Matgeo Presentation

Vijaya Sreyas
AI24BTech11003
IIT Hyderabad

November 6, 2024

Contents

Problem Statement

Verify if the point $\mathbf{P}\,(-2, 4)$ lies on a circle of radius 6 and center $\mathbf{C}\,(3, 5)$.

Setup and Variable Definitions

Variable	Description	Value
$\mathbf{P}$	Given Point	$\begin{pmatrix}-2\\4\end{pmatrix}$
$\mathbf{C}$	Center of circle	$\begin{pmatrix}3\\5\end{pmatrix}$
r	Radius of circle	6

Table: Variables and given data

Circle Equation Setup

We know:

$$\mathbf{u} = -\mathbf{c}, f = \|\mathbf{u}\|^2 - r^2 \tag{3.1}$$

substituting numerical values in (3.1)

$$u = -\begin{pmatrix} 3 \\ 5 \end{pmatrix}, f = -2 \tag{3.2}$$

The equation of the circle is then obtained as

$$\|\mathbf{x}\|^2 - 2\begin{pmatrix} 3 \\ 5 \end{pmatrix}^\top \mathbf{x} - 2 = 0 \tag{3.3}$$

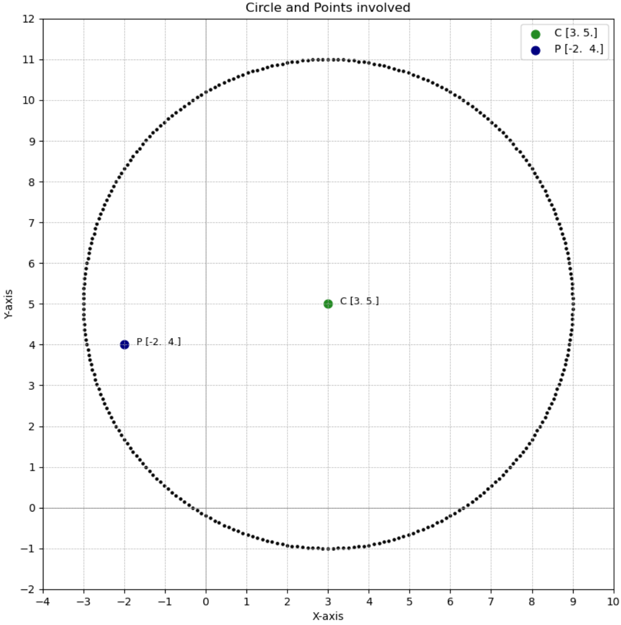
Checking Point Location

Now, by substituting the point **P** in (3.3), we can check where **P** is relative to the circle.

$$
\begin{aligned}
&= \left\| \begin{pmatrix} -2 \\ 4 \end{pmatrix} \right\|^2 - 2 \begin{pmatrix} 3 \\ 5 \end{pmatrix}^\top \begin{pmatrix} -2 \\ 4 \end{pmatrix} - 2 && (3.4) \\
&= 20 - 28 - 2 && (3.5) \\
&= -10 < 0 && (3.6)
\end{aligned}
$$

$\therefore$ we can say that the point **P** does not lie on the mentioned circle, but rather, inside it.

Figure



Circle and Points involved
C [3. 5.]
P [-2. 4.]
C [3. 5.]
P [-2. 4.]
Y-axis
X-axis
12
11
10
9
8
7
6
5
4
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
2
3
4
5
6
7
8
9
10

Generating points on Circle using C I

```c
#include <stdio.h>
#include <math.h>

#define NUM_POINTS 300 // Number of points on the circle

void calculateCirclePoints(double x_C, double y_C, double radius, FILE *file) {
    for (int i = 0; i < NUM_POINTS; i++) {
        double angle = 2 * M_PI * i / NUM_POINTS; // Angle in radians
        double x_p = x_C + radius * cos(angle); // xp coordinate
        double y_p = y_C + radius * sin(angle); // yp coordinate
        fprintf(file, "%.2f %.2f\n", x_p, y_p); // Write points to file
    }
}

int main() {
    // Pre-defined P, C coordinates and radius
    double x_C = 3.0; // Center x-coordinate
    double y_C = 5.0; // Center y-coordinate
    double radius = 6.0; // Radius
```

Generating points on Circle using C II

```
    double x_P = -2.0; // Point P x-coordinate
    double y_P = 4.0;  // Point P y-coordinate

    FILE *file = fopen("output.txt", "w"); //Open file
    if (file == NULL) {
        perror("Error opening file");
        return 1;
    }
    //Print P, C, and circle points
    fprintf(file, "P %.2f %.2f\n", x_P, y_P);
    fprintf(file, "C %.2f %.2f\n", x_C, y_C);
    calculateCirclePoints(x_C, y_C, radius, file);

    fclose(file); // Close the file

    return 0;
}

```

Plotting the figure using Python I

```python
import sys
sys.path.insert(0, '/home/vijaya-sreyas/IITH/EE1030/matgeo/codes/CoordGeo')
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from line.funcs import *
from conics.funcs import *
from triangle.funcs import *
import params
import matplotlib.pyplot as plt

# Read the output from the output.txt file
with open("output.txt", "r") as file:
    output_lines = file.strip().split('\n')

# Get the coordinates for points P and C
```

Plotting the figure using Python II

```python
point_P = np.array(list(map(float, output_lines[0].split()[1:])))  # P coordinates
point_C = np.array(list(map(float, output_lines[1].split()[1:])))  # C coordinates

# Get the circle points
data = np.array(np.vstack(list(map(lambda line: np.fromstring(line, sep=' '), output_lines[2:]))))

# Separate the circle points into x and y coordinates
xp, yp = data[:, 0], data[:, 1]

# Prepare for plotting
plt.figure(figsize=(8, 8))

# Plot the discrete circle points with smaller size
plt.scatter(xp, yp, color='k', marker='o', s=5)  # Smaller discrete circle points
plt.scatter(point_C[0], point_C[1], color='forestgreen', marker='o', s=60, label=f'C {point_C}')  # Center point (C)
```

Plotting the figure using Python III

```
plt.scatter(point_P[0], point_P[1], color='navy', marker='o', s=60,
↪  label=f'P {point_P}')  # Point (P)

# Label the points to the right
plt.text(point_C[0] + 0.3, point_C[1], f'C {point_C}', fontsize=9,
↪  ha='left')
plt.text(point_P[0] + 0.3, point_P[1], f'P {point_P}', fontsize=9,
↪  ha='left')

plt.title('Circle and Points involved')  # Updated title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set graph limits to ensure all points are visible
plt.xlim(-4, 10)  # X-axis limits
plt.ylim(-2, 12)  # Y-axis limits

# Add gridlines for both odd and even integers
plt.grid(which='both', linestyle='--', linewidth=0.5)
plt.xticks(np.arange(-4, 11, 1))  # Set x ticks for odd and even integers
```

Plotting the figure using Python IV

```python
plt.yticks(np.arange(-2, 13, 1))  # Set y ticks for odd and even integers

plt.gca().set_aspect('equal', adjustable='box')  # Equal aspect ratio
plt.axhline(0, color='grey', lw=0.5)
plt.axvline(0, color='grey', lw=0.5)
plt.legend()

# Save the plot as a PNG file
plt.savefig('plot.png')

#Close the plot
plt.close()
```