Assignment: Digital Communication

TITLE: Buliding Virtual Communication channel and performance analysis of different modulation and coding scheme using MATLAB

Prepared by

Student Name : Shahnewaz Ahmed
Student ID: 1206004
Date of Submission: 03.06.17

Course Title: Digital Communication
Instructor Name:
Shiekh Zia Uddin
Lecturer, EEE, BUET

ABSTRACT

In this assignment. The requirement was to build a full virtual digital communicaton channel using source, channel and line coding scheme. In this report, the performance of coding scheme is being compared and implementation of different blocks of the digital communication channel in MATLAB is discussed with optimization notes. The whole system is simulated using MATLAB 2013, GNU octave-4.0.0 and jupyter notebook for measuring different section and blocks of the digital communication channel. In this report I introduced different performance measure to find out the the suitable coding scheme for different SNR and shortcoming of the other techniques.

KEYWORDS

1. Virtual Digital Communication Channel
2. Different Coding Scheme
3. Optimize Implementations
4. Performance Measure
5. Short Coming of Different Techniques

Table of Contents

1 INTRODUCTION

Human race has been experienced four different era of civilization. The modern era is called the era of information. The new era has commence from the invention of the computer and internet technology. The digitizating Information is one of the major achievement of this era and this can be called as the corner stone of this information age. Now it is important for communication engineers to study the fundamental difficulties regarding digital communication and the techniques are already developed to overcome this difficulty to create new improvised techniques to overcome the difficulties in these day and age.

In this assignment, we are implement the virtual digital communication channel using MATLAB and study the performance of different coding scheme. The construction of the report is as follows, first, the required theory for different block will be discussed with the the short description of the algorithms used to implement the respective blocks will be discussed. After that different question in the question will be answered from the simulated result and in the last section I discussed the different problem I have faced to inplement this system and possible solutions regarding this problems.

2 THEORY

2.1 Digital Communication Channel

In this diagram below general block-diagram of the digital communication channel is shown.

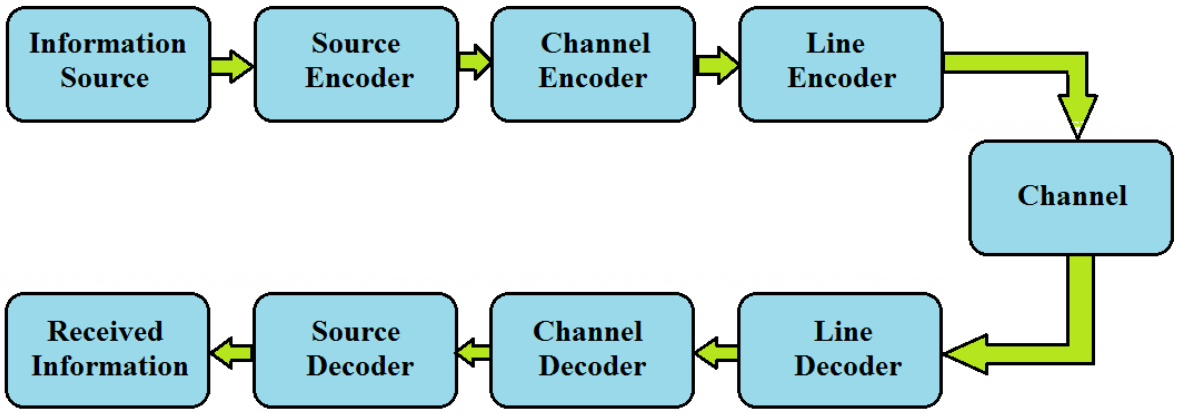


Figure 1: Block Diagram of Digital Communication Channel

2.2 Information Source

For this assignment the information source was file source_data.txt which is actually **The Double Helix : A Personal Account of the Discovery of the Structure of DNA** which is an autobiographical account of the discovery of the double helix structure of DNA written by James D. Watson and published in 1968.

```
In [1]: function [str, symbols, freq , Pr] = source_stat()

          str = fileread('source_data.txt');

        %   str = fileread('test.txt');

          symbols = unique(str);

          freq = histc(str, symbols);

          n = length(str);

          Pr = freq/n;

        end
```

This source_stat() MATLAB function is used to find the extensive source statistics to encode the source information. Mainly the MATLAB function unique() provides necessary information about the source symbol. With the help histc() function frequency of the symbols are being counted, which is used to create Probablities of different symbols.

2.3 Source Coding

In this model of communication of channel we used Huffman Coding for source coding scheme. This coding scheme is particularly very good if the source statistics is known prior to encode. This famous data compression algorithm use the source statistics and assign the less frequent symbol with larger bit string and vice versa. The dictionary building in Huffman coding only based on the statistics of the symbols and each encoded symbol has the encoded with optimal prefix code. In this example below the Huffman coding scheme is pictured and Huffman tree has been shown

P (a_1)	1/2
P (a_4)	1/4
P (a_2)	1/8
P (a_3)	1/8

a_1	0
a_2	110
a_3	111
a_4	10

Figure 2: 4-symbol source statistics and Huffman prefix coding

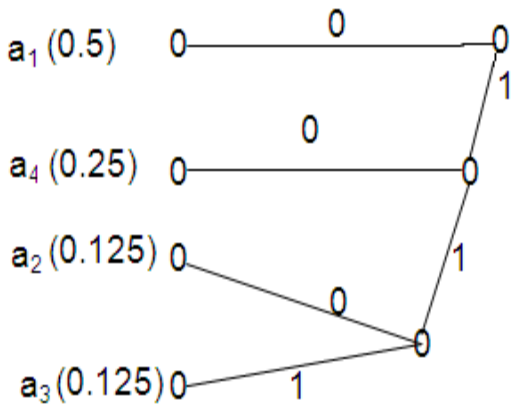


Figure 3: Huffman Tree

From this diagram it is clear that the probalities of the smallest probable symbols are added and 1 and 0 are added to their code word and this process continues until the probabilities are added up to 1.

The MATLAB implementation is done using two steps , first the dictionary building and the sorce coding using this dictionary. The dictionary is built using Huffman tree which will prove to be convenient for decoding optimization.

Optimization Notes: The vectorize Technique in MATLAB is very efficiently implemented here to optimize the encoding time.

2.4 Channel Coding

The source coding is used to eliminate the redundancy in the information source but the channel coding is used to put redundant bit of information to ensure that after the signal corrupted by the noise the information can be decoded from the corrupted signal with greater accuracy. The generalized convolution coding scheme is used here , A sample block diagram is shown here:

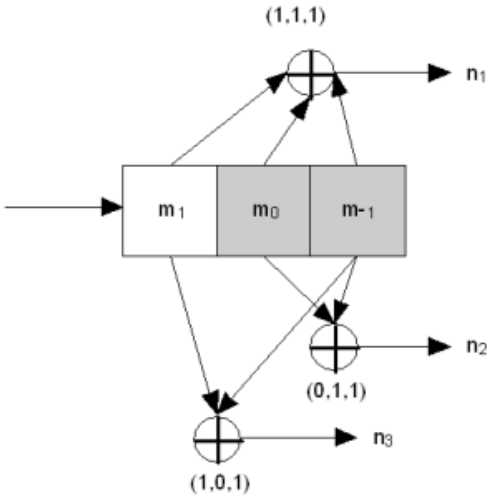


Figure 4: Rate 1/3 non-recursive, non-systematic convolutional encoder with constraint length 3

Optimization notes: MATLAB already has conv() function which is used here to implement the code and if the number of bit shift is greater than 1 than the down-sampling is done in a vectorized way to implement the find the result.

In [1]:
```
function [encoded_bits] = convolution_coding(bit, generators, k)

    [n, block_size] = size(generators);
    [number_of_bit_row, bit_length] = size(bit);
    encoded_bits = [];

    for j = 1:number_of_bit_row
        coded_string = zeros(n, bit_length + block_size - 1);
        for i = 1:n
            coded_string(i,:) = mod(conv(bit(j,:),generators(i,:)),2);
        end
        [~, l] = size(coded_string);
        coded_string = coded_string(:,mod(1:l,k)==0);
        encoded_bits(j,:) = coded_string(:)';
    end

end

pkg load signal
```

2.5 Line Coding

The actual communication channel behaves like a Bandpass filter and the modulation with high frequency is necessary to pass the signal inside the channel. The line coding scheme is set for this work. In this assignment, MASK, MPSK, MFSK and GMSK line coding scheme is asked to be implemented. Here is the short description each technique.

MASK:

The multiple bit are stick together to make a symbol by using gray code scheme. Now M number of different amplitude level carrier wave is used to modulate the base band line coded signal and that is transmitted through the communication channel. The analog characteristics of the channel distorts the signal but demodulation technique can be used to recover the signal.

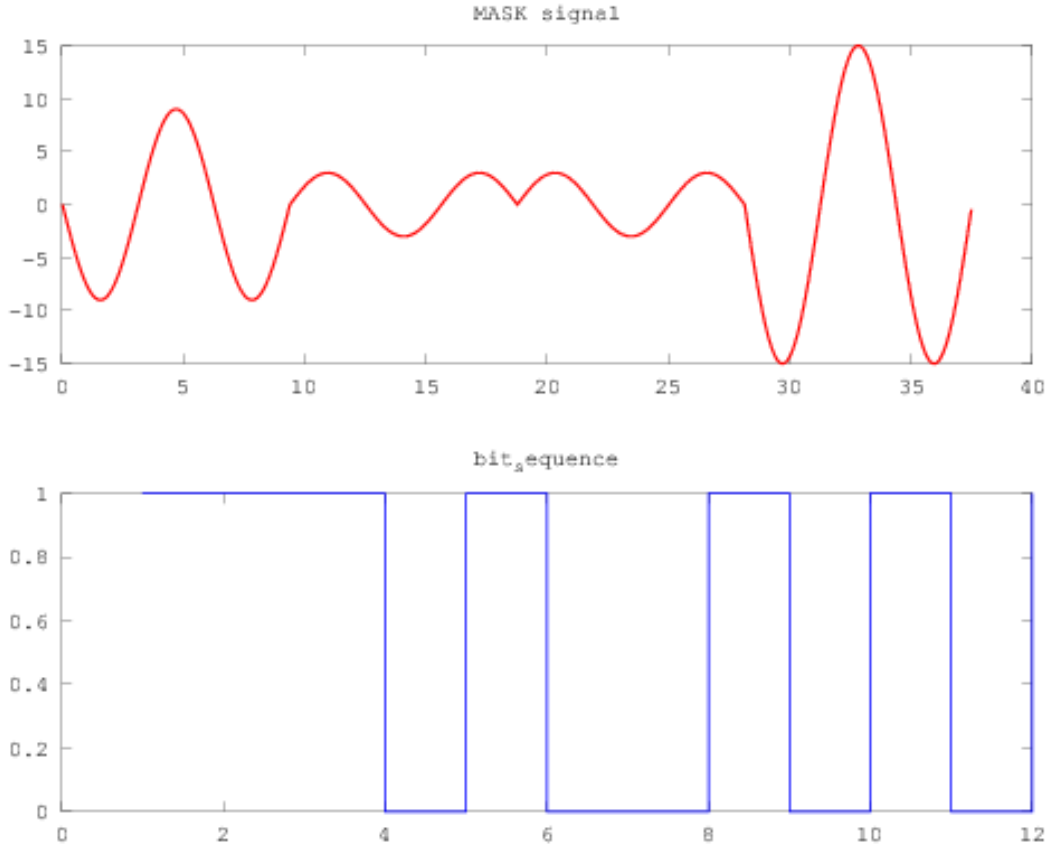


Figure 5: MASK signal with bit stream

MPSK:

In the signal dimension the MPSK signal is two dimensional and the Information of the symbol is encoded inside the phase which can be representated in the constellation diagram with gray encoding.

