Beijing-Dublin International College

EEEN3009J: Digital Communication

Lecturer: Avishek Nag

Mini Project: OFDM of IEEE802.11a WLAN

Name: Litao Cheng	**UCD ID: 17206018**

Honesty Pledge:

I clearly understand the Academic Rules of Beijing University of Technology and University College Dublin, and I am aware of the punishment for violating the Rules of Beijing University of Technology and University College Dublin. I hereby promise to abide by the relevant rules and promise the originality of my work. If found violating the rules, I would accept the punishment thereof.

Date: 18/Dec/2020

Contents

1 Introduction **2**

2 Components and Design of Simulation **3**

3 Channel Filter Coefficients Design **4**

4 System Bit Error Rate Plot **4**

5 Summary **5**

A Appendix **6**

1 Introduction

In this lab, we simulated a communication system of IEEE 802.11a Wireless LAN, using Convolution code 64-QAM modulation in every 64 sub-channels. The And the complex signals is through an impaired AWGN channel, which is design in instruction.

The IEEE 802.11a Wireless LAN standard is a Multi-carrier modulation technology which contains 20 MHz of bandwidth in the 5 GHz unlicensed band. It is the main simulation aim of this lab. OFDM(Orthogonal Frequency Division Multiplexing) is a type of multi-carrier modulation. It is capable of supporting multi-user access by providing high speed serial data transmission in parallel through frequency division multiplexing, with good resistance to multipath fading.

The MATLAB code is attached to the appendix. And the 'test3.m' script contains all simulation process, and it can run without error. Please execute 'test3.m' to test my code.

The time of this execution is acceptable:

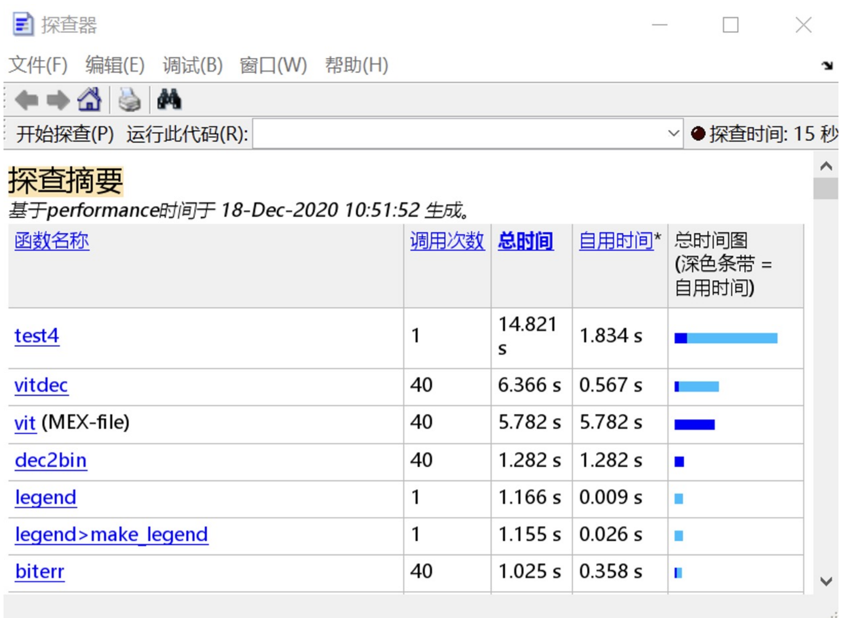

Figure 1: Simulation Profile time

The method I used in the MATLAB code contains matrix multiplying, matrix index converting, awgn() for impaired AWGN channel, convenc() for add the convolutional code, the qammod( ) to simulate the 64-QAM. and Please see the MATLAB code and Comments, they are fairly detailed and readable.

In this simulation, I simulated a $6 \times 2^{18} = 1,572,864$ bits signal transmission, which is quite large. The choices of SNR is due to the accuracy and intuition of picture with more signal symbols (or bits) to be simulated.

2 Components and Design of Simulation

The program should include the OFDM transmitter, equivalent discrete-time channel, AWGN and OFDM demodulator.

The OFDM transmitter is consists by the convolution encoder, 64-QAM constellation bit-to-symbol, mapping and IFFT. The receiver is consists by FFT, 64-QAM symbol-to-bit mapping and decoder.

The discrete-time channel is given by 64-QAM. As shown in figure 2. We can see the whole system structure.

