Cairo university Faculty
of Engineering
Electronics and Communications Department

Communications Project for 4th year students

Submitted by

Name	Section	BN
هيثم عصام عزت درويش	4	46

Table of content

Table of figures

Single Carrier System

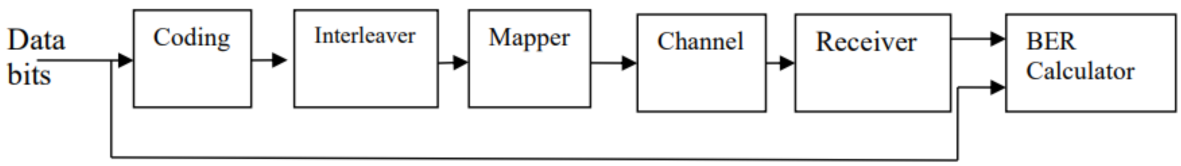
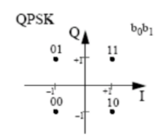


Figure 1 Single carrier communication system.

QPSK



Code

```
clc;clear;%close all;

NumberofBits = 36000;

Data=randi([0,1],[NumberofBits, 1]);

No = 10.^[-10:0.1:3];

%======================================================================%

% For Quadri-Phase Shift Keying (QPSK)

num_rep=3;

Data_rep=repelem(Data,num_rep);

A=1;%/sqrt(num_rep);

% Interleaver


for i=1:16:length(Data_rep)

    temp=reshape(Data_rep(i:i+15), 4, 4)';

    Data_rep(i:i+15)=temp(:);

end

i=1;

SI=zeros(length(Data_rep)/2, 1);

SQ=zeros(length(Data_rep)/2, 1);

da=zeros(length(Data_rep)/2, 2);

Noise=sqrt(No/2).*randn(length(Data_rep)/2, length(No))+sqrt(No/2).*randn(length(Data
_rep)/2, length(No))*j;

v1=randn(length(Data_rep)/2, 1);%+randn(length(Data_rep)/2, 1)*j;

v2=randn(length(Data_rep)/2, 1);%+randn(length(Data_rep)/2, 1)*j;
```

```
R=sqrt(v1.^2+v2.^2)/sqrt(2);
%R=normalize(sqrt(v1.^2+v2.^2)/sqrt(2));
% Mapper
for d=1:2:length(Data_rep),
    da(i, :)=Data_rep(d:d+1);

    if da(i, 1)==0,
        SI(i)=-1;
    else
        SI(i)=1;
    end

    if da(i, 2)==0,
        SQ(i)=-1;
    else
        SQ(i)=1;
    end
    i=i+1;
end
S_QPSK=A*(SI+SQ*j);
X_QPSK=R.*(S_QPSK)+Noise;

QPSK_RX=zeros(length(Data_rep), length(No));
QPSK_RX(1:2:length(Data_rep), :)=real(X_QPSK.*(conj(R)./norm(R)))>0 ;
QPSK_RX(2:2:length(Data_rep), :)=imag(X_QPSK.*(conj(R)./norm(R)))>0;

for no=1:length(No)
    for i=1:16:size(QPSK_RX,1)
        temp=reshape(QPSK_RX(i:i+15, no), 4, 4)';
        QPSK_RX(i:i+15, no)=temp(:);
    end
end
```

```
if num_rep>1
    X=zeros(size(QPSK_RX, 1)/num_rep,size(QPSK_RX, 2), num_rep);
    s=size(QPSK_RX);
    for k=1:s(2)
      jj=1;
      for i=1:num_rep:s(1)
        X(jj, k, :)=QPSK_RX(i:i+num_rep-1, k);
        jj=jj+1;
      end
    end
    QPSK_RX=mode(X,3);
end

QPSK_BER=sum(QPSK_RX~=Data)/NumberofBits;

% Get average symbola dand bit energy
avg_symbol_energy=4*(1+1)/4;
Eb_QPSK=avg_symbol_energy/2;
%theoritical_error_QPSK=1/2*erfc(sqrt(Eb_QPSK./No));
hold on
semilogy(10*log10(Eb_QPSK./No),QPSK_BER, 'r');
%semilogy(10*log10(Eb_QPSK./No),theoritical_error_QPSK, 'r');
hold off
title('BER vs EB/No of QPSK');
xlabel('Eb/No in dB'); ylabel('BER');

legend('BER QPSK fading with repeation (same information bit)', 'BER of QPSK fading w
ith no repeation', 'BER QPSK fading with repeation (same transmitted bit)');

yticks(10.^[-6:1:0])
xticks([0:2:24])
ylim([10^-6 10^0])
xlim([0 18])
set(gca, 'YScale', 'log')
```

Graphs

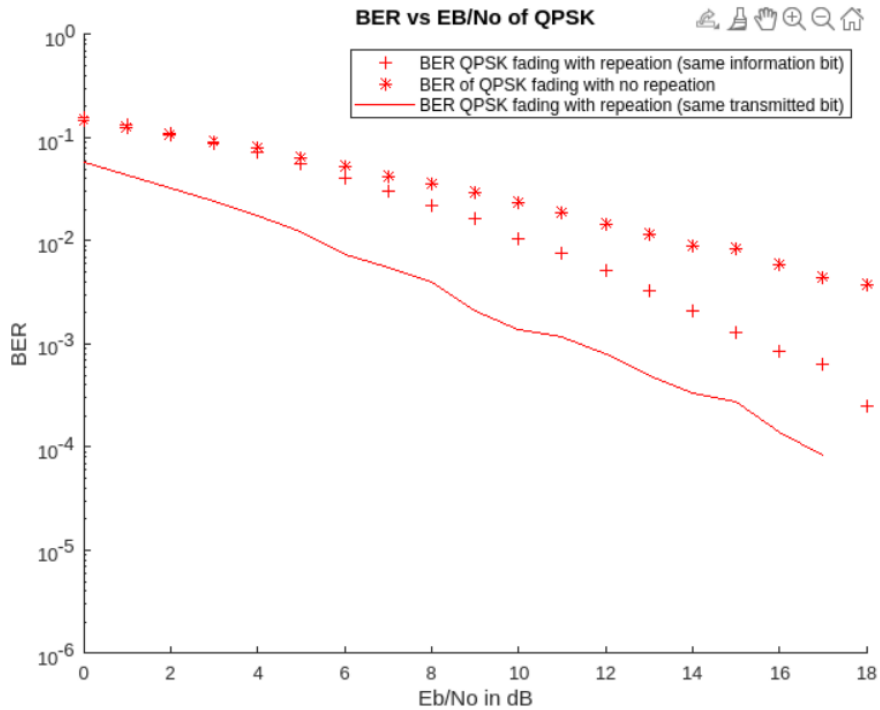


Figure 2 The relation between BER and Eb/No QPSK for differnt coding and no coding in dB

Note: to draw the reptation make num_rep=3 and to make it per information bit make A=1/sqrt(num_rep)(Remove the comment)

Comment

- The coding using same transmitted bits is better than same information bit and same information bit is better than no coding.
- To get better BER we need to use coding but with same transmitted bits.
- The Interleaver is important as if the channel is bad not all coded data drop or damaged and they will be transmitted at different time and that why coding using same information bits is better than no coding although no coding is better in AWGN
- Rayleigh fading change the amplitude only by attenuations.

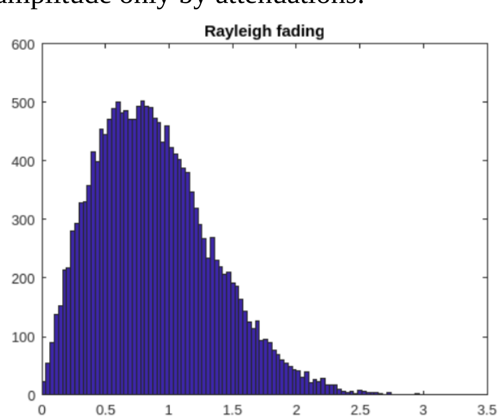
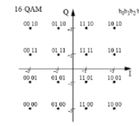


