**Sec. 2**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Ibrahim Abelwahhab Mohamed | 19015169 |
| Haitham Atia Elsayed | 19016857 |
| Hythem Ahmed Shaban | 19016856 |

Digital Communications Lab

**Experiment 2**

Basics of BER calculations and channel models

# Introduction

This is the first experiment in the Digital Communications Lab. In this experiment, we study the most basic method for assessing the performance of digital communication systems, the Bit *Error Rate*. We also study simple models for communication channels and their effect on the communication system from the perspective of bit error rate.

# About the lab

The digital communications lab consists of a set of experiments which introduce essential digital communication concepts to students. The lab consists mainly of MATLAB-based experiments in which the students complete the required tasks of the experiment by writing MATLAB codes. Some experiments of the lab may consist of hardware-based experiments which are performed in the digital communication laboratory on campus.

**Location of the digital communication laboratory on campus:**

**2nd floor, Electrical Engineering Building.**

**Instructions for completing the experiment**

The following describes the procedure for completing MATLAB-based experiments in the digital communication lab. Other experiments that are not MATLAB-based may have different instructions which will be described in the DOCX documents of those experiments.



* An experiment consists of 1) a DOCX document explaining the experiment, 2) a link to a video explaining the experiment, and 3) the MATLAB files required to complete the experiment.
* Each experiment typically consists of several parts.
* The DOCX document is your starting point. It explains in detail the requirement of each experiment part.
* The accompanying video explains the experiment. The explanation is mainly based on the experiment DOCX.
* The MATLAB files are used by the student to complete the experiment. The main MATLAB file for the experiment is usually named LabX\_script.m with X being the experiment number. For example, the main file for Experiment 1 would be called Lab1\_script.m. This is the first file that you should open and start working on. Inside that file, there can be referrals to other MATLAB files that you need to complete.
* Follow the instructions in the DOCX file and the explanation video to fill the provided MATLAB files. The experiment is completed once you generate the outcomes required in all parts of the experiment.

# Lab submission and discussion

* This experiment is due on the weeks **from15th to 28th of October.**
* Please bring a **printed copy of your codes** and **the lab DOCX with your answers in it** (your answers can be handwritten, but it is preferable to type them).

# Lab policy regarding grading and cheating

**Grading**

* Each experiment is graded out of 20 marks. If you sum the total grades of the lab, it would be larger than 20. This means that you have a chance to miss some parts of the lab and still get a full mark.
  + The grading is very lenient. Typically, it would be **very hard to grade that is less than 16**.
  + Any student who does the whole lab by themselves are expected to get a grade of 18 or more, without even getting everything right.

**Cheating**

* In exchange for this grading policy, cheating will not be tolerated.
* If a student cheats in **any part in an experiment**, he/she will get **a zero for that entire lab**.
* If a student cheats in **two or more experiments**, he/she will get **a zero in the entire lab**.
* The reason behind this policy towards cheating is the following: detecting cheating cases requires an enormous amount of work by the instructors. Given how lenient the grading of the lab is expected to be, there is no justification for a student to cheat (this is in addition to the fact that there is no justification for cheating in general).

Note: it is okay to talk with your friends about the experiment if you feel that you are stuck somewhere. But please be aware that doing any of the following **is considered cheating, and all involved students in these cases are treated as cheaters:**

* A group of students write MATLAB codes together.
* A group of students discuss the experiment together, one student of the group writes the code, and all students use the same code.

# Lab inventory

|  |  |
| --- | --- |
| **File name** | **Description** |
| Lab video | **Link:** <https://youtu.be/zBtf8aqwrJE> |
| Lab1\_BER.docx | Contains the details of the experiment. |
| Lab1\_script.m | The main m-file containing the MATLAB script for the experiment. |
| GenerateBits.m | Additional MATLAB function files used by the experiment script file. |
| BSC.m |
| ComputeBER.m |
| DecodeBitsFromSamples.m |
| GenerateSamples.m |

# Background

The most basic target of the study of digital communications is to understand digital communication systems and how digital information can be conveyed from a source or transmitter to a destination or receiver over a channel. Depending on the communication systems, channels can be wired circuits, wireless channels, satellite channels and so on. The study of digital communications begins by transforming the digital communication system into an equivalent mathematical model, and then attempts to design transmitters and receivers which achieves the target of information transmission over the channel in an efficient manner.

