

# Lab 6: Low Density Parity Check Code

Author	Name: 宋宜航 Student ID:12112717	张皓东 12113010
<p><b>Introduction</b></p> <p>In a digital communication system, the efficiency and reliability of information transmission stand as two crucial metrics for evaluating system performance. In the realm of communication systems, random bit errors are an unavoidable occurrence, attributable to factors such as channel noise and interference. Consequently, channel coding techniques prove instrumental in significantly enhancing the reliability of the transmission system.</p> <p>The primary objective of channel coding is to introduce redundancy into transmitted data, endowing the system with the capability to detect and correct errors. By adding redundant information at the transmitting end, the receiving end becomes equipped to detect and rectify errors that may occur partially or entirely during transmission. This fortifies the entire communication system against the challenges posed by noise and interference. Error control techniques, including channel coding, find widespread application in digital communication, particularly in scenarios such as wireless communication, satellite communication, and internet data transmission, ensuring data integrity and reliability.</p> <p>The choice of channel coding technique is typically contingent on the specific requirements and performance demands of the communication system. Common channel coding schemes encompass Hamming codes, convolutional codes, and LDPC (Low-Density Parity-Check codes), among others. Through the introduction of appropriately designed redundant bits, these coding techniques effectively bolster the system's resistance to errors, ensuring the reliability and stability of information transmission. By thoughtfully considering the selection and configuration of channel coding during the design phase, digital communication systems can better adapt to complex communication environments, elevating the success rate and quality of data transmission.</p> <p><b>Generator Matrix</b></p> <p>In channel coding, the generator matrix is a crucial concept, particularly in the context of Linear Block Codes. The generator matrix defines how the linear block code generates codewords through the multiplication operation with the matrix. Linear block codes add redundancy to the message bits to provide error detection and correction capabilities.</p> <p>The generator matrix is typically used to describe the system of a linear block code, and it is a matrix composed of message and codeword bits. The construction of the generator matrix depends on the specific linear block code scheme.</p> <p>Consider a <math>(n, k)</math> linear block code, where <math>n</math> represents the number of coded bits, and <math>k</math> is the</p>		

number of message bits. The generator matrix, denoted as  $G$ , is a  $k \times n$  matrix, and it specifies how to transform  $k$  message bits into  $n$  codeword bits. The encoding process involves representing the message bits as a row vector, denoted as  $m$ , and obtaining the codeword bits, denoted as  $c$ , through matrix multiplication:  $C = m \times G$ . The typical form of the generator matrix is given by:

$$G = [ I_k \mid P ]$$

Here,  $I_k$  is the  $k \times k$  identity matrix, and  $P$  is a  $k \times (n-k)$  matrix representing the redundant part of the encoding. In this case, the codeword  $c$  can be obtained as:

$$C = m \times G = [ m \mid m \times P ]$$

The notation  $[ m \mid m \times P ]$  indicate the concatenation of  $m$  and the result of multiplying  $m$  by matrix  $P$  column-wise.

The selection of the generator matrix is crucial for the performance of the encoding. It needs to be designed to offer robust error detection and correction capabilities and is often optimized based on specific error models and communication channel conditions.

### Parity-check Matrix

In channel coding, the Parity-check Matrix is another critical concept, particularly within the context of Linear Block Codes. The Parity-check Matrix describes how errors in the encoded bits are detected, constituting a fundamental component alongside the Generator Matrix in linear block coding.

Consider a  $(n, k)$  linear block code, where  $n$  represents the number of coded bits, and  $k$  is the number of message bits. The Parity-check Matrix, denoted as  $H$ , is an  $(n-k) \times n$  matrix, defining how errors are detected and located within the encoded bits. The construction of the Parity-check Matrix depends on the specific linear block code scheme.

The Parity-check Matrix is typically represented in the form of a systematic matrix:

$$H = [ -P^T \mid I_{n-k} ]$$

Here,  $P^T$  is the transpose of the redundant matrix part in the Generator Matrix  $G$ , and  $I_{n-k}$  is the  $(n-k) \times (n-k)$  identity matrix.

By performing matrix multiplication of the received encoded bits with the Parity-check Matrix  $H$ , the resulting bits are referred to as "parity check bits." If certain aspects of the parity check bits do not satisfy the predetermined parity conditions, the system can detect the presence of errors. Analyzing the pattern of the parity check bits allows for error localization, and in some cases, error correction.