Figure 3 Rayleigh fading distribution

16QAM



Code

```
clc;clear;%close all;
NumberofBits = 36000;
Data=randi([0,1],[NumberofBits, 1]);
No = 10.^[-10:0.1:3];
%======================================================================%
% For Quadri-Phase Shift Keying (QPSK)
num_rep=3;

Data_rep=repelem(Data,num_rep);
A=1;%/sqrt(num_rep);
% Interleaver

for i=1:16:length(Data_rep)
    temp=reshape(Data_rep(i:i+15), 4, 4)';
    Data_rep(i:i+15)=temp(:);
end

% For Quadrature Amplitude Modulation (QAM)
i=1;
SI_QAM=zeros(length(Data_rep)/4, 1);
SQ_QAM=zeros(length(Data_rep)/4, 1);
da=zeros(length(Data_rep)/4, 4);
Noise=sqrt(No/2).*randn(length(Data_rep)/4, length(No))+sqrt(No/2).*randn(length(Data
_rep)/4, length(No))*j;
v1=randn(length(Data_rep)/4, 1);%+randn(length(Data_rep)/4, 1)*j;
v2=randn(length(Data_rep)/4, 1);%+randn(length(Data_rep)/4, 1)*j;
R=sqrt(v1.^2+v2.^2)/sqrt(2);
%R=normalize(sqrt(v1.^2+v2.^2)/sqrt(2));

for d=1:4:length(Data_rep)
    da(i, :)=Data_rep(d:d+3);
```

```
    if da(i, 1)==0 && da(i, 2)==0,
        SI_QAM(i)=-3;
    elseif da(i, 1)==0 && da(i, 2)==1,
        SI_QAM(i)=-1;
    elseif da(i, 1)==1 && da(i, 2)==1,
        SI_QAM(i)=1;
    elseif da(i, 1)==1 && da(i, 2)==0,
        SI_QAM(i)=3;
    end

    if da(i, 3)==0 && da(i, 4)==0,
        SQ_QAM(i)=-3;
    elseif da(i, 3)==0 && da(i, 4)==1,
        SQ_QAM(i)=-1;
    elseif da(i, 3)==1 && da(i, 4)==1,
        SQ_QAM(i)=1;
    elseif da(i, 3)==1 && da(i, 4)==0,
        SQ_QAM(i)=3;
    end
    i=i+1;
end
S_QAM=A*(SI_QAM+SQ_QAM*j);
X_QAM=R.*S_QAM+Noise;

avg_symbol_energy=(4*(3^2+3^2)+4*(1+1)+8*(3^2+1))/16;
Eb_QAM=avg_symbol_energy/4;
theoritical_error_QAM=3/8*erfc(sqrt(Eb_QAM./(2.5*No)));

QAM_RX=zeros(length(Data_rep), length(No));
QAM_RX(1:4:length(Data_rep), :)=real(X_QAM./R)>0;
QAM_RX(2:4:length(Data_rep), :)=abs(real(X_QAM./R))<2*A;
QAM_RX(3:4:length(Data_rep), :)=imag(X_QAM./R)>0;
QAM_RX(4:4:length(Data_rep), :)=abs(imag(X_QAM./R))<2*A;
```

```
% de-Interleaver
for no=1:length(No)
    for i=1:16:size(QAM_RX,1)
        temp2=reshape(QAM_RX(i:i+15, no), 4, 4)';
        QAM_RX(i:i+15, no)=temp2(:);
    end
end

if num_rep>1
    Y=zeros(size(QAM_RX, 1)/num_rep,size(QAM_RX, 2), num_rep);
    q=size(QAM_RX);
    for k2=1:q(2)
      jj=1;
      for ii=1:num_rep:q(1)
        Y(jj, k2, :)=QAM_RX(ii:ii+num_rep-1, k2);
        jj=jj+1;
      end
    end
    QAM_RX=mode(Y,3);
end
QAM_BER=sum(QAM_RX~=Data)/NumberofBits;
hold on
semilogy(10*log10(Eb_QAM./No),QAM_BER, 'k+');
%semilogy(10*log10(Eb_QAM./No),theoritical_error_QAM, 'k');
hold off
title('BER vs EB/No for 16QAM');
xlabel('Eb/No in dB'); ylabel('BER');
legend('BER QAM fading with repeation (same information bit)', 'BER of QAM fading wit
h no repeation', 'BER QAM fading with repeation (same transmitted bit)');
yticks(10.^[-6:1:0])
xticks([0:2:24])
ylim([10^-6 10^0])
xlim([0 18])
set(gca, 'YScale', 'log')
```

Graphs

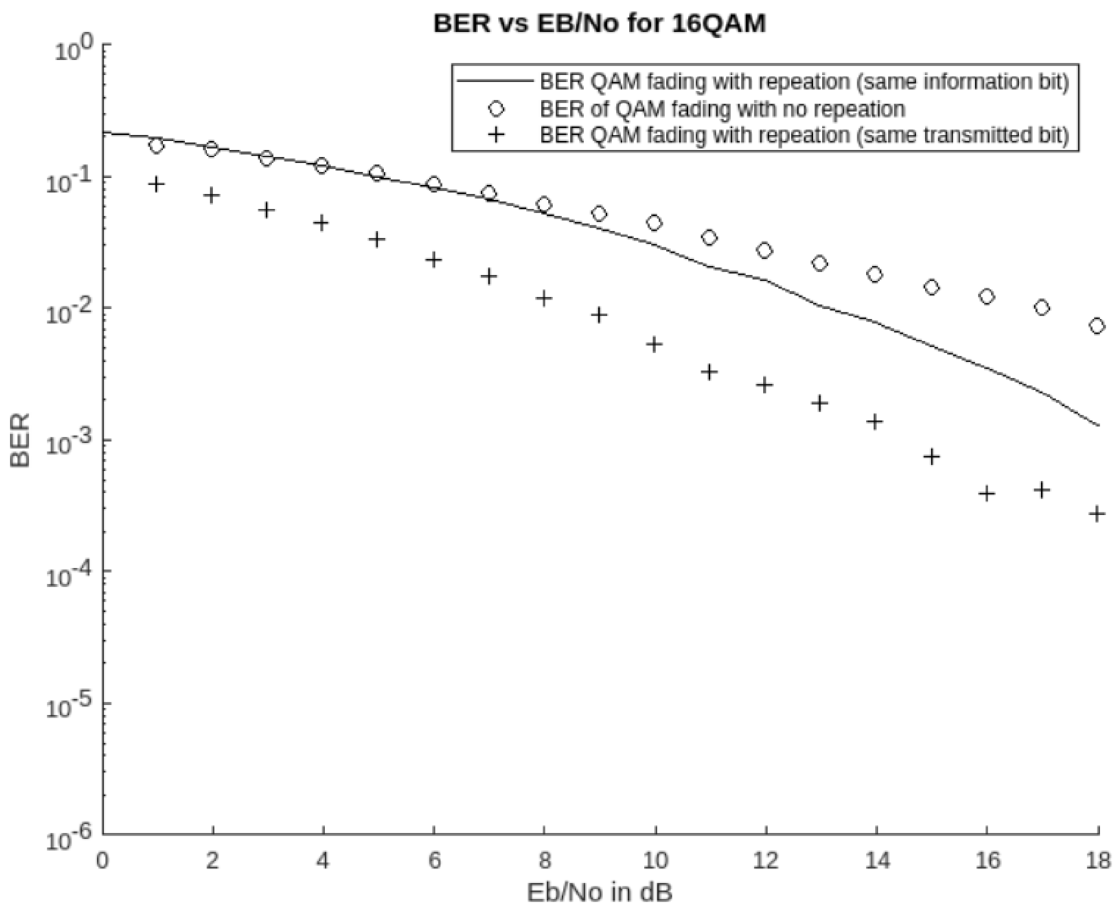
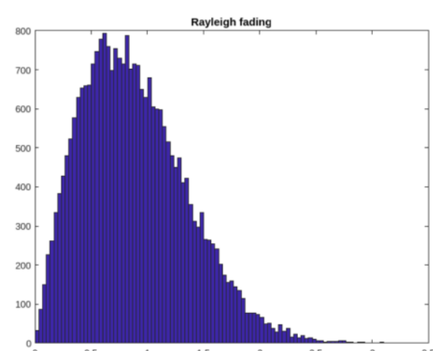


Figure 4 The relation between BER and QAM Eb/No for differnt coding and no coding in dB

Comment

- The coding using same transmitted bits is better than same information bit and same information bit is better than no coding.
- To get better BER we need to use coding but with same transmitted bits.
- The Interleaver is important as if the channel is bad not all coded data drop or damaged and they will be transmitted at different time and that why coding using same information bits is better than no coding although no coding is better in AWGN
- Rayleigh fading change the amplitude only by attenuations.



