Bresenham's Line Example

Let's have a go at this

Let's plot the line from (20, 10) to (30, 17)

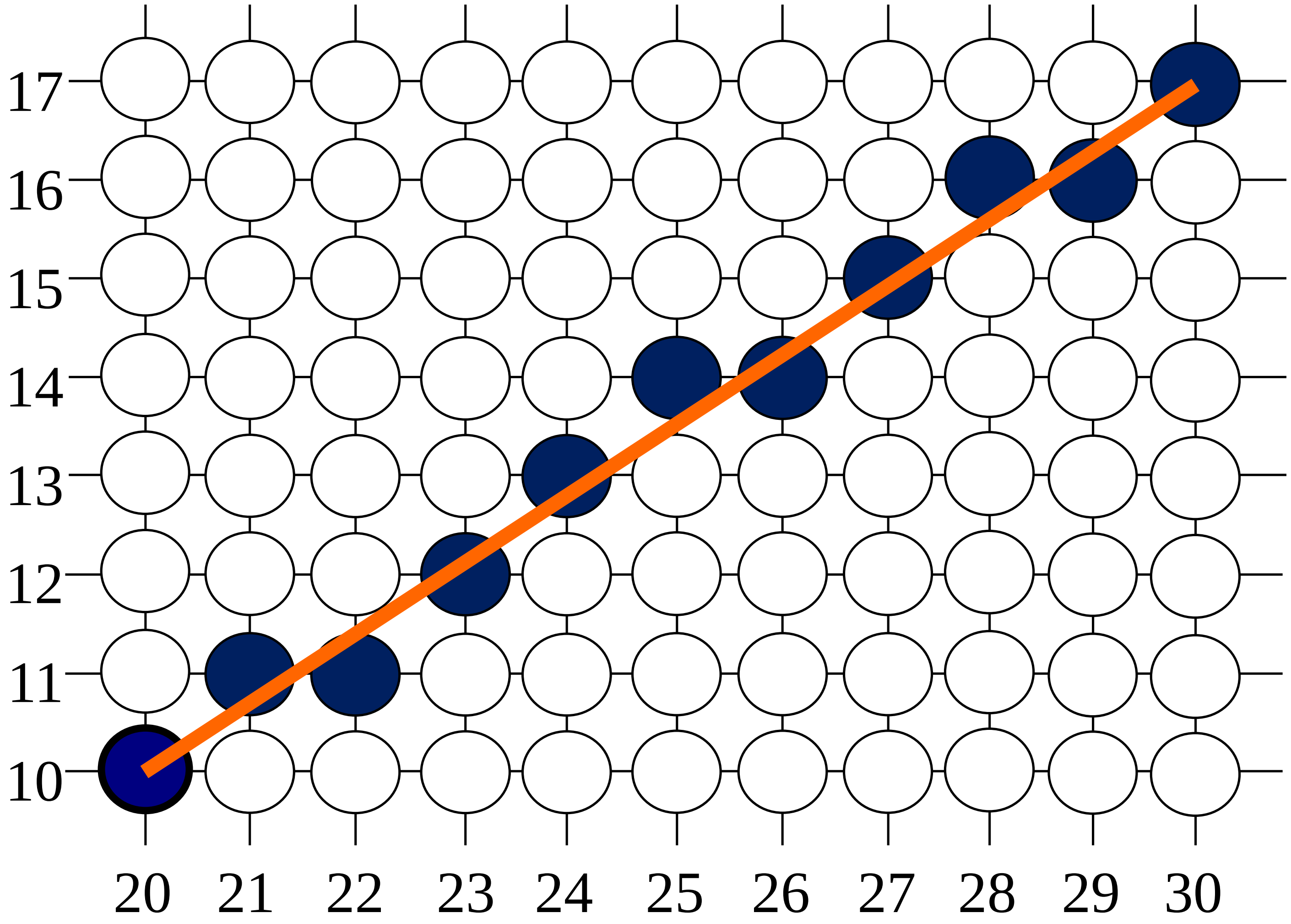
First off calculate all of the constants:

