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

Assignment 3

Team Members

Num	Full Name in ARABIC	SEC	BN
1	Norhan Reda Abdelwahed Ahmed	2	32
2	Hoda Gamal Hamouda Ismail	2	34

Table of contents:

List of Figures

1. Part One

1.1 Gram-Schmidt Orthogonalization

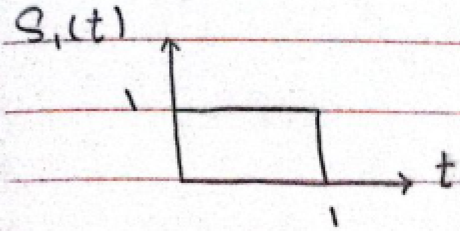
In communication systems, Gram-Schmidt Orthogonalization is used to create a set of orthogonal basis functions for signal transmission. This is done to minimize interference between signals and improve the overall efficiency of the communication system.
By using orthogonal basis functions, the transmitted signals can be easily separated at the receiver, minimizing interference and improving the overall quality of the communication system. This technique is commonly used in wireless communication systems, where multiple signals are transmitted simultaneously over the same frequency band.

Steps to get the orthogonal basis :

```
# Get phi1
   #-------------
   # S1(t) = s11 phi1
   # s11 = sqrt(E1)
   # E1 = integration from 0 to T (S1 ^ 2)
   # phi1 = S1 / s11
```

```
   # Get phi2
   #-------------
   # S2(t) = s21 phi1 + s22 phi2
   # s21 = integration from 0 to T (S2 phi1)
   # g2 = s22 phi2 = S2 - s21 phi1
   # s22 = sqrt(E2)
   # E2 = integration from 0 to T (g2 ^ 2)
   # phi2 = g2 / s22
```

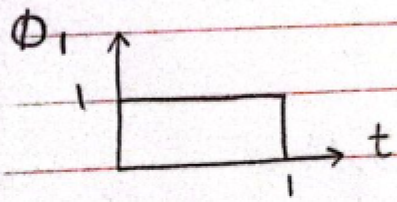
hand written analysis

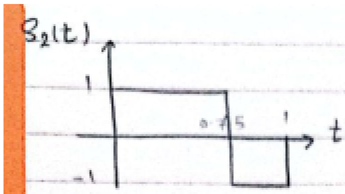


$$E_1 = \int_0^T s_1^2(t)\,dt = \int_0^1 1\,dt = t\Big|_0^1 = 1$$

$$s_{11} = \sqrt{E_1} = 1$$

$$s_1(t) = s_{11}\,\phi_1 \rightarrow \phi_1 = s_1(t) / \underbrace{\cancel{s_{11}}}_{1} = s_1(t)$$



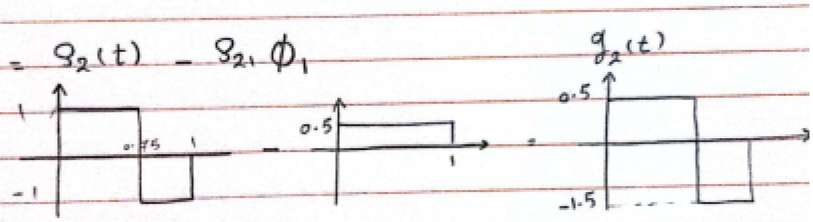


$$s_2(t) = s_{21}\phi_1 + s_{22}\phi_2$$

$$s_{21} = \int_0^T s_2(t)\phi_1\,dt = \int_0^T s_2(t)\,dt =$$

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

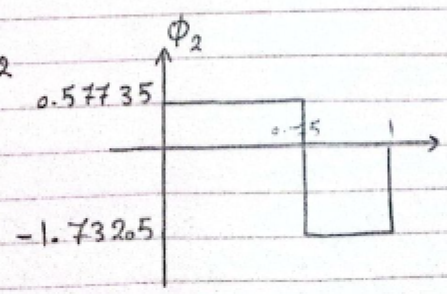
$$g_2 = s_2(t) - s_{21}\phi_1$$



$$s_{22} = \sqrt{\int_0^T g_2^2(t)\,dt} = \sqrt{\int_0^{0.75}(0.5)^2 + \int_{0.75}^{1}(-1.5)^2} =$$

$$\sqrt{0.75} = 0.866025$$

$$\phi_2 = g_2 / s_{22}$$

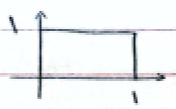


signal space

s_1:

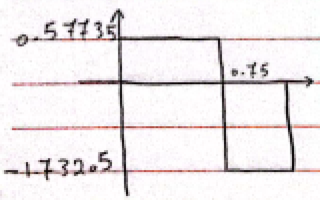
$$s_{ij} = \int_0^T s_i \phi_j \, dt$$

$$V_1 = \int_0^T s_1 \phi_1 \, dt$$



$$V_1 = \int_0^1 1 \, dt = \boxed{1}$$

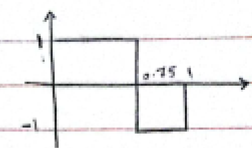
$$V_2 = \int_0^T s_1 \phi_2 \, dt$$



$$V_2 = \int_0^{0.75} 0.57735 + \int_{0.75}^{1} -1.73205 \simeq \boxed{0}$$

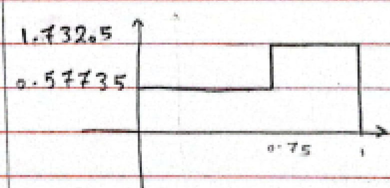
s_2:

$$V_1 = \int_0^T s_2 \phi_1 \, dt$$



$$= \int_0^{0.75} 1 \, dt + \int_{0.75}^{1} -1 = \boxed{0.5}$$

