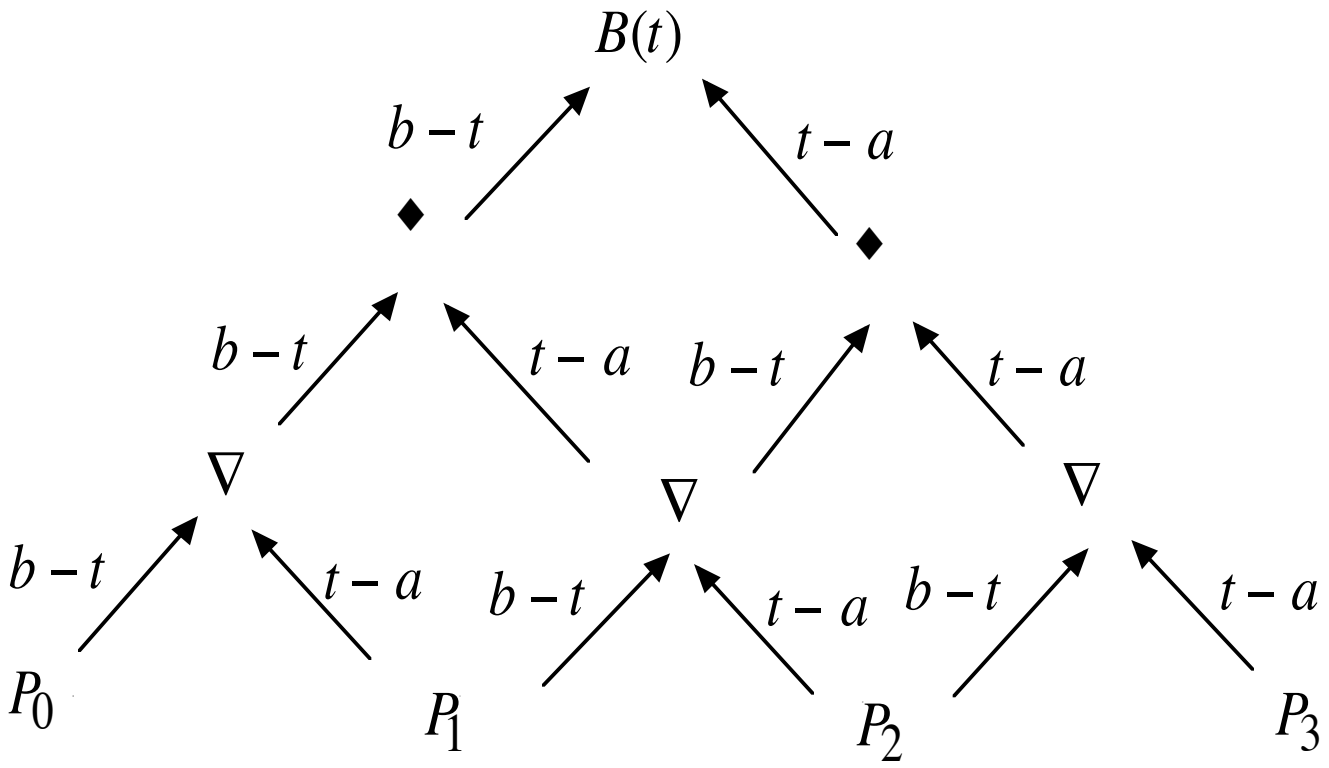
# Bezier Subdivision and De Casteljau's Algorithm

