Smooth Curves

Dr. Scott Schaefer

TAMU Computer Science

Smooth Curves

- Interpolation
  - Interpolation through Linear Algebra
  - Lagrange interpolation
- Bezier curves
- B-spline curves

Smooth Curves

- How do we create smooth curves?

Smooth Curves

- How do we create smooth curves?

- Parametric curves with polynomials

$$p(t) = \left(x(t), y(t)\right)$$

Smooth Curves

- Controlling the shape of the curve

$$x(t) = a + bt + ct^2 + dt^3$$

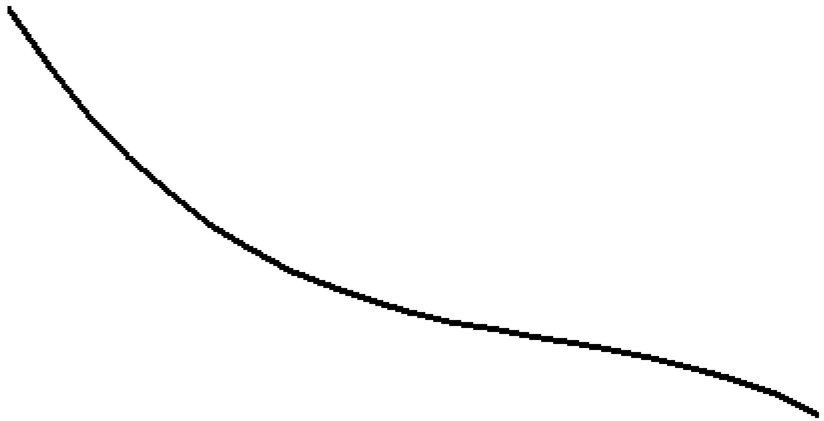
$$y(t) = e + ft + gt^2 + ht^3$$

Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

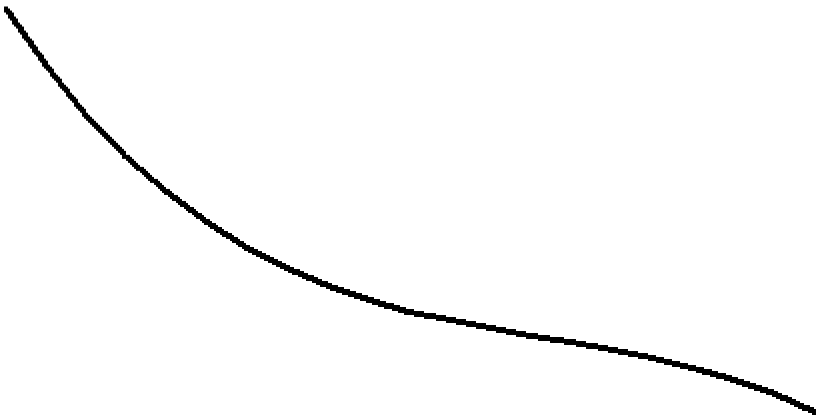


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

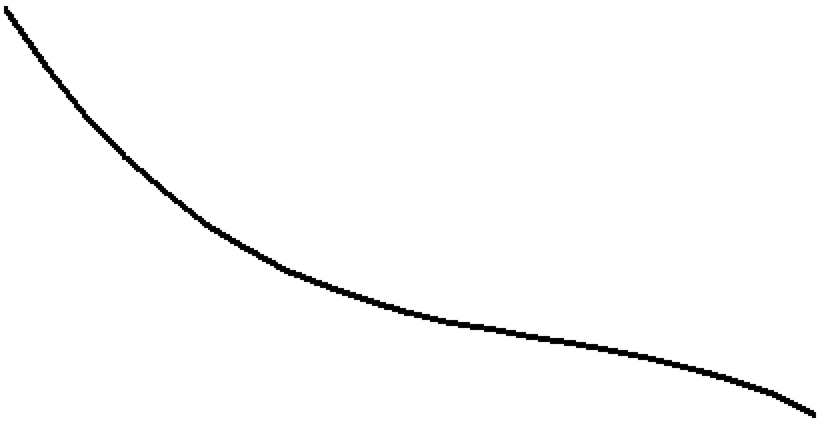


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

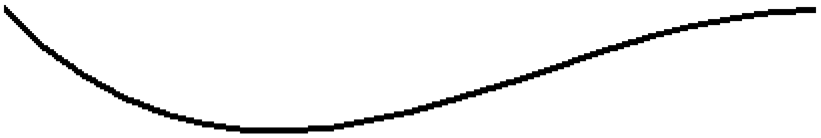


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

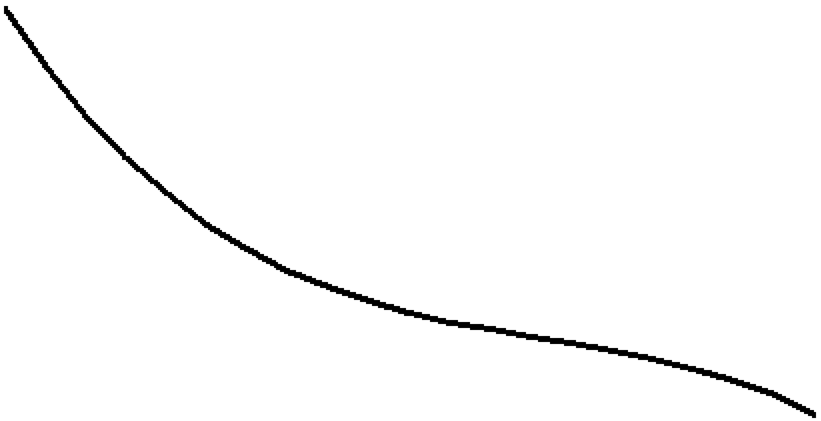


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

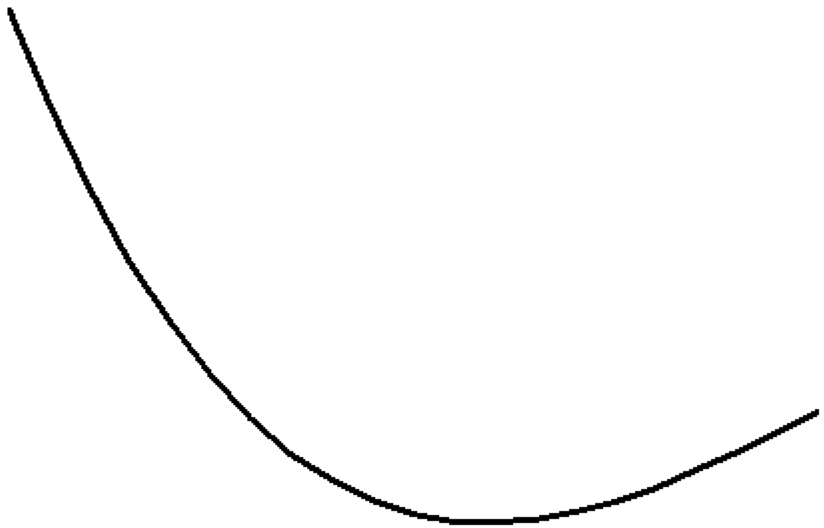


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

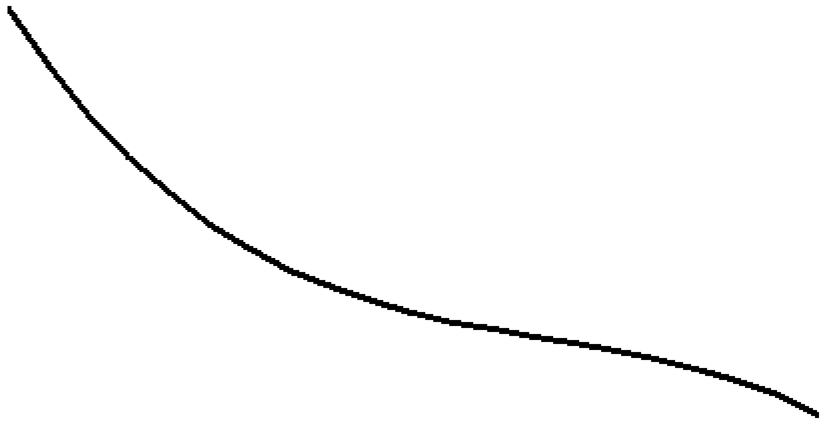


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

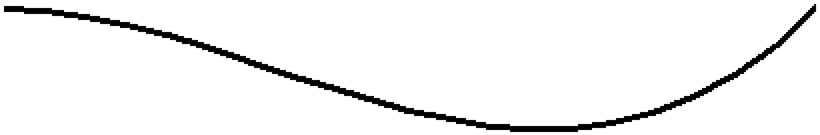


Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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



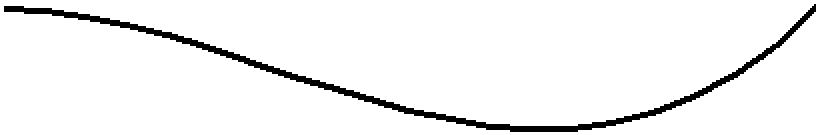
Smooth Curves

- Controlling the shape of the curve

$$x(t) = t$$

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

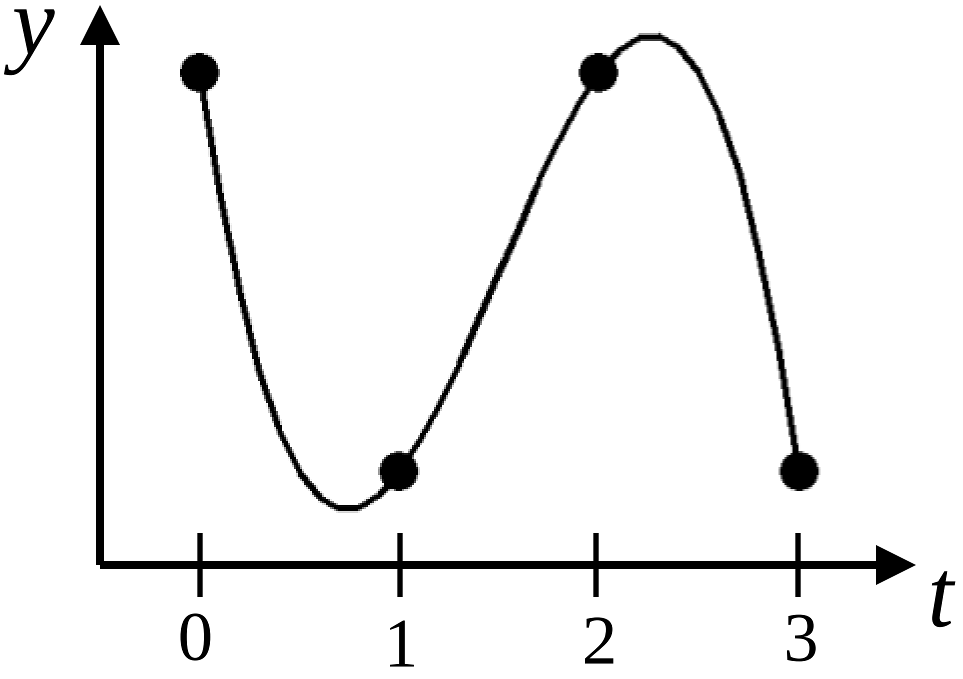
Power-basis coefficients not intuitive
for controlling shape of curve!!!



Interpolation

- Find a polynomial that passes through specified values

$$y(t) = a + bt + ct^2 + dt^3$$



Interpolation

- Find a polynomial that passes through specified values

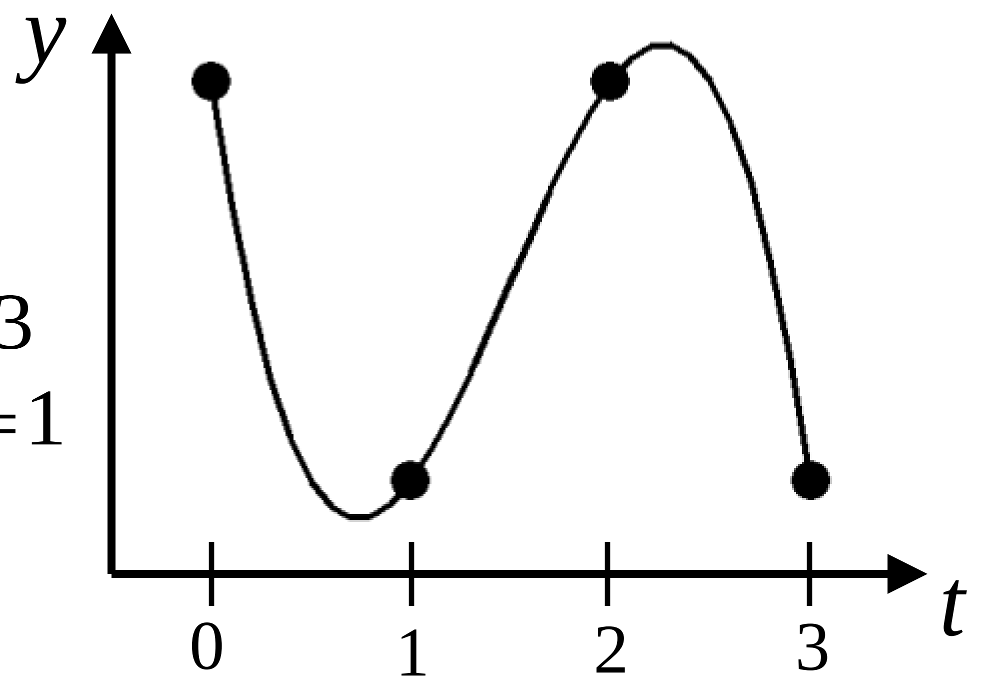
$$y(t) = a + bt + ct^2 + dt^3$$

$$y(0) = a = 3$$

$$y(1) = a + b + c + d = 1$$

$$y(2) = a + 2b + 4c + 8d = 3$$

$$y(3) = a + 3b + 9c + 27d = 1$$

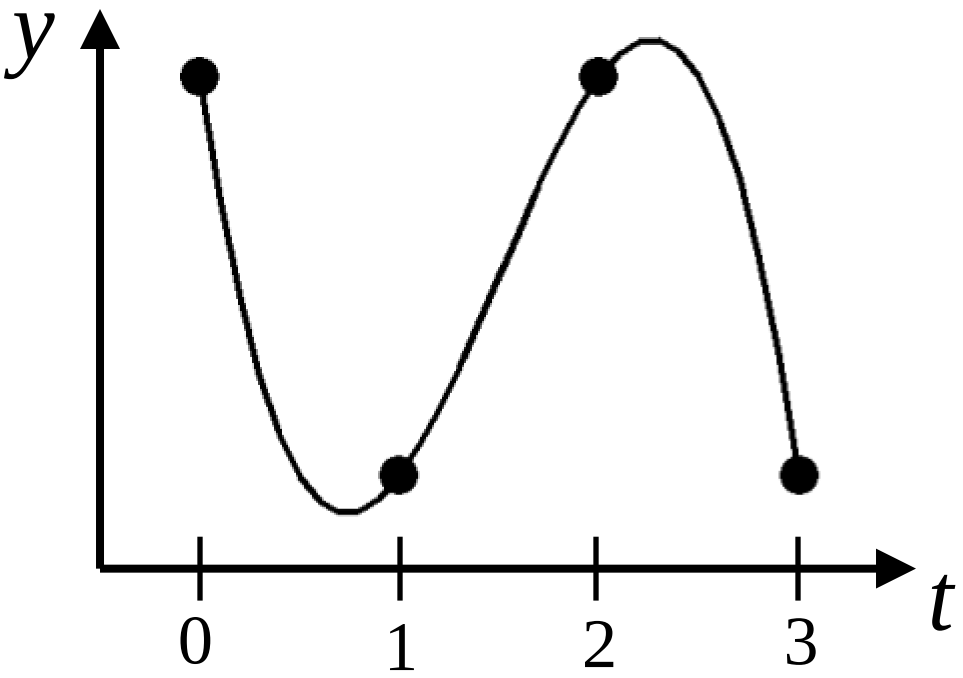


Interpolation

- Find a polynomial that passes through specified values

$$y(t) = a + bt + ct^2 + dt^3$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 3 \\ 1 \end{pmatrix}$$

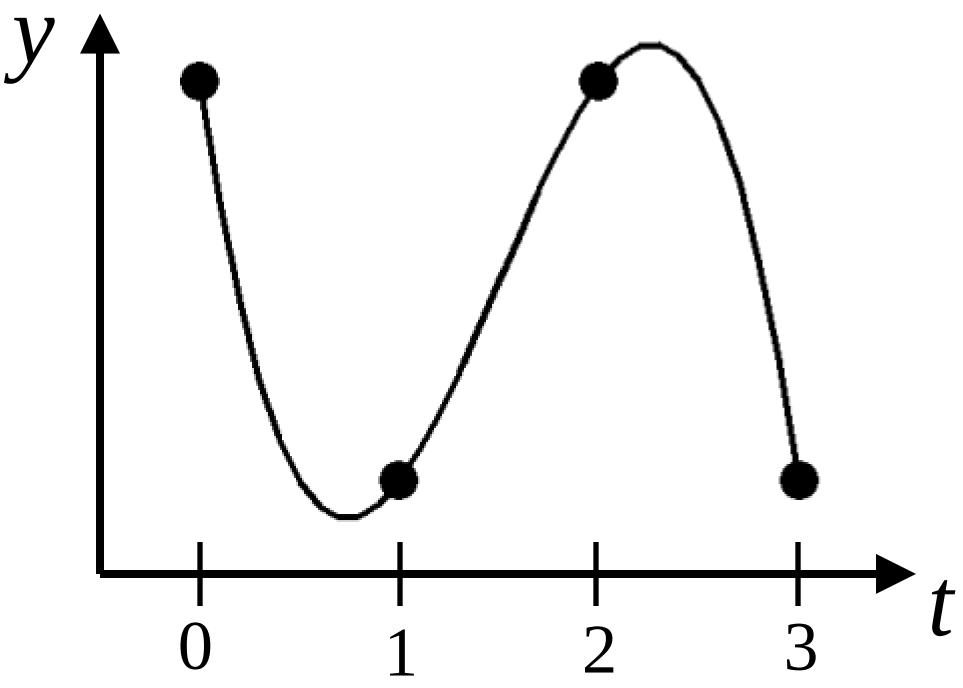


Interpolation

- Find a polynomial that passes through specified values

$$y(t) = a + bt + ct^2 + dt^3$$

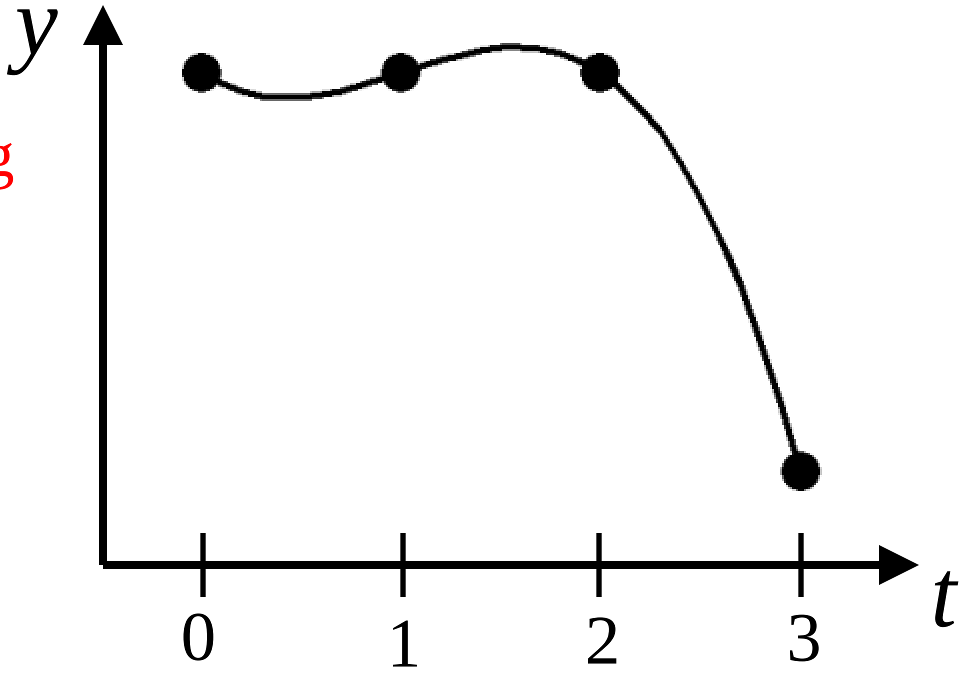
$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 3 \\ -20/3 \\ 6 \\ -1/3 \end{pmatrix}$$



Interpolation

- Find a polynomial that passes through specified values

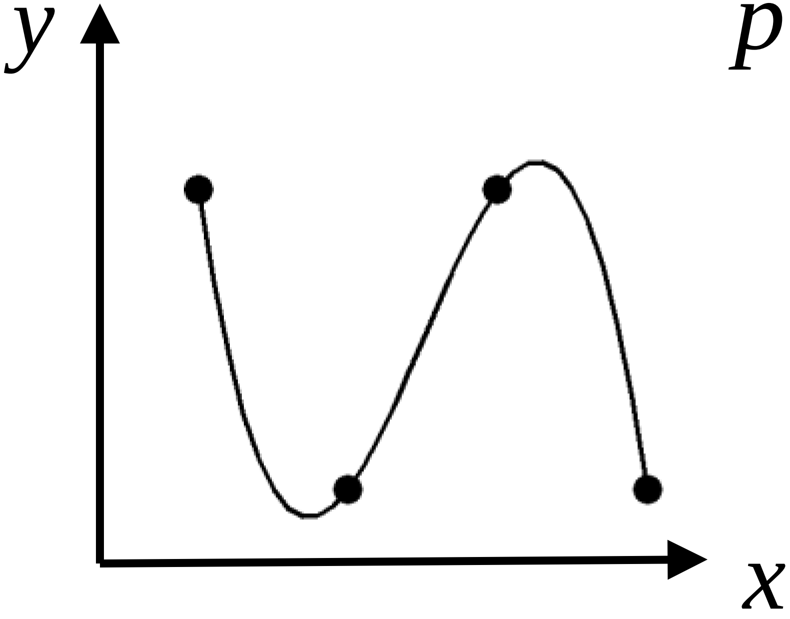
Intuitive control of curve using “control points”!!!



