

**CE3006 Digital Communications Project Report**

Sim Song Qin (U1820620J)

Poh Jun Rong (U1821563B)

Chan Geng Tat (U1821992D)

Tay Kee Kong (U1822051G)

Thin Lat Han (U1822997F)

School of Computer Engineering Nanyang Technological University

5th November 2021

# Table of Contents

[**Table of Contents**](#_heading=h.gjdgxs) **2**

[**Introduction**](#_heading=h.30j0zll) **4**

[Background](#_heading=h.1fob9te) 4

[Aim & Objectives](#_heading=h.3znysh7) 5

[Phase 1: Data Generation](#_heading=h.2et92p0) 5

[Phase 2: Modulation for communication](#_heading=h.tyjcwt) 6

[Phase 3: Basic error control coding to improve the performance](#_heading=h.3dy6vkm) 6

[**Phase 1**](#_heading=h.1t3h5sf) **7**

[Implementation](#_heading=h.4d34og8) 7

[Data Generation](#_heading=h.2s8eyo1) 7

[Noise Generation](#_heading=h.17dp8vu) 7

[Noise Variance](#_heading=h.3rdcrjn) 7

[BER Calculation](#_heading=h.26in1rg) 8

[Experiments and Results](#_heading=h.lnxbz9) 8

[**Phase 2**](#_heading=h.35nkun2) **13**

[Implementation](#_heading=h.1ksv4uv) 13

[Carrier signals](#_heading=h.44sinio) 13

[Sampling](#_heading=h.2jxsxqh) 13

[Noise & SNR](#_heading=h.z337ya) 14

[Structure of Phase 2 codes](#_heading=h.3j2qqm3) 14

[On-Off Keying (OOK)](#_heading=h.1y810tw) 14

[Binary Phase Shift Keying (BPSK)](#_heading=h.4i7ojhp) 15

[Binary Frequency Shift Keying (BFSK)](#_heading=h.2xcytpi) 16

[Experiments and Results](#_heading=h.3whwml4) 17

[Bit Error Rate Performance](#_heading=h.2bn6wsx) 17

[Data Waveform](#_heading=h.qsh70q) 19

[Modulated signals](#_heading=h.3as4poj) 20

[Received signals](#_heading=h.1pxezwc) 21

[Demodulated signals](#_heading=h.49x2ik5) 22

[Decoded signals](#_heading=h.2p2csry) 23

[**Phase 3**](#_heading=h.147n2zr) **24**

[Implementation](#_heading=h.3o7alnk) 24

[Cyclic and Hamming Code](#_heading=h.23ckvvd) 24

[Linear Code](#_heading=h.ihv636) 25

[Experiments and Results](#_heading=h.32hioqz) 25

[Cyclic Original vs Encoded Data Waveform](#_heading=h.1hmsyys) 26

[Cyclic Bit Error Rate Performance](#_heading=h.41mghml) 27

[Hamming Original vs Encoded Data Waveform](#_heading=h.2grqrue) 28

[Hamming Bit Error Rate Performance](#_heading=h.vx1227) 29

[Linear Original vs Encoded Data Waveform](#_heading=h.3fwokq0) 30

[Linear Bit Error Rate Performance](#_heading=h.1v1yuxt) 31

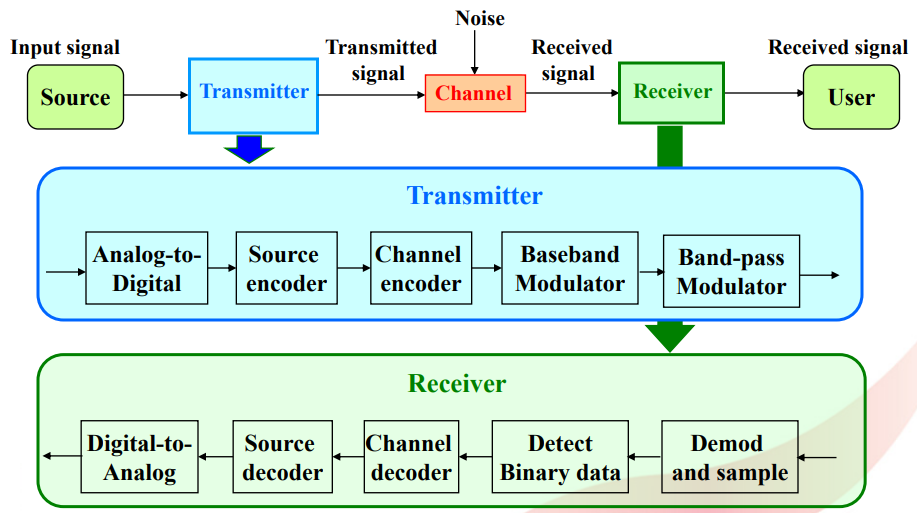
[**Conclusion**](#_heading=h.4f1mdlm) **32**

# Introduction

## Background

Analog communication is becoming obsolete in this technological era with the ever-growing pace of digital technology. Analog communication is the use of continuous time-varying signals only for the transmission of information from the sender to the receiver. Whereas digital communication leverages the usage of digital technology for sending information over a channel. Digital communication provides various key advantages over analog communication such as the immunity to noise and distortions due to the larger signal-to-noise ratio (SNR), the hardware required is more flexible and less complicated than an analog system and generally requires lesser power as it uses higher bandwidth.

In theory, a digital communication system consists of a transmitter, channel, and receiver. Analog input signal is passed through a series of processing steps where it is first converted to a digital signal then encoded and modulated within the transmitter to ensure efficiency and reliability during transmission. The transmission signal will be passed through a communication channel where some inevitable channel noise will be introduced due to the physical medium property of the channel. Therefore, the received signal at the receiver would consist of both the transmitted signal and noise added together. A series of processing steps each corresponding to a reverse process to that in the transmitter will be conducted to remove the noise, demodulated, and decoded to get back a signal which is very close to the original input signal. A visual representation of what is explained can be seen in Figure 1 below.



*Figure 1. Model of a digital communication system*

## Aim & Objectives

This project aims to implement a basic digital communication system using MATLAB and analyze the performance of the system with different SNR values.

This project has been split into 3 phases:

### Phase 1: Data Generation

This phase corresponds to baseband modulation & demodulation, analog to digital conversion & vice versa at the transmitter and receiver. Different noise levels will be generated and added to simulate the transmission through a channel.

### Phase 2: Modulation for communication

This phase corresponds to band-pass modulation & demodulation which are added on top of phase 1 baseband processing. Different kinds of band-pass modulation methods are implemented, and the performance is analyzed.

