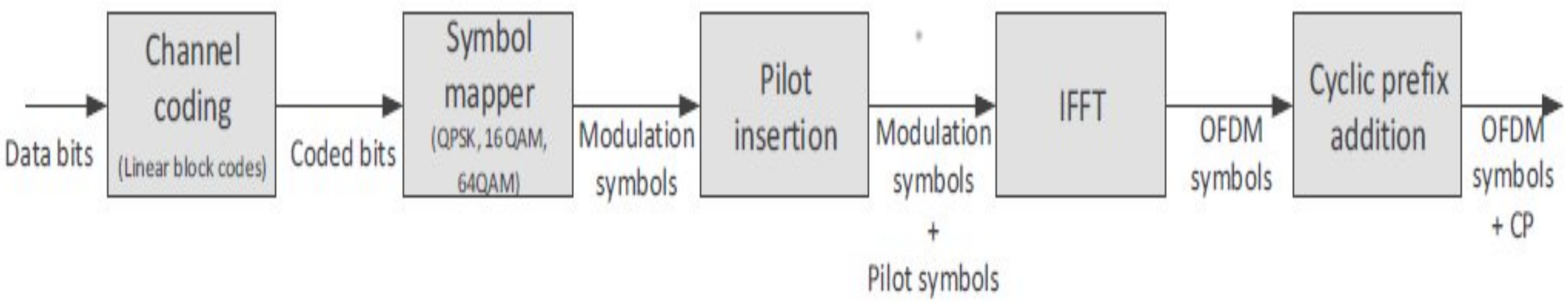
Digital Communication Project

1.Transmitter



OFDM Tx block diagram

Part 1: channel options

MATLAB code:

```
%% Channel Options
EbNo_range=(0:3:40);                          %Eb/No range of simulation dB
NumberFramesPerSNR=1e3;                       %Number of frames sent for every SNR value
ModulationOrder=input('Please insert modulation order[16,64,4]: M = ');  %The number of sent waveforms (M)
Ch_Noise_Type =input('Please insert channel type[1:AWGN, 2:Multi-Path]: ');
Ch_MIMO_SETUP =input('Please insert MIMO setup[1:SISO, 2:SIMO]: ');
NumberBitsPerFrame=1e3*log2(ModulationOrder);    %Number of bits sent for every frame
```

Here we choose which type of channel modulation & noise we desire and the number of antennas and calculate some variables.

Part 2: creating generator matrix

MATLAB code:

```
% (3,6)Generator matrix
Parity_matrix = [1,1,0;1,0,1;0,1,1];
Generator_matrix_1 = [eye(3) Parity_matrix];
% (2,4)Generator matrix
Parity_matrix_2 = [1,1;1,0];
Generator_matrix_2 = [eye(2) Parity_matrix_2];
% (2,1)Generator matrix
Generator_matrix_3 = [1 1];
```

```
if ModulationOrder == 16
    Generator_matrix = Generator_matrix_2;
    NumberInfoBits =2;
elseif ModulationOrder == 64
    Generator_matrix = Generator_matrix_1;
    NumberInfoBits =3;
elseif ModulationOrder == 4
    Generator_matrix = Generator_matrix_3;
    NumberInfoBits = 1;
end
```

Each type of modulation has its own generation matrix to encode it.

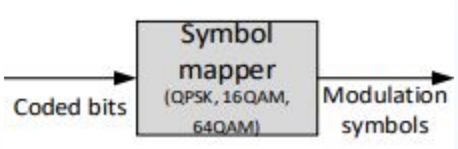
Part 3:Generation & encoding bits

MATLAB code:

- generate random bits
- reshaping
- encode the set of data

```
Bits = randi([0 1],1,NumberInfoBits*1e3);
Bits = reshape(Bits,1000,NumberInfoBits);
Coded_Bits = zeros(NumberBitsPerFrame/log2(ModulationOrder),log2(ModulationOrder));
% Generating random bits each frame
for i = 1: NumberBitsPerFrame/log2(ModulationOrder)
    Coded_Bits(i,:)= mod(Bits(i,:)*Generator_matrix,2);
end
% Obtaining Symbol bits
SymbolBits=reshape(Coded_Bits,log2(ModulationOrder),NumberBitsPerFrame/log2(ModulationOrder))';
```

Part 4: Symbol mapper

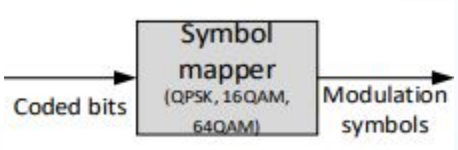


MATLAB code:

```
% Taking first k/2 bits to branch 1
SymbolBits_branch1=SymbolBits(:,[1:log2(ModulationOrder)/2]);
if( ModulationOrder == 16 || ModulationOrder ==64)
    % Transforming bits into intgers: bianry to decimal conversion
    SymbolIndex_branch1 = zeros(1000,1);
    for i = 1:1000
        for j = 1: NumberInfoBits
            SymbolIndex_branch1(i,1) =  SymbolIndex_branch1(i,1)+SymbolBits_branch1(i,j)*power(2,NumberInfoBits-j);
        end
    end
    % Symbol modulation using ASK modulation
    OutputModulator_branch1=2*(SymbolIndex_branch1)-1-(sqrt(ModulationOrder));

elseif ModulationOrder == 4
    OutputModulator_branch1 =2*(SymbolBits_branch1)-1;
end
```

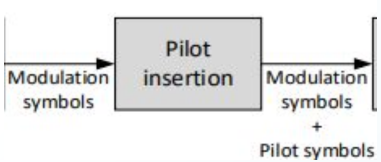
Part 4: Symbol mapper



- We divided the coded bits into 2 branches.
- We map a certain number of bits (according to the modulation type) into symbols.
- We create the transmitted signal according to the following equation.

```
TransmittedSignal= OutputModulator_branch1+1i*OutputModulator_branch2;
```

