Assignment 3

Team Members

Num	Full Name in ARABIC	SEC	BN
1	منة الله احمد مصطفى عبده	2	25
2	مايكل ايهاب ميخائيل عبد الملاك	2	5

Table of contents:

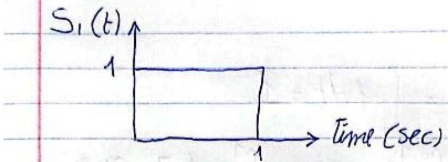
List of Figures

1. Part One

1.1 Gram-Schmidt Orthogonalization

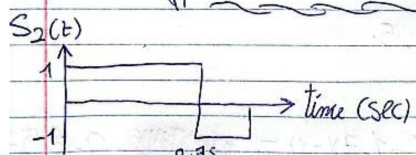
The Gram-Schmidt Orthogonalization in communication systems creates orthogonal signals to reduce interference and increase system efficiency, specially in wireless communications where signals overlap in frequency.

Hand Analysis



$$E = \int_0^T S_1^2(t)\,dt = \int_0^1 1\,dt = (1-0) = 1 \qquad \therefore S_{11} = \sqrt{1} = 1$$

$$S_1(t) = S_{11}\,\phi_1 \qquad \phi_1 = S_1(t)$$



$$S_2(t) = S_{21}\,\phi_1 + S_{22}\,\phi_2$$

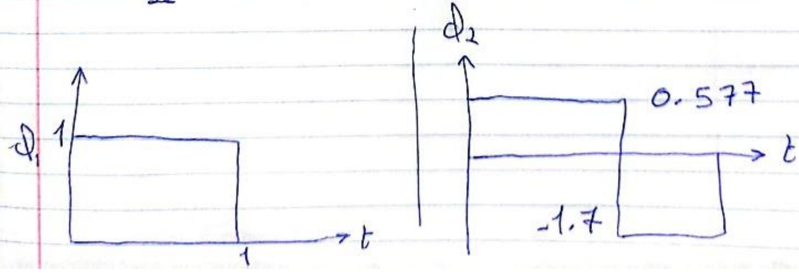
$$S_{21} = \int_0^T S_2(t)\,\phi_1\,dt = \int_0^T S_2(t)\,dt = \int_0^{0.75} 1\,dt + \int_{0.75}^{1} -1\,dt = 0.5$$

$$g_2 = S_2(t) - S_{21}\,\phi_1$$

$$S_{22} = \sqrt{\int_0^T g_2^2(t)\,dt} = \sqrt{\int_0^{0.75} (1-0.5)^2 + \int_{0.75}^{1} (-1-0.5)^2} = 0.866$$

