Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Fall semester, 2023/2024

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الأول, 2024/2023

DIGITAL COMMUNICATIONS LAB

Experiment 1

Basics of BER calculations and channel models

By:

Yahia Walid Mohamad Eldakhakhny (**19016891**)

Zyad Alaa Elsayed Goubashy (**19015728**)

Ragai Ahmed Abdelfattah Awad (**19015655**)

Experiment

Part 1

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.	**Given the channel input has 50% chance of 1s or 0s the BER = 0.5**
What is the reason behind the performance of this receiver?	**Each 1 input gives of an error, while each 0 input is correct, so always 50% of the bits are received incorrectly, assuming the channel input is of truly random 0s and 1s, if 1s are 70% of the input then the BER is 70%**

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.	**chance of 1s and 0s, the BER is 0.5**
What is the reason behind the performance of this receiver?	**Each 1 input has error probability of 50% and each 0 input has error probability of 50%, and at whatever percentage mix of 1s and 0s the error rate will be 50%**

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 depends on the number of 1s in the input (would be 0.5 in case of random input), receiver 2 is always 0.5**
What is the reason behind the performance of these two receivers?	**For receiver 1: Each 1 input gives of an error, while each 0 input is correct, so always 50% of the bits are received incorrectly, assuming the channel input is of truly random 0s and 1s, if 1s are 70% of the input then the BER is 70%** **For receiver 2: Each 1 input has error probability of 50% and each 0 input has error probability of 50%, and at whatever percentage mix of 1s and 0s the error rate will be 50%**
What is the BER of the best receiver?	**Last receiver BER = 0.207**

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Fall semester, 2023/2024

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الأول, 2024/2023

Part 2

Questions	
What is the BER of the best receiver?	**The best receiver with 5 repetitions has a better BER at lower p values than 0.5 specifically at p = 0.2 the BER =0.0608**
What is the expected (theoretical) BER if the number of repetitions is increase to 10?	**Theoretically it should reach 0.049, by analysis of the valid bit condition from symbols (1-2*p) = (1-2*BER)^(N/2)**
What is the cost/downside of using the transmitter in Part 2?	**The number of repetitions means using more resources whether time or bandwidth**

Part 3

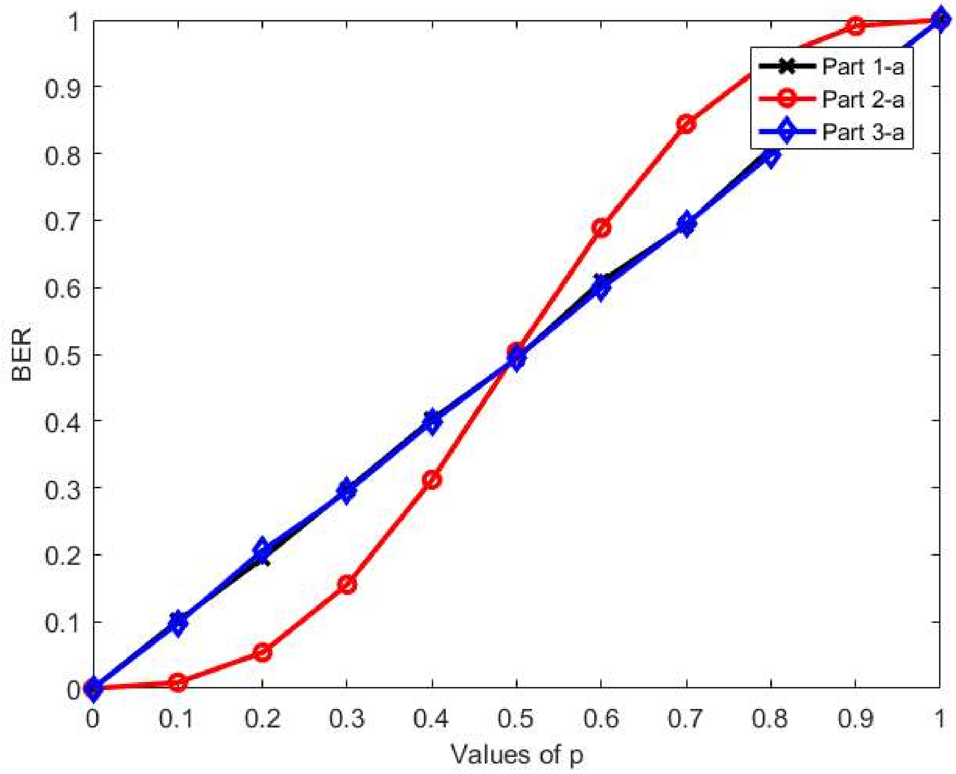
Questions	
What is the BER of the best receiver?	**The last receiver achieved similar results to the part 1 -a receiver with approx. 0.2 BER, the best so far is part2a**
What is the reason behind such a performance?	**The correlation between bits make them reducible to one bit similar to part1-a case**

Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Fall semester, 2023/2024

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الأول, 2024/2023

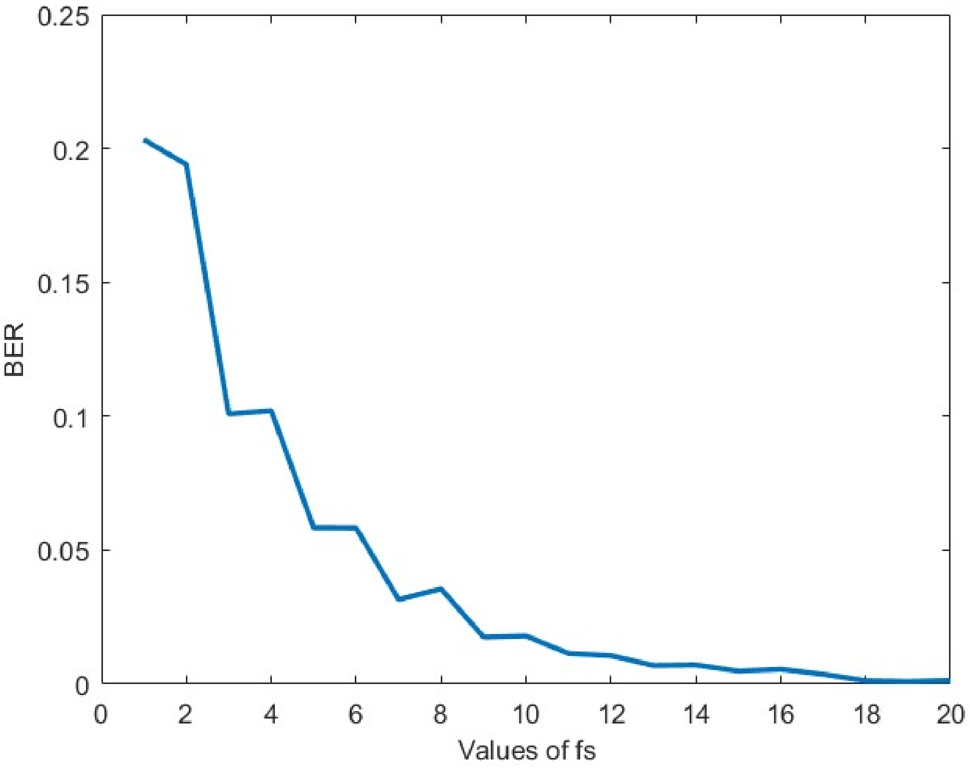
Part 3-a

Please insert the plot here



Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Fall semester, 2023/2024

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الأول, 2024/2023



Which of the three systems have the best performance in terms of BER?	**Part 2-a with 5 repetitions**
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?	**Just flipping the logic of the system, would yield new BER of 1- old BER**

Alexandria University Faculty of Engineering Electrical and Electronic Engineering Department

Course: Digital Communications Lab

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

Contents

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); %[DONE] 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');  % IMPLEMENT THIS: Decode th
e received bits

% Compute the BER
BER_case_1 = ComputeBER(bit_seq,rec_bit_seq); %[DONE] IMPLEMENT THIS: Calculate the bit er
ror 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
```

```
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');
    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); %[DONE] IMPLEMENT THIS: Generate a sequence of s
amples for each bit

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

% Decode bits from received bit sequence
rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_2',fs);    %[DONE] 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
```