- Δx: 30-20 = 10
- Δy: 18-10 = 7
- ΔE: $2\Delta y$ = 14
- ΔNE: $2\Delta y - 2\Delta x$ = –6

Calculate the initial decision parameter d_{init}:

- d_{init}: $2\Delta y - \Delta x = 4$

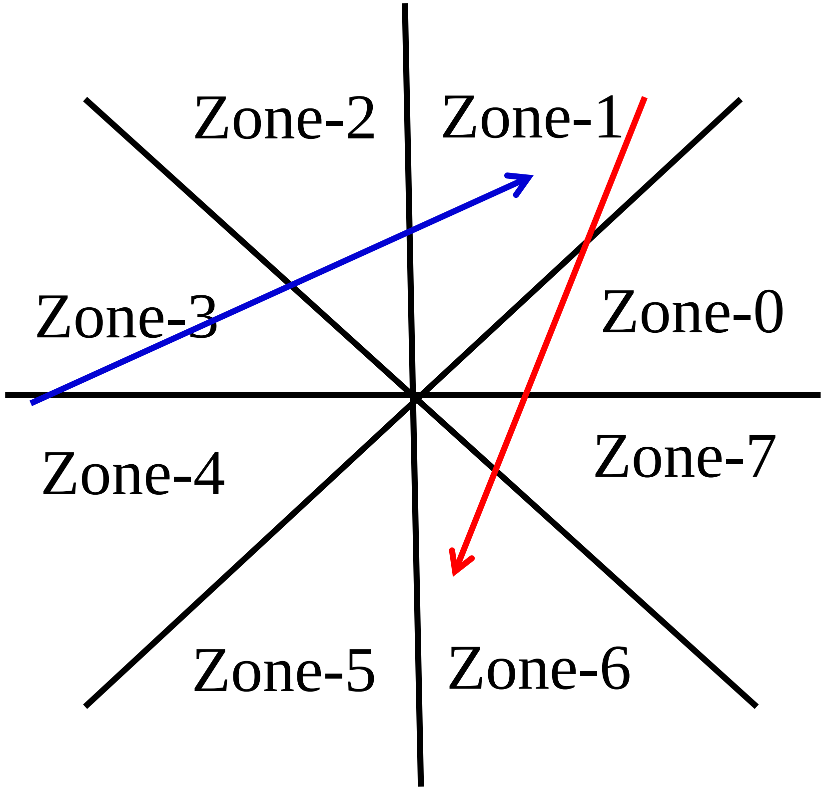
Bresenham's Line Example



i	(x_i, y_i)	d_i	ΔE/ΔNE
0	(20, 10)	4	ΔNE
1	(21, 11)	-2	ΔE
2	(22, 11)	12	ΔNE
3	(23, 12)	6	ΔNE
4	(24, 13)	0	ΔNE
5	(25, 14)	-6	ΔE
6	(26, 14)	8	ΔNE
7	(27, 15)	2	ΔNE
8	(28, 16)	-4	ΔE
9	(29, 16)	10	ΔNE
10	(30, 17)	4	ΔNE

Slope independent Line

- Bresenham's algorithm is very much slope dependent.
- Let us try to make it slope independent.
- If we think line as a vector, then lines can be grouped into following 8 zones.
- The red line in the figure is for zone-5.
- Whereas the blue line is for zone-0
- Let us make an algorithm to find the slope based zone of a line.



Slope Determination Algorithm

```
def slope(x0, y0, x1, y1):
    dx = x1 - x0
    dy = y1 - y0
    print("Dx =", dx, "and Dy =", dy)
    if abs(dx) >= abs(dy):  # zone 0, 3, 4, and 7
    if dx >= 0:
        if dy >= 0:
            zone = 0
        else:
            zone = 7
    else:
        if dy >= 0:
            zone = 3
        else:
            zone = 4
```

```
    else:  # zone 1, 2, 5, and 6
        if dx >= 0:
            if dy >= 0:
                zone = 1
            else:
                zone = 6
        else:
            if dy >= 0:
                zone = 2
            else:
                zone = 5
    print("The line is in Zone", zone)
slope(34, 23, -40, 36)
```

