|  |
| --- |
| HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  KHOA VIỄN THÔNG 1  ----------                        **BÀI TẬP LỚN**  **MÔ PHỎNG HỆ THỐNG TRUYỀN THÔNG**  **Hoàng Đông Tuấn-B20DCVT334**  **Nhóm môn học:06**  **Giảng viên bộ môn:N.T.T.Trang**    **HÀ NỘI** |

MỤC LỤC

[Nhiệm vụ 1 3](#_Toc134174564)

[1. Mô tả nhiệm vụ. 3](#_Toc134174565)

[2. Yêu cầu 3](#_Toc134174566)

[a.Mô tả phương pháp thực hiện chuyển đổi file nhạc yêu cầu thành chuỗi nhị phân và ngược lại. 3](#_Toc134174567)

[b. Chương trình MATLAB thực hiện chuyển đổi file nhạc AssigmentD20.wav thành chuỗi tín hiệu nhị phân và ngược lại: 3](#_Toc134174568)

[c. Xác định các thông số về file nhạc cần chuyển đổi: 5](#_Toc134174569)

[Nhiệm vụ 2 6](#_Toc134174570)

[1. Mô tả nhiệm vụ: 6](#_Toc134174571)

[2. Yêu cầu. 7](#_Toc134174572)

[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. 7](#_Toc134174573)

[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. 8](#_Toc134174576)

[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. 14](#_Toc134174577)

[d.So sánh file nhạc được khôi phục sau khi truyền qua hệ thống mô phỏng tại các mức SNR yêu cầu. 19](#_Toc134174578)

# **Nhiệm vụ 1**

1. **Mô tả nhiệm vụ.**

Đọc và xử lý nguồn tin là một file nhạc AssigmentD20.wav. File này có thể lấy về theo địa chỉ dưới đây:

<https://drive.google.com/file/d/1fown3-91kGSUhlZA6L8eIjS1Oh_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.

1. **Yêu cầu**

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

Phương pháp thực hiện chuyển đổi file nhạc yêu cầu thành chuỗi nhị phân và ngược lại như sau:

1. Chuyển đổi file nhạc thành chuỗi nhị phân:
2. Sử dụng hàm **audioread** để đọc dữ liệu âm thanh từ file nhạc và lấy thông tin về tần số lấy mẫu (Fs).
3. Chuyển đổi giá trị âm thanh về giá trị số nguyên dương trong khoảng từ 0 đến 2^16-1 bằng cách cộng thêm 1, nhân với 2^15-1 và làm tròn giá trị.
4. Sử dụng hàm **dec2bin** để chuyển đổi các giá trị số nguyên thành chuỗi bit nhị phân có độ dài 16 bit.
5. Gộp các chuỗi bit nhị phân thành một chuỗi duy nhất bằng cách chuyển ma trận các chuỗi bit nhị phân thành một vector và chuyển vị.

Chuyển đổi chuỗi nhị phân thành file nhạc:

1. Tách chuỗi nhị phân thành các chuỗi con có độ dài 16 bit bằng cách sử dụng hàm **reshape**.
2. Chuyển đổi các chuỗi con thành các giá trị số nguyên bằng hàm **bin2dec**.
3. Chuyển đổi các giá trị số nguyên về dạng giá trị âm thanh bằng cách chia cho 32767.
4. Chuyển đổi dữ liệu âm thanh về dạng số nguyên bằng cách nhân với 2^15-1 và chuyển về kiểu dữ liệu **int16**.
5. Sử dụng hàm **audiowrite** để ghi dữ liệu âm thanh vào file mới.

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. Chương trình MATLAB thực hiện chuyển đổi file nhạc AssigmentD20.wav thành chuỗi tín hiệu nhị phân và ngược lại:**

***Code***

function audio\_conversion()

function binary = audioToBinary(filename)

% Chuyen doi file nhac thanh chuoi nhi phan

[y, Fs] = audioread(filename); % Doc du lieu tu file nhac

y = round((y + 1) \* (2^15-1)); % Bien doi gia tri y thanh gia tri nguyen duong trong khoang tu 0 den 2^16-1

binary = dec2bin(y, 16); % Chuyen doi cac gia tri nguyen thanh chuoi bit nhi phan do dai 16 bit

binary = binary.'; % Chuyen ve ma tran de ghep cac chuoi bit nhi phan

binary = binary(:).'; % Ghep cac chuoi bit nhi phan lai thanh mot chuoi duy nhat

end

function audio = binaryToAudio(binary, Fs)

% Chuyen lai chuoi nhi phân thành file nhac

binary = reshape(binary, 16, []).'; % tách chuoi nhi phân thành các chuoi bit nhi phân con do dài 16 bit

y = bin2dec(binary); % chuyen các chuoi bit nhi phân con thành các so nguyên

y = double(y) / 32767; % chuyen doi các so nguyên thành du lieu âm thanh

audio = int16(y \* (2^15 - 1)); % chuyen doi du lieu âm thanh sang dang so nguyên

audiowrite('output.wav', audio, Fs); % ghi du lieu âm thanh ra file

end

% Xác dinh thông tin ve file nhac

info = audioinfo('AssigmentD20.wav');

disp(info);

info\_output = audioinfo('output.wav');

disp(info\_output);

% Chuyen doi file nhac thành chuoi nhi phân

binary = audioToBinary('AssigmentD20.wav');

fid = fopen('binary\_data.bin', 'wb');

fwrite(fid, binary, 'char');

fclose(fid);

% Chuyen doi chuoi nhi phân thành file nhac và luu lai vào file 'output.wav'

audio = binaryToAudio(binary, info.SampleRate);

fid = fopen('audio\_data.bin', 'wb');

fwrite(fid, audio, 'int16');

fclose(fid);