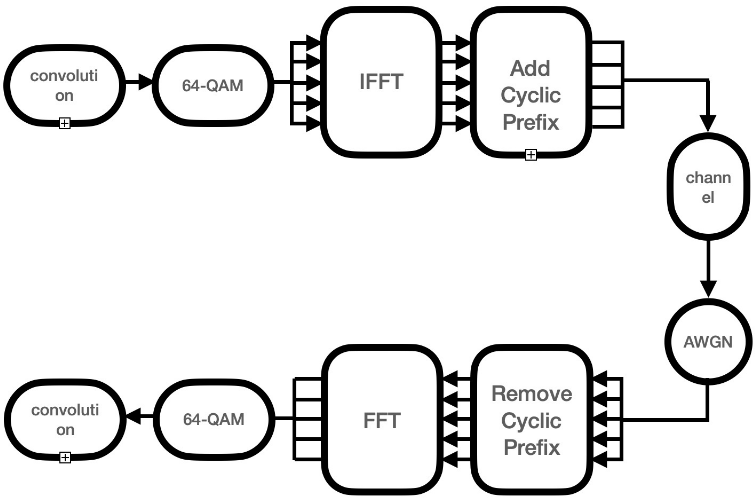

Figure 2: Simulation Structure

The convolution code rate is 1/2, so we can design code like this:

```
% convolutional for rate 1/2 feedback
trellis = poly2trellis(5, [37 33], 37);
tbdepth = 34; % Traceback depth
```

And the result of qammod is a complex number which mapping according to the constellation in figure 3.

Given by MATLAB Code:

```
complex_signals = reshape(complex_signals, [subcarrierNum, 2*bitsNum/(
subcarrierNum*6)]);
% Perform 64 ifft operation, returns the inverse transform of each column.
%padding Y with trailing zeros to length n.
complex_signals = ifft(complex_signals, 64);
```

Then, CP means that copy the final 16 channel to the beginning of the complex channels.

```
%5 Add cyclic prefix
cp_complex_signals = zeros(80, 2*bitsNum/(subcarrierNum*6));
cp_complex_signals(17:end, :) = complex_signals;
cp_complex_signals(1:16, :) = complex_signals(49:64, :);
```

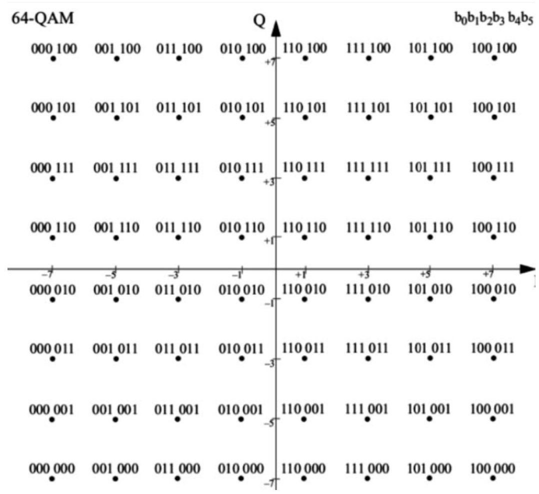

Figure 3: 64-QAM Constellation

And the demodulation process is reverse process of modulation.

3 Channel Filter Coefficients Design

Generate the channel filter coefficients as $h[n]\mu n = 0$ as i.i.d. zero-mean complex Gaussian random variables, with variance 1/2 for real and imaginary parts. Note that the zero-mean complex Gaussian random variable obey distribution combined, and the complex viable obey Rayleigh distribution. It is shown in the MATLAB code:

```
%6.add AWGN
% Generate the channel filter coefficients h[n]un=0
% as independent and identically distributed zero-mean complex Gaussian random,
% variables, with variance 1/2 for real and imaginary parts.
h = 1/(sqrt(0.5*randn+0.5*randn*1i));
channel_rayleigh = h*cp_complex_signals;
noise_gaussian = awgn(channel_rayleigh, snr, 'measured');
cp_complex_signals = h\noise_gaussian;
```

4 System Bit Error Rate Plot

Test your MATLAB program with 64-QAM modulation on all data subcarriers and r c = 1/2 and obtain a plot of the bit error rate versus received E b /N 0 in two case whether exist zeros sub-channels.

Execute the ‘test3.m’ script in MATLAB, and get the result plotted in figure 10.

We can see that the simulation with zeros channels has better performance than no zeros channels. Because they can reduce adjacent channel interference.

