# EXPERIMENT-1

Implement Bresenham’s line algorithm, Midpoint Circle Algorithm, and Midpoint Ellipse Algorithm. Draw different geometric objects on the screen.

**Midpoint Ellipse Algorithm:**

import matplotlib.pyplot as plt

def plot\_ellipse\_points(xc, yc, x, y, points):

# Plot all 4 symmetrical points

points.extend([

(xc + x, yc + y),

(xc - x, yc + y),

(xc + x, yc - y),

(xc - x, yc - y),

])

def midpoint\_ellipse(rx, ry, xc, yc):

x = 0

y = ry

points = []

rx2 = rx \* rx

ry2 = ry \* ry

two\_rx2 = 2 \* rx2

two\_ry2 = 2 \* ry2

p1 = ry2 - (rx2 \* ry) + (0.25 \* rx2)

dx = 0

dy = two\_rx2 \* y

while dx < dy:

plot\_ellipse\_points(xc, yc, x, y, points)

x += 1

dx += two\_ry2

if p1 < 0:

p1 += ry2 + dx

else:

y -= 1

dy -= two\_rx2

p1 += ry2 + dx - dy

p2 = (ry2) \* ((x + 0.5) \*\* 2) + (rx2) \* ((y - 1) \*\* 2) - (rx2 \* ry2)

while y >= 0:

plot\_ellipse\_points(xc, yc, x, y, points)

y -= 1

dy -= two\_rx2

if p2 > 0:

p2 += rx2 - dy

else:

x += 1

dx += two\_ry2

p2 += dx - dy + rx2

return points

rx, ry = 50, 30

xc, yc = 0, 0

ellipse\_points = midpoint\_ellipse(rx, ry, xc, yc)

x\_coords, y\_coords = zip(\*ellipse\_points)

plt.figure(figsize=(6, 6))

plt.plot(x\_coords, y\_coords, 'bo') # Blue dots

plt.axhline(0, color='gray', linestyle='--')

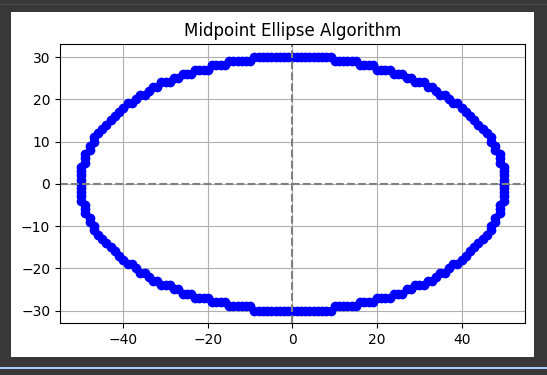
plt.axvline(0, color='gray', linestyle='--')

plt.gca().set\_aspect('equal', adjustable='box')

plt.title("Midpoint Ellipse Algorithm")

plt.grid(True)

plt.show()



**Midpoint Circle Algorithm:**

import matplotlib.pyplot as plt

def midpoint\_circle(r):

x = 0

y = r

p = 1 - r

points = []

while x <= y:

points.extend([

(x, y), (y, x), (-x, y), (-y, x),

(-x, -y), (-y, -x), (x, -y), (y, -x)

])

x += 1

if p < 0:

p += 2 \* x + 1

else:

y -= 1

p += 2 \* (x - y) + 1

return points

radius = 20

circle\_points = midpoint\_circle(radius)

x\_coords = [pt[0] for pt in circle\_points]

y\_coords = [pt[1] for pt in circle\_points]

plt.figure(figsize=(6,6))

plt.scatter(x\_coords, y\_coords, color='blue', s=10)

plt.axhline(0, color='black', linewidth=0.8)

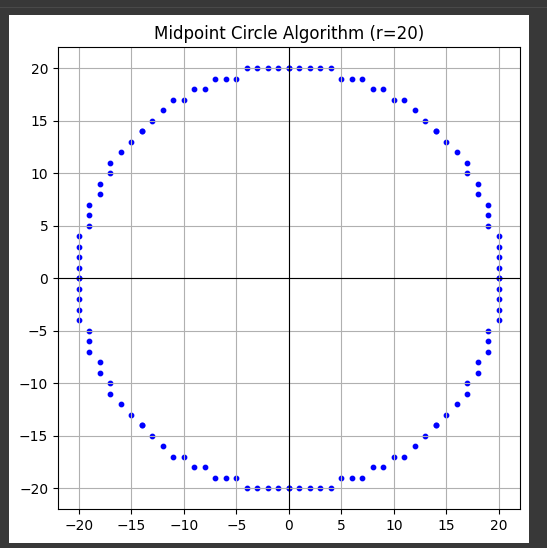
plt.axvline(0, color='black', linewidth=0.8)

plt.title(f'Midpoint Circle Algorithm (r={radius})')

plt.gca().set\_aspect('equal') # Equal aspect ratio for circle shape

plt.grid(True)

plt.show()



**Bresenham's Line Algorithm:**

import matplotlib.pyplot as plt

def bresenham\_line(x0, y0, x1, y1):

points = []

dx = abs(x1 - x0)

dy = abs(y1 - y0)

x, y = x0, y0

sx = 1 if x0 < x1 else -1

sy = 1 if y0 < y1 else -1

if dx > dy:

err = dx / 2.0

while x != x1:

points.append((x, y))

err -= dy

if err < 0:

y += sy

err += dx

x += sx

else:

err = dy / 2.0

while y != y1:

points.append((x, y))

err -= dx

if err < 0:

x += sx

err += dy

y += sy

points.append((x1, y1))

return points

x\_start, y\_start = 2, 3

x\_end, y\_end = 15, 10

line\_points = bresenham\_line(x\_start, y\_start, x\_end, y\_end)

x\_coords, y\_coords = zip(\*line\_points)

plt.figure(figsize=(6,6))

plt.plot(x\_coords, y\_coords, 'r-', linewidth=2) # Red continuous line

plt.scatter(x\_coords, y\_coords, color='blue') # Blue dots on points

plt.grid(True)

plt.title("Bresenham's Line Algorithm")

plt.axhline(0, color='gray', linestyle='--')

plt.axvline(0, color='gray', linestyle='--')

plt.gca().set\_aspect('equal', adjustable='box')

plt.show()

