

B-Spline Curves with Knots in the Complex Plane

Ron Goldman

Konstantinos Tsianos

Department of Computer Science

Rice University

*The shortest path between two truths in the real domain
sometimes passes through the complex domain.*

Jacques Hadamard

Complex B-Splines

References

1. B. Foster, T. Blu, and M. Unser (2006), *Complex B-splines*, Appl. Comp. Harmon. Anal., Vol. 20, pp. 281-282.
2. M. Unser and T. Blu (2000), *Fractional splines and wavelets*, SIAM Review, Vol. 42, No. 1, pp. 43-67.
3. B. Foster and P. Massopust (2007), *Some remarks about the connection between fractional divided differences, fractional B-splines, and the Hermite-Genocchi Formula*, Int. Jour. of Wavelets, Multiresolution and Information Processing.
4. B. Foster and P. Massopust (2007), *Statistical encounters with complex B-splines*, Constructive Approximation.
5. G. Walz (1991), *Spline-Funktionen im Komplexen*, BI-Wiss.-Verl.
6. Many others: H. Chen, G. Opfer, M. Puri, . . .

Bezier and B-Spline Curves in the Plane

Formulas for B-Curves in the Plane

- $B: I \subset \mathbf{R} \rightarrow \mathbf{R}^2 \quad B(t) = (x(t), y(t))$
- $B(t) = \sum_{k=0}^n \binom{n}{k} t^k (1-t)^{n-k} P_k \quad 0 \leq t \leq 1$
- $B(t) = \sum_k N_k^n(t | t_k, \dots, t_{k+n}) P_k \quad t, t_k, \dots, t_{k+n} \in I$
-- $P_k = (x_k, y_k) = x_k + y_k i$

Algorithms for B-Curves in the Plane

- Subdivision Algorithms for Bezier Curves
- Knot Insertion Algorithms for B-Spline Curves

Complex Bezier and B-Splines Functions

Fundamental Observation

- Every Polynomial Identity and Algorithm for Real Numbers is Also Valid for Complex Numbers

$$\text{-- } B : S \subset \mathbb{C} \rightarrow \mathbb{C}$$

$$\text{-- } B(z) = \sum_{k=0}^n \binom{n}{k} z^k (1-z)^{n-k} w_k \quad z \in S, w_0, \dots, w_n \in \mathbb{C}$$

$$\text{-- } B(z) = \sum_k N_k^n(z | z_k, \dots, z_{k+n}) w_k \quad z, z_k, \dots, z_{k+n} \in S$$

$$\text{-- } w_k = x_k + y_k i$$

Algorithms for B-Curves in the Complex Plane

- Subdivision Algorithms for Complex Bezier Functions
- Knot Insertion for Algorithms Complex B-Spline Functions

Questions

Bezier Curves -- Subdivision at Complex Parameter Values

- What happens if we subdivide Bezier curves at complex parameter values?
 - Convergence, Limit Curves . . . ?
- What is the relation of the limit curve to complex Bernstein polynomials?

$$\text{-- } B(z) = \sum_{k=0}^n \binom{n}{k} z^k (1-z)^{n-k} w_k \quad z \in S, w_0, \dots, w_n \in C$$

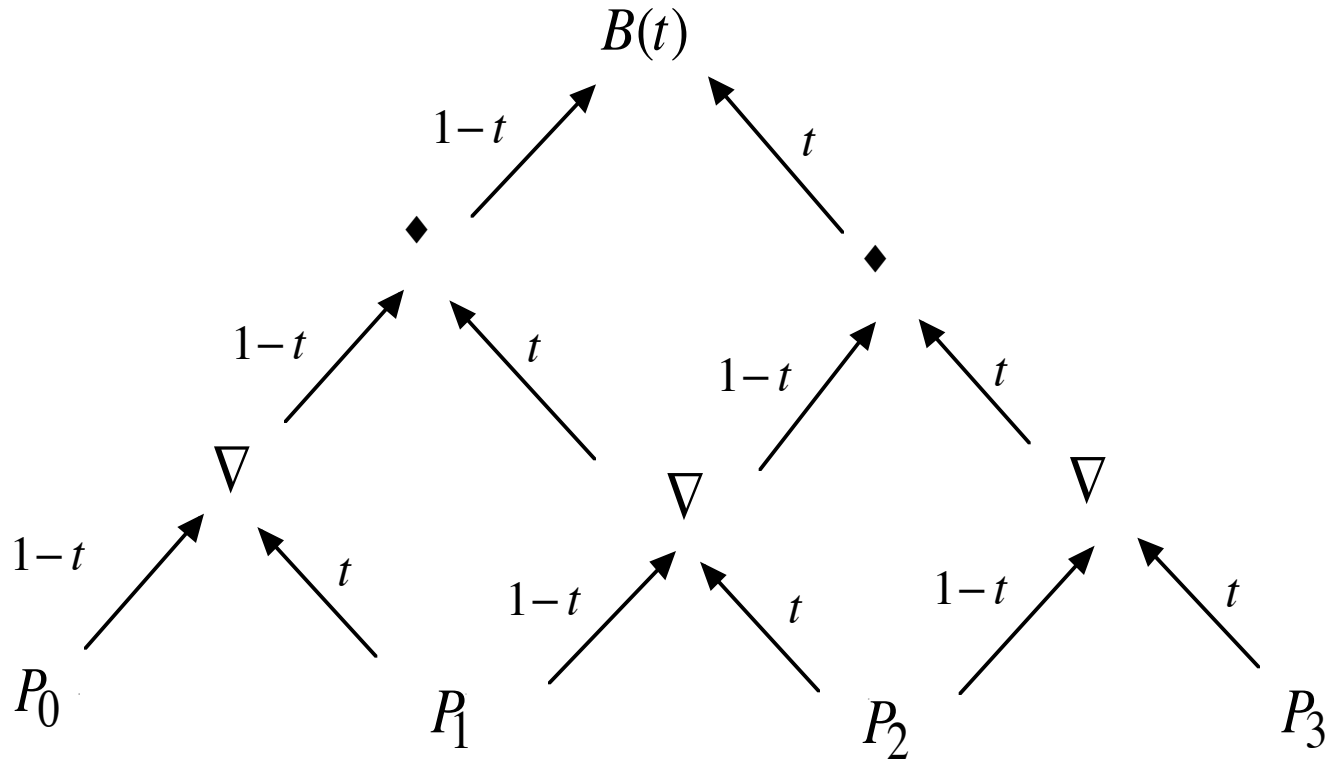
B-Spline Curves -- Knot Insertion at Complex Knots

- What happens if we insert complex knots?
 - Convergence, Limit Curves . . . ?
- What is the relation of the limit curve to complex B-splines?

$$\text{-- } B(z) = \sum_k N_k^n(z | z_k, \dots, z_{k+n}) w_k \quad z, z_k, \dots, z_{k+n} \in S$$

Part I:
Bezier Curves in the Complex Plane

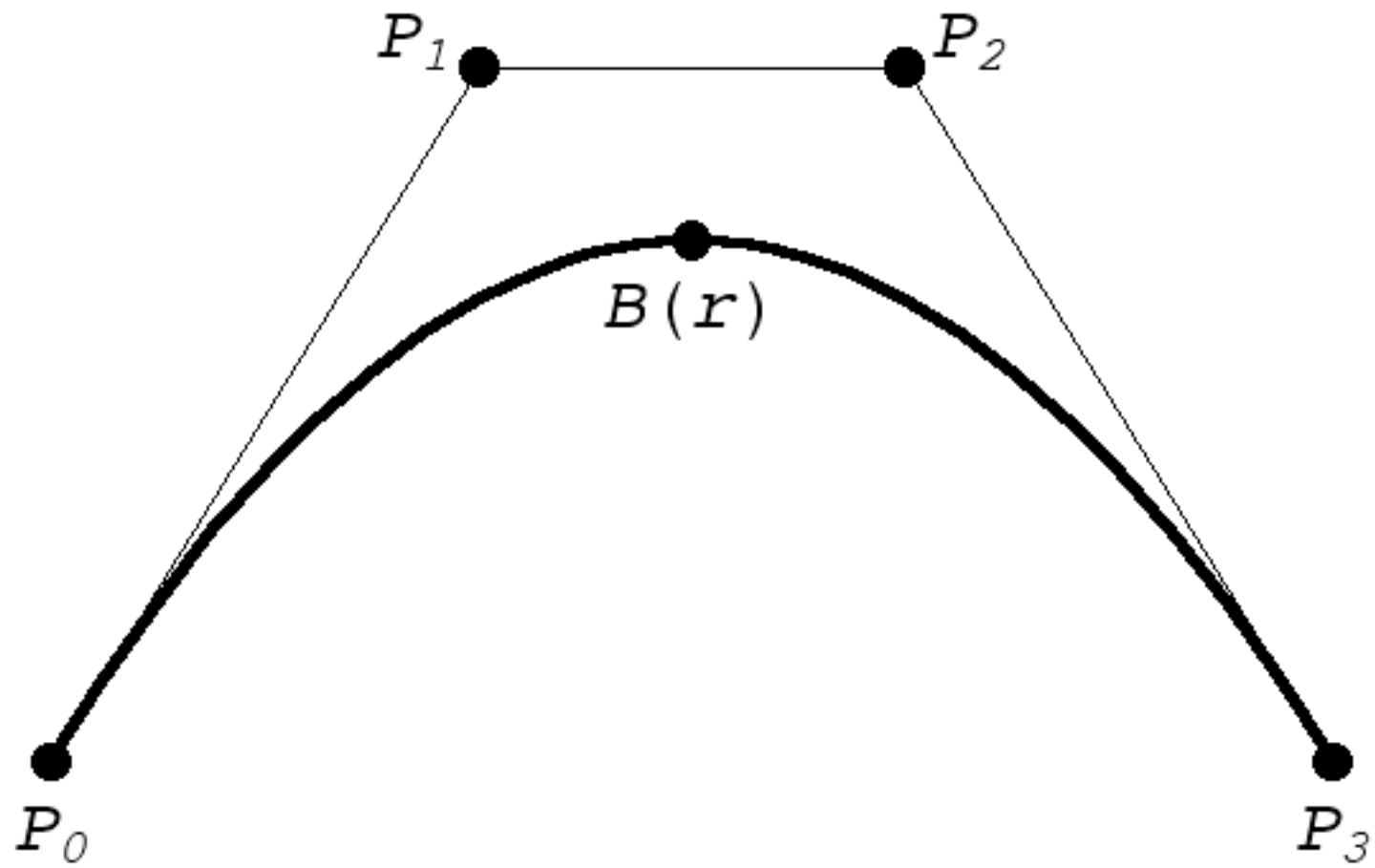
De Casteljau's Evaluation Algorithm



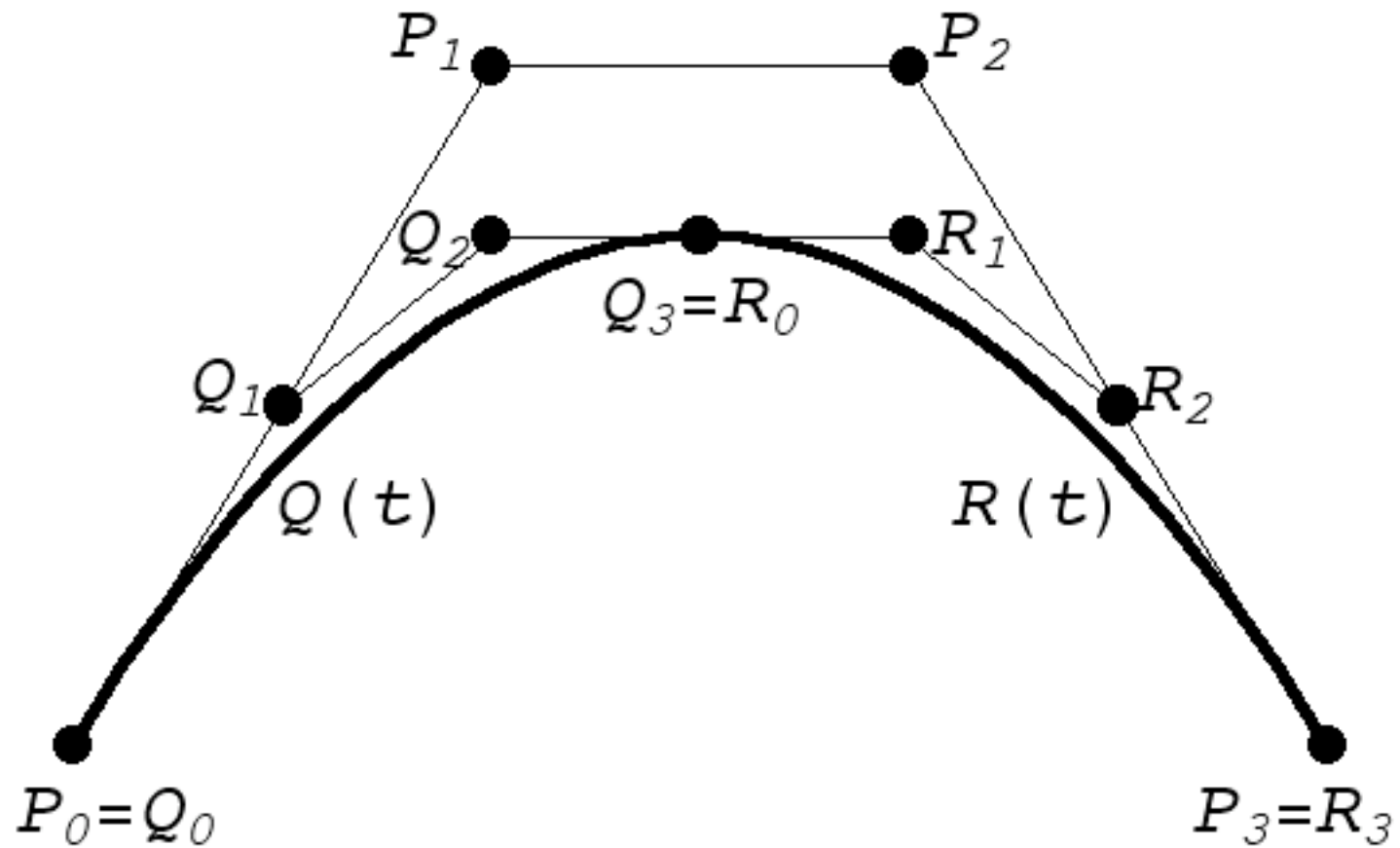
$$B(t) = \sum_{k=0}^n \binom{n}{k} t^k (1-t)^{n-k} P_k = \text{Bezier Curve}$$