The Generator Matrix specifies how to generate encoded bits, while the Parity-check Matrix dictates how errors are detected and localized. The selection and optimization of these matrices are crucial in the design and analysis of channel coding schemes. Their relationship can be represented as follows:

$$\begin{cases} \mathbf{G} = [\mathbf{I}_k & \mathbf{Q}_{k \times r}] \\ \mathbf{H} = [(\mathbf{Q}_{k \times r})^T & \mathbf{I}_r] \end{cases}$$

## Syndrome Vector

In channel coding, the Syndrome Vector is a concept crucial for detecting and correcting errors in the encoded data. Particularly in the context of Linear Block Codes, the Syndrome Vector plays a pivotal role.

In linear block coding, a Parity-check Matrix ( $\mathbf{H}$ ) is commonly used for error detection. The receiver multiplies the received codeword by the transpose of the Parity-check Matrix, resulting in what is referred to as the Syndrome Vector.

Consider a  $(n, k)$  linear block code, where  $n$  represents the number of coded bits, and  $k$  is the number of message bits. The Parity-check Matrix ( $\mathbf{H}$ ) is an  $(n-k) \times n$  matrix that defines how errors are detected and located in the encoded data. The Syndrome Vector is obtained by performing matrix multiplication of the received codeword with the transpose of the Parity-check Matrix.

If the codeword is error-free, the Syndrome Vector will be zero. However, if errors are present, the Syndrome Vector will provide information about the nature of these errors. Analyzing the pattern of the Syndrome Vector allows the receiver to detect the presence of errors and, in some cases, correct them.

The application of the Syndrome Vector is particularly relevant in error correction codes such as Hamming codes and LDPC codes. By detecting a non-zero Syndrome Vector, the receiver can indicate the presence of errors, enhancing the overall reliability of the encoding scheme. The whole structure is as below:

Parity-check matrix

Received bits

Syndrome vector

$$\mathbf{S}^T = \mathbf{H} \cdot \mathbf{R}^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_6 \\ r_5 \\ r_4 \\ r_3 \\ r_2 \\ r_1 \\ r_0 \end{bmatrix} = \begin{bmatrix} r_6 + r_5 + r_4 + r_2 \\ r_5 + r_4 + r_3 + r_1 \\ r_6 + r_5 + r_3 + r_0 \end{bmatrix} = \begin{bmatrix} s_2 \\ s_1 \\ s_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

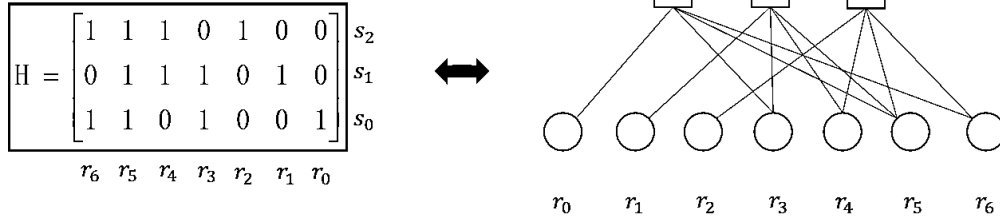
## Tanner Graph and The Loop of LDPC

The Tanner Graph provides an intuitive depiction of the internal structure of a parity-check matrix, aiding in the assessment of its performance. As illustrated in the diagram below, each row in the parity-check matrix  $\mathbf{H}$  represents a constraint equation, corresponding to check nodes in the Tanner Graph denoted by square nodes. Each column in the parity-check matrix  $\mathbf{H}$  represents a codeword, corresponding to variable nodes in the Tanner Graph denoted by circular nodes. A value of 1 in the parity-check matrix  $\mathbf{H}$  indicates a connection between the check nodes and variable nodes.

By establishing this correspondence, one can visualize the Tanner Graph associated with the parity-check matrix  $H$ . Using this graph, we can intuitively evaluate the performance of the parity-check matrix  $H$ . Generally, for a well-designed parity-check matrix, the distribution of connections should be as uniform as possible, minimizing the occurrence of undesirable structures such as 4-cycles and 6-cycles.