Comparison between QPSK and QAM

Graphs

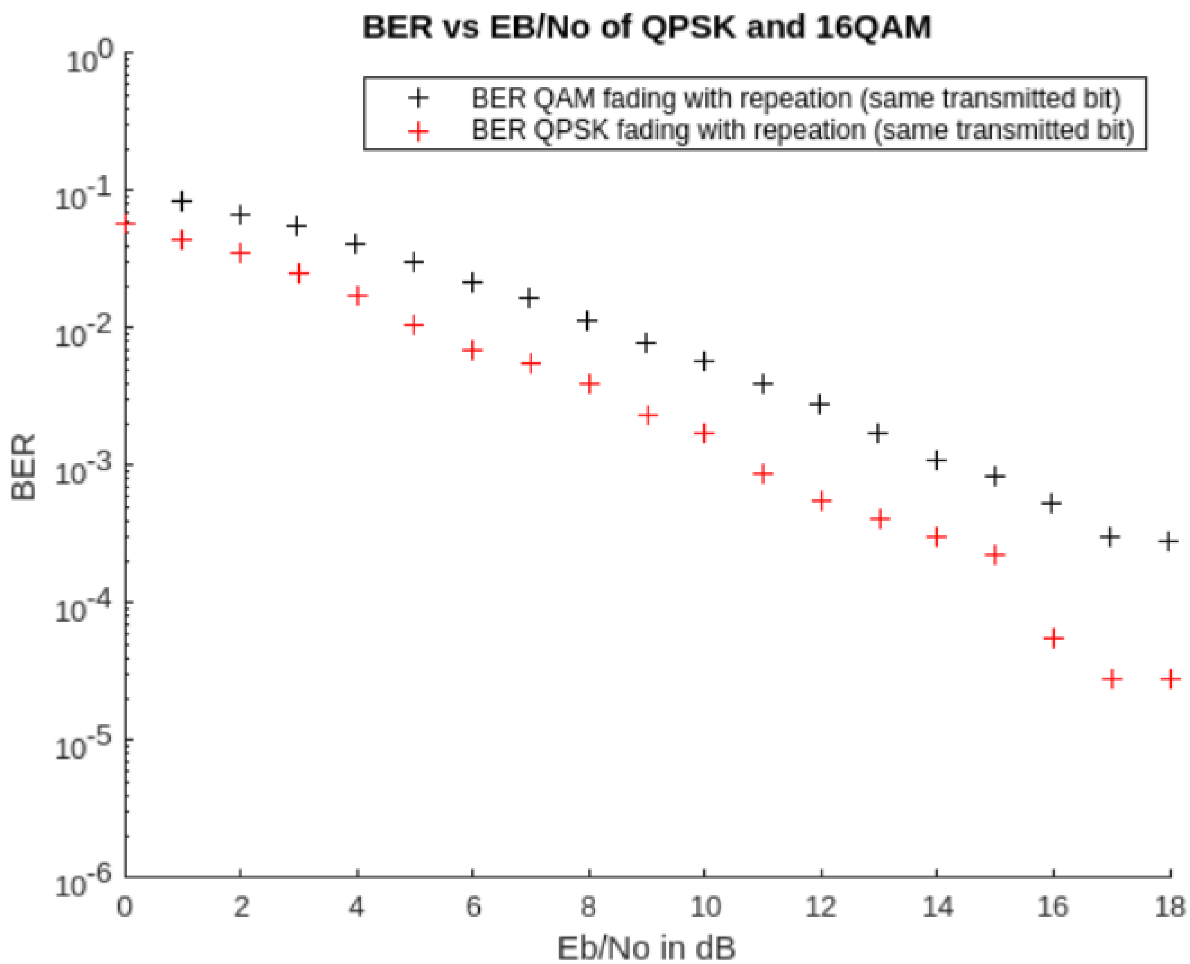


Figure 5 The relation between BER and QPSK and 16QAM Eb/No for coding in dB

Comment

- QAM is better in rate, but QPSK is better in BER

OFDM system simulation

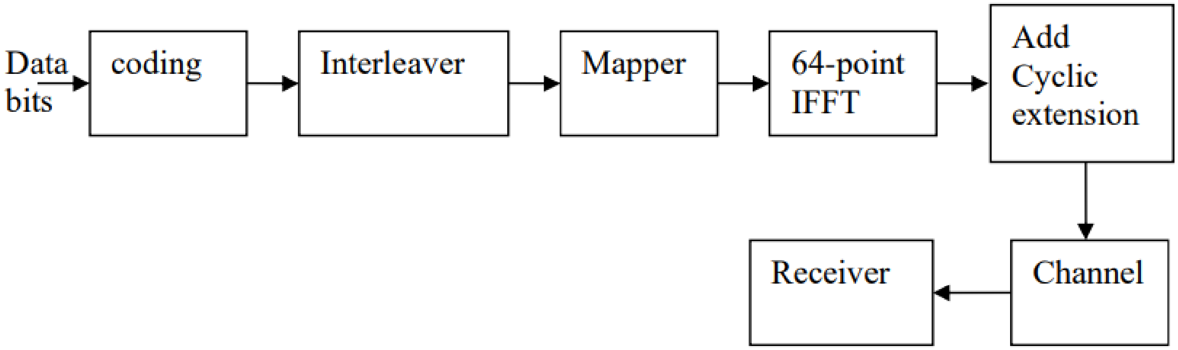
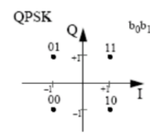


Figure 6 OFDM system

AWGN channel

QPSK



Code

```
clc;clear;%close all;
NumberofBits = 64000;
Data=randi([0,1],[NumberofBits, 1]);
No = 10.^[-10:0.1:3];
%======================================================================%
% For Quadri-Phase Shift Keying (QPSK)
num_rep=3;
Cp=16;% Cyclic extension
FFT_points=64;
Data_rep=repelem(Data,num_rep);
Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/(FFT_points*4))*FFT_points*4-length(Data_rep),1)];
A=1;%/sqrt(num_rep);
% Interleaver
r_qpsk=8;
c_qpsk=16;
for i=1:r_qpsk*c_qpsk:length(Data_rep)
    temp=reshape(Data_rep(i:i+r_qpsk*c_qpsk-1), r_qpsk, c_qpsk)';
    Data_rep(i:i+r_qpsk*c_qpsk-1)=temp(:);
end
```

```
i=1;
SI=zeros(length(Data_rep)/2, 1);
SQ=zeros(length(Data_rep)/2, 1);
da=zeros(length(Data_rep)/2, 2);
% Mapper
for d=1:2:length(Data_rep),
    da(i, :)=Data_rep(d:d+1);
    if da(i, 1)==0,
        SI(i)=-1;
    else
        SI(i)=1;
    end
    if da(i, 2)==0,
        SQ(i)=-1;
    else
        SQ(i)=1;
    end
    i=i+1;
end
S_QPSK=A*(SI+SQ*j);
% 64-IFFT block
inv_fft_qpsk=ifft(reshape(S_QPSK, [FFT_points, size(S_QPSK,1)/FFT_points]),FFT_points
);
% Cyclic extensions
for i=1:size(inv_fft_qpsk, 2)
    cyclic_ex_qpsk(:,i)=[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];
end
% channel
Noise=sqrt(permute(repmat(No/2,1,1,size(cyclic_ex_qpsk,2)),[1 3 2])).* ...
    (randn([size(cyclic_ex_qpsk), length(No)])+randn([size(cyclic_ex_qpsk), length(No
)])*j);
v1=randn(size(cyclic_ex_qpsk));
v2=randn(size(cyclic_ex_qpsk));
R=sqrt(v1.^2+v2.^2)/sqrt(2);
```

```
X_QPSK=R.*(cyclic_ex_qpsk)+Noise;
%% QPSK RX
% Equalizer
QPSK_RX=(X_QPSK./R);
% Removing cyclic extension
for ii=1:size(QPSK_RX,3)
    for i=1:size(QPSK_RX, 2)
        QPSK_RX1(:,i,ii)=QPSK_RX(Cp+1:end, i,ii);
        %[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];
    end
end
%QPSK_RX1=QPSK_RX(Cp+1:end, :);
% FFT
QPSK_RX2=fft(QPSK_RX1, FFT_points);
QPSK_RX2=reshape(QPSK_RX2, [size(S_QPSK,1),size(QPSK_RX2,3)]);
QPSK_RX=zeros(length(Data_rep), length(No));
QPSK_RX(1:2:length(Data_rep), :)=real(QPSK_RX2)>0 ;
QPSK_RX(2:2:length(Data_rep), :)=imag(QPSK_RX2)>0;
% de-inverlever
for no=1:length(No)
    for i=1:r_qpsk*c_qpsk:size(QPSK_RX,1)
        temp=reshape(QPSK_RX(i:i+r_qpsk*c_qpsk-1, no), c_qpsk, r_qpsk)';
        QPSK_RX(i:i+r_qpsk*c_qpsk-1, no)=temp(:);
    end
end
% remove repeations if any
if num_rep>1
    X=zeros(size(QPSK_RX, 1)/num_rep,size(QPSK_RX, 2), num_rep);
    s=size(QPSK_RX);
    for k=1:s(2)
      jj=1;
      for i=1:num_rep:s(1)
        X(jj, k, :)=QPSK_RX(i:i+num_rep-1, k);
        jj=jj+1;
```

```
        end
    end
    QPSK_RX=mode(X,3);
end
QPSK_BER=sum(QPSK_RX~=Data)/NumberofBits;
% Get average symbol and bit energy
avg_symbol_energy=4*(1+1)/4;
Eb_QPSK=avg_symbol_energy/2;
theoritical_error_QPSK=1/2*erfc(sqrt(Eb_QPSK./No));
hold on
semilogy(10*log10(Eb_QPSK./No),QPSK_BER, 'r+');
%semilogy(10*log10(Eb_QPSK./No),theoritical_error_QPSK, 'r');
hold off
title('BER vs EB/No');
xlabel('Eb/No in dB'); ylabel('BER');
legend('BER of QPSK fading no repetition', 'BER of QPSK fading repetition(same inform
ation bits)', 'BER of QPSK fading repetition(same transmitted bits)');
%legend('BER of QAM fading', 'BER QPSK AWGN', 'BER of QAM fading', 'QAM BER AWGN');
yticks(10.^[-6:1:0])
xticks([-30:2:30])
ylim([10^-1 10^0])
%xlim([0 40])
set(gca, 'YScale', 'log')
```

Graphs

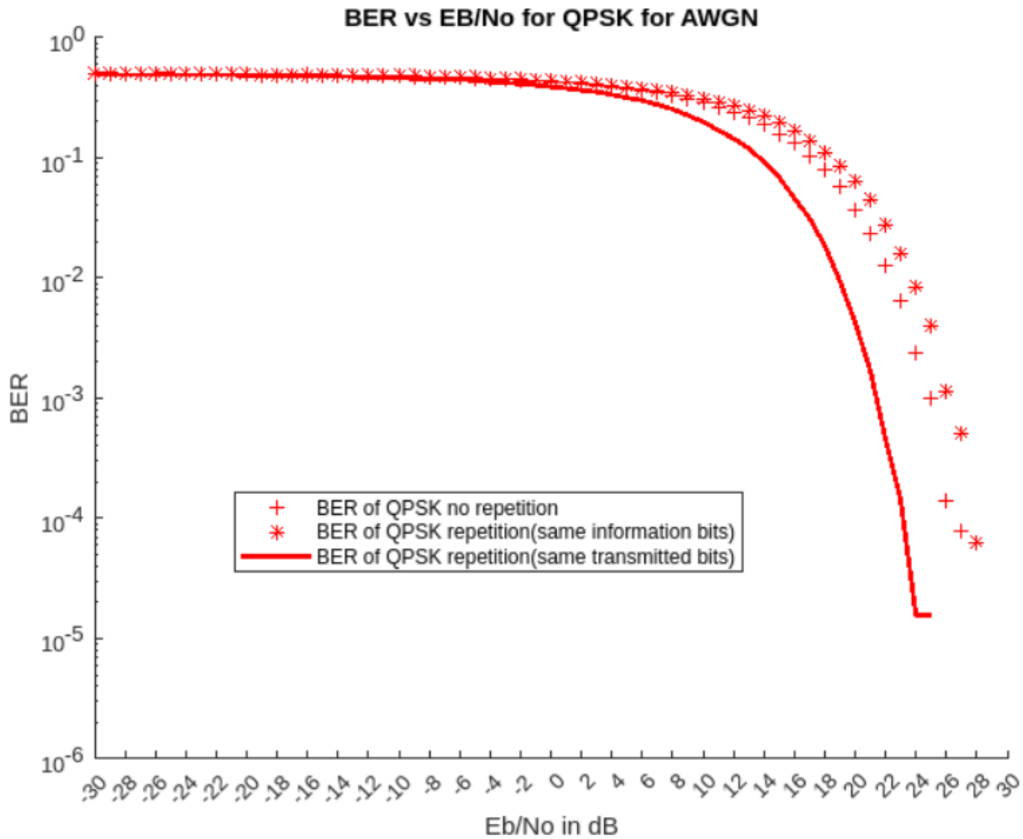


Figure 7 The relation between BER and Eb/No QPSK for differnt coding and no coding in dB for AWGN

