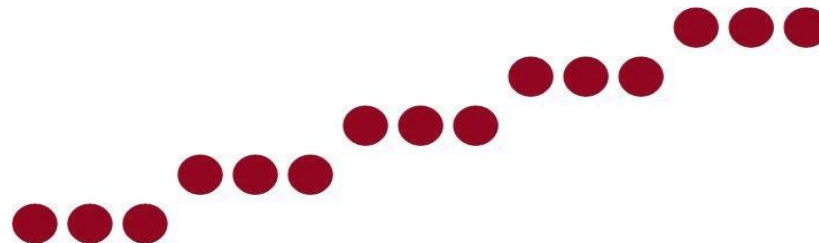


Line Drawing Algorithm

Unit 2

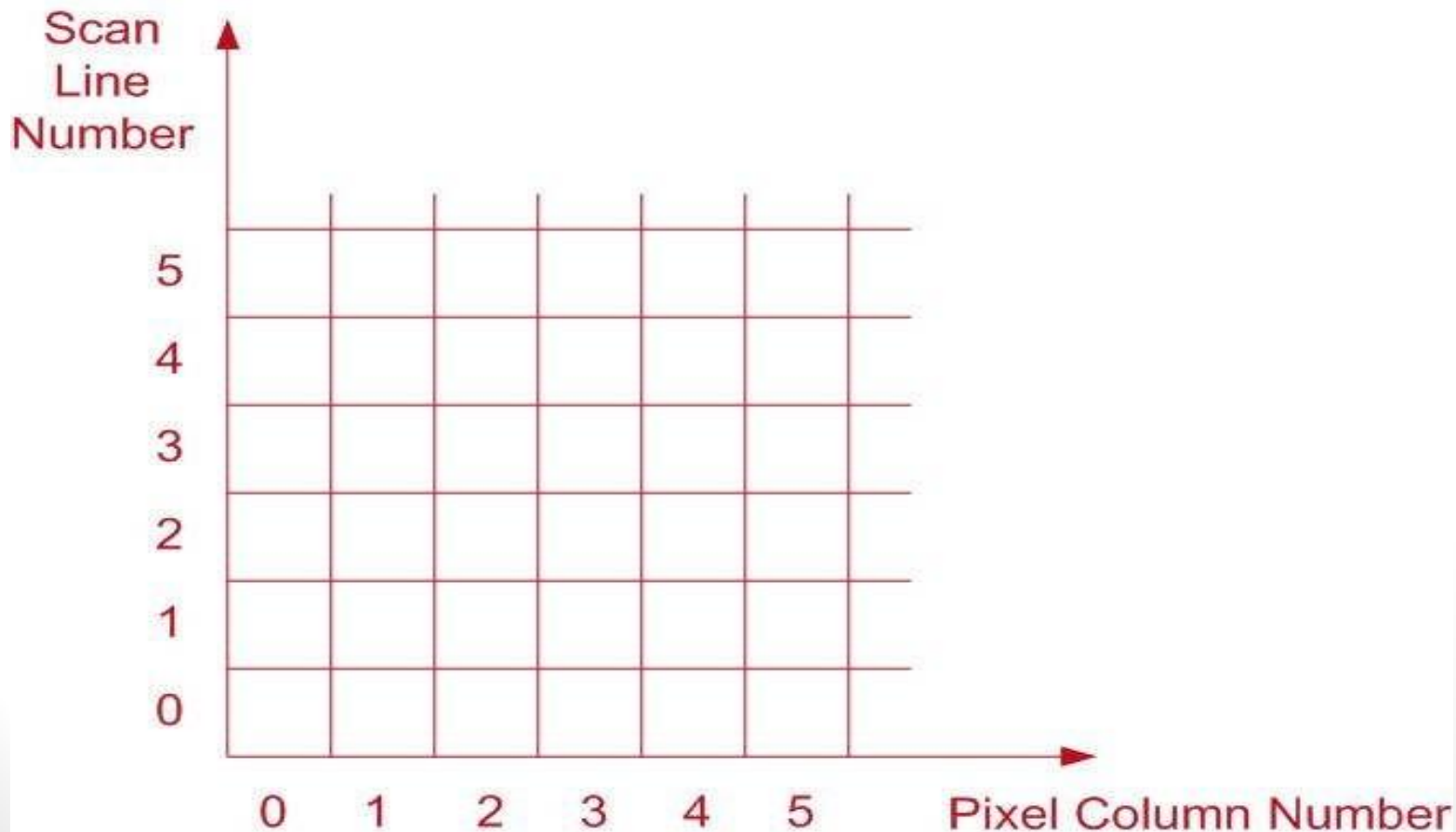
Points and Lines

- Points
 - Plotted by converting co-ordinate position to appropriate operations for the output device (e.g.: in CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location.)
- Line
 - Plotted by calculating intermediate positions along the line path between two specified endpoint positions.
 - Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints → “*the jaggies*”. E.g.: position (10.48,20.51) → (10,21).



Pixel Position

- Pixel position: referenced by scan line number and column number

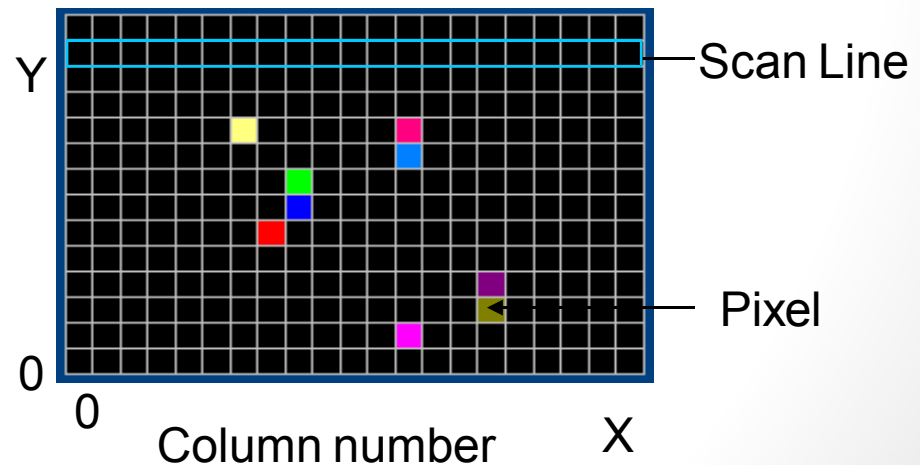
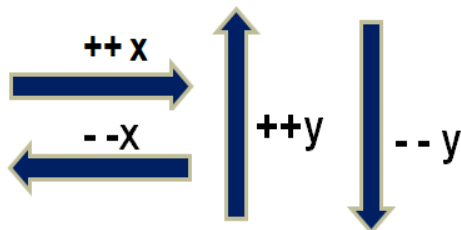


LINE DRAWING ALGORITHM

- DDA Algorithm- (Digital Differential Analyzer)
- Bresenham's Line Algorithm

Note:

- Pixel- picture element → smallest piece of information in an image represented using dots, squares and rectangle
- **Components-**
 - RGB
 - or
 - 4 components
(cyan ,magenta,
yellow & black)



DDA Algorithm

→ Digital Differential Analyzer

Scan-conversion line – algorithm based on calculating either

$$\Delta x \text{ or } \Delta y$$

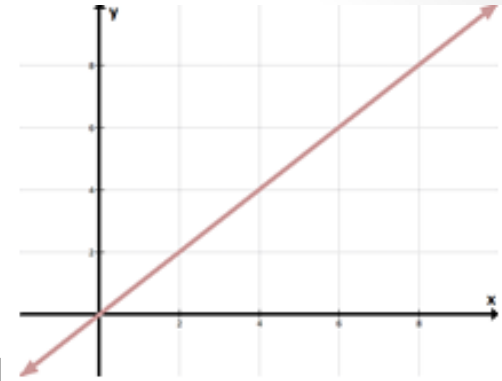
Sample the line at unit intervals in one coordinate

Determine the corresponding integer values nearest the line path in another co-ordinate

CASE 1

Positive Slope

(Y always increase when X increases and
Y always Decreases when X decrease)

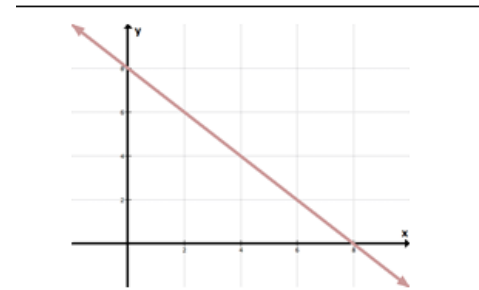


Positive Slope

CASE 2

Negative Slope

(Y always increase when X decrease and
Y always Decreases when X increases)



