Faculty of Engineering
Computer Department
Communications (ELC 325B) – Spring 2023

Assignment 3

Team Members

Num	Full Name in ARABIC	SEC	BN
1	سارة بشير فكري بشير	1	18
2	منه محمد عبدالباسط علي ابراهيم	2	26

Table of contents:

List of Figures

1. Part One

1.1 Gram-Schmidt Orthogonalization

We have plotted Φ1(t) after calculating it using Gram Schmidt as
Φ1(t) is s1(t) divided by the normalization of s1(t)
And then calculate Φ2(t) which is s2(t) but after the removal of the component in Φ1(t) and then normalizing it
To have Φ1, Φ2 orthonormal to each other

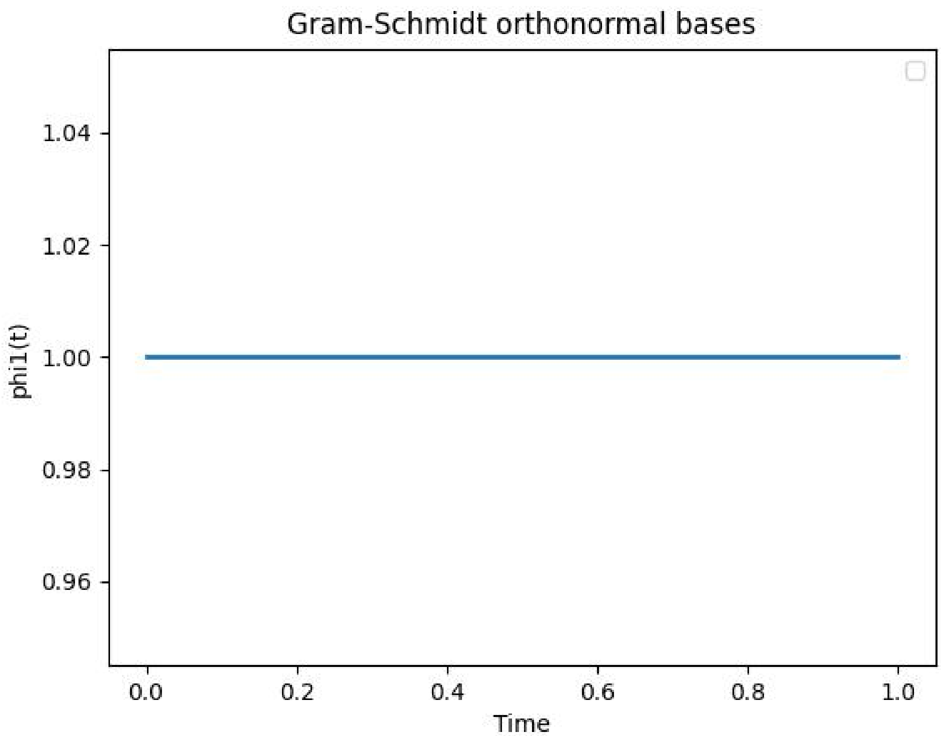

Figure 1 Φ1 VS time after using the GM_Bases function

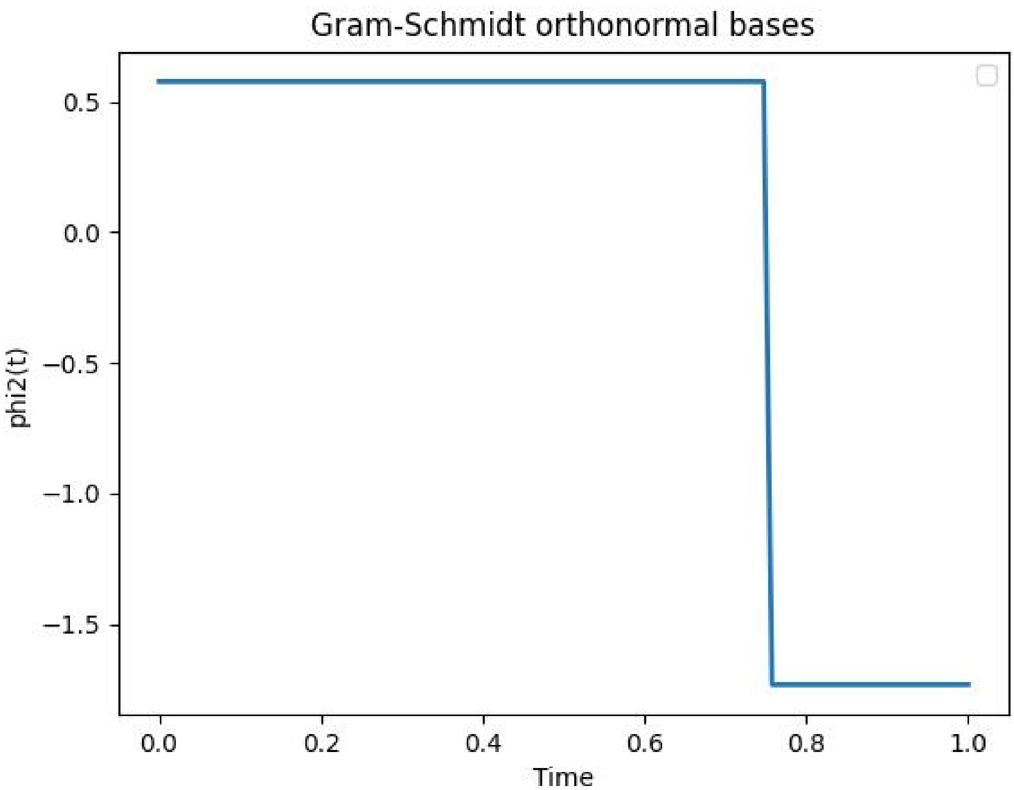

Figure 2 Φ2 VS time after using the GM_Bases function

1.2 Signal Space Representation

Here we represent the signals using the base functions.

