

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA VIỄN THÔNG I



BÁO CÁO BÀI TẬP LỚN
MÔ PHỎNG HỆ THỐNG TRUYỀN THÔNG

Giảng viên	:Ngô Thị Thu Trang
Nhóm lớp học	:Nhóm 5
Họ và tên	:Nguyễn Di Đan
Mã sinh viên	:B20DCVT086
Lớp	:D20CQVT06-B

Hà Nội – 2023

MỤC LỤC

Nhiệm vụ 1	3
1. Mô tả nhiệm vụ.	3
2. Yêu cầu.....	3
a) Mô tả phương pháp thực hiện chuyển đổi file nhạc yêu cầu thành chuỗi nhị phân và ngược lại.	3
b) Viết chương trình MATLAB.....	4
c) Xác định các thông số về file nhạc cần chuyển đổi.....	7
Nhiệm vụ 2	8
1. Mô tả nhiệm vụ:	8
2. Yêu cầu:	9
a) Mô tả hệ thống mô phỏng bằng sơ đồ khối và xác định các tham số của hệ thống.....	9
b) Viết chương trình MATLAB.....	9
c) Biểu diễn biểu đồ chòm sao, dạng sóng tín hiệu, mẫu mắt và phổ của tín hiệu.	16
d) So sánh file nhạc được khôi phục sau khi truyền qua hệ thống mô phỏng tại các mức SNR yêu cầu.....	27

NHIỆM VỤ 1

1. Mô tả nhiệm vụ.

Đọc và xử lý nguồn tin là một file nhạc AssignmentD20.wav. File này có thể lấy về theo địa chỉ dưới đây:

https://drive.google.com/file/d/1fown3-91kGSUhZA6L8eIjS1Oh_JCjyW/view?usp=sharing

Thực hiện đọc và chuyển đổi file nhạc yêu cầu thành chuỗi bit nhị phân làm nguồn tin đầu vào cho nhiệm vụ 2 và thực hiện ngược lại chuyển đổi chuỗi bit nhị phân thu được thành file nhạc.

2. Yêu cầu

a) Mô tả phương pháp thực hiện chuyển đổi file nhạc yêu cầu thành chuỗi nhị phân và ngược lại.

- Chuyển đổi file nhạc yêu cầu thành chuỗi bit nhị phân:
 - Xác định các thông số của file âm thanh cần chuyển đổi bằng hàm *audioinfo* trong thư viện Audio Processing Toolbox của MATLAB.
 - Đọc file âm thanh vào 1 mảng dữ liệu số bằng việc dùng hàm *audioread* trong thư viện Audio Processing Toolbox của MATLAB.
 - Chuyển đổi mảng dữ liệu âm thanh thu được thành mảng dữ liệu số và có kiểu dữ liệu giống với file âm thanh ban đầu (unit16).
 - Chuyển đổi mảng dữ liệu số thành chuỗi bit nhị phân bằng việc sử dụng hàm *dec2bin* trong MATLAB.
- Chuyển đổi chuỗi nhị phân thành file âm thanh.
 - Chuyển đổi chuỗi bit nhị phân thành các mảng số nguyên có các giá trị nhị phân [0,1].
 - Chuyển đổi mảng nhị phân này thành các mảng dữ liệu có kiểu (unit16) bằng cách sử dụng hàm *bi2de*.
 - Tiếp tục chuyển đổi mảng dữ liệu số này sang mảng dữ liệu âm thanh.
 - Ghi mảng dữ liệu âm thanh ra cùng với tần số cùng với tần số của file gốc để tạo thành 1 file âm thanh mới bằng cách sử dụng hàm *audiowrite* trong thư viện Audio Processing Toolbox của MATLAB.
 - Kiểm tra các thông số của file âm thanh mới có giống với file âm thanh gốc không bằng hàm *audioinfo* trong thư viện Audio Processing Toolbox của MATLAB.

b) Viết chương trình MATLAB.

Thực hiện chuyển đổi file nhạc *AssignmentD20.wav* thành chuỗi tín hiệu nhị phân và ngược lại. Có thể xây dựng dưới dạng hàm chuyển đổi để sử dụng trong các chương trình khác.

- Code:

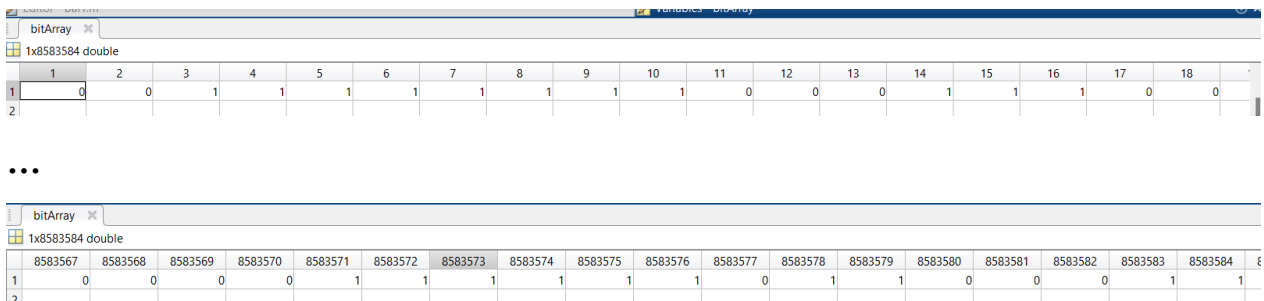
Hàm chuyển đổi file nhạc sang chuỗi bit nhị phân

```
function [samples,bitArray,fs,n,title,artist] = audioToBinary(fileName)
% Doc va luu du lieu am thanh vao mang
[samples, fs] = audioread(fileName);
% Doc cac thong so cua file am thanh goc
originInfo = audioinfo(fileName);
title = originInfo.Title;
artist = originInfo.Artist;
% Luu kích thước mà trên samples vừa đọc được từ am thanh gốc
[m, n] = size(samples);
samples = reshape(samples, [], 1);
% Chuyen doi du lieu am thanh sang du lieu so
% roi chuyen ve chuoit bit nhi phan co do dai 16 bit
bitString = dec2bin(round((samples + 1) * (2^15-1)/2), 16)';
% Vi samples co gia tri [-1,1] => samples + 1 dung de dua cac gia tri ve
cac so duong
bitArray = bitString(:)' - '0'; % Chuyen chuoit bit nhi phan sang vector chua
cac so 0 1
```

Hàm chuyển đổi bit nhị phân sang file nhạc mới

```
function newInfo = binaryToAudio(bitArray, fs, n, title, artist)
% Chuyển đổi chuỗi bit nhị phân thành dãy số nguyên kiểu unit 16
samples = bi2de(reshape(bitArray, 16, []), 'left-msb');
% Chuyển ngược dãy số nguyên thành giá trị âm thanh
% có dạng như mảng dữ liệu ban đầu khi đọc file âm thanh
samples = (samples/(2^15-1))*2-1;
% Tra về kích thước ban đầu của mảng dữ liệu samples
samples = reshape(samples, [], n);
% Ghi dữ liệu âm thanh vào file mới với các thông số từ file gốc
audiowrite('newfile.wav', samples, fs, 'Title', title, 'Artist', artist);
% Kiểm tra thông số của file âm thanh mới so với file gốc
newInfo = audioinfo('newfile.wav');
```

Kết quả sau khi thực hiện hàm *audioToBinary* và *binaryToAudio*:



...

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0
2																		

...

	8583567	8583568	8583569	8583570	8583571	8583572	8583573	8583574	8583575	8583576	8583577	8583578	8583579	8583580	8583581	8583582	8583583	8583584
1	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0	1	1
2																		

110011000101100001000100100011010000000100010000100000011001011111000000110000100110111010100101100101100100111000111000110011001100110011110001100110001

c) **Xác định các thông số về file nhạc cần chuyển đổi.**

Một số hàm công cụ tiện ích trong thư viện Audio Processing Toolbox có thể cần sử dụng để thực hiện nhiệm vụ (để biết chi tiết hơn về từng câu lệnh, sử dụng help):

audioread – đọc dữ liệu từ file nhạc.

audiowrite – ghi file nhạc.

audioinfo – thông tin về file nhạc.

unit16 – chuyển đổi kiểu dữ liệu về unit16.

