Curves

Generation of Curves

- Circular Arc Generation
- Interpolation

Circular Arc Generation

$$x = R\cos\theta + x_0$$
$$y = R\sin\theta + y_0 \qquad \ldots (9.1)$$

where (x_0, y_0) is the center of curvature, and R is the radius of arc. (see Fig. 9.1)

Differentiating equation 9.1 we get

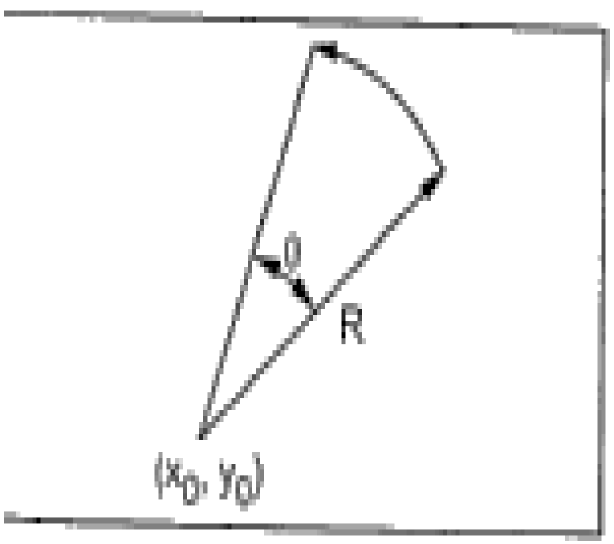


Fig. 9.1

$$dx = -R\sin\theta\, d\theta$$
$$dy = R\cos\theta\, d\theta \qquad \ldots (9.2)$$

From equation 9.1 we can solve for $R\cos\theta$ and $R\sin\theta$ as follows.

$$x = R\cos\theta + x_0$$
$$\therefore \quad R\cos\theta = x - x_0 \text{ and}$$

$$R \sin \theta = y - y_0 \quad \dots (9.3)$$

Substituting values of $R \cos \theta$ and $R \sin \theta$ from equation 9.3 in equation 9.2 we get,

$$dx = -(y - y_0)\, d\theta \text{ and}$$

$$dy = (x - x_0)\, d\theta \quad \dots (9.4)$$

The values of dx and dy indicate the increment in x and y increment, respectively, to be added in the current point on the arc to get the next point on the arc. Therefore, we can write

$$x_2 = x_1 + dx = x_1 - (y_1 - y_0)\, d\theta$$

$$y_2 = y_1 + dy = y_1 + (x_2 - x_0)\, d\theta \quad \dots (9.5)$$

Algorithm

1. Read the center of a curvature, say (x_0, y_0)
2. Read the arc angle, say θ
3. Read the starting point of the arc, say (x, y)
4. Calculate $d\theta$

   $d\theta = \text{Min}(0.01, 1 / (3.2 \times (|x - x_0| + |y - y_0|)))$
5. Initialize Angle = 0
6. While (Angle < θ)

   do

   { Plot (x, y)

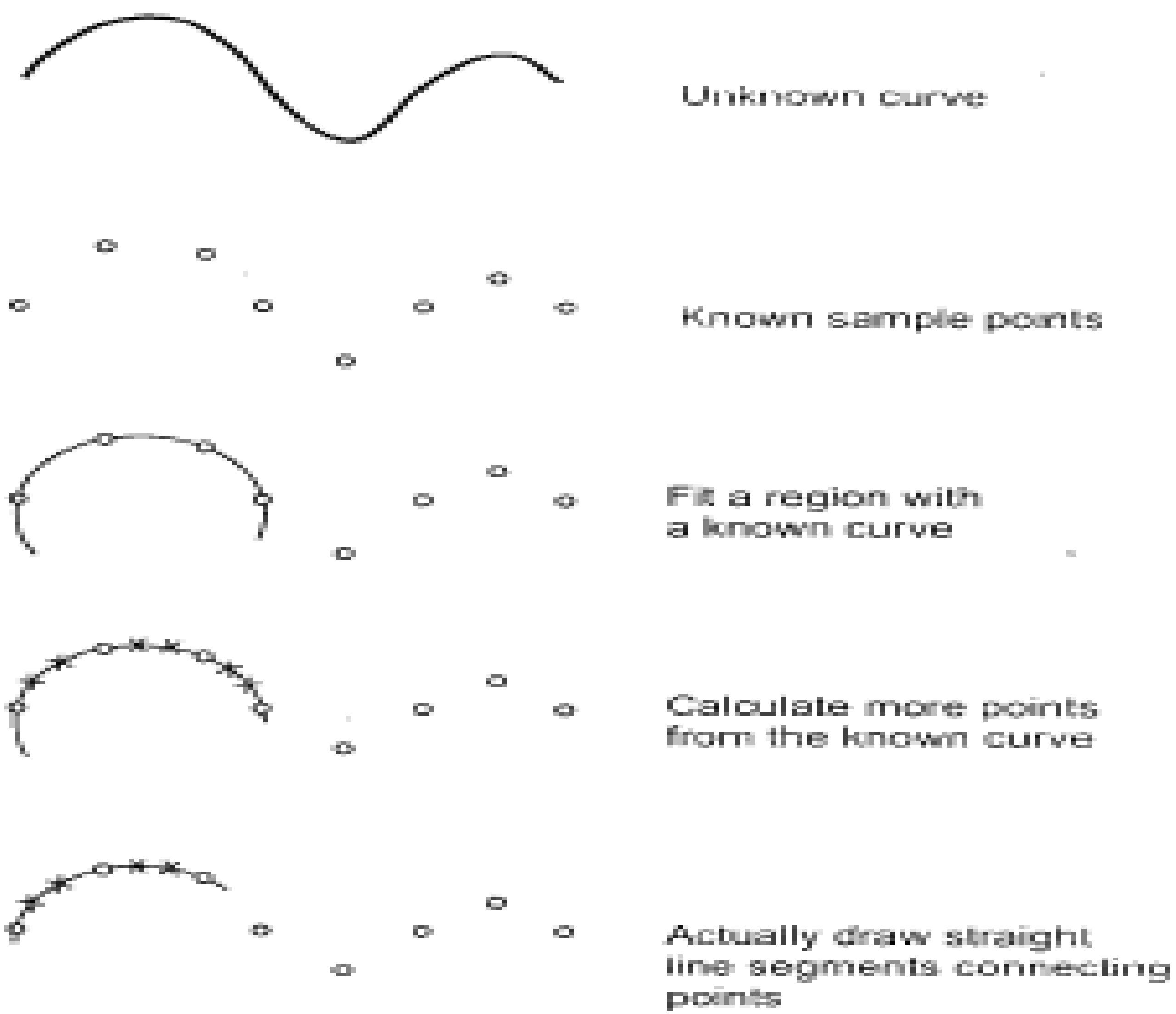
      $x = x - (y - y_0) \times d\theta$

