CE3006 Digital Communications

Project Report

Alfred Huang Haowen (U1721674C)

Poon Jun Yaw (U1720270D)

Do Anh Tu (U1721947F)

Tan Jia Le (U1720600F)

James Tan Chee Min (U1720400A)

School of Computer Engineering

Nanyang Technological University

30th October 2020

TABLE OF CONTENTS

TABLE OF CONTENTS **2**

Introduction **3**

Tasks & Objectives **4**

Phase 1: Data generation 4

Phase 2: Modulation for communication 4

Phase 3: Basic error control coding 5

Implementations **6**

Phase 1: Data generation 6

Phase 2: Modulation for communication 8

Phase 3: Basic error control coding 13

Experiments and Results **14**

Phase 1: Data generation 14

Phase 2: Modulation for Communication 17

Phase 3: Basic error control coding 26

Conclusion **30**

Appendix **31**

1. Introduction

Digital communication systems are systems that transmit digital signals from one point to another through a series of successive processes. These systems are very popular today due to their tolerance to noise and distortion, near-perfect regenerative capabilities, low cost components and more.

Conceptually, digital communication systems models are constructed of transmitter, channel and receiver blocks as shown in the *Figure 1* below. Input signals are passed through a series of processing steps in the transmitter block to ensure efficiency and reliability in transmission, before being passed on to the communication channel. The channel is inherently noisy due to its physical medium property and the received signal at receiver end is often corrupted with these distortions. A series of processing is then conducted at this end to retrieve the original transmitted data, each is a corresponding reverse process to that in the transmitter block.

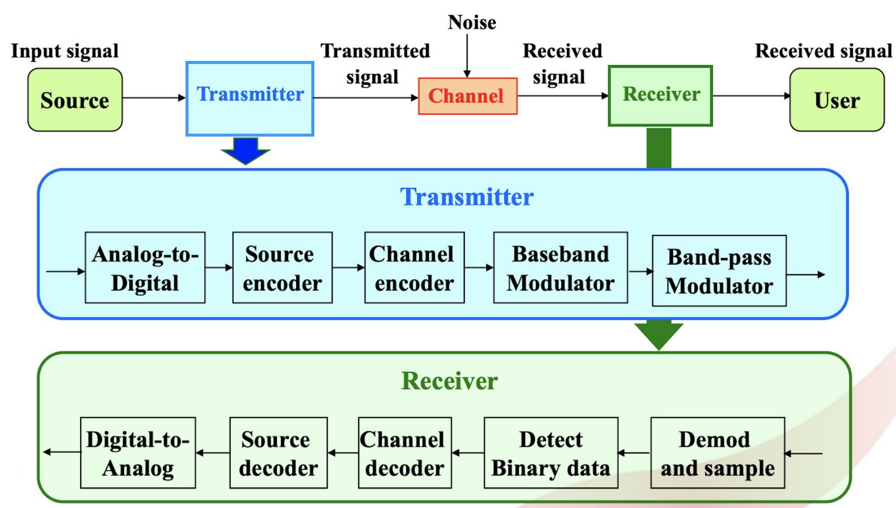

Figure 1. Block diagram of components for Digital Communication Systems

While designing and implementing digital communication systems requires knowledge of circuitry, one may simulate the various building blocks in *Figure 1* with the help of softwares like MATLAB.

Therefore, this project implements a software-based system to study the effect of noise in communication and the working principles for each of the sub-block in transmission and receiver blocks. Implementations and results will be attached in the report to study how data can be transferred over a noisy channel to the recipient.

2. Tasks & Objectives

The main objective of this project is to build and simulate the interaction between various basic building blocks of the digital communication systems as shown in the *Figure 1* above. To do so we split the building of this system into 3 main phases, which corresponding to baseband processing on transmitter and receiver (phase 1), bandpass modulation and demodulation on top of baseband processing (phase 2) and finally coding techniques to increase reliability in transmission of signals (phase 3).

a. Phase 1: Data generation

Phase 1 is concerned with transmission and reception of baseband processed signals through a channel corrupted with Additive White Gaussian (AWG) noise. We have to first generate 1024 bits of binary data for transmission. This will be our transmitted signal as shown in *Figure 2*.

After that, we have to generate the noise that is used to simulate the data transmission through the additive white Gaussian noise channel. The number of bits for the noise is the same as the generated data, which is 1024. This generated noise will be the noise to the channel as shown in *Figure 2*.

The simulation is then done by adding the generated noise to the generated data. The resultant will then be the corresponding received signal from the channel as shown in *Figure 2*, which is corrupted with noise.

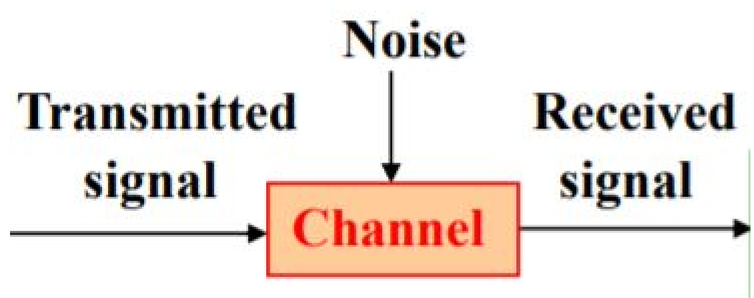

Figure 2. Interaction of Noise and channel on transmitted signal

After both transmitted signal and received signal is ready, we can calculate the bit error rate (BER) by comparing them. This represents the effect of AWG on transmitted signals in digital communication systems.

b. Phase 2: Modulation for communication

Phase 2 is concerned with adding bandpass processes to our simple digital communication systems, building on top of phase 1. In this phase, we explore the effect of modulation, demodulation and detection of signals corrupted with AWG from the channel. Modulation allows us to send a baseband signal over a certain band pass frequency range. In other words, the receiver-end only has to capture data in the specified band pass frequency, creating more efficient data transfer. Other than that, modulation is required in real world communication as we are often sharing channels with other devices as well, different protocols will make use of different frequencies.

We are required to implement the two basic modulation techniques: On Off Keying (OOK) and Binary Phase Shift Keying (BPSK). These two techniques will be used to modulate a 1024 bit message with a carrier frequency of 10kHz. The carrier frequency will also be used to determine our sampling frequency, we have set the sampling frequency to 16 times of the carrier frequency. So, our sampling frequency is set to 160kHz.

In addition to these 2 two amplitude and phase modulation techniques, we attempted to implement a third modulation technique of frequency: Binary Frequency Shift Keying (BFSK). This is achieved over 2 different carrier frequencies of 10kHz and 30kHz.

Once the baseband signal is modulated to bandpass (by OOK, BFSK or BPSK), an additive white gaussian noise is then added to the signal to simulate noise during transmission. The power of the noise will be varied according to the signal-to-noise ratio (SNR), from 0dB to 50dB. Once noise is added to the signal, we then try to recover the message by demodulation and detection of the received signal.

Finally, we compare the original message with the recovered message to calculate the bit error rate (BER) of each modulation technique. This is to confirm our knowledge from this course on bandpass processes.

c. Phase 3: Basic error control coding

In phase 2, we have looked into different modulation methods to achieve an efficient and less interfered transmission from the transmitter to the receiver via the channel. However, the signal is still susceptible to data corruption due to the inherent noise that exists in the channel. Therefore, in phase 3, we will look into ways to enhance the error detection and error correction ability of our system in phase 2 by adding on channel coding blocks. In this phase, we introduce channel coding to improve and reduce the bit error rate of our transmission.

Channel encoding is applied to the baseband signal before it is modulated to bandpass for transmission via the channel. The signal will then be demodulated and decoded to retrieve the original data. In this study, we have implemented two channel coding techniques, **Hamming Code** and **Cyclic Code**. These codes belong to the family of linear block correction codes and provide the error detection and correction ability. We will be implementing both channel encoding techniques with a codeword length of 7 bits for a 4 bit message. Once the encoded signals are modulated, transmitted, demodulated and detected, we obtain the BER by comparing the decoded signal with the original message to be transmitted.

3. Implementations

a. Phase 1: Data generation

Data generation

We start by generating the data for transmission. Here, we used a matlab $rand$ function to generate 1024 random data ranging from 0 to 1. We then round the number so that all the data is either 0 or 1 (unipolar line coding). After that, we multiply the data by 2 and subtract 1 from it so that it ranges from -1 to 1 (bipolar line coding) as required in the project specification.

Noise generation

After that, we generate the noise using a matlab $randn$ function. The generated noise using this function has a zero mean and unit variance. To match the noise power (noise variance) to the desired Signal-to-Noise Ratio (SNR), we calculate the noise power from the desired SNR using the formula: $SNR = 10log_{10}\frac{S}{N}dB$, where $S = Signal\ power$ and $N = Noise\ power$. In this case, signal power S is assumed to be unit (S=1). After the noise power is calculated, we add the calculated noise power (noise variance) into the noise that we have generated earlier: $Noise_{new} = \sqrt{N} * Noise_{old}$, where $Noise_{new} = Noise\ with\ new\ variance$, $Noise_{old} = Noise\ with\ unit\ variance$ and $N = Noise\ power$.

Signal transmission and signal detection

After the noise with desired noise power is generated, we then add the noise to the generated data. The resultant is the received signal, corrupted with AWG. At the receiving end, we have to pass this received signal through a threshold logic to determine whether the received signal is a 1 or -1.

Empirical BER Calculation

We can then compare the output of the threshold logic with the generated data, which is the transmitted signal, and calculate the BER for the selected SNR. All the steps from noise generation to this step are repeated for 20 times so that we can get an average of BER for the selected SNR. This average BER is based on the actual signal and we will refer to it as real BER.

As the noise corrupting our transmitting signal is a Gaussian random process that obeys Central Limit Theorem, more repetitions as mentioned will give a smoother

BER curve as noise is then approximately closer to the random distribution curve (shown in the next part).

Theoretical BER Calculation

We also calculated theoretical BER using the formula: $\frac{1}{2}erfc(\sqrt{\frac{SNR}{2}})$. The derivation of this formula is shown below.

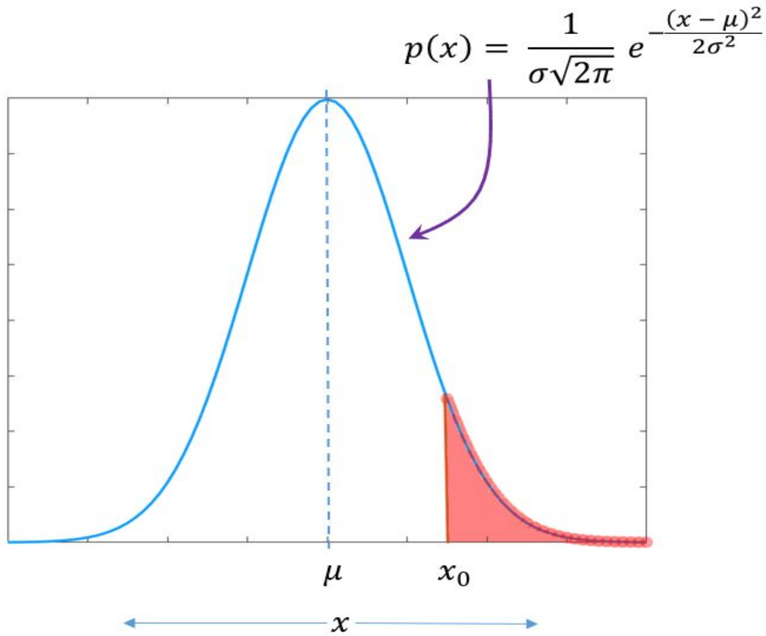

Figure 3. Gaussian distribution pdf (Bell-curve)

<u>Given a noise with 0 mean and unit variance</u>

When transmitting a 1, an error will occur if the noise is less than or equals to -1. So, the probability of error, $P_e(1)$ is equal to the area under the curve for $x_o \leq -1$.