MFSK:

MFSK signal is similar in concept with the other two counterparts but the signal is Multidimentional and its orthogonality condition is assumed in theoretical discussions. The frequency is being increased based on the symbols as discussed above.

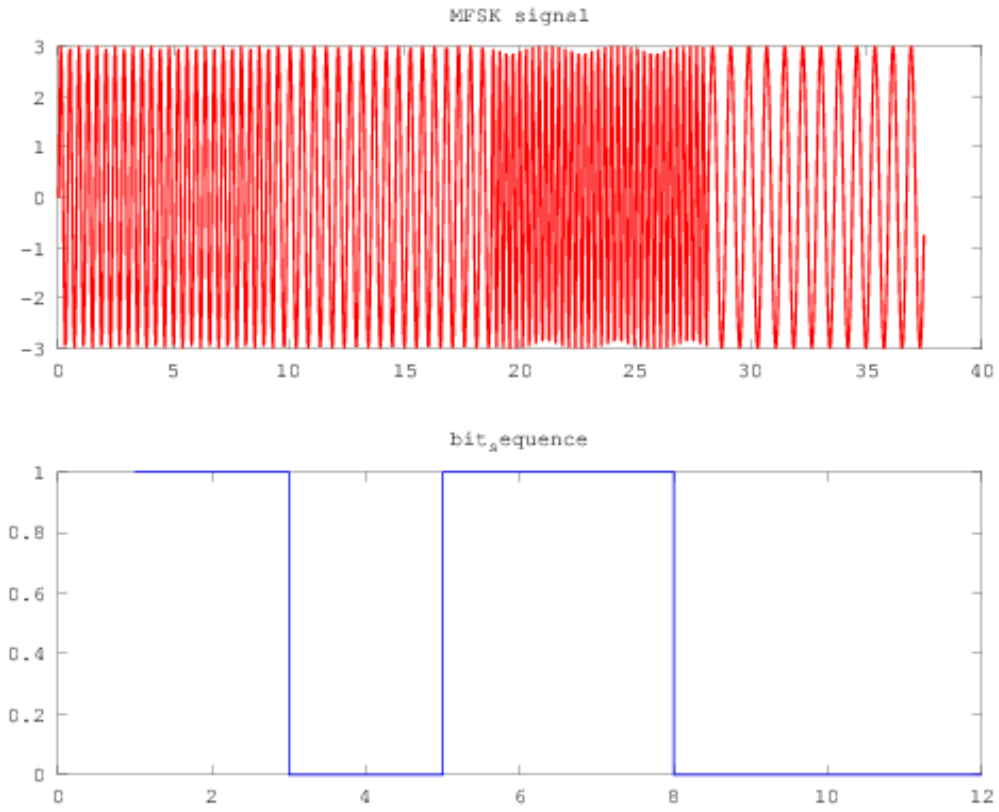


Figure 6: MFSK signal with bit stream

GMSK:

GMSK is quite different from other three modulation scheme and No symbol is used here and the Gaussian is used to modulate the integrated signal and then it is modulated. The special thing about this modulation is it is continuous phase modulation.

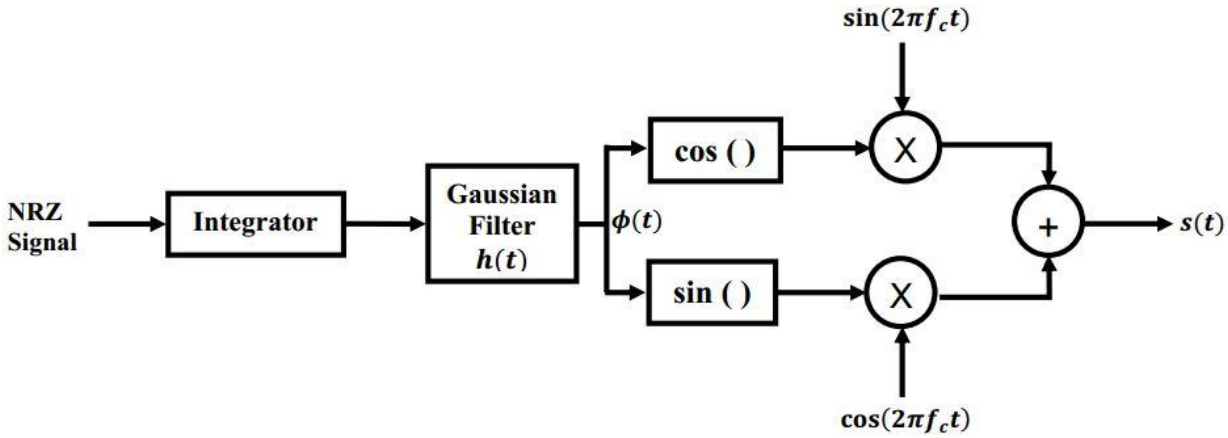


Figure 7: GMSK modulator

Optimization Notes: To reduce the search space the clever indexing method is introduced to index gray code to reduce the search time for the symbol.

In [76]:
```
gray_code = gray_code_generator(k);
gray_code1 = bin2dec(char(gray_code + 48));
test(1,gray_code1+1) = 1:M;
```

In [74]:
```
SIN =  sin(2*pi*frequency(1)*per_symbol_time_window);

modulated_signal = zeros(1, temp*len);

j = 1;
for i = 1:k:temp*k
    ind = test(bin2dec(char(bit(i:i+k-1)+48))+1);
    modulated_signal(j:j+len-1) =  amplitude(ind)*SIN;
    j = j + len;
end

%plot(modulated_signal)
```

In [1]:
```
function [gray_code] = gray_code_generator(dimensions)

  gray_code = [1;0];
  if dimensions ~= 1
    for i = 1:dimensions-1
        l = length(gray_code);
        gray_code = [flipdim(gray_code,1) , ones(l,1); gray_code, zeros(l,1)];
    end
  end
  gray_code = flipdim(flipdim(gray_code,2),1);
end
```

2.6 AWGN Channel

The simple awgn channel is implemented here to add noise to the digitaly modulated signal.

```
In [1]: function [ received_signal ] = awgn_channel( modulated_signal, SNR )

            received_signal = awgn(modulated_signal, SNR, 'measured');

        end
```

2.7 Line Decoding

MASK:

The MATLAB code for MASK signal decoding is used here is very simple. The received signal is modulated using same carrier wave and integrated. The the amplitude is compared with the other amplitude levels to find the signal vector and corresponding gray code is attached to find the original signal.

MPSK

For MPSK demodulation, the received signal is both modulated using sin and cosine carrier wave and integrated to find the inphase and quadrature component respectively. Again the colsest two dimensional signal is taken as the required string.

MFSK

The different frequencies are considered orthogonal to each other and by taking dot product in signal dimension the original is decoded from the noisy received signal.

GMSK

The GMSK demodulator is shown below, due the huge bit strean the GMSK modulation is quite difficult task here.

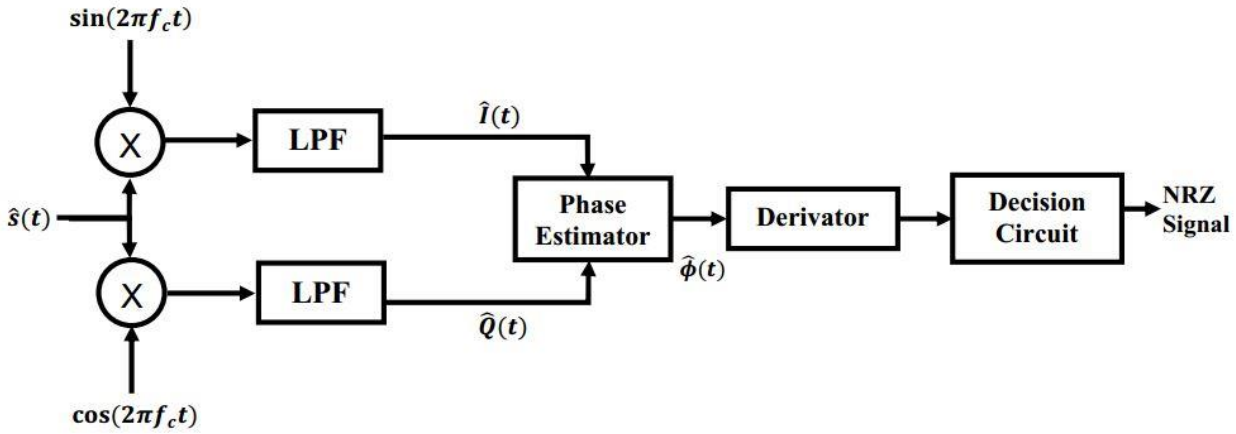


Figure 8: GMSK Decoder

2.8 Viterby Algorithm

The brute force algorithm can be used to find the lowest hamming distance form the all possible string . This dynamic programming programming problem is min-max problem and Viterby algorithm is an iterative technique for this solution of the problem. The dp array is considered here using the trellis diagram and the minimum path is found through the forward traversing and findind cost of the path and traversing back from the lowest cost path. The terminology here used in text is path metric and brach metric, which is the path traversing cost of the problem. The formula for it is:

$$PM[s, i+1] = \min(PM[\alpha, i] + BM[\alpha \rightarrow s], PM[\beta, i] + BM[\beta \rightarrow s])$$

