Splines (1)

Xin (Shane) Li
Oct. 22, 2009

Piecewise linear approximation

- Previous: polygonal representation (meshes) and polylines are first-degree, piecewise linear approximations to surfaces and curves
- When the object is not piecewise linear
  - To improve its approximation accuracy
    → more sample points
      → large number of coordinates to be created and stored
- Interactive manipulation is tedious
- Need a more compact and more manipulable representation
  - To use functions that are of a higher degree

Three general approaches

1) Explicit functions:

→ y=f(x), z=g(x)

- Can't get multiple values of y for a single x → closed curves must be represented by multiple segments
- Not rotationally invariant
- Curves with vertical tangents is difficult (infinite slop)

2) Implicit functions:

→ f(x,y,z)=0

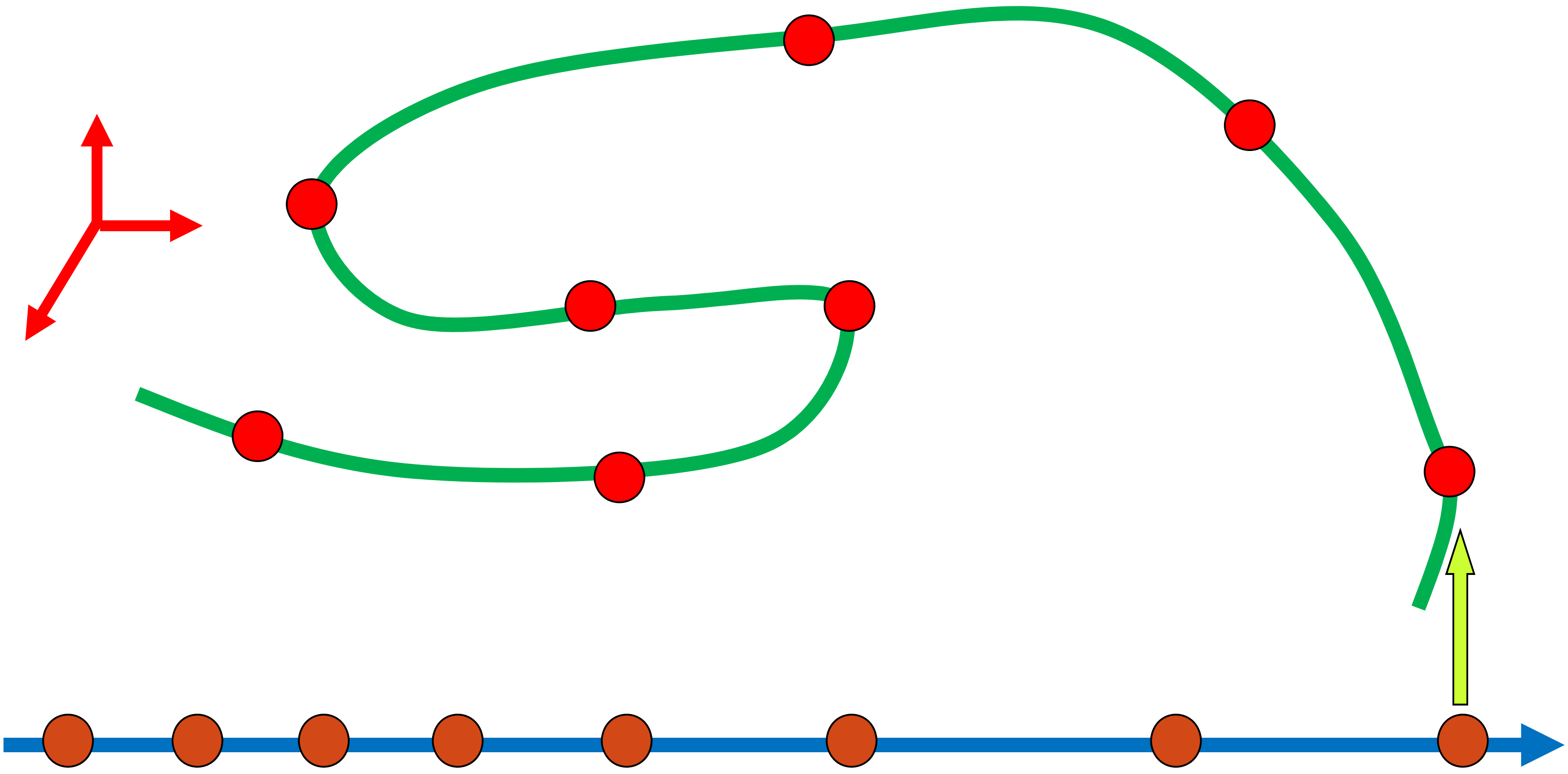
- A simple equation is usually not enough, need several for constraints
  - e.g. : a half circle
- Not easy to merge several simple sub-parts
  - e.g. : when merge two curve segments, difficult to determine whether their tangent directions agree