Similarly, when transmitting a -1, an error will occur if the noise is greater than or equals to 1. So, the probability of error, $P_e(-1)$ is equal to the area under the curve for $x_o \geq 1$ as shown in Figure 3.

These two areas are the same so we can say:

$$P_e(1) = P_e(-1) = Q(1) \text{-------(1)}$$

<u>Given a noise with 0 mean and variance σ^2</u>

If the noise is no longer having a unit variance, we have to rescale the data to zero mean and unit variance in order to use formula (1) to calculate BER. Therefore, a new formula is derived:

$$P_e(1) = P_e(-1) = Q(\frac{1-\mu}{\sigma}) \text{-------(2)}$$

Since noise variance is equal to noise power, $^2 = N$, and $N = \frac{S}{SNR} = \frac{1}{SNR}$, we know that $\sigma = \sqrt{\frac{1}{SNR}}$. Substituting this into formula (2), we get:

$$P_e(1) = P_e(-1) = Q(\sqrt{SNR}) \text{ -------(3)}$$

We also know that $Q(x) = \frac{1}{2}erfc(\frac{x}{\sqrt{2}})$, therefore we get:

$$P_e(1) = P_e(-1) = \frac{1}{2}erfc(\sqrt{\frac{SNR}{2}}) \text{ -------(4)}$$

Since the probability of data 1 is equal to the probability of data -1, we can say that $BER = \frac{1}{2}P_e(1) + \frac{1}{2}P_e(-1) = P_e(-1) = \frac{1}{2}erfc(\sqrt{\frac{SNR}{2}})$.

b. Phase 2: Modulation for communication

Carrier Signal

Modulation in communications requires a carrier signal for our message signal to be modulated into. In our implementation, we will be using a cosine waveform with an amplitude of 5 and a frequency of 10,000Hz for both OOK and BPSK modulation.

$$Signal_{carrier} = 5cos(2\pi * 10000 * t)$$

For BFSK modulation, we make use of another carrier signal of amplitude of 5 and frequency of 30,000Hz besides the above carrier signal for modulating frequency.

Sampling of Signal

Sampling is required to simulate the transmission, hence we will be choosing a sampling frequency of 160,000Hz (as per specification of 16x of carrier frequency). The baseband data rate is also set to 1000bps.

Noise

An Additive White Gaussian Noise (AWGS) will also be generated and added to the signal during transmission to simulate a real transmission. We will be testing out SNR values of 0dB to 50dB to investigate the performance of the modulation techniques implemented.

On-Off Keying Modulation

The implementation of on-off keying is pretty straight forward. If the signal bit is a 1, we will transmit the instantaneous value of the carrier signal and if the signal bit is a 0, we will transmit a 0.

Binary Phase Shift Keying Modulation

The implementation of BPSK is also similar to ASK, as a 180° phase shift for a cosine signal is just the negative of the original signal. Thus, we implement BPSK modulation by doing some preprocessing to the message signal. We transform the 0 bits to carry a value of -1 (bipolar line code), then we multiply it with the carrier signal. With that, we manage to obtain a BPSK modulated signal.

Binary Frequency Shift Keying Modulation (Extra)

In BFSK, we will need 2 different frequencies for transmission. We will be using 10,000Hz and 30,000Hz to modulate our signal. For bit 0, we will transmit the carrier signal with 10,000Hz whereas for bit 1, we will transmit the carrier signal with 30,000Hz. This can be done by generating two OOK signals, one with 10,000Hz and one with 30,000Hz, based on the bit value. For instance, we will generate an OOK signal, which has the 10,000Hz carrier signal when the bit is a 0 and 0 when the bit is a 1. Conversely, we will generate another OOK signal, which has the 30,000Hz carrier signal when the bit is a 1 and 0 when the bit is a 0. We will then add them up and form the BFSK modulated signal.

Demodulation and Detection (OOK and BPSK)

After receiving the transmitted signal, demodulation will be done by implementing 3 parts: multiplication of signal with carrier signal, passing signal through a low-pass filter and sampling and threshold to decode the message into binary data (1s and 0s). We are using coherent detection in both cases of OOK and BPSK, and the ideal block diagrams for demodulation and detection, which is taken from our lecture note is shown below:

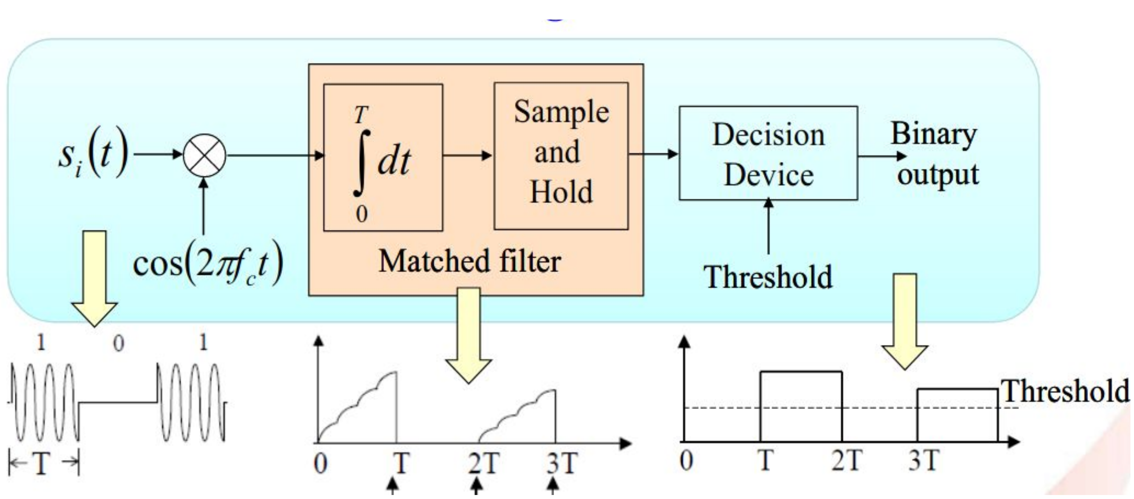

Figure 4. Coherent Detection of ASK

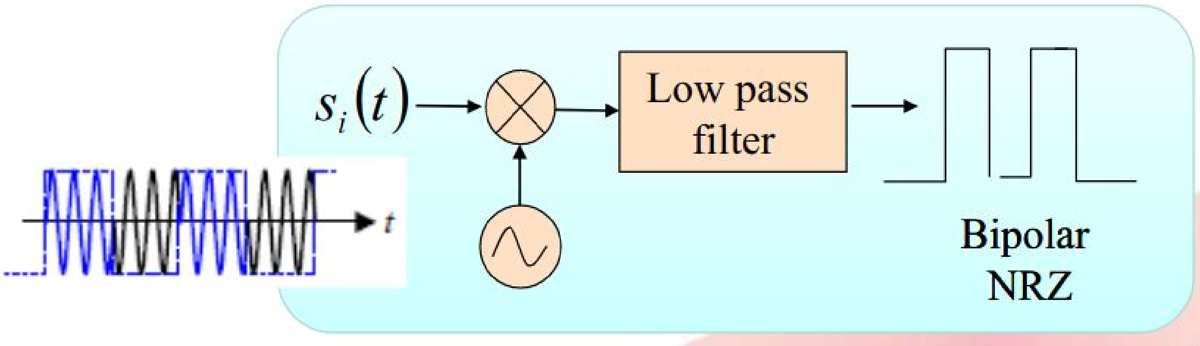

Figure 5. Coherent detection for BPSK

1. Multiplication of Signal

For OOK (*Figure 4*) and BPSK (*Figure 5*), the received signal is multiplied with 2 times the carrier signal as shown below.

$$signal_{squared} = signal_{received} * 2 * signal_{carrier}$$

The purpose of this part is for us to be able to get a constant term (which has low frequency of change), which will be extracted using a low pass filter in part 2, out of the signal. Taking OOK demodulation as an example:

$$A\ cos(2\pi ft) * 5\ cos(2\pi ft) = 5A\ cos^2(2\pi ft)$$

$$= 5A\left[\frac{1 + cos(4\pi ft)}{2}\right]$$

$$= \frac{5A}{2} + \frac{5A}{2}cos(4\pi ft)$$

Since the amplitude of the received signal is affected by noise, we denote the amplitude of the received signal as A. As you can see, after multiplying with the carrier signal, we get a constant term and a cosine term.

2. Low Pass Filter

The multiplied signal is then passed through a 6th order low pass filter with a cut-off frequency of 0.2 (normalised frequency).

The purpose of this part is to extract the constant term as shown in the first part. By passing the signal from part 1 through a low pass filter, we filter out the cosine term and the remaining part is the constant term which has 0 frequency. This constant term can then be used in part 3 to retrieve the original data.

Note that low pass filter and integrator are fundamentally very similar in the way that they attenuate or filter out higher frequency components of the signal. Therefore, in this case, we can safely replace integrators with low pass filters, a heuristic choice of design to simplify our Matlab code.

3. Threshold Decision

Now we have the constant term (DC component) from part 2, we can then use this to retrieve the original data by passing it through a threshold logic. The midpoint of the bits are often selected as the decision threshold for the circuit. This will be discussed further in our experiment and results too.

For OOK, the threshold is set to half of the squared amplitude of the carrier frequency. So, if the amplitude of the signal is higher than 12.5, we take it as a 1, else, it is a 0.

For BPSK and BFSK, the threshold is set to 0, if the amplitude of the signal is positive, we take it as a 1, else it is a 0.

Demodulation and Detection (BFSK - Extra)

Similar to the demodulation process of OOK and BPSK, there are 3 steps to BFSK demodulation and detection, as shown in the *figure 6* below.

