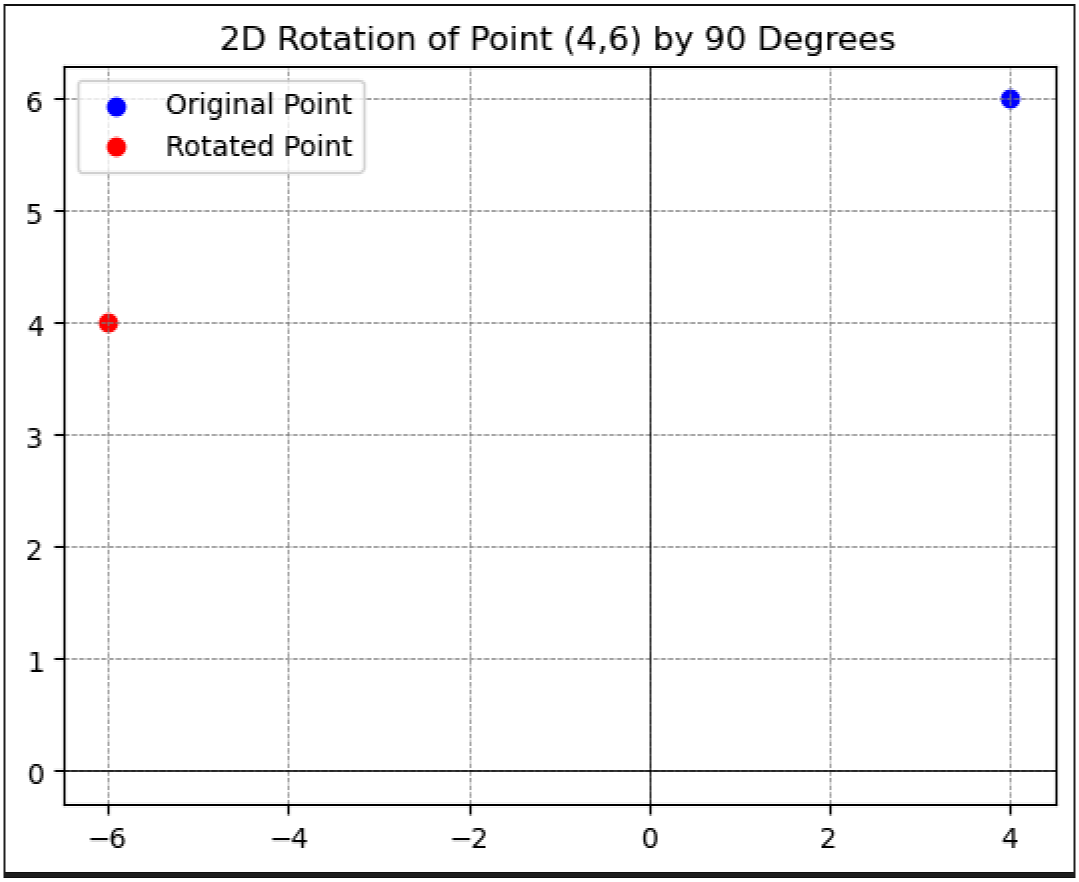
CSE 358 - ASSIGNMENT 6

Arnav Jain - 220002018

Question 1

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

# Input point
point = np.array([4, 6])

# Rotation matrix
angle = np.radians(90)
rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)],
                             [np.sin(angle), np.cos(angle)]])

# Output Point
rotated_point = rotation_matrix @ point

# Plotting
plt.figure()
plt.scatter(*point, color='blue', label='Original Point')
plt.scatter(*rotated_point, color='red', label='Rotated Point')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend()
plt.title('2D Rotation of Point (4,6) by 90 Degrees')
plt.show()
```

✓ 0.6s




