Cairo University

Faculty of Engineering
Cairo University

# Digital Communication

#ELC3253

## Assignment Three

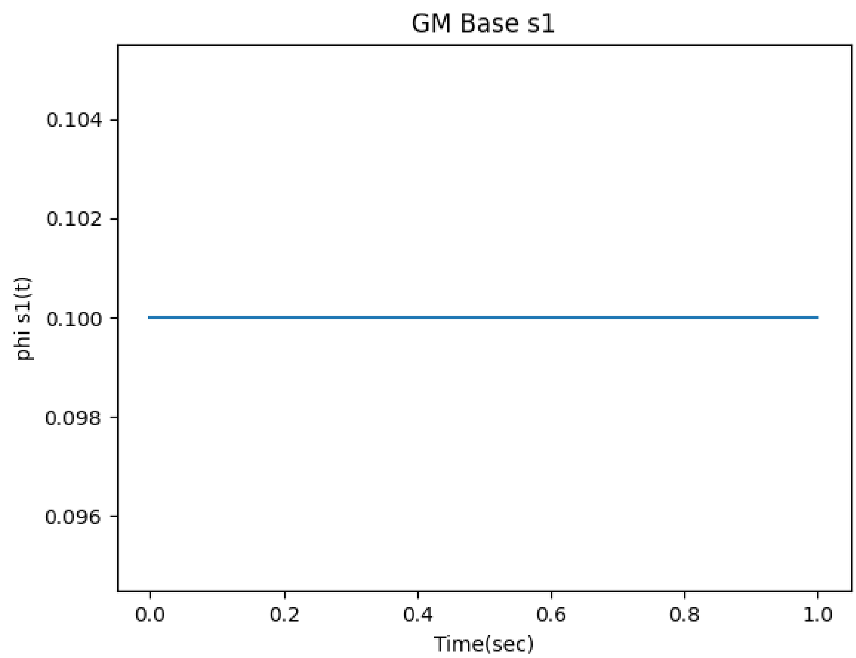
Signal Space

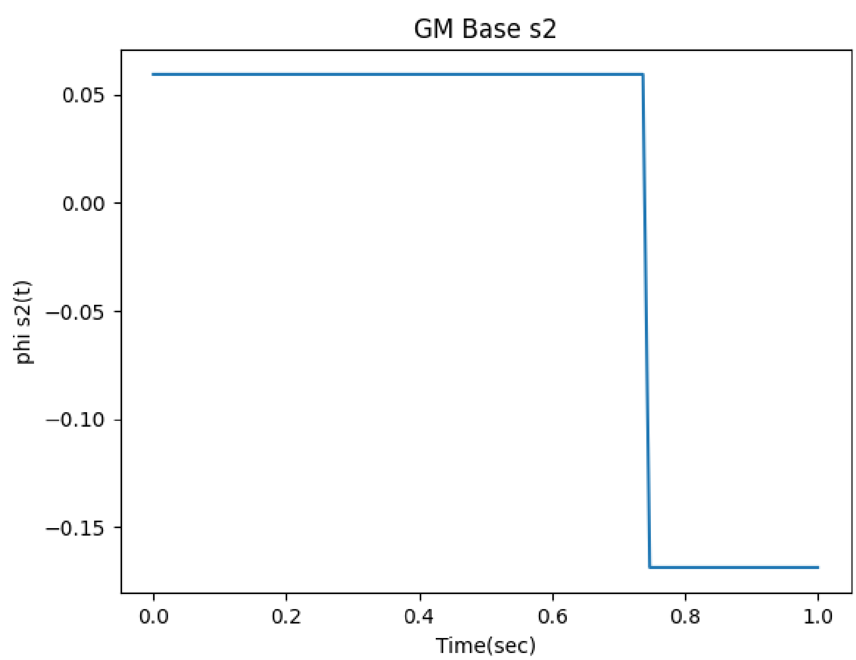
## STUDENTS INFORMATION

| Name | Section | Bench Number |
|---|---|---|
| Yasmine Ashraf Ghanem | 2 | 37 |
| Yasmin Abdullah Nasser | 2 | 38 |