Interpolation

- Perform interpolation for each component separately
- Combine result to obtain parametric curve

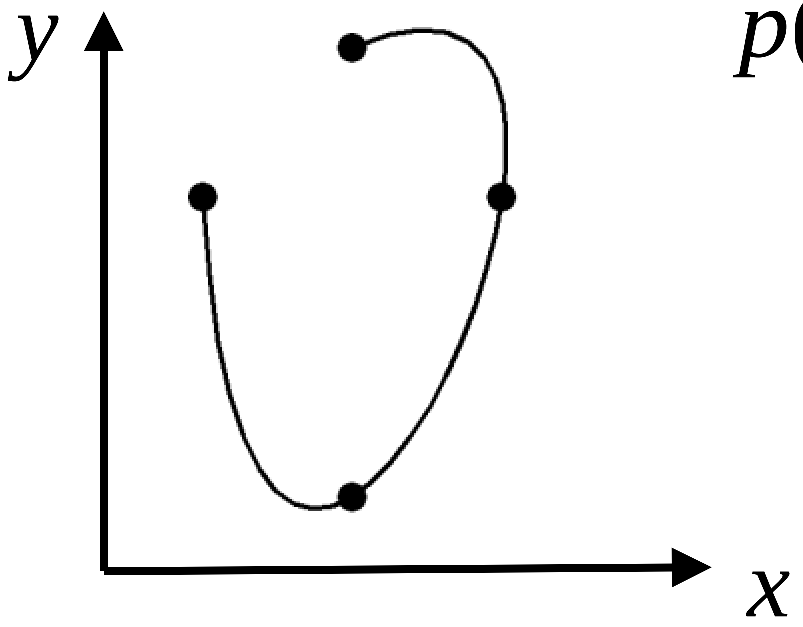
$$p(t) = \big(x(t), y(t)\big)$$



Interpolation

- Perform interpolation for each component separately
- Combine result to obtain parametric curve

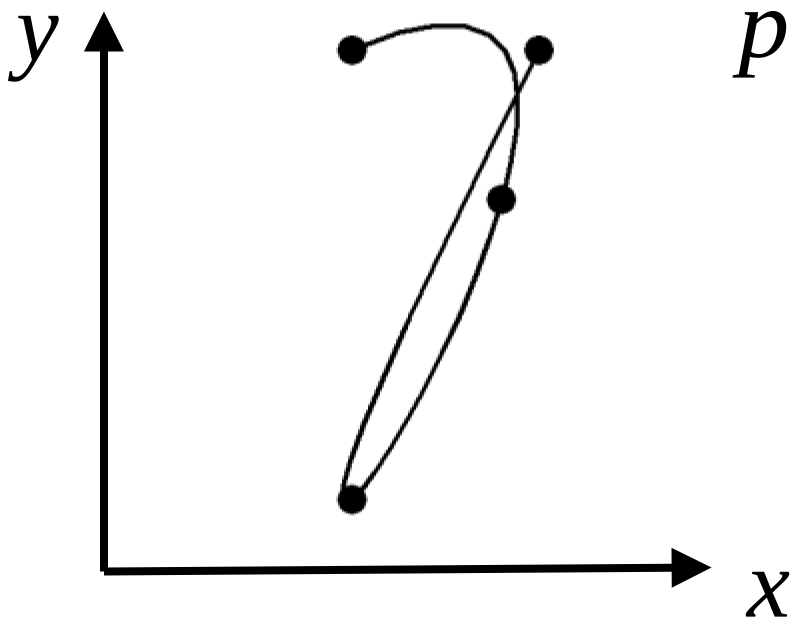
$$p(t) = \left(x(t), y(t)\right)$$



Interpolation

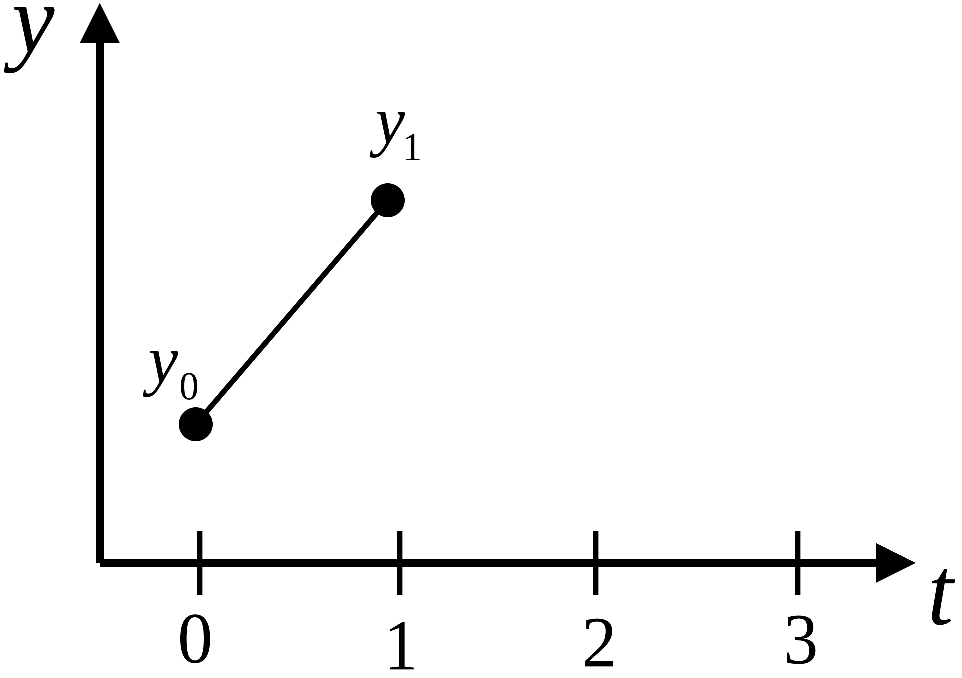
- Perform interpolation for each component separately
- Combine result to obtain parametric curve

$$p(t) = (x(t), y(t))$$



Lagrange Interpolation

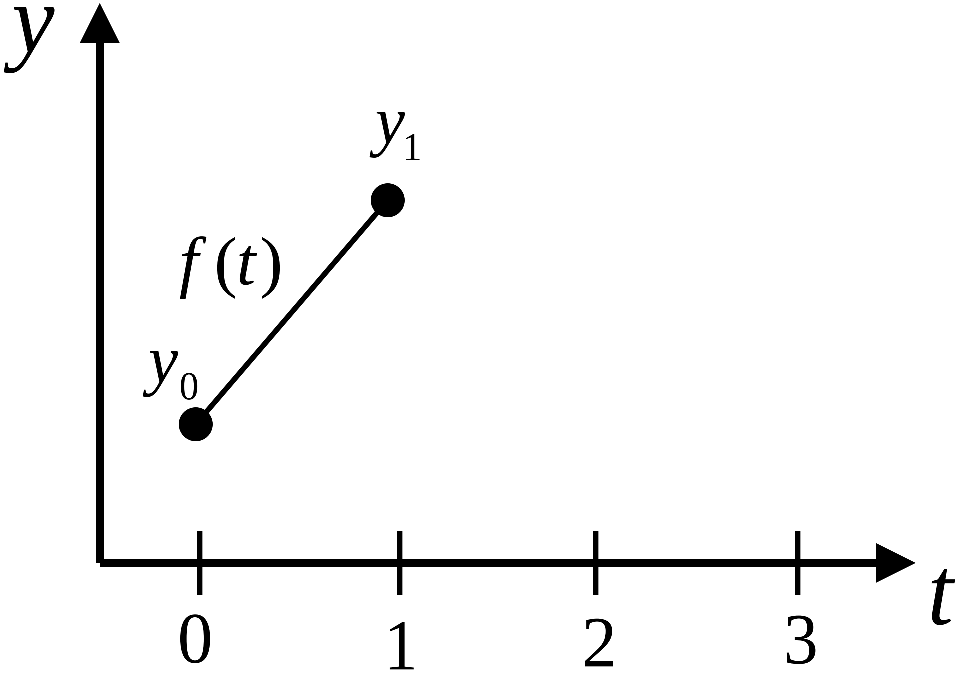
- Identical to matrix method but uses a geometric construction



Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

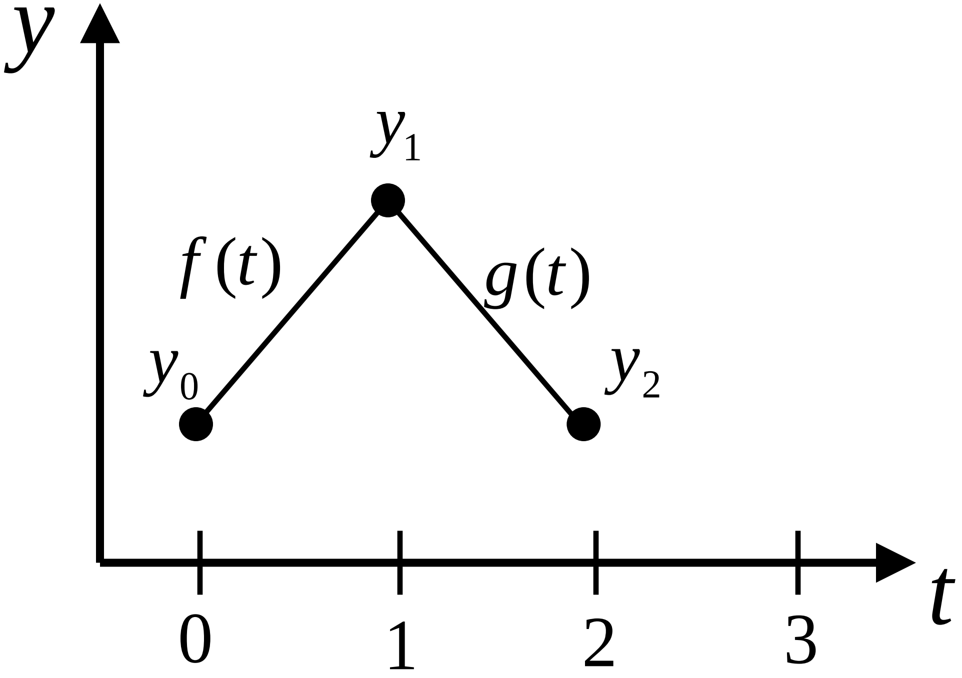
$$f(t) = (1-t)\,y_0 + t\,y_1$$



Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)y_0 + t\,y_1$$
$$g(t) = (2-t)y_1 + (t-1)\,y_2$$

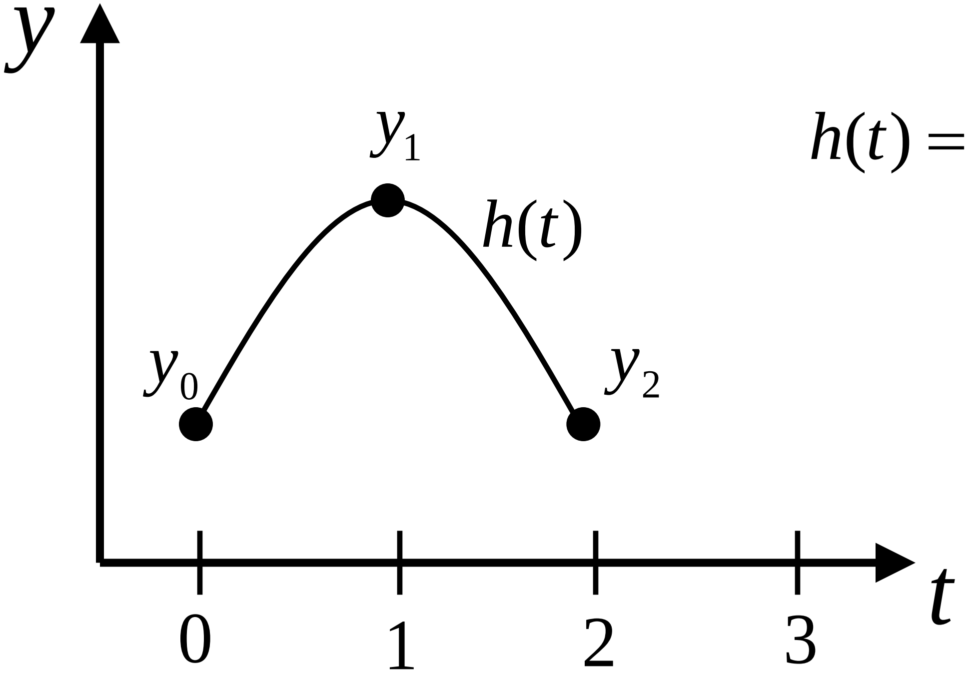


Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)\,y_0 + t\,y_1$$
$$g(t) = (2-t)\,y_1 + (t-1)\,y_2$$

$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

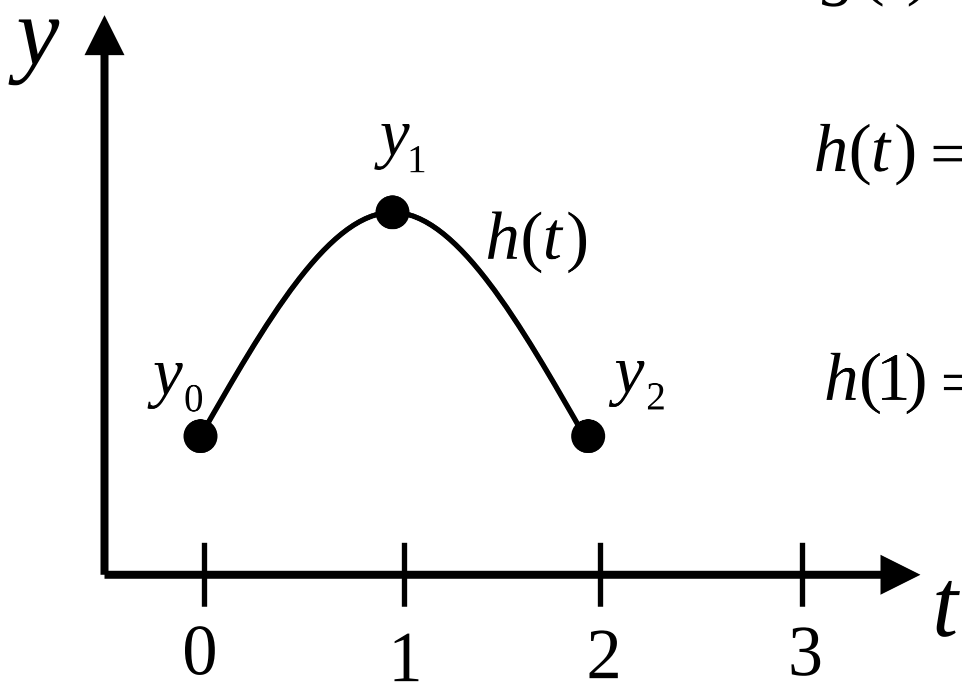


Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)\,y_0 + t\,y_1$$

$$g(t) = (2-t)\,y_1 + (t-1)\,y_2$$



$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(1) = \frac{(2-1)\,f(1) + 1\,g(1)}{2}$$

Lagrange Interpolation

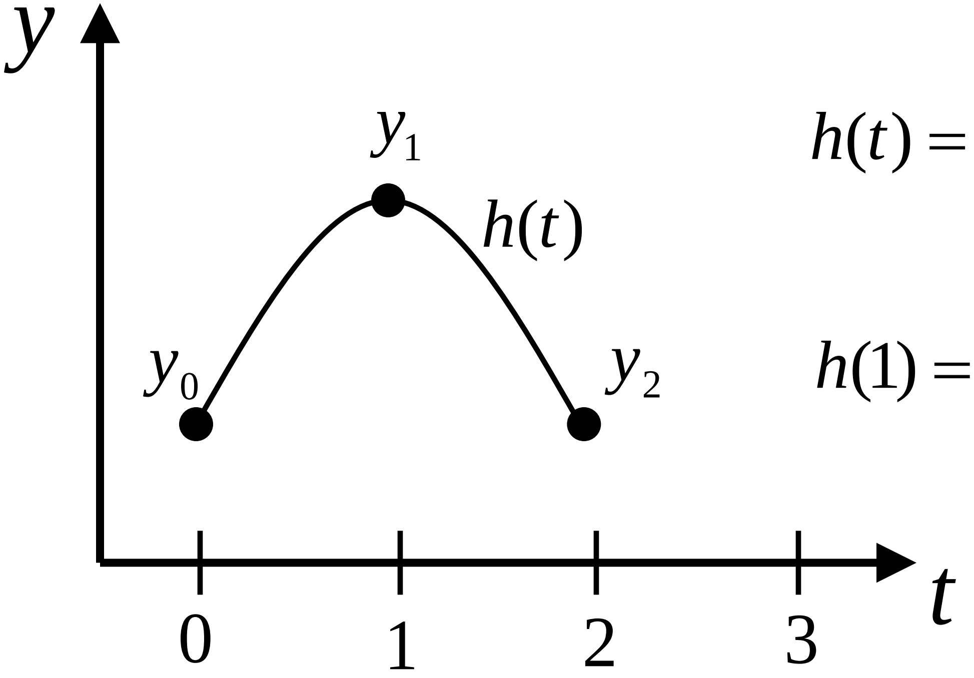
- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)\,y_0 + t\,y_1$$

$$g(t) = (2-t)\,y_1 + (t-1)\,y_2$$

$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(1) = \frac{y_1 + y_1}{2} = y_1$$