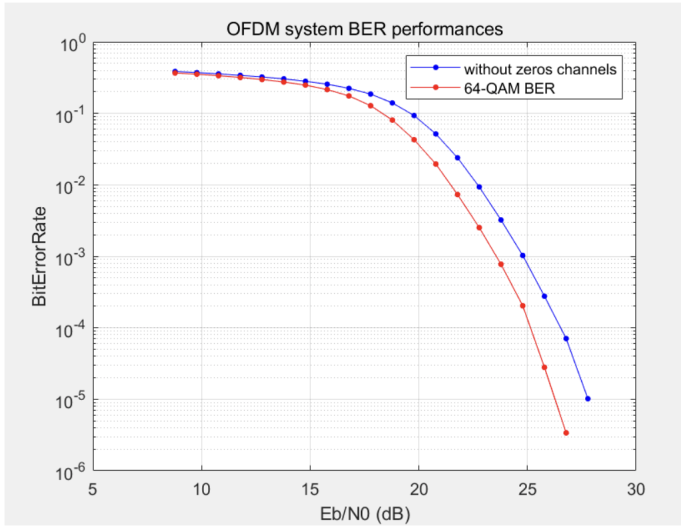

Figure 4: System BER Plot

5 Summary

The OFDM is more efficient in spectrum, it become more obvious while working with it in MATLAB. The use of transmission zeros signals can increase the performance of whole system.