## REQUIREMENTS

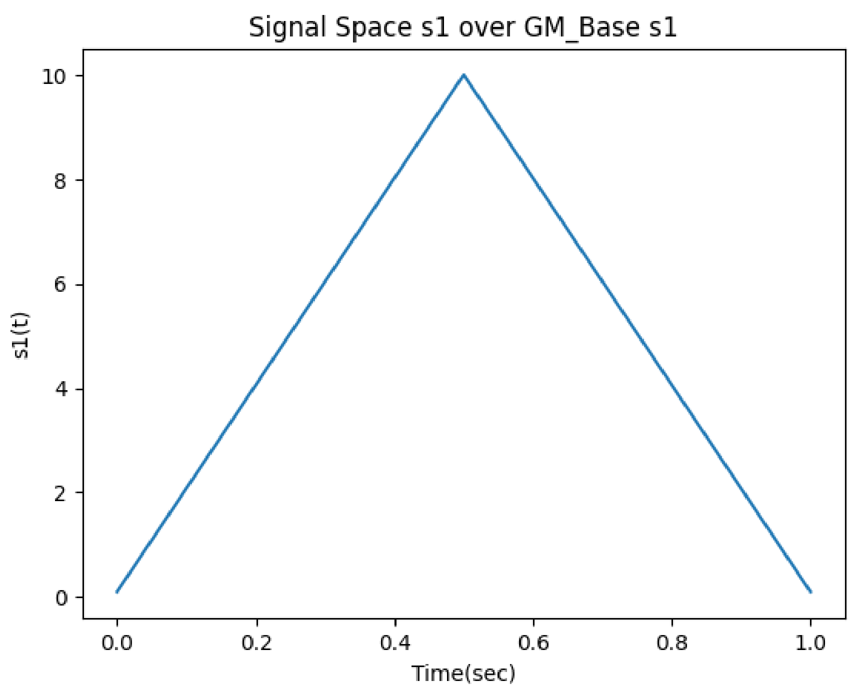
1) **Use your "GM_Bases" function to get the bases functions of s1(t) & s2(t) Figure 1.1. Plot the obtained bases functions.**

