Coding Theory Technical Report

Student Name: Zhe Li

Student Number: 1952468

Date: 5/2/2021

1. Introduction

This report aims to recording the process and discuss the simulation results during exploration of Coding Theory's coursework. The core of this report is investigation of performance of convolutional code and LDPC code in AWGN channel and situation where burst noise occurs by using matlab. The performance of two types of codes are measured by BER versus Eb/N0 and comparison of simulation curve between theoretical curve. This report is only for practical study and there are many omissions and inaccuracies.

2. Methods, Results and Discussion

2.1 Empirical investigation of the error correcting performance of a binary convolutional code.

Task 1.1

2.1.1 Simulation process design and key points explain

In this task 1.1, I write a matlab script to build an implement of convolution code encoder and decoder with the required specifications (code length is 406, data sequence length is 200, code rate is 1/2.). In this task 1.2, I added the inter-leaver to the communication systems and add burst noise in the channel.

In task 1.1, the simulation is implemented in such process: (1) Simulate the AWGN channel. (2) Design encoder: analysis encoder's structure, get the schematic diagram of decoder, get the finite state diagram. (3) Design the BPSK modulator and demodulator. (3) Design the decoder using Viterbi decoding algorithm. (4) Implement the Monte Carlo simulation. (5) Analysis and discussion the results from the BER graph. The structure of simulation communication is shown below.

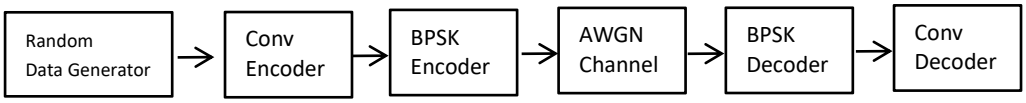


Figure 1. Communication Simulation System's Structure

The Details and key methods in above sections would be explained next.

Random data Generator:

Use randi() built-in function to generate 200 bits binary data sequence.

Simulation of AWGN:

The key methods is to use the *normrnd()* matlab built-in function and adjust the parameter related to variance of AWGN to $\sigma^2 = \frac{N_0}{2E_b}$.

Encoder Design:

The generating polynomials of encoder G=(1+x+x^2+x^3,1+x^3). So the generating Matrix $G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$.

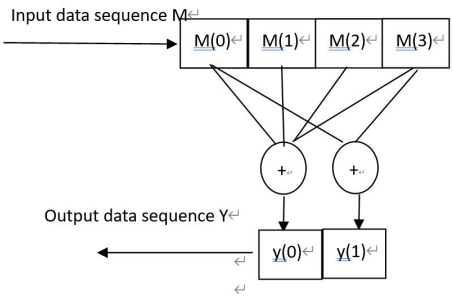


Figure 2. Encoder schematic digram

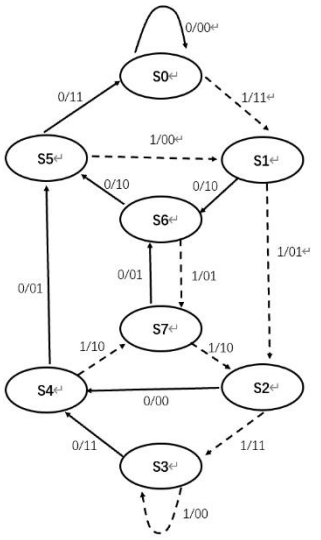


Figure 3. The Finite State Diagram encoder

BPSK Modulation and Demodulation:

Create a 1*2 size matrix storing the modulated value, -1/ (2^0.5) and +1/ (2^0.5). Let the (input data value+1) to be the index of matrix. i.e. The data value 1 would be mapped to +1/ (2^0.5), 0 would be mapped into -1/ (2^0.5). Here I scale the value from +1,-1 to +1/ (2^0.5) and -1/ (2^0.5) because under the transmitting power equal to unity. The bit numbers of modulation double, so the amplitude of BPSK signal should be reduce to 1/(2^0.5). (P=(I^2)*R).

