2.10.78

Hema Havil - EE25BTECH11050

October 6, 2025

Question

The point (4, 1) undergoes the following three transformations successively.
(a) Reflection about the line $y = x$.
(b) Translation through a distance 2 units along the positive direction of x-axis.
(c) Rotation through an angle $\frac{\pi}{4}$ about the origin in the counter clockwise direction.
Then the final position of the point is given by the coordinates.

(a) $\left(\frac{1}{\sqrt{2}}, \frac{7}{\sqrt{2}}\right)$ (b) $\left(-\sqrt{2}, 7\sqrt{2}\right)$ (c) $\left(\sqrt{2}, 7\sqrt{2}\right)$ (d) $\left(-\frac{1}{\sqrt{2}}, \frac{7}{\sqrt{2}}\right)$

Theoretical Solution

Let the given point be P=(4,1) and vector be $\mathbf{P} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$, (a) Reflection matrix for $y = x$ is,

$$\mathbf{M} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

then the reflection of P about $y = x$ is,

$$\mathbf{P_1} = \mathbf{MP} \tag{2}$$

$$\mathbf{P_1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} \tag{3}$$

$$\mathbf{P_1} = \begin{pmatrix} 1 \\ 4 \end{pmatrix} \tag{4}$$

Theoretical Solution

(b) Convert P' into homogeneous form,

$$\mathbf{P_1^h} = \begin{pmatrix} 1 \\ 4 \\ 1 \end{pmatrix} \tag{5}$$

The translation matrix along the x direction is given as,

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

then the translated vector is,

$$\mathbf{P_2^h} = \mathbf{T}\mathbf{P_1^h} \tag{7}$$

Theoretical Solution

$$\mathbf{P_2^h} = \begin{pmatrix} 3 \\ 4 \\ 1 \end{pmatrix} \tag{8}$$

$$\mathbf{P_2} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{9}$$

(c) Rotation matrix is given as,

$$\mathbf{R} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{10}$$

Theoretical Solution

Rotation of P_2 by an angle $\frac{\pi}{4}$ about the origin in the counter clockwise direction,

$$\mathbf{P_3} = \mathbf{R}\mathbf{P_2} \tag{11}$$

$$\mathbf{P_3} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{12}$$

$$\mathbf{P_3} = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \end{pmatrix} \tag{13}$$

Therefore the final position of the point is $P_3 = (-\frac{1}{\sqrt{2}}, \frac{7}{\sqrt{2}})$, option (d) is correct

C Code- Computing the unit vector

```c
/// File: transform.c
#include <stdio.h>

void matvec2x2(double mat[2][2], double vec[2], double result[2])
    {
    for (int i = 0; i < 2; i++) {
        result[i] = 0;
        for (int j = 0; j < 2; j++) {
            result[i] += mat[i][j] * vec[j];
        }
    }
}
```

C Code - Computing the unit vector

```c
void matvec3x3(double mat[3][3], double vec[3], double result
    [3]) {
for (int i = 0; i < 3; i++) {
    result[i] = 0;
    for (int j = 0; j < 3; j++) {
        result[i] += mat[i][j] * vec[j];
    }
}
}
```

Python Code using shared output

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

# Load C shared library
lib = ctypes.CDLL('./libtransform.so')

# Set argument and return types
lib.matvec2x2.argtypes = [np.ctypeslib.ndpointer(dtype=np.float64
    , shape=(2,2)),
                          np.ctypeslib.ndpointer(dtype=np.float64,
                              shape=(2,)),
                          np.ctypeslib.ndpointer(dtype=np.float64,
                              shape=(2,))]
```

Python Code using shared output

```
    lib.matvec3x3.argtypes = [np.ctypeslib.ndpointer(dtype=np.float64, shape=(3,3)),
                              np.ctypeslib.ndpointer(dtype=np.float64, shape=(3,)),
                              np.ctypeslib.ndpointer(dtype=np.float64, shape=(3,))]
     # Initial point
P = np.array([4.0, 1.0])

# Step 1: Reflection over y = x  swap x and y
reflect_mat = np.array([[0.0, 1.0],
                        [1.0, 0.0]])
P1 = np.zeros(2)
lib.matvec2x2(reflect_mat, P, P1)
```

Python Code using shared output

```
    # Step 2: Translation +2 in x-direction using 3x3 matrix
P1_h = np.array([P1[0], P1[1], 1.0])
translate_mat = np.array([[1.0, 0.0, 2.0],
                         [0.0, 1.0, 0.0],
                         [0.0, 0.0, 1.0]])
P2_h = np.zeros(3)
lib.matvec3x3(translate_mat, P1_h, P2_h)
P2 = P2_h[:2]

# Step 3: Rotation by pi/4 (45 counterclockwise)
theta = np.pi / 4
cos_t = np.cos(theta)
sin_t = np.sin(theta)
rotation_mat = np.array([[cos_t, -sin_t],
                         [sin_t, cos_t]])
P3 = np.zeros(2)
lib.matvec2x2(rotation_mat, P2, P3)
```

Python Code using shared output

