22-5-2022

Wireless Communications

Problem 4

Name: 乌斯曼 Usman

Teacher Name: Du Bing

Student ID: M202161031

计算机与通信工程学院

University of Science and Technology Beijing

OFDM:

Orthogonal frequency-division multiplexing (OFDM) is a multi-carrier modulation system where data is transmitted as a combination of orthogonal narrowband signals known as subcarriers. OFDM builds upon single carrier modulation such as QAM and can transmit at similar data rates. However, OFDM is more robust to frequency selective fading and simplifies equalization at the receiver. OFDM is a foundational scheme found in many common wireless communications standards such as WIFI, LTE, and 5G. You can use MATLAB® and Simulink® to configure and generate OFDM waveforms, adhering to these standards to simulate and test a physical layer model of your wireless communications system.

code

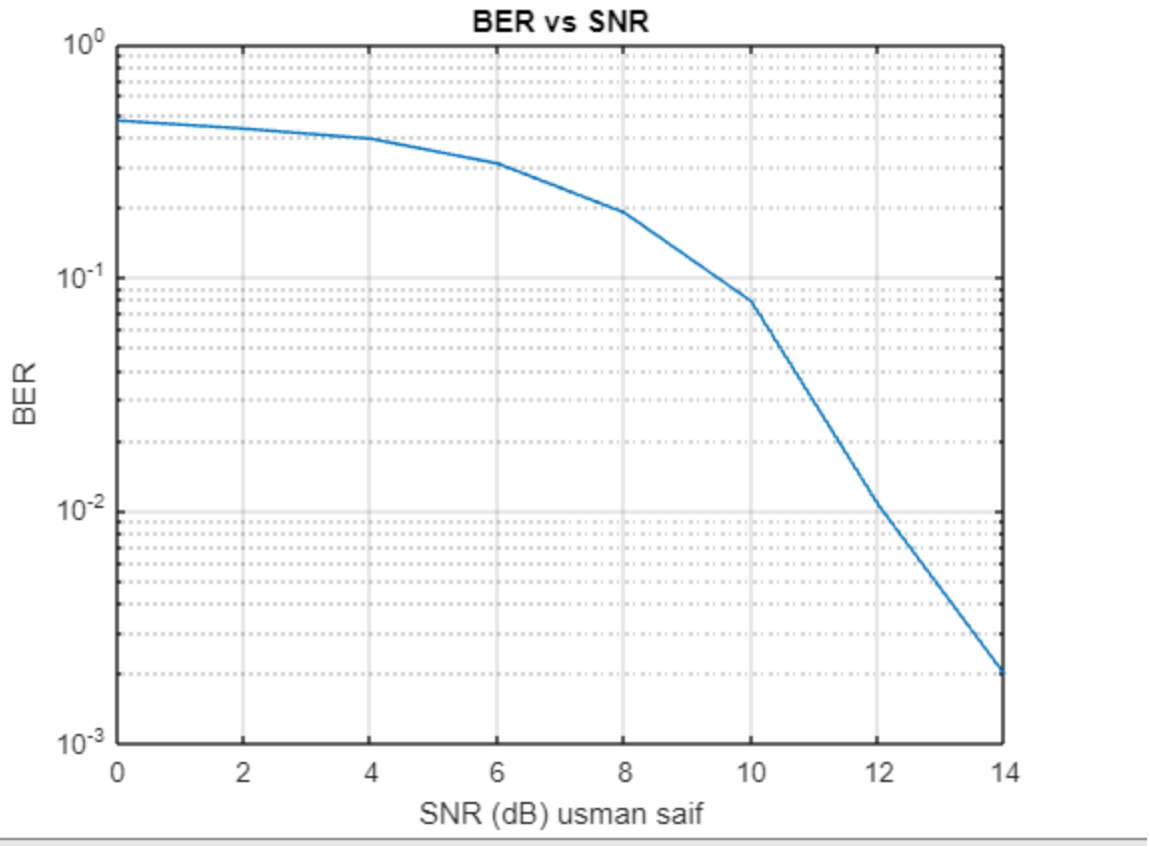
```
% OFDM Code
% Author: usman saif,
%
% No.of Carriers: 64
% coding used: Convolutional coding
% Single frame size: 96 bits
% Total no. of Frames: 100
% Modulation: 16-QAM
% No. of Pilots: 4
% Cylic Extension: 25%(16)
close all
clear all
clc
%%
% Generating and coding data
 t_data=randi([0,1],9600,1)';
x=1;
si=1; %for BER rows
%%
for d=1:100;
data=t_data(x:x+95);
x=x+96;
k=3;
n=6;
s1=size(data,2);  % Size of input matrix
j=s1/k;
%%
% Convolutionally encoding data
constlen=7;
codegen = [171 133];    % Polynomial
trellis = poly2trellis(constlen, codegen);
codedata = convenc(data, trellis);
%%
%Interleaving coded data
s2=size(codedata,2);
j=s2/4;
matrix=reshape(codedata,j,4);
intlvddata = matintrlv(matrix',2,2)'; % Interleave.
```

```
intlvddata=intlvddata';
%%
% Binary to decimal conversion
dec=bi2de(intlvddata','left-msb');
%%
%16-QAM Modulation
M=16;
y = qammod(dec,M);
% scatterplot(y);
%%
% Pilot insertion
lendata=length(y);
pilt=3+3j;
nofpits=4;
k=1;
for i=(1:13:52)

    pilt_data1(i)=pilt;
    for j=(i+1:i+12);
        pilt_data1(j)=y(k);
        k=k+1;
    end
end
pilt_data1=pilt_data1';   % size of pilt_data =52
