DIGITAL SIGNAL PROCESSING LEC 1730-2030 Mon

Prelim Term Problem Set

Coded and submitted by

Columba, Lorenzo Miguel L.

58035

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(200)



def plot_quiv(x, t_mat=np.eye(2)):
    x_prime = x @ t_mat
    size = (2, 2)
    plt.figure(figsize=(4, 4))

    plt.xlim(-size[0], size[0])
    plt.ylim(-size[1], size[1])
    plt.xticks(np.arange((-size[0]), size[0] + 1, 1.0))
    plt.yticks(np.arange((-size[1]), size[1] + 1, 1.0))

    plt.quiver([0, 0], [0, 0], x_prime[0, :], x_prime[1, :],
               angles='xy', scale_units='xy', scale=1,
               color=['red', 'blue'])
    plt.grid()
    plt.show()


def plot_quiv_imag(x, t_mat=np.eye(2)):
    x_prime = x @ t_mat
    size = (2, 2)
    plt.figure(figsize=(4, 4))

    plt.xlim(-size[0], size[0])
    plt.ylim(-size[1], size[1])
    plt.xticks(np.arange((-size[0]), size[0] + 1, 1.0))
    plt.yticks(np.arange((-size[1]), size[1] + 1, 1.0))

    plt.quiver([0, 0], [0, 0], x_prime[0, :].imag, x_prime[1, :].imag,
               angles='xy', scale_units='xy', scale=1, color=['red', 'blue'])


    plt.grid()
    plt.show()


def plot_3d_quiv(x, azimuth=0, elevation=0):
    fig = plt.figure(figsize=(8,8))
    ax1 = fig.add_subplot(projection='3d')
    ax1.set_xlim([-2, 2])
    ax1.set_ylim([-2, 2])
    ax1.set_zlim([-2, 2])
    ax1.set_xlabel("X (roll)")
    ax1.set_ylabel("Y (pitch)")
    ax1.set_zlabel("Z (yaw)")

    origin = (0,0,0)
    ax1.quiver(origin, origin, origin, x[0,:], x[1,:], x[2,:],
               arrow_length_ratio=0.1, colors=['red','blue','green'])
    plt.grid()
    ax1.view_init(azim=azimuth, elev=elevation)
    ax1.set_box_aspect(aspect=None, zoom=0.7)
    plt.show()
```

PART 1

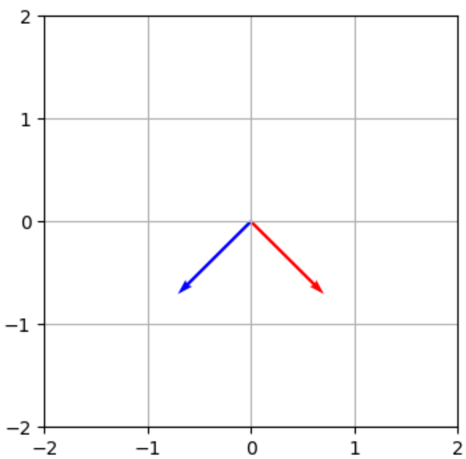
[150 pts] Solve for the magnitude and angle of the linear transformations given. Provide necessary solutions numerically (hand-solved) and computationally (using a Python program). Also provide the necessary solutions vector plots of the given matrices and their transformations (nos. 1-5.)

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} Y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} O = \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

```
H = 1/np.sqrt(2)*np.array([
    [1, 1],
    [1, -1],
  ])

Y = np.array([
    [0, -1j],
    [1j, 0],
])

O = np.array([
    [1, 0],
    [1, 1],
  ])
```

$H \cdot Y$

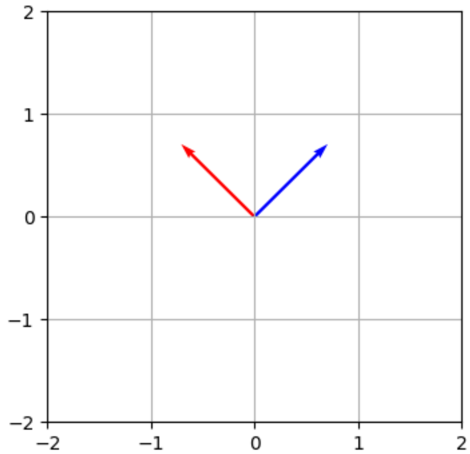
```
plot_quiv_imag(H @ Y)
magnitude= np.abs(np.linalg.norm(H @ Y))
angle = np.rad2deg(np.angle((np.sum(H @ Y))))
print(f'{H @ Y}\n')
print(f"Magnitude: {magnitude} \n")
print(f"Angle:{angle} \n")
```




