Cairo University- Faculty of Engineering
Electronics and Communications Engineering Department
Communications (ELC 306B) – Spring 2020

Final Project

Team Members

Num	**Full Name in ARABIC**	SEC	BN	Role in the project
1	احمد محمد قرني الشويمي	1	29	Part 1
2	محمد عادل جودة	3	49	Part 2
3	محمد عنتر جاد الكريم	3	56	Part 2
4	يحيي زكريا علي زكي	4	58	Part 3

Table of contents:

List of Figures

1. PartOne

1.1 Bases Signals

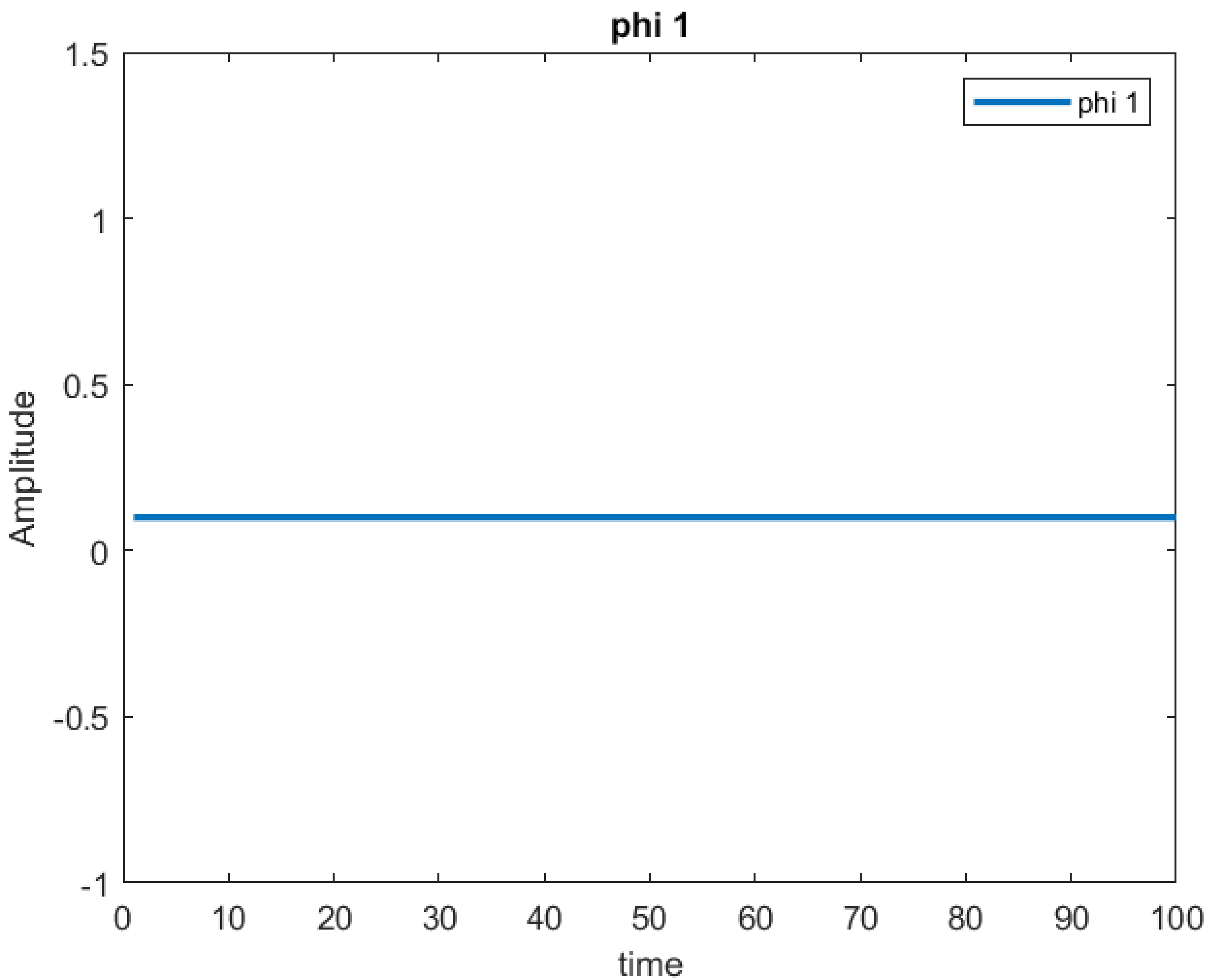