$P_0, \dots, P_n = \text{Control Points}$

Bezier Curve

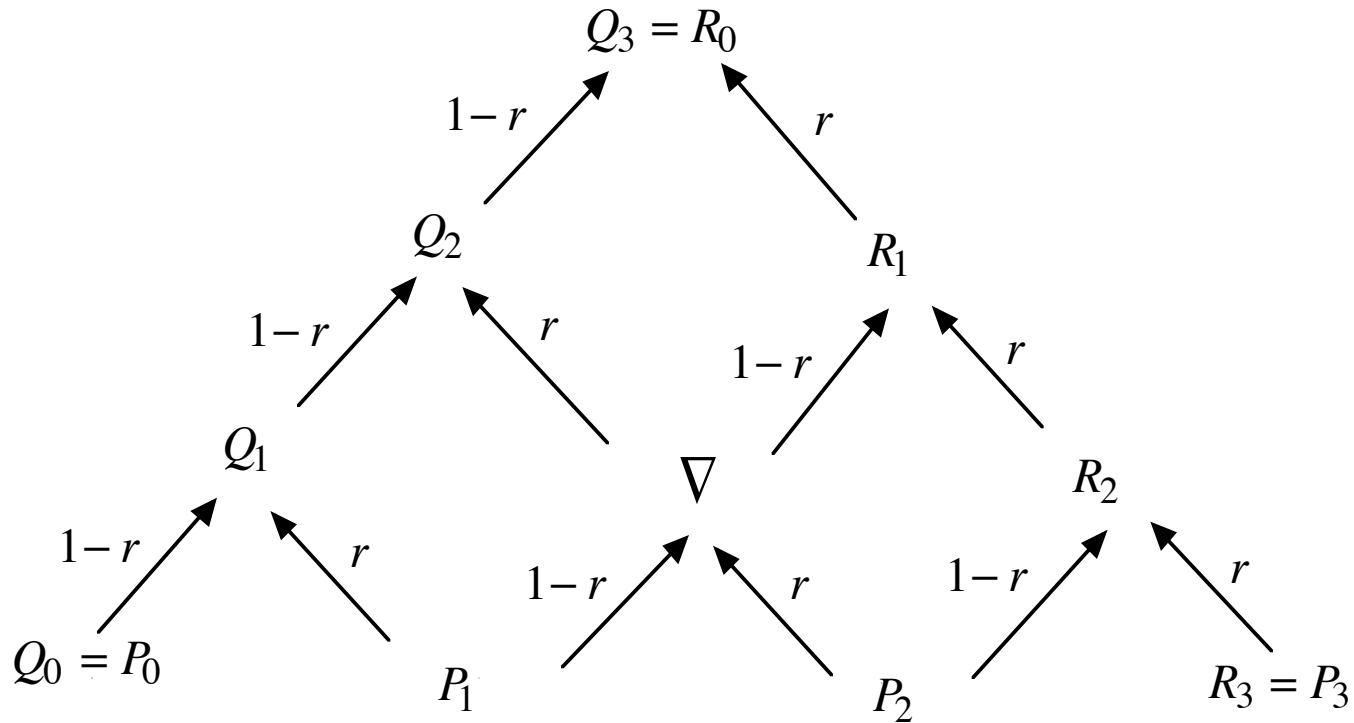


Bezier Subdivision



Problem: Given P_0, \dots, P_n , find Q_0, \dots, Q_n and R_0, \dots, R_n .

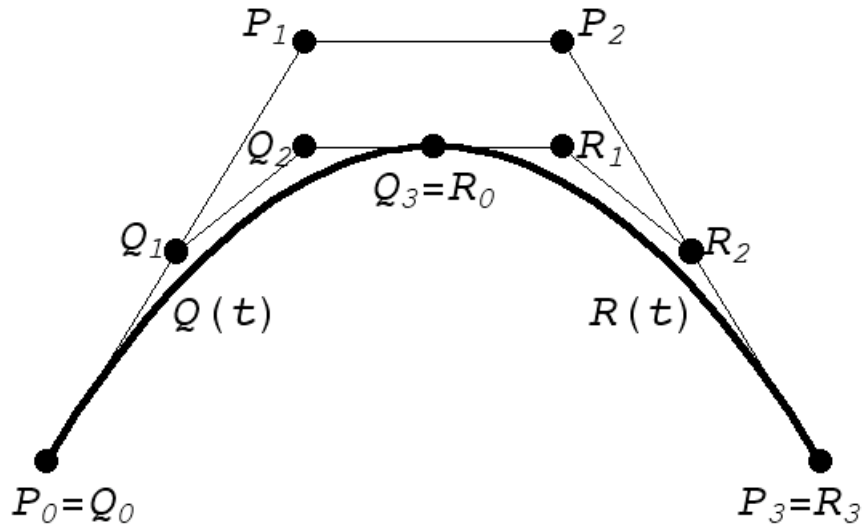
De Casteljau's Subdivision Algorithm



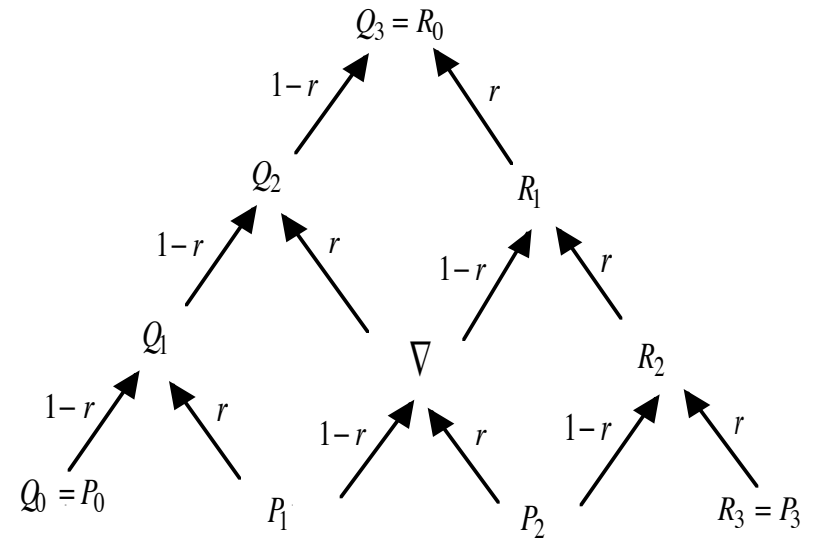
$B(t) = \text{Bezier Curve } 0 \leq t \leq 1$

$P_0, \dots, P_n = \text{Original Control Points}$

De Casteljau Subdivision



Geometry



Algorithm

Algorithms for Bezier Curves

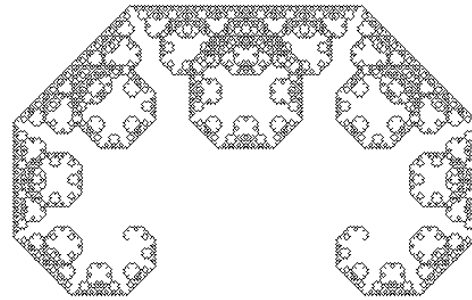
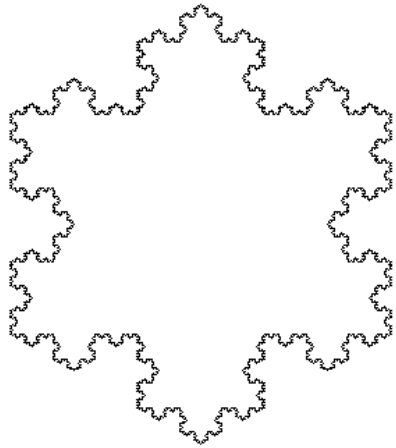
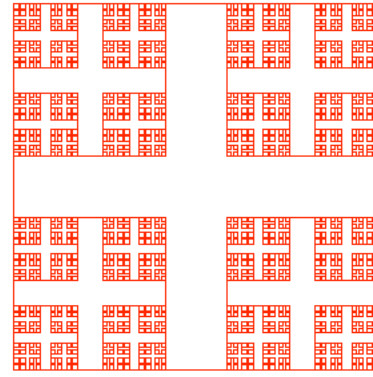
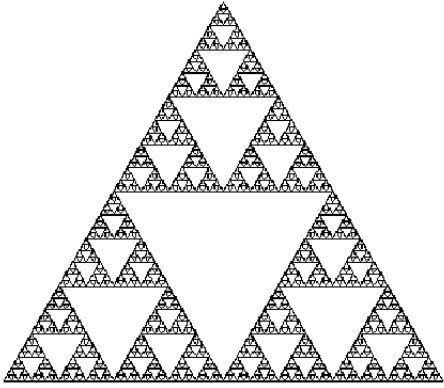
Rendering Algorithm

- 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.

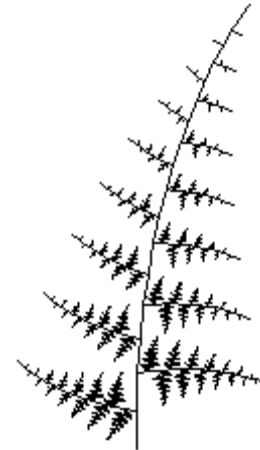
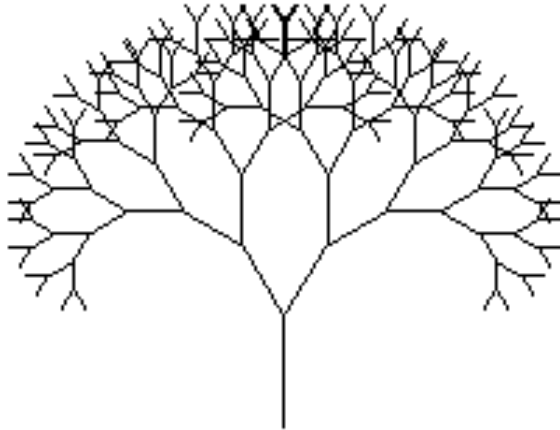
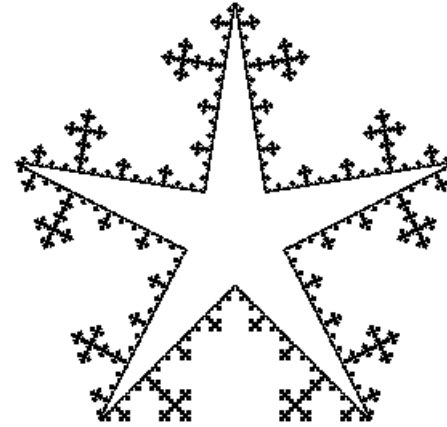
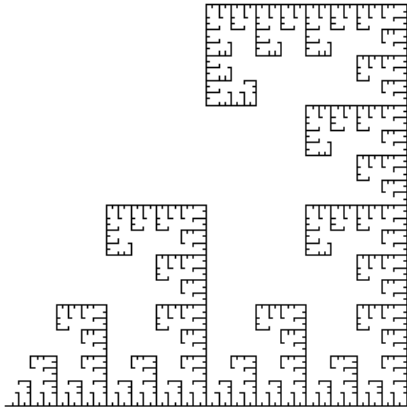
Intersection Algorithm

- 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.

Fractals



More Fractals



Fractals from Iterated Function Systems

Maps on Sets

- $w(S) = \{w(x) \mid x \in S\}$

Iterated Function System (IFS)

- $W = \{w_1, \dots, w_l\}$ -- Collection of Contractive Maps
- $W(S) = w_1(S) \cup \dots \cup w_l(S)$

Fractal Algorithm

- $A_0 = B$ (pick any set B)
- $A_{n+1} = W(A_n) = w_1(A_n) \cup \dots \cup w_l(A_n)$
- A_n converges to the fractal (fixed point) A

Fractals

Properties

- Self Similar
- Fractional Dimension
- Continuous Everywhere, Differentiable Nowhere
- Fixed Points of Iterated Function Systems
- Attractors -- Independent of Base Case

Bezier Curves

Properties

- Self Similar ?
- 1-Dimensional
- Polynomials, Differentiable Everywhere
- Fixed Points of Subdivision Algorithm
- Control Points
- Parametrizations

Comparisons

Bezier Curves

Self Similar ?

