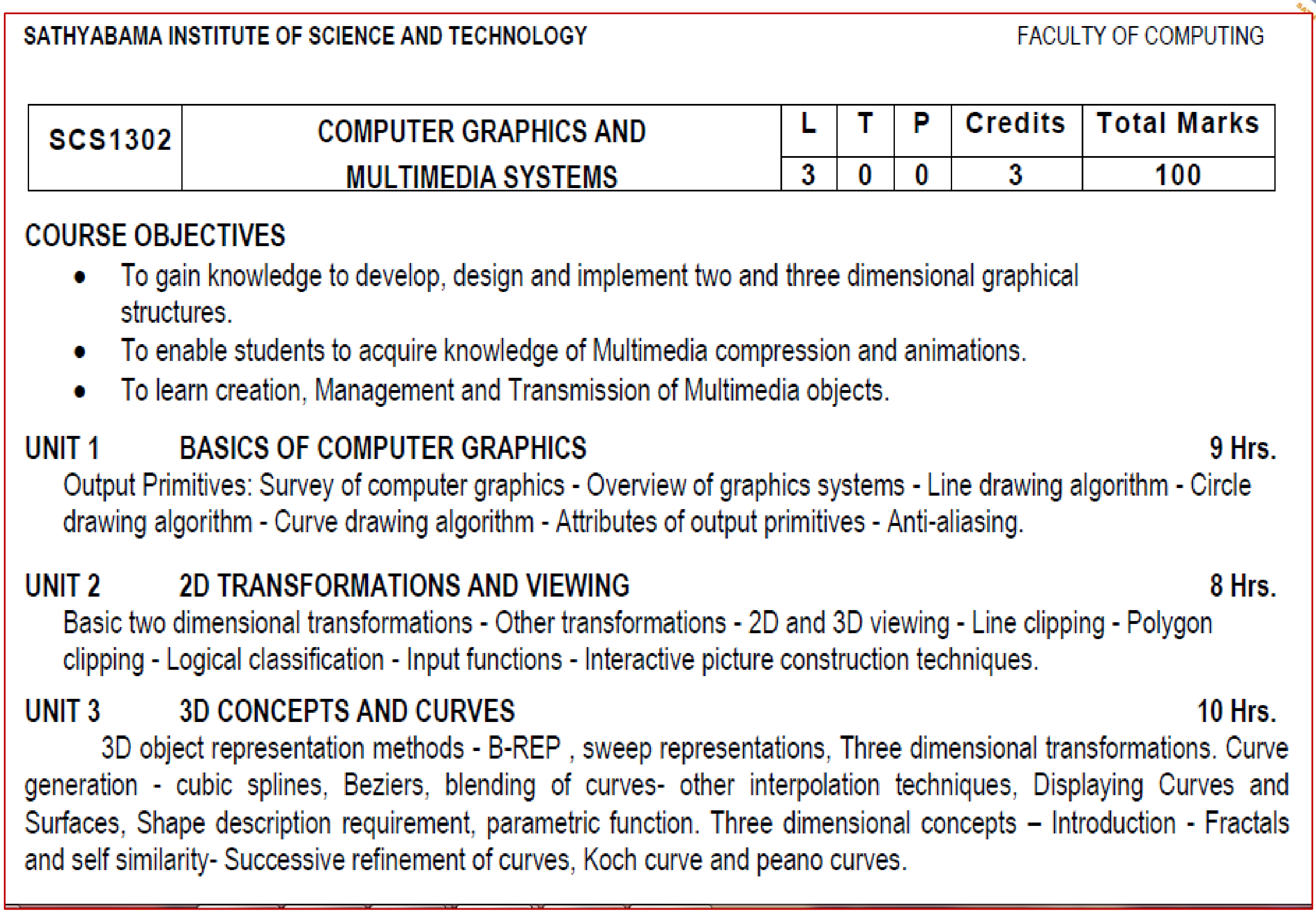
Computer Graphics and Multimedia Systems SCS1302

Unit 1

Syllabus



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY FACULTY OF COMPUTING

		L	T	P	Credits	Total Marks
SCS1302	COMPUTER GRAPHICS AND MULTIMEDIA SYSTEMS	3	0	0	3	100

COURSE OBJECTIVES

- To gain knowledge to develop, design and implement two and three dimensional graphical structures.
- To enable students to acquire knowledge of Multimedia compression and animations.
- To learn creation, Management and Transmission of Multimedia objects.

UNIT 1 BASICS OF COMPUTER GRAPHICS 9 Hrs.

Output Primitives: Survey of computer graphics - Overview of graphics systems - Line drawing algorithm - Circle drawing algorithm - Curve drawing algorithm - Attributes of output primitives - Anti-aliasing.

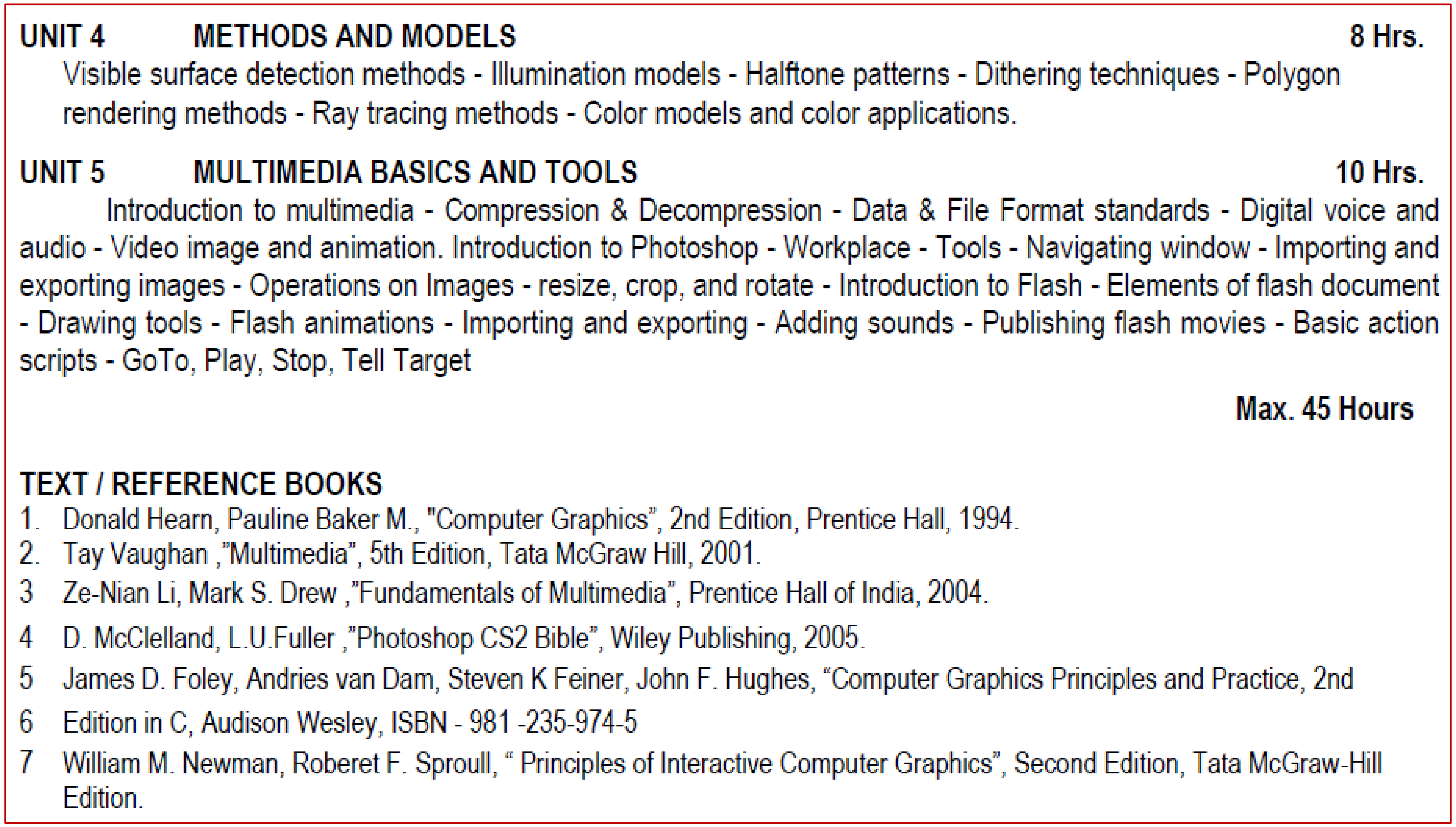
UNIT 2 2D TRANSFORMATIONS AND VIEWING 8 Hrs.

Basic two dimensional transformations - Other transformations - 2D and 3D viewing - Line clipping - Polygon clipping - Logical classification - Input functions - Interactive picture construction techniques.

UNIT 3 3D CONCEPTS AND CURVES 10 Hrs.

3D object representation methods - B-REP , sweep representations, Three dimensional transformations. Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques, Displaying Curves and Surfaces, Shape description requirement, parametric function. Three dimensional concepts – Introduction - Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves.

Syllabus



UNIT 4 METHODS AND MODELS **8 Hrs.**

Visible surface detection methods - Illumination models - Halftone patterns - Dithering techniques - Polygon rendering methods - Ray tracing methods - Color models and color applications.

UNIT 5 MULTIMEDIA BASICS AND TOOLS **10 Hrs.**

Introduction to multimedia - Compression & Decompression - Data & File Format standards - Digital voice and audio - Video image and animation. Introduction to Photoshop - Workplace - Tools - Navigating window - Importing and exporting images - Operations on Images - resize, crop, and rotate - Introduction to Flash - Elements of flash document - Drawing tools - Flash animations - Importing and exporting - Adding sounds - Publishing flash movies - Basic action scripts - GoTo, Play, Stop, Tell Target

Max. 45 Hours

TEXT / REFERENCE BOOKS

1. Donald Hearn, Pauline Baker M., "Computer Graphics", 2nd Edition, Prentice Hall, 1994.
2. Tay Vaughan ,"Multimedia", 5th Edition, Tata McGraw Hill, 2001.
3 Ze-Nian Li, Mark S. Drew ,"Fundamentals of Multimedia", Prentice Hall of India, 2004.
4 D. McClelland, L.U.Fuller ,"Photoshop CS2 Bible", Wiley Publishing, 2005.
5 James D. Foley, Andries van Dam, Steven K Feiner, John F. Hughes, "Computer Graphics Principles and Practice, 2nd
6 Edition in C, Audison Wesley, ISBN - 981 -235-974-5
7 William M. Newman, Roberet F. Sproull, " Principles of Interactive Computer Graphics", Second Edition, Tata McGraw-Hill Edition.