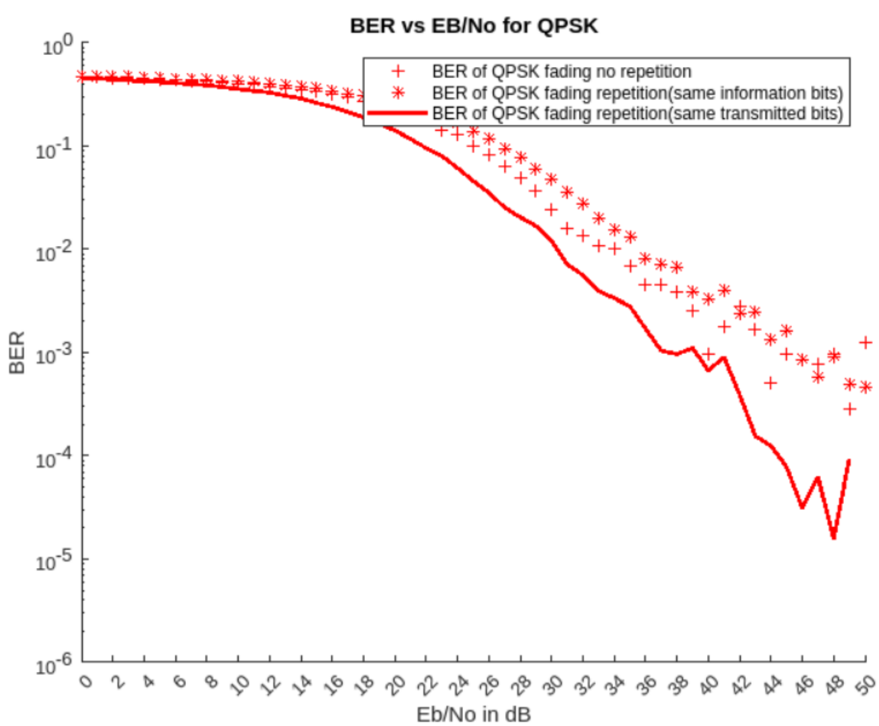
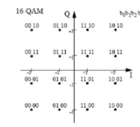


Figure 8 The relation between BER and Eb/No QPSK for differnt coding and no coding in dB for Rayleigh fading

Comment

- The coding using same transmitted bits is better than no coding and no coding is better than same information bit.
- To get better BER we need to use coding but with same transmitted bits.
- Using IFFT gives larger BER for single Carrier



16QAM

Code

```
clc;clear;close all;
NumberofBits = 64000;
Data=randi([0,1],[NumberofBits, 1]);
No = 10.^[-10:0.1:3];
%======================================================================%
% For Quadri-Phase Shift Keying (QPSK)
num_rep=3;
Cp=16;% Cyclic extension
FFT_points=64;
A=1;%/sqrt(num_rep);
Data_rep=repelem(Data,num_rep);
Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/(FFT_points*4))*FFT_points*4-length(Data_rep),1)];
% For Quadrature Amplitude Modulation (QAM)
Data_rep=repelem(Data,num_rep);
Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/FFT_points)*FFT_points-length(Data_rep),1)];
% Interleaver
r_qam=16;
c_qam=16;
for i=1:r_qam*c_qam:length(Data_rep)
    temp=reshape(Data_rep(i:i+r_qam*c_qam-1), r_qam, c_qam)';
    Data_rep(i:i+r_qam*c_qam-1)=temp(:);
end
i=1;
SI_QAM=zeros(length(Data_rep)/4, 1);
SQ_QAM=zeros(length(Data_rep)/4, 1);
```

```
da=zeros(length(Data_rep)/4, 4);
for d=1:4:length(Data_rep)
    da(i, :)=Data_rep(d:d+3);

    if da(i, 1)==0 && da(i, 2)==0,
        SI_QAM(i)=-3;
    elseif da(i, 1)==0 && da(i, 2)==1,
        SI_QAM(i)=-1;
    elseif da(i, 1)==1 && da(i, 2)==1,
        SI_QAM(i)=1;
    elseif da(i, 1)==1 && da(i, 2)==0,
        SI_QAM(i)=3;
    end

    if da(i, 3)==0 && da(i, 4)==0,
        SQ_QAM(i)=-3;
    elseif da(i, 3)==0 && da(i, 4)==1,
        SQ_QAM(i)=-1;
    elseif da(i, 3)==1 && da(i, 4)==1,
        SQ_QAM(i)=1;
    elseif da(i, 3)==1 && da(i, 4)==0,
        SQ_QAM(i)=3;
    end
    i=i+1;
end
S_QAM=A*(SI_QAM+SQ_QAM*j);

% 64-IFFT block
inv_fft_qam=ifft(reshape(S_QAM, [FFT_points, size(S_QAM,1)/FFT_points]),FFT_points);
% Cyclic extensions
for i=1:size(inv_fft_qam, 2)
    cyclic_ex_qam(:,i)=[inv_fft_qam(end-Cp+1:end,i); inv_fft_qam(:,i)];
end
% channel
```

```
Noise=sqrt(permute(repmat(No/2,1,1,size(cyclic_ex_qam,2)),[1 3 2])).* ...
    (randn([size(cyclic_ex_qam), length(No)])+randn([size(cyclic_ex_qam), length(No)]
)*j);
v1=randn(size(cyclic_ex_qam));
v2=randn(size(cyclic_ex_qam));
R=sqrt(v1.^2+v2.^2)/sqrt(2);
%R=normalize(sqrt(v1.^2+v2.^2)/sqrt(2));
X_QAM=R.*(cyclic_ex_qam)+Noise;
%% QPSK RX
% Equalizer
QAM_RX=(X_QAM./R);
% Removing cyclic extension
for ii=1:size(QAM_RX,3)
    for i=1:size(QAM_RX, 2)
        QAM_RX1(:,i,ii)=QAM_RX(Cp+1:end, i,ii);
        %[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];
    end
end
%QPSK_RX1=QPSK_RX(Cp+1:end, :);
% FFT
QAM_RX2=fft(QAM_RX1, FFT_points);
QAM_RX2=reshape(QAM_RX2, [size(S_QAM,1),size(QAM_RX2,3)]);

avg_symbol_energy=(4*(3^2+3^2)+4*(1+1)+8*(3^2+1))/16;
Eb_QAM=avg_symbol_energy/4;
theoritical_error_QAM=3/8*erfc(sqrt(Eb_QAM./(2.5*No)));

QAM_RX=zeros(length(Data_rep), length(No));
QAM_RX(1:4:length(Data_rep), :)=real(QAM_RX2)>0;
QAM_RX(2:4:length(Data_rep), :)=abs(real(QAM_RX2))<2*A;

QAM_RX(3:4:length(Data_rep), :)=imag(QAM_RX2)>0;
QAM_RX(4:4:length(Data_rep), :)=abs(imag(QAM_RX2))<2*A;
```

```
for no=1:length(No)
    for i=1:r_qam*c_qam:size(QAM_RX,1)
        temp2=reshape(QAM_RX(i:i+r_qam*c_qam-1, no), c_qam, r_qam)';
        QAM_RX(i:i+r_qam*c_qam-1, no)=temp2(:);
    end
end

if num_rep>1
    Y=zeros(size(QAM_RX, 1)/num_rep,size(QAM_RX, 2), num_rep);
    q=size(QAM_RX);
    for k2=1:q(2)
      jj=1;
      for ii=1:num_rep:q(1)
        Y(jj, k2, :)=QAM_RX(ii:ii+num_rep-1, k2);
        jj=jj+1;
      end
    end
    QAM_RX=mode(Y,3);
End

QAM_BER=sum(QAM_RX~=Data)/NumberofBits;
hold on
semilogy(10*log10(Eb_QAM./No),QAM_BER, 'k');
%semilogy(10*log10(Eb_QAM./No),theoritical_error_QAM, 'k');
hold off
title('BER vs EB/No');
xlabel('Eb/No in dB'); ylabel('BER');

legend('BER of QPSK no repetition', 'BER of QPSK repetition(same information bits)',
'BER of QPSK repetition(same transmitted bits)');
%legend('BER of QAM fading', 'BER QPSK AWGN', 'BER of QAM fading', 'QAM BER AWGN');

title('BER vs EB/No');
xlabel('Eb/No in dB'); ylabel('BER');
```

```
yticks(10.^[-6:1:0])
xticks([-20:2:52])
ylim([10^-6 10^0])
xlim([-20 52])
set(gca, 'YScale', 'log')
```

