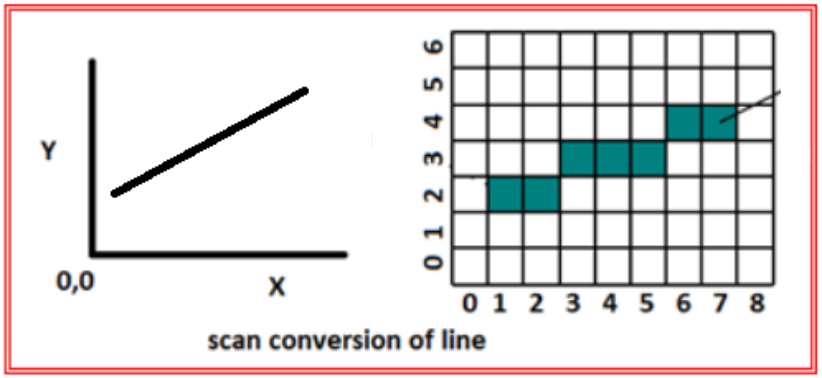
Chapter 2

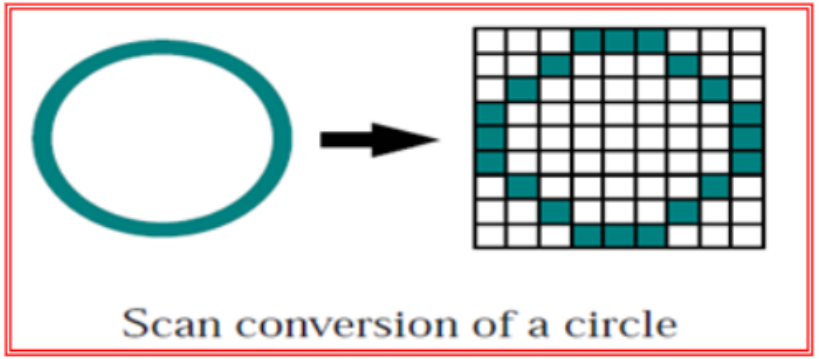
Scan Conversion

What is scan conversion?

- The process of representing continuous graphics objects as a collection of discrete pixels is called **scan conversion**.(Also called **rasterization**)

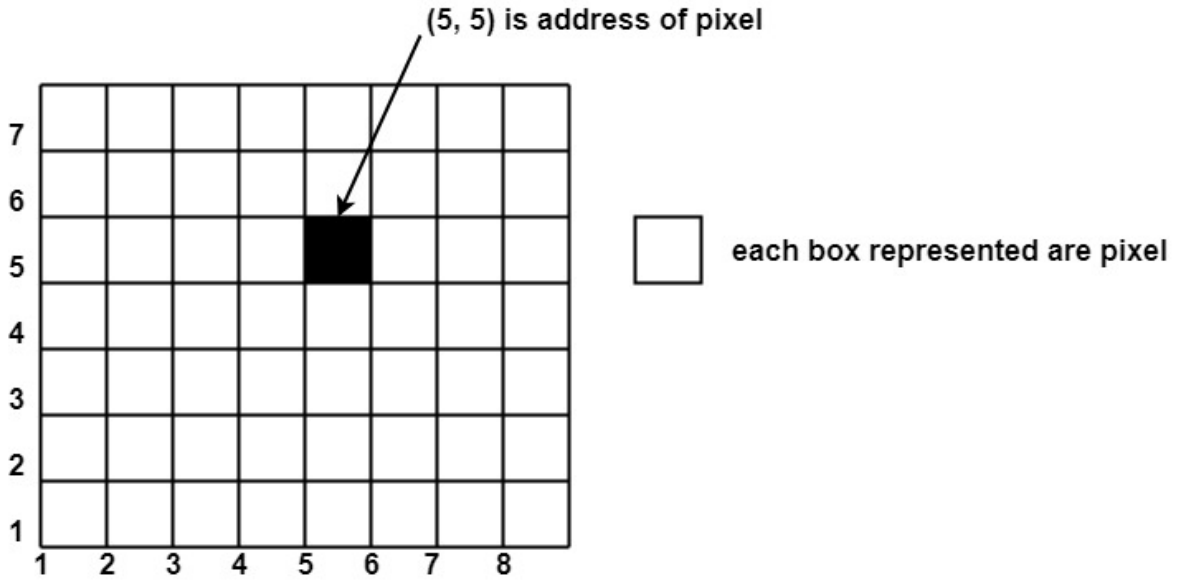


scan conversion of line


Scan conversion of a circle

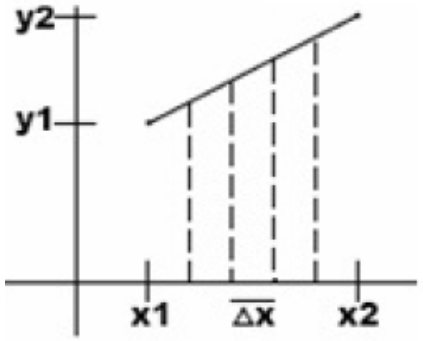
Scan Converting a point

- A point can in represented in a raster screen by just illuminating corresponding pixel.



Scan Converting a straight line

- On raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations.
- Idea: A line is sampled at unit intervals in one coordinate and the corresponding integer values nearest the line path are determined for the other coordinate.
- We have different algorithm for computation



DDA (Digital Differential Analyzer) Algorithm

- DDA algorithm is an incremental scan-conversion method. Such an approach is characterized by performing calculations at each step using results from the preceding step.
- A line is sampled at unit intervals in one coordinate and the corresponding integer values nearest the line path are determined for the other coordinate.
- Straight Line equations

Slope-Intercept Equation

$$y = m.x + b$$

Slope

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Interval Calculation

$$\Delta y = m.\Delta x \qquad \Delta x = \frac{\Delta y}{m}$$

DDA (Digital Differential Analyzer) Algorithm

- Main Calculation

$$x_{k+1} = x_k + \Delta x$$

$$y_{k+1} = y_k + \Delta y$$

- Four Different cases

Case 1: line with positive slope and magnitude less than 1 (*/m/<1*)

Case 2: line with positive slope and magnitude more than 1 (*/m/>1*)

Case 3: line with negative slope and magnitude less than 1 (*/m/<1*)

Case 4: line with negative slope and magnitude more than 1 (*/m/>1*)

DDA (Digital Differential Analyzer) Algorithm

- **Considering line with a positive slope:**

Case I: If slope (m) <=1 then sample at unit x intervals' (Δx=1) and compute each successive y value because the increment in x is more than increment in y.

So, set,

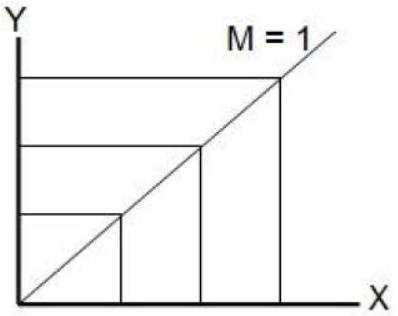
Δx=1 So, Δy=m

i.e.