Dimension = 1

Polynomials,
Differentiable Everywhere

Fixed Point of Subdivision Algorithm

Control Points,
Depend on Base Case

Parametrizations

Fractals

Self Similar

Dimension > 1

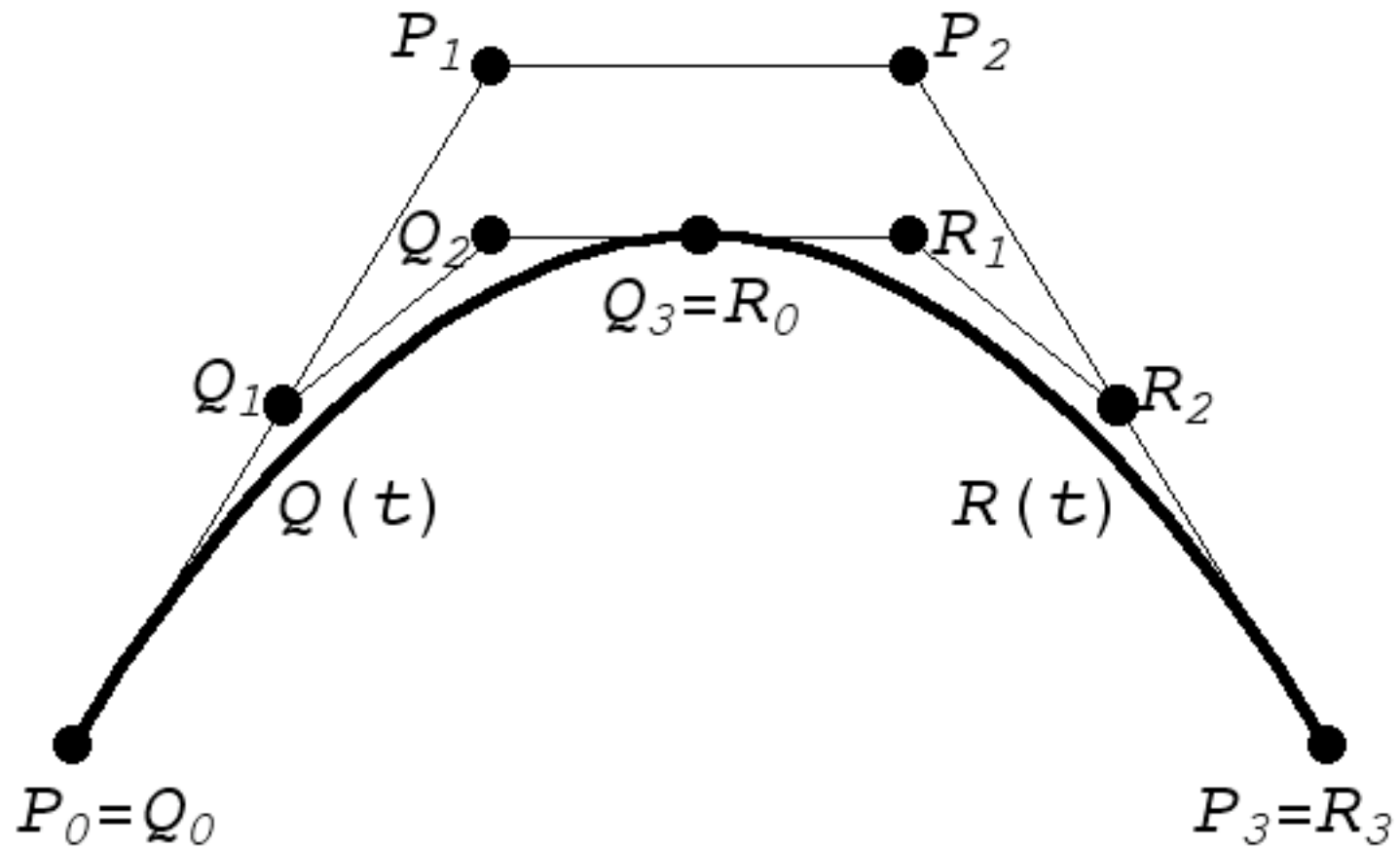
Continuous Everywhere,
Differentiable Nowhere

Fixed Point of Fractal Algorithm

Attractors,
Independent of Base Case

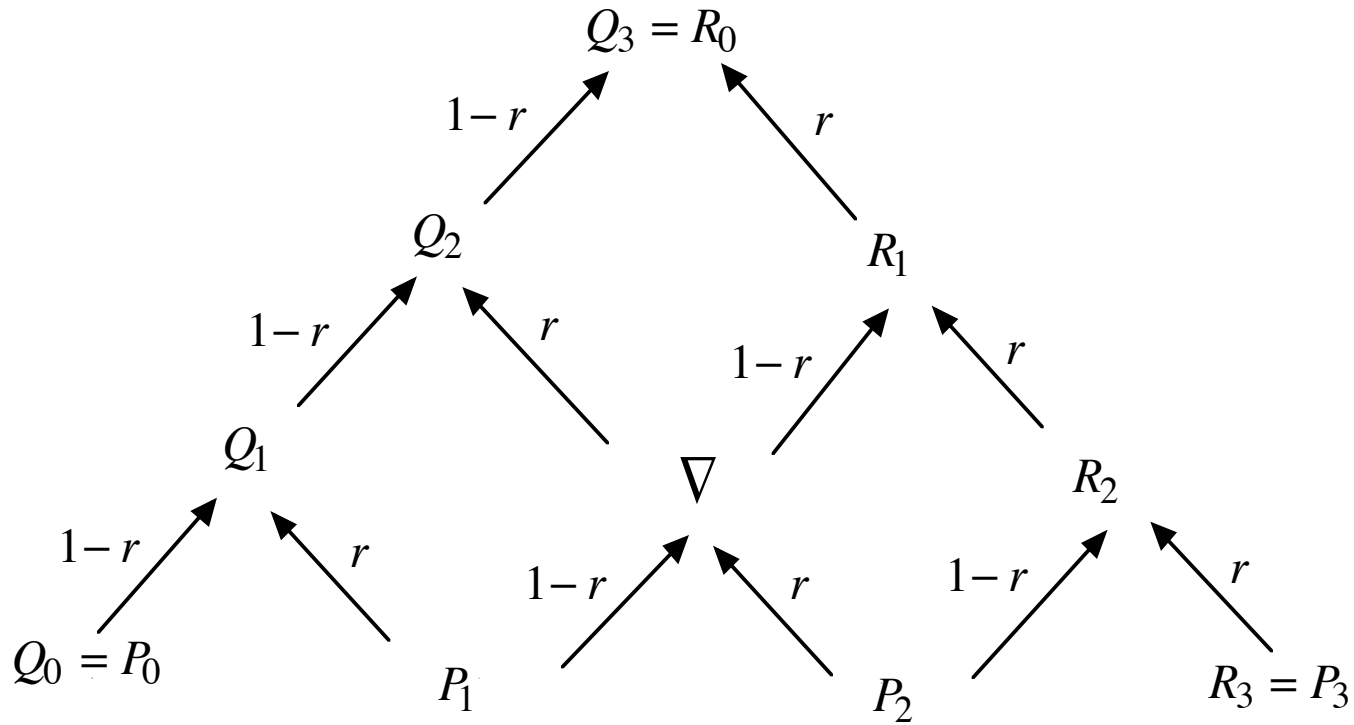
Parametrizations ?

Bezier Subdivision



Problem: Given P_0, \dots, P_n , find Q_0, \dots, Q_n and R_0, \dots, R_n .

De Casteljau's Subdivision Algorithm



$B(t) = \text{Bezier Curve } 0 \leq t \leq 1$

$P_0, \dots, P_n = \text{Original Control Points}$

Subdivision Matrices

Matrices

$$\bullet L(r) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1-r & r & 0 & 0 \\ (1-r)^2 & 2r(1-r) & r^2 & 0 \\ (1-r)^3 & 3r(1-r)^2 & 3r^2(1-r) & r^3 \end{pmatrix}$$

$$\bullet R(r) = \begin{pmatrix} (1-r)^3 & 3r(1-r)^2 & 3r^2(1-r) & r^3 \\ 0 & (1-r)^2 & 2r(1-r) & r^2 \\ 0 & 0 & 1-r & r \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Subdivision

$$\bullet \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix} = L(r) * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} \quad \bullet \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix} = R(r) * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Fractal Nature of Bezier and B-Spline Curves and Surfaces

References

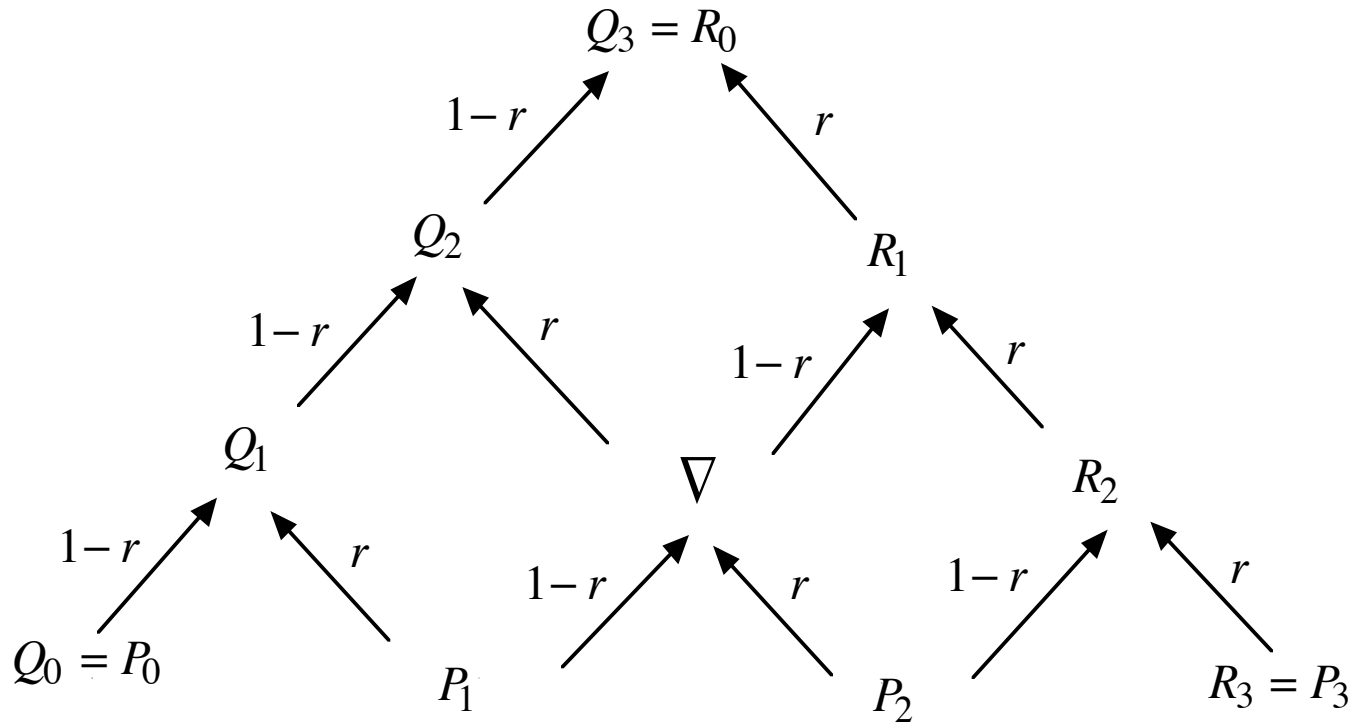
- R. Goldman (2004), The Fractal Nature of Bezier Curves, *Proceedings of Geometric Modeling and Processing (GMP) 2004: Theory and Applications*, pp. 3-11.
- S. Schaefer, R. Goldman, and D. Levin (2005), Subdivision Schemes and Attractors, *Proceedings of Symposium on Geometric Processing (SGP) 2005*, edited by M. Desbrun and H. Pottmann, Vienna, Austria, pp. 171-180.