3) Parametric representation:

→ x=x(t), y=y(t), z=z(t)

- Overcome above problems
- geometric slopes (may be infinite) → parametric tangent vectors (never infinite)
- Piecewise linear shapes → piecewise polynomial shapes

Parametric Curve



Parametric Cubic Curves

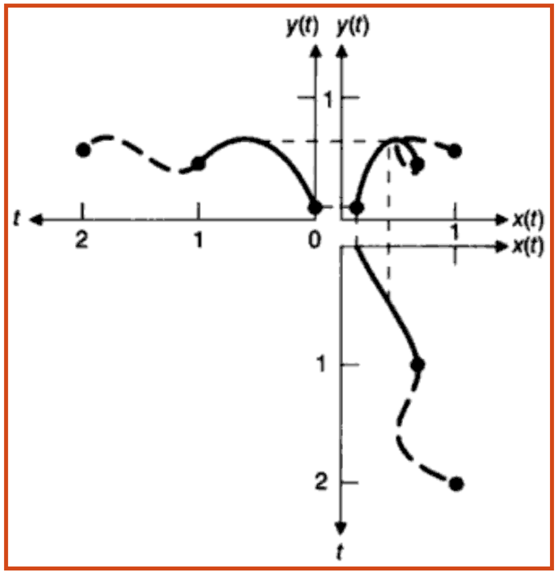
A curve segment defined by the cubic polynomial Q(t)=[x(t) y(t) z(t)]:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z,$$
$$0 \le t \le 1$$

A more compact writing:

$$T = [t^3 \quad t^2 \quad t^1 \quad 1];$$
$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix};$$
$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C$$

An example of two joined parametric cubic curve segments and their polynomials



Continuity

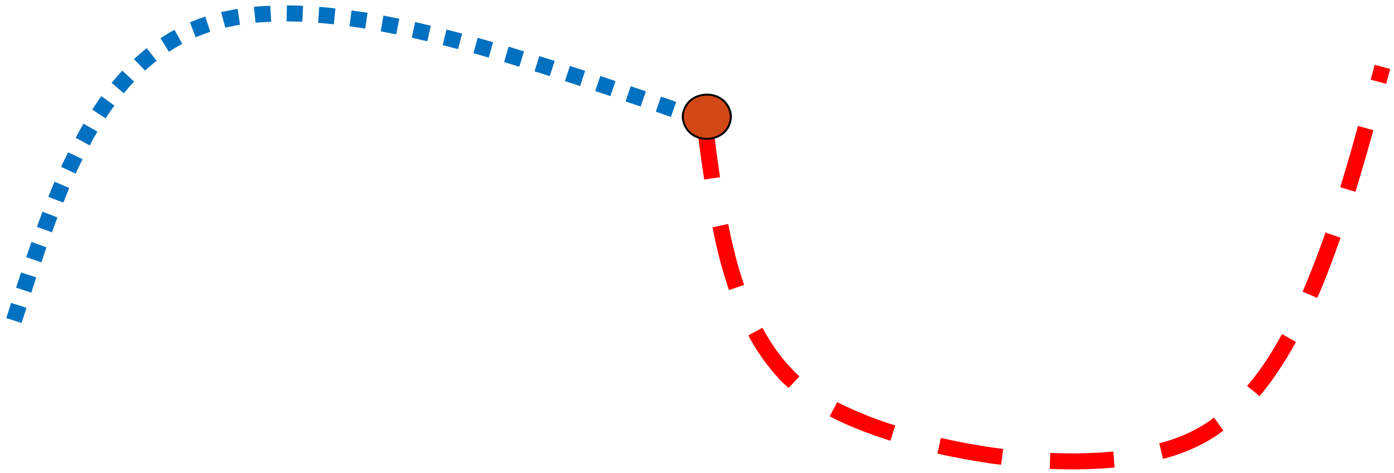
- One of the fundamental concepts
- Commonly used cases:

$$C^0, C^1, C^2$$

- Consider two curves: a(u) and b(u) (u is in [0,1])

