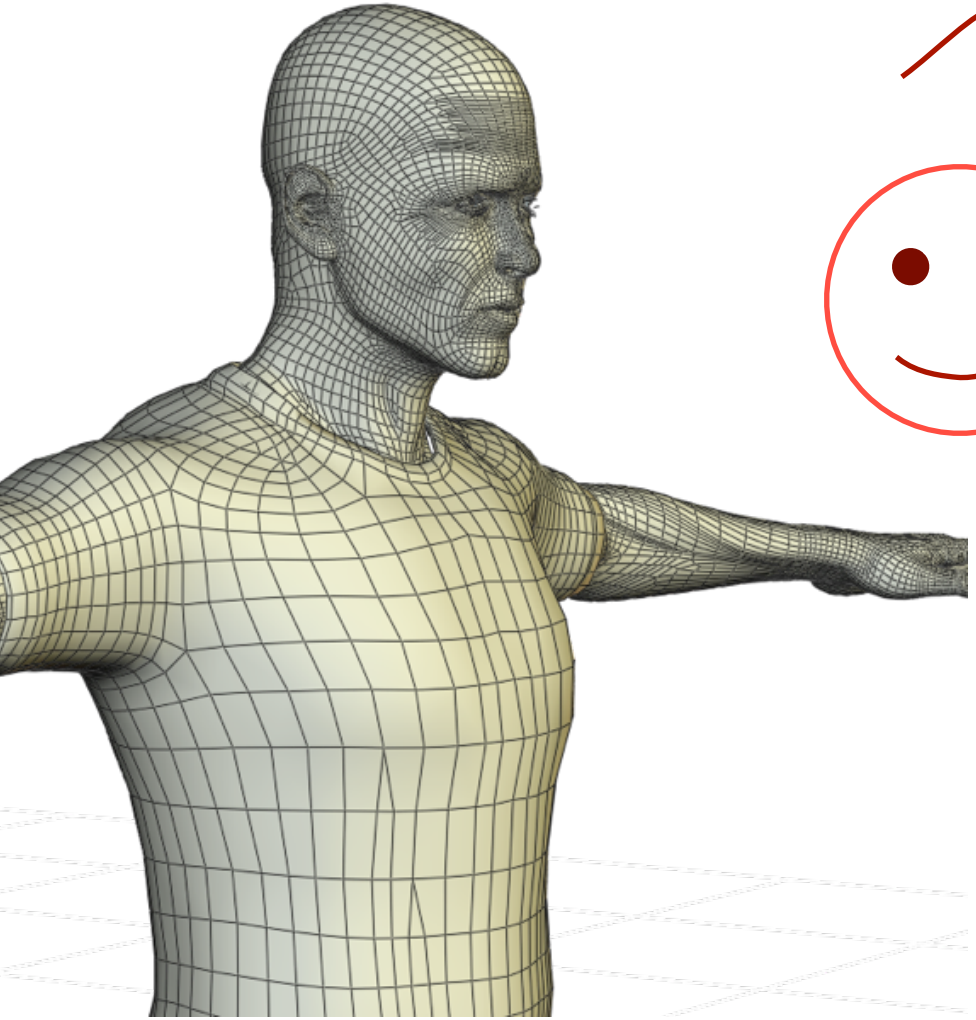
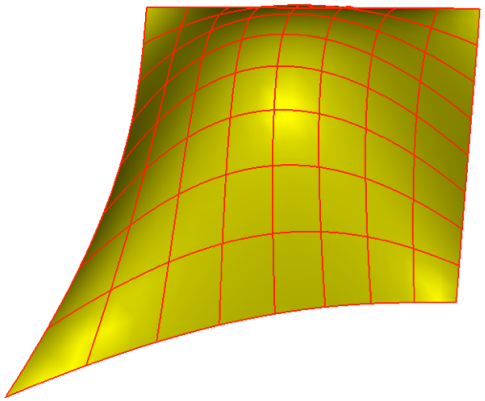
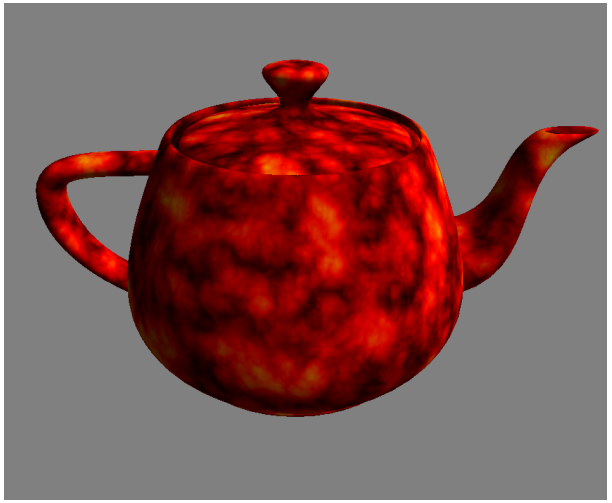
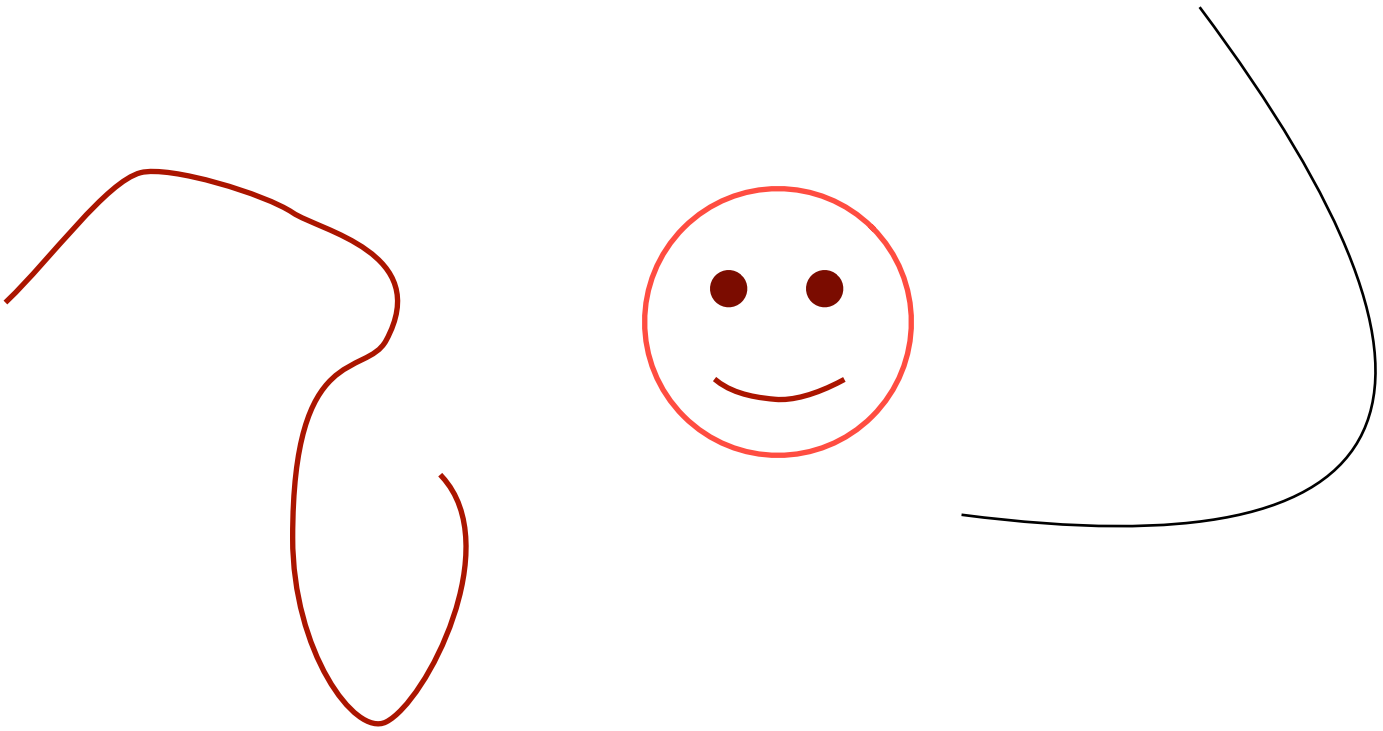
Curves and Surfaces







Curve Design



Representing curves

Explicit: $y = f(x), z=g(x)$

- Cannot get multiple values for single x, infinite slopes
- E.g. cannot represent a circle

Implicit (2D): $f(x,y) = 0$

- Cannot easily compare tangent vectors at joints
- Above/below test, normals from gradient

Parametric: $x=f_x(t), y = f_y(t), z= f_z(t)$

- Often the most convenient formulation
- Tangent of curve $(f(t),g(t),h(t))$ is $(f't), g'(t),h'(t))$ where ' indicates derivative wrt to t

Describing curves by means of polynomials

Reminder:

Lth degre polynomial

$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_L t^L$$

$a_0, \ldots a_L$ are the coefficients

L : is the degree

$L + 1$ is the order

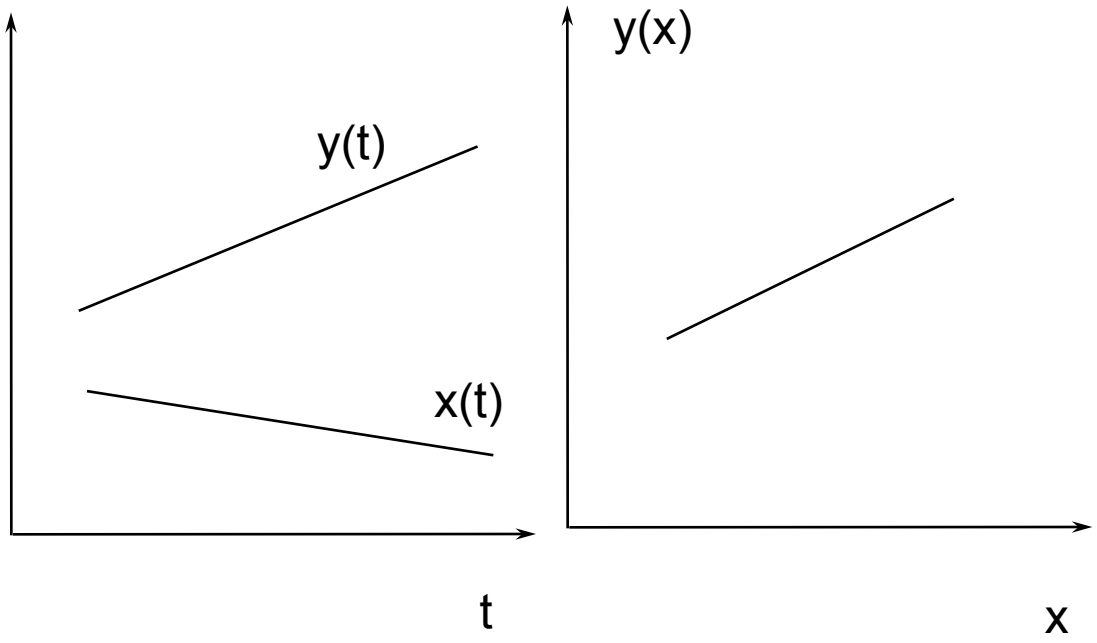
Polynomial Curves of Degree 1

Parametric and implicit forms are linear

$$x(t) = at + b$$

$$y(t) = ct + d$$

$$F(x,y) = kx + ly + m$$



Polynomial Curves of Degree 2

Parametric

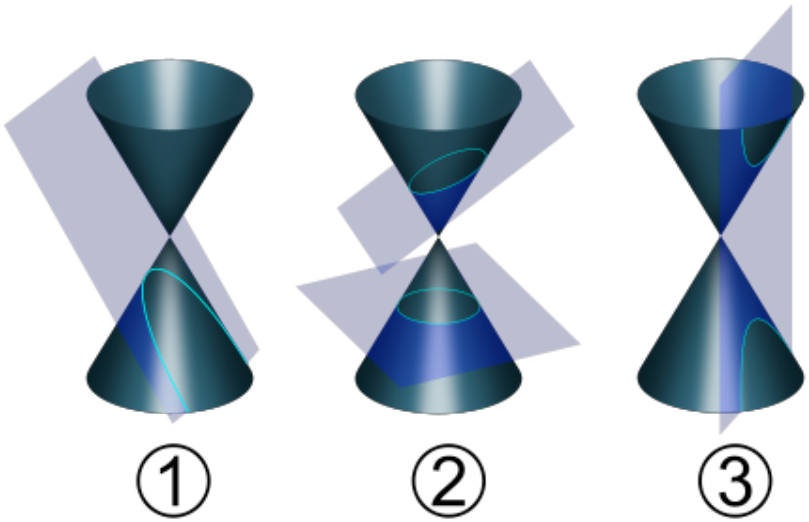
$x(t) = at^2+2bt+c$

$y(t) = dt^2+2et+f$

t *in* $\mathbf{R}$

For any choice of constants

a,d,c,d,e,f →parabola



Implicit

$F(x,y) = Ax^2+2Byx+Cy^2+Dx+Ey+G$

Let $d = AC-B^2$

$d > 0 \rightarrow F(x,y) = 0$ is an ellipse

$d = 0 \rightarrow F(x,y) = 0$ is a parabola

$d < 0 \rightarrow F(x,y) = 0$ is a hyperbola

1. Parabola
2. Circle and Ellipse
3. Hyperbola

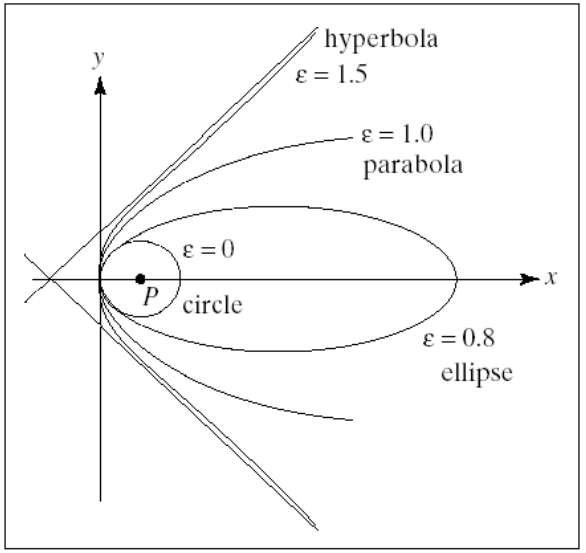
Courtesy of Pbroks13, Wikipedia

Polynomial curves of degree 2

Common Vertex form:

$$y^2 = 2px - (1 - \varepsilon^2)x^2$$

FIGURE 11.5 The common-vertex equations of the conics.



