COURSE:

EE 5368 Wireless Communications & IoT.

#Project 2 Report

System Level Design for A Mobile Wireless User Terminal.

Instructor: Professor QilianLiang

Fall:2022

Date of submission:15 December 2022

Group Project by

Chinmayee Makaraju

Mav ID :1002091569

Kavya Goli

Mav ID:1002040718

INTRODUCTION

Communication via wireless(or wirelessly) occurs when information is transferred from two or more sites without using an electric conductor as a medium.Wireless technology most commonly uses radio waves.Deep-space radio communications can cover millions of kilometers, while Bluetooth can reach as far as a few meters.As well as two-way radios, mobile phones, PDAs, and wireless networks, wireless networking covers a vast array of fixed, mobile, and portable applications.The use of radio wireless technology has been found in a wide variety of household products, including GPS devices, garage door openers, wireless computer mice, keyboards, and headsets, headphones, radio receivers, satellite television, broadcast television, and cordless telephones.Sources of messages may be humans, machines, television images, or data, depending on the source. In input transducer is used to convert source into an electrical waveform which is known as baseband signal. Baseband signal is changed through transmitter for efficient transmission. A pre-emphasizer, a sampler, a quantizer, a coder, and a modulator are some of the subsystems found in a transmitter. Output will be send through a medium which is channel. It might be a wire, a coaxial cable, an optical fiber, or a radio connection, among other things.There are two types of modern communication systems based on channel type: Wired communication and wireless communication systems. The channel and transmitter will do the alterations of the signals reprocessed from the receiver. The output channel will receive decipher message from the distorted and noisy signal from the receiver. The receiving point will include a decoder, a filter, a de-emphasizer and demodulator. The electrical. Signal will get back to its original form with the help of output transducer. To combat distortion and noise, transmitters and receivers are meticulously built. The physical layer will convey data in efficient and accurate manners.

Project Questions :

In this project, you (a team up to 3 graduate students) are requested to perform system-level design for a mobile wireless user terminal using block phase estimation algorithm proposed in [1]. The specifications for its general packet data channel (GPDCH, a logic channel) are listed in the following:

- Modulation: QPSK with bits to symbol mapping:
  $00 \rightarrow 1, 01 \rightarrow j, 11 \rightarrow -1, 10 \rightarrow -j$.
- Burst structure:

Guard Symbols (3 symbols)	Unique Word (48 symbols)	Private inf symbols (payload) (500 symbols)	Guard Symbols (3 symbols)

  and transmitting such a burst needs 0.5ms.
- The channel bandwidth is 1.44MHz, and Rician flat fading is assumed.
- 16 samples/symbol should be used based on hardware requirements.
- The roll-off factor for square root raised cosine filter is 0.3.

Please design a simulation reference system for GPDCH subject to Rician flat fading using MATLAB, and design a demodulator for such a fading channel with QPSK modulation. Provide a report describe your design with the following performance plots:

(1) When the Rician flat fading channel $K = 7dB$, doppler shift (fading bandwidth) $f_d = 20Hz$, plot the raw bit error rate (BER) at $\frac{E_b}{N_0}$=1, 2, …,7dB obtained in your simulation and compare it with the theoretical raw BER at 1, 2, …, 6,7 dB.

```
BER=[0.09 0.0699 0.0527 0.0387 0.0277 0.0193 0.0135];
```

(2) When the Rician flat fading channel $K = 12dB$, $f_d = 100Hz$, plot the raw bit error rate (BER) at $\frac{E_b}{N_0}$=1, 2, …,6, 7dB obtained in your simulation and compare it with the theoretical raw BER at 1, 2, …, 6, 7 dB.

```
BER=[0.06565 0.04664 0.03124 0.01958 0.011396 0.0061246,0.00302];
```

(3) When the Rician flat fading channel $K = 200dB$, $f_d = 0Hz$, plot the raw bit error rate (BER) at $\frac{E_b}{N_0}$ =1, 2, …,6dB obtained in your simulation and compare it with the theoretical raw BER, $Q(\sqrt{\frac{2E_b}{N_0}})$

(4) If four-path diversity with maximal ratio combining (MRC) is used, repeat (1)(2)(3), and observe how many dB gains have achieved in each case.

(5) If Alamouti codes is used for a 2x1 multiple-input and single output (MISO) system, repeat (1)(2)(3), and observe how many dB gains have achieved in each case.

(6) If convolutional codes with coding rate ½ and with connections (in octal number) represented in binary [001 011 011] and [001 111 001] in the encoder and viterbi decoder are used in the design, repeat (1)(2)(3), and observe how many dB gains have achieved in each case.

(7) If puncturing (b2 is punctured in every four bits b0-b3) and block interleaver (10×5) are used in the design, what's the performance for the above three channels? How many dB losses comparing to the same channel in (6)? What's the main advantage of this scheme?

(8) In the above systematic studies, what conclusions can be drawn? Please prepare and submit a technical document. The submission to Canvas includes all m-files and technical document in a zip file.

Description:

Modulation:

In the project, we implemented QPSK modulation, which consists of two information bits per symbol and is carried out using the following mapping:

00 ⟶ 1

01 ⟶ j

11 ⟶ -1

10 ⟶ -j

Channel:

A Rician flat fading channel was used in this project. The fading factor (k) and the Doppler frequency are the factors that define this channel (fd). Jake's model is used for modelingof the Rician channel.

Pulse shaping:

This filter is inserted before the channel and is used to build a continuous waveformusing convolution methods. The square root raised cosine filter has a roll-off factor of 0.35.

Burst Generator:

This block creates the data payload, as well as the guard and the unique word. The burst has a total length of 1108 bits. The payload is 1000 bits long,with 6 guard bits at either end and 96 unique word bits.

Up sampling:

In up sampling it inserts 15 zeros between two symbols, we must multiply the 554 symbols by 16. This makes that the transition between two symbols seamless.

Matched Filter & Down Sampling:

This filter, like the channel, works on the idea of convolution and is used for matching. The square root raised cosine filter has a roll-off factor of 0.35. The downsampler extracts symbols in multiples of 16 characters. The zeros between two symbols are removed, and the burst length is decreased to 554 symbols.

Demodulation and Block Phase Estimation:

In order to correctly execute channel compensation, the compensating angle must be assessed. This is accomplished with the aid of Viterbi paper.

The following are the steps in estimating the block phase:

a. window size of length N (in our example, 60) with a step size of N is chosen (10 in our case)
b. The received signal (z) is increased to the power four after demodulation and down sampling toyield y.
c. Using the formula below, the phase angle is determined across the window size:

$$\theta_{ph} = 14 \angle \tan^{-1}(imareal((yy)))$$

d. Because theta might have values ranging from -4 to +4, the phase angle determined by thisapproach is ambiguous. The accurate phase rotation is resolved using the calculation of the unique word in the burst structure over the length L of unique word: $\theta_{ac} = \theta_{ph} + \frac{k\pi}{4}$

e. The correct phase rotation is determined by multiplying the unique word in the burst structure bythe length L of the unique word:
f. Based on the angle determined using the unique word above, the real value of theta is calculatedthat is closest to the calculated angle of the unique word and is referred to as theta 1.
g. The block window is then increased to [M, M+N-1], and the phase candidate theta h is derived inthe same way.
h. The theta 2 is computed by selecting the one closest to theta 1 I from the theta.
i. Steps a-g above are continued until the entire length of the burst structure is covered.
j. After that, linear interpolation is used to obtain the phase rotation for each symbol in the burst. The demodulation is performed via hard slicing after receiving the phase rotation for each symbol from the preceding operation.

Hard Slicing: Network slicing is divided into two types: hard and soft slicing. Hard slicing network slices must be totally separated from one other, but soft slicing network slices may share certain network resources. Further it can be understood with the diagram given below;

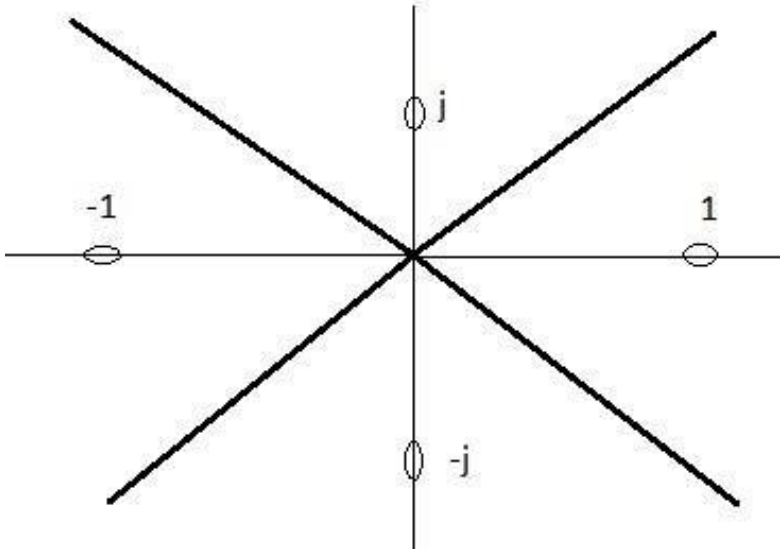


Fig. Hard Slicer

Burst Extraction and BER Calculation: In this block, we remove the unneeded guard and unique word, leaving only the 1200-bit payload. We compute the BER by comparing the produced payload at the transmitter to the received burst at the output of the burst extractor.