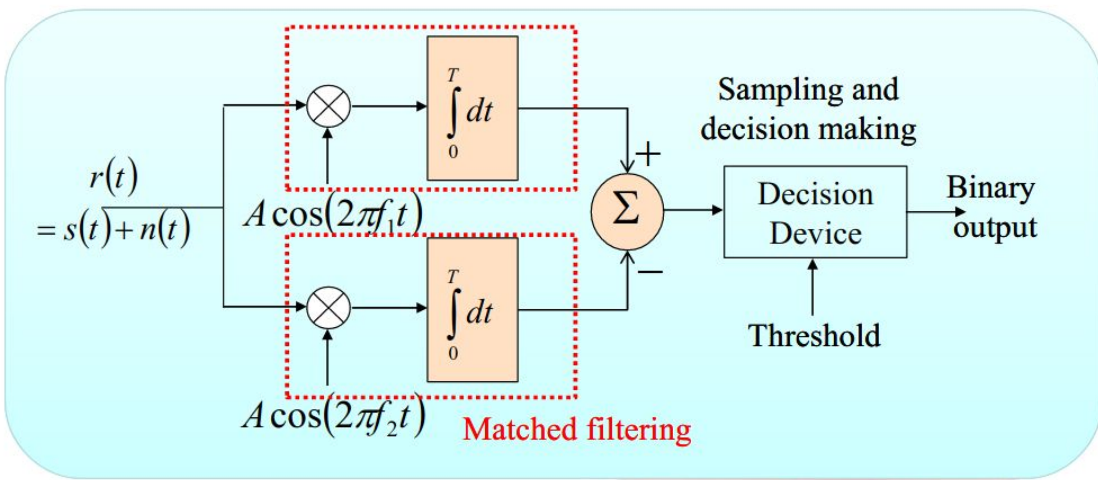

Figure 6. Demodulation and Detection for BFSK

1. Multiplication of signals

For BFSK, the received signal is multiplied with two of the carrier signals that are used in BFSK modulation separately in two different branches. Note that the frequency of carrier signal 1 is 30,000Hz and the frequency of carrier signal 2 is 10,000Hz. This is shown in the block diagram *Figure 6* above of the coherent BFSK detection circuit and the following equations.

$$signal_{squared\_1} = signal_{received} * 2 * signal_{carrier\_1}$$
$$signal_{squared\_2} = signal_{received} * 2 * signal_{carrier\_2}$$

2. Low Pass Filter

On each of the branches, the resultant signals from step 1 are then passed through low pass filters to obtain the low frequency components as discussed in the demodulation process of OOK and BPSK. Similar to them, we may use low pass filters as heuristic replacement for integrators.

Then we combined the low pass filtered output from each of the branches by taking their difference, shown in the equation below. This is used to feed into the threshold device in the next step.

$$signal_{squared} = signal_{squared\_1} - signal_{squared\_2}$$

3. Threshold Decision

With the combined filtered signal from step 2, the binary output is obtained with a sample and threshold device similar to the demodulation and detection of OOK and BPSK above.

Empirical BER Calculation

Once the signal has been reconstructed after demodulation and detection, we calculate the BER at different SNR of the coherent detection techniques of OOK, BPSK and BFSK. This is done by calculating the average BER amongst N noise samples for a SNR value.

By taking the average of N noise samples, we are making use of Central Limit Theorem, which approximates the effect of noise closer to random distribution as more noise samples (or repetitions of adding noise to the modulated signal and detecting them) are factors into the picture. The resultant BER curve will be smoother and tends towards the theoretical BER curve learnt.

Theoretical BER Calculation

As mentioned previously, all demodulation techniques used are coherent demodulation.

The theoretical BER for OOK and BFSK is $P_e = Q\left(\sqrt{\frac{E_b}{N_o}}\right)$ whereas the theoretical BER for BPSK is $P_e = Q\left(\sqrt{\frac{2E_b}{N_o}}\right)$, where E_b is the bit energy and N_o is the noise power spectral density.

The calculation of noise power spectral density, N_o is the same for all modulation techniques, which is $N_o = \frac{N}{2R_b}$ where N is the respective noise power and R_b is the data rate.

The calculation of bit energy, E_b is the same for BPSK and BFSK but different from OOK. Since the probability of bit 1 occurs, P_1 is equal to the probability of bit 0 occurs, and we know that the probability is 0.5, we can deduce that:

$$E_b = P_1E_1 + P_0E_0$$

$$E_b = 0.5E_1 + 0.5E_0$$

OOK derivation

Since OOK modulation technique will only transmit the carrier wave when the bit is 1, and transmit nothing when the bit is 0, we know that $E_1 = \frac{A^2}{2}T$ and $E_0 = 0$ where A is the amplitude of the carrier wave and T is the period of sending 1 bit data. Substituting E_1 and E_2 into E_b, we get:

$$E_b = 0.5\frac{A^2}{2}T + 0$$
$$E_b = \frac{1}{2}\left(\frac{A^2}{2}T + 0\right)$$
$$E_b = \frac{1}{2}\left(\frac{A^2}{2}\frac{1}{R_b}\right)$$

BPSK and BFSK derviation

In BPSK and BFSK modulation techniques, the carrier wave will be transmitted no matter what the bit is. Therefore, the bit energy, E_b is a bit different from OOK.

$$E_b = 0.5\frac{A^2}{2}T + 0.5\frac{A^2}{2}T$$
$$E_b = \frac{1}{2}\left(\frac{A^2}{2}T + \frac{A^2}{2}T\right)$$
$$E_b = \frac{1}{2}\left(A^2\frac{1}{R_b}\right)$$

After we get the bit energy, E_b and the noise power spectral density, N_o, we can get the theoretical BER for all the demodulation techniques. In matlab, we used the erfc function to calculate the theoretical BER.

Since $Q(x) = \frac{1}{2}erfc(\frac{x}{\sqrt{2}})$, therefore we get:

$$Q\left(\sqrt{\frac{E_b}{N_o}}\right) = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{2N_o}}\right) \text{--------- (OOK)}$$
$$Q\left(\sqrt{\frac{E_b}{N_o}}\right) = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{2N_o}}\right) \text{--------- (BPSK)}$$
$$Q\left(\sqrt{\frac{2E_b}{N_o}}\right) = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_o}}\right) \text{--------- (BFSK)}$$

c. Phase 3: Basic error control coding

We implement the different channel coding techniques using MATLAB's encode[1] and decode[2] function. Generator polynomials, corresponding parity matrix and its syndrome table can also be implemented by their own in-built functions.

[1] MATLAB, encode: https://au.mathworks.com/help/comm/ref/encode.html
[2] MATLAB, decode: https://au.mathworks.com/help/comm/ref/decode.html

Hamming

```
hamming_signal = encode(data, 7, 4,'hamming/binary')
data = decode(hamming_signal, 7, 4,'hamming/binary')
```

Cyclic

```
gen_poly = cyclpoly(7, 4)
parity_mat = cyclgen(7, gen_poly)
syn_table = syndtable(parity_matrix)
cyclic_signal = encode(data, 7, 4, 'cyclic/binary', gen_poly)
data = decode(cyclic_signal, 7, 4, 'cyclic/binary', gene_poly, syn_table)
```

The channel coding “encode” segments above are inserted into the codebase for our digital communication system after the data generation step and before any sampling for modulation is done. On the other side of the system (receiver end), the decoding and error correction blocks are added after the demodulation, sample and thresholding blocks. This reflects the block order shown earlier in *figure 1*, where channel coding is to be conducted after source coding and just before modulation. The design of decoding blocks also obey the order of components blocks in the receiver of *figure 1* too where this is after the baseband signal is obtained and just before source decoding happens.

From knowledge in the course, we learnt that the BER for channel encoded signals will be much lower than that of the unencoded signals. Thus, we will be taking a BER average across 200 samples of noise, which allows for a BER semilogy curve that is easier to observe (due to central limit theorem).

4. Experiments and Results

In this section, we will go through the results from the experiments and compare how different methods can affect the result and compare them to their theoretical values calculated using formula.

a. Phase 1: Data generation

In the two *figures 7 and 8* below, we present the result of Bit Error Rate (BER) against different values of Signal to Noise Ratio (SNR) in a step size of 1 and 5. As can be seen from the right *figure 8*, when the SNR is over 10dB, the actual BER drops to 0 as there is no error in the data transmitted. On the left *figure 7*, a plot of similar BER vs SNR is plotted but with a step size of 1, showcasing how BER varies against SNR within the first 10dB.

One deduction we can draw from the two *figures 7 and 8* to explain why there is no error past 10dB when compared to Theoretical BER is that the length of the data is too short for it to have error in the bit content. The BER is so low that the number of possible bits having error is very small and approximating a value of 0. Therefore, in order to better showcase the BER beyond SNR of 10dB, more bits should be used to represent the transmitted data.

Another method we attempted to improve this BER curve was trying to use a larger number of noise samples in our simulation (above 1000 samples) following the useful characteristic of Central Limit Theorem. However, the BER curve produced (with 5dB step size) with this amount of noise samples has not much difference to the plotted *figures 7 and 8* with 20 samples on the right. Therefore, for reducing time complexity of our simulation, we simply use 20 noise samples in our final phase 1 code and conclude that more bits should be used to represent the transmitted data for better 5dB steps BER curve.

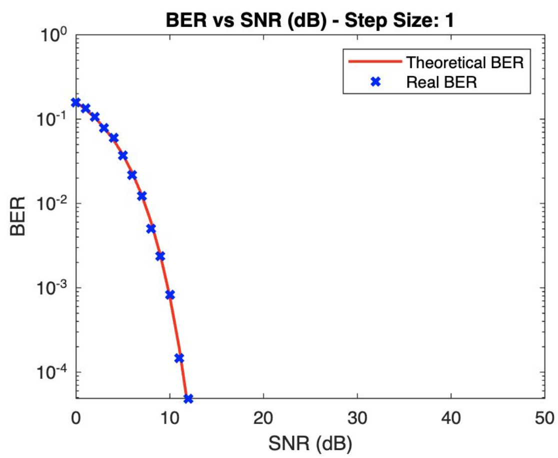

Figure 7: BER vs SNR (Step Size of 1)

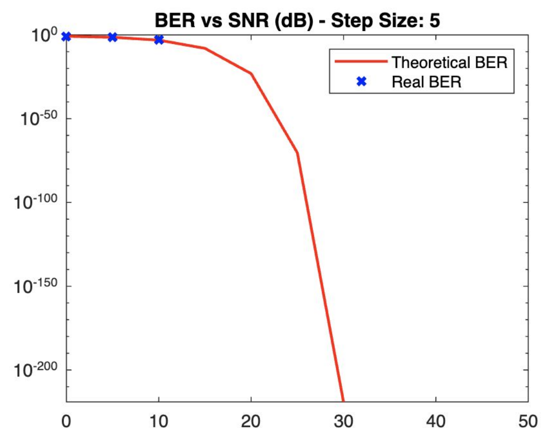

Figure 8: BER vs SNR (Step Size of 5)

We then obtain 2 sets of plots for a noise sample (out of 20 of them), consisting of original generated or transmitted data, the noise sample corrupting it and the received data (before thresholding) for comparison. *Figure 9* shows the set of plots for 5dB while *figure 10* shows the set of plots for 50dB.

