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

Assignment 3

Submitted to Eng Mohamed Khaled

Team Members

Num	**Full Name in ARABIC**	SEC	BN
1	بسمه حاتم فريد الحسيني	1	17
2	زينب معوض فايز	1	29

Table of contents:

List of Figures

1. Part One

1.1 Gram-Schmidt Orthogonalization

In general The Gram–Schmidt orthonormalization process is a procedure for orthonormalizing a set of vectors in an inner product space So we used it here in the vector space representation of the signal so that we can find the basis to represent different messages on a lower dimension (less no of correlators are needed the receiver)

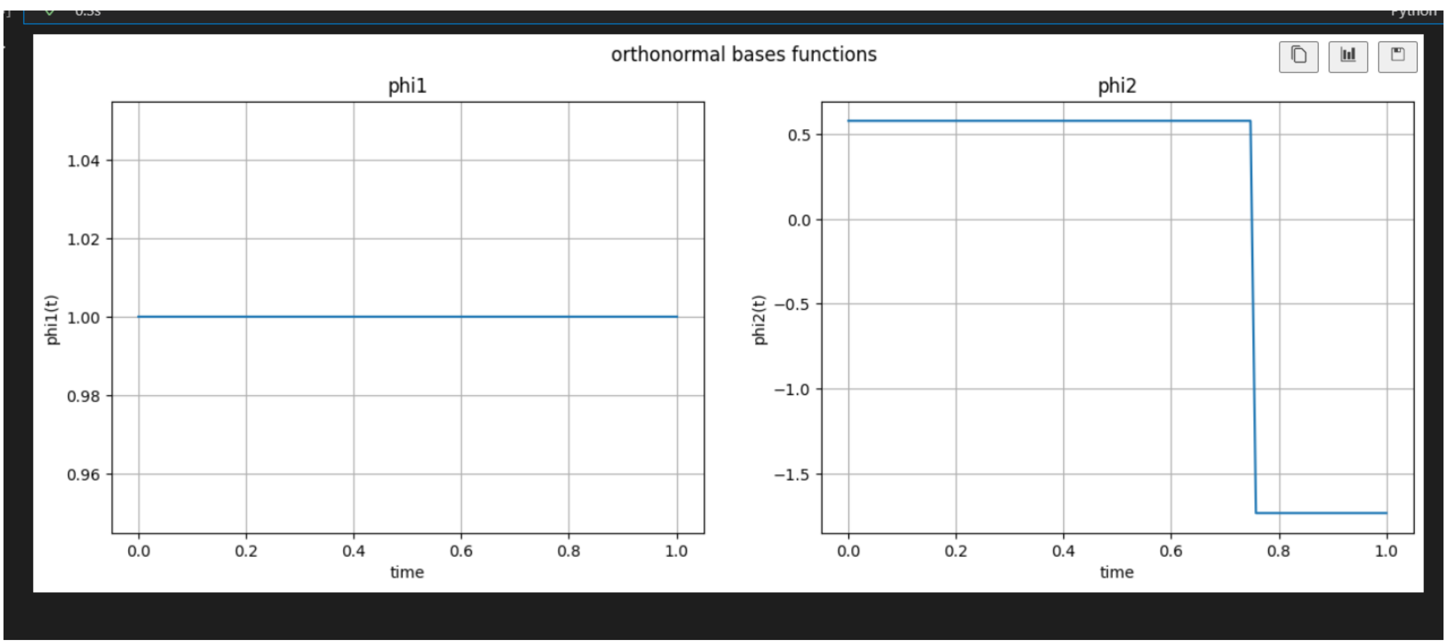

Figure 1 Φ1 VS time after using the GM_Bases function **Figure 2 Φ2 VS time after using the GM_Bases function**

1.2 Signal Space Representation

Here we represent the signals using the base functions.

