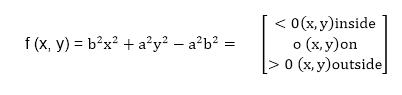
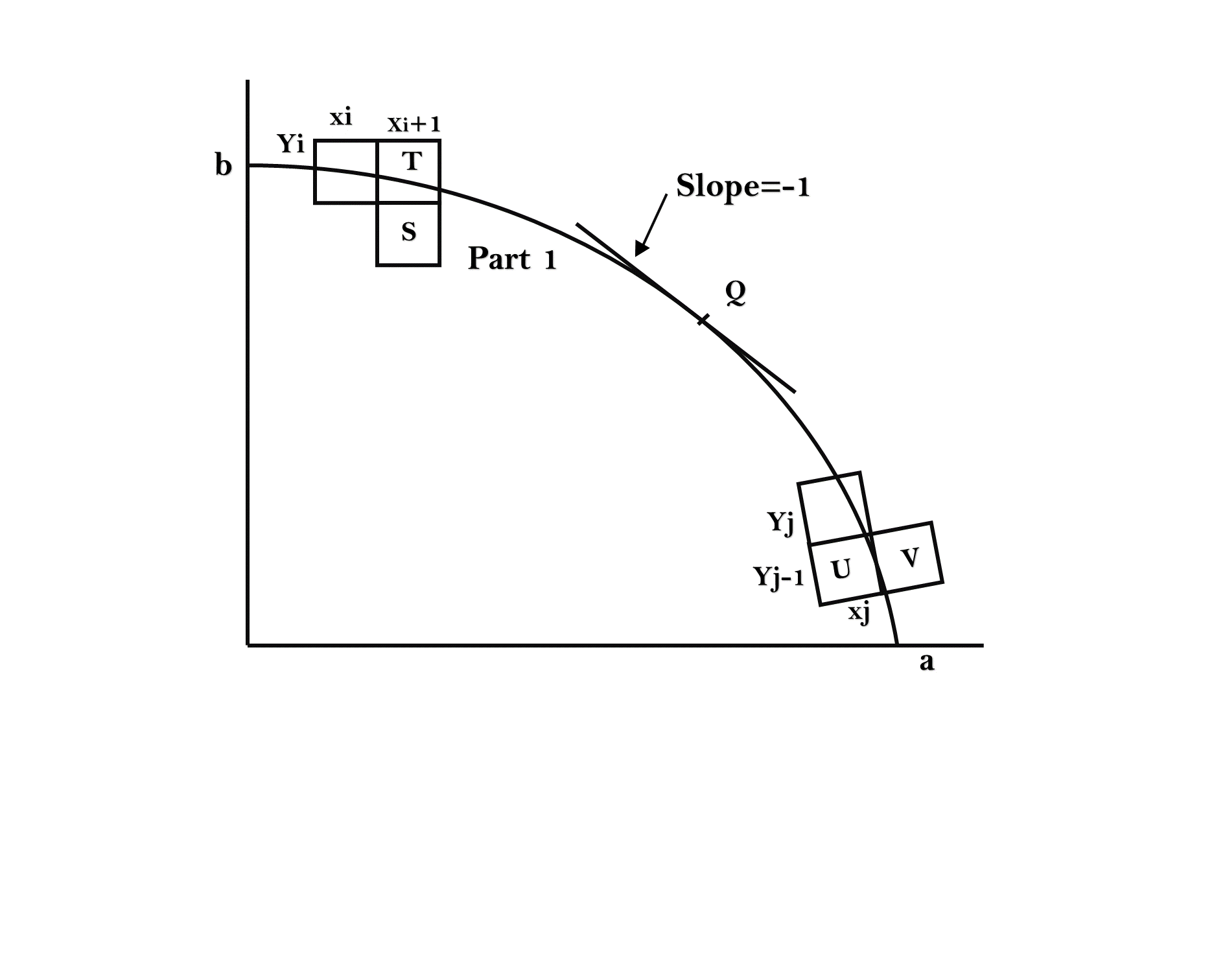
**Midpoint Ellipse Algorithm:**

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:

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 isMidpoint Ellipse Algorithmwhere fx & fy are partial derivatives of f(x, y) with respect to x & y.

