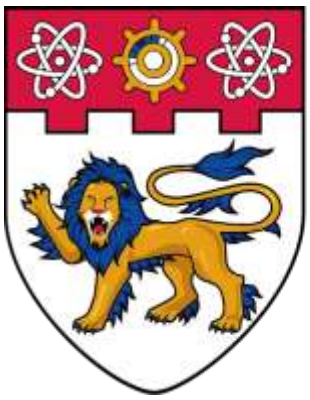

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering (SCSE)

CE3006: Digital Communications

Project Report

Agnesh Ramesh (U1921141K)
Kesarimangalam Srinivasan Abhinaya (U1923302J)
Singh Aishwarya (U1923952C)
Unnikrishnan Malavika (U1923322E)
Vedula Kartikeya (U1923891G)

Table of Contents

1. Project Scope and Objective

The goal of this course project is to create a simple digital communication system employing MATLAB and investigate its functioning through various digital communication processes.

2. Introduction

The conveyance of information from one place to another via a multitude of methods is referred to as communication. It entails the generation of a message signal, the precise description of that signal, the encoding of symbols in a form suitable for transmission over a physical medium of interest, the transmission of encoded symbols to the desired destination, the decoding and reproduction of original symbols, and the recreation of original message signal with a definable degradation in quality.

Communication systems are divided into two types: digital and analog. Analog Communication Systems use a transmission medium to send analog waveforms. They are conceptually simpler, but more complex to construct, and there is no major attempt to adapt the waveform to suit the channel distortions. However, with Digital Communication Systems, the transmitted signal is transformed to digital, which is complicated conceptually but simple to implement. It is feasible to identify a finite set of waveforms that fit channel properties and are thus more resistant to channel distortions. It is capable of efficient and reliable transmission but necessitates a significant quantity of electrical hardware.

Digital communication techniques are utilized. Digital signals are less susceptible to noise and distortion. They can be perfectly rebuilt over great distances by regenerative repeaters, avoiding error propagation. Digital data from several sources may be readily multiplexed. They can also be encoded to rectify mistakes and encrypted for privacy. Digital circuits that are reasonably cheap can be employed.

This project report will explore the implementation of a simple digital communication system – from generating the data to modulating it for transmission. It also explores methods to improve performance by using error control coding.

3. Digital Communication System Overview

The three main components of a digital communication system are as follows –

- Transmitter
- Channel
- Receiver

The signal is received by the transmitter from the source. The sent signal is then routed through the channel, where it travels to the receiver before being delivered to the user. A signal is a mathematical function-based representation of a physical quantity represented as a function of time. In addition, a system is any physical

equipment that generates an output signal in response to an input signal. Digital communication systems are thought to be linear time invariant. This means it has a linear response in both time and frequency. In a digital communication system, the channel is considered to be additive in nature, with other channel effects being disregarded.

On the transmitter side, the signal from the source is routed through an analog to digital converter, which samples, quantizes, and converts the raw analog signals to digital bits for transmission. After that, the digital signal is encoded and modulated (bandpass level) to broadcast at a higher carrier frequency. Continuing, the same signal is demodulated on the receiver side to get the original frequency, decoded, and sent through a digital to analog converter to retrieve the original signal that was broadcast by the source.

This entire process can be represented in Figure 1 below. The scope of this project is to simulate the steps depicted in the figures and explore the different techniques used throughout the digital communication process to transmit signals from source to the receiver.

