Unit V

Curves and Fractals

Curves: Introduction, Interpolation and Approximation, Blending function, B-Spline curve, Bezier curve,
Fractals: Introduction, Classification, Fractal generation: snowflake, Triadic curve, Hilbert curve, Applications.

Course Outcomes:

CO2: Apply mathematics to develop Computer programs for elementary graphic operations.
CO6: Create effective programs using concepts of curves, fractals, animation and gaming.

Text Books:

1. S. Harrington, “Computer Graphics”‖, 2nd Edition, McGraw-Hill Publications, 1987, ISBN 0 – 07 – 100472 – 6.
2. Donald D. Hearn and Baker, “Computer Graphics with OpenGL”, 4th Edition, ISBN-13: 9780136053583.
3. D. Rogers, “Procedural Elements for Computer Graphics”, 2nd Edition, Tata McGraw-Hill Publication, 2001, ISBN 0 – 07 – 047371 – 4.

Reference Books:

1. J. Foley, V. Dam, S. Feiner, J. Hughes, “Computer Graphics Principles and Practice”‖, 2nd Edition, Pearson Education, 2003, ISBN 81 – 7808 – 038 – 9.
2. D. Rogers, J. Adams, “Mathematical Elements for Computer Graphics”‖, 2nd Edition, Tata McGraw Hill Publication, 2002, ISBN 0 – 07 – 048677 – 8.

Unit : 5

Curves & Fractals

Curves :- [Reference – TB1]

Introductions –

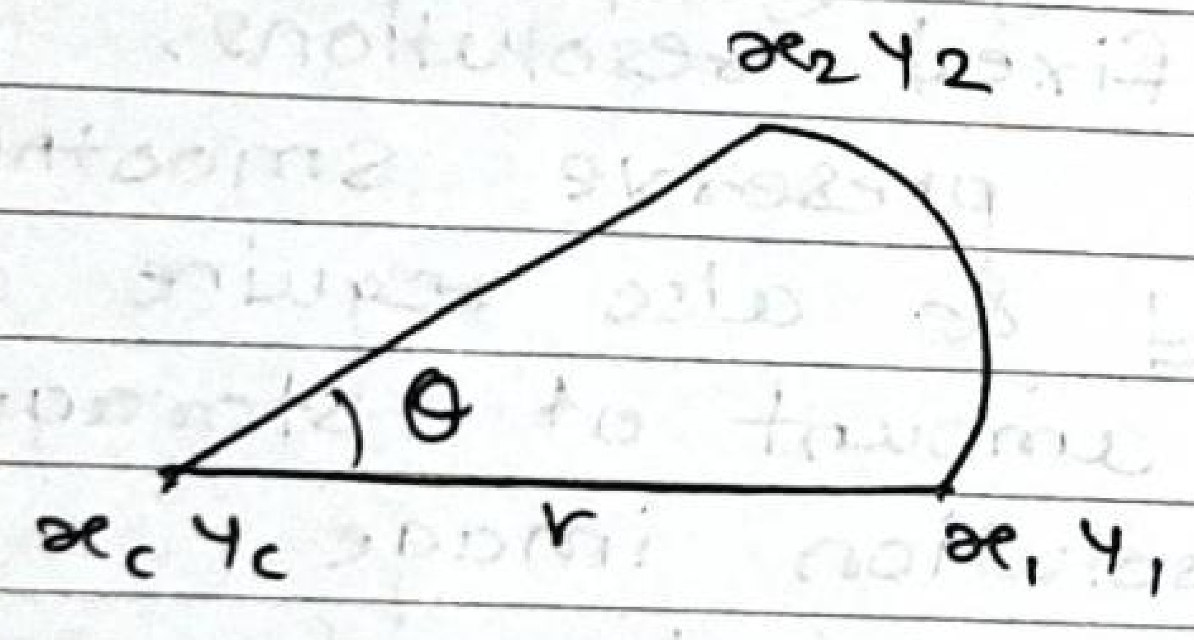
- we have seen how to draw lines, circles, polygons. To draw a line, we are having diff algo. by using these algorithms we can easily form any kind of straight lines.
- In nature are generally having rough & curved surfaces.
- many artificial objects are also having curved surfaces.
- Curves are one of the most essential objects to create high-resolution graphics
- while using many small polylines allows creating graphics that appears smooth at fixed resolutions.
- They do not preserve smoothness when scaled & also require a tremendous amount of storage for any high-resolution image.
- Curves can be stored much easier can be scaled to any resolution without losing smoothness & most importantly provide a much easier way to specify real world objects.

Curve Generation –

- for generation of curves, two methods/options are used
  1) use of a curve generation method
  2) Approximate the curve

★ Curve Generation Method :-

- we can use any readymade curve generation algo to draw a curve such as DDA.
- we can make use of circle generation algo also to form diff. curves.
- by using algo such as DDA, Bresenham circle generation or mid point circle generation, we will get actual curve.
- when we are using any circle generation algo to draw a curve, at that time, we should not draw complete circle but we should draw only some part of that circle which will act as curve.



- when we are using DDA algo to generate a circular arc, we need radius & some angle θ to draw curve.
- If we want to draw a curve from point (x_1, y_1) to $(x_2 y_2)$ i.e of angle θ, then we have to find coordinate of (x_1, y_1) as

$x_1 = x_c + r$ r is radius

& $y_1 = y_c + r$

Scanned by TapScanner

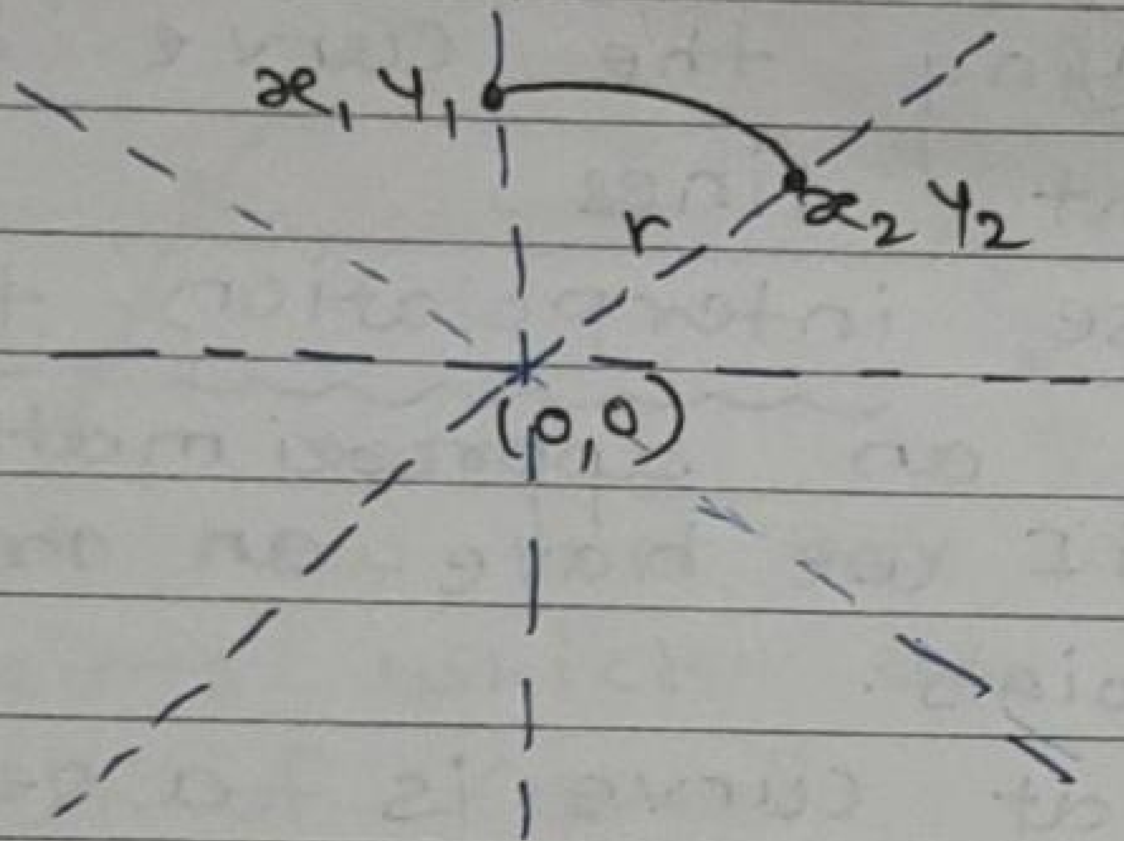
PAGE No
DATE

— This (x, y_1) will be the 1st point on curve & now to plot 2nd point on curve.