**Ron Goldman**
**Department of Computer Science**
**Rice University**

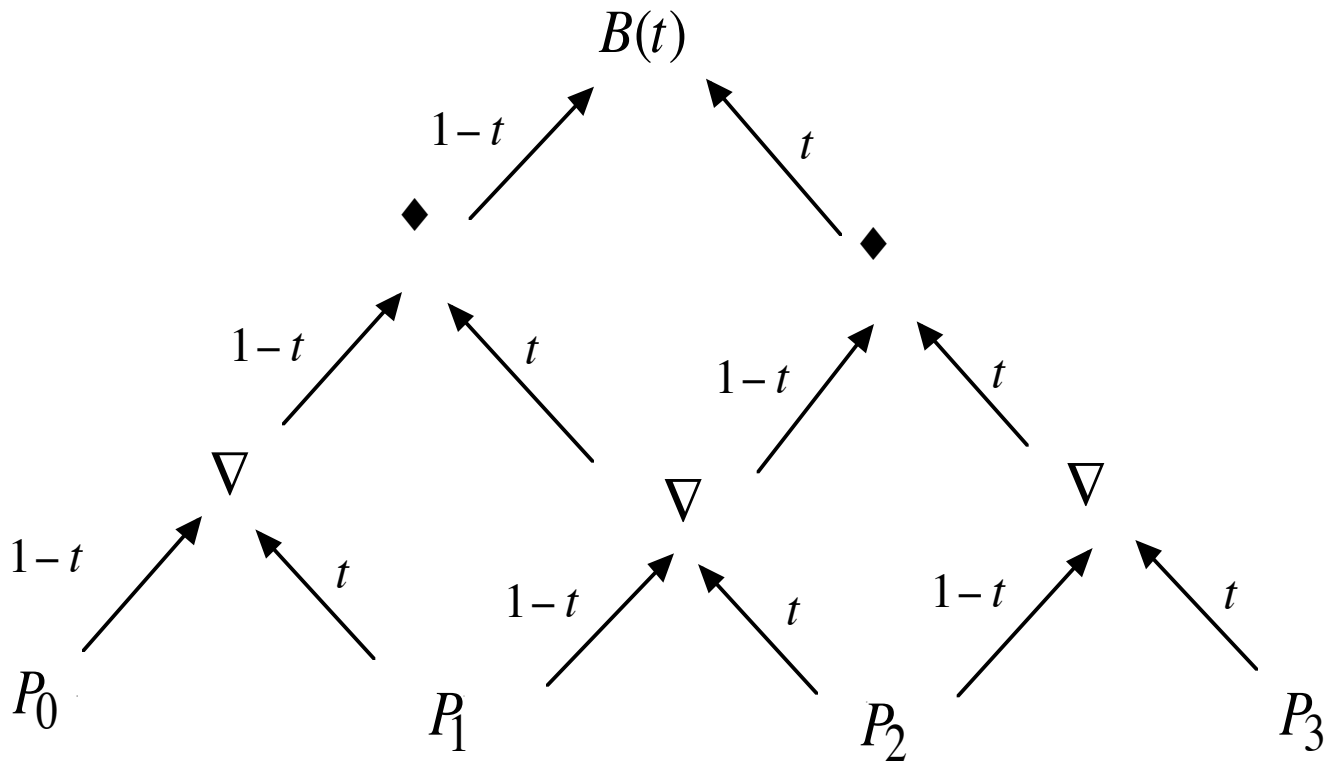
# De Casteljau's Algorithm




$B(t)$ = Bezier Curve $\quad a \le t \le b$

$P_0, \ldots, P_n$ = Control Points

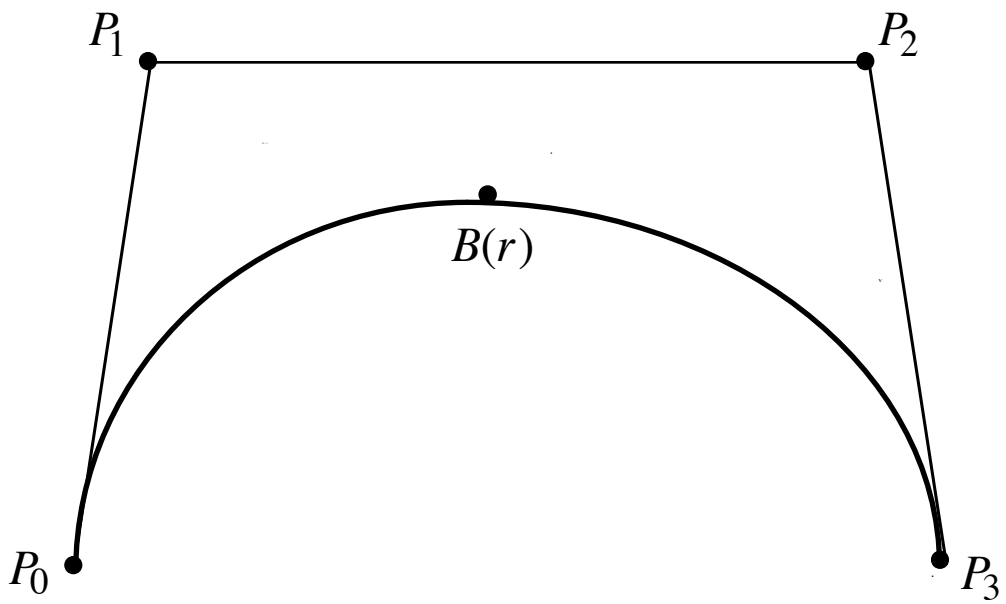
# De Casteljau's Algorithm



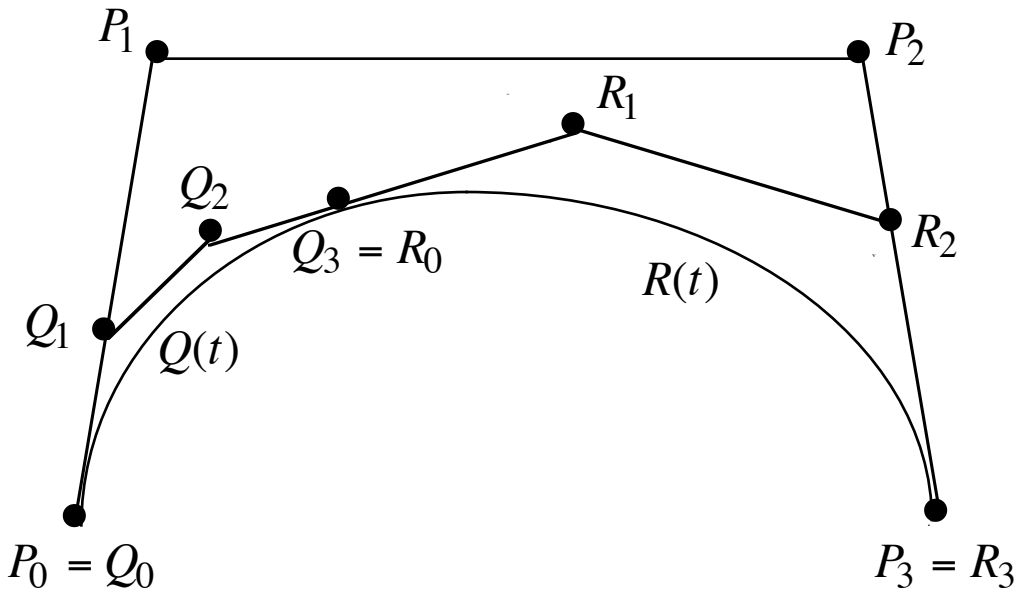

$B(t)$ = Bezier Curve $\quad 0 \le t \le 1$

$P_0, \ldots, P_n$ = Control Points

# Bezier Curve

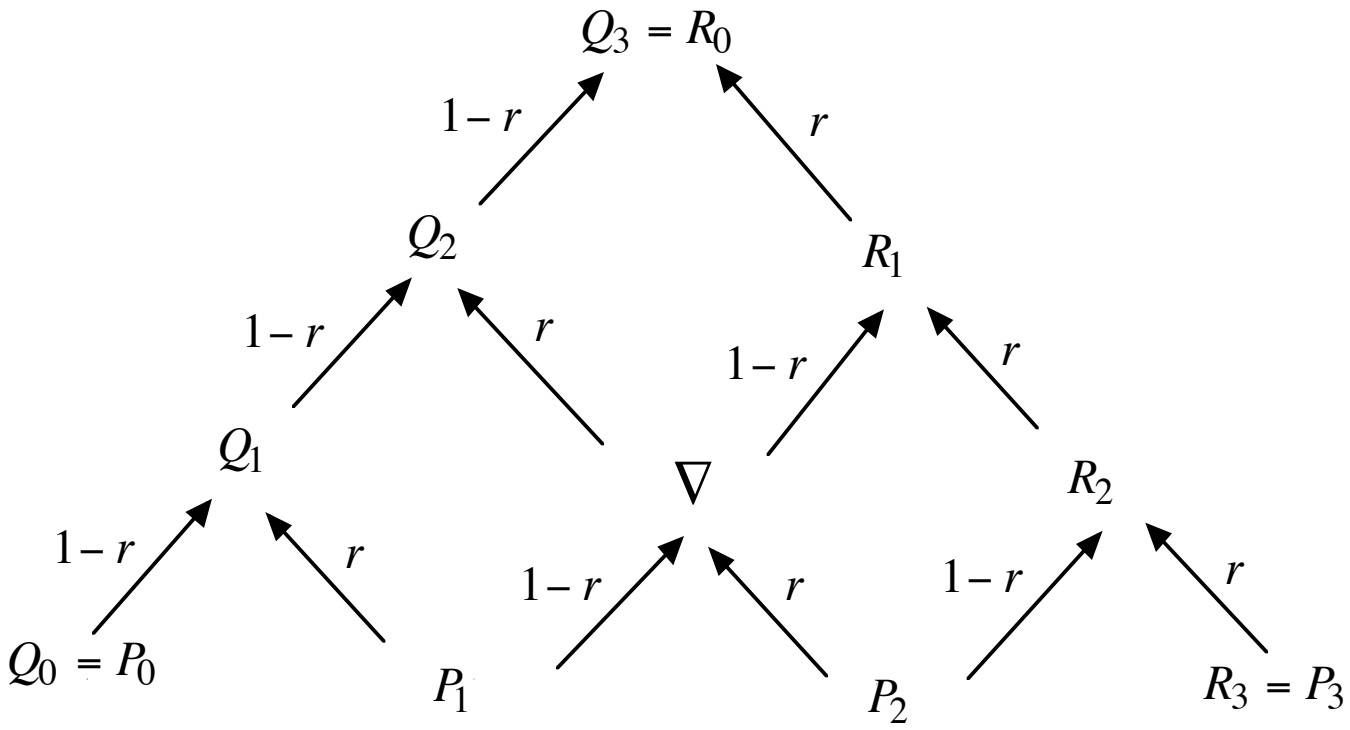


# Bezier Subdivision




*Problem*: Given $P_0, \ldots, P_n$, find $Q_0, \ldots, Q_n$ and $R_0, \ldots, R_n$.

# De Casteljau's Subdivision Algorithm




$B(t)$ = Bezier Curve $\quad 0 \le t \le 1$

$P_0, \ldots, P_n$ = Original Control Points

# Algorithms for Bezier Curves

*Rendering Algorithm*

- If the Bezier curve can be approximated to within tolerance by the straight line joining its first and last control points,
  then draw either this line segment or the control polygon.
- Otherwise *subdivide* the curve (at $r = 1/2$) and render the segments recursively.

*Intersection Algorithm*

- If the convex hulls of the control points of two Bezier curves fail to intersect,
  then the curves themselves do not intersect.
- Otherwise if each Bezier curve can be approximated by the straight line joining its first and last control points,
  then intersect these line segments.
- Otherwise *subdivide* the two curves and intersect the segments recursively.

## Questions and Answers

*Question*: When can a Bezier curve be approximated by a straight line?
*Answer*: When all the control points are within tolerance of the straight line because a Bezier curve lies in the convex hull of its control points.

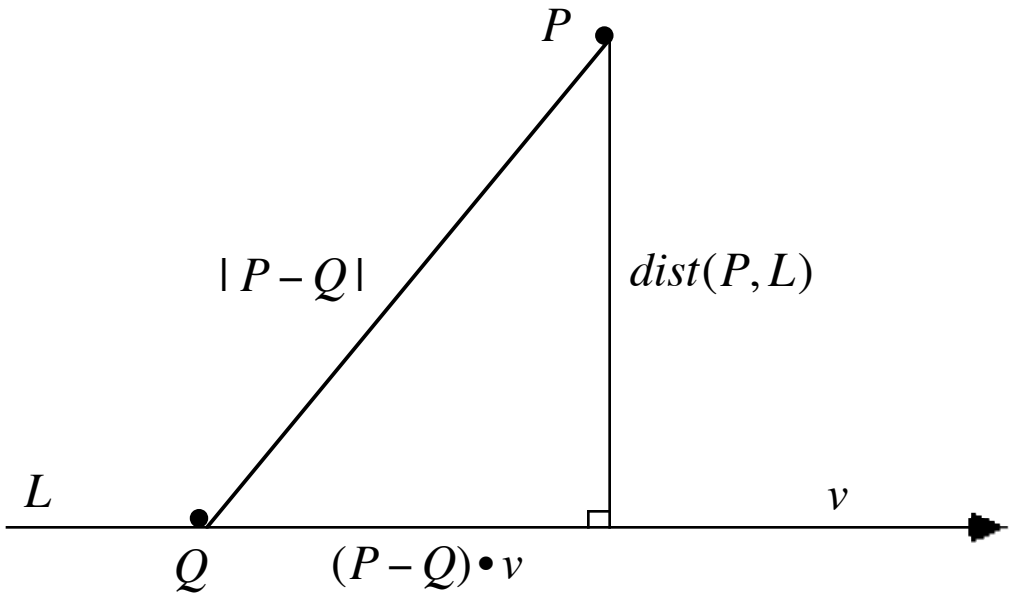
*Question*: What is the distance from a point $P$ to a line $L$?
*Answer*: $dist^2(P,L) = |P-Q|^2 - ((P-Q)\bullet v)^2$

- $Q$ = *point on* $L$
- $v$ = *unit vector parallel to* $L$

*Question*: How can we compute the convex hull of the control points?
*Answer*: Replace the convex hull by a bounding box.

*Question:* How do we compute the intersection of two line segments?
*Answer:* Use vector techniques (see below).

