# 12.582

EE25BTECH11049-Sai Krishna Bakki

October 5, 2025

# Question

The position vector $OP$ of point $\mathbf{P} = (20{,}10)$ is rotated anti-clockwise in the X-Y plane by an angle $\theta = 30$ such that point $\mathbf{P}$ occupies position $\mathbf{Q}$. The coordinates (x,y) of $\mathbf{Q}$ is

# Theoretical Solution

Given

$$\mathbf{P} = \begin{pmatrix} 20 \\ 10 \end{pmatrix}, \theta = 30 \tag{1}$$

we use

$$\mathbf{x_n} = \mathbf{R}\mathbf{x_o} \tag{2}$$

where **R** is Rotation matrix

$$\mathbf{Q} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \mathbf{P} \tag{3}$$

$$\mathbf{Q} = \begin{pmatrix} \cos 30 & -\sin 30 \\ \sin 30 & \cos 30 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} \tag{4}$$

$$\mathbf{Q} = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} \tag{5}$$

$$\mathbf{Q} = \begin{pmatrix} 10\sqrt{3} - 5 \\ 10 + 5\sqrt{3} \end{pmatrix} \tag{6}$$

# Theoretical Solution

Using approximation, the coordinates of $\mathbf{Q}$ is

$$\mathbf{Q} = \begin{pmatrix} 12.32 \\ 18.66 \end{pmatrix} \tag{7}$$

# C Code

```c
#include <math.h>
// To ensure M_PI is defined, which is not standard in older C
    versions
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// Define a structure to hold 2D coordinates.
// This structure will be shared between C and Python.
typedef struct {
    double x;
    double y;
} Point;

void rotate_point_c(Point* p, double angle_degrees) {
    // Convert the angle from degrees to radians for C's math
        functions
    double angle_radians = angle_degrees * M_PI / 180.0;
```

# C Code

```
    // Store the original coordinates before overwriting them
    double x_old = p->x;
    double y_old = p->y;

    // Calculate the new coordinates using the standard 2D
        rotation formulas:
    // x_new = x_old * cos(theta) - y_old * sin(theta)
    // y_new = x_old * sin(theta) + y_old * cos(theta)
    p->x = x_old * cos(angle_radians) - y_old * sin(angle_radians
        );
    p->y = x_old * sin(angle_radians) + y_old * cos(angle_radians
        );
}
```

# Python Code Through Shared Output

```
import ctypes
import os
import matplotlib.pyplot as plt
from matplotlib.patches import Arc
import numpy as np # Required for the plotting function

# Define a Python class that mirrors the C 'Point' struct.
# This tells ctypes how to interpret the block of memory.
class Point(ctypes.Structure):
    _fields_ = [(x, ctypes.c_double),
               (y, ctypes.c_double)]

def plot_rotation(p, q, angle_degrees):

    Generates a plot to visualize the rotation of point P to Q.
    Args:
        p (tuple): The original (x, y) coordinates.
        q (tuple): The rotated (x, y) coordinates.
```

# Python Code Through Shared Output

```
        angle_degrees (float): The angle of rotation.

    fig, ax = plt.subplots(figsize=(8, 8))

    # Plot origin
    ax.plot(0, 0, 'ko', markersize=10, label='Origin (O)')

    # Create formatted strings for the labels
    p_label = f'({p[0]:.2f}, {p[1]:.2f})'
    q_label = f'({q[0]:.2f}, {q[1]:.2f})'

    # Plot vectors and points
    # Vector OP
    ax.arrow(0, 0, p[0], p[1], head_width=0.5, head_length=0.7, fc='blue', ec='blue', length_includes_head=True)
    ax.plot(p[0], p[1], 'bo', markersize=8, label=f'Point P {p_label}')
    ax.text(p[0] + 0.5, p[1] + 0.5, f'P {p_label}', fontsize=12, color='blue')
```

# Python Code Through Shared Output

```
    # Vector OQ
    ax.arrow(0, 0, q[0], q[1], head_width=0.5, head_length=0.7,
        fc='red', ec='red', length_includes_head=True)
    ax.plot(q[0], q[1], 'ro', markersize=8, label=f'Point Q {
        q_label}')
    ax.text(q[0] + 0.5, q[1] + 0.5, f'Q {q_label}', fontsize=12,
        color='red')

    # Add the rotation arc
    radius = np.linalg.norm(p)
    angle_p_rad = np.arctan2(p[1], p[0])
    angle_p_deg = np.degrees(angle_p_rad)
    arc = Arc((0, 0), radius*0.5, radius*0.5, angle=0,
             theta1=angle_p_deg, theta2=angle_p_deg +
                 angle_degrees,
             color='green', linewidth=2, linestyle='--')
    ax.add_patch(arc)
    theta_label_rad = np.radians(angle_p_deg + angle_degrees / 2)
```