$$V_2 = \int_0^T s_2 \phi_2 \, dt$$



$$V_2 = \int_0^{0.75} 0.57735 \, dt + \int_{0.75}^{1} 1.73205 \, dt =$$

$$\boxed{0.866}$$

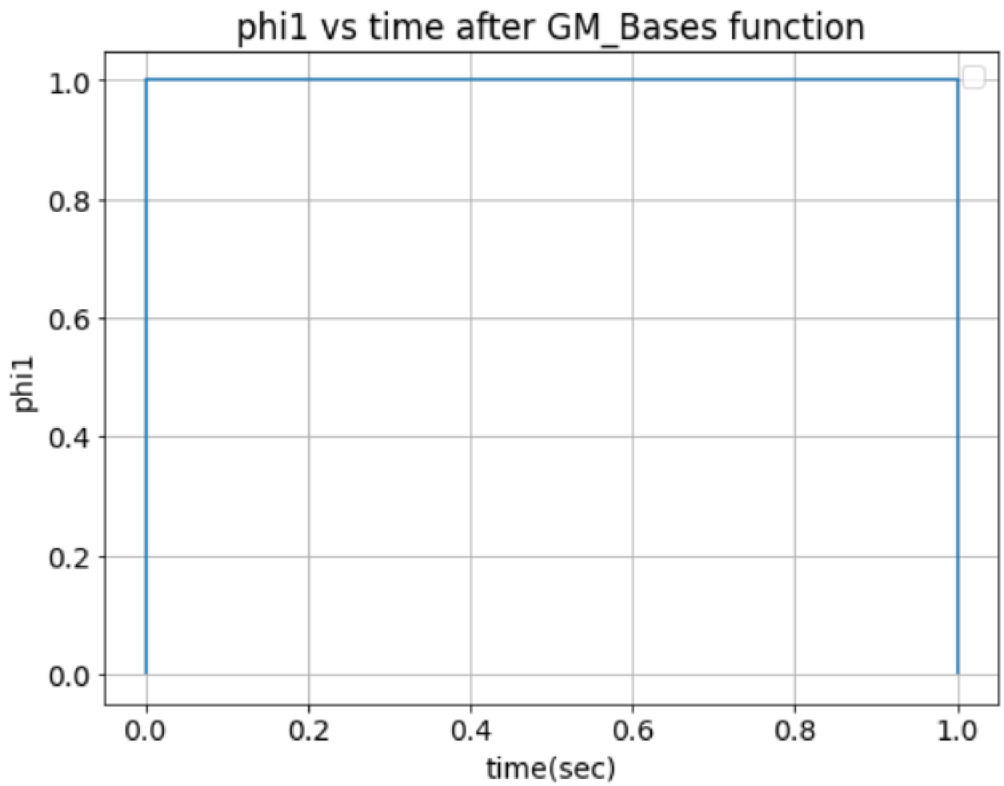


Figure 1 Φ1 VS time after using the GM_Bases function

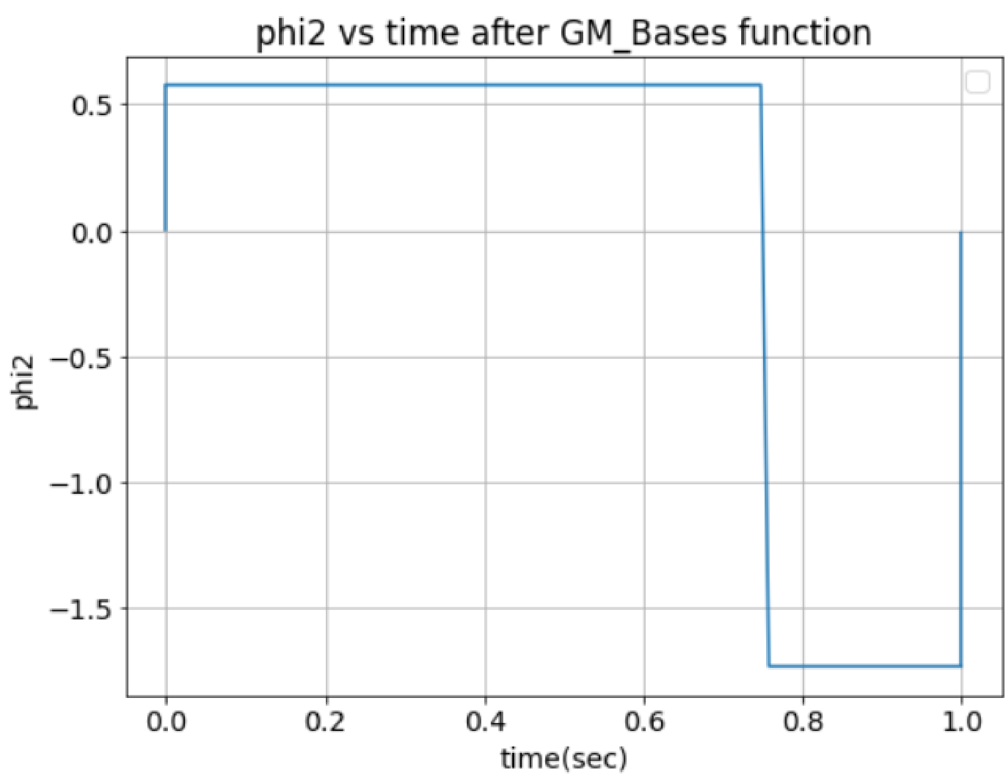


Figure 2 Φ2 VS time after using the GM_Bases function

1.2 Signal Space Representation

Here we represent the signals using the base functions.