Figure 1 shows an example of a digital communication system. The goal of the transmitter and receiver is to deliver the digital data from the source to the sink in the best way possible. There are several ways to define what *best* mean: one of the most common and most important methods to assess the performance of a communication system is the *Bit Error Rate (BER)*.



**Figure 1 An example of a digital communication system**

|  |
| --- |
| **Bit Error Rate (BER):** the rate of error occurrences among an output sequence of bits corresponding to an input sequence of bits. |

An empirical method of computing the BER in a communication system is as follows:

1. Generate a sequence of bits at the input side.
2. Pass the sequence of input bits through the system to receive a corresponding output sequence.
3. Count the number of errors in the output sequence by comparing it to the input sequence; call that number of errors
4. The BER is given by

In this experiment, we will compute the BER of different digital communication systems. These systems differ in their respective channel models and therefore their corresponding transmitter and receiver designs.

# Experiment

## Part 1 (3 Marks)

In this part, we consider a very simple digital communication system, in which the channel takes as input binary digits , and produces the corresponding output according to the following equation.

The channel described above simply flips the input bit with probability or passes the input bit unchanged with probability ; this channel is referred to as the *Binary Symmetric Channel (BSC)*. The system is shown in Figure 2. In this system, we assume that the transmitter takes the input bits coming from the source and passes them unchanged to the channel (i.e., the transmitter does nothing). However, we would like to investigate how the receiver can be designed to produce a good BER.



**Figure 2 A digital communication system with a Binary Symmetric Channel**

Your goal in this task is to design the receiver. You know that the channel takes the data, flips it randomly (with probability ) and gives you the output. What would the receiver do with that output?

Think about the following two receivers and say what is the expected performance of these receivers. As a hint to start, these two receivers are not very good.

**Example 1:** the receiver gives a 0 bit as output. This output does not depend at all on what the channel is giving out.

|  |  |
| --- | --- |
| **Questions** | |
| **What is the corresponding BER for that receiver? You do not need to implement it in the m-file to answer.** | **(Implemented in MATLAB) BER = 0.5**  **In some practical cases, it is considered that BER = 100%**  **As the receiver does not depend at all on what the channel is giving out.** |
| **What is the reason behind the performance of this receiver?** | **The threshold of the receiver is high so all the received bit will be below it then it is 0.**  **As there will be a probability that half bits of the output are the flipping of the input.** |

**Example 2:** the receiver gives random output, i.e., 0s and 1s with a probability of 0.5. Again, this output is not based on what the channel is giving out.

|  |  |
| --- | --- |
| **Questions** | |
| **What is the corresponding BER for that receiver? You do not need to implement it in the m-file to answer.** | **(implemented in MATLAB) BER = 0.5**  **In some practical cases, it is considered that BER = 100%**  **As the receiver does not depend at all on what the channel is giving out.** |
| **What is the reason behind the performance of this receiver?** | **Because the output is 0’s or 1’s with a probability of 0.5. So, There will be a probability of 0.5 that the output is the flipping of input** |

The above two receivers are examples of receivers which clearly would not be considered as good receivers from a BER perspective (why?). In the following part of the experiment, you would design the best receiver and assess its performance by computing the corresponding BER.

**EXP. *Complete PART 1 in the experiment M-file* Lab1\_script.m *and the missing implementation of all included functions. Then answer the following questions:***

|  |  |
| --- | --- |
| **Questions** | |
| **What is the corresponding BER for receivers 1 and 2 above? You do not need to implement the two receivers to answer.** | **RX1: BER = 0.5 RX2: BER = 0.5**  **In some practical cases, it is considered that BER = 100%**  **As the receiver does not depend at all on what the channel is giving out.** |
| **What is the reason behind the performance of these two receivers?** | **As the output does not depend at all on what the channel is giving out.** |
| **What is the BER of the best receiver?** | **BER = p for p <= 0.5 BER = 1 – p for p>=0.5** |

### Part 1-a (2 Marks)In this part, we study the impact of the BSC channel parameter on the BER of the digital communication system. Namely, we vary the value of from 0 to 1, and for each value of we compute the corresponding BER, we save these values in an array, then, later on in Part 3-a, plot the values of BER versus their corresponding parameter value .