# Python Code Through Shared Output

```
    ax.text(radius*0.3 * np.cos(theta_label_rad), radius*0.3 * np
        .sin(theta_label_rad),
            f'={angle_degrees}', fontsize=12, color='green')

    # Set up the plot aesthetics
    ax.axhline(0, color='black',linewidth=0.5)
    ax.axvline(0, color='black',linewidth=0.5)
    ax.grid(True, which='both', linestyle='--', linewidth=0.5)
    ax.set_aspect('equal', adjustable='box')
    ax.set_title('2D Vector Rotation (using C function)',
        fontsize=16)
    ax.set_xlabel('X-axis', fontsize=12)
    ax.set_ylabel('Y-axis', fontsize=12)
    max_val = max(abs(p[0]), abs(p[1]), abs(q[0]), abs(q[1])) *
        1.2
    ax.set_xlim(-5, max_val)
    ax.set_ylim(-5, max_val)
    ax.legend()
    plt.show()
```

# Python Code Through Shared Output

```python
# --- Main execution block ---
if __name__ == __main__:
    # Determine the correct shared library file extension based on the operating system
    if os.name == 'nt': # For Windows
        lib_name = 'rotate_vector.dll'
    else: # For Linux, macOS, etc.
        lib_name = 'rot.so'

    # Construct the full path to the library file, assuming it's in the same directory
    lib_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), lib_name)

    try:
        # Load the compiled C code as a shared library
        c_lib = ctypes.CDLL(lib_path)
    except OSError as e:
        print(fError: Could not load the shared library '{
```

# Python Code Through Shared Output

```
        print(fDetails: {e})
        print(\nPlease compile the C code first. See README.md
            for instructions.)
        exit()

    # Get a handle to the 'rotate_point_c' function inside the
        library
    rotate_point_c = c_lib.rotate_point_c

    # Define the function's signature for ctypes
    rotate_point_c.argtypes = [ctypes.POINTER(Point), ctypes.
        c_double]
    rotate_point_c.restype = None
# --- Use the C function ---
    p = Point(x=20.0, y=10.0)
    theta = 30.0
    # Store the original coordinates before they are modified,
        for plotting
    p_original_coords = (p.x, p.y)
```

# Python Code Through Shared Output

```
print(fOriginal point P: ({p.x}, {p.y}))
print(fRotation angle: {theta})

# Call the C function. This modifies the 'p' object in place.
rotate_point_c(ctypes.byref(p), theta)

# Store the new coordinates for printing and plotting
q_rotated_coords = (p.x, p.y)

print(fNew point Q (calculated by C): ({q_rotated_coords
    [0]:.2f}, {q_rotated_coords[1]:.2f}))

# --- Visualize the result ---
plot_rotation(p_original_coords, q_rotated_coords, theta)
```

# Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Arc

def rotate_point(point, angle_degrees):

    Rotates a 2D point anti-clockwise around the origin.

    Args:
        point (tuple or list): The (x, y) coordinates of the
            point to rotate.
        angle_degrees (float): The angle of rotation in degrees.

    Returns:
        numpy.ndarray: The new (x, y) coordinates after rotation.

    # Convert the angle from degrees to radians for trigonometric
         functions
    angle_radians = np.radians(angle_degrees)
```

# Python Code

```
    # Define the initial point P as a column vector (2x1 matrix)
    p_vector = np.array([[point[0]], [point[1]]])

    # Create the 2D anti-clockwise rotation matrix
    cos_theta = np.cos(angle_radians)
    sin_theta = np.sin(angle_radians)
    rotation_matrix = np.array([
        [cos_theta, -sin_theta],
        [sin_theta, cos_theta]
    ])

    # Perform the matrix multiplication: Q = R * P
    q_vector = np.dot(rotation_matrix, p_vector)

    return q_vector.flatten() # Flatten to a 1D array for easier reading

def plot_rotation(p, q, angle_degrees):
```

# Python Code

```

    Generates a plot to visualize the rotation of point P to Q.

    fig, ax = plt.subplots(figsize=(8, 8))