➢ Code For Question 1

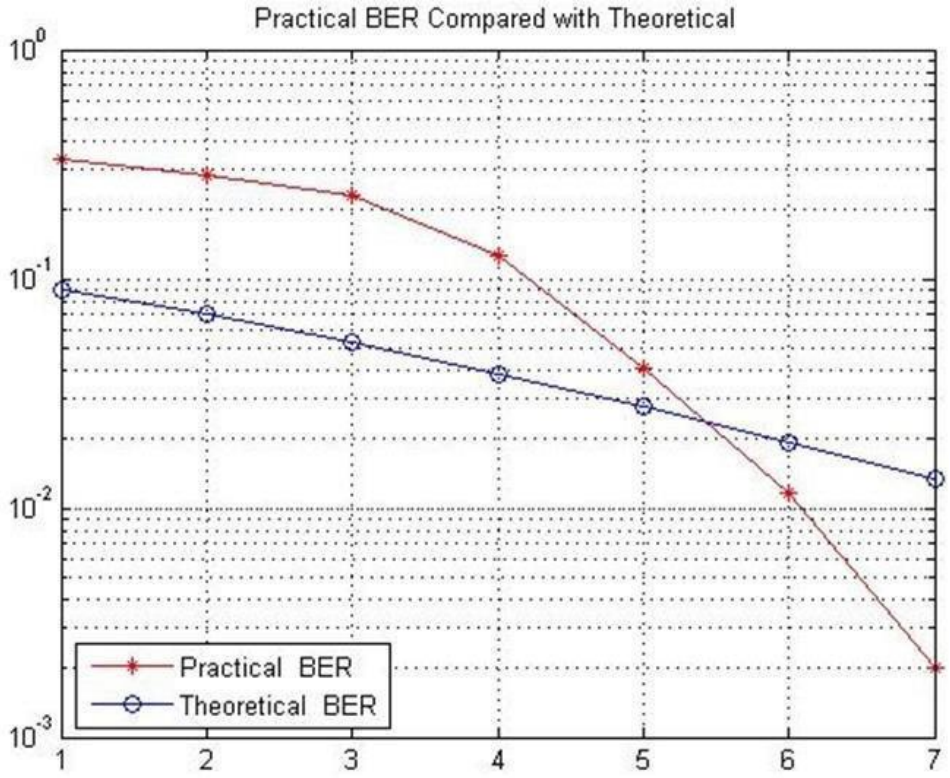
Main:

```
clc;
SNR =1:7
for  count=1:1000
Payload = randi([0,1],1000,1);
Gaurd1= randi([0,1],6,1);Gaurd2=randi([0,1],6,1);
UniqueWord=randi([0,1],96,1);
burst=[Gaurd1;UniqueWord;Payload;Gaurd2]; burst_len= length (burst);
%  QPSK  Modulation
for  m=1:2:length(burst)-1
if  burst(m)==0  &  burst(m+1)==0  %00  first  quadrant modword((m+1)/2)=sqrt(0.5)+j*sqrt(0.5);
elseif  burst(m)==1  &  burst(m+1)==0  %10  second  quadrant modword((m+1)/2)=-sqrt(0.5)+j*sqrt(0.5);
elseif  burst(m)==1  &  burst(m+1)==1  %11  third  quadrant modword((m+1)/2)=-sqrt(0.5)-j*sqrt(0.5);
elseif  burst(m)==0  &  burst(m+1)==1  %01  fourth  quadrant modword((m+1)/2)=sqrt(0.5)-j*sqrt(0.5);
end
end
burst_length=  length(modword); % Upsampling&PulseshapingUpsample=upsample(modword,16); Pulse_shapping=
rcosine(1,16,'sqrt',0.35); Convol=conv(Upsample,Pulse_shapping); Pulse_length=length(Pulse_shapping);
tx=  Convol((Pulse_length-1)/2+1:length(Convol)-(Pulse_length-1)/2);
fd=20;
Sym_rate=2216000;
K=7;
L=1024;
[GI,GQ]=rician_func(fd,L,Sym_rate,K);
%  channel
Rician= (GI+1j*GQ);
signal= Rician.*tx';
snr=30;
A=  (10^(-snr/20));
ric= Rician; % noise
output=A*ric;
ou=output;
rx=signal+ou;
% Matched Filter
Matchfilt=  rcosine(1,16,'sqrt',0.35);
lengthMatchfilt= length(Matchfilt);
Matched_conv=conv(Matchfilt,ou);
lenMatchcon=length(Matched_conv);
%  discard  N-1;
n1=  (lengthMatchfilt-1)/2;
n2=  (lenMatchcon)-n1;
j=1;
for  i=  n1+1  :  n2
Matchout(j)  =Matched_conv(i);
j=j+1;
end



Matchoutlen=length(Matchout);

% DOWNSAMPLING
j=1;
 for  i=  1:16:Matchoutlen
  down(j)=  Matchout(i);
```

```
   j=j+1;
lendownsamp=length(down);

% Block phase estimation
Blocksize=50;     %size of block
step=10;
uw_tx=modword(4:51);
uw_rx=down(4:51);
theta0ref=1/48*sum(angle(uw_rx.*conj(uw_tx)));
k=1;
t_=[];
   ((Blocksize  +  step*(k-1))<=length(down))
   s=sum(down((1+step*(k-1)):(Blocksize+step*(k-1))).^4);
   theta0s ig=1/4*atan2(imag(s),real(s))+[0  pi/4  -pi/4  pi/2  -pi/2  3*pi/4  -3*pi/4  pi];
   a=abs(theta0ref-theta0sig);
   the ta0min(k)=theta0sig(  a==min(a)  );
   t_=[t_  0.5*(Blocksize+step*2*(k-1))];
   the ta0ref=theta0min(k);
k=k+1;
end
thetnewrx=interp1(t_,theta0min,[1:length(down)],'linear');
thetnewrx(1:29)=theta0min(1);
thetnewrx((t_(end)+1:end))=theta0min(end);
Blockphase=down.*exp(-j*thetnewrx);        %  perform  phase  compensation
Blockphaselen=length(Blockphase);

% Hardslicer
n=1;
for  m=1:2:(2*Blockphaselen)-1
ang  =  angle(Blockphase(n));
if   ((ang>  -pi/4)  &  (ang<=0))  |  ((ang>=0)  &  (ang<=pi/4))
burstrx(m)  =  0;  burstrx(m+1)  =  0;
elsif  ((ang<  3*pi/4)  &  (ang>=  pi/4))
burstrx(m)  =  0;  burstrx(m+1)  =  1;
elseif  ((ang>=  3*pi/4)  &  (ang<=  pi))  |  ((ang<=  -3*pi/4)  &  (ang>=  -pi))
burstrx(m)  =  1;  burstrx(m+1)  =  1;
elseif  ((ang<=  -pi/4)  &  (ang>  -3*pi/4  ))
burstrx(m)  =  1;  burstrx(m+1)  =  0;
end
n  =  n+1;
end
BER=0;
for i=  1:1024
if(burstrx(i)~=  burst(i))
   BER=BER+1;
end
end



BER=BER/1108;
ber(count)=BER;
end
biter=sum(ber)/1000;
BITER0(SNR)=biter;
end
```

```
SNR=1:7;
TheBER=[0.09 0.0699 0.0527 0.0387 0.0277 0.0193 0.0135];
figure(1);
semilogy(SNR,(SNR),'r*-'); hold on; semilogy(SNR,TheBER,'bo-');
grid on;
title(' Practical BER Compared with Theoretical ');

legend('Practical BER',...
'Theoretical BER','Location','SouthWest'); hold off
Rician func:
function [GI, GQ] = rician_func(fd,L,Sym_rate,K)
fd=fd;
Sym_rate=Sym_rate; samp=8;
[RI, RQ]=rayleigh(fd,L,Sym_rate,samp);
m_path=10^(-K/10);
RI = RI*sqrt(m_path);
RQ = RQ*sqrt(m_path);
DC =1/sqrt(1+m_path);
RI = RI/sqrt(1+m_path);
RQ=RQ/sqrt(1+m_path); GI=DC+RI;
GQ=RQ;
end
Rayleighfunc:
function [GI, GQ] = rayleigh(fd,Burst_L,Sym_rate,samp)
GI=[];
GQ=[];
N=66;
M=0.5*((N/2)-1);
wm=2*pi*fd; alpha=0; T0=10*rand(1);
for i=1:Burst_L*samp
   T=T0+i*(1/Sym_rate*samp); gI=0;
gQ=0;
for n=1:M
beta(n)=pi*n/M; w(n)=wm*cos(2*pi*n/N); gI=gI+cos(w(n)*t)*2*cos(beta(n)); gQ=gQ+cos(w(n)*t)*2*sin(beta(n));
end
gI =gI+cos(wm*t)*2*cos(alpha)/sqrt(2); gQ=gQ+cos(wm*t)*2*sin(alpha)/sqrt(2);
 gI=gI/sqrt(2*(M+1));
 gQ=gQ/sqrt(2*M);
GI=[GI,gI];
GQ=[GQ,gQ];
end
```