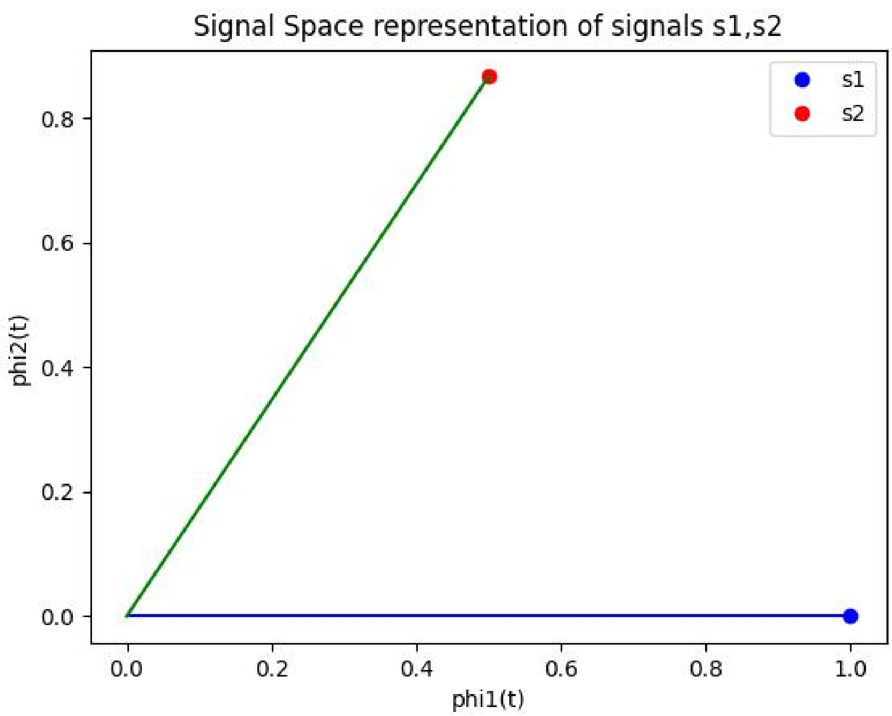

Figure 3 Signal Space representation of signals s1,s2

1.3 Signal Space Representation with adding AWGN

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

Case 1: $10\,log(E/\sigma^2) \;=\; 10\,dB$

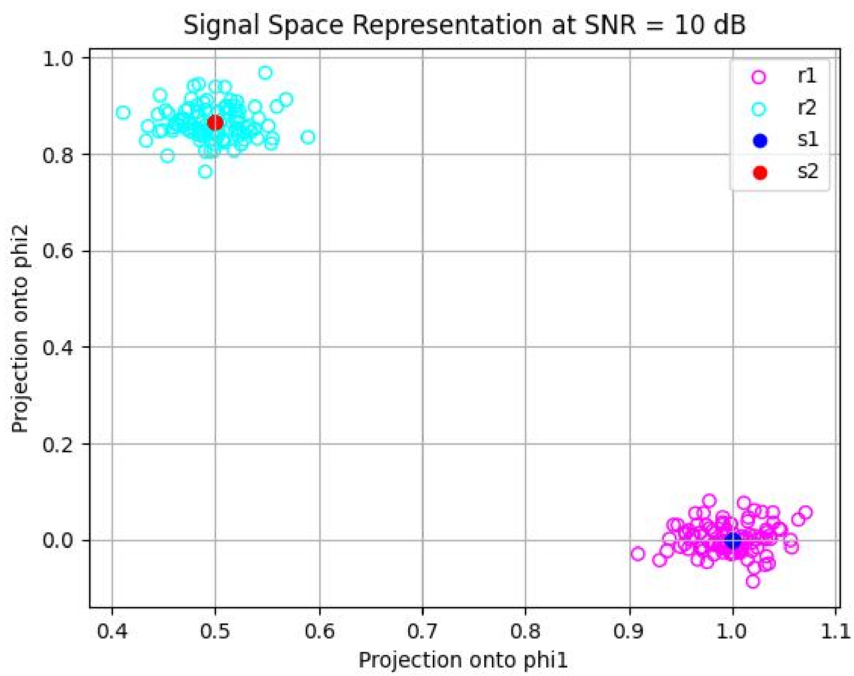

Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

Case 2: $\mathbf{10}\, \boldsymbol{log(E/\sigma^2)} = \mathbf{0}\, \boldsymbol{dB}$

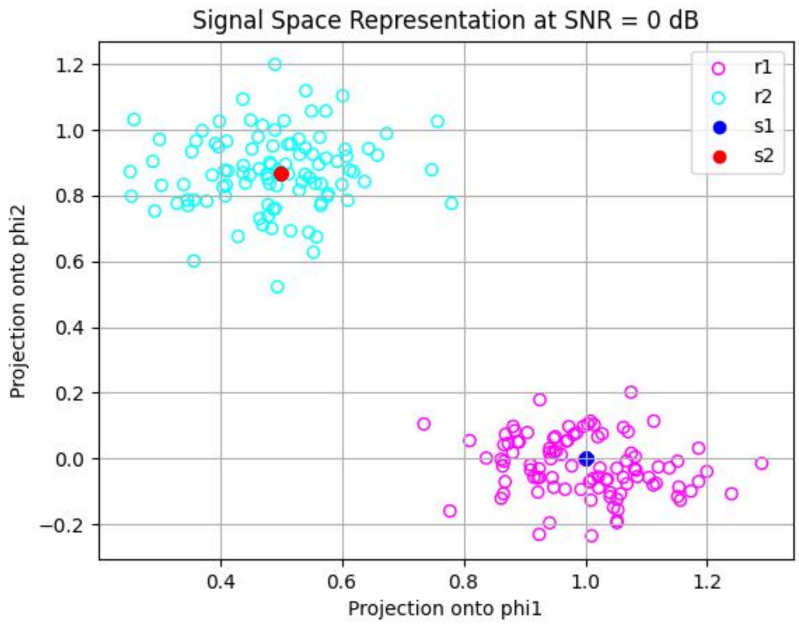

Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

Case 3: $\mathbf{10}\, \boldsymbol{log(E/\sigma^2)} = -\mathbf{5}\, \boldsymbol{dB}$

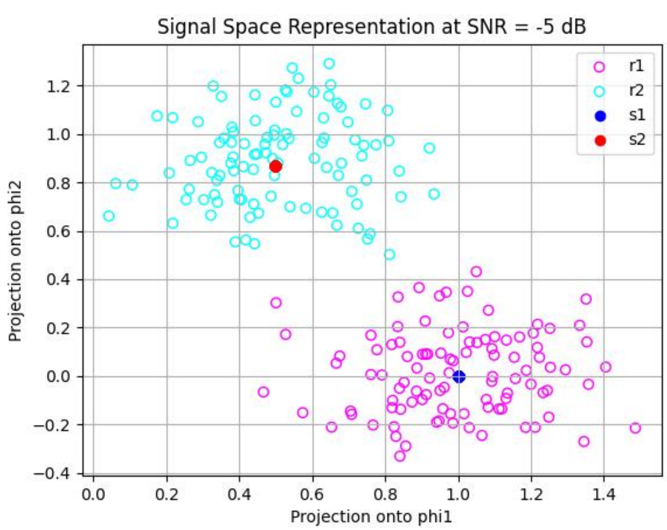

Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB

Comment:

From the previous three plots, It is shown that when the value of SNR increases the points get close more to the intended point that it should belong to as probability of error decreases.