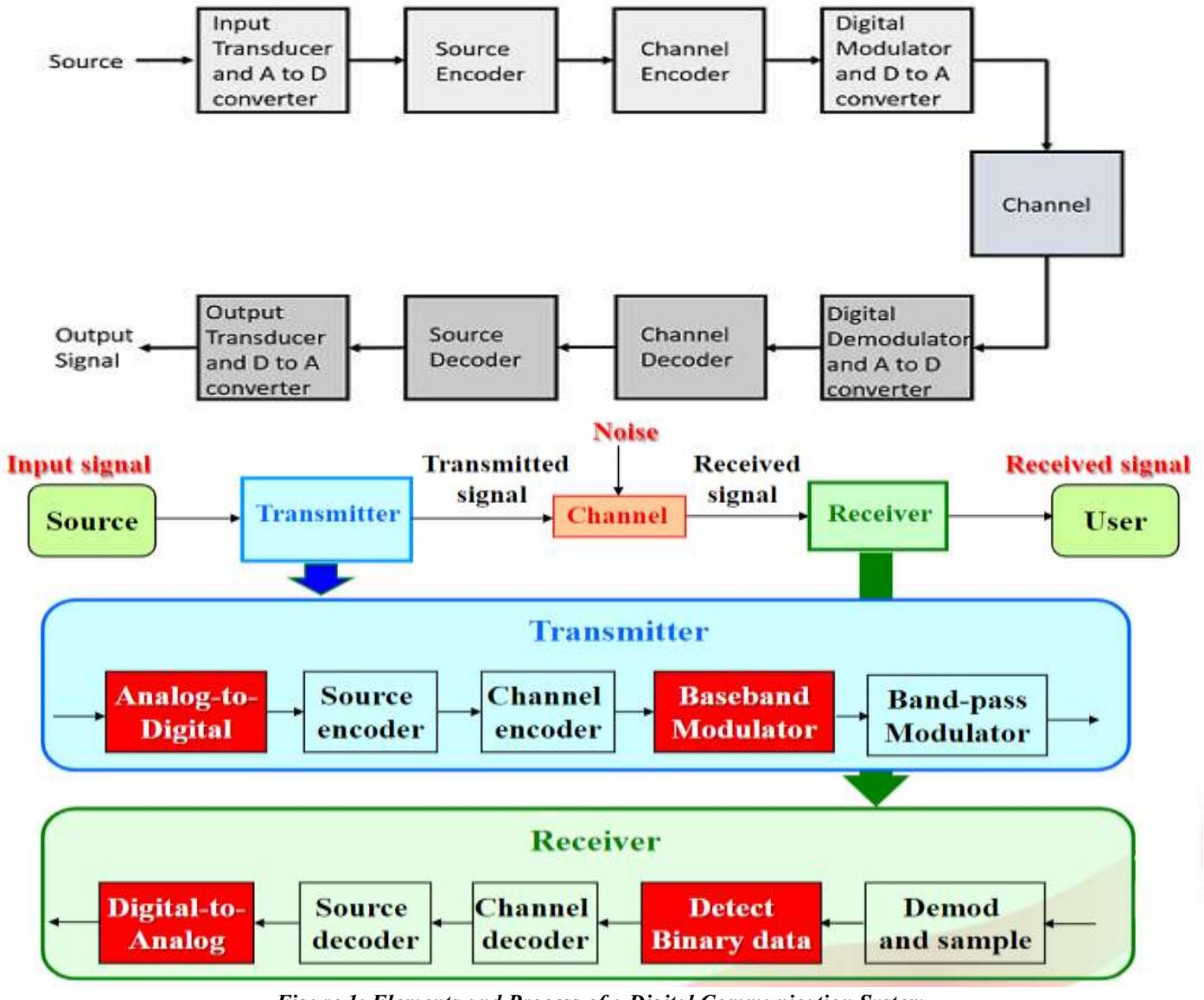

Figure 1: Elements and Process of a Digital Communication System

4. Implementation

4.1. Running the code

To demonstrate the functionality of the digital communication system, run the following files in the given order -

4.2. Phase 1: Data Generation

This phase contains information about baseband modulation and demodulation. The binary bitstream will be produced here and broadcast over an additive white Gaussian noise channel with varying SNR values. The performance of the bit error rate will be analysed and displayed against various SNR levels.

First, we declare variables that will be used in data generation. It is given that the number of bits for transmission is 1024 (which indicates that N = 1024). Then, a series of random binary digits (0 or 1) is generated. Furthermore, binary digits are converted to 1 (that is, 1 to +1 and 0 to -1). This is the information the system will send. Next, an equal amount of noise samples is produced. The produced noise has a normal distribution with a mean of zero and a variance of one unit. The noise power in relation to the SNR (signal to noise ratio) value is then modified. Finally, the SNR value is set to that effect. These definitions are declared in the commented MATLAB code below –

```
%Number of bits = 1024 (given)
nBits = 1024;
%Generating random binary digits
Data = randi([0 1],1,nBits);
%Converting to +1 and -1
Signal = 2 .* (Data - 0.5);
%Generating noise with mean = 0 and variance = 1
Noise = randn(1,nBits);
%Fixing SNR value to 10 dB
SNR = 0:5:50;
%Creating a result array to store error for each value of SNR
Result = zeros([1 11]);
%Current index to track result
index = 1;
```

Following that, SNR is utilized to create noise variance, which is expressed in decibels. The signal (the input data) is assumed to have unit power (S=1). The noise power is then calculated and used in conjunction with the noise samples to produce the desired noise. The noise samples are then mixed in with the data being sent. This is the signal that was received.

```
%Looping over the different values of SNR
for i = 1:length(SNR)
    currSNR = SNR(i);
    %SNR = 10log(S/N), where S = Signal Power, N = Noise Power
```

```
    %Signal has unit power -> S = 1
    %Therefore, using above equation, N = 0.1
    noisePower = 0.1.^(currSNR/10);
   %Adjusting Noise based on the SNR value
    Noise = sqrt(noisePower) .* Noise;
    %Calculating the received signal
    Received = Signal + Noise;
```

At the receiver, a threshold logic is then used. This value is set to 0. If the received signal is greater than or equal to the threshold, it is assigned a value of 1, otherwise it is assigned a value of 0. The bit error rate is then calculated. The threshold logic's output values are compared to the binary digits input. The processes from noise addition through bit error rate computation are repeated for different SNRs.

```
%Fixing a threshold value for the received signal
    Threshold = 0;

    %Calculating the output based on the threshold level
    Output = zeros(1,nBits);
    for k = 1 : nBits
        if Received(k) > Threshold
            Output(k) = 1;
        else
            Output(k) = 0;
        end
    end

    %Computing the total number of errors
    Error = 0;
    for k = 1 : nBits
        if Output(k) ~= Data(k)
            Error = Error + 1;
        end
    end
    %Calculating the bit error rate
    %BER = (Total errors)/(Number of bits)
    BER = double(Error)/double(nBits);
    Result(index) = BER;
    index = index + 1;
```

4.3. Phase 2: Modulation for Communication

This section contains information about band-pass modulation and demodulation. The team has implemented several band-pass modulation algorithms in the project. Simpler modulation methods like On-Off Keying and Binary Phase Shift Keying have been developed as base modulation techniques. In addition to the basic algorithms, the team has also implemented Binary Frequency Shift Keying to explore the impact of different modulation techniques in communication systems.

4.3.1. On-Off Keying (OOK)

The simplest kind of amplitude-shift keying (ASK) modulation is on–off keying (OOK), which expresses digital data as the presence or absence of a carrier wave. In its most basic form, the presence of a carrier for a certain period indicates a binary one, whereas the lack of a carrier for the same duration represents a binary zero.