```
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',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

```
fs=5;
```

```
% 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',fs);    % IMPLEMENT THIS: Deco
de the received bits

% Compute the BER
BER_case_3 = ComputeBER(bit_seq,rec_bit_seq);  % Calculate the bit error rate
```

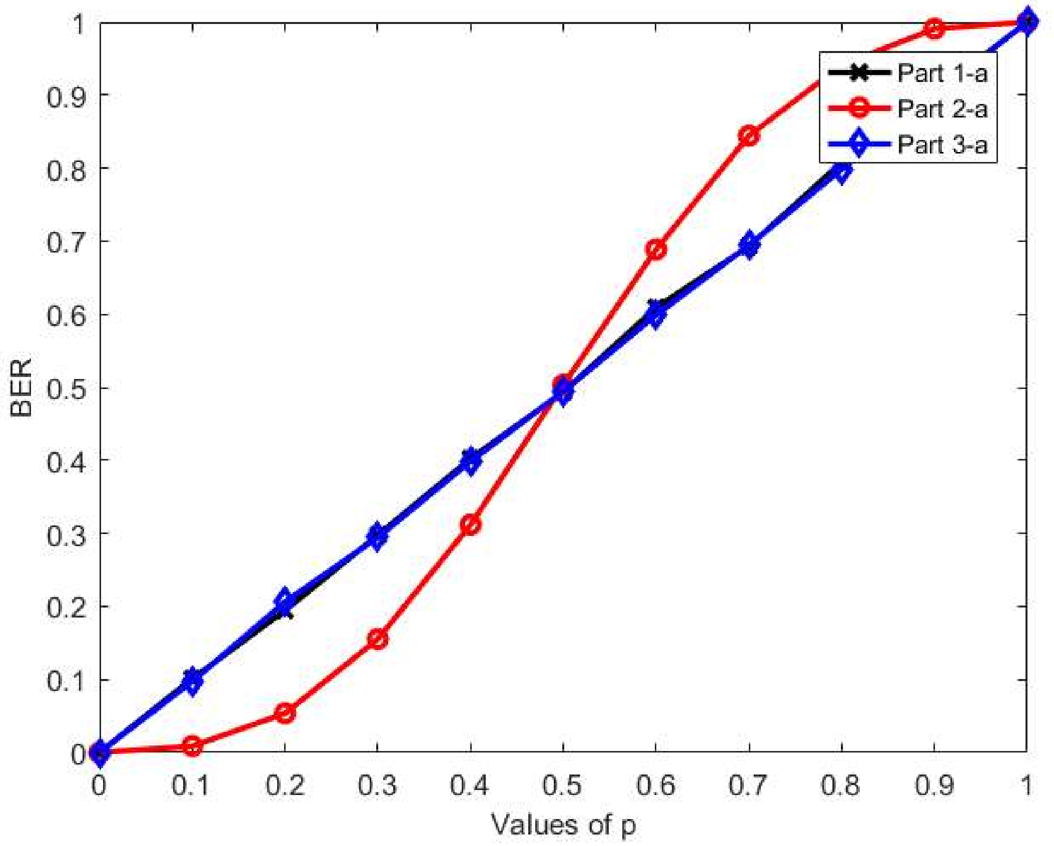
Part 3-a: Effect of bit flipping probability on BER