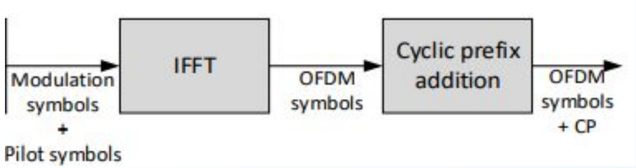
Part 5: Pilot



MATLAB code:

```
Test_Signal(1:1024) = 3+3*i;
Test_Signal_ifft = ifft(Test_Signal',1024).*sqrt(1024);
Test_Signal_CP = Test_Signal_ifft(1024-49:1024,:);
Test_Signal_OUT = [Test_Signal_CP;Test_Signal_ifft];
Test_Signal_OUT = reshape(Test_Signal_OUT',[],1);
```

Part 6: OFDM modulation



MATLAB code:

The modulated signals are passed to IFFT block to perform OFDM-based modulation.

The output is a sequence of OFDM symbols.

```
if Ch_Noise_Type == 1
    y_T = ifft(TransmittedSignal, 1024);
elseif Ch_Noise_Type == 2
    y_T = ifft(TransmittedSignal, 1024) .* sqrt(1024);
end
txcp = y_T(1024-49:1024,:);
txout = [txcp; y_T];
txout2 = reshape(txout',[],1); % Transmitted Sequence
```

Cyclic prefix insertion is performed to prevent Inter-Symbol Interference.

Part 7-A: Multipath fading (SISO)

MATLAB code:

```
%% Channel effect
L = 50;% no of paths
if Ch_Noise_Type == 2                  %Multpath
    if (Ch_MIMO_SETUP) == 1            %SISO
        h = ((randn(1,L)+1i*randn(1,L))/sqrt(2*L)); % equation of paths
        Recieved_Signal=conv(h,txout2);% Recieved signal from L-Path + noise
        RX_Test = conv(h,Test_Signal_OUT);
        N=sqrt(No/2)*(randn(length(Recieved_Signal),1)+1i*randn(length(Recieved_Signal),1));
        Recieved_Signal=Recieved_Signal+N;
        RX_Test2 = RX_Test+N;
```

We create the multipath fading effect (h) according to the following equation :

$$\boldsymbol{h} = \frac{1}{\sqrt{2L}}[randn(1,L) + 1i * randn(1,L)]$$

we convolute the transmitted signal with h, then we add the noise after that.

Part 7-B: Multipath fading (SIMO)

MATLAB code:

We perform the same function that was explained before but for two different channels and antennas.

```
elseif (Ch_MIMO_SETUP) == 2          %SIMO
    h1 = ((randn(1,L)+1i*randn(1,L))/sqrt(2*L)); % equation of paths
    Recieved_Signal1=conv(h1,txout2);% Recieved signal from L-Path + noise
    N1=sqrt(No/2)*(randn(length(Recieved_Signal1),1)+1i*randn(length(Recieved_Signal1),1));
    Recieved_Signal1=Recieved_Signal1+N1;
    RX1_Test = conv(h1,Test_Signal_OUT);
    RX1_Test_2 = RX1_Test + N1;

    h2 = ((randn(1,L)+1i*randn(1,L))/sqrt(2*L)); % equation of paths
    RX2_Test = conv(h2,Test_Signal_OUT);
    Recieved_Signal2=conv(h2, txout2);% Recieved signal from L-Path + noise
    N2=sqrt(No/2)*(randn(length(Recieved_Signal2),1)+1i*randn(length(Recieved_Signal2),1));
    Recieved_Signal2=Recieved_Signal2+N2;
    RX2_Test_2 = RX2_Test + N2;
end
```

Part 7-C: AWGN (SISO & SIMO)

MATLAB code:

Addition of additive gaussian noise to the signal with the same length of the signal.

```
elseif Ch_Noise_Type == 1            %AWGN
    if (Ch_MIMO_SETUP) == 1
        N=sqrt(No/2)*(randn(length(txout2),1)+1i*randn(length(txout2),1));
        Recieved_Signal=txout2+N; % Recieved signal from L-Path
    elseif(Ch_MIMO_SETUP) == 2
        N1=sqrt(No/2)*(randn(length(txout2),1)+1i*randn(length(txout2),1));
        N2=sqrt(No/2)*(randn(length(txout2),1)+1i*randn(length(txout2),1));
        Recieved_Signal1=txout2+N1; % Recieved signal from L-Path
        Recieved_Signal2=txout2+N2; % Recieved signal from L-Path
    end
end
```

Part 8-A: Multipath equalizer (SISO)

MATLAB code:

- Here we remove cyclic prefix
- get the estimated h from pilot signal
- Apply FFT on signal to get the received signal and reduce fading effect.

```
if Ch_Noise_Type == 2                  %Multipath
    Test_R_CP = conj(RX_Test2(51:end,:));
    h_est = (fft(Test_R_CP,1024)./sqrt(1024))./Test_Signal';
    RX_Remove_Cp = conj(Recieved_Signal(51:end,:));
    Rx_FFT = (fft(RX_Remove_Cp , 1024)./sqrt(1024))./h_est;
    ReceivedSignal = Rx_FFT(1:1000,:);
```

Part 8-B: Multipath equalizer (SIMO)

MATLAB code:

