CE 3006
DIGITAL COMMUNICATIONS
COURSE PROJECT

Group Number: Group 6

Group Member: Chen Xueqing (U1220724J)

Wang Hu (U1120710E)

Tan Li Hau (U1222847D)

Lee Chan Khong (U1120009K)

Content Table

1 Introduction

- ## Background

Compared with analog signal, digital signal is broadly used to store music, video, text and other kinds of information, due to its advantages in various areas, such as minimizing interference, saving storage space, speeding up transmission and also defending noise effects. In digital communication system, sender sends analog information to a speaker or recorder, then the signal will immediately converted into digital signal and encoded to get ready for transmission. In order to transmit data with the least error, the digital signal will be modulated and be put to a transmission channel, where some inevitable channel noise will be added. After the digital signal being demodulated and getting rid of the noise, finally it will be decoded and transmit to the receiver side.

There are several ways of modulation, OOK, which stands for On-Off-Keying, PSK, which is the abbreviation of Phase Shift Keying, etc. Different modulation methods will have different performance and most of our present day wireless communication system use digital modulation techniques.

- ## Objective

Our project is to implement a basic communication system using MATLAB. By using MATLAB, we can generate the desired data ourselves, analyse the data, and by applying the knowledge we have learnt in this course, we can observe the outcome of the communication system that we have implemented out, and also get to learn another high leveled language, MATLAB language and may use this tool to simulate other mathematical and digital model. On the other hand, as a group project, we learn mutual understanding and effective collaboration throughout this project.

1.3 Assignment

The course project is further breakdown into three parts, so called the three phases:

- Phase 1: Data Generation
- Phase 2: Modulation for communication
- Phase 3: Basic error control coding to improve the performance

The details of the project implementation has been given. We are just to implement the system according to this guideline. In next section we will discuss more about our approaches to implement this basic communication system.

• Implementation

2.1 Phase 1: Data Generation

Binary data will be randomly generated and transmitted through additive white Gaussian noise (AWGN) channel with different SNR values.

```
clear all; close all; clc;
%Define Signal length
%Assume the number of bits for transmission is 1024
Num_Bit = 1024;
%Define signal power
Signal_Power = 1;
%SNR_dB = 10 log (Signal_Power/Noise_Power)
SNR_dB = 0:1:20;
%==> SNR = Signal_Power/Noise_Power = 10^(SNR_dB/10)
SNR = (10.^(SNR_dB/10));

%Set run times
Total_Run = 20;
%Different SNR value

for i = 1 : length(SNR)
    Avg_Error = 0;
    for j = 1 : Total_Run
        %Input singal
        %Generate random binary digits(0 or 1)
        Data = round(rand(1,Num_Bit));
        %Convert binary digit to (-1 or +1)
        Signal = 2 .* Data - 1;

        %Generate equal number of noise samples
        Noise_Power = Signal_Power ./SNR(i);
        %The randn() function is for normal distribution
        Noise = sqrt(Noise_Power/2) .*randn(1,Num_Bit);
        %Received Signal
        Receive = Signal+Noise;

        Threshold = 0;
        Error = 0;
        %Fix the threshold value as 0
        %If received signal >= threshold value, threshold = 1
        %If received signal < threshold value, threshold = 0
        for k= 1 : Num_Bit
            if (Receive(k)>= Threshold) && Data(k)==0||
                (Receive(k)<Threshold && Data(k)==1)
                Error = Error+1;
            end
        end

        %Calculate bit error rate during transmission
        Error = Error ./Num_Bit;
        %Calculate the average error for every runtime
        Avg_Error = Error + Avg_Error;
    end
    Error_Rate(i) = Avg_Error / Total_Run;
end
```

```
%Calculate analytical Bit Error Rate
Theroy_Rate=(1/2)*erfc(sqrt(SNR));

%Graph and Plot the result
figure(1)
semilogy (SNR_dB,Error_Rate,'k*');
ylabel('Pe');
xlabel('Eb/No')
hold on
semilogy (SNR_dB,Theroy_Rate,'k');
legend('sim','theory',3);
axis([0 20 10^(-5) 1]);
hold off

%data generation
figure(2)
subplot(311)
plot(Signal);
title('Generated data')
%noise generation
subplot(312)
plot(Noise);
title('Generated noise')
%received data generation
subplot(313)
plot(Receive);
title('Generated received data')
```