The structure of the Tanner Graph provides insights into the encoding scheme, facilitating the analysis of the quality and performance of the parity-check matrix. Through careful examination of the Tanner Graph, we gain a better understanding of the relationships between constraints and variables in the coding system. This insight enables improvements and optimizations in channel coding schemes, enhancing error correction performance and overall system reliability.

$$\begin{bmatrix} s_2 \\ s_1 \\ s_0 \end{bmatrix} = \begin{bmatrix} r_6 + r_5 + r_4 + r_2 \\ r_5 + r_4 + r_3 + r_1 \\ r_6 + r_5 + r_3 + r_0 \end{bmatrix}$$

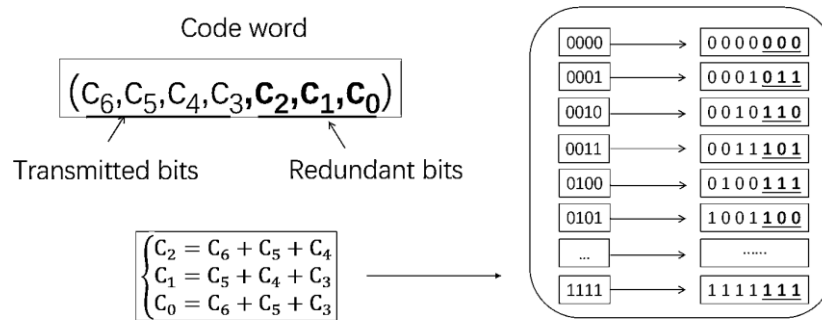


If there is a closed loop formed between nodes in a graph, it is referred to as a cycle or "girth." The presence of cycles has a significant impact on the performance of Low-Density Parity-Check (LDPC) codes. In graphical representation, the length of cycles is typically termed as "girth," and its existence influences the decoding performance of LDPC codes. Larger girth is often associated with better decoding performance as it helps reduce misleading paths during decoding, enhancing the accuracy of decoding. The terms "4-cycle" and "6-cycle" specifically denote structures where the minimum length of cycles is 4 and 6, respectively. The presence of these cycles can affect the performance of LDPC codes in the following ways: A 4-cycle refers to the minimum length of cycles being 4 in the LDPC graph. The existence of 4-cycles may lead to the emergence of erroneous cyclic paths during decoding, thereby reducing decoding performance. This is because, during the iterative decoding process, information may circulate among four nodes, resulting in incorrect decisions. Similarly, a 6-cycle refers to the minimum length of cycles being 6 in the LDPC graph. The presence of 6-cycles may lead to more complex misleading paths, further lowering decoding accuracy. During decoding, information may circulate among six nodes, increasing the difficulty of decoding.

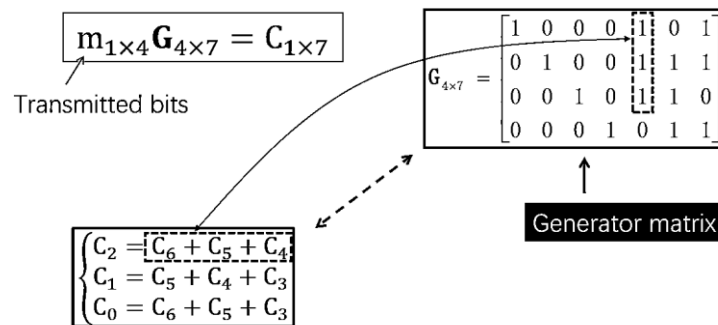
In general, smaller girth (minimum cycle length) is often associated with better decoding performance. Therefore, when designing LDPC codes, engineers typically strive to minimize the number of 4-cycles and 6-cycles or choose structures with larger girth to improve code performance. This may involve adopting more sophisticated LDPC code design algorithms or combining with other design principles to reduce undesirable cyclic structures.