Figure 1: phi1

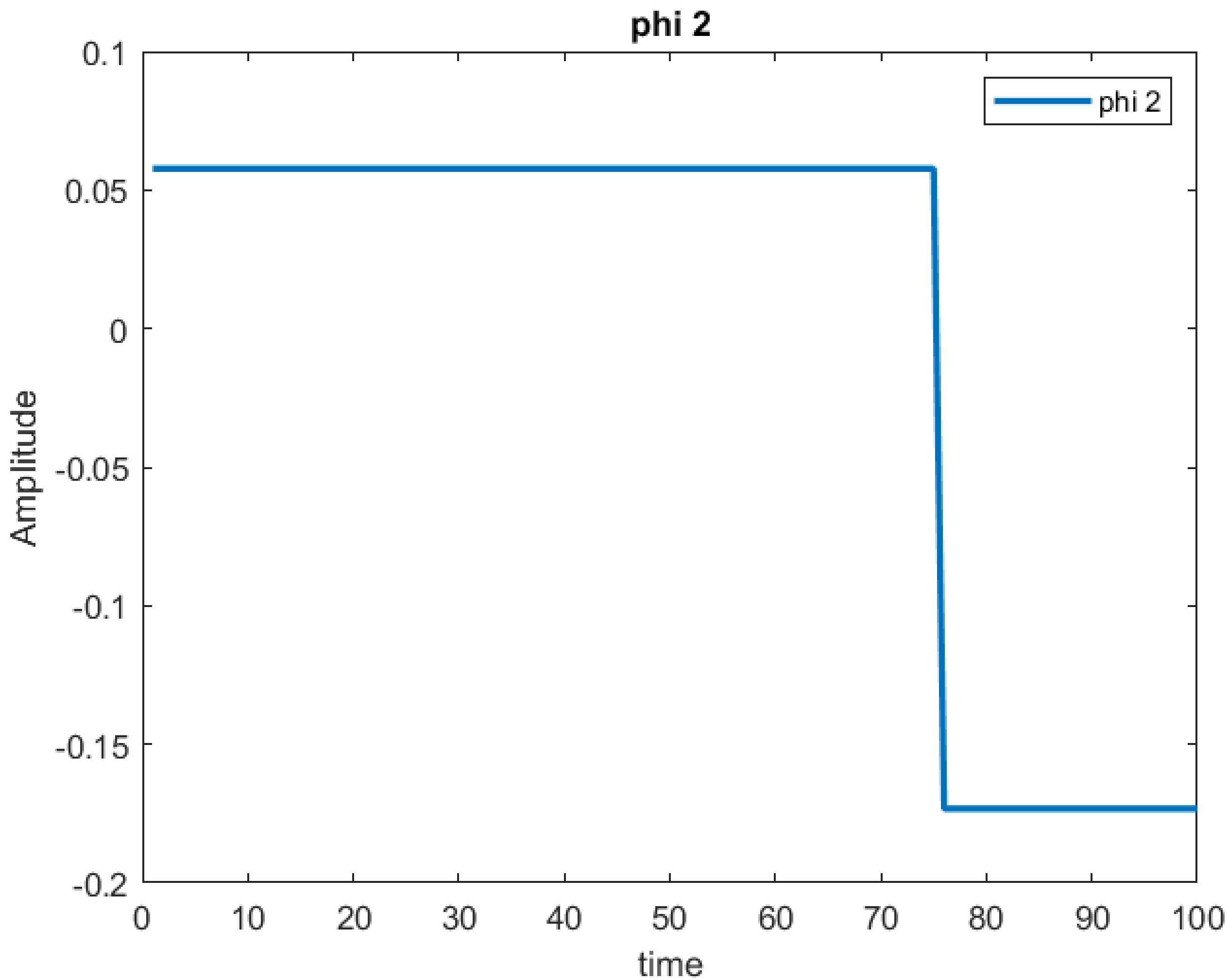

Figure 2 phi2

1.2 Signal space representation

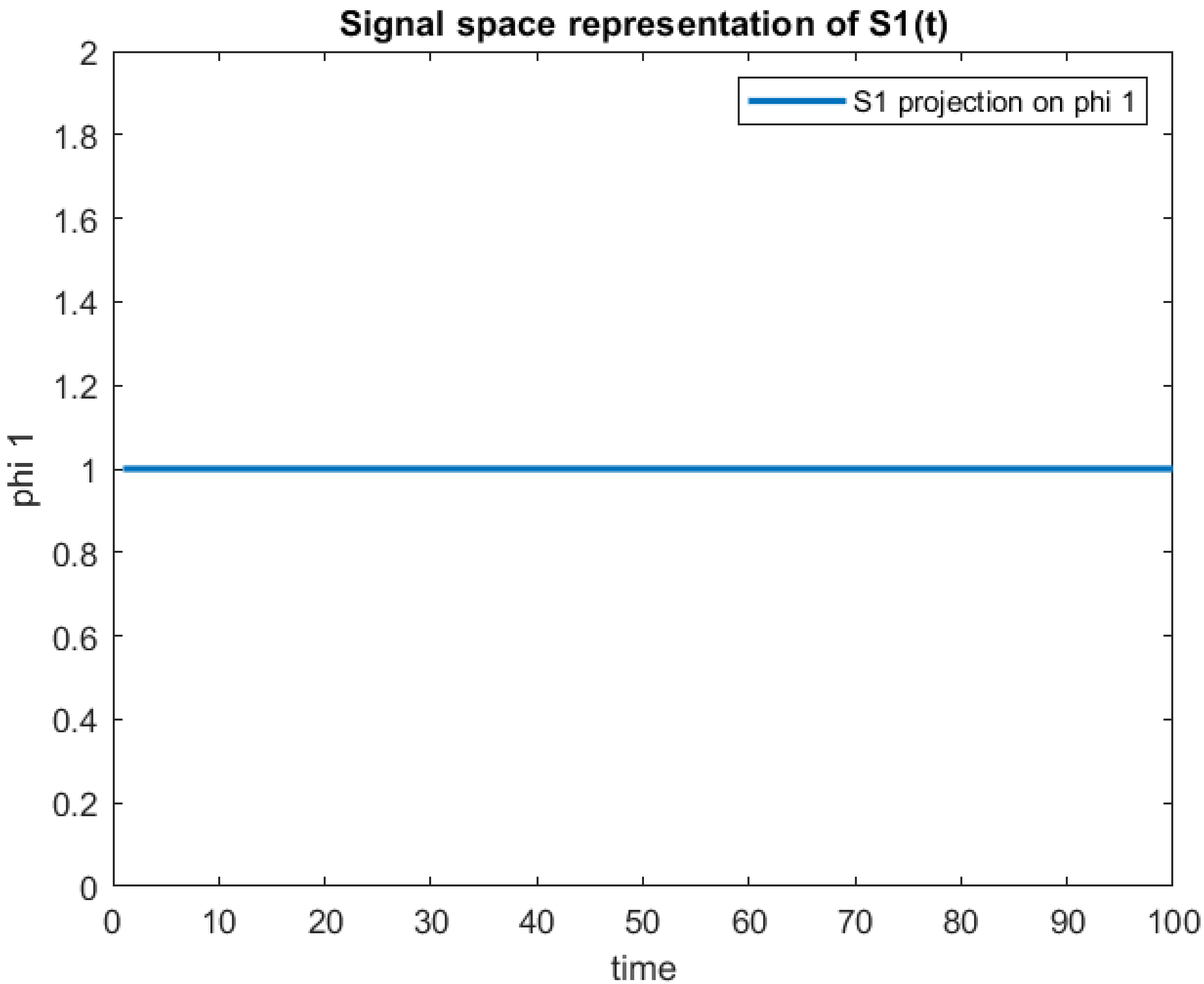

Figure 3 s1 representation on phi1

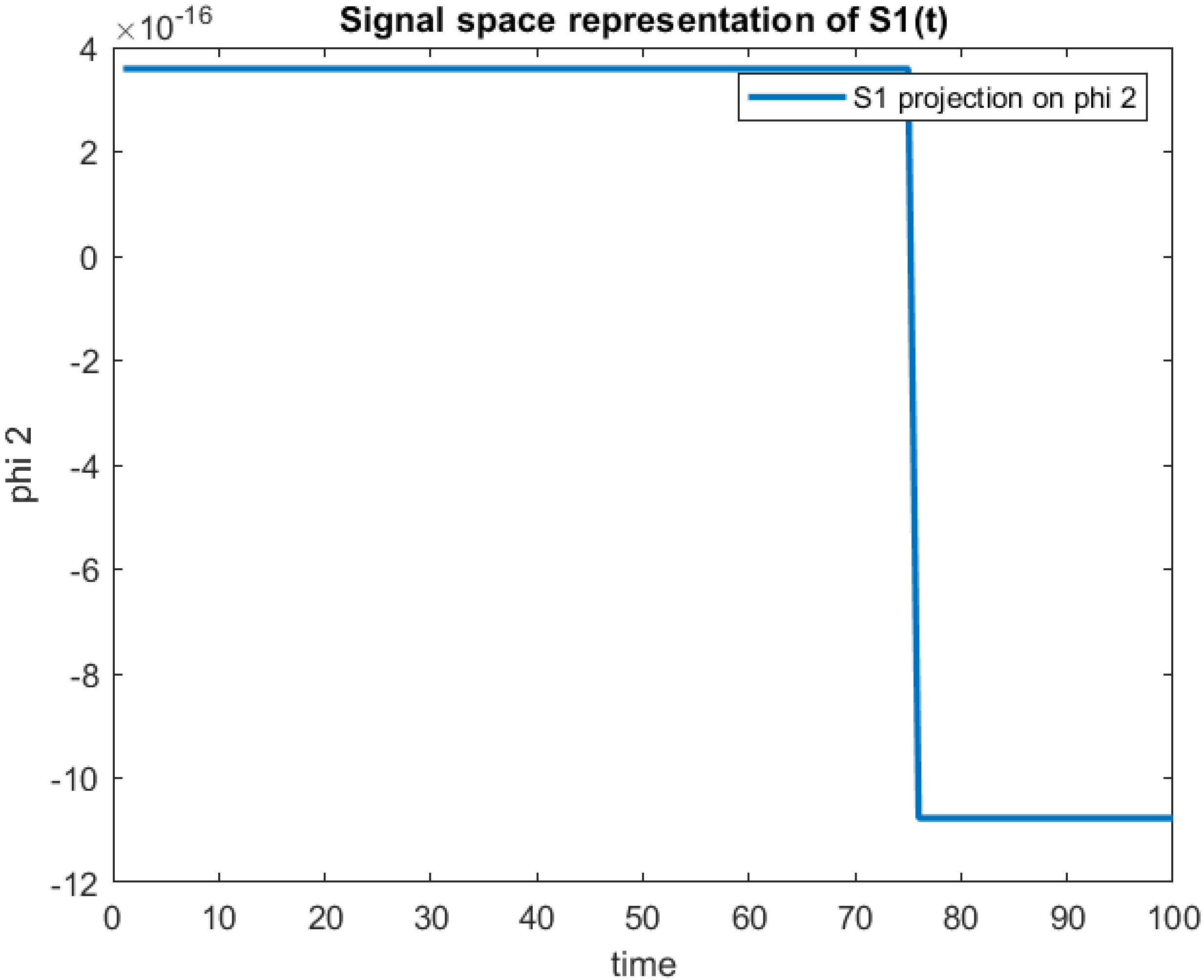

Figure 4 s1 representation on phi2

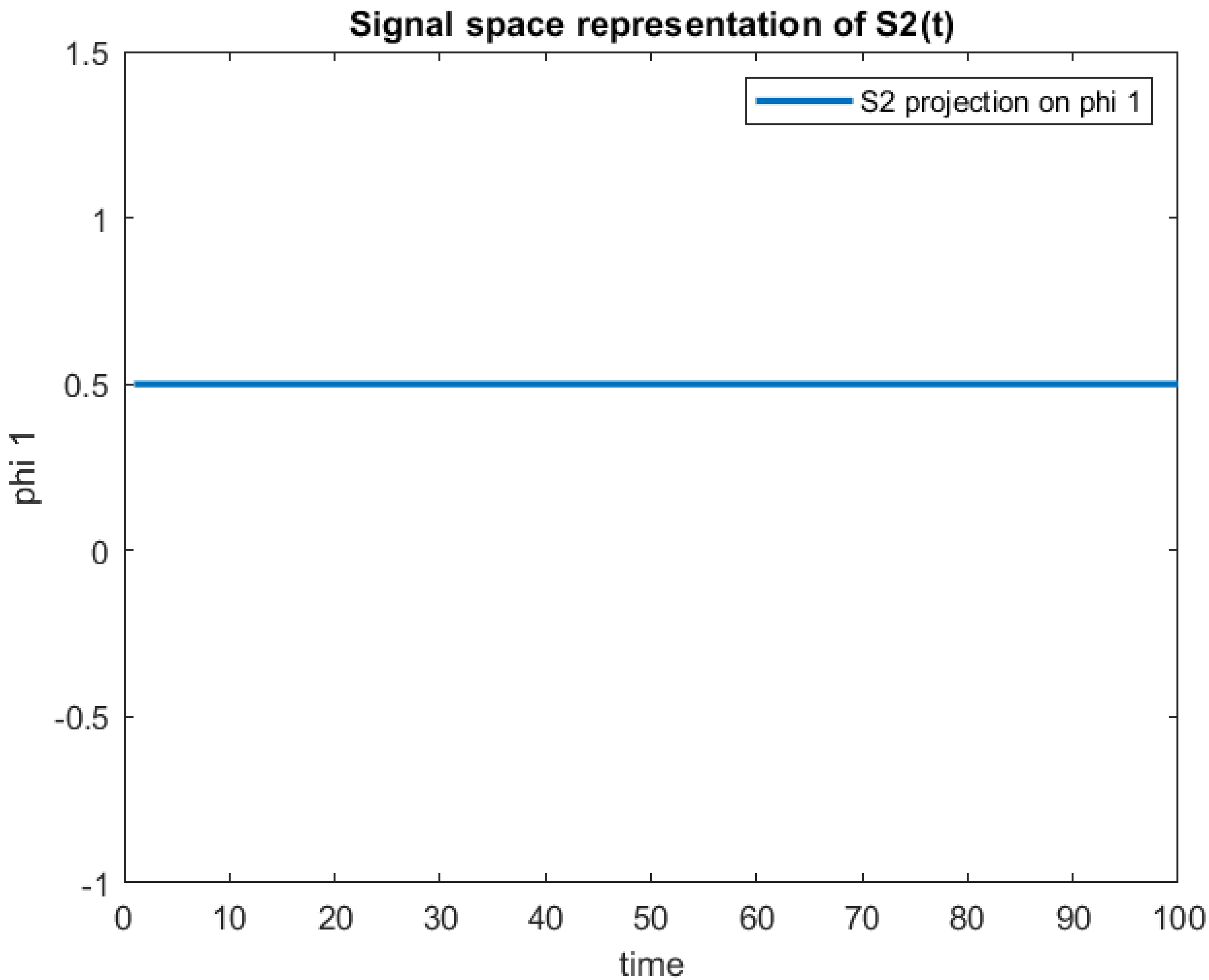

Figure 5 : s2 representation on phi1

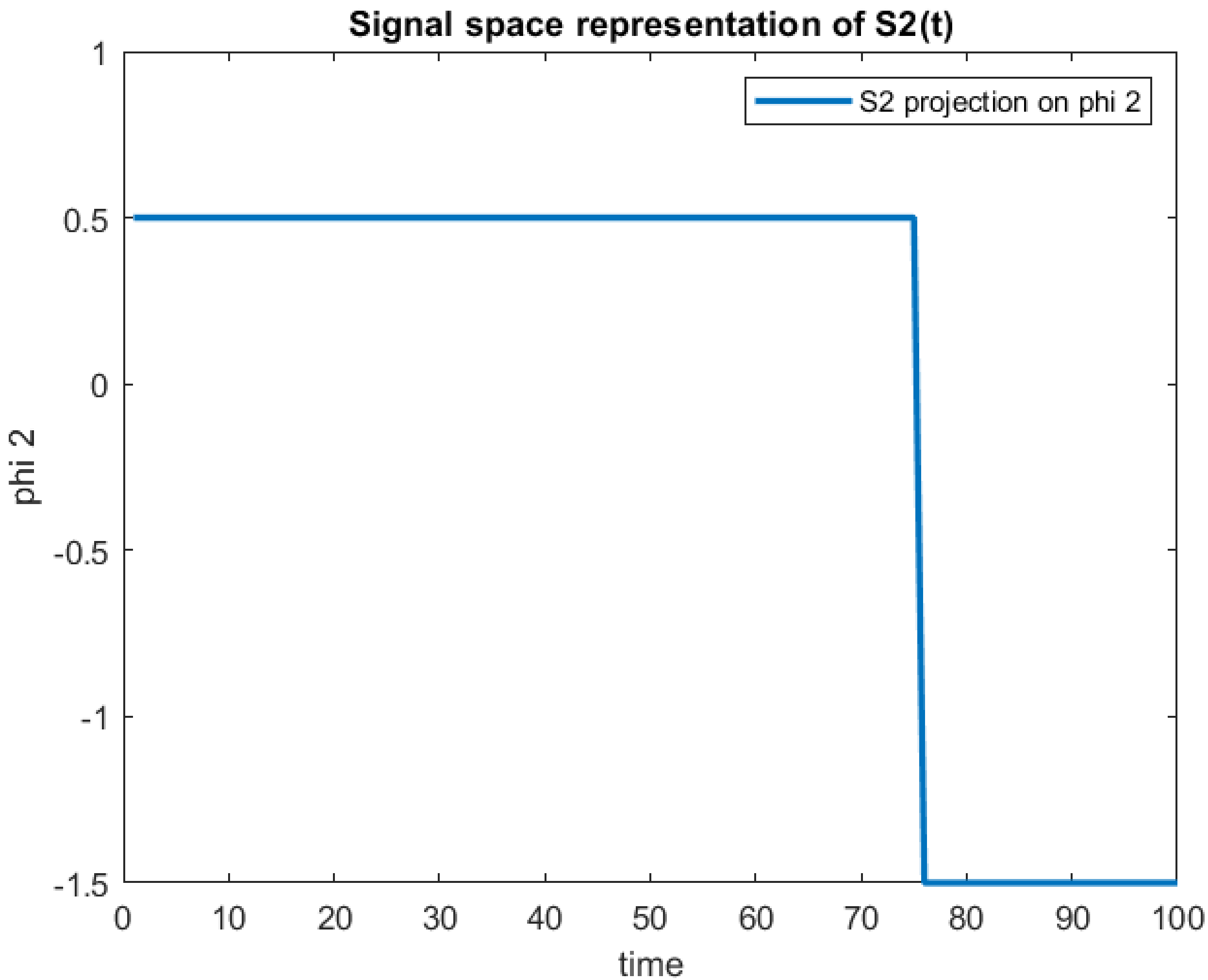

Figure 6 s2 representation on phi2

1.3 Noise effect