Viterbi Decoding:

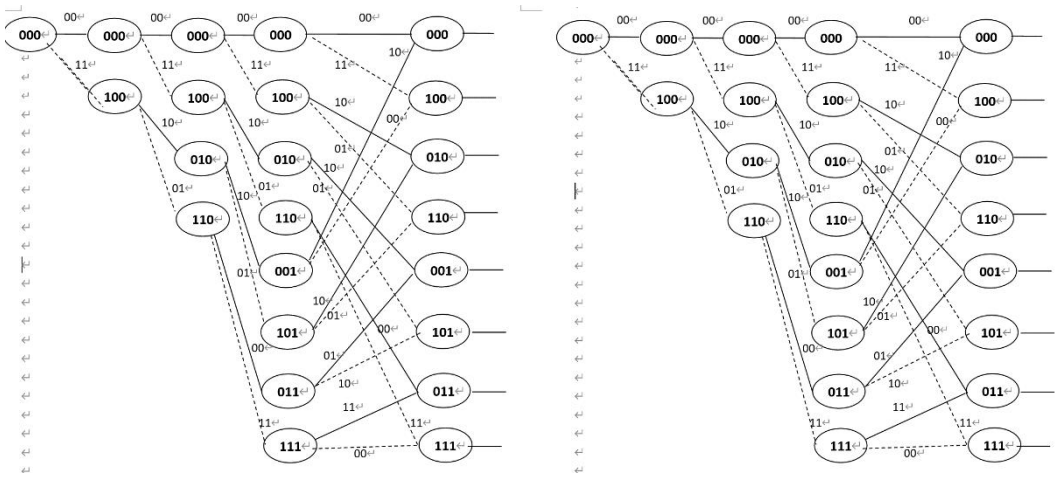


Figure 4. Trails diagram

1.The Structure of code:
(1) Initialization: Create the variable to store cumulative metric and paths, some tables which were used in process of finding the shortest path.
(2) Cycle Structure: A For Loop repeating 204 times. One repeat is used to find the shortest path for the next step in trails diagram.
(3) Translation: Find the shortest path from the eight path and then translate the shortest path into the 203-length data.

2. Description of variable

Variable Name	**cumulative_metric**	**temporary_metric**	**Input_table**
Size	1 * 8	1*8	8*8
Function	Store the cumulative metric of 8 state on the column before current column on trails diagram	Store the temporary metric of 8 state. It is used in the every cycle as a intermediate variable .	Value in table: 0 and 1-- input value needed to make current state i (column order number, in decimal) transfer to next state j (row order column, in decimal) -1 – means current state i(column order number) can not transfer to next state i (row order number).
Variable Name	**path**	**Path_temporary**	**output_table**
Size	8 * 240	8*240	2*8
Function	Row order number means the current state in decimal and column order number means this node is in which step in trails diagram. The value in (2, 3) is stage value in decimal passed at the step 2, and the destination of this path is 011.	Same as the path, the only difference is this matrix store the temporary changed path when program run into cycle structure.	The order number of columns means decimal state number on registers. The order number of rows means the input number. The value at the 2nd column and 2nd row means output value when input is 1 and state is 010.

Monte Carlo Simulation:

In order to make the practical BER close to the theoretical BER, the times of repetition I set is 200*(1/theoretical BER) time. The program has more simulations at the higher SNR range.