The first step is to declare all variables that are to be used in the developed algorithm. From the previous phase, the number of bits for transmission remains the same at 1024 bits. The carrier frequency is given as 10kHz, where the carrier signal is oversampled by 16 times, as defined by the variable `carrierSignal`. The baseband data rate is given as 1kbs. Next, the sampling period formula is defined in the code as the ratio of the carrier signal and the data rate. Then, a low pass butterworth filter is implemented using a MATLAB library function `butter`. The amplitude is then set as 1, and the time period range is set, as defined in the equation given in the MATLAB code (`t`). Additionally, the Signal to Noise Ratio is declared (in dB as well), and the error rate of the OOK algorithm is initialised to an array of zeros with the length of SNR. Finally, the carrier is defined, for demodulation by the formula given as `amp.*cos(2*pi*carrierFrequency*t)` and the length of the signal, along with the number of runs of this algorithm are declared in the code snippet shown below -

```
nBits = 1024;
carrierFrequency = 10000;
carrierSignal = carrierFrequency * 16;
dataRate = 1000;
samplingPeriod = carrierSignal / dataRate;
[lowB, lowA] = butter(6,0.2);
amp = 1;
t = 0: 1/carrierSignal : nBits/dataRate;
SNR_dB = 0:5:50;
SNR = (10.^(SNR_dB/10));
modifySNR_dB = 5;
errorRateOOK = zeros(length(SNR));
carrier = amp .* cos(2*pi*carrierFrequency*t);
signalLength = carrierSignal*nBits/dataRate + 1;
numRuns = 10;
```

The next step was to design a function for simulating a decision device and sampling. The sampling function `sample` takes in 3 parameters – the output signal of the low pass filter, the sampling period and the number of bits. The input signal is then sampled to produce a sampled signal which is then passed to the decision device, where the created function is called. The `decision_device` function takes in 3 parameters as well, namely the sampled signal, the number of bits and the threshold value

(half of amplitude in this case). The output of the decision device is then used for error calculations. The code snippet below depicts the two functions -

```
% Sampling and Decision Device Simulation
function sampled = sample(x,sampling_period,num_bit)
    sampled = zeros(1, num_bit);
    for n = 1: num_bit
        sampled(n) = x((2 * n - 1) * sampling_period / 2);
    end
end

function binary_out = decision_device(sampled,num_bit,threshold)
    binary_out = zeros(1,num_bit);
    for n = 1:num_bit
        if(sampled(n) > threshold)
            binary_out(n) = 1;
        else
            binary_out(n) = 0;
        end
    end
end
```

The final step is to put everything together and create the OOK algorithm. As depicted in Figure 1 of Digital Communication System Overview, there are different steps that are involved in modulation and demodulation of a signal. In the OOK algorithm, first, the noise is generated, then, the transmitted signal is obtained (sum of signal and noise). Next, the transmitted signal is passed to a non-coherent detector – square law detector – and is then passed through a low pass filter, sampled, and the final signal is retrieved. Lastly, the error is calculated. The code snippet below depicts the working of the aforementioned OOK algorithm –

```
% Noise Generation
OOKSignalPower = (norm(OOKSignal)^2)/signalLength;
OOKNoise = OOKSignalPower ./SNR(i);
OOKNoise = sqrt(OOKNoise/2) .*randn(1,signalLength);

% Transmit signal
OOKTransmit = OOKSignal + OOKNoise;

% Use non-coherent detection - square law detector
sqLawOOK = OOKTransmit .* OOKTransmit;

% Pass this through the LP filter
LPfilterOOK = filtfilt(lowB, lowA, sqLawOOK);

% Sample the signal
OOKsample = sample(LPfilterOOK, samplingPeriod, nBits);
finalResultOOK = decision_device(OOKsample,nBits, amp/2);
```

```
% Calculate Error
errorOOK = 0;
for k = 1: nBits - 1
     if(finalResultOOK(k) ~= data(k))
          errorOOK = errorOOK + 1;
     end
end
avgErrorOOK = errorOOK + avgErrorOOK;
```

4.3.2. Binary Phase Shift Keying (BPSK)

Binary phase shift keying (BPSK) is the most primitive form of PSK, with "binary" referring to the usage of two-phase offsets (one for logic high, one for logic low). We may intuitively see that more separation between these two stages will make the system more resilient. Because we have only 360° of phase to deal with, the greatest difference between the logic-high and logic-low phases is 180°. Inverting a sinusoid is the same as shifting it by 180°; hence, BPSK is just inverting the carrier in response to one logic state and leaving it alone in response to the other logic state.

The implementation of BPSK is similar to that of OOK in section 4.3.1. For BPSK, however, the data signal needs to be represented in +1 or -1 format. Other than the given format, the steps for modulation and demodulation are the same from the OOK section, with the sampling and decision device functions being recycled for BPSK as well. The code snippet below depicts the BPSK algorithm –

```
bpskSourceSignal = 2.* dataStream - 1;
bpskSignal = carrier .* bpskSourceSignal;

bpskSignalPower = (norm(bpskSignal)^2)/signalLength;
bpskNoisePower = bpskSignalPower ./ SNR(i);
bpskNoise = sqrt(bpskNoisePower/2) .* randn(1,signalLength);

bpskTransmit = bpskSignal + bpskNoise;

bpskDemodulated = bpskTransmit .* (2 .* carrier);
bpskFiltered = filtfilt(lowB, lowA,  bpskDemodulated);

bpskSample = sample(bpskFiltered, samplingPeriod, nBits);
finalResultBPSK = decision_device(bpskSample,nBits, 0);

errorBPSK = 0;
for k = 1: nBits - 1
     if(finalResultBPSK(k) ~= data(k))
          errorBPSK = errorBPSK + 1;
     end
end
avgError = avgError + errorBPSK/nBits;
```

4.3.3. Binary Frequency Shift Keying (BFSK)

The bit stream in frequency-shift keying (FSK) is represented by changes between two frequencies. In a binary frequency-shift key mechanism, the two binary states, logic 0 (low) and 1 (high), are each represented by an analog waveform. Logic 0 is represented by a certain frequency wave, while logic 1 is represented by a different frequency wave.