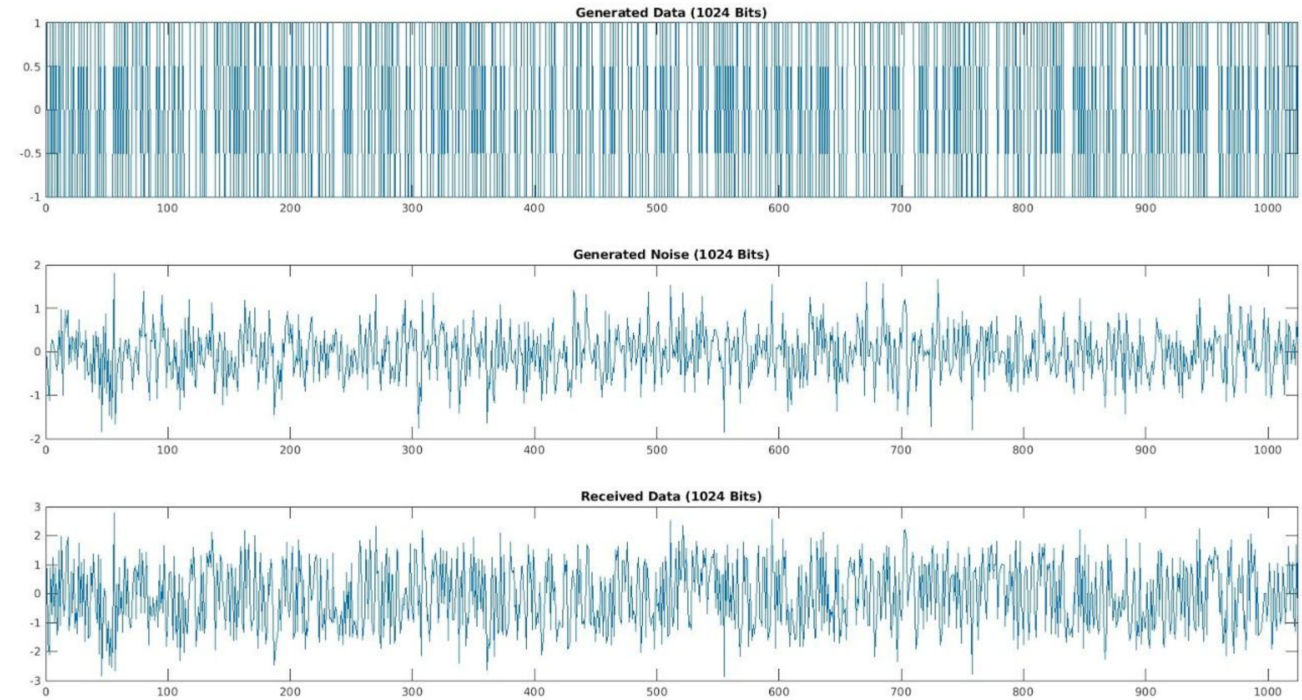

Figure 9. Transmitted, Noise and Noise corrupted signal for 5dB

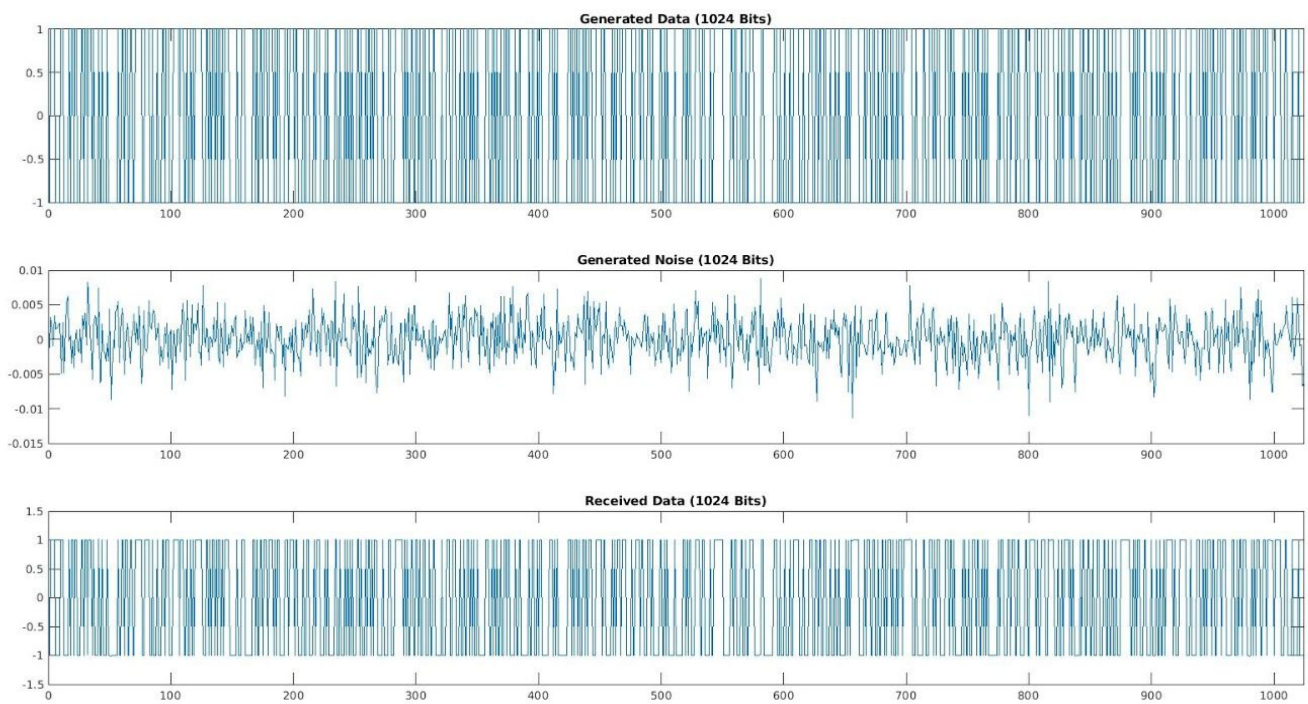

Figure 10. Transmitted, Noise and Noise corrupted signal for 50dB

The *figure 9 and 10* above showcases the transmitted data of length 1024(bits) that is being generated using matlab's *rand* function that ranges from -1 to 1, implying that the data is bipolar encoded.

With signal power S = 1 for both cases, we use *randn* function to generate the noise samples. As expected, both noise samples have mean of 0. The noise variance and standard variation differs at 5 and 50dB. The variance and std at 5dB is about 0.316 and 0.562 respectively (*figure 9*, plot 2). The noise variance and std at 50dB is about 1e-5 and 0.00316 respectively (*figure 10*, plot 2).

The received signal (corrupted with noise) from both 5dB and 50dB has a midpoint or mean of 0 (*figure 9 and 10*). Thus this confirms that the threshold value of 0 as required in the question is logical in detecting the transmitted data from the corrupted signal. We also notice that the amplitude/power contribution from noise in 5dB is to a greater extent than that in 50dB, causing a huge discrepancy between transmitted and received signal (*figure 9*). This explains the trends in *figure 7 and 8* where at 50dB, the bit error rate is very low, negligible while at 5dB, the error rate is of magnitude between 10^0 and 10^-1.

We conclude that AWG noise in communication channels can distort the signal we are trying to transmit (shown in *figure 9* and the BER in *figure 7*). To overcome this, we can send signals with higher signal power (so that the SNR is higher) as shown in very low BER at 50dB (*figure 10*).

b. Phase 2: Modulation for Communication

The figures below show the characteristics of the signals from data generation to modulation to demodulation and detection of each modulation technique implemented, namely: On-Off Keying, Binary Phase Shift Keying and Binary Frequency Shift Keying. These plots are obtained from a particular noise sample out of the N samples we generated in our experiment. The SNRdb value in this case is 5dB which allows us to observe clearly the effect of noise on our signal transmission.

Firstly, the data to be transmitted is generated in a similar manner to that of phase 1. The generated data of 1024 bits is plotted in figure 11. This data is then oversampled at frequency of 160,000 Hz before fed into the modulation blocks.

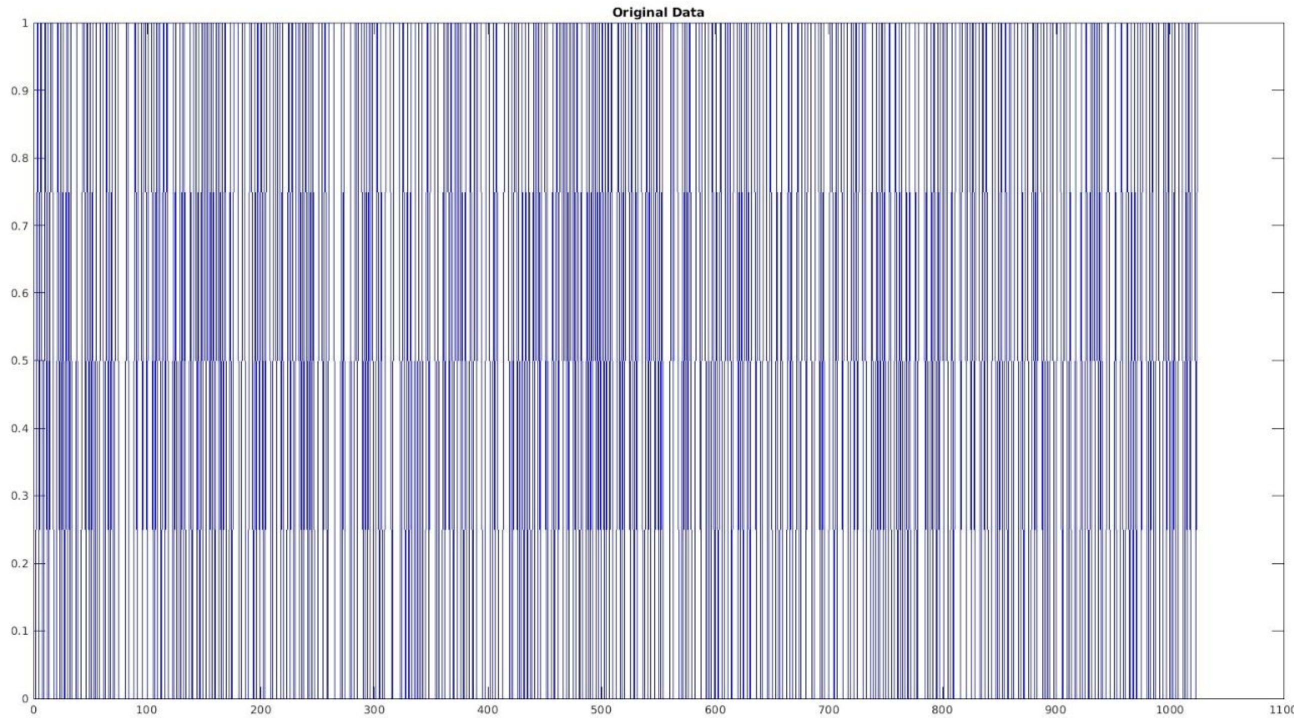

Figure 11. Plot of generated data (1024 bits)

The oversampled signal has more number of bits than the original data. When it is fed into the modulation blocks (OOK, BPSK and BFSK), the modulated results have more number of bits than the original data too. Therefore, we only take a snippet of the first 1000 bits used in transmission (modulated signal to be transmitted across the noisy channel) for plotting in figure 12 below.