Today's Theme

Spline Nature of Fractal Curves

- Fractals with Control Points
- Fractals with Parametrizations
- Fractals as Complex Bezier and B-Spline Curves

De Casteljau's Subdivision Algorithm



$B(t) = \text{Bezier Curve } 0 \leq t \leq 1$

$P_0, \dots, P_n = \text{Original Control Points}$

Subdivision Computations

Real = Scaling

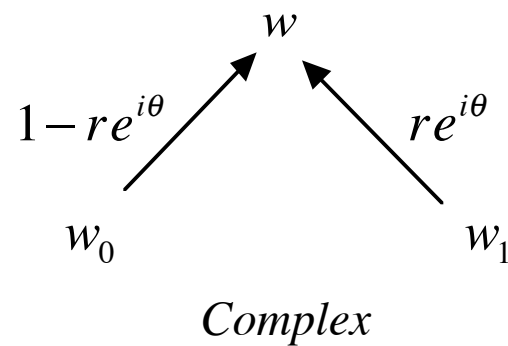
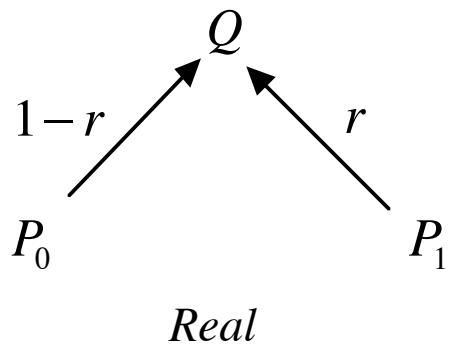
- $Q = (1-r)P_0 + rP_1 = P_0 + r(P_1 - P_0)$

Complex = Scaling and Rotation

- $w = (1-re^{i\theta})w_0 + re^{i\theta}w_1 = w_0 + re^{i\theta}(w_1 - w_0)$

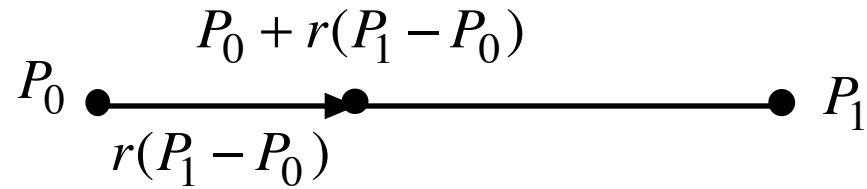
- $w = (1-(x+iy))w_0 + (x+iy)w_1 = w_0 + x(w_1 - w_0) + iy(w_1 - w_0)$

Linear Subdivision



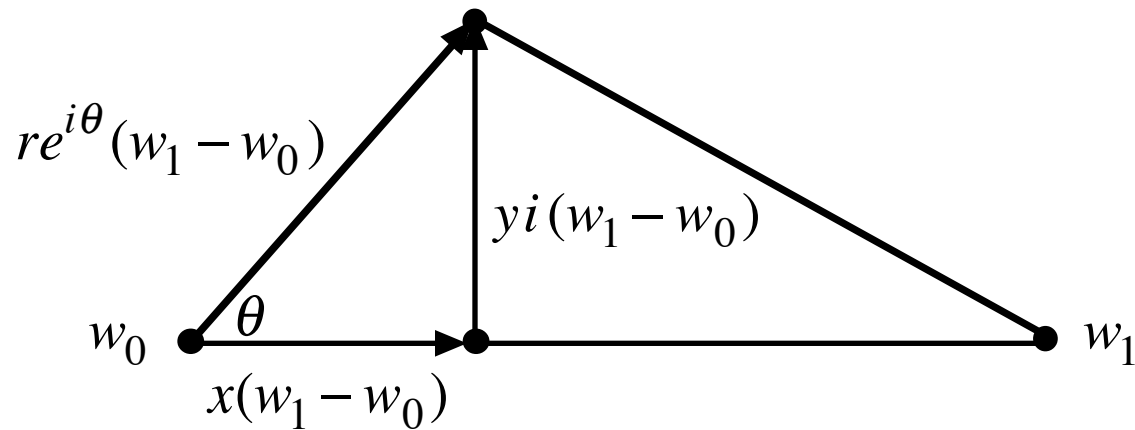
Subdivision Computations

Real = Scaling



Complex = Scaling and Rotation

$$w_0 + re^{i\theta}(w_1 - w_0) = w_0 + x(w_1 - w_0) + yi(w_1 - w_0)$$



Fractals and Bezier Subdivision

Theorem

Every fractal in the plane generated by a Conformal Iterated Function System (translation, rotation, and uniform scaling) can be generated by Bezier subdivision in the complex plane (degree 1 suffices).

Algorithm

There is a simple algorithm to convert a fractal generated by a Conformal Iterated Function System into Bezier curve generated by recursive subdivision.

Observation

The converse is false for $\text{degree} > 1$. Not every curve generated by Bezier subdivision in the complex plane can be generated by a Conformal Iterated Function Systems (need nonlinearity).

Convergence

Theorem 1

The control polygons generated by recursive subdivision converge whenever:

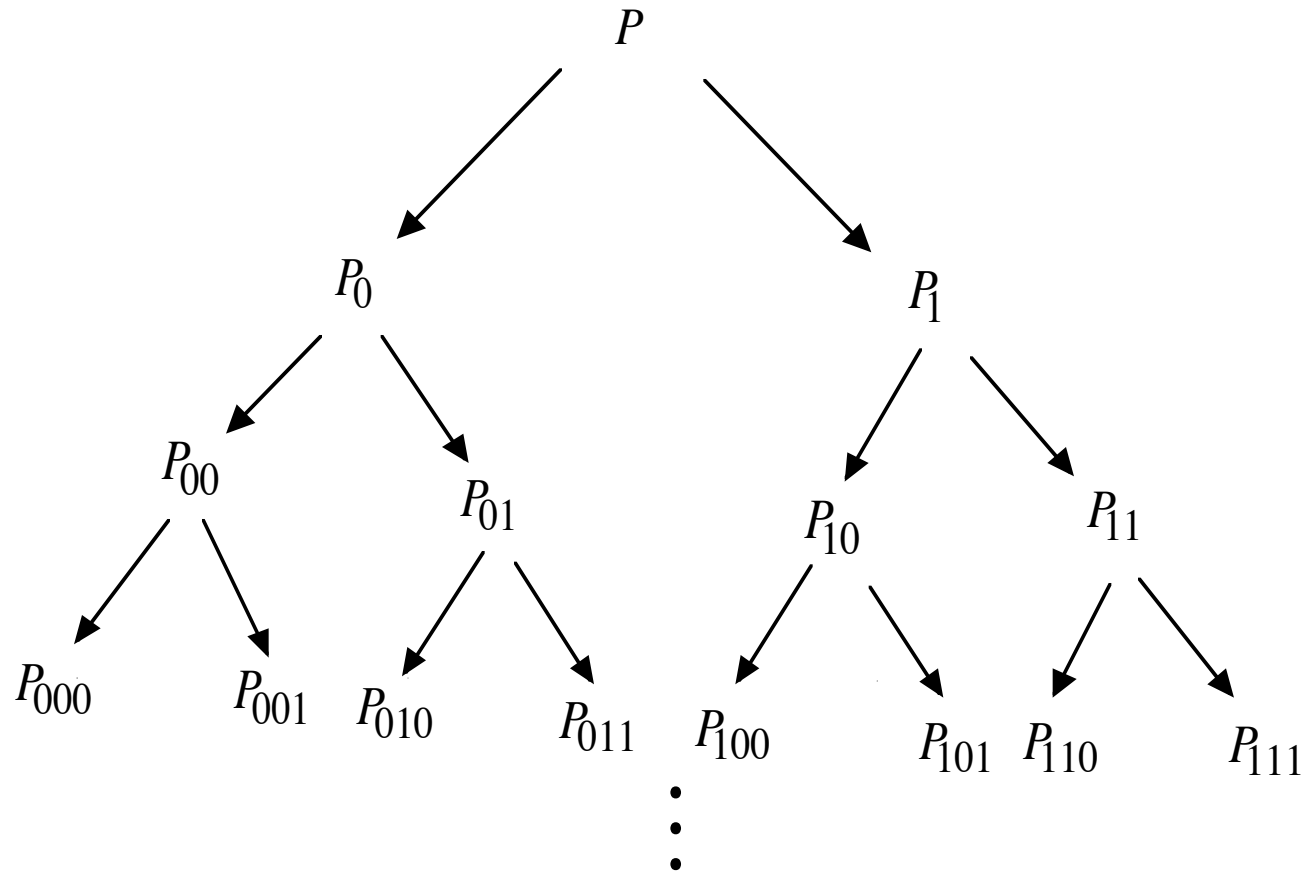
- $0 < r < 1$ *r real*
- $0 < |r| < 1$ *r complex*

Theorem 2

The control polygons generated by recursive subdivision converge to:

- $$P(t) = \sum_{k=0}^n B_k^n(z(t)) w_k$$
- $z(t) = \text{Curve of Degree One Generated by Recursive Subdivision}$
- $w_k = \text{Original Control Points}$