```
        # Plotting the transformation steps
points = np.array([P, P1, P2, P3])
labels = ['Original (4,1)', 'After Reflection', 'After
    Translation', 'After Rotation (Final)']
colors = ['blue', 'orange', 'green', 'red']
plt.figure(figsize=(8, 8))

for i, point in enumerate(points):
    plt.plot(point[0], point[1], 'o', label=labels[i], color=
        colors[i])
    plt.text(point[0]+0.1, point[1]+0.1, f'{labels[i]}')
```

Frame Title

```
        plt.plot(points[:,0], points[:,1], '--k', alpha=0.5)
plt.grid(True)
plt.axhline(0, color='black', lw=1)
plt.axvline(0, color='black', lw=1)
plt.legend()
plt.title(Transformation of Point (4,1))
plt.xlabel(X)
plt.ylabel(Y)
plt.axis('equal')
plt.show()

# Final result
print(fFinal coordinates after all transformations: ({P3[0]}, {P3
    [1]}))
```

Plot by python using shared output from c

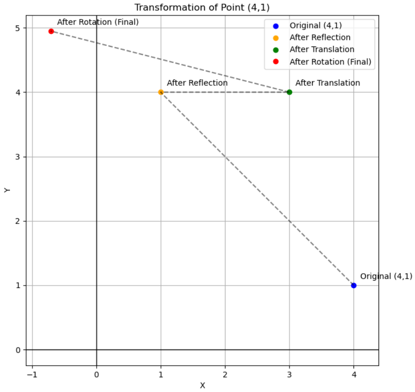

Figure: Plot for the transformations of P

Python code for the plot

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

# Step 0: Initial point
P = np.array([4.0, 1.0])
points = [P]
labels = [Original (4,1)]

# Step 1: Reflection about the line y = x
# Matrix: [[0, 1], [1, 0]]
reflect_matrix = np.array([[0, 1],
                           [1, 0]])
P1 = reflect_matrix @ P
points.append(P1)
labels.append(After Reflection)
```

Python code for plot

```
    # Step 2: Translation by 2 units along +x-axis
# Use homogeneous coordinates: convert P1 to 3D vector
P1_h = np.array([P1[0], P1[1], 1.0])
translate_matrix = np.array([[1, 0, 2],
                              [0, 1, 0],
                              [0, 0, 1]])
P2_h = translate_matrix @ P1_h
P2 = P2_h[:2]
points.append(P2)
labels.append(After Translation)

# Step 3: Rotation by /4 counter-clockwise
theta = np.pi / 4
cos_t = np.cos(theta)
sin_t = np.sin(theta)
rotate_matrix = np.array([[cos_t, -sin_t],
                          [sin_t, cos_t]])
```

Python code for plot

```
        P3 = rotate_matrix @ P2
points.append(P3)
labels.append(After Rotation )

# Convert all points to a NumPy array
points = np.array(points)

# Plotting
colors = ['blue', 'orange', 'green', 'red']
plt.figure(figsize=(8, 8))
for i, point in enumerate(points):
    plt.plot(point[0], point[1], 'o', color=colors[i], label=
        labels[i])
    plt.text(point[0] + 0.2, point[1] + 0.2, f{labels[i]}\n({
        point[0]:.2f}, {point[1]:.2f}))
# Draw arrows between steps
   for i in range(len(points) - 1):
```

Python code for the plot

```
    plt.arrow(points[i][0], points[i][1],
              points[i+1][0] - points[i][0],
              points[i+1][1] - points[i][1],
              head_width=0.2, length_includes_head=True,
              fc='gray', ec='gray', linestyle='dashed')

plt.axhline(0, color='black', linewidth=1)
plt.axvline(0, color='black', linewidth=1)
plt.grid(True)
plt.legend()
plt.axis('equal')
plt.title(Transformations of Point (4, 1))
plt.xlabel(X-axis)
plt.ylabel(Y-axis)
plt.show()

# Final output
print(fFinal coordinates: ({P3[0]:.4f}, {P3[1]:.4f}))
```

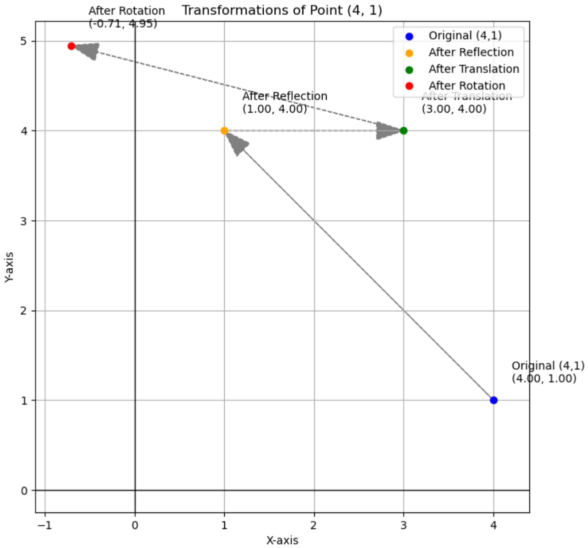

Figure: Plot for the transformations of P