**Assignment 3**

**Digital Communication**

**Team Members**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Num** | **Full Name in ARABIC** | SEC | BN | Code |
| 1 | عبدالرحمن حمدي احمد مخيمر | 1 | 36 | 9202833 |
| 2 | زياد طارق خيري علي | 1 | 28 | 9202588 |
| 3 | عبدالرحمن نعمان لقمان ابراهيم | 2 | 2 | 9202851 |

*Submitted to*

**Eng. Mohamed Khaled**

**Table of contents:**

[1. Part One 3](#_Toc135281088)

[1.1 Gram-Schmidt Orthogonalization 3](#_Toc135281089)

[1.2 Signal Space Representation 5](#_Toc135281090)

[1.3 Signal Space Representation with adding AWGN 7](#_Toc135281091)

[1.4 Noise Effect on Signal Space 10](#_Toc135281092)

[2. Appendix A: Codes for Part One: 11](#_Toc135281093)

[A.1 Code for Gram-Schmidt Orthogonalization 11](#_Toc135281094)

[A.2 Code for Signal Space representation 11](#_Toc135281095)

[A.3 Code for plotting the bases functions 12](#_Toc135281096)

[A.4 Code for plotting the Signal space Representations 12](#_Toc135281097)

[A.5 Code for effect of noise on the Signal space Representations 14](#_Toc135281098)

**List of Figures**

[Figure 1 Φ1 VS time after using the GM\_Bases function 4](#_Toc135281077)

[Figure 2 Φ2 VS time after using the GM\_Bases function 4](#_Toc135281078)

[Figure 3 Signal Space representation of signals s1,s2 5](#_Toc135281079)

[Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB 7](#_Toc135281080)

[Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB 8](#_Toc135281081)

[Figure 6 Signal Space representation of signals s1,s2 with E/σ¬2 =-5dB 9](#_Toc135281082)

# Part One

## 1.1 Gram-Schmidt Orthogonalization

* Gram-Schmidt Orthogonalization is a method for taking a set of vectors and finding a new set of orthogonal vectors that span the same space.
* The process involves taking the first vector in the set and normalizing it to create the first orthogonal vector.
* Then, for each subsequent vector, we subtract the projection of that vector onto the previous orthogonal vectors, creating a new vector that is orthogonal to all the previous vectors.
* We normalize this new vector to create the next orthogonal vector, and repeat this process until we have created a set of orthogonal vectors that span the same space.
* The resulting set of orthogonal vectors can be useful for a variety of applications, including solving systems of linear equations, finding eigenvalues and eigenvectors, and performing least-squares approximations.

NUMBER OF SAMPLES = 100

Figure 1 Φ1 VS time after using the GM\_Bases function

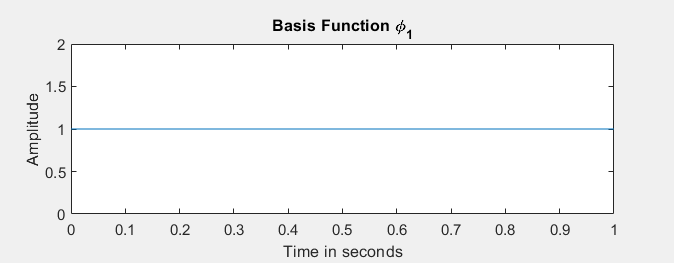
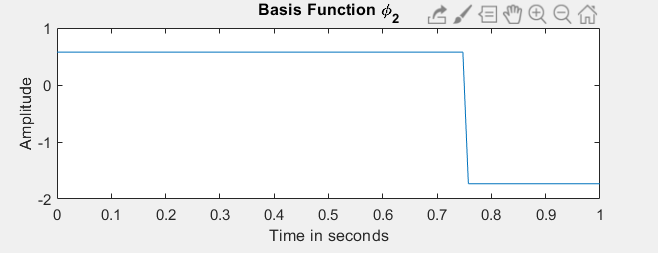