# Distance from a Point to a Line




$$dist^2(P,L) = |P-Q|^2 - \left((P-Q)\bullet v\right)^2$$

## Intersection Between Two Line Segments

*Given*

Line #1: $L_1(s) = (1-s)P_0 + sP_1 = P_0 + s(P_1 - P_0) \qquad 0 \le s \le 1$

Line #2: $L_2(t) = (1-t)Q_0 + tQ_1 = Q_0 + t(Q_1 - Q_0) \qquad 0 \le t \le 1$

*To Find Intersection*

$$L_1(s) = L_2(t)$$

$$P_0 + s(P_1 - P_0) = Q_0 + t(Q_1 - Q_0)$$

$$s(P_1 - P_0) - t(Q_1 - Q_0) = (Q_0 - P_0)$$

*Apply Dot Products*

$$s\{(P_1 - P_0) \bullet (P_1 - P_0)\} - t\{(Q_1 - Q_0) \bullet (P_1 - P_0)\} = (Q_0 - P_0) \bullet (P_1 - P_0)$$

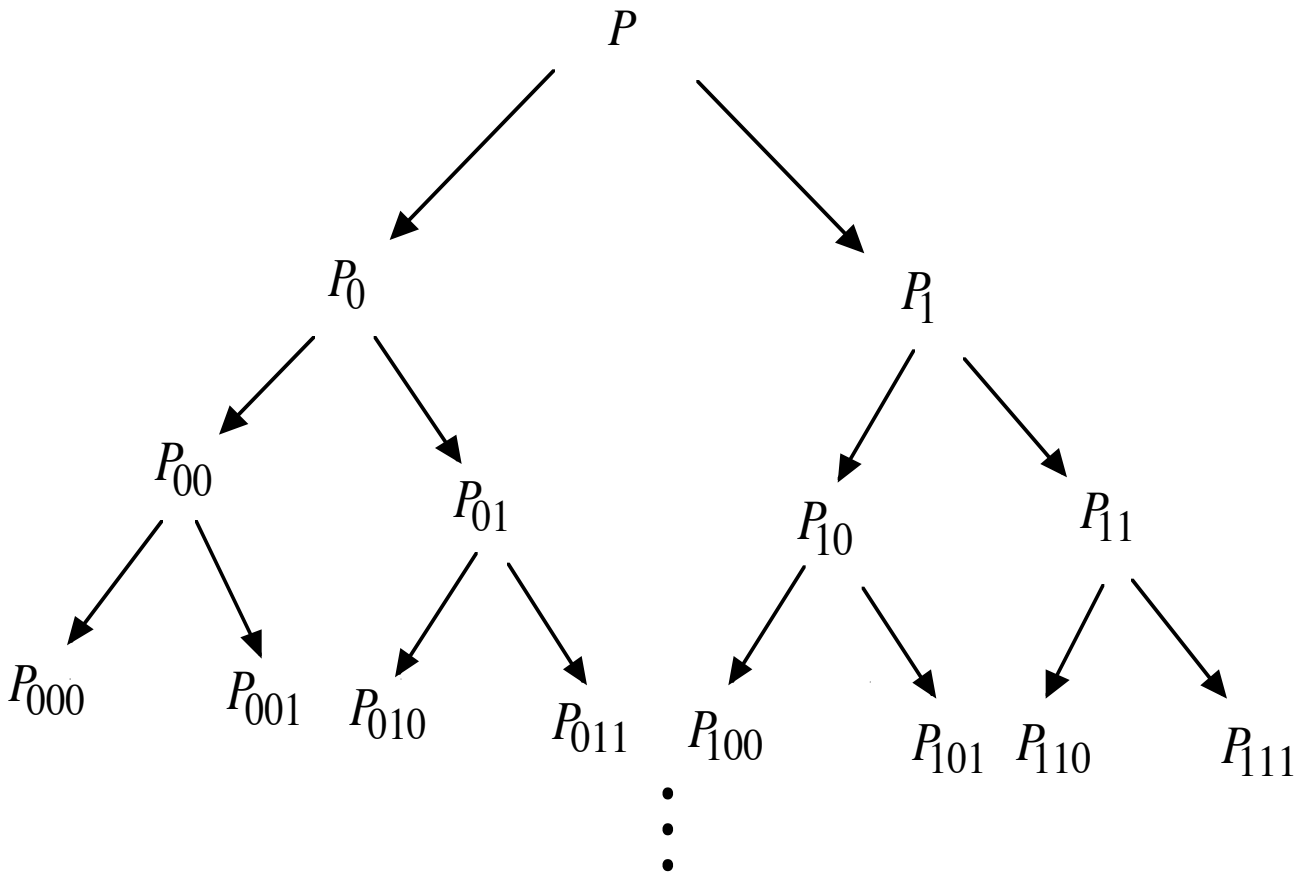
$$s\{(P_1 - P_0) \bullet (Q_1 - Q_0)\} - t\{(Q_1 - Q_0) \bullet (Q_1 - Q_0)\} = (Q_0 - P_0) \bullet (Q_1 - Q_0)$$

*Solve Two Linear Equations in Two Unknowns* $\{s,t\}$.

*Test* $0 \le s,t \le 1$ *to Determine whether the Intersection Lies on the Line Segments.*

*Substitute Back into* $L_1(s)$ *or* $L_2(t)$ *to Find the Intersection Point.*

## Recursive Subdivision




Control Polygons Generated by Recursive Subdivision

$$.b_1 \cdots b_n \rightarrow b \Rightarrow P_{b_1 \cdots b_n} \rightarrow B(b)$$

**Theorem:** *The control polygons generated by recursive subdivision converge to the original Bezier curve.*

Proof: Let $d$ = the maximum distance between any two adjacent control points.

The points on any level of the de Casteljau algorithm for $t = 1/2$ lie at the midpoints of the edges of the polygons generated by the previous level.

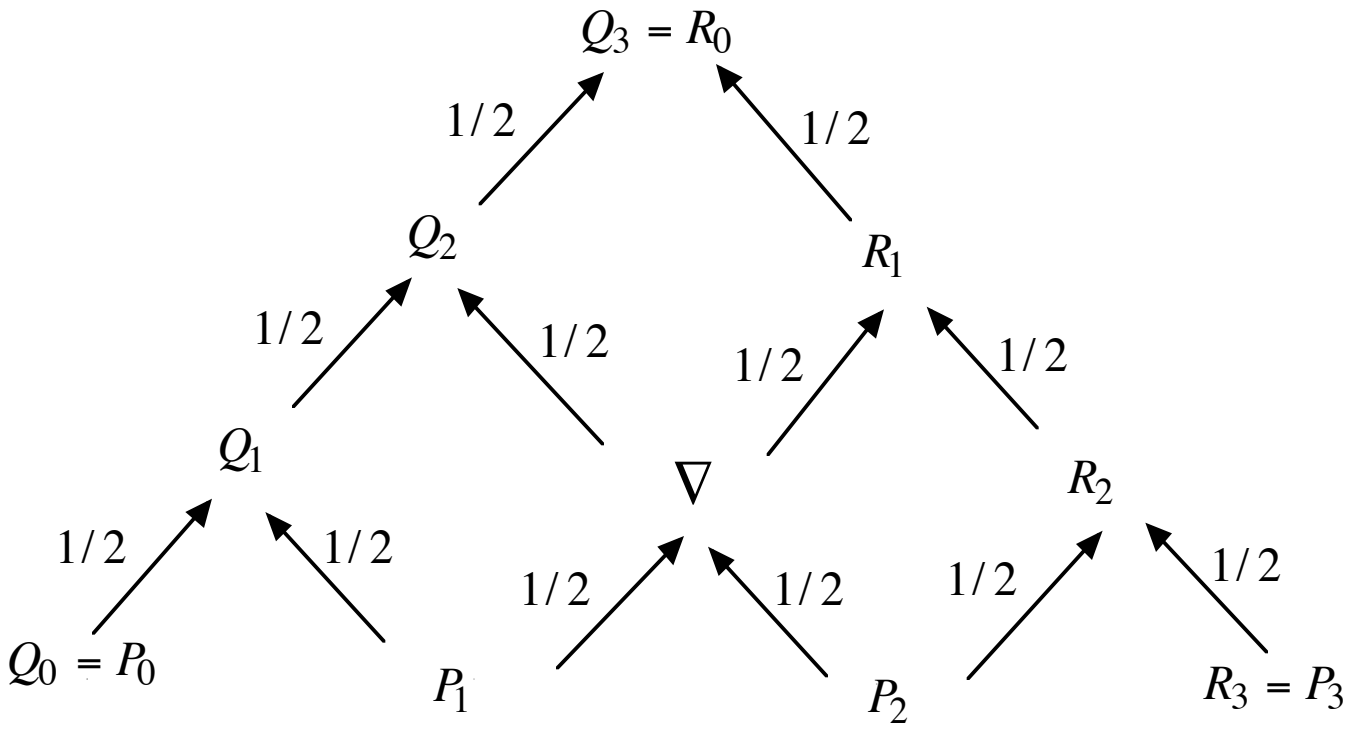
Therefore, by induction, adjacent points on any level of the de Casteljau diagram for $t = 1/2$ are no further than $d$ apart.

By the same midpoint argument, as we proceed up the diagram, adjacent points along the left (right) lateral edge of the triangle can be no further than $d/2$ apart.

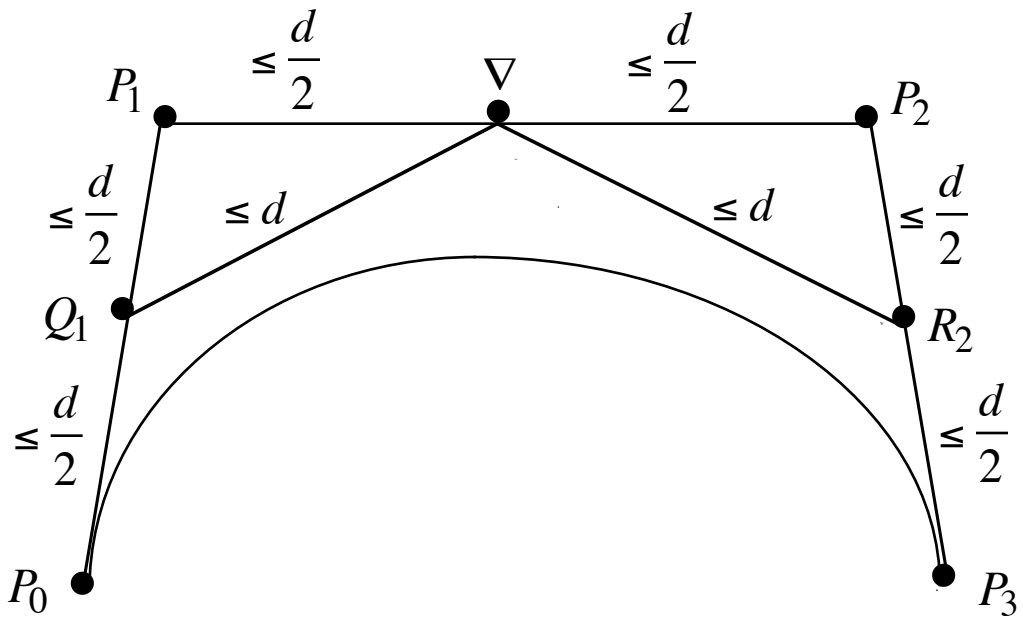
Hence, as we apply recursive subdivision, the distance between the control points of any single control polygon must converge to zero.