Following the algorithms from sections 4.3.1 and 4.3.2, BFSK algorithm has been designed to have 2 different carrier signals**,** one with a frequency of 10 kHz and the other with 5 kHz. After this step, the modulated signal is calculated by calculated based on the signal required to be transmitted. Iterating through the range of SNR values from -50 to 50 with a step size of 5, we calculate the BER for each value of SNR. For transmission, a noise with noise power based on the current iteration of SNR is added to the signal. Coherent demodulation is used on the receiver's end. After demodulation, the signal is sampled to generate the result signal. This result signal is used to compute the bit error rate. The code snippet below demonstrates the working of BFSK –

```
% Define the 2 carrier functions
carrier1 = cos(2*pi*carrierFrequency*t);
carrier0 = cos(pi*carrierFrequency*t);

% Find the modulated signal
FSKmodulated1 = carrier1 .* (transmittedSignal == 1);
FSKmodulated2 = carrier0 .* (transmittedSignal == -1);
modulatedSig = FSKmodulated1 + FSKmodulated2;
sigPower = rms(modulatedSig)^2;

index = 0;
SNRvalues = -50:5:50;
meanBitError = zeros([1 length(SNRvalues)]);
theoreticalError = zeros([1 length(SNRvalues)]);
for SNR = SNRvalues
    index = index+1;
    bitErrorRate = zeros([1 nBits]);
    % Calculate noise power from SNR
    noisePower = sigPower/(10^(SNR/10));

    % Run the experiment for each sample 20 times
    for sample = 1:20
        % Generate the noise
        Noise = randn(1,length(modulatedSig));
        Noise = sqrt(noisePower) .* Noise;

        % Find the transmitted signal after noise
        transmittedFSK = modulatedSig+Noise;

        %Coherent demodulation of FSK
```

```
BFSK_demod1 = transmittedFSK .* 2.* carrier1;
BFSK1_filter = filtfilt(b,a,BFSK_demod1);
BFSK_demod0 = transmittedFSK .*2 .* carrier0;
BFSK0_filter = filtfilt(b,a,BFSK_demod0);
BFSK_demod = BFSK1_filter -BFSK0_filter ;

count = 0;
result = zeros([1 nBits]);
```

4.4. Phase 3: Error Control Coding

The fundamental difficulty of communication is duplicating a message picked at one location, either exactly or approximately, at another site. This jeopardizes the digital communication system's dependability. The coding process used to control the incidence of mistakes is known as error control coding. These strategies aid in the detection and correction of errors. Depending on the mathematical concepts used, there are several error correcting codes. However, these codes have been generically classed as Linear Block codes and Cyclic codes.

The parity bits and message bits in linear block codes have a linear combination, which implies that the final code word is the linear combination of any two code words. Until it is changed, every linear block code can be a systematic code. As a result, an unchanged block code is referred to as a systematic code. It indicates that data encryption should not modify the data.

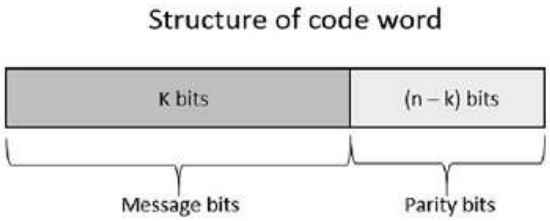

Figure 2: Linear Block Code representation

Convolutional or cyclic coding is a coding method in which the output code bits are defined by logic operations on the current bit in a stream and a limited number of prior bits rather than by blocks of bits. Data bits are sent into a shift register in the encoder. As each bit enters the register from the left, the previous bits shift to the right, and the oldest bit in the register is eliminated. Two or more binary summing operations result in code bits that are produced during a single data flow period.

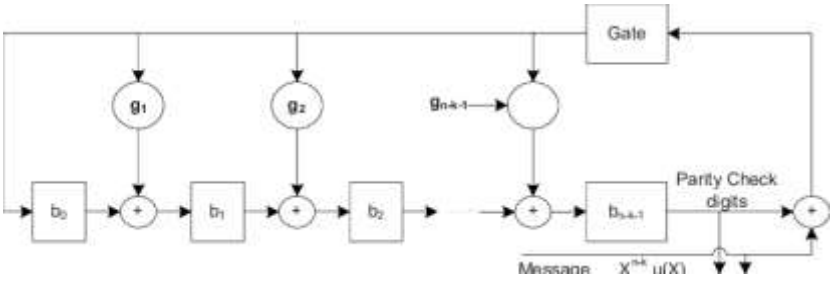

Figure 3: Cyclic Code representation

4.4.1. Hamming Codes

The code word's linearity feature states that the total of two code words is likewise a code word. Hamming codes are a sort of linear error correcting code that may detect up to two-bit faults or repair one-bit errors while ignoring untreated errors. Extra parity bits are utilized when employing hamming codes to identify a single bit mistake. A few bits in the data must be modified to transition from one bit pattern to the other. This number of bits is known as the Hamming distance. A one-bit flip can be detected if the parity is separated by two bits. This, however, cannot be changed. Furthermore, any two-bit flip cannot be recognized. In the proposed digital communication system, Hamming Codes have been implemented for all the chosen techniques in the report. The functions used are from the MATLAB error control code library, `encode` and `decode`. The parameters passed are the signal/data stream generated, the received signal, and the message and codeword lengths.

4.4.2. Cyclic Codes

In coding theory, a cyclic code is a block code in which the circular shift of each codeword yields another codeword. They are error – correcting codes that make efficient error detection and correction possible. In this method, the information is sent across the channel by combining with parity bits. The properties of linearity and cyclic shifting hold for this method. MATLAB functions are used to implement the same.

5. Results

This section discusses the results obtained from the implementation phase for the proposed digital communication system.