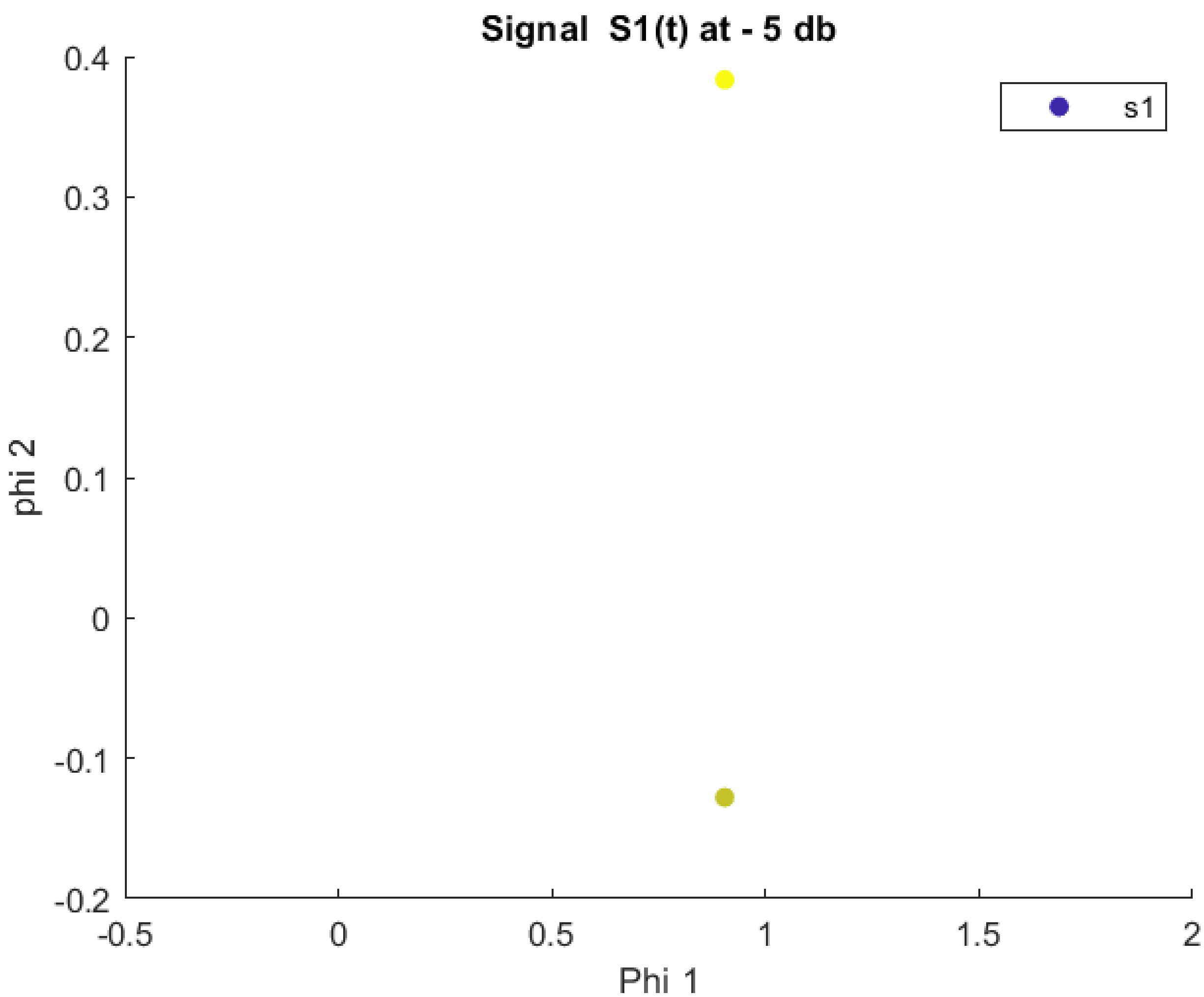

Figure 7 (s1 +noise) at -5db

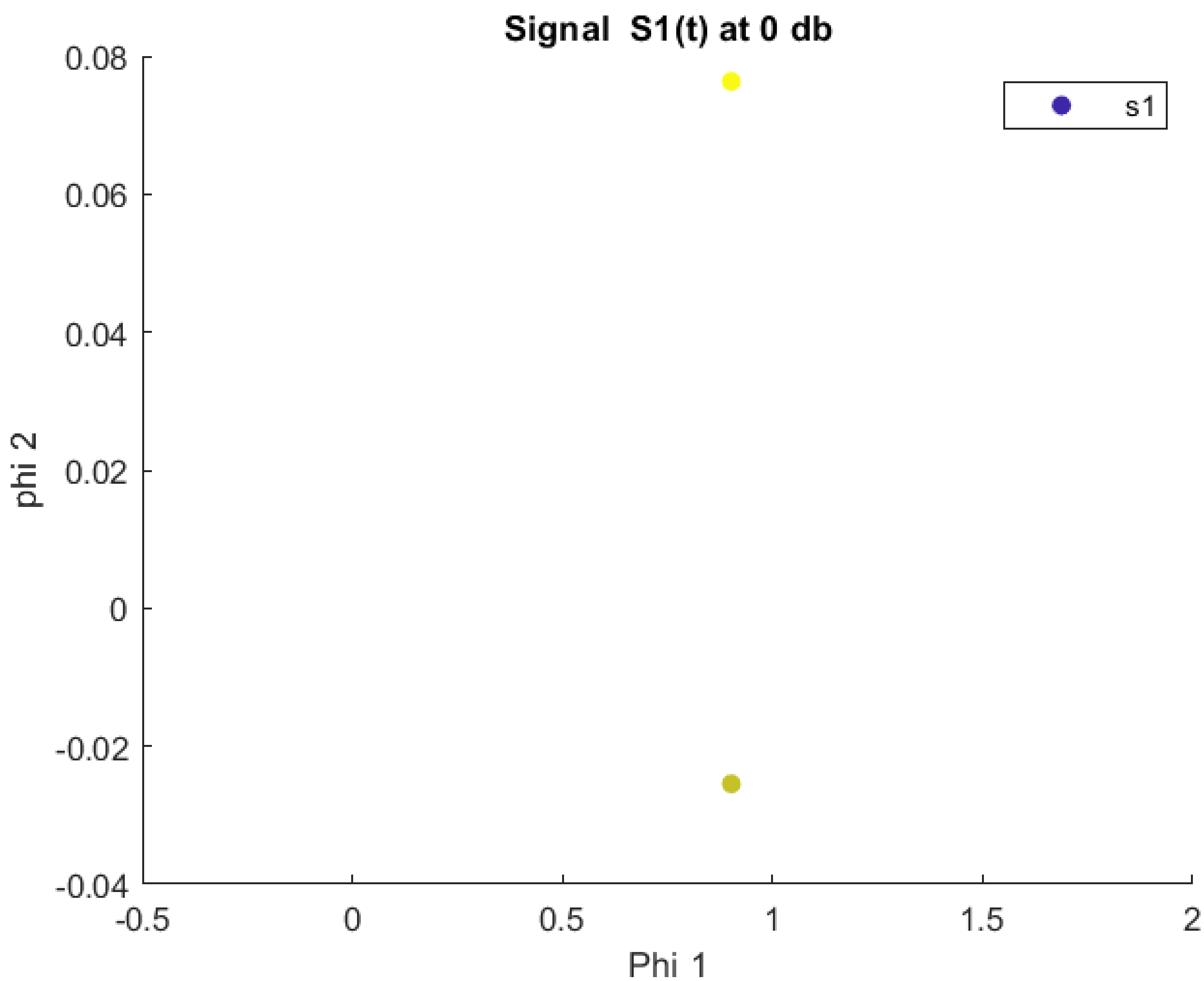

Figure 8 : (s1 +noise) at 0db

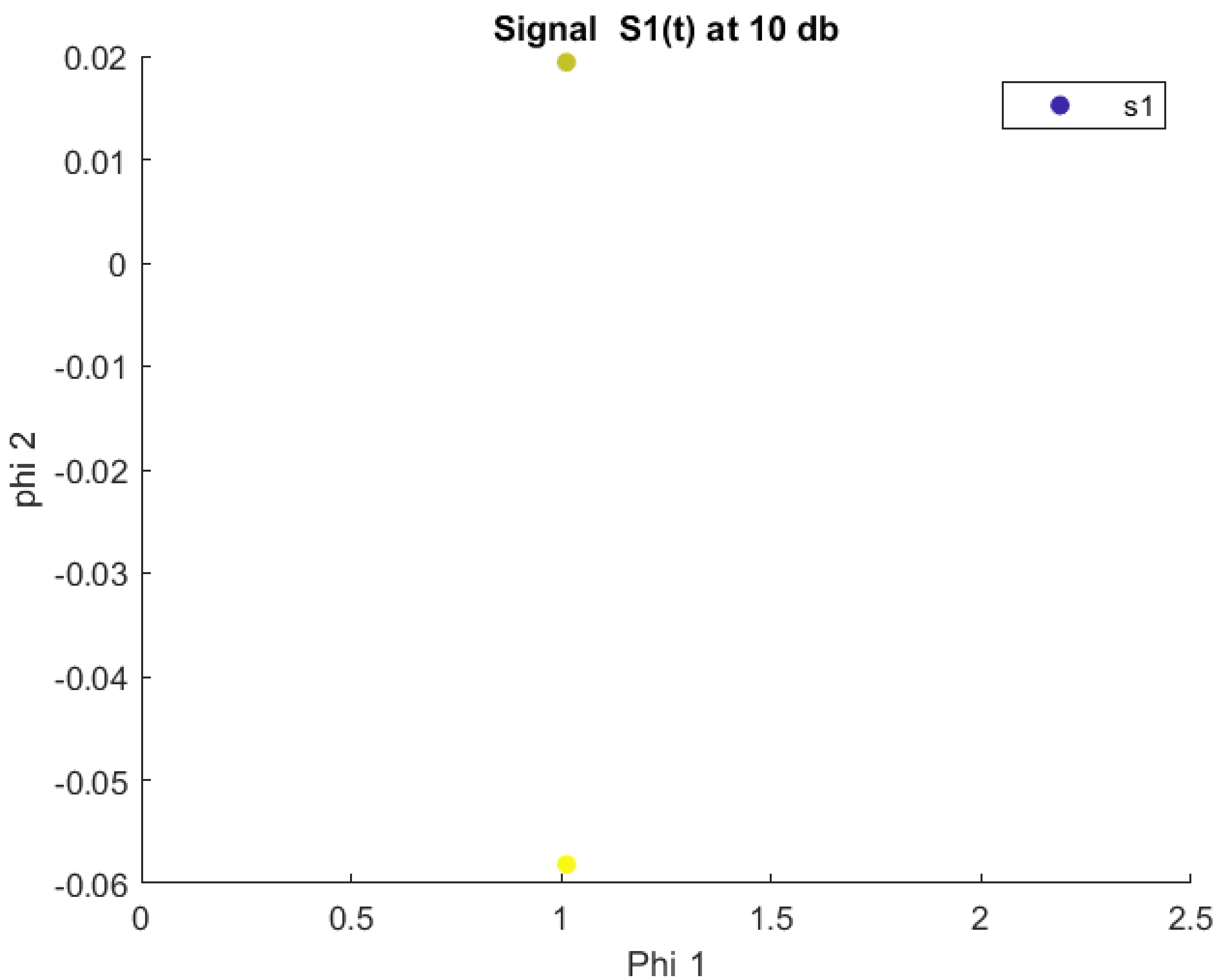

Figure 9 (s1 +noise) at 10db

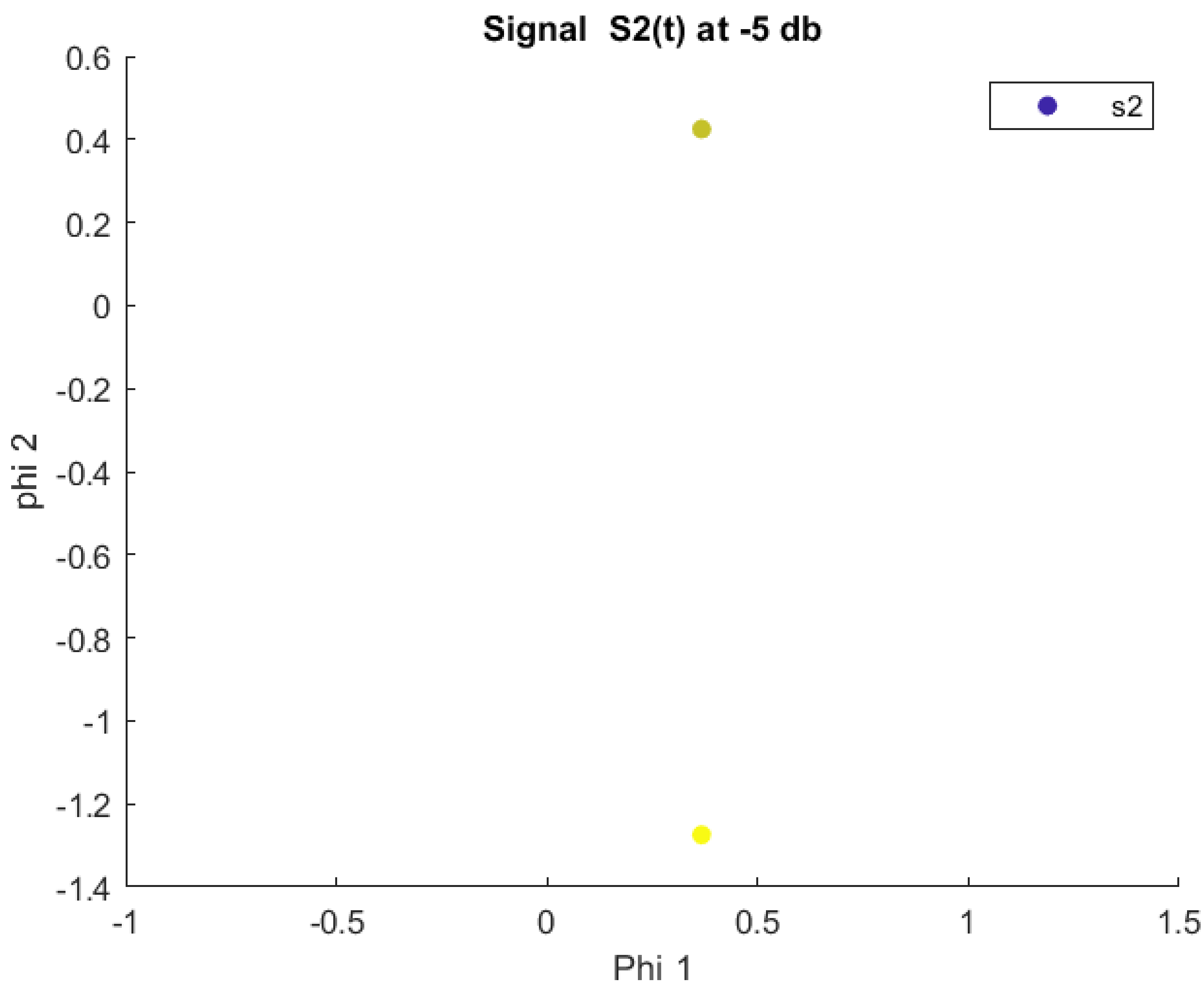

Figure 10 : (s2 +noise) at -5db

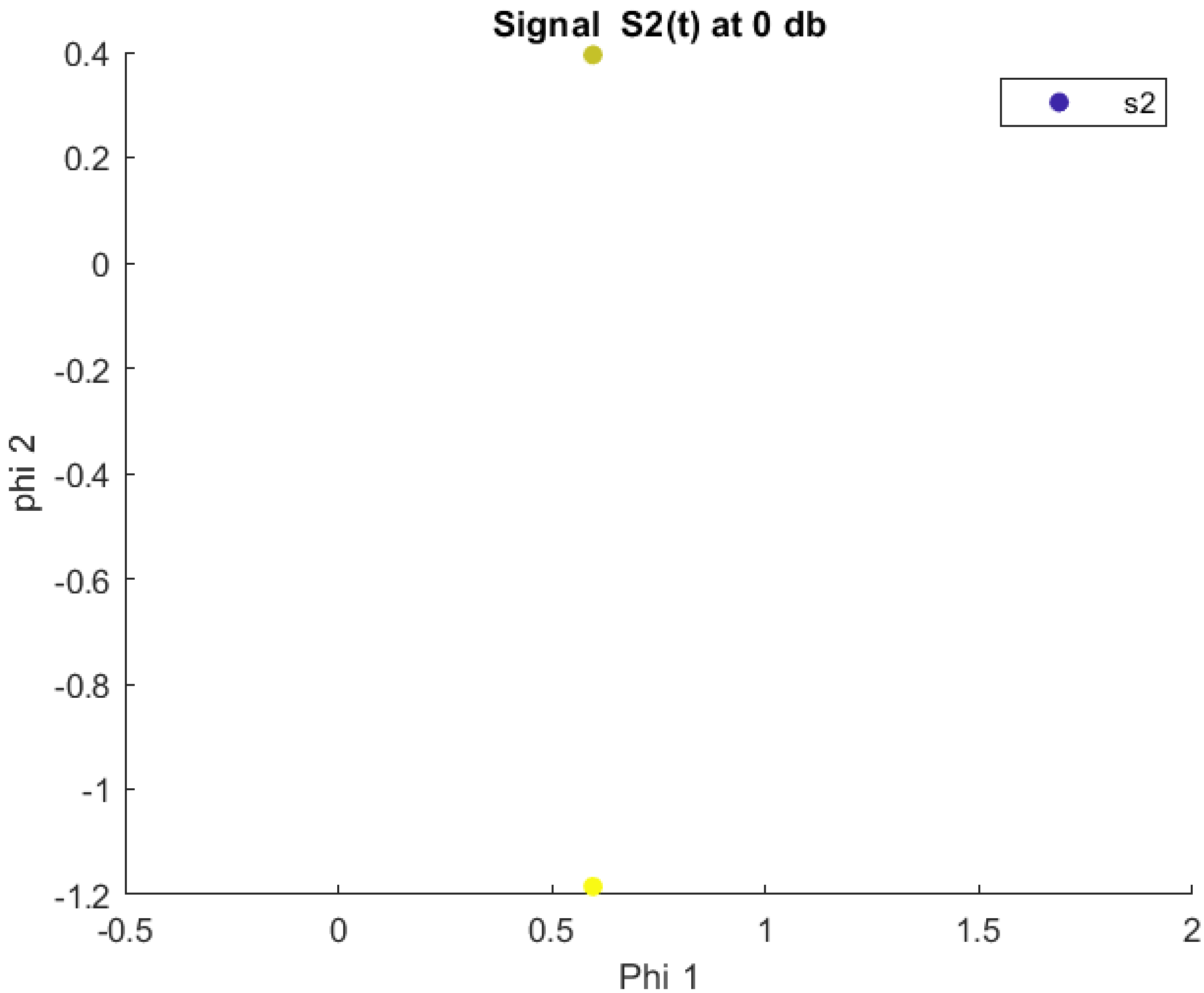

Figure 11 (s2 +noise) at 0db

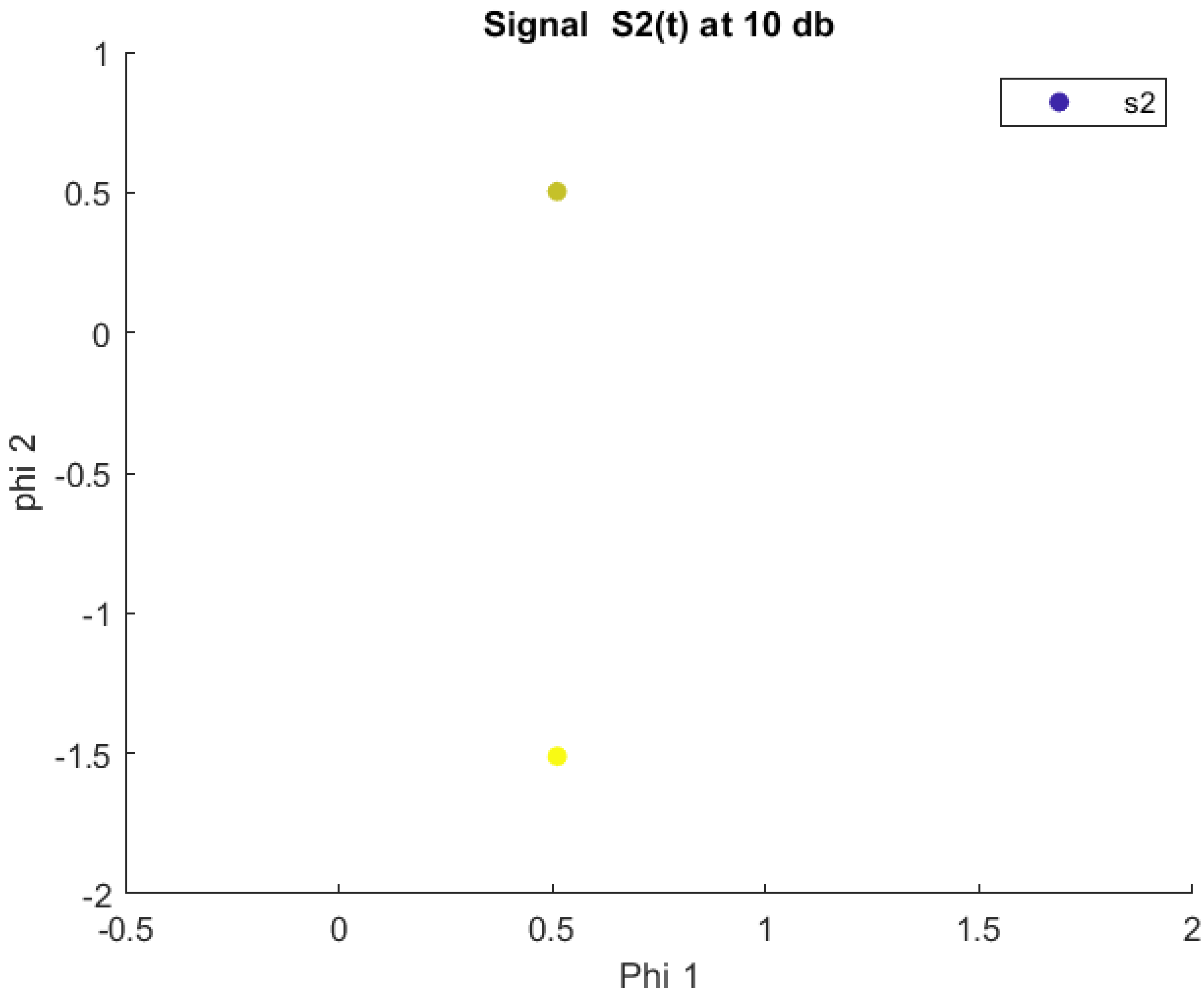

Figure 12 : (s2 +noise) at 10db

1.4 comment

How does the noise affect the signal space? Does the noise effect increase or decrease with increasing sigma square?
Yes it has an effect on signal space representation it make points doesn't allocate correctly the effect of noise increase by increasing the power of the noise comparing to the power of the signal.

2. Part Two

2.1 Calculations of the BER of the four modulation schemes

The BER of the four modulation schemes at SNR=10 dB:

1. BPSK: BER=e^{-5}
2. QPSK: BER=0
3. 8 QAM: BER=4.9 $* e^{-4}$
4. 16-PSK:BER=0.0206

N.T: these values were calculated from the MATLAB simulation

2.2 Decision regions

2.2.1 BPSK:

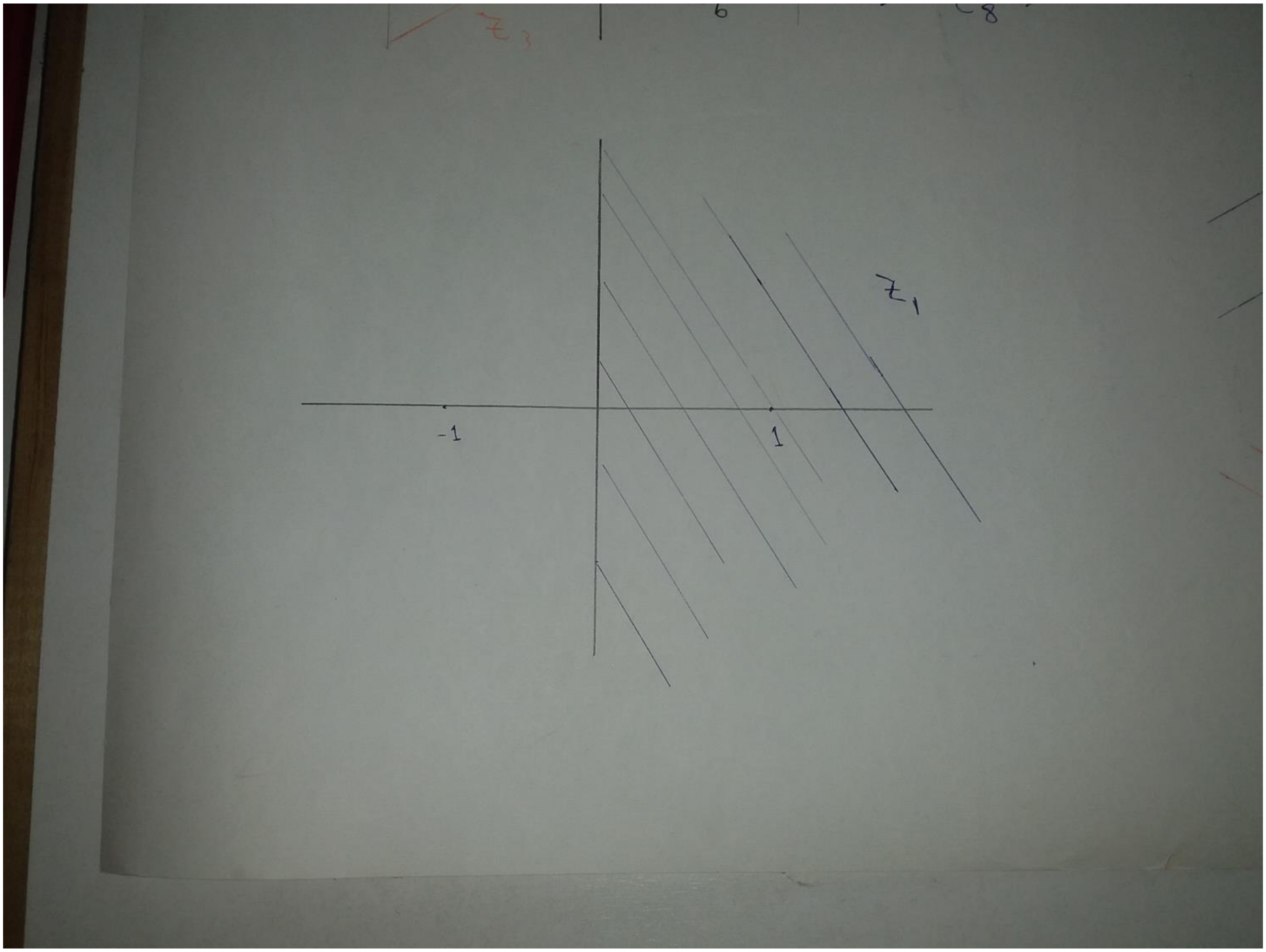

Figure 13decision region for BPSK

2.2.2 QPSK:

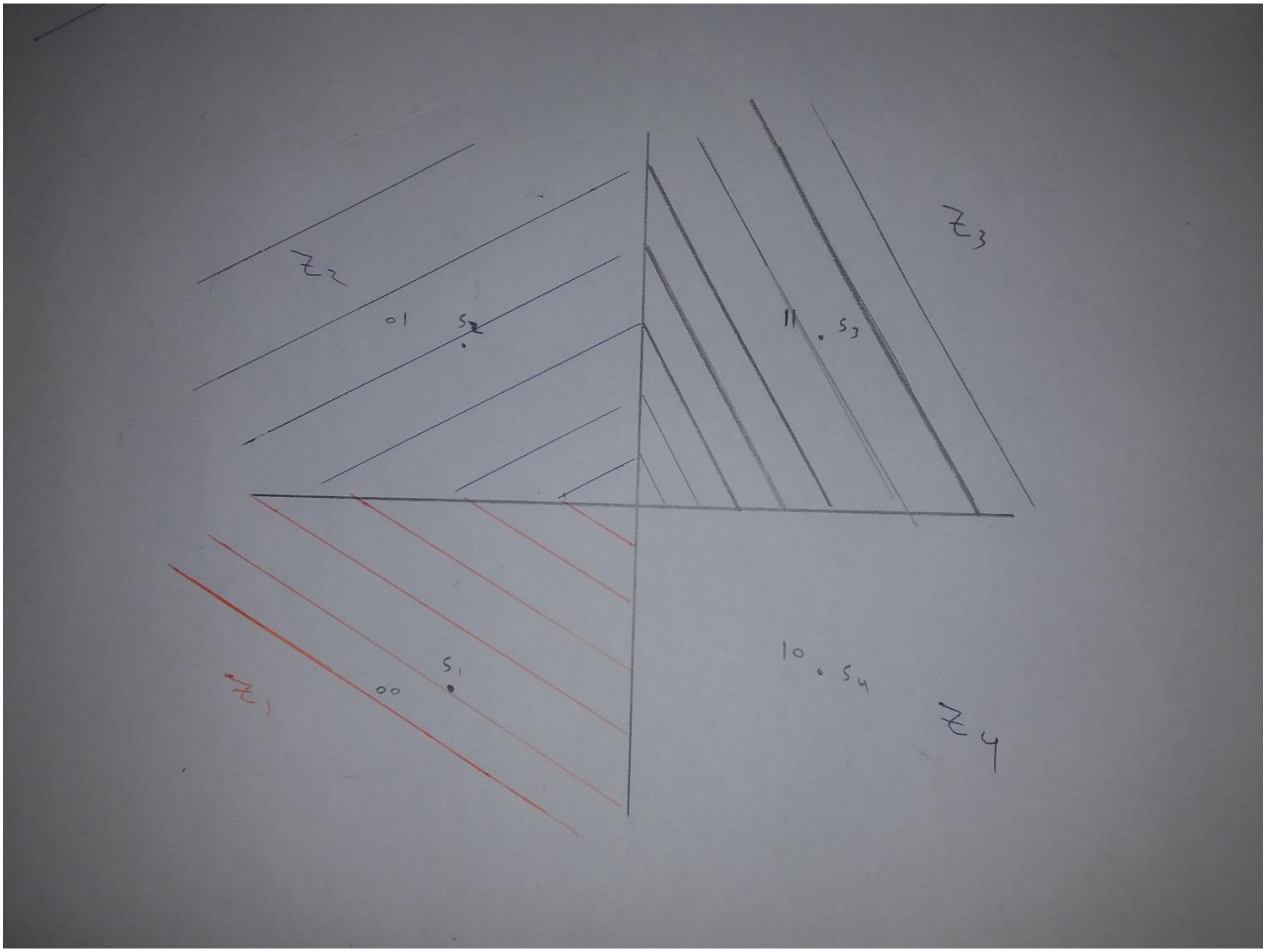

Figure 14 decision region for QPSK

2.2.3 8QAM:

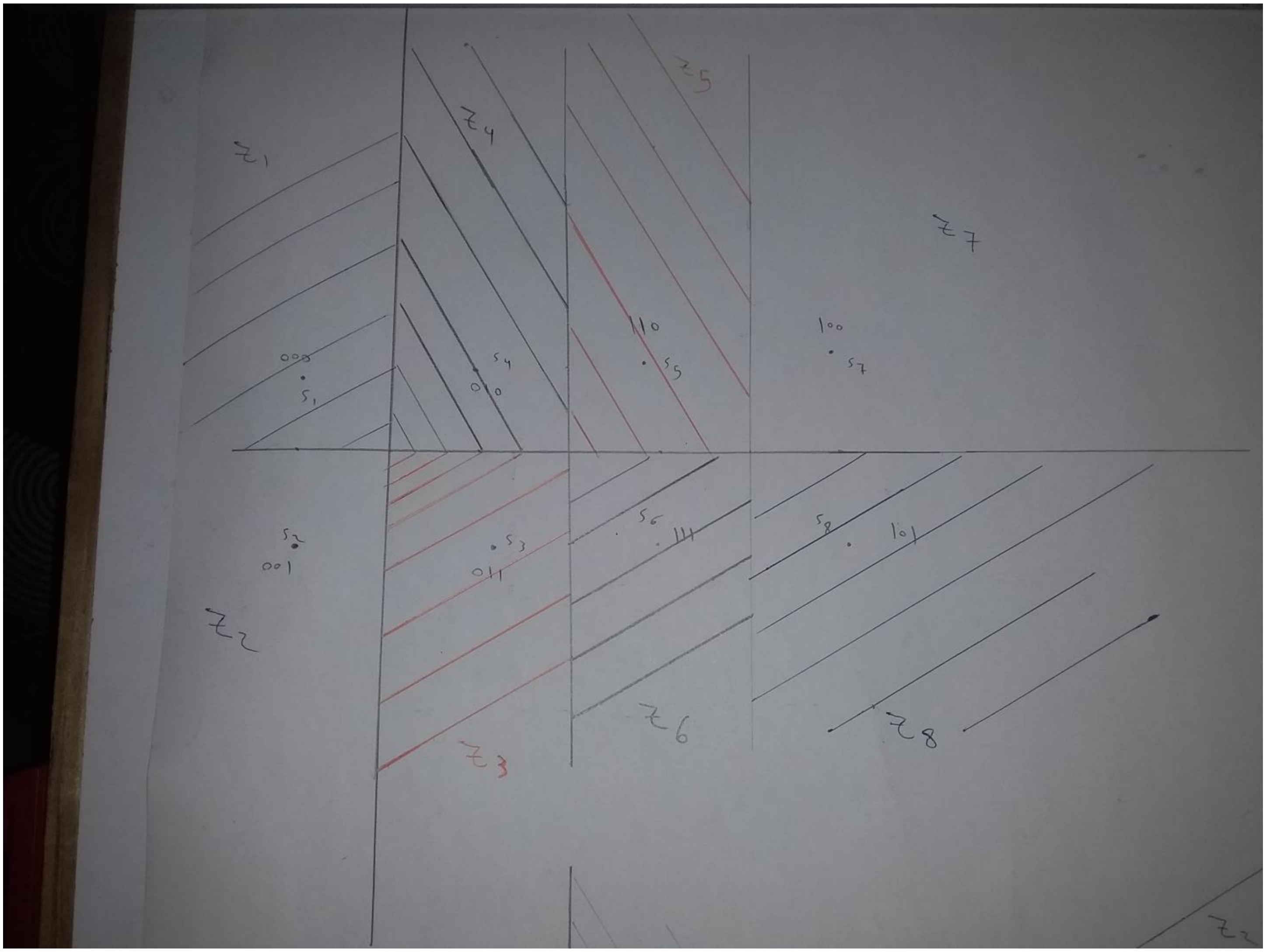

Figure 15 decision region for 8QAM

2.2.4 16PSK:

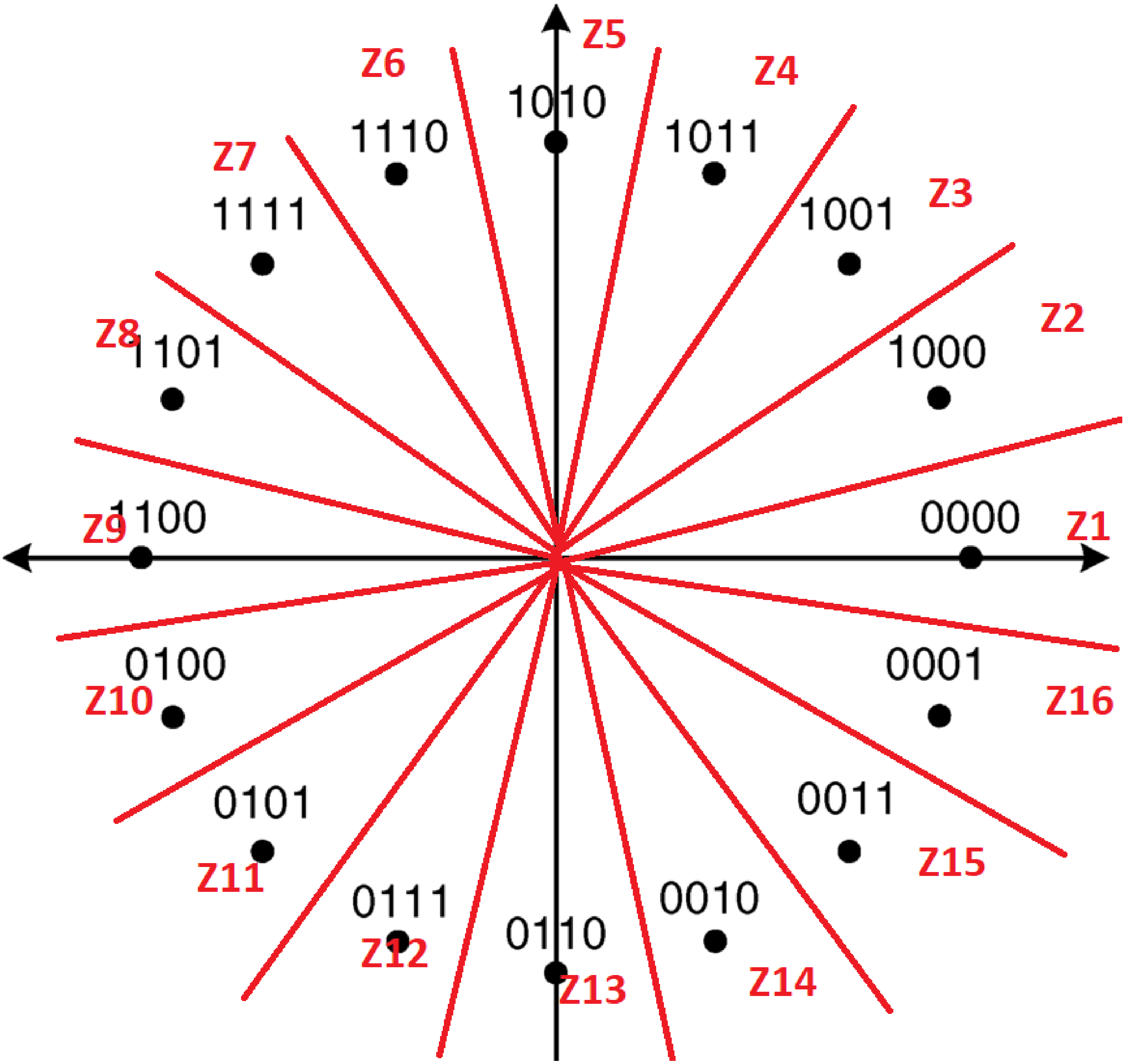

Figure 16decision region for 16 PSK

2.3 Theoretical BER proofs for BPSK, QPSK, QAM and a tight upper bound to the BER of the 16PSK

2.3.1 In a BPSK system the received signal is:

$$x(t) = s_i(t) + w(t)$$

$$x_1 = \int_0^{T_b} x(t)\varphi_1(t)\,dt = \int_0^{T_b} \big(s_i(t) + w(t)\big)\varphi_1(t)\,dt$$

$$f(x_1|1) = N(\sqrt{E_b}, N_o/2)$$

$$f(x_1|0) = N(-\sqrt{E_b}, N_o/2)$$

$$P_{10} = \text{prob}(\text{dec } 1 \mid 0 \text{ sent}) = \int_0^{\infty} f(x_1 \mid 0)\,dx_1 = \frac{1}{\sqrt{\pi N_o}}\int_0^{\infty} e^{-(x_1+\sqrt{E_b})^2/N_o}\,dx_1$$

$$\text{let } \frac{x_1+\sqrt{E_b}}{\sqrt{N_o}} = z$$

$$P_{10} = \frac{1}{\sqrt{\pi}}\int_{\sqrt{\frac{E_b}{N_o}}}^{\infty} e^{-(z)^2}\,dz = 0.5\,\text{erfc}\left(\sqrt{\frac{E_b}{N_o}}\right)$$