Sử dụng lệnh *originInfo = audioinfo('AssignmentD20.wav');* ta được các thông số của file nhạc cần chuyển đổi:

```
originInfo =
```

```
struct with fields:
```

```
    Filename: 'D:\User\Downloads\Documents\KI6\MoPhongHeThongTruyenThong\Assignment\AssignmentD20.wav'
CompressionMethod: 'Uncompressed'
    NumChannels: 2
    SampleRate: 8000
    TotalSamples: 268237
    Duration: 33.5296
        Title: 'Impact Moderato'
    Comment: []
    Artist: 'Kevin MacLeod'
    BitsPerSample: 16
```

- *Filename*: Tên và đường dẫn của file mới.
- *CompressionMethod*: Phương pháp nén file, ở đây là "Uncompressed" tức không nén.
- *NumChannels*: Số kênh của file, ở đây là 2 kênh (đã được ghi rõ trong mã code MATLAB).
- *SampleRate*: Tốc độ lấy mẫu của file, ở đây là 8000 Hz (đã được ghi rõ trong mã code MATLAB).
- *TotalSamples*: Tổng số mẫu trong file, ở đây là 268237 mẫu.
- *Duration*: Thời lượng của file, ở đây là 33.5296 giây.
- *Title*: Tiêu đề của file, ở đây là "Impact Moderato".
- *Comment*: Chú thích (nếu có) của file.
- *Artist*: Tác giả của file, ở đây là "Kevin MacLeod".
- *BitsPerSample*: Số bit trên mẫu âm thanh của file, ở đây là 16 bit.

Nhiệm vụ 2

1. Mô tả nhiệm vụ:

Mô phỏng hệ thống truyền dẫn số tại tốc độ dữ liệu N Mb/s, với N là số cuối cùng của mã số sinh viên (nếu số đó là 0 thì sẽ lựa chọn số liền kề bên cạnh). Nguồn tin của hệ thống được lấy từ file nhạc thực hiện trong nhiệm vụ 1, trong trường hợp không thực hiện lấy nguồn tin từ nhiệm vụ 1 được hãy thay thế bằng một chuỗi tín hiệu nhị phân ngẫu nhiên tương đương. Mỗi sinh viên sẽ lựa chọn một trong các kỹ thuật điều chế sau cho hệ thống mô phỏng của mình:

- Điều chế M-QAM nếu số cuối cùng trong mã sinh viên là lẻ, với $M = 16$ nếu số liền kề là lẻ và $M = 4$ nếu số liền kề là chẵn.
 - Điều chế M-DPSK nếu số cuối cùng trong mã sinh viên là chẵn, với $M = 4$ nếu số liền kề là lẻ và $M = 2$ nếu số liền kề là chẵn.
- Sử dụng mô hình mô phỏng tương đương băng gốc, tín hiệu phát có thể được biểu diễn như sau:

$$s(t) = \left[\sum_{k=-\infty}^{\infty} d_k p(t - kT_{sym}) \right] e^{j\Phi_0}$$

trong đó d_k là các kí hiệu (symbol) phức được xác định từ chuỗi bản tin đầu vào và kỹ thuật điều chế; T_{sym} là chu kỳ của symbol; Φ_0 là pha của tín hiệu phát và $p(t)$ xác định dạng xung được phát, với:

$$p(t) = \sqrt{\frac{2E_s}{T_{sym}}} \left[1 - \cos\left(\frac{2\pi t}{T_{sym}}\right) \right] \quad \text{cho tín hiệu M-DPSK}$$

$$p(t) = \sqrt{\frac{2E_s}{T_{sym}}} \left[1 - \sin\left(\frac{2\pi t}{T_{sym}}\right) \right] \quad \text{cho tín hiệu M-QAM}$$

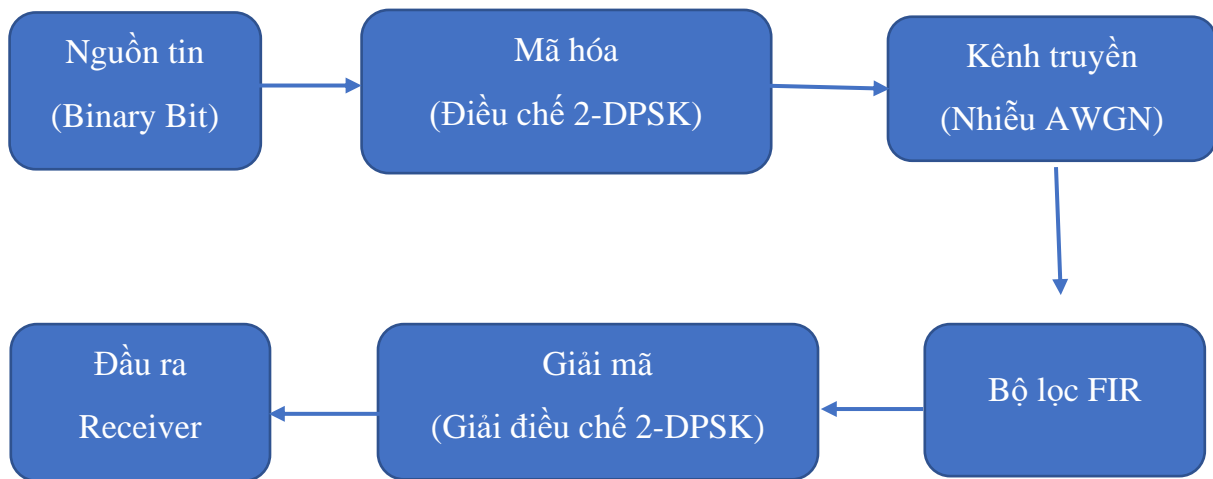
với E_s là năng lượng mỗi symbol.

⇒ **Hệ thống 2-DPSK với $N = 6$ Mb/s.**

2. Yêu cầu.

a) Mô tả hệ thống mô phỏng bằng sơ đồ khối và xác định các tham số của hệ thống, có thể bao gồm cả các bộ lọc sử dụng để có được bộ thu tối ưu.

- Sơ đồ khối hệ thống 2-DPSK:



Sử dụng bộ điều chế 2-DPSK. Các chuỗi bit đầu vào sẽ được tạo thành các xung, sau đó các xung này sẽ được đưa vào bộ điều chế 2-DPSK và được đưa lên kênh AWGN để tới được máy thu. Tại máy thu các tín hiệu sẽ được đưa qua bộ lọc và giải điều chế để khôi phục chuỗi bit ban đầu.

- Xác định tham số của hệ thống 2-DPSK:

- Tốc độ dữ liệu $R_b = 6 \text{ Mb/s}$.
- Năng lượng symbol $E_s = 10 \text{ J}$.
- Tần số lấy mẫu $F_s = 300 \text{ KHz}$.
- Tần số sóng mang $F_c = 10 \text{ KHz}$.
- $\text{SNR} = 5 \text{ dB}$.
- BER tại các mức SNR $(E_s/N_0) = [2 \ 5 \ 10] \text{ dB}$.
- Pha ban đầu của tín hiệu $\phi_0 = 0$.

- b) Bằng việc sử dụng MATLAB, viết chương trình mô phỏng hệ thống truyền dẫn số sử dụng kỹ thuật điều chế đã lựa chọn trên kênh AWGN với nguồn tín hiệu là tín hiệu thu được ở nhiệm vụ 1. Ước tính xác suất lỗi tại các mức tỉ số tín hiệu trên nhiễu SNR lần lượt bằng 2, 5 và 10 dB theo phương pháp Monte Carlo.