Output: Dx = -74 and Dy = 13
The line is in Zone 3

Slope Independent Line

There can have to ways to make the line-drawing algorithm slope independent:

1. **Simple way:** In this algorithm
   1. First the zone of the line will be selected (like above program) ;
   2. Then it will call the line-drawing function for respective zone. i.e., if the line is in zone-3, instead of writing **zone = 3**, it will call **drawLine_3(x0, y0, x1, y1)** and so on.
   3. The program size is bigger here as there are 8 separate functions are required for 8 zones.
   4. Though the program size is big, it is efficient enough.

Slope Independent Line

The other way of making the line-drawing algorithm slope independent is:

2. **Utilizing 8-way Symmetry:**
   - 8-way symmetry refers to a type of symmetry characterized by the presence of eight-fold rotational symmetry.
   - It means that if we choose any point on the circle, then by changing the sign of the coordinate, we get 3 more points on the circle.
   - Again by swapping the coordinates we get 4 more points.
   - i.e., a point has 8 replications like Fig.1

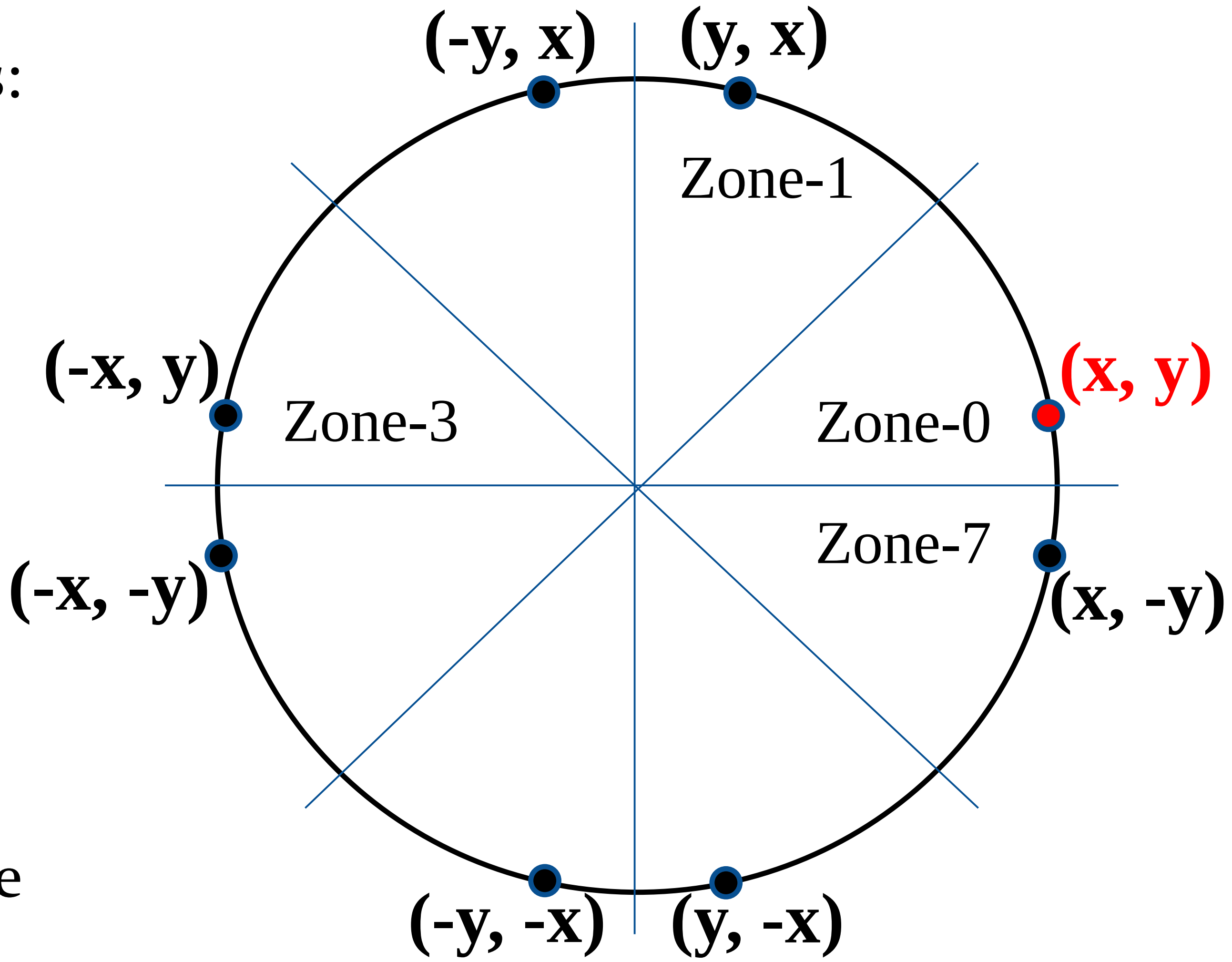


Fig.1: 8-way Symmetry (Zone-0 to others)

Slope Independent Line

- 8-way symmetry can be is used to improve circle drawing efficiency.
- It is seen in the fig.1 that a point in one octant, it can be easily shifted to other octants by changing coordinate sign or swapping.
- By doing so, lines from all octants can be brought into one octant for processing, (1st-all zone to zone-0), which can simplify calculations and make certain operations easier. That is we will use only **drawLine_0**() for all cases.
- After the necessary computations are performed in the transformed octant, the lines can be returned to their original octants for plotting, (2nd-zone-0 to previous zone)
- This is the core idea of utilizing this symmetry to make slope independent line.

Slope Independent Line

This a important to remember that, the 2 ways of shifting may not be easy:

- Fig.1 show the ways of shifting from zone-0 to other zones, and **Fig.2** show the ways of shifting from zone-2 to other zones.
- To make it, we have to remember that always source zone is assumed as (x, y).

