Project 2: OFDM MATLAB Simulation

Zhuohao Lee ✉*

April 27, 2022

Contents

*edith_lzh@sjtu.edu.cn | 519021911248 | Project Repo

1 Introduction

OFDM is a special multi-carrier transmission scheme, which can be regarded as a modulation technology or as a multiplexing technology.

Simply put: OFDM is a multi-carrier transmission method, which divides the frequency band into multiple sub-channels to transmit data in parallel, divides the high-speed data stream into multiple parallel low-speed data streams, and then modulates to the sub-carriers of each channel for transmission. Because it converts non-flat fading wireless channels into multiple orthogonal flat fading sub-channels, it can eliminate the interference between channel waveforms and achieve the purpose of combating multipath fading.

Orthogonal Frequency Division Multiplexing (OFDM) is an improvement on multi-carrier modulation (MCM) in. Its characteristic is that each subcarrier is orthogonal to each other, so the spectrum after spread spectrum modulation can overlap each other, which not only reduces the mutual interference between subcarriers, but also greatly improves the spectrum utilization rate.

A big reason for choosing OFDM is that the system can resist frequency selective fading and narrowband interference well. In a single carrier system, one fading or interference will cause the entire link to fail, but in a multi-carrier system, only a small number of subchannels will be affected by deep fading at a certain time.

This lab is finished individually by Zhuohao

2 Lab

2.1 Digital baseband modulation and demodulation & Simulation of OFDM Communication System

2.1.1 Analysis

Attached file `./ofdm.m` is the source code I'll refer to in this lab.

In this lab, we select channel as **AWGN**, and did modulation with **BPSK, QPSK, 16-QAM, 64-QAM**. Through Monte Carlo simulation, the performance of various modulation methods is verified, and MATLAB tools BER comparison is carried out bertool the obtained theoretical values to verify the correctness of the simulation.I n this part, we're gonna write and test the performance of the system under 16/64 QAM modulation and compare it with the theoretical curve. And we're gonna analyze the performance, advantages and disadvantages of various modulation and demodulation methods.

We'll use **IEEE 802.11a** protocol. As for its configuration, 48 subcarriers are used to transmit data, 4 subcarriers are used for pilot symbols (pilot symbols are zero in this experiment),12 subcarriers are empty (input zero is enough),OFDM configuration is completed in the OFDM mapping module in fig. 1, which is described in detail in the appendix. A total of 64 subcarriers, 64 points of IFFT/FFT.

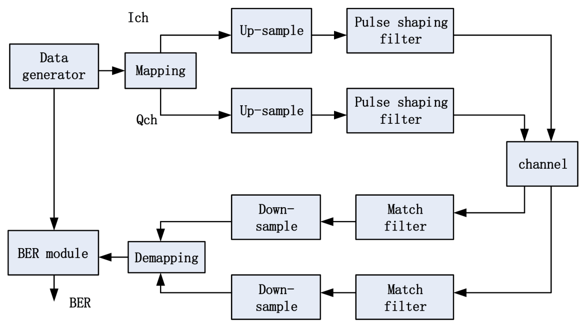

Figure 1: Modulation and demodulation - System block diagram

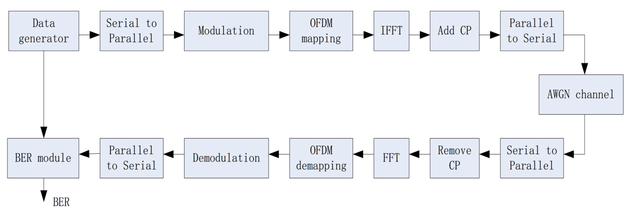

Figure 2: OFDM - System block diagram

2.1.2 Code

Modulation

`ofdm.m` is the main function used in this lab and we're gonna modify `ofdmmod.m` and `ofdmdemod.m` to implement specifc modulation and demodulation.