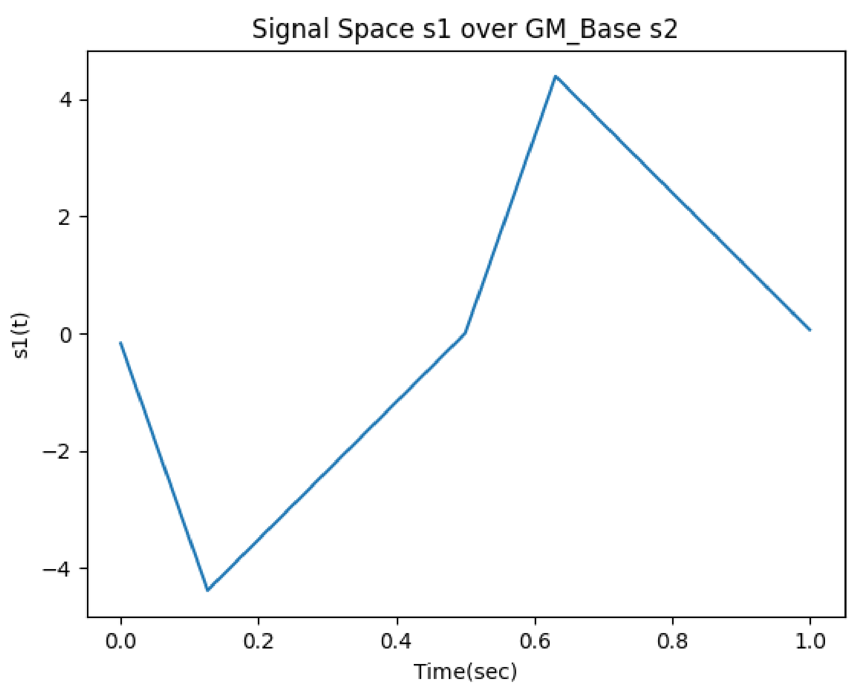


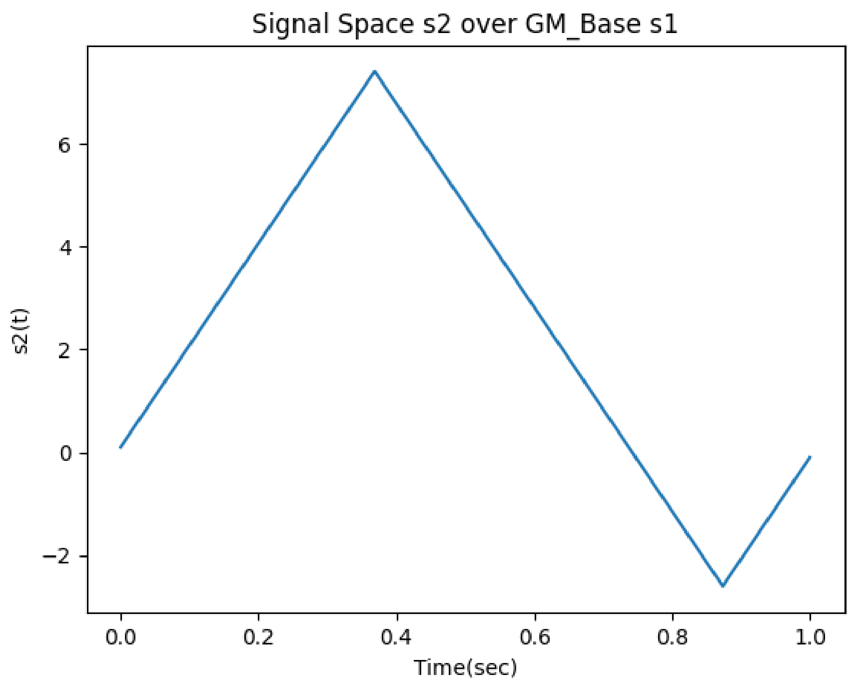
2) **Use your "signal_space" function (along with the bases from 1) to get the signal space representation of s1(t) & s2(t) in Figure 1.1. Plot the signal space representation.**