5.1. Phase 1: Data Generation

Phase 1 implemented baseband processing. The signal contains randomly generated bits between -1 and 1. The noise added to the signal is additive white gaussian noise whose power is determined based on the

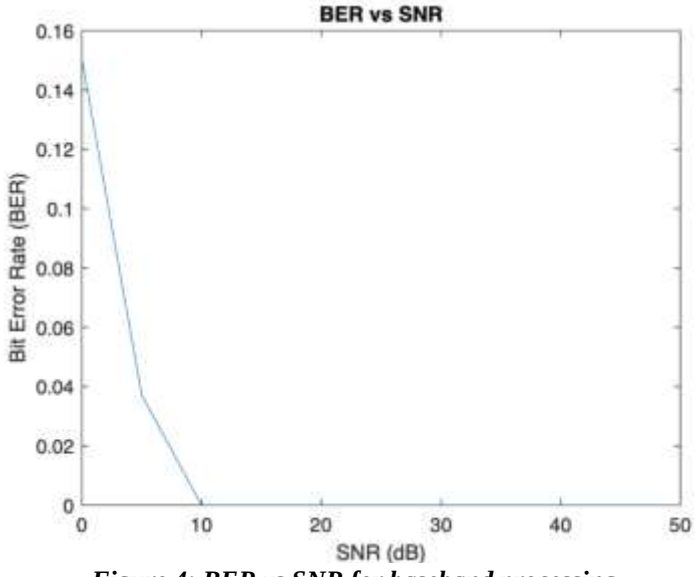

Figure 4: BER vs SNR for baseband processing

chosen SNR value for the current iteration. The BER obtained for different values of SNR can be observed as shown. As expected, we notice that the BER values reduces as the SNR value increases. This shows that as the signal power increases, the probability of receiving an erroneous signal at the receiver's end decreases.

5.2. Phase 2: Modulation for communication

The team implemented three methods for band – pass modulation, namely, On – Off Keying, Binary Phase Shift Keying and Binary Frequency Shift Keying. This helped observe all three modulation techniques and compare the outputs of the same. Coherent detection techniques were used at the receiver's end for demodulation.

5.2.1. On – Off Keying

On – Off Keying is the simplest form of Amplitude Shift Keying. The results of performing OOK Modulation Scheme are as follows:

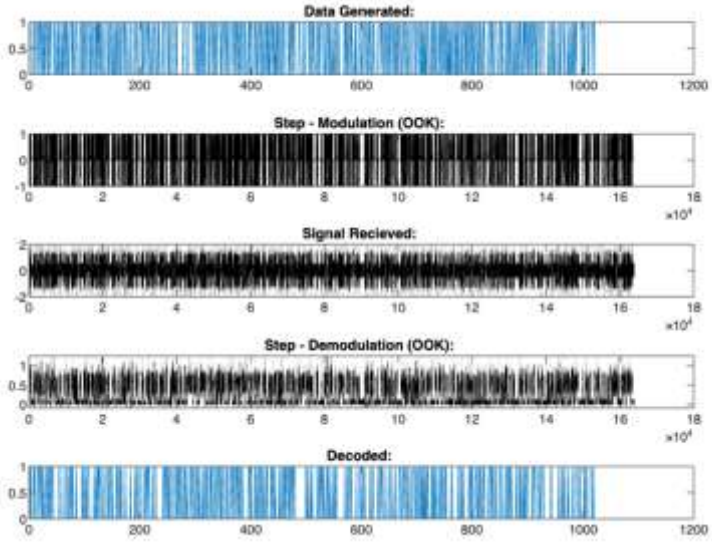

Figure 5: Stages of Modulation and Demodulation

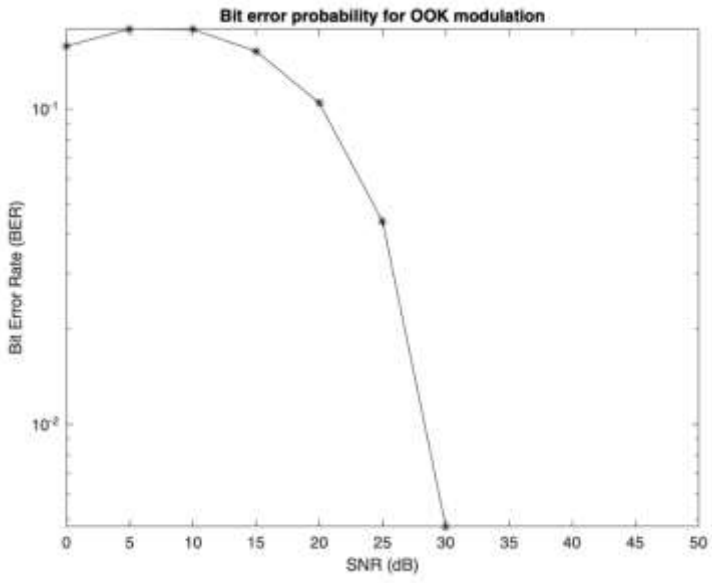

Figure 6: BER vs SNR for OOK

The observed plots follow the expected output as the SNR and BER values follow an inverse relation.