Negative Slope

CASE 1

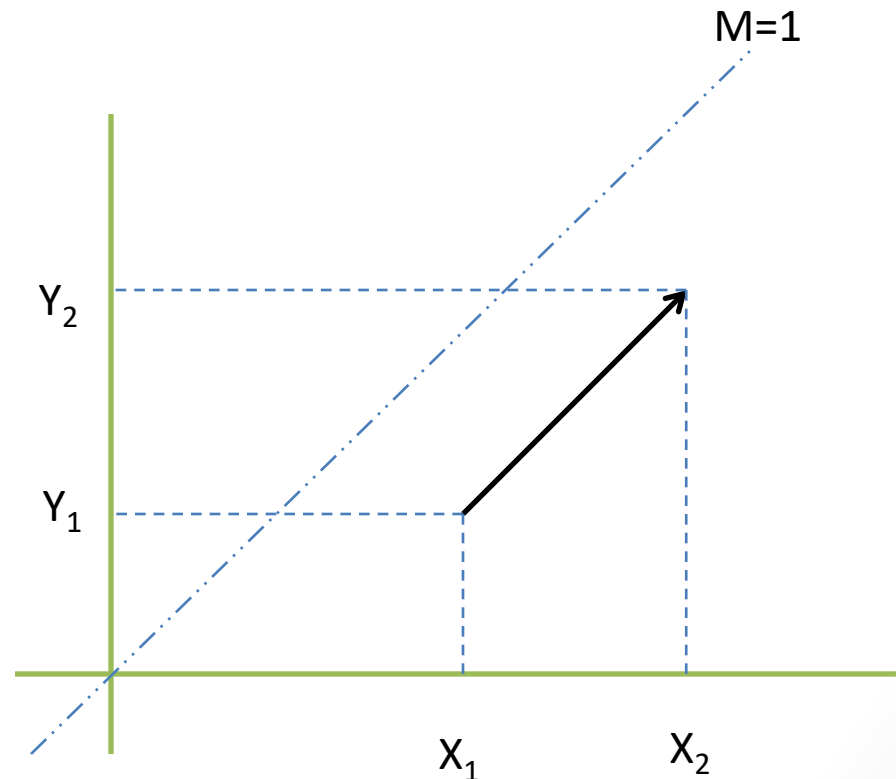
Positive Slope

1. Slope less than 1 ($|M| < 1$) and moving **Left to Right**

$$\Delta x = 1$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + M$$



CASE 1

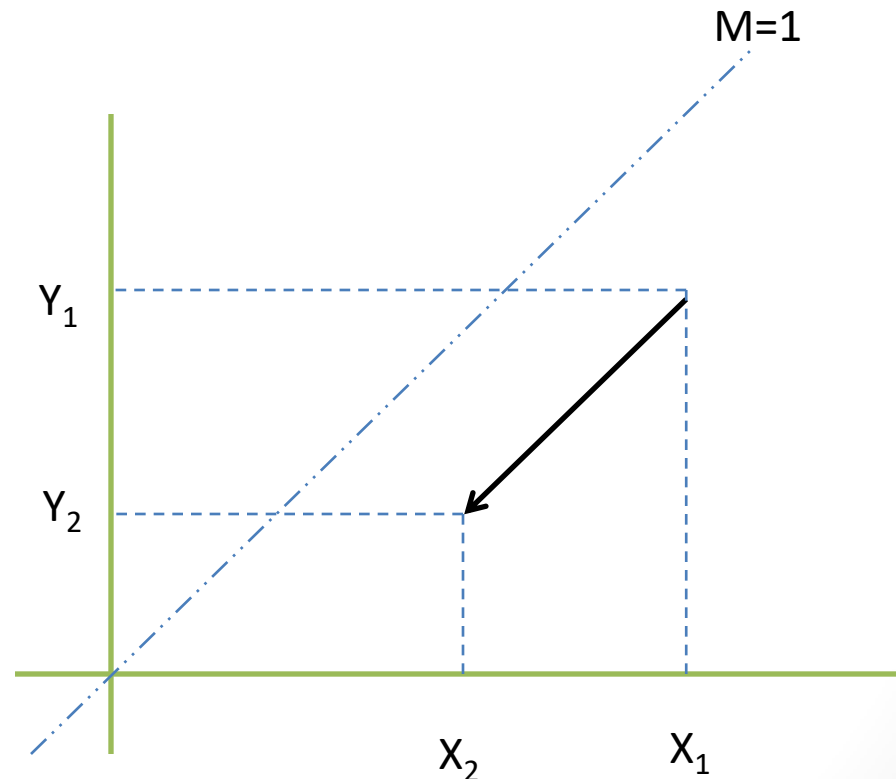
Positive Slope

2. Slope less than 1 ($|M| < 1$) and
moving **Right to Left**

$$\Delta x = -1$$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k - M$$



CASE 1

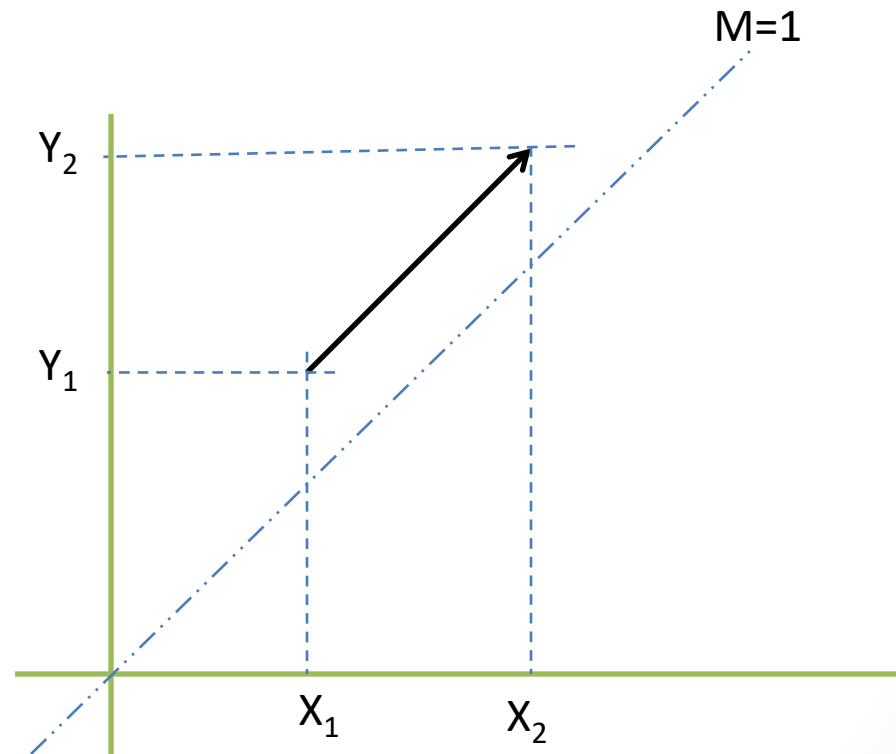
Positive Slope

3. Slope greater than 1 ($|M| > 1$) and
moving **Left to Right**

$$\Delta y = 1$$

$$X_{k+1} = X_k + 1/M$$

$$Y_{k+1} = Y_k + 1$$



CASE 1

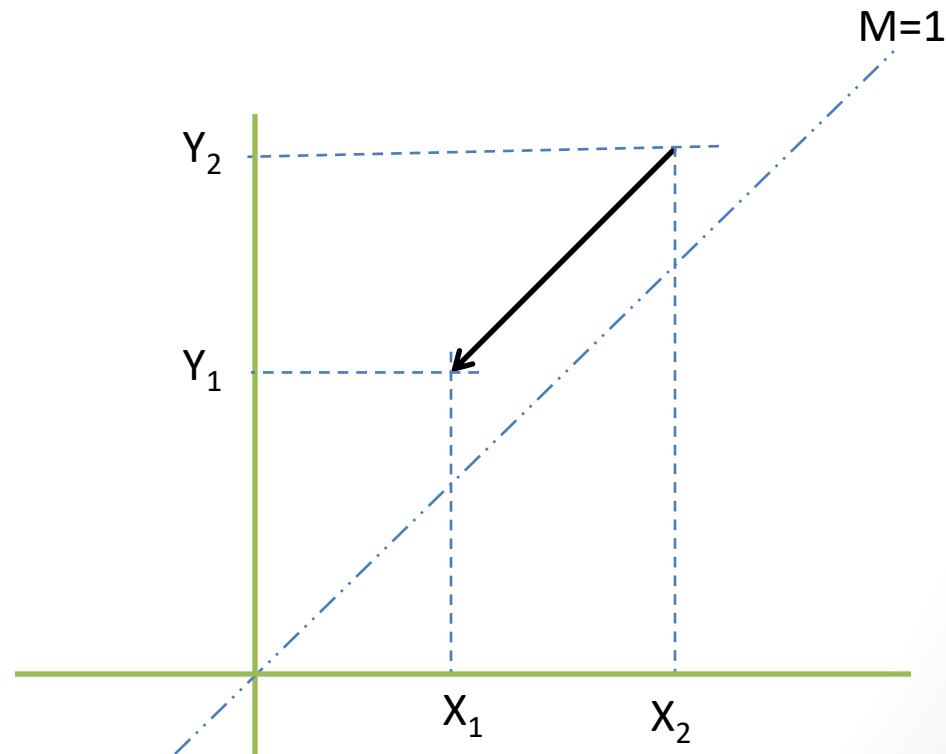
Positive Slope

4. Slope Greater than 1 ($|M| > 1$) and
moving **Right to Left**

$$\Delta y = -1$$

$$X_{k+1} = X_k - 1/M$$

$$Y_{k+1} = Y_k - 1$$



CASE 2

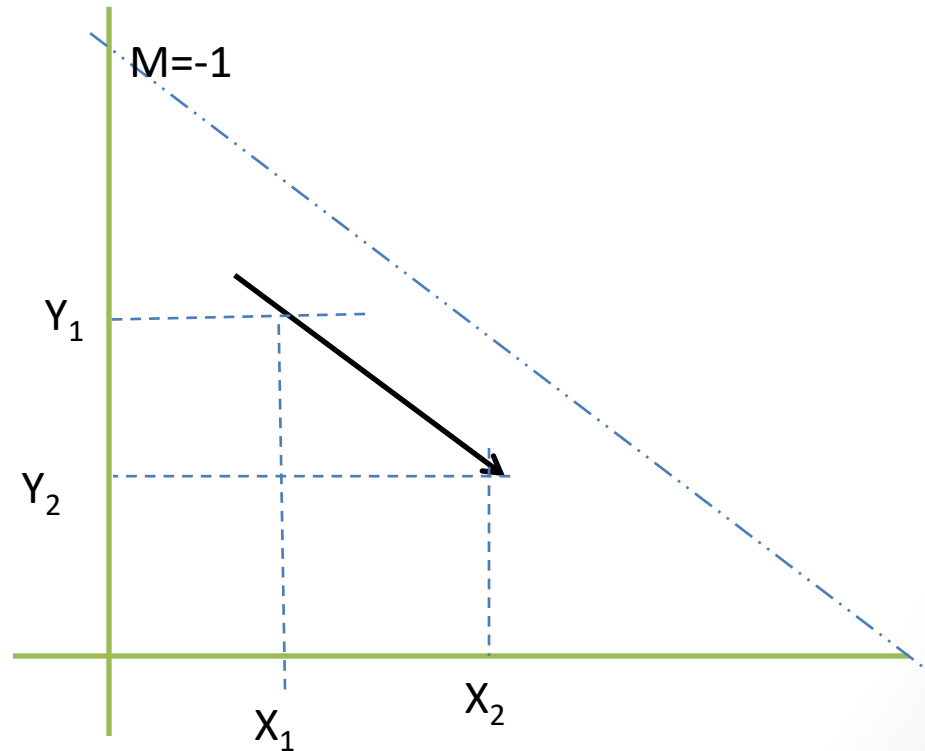
Negative Slope

1. Slope less than 1 ($|M| < 1$) and moving **Left to Right**

$$\Delta x = 1$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + M$$



CASE 2

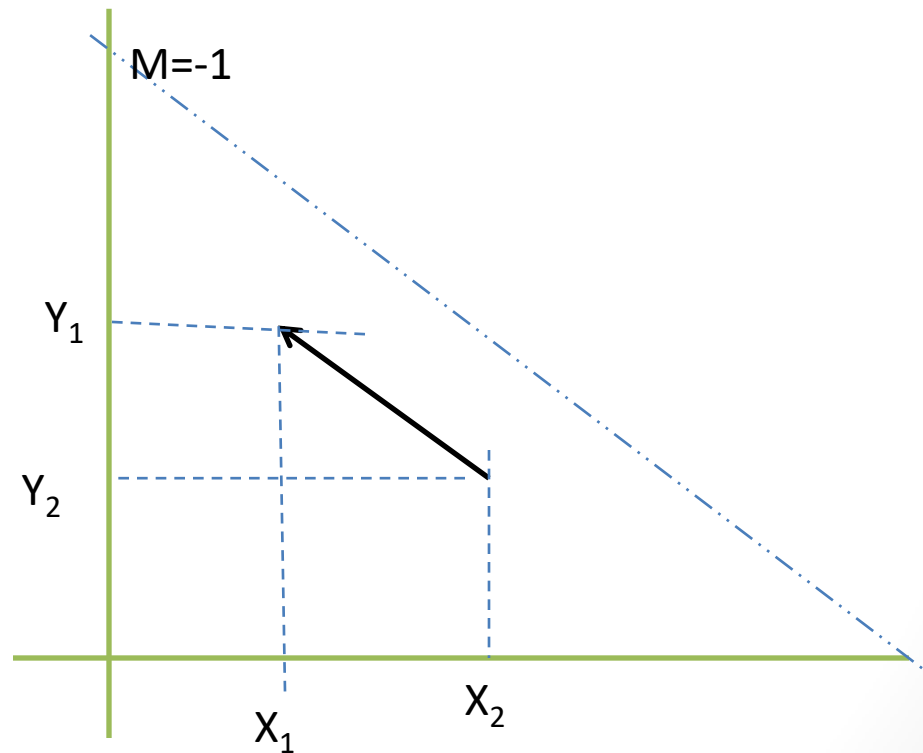
Negative Slope

2. Slope less than 1 ($|M| < 1$) and
moving **Right to Left**

$$\Delta x = -1$$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k - M$$



CASE 2

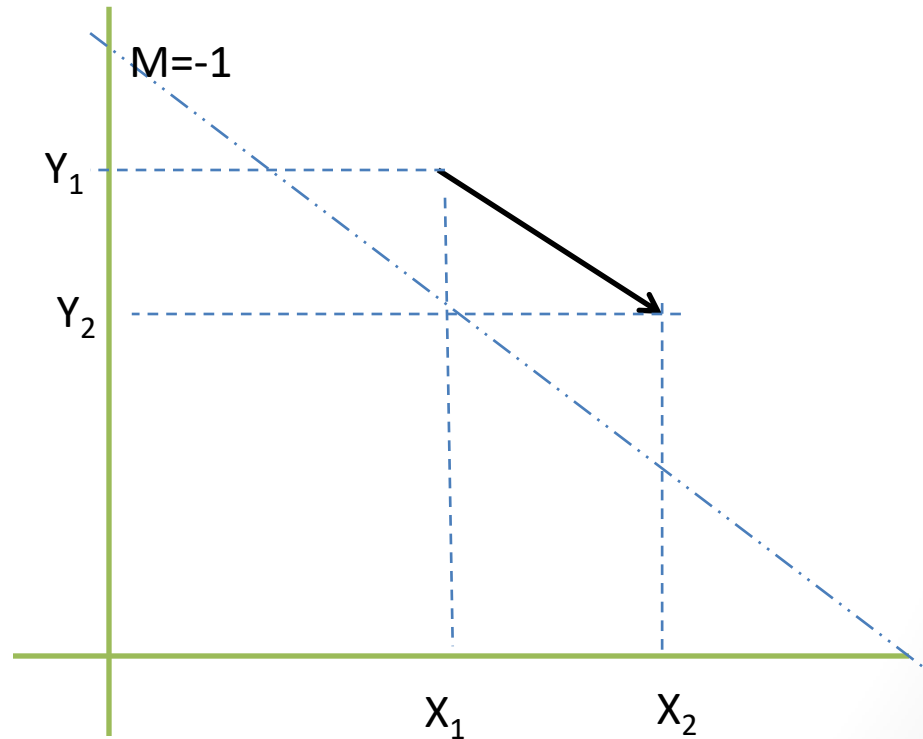
Negative Slope

3. Slope greater than 1 ($|M| > 1$) and moving **Left to Right**

$$\Delta y = -1$$

$$X_{k+1} = X_k - 1/M$$

$$Y_{k+1} = Y_k - 1$$



CASE 2

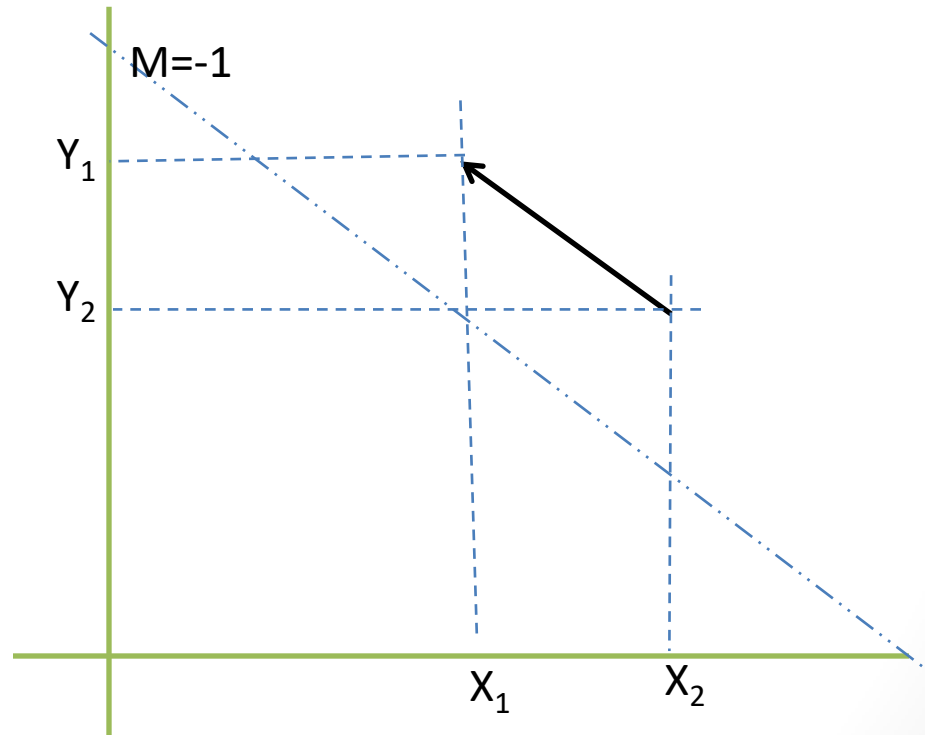
Negative Slope

4. Slope Greater than 1 ($|M| > 1$) and moving **Right to Left**

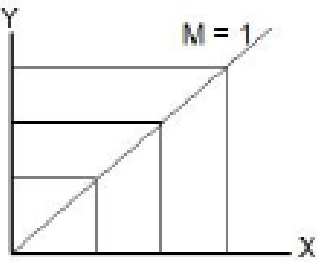
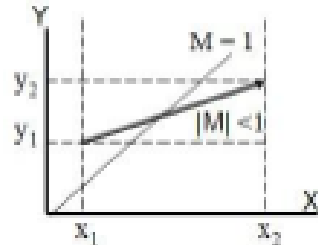
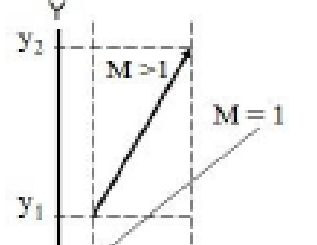
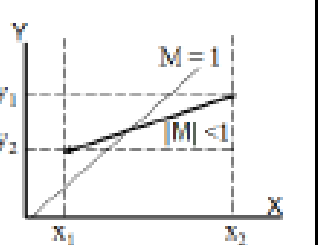
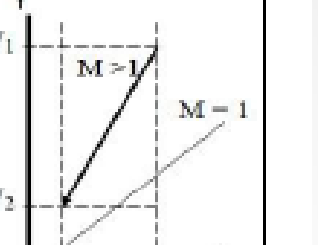
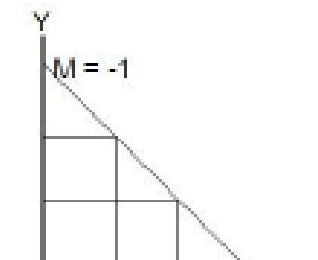
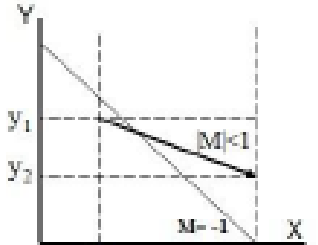
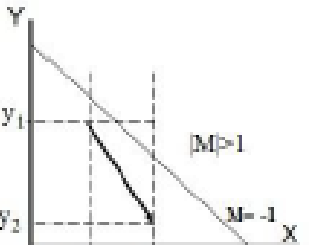
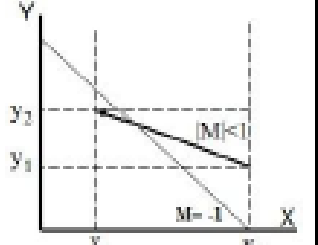
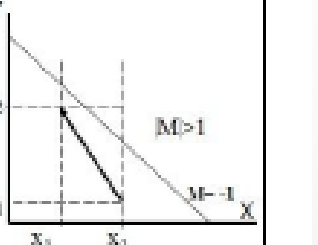
$$\Delta y = 1$$

$$X_{k+1} = X_k + 1/M$$

$$Y_{k+1} = Y_k + 1$$



DDA

	Moving Left to Right		Moving Right to Left	
	Slope (m) < 1	Slope (m) > 1	Slope (m) < 1	Slope (m) > 1
Positive Slope  <i>Fig. Positive Slope</i>	 $x_{k+1} = x_k + 1$ $y_{k+1} = y_k + m$	 $x_{k+1} = x_k + \frac{1}{m}$ $y_{k+1} = y_k + 1$	 $x_{k+1} = x_k - 1$ $y_{k+1} = y_k - m$	 $x_{k+1} = x_k - \frac{1}{m}$ $y_{k+1} = y_k - 1$
Negative Slope  <i>Fig. Negative Slope</i>	 $x_{k+1} = x_k + 1$ $y_{k+1} = y_k - m$	 $x_{k+1} = x_k + \frac{1}{m}$ $y_{k+1} = y_k - 1$	 $x_{k+1} = x_k - 1$ $y_{k+1} = y_k + m$	 $x_{k+1} = x_k - \frac{1}{m}$ $y_{k+1} = y_k + 1$

Q.> Digitize a Line with end point A(2,3) and B(6,8) , using DDA.

A.> Here , Slope (M) = $(8-3)/(6-2) = 1.25$

here Slope is positive and greater than 1

And moving left to right and $1/m = 0.8$

So , we use

$$X_{k+1} = X_k + 1/M$$

$$Y_{k+1} = Y_k + 1$$

K	$X_{k+1} = X_k + 1/M$	$Y_{k+1} = Y_k + 1$	(X,Y)
1	$=2 + 0.8 = 2.8 \sim 3$	4	(3,4)
2	$=2.8 + 0.8 = 3.6 \sim 4$	5	(4,5)
3	$=3.6 + 0.8 = 4.4 \sim 4$	6	(4,6)
4	$=4.4 + 0.8 = 5.2 \sim 5$	7	(5,7)
5	$=5.2 + 0.8 = 6$	8	(6,8)

DDA Examples

<https://genuinenotes.com>

Q.> Digitize a Line with end point A(2,3) and B(6,8) , using DDA
Right to left.

Q.> Digitize a Line with end point A(3,2) and B(8,4) , using DDA

Q.>Digitize a Line with end point A(2,6) and B(4,2) , using DDA

DDA Final Exercise

1. Digitize the line from A(1,1) to B(10,8) using scan conversion line algorithm.
2. Draw a line with two end point P(5,3) and Q(1,2) using DDA
3. Digitize the line with endpoints A(1,7) and B(6,3) using digital differential analyzer line drawing algorithm. Show all necessary steps. (TU) .
4. Digitize the line with end points (1,2) and (5,6) using digital differential analyzer method.(TU).
5. Digitize the given line endpoints (1,1) and (5, 5) using DDA.
6. Draw a line using DDA with two end points A(0,0) and B(-10,8).
7. Digitize line with start at (5,3) and end at (1,5) using DDA.
8. Digitize DDA with all necessary steps with two end points (0,0) and (10,0).

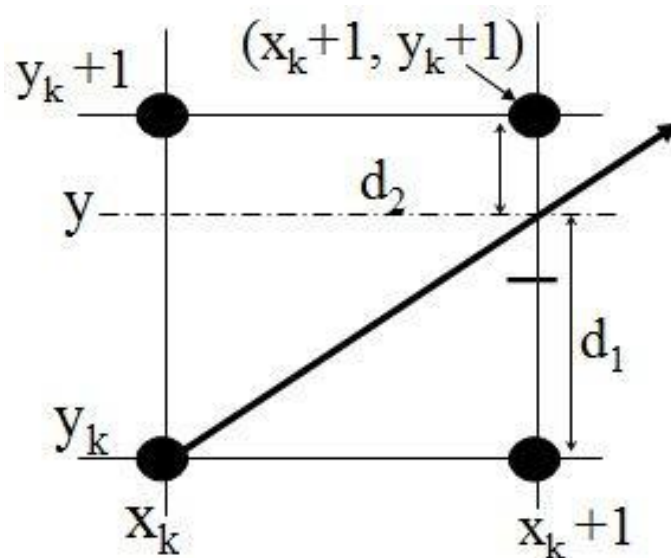
Bresenham's Line Algorithm

The BLA is a more efficient method used to plot pixel position along a straight-line path.

Advantage of BLA over DDA

- In DDA algorithm each successive point is computed in ***floating point***, so it requires ***more time*** and ***more memory space***. While in BLA each successive point is calculated in ***integer value*** or ***whole number***. So it requires less time and less memory space.
- In DDA, since the calculated point value is floating point number, it should be rounded at the end of calculation but in BLA it does not need to round, so there is no accumulation of rounding error.
- Due to rounding error, the line drawn by DDA algorithm is not accurate, while in BLA line is accurate.
- DDA algorithm cannot be used in other application except line drawing, but BLA can be implemented in other application such as circle, ellipse and other curves.