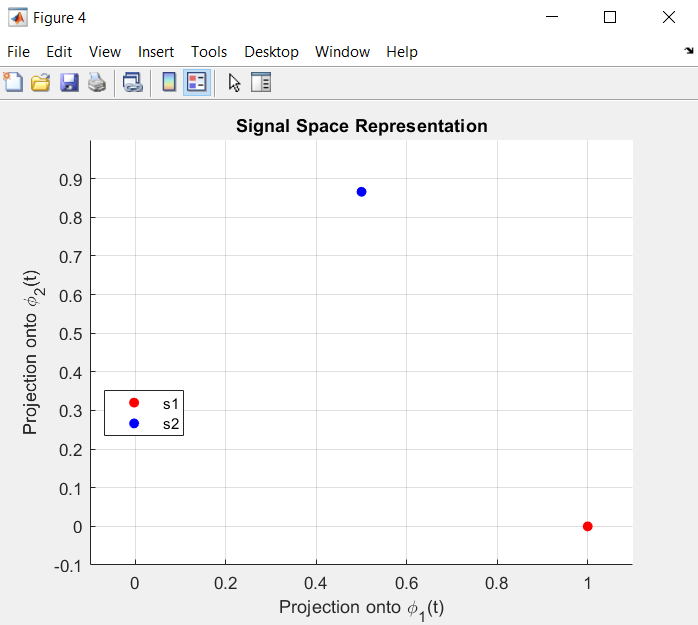
Figure 2 Φ2 VS time after using the GM\_Bases function

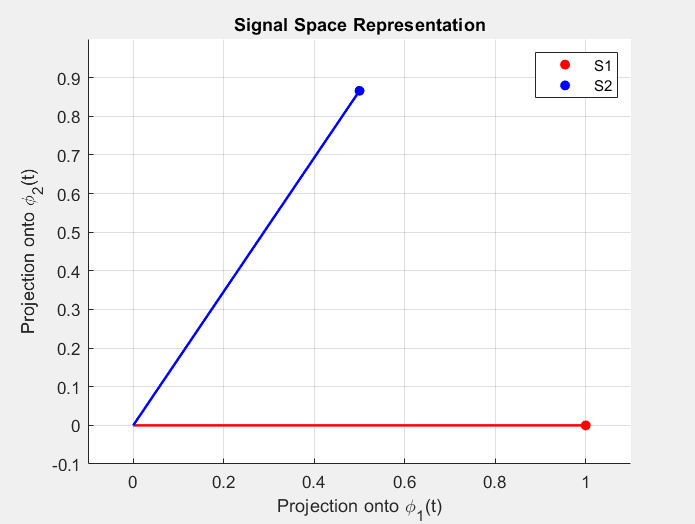


## 1.2 Signal Space Representation

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**:

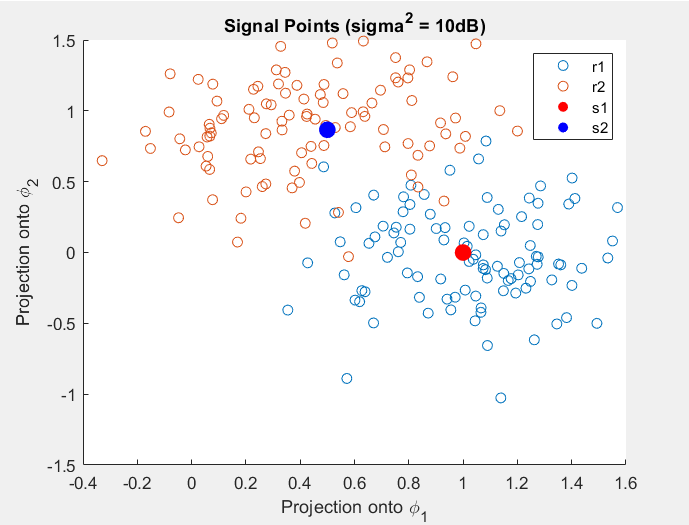


Figure 4 Signal Space representation of signals s1,s2 with E/σ¬2 =10dB

**Case 2**:

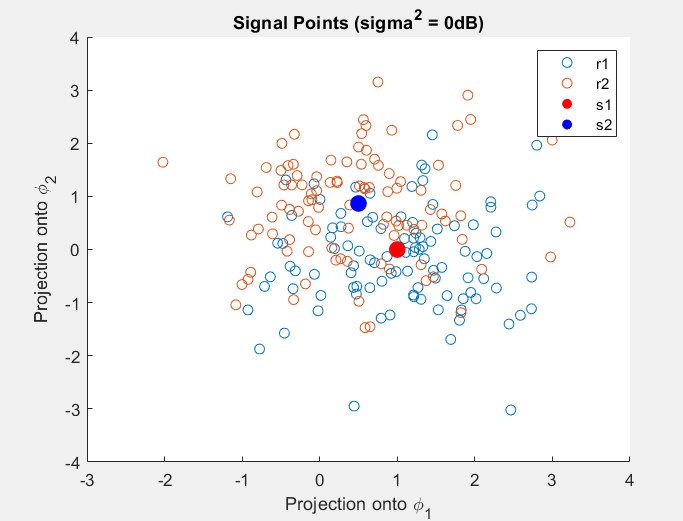


Figure 5 Signal Space representation of signals s1,s2 with E/σ¬2 =0dB

**Case 3**:

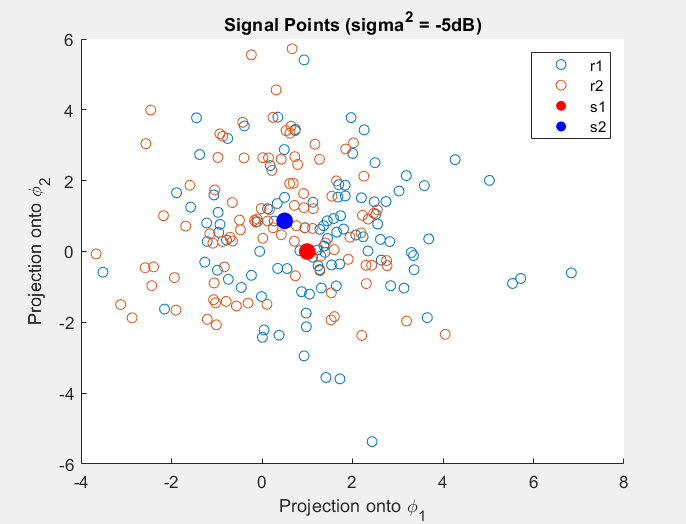
****

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

## 1.4 Noise Effect on Signal Space

The noise affects the signal space representation by introducing variability and spreading out the signal points. As the noise variance σ^2 increases, the effect of noise becomes more prominent and the signal points become more scattered.

Specifically, when the noise variance σ^2 is small, the signal points tend to cluster closely around the ideal signal representation. As the noise variance increases, the signal points become more spread out and exhibit greater dispersion. This dispersion is due to the random nature of the noise, which introduces variations in the observed signal values.

The effect of noise on the signal space representation can be observed in the scatter plots. As the noise level (variance) increases, the signal points become more spread out, leading to a larger variation in the signal space representation. This can be seen as an increase in the dispersion of the signal points in the scatter plots. The noise affects the signal space representation by introducing uncertainty and causing deviations from the ideal signal projections onto the basis functions. Therefore, increasing the noise level (increasing σ 2) generally results in a larger effect of noise on the signal space representation.

