****

**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](#_Toc133706754)

[**1.** **Mô tả nhiệm vụ.** 3](#_Toc133706755)

[**2.** **Yêu cầu** 3](#_Toc133706756)

[**a)** **Mô tả phương pháp thực hiện chuyển đổi file nhạc yêu cầu thành chuỗi nhị phân và ngược lại.** 3](#_Toc133706757)

[**b)** **Viết chương trình MATLAB.** 4](#_Toc133706758)

[**c)** **Xác định các thông số về file nhạc cần chuyển đổi.** 7](#_Toc133706759)

[**Nhiệm vụ 2** 8](#_Toc133706760)

[**1.**  **Mô tả nhiệm vụ:** 8](#_Toc133706761)

[**2.** **Yêu cầu:** 9](#_Toc133706762)

[**a) Mô tả hệ thống mô phỏng bằng sơ đồ khối và xác định các tham số của hệ thống.** 9](#_Toc133706763)

[**b) Viết chương trình MATLAB.** 9](#_Toc133706764)

[**c) Biểu diễn biểu đồ chòm sao, dạng sóng tín hiệu, mẫu mắt và phổ của tín hiệu.** 16](#_Toc133706765)

[**d) So sánh file nhạc được khôi phục sau khi truyền qua hệ thống mô phỏng tại các mức SNR yêu cầu.** 27](#_Toc133706766)

# 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**
2. **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.

1. **Viết chương trình MATLAB.**

**Thực hiện chuyển đổi file nhạc *AssigmentD20.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:**

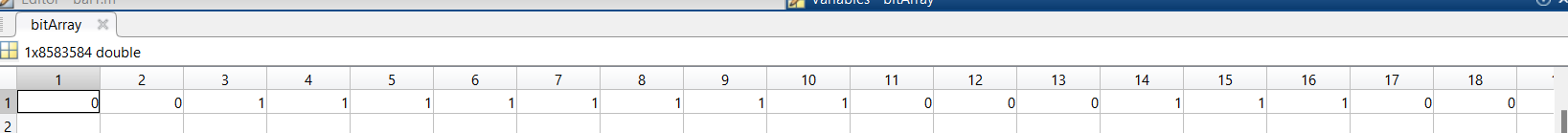
**Hàm chuyển đổi file nhạc sang chuỗi bit nhị phân**

|  |
| --- |
| function [samples,bitArray,fs,n,title,artist] = audioToBinary(fileName)  % Doc va luu du lieu am thanh vao mang  [samples, fs] = audioread(fileName);  % Ðoc cac thong so cua file am thanh goc  originInfo = audioinfo(fileName);  title = originInfo.Title;  artist = originInfo.Artist;  % Luu kich thuoc ma tran samples vua doc duoc tu am thanh goc  [m, n] = size(samples);  samples = reshape(samples, [], 1);  % Chuyen doi du lieu am thanh sang du lieu so  % roi chuyen ve chuoi 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 chuoi bit nhi phan sang vector chua cac so 0 1 |

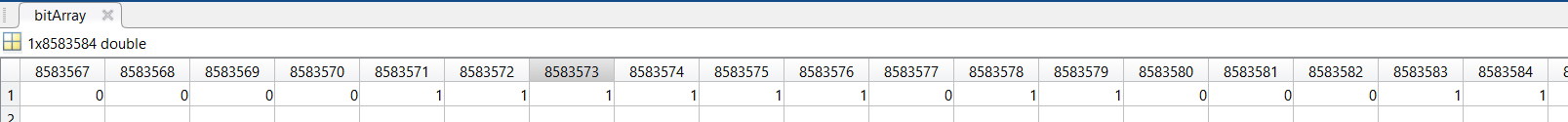
**Hàm chuyển đổi bit nhị phân sang file nhạc mới**