Where

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-z^2}\,dz$$

Since the signal space is symmetric, therefore P_{01}, the conditional probability of the receiver deciding in favour of symbol 0, given that 1 was transmitted also has the same value as P_{10}.

$$P_{10} = P_{01} = \frac{1}{2}\,\text{erfc}\left(\sqrt{\frac{E_b}{N_o}}\right)$$

Therefore the average probability of symbol error P_e is

$$P_e = P_0 * P_{10} + P_1 * P_{01} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_o}}\right)$$

Where$P_0 = P_1 = 0.5$

2.3.2 In a QPSK system the received signal can be written as:

$$x(t) = s_i(t) + w(t)$$

$$x_1 = \int_0^{T_b} x(t)\varphi_1(t)\,dt = \int_0^{T_b} \big(s_i(t) + w(t)\big)\varphi_1(t)\,dt = \pm\sqrt{\frac{E}{2}} + w_1$$

$$x_2 = \int_0^{T_b} x(t)\varphi_2(t)\,dt = \int_0^{T_b} \big(s_i(t) + w(t)\big)\varphi_2(t)\,dt = \pm\sqrt{\frac{E}{2}} + w_2$$

$x_1$and x_2 are independent Gaussian random variables with means(μ) $=\pm\sqrt{\frac{E}{2}}$ and variance (σ^2) $=\frac{N_o}{2}$

Coherent QPSK is equivalent to 2 coherent BPSK systems working in parallel & using 2 carriers that are in phase quadrature. x_1 and x_2 can be viewed as the individual O/Ps of the 2 coherent BPSK systems, but note that the signal energy is E/2.

$$P`_e = 0.5\, erfc\,(\sqrt{\frac{E_b}{N_o}})$$

E_b= E/2

$$P`_e = 0.5\, erfc\,(\sqrt{\frac{E/2}{N_o}})$$

Probability of symbol error $P_e=2P`_e-P`_e{}^2$

2.3.3 In 8-QAM system:

probability of error for 4_ASK:

$$P_{e\ of\ 4ASK} = \frac{3}{4} * erfc\left(\sqrt{\frac{E_o}{2N_o}}\right)$$

The probability of symbol error $P_{e\ of\ PBSK}$= probability of **BPSK**= $\frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_o}}\right)$

Probability of correct detection for **8QAM**:

$$P_c = (1 - P_e)(1 - P_{e\ of\ \mathbf{BPSK}})$$

$$P_c = \left(1 - \frac{3}{4} * erfc\left(\sqrt{\frac{E_o}{2N_o}}\right)\right)\left(1 - \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_o}}\right)\right)$$

The probability of symbol error $P`_e$:

$$P`_e = 1 - P_c$$

$$P`_e = 1.25 * \ erfc\left(\sqrt{\frac{E_o}{2N_o}}\right) - \frac{3}{8} erfc^2\left(\sqrt{\frac{1}{N_o}}\right)$$

$$so\ we\ can\ consider\ P`_e = 1.25 * \ erfc\left(\sqrt{\frac{E_o}{2N_o}}\right) = 1.25 * \ erfc\left(\sqrt{\frac{1}{N_o}}\right)$$

2.3.4 In 16-PSK system:

We can't get analytical expression, so we use union bounds.

The conditional probability of symbol error when m_i is sent $P_e(m_i)$ is equal to the probability of union of events $A_{i1}, A_{i2}, \dots, A_{iM}$.

The probability of finite union of events is upper bounded by the sum of the probability of constituent events.

$$P_e(m_i) \le \sum_{k=1, k\neq i}^{M} P\,(A_{ik})$$

$$P(A_{ik}) = \int_{d/2}^{\infty} \frac{1}{\sqrt{\pi N_o}} e^{-v^2/N_o} dv = \frac{1}{2}\, erfc(\frac{d_{ik}}{2\sqrt{N_o}})$$

$$P_e(m_i) \le \sum_{k=1, k\neq i}^{M} \frac{1}{2}\, erfc\left(\frac{d_{ik}}{2\sqrt{N_o}}\right) \le \frac{M-1}{2}\, erfc\left(\frac{d_{min}}{2\sqrt{N_o}}\right)$$

Where d_{min} is he minimum distance between any 2 transmitted signal points

For M=16

$$\mathrm{E} = (\log_2 \mathrm{M}) * \mathrm{E_b}$$

$$\mathrm{d_{min}} = 2\sqrt{\mathrm{E}}\sin(\frac{\pi}{\mathrm{M}})$$

$$P_e \le \frac{M-1}{2}\, erfc\left(\sqrt{\frac{4}{N_o}}\sin(\frac{\pi}{M})\right)$$

In MPSK, the 2 neighboring points cover the whole error area, thus the tight bound is:

$$P_e = erfc\left(\sqrt{\frac{4}{N_o}}\sin(\frac{\pi}{M})\right)$$

$$BER = \frac{P_e}{\log_2 M}$$

$$BER = \frac{1}{4} erfc\left(\frac{2}{\sqrt{N_o}}\sin(\frac{\pi}{16})\right)$$

2.4 Plot for the simulated BER Verses E_b/N_o

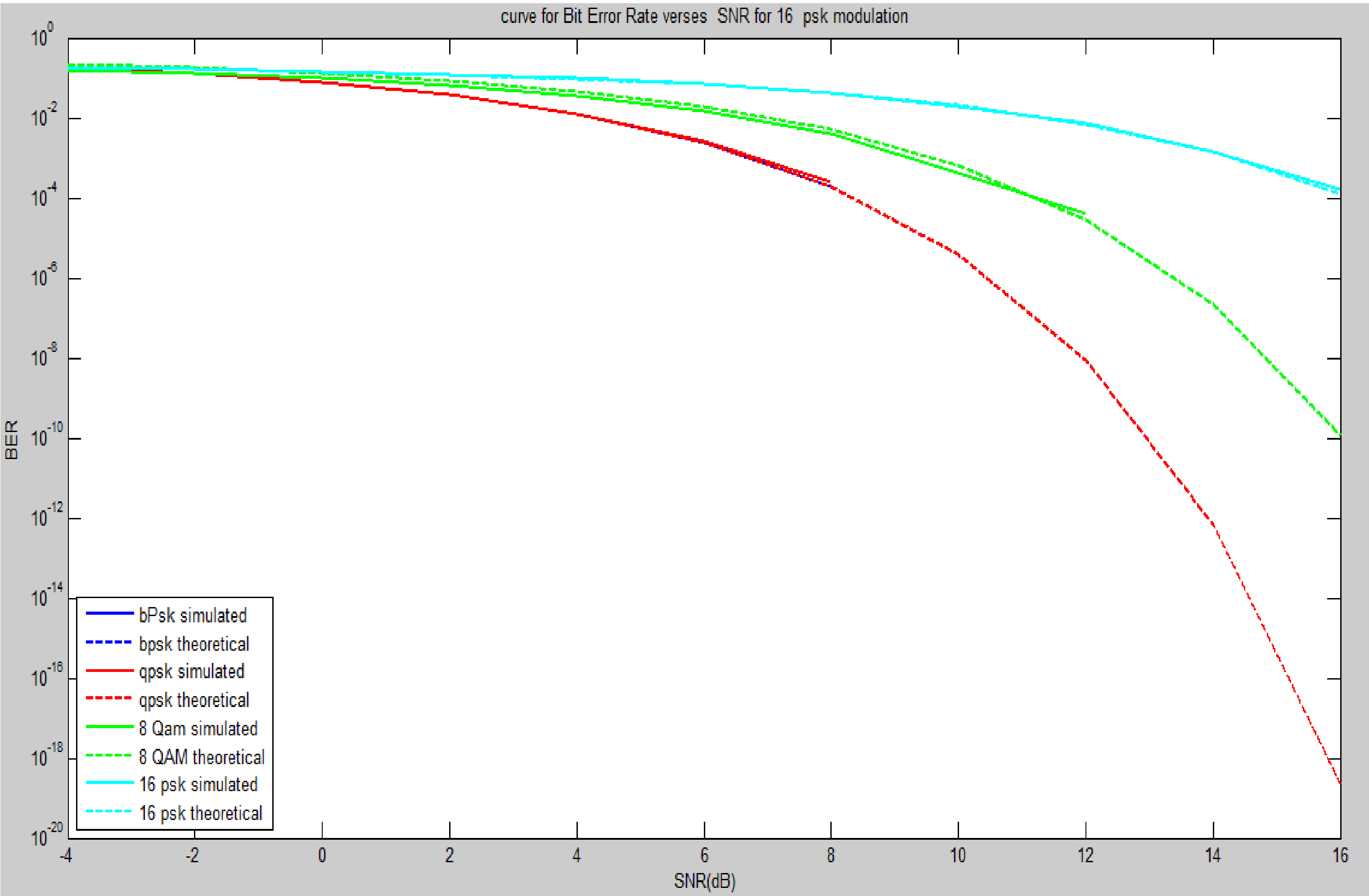

Figure 17 plot of the simulated BER

2.5 Requirement six

it is required to design a system that uses M-ary PSK so that SNR doesn't exceed 5dB and BER doesn't exceed 10^{-2} then 8-PSK can't be used as it doesn't meet the requirements,

And the bit rate required equals 0.5Mbps and the available bandwidth is 0.5MHz centered at carrier frequency 5 MHz.

$$Bw_{Max} = \frac{2 * r_b}{\log_2 M_{Min}}$$

Therefore

$$M_{Min} = 2^{\frac{2*0.5}{0.5}} = 4$$

So the only PSK modulation that can be used is QPSK as it has M equal 4

3 Part Three

3.1. Drawing the output of the transmitter

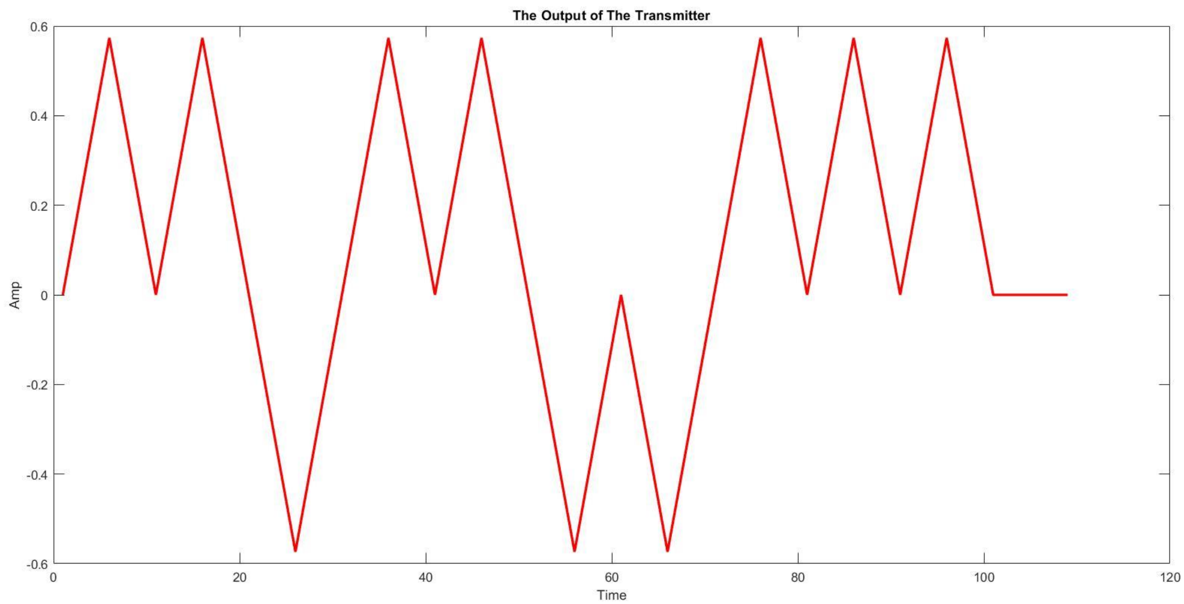

Figure 18output of transmitter

This plot is a result of generating 10 random bits and mapping them between 1 and -1 with bit rate (0.1 sec.).