Lagrange Interpolation

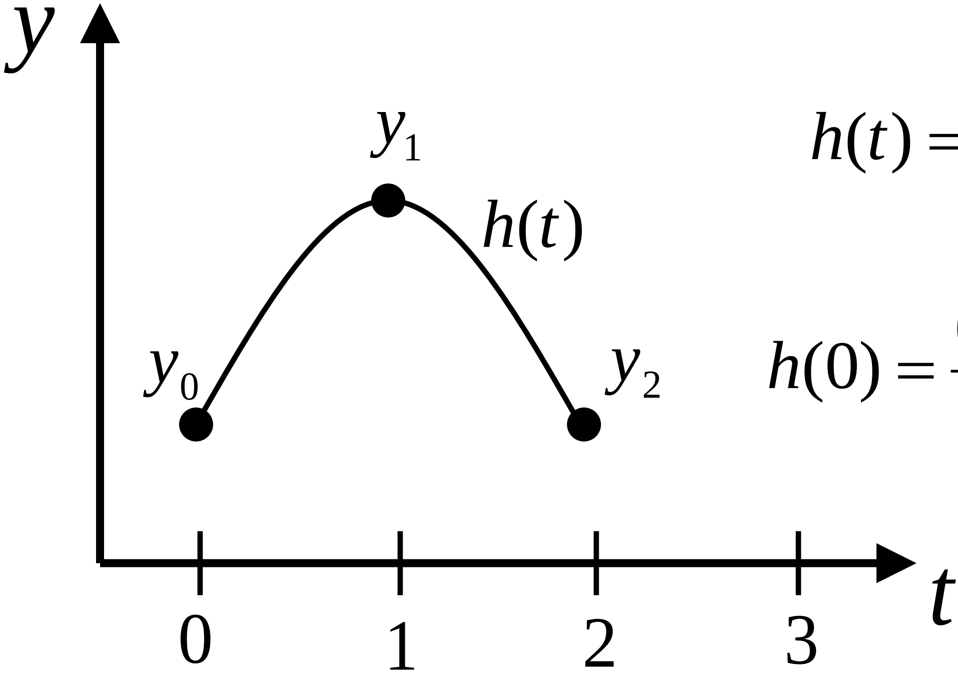
- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)\,y_0 + t\,y_1$$

$$g(t) = (2-t)\,y_1 + (t-1)\,y_2$$

$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(0) = \frac{(2-0)\,f(0) + 0\,g(0)}{2}$$



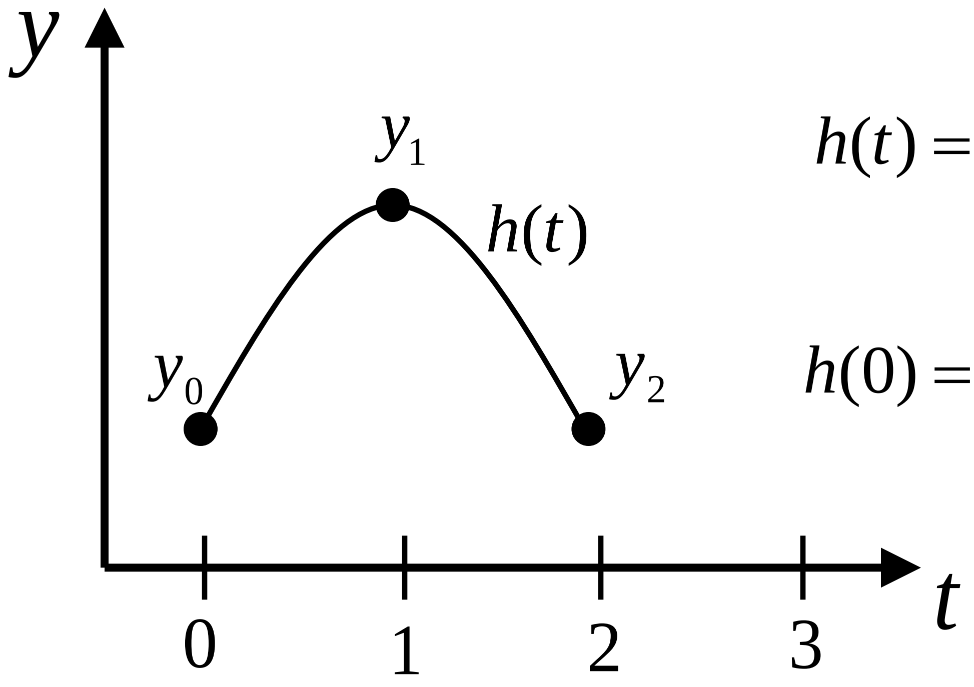
Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)\,y_0 + t\,y_1$$
$$g(t) = (2-t)\,y_1 + (t-1)\,y_2$$

$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(0) = \frac{2y_0}{2} = y_0$$



Lagrange Interpolation

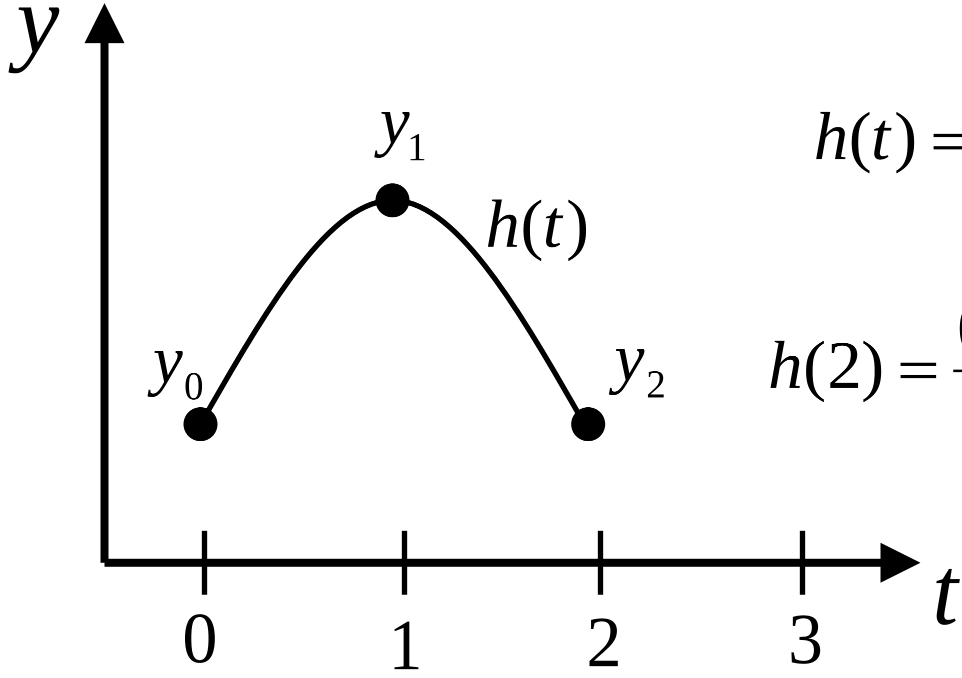
- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)y_0 + t\,y_1$$

$$g(t) = (2-t)y_1 + (t-1)\,y_2$$

$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(2) = \frac{(2-2)\,f(2) + 2\,g(2)}{2}$$



Lagrange Interpolation

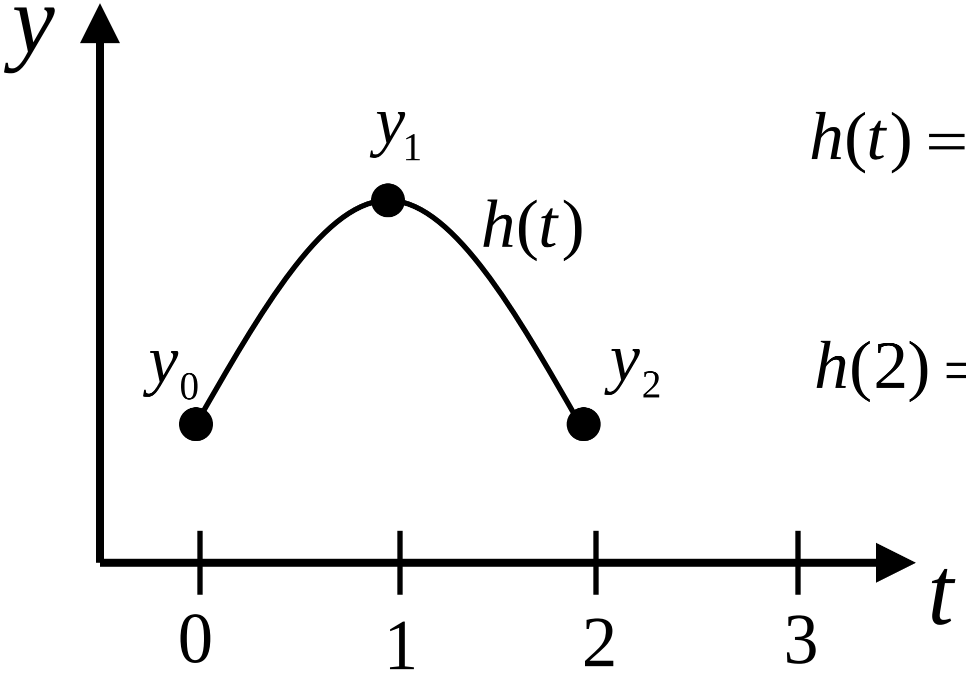
- Identical to matrix method but uses a geometric construction

$$f(t) = (1-t)y_0 + t\,y_1$$

$$g(t) = (2-t)y_1 + (t-1)\,y_2$$

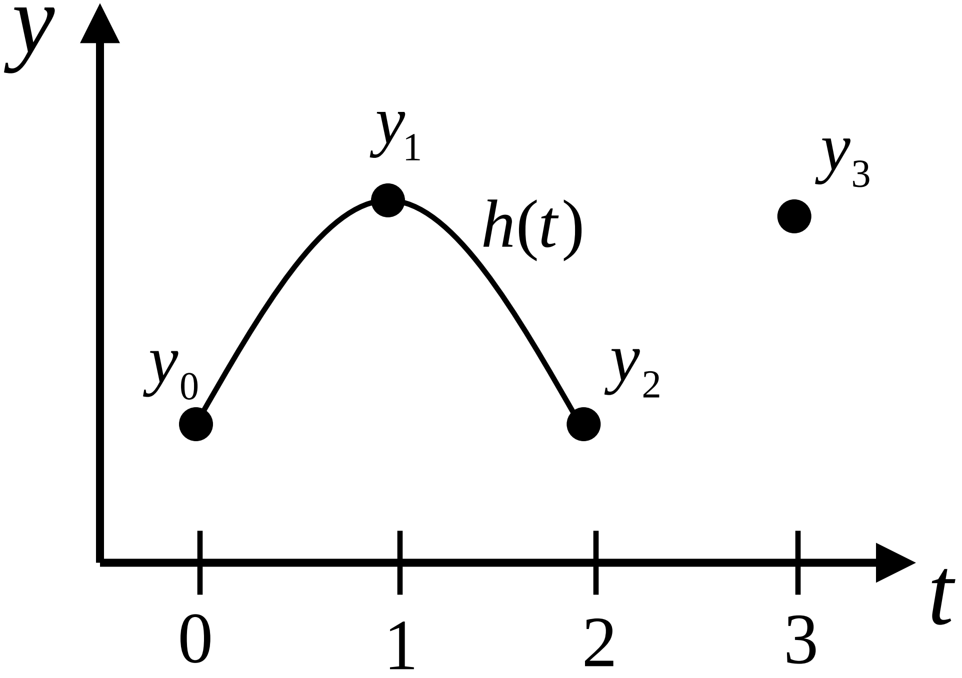
$$h(t) = \frac{(2-t)\,f(t) + t\,g(t)}{2}$$

$$h(2) = \frac{2\,y_2}{2} = y_2$$



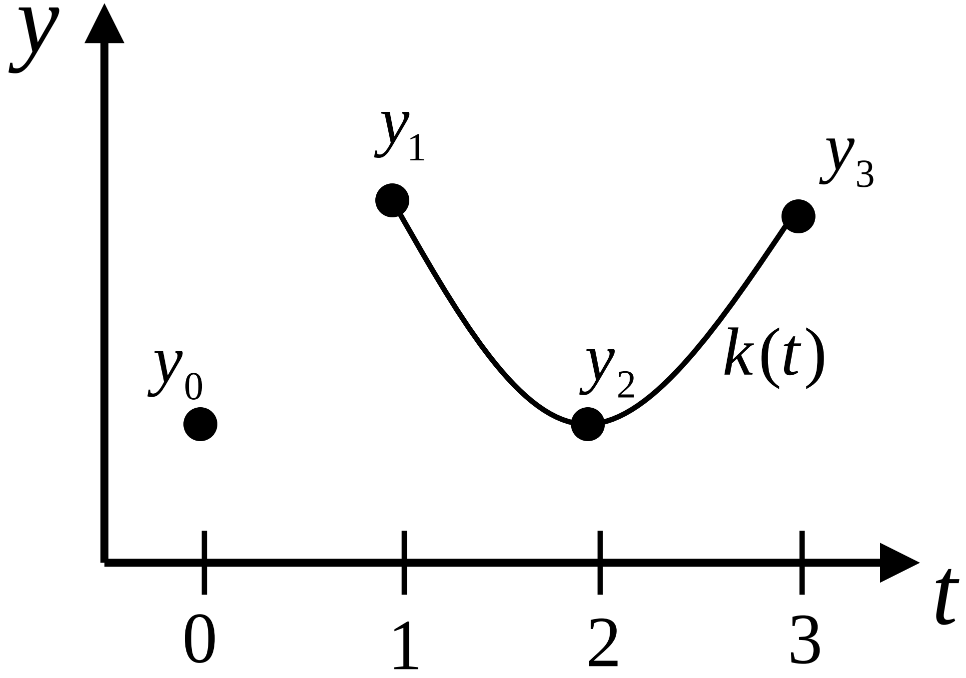
Lagrange Interpolation

- Identical to matrix method but uses a geometric construction



Lagrange Interpolation

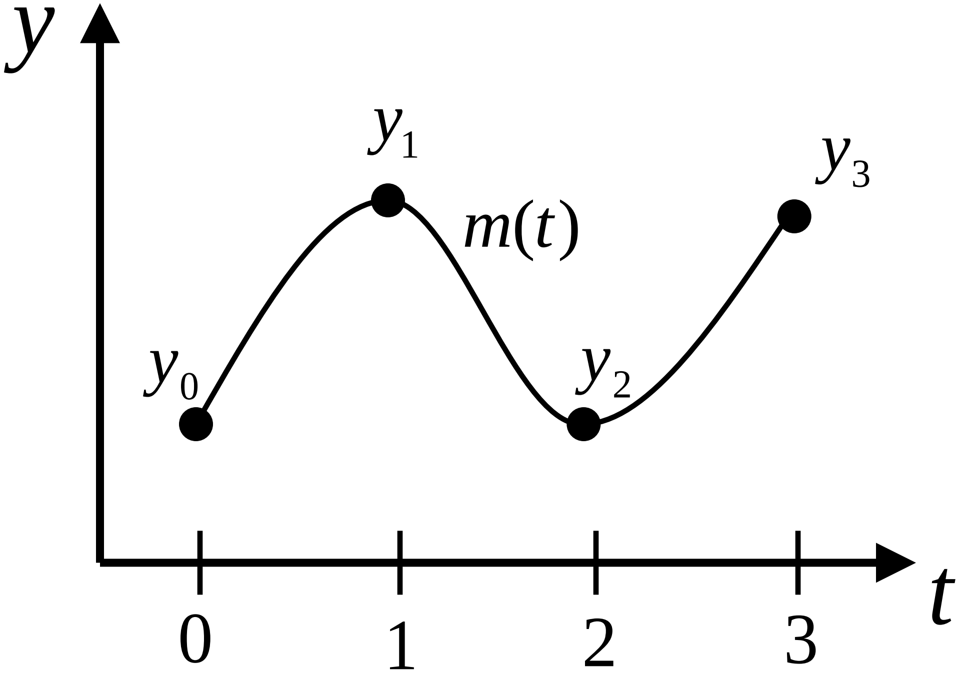
- Identical to matrix method but uses a geometric construction



Lagrange Interpolation

- Identical to matrix method but uses a geometric construction

$$m(t) = \frac{(3-t)h(t) + t\,k(t)}{3}$$



Uniform Lagrange Interpolation

- Base Case
  - Linear interpolation between two points
- Inductive Step
  - Assume we have points $y_{\mathrm{i}}, \ldots, y_{n+1+\mathrm{i}}$
  - Build interpolating polynomials $f(t)$, $g(t)$ of degree n for $y_{\mathrm{i}}, \ldots, y_{n+i}$ and $y_{\mathrm{i}+1}, \ldots, y_{n+1+\mathrm{i}}$
  - $h(t) = \dfrac{(n+1+i-t)\,f(t)+(t-i)\,g(t)}{n+1}$
  - $h(t)$ interpolates all points $y_{\mathrm{i}}, \ldots, y_{n+1+\mathrm{i}}$ and is of degree $n+1$
  - Moreover, $h(t)$ is unique

Lagrange Evaluation Pseudocode

Lagrange(*pts*, *i*, *t*)

    n = length(*pts*)-2

    if *n* is 0

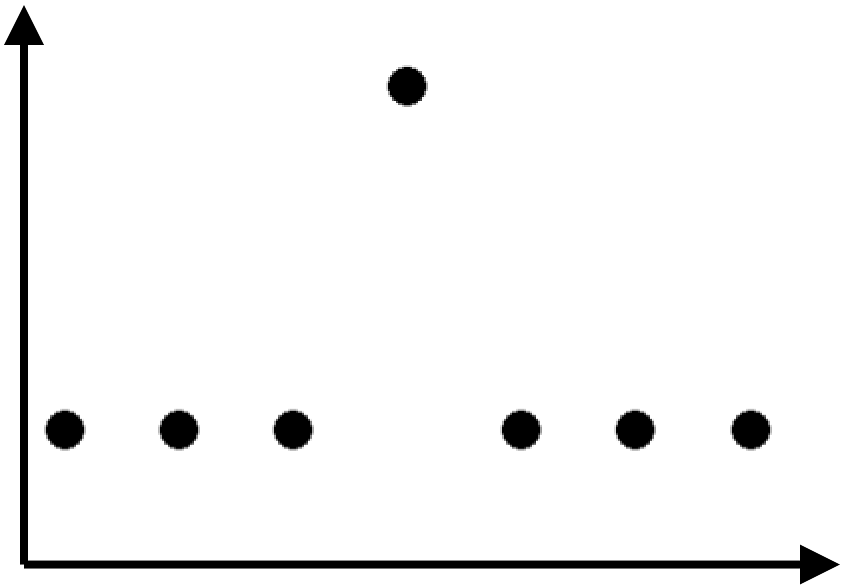
        return *pts*[0](*i*+1-*t*)+*pts*[1](*t*-*i*)

    f=Lagrange(*pts* without last element, *i*, *t*)