      $y = y + (x - x_0) \times d\theta$

      Angle = Angle + $d\theta$

   }
7. Stop.

Interpolation



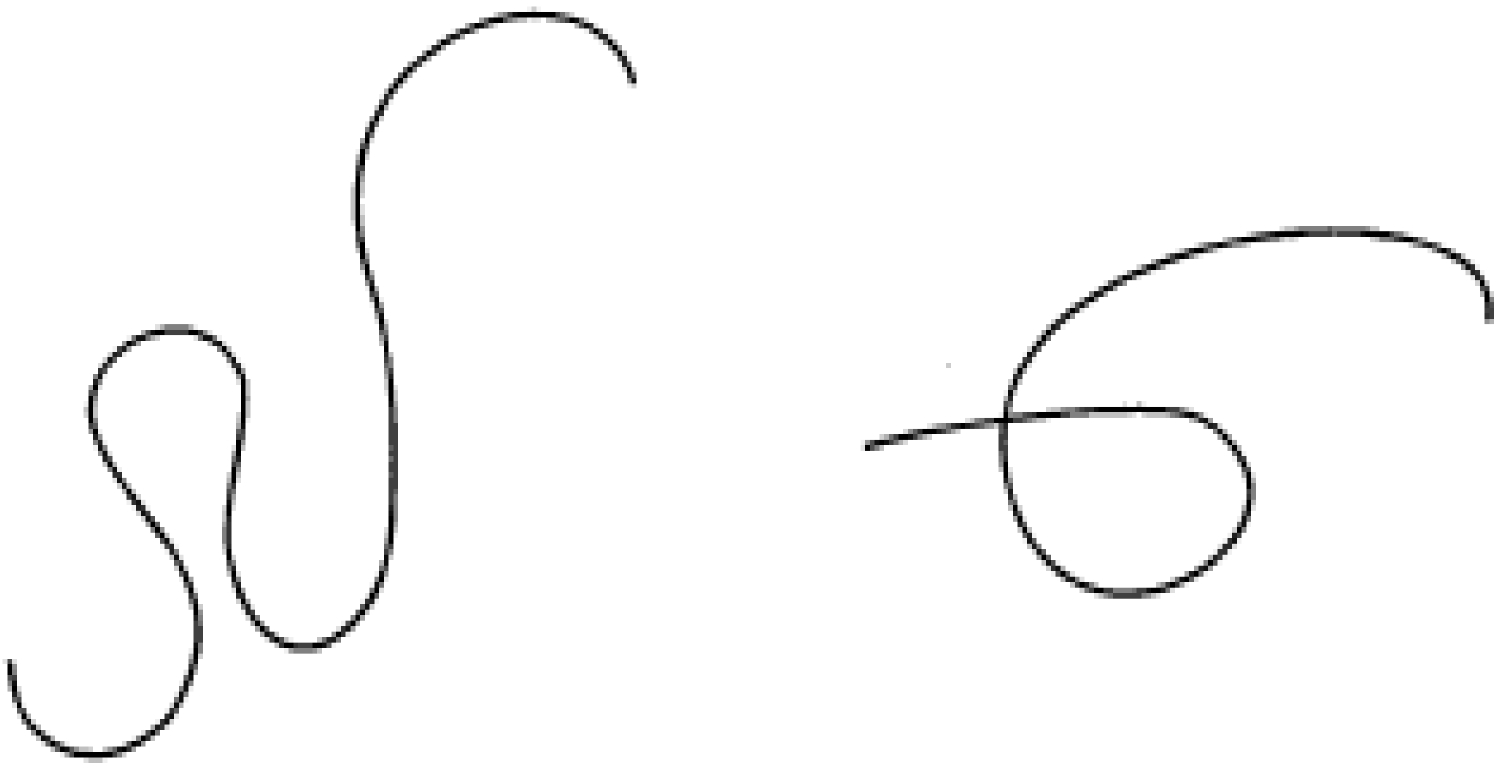


Fig. 9.3 Representation of curves with double back or crossing themselves

Interpolating Algorithm

1. Get the sample points.
2. Get intermediate values of u to determine intermediate points.
3. Calculate blending function values for middle section of the curve.
4. Calculate blending function values for first section of the curve.
5. Calculate blending function values for the last section of the curve.
6. Multiply the sample points by blending functions to give points on approximation curve.
7. Connect the neighbouring points using straight line segments
8. Stop.