Positional Continuity

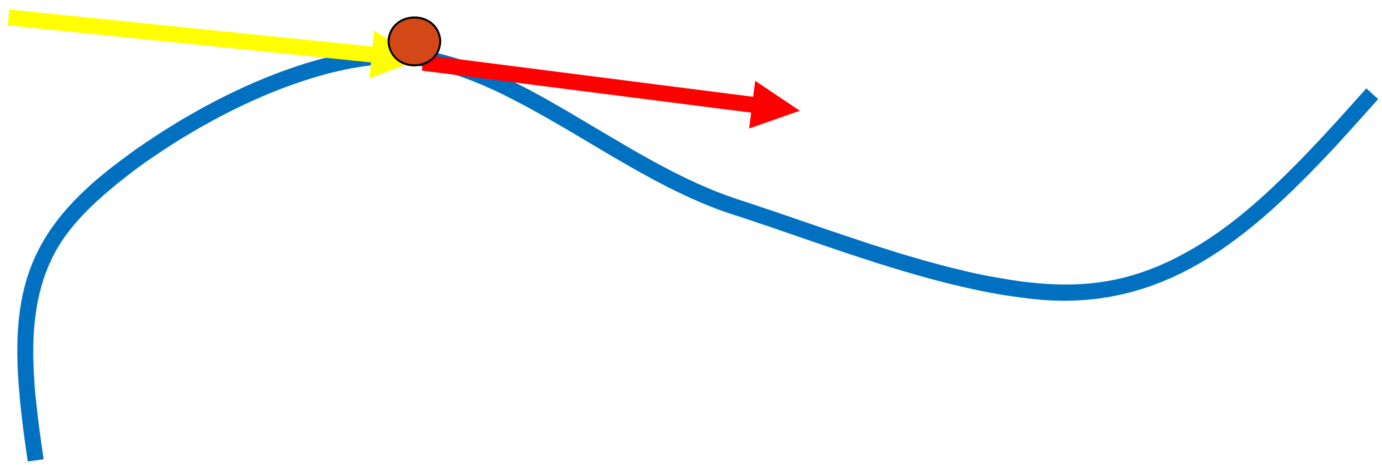
$$\mathbf{a}(1) = \mathbf{b}(0)$$



Derivative Continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

$$\mathbf{a}'(1) = \mathbf{b}'(0)$$

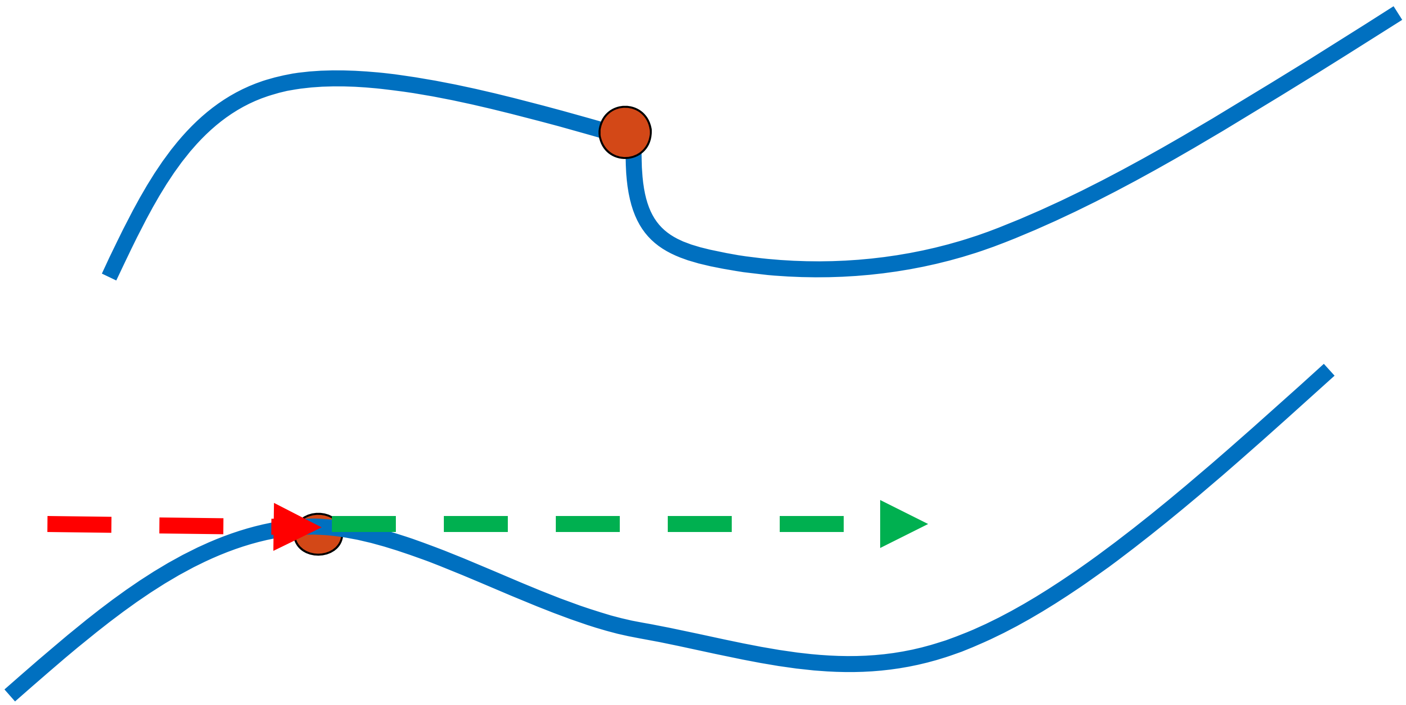


General Continuity

- C^n continuity: derivatives (up to n-th) are the same at the joining point
$$\mathbf{a}^{(i)}(1) = \mathbf{b}^{(i)}(0)$$
$$i = 0, 1, 2, \ldots, n$$
- The prior definition is for parametric continuity
- Parametric continuity depends of parameterization. But, parameterization is not unique.
- Different parametric representations may express the same geometry
- Re-parameterization can be implemented
- Another type of continuity: *geometric continuity*, or G^n

Geometric Continuity

- G^0 and G^1



Geometric Continuity

- Depend on the curve geometry
- DO NOT depend on the underlying parameterization
- G^0 : the same joint
- G^1: two curve tangents at the joint align, but may (or may not) have the same magnitude
- G^n : → C^n after the reparameterization
- Which condition is stronger?
  - geometric continuity is a relaxed for of parametric continuity
  - parametric continuity disallows many parametrizations which generate geometrically smooth curves