Rational Quadratic Parametric Curves

$$P(t) = \frac{P_0(1-t)^2 + 2wP_1t(1-t) + P_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}$$

w < 1 ellipse

w = 1 parabola

w > 1 hyperbola

Other kinds of curves

Sinusoidal

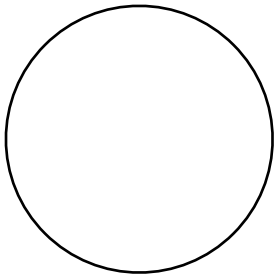
Exponential

Complex

Fractals

Example:

Canonical Circle:



Implicit

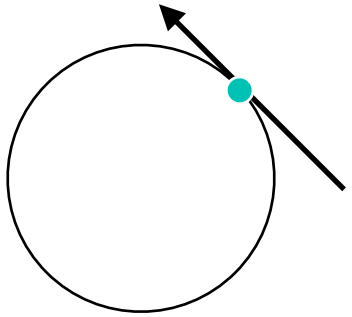
$x^2 + y^2 = 1$

Parametric

$(x = \cos(t), y = \sin(t)), t \in [0, 2\pi]$

Differential Geometry Concepts

Tangent vector



- Given a parameterized curve
  $p(t)$: $\tau(t) = p'(t) = dp(t)/dt$
  defines the forward orientation of the curve and the parametric velocity
- Notice that it is a function of t and that it is a **vector**
- It defines locally (instantaneously) the shape of the curve at each point and can be used for interactive design tools
- Unit tangent: $\mathbf{T}(t) = \tau(t) \,/\, ||\tau(t)||$

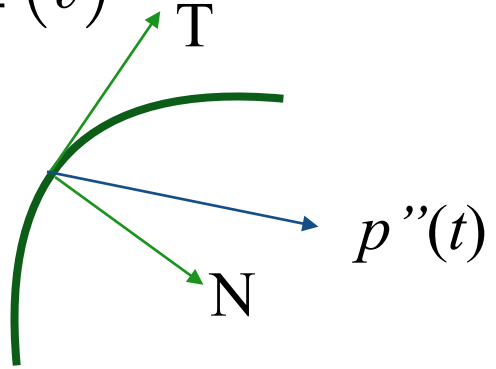
Differential Geometry Concepts

Normal vector (AKA curvature vector)

- Given the **unit** tangent vector $\mathbf{T}(t)$:

$$e_2(t) = p''(t) - (p''(t) \cdot \mathbf{T}(t))\,\mathbf{T}(t)$$

$$\mathbf{N}(t) = \frac{e_2(t)}{||e_2(t)||}$$



- Indicates how far from a straight line the curve is (Line has 0 curvature)
- Normal to the tangent vector $\mathbf{T}$

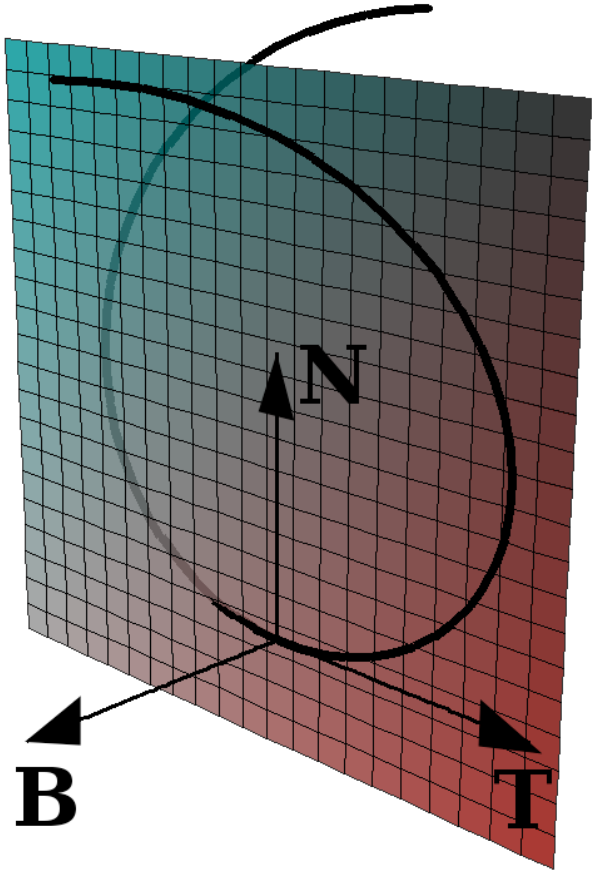
Differential Geometry Concepts

Binormal

- Normal to both tangent and normal vectors
- In 3D:
  $\mathbf{B} = \mathbf{T} \otimes \mathbf{N}$

Frenet frame

The three vectors together form the Frenet frame



Osculating plane defined by $\mathbf{N},\mathbf{T}$

Curve design in graphics and animation

- In graphics we often want to interpolate or approximate a set of values in an efficient and predictable way

- We use parametric polynomials and constrain them to create desired types of curves

Parametric polynomial curve design in graphics

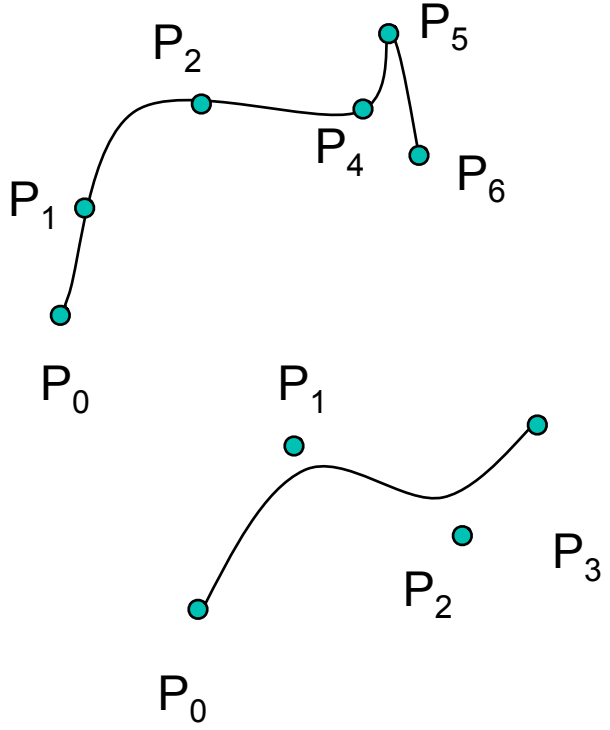
Geometric approach

Any t

$P_0, \ldots, P_L \rightarrow$ Curve generation $\rightarrow P(t)$

Constraints *Polynomial* *Curve*

Interpolation vs Approximation



Parametric polynomial curve design in graphics

Geometric approach

Any t

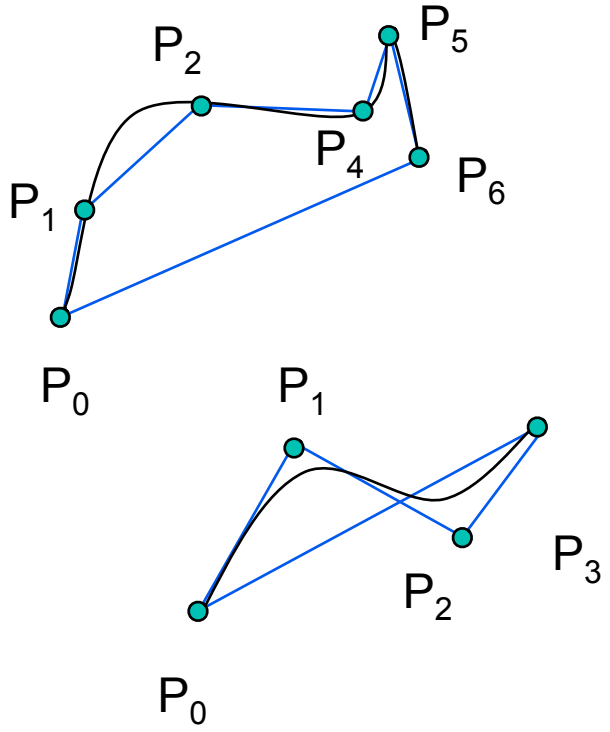
$P_0, \ldots, P_L \rightarrow$ Curve generation $\rightarrow P(t)$

Constraints *Polynomial* *Curve*

P_i control points

$P_0 \ldots P_L$ control polygon

Interpolation vs Approximation



De Casteljau Algorithm

Tweening

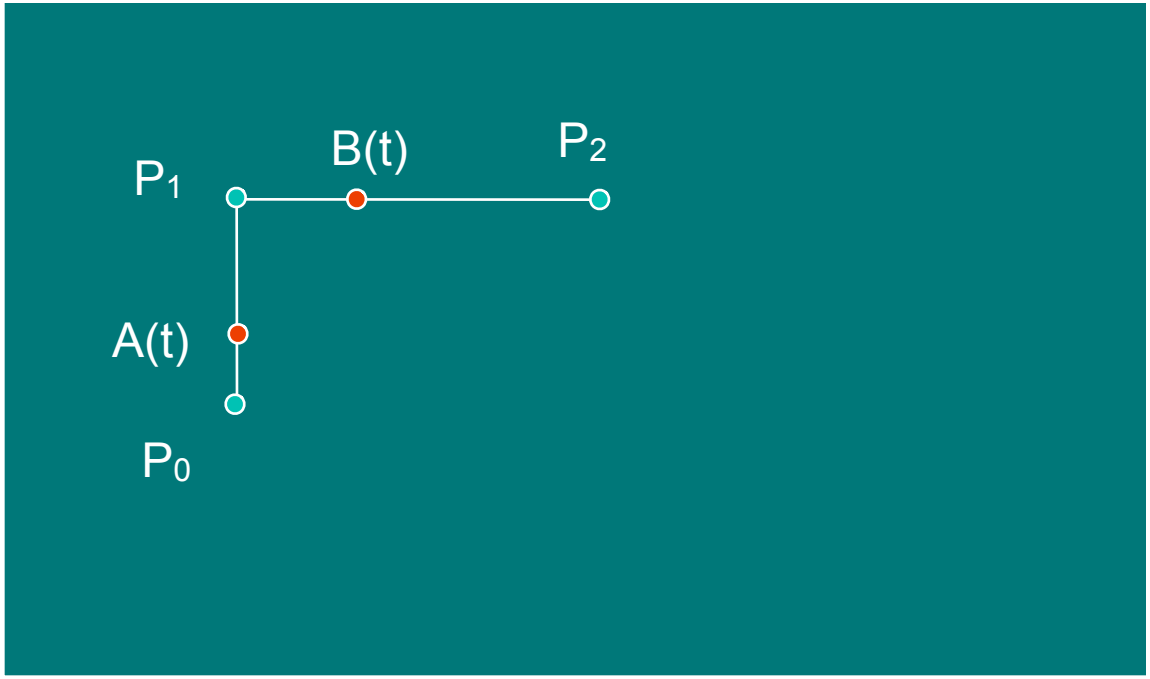
Two points=line

$A(t) = (1-t)P_0 + tP_1$

De Casteljau Algorithm

Tweening

Three points



$A(t) = (1-t)P_0 + tP_1$

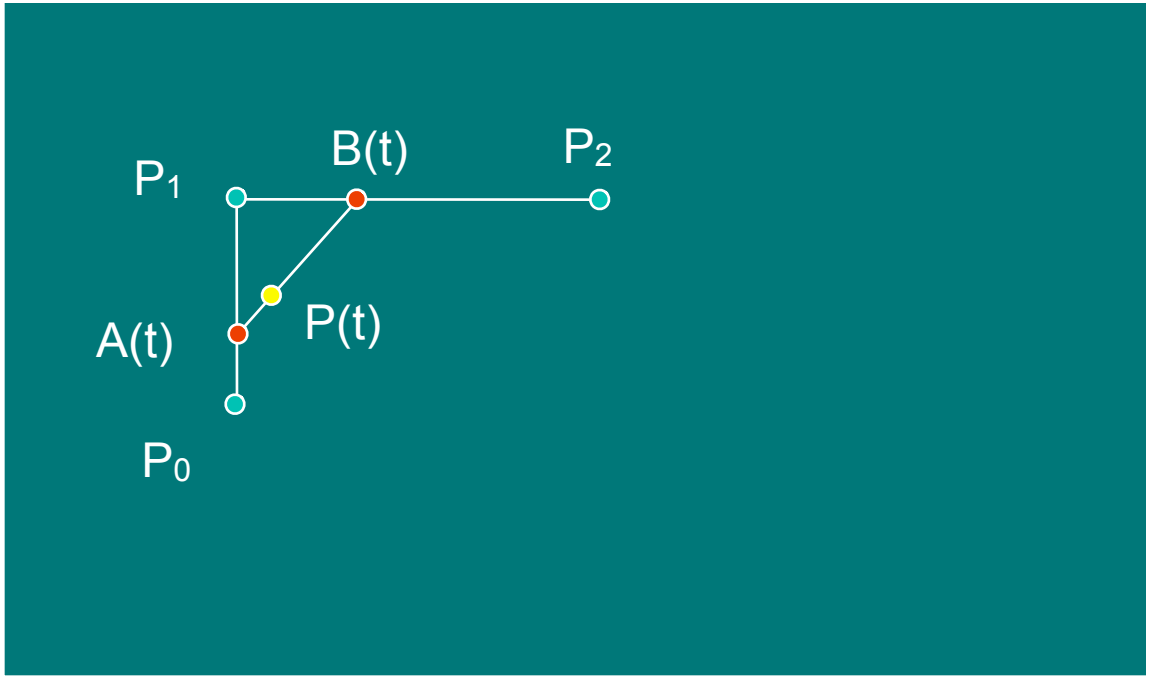
$B(t) = (1-t)P_1 + tP_2$

De Casteljau Algorithm

Tweening

Three points

(parabola)



$A(t) = (1-t)P_0 + tP_1$

$B(t) = (1-t)P_1 + tP_2$

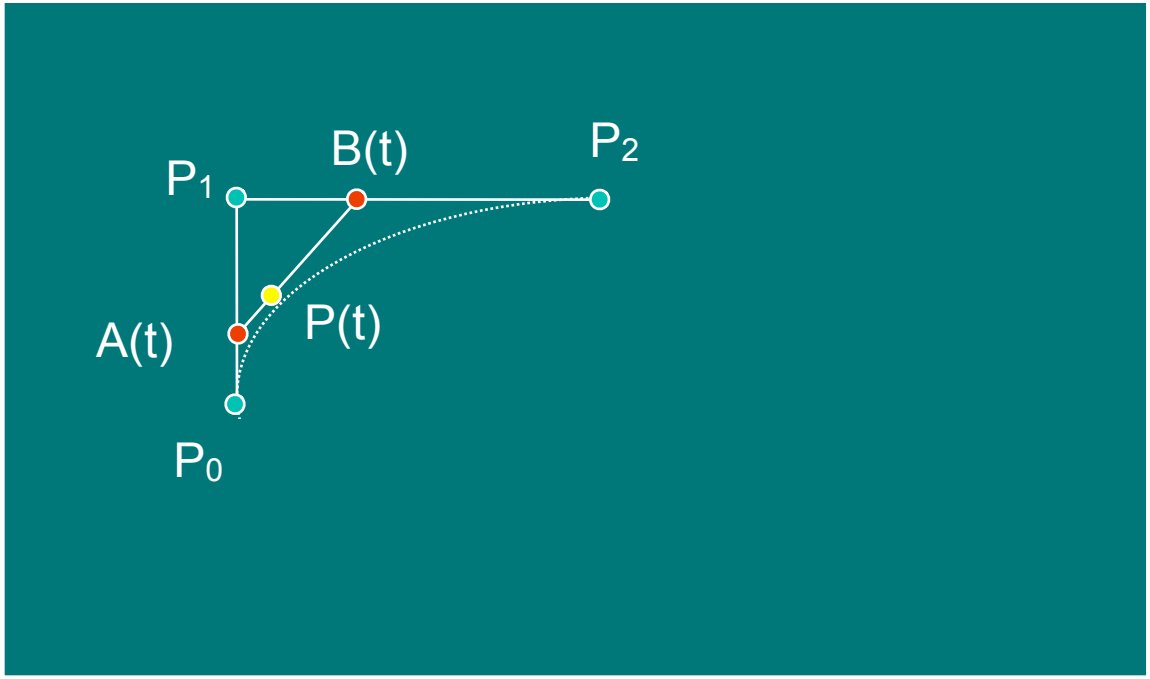
$P(t) = (1-t)\,A(t) + tB(t) = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2$

De Casteljau Algorithm

Tweening

Three points

(parabola)



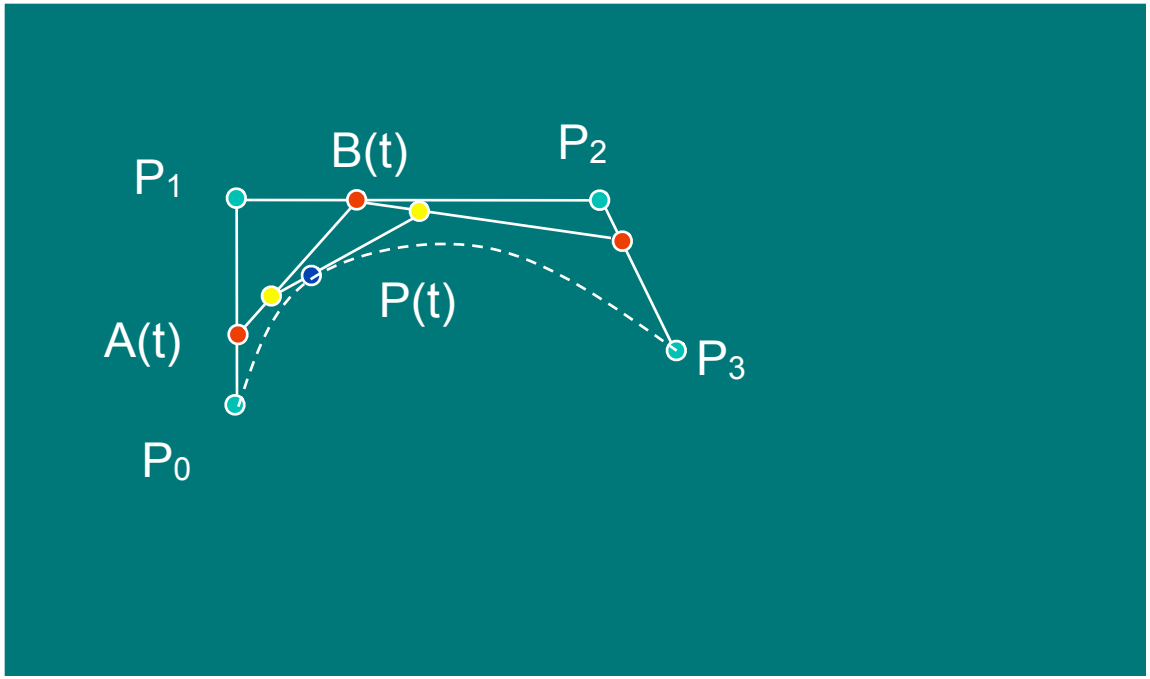
$A(t) = (1-t)P_0 + tP_1$

$B(t) = (1-t)P_1 + tP_2$

$P(t) = (1-t)\,A(t) + tB(t) = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2$