$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta$$

— where θ will be some stepping angle. like this we have to find all the points from (x, y_1) to $(x_2 y_2)$ on the path of curve. by increasing every time angle θ by some stepping value till θ will reach to final angle value.



— we can use any circle generation algo also to draw a curve.

— when we are using any circle generating algo. we should not use symmetry property of circle. because we have to plot only some part of circle.

— After finding any particular point on the path of curve.

— we have to plot only that point, we should not replicate it in remaining quadrants or octants.

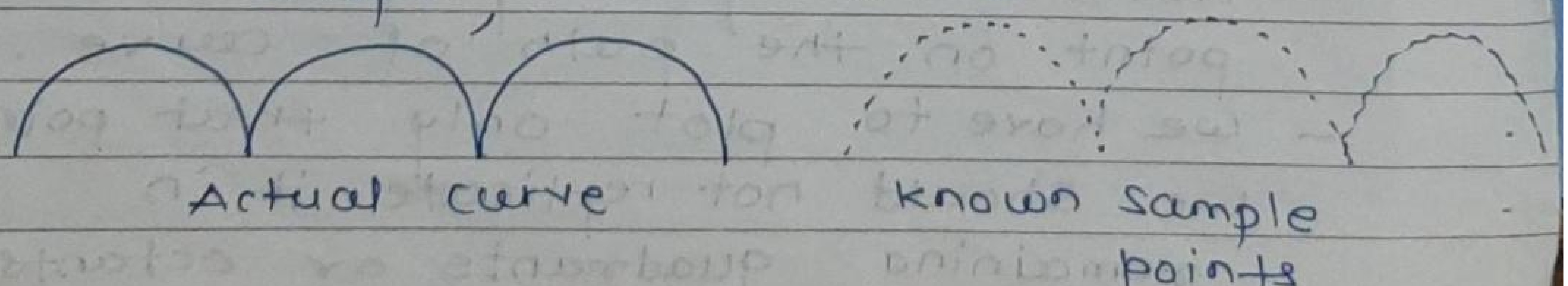
Scanned by TapScanner

Disadvantages :-

- we need more info^m than just its end points.
- when we scale a line, it remains line only. but if we scale a curve it may behave differently. i.e when a circle is scaled in one dire^n. it acts as an ellipse. So basic properties are getting change.
- clipping will be difficult for curves.

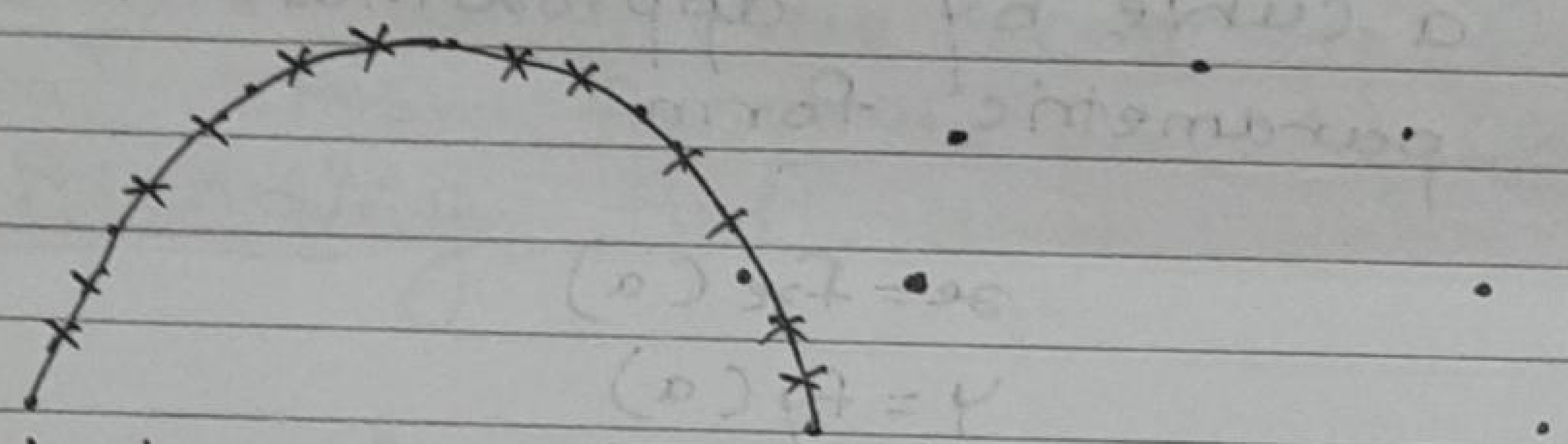
Approximation :-

- instead of using readymade algo, we are approximating the curve by small straight lines.
- we have to use interpolation technique. we can draw an approximation to a curve. if we have an array of sample points.
- we know that curve is a set of points or pixels. The sample points are nothing but some points on the estimated path of curve.
- from these few sample points we can guess the shape of the curve. If sample points are closer to each other, we can easily guess a correct curve.
- suppose we want to draw a curve of shape, which is as



we are having set of sample points.

- from the known sample points, we can draw a curve which can pass through the nearby sample points.
- Now we will try to fit the known curve in these sample points. we can fill the gaps betn the sample points.
- for this we have to find the coordinates betn two sample points on the curve & then we have to connect these points by using small line segments



- But how we are going to find the points which are lying on the path of curve?

Generally to draw a line or circle we need equation of line or circle. with the help of that equation we can find whether the specified point is lying on boundary of that line or circle.

- after placing a point in the equation of line or circle. if that equation becomes true then it means the specified point lies on that line or circle.
- for line we are having equation $y = mx + b$. for circle, eqn $= r^2 = x^2 + y^2$

PAGE No.
DATE

— if we want to draw any curve we must know the equation of that curve. So that we can place a point in that eqⁿ & if the equation becomes true it means the specified point is lying on curve. But here we do not know the eqⁿ of curve.

— There are many forms of funⁿ, which we can make we by adjusting parameters to find the points between the sample points.

— Normally functions are expressed in the form of $y = f(x)$ i.e y is a function of x, but to generate a curve by approximate method parametric form.

$$x = f_x(a)$$
$$y = f_y(a)$$
$$z = f_z(a)$$

— we are using parametric form because it considers all 3 dirⁿ equally & it allows multiple values so that curve can cross itself.

— suppose we want a curve that passes through n sample points

$(x_1\ y_1\ z_1), (x_2\ y_2\ z_2) \cdots (x_n\ y_n\ z_n)$

— As we are using parametric form, the eqⁿ of curve should be $x = f_x(a)$, it means we have to construct some funⁿ for this curve.