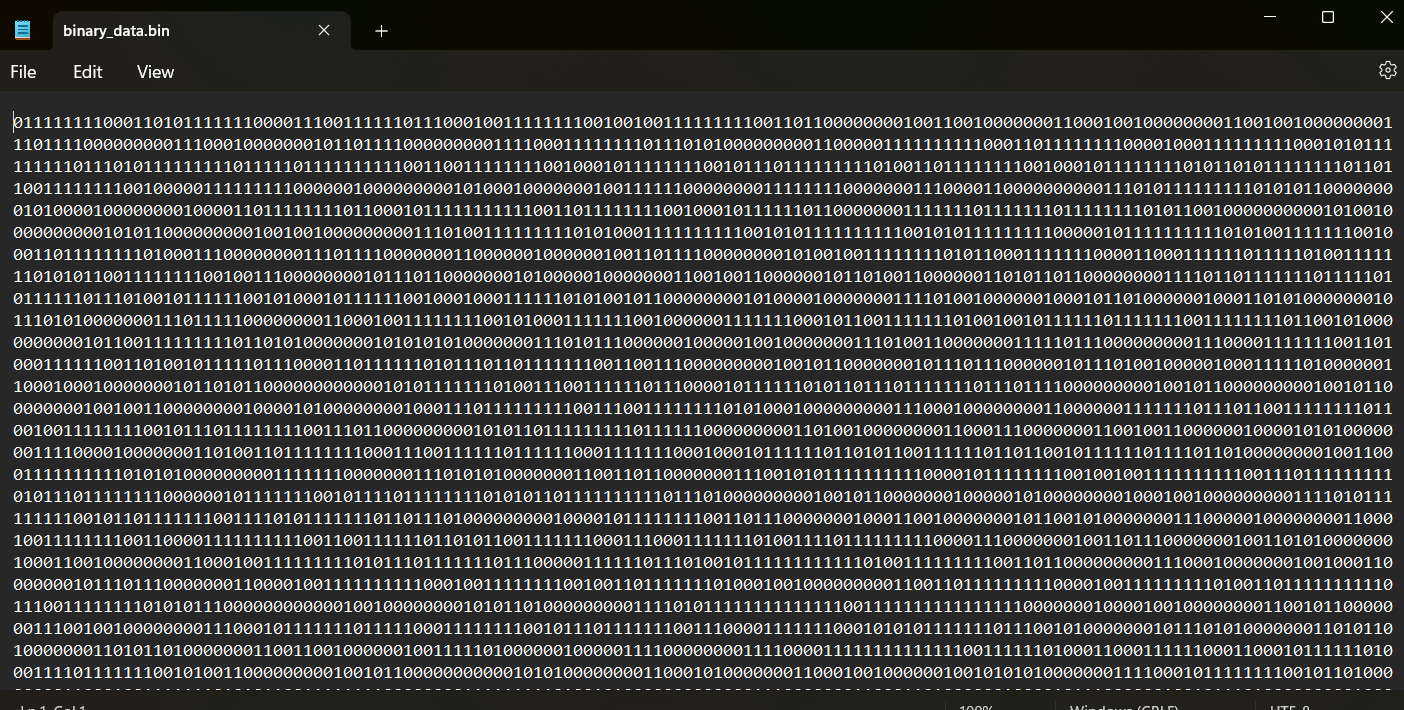
end

**Sau khi chạy ta thu được file nhạc mới**

****



**Dãy bit nhị phân:**



### **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.**

Ta dùng hàm

% Xác dinh thông tin ve file nhac

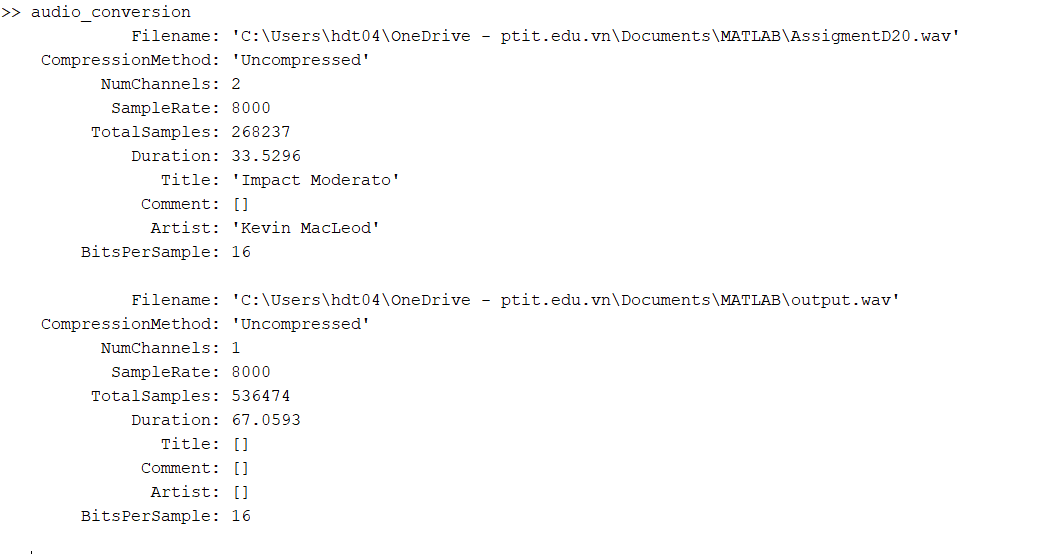
info = audioinfo('AssigmentD20.wav');

disp(info);

info\_output = audioinfo('output.wav');

disp(info\_output);

để xác định thông tin của file nhạc



* *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 kĩ 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:



trong đó *dk* là các kí hiệu (symbol) phức được xác định từ chuỗi bản tin đầu vào và kỹ thuật điều chế; *Tsym* là chu kỳ của symbol; ϕ*0* là pha của tín hiệu phát và *p(t)* xác định dạng xung được phát, với:  
 cho tín hiệu M-DPSK  
 cho tín hiệu M-QAM  
với *Es* là năng lượng mỗi symbol.

Mã Sinh Viên B20DCVT334 => N=4,M=4

* **Hệ thống 4-DPSK với N = 4 Mb/s.**

1. **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 4-DPSK:**

Đầu vào

(Binary Bit)

Wave Filter

Modulation Mapper

Mã hóa

(Điều chế 4-DPSK)

Đầu ra

Receiver

Kênh truyền

(Nhiễu AWGN)

Giải mã

(Giải điều chế 4-DPSK)

Bộ lọc

Wave Filter

Sử dụng bộ điều chế 4-DPSK. Các chuỗi bit đầu vào sẽ được tạo thành các xung, sau đó các xung này sẽ được đưa vào bộ điều chế 4-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 4-DPSK:**
* Tốc độ dữ liệu Rb = 4 Mb/s.
* Năng lượng symbol Es = 10 J.
* Tần số lấy mẫu Fs = 200 KHz.
* Tần số sóng mang Fc = 10 KHz.
* SNR = 5 dB.
* BER tại các mức SNR (Es/No) = [2 5 10] dB.
* Pha ban đầu của tín hiệu ϕ0 = 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**

%----Hoang Dong Tuan B20DCVT344----(4-QPSK,N=4Mb/s)

clear,clc,close all

M = 4; % Dieu che 4\_QPSK

Es =10; % Nang luong cho mot ky hieu

Rb=4e6; % Toc do bit- bit rate

Nsym=Rb/log2(M); % Toc do ki hieu- Symbol rate

Tsym=1/Nsym;% Chu ki cua ki hieu

n = 1e4; % So bit dau vao

phi=0; % Pha tin hieu phat

fs=3e5; % Tan so lay mau

fc=1e4; % Tan so song mang

ts=1/fs; % Chu ky lay mau

bf = 7e6; % Bang thong bo loc

%--------------------------------

%Tao tin hieu phat

bit\_input = randi([0 1], 1, n); % Tao chuoi bit ngau nhien dau vao

binary = bi2de(reshape(bit\_input,log2(M),length(bit\_input)/log2(M)).','left-msb'); % Chuyen doi nhi phan sang thap phan.

d = dpskmod(binary,M);

%-------------------------------

%\_\_ham s(t) dau bai cho\_\_\_\_\_\_\_\_

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

plot(t,s);

xlabel(' Thoi gian dieu che(s)');

ylabel( ' Bien do ');

title(' Do thi tin hieu goc ');

eyec=s.\*exp(1i\*phi); %tin hieu sau dieu che

signal\_fs=eyec.\*exp(1i\*2\*pi\*fc\*t); %Tin hieu de xac dinh mau mat

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%Do thi va pho cua tin hieu dieu che 4\_QPSK

figure

subplot(2,1,1);

plot(t,real(signal\_fs))

xlabel(' Thoi gian (s)'); ylabel( ' Bien do ');

title('Do thi tin hieu sau dieu che');

subplot(212);

spectrum(signal\_fs,fs);

title('Pho cua tin hieu sau dieu che');

eyediagram(real(signal\_fs),20);

title(' Bieu do mat tin hieu 4-DPSK');

scatterplot(d,1,0,'or');

title(' Bieu do chom sao cua tin hieu sau dieu che ' );

figure

signal\_noise=addNoiseAWGN(signal\_fs,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');

eyediagram(real(signal\_noise),20);

title('Bieu do mat tin hieu 4-DPSK khi qua AWGN');

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');