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
```

```
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
```

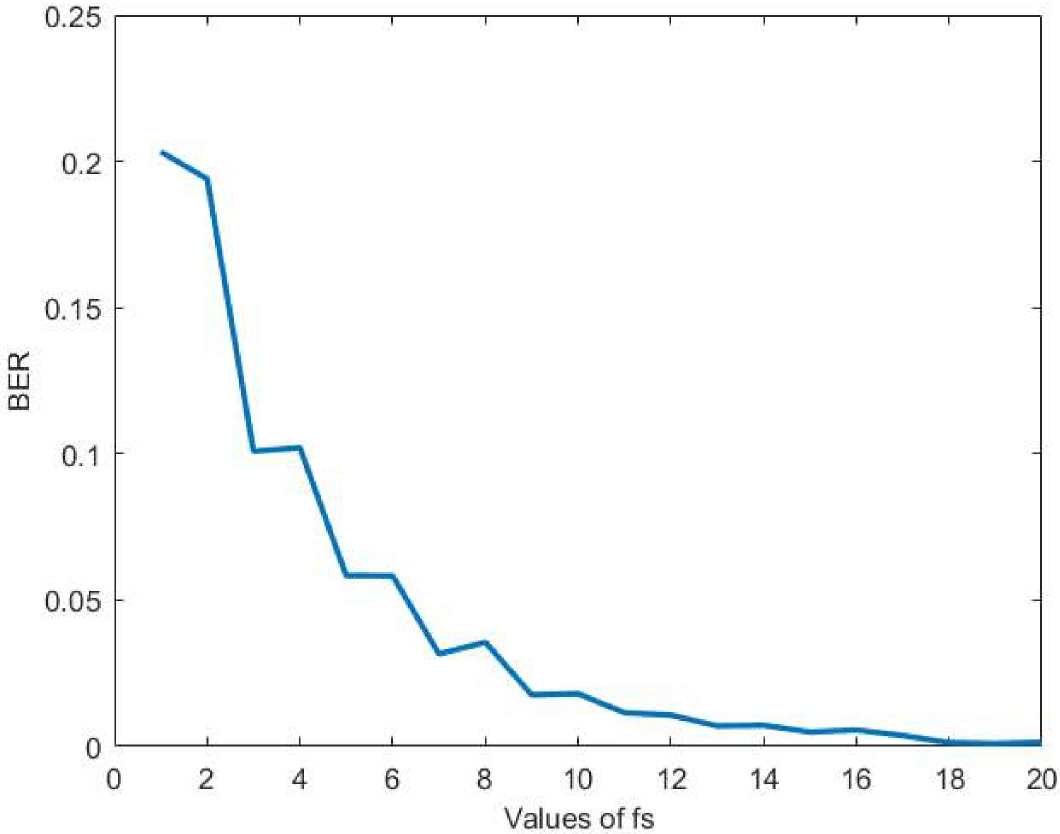
```
% Plotting results

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

```
% Generate a bit sequence
fs_vect          = 1:20;              % Use this vector to extract different values of p i
n your code
p=0.2;
BER_case_4_vec   = zeros(size(fs_vect));  % Use this vector to store the resultant BER

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


for fs_ind = 1:length(fs_vect)
    % Generate samples from bits
    sample_seq = GenerateSamples(bit_seq,fs_vect(fs_ind)); %[DONE] IMPLEMENT THIS: Generat
e a sequence of samples for each bit

    rec_sample_seq = BSC(sample_seq,fs_vect(fs_ind),p);

    rec_bit_seq = DecodeBitsFromSamples(rec_sample_seq,'part_2',fs_vect(fs_ind));

    BER_case_4_vec(fs_ind) = ComputeBER(bit_seq,rec_bit_seq);
end

figure
plot(fs_vect,BER_case_4_vec,'linewidth',2); hold on;

xlabel('Values of fs','fontsize',10)
ylabel('BER','fontsize',10)
```


0.25
0.2
0.15
0.1
0.05
0
BER
0
2
4
6
8
10
12
14
16
18
20
Values of fs

```
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 = [];
```

```
    for index = 1:length(bit_seq)
        if bit_seq(index) == 1
            sample_seq = [sample_seq ones(1, fs)];
        else
            sample_seq = [sample_seq zeros(1, fs)];
        end
    end
```

```
end
```

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

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

```
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
```

```
counter =0;
l=length(bit_seq);
```

```
for i=1:1:l
    if bit_seq(i)~=rec_bit_seq(i)
    counter=counter+1;
    end
end
BER=counter/length(bit_seq);
end
```

```
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

switch channel_type

    case 'independent'
        channel_effect = rand(size(rec_sample_seq))<=p;
    case 'correlated'
        channel_effect = rand(1,length(rec_sample_seq)/fs)<=p;
        channel_effect = repmat(channel_effect,fs,1);
        channel_effect = channel_effect(:)';
end

rec_sample_seq = xor(sample_seq,channel_effect);
rec_sample_seq = rec_sample_seq + 0;
```