2.1.2 Simulation Result Analysis

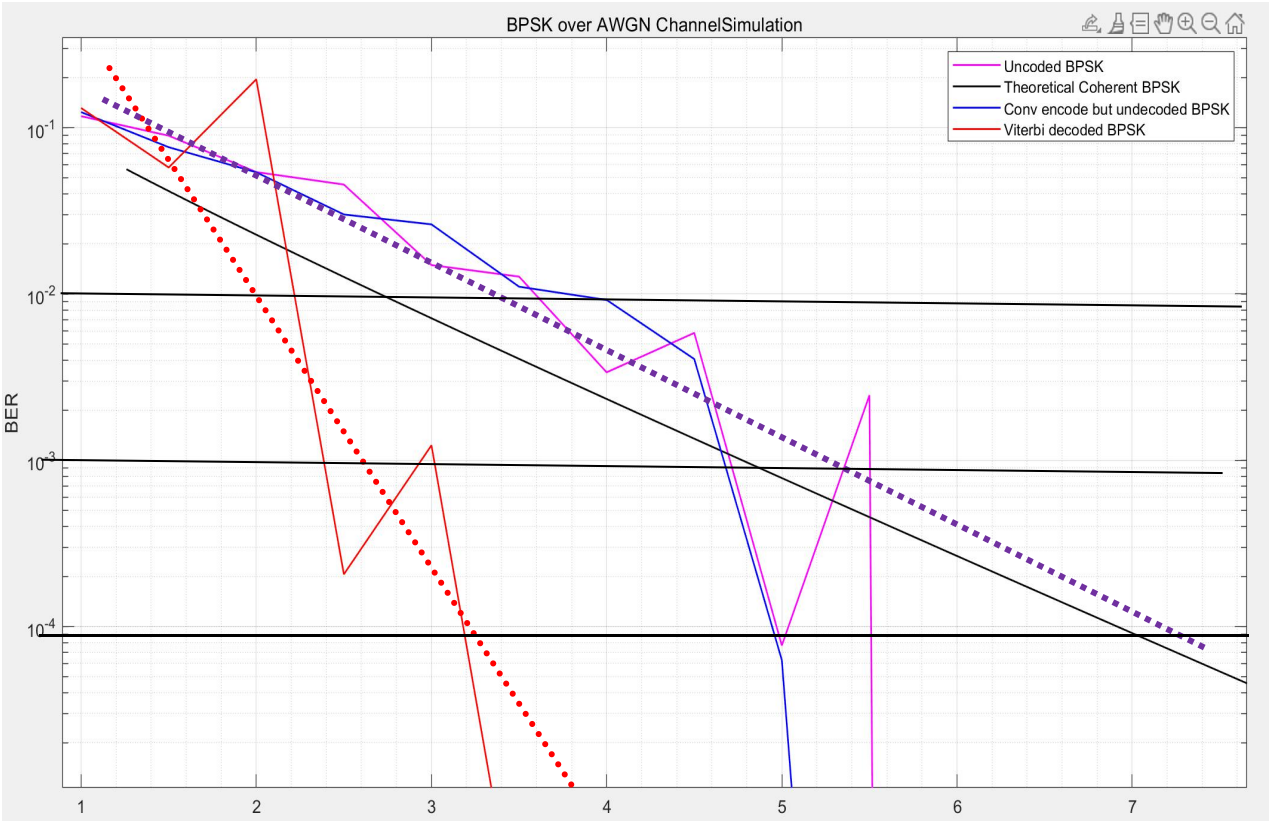


Figure 5. Simulation result for uncoded BPSK modulation, uncoded BPSK transmission, and my code. (Value interval on Eb/N0 is 0.5 dB)

From the Figure, it can be seen that there are some large shakes on red line and purple line. The reason for shakes is that the number of repetition is not enough. For time reason, I didn't continue to optimize it. The red dot line and purple dot line is the guide line I draw, which means they are not calculated curve. The dot lines are estimation of practical curve, trying to ignore the influence of MC Simulation.
The code gain at BER=0.01 is 1.5 dB. At BER=0.001, it is 2.8dB. At BER=0.0001, it is 4 dB. Besides, the figure shows that the lower BER is, the larger code gain is.

Task 1.2

Interleaver can permute coded symbols according to certain rules. The process of permutation is called interleaving coding. If there are continuous errors in the transmission, they must be concentrated in a segment of the interleaved code sequence. Because the symbol sequence of the interleaved code is permuted, the concentrated error code can be dispersed and replaced back into the original code sequence by the reverse permute of the receiver. These errors can be corrected by using error correcting coding technology.