— As this curve is passing from n sample points the functions must be the sum of the n terms i.e one term for each sample point.

$$f_{(x)}(a) = \sum_{i=1}^{n} x_i B_i(a)$$

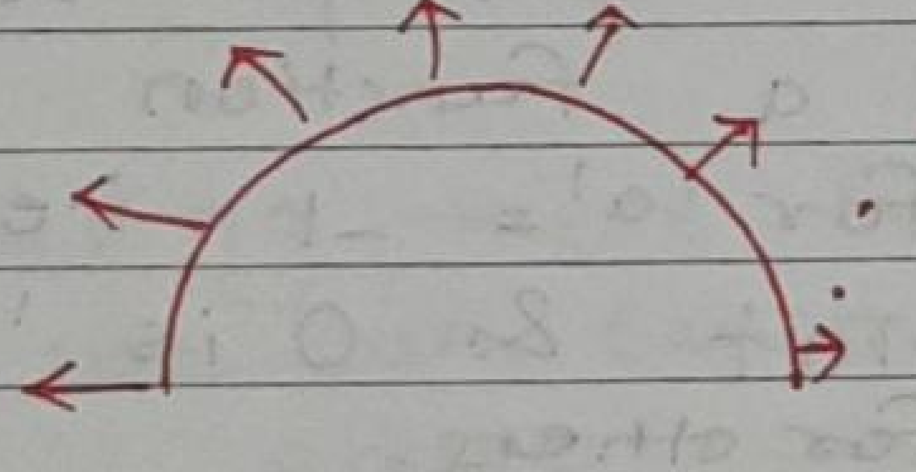
$$f_y(a) = \sum_{i=1}^{n} y_i B_i(a)$$

$$f_z(a) = \sum_{i=1}^{n} z_i B_i(a)$$

- for each function $f_x(a)$ $f_y(a)$ & $f_z(a)$ we are using one common funⁿ $B_i(a)$. This function $B_i(a)$ called <u>Blending</u> funⁿ.

Blending functions :-

- To achieve perfect curve, it must pass through all the sample points.
- it means each sample point tries to pull the curve in his direⁿ



- The curve will pass through the i^{th} sample point if for some value of 'a', $B_i(a) = 1$. It means the curve is passing through that point. The curve will not pass through the i^{th} sample point if the blending function's value is zero. if at a different value of 'a', one of the other sample points has complete control, then the curve will pass through this point as well.

- we have to develop a blending funⁿ such that its value could be 1 for the starting sample point.
- The value of 'a' is −1 for 1st point, 0 for 2nd, 1 for 3rd & so on.
- so our requirement is blending function's value for 1st sample point must be equal to 1.

it is slightly reverse way. we know result i.e funⁿs value must be 1 & from this we have to develop a funⁿ

i.e $B_i(a) = 1$

So we have to find,

$B_i(a) = ? = 1$

- To generate a smooth curve, it must pass through each sample point. for this we need a function which will be 1 for 'a' = −1 i.e for 1st sample point & 0 is 'a' = 0, 1, 2, ... n−2 i.e for others.

it could be

$a(a-1)(a-2) \ldots (a-(n-2))$

at $a = -1$

$(-1)(-2)(-3) \ldots (1-n)$

= some constant

But if we need $B_i(a) = 1$, then we will write as

$$B_i(a) = \frac{xyz}{const} = 1$$

In order to get $B_i(a) = 1$ the value of xyz must be $a(a-1)(a-2)\cdots(a(n-2))$

$$\therefore B_i(a) = \frac{a(a-1)(a-2)\cdots(a(n-2))}{(-1)(-2)(-3)\cdots(n-1)}$$

let's consider case where there are four sample points.

— In such situations we need four blending funⁿs.

$$B_1(a) = \frac{a(a-1)(a-2)}{(-1)(-2)(-3)} \qquad B_2(a) = \frac{(a+1)(a-1)(a-2)}{(1)(-1)(-2)}$$

$$B_3(a) = \frac{(a+1)(a)(a-2)}{(2)(1)(-1)} \qquad B_4(a) = \frac{(a+1)(a)(a-1)}{(3)(2)(1)}$$

using these funⁿ & four sample points, we can construct a curve which passes through four sample points

$$x = x_1 B_1(a) + x_2 B_2(a) + x_3 B_3(a) + x_4 B_4(a)$$
$$y = y_1 B_1(a) + y_2 B_2(a) + y_3 B_3(a) + y_4 B_4(a)$$
$$z = z_1 B_1(a) + z_2 B_2(a) + z_3 B_3(a) + z_4 B_4(a)$$

B-spline Curve

- Interpolation Technique used to generate a smooth curve it must pass through all sample points.
- It means the sum of blending functions must be 1. for every sample point for integer values of 'a'. we may get blending function as 1 but for fractional values of 'a' we may not be able to pass the curve through the sample points.
- Suppose five sample points are their. we may fix up a curve from these sample points as if blending funⁿ is 1 for all values of 'a'

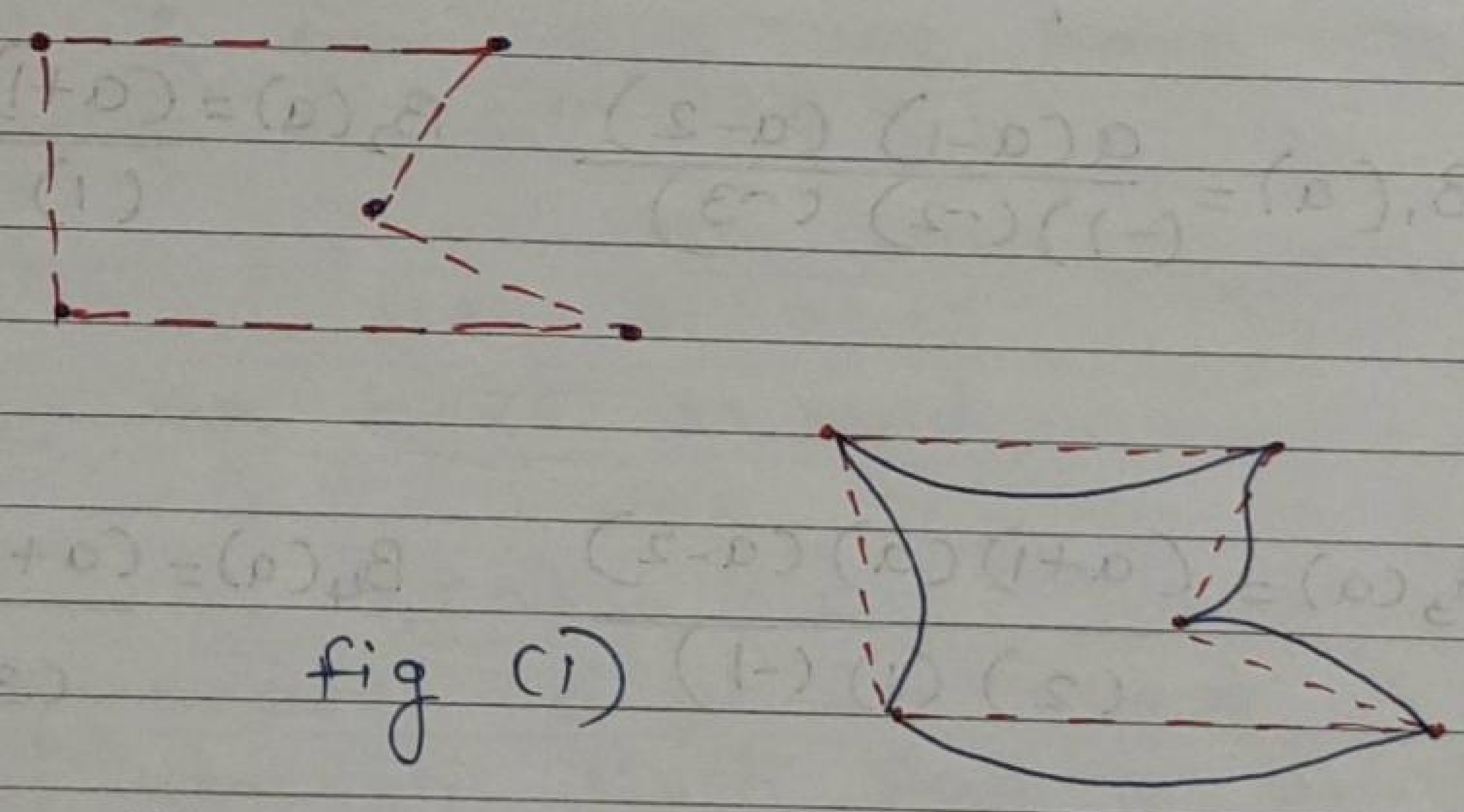
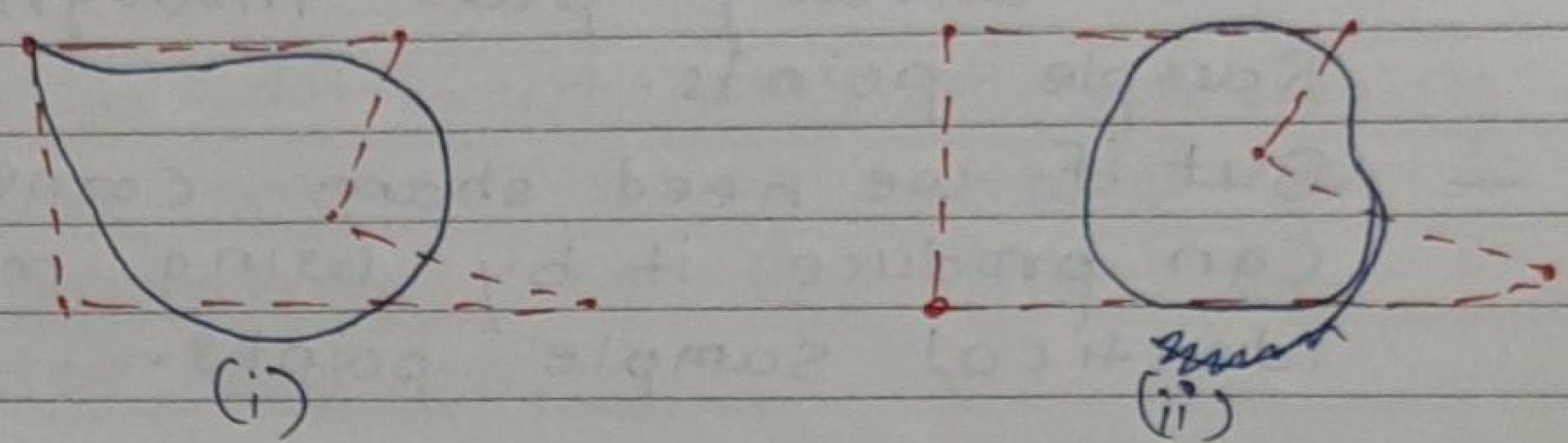


