1.5.20

Kartik Lahoti - EE25BTECH11032

August 25,2025

Question

Find the coordinates of a point **A** where **AB** is the diameter of the circle with center $\begin{pmatrix} -2 \\ 2 \end{pmatrix}$ and **B** is the point $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$.

Theoretical Solution

Let the vectors **A** be

$$\mathbf{A} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

Given,
Circle with Center (say) **P** and Diameter **AB**

$$\mathbf{B} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \mathbf{P} = \begin{pmatrix} -2 \\ 2 \end{pmatrix} \tag{2}$$

Equation

Center of a circle is the mid point of the diameter. For a circle with center O and ends of diameters represented by vectors A and B

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

Theoretical Solution

To find vector **A** , we know that **P** divides diameter **AB** in ratio 1 : 1

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

$$\begin{pmatrix} -2 \\ 2 \end{pmatrix} = \frac{\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 3 \\ 4 \end{pmatrix}}{2} \tag{5}$$

Theoretical Solution

Rearranging the terms , we get

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -7 \\ 0 \end{pmatrix} \tag{6}$$

Hence ,

$$\mathbf{A} = \begin{pmatrix} -7 \\ 0 \end{pmatrix} \tag{7}$$

C Code (1) - Function to find A matrix

```c
#include <stdio.h>
#include <math.h>
void func(double *P, double *B, double *A , int m )
{
    for ( int i = 0 ; i < m ; i++ )
    {
        A[i] = 2*P[i] - B[i] ;
    }

}
```

C Code (1) - Function to Find Radius

```c
double radius(double *P , double *B , int m )
{
    double sum = 0.0;
    for ( int i = 0 ; i < m ; i++ )
    {
        sum += pow(P[i]-B[i] , 2 );
    }
    return sqrt(sum) ;
}
```

C Code (2) - Function to Generate Points on Circle

```c
#include <math.h>

void circle_gen(double *X , double *Y , double *P, int n , double
     r)
{
// n is no. of points to generates. x stores x coor , y stores y
    coor
    for (int i = 0 ; i < n ; i++ )
    {
        double theta = 2.0 * M_PI * i / n ;
        X[i] = P[0] + r * cos(theta);
        Y[i] = P[1] + r * sin (theta);
    }
}
```

C Code (2) - Function to Generate Points on Line

```
void line_gen (double *X, double *Y , double *A , double *B , int
     n , int m )
{
    double temp[m] ;
    for (int i = 0 ; i < m ; i++)
    {
        temp [ i ] = (B[i]- A[i]) /(double) n ;
    }
    for (int i = 0 ; i <= n ; i++ )
    {
        X[i] = A[0] + temp[0] * i ;
        Y[i] = A[1] + temp[1] * i ;
    }
}
```

Python Code - Using Shared Object

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

handc1 = ctypes.CDLL("./formula.so")

handc1.func.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_int
]
handc1.func.restype = None # void function
```

Python Code - Using Shared Object

```
m = 2

P = np.array([[-2],[2]], dtype=np.float64)
B = np.array([[3],[4]], dtype=np.float64)
A = np.zeros(m, dtype=np.float64)

handc1.func(
    P.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    m #len(P) alternate
)
```

Python Code - Using Shared Object

```
handc1.radius.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_int
]

handc1.radius.restype = ctypes.c_double #return double

radius = handc1.radius(
    P.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    m
)
```

Python Code - Using Shared Object

```
handc2 = ctypes.CDLL("./circle_line.so")
handc2.line_gen.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_int,
    ctypes.c_int
]
handc2.line_gen.restype = None
n = 20,m = 2
X_l = np.zeros(n,dtype=np.float64)
Y_l = np.zeros(n,dtype=np.float64)
```

Python Code - Using Shared Object

```
handc2.line_gen(
    X_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    Y_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    n,m)

plt.figure()

#plotting line
plt.plot([X_l[0],X_l[-1]],[Y_l[0],Y_l[-1]],"g--",label="Diameter"
    )
plt.scatter(A[0],A[1],color = "red",s=50)
plt.scatter(B[0],B[1],color = "red",s=50)
plt.scatter(P[0],P[1],color = "red",s=50,label = "Center of
    Circle")
```

Python Code - Using Shared Object

```
handc2.circle_gen.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_int,
    ctypes.c_double]