5.2.2. Binary Phase Shift Keying

Binary Phase Shift Keying is the simplest form of Phase Shift Keying. The results of performing BPSK are observed as follows:

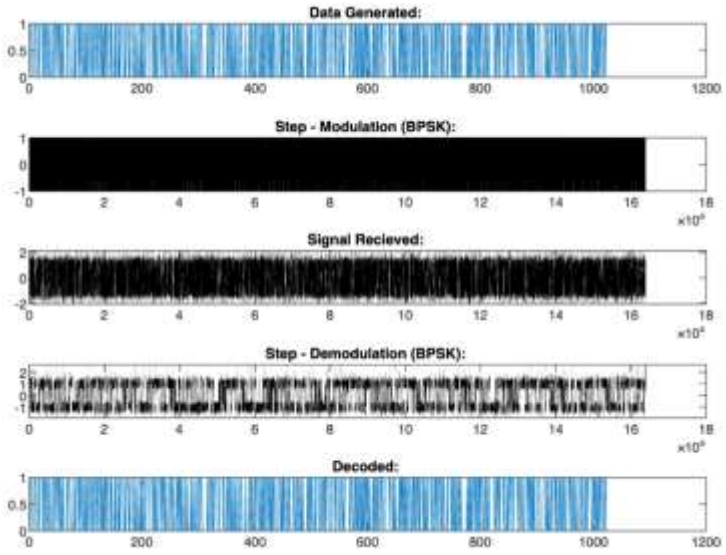

Figure 7: Stages of Modulation and Demodulation

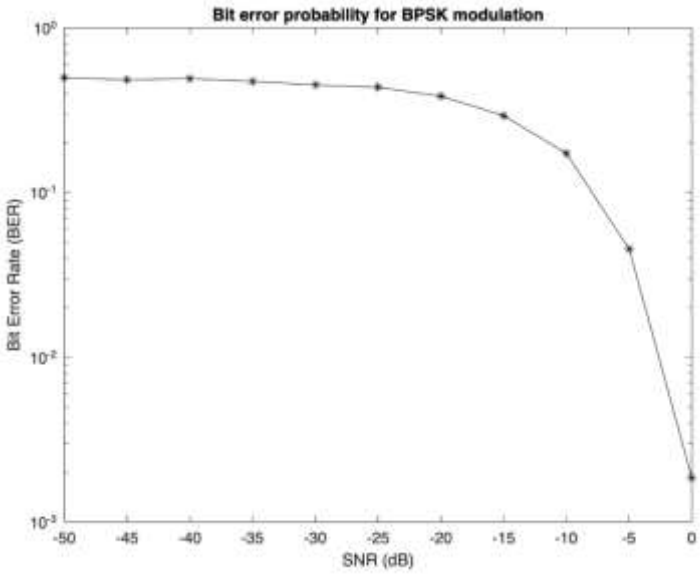

Figure 8: BER vs SNR for BPSK

The observed plots follow the expected output as the SNR and BER values follow an inverse relation.

5.2.3. Binary Frequency Shift Keying

Binary Frequency Shift Keying is the simplest form of Frequency Shift Keying. The results of performing BFSK are observed as follows:

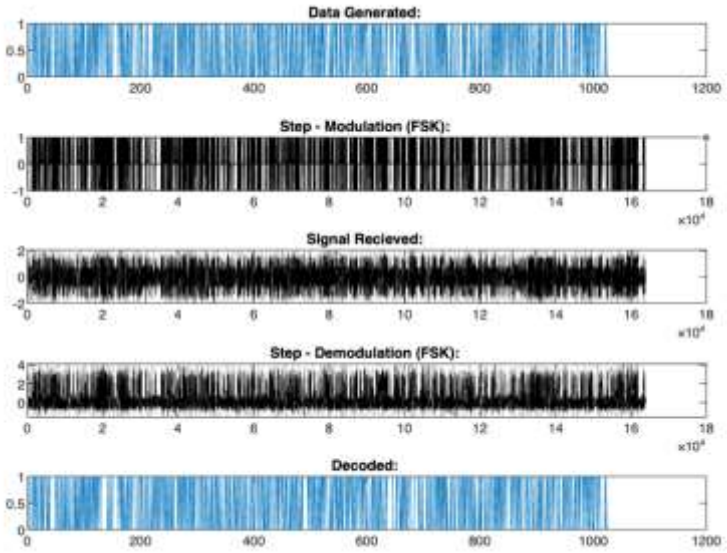

Figure 9: Stages of Modulation and Demodulation

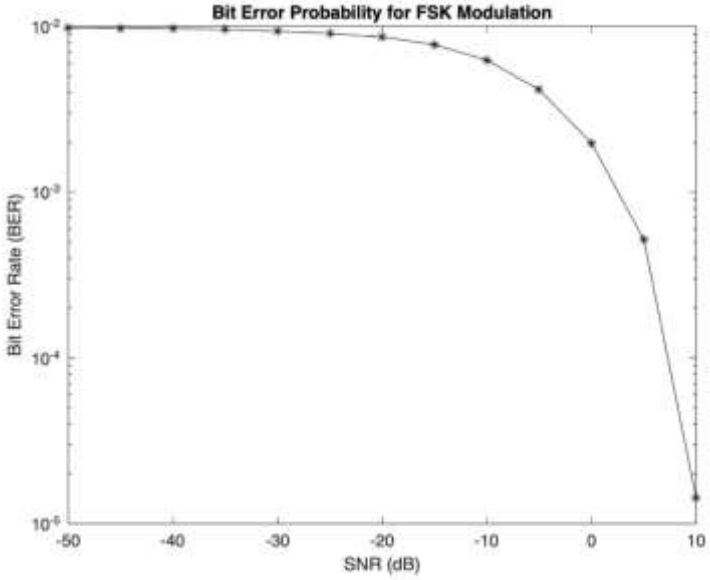

Figure 10: BER vs SNR for BFSK

The observed plots follow the expected outputs as the BER and SNR values follow and inverse relation.

5.3. Phase 3: Error Control Coding

Channel Coding helps in error detection and correction on the receiver's side. Two methods of channel coding were implemented, namely, Hamming and Cyclic codes were implemented for each of the modulation schemes that were implemented in Phase 2 and were compared.

5.3.1. Channel Codes applied to On – Off Keying

The results obtained by applying hamming and cyclic code techniques show an improvement in the BER performance for the communication system.