pilt_data(1:52)=pilt_data1(1:52);    % upsizing to 64
pilt_data(13:64)=pilt_data1(1:52);   % upsizing to 64
for i=1:52

    pilt_data(i+6)=pilt_data1(i);

end
%%
% IFFT
ifft_sig=ifft(pilt_data',64);
%%
% Adding Cyclic Extension
cext_data=zeros(80,1);
cext_data(1:16)=ifft_sig(49:64);
for i=1:64

    cext_data(i+16)=ifft_sig(i);

end
%%
% Channel
 % SNR
 o=1;
for snr=0:2:50
ofdm_sig=awgn(cext_data,snr,'measured'); % Adding white Gaussian Noise
% figure;
% index=1:80;
% plot(index,cext_data,'b',index,ofdm_sig,'r'); %plot both signals
% legend('Original Signal to be Transmitted','Signal with AWGN');
%%
%                   RECEIVER
```

```
%%
%Removing Cyclic Extension
for i=1:64

    rxed_sig(i)=ofdm_sig(i+16);

end
%%
% FFT
ff_sig=fft(rxed_sig,64);
%%
% Pilot Synch%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:52

    synched_sig1(i)=ff_sig(i+6);

end
k=1;
for i=(1:13:52)

    for j=(i+1:i+12);
        synched_sig(k)=synched_sig1(j);
        k=k+1;
    end
end
% scatterplot(synched_sig)
%%
% Demodulation
dem_data= qamdemod(synched_sig,16);
%%
% Decimal to binary conversion
bin=de2bi(dem_data','left-msb');
bin=bin';
%%
% De-Interleaving
deintlvddata = matdeintrlv(bin,2,2); % De-Interleave
deintlvddata=deintlvddata';
deintlvddata=deintlvddata(:)';
%%
%Decoding data
n=6;
k=3;
decodedata =vitdec(deintlvddata,trellis,5,'trunc','hard');  % decoding datausing
veterbi decoder
rxed_data=decodedata;
%%
% Calculating BER
rxed_data=rxed_data(:)';
errors=0;
c=xor(data,rxed_data);
errors=nnz(c);
% for i=1:length(data)
%
%
%     if rxed_data(i)~=data(i);
```

```
%         errors=errors+1;
%
%     end
% end
BER(si,o)=errors/length(data);
o=o+1;
 end % SNR loop ends here
 si=si+1;
end % main data loop
%%
% Time averaging for optimum results
for col=1:25;        %%%change if SNR loop Changed
    ber(1,col)=0;
for row=1:100;


        ber(1,col)=ber(1,col)+BER(row,col);
    end
end
ber=ber./100;
%%
figure
i=0:2:48;
semilogy(i,ber);
title('BER vs SNR');
ylabel('BER');
xlabel('SNR (dB) usman saif');

grid on
```

