Trường Đại học Bách Khoa – Đại học Đà Nẵng

Khoa Điện Tử - Viễn Thông

PBL 4: BÁO CÁO MÔ PHỎNG CHƯƠNG 7

Nhóm 6

Sinh viên thực hiện

Đinh Văn Quang 20DT1

Nguyễn Anh Tuấn 20DT1

Giảng viên hướng dẫn

TS. Võ Duy Phúc

Đà Nẵng, 5/2024

CHƯƠNG 7: MÔ PHỎNG KỸ THUẬT OFDM

Bài tập 1: Chạy đoạn code trên, viết đoạn code Matlab để thực hiện điều chế OFDM đối với số lượng ký tự truyền đi là 20000, băng thông là 4 MHz. Biểu diễn kết quả thu được trên một hình để so sánh, rút ra kết luận.

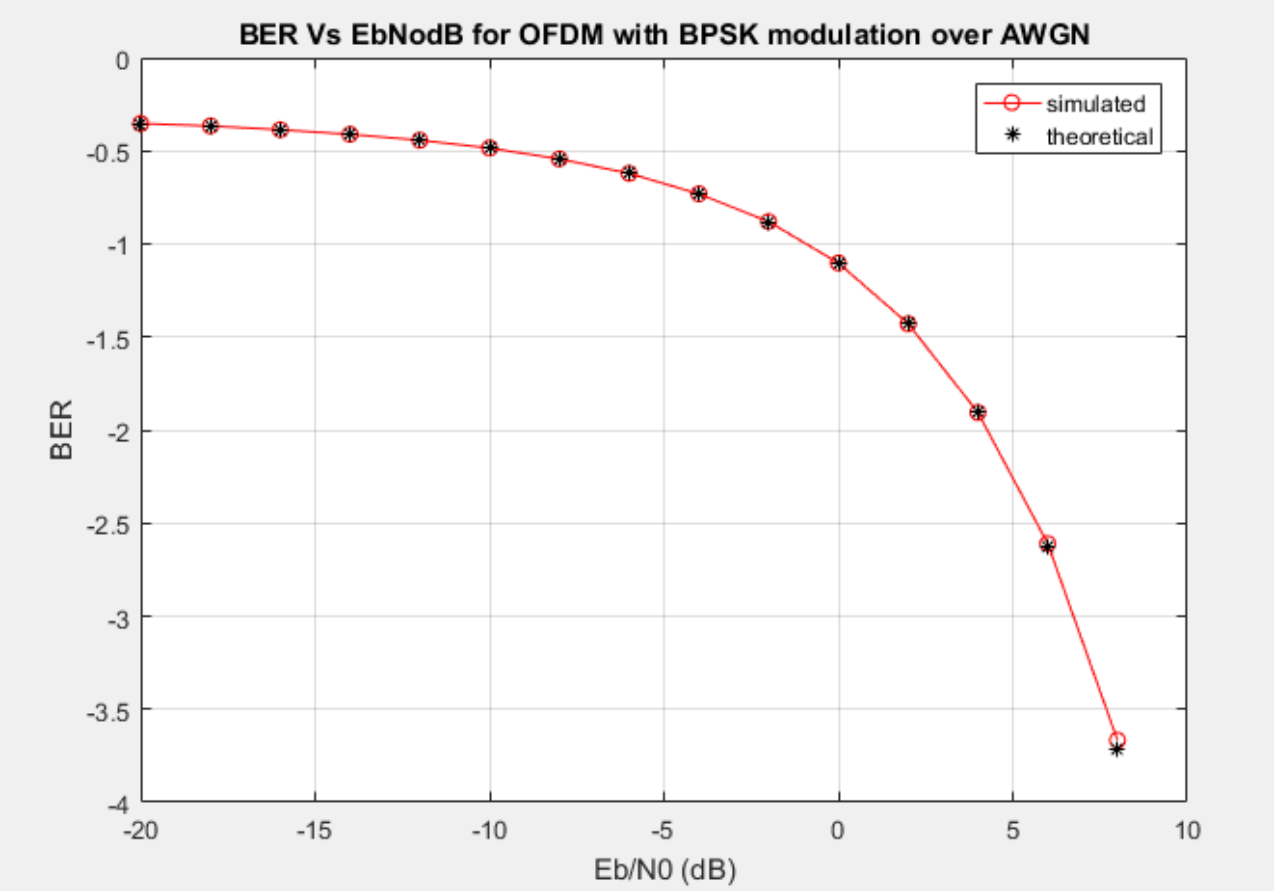
Code Matlab:

```
clear; clc;
%--------Simulation parameters----------------nSym=10^4;
%Number of OFDM Symbols to transmit
nSym=20000;
ofdmBW = 4e6;
EbN0dB = -20:2:8; % bit to noise ratio
N=64;
Nsd=48;%Number of data subcarriers 48
Nsp=4;%Numberof pilot subcarriers 4
%OFDMbandwidth
deltaF = ofdmBW/N;  %=20 MHz/64 = 0.3125 MHz
Tfft = 1/deltaF; % IFFT/FFT period = 3.2us
Tgi = Tfft/4;%Guard interval duration - duration of cyclic prefix
Tsignal = Tgi+Tfft; %duration of BPSK-OFDM symbol
Ncp = N*Tgi/Tfft; %Number of symbols allocated to cyclic prefix
Nst = Nsd + Nsp;  %Number of total used subcarriers
nBitsPerSym=Nst; %For BPSK the number of Bits per Symbol is same as num of subcarriers
EsN0dB = EbN0dB + 10*log10(Nst/N) + 10*log10(N/(Ncp+N)); % converting to symbol to noise ratio
errors= zeros(1,length(EsN0dB));
theoreticalBER = zeros(1,length(EsN0dB));
%Monte Carlo Simulation
for i=1:length(EsN0dB),
for j=1:nSym
s=2*round(rand(1,Nst))-1; X_Freq=[zeros(1,1) s(1:Nst/2) zeros(1,11) s(Nst/2+1:end)];
% Pretending the data to be in frequency domain and converting to time domain
x_Time=N/sqrt(Nst)*ifft(X_Freq);
%Adding Cyclic Prefix
ofdm_signal=[x_Time(N-Ncp+1:N) x_Time];
noise=1/sqrt(2)*(randn(1,length(ofdm_signal))+1i*randn(1,length(ofdm_signal)));
r=sqrt((N+Ncp)/N)*ofdm_signal+10^(-EsN0dB(i)/20)*noise;
%-----------------Receiver--------------------%Removing cyclic prefix
r_Parallel=r(Ncp+1:(N+Ncp));
%FFTBlock
r_Time=sqrt(Nst)/N*(fft(r_Parallel));
%Extracting the data carriers from the FFT output
R_Freq=r_Time([(2:Nst/2+1) (Nst/2+13:Nst+12)]);
%BPSK demodulation / Constellation Demapper. Force +ve value --> 1, -ve value --> -1
R_Freq(R_Freq>0) = +1;
R_Freq(R_Freq<0) = -1;
```

```
s_cap=R_Freq;
numErrors = sum(abs(s_cap-s)/2); %Count number of errors
%Accumulate bit errors for all symbols transmitted
errors(i)=errors(i)+numErrors;
end
theoreticalBER(i)=(1/2)*erfc(sqrt(10.^(EbN0dB(i)/10))); %Same as BER for BPSK over A
end
simulatedBER = errors/(nSym*Nst);
plot(EbN0dB,log10(simulatedBER),'r-o');
hold on;
plot(EbN0dB,log10(theoreticalBER),'k*');
grid on;
title('BER Vs EbNodB for OFDM with BPSK modulation over AWGN');
xlabel('Eb/N0 (dB)');ylabel('BER');legend('simulated','theoretical');
```