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');

h = scatterplot(source,1,0,'xb');

title('Bieu do chom sao cua tin hieu duoc khoi phuc');

eyediagram(real(source),20);

title('Mau mat duoc khoi phuc');

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\_input); %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');

%ham them kenh nhieu AWGN

function yNoise = addNoiseAWGN(y,SNR\_dB)

% y - Tin hieu dau v?o

% SNRdB - Muc SNR dB

% yNoise ? Tin hieu nhieu dau ra

SNR = 10^(SNR\_dB/10);

VarN = abs(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 tinh ber = phuong phap monteCarlo

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 = 4;

% 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 xu li bo loc

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

%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));

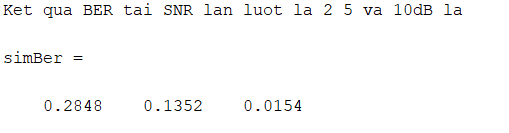
grid;

xlabel('Tan so (Hz)');

ylabel('Bien do');

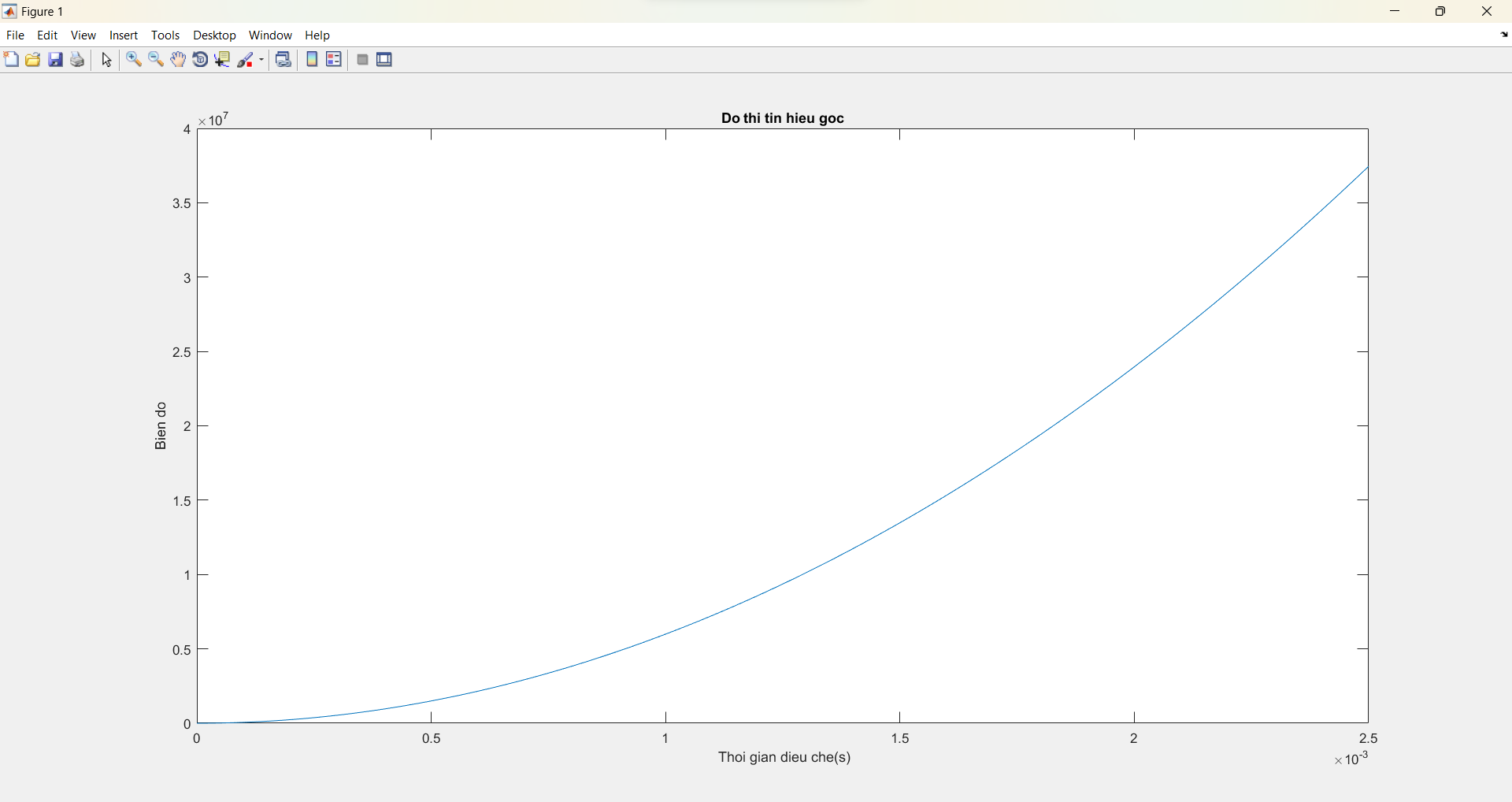
end

**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.**

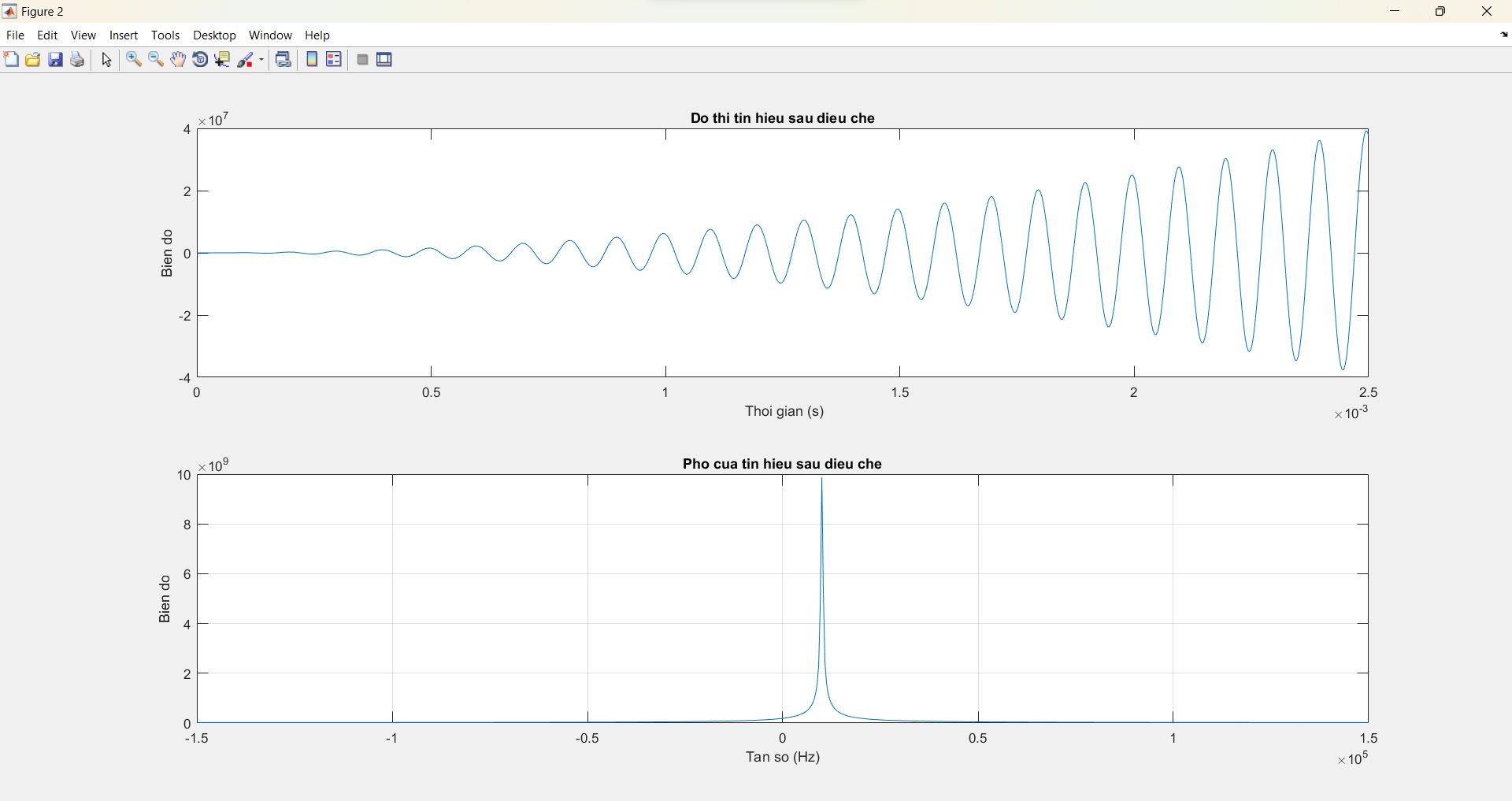


### **c.Biểu diễn biểu đồ chòm sao, dạng sóng tín hiệu, mẫu mắt và phổ của tín hiệu tại các điểm sau trên hệ thống: đầu ra bộ điều chế, sau khi truyền qua kênh AWGN tại SNR = 5dB, sau khi được xử lý và khôi phục tại bộ thu.**