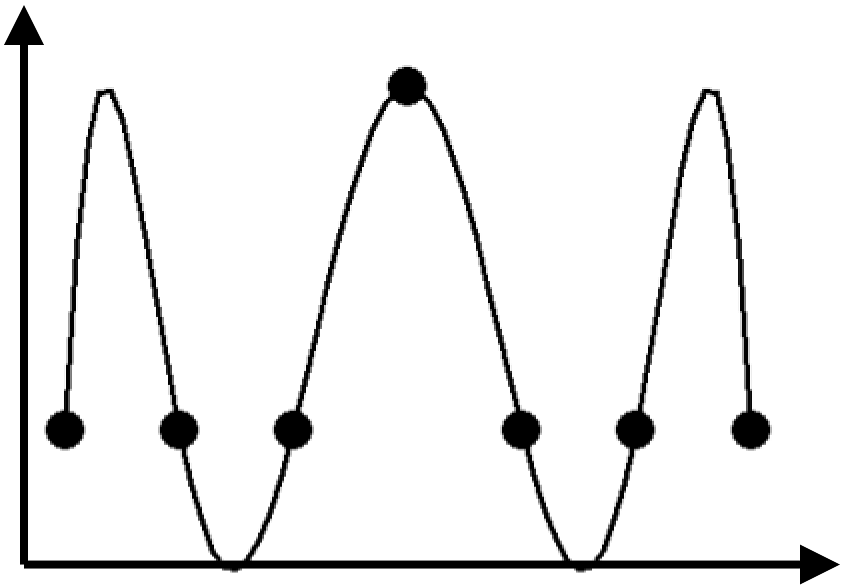
    g=Lagrange(*pts* without first element, *i*+1, *t*)

    return ((*n*+1+*i*-*t*)f+(*t*-*i*)*g*)/(*n*+1)

Problems with Interpolation

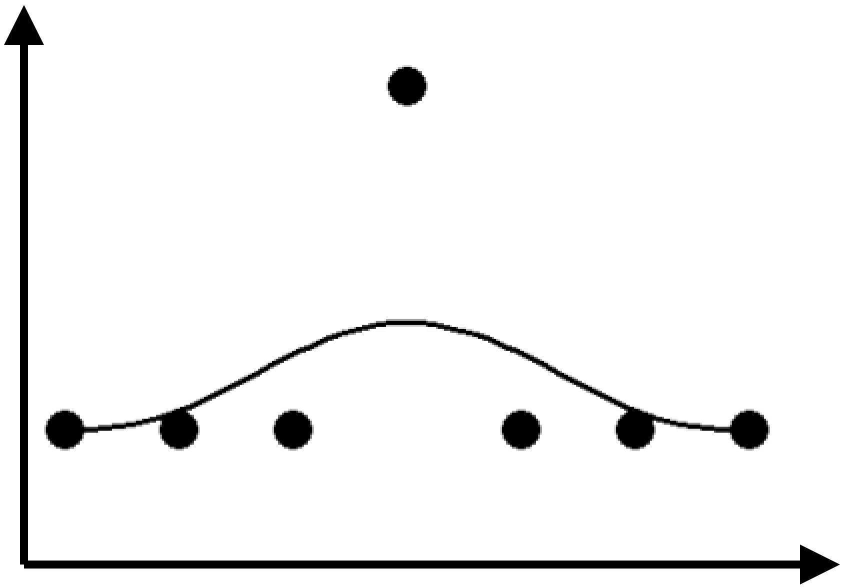


Problems with Interpolation



Bezier Curves

- Polynomial curves that seek to approximate rather than to interpolate



Bernstein Polynomials

- Degree 1: $(1-t)$, t
- Degree 2: $(1-t)^2$, $2(1-t)t$, t^2
- Degree 3: $(1-t)^3$, $3(1-t)^2t$, $3(1-t)t^2$, t^3

Bernstein Polynomials

- Degree 1: $(1\text{-}t)$, t
- Degree 2: $(1\text{-}t)^2$, $2(1\text{-}t)t$, t^2
- Degree 3: $(1\text{-}t)^3$, $3(1\text{-}t)^2t$, $3(1\text{-}t)t^2$, t^3
- Degree 4: $(1\text{-}t)^4$, $4(1\text{-}t)^3t$, $6(1\text{-}t)^2t^2$, $4(1\text{-}t)t^3$,t^4

Bernstein Polynomials

- Degree 1: $(1\text{-}t)$, t
- Degree 2: $(1\text{-}t)^2$, $2(1\text{-}t)t$, t^2
- Degree 3: $(1\text{-}t)^3$, $3(1\text{-}t)^2t$, $3(1\text{-}t)t^2$, t^3
- Degree 4: $(1\text{-}t)^4$, $4(1\text{-}t)^3t$, $6(1\text{-}t)^2t^2$, $4(1\text{-}t)t^3$,t^4
- Degree 5: $(1\text{-}t)^5$, $5(1\text{-}t)^4t$, $10(1\text{-}t)^3t^2$, $10(1\text{-}t)^2t^3$,$5(1\text{-}t)t^4$,t^5
- …
- Degree n: $\binom{n}{i}(1-t)^{n-i}t^i$ for $0 \le i \le n$

Bezier Curves

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

Bezier Curves

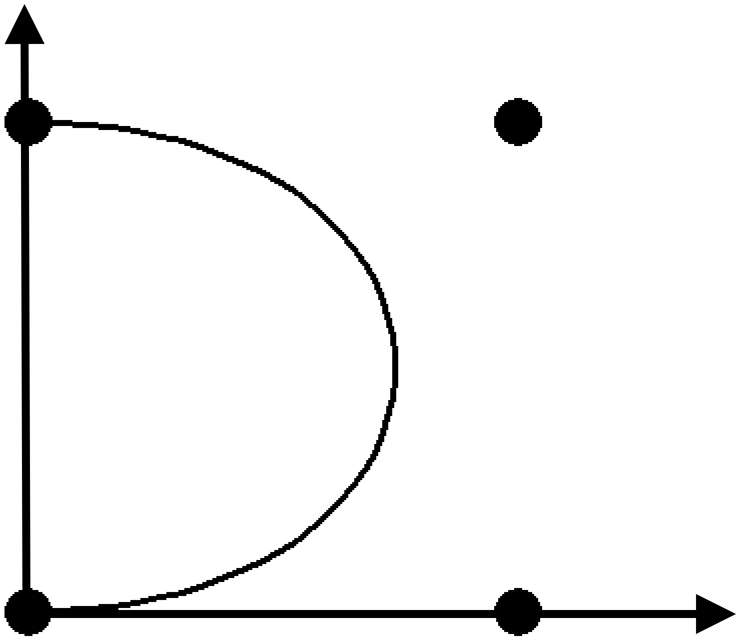
$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

$$p(t) = (1-t)^3(0,0) + 3(1-t)^2 t(1,0) + 3(1-t)t^2(1,1) + t^3(0,1)$$

Bezier Curves

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

$$p(t) = (3(1-t)t, (3-2t)t^2)$$



Bezier Curve Properties

- Interpolate end-points

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

Bezier Curve Properties

- Interpolate end-points

$$p(0) = \sum_{i=0}^{n} \binom{n}{i} (1-0)^{n-i} 0^i p_i = p_0$$

Bezier Curve Properties

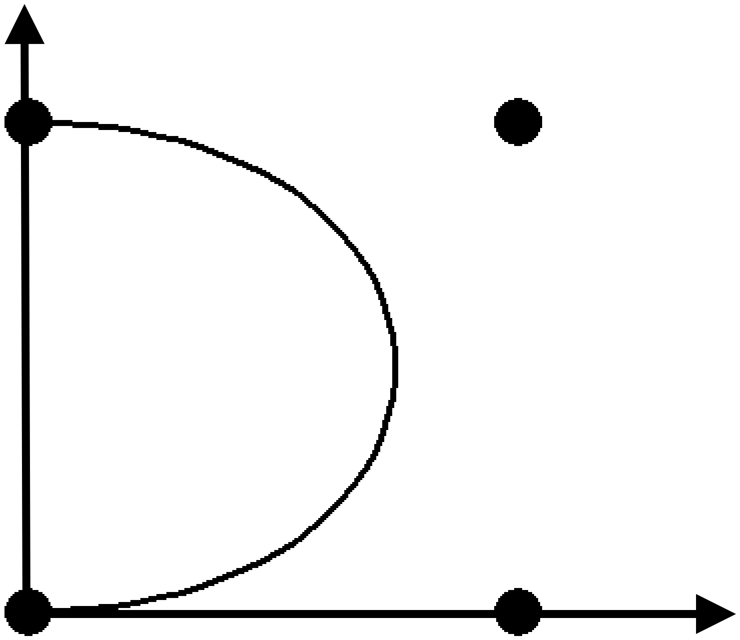
- Interpolate end-points

$$p(1)=\sum_{i=0}^{n}\binom{n}{i}(1-1)^{n-i}1^{i}p_i=p_n$$

Bezier Curve Properties

- Interpolate end-points

$$p(1) = \sum_{i=0}^{n} \binom{n}{i} (1-1)^{n-i} 1^i \, p_i = p_n$$



Bezier Curve Properties

- Interpolate end-points
- Tangent at end-points in direction of first/last edge

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

Bezier Curve Properties

- Interpolate end-points
- Tangent at end-points in direction of first/last edge

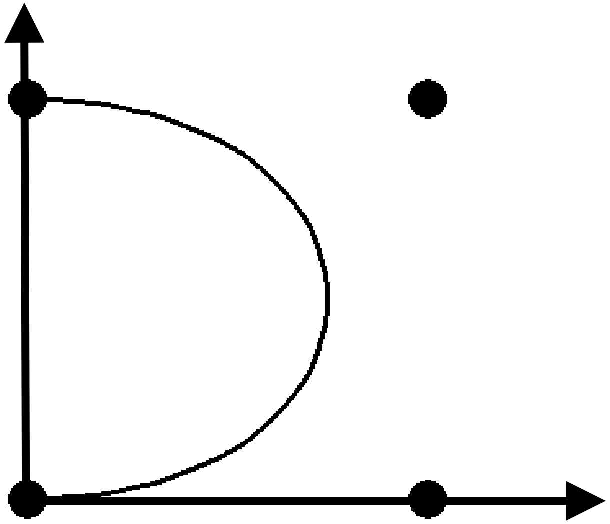
$$\frac{\partial p(t)}{\partial t} = \frac{\partial}{\partial t} \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^{i} p_i = \sum_{i=0}^{n} \binom{n}{i} p_i \left(i(1-t)^{n-i} t^{i-1} - (n-i)(1-t)^{n-i-1} t^{i} \right)$$

Bezier Curve Properties

- Interpolate end-points
- Tangent at end-points in direction of first/last edge

$$\frac{\partial p(0)}{\partial t} = n(p_1 - p_0)$$

$$\frac{\partial p(1)}{\partial t} = n(p_n - p_{n-1})$$

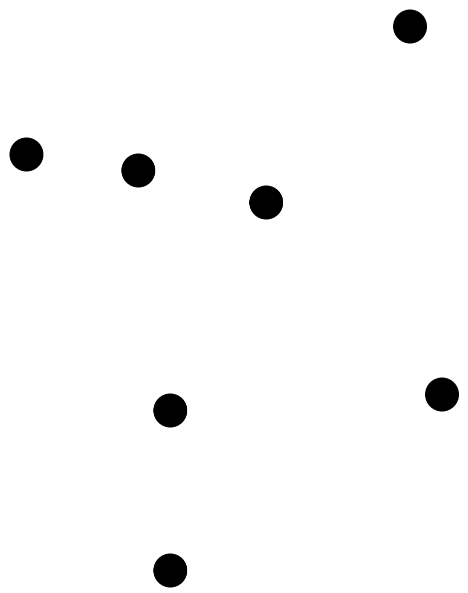


Bezier Curve Properties

- Interpolate end-points
- Tangent at end-points in direction of first/last edge
- Curve lies within the convex hull of the control points

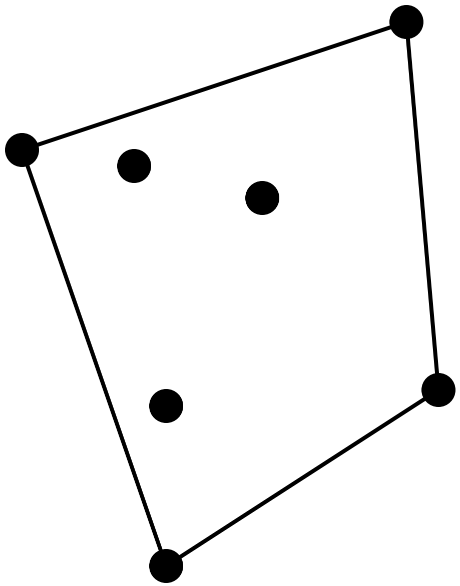
Convex Hull

- The smallest convex set containing all points p_i



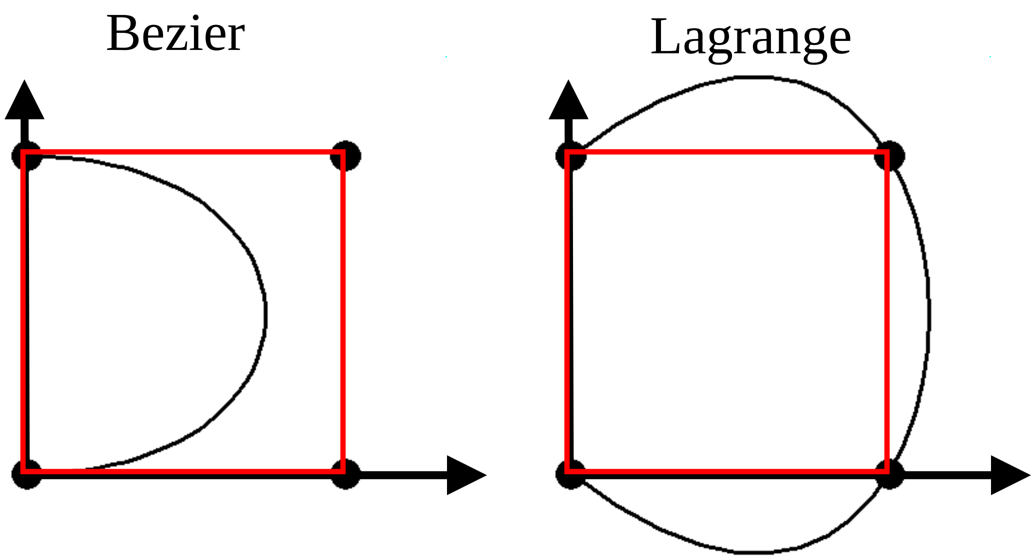
Convex Hull

- The smallest convex set containing all points p_i



Bezier Curve Properties

- Interpolate end-points
- Tangent at end-points in direction of first/last edge
- Curve lies within the convex hull of the control points

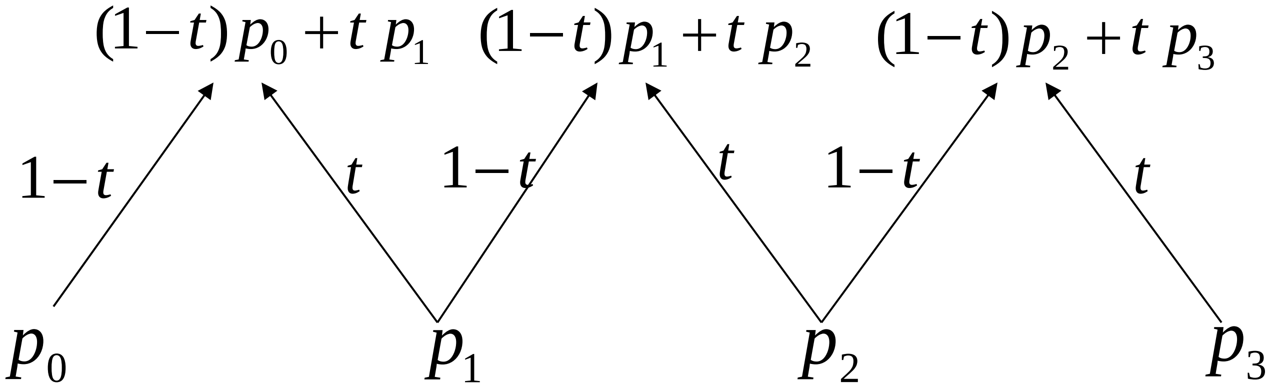


Pyramid Algorithms for Bezier Curves

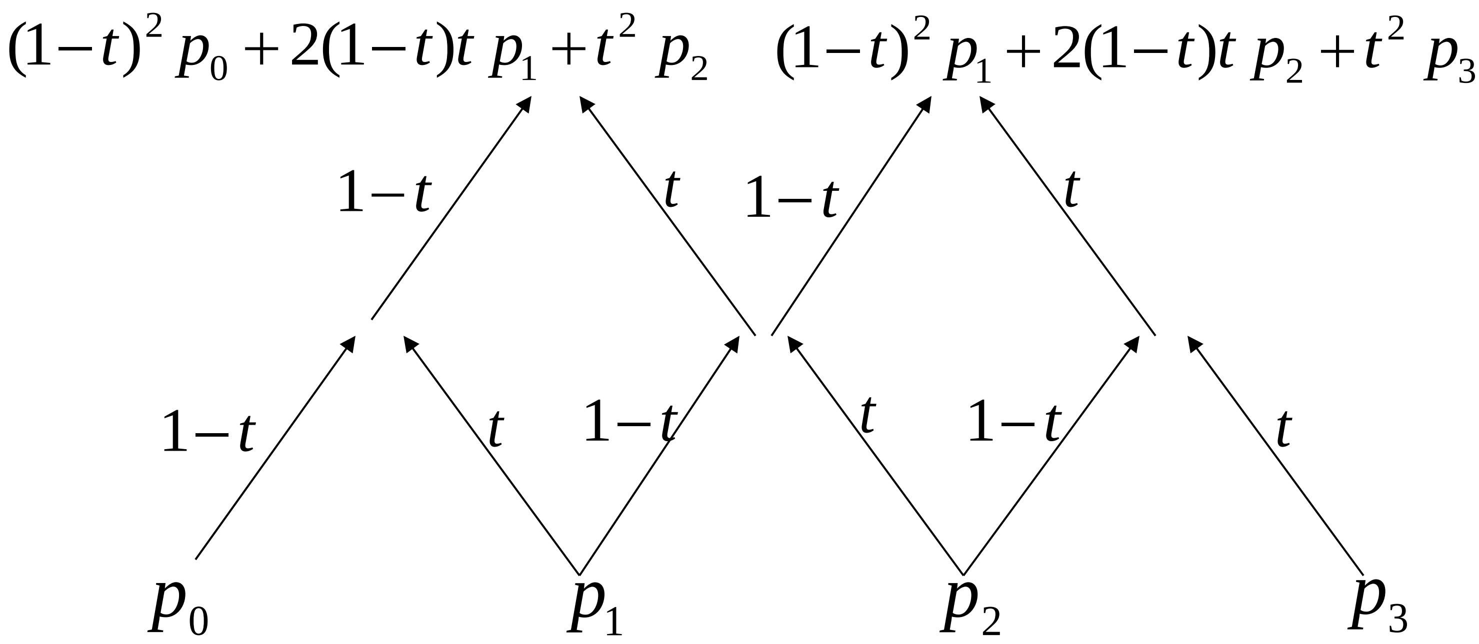
- Polynomials aren't pretty
- Is there an easier way to evaluate the equation of a Bezier curve?