Recursive Subdivision and Parametrization

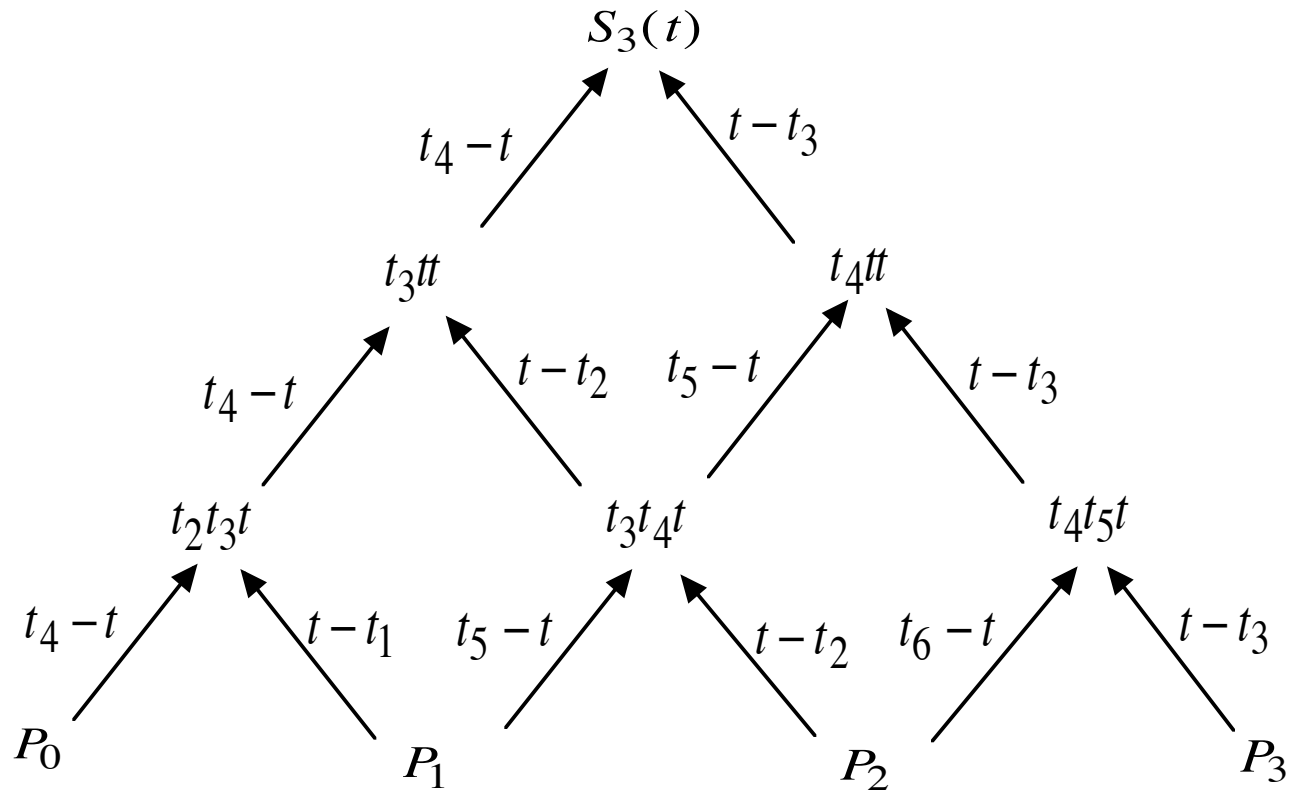


Control Polygons Generated by Recursive Subdivision

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

Part II:
B-Spline Curves with Complex Knots

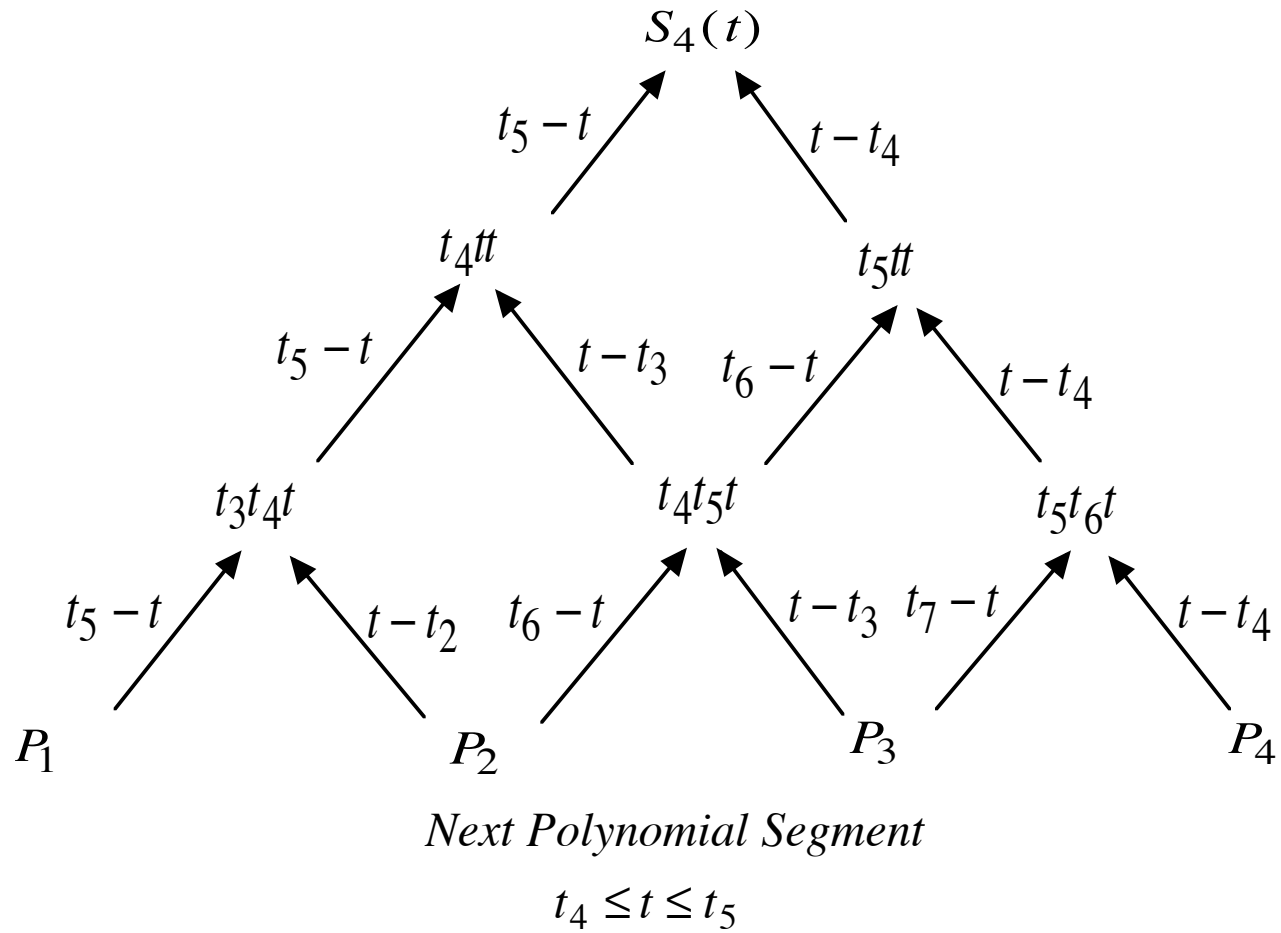
De Boor Algorithm



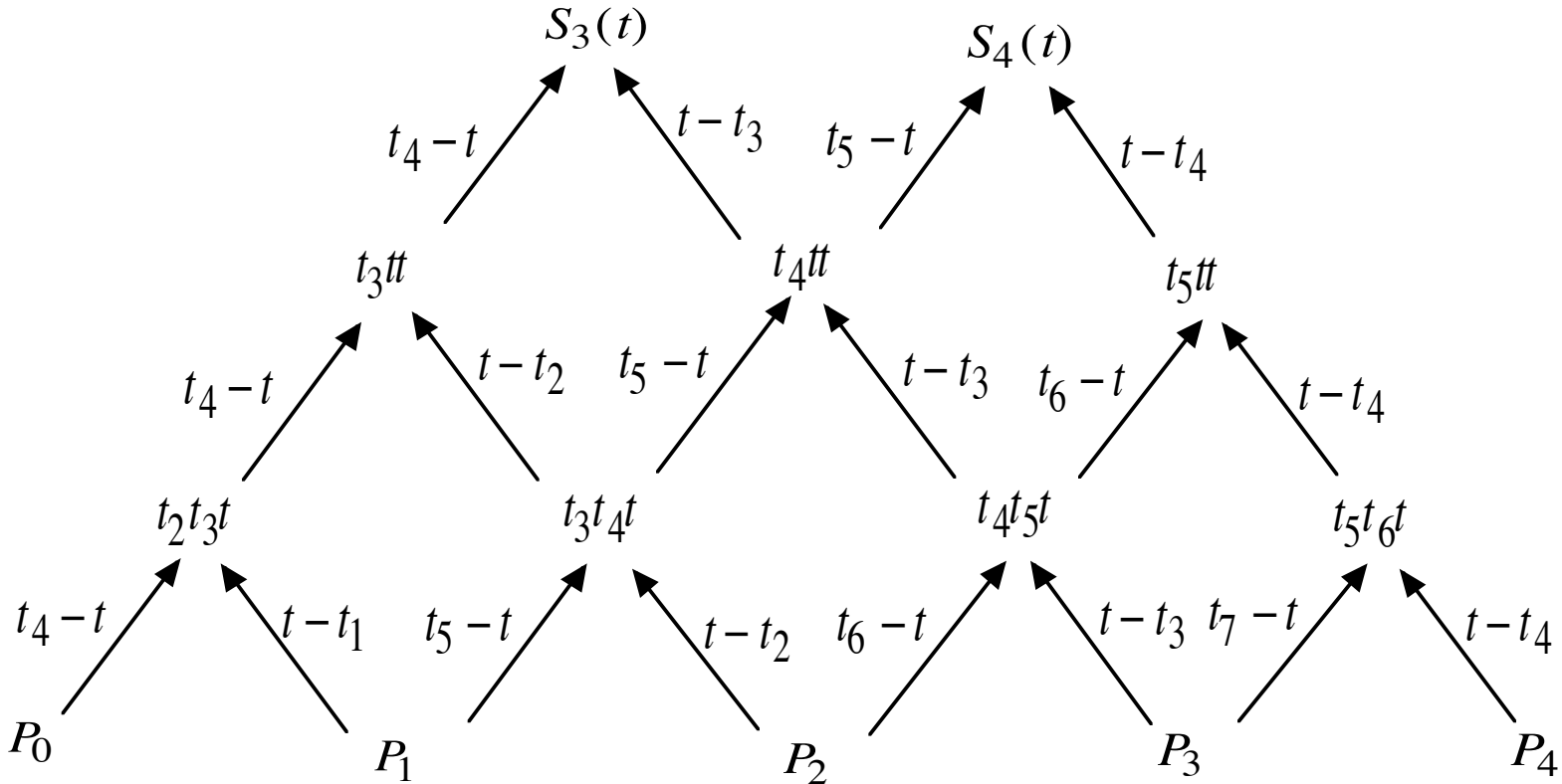
One Polynomial Segment

$$t_3 \leq t \leq t_4$$

De Boor Algorithm



De Boor Algorithm



Two Polynomial Segments

$$t_3 \leq t \leq t_5$$

Knot Insertion Algorithm

Input

- $T = \{t_1, \dots, t_{v+n}\}$ -- knot sequence
- $P = \{P_0, \dots, P_v\}$ -- control points
- $\Gamma = \{\tau_1, \dots, \tau_{\mu+n}\}$ -- new knot sequence -- $\Gamma \supset T$

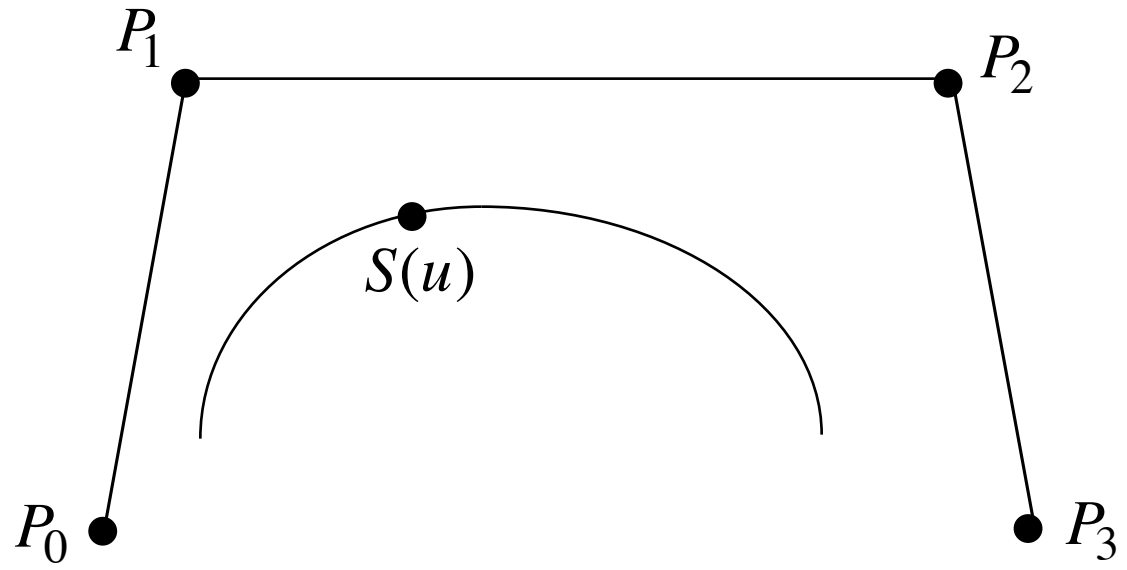
Output

- $Q = \{Q_0, \dots, Q_\mu\}$ -- new control points

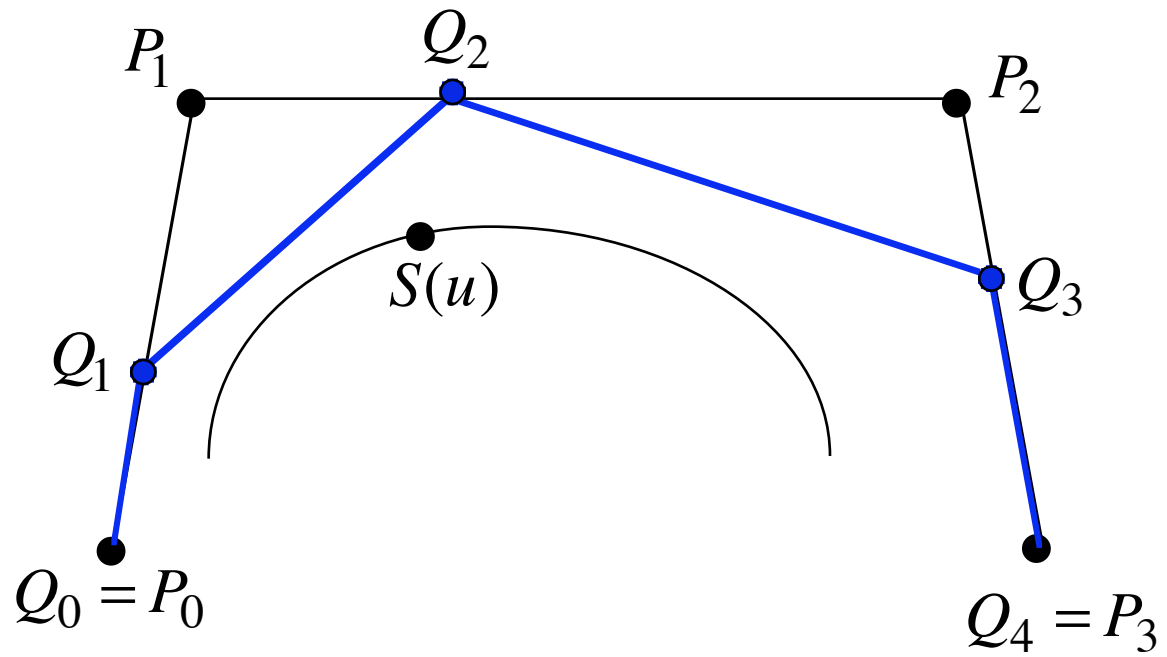
Constraint

- $S(t | T, P) = S(t | \Gamma, Q)$

B-Spline Curve



Knot Insertion



Theorems

Existence

- All splines are B-splines.

Convergence

- The control polygons generated by knot insertion converge to the B-spline curve for the original control polygon as the knot spacing approaches zero.

Corner Cutting

- Knot insertion is a corner cutting procedure.