# Appendix A: Codes for Part One:

## A.1 Code for Gram-Schmidt Orthogonalization

function [phi1, phi2] = GM\_Bases(s1, s2)

% Calculate the first basis function (phi1)

phi1 = s1 / norm(s1);

% Check if s2 is linearly independent from s1

if dot(s2, phi1) ~= 0

% Calculate the second basis function (phi2)

phi2 = s2 - dot(s2, phi1) \* phi1;

phi2 = phi2 / norm(phi2);

else

% s2 is linearly dependent on s1, so phi2 is a zero vector

phi2 = zeros(size(s2));

end

phi1 = phi1 \* sqrt(length(s1));

phi2 = phi2 \* sqrt(length(s2));

end

## A.2 Code for Signal Space representation

function [v1, v2] = signal\_space(s, phi1, phi2)

% Get the length of each basis function

phi1\_len = length(phi1);

phi2\_len = length(phi2);

% Set new phi1 after dividing by the square root of their lengths

phi1 = phi1 / sqrt(phi1\_len);

phi2 = phi2 / sqrt(phi2\_len);

% Calculating the dot product between the signal and phi just like the

% lecture

dotProduct\_1 = dot(s, phi1);

dotProduct\_2 = dot(s, phi2);

% Calculate the projections (correlations) of s over phi1 and phi2

% Then divide by the square root of each phi

v1 = dotProduct\_1 / sqrt(phi1\_len);

v2 = dotProduct\_2 / sqrt(phi2\_len);

end

## A.3 Code for plotting the bases functions

% Get phi1 & phi2 from the GM\_Bases function

[phi1, phi2] = GM\_Bases(s1, s2);

% Plot phi1

figure;

subplot(2,1,1);

plot(T, phi1);

title('Basis Function \phi\_1');

xlabel('Time in seconds');

ylabel('Amplitude');

% Plot phi2

subplot(2,1,2);

plot(T, phi2);

title('Basis Function \phi\_2');

xlabel('Time in seconds');

ylabel('Amplitude');

## A.4 Code for plotting the Signal space Representations

% Calculate the signal space representation using signal\_space function

% For s1(t)

[s1\_v1, s1\_v2] = signal\_space(s1, phi1, phi2);

% For s2(t)

[s2\_v1, s2\_v2] = signal\_space(s2, phi1, phi2);

% Plot the signal space representation we just obtained

figure;

scatter(s1\_v1, s1\_v2, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on;

scatter(s2\_v1, s2\_v2, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

title('Signal Space Representation');

xlabel('Projection onto \phi\_1(t)');

ylabel('Projection onto \phi\_2(t)');

legend('Location', 'best');

axis([-0.1 1.1 -0.1 1]);

grid on;

% Line Plot

figure;

scatter(s1\_v1, s1\_v2, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on;

scatter(s2\_v1, s2\_v2, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

plot([0, s1\_v1], [0, s1\_v2], 'r', 'LineWidth', 1.5);

plot([0, s2\_v1], [0, s2\_v2], 'b', 'LineWidth', 1.5);

title('Signal Space Representation');

xlabel('Projection onto \phi\_1(t)');

ylabel('Projection onto \phi\_2(t)');

legend('S1', 'S2');

axis([-0.1 1.1 -0.1 1]);

grid on;

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

% Random Samples

% Set the random number generator seed for reproducibility

rng(0);

% Number of samples

numberOfSamples = 100;

% Variance values (dB) for different noise levels

values = [-5, 0, 10];

% Set the length of values

L = length(values);

% Get size of s1 & s2

sz\_s1 = size(s1);

sz\_s2 = size(s2);

% Get length of s1 & s2

len\_s1 = length(s1);

len\_s2 = length(s2);

for i = 1:L

r1V1 = [];

r1V2 = [];

r2V1 = [];

r2V2 = [];

for sample = 1:numberOfSamples

% Convert dB to linear scale

val = 10^(values(i)/10);

sq\_s1 = s1.^2;

sq\_s2 = s2.^2;

energy1 = sum(sq\_s1);

energy2 = sum(sq\_s2);

sigma1 = sqrt(energy1 / val);

sigma2 = sqrt(energy2 / val);

% Add noise to the original signal

w1 = sigma1 \* randn(sz\_s1);

w2 = sigma2 \* randn(sz\_s2);

r1 = w1 + s1;

r2 = w2 + s2;