fig (i)

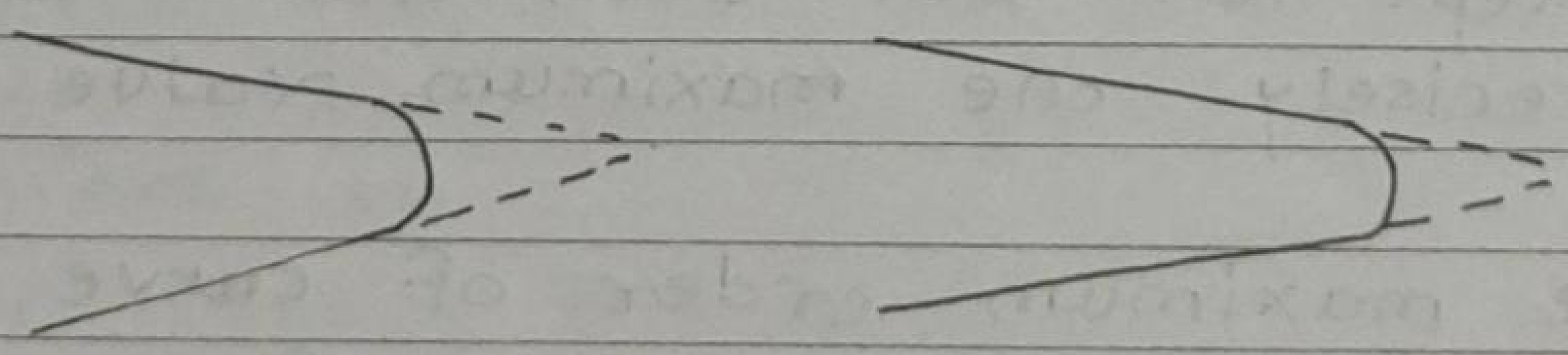
- we will come to know that each section of the curve is connected to the other section of curve at sample point. but the slopes of two sections may be diff so we are getting corners in the curve.
- It means the curve which we are getting is not smooth. in order to go near to the smoothness of curve

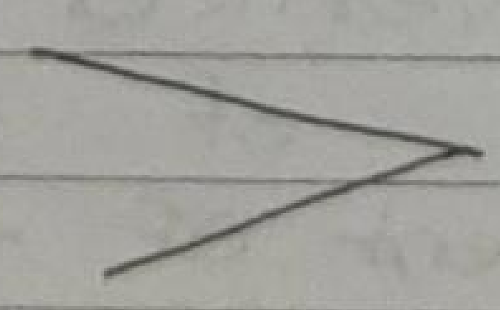
Scanned by TapScanner

- we will pull the curve in such a way that curve will pass through one sample point only. it ~~be~~ means blending funⁿ will be 1 for only one sample point.
- By using this technique we may eliminate the sharp Corners.



- But this will ~~go~~ not give us natural appearance of the curve. we will not try to force the curve to pass through the sample points, but rather gently pull it into the neighboor-hood of sample point.
- So by doing this we will get a smooth curve which is not passing through any of the sample points but still maintains the original shape of curve





- A set of blending functions which takes this approach is called B-splines.
- spline means a strip, which we have to move around the sample points.

- Generally B spline blending fun^n were designed to eliminate sharp corners in the curve & curve does not usually pass through the sample points.
- But if we need sharp corners we can produce it by using many identical sample points.

properties of B-spline curve :-

1. The sum of B-spline basis fun^n for any parameter value u is 1

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

2. each basis fun^n is positive or zero for all parameters value

   i.e $N_{i,k} >= 0$

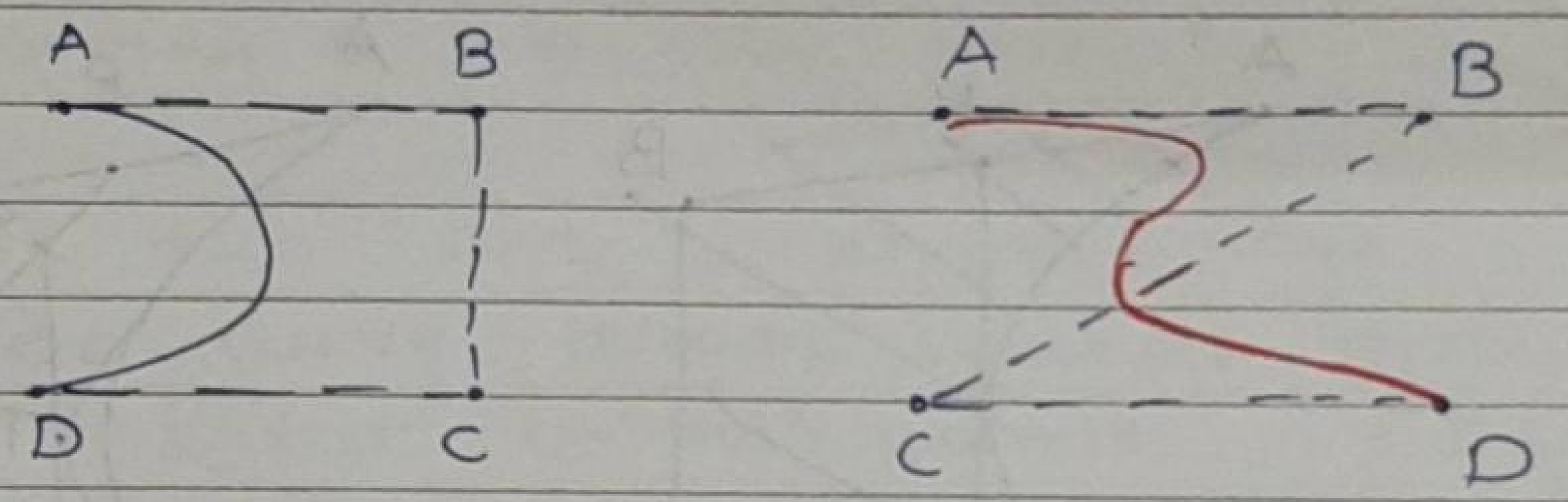
3. except for $k=1$ each basis fun^n has precisely one maximum value.