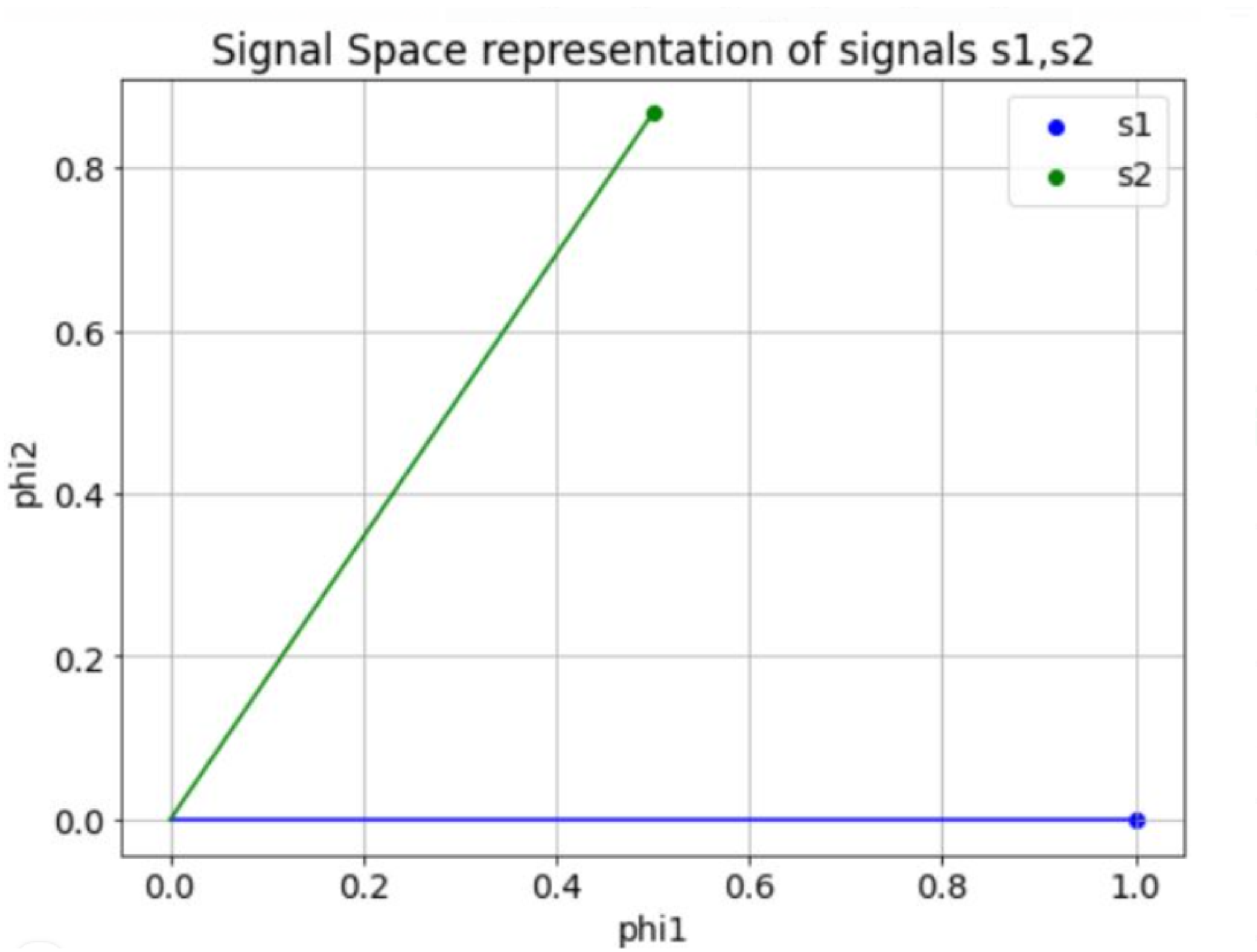


Figure 3 Signal Space representation of signals s1,s2

1.3 Signal Space Representation with adding AWGN

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

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

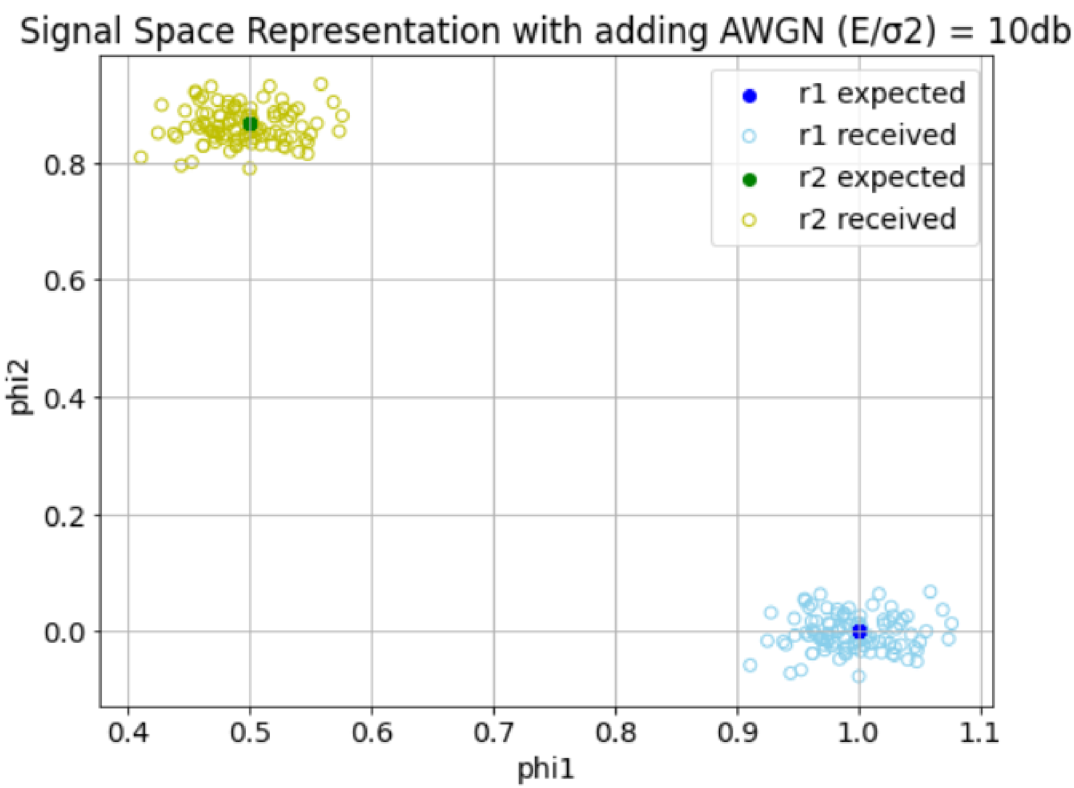


Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

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

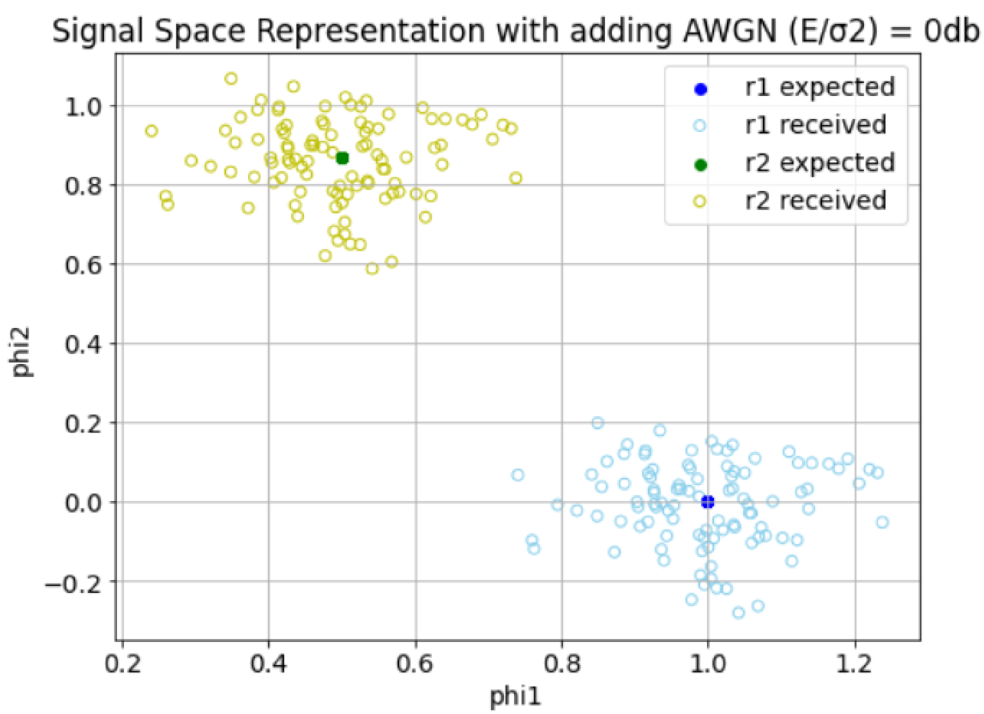


Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

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

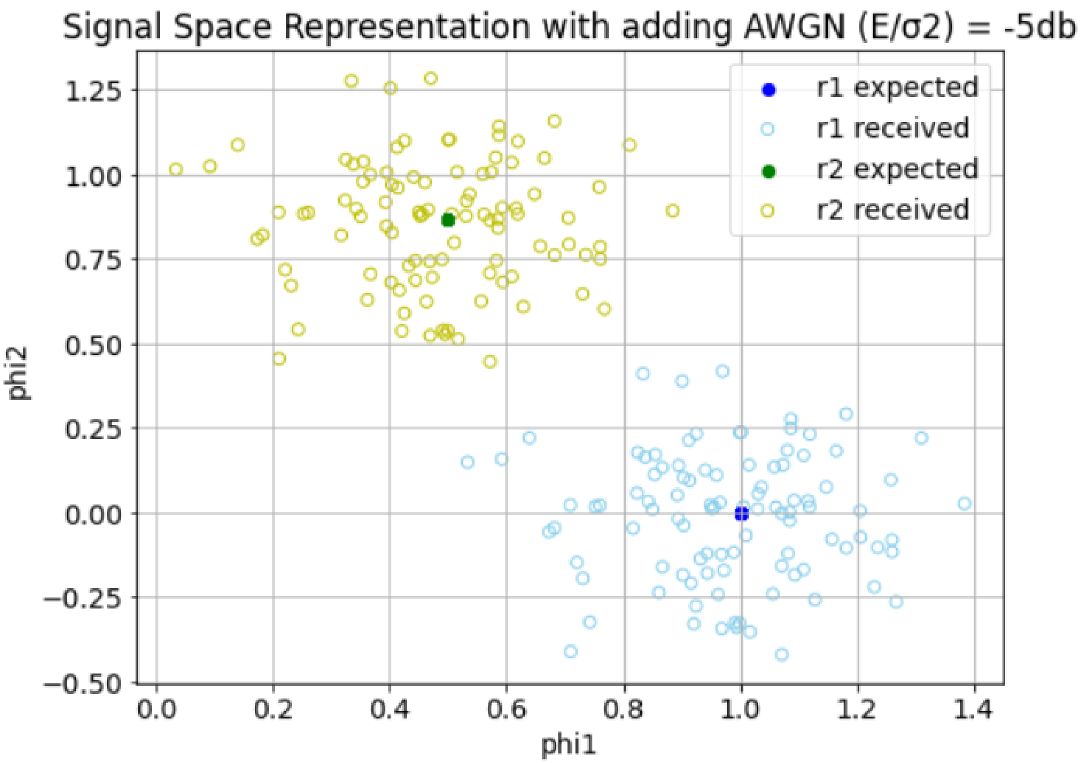


Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB

1.4 Noise Effect on Signal Space

- As σ^2 increase the $E/\ \sigma^2$ decreases so the effect of noise increases and the received points are far from the expected point

- As σ^2 decreases the $E/\ \sigma^2$ increases so the effect of noise decreases and the received points are near from the expected point