Reffered to lecture notes and constellation figure of 16-QAM and 64-QAM. The `ofdmmod.m` shows like below:

```
switch (OFDM.mod_lev)
    case 1 % BPSK
        OFDM.Constellations = [-1, 1];
    case 2 % 4-QAM / QPSK
        OFDM.Constellations = [-1- 1i, -1+1i, ...
                               +1-1i, +1+1i];
    case 4 % 16-QAM
        OFDM.Constellations = [-3-3i, -3-1i, -3+1i, -3+3i, ...
                               -1+3i, -1+1i, -1-1i, -1-3i, ...
                               1-3i , 1-1i , 1+1i , 1+3i , ...
                               3+3i , 3+1i , 3-1i , 3-3i];
                               % you should fix it;
    case 6 % 64-QAM
```

```
        OFDM.Constellations = [-7-7i, -7-5i, -7-3i, -7-1i, ...
                                -7+1i, -7+3i, -7+5i, -7+7i, ...
                                -5+7i, -5+5i, -5+3i, -5+1i, ...
                                -5-1i, -5-3i, -5-5i, -5-7i, ...
                                -3-7i, -3-5i, -3-3i, -3-1i, ...
                                -3+1i, -3+3i, -3+5i, -3+7i, ...
                                -1-7i, -1-5i, -1-3i, -1-1i, ...
                                -1+1i, -1+3i, -1+5i, -1+7i, ...
                                1+7i, 1+5i, 1+3i, 1+1i, ...
                                1-1i, 1-3i, 1-5i, 1-7i, ...
                                3-7i, 3-5i, 3-3i, 3-1i, ...
                                3+1i, 3+3i, 3+5i, 3+7i, ...
                                5+7i, 5+5i, 5+3i, 5+1i, ...
                                5-1i, 5-3i, 5-5i, 5-7i, ...
                                7-7i, 7-5i, 7-3i, 7-1i, ...
                                7+1i, 7+3i, 7+5i, 7+7i, ];
                                % you should fix it;
```

Demodulation

The basic idea to do demodulation is **iteration** and **condition**. Here I only take `qam16demod` as an example.

```
demodata=zeros(para,ml*nd); % output data
bitcount=ml/2; % which is 3 in 16-QAM
count=1; % counter

for ii=1:nd %from 1 to 6
   for jj=1:para % from 1 to 48
%*********************************************************************

       if(idata(jj,ii)<-6)
          demodata(jj,count:count+bitcount-1)=[0,0,0];
       elseif((idata(jj,ii)>=-6)&&(idata(jj,ii)<-4))
          demodata(jj,count:count+bitcount-1)=[0,0,1];
       elseif((idata(jj,ii)>=-4)&&(idata(jj,ii)<-2))
          demodata(jj,count:count+bitcount-1)=[0,1,1];
       elseif((idata(jj,ii)>=-2)&&(idata(jj,ii)<0))
          demodata(jj,count:count+bitcount-1)=[0,1,0];
       elseif((idata(jj,ii)>=0)&&(idata(jj,ii)<2))
          demodata(jj,count:count+bitcount-1)=[1,1,0];
       elseif((idata(jj,ii)>=2)&&(idata(jj,ii)<4))
          demodata(jj,count:count+bitcount-1)=[1,1,1];
       elseif((idata(jj,ii)>=4)&&(idata(jj,ii)<6))
          demodata(jj,count:count+bitcount-1)=[1,0,1];
       else
          demodata(jj,count:count+bitcount-1)=[1,0,0];
```

```
        end

        if(qdata(jj,ii)<-6)
            demodata(jj,count+bitcount:count+ml-1)=[0,0,0];
        elseif((qdata(jj,ii)>=-6)&&(qdata(jj,ii)<-4))
            demodata(jj,count+bitcount:count+ml-1)=[0,0,1];
        elseif((qdata(jj,ii)>=-4)&&(qdata(jj,ii)<-2))
            demodata(jj,count+bitcount:count+ml-1)=[0,1,1];
        elseif((qdata(jj,ii)>=-2)&&(qdata(jj,ii)<0))
            demodata(jj,count+bitcount:count+ml-1)=[0,1,0];
        elseif((qdata(jj,ii)>=0)&&(qdata(jj,ii)<2))
            demodata(jj,count+bitcount:count+ml-1)=[1,1,0];
        elseif((qdata(jj,ii)>=2)&&(qdata(jj,ii)<4))
            demodata(jj,count+bitcount:count+ml-1)=[1,1,1];
        elseif((qdata(jj,ii)>=4)&&(qdata(jj,ii)<6))
            demodata(jj,count+bitcount:count+ml-1)=[1,0,1];
        else
            demodata(jj,count+bitcount:count+ml-1)=[1,0,0];
        end
    end
    count=count+ml;
end
```