### (7,4) Hamming Encoding

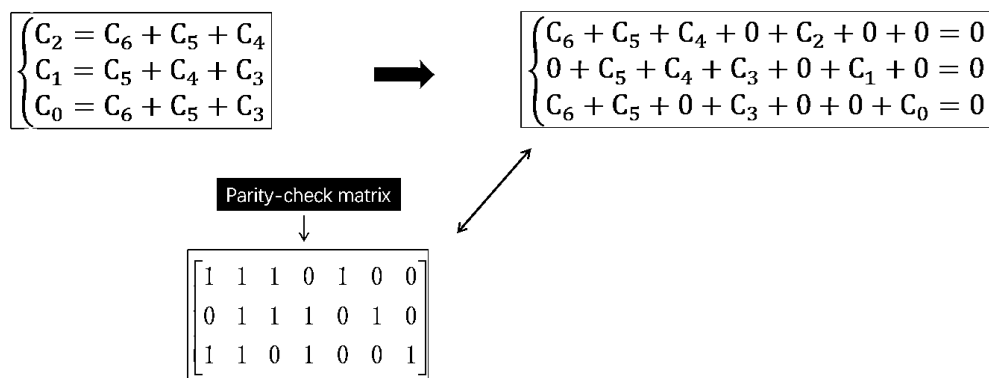
The 7,4 Hamming Code is a linear block coding scheme designed for single-error correction and double-error detection, aiming to enhance the reliability of data transmission by introducing redundant bits. In the 7,4 Hamming Code, each message block comprises 4 data bits and 3 parity bits, totaling 7 bits. There are 4 data bits used for the actual transmission of data. Three parity bits are incorporated to enable single-error correction and double-error detection. The positions of these three parity bits are determined based on powers of 2 and are typically denoted as P1, P2, and P4, corresponding to positions 1, 2, and 4.



The generator matrix for the 7,4 Hamming Code can be represented as:



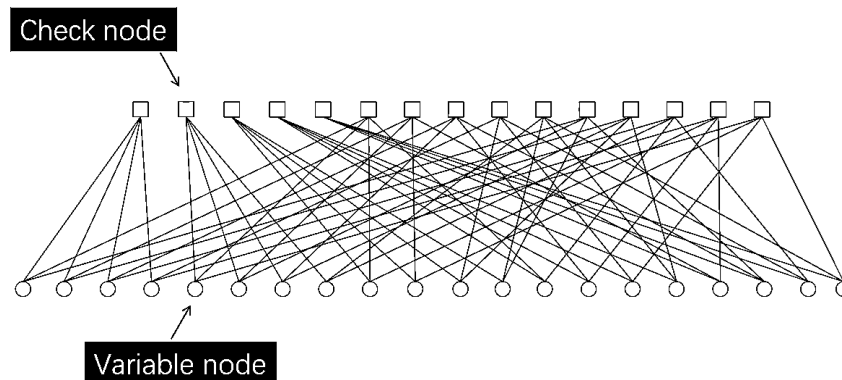
The generator matrix is used to encode 4-bit messages into 7-bit codewords. The parity-check matrix for the 7,4 Hamming Code can be represented as:



By employing the 7,4 Hamming Code, it is possible to detect and correct single-bit errors during transmission and detect double errors. This encoding scheme finds widespread application in communication and storage systems, particularly in scenarios where data reliability is of paramount importance.

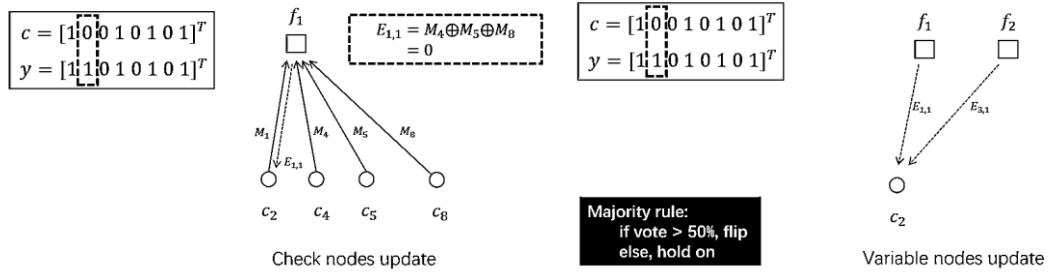
## LDPC Encoding

Low-Density Parity-Check (LDPC) coding is a widely used linear block coding scheme in channel coding, renowned for its outstanding error correction performance, especially in communication and storage systems. LDPC coding employs a sparse parity-check matrix, where the majority of elements are zero. This sparsity property enhances the efficiency of LDPC coding, particularly in hardware implementations. The parity-check matrix of LDPC coding can be represented graphically, known as a Tanner graph. In an LDPC graph, the connections between check nodes and variable nodes are represented by edges, and the nodes have relatively low degrees (number of connections).