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

This phase corresponds to channel encoding and decoding which are added on top of both phase 1 and phase 2 to further increase reliability in the transmission of signals. Different kinds of channel coding methods are implemented, and the performance is analyzed.

# Phase 1

## Implementation

### Data Generation

We assume the number of bits to be transmitted to be 1024. The data values will be either -1 or +1. This can be achieved by using the *rand* function, which returns a single uniformly distributed random number in the interval (0,1). Then, the *round* function is used to round the numbers to 0 or 1, multiply them by 2 and subtract by 1, which will eventually give us the bit values of either -1 or +1. These 1024 bits generated will be our data for transmission.

### Noise Generation

The next step is to generate the same number (1024) of bits of noise samples using the *randn* function. The noise generated will have a normal distribution with zero mean and unit variance.

### Noise Variance

We will change the noise variance concerning the SNR (signal to noise ratio) value. By using the equation, SNR = 10log10(S/N), where S is the signal variance and N is the noise variance. We fix the SNR value to be 10. Signal will have unit power, that is S=1. We can then obtain the value of N by substituting the values into our SNR equation. Therefore, the noise with this new variance will be. is the noise with unit variance, is simply the noise power.

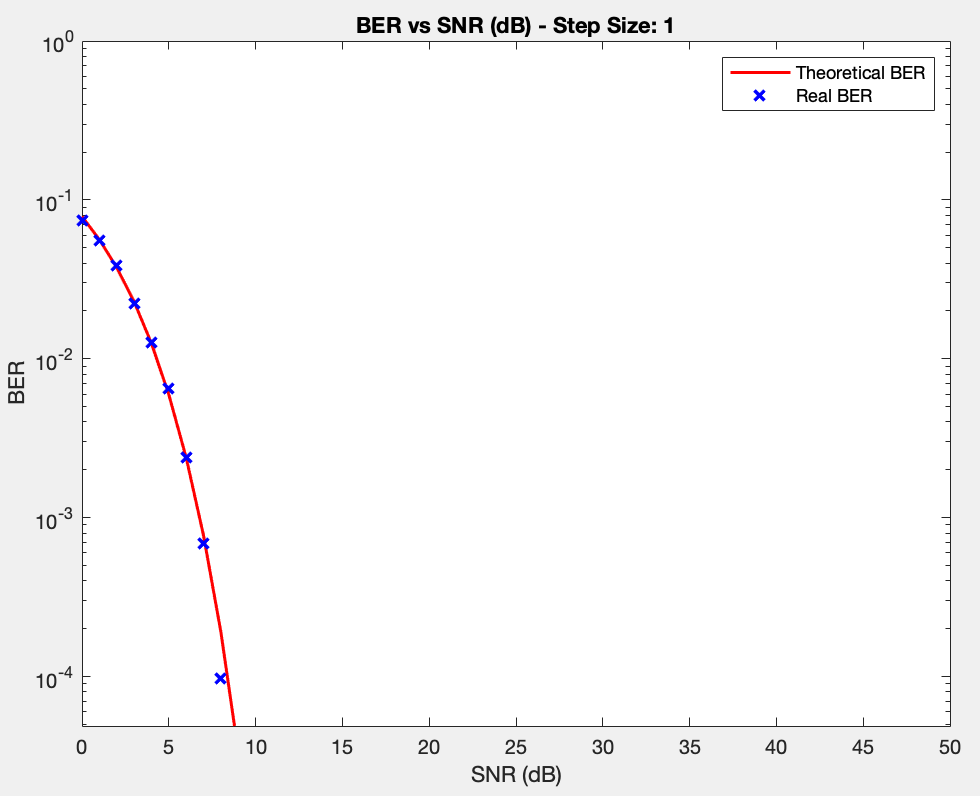
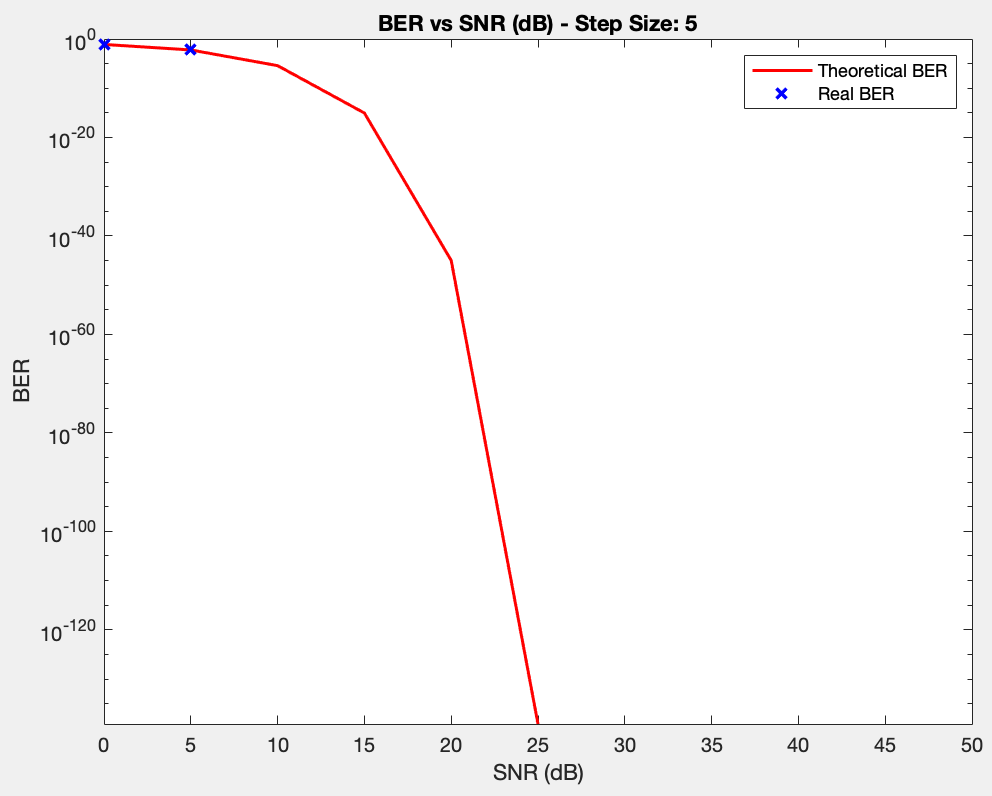
The received signal will be the transmitted data added with the noise samples. We then pass this signal to a threshold logic to determine whether the received signal is a +1 or -1.

### BER Calculation

The output from the threshold logic can then be compared with the input data to calculate the BER. All steps from noise generation to this part will be repeated 20 times to get an average of BER. As the noise is a Gaussian random process that obeys the central limit theorem, repetitions are made such that we can get a smoother curve of BER as the noise will then be approximately closer to the random distribution curve. The theoretical SNR is also calculated.