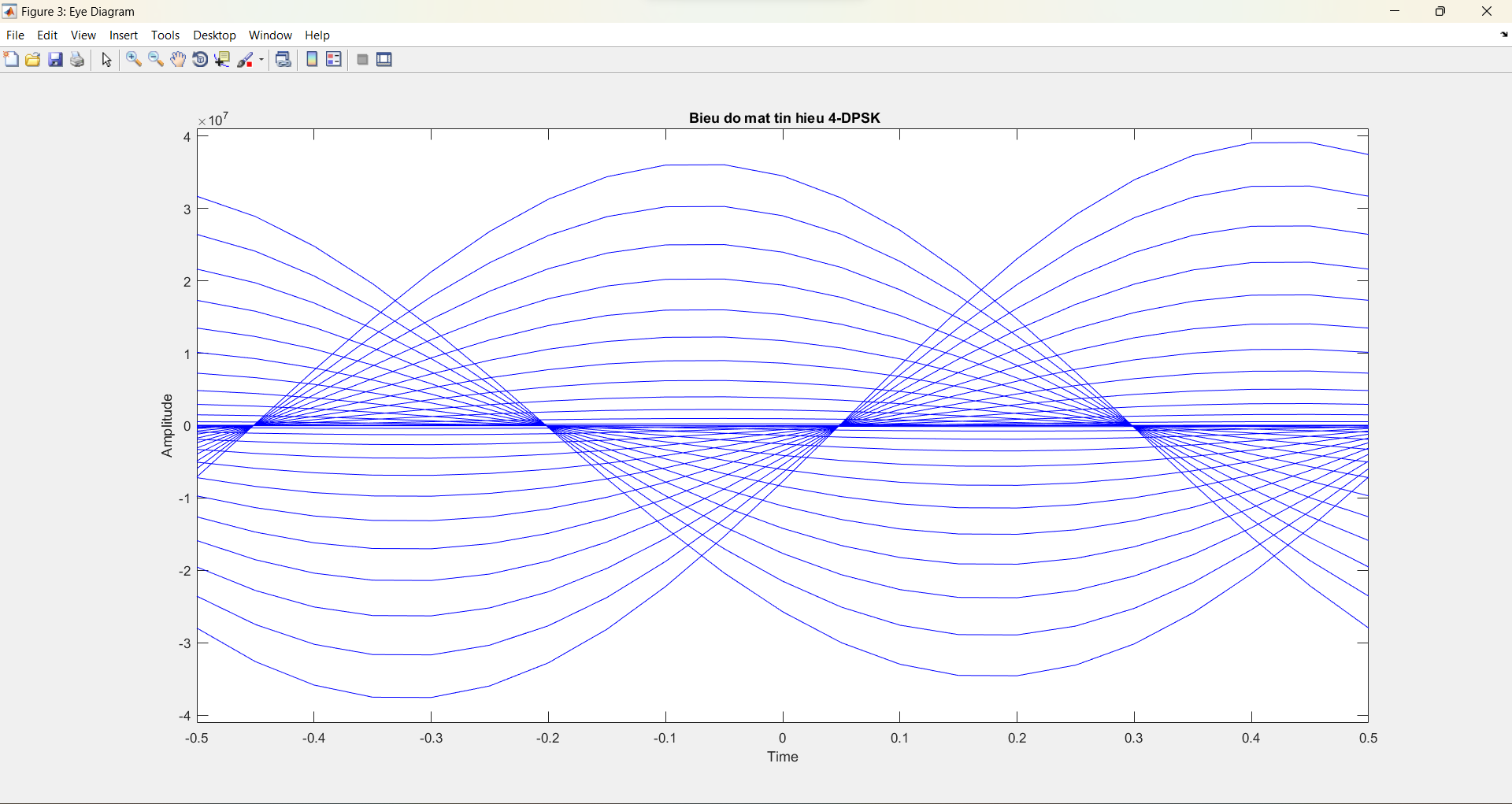
**Kết quả mô phỏng tại SNR = 5dB:**

****

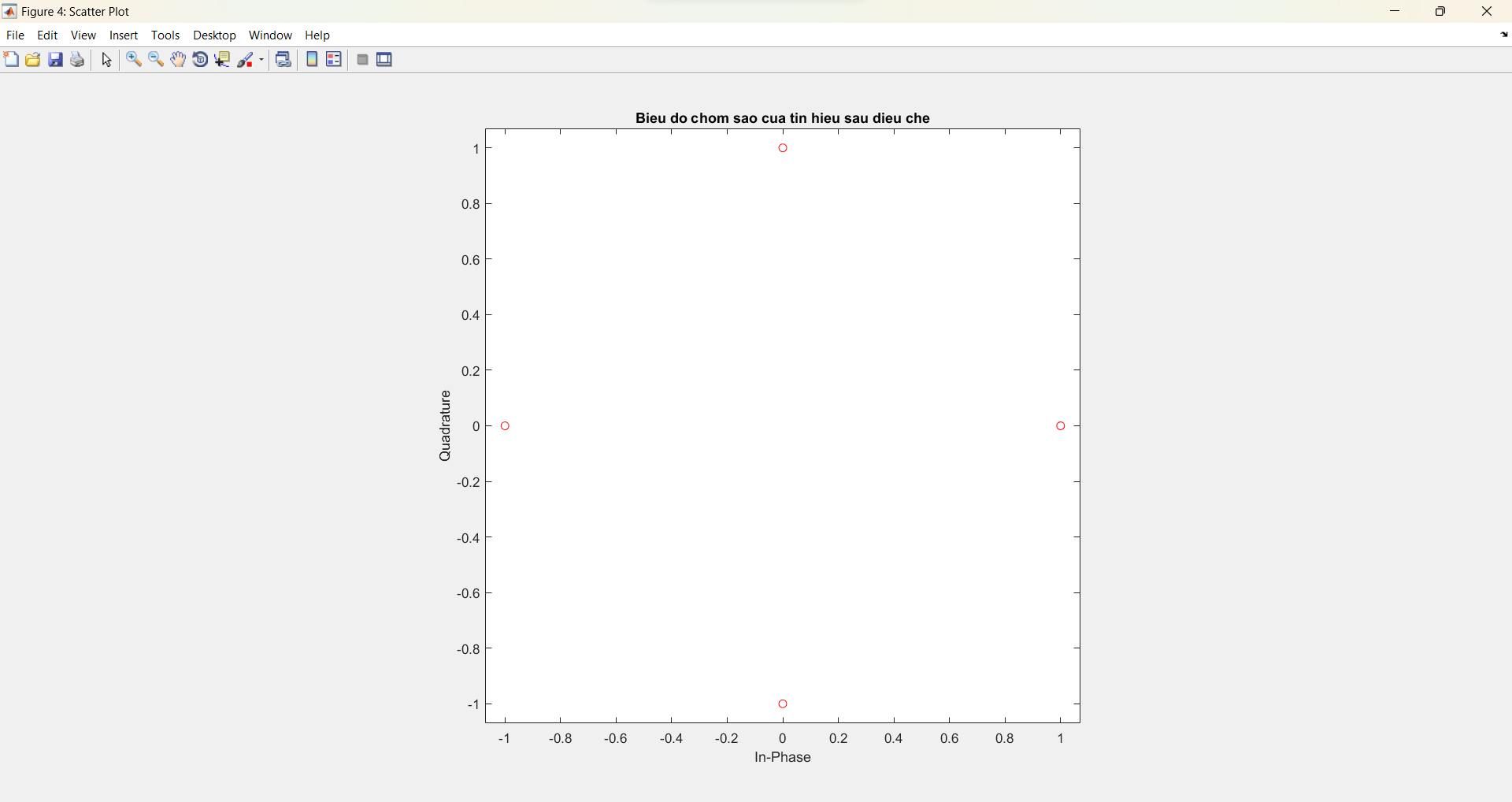