De Calsteljau (cont)

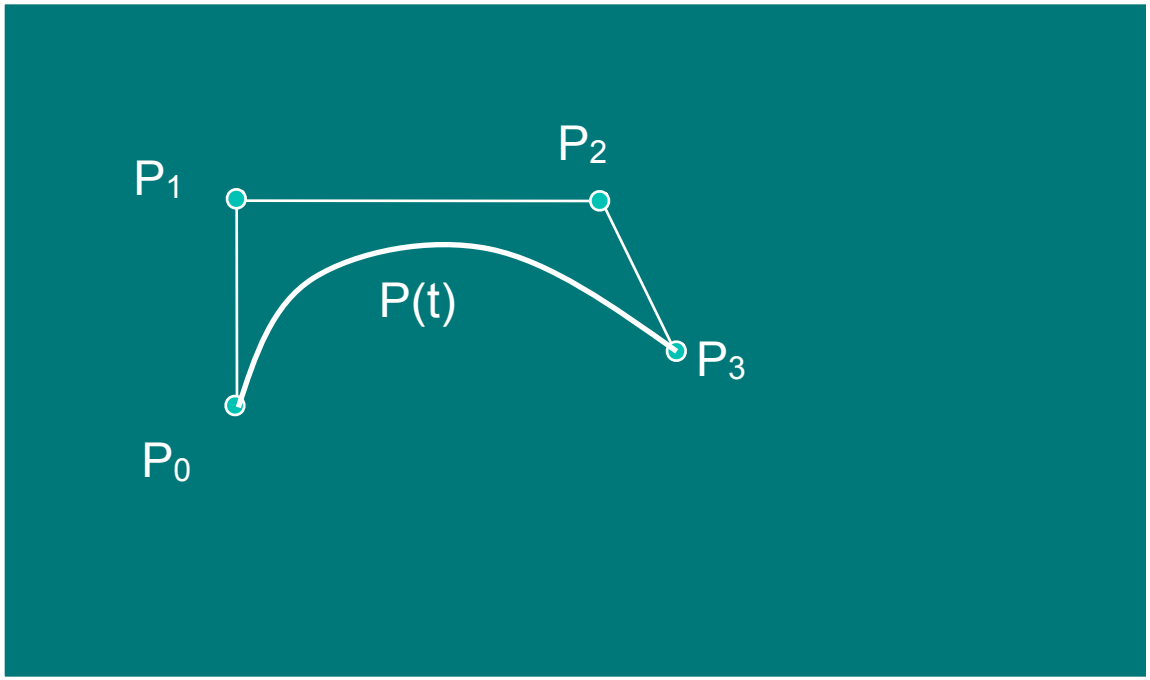
Tweening with four points



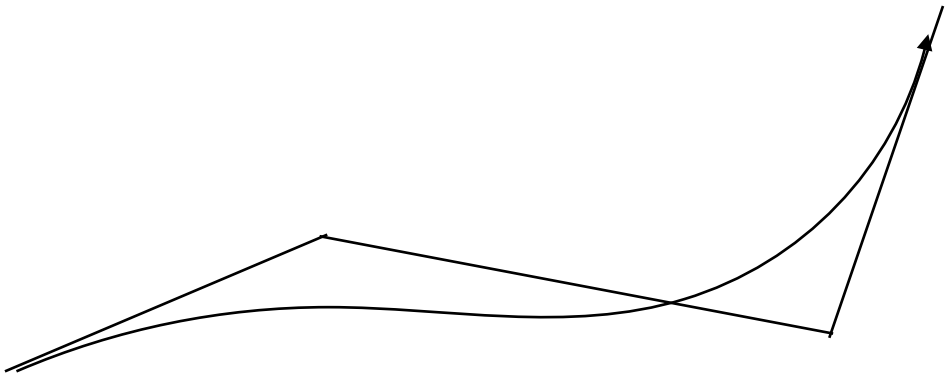
$$P(t) = (1-t)^3P_0+3(1-t)^2tP_1+3(1-t)t^2P_2+t^3P_3$$

Bezier Curves

One of the most fundamental concepts in curve design

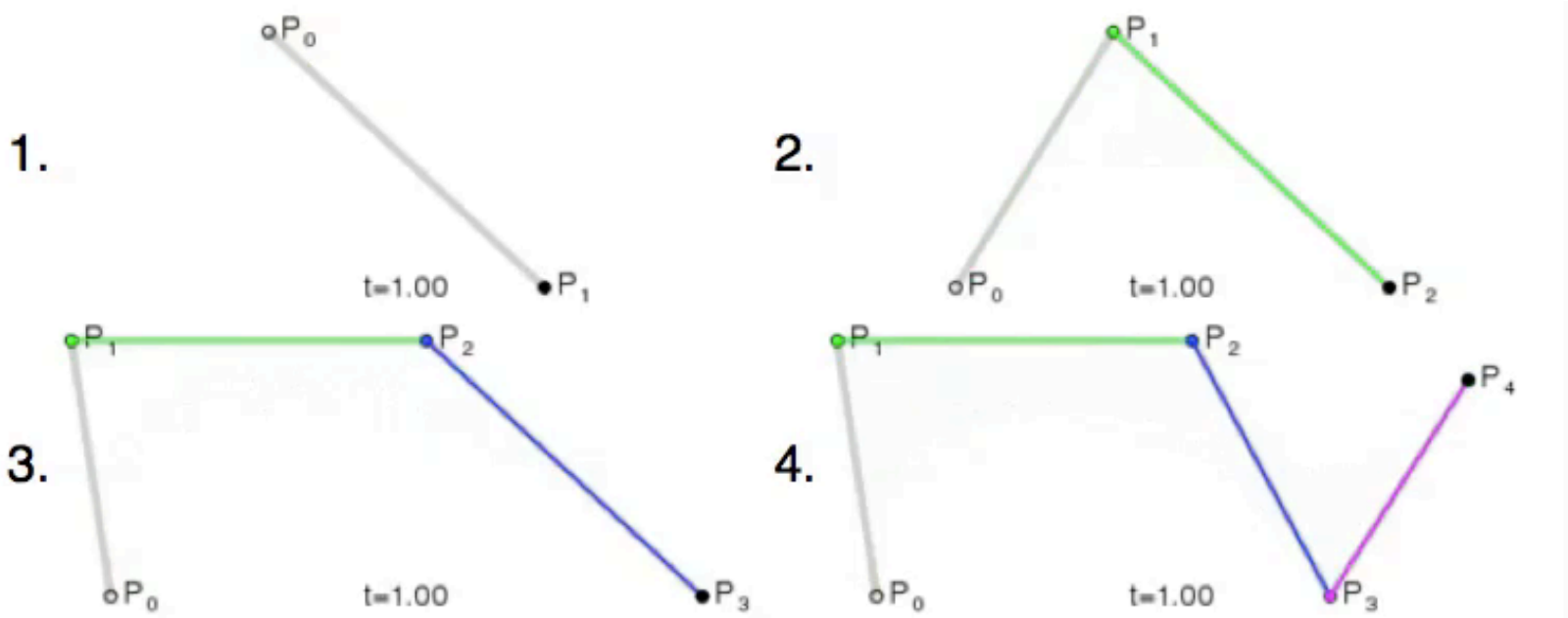


$$P(t) = (1-t)^3P_0+3(1-t)^2tP_1+3(1-t)t^2P_2+t^3P_3$$



Visualization

- Courtesy of Phil Tregoning, Wikipedia



Bézier curves: 1. linear; 2. quadratic; 3. cubic; 4. quartic.

Coefficients of Bezier Curves: Berstein polynomials

$$P(t) = (1-t)^3P_0+3(1-t)^2tP_1+3(1-t)t^2P_2+t^3P_3$$

$$B^3_0(t) = (1-t)^3$$

$$B^3_1(t) = 3(1-t)^2t$$

$$B^3_2(t) = 3(1-t)t^2$$

$$B^3_3(t) = t^3$$

Expansion of $[(1-t)+t]^3 = (1-t)^3 + 3(1-t)^2t + 3(1-t)t^2 + t^3 \rightarrow$

$$\Sigma B^3_k(t) = 1,\ k = 0,1,2,3$$

Affine combination of points

Berstein Polynomials of L degree

L + 1 control points, P_k

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

$$B_k^L(t) = \begin{pmatrix} L \\ K \end{pmatrix} (1-t)^{L-k} t^k$$

$$\begin{pmatrix} L \\ K \end{pmatrix} = \frac{L!}{k!(L-k)!}, \quad \text{for} \quad L \geq k$$

$$\text{Affine combination:} \sum_{k=0}^{L} B_k^L(t) = 1, \quad \text{for all} \quad t$$

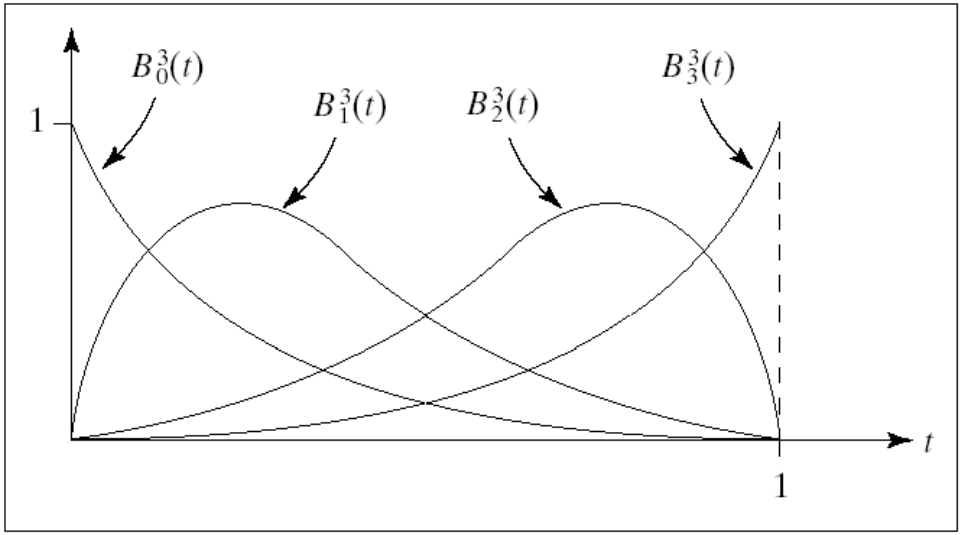
Expansion of $[(1-t) + t]^L$

Other Berstein Polynomials Properties

Allways positive

Zero only at t =0 or 1

Degree 3



Properties of Bezier curves

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

Derivatives of Bezier curves

It can be shown that:

Any derivative also a Bezier curve of lower degree

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

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

Which degree is best?

Cubic curves

- Lower order not enough flexibility
- Higher order too many wiggles and computationally expensive
- Cubic curves are lowest degree polynomial curves that are not planar in 3D

More complex curves

- Piecewise cubics

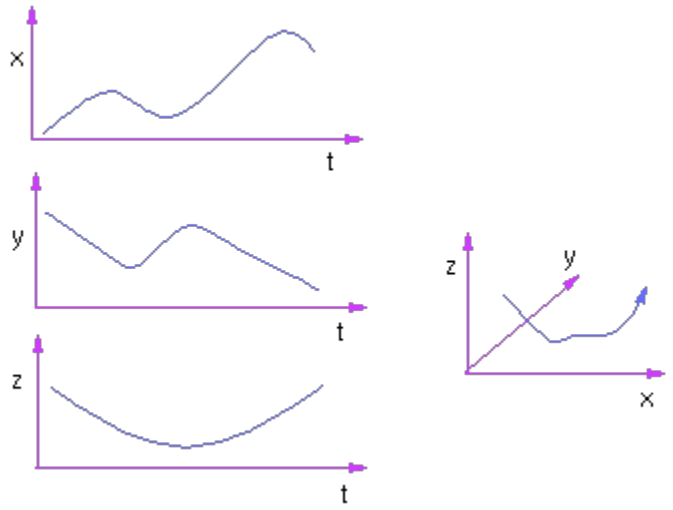
The general case for Cubic Parametric Curves (piece)

- Bezier Cubic Curves are only one possible family of cubic curves.

- We want more control of the constraints on the curves
  - *location*
  - *shape*

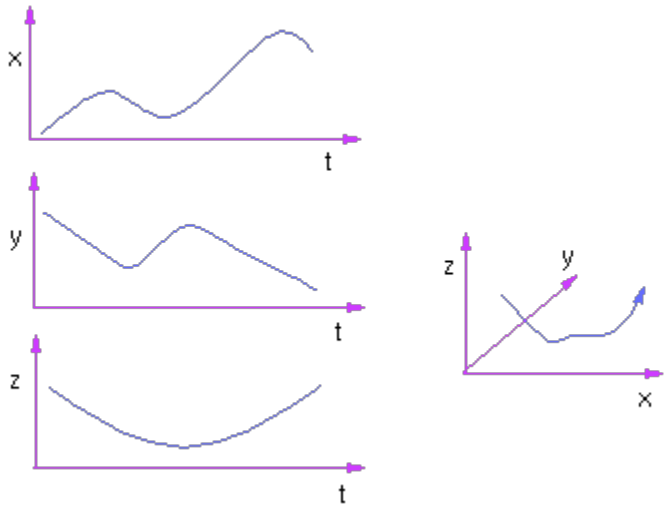
General form of Cubic parametric curves

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$$
$$z(t) = c_3t^3 + c_2t^2 + c_1t + c_0$$
$$t \in [0, 1]$$



Matrix Form

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$$
$$z(t) = c_3t^3 + c_2t^2 + c_1t + c_0$$
$$t \in [0, 1]$$



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

$$x(t) = TA$$
$$y(t) = TB$$
$$z(t) = TC$$

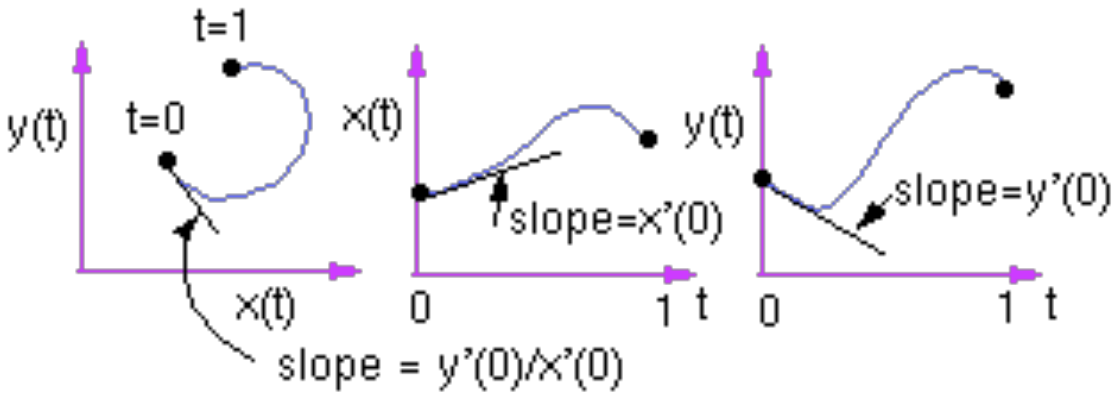
Derivative of Cubic Parametric Curves - tangent vector