Input bit ($b_0 b_1 b_2$)	I-out
000	-7
001	-5
011	-3
010	-1
110	1
111	3
101	5
100	7

Input bit ($b_3 b_4 b_5$)	Q-out
000	-7
001	-5
011	-3
010	-1
110	1
111	3
101	5
100	7

Following the encode table mentioned above. Just do the corresponding condition then we can get the correct simulation result.

Vetorization to Speed Up

We can also use vectorization technique to speed it up. MATLAB is very good at matrix or vectors operations like Pytorch or Tensorflow. By vectorizing the variables you can have bonus on efficiency.

2.1.3 Result

Theorical curve is generated by `bertools` just following lectures assigned. (select modulation way → plot → export data).

And the following results are listed:

- **BPSK**

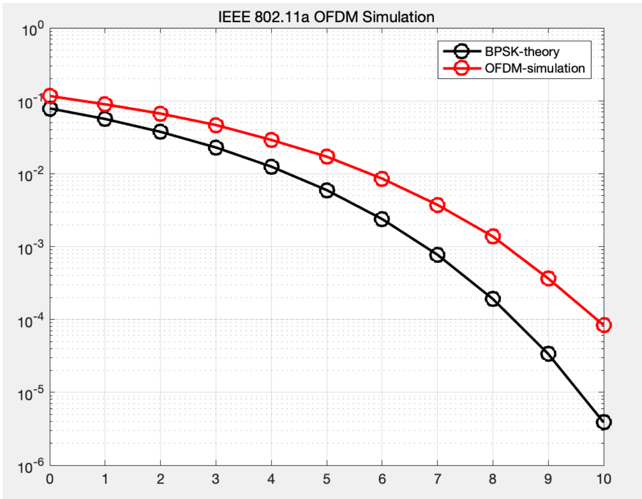

Figure 3: BPSK

  The BPSK result is shown as Fig.3.

- **QPSK**

  The QPSK result is shown as Fig.4.

- **16-QAM**

  The 16-QAM result is shown as Fig.5.

- **64-QAM**

  The 64-QAM result is shown as Fig.6.

3 Conclusion

- We could clear see that **BPSK** and **QPSK** has exactly same curve no matter in geometry or data

- When $\frac{E_0}{B_0}$ increases, EPR decreases.