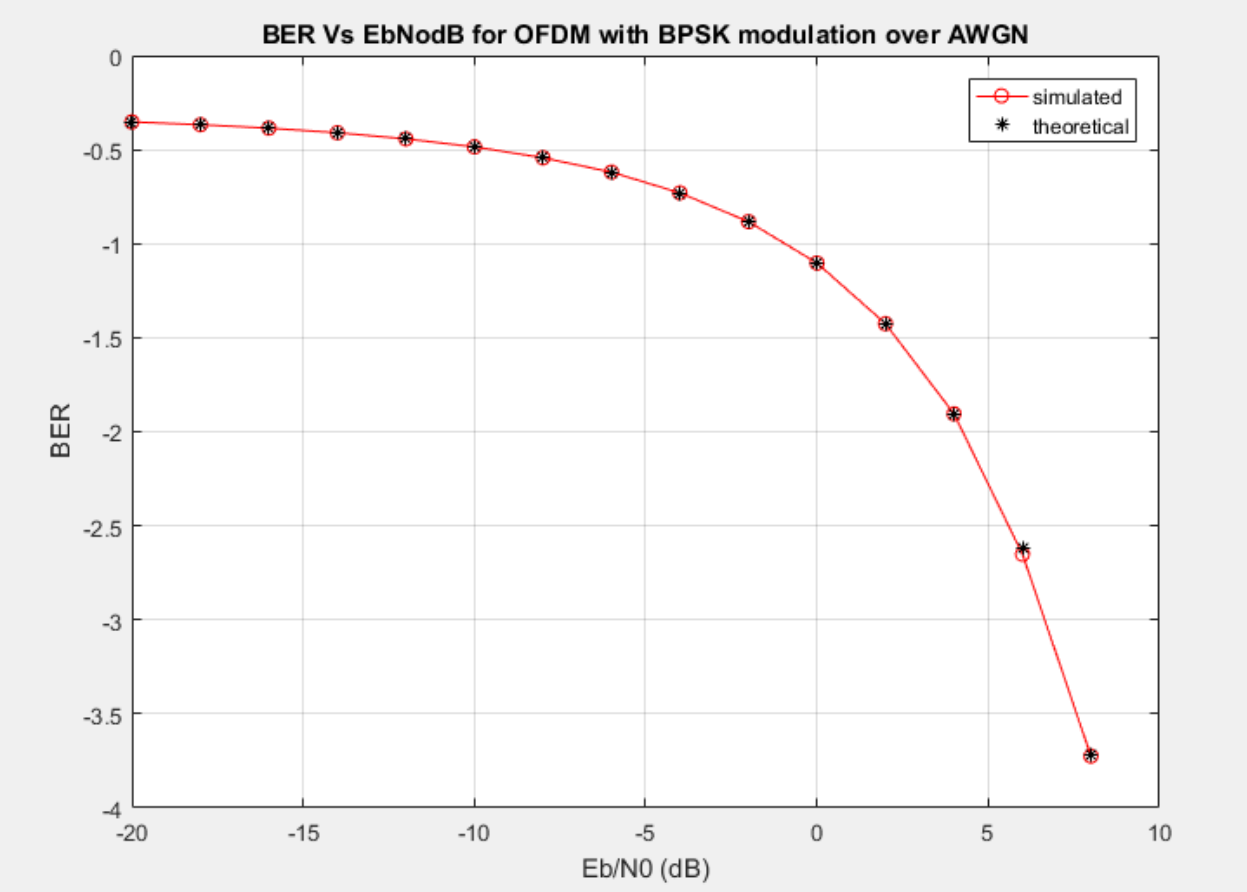
Kết quả thu được:

Trường hợp : nSym=10^4, ofdmBW=20*10^6



Hình 1. *Trường hợp : nSym=10^4, ofdmBW=20*10^6*

Trường hợp: nSym = 20000; ofdmBW = 4e6;



Hình 2. Trường hợp: *nSym = 20000; ofdmBW = 4e6;*

So sánh: Khi thay đổi các thông số băng thông và số lượng ký tự thì BER và SNR không thay đổi.

Kết luận: Các thông số BER và SNR không phụ thuộc vào số lượng ký tự và bang thông.

Bài tập 2: Chạy đoạn code trên, viết đoạn code Matlab trên với giá trị Nf=20, Nc=32, T=0.0001. Chạy kết quả trên một hình, so sánh và rút ra kết luận. Giải thích các đoạn lệnh được bôi đen ở trên.

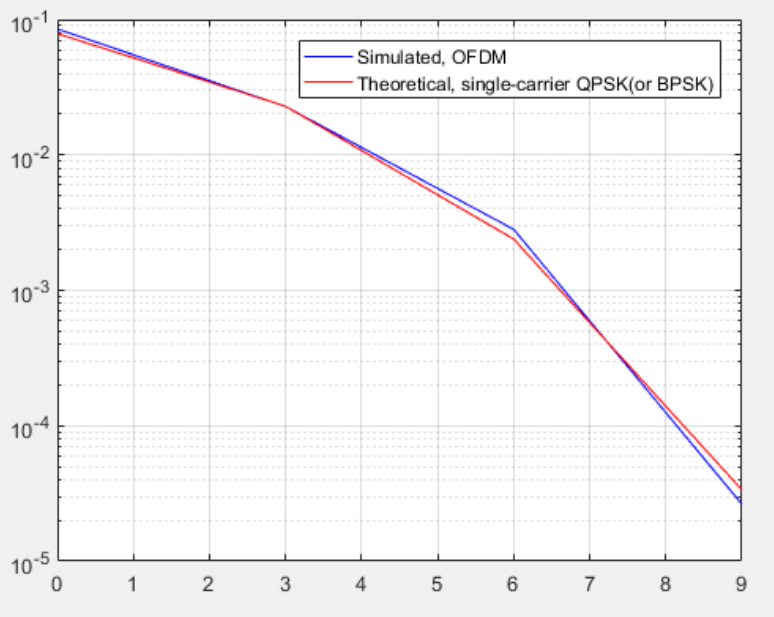
Code Matlab:

```
clc;
close all;
clear all;
Nf = 20;
Nc = 32;
T = 0.0001;
f_delta=1/T;
t_step=T/Nc;
t_vector=0:t_step:(T-t_step); %=t_step*(0:Nc-1)
Ns=length(t_vector); %Number of samples in one OFDM symbol duration T
Eb=1;
EbN0dBvector=0:3:9;
for k=0:(Nc-1)
   k_th_subcarrier=1/sqrt(T)*exp(j*2*pi*k*f_delta*t_vector);
   subcarrier_matrix(k+1,:)=k_th_subcarrier;
end
for snr_i=1:length(EbN0dBvector) %vong lap for duyet qua tung phan tu trong vector EbN0dBvector de tinh toan
cac gia tri tuong ung cua Eb/N0 va N0
   EbN0dB=EbN0dBvector(snr_i); %lay gia tri Eb/N0 tu vector EbN0dBvector tai vi tri thu snr_i va gan cho bien
EbN0dB .
   EbN0=10^(EbN0dB/10);% chuyen doi gia tri Eb/N0 tu don vi dB sang don vi ti le tin hieu/nhieu bang cach su dung
cong thuc  10^(EbN0dB/10).
   N0=Eb/EbN0;% tinh toan gia tri N0( nhieu trung binh tren moi bit) bang cach chia nang luong trung binh moi bit
Eb cho ti le tin hieu tren nhieu Eb/N0
   vn=N0/(2*t_step);
   bitcnt=0;
   errcnt=0;
   while errcnt<100
      OFDM_frame=[];
      for m=1:Nf
         data_symbols_in_OFDMsymbol=sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);
         data_symbols_in_OFDMframe(m,:)=data_symbols_in_OFDMsymbol;
         xt=zeros(1,Ns);
         for k=0:(Nc-1)%vong lap nay duyet qua cac gia tri tu 0 den Nc-1 ,he thong nay cho Nc = 32
            s_k=data_symbols_in_OFDMsymbol(k+1);%dong nay gan gia tri cua ki hieu du lieu tuong ung voi
subcarrier thu k trong 1 ki hieu OFDM cho bien s_k. Bien data_symbols_in_OFDMsymbol co the chua cac ki hieu du
kieu de truyen qua he thong OFDM  OFDM.
            xt=xt+s_k*subcarrier_matrix(k+1,:);% tinh toan tin hieu dau ra xt cua he thong OFDM . Moi lan lap ,no
them vao xt mot phan cua tin hieu dua tren gia tri cua s_k va ma tran subcarrier . subcarrier_matrix co the la nhan
ma tran
         end
      OFDM_frame=[OFDM_frame xt];
      end
      noise=sqrt(vn)*(randn(1,length(OFDM_frame))+j*randn(1,length(OFDM_frame)));
```

```
      rt_frame=OFDM_frame+noise;
      for m=1:Nf
         mth_OFDMsymbol_in_rt=rt_frame((m-1)*Ns+(1:Ns));
         for k=0:(Nc-1)
            k_th_subcarrier =subcarrier_matrix(k+1,:);
            D=t_step*sum( mth_OFDMsymbol_in_rt.*conj(k_th_subcarrier));
            estimated_data_symbols_in_OFDMframe(m,k+1)=sign(real(D))+j*sign(imag(D));
         end
      end
      Ierrs=sum(sum(real(data_symbols_in_OFDMframe)~=real(estimated_data_symbols_in_OFDMframe)));
      Qerrs=sum(sum(imag(data_symbols_in_OFDMframe)~=imag(estimated_data_symbols_in_OFDMframe)));
      errcnt=errcnt+(Ierrs+Qerrs);
      bitcnt=bitcnt+Nc*Nf*2;
   end
   BER(snr_i)=errcnt/bitcnt
   BERtheory(snr_i)=0.5 *erfc(sqrt(EbN0));
end
figure
semilogy(EbN0dBvector, BER,'b')
hold on
semilogy(EbN0dBvector, BERtheory,'r')
grid on
legend('Simulated, OFDM','Theoretical, single-carrier QPSK(or BPSK)')
```