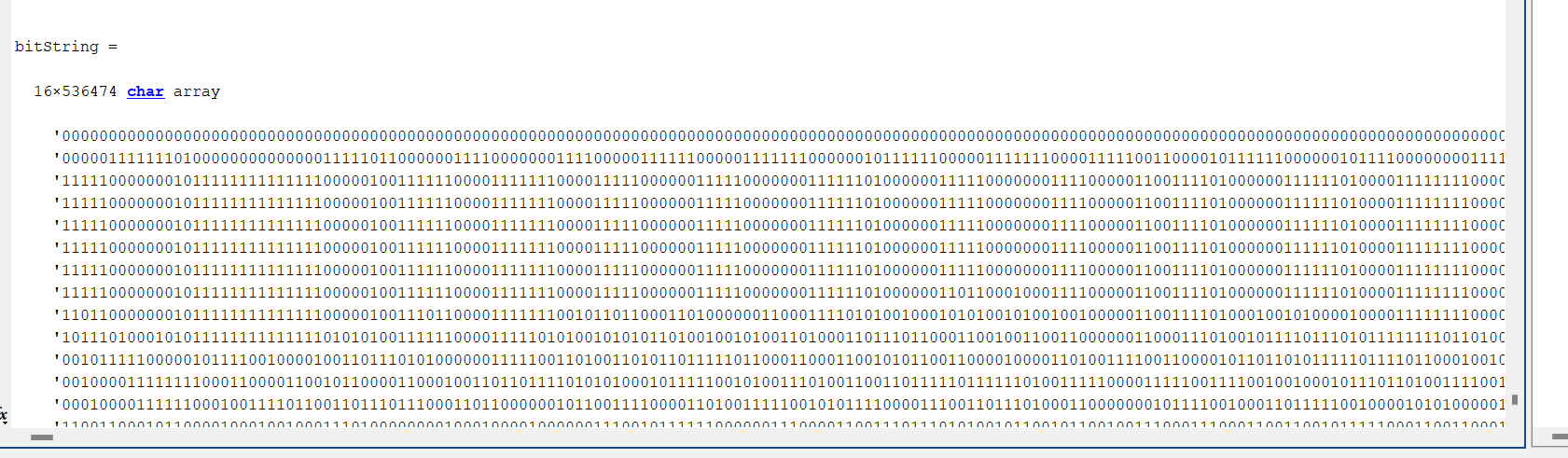
|  |
| --- |
| function newInfo = binaryToAudio(bitArray, fs, n, title, artist)  % Chuyen doi chuoi bit nhi phan thanh day so nguyen kieu unit 16  samples = bi2de(reshape(bitArray, 16, [])', 'left-msb');  % Chuyen nguoc day so nguyen thanh gia tri am thanh  % co dang nhu mang du lieu ban dau khi doc file am thanh  samples = (samples/(2^15-1))\*2-1;  % Tra ve kich thuoc ban dau cua mang du lieu samples  samples = reshape(samples, [], n);  % Ghi du lieu am thanh vao file moi voi cac thong so tu file goc  audiowrite('newfile.wav', samples, fs, 'Title', title, 'Artist', artist);  % Kiem tra thong so cua file am thanh moi so voi file goc  newInfo = audioinfo('newfile.wav'); |

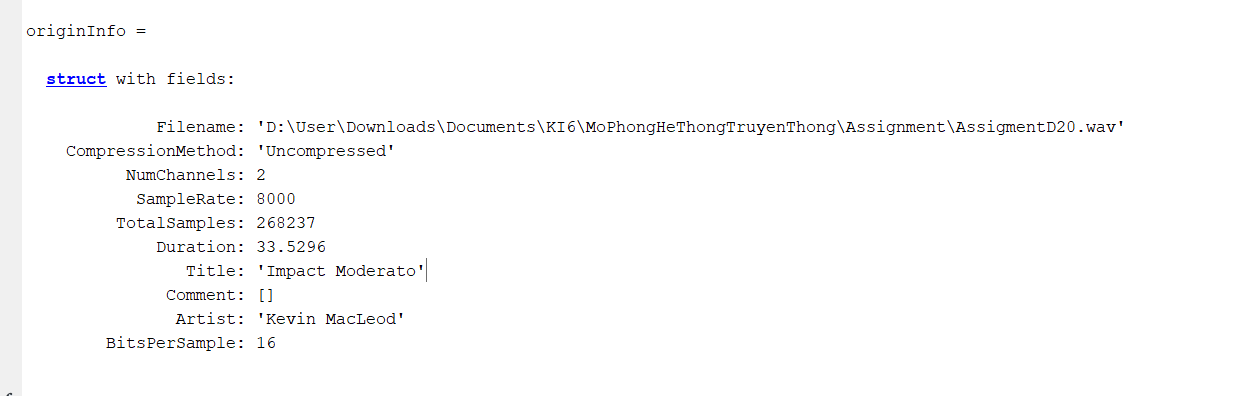
**Kết quả sau khi thực hiện hàm *audioToBinary* và *binaryToAudio:***