Code:

```
%% -----Nguyen Di Dan - B20DCVT086 (2-DPSK N= 6MB/s) -----
clear,clc,close all
M = 2;    % Số mức điều chế
Es = 10;   % Năng lượng cho một ký hiệu
Rb=6e6;    % Tốc độ bit- bit rate
Nsym=Rb/log2(M); % Tốc độ ký hiệu- Symbol rate
Tsym=1/Nsym;% Chu kỳ ký hiệu
n = 1e4;   % Số bit đầu vào
phi=0;     % Pha tín hiệu phát
fs=3e5;    % Tần số lấy mẫu
fc=1e4;    % Tần số sóng mang
ts=1/fs;   % Chu kỳ lấy mẫu
bf = 7e6;  % Băng thông bỏ lọc
%% Nhập tín hiệu phát
bit = randi([0 1], 1, n); % Chuỗi bit đầu vào ngẫu nhiên
dec = bi2de(reshape(bit,log2(M),length(bit)/log2(M)).','left-msb'); %
Chuyển đổi nhị phân sang thập phân.
d = dpskmod(dec,M);    % Điều chế 2-DPSK
%% Xây dựng hàm s(t)
t=0:ts:n/log2(M)*Tsym;
for i = 1:length(t)
    s(i) = 0;
    for k = 1:n/log2(M)
        pt(k) = sqrt((2*Es)/Tsym)*((1-cos(2*pi*t(i)))/Tsym);
        s(i) = s(i) + d(k)*pt(k);
    end
end
%% Đồ thị tín hiệu băng gốc
plot(t,s);
```

```

xlabel(' Thoi gian (s)');
ylabel( ' Bien do ');
title(' Do thi tin hieu goc ');
signalA=s.*exp(1i*phi); %tin hieu sau dieu che
signalE=signalA.*exp(1i*2*pi*fc*t); %Tin hieu de xac dinh mau mat
%% Do thi va pho cua tin hieu sau dieu che 2-DPSK
figure
subplot(2,1,1);
plot(t,real(signalE))
xlabel(' Thoi gian (s)');
ylabel( ' Bien do ');
title('Do thi tin hieu sau dieu che');
subplot(212);
spectrum(signalE,fs);
title('Pho cua tin hieu sau dieu che');
%% Mau mat cua tin hieu sau khi dieu che 2-DPSK
eyediagram(real(signalE),10);
title(' Bieu do mat tin hieu 2-DPSK');
%% Chom sao cua tin hieu sau khi dieu che
scatterplot(d,1,0,'or');
title(' Bieu do chom sao cua tin hieu sau dieu che ' );
%% Do thi va pho cua tin hieu sau khi qua AWGN
figure
signal_noise=addNoiseAWGN(signalE,5);
subplot(211);
plot(t, real(signal_noise));
xlabel(' time s'); ylabel( ' Bien do ');
title(' Do thi khi bi AWGN ');
subplot(212);
spectrum(signal_noise,fs);
title('Pho cua tin hieu bi AWGN');
%% Mau mat cua tin hieu sau khi qua kenh AWGN
eyediagram(real(signal_noise),10);
title('Bieu do mat tin hieu 2-DPSK khi qua AWGN');

```

```

%% Chom sao cua tin hieu sau khi qua kenh AWGN va so sanh voi chom
sao goc
d_noise = addNoiseAWGN(d,5);
h = scatterplot(d_noise,1,0,'x');
hold on;
scatterplot(d,1,0,'or',h);
title('Bieu do chom sao cua tin hieu sau khi qua kenh AWGN');
%% Xu ly va khoi phuc tai bo thu
figure
source1 = signal_noise.*exp(-1i*phi).*exp(-1i*2*pi*fc*t); %Tin hieu thu
truoc bo loc
source = raisedCosFilter(source1,bf,ts,2); % Khoi phuc lai tin hieu
plot(t,real(source)) % Do thi tin hieu sau khi khoi phuc va pho cua no
xlabel('Thoi gian(s)');
ylabel('Bien do');
title('Do thi tin hieu khoi phuc');
figure
spectrum(source,fs);
title('Pho cua tin hieu duoc khoi phuc');
%% Chom sao tin hieu duoc khoi phuc
h = scatterplot(source,1,0,'xb');
title('Bieu do chom sao cua tin hieu duoc khoi phuc');
%% Mau mat tin hieu sau khi khoi phuc
eyediagram(real(source),10);
title('Mau mat duoc khoi phuc');
%% Tinh BER bang phuong phap Monte-Carlo
SNR_dB=[2 5 10] ;
%=====vong lap=====
for i=1:length(SNR_dB)
    SNR=exp(SNR_dB(i)*log(10)/10); %Chuyen dB sang so lan
    theoryBer(i)=2*qfunc(sqrt(SNR)); %Tinh BER theo ly thuyet
    simBer(i)=monteCarlo(SNR_dB(i), n, bit); %Tinh BER theo thuc nghiem
end
% Uoc tinh xac suat loi tai BER = [ 2 5 10 ]
disp('Ket qua BER tai SNR lan luot la 2 5 va 10dB la')

```

```

simBer = single(simBer) %Lam tron 5 so.
%Do thi
semilogy(SNR_dB,theoryBer,'LineWidth',2);hold on;
semilogy(SNR_dB,simBer,'r-*','LineWidth',2);grid on;
xlabel('SNR (dB)'); ylabel('BER');
title('Truyen tin hieu DPSK qua kenh AWGN');
legend('BER_l_y_t_h_u_y_e_t','BER_m_o_p_h_o_n_g');

%% ===== FUNCTION =====

%% ===== Ham them kenh nhieu AWGN ===== %%
function yNoise = addNoiseAWGN(y,SNR_dB)
% y - Tin hieu dau vao
% SNRdB - Muc SNR dB
% yNoise – Tin hieu nhieu dau ra
SNR = 10^(SNR_dB/10);
VarN = var(y)/SNR;
if (isreal(y))
    yNoise = y + sqrt(VarN)*randn(size(y));
else
    yNoise = y + sqrt(VarN/2)*(randn(size(y))+1i*randn(size(y)));
end
end

%% ===== Ham them kenh nhieu Gauss ===== %%
function yNoise = addNoiseGauss(y, SNR_dB)
% y - Tin hieu dau vao
% SNR_db - Ty le tin hieu nhieu dB
% yNoise - Tin hieu dau ra qua nhieu
    SNR_lin = 10^(SNR_dB/10);
    noise = randn(size(y));
    power_signal = mean(abs(y).^2);
    power_noise = power_signal/SNR_lin;
    noise = noise*sqrt(power_noise);
    yNoise = y + noise;
end

```

```

%% ===== Ham xu ly bo loc (Dung bo loc Raised Cosine ===== %%
function y = raisedCosFilter(x,bf,Ts,beta)
% x - dau vao
% bf - bang thong cua bo loc
% Ts - chu ky lay mau
% beta - he so giam doc (rolloff factor)
% y - dau ra bo loc
Ns = length(x);
Tb = 1/bf;
beta = beta*bf;
% Mien tan so
f = [0:Ns/2-1 -Ns/2:-1]/(Ns*Ts);
Xf = fft(x);
Yf = zeros(size(Xf));
ind = (abs(f) <= (bf/2-beta));
Yf(ind) = Xf(ind).*Tb;
ind = (abs(f) <= (bf/2+beta) & abs(f) > (bf/2-beta));
Yf(ind) = Xf(ind).*(Tb*cos(pi/(4*beta)*(abs(f(ind))-bf/2+beta)).^2);
ind = (abs(f) > (bf/2+beta));
Yf(ind) = Xf(ind).*0;
% Chuyen doi sang mien thoi gian
y = ifft(Yf)./Tb;
end

%% ===== Ham tinh BER bang Monte Carlo ===== %%
function berMonteCarlo = monteCarlo(SNR_dB, n, bit)
% SNR_dB: Ty le tin hieu nhieu
% n: So lan lap danh gia hieu nang
% berMonteCarlo: BER trung binh dua tren ky thuat Monte Carlo
M = 2; % 2-DPSK
% BER trung binh
berSum = 0;
for i = 1:n
    % Tin hieu dieu che
    bitmodulated = dpskmod(bit, M);
    % Tao ra tin hieu qua nhieu Gauss