$x_{k+1}=x_k+\Delta x=x_k+1$

$y_{k+1}=y_k+\Delta y=y_k+m$

$$\Delta y = m.\Delta x \qquad \Delta x = \frac{\Delta y}{m}$$



Case II: If m > 1, then the increment in x (i.e. Δx) is smaller than increment in y (i.e. Δy),

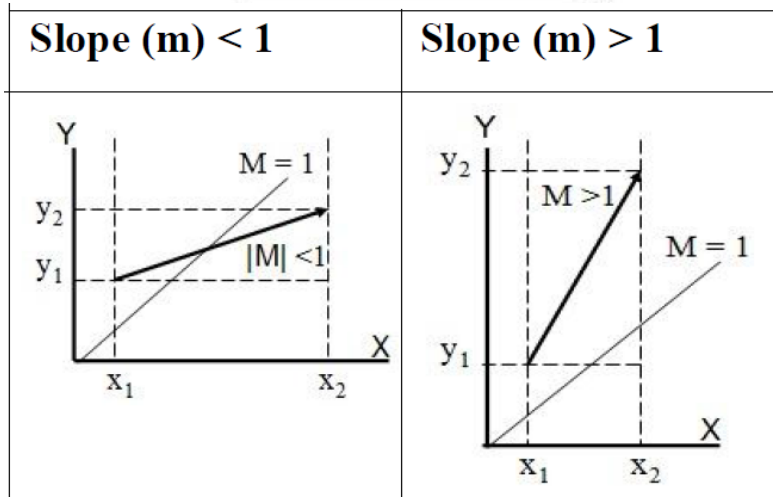
So set,

Δy=1 So, Δx=1/m

i.e.

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

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



DDA (Digital Differential Analyzer) Algorithm

- Here, in the both case m < = 1 and m>1, the algorithm based on the assumption that lines are to be processed from the left end point to the right end point. If this process is reversed, relation required to change in both case.

Case I: if $m<=1$, $\Delta x=-1$, so, $\Delta y=-m$

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

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

Case II: if $m>1$, $\Delta y=-1$ and $\Delta x=-1/m$

$x_{k+1}=x_k-1/m$

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

DDA (Digital Differential Analyzer) Algorithm

- **Considering line with a negative slope:**

Case I: If / m/< = 1 then assume start end point is the left then Δx=1 and Δy = m (m is negative).

i.e.

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

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

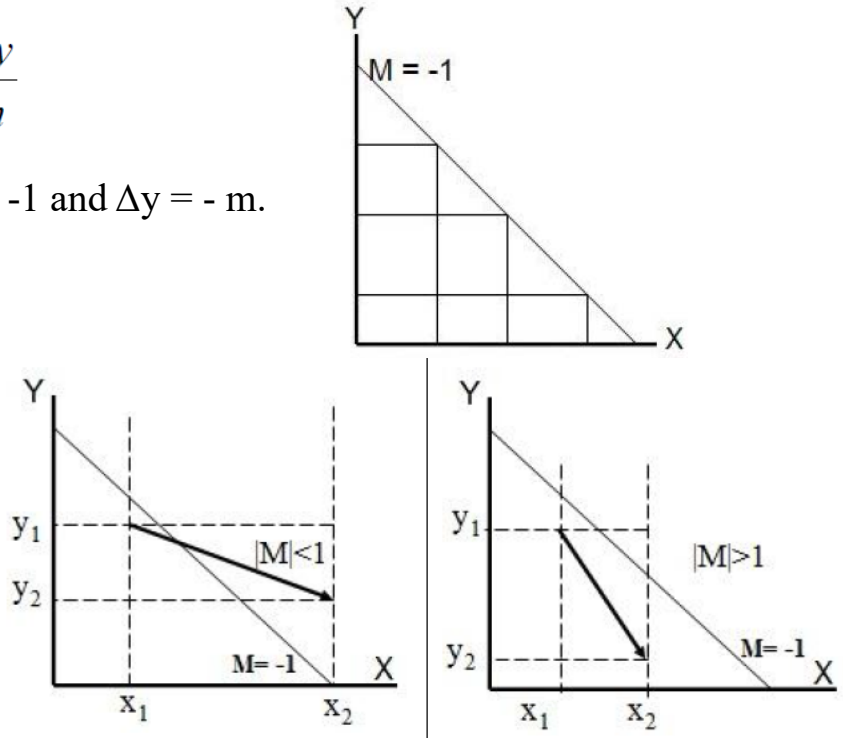
$$\Delta y = m.\Delta x \qquad \Delta x = \frac{\Delta y}{m}$$

If algorithm is required to proceed right to left then set on Δx= -1 and Δy = - m.

i.e.

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

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



DDA (Digital Differential Analyzer) Algorithm

- **Considering line with a negative slope:**

Case II: If / m/ > 1 then assume start end is at left and set $\Delta y = -1$ and $\Delta x = 1/m$ (m is negative)

i.e.

$x_{k+1}=x_k-1/m$

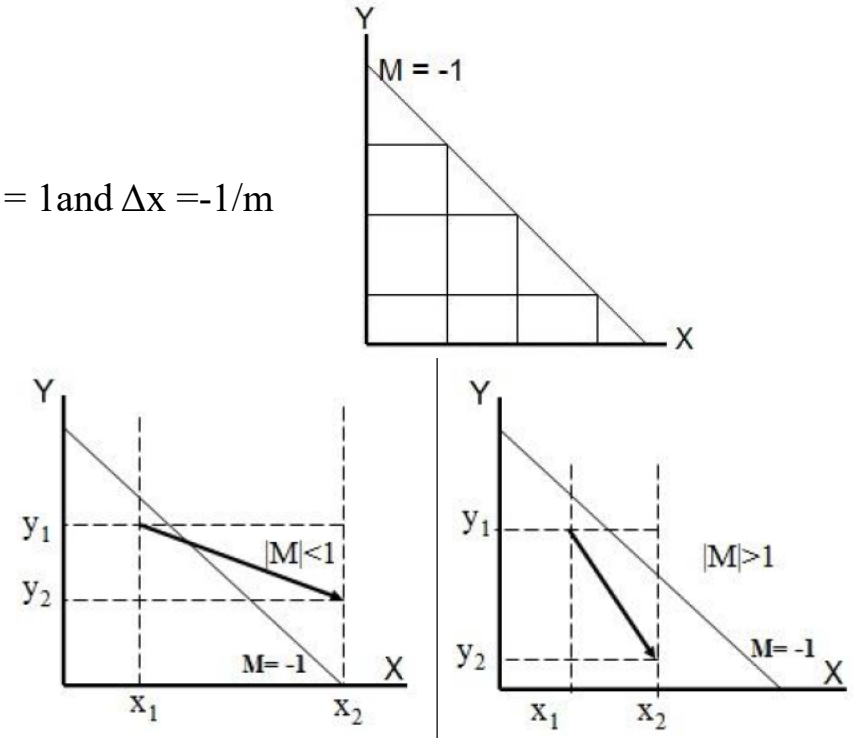
$y_{k+1}=y_k-1$

If the algorithm is required to proceed right to left then set $\Delta y = 1$and $\Delta x = -1/m$

i.e.

