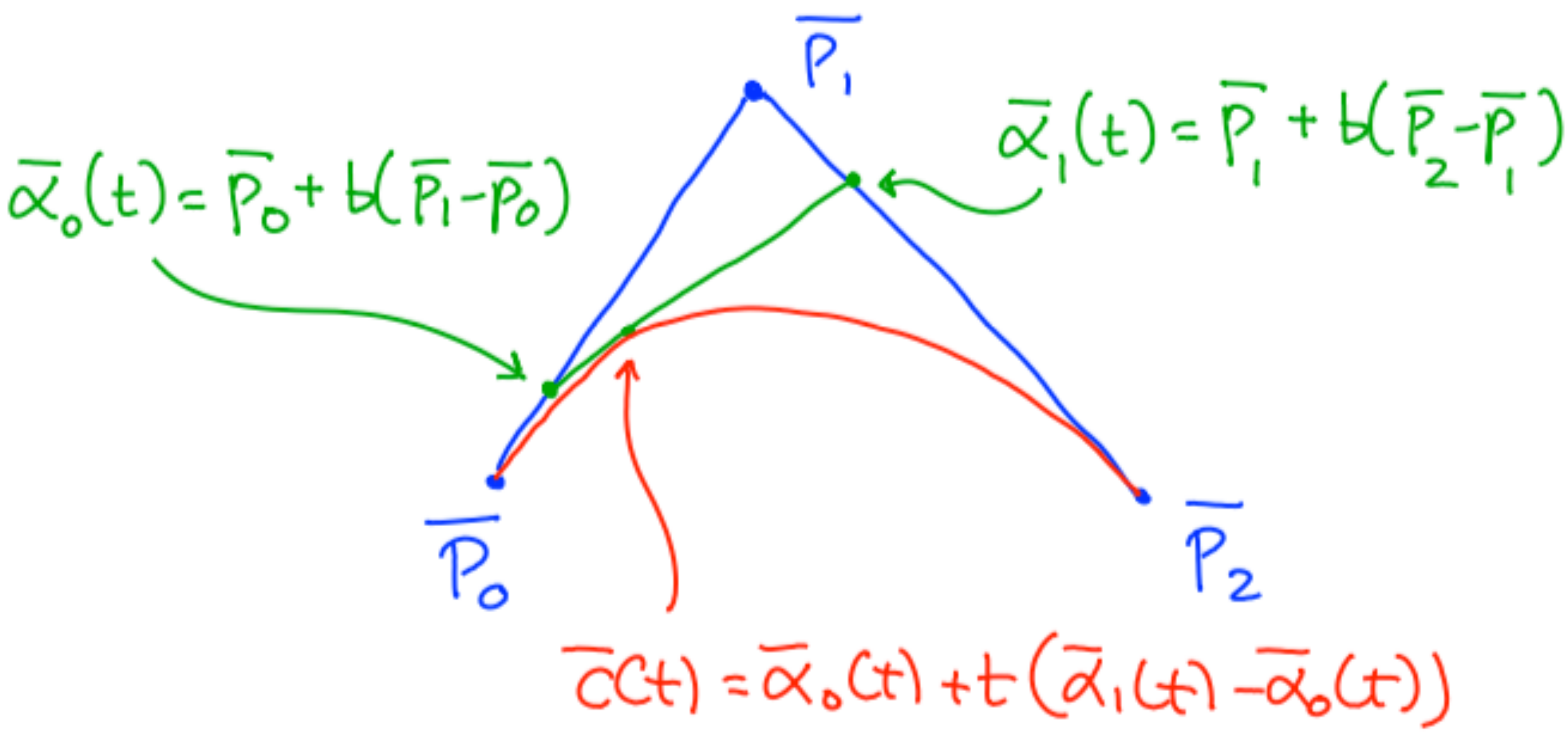
- Polynomial curves defined via endpoints and derivative constraints
- Derivative constraints defined <u>implicitly</u> through extra control points (that are not interpolated)
- They are <u>approximating</u> curves, not interpolating curves



Bézier Curves: Main Idea

Polynomial and its derivatives expressed as a <u>cascade of linear interpolations</u>

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 $\bar{c}(t)$

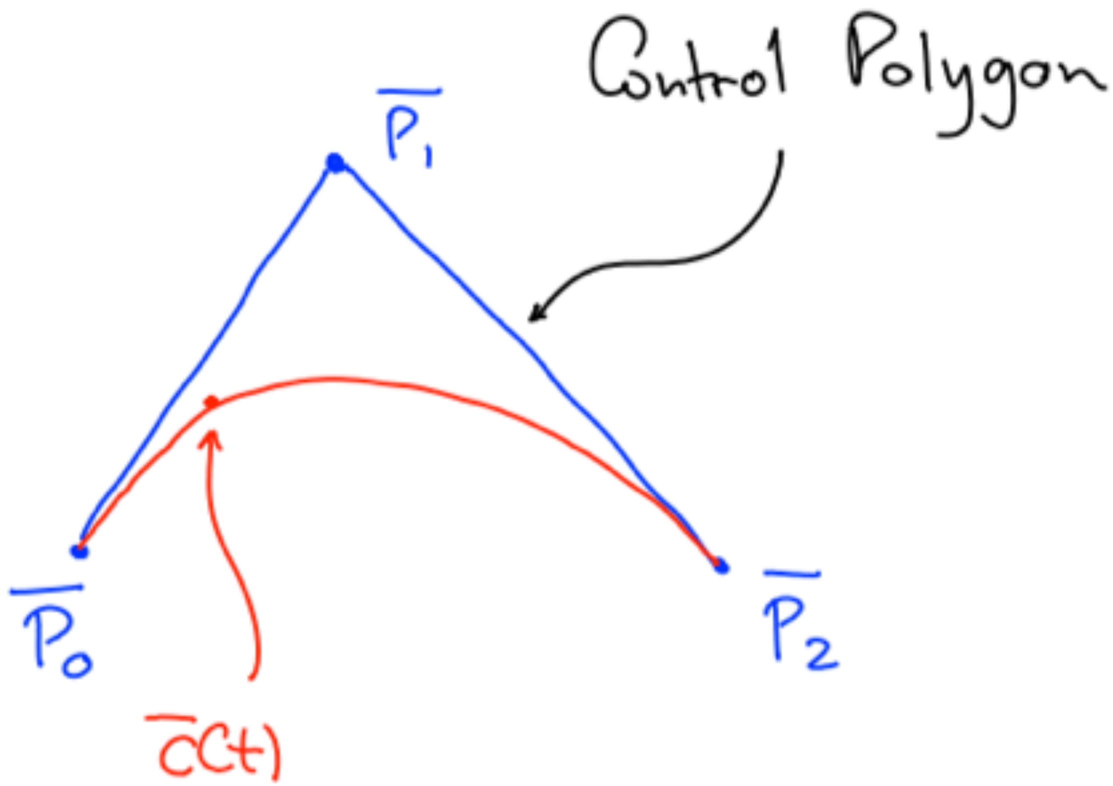
Q: Where have we seen such a cascade before?

Bézier Curves: Control Polygon

A Bézier curve is completely determined by its control polygon

We manipulate the curve by manipulating its polygon

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. lineary 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 $\bar{c}(t)$