% Calculate the signal points of the input using signal\_space function

[r1\_v1, r1\_v2] = signal\_space(r1, phi1, phi2);

[r2\_v1, r2\_v2] = signal\_space(r2, phi1, phi2);

% Append to lists

r1V1(end + 1) = r1\_v1;

r1V2(end + 1) = r1\_v2;

r2V1(end + 1) = r2\_v1;

r2V2(end + 1) = r2\_v2;

end

disp(values(i));

% Plot the signal points of generated samples of r1 & r2

figure;

% Plot r1 samples

scatter(r1V1, r1V2, 'DisplayName', 'r1');

hold on;

% Plot r2 samples

scatter(r2V1, r2V2, 'DisplayName', 'r2');

hold on

% Plot the input signal s1(t)

scatter(s1\_v1, s1\_v2, 100, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on

% Plot the input signal s2(t)

scatter(s2\_v1, s2\_v2, 100, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

% Title/xLabel/yLabel

title(sprintf('Signal Points (sigma^2 = %ddB)', values(i)));

xlabel('Projection onto \phi\_1');

ylabel('Projection onto \phi\_2');

legend('r1', 'r2', 's1', 's2');

%axis([-4 5 -1 2]);

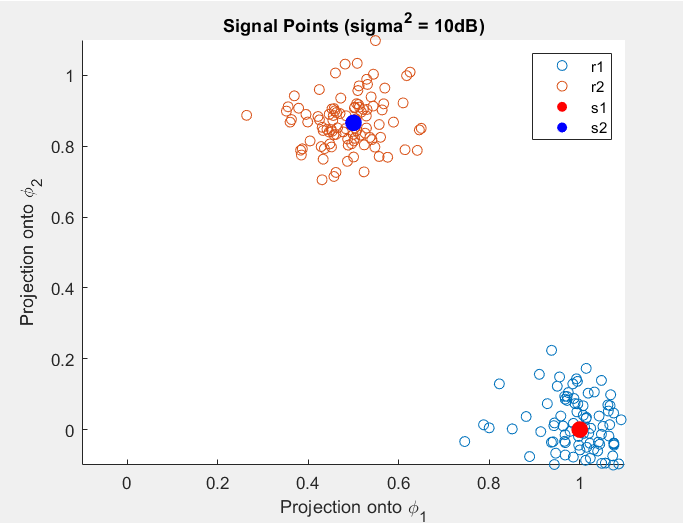
end

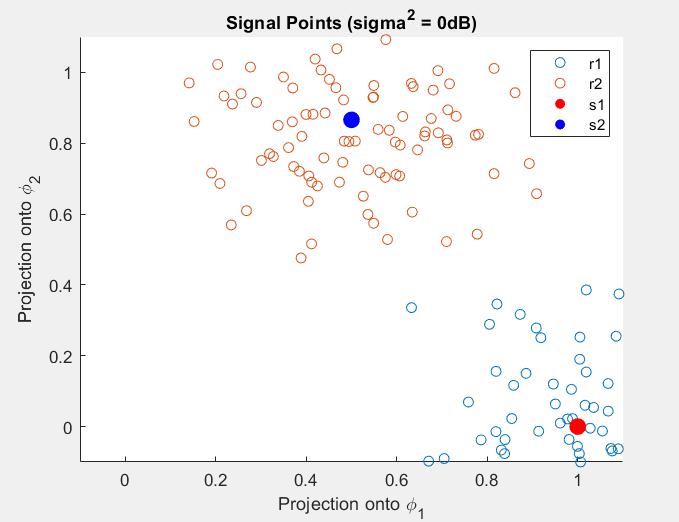
IMPORTANT NOTE:

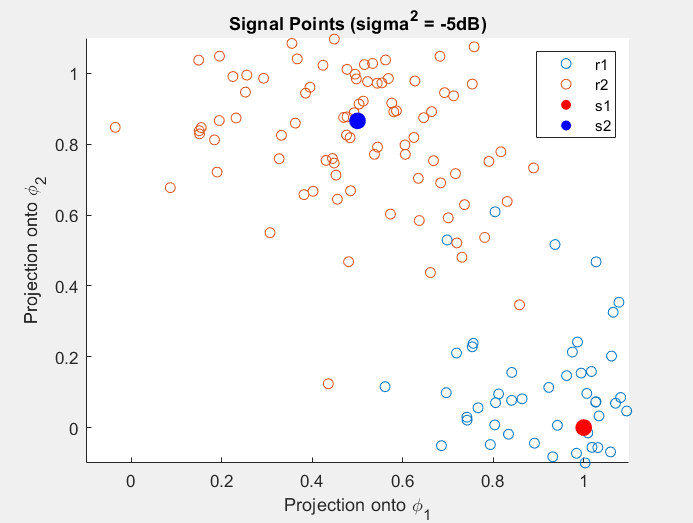
Manually calculating the noise from the equation SNR = E / sigma^2.. We already have the linear SNR value and the **energy** is calculated by summing the square of the signal components, we can easily get sigma. Then, to calculate the noise, it’ll be the sigma multiplied by the random number from 1 to the size of the signal. This noise is then added on to the original signal to obtain r1 and r2. This way gave us the above plots which we submitted.

The above approach was the most logical one but didn’t give us the same graph shapes in the assignment video.

When we attempted to draw the plots the way you did in the tutorial, and using the **awgn** function we got the plots you showed us in the tutorial but there was a problem that when we change the number of samples, the scatter changes and no longer contains the same correlation. Therefore, it’s not considered correct. This gave us the following plots which are exactly like yours:





****

**COMPLETE CODE**

clc;

clear all;

close all;

% Time axis

N = 100;

T = linspace(0, 1, N);

% Input signal s1(t)

s1 = ones(1, N);

% Input signal s2(t)

s2 = zeros(1, N);

% Setting from 0 to 0.75 as 1, else -1

s2(1: floor(0.75 \* N)) = 1;

s2(floor(0.75 \* N) + 1:end) = -1;

% Plot s1(t)

figure;

plot(T, s1);

title('Input Signal s1(t)');

xlabel('Time in seconds');

ylabel('Amplitude');

% Plot s2(t)

figure;

plot(T, s2);

title('Input Signal s2(t)');

xlabel('Time in seconds');

ylabel('Amplitude');

% Get phi1 & phi2 from the GM\_Bases function

[phi1, phi2] = GM\_Bases(s1, s2);

% Plot phi1

figure;

subplot(2,1,1);

plot(T, phi1);

title('Basis Function \phi\_1');

xlabel('Time in seconds');

ylabel('Amplitude');

% Plot phi2

subplot(2,1,2);

plot(T, phi2);

title('Basis Function \phi\_2');

xlabel('Time in seconds');

ylabel('Amplitude');

% Calculate the signal space representation using signal\_space function

% For s1(t)

[s1\_v1, s1\_v2] = signal\_space(s1, phi1, phi2);

% For s2(t)

[s2\_v1, s2\_v2] = signal\_space(s2, phi1, phi2);

% Plot the signal space representation we just obtained

figure;

scatter(s1\_v1, s1\_v2, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on;

scatter(s2\_v1, s2\_v2, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

title('Signal Space Representation');

xlabel('Projection onto \phi\_1(t)');

ylabel('Projection onto \phi\_2(t)');

legend('Location', 'best');

axis([-0.1 1.1 -0.1 1]);

grid on;

% Line Plot

figure;

scatter(s1\_v1, s1\_v2, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on;

scatter(s2\_v1, s2\_v2, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

plot([0, s1\_v1], [0, s1\_v2], 'r', 'LineWidth', 1.5);

plot([0, s2\_v1], [0, s2\_v2], 'b', 'LineWidth', 1.5);

title('Signal Space Representation');

xlabel('Projection onto \phi\_1(t)');

ylabel('Projection onto \phi\_2(t)');

legend('S1', 'S2');

axis([-0.1 1.1 -0.1 1]);

grid on;