Observation

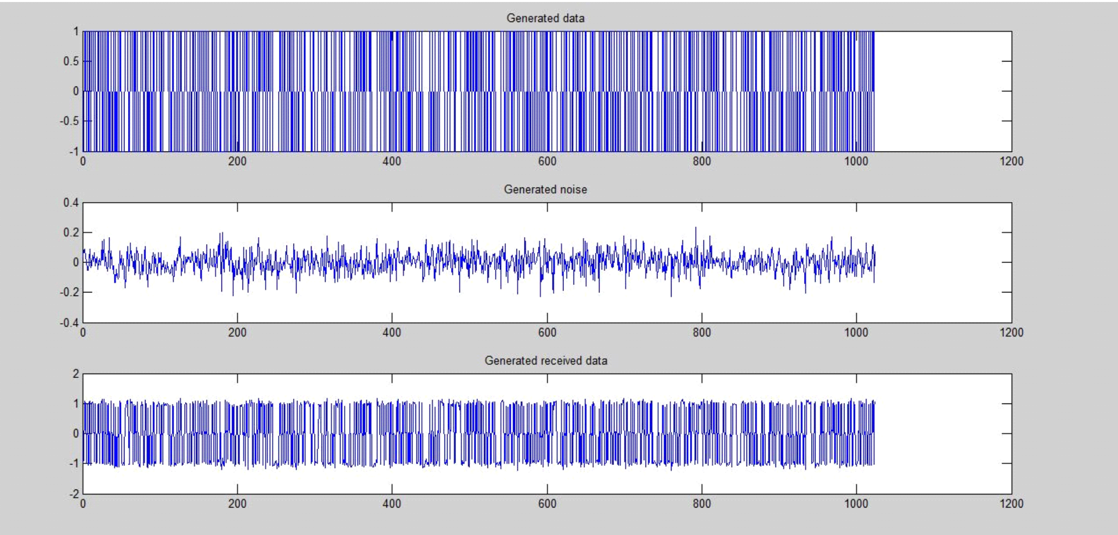


Figure 1

The comment in the code below has explained that what our code does:

- Data Generation

In phase 1, we assume the number of bits for transmission is 1024, which data was generated with random choose 0 and 1 and consist of 1024 values. (0,1,0,0,1,0,1,1...)

Then we convert the 0 to -1 and 1 to 1, which means the message data transform to bipolar non-return-to-zero.

Code in Matlab

```
Num_Bit = 1024; %Define Signal length

Data = round(rand(1,Num_Bit));  %Generate random binary digits(0 or 1)

Signal = 2 .* Data - 1;  %Convert binary digit to (-1 or +1)
```

- Noise Generation

Signals will add with white gaussian noise, we generated the noise by use this formula: Noise = sqrt(Noise_Power/2) .*randn(1,Num_Bit);

- Received Data

After we generated the input signals and noise, we add them together, this is the received signal. The Figure 1 shows the result of Data Generation, Noise Generation, and Received Data