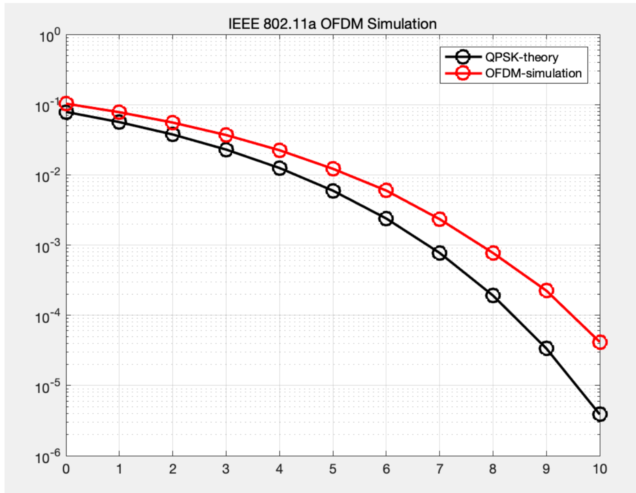

Figure 4: QPSK

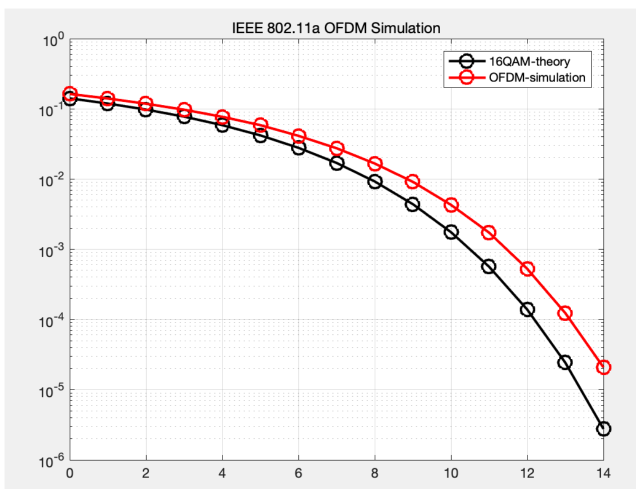

Figure 5: 16-QAM

- As the order of modulation increases,, the higher $\frac{E_0}{B_0}$ is needed to reach the same EPR
- OFDM simulation result is not exact the same as theorically. Generally, OFDM result is a little higher than we thought. We call is as shift.
- The more order is, those shift is less, which means we can match it well.

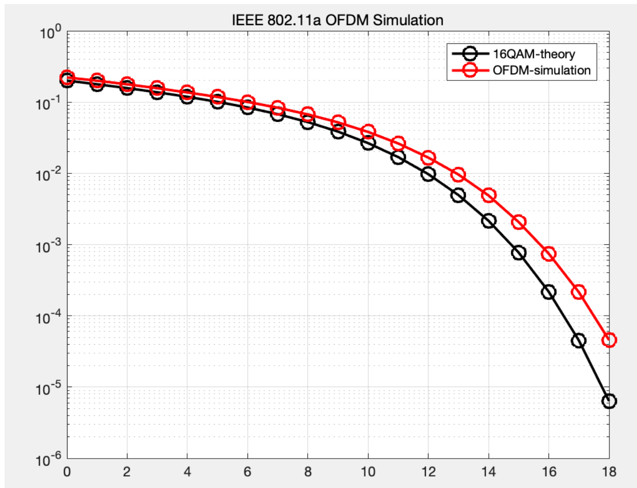

Figure 6: 64-QAM

3.1 Rethinking& Answer to Questions in Lecture

- **Functions of OFDM Mapping**

  The `ofdmmap.m` is as belows:

```
function data_out = ofdmmap(data_in, OFDM)

%****************** variables *************************
% data_in   : Input ch data
% data_out : Output ch data
% fftlen   : Length of FFT (points)
% nd       : Number of OFDM symbols
% *****************************************************

% fftlen = 64;
% nd = 6;

data_out = zeros(OFDM.fftlen, OFDM.Nd);
data_out(OFDM.ToneMap, :) = data_in(OFDM.subcarrier_order, :);

end
```

  As we can see, `ofdmmap.m` lets 48 channels of data map to IFFT input. The sender converts the transmitted digital signal into a mapping of subcarrier amplitude and phase. This ensures that adjacent coded bits are mapped to non-adjacent subcarriers and are mapped to important and non-important bits of the constellation respectively, avoiding long-term low-bit mapping.