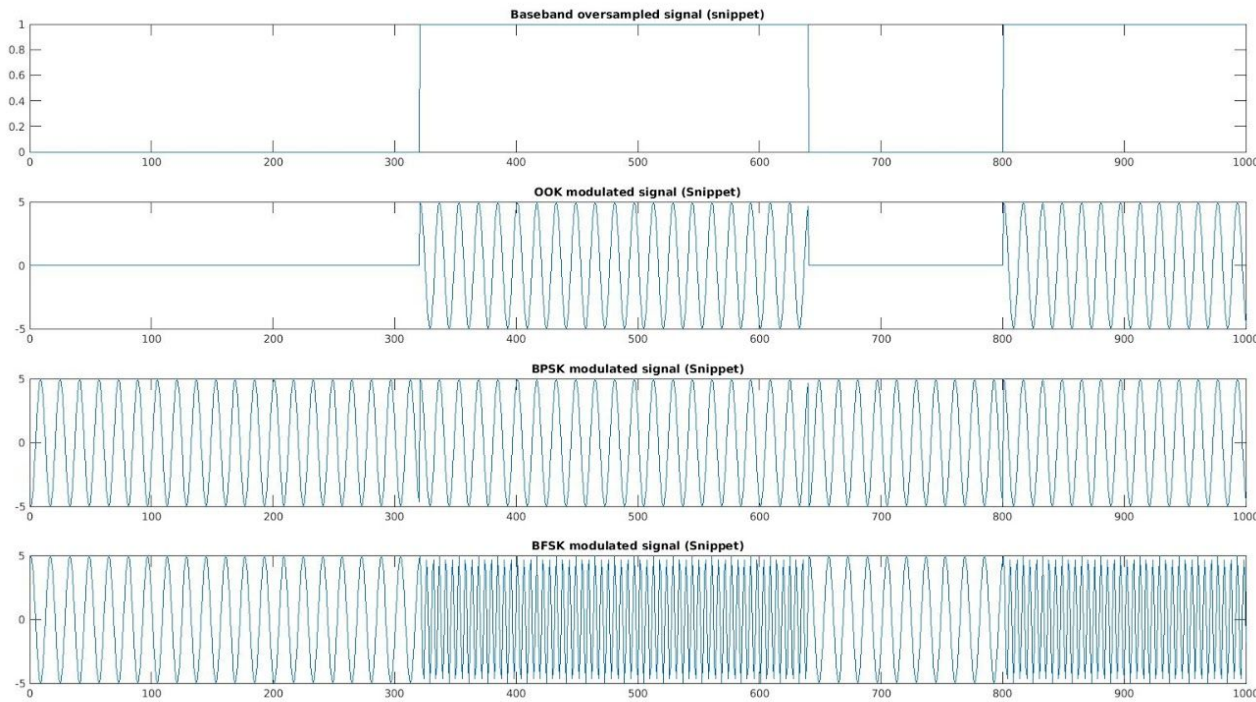

Figure 12. Modulated signal for transmission (snippet of first 1000 bits transmitted)

From figure 12, it can be concluded that the modulation techniques are implemented correctly. For OOK, we can clearly see 0 amplitude is transmitted for 0 sampled data and a carrier signal of amplitude 5 is transmitted otherwise for 1 sampled data. For BFSK, we can observe that a higher frequency = 30,000kHz carrier is transmitted for 1 sampled data and lower frequency = 10,000kHz for 0 sampled data. Both of these carrier signals have amplitude = 5 which is similar to OOK's carrier signal. Lastly for BPSK, We can observe a phase transition of carrier frequency when transition between 0 and 1 in sampled signal is transmitted to the channel. These showed that the modulation of amplitude, frequency and phase in our experiments are correct.

The modulated signal transmitted across the channel will be added with AWG, and the resultant signals on receivers are shown in figure 13. Despite noise corruption from the channel, the received signal still possesses characteristics of their corresponding modulation techniques. For example, we can see the sharp change in amplitude for OOK, the difference in carrier frequencies of BFSK and the phase transition in BPSK. Similar to modulated signals, we will be displaying a snippet of 1000 bits only for ease of observation.

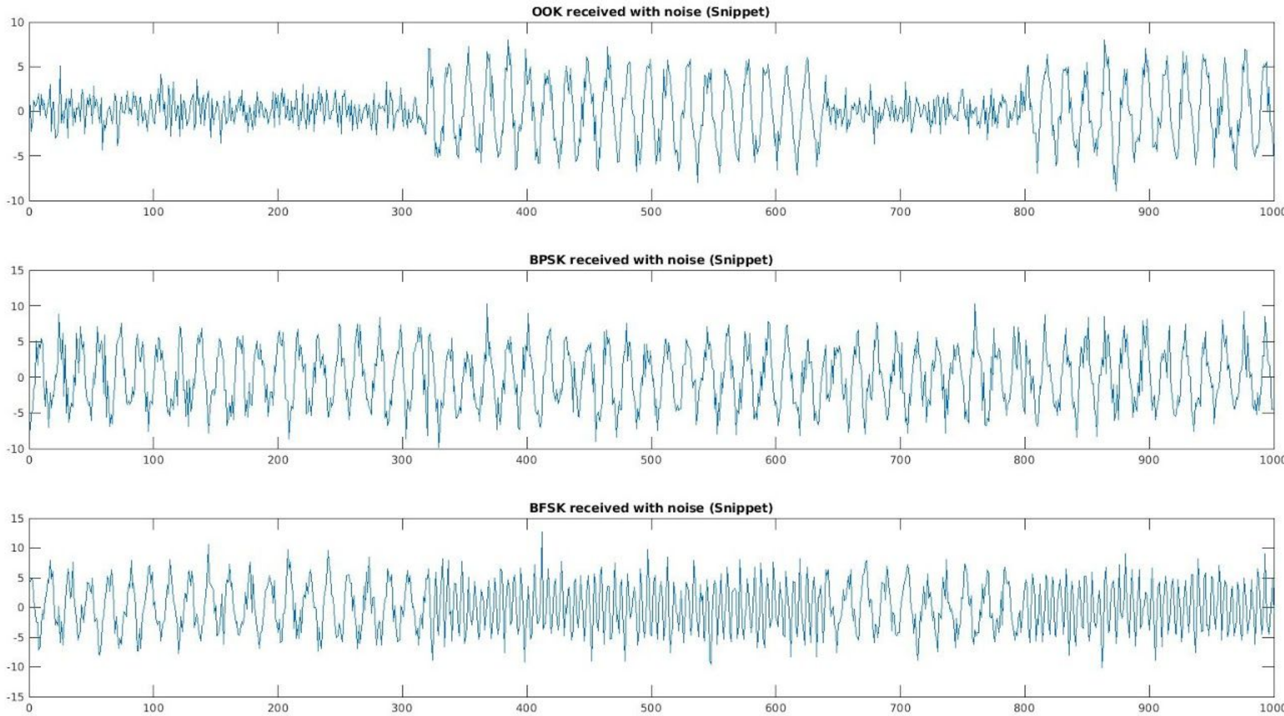

Figure 13. Corrupted modulated signal (a.k.a received signal) snippets of 1000 bits.

In addition, we plot the spectrogram of the transmitted modulated signal and receive the modulated signal in figure 14, 15 and 16. The spectrogram of the signals are plotted using MATLAB's spectrogram function[3]. Spectrograms are visual representations of the spectrum of frequencies of a signal as it varies with time. From a spectrogram we can infer which frequency component has higher energy levels and even how these energy levels vary over time. Spectrogram has 3 dimensions for observation: frequency, time and colour.

[3] MATLAB, spectrogram: https://www.mathworks.com/help/signal/ref/spectrogram.html