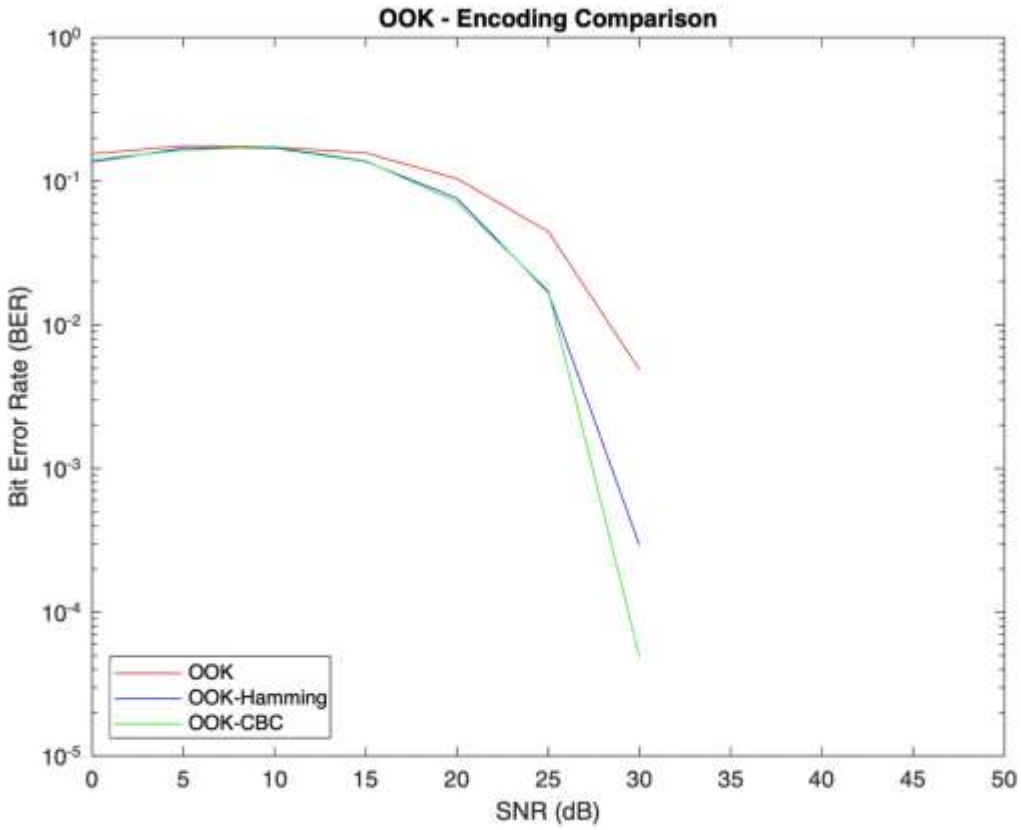

Figure 11: Comparison of channel coding techniques for OOK

5.3.2. Channel Codes applied to Binary Phase Shift Keying

The results obtained by applying hamming and cyclic code techniques show an improvement in the BER performance for the communication system.

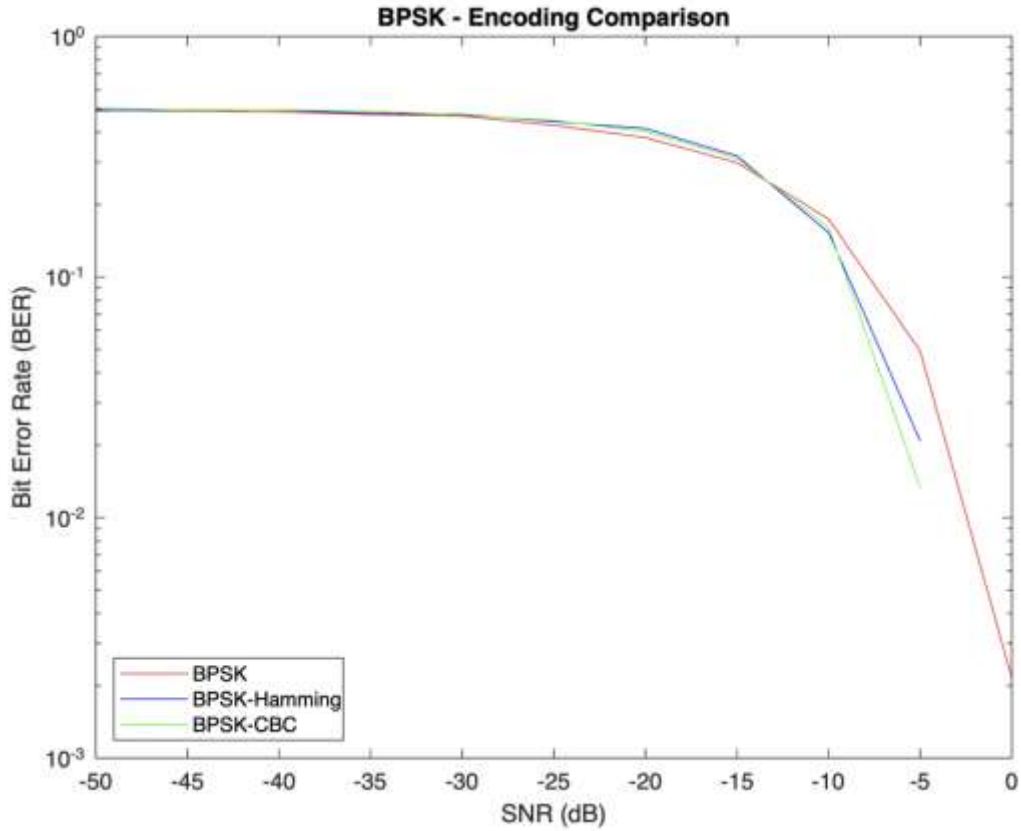

Figure 12: Comparison of channel coding techniques for BPSK

5.3.3. Channel Codes applied to Binary Frequency Shift Keying

The results obtained by applying hamming and cyclic code techniques show an improvement in the BER performance for the communication system.