OFDM Output:



Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states called the Viterbi path that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM). The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, speech synthesis, diarization, keyword spotting, computational linguistics, and bioinformatics. For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal.

Code:

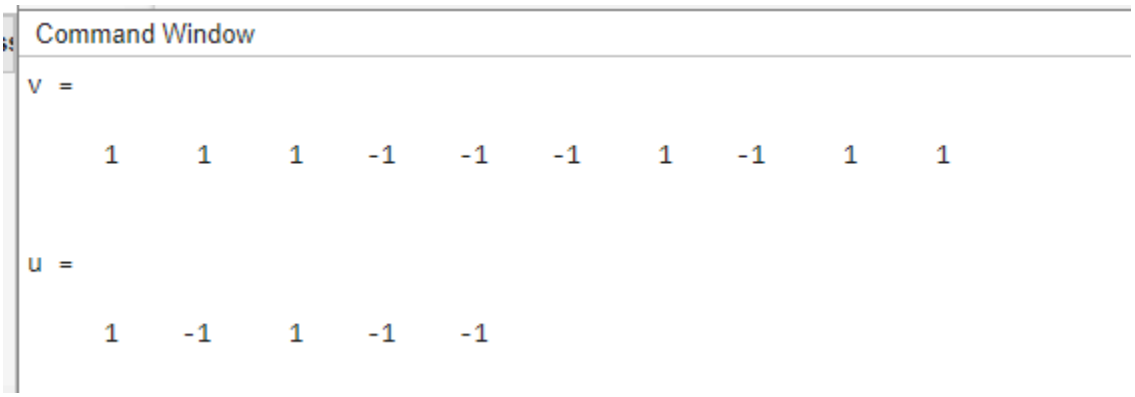
```
%% Viterbi Algorithm

% usman saif
% Telecommunications Engineering,
%% Matriz H
%            S0 S1 S2... Sj
%        S0|                |
%        S1|                |
%        S2|                |
%   H =   .|                |
```

```
%        Si|                |
%          |                |
clc; clearvars; close all;
%% Input Parameters
r = [1.2 0.9 -1.1 0.5 -0.1 -0.8 0.3 0.4 1.2 -0.1];   % Received sequence. The
original sequence without noise was v = [1 1 -1 1 -1 -1 -1 1 1 1] created from u = [1
0 1]. Two zeros must be added to u take the trellis to the So (00) state.
%r = [0.7 0.4 0.7 0.6 0.3 0.4 1.1 -0.3 0.4 0.4];     % Received sequence. The
original sequence without noise was v = [1 1 1 -1 -1 1 1 -1 1 1] created from u = [1
1 1]. Two zeros must be added to u take the trellis to the So (00) state.
x = 0;
H(:,:,1) = [1 x 1 x; 1 x 1 x; x 1 x 1; x 1 x 1];   % State relation matrix (M is a
square matrix) - Put a 1 if the state Si is connected with the state Sj (Example for
code (2,1,2))
H(:,:,2) = [0 x 1 x; 0 x 1 x; x 0 x 1; x 0 x 1];   % Matrix of relation of states
(inputs) - Put the input that takes from the state Si to the state Sj (Example for
code (2,1,2))
H(:,:,3) = [0 x 3 x; 3 x 0 x; x 2 x 2; x 2 x 1];   % Matrix of relation of states
(outputs) - Write in decimal - Put the output when I go from the state Si to the
state Sj (Example for code (2,1,2))
k = 1;                                              % Number of decoder entries
n = 2;                                              % Number of decoder outputs
m = 2;                                              % Number of decoder memories
l = 3;                                              % Information Sequence length to
be decoded (without including the 0s that will bring back the memories to the state
0)
mode = 'pm1';                                       % If we put 'pm1' the algorithm
works with +1 and -1, if we put something else it works with 0 and +1
%% Calculated parameters
nro_estados = length(H);                            %Número de estados
L = l/k;                                            %Número de blocos nos quais devo
dividir a sequência
I = L + m + 1;                                      %Instantes de tempo necesários na
treliça
interv = I - 1;                                     %Intervalos na treliça (O número
de intervalos é o número de instantes menos 1)
%% Branches and their metrics (Determining the branches with the lowest metric)
aux = 1e6;
metricas = aux*ones(nro_estados,nro_estados,interv);
estado = [1 zeros(1,nro_estados-1)];           % We put a 1 in state 0 to start the
lattice in that state.
metrica_caminho = zeros(1,nro_estados);
for tempo = 1:interv
    for kk = 1:nro_estados
        if estado(kk) == 1
            for kkk = 1:nro_estados
                if (H(kk,kkk,1) == 1 && tempo <= interv - m) || (H(kk,kkk,1) == 1 &&
H(kk,kkk,2) == 0 && tempo > interv - m)
                    z = de2bi(H(kk,kkk,3),n,'left-msb');
                    if strcmp(mode,'pm1')
                       z(z==0) = -1;
                    end
                    metricas(kk,kkk,tempo) = metrica_caminho(kk) + sum((r(n*tempo-
n+1:n*tempo) - z).^2);
                end
```

```
            end
        end
    end
    metrica_caminho = min(metricas(:,:,tempo));
    estado = zeros(1,nro_estados);
    estado(metrica_caminho < 0.99*aux) = 1;
end
%% Determination of the branches that generated the path with the lowest metric
estados = zeros(interv,2);
[estados(interv,1),estados(interv,2)] = find(metricas(:,:,tempo) ==
min(metrica_caminho)); % [Initial state Final state] of the last instante of the
trellis
for tempo = interv-1:-1:1
    estados(tempo,2) = estados(tempo+1,1);                          % The initial
state of state i + 1 is the final state of state i-1
    [~,estados(tempo,1)] = min(metricas(:,estados(tempo,2),tempo));   % As we already
know the final state, we need to know the initial state, for that, we fix the column
of the final state and look for the position of the lowest value (this position is
the initial state)
end
v = zeros(1,length(interv));
u = zeros(1,length(interv));
for tempo = 1:interv
   v(1,tempo) = H(estados(tempo,1),estados(tempo,2),3); % Decoded vector recovered
with the Viterbi algorithm in decimal numbers
   u(1,tempo) = H(estados(tempo,1),estados(tempo,2),2); % Information vector
recovered with the Viterbi algorithm
end
v = de2bi(v,n,'left-msb');
[fil,col] = size(v);
v = reshape(v.',[1,fil*col]);
u = de2bi(u,k,'left-msb');
[fil,col] = size(u);
u = reshape(u.',[1,fil*col]);
if strcmp(mode,'pm1')
    v(v==0) = -1
    u(u==0) = -1
end
```

Output:

22-5-2022



END