**Assignment 3**

**Team Members**

|  |  |  |  |
| --- | --- | --- | --- |
| **Num** | **Full Name in ARABIC** | SEC | BN |
| 1 | علياء عبدالعزيز علي عبدالله | 1 | 26 |
| 2 |  |  |  |

**Table of contents:**

[1. Part One 3](#_Toc135281088)

[1.1 Gram-Schmidt Orthogonalization 3](#_Toc135281089)

[1.2 Signal Space Representation 4](#_Toc135281090)

[1.3 Signal Space Representation with adding AWGN 5](#_Toc135281091)

[1.4 Noise Effect on Signal Space 6](#_Toc135281092)

[2. Appendix A: Codes for Part One: 7](#_Toc135281093)

[A.1 Code for Gram-Schmidt Orthogonalization 7](#_Toc135281094)

[A.2 Code for Signal Space representation 7](#_Toc135281095)

[A.3 Code for plotting the bases functions 7](#_Toc135281096)

[A.4 Code for plotting the Signal space Representations 8](#_Toc135281097)

[A.5 Code for effect of noise on the Signal space Representations 8](#_Toc135281098)

**List of Figures**

[Figure 1 Φ1 VS time after using the GM\_Bases function 5](#_Toc135281077)

[Figure 2 Φ2 VS time after using the GM\_Bases function 5](#_Toc135281078)

[Figure 3 Signal Space representation of signals s1,s2 6](#_Toc135281079)

[Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB 7](#_Toc135281080)

[Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB 7](#_Toc135281081)

[Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB 8](#_Toc135281082)

# Part One

## 1.1 Gram-Schmidt Orthogonalization

Gram-Schmidt orthogonalization is a process of transforming a set of linearly independent vectors into an orthonormal basis for the same subspace.

A graph with a line

Description automatically generated

Figure 1 Φ1 VS time after using the GM\_Bases function

A graph of a graph

Description automatically generated

Figure 2 Φ2 VS time after using the GM\_Bases function

## 

## 1.2 Signal Space Representation

Here we represent the signals using the base functions.

A graph on a white grid

Description automatically generated

Figure 3 Signal Space representation of signals s1,s2

## 1.3 Signal Space Representation with adding AWGN

-the expected real points will be solid and the received will be hollow

**Case 1**:

A diagram of a signal representation

Description automatically generated

Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

**Case 2**:

A diagram of a signal representation

Description automatically generated

Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

**Case 3**:

A diagram of a signal representation

Description automatically generated

Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB

## 1.4 Noise Effect on Signal Space

Presence of noise in signal space can have multiple effects on representation of signals:

* As SNR decreases the signal power becomes weaker than noise power making it more challenging to distinguish between signal and noise.

In terms of how SNR work remember its calculation led to (E/sigma^2) where (E) for signal and (sigma^2) is variance of noise

* Increasing variance of noise led to increased variability of received signal in terms of (amplitude, frequency, phase)
  + Higher (E/sigma^2) (SNR) values reduce deviation of received signal from original by decreasing variance of AWGN (additive white gaussian noise) and vice versa.

# Appendix A: Codes for Part One:

## A.0 Code for Utils, Constants, Libraries

import numpy as np

import matplotlib.pyplot as plt

num\_samples = 100

def get\_sampled\_energy(s):

    return np.sum( s\*s ) / num\_samples

## A.1 Code for Gram-Schmidt Orthogonalization

def gram\_schmidt(s1, s2):

    # take phi\_1 as normalized of first signal

    phi\_1 = s1 / get\_sampled\_energy(s1)\*\*0.5

    # calculate phi\_2

    phi\_2 = s2 - np.dot(phi\_1, s2) / num\_samples \* phi\_1

    phi\_2\_norm = get\_sampled\_energy(phi\_2)\*\*0.5

    if phi\_2\_norm == 0:

        phi\_2 = np.zeros\_like(phi\_2)

    else:

        phi\_2 /= phi\_2\_norm

    return phi\_1, phi\_2

## A.2 Code for Signal Space representation

def signal\_space(s, phi\_1, phi\_2):

    return np.dot(s, phi\_1)/num\_samples, np.dot(s, phi\_2)/num\_samples

## A.3 Code for plotting the bases functions

Construct time (x-axis)

time = np.arange(0, 1, 1 / num\_samples)

Construct s1(t), s2(t)

s1 = np.ones\_like(time)

s1[time >= 1] = 0

s2 = np.ones\_like(time)

s2[time > 0.75] = -1

phi\_1, phi\_2 = gram\_schmidt(s1, s2)

plt.plot(time, phi\_1)

plt.title("phi\_1(t)")

plt.xlabel("Time (sec)")

plt.ylabel("Amplitude")

plt.grid(True)

plt.show()

plt.plot(time, phi\_2)

plt.title("phi\_2(t)")

plt.xlabel("Time (sec)")

plt.ylabel("Amplitude")

plt.grid(True)

plt.show()

## A.4 Code for plotting the Signal space Representations

x1, y1 = signal\_space(s1, phi\_1, phi\_2)

x2, y2 = signal\_space(s2, phi\_1, phi\_2)

plt.figure(figsize=(10,7))

plt.plot(x1, y1, 'bo', label="signal 1 (s1)", linewidth=2)

plt.plot(x2, y2, 'ro', label="signal 2 (s2)", linewidth=2)

plt.quiver(0, 0, x1, y1,  angles='xy', scale\_units='xy', scale=1, color='b')

plt.quiver(0, 0, x2, y2,  angles='xy', scale\_units='xy', scale=1, color='r')

plt.title('Signal Space Representation')

plt.xlabel("phi\_1")

plt.ylabel("phi\_2")

plt.legend()

plt.grid()

## A.5 Code for effect of noise on the Signal space Representations

sigma\_db\_values = [-5, 0, 10]

x1, y1 = signal\_space(s1, phi\_1, phi\_2)

x2, y2 = signal\_space(s2, phi\_1, phi\_2)

for sigma\_db in sigma\_db\_values:

    # add noise

    r1 = add\_AWGN(s1, sigma\_db, num\_samples)

    r2 = add\_AWGN(s2, sigma\_db, num\_samples)

    # get signal space

    v11, v12 = signal\_space(r1, phi\_1, phi\_2)

    v21, v22 = signal\_space(r2, phi\_1, phi\_2)

    # plot signal

    plt.scatter(v11, v12, label='s1 + noise', facecolors='none', edgecolors='orange')

    plt.scatter(v21, v22, label='s2 + noise', facecolors='none', edgecolors='g')

    plt.scatter(x1, y1, label='s1', color='r')

    plt.scatter(x2, y2, label='s2', color='b')

    plt.title(f"Signal Representation with E/Var = {sigma\_db} db")

    plt.xlabel("phi\_1")

    plt.ylabel("phi\_2")

    plt.legend()

    plt.show()