4. The maximum order of curve is equal to no. of vertices of defining polygon.
5. The degree of B-spline polynomial is independent of the no. of vertices of defining polygon.

6. B-spline allows local control over the curve surface because each vertex affects the shape of a curve only over range of parameter values where it is associated basic function is non zero.

Bezier Curve :-

- it is a different way of specifying a curve, rather same shapes can be represented by B-spline & Bezier curves.
-



- Curve begins at 1st sample point & ends at fourth point. if we need another Bezier curve then we need another four sample points.
- But if we need two Bezier curves connected to each other, then with six sample points we can achieve it.
- for this 3rd & 4th point of 1st curve should be made same as 1st & 2nd point of second curve.

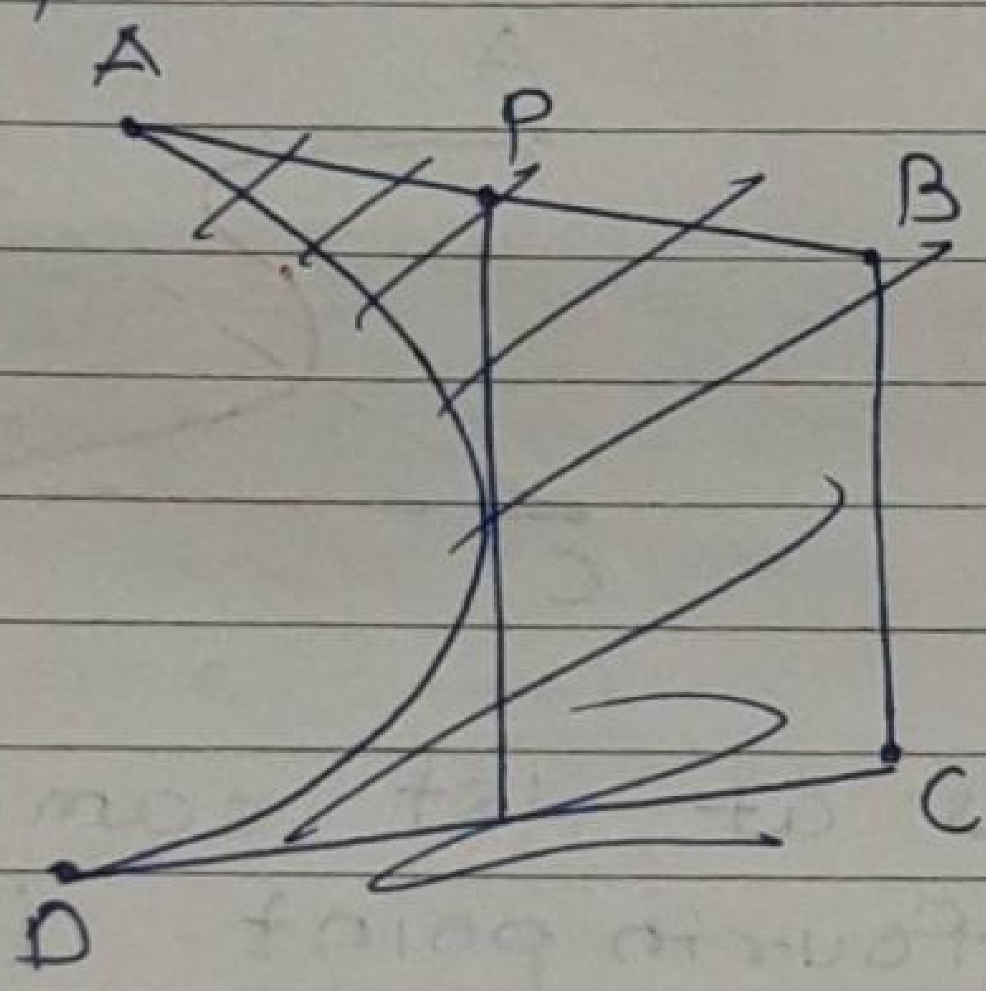
- The eq^n for Bezier curve

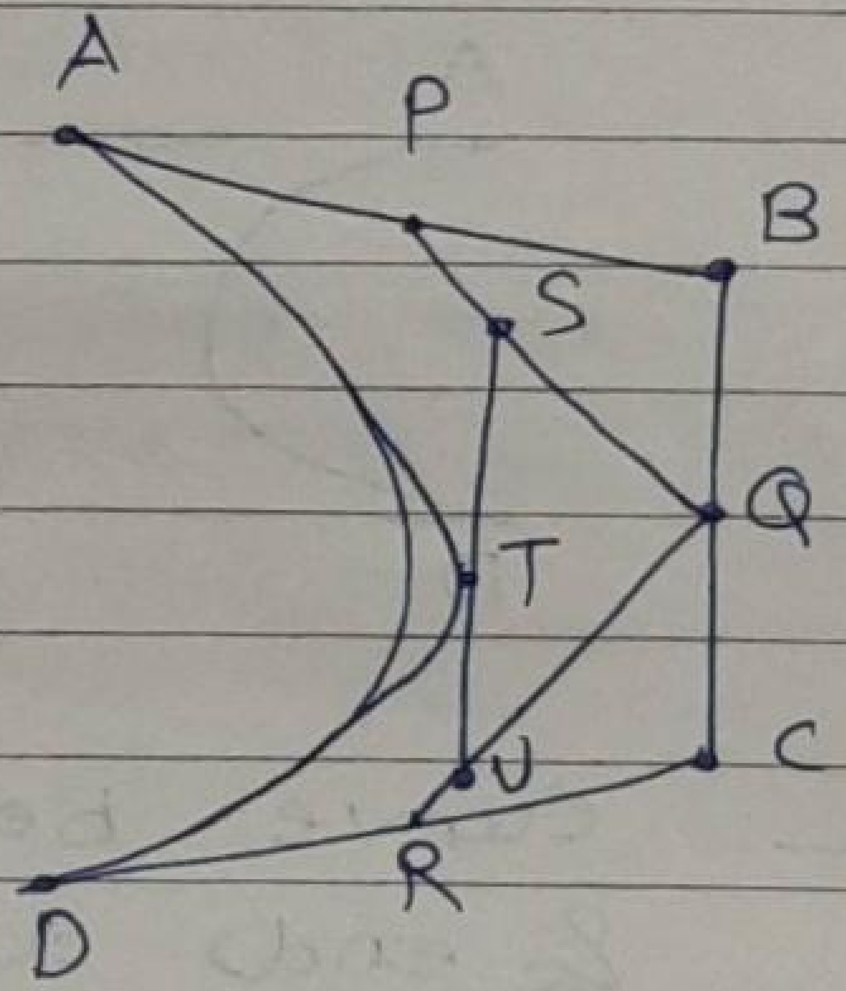
$$X = x_4 a^3 + 3x_3 a^2 (1-a) + 3x_2 a (1-a)^2 + x_1 (1-a)^3$$

$$Y = y_4 a^3 + 3y_3 a^2 (1-a) + 3y_2 a (1-a)^2 + y_1 (1-a)^3$$

$$Z = z_4 a^3 + 3z_3 a^2 (1-a) + 3z_2 a (1-a)^2 + z_1 (1-a)^3$$

- Here as value of 'a' moves from 0 to 1, curve travels from 1st to 4th sample point. But we can construct a Bezier curve without referencing to the above expression.
- it is constructed by simply taking midpoints.





