Construction of Low Density Parity Check Codes Without Short Cycles

Lizhi Wu

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

March 2003

ABSTRACT

Construction of Low Density Parity Check Codes Without Short Cycles

Lizhi Wu

With the rapid expansion of communication networks, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. Many efficient codes have been developed. The LDPC code is one of them. In this thesis, the sum-product algorithm is used in the decoding of LDPC codes. Some schemes for encoding LDPC codes have been studied. In particular, two methods of producing regular **H** matrices have been attempted that include short cycles of length four with code rates of 0.5, and we present three schemes of finding regular **H** matrices which do not include short cycles of length four with code rate being 0.5. The effect of short cycles in the bipartite graph of regular LDPC codes has been considered. The simulation results show that the BER performances of regular **H** matrices that do not include short cycles of length four based on BPSK or 8PSK on AWGN channel is better than those of regular **H** matrices that include short cycles of length four. In conclusion, in order to obtain good performance with LDPC code, one should design **H** matrix related to bipartite graph without short cycles.

Dedicated to my mother, wife, and son .....

ACKNOWLEDGEMENTS

It is with the utmost sincerity that I would like to express my appreciation to my supervisors Dr. M. R. Soleymani, for his continuous guidance and encouragement. Without his patient instruction and stimulating suggestions, I would not have completed my thesis. I would also like to thank Dr. Paul Guinand and Dr. Xiangmin Li who provided a lot of help and instructions. Their instructions have deeply attracted me to engage in researching in the area of communications, a field so far from electrical power system in which I worked in the past. Therefore, I would really like to continue further research with Dr. M. R. Soleymani .

I would also like to thank my classmates in the Wireless and Satellite Communications Laboratory. It is because of their help that I overcame some technical problems when my knowledge to solve some problem had been exhausted.

Finally, I want to thank my lovely wife and son. Their trust, love and support inspired me throughout all the three years.

I am really obligated to all of you. Let me say it again. Thank you all very much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

$a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$
A symbol of three bits mapped to a complex value after 8PSK modulation

A(m) The set of bits n that take part in check m

B(n) The set of checks that bit n takes part in

c Ones in each column of the binary parity-check matrix

C_M^c all columns with c fixed numbers 1 or c weights

Δq_{mn} The difference $q_{mn}^{x_n=0} - q_{mn}^{x_n=1}$

Δr_{mn} The difference $r_{mn}^{x_n=0} - r_{mn}^{x_n=1}$

$f^1_{n(i)}$ The effective probability of 1 in bit i of the noise vector **n**

G A generator matrix of LDPC code

g An integer

H Binary parity-check matrix

H' Binary parity-check matrix, $\mathbf{H'} = \mathbf{H_2}^{-1}\mathbf{H}$

$\mathbf{H_1}$ and $\mathbf{H_2}$ Very sparse matrices

h1, h2, h3, h4, h5 Binary submatrices

H_{mn} The entry of the m row and the n column of the **H** matrix.

hT Binary lower triangular submatrix form

$\mathbf{I_k}$ A k×k identity matrix

$\mathbf{I_M}$ Binary identity matrix $\mathbf{I_M} = \mathbf{H_2}^{-1}\mathbf{H_2}$

I_{nt} An integer

k Binary source information string

k An integer

l_0, l_1, l_2 A position index of three bits mapped to a complex value after 8PSK modulation

l_{ij} The definition of log-likelihood ratio

$l_{ij} = \Lambda(n_{ij})$

M An integer

m An information sequence

n Transmitted blocklength of a LDPC code (integer)

N Integer

n Noise binary vector

n_{ij} Noise bit, i and j are index

N_{size} The sample size

$\hat{p}$ The sample proportion

P Binary parity block in a systematic parity check matrix or generator matrix

$\mathbf{p}_1$, $\mathbf{p}_2$ The parity part of binary transmitted block

$q_{mn}^{x_n}$ The message passed from the bit node to the check node along the edge between connecting points

R Code rate

r The received signals

r_{i0}, r_{i1}, r_{i2} The three binary bits after demodulation of soft decision

$r_{mn}^{x_n}$ The message passed from the check node to the bit node along the edge between connecting points

rw Number of 1's in each row of the binary parity-check matrix

s An integer

s Binary source information string

ss An real number

S The syndrome of received **r**

t_t Time

t Binary transmitted block , or transmitted string

t' The estimate of the transmitted signal vector

t_{i0} , t_{i1}, t_{i2} Three binary bits , i is index

v_i A zero-mean Gaussian noise of variance σ^2

v_I A real part of complex noise symbol

v_Q A imaginary part of complex noise symbol

W1, W2, W3, W4, W5

Some methods of constructing M rows by 2*M columns binary parity-check matrix

$w(t)$ A pulse

x_I A real part of three bits mapped to a complex symbol after 8PSK modulation

$\mathbf{x_n}$ The noise vector

X_{num} The number of successes

x_Q Imaginary part of three bits mapped to a complex symbol after 8PSK modulation

$y_I + jy_Q = (x_I + v_I) + j(x_Q + v_Q)$

y_i The received signal

$\mathbf{z}$ The syndrome vector

Z_{sn} Approximately standard normally distributed [26]

ε_s The energy in the waveform w(t)

σ^2 Variance of channel noise

Φ_{mn} Coefficient

Φ_n Coefficient

Chapter 1

Introduction

1.1 Structure of Digital Communication Systems

In recent years, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. In order to understand the role of error control coding in these systems, a model of a general communication system is shown in Fig. 1.1.

In the system, the task of the transmitter is to transform the information generated by a source into a form that can tolerate the interference of noise over the transmission medium.

The information source can be either a person, a machine, words or code symbols. The source output, which is to be communicated to the destination, can be either a continuous waveform or a sequence of discrete symbols.

Because the output of the information source is in general not suitable for transmission as it might contain too much redundancy, the source encoder

transforms the source output into a sequence of binary digits with minimum redundancy.

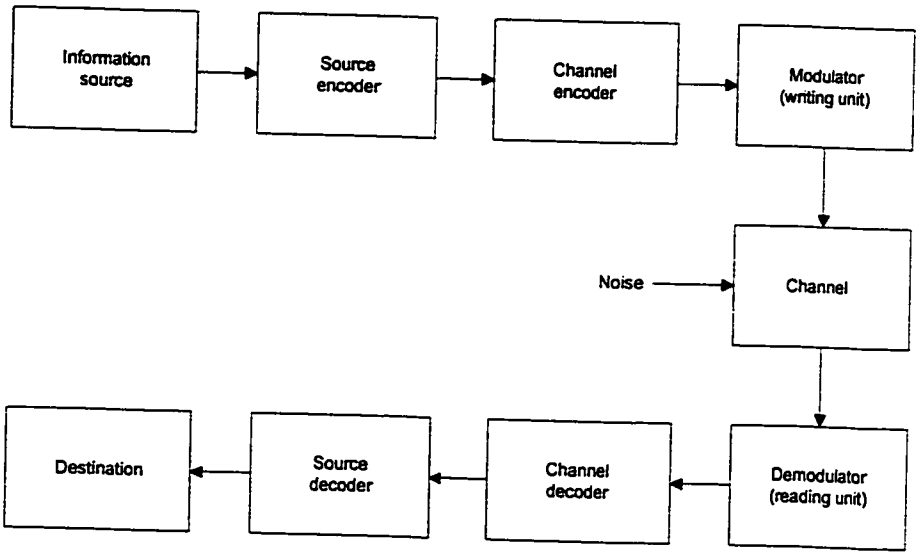


Fig. 1.1 Block diagram of a digital communication system

The channel encoder transforms the information sequence into a discrete encoded sequence called a code word. The channel impairments cause errors in the received signal. The channel encoder is incorporated in the system to add redundancy to the information sequence in order to minimize transmission errors.

The output of the channel encoder is not normally suitable for transmission. The modulator transforms each output symbol of the channel encoder into a waveform of duration T seconds that are suitable for transmission. The waveform enters the channel (or storage medium) and is corrupted by noise. Typical transmission channels include wire lines, microwave radio links over free space, satellite links, fiber optic channels, etc. On a telephone line, the disturbance may come from switching impulse noise, crosstalk from other lines, thermal noise, or lightning. In

the receiver, the demodulator (or reading unit) processes each received waveform of duration T and produces an output that may be discrete (quantized) or continuous (unquantized). The demodulator outputs corresponding to the encoded sequence are called the received sequence.

The basic decoding idea is based on the rules of channel encoding and the noise characteristics of the channel. The channel decoder makes an estimate of the actually transmitted message. Although the noise may cause some decoding errors, the goal of the decoder is to minimize the effect of channel noise.

Based on the source encoding rule, the source decoder transforms the estimated sequence into an estimate of the source output and delivers this estimate to the user (destination).

It is possible for a proper design of the transmitter-receiver system to remove or decrease the effects of attenuation and distortion and to minimize the interference of noise. But, the impact of noise can not be totally removed because the complete knowledge of noise still confuse us.

1.2 Motivation

Assume we want to send information through a noisy communication channel reliably. Shannon proved that arbitrarily reliable transmission is possible through this channel if the information rate in bits per channel use is less than the capacity of the channel. The performance of error-correcting codes was bounded by Shannon in 1948[1]. But, practical coding methods did not approach the Shannon limit until the discovery of Turbo codes in 1993 [2]. Therefore, the Turbo codes opened the new era of near Shannon limit performance for the additive white Gaussian noise channel, significantly improving on all previous methods. Not only have the Turbo codes approached the Shannon limit, but also Low Density Parity Check codes (LDPC) that Gallager found in 1963 [3] and MacKay and Neal [4] [5] rediscovered provide an excellent performance.

Low density parity check codes are a class of linear error-correcting block codes. In 1963, Gallager introduced the LDPC codes which are defined in terms of a sparse parity-check matrix: each codeword satisfies a small number of linear constraints. Gallager proposed layouts, a description of an iterative probabilistic decoding algorithm and theory that in some aspects goes beyond what is known today as Turbo codes. Unfortunately, LDPC codes were almost forgotten for more than thirty years. It wasn't until 1995, that the excellent performance of LDPC codes was found by MacKay and Neal [5].

Gallager considered codes whose parity check matrix had fixed row and column weights (a construction which is refered to as 'regular'). MacKay, Neal, Michael G.Luby, Michael Mitzenmacher, M Amin Shokrollahi, Daniel A. Spielman, and Matthew C. Davey gave up this constraint and produced 'irregular' LDPC codes that have a variety of row and column weights.[6] [7]. High weight columns can make the decoder identify some errors quickly; The remaining errors are easier to correct.

Some performance comparison of LDPC codes have been shown in some papers. MacKay [4] reported a good regular binary LDPC code which was introduced by Gallager in 1962. Its bit-error probability is at 10^{-6} when the Eb/No is 0.9 dB , code rate ¼, and blocklength 40000 bits. Luby ,Mitzenmacher, Shokrollahi and Spielman first showed irregular construction of LDPC codes and reported the performance of irregular binary LDPC codes in 1998 [8]. Its bit-error probability is 10^{-3} when value of Eb/No is 0.5 dB, code rate ¼ , and blocklength 64000 bits. The code 'Irreg GF(8)' beats the best known turbo codes, at least for bit error rates above 10^{-5}, making it the best (at this error rate) error correcting code of rate ¼ for the Gaussian channel currently known. The error-correction performance is better than that of the turbo code; besides, its blocklength is less than that of the turbo code. Another important difference between turbo codes and LDPC codes is that all errors made by the LDPC decoding algorithm are detected errors. In other words, the decoder reports the fact that some blocks have been incorrectly decoded.

Richardson, Shokrollahi and Urbanke [9] reported that extremely long blocklength (10^6 bits) irregular LDPC codes can perform within 0.1 dB of the Shannon limit.

1.3 Organization of The Thesis

In Chapter 2, a brief review of low-density parity-check codes is given. Gallager's codes received little attention prior to 1995; but now, there is a huge interest since their performance has been recognized . We describe several methods for constructing a suitable low-density matrix. The matrices constructed by method we denote as W1 and W2 will be suitable bipartite graph with short cycle of length four and code rate 0.5. The matrices constructed by W3, W4, and W5 will correspond to the bipartite graph without short cycle of length four and code rate 0.5. The definitions of W1, W2, W3, W4, and W5 is presented in Chapter 2.

In Chapter 3, an efficient encoding based on approximate lower triangular form is given [10]. The method formulates an algorithm for constructing efficient encoders for LDPC codes. The efficiency of the encoder arises from the sparseness of the parity-check matrix H and the algorithm can be applied to any sparse H matrix. We also introduce a systematic encoders based on a generator matrix.[4].

In Chapter 4, we introduce binary input symmetric-output memoryless channels and details of the mathematical equations. In Section 4.2, we show how to map bits to BPSK based on AWGN. In Section 4.3, Gray mapping of bits to 8PSK symbols on AWGN is presented. In Section 4.4, we present an anti-Gray mapping of bits to 8PSK on AWGN.

In Chapter 5, we describe what the bipartite graph is, and describe the condition of having short cycles in the bipartite graph. Then, some basic concepts of decoding LDPC codes are presented in Section 5.2. In Section 5.3, we state the sum-product algorithm in detail.

In Chapter 6, five different H matrices whose sizes are 30*60, 48*96, 60*120, 92*192, 252*504 are discussed. The bipartite graphs related to these H matrices do not include short cycles of length four. In Section 6.1, we compare the performance of mapping of bits to BPSK in the AWGN channel for different H matrices. In Section 6.2, performance comparison for Gray mapping of bits to 8PSK combined on AWGN is presented. In Section 6.3, we also compare the performance with an anti-Gray mapping of bits to 8PSK on AWGN.

In Chapter 7, we summarize, and provide conclusions.

1.4 Contributions of the thesis

The main contributions of the thesis are stated as follows:

- Present two methods W1 and W2 of designing regular **H** matrices which include short cycles of length four with code rates of 0.5. The details are shown in Section 2.4.3.

- Present three schemes W3, W4, and W5 of finding regular **H** matrices which do not include short cycles of length four with code rates being 0.5. The details have also been shown in Section 2.4.3.

- Comparison of the effect of short cycles in the bipartite graph of regular LDPC codes with 0.5 code rates. The simulation results show that the BER performances of regular **H** matrices which do not include short cycles of length four based on BPSK on AWGN channel is better than those of regular **H** matrices which include short cycles of length four. The difference in BER performances between having short cycles and no short cycles in the bipartite graphs have been shown in Section 6.2 .

- Similarly, we analyze the BER performances of some regular **H** matrices that are used to draw bipartite graphs which do not include short cycles of length four and decode LDPC codes based on Gray mapping 8PSK on AWGN channel. The performances show that they are better than those of regular **H** matrices that are used to draw bipartite graphs that include short cycles of length four and decode LDPC codes . The bar charts of statistical scheme clearly express results that show the difference in BER performances of those having short cycles and those with no short cycles in bipartite graph. The details have been shown in Section 6.3.

- We also research the BER performance of LDPC codes combined with anti-Gray mapping 8PSK on AWGN channel and compare regular **H** matrices with short cycles of length four and **H** matrices with no short cycles of length four for a code rate of 0.5.

Chapter 2

Low Density Parity Check Codes

2.1 Developing History of Low Density Parity Check Codes

Low–density parity-check codes were introduced by Gallager in his Ph.D thesis[3]. The term low-density means that the number of 1's in each row and column of the parity check matrix is small compared to the block length. LDPC codes are a class of linear error-correcting codes. Linear codes are defined and described in terms of generator and parity-check matrices. The codes can be explained as using a generator matrix **G** to map information **k** to transmitted blocks **t** called codewords. For a generator matrix, **G**, there is a parity-check matrix **H**. In terms of the related parity-check matrix **H**, all codewords must satisfy $\mathbf{Ht=0}$.

In 1963, Gallager defined (n, c, rw) LDPC codes which have a blocklength n, exactly c ones per column, and rw ones per row in a parity check matrix, where $c \geq 3$. Fig. 2.1 shows the H matrix of a (20, 3, 4) code structured by Gallager.[3].

