Mohadeseh Azari
810196404

wireless project
1400/05

Transmitter

```
n = 10^7; % Number of Bits
V = randi([0, 1], [1, n]);
m = n/2;
QPSK = zeros(1,m);
for i=1:m
    A = [V(2*i-1),V(2*i)];
    b = num2str(A);
    b(b==' ')=[];
    switch b
        case '00'
            QPSK(i)=0;
        case '01'
            QPSK(i)=1;
        case '10'
            QPSK(i)=2;
        case '11'
            QPSK(i)=3;
    end
end
```

In this part first, the vector V is the random vector with length 10^7.

Then due to QPSK modulation we must choose 2 bits of V and convert it into decimal which is QPSK vector here.

```
nc = 400; %number of carrier
sfc = ceil(2^13/nc);
dimension = ceil(m/(nc*sfc));
Symbol = [QPSK, zeros (1, sfc*nc*dimension - length(QPSK))]; %padding zero
MatRix=zeros (sfc+1, nc, dimension);
for k=1: dimension
    for i=1:nc
        for j=1:21
            MatRix (j+1, i, k) = Symbol((k-1) *8400+(i-1) *21+j);
        end
    end
end
A= [0 1 2 3];
for k=1: dimension
    for i=1:nc
            MatRix (1, i, k) = randsample(A,1); %reference row
    end
end
```

Here in this part after creating QPSK vector we must append zero at the end of this matrix in order to reach the size of each frame.

*If size of each frame is 22*400 (21+1) so for (10^7 / 2) symbol, we need 596 data frames.*

*Now we have a 22*400 matrix with 596 dimension.*

```
for k=1:dimension
    for n=1:nc
        for m=1:21
            MatRix(m+1,n,k) = MatRix(m+1,n,k) + MatRix(m,n,k); %adding
numbers with lower cell
        end
    end
end
for k=1:dimension
    for n=1:nc
        for m=1:21
            MatRix(m+1,n,k) = mod(MatRix(m+1,n,k),4); %calculate modulo 4
        end
    end
end
for k=1:dimension
    for n=1:nc
        for m=1:22
            MatRix(m,n,k) = deg2rad(90 * MatRix(m,n,k));
            MatRix(m,n,k) = exp(1i*MatRix(m,n,k)); %converting into e^jφ
        end
    end
end
```

DPSK modulation has 3 level:

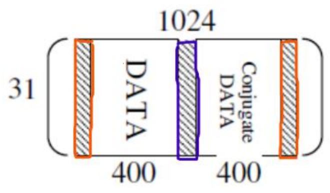
At first we have to add each cell with the upper cell for example cell (2) = cell (2) + cell (1).

Then we calculate the residue of each cell with respect to 4. QPSK

After that convert each cell into [0,90,180,270]

```
for k=1:dimension
    for i=1:nc
        for j=1:22
            MatRix_conj(j,i,k) = conj(MatRix(j,i,k)); %conjugate
        end
    end
end
between = zeros(22,76);
trailer = zeros(22,74);
C1=cat(2,trailer,MatRix);
C2=cat(2,C1,between);
C3=cat(2,C2,MatRix_conj);
Frame=cat(2,C3,trailor);
```

In this part, the total frame is created



The middle matrix is called between and the other matrices are trailers.

```
for k=1:dimension
    Frame_IFFT(:,:,k) = real(ifft(Frame(:,:,k)'))';
end
CP = Frame_IFFT(:,(769:1024),:); %Cyclic Prefix
Final_Frame = cat(2,CP,Frame_IFFT);
serial_size = 22 * 1280 ;
for k=1:dimension
    for i=1:1280
        for j=1:22
            serial_Frame(1,22*(i-1)+j,k) = Final_Frame(j,i,k); %convert into
serial frame
        end
    end
end
```

In the last level of transmitter at first, calculate the IFFT of the matrix and then add 25% cyclic prefix and finally convert matrix frame into serial frame. Ready to be transmitted.

Number of frame is 596 frames.

Channel

the communication channel is an AWGN channel with clipping. What is clipping?

Clipping, limits the max peak of a signal at one specific level.