```

```

signal_noise = addNoiseGauss(bitmodulated, SNR_dB);
% Giai dieu che tin hieu
bitdemodulated = dpskdemod(signal_noise, M);
% Tinh so luong bit sai khac giua bit ban dau va bit sau giai dieu che
num_errors = sum(bit ~= bitdemodulated);
% Tinh loi bit
berSum = berSum + num_errors/n;
end
berMonteCarlo = berSum/n;
end

%% ===== Ham ve pho tin hieu
===== %%
function st_fft_fre = spectrum(x,fs)
% pho cua tin hieu x
% fs - tan so lay mau
st_fft = fft(x);
st_fft = fftshift(st_fft);
st_fft_fre = fs/2*linspace(-1, 1, length(st_fft));
plot(st_fft_fre, abs(st_fft));
grid;
xlabel('Tan so (Hz)');
ylabel('Bien do');
end

```

Ước tính xác suất lỗi tại các mức tỉ số tín hiệu trên nhiễu SNR lần lượt bằng 2, 5 và 10 dB theo phương pháp Monte Carlo.

Ket qua BER tại SNR lan luot la 2 5 va 10dB la

simBer =

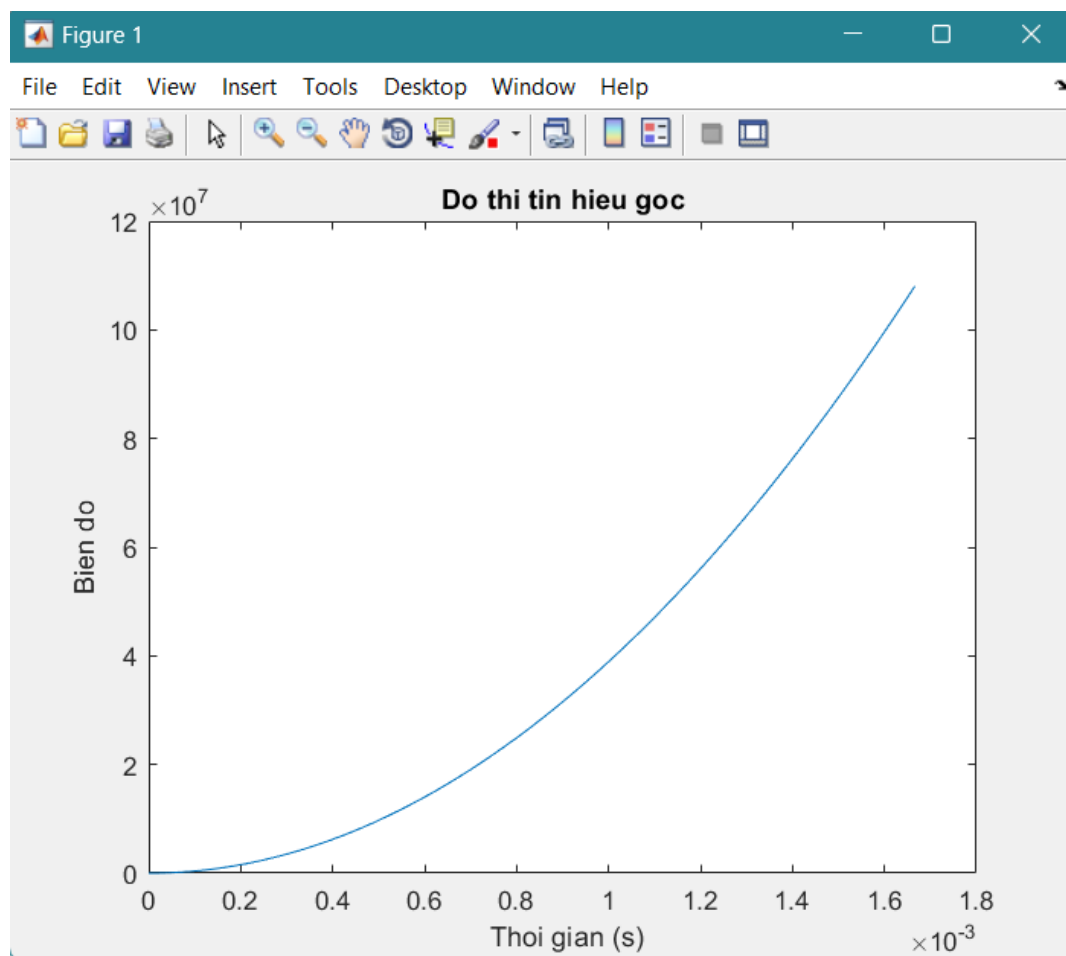
1×3 single row vector

0.1864 0.0726 0.0016

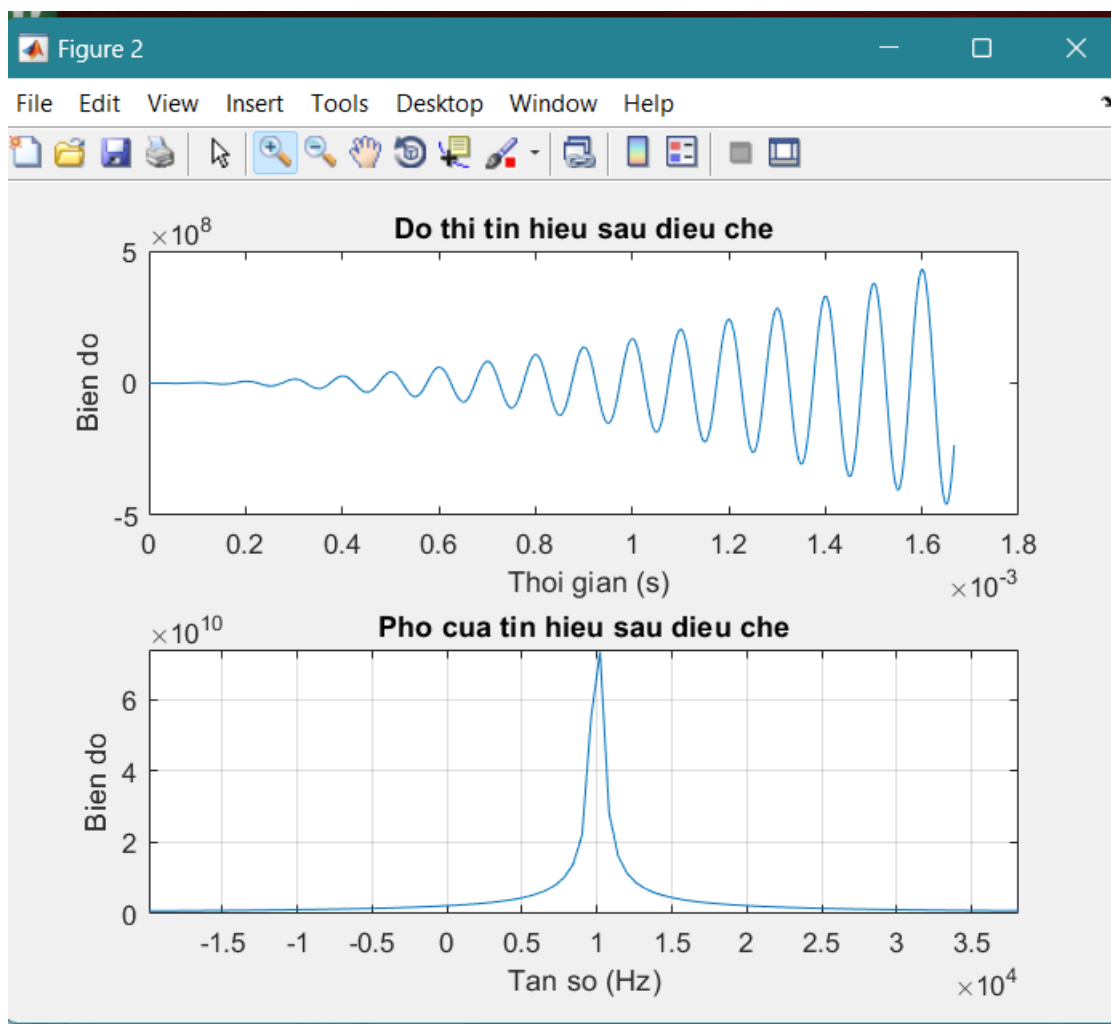
simBer				
1x3 single				
	1	2	3	4
1	0.1864	0.0726	0.0016	
2				