Defining and Merging Curve Segments

- A curve segment is defined by constraints on endpoints, tangent vectors (or higher degree derivatives)
- e.g. : on each dimention, a cubic polynomial curve has four coefficients ← four constraints will be needed to solve for the unknowns

→ Most commonly used in computer graphics
  - → Lower-degree polynomials give too little flexibility in controlling the shape of the curve (on position + tangent interpolation)
  - → Higher-degree polynomials can introduce unwanted wiggles and also require more computation
- Three common types of curve segments:
  - Hermite : defined by 2 endpoints + 2 endpoint tangent vectors
  - Bezier : defined by 2 endpoints and 2 other points (that control the endpoint tangent vectors)
  - Several kinds of splines: defined by 4 control points

How coefficients depend on constraints

- Rewrite:

$$T = [t^3 \quad t^2 \quad t^1 \quad 1]; C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix};$$

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C$$

$$= T \cdot M \cdot G = [t^3 \quad t^2 \quad t^1 \quad 1] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

Basis matrix

Geometric vectors (constraints, e.g. end points, tangent)

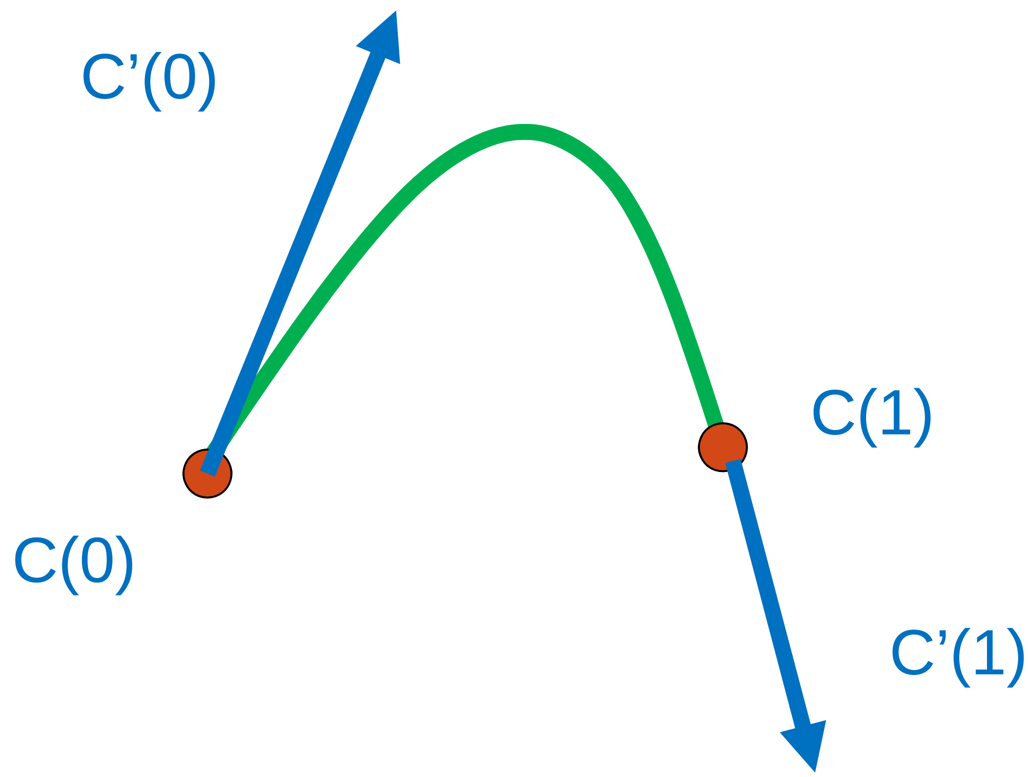
- On x(t):