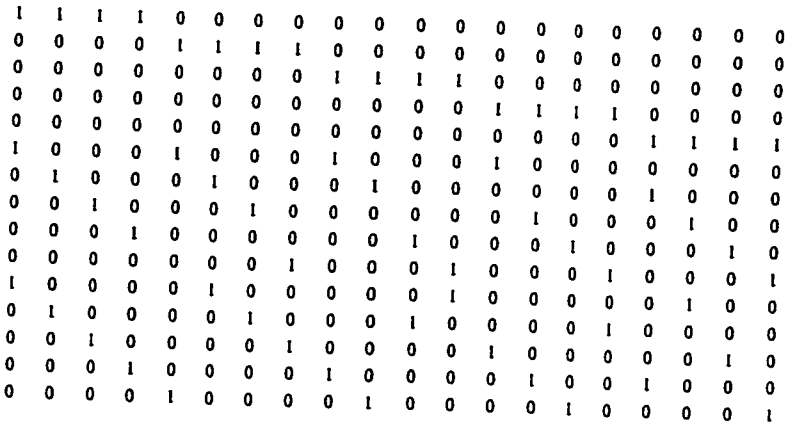


Fig. 2.1 Example of a parity-check matrix for a (20,3,4) LDPC code

Why had Gallager's codes, introduced in 1963, had been forgotten for a long time? The main reason is that the storage requirements of encoding, and the computational demands of decoding made them impractical at that time.

In 1981, Tanner[4] described generalized constraints of codes defined on bipartite graphs , which represented the relationship between codeword symbols and general constraints, and stated the optimality of the sum-product algorithm for decoding codes based on the bipartite graphs.

In 1995, Wiberg, Loeliger and Koetter [12] [13] rediscovered Tanner's constraints and introduced states into Tanner's graphs, as a result, connections to trellises and to turbo codes became possible. Wiberg carried out some performance analysis for iterative decoding, and evaluated some performance of some low-density parity check codes.

The most important and interesting thing is that, with the introduction of LDPC codes by Mackay et. al [4][5][14] and Spielman et al [15], the connections to Bayesian networks were recognized [16][17]. MacKay showed that LDPC codes are 'very good' when decoded with an optimal decoder.

In 1998, Davey and MacKay [18] presented non-binary Gallager's codes and irregular non-binary Gallager's codes, and reported that performance of those could be significantly improved. Luby, Mitzenmacher, Shokrollahi and Spielman [8] defined irregular LDPC codes which have highly non-uniform column-weight

distribution. The irregular LDPC codes could improve the performance of decoding. Finally, they proved that the performance of irregular codes exceeds that of turbo codes.

It is well known that the LDPC codes have become more and more important in communication field. A lot of papers have been published on this subject.

2.2 Bipartite graph and short cycles

In this section, we will present the bipartite graph and the related short cycles in the more detail. The bipartite graph corresponding to the parity check matrix **H** is also called the Tanner graph [11]. A Tanner graph is shown in Figure 2.2.

Example 2.1 Consider the following parity check matrix

$$H=\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

What are its linear equation system of six variables, Tanner graph, and corresponding bipartite graph ?

Clearly, if the six variables are $\mathbf{x} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, then $\mathbf{Hx} = \mathbf{0}$, and their linear equations are the following:

$$x_1 + x_2 + x_6 = 0$$
$$x_2 + x_3 + x_4 = 0$$
$$x_3 + x_5 + x_6 = 0$$
$$x_1 + x_4 + x_5 = 0$$

Then, we can find the Tanner graph corresponding to the linear equations.

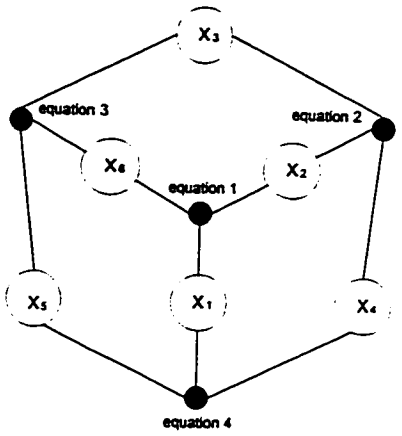


Fig. 2.2 An example of a Tanner graph

According to the Fig. 2.2, we find certain edges which exist between equation points and variables x. Each variable x, is put on the left and named a "left node", "bit node" or "message node" represents a bit of the codeword . Each equation point, is put on the right and named a "right node" or "check node", represents a parity check of the code. According to the definition, we can rearrange Fig.2.2 and get bipartite graph of Fig 2.3 . In Fig. 2.3, an edge exists between a bit node and a check node only when there is a 1 element in the corresponding to position in the parity check matrix. The situation is absolutely the same as Fig 2.2 . Therefore, a Tanner graph is a bipartite graph representation for a check structure.

In example 2.1, we describe what the bipartite graph is and what the relationship between the bipartite graph and Tanner graph is. However, the number of 0 elements is equal to number of 1 elements in the parity check **H** matrix. As a result, the parity check **H** matrix is not low density because, in low density parity check codes, the number of 1 elements is less than the number of 0 elements in the **H** matrix.

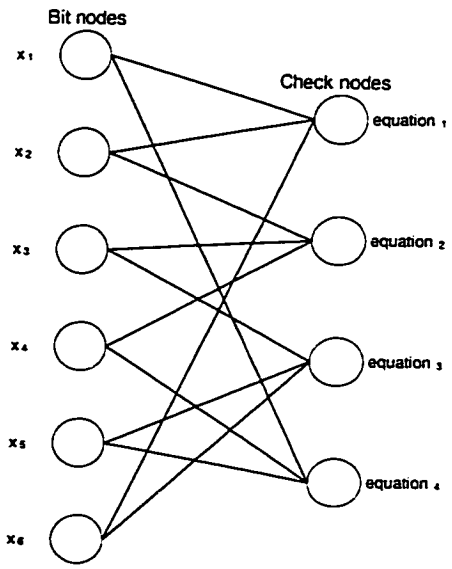


Fig. 2.3 The bipartite graph in example 2.2

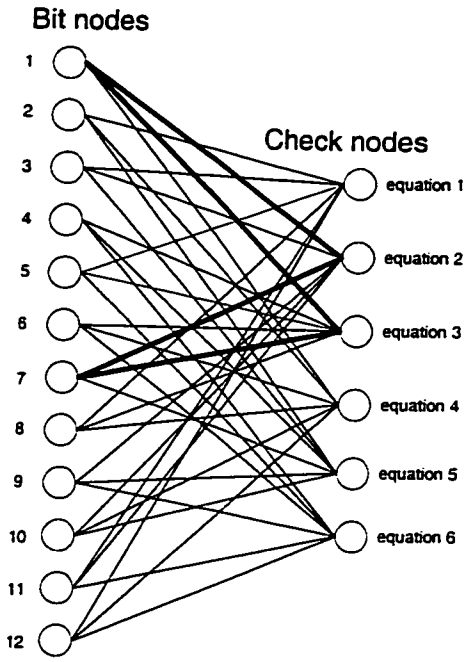
In order to describe the relationship between bipartite graph and short cycles in low density parity check codes, we give a six rows by twelve columns **H** matrix as follows.

From the **H** matrix, we find that there are two overlaps between the first column and the seventh column. The overlaps are shown in the Fig. 2.4 (b). Obviously, the four dark lines have one to one relationship with the 1 elements of the two overlaps in the **H** matrix.

In Fig. 2.4 (b), the four dark lines form a cycle. In general, we will call the cycle a short cycle when its length is four. In chapter 6 , we will find that a short cycle affects the performance when we do some simulations.

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ \{1\} & 0 & 1 & 0 & 0 & 0 & \{1\} & 0 & 1 & 1 & 1 & 0 \\ \{1\} & 0 & 0 & 1 & 1 & 1 & \{1\} & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

(a) A low density parity check matrix in Fig 2.4



(b) Bipartite graph corresponding to **H** matrix in Fig. 2.4

Fig. 2.4 An example of a parity check matrix and its corresponding bipartite graph

2.3 Rule of constructing regular LDPC codes

Gallager's codes have fixed row and column weights. However, with the development of LDPC codes theory, the constraint has been relaxed. If a column weight is greater than 2 and the parity check matrix **H** is produced at random, we want to constrain the distribution of row and column weights to be as uniform as possible. Besides, we also require that there is no overlap between any two columns in the matrix **H**.

Because the matrix **H** is produced randomly, the matrix **H** is not in systematic form, and we can use Gaussian elimination and reordering of columns to derive an equivalent parity-check matrix in a systematic form.

This thesis concentrates on LDPC codes (N,3,6) with no short cycles in their bipartite graph and investigates the performance of these codes in terms of BER vs E_b/N_0 (dB).

2.4 Construction of LDPC matrices

2.4.1 Gallager's construction of H matrices

In order to express Gallager's construction of **H** matrices, we use an example of a parity-check matrix for a (20,3,4) LDPC code. N is 20, c =3, and rw = 4 .

Firstly, consider an Nc/rw by N matrix shown in Fig. 2.5. The matrix has been divided into 12 submatrices, and each has 5 rows and 5 columns. The first row and column of submatrices are identity matrices. The rest of the submatrices contain the letter L in each main diagonal and X in the other positions. The next step is to find 5 non zero elements in each submatrix and guarantee that there is no more than one overlap between any two columns in the **H** matrix . The letter X is used to denote an acceptable position in which to put a 1 without forming more than one overlap between any two columns in the **H** matrix. The letter L expresses an unacceptable position because the 1 of the position must create an overlap of more than one between some two columns in the **H** matrix. (see [3] for more details).

Secondly, we choose a submatrix with L and X elements, and in the first row, arbitrarily pick a position containing a X and make the X a 1. Meanwhile, the rest of the positions in the submatrix that are in the same row or column cannot be made 1. Also, if the same position of some other submatrix which is corresponding to the chosen position of the submatrix can not be made a 1 it can only be a 0. See Fig.2.6 for more details.

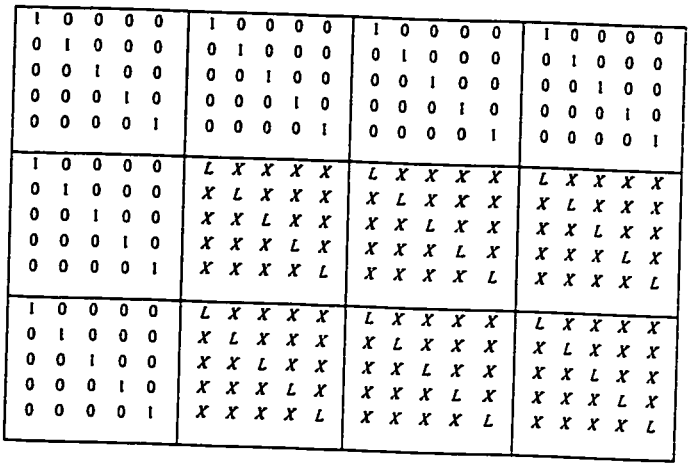


Fig. 2.5 The first step of Gallager's construction of H matrices

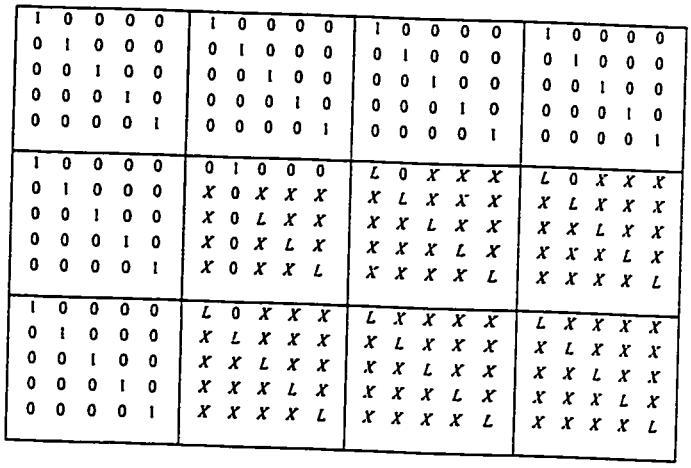


Fig. 2.6 The second step of Gallager's construction of H matrices

According to the method above, we can continually do the transformation in the second row , the third row , etc, and other submatrices. Finally , each row has only one 1 element in each submatrix . The matrix which is made from the submatrices does not include more than one overlap between any two columns.

2.4.2 MacKay's description of H matrices.

In 1999, MacKay[4] introduced methods to construct parity check matrices which are related to the bipartite graph . Fig.2.4 shows the relationship between the **H** matrix and the bipartite graph.

From the example, we find that there is at least a cycle of length 4 (dark lines) in the bipartite graph. MacKay [4] reported some construction methods of matrices without cycles of length 4. They are as follows:

♣ Construction 1A.

If we fix a weight per column c (e.g., c=3) and construct an M rows by N columns matrix at random with weight per row as uniform as possible, the overlap between any two columns is not more than 1. Shown in figure 2.7 (a).

♣ Construction 2A.

Design m/2 columns with weight 2, these are constructed without overlap between any two columns. The rest of the columns are produced at random with weight 3, with the weight per row as uniform as possible, and overlap between any two columns of the entire matrix not more than 1. Then, the M rows by N columns matrix can be constructed. Shown in Fig. 2.7 (b).

♣ Construction 1B and 2B.

If we choose some columns carefully from a 1A or a 2A matrix , delete them and make the bipartite graph of the matrix have no short cycles of length less than some specified length l , (e.g., l=6).

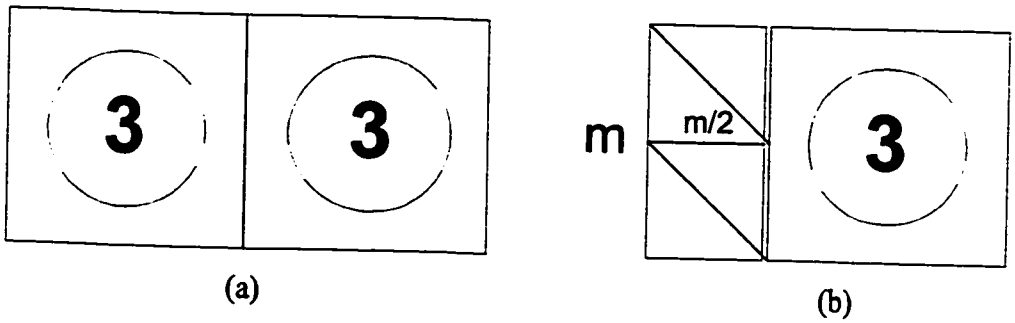


Fig. 2.7 (a) Construction 1A for a Gallager code for t=3, t_r=6, and code rate =1/2,
(b) Construction 2A for a Gallager code with code rate 1/3.

2.4.3 New schemes for constructing LDPC codes matrices

Although arbitrary sparse **H** matrices which satisfy the definition of Gallager's codes can produce correct encoding, different H matrices which are constructed by different schemes will have different encoding and decoding. The low density parity-check matrices can also be created as follows:

W1: An M rows by 2*M columns matrix is created. Firstly, fix M rows and make every row have a fixed number of 1 elements at random, (for example, six 1 elements), then by modifying 1 position in each row, make every column have a fixed number of 1 elements, (for example, three 1 elements), and still keep in every row a fixed number of 1 elements. Fig. 2.8 shows the scheme. We can apply the matrix to encode and decode a regular LDPC codes with code rate =0.5.

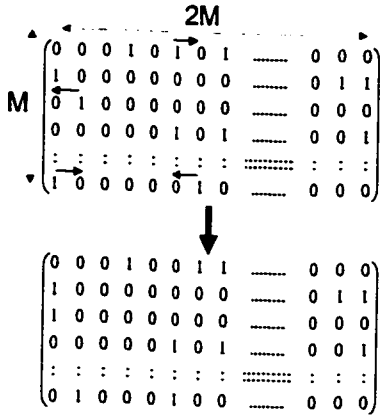


Fig. 2.8 The W1 scheme for constructing H matrix

W2: An M rows by 2*M columns matrix is created. Firstly, fix 2*M columns and make every column have a fixed number of 1 elements at random, (for example, three 1 elements), then modifying positions of 1 in every column, make every row have a fixed number of 1 elements, (for example, six elements 1) and still keep a fixed number of 1 elements in every column. We can also apply the matrix to encode and decode a regular LDPC codes with code rate =0.5.