We perform the same function that was explained before but for two different channels and antennas.

```
if(Ch_MIMO_SETUP) == 2                    %SIMO equalizer
    if Ch_Noise_Type == 2

        RX1_Test_2_Cp1 = conj(RX1_Test_2(51:end,:)) ;
        h_est1 = (fft(RX1_Test_2_Cp1,1024)./sqrt(1024))./Test_Signal';
        RX_Remove_Cp1 = conj(Recieved_Signal1(51:end,:));
        Rx_FFT1 = (fft(RX_Remove_Cp1 , 1024)./sqrt(1024))./h_est1;
        Rx_FFT1 = Rx_FFT1(1:1000,:);
        ReceivedSignal1 = Rx_FFT1;

        RX2_Test_2_Cp2 = conj(RX2_Test_2(51:end,:)) ;
        h_est2 = (fft(RX2_Test_2_Cp2,1024)./sqrt(1024))./Test_Signal';
        RX_Remove_Cp2 = conj(Recieved_Signal2(51:end,:));
        Rx_FFT2 = (fft(RX_Remove_Cp2 , 1024)./sqrt(1024))./h_est2;
        Rx_FFT2 = Rx_FFT2(1:1000,:);
        ReceivedSignal2 = Rx_FFT2;

        ReceivedSignal = (ReceivedSignal1 + ReceivedSignal2)/2;
```

Part 8-C: AWGN (SISO & SIMO)

MATLAB code:

Siso

- Removing cyclic prefix
- Applying FFT

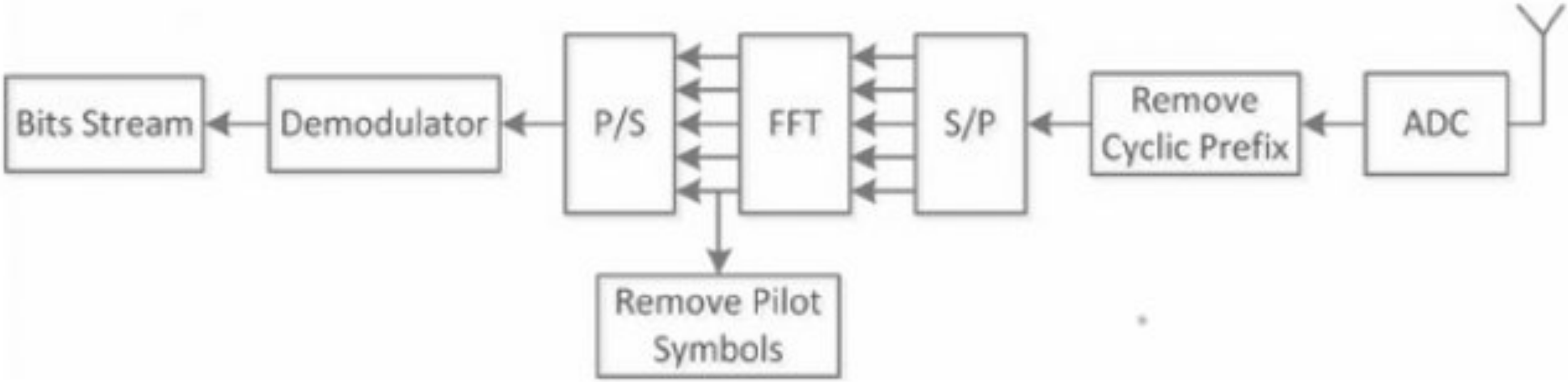
```
elseif Ch_Noise_Type == 1              %AWGN
    RX_Remove_Cp = conj(Recieved_Signal(51:end,:));
    Rx_FFT = fft(RX_Remove_Cp , 1024);
    Rx_FFT = Rx_FFT(1:1000,:);
    ReceivedSignal = Rx_FFT;
end
```

Simo

- Same as SISO except for 2 antennas at the receiver.

```
elseif Ch_Noise_Type == 1
    RX_Remove_Cp1 = conj(Recieved_Signal1(51:end,:));
    RX_Remove_Cp2 = conj(Recieved_Signal2(51:end,:));
    Rx_FFT1 = fft(RX_Remove_Cp1 , 1024);
    Rx_FFT2 = fft(RX_Remove_Cp2 , 1024);
    Rx_FFT1 = Rx_FFT1(1:1000,:);
    Rx_FFT2 = Rx_FFT2(1:1000,:);
    ReceivedSignal1 = Rx_FFT1;
    ReceivedSignal2 = Rx_FFT2;
    ReceivedSignal = (ReceivedSignal1 + ReceivedSignal2)/2;
end
```

2.Receiver



OFDM Tx block diagram

Part 1: Threshold detection (QAM)

MATLAB code:

```
%% Receiver Operation: Receiver Branch 1
% In-phase component is the real part of the signal
ReceivedSignal_branch1=real(ReceivedSignal);
if( ModulationOrder == 16 || ModulationOrder ==64)
    % Receiver operation is threshold operation
    % Threshold is {..., -4, -2, 0, 2, 4, ...}
    for threshold=-sqrt(ModulationOrder)+2:2:sqrt(ModulationOrder)-4
        DetectedSymbols_branch1((ReceivedSignal_branch1>threshold) &(ReceivedSignal_branch1<=threshold+2))=threshold+1;
    end
    % Detecting edge symbols
    DetectedSymbols_branch1(ReceivedSignal_branch1>sqrt(ModulationOrder)-2)=sqrt(ModulationOrder)-1;
    DetectedSymbols_branch1(ReceivedSignal_branch1<=-sqrt(ModulationOrder)+2)=-sqrt(ModulationOrder)+1;
    % Transform detected symbols into symbol index
    ReceivedSymbolIndex_branch1=((DetectedSymbols_branch1+sqrt(ModulationOrder)+1)/2-1)+1;
    for i=1:length(ReceivedSymbolIndex_branch1)
        if ReceivedSymbolIndex_branch1(i) > sqrt(ModulationOrder)-1
            ReceivedSymbolIndex_branch1(i) = sqrt(ModulationOrder)-1;
        elseif ReceivedSymbolIndex_branch1(i) < -sqrt(ModulationOrder)-1
            ReceivedSymbolIndex_branch1(i) = -sqrt(ModulationOrder)-1;
        end
    end
    X=dec2bin(ReceivedSymbolIndex_branch1,NumberInfoBits).'-'0';
    DetectedBits_branch1 = reshape(X,log2(ModulationOrder)/2,[])';
```

Part 1: Threshold detection (QAM)

- We divide the the complex received signal back to 2 branches.
- We set threshold values.
- We compare them with the received signal.
- We split each symbol back to its bits (inverse operation of mapper).