$$x(t) = T \cdot M \cdot G_x = \sum_{i=1}^{i=4} T \begin{bmatrix} m_{1i} \\ m_{2i} \\ m_{3i} \\ m_{4i} \end{bmatrix} g_{ix}$$

→a curve is a weighted sum of a column (x, or y, or z) of elements of the geometry matrix

- A generalization of straight-line approximation

Cubic Hermite Curve



Cubic Hermite Curve

- Hermite curve

$$\mathbf{c}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}$$

- Two end-points and two tangents at end-points

- Therefore:

$$\begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix} = G_x^H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot M^H \cdot G_x^H$$

M^H=

matrix inverse:

$$x(t) = T \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix}$$

$$y(t) = T \cdot M^H \cdot \begin{bmatrix} y(0) & y(1) & y'(0) & y'(1) \end{bmatrix}^T$$

$$z(t) = T \cdot M^H \cdot \begin{bmatrix} z(0) & z(1) & z'(0) & z'(1) \end{bmatrix}^T$$

Hermite Curve

$$Q(t) = T \cdot M^H \cdot G^H = B^H \cdot G^H$$

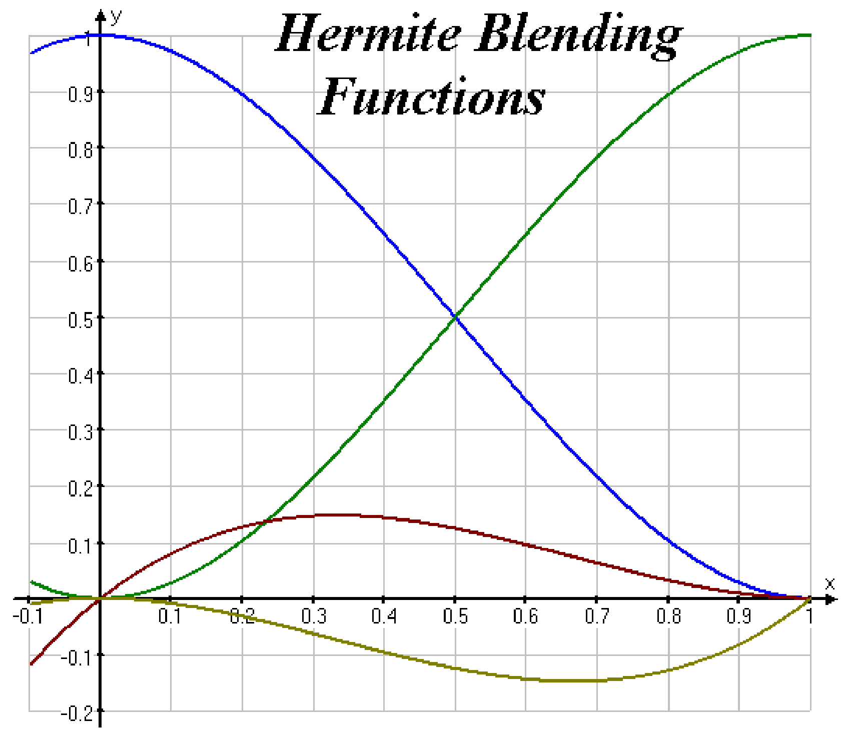
- Basis functions

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

$$f_2(t) = -2t^3 + 3t^2$$

$$f_3(t) = t^3 - 2t^2 + t$$

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



$$\mathbf{c}(t) = \mathbf{c}(0) f_1(t) + \mathbf{c}(1) f_2(t) + \mathbf{c}'(0) f_3(t) + \mathbf{c}'(1) f_4(t)$$

Cubic Hermite Splines

- Two vertices and two tangent vectors:

$$\mathbf{c}(0) = \mathbf{v}_0, \mathbf{c}(1) = \mathbf{v}_1;$$
$$\mathbf{c}^{(1)}(0) = \mathbf{d}_0, \mathbf{c}^{(1)}(1) = \mathbf{d}_1;$$