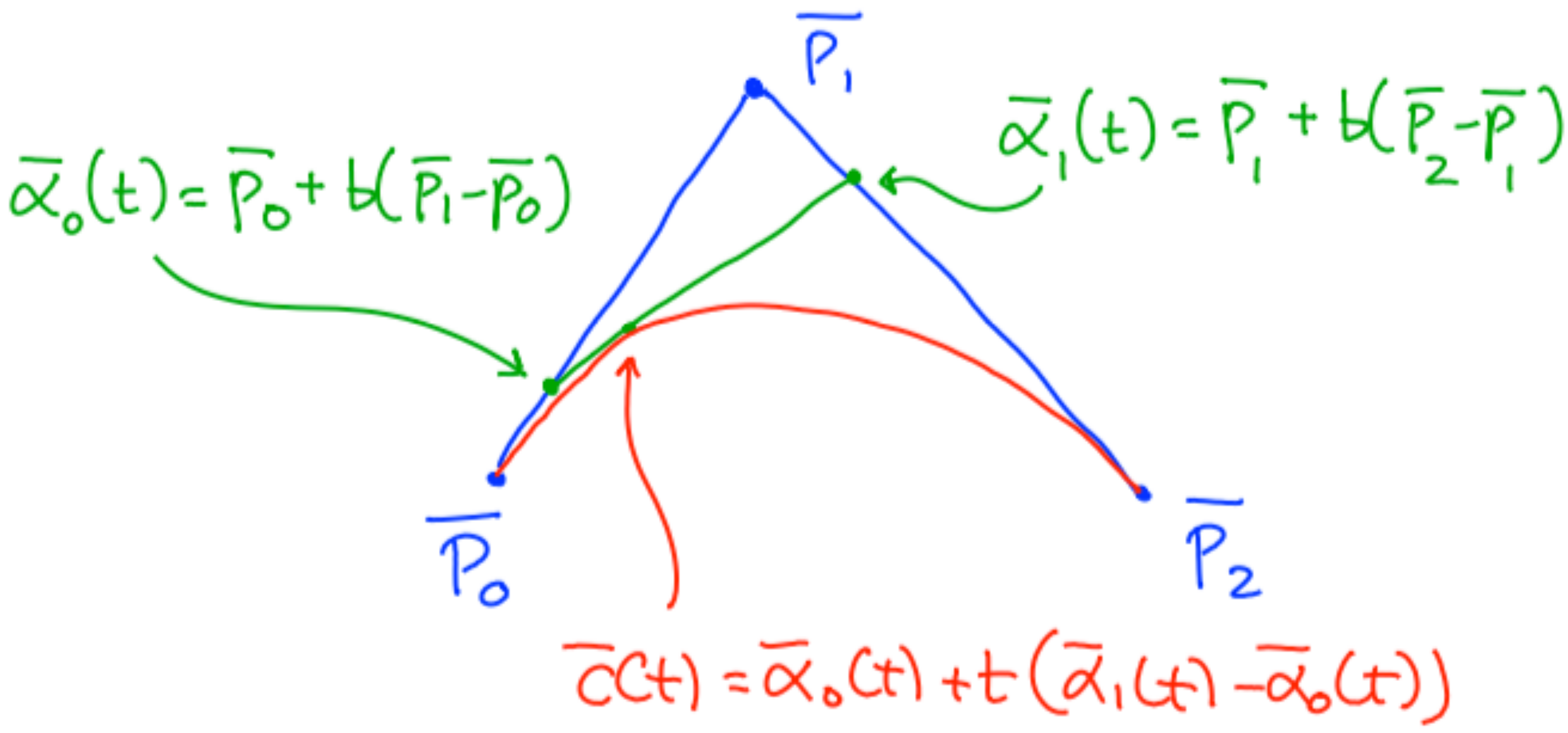
Expressing the Bézier Curve as a Polynomial

Computing the polynomial

$$\bar{c}(t) = \left[\bar{P}_0 + t(\bar{P}_1 - \bar{P}_0)\right] + t\left[\bar{P}_1 + t(\bar{P}_2 - \bar{P}_1) - \bar{P}_0 - t(\bar{P}_1 - \bar{P}_0)\right]$$

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

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



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. lineary 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:

$$\frac{d}{dt}c(t) = -2(1-t)P_0 + 2P_1(1-2t) + P_2\,2t$$

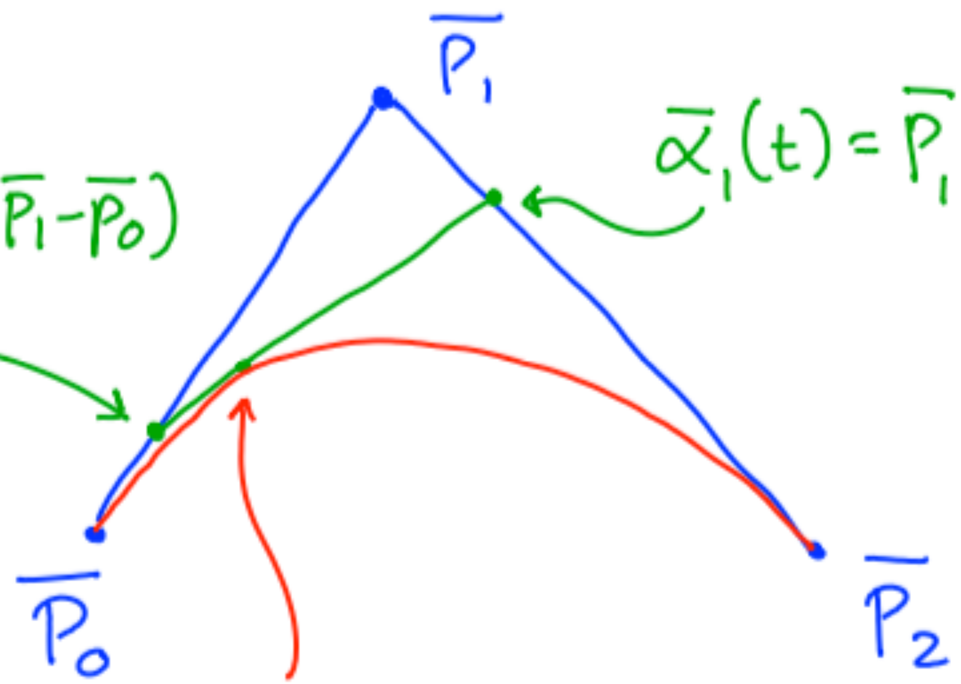
$= 2(P_1 - P_0)$ at $t=0$

$= 2(P_2 - P_1)$ at $t=1$

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

$\bar{\alpha}_0(t) = \bar{P}_0 + t(\bar{P}_1 - \bar{P}_0)$

$\bar{\alpha}_1(t) = \bar{P}_1 + t(\bar{P}_2 - \bar{P}_1)$



$\bar{c}(t) = \bar{\alpha}_0(t) + t(\bar{\alpha}_1(t) - \bar{\alpha}_0(t))$

algorithm:

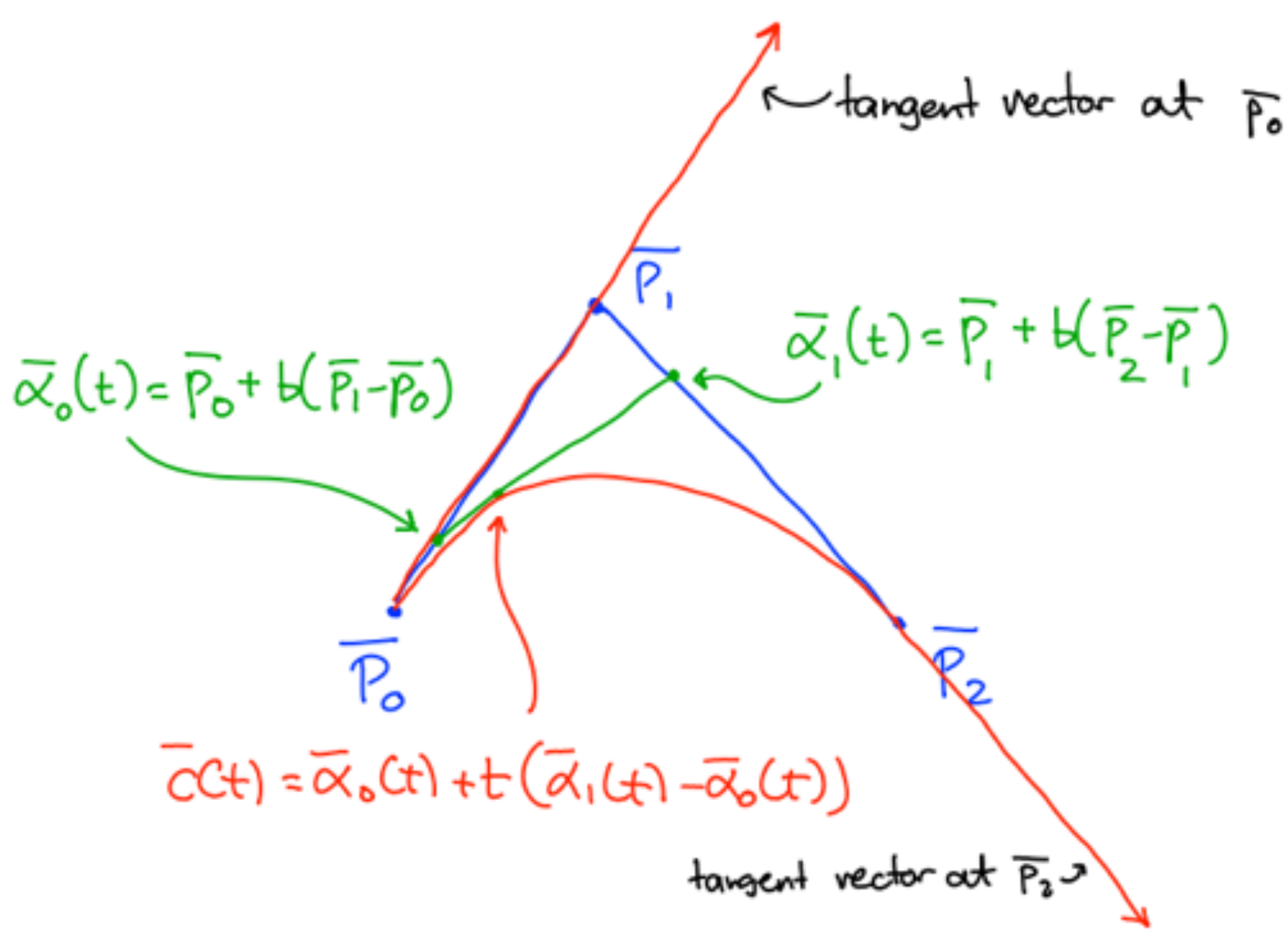
given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t