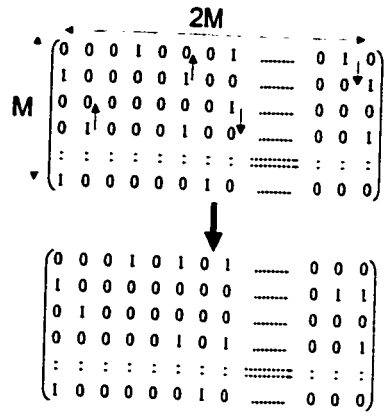


Fig. 2.9 The W2 scheme for constructing H matrix

W3: In order to create M rows by 2*M columns matrix , firstly, find all columns with a fixed number of 1 elements or weights (c). The total number of such columns is $C_M{}^c = \binom{M}{c}$. Then, from the $\binom{M}{c}$ columns, choose 2*M columns with no more than an overlap of elements 1 between any two columns, keep every row with fixed 1 elements (six elements 1, in the thesis), and every column with another fixed number of 1s (in the case , three elements 1) . For example, if we want to build a 50*100 H matrix with three weights for every column and six weights for every row. Firstly, find all $\binom{50}{3} = 117600$ columns. Then, we can choose one hundred columns from the 970200 columns without more than an overlap of elements 1 between the any two columns of the one hundred columns. Also, every row in the H matrix constructed must contain six elements 1. Thus, the 50*100 H matrix is

constructed. However, the scheme is only suitable for small size H matrices because of memory usage. Anyway, we can still employ the small size H matrix to encode and decode a regular LDPC codes with code rate =0.5 .

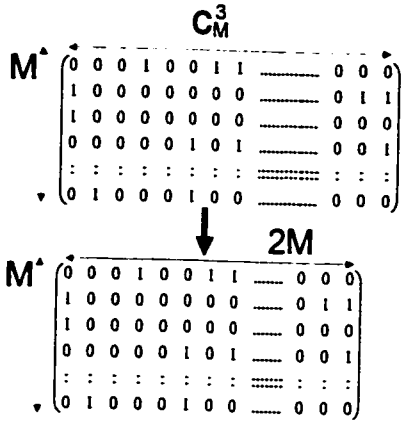


Fig. 2.10 The W3 scheme for constructing H matrix

W4: If an M rows by 2*M columns matrix is created, we can firstly fix the first column which has a fixed number of 1 elements as uniformly as possible. Next, produce another column which has the same number of 1 elements at random as uniformly as possible, but the overlap of elements 1 between the column and the first column is not more than 1. If this is true, the second column has been fixed. Next, produce the third column which has the same number of fixed 1 elements at random as uniformly as possible, and check whether there is any overlap of elements 1 of more than 1 between the column and all former columns. If there is no overlap, the third column is been fixed. Then, proceeding step by step, an M*2M **H** matrix without has more than 1 overlap of elements 1 is obtained. However, when we use the method to produce the **H** matrix, we may meet the problem of no solution in the last few columns. If the problem arises, we want to put the fixed number of 1 elements in the last few columns and keep every row having the same weights. But, the result may cause some overlaps of more than 1 in some columns. In order to solve the problem, we need to exchange the 1 position between the columns with more than 1 overlap of elements 1 and confirm that for whole matrix the number of

more than 1 overlaps of elements 1 are decreased after the exchanging. Finally, the H matrix will be constructed. The matrix can be used to produce regular LDPC codes with code rate 0.5.

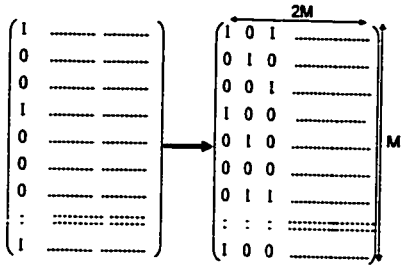


Fig. 2.11 The W4 scheme for constructing H matrix

W5: When we want to create M rows by 2*M columns H matrix, firstly check whether the M rows divided by integer I_{nt}, for example 3. Next, if it is true, produce the M/I_{nt} by M/I_{nt} identical **H** submatrices which are put in the former M/I_{nt} columns of the H matrix . Then, produce the other M - M/I_{nt} columns at which every column have only I_{nt} elements 1 at random as uniform as possible and there is not more than 1 overlap of elements 1 between any two columns in the **H** matrix. However, when one uses the method of producing the **H** matrix, one may also meet the condition of no solution in the last few columns. If this is a problem, one want to put the I_{nt} elements 1 into the last few columns and keep every row having the same weights. But, the result may cause some more than 1 overlaps of elements 1 between some columns. In order to overcome the problem, one needs to exchange the positions of 1 between the columns of the rest M - M/I_{nt} columns with more than one overlap of elements 1 and confirm the number of more than one overlaps of elements 1 in the H matrix is decreased after the exchanging. Finally, the H matrix will be constructed. The matrix can also be used to make a regular LDPC codes with code rate =0.5.

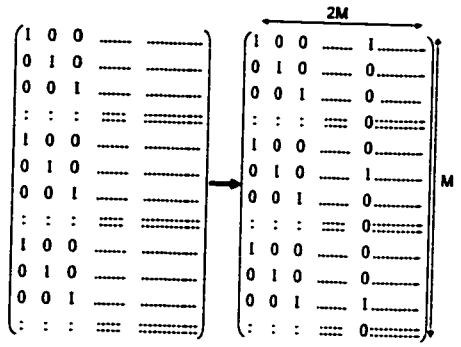


Fig. 2.12 The W5 scheme for constructing H matrix

Using one of the above five methods, one can get five different **H** matrices. Each **H** matrix will correspond to a bipartite graph. The bipartite graphs that are produced by methods W1 and W2 may include short cycles of length four. Meanwhile, the bipartite graphs which are produced by the W3, W4, and W5 will not include short cycles of length 4.

2.4.4 Some comments on the H matrices constructed using the new schemes

- H matrices constructed using W1 and W2.

One assumes the **H** matrix is M rows by 2M columns. One also suppose that every column has a fixed weight (i.e. 3) for W2. Then, the total number of 1s is 3*columns.

That is 3*No. of columns =2*3*rows=6*rows

Obviously, weight of every row can be kept as 6. Suppose that the ratio of the number of columns to the number of rows, is

$$ss = \frac{columns}{rows}$$

is not an integer and let n be the weight of every column. Then, the weight of every row is n*ss. Assume n=3, ss=1.5, then n*ss =4.5. In fact, we can not keep every row to have weight 4.5. As a result, we always assume that $ss = \frac{columns}{rows}$ is an integer. In conclusion, if the weight of columns is fixed (n), the weight of every row can always be kept as another fixed weight ss*n after making some permutation of weight according to rule of W2 as well as W1. But, the schemes can not eliminate short cycle of length 4 in their bipartite graph.

- H matrices constructed by use of W3, W4, and W5

Although the methods of constructing **H** matrix by use of W3, W4, and W5 have some differences , they all can be used to construct the same size **H** matrix. Using a C program, M rows by 2M columns H matrices of (N,3,6) was constructed one by one. The results show that the minimum size of M rows by 2M columns **H** matrix with no more than 1 overlap between any two columns is 7 by 14. In other words, the lower bound of the M rows by 2M columns **H** matrix of (N,3,6) is 7 by 14 if short cycles are to be avoided . Therefore, **H** matrix of M rows by 2M columns without short cycle of length 4 can be built for encoding and decoding LDPC codes.

In conclusion, the three schemes can be easily realized by C program.

Chapter 3

Encoding Methods of LDPC Codes

It is well known that the low density parity check codes have high encoding complexity. In [15][27], a scheme of choosing the number of stages and the relative size of each stage was suggested; therefore, one can construct encodable and decodable linear codes. In [6], the scheme is to force the parity check matrix into an almost lower triangular form. Although these schemes can be used to encode LDPC codes, their complexity is high. Recently, some new schemes to encode LDPC codes have been presented. In Section 3.2, we will present an efficient encoding scheme in detail. But, we first present a systematic encoding scheme based on a generator matrix in Section 3.1.

3.1 Systematic encoders based on a generator matrix

Suppose the parity check matrix can be written as follows:

$$\mathbf{H} = [\mathbf{H}_1 \mid \mathbf{H}_2] \tag{3.1}$$

Where $\mathbf{H}_1$ and $\mathbf{H}_2$ are two very sparse matrices and the rows in $\mathbf{H}$ are linearly independent. The matrix $\mathbf{H}_2$ is a square k*k matrix and invertible. If $\mathbf{H}_2$ is not invertible, we reorder the columns of the $\mathbf{H}$ matrix. The new $\mathbf{H}$ matrix defines a code which is equivalent to the original code , but the codewords which are produced by

the new **H** matrix will be different from those encoded by previous **H** matrix. The matrix $\mathbf{H}_1$ is a rectangular k*(n-k). We can then derive an equivalent parity check matrix[4]:

$$\mathbf{H'} = \mathbf{H}_2^{-1}\,\mathbf{H} = \mathbf{H}_2^{-1}[\mathbf{H}_1 \mid \mathbf{H}_2] = [\,\mathbf{P} \mid \mathbf{I}_M\,] \tag{3.2}$$

Where $\mathbf{P} = \mathbf{H}_2^{-1}\mathbf{H}_1$ and $\mathbf{I}_M = \mathbf{H}_2^{-1}\mathbf{H}_2$

The generator matrix of the LDPC is given by,

$$\mathbf{G}^T = \begin{bmatrix} I_K \\ P \end{bmatrix} = \begin{bmatrix} I_K \\ H_2^{-1}H_1 \end{bmatrix} \tag{3.3}$$

and

$$\mathbf{t} = \mathbf{G}^T\,\mathbf{k} \tag{3.4}$$

Where $\mathbf{I}_K$ is the k*k identity matrix . **t** is a codeword and **k** is source information. $\mathbf{H} * \mathbf{G}^T = \mathbf{H'} * \mathbf{G}^T = \mathbf{0}$. Example 3.1 shows the procedure of the LDPC codes encoding.

Example3.1: Suppose we have a **H** matrix and information sequence k_0=(0,0,0,0,0,1), k_1=(0,0,0,0,1,0), k_2=(0,0,0,1,0,0).

$$H = [H_1 \quad H_2] = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

After using Gaussian elimination, the **H** matrix has the following form:

$$H' = [P \quad I_n] = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad H_2^{-1} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$G^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

According to formula (3.4), we can get the three codewords which are $\mathbf{t}_1$=(0,0,0,0,0,1,0,0,1,0,0,0), $\mathbf{t}_2$ = (0,0,0,0,1,0,1,0,1,0,1,0) , $\mathbf{t}_3$=(0,0,0,1,0,0,1,0,1,1,1,1). The three codewords satisfy the condition $\mathbf{Ht}^T$=0.

Obviously, the scheme is available for encoding LDPC codes. But, because we use the Gaussian elimination in the scheme, the complexity of calculation is high. Furthermore, when the length of the codewords is increased, the complexity of calculation will increase. In order to decrease the complexity, we introduce another scheme as follows.

3.2 An efficient encoding based on approximate lower triangular form

In [10], an efficient encoder for low density parity check codes is introduced. Now, let us review the algorithm briefly. The algorithm of the encoding comes from the sparseness of the parity check matrix **H**. Firstly, we suppose throughout that the rows of **H** are linearly independent. If the rows are linearly dependent, some redundant rows from **H** in the encoding process can be removed.

If we have an s×n parity check **H** matrix, LDPC codes can be expressed as the null space of a parity check matrix **H** , i.e., **t** is a codeword if and only if

$$\mathbf{Ht} = \mathbf{0} \tag{3.5}$$

So when we have an **H** matrix, we perform row and column permutations so as to make the parity check matrix become of the form indicated in Fig.3.1.[10]. The new **H** matrix is defined as being of approximate lower triangular form. The **H** matrix is still sparse because the transformation of the **H** matrix was accomplished solely by permutation.

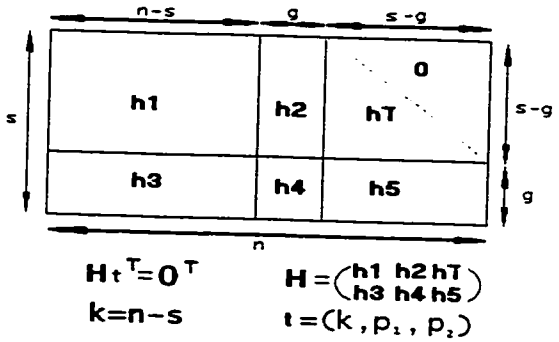


Fig.3.1 An approximate lower triangular form of parity-check matrix

More precisely, suppose the matrix is of the following form

$$H = \begin{pmatrix} h1 & h2 & hT \\ h3 & h4 & h5 \end{pmatrix} \tag{3.6}$$

Where **h1** is (s-g) rows by (n-s) columns, **h2** is (s-g) rows by g columns, **h3** is g rows by (n-s) columns , **h4** is g rows by g columns, **h5** is g rows by (s-g) columns, **hT** is (s-g) rows by (s-g) columns. Obviously, all submatrices should be sparse, and **hT** is lower triangular form with 1 elements in every position of the diagonal.

Premultiplying this matrix by

$$\begin{pmatrix} I & 0 \\ -h5*hT^{-1} & I \end{pmatrix} \tag{3.7}$$

The result is that

$$\begin{pmatrix} h1 & h2 & hT \\ -h5*hT^{-1}h1+h3 & -h5*hT^{-1}h2+h4 & 0 \end{pmatrix} \tag{3.8}$$

Let t=(**k**, $\mathbf{p}_1$, $\mathbf{p}_2$) where **k** denotes the systematic part (source information is random) , $\mathbf{p}_1$ and $\mathbf{p}_2$ combined denote the parity part , $\mathbf{p}_1$ has length g, and $\mathbf{p}_2$ has length (s-g). So, according to formula (3.5) , we can have formula (3.9)

$$\begin{pmatrix} h1 & h2 & hT \\ -h5*hT^{-1}h1+h3 & -h5*hT^{-1}h2+h4 & 0 \end{pmatrix} \begin{pmatrix} k^T \\ p_1{}^T \\ p_2{}^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.9}$$

Now, we can define **F=-h5*hT^{-1}h2+h4** and suppose for the moment that **F** is nonsingular. Then from (3.9) we conclude that

$$p_1{}^T = -F^{-1}(-h5*hT^{-1}h1+h3)k^T \tag{3.10}$$

The next task is to use formula (3.5), (3.9), and (3.10) to find codewords . Simply, we list these operation in table 3.1 and table 3.2.[10]

So far, according to a random **k** information sequences, we encode a codeword t=(**k**, $\mathbf{p}_1$, $\mathbf{p}_2$). Example 3.1 will show the encoding method in more details. Although the example is binary, the algorithm can also be used for low density check H matrices whose entries belong to a different field.

