Linear Algebra

Question 1

Problem

Rotation, Scaling and reflection:

Solution

In order to determine the transformation matrix for a given operation, we take 2 vectors and form a 2×2 matrix then find their inverse and multiply it by the transformed vector matrix to find the transformation matrix.

$$\begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = B \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = B \tag{2}$$

Therfore we use any 2 independent vectors and their transformed versions to find the transformation matrix and get:

First transformation:

$$\boxed{B_1 = \begin{bmatrix} -0.5 & 0 \\ 0 & 0.5 \end{bmatrix}}$$

Second transformation:

$$\boxed{B_2 = \begin{bmatrix} 0 & -0.5 \\ -0.5 & 0 \end{bmatrix}}$$

Question 2

Problem

Translation, scaling and reflection:

Solution

We form a general equation for the transformation and the translation and convert it to a system of linear equations and solve for each unknown variable.

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = B \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

After simplifying we get:

$$a' = B_{11}a + B_{12}b + x \tag{4}$$

$$b' = B_{21}a + B_{22}b + y \tag{5}$$

We can then form a 3×3 system of linear equations to solve for the unknowns in the transformation matrix

Transformation matrix:

$$\boxed{B = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & -2 \end{bmatrix}}$$

Translation vector:

$$\boxed{V = \begin{bmatrix} 5 \\ 0 \end{bmatrix}}$$

Code used

0.1 First Question

```
import numpy as np
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt


X = np.array([[16,  10],
              [9,   9]])

Y1 = np.array([[-8,  -5],
               [4.5,  4.5]])

Y2 = np.array([[-4.5,  -4.5],
               [-8,    -5]])

B1 = Y1 @ np.linalg.inv(X)
B2 = Y2 @ np.linalg.inv(X)

print("Transformation Matrix B1:\n", B1)
print("Transformation Matrix B2:\n", B2)
original_shape = np.array([

    [6,  5],
    [10,  9],
    [16,  9],
    [20,  5],
    [18,  2],
    [8,  2]
]).T


shape_B1 = B1 @ original_shape
shape_B2 = B2 @ original_shape

def close_shape(S):
    return np.hstack([S, S[:, [0]]])

orig = close_shape(original_shape)
t1   = close_shape(shape_B1)
t2   = close_shape(shape_B2)
```

```
plt.figure(figsize=(10, 6))

plt.plot(orig[0], orig[1], 'g-', linewidth=2, label="Original Image")
plt.scatter(orig[0], orig[1], color='g')

plt.plot(t1[0], t1[1], 'r-', linewidth=2, label="Transformation 1 (B1)")
plt.scatter(t1[0], t1[1], color='r')

plt.plot(t2[0], t2[1], 'b-', linewidth=2, label="Transformation 2 (B2)")
plt.scatter(t2[0], t2[1], color='b')

plt.axhline(0, color='gray', linewidth=0.8)
plt.axvline(0, color='gray', linewidth=0.8)
plt.grid(True)
plt.gca().set_aspect('equal', adjustable='box')

plt.legend()
plt.title("Original Image and Linear Transformations")
plt.xlabel("X")
plt.ylabel("Y")

plt.show()
```

0.2 Second Question

```
import numpy as np
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt

original_shape = np.array([
    [0, 1, 1, 5, 5, 6,4.5,4.5,4,4, 3],
    [3, 3, 1, 1, 3, 3, 4, 4.5,4.5,4.3, 5]
])

trans_matrix = np.array([
    [0.25,  0],
    [0,    -2]
])
sliding_vector = np.array([[5], [0]])

transformed_shape = (trans_matrix @ original_shape)+ sliding_vector

def complete_shape(S):
    return np.hstack([S, S[:, [0]]])
```

```
orig  = complete_shape(original_shape)
trans = complete_shape(transformed_shape)
################################################
plt.figure(figsize=(20, 6))

################################################
plt.plot(orig[0], orig[1], 'k-', linewidth=2, label="Original Shape")
plt.scatter(orig[0], orig[1], color='k')

# Transformed shape
plt.plot(trans[0], trans[1], 'r-', linewidth=2, label="Transformed Shape")
plt.scatter(trans[0], trans[1], color='r')

plt.axhline(0, color='gray', linewidth=0.7)
plt.axvline(0, color='gray', linewidth=0.7)
plt.grid(True)
plt.gca().set_xlim(0,8)#.set_aspect('equal', adjustable='box')
plt.gca().set_xlim(0,10)


plt.xlabel("X")
plt.ylabel("Y")
plt.title("Original Shape and Linear Transformation")
plt.legend()

plt.show()
```