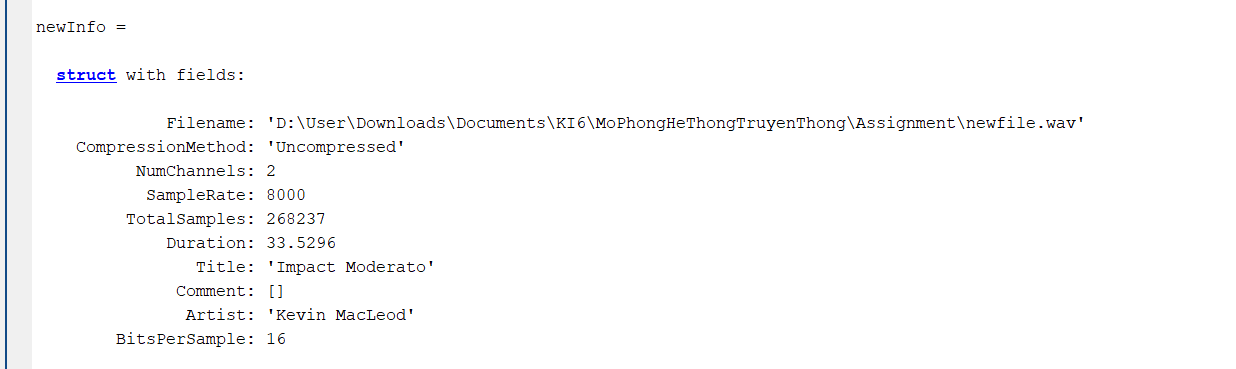
****

**…**

****

****

****

****

1. **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):**

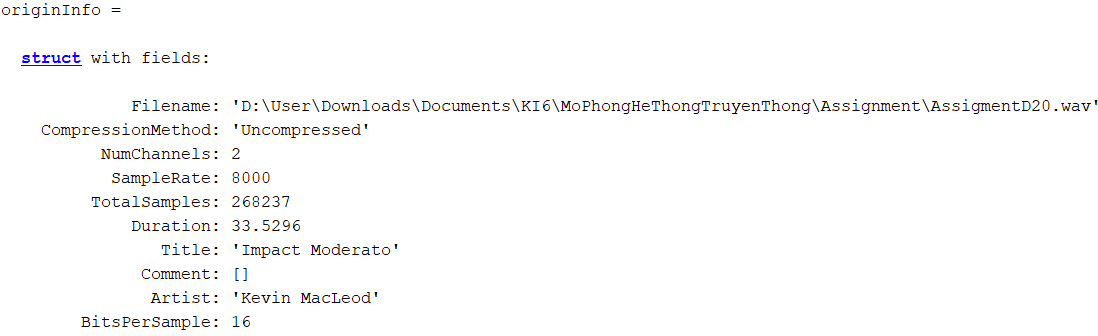
***audioread* – đọc dữ liệu từ file nhạc.**

***audiowrite* – ghi file nhạc.**

***audioinfo* – thông tin về file nhạc.**

***unit16* – chuyển đổi kiểu dữ liệu về unit16.**

Sử dụng lệnh *originInfo = audioinfo(‘AssigmentD20.wav’);* ta được các thông số của file nhạc cần chuyển đổi:



* *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.

* **Hệ thống 2-DPSK với N = 6 Mb/s.**

1. **Yêu cầu****.**
2. **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:**

Nguồn tin

(Binary Bit)

Kênh truyền

(Nhiễu AWGN)

Mã hóa

(Điều chế 2-DPSK)

Đầu ra

Receiver

Giải mã

(Giải điều chế 2-DPSK)

Bộ lọc FIR

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.

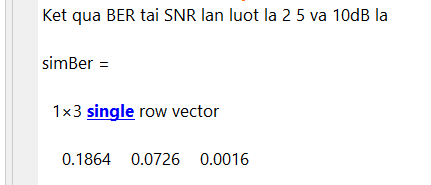
* **Xác định tham số của hệ thống 2-DPSK:**
* Tốc độ dữ liệu Rb = 6 Mb/s.
* Năng lượng symbol Es = 10 J.
* Tần số lấy mẫu Fs = 300 KHz.
* Tần số sóng mang Fc = 10 KHz.
* SNR = 5 dB.
* BER tại các mức SNR (Es/No) = [2 5 10] dB.
* Pha ban đầu của tín hiệu ϕ0 = 0.