$x_{k+1}=x_k \quad +1/m$

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



DDA (Digital Differential Analyzer) Algorithm

1. Input points (x1,y1) and (x2,y2)
2. Compute dx = x2- x1 & dy= y2- y1
3. If |dx|>|dy| then steps=|dx|

   otherwise steps =|dy|
4. x_{inc} = dx/ steps

   y_{inc} = dy/ steps
5. x= x1 , y= y1 , k=0
6. do:

   plot(round (x), round (y));

   x=x+ x_{inc}

   y=y+ x_{inc}

   k ++

while (k<=steps)

DDA (Digital Differential Analyzer) Algorithm

Advantages

- Faster method than simple line drawing algorithm for calculating pixel position as it eliminates multiplication.
- Avoids directly deal of equation of st. line $y = mx+c$.

Disadvantages

- 'm' is usually stored in floating point number.
- There could be round off error.
- The line will move away from the true line path, especially when it is long due to successive round of error.
- Accumulation of round off error in successive additions of floating point increment.

DDA (Digital Differential Analyzer) Algorithm

Example-1: Digitize the line with end points (2, 1) and (8, 3) using DDA.

Solution

Here,

Starting point of line = (x1, y1) = (2, 1) and

Ending point of line = (x2, y2) = (8, 3)

Thus, slope of line, m = Δy / Δx = *y2-y1 / x2-x1* = (3-1) / (8-2) = 1/3 =0.3333

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

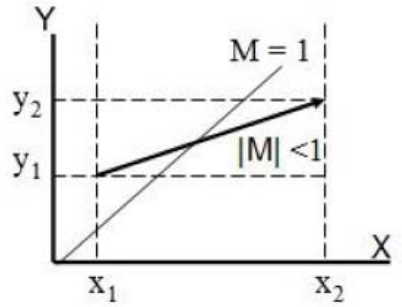
m = 1/3 <1

Thus,

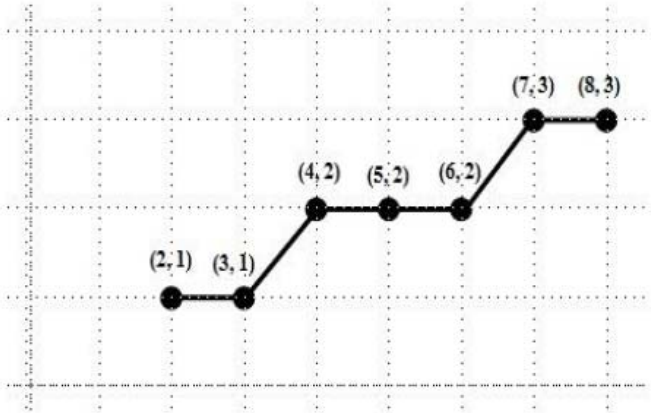
Δx=1
Δy=m

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

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



X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
2	1	(2,1)
2+1 = 3	1+0.3333 = 1.333 ≈ 1	(3, 1)
3+1 = 4	1+2 *0.3333 = 1.666 ≈ 2	(4, 2)
4+1 = 5	1+3*0.3333 = 1.999 ≈ 2	(5, 2)
5+1 = 6	1+4*0.3333 = 2.333 ≈ 2	(6, 2)
6+1 = 7	1+5*0.3333 = 2.666 ≈ 3	(7, 3)
7+1 = 8	1+6*0.3333 = 2.999 ≈ 3	(8, 3)



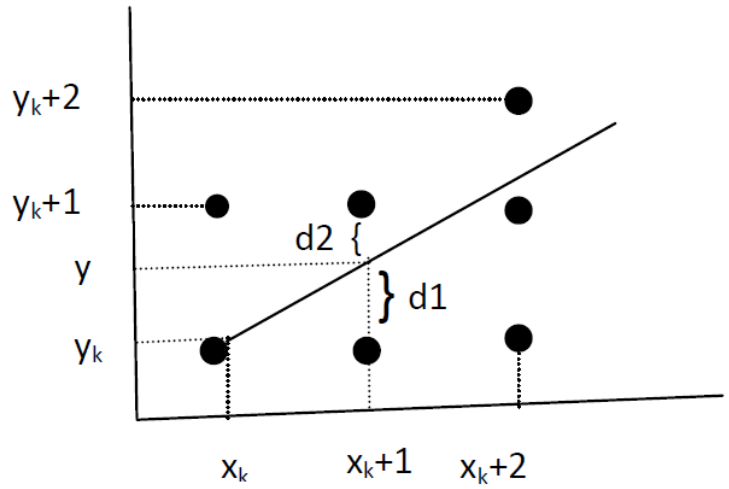
Bresenham's Line Algorithm

- Accurate and efficient line drawing algorithm.
- Brenham algorithm use only integer arithmetic to find the next position to be plot.
- It avoids incremental error.
- The major concept of Bresenham algorithm is to determine the nearest pixel position.

Bresenham's Line Algorithm

- Pixel positions are determined by sampling at unit x intervals.
- - Starting from left end position (x_0, y_0) of a given line, we step to each successive column (x-position) and plot the pixel whose scan line y value is closed to the line path.
- - Assuming the pixel at (x_k, y_k) to be displayed is determined, we next to decide which pixel to plot in column x_k +1, our choices are the pixels at positions.

(x_k + 1, y_k) and (x_k + 1, y_k + 1)



Bresenham's Line Algorithm

- At sampling position $x_k + 1$, we label vertical pixel separations from the mathematical line path d1 and d2.
- The y-co-ordinate on the mathematical at pixel column position $x_k + 1$ is calculated as,