Table 3.1

Computation of $p_1^T = -F^{-1}(-h5 * hT^{-1}h1 + h3)k^T$

Operation	Comment
$h1\ k^T$	Multiplication by sparse matrix
$hT^{-1}[h1\ k^T]$	hT^{-1} multiplyed by $h1\ k^T$
$-h5\ [hT^{-1}h1 * k^T]$	Multiplication by sparse matrix
$h3 * k^T$	Multiplication by sparse matrix
$[-h5\ hT^{-1}h1\ k^T] + [h3\ k^T]$	Addition
$p_1^T = -F^{-1}[-h5\ hT^{-1}h1\ k^T + h3\ k^T]$	Multiplication by dense g×g matrix
$F = -h5 * hT^{-1}h2 + h4$	Definition

Table 3.2

Computation of $p_2^T = -hT^{-1}(h1k^T + h2p_1^T)$

Operation	Comment
$h2\ p_1^T$	Multiplication by sparse matrix
$[h1\ k^T] + [h2\ p_1^T]$	Addition
$-hT^{-1}[h1\ k^T + h2\ p_1^T]$	$-hT^{-1}[h1\ k^T + h2\ p_1^T] = y^T \Leftrightarrow -[h1\ k^T + h2\ p_1^T] = hTy^T$

Example 3.1: Suppose we have a **H** matrix:

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{3.11}$$

If we simply reorder the columns , we can get the following **H** matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

Considering (3.6) and (3.12) , we can divide the H matrix of (3.12) into six submatrixes.

$$h1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad h2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \quad hT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

$$h3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad h4 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad h5 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Now, let us produce an information sequence at random. Suppose the information sequence is (1,1,0,0,1,1) . Then we can get the codeword step by step as follows:

$$h1 * k^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$hT^{-1} * h1 * k^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

$$-h5*hT^{-1}*h1*k^T=\begin{pmatrix}0&0&1&1\\1&0&0&1\end{pmatrix}\begin{pmatrix}1\\0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}$$

$$-h5*hT^{-1}*h1*k^T+h3*k^T=\begin{pmatrix}0\\0\end{pmatrix}+\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$$

$$h3*k^T=\begin{pmatrix}0&1&0&1&0&0\\0&0&1&0&1&1\end{pmatrix}\begin{pmatrix}1\\1\\0\\0\\1\\1\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$$

$$p_1{}^T=\begin{pmatrix}1&0\\1&1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\1\end{pmatrix}$$

$$h2*p_1{}^T=\begin{pmatrix}1&0\\0&1\\1&0\\0&0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}1\\1\\1\\0\end{pmatrix}$$

$$h1*k^T+h2*p_1{}^T=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+\begin{pmatrix}1\\1\\1\\0\end{pmatrix}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$$

$$p_2{}^T=-hT^{-1}*\left(h1*k^T+h2*p_1{}^T\right)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\1&1&1&0\\1&0&1&1\end{pmatrix}\begin{pmatrix}0\\1\\1\\0\end{pmatrix}=\begin{pmatrix}0\\1\\0\\1\end{pmatrix}$$

Finally, we get the codeword **t** is ($\mathbf{k}$, $\mathbf{p_1}$, $\mathbf{p_2}$) = (1,1,0,0,1,1,1,1,0,1,0,1). If we use the equation $\mathbf{Ht^T=0^T}$ to check whether the codeword is right or wrong, we find that

the codeword is correct. Therefore, the encoding method based on approximate lower triangulations is valid for arbitrary **H** matrix of LDPC codes.

In summary, because the scheme is applied to encode LDPC codes without using Gaussian elimination, the complexity of calculation decreases. However, we use the systematic encoding scheme based on a generator matrix to encode the LDPC codes in the thesis because the H matrix size is not so big. The small H matrix size can be easily used to encode the LDPC codes and the complexity of calculation is not very high.

Chapter 4

Real-Output Channel

4.1 The error probability for binary modulation

First, let us consider binary signals whose waveforms are $s_1(t_t)=w(t_t)$ and $s_2(t_t)=-w(t_t)$.

Where $w(t_t)$ is a pulse whose definition is in formula (4.1)

$$w(t_t)=\begin{cases} m & 0 \le t_t \le T \\ 0 & elsewhere \end{cases} \tag{4.1}$$

$$\varepsilon_s = \int_0^T w^2(t_t)dt_t = m^2T \tag{4.2}$$

Where t_t is the time. These signals are called antipodal because $s_1(t_t)= -s_2(t_t)$. The energy in the waveform $w(t_t)$ is ε_s . Fig. 4.1 shows the two signal points after matched filtering.

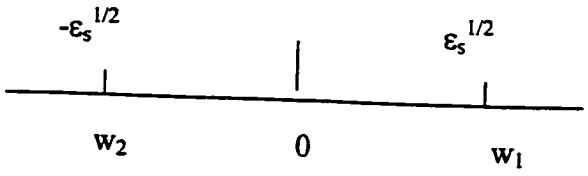


Fig.4.1 Signal points of binary antipodal signals

Suppose that the two signals are equally probable. When signals $s_1(t_t)$ or $s_2(t_t)$ are transmitted, the received signals from the demodulator are the following:

$$y = \varepsilon_s^{1/2} + \text{noise}$$
$$y = -\varepsilon_s^{1/2} + \text{noise} \qquad (4.3)$$

Where noise means the additive Gaussian noise component, which has zero mean and variance $\sigma^2 = \frac{1}{2}N_0$. Therefore, if y>0, it is assumed $w_1(t_t)$ was transmitted, and if y<0, that $w_2(t_t)$ was transmitted. Formula (4.4) and (4.5) show the two conditional PDF-s of the received signal.

$$p(y \mid w_1) = \frac{1}{\sqrt{\pi N_0}} e^{\frac{(y-\sqrt{\varepsilon_s})^2}{N_0}} \qquad (4.4)$$

$$p(y \mid w_2) = \frac{1}{\sqrt{\pi N_0}} e^{\frac{(y+\sqrt{\varepsilon_s})^2}{N_0}} \qquad (4.5)$$

The two conditional PDF-s are shown in Fig. 4.2

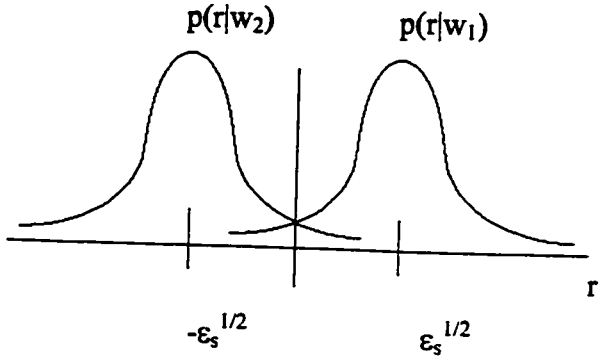


Fig. 4.2 Two conditional pdfs of received signals

Obviously, when $w_1(t_t)$ is transmitted, the probability of error is $P(error|w_1)$ during the period of $r<0$.(The detail is shown in formula 4.6)

$$P(error \mid w_1) = \int_{-\infty}^{0} p(y \mid w_1)dr = \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} e\left(-\frac{\left(y-\sqrt{\varepsilon_s}\right)^2}{N_0}\right)dr$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\sqrt{2\varepsilon_s / N_0}} e^{-x^2/2}dx = \frac{1}{\sqrt{2\pi}} \int_{\sqrt{2\varepsilon_s / N_0}}^{\infty} e^{-x^2/2}dx = Q(\sqrt{\frac{2\varepsilon_s}{N_0}}) \qquad (4.6)$$

Where Q(x) is the Q-function. Using the same idea, we also get $P(error|w_2)=Q(\sqrt{\frac{2\varepsilon_s}{N_0}})$.

Finally, the average probability of eeror can be calculated as follows:

$$P_{avarage} = \frac{1}{2}\left(P(error \mid w_1) + P(error \mid w_2)\right) = Q(\sqrt{\frac{2\varepsilon_s}{N_0}}) \qquad (4.7)$$

4.2 Mapping of bits to BPSK based on AWGN

As in Section 4.1, the binary input real output Gaussian channel is shown in Fig.4.3 again.

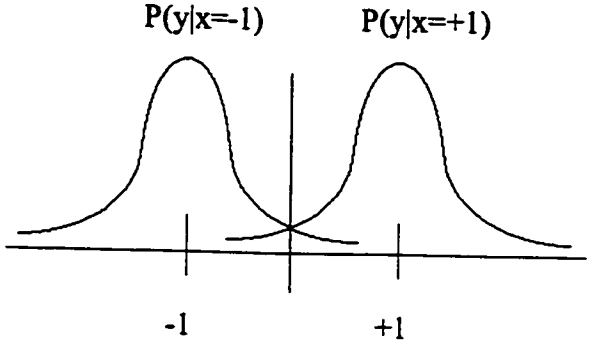


Fig. 4.3 Binary input real output Gaussian channel

Now, let us focus on the simple case of a channel with inputs of $t \in \{0,1\}$ and real valued outputs. Here, an arbitrarily selected received vector **r** includes an implicit noise vector **n**. Here **r** and **n** are vectors over GF(2) resulting from the decisions made in the receiver. That is if y>0 we decide r_i=1 and vice versa. For independent noise, the bits being 1 is determined by the likelihood ratio for the received signal. For a Gaussian channel with input of $x_{in} = \pm 1$, the received signal is $y_i = (2\ t - 1) + v_i$, where v_i is a zero-mean Gaussian noise of variance σ^2. The effective probability for a 1 in bit i of the noise vector **n** (based on **r** =0) means that we want to find out the probability that n_i=1 when r_i=0. Since $r_i = t_i \oplus n_i$, that is like saying we want to find the probability that t_i =1 given what we have received (y_i), i.e., $P(t_i = 1 \mid y_i)$, and y_i is the real output of the channel at time i.

Using:

$$P(y_i \mid t_i = 1)P(t_i = 1) + P(y_i \mid t_i = 0)P(t_i = 0) = P(y_i) \tag{4.8}$$

with
$$P(t_i = 1) = P(t_i = 0) = 1/2. \tag{4.9}$$

And:

$$P(y_i \mid t_i = 1) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-(y_i-1)^2}{2\sigma^2}\right)} \tag{4.10}$$

$$P(y_i \mid t_i = 0) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-(y_i+1)^2}{2\sigma^2}\right)} \tag{4.11}$$

$$\frac{P(y_i \mid t_i = 0)}{P(y_i \mid t_i = 1)} = e^{\left(\frac{-2y_i}{\sigma^2}\right)} \tag{4.12}$$

We have:

$$P(t_i = 1 \mid y_i) = f^1{}_{n(i)} = \frac{P(y_i \mid t_i = 1)P(t_i = 1)}{P(y_i)} \tag{4.13}$$

(using Bayes' theorem)

$$f^1{}_{n(i)} = \frac{\frac{1}{2}P(y_i \mid t_i = 1)}{\frac{1}{2}P(y_i \mid t_i = 1) + \frac{1}{2}P(y_i \mid t_i = 0)} \tag{4.14}$$

$f^1{}_{n(i)}\left[P(y_i \mid t_i = 1) + P(y_i \mid t_i = 0)\right] = P(y_i \mid t_i = 1)$ divided by $P(y_i \mid t_i = 1)$

$$f^1{}_{n(i)}\left[1 + \frac{P(y_i \mid t_i = 0)}{P(y_i \mid t_i = 1)}\right] = 1 \tag{4.15}$$

$$f^1{}_{n(i)} = \frac{1}{1 + e^{\left(\frac{-2y_i}{\sigma^2}\right)}} = P(t_i = 1 \mid y_i). \tag{4.16}$$

What happens if $\mathbf{r} \neq 0$?

If $r_i = 1$ ($r_i \in (0,1)$) $r_i = t_i \oplus n_i$ $r_i = 1 \Rightarrow$ $t_i = 0$ $n_i = 1$ or $t_i = 1$ $n_i = 0$ The effective probability for a 1 in bit i of the noise vector **n** (based on $\mathbf{r} \neq 0$) is then

$$f^{1}{}_{n(i)} = \frac{1}{1 + e^{\left(\frac{2y_i}{\sigma^2}\right)}} \qquad \{ t_i = 0 \; n_i = 1 \; \} \tag{4.17}$$

Since $r_i = t_i \oplus n_i$, that is like saying we want to find the probability that $t_i = 0$ given what we have received (y_i) $P(t_i = 0 \mid y_i) = f^{0}{}_{n(i)}$

Using:

$$P(y_i \mid t_i = 1)P(t_i = 1) + P(y_i \mid t_i = 0)P(t_i = 0) = P(y_i) \tag{4.18}$$

with $$P(t_i = 1) = P(t_i = 0) = 1/2 .$$

And:

$$P(y_i \mid t_i = 1) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-(y_i - 1)^2}{2\sigma^2}\right)} \tag{4.19}$$

$$P(y_i \mid t_i = 0) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-(y_i + 1)^2}{2\sigma^2}\right)} \tag{4.20}$$

$$\frac{P(y_i \mid t_i = 1)}{P(y_i \mid t_i = 0)} = e^{\left(\frac{-2y_i}{\sigma^2}\right)} \tag{4.21}$$

We have:

$$P(t_i = 0 \mid y_i) = f^{0}{}_{n(i)} = \frac{P(y_i \mid t_i = 0)P(t_i = 0)}{P(y_i)} \tag{4.22}$$

$$f^{0}{}_{n(i)} = \frac{1}{1 + e^{\left(\frac{2y_i}{\sigma^2}\right)}} \tag{4.23}$$

Using the same idea , we can derive $f^{1}_{n(i)} = \frac{1}{1+e^{\left(\frac{2y_i}{\sigma^2}\right)}}$ { $t_i=1$ $n_i=0$ } (4.24)

Finally, we always get , when $r_i=1$,

$$f^{1}_{n(i)} = \frac{1}{1+e^{\left(-\frac{2y_i}{\sigma^2}\right)}} \tag{4.25}$$

When $r_i=0$,

$$f^{1}_{n(i)} = \frac{1}{1+e^{\left(\frac{2y_i}{\sigma^2}\right)}} \tag{4.26}$$

Next, if the binary inputs are (+1, -1), the real output y has a conditional Gaussian distribution which has mean 1 and variance σ^2

$$F(y \mid 1) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{y} e^{-(y-1)^2/2\sigma^2} dy \tag{4.27}$$

In this thesis, we set the input signal amplitude to 1 and vary σ to correspond to the signal-to-noise ratio. For the uncoded channel, $E_s/N_0=E_b/N_0$, because there is one channel symbol per bit. However, for the coded channel,

$$E_s/N_0=(E_b/N_0)*(k/n)=(E_b/N_0)*R,$$
$$10\log_{10}Es/N_0=10\log_{10}E_b/N_0+10\log_{10}(k/n).$$

For example, for code rate = 1/2 ,

$$10\log_{10}E_s/N_0= 10\log_{10}E_b/N_0 +10\log_{10}(1/2)= 10\log_{10}E_b/N_0 - 3.01\text{dB}.$$

Similarly, for code rate = 2/3,

$$E_s/N_0=E_b/N_0+10\log_{10}(2/3)=E_b/N_0-1.76\text{dB}.$$

In the AWGN channel, the signal is corrupted by additive noise, which has the power spectrum watts/Hz. The variance σ^2 of the noise is equal to $N_0/2$. If we communicate using a code rate R, the signal to noise ratio can be expressed in formula (4.28) or (4.29).

$$10^{\frac{1}{10}\frac{E_s}{N_0}}=\frac{1}{2\sigma^2} \tag{4.28}$$

$$10^{\frac{1}{10}\frac{E_b}{N_0}}=\frac{1}{2R\sigma^2} \tag{4.29}$$

So we can simulate a Gaussian channel using the above conditions and formulas.

4.3 Gray mapping of bits to 8PSK on AWGN

For 8PSK modulation, the eight signal waveforms are shown in Fig.4.4. We note that these signal waveforms are using Gray mapping of bits to 8PSK and they have equal energy that is:

$$\varepsilon=\int_0^T s_m{}^2(t_t)dt_t=\frac{1}{2}\int_0^T w^2(t_t)dt_t=\frac{1}{2}\varepsilon_s \tag{4.30}$$

Where $w(t_t)$ is the signal pulse and the ε_s denotes the energy in the pulse $w(t_t)$.

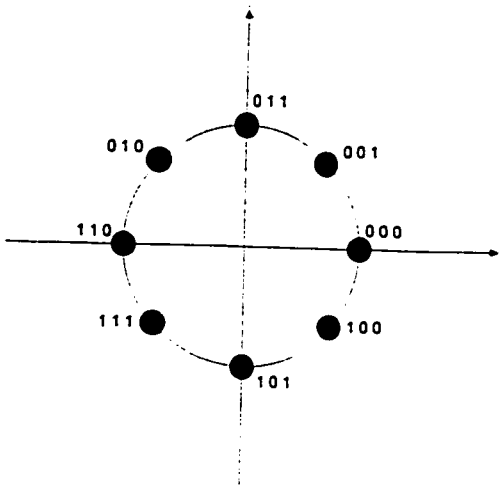


Fig. 4.4 Signal space constellation of Gray mapping for 8PSK signals

Assuming unit amplitude this mapping is given by:

$$
\begin{array}{llll}
a_0 \text{ --- } 0 & 0 & 0 \text{ --- } & 1+j0 \\
a_1 \text{ --- } 0 & 0 & 1 \text{ --- } & \frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2} \\
a_2 \text{ --- } 0 & 1 & 1 \text{ --- } & 0+j1 \\
a_3 \text{ --- } 0 & 1 & 0 \text{ --- } & -\frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2} \\
a_4 \text{ --- } 1 & 1 & 0 \text{ --- } & -1+j0 \\
a_5 \text{ --- } 1 & 1 & 1 \text{ --- } & -\frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2} \\
a_6 \text{ --- } 1 & 0 & 1 \text{ --- } & 0-j1 \\
a_7 \text{ --- } 1 & 0 & 0 \text{ --- } & \frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2} \\
: & : & : & \\
l_0 & l_1 & l_2 &
\end{array}
$$

Clearly, three bits will be mapped to a complex value after 8PSK modulation. Fig.4.5 describes the procedure in detail.