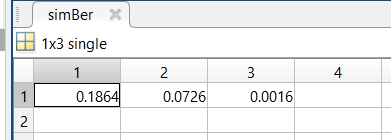
1. **Bằng việc sử dụng MATLAB, viết chương trình mô phỏng hệ thống truyền dẫn số sử dụng kỹ thuật điều chế đã lựa chọn trên kênh AWGN với nguồn tín hiệu là tín hiệu thu được ở nhiệm vụ 1. Ước tính xác suất lỗi tại các mức tỉ số tín hiệu trên nhiễu SNR lần lượt bằng 2, 5 và 10 dB theo phương pháp Monte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

**Code:**

|  |
| --- |
| %% -----Nguyen Di Dan - B20DCVT086 (2-DPSK N= 6MB/s) ---------------  clear,clc,close all  M = 2; % So muc dieu che  Es =10; % Nang luong cho mot ky hieu  Rb=6e6; % Toc do bit- bit rate  Nsym=Rb/log2(M); % Toc do ki hieu- Symbol rate  Tsym=1/Nsym;% Chu k? kí 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  %% Nhap tin hieu phat  bit = randi([0 1], 1, n); % Chuoi bit dau vao ngau nhien  dec = bi2de(reshape(bit,log2(M),length(bit)/log2(M)).','left-msb'); % Chuyen doi nhi phan sang thap phan.  d = dpskmod(dec,M); % Dieu che 2-DPSK  %% Xay dung ham 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  %% Do thi tin hieu bang goc  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 |

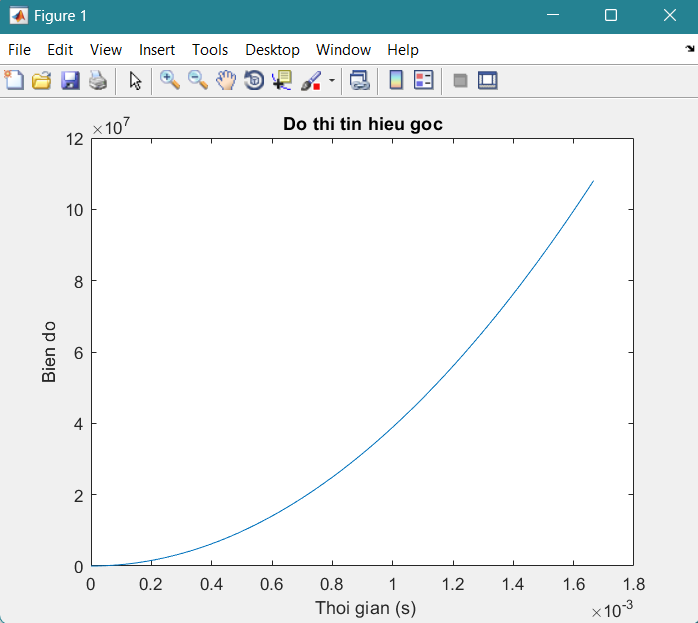
**Ước tính xác suất lỗi tại các mức tỉ số tín hiệu trên nhiễu SNR lần lượt bằng 2, 5 và 10 dB theo phương pháp Monte Carlo.**