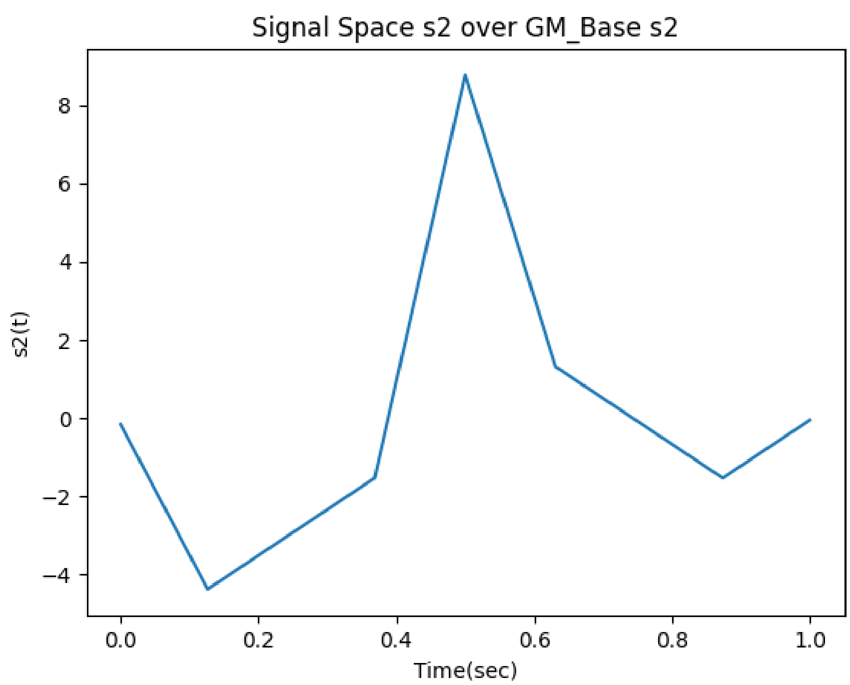
   a) *First Signal Space*



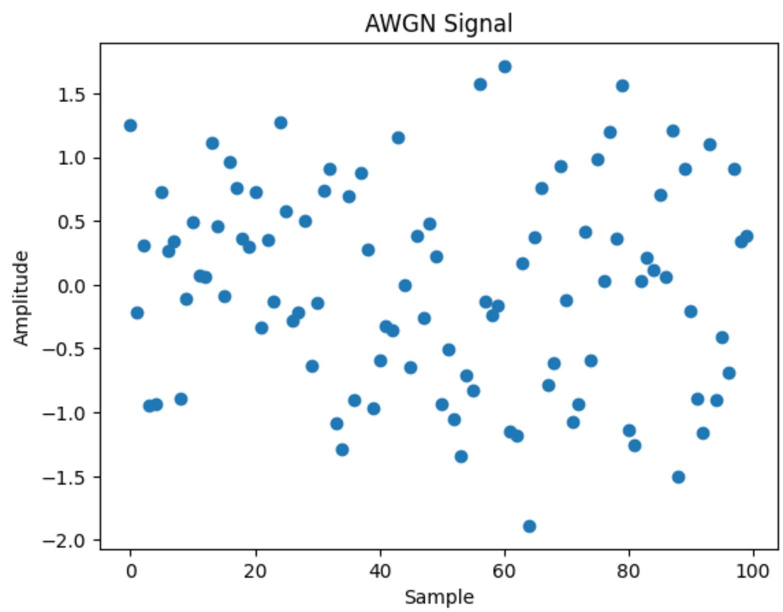
*b) Second Signal Space*

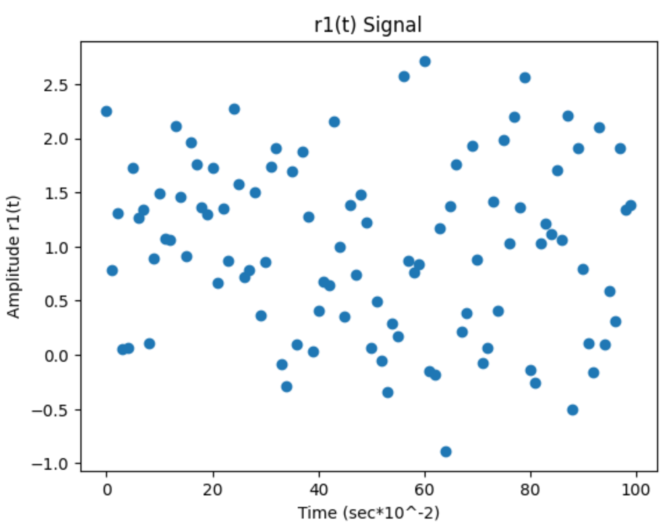


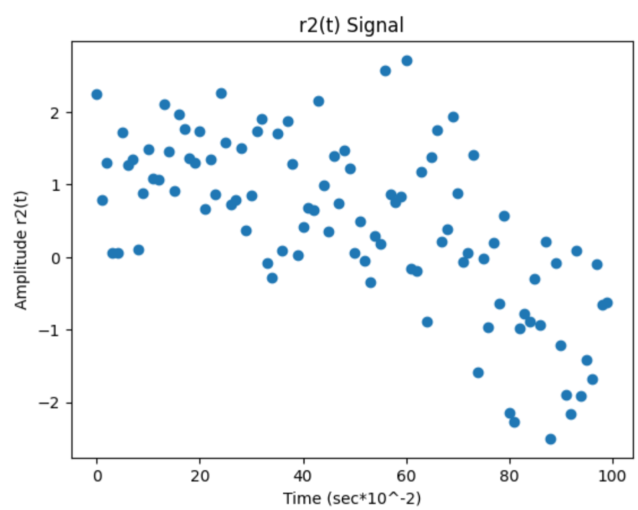



3) **Generate samples of $r1(t)$ and $r2(t)$ using s1(t) & s2(t) and random noise samples. Plot the signal points of the generated samples of $r1(t)$ and $r2(t)$ at E σ2 = −5 dB, 0 dB, 10 dB, where $E$ is the energy of s1 (t) or s2(t).**