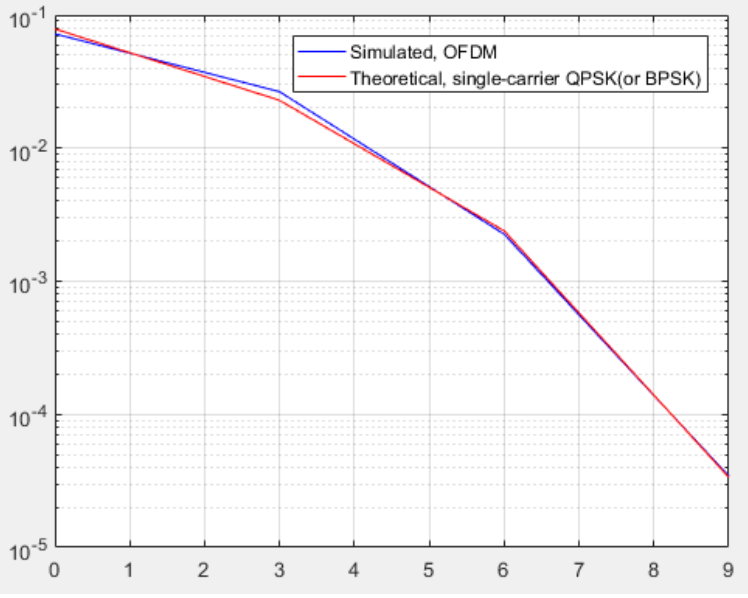
Kết quả thu được:

Trường hợp: Nf=10; Nc=16; T=10e-5;



Hình 3. *Trường hợp: Nf=10; Nc=16; T=10e-5;*

Trường hợp: Nf=20; Nc=32; T=0.0001;



Hình 4. Trường hợp: *Nf=20; Nc=32; T=0.0001;*

Kết luận:

- Hiệu quả của OFDM: Đoạn code cho thấy cách OFDM có thể giảm thiểu các tác động của nhiễu và lỗi trong môi trường truyền thông, điều quan trọng trong các hệ thống truyền dẫn không dây hiện đại.

So sánh thực nghiệm và lý thuyết:

- Việc so sánh BER thực nghiệm với BER lý thuyết cho phép đánh giá tính hiệu quả của mô hình mô phỏng và cung cấp thông tin quan trọng cho việc nghiên cứu và cải tiến hệ thống.

- Đồ thị trực quan: Sử dụng đồ thị semilog để trực quan hóa sự khác biệt giữa kết quả thực nghiệm và lý thuyết, giúp dễ dàng nhận thấy mức độ chính xác của mô phỏng.

Giải thích các đoạn lệnh được bôi đen

```
for snr_i=1:length(EbN0dBvector) %vong lap for duyet qua tung phan tu trong vector EbN0dBvector de tinh toan cac gia tri tuong ung cua Eb/N0 va N0
   EbN0dB=EbN0dBvector(snr_i); %lay gia tri Eb/N0 tu vector EbN0dBvector tai vi tri thu snr_i va gan cho bien EbN0dB .
   EbN0=10^(EbN0dB/10);% chuyen doi gia tri Eb/N0 tu don vi dB sang don vi ti le tin hieu/nhieu bang cach su dung cong thuc  10^(EbN0dB/10).
   N0=Eb/EbN0;% tinh toan gia tri N0( nhieu trung binh tren moi bit) bang cach chia nang luong trung binh moi bit Eb cho ti le tin hieu tren nhieu Eb/N0
```

```
for k=0:(Nc-1)%vong lap nay duyet qua cac gia tri tu 0 den Nc-1 ,he thong nay cho Nc = 32
        s_k=data_symbols_in_OFDMsymbol(k+1);%dong nay gan gia tri cua ki hieu du lieu tuong ung voi subcarrier thu k trong 1 ki hieu OFDM cho bien s_k. Bien data_symbols_in_OFDMsymbol co the chua cac ki hieu du kieu de truyen qua he thong OFDM  OFDM.
        xt=xt+s_k*subcarrier_matrix(k+1,:);% tinh toan tin hieu dau ra xt cua he thong OFDM . Moi lan lap ,no them vao xt mot phan cua tin hieu dua tren gia tri cua s_k va ma tran subcarrier . subcarrier_matrix co the la nhan ma tran
```

Bài tập 3: Chạy đoạn code trên, viết đoạn code Matlab trên với giá trị L=2, 3, 5. Chạy kết quả trên một hình, so sánh và rút ra kết luận. Giải thích các đoạn lệnh được bôi đen ở trên.

Code Matlab:

```
clear; clc;
Nf = 10;
L = 2;
max_delay = 1.25e-5;
decay_base = 1;
Nc = 16;
T = 8*max_delay;
t_step = (T/Nc)/16;
f_delta = 1/T;
t_vector = 0:t_step:(T-t_step); %=t_step*(0:Nc-1)
Ns = length(t_vector); %Number of samples per OFDM symbol(T seconds) before inserting Cyclic Prefix
GI = 1/4;
Ns_in_GI = ceil(Ns*GI);
Ns_total = Ns+Ns_in_GI;
Eb = 1;
EbN0dBvector = 0:3:18;
for k = 0:(Nc-1)
   subcarrier = 1/sqrt(T)*exp(j*2*pi*k*f_delta*t_vector);
   subcarrier_matrix(k+1,:) = subcarrier;
end
pilot_sk = ones(1,Nc);
xt = zeros(1,Ns);
for k = 0:(Nc-1)
   s_k = pilot_sk(k+1);
   xt = xt+s_k*subcarrier_matrix(k+1,:);
end
xt_tail = xt((Ns-Ns_in_GI+1):Ns);
pilot_OFDM_symbol = [xt_tail xt];
for snr_i = 1:length(EbN0dBvector)
   EbN0dB = EbN0dBvector(snr_i);
   EbN0 = 10^(EbN0dB/10);
   N0 = Eb/EbN0;
   vn = N0/(2*t_step); % Refer to Section 1.C and 1.D of Chapter 21.
   bitcnt = 0; errcnt=0;
      while errcnt < 1000 %Reduce 1000 to a smaller value if the simulation takes long time. The simulation error
will increase instead.
         OFDM_frame = [];
      for m=1:Nf
         datasymbols_in_OFDM_symbol = sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);
         datasymbols_in_OFDMframe(m,:) = datasymbols_in_OFDM_symbol;
         xt = zeros(1,Ns);
         for k = 0:(Nc-1)
            s_k = datasymbols_in_OFDM_symbol(k+1);
            xt = xt+s_k*subcarrier_matrix(k+1,:);
```