Graphs

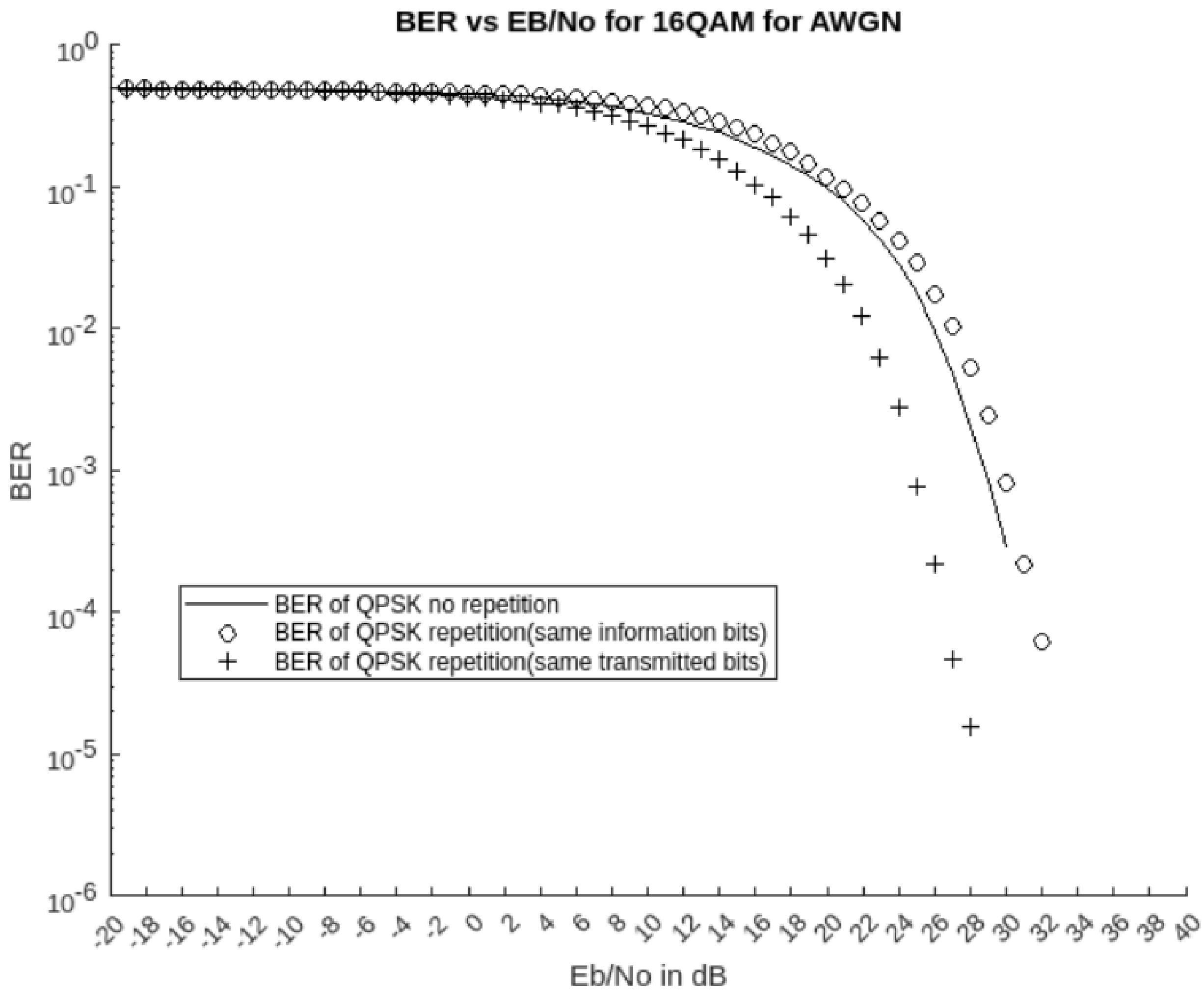


Figure 9The relation between BER and Eb/No 16QAM for differnt coding and no coding in dB

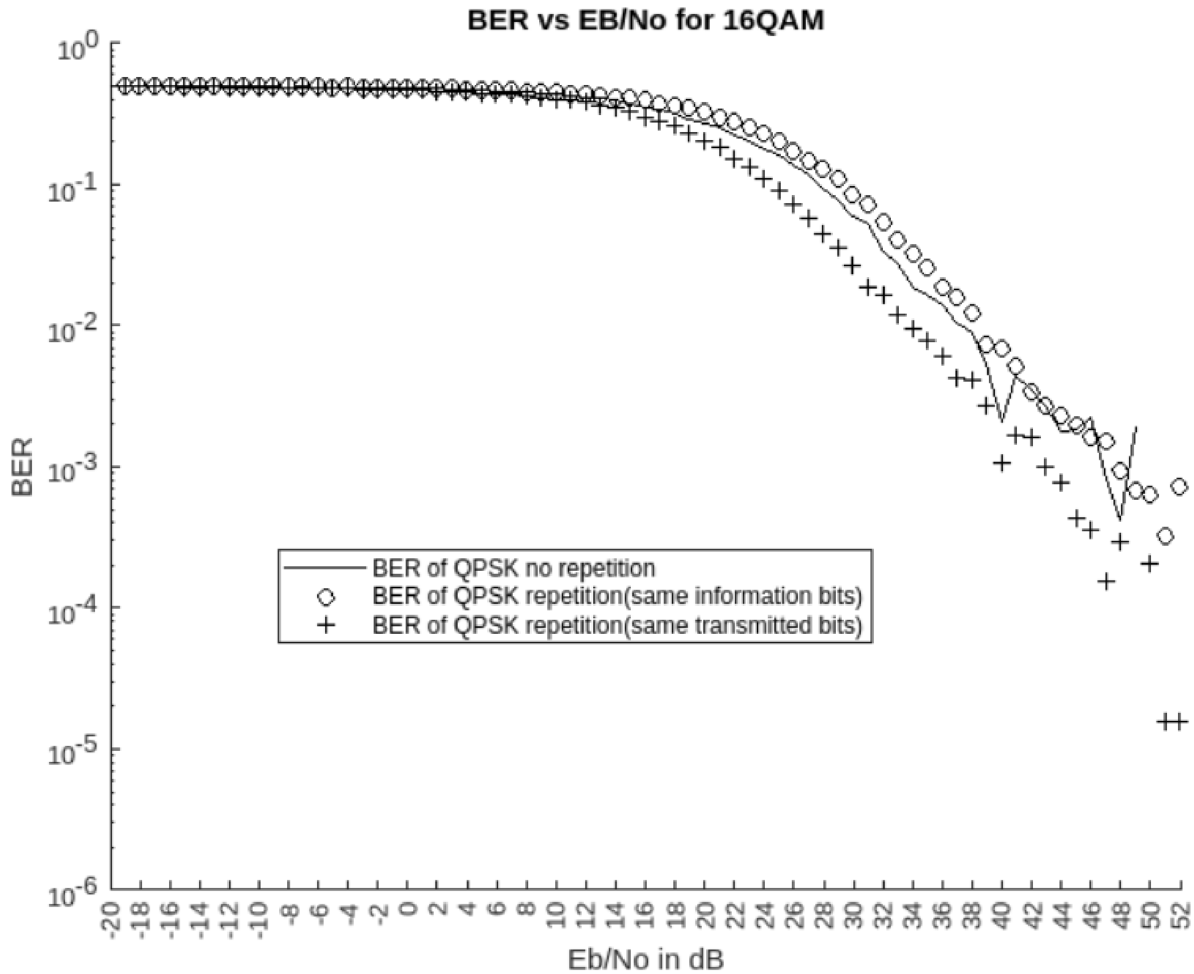


Figure 10 The relation between BER and Eb/No 16QAM for differnt coding and no coding in dB for Rayleigh fading

Comment

- The coding using same transmitted bits is better than no coding and no coding is better than same information bit.
- To get better BER we need to use coding but with same transmitted bits.
- Using IFFT gives larger BER than single Carrier

Comparison between QPSK and 16QAM

Graphs

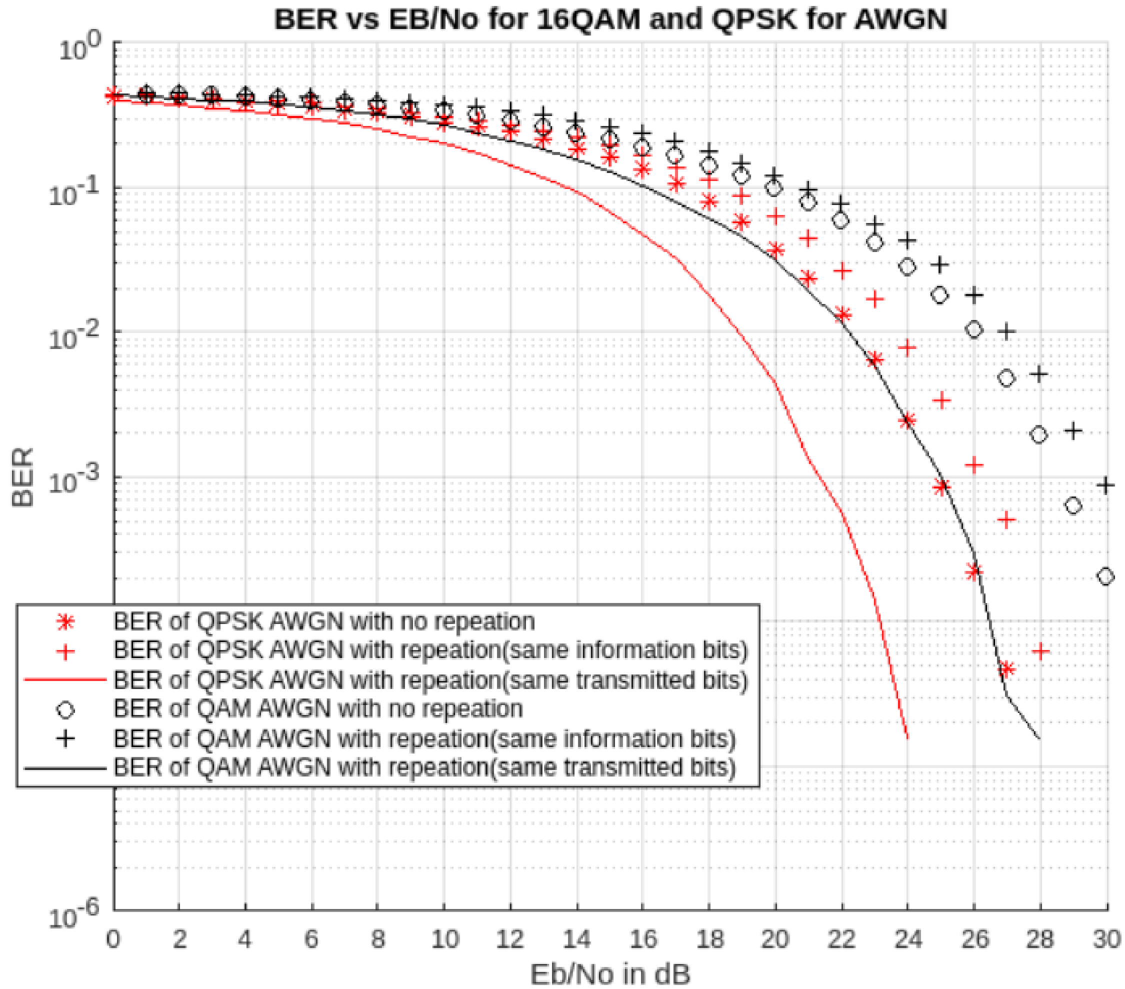


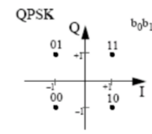
Figure 11 The relation between BER and QPSK and 16QAM Eb/No for coding and no coding in dB