LDPC coding utilizes an iterative decoding process called the message passing algorithm. This algorithm alternates between passing messages among check nodes and variable nodes, updating probability distributions on nodes to perform error correction. This iterative approach contributes to the superior error correction performance of LDPC codes. LDPC coding approaches the Shannon limit at high signal-to-noise ratios (SNR), meaning its error correction performance is very close to the theoretical maximum. This makes LDPC coding a preferred choice in many communication systems.

The Bit-Flipping algorithm is an iterative decoding algorithm used for LDPC codes, aiming to reduce error rates and enhance error correction performance. This algorithm is primarily designed to correct variable node values to minimize discrepancies and improve decoding accuracy. Initially, all variable nodes are set to the received information, i.e., the bits transmitted over the channel. This serves as the starting point for the iterations. The Bit-Flipping algorithm iteratively updates the values of variable nodes by examining the parity of each check node. For each check node, it checks the parity of all connected variable nodes. If the sum of their parities is odd, it flips (changes) the value of the variable node. The iteration process can be repeated multiple times, or a stopping criterion can be set, such as reaching a maximum number of iterations or achieving a specific error rate.

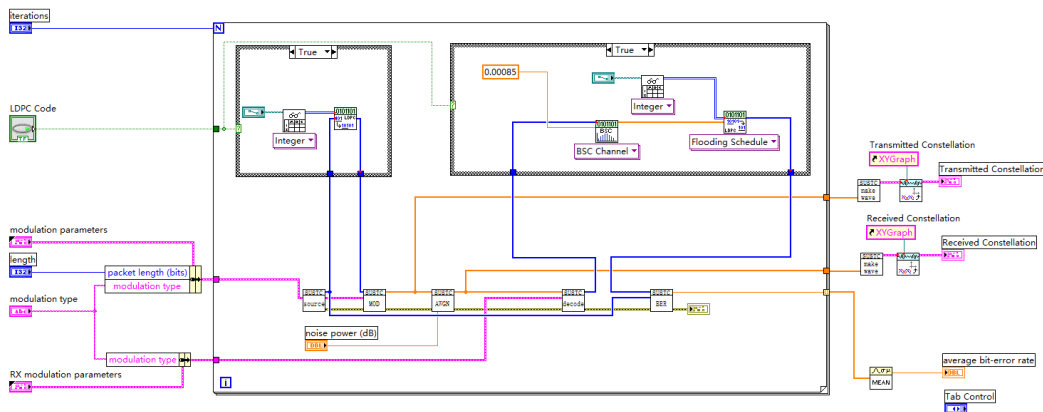


The Bit-Flipping algorithm is a simple yet effective iterative decoding method, particularly suitable for Low-Density Parity-Check codes. However, for LDPC codes with larger girth (the minimum length of cycles), Bit-Flipping may converge more slowly in certain cases. Therefore, in practical applications, it may be combined with other more sophisticated iterative decoding algorithms to improve performance.

## Lab results & Analysis:

### LDPC code-4QAM (block diagram, programming process, simulation results)

#### Block diagram:



#### Program process:

In the real communication system, after we generating the source bits, the next step we should do is to generate the channel code. So we need to have a generating matrix  $G$ , here we use a path to introduce this, since we use LDPC code here, we should also add a LDPC block. After generating the bit signal, we should then enter the bit into the modulation block. And the same is in the demodulation, we use  $H$  matrix to get the final signal message.

#### Simulation result:

4QAM = QPSK

modulation parameters

modulation type: QPSK packet length (bits): 500 number of data symbols (derived): 100

TX oversample factor: 20 TX sample rate: 4M

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

iterations: 100 average bit-error rate: 0.0008

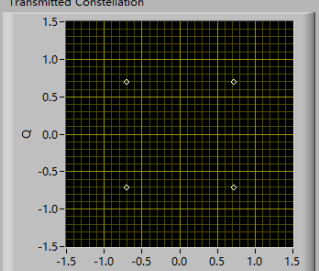
length: 1000 LDPC Code: ☒

modulation type: QPSK noise power (dB): -10

error out: status: ☒ code: 0 source:

Data Symbols Received Symbols

Transmitted Constellation



RX modulation parameters

modulation type: QPSK RX oversample factor: 20 RX Sample Rate: 4M number of data symbols (derived): 100