3.2. Drawing the output of the receiver

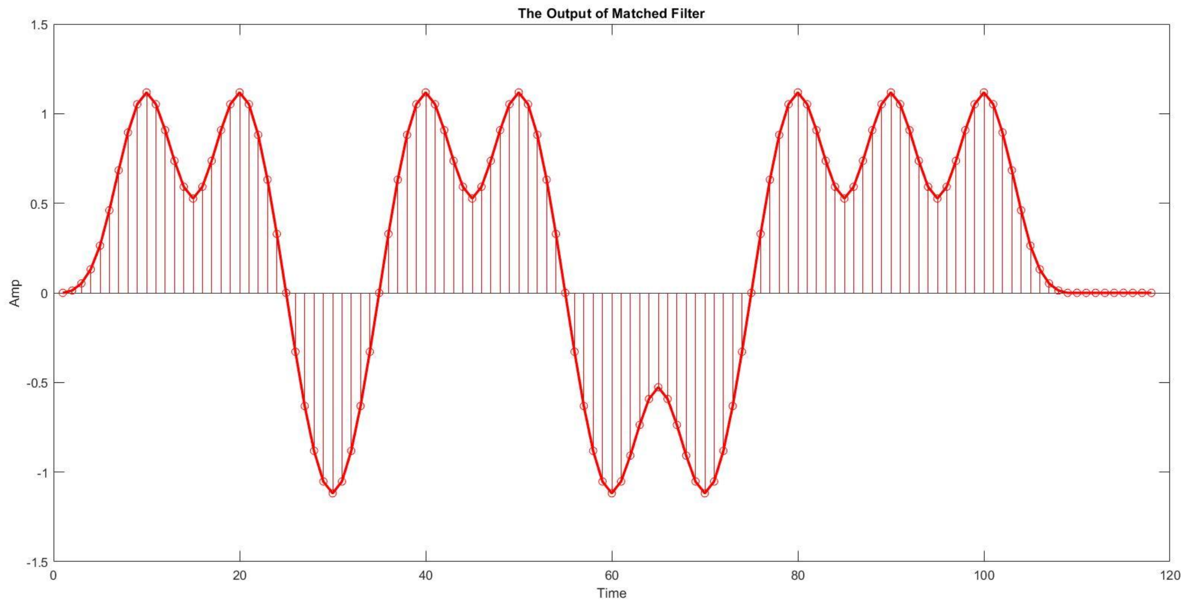

Figure 19output of receiver

After transmitting the bits and receive it in the receiver part, it passes throw a matched filter and after that returning into almost the transmitted bits values. All we need to do is to sample that curve to get the transmitted bits.

3.3. Drawing the bits after sampling

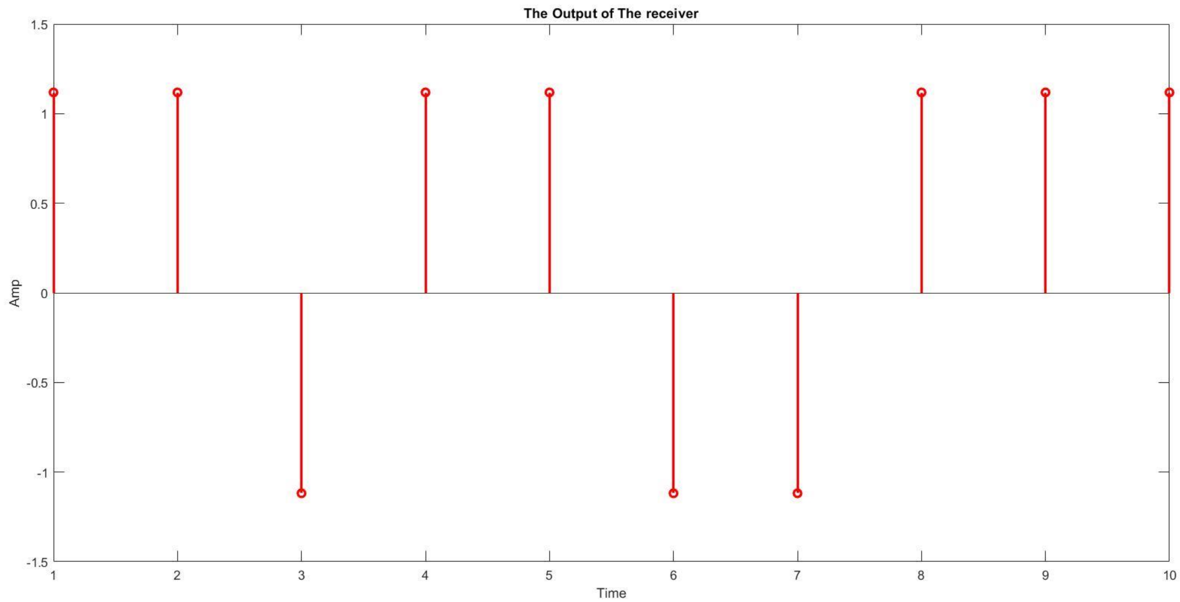

Figure 20output of demodulated signal(bits)

After sampling we get exactly the transmitted bits.

Appendix A: Codes for Part One:

A.1 Gm BASES function

```
function [phi1,phi2]=GM_Bases(S1,S2)


phi1=S1/(sqrt(sum(S1.*S1)));
phi2=S2-(((sum(phi1.*S2))*phi1))
phi2=phi2/(sqrt(sum(phi2.*phi2)))

end
```

A.2 signal Space

```
function [v1,v2]=signal_space(S1, phi1,phi2)
v1=[];
v2=[];
v1=(dot(S1,phi1)/norm(phi1)^2)*phi1;
v2=(dot(S1,phi2)/norm(phi2)^2).*phi2;
end
```

A.3 noise function

```
unction [noise_signal]=noise(SNRDB,Signal)
SNRdB=SNRDB; %Eb/No in dB
SNR=10.^(SNRdB/10); %Eb/No
N0=1./(SNR);

for i =1:length(Signal)
%generating AWGN
Noise(i) = sqrt(N0)*randn(1,1);
end
%Adding noise to the signal
noise_signal=Signal+Noise;
```

```
end
```

A.4 Generating s1&s2

```
S1 = ones(1,100);
L = ones(1,75);
c=zeros(1,25)
S2=[L c]
for i=1:length(S2)
    S2(i)=S2(i)*2-1;
end
```

A.5 ploting output

```
[phi1,phi2]=GM_Bases(S1,S2)
plot(phi1,'LineWidth',2);
xlabel('time');
ylabel('Amplitude');
title('phi 1');
legend('phi 1');
figure;
plot(phi2,'LineWidth',2);
xlabel('time');
ylabel('Amplitude');
title('phi 2');
legend('phi 2');
%axis([0 1 -.1 1.2])
figure;
[v1,v2]=signal_space(S1, phi1,phi2)
plot(v1,'LineWidth',2);
xlabel('time');
ylabel('phi 1');
title('Signal space representation of S1(t)');
legend('S1 projection on phi 1');
figure;
plot(v2,'LineWidth',2);
xlabel('time');
```

```
ylabel('phi 2');
title('Signal space representation of S1(t)');
legend('S1 projection on phi 2');
figure;
[v3,v4]=signal_space(S2, phi1,phi2)
plot(v3,'LineWidth',2);
xlabel('time');
ylabel('phi 1');
title('Signal space representation of S2(t)');
legend('S2 projection on phi 1');
figure;
plot(v4,'LineWidth',2);
xlabel('time');
ylabel('phi 2');
title('Signal space representation of S2(t)');
legend('S2 projection on phi 2');
noise_r1=noise(10,S1);
noise_r2=noise(-5,S2);
%========================================================
[v1,v2]=signal_space(noise_r1, phi1,phi2)
figure;
c = linspace(1,10,length(v1));
scatter(v1,v2,[],c,'filled')
legend(' s2' );
xlabel('Phi 1');
ylabel('phi 2');
title('Signal  S2(t) at -5 db');
```

Appendix B: Codes for Part Two:

B.1 Code for Binary psk:

```
%XXXXXXX code of bpsk(binary phase shift keying)XXXXXXXXXXXXXXX

clc
clear
numberofbits=100000; %number of generated bits
data= randint(1,numberofbits); % generating random bits(1 or 0)
```

```
s=2*data-1; % convertin 0 to -1 and 1 to 1
Eb=1; %Energy bit
SNRdB=-4:2:16; %Eb/No in dB
SNR=10.^(SNRdB/10); %Eb/No
N0=1./SNR;
for k=1:length(SNRdB)
%generating AWGN
noise = sqrt(N0(k)/2)*randn(1,length(s));
%Adding noise to the signal
x=s+noise;
error=0;
for i=1:numberofbits
if(x(i)>0 && data(i)==0 || x(i)<0 && data(i)==1) % if error occurs
error=error+1; %increment error counter
end
end
error=error/numberofbits; % calculating error/bit
BER(k)=error;
end

semilogy(SNRdB,BER,'b','linewidth',2);
holdon;
BER_theoritical=(1/2)*erfc(sqrt(SNR));
semilogy(SNRdB,BER_theoritical,'b--','linewidth',2);
holdon;


title(' curve for Bit Error Rate verses SNR for bpsk');
xlabel(' SNR(dB)');
ylabel('BER');

%XXXXXXXXXXXXXXXXXXXXX End of bPsk XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

B.2 Code for QPSK:

```
%XXXXXXX code of qpsk(quadrature phase shift keying) XXXXXXXXXX

clc
clear
numberofbits=100000; %number of generated bits
data= randint(1,numberofbits); % generating random bits(1 or 0)
s=2*data-1; % convertin 0 to -1 and 1 to 1
Eb=1; %Energy bit
SNRdB=-4:2:16; %Eb/No in dB
```

```
SNR=10.^(SNRdB/10); %Eb/No
N0=1./SNR;
symbols=[11,01,00,10]; %array of symbols

%..............................QPSK................................

k=1;
%  looping on each two bits in the stream to get (x,y)
for i =1:2:100000-1
if(data(i)==1)
x(k)=1;
else
x(k)=-1;
end
if  (data(i+1)==1)
y(k)=1;
else
y(k)=-1;
end
k=k+1;
end

%llopin on different SNRs
for k=1:length(SNRdB)
error=0;
fori=1:(numberofbits/2)
noise = sqrt(N0(k)/2)*randn(1); %generating noise for x component
    noise2 = sqrt(N0(k)/2)*randn(1); %generating noise for y component
xn(i)=x(i)+noise;
yn(i)=y(i)+noise2;

if((xn(i)>0 && x(i)==-1) || (xn(i)<0 && x(i) ==1) )
error=error+1;
end
if( (yn(i)>0 && y(i)==-1) ||(yn(i)<0 && y(i)==1))
error=error+1;
end
end
error=((error)/numberofbits); % calculating error/bit
BERqpsk(k)=error;
end

semilogy(SNRdB,BERqpsk,'r','linewidth',2);
holdon;
```

```
BER_theoritical_qpsk=(1/2)*erfc(sqrt(SNR));
semilogy(SNRdB,BER_theoritical_qpsk,'r--','linewidth',2);
title(' curve for Bit Error Rate verses  SNR for Binary PSK modulation');
xlabel(' SNR(dB)');
ylabel('BER');

%XXXXXXXXXXXXXXXXXXXXX End of QPsk XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