Output:


Practical BER Compared with Theoretical
10^0
10^-1
10^-2
10^-3
1
2
3
4
5
6
7
Practical BER
Theoretical BER

➢ Code for Question 2

Main code:

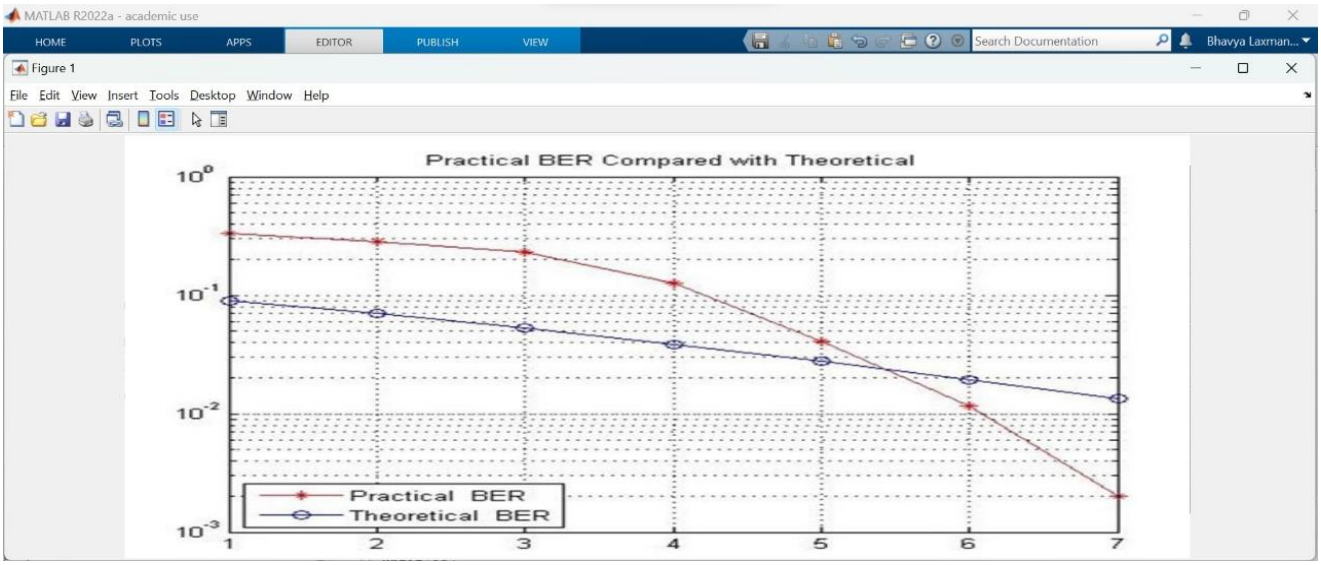
```
clc; clear all;
for SNR=1:7
for count=1:1000
Payload = randi([0,1],1000,1); Gaurd1= randi([0,1],6,1);
Gaurd2=randi([0,1],6,1); UniqueWord=randi([0,1],96,1);
burst=[Gaurd1;UniqueWord;Payload;Gaurd2]; burst_len= length (burst);

% QPSK Modulation
for m=1:2:length(burst)-1
if burst(m)==0 & burst(m+1)==0 %00 first quadrant
modword((m+1)/2)=sqrt(0.5)+j*sqrt(0.5);
elseif burst(m)==1 & burst(m+1)==0 %10 second quadrant
modword((m+1)/2)=0-sqrt(0.5)+j*sqrt(0.5);
elseif burst(m)==1 & burst(m+1)==1 %11 third quadrant
 modword((m+1)/2)=-sqrt(0.5)-j*sqrt(0.5);
elseif burst(m)==0 & burst(m+1)==1 %01 fourth quadrant
modword((m+1)/2)=sqrt(0.5)-j*sqrt(0.5);
end
end

burst_length= length(modword);

% Upsampling&Pulseshaping
Upsample=upsample(modword,16);
Pulse_shapping= rcosine(1,16,'sqrt',0.35);
Convol=conv(Upsample,Pulse_shapping);
Pulse_length=length(Pulse_shapping);
tx= Convol((Pulse_length-1)/2+1:length(Convol)-(Pulse_length-1)/2);
fd=100;
Sym_rate=2216000; K=12;
L=1024;
[GI,GQ]=rician_func(fd,L,Sym_rate,K);% channel
Rician= (GI+1j*GQ); signal= Rician.*tx'; snr=30;
A= (10^(-snr/20));
ric= Rician;
% noise output=A*ric;
ou=output;
 rx=signal+ou;
% Matched Filter
Matchfilt= rcosine(1,16,'sqrt',0.35); lengthMatchfilt= length(Matchfilt); Matched_conv=conv(Matchfilt,ou);
lenMatchcon=length(Matched_conv);
% discard N-1;
n1= (lengthMatchfilt-1)/2; n2= (lenMatchcon)-n1; j=1;
for i= n1+1 : n2
Matchout(j) =Matched_conv(i); j=j+1;
end
Matchoutlen=length(Matchout);
```

```
% DOWNSAMPLING
j=1;
for
i= 1:16:Matchoutlen
down(j)= Matchout(i);
j=j+1;
end
lendownsamp=length(down);
% Block phase estimation
Blocksize=50;     %size of block
step=10;
uw_tx=modword(4:51); uw_rx=down(4:51);
theta0ref=1/48*sum(angle(uw_rx.*conj(uw_tx))); k=1;
t_=[];
while
   ((Blocksize + step*(k-1))<=length(down)) s=sum(down((1+step*(k-1)):(Blocksize+step*(k-1))).^4);
theta0sig=1/4*atan2(imag(s),real(s)) + [0 pi/4 -pi/4 pi/2 -pi/2 3*pi/4 -3*pi/4 pi]; a=abs(theta0ref-theta0sig);
theta0min(k)=theta0sig( a==min(a) ); t_=[t_ 0.5*(Blocksize+step*2*(k-1))]; theta0ref=theta0min(k);
k=k+1;
end
thetnewrx=interp1(t_,theta0min,[1:length(down)],'linear');
thetnewrx(1:29)=theta0min(1);
 thetnewrx((t_(end)+1:end))=theta0min(end);
Blockphase=down.*exp(-j*thetnewrx);        % perform phase compensation
Blockphaselen=length(Blockphase);

% Hardslicer
 n=1;
for
   m=1:2:(2*Blockphaselen)-1
ang = angle(Blockphase(n));
if ((ang> -pi/4) & (ang<=0)) | ((ang>=0) & (ang<=pi/4))
burstrx(m) = 0; burstrx(m+1) = 0;
elseif ((ang< 3*pi/4) & (ang>= pi/4))
burstrx(m) = 0; burstrx(m+1) = 1;
elseif ((ang>= 3*pi/4) & (ang<= pi)) | ((ang<= -3*pi/4) & (ang>= -pi))
burstrx(m) = 1; burstrx(m+1) = 1;
elseif ((ang<= -pi/4) & (ang> -3*pi/4 ))
burstrx(m) = 1; burstrx(m+1) = 0;
end
n = n+1;
BER=0;
for
i= 1:1024
if(burstrx(i)~= burst(i))
   BER=BER+1;
end
end BER=BER/1108;
ber(count)=BER; end
biter=sum(ber)/1000; BITER0(SNR)=biter;
end
```

```
SNR=1:7;
TheBER=[0.06565  0.04664  0.03124  0.01958  0.011396  0.0061246,0.00302];


figure(1); semilogy(SNR,SNR,'r*-'); hold  on; semilogy(SNR,TheBER,'bo-');
grid  on;
title(' Practical  BER  Compared  with  Theoretical  ');
legend('Practical  BER',...
'Theoretical        BER','Location','SouthWest'); hold off
Rician func:
function  [GI,  GQ]  =  rician_func(fd,L,Sym_rate,K)
 fd=fd;
Sym_rate=Sym_rate; samp=8;
[RI, RQ]=rayleigh(fd,L,Sym_rate,samp);
m_path=10^(-K/10);
RI = RI*sqrt(m_path);
RQ  =  RQ*sqrt(m_path);
DC =1/sqrt(1+m_path);
RI  =  RI/sqrt(1+m_path);
RQ=RQ/sqrt(1+m_path);
GI=DC+RI;
GQ=RQ;
end

Rayleigh func:
function  [GI,  GQ]  =  rayleigh(fd,Burst_L,Sym_rate,samp)
GI=[];
GQ=[];
N=66;
M=0.5*((N/2)-1);
wm=2*pi*fd; alpha=0;
T0=10*rand(1);
for
i=1:Burst_L*samp
 t=T0+i*(1/Sym_rate*samp);
gI=0;
gQ=0;
for n=1:M
beta(n)=pi*n/M; w(n)=wm*cos(2*pi*n/N);
gI=gI+cos(w(n)*t)*2*cos(beta(n));
gQ=gQ+cos(w(n)*t)*2*sin(beta(n));
end
gI=gI+cos(wm*t)*2*cos(alpha)/sqrt(2);
gQ=gQ+cos(wm*t)*2*sin(alpha)/sqrt(2);

gI=gI/sqrt(2*(M+1));
gQ=gQ/sqrt(2*M);

GI=[GI,gI];
GQ=[GQ,gQ];

end
```