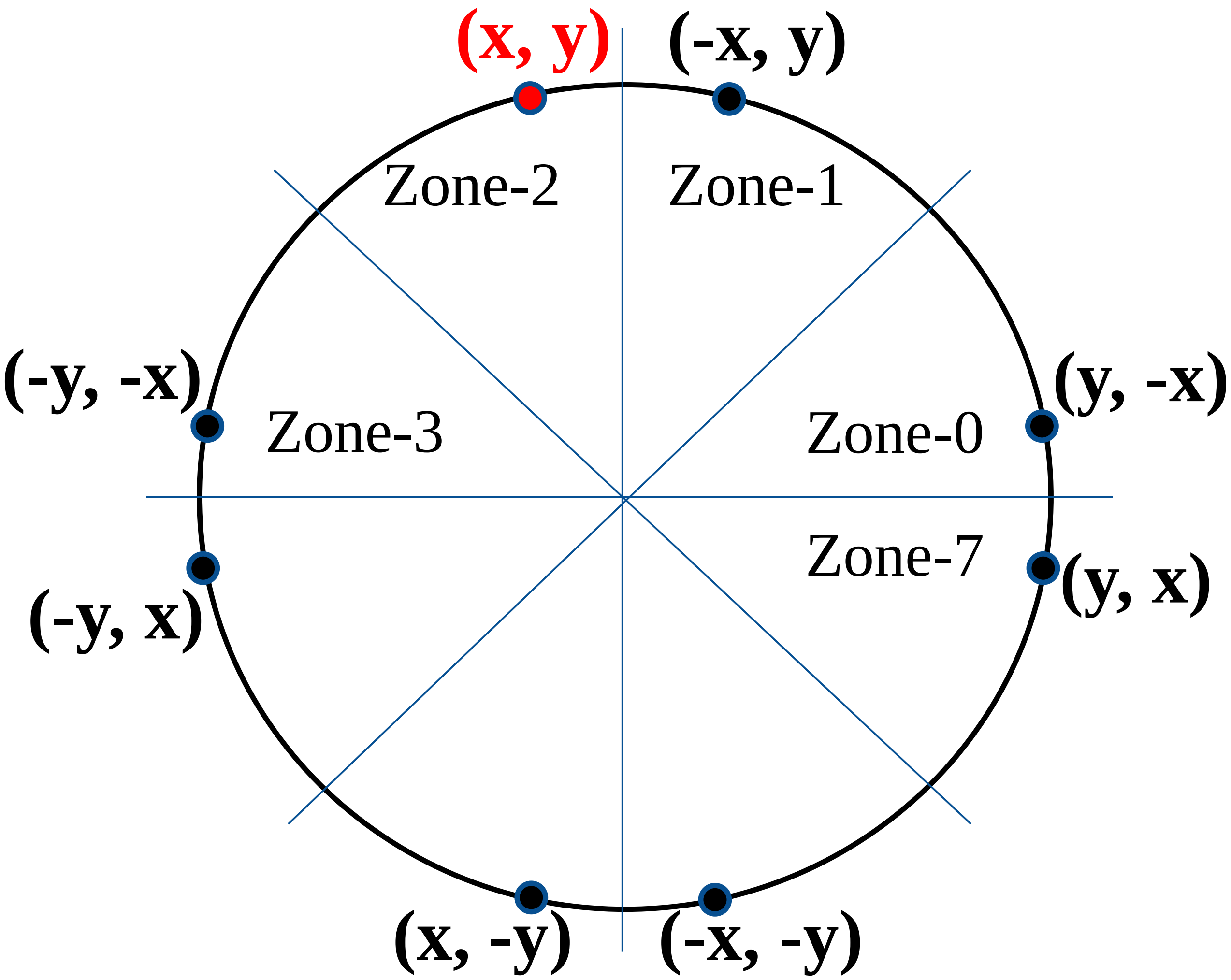


Fig.2: 8-way Symmetry (Zone-2 to others)

Slope Independent Line

Now using this shifting concept:

- If the line is in zone-6, we will call **drawLine_0(-y0, x0, -y1, x1, 6),** which is actually shifted to zone-0, (see Fig.3)
- The 5th passing parameter in the above function will not interfere processing, but will help to plot the line in zone-6.
- So, the drawPixel() in drawLine_0() need to modify so that the line can be plotted in the original zone.