Course Objective(CO)

CO1: Construct lines and circles for the given input.

CO2: Apply 2D transformation techniques to transform the shapes to fit them as per the picture definition.

CO3: Construct splines, curves and perform 3D transformations

CO4: Apply colour and transformation techniques for various applications.

CO5: Analyse the fundamentals of animation, virtual reality, and underlying technologies.

CO6: Develop photo shop applications

Line Drawing Algorithms

- A line connects two points. It is a basic element in graphics.
- Line drawing algorithms:
  - **DDA Algorithm(Digital Differential Analyzer)**
  - **Bresenham's Line drawing Algorithm**

- To draw a line, you need two points between which you can draw a line

$$\mathbf{y=mx+c}$$

Where, **m** is the slope , **c** is the y intercept

Digital Differential Analyzer (DDA) line drawing algorithm

- The DDA is a **scan conversion line algorithm** based on calculating either dy or dx.
- A line is sampled at **unit intervals** in one coordinate and corresponding integer values nearest the line path are determined for other coordinates.

Steps in DDA algorithm

- Get the input of two end points
- Calculate the **difference** between two end points.
- Based on the calculated difference, If **dx >dy**, then number of **steps = dx** else number of **steps =dy**
- Calculate the **increment in x** coordinate and **y coordinate.**
- Put the pixel by successfully **incrementing x and y** coordinates the drawing of the line.

DDA

- DDA Algorithm is the **simplest line drawing** algorithm.
- Given the starting and ending coordinates of a line, DDA Algorithm attempts to generate the points between the starting and ending coordinates.

Procedure

Given

Starting coordinates = (X_a, Y_a)

Ending coordinates = (X_b, Y_b)

The points generation using DDA Algorithm involves the following steps-

Step-01:

Calculate **dx,dy** and **m** from the given input.

These parameters are calculated as-

$$dx = X_b - X_a$$

$$dy = Y_b - Y_a$$

$$m = dy/dx$$

and

$$X = X_a$$

$$Y = Y_a$$

Step-02:

Find the number of steps or points in between the starting and ending coordinates.

```
if (ab (dx) > abs (dy))
Steps = abs (dx);
else
Steps = abs (dy);
```

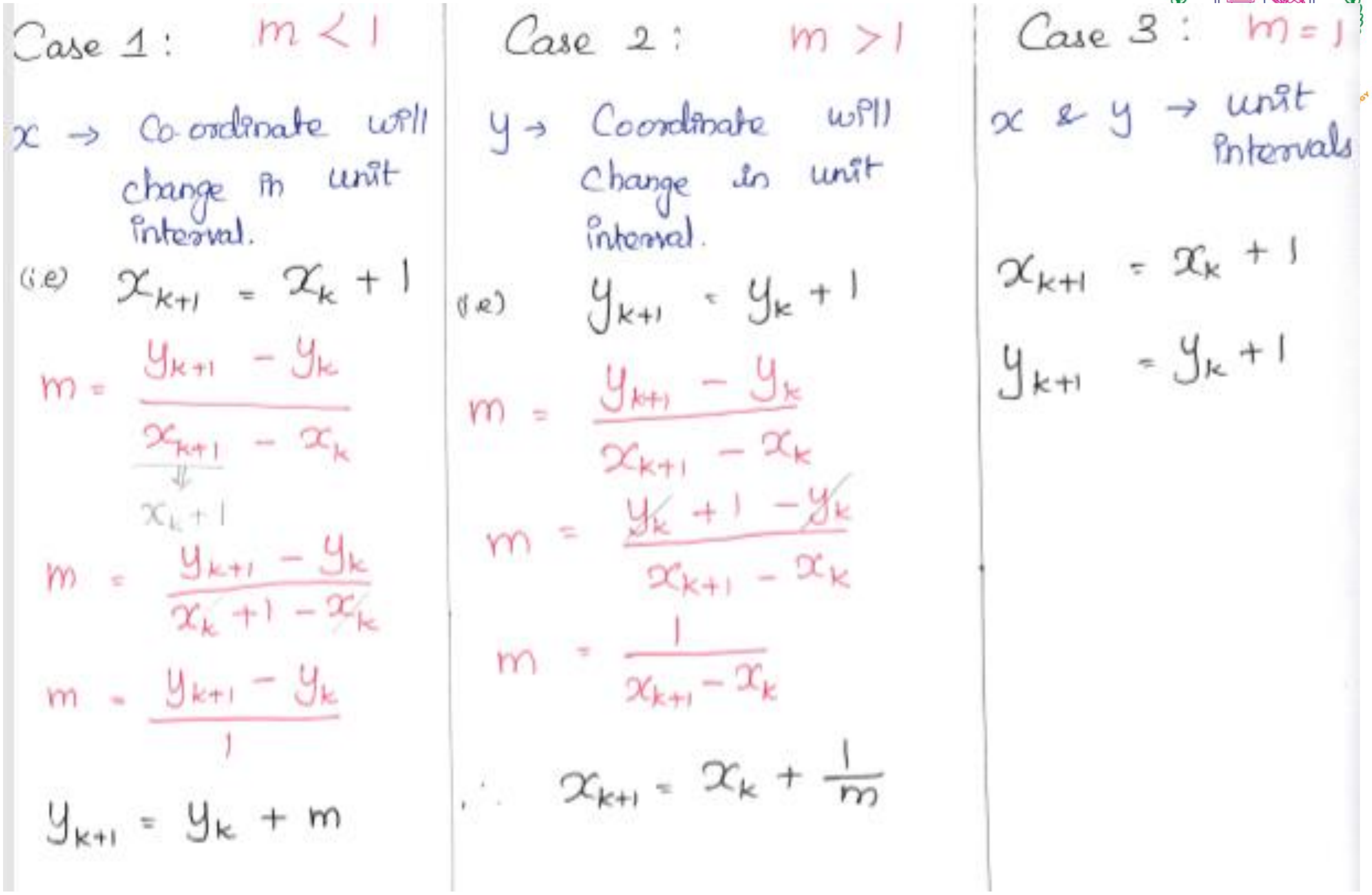
Step-03:

Find x increment and y increment,

$$X_{increment} = dx/steps$$

$$Y_{increment} = dy/steps$$

- **Step-04:**
- Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).
- Find the next point by following the below three cases-
- **Case 1 : m < 1**
- **Case 2: m > 1**
- **Case 3: m = 1**



Case 1 : $m < 1$

$x \rightarrow$ Co-ordinate will change in unit interval.

(i.e) $x_{k+1} = x_k + 1$

$$m = \frac{y_{k+1} - y_k}{\underset{\underset{x_k + 1}{\Downarrow}}{x_{k+1}} - x_k}$$

$$m = \frac{y_{k+1} - y_k}{\cancel{x_k} + 1 - \cancel{x_k}}$$

$$m = \frac{y_{k+1} - y_k}{1}$$

$$y_{k+1} = y_k + m$$

Case 2 : $m > 1$

$y \rightarrow$ Coordinate will change in unit interval.