Comment

- QAM is better in rate, but QPSK is better in BER

Frequency selective Fading channel

QPSK



Code

```
clc;clear;close all;

NumberofBits = 64000;

Data=randi([0,1],[NumberofBits, 1]);

No = 10.^[-10:0.1:3];

h=[0.4 0 0.26 0 0 0.4 0 0.6 0 0.5];

%======================================================================%

% For Quadri-Phase Shift Keying (QPSK)

num_rep=3;

Cp=16;% Cyclic extension

FFT_points=64;

A=1/sqrt(num_rep);

Data_rep=repelem(Data,num_rep);

Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/(FFT_points*4))*FFT_points*4-length(D
ata_rep),1)];

% Interleaver

r_qpsk=8;

c_qpsk=16;

for i=1:r_qpsk*c_qpsk:length(Data_rep)

    temp=reshape(Data_rep(i:i+r_qpsk*c_qpsk-1), r_qpsk, c_qpsk)';

    Data_rep(i:i+r_qpsk*c_qpsk-1)=temp(:);

End


i=1;

SI=zeros(length(Data_rep)/2, 1);

SQ=zeros(length(Data_rep)/2, 1);

da=zeros(length(Data_rep)/2, 2);

% Mapper

for d=1:2:length(Data_rep),

    da(i, :)=Data_rep(d:d+1);
```

```
    if da(i, 1)==0,
        SI(i)=-1;
    else
        SI(i)=1;
    end

    if da(i, 2)==0,
        SQ(i)=-1;
    else
        SQ(i)=1;
    end

    i=i+1;
end
S_QPSK=A*(SI+SQ*j);

% 64-IFFT block
inv_fft_qpsk=ifft(reshape(S_QPSK, [FFT_points, size(S_QPSK,1)/FFT_points]),FFT_points
);
% Cyclic extensions
for i=1:size(inv_fft_qpsk, 2)
    cyclic_ex_qpsk(:,i)=[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];
end
% channel
conv_cyclic=cyclic_ex_qpsk(:);
Noise=sqrt(No/2).* ...
    (randn([size(conv_cyclic,1), length(No)])+randn([size(conv_cyclic,1), length(No)]
)*j);
%Noise=zeros(size(cyclic_ex_qpsk(:),1), length(No));
X_QPSK=fft(conv_cyclic, length(conv_cyclic));
H=fft(h, length(conv_cyclic));
X_QPSK = X_QPSK.*H';
%conv(conv_cyclic,h);
X_QPSK=ifft(X_QPSK, length(conv_cyclic))+Noise;
```

```
%% QPSK RX
% Equalizer
QPSK_RX_H=ifft(fft(X_QPSK, length(conv_cyclic))./H',length(conv_cyclic));
for kk=1:length(No)
    QPSK_RX7(:,:, kk)=reshape(QPSK_RX_H(:, kk), FFT_points+Cp,size(QPSK_RX_H,1)/(FFT_
points+Cp));
end
% Removing cyclic extension
for ii=1:size(QPSK_RX7,3)
    for i=1:size(QPSK_RX7, 2)
        QPSK_RX1(:,i,ii)=QPSK_RX7(Cp+1:end, i,ii);
        %[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];
    end
end
% FFT
QPSK_RX2=fft(QPSK_RX1, FFT_points);
QPSK_RX2=reshape(QPSK_RX2, [size(S_QPSK,1),size(QPSK_RX2,3)]);
QPSK_RX=zeros(length(Data_rep), length(No));
QPSK_RX(1:2:length(Data_rep), :)=real(QPSK_RX2)>0 ;
QPSK_RX(2:2:length(Data_rep), :)=imag(QPSK_RX2)>0;

% de-inverlever
for no=1:length(No)
    for i=1:r_qpsk*c_qpsk:size(QPSK_RX,1)
        temp=reshape(QPSK_RX(i:i+r_qpsk*c_qpsk-1, no), c_qpsk, r_qpsk)';
        QPSK_RX(i:i+r_qpsk*c_qpsk-1, no)=temp(:);
    end
end

% remove repeations if any
if num_rep>1
    X=zeros(size(QPSK_RX, 1)/num_rep,size(QPSK_RX, 2), num_rep);
    s=size(QPSK_RX);
    for k=1:s(2)
```

```
      jj=1;
      for i=1:num_rep:s(1)
        X(jj, k, :)=QPSK_RX(i:i+num_rep-1, k);
        jj=jj+1;
      end
    end
    QPSK_RX=mode(X,3);
end
QPSK_BER=sum(QPSK_RX~=Data)/NumberofBits;

% Get average symbola dand bit energy
avg_symbol_energy=4*(1+1)/4;
Eb_QPSK=avg_symbol_energy/2;
theoritical_error_QPSK=1/2*erfc(sqrt(Eb_QPSK./No));
hold on
semilogy(10*log10(Eb_QPSK./No),QPSK_BER, 'r');
hold off
title('BER vs EB/No of QPSK frequency selective');
xlabel('Eb/No in dB'); ylabel('BER');

legend('BER of QPSK with no repeation', 'BER of QPSK with repeation(same information bits)', ...
    'BER of QPSK with repeation(same transmitted bits)');

title('BER vs EB/No');
xlabel('Eb/No in dB'); ylabel('BER');
yticks(10.^[-6:1:0])
xticks([-20:2:50])
ylim([10^-6 10^0])
xlim([-20 50])
set(gca, 'YScale', 'log')
```

Graphs

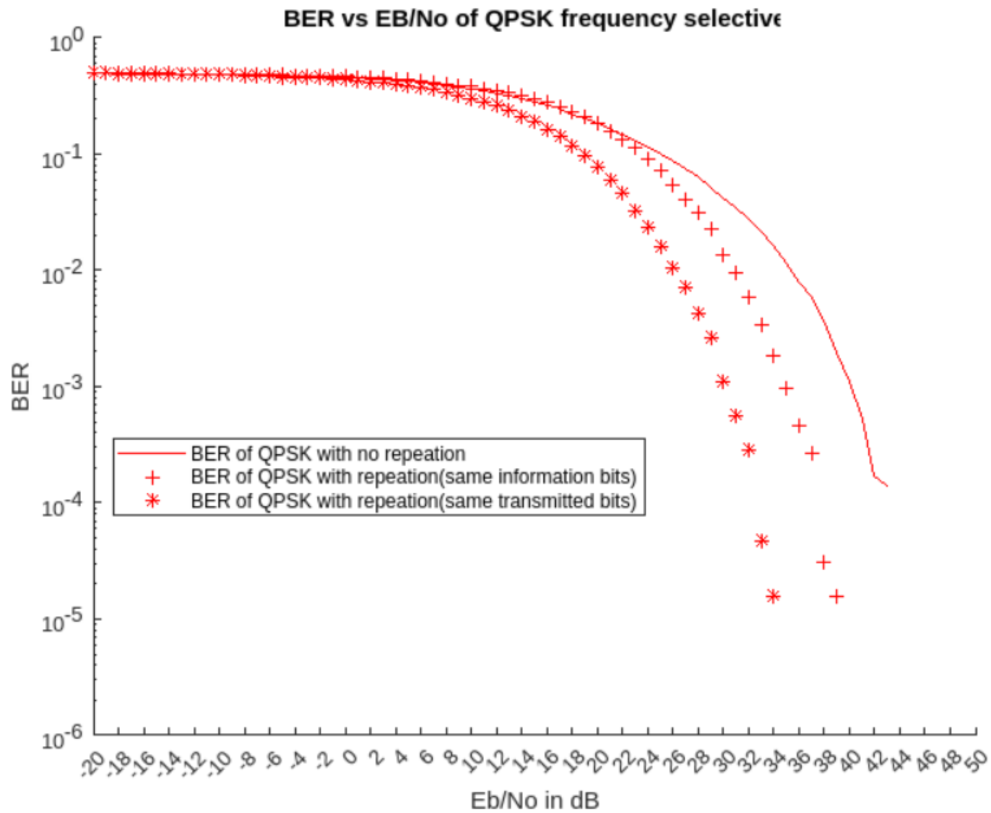
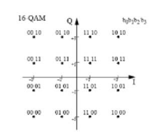


Figure 11 The relation between BER and Eb/No QPSK for differnt coding and no coding in dB for frequency selective

Comment

- The coding using same transmitted bits is better than no coding and no coding is better than same information bit.
- To get better BER we need to use coding but with same transmitted bits.
- To check which is better AWGN or frequency selective while using QPSK we need to compare between them at the end.