```
[[0.+0.70710678j 0.-0.70710678j]
 [0.-0.70710678j 0.-0.70710678j]]
```

$Y \cdot H$

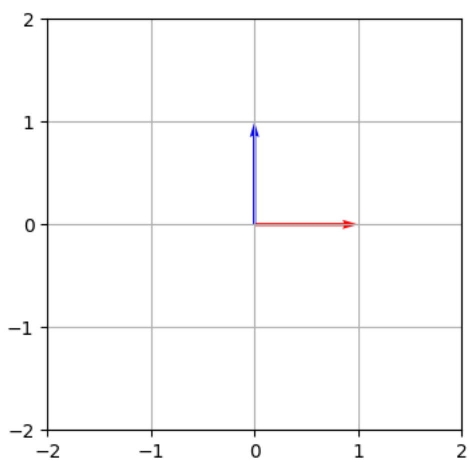
```
plot_quiv_imag(Y @ H)
magnitude= np.abs(np.linalg.norm(Y @ H))
angle = np.rad2deg(np.angle((np.sum(Y @ H))))
print(f'{Y @ H}\n')
print(f"Magnitude: {magnitude} \n")
print(f"Angle: {angle} \n")
```



```
[[0.-0.70710678j 0.+0.70710678j]
 [0.+0.70710678j 0.+0.70710678j]]
```

$H \cdot H$

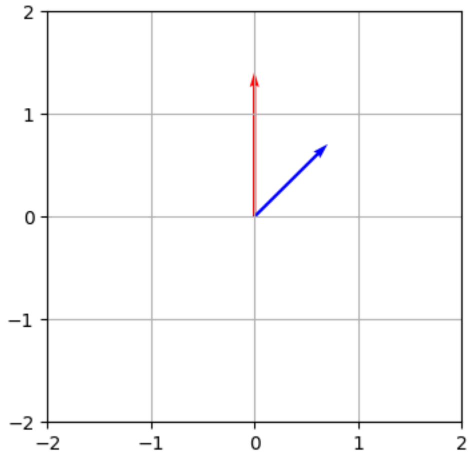
```
mag = np.linalg.norm(H @ H)
angle = np.rad2deg(np.angle((np.sum(H @ H))))
plot_quiv(H @ H)
print(f'{H @ H}\n')
print(f"Magnitude : {mag}\n")
print(f"Angle: {angle} \n")
```



```
[[ 1.00000000e+00 -2.23711432e-17]
 [-2.23711432e-17  1.00000000e+00]]
```

$Y \cdot H \cdot O$

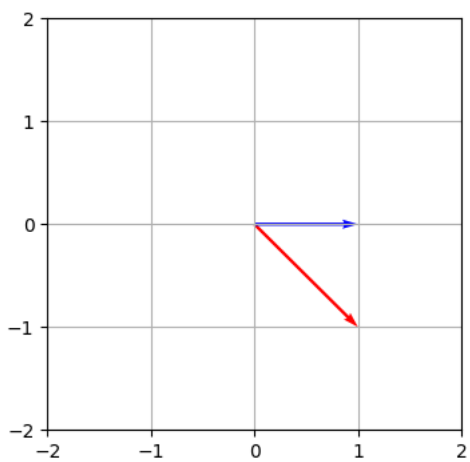
```
plot_quiv_imag(Y @ H @ O)
magnitude= np.linalg.norm((Y @ H @ O))
angle = np.rad2deg(np.angle((np.sum(Y @ H @ O))))
print(f'{Y @ H @ O}\n')
print(f"Magnitude:\n {magnitude} \n")
print(f"Angle: {angle} \n")
```




```
[[0.+0.j         0.+0.70710678j]
 [0.+1.41421356j 0.+0.70710678j]]
```

$H \cdot Y \cdot H \cdot O$

```
plot_quiv_imag(H @ Y @ H @ O)
magnitude= np.linalg.norm((H @ Y @ H @ O))
angle = np.rad2deg(np.angle((np.sum(H @ Y @ H @ O))))
print(f'{H @ Y @ H @ O}\n')
print(f"Magnitude:\n {magnitude} \n")
print(f"Angle: {angle} \n")
```




```
[[0.+1.00000000e+00j 0.+1.00000000e+00j]
 [0.-1.00000000e+00j 0.+2.23711432e-17j]]
```

PART 2

[50 pts] Solve for the determinants of H and Y. Provide necessary solutions numerically (handsolved) and computationally (using a Python program).