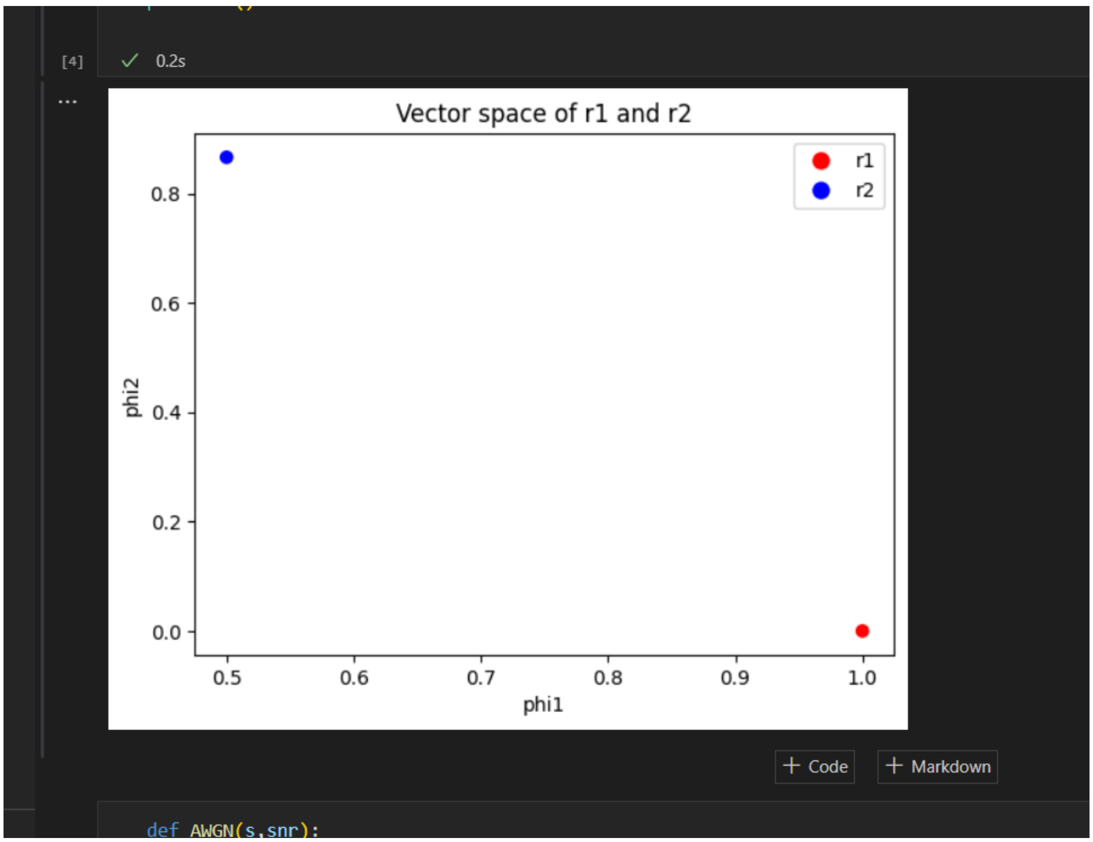

Figure 3 Signal Space representation of signals s1,s2

1.3 Signal Space Representation with adding AWGN

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

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

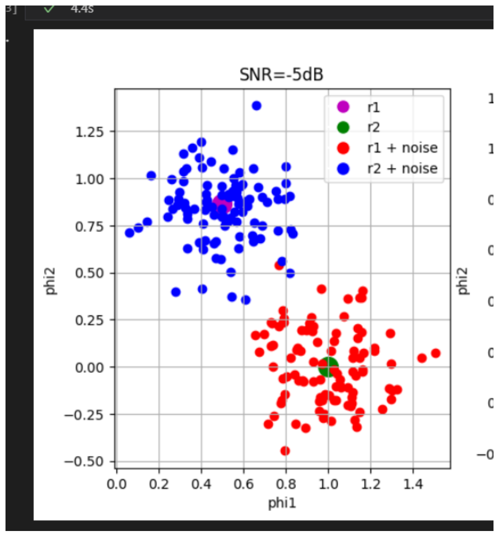

Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

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

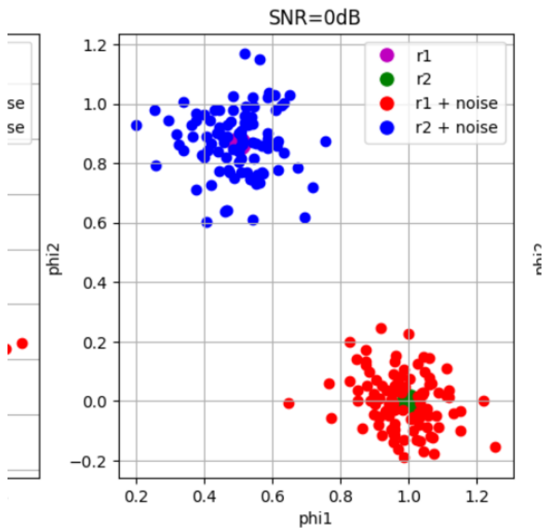

Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

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

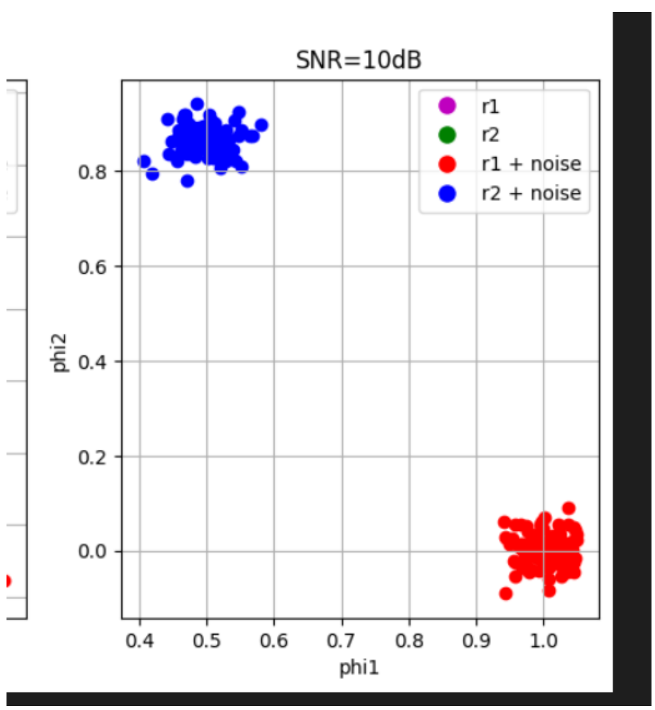

Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB

1.4 Noise Effect on Signal Space

How does the noise affect the signal space?

📝 It is clear that the noise makes signal point [representation] in the vector space noisy (scattered around the true value without noise)

Does the noise effect increase or decrease with increasing $\sigma 2$?

📝 It is clear that as snr values increases the effect AEGN on the signal [shift from original value] decreases it is logic bec snr is high means that the signal power is more than that of noise ie. as $\sigma 2$ increase(snr decrease) the effect of noise decreases

2. Appendix A: Codes for Part One:

A.1 Code for Gram-Schmidt Orthogonalization

```
def GM_Bases(s1,s2):
    '''
    The function calculates the Gram-Schmidt orthonormal bases functions (phi1 & phi 2) for two input signals (s1 & s2)