c) Biểu diễn biểu đồ chòm sao, dạng sóng tín hiệu, mẫu mắt và phổ của tín hiệu tại các điểm sau trên hệ thống: đầu ra bộ điều chế, sau khi truyền qua kênh AWGN tại SNR = 5dB, sau khi được xử lý và khôi phục tại bộ thu.

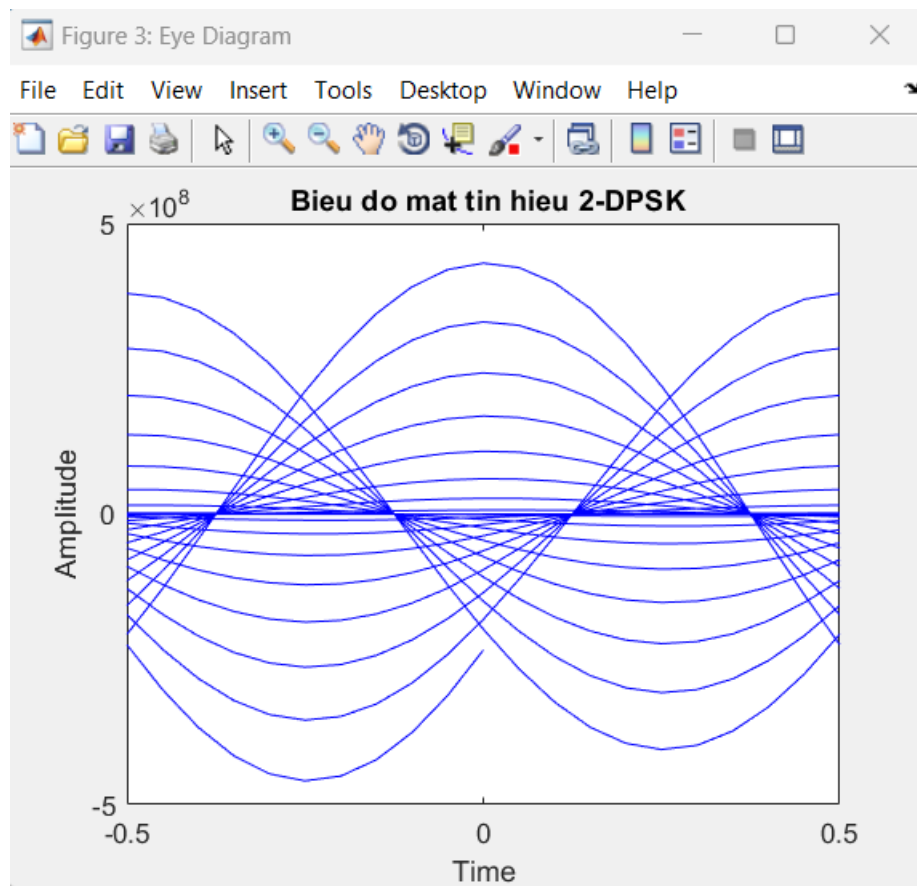
Kết quả mô phỏng tại SNR = 5dB:



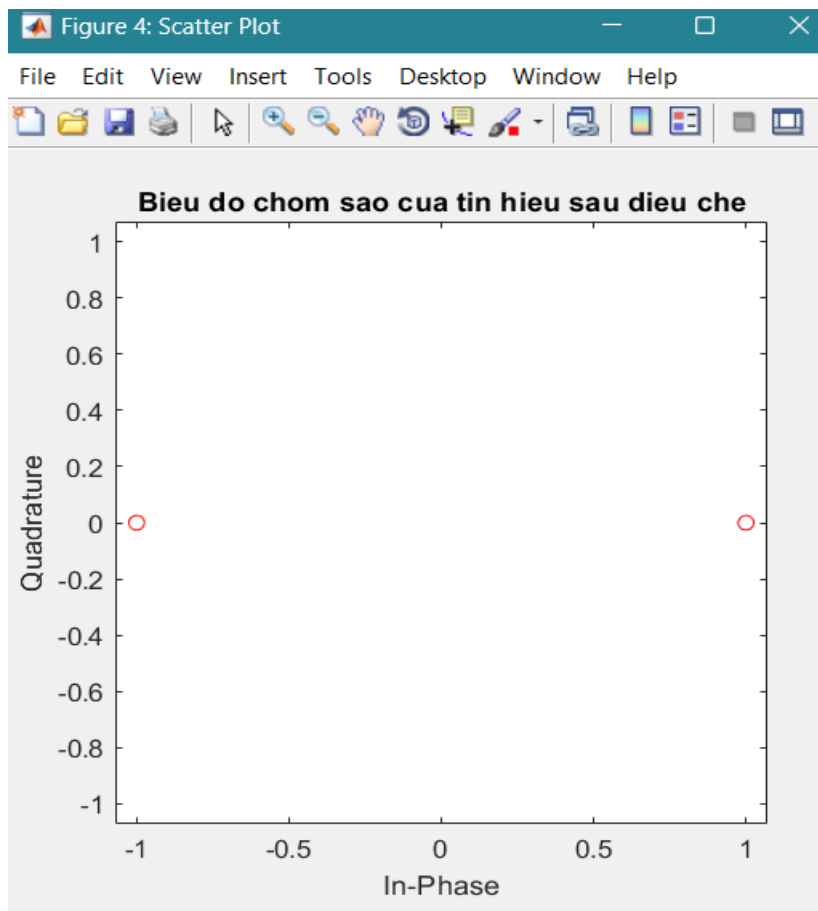
Hình 1. Tín hiệu băng gốc



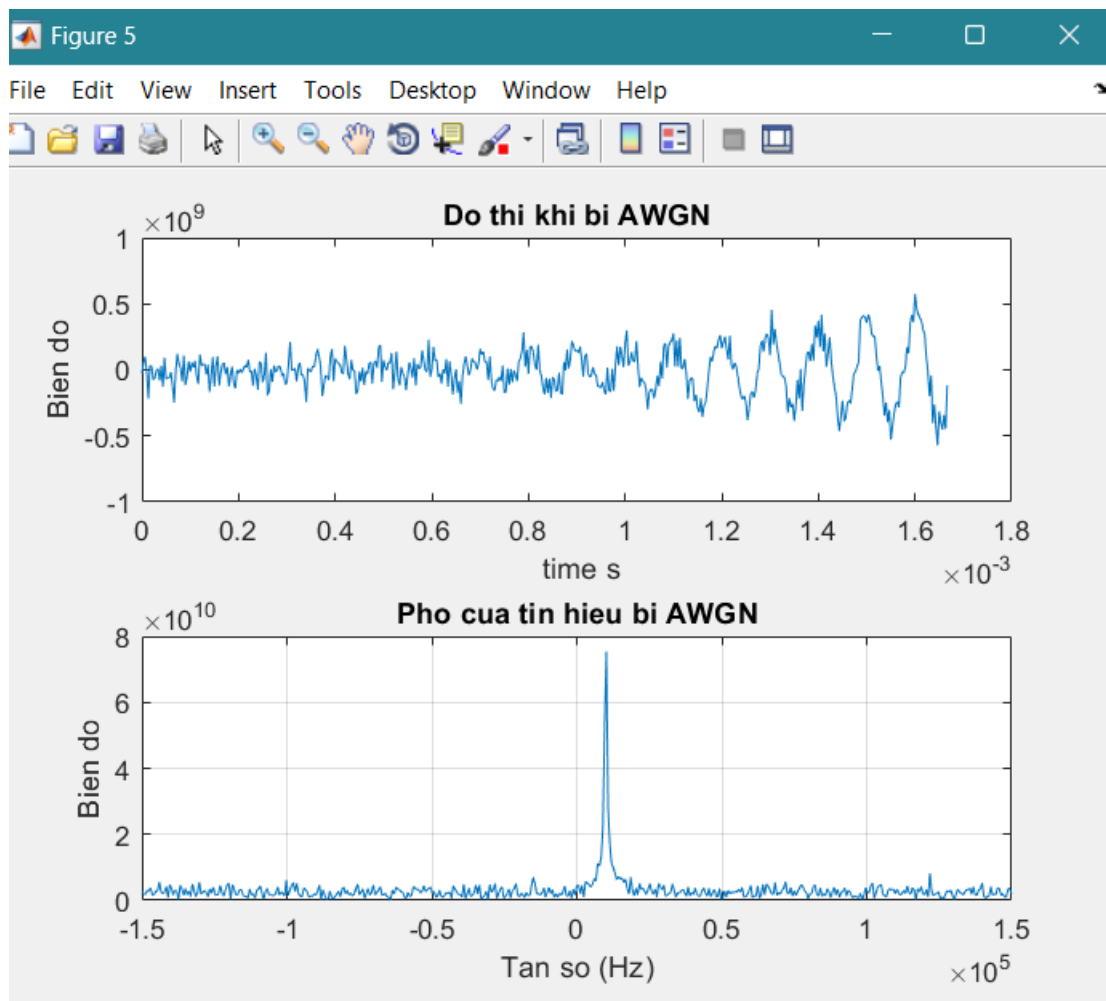
Hình 2. Tín hiệu và phổ của tín hiệu sau điều chế



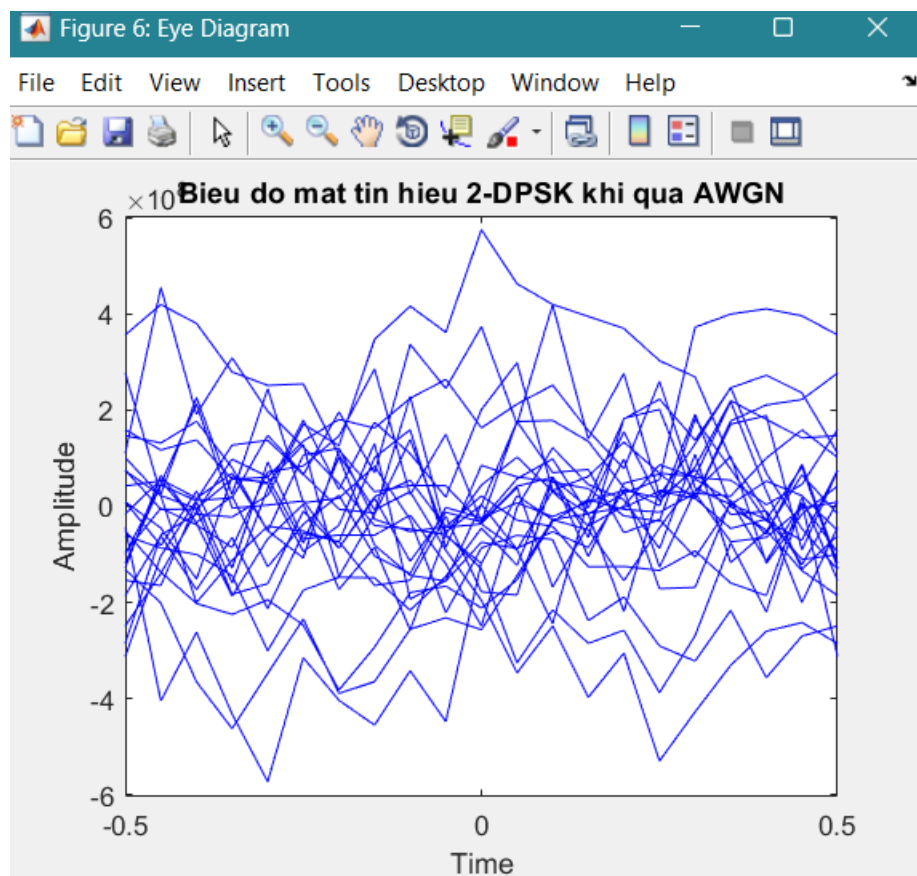
Hình 3. Biểu đồ mắt của tín hiệu sau điều chế



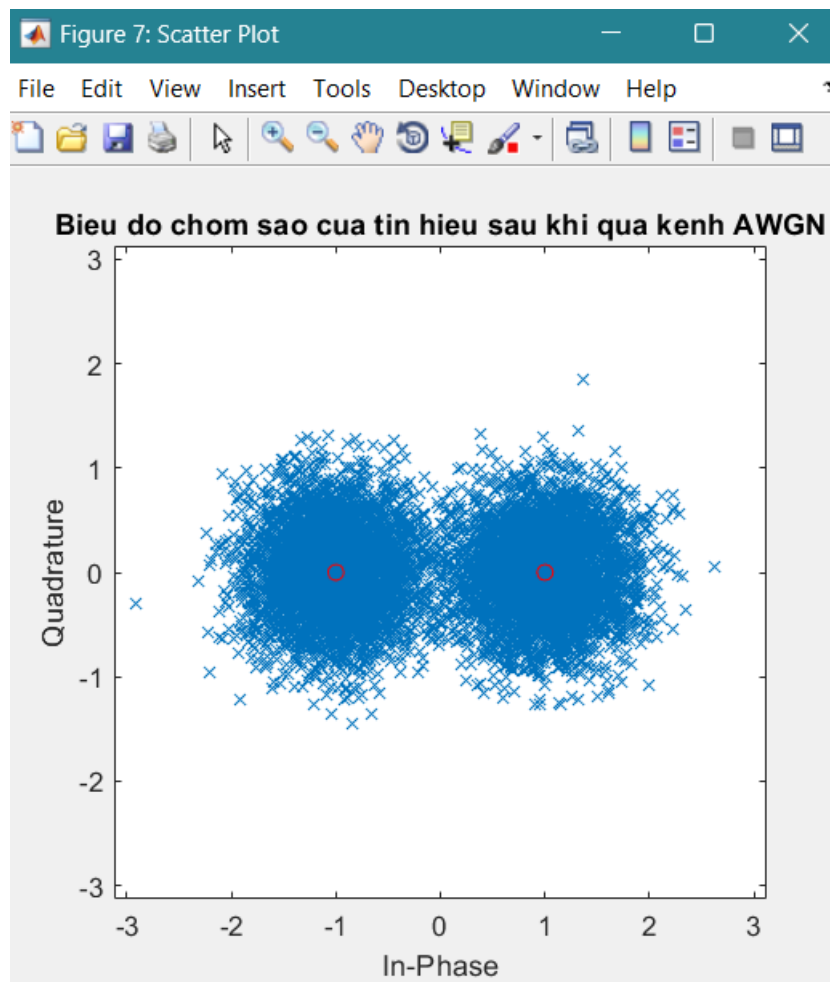
Hình 4. Chòm sao của tín hiệu sau điều chế