**EXP. *Complete PART 1-a in the experiment M-file* Lab1\_script.m*. The final figure containing the required plot will be generated at the end of Part 3-a of the experiment.***

## Part 2 (3 Marks)

In this part, we again consider the system proposed in Figure 2 but we try to improve the transmitter a bit. Namely, the transmitter works as follows: for each input bit , the transmitter generates a set of 5 copies of the bit which are then passed sequentially through the channel. Note that this behavior leads to the increase in the number of bits being passed through the channel (is that good or bad?). The system is shown in Figure 3. For this transmitter, the receiver expects to receive a sequence of 5 channel outputs, all corresponding to the same input bit. Therefore, we expect that the receiver can use these outputs for a better decoding performance. In this part, we investigate how to design the best receiver and the corresponding BER performance.



**Figure 3 A digital communication system with a Binary Symmetric Channel and a modified transmitter**

**EXP. *Complete PART 2 in the experiment M-file* Lab1\_script.m *and the missing implementation of all included functions. Then answer the following questions:***

|  |  |
| --- | --- |
| **Questions** | |
| **What is the BER of the best receiver?** | **0.06**  **Theoretical: 0.2/5=0.04** |
| **What is the expected (theoretical) BER if the number of repetitions is increase to 10?** | **p/fs**  **0.2/10=0.02** |
| **What is the cost/downside of using the transmitter in Part 2?** | **Bit rate will decrease. (Large time to send information)**  **Need large B.W to send at the same time.** |

### Part 2-a (2 Marks)

Similar to Part 1-a, in this part, we study the impact of the BSC channel parameter on the BER of the digital communication system in Part 2.

**EXP. *Complete PART 2-a in the experiment M-file* Lab1\_script.m*. The final figure containing the required plot will be generated at the end of Part 3-a of the experiment.***

## Part 3 (3 Marks)

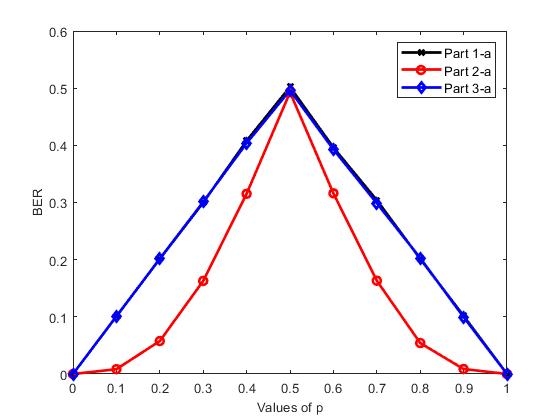
In part 3, we consider the same system in Part 2. However, the channel in Part 3 generates correlated outputs among the 5 transmitter outputs that correspond to the same input bit. For example, for a 0 input bit to the transmitter and a corresponding five copies of the bit 0, the channel output either generates a set of five 0’s with probability or a set of five 1’s with probability . In this case, we investigate the design of the best receiver and the corresponding BER.

**EXP. *Complete PART 3 in the experiment M-file* Lab1\_script.m *and the missing implementation of all included functions. Then answer the following questions:***

|  |  |
| --- | --- |
| **Questions** | |
| **What is the BER of the best receiver?** | **0.2** |
| **What is the reason behind such a performance?** | **The error is due to the channel which flips all the repeated bits with probability p (0.2).** |

### Part 3-a (2 Marks)

Finally, we study the impact of the BSC channel parameter on the BER of the digital communication system in Part 3.