****

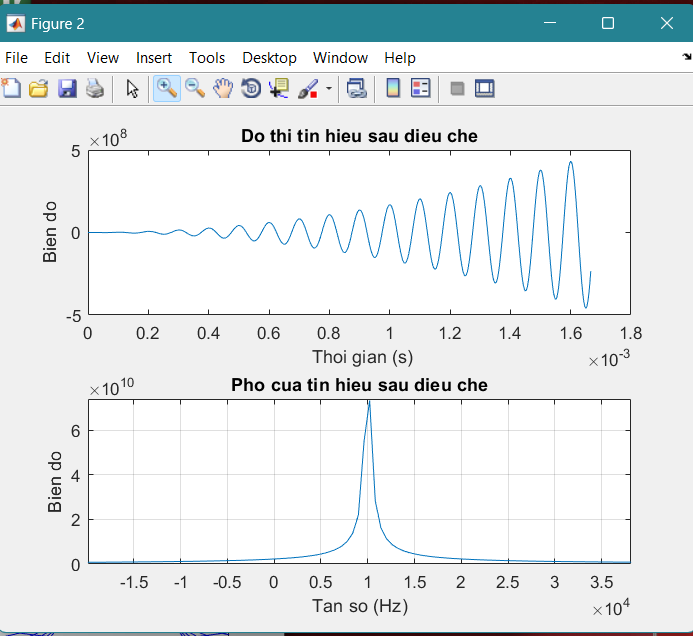
****

**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.**

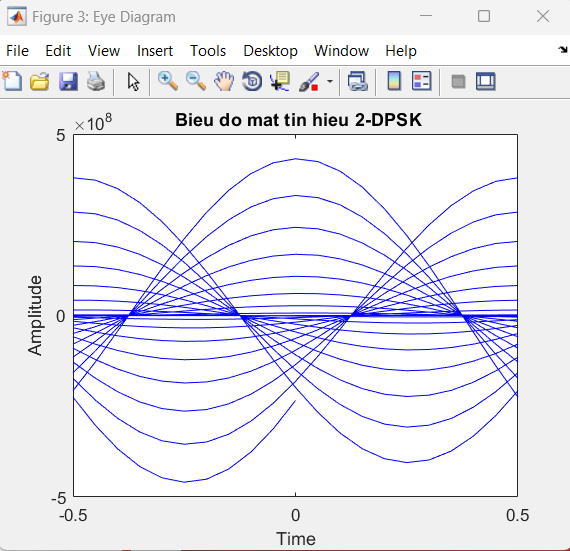
**Kết quả mô phỏng tại SNR = 5dB:**

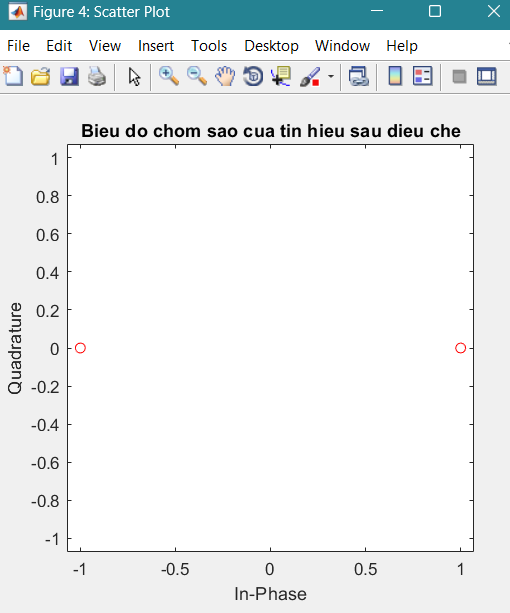
****

**Hình 1. Tín hiệu băng gốc**

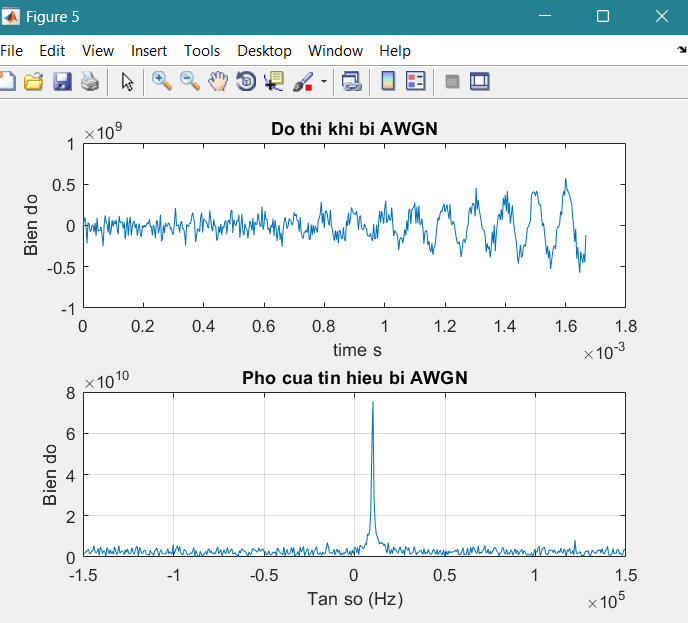