Max(abs(signal)) – clipping = peak clipped in dB

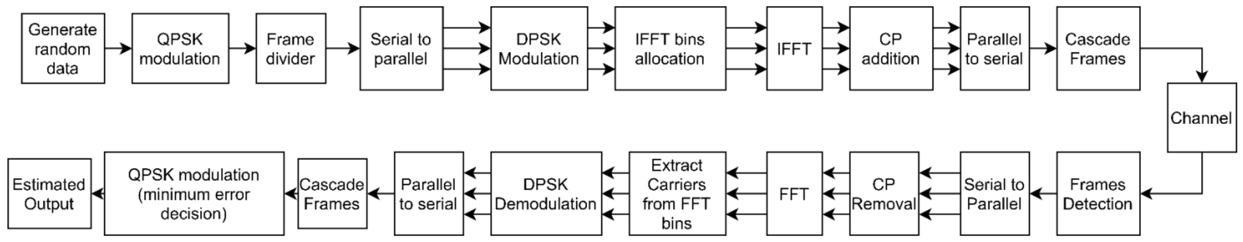
```
function [ output_serial ] = Channel( input_serial,clipping,SNR )
%This is the function for the communication Channel
%clipping 3dB
peak_clipped = (10^(0-(clipping/20)))*max(abs(input_serial));
%σ_s
s = var(input_serial);
```

```
%length of signal
N = length(input_serial);
output_serial = input_serial;
%Clipping
limit = abs(output_serial(:))>=peak_clipped;
output_serial(limit)=
peak_clipped.*output_serial(limit)./abs(output_serial(limit));
%converting from dB into linear
SNR_ = 10^(SNR/10);
%calculate the sigma of noise signal
```
%$\sigma = \frac{\sigma_S}{\sqrt{SNR}}$
```
sigma = sqrt(s/SNR_);
noise = randn(1,N) * sigma;
%adding noise to signal
output_serial = output_serial + noise;
end
```

First the input signal must be clipped, so we have to calculate the peak clipped of that signal. After that all the max(abs) of the signal are at the same level.

After that the random noise will be created and added to the whole signal.

The output signal is both clipped and noisy.



Transmitter:

Generate random data: At first the 10^7 random bit is generated using rand function

QPSK modulation: each two bits are collected to form QPSK symbol, {00,01,10,11}

*Frame divider: a 21*400 matrix is generated for data frame. 21 is sfc and 400 is nc*

*Serial to parallel: all the data which were converted into decimal form {0,1,2,3} are stored in the 21*400 matrix*

DPSK modulation: each cell of the matrix must be added to the lower cell and then each cell is replaced with its remaining on 4

IFFT bins allocation: the conjugate of the data is added to the matrix

IFFT: IFFT of data is calculated and its real part must be stored

CP addition: cyclic prefix must be added, 25% of the data is added

Parallel to serial: the matrix is converted into serial frames ready to be transmitted

Cascade frame: each frame will be sent consecutively

Channel: data is sent through an AWGN channel

Receiver: receiver is exactly like transmitter in reverse

- *The only important consideration here is that each data and its conjugate must be in the form of{l,1024-l+2}. In this case no data will be lost.*

Receiver

As explained before receiver is somehow like transmitter in reverse.

```
clipping = 3;
SNR = 20;
Length_frame = size(serial_Frame);
output_serial = serial_Frame;
for i=1:Length_frame(3)
    output_serial(:,:,i)  = Channel( serial_Frame(:,:,i),clipping,SNR );
end
```

Clipping and SNR are provided so they are known, the channel function simply add noise and clip the signal so the input of out receiver is output_serial

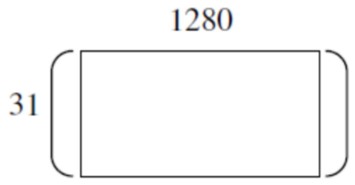
---

```
for k=1:dimension
    for i=1:1280
        for j=1:22
            parallel_Frame(j,i,k) = output_serial(1,(i-1)*22+j,k); %convert
into parallel frame
        end
    end
end
serial_size = 22 * 1280 ;
parallel_Frame(:,(1025:1280),:)=[]; %Cyclic Prefix

for k=1:dimension
    Frame_FFT(:,:,k) = fft(parallel_Frame(:,:,k)); %fft
end
```

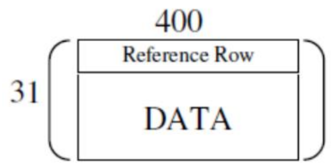
In receiver every step is reverse with respect to transmitter. First, serial data is converted into parallel form and the cyclic prefix will be removed from the matrix. And finally FFT of the signal is replaced with the old data.

```
Data = Frame_FFT(:,(75:474),:);
for k=1:dimension
    for n=1:nc
```

```
        for m=1:22
            MATRIX(m,n,k) = angle(Data(m,n,k)); %calculating phase of each
element
            MATRIX(m,n,k) = rad2deg(MATRIX(m,n,k))+ 180 ;
        end
    end
end
for k=1:dimension
    for n=1:nc
        for m=1:22
            MATRIX(m,n,k) = MATRIX(m,n,k)./ 90 ; %translating into symbol
        end
    end
end
for k=1:dimension
    for n=1:nc
        for m=21:1
            MATRIX(m+1,n,k) = MATRIX(m+1,n,k) - MATRIX(m,n,k) ; %Subtracting
numbers with upper cell
        end
    end
end
MATRIX(1,:,:)=[];
MATRIX = round(MATRIX);
MATRIX = mod(MATRIX,4);

for k=1:dimension
    for n=1:nc
        for m=1:21
            Symbol_receive(m+(n-1)*21+(k-1)*8400) = MATRIX(m,n,k) ;
%Subtracting numbers with upper cell
        end
    end
end
```



