Ellipse Generation Algorithms

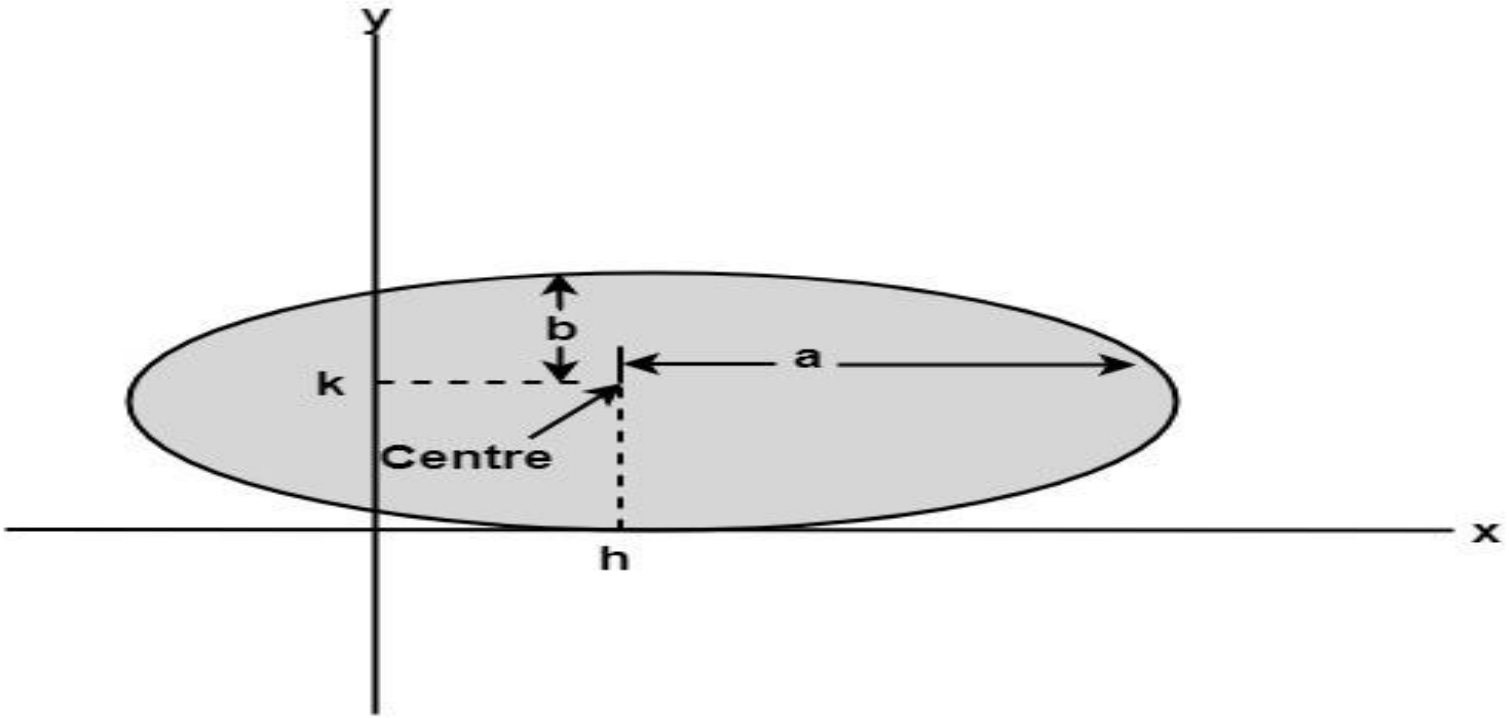
- Scan Converting Ellipse
- Polynomial Method
- Trigonometric Method
- Midpoint Ellipse Algorithm
- Scan Converting arcs and sectors

Scan Converting Ellipse

- The ellipse is also a symmetric figure like a circle but is four-way symmetry rather than eight-way.
- Ellipse is defined as the locus of a point in a plane which moves in a plane in such a manner that the ratio of its distance from a fixed point called focus in the same plane to its distance from a fixed straight line called directrix is always constant, which should always be less than unity.
- There two methods of defining an Ellipse: Polynomial Method of defining an Ellipse, Trigonometric method of defining an Ellipse
- **The midpoint ellipse method** is applied throughout the first quadrant in two parts. Now let us take the start position at **(0,b)** and step along the ellipse path in clockwise order throughout the first quadrant.

Scan Converting Ellipse

- The ellipse is also a symmetric figure like a circle but is four-way symmetry rather than eight-way.
- There two methods of defining an Ellipse: Polynomial Method of defining an Ellipse, Trigonometric method of defining an Ellipse



Polynomial Method

- The ellipse has a major and minor axis. If a and b a re major and minor axis respectively. The centre of ellipse is (i, j). The value of x will be incremented from i to a and value of y will be calculated using the following formula

$$y = b\sqrt{1 - \frac{(x-i)^2}{a^2}} + j$$

Drawback of Polynomial Method

- It requires squaring of values. So floating point calculation is required.
- Routines developed for such calculations are very complex and slow.

Polynomial Method

Algorithm

- Step 1: Set the initial variables: a = length of major axis; b = length of minor axis; (h, k) = coordinates of ellipse center; x = 0; i = step; x_{end} = a.
- Step 2. Test to determine whether the entire ellipse has been scan-converted. If x > x_{end}, stop.
- Step 3. Compute the value of the y coordinate:

$$y = b\sqrt{1 - \frac{x^2}{a^2}}$$

- Step 4: Plot the four points, found by symmetry, at the current (x, y) coordinates:

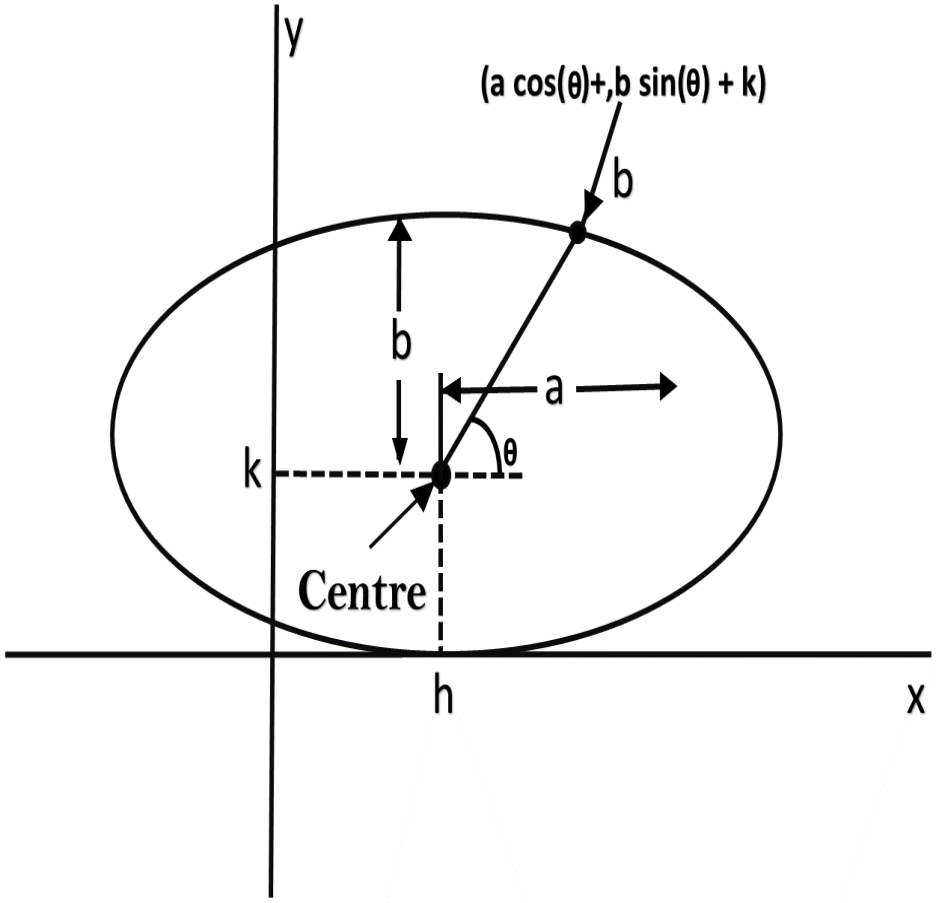
  Plot (x + h, y + k) Plot (-x + h, -y + k)

  Plot (-y + h, x + k) Plot (y + h, -x + k)

- Step 5. Increment x; x = x + i. Step 6. Go to step 2.

Trigonometric Method

- The following equation defines an ellipse trigonometrically as shown in fig:
- x = a * cos (θ) +h and
  y = b * sin (θ)+k
  where (x, y) = the current coordinates
  a = length of major axis
  b = length of minor axis
  θ= current angle
  (h, k) = ellipse center
- In this method, the value of θ is varied from 0 to PI/2 radians. The remaining points are found by symmetry.



Trigonometric Method

Drawback

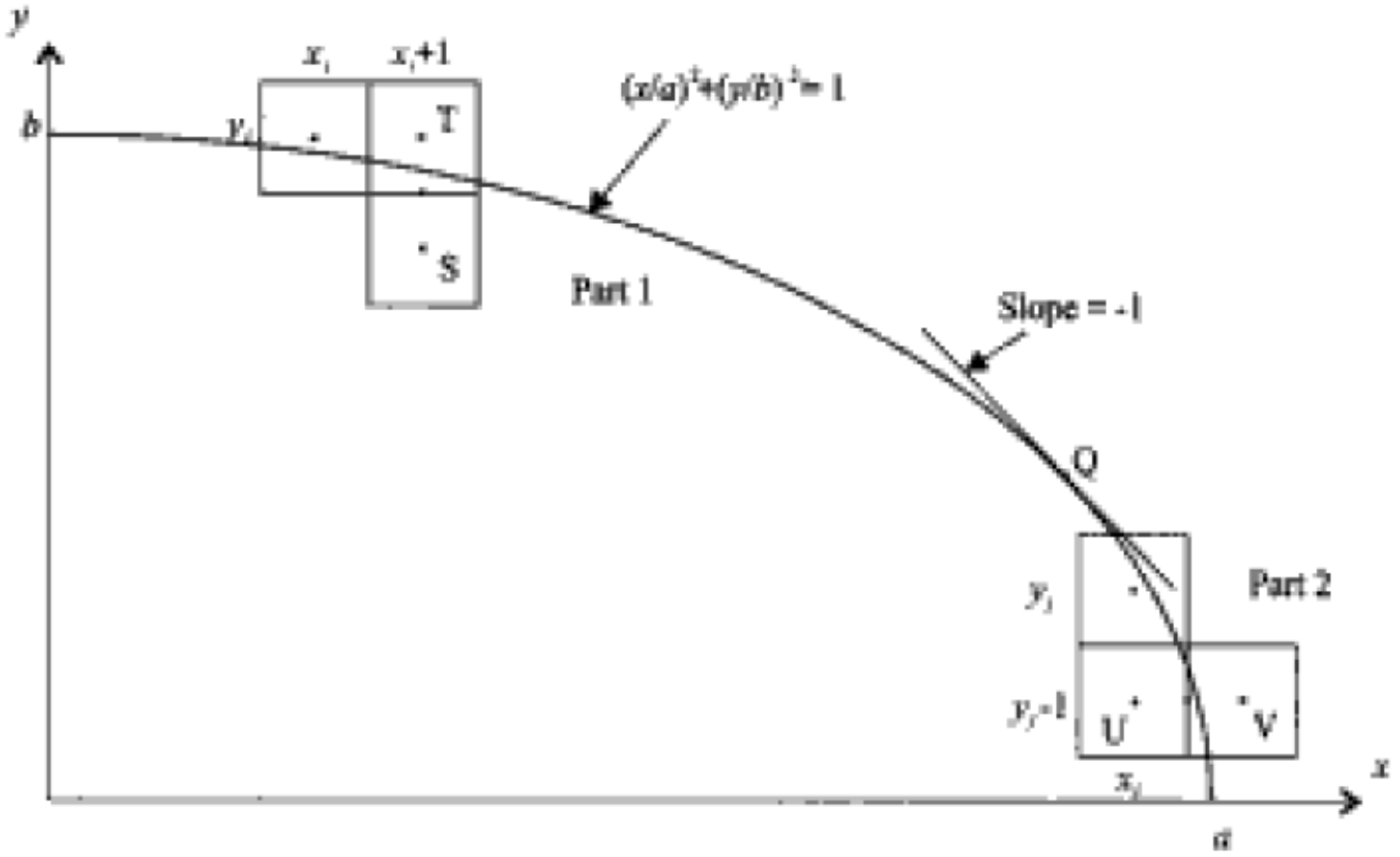
- This is an inefficient method.
- It is not an interactive method for generating ellipse.
- The table is required to see the trigonometric value.
- Memory is required to store the value of θ.