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

channel estimation/equalizer parameters

Equalization Method: Direct

channel estimate length: 1

equalizer length: 1

equalizer delay: -1 (set delay to -1 for equalizer to choose optimal delay)

synchronization options

Synchronization Method: Timing Estimation fixed offset: 0

Symbol Timing Recovery Method: Max Energy

Frame Detection Method: Sliding Correlator

Correct Frequency Offset: ☒

file path (G): C:\Users\Administrator\Desktop\GRegular

file path (H): C:\Users\Administrator\Desktop\HRegular

In the first situation, if we do not use the LDPC, we can see that the constellation points received is a little dispersed and the bit error rate equals to 0.0008. Although the number is quite small, it is still not perfect.



modulation parameters

modulation type: QPSK packet length (bits): 500 number of data symbols (derived): 100

TX oversample factor: 20 TX sample rate: 4M

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

iterations: 100 average bit-error rate: 0

length: 1000 LDPC Code

modulation type: QPSK

noise power (dB): -10

error out: status: code: 0 source:

Data Symbols Received Symbols

Transmitted Constellation

Received Constellation

RX modulation parameters

modulation type: QPSK RX oversample factor: 20 RX Sample Rate: 4M number of data symbols (derived): 100

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

channel estimation/equalizer parameters

Equalization Method: Direct

channel estimate length: 1

equalizer length: 1

equalizer delay: -1 (set delay to -1 for equalizer to choose optimal delay)

synchronization options

Synchronization Method: Timing Estimation fixed offset: 0

Symbol Timing Recovery Method: Max Energy

Frame Detection Method: Sliding Correlator

Correct Frequency Offset: Correct Frequency Offset

file path (G): C:\Users\Administrator\Desktop\GRegular

file path (H): C:\Users\Administrator\Desktop\HRegular

modulation parameters

modulation type: QPSK packet length (bits): 500 number of data symbols (derived): 100

TX oversample factor: 20 TX sample rate: 4M

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

iterations: 100 average bit-error rate: 0

length: 1000 LDPC Code

modulation type: QPSK

noise power (dB): -10

error out: status: code: 0 source:

Data Symbols Received Symbols

Received Constellation

RX modulation parameters

modulation type: QPSK RX oversample factor: 20 RX Sample Rate: 4M number of data symbols (derived): 100

control information for packet header/tail

Training Sequence Type: Length 11 Barker Sequence

Training Sequence: 0 707m + 707m i

Zero Pad Length: 8 Zero Pad Sequence: 1 + 0 i

pulse shaping parameters

modulation type: PSK pulse shaping filter: Root Raised filter parameter: 0.5 filter length (symbols): 8

channel estimation/equalizer parameters

Equalization Method: Direct

channel estimate length: 1

equalizer length: 1

equalizer delay: -1 (set delay to -1 for equalizer to choose optimal delay)

synchronization options

Synchronization Method: Timing Estimation fixed offset: 0

Symbol Timing Recovery Method: Max Energy

Frame Detection Method: Sliding Correlator

Correct Frequency Offset: Correct Frequency Offset

file path (G): C:\Users\Administrator\Desktop\GRegular

file path (H): C:\Users\Administrator\Desktop\HRegular

Here is the example that if we use LDPC code, from the simulation result, we can see that the bit error rate is exactly zero, which is better than the previous situation. However, we can see that running the program is much slower and this can be easily explained that when we use LDPC technology, there are more bits to be transferred.

## Experience

### Experience and Learning

In this experiment, we learned the low-density parity check code and deeply understood the channel coding algorithm in digital communication. Among them, we understand the generation matrix, check matrix, adjacency, Taylor graph, loop and other basic concepts in LDPC, and use two channel coding algorithms (7,4) Hamming code and LDPC code for simulation, and get the ideal simulation results.

Finally, we achieve 4QAM by replacing the block in the previous block. I realize the magic of the channel coding. Hope I can learn more in the next term's course.

### Contribution

Haodong Zhang completed the introduction and analysis of two channel coding algorithms: generating matrix, check matrix, concomitant, Taylor diagram, ring in LDPC, (7,4) Hamming code and LDPC code, and Yihang Song completed the experimental validation of the 4QAM modulation LDPC code, and got the result of small bit error rate

**Score**

98