OUTPUT:

K=12,
Fd=100;



➢ Code for Question 3

Main code:

```
clc; clear;
for  SNR=1:7
for  count=1:1000
Payload = randi([0,1],1000,1); Gaurd1= randi([0,1],6,1);
Gaurd2=randi([0,1],6,1); UniqueWord=randi([0,1],96,1);
burst=[Gaurd1;UniqueWord;Payload;Gaurd2]; burst_len= length (burst);
%  QPSK  Modulation
for  m=1:2:length(burst)-1
if  burst(m)==0  &&  burst(m+1)==0  %00  first  quadrant
 modword((m+1)/2)=sqrt(0.5)+j*sqrt(0.5);
elseif  burst(m)==1  &&  burst(m+1)==0  %10  second  quadrant
 modword((m+1)/2)=-sqrt(0.5)+j*sqrt(0.5);
elseif  burst(m)==1  &&  burst(m+1)==1  %11  third  quadrant
modword((m+1)/2)=-sqrt(0.5)-j*sqrt(0.5);
elseif  burst(m)==0  &&  burst(m+1)==1  %01  fourth  quadrant
 modword((m+1)/2)=sqrt(0.5)-j*sqrt(0.5);
end
end
burst_length=  length(modword);

% Upsampling&Pulseshaping
Upsample=upsample(modword,16);
Pulse_shapping=  rcosine(1,16,'sqrt',0.35);
Convol=conv(Upsample,Pulse_shapping);
 Pulse_length=length(Pulse_shapping);
tx=  Convol((Pulse_length-1)/2+1:length(Convol)-(Pulse_length-1)/2);
fd=0;
Sym_rate=2216000
K=200;
L=1024;
[GI,GQ]=rician_func(fd,L,Sym_rate,K);
%  channel
Rician= (GI+1j*GQ);
signal= Rician.*tx';
snr=30;
A=  (10^(-snr/20));
ric= Rician;
% noise
 output=A*ric;
ou=output;
 rx=signal+ou;
% Matched Filter
Matchfilt=  rcos(1,16,'sqrt',0.35);
lengthMatchfilt= length(Matchfilt);
Matched_conv=conv(Matchfilt,ou);
lenMatchcon=length(Matched_conv);
%  discard  N-1;
n1=  (lengthMatchfilt-1)/2;
n2=  (lenMatchcon)-n1;
  j=1;
```

```
for
i= n1+1 : n2 Matchout(j) = Matched_conv(i);
 j=j+1;
end


Matchoutlen=length(Matchout);
% DOWNSAMPLING
 j=1;
for
i= 1:16:Matchoutlen
   down(j)= Matchout(i);
   j=j+1;
end
lendownsamp=length(down); % Block phase estimation
Blocksize=50;     %size of block
 step=10;
uw_tx=modword(4:51);
uw_rx=down(4:51);
theta0ref=1/48*sum(angle(uw_rx.*conj(uw_tx))); k=1;
t_=[];
while
   ((Blocksize + step*(k-1))<=length(down)) s=sum(down((1+step*(k-1)):(Blocksize+step*(k-1))).^4);
theta0sig=1/4*atan2(imag(s),real(s)) + [0 pi/4 -pi/4 pi/2 -pi/2 3*pi/4 -3*pi/4 pi]; a=abs(theta0ref-theta0sig);
theta0min(k)=theta0sig( a==min(a) ); t_=[t_ 0.5*(Blocksize+step*2*(k-1))]; theta0ref=theta0min(k);
k=k+1;
end
thetnewrx=interp1(t_,theta0min,[1:length(down)],'linear');
thetnewrx(1:29)=theta0min(1); thetnewrx((t_(end)+1:end))=theta0min(end);
Blockphase=down.*exp(-j*thetnewrx);        % perform phase compensation
 Blockphaselen=length(Blockphase);
% Hardslicer
n=1;
for m=1:2:(2*Blockphaselen)-1
ang = angle(Blockphase(n));
if ((ang> -pi/4) && (ang<=0))
   ((ang>=0) & (ang<=pi/4))
burstrx(m) = 0; burstrx(m+1) = 0;
elseif ((ang< 3*pi/4) & (ang>= pi/4))
burstrx(m) = 0; burstrx(m+1) = 1;
elseif ((ang>= 3*pi/4) & (ang<= pi)) | ((ang<= -3*pi/4) & (ang>= -pi))
burstrx(m) = 1; burstrx(m+1) = 1;
elseif ((ang<= -pi/4) & (ang> -3*pi/4 ))
burstrx(m) = 1; burstrx(m+1) = 0;
end
n = n+1;
end
BER=0;
for
i= 1:1024
if(burstrx(i)~= burst(i))
   BER=BER+1;
end
end
```

```
BER=BER/1108;
ber(count)=BER;
end
biter=sum(ber)/1000; BITER0(SNR)=biter;
end
SNR=1:7;
TheBER=[0.06565 0.04664 0.03124 0.01958 0.011396 0.0061246,0.00302];
figure(1); semilogy(SNR,SNR,'r*-'); hold on; semilogy(SNR,TheBER,'bo-');
grid on;
title(' Practical BER Compared with Theoretical ');
legend('Practical BER',...
'Theoretical BER','Location','SouthWest');
 hold off
```

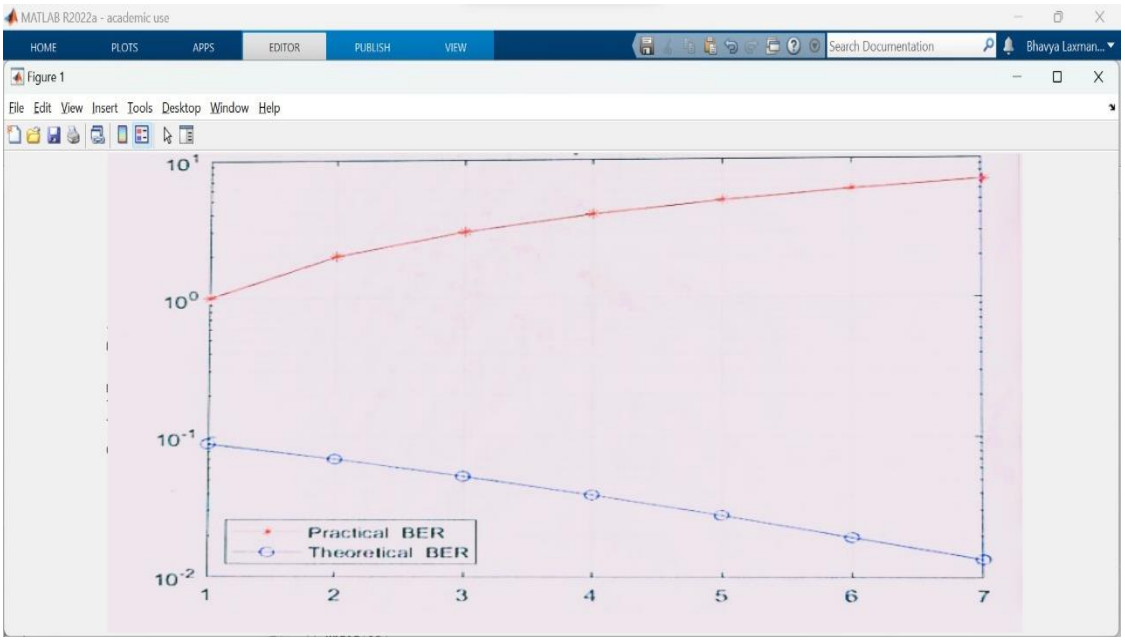
Rician func:

```
function [GI, GQ] = rician_func(fd,L,Sym_rate,K) fd=fd;
Sym_rate=Sym_rate; samp=8;
[RI, RQ]=rayleigh(fd,L,Sym_rate,samp);
m_path=10^(-K/10);
RI = RI*sqrt(m_path);
RQ = RQ*sqrt(m_path); DC =1/sqrt(1+m_path);
RI = RI/sqrt(1+m_path);
RQ=RQ/sqrt(1+m_path); GI=DC+RI;
GQ=RQ;
End
```

Rayleigh func:

```
function
[GI, GQ] = rayleigh(fd,Burst_L,Sym_rate,samp)
GI=[];
GQ=[];
N=66;
M=0.5*((N/2)-1);
wm=2*pi*fd; alpha=0; T0=10*rand(1);
for
i=1:Burst_L*samp t=T0+i*(1/Sym_rate*samp);
gI=0; gQ=0;
for n=1:M
beta(n)=pi*n/M; w(n)=wm*cos(2*pi*n/N);
gI=gI+cos(w(n)*t)*2*cos(beta(n));
gQ=gQ+cos(w(n)*t)*2*sin(beta(n));
end
gI=gI+cos(wm*t)*2*cos(alpha)/sqrt(2);
gQ=gQ+cos(wm*t)*2*sin(alpha)/sqrt(2);
gI=gI/sqrt(2*(M+1)); gQ=gQ/sqrt(2*M);
GI=[GI,gI];
GQ=[GQ,gQ];

end
```