## Experiments and Results

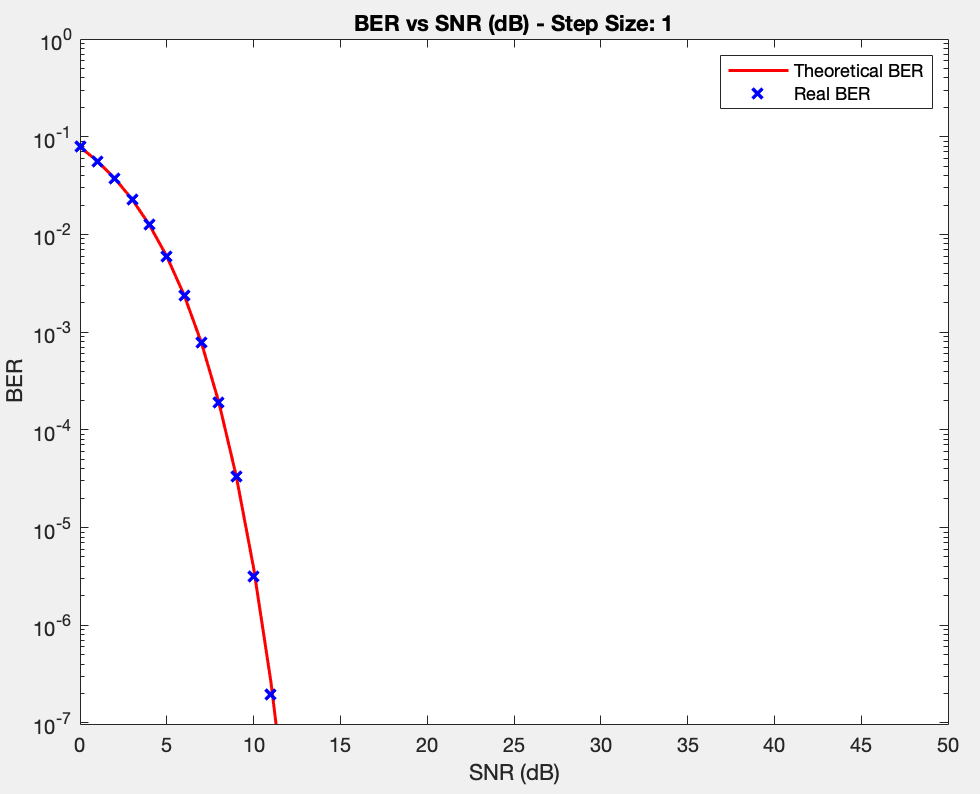
The two figures below show the result of bit error rate (BER) against signal-to-noise (SNR) ratio in step size of 1 and 5 respectively. By observation from Figure 2, we can tell that no error occurs beyond 10dB. By decreasing the step size from 5 to 1, we can see closer how BER varies against SNR within 0 to 10 dB, this graph is shown in Figure 2.

*Figure 2: SNR vs BER (Step Size of 1) Figure 3: SNR vs BER (Step Size of 5)*

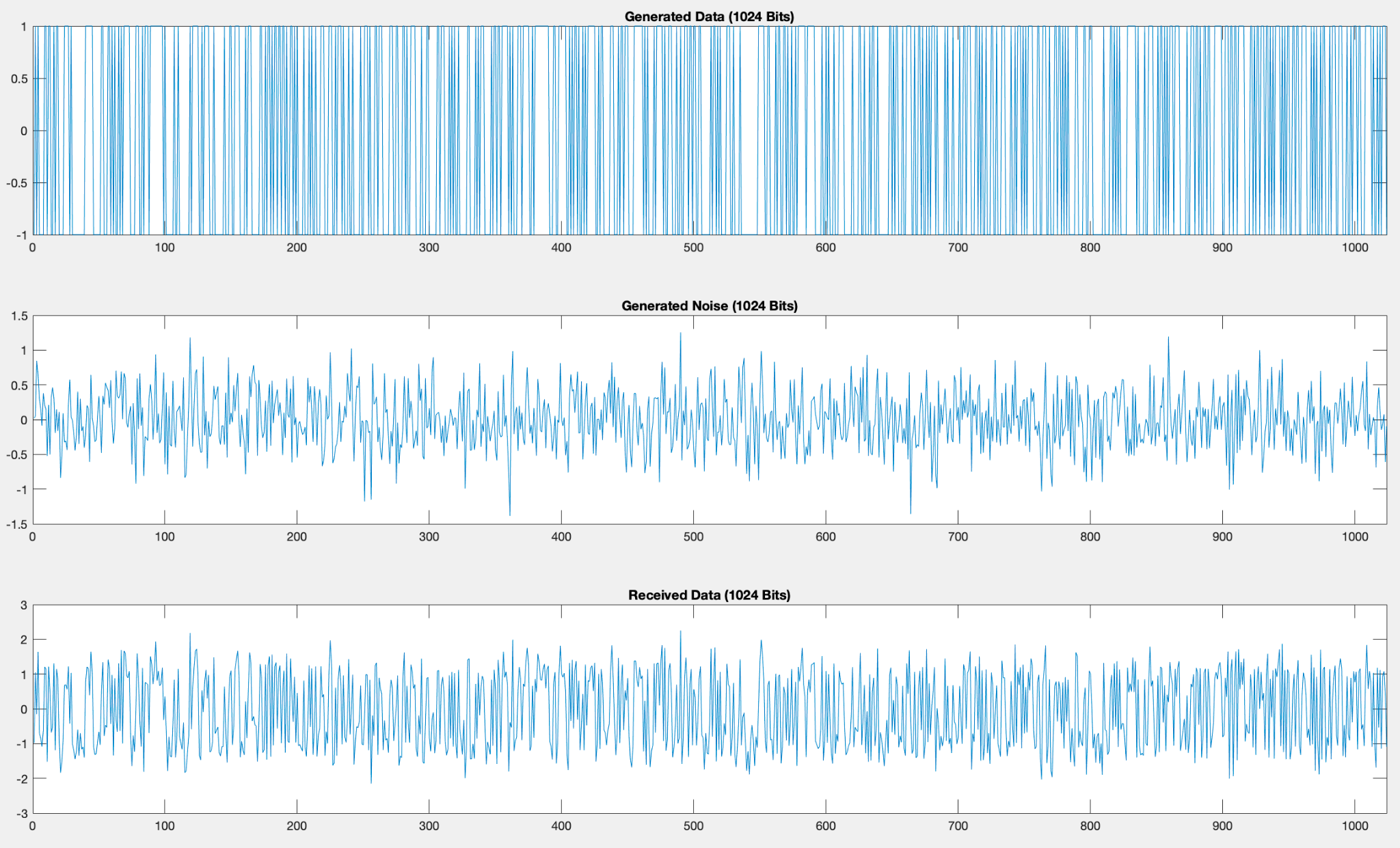
One possibility where there is no bit error (real BER) after 10 dB is the length of the data is too short, hence no error is contained within the data. We can see that the theoretical BER is relatively small and that the number of possible bits having the error is approximately zero. Hence, more bits should be used to represent the data if we want a better visualization of the performance of BER after 10 dB.