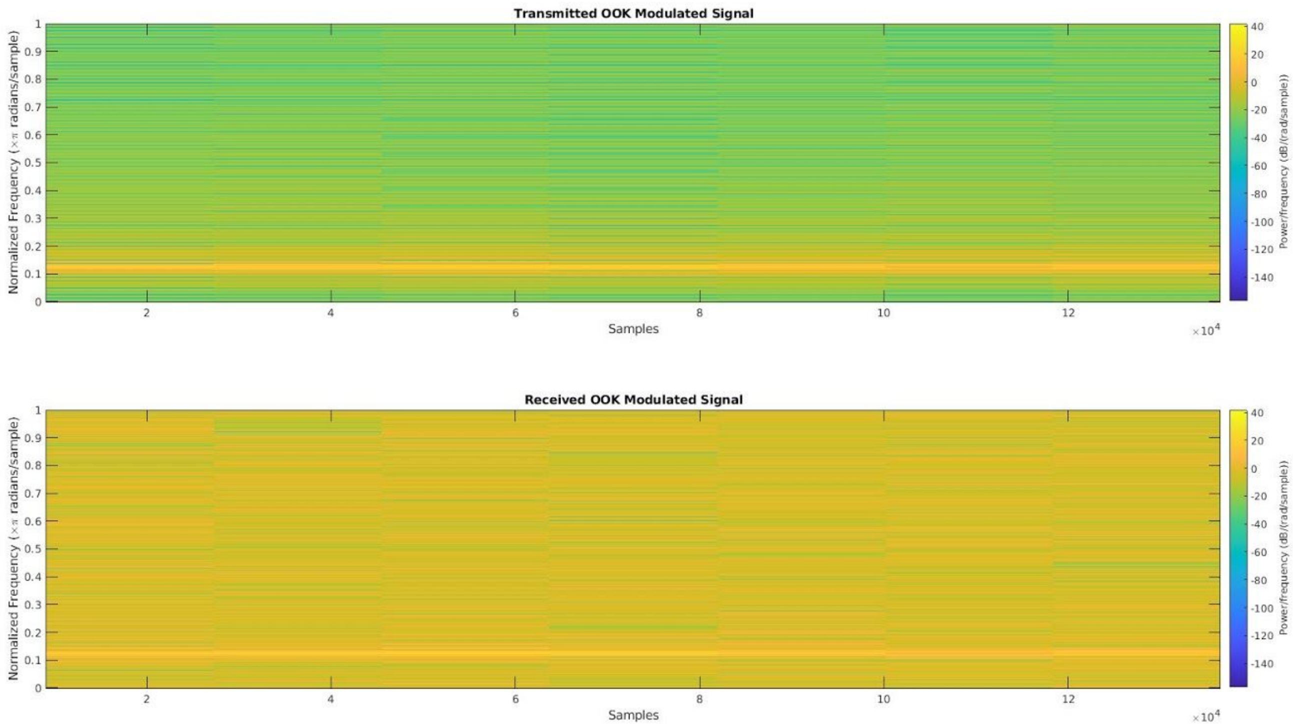

Figure 14. Spectrograms of transmitted and received OOK signal

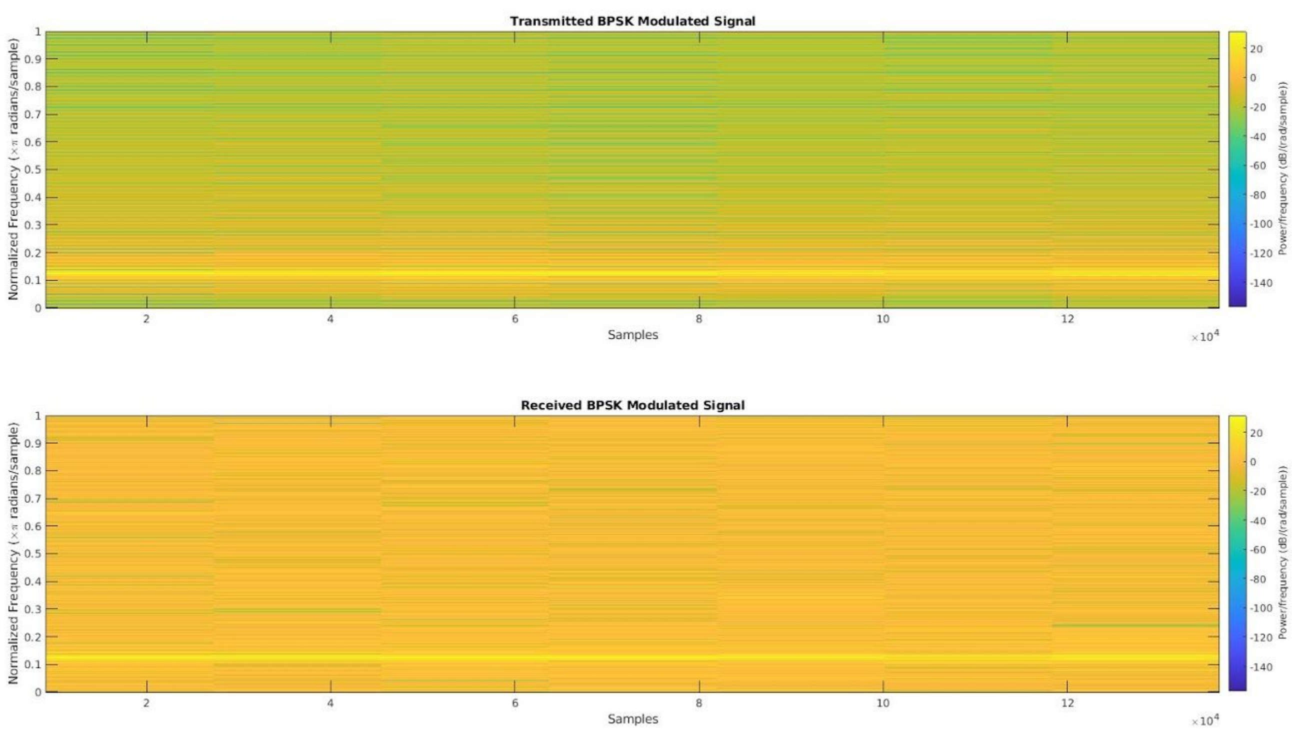

Figure 15. Spectrograms of transmitted and received BPSK signal

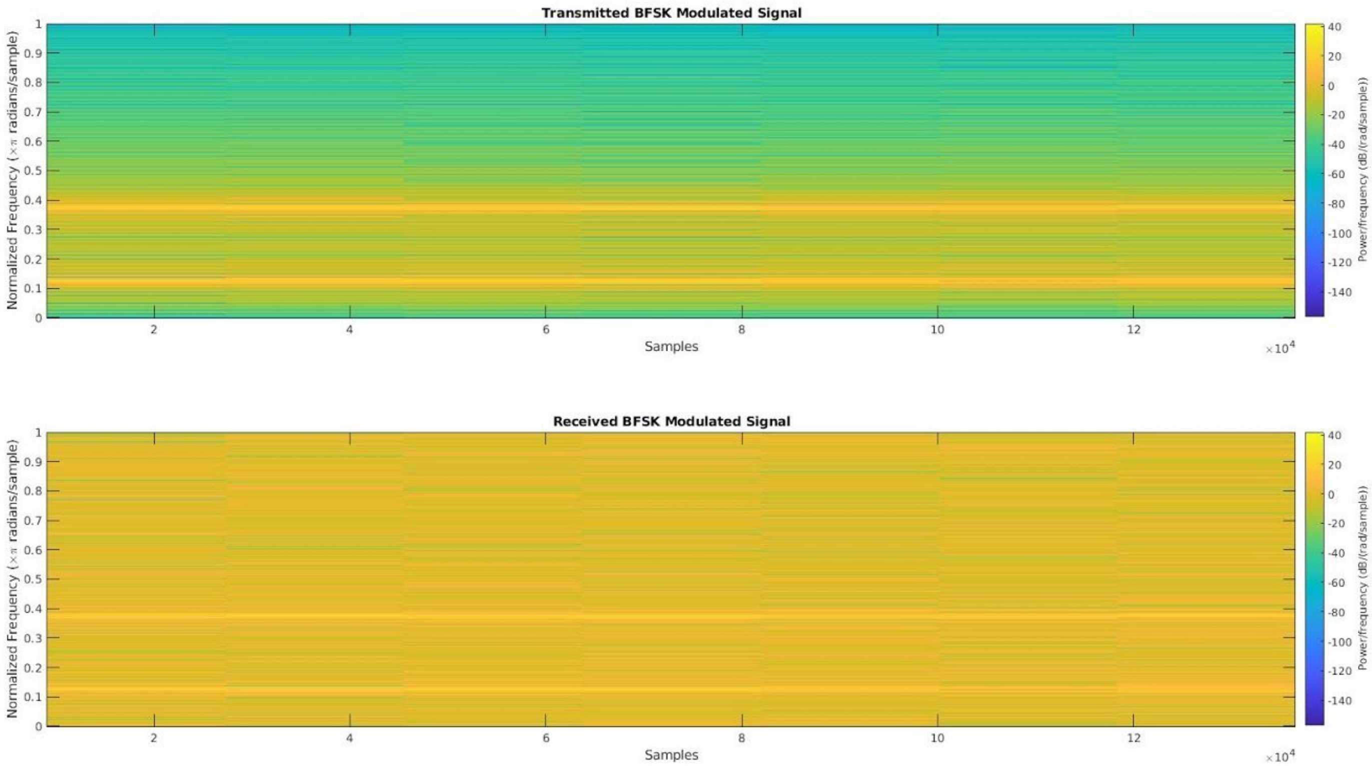

Figure 16. Spectrograms of transmitted and received BFSK signal

Observing the spectrograms for transmitted signals in figure 14, 15 and 16, we can see the energy levels are much higher for the carrier signals' frequency than other frequency components in the spectrum. This is shown by 1 (OOK and BPSK) and 2 (BFSK) bright yellowish lines. Key values for the spectrogram are 0.125□ rads (for OOK, BPSK, BFSK) and 0.375□ rads (for BFSK only), which represent 10,000Hz and 30,000Hz respectively. Another observation is that this energy level is constant with respect to time, as these bright lines are sustained throughout the duration of transmitting signal.

The spectrograms for received signals on the other hand have shown the correct characteristics of Additive Gaussian White noise of the communication channel. "White and additive" characteristic is shown when the spectrum of the received signals are brightened up at every frequency component, making the spectrograms to have bright colour in general as white noise is present for every frequency in the spectrum. Yet as noise is added for every frequency component, we can still observe much brighter lines at key frequencies discussed in the transmitted signal spectrograms. This agrees with our observations from figure 13, where corrupted received signals still possess distinctive characteristics from their modulation techniques (2 bright lines for 2 different carrier frequencies of BFSK for instance).

Moving on, the received signals are demodulated and sampled with the methodology discussed in part 3 Implementations. This reduces the number of samples to 1024 bits, which is similar to the original data size. The demodulated and sampled signals are shown in figure 17 below.

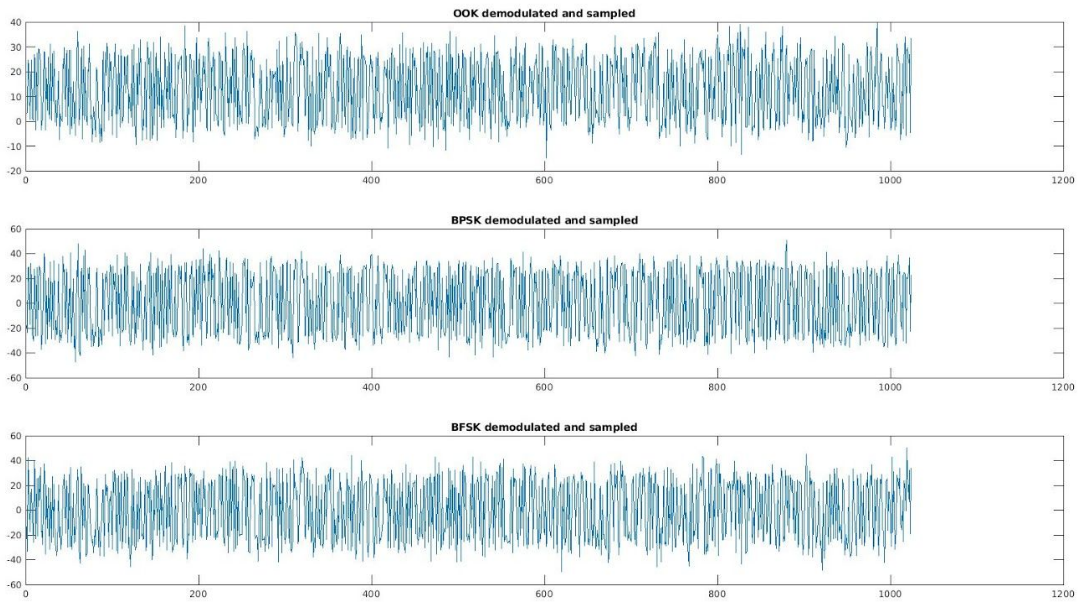

Figure 17. Demod and Sampled signals

By inspection, we can see the midpoints that can be used for thresholding logic are 0 for BPSK and BFSK and half of squared amplitude of carrier signal for OOK. This is agreeable to the theoretical implementation we have chosen and discussed earlier. The result from thresholding logic is the “decoded” or retrieved data, shown in figure 18 below.