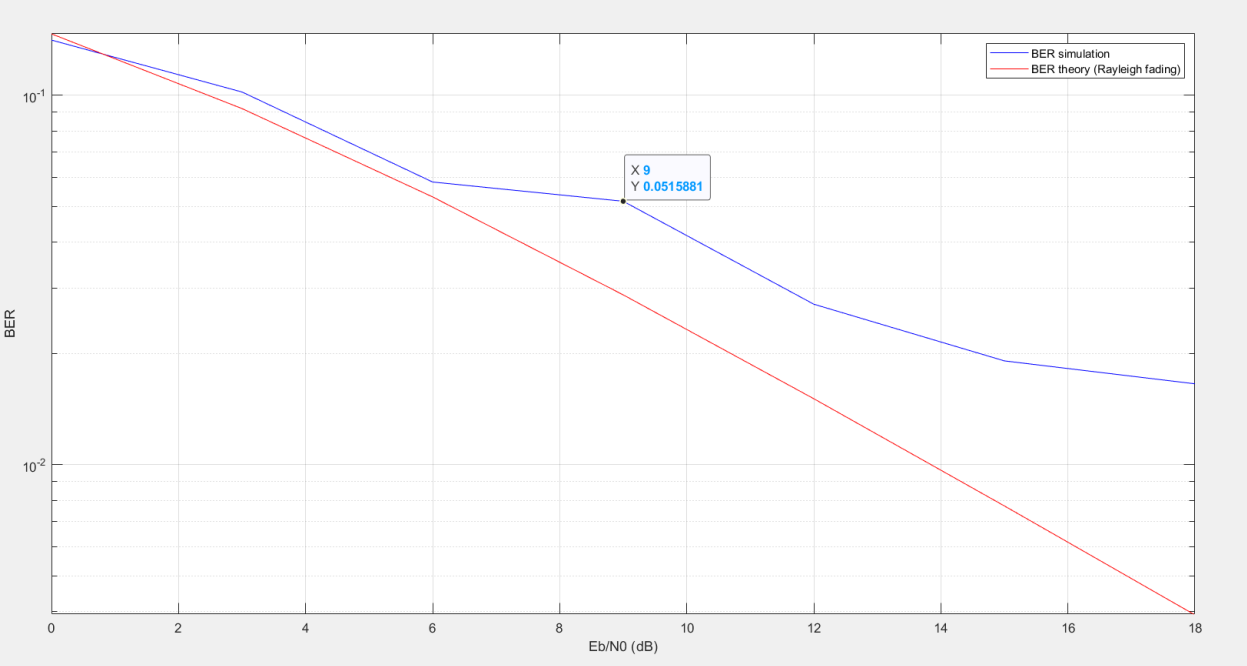
```
        end
        xt_tail = xt((Ns-Ns_in_GI+1):Ns);
        xt = [xt_tail xt];
        OFDM_frame = [OFDM_frame xt];
    end
ht = ht_mp_ch(max_delay,L,decay_base,t_step);
OFDM_frame_after_ht = conv(OFDM_frame,ht);
frame_sample_length = length(OFDM_frame_after_ht);
noise = sqrt(vn)*(randn(1,frame_sample_length)+j*randn(1,frame_sample_length));
rt_frame = OFDM_frame_after_ht+noise;
rt_pilot = conv(pilot_OFDM_symbol,ht);
for k = 0:(Nc-1)
   D = t_step*sum(rt_pilot(Ns_in_GI+(1:Ns)).*conj(subcarrier_matrix(k+1,:))/sqrt(T));
   F(k+1) = D/pilot_sk(k+1);
end
for m = 1:Nf
   first_index_of_mth_OFDMsymbol=(m-1)*Ns_total+ Ns_in_GI + 1;
   mth_OFDM_symbol_in_rt = rt_frame(first_index_of_mth_OFDMsymbol+(0:Ns-1));
for k = 0:(Nc-1)
D = t_step*sum(mth_OFDM_symbol_in_rt.*conj(subcarrier_matrix(k+1,:))/sqrt(T));
Dc = D/F(k+1);
estimated_data_symbols_in_OFDMframe(m,k+1) = sign(real(Dc))+j*sign(imag(Dc));
end
end
Ierrs = sum(sum(real(datasymbols_in_OFDMframe) ~= real(estimated_data_symbols_in_OFDMframe)));
Qerrs = sum(sum(imag(datasymbols_in_OFDMframe) ~= imag(estimated_data_symbols_in_OFDMframe)));
errcnt = errcnt+(Ierrs+Qerrs);
bitcnt = bitcnt+Nc*Nf*2;
end
BER(snr_i) = errcnt/bitcnt
%BERtheory(snr_i)=qfunc(sqrt(2*EbN0))
BERtheory(snr_i) = 1/2 - EbN0^(1/2)/(2*(EbN0 + 1)^(1/2))
end
figure

semilogy(EbN0dBvector, BER,'b')
hold on
semilogy(EbN0dBvector, BERtheory,'r')
grid
xlabel('Eb/N0 (dB)');
ylabel('BER')
legend('BER simulation','BER theory (Rayleigh fading)')
```

Code function:

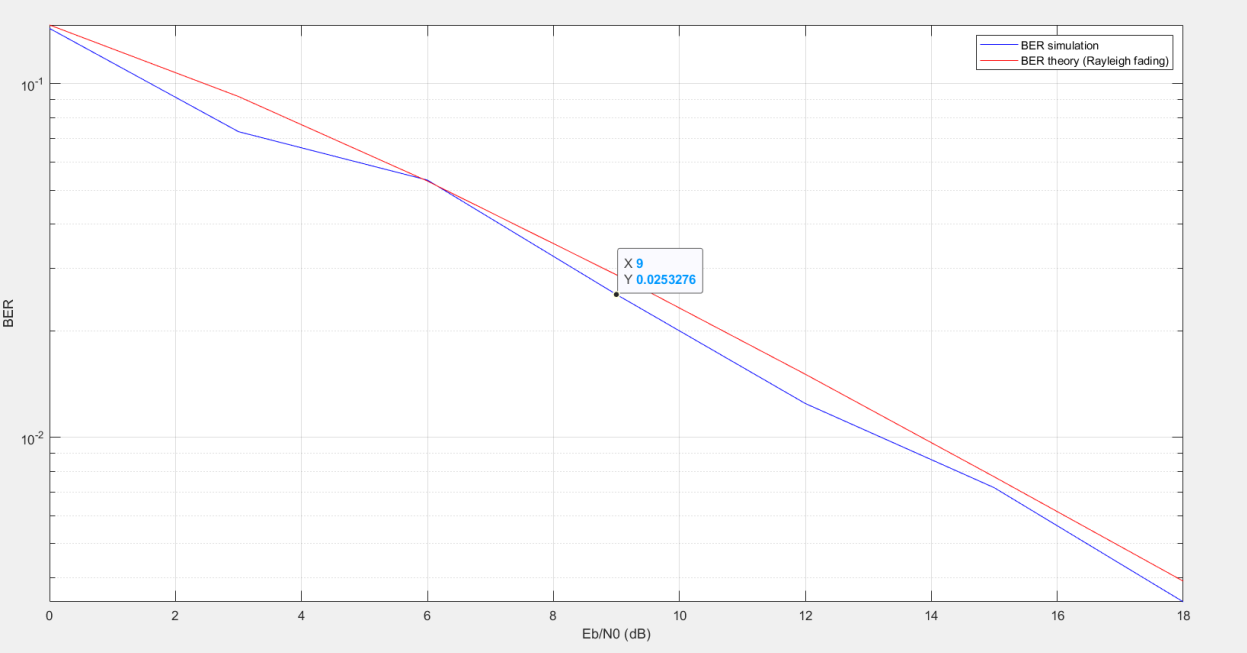
```
function impulse_response=ht_mp_ch(max_delay,L,decay_base,t_step)
t_vector=0:t_step:max_delay;
mp_tmp=0*(t_vector);
path_delays=[0 sort(rand(1,L-1)*max_delay)];
impulse_positions=floor(path_delays/t_step);
mp_tmp(impulse_positions+1)=exp(j*2*pi*rand(1,L));
```

```
mp_tmp=mp_tmp.*(decay_base.^(t_vector/max_delay));
impulse_response=mp_tmp/sqrt(sum(abs(mp_tmp).^2));
```