(i.e) $y_{k+1} = y_k + 1$

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

$$m = \frac{\cancel{y_k} + 1 - \cancel{y_k}}{x_{k+1} - x_k}$$

$$m = \frac{1}{x_{k+1} - x_k}$$

$$\therefore \; x_{k+1} = x_k + \frac{1}{m}$$

Case 3 : $m = 1$

x & $y \rightarrow$ unit intervals

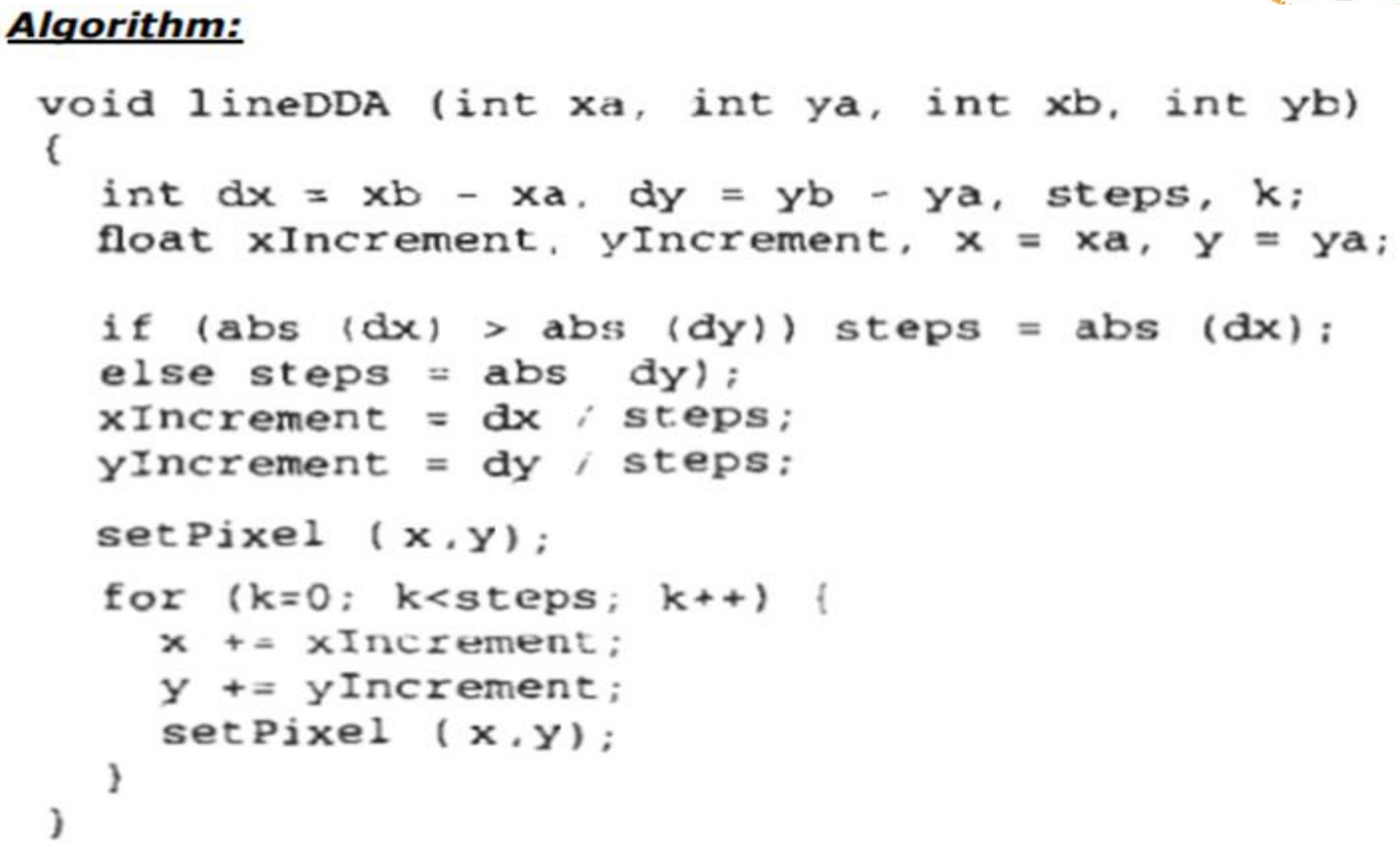
$$x_{k+1} = x_k + 1$$

$$y_{k+1} = y_k + 1$$

Step-05:

- Keep repeating Step-04 until the end point is reached or the number of generated new points (including the starting and ending points) equals to the steps count.

DDA algorithm



Algorithm:

```
void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs  dy);
  xIncrement = dx / steps;
  yIncrement = dy / steps;

  setPixel (x,y);

  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (x,y);
  }
}
```

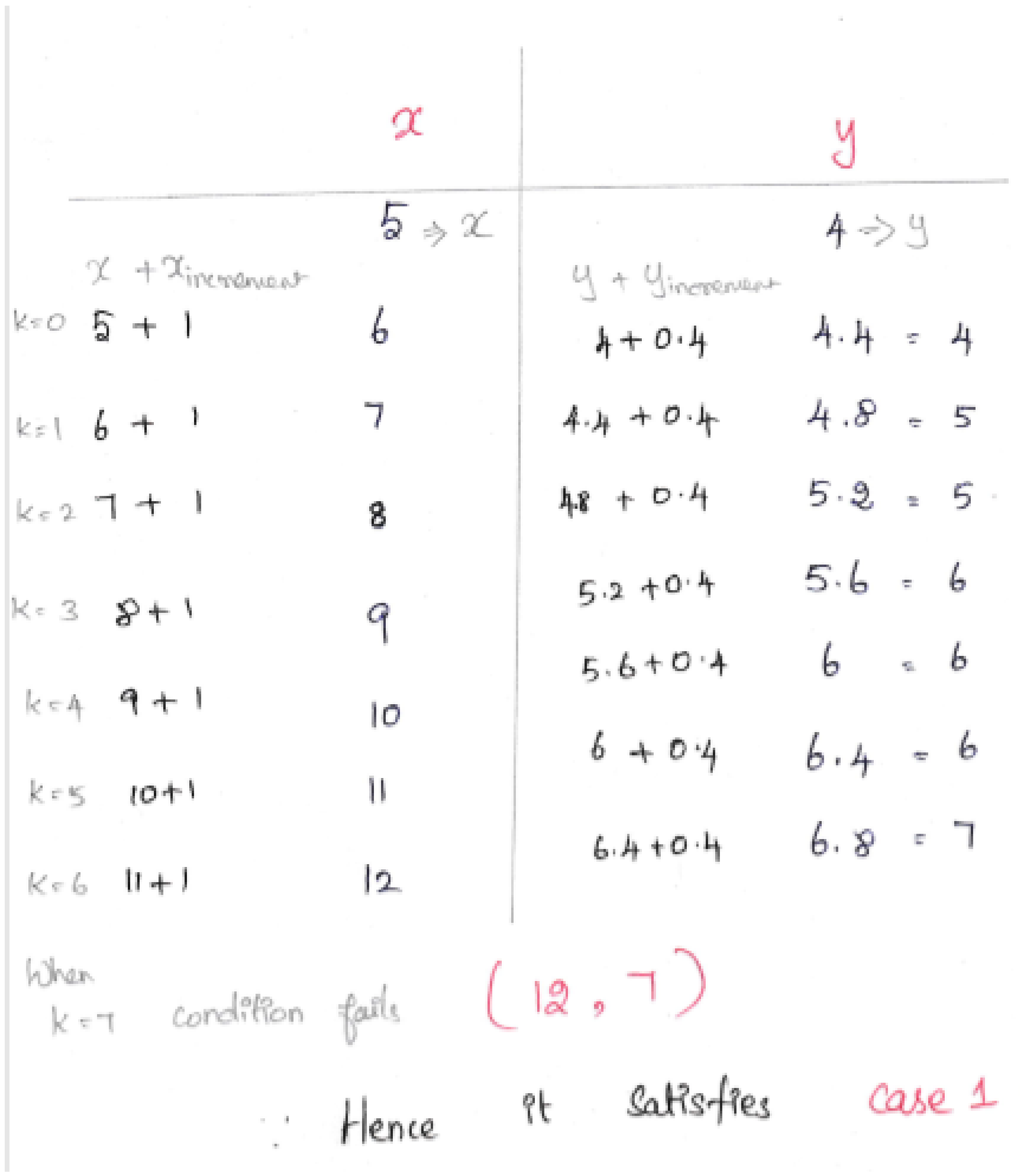
Example 1 :- Case 1 : $\boxed{m < 1}$

Given $(x_a , y_a) = (5 . 4)$ and

$(x_b , y_b) = (12, 7)$.

```
① dx = $x_b - x_a$
   dy = $y_b - y_a$
   x = $x_a$ ; y = $y_a$
② if abs(dx) > abs(dy)
      steps = abs(dx)
   else
      step = abs(dy)
③ $x_{increment} = \frac{dx}{steps}$
   $y_{increment} = \frac{dy}{steps}$
④ setpixel(x, y)
⑤ for(k=0 ; k < steps ; k++)
   {
      x = x + $x_{increment}$
      y = y + $y_{increment}$
      setpixel (x, y)
   }
```

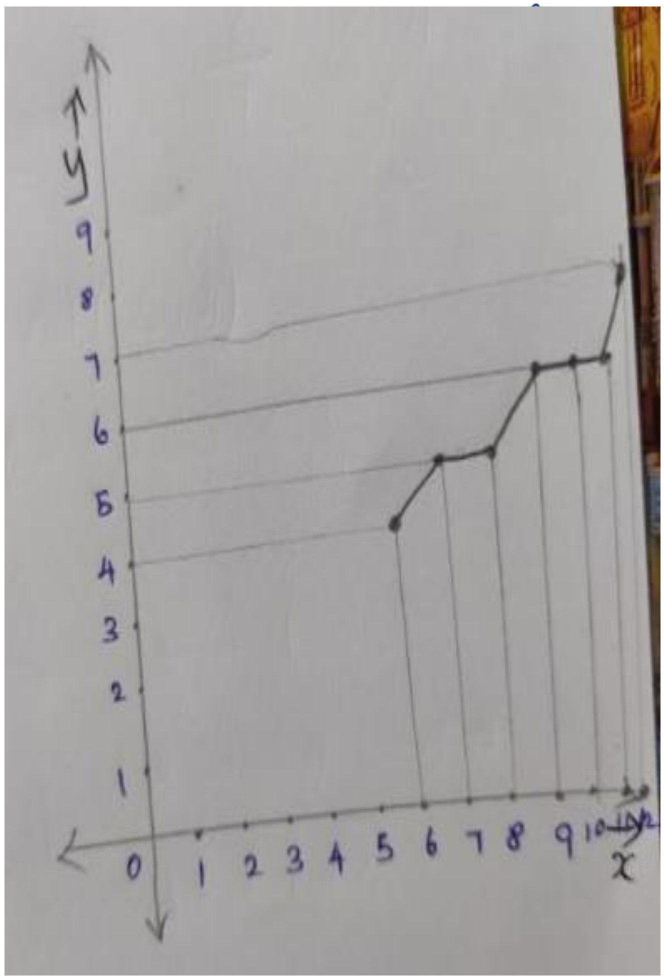
```
① dx = 12 - 5 = 7
   dy = 7 - 4 = 3
   $\boxed{x = 5 ; y = 4}$
② if (7 > 3)
      steps = 7
③ $x_{increment} = \frac{7}{7} = 1$
   $y_{increment} = \frac{3}{7} = 0.4$
④ setpixel (5, 4)
⑤ for ( k = 0 ; k < 7 ; k++)
   {
      x = 5 + 1 → 6
      y = 4 + 0.4 ⇒ 4.4
      setpixel ( 6 , 4)
```



		x		y
		$5 \Rightarrow x$		$4 \to y$
	$x + x_{increment}$		$y + y_{increment}$	
k=0	5 + 1	6	4 + 0.4	4.4 = 4
k=1	6 + 1	7	4.4 + 0.4	4.8 = 5
k=2	7 + 1	8	4.8 + 0.4	5.2 = 5
k=3	8 + 1	9	5.2 + 0.4	5.6 = 6
k=4	9 + 1	10	5.6 + 0.4	6 = 6
k=5	10 + 1	11	6 + 0.4	6.4 = 6
k=6	11 + 1	12	6.4 + 0.4	6.8 = 7

When k=7 condition fails (12, 7)

∴ Hence it satisfies case 1 $(m < 1)$



Example 2

Calculate the points between the starting point (5, 6) and ending point (8, 12).