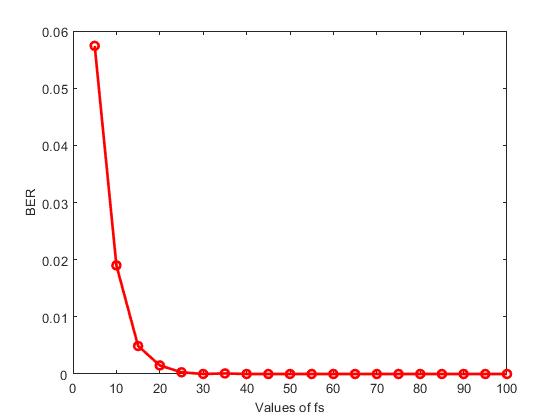
**EXP. *Complete PART 3-a in the experiment M-file* Lab1\_script.m*. The final figure containing the plots from all three parts can now be generated. Please add the generated plot in the box below.***

|  |
| --- |
| **Please insert the plot here** |

|  |  |
| --- | --- |
| **Questions** | |
| **Which of the three systems have the best performance in terms of BER?** | **System of part 2** |
| **If the receiver you designed in any of the previous parts attain a BER more than 0.5, how can it be changed to attain a maximum of 0.5 BER?** | **We already designed all the receivers with maximum BER = 0.5** |

## Part 4 (8 Marks)

In this part, we go back to the system considered in Part 2, namely the system with a transmitter which generated a set of 5 repetitions to the input bit. Now, we would like to investigate the effect of changing the number of repetitions on the decoding performance. You need to generate a figure where the x-axis shows the number of repetitions, and the y-axis shows the corresponding BER. In this part, you can consider p = 2.

**EXP. *Write your own code in PART 4 in the experiment M-file* Lab1\_script.m*. Your code should generate a figure as described in the discussion above.***

**Lab1\_script**

%%

% Alexandria University

% Faculty of Engineering

% Electrical and Electronic Engineering Department

%

% Course: Digital Communications Lab

%

% Lab No. 1: Basics of BER calculation and channel modeling

%% Simulation parameters

N\_bits = 10000; % Total number of bits

p = 0.2; % Channel parameter (probability of bit flipping)

%% Part 1: BER for simple BSC channel

% Generate a bit sequence

bit\_seq = GenerateBits(N\_bits); % IMPLEMENT THIS: Generate a sequence of bits equal to the total number of bits

% Pass the bit sequence through the channel

rec\_sample\_seq = BSC(bit\_seq,1,p); % Generate the received samples after passing through the bit flipping channel

% Decode bits from received bit sequence

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_1',p); % IMPLEMENT THIS: Decode the received bits

% Compute the BER

BER\_case\_1 = ComputeBER(bit\_seq,rec\_bit\_seq); % IMPLEMENT THIS: Calculate the bit error rate

%% Part 1-a: Effect of bit flipping probability on BER

% GOAL: Make a plot for the BER versus different values of the channel

% parameter p

p\_vect = 0:0.1:1; % Use this vector to extract different values of p in your code

BER\_case\_1\_vec = zeros(size(p\_vect)); % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE

for p\_ind = 1:length(p\_vect)

rec\_sample\_seq = BSC(bit\_seq,1,p\_vect(p\_ind));

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_1',p\_vect(p\_ind));

BER\_case\_1\_vec(p\_ind) = ComputeBER(bit\_seq,rec\_bit\_seq);

end

%%%

%% Part 2: BER for simple bit-flipping channel with multiple samples

% System parameters

fs = 5; % Number of samples per symbol (bit)

% Generate a bit sequence

bit\_seq = GenerateBits(N\_bits); % Generate a sequence of bits equal to the total number of bits

% Generate samples from bits

sample\_seq = GenerateSamples(bit\_seq,fs); % IMPLEMENT THIS: Generate a sequence of samples for each bit

% Pass the sample sequence through the channel

rec\_sample\_seq = BSC(sample\_seq,fs,p); % Generate the received samples after passing through the bit flipping channel

% Decode bits from received bit sequence

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_2', p, fs); % IMPLEMENT THIS: Decode the received bits

% Compute the BER

BER\_case\_2 = ComputeBER(bit\_seq,rec\_bit\_seq); % Calculate the bit error rate