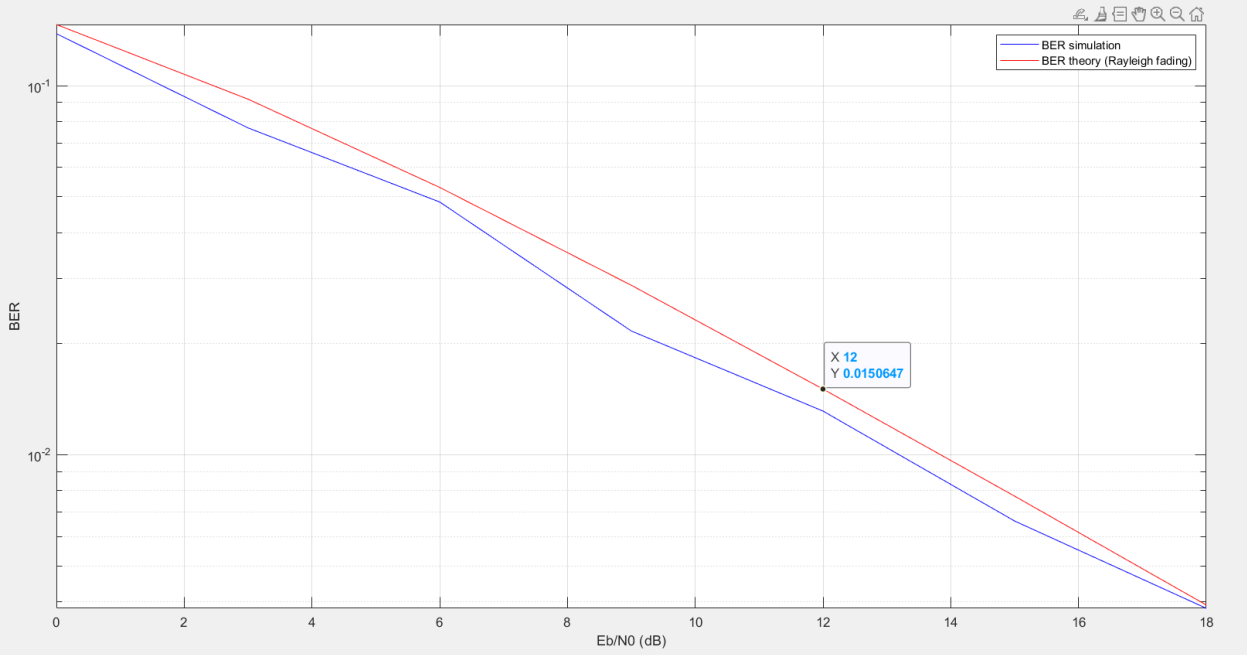
Kết quả:



Hình 5. *Trường hợp L=2*



Hình 6. *Trường hợp L = 3*



Hình 7. *Trường hợp L = 5*

- So sánh và rút ra kết luận.

+ Khi L tăng thì BER simulation thu được từ mô phỏng sẽ càng gần với BER lý thuyết (theory).

+ Việc so sánh BER lý thuyết và BER mô phỏng có thể giúp xác định mức độ chính xác của mô hình mô phỏng và xác định yêu cầu điều chỉnh thêm để phản ánh đúng hiệu suất của hệ thống.

- Giải thích các đoạn lệnh được bôi đen ở trên.

pilot_sk=ones(1,Nc);

xt=zeros(1,Ns);

for k=0:(Nc-1)

s_k=pilot_sk(k+1);

xt=xt+s_k*subcarrier_matrix(k+1,:);

end

Giải thích:

pilot_sk = ones(1, Nc); Dòng này khởi tạo một vectơ pilot_sk của kích thước 1 x Nc. Vectơ này được gọi là Pilot_sk và được sử dụng làm hệ số điều chế.

xt = zeros(1, Ns);: Dòng này khởi tạo một vectơ xt có kích thước 1 x Ns chứa đầy các số 0. Vector này sẽ giữ tín hiệu thu được sau khi điều chế.

for k = 0 : (Nc - 1): Dòng này bắt đầu một vòng lặp lặp qua từng chỉ mục từ 0 đến Nc – 1

s_k = Pilot_sk(k + 1);: Dòng này lấy giá trị của Pilot_sk tại chỉ số k + 1 và gán cho biến s_k. Giá trị này đại diện cho hệ số điều chế.

xt = xt + s_k * subcarrier_matrix(k + 1, :);: Đường này thực hiện điều chế. Nó truy xuất hàng thứ k + 1 của ma trận sóng mang con, đại diện

cho dạng sóng sóng mang con và nhân nó với hệ số s_k tương ứng. Sau đó, nó thêm kết quả vào vectơ xt.

OFDM_frame=[];

for m=1:Nf

Giải thích:

OFDM_frame = [];: Dòng này khởi tạo một ma trận trống có tên OFDM_frame. Ma trận này sẽ được sử dụng để lưu trữ toàn bộ khung OFDM, bao gồm nhiều ký hiệu OFDM.

for m = 1 : Nf: Dòng này bắt đầu một vòng lặp lặp từ 1 đến Nf, trong đó Nf biểu thị số lượng ký hiệu OFDM trong khung. Vòng lặp này chịu trách nhiệm tạo ra từng ký hiệu OFDM và tập hợp chúng thành khung hoàn chỉnh.

datasymbols_in_OFDM symbol=sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);

datasymbols_in_OFDMframe(m,:)=data symbols_in_OFDM symbol;

Giải thích:

datasymbols_in_OFDMsymbol: Dòng này tạo ra một vectơ có độ dài Nc chứa các ký hiệu QPSK (Khóa dịch chuyển pha cầu phương) ngẫu nhiên. Mỗi ký hiệu được tạo bằng cách lấy dấu của một số ngẫu nhiên trong khoảng từ -0,5 đến 0,5 cho cả phần thực và phần ảo. Các ký hiệu này đại diện cho các ký hiệu dữ liệu cho ký hiệu OFDM hiện tại.

datasymbols_in_OFDMframe(m, :) = datasymbols_in_OFDMsymbol;: Dòng này lưu trữ các ký hiệu QPSK được tạo cho ký hiệu OFDM hiện tại m trong một ma trận có tên datasymbols_in_OFDMframe. Mỗi hàng của ma trận này đại diện cho một ký hiệu OFDM.

rt_pilot=conv(pilot_OFDM symbol,ht);

for k=0:(Nc-1)

D=t_step*sum(rt_pilot(Ns_in_GI+(1:Ns)).*conj(subcarrier_matrix(k+1, :))/

sqrt(T));

F(k+1)=D/pilot_sk(k+1);

End

Giải thích:

rt_pilot = conv(pilot_OFDMsymbol, ht);: Dòng này kết hợp ký hiệu OFDM Pilot_OFDMsymbol với đáp ứng xung kênh ht. Tích chập này mô phỏng tác động của kênh lên các ký hiệu hoa tiêu được truyền đi và tạo ra các ký hiệu hoa tiêu nhận được rt_pilot.

Vòng lặp tiếp theo tính toán hệ số đáp ứng kênh miền tần số F(k + 1) cho mỗi sóng mang con. Nó lặp qua từng chỉ mục sóng mang con k và với mỗi chỉ mục:

D = t_step * sum(rt_pilot(Ns_in_GI + (1 : Ns)) .* conj(subcarrier_matrix(k + 1, :)) / sqrt(T));: Dòng này tính toán mối tương quan giữa các ký hiệu hoa tiêu nhận được và liên hợp của sóng mang con tương ứng. Sau đó, nó chia tỷ lệ kết quả theo nghịch đảo của căn bậc hai của khoảng thời gian ký hiệu T và bước thời gian t_step.

F(k + 1) = D / Pilot_sk(k + 1);: Dòng này tính toán hệ số đáp ứng kênh miền tần số F(k + 1) bằng cách chia mối tương quan D cho ký hiệu hoa tiêu đã biết Pilot_sk(k + 1) cho sóng mang phụ hiện tại.