BLA for slope +ve and $|m| \leq 1$

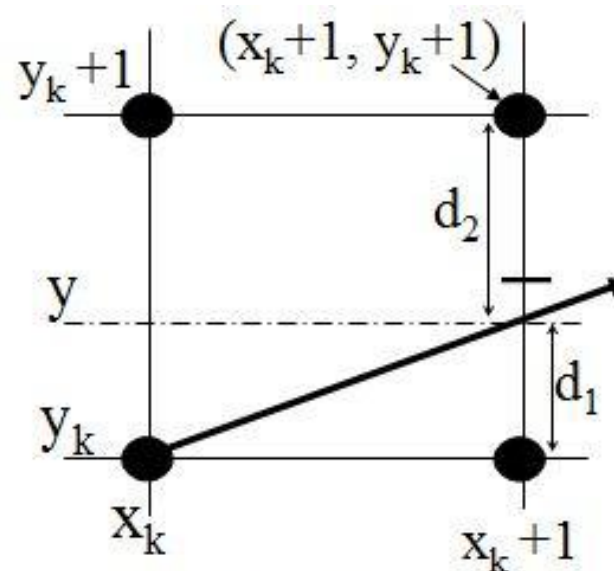


$$P_k = d_1 - d_2 = \text{Positive}$$

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

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

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



$$P_k = d_1 - d_2 = \text{Negative}$$

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

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

$$P_{k+1} = P_k + 2\Delta y$$

Fig. Bresenham's Line Algorithm for $|m| \leq 1$

Let us assume that pixel (x_k, y_k) is already plotted assuming that the **sampling direction** is along X-axis i.e. (x_k+1, y_k) or (x_k+1, y_k+1) . Thus, the common equation of the line is

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

Now,

$$d1 = y - y_k = m(x_k+1) + c - y_k$$

and

$$d2 = (y_k+1) - y = y_{k+1} - \{m(x_k+1) + c\}$$

The difference betⁿ these two separation is,

$$d1 - d2 = [m (x_k + 1) + c - y_k] - [y_k + 1 - \{m (x_k + 1) + c\}]$$

Or,

$$d1 - d2 = 2m (x_k + 1) + 2c - 2y_k - 1$$

Since, slope of line (**m**) = **$\Delta y / \Delta x$** ,

We have,

$$\Delta x (d1 - d2) = 2 \Delta y (x_k + 1) + 2 \Delta x C - 2 \Delta x Y_k - \Delta x$$

Define Decision parameter at K^{th} step,

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

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

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

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

Now, $K+1^{\text{th}}$ term

$$P_{k+1} = 2 \Delta y X_{k+1} + 2 \Delta y + 2 \Delta x c - 2 \Delta x y_{k+1} - \Delta x$$

Here,

$$P_{k+1} - P_k = 2 \Delta y X_{k+1} + 2 \Delta y + 2 \Delta x c - 2 \Delta x y_{k+1} - \Delta x - \{2 \Delta y X_k + 2 \Delta y + 2 \Delta x c - 2 \Delta x y_k - \Delta x\}$$

$$= 2 \Delta y X_{k+1} + \cancel{2 \Delta y} + \cancel{2 \Delta x c} - 2 \Delta x y_{k+1} - \cancel{\Delta x} - 2 \Delta y X_k - \cancel{2 \Delta y} - \cancel{2 \Delta x c} + 2 \Delta x y_k + \cancel{\Delta x}$$

$$P_{k+1} = P_k + 2 \Delta y X_{k+1} - 2 \Delta x y_{k+1} - 2 \Delta y X_k + 2 \Delta x y_k$$

$$P_{k+1} = P_k + 2 \Delta y (X_{k+1} - X_k) - 2 \Delta x (Y_{k+1} - Y_k)$$

Case 1

If $P_k < 0$ (i.e. $d1-d2$ is Negative)

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

$$P_{k+1} = P_k + 2 \Delta y$$

Case 2

If $P_k \geq 0$ (i.e. $d1-d2$ is Positive)

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

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

Initial Decision Parameter (P_0)

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

Let, $X_0 = 0$, $Y_0 = 0$ then $C = 0$

$$\Delta x (d1 - d2) = 2 \Delta y X_k + 2 \Delta y + 2 \Delta x c - 2 \Delta x y_k - \Delta x$$
$$P_0 = 0 + 2 \Delta y + 0 + 0 - \Delta x$$

$$\mathbf{P_0 = 2 \Delta y - \Delta x}$$

Example-1: Digitize the line with end points (20, 10) and (30, 18) using BLA.

Solution :

Here, Starting point of line = $(x_1, y_1) = (20, 10)$ and

Ending point of line = $(x_2, y_2) = (30, 18)$

$$\begin{aligned}\text{Thus, slope of line, } m &= \Delta y / \Delta x = y_2 - y_1 / x_2 - x_1 \\ &= (18 - 10) / (30 - 20) \\ &= 8/10\end{aligned}$$

As the given points, it is clear that the line is moving left to right with the positive slope

$$|m| = 0.8 < 1$$

Thus,

<https://genuinenotes.com>

The initial decision parameter (P_0) = $2\Delta y - \Delta x = 2*8 - 10 = 6$

Since, for the Bresenham's Line drawing Algorithm of slope, $|m| \leq 1$, we have

If $P_k < 0$ (i.e. $d_1 - d_2$ is Negative)

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

$$P_k = P_k + 2 \Delta y$$

If $P_k \geq 0$

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

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

(20,

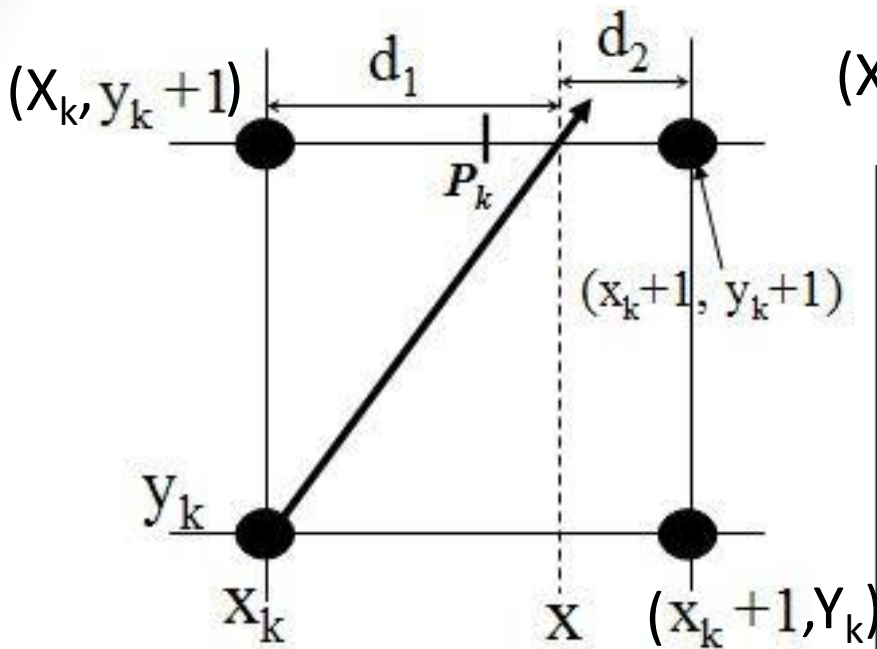
10)

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0.	6	$20+1 = 21$	11	(21, 11)
1.	$6 + 2*8 - 2*10 = 2$	$21+1 = 22$	12	(22, 12)
2.	$2 + 2*8 - 2*10 = -2$	$22+1 = 23$	12	(23, 12)
3.	$-2 + 2*8 = 14$	$23+1 = 24$	13	(24, 13)
4.	$14 + 2*8 - 2*10 = 10$	$24+1 = 25$	14	(25, 14)
5.	$10 + 2*8 - 2*10 = 6$	$25+1 = 26$	15	(26, 15)
6.	$6 + 2*8 - 2*10 = 2$	$26+1 = 27$	16	(27, 16)
7.	$2 + 2*8 - 2*10 = -2$	$27+1 = 28$	16	(28, 16)
8.	$-2 + 2*8 = 14$	$28+1 = 29$	17	(29, 17)
9.	$14 + 2*8 - 2*10 = 10$	$29+1 = 30$	18	(30, 18)

Example-2: Digitize the line with end points (15, 5) and (30, 10) using BLA.

Example-3: Digitize the line with end points (20, 5) and (25, 10) using BLA.

BLA for slope +ve and $|m| > 1$

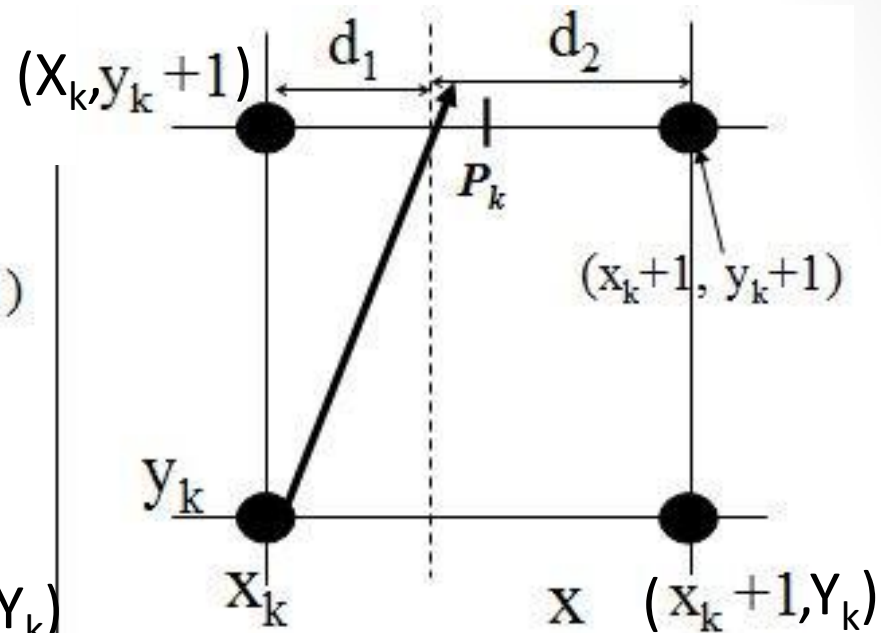


$$P_k = d_1 - d_2 = \text{Positive}$$

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

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

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



$$P_k = d_1 - d_2 = \text{Negative}$$

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

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

$$P_{k+1} = P_k + 2\Delta x$$

Fig. Bresenham's Line Algorithm for $|m| > 1$

Let us assume that pixel (x_k, y_k) is already plotted assuming that the **sampling direction** is along X-axis i.e. (x_k, y_k+1) or (x_k+1, y_k+1) . Thus, the common equation of the line is

$$\underline{Y=MX+C}$$

$$y_k+1 = mx + c$$

$$X = \{y_k+1 - c\}/m$$

Now,

$$d1 = X - X_k = \{y_k + 1 - c\}/m - x_k$$

and

$$d2 = (X_k + 1) - X = x_k + 1 - \{y_k + 1 - c\}/m$$

So,

$$d1 - d2 = [\{y_k + 1 - c\}/m - x_k] - [x_k + 1 - \{y_k + 1 - c\}/m]$$

Or,

$$d1 - d2 = 2/m * (y_k + 1) - 2c/m - 2x_k - 1$$

Since, slope of line **(m) = $\Delta y / \Delta x$** ,

we have

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

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

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

Now, $K+1^{\text{th}}$ term

$$P_{k+1} = 2\Delta x (y_{k+1} + 1) - 2c \Delta x - 2\Delta y x_{k+1} - \Delta y$$

Here,

$$\begin{aligned} P_{k+1} - P_k &= \{2\Delta x (y_{k+1} + 1) - 2c \Delta x - 2\Delta y x_{k+1} - \Delta y\} - \\ &\quad \{2\Delta x (y_k + 1) - 2c \Delta x - 2\Delta y x_k - \Delta y\} \\ &= 2\Delta x (y_{k+1} - y_k) - 2\Delta y (x_{k+1} - x_k) \end{aligned}$$

Or,

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

<https://genuinenotes.com>

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

Case 1

If $P_k < 0$ (i.e. $d1-d2$ is Negative)

then,

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_{k+1} = P_k + 2 \Delta x$$

Case 2

If $P_k \geq 0$ (i.e. $d1-d2$ is Positive)

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

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

Initial Decision Parameter (P_0)

We Know,

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

Let, $X_0 = 0$, $Y_0 = 0$ then $C = 0$

Then,

$$P_0 = 2\Delta x (y_0 + 1) - 2c \Delta x - 2\Delta y x_0 - \Delta y$$

$$P_0 = 2 \Delta x - \Delta y$$

Example-4: Digitize the line with end points (1, 0) and (3, 3) using BLA.

Solution :

Here,

Starting point of line = $(x_1, y_1) = (1, 0)$ and

Ending point of line = $(x_2, y_2) = (3, 3)$

Thus,

$$\begin{aligned}\text{slope of line, } m &= \Delta y / \Delta x = y_2 - y_1 / x_2 - x_1 \\ &= (3 - 0) / (3 - 1) \\ &= 3/2\end{aligned}$$

As the given points, it is clear that the line is moving left to right with the positive slope,

$$|m| = 1.5 > 1$$

Thus, the initial decision parameter

$$(P_0) = 2\Delta x - \Delta y = 2*2 - 3 = 1$$

we have

$$\Delta x = x_2 - x_1 = 3 - 1 = 2$$

$$\Delta y = y_2 - y_1 = 3 - 0 = 3$$

If $P_k < 0$

then,

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

If $P_k \geq 0$

then,

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

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

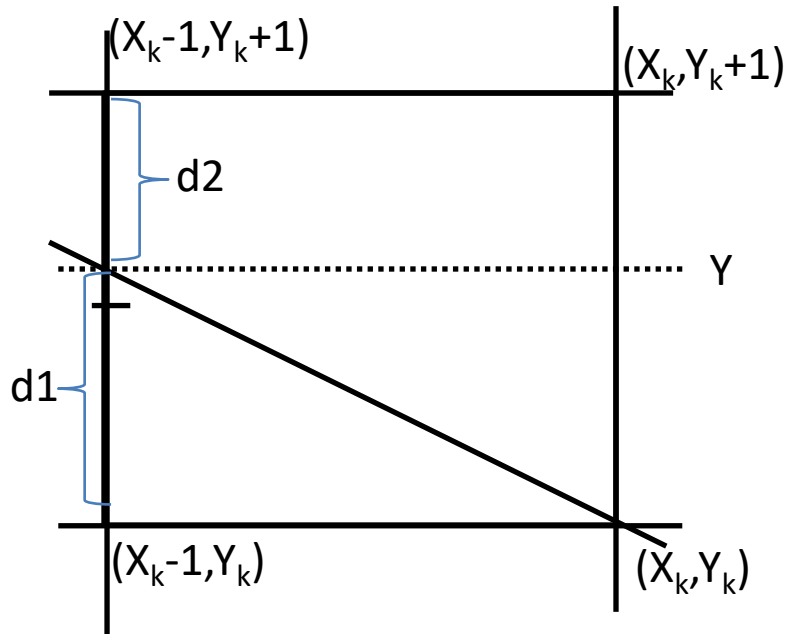
k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0.	1	2	1	(2, 1)
1.	$1 + 2*2 - 2*3 = -1$	2	$1+1 = 2$	(2, 2)
2.	$-1 + 2*2 = 3$	3	$2+1 = 3$	(3, 3)

Example-5: Digitize A(5,2) and B(7,8) using BLA.