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i$$

Pyramid Algorithms for Bezier Curves

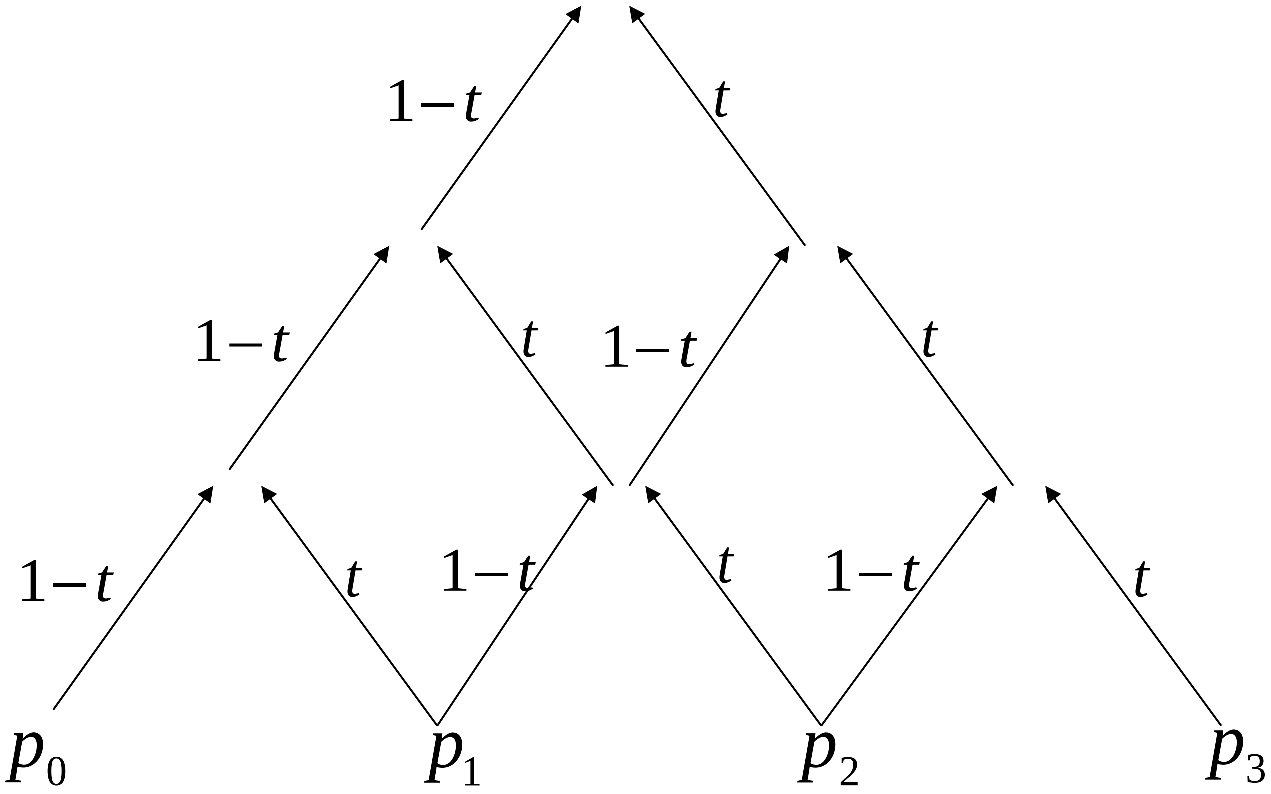


Pyramid Algorithms for Bezier Curves



Pyramid Algorithms for Bezier Curves

$$(1-t)^3 p_0 + 3(1-t)^2 t\, p_1 + 3(1-t)t^2 p_2 + t^3\, p_3$$

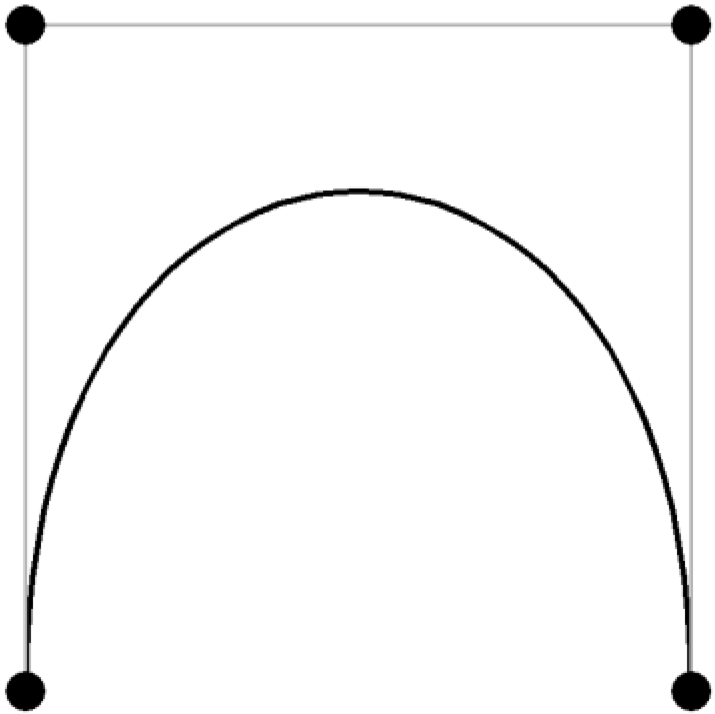


Bezier Evaluation Pseudocode

```
Bezier(pts, t)
    if ( length(pts) is 1)
        return pts[0]
    newPts={}
    For ( i = 0, i < length(pts) – 1, i++ )
        Add to newPts (1-t)pts[i] + t pts[i+1]
    return Bezier ( newPts, t )
```

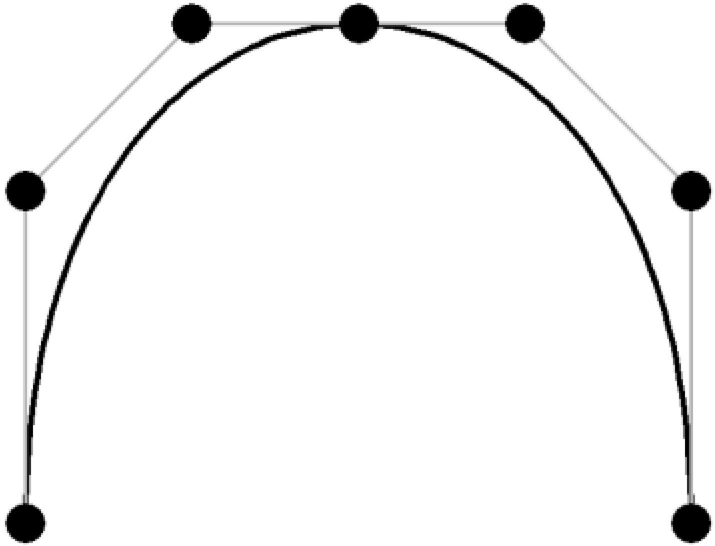
Subdividing Bezier Curves

- Given a single Bezier curve, construct two smaller Bezier curves whose union is exactly the original curve



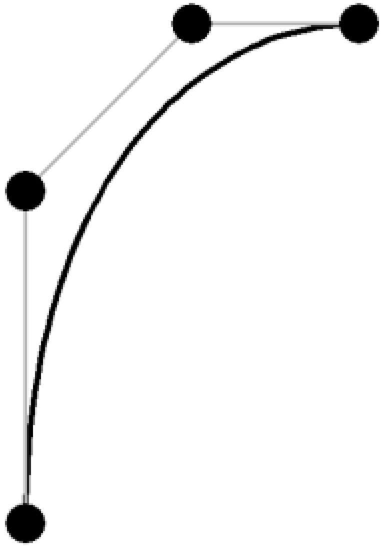
Subdividing Bezier Curves

- Given a single Bezier curve, construct two smaller Bezier curves whose union is exactly the original curve



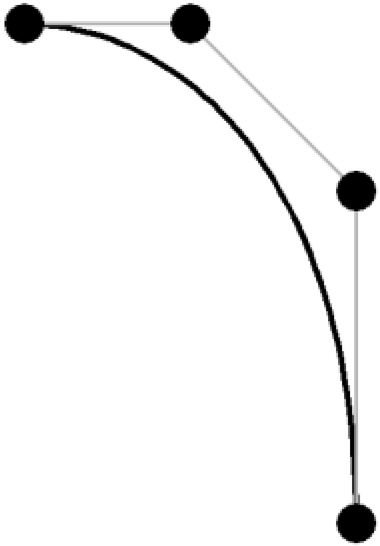
Subdividing Bezier Curves

- Given a single Bezier curve, construct two smaller Bezier curves whose union is exactly the original curve



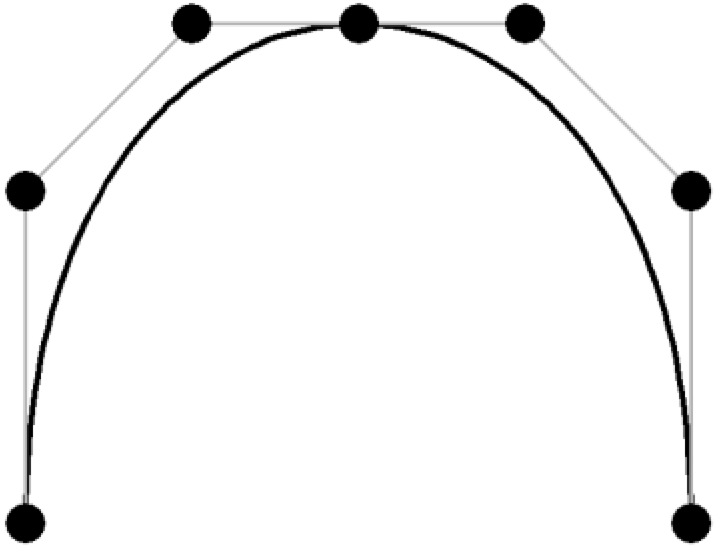
Subdividing Bezier Curves

- Given a single Bezier curve, construct two smaller Bezier curves whose union is exactly the original curve



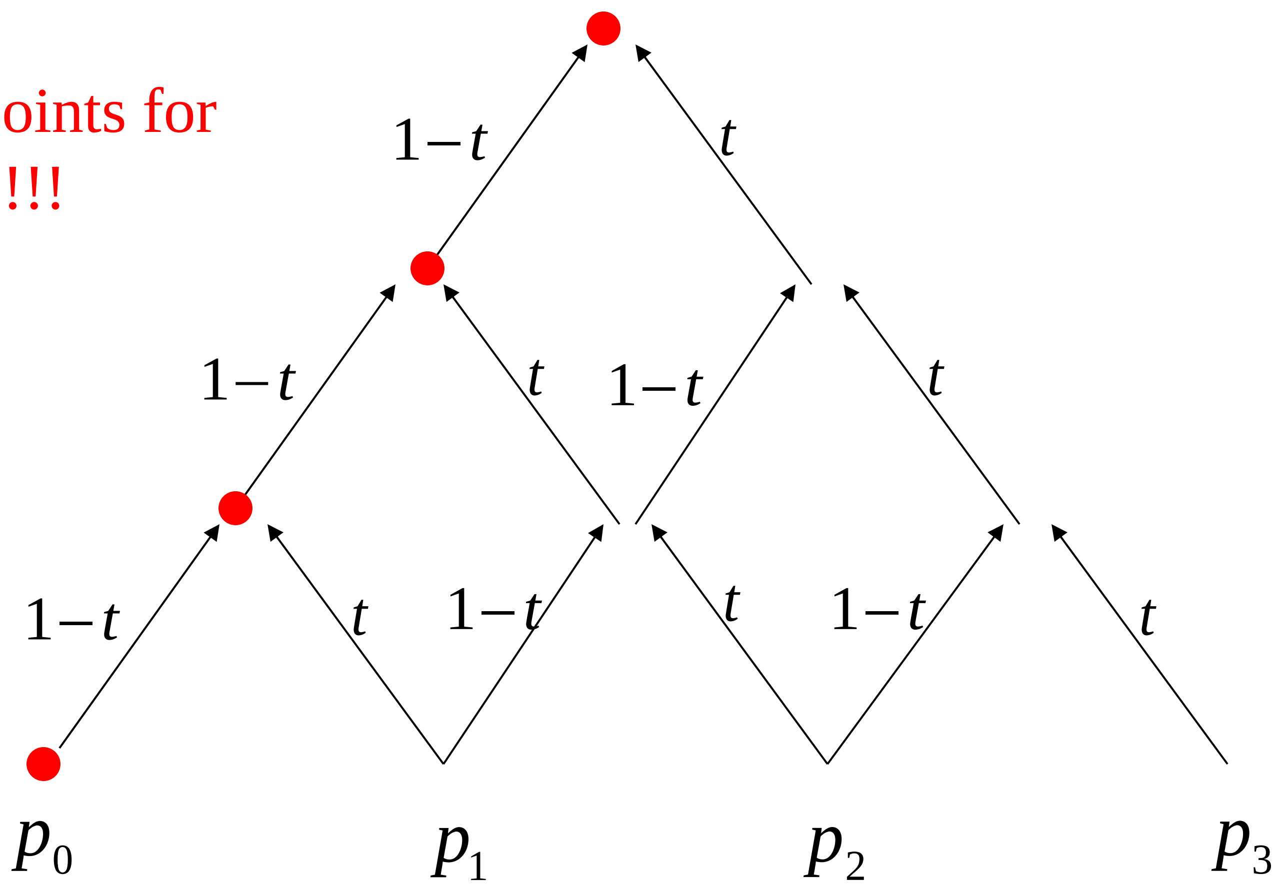
Subdividing Bezier Curves

- Given a single Bezier curve, construct two smaller Bezier curves whose union is exactly the original curve

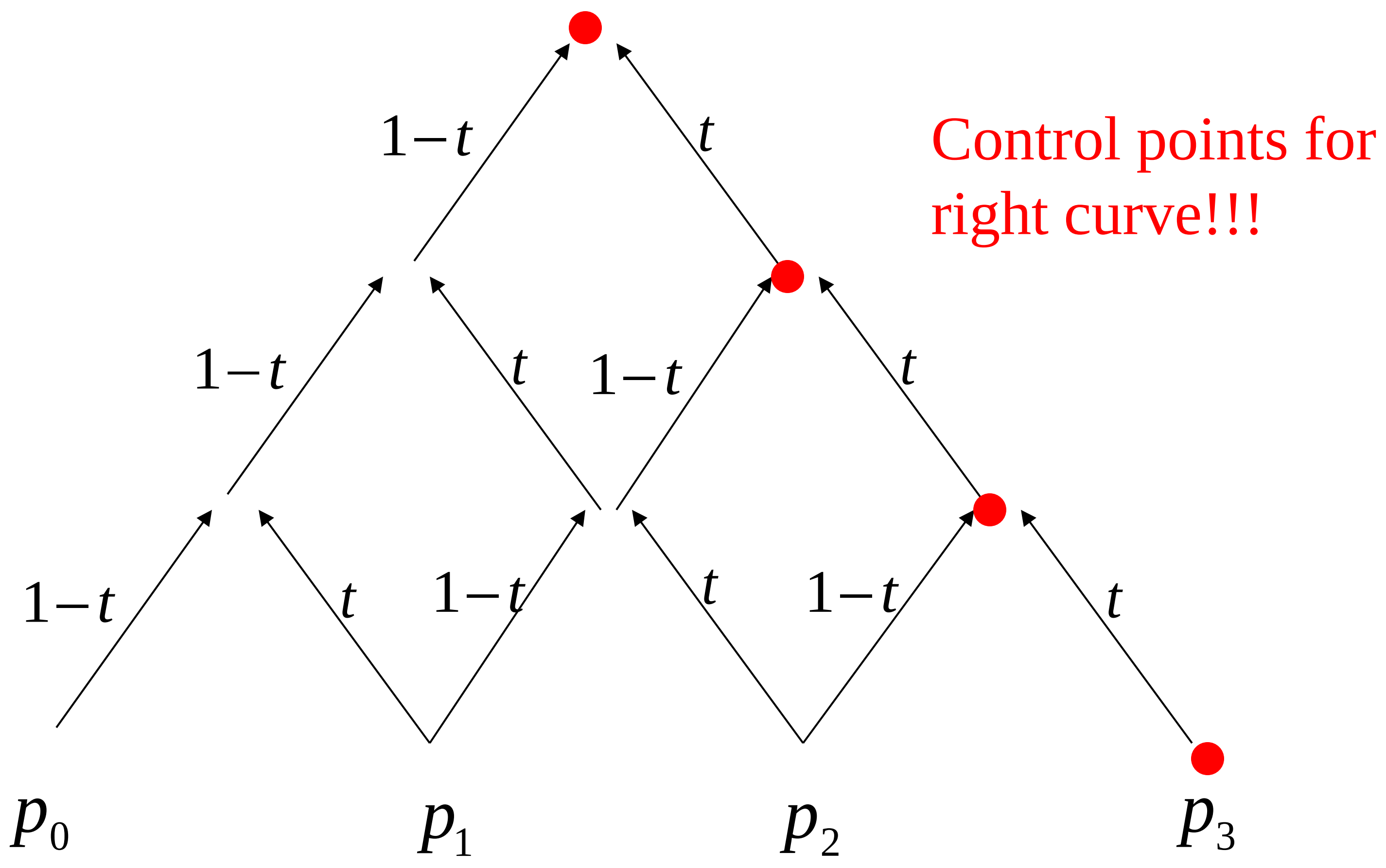


Subdividing Bezier Curves

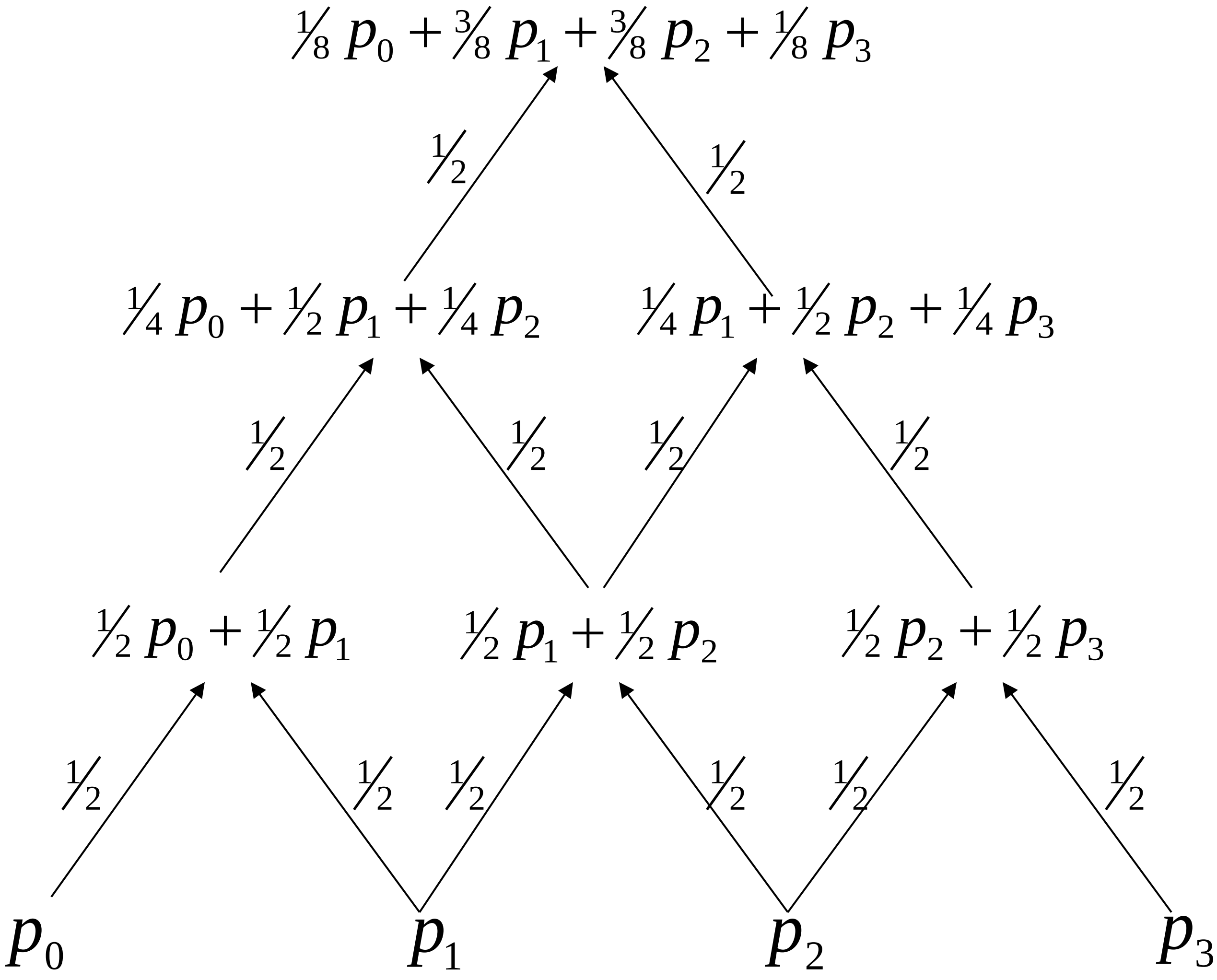
Control points for
left curve!!!