To improve the performance of the BER curve, we also tried to increase the number of noise samples taken (more than 10k in our experiment). As we can see from Figure 4, the performance did improve as compared to just 20 samples taken but not much, and the time taken to execute the code is longer as compared to only 20 samples taken. Therefore, instead of increasing the number of samples taken, we think that increasing the data length might be a better method to take to visualize a better SNR vs BER curve.

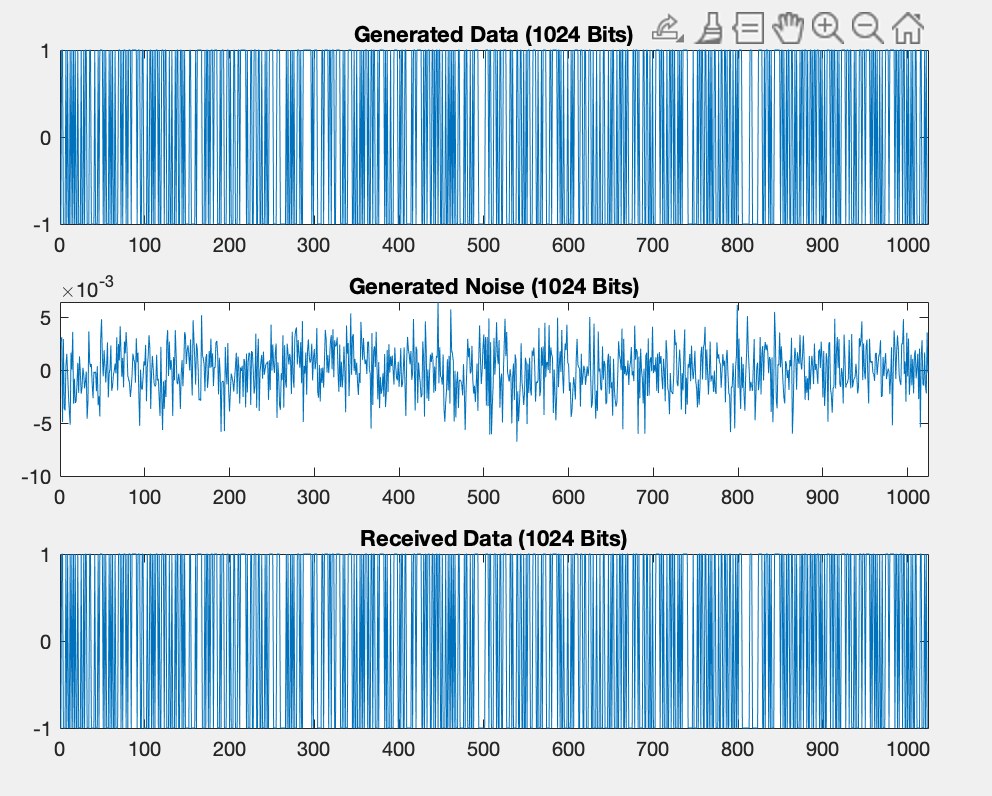


*Figure 4: SNR vs BER (Step Size of 1)*

We then produce 2 sets of plots for a noise sample out of the 20 noise samples, where each plot consists of the original generated or transmitted data, the noise sample corrupting it, and the received data (before thresholding) for comparison. Figure 5 shows the set of plots for 5dB while Figure 6 shows the set of plots for 50dB.



*Figure 5: Transmitted, noise, and received signal at 5 dB respectively*



*Figure 6: Transmitted, noise, and received signal at 50 dB respectively*

The transmitted data of 1024 bits are generated using the *rand* function. The MATLAB function *randn* can be used to generate our noise samples. We set the signal power to be one. Then, we can use the built-in functions of MATLAB to calculate the mean, standard deviation, and variance of the noise respectively.

|  | Mean | Standard deviation | Variance |
| --- | --- | --- | --- |
| 5 dB | 7x10-4 | 0.4167 | 0.1737 |
| 50 dB | 5x10-5 | 0.0022 | 5x10-6 |

From the table above, we can see that the mean for both SNRs is approximately zero. However, values for standard deviation and variance differ for both.

The received signals from both 5dB and 50dB have a mean of 0. Thus, this shows that the threshold value of 0 as required in the question is logical in detecting the transmitted data from the corrupted signal. The amplitude of noise in 5 dB also affects the received signal more compared to that of 50 dB. This is also shown in Figures 2 and 3, where the bit error rate at 50 dB is relatively low but for 5 dB, the error bit rate ranges from 10-1 to 10-4.

To conclude phase 1, we showed that additive white Gaussian noise may distort the transmitted data. The solution to this problem is to increase the signal power to increase the SNR value.

# Phase 2

## Implementation

3 different methods of bandpass modulation have been implemented. They are On-Off Keying (OOK) which is the simplest form of Amplitude Shift Keying (ASK), Binary Phase Shift Keying (BPSK), and Binary Frequency Shift Keying (BFSK).

### Carrier signals

A carrier signal is needed for bandpass modulation so that our data signal can be modulated. We will be using a cosine waveform with an amplitude of 2 and a frequency of 10KHz.

An additional carrier signal is needed for BFSK but at a different frequency. Therefore, we chose another carrier signal which is 5 times higher in frequency making it a frequency of 50KHz.

Need to keep in mind to choose a frequency that would be 2 times lesser than the sampling frequency. If not, Nyquist's theorem would not be met, and the recovered sampled signal would not be accurate.

### Sampling

The sampling frequency is set to be 16 times the carrier frequency as the requirement states that the carrier signal is 16 times oversampled. Therefore, the sampling frequency is 160KHz. And therefore, all carrier signals used must not be higher than 80Khz as of Nyquist's theorem. The baseband data rate is also set to 1kbps.