Example

- 1. Digitize the line A(1,1) and B(5,6) using BLA.**
- 2. Digitize the line A(7,8) and B(1,4) using BLA.**
- 3. Digitize the line A(1,1) and B(5,6) using DDA.**
- 4. Digitize the line A(5,2) and B(7,8) using BLA.**
- 5. Digitize the line A(0,0) and B(10,10) using BLA.**
- 6. Digitize the line A(1,3) and B(10,10) using BLA and DDA.**

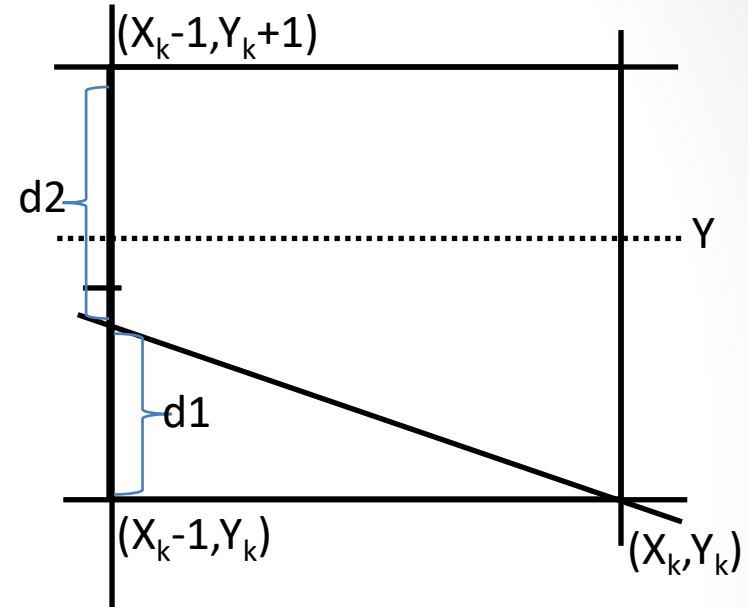
Slope -Ve and $|M| \leq 1$



$d1-d2 > 0$ (Positive)

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$



$d1-d2 < 0$ (Negative)

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k$$

Let us assume that pixel (x_k, y_k) is already plotted assuming that the **sampling direction** is along X-axis i.e. (x_k-1, y_k+1) or (x_k-1, y_k) . Thus, the common equation of the line is

$$Y = Mx + C$$

Here,

$$Y = m(X_k - 1) + c$$

Now,

$$d1 = Y - Y_k$$

And

$$d2 = (Y_k + 1) - Y$$

Or,

$$d1 - d2 = Y - Y_k - \{(Y_k + 1) - Y\}$$

$$= Y - Y_k - Y_k - 1 + Y$$

$$= 2Y - 2Y_k - 1$$

$$= 2m(X_k - 1) + 2c - 2Y_k - 1$$

Here slope of line **(m)** = $-(\Delta y / \Delta x)$, {for -ve slope}

$$d1 - d2 = 2m(X_k - 1) + 2c - 2y_k - 1$$

Now , (m) = - (Δy / Δx),

$$d1 - d2 = -2 \{(\Delta y / \Delta x)\} (X_k - 1) + 2c - 2y_k - 1$$

$$\Delta x(d1 - d2) = -2 \Delta y (X_k - 1) + 2c \Delta x - 2 \Delta x Y_k - \Delta x$$

$$\text{Here } P_k = \Delta x(d1 - d2)$$

$$= -2 \Delta y (X_k - 1) + 2c \Delta x - 2 \Delta x Y_k - \Delta x$$

$$P_k = -2 \Delta y X_k + 2 \Delta y + 2c \Delta x - 2 \Delta x Y_k - \Delta x$$

Now $k+1^{\text{th}}$ term

$$P_{k+1} = -2 \Delta y X_{k+1} + 2 \Delta y + 2c \Delta x - 2 \Delta x Y_{k+1} - \Delta x$$

Or,

$$P_{k+1} - P_k = -2 \Delta y X_{k+1} + 2 \Delta y + 2c \Delta x - 2 \Delta x Y_{k+1} - \Delta x$$

$$+ 2 \Delta y X_k - 2 \Delta y - 2c \Delta x + 2 \Delta x Y_k + \Delta x$$

$$= -2\Delta y(X_{k+1} - X_k) + 2\Delta x (-Y_{k+1} + Y_k)$$

$$P_{k+1} = P_k - 2\Delta y(X_{k+1} - X_k) + 2\Delta x (-Y_{k+1} + Y_k)$$

<https://genuinenotes.com>

$$P_{k+1} = P_k - 2\Delta y(X_{k+1} - X_k) - 2\Delta x(Y_{k+1} - Y_k)$$

Case 1

If $P_k < 0$ (i.e. $d1-d2$ is Negative)

then,

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k$$

$$P_{k+1} = P_k + 2 \Delta y$$

Case 2

If $P_k \geq 0$ (i.e. $d1-d2$ is Positive)

then,

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

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

Initial Decision Parameter (P_0)

We Know,

$$P_k = -2 \Delta y X_k + 2 \Delta y + 2 c \Delta x - 2 \Delta x Y_k - \Delta x$$

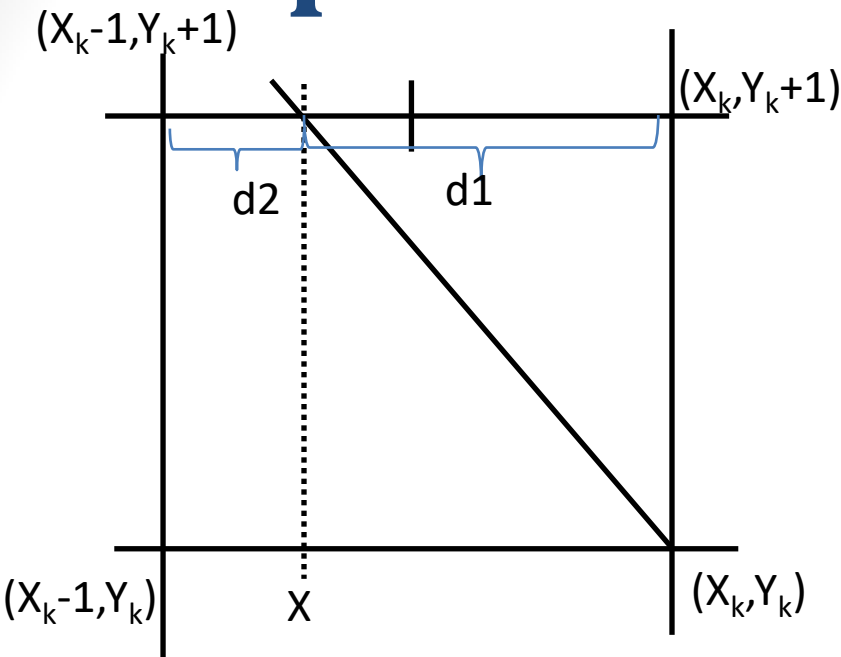
Let, $X_0 = 0$, $Y_0 = 0$ then $C = 0$

Then,

$$P_0 = -2 \Delta y X_0 + 2 \Delta y + 2 c \Delta x - 2 \Delta x Y_0 - \Delta x$$

$$P_0 = 2 \Delta y - \Delta x$$

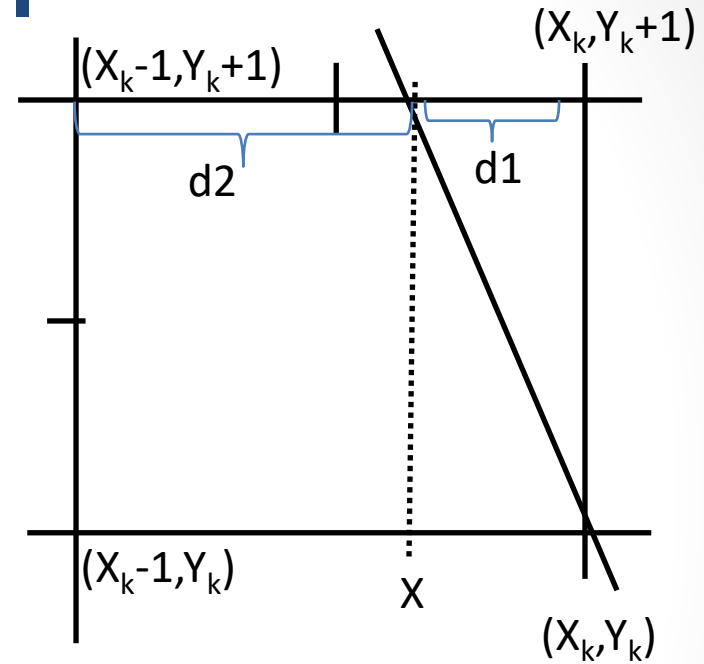
Slope -Ve and $|M| > 1$



$d1 - d2 > 0$ (Positive)

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$



$d1 - d2 < 0$ (Negative)

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

Let us assume that pixel (x_k, y_k) is already plotted assuming that the **sampling direction** is along X-axis i.e. (x_k-1, y_k+1) or (x_k, y_k+1) . Thus, the common equation of the line is

$$Y = Mx + C$$

Here,

$$Y_k + 1 = mX + C$$

$$X = \{Y_k + 1 - C\} / M$$

Now,

$$d1 = X_k - X$$

Slope -Ve and $|M| > 1$

And

$$d2 = X - \{X_k - 1\}$$

Or,

$$\begin{aligned} d1 - d2 &= X_k - X - \{X - \{X_k - 1\}\} \\ &= X_k - X - X + X_k - 1 \\ &= 2X_k - 2X - 1 \\ &= 2X_k - 2\{Y_k + 1 - C\} / M - 1 \end{aligned}$$

Here slope of line **(m) = - ($\Delta y / \Delta x$), {for -ve slope}**

$$d1 - d2 = 2X_k - 2 \{Y_k + 1 - C\} / M - 1$$

$$= 2X_k - 2 \{Y_k + 1 - C\} / \{- (\Delta y / \Delta x)\} - 1$$

$$= 2X_k + 2 \{Y_k + 1 - C\} / \{ (\Delta y / \Delta x)\} - 1$$

$$\Delta y (d1 - d2) = 2 \Delta y X_k + 2 \Delta x \{Y_k + 1 - C\} - \Delta y$$

$$= 2 \Delta y X_k + 2 \Delta x Y_k + 2 \Delta x - 2 \Delta x C - \Delta y$$

$$P_k = \Delta y (d1 - d2) \text{ for decision parameter}$$

$$P_k = 2 \Delta y X_k + 2 \Delta x Y_k + 2 \Delta x - 2 \Delta x C - \Delta y$$

Now K+1 th term

$$P_{k+1} = 2 \Delta y X_{k+1} + 2 \Delta x Y_{k+1} + 2 \Delta x - 2 \Delta x C - \Delta y$$

Or,

$$P_{k+1} - P_k = 2 \Delta y X_{k+1} + 2 \Delta x Y_{k+1} + 2 \Delta x - 2 \Delta x C - \Delta y$$

$$- 2 \Delta y X_k - 2 \Delta x Y_k - 2 \Delta x + 2 \Delta x C + \Delta y$$

$$= 2 \Delta y (X_{k+1} - X_k) + 2 \Delta x (Y_{k+1} - Y_k)$$

$$P_{k+1} = P_k + 2 \Delta y (X_{k+1} - X_k) + 2 \Delta x (Y_{k+1} - Y_k)$$

<https://genuineenotes.com>

$$P_{k+1} = P_k + 2 \Delta y (X_{k+1} - X_k) + 2 \Delta x (Y_{k+1} - Y_k)$$

Case 1

If $P_k < 0$ (i.e. $d1-d2$ is Negative)

then,

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

Case 2

If $P_k \geq 0$ (i.e. $d1-d2$ is Positive)

then,

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

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

Initial Decision Parameter (P_0)

We Know,

$$P_k = 2 \Delta y X_k + 2 \Delta x Y_k + 2 \Delta x - 2 \Delta x C - \Delta y$$

Let, $X_0 = 0$, $Y_0 = 0$ then $C = 0$

Then,

$$P_0 = \cancel{2 \Delta y X_0} + \cancel{2 \Delta x Y_0} + 2 \Delta x - \cancel{2 \Delta x C} - \Delta y$$

$$P_0 = 2 \Delta x - \Delta y$$

Example: Digitize the given line endpoints (5,10) and (10,7) using Bresenham's line drawing algorithm. *(TU2014)*

Solution:

Here, $(X_1,Y_1)=(10,7)$ &
 $(X_2,Y_2)=(5,10)$

$M=(10-7)/(5-10) = -3/5$ –ve slope and $|M| < 1$

$\Delta x = |5-10| = 5$
 $\Delta y = |10-7| = 3$

here,
Initial $(P_0) = 2 \Delta y - \Delta x$
 $= 2 \times 3 - 5$
 $= 1$

If $P_k < 0$
 $X_{k+1} = X_k - 1$
 $Y_{k+1} = Y_k$
 $P_k = P_k + 2 \Delta y$

If $P_k \geq 0$
 $X_{k+1} = X_k + 1$
 $Y_{k+1} = Y_k - 1$
 $P_k = P_k + 2 \Delta y - 2 \Delta x$

(10, 7)				
K	P _k	X _{k+1}	Y _{k+1}	(X _{k+1} , Y _{k+1})
0	1	9	8	(9,8)
1	= 1 + 2x3 - 2x5 =-3	8	8	(8,8)
2	= -3 + 2x3 = 3	7	9	(7,9)
3	= 3 + 2x3 - 2x5 = -1	6	9	(6,9)
4	= -1 + 2x3 = 5	5	10	(5,10)

Example: Digitize line with endpoints (3, 10) and (6, 2) using Bresenham's Line Drawing Algorithm. (**TU 2016**)

Solution:

Here, $(X_1, Y_1) = (6, 2)$ &
 $(X_2, Y_2) = (3, 10)$

$M = (10 - 2) / (3 - 6) = 8 / -3$ -ve slope and $|M| > 1$

If $P_k < 0$

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

If $P_k \geq 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

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

$$\Delta x = |3 - 6| = 3$$

$$\Delta y = |10 - 2| = 8$$

here,

$$\begin{aligned} \text{Initial } (P_0) &= 2 \Delta x - \Delta y \\ &= 2 \times 3 - 8 \\ &= -2 \end{aligned}$$

(6, 2)				
K	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-2	6	3	(6,3)
1	$= -2 + 2 \times 3 = 4$	5	4	(5,4)
2	$= 4 + 6 - 16 = -6$	5	5	(5,5)
3	$= -6 + 6 = 0$	4	7	(4,7)
4	$= 0 + 6 - 16 = -10$	4	8	(4,8)
5	$= -10 + 6 = -4$	4	9	(4,9)
6	$= -4 + 6 = 2$	3	10	(3,10)

Example: Digitize the given line endpoints (15, 15) and (10, 18) using Bresenham's line drawing algorithm.

Solution:

Here, $(X_1, Y_1) = (15, 15)$ &
 $(X_2, Y_2) = (10, 18)$

$M = (18 - 15) / (10 - 15) = 3 / -5$ -ve slope and $|M| < 1$

$\Delta x = |10 - 15| = 5$

$\Delta y = |18 - 15| = 3$

here,
Initial $(P_0) = 2 \Delta y - \Delta x$
 $= 2 \times 3 - 5$
 $= 1$

If $P_k < 0$
 $X_{k+1} = X_k - 1$
 $Y_{k+1} = Y_k$
 $P_k = P_k + 2 \Delta y$

If $P_k \geq 0$
 $X_{k+1} = X_k + 1$
 $Y_{k+1} = Y_k - 1$
 $P_k = P_k + 2 \Delta y - 2 \Delta x$

(15, 15)				
K	P _k	X _{k+1}	Y _{k+1}	(X _{k+1} , Y _{k+1})
0	1	14	16	(14, 16)
1	= 1 + 2x3 - 2x5 = -3	13	16	(13, 16)
2	= -3 + 2x3 = 3	12	17	(12, 17)
3	= 3 + 2x3 - 2x5 = -1	11	17	(11, 17)
4	= -1 + 2x3 = 5	10	18	(10, 18)

Example: Digitize the given line endpoints (10, 10) and (20, 5) using Bresenham's line drawing algorithm.

Solution:

Here, $(X_1, Y_1) = (20, 5)$ &
 $(X_2, Y_2) = (10, 10)$

$M = (10 - 5) / (10 - 20) = 5 / -10$ -ve slope and $|M| < 1$

$\Delta x = |10 - 20| = 10$
 $\Delta y = |10 - 5| = 5$

here,
Initial $(P_0) = 2 \Delta y - \Delta x$
 $= 2 \times 5 - 10$
 $= 0$

If $P_k < 0$
 $X_{k+1} = X_k - 1$
 $Y_{k+1} = Y_k$

$P_k = P_k + 2 \Delta y$

If $P_k \geq 0$
 $X_{k+1} = X_k + 1$
 $Y_{k+1} = Y_k - 1$

$P_k = P_k + 2 \Delta y - 2 \Delta x$
(20, 5)

K	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	0	19	6	(19,6)
1	$= 0 + 2 \times 5 - 2 \times 10 = -10$	18	6	(18,6)
2	$= -10 + 2 \times 5 = 0$	17	7	(17,7)
3	$= 0 + 2 \times 5 - 2 \times 10 = -10$	16	7	(16,7)
4	$= -10 + 2 \times 5 = 0$	15	8	(15,8)
5	$= 0 + 2 \times 5 - 2 \times 10 = -10$	14	8	(14,8)
6	$= -10 + 2 \times 5 = 0$	13	9	(13,9)
7	$= 0 + 2 \times 5 - 2 \times 10 = -10$	12	9	(12,9)
8	$= -10 + 2 \times 5 = 0$	11	10	(11,10)
9	$= 0 + 2 \times 5 - 2 \times 10 = -10$	10	10	(10,10)

Example: Digitize line with endpoints (5, 6) and (6, 3) using Bresenham's Line Drawing Algorithm.