Where the curve goes for some t

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

How the curve is shaped for some t

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



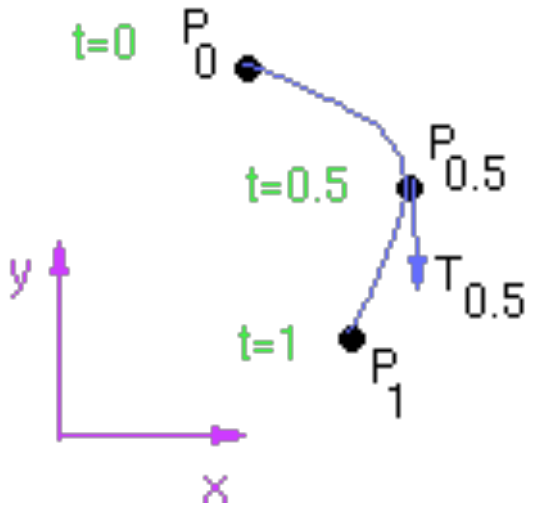
For a given set of constraints we get a family of curves

Constraints

Endpoints and a tangent at midpoint

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

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

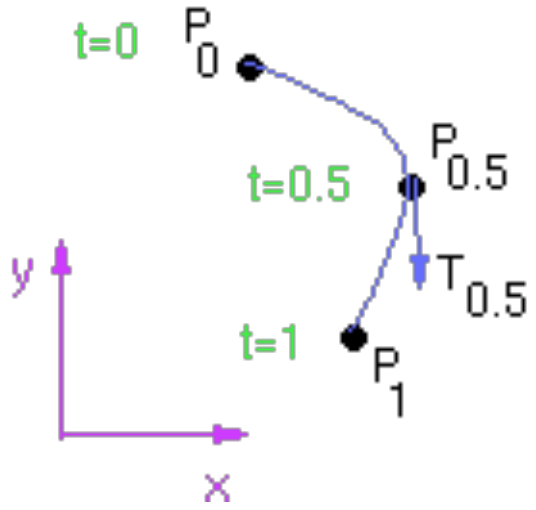


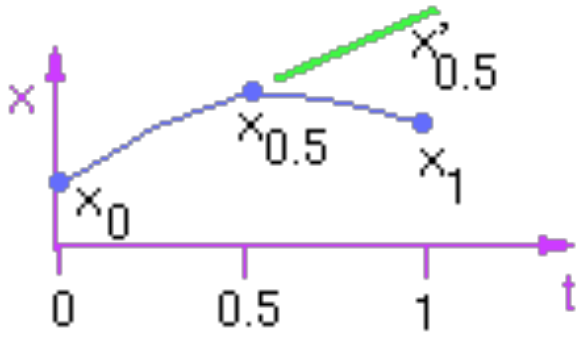
Setting up the curve

Constraints

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

$$x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A$$
$$x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5 & 1 \end{bmatrix} A$$
$$x'(0.5) = \begin{bmatrix} 3(0.5)^2 & 2(0.5) & 1 & 0 \end{bmatrix} A$$
$$x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A$$
$$G_x = BA$$





Solving for A

Constraints

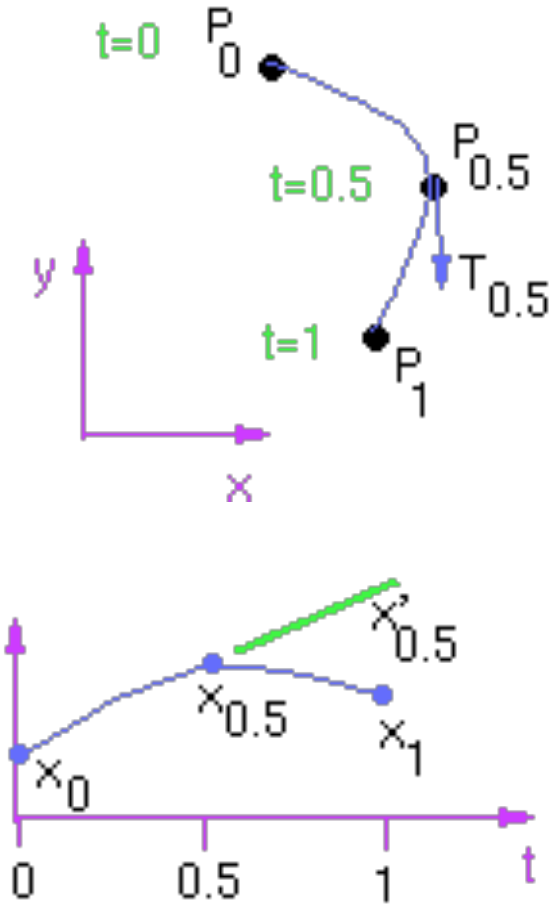
$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} A = TA$$

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

$$\begin{bmatrix} x_0 \\ x_{0.5} \\ x'_{0.5} \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.5^3 & 0.5^2 & 0.5 & 1 \\ 3(0.5)^2 & 2(0.5) & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} A$$

$$G_x = BA \Rightarrow A = B^{-1}G_x$$

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



Final form

Basis matrix

$$x(t) = TB^{-1}G_x$$

$$Set \;\; M = B^{-1}$$

$$x(t) = TMG_x$$

$$y(t) = TMG_y$$

$$z(t) = TMG_z$$

For the example

$$P(t) = TMG$$

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

Blending functions

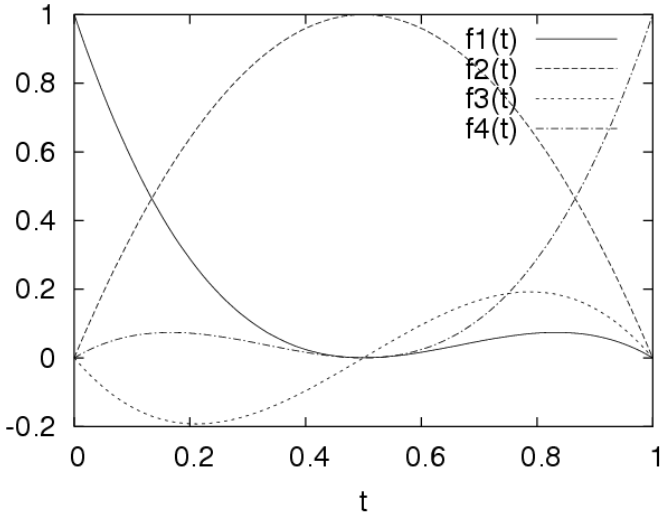
T*M

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

For the example

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

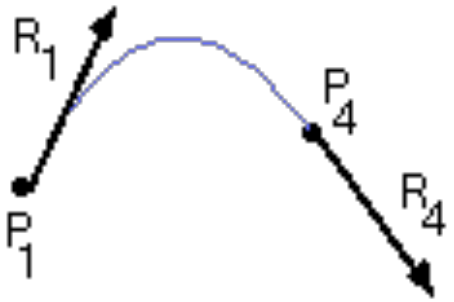
Each blending function weights the contribution of one of the constraints



Hermite Curves

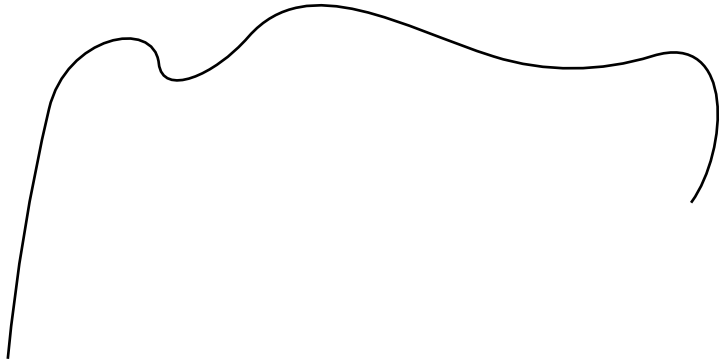
Constraints

Two points and two tangents



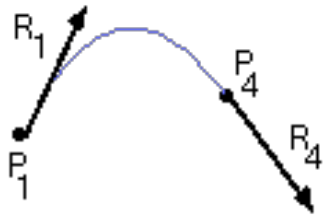
$$G_h = \begin{bmatrix} P_1 & P_4 & R_1 & R_4 \end{bmatrix}$$
$$x(t) = TA_h = TM_hG_h$$

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



Hermite Curves

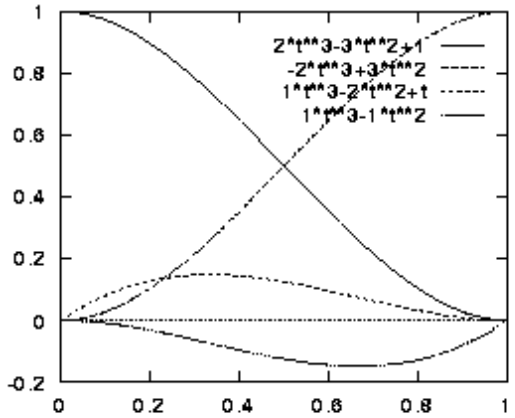
Blending functions



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

$$x(t) = TM_hG_h \Rightarrow$$

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



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

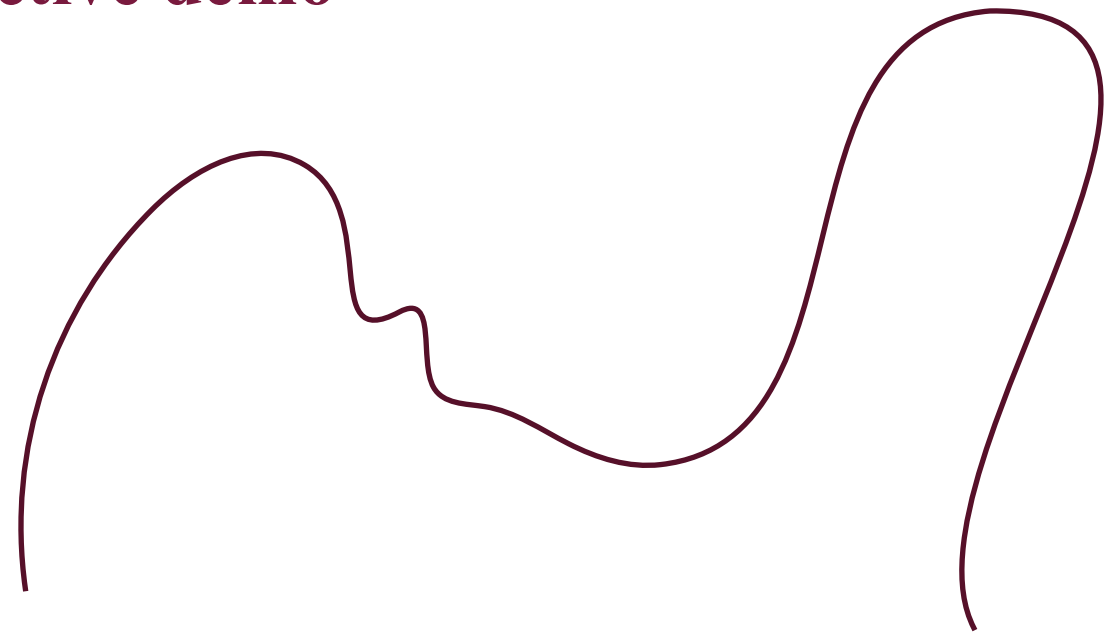
$$f_2(t) = -2t^3 + 3t^2$$

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

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

What does the magnitude of the tangent do?

Interactive demo



Bezier Curves

Special case of Hermite curves

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

Bezier Curves

Special case of Hermite curves

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

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

Bezier Curves

Special case of Hermite curves

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

$$G_h = M_{bh} G_b$$

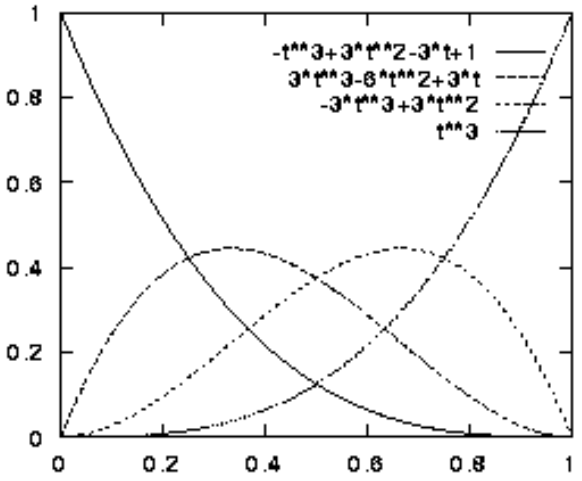
$$P(t) = T M_h G_h \Rightarrow P(t) = T M_h M_{bh} G_b \Rightarrow$$
$$P(t) = T M_b G_b$$

Bezier Curves

Special case of Hermite curves

We can verify that TM_b are the bernstein polynomials

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

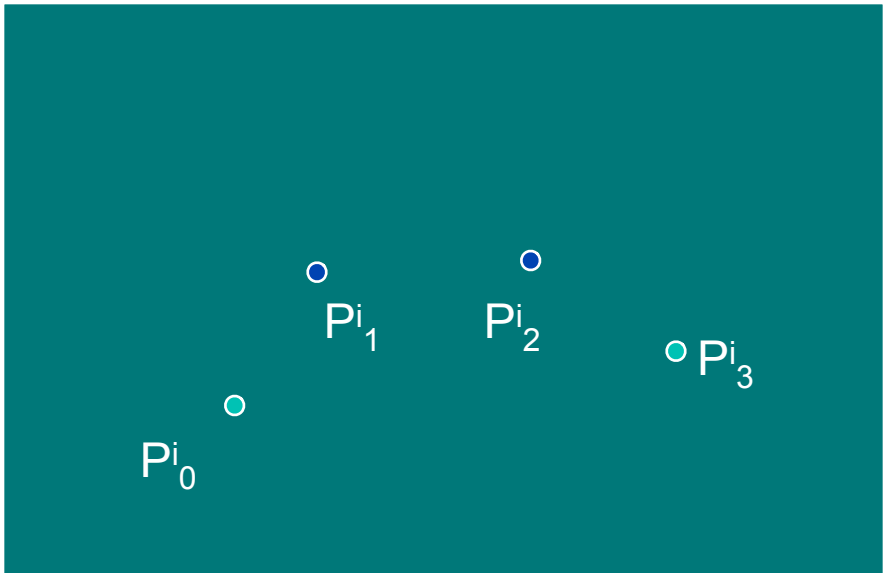


Transforming between representations

Just like Bezier and Hermite curves can be transformed into each other with a matrix multiplication, other families of curves can do so as well

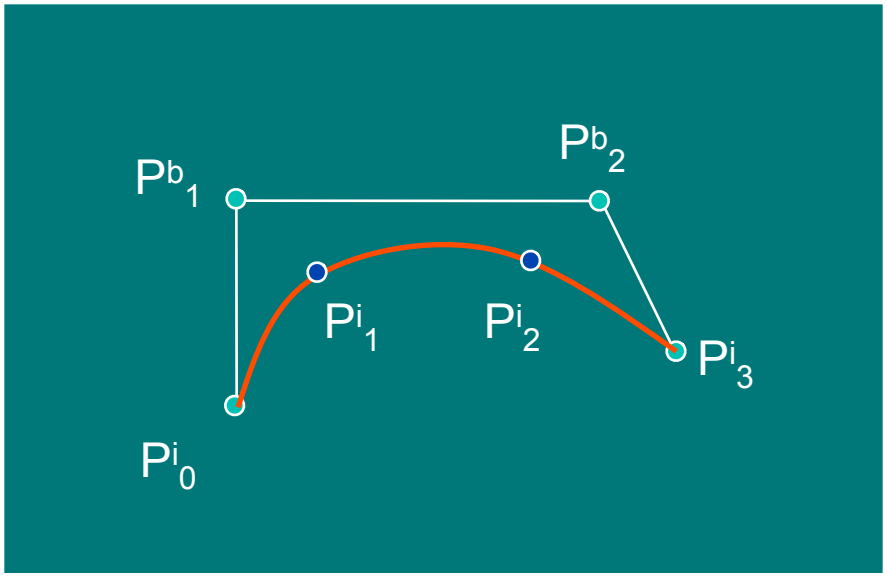
Bezier to Interpolating curves

- Curve must interpolate $P^i_0, P^i_1, P^i_2, P^i_3$



Bezier to Interpolating curves

- Curve must interpolate $P^i_0, P^i_1, P^i_2, P^i_3$
- How can we find the Bezier points P^b from the P^is so that the resulting Bezier curve interpolates the Pi points?