****

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

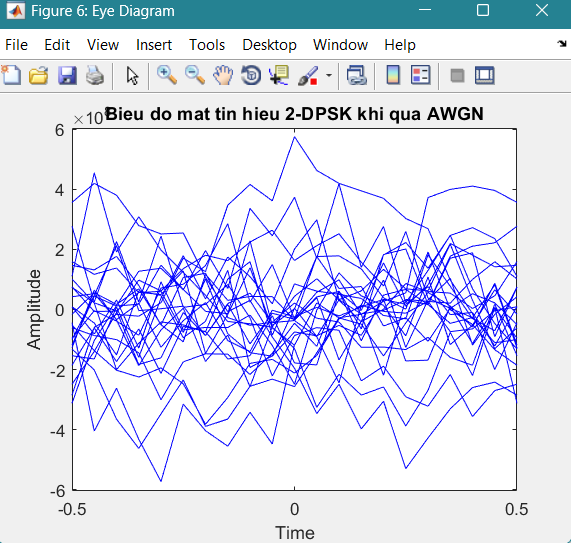
**  
Hình 3. Biểu đồ mắt của tín hiệu sau điều chế**

****

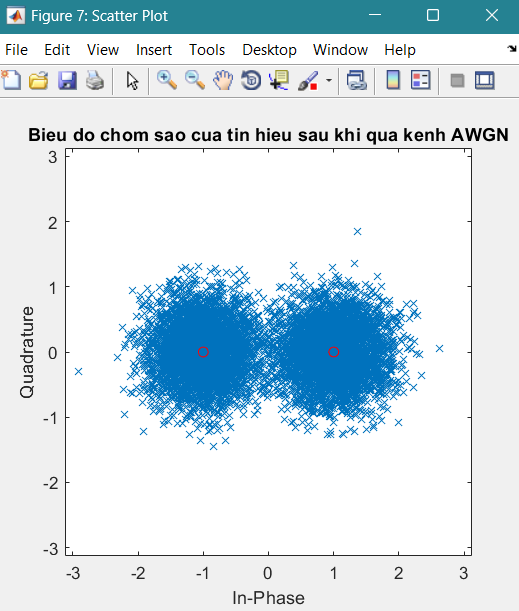
**Hình 4. Chòm sao của tín hiệu sau điều chế**

****

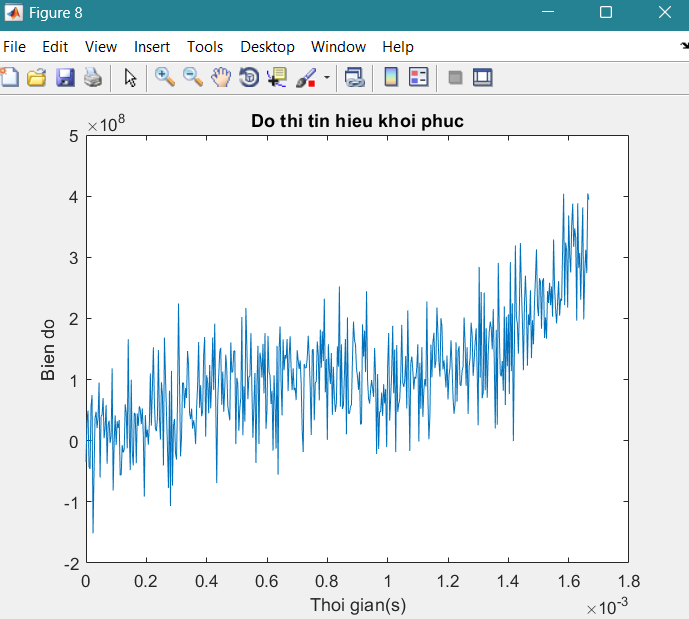
**Hình 5. Tín hiệu và phổ của nó sau khi qua AWGN**