Output:


MATLAB R2022a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Bhavya Laxman...
Figure 1
File Edit View Insert Tools Desktop Window Help
10^{1}
10^{0}
10^{-1}
10^{-2}
1
2
3
4
5
6
7
Practical BER
Theoretical BER

- **Code for Question 5**
- **RICIAN WITH MRC**

Main code:

```
clc;
clear all;
close  all;
 SNR =20;
Sym_rate=2216000;
Length=500;
message=randi([0,  1],Length,1);
 trellis = poly2trellis(7,[133 171]);
information_coded=convenc(message,trellis);

Guard=randi([0, 1],12,1);
U_W =randi([0 1],96,1);


burst_data=[Guard;U_W;information_coded];
for  m=1:2:length(burst_data)-1
ifburst_data(m)==0  &&burst_data(m+1)==0            %first  quadrant
modulation((m+1)/2)=sqrt(0.5)+1i*sqrt(0.5);
elseifburst_data(m)==1  &&burst_data(m+1)==0       %second  quadrant
 modulation((m+1)/2)=-sqrt(0.5)+1i*sqrt(0.5);
elseifburst_data(m)==1  &&burst_data(m+1)==1       %third  quadrant
 modulation((m+1)/2)=-sqrt(0.5)-1i*sqrt(0.5);
elseif
burst_data(m)==0  &&burst_data(m+1)==1 %fourth  quadrant
 modulation((m+1)/2)=sqrt(0.5)-1i*sqrt(0.5);
end
end
%Upsampling
Upsampling=upsample(modulation,16); pulseshaping=rcosdesign(1,16,'sqrt',0.35); Convolution=conv(Upsampling,pulseshaping);
Leng_pulse=length(pulseshaping);
pu_up=Convolution((Leng_pulse-1)/2+1:length(Convolution)-(Leng_pulse-1)/2);
%Richannel 1
fd =20;
K=7;
[GI,GQ]=rician_func(fd,1108,Sym_rate,K);
richannel= GI+1i*GQ;
signal=richannel.*pu_up;
A=10.^-(SNR/10);
n1  =richannel  ; noise=A*n; rx=noise+signal;
%Matched  filter_Richannel  1
 a=rcosine(1,16,'sqrt',0.35);
 Leng_p=length(a);
Convl=conv(a,rx);
down out=Convl((Leng_p-1)/2+1:length(Convl)-(Leng_p-1)/2);
%downsampling_Richannel  1
 down1=downsample(downout,16);
```

```
%Richannel  2
fd=100;
K=12;
[GI,GQ]=rician_func(fd,1108,Sym_rate,K);
richannel= GI+1i*GQ;
signal=richannel.*pu_up;
A=10.^-(SNR/10);
n2 =richannel ;
noise=A*n; rx=noise+signal;
 %Matched  filter_Richannel  2
a=rcosine(1,16,'sqrt',0.35);
Leng_p=length(a);
Convl=conv(a,rx);
down out=Convl((Leng_p-1)/2+1:length(Convl)-(Leng_p-1)/2);
%downsampling_Richannel  2
down2=downsample(downout,16);

%Richannel 3
fd=20;
K=7;
[GI,GQ]=rician_func(fd,1108,Sym_rate,K); richannel= GI+1i*GQ; signal=richannel.*pu_up;
A=10.^-(SNR/10);
n3 =richannel ; noise=A*n; rx=noise+signal;
 %Matched  filter_Richannel  3
 a=rcosine(1,16,'sqrt',0.35);
 Leng_p=length(a);
Convl=conv(a,rx);
down out =Convl((Leng_p-1)/2+1:length(Convl)-(Leng_p-1)/2);
%downsampling_Richannel  3 down3=downsample(downout,16);
%Richannel 4
fd=100;
K=12;
[GI,GQ]=rician_func(fd,1108,Sym_rate,K); richannel= GI+1i*GQ; signal=richannel.*pu_up;
A=10.^-(SNR/10);
n4 =richannel ; noise=A*n; rx=noise+signal;
%Matched  filter_Richannel  4
a=rcosine(1,16,'sqrt',0.35);
Leng_p=length(a);
Convl=conv(a,rx);
downout=Convl((Leng_p-1)/2+1:length(Convl)-(Leng_p-1)/2);
%downsampling_Richannel  4
 down4=downsample(downout,16);
rxburst_1=n1(1:16:end);
rxburst_2=n2(1:16:end);
 rxburst_3=n3(1:16:end);
 rxburst_4=n4(1:16:end);
mu=zeros(1,4); s=[  1  j  -1  -j  ];
for  a  =  1:length(Blockphase1)
rsum(a)=  conj(rxburst_1)*Blockphase1(a)  +  conj(rxburst_2)*Blockphase2(a)  +  ...conj(rxburst_3)*Blockphase3(a)  +
conj(rxburst_4)*Blockhpahse4(a);
mu(1)=rsum(a)*conj(s(1));
mu(2)=rsum(a)*conj(s(2));
mu(3)=rsum(a)*conj(s(3));
mu(4)=rsum(a)*conj(s(4));
[Y,I]=max(real(mu));
blockphaseout(a)=s(I);
end
```

```
Blockphase=hardslice(blockphaseout);
[number,ratio(burstcount)] = biterr(Blockphase(103:1102),Burst1(103:1102));
end

Ratio(SNR)=sum(ratio)/burstcount; disp(Ratio(SNR));
end;
SNR=1:7;
TheBER=[0.09 0.0699 0.0527 0.0387 0.0277 0.0193 0.0135];
figure(1); semilogy(SNR,Ratio(SNR),'r*-');
hold on; semilogy(SNR,TheBER,'bo-'); grid on;
title(' Practical BER Compared with Theoretical ');
legend('Practical BER',...
'Theoretical BER','Location','SouthWest');
hold off;
```

ALAMOUTI CODES

```
L=500
%generate the raw information bits
msg = randi([0,1],L,1);
msg=msg'
SNR=20;
%generate the trellis for the encoder
trel = poly2trellis(7,[133 171]);
%convolution encoder
Info_coded=convenc(msg,trel);
%for Guard and UW
GUW=rand(1,102)
GUW=round(GUW)
% End Guard
EG= rand(1,6)
EG=round(EG)
%Build Burst
Burst = [ GUW info_coded EG]
```

```
%Modulation of Burst
for
  i = 1:1108;
     if (mod(i.2)==0)
                    if(Burst(1,i)==0 && Burst(1,i-1)==0)
                      Aftermodulation(1,i/2)=1
                       elseif(Burst(1,i)==1 &&  Burst(1,i-1)==0)
                              Aftermodulation(1,i/2)=j
                         elseif(Burst(1,i)==1 && Burst(1,i-1)==1)


                 Aftermodulation(1,i/2)=-1
                  else (Burst(1,i)==0 && Burst(1,i-1)==1)
                 Aftermodulation(1,i/2)=-j
                 end
               end
              end
  %Alamaunti code

     for i=1:554
            if(mod(i,2)==0)
                 Alamaunti(1,i-1)=-conj(Aftermodulation(1,i))
                  Alamaunti(1,i)=conj(Aftermodulation(1,i-1))
       end

       end

       %Provided upsampling
       SampledupT1=upsample(Aftermodulation,16)
       T1Filtered=rcosine(1,16,'sqrt',0.35)
       w1=conv(SampledupT1,T1Filtered)
       o1=length(T1Filtered)
      tx1=w1((o1-1)/2+1:length(w1)-(o1-1)/2)
       fd=0
       L=1108
       Sym_rate=2216000
       K=7
      [GI,GQ]=rician_func(fd,L,Sym_rate,K)
      F1= (GI+1j*GQ)
      Ff1=F1
      signal1= Ff1.*tx1
      SampledupT2=upsample(Alamaunti,16)
       T2Filtered=rcosine(1,16,'sqrt',0.35)
       w2=conv(SampledupT1,T2Filtered)
       o2=length(T2Filtered)
        tx2=w2((o2-1)/2+1:length(w2)-(o2-1)/2)
        F2= (GI+1j*GQ)
        Ff2=F2
       signal2=Ff2.*tx2 signal=signal1+signal2
       %Adding noise
```

```
%Adding noise into Channel
   n =signal ; noise=A*n;
    rx=noise+signal;
    pulseshaping1=rcosine(1,16,'sqrt',0.35);
    A=downsample(rx,16);
```

- **Code for Question 6**
- **Rician channel with Convolution**

Main Code:

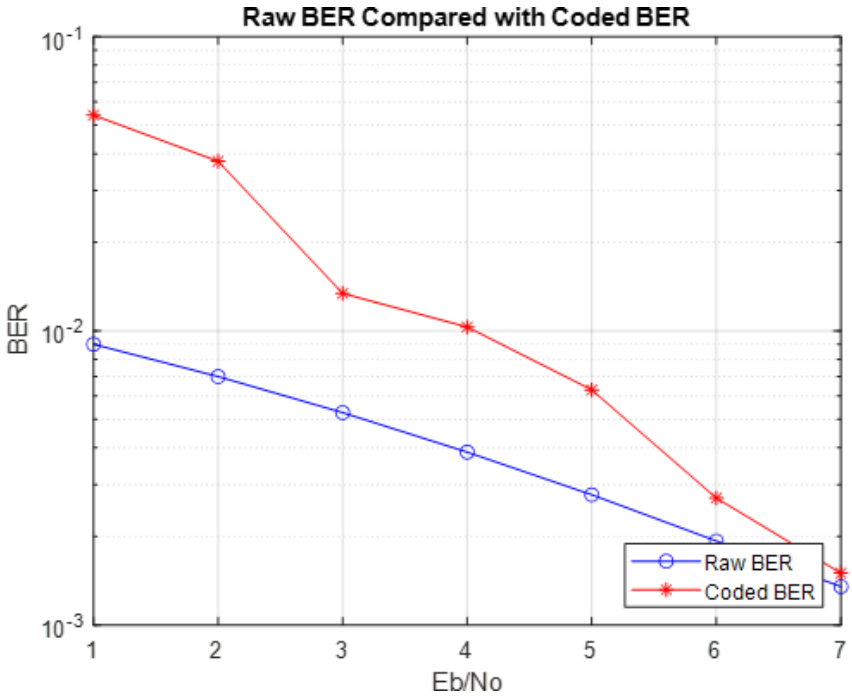
```
clc; clear all;
close all;
Sym_rate=2216000;
for SNR=1:7
for cou=1:1:100

    %Inf Bits
Length=500;
message=randi([0, 1],Length,1); trellis =
poly2trellis(7,[133 171]);
information_coded=convenc(message,trellis);

%Guard
Guard=randi([0, 1],12,1);

%Unique word
U_W =randi([0 1],96,1);

%Burst builder burst_data=[Guard;U_W;information_coded];

%QPSK modulation

for a=1:2:length(burst_data)-1
    if burst_data(a)==0 & burst_data(a+1)==0                    %first quadrant
        modulation((a+1)/2)=sqrt(0.5)+1i*sqrt(0.5);
    elseif burst_data(a)==1 & burst_data(a+1)==0                %second quadrant
        modulation((a+1)/2)=-sqrt(0.5)+1i*sqrt(0.5);
    elseif burst_data(a)==1 & burst_data(a+1)==1                %third quadrant
        modulation((a+1)/2)=-sqrt(0.5)-1i*sqrt(0.5);
    elseif burst_data(a)==0 & burst_data(a+1)==1                %fourth quadrant
        modulation((a+1)/2)=sqrt(0.5)-1i*sqrt(0.5);
    end
end

%upsampling
Upsampling=upsample(modulation,16);
```

```
pulseshaping=rcosine(1,16,'sqrt',0.35);
Convolution=conv(Upsampling,pulseshaping);
Leng_pulse=length(pulseshaping);
pu_up=Convolution((Leng_pulse-1)/2+1:length(Convolution)-(Leng_pulse-1)/2);

%richannel  fd=100;
K=12;
[GI,GQ]=rician_func(fd,1108,Sym_rate,K);richannel=
GI+1i*GQ; signal=richannel.*pu_up;
A=10.^-(SNR/10);
n =richannel ;
noise=A*n;
rx=noise+signal;

%Matched filter
a=rcosine(1,16,'sqrt',0.35);
Leng_p=length(a); Convl=conv(a,rx);
downout=Convl((Leng_p-1)/2+1:length(Convl)-(Leng_p-1)/2);

%downsampling
down=downsample(downout,16);

%Block phase estimation
Blocklength=48; step=10;
tx=modulation(1:40);
rx=down(1:40);

t_ref=1/40*sum(angle(rx.*conj(tx)));a=1;
t_=[];
while((Blocklength + step*(a-1))<=length(downout)) s=sum(downout((1+step*(a-
      1)):(Blocklength+step*(a-1))).^4);
      t_sig=1/4*atan2(imag(s),real(s)) + [0 pi/4 -pi/4 pi/2 -pi/2 3*pi/4 -3*pi/4 pi];t_min(a)=t_sig( a==min(a) );
      t_=[t_ 0.5*(Blocklength+step*2*(a-1))];t_ref=t_min(a);
      a=a+1;
end

t_rx=interp1(t_,t_min,[1:length(downout)],'linear');
t_rx(1:29)=t_min(1); t_rx((t_(end)+1:end))=t_min(end);

phase=downout.*exp(-j*t_rx);n=1;
for
a=1:2:(2*length(phase)-1)hardslice
=angle(phase(n));
          if ((hardslice> -pi/4) & (hardslice<=0)) | ((hardslice>=0) & (hardslice<=pi/4))burst(a) = 0; burst(a+1) = 0;
          elseif ((hardslice< 3*pi/4) & (hardslice>= pi/4))burst(a) = 0;
                burst(a+1)= 1;
           elseif ((hardslice>= 3*pi/4) & (hardslice<= pi)) | ((hardslice<= -3*pi/4) &(hardslice>= -pi))
               burst(a) = 1; burst(a+1) = 1; burst(a+1) = 0;
            elseif ((hardslice<= -pi/4) & (hardslice> -3*pi/4 ))burst(a) = 1;
        end
          n = n+1;
      end
```

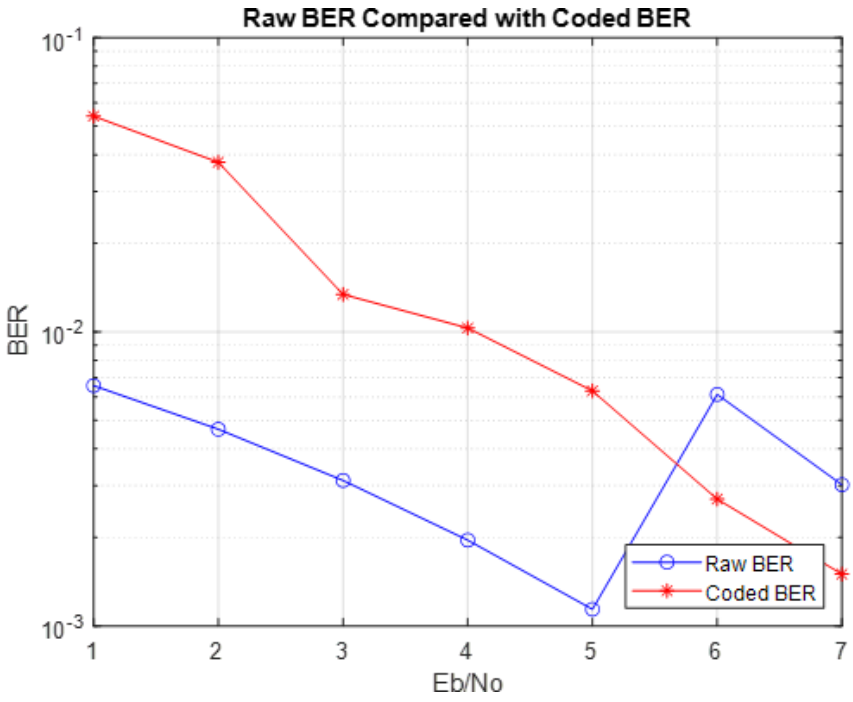
```
%Viterbi Decoder

  trellis=poly2trellis(7,[133 171]);                    % generate trellis structure at the
  receiverbur=burst(103:1102)
  Demod_viterbi=vitdec(bur,trellis,Length,'term','hard');
reciever_burst= length(Demod_viterbi)

[num,rat(cou)] =
      biterr(Demod_viterbi,message');end;

%ber calculation

Ratio(SNR)=sum(r
at)/cou;
disp(Ratio(SNR));
 end;
SNR:1:7;
BER=[0.09 0.0699 0.0527 0.0387 0.0277 0.0193 0.0135];
figure;
semilogy(SNR,Ratio
(SNR),'bo-');hold on;
semilogy(SNR,BER
(SNR),'r*-'); grid on;
xlabel('Eb/No');
ylabel('BER');
legend('Coded   BER',...
              'Raw
BER','Location','SouthEast'); title('
Raw BER Compared with Coded
BER ');hold off
```

Output:

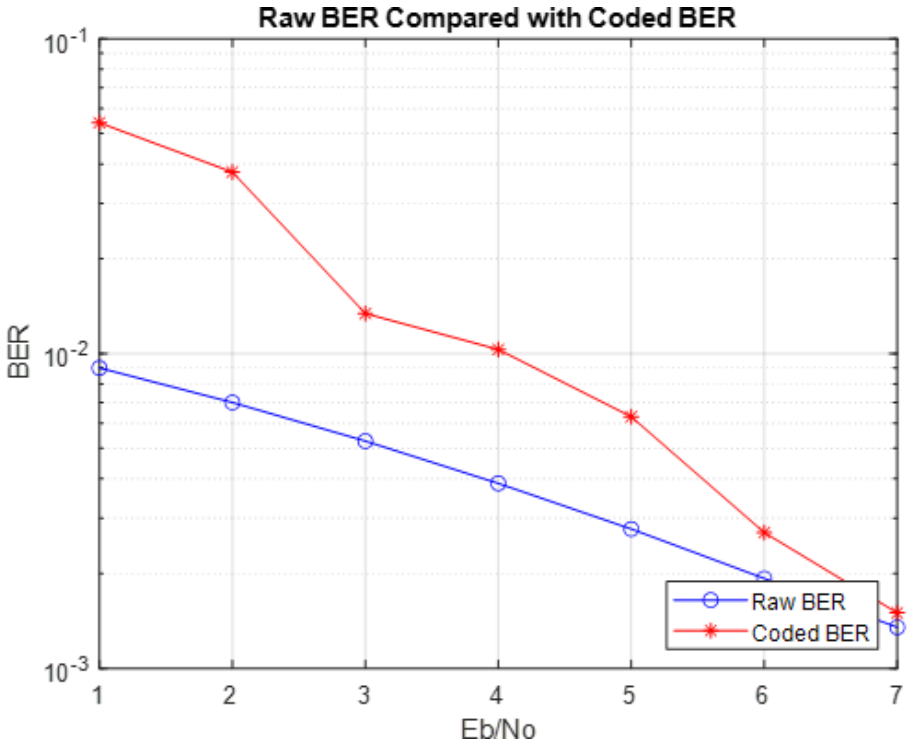
K=7, fd=20, SNR=1-7



K=20, fd=100, SNR=1-7



K=200, fd=0 ,SNR=1-7



➢ Code for Question 7

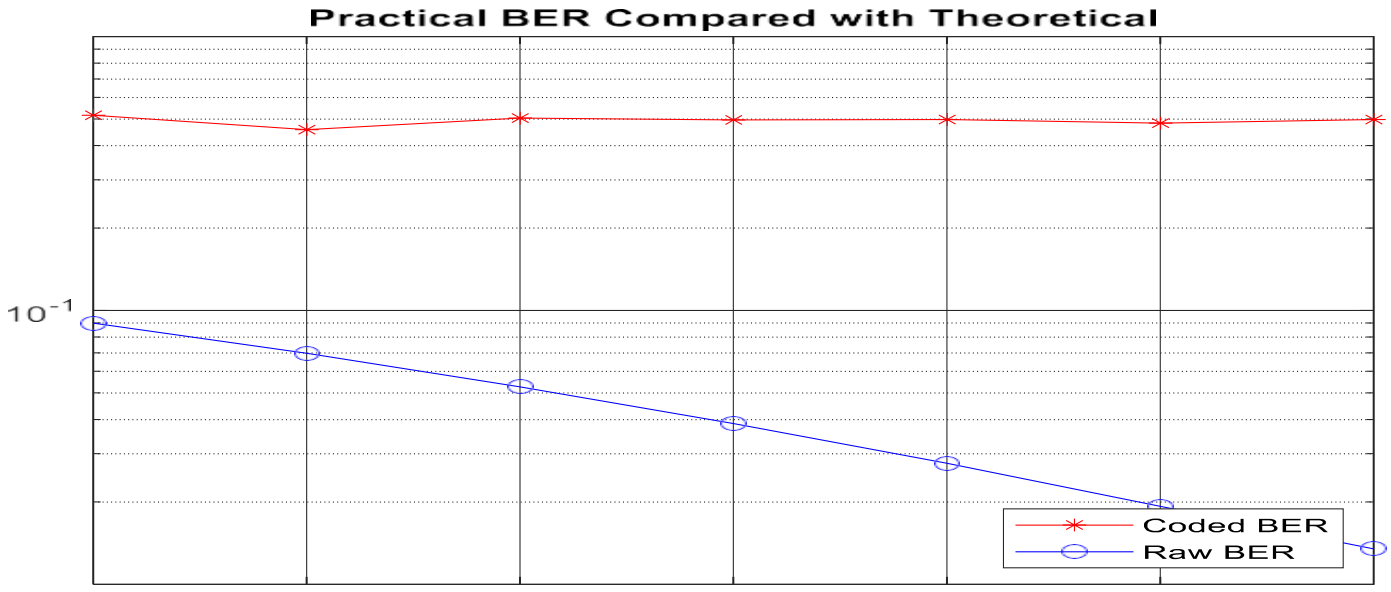
```
clc; clear  all;
for
   SNR=1:7
%for burstcount=1:5
 L=666
%generate  the  raw  information  bits
 msg = randi([0,1],L,1)
msg=msg'
%for Guard and UW
GUW=rand(1,102)
GUW=round(GUW)
% End Guard
 EG=rand(1,6)
 EG=round(EG)
%generate the trellis for the encoder
trel = poly2trellis(7,[133 171]);
%convolution   encoder
info_coded = convenc(msg,trel);
%Coding for Puncturing
 k=1
for
i=1:1332
if(i<4)
puncturedbufferoutput(1,i)=info_coded(1,i) end
if(i==4)
puncturedbufferoutput(1,3)=info_coded(1,4)
end
if(i>4)
if(mod(i,3  +4*k)==0)
k=k+1
end
puncturedbufferoutput(1,i-k)=info_coded(1,i)
end
end
puncturedbufferoutput(1,1000)=1
%Coding  for  Interleaving l=1;
for
   j=1:5 k=j
while
   (k<=1000)

Interleveroutput(1,l)=puncturedbufferoutput(1,k) k=k+5
l=l+1

end
end


%Build Burst
Burst=[GUW  Interleveroutput  EG]
```

```
%Modulation  of  Burst
for  i=1:1108
if
   (mod(i,2)==0)
if
   (Burst(1,i)==0  &&  Burst(1,i-1)==0)

Aftermodulation(1,i/2)=1
elseif
   (Burst(1,i)==1  &&  Burst(1,i-1)==0) Aftermodulation(1,i/2)=j
elseif
   (Burst(1,i)==1  &&  Burst(1,i-1)==1)
Aftermodulation(1,i/2)=-1

else
   (Burst(1,i)==0  &&  Burst(1,i-1)==1) Aftermodulation(1,i/2)=-j
end
end
end

%Provided upsampling
 z=upsample(Aftermodulation,16)
%Match filtering
r=rcosine(1,16,'sqrt',0.35)
w=conv(z,r)
o=length(r)
tx=w((o-1)/2+1:length(w)-(o-1)/2)
%x=w(25:1024*8+24)
fd=100 L=1108
Sym_rate=2216000 K=12
[GI,GQ]=rician_func(fd,L,Sym_rate,K)
F= (GI+1j*GQ)
Ff=F
signal= Ff.*tx
%Adding noise into Channel
A=10.^-(SNR/10);
n  =signal  ;
noise=A*n; rx=noise+signal;
pulseshaping1=rcosine(1,16,'sqrt',0.35); A=downsample(rx,16);

% Block phase estimation
Window_length=50; wstep=10;
uw_tx=Aftermodulation(4:43); uw_rx=A(4:43);
theta_ref=1/40*sum(angle(uw_rx.*conj(uw_tx)));



k=1;

t_=[];

while
   ((Window_length  +wstep*(k-1))<=length(A)) s=sum(A((1+wstep*(k-1)):(Window_length+wstep*(k-1))).^4);
theta_sig=1/4*atan2(imag(s),real(s))  +  [0  pi/4  -pi/4  pi/2  -pi/2  3*pi/4  -3*pi/4  pi];
```