Given-

- Starting coordinates = (X_0, Y_0) = (5, 6)
- Ending coordinates = (X_n, Y_n) = (8, 12)

Step-01:

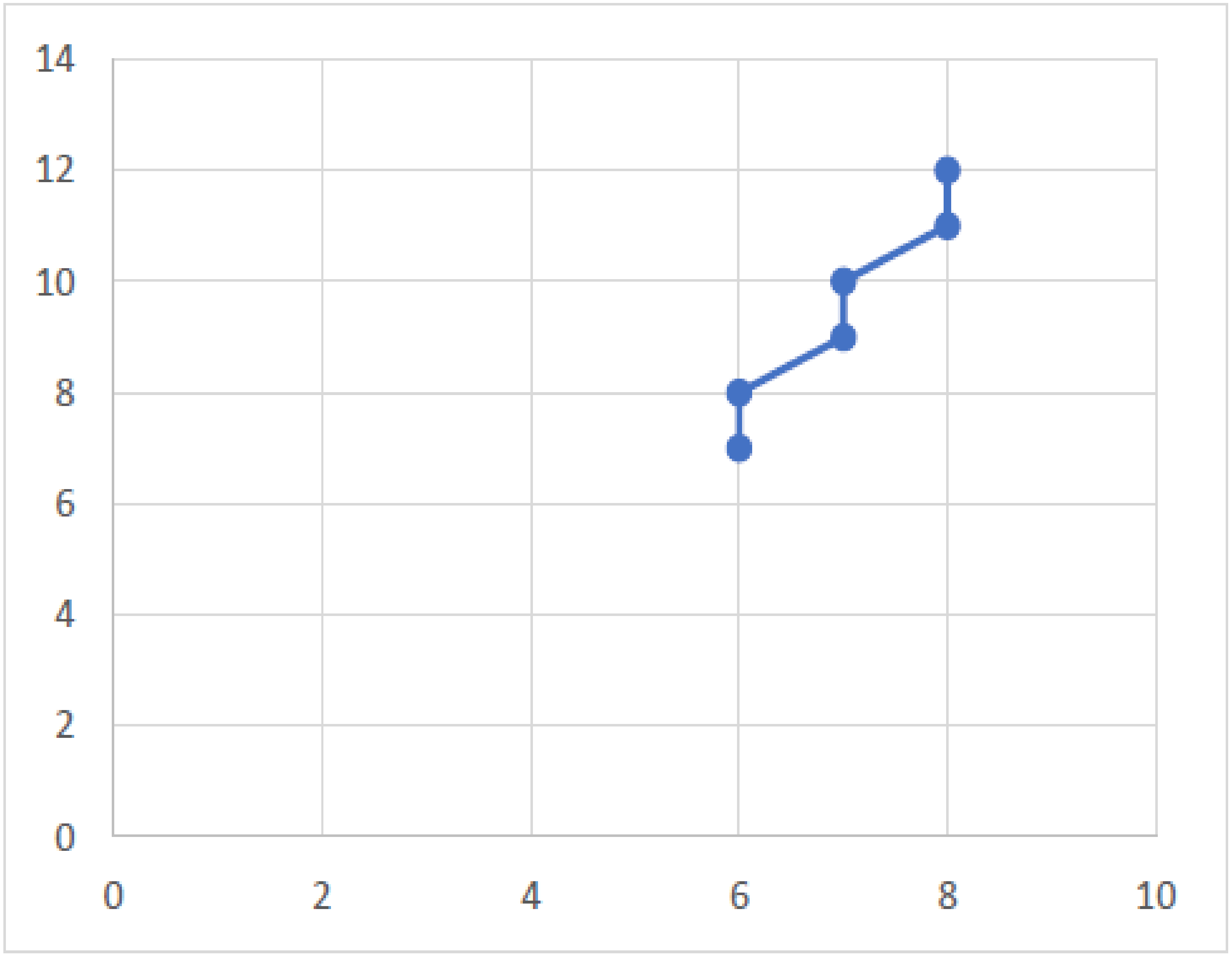
- Calculate ΔX, ΔY and M from the given input.
- $\Delta X = X_n - X_0 = 8 - 5 = 3$
- $\Delta Y = Y_n - Y_0 = 12 - 6 = 6$
- $M = \Delta Y / \Delta X = 6 / 3 = 2$

Step-02:

- Calculate the number of steps.
- As $|\Delta X| < |\Delta Y| = 3 < 6$, so number of steps
- $\Delta Y = 6$
- **Step-03:**
- As $M > 1$, so case-03 is satisfied.

Now, Step-03 is executed until Step-04 is satisfied.

X_p	Y_p	X_{p+1}	Y_{p+1}	Round off (X_{p+1}, Y_{p+1})
5	6	5.5	7	(6, 7)
		6	8	(6, 8)
		6.5	9	(7, 9)
		7	10	(7, 10)
		7.5	11	(8, 11)
		8	12	(8, 12)


14
12
10
8
6
4
2
0
0
2
4
6
8
10

Example 3

Calculate the points between the starting point (5, 6) and ending point (13, 10).

Given-

- Starting coordinates = (X_0, Y_0) = (5, 6)
- Ending coordinates = (X_n, Y_n) = (13, 10)

Step-01:

- Calculate ΔX, ΔY and M from the given input.
- $\Delta X = X_n - X_0 = 13 - 5 = 8$
- $\Delta Y = Y_n - Y_0 = 10 - 6 = 4$
- $M = \Delta Y / \Delta X = 4 / 8 = 0.50$

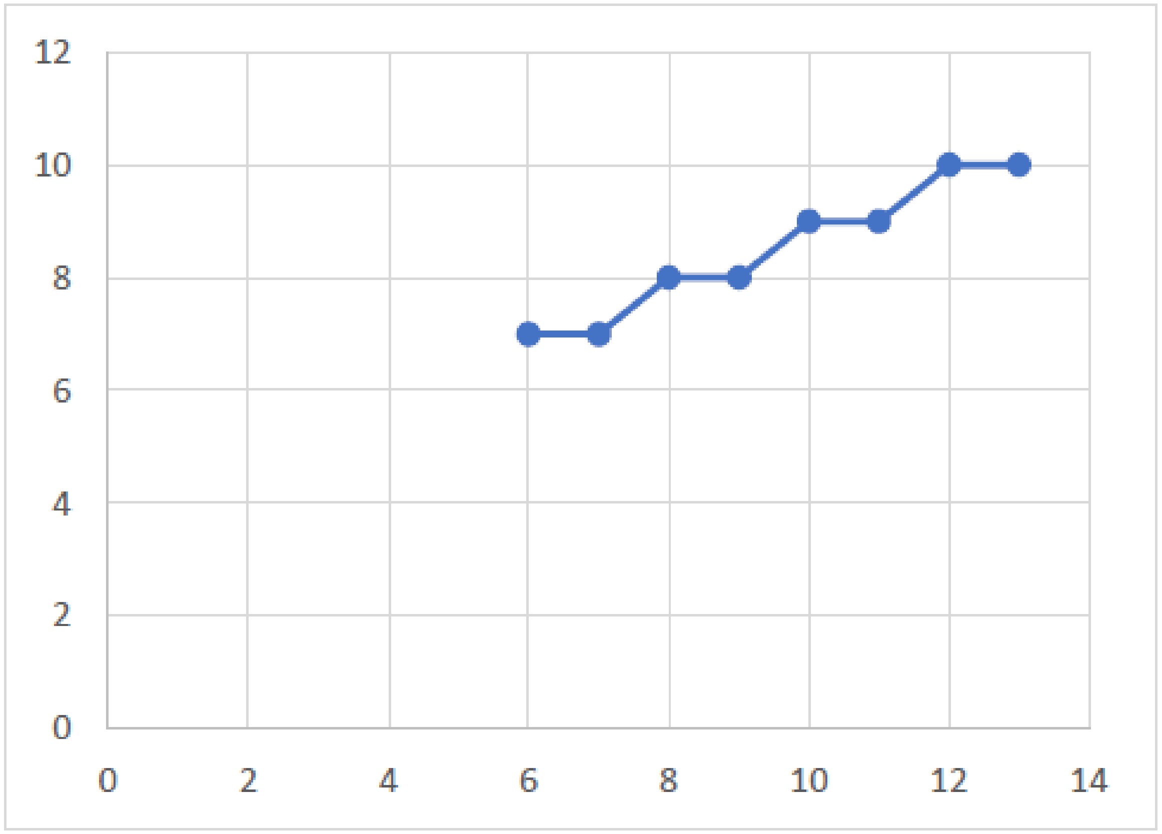
Step-02:

- Calculate the number of steps.
- As $|\Delta X| > |\Delta Y| = 8 > 4$, so number of steps = $\Delta X = 8$

Step-03:

- As $M < 1$, so case-01 is satisfied.
- Now, Step-03 is executed until Step-04 is satisfied.

X_p	Y_p	X_{p+1}	Y_{p+1}	Round off (X_{p+1}, Y_{p+1})
5	6	6	6.5	(6, 7)
		7	7	(7, 7)
		8	7.5	(8, 8)
		9	8	(9, 8)
		10	8.5	(10, 9)
		11	9	(11, 9)
		12	9.5	(12, 10)
		13	10	(13, 10)


12
10
8
6
4
2
0
0
2
4
6
8
10
12
14

Example 4

Calculate the points between the starting point (1, 7) and ending point (11, 17).

