Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Spring semester, 2021/2022

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثاني, 2022/2021

Lab	ID	Name
7	18012007	نوران عادل محمد فوزى
7	18010724	زياد خليل البكري الضوي
7	18011290	كيرلس يوسف غبريال

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Spring semester, 2021/2022

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثاني, 2022/2021

DIGITAL COMMUNICATIONS LAB

Experiment 1

Basics of BER calculations and channel models

The code

Lab1 script

```
%% 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 to

% Pass the bit sequence through the channel
rec_sample_seq = BSC(bit_seq,1,p);  % Generate the received samples after passing through the


% Decode bits from received bit sequence
rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_1');  % IMPLEMENT THIS: Decode the r

% Compute the BER
BER = ComputeBER(bit_seq,rec_bit_seq); % IMPLEMENT THIS: Calculate the bit error rate


```

Credits: Dr. Mohammed Karmoose

```
%% 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
BER_case_1_vec  = zeros(size(p_vect));  % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE
for p_diff = 1:length(p_vect)
    rec_sample_seq = BSC(bit_seq,1,p_vect(p_diff));
    rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_1');
    BER_case_1_vec(p_diff) = 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 bi

% Generate samples from bits
sample_seq = GenerateSamples(bit_seq,fs); % IMPLEMENT THIS: Generate a sequence of samples fo

% Pass the sample sequence through the channel
rec_sample_seq = BSC(sample_seq,fs,p);   % Generate the received samples after passing throug

% Decode bits from received bit sequence
rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_2',fs);    % IMPLEMENT THIS: Decode

% 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
BER_case_2_vec  = zeros(size(p_vect));  % Use this vector to store the resultant BER

%%% WRITE YOUR CODE HERE
for p_diff = 1:length(p_vect)
    rec_sample_seq = BSC(sample_seq,fs,p_vect(p_diff));
    rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_2',fs);
    BER_case_2_vec(p_diff) = ComputeBER(bit_seq,rec_bit_seq);
end
%%%
```

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

% Generate a bit sequence
bit_seq = GenerateBits(N_bits); % Generate a sequence of bits equal to the total number of bi

% 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 pa

% Decode bits from received bit sequence
rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_3',fs);    % IMPLEMENT THIS: Decode

% 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
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',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


%%% WRITE YOUR CODE HERE
% Generate a bit sequence
bit_seq = GenerateBits(N_bits); % Generate a sequence of bits equal to the total number of bi

fs_vect         = 1:10;              % Use this vector to extract different values of fs in y
BER_case_4_vec  = zeros(size(fs_vect));  % Use this vector to store the resultant BER

for fs_diff = 1:length(fs_vect)
    sample_seq = GenerateSamples(bit_seq,fs_vect(fs_diff));
    rec_sample_seq = BSC(sample_seq,fs_vect(fs_diff),p);
    rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_2',fs_vect(fs_diff));
    BER_case_4_vec(fs_diff) = ComputeBER(bit_seq,rec_bit_seq);
end

figure
plot(fs_vect,BER_case_4_vec,'o-r','linewidth',2);

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
bit_seq = randi([ 0 , 1] ,[1 ,N_bits] ) ;
%%%
```

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Spring semester, 2021/2022

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثاني, 2022/2021

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
sample_seq = zeros(1,length(bit_seq)*fs); %% fs=5
n = 1;
for i = 1:1:(length(bit_seq)*fs)     %%fs=5
    sample_seq(1,i) = bit_seq(1,n);
    if mod(i,fs) == 0   %%fs=5
        n = n + 1;
    end
end
%%%
```

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

sample_seq      = ~~sample_seq;
rec_sample_seq  = zeros(size(sample_seq));
rec_sample_seq  = ~~rec_sample_seq;

if (nargin <= 3)
    channel_type = 'independent';
end
```

Credits: Dr. Mohammed Karmoose

```
22
23 -    switch channel_type
24
25 -        case 'independent'
26 -            channel_effect = rand(size(rec_sample_seq))<=p;
27 -        case 'correlated'
28 -            channel_effect = rand(1,length(rec_sample_seq)/fs)<=p;
29 -            channel_effect = repmat(channel_effect,fs,1);
30 -            channel_effect = channel_effect(:)';
31 -    end
32
33 -    rec_sample_seq = xor(sample_seq,channel_effect);
34 -    rec_sample_seq = rec_sample_seq + 0;
```

DecodeBitsFromSamples