Applications of Knot Insertion

- Rendering
- Intersection
- Conversion from B-spline to Piecewise Bezier Form
- Proof of the Variation Diminishing Property

Types of Knot Insertion Algorithms

Local Knot Insertion

- $T = \{t_1, \dots, t_{v+n}\}$
- $\Gamma = \{t_1, \dots, t_k, u_{k,1}, \dots, u_{k,d_k}, t_{k+1}, \dots, t_{v+n}\}$

Global Knot Insertion

- $T = \{t_1, \dots, t_{v+n}\}$
- $\Gamma = \{t_1, u_{1,1}, \dots, u_{1,d_1}, t_2, \dots, t_{v+n-1}, u_{v+n-1,1}, \dots, u_{v+n-1,d_{v+n-1}}, t_{v+n}\}$

Examples of Knot Insertion Algorithms

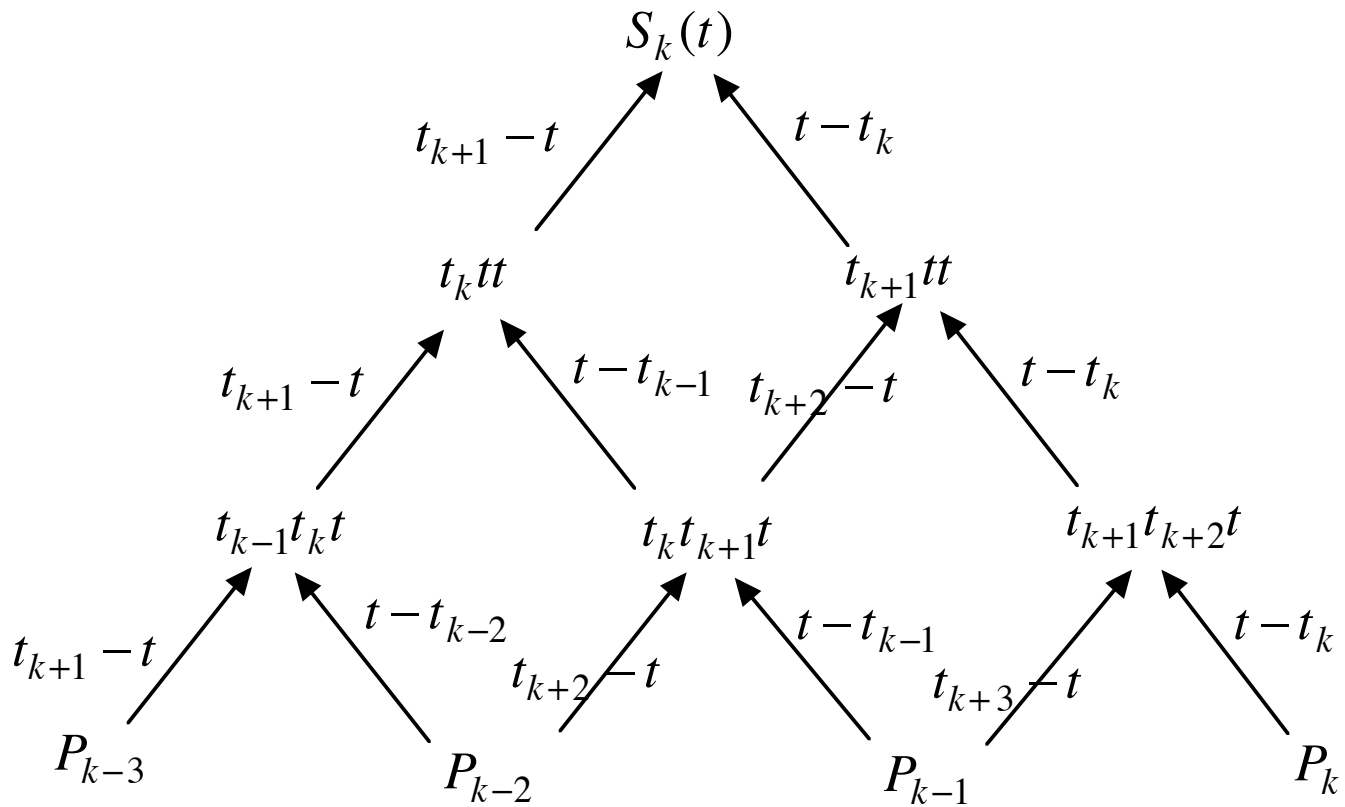
Local Knot Insertion Algorithms

- Boehm's Algorithm
- Oslo Algorithm
- Sablonniere's Algorithm
- Factored Knot Insertion

Global Knot Insertion Algorithms

- Chaikin's Algorithm
- Lane-Riesenfeld Algorithm
- Goldman-Warren Algorithm
- Schaefer's Algorithm -- NEW

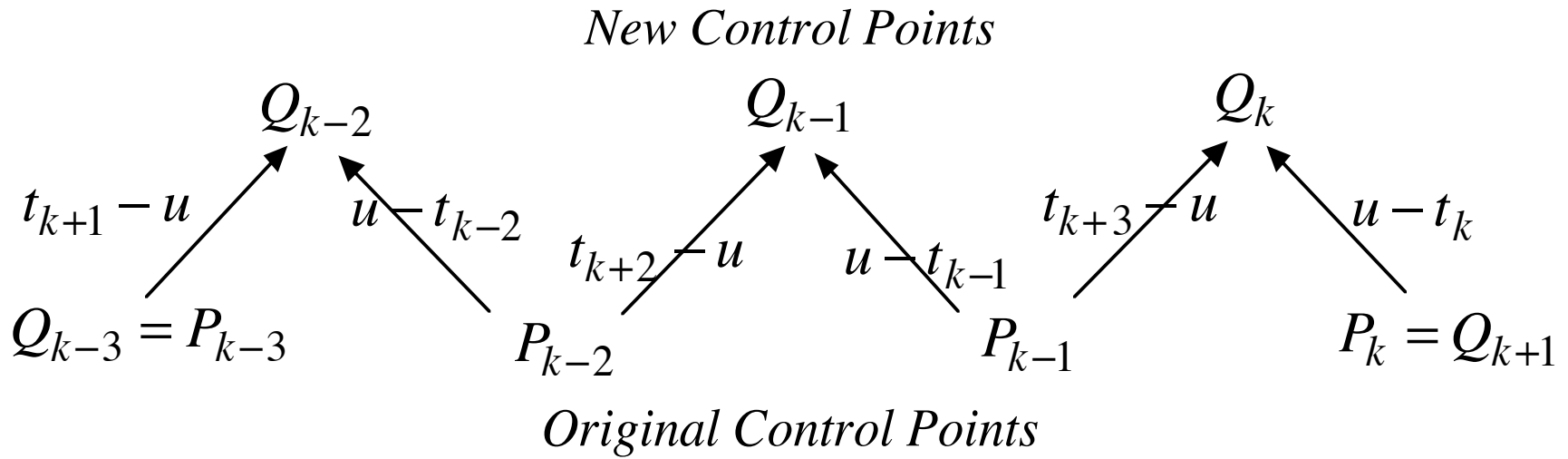
De Boor Algorithm



One Polynomial Segment

$$t_k \leq t \leq t_{k+1}$$

Boehm's Knot Insertion Algorithm -- Simple Knot

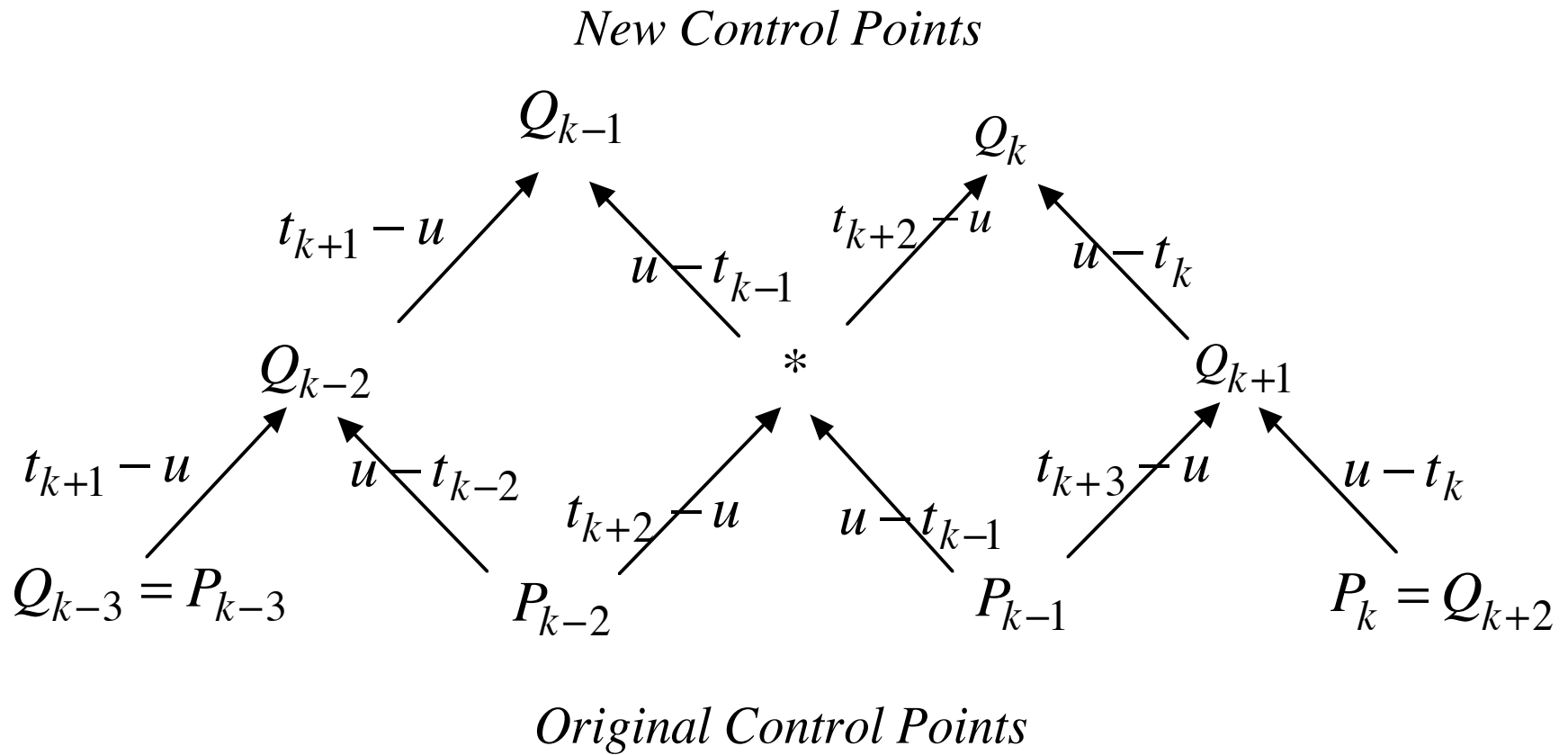


Original Knots = $\dots, t_{k-2}, t_{k-1}, t_k, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

New Knots = $\dots, t_{k-2}, t_{k-1}, t_k, u, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

↑

Boehm's Knot Insertion Algorithm -- Double Knot

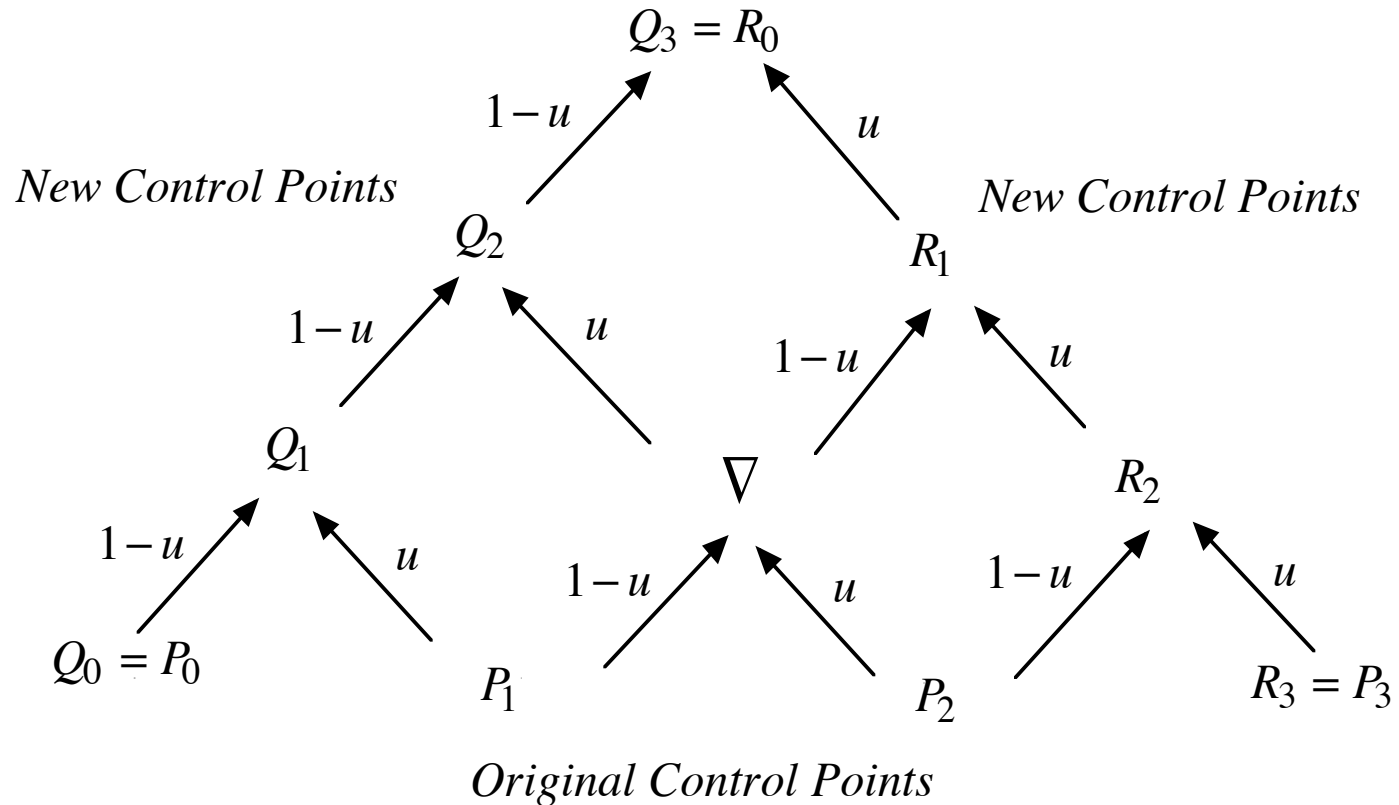


Original Knots = $\dots, t_{k-2}, t_{k-1}, t_k, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