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

Then,

$$d1=y-y_k =m(x_k+1)+b-y_k.....(2)$$

and

$$d2=(y_k+1)-y = y_k+1-m(x_k+1)-b.....(3)$$

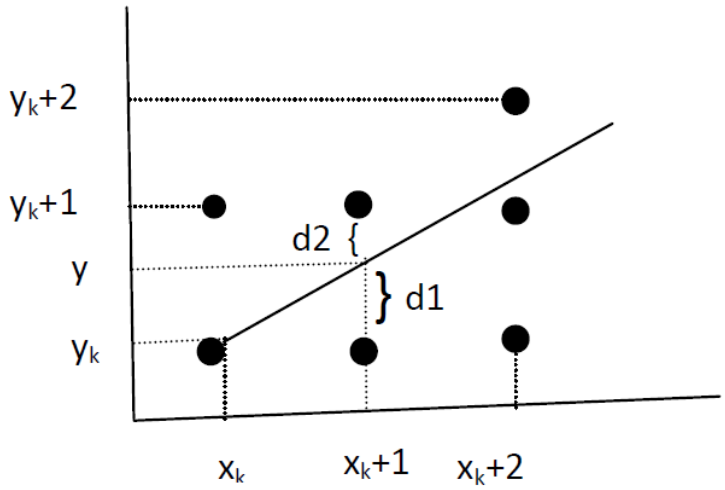
now,

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

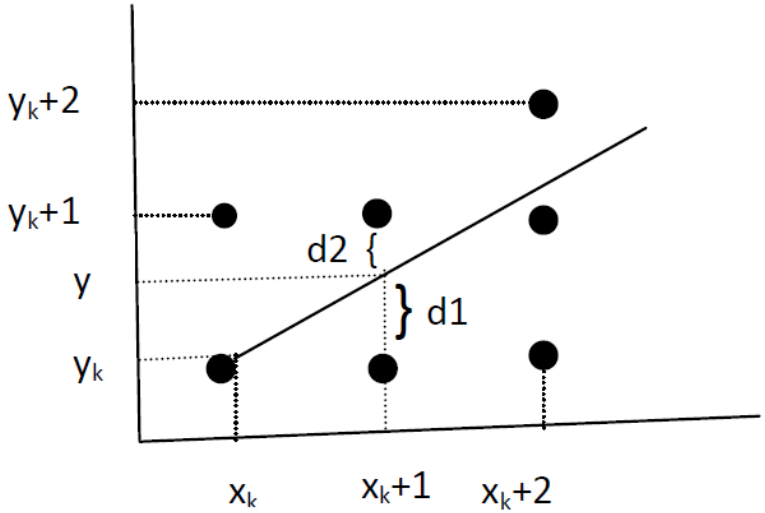
$$=2*m(x_k+1)-2y_k+2b-1$$

$$=2\Delta y/\Delta x(x_k+1)-2y_k+2b-1 \text{ [since } m=\Delta y/\Delta x]$$

$$\Delta x(d1-d2)= 2\Delta y(x_k+1)-2 \Delta xy_k+2\Delta xb-\Delta x$$



Bresenham's Line Algorithm



Defining decision parameter $p_k=\Delta x(d1-d2)$

$p_k=2\Delta y(x_k+1)-2\Delta xy_k+2\Delta xb-\Delta x$

$=2\Delta yx_k+2\Delta y-2\Delta xy_k+\Delta x(2b-1)$

$=2\Delta yx_k-2\Delta xy_k+2\Delta y+\Delta x(2b-1)$

$=2\Delta yx_k-2\Delta xy_k+c$.....(4)

where $c=2\Delta y+\Delta x(2b-1)$

- Case I:

If $p_k>=0$ then $d2 < d1$ which implies that $y_k + 1$ is nearer than y_k. So pixel at $(y_k + 1)$ is better to choose which reduce error than pixel at y_k. This determine next pixel co-ordinate to plot is $(x_k + 1, y_k + 1)$

- Case II:

If $p_k< 0$ then $d1< d2$ which implies pixel at y_k is nearer than pixel at $(y_k + 1)$. So pixel at y_k is better to choose which reduce error than pixel at $(y_k + 1)$. This determine next pixel co-ordinate to plot is $(x_k + 1, y_k)$

Bresenham's Line Algorithm

Now,

Similarly, pixel as $(x_k + 2)$ can be determine whether it is $(x_k + 2, y_{k+1})$ or $(x_k + 2, y_{k+2})$ by looking the sign of deciding parameter p_{k+1} assuming pixel as $(x_k + 1)$ is known.

$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$ where c is same as in p_k

Now,

$p_{k+1} - p_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c - (2\Delta y x_k - 2\Delta x y_k + c)$

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

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

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

Bresenham's Line Algorithm

$p_{k+1}-p_k = 2\Delta y-2\Delta x(y_{k+1}-y_k)$

This implies that decision parameter for the current column can be determined if the decision parameter of the last column is known.

Here $(y_{k+1} - y_k)$ could either 0 or 1 which depends on sign of p_k

If $p_k >= 0$ (i.e.d2< d1), $y_{k+1} = y_k + 1$ which implies $(y_{k+1} - y_k) = 1$

That is at $p_k >= 0$ the pixel to plot is $(x_k + 1, y_k + 1)$ and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

If $p_k <0$(d1<d2), y_{k+1}=yk which implies ($y_{k+1}-y_k=0$)

That is at $p_k<0$ then pixel to be plotted is (x_k+1,y_k) and

$$p_{k+1}=p_k+2\Delta y$$

Bresenham's Line Algorithm

- **Initial decision parameter (p_0)**

  $p_k=2\Delta yx_k-2\Delta xy_k+2\Delta y+\Delta x(2b-1)$

  $p_0=2\Delta yx_0-2\Delta xy_0+2\Delta y+\Delta x(2b-1)$

  We have $b=y_0-mx_0=y_0-\Delta y/\Delta x(x_0)$

  $=2\Delta yx_0-2\Delta xy_0+2\Delta y+2\Delta xy_0-2\Delta yx_0-\Delta x$

  $p_0=2\Delta y-\Delta x$

Bresenham's Line Algorithm

Step 1. Start

Step 2. Declare variables x_1,y_1,x_2,y_2,lx,ly, Δx, Δy,p0, p_k, p_{k+1}.

Step 3. Read Values of x_1, y_1,x_2,y_2.

Step 4. Calculate Δx=absolute(x_2-x_1)

Δy=absolute(y_2-y_1)

Step 5. if (x_2>x_1)

assign lx=1

else

assign lx=-1

Step 6. if (y_2>y_1)

assign ly=1

else

assign ly=-1

Bresenham's Line Algorithm

Step 7. Plot (x_1, y_1)

Step 8. if $\Delta x > \Delta y$(i.e. /m/<1)

compute $p_0 = 2\Delta y - \Delta x$

starting at k=0 to Δx times, repeat

if $(p_k < 0)$

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

$y_{k+1} = y_k$

$p_{k+1} = p_k + 2\Delta y$

else

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

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

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

plot(x_{k+1}, y_{k+1})

Bresenham's Line Algorithm

else

calculate $p_0=2\Delta x-\Delta y$

starting at k=0 to Δy times, repeat

if($p_k<0$)

$x_{k+1}=x_k$

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

$p_{k+1}=p_k+2\Delta x$

else

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

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

$p_{k+1}=p_k+2\Delta x-2\Delta y$

Plot(x_{k+1},y_{k+1})

Step 9. Stop

Bresenham's Line Algorithm

Difference between DDA and BLA (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. 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 can not be used in other application except line drawing, but BLA can be implemented in other application such as circle, ellipse and other curves.

Bresenham's Line Algorithm

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

Solution

Here, Starting point of line = (x1, y1) = (20, 10) and

Ending point of line = (x2, y2) = (30, 18)

Thus, slope of line, $m = \Delta y / \Delta x = y2\text{-}y1 / x2\text{-}x1 = (18\text{-}10) / (30\text{-}20) = 8/10$

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

$|m| = 0.8 < 1$

Thus,

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 $P_k > 0$,
  $P_{k+1} = P_k + 2\Delta y - 2\Delta x$, and $y_{k+1} = y_k + 1$ & $x_{k+1} = x_k + 1$
- Else
  If $P_k < 0$,
  $P_{k+1} = P_k + 2\Delta y$, and $y_{k+1} = y_k$ & $x_{k+1} = x_k + 1$

Bresenham's Line Algorithm

Thus,

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)

Bresenham's Line Algorithm

Conclusion: Bresenham's Line drawing Algorithm for slope, $|m| \leq 1$

a) Calculate the starting value of the decision parameter: $P_0 = 2\Delta y - \Delta x$.

b) At each x_k along the line, starting at $k = 0$, perform the following test.