B.3 Code for 8QAM:

```
%XXXX code of 8-QAM(8 quadrature phase shift keying) XXXXXX

clc
clear
numberofbits=100000; %number of generated bits
data= randint(1,numberofbits); % generating random bits(1 or 0)
s=2*data-1; % convertin 0 to -1 and 1 to 1
Eb=2; %Energy bit
SNRdB=-4:2:16; %Eb/No in dB
SNR=10.^(SNRdB/10); %Eb/No
N0=2./SNR;

%.......................8-Qpsk..............................
symbols=[100,110,010,000,001,011,111,101];
k=1;
%  looping on each 3 bits in the stream to get (x,y)
for i =1:3:100000-2
if(data(i)==1)
x(k)=1;
else
x(k)=-1;
end
if (data(i+2)==1)
y(k)=-1;
else
y(k)=1;
end
if(data(i+1)==0)
x(k)=x(k)*3;
y(k)=y(k)*3;
end
k=k+1;
end
```

```
sent = detectsymbolnew(x,y);
%loopin on different SNRs
for k=1:length(SNRdB)
noise = sqrt(N0(k)/2)*randn(1,length(x)); %generating noise for x
component
    noise2 = sqrt(N0(k)/2)*randn(1,length(y));%generating noise for y
component
xn=x+noise;
yn=y+noise2;
received2 = detectsymbolnew(xn,yn);

error=0;
for i =1:length(sent)
if(sent(i) ~= received2(i)) %if error detected
error=error+1; %increment error counter
end
end
BER(k)=error/(numberofbits);% calculating error/bit
r(k,:)=received2;
end

semilogy(SNRdB,BER,'g','linewidth',2);
holdon;
BER_theoritical_8qam =5/12*erfc(sqrt(Eb./(2*N0)));

semilogy(SNRdB,BER_theoritical_8qam,'g--','linewidth',2);
title(' curve for Bit Error Rate verses  SNR for 8  QAM modulation');
xlabel(' SNR(dB)');
ylabel('BER');


%XXXXXXXXXXXXXXXXXXXXX End of 8-QPsk XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

B.4 Code for 16QPSK:

```
%XXXXXXXXXX code of 16-Psk(16 phase shift keying) XXXXXXXXXXX

clc
clear
numberofbits=100000; %number of generated bits
data= randint(1,numberofbits); % generating random bits(1 or 0)
s=2*data-1; % convertin 0 to -1 and 1 to 1
Eb=1/4; %Energy bit
SNRdB=-4:2:16; %Eb/No in dB
SNR=10.^(SNRdB/10); %Eb/No
N0=1./(4*SNR);
symbols=[0000,1000,1001,1011,1010,1110,1111,1101,1100,0100,0101,0111,0110,
0010,0011,0001];

k=1;
%  looping on each 4 bits in the stream to get theta (each symbol has
%  specific theta)
for i =1:4:numberofbits
if(data(i)==1)
if(data(i+1)==1)
if(data(i+2)==1)
if(data(i+3)==1)
theta(k)=225;
else
theta(k)=247.5;
end
else
if(data(i+3)==1)
theta(k)=202.5;
else
theta(k)=180;
end
end
quad(k)=3;
else

if(data(i+2)==1)
if(data(i+3)==1)
theta(k)=292.5;
else
theta(k)=270;
end
```

```
else
if(data(i+3)==1)
theta(k)=315;
else
theta(k)=337.5;
end
end
quad(k)=4;
end
else


if(data(i+1)==1)
if(data(i+2)==1)
if(data(i+3)==1)
theta(k)=112.5;
else
theta(k)=90;
end
else
if(data(i+3)==1)
theta(k)=135;
else
theta(k)=157.5;
end
end
quad(k)=2;
else

if(data(i+2)==1)
if(data(i+3)==1)
theta(k)=45;
else
theta(k)=67.5;
end
else
if(data(i+3)==1)
theta(k)=22.5;
else
theta(k)=0;
end
end
quad(k)=1;
end
```

```
end
 k=k+1;

end

x= cosd(theta);
y=sind(theta);
%loopin on different SNRs
for k=1:length(SNRdB)
noisex = sqrt(N0(k)/2)*randn(1,length(x));%generating noise for x
component
noisey = sqrt(N0(k)/2)*randn(1,length(y));%generating noise for y
component
xn=x+noisex;
yn=y+noisey;

fori=1:length(xn)
phrecieved=atand(abs(yn(i)/xn(i))); %get theta recieved(first quad i.e
range from 0 to 90)
% get theta range (from 0 to 360)
if(xn(i)<=0 &&yn(i)<0)
quadr(i)=3;
phrecieved=180+phrecieved;
elseif (xn(i)>=0 &&yn(i)>0)
quadr(i)=1;
elseif (xn(i)>0 &&yn(i)<0)
quadr(i)=4;
phrecieved=360-phrecieved;
elseif (xn(i)<0 &&yn(i)>0)
quadr(i)=2;
phrecieved=180-phrecieved;
else
% y(i)==0
%theta(i)=90 or 270
if(x(i)>0)
thetarecieved(i)=90;
if (theta(i)~=90)
error=error+1;
continue;
end
else
%x(i)=-1
%theta =270
thetarecieved(i)=270;
```

```
if (theta(i)~=270)
error=error+1;

end
continue;
end
end

%referring each theta recieved to the nearest symbol theta(multiples of
%(2*pi/16)
if((phrecieved>=348.75 &&phrecieved<=360 )||(phrecieved>=0
&&phrecieved<=11.25))
thetarecieved(i)=0;
elseif(phrecieved>11.25 &&phrecieved<=33.75)
thetarecieved(i)=22.5;
elseif(phrecieved>33.75 &&phrecieved<=56.25)
thetarecieved(i)=22.5*2;
elseif(phrecieved>56.25 &&phrecieved<=78.75)
thetarecieved(i)=22.5*3;
elseif(phrecieved>78.75 &&phrecieved<=101.25)
thetarecieved(i)=22.5*4;
elseif(phrecieved>101.25 &&phrecieved<=123.75)
thetarecieved(i)=22.5*5;
elseif(phrecieved>123.75 &&phrecieved<=146.25)
thetarecieved(i)=22.5*6;
elseif(phrecieved>146.25 &&phrecieved<=168.75)
thetarecieved(i)=22.5*7;
elseif(phrecieved>168.75 &&phrecieved<=191.25)
thetarecieved(i)=22.5*8;
elseif(phrecieved>191.25 &&phrecieved<=213.75)
thetarecieved(i)=22.5*9;
elseif(phrecieved>213.75 &&phrecieved<=236.25)
thetarecieved(i)=22.5*10;
elseif(phrecieved>235.25 &&phrecieved<=258.75)
thetarecieved(i)=22.5*11;
elseif(phrecieved>258.75 &&phrecieved<=281.25)
thetarecieved(i)=22.5*12;
elseif(phrecieved>281.25 &&phrecieved<=303.75)
thetarecieved(i)=22.5*13;
elseif(phrecieved>303.75 &&phrecieved<=326.25)
thetarecieved(i)=22.5*14;
elseif(phrecieved>326.25 &&phrecieved<=348.75)
thetarecieved(i)=22.5*15;
```

```
end
end

error=0;
fori =1:length(thetarecieved)
if(thetarecieved(i) ~= theta(i)) %detecting error
error=error+1;
end
end
BER(k)=error/(numberofbits);
end
semilogy(SNRdB,BER,'c','linewidth',2);
holdon;
BER_theoritical_16psk =0.25*erfc(sqrt(1./N0)*sind(180/16));

semilogy(SNRdB,BER_theoritical_16psk,'c--','linewidth',2);
title(' curve for Bit Error Rate verses  SNR for different modulations');
xlabel(' SNR(dB)');
ylabel('BER');
legend({'bPsk simulated','bpsk theoretical','qpsk simulated','qpsk theoretical','8 Qam simulated','8 QAM theoretical','16 psk simulated','16 psk theoretical'},'Location','southwest')


%XXXXXXXXXXXXXXXXXXXX End of 16-Psk XXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

B.5 Code for ‘detectsymbol’ function:

```
%XXXXXXXXXXXXXXXXXXXX function to detect symbols XXXXXXXXXXXXXXXXXXXXXXXXXXXX

function symbol = detectsymbolnew(xn,yn)
k=1;
%array of symbols
symbols=[100,110,010,000,001,011,111,101];
%loop on each 3 bits
for i =1:3:100000-2
if(xn(k) > 2)
if(yn(k) > 0)
%if x>2 and y>0 (decision region of symbol 1)
symbol(k)=symbols(1);
else
```

```
%if x>2 and y<0 (decision region of symbol 8)
symbol(k)=symbols(8);
end
elseif(xn(k) > 0 )
if(yn(k) > 0)
%if 0<x<2 and 0<y (decision region of symbol 2)
symbol(k)=symbols(2);
else
%if 0<x<2 and y<0 (decision region of symbol 7 )
symbol(k)=symbols(7);
end

elseif (xn(k)<-2)
if(yn(k)>0)
%if x<-2 and 0<y (decision region of symbol 4)
symbol(k)=symbols(4);
else
%if x<-2 and 0>y (decision region of symbol 5)
symbol(k)=symbols(5);
end
else
if(yn(k)>0)
%if 0>x>-2 and 0<y (decision region of symbol 3)
symbol(k)=symbols(3);
else
%if 0>x>-2 and 0>y (decision region of symbol 2)
symbol(k)=symbols(6);
end
end
    k=k+1;
end

%XXXXXXXXXXXXXXXXXXXX End of function XXXXXXXXXXXXXXXXXXXXXXXXX
```

Appendix C: Codes for Part Three:

C.1 Code for generating 10 random bits of 1 and -1

```
rand_bits = randi([0 1],1,10);
rand_bits = ((2*rand_bits(1,:))-1);
```

C.2 Code for generating impulse train and pulse shaping function

```
% generating an impulse train
imp_train = [1 1 1 1 1 1 1 1 1 1];
imp_train = rand_bits.*imp_train;
imp_train = upsample(imp_train,10);

% generating the discrete pulse shaping function
p = [0 2 4 6 8 10 8 6 4 2 ]/sqrt(304);
```

C.3 Code for matched filter

```
% convolving
y = conv(p,imp_train);


% Matching Filters
MF1 = fliplr(p);


% o/p of Matching Filters
OP1 = conv(y,MF1);
```

C.4 Code for the receiver

```
%o/p of receiver
srr=[];
for i=10:10:length(OP1)-10
sr=OP1(i);
srr = [srrsr];
end
```

C.5 Code for plotting the transmitter output

```
% plotting o/p of transmitter
figure(1)
plot(y,'r');
title('The Output of The Transmitter');ylabel('Amp');xlabel('Time');
```

C.6 Code for plotting the filter output

```
% plotting o/p of matched filter
figure(2)
plot(OP1,'r');
holdon
stem(OP1,'r');
title('The Output of Matched Filter');ylabel('Amp');xlabel('Time');
```

C.7 Code for plotting the receiver output

```
% plotting o/p of receiver
figure(3)
stem(srr,'r');
title('The Output of The receiver');ylabel('Amp');xlabel('Time');
```

C.8 The whole code

```
closeall;
clearall;
clc;
```

```
%%%%%%%%%%%%%%%%%%%%  Matched Filters  %%%%%%%%%%%%%%%%%%%%%%%
% generating 10 random bits of 1's & -1's
rand_bits = randi([0 1],1,10);
rand_bits = ((2*rand_bits(1,:))-1);

% generating an impulse train
imp_train = [1 1 1 1 1 1 1 1 1 1];
imp_train = rand_bits.*imp_train;
imp_train = upsample(imp_train,10);

% generating the discrete pulse shaping function
p = [0 2 4 6 8 10 8 6 4 2 ]/sqrt(304);

% convolving
y = conv(p,imp_train);

% Matching Filters
MF1 = fliplr(p);

% o/p of Matching Filters
OP1 = conv(y,MF1);

%o/p of receiver
srr=[];
for i=10:10:length(OP1)-10
sr=OP1(i);
srr = [srrsr];
end


% plotting o/p of transmitter
figure(1)
plot(y,'r');
title('The Output of The Transmitter');ylabel('Amp');xlabel('Time');


% plotting o/p of matched filter
figure(2)
plot(OP1,'r');
holdon
stem(OP1,'r');
title('The Output of Matched Filter');ylabel('Amp');xlabel('Time');
```

```
% plotting o/p of receiver
figure(3)
stem(srr,'r');
title('The Output of The receiver');ylabel('Amp');xlabel('Time');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```