Subdividing Bezier Curves



Subdividing Bezier Curves

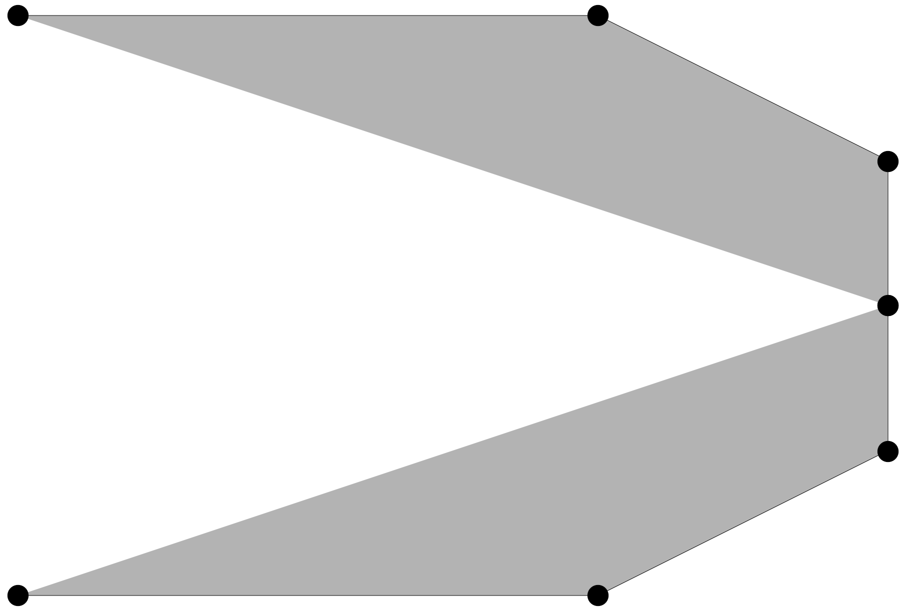


Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces

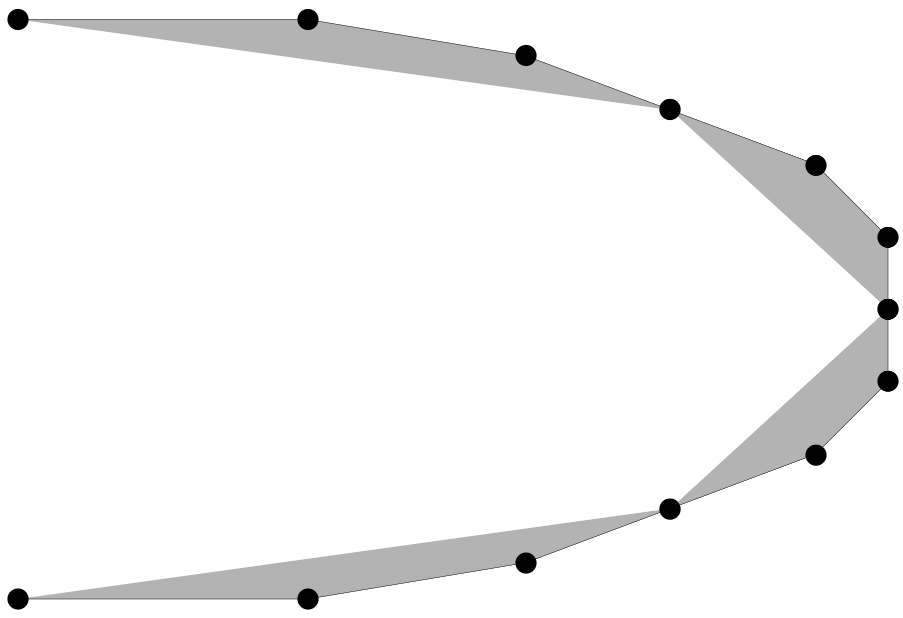
Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces



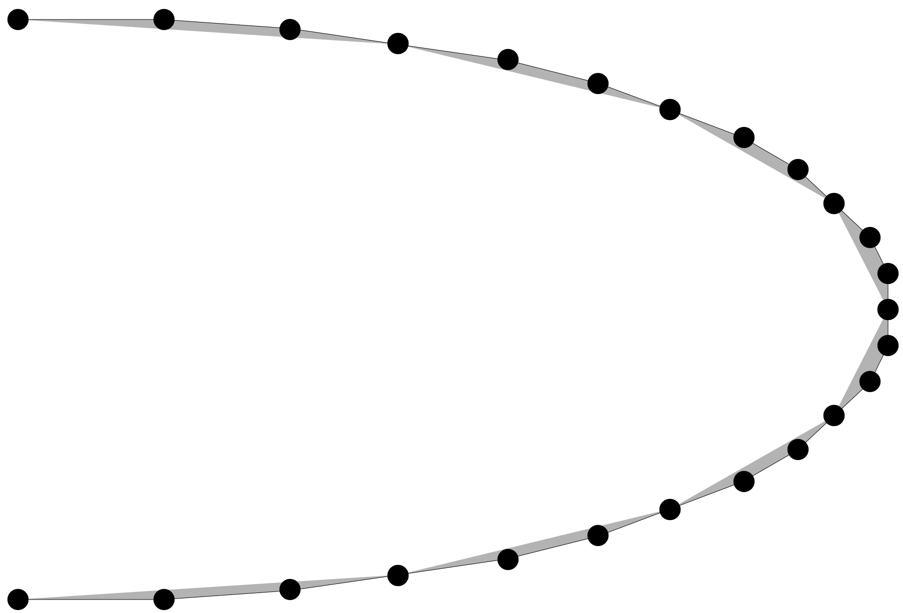
Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces



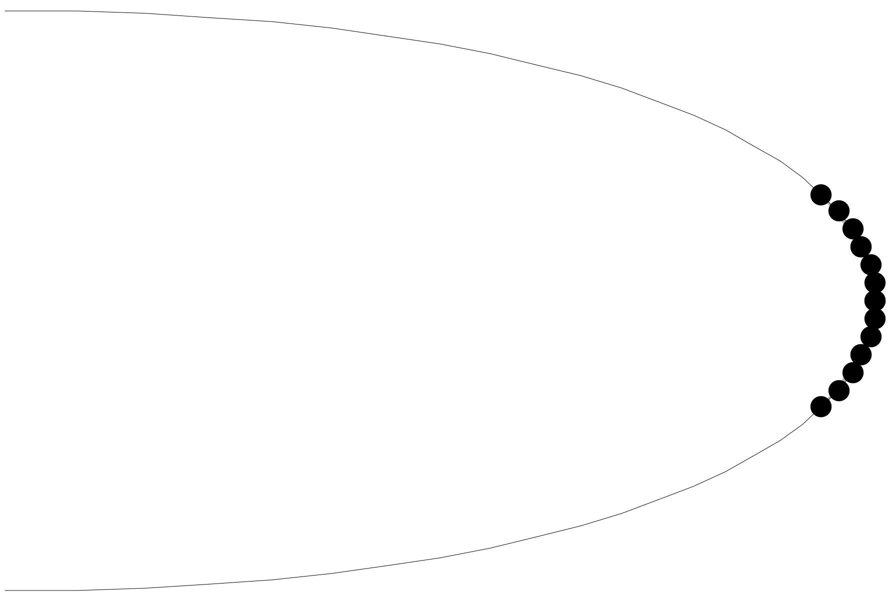
Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces



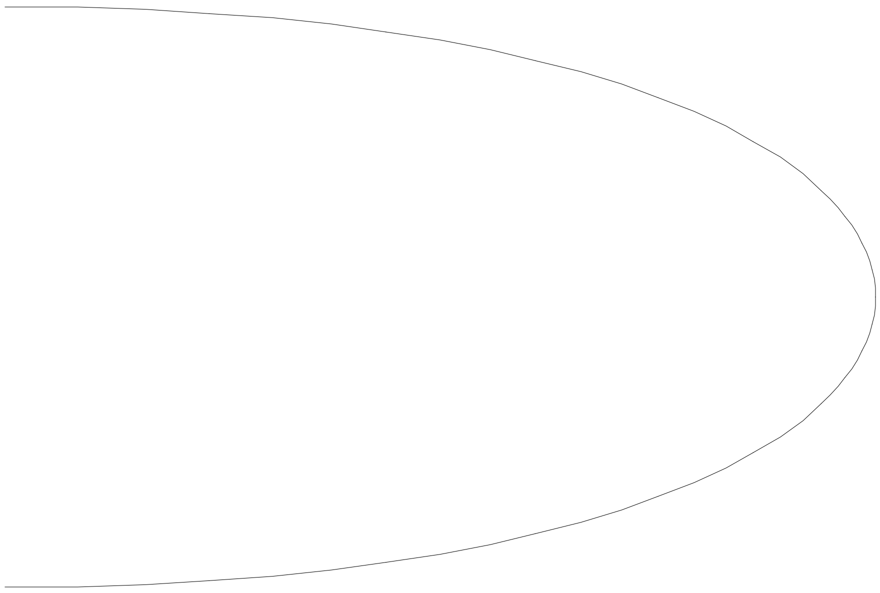
Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces



Adaptive Rendering of Bezier Curves

- If control polygon is close to a line, draw the control polygon
- If not, subdivide and recur on subdivided pieces



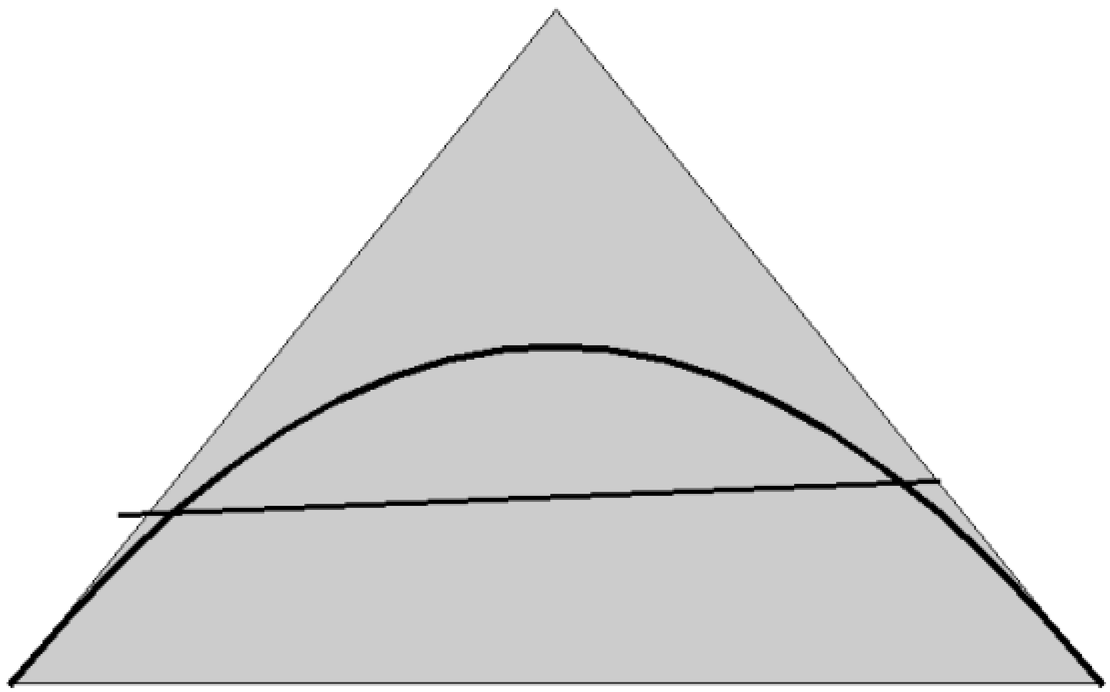
Applications: Intersection

- Given two Bezier curves, determine if and where they intersect

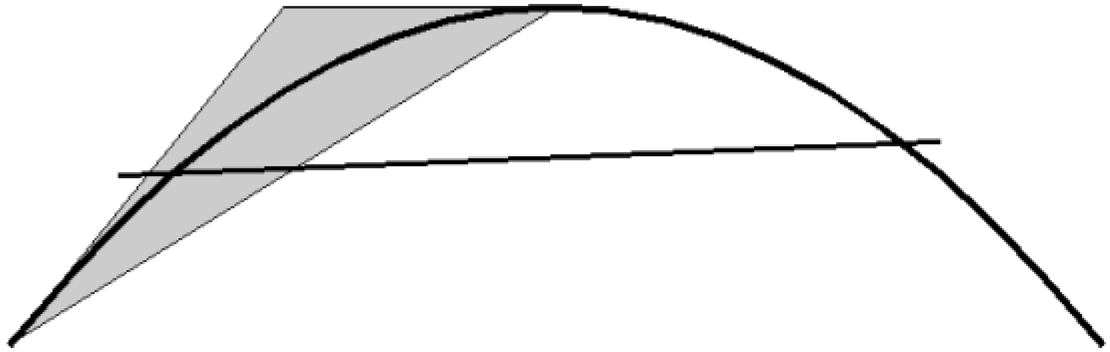
Applications: Intersection

- Check if convex hulls intersect
- If not, return no intersection
- If both convex hulls can be approximated with a straight line, intersect lines and return intersection
- Otherwise subdivide and recur on subdivided pieces