16QAM

Code

```
clc;clear;close all;
NumberofBits = 64000;
Data=randi([0,1],[NumberofBits, 1]);
No = 10.^[-10:0.1:3];
h=[0.4 0 0.26 0 0 0.4 0 0.6 0 0.5];
%=========================================================================%
num_rep=3;
Cp=16;% Cyclic extension
```

```
FFT_points=64;

A=1/sqrt(num_rep);

Data_rep=repelem(Data,num_rep);

Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/(FFT_points*4))*FFT_points*4-length(D
ata_rep),1)];

% For Quadrature Amplitude Modulation (QAM)

Data_rep=repelem(Data,num_rep);

Data_rep=[Data_rep; zeros(ceil(length(Data_rep)/FFT_points)*FFT_points-length(Data_re
p),1)];

% Interleaver

r_qam=16;

c_qam=16;

for i=1:r_qam*c_qam:length(Data_rep)

    temp=reshape(Data_rep(i:i+r_qam*c_qam-1), r_qam, c_qam)';

    Data_rep(i:i+r_qam*c_qam-1)=temp(:);

end

i=1;

SI_QAM=zeros(length(Data_rep)/4, 1);

SQ_QAM=zeros(length(Data_rep)/4, 1);

da=zeros(length(Data_rep)/4, 4);


for d=1:4:length(Data_rep)

    da(i, :)=Data_rep(d:d+3);


    if da(i, 1)==0 && da(i, 2)==0,

        SI_QAM(i)=-3;

    elseif da(i, 1)==0 && da(i, 2)==1,

        SI_QAM(i)=-1;

    elseif da(i, 1)==1 && da(i, 2)==1,

        SI_QAM(i)=1;

    elseif da(i, 1)==1 && da(i, 2)==0,

        SI_QAM(i)=3;

    end
```

```
    if da(i, 3)==0 && da(i, 4)==0,
        SQ_QAM(i)=-3;
    elseif da(i, 3)==0 && da(i, 4)==1,
        SQ_QAM(i)=-1;
    elseif da(i, 3)==1 && da(i, 4)==1,
        SQ_QAM(i)=1;
    elseif da(i, 3)==1 && da(i, 4)==0,
        SQ_QAM(i)=3;
    end
    i=i+1;
end
S_QAM=A*(SI_QAM+SQ_QAM*j);

% 64-IFFT block
inv_fft_qam=ifft(reshape(S_QAM, [FFT_points, size(S_QAM,1)/FFT_points]),FFT_points);
% Cyclic extensions
for i=1:size(inv_fft_qam, 2)
    cyclic_ex_qam(:,i)=[inv_fft_qam(end-Cp+1:end,i); inv_fft_qam(:,i)];
end
% channel
conv_cyclic=cyclic_ex_qam(:);
Noise=sqrt(No/2).* ...
    (randn([size(conv_cyclic,1), length(No)])+randn([size(conv_cyclic,1), length(No)]
)*j);
%Noise=zeros(size(cyclic_ex_qpsk(:),1), length(No));
X_QAM=fft(conv_cyclic, length(conv_cyclic));
H=fft(h, length(conv_cyclic));
X_QAM = X_QAM.*H';
%conv(conv_cyclic,h);
X_QAM=ifft(X_QAM, length(conv_cyclic))+Noise;
%% QPSK RX
% Equalizer
QAM_RX_H=ifft(fft(X_QAM, length(conv_cyclic))./H',length(conv_cyclic));
for kk=1:length(No)
```

```
    QAM_RX7(:,:, kk)=reshape(QAM_RX_H(:, kk), FFT_points+Cp,size(QAM_RX_H,1)/(FFT_points+Cp));

end

% Removing cyclic extension

for ii=1:size(QAM_RX7,3)

    for i=1:size(QAM_RX7, 2)

        QAM_RX1(:,i,ii)=QAM_RX7(Cp+1:end, i,ii);

        %[inv_fft_qpsk(end-Cp+1:end,i); inv_fft_qpsk(:,i)];

    end

end

%QPSK_RX1=QPSK_RX(Cp+1:end, :);

% FFT

QAM_RX2=fft(QAM_RX1, FFT_points);

QAM_RX2=reshape(QAM_RX2, [size(S_QAM,1),size(QAM_RX2,3)]);


avg_symbol_energy=(4*(3^2+3^2)+4*(1+1)+8*(3^2+1))/16;

Eb_QAM=avg_symbol_energy/4;

theoritical_error_QAM=3/8*erfc(sqrt(Eb_QAM./(2.5*No)));


QAM_RX=zeros(length(Data_rep), length(No));

QAM_RX(1:4:length(Data_rep), :)=real(QAM_RX2)>0;

QAM_RX(2:4:length(Data_rep), :)=abs(real(QAM_RX2))<2*A;


QAM_RX(3:4:length(Data_rep), :)=imag(QAM_RX2)>0;

QAM_RX(4:4:length(Data_rep), :)=abs(imag(QAM_RX2))<2*A;


for no=1:length(No)

    for i=1:r_qam*c_qam:size(QAM_RX,1)

        temp2=reshape(QAM_RX(i:i+r_qam*c_qam-1, no), c_qam, r_qam)';

        QAM_RX(i:i+r_qam*c_qam-1, no)=temp2(:);

    end

end


if num_rep>1
```

```
    Y=zeros(size(QAM_RX, 1)/num_rep,size(QAM_RX, 2), num_rep);
    q=size(QAM_RX);
    for k2=1:q(2)
      jj=1;
      for ii=1:num_rep:q(1)
        Y(jj, k2, :)=QAM_RX(ii:ii+num_rep-1, k2);
        jj=jj+1;
      end
    end
    QAM_RX=mode(Y,3);
end

QAM_BER=sum(QAM_RX~=Data)/NumberofBits;

hold on
semilogy(10*log10(Eb_QAM./No),QAM_BER, 'ko');
hold off
title('BER vs EB/No of 16QAM frequency selective');
xlabel('Eb/No in dB'); ylabel('BER');
legend('BER of QAM with no repeation', 'BER of QAM with repeation(same information bi
ts)', 'BER of QAM with repeation(same transmitted bits)');



yticks(10.^[-6:1:0])
xticks([-20:2:50])
ylim([10^-6 10^0])
xlim([-20 50])
set(gca, 'YScale', 'log')
```

Graphs

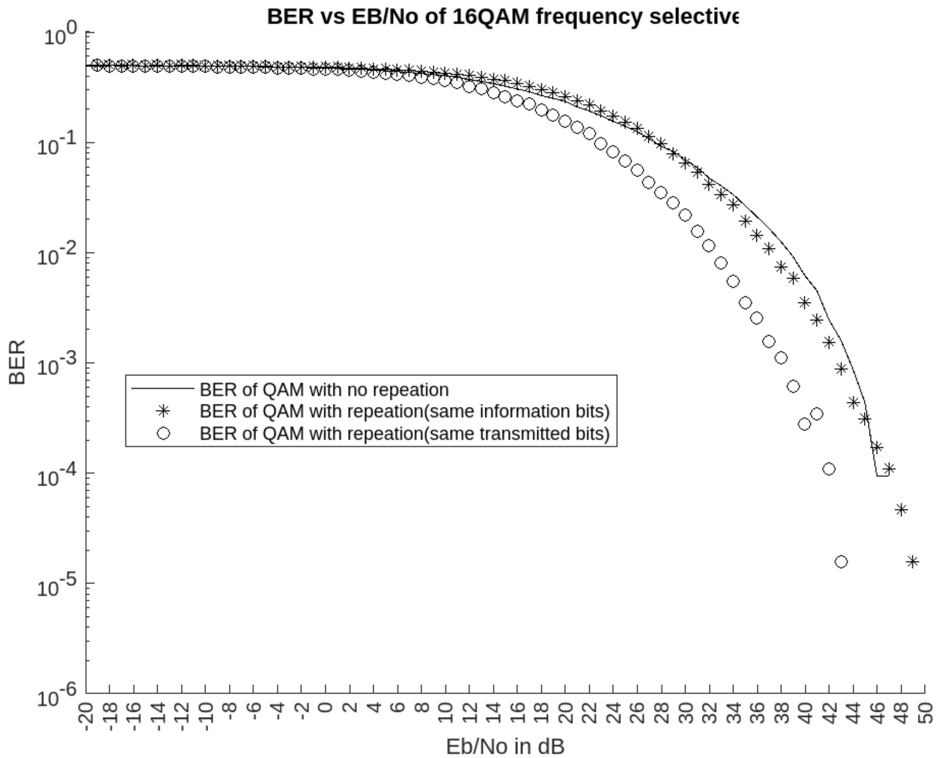


Figure 12 The relation between BER and Eb/No 16QAM for differnt coding and no coding in dB for frequency selective

Comment

- The coding using same transmitted bits is better than no coding and no coding is better than same information bit.
- To get better BER we need to use coding but with same transmitted bits.
- To check which is better AWGN or frequency selective while using QAM we need to compare between them at the end.