- The point A, B, C, D are the original Bezier curve control points. Here we are having 3 lines, AB, BC & CD, then we have to find the midpoints of these lines as 'P', 'Q', 'R' respectively.
- after that we have to join PQ & QR. Then again find the mid point of these newly generated lines as 'S' & 'U'.
- Then form a line segment between 'S' & 'U' & find midpoint of this lines as 'T'. now point 'T' will be

on Bezier curve. This point 'T' divides the curve into two sections. One is (A, P, S & T) & second will be (D, R, U & T).

- Thus by taking midpoints, we can find a point on curve & also split the curve into two sections.
- We can continue to split the curve into smaller sections, until we have sections so short that they cannot be replaced by straight lines or till the size of section is not greater than size of pixel.

Properties of Bezier Curve :-

1. Basic funⁿ are real in nature.
2. Bezeier curve always passes through 1st & last control points. i.e curve has same end points as guiding polygon.
3. Curve generally follows the shape of the defining polygon.
4. The degree of polynomial defining the curve segment is one less than the no. of defining polygon point. Therefore for 4 control points the degree of polynomial is 3. i.e cubic polynomial.
5. The Direction of the tangent vector at the end points is the same as that of the vector determined by 1st & last segments.

Fractals :-

Introduction :-

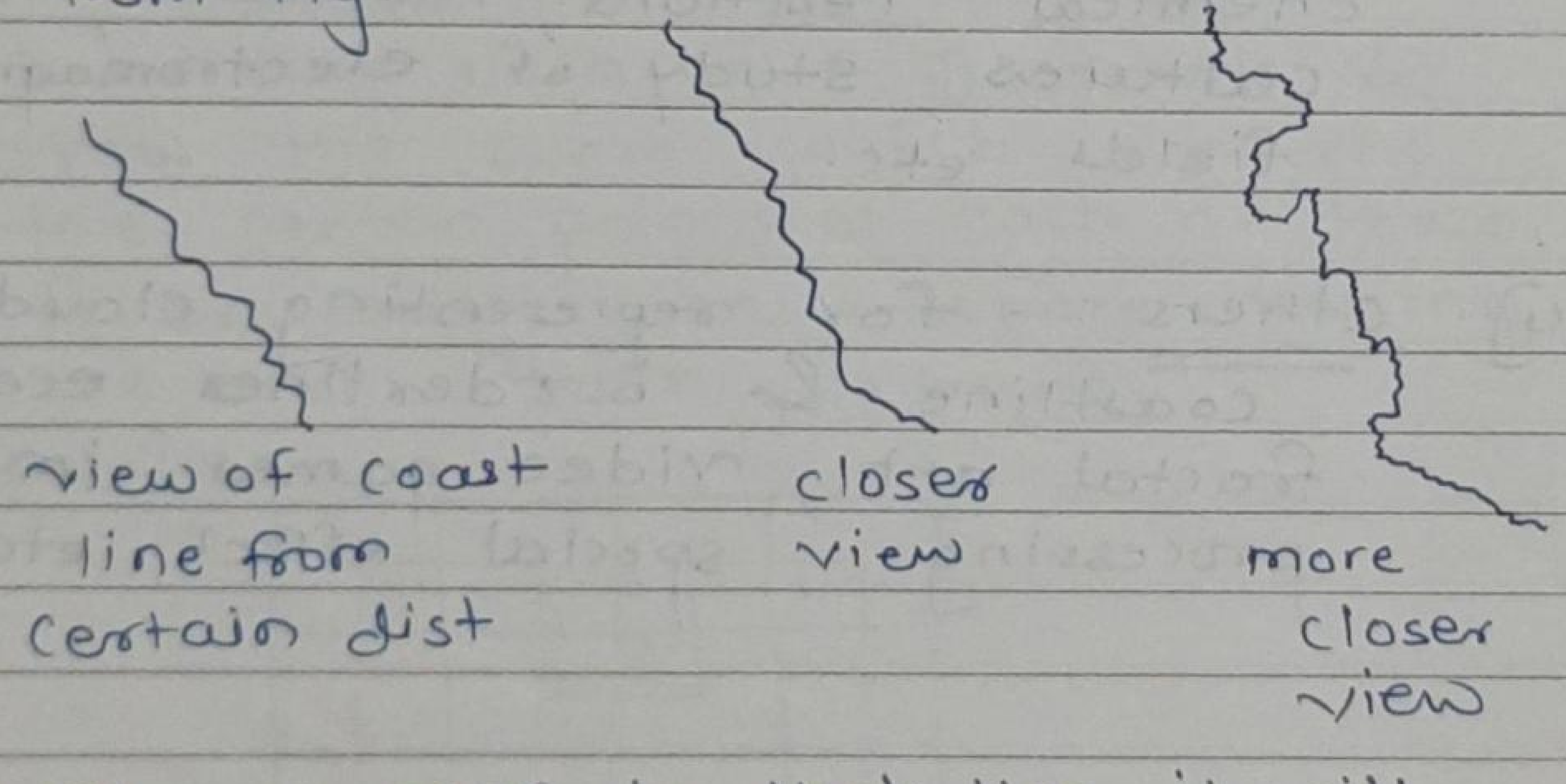
- The objects which are having smooth surfaces & regular shapes are generally described by using equations.
- But natural objects like mountains, trees ocean waves & clouds have irregular shapes.
- it will be very difficult to draw these shapes by using normal equation.
- There are many methods of modeling these natural objects, but one of the most intresting from a mathematical perspective is that of fractals.
- we can describe natural objects by using fractals, where procedures rather than equations are used to model the objects.
- procedurally defined objects have characteristics quite diff. from objects described with equations.
- One of the basic properties that characterize fractals is self similarity.
- self similarity property of an object can take diff forms, depending on the choice of fractal representation.
- self similarity means if we zoom into a piece of a fractal we will keep seeing the same stru. repeated over & over.

Classification -

* Based on self similarity property we can classify fractals in 3 forms

1) exact self similarity :- in this type the fractal appears exactly same or identical at diff scales. it is a strong from of self similarity.
2) Quasi-self Similarity :- in this type fractal appears approximately but not exactly identical at diff. scales.
3) statistical self-similarity :- it is the weakest type of self similarity. in this type of fractal has mathematical parameters maintained throughout the scales. Random fractal is good e.g of this category.

- from a certain dist. we will see coastline as simple, quite smooth line. from fig.



- as we go near to that line, it will appear more rough. if take closer view we will see the whole line as jaggy & rough. There is no limit how many times we are zoom in.

- fractals are complex pict. generated by comp. from a single formula. They are created using repetitive steps.
- This means one formula is repeated again & again with slight variation in values.
- every next iterative step depends on results of previous step.

Application -

1] **Space Research** :- for study of galaxies, rings of saturn, simulation of stars & their behavior, generation of stars & planets.

2] **Medical Science** :- for analyzing bacteria cultures, study of jeans, human anatomy etc.