Solution-

Given-

- Starting coordinates = $(X_0, Y_0) = (1, 7)$
- Ending coordinates = $(X_n, Y_n) = (11, 17)$

Step-01:

- Calculate ΔX, ΔY and M from the given input.
- $\Delta X = X_n - X_0 = 11 - 1 = 10$
- $\Delta Y = Y_n - Y_0 = 17 - 7 = 10$
- $M = \Delta Y / \Delta X = 10 / 10 = 1$

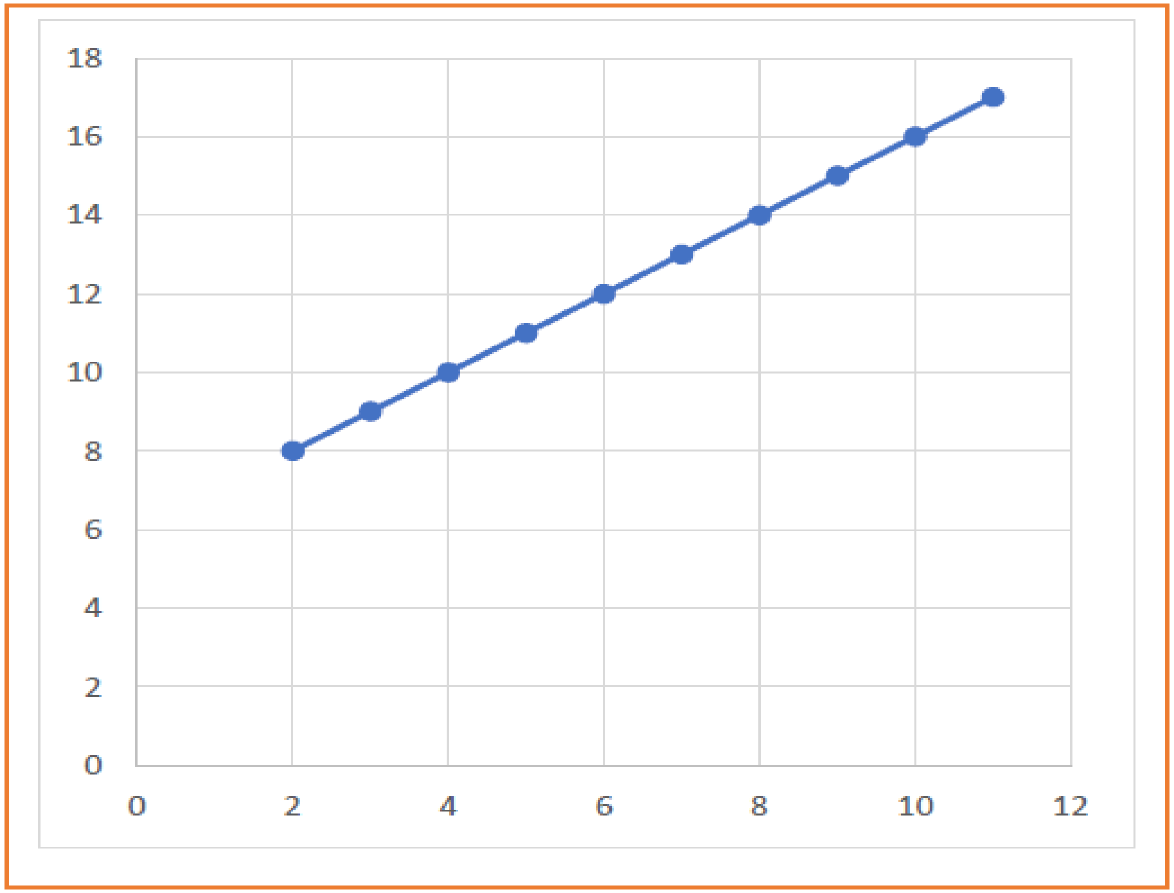
Step-02:

- Calculate the number of steps.
- As $|\Delta X| = |\Delta Y| = 10 = 10$, so number of steps = $\Delta X = \Delta Y = 10$

Step-03:

- As M = 1, so case-02 is satisfied.
- Now, Step-03 is executed until Step-04 is satisfied.

X_p	Y_p	X_{p+1}	Y_{p+1}	Round off (X_{p+1}, Y_{p+1})
1	7	2	8	(2, 8)
		3	9	(3, 9)
		4	10	(4, 10)
		5	11	(5, 11)
		6	12	(6, 12)
		7	13	(7, 13)
		8	14	(8, 14)
		9	15	(9, 15)
		10	16	(10, 16)
		11	17	(11, 17)


18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12

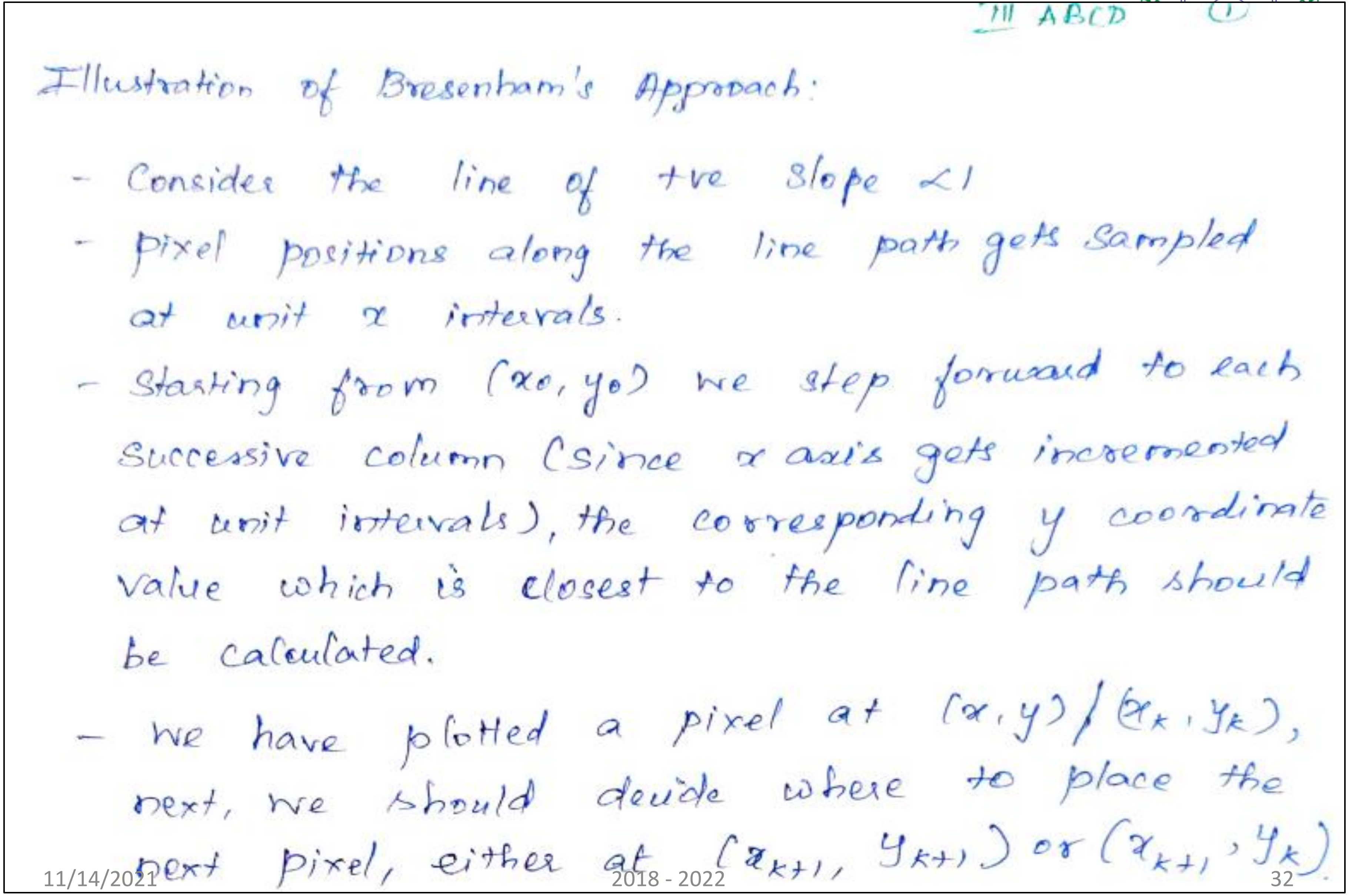
SIST
JUSTICE PEACE REVOLUTION
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEEMED UNIVERSITY

Bresenham's Line Drawing Algorithm

Bresenhams Line Drawing Algorithm

- This algorithm is used for scan converting a line.
- It was developed by Bresenham.
- It is an efficient method because it involves only integer addition, subtractions, and multiplication operations.
- These operations can be performed very rapidly so lines can be generated quickly.

Bresenhams Line Drawing Algorithm

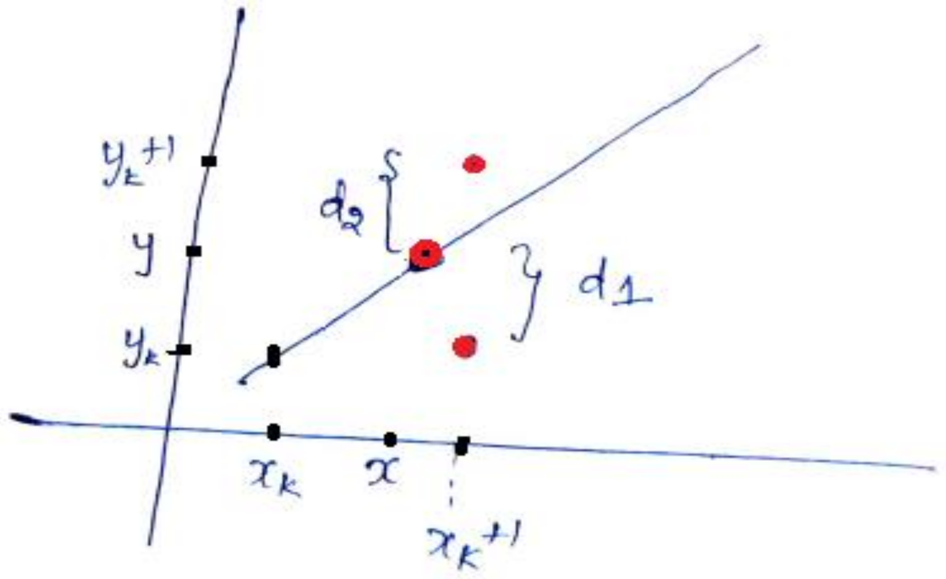


III ABCD ①

Illustration of Bresenham's Approach:

- Consider the line of +ve slope <1
- pixel positions along the line path gets sampled at unit x intervals.
- Starting from (x_0, y_0) we step forward to each successive column (since x axis gets incremented at unit intervals), the corresponding y coordinate value which is closest to the line path should be calculated.
- we have plotted a pixel at (x, y) / (x_k, y_k), next, we should decide where to place the next pixel, either at (x_{k+1}, y_{k+1}) or (x_{k+1}, y_k).