    '''
    # Getting phi1
    # s1=s11* phi1
    # s11=root(E1)
    E1=np.sum(s1**2)/samples
    s11=math.sqrt(E1)
    phi1=s1/s11


    # Getting phi2
    # s2= s21*phi1 + s22*phi2
    # Getting s21 =intg(0-T)(s2 phi1) 4
    s21=np.sum(s2*phi1)/samples
    # s22 phi2=s2-s21 phi1 = g2(t)
    # computing s22=root(E2)
    g2=s2-s21*phi1
    E2=np.sum(g2**2)/samples
    s22=math.sqrt(E2)
    phi2=g2/s22

    return phi1,phi2
```

A.2 Code for Signal Space representation

```
def signal_space(s, phi1, phi2):
    '''The function calculates the signal space representation of input signal s over the
    orthonormal bases functions (phi1 & phi 2)'''

    # si=si1*phi1 + si2*phi2
    # step 1 compute si1=intg 0-T si*phi1
    si1 = np.sum(s*phi1)/samples

    # step 2 compute si2
    # si2 * phi2=si-si1*phi1=g2
    # si2 = intg 0-T g2*phi2
    g2 = s-si1*phi1
    si2 = np.sum(g2*phi2)/samples

    return [si1, si2]
```

A.3 Code for plotting the bases functions

```
phi1,phi2=GM_Bases(r1,r2)

#plot orthonormal bases functions 😉
figure, (ax1,ax2) = plt.subplots(1, 2, figsize=(15, 5))

figure.suptitle(' orthonormal bases functions')

ax1.grid()
ax1.set_title('phi1')
ax1.set_xlabel('time')
ax1.set_ylabel('phi1(t)')

ax2.set_title('phi2')
ax2.set_xlabel('time')
ax2.set_ylabel('phi2(t)')
ax2.grid()


#Plot s1
ax1.plot(t, phi1)
# Plot s2
ax2.plot(t, phi2)

plt.show()
```