- Hermite curve

$$\mathbf{c}(t) = \mathbf{v}_0 H_0^3(t) + \mathbf{v}_1 H_1^3(t) + \mathbf{d}_0 H_2^3(t) + \mathbf{d}_1 H_3^3(t);$$
$$H_0^3(t) = f_1(t), H_1^3(t) = f_2(t), H_2^3(t) = f_3(t), H_3^3(t) = f_4(t)$$

Hermite Splines

- Higher-order polynomials

$$\mathbf{c}(t) = \mathbf{v}_0^0 H_0^n(t) + \mathbf{v}_0^1 H_1^n(t) + \ldots + \mathbf{v}_0^{(n-1)/2} H_{(n-1)/2}^n(t)$$

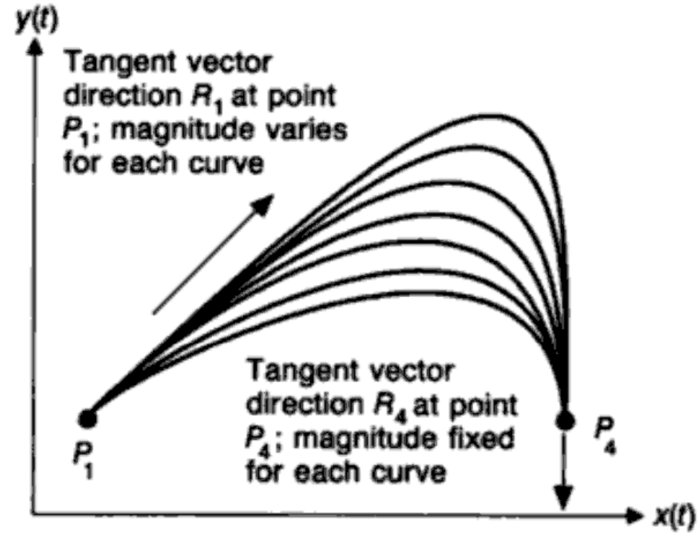
$$+ \mathbf{v}_1^{(n-1)/2} H_{(n+1)/2}^n(t) + \ldots + \mathbf{v}_1^1 H_{(n-1)}^n(t) + \mathbf{v}_1^0 H_n^n(t);$$

$$\mathbf{v}_0^i = \mathbf{c}^{(i)}(0), \mathbf{v}_1^i = \mathbf{c}^{(i)}(1), i = 0, \ldots (n-1)/2;$$

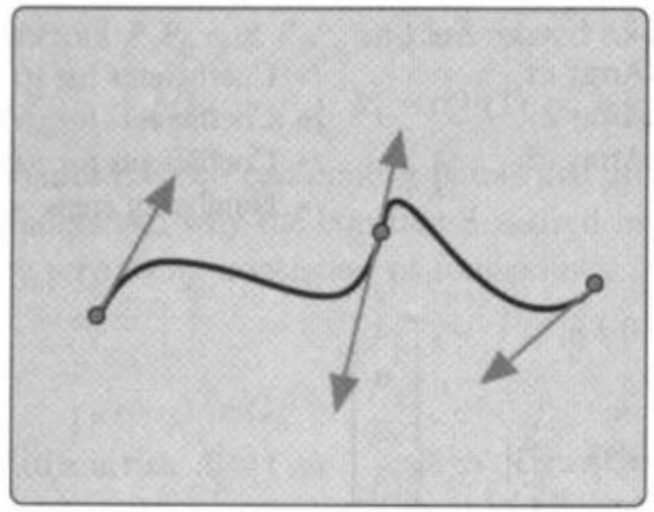
- Note that, n is odd!
- Geometric intuition
- Higher-order derivatives are required

Series of Hermite Curves

- Tangent vector direction and the curve shape
  - see the right figure for an example, increasing magnitude of R_1 → higher cuves



- Continuity between two connecting Hermite cubic curves:
  - Same end-points
  - Same tangent vectors



High-Degree polynomials VS Piecewise Polynomial

- More degrees of freedom
- Easy to formulate
- Infinitely differentiable
- Drawbacks:
  - High-order
  - Global control
  - Expensive to compute, complex
  - undulation

Piecewise Polynomials

- Piecewise --- different polynomials for different parts of the curve
- Advantages --- flexible, low-degree
- Disadvantages --- how to ensure smoothness at the joints (continuity)

Piecewise Hermite Curves

- How to build an interactive system to satisfy various constraints
- C0 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