```
1   function rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,case_type,fs)
2   %
3   % Inputs:
4   %   rec_sample_seq: The input sample sequence to the channel
5   %   case_type:      The sampling frequency used to generate the sample sequence
6   %   fs:             The bit flipping probability
7   % Outputs:
8   %   rec_sample_seq: The sequence of sample sequence after passing through the channel
9   %
10  % This function takes the sample sequence after passing through the
11  % channel, and decodes from it the sequence of bits based on the considered
12  % case and the sampling frequence
13
14  if (nargin <= 2)
15      fs = 1;
16  end
17
18  switch case_type
19
20      case 'part_1'
21          %%% WRITE YOUR CODE FOR PART 1 HERE
22          rec_bit_seq = zeros(1,length(rec_sample_seq));
23          rec_bit_seq= rec_sample_seq ;
24          %%%
```

```
    case 'part_2'
        %%% WRITE YOUR CODE FOR PART 2 HERE
        rec_bit_seq = zeros(1,length(rec_sample_seq)/fs);
        n = 1;
        num_of_ones = 0;
        num_of_zeros = 0;
        for i = 1:1:length(rec_sample_seq)
            if (rec_sample_seq(1,i) == 1)
                 num_of_ones =  num_of_ones + 1;
            else
                num_of_zeros = num_of_zeros + 1 ;
            end
            if (mod(i,fs) == 0)
                if num_of_ones  > num_of_zeros
                    rec_bit_seq(1,n) = 1;
                else
                    rec_bit_seq(1,n) = 0;
                end
                n = n + 1;
                num_of_ones  = 0;
                num_of_zeros = 0;
            end
        end
        %%%
```

```
     case 'part_3'
         %%% WRITE YOUR CODE FOR PART 3 HERE
         rec_bit_seq = zeros ( 1 , length(rec_sample_seq)/fs)   ;
         n = 1;
         for i=1 : 5 : length(rec_sample_seq)-4
                rec_bit_seq (1, n) = rec_sample_seq (1, i) ;
                n = n + 1 ;
         end
         %%%
end
```

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Spring semester, 2021/2022

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثاني, 2022/2021

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
BER= sum ( xor (bit_seq , rec_bit_seq) )/length(bit_seq) ;
%%%


```

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Spring semester, 2021/2022

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثاني, 2021/2022

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)*.

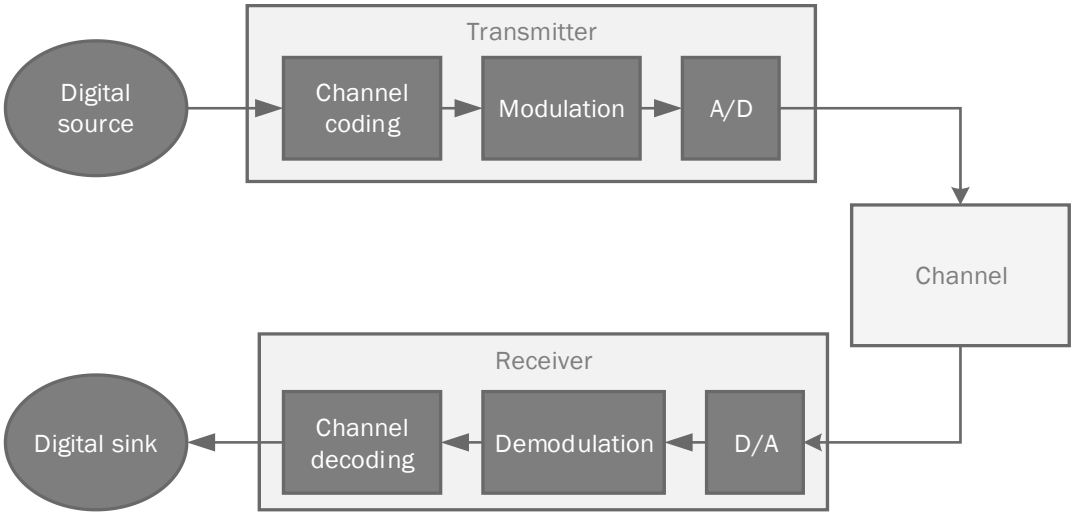


Figure 1 An example of a digital communication system

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 N 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 E
4. The BER is given by $BER = \frac{E}{N}$