TYPES OF CURVES

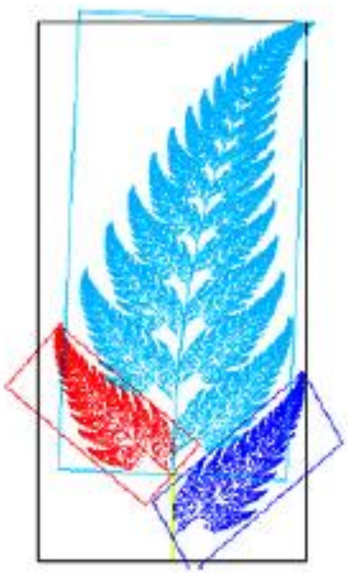
FRACTALS

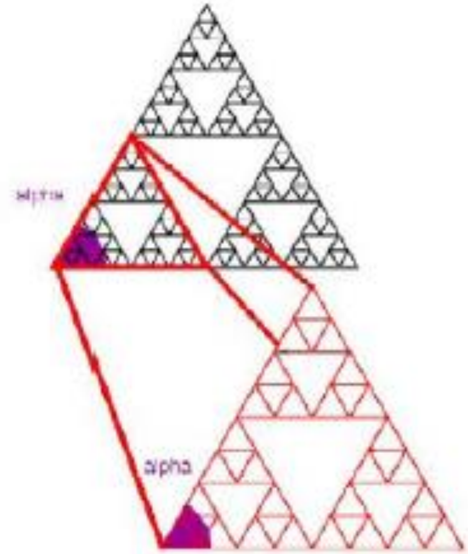
What are Fractals?

- Mathematical expressions to generate pretty pictures
- Evaluate math functions to create drawings
  - approach infinity -> converge to image
- Utilizes recursion on computers
- Popularized by Benoit Mandelbrot (Yale university)
- Dimensional:
  - Line is 1-dimensional
  - Plane is 2-dimensional
- Defined in terms of self-similarity

Fractals: Self-similarity

- See similar sub-images within image as we zoom in
- Example: surface roughness or profile same as we zoom in
- Types:
  - Exactly self-similar
  - Statistically self-similar



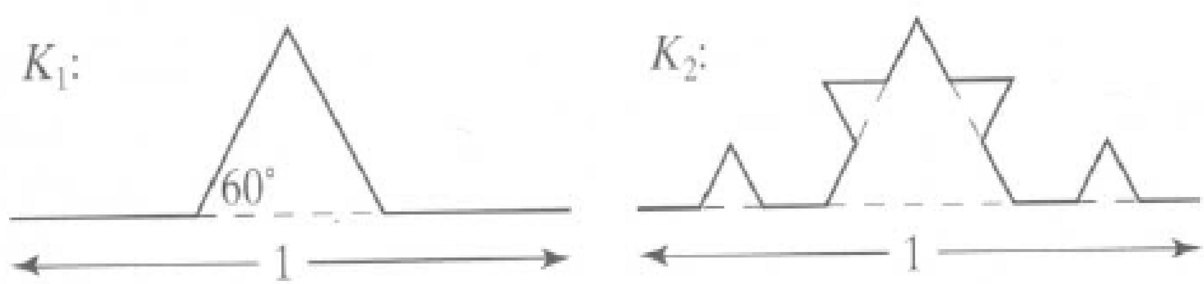


Examples of Fractals

- Clouds
- Grass
- Fire
- Modeling mountains (terrain)
- Coastline
- Branches of a tree
- Surface of a sponge
- Cracks in the pavement
- Designing antennae (www.fractenna.com)