Since the first and last control points of a Bezier control polygon always lie on the curve, these control polygons must converge to points along the original curve.

# De Casteljau's Subdivision Algorithm



# One Level of the de Casteljau Algorithm



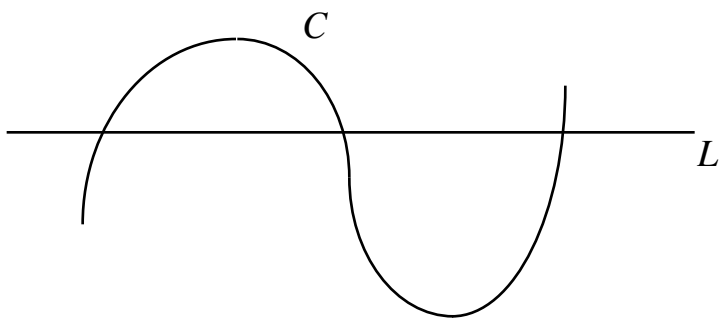
## Variation Diminishing Property

*Theorem: Bezier curves never oscillate more than their control points.*

*Remark: Lagrange interpolating curves often oscillate more than the data points.*

*Question: How do we measure oscillations?*

*Answer: Compute intersections with a straight line.*



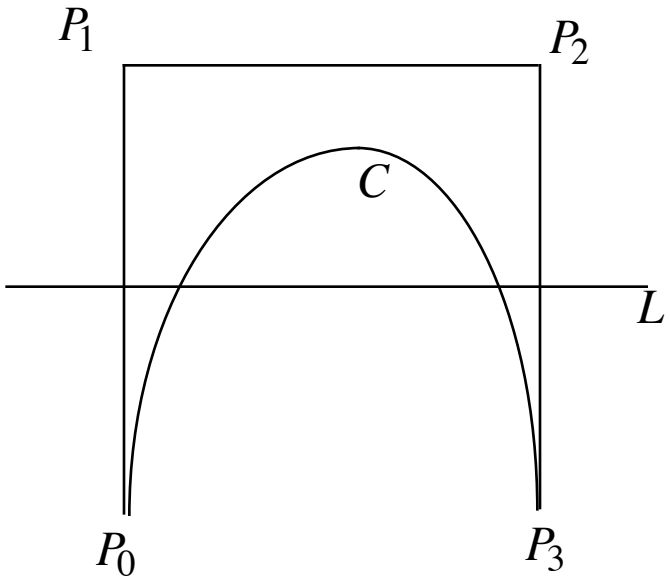
# Variation Diminishing Curves

*Definition*

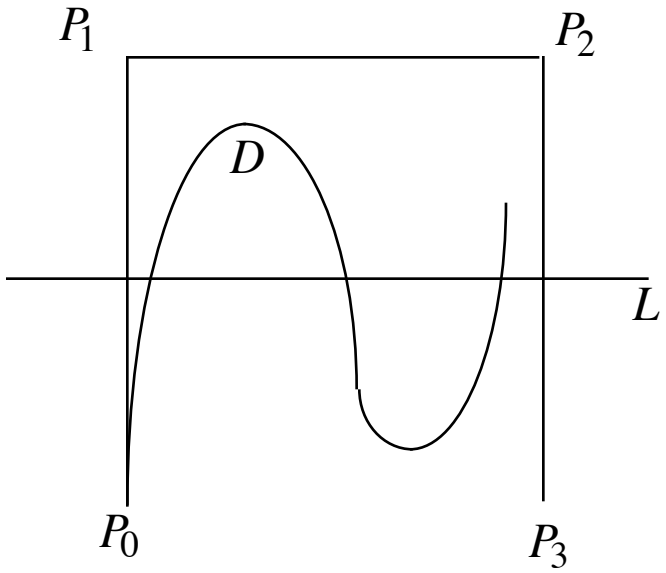
A curve scheme $B(t)$ is said to be *variation diminishing* if for every line $L$