Midpoint Ellipse Algorithm

- This is an incremental method for scan converting an ellipse that is centered at the origin in standard position i.e., with the major and minor axis parallel to coordinate system axis.
- It is very similar to the midpoint circle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.
- Let's first rewrite the ellipse equation and define the function ‘f’ that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

$$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{cases} < 0 \;\; if\,(x,y) \;\; inside \\ = 0 \;\; if\,(x,y) \;\; on \\ > 0 \;\; if\,(x,y) \;\; outside \end{cases}$$

Midpoint Ellipse Algorithm



Midpoint Ellipse Algorithm

- Now divide the elliptical curve from (0, b) to (a, 0) into two parts at point Q where the slope of the curve is -1.
- Slope of the curve is defined by the f(x, y) = 0 is $\frac{dy}{dx} = -\frac{fx}{fy}$

  where fx & fy are partial derivatives of f(x, y) with respect to x & y.
- We have fx = $2b^2$ x, fy=$2a^2$ y & $\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$
- Hence we can monitor the slope value during the scan conversion process to detect point Q. Our starting point is (0, b)
- Suppose that the coordinates of the last scan converted pixel upon entering step i are (x_i,y_i). We are to select either T (x_i+1, y_i) or S (x_i+1, y_i-1) to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

Our starting point is $(0, b)$. Suppose that the coordinates of the last scan-converted pixel upon entering step i are (x_i, y_i). We are to select either $T(x_i + 1, y_i)$ or $S(x_i + 1, y_i - 1)$ to be the next pixel. The midpoint of the vertical line connecting T and S is used to define the following decision parameter:

$$p_i = f(x_i + 1, y_i - \tfrac{1}{2}) = b^2(x_i + 1)^2 + a^2(y_i - \tfrac{1}{2})^2 - a^2b^2$$

If $p_i < 0$, the midpoint is inside the curve, and we choose pixel T. On the other hand, if $p_i \geq 0$, the midpoint is outside or on the curve, and we choose pixel S. Similarly, we can write the decision parameter for the next step:

$$p_{i+1} = f(x_{i+1} + 1, y_{i+1} - \tfrac{1}{2}) = b^2(x_{i+1} + 1)^2 + a^2(y_{i+1} - \tfrac{1}{2})^2 - a^2b^2$$

Since $x_{i+1} = x_i + 1$, we have

$$p_{i+1} - p_i = b^2[(x_{i+1} + 1)^2 - x_{i+1}^2] + a^2[(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$

Hence

$$p_{i+1} = p_i + 2b^2x_{i+1} + b^2 + a^2[(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$

If T is the chosen pixel (meaning $p_i < 0$), we have $y_{i+1} = y_i$. On the other hand, if pixel S is chosen (meaning $p_i \geq 0$), we have $y_{i+1} = y_i - 1$. Thus we can express p_{i+1} in terms of p_i and (x_{i+1}, y_{i+1}):

$$p_{i+1} = \begin{cases} p_i + 2b^2x_{i+1} + b^2 & \text{if } p_i < 0 \\ p_i + 2b^2x_{i+1} + b^2 - 2a^2y_{i+1} & \text{if } p_i \geq 0 \end{cases}$$

The initial value for this recursive expression can be obtained by evaluating the original definition of p_i with $(0, b)$:

We now move on to derive a similar formula for part 2 of the curve. Suppose pixel (x_j, y_j) has just been scan-converted upon entering step j. The next pixel is either $U(x_j, y_j - 1)$ or $V(x_j + 1, y_j - 1)$. The midpoint of the horizontal line connecting U and V is used to define the decision parameter

$$q_j = f(x_j + \tfrac{1}{2}, y_j - 1) = b^2(x_j + \tfrac{1}{2})^2 + a^2(y_i - 1)^2 - a^2b^2$$

If $q_j < 0$, the midpoint is inside the curve and V is chosen. On the other hand, if $q_j \geq 0$, it is outside or on the curve and U is chosen. We also have

$$q_{j+1} = f(x_{j+1} + \tfrac{1}{2}, y_{j+1} - 1) = b^2(x_{j+1} + \tfrac{1}{2})^2 + a^2(y_{j+1} - 1)^2 - a^2b^2$$

Since $y_{j+1} = y_j - 1$, we have

$$q_{j+1} - q_j = b^2[(x_{j+1} + \tfrac{1}{2})^2 - (x_j + \tfrac{1}{2})^2] + a^2[(y_{j+1} - 1)^2 - y_{j+1}^2]$$

Hence

$$q_{j+1} = q_j + b^2[(x_{j+1} + \tfrac{1}{2})^2 - (x_j + \tfrac{1}{2})^2] - 2a^2y_{j+1} + a^2$$

If V is the chosen pixel (meaning $q_j < 0$), we have $x_{j+1} = x_j + 1$. On the other hand, if U is chosen (meaning $q_j \geq 0$), we have $x_{j+1} = x_j$. Thus we can express q_{j+1} in terms of q_j and (x_{j+1}, y_{j+1}):

$$q_{j+1} = \begin{cases} q_j + 2b^2x_{j+1} - 2a^2y_{j+1} + a^2 & \text{if } q_j < 0 \\ q_j - 2a^2y_{j+1} + a^2 & \text{if } q_j \geq 0 \end{cases}$$

The initial value for this recursive expression is computed using the original definition of q_j and the coordinates (x_k, y_k) of the last pixel chosen for part 1 of the curve:

$$q_1 = f(x_k + \tfrac{1}{2}, y_k - 1) = b^2(x_k + \tfrac{1}{2})^2 + a^2(y_k - 1)^2 - a^2b^2$$