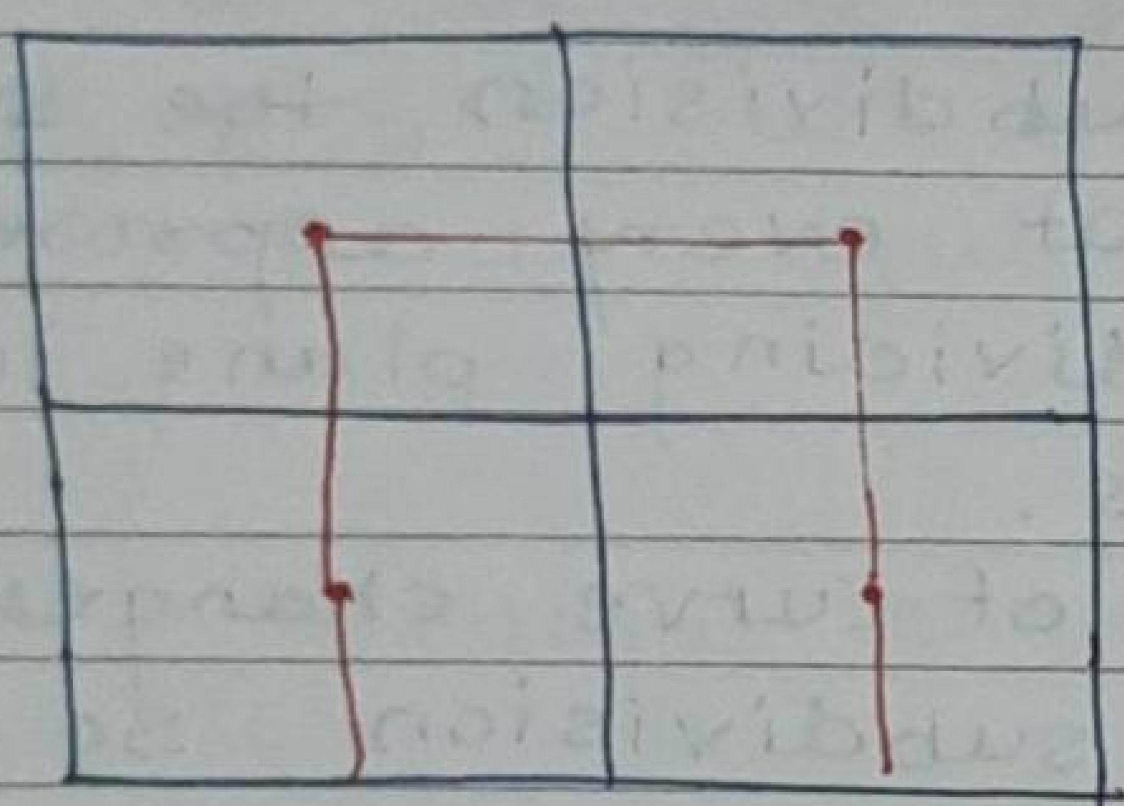
3] **chemical Research** :- for simulation of chemical reactions molecules, atomic cultures, study of electromagnetic fields etc.

4] **Others** :- for representing clouds, coastline & borderlines economy, fractal art, video games, image processing, special effect. etc.

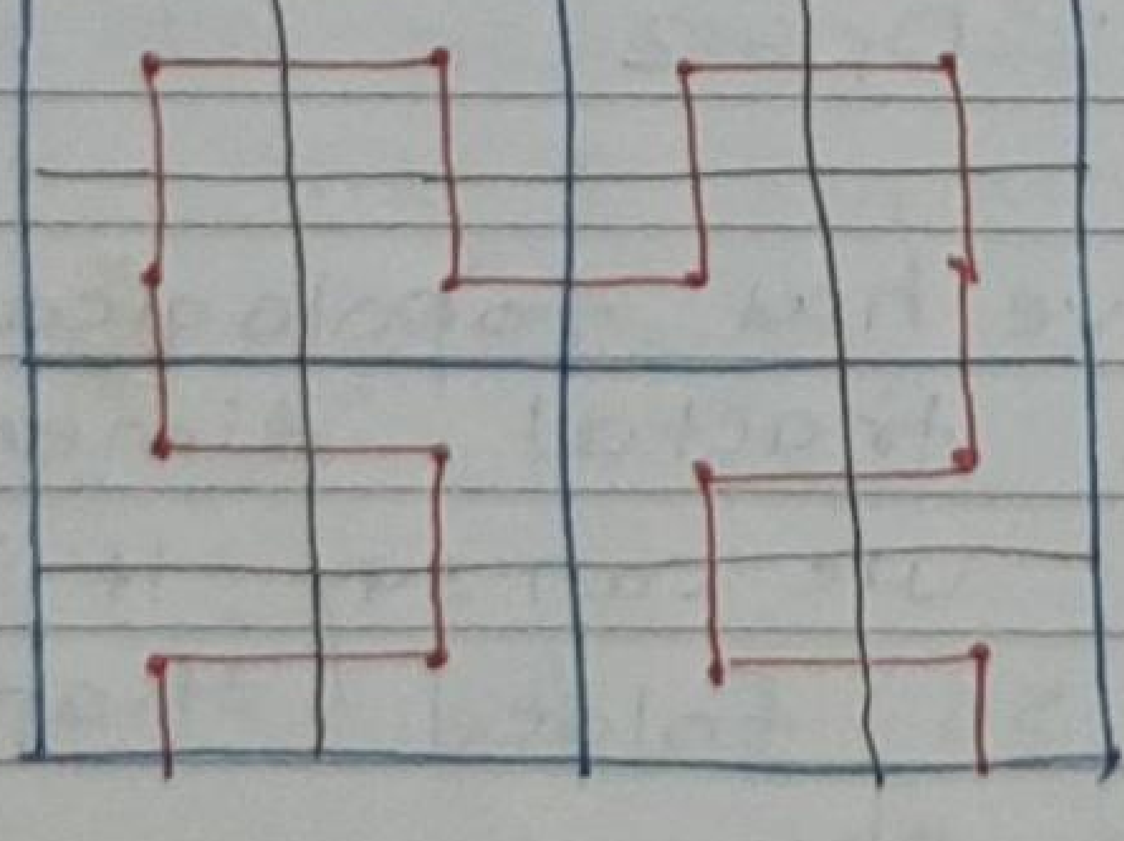
Fractal Generation :-

1] Hilbert's curve :-

- is also called as peano curve.
- very easy to implement.
- curve begins with initial square.
- The generation of curve requires successive approximations.
- In 1st approximation, we are dividing square into 4 quadrants & then drawing curve which connects the center points of each quadrant.



- 2nd approximation will be further subdivide each of 4 Quadrants & draw the curve which connects the center points of each of these finer subdivisions before moving the next major quadrant.



- 3^{rd} approximation again subdivides each quadrant & process continues.
- we will come to know that the curve is not getting overlap at any point.
- Curve is arbitrarilly close to every point in the square. There is no limit to this subdivision.
- Curve fills the square. ideally the length of curve is infinite. with each subdivision the length increases by factor 4. since this curve is equivalent to line only. its topological dimension (D_t) must be equal to 1.