%% Part 2-a: Effect of bit flipping probability on BER

% GOAL: Make a plot for the BER versus different values of the channel

% parameter p

p\_vect = 0:0.1:1; % Use this vector to extract different values of p in your code

BER\_case\_2\_vec = zeros(size(p\_vect)); % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE

for p\_ind = 1:length(p\_vect)

rec\_sample\_seq = BSC(sample\_seq,fs,p\_vect(p\_ind));

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_2', p\_vect(p\_ind), fs);

BER\_case\_2\_vec(p\_ind) = ComputeBER(bit\_seq,rec\_bit\_seq);

end

%%%

%% Part 3: BER for simple bit-flipping channel with multiple samples and correlated channel

% System parameters

fs = 5; % Number of samples per symbol (bit)

% Generate a bit sequence

bit\_seq = GenerateBits(N\_bits); % Generate a sequence of bits equal to the total number of bits

% Generate samples from bits

sample\_seq = GenerateSamples(bit\_seq,fs); % Generate a sequence of samples for each bit

% Pass the sample sequence through the channel

rec\_sample\_seq = BSC(sample\_seq,fs,p,'correlated'); % Generate the received samples after passing through the bit flipping channel

% Decode bits from received bit sequence

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_3',p ,fs); % IMPLEMENT THIS: Decode the received bits

% Compute the BER

BER\_case\_3 = ComputeBER(bit\_seq,rec\_bit\_seq); % Calculate the bit error rate

%% Part 3-a: Effect of bit flipping probability on BER

% GOAL: Make a plot for the BER versus different values of the channel

% parameter p

p\_vect = 0:0.1:1; % Use this vector to extract different values of p in your code

BER\_case\_3\_vec = zeros(size(p\_vect)); % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE

for p\_ind = 1:length(p\_vect)

rec\_sample\_seq = BSC(sample\_seq,fs,p\_vect(p\_ind),'correlated');

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_3',p\_vect(p\_ind),fs);

BER\_case\_3\_vec(p\_ind) = ComputeBER(bit\_seq,rec\_bit\_seq);

end

%%%

% Plotting results

figure

plot(p\_vect,BER\_case\_1\_vec,'x-k','linewidth',2); hold on;

plot(p\_vect,BER\_case\_2\_vec,'o-r','linewidth',2); hold on;

plot(p\_vect,BER\_case\_3\_vec,'d-b','linewidth',2); hold on;

xlabel('Values of p','fontsize',10)

ylabel('BER','fontsize',10)

legend('Part 1-a','Part 2-a','Part 3-a','fontsize',10)

%% Part 4: Effect of number of repetitions on BER

% GOAL: Make a plot for the BER versus the number of repetitions used in

% the transmitter of part 2

% There is no template code for this part. Please write your own complete

% code here. You can re-use any of the codes in the previous parts

% Generate a bit sequence

bit\_seq = GenerateBits(N\_bits); % Generate a sequence of bits equal to the total number of bits

fs\_vect = 5:5:100; % Use this vector to extract different values of fs in your code

BER\_case\_2\_vec = zeros(size(fs\_vect)); % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE

for fs\_ind = 1:length(fs\_vect)

sample\_seq = GenerateSamples(bit\_seq,fs\_vect(fs\_ind));

rec\_sample\_seq = BSC(sample\_seq,fs\_vect(fs\_ind),p);

rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,'part\_2', p, fs\_vect(fs\_ind));

BER\_case\_2\_vec(fs\_ind) = ComputeBER(bit\_seq,rec\_bit\_seq);

end

figure

plot(fs\_vect,BER\_case\_2\_vec,'o-r','linewidth',2); hold on;

xlabel('Values of fs','fontsize',10)

ylabel('BER','fontsize',10)

%%%

**GenerateBits**

function bit\_seq = GenerateBits(N\_bits)

%

% Inputs:

% N\_bits: Number of bits in the sequence

% Outputs:

% bit\_seq: The sequence of generated bits

%

% This function generates a sequence of bits with length equal to N\_bits

bit\_seq = zeros(1,N\_bits);