$$S_2(t) = S_{21}\,\phi_1 + S_{22}\,\phi_2$$

$$\phi_2 = \frac{g_2}{S_{22}} \qquad = 0.5\,\phi_1 + 0.866\,\phi_2$$



Simulation Results

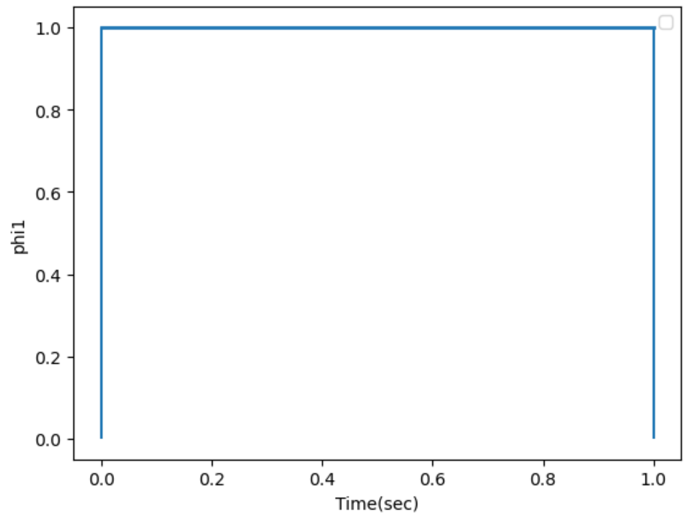


Figure 1 Φ1 VS time after using the GM_Bases function

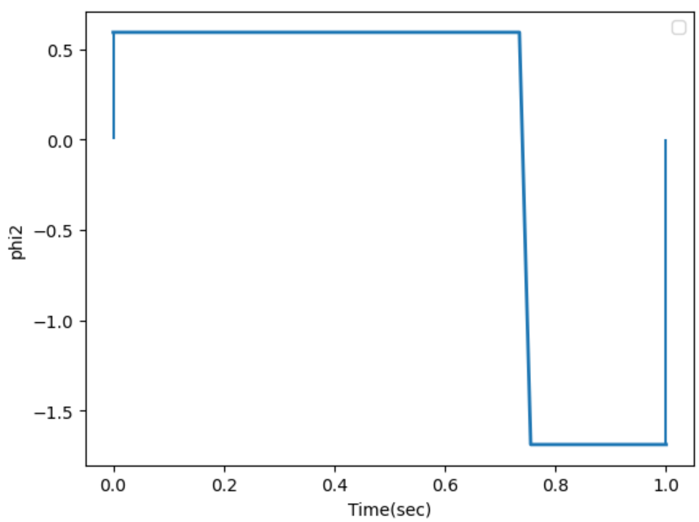


Figure 2 Φ2 VS time after using the GM_Bases function

1.2 Signal Space Representation

Here we represent the signals using the base functions.

Hand Analysis:

For S_1:

$$V_1 = \int_0^T S_1 \phi_1 \, dt = \int_0^1 1 \, dt = 1$$

$$V_2 = \int_0^T S_1 \phi_2 \, dt = \int_0^{0.75} 0.577 + \int_{0.75}^1 -1.7 \simeq 0$$

For S_2:

$$V_1 = \int_0^{0.75} 1 + \int_{0.75}^1 -1 = 0.5$$

$$V_2 = \int_0^{0.75} (0.577 \times 1) + \int_{0.75}^1 (-1.7 \times -1) = 0.85775$$

Simulation Results

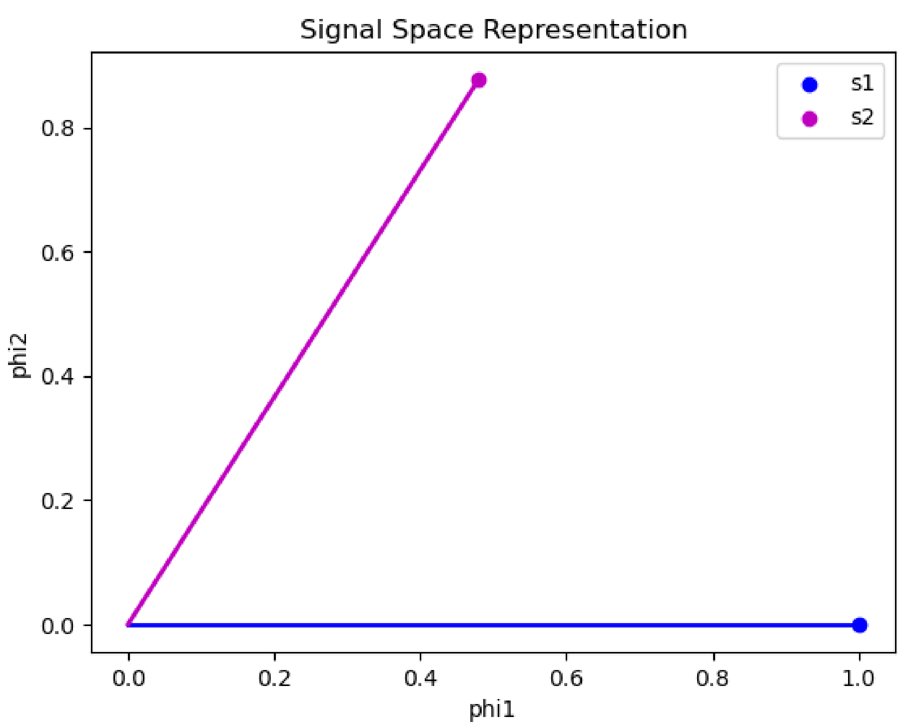


Figure 3 Signal Space representation of signals s1,s2

1.3 Signal Space Representation with adding AWGN

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

Case 1: $\mathbf{10}\ \boldsymbol{log}(\boldsymbol{E}/\boldsymbol{\sigma}^{\mathbf{2}}) = \mathbf{10}\ \boldsymbol{dB}$