**Hình 1. Tín hiệu băng gốc**

****

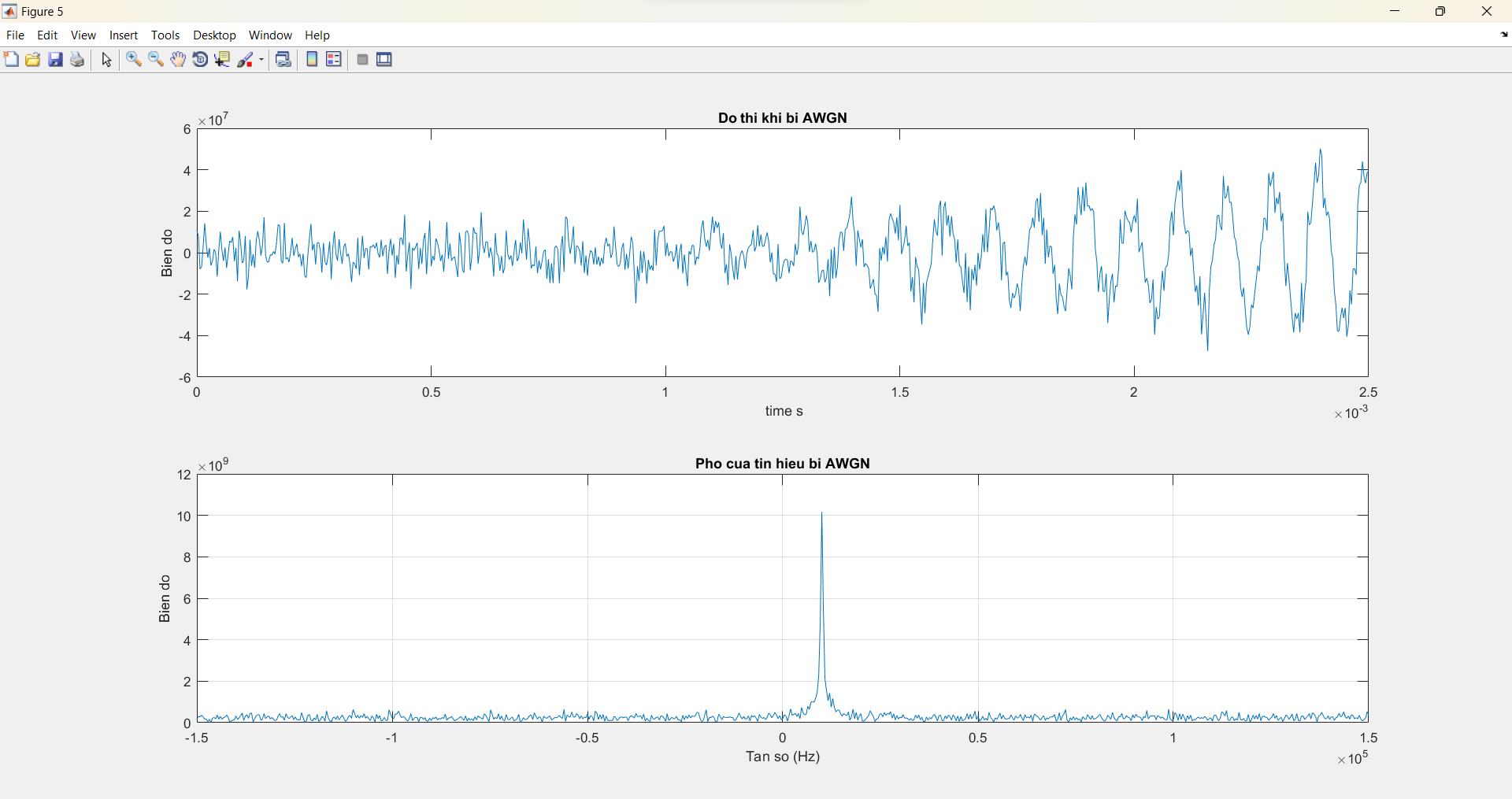
**Hình 2. Tín hiệu và phổ của tín hiệu sau điều chế**