### Noise & SNR

After modulation, an Additive White Gaussian Noise (AWGS) will also be similarly generated just like in phase 1, according to the corresponding SNR value, and then added to the signal to simulate the transmission process. We will be testing out SNR values of -15dB to 15dB with a step size of 1 to analyze the performance of the modulation techniques implemented. We chose this range and step size of SNR values as we have seen from phase 1 that there will not be any error in the data when the SNR is higher than roughly 15dB. And if the SNR values start from 0dB we would not be able to see the trend of the BPSK properly, so we decided to start from -15dB instead with a step size of 1.

### Structure of Phase 2 codes

The outer for loop will be for looping through the different SNR values stated above.   
As the data and noise generated are random, it would be a more accurate representation if we could run the entire process multiple times and get the averages of it. That is the purpose of the inner for loop. For each of the inner loops, a new random set of data will be generated. Followed by modulation, noise generation, transmission, demodulation, and finally sampling & decision logic of all 3 modulation techniques. Lastly, before the loop ends the bit errors are calculated and summed up.

### On-Off Keying (OOK)

For OOK modulation, 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. To achieve just that, all we must do is to multiply the data signal with the carrier signal.

For OOK demodulation and detection, it will be a 3 steps process. Firstly, take the received signal and multiply twice the carrier signal. Doing this will result in a constant term (that has very low to no change of frequency) and a cosine term. Secondly, the signal will be then passed through a 6th order low pass filter with a cut-off frequency of 0.2. This would filter out the cosine term leaving the constant term left. This constant term would be sampled at the sampling rate and using a decision threshold to decide if it would be a bit 1 or a bit 0. If it is above the threshold, it would be bit 1 else it would be a bit 0. The decision threshold for OOK would be half of the amplitude of the carrier signal. As the signal would either be the instantaneous value of the carrier signal or 0. So we will take the midpoint as the decision threshold. As it involves the knowledge of the carrier signal, it is a coherent detection method.

### Binary Phase Shift Keying (BPSK)

For BPSK modulation, a different phase is needed to differentiate between bit 1 and bit 0. The simplest implementation of this is to take a 180° phase shift as it is simply the negative (inverse) of the original signal. So instead of having a 1 and 0 for the data, we would need 1 and -1 to achieve the negative part of the original signal. This is done by multiplying the original signal by 2 and then minus 1. Finally multiplying the carrier signal. If the signal is 1, it will just be the carrier signal while if the signal is -1, it will be the carrier signal with a 180° phase shift. (Which is exactly the opposite of the carrier signal)

For BPSK demodulation and detection, it would be the same as OOK demodulation and detection. The only difference now is that the decision threshold would be 0.

### Binary Frequency Shift Keying (BFSK)

For BFSK modulation, a different frequency is needed to differentiate between bit 1 and bit 0. So, for a bit 0, we would use the carrier signal frequency, 10KHz, and for a bit 1, we would use 5 times the carrier signal frequency, 50KHz. To achieve this, we would first generate 2 signals, the first signal would have the 10Khz carrier frequency when the signal is 0 and zero when the signal is 1. Conversely, the second signal would have the 10Khz carrier frequency when the signal is 1 and zero when the signal is 0. Lastly, we will add both signals together.

For BFSK demodulation and detection, it would be like BPSK demodulation and detection. Just that since there are 2 carrier frequencies, the received signal must be multiplied and filtered separately then add both constant terms together. The decision threshold would remain the same at 0.

## 

## Experiments and Results

As a new set of data and noise will be generated each time, results will not be the same and will vary slightly but the main trend will be similar.

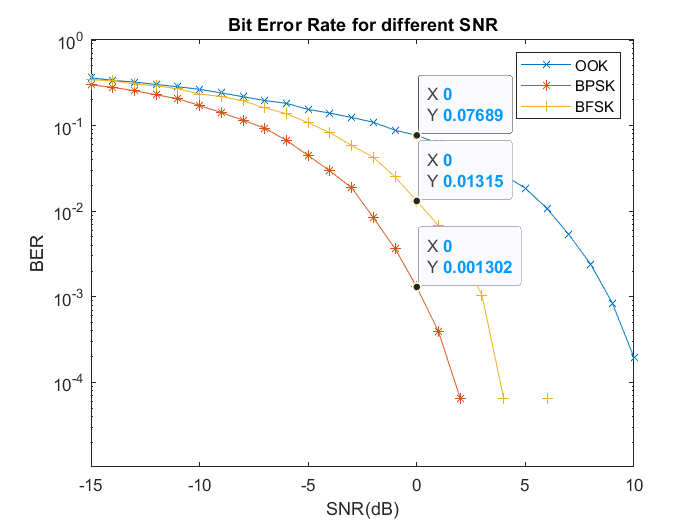
### Bit Error Rate Performance

*Figure 7: BER for OOK, BFSK & BPSK*

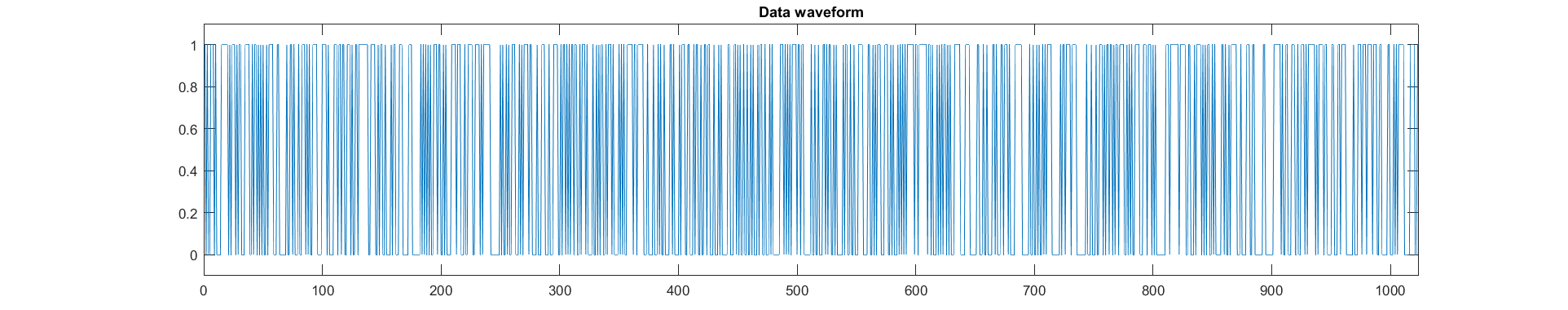
As seen from the Figure above, all 3 modulations fit the trend of having a lower BER when having a larger SNR. This was one of the signs that meant that our implementation of a basic communication system was accurate.