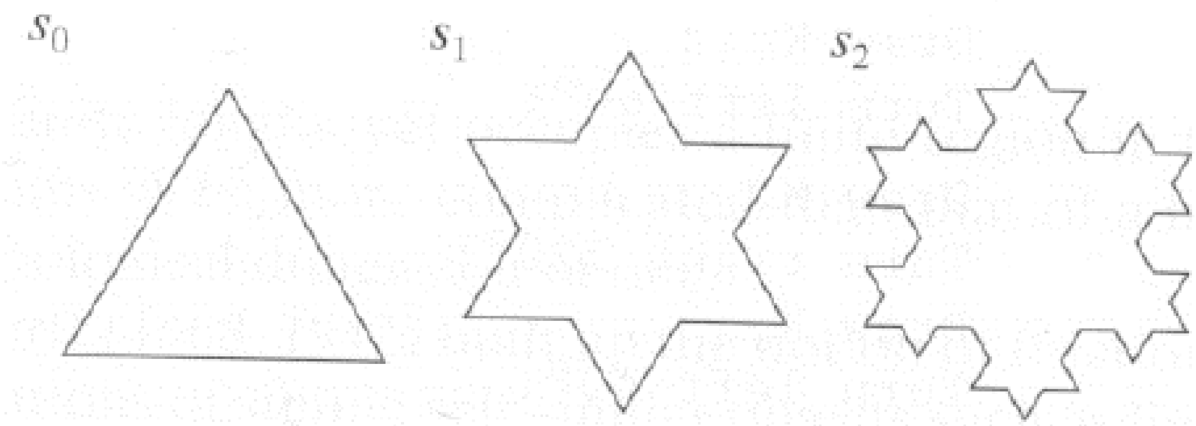
Koch Curves

- Discovered in 1904 by Helge von Koch
- Start with straight line of length 1
- Recursively:
  - Divide line into 3 equal parts
  - Replace middle section with triangular bump, sides of length 1/3
  - New length = 4/3

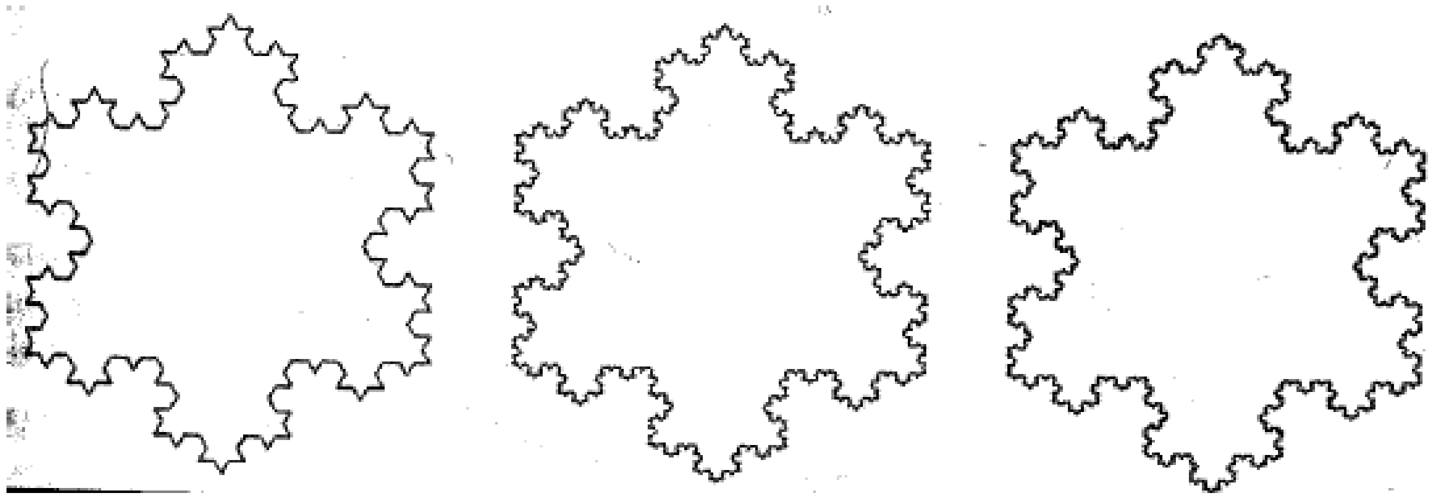


Koch Curves

Can form Koch snowflake by joining three Koch curves

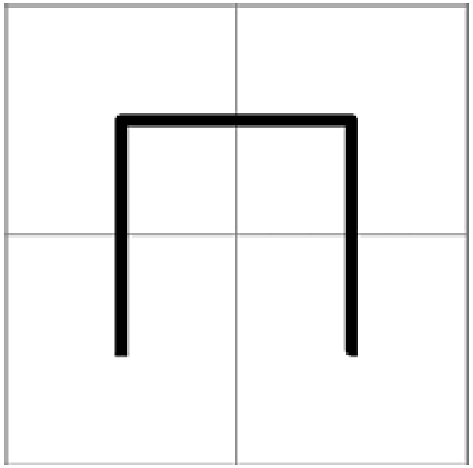


S_3, S_4, S_5,



Hilbert Curve

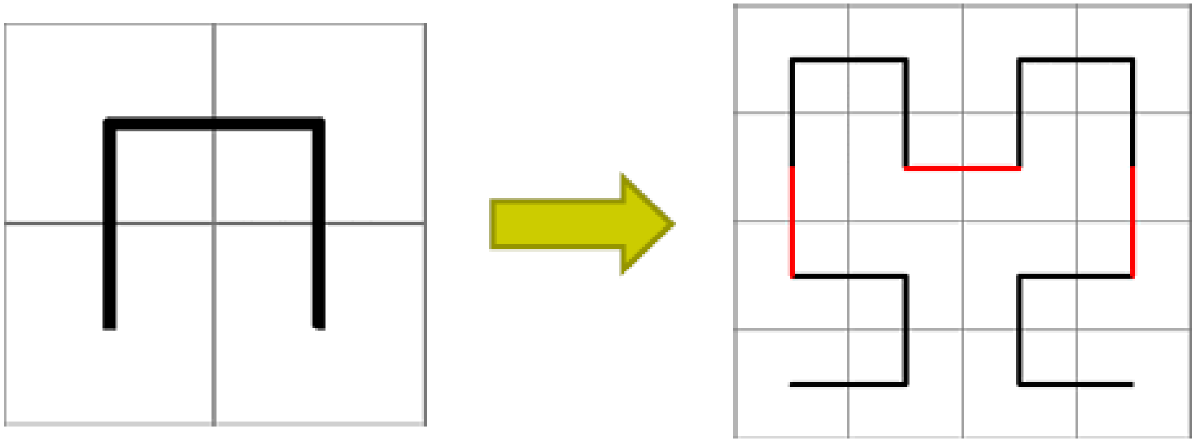
- Discovered by German Scientist, David Hilbert in late 1900s
- Space filling curve
- Drawn by connecting centers of 4 sub-squares, make up larger square.
- Iteration 0: 3 segments connect 4 centers in upside-down U