```python
print("Original Point:")
print(point)
print("Rotated Point:")
print(rotated_point)
print("Roatatation Matrix:")
print(rotation_matrix)
```

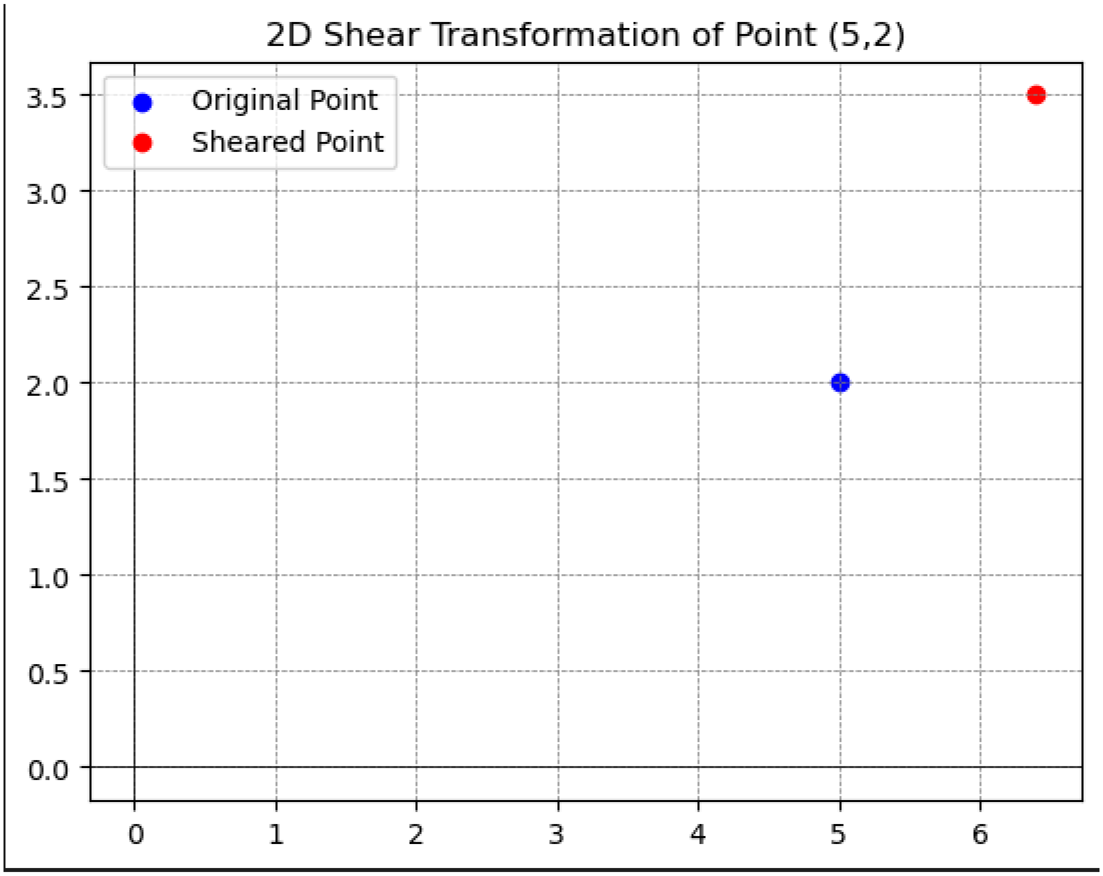
[3] ✓ 0.0s

```
Original Point:
[4 6]
Rotated Point:
[-6.  4.]
Roatatation Matrix:
[[ 6.123234e-17 -1.000000e+00]
 [ 1.000000e+00  6.123234e-17]]
```

Question 2

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

# Original point
point = np.array([5, 2])

# Shear factors
shear_x = 0.7
shear_y = 0.3

# Shear matrix
shear_matrix = np.array([[1, shear_x],
                          [shear_y, 1]])

# Sheared point
sheared_point = shear_matrix @ point

# Visualization
plt.figure()
plt.scatter(*point, color='blue', label='Original Point')
plt.scatter(*sheared_point, color='red', label='Sheared Point')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend(loc='upper left')
plt.title('2D Shear Transformation of Point (5,2)')
plt.show()
```

✓ 0.1s



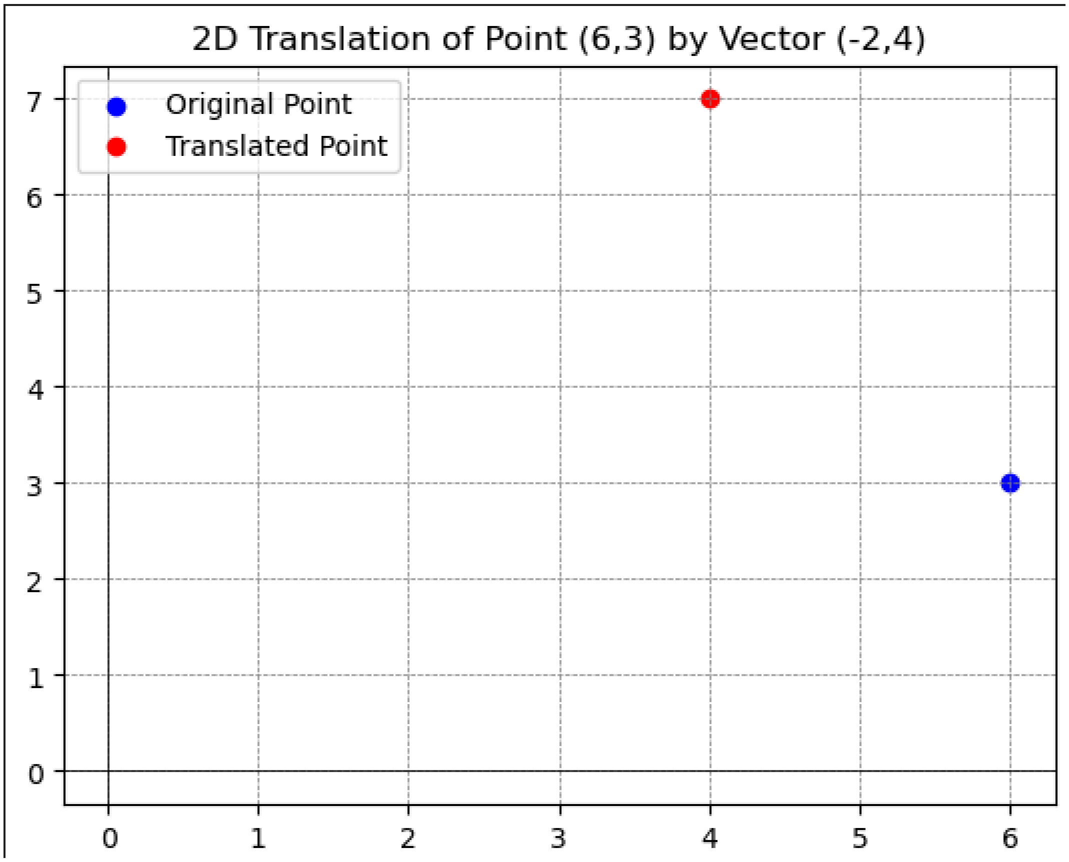

```python
print("Original Point: ", point)
print(f'Sheared Point: {sheared_point}')
print("Shear Matrix: ")
print(shear_matrix)
```