- C1 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

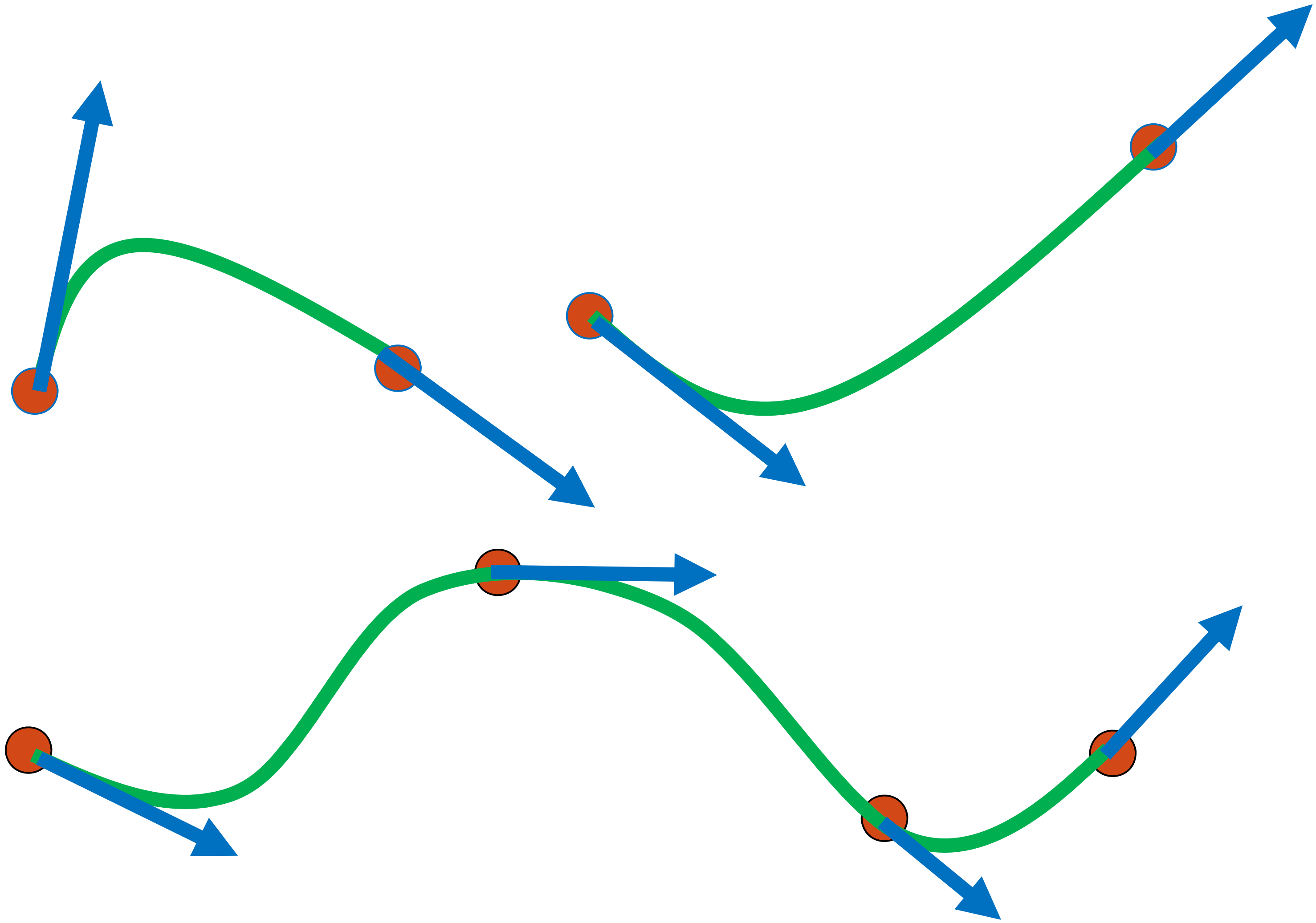
$$\mathbf{a}'(1) = \mathbf{b}'(0)$$

- G1 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

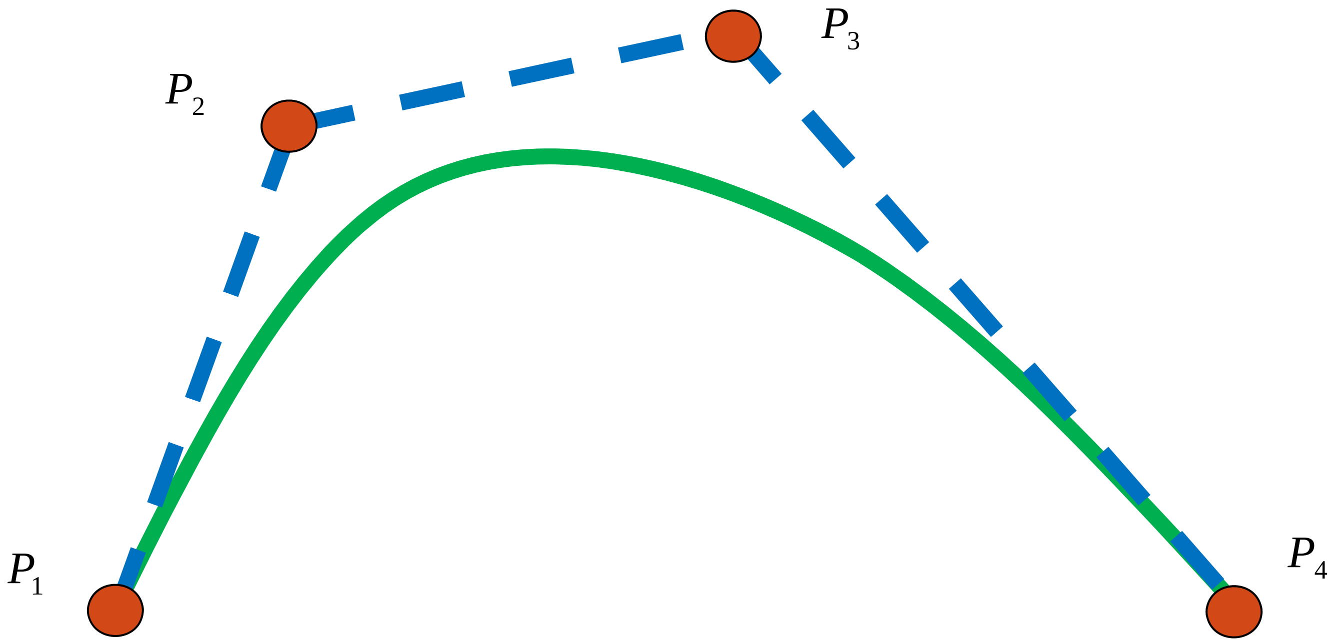
$$\mathbf{a}'(1) = \alpha\,\mathbf{b}'(0)$$

Piecewise Hermite Curves



Bezier Curve

Interpolate the two end control points,
and approximates the other two points:



$$Q'(0) = 3(P_2 - P_1);\ Q'(1) = 3(P_4 - P_3)$$

Basis Matrix for Bezier Curve

- Following the last equation:

$$\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix} = G_x^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \cdot \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = M^{HB} \cdot G^B$$

Bezier geometry vector

- Therefore, we derive the Bezier basis matrix from the Hermit form:

$$G^H = M^{HB} \cdot G^B ; M^B = M^H \cdot M^{HB} ;$$