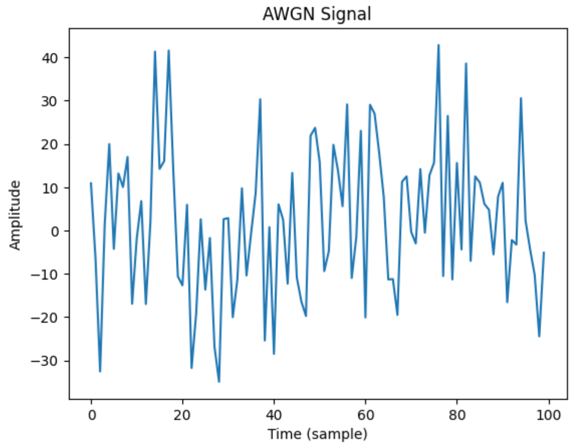
   a) *At variance = 0.5 (Randomly generated at first)*

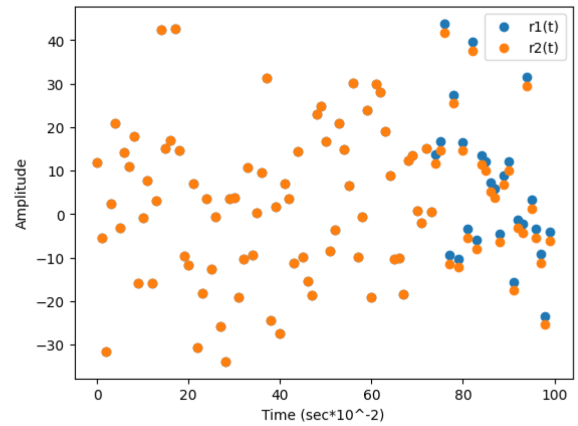


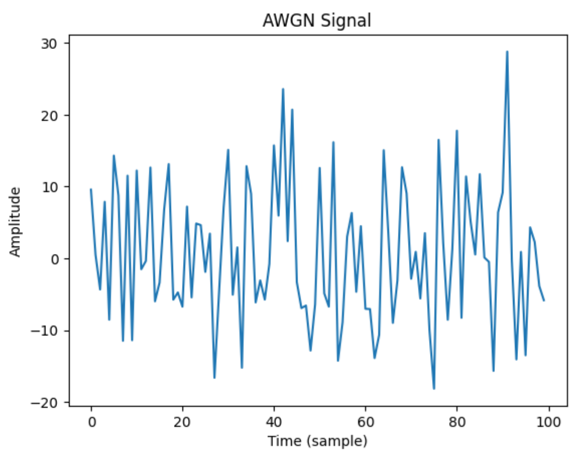



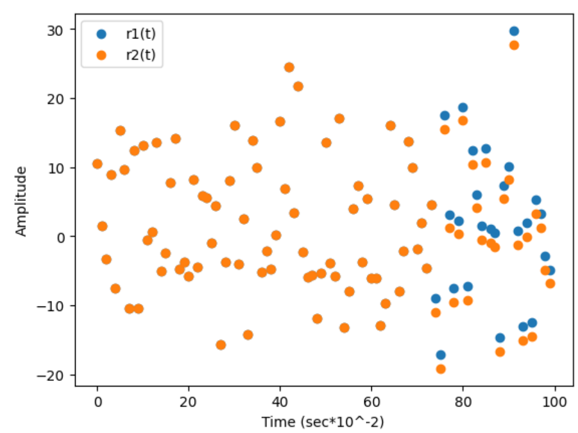
b) *At E/σ^2 = -5*



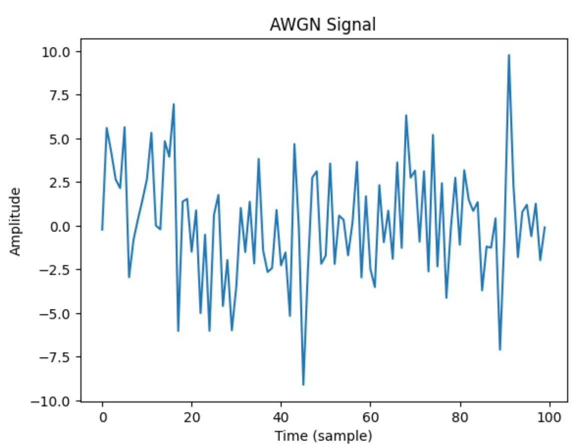



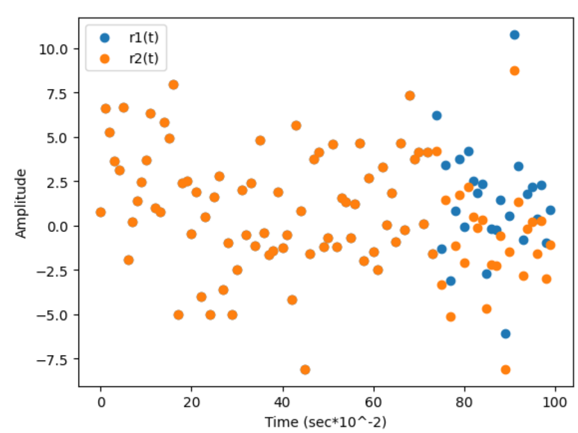

c) *At E/σ^2 = 0*




d) *At E/σ^2 = 10*



**4) How does the noise affect the signal space? Does the noise effect increase or decrease with increasing $\sigma$ 2 ?**

The noise distorts the signal space very much when introduced to it.

As the variance increases the noise effect decreases and the signal tends to go more to its original value. Therefore it’s inversely proportional/correlated.

## CODE

1-“GM_Bases” function

```
def GM_Bases(s1,s2):
    '''
    The function calculates the Gram-Schmidt orthonormal bases functions
(phi1 & phi 2) for two input signals (s1 & s2)
    The inputs s1 and s2: are two 1×N vectors that represent the input
signals
    The outputs phi1 & phi2: are two 1×N vectors that represent the two
orthonormal bases functions (using Gram-Schmidt).
      If s1 & s2 have one basis function, then phi2 is 1×N zero vector
'''

    #We set the first vector same as original & then normalize the rest on
it

    phi1 = s1 / np.linalg.norm(s1)  # Normalize s1 to get phi1

    v2 = s2 - np.dot(s2, phi1) * phi1  # Calculate the orthogonal
component of s2 to phi1
    if np.linalg.norm(v2) > 0:  # Check if v2 is non-zero
        phi2 = v2 / np.linalg.norm(v2)  # Normalize v2 to get phi2
    else:
        phi2 = np.zeros_like(s2)  # phi2 is a zero vector if v2 is zero

    return phi1, phi2
```

2- “signal_space” function

```
def signal_space(s, phi1,phi2):
    '''
    o The function calculates the s1 space representation of input s1 s