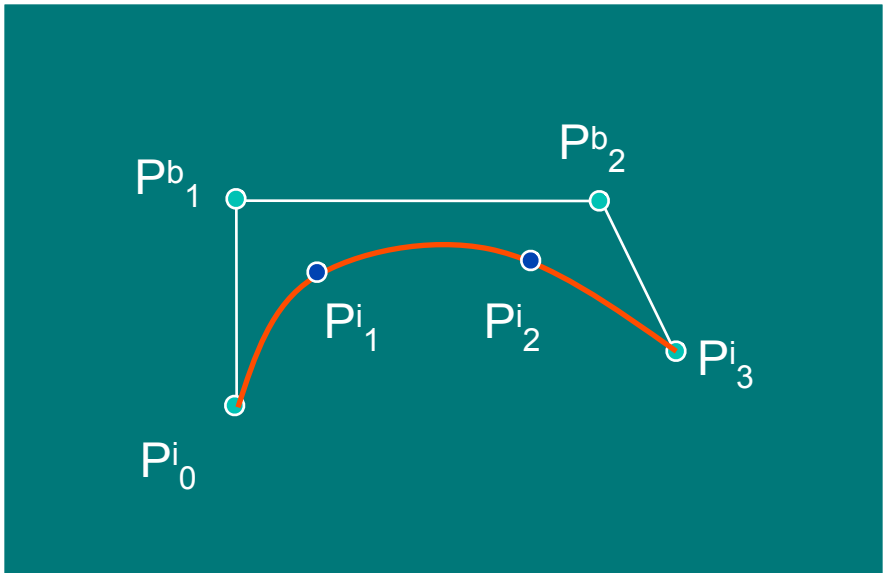


Bezier to Interpolating curves

For the next three slides points are row vectors!!

$$P_j^i = [\ P_j^i, x \quad P_j^i, y \quad P_j^i, z\ ]$$

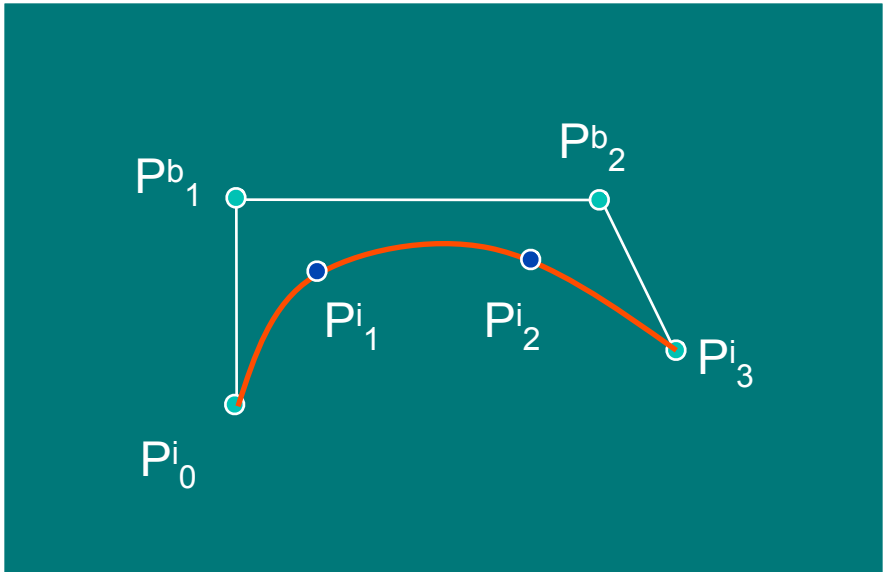
$$G^b = \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix}$$



The t-values are chosen arbitrarily

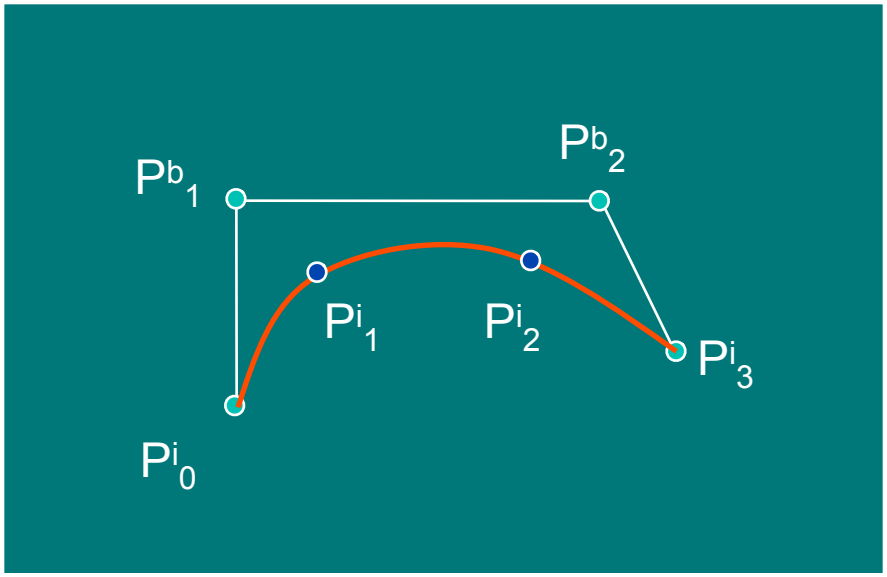
$$\begin{aligned} P_0^i &= T(0)M_bG^b \\ P_1^i &= T(\tfrac{1}{3})M_bG^b \\ P_2^i &= T(\tfrac{2}{3})M_bG^b \\ P_3^i &= T(1)M_bGx^b \end{aligned} \quad \rightarrow \quad \begin{pmatrix} P_0^i \\ P_1^i \\ P_2^i \\ P_3^i \end{pmatrix} = \begin{pmatrix} T(0) \\ T(\frac{1}{3}) \\ T(\frac{2}{3}) \\ T(1) \end{pmatrix} M_b \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix}$$

Bezier to Interpolating curves



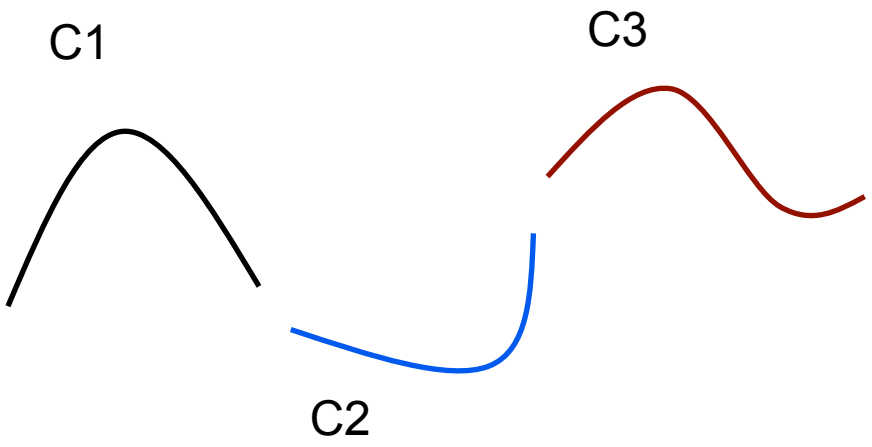
$$\begin{pmatrix} P_0^i \\ P_1^i \\ P_2^i \\ P_3^i \end{pmatrix} = \begin{pmatrix} T(0) \\ T(\frac{1}{3}) \\ T(\frac{2}{3}) \\ T(1) \end{pmatrix} M_b \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix} \Rightarrow \mathbf{P}^i = TM_b\mathbf{P}^b$$

Bezier to Interpolating curves



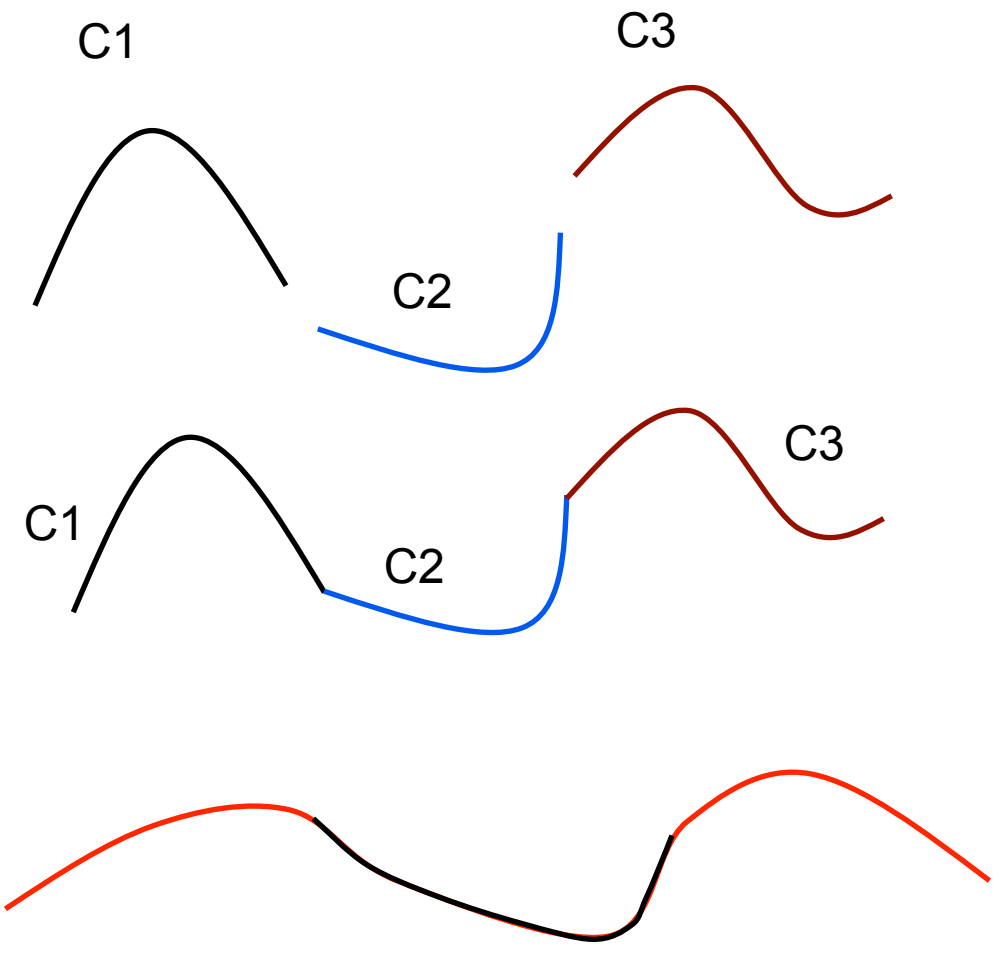
$$\mathbf{P}^i = TM_b\mathbf{P}^b \Leftrightarrow \mathbf{P}^b = (TM_b)^{-1}\mathbf{P}^i$$

Piecewise cubic curves



Connection?

Examples



Continuity - Parametric

Parametric C^k-continuity

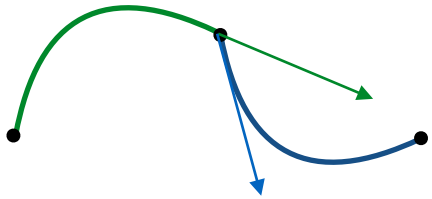
$P^{(i)}$ exists and is continuous ∀t in [a,b] and the joints for ∀i = 0,…,k

Terminology:

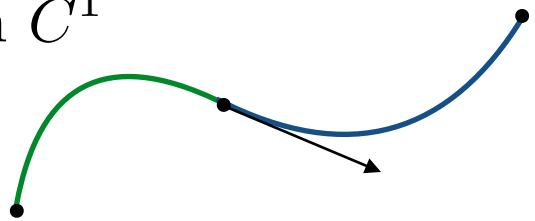
- P is k-smooth
- P has kth-order continuity

($P^{(i)}$ is the i-th derivative, +/- indicate from which side of a point the derivative is considered)

$$\forall t \in \Re, P(-t) = P(+t) \text{ and}$$
$$\exists t \in \Re, P'(-t) \neq P'(+t)$$
$$\text{then } C^0 \text{ and no higher}$$



$$C^0 \text{ and } \forall t \in \Re, P'(-t) = P'(+t)$$
$$\text{then } C^1$$



Continuity - Geometric

Geometric G^k-continuity

Independent of parameterization, although it is usually defined from parametric forms.

Directly from the geometry of the curve.

Unintuitive for k > 2

From Parametric form

$P^{(i)}(t-) = c_i P^{(i)}(t+)$ t in [a,b]

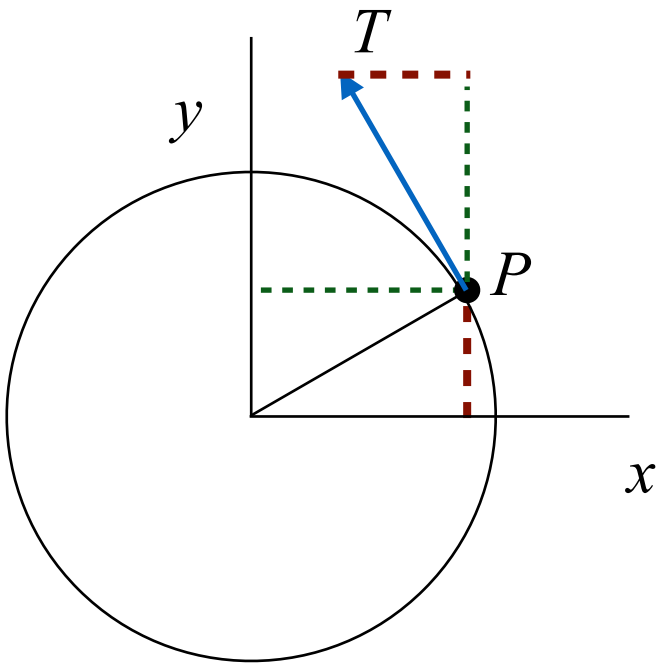
for i = 0,…,k and

for some c_i positive constants

($P^{(i)}$ is the i-th derivative)

Is a C^k-continuous function G^K continuous as well?

Example



Circle centred at the origin, radius R

Counter-clockwise normal to *P*

$$P = (x, y) \rightarrow T = (-y, x)$$

From parametric form

$$(x = R\cos(\theta), \quad y = R\sin(\theta)) \quad \theta \in [0, 2\pi]$$

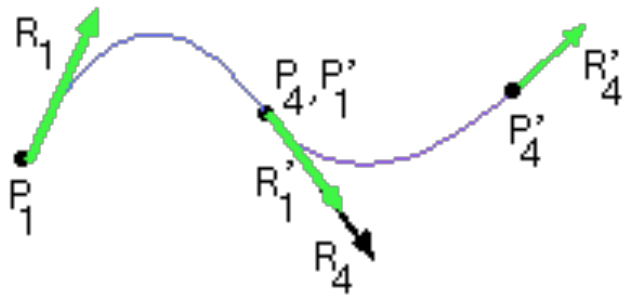
$$P = (R\cos(\theta), R\sin(\theta)) \rightarrow P' = (-R\sin(\theta), R\cos(\theta))$$

Parametric vs Geometric

There are cases were a curve is C1 continuous but not G1 continuous.

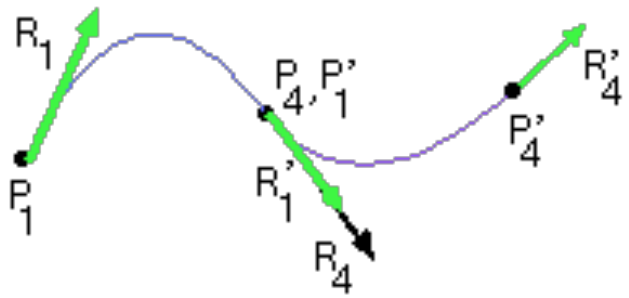
For example consider a parameterization that follows the drawing of a curve. The pen stops at a point, stays for a few seconds, and then starts in a different direction. The parametric derivatives at the joints may both be 0 i.e equal. However the tangent vectors may not be aligned. The parametric continuity is an artifact of the specific parametrization in this case.