We have fx = 2b2 x, fy=2a2 y & Midpoint Ellipse AlgorithmHence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is (0, b)

Suppose that the coordinates of the last scan converted pixel upon entering step i are (xi,yi). We are to select either T (xi+1),yi) or S (xi+1,yi-1) to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

pi = f(xi+1),yi-Midpoint Ellipse Algorithm)  
pi=b2 (xi+1)2+a2 (yi-Midpoint Ellipse Algorithm)2-a2 b2

If pi<0, the midpoint is inside the curve and we choose pixel T.

If pi>0, the midpoint is outside or on the curve and we choose pixel S.

Decision parameter for the next step is:

pi+1=f(xi+1+1,yi+1-Midpoint Ellipse Algorithm)  
          = b2 (xi+1+1)2+a2 (yi+1-Midpoint Ellipse Algorithm)2-a2 b2

Since xi+1=xi+1,we have  
          pi+1-pi=b2[((xi+1+1)2+a2 (yi+1-Midpoint Ellipse Algorithm)2-(yi -Midpoint Ellipse Algorithm)2]  
          pi+1= pi+2b2 xi+1+b2+a2 [(yi+1-Midpoint Ellipse Algorithm)2-(yi -Midpoint Ellipse Algorithm)2]

If T is chosen pixel (pi<0), we have yi+1=yi.

If S is chosen pixel (pi>0) we have yi+1=yi-1. Thus we can express

          pi+1in terms of pi and (xi+1,yi+1):           pi+1= pi+2b2 xi+1+b2          if pi<0           = pi+2b2 xi+1+b2-2a2 yi+1 if pi>0

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

p1 = (b2+a2 (b-Midpoint Ellipse Algorithm)2-a2 b2   
= b2-a2 b+a2/4

Suppose the pixel (xj yj) has just been scan converted upon entering step j. The next pixel is either U (xj ,yj-1) or V (xj+1,yj-1). The midpoint of the horizontal line connecting U & V is used to define the decision parameter:

qj=f(xj+Midpoint Ellipse Algorithm,yj-1)  
qj=b2 (xj+Midpoint Ellipse Algorithm)2+a2 (yj -1)2-a2 b2

If qj<0, the midpoint is inside the curve and we choose pixel V.

If qj≥0, the midpoint is outside the curve and we choose pixel U.Decision parameter for the next step is:

qj+1=f(xj+1+Midpoint Ellipse Algorithm,yj+1-1)  
          = b2 (xj+1+Midpoint Ellipse Algorithm)2+ a2 (yj+1-1)2- a2 b2

Since yj+1=yj-1,we have  
qj+1-qj=b2 [(xj+1+Midpoint Ellipse Algorithm)2-(xj +Midpoint Ellipse Algorithm)2 ]+a2 (yj+1-1)2-( yj+1)2 ]  
qj+1=qj+b2 [(xj+1+Midpoint Ellipse Algorithm)2-(xj +Midpoint Ellipse Algorithm)2]-2a2 yj+1+a2

If V is chosen pixel (qj<0), we have xj+1=xj.

If U is chosen pixel (pi>0) we have xj+1=xj. Thus we can express

qj+1in terms of qj and (xj+1,yj+1 ):  
qj+1=qj+2b2 xj+1-2a2 yj+1+a2          if qj < 0  
=qj-2a2 yj+1+a2          if qj>0

The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of (xk yk) of the last pixel choosen for the part 1 of the curve:

q1 = f(xk+Midpoint Ellipse Algorithm,yk-1)=b2 (xk+Midpoint Ellipse Algorithm)2-a2 (yk-1)2- a2 b2