# intersections of $B(t)$ and $L \leq$ # intersections of the control polygon and $L$ .

*Examples*

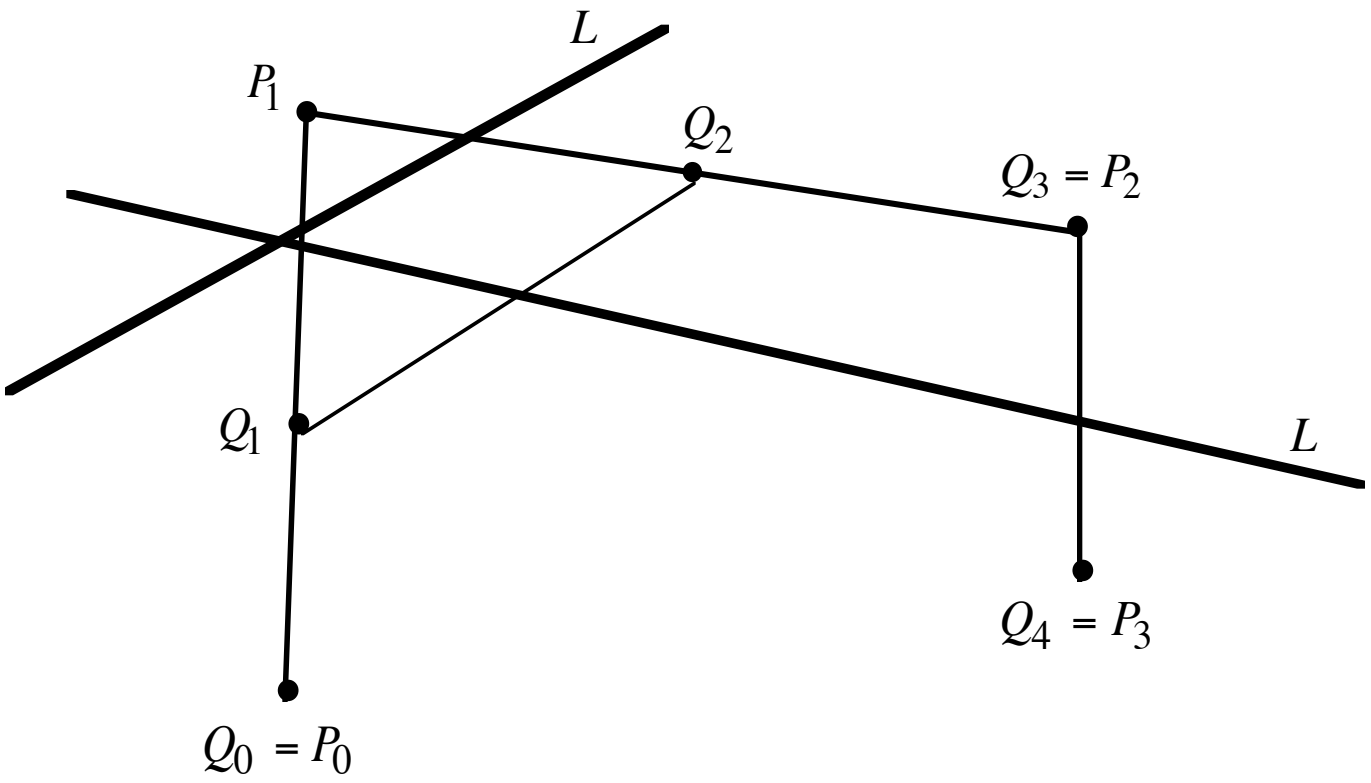



*Variation Diminishing*



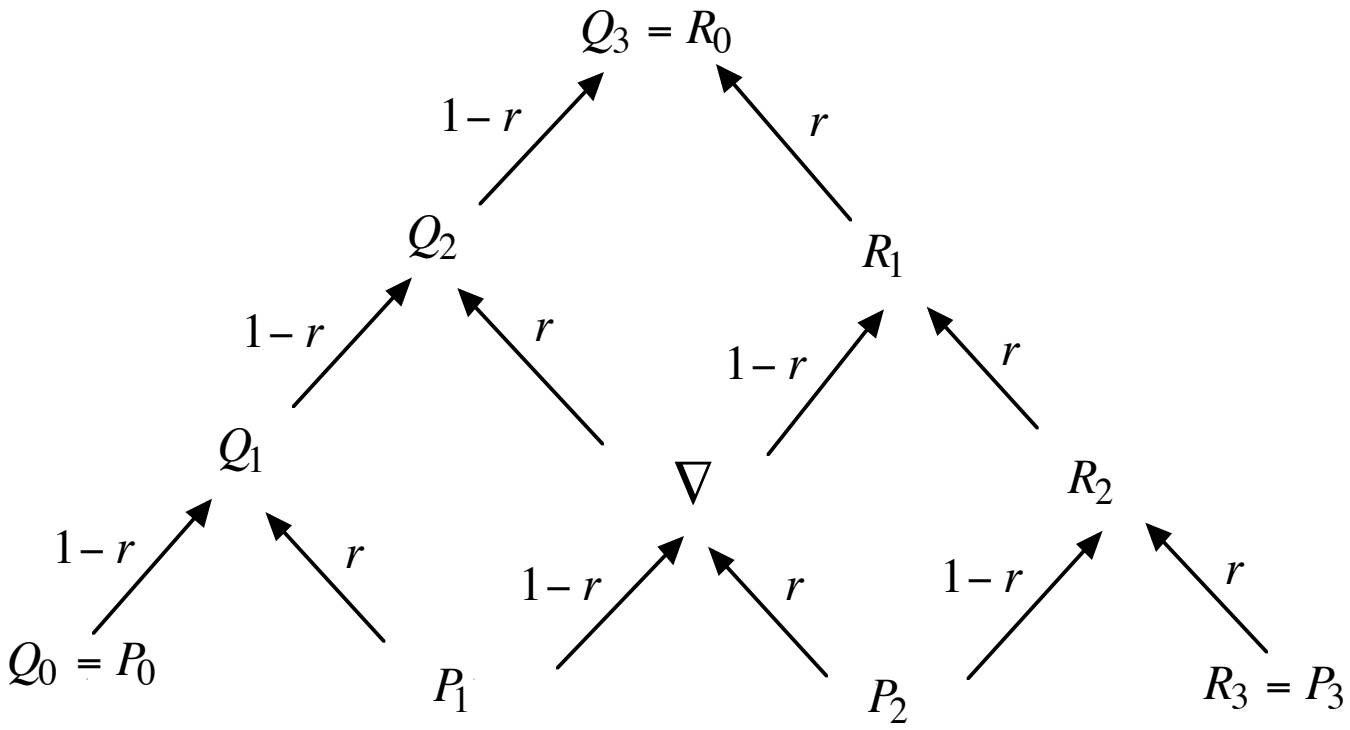

*Not Variation Diminishing*

# Corner Cutting

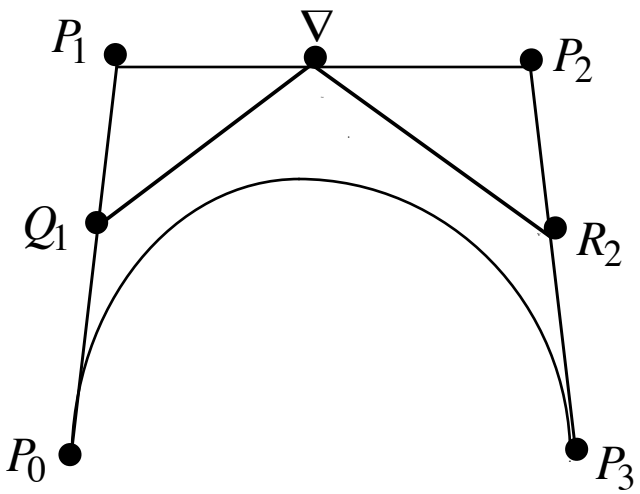



*#intersection of L and Q ≤ #intersection of L and P*

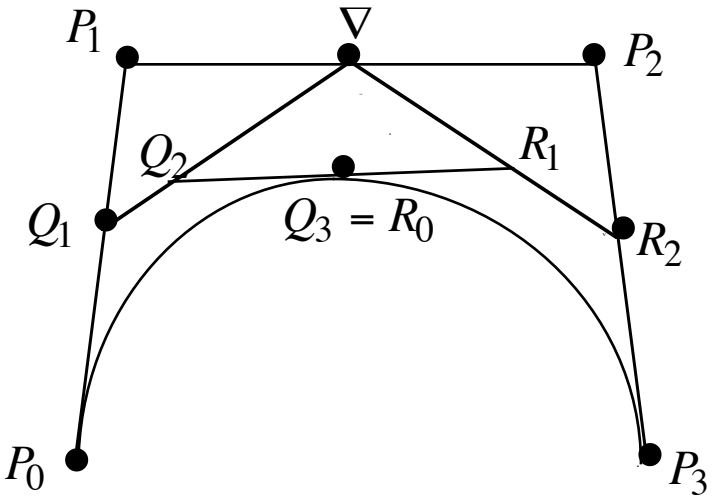
# De Casteljau's Subdivision Algorithm



# de Casteljau Subdivision and Corner Cutting




*First step*




*Second step*

**Corollary:** *Bezier curves are variation diminishing.*

Proof: Since recursive subdivision is a corner cutting procedure, the limit curve must be variation diminishing with respect to the original control polygon.

But we have proved that the Bezier curve is the limit curve of recursive subdivision.

Hence Bezier curves are variation diminishing.

**Corollary:** *The arc length of a Bezier curve is always less than or equal to the length of its control polygon.*

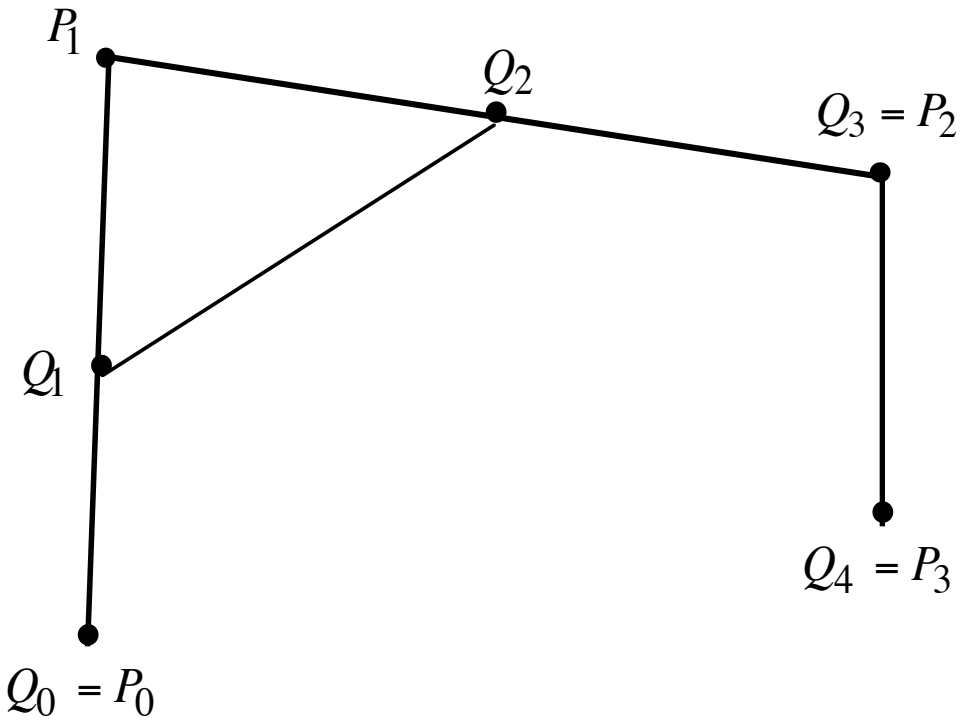
Proof: Corner cutting reduces length.

Since recursive subdivision is a corner cutting procedure, the arc length of the limit curve must be must be less than or equal to the length of the control polygon.

But we have proved that the Bezier curve is the limit curve of recursive subdivision.

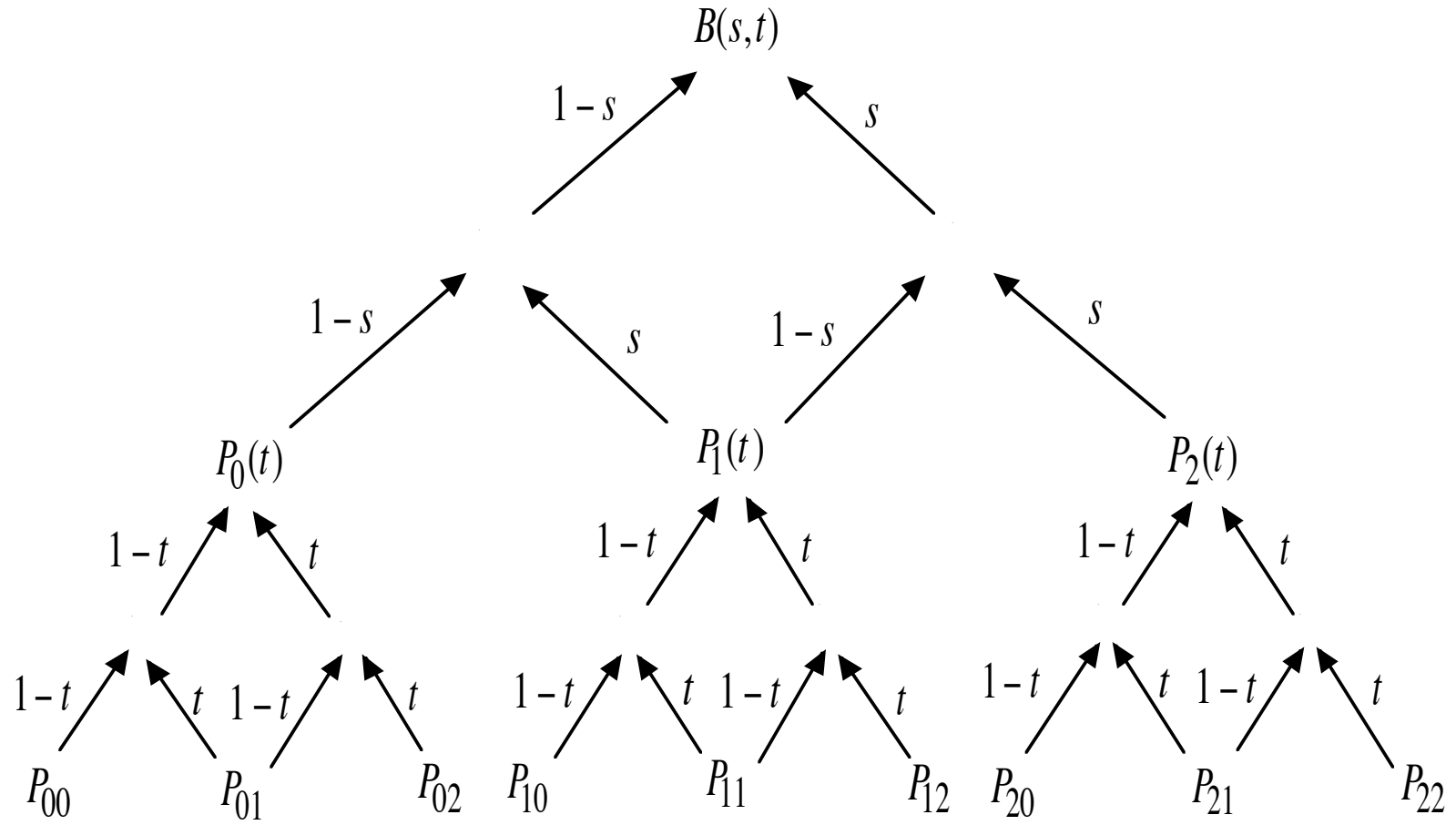
Hence the arc length of a Bezier curve is always less than or equal to the length of its control polygon.