A Appendix

MATLAB code:

```

clear all;
clc;
close;
%
%17206018 ChengLitao
% 1. The program should include the OFDM transmitter,
% equivalent discrete-time channel, AWGN and OFDM demodulator.
%
% 2. For the equivalent discrete-time channel,
% generate the channel filter coefficients h[n]un=0
% as independent and identically distributed zero-mean complex Gaussian
% random variables,
% with variance 1/2 for real and imaginary parts.
%
% 3. For the OFDM demodulator you can assume that the channel coefficients
% h[n]un=0 are perfectly known.
%
% 4. Test your MATLAB program with 64-QAM modulation on all data
% subcarriers and rc = 1/2
% and obtain a plot of the bit error rate versus received Eb/N0.

SNR = 1:1:20;
BER1 = zeros(1, length(SNR)); % bit error rate
subcarrierNum = 48; % the number of subcarrier with data
groupNum=2^12;
bitsNum =6*subcarrierNum*groupNum; % data scale

% convolutional for rate 1/2 feedback
trellis = poly2trellis(5, [37 33], 37);
tbdepth = 34; % Traceback depth

convertor = [32*ones(1, 2*bitsNum/6); 16*ones(1, 2*bitsNum/6);
8*ones(1, 2*bitsNum/6);4*ones(1, 2*bitsNum/6); 2*ones(1, 2*bitsNum/6);
ones(1, 2*bitsNum/6)]; % bin2dec convertor

for snr = SNR
%1.CreatbitSignal
    bits_in = randi([0 1], bitsNum, 1);

    %% OFDM Transmitter
%2convolutional code
    coded_bits = convenc(bits_in, trellis);

%3.64-QAM modulations
    coded_bits = reshape(coded_bits, [6,  2*subcarrierNum*groupNum]);
    coded_bits = sum(coded_bits.*convertor);
    coded_bits=reshape(coded_bits,[subcarrierNum,2*groupNum]);
    complex_signals = qammod(coded_bits, 64);

%4 ifft
```

```
        complex_signals = reshape(complex_signals, [subcarrierNum, 2*bitsNum/(
            subcarrierNum*6)]);
        % Perform 64 ifft operation
        % returns the inverse transform of each column of the matrix.
        %padding Y with trailing zeros to length n.
        complex_signals = ifft(complex_signals, 64);

    %5 Add cyclic prefix
        cp_complex_signals = zeros(80, 2*bitsNum/(subcarrierNum*6));
        cp_complex_signals(17:end, :) = complex_signals;
        cp_complex_signals(1:16, :) = complex_signals(49:64, :);


        %% Equivalent discrete-time channel
    %6.add AWGN
        % Generate the channel filter coefficients h[n]un=0
        % as independent and identically distributed zero-mean complex
        % Gaussian random variables,
        % with variance 1/2 for real and imaginary parts.
        h = 1/(sqrt(0.5*randn+0.5*randn*1i));
        channel_rayleigh = h*cp_complex_signals;
        noise_gaussian = awgn(channel_rayleigh, snr, 'measured');
        cp_complex_signals = h\noise_gaussian;

        %% OFDM Receiver
    %7remove CP
        complex_signals = cp_complex_signals(17:end, :);

    %8fft
        complex_signals = fft(complex_signals, 64);
        a=zeros(subcarrierNum,2*bitsNum/(subcarrierNum*6));
        a=complex_signals(1:48,:);



    %9.64-QAM demodulation
        complex_signals = reshape(a, [2*bitsNum/6, 1]);
        coded_bits = qamdemod(a, 64);
        coded_bits = dec2bin(coded_bits, 6)=='1';

    %10remove convolutional code
        coded_bits = reshape(coded_bits', [bitsNum*2 ,1]);
        bit_out = vitdec(coded_bits, trellis, tbdepth, 'trunc', 'hard');

    %11calculate BER
        [number, ratio] = biterr(bit_out, bits_in);
        BER1(snr) = ratio;
    end


    %%without zeros channels simulation

    SNR = 1:1:20;
    BER2 = zeros(1, length(SNR)); % bit error rate
```

```
    subcarrierNum = 64;
    groupNum=2^12;
    bitsNum =6*subcarrierNum*groupNum; % data scale

    % convolutional for rate 1/2 feedback
    trellis = poly2trellis(5, [37 33], 37);
    tbdepth = 34; % Traceback depth

    convertor = [32*ones(1, 2*bitsNum/6); 16*ones(1, 2*bitsNum/6);
    8*ones(1, 2*bitsNum/6);4*ones(1, 2*bitsNum/6); 2*ones(1, 2*bitsNum/6);
    ones(1, 2*bitsNum/6)]; % bin2dec convertor

    for snr = SNR
    %1.CreatbitSignal
        bits_in = randi([0 1], bitsNum, 1);

        %% OFDM Transmitter
    %2convolutional code
        %for i=1:1:subcarrierNum*groupNum
        coded_bits = convenc(bits_in, trellis);
        %end

    %3.64-QAM modulations
        coded_bits = reshape(coded_bits, [6, 2*bitsNum/6]);
        coded_bits = sum(coded_bits.*convertor);
        complex_signals = qammod(coded_bits, 64);

    %4 ifft
        complex_signals = reshape(complex_signals, [subcarrierNum, 2*bitsNum/
            (subcarrierNum*6)]); % reshape to a matrix who has 2^i row

        % Perform 2^i-point ifft operation
        complex_signals = ifft(complex_signals, subcarrierNum);

    %5 Add cyclic prefix
        cp_complex_signals = zeros(80, 2*bitsNum/(subcarrierNum*6));
        cp_complex_signals(17:end, :) = complex_signals;
        cp_complex_signals(1:16, :) = complex_signals(49:64, :);


        %% Equivalent discrete-time channel
    %6.add AWGN
        % Generate the channel filter coefficients h[n]un=0
        % as independent and identically distributed zero-mean complex Gaussian
        % random variables,
        % with variance 1/2 for real and imaginary parts.
        h = 1/(sqrt(0.5*randn+0.5*randn*1i));
        channel_rayleigh = h*cp_complex_signals;
        noise_gaussian = awgn(channel_rayleigh, snr, 'measured');
        cp_complex_signals = h\noise_gaussian;

        %% OFDM Receiver
    %7remove CP
        complex_signals = cp_complex_signals(17:end, :);
```

```

    %8fft
        complex_signals = fft(complex_signals, subcarrierNum);

    %9.64—QAM demodulation
        complex_signals = reshape(complex_signals, [2*bitsNum/6, 1]);
        coded_bits = qamdemod(complex_signals, 64);
        coded_bits = dec2bin(coded_bits, 6)=='1';

    %10remove convolutional code
        coded_bits = reshape(coded_bits', [bitsNum*2 ,1]);
        dec_bits = vitdec(coded_bits, trellis, tbdepth, 'trunc', 'hard');

    %11calculate BER
        [number, ratio] = biterr(dec_bits, bits_in);
        BER2(snr) = ratio;
    end


    % plot
    figure(1);
    semilogy(SNR+10.*log10(6), BER2,'-b.','MarkerSize',10);
    hold on;
    semilogy(SNR+10.*log10(6), BER1,'-r.','MarkerSize',10);
    title("OFDM system BER performances")
    xlabel("Eb/N0 (dB)")
    ylabel("BitErrorRate")
    legend("without zeros channels","64-QAM BER")
    grid on;
```