Iteration 0

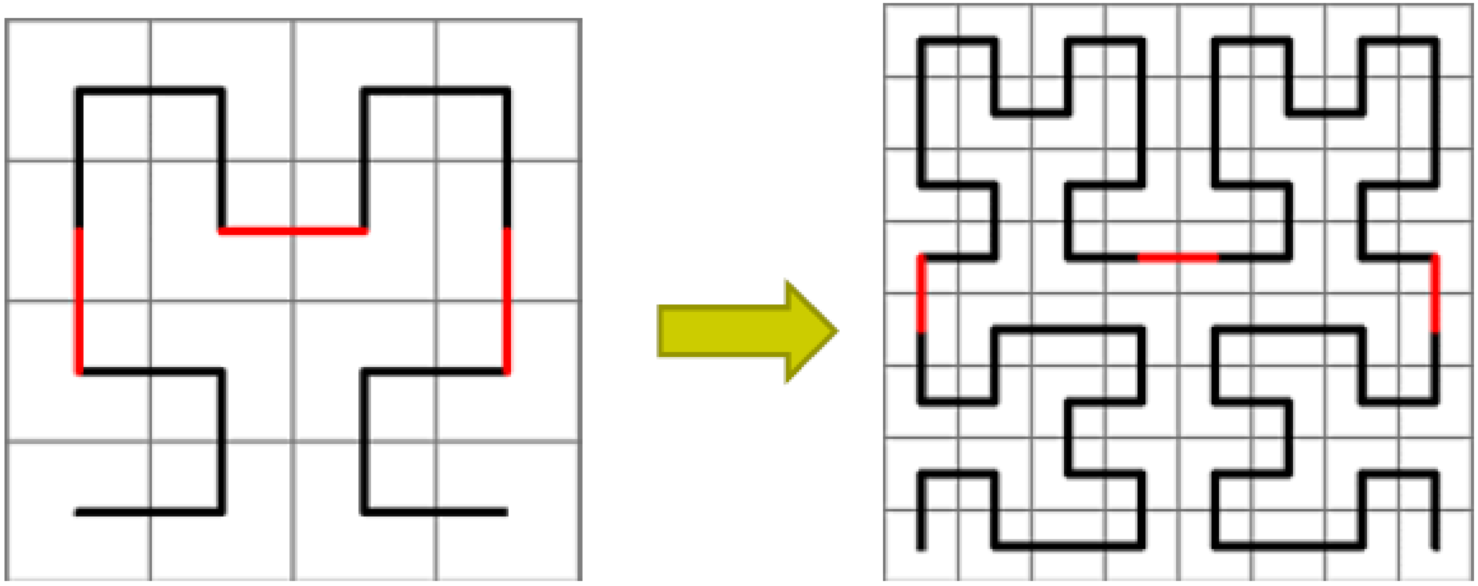
Hilbert Curve: Iteration 1

- Each of 4 squares divided into 4 more squares
- U shape shrunk to half its original size, copied into 4 sectors
- In top left, simply copied, top right: it's flipped vertically
- In the bottom left, rotated 90 degrees clockwise,
- Bottom right, rotated 90 degrees counter-clockwise.
- 4 pieces connected with 3 segments, each of which is same size as the shrunken pieces of the U shape (in red)



Hilbert Curve: Iteration 2

- Each of the 16 squares from iteration 1 divided into 4 squares
- Shape from iteration 1 shrunk and copied.
- 3 connecting segments (shown in red) are added to complete the curve.
- Implementation? Recursion is your friend!!



Spline Representation

Bezier Curves

Properties of Bezier curve

1. The basis functions are real.
2. Bezier curve always passes through the first and last control points i.e. curve has same end points as the guiding polygon.
3. The degree of the polynomial defining the curve segment is one less than the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is three, i.e. cubic polynomial.
4. The curve generally follows the shape of the defining polygon.
5. The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segments.
6. The curve lies entirely within the convex hull formed by four control points.
7. The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
8. The curve exhibits the variation diminishing property. This means that the curve does not oscillate about any straight line more often than the defining polygon.
9. The curve is invariant under an affine transformation.