****

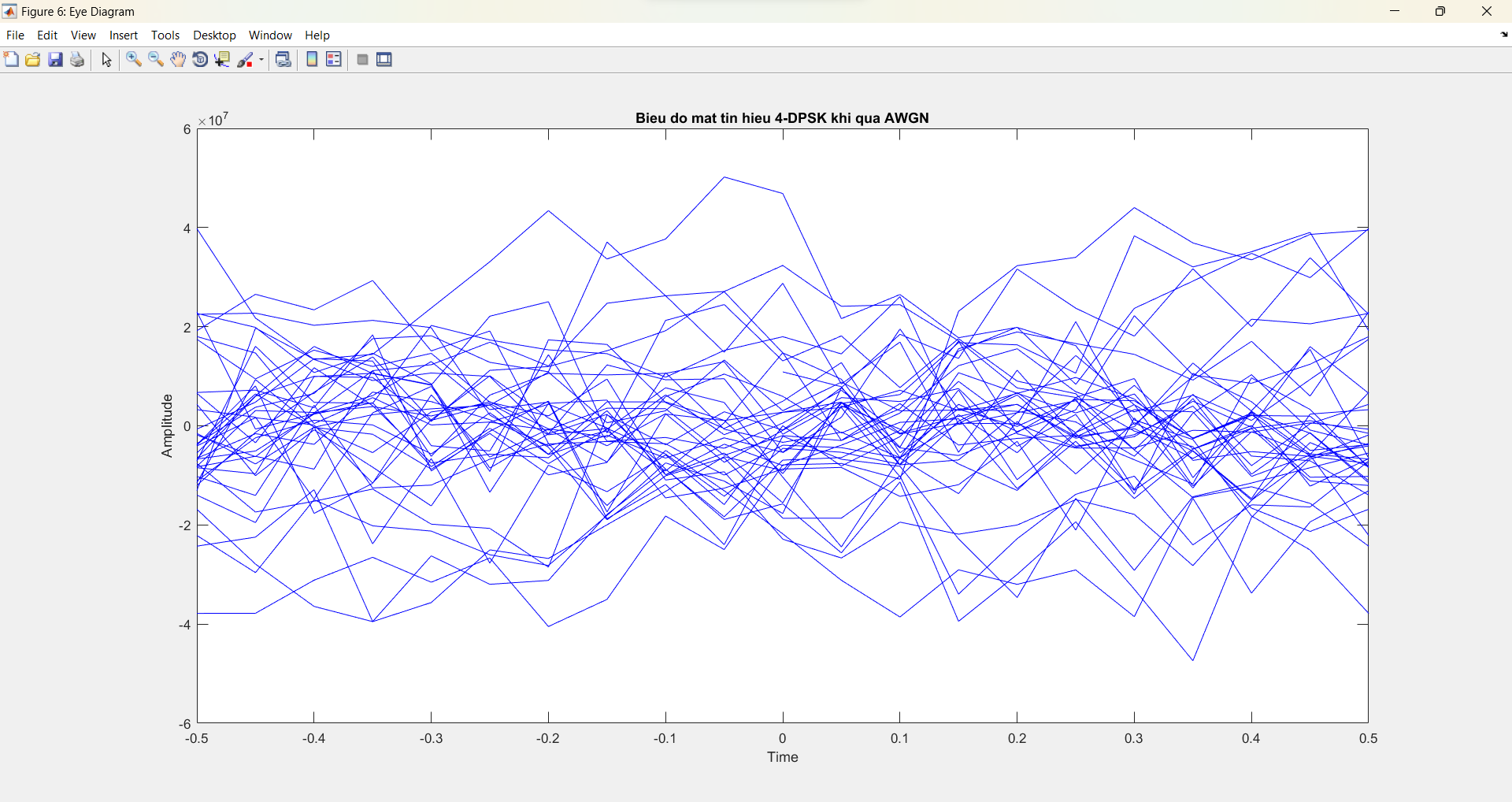
**Hình 3. Biểu đồ mắt của tín hiệu sau điều chế**

****

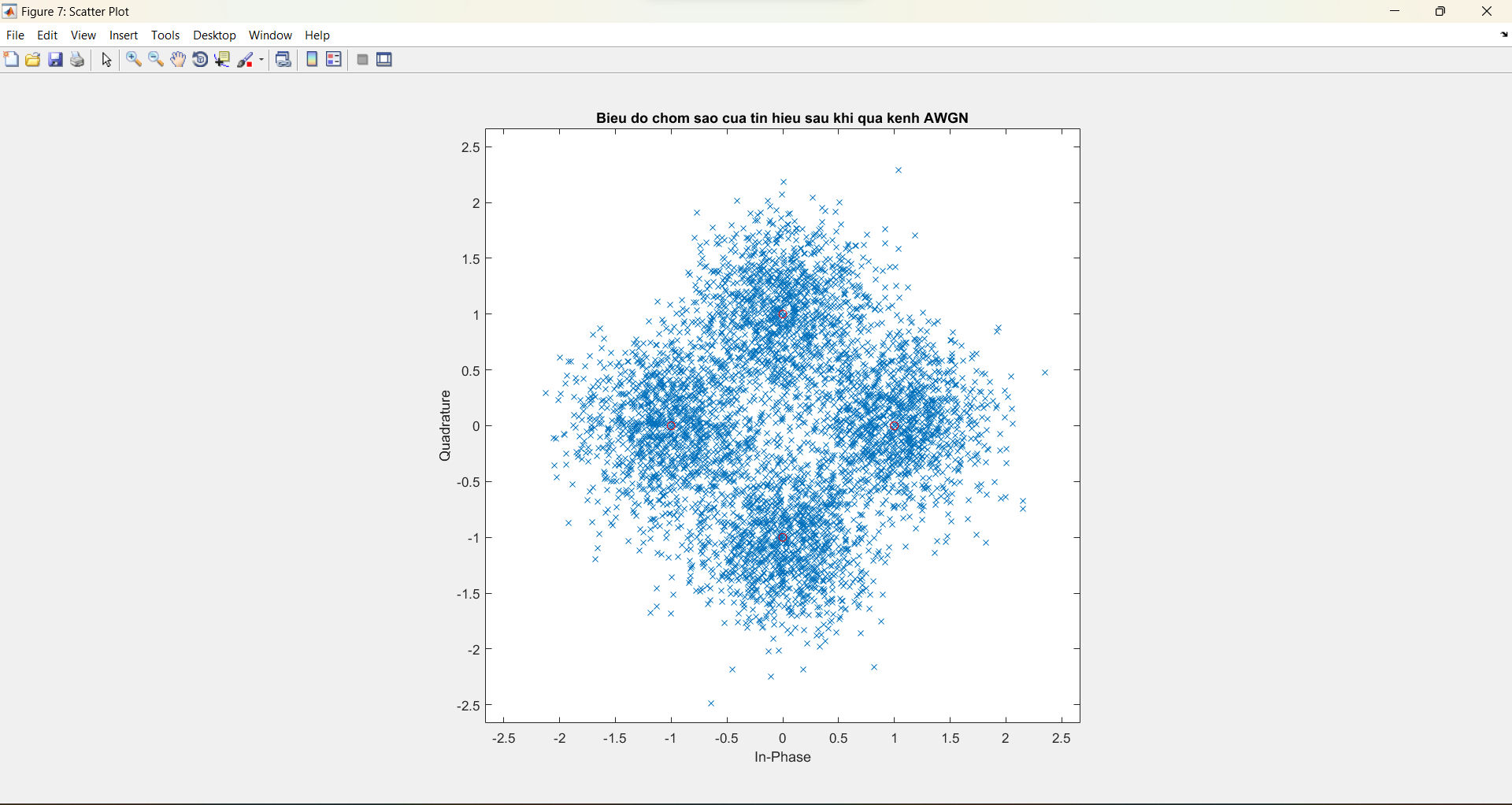
**Hình 4. Chòm sao của tín hiệu sau điều chế**