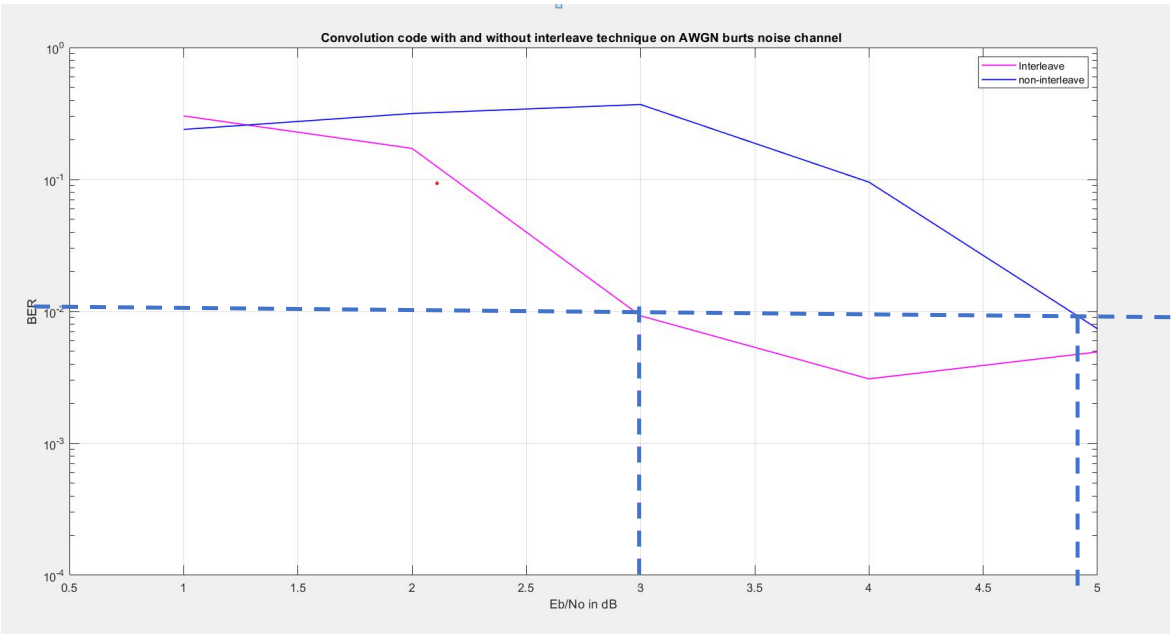


Figure 6. Convolution code with and without interleave technique on AGWN burst noise channel. (noise burst in 101-110 bit on codework, value interval on Eb/N0 is 1dB)

The figure shows that from 1 to 2 dB, the difference between the code applied interleave technique and the one not is not significant. But from 2 dB To 4 dB, the difference begin to enlarge and maintain 2dB improve from BER=1*e^-2 to BER=1*e^-1. When Eb/N0 exceed the 4dB, the effect of interleave decrease.Here I did not simulation the condition when Eb/N0 increases from 5dB to 10dB. After I simulated this case and next time when I tried to simulation case from 1dB to 10dB, the figure is very mess and strange. I reasoning that I must should change some places in my code after last simulation. But for time reason, I gave up fixing it. I attached it below.

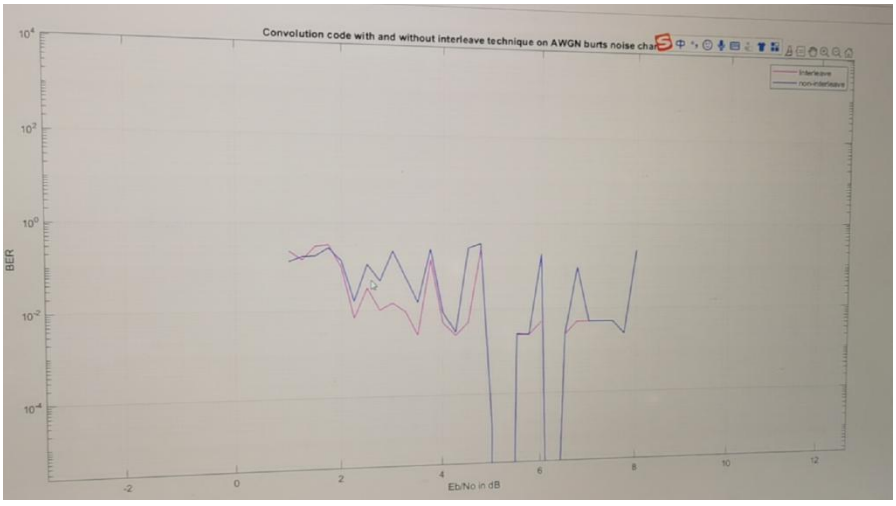


Figure7. Convolution code with and without interleave technique on AGWN burst noise channel from 1-10dB. (time interval is 0.5dB)