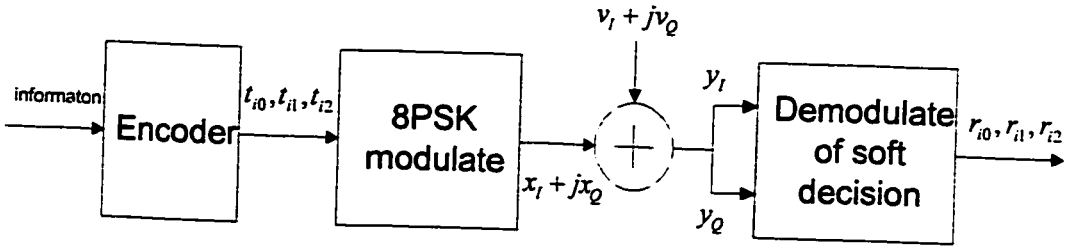


Fig. 4.5 Simplified model of a coded system using 8PSK modulation

From Fig.4.5, we can know information $\Rightarrow$ encoder $\Rightarrow (t_{i0}, t_{i1}, t_{i2}) \Rightarrow$ 8PSK modulate $\Rightarrow$ $x_I + jx_Q$ $\Rightarrow$ $y_I + jy_Q = x_I + jx_Q + v_I + jv_Q$ $\Rightarrow$ demodulation of soft decision $\Rightarrow$ r_{i0}, r_{i1}, r_{i2} .

$r_{ij} = t_{ij} \oplus n_{ij}$ $\quad j \in \{0,1,2\}$, i is the symbol sequence index

$$f^1{}_{n_{ij}} = p(n_{ij} = 1 \mid y_i) = p(t_{ij} \neq r_{ij} \mid y_i) \tag{4.31}$$

The $f^1{}_{n_{ij}}$ is the effective probability for a 1 in bit i of the noise vector $\mathbf{n}$. Now , let us analyze two different situations based on $r_{ij} = 0$ and $r_{ij} \neq 0$.

1) if $r_{ij} = 0$ $\Rightarrow$ $f^1{}_{n_{ij}} = p(t_{ij} = 1 \mid y_i)$ (4.32)

Firstly, we should define

$$l_{ij} = \Lambda(n_{ij}) = \ln \frac{p(t_{ij} = 1 \mid y_i)}{p(t_{ij} = 0 \mid y_i)} = \ln \frac{p(y_i \mid t_{ij} = 1)}{p(y_i \mid t_{ij} = 0)} \tag{4.33}$$

$$l_{ij} = \ln \frac{\sum_{t_{ij} \in s(1)} p(y_i \mid t_{ij})}{\sum_{t_{ij} \in s(0)} p(y_i \mid t_{ij})} \tag{4.34}$$

Where s(1) is set of element 1 in sequence i and position j, and s(0) is set of element 0 in sequence i and position j.

$$l_{i0} = \ln \frac{\sum_{t_{ij} \in a_4, a_5, a_6, a_7} p(y_i \mid t_{ij})}{\sum_{t_{ij} \in a_0, a_1, a_2, a_3} p(y_i \mid t_{ij})} = \ln \{ \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{4I})^2 + (y_Q - a_{4Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{5I})^2 + (y_Q - a_{5Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{6I})^2 + (y_I - a_{6Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{7Q})^2 + (y_Q - a_{7Q})^2}{2\sigma^2}]) / \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{0I})^2 + (y_I - a_{0Q})^2}{2\sigma^2}] \quad (4.35)$$
$$+ \exp[-\frac{(y_I - a_{1I})^2 + (y_Q - a_{1Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{2I})^2 + (y_I - a_{2Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{3I})^2 + (y_Q - a_{3Q})^2}{2\sigma^2}]\}$$

$$l_{i1} = \ln \frac{\sum_{t_{ij} \in a_2, a_3, a_4, a_5} p(y_i \mid t_{ij})}{\sum_{t_{ij} \in a_0, a_1, a_6, a_7} p(y_i \mid t_{ij})} = \ln \{ \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{2I})^2 + (y_Q - a_{2Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{3I})^2 + (y_Q - a_{3Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{4I})^2 + (y_I - a_{4Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{5Q})^2 + (y_Q - a_{5Q})^2}{2\sigma^2}]) / \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{0I})^2 + (y_I - a_{0Q})^2}{2\sigma^2}] \quad (4.36)$$
$$+ \exp[-\frac{(y_I - a_{1I})^2 + (y_Q - a_{1Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{6I})^2 + (y_I - a_{6Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{7I})^2 + (y_Q - a_{7Q})^2}{2\sigma^2}]\}$$

$$l_{i2} = \ln \frac{\sum_{t_{ij} \in a_1, a_2, a_5, a_6} p(y_i \mid t_{ij})}{\sum_{t_{ij} \in a_0, a_3, a_4, a_7} p(y_i \mid t_{ij})} = \ln \{ \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{1I})^2 + (y_Q - a_{1Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{2I})^2 + (y_Q - a_{2Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{5I})^2 + (y_I - a_{5Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{6Q})^2 + (y_Q - a_{6Q})^2}{2\sigma^2}]) / \frac{1}{2\pi\sigma^2} (\exp[-\frac{(y_I - a_{0I})^2 + (y_I - a_{0Q})^2}{2\sigma^2}] \quad (4.37)$$
$$+ \exp[-\frac{(y_I - a_{3I})^2 + (y_Q - a_{3Q})^2}{2\sigma^2}] + \exp[-\frac{(y_I - a_{4I})^2 + (y_I - a_{4Q})^2}{2\sigma^2}]$$
$$+ \exp[-\frac{(y_I - a_{7I})^2 + (y_Q - a_{7Q})^2}{2\sigma^2}]\}$$

from (4.32) and (4.33) we get that

$$l_{ij} = \ln \frac{f^1{}_{n_{ij}}}{1 - f^1{}_{n_{ij}}} \longrightarrow \frac{f^1{}_{n_{ij}}}{1 - f^1{}_{n_{ij}}} = e^{l_{ij}}$$

Then

$$f^1{}_{n_{ij}} = \frac{e^{l_{ij}}}{1 + e^{l_{ij}}} = \frac{1}{1 + e^{-l_{ij}}} \quad (4.38)$$

2) If $r_{ij} = 1$ $r_{ij} = t_{ij} \oplus n_{ij}$

$t_{ij} \oplus n_{ij}$ $\{t_{ij}=1 \ n_{ij}=0$ or $t_{ij}=0 \ n_{ij}=1\}$ $f^1{}_{n_{ij}} = p(t_{ij}=0|y_i)$

$$l_{ij} = \ln \frac{1 - f^1{}_{n_{ij}}}{f^1{}_{n_{ij}}} \longrightarrow \frac{1 - f^1{}_{n_{ij}}}{f^1{}_{n_{ij}}} = e^{l_{ij}}$$

$$f^1{}_{n_{ij}} = \frac{1}{1 + e^{l_{ij}}} \quad (4.39)$$

We also get the receive vector **r** from formula (4.35) (4.36) (4.37)

$$\begin{array}{llllll} \text{If} & l_{i0} > 0 & r_{i0} = 1 & \text{and} & l_{i0} <= 0 & r_{i0} = 0 \\ \text{Similarly} & l_{i1} > 0 & r_{i1} = 1 & \text{and} & l_{i1} <= 0 & r_{i1} = 0 \\ & l_{i2} > 0 & r_{i2} = 1 & \text{and} & l_{i2} <= 0 & r_{i2} = 0 \end{array} \qquad (4.40)$$

4.4 With an anti-Gray mapping of bits to 8PSK on AWGN

In Section 4.3, we have discussed Gray mapping of bits to 8PSK on AWGN. In the section, we will analyse an anti-Gray mapping (The definition can be found in [19], i.e., the Euclidean distance between adjacent signal points is either 2 or 3) of bits to 8PSK depending on AWGN. Fig.4.6 shows the signal constellation.

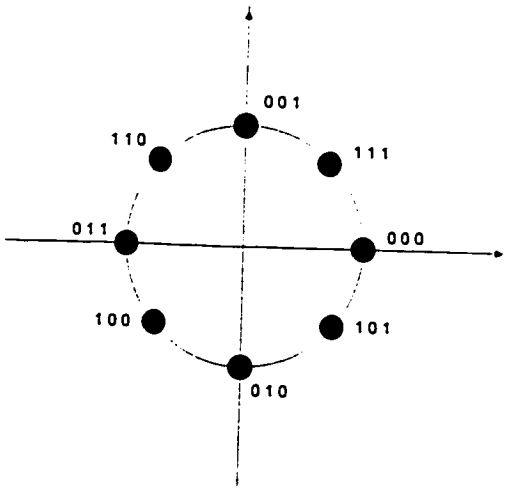


Fig. 4.6 Signal space constellation of anti Gray mapping for 8PSK signals

From the anti Gray mapping, we find that the weight between adjacent signal points is either 2 or 3. Assuming unit symbol energy this mapping is given by:

$$
\begin{array}{lccc}
a_0 \text{ --- } 0 & 0 & 0 \text{ --- } & 1+j0 \\
a_1 \text{ --- } 1 & 1 & 1 \text{ --- } & \frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2} \\
a_2 \text{ --- } 0 & 0 & 1 \text{ --- } & 0+j1 \\
a_3 \text{ --- } 1 & 1 & 0 \text{ --- } & -\frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2} \\
a_4 \text{ --- } 0 & 1 & 1 \text{ --- } & -1+j0 \\
a_5 \text{ --- } 1 & 0 & 0 \text{ --- } & -\frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2} \\
a_6 \text{ --- } 0 & 1 & 0 \text{ --- } & 0-j1 \\
a_7 \text{ --- } 1 & 0 & 1 \text{ --- } & \frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2} \\
: & : & : & \\
l_0 & l_1 & l_2 &
\end{array}
$$

We can also apply formulas (4.31) – (4.40) to this case.

Chapter 5

The Decoding Problem for LDPC Codes

5.1 Linear block codes

In general, linear block codes are defined and described in terms of generator and parity-check matrices. The encoding of an (n, k) linear block code changes a sequence of k information symbols into a longer sequence of n symbols which is called a codeword. A binary linear block code has three properties. The first property is that component modulo-2 sum of two codewords is another codeword. The second property is that the code contains the all-zero sequence. The third property of a linear block code is that a codeword only depends on the current input information sequence and not on the past information. In other words, the encoder does not have memory. More generally, not only can the information consist of nonbinary symbols, but also the codewords can be sequences of nonbinary symbols. But, in this thesis, only binary symbols are considered.

In an (n,k) linear block code, the code rate is $R=k/n$. In total, there are 2^k different possible information inputs. We get 2^k different codewords because of the one to one relationship between the possible information sequences and codewords.

In a linear block code, if we have a set of k linearly independent binary n-tuples $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \ldots\ldots \mathbf{g}_k$, The codeword for an information sequence $\mathbf{k} = (k_1, k_2, \ldots\ldots k_k)$ can be represented as linear combination as follows:

$$\mathbf{t} = k_1\mathbf{g}_1 + k_2\mathbf{g}_2 + k_3\mathbf{g}_3 + \ldots\ldots + k_k\mathbf{g}_k \tag{5.1}$$

where $\mathbf{t}$ is a codeword , $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \ldots\ldots \mathbf{g}_k$ can also be written as a k×n matrix :

$$G = \begin{pmatrix} G_1 \\ G_2 \\ : \\ : \\ G_k \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & \ldots\ldots & g_{1k} \\ g_{21} & g_{22} & \ldots\ldots & g_{2k} \\ : & : & : & : \\ : & : & : & : \\ g_{k1} & g_{k2} & \ldots\ldots & g_{nk} \end{pmatrix} \tag{5.2}$$

where $\mathbf{G}$ is called the generator matrix of the code. If the generator matrix is written as follows, the linear block code with this structure is referred to as a linear systematic block code.

$$G = \begin{pmatrix} G_1 \\ G_2 \\ : \\ : \\ G_k \end{pmatrix} = \begin{pmatrix} P & I_k \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & \ldots\ldots & p_{1n-k} & 1 & 0 & \ldots\ldots & 0 \\ p_{21} & p_{22} & \ldots\ldots & p_{2n-k} & 0 & 1 & \ldots\ldots & 0 \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ p_{k1} & p_{k2} & \ldots\ldots & p_{kn-k} & 0 & 0 & \ldots\ldots & 1 \end{pmatrix} \tag{5.3}$$

where $\mathbf{P}$ is k ×n-k matrix, and $\mathbf{I}_k$ is a k×k identity matrix.

A code can also be defined by a parity check matrix. An n-tuple $\mathbf{t}$ is a code word in the code generated by $\mathbf{G}$ if and only if $\mathbf{t} \cdot \mathbf{H}^T = \mathbf{0}$. The matrix $\mathbf{H}$ is called a parity check matrix of the code. Based on this idea, we describe the basic concept of decoding LDPC codes in Section 5.2

5.2 Basic concept of decoding LDPC codes

Equation 5.3 can also be written as follows:

$$G^T = \begin{pmatrix} I_k \\ P \end{pmatrix} \tag{5.4}$$

Where $\mathbf{I}_k$ is a k×k identity matrix and **P** is a binary matrix. Equation (5.4) is another form of equation (5.3), i.e. transpose of the generator matrix. Therefore, formula (3.3) is more suitable to encode LDPC codes

$$\mathbf{t} = \mathbf{G}^T \mathbf{k} = \begin{bmatrix} I_K \\ H_2^{-1} H_1 \end{bmatrix} \mathbf{k} \tag{5.5}$$

In Equation (5.5) means that $\begin{bmatrix} I_K \\ H_2^{-1} H_1 \end{bmatrix}$ is in systematic form and a source vector **k** of length **k** is encoded into a transmitted vector **t** defined by $\mathbf{t} = \mathbf{G}^T \mathbf{k}$ mod 2. The channel introduces noise and we receive the vector $\mathbf{r} = (\mathbf{G}^T \mathbf{k} + \mathbf{n})$ mod 2, where **n** is the noise vector. Of course, the receiver does not know either **t** or **n** . For the received **r**, the decoder must first detect whether **r** includes transmission errors or not. If the errors are found, the decoder will either mark the errors and correct them or request for a retransmission of **t**.

In the case of a binary-symmetric channel, **n** is assumed to be a sparse random vector with independent and identically distributed bits of density f_n, and **r** will be corresponding to the sign of the real output **y**. We declare the received bit r=1 if y>0 and r=0 if y<0.

When **r** is received, the syndrome of **r** can be calculated by decoder as follows

$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T \bmod 2 \tag{5.6}$$

Clearly, the syndrome is zero if and only if **r** is a code word, and the syndrome is not zero if and only if **r** is not a code word. As a result, when the syndrome is not zero, we can understand that the errors in the **r** have been detected and **r** is not a codeword. Otherwise, if the syndrome is zero, **r** is a code word and the receiver accepts **r** as the transmitted code word . However, we should pay attention that , **r** may contain errors, even if $\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{0}$. This is possible because the errors in some error vectors are not detectable. In this situation, the error patterns are called undetectable error patterns. Naturally, when an undetectable error pattern occurs, the decoder makes a decoding error. But, the probability of an undetected error for a binary symmetric channel (BSC) is derived [20] and shows that the error probability can be very small.

In fact, the syndrome computed from the received vector **r** only depends on the noise vector **n**, and does not have any relationship with the transmitted code word **t**. It is clear that **r** is the vector sum of **t** and **n** . From Equation (5.6), we can derive the new Equation (5.7)

$$\begin{aligned} \mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T \bmod 2 &= (\mathbf{t} + \mathbf{n}) \cdot \mathbf{H}^T \bmod 2 \\ &= (\mathbf{t} \cdot \mathbf{H}^T + \mathbf{n} \cdot \mathbf{H}^T) \bmod 2 \end{aligned} \tag{5.7}$$

Where the $\mathbf{t} \cdot \mathbf{H}^T \bmod 2 = 0$. As a result, the relationship between syndrome and noise vector can be expressed as follows:

$$\mathbf{S} = \mathbf{n} \cdot \mathbf{H}^T \bmod 2 \tag{5.8}$$

For LDPC codes , we must first get **H** matrix according to Section 2.4. The **H** matrix is a parity-check matrix for **G** i.e. $\mathbf{H} \cdot \mathbf{G}^T = \mathbf{0} \bmod 2$. As a result, the decoding problem is to discover **t** by finding the most likely **n** that satisfies the equation

$$\mathbf{n} \cdot \mathbf{H}^T = \mathbf{z} \bmod 2$$

Where **z** is the syndrome vector $\mathbf{z} \equiv \mathbf{r} \cdot \mathbf{H}^T = \mathbf{G}^T \cdot \mathbf{k} \cdot \mathbf{H}^T + \mathbf{n} \cdot \mathbf{H}^T = \mathbf{n} \cdot \mathbf{H}^T$.