2. Appendix A: Codes for Part One:

A.1 Code for Gram-Schmidt Orthogonalization

```
def GM_Bases(s1,s2):

    # Get phi1
    #-------------
    # S1(t) = s11 phi1
    # s11 = sqrt(E1)
    # E1 = integration from 0 to T (S1 ^ 2)
    # phi1 = S1 / s11

    E1 = np.sum(s1 ** 2)/number_of_samples
    print(E1)
    s11 = np.sqrt(E1)
    phi1 = s1 / s11

    # Get phi2
    #-------------
    # S2(t) = s21 phi1 + s22 phi2
    # s21 = integration from 0 to T (S2 phi1)
    # g2 = s22 phi2 = S2 - s21 phi1
    # s22 = sqrt(E2)
    # E2 = integration from 0 to T (g2 ^ 2)
    # phi2 = g2 / s22

    s21 = np.sum(s2 * phi1)/number_of_samples
    g2 = s2 - s21 * phi1
    E2 = np.sum(g2 ** 2)/number_of_samples
    print(E2)
    s22 = np.sqrt(E2)
    phi2 = g2 / s22

    # if s1&s2 have one basis function -> make phi2 zero vector
    if (np.array_equal(phi1,phi2)):
        phi2=np.zeros(number_of_samples)


    return phi1,phi2
```

A.2 Code for Signal Space representation

```
def signal_space(s, phi1, phi2):

    # Sij = integration from 0 to T (Si phij)
    v1 = np.sum(s * phi1)/number_of_samples
    v2 = np.sum(s * phi2)/number_of_samples

    return v1,v2
```

A.3 Code for plotting the bases functions

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

x_axis_time = np.linspace(0,1,number_of_samples)

def plot_GM_Bases(phi,index):
    plt.figure(figsize=(8, 6))
    plt.grid(True)

    plt.plot(x_axis_time,phi)
    plt.vlines(x=0,ymin=0, ymax=phi[0])
    plt.vlines(x=1,ymin=phi[-1], ymax=0)

    plt.rcParams.update({'font.size': 14})
    plt.title('phi' + str(index) +' vs time after GM_Bases function')
    plt.xlabel('time(sec)')
    plt.ylabel('phi'+ str(index))
    plt.legend()

plot_GM_Bases(phi1,1)
plot_GM_Bases(phi2,2)
```

A.4 Code for plotting the Signal space Representations

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

# plot
plt.figure(figsize=(8, 6))
plt.grid(True)

plt.scatter(v11, v21, c='b', label='s1')
plt.scatter(v12, v22, c='g', label='s2')
plt.plot([0, v11], [0, v21], 'b')
plt.plot([0, v12], [0, v22], 'g')

plt.rcParams.update({'font.size': 14})
plt.title('Signal Space representation of signals s1,s2')
plt.xlabel('phi1')
plt.ylabel('phi2')
plt.legend()
```

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

```
E_sig2 = [-5, 0, 10]

for i in range(0, 3):

    plt.figure(figsize=(8, 6))
    plt.grid(True)
    plt.rcParams.update({'font.size': 14})
    plt.title('Signal Space Representation with adding AWGN (E/σ2) = '+ str(E_sig2[i]) +
'db')
    plt.xlabel('phi1')
    plt.ylabel('phi2')

    for j in range(number_of_samples):

        # generate gaussian noise with mean=0,variance=sigma^2 (standard deviation=sigma)
        E_sig = 10 ** ( E_sig2[i] / 10) # convert from db -> E/sigma^2
        E = np.sum(s1 ** 2)/number_of_samples # E is the energy of s1(t)
        sigma = np.sqrt(E/(E_sig)) # variance: sigma^2 = E/(E/sigma^2)
        w = np.random.normal(0 , sigma, number_of_samples) # noise

        # s1
        r1 = s1 + w
        v11_noise, v21_noise = signal_space(r1,phi1,phi2)
        plt.scatter(v11, v21, c='b', label='r1 expected')
        plt.scatter(v11_noise, v21_noise, marker='o',label='r1
received',facecolors='none',edgecolors='skyblue')

        #s2
        r2 = s2 + w
        v12_noise, v22_noise = signal_space(r2,phi1,phi2)
        plt.scatter(v12, v22, c='g', label='r2 expected')
        plt.scatter(v12_noise, v22_noise, marker='o',label='r2
received',facecolors='none',edgecolors='y')


    plt.legend(['r1 expected','r1 received','r2 expected','r2 received'])
```