And the iteration is initialize using the infinity and calculating the path metric in each iteration and traverse back with lowest cost path,

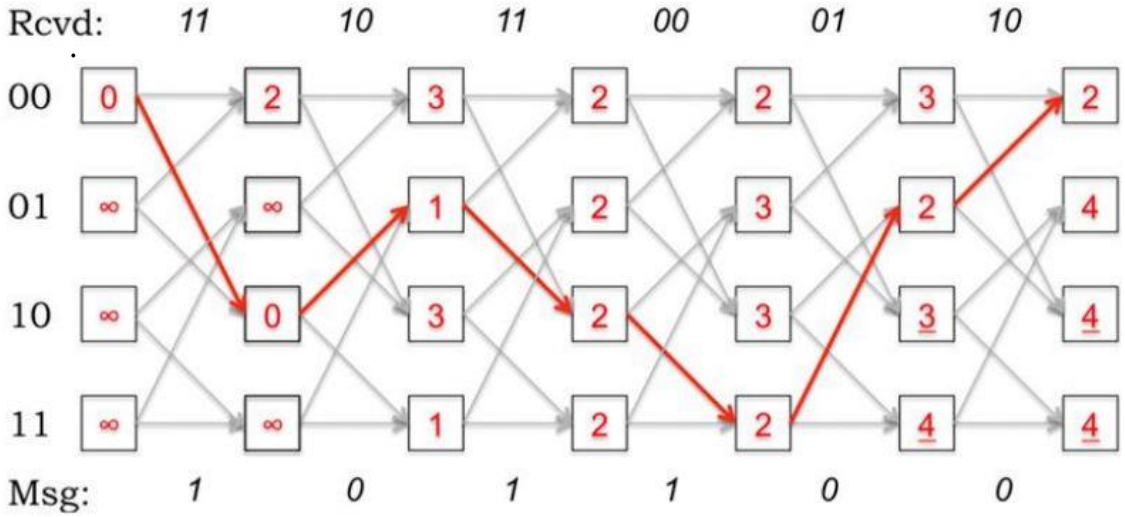


Figure 9: Minimum Hamming Distance in Trelis Diagram

Optimization Notes:

The technique to save the both source and destination of trellis is saved by a single intger in the index of the solution, which is kindof resemble the bit mask dp, although it is not the same.

```
flag = 1;
path_metric = zeros(col, 2*length_of_recieved_stream / blocks + 2);
path_metric(:,1:2) = inf*ones(col,2);

path_metric(1,1) = 0;

for i = 1:blocks:length_of_recieved_stream

        checker_matrix = meshgrid(received_stream(i:i+blocks-1), ones(row,1));
        branch_metric = sum(xor(checker_matrix , possible_solutions),2);

        result_matrix = inf * ones(col,2);

        for j = 1:row

            if (branch_metric(j)+path_metric(ceil(j/k_len),flag)) < result_matrix(possible_transitions(j)+1,1);
                result_matrix(possible_transitions(j)+1,1) = (branch_metric(j)+path_metric(ceil(j/k_len),flag));
                result_matrix(possible_transitions(j)+1,2) = j;
            end

        end
        path_metric(:,flag+2:flag+3) =  result_matrix;
        flag = flag + 2;
end
k_len;
path_metric;
```

2.9 Huffman Decoder:

Huffman decoder can be implemented using representing the dictionary with tree structure. And it is also convenient to search through the dictionary using binary spanning tree, which is implemented here. That reduces the algorithm time at logarithmic searching time.

In [2]:
```
function tree = dict2tree(dict)

  L = length(dict);
  lengths = zeros(1, L);

  for i = 1:L
    lengths(i) = length(dict{i});
  end
  m = max (lengths);

  tree = zeros(1, 2^(m+1)-1)-1;

  for i = 1:L
    pointer = 1;
    word = dict{i};
    for bit = word
      pointer = 2 * pointer + bit;
    end
    tree(pointer) = i;
  end

end
```

3 RESULTS

1. Effect of SNR on Noise performance of different Line Coding:

The tabulated result shows the maximum SNR for different line coding scheme the bit error or symbol error is zero and 40%.

For 0% bit error Rate

Method	M = 2	M = 4	M= 8	M = 16
MASK	-10	-5	0	5
MPSK	-10	-10	-5	0
MFSK	-5	-10	-5	-10
GMSK	>10	>10	>10	>10

For 40% bit error Rate

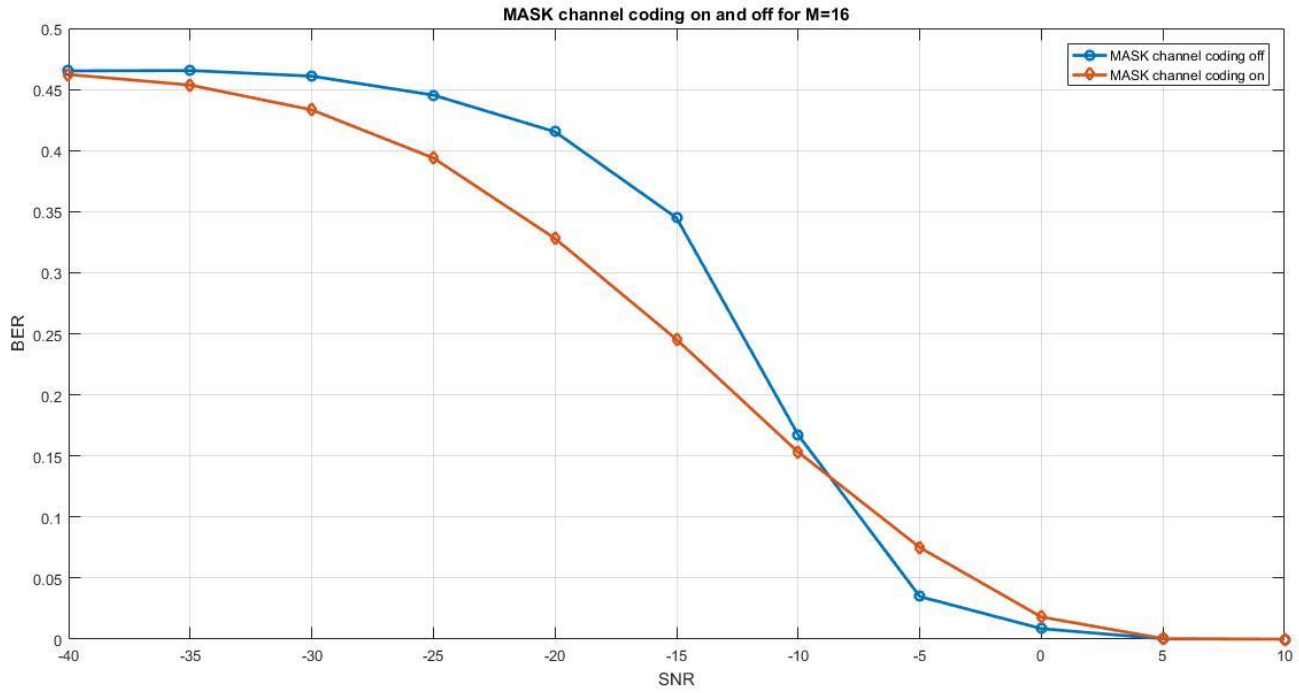
Method	M = 2	M = 4	M= 8	M = 16
MASK	-22	-23	-22	-18
MPSK	-27	-27	-25	-22
MFSK	-27	-24	-25	-25
GMSK	-5	-5	-5	-5

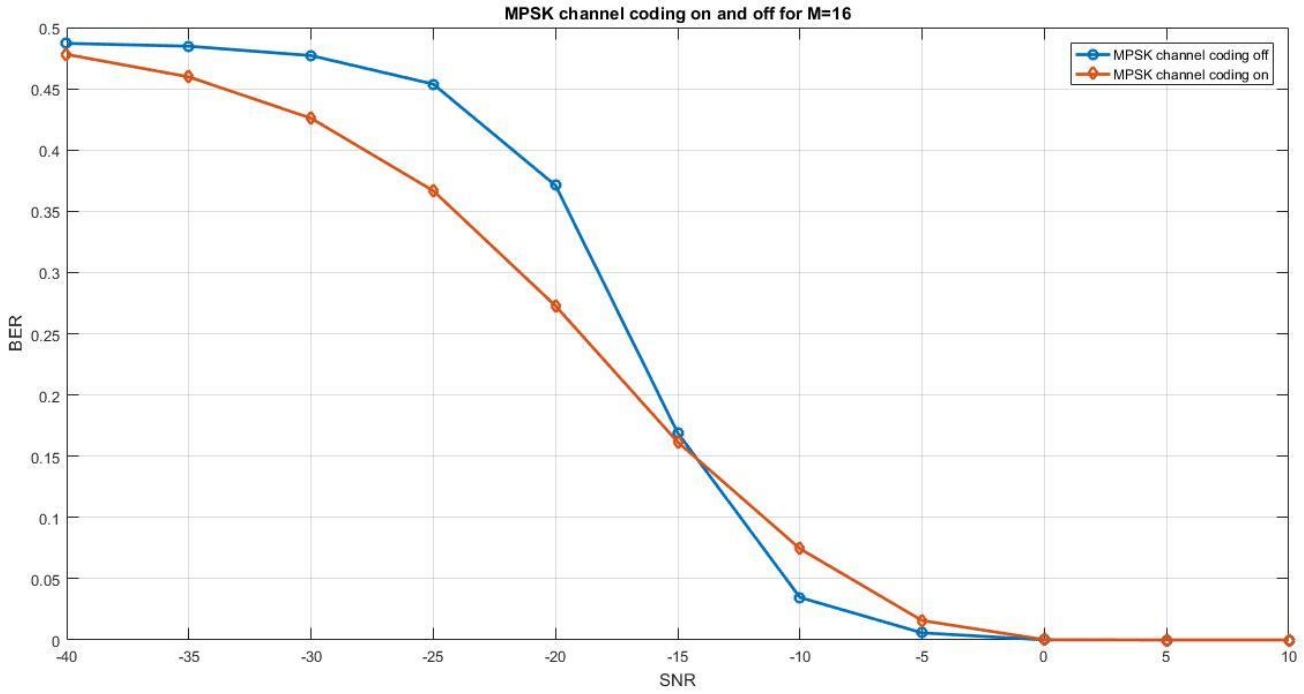
Discussion:

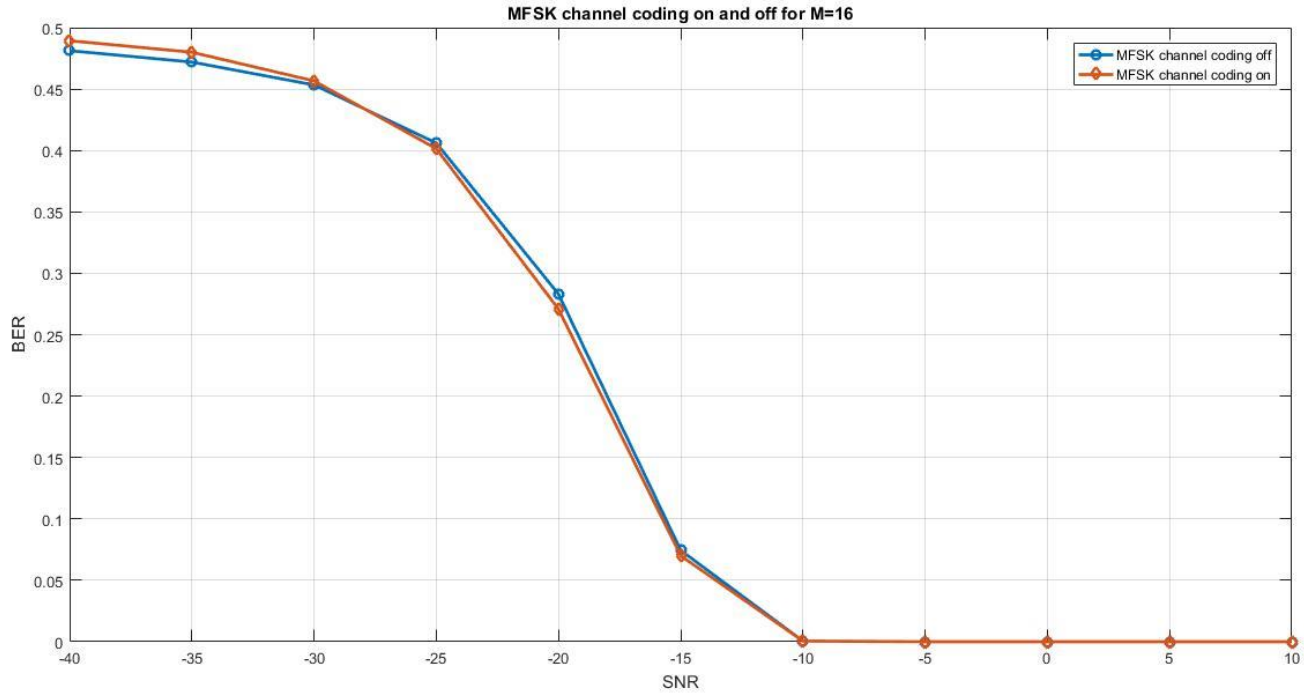
The table indicates a theoretical concept about channel and SNR. If we increase SNR, the information in the signal gets less corrupted. Hence, the error rate also decreases.

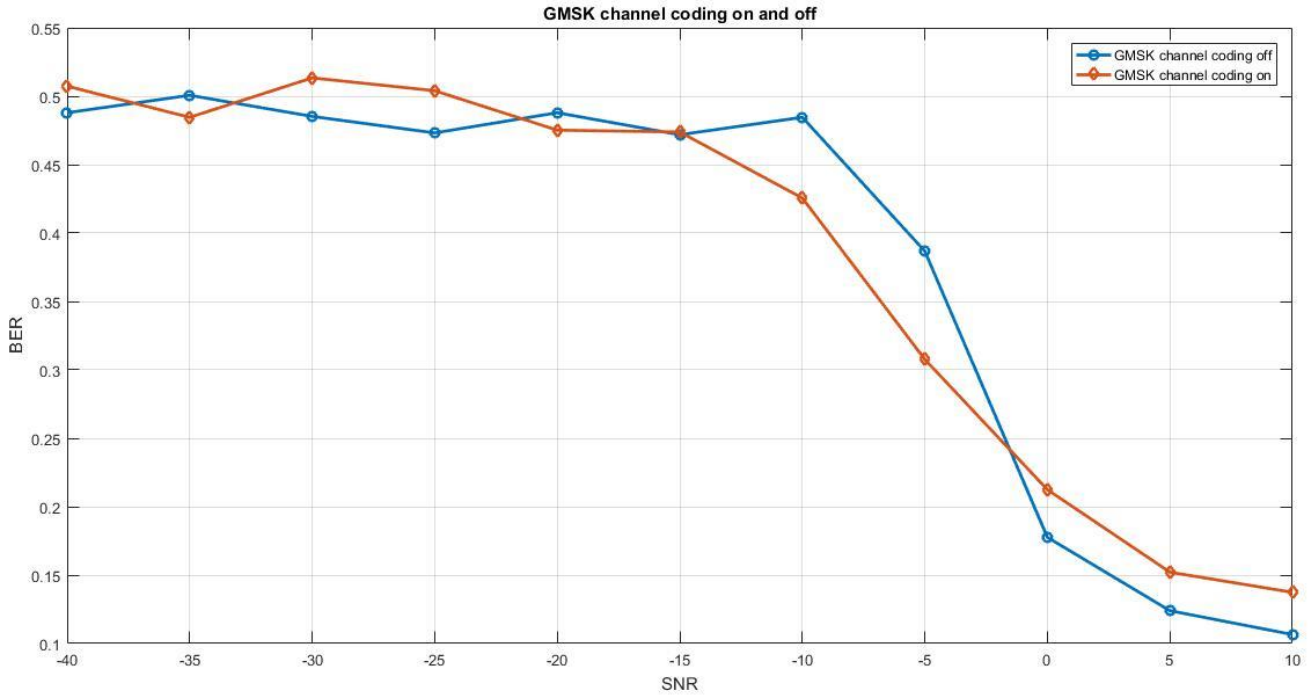
2. Effect of Convolution Coding in BER

Channel coding performance on different coding scheme is given here.






MFSK channel coding on and off for M=16
MFSK channel coding off
MFSK channel coding on
BER
SNR
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-40
-35
-30
-25
-20
-15
-10
-5
0
5
10

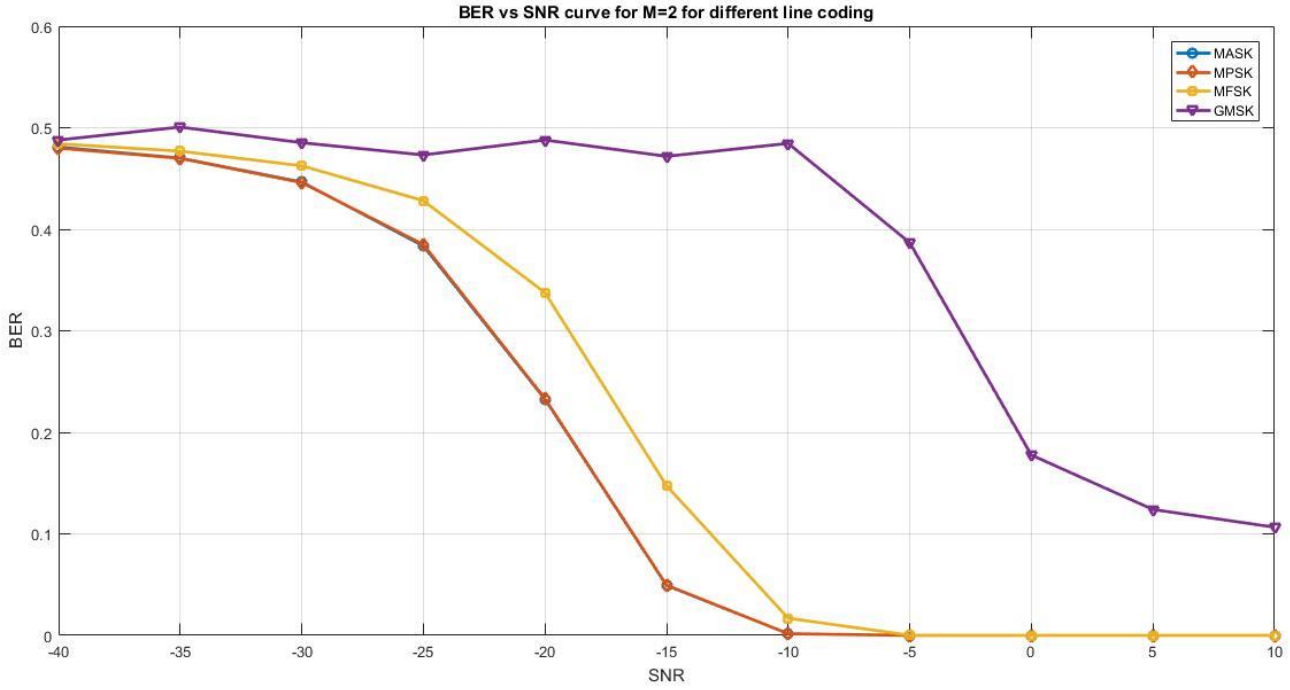

GMSK channel coding on and off
GMSK channel coding off
GMSK channel coding on
BER
SNR
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
-40
-35
-30
-25
-20
-15
-10
-5
0
5
10