Part 2: Threshold detection (QPSK)

MATLAB code:

- Threshold value is fixed at 0.
- We compare the received signal with the threshold.
- If the received bit is greater the receiver detects the value as 1.
- If it equals the threshold the receiver detects the value as 0.

```
elseif ModulationOrder == 4
    DetectedBits_branch1 =zeros(1000,1);
    thershold =0;
    for i=1:1000
        if(ReceivedSignal_branch1(i,1)>thershold)
            DetectedBits_branch1(i,1)=1;
        else
            DetectedBits_branch1(i,1)=0;
        end
    end
end
```

Part 3: Channel decoding

MATLAB code:

```
%% channel decoding
if( ModulationOrder == 16 || ModulationOrder ==64 || ModulationOrder  == 4)
    a = mod(dec2bin(0:(power(2,NumberInfoBits)-1)),2);
    codewords = mod(a*Generator_matrix,2);
    index=0;
    Decoded_ReceivedBits= [];
    no_col_ReceivedBits= size(ReceivedBits);
    for k = 1:no_col_ReceivedBits(1)
        min_E = no_col_ReceivedBits(2);
        for j = 1:power(2,NumberInfoBits)
            Error =0;
            for i = 1:log2(ModulationOrder)
                if ReceivedBits(k,i)~= codewords(j,i)
                    Error = Error+1;
                end
            end
            if Error < min_E
                min_E= Error;
                index = j;
            end
        end
        Decoded_ReceivedBits(k,:) = codewords(index,1:NumberInfoBits);
    end
```

Part 4: BER Calculations

MATLAB code:

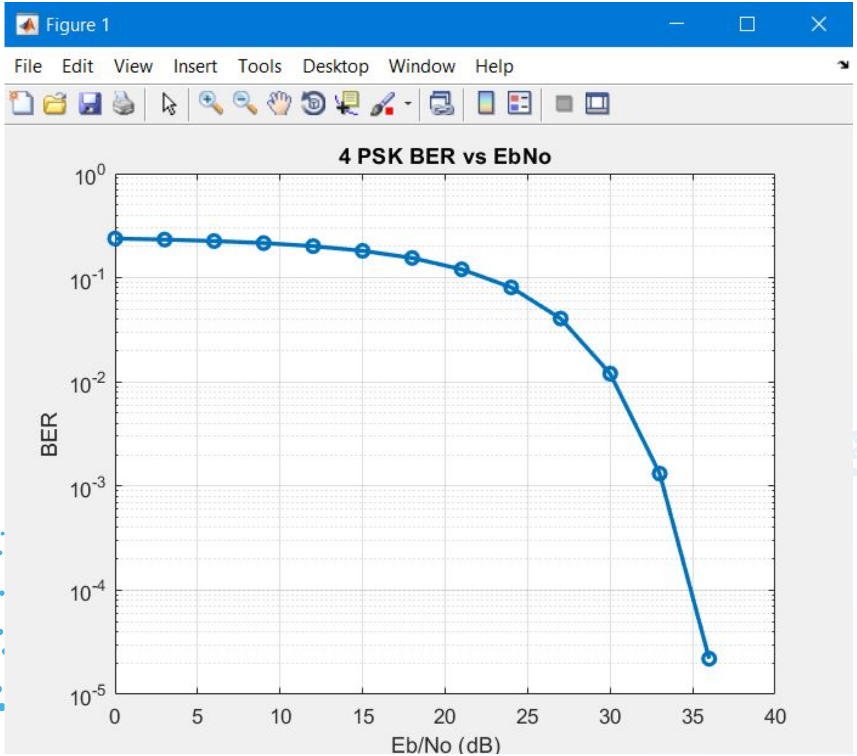
- Converting parallel data to serial.
- Calculate BER.

```
%% Serializing output
 if( ModulationOrder == 16 || ModulationOrder ==64 || ModulationOrder == 4)
Serial_Decoded_ReceivedBits = reshape(Decoded_ReceivedBits,1,NumberInfoBits*1e3);
Serial_input_bits = reshape(Bits,1,[]);
 end
%% BER calculation
if( ModulationOrder == 16 || ModulationOrder ==64 || ModulationOrder ==4 )
    prob_error_frame=sum(xor(Serial_input_bits,Serial_Decoded_ReceivedBits))/NumberBitsPerFrame;
end
sum_prob_error=sum_prob_error+prob_error_frame;
end
 BER = [BER sum_prob_error/NumberFramesPerSNR];
 if sum(sum_prob_error)==0
     break
 end
end
```