[3] ✓ 0.0s

```
Original Point:  [5 2]
Sheared Point: [6.4 3.5]
Shear Matrix: 
[[1.  0.7]
 [0.3 1. ]]
```

Question 3

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

# Original point
point = np.array([6, 3])

# Translation vector
translation_vector = np.array([-2, 4])

# Translated point
translated_point = point + translation_vector

# Visualization
plt.figure()
plt.scatter(*point, color='blue', label='Original Point')
plt.scatter(*translated_point, color='red', label='Translated Point')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend(loc='upper left')
plt.title('2D Translation of Point (6,3) by Vector (-2,4)')
plt.show()
```
✓ 0.1s




```python
print(f"Original Point: {point}")
print(f'Translated Point: {translated_point}')
print(f'Translation Vector: {translation_vector}')
```

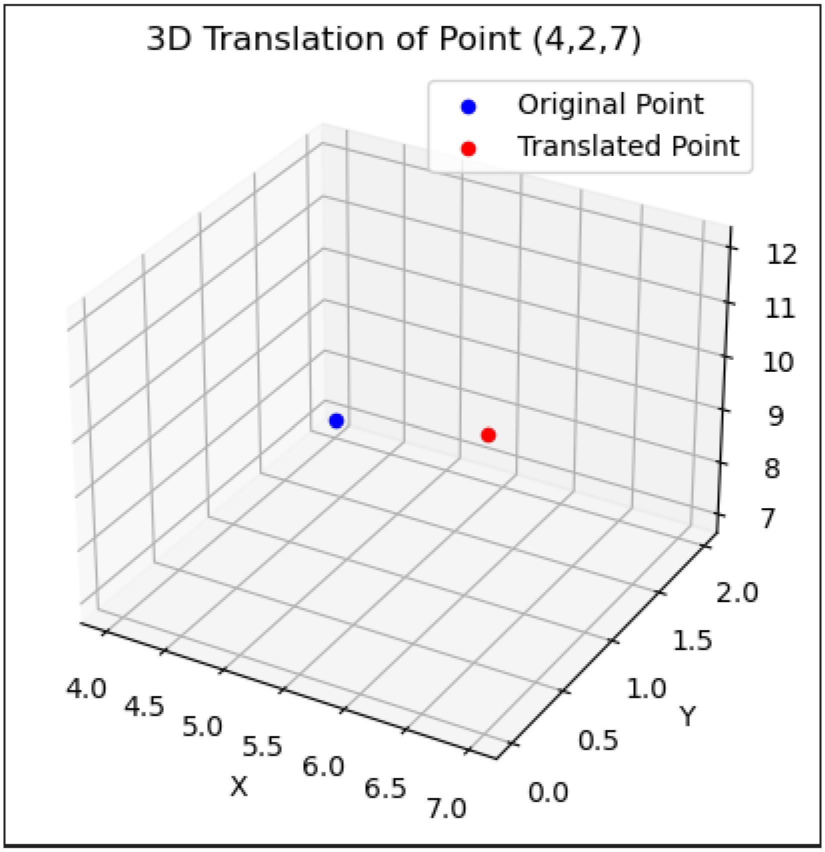
✓ 0.0s

```
Original Point: [6 3]
Translated Point: [4 7]
Translation Vector: [-2  4]
```

Question 4

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

# Original point in homogeneous coordinates
point = np.array([4, 2, 7, 1])

# Translation values
translation_values = np.array([3, -2, 5])

# Translation matrix (4x4)
translation_matrix = np.eye(4)
translation_matrix[:3, 3] = translation_values

# Translated point
translated_point = translation_matrix @ point

# Visualization
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(*point[:3], color='blue', label='Original Point')
ax.scatter(*translated_point[:3], color='red', label='Translated Point')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.title('3D Translation of Point (4,2,7)')
plt.show()
```