Piecewise Cubic Hermite Curves



What are the conditions for G1 continuity?

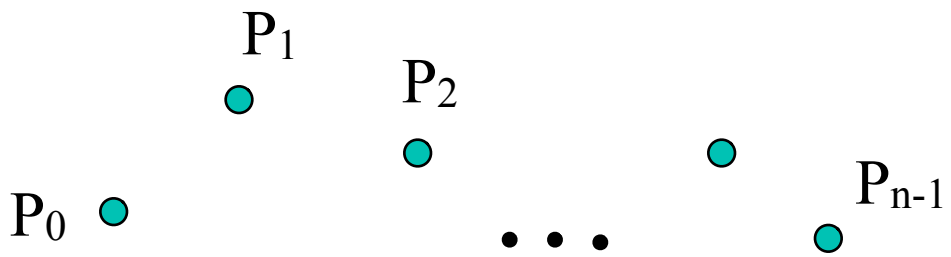
Piecewise Cubic Hermite Curves



R’1 = kR4

P1’= P4

Catmull-Rom splines



Fit a piece wise cubic curve to the points

- P_i, i $= 0,1,\ldots,n\text{-}1$

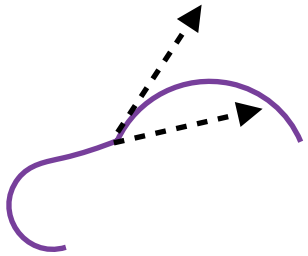
Setting Parameters for Hermite Splines

Points and tangents

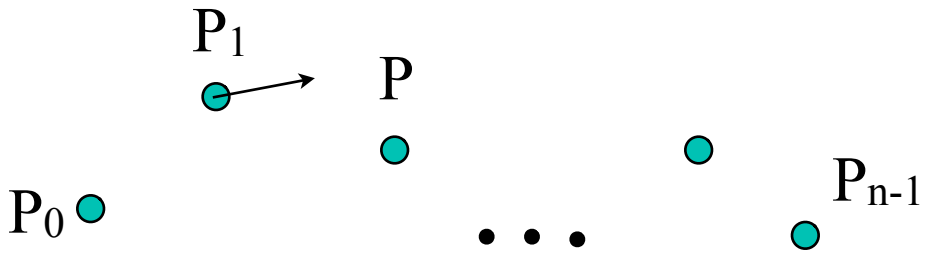
Usually just specify one slope at each knot

But a useful capability: use a different slope on each side of a knot

- We break C^1 smoothness, but gain control
- Can create motions that abruptly change, like collisions



Catmull-Rom Splines

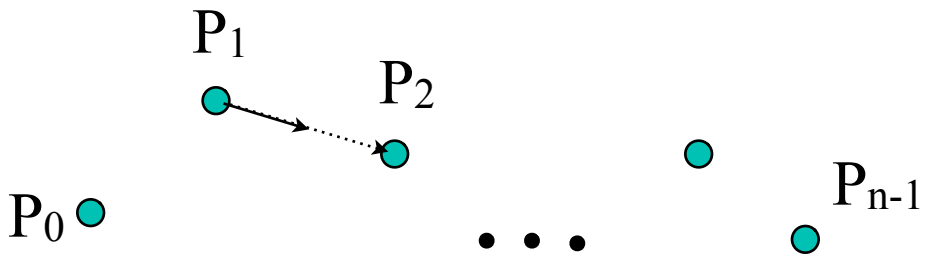


Fit a C1-continuous piece wise Hermite curve to the points, with automatically computed tangents

- P_i, $i = 0,1,\ldots,n-1$

How do we compute the tangents?

Catmull-Rom Splines



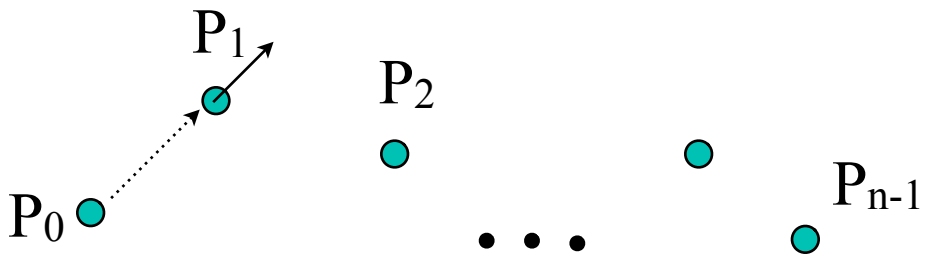
Fit a C1-continuous piece wise hermite curve to the points, y_i with automatically computed tangents

- P_i, i = 0,1,…,n-1

First order forward difference tangents, s_i

- $s_i = P_{i+1} - P_i$, for i = 0,…,n-2
- $s_{n-1} = P_{n-1} - P_{n-2}$ or 0? How does the curve look?

Catmull-Rom Splines



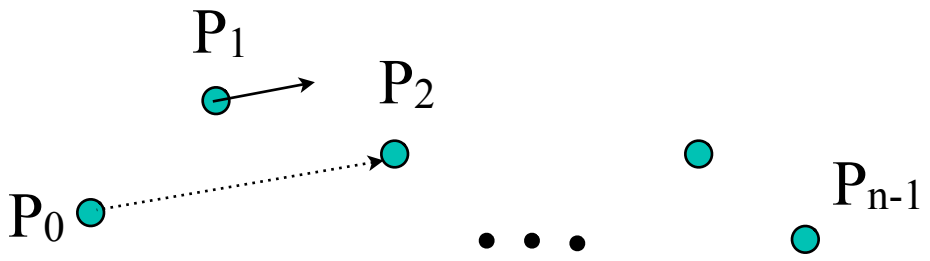
Fit a C1-continuous piece wise hermite curve to the points, y_i with automatically computed tangents

- P_i, i = 0,1,…,n-1

First order backward difference tangents, s_i

- $s_i = P_{i+1} - P_i$, for i = 0,…,n-2
- $s_{n-1} = P_{n-1} - P_{n-2}$ or 0? How does the curve look?

Catmull-Rom Splines



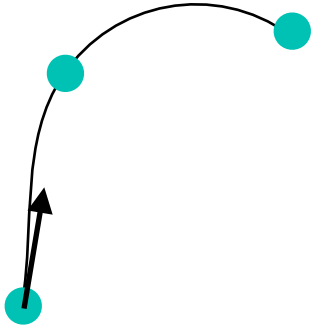
Fit a C1-continuous piece wise hermite curve to the points, y_i with automatically computed tangents

- P_i, i = 0,1,…,n-1

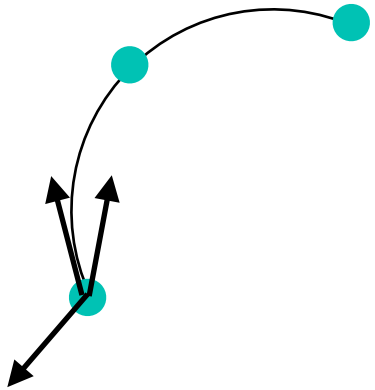
Second order accurate tangents, s_i

- $s_i = (P_{i+1} - P_{i-1})/2.0$ for i = 1,…,n-2
- $s_0 = 2(P_1 - P_0) - (P_2-P_0)/2$, similarly for s_{n-1}

Comparison



First order accurate
first tangent



Second order accurate
first tangent

(Magnitudes not accurate)

Other forms of tangent control

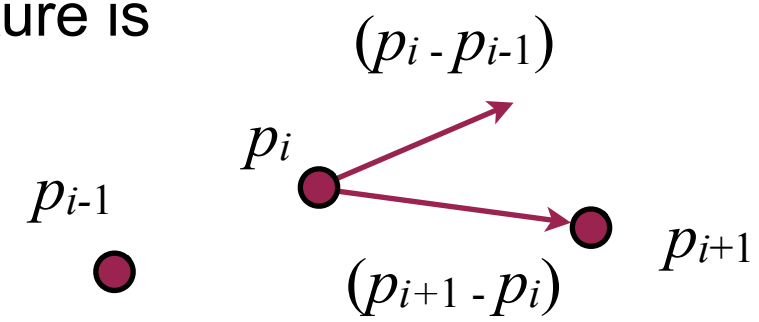
Kochanek-Bartels splines

- For each point p_i we define tangents s_i^-, s_i^+ as follows

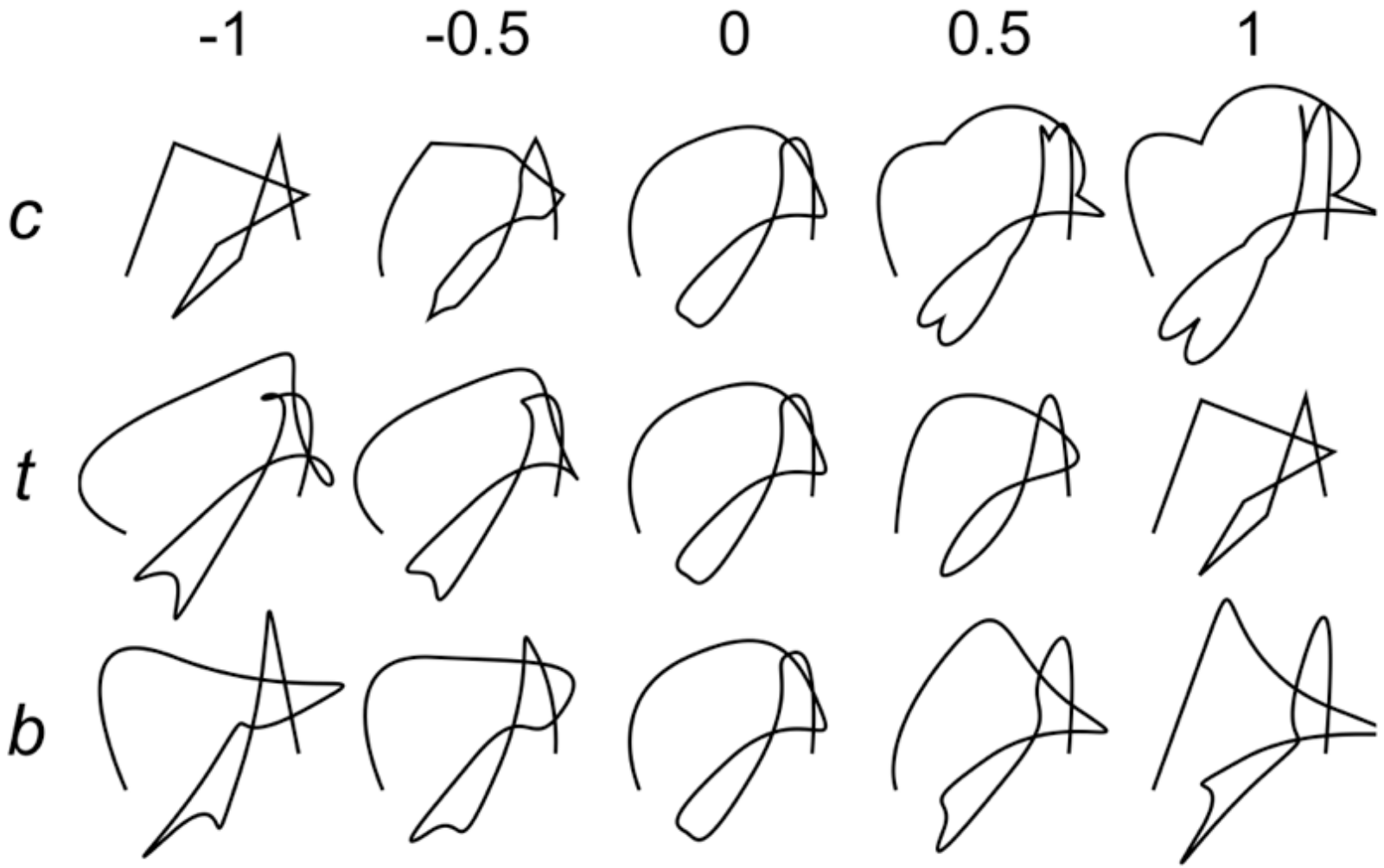
$$s_i^- = \frac{(1-t)(1+b)(1+c)}{2}(p_i - p_{i-1}) + \frac{(1-t)(1-b)(1-c)}{2}(p_{i+1} - p_i)$$

$$s_i^+ = \frac{(1-t)(1+b)(1-c)}{2}(p_i - p_{i-1}) + \frac{(1-t)(1-b)(1+c)}{2}(p_{i+1} - p_i)$$

- Each tangent is mixture of the *backward and* and *forward* tangents. The mixture is controlled by the parameters:
  t: tension
  b: bias
  c: continuity