handc2.circle_gen.restypes = None

n = 200
#r = radius

X_c = np.zeros(n,dtype=np.float64)
Y_c = np.zeros(n,dtype=np.float64)
```

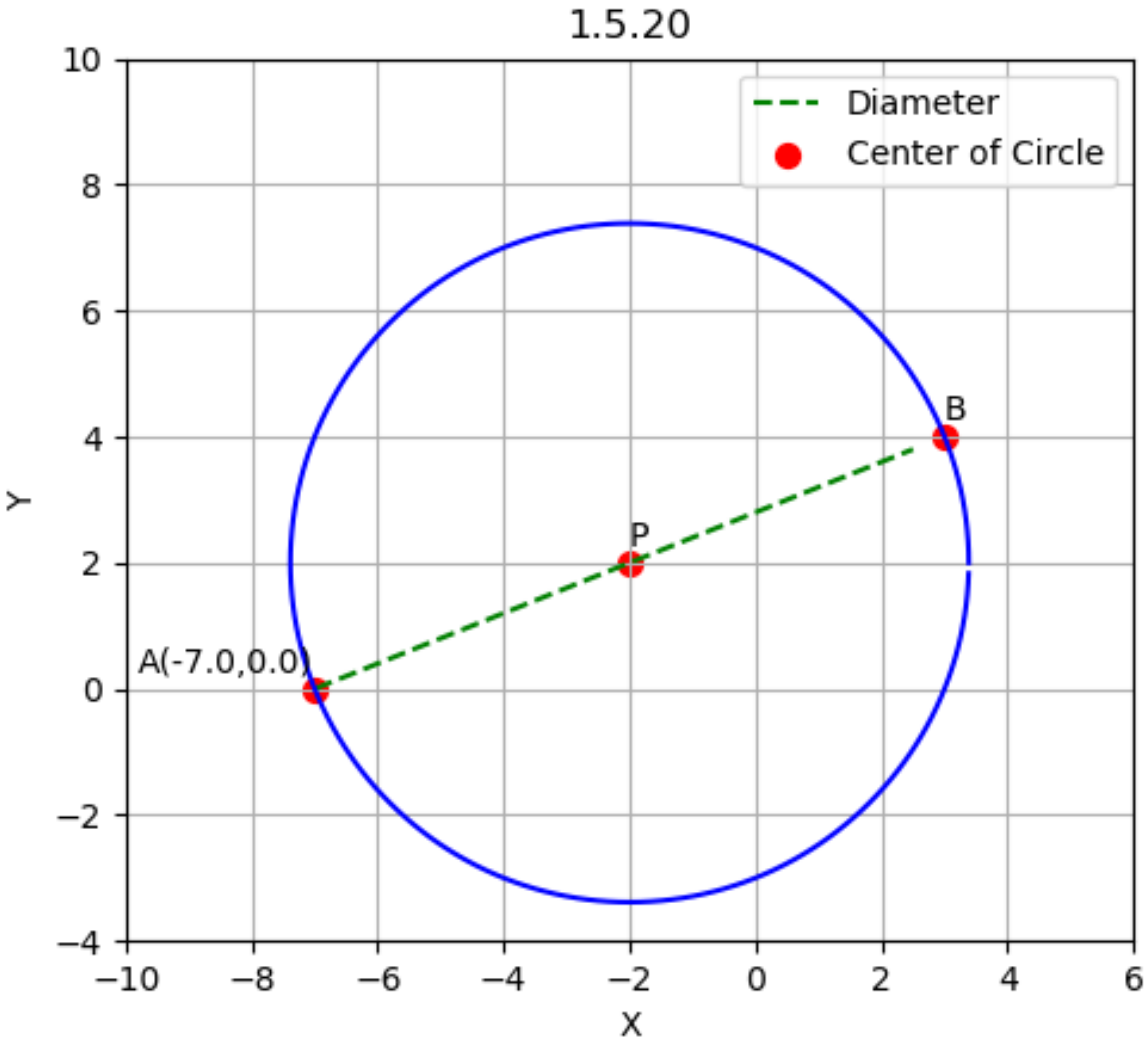
Python Code - Using Shared Object

```
handc2.circle_gen(

    X_c.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    Y_c.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    P.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    n , radius
)

#plotting circle
plt.plot(X_c,Y_c , "b-")

plt.annotate(f"A({A[0]},{A[1]})", (A[0], A[1]), textcoords="
    offset points", xytext=(0,5), ha='right')
plt.annotate("B", (B[0], B[1]), textcoords="offset points",
    xytext=(0,5), ha='left')
plt.annotate("P", (P[0], P[1]), textcoords="offset points",
    xytext=(0,5), ha='left')
```

Python Code - Using Shared Object

```
# Equal scaling for axes (important for circles!)
plt.gca().set_aspect("equal", adjustable="box")

plt.xlim([-10,6])
plt.ylim([-4,10])

# Labels and title
plt.xlabel("X")
plt.ylabel("Y")
plt.title("1.5.20")

plt.legend(loc = 'upper right')
plt.grid(True)

plt.savefig('figs/circle_graph.png')
subprocess.run(shlex.split("termux-open figs/circle_graph.png"))
```

Plot-Using Both C and Python



Python Code

```
import math
import sys #for path to external scripts
#sys.path.insert(0, '/home/kartik-lahoti/matgeo/codes/CoordGeo')
sys.path.insert(0, '/sdcard/matgeo/codes/CoordGeo')
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

#local imports
from line.funcs import *
from triangle.funcs import *
from conics.funcs import circ_gen

#if using termux
import subprocess
import shlex
```

Python Code - Using Shared Object

```
def func(P, B):
    # NumPy automatically applies operations to each element in
        the array
    return 2*P -B

def func_radius(P,B) :
    return LA.norm(P-B)

P = np.array([-2,2]).reshape(-1,1)
B = np.array([3,4]).reshape(-1,1)

A = func(P,B).reshape(-1,1)

x_AB = line_gen_num(A,B,20)
radius = func_radius(P,B)
```

Python Code - Using Shared Object

```
    x_circ = circ_gen(P,radius)
plt.plot(x_circ[0,:],x_circ[1,:],"red",label="Circle")
plt.plot(x_AB[0,:],x_AB[1,:],"g--",label="Diameter")

tri_coords = np.block([[A,B,P]])
plt.scatter(tri_coords[0,:], tri_coords[1,:])
vert_labels = ['A','B','P']
for i , txt in enumerate(vert_labels):
    plt.annotate(txt,(tri_coords[0,i],tri_coords[1,i]),textcoords
        ="offset points", xytext=(0,10),ha='center')
```

Python Code - Using Shared Object

```
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid() # minor
plt.axis('equal')
#plt.savefig("../figs/circle_graph2.png")
#plt.show()

plt.savefig('/figs/circle_graph2.png')
subprocess.run(shlex.split("termux-open figs/circle_graph2.png"))
```

Plot-Using only Python