Solution:

Here, $(X_1, Y_1) = (6, 3)$ &
 $(X_2, Y_2) = (5, 6)$

$M = (6-3)/(5-6) = 3/-1$ -ve slope and $|M| > 1$

If $P_k < 0$

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

If $P_k \geq 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

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

$$\Delta x = |5-6| = 1$$

$$\Delta y = |6-3| = 3$$

here,

$$\begin{aligned} \text{Initial } (P_0) &= 2 \Delta x - \Delta y \\ &= 2 \times 1 - 3 \\ &= -1 \end{aligned}$$

(6, 3)				
K	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-1	6	4	(6,4)
1	$= -1 + 2 \times 1 = 1$	5	5	(5,5)
2	$= 1 + 2 \times 1 - 2 \times 3 = -3$	5	6	(5,6)

$$|M| \leq 1$$

if $P_k < 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

$$P_k = P_k + 2 \Delta y$$

If $P_k \geq 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

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

$$P_0 = 2 \Delta y - \Delta x$$

$$|M| > 1$$

If $P_k < 0$

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

If $P_k \geq 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k + 1$$

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

$$P_0 = 2 \Delta x - \Delta y$$

Bresenham's Line Algorithm

BLA for slope -ve

$$|M| \leq 1$$

If $P_k < 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k$$

$$P_k = P_k + 2 \Delta y$$

If $P_k \geq 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

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

$$P_0 = 2 \Delta y - \Delta x$$

$$|M| > 1$$

If $P_k < 0$

$$X_{k+1} = X_k$$

$$Y_{k+1} = Y_k + 1$$

$$P_k = P_k + 2 \Delta x$$

If $P_k \geq 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

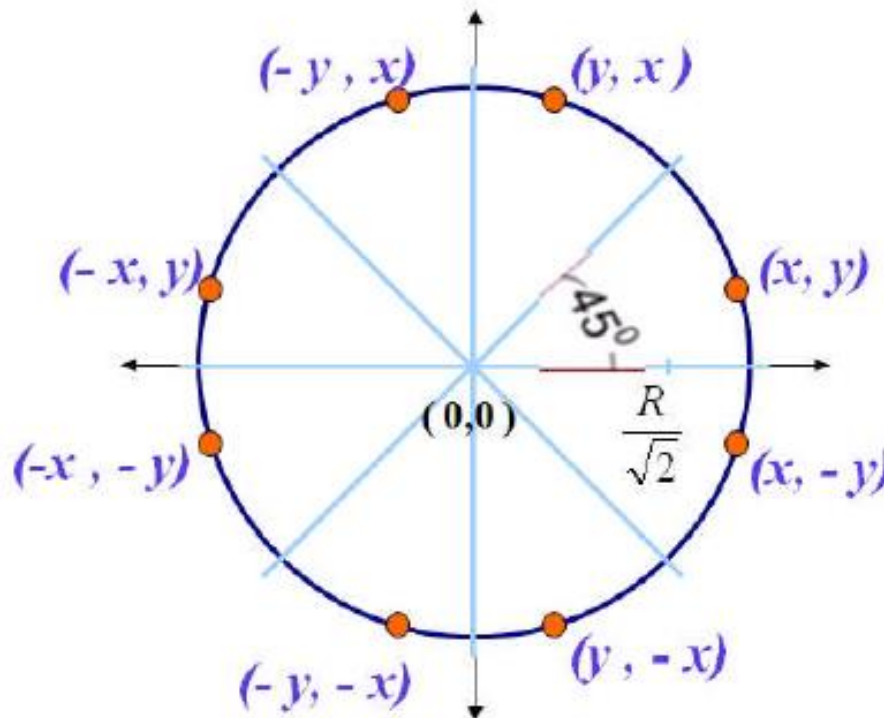
$$P_k = P_k + 2 \Delta x - 2 \Delta y$$

$$P_0 = 2 \Delta x - \Delta y$$

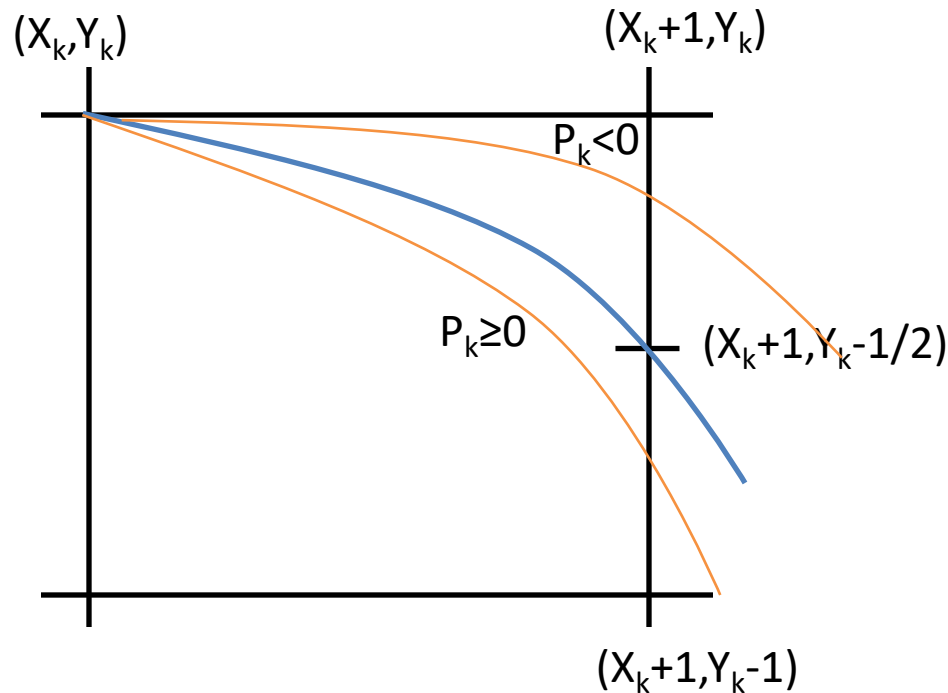
Circle Algorithm

What is Circle ?

- Similarly to the case with lines, there is an incremental algorithm for drawing circles – the *mid-point circle algorithm*
- In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points



Mid Point Circle Algorithm



$$P_k < 0$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

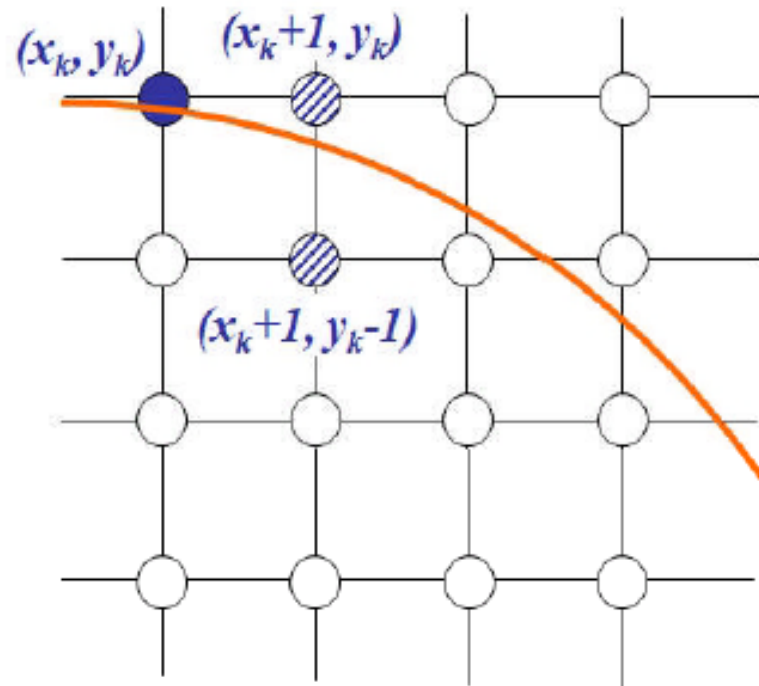
$$P_k \geq 0$$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k - 1$$

Mid Point Circle Algorithm

- Assume that we have just plotted point (x_k, y_k)
- The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)
- We would like to choose the point that is nearest to the actual circle
- So how do we make this choice?



Mid Point Circle Algorithm

- Circle function defined as:

$$f_{circle}(x, y) = x^2 + y^2 - r^2$$

- Any point (x , y) satisfies following conditions

$$f_{circle}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

By evaluating this function at the midpoint between the candidate pixels we can make our decision

- Decision parameter is the circle function: evaluated as:

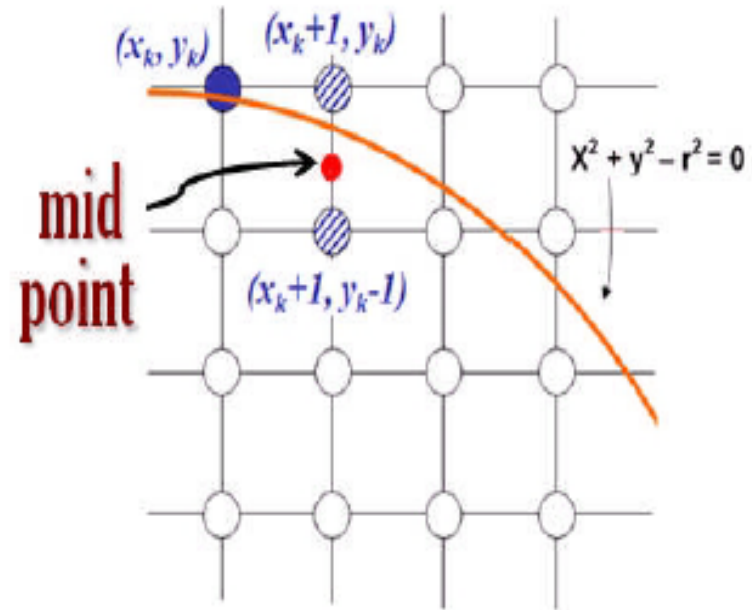
$$p_k = f_{circle} \left(x_k + 1, y_k - \frac{1}{2} \right)$$

$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2} \right)^2 - r^2$$

$$p_{k+1} = f_{circle} \left(x_{k+1} + 1, y_{k+1} - \frac{1}{2} \right)$$

$$= [(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2} \right)^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$



$$\therefore [p_{k+1} - p_k]$$

$$P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

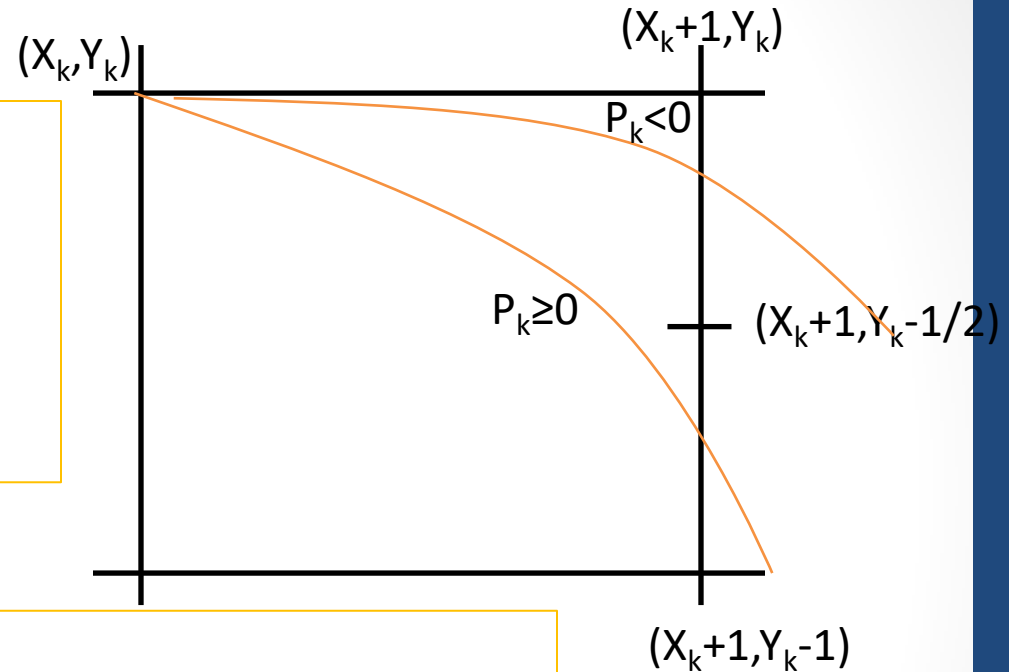
Case 1 : $P_k < 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

$$P_{k+1} = P_k + 2(X_k + 1) + 1$$

$$P_{k+1} = P_k + 2X_{k+1} + 1$$



Case 2: $P_k \geq 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k - 1$$

$$\begin{aligned} P_{k+1} &= P_k + 2(x_k + 1) + [(y_k - 1)^2 - y_k^2] - (y_k - 1 - y_k) + 1 \\ &= P_k + 2(x_k + 1) + [y_k^2 - 2y_k + 1 - y_k^2] - (y_k - 1 - y_k) + 1 \\ &= P_k + 2(x_k + 1) - 2y_k + 1 + 1 + 1 \\ &= P_k + 2(x_k + 1) - 2(y_k + 1) + 1 \end{aligned}$$

$$P_{k+1} = P_k + 2X_{k+1} - 2Y_{k+1} + 1$$

Initial decision parameter

i.e.,

$$(x_0, y_0) = (0, r)$$

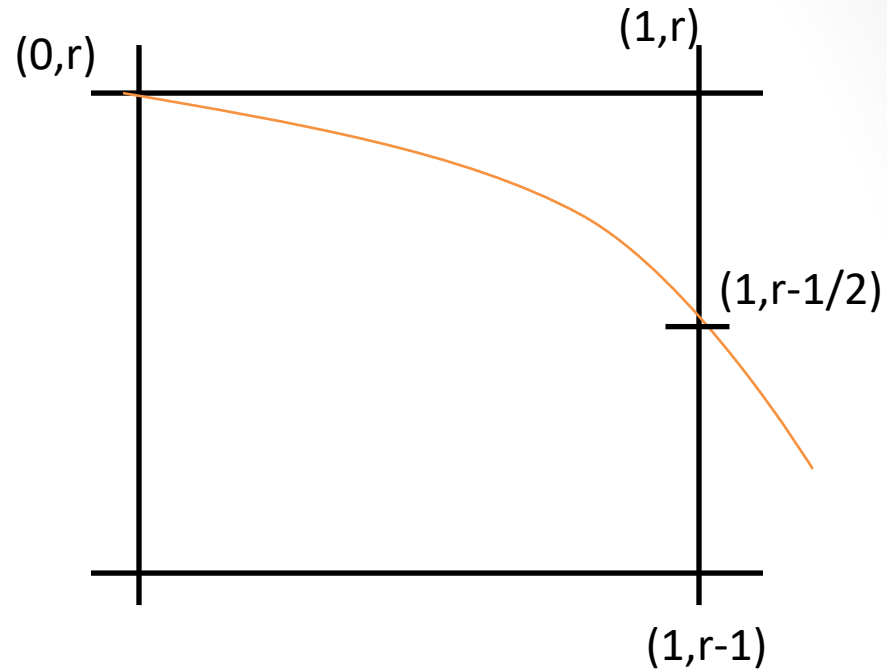
Here ,

$$F_{\text{circle}}(1, r-1/2) = P_0$$

Now,

$$\begin{aligned} P_0 &= 1^2 + (r-1/2)^2 - r^2 \\ &= 1 + r^2 - r + 1/4 - r^2 \\ &= 5/4 - r \end{aligned}$$

$$P_0 \approx 1-r$$



$$P_0 \approx 1-r$$

Example : Digitize a circle with radius 10 at center (0,0)

Solution

Here, the initial decision parameter (P_0) = $1 - r = 1 - 10 = -9$

Since, for the Midpoint Circle Algorithm of initial point (0, r) & center at origin (0, 0) rotating at clockwise direction, we have

Initial point $(x_0, y_0) = (0, 10)$

$$P_0 = 1 - r = 1 - 10 = -9$$

Case 1 : $P_k < 0$

$$X_{k+1} = X_k + 1$$

$$Y_{k+1} = Y_k$$

$$P_{k+1} = P_k + 2X_{k+1} + 1$$

Case 2: $P_k \geq 0$

$$X_{k+1} = X_k + 1$$

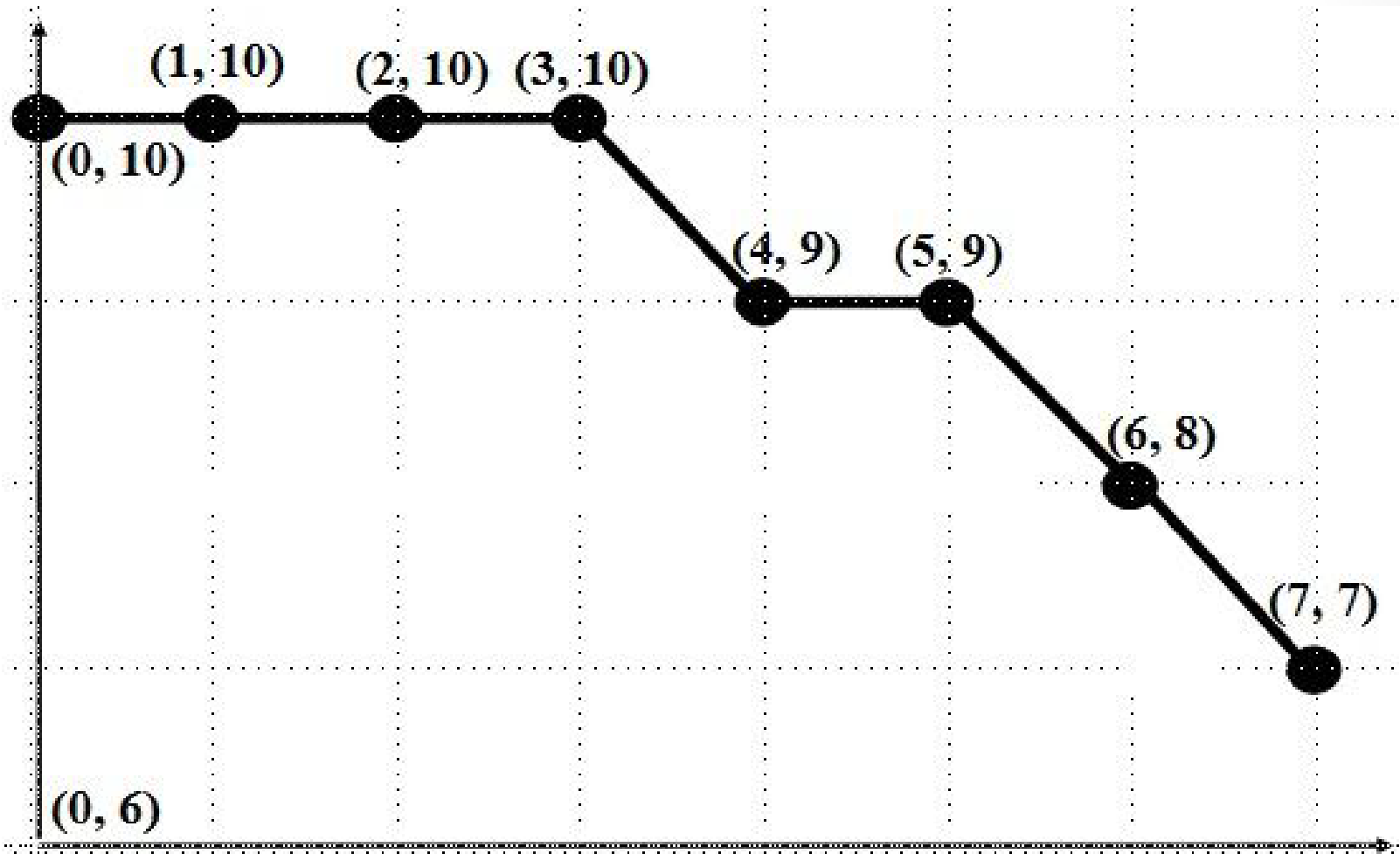
$$Y_{k+1} = Y_k - 1$$

$$P_{k+1} = P_k + 2X_{k+1} - 2Y_{k+1} + 1$$

Thus,

(0 , 10)

K	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-9	$0+1 = 1$	10	(1,10)
1	$= -9 + 2*1 +1 = -6$	$1+1 = 2$	10	(2, 10)
2	$= -6 + 2*2 +1 = -1$	$2+1 = 3$	10	(3, 10)
3	$= -1 + 2*3 +1 = 6$	$3+1 = 4$	$10-1=9$	(4, 9)
4	$= 6 + 2*4 - 2*9 +1 = -3$	$4+1 = 5$	9	(5, 9)
5	$= -3 + 2*5 +1 = 8$	$5+1 = 6$	$9-1=8$	(6, 8)
6	$= 8 + 2*6 - 2*8 +1 = 5$	$6+1 = 7$	$8-1=7$	(7, 7)
7	$= 5 + 2*7 - 2*7 +1 = 6$	$7+1 = 8$	$7-1=6$	(8,6)



Example : Digitize a circle with radius 8

Example: Digitize a circle with radius 9 and center at (6, 7).