New Knots = $\dots, t_{k-2}, t_{k-1}, t_k, u, u, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

$\uparrow \uparrow$

De Casteljaou's Subdivision Algorithm

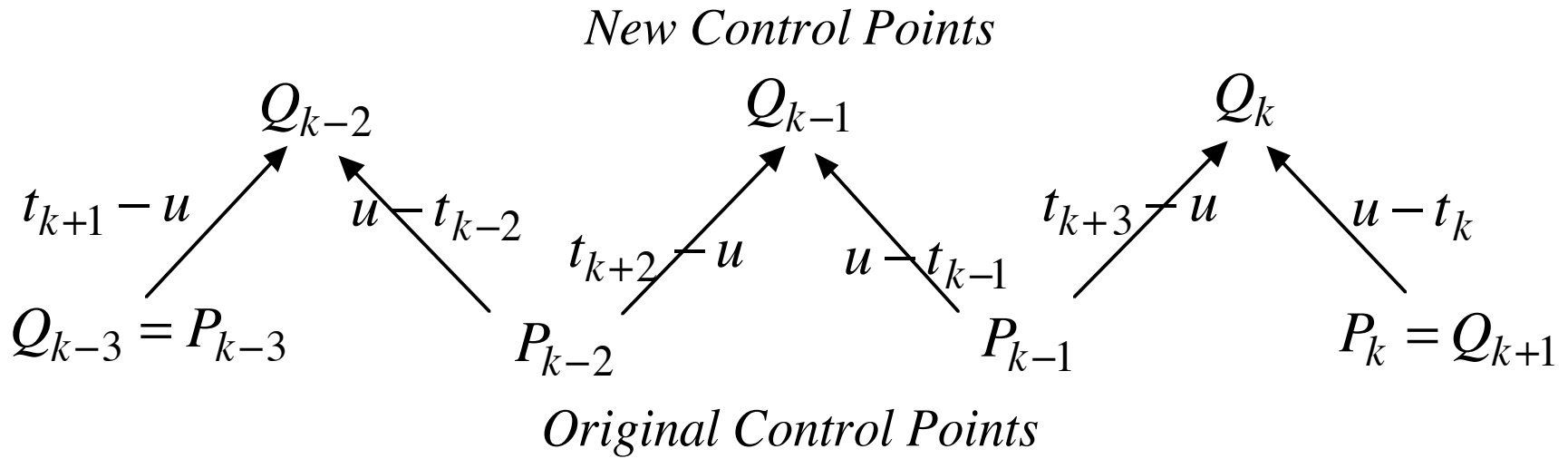


Original Knots = 0,0,0,1,1,1

New Knots = 0,0,0, u , u , u ,1,1,1

↑ ↑ ↑

Boehm's Knot Insertion Algorithm -- Simple Knot

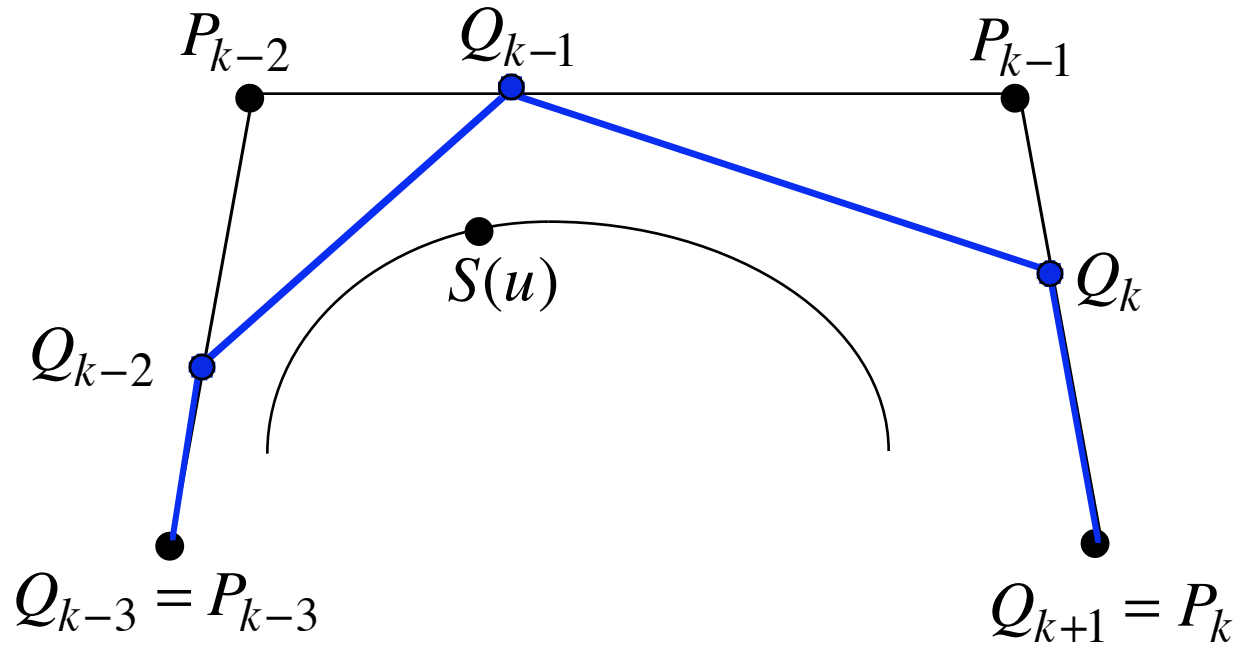


Original Knots = $\dots, t_{k-2}, t_{k-1}, t_k, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

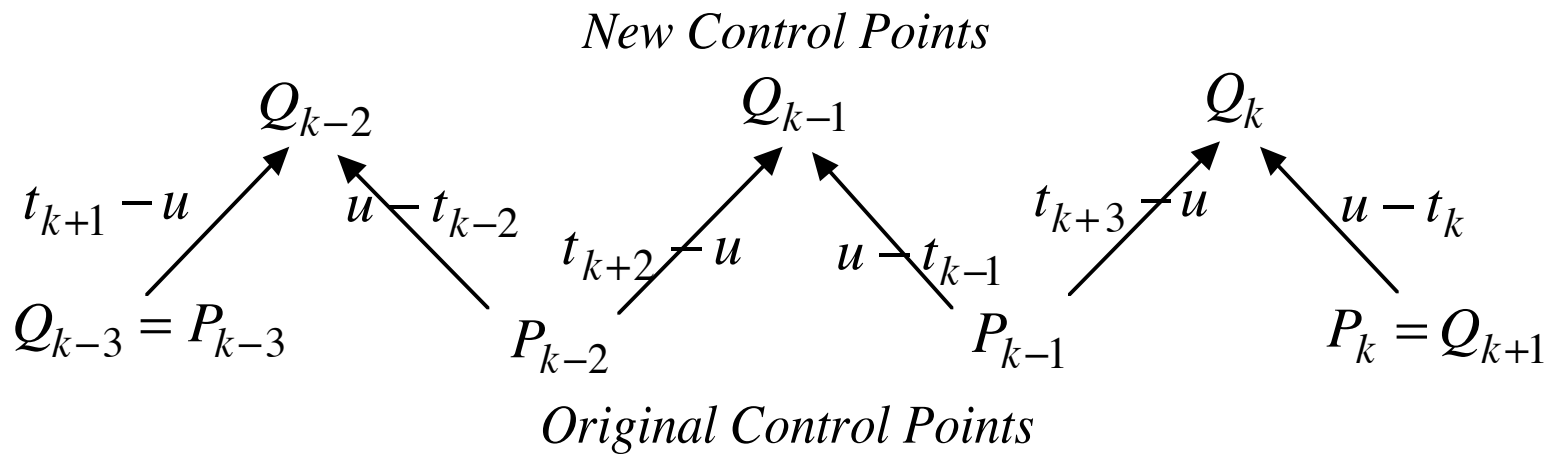
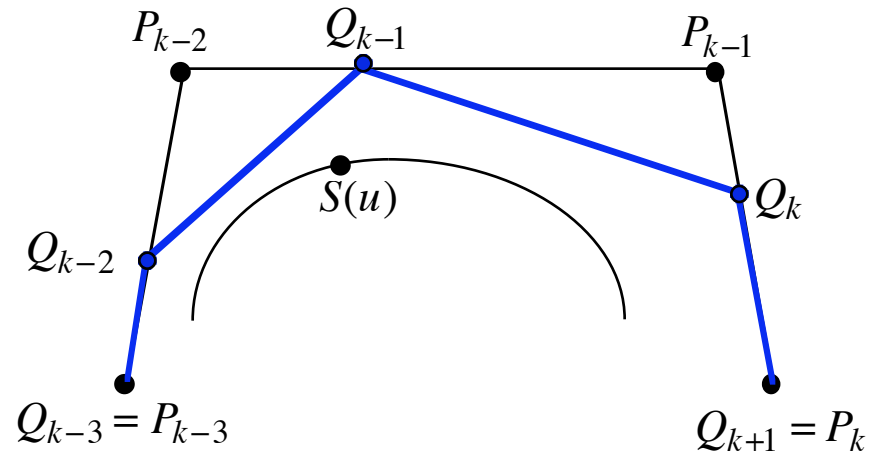
New Knots = $\dots, t_{k-2}, t_{k-1}, t_k, u, t_{k+1}, t_{k+2}, t_{k+3}, \dots$

↑

Boehm's Knot Insertion Algorithm



Boehm Knot Insertion



Fractals and B-Spline Knot Insertion

Theorem

Every fractal in the plane generated by a Conformal Iterated Function System (translation, rotation and uniform scaling) can be generated by B-Spline knot insertion in the complex plane (degree 1 suffices).

Observation

The converse is false for degree > 1. Not every curve generated by B-Spline knot insertion in the complex plane can be generated by a Conformal Iterated Function Systems (need nonlinearity).

Convergence

Theorem 1

- a. *The control polygons generated by knot insertion converge, whenever the distance between the knots approaches zero.*
- b. *The control polygons converge to a smooth curve, whenever the knots converge to a smooth curve.*
- c. *The control polygons need not converge to a closed curve, whenever the knots converge to a closed curve.*

Theorem 2

The control polygons generated by knot insertion converge to:

- $$P(t) = \sum_k N_k^n(z(t) | z_k, \dots, z_{k+n}) w_k$$
- $z(t) = \text{Knot Curve}$
- $w_k = \text{Original Control Points}$

Summary

Fundamental Observation

- Every Polynomial Algorithm and Identity for Real Numbers is Also Valid for Complex Numbers

Algorithms for Bezier and B-Spline Curves in the Complex Plane

- Subdivision Algorithms for Complex Bezier Functions
- Knot Insertion for Algorithms Complex B-Spline Functions
- Convergence and Smoothness

Conclusions

- Splines are Fractals -- Attractors
- Fractals are Splines -- Control Points, Parametrizations

Future Research

Applications Beyond Fractals?

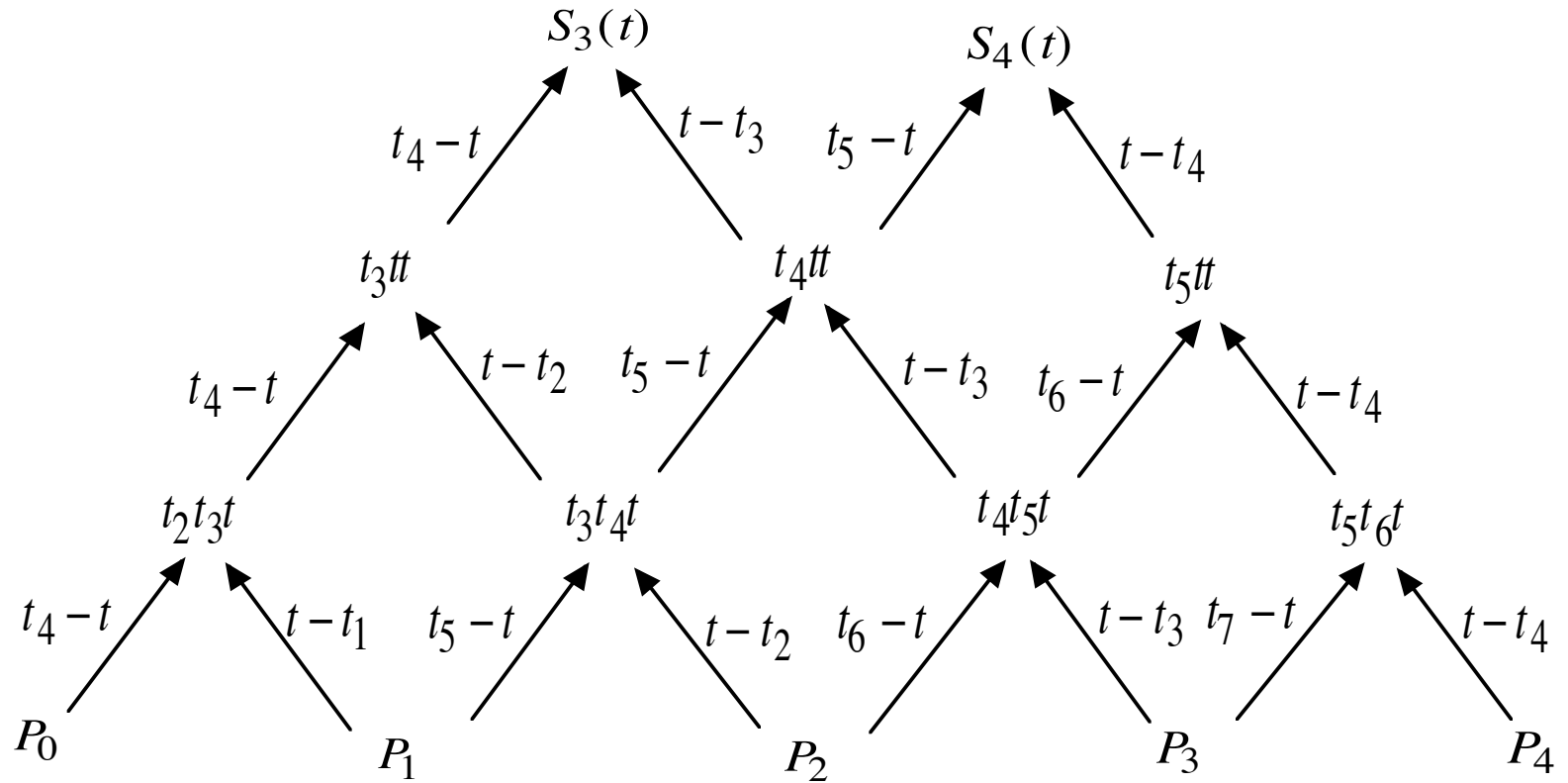
- Geometric Modeling
- Trigonometric Blending Functions -- Affine Invariant