When we perform the syndrome decoding in LDPC codes, we find the most probable noise vector, $\mathbf{x}_n$, that explains the observed syndrome vector

$$\mathbf{n} \cdot \mathbf{H}^T = \mathbf{x}_n \cdot \mathbf{H}^T = \mathbf{z} \bmod 2$$

In other words, $\mathbf{x}_n$ is our estimate of the noise vector. From $\mathbf{x}_n$, we can get our estimate of the transmitted signal vector (codeword).

$$\mathbf{t}' = (\mathbf{r} + \mathbf{x}_n) \bmod 2$$

Example 5.1. In example3.2, Suppose we have received three sequences $\mathbf{r}_1$=(1,0,0,0,0,1,0,0,1,0,0,0), $\mathbf{r}_2$=(0,1,0,0,1,0,1,0,1,0,1,0) , $\mathbf{r}_3$=(0,0,1,1,0,0,1,0,1,1,1,1). What are the noise vectors ?

Answer:

$$H^T = [H_1 \quad H_2]^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}^T$$

$$\mathbf{S}_1 = \mathbf{r}_1 \cdot \mathbf{H}^T \bmod 2 \,, \quad \mathbf{S}_2 = \mathbf{r}_2 \cdot \mathbf{H}^T \bmod 2,$$

$$\mathbf{S}_3 = \mathbf{r}_3 \cdot \mathbf{H}^T \bmod 2$$

Therefore, $\mathbf{S}_1$ = (1,1,0,1,0,0), $\mathbf{S}_2$ = (1,1,0,0,1,0), $\mathbf{S}_3$ = (1,1,0,0,0,1). The three codewords being considered are $\mathbf{t}_1$=(0,0,0,0,0,1,0,0,1,0,0,0), $\mathbf{t}_2$=

(0,0,0,0,1,0,1,0,1,0,1,0) , $\mathbf{t}_3$=(0,0,0,1,0,0,1,0,1,1,1,1) in example 3.2. It is very easy for us to find the three noise vectors are as follows:

$$\mathbf{n}_1 = (\mathbf{r}_1+\mathbf{t}_1) \bmod 2 = (1,0,0,0,0,0,0,0,0,0,0,0)$$
$$\mathbf{n}_2 = (\mathbf{r}_2+\mathbf{t}_2) \bmod 2 = (0,1,0,0,0,0,0,0,0,0,0,0)$$
$$\mathbf{n}_2 = (\mathbf{r}_3+\mathbf{t}_3) \bmod 2 = (0,0,1,0,0,0,0,0,0,0,0,0)$$

However, when we solve a practical decoding problem, it is not so easy to find the noise vector . Therefore, our goal is to estimate the noise vectors using some algorithm. The next few sections will address this problems.

5.3. The sum-product algorithm

When the block length of a block code becomes large, for example LDPC code or RM block codes, maximum-likelihood decoding often becomes difficult . What is called belief propagation in the artificial intelligence community, which is also called sum-product decoding, can be thought of as using Bayes's rule locally and iteratively to calculate approximate marginal a posteriori probabilities for the codes. Because the decoding complexity of this approach per iteration is linear in block codes size, the sum product algorithm is practical. If a bipartite graph corresponding to an **H** matrix is no cycle, then it can be easily proved that the sum-product algorithm calculates marginal posterior probabilities exactly.

When we review the bipartite graph of Fig.2.4 (b), there are many short cycles which exist in the graph. In this situation, we still want to run the sum product algorithm because we want to compare the performance of graphs of a fixed size with and without short cycles. Our goal is to find the effect of the short cycles on the sum product algorithm. Although there is not enough theoretical understanding about the turbo codes and low density parity check codes, the two types of codes provide

excellent performance, and their huge success has ignited further research in the area.

In Chapter 2, we described an LDPC code as a linear block code specified by a very sparse check matrix. As a linear block code, an LDPC code can be represented by a bipartite graph. Besides, if we assume the low density parity check **H** matrix has N columns and M rows , then the designed code rate is as follows:

$$R = \frac{N-M}{N} = 1 - \frac{M}{N} = 1 - \frac{c}{rw} \tag{5.9}$$

Based on Fig.2.4 (b) and sum product algorithm, we can designate the set (i.e. a group) of bits n that takes part in check m by A(m) = { n : H_{mn}=1 }. Similarly, we can also designate the set of checks that bit n takes part in as B(n) ={m : H_{mn}=1}. As a result , Fig.2.4 (b) , the bipartite graph corresponding to the **H** matrix in Fig. 2.4 (a) can be divided into two parts . The sum product algorithm is an example of an algorithm that works by passing messages between the nodes of the graph representing the code. The first part is that we only consider message passing from bit nodes to check nodes. The second part is that we only analyse message passing from check nodes to bit nodes. The details are shown in Fig5.1 and Fig 5.2 .

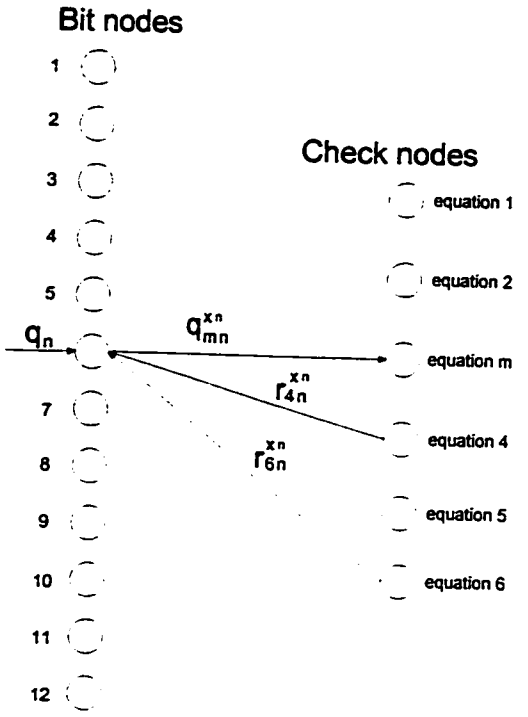


Fig.5.1 Message passing from bit node n to check node m in LDPC codes

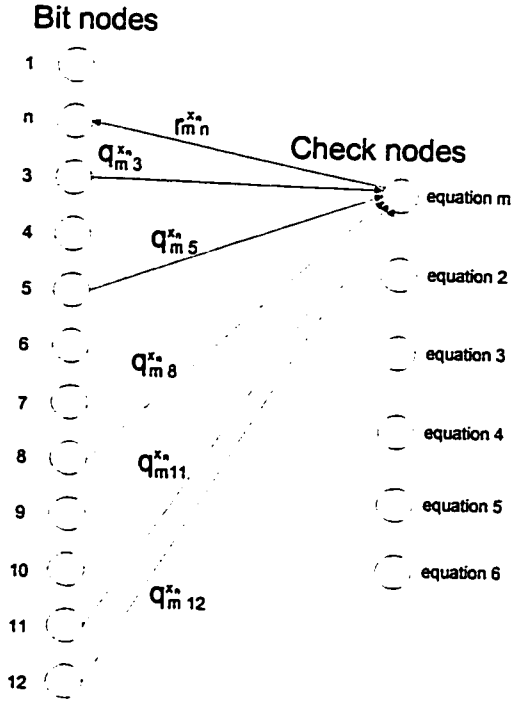


Fig.5.2 Message passing from check node n to bit node m in LDPC codes

In Fig. 5.1 and 5.2, we have described how the message passes from bit nodes to check nodes and from check nodes to bit nodes for each edge in bipartite graph. The decoding algorithm iteratively will update two kinds of log a posteriori probability ratio messages, q and r. The quantity q_n is the message coming from the channel. The quantity $q_{mn}^{x_n}$ is the message passed from the bit node to the check node along the edge between connecting points, and the $q_{mn}^{x_n}$ can be represented in the equation (5.10)

$$q_{mn}^{x_n} = \log \frac{p(x_n = 1 \mid u)}{p(x_n = 0 \mid u)} \tag{5.10}$$

Where x_n expresses the value of the bit node, and u expresses all messages coming from the edge connected to the bit node and the channel, except the edge being considered. The quantity $r_{mn}^{x_n}$ expresses the message passed from the check node to the bit node along the edge between connecting points , and the $r_{mn}^{x_n}$ can also be represented in the equation (5.11)

$$r_{mn}^{x_n} = \log \frac{p(x_n = 1 \mid v)}{p(x_n = 0 \mid v)} \tag{5.11}$$

Where x_n has the same meaning as equation (5.10) , and v represents all messages coming from the edge connected to the check node, other than the edge being considered. When we use the sum product algorithm to decode LDPC codes, it is very important for us to pay attention to the fact that the incoming message along the edge can not be considered in determining the outgoing message along the edge where the message is updated. Obviously, only the extrinsic message is calculated. Based on this point, this should be similar to turbo decoding [21].

5.3.1 Initialization

In the Fig. 5.1 and 5.2, the bit nodes are first used to accept information from the channel. Therefore, we set the prior probability of x_n =1 to be is P(x_n =1). Naturally, P(x_n =0) =1- P(x_n =1). When we consider the LDPC codes using a binary symmetric channel, and assume that the channel is a binary input Gaussian channel with a real output, P(x_n =1) can be expressed as follows: (details are in section 4.2)

$$P(x_n=1) = f^1_{n(i)} = \frac{1}{1+e^{\left(\frac{2y_i}{\sigma^2}\right)}} \tag{5.12}$$

The formula (5.12) means that P(x_n =1) can be initialized to the appropriate normalized likelihood. In other words, if there is H_{mn}=1 for every edge from n to m or from m to n, the quantity $q_{mn}^{x_n}$ is initialized to P(x_n). If x_n=1, $q_{mn}^{x_n=1}$ is initialized to P(x_n =1). If x_n=0, $q_{mn}^{x_n=0}$ is initialized to P(x_n =0).

5.3.2 Row calculation

In the row calculation of the sum product algorithm, the information $r_{mn}^{x_n}$ that check m delivers to bit n should be the probability of check m being satisfied if the bit in question has x_n=1 or x_n=0. Therefore, when we calculate $r_{mn}^{x_n=0}$ which is the probability of the syndrome $\mathbf{x_n} \bullet \mathbf{H}^T = \mathbf{z} \mod 2$ appearing when x_n=0, and we know the other bits $\{x_{n'} : n' \neq n\}$ which are corresponding to the probabilities $\{q_{mn}^0, q_{mn}^1\}$, the $r_{mn}^{x_n=0}$ can be defined as follows:

$$r_{mn}^{x_n=0} = \sum_{[x_{n'}: n' \in A(m) \setminus n]} P(z_m \mid x_n = 0, \{x_{n'} : n' \in A(m) \setminus n\}) \times \prod_{n' \in A(m) \setminus n} q_{mn'}^{x_{n'}} \qquad (5.13)$$

Where z_m is syndrome which is observed from check node m corresponding to x_n=0 and check equation m. Similarly, when x_n=1 , the corresponding $r_{mn}^{x_n=1}$ can be expressed in equation (5.14)

$$r_{mn}^{x_n=1} = \sum_{[x_{n'}: n' \in A(m) \setminus n]} P(z_m \mid x_n = 1, \{x_{n'} : n' \in A(m) \setminus n\}) \times \prod_{n' \in A(m) \setminus n} q_{mn'}^{x_{n'}} \qquad (5.14)$$

Where z_m is syndrome which is observed from m check node corresponding to x_n=1 and check equation m.

In formula (5.13) and (5.14) , $n' \in A(m) \setminus n$ has the same meaning. This indicates that n' belongs to the set A(m), but the n is not considered. The $x_{n'}$ still has its original value (1 or 0) and its value is not constrained by the formula. The summation of these conditional probabilities are either 1 or 0 which depends on whether the viewed syndrome z_m matches the noise vector $\mathbf{x}_n$ $\{x_1, x_2, \ldots. x_n, \ldots. x_N\}$. References [22][23] also provides some forward-backward algorithms to calculate the probabilities for syndrome z_m having its observed value as either x_n =1 or x_n =0.

Using the forward and backward method, a very convenient way of using the product of differences $\Delta q_{mn} = q_{mn}^{x_n=0} - q_{mn}^{x_n=1}$ is calculated.

Now, suppose we can get $\Delta r_{mn} = r_{mn}^{x_n=0} - r_{mn}^{x_n=1}$ from Δq_{mn}. If this is true, we can get following equations:

Since $r_{mn}^{x_n=0} + r_{mn}^{x_n=1} = 1$

$$r_{mn}^{x_n=0} = (1 + \Delta r_{mn})/2 \qquad (5.15)$$

$$r_{mn}^{x_n=1} = (1 - \Delta r_{mn})/2 \tag{5.16}$$

The next step is to find Δr_{mn}. In order to get the Δr_{mn}, we will use a simple iterative method. Firstly, we assume that the $\beta = x_i + x_j \mod 2$, and x_i and x_j have probabilities q_i^0, q_j^0 and q_i^1, q_j^1 of being 0 or 1, then

$$P(\beta=0) = q_i^0 q_j^0 + q_i^1 q_j^1 \tag{5.17}$$

$$P(\beta=1) = q_i^1 q_j^0 + q_i^0 q_j^1 \tag{5.18}$$

Hence

$$P(\beta=0) - P(\beta=1) = q_i^0 q_j^0 + q_i^1 q_j^1 - q_i^1 q_j^0 - q_i^0 q_j^1$$

$$= (q_i^0 - q_i^1)(q_j^0 - q_j^1) \tag{5.19}$$

Based on the (5.13), (5.14), and (5.19), we can obtain the equation (5.20)

$$\Delta r_{mn} = r_{mn}^{x_n=0} - r_{mn}^{x_n=1} = (-1)^{z_m} \prod_{n' \in A(m) \setminus n} \Delta q_{mn'} \tag{5.20}$$

5.3.3 Column calculation

In Section 5.3.2 section, we have calculated the values of $r_{mn}^{x_n=0}$ and $r_{mn}^{x_n=1}$. The next step of column calculation is to update the values of the probabilities $q_{mn}^{x_n=0}$ and $q_{mn}^{x_n=1}$ according to the $r_{mn}^{x_n=0}$ and $r_{mn}^{x_n=1}$. For each bit node n, we can get the following equations:

$$q_{mn}^{x_n=0} = \Phi_{mn} P(x_n = 0) \prod_{m' \in B(n) \setminus m} r_{m'n}^{x_n=0} \tag{5.21}$$

$$q_{mn}^{x_n=1} = \Phi_{mn} P(x_n = 1) \prod_{m' \in B(n) \setminus m} r_{m'n}^{x_n=1} \tag{5.22}$$

In Equation (5.21) and (5.22) , $m' \in B(n) \setminus m$ has the same meaning. This indicates that m' belongs to the set B(n), but m is not considered. Φ_{mn} is a coefficient and it satisfies the following condition of Equation (5.23)

$$q_{mn}^{x_n=0} + q_{mn}^{x_n=1} = 1 \tag{5.23}$$

Clearly, depending on Equations (5.21) and (5.22) , we can efficiently calculate these products in an upward pass and downward pass.