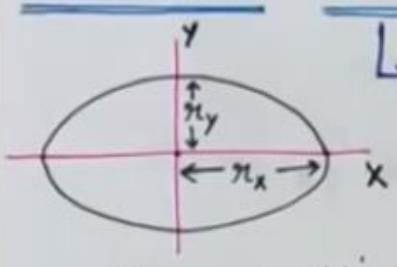
We can now put together a pseudo-code description of the midpoint algorithm for scan-converting the elliptical curve in the first quadrant. There are a couple of technical details that are worth noting. First, we keep track of the slope of the curve by evaluating the partial derivatives fx and fy at each chosen pixel position. This means that $fx = 2b^2x_i$ and $fy = 2a^2y_i$ for position (x_i, y_i). Since we have $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i$ or $y_i - 1$ for part 1, and $x_{j+1} = x_j$ or $x_j + 1$ and $y_{j+1} = y_j - 1$ for part 2, the partial derivatives can be updated incrementally using $2b^2$ and/or $-2a^2$. For example, if $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i - 1$, the partial derivatives for position (x_{i+1}, y_{i+1}) are $2b^2x_{i+1} = 2b^2x_i + 2b^2$ and $2a^2y_{i+1} = 2a^2y_i - 2a^2$. Second, since $2b^2x_{i+1}$ and $2a^2y_{i+1}$ appear in the recursive expression for the decision parameters, we can use them to efficiently compute p_{i+1} as well as q_{j+1}:

```
int x = 0, y = b;       /* starting point */
int aa = a*a, bb = b*b, aa2 = aa*2, bb2 = bb*2;
int fx = 0, fy = aa2*b;       /* initial partial derivatives */
int p = bb - aa*b + 0.25*aa;       /* compute and round off p1 */
while (fx < fy) { /* |slope| < 1 */
   setPixel(x, y);
   x++;
   fx = fx + bb2;
   if (p < 0)
      p = p + fx + bb;
   else {
      y--;
      fy = fy - aa2;
      p = p + fx + bb - fy;
   }
}
setPixel(x, y);       /* set pixel at (x, y) */
```

```
p = bb(x + 0.5)(x + 0.5) + aa(y - 1)(y - 1) - aa*bb;      /* set q1 */
while (y > 0) {
   y--;
   fy = fy - aa2;
   if (p >= 0)
      p = p - fy + aa;
   else {
      x++;
      fx = fx + bb2;
      p = p + fx - fy + aa;
   }
   setPixel(x, y);
}
```

Mid Point Ellipse Algorithm

MIDPOINT ELLIPSE DRAWING



↳ is an elongated circle.

Equation of ellipse centred at (0,0) :

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

a^2 → $2r_x$, b^2 → r_y

Major Axis $= 2a = 2r_x$

Minor Axis $= 2b = 2r_y$

Semi-Major Axis $= a = r_x$

Semi-Minor Axis $= b = r_y$

$$\frac{x^2b^2 + y^2a^2 = 1}{a^2b^2}$$

$$\Rightarrow b^2x^2 + a^2y^2 = a^2b^2$$

$$\Rightarrow b^2x^2 + a^2y^2 - a^2b^2 = 0$$

ALGORITHM PART 1

$$r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2 = 0 \quad \text{— (1)}$$

If we put any point in eqn (1)

$= 0$	< 0	> 0
Point lies ON ellipse	INSIDE the ellipse	OUTSIDE the ellipse

Difference between Circle & Ellipse

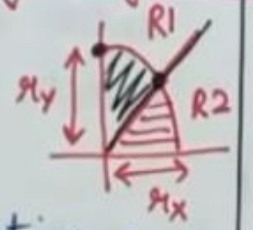
★ Circle has 8-way symmetry
Ellipse has 4-way symmetry.

★ In circle we need to plot only 1 octant of any quadrant, but in ellipse we need to plot 2 octants i.e 1 complete quadrant to plot entire ellipse.

Mid Point Ellipse Algorithm

Midpoint Ellipse Drawing Algorithm PART 2

Quadrant 1 → Region 1



- ★ Start Point : $(0, r_y)$
- ★ Slope of Curve < -1
- ★ Take unit steps in positive x direction till boundary between the 2 regions is reached → $x \to x+1$

Quadrant 1 → Region 2

- ∘ Slope of Curve > -1 → $y \to y-1$
- ∘ Take unit steps in negative y direction till the end of quadrant

On the boundary between the 2 regions/octants, the slope of the curve is -1.

SLOPE OF CURVE → ellipse

$$\frac{dy}{dx}\left\{r_y^2 x^2 + r_x^2 y^2 - a^2 b^2\right\} = 0$$

$r_x^2 r_y^2$

$$\frac{dy}{dx}(y^2) = \frac{dy}{dx}\left(\frac{a^2b^2 - r_y^2 x^2}{r_x^2}\right)$$

$$\frac{dy}{dx}(y^2) = \frac{dy}{dx}\left(\frac{r_x^2 r_y^2}{r_x^2}\right) - \frac{dy}{dx}\left(\frac{r_y^2 x^2}{r_x^2}\right)$$

$$\frac{dy}{dx} \cdot 2y = -\frac{2 r_y^2 x}{r_x^2}$$

$$\left\{\frac{dy}{dx} = -\frac{2 r_y^2 x}{2 r_x^2 y}\right\} \rightarrow \text{Slope of ellipse.}$$

$$\frac{-2 r_y^2 x}{2 r_x^2 y} = -1 \Rightarrow \boxed{-2 r_y^2 x = -2 r_x^2 y}$$

Mid Point Ellipse Algorithm

Region 1 $\longrightarrow (x_k+1 ; y_k)$, (x_k+1 , y_k-1) $\Rightarrow (x_k+1 , y_k-\frac{1}{2})$ Midpoint

Eqn: $r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 = 0$

$\Rightarrow r_y^2 (x_k+1)^2 + r_x^2 (y_k-\frac{1}{2}) - r_x^2 r_y^2 = p1_k$ — (1)

$\Rightarrow r_y^2 (x_{k+1}+1)^2 + r_x^2 (y_{k+1}-\frac{1}{2})^2 - r_x^2 r_y^2 = p1_{k+1}$