- **Theoretical bit error rate curves of BPSK and QPSK are the same, why?**

  In textbook 4.8.4, we can find the explanation. Due to the formula about error rate:
  $$\frac{E_b}{N_0} = \frac{S}{N}\frac{W}{R}$$
  QPSK can actually be represented by 2 orthogonal BPSK. Generally, QPSK can be divided into 2 sub streams which is I and Q, I modulates $cosw_0t$ and Q modulates $sinw_0t$ at the half speed of origin. Attitude of I and Q is $\frac{1}{\sqrt{2}}$ of A. So each power of BPSK orthogonally is half of QPSK. Assume that bit speed of QPSK is R b/s, average power is S W, then each sub BPSK has bit speed of QPSK is $R/2$ b/s, average power is $S/2$ W. So
  $$\frac{E_b}{N_0} = \frac{S/2}{N_0}\frac{W}{R/2} = \frac{S}{N}\frac{W}{R}$$
  Exactly same with former one. That's why QPSK and BPSK has the same error curve. The figure theorically is shown as Fig.7

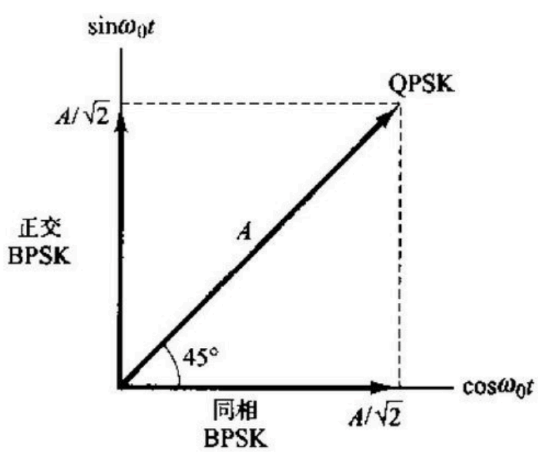

Figure 7: QPSK and orthogonal BPSK

The result in MATLAB by bertools verifies as Fig.8:

- **Try to explain the deviation between OFDM simulation error rate curve and theoretical curve.**

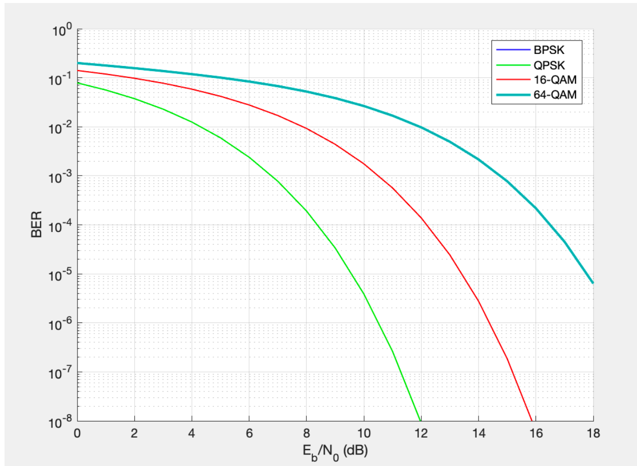

Figure 8: Result theorically

The main reason of this is because the protect interval is introduced in OFDM, it is a kind of **redundant information**, so compared with the theoretical error rate curve has happened. In this system, the protect interval uses Cyclic Prelix (CP). The latter part of the OFDM symbol is copied and added to the front of the OFDM symbol.

3.2 Problems

- The source code provided is slow. So I rewrite another verisio nto speed it up. I seperates every occasion so it can work efficiently. Besides, using **tensorl normalization** and **vectorization** could help to increase parallelisim so improve process efficiency.
- Some functions in MATLAB is not so famillar with. I browsed official ducument and look deep in them. *REPORT: MATLAB 2016Rb has some bugs when using `bertool`

3.3 Achievements

In this Lab, I have a clearer basic understanding of the communication system. We have a certain understanding of modulation and demodulation, and OFDM orthogonal frequency division multiplexing. The cost of the increase in frequency band utilization is the reduction of the signal-to-noise ratio. I have a certain understanding of the choice of various parameters of the system and how to compromise. At the same time, I know more about Matlab. The harvest is great.

Thanks for TAs for discussing and resolvutions!