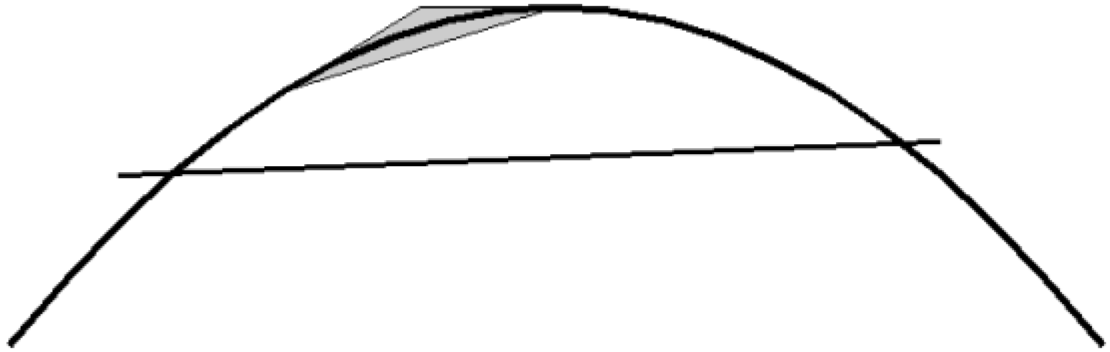
Applications: Intersection



Applications: Intersection



Applications: Intersection

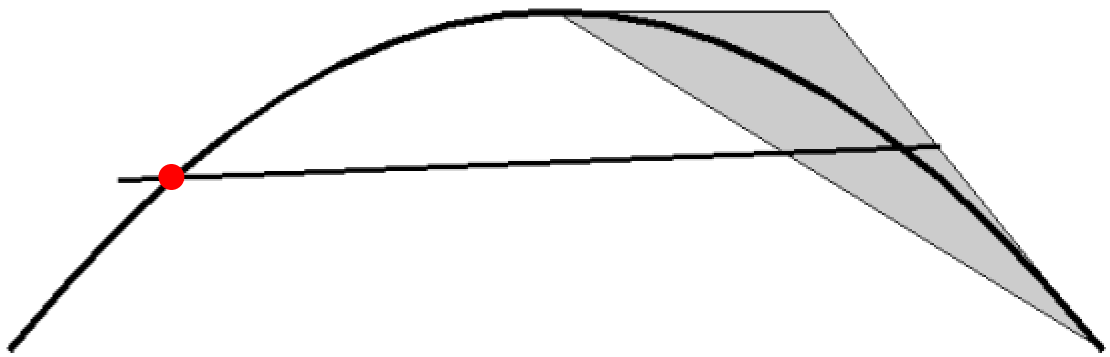


Applications: Intersection

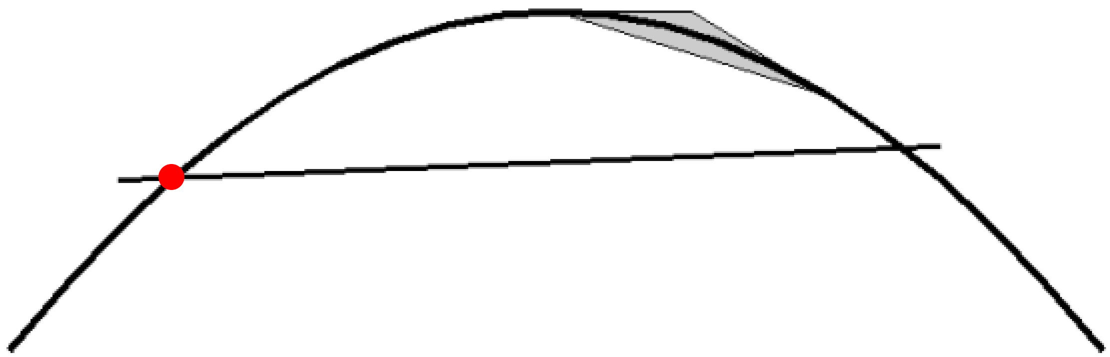
Applications: Intersection

Applications: Intersection

Applications: Intersection



Applications: Intersection

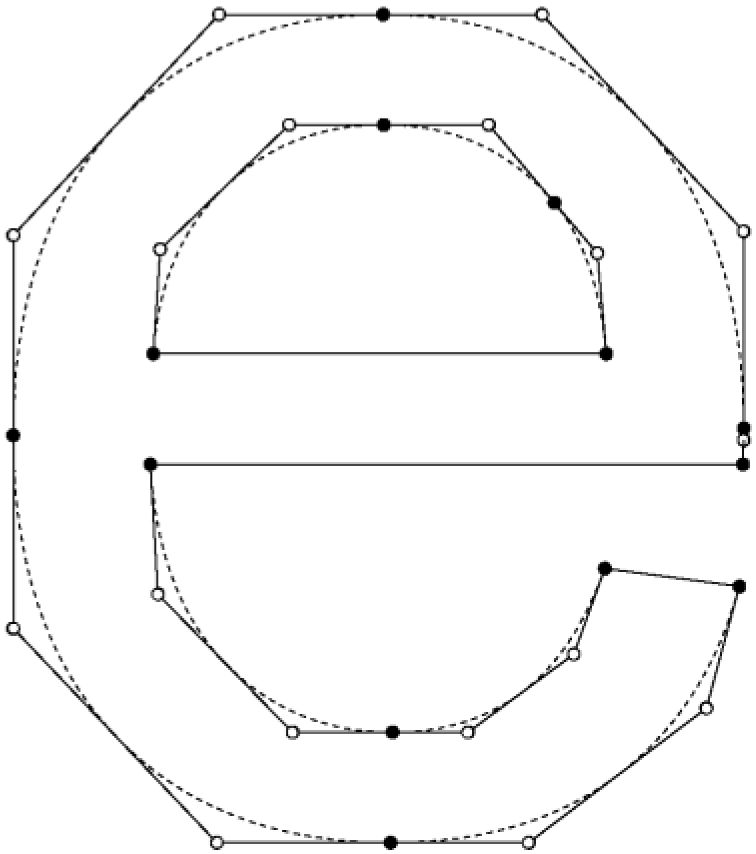


Applications: Intersection

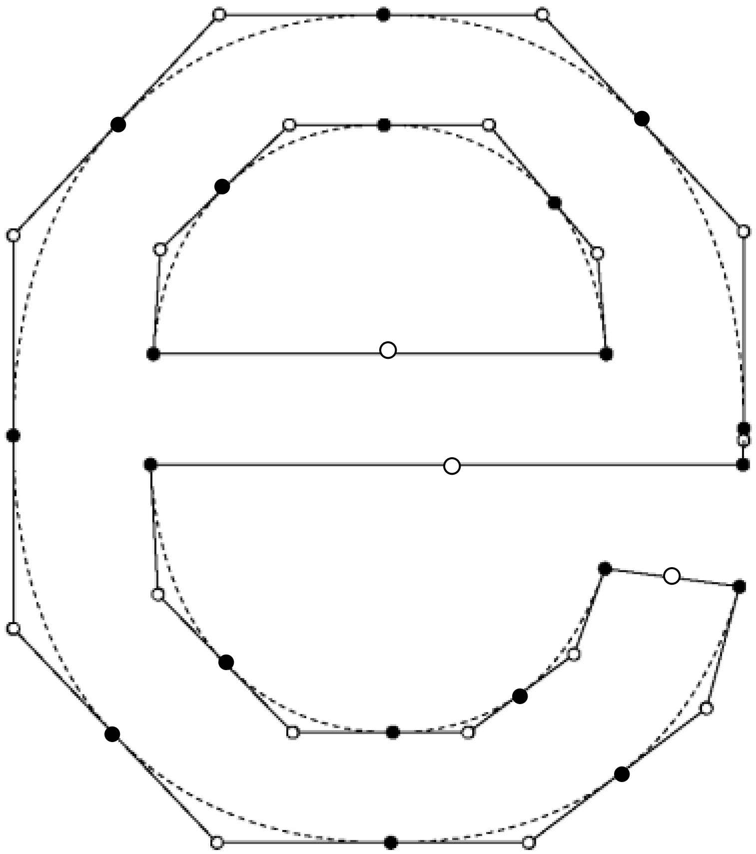
Applications: Intersection

Applications: Intersection

Application: Font Rendering

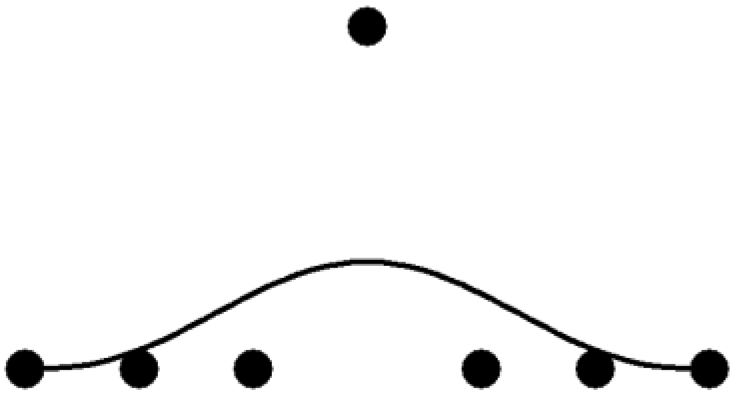


Application: Font Rendering



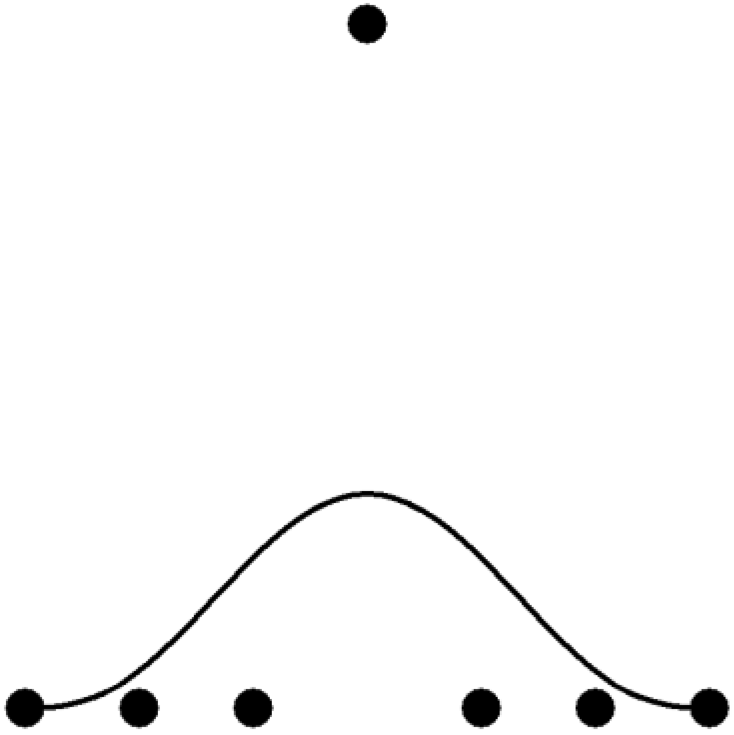
Problems with Bezier Curves

- More control points means higher degree
- Moving one control point affects the entire curve



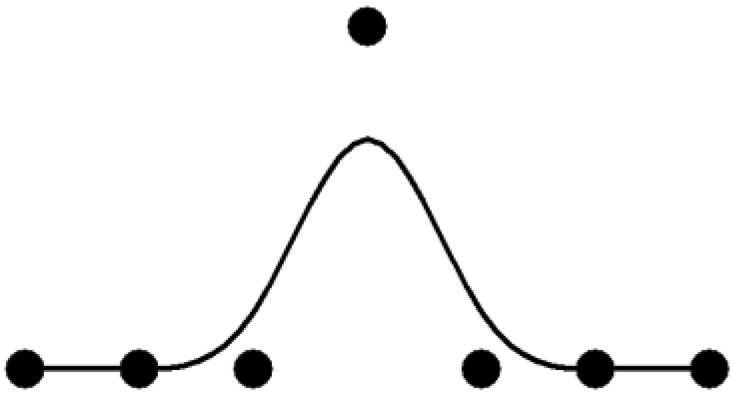
Problems with Bezier Curves

- More control points means higher degree
- Moving one control point affects the entire curve



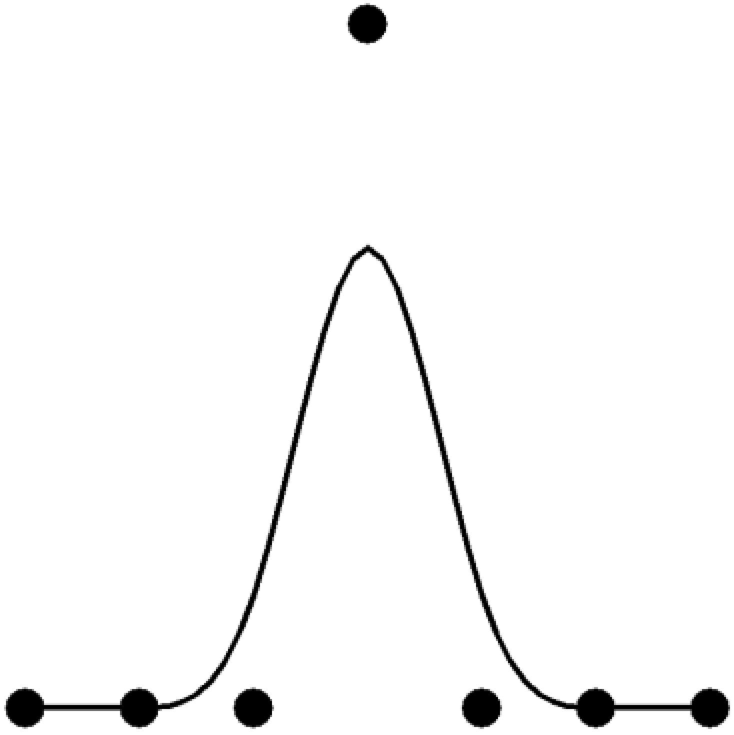
B-spline Curves

- Not a single polynomial, but lots of polynomials that meet together smoothly
- Local control



B-spline Curves

- Not a single polynomial, but lots of polynomials that meet together smoothly
- Local control



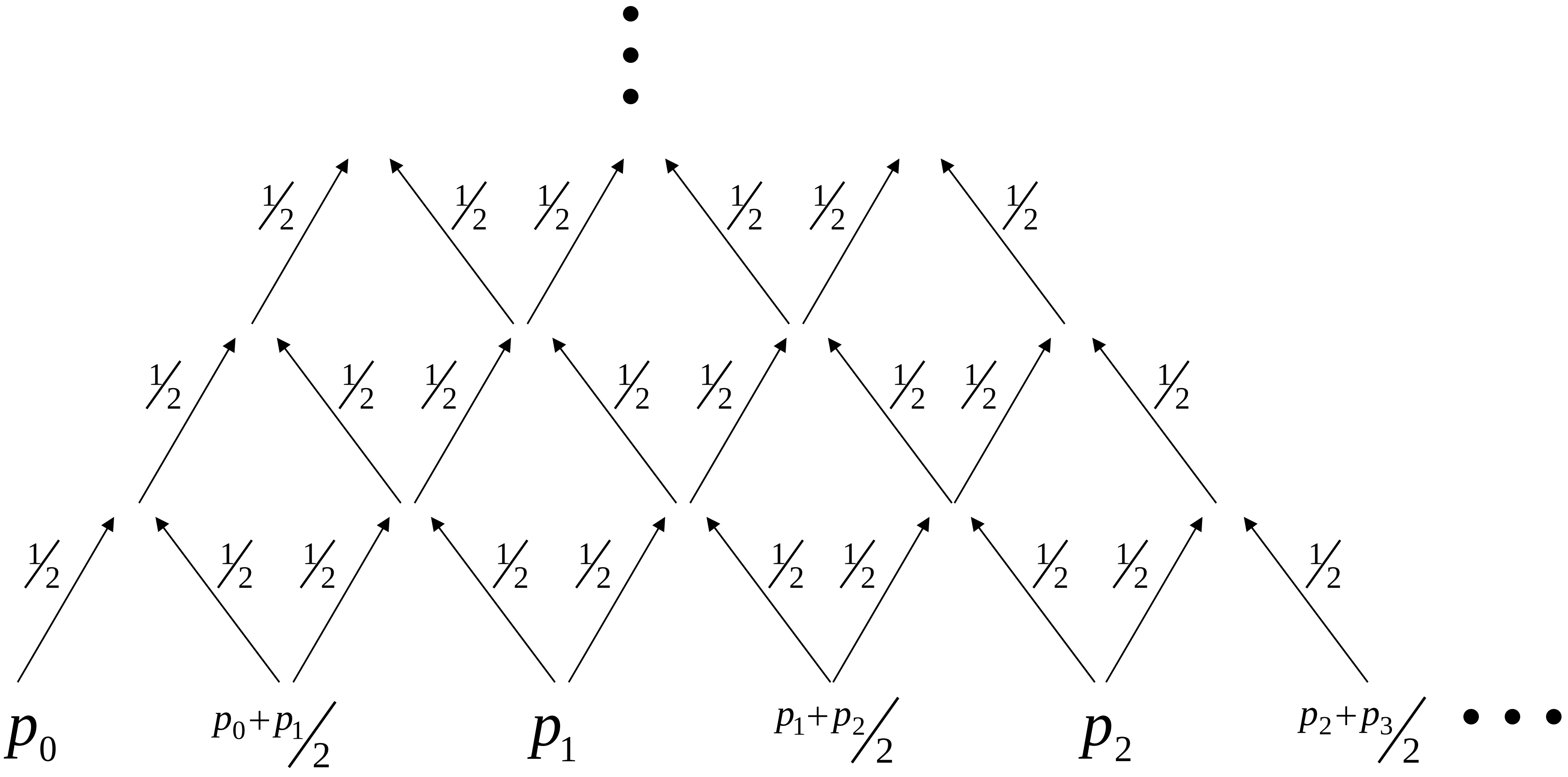
Rendering B-spline Curves

Lane-Reisenfeld subdivision algorithm

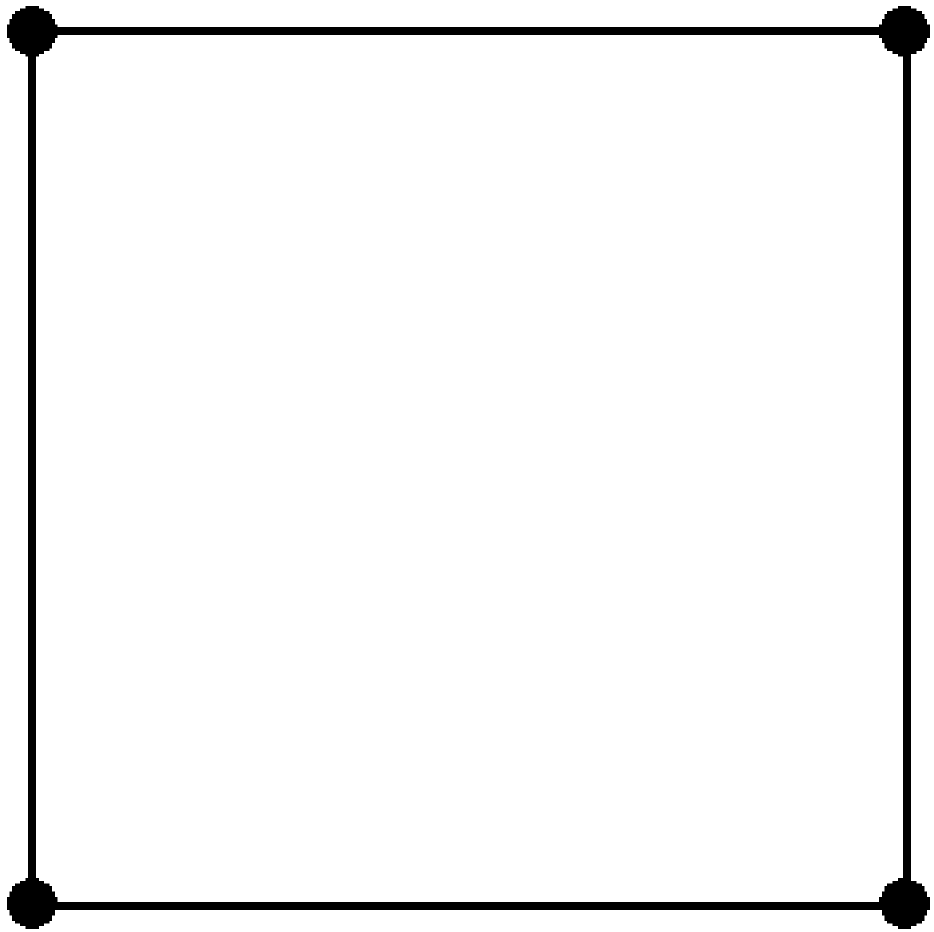
- Linearly subdivide the curve by inserting the midpoint on each edge
- Perform averaging by replacing each edge by its midpoint d times