$\Rightarrow r_y^2 ((x_k+1)+1)^2 + r_x^2 (y_{k+1}-\frac{1}{2})^2 - r_x^2 r_y^2 = p1_{k+1}$ — (2)

(2) − (1) : $p1_{k+1} - p1_k$

$\Rightarrow r_y^2 \{(x_k+1)+1\}^2 - r_y^2 (x_k+1)^2 + r_x^2 (y_{k+1}-\frac{1}{2})^2 - r_x^2 (y_k-\frac{1}{2})^2 - r_x^2 r_y^2 + r_x^2 r_y^2$

$\Rightarrow r_y^2 \{(x_k+1)^2 + 1 + 2(x_k+1)\} - r_y^2 (x_k^2 + 1 + 2x_k) + r_x^2 (y_{k+1}^2 + \frac{1}{4} - y_{k+1}) - r_x^2 (y_k^2 + \frac{1}{4} - y_k)$

$\Rightarrow r_y^2 \{x_k^2 + 1 + 2x_k + 1 + 2x_k + 2 - x_k^2 - 1 - 2x_k\} + r_x^2 \{y_{k+1}^2 + \frac{1}{4} - y_{k+1} - y_k^2 - \frac{1}{4} + y_k\}$

$\Rightarrow r_y^2 \{2(x_k+1) + 1\} + r_x^2 \{y_{k+1}^2 - y_k^2 - y_{k+1} + y_k\} = p1_{k+1} - p1_k$

x_{k+1} $\quad p1_k < 0 : y_k$ $\quad y_k - 1 : p1_k \geq 0$

Mid Point Ellipse Algorithm

$$p1_{k+1} - p1_k = r_y^2\left[2(x_k+1)+1\right] + r_x^2\left[y_{k+1}^2 - y_k^2 - y_{k+1} + y_k\right]$$

★ When $p1_k < 0 \longrightarrow y_{k+1} = y_k$

$$p1_{k+1} = p1_k + r_y^2(2x_{k+1}+1) \longrightarrow \boxed{p1_{k+1} = p1_k + r_y^2 2x_{k+1} + r_y^2}$$

★ When $p1_k \geq 0 \longrightarrow y_{k+1} = y_k - 1$

$$p1_{k+1} - p1_k = 2x_{k+1} r_y^2 + r_y^2 + r_x^2\left[(y_k-1)^2 - y_k^2 - (y_k-1) + y_k\right]$$

$$p1_{k+1} = p1_k + 2x_{k+1} r_y^2 + r_y^2 + r_x^2\left[y_k^2 + 1 - 2y_k - y_k^2 - y_k + 1 + y_k\right]$$

$$\boxed{p1_{k+1} = p1_k + 2x_{k+1} r_y^2 + r_y^2 - 2y_{k+1} r_x^2}$$

Initial Decision Parameter

↳ Put $(0, r_y)$ in $p1_k = r_y^2(x_k+1)^2 + r_x^2\left(y_k - \frac{1}{2}\right)^2 - r_x^2 r_y^2$

initial/first point of R1

↳ $p1_0 = r_y^2(0+1)^2 + r_x^2\left(r_y - \frac{1}{2}\right)^2 - r_x^2 r_y^2$

$$p1_0 = r_y^2 + r_x^2\left\{r_y^2 + \frac{1}{4} - r_y\right\} - r_x^2 r_y^2$$

$$\boxed{p1_0 = r_y^2 + r_x^2/4 - r_y r_x^2}$$

Mid Point Ellipse Algorithm

Region 2 : $(x_k, y_k-1), (x_k+1, y_k-1) \longrightarrow (x_k+\frac{1}{2}, y_k-1)$ → Midpoint

- Eqn: $r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$ ✓

$$r_y^2\left(x_k+\tfrac{1}{2}\right)^2 + r_x^2(y_k-1)^2 - r_x^2 r_y^2 = p2_k \quad \text{— ①}$$

$$\Rightarrow r_y^2\left(x_{k+1}+\tfrac{1}{2}\right)^2 + r_x^2(y_{k+1}-1)^2 - r_x^2 r_y^2 = p2_{k+1}$$

$$r_y^2\left(x_{k+1}+\tfrac{1}{2}\right)^2 + r_x^2\left((y_k-1)-1\right)^2 - r_x^2 r_y^2 = p2_{k+1} \quad \text{— ②}$$

② − ① : $p2_{k+1} - p2_k$

$$\Rightarrow r_y^2\left(x_{k+1}+\tfrac{1}{2}\right)^2 - r_y^2\left(x_k+\tfrac{1}{2}\right)^2 + r_x^2\underbrace{\left((y_k-1)-1\right)^2}_{a\ \ b} - r_x^2(y_k-1)^2 - r_x^2 r_y^2 + r_x^2 r_y^2$$

$$\Rightarrow r_y^2\left[x_{k+1}^2+\tfrac{1}{4}+x_{k+1}\right] - r_y^2\left[x_k^2+\tfrac{1}{4}+x_k\right] + r_x^2\left[(y_k-1)^2+1-2(y_k-1)\right] + r_x^2\left[y_k^2+1+2y_k\right]$$

$$r_y^2\left\{x_{k+1}^2+\tfrac{1}{4}+x_{k+1}-x_k^2-\tfrac{1}{4}-x_k\right\} + r_x^2\left\{y_k^2+1-2y_k+1-2y_k+2-y_k^2-1+2y_k\right\}$$

$$\Rightarrow r_y^2\left\{x_{k+1}^2+x_{k+1}-x_k^2-x_k\right\} + r_x^2\left\{-2y_{k+1}+1\right\}$$

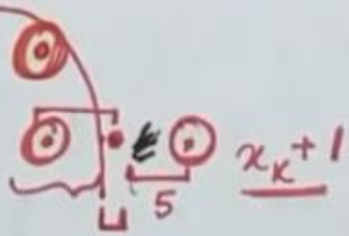
$$p2_{k+1} - p1_k = r_y^2\left\{x_{k+1}^2+x_{k+1}-x_k^2-x_k\right\} + r_x^2\left\{1-2y_{k+1}\right\}$$

Mid Point Ellipse Algorithm

$$p2_{k+1} = p2_k + r_y^2\{x_{k+1}^2 + x_{k+1} - x_k^2 + x_k\} - 2y_{k+1}r_x^2 + r_x^2$$