    # Plot origin
    ax.plot(0, 0, 'ko', markersize=10, label='Origin (O)')

    # Create formatted strings for the labels to ensure clean
        output
    p_label = f'({p[0]:.2f}, {p[1]:.2f})'
    q_label = f'({q[0]:.2f}, {q[1]:.2f})'

    # Plot vectors and points
    # Vector OP
    ax.arrow(0, 0, p[0], p[1], head_width=0.5, head_length=0.7,
        fc='blue', ec='blue', length_includes_head=True)
    ax.plot(p[0], p[1], 'bo', markersize=8, label=f'Point P {
        p_label}')
```

# Python Code

```
    ax.text(p[0] + 0.5, p[1] + 0.5, f'P {p_label}', fontsize=12,
        color='blue')
    # Vector OQ
    ax.arrow(0, 0, q[0], q[1], head_width=0.5, head_length=0.7,
        fc='red', ec='red', length_includes_head=True)
    ax.plot(q[0], q[1], 'ro', markersize=8, label=f'Point Q {
        q_label}')
    ax.text(q[0] + 0.5, q[1] + 0.5, f'Q {q_label}', fontsize=12,
        color='red')

    # Add the rotation arc
    radius = np.linalg.norm(p)
    # Angle for arc starts from the angle of vector P
    angle_p_rad = np.arctan2(p[1], p[0])
    angle_p_deg = np.degrees(angle_p_rad)
    arc = Arc((0, 0), radius*0.5, radius*0.5, angle=0,
             theta1=angle_p_deg, theta2=angle_p_deg +
                 angle_degrees,
             color='green', linewidth=2, linestyle='--')
```

# Python Code

```python
    ax.add_patch(arc)
    # Add theta label near the arc
    theta_label_rad = np.radians(angle_p_deg + angle_degrees / 2)
    ax.text(radius*0.3 * np.cos(theta_label_rad), radius*0.3 * np.sin(theta_label_rad),
           f'={angle_degrees}', fontsize=12, color='green')

    # Set up the plot
    ax.axhline(0, color='black',linewidth=0.5)
    ax.axvline(0, color='black',linewidth=0.5)
    ax.grid(True, which='both', linestyle='--', linewidth=0.5)
    ax.set_aspect('equal', adjustable='box')
    ax.set_title('2D Vector Rotation', fontsize=16)
    ax.set_xlabel('X-axis', fontsize=12)
    ax.set_ylabel('Y-axis', fontsize=12)

    # Set axis limits to give some space around the vectors
    max_val = max(np.abs(p).max(), np.abs(q).max()) * 1.2
```

# Python Code

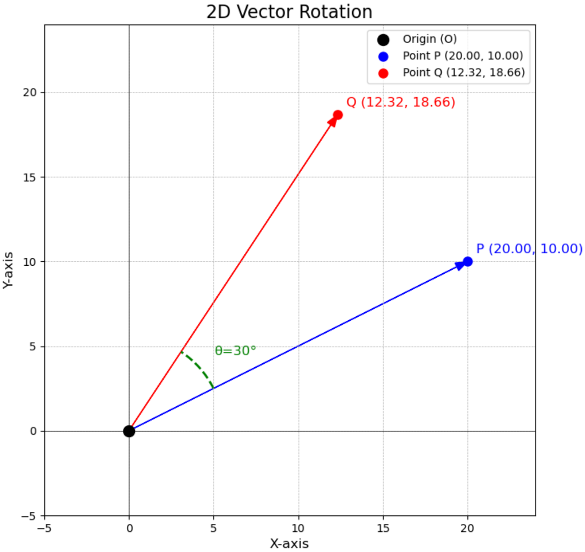
```
    ax.set_xlim(-5, max_val)
    ax.set_ylim(-5, max_val)
    ax.legend()
    plt.show()

# --- Main execution ---
if __name__ == __main__:
    # Initial point P
    P = np.array([20, 10])

    # Angle of rotation in degrees
    theta = 30

    # Calculate the new position Q
    Q = rotate_point(P, theta)
    print(fOriginal point P: {tuple(P)})
    print(fRotation angle: {theta})
    print(fNew point Q (x, y): ({Q[0]:.2f}, {Q[1]:.2f}))
    plot_rotation(P, Q, theta)
```

# Plot By C code and Python Code


2D Vector Rotation
Origin (O)
Point P (20.00, 10.00)
Point Q (12.32, 18.66)
Q (12.32, 18.66)
P (20.00, 10.00)
θ=30°
Y-axis
X-axis