✓ 0.1s



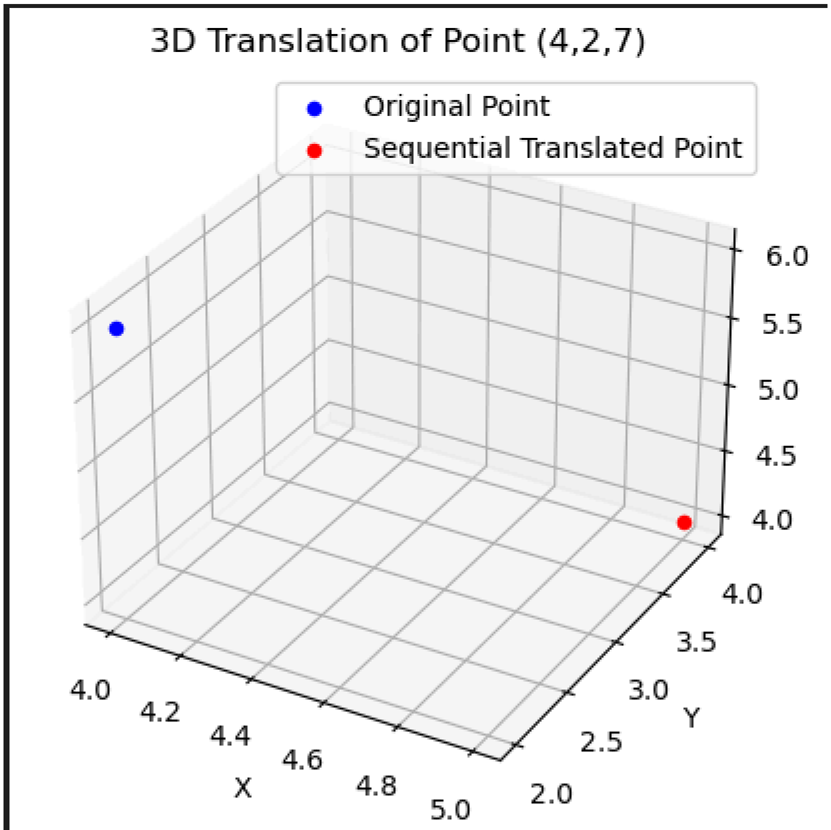

```python
print(f'Original Point: {point}')
print(f'Translation Matrix:\n{translation_matrix}')
print(f'Translated Point: {translated_point}')
```

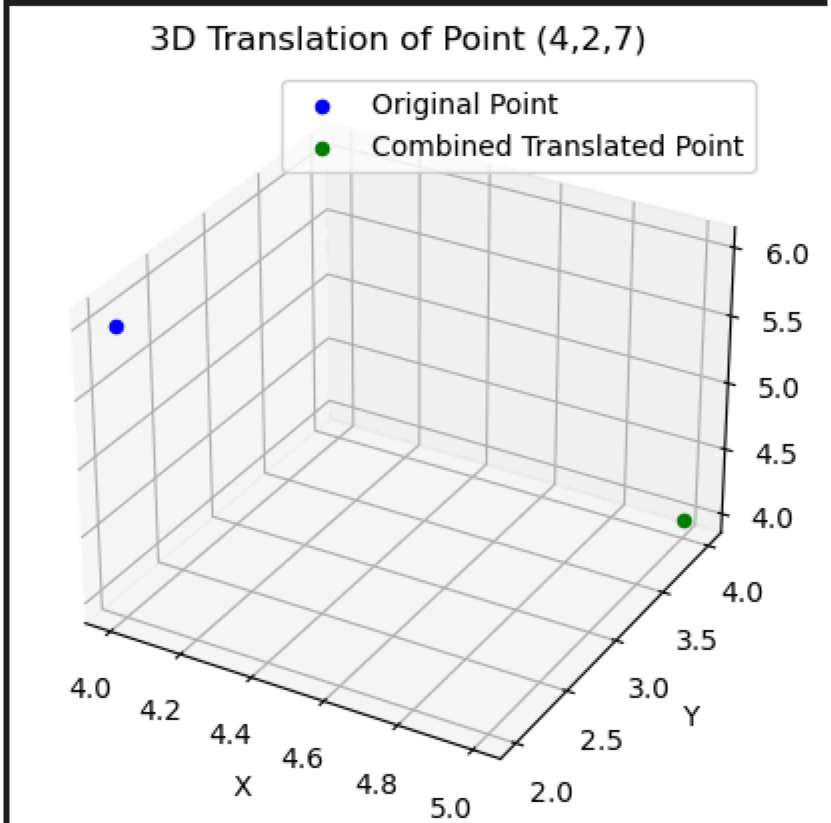
[2] ✓ 0.0s

```
Original Point: [4 2 7 1]
Translation Matrix:
[[ 1.  0.  0.  3.]
 [ 0.  1.  0. -2.]
 [ 0.  0.  1.  5.]
 [ 0.  0.  0.  1.]]
Translated Point: [ 7.  0. 12.  1.]
```

Question 5

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

# Original point in homogeneous coordinates
point = np.array([4, 2, 6, 1])

# Translation vectors and matrices
translation_vector1 = np.array([3, -2, 1])
translation_vector2 = np.array([-2, 4, -3])
translation_matrix1 = np.eye(4)
translation_matrix1[:3, 3] = translation_vector1
translation_matrix2 = np.eye(4)
translation_matrix2[:3, 3] = translation_vector2

# Sequential translations
point_after_first_translation = translation_matrix1 @ point
point_after_second_translation = translation_matrix2 @ point_after_first_translation

# Combined translation
combined_translation_vector = translation_vector1 + translation_vector2
combined_translation_matrix = np.eye(4)
combined_translation_matrix[:3, 3] = combined_translation_vector

point_after_combined_translation = combined_translation_matrix @ point

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(*point[:3], color='blue', label='Original Point')
ax.scatter(*point_after_second_translation[:3], color='red', label='Sequential Translated Point')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.title('3D Translation of Point (4,2,7)')
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(*point[:3], color='blue', label='Original Point')
ax.scatter(*point_after_combined_translation[:3], color='green', label='Combined Translated Point')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.title('3D Translation of Point (4,2,7)')
plt.show()
```


3D Translation of Point (4,2,7)
Original Point
Sequential Translated Point
X
Y
4.0
4.2
4.4
4.6
4.8
5.0
2.0
2.5
3.0
3.5
4.0
4.0
4.5
5.0
5.5
6.0


3D Translation of Point (4,2,7)
Original Point
Combined Translated Point
X
Y
4.0
4.2
4.4
4.6
4.8
5.0
2.0
2.5
3.0
3.5
4.0
4.0
4.5
5.0
5.5
6.0