The lower matrix is extracted from the upper matrix in this state.



And then the phase of each element is calculated. Due to the fact that angle function returns the phase in radius, a radius to degree function is needed to translate angles in [0,360] interval.

After that each phase is divided to 90° *in order to translate symbols into {0,1,2,3}*

If the numbers exceeded this numbers, calculate the remainder of 4

And at the final stage each element is subtracted from its upper element.

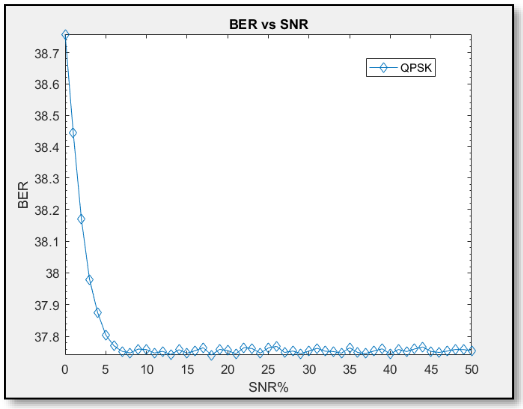
rounding is needed at the end.

The Symbol_receive matrix is the final matrix

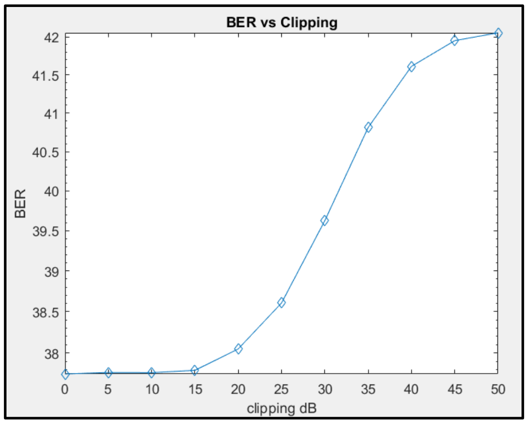
ERROR Rate

```
>> Error
Bit Error Rate (BER) = 0.375181%
>>
```

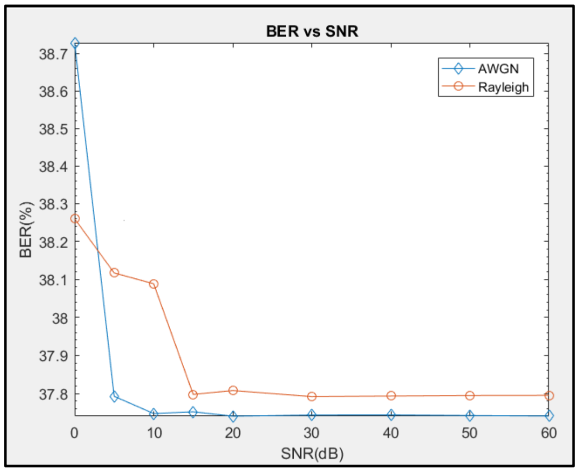
```
errors = find(Symbol_receive(1:length(QPSK))~=QPSK);
% Bit Error Rate
fprintf('Bit Error Rate (BER) = %f%%\n',length(errors)/length(QPSK))
```



a mistake has been made here, SNR is in dB and BER is in %



BER vs clipping, as we can see changing clipping wont effect Bit Error Rate



differences between AWGN and Rayleigh channel

```
function [ Grid ] = MMSE( rxWaveform )
L = length(rxWaveform);
h = raylrnd(1,1,L);
H = fourier(h);
sigma = var(rxWaveform);
Equalizer = zeros(1,L);
for m=1:L
    Equalizer(m) = conj(H(m)) / (abs(H(m))^2 + sigma^2);
end
Grid = Equalizer .* rxWaveform;
End
```

The MMSE equalizer is added before the receiver, unfortunately Matlab crashed when showing the plot and I couldn't save the plot. You can see the plot by running BER_vs_SNR.m