****

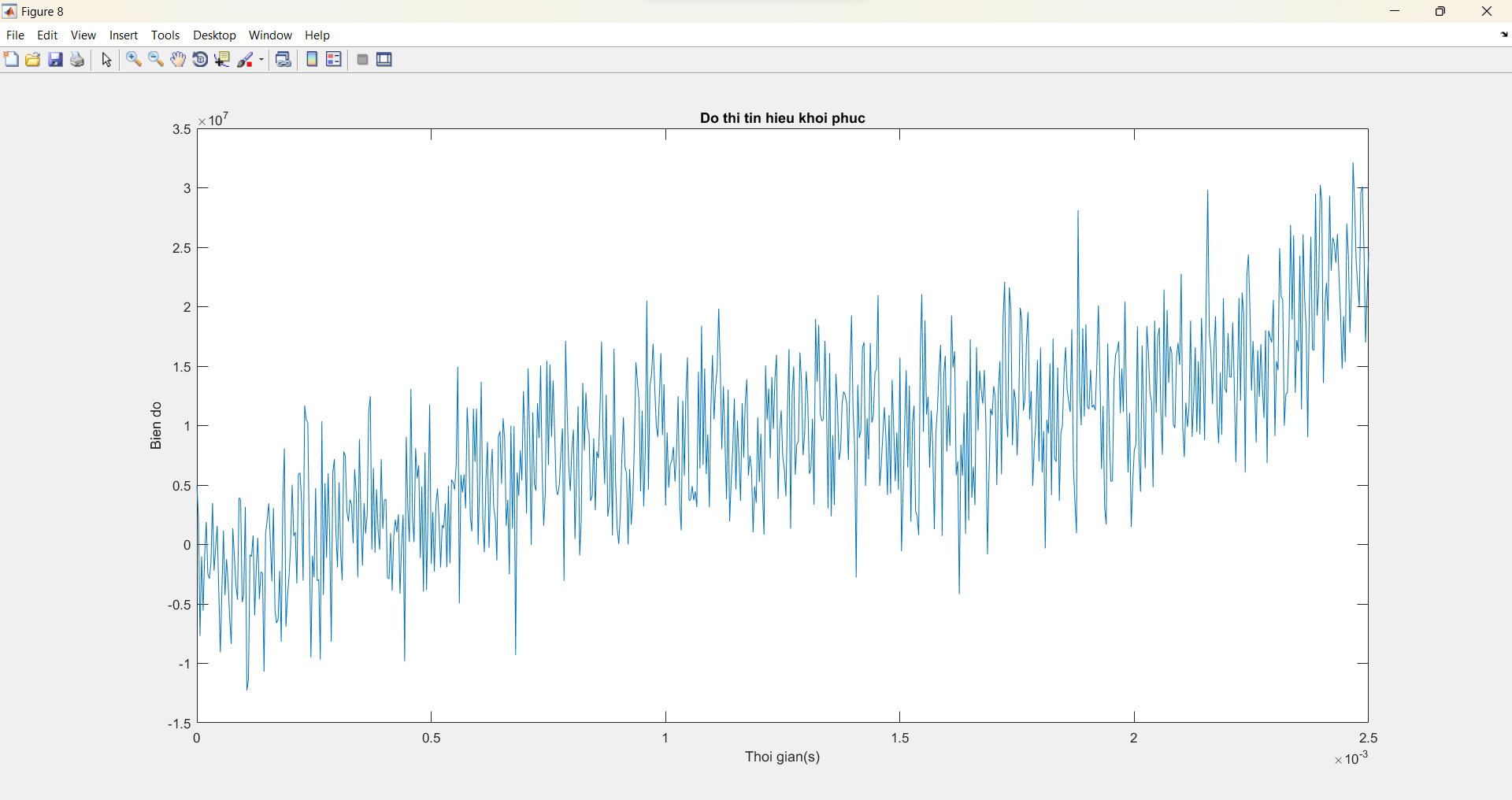
**Hình 5. Tín hiệu và phổ của nó sau khi qua AWGN**

****

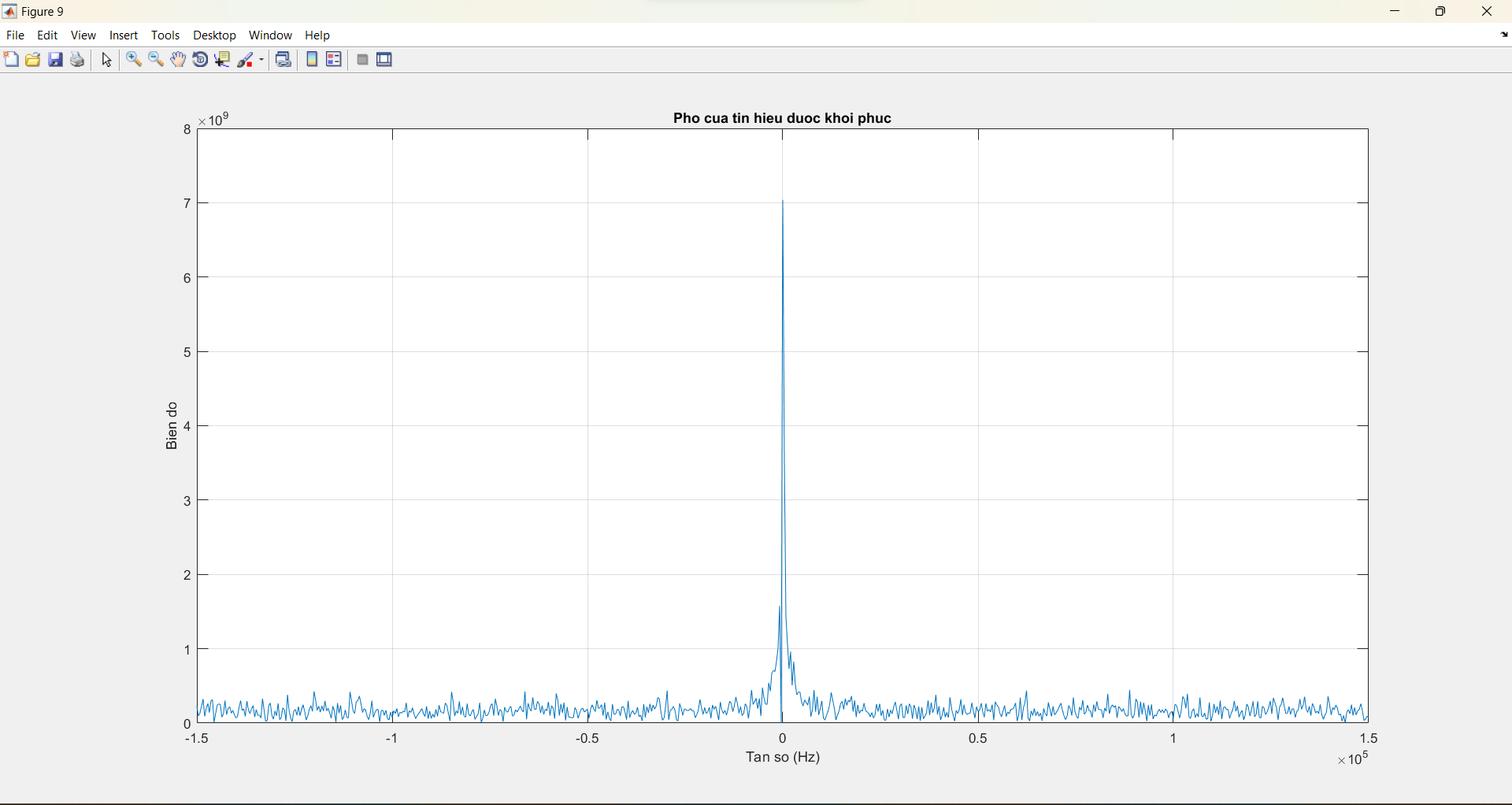
**Hình 6. Biểu đồ mắt của tín hiệu sau khi qua kênh AWGN**