Credits: Dr. Mohammed Karmoose

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 $b = \{0,1\}$, and produces the corresponding output according to the following equation.

$$y = \begin{cases} b & with\ probability\ 1-p \\ \overline{b} & with\ probability\ p \end{cases}$$

The channel described above simply flips the input bit with probability p or passes the input bit unchanged with probability $1-p$; 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.

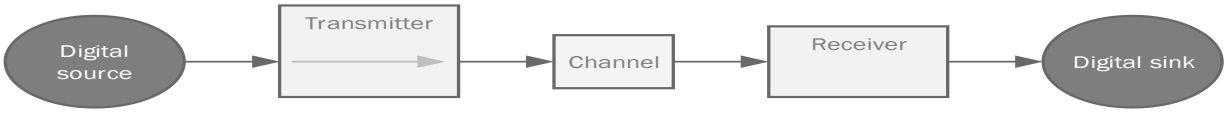


Figure 2 A digital communication system with a Binary Symmetric Channel

Your goal in this task is to design the receiver. You know that the channel takes the data, flips it randomly (with probability p) 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.	**It will depend on number of input bits (ones).**
What is the reason behind the performance of this receiver?	**The all bits which equal to 1 will be received as 0 and all bits which equal to 0 will be received as 0 even if it became as 1 through the channel.** **For example if we send the number ones equal to number of zeros (equiprobable) the BER will equal to 0.5.**

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.	**I cannot give a fixed number.**
What is the reason behind the performance of this receiver?	**Because every bit has probability of error equal to 0.5 so I cannot expect total bits which have error so BER will be different every time.**

The above two receivers are examples of receivers which clearly would not be considered as good receivers from a BER perspective (**why?**).

>> Because I cannot work on a system that sends only zeros or ones as in example one or a system that has not fixed BER as in example two.

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.	**Receiver 1 will depends on the input bits and receiver 2 will give unfixed BER.**
What is the reason behind the performance of these two receivers?	**Because it expects the output whatever the channel generate.**
What is the BER of the best receiver?	**In this case the BER approximately equals to "p" = 0.2 and "p" depends on channel as it becomes close to zero we can have the best receiver.**

Part 1-a (2 Marks)In this part, we study the impact of the BSC channel parameter p on the BER of the digital communication system. Namely, we vary the value of p from 0 to 1, and for each value of p 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 p.

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 b, the transmitter generates a set of 5 copies of the bit b 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?**).

>> **It is bad for channel capacity so we should send number of bits suitable for both channel capacity and error detection to assure that information content being input to the channel is no greater than the information carrying capability of the channel.**

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.

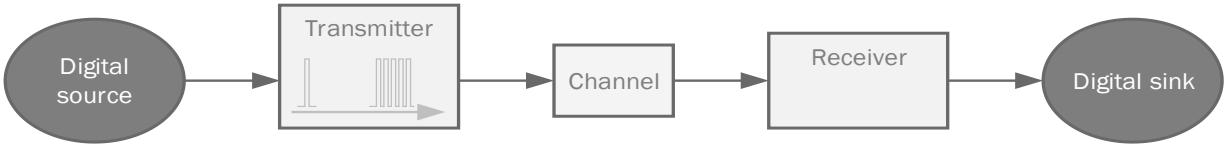


Figure 3 A digital communication system with a Binary Symmetric Channel and a modified transmitter

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?	**< 0.2**
What is the expected (theoretical) BER if the number of repetitions is increase to 10?	**BER will decrease corresponding to the more repetitions with probability of error = 0.2 for each repetitive bit.**

What is the cost/downside of using the transmitter in Part 2?	**The number of bits will increase which affect the channel capacity.**

Part 2-a (2 Marks)

Similar to Part 1-a, in this part, we study the impact of the BSC channel parameter p 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 $1-p$ or a set of five 1's with probability p. In this case, we investigate the design of the best receiver and the corresponding BER.