```python
print("Original Point: ", point)
print("Point after sequential translations: ", point_after_second_translation)
print("Point after combined translation: ", point_after_combined_translation)
```

[2] ✓ 0.0s

```
Original Point:  [4 2 6 1]
Point after sequential translations:  [5. 4. 4. 1.]
Point after combined translation:  [5. 4. 4. 1.]
```

```python
print("Translation vector 1: ", translation_matrix1)
print("Translation vector 2: ", translation_matrix2)
print("Combined translation matrix: ", combined_translation_matrix)
```

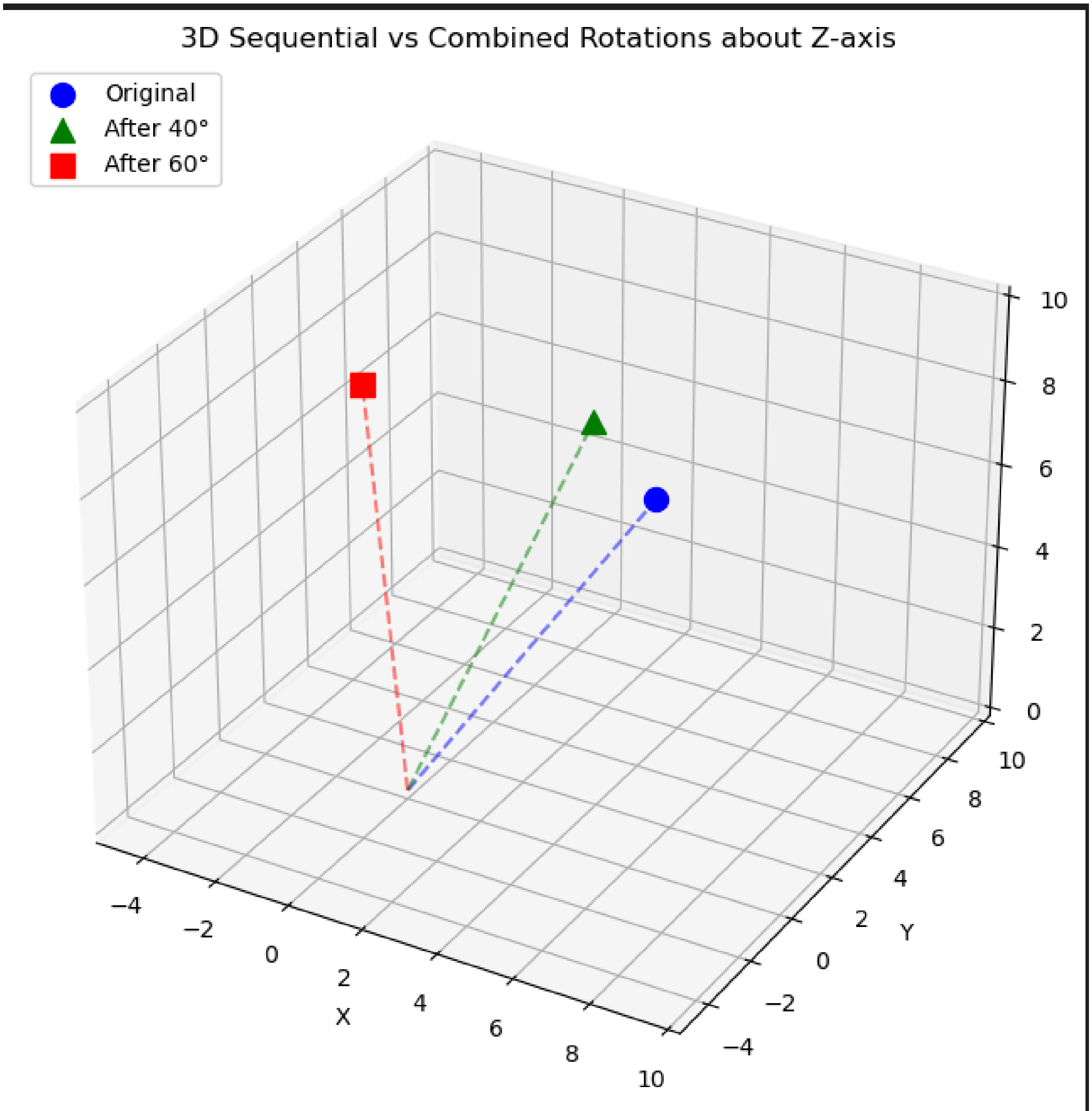
[4] ✓ 0.0s

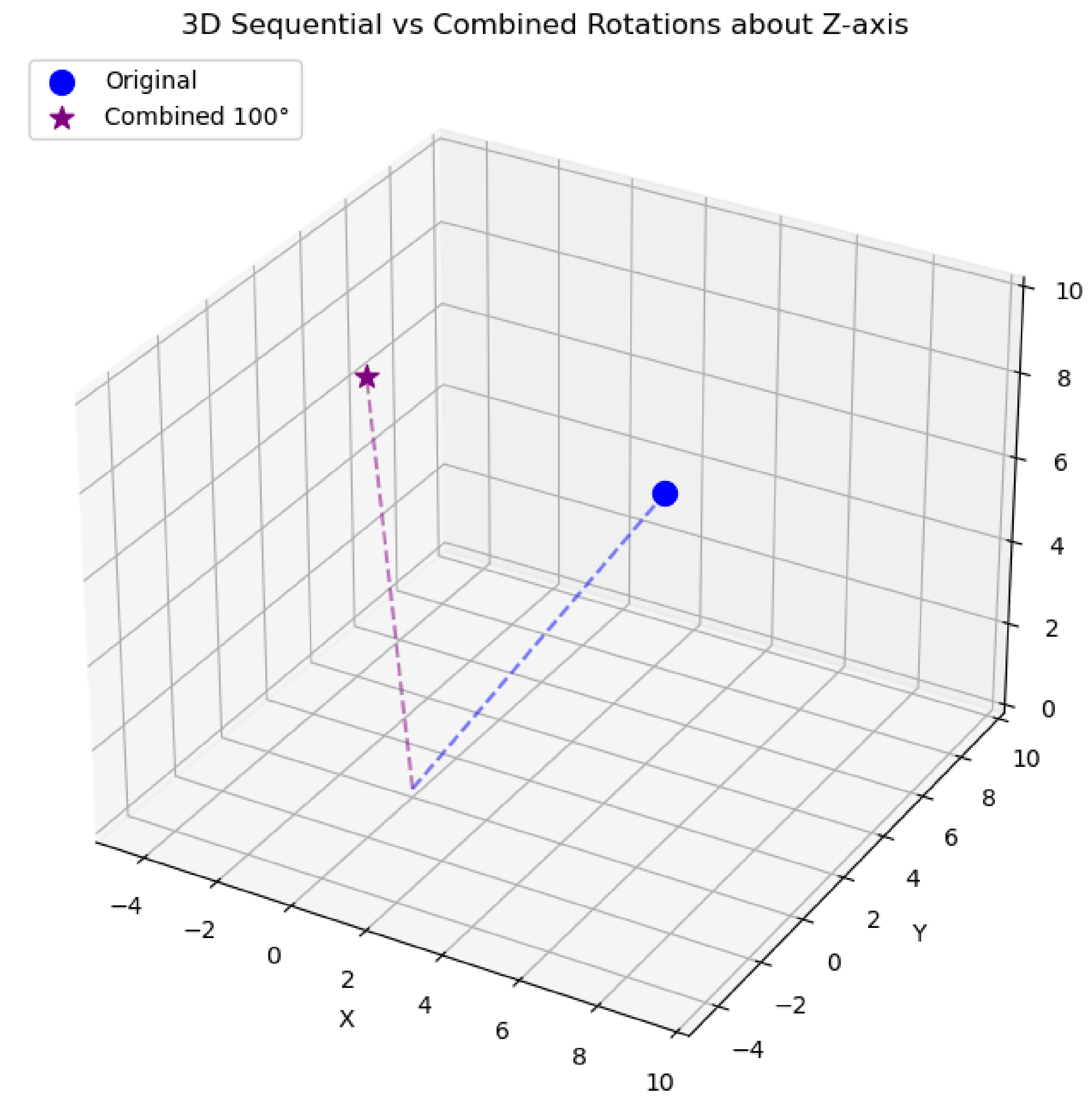
```
Translation vector 1:  [[ 1.  0.  0.  3.]
 [ 0.  1.  0. -2.]
 [ 0.  0.  1.  1.]
 [ 0.  0.  0.  1.]]
Translation vector 2:  [[ 1.  0.  0. -2.]
 [ 0.  1.  0.  4.]
 [ 0.  0.  1. -3.]
 [ 0.  0.  0.  1.]]
Combined translation matrix:  [[ 1.  0.  0.  1.]
 [ 0.  1.  0.  2.]
 [ 0.  0.  1. -2.]
 [ 0.  0.  0.  1.]]
```

Question 6

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

# Original point in homogeneous coordinates
point = np.array([5, 3, 7, 1])

# Rotation angles (degrees)
angle1 = 40
angle2 = 60

# Create rotation matrices (Z-axis)
def z_rotation_matrix(degrees):
    theta = np.radians(degrees)
    return np.array([
        [np.cos(theta), -np.sin(theta), 0, 0],
        [np.sin(theta), np.cos(theta), 0, 0],
        [0, 0, 1, 0],
        [0, 0, 0, 1]
    ])

# Sequential rotations
R1 = z_rotation_matrix(angle1)
R2 = z_rotation_matrix(angle2)
sequential_result = R2 @ R1 @ point

# Combined rotation
combined_angle = angle1 + angle2