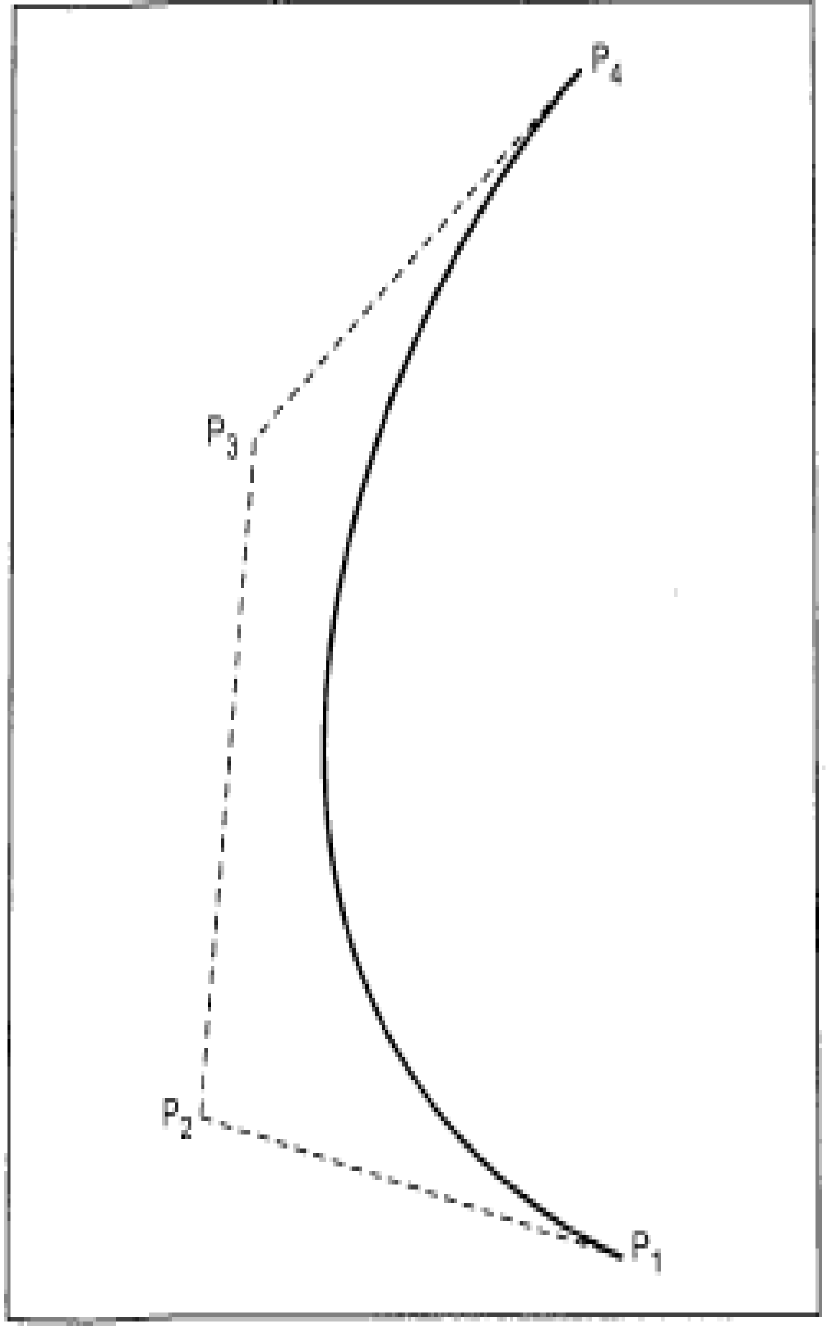


Fig. 9.6 A cubic Bezier spline

The Bezier matrix for periodic cubic polynomial is

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\therefore \quad P(u) = U \cdot M_B \cdot G_B$$

where
$$G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

and the product $P(u) = U.M_B.G_B$ is

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u) P_3 + u^3 P_4$$

Construct the Bezier curve of order 3 and with 4 polygon vertices A(1, 1), B(2, 3), C(4, 3) and D(6, 4).

Sol. : The equation for the Bezier curve is given as

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u) P_3 + u^3 P_4$$

for $0 \le u \le 1$

where P(u) is the point on the curve P_1, P_2, P_3, P_4

Let us take $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

$\therefore \quad P(0) = P_1 = (1, 1)$

$$\therefore \quad P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 P_1 + 3\frac{1}{4}\left(1-\frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right)P_3 + \left(\frac{1}{4}\right)^3 P_4$$

$$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(6,4)$$

$$= \left[\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 6, \quad \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 4\right]$$

$$= \left[\frac{123}{64}, \frac{139}{64}\right]$$

$$= (1.9218, 2.1718)$$

$$\therefore \quad P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 P_1 + 3\frac{1}{2}\left(1-\frac{1}{2}\right)^2 P_2 + 3\left(\frac{1}{2}\right)^2\left(1-\frac{1}{2}\right)P_3 + \left(\frac{1}{2}\right)^3 P_4$$

$$= \frac{1}{8}(1, 1) + \frac{3}{8}(2, 3) + \frac{3}{8}(4, 3) + \frac{1}{8}(6, 4)$$

$$= \left[\frac{1}{8}\times 1 + \frac{3}{8}\times 2 + \frac{3}{8}\times 4 + \frac{1}{8}\times 6, \quad \frac{1}{8}\times 1 + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}\times 4\right]$$

$$= \left[\frac{25}{8}, \frac{23}{8}\right]$$

$$= (3.125, \quad 2.875)$$

$$\therefore \quad P\left(\frac{3}{4}\right) = \left(1-\frac{3}{4}\right)^3 P_1 + 3\frac{3}{4}\left(1-\frac{3}{4}\right)^2 P_2 + 3\left(\frac{3}{4}\right)^2\left(1-\frac{3}{4}\right)P_3 + \left(\frac{3}{4}\right)^3 P_4$$