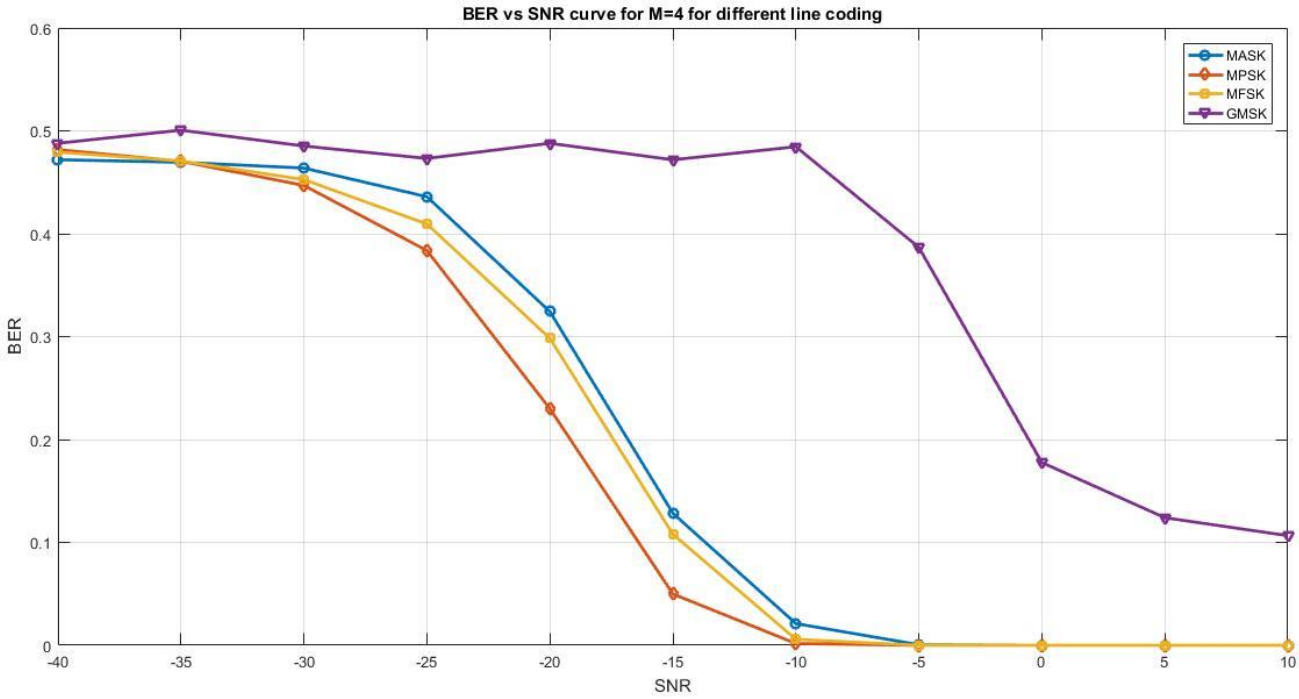
1206004

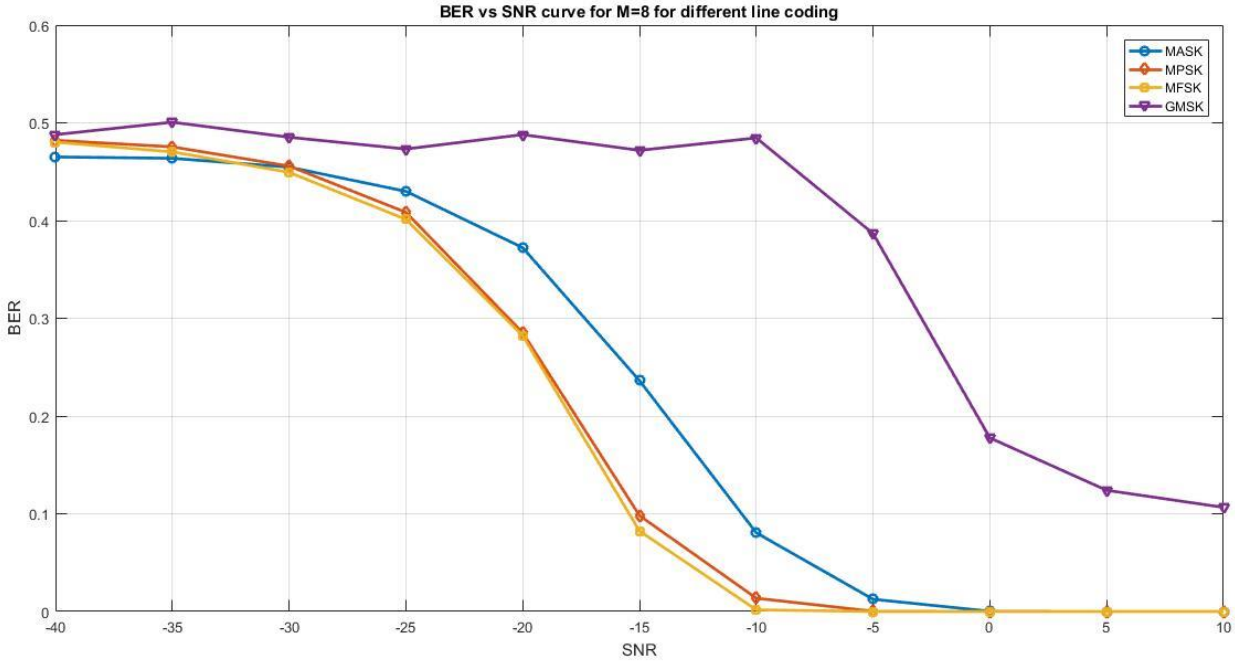
Discusssions: These plots shows that regardless of line coding scheme, the convolution coding scheme performs better for high SNR but performs poorer for low SNR. Convolution coding assumes that the error does not effect the hamming distance by a great margin. It tries to minimize the hamming distance while decoding. But at low SNR, noise causes a lot of error. So the assumption is violated and convolution coding cannot provide a better solution.

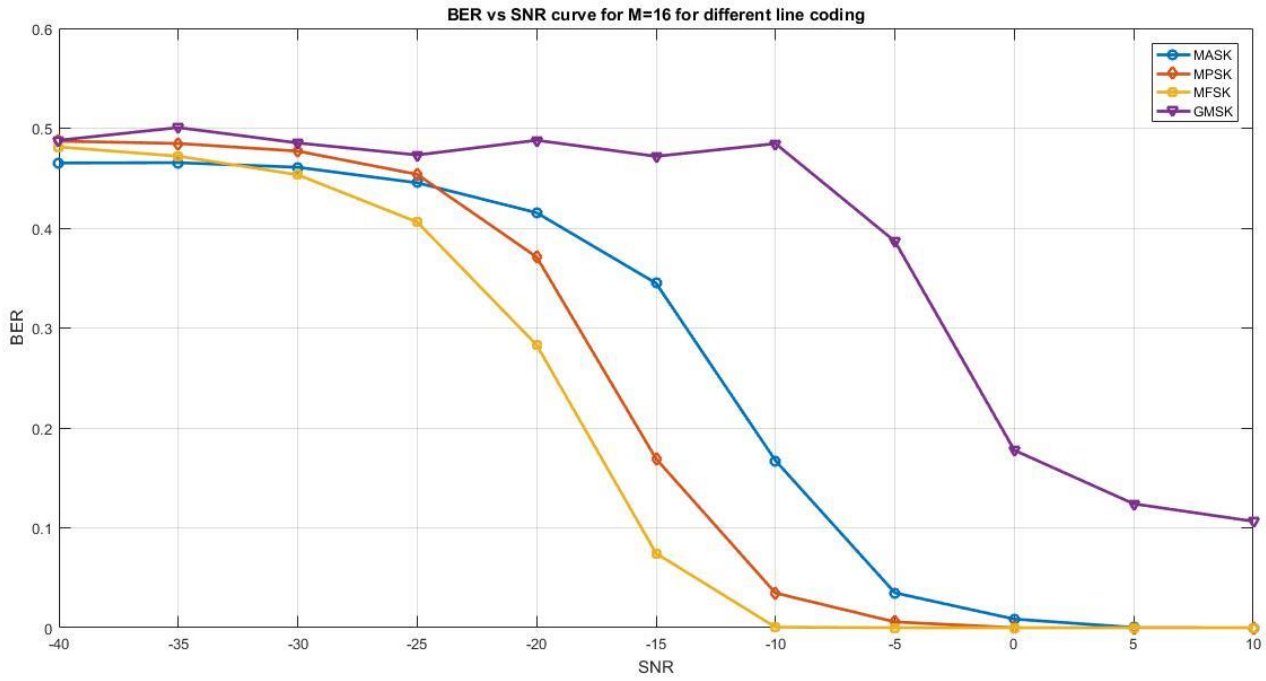
3. Effect of Line Coding in BER

For Different line Coding scheme and BER vs SNR is given here




BER vs SNR curve for M=4 for different line coding
BER
SNR
MASK
MPSK
MFSK
GMSK
0.6
0.5
0.4
0.3
0.2
0.1
0
-40
-35
-30
-25
-20
-15
-10
-5
0
5
10


BER vs SNR curve for M=8 for different line coding
BER
SNR
MASK
MPSK
MFSK
GMSK
0.6
0.5
0.4
0.3
0.2
0.1
0
-40
-35
-30
-25
-20
-15
-10
-5
0
5
10