# Corner Cutting



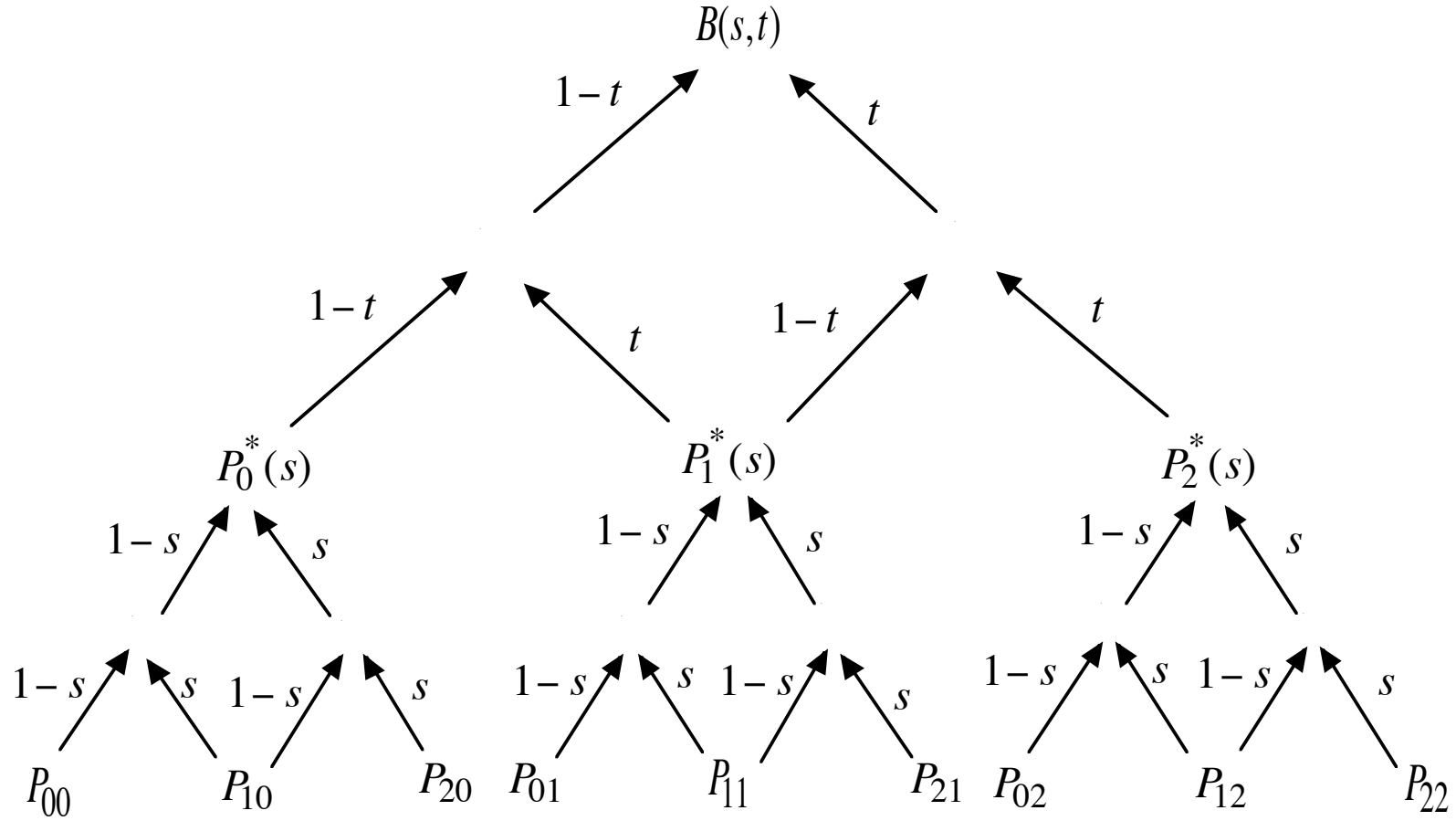

*Corner Cutting Reduces Length*

# Recursive Subdivision for Tensor Product Bezier Surfaces




*Subdivide each of the rail curves $P_k(t)$ using the de Casteljau subdivision algorithm for Bezier curves.*

# Recursive Subdivision for Tensor Product Bezier Surfaces




*Subdivide each of the rail curves $P_k^*(s)$ using the de Casteljau subdivision algorithm for Bezier curves.*

# Algorithms for Bezier Surfaces

*Rendering Algorithm*

- If the Bezier surface can be approximated to within tolerance by two triangles joining three of its four corner points, then draw the two triangles determined by the four corner points.
- Otherwise *subdivide* the surface (at $t = 1/2$) and render the segments recursively.

*Intersection Algorithm*

- If the convex hulls of the control points of two Bezier surfaces fail to intersect,
  then the surfaces themselves do not intersect.
- Otherwise if each Bezier surface can be approximated by two triangles joining three of its four corner points,
  then intersect the corresponding triangles.
- Otherwise *subdivide* the two surfaces and intersect the segments recursively.

## Questions and Answers

*Question*: When can a Bezier surface be approximated by two planes?
*Answer*: When all the control points are within tolerance of the two planes because a Bezier surface lies in the convex hull of its control points.

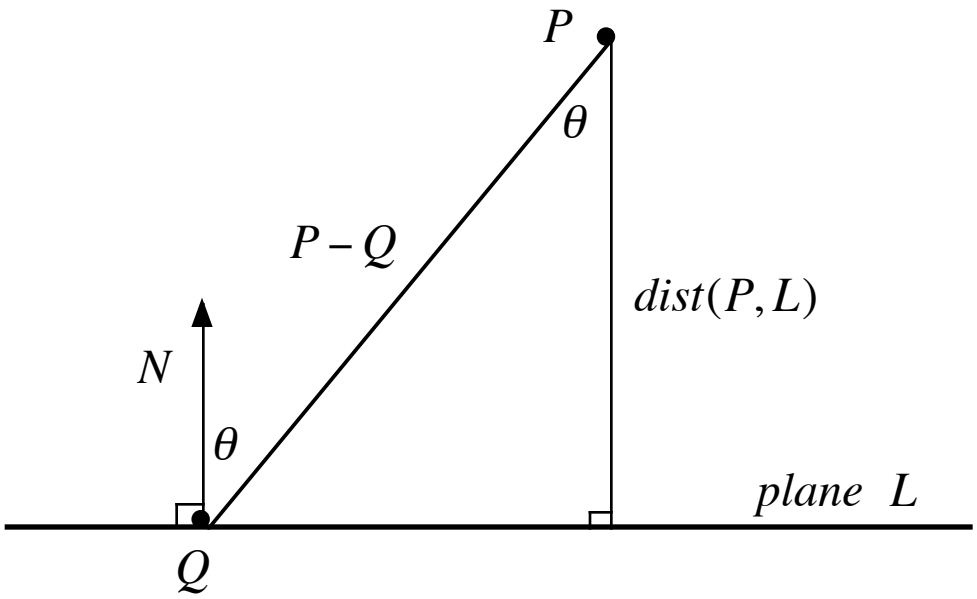
*Question*: What is the distance from a point $P$ to a plane $L$?
*Answer*: $dist(P, Plane) = |(P - Q) \bullet N|$

- $Q$ = *point on plane*
- $N$ = *unit vector normal to plane*

*Question*: How can we compute the convex hull of the control points?
*Answer*: Replace the convex hull by a bounding box.

*Question:* How do we compute the intersection of two planes?
*Answer:* Use vector techniques (see below).