%%% WRITE YOUR CODE HERE

% randi([imin,imax], sz1, sz2) function returns an array containing integers drawn from

% the discrete uniform distribution on the interval [imin,imax] with dims sz1\*sz2

% So, randi([0 1], 1, N\_bits) returns an array of size 1 \* N\_bits of random

% values of 0's and 1's

bit\_seq = randi([0 1], 1, N\_bits);

%%%

**BSC**

function rec\_sample\_seq = BSC(sample\_seq,fs,p,channel\_type)

%

% Inputs:

% sample\_seq: The input sample sequence to the channel

% fs: The sampling frequency used to generate the sample sequence

% p: The bit flipping probability

% channel\_type: The type of channel, 'independent' or 'correlated'

% Outputs:

% rec\_sample\_seq: The sequence of sample sequence after passing through the channel

%

% This function takes the sample sequence passing through the channel, and

% generates the output sample sequence based on the specified channel type

% and parameters

% I think ~~ is not valuable!

sample\_seq = ~~sample\_seq;

rec\_sample\_seq = zeros(size(sample\_seq));

rec\_sample\_seq = ~~rec\_sample\_seq;

if (nargin <= 3)

channel\_type = 'independent';

end

switch channel\_type

case 'independent'

% rand(n) function returns an array of size n in the range from 0

% to 1

% here we compare each element of the generated random array with

% the probability of fliping

% if the element is less than or equal to p (with prob p) then the

% output of the channel should be flipped

% if the element is greater than p (with prob 1-p) then the output

% of the channel should be the same as input

channel\_effect = rand(size(rec\_sample\_seq))<=p;

case 'correlated'

% rand(n) function returns an array of size n in the range from 0

% to 1

% here the correlated channel has an input from the Tx with

% repeatation of fs and correlated outputs among the 5

% transmitter outputs that correspond to the same input bit

% repmat(channel\_effect, fs, 1) is a function that returns a

% repeated version of array channel\_effect with size fs\*1

% channel\_effect(:)' is used to transpose to 1\*fs vector

channel\_effect = rand(1,length(rec\_sample\_seq)/fs)<=p;

channel\_effect = repmat(channel\_effect,fs,1);

channel\_effect = channel\_effect(:)';

end

% here we xor between input bit\_seq and the channel\_effect

% if out is 1 it means that bit\_seq and channel\_effect are difference so

% flipping is occurred

% if out is 0 it means that bit\_seq and channel\_effect are similar so

% no change

rec\_sample\_seq = xor(sample\_seq,channel\_effect);

% I think + 0 is not valuable!

rec\_sample\_seq = rec\_sample\_seq + 0;

**DecodeBitsFromSamples**

function rec\_bit\_seq = DecodeBitsFromSamples(rec\_sample\_seq,case\_type,p,fs)

%

% Inputs:

% rec\_sample\_seq: The input sample sequence to the channel

% case\_type: The sampling frequency used to generate the sample sequence

% fs: The bit flipping probability

% Outputs:

% rec\_sample\_seq: The sequence of sample sequence after passing through the channel

%

% This function takes the sample sequence after passing through the

% channel, and decodes from it the sequence of bits based on the considered

% case and the sampling frequence

if (nargin <= 3)

fs = 1;

end

switch case\_type

case 'part\_1'

%%% WRITE YOUR CODE FOR PART 1 HERE

%ex1

%rec\_bit\_seq = zeros(size(rec\_sample\_seq));

%ex2

%rec\_bit\_seq = rand(size(rec\_sample\_seq))<=0.5;

%best Rx

% this is the best Rx as it is dependent on the channel output and

% it states that if the probability of flipping is less than 0.5

% then it is better that the output of the receiver to be the same

% as the input of the receiver, and if the probability of flipping

% is greater than 0.5 then it is better that the output of the

% receiver to be the flipped of the output of the channel

% for example: if input 1010101010, and with p = 0.2 then the

% output of the channel might be 1110100010 then it is better that

% the Rx outputs the same as the output of the channel because

% there are flippings in two bits only

if p <= 0.5

rec\_bit\_seq = rec\_sample\_seq;

end

if p > 0.5

rec\_bit\_seq = ~rec\_sample\_seq;

end

%%%

case 'part\_2'