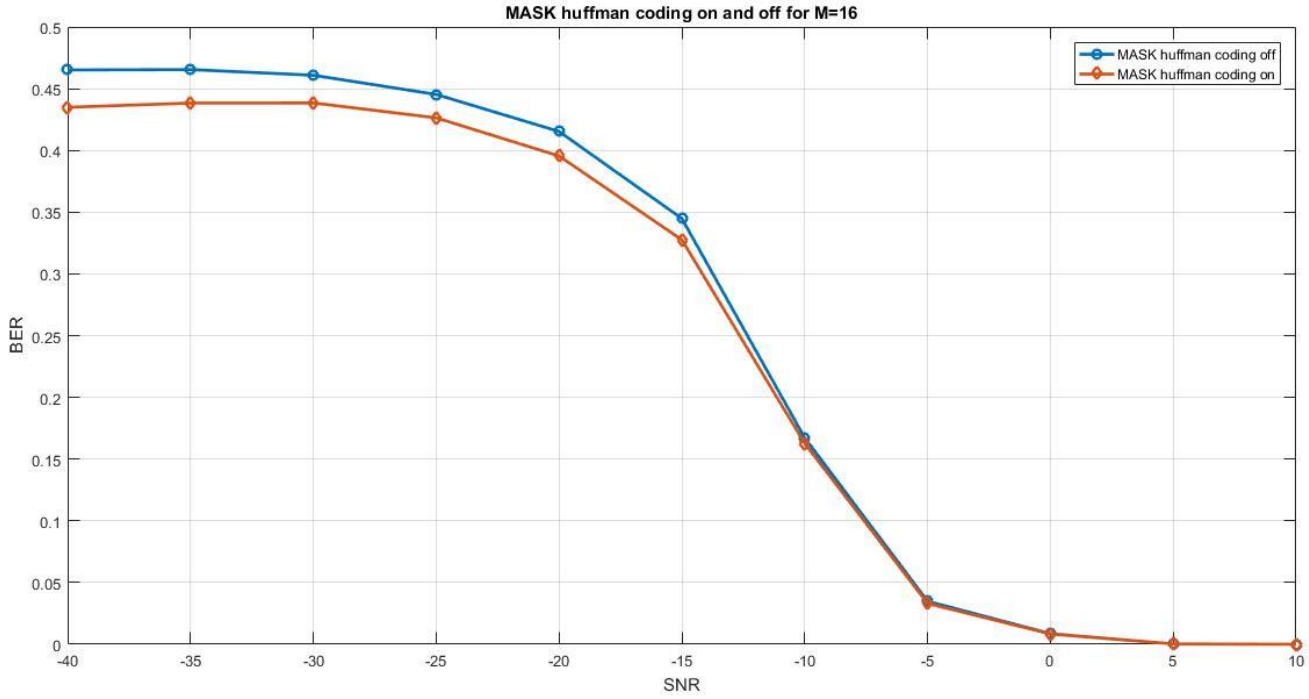


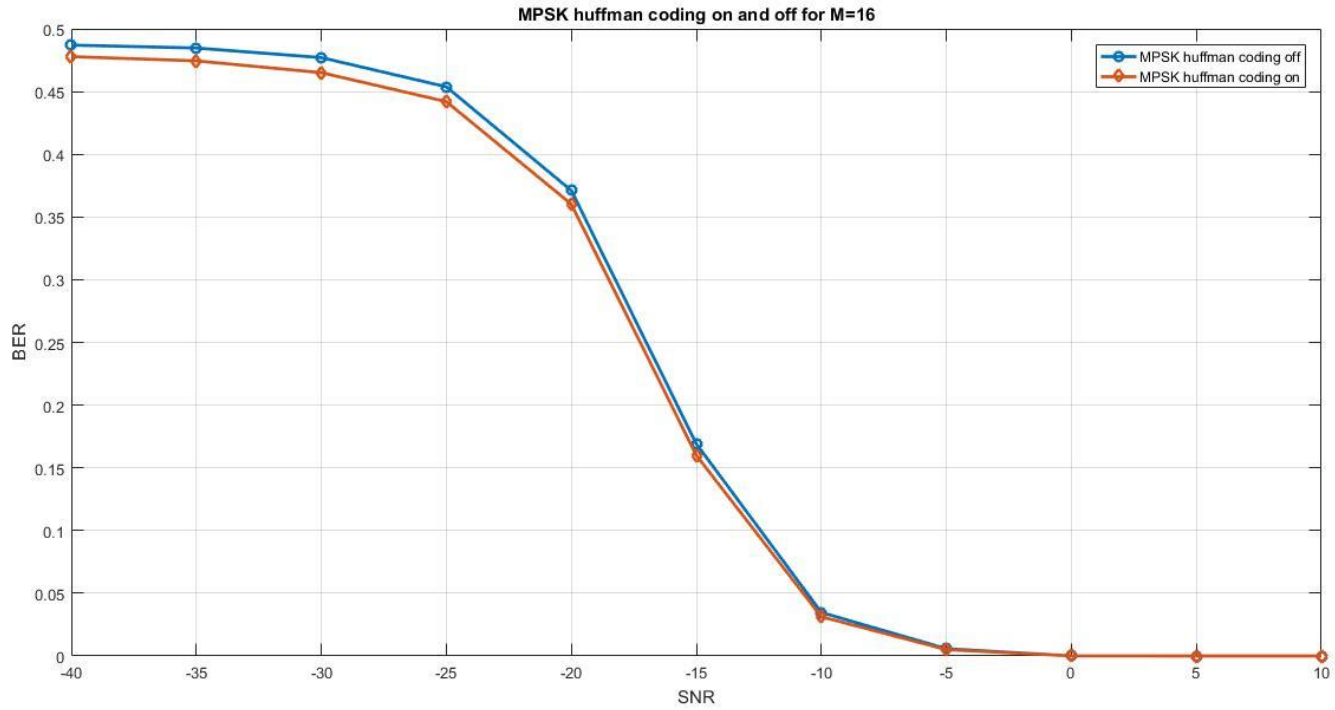
Discussion:

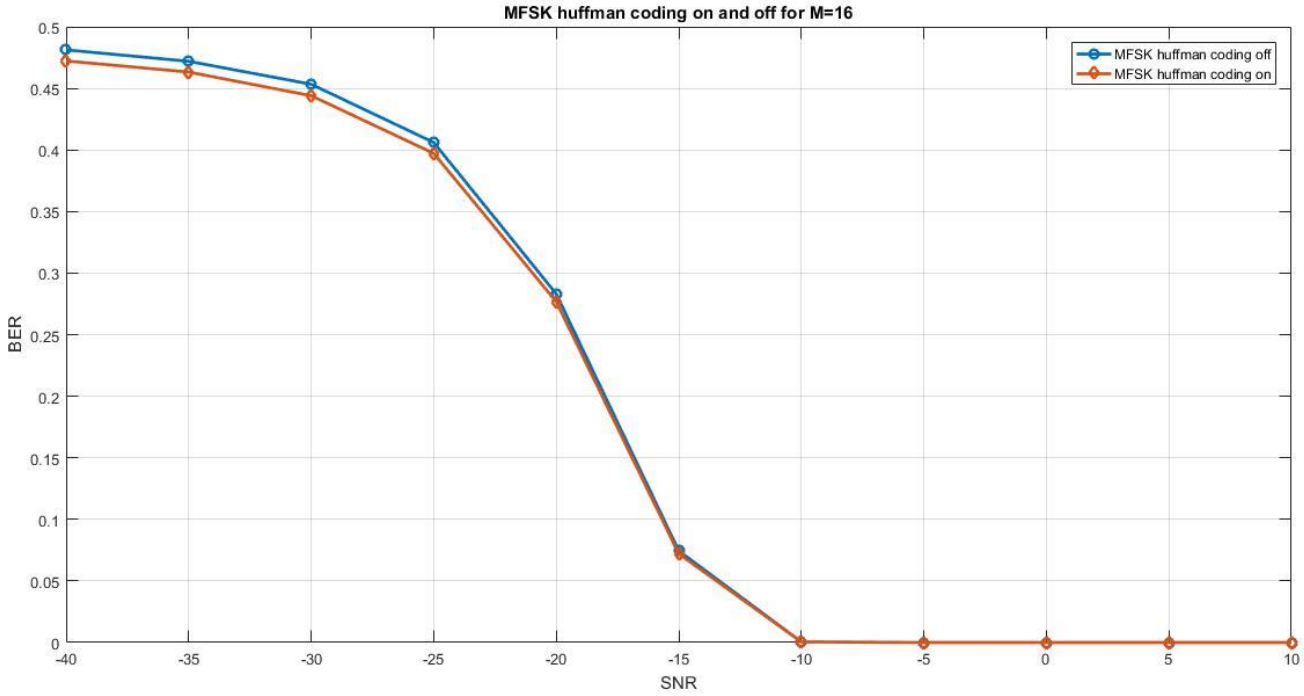
1. We can see that GMSK always have the worst performance because at high noise phase is corrupted by a great number. MPSK and MFSK performs better than MASK. This result is expected. Noise effects the amplitude more than the phase. In MASK, information is stored in the amplitude whereas in MFSK and MPSK information is stored in the phase. So they perform better.
2. Interstingly, if for low M, MPSK peforms better than MFSK. But as M increases, MPSK's performance start to degrade and falls below MFSK. The reason behind that is as we increase M, the interval between two consecutive phases decreases. Hence the probability of getting an error becomes larger. But if we increase, M different frequencies of MFSK still remains orthogonal to one another.

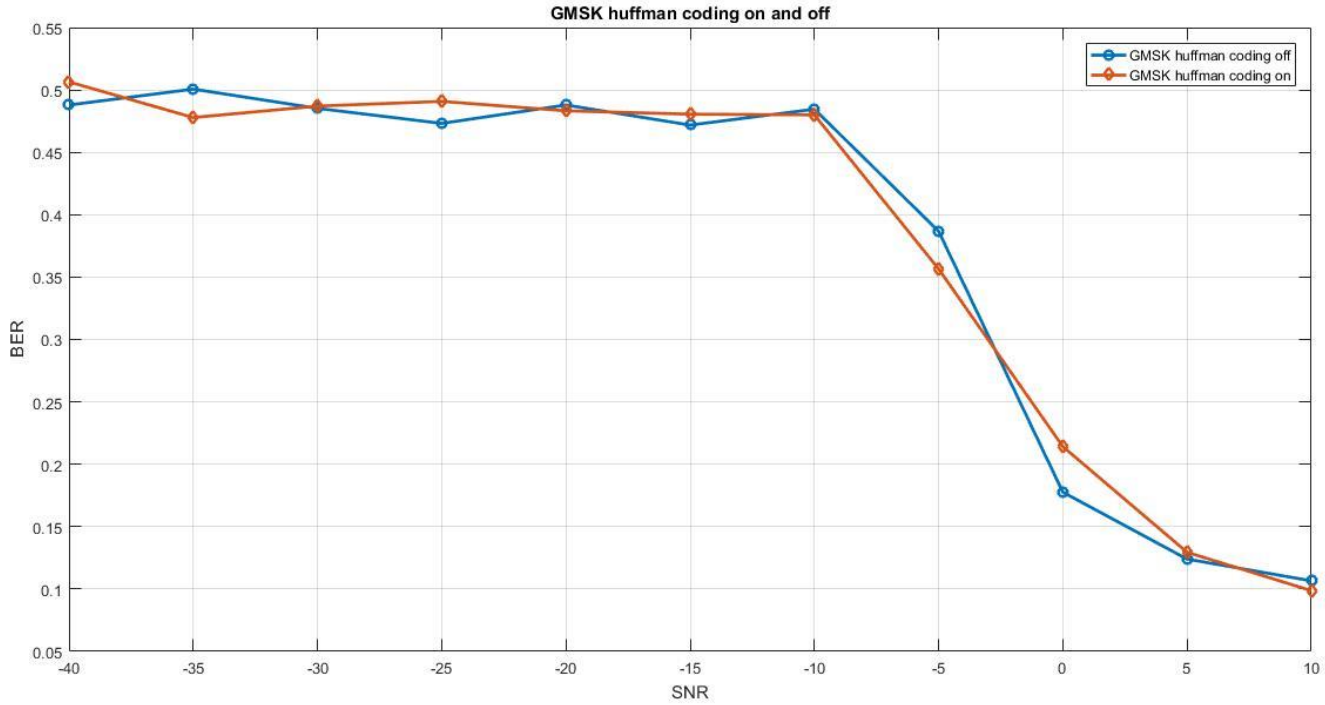
4. Huffman Coding on and off

Effect of turning of the Huffman coding scheme is given here.