R_combined = z_rotation_matrix(combined_angle)
combined_result = R_combined @ point


# Visualization
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot points
points = {
    'Original': point[:3],
    'After 40°': (R1 @ point)[:3],
    'After 60°': sequential_result[:3],
}

colors = {'Original': 'blue', 'After 40°': 'green', 'After 60°': 'red', }
markers = {'Original': 'o', 'After 40°': '^', 'After 60°': 's'}
```

```python
for label, pt in points.items():
    ax.scatter(*pt, color=colors[label], marker=markers[label], s=100, label=label)
    ax.plot([0, pt[0]], [0, pt[1]], [0, pt[2]], color=colors[label], linestyle='dashed', alpha=0.5)

ax.set_xlim([-5, 10])
ax.set_ylim([-5, 10])
ax.set_zlim([0, 10])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend(loc='upper left')
plt.title('3D Sequential vs Combined Rotations about Z-axis')
plt.show()


# Visualization
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot points
points = {
    'Original': point[:3],
    'Combined 100°': combined_result[:3]
}

colors = {'Original': 'blue', 'Combined 100°': 'purple'}
markers = {'Original': 'o', 'Combined 100°': '*'}

for label, pt in points.items():
    ax.scatter(*pt, color=colors[label], marker=markers[label], s=100, label=label)
    ax.plot([0, pt[0]], [0, pt[1]], [0, pt[2]], color=colors[label], linestyle='dashed', alpha=0.5)

ax.set_xlim([-5, 10])
ax.set_ylim([-5, 10])
ax.set_zlim([0, 10])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend(loc='upper left')
plt.title('3D Sequential vs Combined Rotations about Z-axis')
plt.show()
```