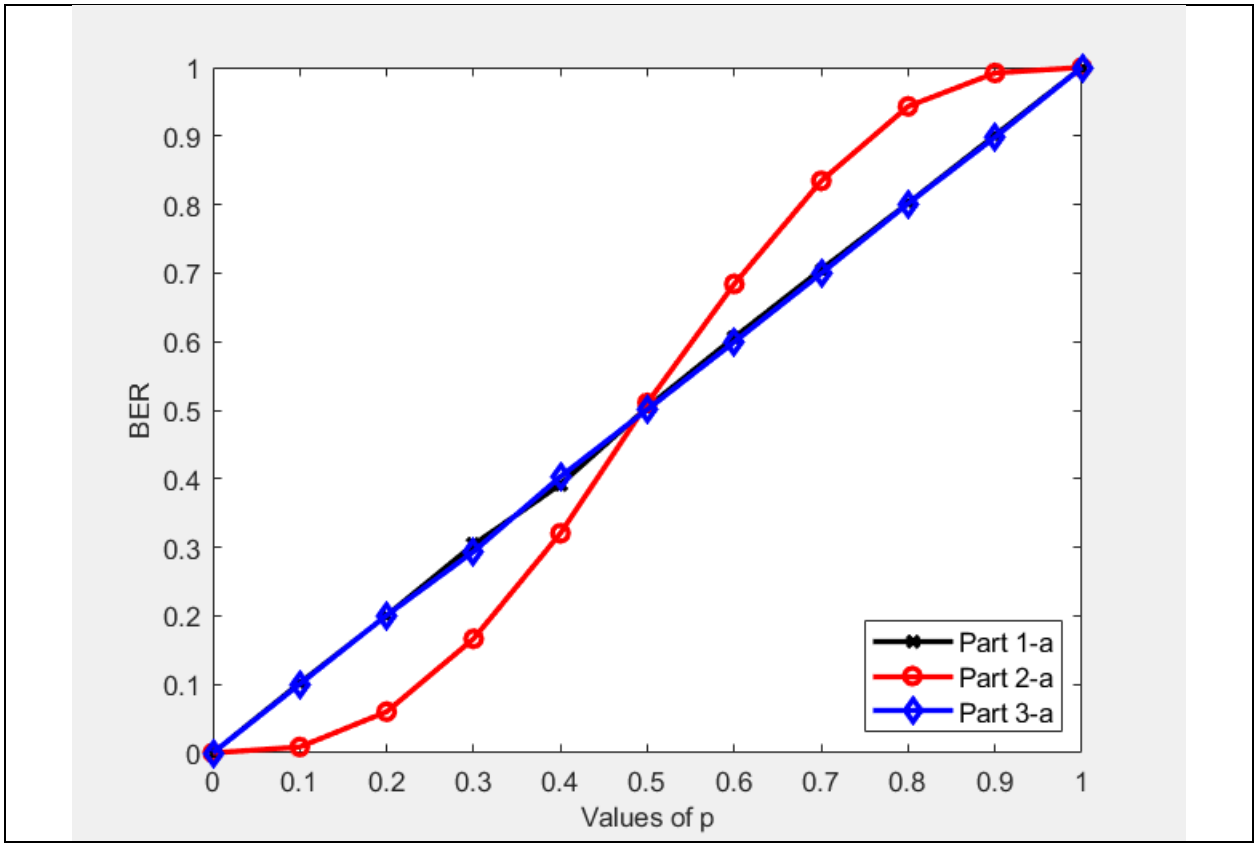
EXP. ***Complete PART 3 in the experiment M-file*** **Lab1_script.m** ***and the missing implementation of all included functions. Then answer the following questions:***

Questions	
What is the BER of the best receiver?	**For p = 0.2 → BER ≤ 0.2**
What is the reason behind such a performance?	**As the channel flips the first input bit with probability p= 0.2 and the rest of bits (4 bits) follow the first bit (correlated outputs). The 5 bits are the same so the repetition does not affect BER.**

Part 3-a (2 Marks)

Finally, we study the impact of the BSC channel parameter p 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.***



Questions	
Which of the three systems have the best performance in terms of BER?	**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 can add an inverter at the receiver to flip the received bits thus BER will increase as BER_new = 1 – BER.**

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.***