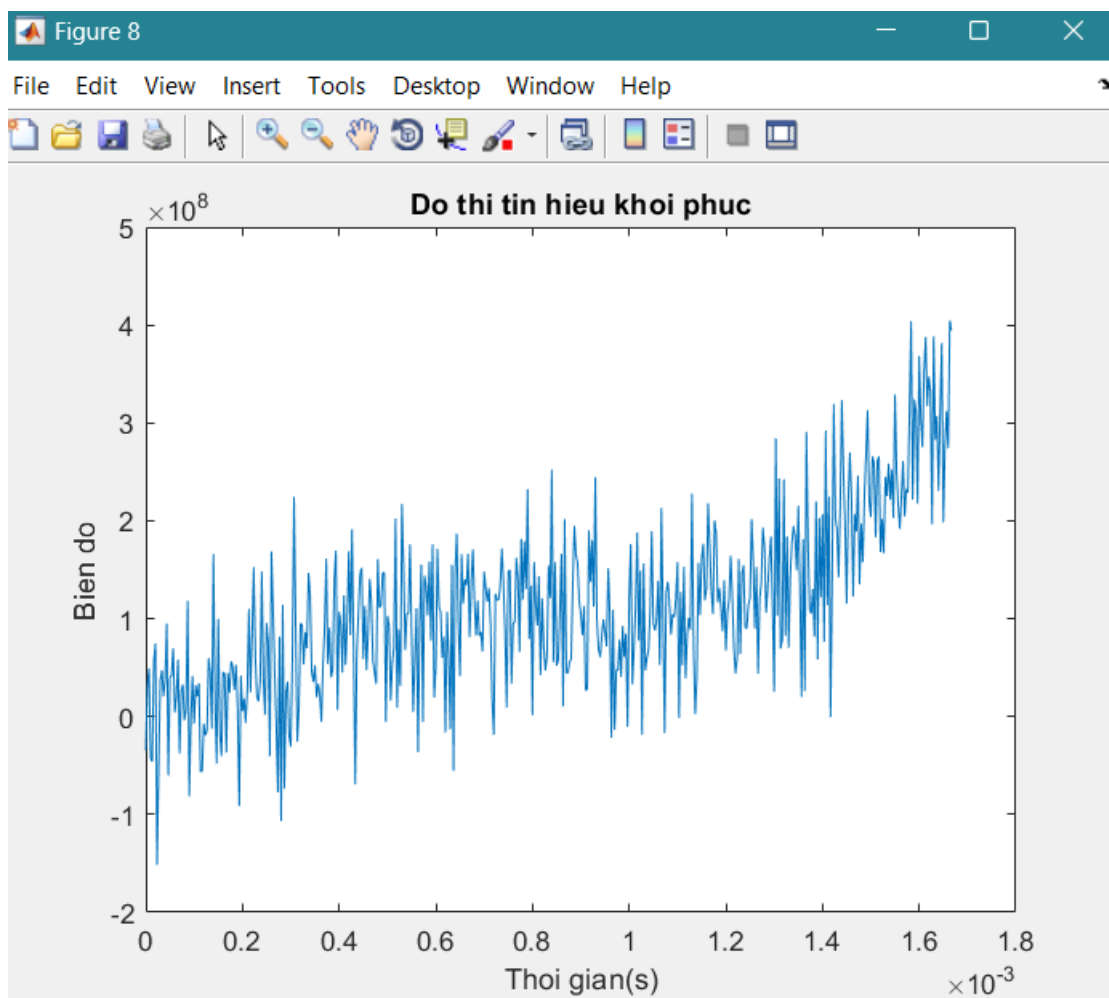
Hình 5. Tín hiệu và phổ của nó sau khi qua AWGN



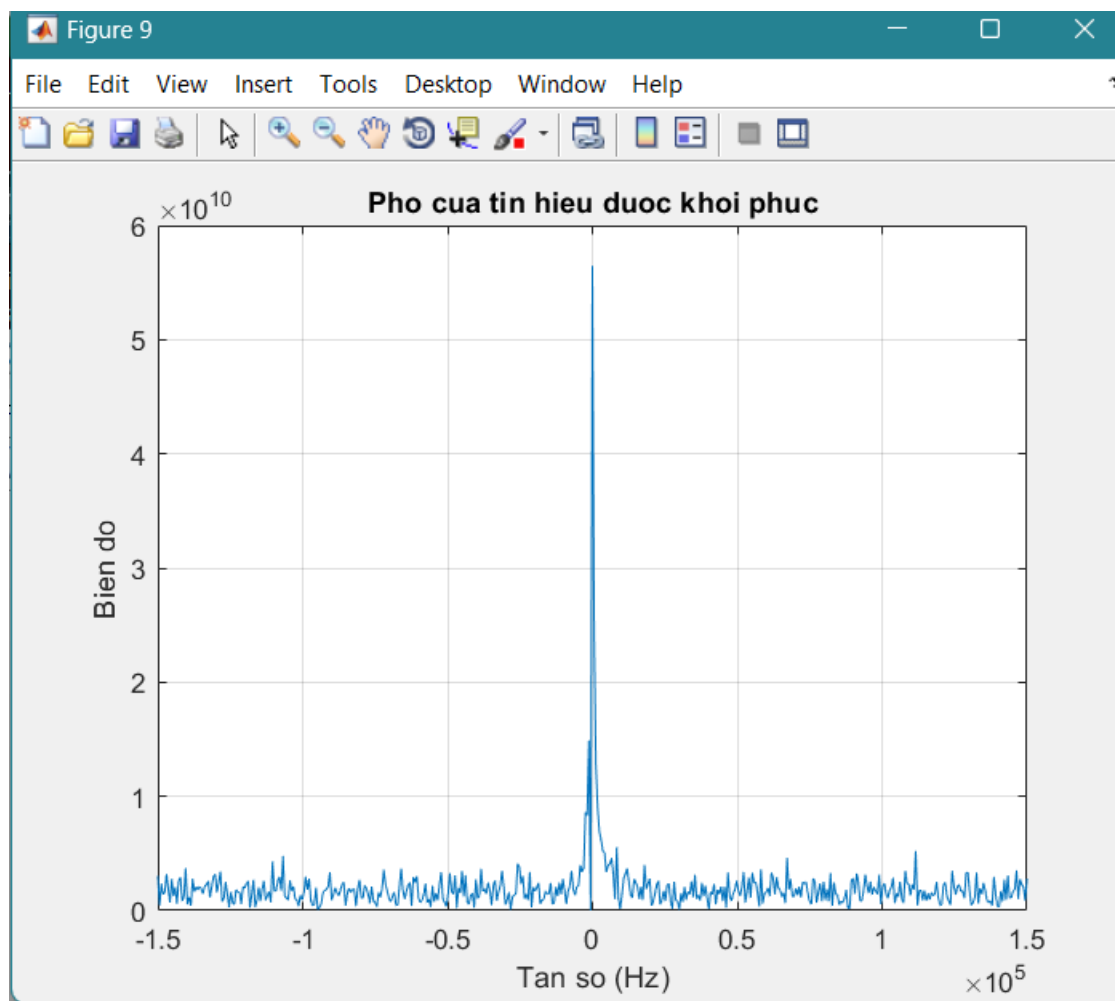
Hình 6. Biểu đồ mắt của tín hiệu sau khi qua kênh AWGN