3D Sequential vs Combined Rotations about Z-axis
Original
After 40°
After 60°
X
Y
−4
−2
0
2
4
6
8
10


3D Sequential vs Combined Rotations about Z-axis
Original
Combined 100°
X
Y
−4
−2
0
2
4
6
8
10

```python
print("Original point:", point)
print("After 40° rotation:", (R1 @ point)[:3])
print("After 60° rotation:", sequential_result[:3])
print("Combined 100° rotation:", combined_result[:3])
print("\n40° Rotation matrix:\n", R1)
print("\n60° Rotation matrix:\n", R2)
print("\n100° Combined rotation matrix:\n", R_combined)
```

✓ 0.0s

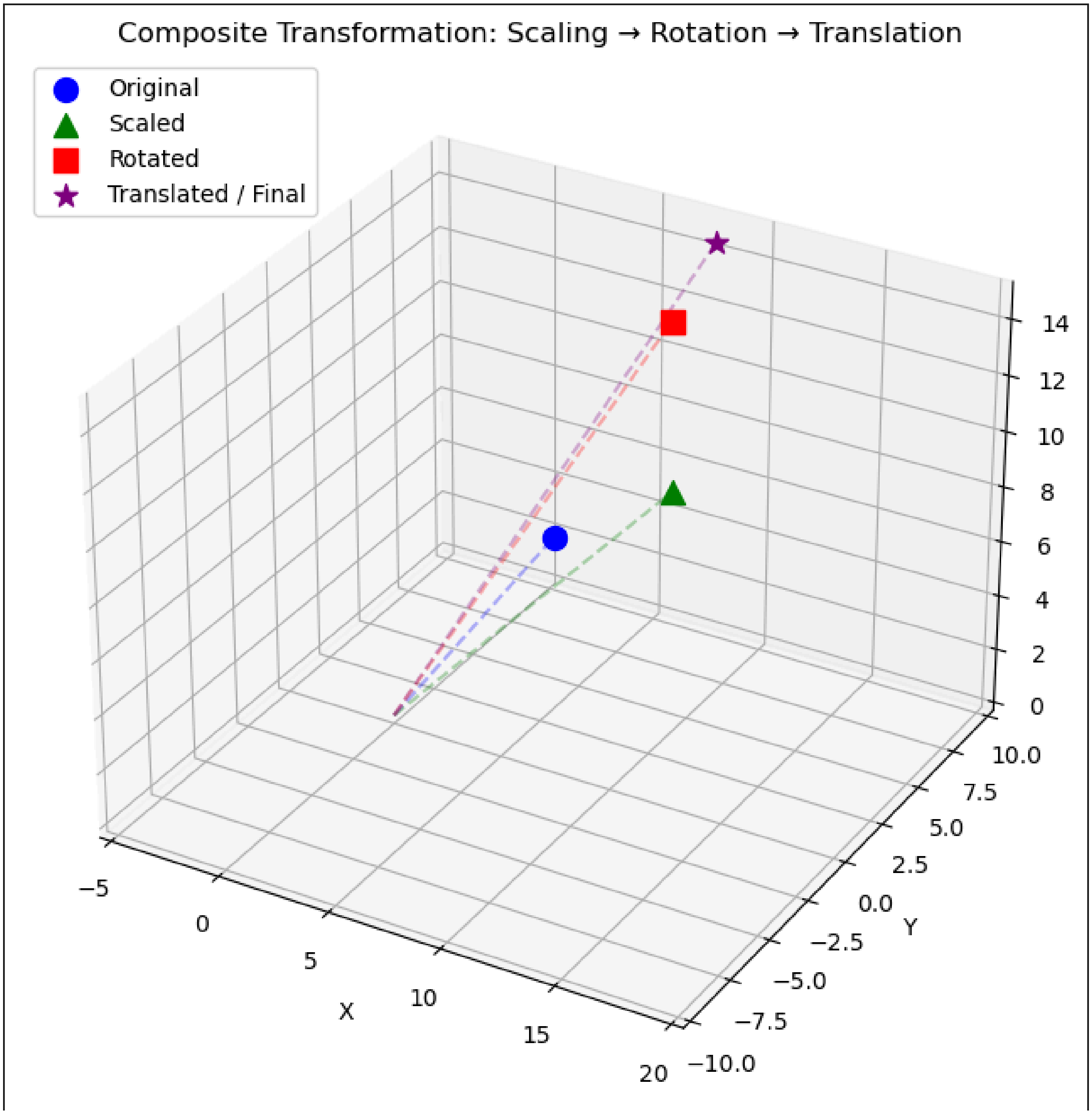
```
Original point: [5 3 7 1]
After 40° rotation: [1.90185939 5.51207138 7.        ]
After 60° rotation: [-3.82266415  4.40309423  7.        ]
Combined 100° rotation: [-3.82266415  4.40309423  7.        ]

40° Rotation matrix:
 [[ 0.76604444 -0.64278761  0.          0.        ]
 [ 0.64278761  0.76604444  0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.          0.          1.        ]]

60° Rotation matrix:
 [[ 0.5       -0.8660254  0.         0.       ]
 [ 0.8660254  0.5        0.         0.       ]
 [ 0.         0.         1.         0.       ]
 [ 0.         0.         0.         1.       ]]

100° Combined rotation matrix:
 [[-0.17364818 -0.98480775  0.          0.        ]
 [ 0.98480775 -0.17364818  0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.          0.          1.        ]]
```