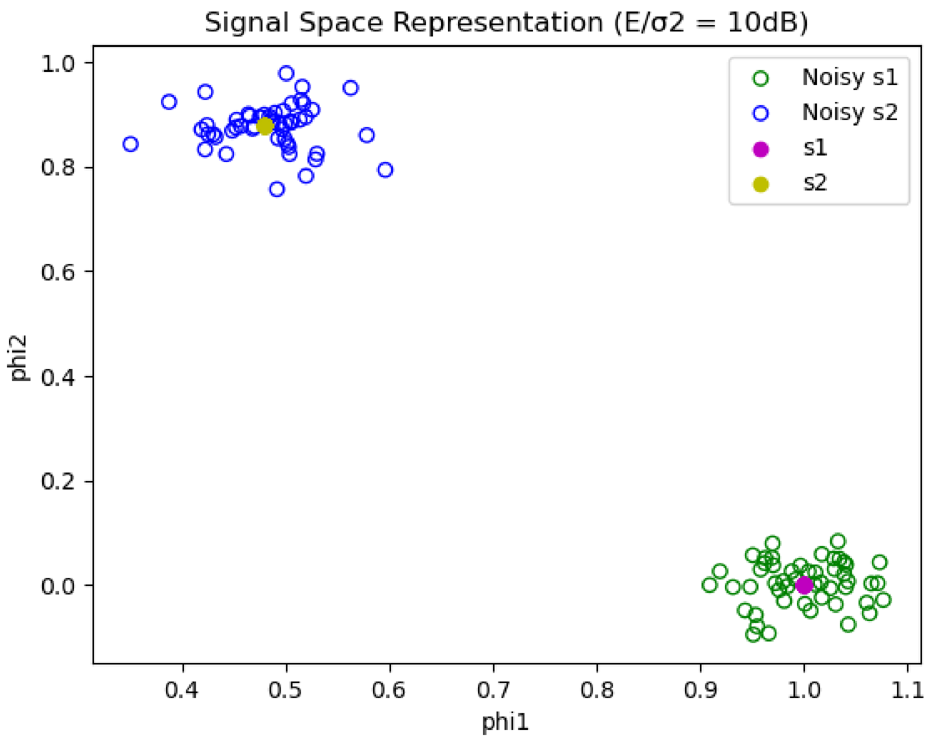


Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

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

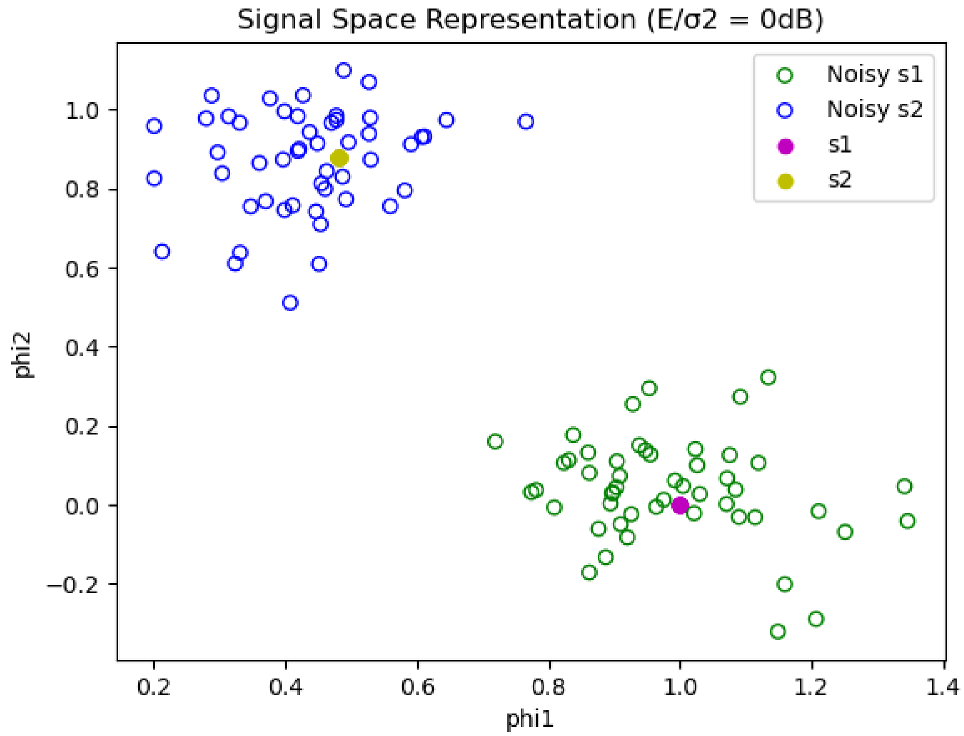


Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

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

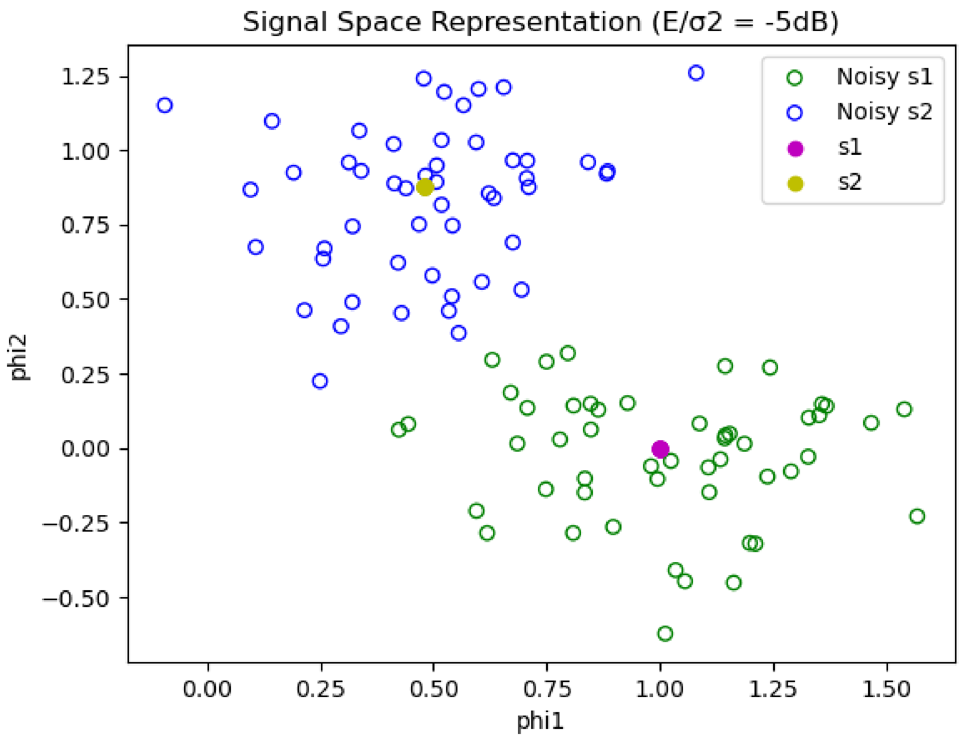


Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB

1.4 Noise Effect on Signal Space

Question: How does the noise affect the signal space? Does the noise effect increase or decrease with increasing σ^2?

Answer:

As σ^2 increases the E/σ^2 decreases so the noise effect increases and the received points are far from the expected point (As seen in figure 6)

As σ^2 decreases the E/σ^2 increases so the noise effect decreases and the received points are close to the expected point (As seen in figure 4)

2. Appendix A: Codes for Part One:

A.1 Code for Gram-Schmidt Orthogonalization

```
def GM_Bases(s1, s2):
    # Calculate the energy of the first signal
    E1 = np.sum(s1**2) / numOfSamples
    # Create the first orthogonal basis (phi1) by normalizing the
first signal
    phi1 = s1 / np.sqrt(E1)

    # Project the second signal onto the first basis (phi1)
    s21 = np.sum(s2*phi1) / numOfSamples
    # Subtract the projection from the second signal to make it
orthogonal to the first basis
    g2 = s2 - s21 * phi1
    # Calculate the energy of the new, orthogonalized second
signal
    E2 = np.sum(g2**2) / numOfSamples

    # Create the second orthogonal basis (phi2) by normalizing
the new second signal
    phi2 = g2 / np.sqrt(E2)
    # Check if the two bases are identical, if so, set the second
basis to a zero vector
    if np.array_equal(phi1, phi2):
        phi2 = np.zeros(numOfSamples)

    # Return the two orthogonal bases
```

```
    return phi1, phi2
```