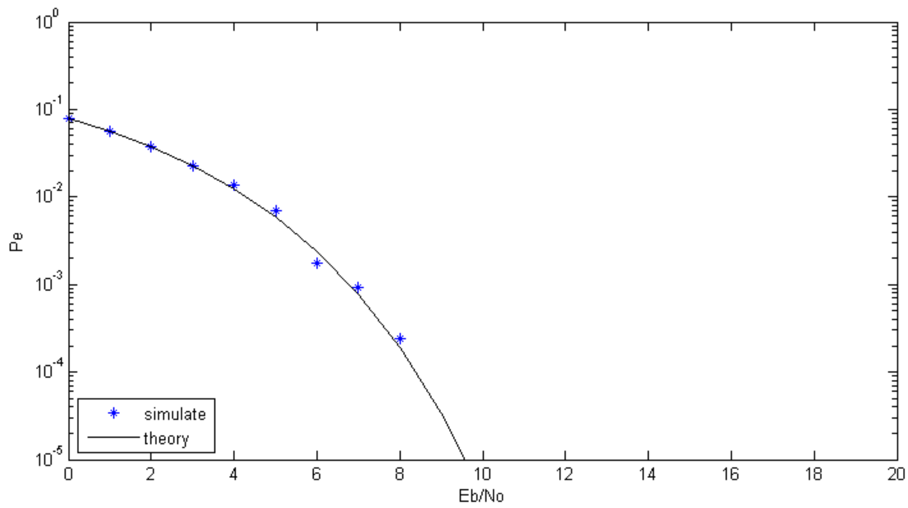


Figure 2

- Bit Error Analysis

  For received data, we set the threshold, then the error will occur, so we use the nested loop function to calculated the bit error in each run times with different SNR values, finally calculated the Bit Error Rate. Figure 2 shows the result of our BER.

2.2 Phase 2: Modulation for communication

The second phase is related to band-pass modulation and demodulation. Different band-pass modulation methods such as On-Off Keying (OOK), Binary Phase Shift Keying(BPSK) and coherent Binary Phase Shift Keying(CBPSK) are implemented in the context in the modulation process.

2.3 Phase 3: Basic error control coding to improve the performance

The last phase is about error control. We are required to implement our own error control codes or either using the codes available in MATLAB for further improvement in the performance of the proposed communication system. In this phase we have use Hamming (7, 4) - code to do the error controlling.

Hamming (7, 4) encodes 4-bits of data into 7-bits by adding 3 parity bits. This algorithm can detect up to 2-bit errors or correct 1-bit errors while without detecting other uncorrected errors. That is, Hamming codes can achieve highest possible rate for codes with block length and minimum distance.

We managed to integrate the two phases together. With the use of the data generated in phase one.

```matlab
clear all; close all; clc;
%define carrier frequency
Fc = 10000; %10kHz
%Assume carrier signal is 16 times oversampled
%define sampling frequency
Fs = 16 * Fc;
%define data rate
dataRate = 1000; %10kbps
%Define Signal length
Num_Bit = 1024;
%define num bit after encoding
Enc_Num_Bit = 1792;
%Define Amplitude
Amplitude = 5;

%low pass butterworth filter
%6th order, 0.2 cutoff frequency
[b, a] = butter(6, 0.2);

%high pass butterworth filter
[d, c] = butter(6, 0.2, 'high');

%time
t = 0: 1/Fs : Enc_Num_Bit/dataRate;

%Carrier
Carrier = Amplitude .* cos(2*pi*Fc*t);

%signal length
SignalLength = Fs*Enc_Num_Bit/dataRate + 1;

%SNR dB = 10 log (Signal Power/Noise Power)

SNR_dB = 0:1:20;
%==> SNR = Signal_Power/Noise_Power = 10^(SNR_dB/10)
SNR = (10.^(SNR_dB/10));

% Set run times
Total_Run = 20;

Error_RateOOK = zeros(length(SNR));
Error_RateBPSK = zeros(length(SNR));

%Different SNR value
for i = 1 : length(SNR)
    Avg_ErrorOOK = 0;
    Avg_ErrorBPSK = 0;

    for j = 1 : Total_Run
        %Generate Data
        Data = round(rand(1,Num_Bit));
        %encode
        EncodeHamming= encode(Data,7,4,'hamming/fmt');

        ContinuousData = zeros(1, SignalLength);
        for k = 1: SignalLength - 1
            ContinuousData(k) = EncodeHamming(ceil(k*dataRate/Fs));
        end
        ContinuousData(SignalLength) = ContinuousData(SignalLength - 1);

        %on-off keying
        SignalOOK = Carrier .* ContinuousData;
```