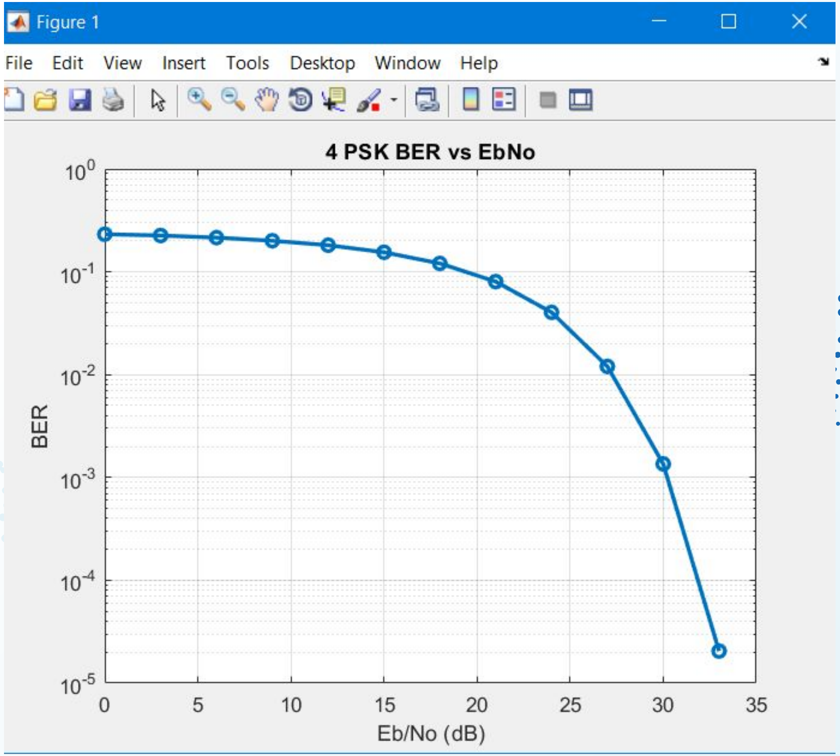
3.Results

4-QAM(QPSK)AWGN Channel :

SISO

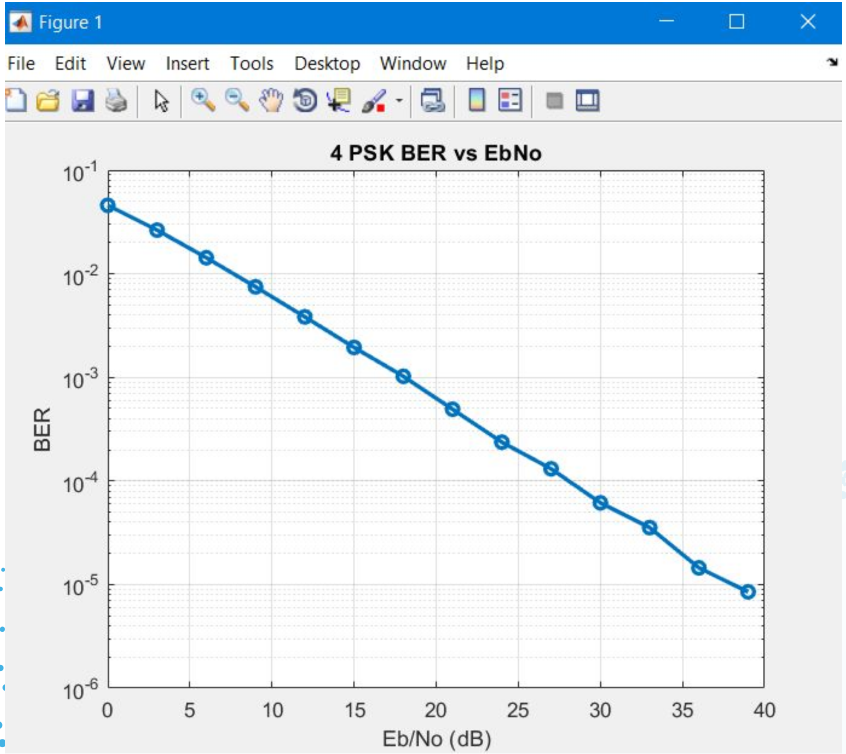


SIMO

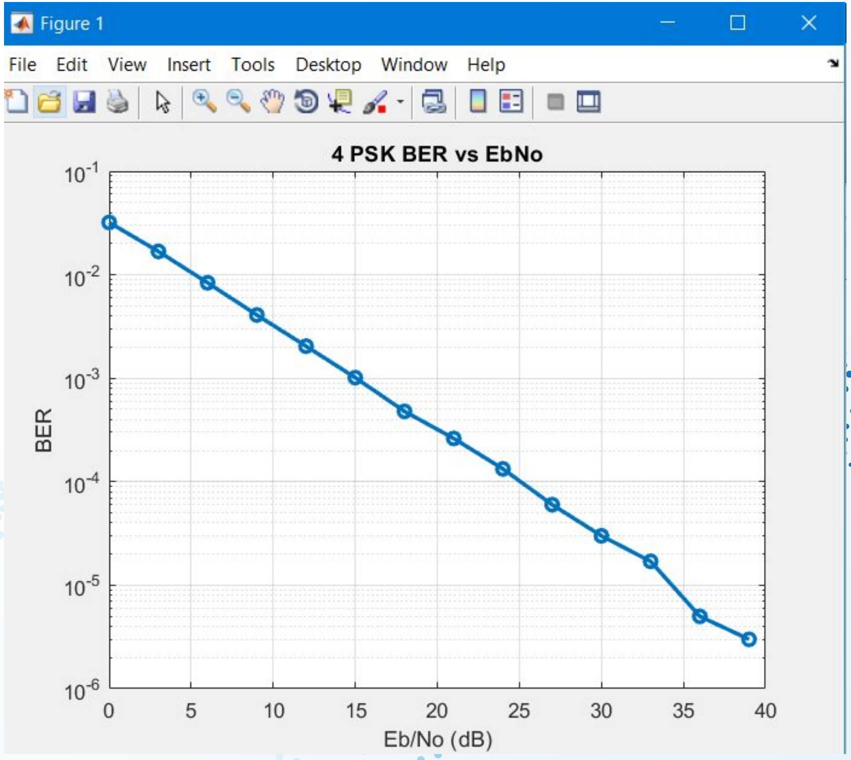


4-QAM(QPSK)Multipath Fading Channel :