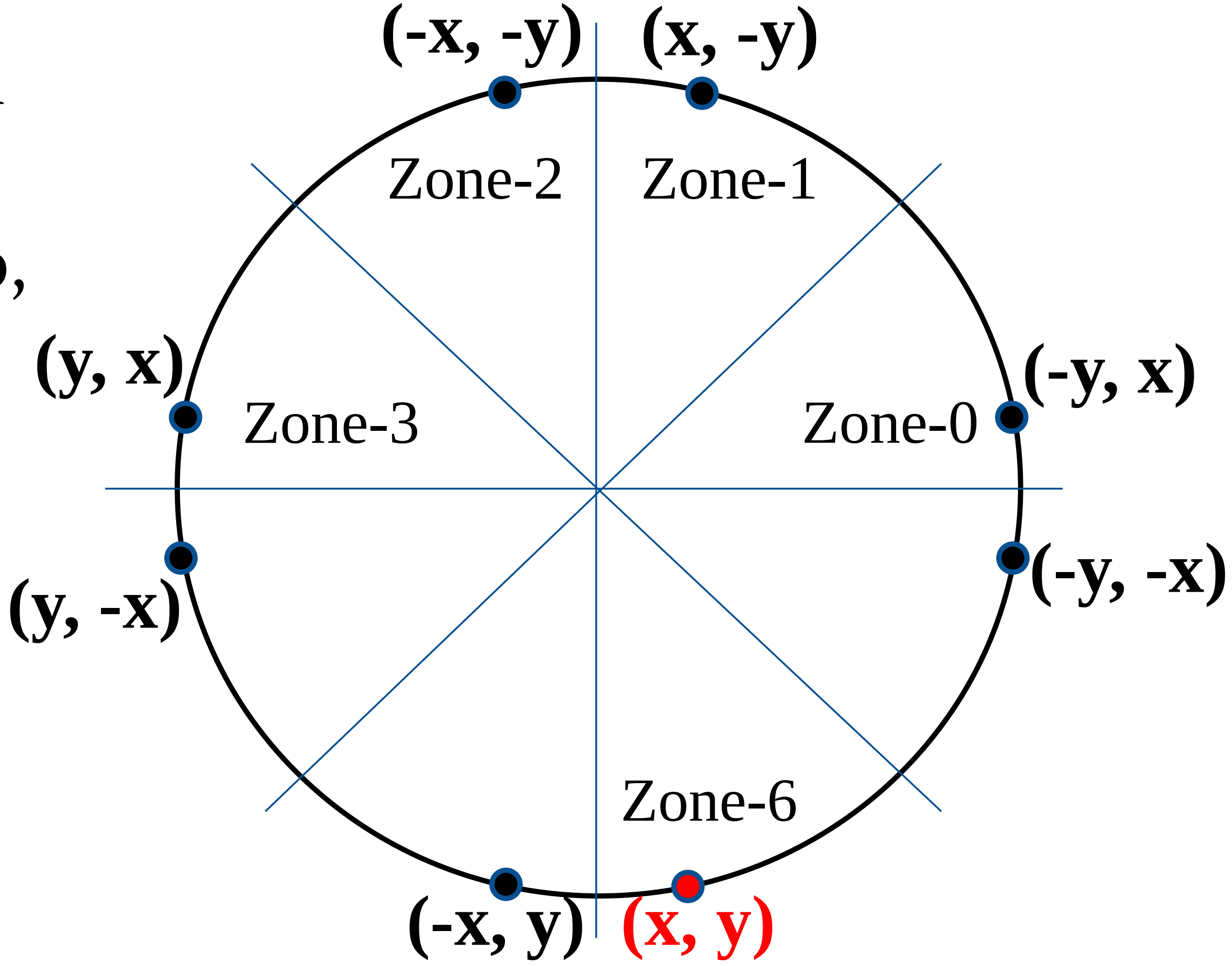


Fig.3: 8-way Symmetry (Zone-6 to others)

Slope Independent Line

Lets rewrite drawPixel() as shifted in **Fig.1** (Zone-0 to others):

```
def draw8way(x, y, slope):
    if slope==0:
        drawPixel(x, y)
    if slope==1:
        drawPixel(y, x)
    ...
    if slope==7
        drawPixel(x, -y)
```