We can also see that BPSK has the best performance out of the 3, followed by BFSK and lastly OOK. This meant that BPSK would require a lesser signal power than that of both BFSK and OOK to achieve the same bit error rate. As an example, for the BER of 10-3, BPSK would need roughly 0dB while BFSK would need roughly 3dB and OOK would need roughly 9dB.

The rest of the Figures below for phase 2 would be taken at a specific SNR value that could be changed in the codes. For our case now, we would be showing at 0dB SNR value. So, the Figure below shows the BER at 0dB SNR.

*Figure 8: BER rates at 0dB SNR*

### Data Waveform

*Figure 9: Plot of 1024 bits of generated data*

Since the signal is oversampled 16 times plus the board rate, the number of bits after modulation would be 160 times more bits than the original data bits. Plotting all the bits would not be beneficial for viewing and analyzing therefore we would only be viewing the first 3000 bits by limiting the x-axis of the plots.

### Modulated signals

*Figure 10: Modulated signals before transmission*

From Figure 10, it can be reasoned that the implementations for all the 3 modulations techniques are correct. For OOK, when the data signal is 1 the carrier signal with an amplitude of 2 is transmitted, and when the data signal is 0 a zero signal would be transmitted. For BPSK, we can see a phase transition of the carrier signal whenever there is a bit transition as well. For BFSK, when the data signal is 1 the higher frequency carrier of 50Khz would be transmitted, and when the data signal is 0 the lower frequency carrier of 10Khz would be transmitted.

### Received signals

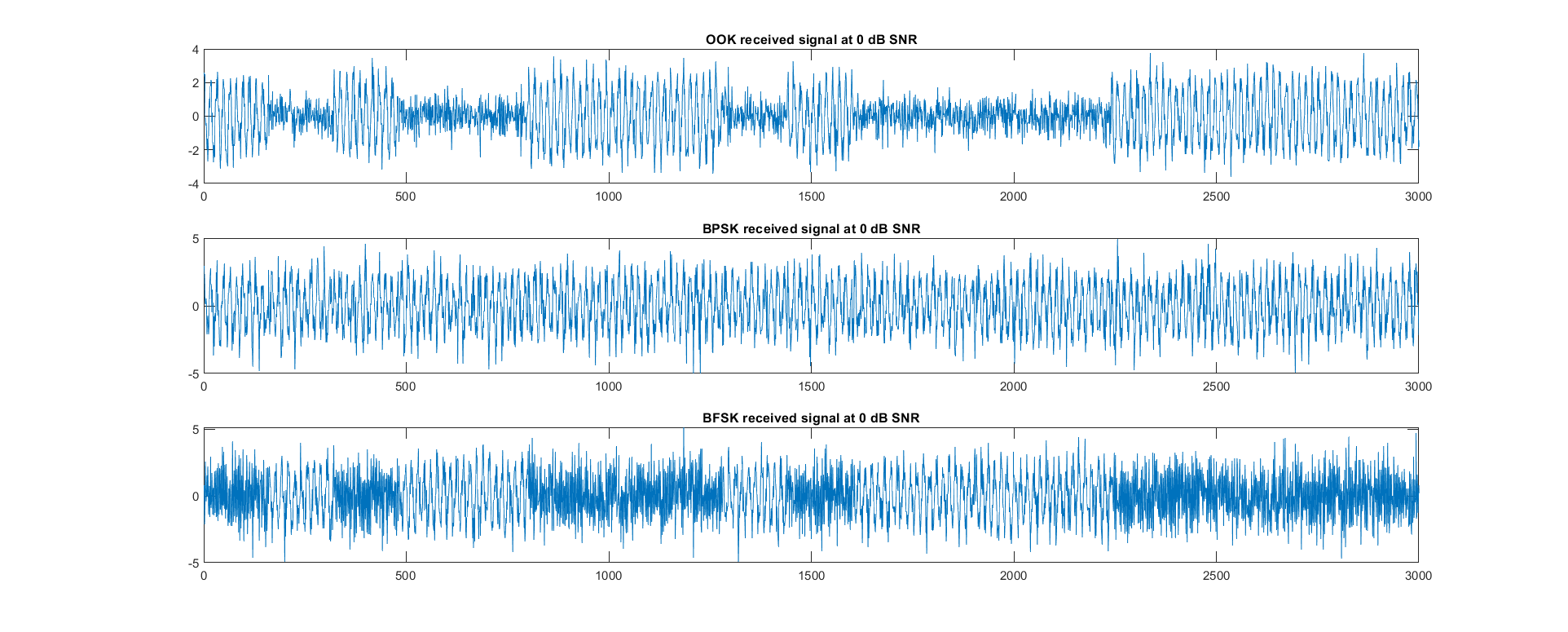
*Figure 11: Modulated signals with noise (Received signals)*

Figure 11 shows the signals received at the receiver. The modulated signals now are corrupted with AWGS. And as we can see the signals now are distorted randomly which would accurately simulate what would happen when transmitted through a channel.