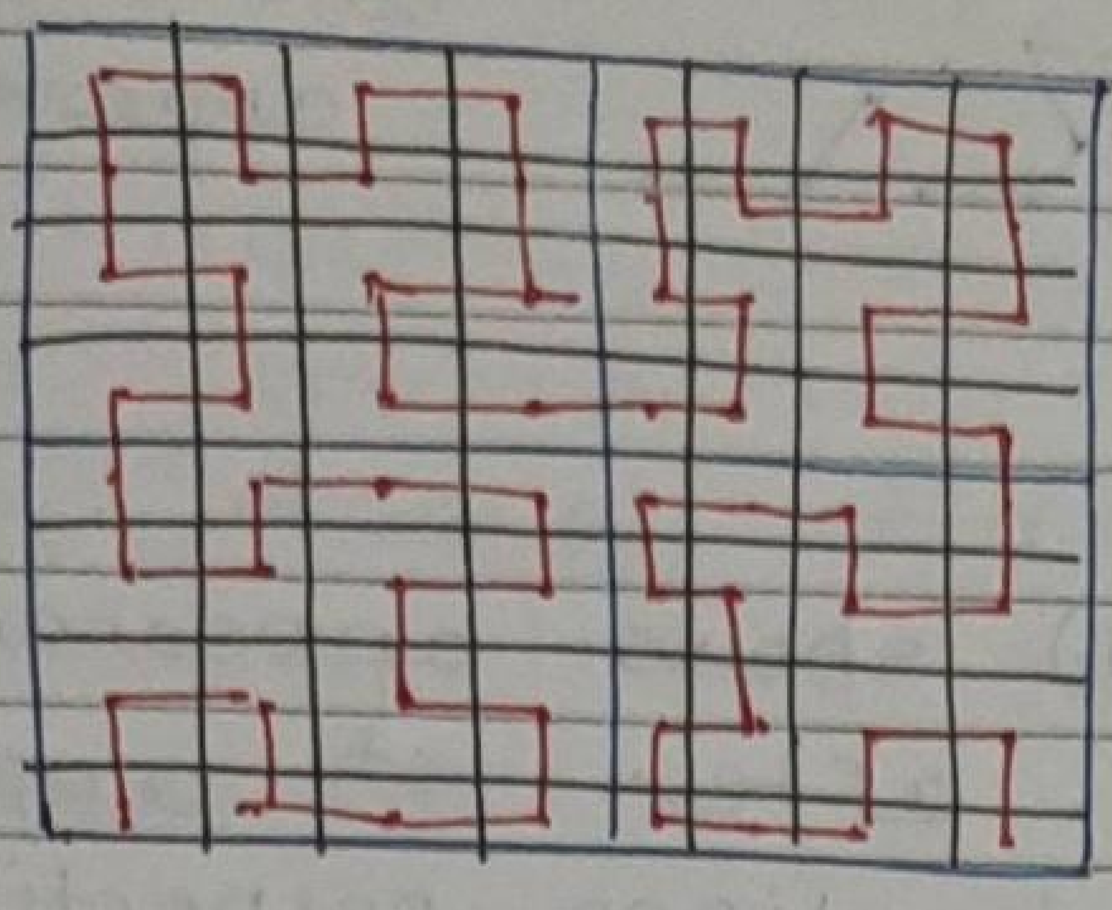
- At each subdivision the scale changes by 2 i.e at every approximation we are dividing plane into 4 quadrants.
- But length of curve changes by 4, at each subdivision. so we need 4 scaled curves to form original curve.

$$\text{i.e } N = S^{D_f}$$

$$\therefore 4 = 2^{D_f}$$

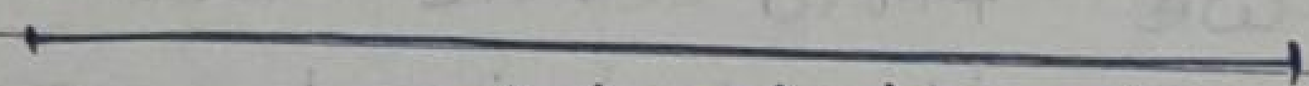
$$\therefore D_f = 2$$

- Hilbert curve has topological dimension (D_t) = 1 & fractal dimension (D_f) = 2. We can say, it is line only. but it is so folded that it looks like 2D obj.



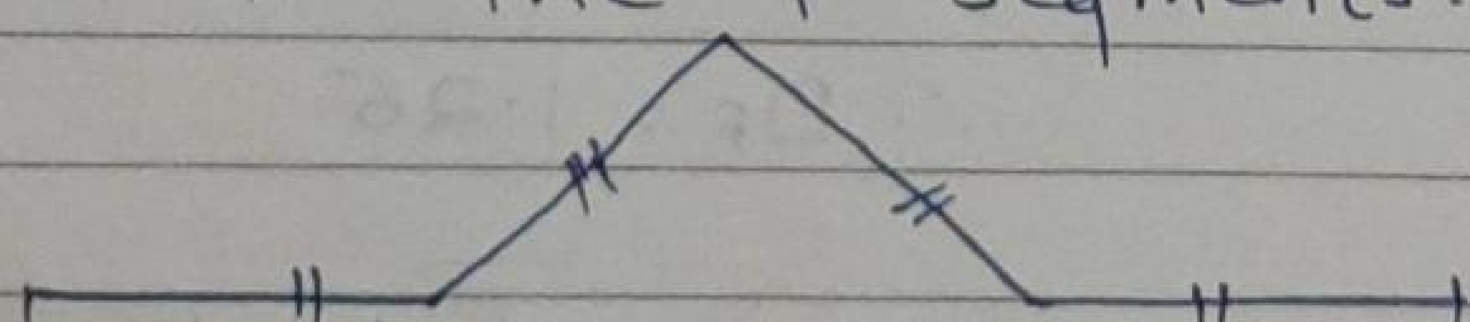
Triadic Koch Curve :-

- for this curve, we have to start with line segment.
- Then we are dividing that line segment into 3 equal parts.
- after that we are replacing the central 3rd part by two adjacent sides of an equilateral triangle.



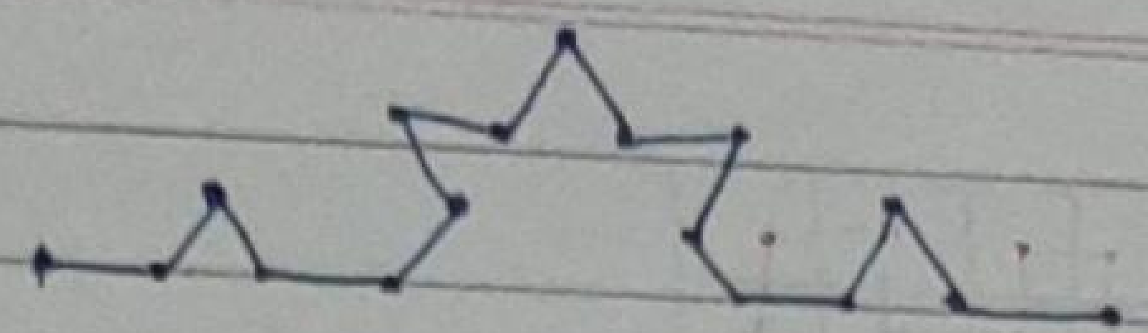
original length
(i)

- This gives us a curve whose end co-ordinates are same as that of original segments but is built of 4 equal length segments, each 1/3 the original length.
- new length of curve will be 4/3 of original length.
- we have to repeat this process for each of the 4 segments.



1st approximation

(ii)



2nd approximation

(iii)

fig (ii) & fig (iii) shows appearance of Koch curve after 1st approximation & 2nd approximations, respectively.

- after 2nd approximation the length of the curve becomes 16/9 times the original.

- length of Koch curve is infinite.
- This curve doesn't fill the whole area, like Hilbert's curve.
- it doesn't deviate much from its original shape.
- if we reduce the scale of curve by 3, we find the curve looks just like the original one, but we must gather 4 such curves to make the original, so we will say

$$4 = 3^{D_f}$$

Solving for Df

$$\therefore D_f = \frac{\log 4}{\log 3}$$

$$\therefore D_f = 1.26$$

- ∴ Koch curve topological dimension is one & fractal dimension is 1.26.
- Curve whose fractal dimension is greater than its topological

dimension then they are called as fractals.

- so Hilbert's curve & koch curve are fractals.
- their fractal dimension are greater than their topological dimension.

Snowflakes :-

- Koch curve can be extended to generate snowflakes.
- for generation of snowflakes we need to apply koch curve to every edge of a triangle as fig. (1)
- The length of each edge of a triangle goes on increasing with more & more roughness, in no. of iterations
- if we zoom a particular part of snowflake.
- we will come to know that it is process exactly self similarity property of fractals.