# Distance from a Point to a Plane




$$dist(P,L) = |P - Q| \cos\theta = (P - Q) \bullet N$$

# Intersection Between Two Planes

*Given*

Plane #1: $N_1 \bullet (P - Q_1) = 0$

Plane #2: $N_2 \bullet (P - Q_2) = 0$

*To Find Intersection Line*

*Direction Vector* $= N_1 \times N_2$

*Point on Line*

- *Solve 2 Equations in 3 Unknowns {P = (x, y, z)}*

  $N_1 \bullet (P - Q_1) = 0$

  $N_2 \bullet (P - Q_2) = 0$

- *Solve 3 Equations in 3 Unknowns {P = (x, y, z)}*

  $N_1 \bullet (P - Q_1) = 0$

  $N_2 \bullet (P - Q_2) = 0$

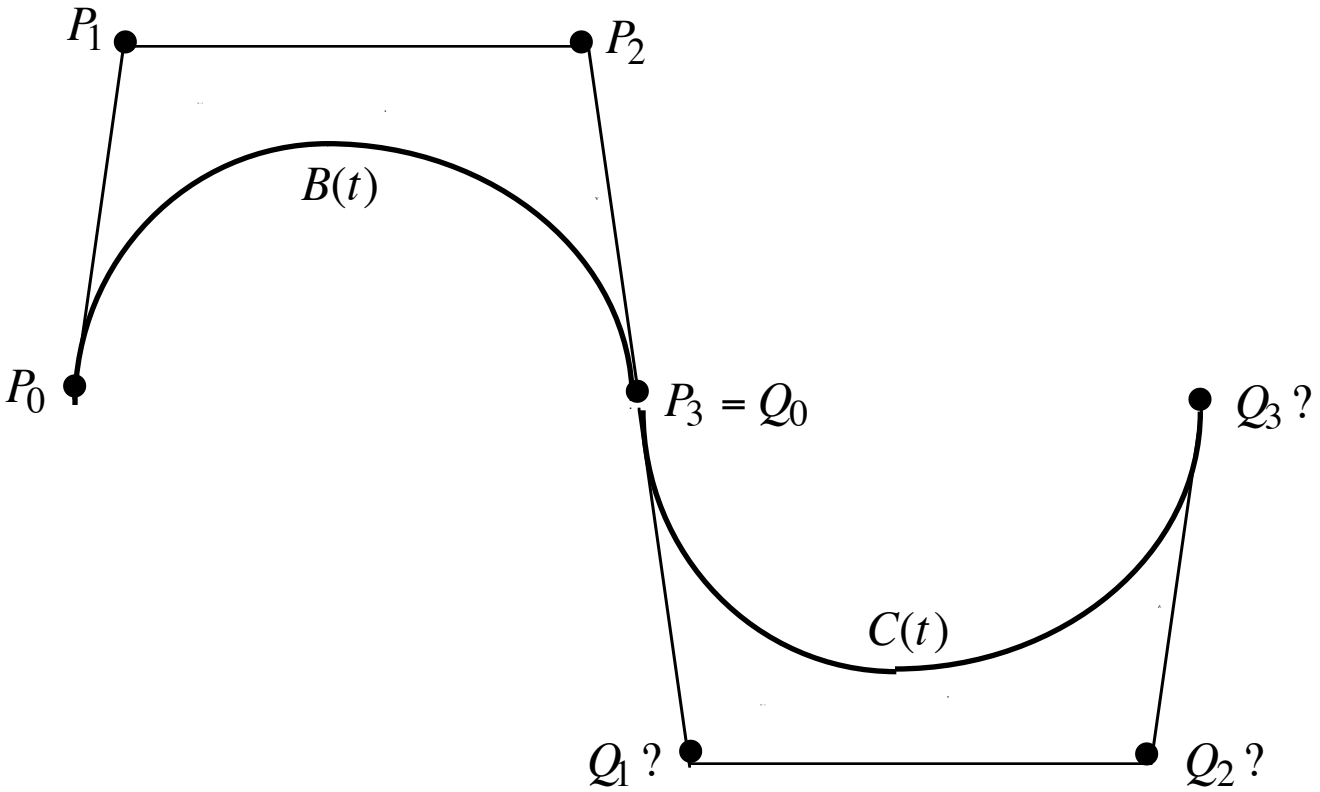
  $(N_1 \times N_2) \bullet (P - Q_3) = 0$

## Observations

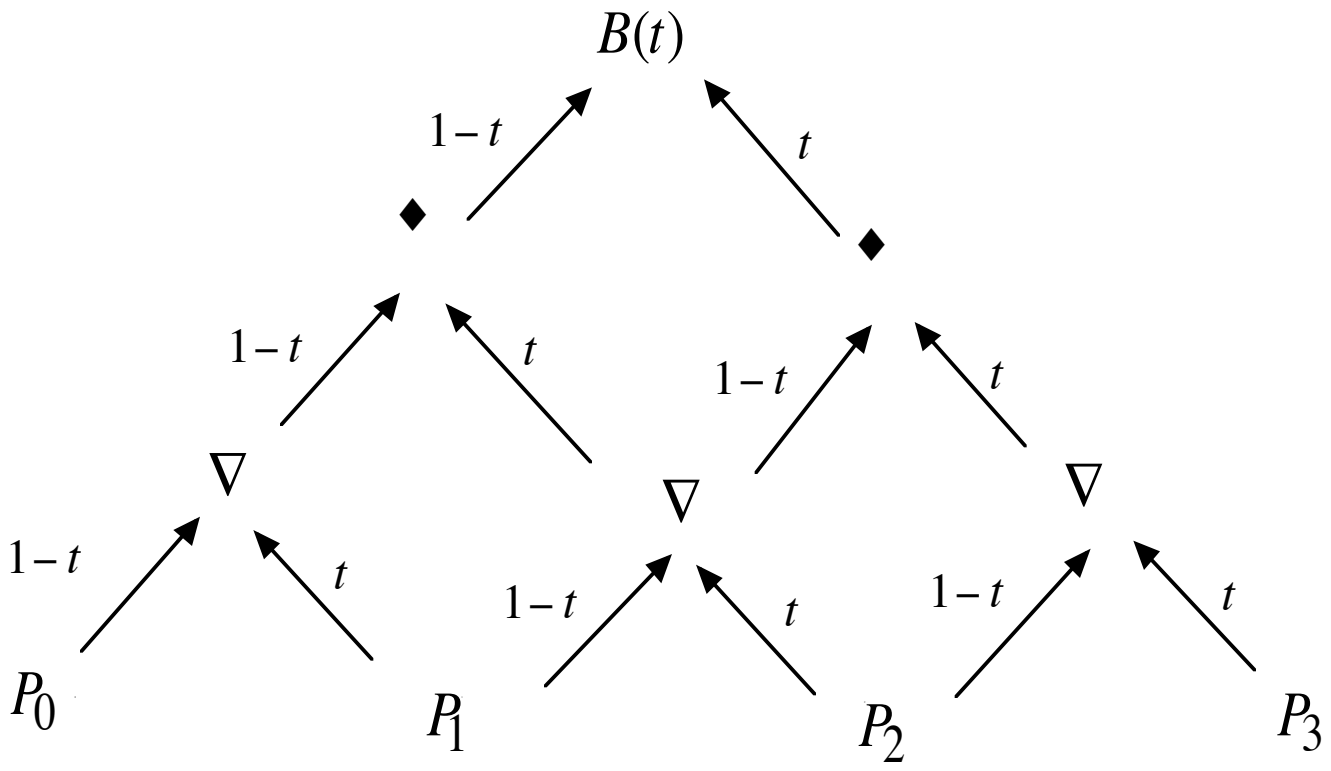
**Theorem:** *The control polyhedra generated by recursive subdivision converge to the original tensor product Bezier surface provided that the subdivision is done in both the s and t directions.*