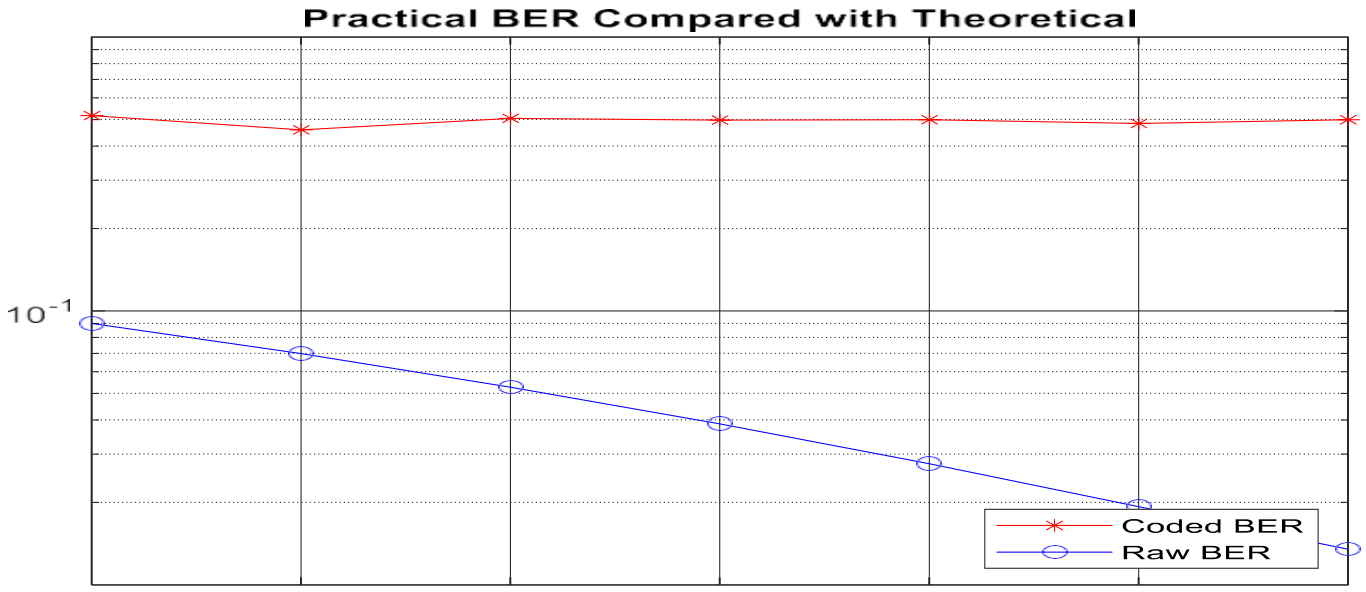
```
%  8 possible angles

a=abs(theta_ref-theta_sig);

theta_min(k)=theta_sig(  a==min(a)  );
%Remove  the  ambiguity  in  angle  calculation  by comparing with the UW.

t_=[t_  0.5*(Window_length+wstep*2*(k-1))]; theta_ref=theta_min(k);
k=k+1;

end
theta_rx=interp1(t_,theta_min,[1:length(A)],'linear');  %  Perform  Linear  Interpolation  to  get the entire range of
angles
theta_rx(1:29)=theta_min(1); theta_rx((t_(end)+1:end))=theta_min(end);
blockphaseout=A.*exp(-j*theta_rx);              %  perform  phase  compensation  (rotate  counter-clock wise)
% Hard slicing n=1;
for
   m=1:2:(2*length(blockphaseout)-1)
angl  =  angle(blockphaseout(n));
if  ((angl>  -pi/4)  &  (angl<=0))  |  ((angl>=0)  &  (angl<=pi/4))
burst_rx(m) = 0; burst_rx(m+1) = 0;
elseif  ((angl<  3*pi/4)  &  (angl>=  pi/4))
burst_rx(m) = 0; burst_rx(m+1) = 1;
elseif  ((angl>=  3*pi/4)  &  (angl<=  pi))  |  ((angl<=  -3*pi/4)  &  (angl>=  -pi))
burst_rx(m) = 1; burst_rx(m+1) = 1;
elseif  ((angl<=  -pi/4)  &  (angl>  -3*pi/4  ))
burst_rx(m) = 1; burst_rx(m+1) = 0;
end
n = n+1; end

% Hard Slicing ( Demodulator)
trellis=poly2trellis(7,[133  171]);  %  generate  trellis  structure  at  the  receiver tblen=3;
burst_rx=burst_rx(103:1102)
%  Demod_viterbi=vitdec(burstDemod(2*(3+40)+1:end-6),trellis,tblen,'trunc','hard');  %  viterbi decoder on Payload
% Coding for deinterleaving


o=1 love=1 b=1
for  love=1:5
while(o<=200)
burstafterinterleavi(love,o)=burst_rx(1,b) o=o+1
b=b+1 end
o=1 end
burstafterinterleavi=burstafterinterleavi'
e=1 c=1 q=1
for  e=1:200
while(q<=5)
burstafterinterleave(1,c)=burstafterinterleavi(e,q) q=q+1;
c=c+1; end q=1
end
```

```
%Code for depuncturing w=1
t=1 i=1
for  w=1:1000
if(w<3)
depuncturedbufferoutpu(1,i)=burstafterinterleave(1,i) i=i+1
end
if(w==3)
depuncturedbufferoutpu(1,w)=1 depuncturedbufferoutpu(1,w+t)=burstafterinterleave(1,i) i=i+1
end
if(w>3  &&  mod(i,3)==0)
depuncturedbufferoutpu(1,w+t)=1 t=t+1
depuncturedbufferoutpu(1,w+t)=burstafterinterleave(1,i) i=i+1
elseif(w>3  &&  mod(i,3)  ~=  0)
depuncturedbufferoutpu(1,w+t)=burstafterinterleave(1,i) i=i+1
else
end
end
depuncturedbufferoutput=depuncturedbufferoutpu(1:1332);%Entering into Depuncturing output into vitbit decoder
Demod_viterbi=vitdec(depuncturedbufferoutput,trellis,tblen,'trunc','hard');
trellis=poly2trellis(7,[133  171]);  % generate  trellis  structure  at  the  receiver tblen=3;
Demod_viterbi=vitdec(depuncturedbufferoutput,trellis,tblen,'trunc','hard');  %  1500  -->  750
% BER CALCULATION
burstcount=1
[number,ratio(burstcount)] = biterr(Demod_viterbi(1:666),msg(1:666));
Ratio(SNR)=sum(ratio)/burstcount; disp(Ratio(SNR));
end
SNR=1:7;
TheBER=[0.09 0.0699 0.0527 0.0387 0.0277 0.0193 0.0135];
figure; semilogy(SNR,Ratio(SNR),'r*-'); hold  on; semilogy(SNR,TheBER(SNR),'bo-'); grid  on;
xlabel('Eb/No  in  dB'); ylabel('BER');
title('  Practical  BER  Compared  with  Theoretical  ');
legend('Coded  BER',...
'Raw  BER','Location','SouthEast');
```

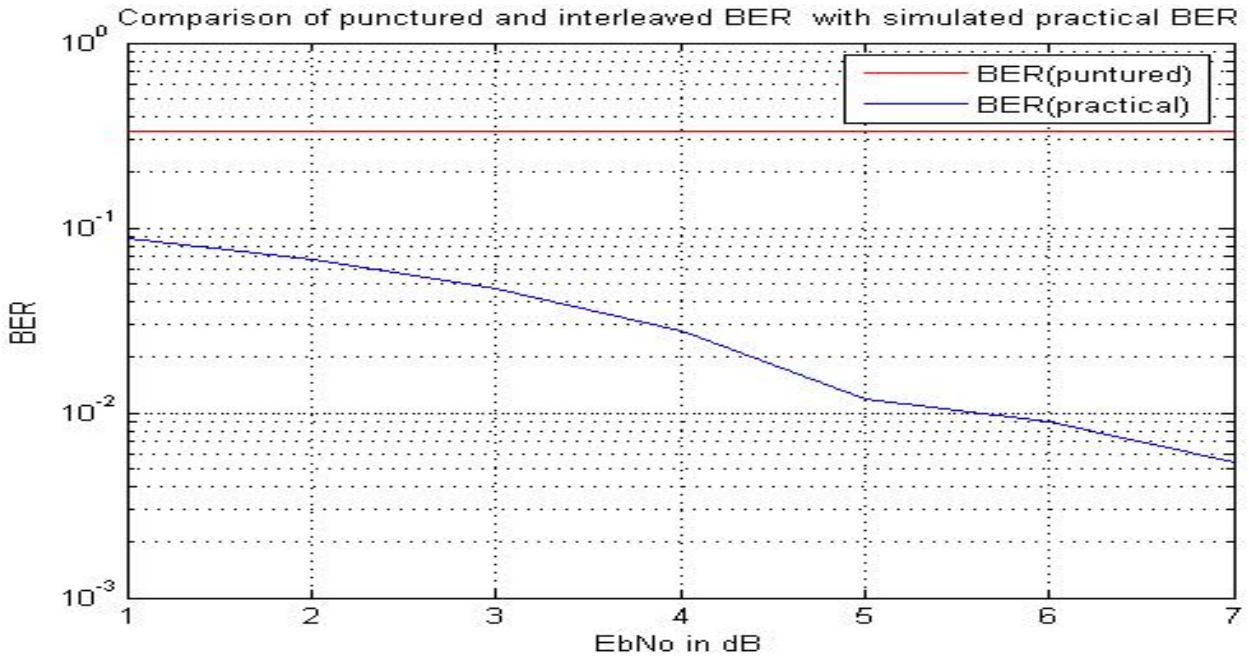
OUTPUT
Fd=20 k=7



Fd=0hz K=200db



K=12db, fd=100hz


Comparison of punctured and interleaved BER with simulated practical BER
BER(puntured)
BER(practical)
BER
EbNo in dB
10^0
10^-1
10^-2
10^-3
1
2
3
4
5
6
7

➢ Question 8

1. Simulation of the rician channel was performed. EB/N0 increases, BER decreases as for a given Rician factor and Doppler shift. Theoretical values were observed to be identical as practical values. In comparison with the first two instance, third instance performed well as the doppler shift was 0. When the performance of case 2 was compared with case 1, it was found that case 2 performed better than case 1.

2. The BER is significantly reduced stating that Convolutional coding improves the result, which is consistent with the effect of convolutional coding. on BER according to theoretical estimation. Calculations.

3. It was found that the BER decreased compared to the Basic Rician channel when we combined punctuation and interleaving with convolutional coding and decoding. This is mainly due to piercing. Piercing is only useful if a harsh environment causes an explosion.