2.2 LDPC codes

2.2.1 Simulation design and key point explain

Task 1

In task 1, I didn't generate G matrix. I try to build a Non-structured matrices H mentioned in Lecture about 5G LDPC code in practice.

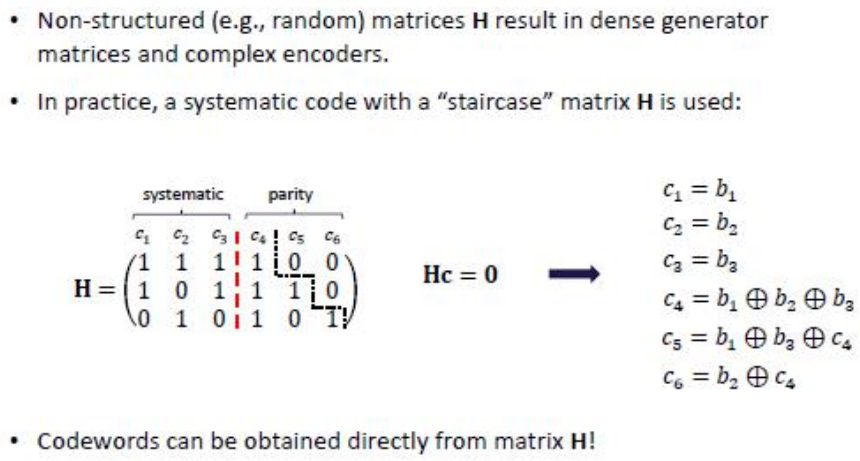


Figure 7. Illustration in *5G LDPC in practice UoB Lecture – BluWireless.pdf*

The design of generate metric H:

1. Generate a full rank matrix B, the size of B is 200*200 and and B is a Double diagonal matrix. (1 only appear at the double diagonal position). The reason for choose double diagonal structure is to guarantee that in each parity formula (every row of H) , there are only two parity bits involved. It can reduce the complexity of calculation.
2. Generate a square matrix A1. The number of 0 is about 90% of the total number of elements in A1. The position of 1 is totally random.
3. Build Parity matrix H. H = [A1, B].

Task 2

There are a lot of multiplications in the variable node and check node operation of LDPC code's probability domain BP algorithm, so the calculation amount and complexity are very high. If the above probability information of 0 and 1 is expressed by log likelihood ratio, the multiplication operation can be converted into addition operation, which greatly reduces the calculation amount. This algorithm is called sum product decoding algorithm.

BP algorithm I used is sum-product algorithm.

1.Initialisation:

$q_n^{[0]}[0]=LLRn=In\frac{\mathrm{p(y|bn=0)p}}{\mathrm{p(y|bn=1)}}$), where $yn$is the received modulated symbol and $bn$is the encoded bit.

$$r_{mn}^{[0]} = 0$$

2.Check node update at iteration i>0(after messages from bit nodes received):

$$r_{mn}^{[i]} = 2tanh^{-1}[\prod_{k:H_{mn}=1,k\neq n} \tanh\left(\frac{q_k^{i-1} - r_{mk}^{i-1}}{2}\right)]$$