### Demodulated signals

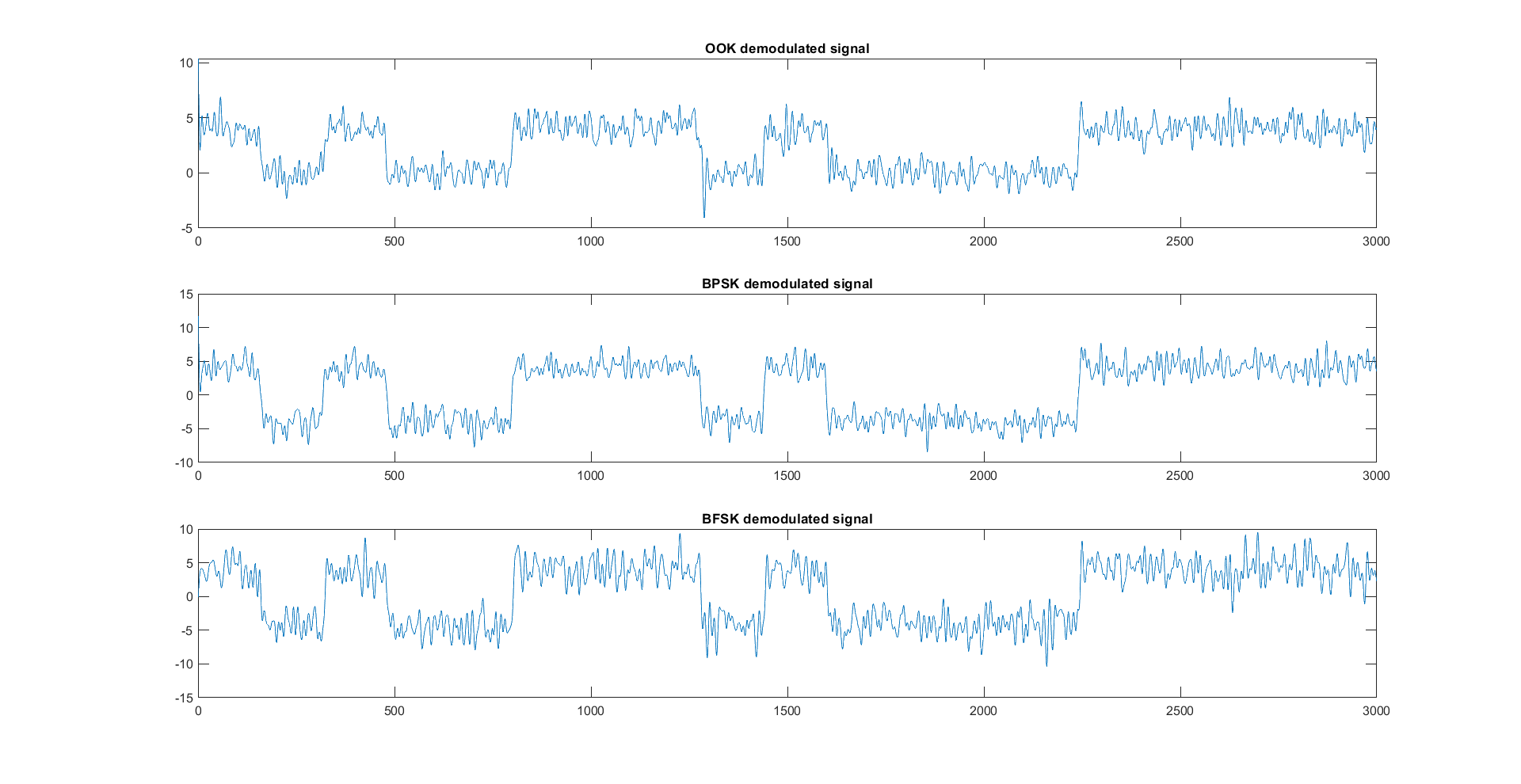
*Figure 12: Demodulated signals before sample and decode*

Figure 12 shows the demodulated signals before sample and decode. At this point, we can see how the noise has affected the amplitude of the signals. From this, the signals are sampled and passed through a decision threshold. For OOK, it would be half of the amplitude of the carrier signal and for BPSK & BFSK it would be 0. Anything above the threshold value would be bit 1 and anything below would be bit 0. And the results would be the decoded signals below.

### Decoded signals

*Figure 13: Sampled and decoded signals plus original data waveform.*

Figure 13 shows the Sampled and decoded signals plus the original data waveform. We can roughly see the similarity of the signal from the original data, and thus also conclude that our modulation and demodulation implementations are correct and thus have a correct implementation of a basic communication system. However, we can also still see some errors. For example, for OOK slightly to the right of the 800-bit mark, we can see the difference from the original data signal.

# Phase 3

## Implementation

We use channel coding in this step to improve and lower the bit error rate of our transmission.

Before the baseband signal is modulated to bandpass for transmission across the channel, channel encoding is applied. To recover the original data, the signal will be demodulated and decoded.

3 different types of error control coding techniques were used to further improve the performance of the proposed communication system. They are cyclic codes, hamming codes, and linear codes. These codes are part of the linear block correction code family, and they are used to detect and rectify errors. Both channel encoding techniques will be used, with a codeword length of 7 bits for a 4-bit message. We calculate the BER by comparing the decoded signal to the original message to be transmitted after the encoded signals have been modulated, transmitted, demodulated, and detected.

### Cyclic and Hamming Code

Cyclic codes and Hamming codes were implemented using MATLAB’s encode and decode function.

| Hamming | hamming\_sig = encode (data,7,4,'hamming/binary')  data= decode(hamming\_signal,7,4,'hamming/binary') |
| --- | --- |
| Cyclic | cyclic\_sig = encode (data,7,4,'cyclic/binary')  data= decode(cyclic\_signal,7,4,'cyclic/binary') |

At the transmission part, the encoding section was inserted into the codebase after the data generation step and before the sampling step for modulation.

At the receiving part, the decoding section was added after the demodulation, sampling, and thresholding step.

### Linear Code

Linear codes were also implemented using MATLAB’s encode and decode function. Before running encode/decode functions, a few codes were needed.

| Linear | pol = cyclpoly (7,4)  parmat = cyclgen (7, pol)  genmat = gen2par(parmat) |
| --- | --- |

Encode/Decode functions:

| Linear | linear\_sig = encode (data,7,4,'linear/binary', genmat)  data= decode (linear\_sig,7,4,'linear/binary', genmat); |
| --- | --- |

At the transmission part, the encoding section was inserted into the codebase after the data generation step and before the sampling step for modulation.

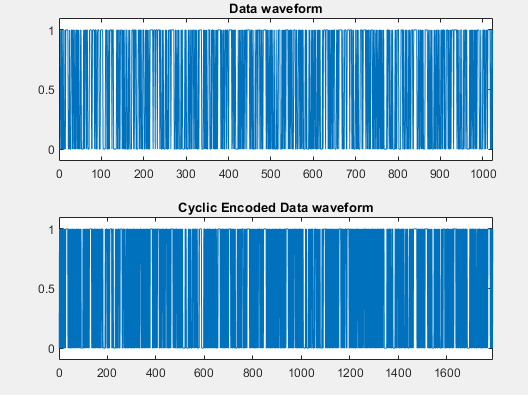
At the receiving part, the decoding section was added after the demodulation, sampling, and thresholding step.

## Experiments and Results

By using error control coding, we were able to see considerable reductions in Bit Error Rate for all SNR dB levels in phase 3 by adding channel coding blocks on both the transmitting and receiving ends of the communication system. Because these channel coding and decoding blocks are built on top of parts 2 and 1, the signal at the data creation stage, modulation, demodulation, sampling, and threshold have the same shape and patterns as those discussed in phase 2 and phase 1. The encoded data before modulation and after demodulation would be the sole differences.