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: Endpoints and Tangent Constraints

Computing the polynomial's derivatives:

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



General Behaviour

- 1st and 3rd control points define the endpoints.
- 2nd control point defines the tangent vector at the endpoints.

Bézier Curves: Generalization to N+1 points

Expression in compact form:

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

curve

control pt

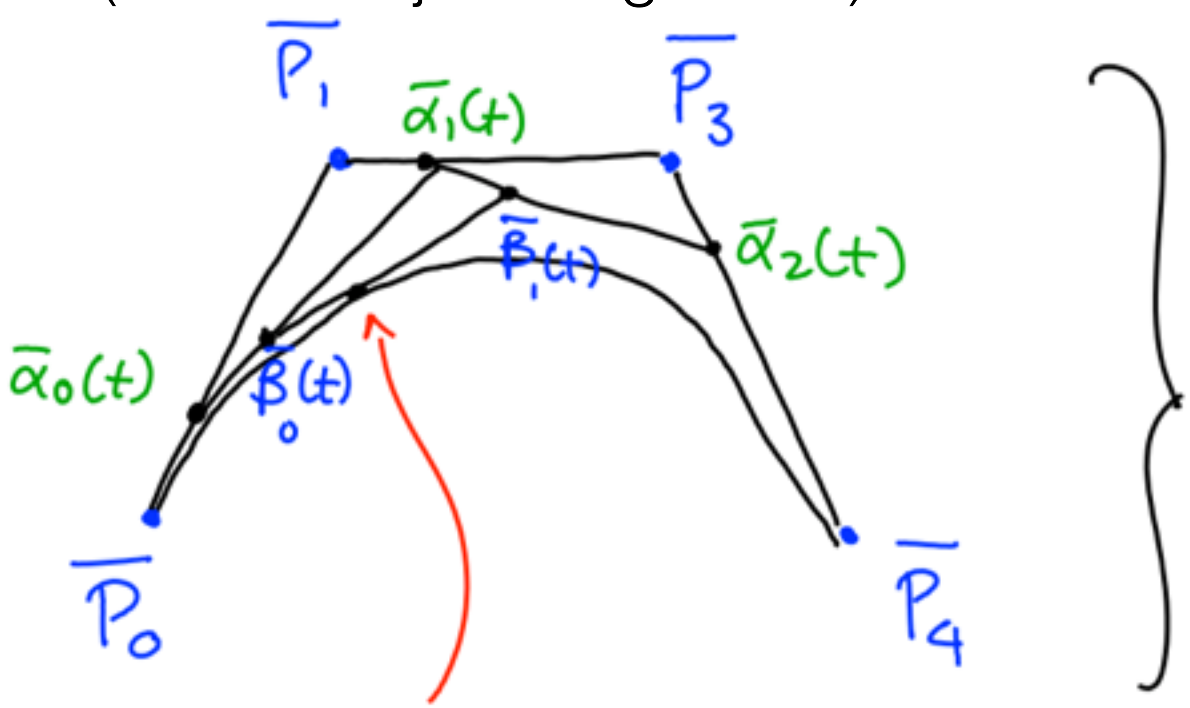
Where:

called the Bernstein Polynomials of degree N

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

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

Curve defined by N linear interpolation cascades (De Casteljau's algorithm):



Example for 4 control 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:

$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i \, B_i^N(t)$$

curve

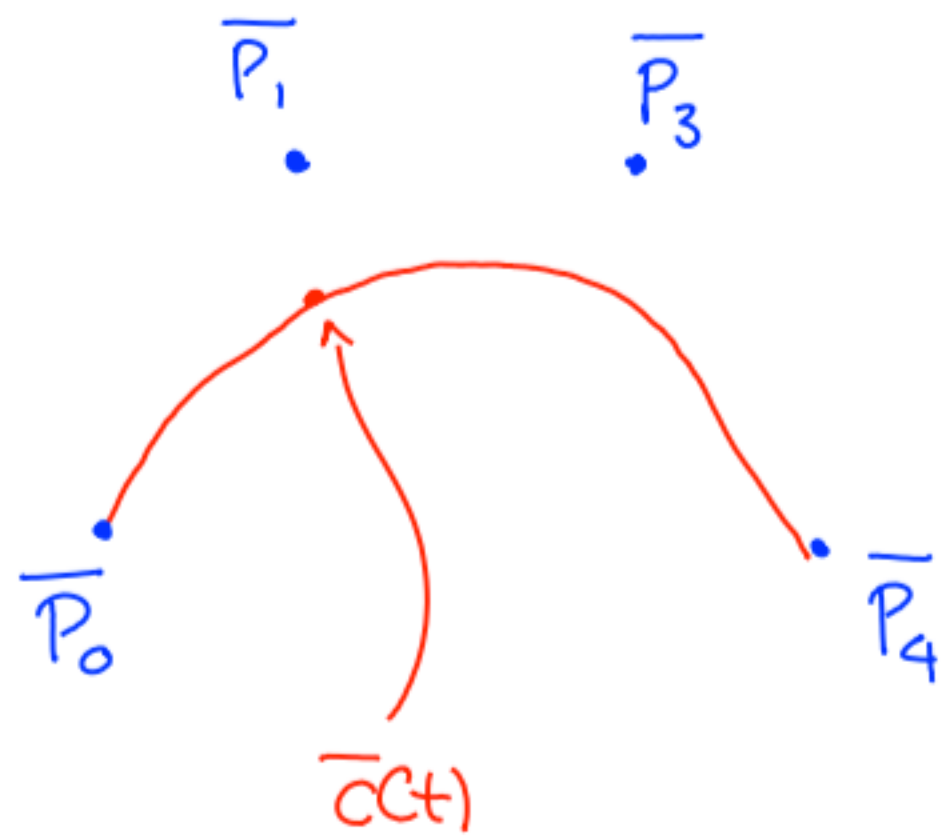
control pt

Where:

called the Bernstein Polynomials of degree N

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

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



- Each curve point c(t) is a “blend” of the 4 control points.
- The blend coefficients depend on t
- They are Bernstein polynomials

Bézier Curves as “blends” of the Control Points

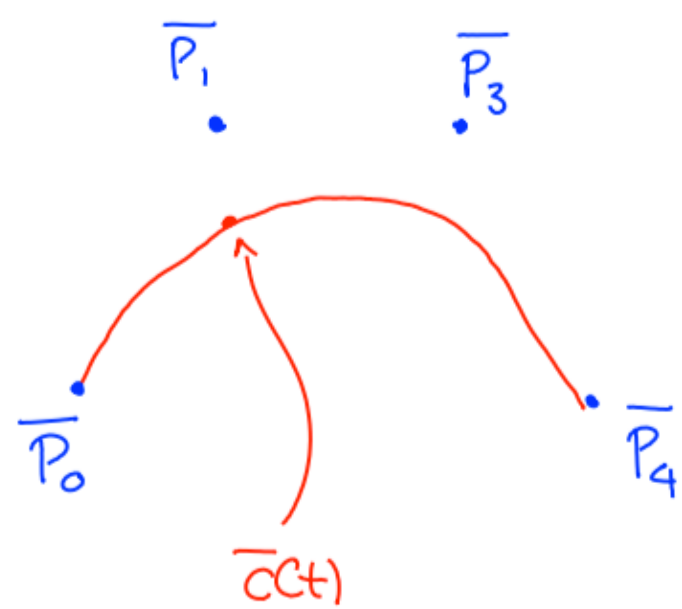
Expression in compact form:

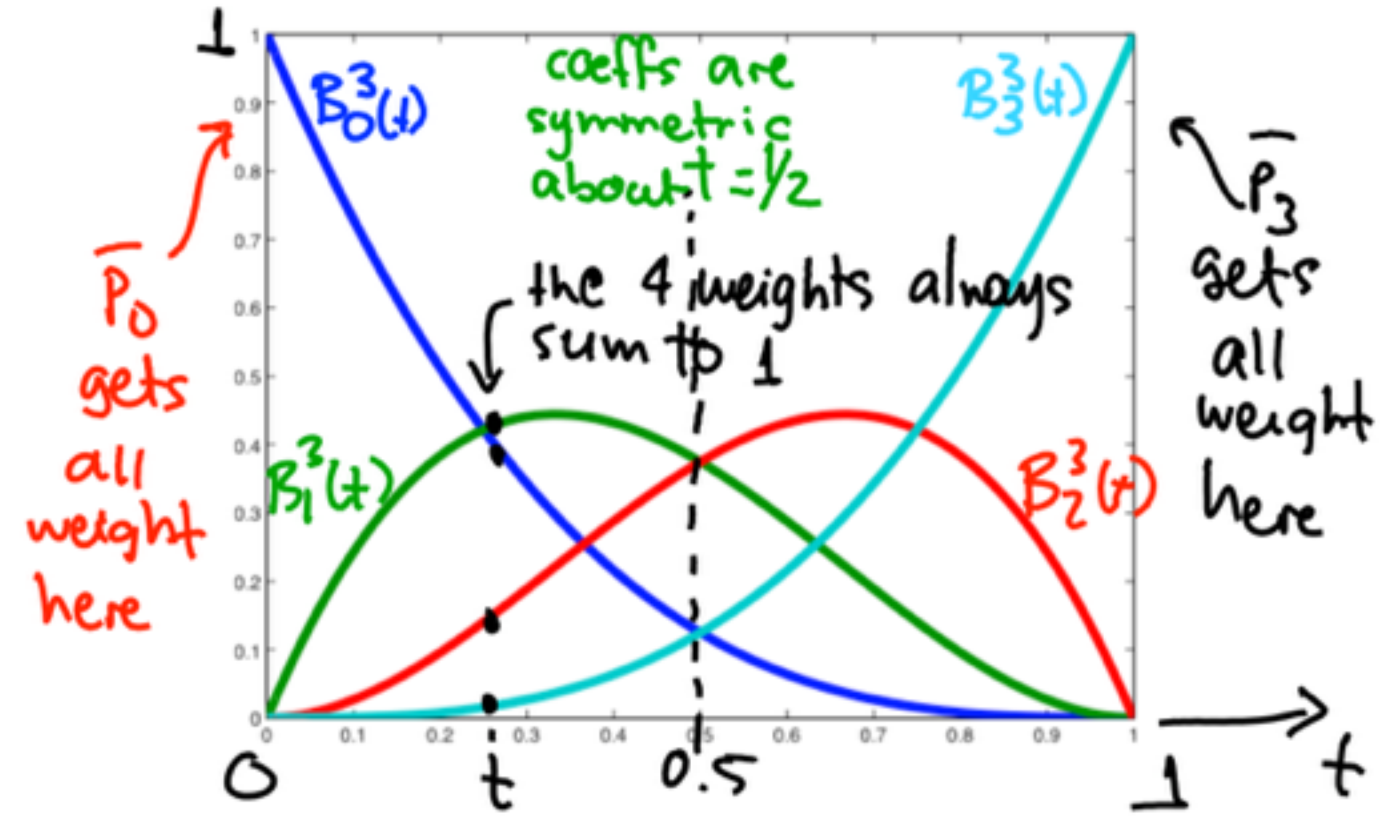
$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i \, B_i^N(t)$$

curve

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 points.
- The blend coefficients depend on t
- They are Bernstein polynomials

Bézier Curves: Useful Properties

Expression in compact form:

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

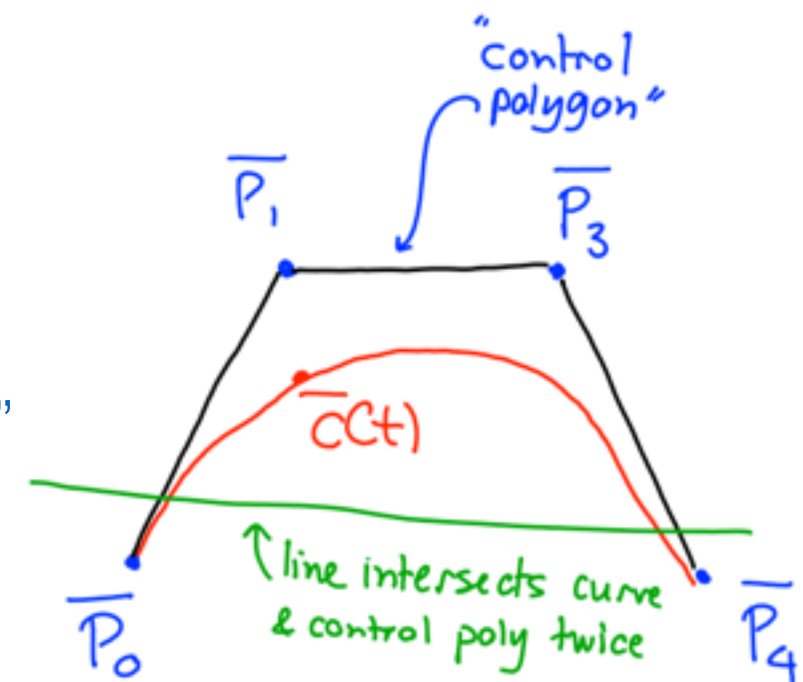
Where:

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 an affine transform T is equivalent to transforming its control points by T
2. Diminishing Variation
   - No line will intersect the curve at more points than the control polygon
     - 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:

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

Where:

called the Bernstein Polynomials of degree N:

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

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

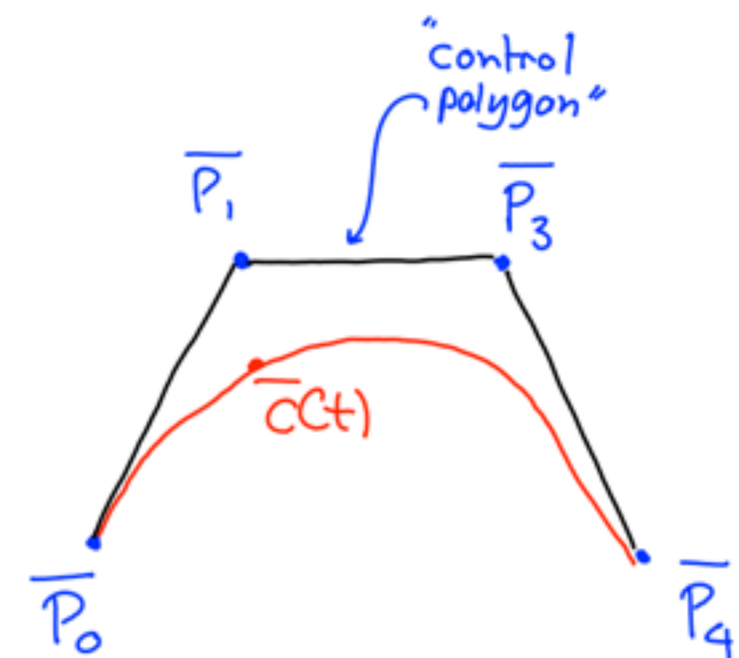
4. Tangents at endpoints are along the 1st and last edges of control polygon:

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

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

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

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



Bézier Curves: Pros and Cons

Expression in compact form:

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

Where:

called the Bernstein Polynomials of degree N:

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

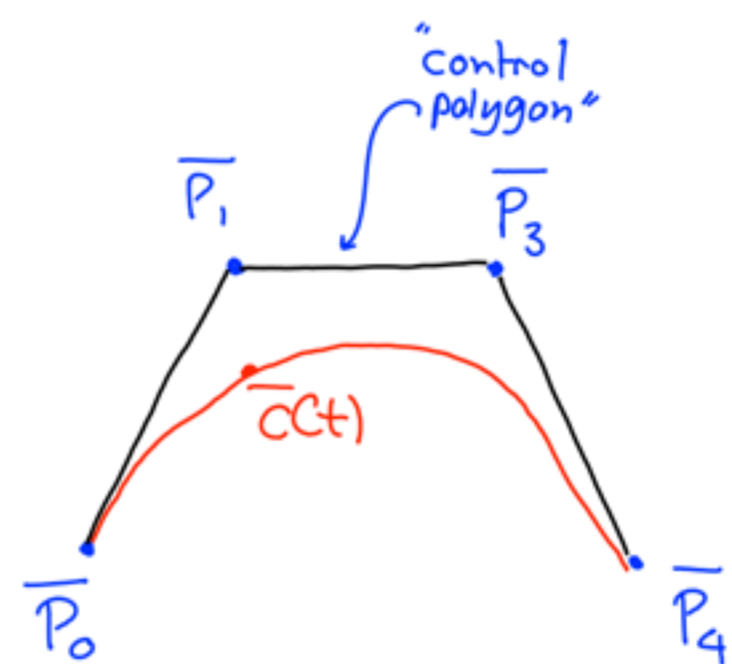
$$= \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

Advantages:

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

Disadvantages:

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

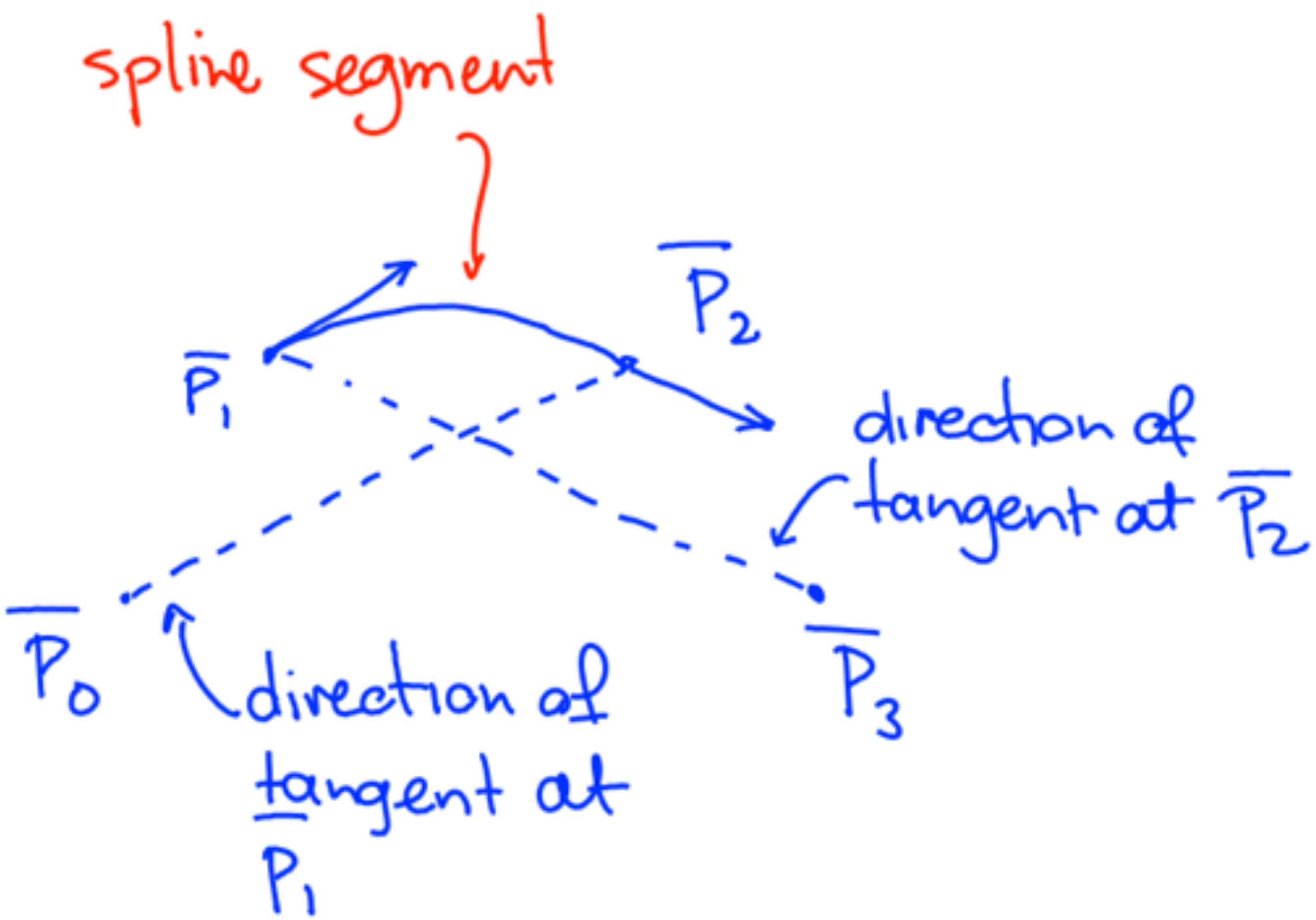


Topic 12: Interpolating Curves

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

Cubic Cardinal Splines: Defining 1st Segment

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}



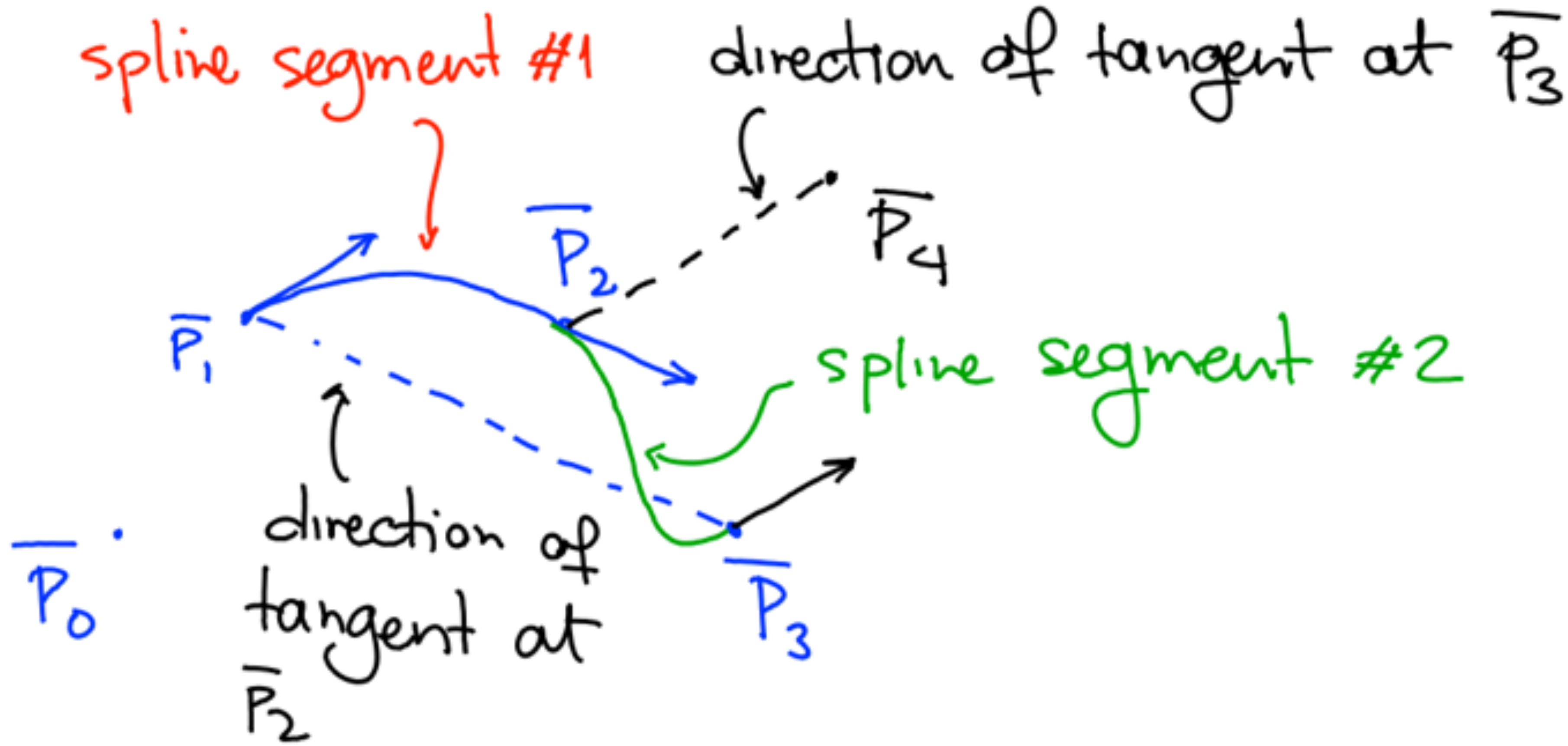
$\bar{P}_1, \bar{P}_2$ act as endpoint constraints