5.3.4 Tentative decoding

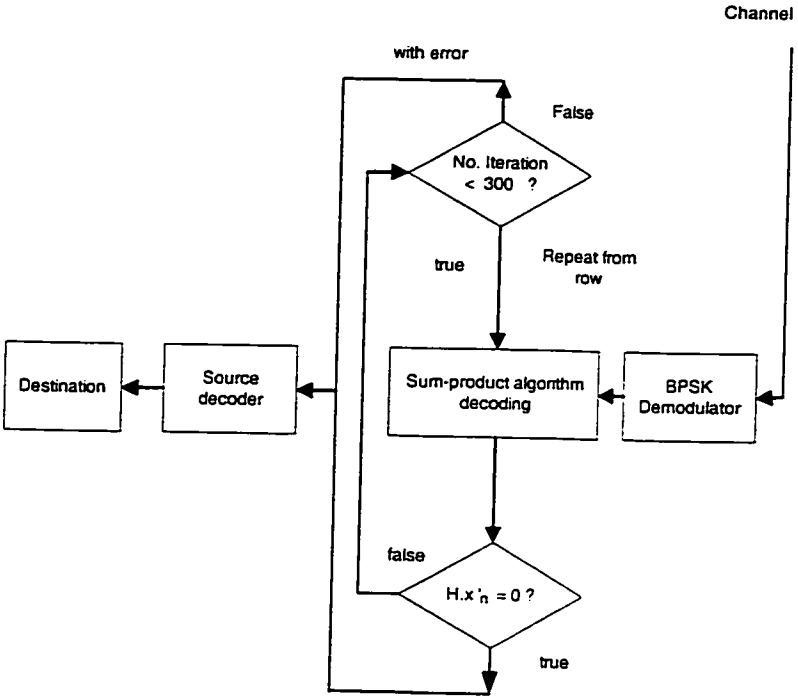
After the iteration, we have obtained updating information of $q_{mn}^{x_n=0}$, $q_{mn}^{x_n=1}$, $r_{mn}^{x_n=0}$, and $r_{mn}^{x_n=1}$. Therefore, we can also calculate the probabilities $q_n^{x_n=0}$, $q_n^{x_n=1}$ during the iteration. The formula is shown in equations (5.24) and (5.25)

$$q_n^{x_n=0} = \Phi_n P(x_n = 0) \prod_{m \in B(n)} r_{mn}^{x_n=0} \tag{5.24}$$

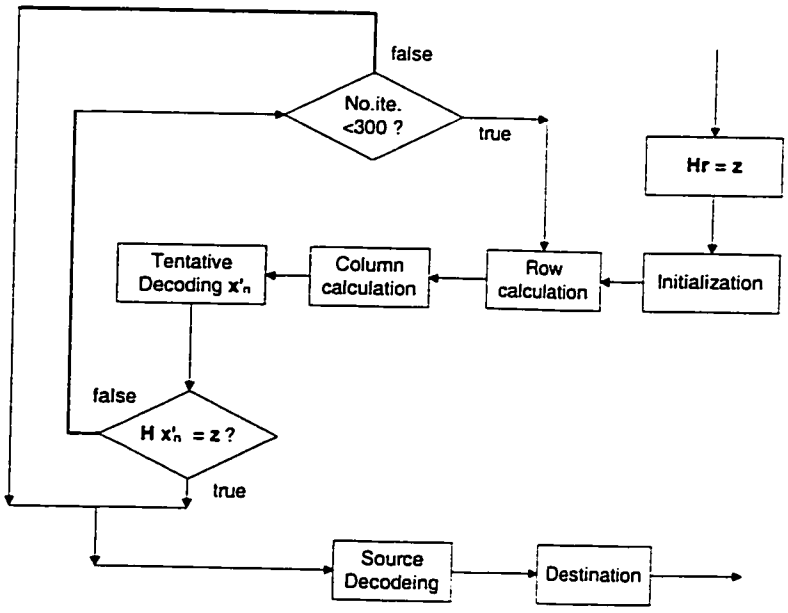
$$q_n^{x_n=1} = \Phi_n P(x_n = 1) \prod_{m \in B(n)} r_{mn}^{x_n=1} \tag{5.25}$$

Where Φ_n is a coefficient and the following condition (5.26) is satisfied.

$$q_n^{x_n=0} + q_n^{x_n=1} = 1 \tag{5.26}$$



(a)



(b)

Fig.5.3 The block diagram of sum product algorithm decoding LDPC codes

These probabilities of $q_n^{x_n=0}$, $q_n^{x_n=1}$ are used to produce a tentative decoding $\mathbf{x}_n$'. The decoding method is to set $x'_n=1$ if $q_n^{x_n=1}>0.5$. If the vector $\mathbf{x}_n$' satisfies the syndrome equation ,

$$\mathbf{x}_n' \bullet \mathbf{H}^T = \mathbf{n} \bullet \mathbf{H}^T = \mathbf{z} \mod 2$$

. we stop the decoding algorithm and declare a success. Otherwise, the iteration will continue to update $q_{mn}^{x_n=0}$, $q_{mn}^{x_n=1}$, $r_{mn}^{x_n=0}$, and $r_{mn}^{x_n=1}$ until the decoding is successful. If the maximum number of iteration , for example , three hundred, has occurred and the successful decoding is not found, then we declare a failure and the iteration is stopped. Another new message sequence will enter the encoder, modulator, AWGN channel, demodulator, and decoder. Fig 5.3 gives an outline of the sum product algorithm.

Chapter 6

Simulation Results Based on Five Different H Matrices

6.1 Comparison of different **H** matrices

Codes (of the same rate and size), but defined by different **H** matrices, may produce differing simulation results. However, in this chapter, we only list the simulation results based on **W1, W2, W3, W4, W5 H** matrices. The **H** matrices produced by **W1, W2** include short cycles of length 4 in their bipartite graph. When the two **H** matrices are used in encoding and decoding of LDPC codes, the simulation results are the similar. The **H** matrices constructed by **W3, W4, W5** do not have short cycles of length 4 in their bipartite graph. As a result, the performances of LDPC code corresponding to these H matrices for the BPSK and 8PSK in AWGN channel are better than those related to **W1, W2.** The three **H** matrices constructed by **W3, W4, W5** are similar to MacKay's H matrices of construction 1A. However, there is little difference between the two construction of **H** matrices. In MacKay's H matrices of construction 1A, all row weights are not guaranteed to be equal. In other words, some rows have less weight than other rows. But, every row weight is equal in the H matrices of constructions **W3, W4, W5.** The

LDPC codes based on these **H** matrices are regular which corresponds to Gallager's construction of **H** matrices. The difference is that Gallager's construction of **H** matrices has high complexity of computation and the three **H** matrices constructed by **W3, W4, W5** are easily realized by C program. Although **H** matrices constructed by **W3, W4, W5** are different, they keep the basic property whose every column has the same weight and every row also has the same weight. The simulation result shows that performance of codes obtained employing the three **H** matrices. The following sections describe performance comparisons. Besides, for simple reason, all M by 2M **H** matrices will be defined by M*2M in the following sections.

6.2 Performance comparison for mapping of bits to BPSK for AWGN

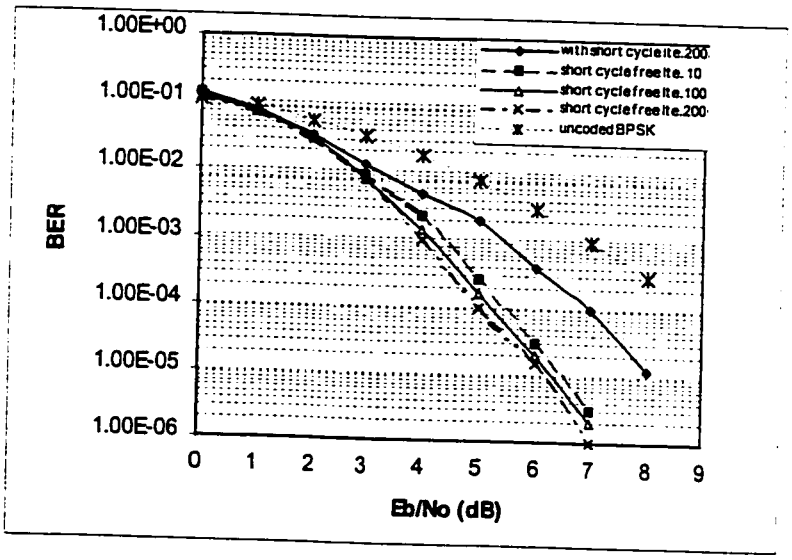


Fig.6.1 Performance comparison of LDPC code of 30*60 H matrix based on BPSK in AWGN channel

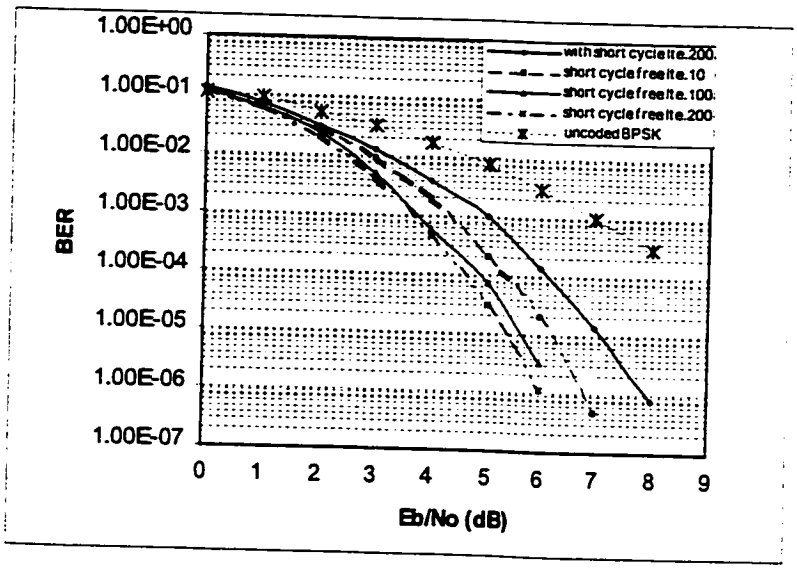


Fig.6.2 Performance comparison of LDPC code of 48*96
H matrix based on BPSK in AWGN channel

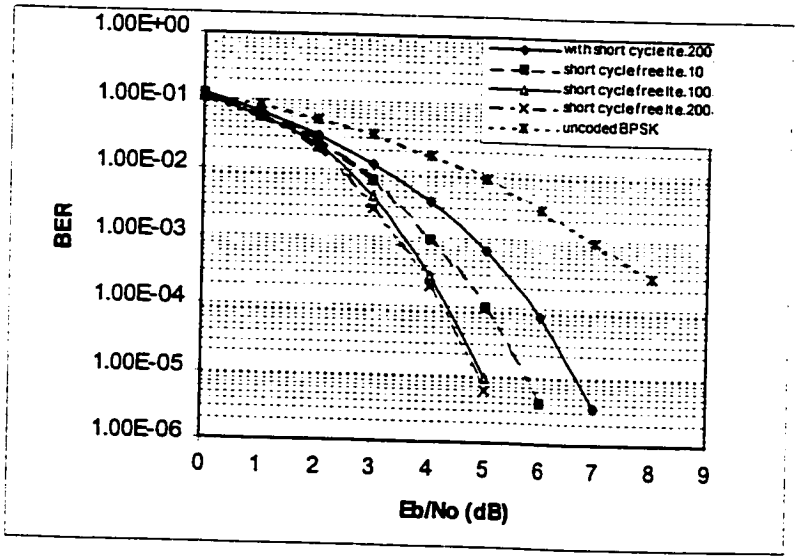


Fig.6.3 Performance comparison of LDPC code of 60*120
H matrix based on BPSK in AWGN channel

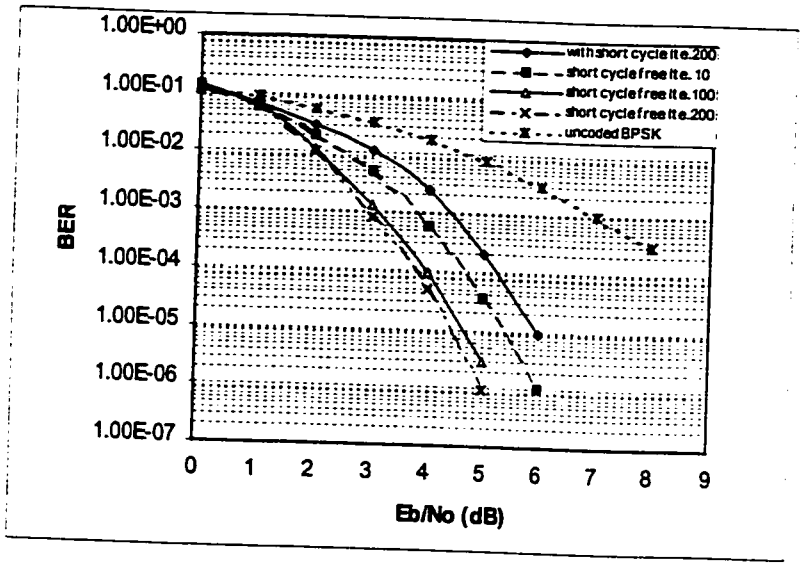


Fig.6.4 Performance comparison of LDPC code of 96*192 H matrix based on BPSK in AWGN channel

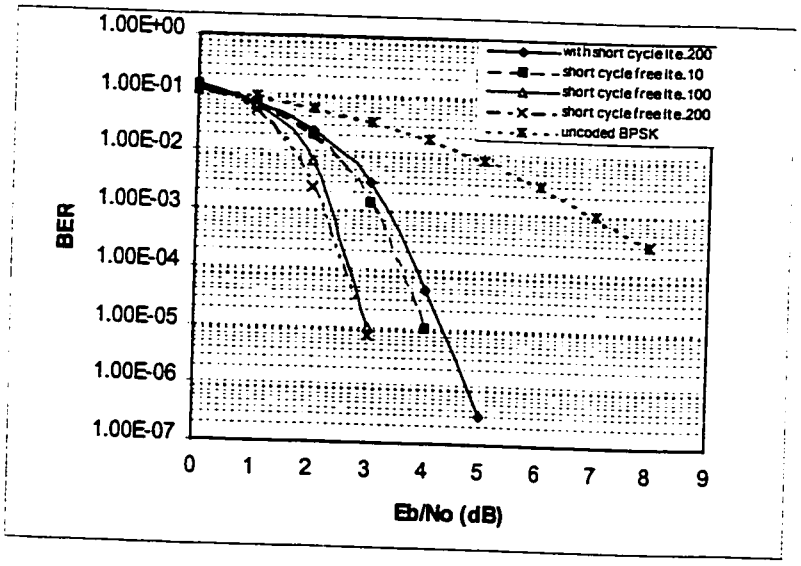


Fig.6.5 Performance comparison of LDPC code of 252*504 H matrix based on BPSK in AWGN channel

Figures.6.1, 6.2, 6.3, 6.4, and 6.5 show the performance comparison of LDPC codes of five different H matrices for BPSK in AWGN channel. Cycle free indicates

a construction that doesn't have cycles of length 4. According to the figures, we list some results in table 6.1. From the table 6.1 and also bar chart Fig.6.6, we find that, with the increasing number of iterations, the bit errors will decrease. When the number of iterations is 200, the convergence basically is stopped. Furthermore, we also note that the size of the low density parity check **H** matrix affects the bit errors probabilities. The bigger the size of low density parity check **H** matrix is used in encoding and decoding of LDPC codes, the better the BER performance will be.

TABLE 6.1

Comparison of E_b/N_0 (dB) at BER 10^{-4} about LDPC Codes based on BPSK and AWGN channel

Number of Ite.\H matrix size	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	7.1	6.2	5.8	5.3	3.8
Free cycle Ite. 10	5.45	5.3	5	4.6	3.6
Free cycle Ite. 100	5.3	4.9	4.3	4	2.8
Free cycle Ite. 200	5	4.7	4.2	3.8	2.65

TABLE 6.2

Gain in Performance by eliminating for cycles, difference E_b/N_0 (dB) at BER 10^{-4} about LDPC Codes based on BPSK and AWGN channel

Number of Ite.\H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	0	0	0	0	0
Free cycle Ite. 10	1.65	0.9	0.8	0.7	0.2
Free cycle Ite. 100	1.8	1.3	1.5	1.3	1
Free cycle Ite. 200	2.1	1.5	1.6	1.5	1.15

note* the value of table 6.2 is equal to that a **E_b/N_0 (dB)** - E_b/N_0 (dB) of cycle Ite. 200 in same **H** matrix

After analyzing Table 6.2 and also from bar chart of Fig. 6.7, we see that the existence of short cycles in bipartite graph seriously affects the performance of LDPC codes based on BPSK and an AWGN channel. Firstly, the effect of short cycles gradually becomes obvious with the increase in the number of iterations. The effect means that, if there is some short cycle in the bipartite graph of the same size **H** matrix, the same performance will occur at different E_b/N_0 s. For example, when we use **H** matrix of size 252*504 to encode and decode LDPC code , one way is to use an **H** matrix 252*504 produced by W1,or W2 with short cycles of length 4 in the bipartite graph, and another way is to apply an **H** matrix 252*504 constructed by methods W3,W4,or W5 without short cycles of length 4 to do same thing. Table 6.2 and bar chart Fig.6.7 show the difference.