$$\text{-- } B_k^n(e^{i\theta}) = \binom{n}{k} (\cos(\theta) + i\sin(\theta))^k (1 - \cos(\theta) - i\sin(\theta))^{n-k}$$

$$\text{-- } \sum_{k=0}^n B_k^n(e^{i\theta}) \equiv 1 \quad \Rightarrow \quad \sum_{k=0}^n \operatorname{Re}(B_k^n(e^{i\theta})) \equiv 1 \quad \text{and} \quad \sum_{k=0}^n \operatorname{Im}(B_k^n(e^{i\theta})) \equiv 0$$

- Quaternion Curves
- Subdivision Surfaces
- Julia Sets, Mandelbrot Set, . . .

De Boor Algorithm -- Real Knots

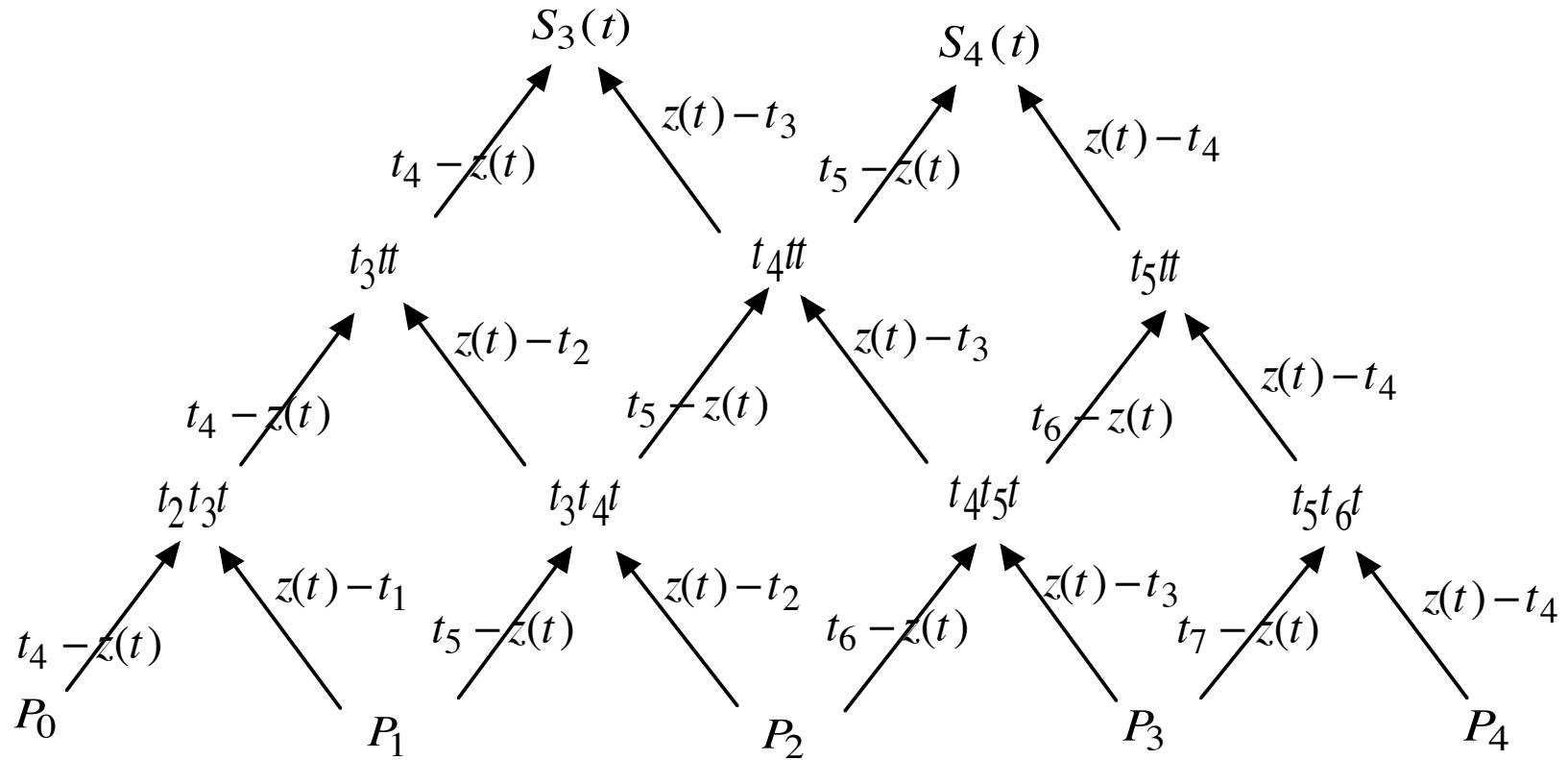


Two Polynomial Segments

$$S_3: t_3 \leq t \leq t_4$$

$$S_4: t_4 \leq t \leq t_5$$

De Boor Algorithm -- Complex Knots

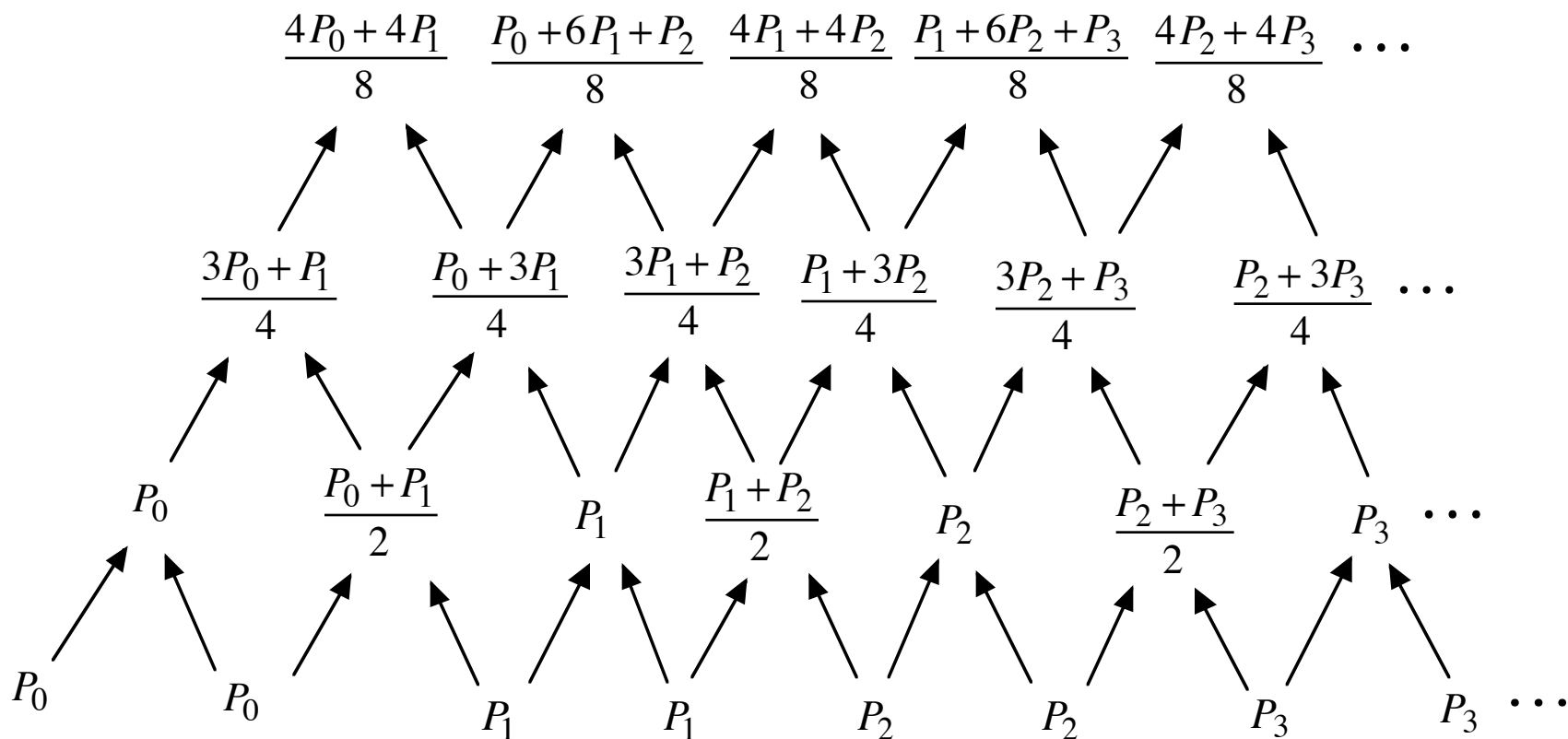


Two Polynomial Segments

$$S_3 : z(t) = (1-t)t_3 + tt_4 \quad 0 \leq t \leq 1$$

$$S_4 : z(t) = (1-t)t_4 + tt_5 \quad 0 \leq t \leq 1$$

Lane-Riesenfeld Algorithm:
Cubic B-splines -- Uniform Knots



Split and Average

Subdivision Matrices

Matrices

$$\bullet L(r) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1-r & r & 0 & 0 \\ (1-r)^2 & 2r(1-r) & r^2 & 0 \\ (1-r)^3 & 3r(1-r)^2 & 3r^2(1-r) & r^3 \end{pmatrix}$$

$$\bullet R(r) = \begin{pmatrix} (1-r)^3 & 3r(1-r)^2 & 3r^2(1-r) & r^3 \\ 0 & (1-r)^2 & 2r(1-r) & r^2 \\ 0 & 0 & 1-r & r \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Subdivision

$$\bullet \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix} = L(r) * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} \quad \bullet \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix} = R(r) * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Lifting the Control Points to Higher Dimensions

Quadratics

$$P^* = \begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}$$

Cubics

$$P^* = \begin{pmatrix} P_1 & 1 & 0 \\ P_2 & 1 & 0 \\ P_3 & 1 & 0 \\ P_4 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 \\ x_2 & y_2 & 1 & 0 \\ x_3 & y_3 & 1 & 0 \\ x_4 & y_4 & 1 & 1 \end{pmatrix}$$

Quartics

$$P^* = \begin{pmatrix} P_1 & 1 & 0 & 0 \\ P_2 & 1 & 0 & 0 \\ P_3 & 1 & 0 & 0 \\ P_4 & 1 & 1 & 0 \\ P_5 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 \\ x_4 & y_4 & 1 & 1 & 0 \\ x_5 & y_5 & 1 & 1 & 1 \end{pmatrix}$$

Iterated Function Systems

Observations

- $\text{Det} \begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \end{pmatrix} = \text{Det} \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix} = 2 \text{ Area}(\Delta P_1 P_2 P_3)$
- P_1, P_2, P_3 not collinear $\Leftrightarrow \text{Det} \begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \end{pmatrix} \neq 0 \Leftrightarrow (P^*)^{-1}$ exists

Iterated Function Systems (2-Dimensions)

- $L_P(r) = (P^*)^{-1} * L(r) * P^*$
- $R_P(r) = (P^*)^{-1} * R(r) * P^*$

Start with Any Shape in N -Dimensions and Project the Result
Generated by the IFS to 2-Dimensions

Bezier Curves as Projections of Normal Bezier Curves

Normal Bezier Curve

- $B(t) = (B_0^n(t), \dots, B_n^n(t)) = \sum_{k=0}^n C_k B_k^n(t)$
- $C_k = (0, \dots, 1, \dots, 0)$

Arbitrary Bezier Curve

- $P(t) = \sum_{k=0}^n P_k B_k^n(t)$
- $P = (P_0, \dots, P_n)$

Projections of Normal Curves

- $P : A^n \rightarrow A^2$
- $P(t) = B(t) * P^T$