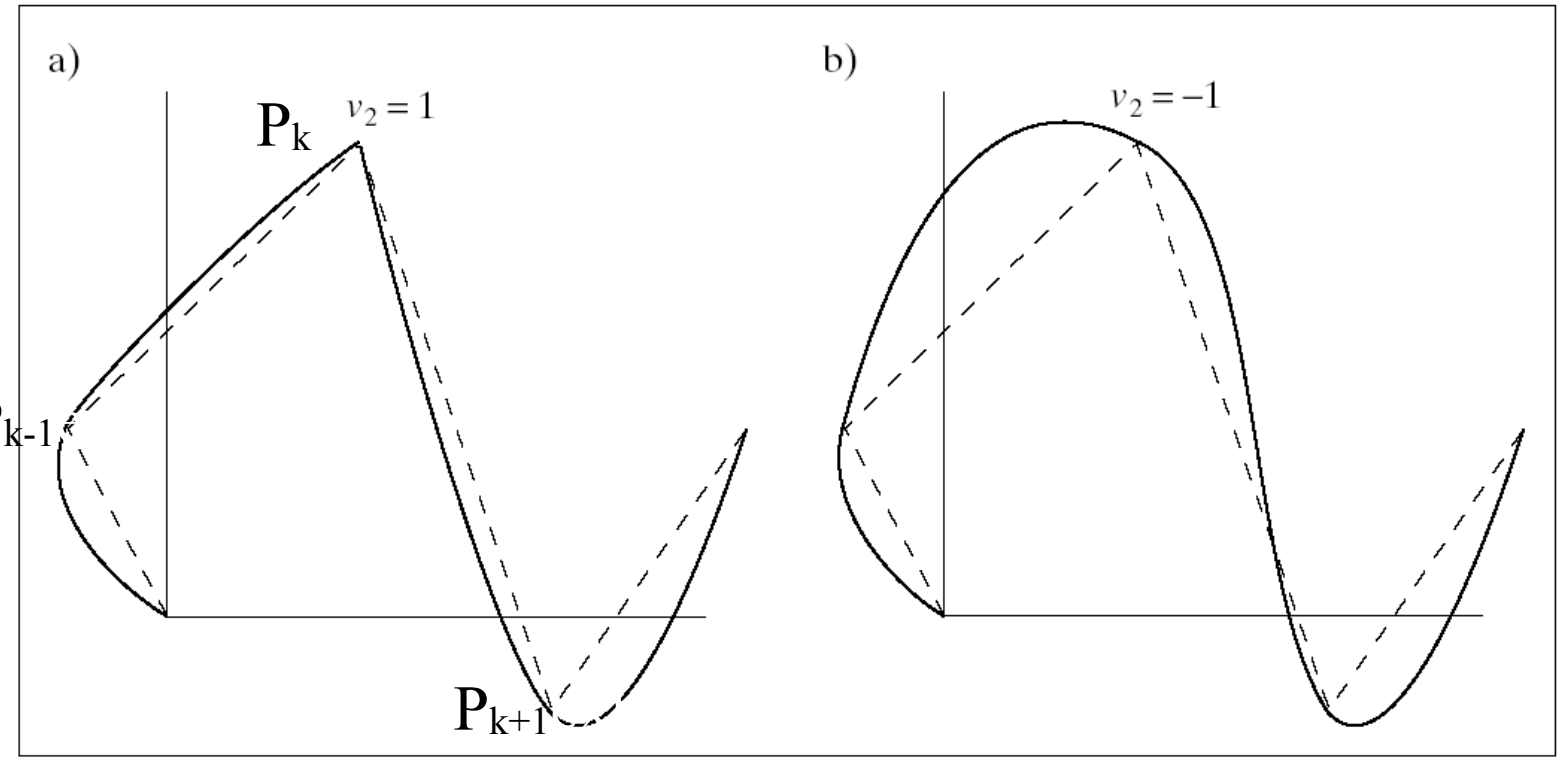


Example from Wikipedia



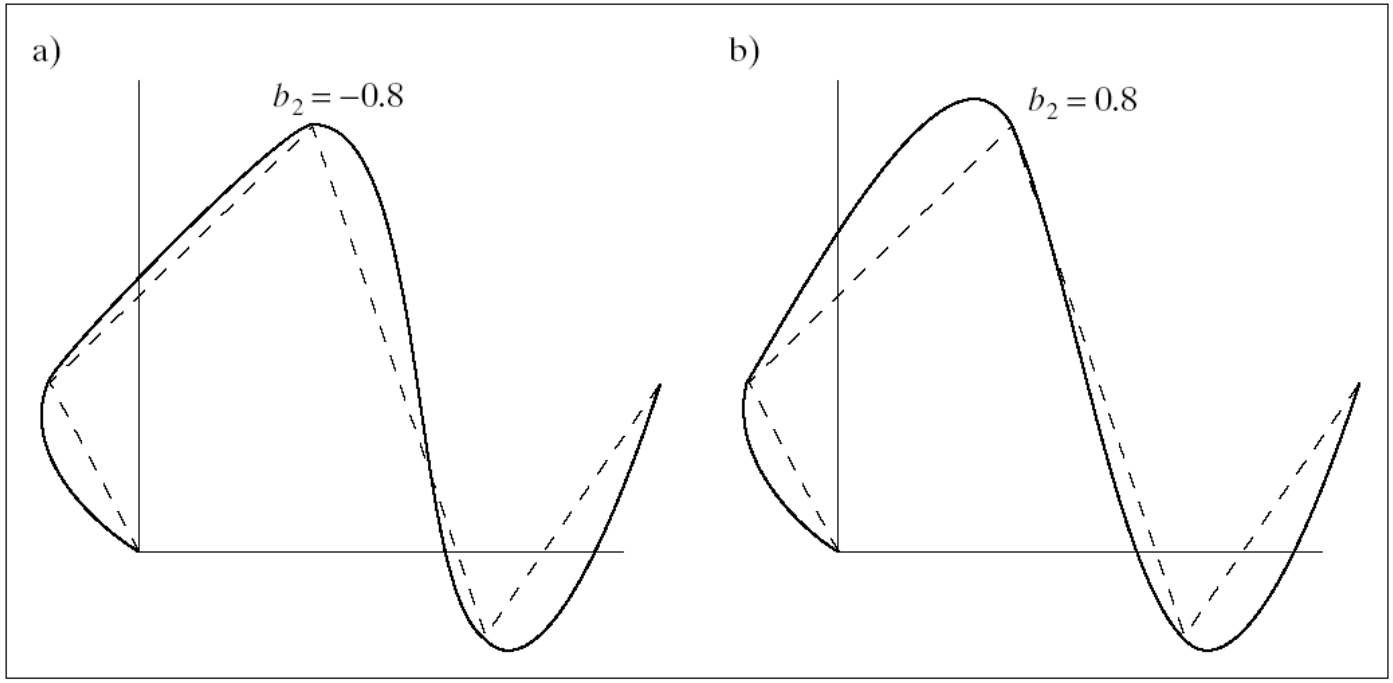
Separate examples: Adding tension control

$S_k^+ = S_k^- = \frac{1}{2}(1-v_k)(P_k - P_{k-1}) + \frac{1}{2}(1-v_k)(P_{k+1} - P_k)$



Adding bias control

$$S_k^+ = S_k^- = \tfrac{1}{2}(1-b_k)(P_k - P_{k-1}) + \tfrac{1}{2}(1+b_k)(P_{k+1} - P_k)$$

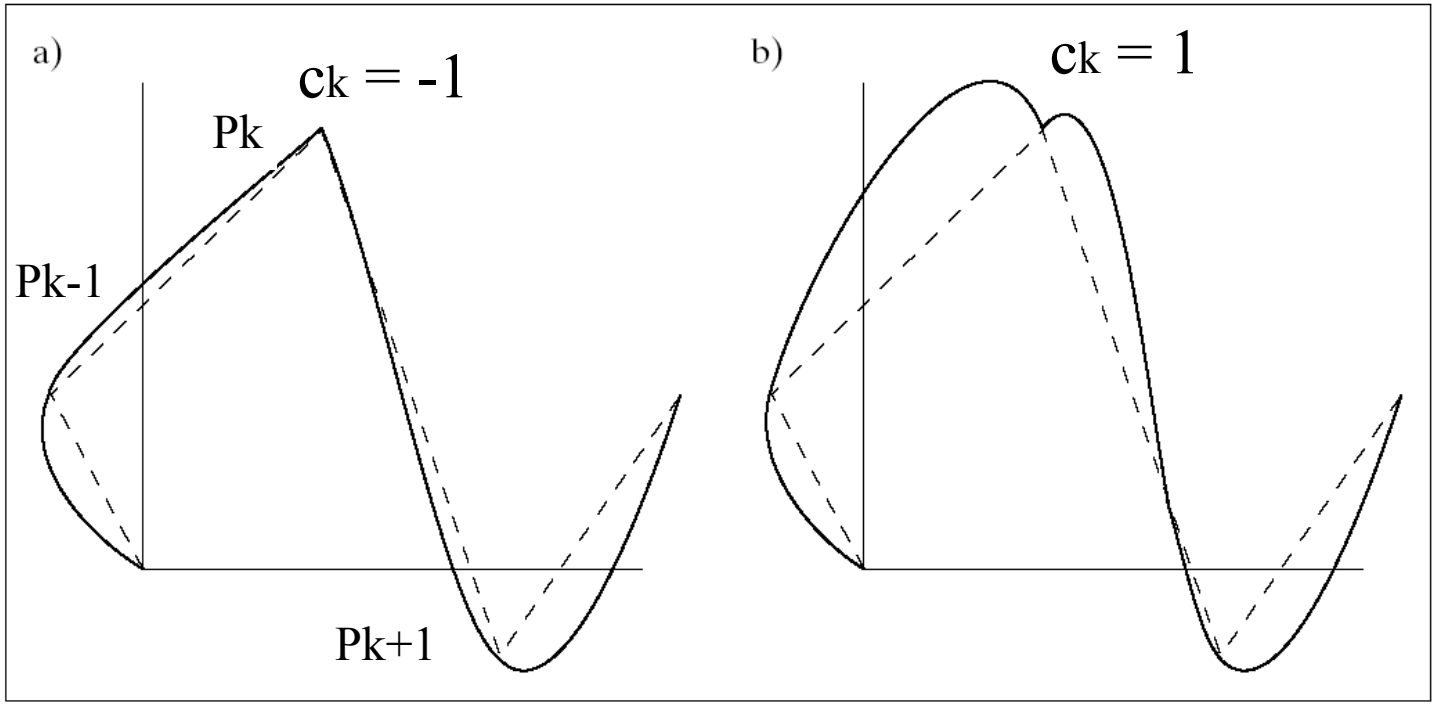


Adding continuity control

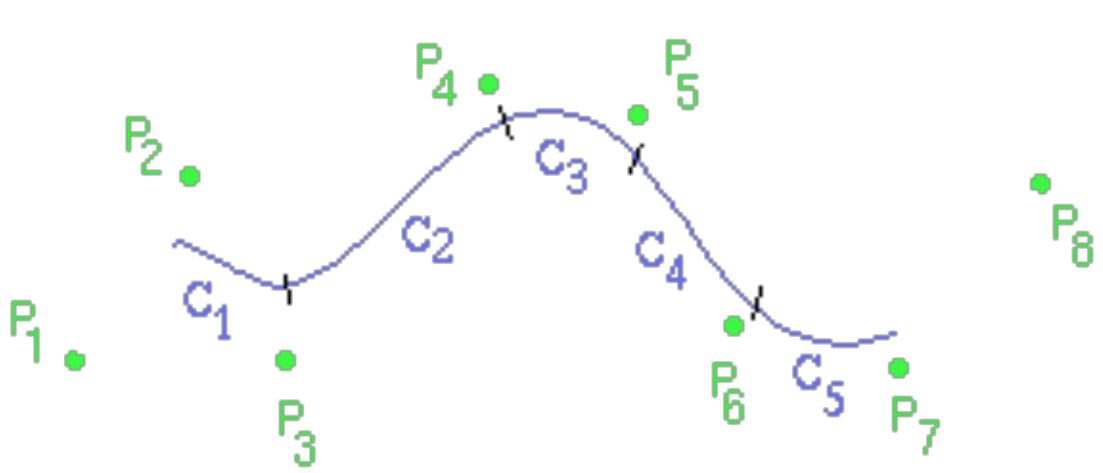
Two distinct derivatives at each point (before “-“ and after “+”)

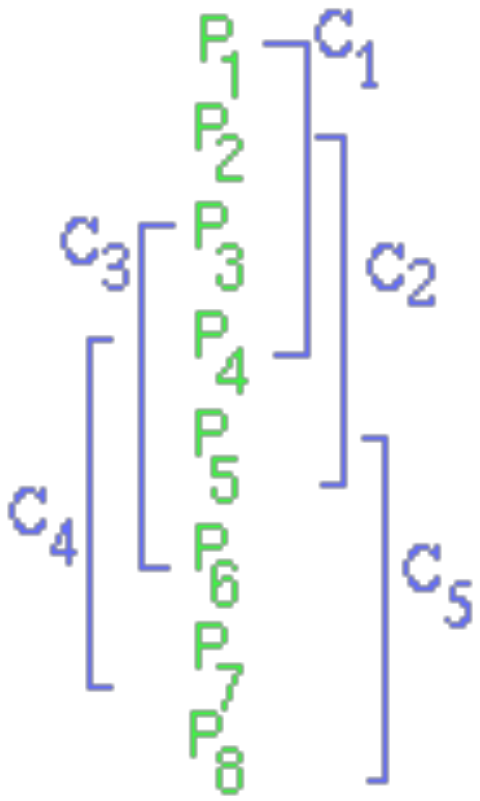
$$S_k^- = \frac{1}{2}(1 - c_k)(P_k - P_{k-1}) + \frac{1}{2}(1 + c_k)(P_{k+1} - P_k)$$

$$S_k^+ = \frac{1}{2}(1 + c_k)(P_k - P_{k-1}) + \frac{1}{2}(1 - c_k)(P_{k+1} - P_k)$$



Uniform BSplines





Matrix form

For a bspline curve with:

- m+1 control points P_0, …, P_m
- m-2 segments Q_3,…, Q_m
- t in [3,…,m]

$$Q_i(t) = \begin{bmatrix} (t-t_i)^3 & (t-t_i)^2 & (t-t_i) & 1 \end{bmatrix} \mathbf{M}_{bspline} \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}$$

Properties

C2 continuous

Convex hull property

NO invariace under perspective projection!

NURBS: Nonuniform Rational B-splines

$$X(t) = X(t) / W(t)$$

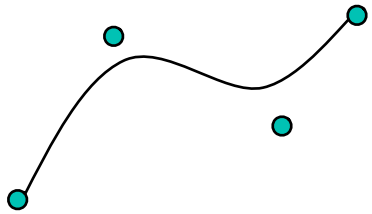
$$Y(t) = Y(t) / W(t)$$

$$Z(t) = Z(t) / W(t)$$

- Exact conic sections
- Invariance under perspective projection

Summary: General problem

$P_0, \ldots, P_L \rightarrow$ Curve generation $\rightarrow P(t)$



$$P(t) = \sum_{i=0}^{L} B_i(t) P_i, \quad t \in [a, b], \quad a, b \in \Re$$

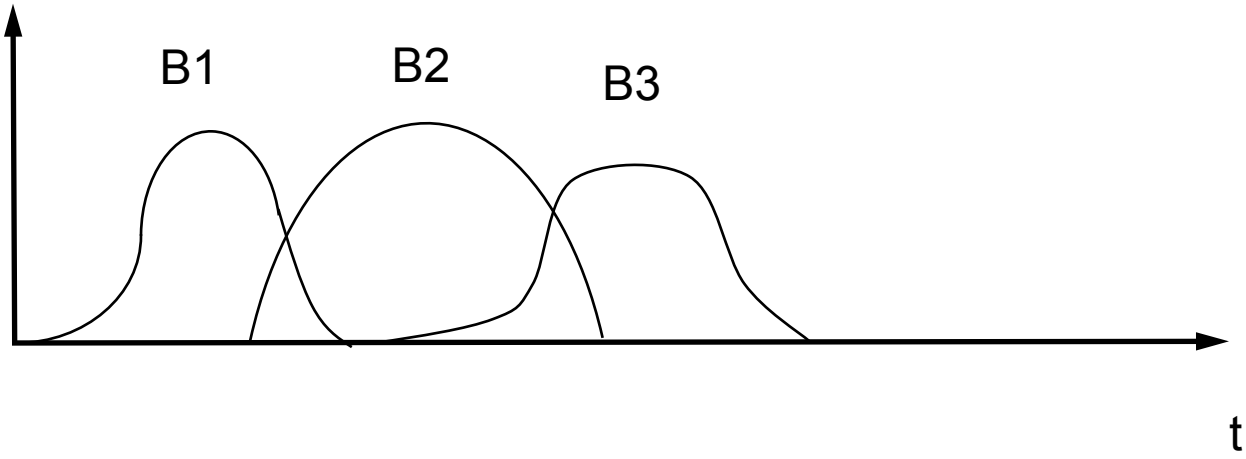
where

$B_i(t)$: Blending functions

$P_i, i = 1, \cdots, L$: Control Points

Blending functions

Weight the influence of each constraint (e.g. control point) on the curve created.



Wish list for blending functions

- Easy to compute and stable
- Sum to unity for every t in [a,b]
- Support over portion of [a,b]
- Interpolate certain control points
- Sufficient smoothness

Example: Bezier curves

- Sum up to unity
- Smooth
- Interpolate first and last
- Expensive to compute for large L
- No local control

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

$$B_k^L(t) = \begin{pmatrix} L \\ K \end{pmatrix} (1-t)^{L-k} t^k$$

$$\begin{pmatrix} L \\ K \end{pmatrix} = \frac{L!}{k!(L-k)!}, \quad \text{for} \quad L \geq k$$

$$\text{Affine combination:} \sum_{k=0}^{L} B_k^L(t) = 1, \quad \text{for all} \quad t$$

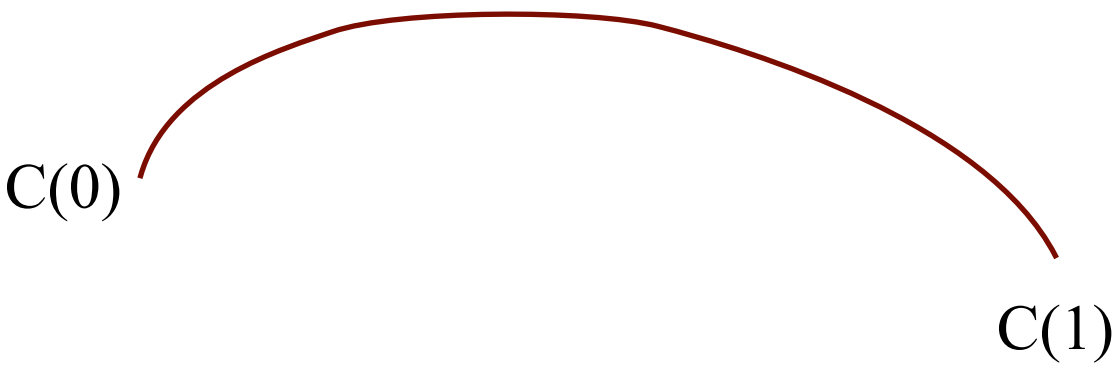
Rendering parametric curves

Transform into primitives we know how to handle

- Line segments

Single cubic segment

C(t): t in [0,1]