```
detH = np.linalg.det(H)
print(f'Determinant = {detH}')
```

```
Determinant = -0.9999999999999999
```

```
detY = np.linalg.det(Y)
print(f'Determinant = {detY}')
```

```
Determinant = (-1+0j)
```

PART 3

[50 pts] Determine whether the resulting linear transformations are linearly dependent. Provide necessary solutions both numerically using determinants (hand-solved) and computationally (using a Python program).

$$1.\begin{pmatrix}5&0&0\\0&5&0\\0&0&5\end{pmatrix}\cdot\begin{pmatrix}1&0&5\\2&7&6\\6&4&7\end{pmatrix}$$

```
X = np.eye(3)
L = np.array([
    (5, 0, 0),
    (0, 5, 0),
    (0, 0, 5),
  ])

K = np.array([
    (1, 0, 5),
    (2, 7, 6),
    (6, 4, 7),
  ])

detL = np.linalg.det(L @ K)

if detL == 0:
    print("The vectors are linearly dependent. \n")
else:
    print("The vectors are linearly independent. \n")

print(f'{L @ K} \n')
print(f'Determinant = {detL}')

plot_3d_quiv(L@X, 45, 15)
```

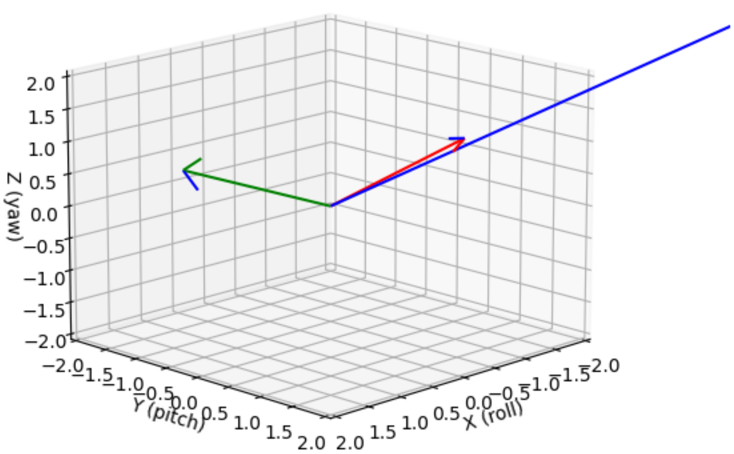
```
The vectors are linearly independent.
```

$$2. \begin{pmatrix} 1 & 2 & 6 \\ 3 & 15 & 4 \\ 2 & 10 & 3 \end{pmatrix} \cdot \begin{pmatrix} 5 & 2 & 4 \\ 6 & 2 & 4 \\ 0 & 1 & 1 \end{pmatrix}$$

```
Determinant = -18124.99999999999
```

```
X = np.eye(3)
I = np.array([
    (1, 2, 6),
    (3, 15, 4),
    (2, 10, 3),
  ])
R = np.array([
    (5, 2, 4),
    (6, 2, 4),
    (0, 1, 1),
  ])

detI = np.linalg.det(I @ R)

if detI == 0:
    print("The vectors are linearly dependent. \n")
else:
    print("The vectors are linearly independent. \n")

print(f'{I @ R} \n')
print(f'Determinant = {detI}')

plot_3d_quiv(I @ X, 45, 15)
```

```
The vectors are linearly independent.

[[ 17  12  18]
 [105  40  76]
 [ 70  27  51]]

Determinant = 6.000000000001367
```



PART 4

[50 pts] Plot the signal corresponding to the following vectors whereas T is the time vector and G is the vector corresponding to the amplitudes of the signal.

$$T = \begin{pmatrix} 1 & \frac{\pi}{4} & \frac{\pi}{2} & \frac{3\pi}{4} & \pi \end{pmatrix}; G = \begin{pmatrix} 5 & 3 & 0 & -3 & 5 \end{pmatrix}$$

```
T = np.array([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi])
```

```
G = np.array([5, 3, 0, -3, 5])
```

```
plt.figure(figsize=(8, 4))
plt.plot(T, G, marker='o', linestyle='-', color='b', markersize=8)
plt.xlabel('Time (T)')
plt.ylabel('Amplitude (G)')
plt.title('Signal Plot')
plt.grid(True)
plt.show()
```