1.4 Noise Effect on Signal Space

As SNR increases the noise effect decreases as sigma decreases

so it enhances the difference between the two signals in the signal space and minimizes the probability of error

2. Appendix A: Codes for Part One:

A.1 Code for Gram-Schmidt Orthogonalization

```
def mag(s):
    return np.sqrt(np.dot(s,s)/sample_num)
def GM_Bases(s1, s2):
    phi1 = s1 / mag(s1)
    phi2 = s2 -((np.dot(phi1, s2) * phi1)/sample_num)
    phi2 = phi2 / mag(phi2)
    return phi1, phi2
```

A.2 Code for Signal Space representation

```
def signal_space(s, phi1, phi2):
    v1 = np.dot(s, phi1)/sample_num
    v2 = np.dot(s, phi2)/sample_num
    return v1, v2
```

A.3 Code for plotting the bases functions

```
phi1,phi2=GM_Bases(s1,s2)

plt.plot(t,phi1,linewidth=2)
plt.title("Gram-Schmidt orthonormal bases")
plt.xlabel("Time")
plt.ylabel("phi1(t)")
plt.legend()
plt.show()

plt.plot(t,phi2,linewidth=2)
plt.title("Gram-Schmidt orthonormal bases")
plt.xlabel("Time")
plt.ylabel("phi2(t)")
plt.legend()
plt.show()
```

A.4 Code for plotting the Signal space Representations

```
v1,v2=signal_space(s1,phi1,phi2)
v3,v4=signal_space(s2,phi1,phi2)

# Plot the signal space representation for the two signals
plt.plot(v1,v2,'bo',label='s1')
plt.plot(v3,v4,'ro',label='s2')
plt.plot([0, v1], [0, v2], 'b')
plt.plot([0, v3], [0, v4], 'g')
plt.title("Signal Space representation of signals s1,s2")
plt.xlabel("phi1(t)")
plt.ylabel("phi2(t)")
plt.legend()
plt.show()
```

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

```
# calculate the Energy of each signal
e1 = np.dot([v1,v2],[v1,v2])
e2 = np.dot([v3,v4],[v3,v4])



# Function to add noise to a signal based on SNR in dB
def add_noise(s, SNR_dB,E):
    sigma2 = E * 10**(-SNR_dB/10)
    noise = np.random.normal(0, np.sqrt(sigma2), sample_num)
    return s + noise



# SNRs to test
SNRs = [-5, 0, 10]

for SNR in SNRs:
    # Signal space projection
    v1_r1_org, v2_r1_org = signal_space(s1, phi1, phi2)
    v1_r2_org, v2_r2_org = signal_space(s2, phi1, phi2)
    plt.figure()
    plt.title('Signal Space Representation at SNR = {} dB'.format(SNR))

    for i in range(sample_num):
```

```
    r1 = add_noise(s1, SNR,e1)
    r2 = add_noise(s2, SNR,e2)

    v1_r1, v2_r1 = signal_space(r1, phi1, phi2)
    v1_r2, v2_r2 = signal_space(r2, phi1, phi2)
    # Plotting
    plt.scatter(v1_r1, v2_r1, color='magenta',facecolors='none', marker='o',label='r1')
    plt.scatter(v1_r2, v2_r2, color='cyan',facecolors='none',marker='o',label='r2')

    plt.scatter(v1_r1_org, v2_r1_org, color='blue', label='s1')
    plt.scatter(v1_r2_org, v2_r2_org, color='red', label='s2')
plt.legend(['r1','r2','s1','s2'])
plt.grid(True)
plt.xlabel('Projection onto phi1')
plt.ylabel('Projection onto phi2')
plt.show()
```