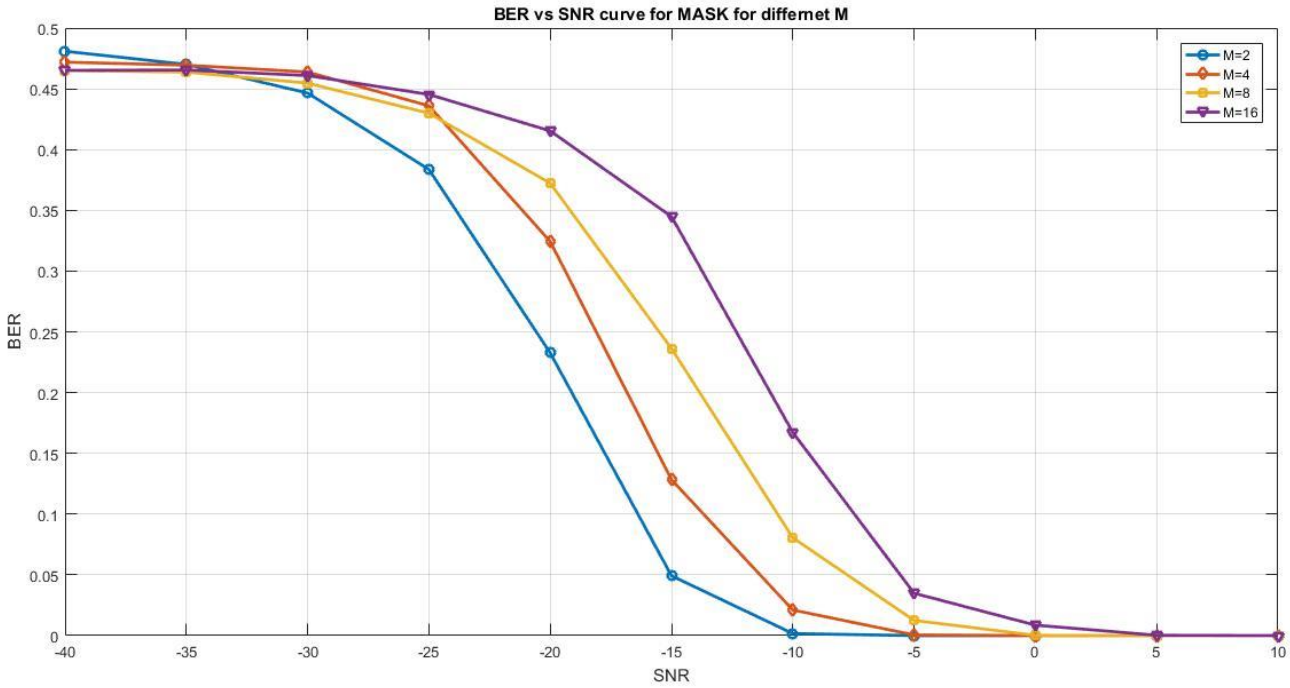


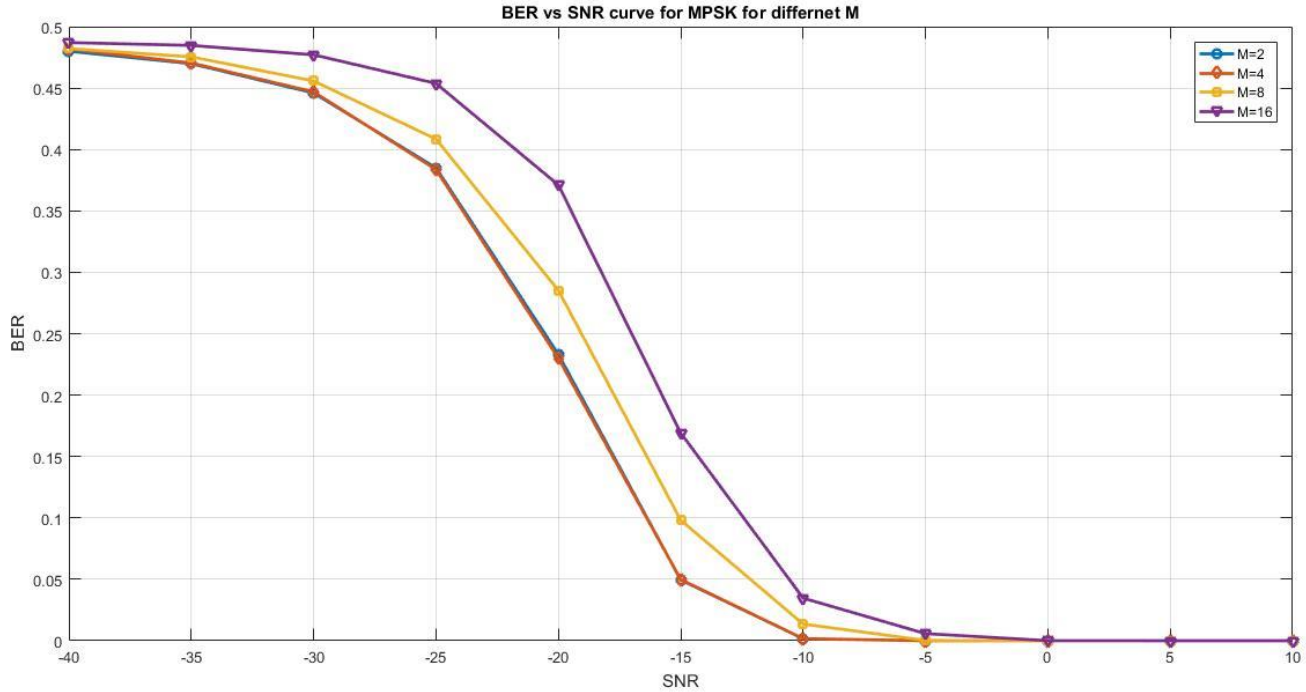


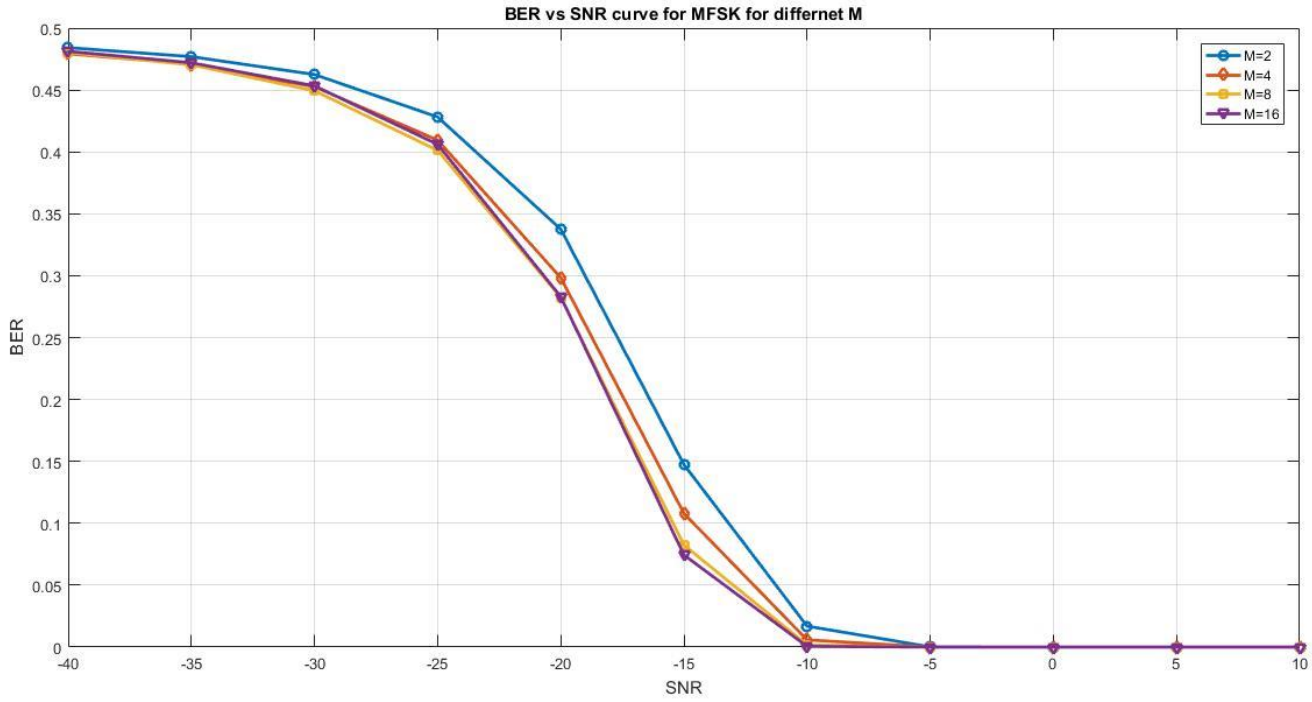


Discussion: Huffman encoding does not effect bit error rate at a significant amount.

5. Effect of varying M for different line Coding Scheme







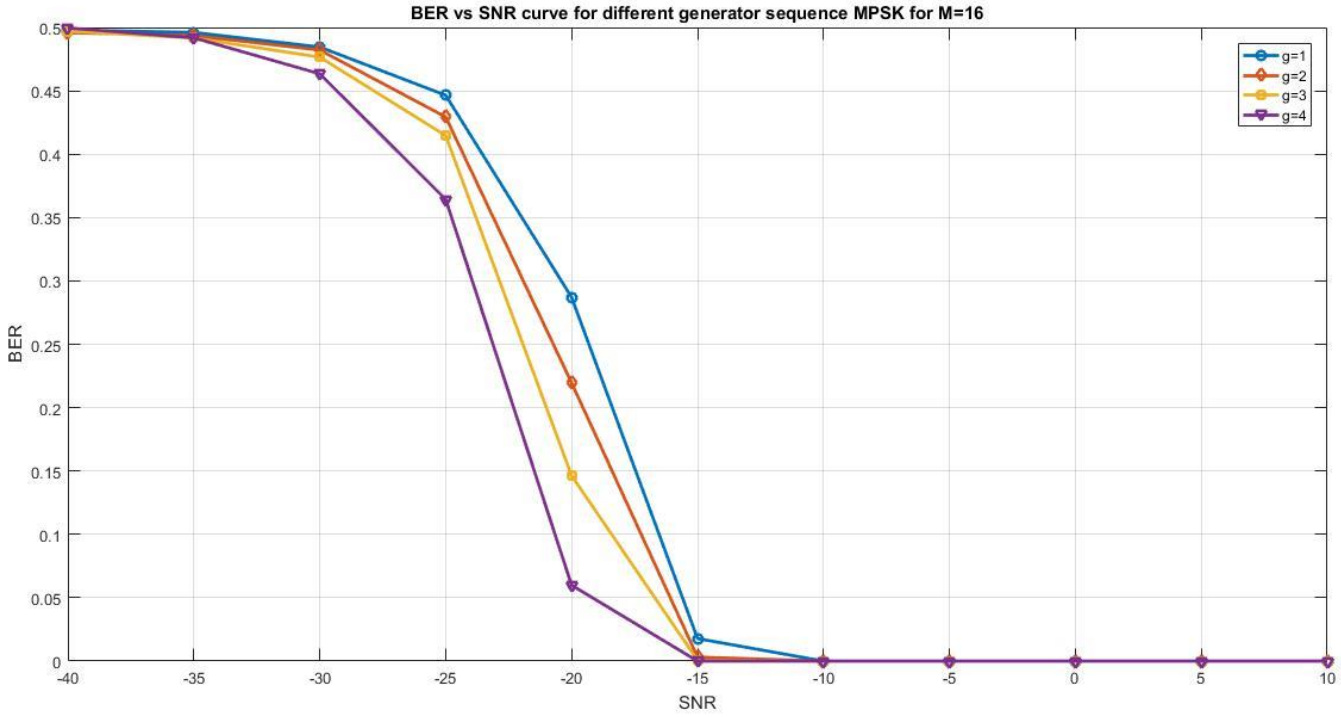
Discussion: In every scheme, increasing M increases bit error rate.MASK is affected mostly because of it. But the changes in MFSK is not that much significant. If we increase M, the interval between two consecutive signal vector decreases. Thus a little amount of noise can cause greater error. Hence the performance degrades.