SISO

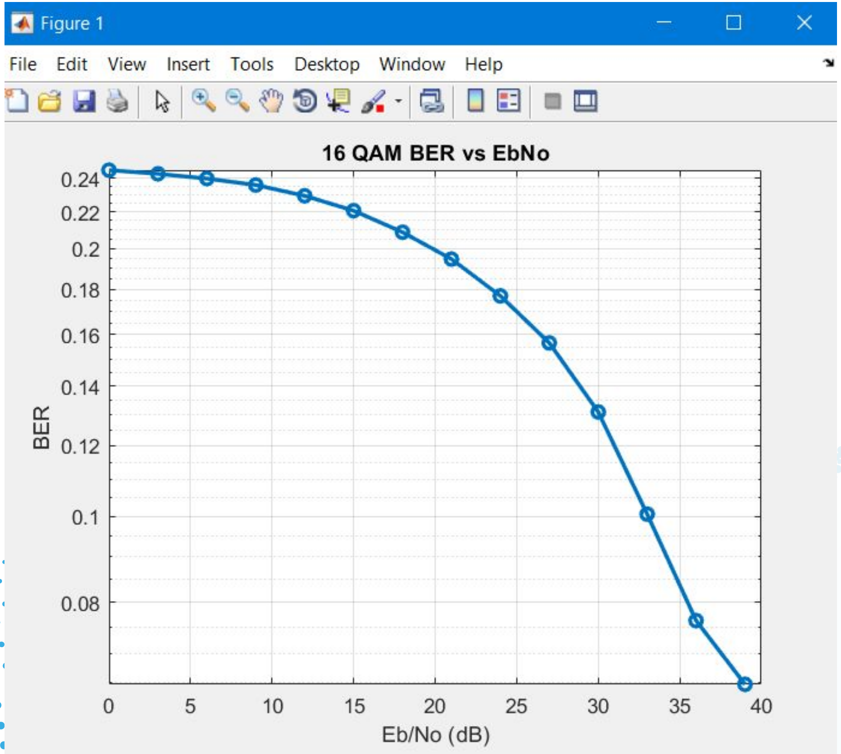


SIMO

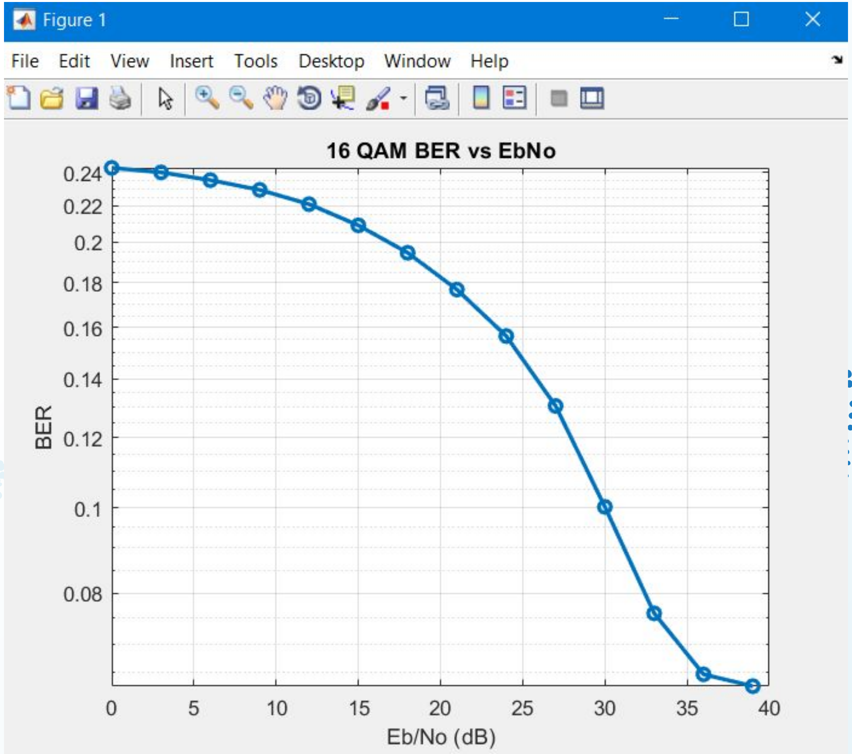


16-QAM AWGN Channel :