Here, the initial decision parameter (P_0)

$$P_0 = 1 - r = 1 - 9 = -8$$

Since, for the Midpoint Circle Algorithm of starting point (0, r) & centre at origin (0, 0) rotating at clockwise direction, we have

If $P < 0$

Plot ($x_k + 1, y_k$)

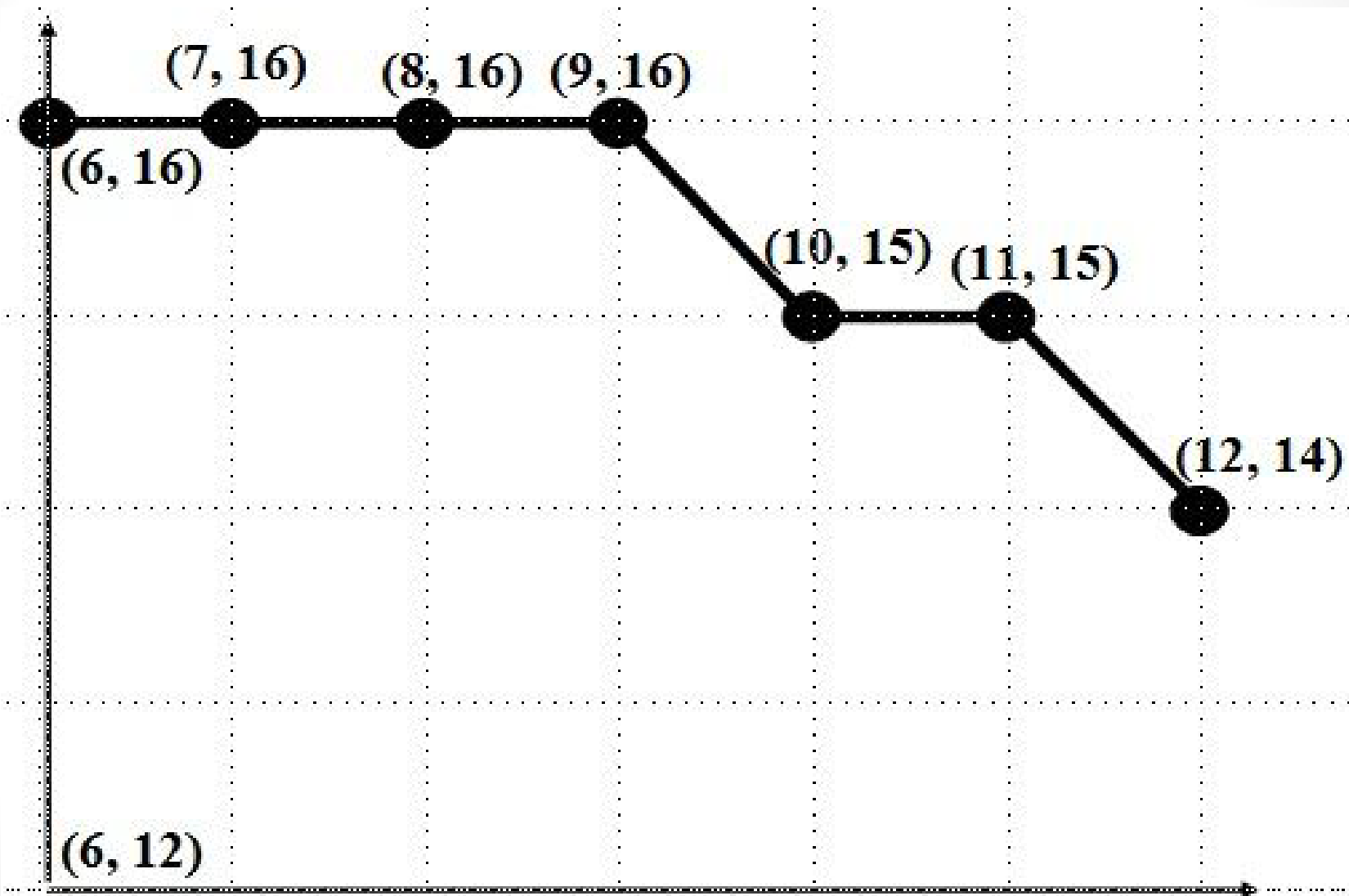
$$P_{k+1} = P_k + 2x_{k+1} + 1$$

Else ($P \geq 0$)

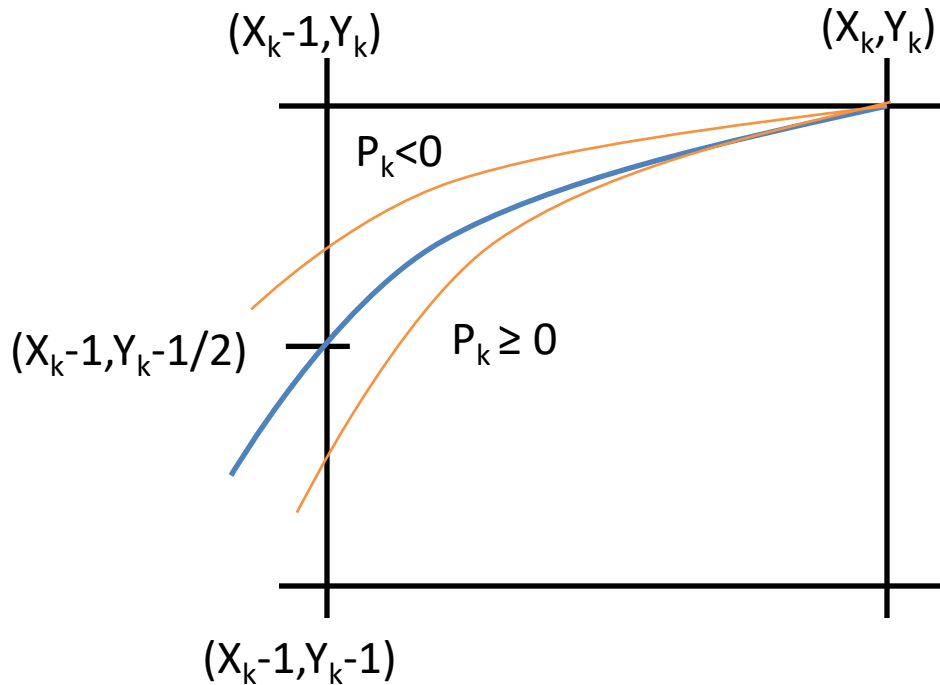
Plot ($x_k + 1, y_k - 1$)

$$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$$

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1}) At (0, 0)	(X_{k+1}, Y_{k+1}) At (6, 7)
0.	-8	$0+1 = 1$	9	(1, 9)	$(1+6, 9+7) = (7, 16)$
1.	$= -8 + 2*1 +1= -5$	$1+1 = 2$	9	(2, 9)	$(2+6, 9+7) = (8, 16)$
2.	$= -5 + 2*2 +1= 0$	$2+1 = 3$	8	(3, 8)	$(3+6, 8+7) = (9, 15)$
3.	$= 0 + 2*3 - 2*8 +1=-9$	$3+1 = 4$	8	(4, 8)	$(4+6, 8+7) = (10,15)$
4.	$= -9 + 2*4 +1= 0$	$4+1 = 5$	7	(5, 7)	$(5+6, 8+7) = (11,15)$
5.	$= 0 + 2*5 - 2*7 +1=-3$	$5+1 = 6$	7	(6, 7)	$(6+6, 7+7) = (12,14)$
6.	$= -3 + 2*6 +1= 10$	$6+1 = 7$	6	(7, 6)	$(7+6, 6+7) = (13,13)$



Mid Point Circle Algorithm



$$P_k < 0$$

$$X_{K+1} = X_k - 1$$

$$Y_{K+1} = Y_k$$

$$P_k \geq 0$$

$$X_{K+1} = X_k - 1$$

$$Y_{K+1} = Y_k - 1$$

Here,

<https://genuinenotes.com>

$$\begin{aligned}\text{Decision parameter}(P_k) &= F_{\text{circle}}(X_k - 1, Y_k - 1/2) \\ &= (X_k - 1)^2 + (Y_k - 1/2)^2 - r^2\end{aligned}$$

Then, $K+1^{\text{th}}$ term is,

$$P_{k+1} = (X_{k+1} - 1)^2 + (Y_{k+1} - 1/2)^2 - r^2$$

Now,

$$\begin{aligned}P_{k+1} - P_k &= (X_{k+1} - 1)^2 + (Y_{k+1} - 1/2)^2 - r^2 - \{(X_k - 1)^2 + (Y_k - 1/2)^2 - r^2\} \\ &= -2(x_k - 1) + (Y_{k+1}^2 - Y_k^2) - (Y_{k+1} - Y_k) + 1\end{aligned}$$

[$\because X_{k+1} = X_k - 1$ in both condition]

$$P_{k+1} = P_k - 2x_{k+1} + (Y_{k+1}^2 - Y_k^2) - (Y_{k+1} - Y_k) + 1$$

Case 1 : $P_k < 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k$$

$$P_{k+1} = P_k - 2X_{k+1} + 1$$

Case 2: $P_k \geq 0$

$$X_{k+1} = X_k - 1$$

$$Y_{k+1} = Y_k + 1$$

$$P_{k+1} = P_k - 2X_{k+1} - 2Y_{k+1} + 1$$

Initial decision parameter

i.e.,

$$(x_0, y_0) = (0, r)$$

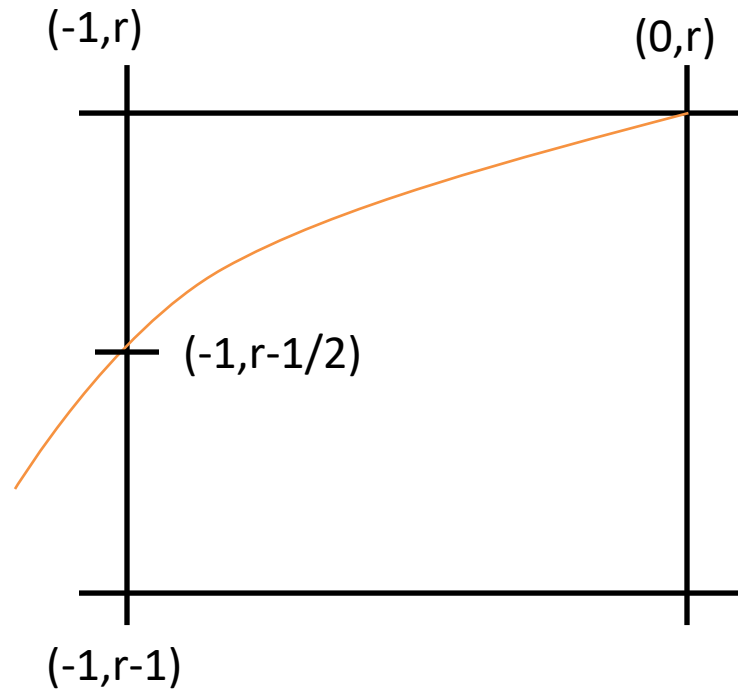
Here ,

$$F_{\text{circle}}(-1, r-1/2) = P_0$$

Now,

$$\begin{aligned} P_0 &= (-1)^2 + (r-1/2)^2 - r^2 \\ &= 1 + r^2 - r + 1/4 - r^2 \\ &= 5/4 - r \end{aligned}$$

$$P_0 \approx 1-r$$



$$P_0 \approx 1-r$$

Ellipse

Midpoint Ellipse Algorithm

Our approach here is similar to that used in displaying a raster circle but the ellipse has 4-way symmetry. The midpoint ellipse method is applied throughout the first quadrant in two parts or region as shown in figure. The region-1 just behaves as the circular property and the region-2 is slightly straight curve.

The equation of ellipse, whose centre at (0, 0) is

$$x^2/a^2 + y^2/b^2 = 1$$

Hence, we define the ellipse function for centre at origin (0, 0) as:

$$F_{\text{ellipse}}(x, y) = x^2b^2 + y^2a^2 - a^2b^2$$

This has the following properties:

- $F_{\text{ellipse}}(x, y) < 0$, if (x, y) is inside the ellipse boundary
- $= 0$, if (x, y) is on the ellipse boundary
- > 0 , if (x, y) is outside the ellipse boundary

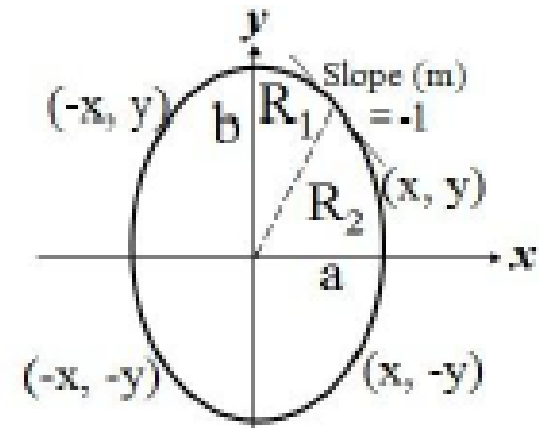


Fig. Ellipse with 4-symmetry & two region in each quadrant

Midpoint Ellipse Algorithm (cont..)

Starting at **(0, b)**, we take unit steps in the **x** direction until we reach the boundary between region 1 and region 2. Then we switch to unit steps in the **y** direction over the remainder of the curve in the first quadrant. At each step, we need to test the value of the slope of the curve. The ellipse slope is calculated by differentiating the ellipse function as:

$$2xb^2 + 2ya^2 * dy/dx = 0 \text{ Or } dy/dx = - 2xb^2 / 2ya^2$$

At the boundary between region 1 and region 2, **dy/dx = - 1** and $2xb^2 = 2ya^2$. Therefore, we move out of region **1** whenever $2xb^2 \geq 2ya^2$.

For Region – 1: Condition ($2xb^2 \geq 2ya^2$)

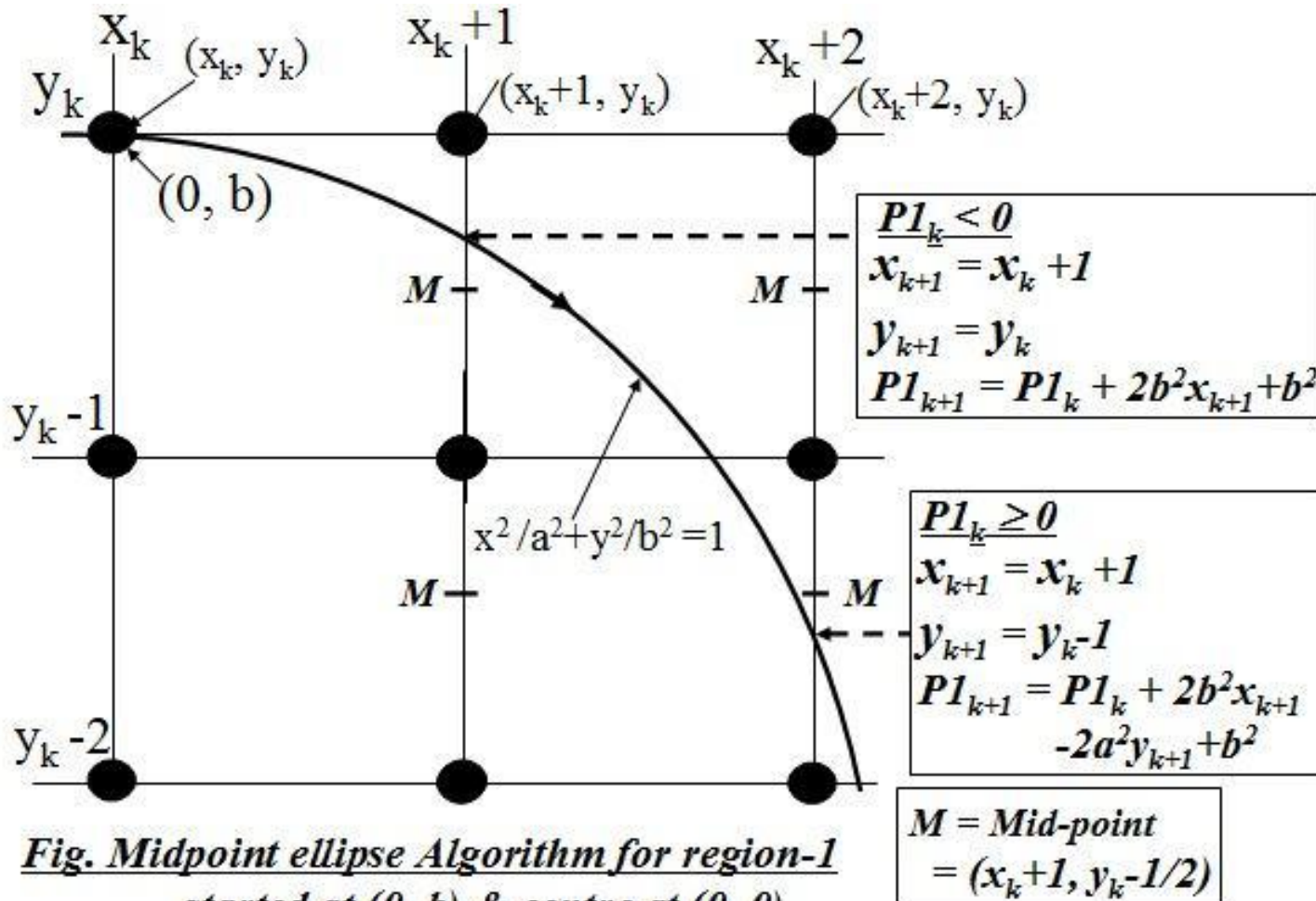


Fig. Midpoint ellipse Algorithm for region-1
started at $(0, b)$ & centre at $(0, 0)$

Midpoint Ellipse Algorithm (cont..)