Question 7

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

# Original point
point = np.array([6, 2, 7, 1])

# Scaling matrix
S = np.diag([2.0, 0.5, 1.5, 1])

# Rotation matrix (60° about Z-axis)
theta = np.radians(60)
R = np.array([
    [np.cos(theta), -np.sin(theta), 0, 0],
    [np.sin(theta), np.cos(theta), 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])

# Translation matrix
T = np.eye(4)
T[:3, 3] = [4, -3, 5]

# 2. Composite transformation matrix
composite_matrix = T @ R @ S

# 3. Apply transformations
transformed_point = composite_matrix @ point

# Intermediate points
scaled_point = S @ point
rotated_point = R @ scaled_point
translated_point = T @ rotated_point
```

```python
# Visualization
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot points and dashed lines
points = {
    'Original': point[:3],
    'Scaled': scaled_point[:3],
    'Rotated': rotated_point[:3],
    'Translated / Final': transformed_point[:3]
}

colors = {'Original': 'blue', 'Scaled': 'green', 'Rotated': 'red', 'Translated / Final': 'purple'}
markers = {'Original': 'o', 'Scaled': '^', 'Rotated': 's', 'Translated / Final': '*'}

for label, pt in points.items():
    ax.scatter(*pt, color=colors[label], marker=markers[label], s=100, label=label)
    ax.plot([0, pt[0]], [0, pt[1]], [0, pt[2]],
            color=colors[label], linestyle='dashed', alpha=0.3)

ax.set_xlim([-5, 20])
ax.set_ylim([-10, 10])
ax.set_zlim([0, 15])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend(loc='upper left')
plt.title('Composite Transformation: Scaling → Rotation → Translation')
plt.show()

# Print final result
print("Final Transformed Point (x,y,z):", transformed_point[:3].round(2))
```

✓ 0.5s


Composite Transformation: Scaling → Rotation → Translation
Original
Scaled
Rotated
Translated / Final
X
Y
−5
0
5
10
15
20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
0
2
4
6
8
10
12
14

```python
print("Original Point:", point[:3])
print("Scaled Point:", scaled_point[:3])
print("Rotated Point:", rotated_point[:3])
print("Translated Point:", translated_point[:3])
print("Final Transformed Point:", transformed_point[:3])
```

✓ 0.0s

```
Original Point: [6 2 7]
Scaled Point: [12.   1.  10.5]
Rotated Point: [ 5.1339746  10.89230485 10.5        ]
Translated Point: [ 9.1339746   7.89230485 15.5        ]
Final Transformed Point: [ 9.1339746   7.89230485 15.5        ]
```

```python
print("Composite Transformation Matrix:")
print(composite_matrix)
print()
print("Scaled Matrix:")
print(S)
print()
print("Rotation Matrix:")
print(R)
print()
print("Translation Matrix:")
print(T)
print()
```

✓ 0.0s

```
Composite Transformation Matrix:
[[ 1.         -0.4330127   0.          4.        ]
 [ 1.73205081  0.25        0.         -3.        ]
 [ 0.          0.          1.5         5.        ]
 [ 0.          0.          0.          1.        ]]

Scaled Matrix:
[[2.  0.  0.  0. ]
 [0.  0.5 0.  0. ]
 [0.  0.  1.5 0. ]
 [0.  0.  0.  1. ]]

Rotation Matrix:
[[ 0.5       -0.8660254  0.         0.        ]
 [ 0.8660254  0.5        0.         0.        ]
 [ 0.         0.         1.         0.        ]
 [ 0.         0.         0.         1.        ]]

Translation Matrix:
[[ 1.  0.  0.  4.]
 [ 0.  1.  0. -3.]
 [ 0.  0.  1.  5.]
 [ 0.  0.  0.  1.]]
```

Code (GitHub)

https://github.com/arnavjain2710/Computer-Graphics-Lab/tree/main/LAB%206