SISO

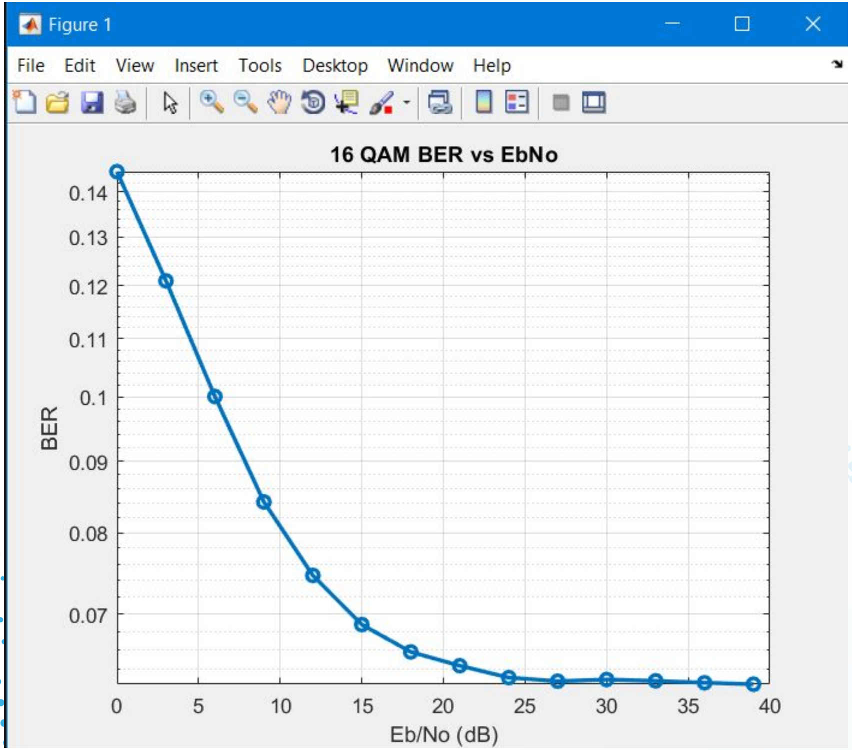


SIMO

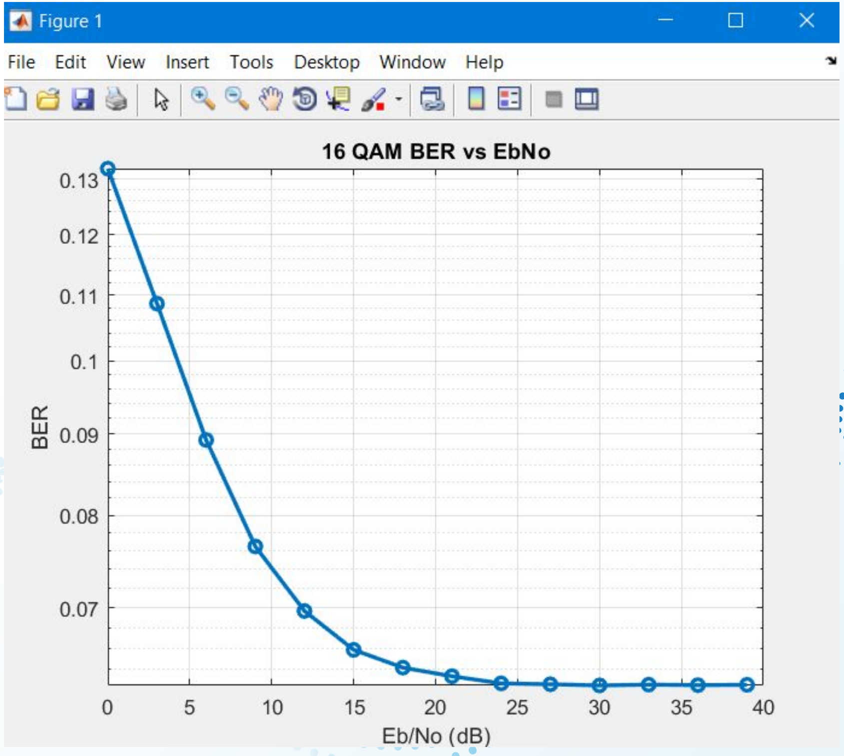


16-QAM Multipath Fading Channel :