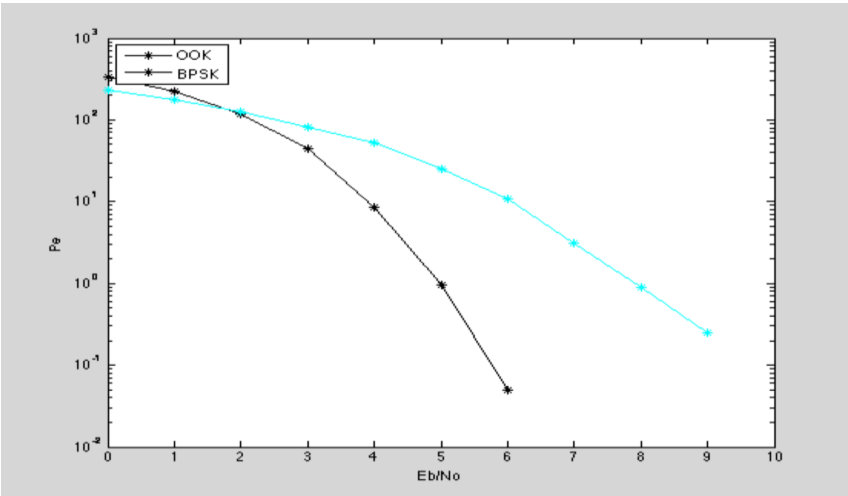
```
%binary phase shift keying
ContinuousDataBPSK = ContinuousData .* 2 - 1;
SignalBPSK = Carrier .* ContinuousDataBPSK;

Signal_Power_OOK = (norm(SignalOOK)^2)/SignalLength;
Signal_Power_BPSK = (norm(SignalBPSK)^2)/SignalLength;

%Generate Noise OOK
Noise_Power_OOK = Signal_Power_OOK ./SNR(i);
NoiseOOK = sqrt(Noise_Power_OOK/2) .*randn(1,SignalLength);
%Received Signal OOK
ReceiveOOK = SignalOOK+NoiseOOK;

%OOK detection
SquaredOOK = ReceiveOOK .* ReceiveOOK;
%low pass filter
FilteredOOK = filtfilt(b, a, SquaredOOK);

%Generate Noise BPSK
Noise_Power_BPSK = Signal_Power_BPSK ./SNR(i);
NoiseBPSK = sqrt(Noise_Power_BPSK/2) .*randn(1,SignalLength);
%Received Signal BPSK
ReceiveBPSK = SignalBPSK+NoiseBPSK;

%non-coherent detection
SquaredBPSK = ReceiveBPSK .* ReceiveBPSK;
%high pass filter (supposingly band pass filter)
FilteredBPSK = filtfilt(d, c, SquaredBPSK);

%frequency divider
DividedBPSK = interp(FilteredBPSK, 2);
```

```
%frequency divider
DividedBPSK = interp(FilteredBPSK, 2);
DividedBPSK = DividedBPSK(1:length(FilteredBPSK));

%Multiple and Low Pass Filter
MultipliedBPSK = DividedBPSK .* ReceiveBPSK;
OutputBPSK = filtfilt(b, a, MultipliedBPSK);

%demodulate
%sampling AND threshold
samplingPeriod = Fs / dataRate;
[sampledOOK, resultOOK] = sample_and_threshold(FilteredOOK, samplingPeriod, Amplitude/2, Enc_Num_
[sampledBPSK, resultBPSK] = sample_and_threshold(OutputBPSK, samplingPeriod, 0, Enc_Num_Bit);

%Calculate the average error for every runtime
%Avg_ErrorOOK = num_error(resultOOK, EncodeHamming, Num_Bit) + Avg_ErrorOOK;
%Avg_ErrorBPSK = num_error(resultBPSK, EncodeHamming, Num_Bit) + Avg_ErrorBPSK;

decodedOOK = decode(resultOOK,7,4,'hamming/fmt');
decodedBPSK = decode(resultBPSK,7,4,'hamming/fmt');

ErrorOOK = 0;
ErrorBPSK = 0;
for k = 1: Num_Bit - 1
    if(decodedOOK(k) ~= Data(k))
        ErrorOOK = ErrorOOK + 1;
    end
    if(decodedBPSK(k) ~= Data(k))
        ErrorBPSK = ErrorBPSK + 1;
    end
end
```

```
                if(decodedBPSK(k) ~= Data(k))
                    ErrorBPSK = ErrorBPSK + 1;
                end
            end
            Avg_ErrorOOK = ErrorOOK + Avg_ErrorOOK;
            Avg_ErrorBPSK = ErrorBPSK + Avg_ErrorBPSK;

        end
        Error_RateOOK(i) = Avg_ErrorOOK / Total_Run;
        Error_RateBPSK(i) = Avg_ErrorBPSK / Total_Run;
end

figure(1)
semilogy (SNR_dB,Error_RateOOK,'k-*');
hold on
semilogy(SNR_dB, Error_RateBPSK, 'c-*');
%axis([0 20 10^(-5) 1]);
hold off
legend('OOK', 'BPSK',2);
ylabel('Pe');
xlabel('Eb/No')



figure(2)
title('Received Signal OOK');
plot(ReceiveOOK, 'k')

figure(3)
title('Squared OOK');
plot(SquaredOOK, 'k');
```

```
figure(4)
title('Filtered Signal OOK');
plot(FilteredOOK, 'k');

figure(5)
title('Captured Data');
plot(sampledOOK);
```