* If $p2_k > 0 \rightarrow x_{k+1} = x_k$

↳ $$p2_{k+1} = p2_k - 2y_{k+1}r_x^2 + r_x^2$$



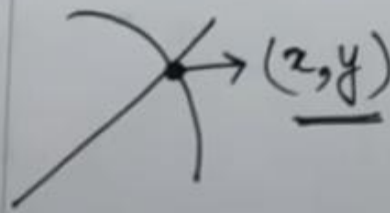
* If $p2_k \leq 0 \rightarrow x_{k+1} = x_k + 1$

↳ $$p2_{k+1} = p2_k + r_y^2[x_k^2 + 1 + 2x - x_k^2 + x_k + 1 - x_k] - 2y_{k+1}r_x^2 + r_x^2$$

$$p2_{k+1} = p2_k + r_y^2(2x_{k+1}) - 2y_{k+1}r_x^2 + r_x^2$$

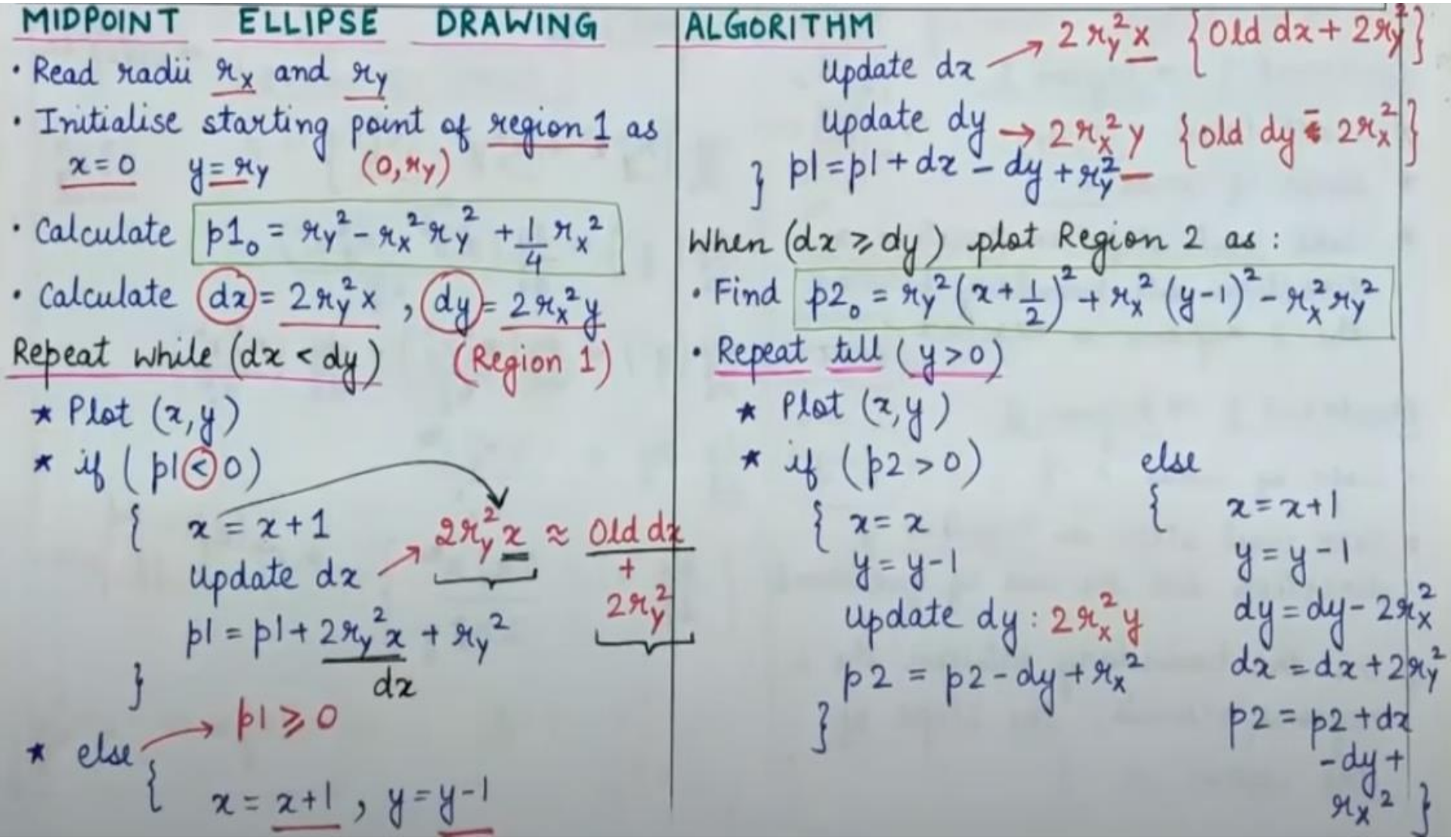
Initial Decision Parameter

↳ Obtained by putting the last point of Region 1 in the equation: $p2_k = r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$