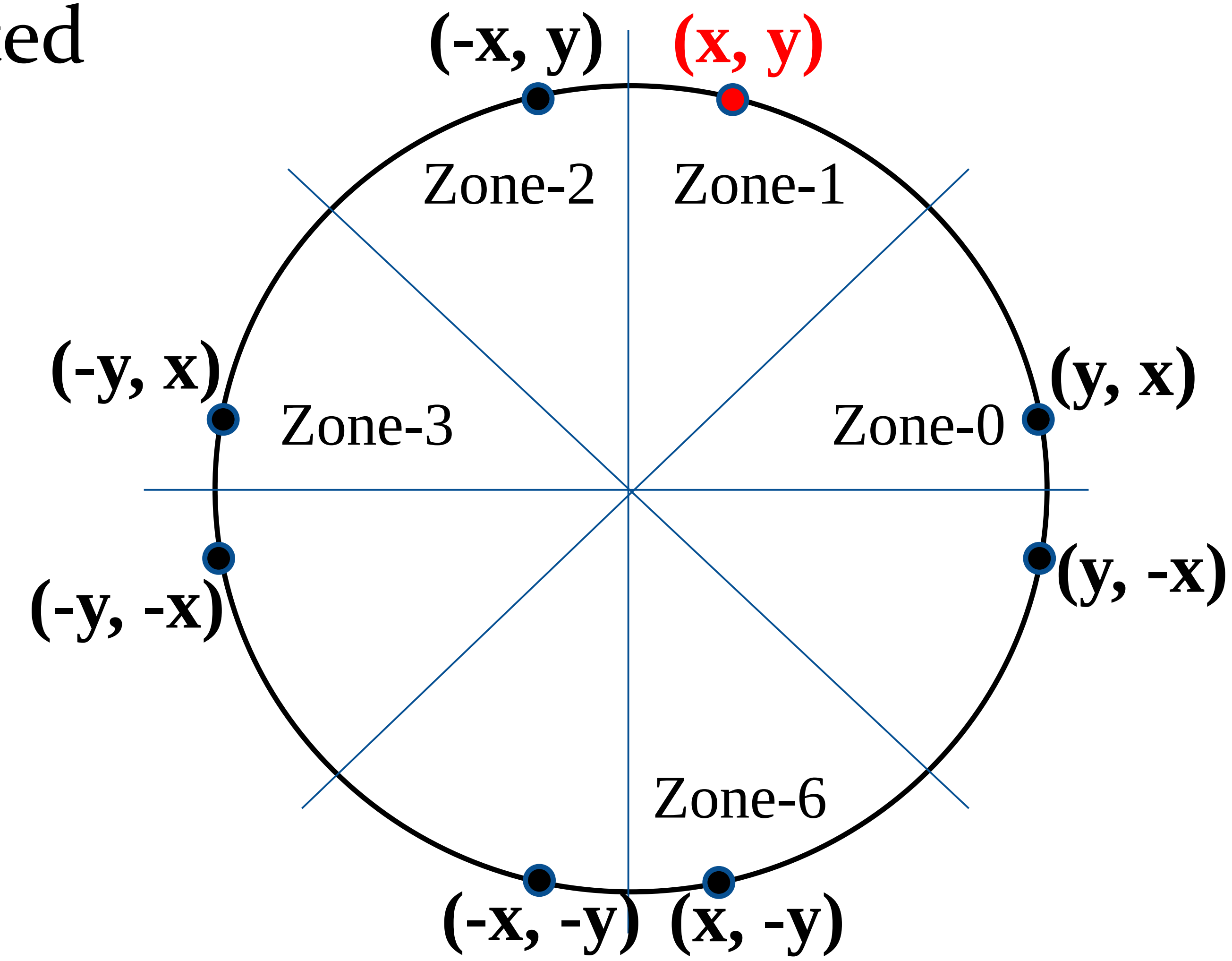


Fig.4: 8-way Symmetry (Zone-1 to others)

Code for implementing slope-independent line

all zone to zone-0

```
def drawLine(x0, y0, x1, y1):
    dx = x1 - x0
    dy = y1 - y0
    if abs(dx) >= abs(dy):  # zone 0, 3, 4, and 7
        if dx >= 0:
            if dy >= 0:
                drawLine_0(x0, y0, x1, y1, 0)
            else:
                drawLine_0(x0, y0, -x1, -y1, 7)
        else:
            if dy >= 0:
                drawLine_0(-x0, y0, -x1, y1, 3)
            else:
                drawLine_0(-x0, -y0, -x1, -y1, 4)
    else:  # zone 1, 2, 5, and 6
        if dx >= 0:
            if dy >= 0:
                drawLine_0(y0, x0, y1, x1, 1)
            else:
                drawLine_0(-y0, x0, -y1, x1, 6)
        else:
            if dy >= 0:
                drawLine_0(y0, -x0, y1, -x1, 2)
            else:
                drawLine_0(-y0, -x0, -y1, -x1, 5)

drawLine(34, 23, -40, 36)
```

Code for implementing slope-independent line

```
def drawLine_0(x0, y0, x1, y1, slope):
        dx = x1 - x0
        dy = y1 - y0
        delE = 2 * dy
        delNE = 2 * (dy - dx)
        d = 2 * dy - dx
        x = x0
        y = y0
        while x < x1:
                draw8way(x, y, slope)
                        if d < 0:
                                d += delE
                                x += 1
                        else:
                                d += delNE
                                x += 1
                                y += 1
```

all zone to zone-0

```
def draw8way(x, y, slope):
        switch slope:
                case 0:
                        drawPixel(x, y)
                case 1:
                        drawPixel(y, x)
                ….
                ….
                case 7:
                        drawPixel(x, -y)
```