Assuming that the position (x_k, y_k) has been plotted, we determine next position (x_{k+1}, y_{k+1}) as either (x_{k+1}, y_k) or (x_{k+1}, y_{k-1}) by evaluating the decision parameter $P1_k$

as:

$$P1_k = F_{\text{ellipse}}(x_k+1, y_{k-1/2}) \\ = b^2 (x_k+1)^2 + a^2 (y_{k-1/2})^2 - a^2 b^2 \text{ ---- I}$$

At next sampling position, the decision parameter will be

$$P1_{k+1} = F_{\text{ellipse}}(x_{k+1}+1, y_{k+1}-1/2) \\ = b^2 (x_{k+1}+1)^2 + a^2 (y_{k+1}-1/2)^2 - a^2 b^2 \\ = b^2 \{(x_k+1) + 1\}^2 + a^2 (y_{k+1}-1/2)^2 - a^2 b^2$$

Now, subtracting I from II, we have

$$P_{k+1} - P_k = 2b^2(x_k+1) + a^2[(y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k)] + b^2$$

If $P_k < 0$, the midpoint is inside the ellipse and the pixel on the scan line y_k closer to the ellipse boundary. Otherwise, the midpoint is outside or on the ellipse boundary, and we select the pixel on scan line y_{k-1} .

Case – I: $P_k < 0$

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

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

$$P_{k+1} = P_k + 2b^2(x_{k+1}) + b^2$$

$$P_{k+1} = P_k + 2b^2x_{k+1} + b^2$$

Case – I: $P_k \geq 0$

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

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

$$P_{k+1} = P_k + 2b^2(x_{k+1}) + a^2[(y_{k+1} - 1)^2 - y_k^2] - a^2(y_{k+1} - 1 - y_k) + b^2$$

$$= P_k + 2b^2(x_{k+1}) + a^2[(y_k^2 - 2y_{k+1} + 1 - y_k^2)] - a^2(y_{k+1} - 1 - y_k) + b^2$$

$$= P_k + 2b^2(x_{k+1}) - 2a^2y_k + a^2 + a^2 + b^2$$

$$= P_k + 2b^2(x_{k+1}) - 2a^2(y_{k+1}) + b^2$$

$$P_{k+1} = P_k + 2b^2x_{k+1} - 2a^2y_{k+1} + b^2$$

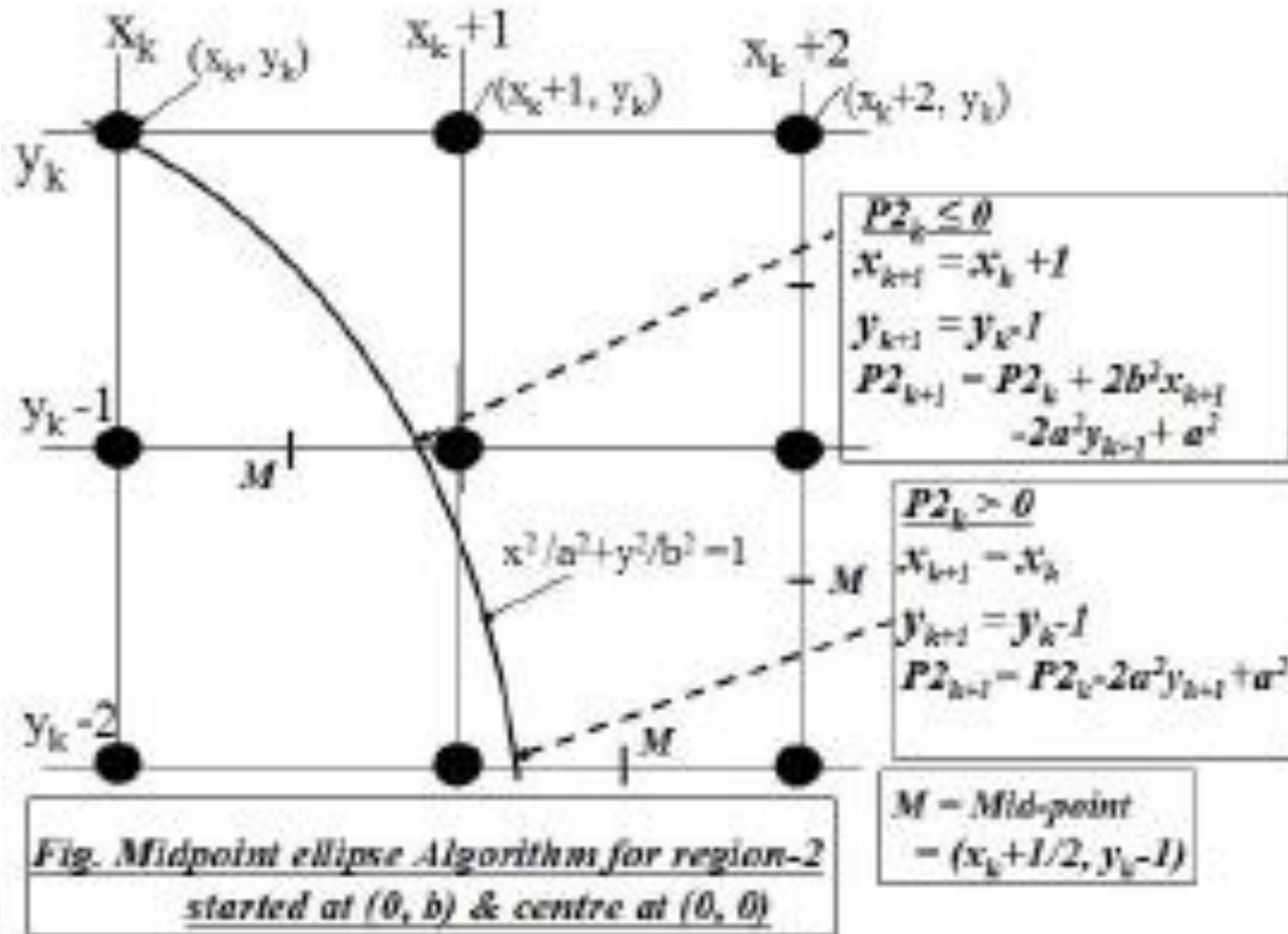
Initial decision parameter ($P1_0$) for region-1 of ellipse

The initial decision parameter is obtained by evaluating the ellipse function at the start position $(x_0, y_0) = (0, b)$. Here, the next pixel will either be $(1, b)$ or $(1, b-1)$ where the midpoint is $(1, b - 1/2)$. Thus, the initial decision parameter is given by:

$$\begin{aligned} P1_0 &= F_{\text{ellipse}}(1, b-1/2) = b^2 + a^2 (b - 1/2)^2 - a^2 b^2 \\ &= b^2 - a^2 b^2 + a^2 * 1/4 \end{aligned}$$

$$\text{Thus, } P1_0 = b^2 - a^2 b^2 + a^2 * 1/4$$

For Region – 2: Condition ($2x^2 < 2ya^2$)



Assuming that the position (x_k, y_k) has been plotted, we determine next position (x_{k+1}, y_{k+1}) as either (x_{k+1}, y_{k-1}) or (x_k, y_{k-1}) by evaluating the decision parameter $P2_k$ as:

$$\begin{aligned} P2_k &= F_{\text{ellipse}}(x_{k+1}/2, y_{k-1}) \\ &= b^2 (x_{k+1}/2)^2 + a^2 (y_{k-1})^2 - a^2 b^2 \quad \text{--- I} \end{aligned}$$

At next sampling position, the decision parameter will be

$$\begin{aligned} P2_{k+1} &= F_{\text{ellipse}}(x_{k+1}+1/2, y_{k+1}-1) \\ &= b^2 (x_{k+1}+1/2)^2 + a^2 (y_{k+1}-1)^2 - a^2 b^2 \\ &= b^2 (x_{k+1} + 1/2)^2 + a^2 \{(y_{k-1}) - 1\}^2 - a^2 b^2 \\ &= b^2 (x_{k+1} + 1/2)^2 + a^2 \{(y_{k-1})^2 - 2(y_{k-1}) \\ &\quad + 1\} - a^2 b^2 \end{aligned}$$

$$P2_{k+1} - P2_k = b^2 [(x_{k+1})^2 - x_k^2] + (x_{k+1} - x_k) - 2a^2 (y_{k-1}) + a^2$$

If $P_k < 0$, the midpoint is inside the ellipse and the pixel on the scan line x_k is closer to the ellipse boundary. Otherwise, the midpoint is outside or on the ellipse boundary, and we select the pixel on scan line x_{k+1} .

Case - I: $P2_k \leq 0$

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

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

$$\begin{aligned} P2_{k+1} &= P2_k + b^2 [(x_{k+1})^2 - x_k^2] + (x_{k+1} - x_k) - 2a^2 (y_{k-1}) + a^2 \\ &= P2_k + b^2 [(x_k^2 + 2x_k + 1 - x_k^2) + (x_{k+1} - x_k)] - 2a^2 (y_{k-1}) + a^2 \\ &= P2_k + b^2 [2x_k + 1] - 2a^2 (y_{k-1}) + a^2 \\ &= P2_k + 2b^2 x_{k+1} - 2a^2 (y_{k-1}) + a^2 \\ P2_{k+1} &= P2_k + 2b^2 x_{k+1} - 2a^2 y_{k+1} + a^2 \end{aligned}$$

Case - I: $P_k \geq 0$

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

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

$$\begin{aligned} P2_{k+1} &= P2_k - 2a^2 (y_{k-1}) + a^2 \\ P_{k+1} &= P2_k - 2a^2 y_{k+1} + a^2 \end{aligned}$$

Initial decision parameter ($P2_0$) for region-2 of ellipse

When we enter region 2, the initial position (x_0, y_0) is taken as the last position selected in region 1 and the initial decision parameter in region 2 is given by:

$$P2_0 = F_{\text{ellipse}}(x_0 + 1/2, y_0 - 1)$$

$$P2_0 = b^2(x_0 + 1/2)^2 + a^2(y_0 - 1)^2 - a^2 b^2$$

Note: After the calculation of all the points in one quadrant, determine the symmetry points in the other three quadrants as shown in figure.

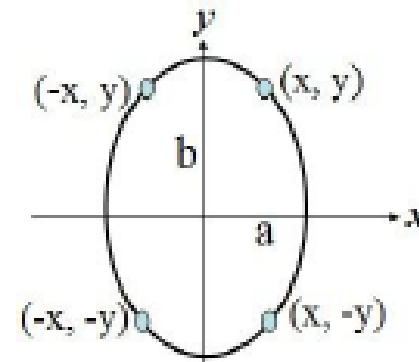


Fig. Ellipse with 4-symmetry

Example: Given input ellipse parameters $r_x = a = 8$ and $r_y = b = 6$, we illustrate the steps in the midpoint ellipse algorithm by determining raster positions along the ellipse path in the first quadrant.

$$\begin{aligned} 2b^2x &= 0 && \text{(with increment } 2b^2 = 72) \\ 2a^2y &= 2a^2b && \text{(with increment } -2a^2 = -128) \end{aligned}$$

For region-1

The initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 6)$, and the initial decision parameter value is: $P_{10} = b^2 - a^2b^2 + a^2/4 = -332$

Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as:

k	P_{1k}	$(X_{k+1}, Y_{k+1})_{At (0,0)}$	$2b^2X_{k+1}$	$2a^2Y_{k+1}$
0.	-332	(1, 6)	72	768
1.	-224	(2, 6)	144	768
2.	-44	(3, 6)	216	768
3.	208	(4, 5)	288	640
4.	-108	(5, 5)	360	640
5.	288	(6, 4)	432	512
6.	244	(7, 3)	540	384

We now move out of region 1, since $2b^2X > 2a^2Y$

For region-2

The initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is:

$$P2_0 = b^2(x_0 + 1/2)^2 + a^2(y_0 - 1)^2 - a^2b^2 = -151$$

The remaining positions along the ellipse path in the first quadrant are then calculated as:

k	$P1_k$	$(X_{k+1}, Y_{k+1})_{At (0,0)}$	$2b^2X_{k+1}$	$2a^2Y_{k+1}$
0.	-151	(8, 2)	576	256
1.	233	(8, 1)	576	128
2.	748	(8, 0)	-	-

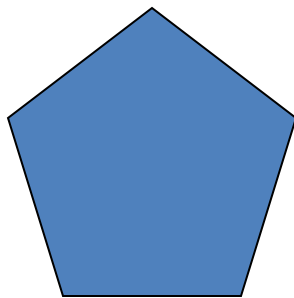
Fill Area Algorithms

Unit 1

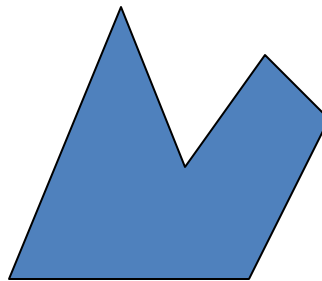
Polygon Fill Algorithm

- ***Different types of Polygons***

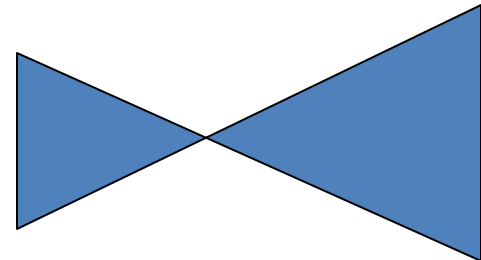
- Simple Convex
- Simple Concave
- Non-simple : self-intersecting
- With holes



Convex



Concave



Self-intersecting

Area Fill Algorithm

- An alternative approach for filling an area is to start at a point inside the area and “paint” the interior, point by point, out to the boundary.
- This is a particularly useful technique for filling areas with irregular borders, such as a design created with a paint program.
- The algorithm makes the following assumptions
 - one interior pixel is known, and
 - pixels in boundary are known.

Area Fill Algorithm

- If the boundary of some region is specified in a single color, we can fill the interior of this region, pixel by pixel, until the boundary color is encountered.
- This method, called the **boundary-fill algorithm**, is employed in interactive painting packages, where interior points are easily selected.

Example

- One can sketch a figure outline, and pick an interior point.
- The figure interior is then painted in the fill color as shown in these Figures

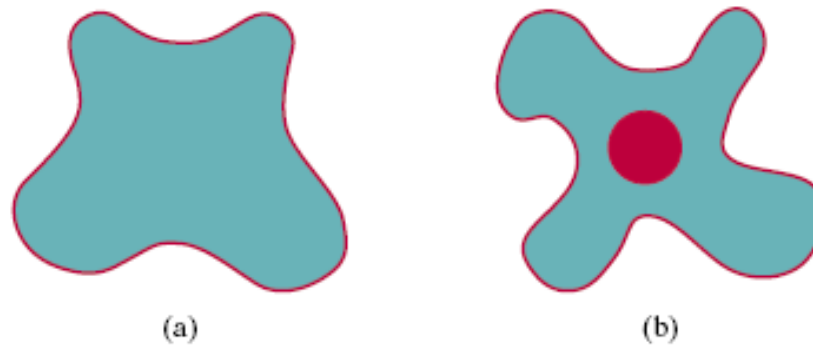


FIGURE 4-26 Example color boundaries for a boundary-fill procedure.

Area Fill Algorithm

- Basically, a boundary-fill algorithm starts from an interior point (x, y) and sets the neighboring points to the desired color.
- This procedure continues until all pixels are processed up to the designated boundary for the area.

Area Fill Algorithm

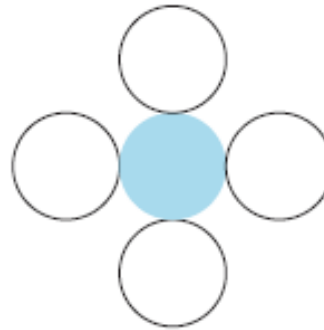
- There are two methods for processing neighboring pixels from a current point.
 1. Four neighboring points.
 - These are the pixel positions that are right, left, above, and below the current pixel.
 - Areas filled by this method are called **4-connected**.