$$= \frac{1}{64}P_1 + \frac{9}{64}P_2 + \frac{27}{64}P_3 + \frac{27}{64}P_4$$

$$= \frac{1}{64}(1,1) + \frac{9}{64}(2, 3) + \frac{27}{64}(4, 3) + \frac{27}{64}(6,4)$$

$$= \left[\frac{1}{64}\times 1 + \frac{9}{64}\times 2 + \frac{27}{64}\times 4 + \frac{27}{64}\times 6, \quad \frac{1}{64}\times 1 + \frac{9}{64}\times 3 + \frac{27}{64}\times 3 + \frac{27}{64}\times 4\right]$$

$$= \left[\frac{289}{64}, \frac{217}{64}\right]$$

$$= (4.5156, \quad 3.375)$$

$P(1) = P_3 = (6, 4)$

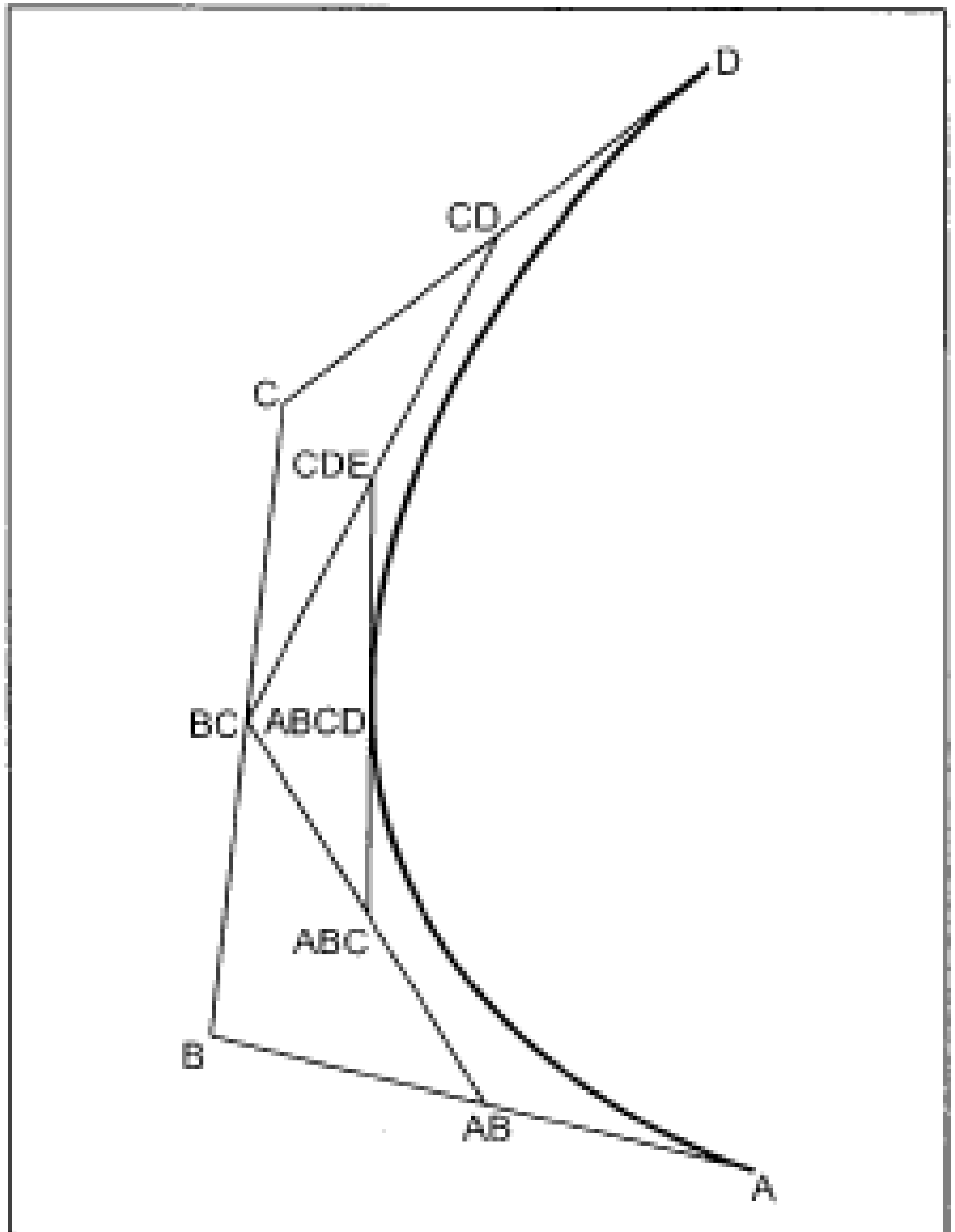


Fig. 9.8 Subdivision of a Bezier spline

Algorithm

1. Get four control points say A (x_A, y_A), B (x_B, y_B), C (x_C, y_C), D (x_D, y_D)
2. Divide the curve represented by points A, B, C and D in two sections