- If $P_k > 0$,
  $P_{k+1} = P_k + 2\Delta y - 2\Delta x$, and $y_{k+1} = y_k + 1$ & $x_{k+1} = x_k + 1$
- Else If $P_k < 0$,
  $P_{k+1} = P_k + 2\Delta y$, and $y_{k+1} = y_k$ & $x_{k+1} = x_k + 1$

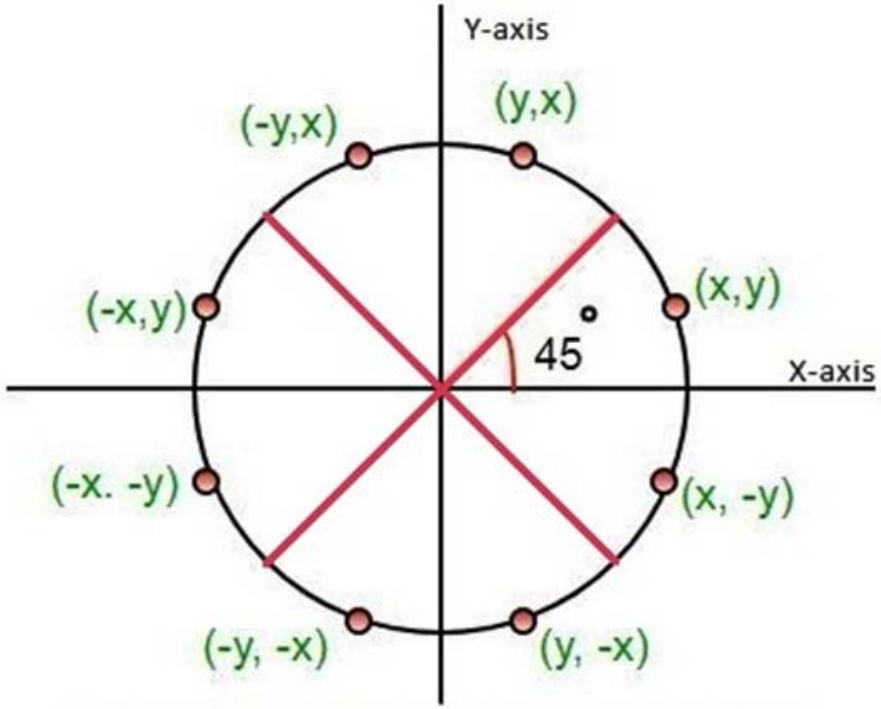
Scan Converting a circle

- The equation of circle in Cartesian form

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

- We sample at unit intervals and determine the closest pixel position to the specified circle at each steps.
- Non uniform spacing of plotted pixel is a problem
- To solve the computational complexity, we use symmetry of circle i.e. Calculate for one octant and use symmetry for others.
- Bresenham's line algorithm for raster displays is adapted for circle generation by setting up decision parameters for finding the closest pixel to the circumference at each sampling step.



For each pixel (x,y) all possible pixels in 8 octants

- We test the halfway position between two pixels to determine if this midpoint is inside or outside the circle boundary.

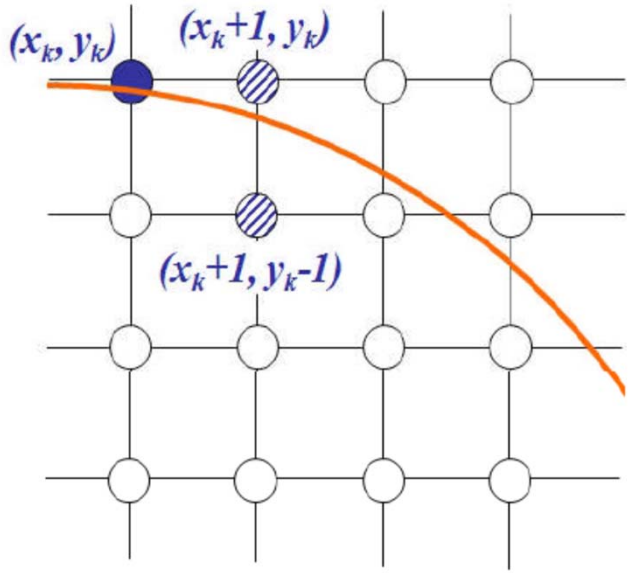
Midpoint Circle Algorithm

- The equation of circle with center (0, 0) is $x^2 + y^2 = r^2$
- With center (0,0) and radius r, function of circle $f_{circle}(x, y) = x^2 + y^2 - r^2$
- We know,

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

- Assume (x_k, y_k) is plotted, then next point close to the circle is

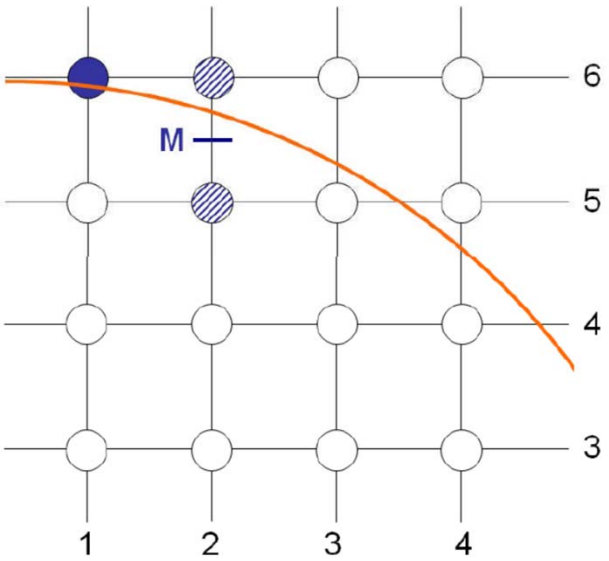
  (x_{k+1}, y_k) or (x_{k+1}, y_{k-1})
- Decision parameter is the circle function evaluated at the mid point between these two points.

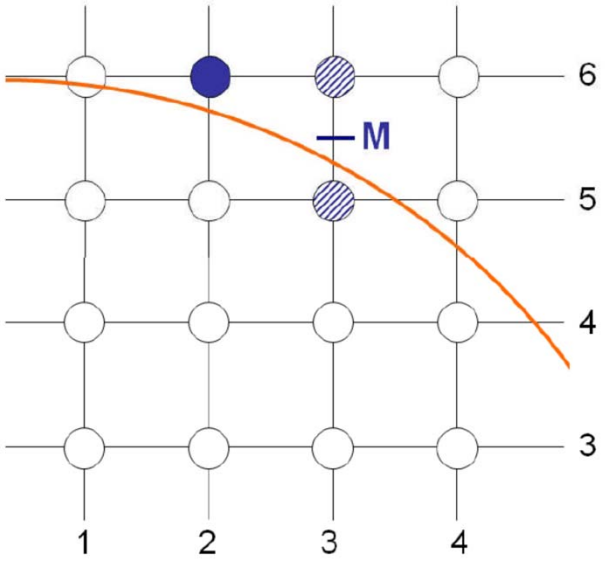


Midpoint Circle Algorithm

- $P_k = f_{circle}(x_k+1, y_k - ½)$
- $P_k = (x_k+1)^2 + (y_k - ½)^2 - r^2$
- So, if $P_k<0$ then midpoint is inside the circle and y_k is closer to circle boundary.

  else, midpoint is outside the circle. And y_k-1 is closer to the circle boundary





Midpoint Circle Algorithm

- successive decision parameter are obtaining using incremental calculations
- ie. Next decision parameter is obtained at

$x_{k+1}+1$ and $y_{k+1}- ½$ (mid point of y_{k+1} and $y_{k+1}-1$)

$P_{k+1} = f_{circle}(x_{k+1}+1, y_{k+1} – ½)$

$= (x_{k+1}+1)^2 +(y_{k+1} - ½)^2 – r^2$

$= (x_k+1+1)^2 +(y_{k+1} - ½)^2 – r^2$ because $x_{k+1} = x_k+1$

Now,

$P_{k+1} – P_k = (x_k+1+1)^2 +(y_{k+1} - ½)^2 – r^2 – (x_k+1)^2 - (y_k – ½)^2 + r^2$