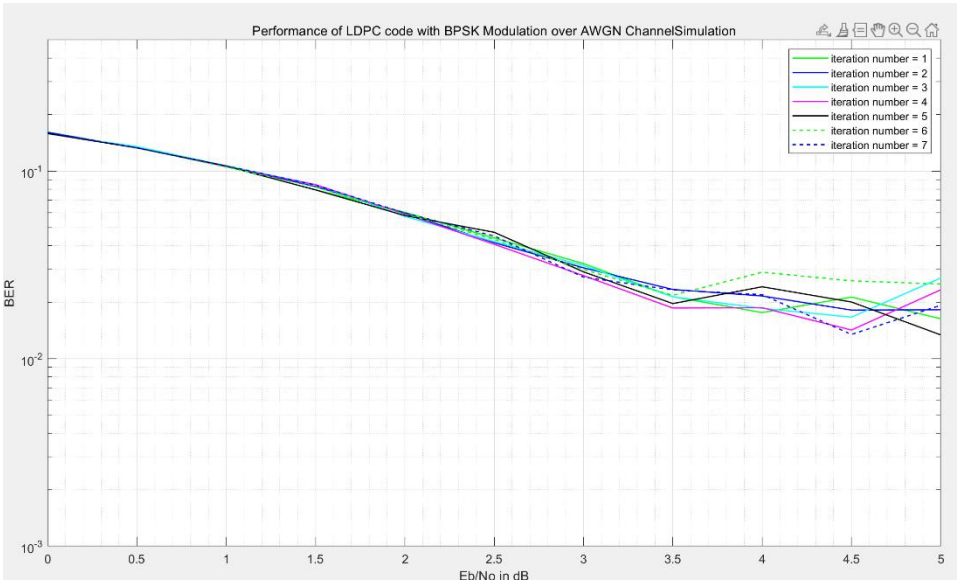
3. Bit node update at iteration i>0 (after messages form check nodes received):

$$q_n^{[i]} = LLR_n + \sum_{m:Hmn=1,k\neq n} r_{mn}^{[i]}$$

Code structure:

1. Initialize the program: create the variable matrix to represent.
2. A loop structure (repeat time equal to iteration time in BP) to update the check and bit node again and agin.

2.2.2 Simulation result Analysis



Figrue 7.BER curves at different iterations after simulation

From the figure, the total trend of BER curves at different iterations is identical. With the increase of Eb/N0, the BER decrease. Here the practical curves is not identical to the theorical result. In theory, when iteration time increase from 0 to 7, the performance of BER curves should be improved gradually and the improvement dB should be more and more implicit.

And the performance of BER curves is bad, at 5dB, the BER is closed to 0.01. This is far from the ideal curves. The reason for this, I think, is the structure of H. In my designed H, the A1 part is totally random generated. So the check node degree and variable degree is not fixed. Weight of each row and column is different and uncertain. All this lead to the poor performance of my LDPC code.

Because of the sparsity of the parity check matrix, the information bits far away from each other participate in the unified parity check in the long coding packet, which

makes the continuous burst error have little effect on the decoding. The coding itself has the characteristics of anti-burst error. Its randomness does not require the introduction of interleaver as turbo code, so there is no delay due to the existence of appliances.

Besides, the countermeasures against noise “burst” with my LDPC code is change binary LDPC code into M-ary LDPC code. In the Tanner graph of LDPC, the variable nodes of binary LDPC codes represent 1 bit, while the M-ary LDPC codes represent multiple bits, in another word, a symbol. A symbol is composed of multiple bits. In the iterative decoding process, the wrong codeword can be corrected to the correct codeword. The wrong codeword may be caused by one bit error or multiple bit errors. That is to say, when a symbol is wrong, any bit in the symbol may be wrong. In the iterative decoding process, correcting one symbol is equivalent to correcting several bits at the same time. For burst noise, if 8 consecutive bits receive influence of burst noise, only 2 or 3 symbols will be affected. For the hexadecimal decoder, these interfered symbols can be corrected iteratively as independent nodes.

Because the performance of my designed code is not good. So I didn’t compare this code with convolution code. The result is obvious. I'm very sorry about that.

Reference

1. Tao Wei, Zhang Wei, Liu Xingan, Yang Xuefei, Liu Bing, “Performance of Nonbinary LDPC CodeS over Noise Burst Channels”, Journal of Beijing University of Posts of Telecommunications.Vol,34.Sep.