6. Effect of Generating sequence

Effect of turning of the Huffman coding scheme is given below.

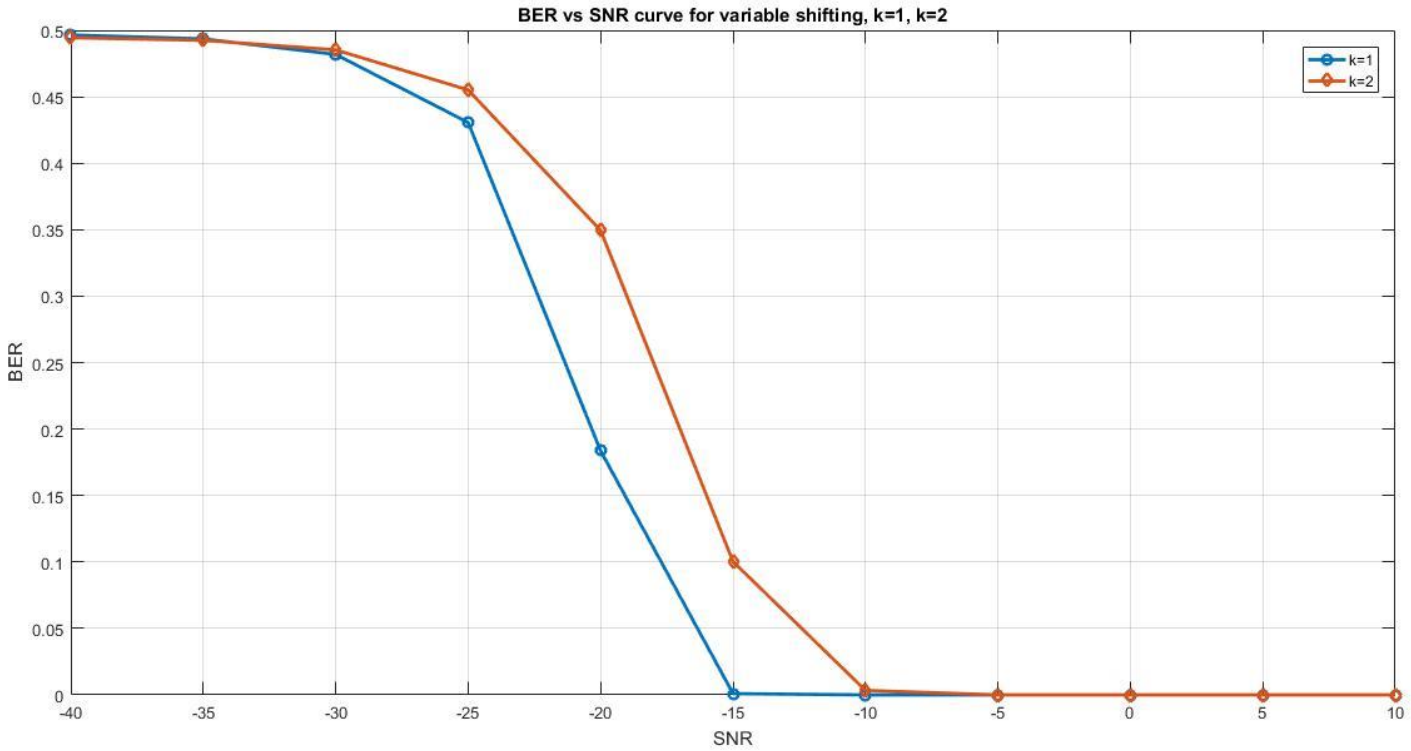
Effect of varying number of outputs:

For generating sequence g1=[1 0 1; 1 1 1], g2=[1 0 1; 1 1 1;1 1 1], g3=[1 0 1; 1 1 1;1 1 1; 1 1 1], g4=[1 1 1; 1 1 1;1 1 1; 1 0 1; 1 0 1] and L=4, n=3:



Discussion: The graph shows that increasing the number of outputs(n) decreases the bit error rate. If we increase n for fixed k, we are representing same number of bits with more outputs. That means we are adding more information. So BER decreases.

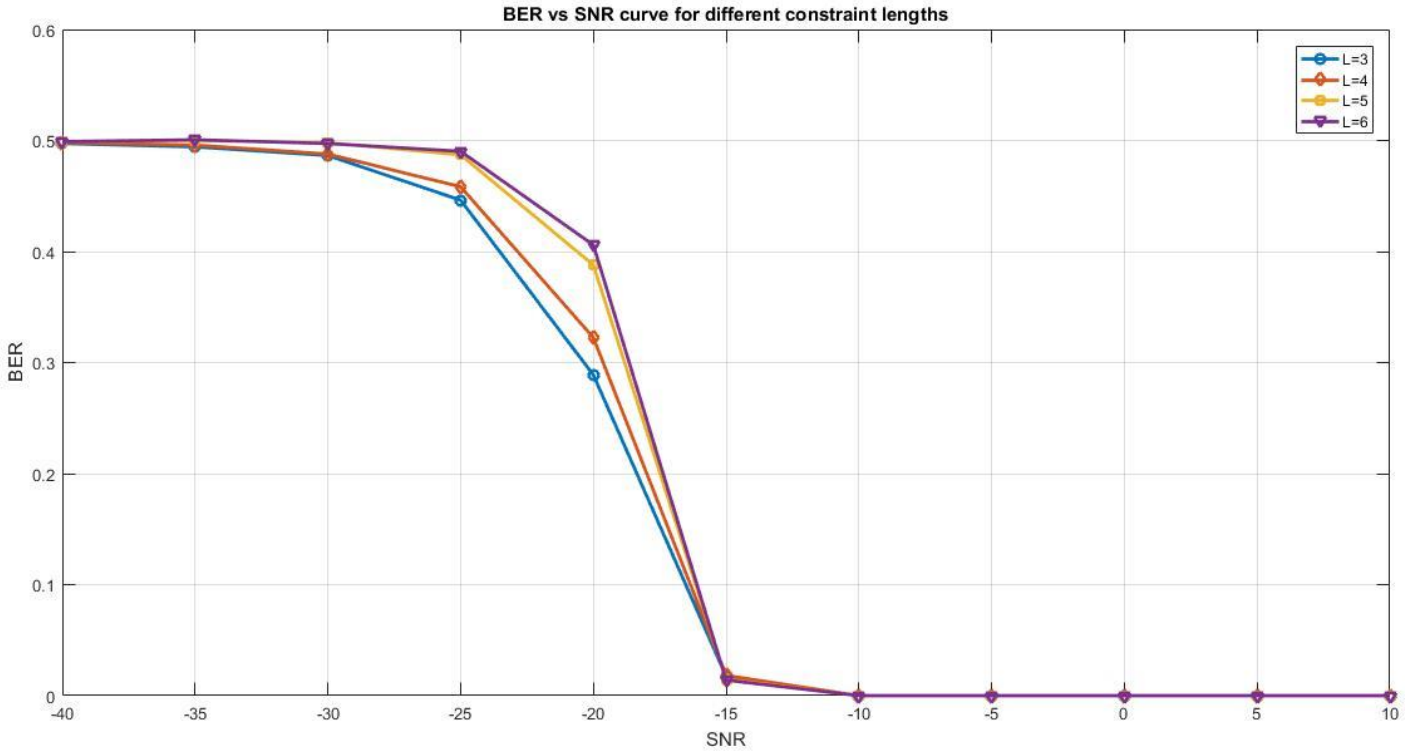
Effect of varying number of bit entered into the register (k):
For L=4,n=3 and G[1 0 1 1; 1 1 0 1; 1 1 1 1] and k=1,2:



Discussion: If we increase k, error performance degrades. Incrasing k for same number of output bits means that we are representating more number of bits with a fixed output bits. So some information is lost and hence BER increases.

Effect of varying constraint length(L):

For k=1,n=2 and g1=[1 0 1; 1 1 1] for length(M) = ; g2=[1 1 0 1; 1 1 1 1]; g3=[1 0 0 1 1; 1 1 1 0 1]; g4=[1 0 1 0 1 1; 1 1 1 1 0 1]



Discussion: It does not have any significant amount of change in bit error rate because the number of input and output bits do not change. So information remains the same.

4 CONCLUSION

These code snippets are form the jupyter notebook with octave kernel installed into it. And the final program is tested in MATLAB. The large level of optimization results into shorter cpu time. But still the GMSK can not be implemented with 4GB RAM processor. I Think that using external memory to solve the RAM problem by saving and using data from Hard-Drive. But the saving and loading of this huge file would cost a large amount of time. On the other hand the symbol error rate can be checked using longest common subsequence , but $O(n^2)$ time algorithm is not suitable for this large bit stream. Another solution for this problem is to use data sequence matcher which is efficiently implemented in Python. I didn't get much time to investigate in this regard, so I could not attach the symbol rate error with this report. But this assignment is very challenging and introduce us with different concepts which we are unware of. Comparing methods and other optimization techniques gives us the view of real engineering.