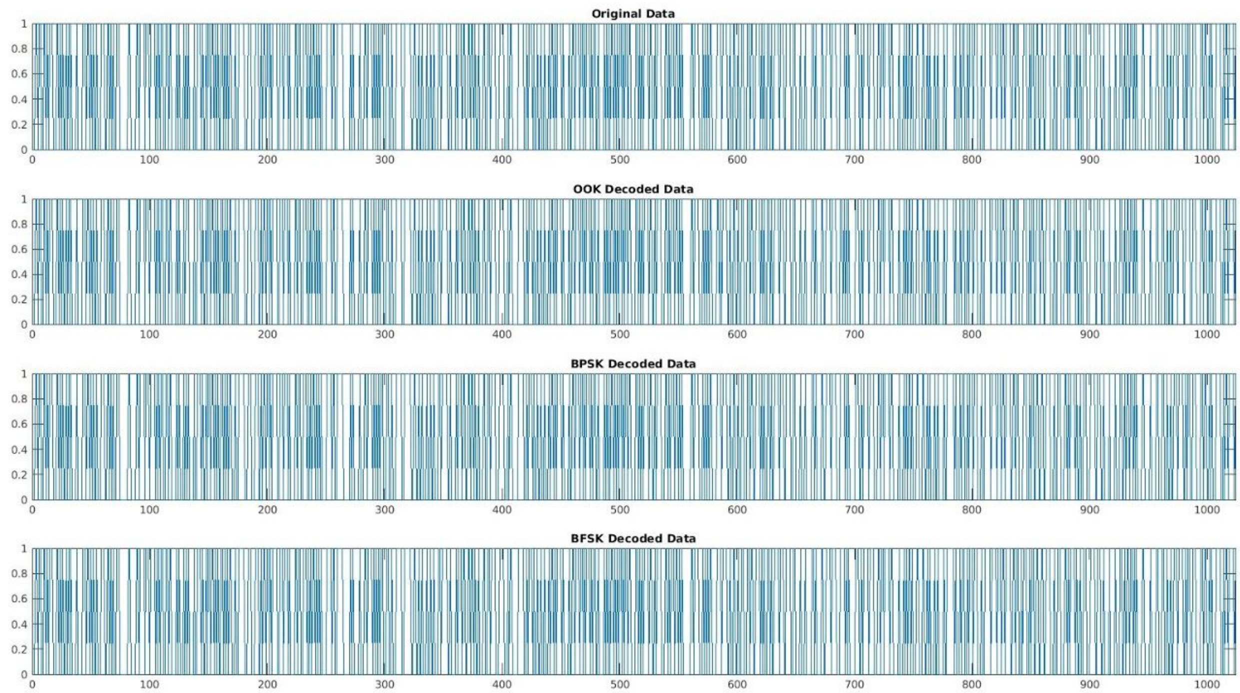

Figure 18. Recovered Data from coherent detection techniques vs the original generated Data

We can see that the general shape of these decoded data are very similar to the original data we wish to transmit to the receiver. This shows that we have achieved a basic, functional digital communication system from Matlab.

For each of the noise samples generated for the range of 0-50dB, we obtain the corresponding BER and calculate the average BER amongst the noise samples, representing the empirical BER for each of the SNR in the range above. Using a semilogy graph, the empirical BER from 0 to 50dB is shown below in figure 19.

Similar to the semilogy in phase 1, data generation, we faced difficulty in plotting the semilogy with SNR increasing in step size = 5dB due to the limitation in size of the data to be transmitted. We have also attempted to conduct the experiments with a larger number of test samples = 1500 and more, which greatly add time complexity to our experiments. Yet the resulting BER semilogy are not much better, and hence we will be using this semilogy with step size = 1dB and running at average of 100 noise samples, which give us an acceptable ber curve and low time complexity of simulation.

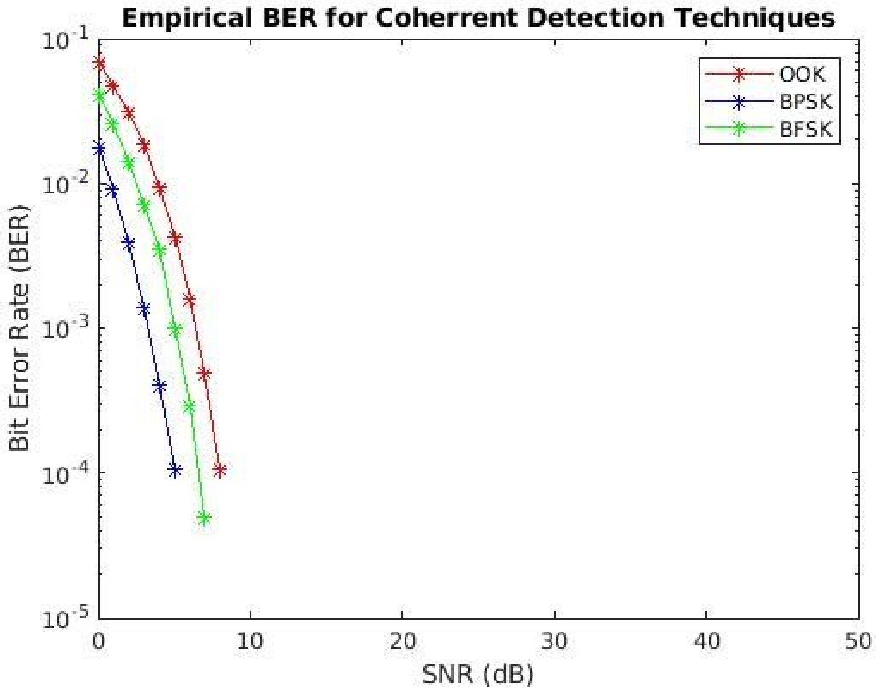

Figure 19. Empirical BER obtained from experiments of phase 2

As expected, BPSK requires less signal power in transmission and detection than that of BFSK and then OOK to achieve the same bit error rate. This is clear as to achieve 10^-3 error rate, BPSK only requires about 5dB, BFSK requires about 8dB and that of OOK is about 9dB. The empirical BER also respects the theoretical BER trends for coherent detection of ASK, FSK and PSK as shown in *figure 20* below. We can see that the scatter points from our experiments are very close to the trends from the theoretical BER plotted.

The empirical BER and theoretical BER trends from *figure 19 and 20* have shown that our digital communication system's demodulation and detection blocks are functional and have very good performance.

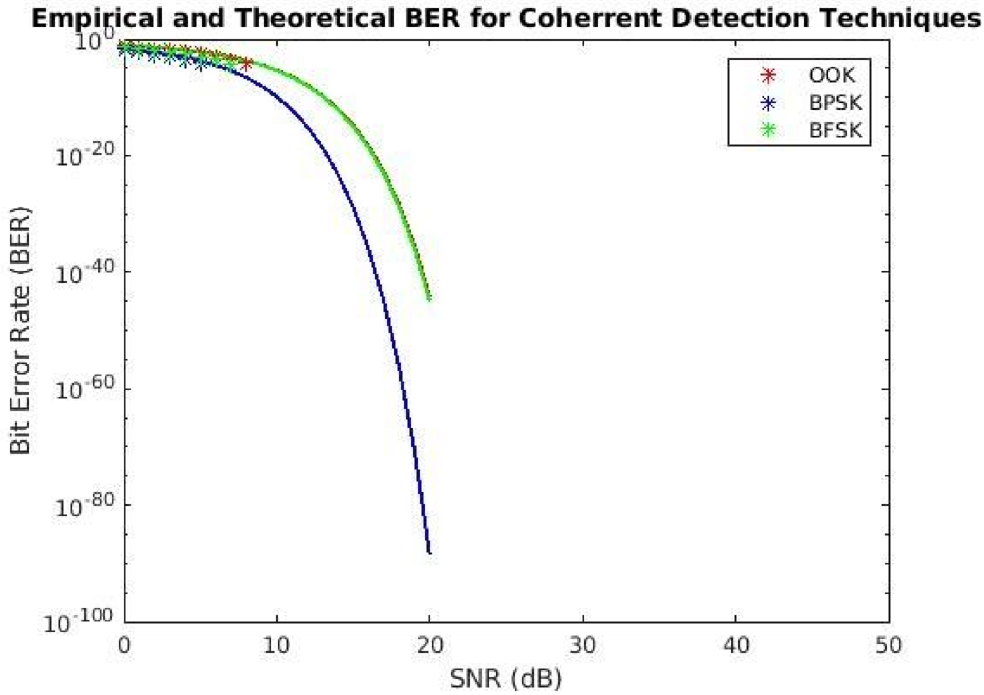

Figure 20. Theoretical and Empirical BER for OOK,BPSK and BFSK.

To sum up, we have attempted to add on bandpass modulation and demodulation blocks on top of baseband blocks to our simple digital communication system in phase 2. While the communication system is functional and we can stop here, we will be adding channel coding blocks in the next phase to enhance the performance in terms of noise tolerance and correction of our software simulated system.

c. Phase 3: Basic error control coding

In phase 3, by adding the channel coding blocks on both transmitting and receiving ends of the communication system, we were able to observe significant reduction in Bit Error Rate for all SNRdB values. As these channel coding and decoding blocks are built on top of part 2 and part 1, the signal at data generation stage, modulation, demodulation, sampling and threshold have similar shape and trends as shown in phase 2 and phase 1 discussion. The only different would be the encoded data before modulation and after demodulation, shown in figure 21 and 22.

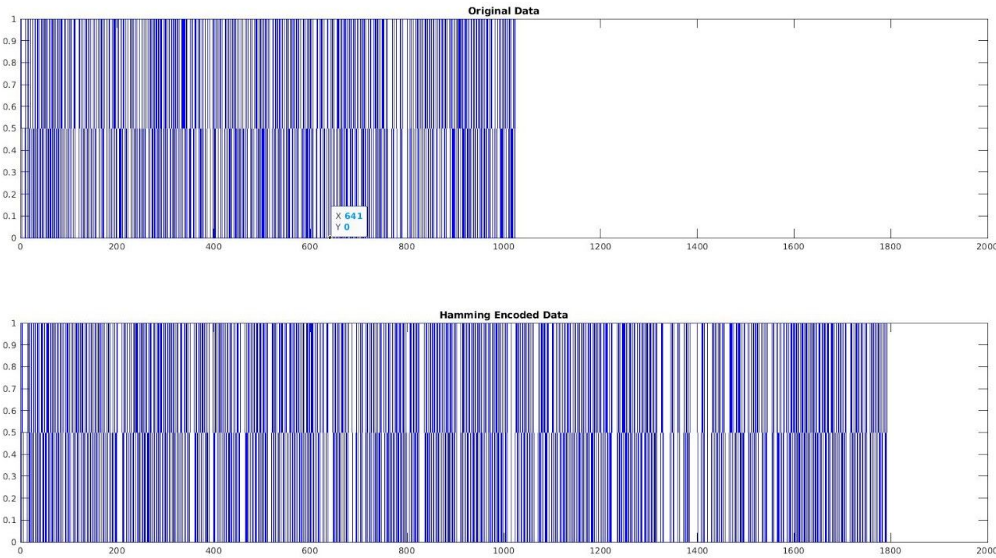

Figure 21. Hamming encoded Data vs Original generated data

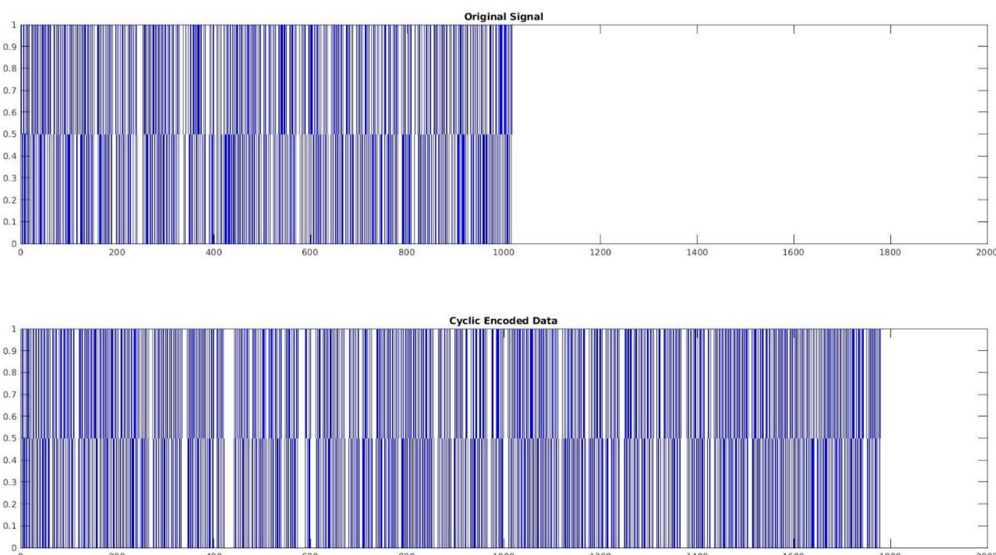

Figure 22. Cyclic encoded Data vs Original generated data

It can be observed that the encoded signal is longer than the original generated data as we have added some redundancy or parity check bits into the signal before feeding it for modulation and transmission, which supposedly improve its BER performance.