Area Fill Algorithm

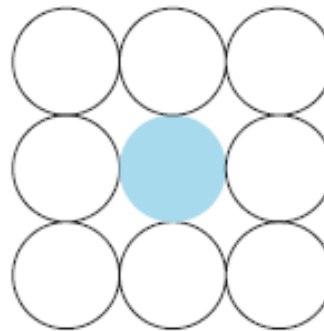
2. Eight neighboring points.

- This method is used to fill more complex figures.
- Here the set of neighboring points to be set includes the four diagonal pixels, in addition to the four points in the first method.
- Fill methods using this approach are called **8-connected**.

Area Fill Algorithm



(a)



(b)

FIGURE 4-27 Fill methods applied to a 4-connected area (a) and to an 8-connected area (b). Hollow circles represent pixels to be tested from the current test position, shown as a solid color.

Area Fill Algorithm

- Consider the Figure in the next slide.
- An 8-connected boundary-fill algorithm would correctly fill the interior of the area defined in the Figure.
- But a 4-connected boundary-fill algorithm would only fill part of that region.

Area Fill Algorithm

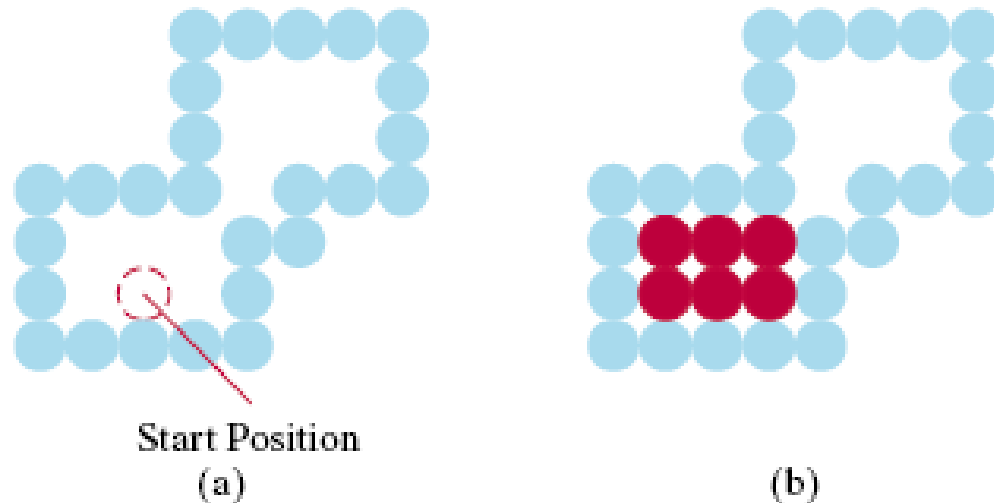


FIGURE 4-28 The area defined within the color boundary (a) is only partially filled in (b) using a 4-connected boundary-fill algorithm.

Area Fill Algorithm

- The following procedure illustrates a recursive method for painting a 4-connected area with a solid color, specified in parameter **fillColor**, up to a boundary color specified with parameter **borderColor**.
- We can extend this procedure to fill an 8-connected region by including four additional statements to test the diagonal positions $(x \pm 1, y \pm 1)$.

Area Fill Algorithm

```
void boundaryFill4 (int x, int y, int fillColor, int borderColor)
{
    int interiorColor;

    /* Set current color to fillColor, then perform following oprations. */
    getPixel (x, y, interiorColor);
    if ((interiorColor != borderColor) && (interiorColor != fillColor)) {
        setPixel (x, y);    // Set color of pixel to fillColor.
        boundaryFill4 (x + 1, y , fillColor, borderColor);
        boundaryFill4 (x - 1, y , fillColor, borderColor);
        boundaryFill4 (x , y + 1, fillColor, borderColor);
        boundaryFill4 (x , y - 1, fillColor, borderColor)
    }
}
```

Area Fill Algorithm

- Some times we want to fill in (or recolor) an area that is not defined within a single color boundary.
- Consider the following Figure.

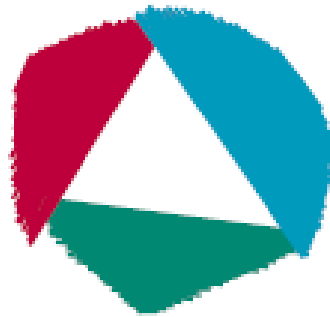


FIGURE 4-30 An area defined within multiple color boundaries.

Area Fill Algorithm

- We can paint such areas by replacing a specified interior color instead of searching for a particular boundary color.
- This fill procedure is called a **flood-fill algorithm**.

Area Fill Algorithm

- We start from a specified interior point (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.
- If the area we want to paint has more than one interior color, we can first reassign pixel values so that all interior points have the same color.

Area Fill Algorithm

- Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted.
- The following procedure flood fills a 4-connected region recursively, starting from the input position.

Area Fill Algorithm

```
void floodFill4 (int x, int y, int fillColor, int interiorColor)
{
    int color;

    /* Set current color to fillColor, then perform following operations. */
    getPixel (x, y, color);
    if (color == interiorColor) {
        setPixel (x, y);    // Set color of pixel to fillColor.
        floodFill4 (x + 1, y, fillColor, interiorColor);
        floodFill4 (x - 1, y, fillColor, interiorColor);
        floodFill4 (x, y + 1, fillColor, interiorColor);
        floodFill4 (x, y - 1, fillColor, interiorColor)
    }
}
```

Problems with Fill Algorithm (1)

- Recursive boundary-fill algorithms may not fill regions correctly if some interior pixels are already displayed in the fill color.
- This occurs because the algorithm checks next pixels both for boundary color and for fill color.

Problems with Fill Algorithm

- To avoid this, we can first change the color of any interior pixels that are initially set to the fill color before applying the boundary-fill procedure.
- Encountering a pixel with the fill color can cause a recursive branch to terminate, leaving other interior pixels unfilled.

Problems with Fill Algorithm (2)

- This procedure requires considerable stacking of neighboring points, more efficient methods are generally employed.
- These methods fill horizontal pixel spans across scan lines, instead of proceeding to 4-connected or 8-connected neighboring points.

Problems with Fill Algorithm (2)

- Then we need only stack a beginning position for each horizontal pixel span, instead of stacking all unprocessed neighboring positions around the current position.
- Starting from the initial interior point with this method, we first fill in the contiguous span of pixels on this starting scan line.

Problems with Fill Algorithm (2)

- Then we locate and stack starting positions for spans on the adjacent scan lines, where spans are defined as the contiguous horizontal string of positions bounded by pixels displayed in the border color.
- At each subsequent step, we retrieve the next start position from the top of the stack and repeat the process.

Area Fill Algorithm

The algorithm can be summarized as follows:

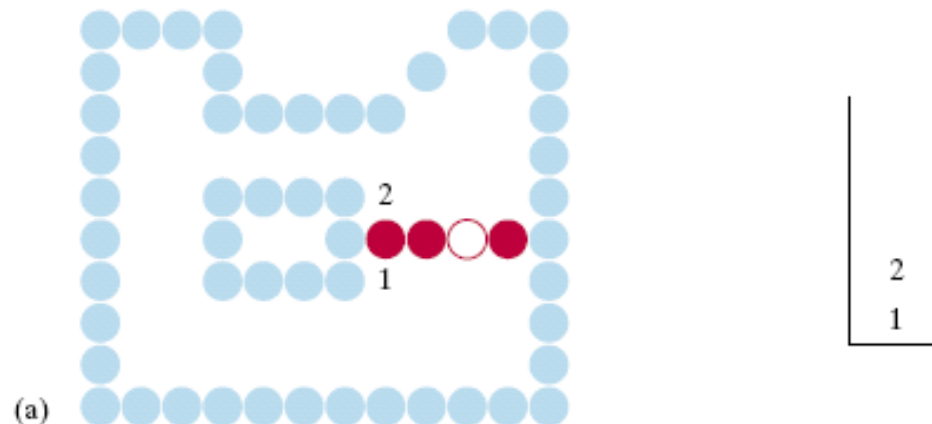
1. define seed point,
2. fill scan line containing seed point,
3. for scan lines above and below, define new seed points as:
 - i) first point inside left boundary,
 - ii) subsequent points within boundary whose left neighbor is outside,
4. d) repeat algorithm with the new set of seed points.

Example

- In this example, we first process scan lines successively from the start line to the top boundary.
- After all upper scan lines are processed, we fill in the pixel spans on the remaining scan lines in order down to the bottom boundary.
- The leftmost pixel position for each horizontal span is located and stacked, in left to right order across successive scan lines.

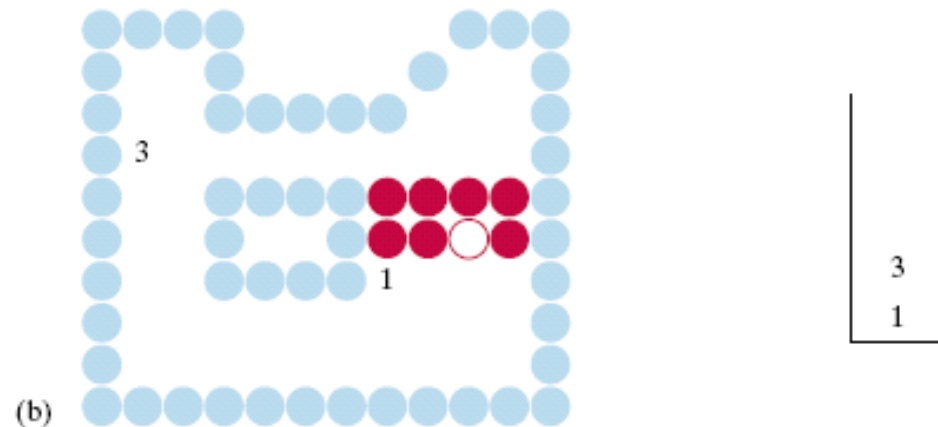
Example

- In (a) of this figure, the initial span has been filled, and starting positions 1 and 2 for spans on the next scan lines (below and above) are stacked.



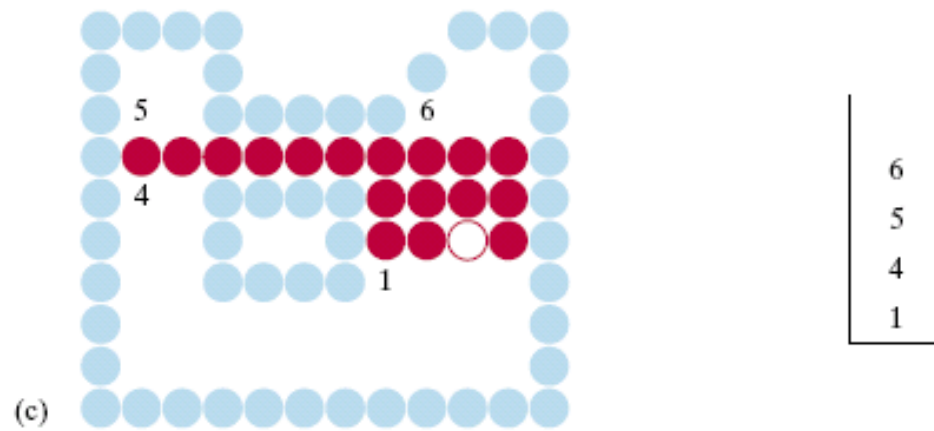
Example

- In Fig.(b), position 2 has been unstacked and processed to produce the filled span shown, and the starting pixel (position 3) for the single span on the next scan line has been stacked.



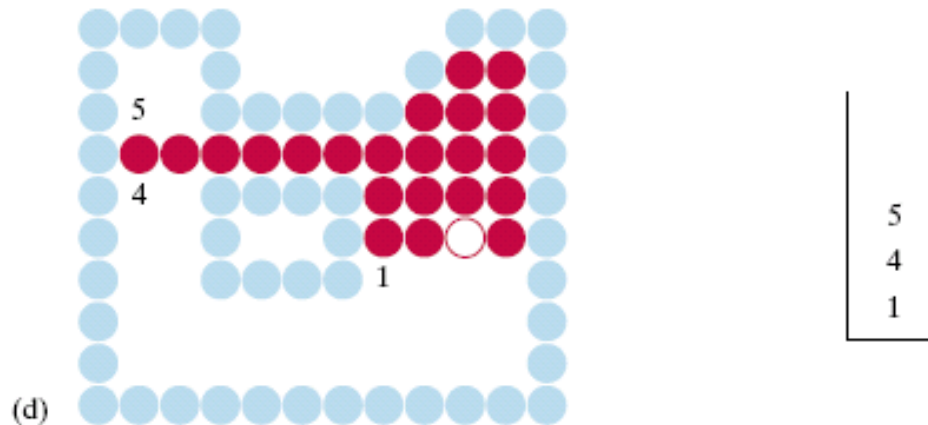
Example

- After position 3 is processed, the filled spans and stacked positions are as shown in Fig. (c).



Example

- And Fig.(d) shows the filled pixels after processing all spans in the upper right of the specified area.

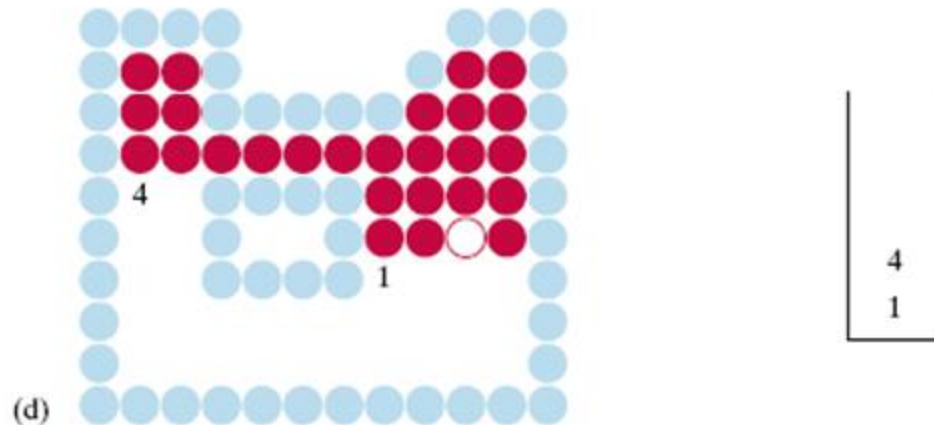


115

-
- (d)

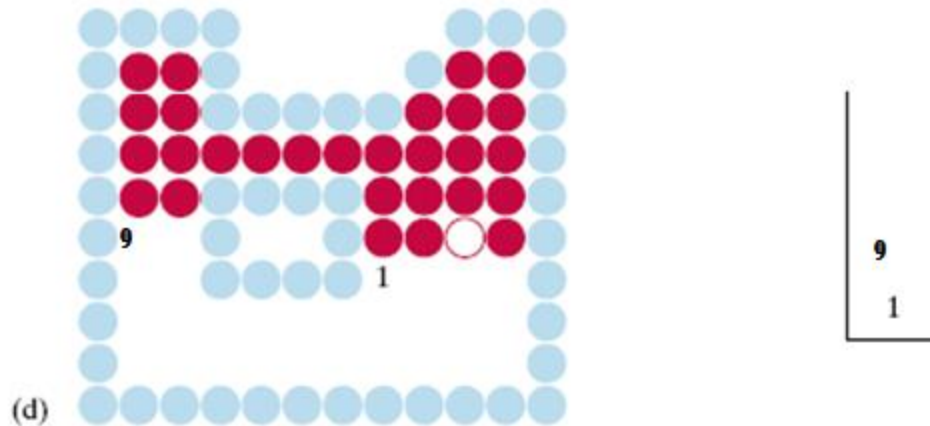
Example

- Finish up the upper scan lines.



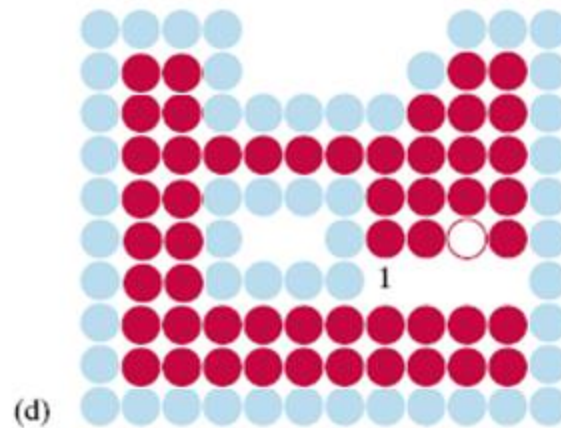
Example

- Start the bottom scan lines.



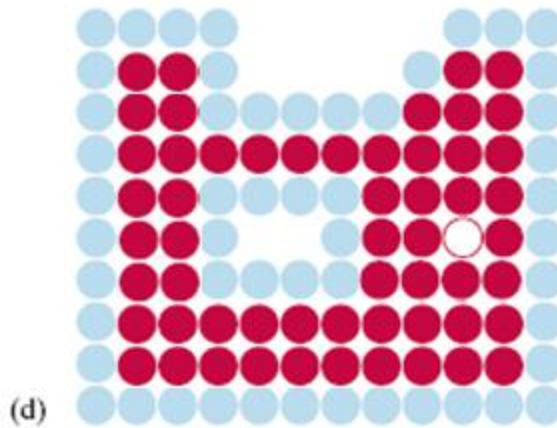
Example

- Finish up the bottom scan lines.



Example

- Finish up the bottom scan lines.



Chapter 2

Finished