Comparison between QPSK and 16QAM

Graphs

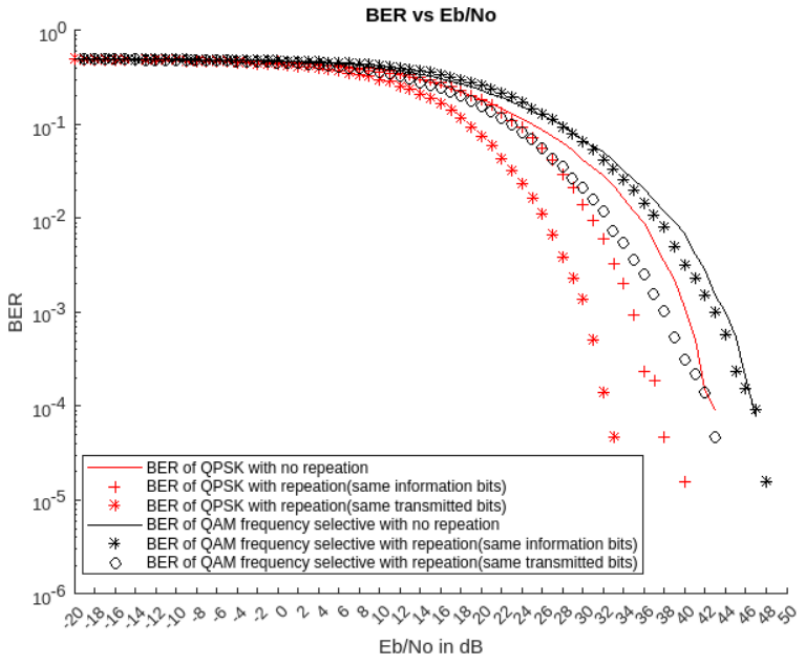


Figure 13 The relation between BER and QPSK and 16QAM Eb/No for coding and no coding in dB

Comment

- QAM is better in rate, but QPSK is better in BER

Comparison between AWGN VS frequency selective

Graphs

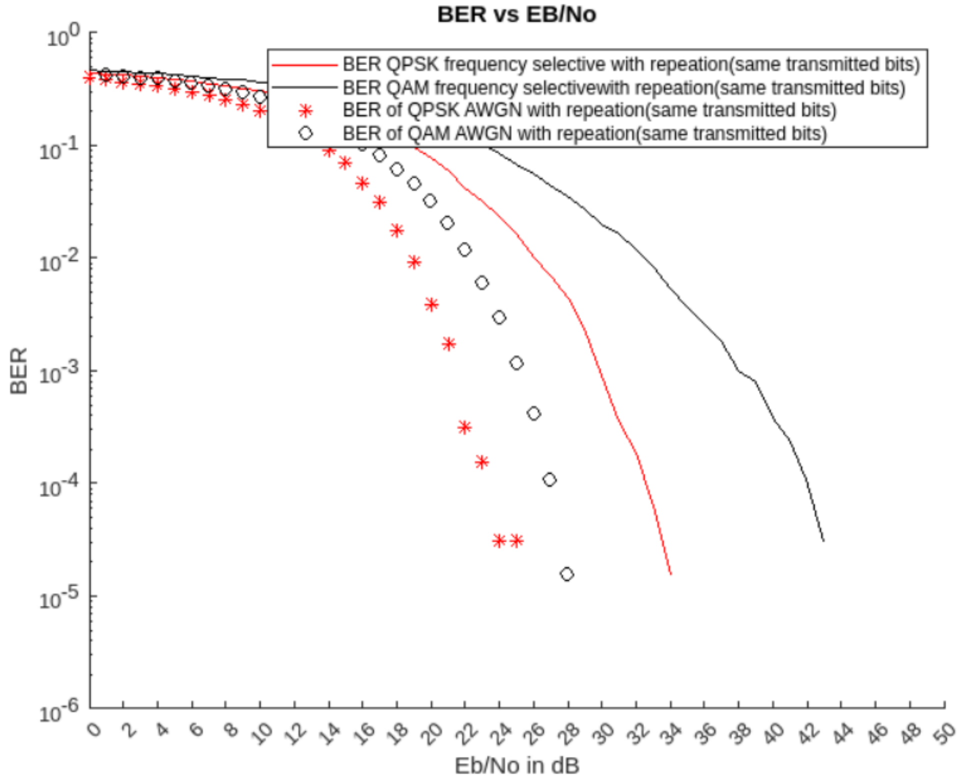


Figure 14 The relation between BER for AWGN VS frequency selective for Eb/No for coding and no coding in dB

Comment

- AWGN has better BER than Frequency selective and that was expected as the channel get worser the BER will get worser

Water-filling

Code

```
clc;clear;close all;
N_points=16;
N_channel=16;
h=[0.4 0 0.26 0 0 0.4 0 0.6 0 0.5];
%HF=abs( fftshift(fft(h, N_points)).^2);
HF=abs( fft(h, N_points).^2);
h_squared=HF;
SNR_gap =2;
noise=1;
p_total=200;
p_array=zeros(1, length(h));
gn_h=SNR_gap*noise.^2./h_squared;
gn_h_=gn_h;
k=min(gn_h);

while p_total>0
  indx=find(gn_h==min(gn_h));

  if (max(gn_h(find(gn_h!=inf)))==min(gn_h))
    indx=find(gn_h!=inf);
    p_array(indx)=p_total/length(indx);
   elseif ((min(gn_h(find(gn_h!=min(gn_h))))-min(gn_h))> p_total/length(indx))
    indx=find(gn_h<min(gn_h(find(gn_h!=min(gn_h)))));
    p_array(indx)=p_total/length(indx);
   else
    p_array(indx)=min(gn_h(find(gn_h!=min(gn_h))))-min(gn_h);
  endif

  if p_total/length(indx)>=p_array(indx)(1)
    gn_h(indx)=gn_h(indx)+p_array(indx);
  endif
```

```
  k=k+p_array(indx)(1)
  p_total=p_total-sum(p_array(indx));
end

Power=gn_h-gn_h_;

bar([gn_h_', Power'],'stacked')
title('Water-Filling     Interpretation');
xlabel('Subchannel       index    n');
legend('Γσ^2/g^2', 'Power');

Rate=1/2*sum(log2(1+Power(Power>0)./gn_h_(gn_h_!=inf)))
```

Graphs

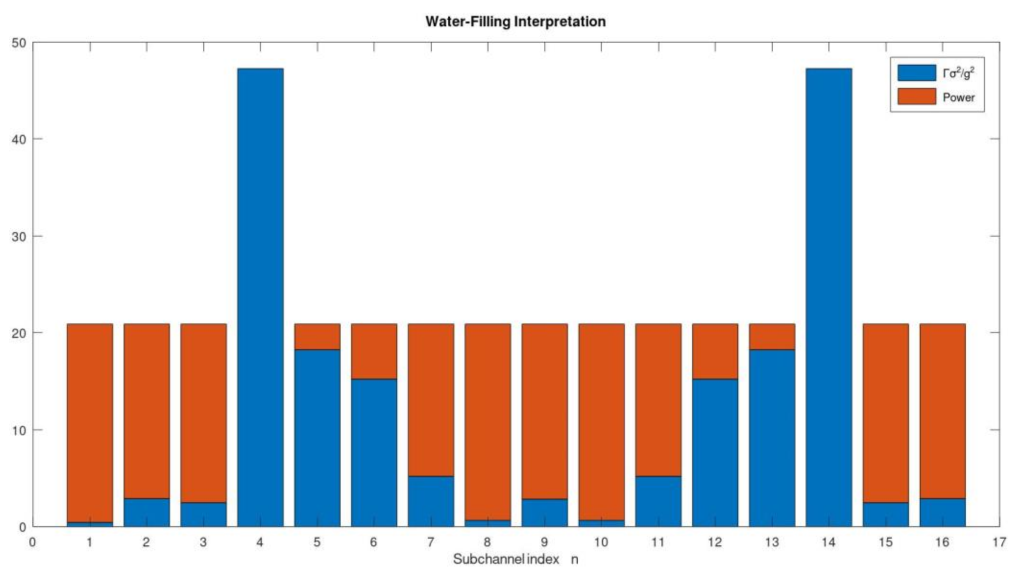


Figure 15 Water-filling

```
>> k
k = 20.898
>> Power
Power =

 Columns 1 through 12:

   20.4696   18.0117   18.4283         0    2.6501    5.6753   15.7098   20.2580   18.0638   20.2580   15.7098    5.6753

 Columns 13 through 16:

    2.6501         0   18.4283   18.0117
```

Comment

- As gain increase and noise and SNR gap(Γ) decreases the transmitted power increases but at some point, the transmitted power will be redistributed as the following figure and at some point the power may me the same to get higher rate.

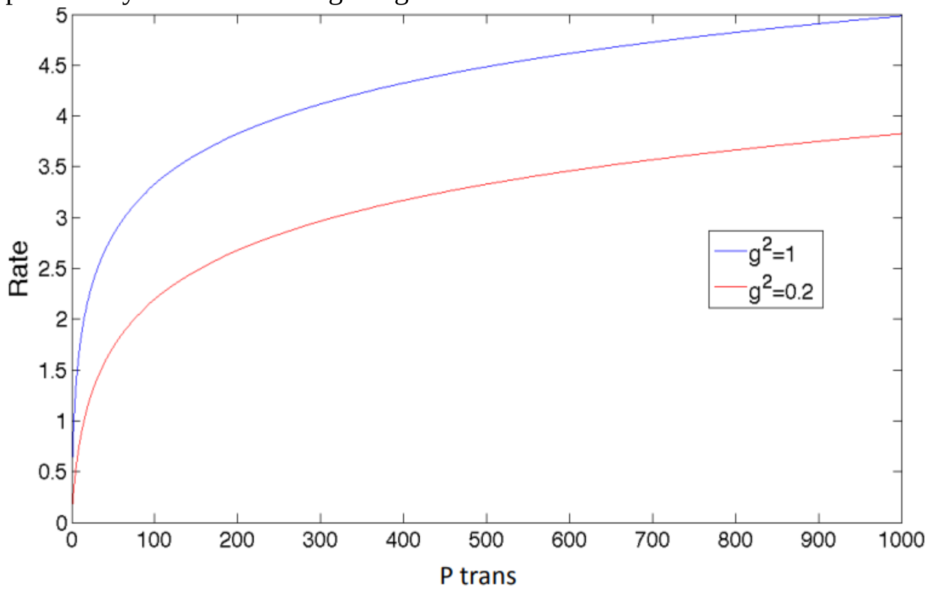


Figure 16 Relation between rate and power transmitted

- The Water-filling can be calculated analytically and graphically but here we calculate it graphically.