Bresenhams Line Drawing Algorithm



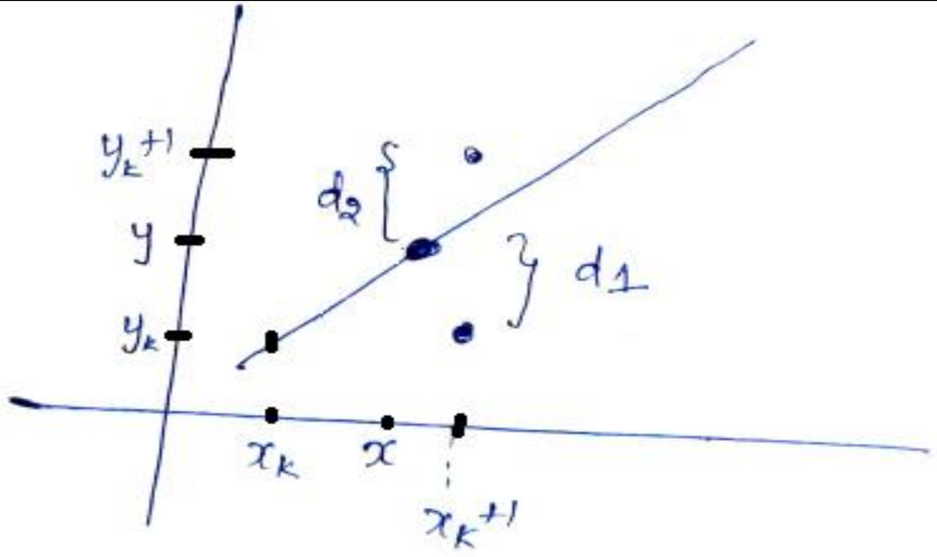
- The y coordinate of the mathematical line at pixel position x_{k+1} is given by,

$$y = (x_k+1)m + b$$

$$d1 = y - y_k = (x_k+1)m + b - y_k \longrightarrow ①$$

$$d2 = y_k+1 - y = y_k+1 - (x_k+1)m - b \longrightarrow ②$$

Bresenhams Line Drawing Algorithm



- The y coordinate of the mathematical line at pixel position x_{k+1} is given by,

$$y = (x_k+1)m + b$$

$$d1 = y - y_k = (x_k+1)m + b - y_k \longrightarrow \textcircled{1}$$

$$d2 = y_k+1 - y = y_k+1 - (x_k+1)m - b \longrightarrow \textcircled{2}$$

Bresenhams Line Drawing Algorithm

2

Perform d1 - d2, ① - ②,

$$d1 - d2 = (x_k + 1)m + b - y_k - \left[y_k + 1 - (x_k + 1)m - b \right]$$

$$= (x_k + 1)m + b - y_k - y_k - 1 + (x_k + 1)m + b$$

$$= 2(x_k + 1)m + 2b - 2y_k - 1$$

$$\boxed{\therefore d1 - d2 = 2(x_k + 1)m + 2b - 2y_k - 1} \longrightarrow ③$$

Substitute $m = \dfrac{\Delta y}{\Delta x}$ in ③

Bresenhams Line Drawing Algorithm

Substitute $m = \frac{\Delta y}{\Delta x}$ in ③

$$d1 - d2 = 2\frac{\Delta y}{\Delta x}(x_k + 1) + 2b - 2y_k - 1$$

$$\Delta x(d1 - d2) = 2\Delta y(x_k + 1) + 2b\Delta x - 2y_k\Delta x - \Delta x$$

$$= 2\Delta y x_k + 2\Delta y + 2b\Delta x - 2y_k\Delta x - \Delta x$$

$$= 2\Delta y x_k + 2b\Delta x - \Delta x - 2y_k\Delta x + 2\Delta y$$

$$= 2\Delta y x_k + \Delta x(2b - 1) - 2\Delta x y_k + 2\Delta y$$

$$= 2\Delta y x_k - 2\Delta x y_k + \underbrace{2\Delta y + \Delta x(2b - 1)}_{c}$$

$$= 2\Delta y x_k - 2\Delta x y_k + c$$

$$\boxed{c = 2\Delta y + \Delta x(2b - 1)}$$

Bresenhams Line Drawing Algorithm

$$\boxed{C = 2\Delta y + \Delta x(2b-1)}$$

$$\therefore \underbrace{\Delta x(d1-d2)} = 2\Delta y x_k - 2\Delta x y_k + C$$

P_k (decision Parameter)

$$P_k = \Delta x(d1-d2)$$

Hence, $\boxed{P_k = 2\Delta y x_k - 2\Delta x y_k + C} \longrightarrow (4)$

- If y_k is closer, then plot it at (x_k+1, y_k) else plot it at (x_k+1, y_k+1)

Bresenhams Line Drawing Algorithm

At step k+1, the decision Parameter is evaluated as,

From ④,

$$P_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c \longrightarrow ⑤$$

Now, ⑤ - ④,

$$P_{k+1} - P_k = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} \cancel{+c}$$
$$- 2\Delta y\, x_k + 2\Delta x\, y_k \cancel{-c}$$

$$P_{k+1} - P_k = 2\Delta y\,(x_{k+1} - x_k) - 2\Delta x\,(y_{k+1} - y_k)$$

$$P_{k+1} = P_k + 2\Delta y\,(x_{k+1} - x_k) - 2\Delta x\,(y_{k+1} - y_k)$$

Bresenhams Line Drawing Algorithm

$$P_{k+1} = P_k + 2\Delta y\,(x_{k+1} - x_k) - 2\Delta x\,(y_{k+1} - y_k)$$

But $x_{k+1} = x_k + 1$,

$$\therefore\ P_{k+1} = P_k + 2\Delta y\,(\cancel{x_k} + 1 - \cancel{x_k}) - 2\Delta x\,(y_{k+1} - y_k)$$

$$= P_k + 2\Delta y - 2\Delta x\,(\underbrace{y_{k+1} - y_k})$$

either 0 or 1 based on the sign of parameter P_k.

If $P_k < 0$, $P_{k+1} = P_k + 2\Delta y$

$P_k > 0$, $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_k < 0$, sub, $y_{k+1} = y_k$

$P_k > 0$, sub $y_{k+1} = y_k + 1$

If $P_k = +ve$, plot it at y_{k+1}

$P_k = -ve$, plot it at y_k

Bresenhams Line Drawing Algorithm

(7)

The first parameter P_0, with starting pixel position (x_0, y_0) will be Calculated from (4),

$$P_k = 2\Delta y x_k - 2\Delta x y_k + c$$

put, $k=0$,

$$P_0 = 2\Delta y x_0 - 2\Delta x y_0 + c$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x(2b-1)$$

$$= 2\Delta y x_0 + 2\Delta y - 2\Delta x y_0 + 2b\Delta x - \Delta x$$

Substitute, y $b = y_0 - mx_0$

$$= 2\Delta y x_0 + 2\Delta y - 2\Delta x y_0 - \Delta x + 2(y_0 - mx_0)\Delta x$$

$$= 2\Delta y x_0 + 2\Delta y - \cancel{2\Delta x y_0} - \Delta x + \cancel{2\Delta x y_0} - 2\Delta x m x_0$$

$$= 2\Delta y x_0 + 2\Delta y - 2\Delta x m x_0 - \Delta x$$

Bresenhams Line Drawing Algorithm

$$= 2\Delta y x_0 + 2\Delta y - \cancel{2\Delta x y_0} - \Delta x + \cancel{2\Delta x y_0} - 2\Delta x m x_0$$

$$= 2\Delta y x_0 + 2\Delta y - 2\Delta x m x_0 - \Delta x$$

Substitute, $m = \frac{\Delta y}{\Delta x}$,

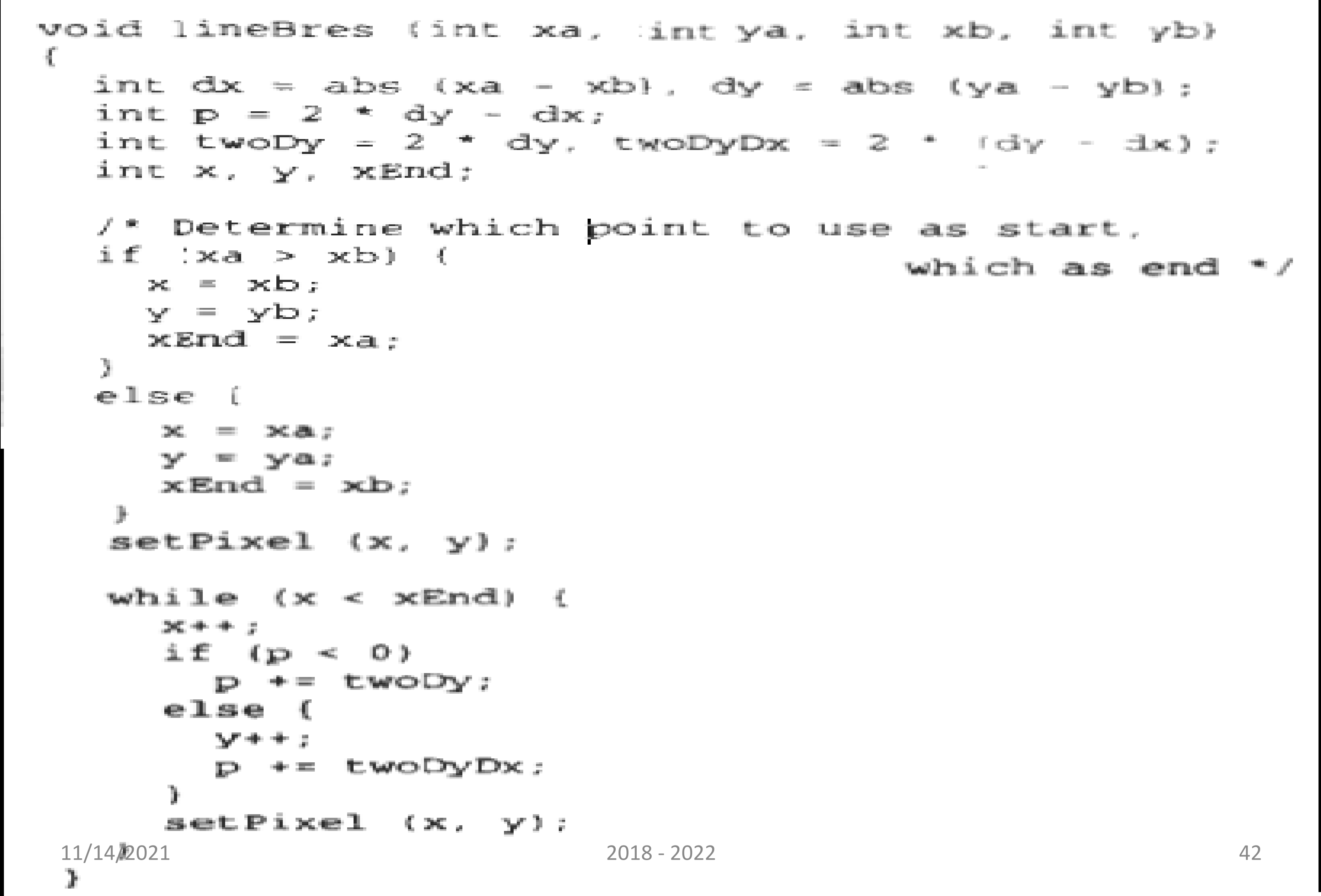
$$= 2\Delta y x_0 + 2\Delta y - 2\cancel{\Delta x}\frac{\Delta y}{\cancel{\Delta x}} x_0 - \Delta x$$