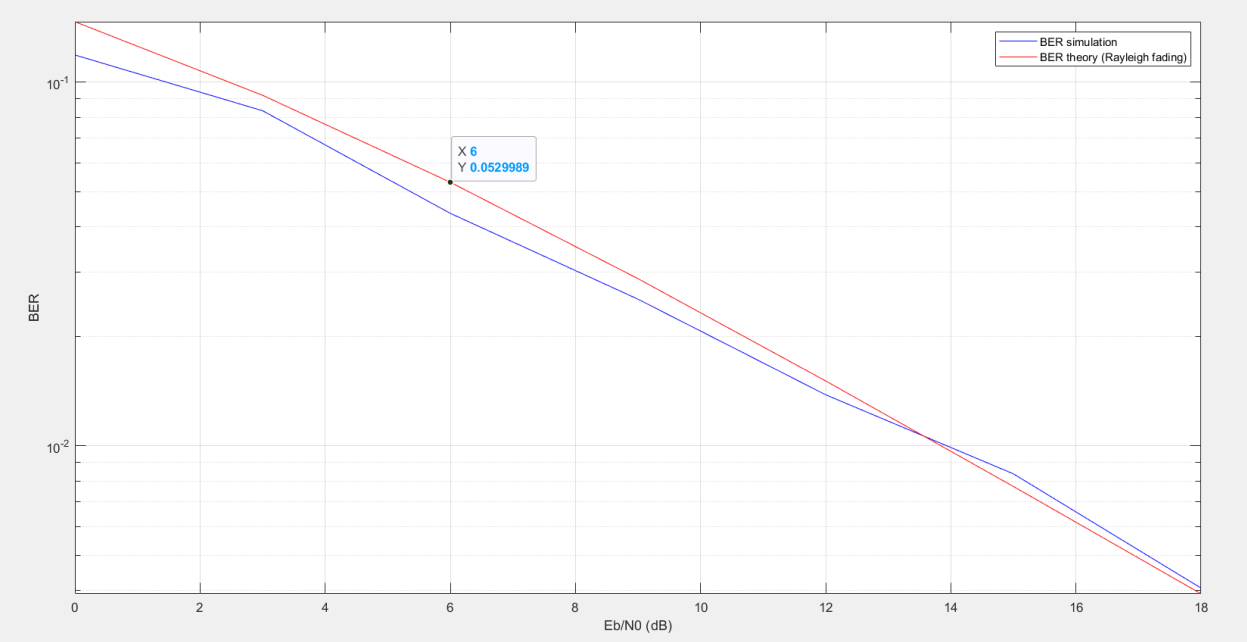
Bài tập 4: Thực hiện kết quả Matlab đối với trường hợp: L = 5, GI = 1/4, and decay_base = 1 khi Nc = 4, 16, and 64. Chạy các kết quả ở trên và cho vào báo cáo.

Code Matlab:

```
clear; clc;
Nf = 10;
L = 5;
max_delay = 1.25e-5;
decay_base = 1;
Nc = 4;
T = 8*max_delay;
t_step = (T/Nc)/16;
f_delta = 1/T;
t_vector = 0:t_step:(T-t_step); %=t_step*(0:Nc-1)
Ns = length(t_vector); %Number of samples per OFDM symbol(T seconds) before inserting Cyclic Prefix
GI = 1/4;
Ns_in_GI = ceil(Ns*GI);
Ns_total = Ns+Ns_in_GI;
Eb = 1;
EbN0dBvector = 0:3:18;
for k = 0:(Nc-1)
   subcarrier = 1/sqrt(T)*exp(j*2*pi*k*f_delta*t_vector);
   subcarrier_matrix(k+1,:) = subcarrier;
end
pilot_sk = ones(1,Nc);
xt = zeros(1,Ns);
for k = 0:(Nc-1)
   s_k = pilot_sk(k+1);
   xt = xt+s_k*subcarrier_matrix(k+1,:);
end
xt_tail = xt((Ns-Ns_in_GI+1):Ns);
pilot_OFDM_symbol = [xt_tail xt];
for snr_i = 1:length(EbN0dBvector)
   EbN0dB = EbN0dBvector(snr_i);
   EbN0 = 10^(EbN0dB/10);
   N0 = Eb/EbN0;
   vn = N0/(2*t_step); % Refer to Section 1.C and 1.D of Chapter 21.
   bitcnt = 0; errcnt=0;
      while errcnt < 1000 %Reduce 1000 to a smaller value if the simulation takes long time. The simulation error
will increase instead.
         OFDM_frame = [];
      for m=1:Nf
         datasymbols_in_OFDM_symbol = sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);
         datasymbols_in_OFDMframe(m,:) = datasymbols_in_OFDM_symbol;
         xt = zeros(1,Ns);
         for k = 0:(Nc-1)
```

```
            s_k = datasymbols_in_OFDM_symbol(k+1);
            xt = xt+s_k*subcarrier_matrix(k+1,:);
        end
        xt_tail = xt((Ns-Ns_in_GI+1):Ns);
        xt = [xt_tail xt];
        OFDM_frame = [OFDM_frame xt];
    end
ht = ht_mp_ch(max_delay,L,decay_base,t_step);
OFDM_frame_after_ht = conv(OFDM_frame,ht);
frame_sample_length = length(OFDM_frame_after_ht);
noise = sqrt(vn)*(randn(1,frame_sample_length)+j*randn(1,frame_sample_length));
rt_frame = OFDM_frame_after_ht+noise;
rt_pilot = conv(pilot_OFDM_symbol,ht);
for k = 0:(Nc-1)
   D = t_step*sum(rt_pilot(Ns_in_GI+(1:Ns)).*conj(subcarrier_matrix(k+1,:))/sqrt(T));
   F(k+1) = D/pilot_sk(k+1);
end
for m = 1:Nf
   first_index_of_mth_OFDMsymbol=(m-1)*Ns_total+ Ns_in_GI + 1;
   mth_OFDM_symbol_in_rt = rt_frame(first_index_of_mth_OFDMsymbol+(0:Ns-1));
for k = 0:(Nc-1)
D = t_step*sum(mth_OFDM_symbol_in_rt.*conj(subcarrier_matrix(k+1,:))/sqrt(T));
Dc = D/F(k+1);
estimated_data_symbols_in_OFDMframe(m,k+1) = sign(real(Dc))+j*sign(imag(Dc));
end
end
Ierrs = sum(sum(real(datasymbols_in_OFDMframe) ~= real(estimated_data_symbols_in_OFDMframe)));
Qerrs = sum(sum(imag(datasymbols_in_OFDMframe) ~= imag(estimated_data_symbols_in_OFDMframe)));
errcnt = errcnt+(Ierrs+Qerrs);
bitcnt = bitcnt+Nc*Nf*2;
end
BER(snr_i) = errcnt/bitcnt
%BERtheory(snr_i)=qfunc(sqrt(2*EbN0))
BERtheory(snr_i) = 1/2 - EbN0^(1/2)/(2*(EbN0 + 1)^(1/2))
end
figure

semilogy(EbN0dBvector, BER,'b')
hold on
semilogy(EbN0dBvector, BERtheory,'r')
grid
xlabel('Eb/N0 (dB)');
ylabel('BER')
legend('BER simulation','BER theory (Rayleigh fading)')
```

Kết quả:



Hình 8. *Trường hợp L = 5, GI = 1/4, and decay_base = 1 khi Nc = 4*

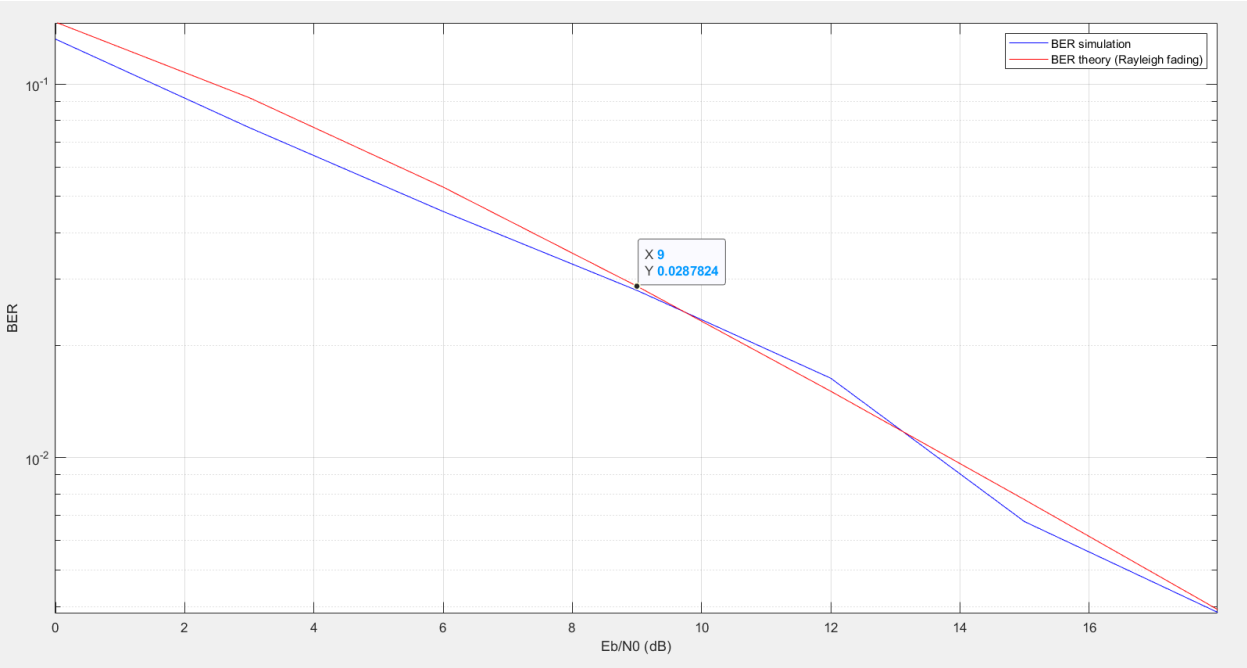
Code Matlab

```
clear; clc;
Nf = 10;
L = 5;
max_delay = 1.25e-5;
decay_base = 1;
Nc = 16;
T = 8*max_delay;
t_step = (T/Nc)/16;
f_delta = 1/T;
t_vector = 0:t_step:(T-t_step); %=t_step*(0:Nc-1)
Ns = length(t_vector); %Number of samples per OFDM symbol(T seconds) before inserting Cyclic Prefix
GI = 1/4;
Ns_in_GI = ceil(Ns*GI);
Ns_total = Ns+Ns_in_GI;
Eb = 1;
EbN0dBvector = 0:3:18;
for k = 0:(Nc-1)
   subcarrier = 1/sqrt(T)*exp(j*2*pi*k*f_delta*t_vector);
   subcarrier_matrix(k+1,:) = subcarrier;
end
pilot_sk = ones(1,Nc);
xt = zeros(1,Ns);
for k = 0:(Nc-1)
```

```
   s_k = pilot_sk(k+1);
   xt = xt+s_k*subcarrier_matrix(k+1,:);
end
xt_tail = xt((Ns-Ns_in_GI+1):Ns);
pilot_OFDM_symbol = [xt_tail xt];
for snr_i = 1:length(EbN0dBvector)
   EbN0dB = EbN0dBvector(snr_i);
   EbN0 = 10^(EbN0dB/10);
   N0 = Eb/EbN0;
   vn = N0/(2*t_step); % Refer to Section 1.C and 1.D of Chapter 21.
   bitcnt = 0; errcnt=0;
      while errcnt < 1000 %Reduce 1000 to a smaller value if the simulation takes long time. The simulation error
will increase instead.
         OFDM_frame = [];
      for m=1:Nf
         datasymbols_in_OFDM_symbol = sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);
         datasymbols_in_OFDMframe(m,:) = datasymbols_in_OFDM_symbol;
         xt = zeros(1,Ns);
         for k = 0:(Nc-1)
            s_k = datasymbols_in_OFDM_symbol(k+1);
            xt = xt+s_k*subcarrier_matrix(k+1,:);
         end
         xt_tail = xt((Ns-Ns_in_GI+1):Ns);
         xt = [xt_tail xt];
         OFDM_frame = [OFDM_frame xt];
      end
ht = ht_mp_ch(max_delay,L,decay_base,t_step);
OFDM_frame_after_ht = conv(OFDM_frame,ht);
frame_sample_length = length(OFDM_frame_after_ht);
noise = sqrt(vn)*(randn(1,frame_sample_length)+j*randn(1,frame_sample_length));
rt_frame = OFDM_frame_after_ht+noise;
rt_pilot = conv(pilot_OFDM_symbol,ht);
for k = 0:(Nc-1)
   D = t_step*sum(rt_pilot(Ns_in_GI+(1:Ns)).*conj(subcarrier_matrix(k+1,:))/sqrt(T));
   F(k+1) = D/pilot_sk(k+1);
end
for m = 1:Nf
   first_index_of_mth_OFDMsymbol=(m-1)*Ns_total+ Ns_in_GI + 1;
   mth_OFDM_symbol_in_rt = rt_frame(first_index_of_mth_OFDMsymbol+(0:Ns-1));
for k = 0:(Nc-1)
D = t_step*sum(mth_OFDM_symbol_in_rt.*conj(subcarrier_matrix(k+1,:))/sqrt(T));
Dc = D/F(k+1);
estimated_data_symbols_in_OFDMframe(m,k+1) = sign(real(Dc))+j*sign(imag(Dc));
end
end
Ierrs = sum(sum(real(datasymbols_in_OFDMframe) ~= real(estimated_data_symbols_in_OFDMframe)));
Qerrs = sum(sum(imag(datasymbols_in_OFDMframe) ~= imag(estimated_data_symbols_in_OFDMframe)));
errcnt = errcnt+(Ierrs+Qerrs);
bitcnt = bitcnt+Nc*Nf*2;
end
BER(snr_i) = errcnt/bitcnt
%BERtheory(snr_i)=qfunc(sqrt(2*EbN0))
BERtheory(snr_i) = 1/2 - EbN0^(1/2)/(2*(EbN0 + 1)^(1/2))
```

```
end
figure

semilogy(EbN0dBvector, BER,'b')
hold on
semilogy(EbN0dBvector, BERtheory,'r')
grid
xlabel('Eb/N0 (dB)');
ylabel('BER')
legend('BER simulation','BER theory (Rayleigh fading)')
```

Kết quả:



Hình 9. *Trường hợp L = 5, GI = 1/4, and decay_base = 1 khi Nc = 16*

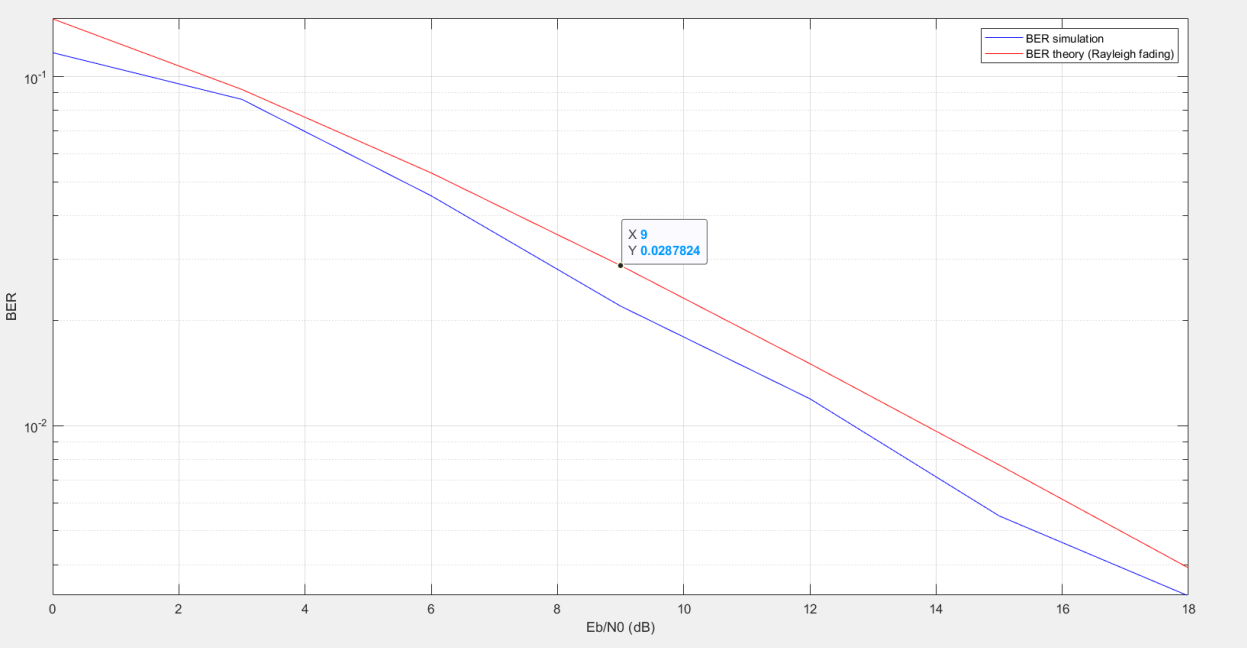
Code Matlab:

```
clear; clc;
Nf = 10;
L = 5;
max_delay = 1.25e-5;
decay_base = 1;
Nc = 64;
T = 8*max_delay;
t_step = (T/Nc)/16;
f_delta = 1/T;
t_vector = 0:t_step:(T-t_step); %=t_step*(0:Nc-1)
Ns = length(t_vector); %Number of samples per OFDM symbol(T seconds) before inserting Cyclic Prefix
GI = 1/4;
```

```
Ns_in_GI = ceil(Ns*GI);
Ns_total = Ns+Ns_in_GI;
Eb = 1;
EbN0dBvector = 0:3:18;
for k = 0:(Nc-1)
   subcarrier = 1/sqrt(T)*exp(j*2*pi*k*f_delta*t_vector);
   subcarrier_matrix(k+1,:) = subcarrier;
end
pilot_sk = ones(1,Nc);
xt = zeros(1,Ns);
for k = 0:(Nc-1)
   s_k = pilot_sk(k+1);
   xt = xt+s_k*subcarrier_matrix(k+1,:);
end
xt_tail = xt((Ns-Ns_in_GI+1):Ns);
pilot_OFDM_symbol = [xt_tail xt];
for snr_i = 1:length(EbN0dBvector)
   EbN0dB = EbN0dBvector(snr_i);
   EbN0 = 10^(EbN0dB/10);
   N0 = Eb/EbN0;
   vn = N0/(2*t_step); % Refer to Section 1.C and 1.D of Chapter 21.
   bitcnt = 0; errcnt=0;
      while errcnt < 1000 %Reduce 1000 to a smaller value if the simulation takes long time. The simulation error
will increase instead.
         OFDM_frame = [];
      for m=1:Nf
         datasymbols_in_OFDM_symbol = sign(rand(1,Nc)-0.5)+j*sign(rand(1,Nc)-0.5);
         datasymbols_in_OFDMframe(m,:) = datasymbols_in_OFDM_symbol;
         xt = zeros(1,Ns);
         for k = 0:(Nc-1)
            s_k = datasymbols_in_OFDM_symbol(k+1);
            xt = xt+s_k*subcarrier_matrix(k+1,:);
         end
         xt_tail = xt((Ns-Ns_in_GI+1):Ns);
         xt = [xt_tail xt];
         OFDM_frame = [OFDM_frame xt];
      end
ht = ht_mp_ch(max_delay,L,decay_base,t_step);
OFDM_frame_after_ht = conv(OFDM_frame,ht);
frame_sample_length = length(OFDM_frame_after_ht);
noise = sqrt(vn)*(randn(1,frame_sample_length)+j*randn(1,frame_sample_length));
rt_frame = OFDM_frame_after_ht+noise;
rt_pilot = conv(pilot_OFDM_symbol,ht);
for k = 0:(Nc-1)
   D = t_step*sum(rt_pilot(Ns_in_GI+(1:Ns)).*conj(subcarrier_matrix(k+1,:))/sqrt(T));
   F(k+1) = D/pilot_sk(k+1);
end
for m = 1:Nf
   first_index_of_mth_OFDMsymbol=(m-1)*Ns_total+ Ns_in_GI + 1;
   mth_OFDM_symbol_in_rt = rt_frame(first_index_of_mth_OFDMsymbol+(0:Ns-1));
for k = 0:(Nc-1)
D = t_step*sum(mth_OFDM_symbol_in_rt.*conj(subcarrier_matrix(k+1,:))/sqrt(T));
Dc = D/F(k+1);
```

```
estimated_data_symbols_in_OFDMframe(m,k+1) = sign(real(Dc))+j*sign(imag(Dc));
end
end
Ierrs = sum(sum(real(datasymbols_in_OFDMframe) ~= real(estimated_data_symbols_in_OFDMframe)));
Qerrs = sum(sum(imag(datasymbols_in_OFDMframe) ~= imag(estimated_data_symbols_in_OFDMframe)));
errcnt = errcnt+(Ierrs+Qerrs);
bitcnt = bitcnt+Nc*Nf*2;
end
BER(snr_i) = errcnt/bitcnt
%BERtheory(snr_i)=qfunc(sqrt(2*EbN0))
BERtheory(snr_i) = 1/2 - EbN0^(1/2)/(2*(EbN0 + 1)^(1/2))
end
figure

semilogy(EbN0dBvector, BER,'b')
hold on
semilogy(EbN0dBvector, BERtheory,'r')
grid
xlabel('Eb/N0 (dB)');
ylabel('BER')
legend('BER simulation','BER theory (Rayleigh fading)')
```

Kết quả:



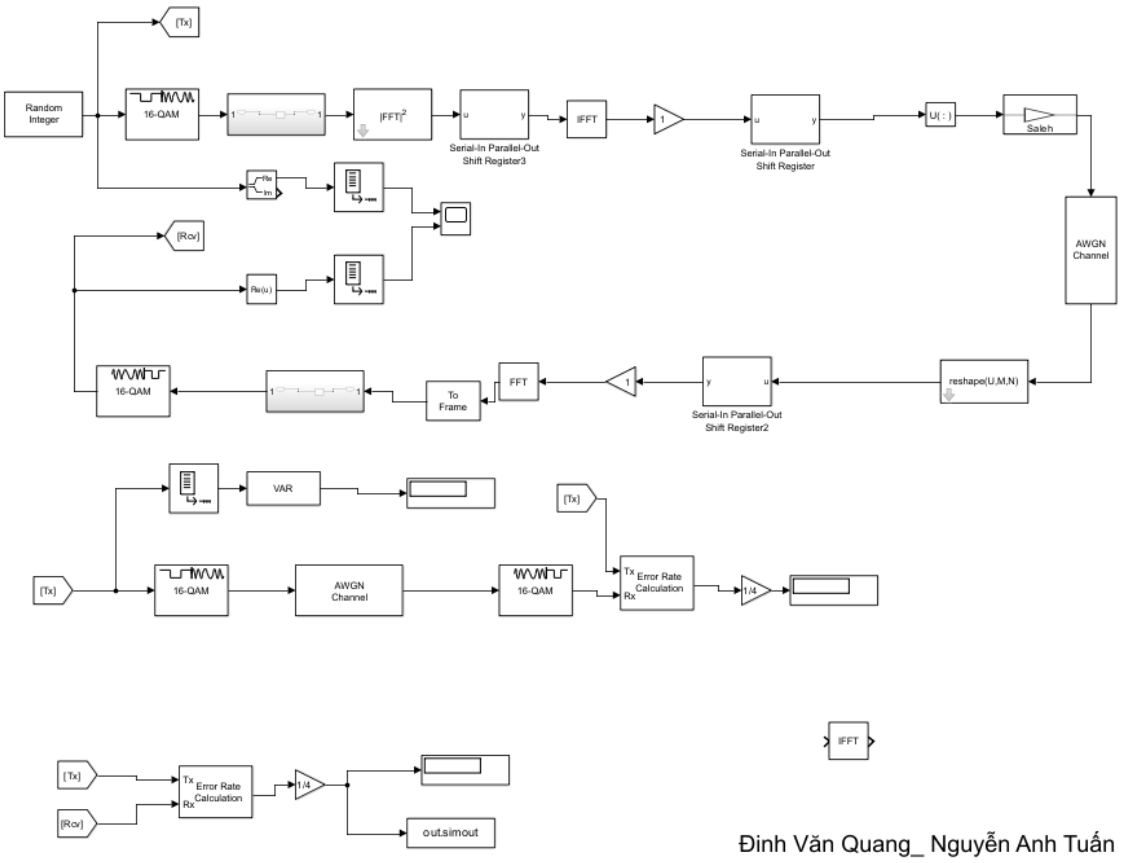
Hình 10. *Trường hợp L = 5, GI = 1/4, and decay_base = 1 khi Nc = 64*

Nhận xét:

- Khi Nc tăng thì BER thu được từ mô phỏng gần như không đổi với BER lý thuyết

- Việc so sánh BER lý thuyết và BER mô phỏng có thể giúp xác định mức độ chính xác của mô hình mô phỏng và xác định yêu cầu điều chỉnh thêm để phản ánh đúng hiệu suất của hệ thống.

Bài tập 5: Sinh viên thực hiện mô hình Simulink ở slide dưới đây cho trường hợp Eb/N0=10dB và 3dB.



Hình 11. *Mô hình Simulink*