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

Cubic Cardinal Splines: Defining 2nd Segment

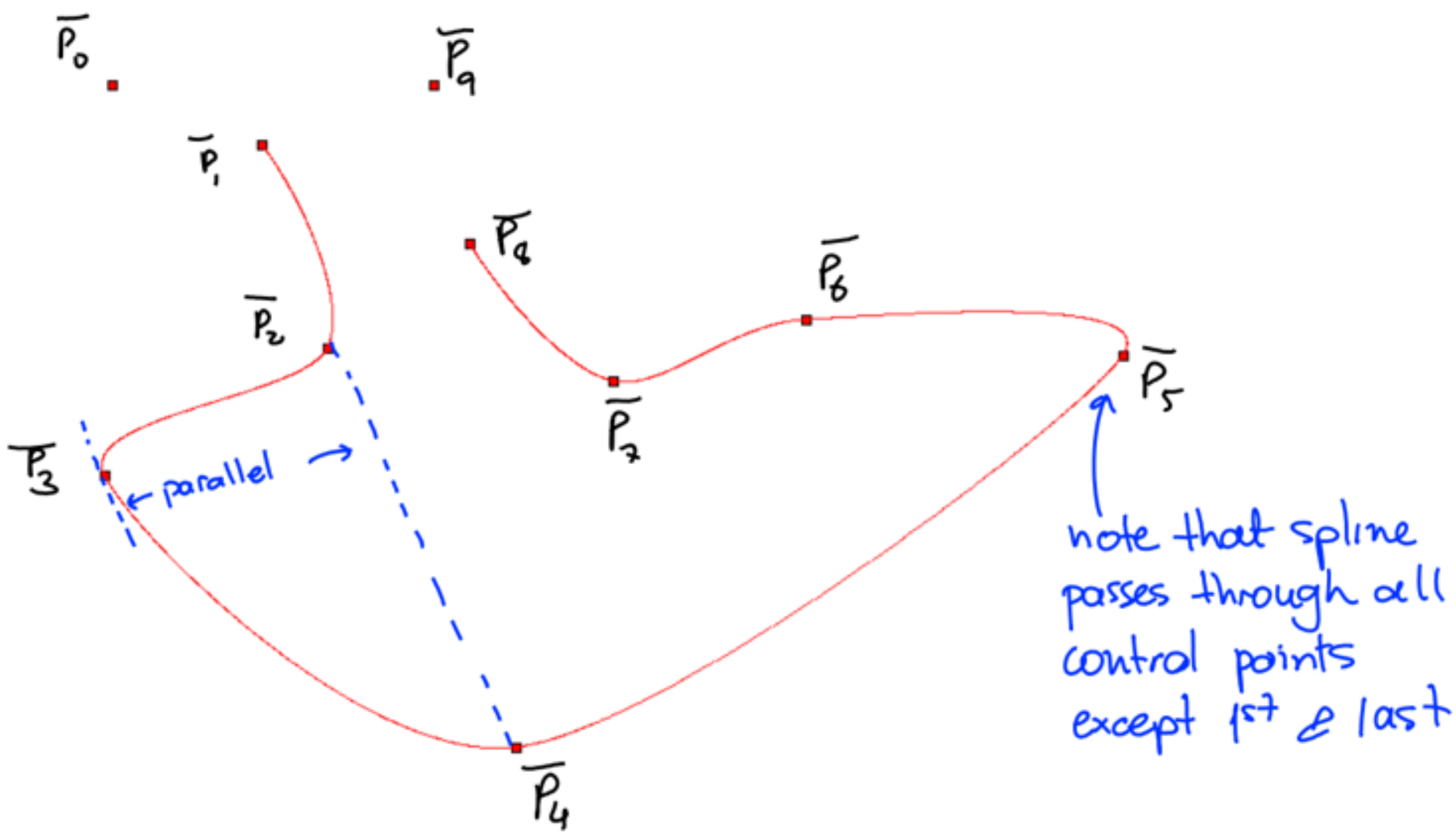
- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