Subdivide the curve repeatedly

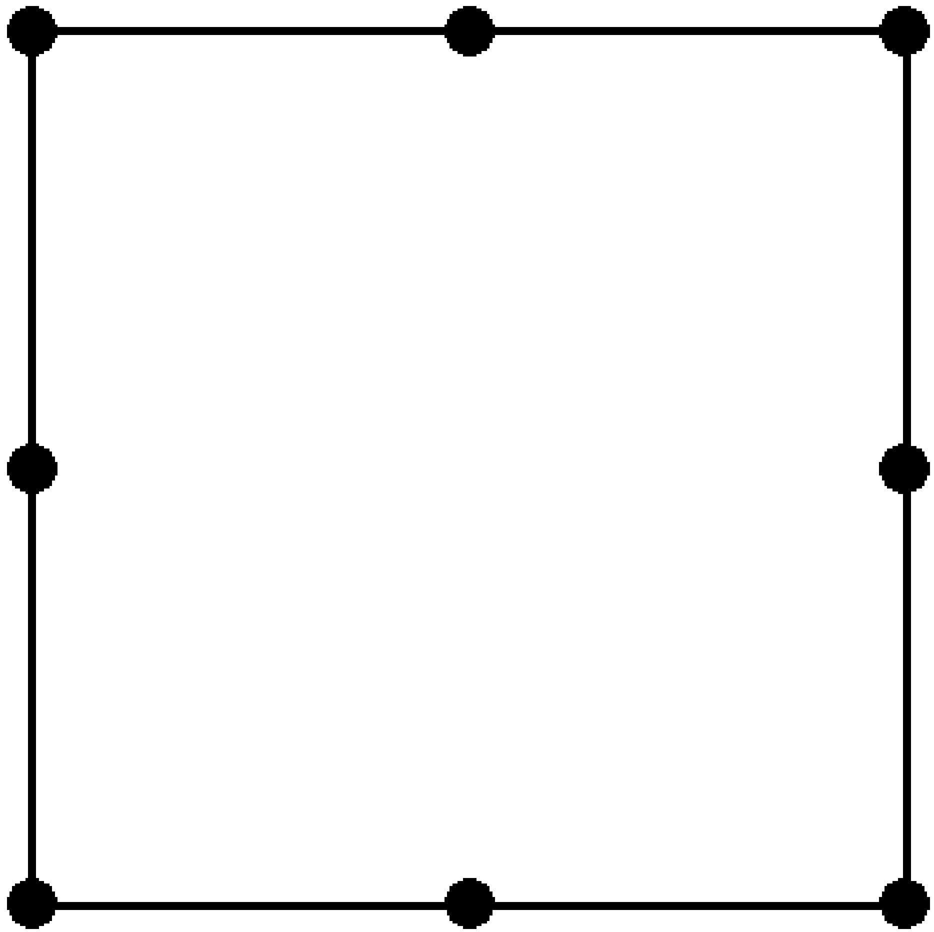
Rendering B-spline Curves



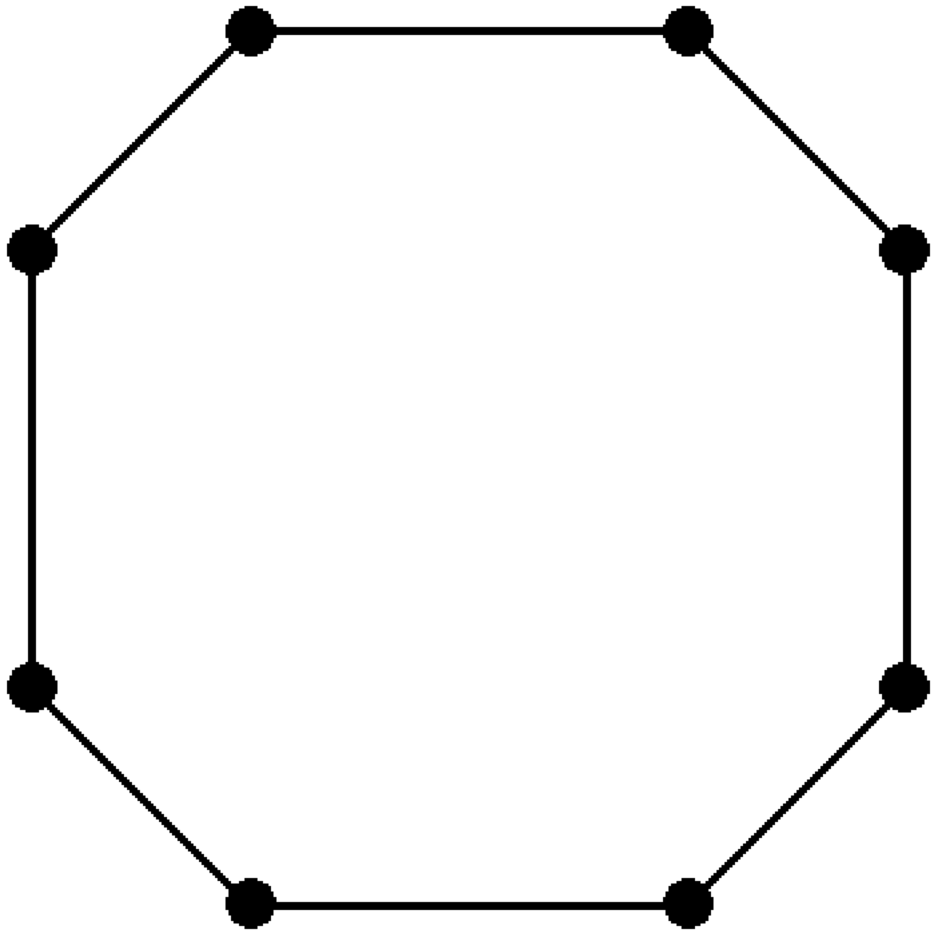
Rendering B-spline Curves



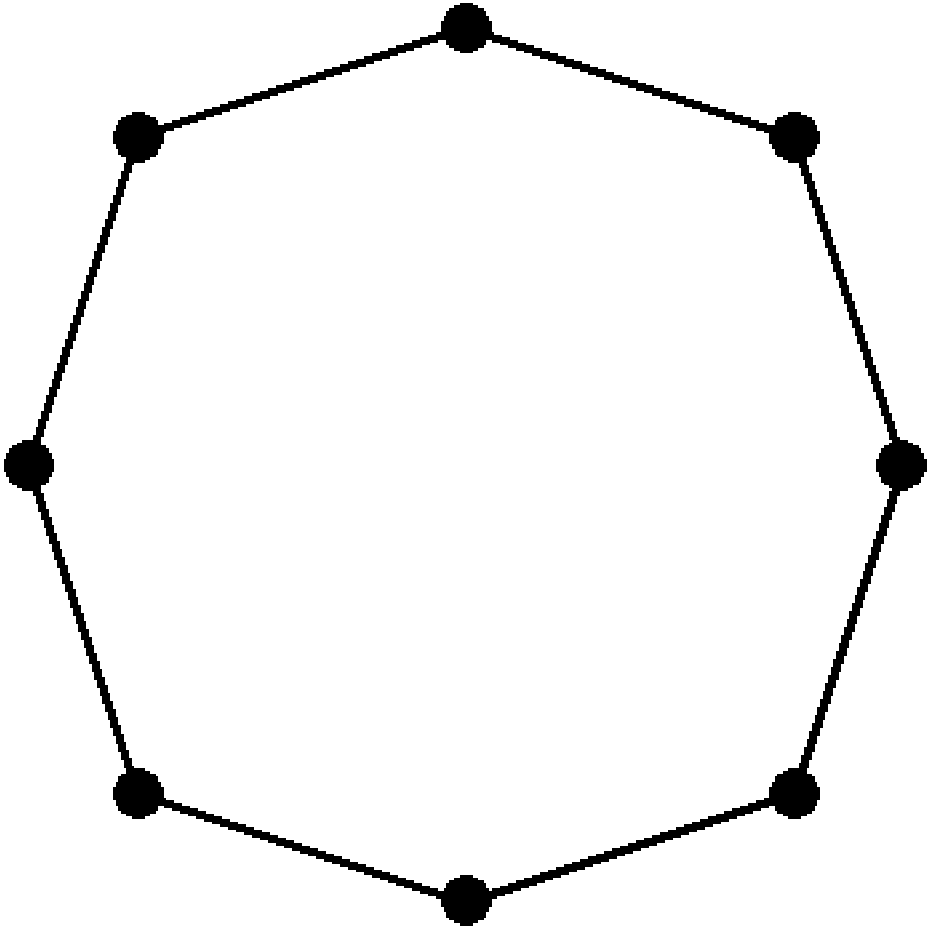
Rendering B-spline Curves



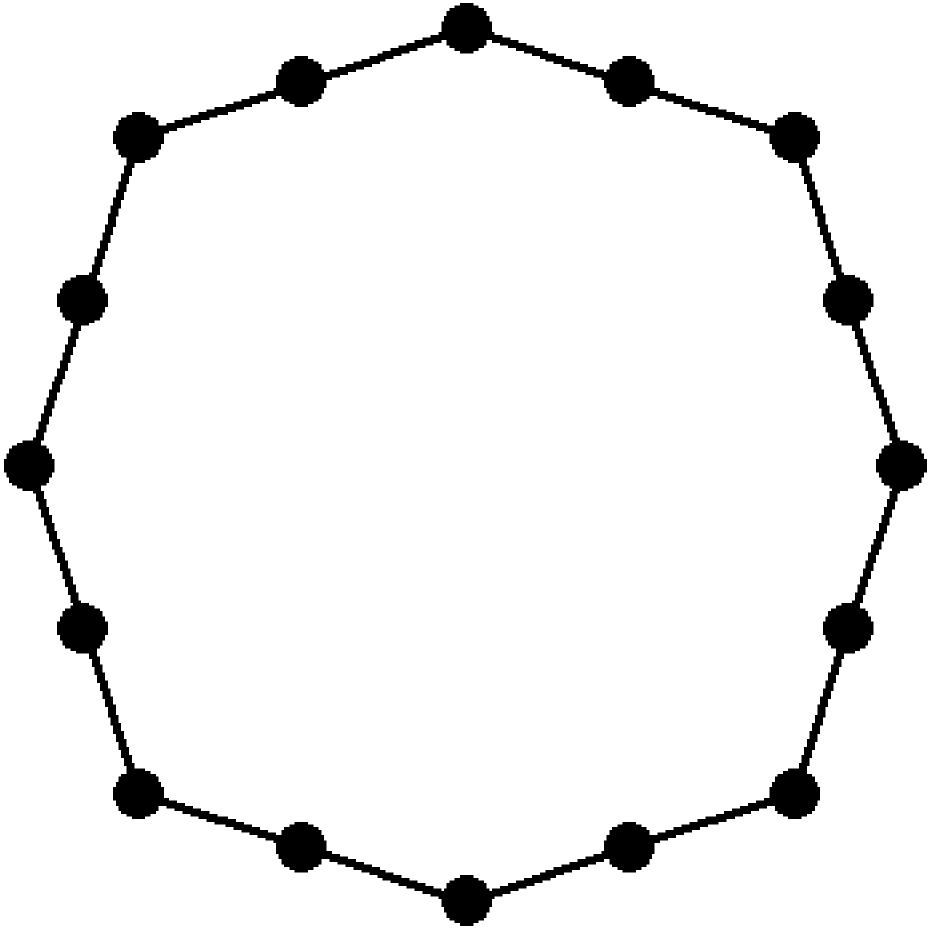
Rendering B-spline Curves



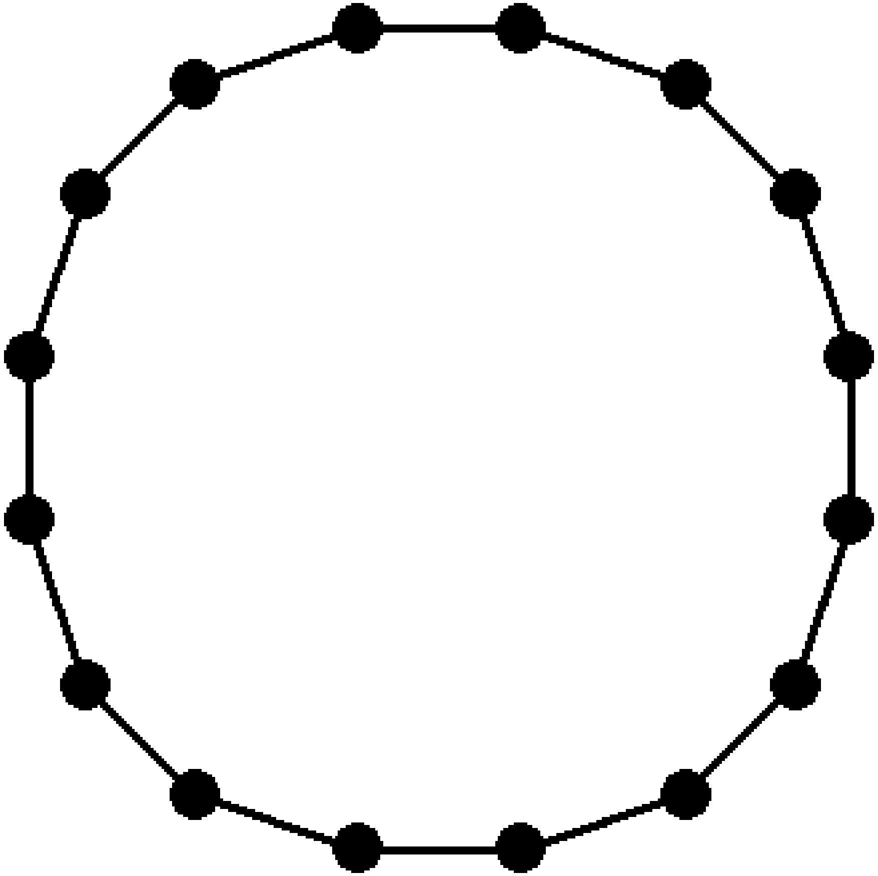
Rendering B-spline Curves



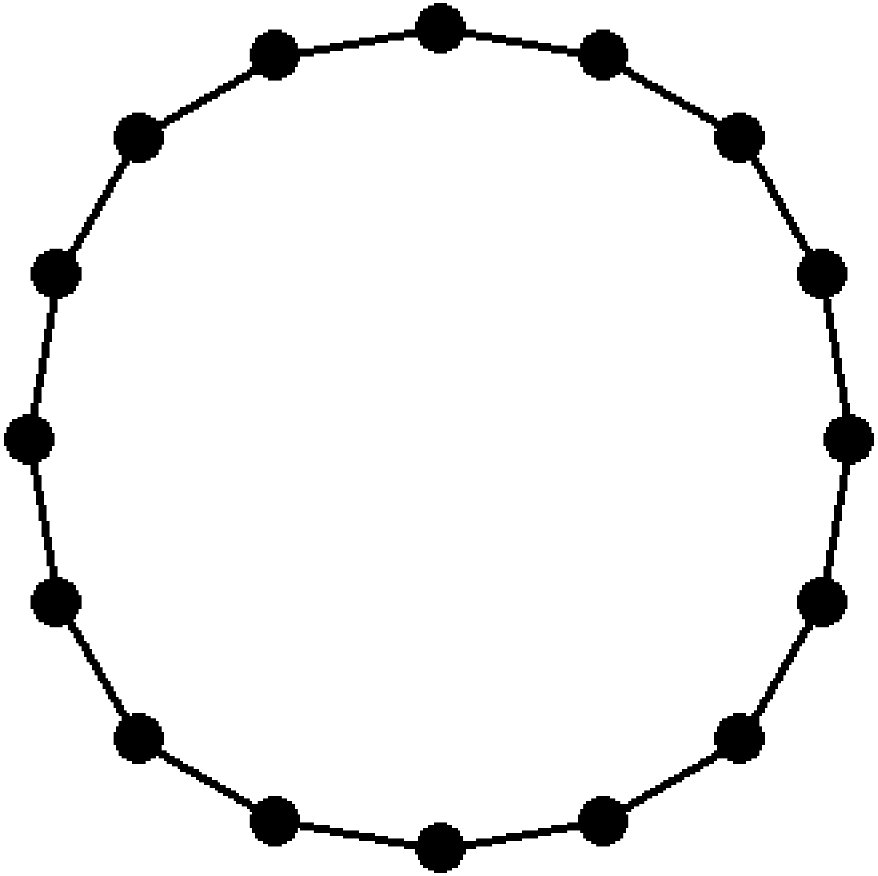
Rendering B-spline Curves



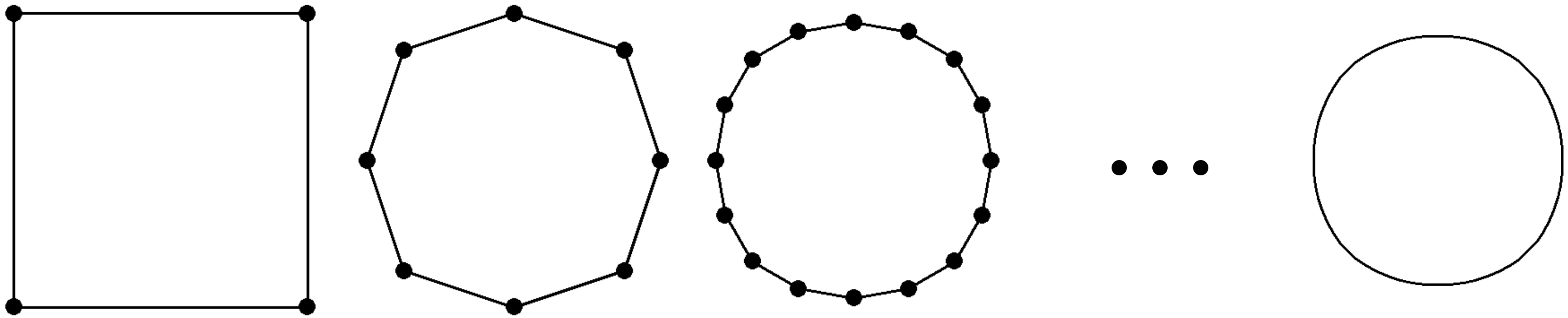
Rendering B-spline Curves



Rendering B-spline Curves



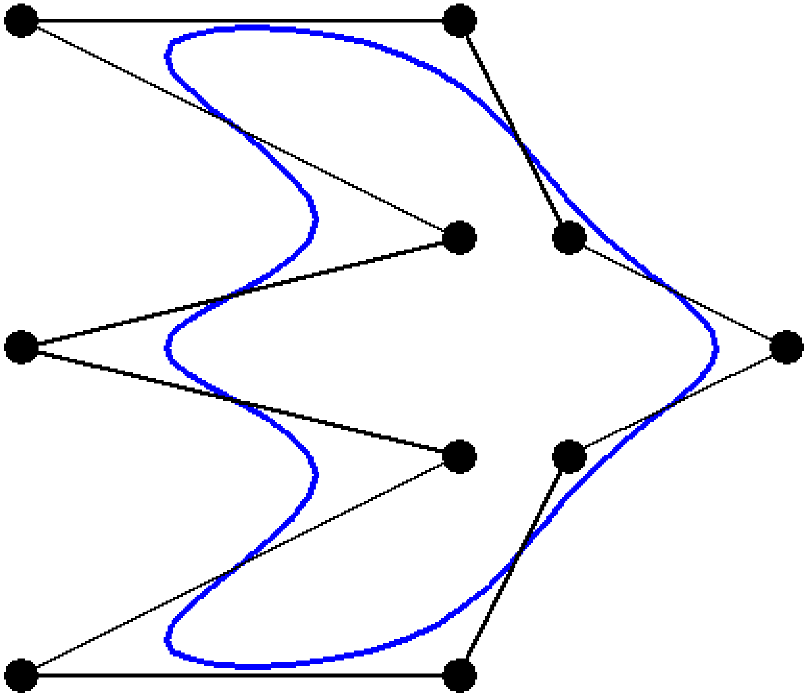
Rendering B-spline Curves



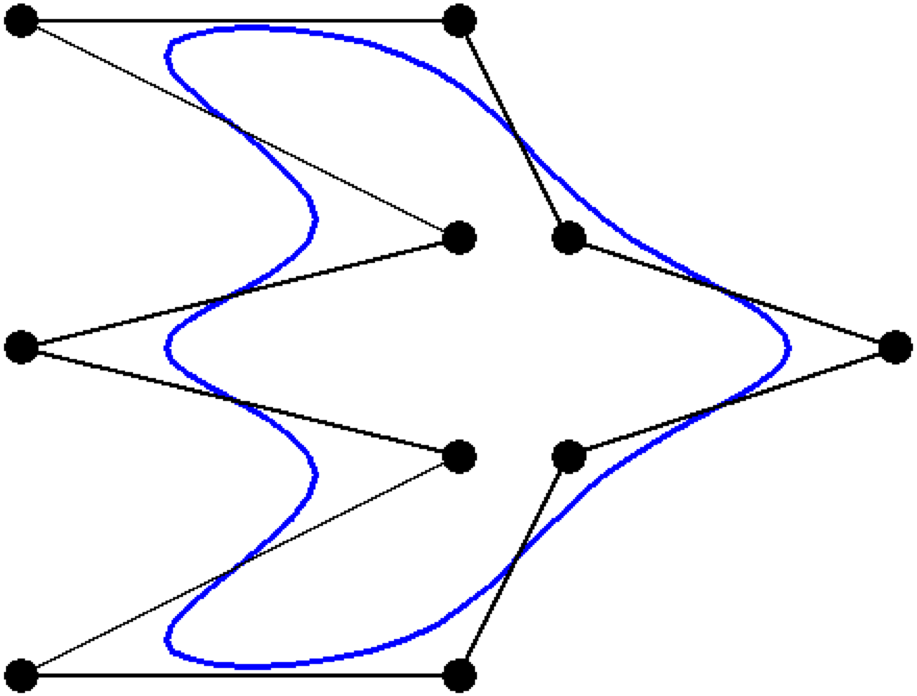
B-spline Properties

- Curve lies within convex hull of control points
- Variation Diminishing: Curve wiggles no more than control polygon
- Influence of one control point is bounded
- Degree of curve increases by one with each averaging step
- Smoothness increases by one with each averaging step

B-spline Curve Example



B-spline Curve Example



B-spline Curve Example