1.5.30

EE25BTECH11043 - Nishid Khandagre

August 28, 2025

Question

If the coordinates of one end of a diameter of a circle are $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ and the coordinates of its centre are $\begin{pmatrix} -2 \\ 5 \end{pmatrix}$, then the coordinates of the other end of the diameter are?

Theoretical Solution

Let the coordinates of the known end of the diameter be vector **B**. Let the coordinates of the center of the circle be vector **P**. Let the coordinates of the other end of the diameter be vector **A** (the required vector).

Given:

$$\mathbf{B} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{1}$$

$$\mathbf{P} = \begin{pmatrix} -2 \\ 5 \end{pmatrix} \tag{2}$$

Equation

The center of a circle is the midpoint of its diameter. For a circle with center **P** and ends of diameters represented by vectors **A** and **B**, the relationship is:

$$\mathbf{P} = \frac{\mathbf{A} + \mathbf{B}}{2} \tag{3}$$

Theoretical Solution

To find vector **A**, we use the midpoint formula. We know that **P** divides diameter **AB** in ratio 1:1.
Substituting the given values into the equation:

$$\mathbf{P} = \frac{\mathbf{A} + \begin{pmatrix} 2 \\ 3 \end{pmatrix}}{2} \tag{4}$$

$$2 \cdot \begin{pmatrix} -2 \\ 5 \end{pmatrix} = \mathbf{A} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} -4 \\ 10 \end{pmatrix} = \mathbf{A} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{6}$$

Theoretical Solution

Rearranging the terms to solve for **A**:

$$\mathbf{A} = \begin{pmatrix} -4 \\ 10 \end{pmatrix} - \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{7}$$

$$\mathbf{A} = \begin{pmatrix} -4-2 \\ 10-3 \end{pmatrix} \tag{8}$$

Hence, the coordinates of the other end of the diameter are:

$$\mathbf{A} = \begin{pmatrix} -6 \\ 7 \end{pmatrix} \tag{9}$$

C Code

```c
#include <stdio.h>

// Function to calculate the coordinates of the other end of the diameter
void formula(double x1, double y1, double xc, double yc, double *x2, double *y2) {
    *x2 = 2 * xc - x1;
    *y2 = 2 * yc - y1;
}
```

C Code

```c
int main() {
    double x1 = 2;
    double y1 = 3;
    double xc = -2;
    double yc = 5;
    double x2, y2;

    formula(x1, y1, xc, yc, &x2, &y2);

    return 0;
}
```

Python Code through shared output

```python
import ctypes
import numpy as np
import matplotlib.pyplot as plt

# Load the shared library
lib_diameter = ctypes.CDLL(./code.so)

# Define the argument types and return type for the C function
lib_diameter.findOtherEndOfDiameter.argtypes = [
    ctypes.c_double, # x1
    ctypes.c_double, # y1
    ctypes.c_double, # xc
    ctypes.c_double, # yc
    ctypes.POINTER(ctypes.c_double), # x2
    ctypes.POINTER(ctypes.c_double) # y2
```

Python Code through shared output

```
]
lib_diameter.findOtherEndOfDiameter.restype = None

# Given coordinates
x1_given, y1_given = 2.0, 3.0 # One end of the diameter
xc_given, yc_given = -2.0, 5.0 # Center of the circle

# Create ctypes doubles to hold the results
x2_result = ctypes.c_double()
y2_result = ctypes.c_double()

# Call the C function to find the other end of the diameter
lib_diameter.findOtherEndOfDiameter(
    x1_given, y1_given,
    xc_given, yc_given,
    ctypes.byref(x2_result),
    ctypes.byref(y2_result)
)
```

Python Code through shared output

```

x2_found = x2_result.value
y2_found = y2_result.value

print(fThe coordinates of the other end of the diameter are ({x2_found:.2f}, {y2_found:.2f}))

# Calculate the radius for plotting the circle
radius = np.sqrt((x1_given - xc_given)**2 + (y1_given - yc_given)**2)

# Generate points for the circle
theta = np.linspace(0, 2 * np.pi, 200)
circle_x = xc_given + radius * np.cos(theta)
circle_y = yc_given + radius * np.sin(theta)
```

Python Code through shared output

```python
# Plotting
plt.figure(figsize=(8, 8))

# Plot the circle
plt.plot(circle_x, circle_y, 'b-', label='Circle')

# Plot the given diameter end
plt.scatter(x1_given, y1_given, color='red', s=100, zorder=5)
plt.annotate(f'({x1_given},{y1_given})', (x1_given, y1_given),
    textcoords=offset points, xytext=(5,5), ha='left')
```

Python Code through shared output

```
# Plot the center
plt.scatter(xc_given, yc_given, color='green', s=100, zorder=5)
plt.annotate(f'({xc_given},{yc_given})', (xc_given, yc_given),
    textcoords=offset points, xytext=(5,5), ha='left')

# Plot the calculated other end of the diameter
plt.scatter(x2_found, y2_found, color='purple', s=100, zorder=5)
plt.annotate(f'({x2_found:.2f},{y2_found:.2f})', (x2_found,
    y2_found), textcoords=offset points, xytext=(5,5), ha='left')
```