        over the orthonormal bases functions (phi1 & phi 2)
    o The inputs s: is a 1×N vectors that represent the input s1
    o The inputs phi1 & phi2: are two 1×N vectors that represent the two
orthonormal bases functions .
```

```
    o The output [v1,v2]: is the projections (i.e. the correlations) of s
over phi1 and phi2 respectively.
    '''
    # Projection of s signal onto phi1 and phi2
    v1 = np.correlate(s, phi1, mode='full')
    v2 = np.correlate(s, phi2, mode='full')

    return v1, v2
```

4- Helper Functions

```
#HELPER FUNCTIONS

# Generate the unit step signals
def unit_step(t):
    return np.heaviside(t, 1)

# Plot signals
def plot_signal(x,y,labelx,labely,title,type='plot'):
    if(type == 'scatter'):
        plt.scatter(x, y)
    else:
        plt.plot(x, y)
    plt.xlabel(labelx)
    plt.ylabel(labely)
    plt.title(title)
    plt.show()
    return

def scatter_signals(x1,y1,x2,y2,labelx,labely,title1,title2):
    # Create figure and axes objects
    __, ax = plt.subplots()

    # Plot Figure 1
    ax.scatter(x1, y1, label = title1)

    # Plot Figure 2
    ax.scatter(x2, y2, label = title2)

    # Customize the plot
```

```
    ax.set_xlabel(labelx)
    ax.set_ylabel(labely)
    ax.legend()

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

4- Generating Signals

```
# Generate s1
tS1 = np.linspace(0, 1, 100)
s1 = unit_step(tS1)
plot_signal(tS1,s1,'Time(sec)','s1(t)','1st Signal')

# Generate s2
tS2_1 = np.linspace(0, 0.74, 74)
tS2_2 = np.linspace(0.75, 1, 26)

s2_a = unit_step(tS2_1)
s2_b = -(unit_step(tS2_2))

s2 = np.concatenate((s2_a,s2_b),axis=0)
tS2 = np.concatenate((tS2_1, tS2_2), axis=0)
plot_signal(tS2,s2,'Time(sec)','s2(t)','2nd Signal')
```

5- Calculating GM Bases