### Cyclic Original vs Encoded Data Waveform

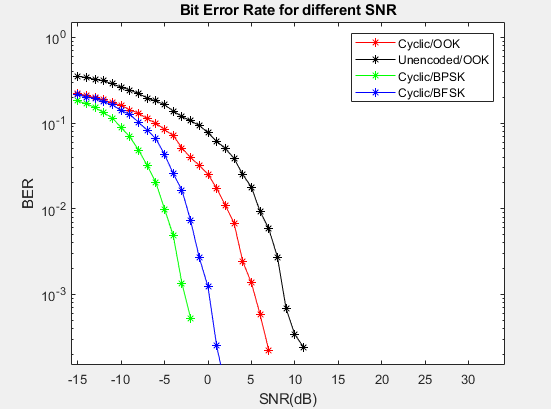
From Figure 14, it can be observed that the original data waveform is 1024 bits while the cyclic encoded data waveform is 1792 bits. Encoded signals are longer as redundancy bits are added into the signal to improve the BER performance. However, these redundancy bits will result in poorer bandwidth efficiency as only 1024/1792 bits in an encoded signal are meaningful data.



*Figure 14: Cyclic encoded data waveform vs original data waveform*

### Cyclic Bit Error Rate Performance

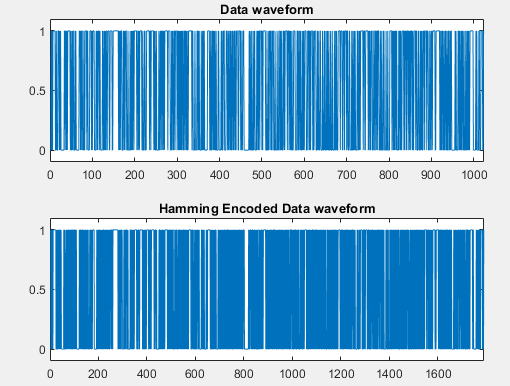
In Figure 15, the improvement of overall BER can be observed. The higher the SNR, the larger the improvement (BER) can be observed in the encoded signal.



*Figure 15: Cyclic Encoded BER against Unencoded BER*

### Hamming Original vs Encoded Data Waveform

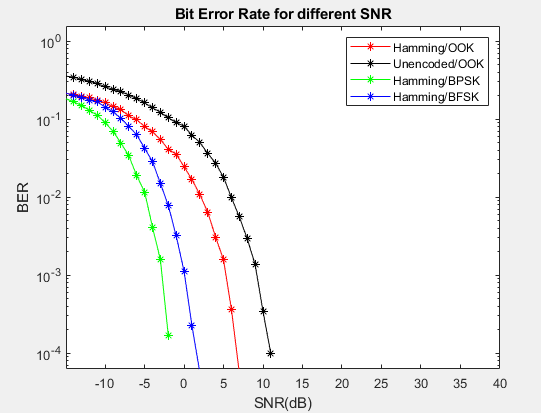
From Figure 16, similarly to what has been mentioned for cyclic code. It can be observed that the original data waveform is 1024 bits while the hamming encoded data waveform is 1792 bits. Encoded signals are longer as redundancy bits are added into the signal to improve the BER performance. However, these redundancy bits will result in poorer bandwidth efficiency as only 1024/1792 bits in encoded signal is meaningful data.



*Figure 16: Hamming encoded data waveform vs Original data waveform*

### Hamming Bit Error Rate Performance

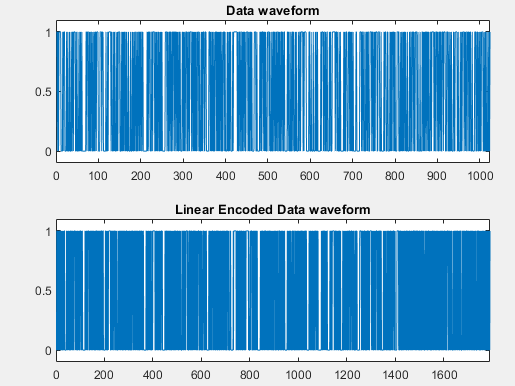
In Figure 17, the improvement of overall BER can be observed. The higher the SNR, the larger the improvement (BER) can be observed in the encoded signal.



*Figure 17: Hamming Encoded BER against Unencoded BER*

### Linear Original vs Encoded Data Waveform

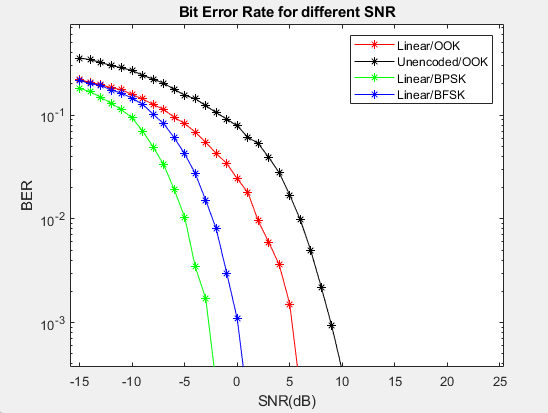
Lastly for Linear encoding, from Figure 18, it can be observed that the original data waveform is 1024 bits while the linear encoded data waveform is 1792 bits. Encoded signals are longer as redundancy bits are added into the signal to improve the BER performance. However, these redundancy bits will result in poorer bandwidth efficiency as only 1024/1792 bits in encoded signal is meaningful data.



*Figure 18: Linear encoded data waveform vs Original data waveform*

### Linear Bit Error Rate Performance

In Figure 19, the improvement of overall BER can be observed. The higher the SNR, the larger the improvement (BER) can be observed in the encoded signal.



*Figure 19: Linear Encoded BER against Unencoded BER*

# Conclusion

We successfully created a basic digital communication system in MATLAB and looked at three essential stages as well as compared the performance of BER in various conditions in this study.

In phase 1, we simulated data transmission by creating noises based on various SNR levels and investigating how noise might alter and damage the data received. Following that, in phase 2, we used various modulation schemes (OOK, BPSK, and BFSK) to transmit signals in the bandpass region of the channel and compared their BER performance. We used channel encoding in phase 3 to give error detection and correction capability in the receiving end of the channel, and the three encoding techniques used here are Hamming Code, Linear Code, and Cyclic Code.

When compared to the vanilla implementation of data transmission in phase 1, our research shows that modulation and channel encoding can improve BER performance. We presented and compared the performance of the signal modulation techniques in phases 2 and 3, and we can see that BPSK performs the best of all the modulation techniques, with BFSK coming in second. Implementations that use hamming, linear, or cyclic code outperform those that are modulated and unencoded in phase 3. The implementation with BPSK Modulation and Channel Encoding performs the best of all the implementations.

Finally, while our implementations were able to reduce the signal's BER by encoding the message bits with redundant bits for error detection and correction, we believe that bandwidth efficiency can be further improved by implementing source encoding, which maximizes entropy (optimal code length) while transmitting at the maximum data rate with a low error rate. In conclusion, numerous methodologies have been developed to eliminate any error that may occur during transmission, and we were able to demonstrate in this study that BER may be lowered by using the techniques we learned in this course.