%%% WRITE YOUR CODE FOR PART 2 HERE

% first we initilize rec\_bit\_seq to be the same size as the input

%seq

% we generater for loop from 1 to length(rec\_sample\_seq)/fs

% output bit is the most frequently value in the fs bits

% then we use mode function to find most frequently value in the

% array within the range from (i-1)\*fs+1 to (i-1)\*fs+1+(fs-1)

% for example: if fs = 5 then for first 5 bits (from 1 to 5)

% rec\_bit\_seq(1) = mode(rec\_sample\_seq(1 : 5));

rec\_bit\_seq = zeros(size(length(rec\_sample\_seq)/fs));

if p <= 0.5

for i = 1 : length(rec\_sample\_seq)/fs

rec\_bit\_seq(i) = mode(rec\_sample\_seq((i-1)\*fs+1 : (i-1)\*fs+1+(fs-1)));

end

end

if p > 0.5

rec\_sample\_seq = ~ rec\_sample\_seq;

for i = 1 : length(rec\_sample\_seq)/fs

rec\_bit\_seq(i) = mode(rec\_sample\_seq((i-1)\*fs+1 : (i-1)\*fs+1+(fs-1)));

end

end

%%%

case 'part\_3'

%%% WRITE YOUR CODE FOR PART 3 HERE

rec\_bit\_seq = zeros(size(length(rec\_sample\_seq)/fs));

if p <= 0.5

for i = 1 : length(rec\_sample\_seq)/fs

rec\_bit\_seq(i) = mode(rec\_sample\_seq((i-1)\*fs+1 : (i-1)\*fs+1+(fs-1)));

end

end

if p > 0.5

rec\_sample\_seq = ~ rec\_sample\_seq;

for i = 1 : length(rec\_sample\_seq)/fs

rec\_bit\_seq(i) = mode(rec\_sample\_seq((i-1)\*fs+1 : (i-1)\*fs+1+(fs-1)));

end

end

%%%

end

**ComputeBER**

function BER = ComputeBER(bit\_seq,rec\_bit\_seq)

%

% Inputs:

% bit\_seq: The input bit sequence

% rec\_bit\_seq: The output bit sequence

% Outputs:

% BER: Computed BER

%

% This function takes the input and output bit sequences and computes the

% BER

%%% WRITE YOUR CODE HERE

wrong\_bits = abs(rec\_bit\_seq - bit\_seq);

num\_wrong\_bits = sum(wrong\_bits);

BER = num\_wrong\_bits / length(bit\_seq);

%%%

**GenerateSamples**

function sample\_seq = GenerateSamples(bit\_seq,fs)

%

% Inputs:

% bit\_seq: Input bit sequence

% fs: Number of samples per bit

% Outputs:

% sample\_seq: The resultant sequence of samples

%

% This function takes a sequence of bits and generates a sequence of

% samples as per the input number of samples per bit

sample\_seq = zeros(size(bit\_seq\*fs));

%%% WRITE YOUR CODE FOR PART 2 HERE

% we generate neested loop

% outter loop from 1 to length(bit\_seq) for input bit\_seq

% inner loop from 1 to fs to repeat each input bit fs times

% for ex: fs = 5

% i = 1

% sample\_seq(0 + 1) = bit\_seq(1)

% sample\_seq(0 + 2) = bit\_seq(1)

% sample\_seq(0 + 3) = bit\_seq(1)

% sample\_seq(0 + 4) = bit\_seq(1)

% sample\_seq(0 + 5) = bit\_seq(1)

% i = 2

% sample\_seq(1 \* 5 + 1) = bit\_seq(2)

% sample\_seq(1 \* 5 + 2) = bit\_seq(2) and so on

for i = 1 : length(bit\_seq)

for j = 1 : fs

sample\_seq((i - 1) \* fs + j) = bit\_seq(i);

end

end

%%%