****

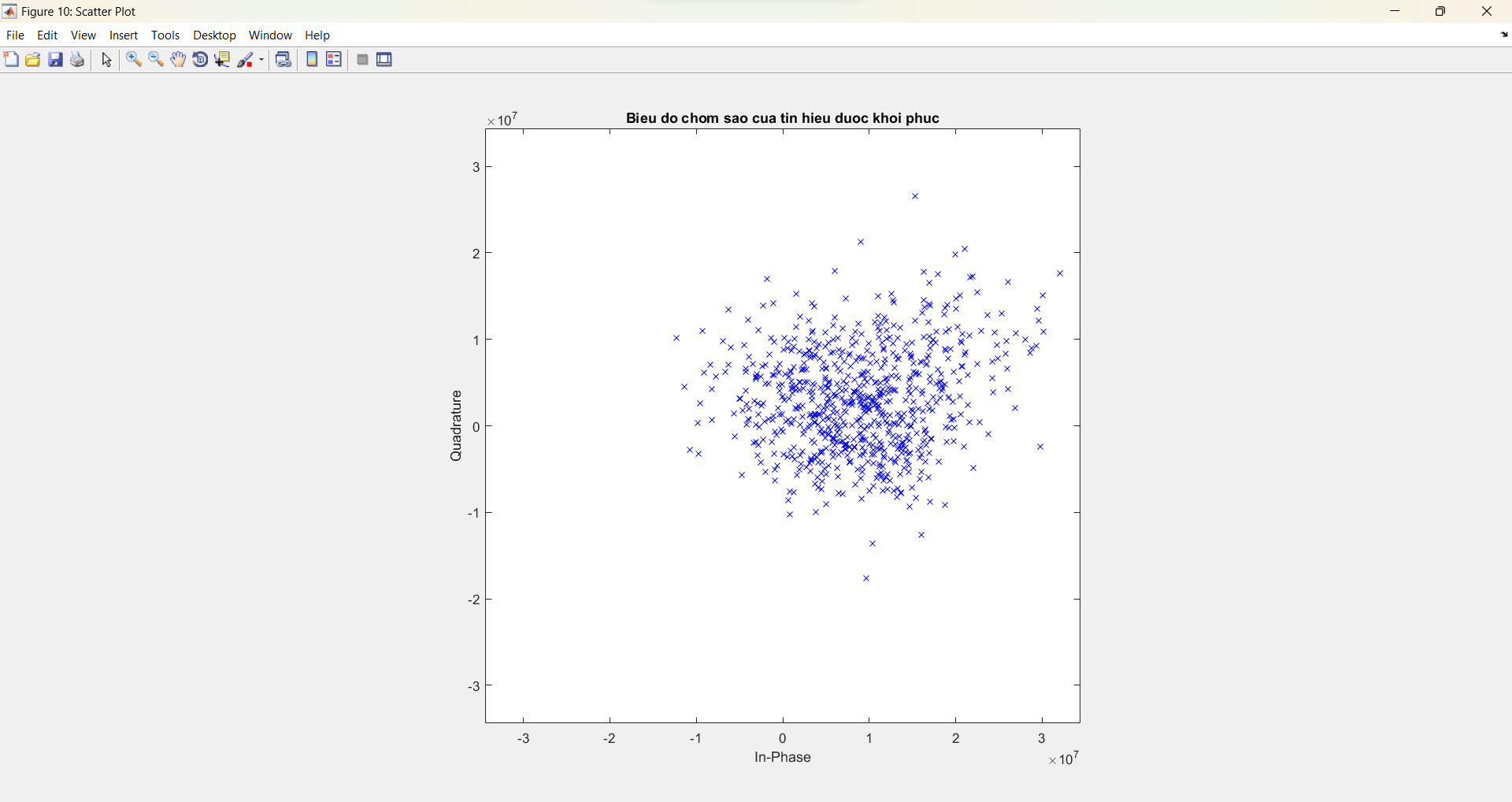
**Hình 7. Chòm sao của tín hiệu sau khi qua kênh AWGN**

****

**Hình 8. Tín hiệu sau khi khôi phục**

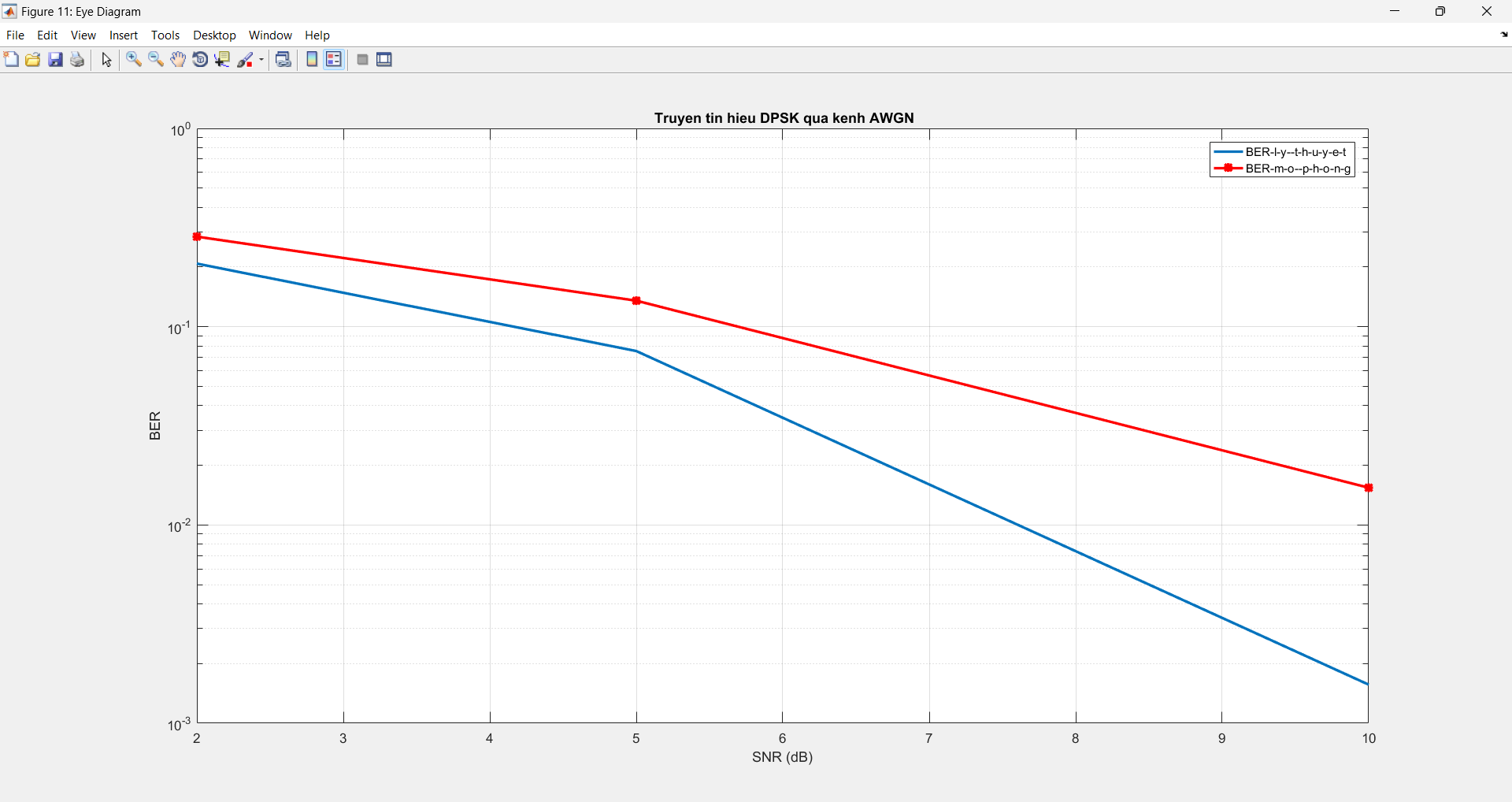
****

**Hình 9. Phổ của tín hiệu sau khi khôi phục**

****

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

### **d.So sánh file nhạc được khôi phục sau khi truyền qua hệ thống mô phỏng tại các mức SNR yêu cầu.**

****

**Hình 11. So sánh BER mô phỏng bằng Monte Carlo với BER được tính toán theo lý thuyết**