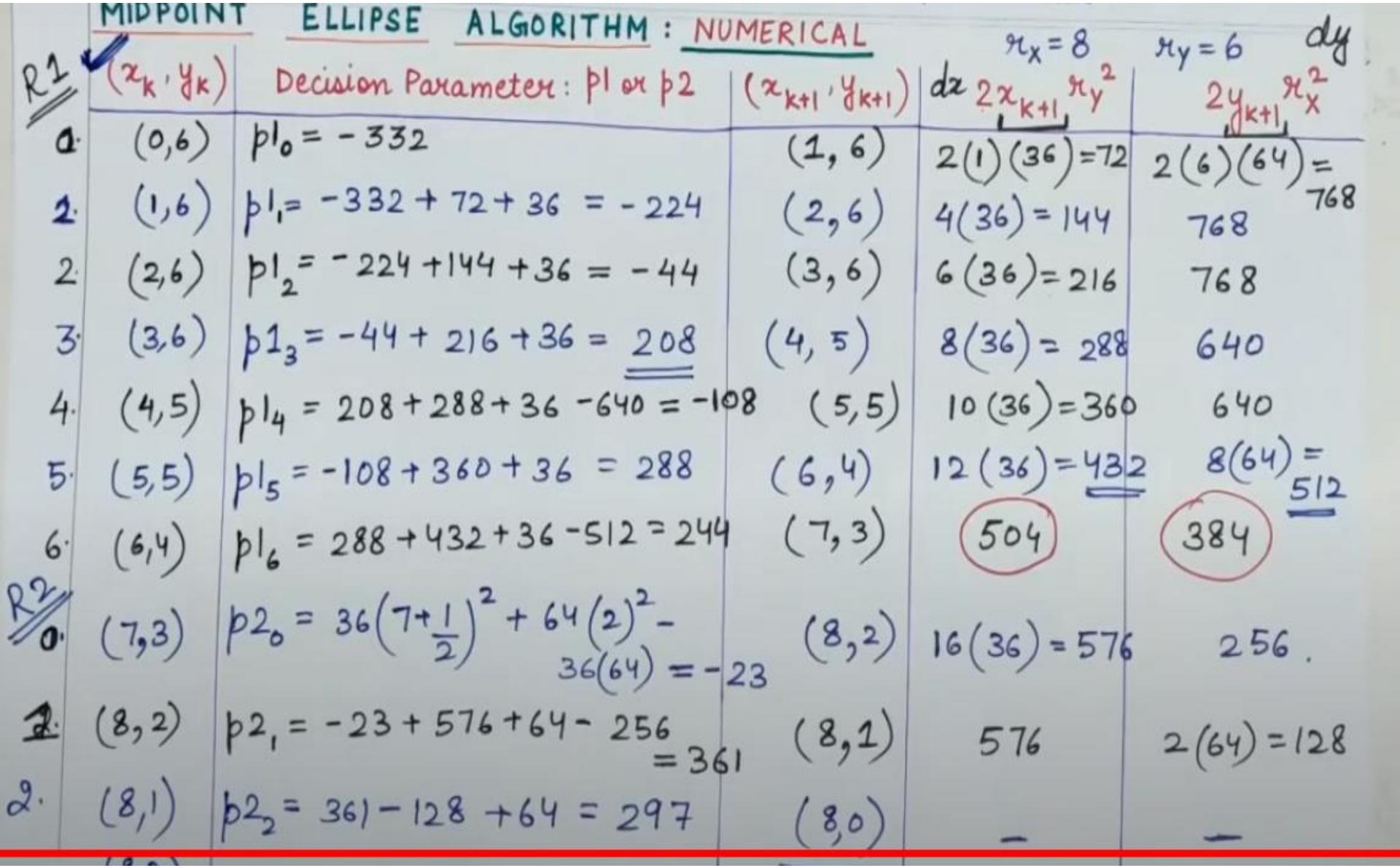


$$p2_k = r_y^2\left(x + \frac{1}{2}\right)^2 + r_x^2(y - 1)^2 - r_x^2 r_y^2$$

Mid Point Ellipse Algorithm



Mid Point Ellipse Algorithm Example



MIDPOINT ELLIPSE ALGORITHM : NUMERICAL

$r_x = 8$ $r_y = 6$

	(x_k, y_k)	Decision Parameter : $p1$ or $p2$	(x_{k+1}, y_{k+1})	dx $2x_{k+1} r_y^2$	dy $2y_{k+1} r_x^2$
R1					
0.	(0,6)	$p1_0 = -332$	(1,6)	$2(1)(36) = 72$	$2(6)(64) = 768$
1.	(1,6)	$p1_1 = -332 + 72 + 36 = -224$	(2,6)	$4(36) = 144$	768
2.	(2,6)	$p1_2 = -224 + 144 + 36 = -44$	(3,6)	$6(36) = 216$	768
3.	(3,6)	$p1_3 = -44 + 216 + 36 = 208$	(4,5)	$8(36) = 288$	640
4.	(4,5)	$p1_4 = 208 + 288 + 36 - 640 = -108$	(5,5)	$10(36) = 360$	640
5.	(5,5)	$p1_5 = -108 + 360 + 36 = 288$	(6,4)	$12(36) = 432$	$8(64) = 512$
6.	(6,4)	$p1_6 = 288 + 432 + 36 - 512 = 244$	(7,3)	504	384
R2					
0.	(7,3)	$p2_0 = 36\left(7+\frac{1}{2}\right)^2 + 64(2)^2 - 36(64) = -23$	(8,2)	$16(36) = 576$	256
1.	(8,2)	$p2_1 = -23 + 576 + 64 - 256 = 361$	(8,1)	576	$2(64) = 128$
2.	(8,1)	$p2_2 = 361 - 128 + 64 = 297$	(8,0)	—	—

Mid Point Ellipse Algo: pseudocode

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void ellipseMidPoint (int xCenter, int yCenter, int Rx, int Ry)
{
   int Rx2 = Rx*Rx;
   int Ry2 = Ry*Ry;
   int twoRx2 = 2*Rx2;
   int twoRy2 = 2*Ry2;
   int p;
   int x = 0;
   int y = Ry;
   int px = 0;
   int py = twoRx2 * y;
   void ellipsePlotPoints (int, int, int, int);

   /* Plot the first set of points */
   ellipsePlotPoints (xCenter, yCenter, x, y);

   /* Region 1 */
   p = ROUND (Ry2 - (Rx2 * Ry) + (0.25 * Rx2));
   while (px < py) {
      x++;
      px += twoRy2;
      if (p < 0)
         p += Ry2 + px;
      else {
         y--;
         py -= twoRx2;
         p += Ry2 + px - py;
      }
      ellipsePlotPoints (xCenter, yCenter, x, y);
   }

   /* Region 2 */
   p = ROUND (Ry2*(x+0.5)*(x+0.5) + Rx2*(y-1)*(y-1) - Rx2*Ry2);
   while (y > 0) {
      y--;
      py -= twoRx2;
      if (p > 0)
         p += Rx2 - py;
      else {
         x++;
         px += twoRy2;
         p += Rx2 - py + px;
```

Mid Point Ellipse Algo: pseudocode