$= (x_k+1)^2 +2(x_k+1) +1 + y_{k+1}^2 – y_{k+1} + ¼ - r^2 - (x_k+1)^2 – y_k^2 + y_k - ¼ + r^2$

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

Midpoint Circle Algorithm

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

where y_{k+1} is either y_k or y_k-1 depending on sign of P_k

- If $P_k<0$ then next pixel is at (x_k+1, y_k)

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

- If $P_k>=0$ then next pixel is at (x_k+1, y_k-1)

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

  $= P_k+2(x_k+1) + (y_k^2 - 2y_k +1 - y_k^2) +1 +1$

  $= P_k+2(x_k+1) - 2y_k + 2 +1$

  $= P_k+2(x_k+1) - 2(y_k - 1) +1$

  $= P_k+2x_{k+1} - 2y_{k+1} +1$ because $x_{k+1} = x_k+1$ and $y_{k+1} = y_k-1$

Midpoint Circle Algorithm

- **Initial decision parameter**

The initial decision parameter is obtained by evaluating the circle function at starting point (x0, y0)= (0,r)

$P_0 = 1 + (r - ½)^2 - r^2$ because $P_k = (x_k+1)^2 + (y_k - ½)^2 - r^2$

$= 1 + r^2 - r + ¼ - r^2$

$P_0 = 5/4 - r$

If the radius 'r' is specified as an integer, we can simply round to $P_0=1-r$

Midpoint Circle Algorithm

Conclusion

1. Calculate the initial decision parameter (P_0) $= 1 - r$
2. If $P < 0$

Plot $(x_k + 1, y_k)$

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

Else ($P \geq 0$)

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

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

Note : If the center of the circle is not at origin then first calculate the octant points of the circle in the same way as the center at origin & then add the given circle center on each calculated pixels.

Midpoint Circle Algorithm

Step 1. Start

Step 2. Declare variables x_c, y_c, r, x_0, y_0, P_0, P_k, P_{k+1}.

Step 3. Read Values of x_c, y_c, r.

Step 4. Initialize the x_0 and y_0 i. e. set the co-ordinates for the first point on the circumference of the circle centered at origin as.

$x_0=0$

$y_0=r$

Step 5. Calculate initial value of decision parameter

$P_0 = 5/4 - r$

Step 6. At each x_k position, starting from k=0

If $P_k<0$

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

$y_{k+1}= y_k$

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

Midpoint Circle Algorithm

else

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

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

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

Step 7. Determine the symmetry in other seven octants.

Step 8. Move each calculated pixel position (x,y) onto the circular path centered on (x_c, y_c)

Step 9. Plot the co-ordinates values

$x=x+x_c$

$y=y+y_c$

Step 10. Repeat steps 6 to 9 until $x \geq y$

Step 11. Stop

Midpoint Circle Algorithm

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

Solution

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 $P < 0$

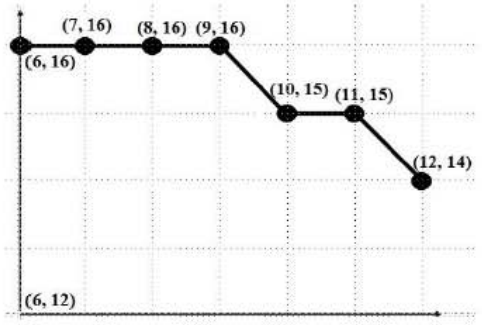
Plot $(x_k + 1, y_k)$

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

Else ($P \geq 0$)

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

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

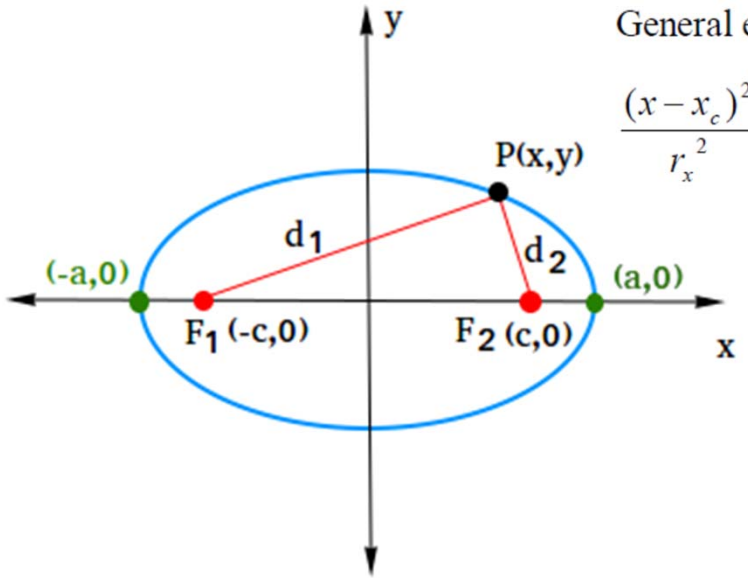


Thus,

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) (Halt)	(6+6, 7+7) = (12, 14)
6.	-3 + 2*6 +1= 10	6+1 = 7	6	(7, 6) (Only for checking)	

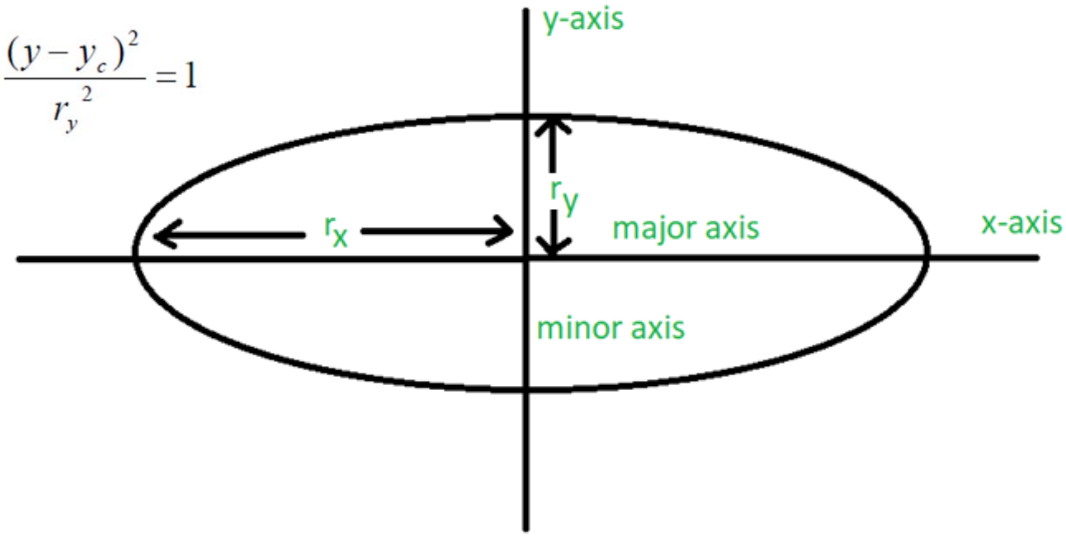
Scan Converting a ellipse

- An ellipse is described as a curve on a plane that surrounds two focal points such that the sum of the distances to the two focal points is constant for every point on the curve.