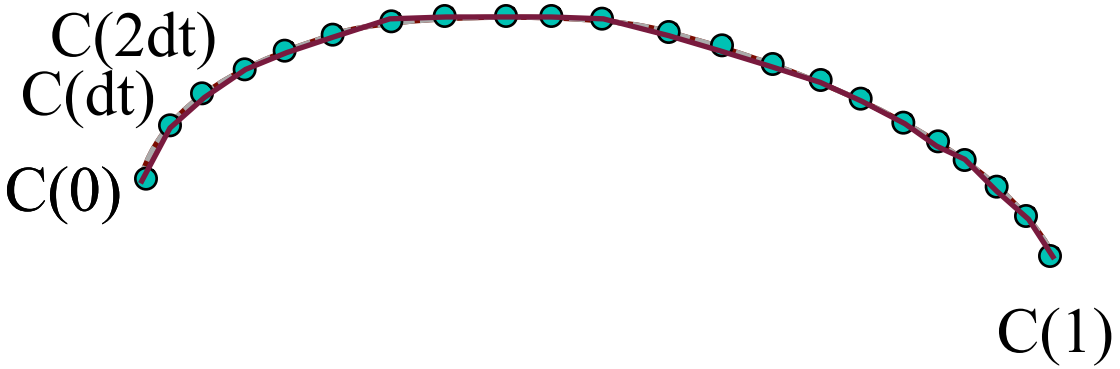
Converting to Lines

Straightforward Uniform subdivision

Evaluate $C(t)$ at n points spaced by dt=1/(n-1)

$$C(0), C(dt), C(2dt),\ldots,C((n-1)*dt=1)$$

Draw as lines



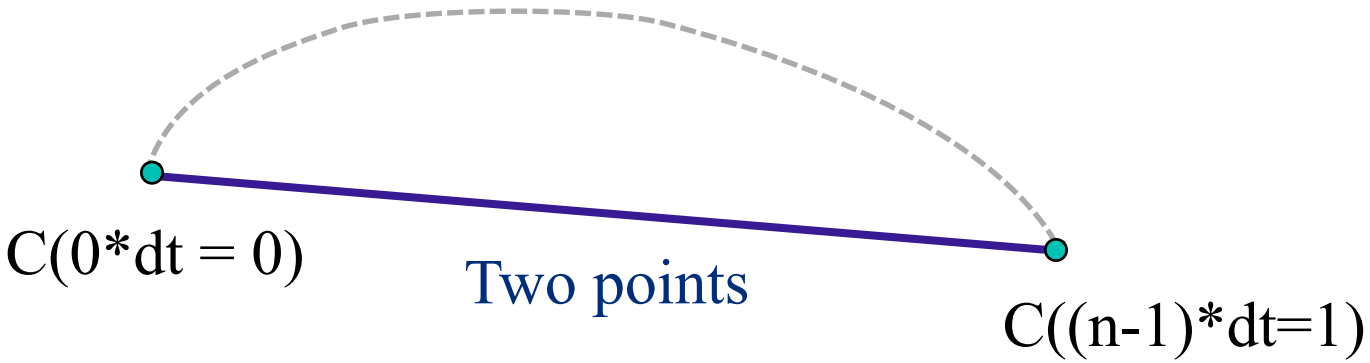
Converting to Lines

Straightforward Uniform subdivision

Evaluate $C(t)$ at n points spaced by dt=1/(n-1)

$$C(0), C(dt), C(2dt),\ldots,C((n-1)*dt=1)$$

Here n = 2 and dt = 1/(n-1) = 1



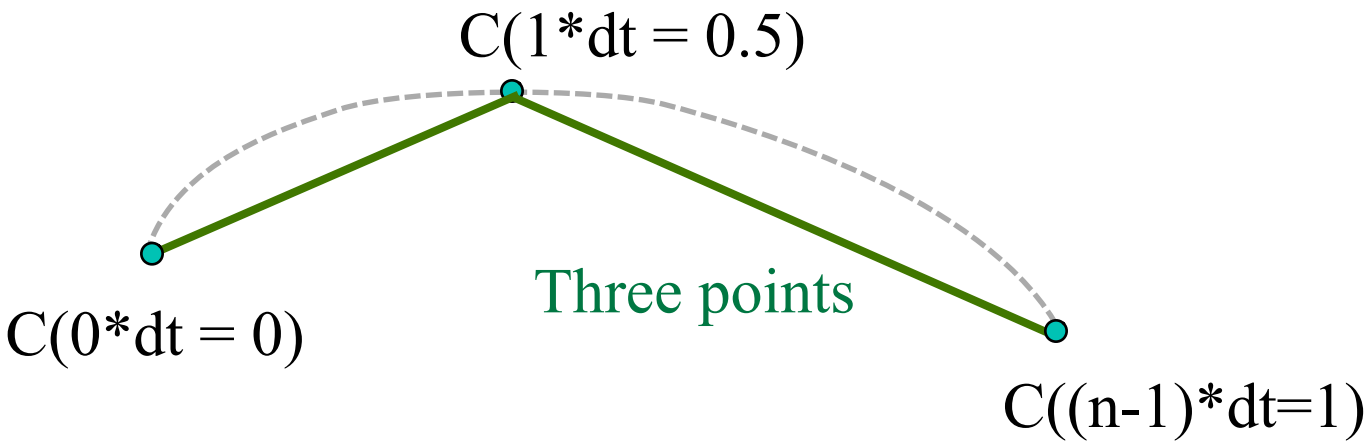
Converting to Lines

Straightforward Uniform subdivision

Evaluate $C(t)$ at n points spaced by dt=1/(n-1)

$$C(0), C(dt), C(2dt),\ldots,C((n-1)*dt=1)$$

Here n = 3 and dt = 1/(n-1)= 0.5



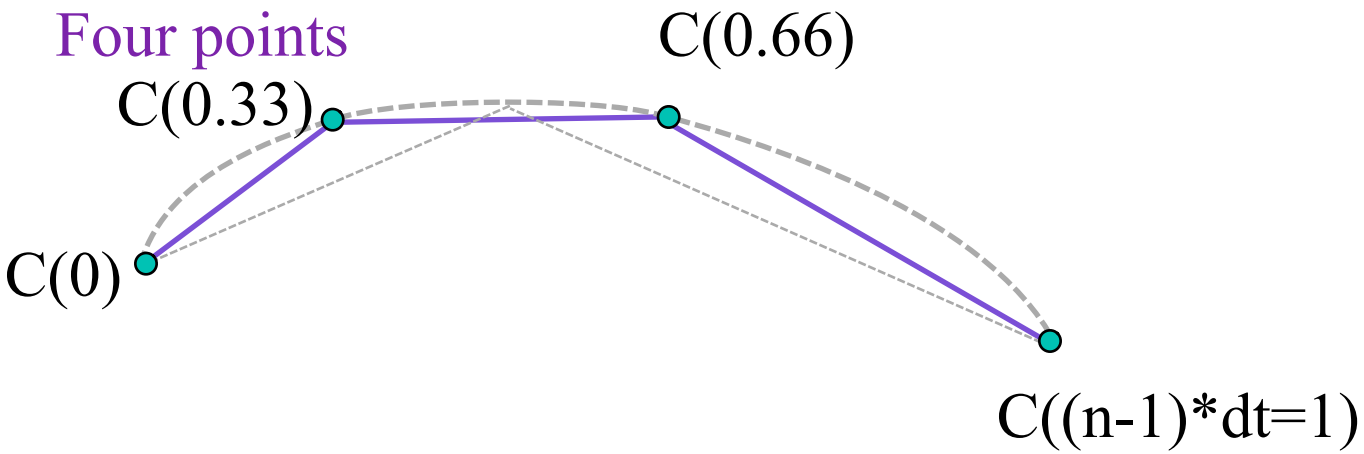
Converting to Lines

Straightforward Uniform subdivision

Evaluate $C(t)$ at n points spaced by dt=1/(n-1)

$$C(0), C(dt), C(2dt),\ldots,C((n-1)*dt=1)$$

Here n = 4 and dt = 1/(n-1) = 0.33



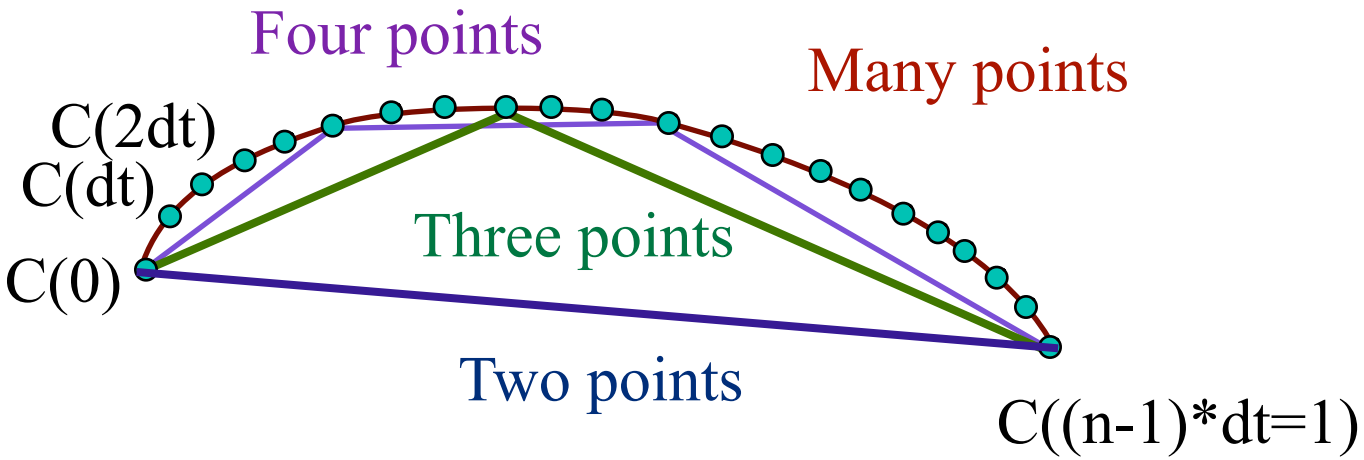
Converting to Lines

Straightforward Uniform subdivision

Evaluate $C(t)$ at n points spaced by dt=1/(n-1)

$$C(0), C(dt), C(2dt),\ldots,C((n-1)*dt=1)$$

All together for comparison

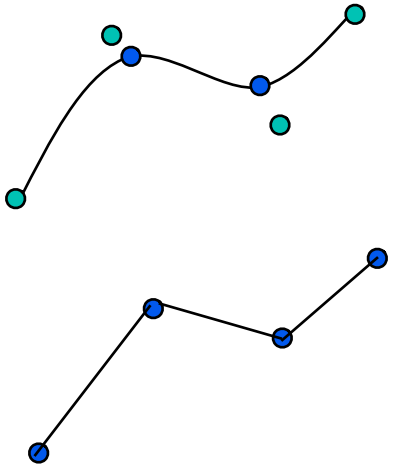


How many evaluation points are enough for Bezier curves?

Not too few

Not too many

Ok, how many?



Subdivision schemes

Uniform

Non-uniform

- Adaptive

Error metrics that define how much information we are losing

An example...

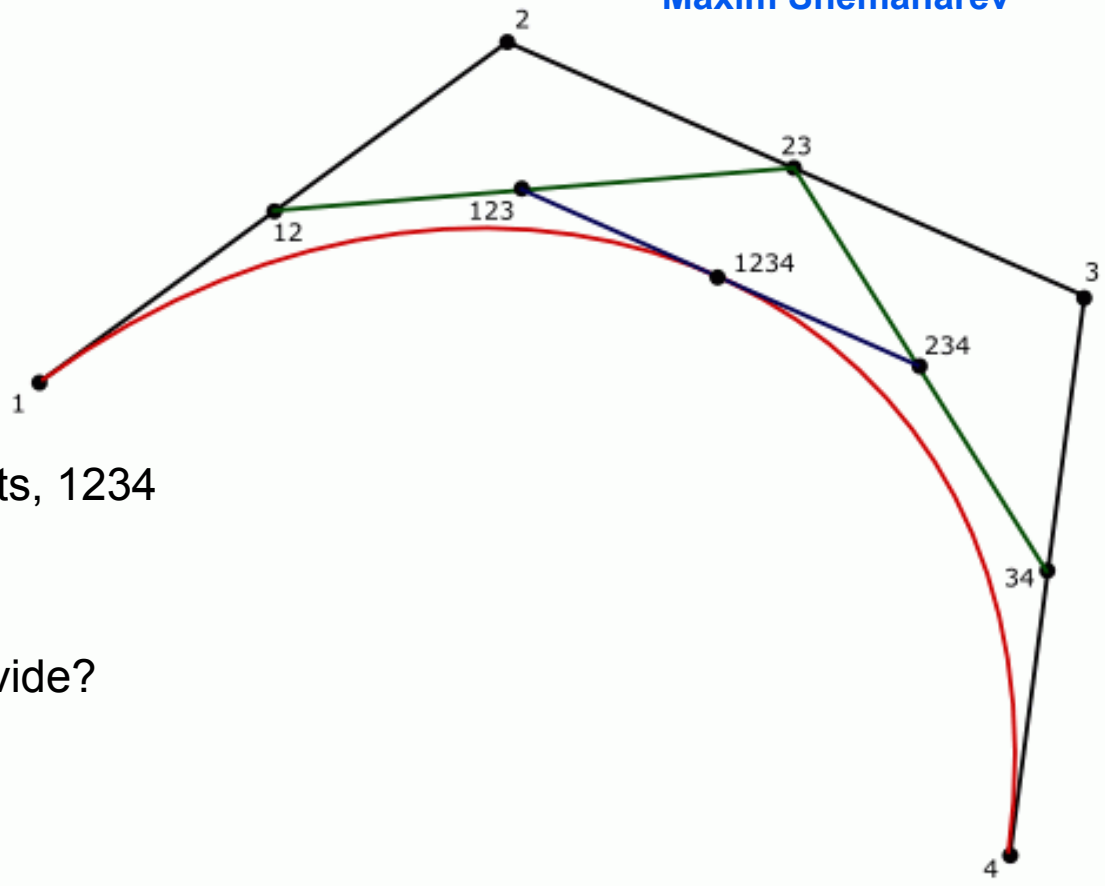
Adaptive Subdivision of Bezier Curves

de Casteljau subdivision

One Bezier curve
becomes 2 flatter
curves

Original points 1,2,3,4 →
Midpoints 12, 23, 34
Midpoints of midpoints: 123, 234
Midpoints of midpoints of midpoints, 1234
Remember: tweening for t = 0.5
Can chose any t we want
Ok, how many times do we subdivide?

Images courtesy of
Maxim Shemanarev

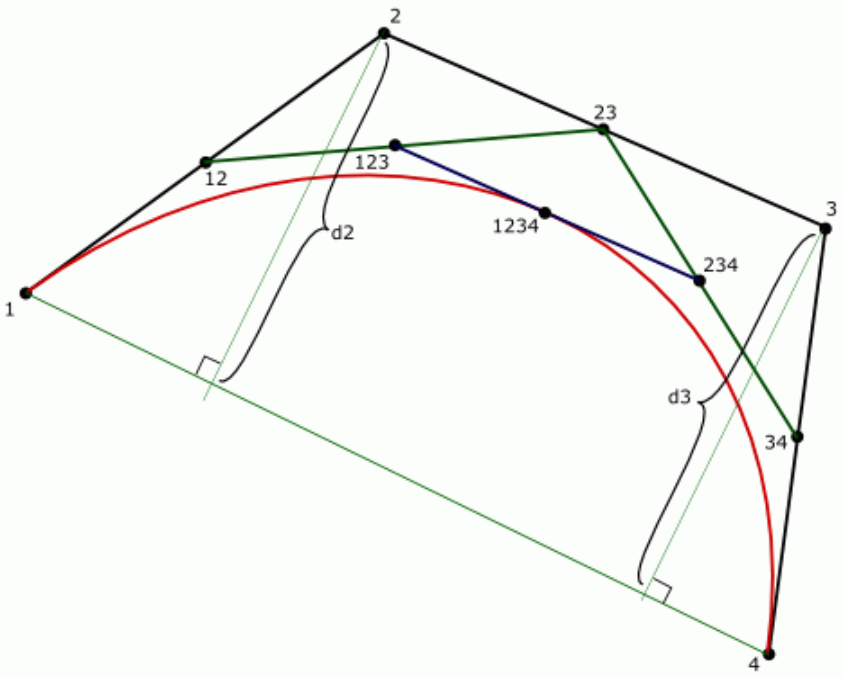


Error metrics

Images courtesy of
Maxim Shemanarev

Examples:

Point distance



Tangent distance