$$= \cancel{2\Delta y x_0} + 2\Delta y - \cancel{2\Delta y x_0} - \Delta x$$

$$= 2\Delta y - \Delta x$$

$$\therefore \boxed{P_0 = 2\Delta y - \Delta x}$$

Bresenhams Line Drawing Algorithm

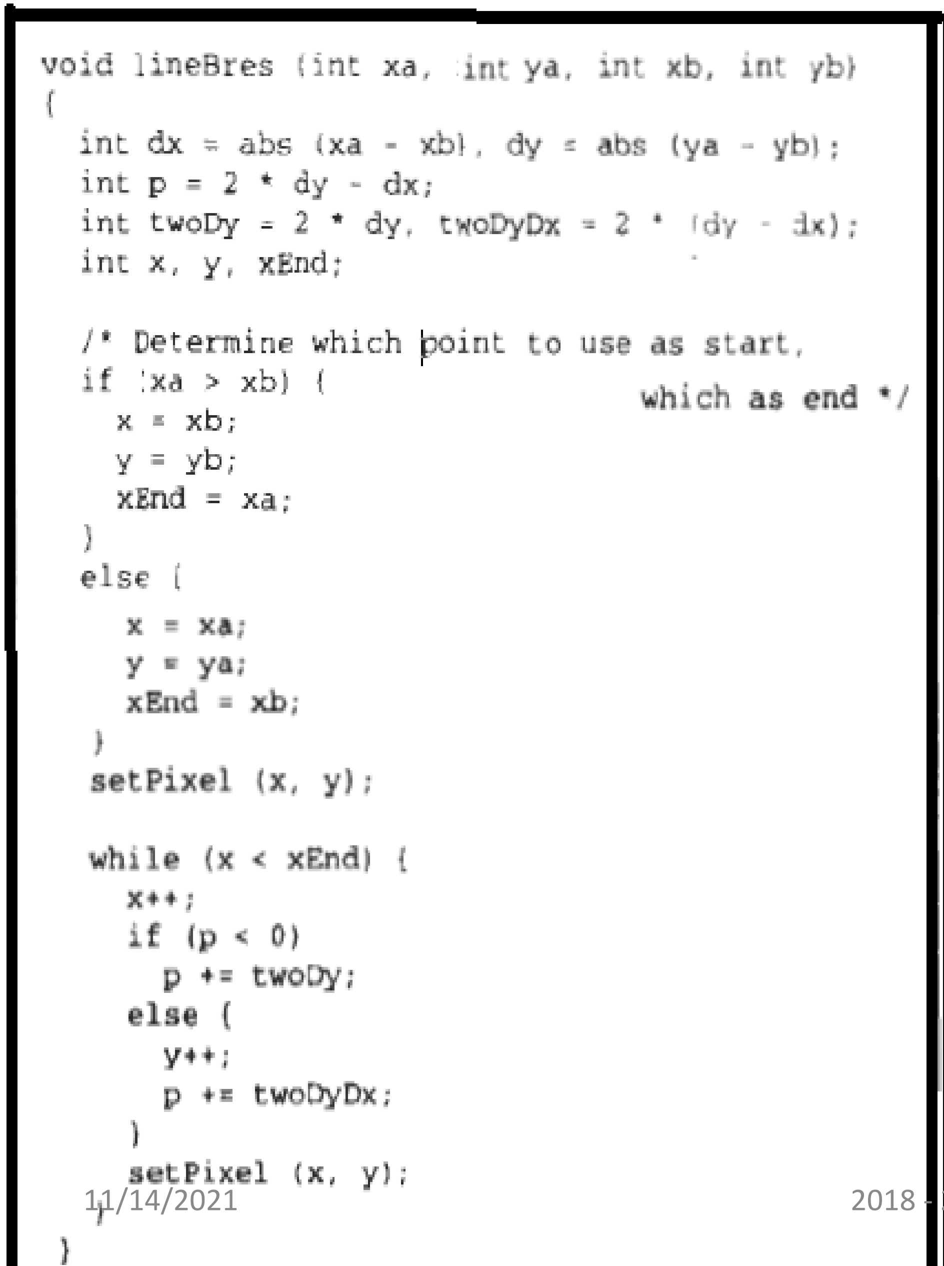


```
void lineBres (int xa, int ya, int xb, int yb)
{
   int dx = abs (xa - xb), dy = abs (ya - yb);
   int p = 2 * dy - dx;
   int twoDy = 2 * dy, twoDyDx = 2 * (dy - dx);
   int x, y, xEnd;

   /* Determine which point to use as start, which as end */
   if (xa > xb) {
      x = xb;
      y = yb;
      xEnd = xa;
   }
   else {
      x = xa;
      y = ya;
      xEnd = xb;
   }
   setPixel (x, y);

   while (x < xEnd) {
      x++;
      if (p < 0)
         p += twoDy;
      else {
         y++;
         p += twoDyDx;
      }
      setPixel (x, y);
   }
}
```

Example- Bresenhams Line drawing



```
void lineBres (int xa, int ya, int xb, int yb)
{
  int dx = abs (xa - xb), dy = abs (ya - yb);
  int p = 2 * dy - dx;
  int twoDy = 2 * dy, twoDyDx = 2 * (dy - dx);
  int x, y, xEnd;

  /* Determine which point to use as start,
  if (xa > xb) {                      which as end */
    x = xb;
    y = yb;
    xEnd = xa;
  }
  else {
    x = xa;
    y = ya;
    xEnd = xb;
  }
  setPixel (x, y);

  while (x < xEnd) {
    x++;
    if (p < 0)
      p += twoDy;
    else {
      y++;
      p += twoDyDx;
    }
    setPixel (x, y);
  }
}
```

- Given (xa,ya)=(20,10) and (xb,yb)=(30,18)

Solution:

dx= 30-20 =10

dy= 18-10 = 8

P= 2*dy-dx

= 2*8-10

=6

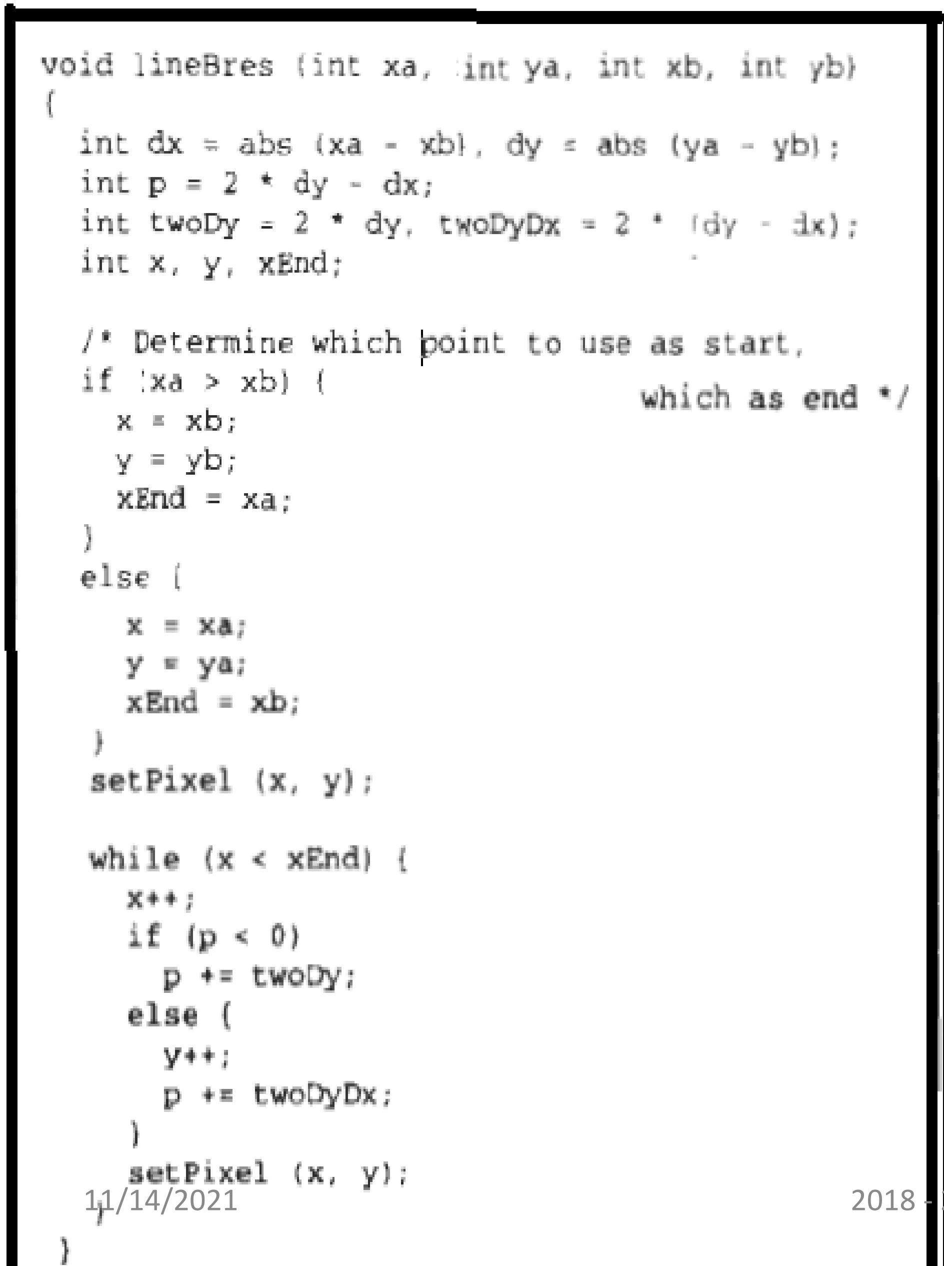
twody= 2*dy = 2*8 = 16

twodydx= 2*(dy-dx)