% Random Samples

% Set the random number generator seed for reproducibility

rng(0);

% Number of samples

numberOfSamples = 100;

% Variance values (dB) for different noise levels

values = [-5, 0, 10];

% Set the length of values

L = length(values);

% Get size of s1 & s2

sz\_s1 = size(s1);

sz\_s2 = size(s2);

% Get length of s1 & s2

len\_s1 = length(s1);

len\_s2 = length(s2);

for i = 1:L

r1V1 = [];

r1V2 = [];

r2V1 = [];

r2V2 = [];

for sample = 1:numberOfSamples

% Convert dB to linear scale

val = 10^(values(i)/10);

sq\_s1 = s1.^2;

sq\_s2 = s2.^2;

energy1 = sum(sq\_s1);

energy2 = sum(sq\_s2);

sigma1 = sqrt(energy1 / val);

sigma2 = sqrt(energy2 / val);

% Add noise to the original signal

w1 = sigma1 \* randn(sz\_s1);

w2 = sigma2 \* randn(sz\_s2);

r1 = w1 + s1;

r2 = w2 + s2;

% Calculate the signal points of the input using signal\_space function

[r1\_v1, r1\_v2] = signal\_space(r1, phi1, phi2);

[r2\_v1, r2\_v2] = signal\_space(r2, phi1, phi2);

% Append to lists

r1V1(end + 1) = r1\_v1;

r1V2(end + 1) = r1\_v2;

r2V1(end + 1) = r2\_v1;

r2V2(end + 1) = r2\_v2;

end

disp(values(i));

% Plot the signal points of generated samples of r1 & r2

figure;

% Plot r1 samples

scatter(r1V1, r1V2, 'DisplayName', 'r1');

hold on;

% Plot r2 samples

scatter(r2V1, r2V2, 'DisplayName', 'r2');

hold on

% Plot the input signal s1(t)

scatter(s1\_v1, s1\_v2, 100, 'filled', 'MarkerFaceColor', 'red', 'DisplayName', 's1');

hold on

% Plot the input signal s2(t)

scatter(s2\_v1, s2\_v2, 100, 'filled', 'MarkerFaceColor', 'blue', 'DisplayName', 's2');

% Title/xLabel/yLabel

title(sprintf('Signal Points (sigma^2 = %ddB)', values(i)));

xlabel('Projection onto \phi\_1');

ylabel('Projection onto \phi\_2');

legend('r1', 'r2', 's1', 's2');

%axis([-4 5 -1 2]);

end

function [phi1, phi2] = GM\_Bases(s1, s2)

% Calculate the first basis function (phi1)

phi1 = s1 / norm(s1);

% Check if s2 is linearly independent from s1

if dot(s2, phi1) ~= 0

% Calculate the second basis function (phi2)

phi2 = s2 - dot(s2, phi1) \* phi1;

phi2 = phi2 / norm(phi2);

else

% s2 is linearly dependent on s1, so phi2 is a zero vector

phi2 = zeros(size(s2));

end

phi1 = phi1 \* sqrt(length(s1));

phi2 = phi2 \* sqrt(length(s2));

end

function [v1, v2] = signal\_space(s, phi1, phi2)

% Get the length of each basis function

phi1\_len = length(phi1);

phi2\_len = length(phi2);

% Set new phi1 after dividing by the square root of their lengths

phi1 = phi1 / sqrt(phi1\_len);

phi2 = phi2 / sqrt(phi2\_len);

% Calculating the dot product between the signal and phi just like the

% lecture

dotProduct\_1 = dot(s, phi1);

dotProduct\_2 = dot(s, phi2);

% Calculate the projections (correlations) of s over phi1 and phi2

% Then divide by the square root of each phi

v1 = dotProduct\_1 / sqrt(phi1\_len);

v2 = dotProduct\_2 / sqrt(phi2\_len);

end