Example: Adding a fifth point adds a new segment



Cubic Cardinal Splines: General Case

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

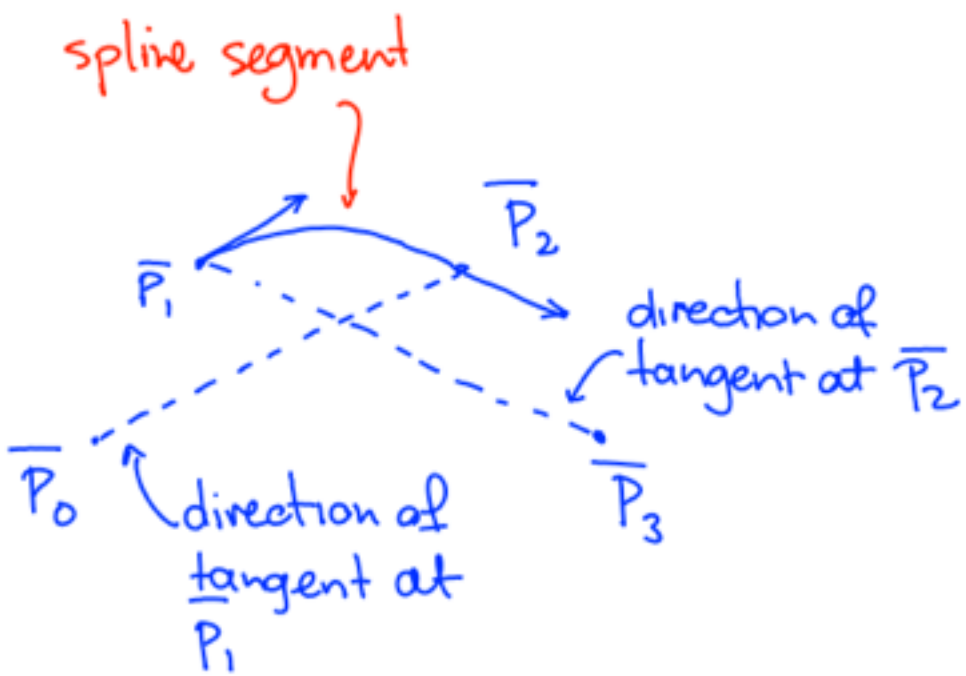


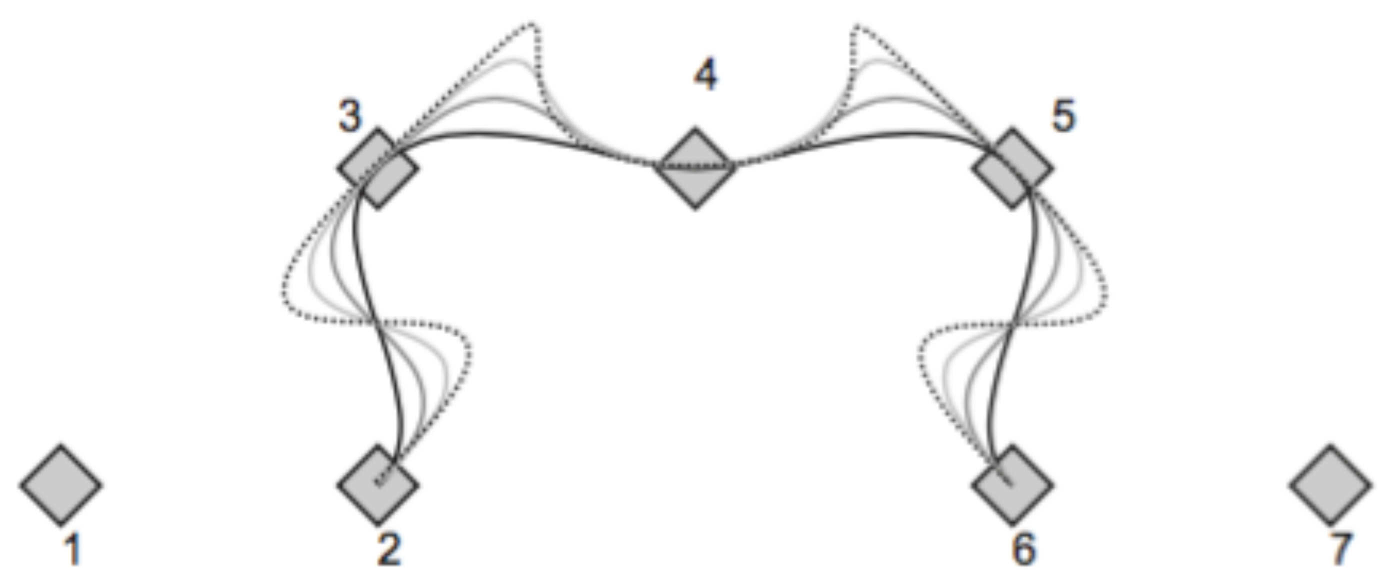
Cubic Cardinal Splines: The Strain Parameter

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

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

called a strain parameter





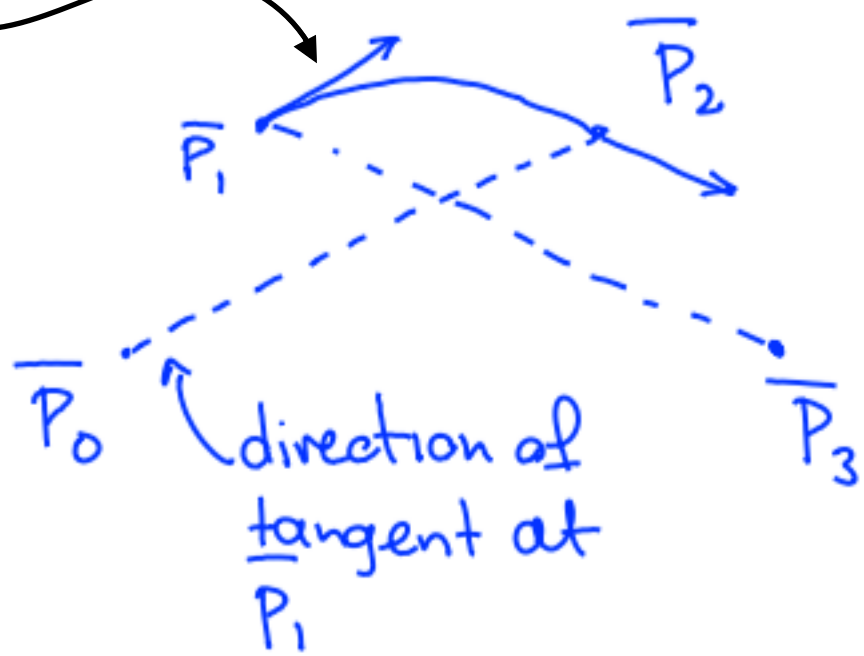
Catmull-Rom Splines

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

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

called a strain parameter

length of tangent = 1/2 distance between p_0 and p_2

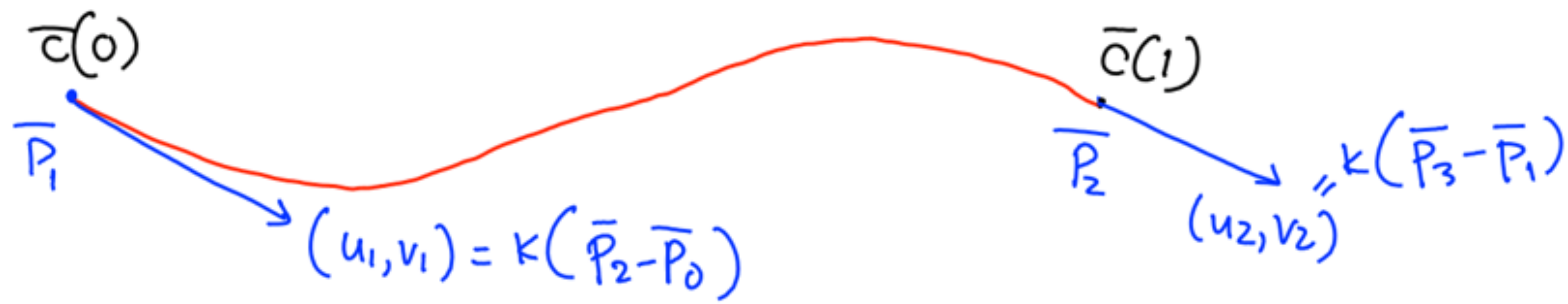


Note: If k = 0.5, the spline is called a Catmull-Rom Spline

Specifying the Poly via Tangent Constraints

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

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



Cardinal Splines: Solving for the Segment Coeffs

for t = 0

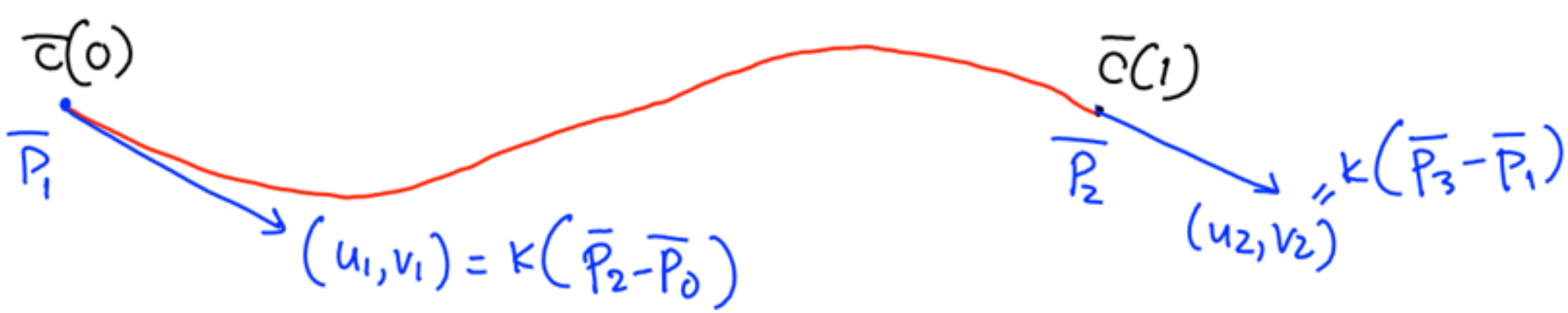
$$[x_1 \;\; y_1] = [1 \;\; \underset{0}{\cancel{t}} \;\; \underset{0}{\cancel{t^2}} \;\; \underset{0}{\cancel{t^3}}] \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

endpoint constraint

$$[k(x_2 - x_0) \;\; k(y_2 - y_0)] = [1 \;\; \underset{0}{\cancel{2t}} \;\; \underset{0}{\cancel{3t^2}}] \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

derivative constraint

+ 2 more equations for other endpoint (t = 1)

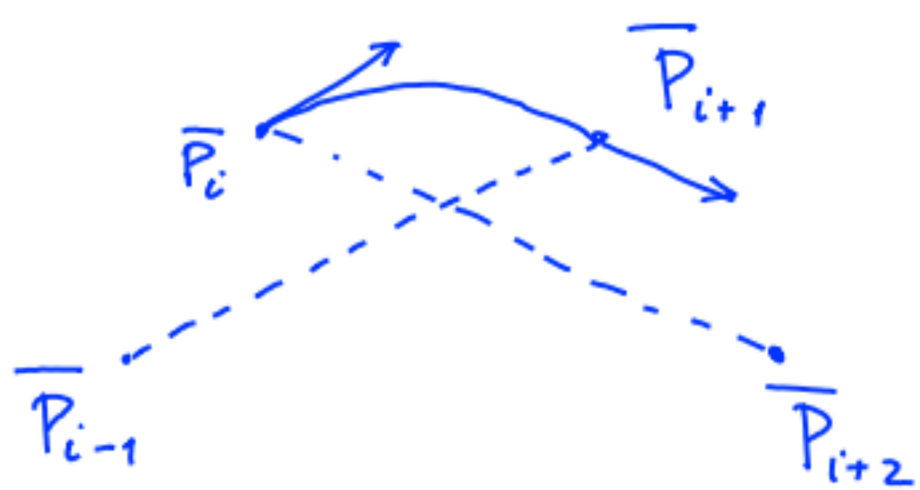


Cubic Cardinal Spline Segment vs Bézier Curve

The two curves are actually <u>equivalent</u>:

given a cardinal spline, we can compute the control polygon of the equivalent Bézier curve