   $x_{AB} = (x_A + x_B) / 2$

   $y_{AB} = (y_A + y_B) / 2$

   $x_{BC} = (x_B + x_C) / 2$

   $y_{BC} = (y_B + y_C) / 2$

   $x_{CD} = (x_C + x_D) / 2$

   $y_{CD} = (y_C + y_D) / 2$

   $x_{ABC} = (x_{AB} + x_{BC}) / 2$

   $y_{ABC} = (y_{AB} + y_{BC}) / 2$

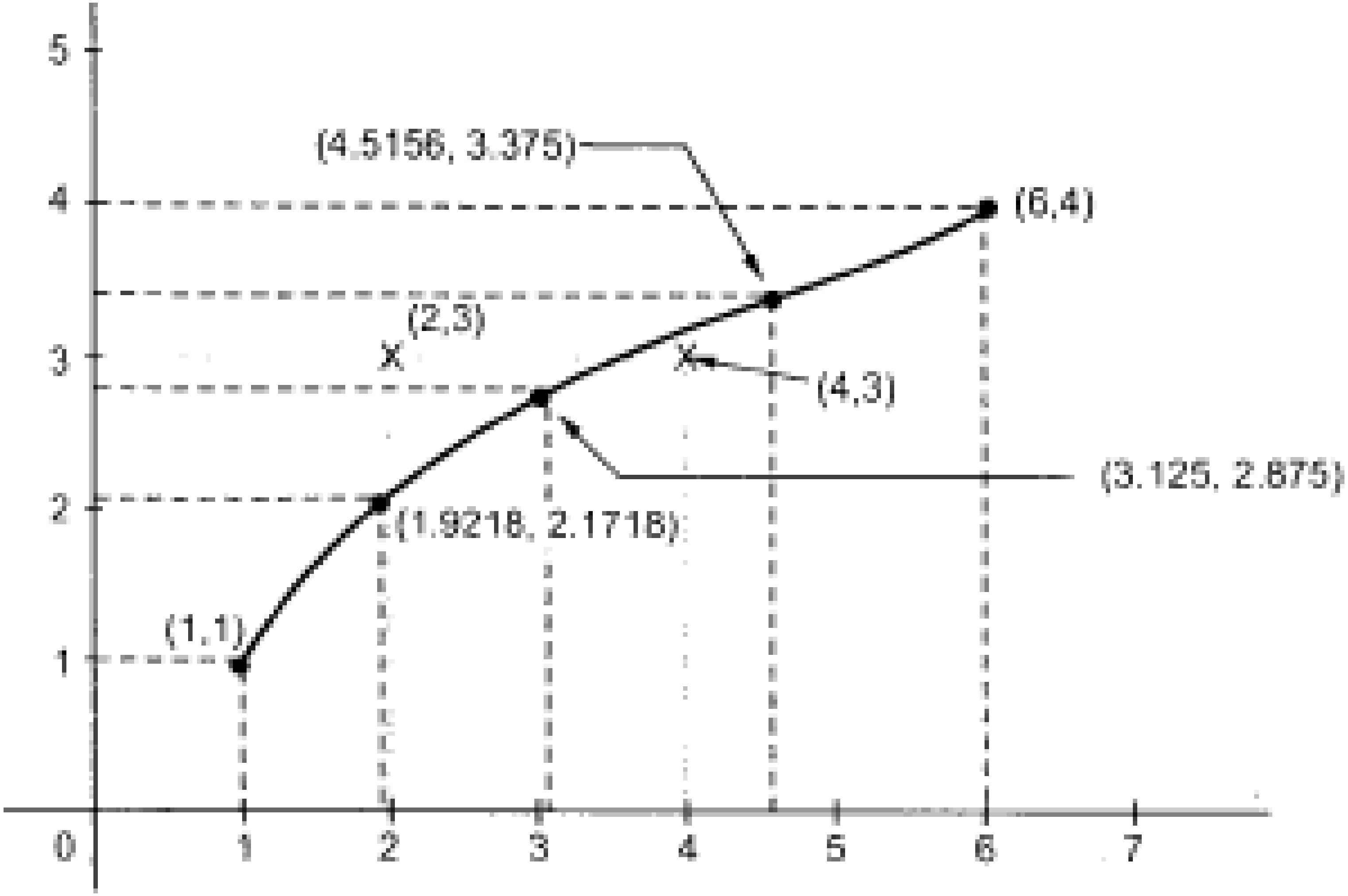
   $x_{BCD} = (x_{BC} + x_{CD}) / 2$

   $y_{BCD} = (y_{BC} + y_{CD}) / 2$

   $x_{ABCD} = (x_{ABC} + x_{BCD}) / 2$

   $y_{ABCD} = (y_{ABC} + y_{BCD}) / 2$
3. Repeat the step 2 for section A, AB, ABC and ABCD and section ABCD, BCD, CD and D
4. Repeat step 3 until we have sections so short that they can be replaced by straight lines.
5. Replace small sections by straight lines.
6. Stop

B – Spline Curve



Properties of B-spline curve

- The sum of the B-spline basis functions for any parameter value u is 1.

  i.e. $\sum_{i=1}^{n+1} N_{i,k}(u) = 1$

- Each basis function is positive or zero for all parameter values, i.e., $N_{i,k} \geq 0$.
- Except for k = 1 each basis function has precisely one maximum value.
- The maximum order of the curve is equal to the number of vertices of defining polygon.
- The degree of B-spline polynomial is independent on the number of vertices of defining polygon (with certain limitations).
- B-spline allows local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property. Thus the curve does not oscillate about any straight line move often than its defining polygon.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.