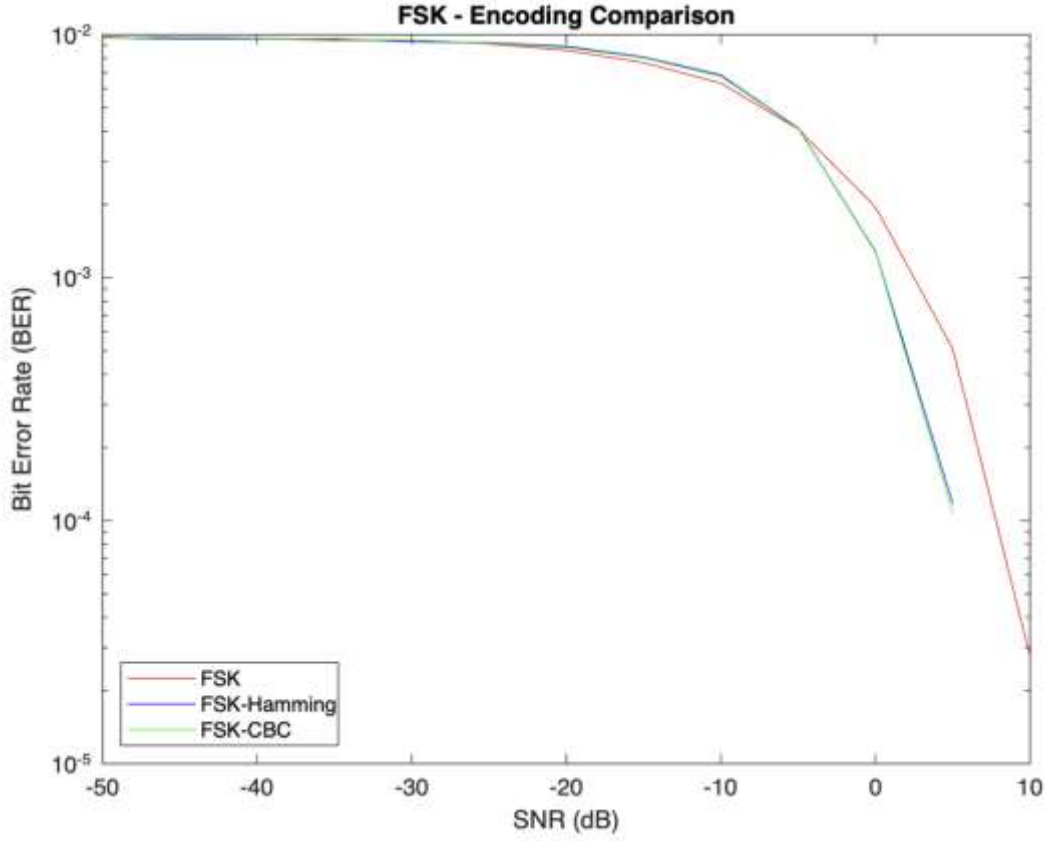

Figure 13: Comparison of channel coding techniques for BFSK

6. Conclusion

In conclusion, the team was able to observe first hand the impact of noise on signal transmission from the transmitter to the receiver. We were able to implement different techniques to improve the communication system performance and reduce the bit error rate of transmission. We observed both baseband and bandpass signal processing as well as channel coding mechanisms.

- Among the different bandpass modulation schemes, OOK performed the worst with the BER reaching zero only at an SNR value of 30 dB. On the other hand, BPSK and BFSK were able to achieve zero bit error rate at 5 dB and 10 dB respectively. This is concurrent with theory that states that amplitude shift keying is sensitive to amplitude changes in the channel while FSK and PSK are insensitive to the same.
- Applying channel coding techniques like hamming and cyclic codes improved the performance for all three modulation techniques. CBC, in general, performed better than Hamming when applied to OOK, BPSK and FSK.

7. Scope for Improvement

- Implement M-ary modulation schemes to compare the trade-off between bandwidth efficiency and power efficiency.
- Implement Quadrature Amplitude modulation to inspect the modulation of both amplitude and phase in the same modulation scheme.

8. Bibliography

Academic.microsoft.com. 2021. Microsoft Academic. [online] Available at: <https://academic.microsoft.com/topic/170030856/publication/search?q=On-off%20keying&qe=And(Composite(F.FId%253D170030856)%252CTy%253D%270%27)&f=&orderBy=0> [Accessed 28 October 2021].

Keying, B., 2021. BPSK - Binary Phase Shift Keying. [online] Mpirical. Available at: <https://www.mpirical.com/glossary/bpsk-binary-phase-shift-keying> [Accessed 28 October 2021].

Northern Illinois University. 2021. Binary Frequency Shift Keying - NIU - Internet Accessible Remote Laboratories. [online] Available at: <https://www.niu.edu/remote-lab/resources/graphical-user-interfaces/frequency-shift.shtml> [Accessed 28 October 2021].

Encyclopedia Britannica. 2021. frequency-shift keying | communications. [online] Available at: <https://www.britannica.com/technology/frequency-shift-keying> [Accessed 28 October 2021].

A. Froehlich, “What is frequency-shift keying (FSK)?,” SearchNetworking, 01-Oct-2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/frequency-shift-keying. [Accessed: 28-Oct-2021].

https://www.tutorialspoint.com/digital_communication/digital_communication_error_control_coding.htm#:~:text=Error%20control%20coding%20is%20the,mathematical%20principles%20applied%20to%20them.

https://www.sciencedirect.com/topics/engineering/convolutional-coding