A.2 Code for Signal Space representation

```
def signal_space(s, phi1, phi2):
    # Project the signal 's' onto the first basis 'phi1' to find its coefficient 'v1'
    v1 = np.sum(s*phi1) / numOfSamples
    # Project the signal 's' onto the second basis 'phi2' to find its coefficient 'v2'
    v2 = np.sum(s*phi2) / numOfSamples
    # Return the coefficients 'v1' and 'v2' which represent the signal 's' in the new basis
    return v1, v2
```

A.3 Code for plotting the bases functions

```
# Calculate bases functions for given s1,s2
phi1, phi2 = GM_Bases(s1, s2)

plt.figure()
plt.plot(time_axis, phi1, linewidth=2)
plt.vlines(x=0, ymin=0, ymax=phi1[0])
plt.vlines(x=1, ymin=phi1[-1], ymax=0)
plt.xlabel("Time(sec)")
plt.ylabel("phi1")
plt.legend()

plt.figure()
```

```
plt.plot(time_axis, phi2, linewidth=2)
plt.vlines(x=0, ymin=0, ymax=phi2[0])
plt.vlines(x=1, ymin=phi2[-1], ymax=0)
plt.xlabel("Time(sec)")
plt.ylabel("phi2")
plt.legend()
```

A.4 Code for plotting the Signal space Representations

```
v11, v12 = signal_space(s1, phi1, phi2)
v21, v22 = signal_space(s2, phi1, phi2)

# Plot signal space representation
plt.figure()
plt.scatter(v11, v12, label='s1', c='b')
plt.scatter(v21, v22, label='s2', c='m')
plt.plot([0, v11], [0, v12], 'b', linewidth=2)
plt.plot([0, v21], [0, v22], 'm', linewidth=2)
plt.title('Signal Space Representation')
plt.xlabel("phi1")
plt.ylabel("phi2")
plt.legend()
plt.show()
```

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

```
# An array of E/σ^2 values in dB
E_over_sigma_2_arr = [-5, 0, 10]

for E_over_sigma_2_db in E_over_sigma_2_arr:
    plt.figure()
    # Set the title of the plot with the current E/σ^2 value
    plt.title('Signal Space Representation (E/σ^2 = '+
str(E_over_sigma_2_db) + 'dB)')
    plt.xlabel('phi1')
    plt.ylabel('phi2')

    for n in range(numOfSamples):
        # Convert E/σ^2 from dB to linear scale
        E_over_sigma_2 = 10**(E_over_sigma_2_db / 10)

        # Calculate noise for S1
        E1 = np.sum(s1**2) / numOfSamples  # Calculate the energy
of S1
        sigma_squared = E1 / E_over_sigma_2  # Calculate the
noise variance for S1
        sigma = np.sqrt(sigma_squared)  # Calculate the standard
deviation of the noise
        noise1 = np.random.normal(0, sigma, numOfSamples)  #
Generate noise for S1

        # Calculate noise for S2
        E2 = np.sum(s2**2) / numOfSamples  # Calculate the energy
of S2
```

```
        sigma_squared = E2 / E_over_sigma_2  # Calculate the
noise variance for S2
        sigma = np.sqrt(sigma_squared)  # Calculate the standard
deviation of the noise
        noise2 = np.random.normal(0, sigma, numOfSamples)  #
Generate noise for S2

        r1 = s1 + noise1  # Add noise to S1 to get the received
signal R1
        r2 = s2 + noise2  # Add noise to S2 to get the received
signal R2

        # Project the noisy received signals onto the signal
space
        v11_noisy, v12_noisy = signal_space(r1,phi1,phi2)
        v21_noisy ,v22_noisy = signal_space(r2,phi1,phi2)

        # Plot the noisy projections of S1 and S2
        plt.scatter(v11_noisy, v12_noisy, facecolors='none',
edgecolors='g')
        plt.scatter(v21_noisy, v22_noisy, facecolors='none',
edgecolors='b')

        # Plot the original projections of S1 and S2 without
noise
        plt.scatter(v11, v12, c='m')
        plt.scatter(v21, v22, c='y')
```

```python
plt.legend(['Noisy s1','Noisy s2','s1','s2'])
```