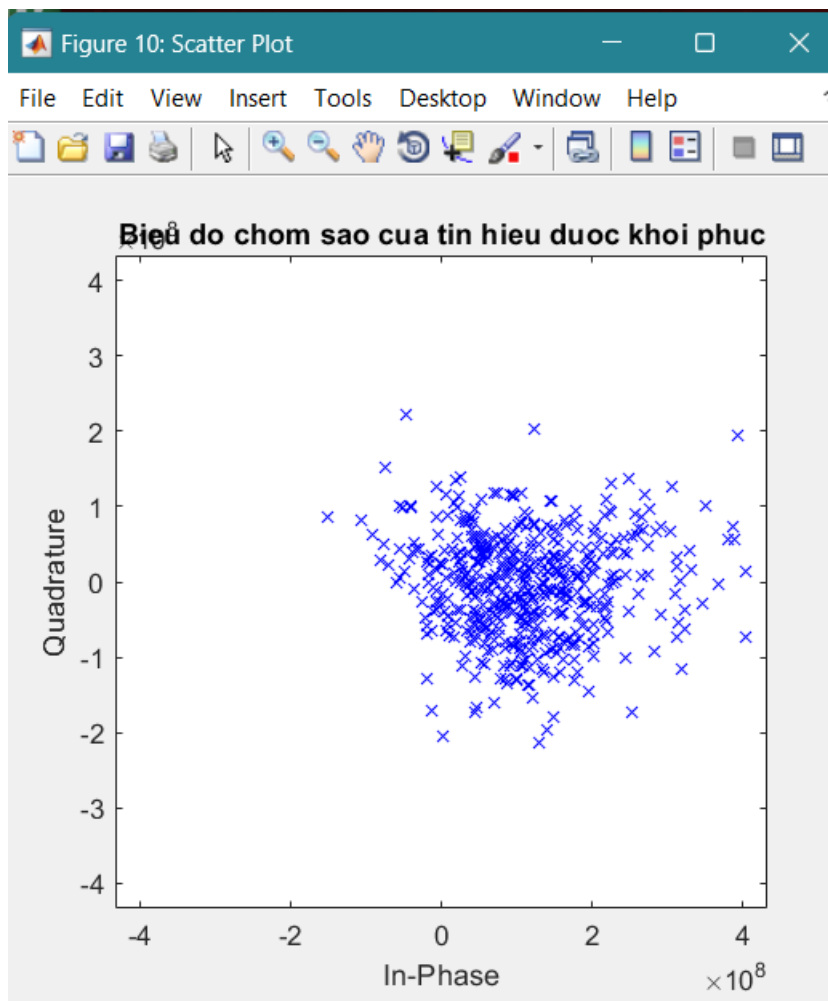
Hình 7. Chòm sao của tín hiệu sau khi qua kênh AWGN



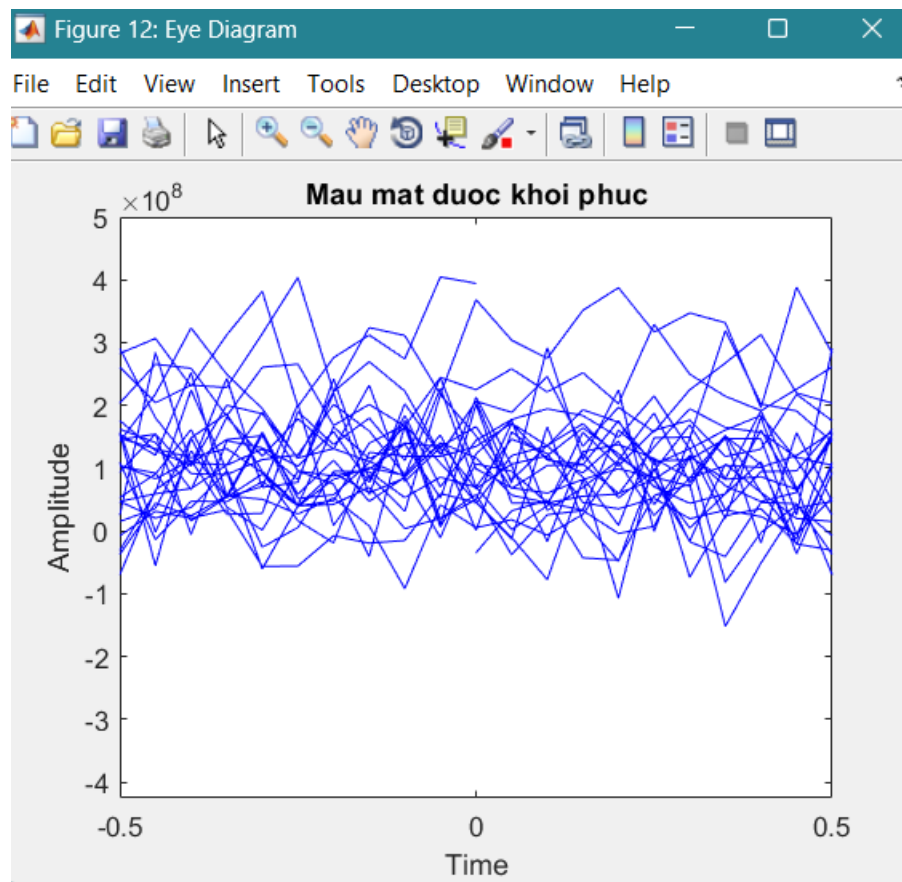
Hình 8. Tín hiệu sau khi khôi phục



Hình 9. Phổ của tín hiệu sau khi khôi phục

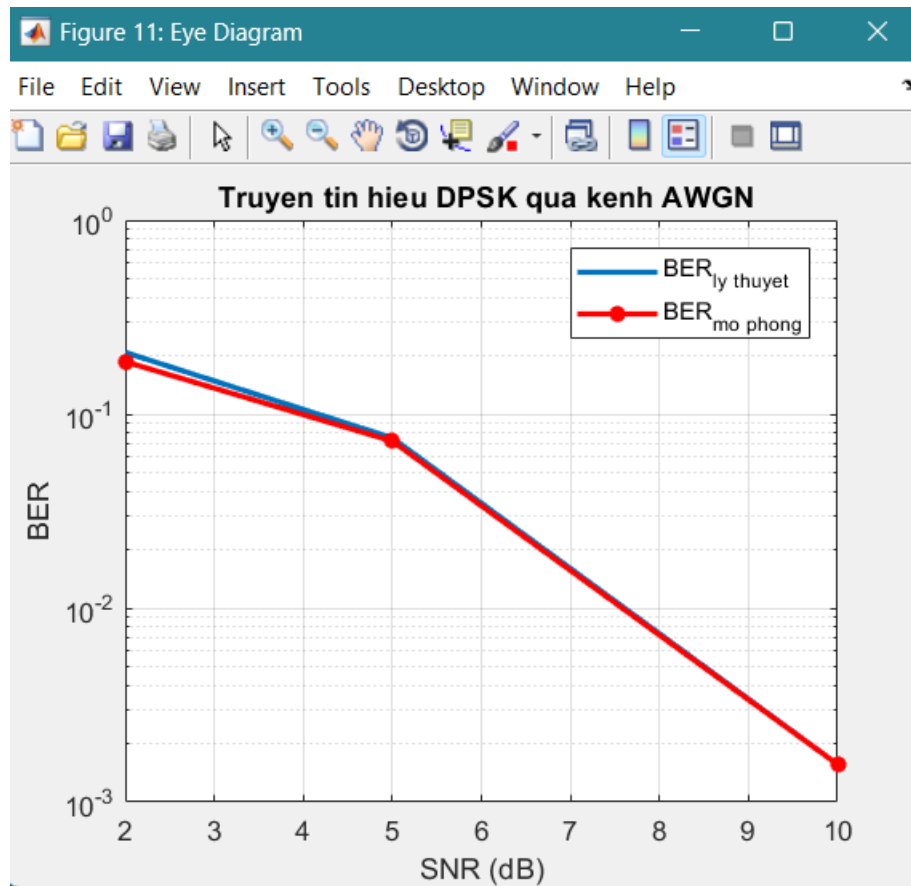


Hình 10. Chòm sao của tín hiệu sau khi khôi phục



Hình 11. Biểu đồ mẫu mắt sau khi khôi phục

- c) So sánh file nhạc được khôi phục sau khi truyền qua hệ thống mô phỏng tại các mức SNR yêu cầu.



Hình 12. So sánh BER mô phỏng bằng Monte Carlo với BER được tính toán theo lý thuyết

End