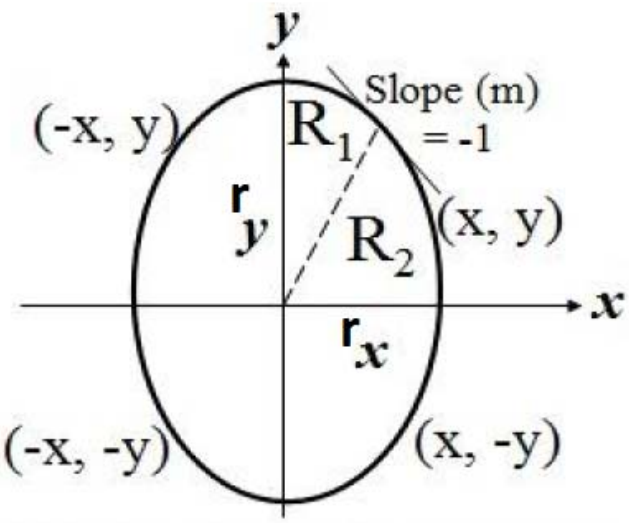
General equation:

$$\frac{(x-x_c)^2}{r_x^{\,2}}+\frac{(y-y_c)^2}{r_y^{\,2}}=1$$



Midpoint Ellipse Algorithm

- This algorithm is applied throughout the 1st quadrant in two parts
- Figure shows the division of 1st quadrant according to the slope of an ellipse with $r_x < r_y$
- process this quadrant by taking unit steps in x-direction where slope of curve has magnitude less than 1 i.e. Region 1 (R1), and taking unit step in y-direction where slope has magnitude greater than 1 i.e. Region 2 (R2).
- Two approach
  - start at position $(0, r_y)$ i.e. R1 and step clock wise along the elliptical path in first quadrants, shifting from unit step in x to unit step in y when magnitude of slope becomes greater than 1 i.e R2
  - alternatively, start at position $(r_x, 0)$ i.e R2 and select points in counter clockwise, shifting from unit step in y to unit step in x when the magnitude of slope becomes less than 1 i.e. R1.
- Here, we start at position $(0, r_y)$
- -we define an ellipse function with $(x_c, y_c)=(0,0)$

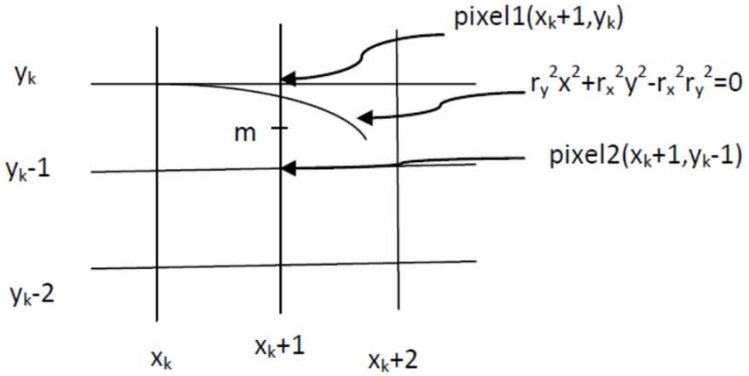


Midpoint Ellipse Algorithm

$f_{ellipse}(x,y)=r_y^2x^2+r_x^2y^2-r_x^2r_y^2$ .....(i)

With properties

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



- Thus ellipse function serves as the decision parameter
- At each sampling position, we select the next pixel along the ellipse path according to the sign of the ellipse function evaluated at the midpoint between the two candidate pixels.

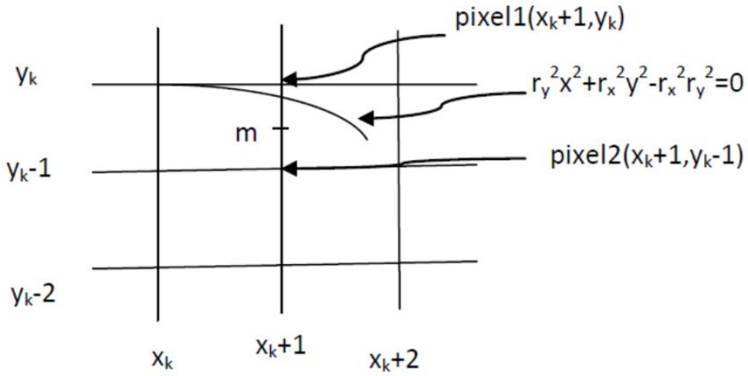
Midpoint Ellipse Algorithm

- at each step, we test the value of the slope of the curve,
- We have, $r_y^2x^2+r_x^2y^2= r_x^2r_y^2$

Differentiating with respect to x,

$$2r_y^2x + 2r_x^2y\frac{dy}{dx} = 0$$

$$\therefore \frac{dy}{dx} = -\frac{2r_y^2x}{2r_x^2y} \quad \text{.....(ii)}$$



-at the boundary between region 1 and region 2, slope=-1, so,

$$\frac{dy}{dx} = -1 = \frac{-2x_y^2x}{2r_x^2y}$$

$$\therefore 2r_y^2x = 2r_x^2y$$

-we move out of region 1, whenever

$$2r_y^2x \geq 2r_x^2y$$

Midpoint Ellipse Algorithm

-Assuming (x_k, y_k) has been illuminated (selected) we determine the next position along the ellipse path by evaluating the decision parameter at the midpoint $(x_k+1, y_k-\frac{1}{2})$

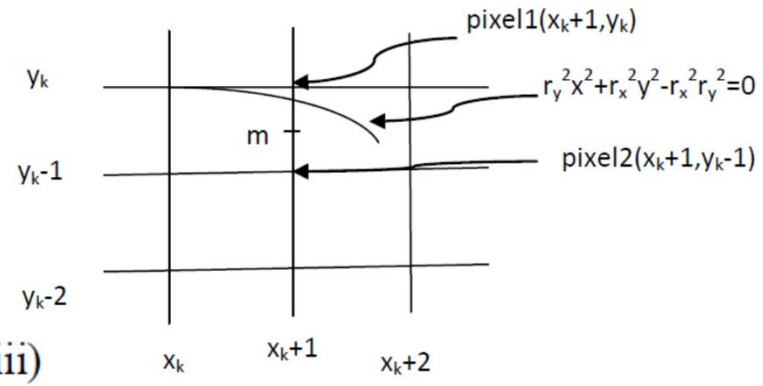
we have to determine the next point

(x_k+1, y_k) or (x_k+1, y_k-1)

We define decision parameter at mid-point as

$$P_{1k}=f_{ellipse}(x_k+1, y_k-\frac{1}{2})$$

$$P_{1k}==r_y^2(x_k+1)^2+r_x^2(y_k-\frac{1}{2})^2-r_x^2r_y^2 \quad \text{.....(iii)}$$



If $p_{1k}<0$, the mid-point is inside the ellipse and the pixel on scan line y_k is closer to the ellipse boundary

Otherwise,

The mid-point is outside or on the boundary and we select the pixel on scan line y_k-1

Midpoint Ellipse Algorithm

- At the next sampling position ($x_{k+1}+1=x_k+2$), the decision parameter for region 1 is evaluated as $P_{1k+1}=f_{ellipse}(x_{k+1}+1, y_{k+1}-\frac{1}{2})$