Cardinal Spline Segment

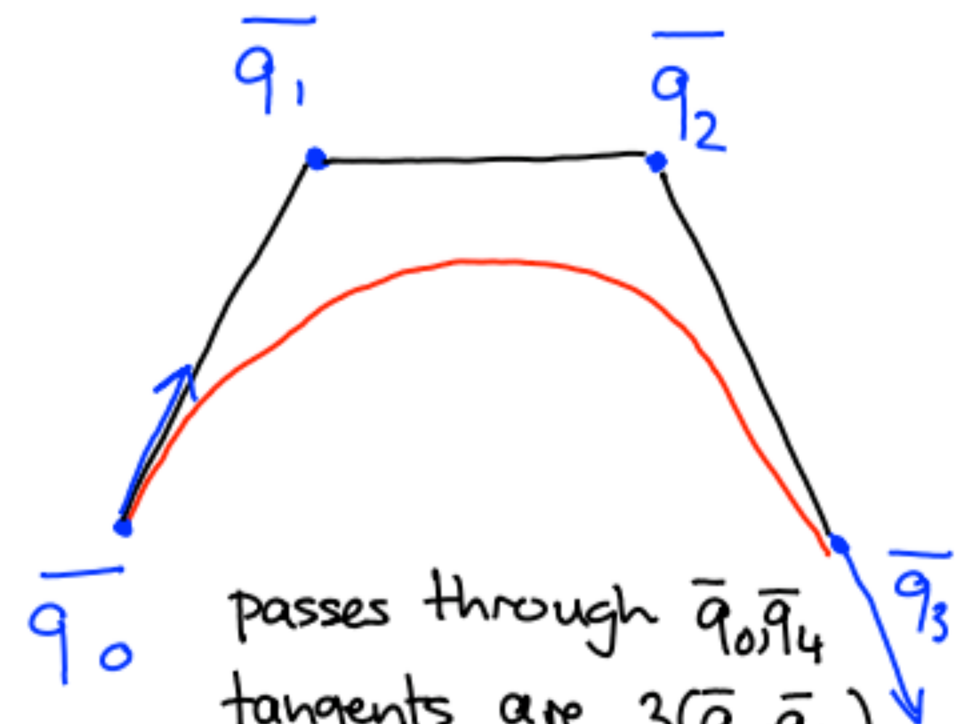


passes through $\bar{p}_i, \bar{p}_{i+1}$

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

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

Bézier Curve



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

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

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

Cubic Cardinal Spline Segment vs Bézier Curve

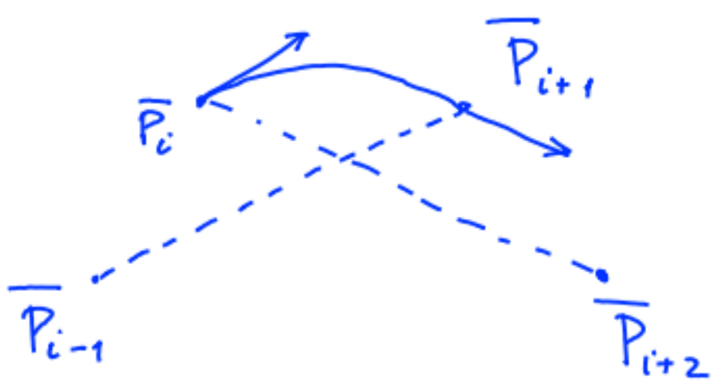
In order to have c(t) = r(t) for all t, it must be:

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

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

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

Cardinal Spline Segment

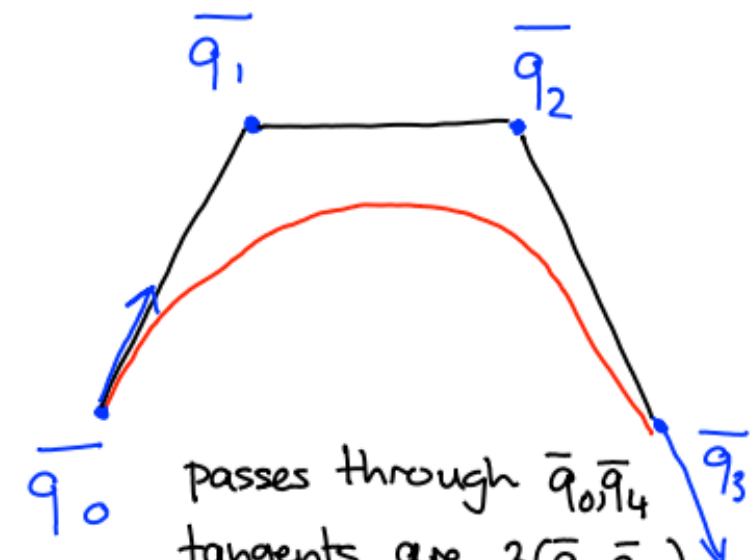


passes through $\bar{P}_i, \bar{P}_{i+1}$

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

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

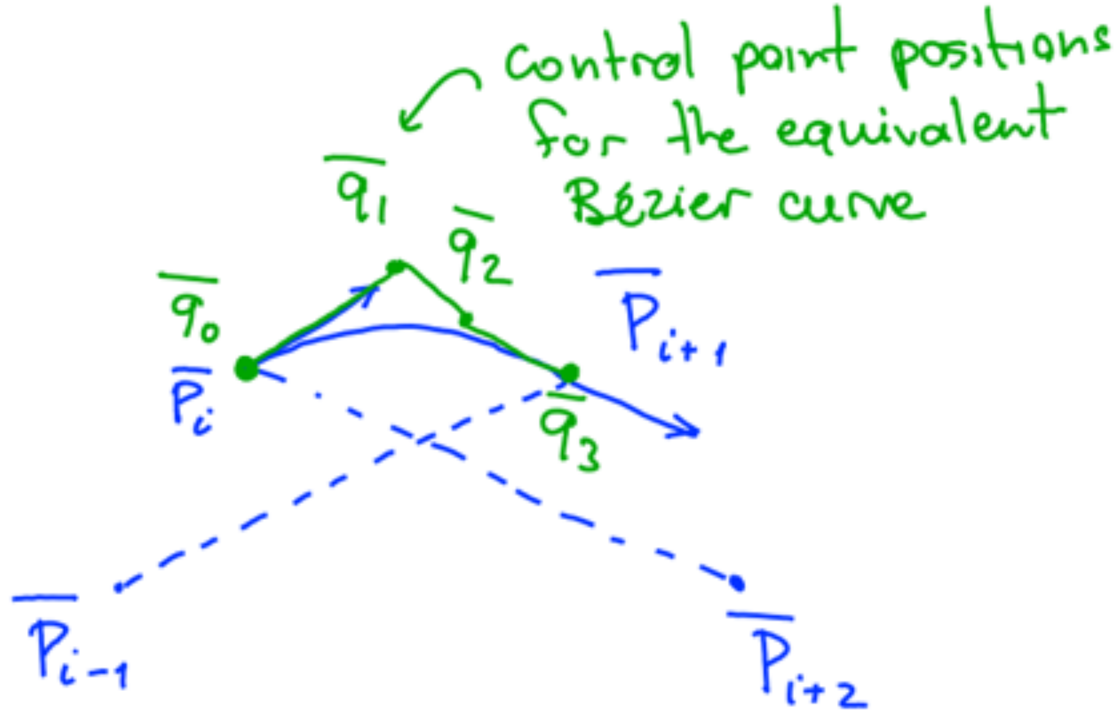
Cubic Cardinal Spline Segment vs Bézier Curve

In order to have c(t) = 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