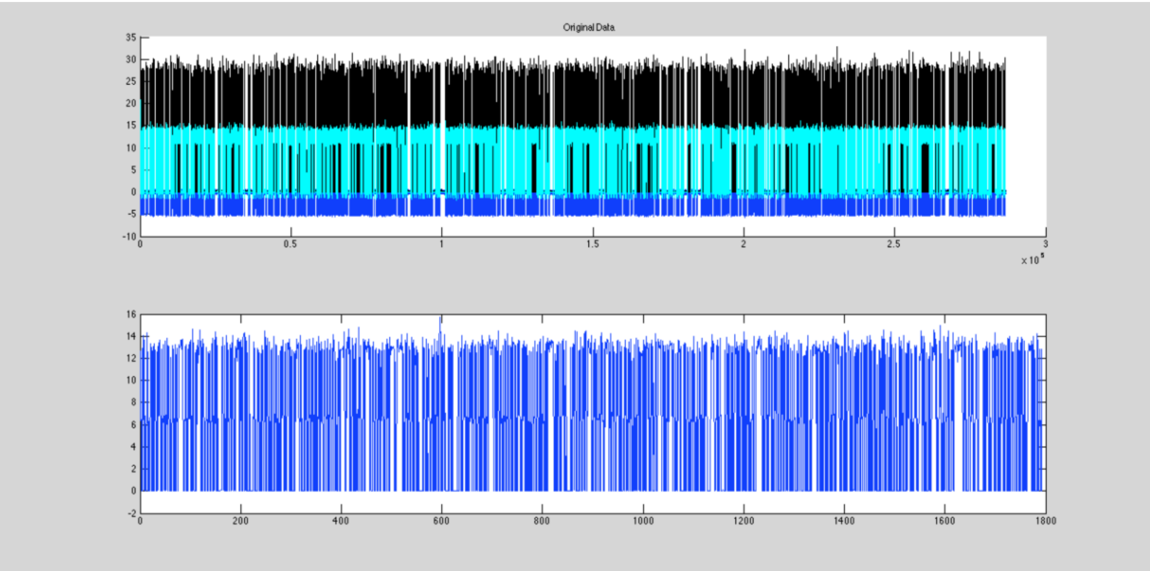
Observation



Graph one

Comments on Graph one:

Graph one plotted bit error rates of two different modulation methods with respect to different SNR. By observation, the bit error rate of OOK(Amplitude Shift Keying) are much more accurate than BPSK(Phase Shift Keying) as we can see on the condition of the same SNR, the bit error rate of OOK is bigger than the bit error rate of BPSK. So within the range 1 to 20, OOK modulation has better performance as BPSK.



Graph two

Comment on Graph two:

Graph two was plotted quite vague; the data is generated in a way that the Original, received, squared, filtered data almost together at the same position. The captured OOK signal is much larger than the original binary data signal.

Conclusion

In this project, we generate digital data and compare bit data rate with respect to the SNR, and got a brief understanding of the implementation of three different modulation methods.

We plotted the graph and also did proper analyses and documentation.

During this project, though almost all the processes are done individually, we helped each other finished roughly. Signal communication has a broad range of application stage and we are inspired to learn more and did better in the future.

Our project can be considered being finished beautifully.