In the 2 figures 23 and 24 below, the improvement that channel coding blocks brings to the overall BER of the digital communication system can be seen. The higher the SNR, the larger the improvement gained in using channel coding. Comparing figures 23 and 24 to figure 19, we can see that the encoded signal in both hamming and cyclic channel coding has led to significant improvement in BER performance. While it needs approximately 5, 8 and 9dB approximately for the unencoded OOK, BPSK, BFSK signals to reach 10^-3 bit error rate (figure 19), the observed required SNR for both hamming and cyclic encoding signals (OOK, BPSK and BFSK correspondingly) are less than 5dB to achieve this BER level (figure 23 and 24).

The general BER trend of OOK, BPSK and BFSK are conserved with channel coding however, with BPSK requiring the least SNRdb, followed by BFSK and lastly OOK to achieve an arbitrary bit error rate level (figure 23 and 24). BPSK has the greatest boost in error rate performance as it only has 1 point on the empirical semilogy plot at 0dB, implying the remaining range of dB, it has negligible error rate. The BER semilogy curve, again, is limited in experiment due to the small data size we are using in our experiment.

We attempted to plot the unencoded OOK signal BER alongside with the Hamming and Cyclic encoded OOK signal in both figure 23 and 24.. To generalise, we may see about 7-10dB difference between the encoded and decoded version in achieving the same BER, proving our point that channel coding can significantly reduce error in transmission through channel corrupted with AWG noise.

The improvement can be explained due to the additional bits that are brought over by the encoding of the signal. For every 4 bits, we instead use 7 bits to represent it. With a 7 bit code word representing a 4 bit message, this enables us to correct errors up to 2 bits and allow error detection of 3 bits. By adding redundancy and parity checking from channel encoding, we have shown that the empirical BER performance improves significantly.

However, adding in channel encoding will result in poorer bandwidth efficiency due to reduction in data rate (meaningful data). This is because out of 1792 bits of data transmitted, it only represents 1024 message bits, whereas the rest of the data bits are all redundancy and parity checks which serve to reduce the error rate in transmission. This effectively means the transmission is wasting some bandwidth for redundancy, which explains why channel coding is exchanging this bandwidth resource to achieve higher error tolerance. A balance between bandwidth efficiency and error rate given certain signal power should be considered in designing digital communication systems to achieve our specific needs.

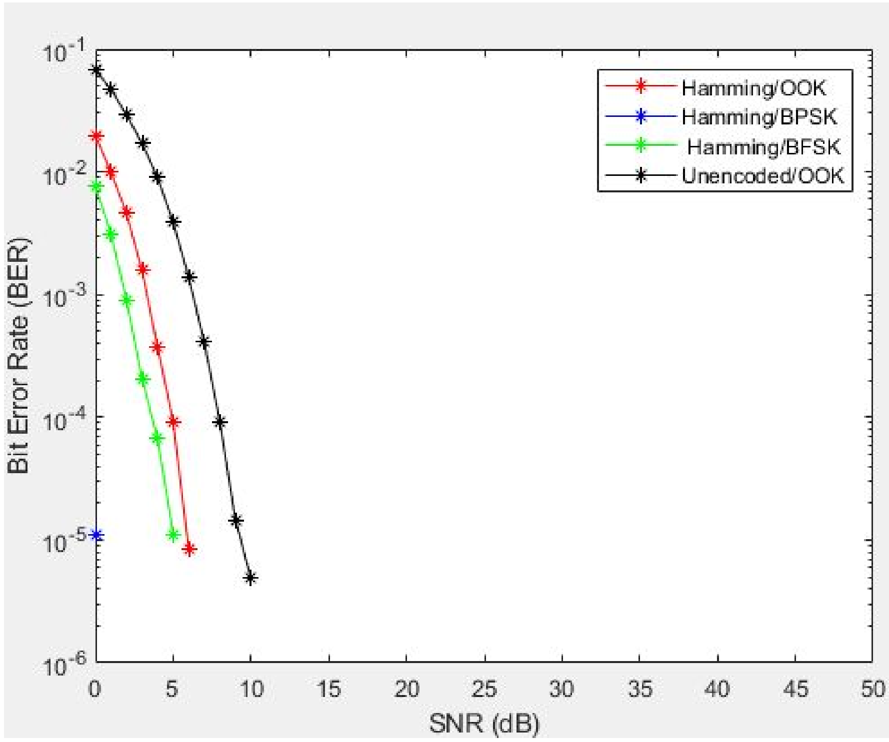

Figure 23. Hamming encoded vs Unencoded BER

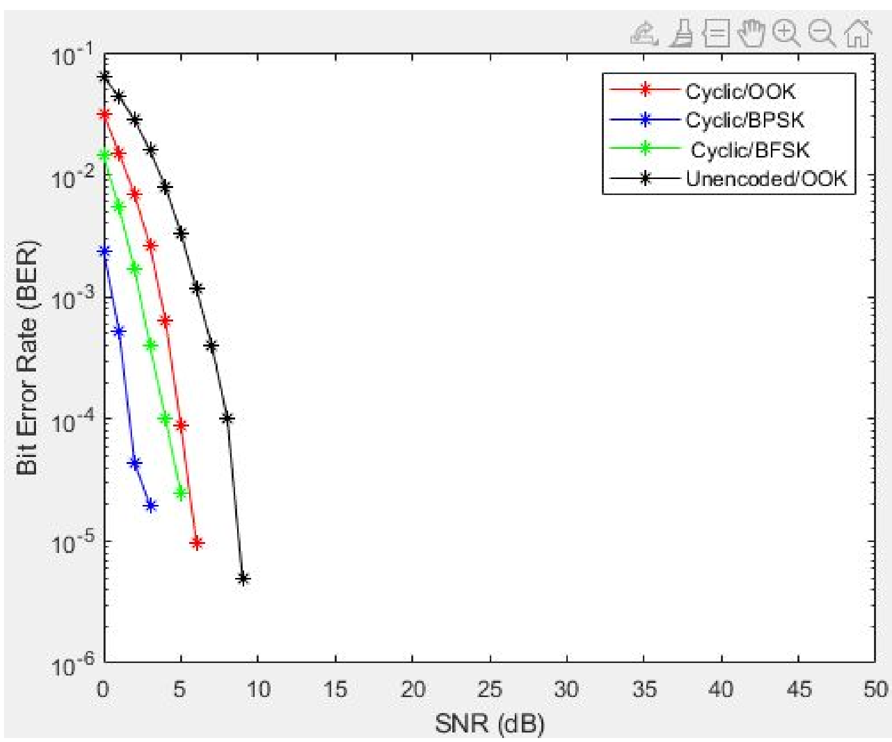

Figure 24. Cyclic Encoded vs Unencoded BER

5. Conclusion

In this study, we have successfully built a basic digital communication system in Matlab and we also looked at three important stages as well as compared the performance of BER under different settings. In phase 1, we simulated data transmission by generating noises based on different values of SNR and studied how noise can affect and corrupt the received data. Subsequently in phase 2, we employed different modulation techniques (OOK, BPSK, BFSK) to transmit signals in the bandpass region of the channel and compared the BER performance across them. To further reduce the error introduced by the noisy channel, we applied channel encoding in phase 3 to provide error detection and correction ability in the receiving end of the channel, the two encoding techniques implemented here are **Hamming Code** and **Cyclic Code**.

Our study indicates that modulation and channel encoding is able to increase the performance of BER when compared to the vanilla implementation of data transmission in phase 1. In phase 2 and 3, we presented and compared the performance when the signal is modulated and we can observe that BPSK performs the best out of all the modulation techniques with BFSK coming in second. In phase 3, implementations with either hamming and cyclic code perform better than those that are modulated but unencoded. In particular, the implementation with BPSK Modulation along with Channel Encoding performs the best out of all the implementations.

Lastly, while our implementations were able to reduce the BER of the signal by encoding the message bits with redundant bits for error detection and correction, we believe that bandwidth efficiency can be further increased by having source encoding implemented, in order to maximize the entropy (optimal code length) while transmitting at the maximum data rate with a small error rate. In conclusion, various approaches have been worked on to reduce any error that could happen during a transmission and in this study we were able to showcase that BER can be reduced by employing these approaches we learned in this course.

Appendix

The following section describes the necessary steps to replicate the results and plots attached in each phase of the study. All necessary functions and codes for three different phases are included in the zip file submitted for grading purposes. The replicated results and plots might not be the exact same as the ones attached in this report as the data and noise are generated randomly but the outcome of the execution should behave in a similar manner to the ones shown in this report.

Common Functions Definition

- convert_dB_to_dec.m
- convert_dec_to_dB.m
- generate_SNR.m
- generate_data.m
- generate_noise.m
- get_error_rate.m
- sample_and_threshold.m

Phase 1

- Execute the file ***phase_1.m*** to generate *Figure 7 and Figure 8* in MATLAB*.*
- Execute the file **phase_1_plot_at_5dB.m** to visualize the generated data, noise and received data as shown in *Figure 9 and Figure 10.*
- The figures should be displayed in multiple windows within MATLAB.

Phase 2

- Execute the file ***phase_2.m*** to generate *Figure 11-20* in MATLAB*.*
- The figures should be displayed in multiple windows within MATLAB.

Phase 3

- Execute the file ***phase_3_hamming.m*** to generate *Figure 21 and 23* in MATLAB*.*
- Execute the file ***phase_3_cyclic.m*** to generate *Figure 22 and 24.*
- The figures should be displayed in multiple windows within MATLAB.