A.4 Code for plotting the Signal space Representations

```
# Test this function by passing s1 and s2 to it 🫤
v_1 = signal_space(r1, phi1, phi2)
assert v_1 == [1.0, 0.0], "Error:Vector space representation of s1 is wrong"

# plot vector space of s1 and s2

v_2 = signal_space(r2, phi1, phi2)
v_2 = ['%.3f' % v for v in v_2]
v_2 = [float(x) for x in v_2]
assert v_2 == [
    0.500, 0.866], "Error:Vector space representation of s2 is wrong"


# # Set the width of the plot
# fig = plt.gcf()
# fig.set_size_inches(10, 5)


# Set axis labels and title
plt.xlabel('phi1')
plt.ylabel('phi2')
plt.title('Vector space of r1 and r2')

# plt.
plt.scatter([v_1[0],v_2[0]], [v_1[1],v_2[1]], c=['r','b'])

# Legend
# Create a custom legend for the colors
legend_elements = [plt.Line2D([0], [0], marker='o', color='w', label='r1',
                              markerfacecolor='r', markersize=10),
                   plt.Line2D([0], [0], marker='o', color='w', label='r2',
                              markerfacecolor='b', markersize=10)]

# Add the legend to the plot
plt.legend(handles=legend_elements)


# Show the plot
plt.show()
```

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

```python
def AWGN(s,snr):
    '''
    s: signal to which we want to add noise :(
    snr: Signal to noise ratio in db
    no_samples: for the noise generated

    '''
    #Generate AWGN
    #Where w(t) is a zero mean AWGN with variance σ^2
    # SNR=E/σ^2   E: Energy of the signal and σ^2 is variance of noise
    mu=0
    E=np.sum(s**2)/len(s)


    # Variance of Noise Signal
    # SNR = 10 log (E ÷ σ^2)
    var=E/(10**(snr/10))

    # Standard Deviation of the noise Signal
    std=math.sqrt(var)
    # genrating AWGN of samples = samples for r1 and r2
    gaussian_noise=np.random.normal(mu, std, size = len(s))

    # Add this noise to input signal to be noisy
    s_noisy=s+gaussian_noise

    return s_noisy
```

```python
#plot AWGN 🔈🔈
figure, axes = plt.subplots(1, 3, figsize=(15, 5))


figure.suptitle('AWGN')

axes[0].grid()
axes[1].grid()
axes[2].grid()

axes[0].set_title('SNR=-5dB')
axes[0].set_xlabel('phi1')
axes[0].set_ylabel('phi2')

axes[1].set_title('SNR=0dB')
axes[1].set_xlabel('phi1')
axes[1].set_ylabel('phi2')


axes[2].set_title('SNR=10dB')
axes[2].set_xlabel('phi1')
axes[2].set_ylabel('phi2')
```

```python
SNR=[-5,0,10] #Required SNRs to evaluate with in dB :D
index=0
noise_signals_number=100  #No of noise signals to be added to the signals r1 and r2 to see effect of SNR value
for snr in SNR:
    # plot original signal vector space without noise :D

    axes[index].scatter(v_1[0], v_1[1],color='g',marker='o',s=200)
    axes[index].scatter(v_2[0], v_2[1],color='m',marker='o',s=200)

    for j in range(noise_signals_number):
        # for each snr value generate 100 random noises each of samples=samples [same samples no of r1 and r2]
        # Adding AWGN to r1 such that gaussian_noise_r1(t) = r1(t) + w(t);
        gaussian_noise_r1=AWGN(r1,snr)
        # Adding AWGN to r2 such that gaussian_noise_r2(t) = r2(t) + w(t);
        gaussian_noise_r2=AWGN(r2,snr)

        # Project on the space vector
        [si1_1, si2_1]=signal_space(gaussian_noise_r1, phi1, phi2)
        [si1_2, si2_2]=signal_space(gaussian_noise_r2, phi1, phi2)


        # plot this on Scatter diagram
        axes[index].scatter(si1_1,si2_1,color='r')
        axes[index].scatter(si1_2,si2_2,color='b')

    # Legend
    # Create a custom legend for the colors
    legend_elements = [
                    plt.Line2D([0], [0], marker='o', color='w', label='r1',
                            markerfacecolor='m', markersize=10),
                    plt.Line2D([0], [0], marker='o', color='w', label='r2',
                            markerfacecolor='g', markersize=10),
                    plt.Line2D([0], [0], marker='o', color='w', label='r1 + noise',
                            markerfacecolor='r', markersize=10),
                    plt.Line2D([0], [0], marker='o', color='w', label='r2 + noise',
                            markerfacecolor='b', markersize=10)]


    # Add the legend to the plot
    axes[index].legend(handles=legend_elements)
    index+=1

plt.show()
```