- $p_{1k+1} = r_y^2[(x_k+1)+1]^2 + r^2 x(y_{k+1}-\frac{1}{2})^2 - r_x^2 r_y^2$ .....(iv)

Subtracting equation (iii) from (iv)

$$p_{1k+1} - p_{1k} = r_y^2[(x_k+1+1)^2 - (x_k+1)^2] + r_x^2\left[\left(y_{k+1}-\frac{1}{2}\right)^2 - \left(y_k-\frac{1}{2}\right)^2\right] - r_x^2 r_y^2 + r_x^2 r_y^y$$

$$= r_y^2\left[(x_k+1)^2 + 2(x_k+1) + 1 - (x_k+1)^2\right] + r_x^2\left[y^2_{k+1} - 2y_{k+1}\frac{1}{2} + \frac{1}{4} - y_k^2 + 2y_k.\frac{1}{2} - \frac{1}{4}\right]$$

$$= r_y^2.2(x_k+1) + r_y^2 + r_x^2\left[\left(y^2_{k+1} - y_k^2\right) - (y_{k+1} - y_k)\right]$$

$$= 2r_y^2(x_k+1) + r_y^2 + r_x^2\left[\left(y^2_{k+1} - y_k^2\right) - (y_{k+1} - y_k)\right]$$

Midpoint Ellipse Algorithm

Where y_{k+1} is either y_k or y_k-1, depending on the sign of p_{1k}

$p_{1k} \le 0$ i.e. $y_{k+1}=y_k$ then decision parameter is

$$p_{1k+1} = p_{1k} + 2r_y^{\,2}(x_k + 1) + r_y^{\,2}$$

If $p_{1k} > 0$, i.e. $y_{k+1} = y_k - 1$, then decision parameter is

$$p_{1k+1} = p_{1k} + 2r_y^{\,2}(x_k + 1) + r_y^{\,2} - 2r_x^{\,2}y_{k+1}$$

The initial decision parameter is evaluated at start position $(x_0, y_o)=(0, r_y)$ as

$$P_{10} = f_{ellipse}\left(1, r_y - \frac{1}{2}\right)$$

$$= r_y^{\,2} + r_x^{\,2}\left(r_y - \frac{1}{2}\right)^2 - r_x^{\,2}y_x^{\,2}$$

$$p_{10} = r_y^{\,2} - r_x^{\,2}r_y + \frac{1}{4}\,r_x^{\,2} \quad \ldots\ldots\ldots\ldots \quad (v)$$

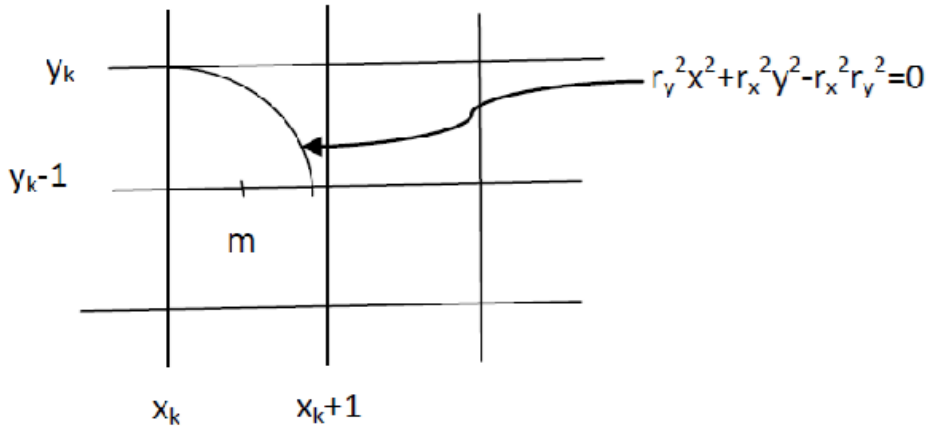
Midpoint Ellipse Algorithm

Region 2

We sample at unit steps in the negative y direction and the mid point is now taken between horizontal pixels at each step.

The decision parameter is

$P_{2k} = f_{ellipse}(x_k + \frac{1}{2}, y_k - 1)$



$$p_{2k} = r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2 \quad \text{.....} \quad \text{(vi)}$$

If $p_{2k} > 0$, the mid-point is outside the ellipse boundary and we select the pixel x_kIf $p_{2k} \leq 0$, the midpoint is inside or on the ellipse boundary and we select pixel position x_k+1

Note: Initial decision parameter for region 2 is calculated by taking last point of region 1 as (x_k, y_k)

Midpoint Ellipse Algorithm

Now, at next sampling position $y_{k+1}-1=y_k-2$

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

$$p_{2k+1} = r_y^2\left(x_{k+1}+\frac{1}{2}\right)^2 + r_x^2(y_{k+1}-1)^2 - r_x^2 r_y^2 \quad \text{.....(vii)}$$

Subtracting equation (vi) from (vii)

$$p_{2k+1} = p_{2k} + r_y^2\left[\left(x_{k+1}+\frac{1}{2}\right)^2 - \left(x_k+\frac{1}{2}\right)^2\right] + r_x^2\left[(y_k-1-1)^2 - (y_k-1)^2\right]$$

$$= \left[p_{2k} + r_x^2(y_k-1)^2 - 2(y_k-1)+1-(y_k-1)^2\right] + r_y^2\left[\left(x_{k+1}+\frac{1}{2}\right)^2 - \left(x_k+\frac{1}{2}\right)^2\right]$$

If $p_{2k} > 0$, then

$x_{k+1}=x_k$

$$p_{2k}+1 = p_{2k} - 2r_x^2 y_{k+1} + r_x^2$$

If $p_{2k} \le 0$, then

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

$$p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_y^2 y_{k+1} + r_y^2$$

Midpoint Ellipse Algorithm

Step 1. Start

Step 2. Declare variables x_c, y_c, r_x, r_y, x_0, y_0, p_{10}, p_{1k}, p_{1k+1}, p_{20}, p_{2k}, p_{2k+1}

Step 3. Read Values of x_c, y_c, r_x, r_y,

Step 4. obtain the first point on an ellipse centered on origin (x_0, y_0) by initializing the x_0 and y_0 as $x_0=0$, $y_0=r$

Step 5. Calculate the initial value of the decision parameter in region 1 as

$$p_{10} = r_y^2 - r_x^2 r_y + \frac{1}{4} r_y^2$$

Step 6. For each x_k position in region 1, starting at k=0, perform the following test.

If $P_{1k}<0$,

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

$y_{k+1}=y_k$

$$p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$$

else

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

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

$$p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} - 2r_y^2 y_{k+1} + r_y^2$$

and continue until $2r_y^2 x \geq 2r_y^2 y$

Midpoint Ellipse Algorithm

Step 7. Calculate the initial decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as

$$p_{20} = r_y^2\left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$$

Step 8. At each y_k position in region 2, starting at k=0, perform the following test.

If $P_{2k}>0$,

$x_{k+1}=x_k$

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

$$p_{2k+1} = p_{2k} - 2r_x^2 y_{k+1} + r_x^2$$

else

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

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

$$p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Repeat the steps for region 2 until $y<0$.

Step 9. Determine the symmetry points in the other three quadrants.

Step 10. Move each calculated pixel position(x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values.

$x=x+x_c$ and $y=y+y_c$

Step 11. Stop.