SISO

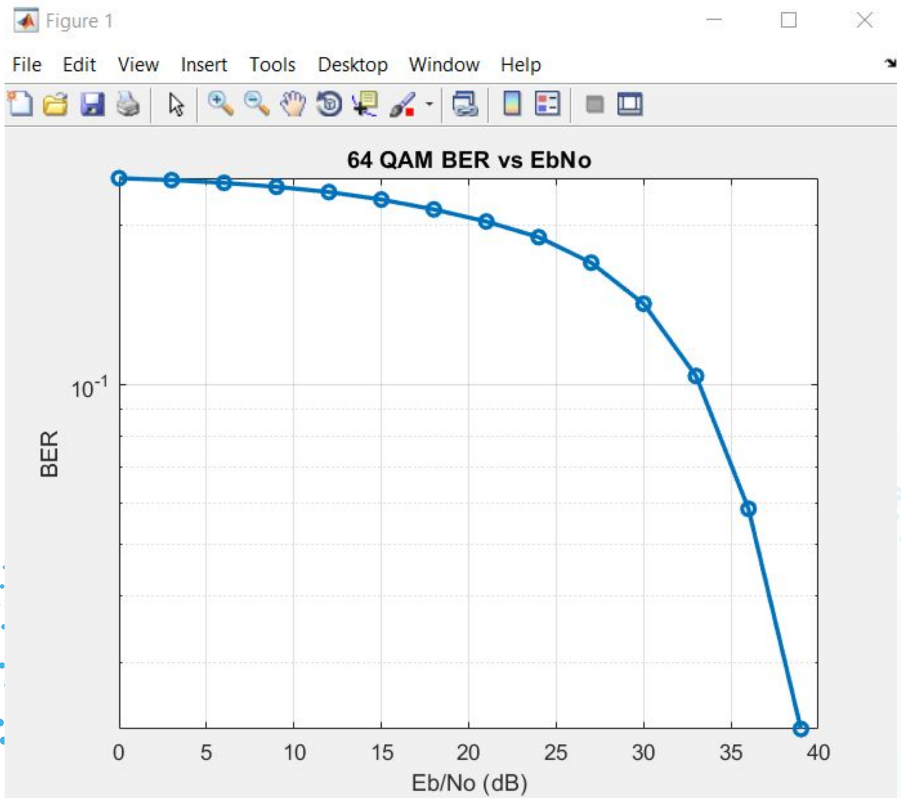


SIMO

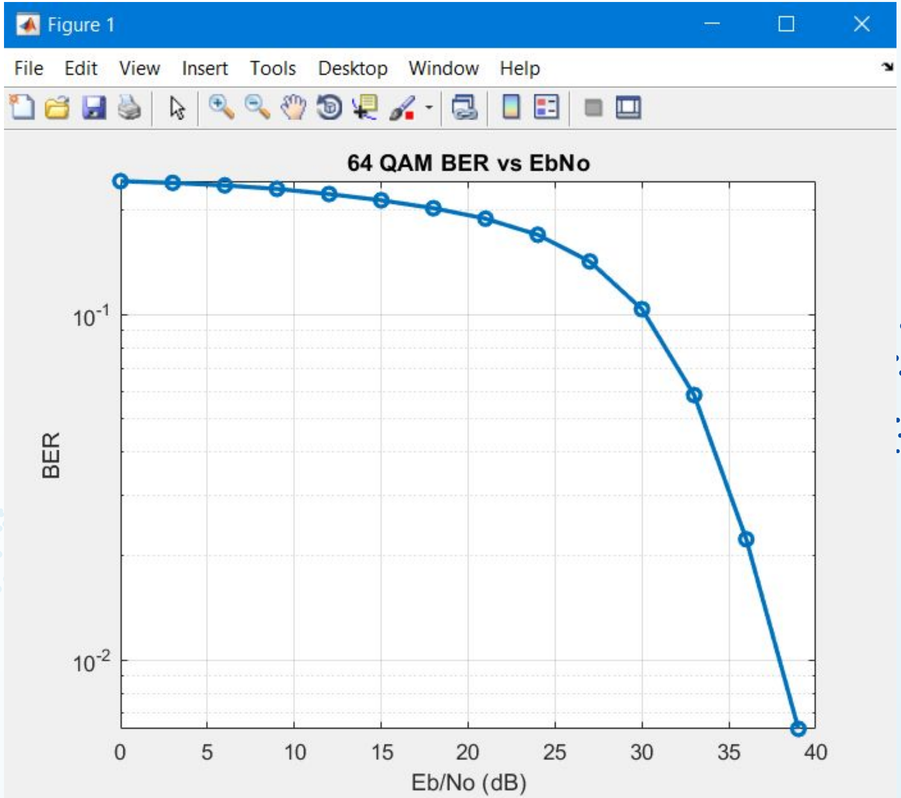


64-QAM AWGN Channel :

SISO

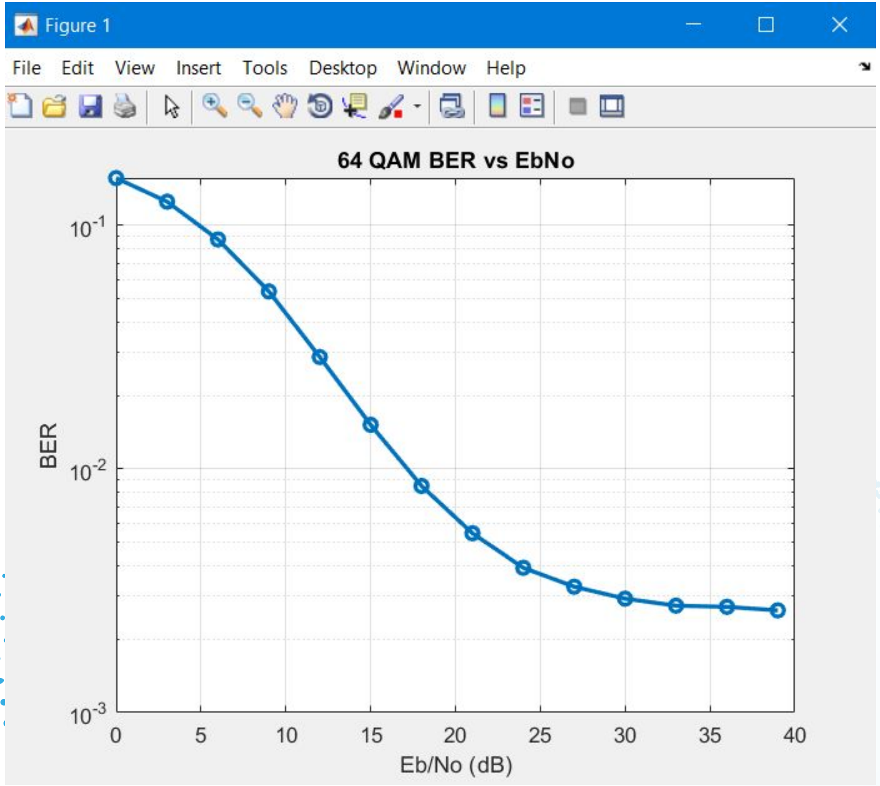


SIMO

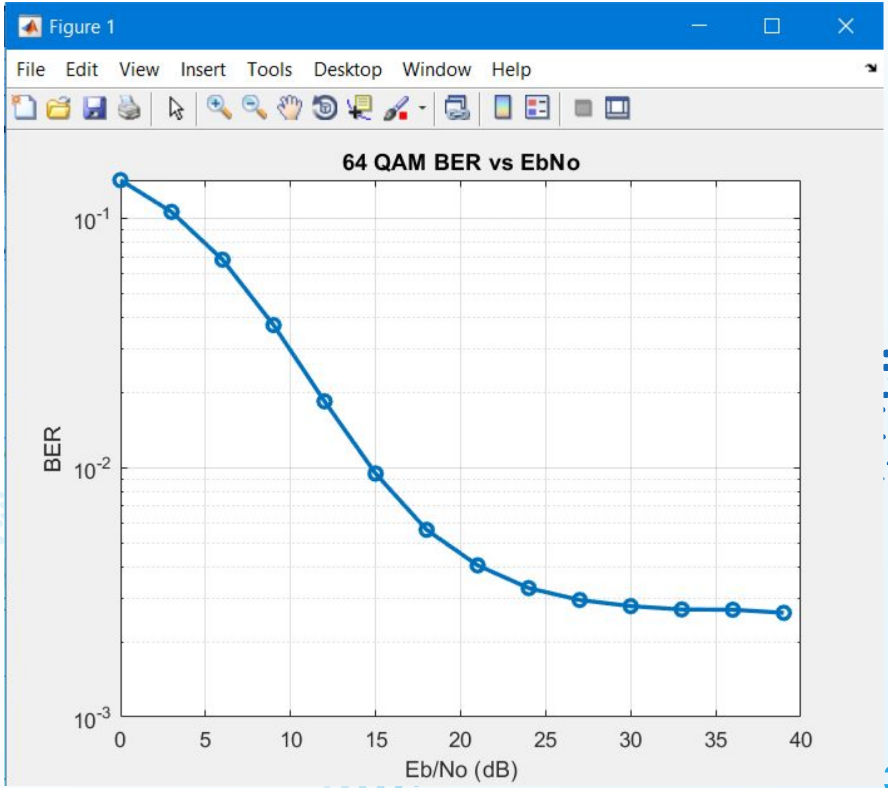


64-QAM Multipath Fading Channel :

SISO



SIMO



Thanks!