```
GM_Base_s1, GM_Base_s2 = GM_Bases(s1,s2)

T = np.linspace(0,1, 100)

plot_signal(T,GM_Base_s1, 'Time(sec)', 'phi s1(t)', 'GM Base s1')
plot_signal(T,GM_Base_s2, 'Time(sec)', 'phi s2(t)', 'GM Base s2')
```

6- Calculating Signal Spaces

```
GM_Base_s1_a, GM_Base_s1_b = signal_space(s1,GM_Base_s1,GM_Base_s2)
GM_Base_s2_a, GM_Base_s2_b = signal_space(s2,GM_Base_s1,GM_Base_s2)


# print(GM_Base_s1_a)
T = np.linspace(0, 1, 199)
#Signal space of s1 over GM_Base s1 and s2 respectively
plot_signal(T,GM_Base_s1_a, 'Time(sec)', 's1(t)', 'Signal Space s1 over
GM_Base s1')
plot_signal(T,GM_Base_s1_b, 'Time(sec)', 's1(t)', 'Signal Space s1 over
GM_Base s2')

#Signal space of s2 over GM_Base s1 and s2 respectively
plot_signal(T,GM_Base_s2_a, 'Time(sec)', 's2(t)', 'Signal Space s2 over
GM_Base s1')
plot_signal(T,GM_Base_s2_b, 'Time(sec)', 's2(t)', 'Signal Space s2 over
GM_Base s2')
```

7- Generating the r signals with noise and calculating energy of s1

```
#1- Generate the AWGN signal
variance = 0.5
num_samples = 100  #No. of samples
w_t = np.random.normal(loc=0, scale=np.sqrt(variance), size=num_samples)
t = np.arange(num_samples)
plot_signal(t,w_t,'Sample','Amplitude','AWGN Signal','scatter')

# #2- Generate r1(t) & r2(t)
r1 = s1 + w_t
plot_signal(t,r1,'Time (sec*10^-2)','Amplitude r1(t)','r1(t)
Signal','scatter')
r2 = s2 + w_t
plot_signal(t,r2,'Time (sec*10^-2)','Amplitude r2(t)','r2(t)
Signal','scatter')

#3- Get energy of s1
energy = np.sum(s1**2)
```

8- Calculating the effect of AWGN on different values of  E/^2

```
#At E/σ2 = -5 dB
variance = energy/(10**(-5/10))
    #1- Generate the AWGN signal
w_t = np.random.normal(loc=0, scale=np.sqrt(variance), size=num_samples)
t = np.arange(num_samples)
plot_signal(t,w_t,'Time (sample)','Amplitude','AWGN Signal')
    #2- Generate r1(t) & r2(t)
r1 = s1 + w_t
r2 = s2 + w_t
scatter_signals(t,r1,t,r2,'Time (sec*10^-2)','Amplitude','r1(t)','r2(t)')
```

```
#At E/σ2 = 0 dB
variance = energy/(10**(0/10))
    #1- Generate the AWGN signal
w_t = np.random.normal(loc=0, scale=np.sqrt(variance), size=num_samples)
t = np.arange(num_samples)
plot_signal(t,w_t,'Time (sample)','Amplitude','AWGN Signal')
    #2- Generate r1(t) & r2(t)
r1 = s1 + w_t
r2 = s2 + w_t
scatter_signals(t,r1,t,r2,'Time (sec*10^-2)','Amplitude','r1(t)','r2(t)')
```

```
#At E/σ2 = 10 dB
variance = energy/(10**(10/10))
    #1- Generate the AWGN signal
w_t = np.random.normal(loc=0, scale=np.sqrt(variance), size=num_samples)
t = np.arange(num_samples)
plot_signal(t,w_t,'Time (sample)','Amplitude','AWGN Signal')
    #2- Generate r1(t) & r2(t)
r1 = s1 + w_t
r2 = s2 + w_t
scatter_signals(t,r1,t,r2,'Time (sec*10^-2)','Amplitude','r1(t)','r2(t)')
```