```
    }
    ellipsePlotPoints (xCenter, yCenter, x, y);
  }
}

void ellipsePlotPoints (int xCenter, int yCenter, int x, int y)
{
  setPixel (xCenter + x, yCenter + y);
  setPixel (xCenter - x, yCenter + y);
  setPixel (xCenter + x, yCenter - y);
  setPixel (xCenter - x, yCenter - y);
}
```

Scan converting arcs and sectors

Arcs

An arc [Fig. 3-13(*a*)] may be generated by using either the polynomial or the trigonometric method. When the trigonometric method is used, the starting value is set equal to θ_1 and the ending value is set equal to θ_2 [see Figs. 3-13(*a*) and 3-13(*b*)]. The rest of the steps are similar to those used when scan-converting a circle, except that symmetry is not used.

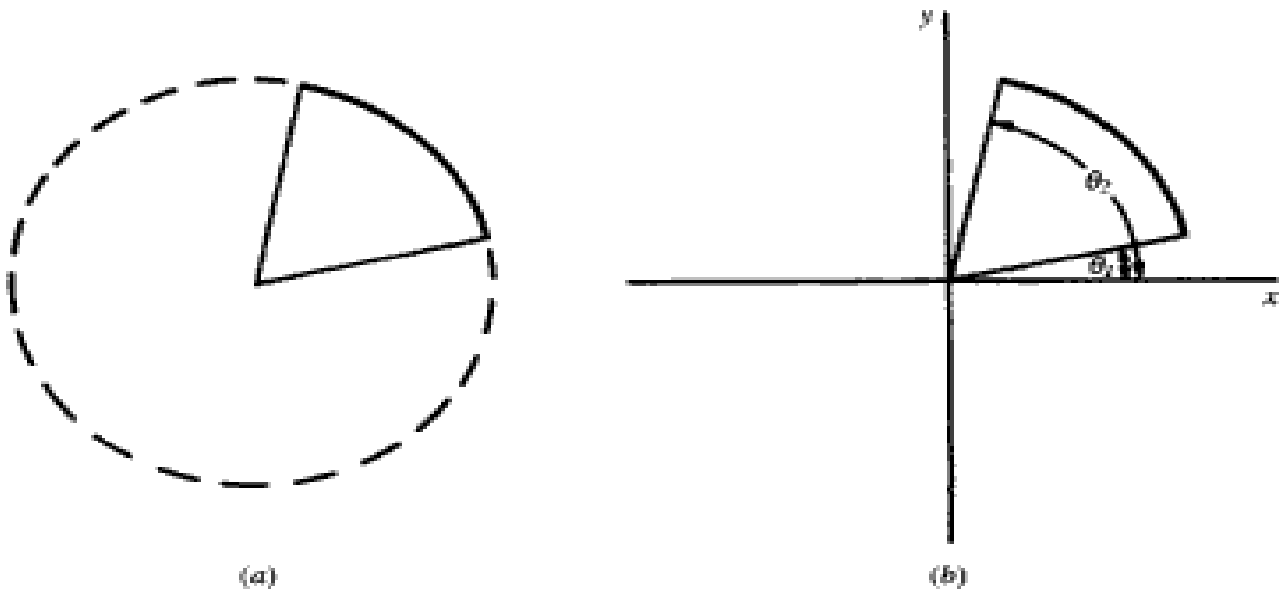


Fig. 3-13

When the polynomial method is used, the value of x is varied from x_1 to x_2 and the values of y are found by evaluating the expressio $\sqrt{r^2 - x^2}$ (Fig. 3-14).

Scan converting arcs and sectors

From the graphics programmer's point of view, arcs would appear to be nothing more than portions of circles. However, problems occur if algorithms such as Bresenham's circle algorithm are used in drawing an arc. In the case of Bresenham's algorithm, the endpoints of an arc must be specified in terms of the x, y coordinates. The general formulation becomes inefficient when endpoints must be found (see Fig. 3-15). This occurs because the endpoints for each 45° increment of the arc must be found. Each of the eight points found by reflection must be tested to see if the point is between the specified endpoints of the arc. As a result, a routine to draw an arc based on Bresenham's algorithm must take the time to calculate and test

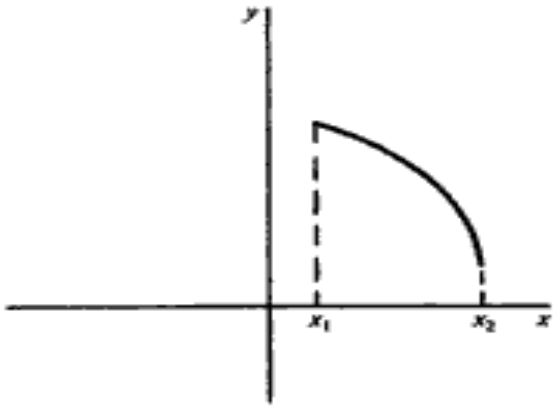


Fig. 3-14

every point on the circle's perimeter. There is always the danger that the endpoints will be missed when a method like this is used. If the endpoints are missed, the routine can become caught in an infinite loop.

Scan converting arcs and sectors

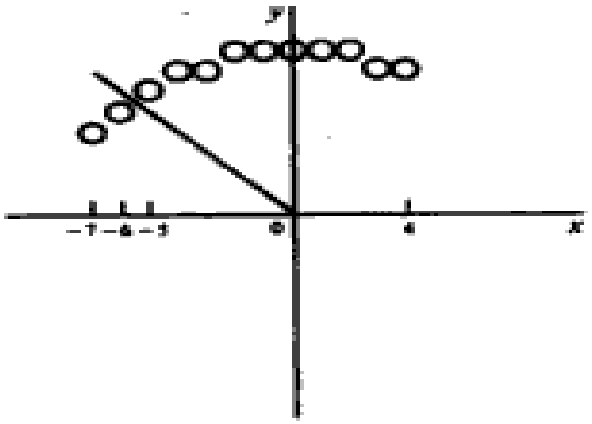


Fig. 3-15

Sectors

A sector is scan-converted by using any of the methods of scan-converting an arc and then scan-converting two lines from the center of the arc to the endpoints of the arc.

For example, assume that a sector whose center is at point (h, k) is to be scan-converted. First, scan-convert an arc from θ_1 to θ_2. Next, a line would be scan-converted from (h, k) to $(r\cos(\theta_1) + h, r\sin(\theta_1) + k)$. A second line would be scan-converted from (h, k) to $(r\cos(\theta_2) + h, r\sin(\theta_2) + k)$.