**Remark:** *There is No Known Variation Diminishing Property for Tensor Product Bezier Surfaces.*

# Smooth Piecewise Bezier Curves



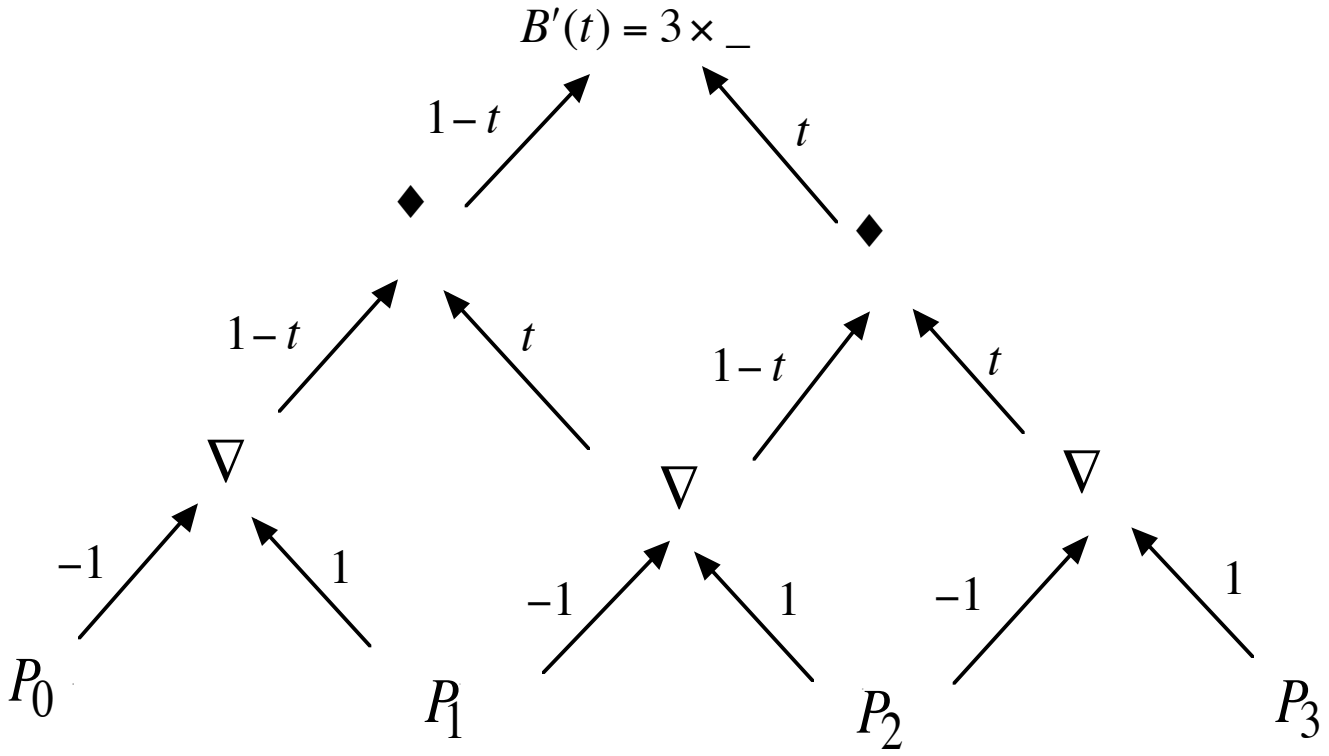
# De Casteljau's Evaluation Algorithm




$B(t)$ = Bezier Curve $\quad 0 \le t \le 1$

$P_0, \ldots, P_n$ = Control Points

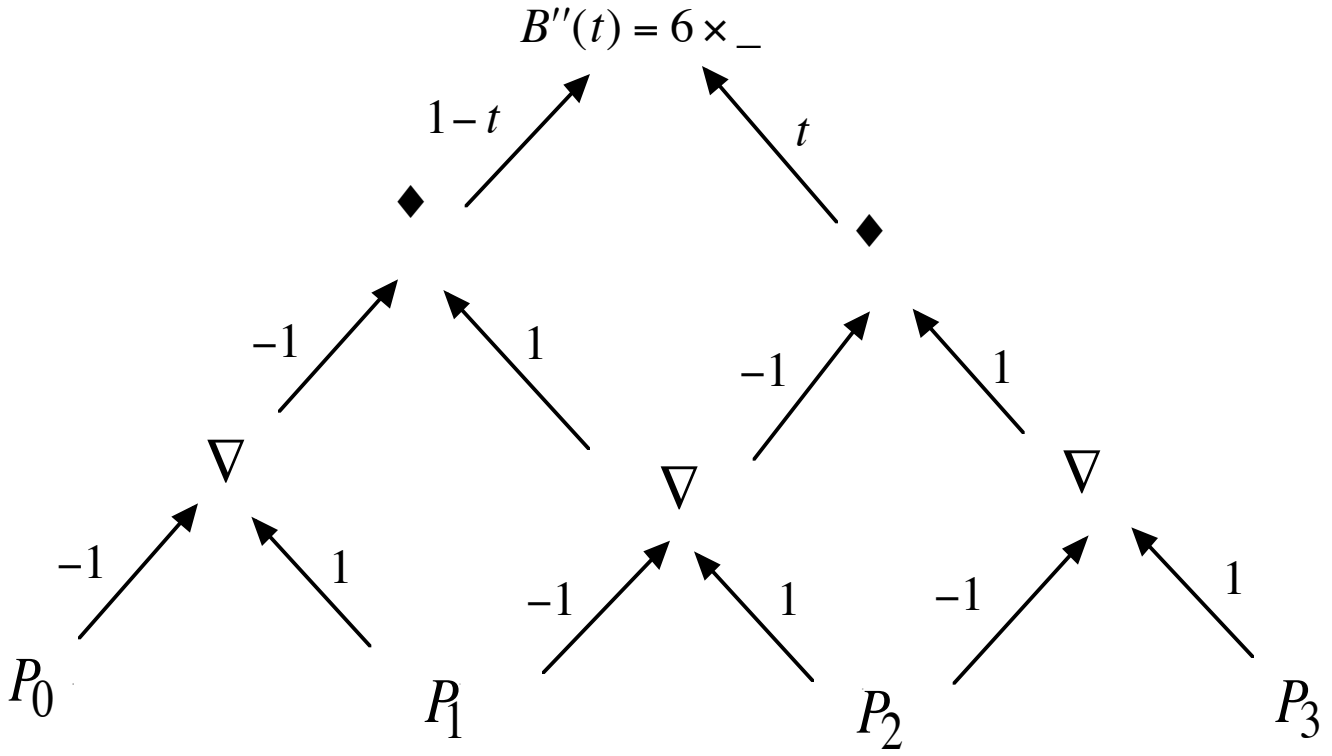
# Differentiating de Casteljau's Algorithm -- First Derivative




$B(t)$ = Bezier Curve $\quad 0 \le t \le 1$

$P_0, \ldots, P_n$ = Control Points

# Differentiating de Casteljau's Algorithm -- Second Derivative




$B(t)$ = Bezier Curve $\quad 0 \le t \le 1$

$P_0, \ldots, P_n$ = Control Points

# Differentiating Bezier Curves

*Observations*

- To differentiate the de Casteljau algorithm $k$ times, differentiate any $k$ levels and multiply the result by $\frac{n!}{(n-k)!}$ .

- The $k^{th}$ derivative at either end point $(t = 0,1)$ depends only on the adjacent $k$ control points.

## Continuity Conditions for Adjacent Bezier Curves

*Derivatives at End Points*

- $B(0) = P_0$ $\qquad$ $B(1) = P_n$
- $B'(0) = n(P_1 - P_0)$ $\qquad$ $B'(1) = n(P_n - P_{n-1})$
- $B''(0) = n(n-1)(P_2 - 2P_1 + P_0)$ $\qquad$ $B''(1) = n(n-1)(P_n - 2P_{n-1} + P_{n-2})$

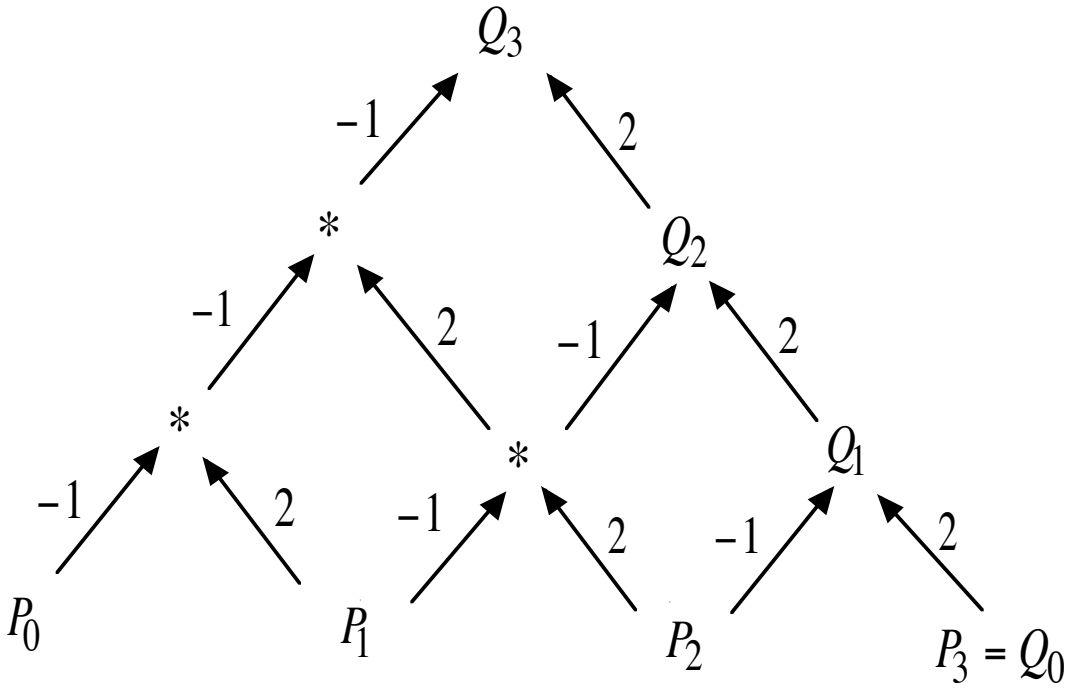
*Continuity Conditions -- Matching k Derivatives*

$k = 0$: $\quad Q_0 = P_n$

$k = 1$: $\quad Q_1 - Q_0 = P_n - P_{n-1} \Rightarrow Q_1 = P_n + (P_n - P_{n-1})$

$k = 2$: $\quad Q_2 - 2Q_1 + Q_0 = P_n - 2P_{n-1} + P_{n-2} \Rightarrow Q_2 = P_{n-2} + 4(P_n - P_{n-1})$

# Subdivision and Differentiation




*Subdivision at* $r = 2$

## Subdivision and Differentiation

*Observations*

1. The two curves
   - $B(t) = B[P_0, \ldots, P_n](t) \quad 0 \le t \le 1$
   - $B(t) = B[P_0, \ldots, P_n](t) \quad 1 \le t \le 2$

   meet smoothly, since they are the same polynomial.

2. The $k^{th}$ derivative at either end point $(t = 0,1)$ depends only on the adjacent $k$ control points.

3. Therefore subdividing a Bezier curve at $r = 2$ generates the locations of the control points $\{Q_k\}$ for arbitrary smooth piecewise Bezier curves.

# Subdivision

*Key Ideas*

- de Casteljau's Evaluation Algorithm is also a Subdivision Procedure
- Subdivision = Divide and Conquer
- Subdivision is the Basic Tool for Analyzing Bezier Curves and Surfaces
    - Rendering
    - Intersection
    - Joining Curves Smoothly
    - Variation Diminishing Property