**Algorithm:**

**int x=0, y=b; [starting point]**

**int fx=0, fy=2a2 b [initial partial derivatives]**

**int p = b2-a2 b+a2/4**

**while (fx<="" 1="" {="" set="" pixel="" (x,="" y)="" x++;="" fx="fx" +="" 2b2;**

**if (p<0)**

**p = p + fx +b2;**

**else**

**{**

**y--;**

**fy=fy-2a2**

**p = p + fx +b2-fy;**

**}**

**}**

**Setpixel (x, y);**

**p=b2(x+0.5)2+ a2 (y-1)2- a2 b2**

**while (y>0)**

**{**

**y--;**

**fy=fy-2a2;**

**if (p>=0)**

**p=p-fy+a2**

**else**

**{**

**x++;**

**fx=fx+2b2**

**p=p+fx-fy+a2;**

**}**

**Setpixel (x,y);**

**}**

**Program to draw an ellipse using Midpoint Ellipse Algorithm:**

1. **#include <graphics.h>**
2. **#include <stdlib.h>**
3. **#include <math.h>**
4. **#include <stdio.h>**
5. **#include <conio.h>**
6. **#include <iostream.h>**
8. **class bresen**
9. **{**
10. **float x,y,a, b,r,p,h,k,p1,p2;**
11. **public:**
12. **void get ();**
13. **void cal ();**
14. **};**
15. **void main ()**
16. **{**
17. **bresen b;**
18. **b.get ();**
19. **b.cal ();**
20. **getch ();**
21. **}**
22. **void bresen :: get ()**
23. **{**
24. **cout<<"\n ENTER CENTER OF ELLIPSE";**
25. **cout<<"\n ENTER (h, k) ";**
26. **cin>>h>>k;**
27. **cout<<"\n ENTER LENGTH OF MAJOR AND MINOR AXIS";**
28. **cin>>a>>b;**
29. **}**
30. **void bresen ::cal ()**
31. **{**
32. **/\* request auto detection \*/**
33. **int gdriver = DETECT,gmode, errorcode;**
34. **int midx, midy, i;**
35. **/\* initialize graphics and local variables \*/**
36. **initgraph (&gdriver, &gmode, " ");**
37. **/\* read result of initialization \*/**
38. **errorcode = graphresult ();**
39. **if (errorcode ! = grOK)    /\*an error occurred \*/**
40. **{**
41. **printf("Graphics error: %s \n", grapherrormsg (errorcode);**
42. **printf ("Press any key to halt:");**
43. **getch ();**
44. **exit (1); /\* terminate with an error code \*/**
45. **}**
46. **x=0;**
47. **y=b;**
48. **// REGION 1**
49. **p1 =(b \* b)-(a \* a \* b) + (a \* a)/4);**
50. **{**
51. **putpixel (x+h, y+k, RED);**
52. **putpixel (-x+h, -y+k, RED);**
53. **putpixel (x+h, -y+k, RED);**
54. **putpixel (-x+h, y+k, RED);**
55. **if (p1 < 0)**
56. **p1 += ((2 \*b \* b) \*(x+1))-((2 \* a \* a)\*(y-1)) + (b \* b);**
57. **else**
58. **{**
59. **p1+= ((2 \*b \* b) \*(x+1))-((2 \* a \* a)\*(y-1))-(b \* b);**
60. **y--;**
61. **}**
62. **x++;**
63. **}**
64. **//REGION 2**
65. **p2 =((b \* b)\* (x + 0.5))+((a \* a)\*(y-1) \* (y-1))-(a \* a \*b \* b);**
66. **while (y>=0)**
67. **{**
68. **If (p2>0)**
69. **p2=p2-((2 \* a \* a)\* (y-1))+(a \*a);**
70. **else**
71. **{**
72. **p2=p2-((2 \* a \* a)\* (y-1))+((2 \* b \* b)\*(x+1))+(a \* a);**
73. **x++;**
74. **}**
75. **y--;**
76. **putpixel (x+h, y+k, RED);**
77. **putpixel (-x+h, -y+k, RED);**
78. **putpixel (x+h, -y+k, RED);**
79. **putpixel (-x+h, y+k, RED);**
80. **}**
81. **getch();**
82. **}**

**Output:**