Python Code through shared output

```python

plt.plot([x1_given, x2_found], [y1_given, y2_found], 'r--', label
    ='Diameter')

plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Circle and its Diameter')
plt.grid(True)
plt.legend()
plt.show()
```

Python Code : Direct

```python
import sys
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt


def line_gen_num(A, B, num_points):

    A = A.flatten()
    B = B.flatten()
    t = np.linspace(0, 1, num_points)
    points = np.outer(A, (1-t)) + np.outer(B, t)
    return points
```

Python Code : Direct

```python

def circ_gen(center, radius, num_points=100):

    center = center.flatten()
    theta = np.linspace(0, 2*np.pi, num_points)
    x = center[0] + radius * np.cos(theta)
    y = center[1] + radius * np.sin(theta)
    return np.array([x, y])

# Given coordinates
B = np.array([2, 3]).reshape(-1, 1)
P = np.array([-2, 5]).reshape(-1, 1)
```

Python Code : Direct

```python

# Function to calculate the other end of the diameter
def func_other_end(center, one_end):
    return 2 * center - one_end

# Function to calculate the radius
def func_radius(center, point_on_circumference):
    return LA.norm(center - point_on_circumference)

# Calculate the other end of the diameter (A)
A = func_other_end(P, B).reshape(-1, 1)

# Calculate the radius of the circle
radius = func_radius(P, B)
```

Python Code : Direct

```

print(fThe coordinates of the other end of the diameter are ({A
    [0,0]}, {A[1,0]}))

# Generate points for the diameter line
x_AB = line_gen_num(A, B, 20)

# Generate points for the circle
x_circ = circ_gen(P, radius)

# Plotting
plt.plot(x_circ[0,:], x_circ[1,:], red, label=Circle)
plt.plot(x_AB[0,:], x_AB[1,:], g--, label=Diameter)

# Plot the points
tri_coords = np.block([[A, B, P]])
plt.scatter(tri_coords[0,:], tri_coords[1,:], s=50, zorder=5) # s
     for size, zorder to ensure visibility
```

Python Code : Direct

```python

# Add labels to the points
vert_labels = [f'A({A[0,0]:.0f},{A[1,0]:.0f})', f'B({B[0,0]:.0f},{B[1,0]:.0f})', f'P({P[0,0]:.0f},{P[1,0]:.0f}) (Center)']
for i , txt in enumerate(vert_labels):
    plt.annotate(txt, (tri_coords[0,i], tri_coords[1,i]), textcoords=offset points, xytext=(5,5), ha='left')

plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.title(Diameter of a Circle)
plt.axis('equal') # Important to make the circle appear circular
plt.savefig(fig1.png)
plt.show()

print(Figure saved as fig1.png)
```

Plot by Python only

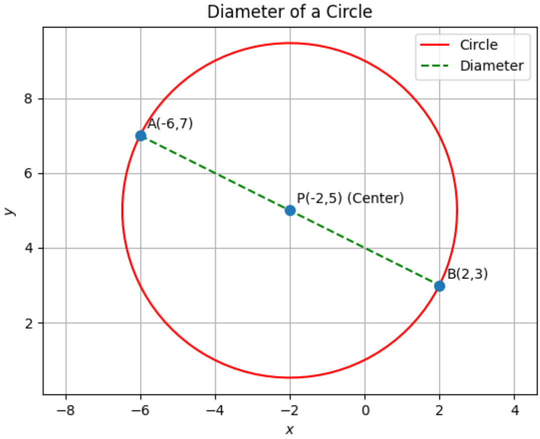
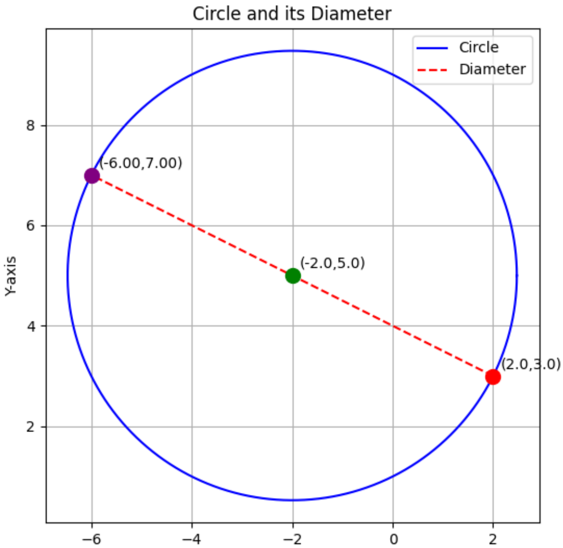


Figure:

Plot by Python using shared output from C


Circle and its Diameter
Circle
Diameter
(-6.00,7.00)
(-2.0,5.0)
(2.0,3.0)
Y-axis