= 2*(8-10)

= -4

Example- Bresenhams Line drawing



```
void lineBres (int xa, int ya, int xb, int yb)
{
  int dx = abs (xa - xb), dy = abs (ya - yb);
  int p = 2 * dy - dx;
  int twoDy = 2 * dy, twoDyDx = 2 * (dy - dx);
  int x, y, xEnd;

  /* Determine which point to use as start,
  if (xa > xb) {                          which as end */
    x = xb;
    y = yb;
    xEnd = xa;
  }
  else {
    x = xa;
    y = ya;
    xEnd = xb;
  }
  setPixel (x, y);

  while (x < xEnd) {
    x++;
    if (p < 0)
      p += twoDy;
    else {
      y++;
      p += twoDyDx;
    }
    setPixel (x, y);
  }
}
```

If(20>30)

{ x=20; y=10; xend=30; }

setpixel(20,10);

while(20<30)

{ x=21;

If(6<0)

y=11;

p=6-4 = 2;

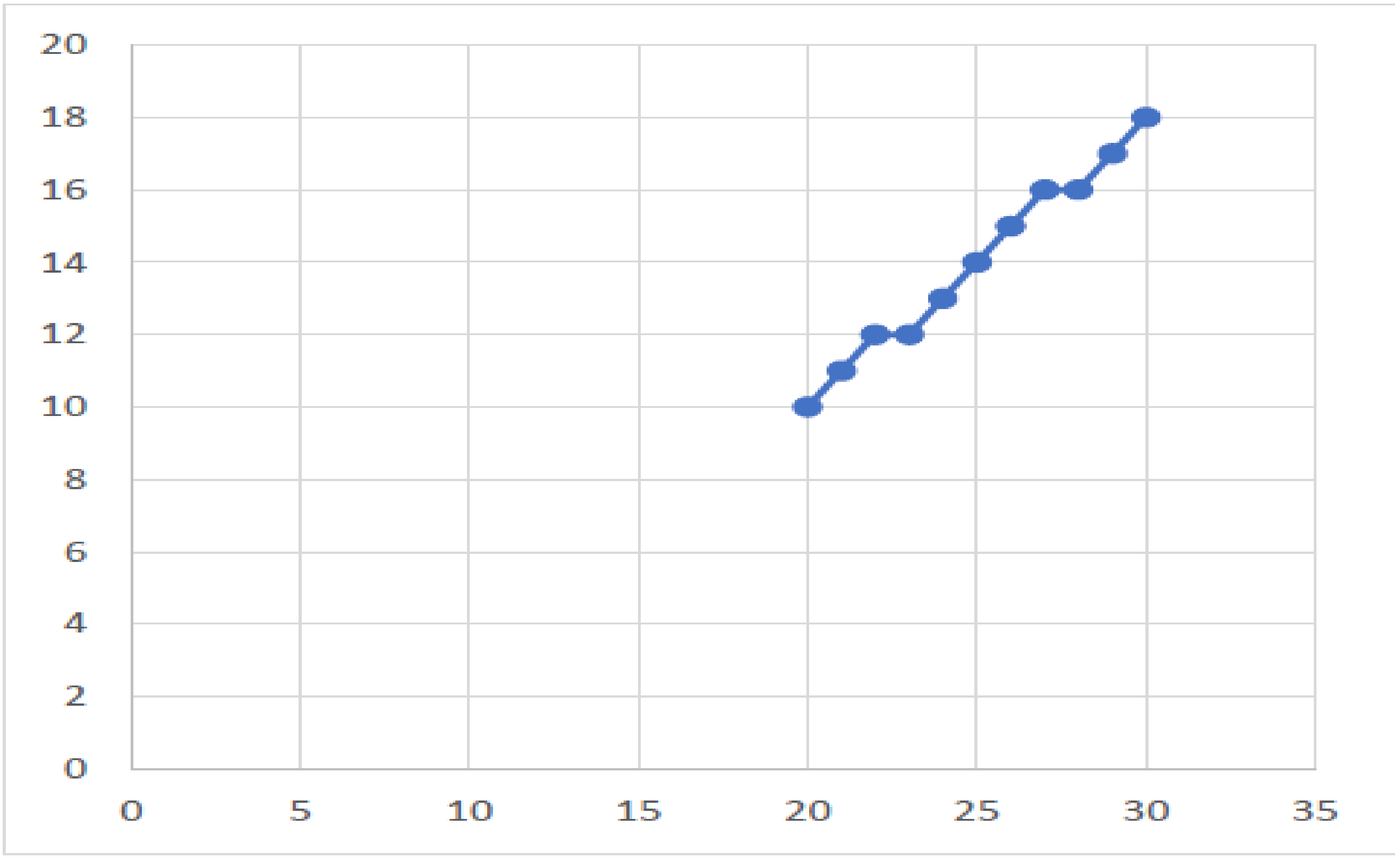
}

setpixel(21,11);

Example 1- Bresenhams Line drawing

k	P_k	(X_{k+1}, Y_{k+1})
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	10	(25,14)
5	6	(26,15)
6	2	(27,16)
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)

Example 1- Bresenhams Line drawing



Advantages of Bresenhams Line drawing algorithm

Disadvantage of DDA:

- The accumulation of round of error is successive addition of the floating point increments is used to find the pixel position but it take lot of time to compute the pixel position.

Advantages of bresenham's line drawing algorithm.

The Bresenham line algorithm has the following advantages:
– An fast incremental algorithm
– Uses only integer calculations

The Bresenham algorithm is another incremental scan conversion algorithm
The big advantage of this algorithm is that it uses only integer calculations such as addition/subtraction and bit shifting.
The main advantage of Bresenham's algorithm is speed.

The disadvantage of such a simple algorithm is that it is meant for basic line drawing. The "advanced" topic of antialiasing isn't part of Bresenham's algorithm, so to draw smooth lines, you'd want to look into a different algorithm.

Advantages of bresenham's line drawing algorithm.

The advantages of Bresenham Line Drawing Algorithm are-
- It is easy to implement.
- It is fast and incremental.
- It executes fast but less faster than DDA Algorithm.
- The points generated by this algorithm are more accurate than DDA Algorithm.
- It uses fixed points only.

Disadvantages of Bresenham Line Drawing Algorithm-

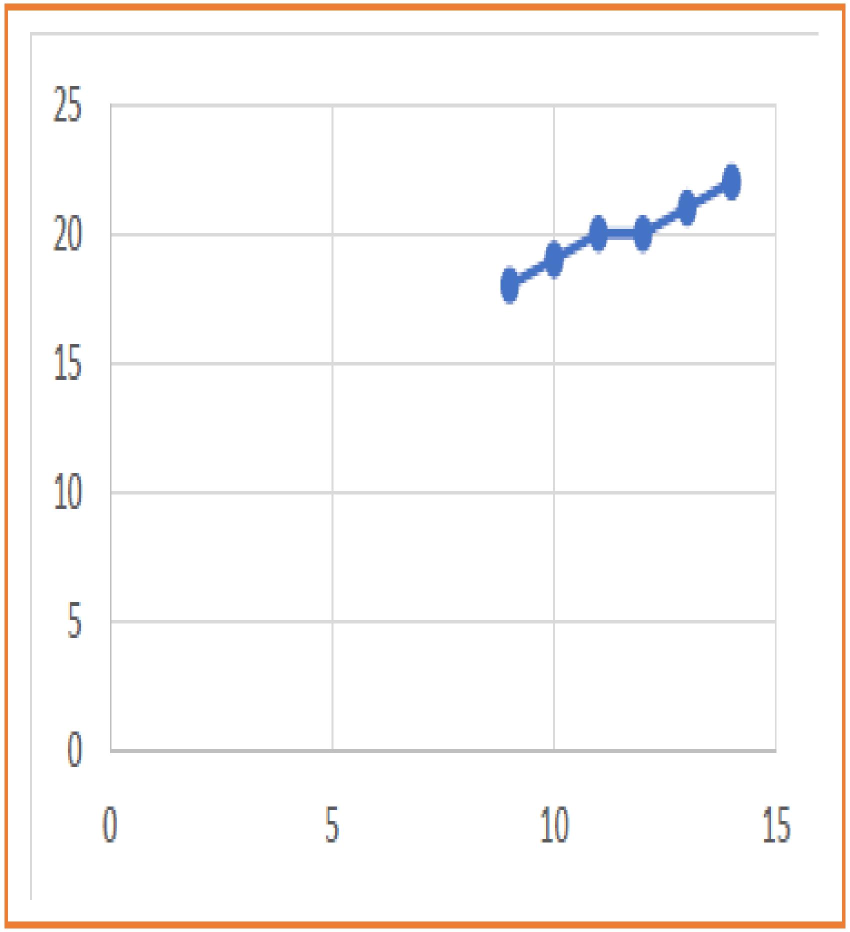
The disadvantages of Bresenham Line Drawing Algorithm are-
- Though it improves the accuracy of generated points but still the resulted line is not smooth.
- This algorithm is for the basic line drawing.
- It can not handle diminishing jaggies.

Example 2

- Calculate the points between the starting coordinates (9, 18) and ending coordinates (14, 22).

Example 2-cont…

P_k	P_{k+1}	X_{k+1}	Y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22



- https://developerinsider.co/download-turbo-c-for-windows-7-8-8-1-and-windows-10-32-64-bit-full-screen/
- **developerinsider.in/turbocpp**