$$Q(t) = T \cdot M^H \cdot G^H = T \cdot M^H (M^{HB} \cdot G^B) = T \cdot M^B \cdot G^B ;$$

$$M^B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \Rightarrow T \cdot M^B = \begin{bmatrix} B_0^3(t) = (1-t)^3 \\ B_1^3(t) = 3t(1-t)^2 \\ B_2^3(t) = 3t^2(1-t) \\ B_3^3(t) = t^3 \end{bmatrix}$$

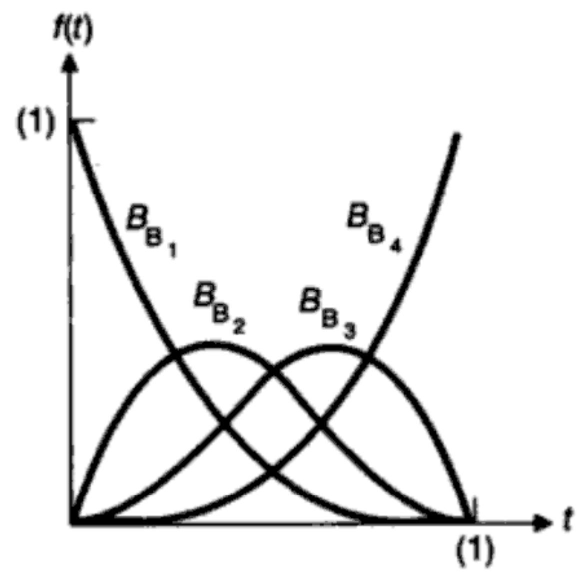
Bernstein Polynomials

- Bezier curve

$$\mathbf{c}(t) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(t)$$

- Control points and basis functions

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



Recursive Evaluation

- Recursive linear interpolation

$$(1-t) \quad (t)$$

$$\mathbf{p}_0^0 \quad \mathbf{p}_1^0 \quad \mathbf{p}_2^0 \quad \mathbf{p}_3^0$$

$$\mathbf{p}_0^1 \quad \mathbf{p}_1^1 \quad \mathbf{p}_2^1$$

$$\mathbf{p}_0^2 \quad \mathbf{p}_1^2$$

$$\mathbf{p}_0^3 = \mathbf{c}(t)$$