****

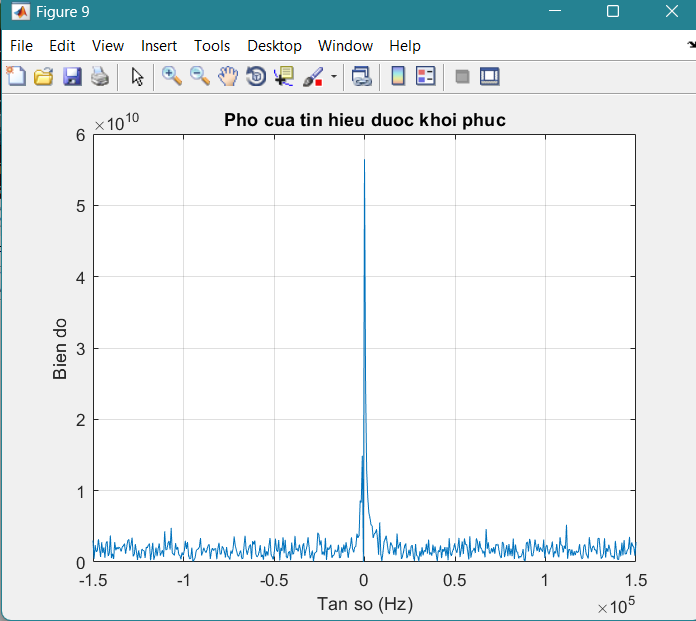
**Hình 6. Biểu đồ mắt của tín hiệu sau khi qua kênh AWGN**

****

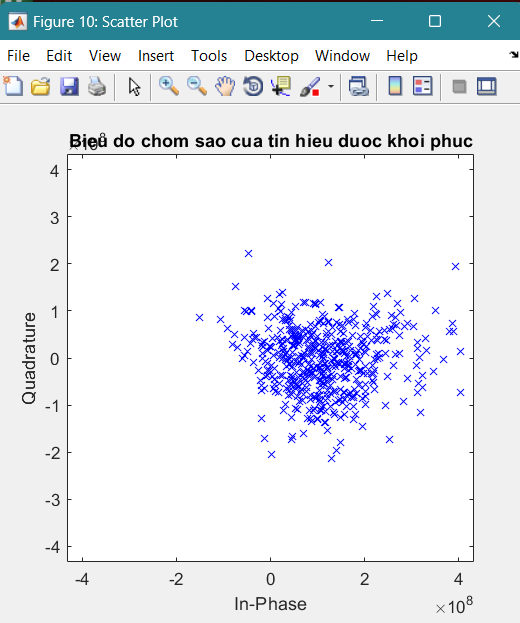
**Hình 7. Chòm sao của tín hiệu sau khi qua kênh AWGN**

****

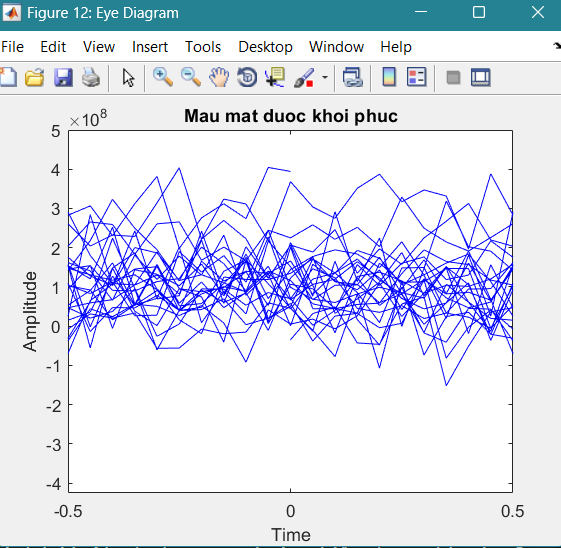
**Hình 8. Tín hiệu sau khi khôi phục**

****

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

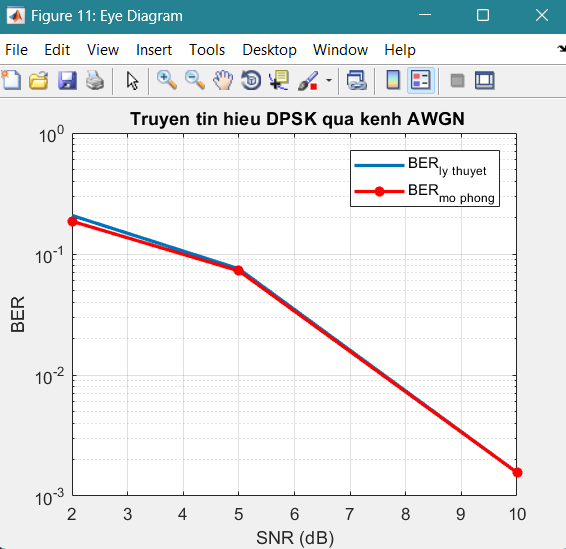
****

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

****

**Hình 11. Biểu đồ mẫu mắt sau khi khôi phục**

1. **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**