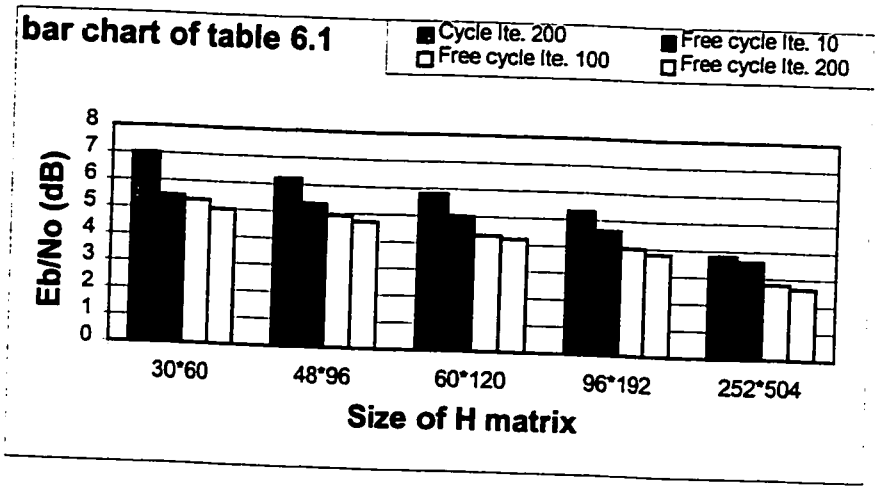


Fig.6.6 E_b/N_0 bar chart of LDPC code of different H matrix based on BPSK in AWGN channel at bit errors probabilities 10^{-4}

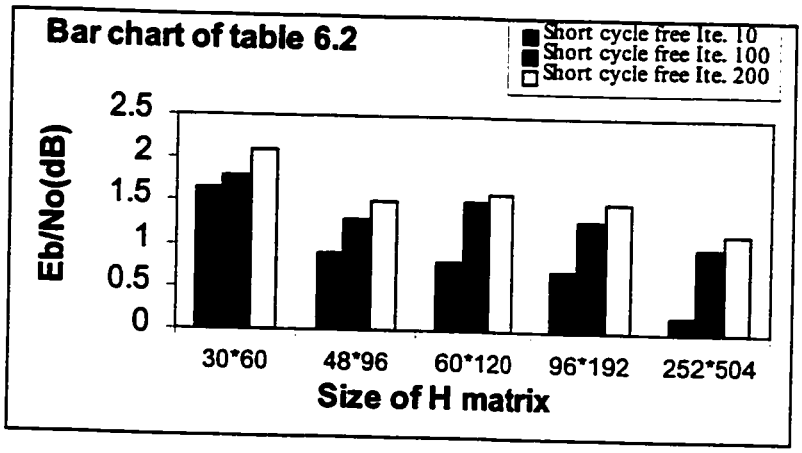


Fig.6.7 Short cycle affecting saving E_b/N_0 bar chart of LDPC code of different H matrix based on BPSK in AWGN channel at bit errors probabilities 10^{-4}

In this case, if the number of iterations is the same number 200 with the **H** matrix of size 252*504, the difference in E_b/N_0 is 1.15 dB. This is an interesting result. The result indicates that using an **H** matrix of size 252*504 constructed by W3, W4, or W5 can save energy 1.15 dB over using an **H** matrix of size 252*504 produced by W1, or W2 . The bar chart of Fig.6.7 show the relationship between the energy saving in dB for E_b/N_0 and the size of the **H** matrix and the iteration number. For the same **H** matrix, more iterations will increase the energy saving in dB for E_b/N_0. These effects occur for all five different sizes of **H** matrices. However, little difference is shown for **H** matrices of sizes 48*96 and 60*120. It seems that the energy saving in E_b/N_0 for an **H** matrix of size 48*96 should be more than the saving for an **H** matrix of size 60*120. But, the results do not show this. The reason should be caused by irregularly separated quality of **H** matrix, (or the reference bars of **H** matrix at Fig. 6.7 are different) . It means that, although the **H** matrices constructed by methods W3, W4, or W5 do not include short cycles of length 4, we do not guarantee that these **H** matrices will always keep no irregularly separated quality. In other words, when we construct these **H** matrices according to the method of W3, W4, or W5, we find that these **H** matrices possess a certain property at random. Just as the property, these **H** matrices will have irregularly separated quality. As a result ,

it is possible not to decrease the energy saving for E_b/N_0 linearly with the increase of the **H** matrix size. Anyway, it is a fact that the short cycles really affect the bit error probabilities. The Table 6.2 and bar chart of Figure 6.7 have shown this.

In Table 6.1 and bar chart of Fig. 6.6, we also find another interesting phenomenon which is that, if we only consider the BER performance produced by employing **H** matrices constructed using W1 or W2 with short cycles in their bipartite graph for encoding and decoding LDPC codes, the bit errors probabilities decrease with the increase of the size of **H** matrix.

Similarly, we can write Tables 6.3 and 6.4 at BER 10^{-5} using Fig. 6.1, 6.2, 6.3, 6.4, 6.5 . We can also draw the corresponding bar charts of Fig. 6.8 and 6.9. Then , analyze them and compare the differences between table 6.1, 6.2 and table 6.3 , 6.4 as well as between bar chart Fig.6.6, and 6.7 and bar chart Fig. 6.8 and 6.9 as follows:

TABLE 6.3

Comparison of E_b/N_0 (dB) at BER 10^{-5} about LDPC Codes based on BPSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	8	7.2	6.7	6	4.3
Free cycle Ite. 10	6.5	6.3	5.7	5.4	3.9
Free cycle Ite. 100	6.3	5.6	5	4.7	3
Free cycle Ite. 200	6.2	5.3	4.9	4.4	2.95

TABLE 6.4

Comparison of the short cycle affecting saving E_b/N_0 (dB) at BER 10^{-5} about LDPC Codes based on BPSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	0	0	0	0	0
Free cycle Ite. 10	1.5	0.9	1	0.6	0.4
Free cycle Ite. 100	1.7	1.2	1.7	1.3	1.3
Free cycle Ite. 200	1.8	1.9	1.8	1.6	1.35

note* the value of table 6.4 is equal to that a **E_b/N_0 (dB)** - E_b/N_0 (dB) of cycle Ite. 200 in same **H** matrix

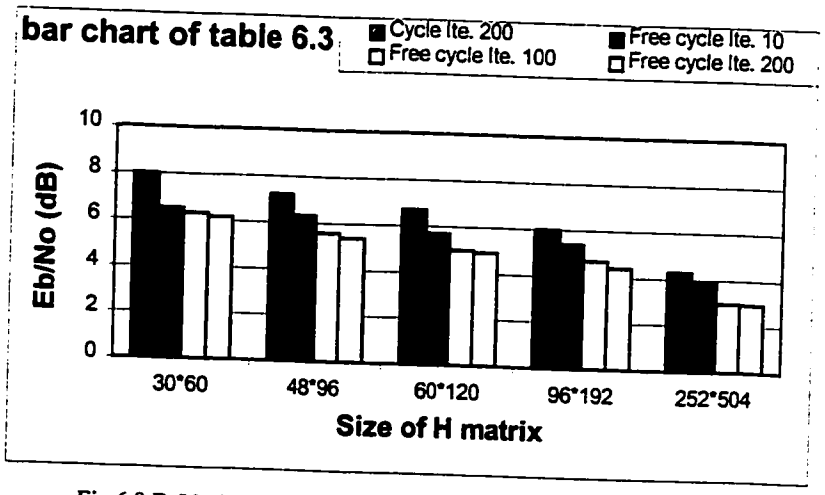


Fig.6.8 E_b/N_0 bar chart of LDPC code of different H matrix based on BPSK in AWGN channel at bit errors probability of 10^{-5}

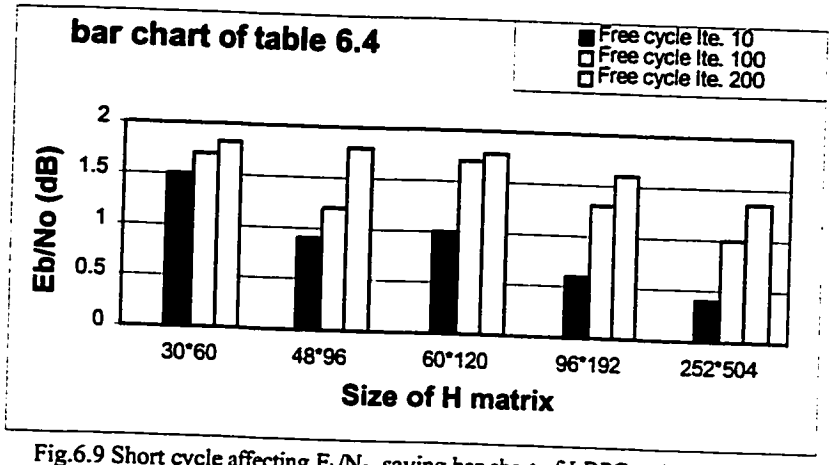


Fig.6.9 Short cycle affecting E_b/N_0 saving bar chart of LDPC code of different H matrix based on BPSK in AWGN channel at bit errors probability of 10^{-5}

Using these tables and bar charts, we can make some comparisons between Table 6.1 and Table 6.3, Table 6.2 and Table 6.4, Fig. 6.6 and Fig. 6.8, as well as Fig.6.7 and Fig 6.9. In general, these comparisons are similar in nature and conclusions with the previous analysis. But, when we study the tables and figures in more detail, we can still find some differences between them. Fig. 6.10 and Fig. 6.11 express these differences. In Fig. 6.10, the result shows that E_b/N_0 saving differences of LDPC code of five sizes of H matrices based on BPSK on AWGN channel between bit errors probabilities 10^{-5} and 10^{-4} still have some commonality. The differences basically decrease with the increase of the iteration number. Besides, irregularly separated quality of **H** matrix also affects the differences so that the differences can not vary according to expected value with the increase of the iteration number and the size of **H** matrices. However, the smaller and smaller differences indicate that the effect of short cycles decreased. In Fig. 6.11, the positive bars express that the energy saving for E_b/N_0 at bit errors probabilities 10^{-5} is greater than the value at 10^{-4}. Similarly, the negative bars indicate a reduced energy saving. Furthermore, the irregular bars prove that the variability of 1 position in different **H** matrices is irregularly separated quality.

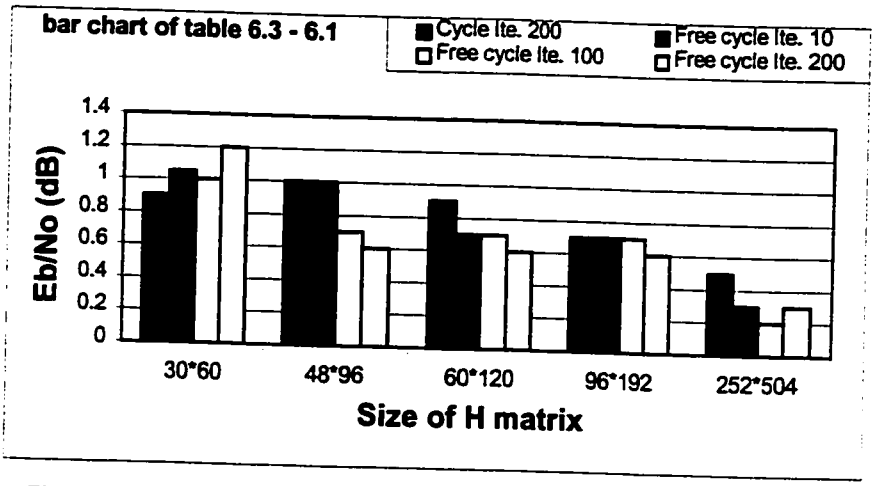


Fig.6.10 E_b/N_0 bar chart difference of LDPC code of five H matrix based on BPSK in AWGN channel between bit errors probabilities 10^{-5} and 10^{-4}

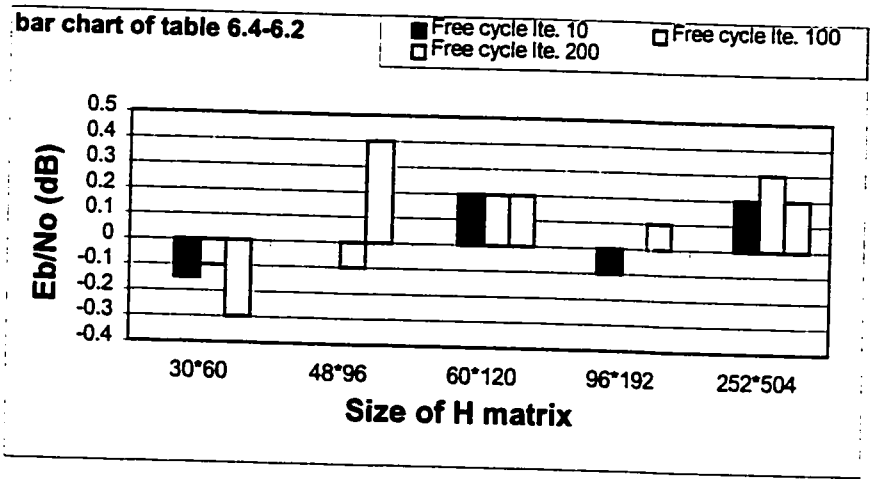


Fig.6. 11 Short cycle affecting E_b/N_0 bar chart difference of LDPC code of five H matrix based on BPSK in AWGN channel between bit errors probabilities 10^{-5} and 10^{-4}

In conclusion, short cycles of length 4 in the bipartite graph affect the bit error probabilities for a given E_b/N_0. Generally speaking, the performance in a short cycle free bipartite graph will save 0.2 ~ 1.9 E_b/N_0 (dB) energy over a code that has short cycles in its bipartite graph. The reason for the difference is that there is irregularly separated quality in different sizes of **H** matrix. However, the effect will be diminished when we employ huge **H** matrixes to encode and decode LDPC codes.

6.3 Performance comparison for Gray mapping of bits to 8PSK on AWGN

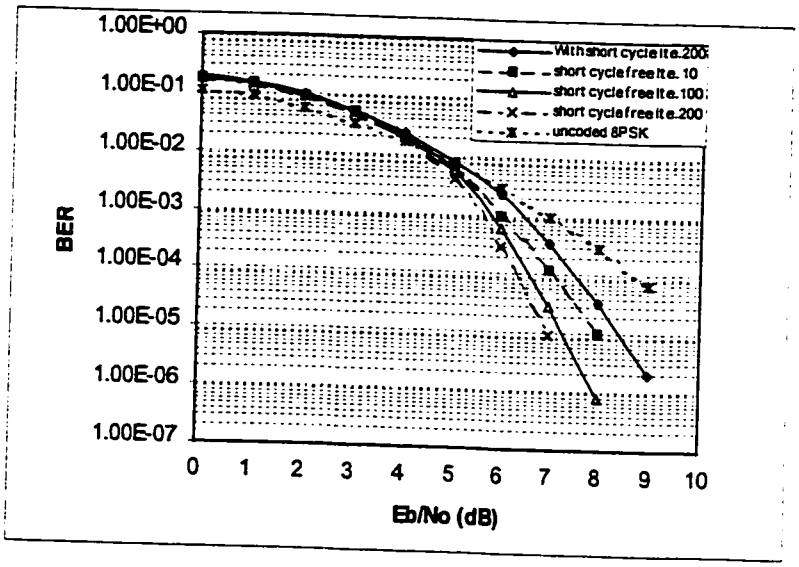


Fig.6.12 Performance comparison of LDPC code of 30*60 H matrix based on 8PSK in AWGN channel

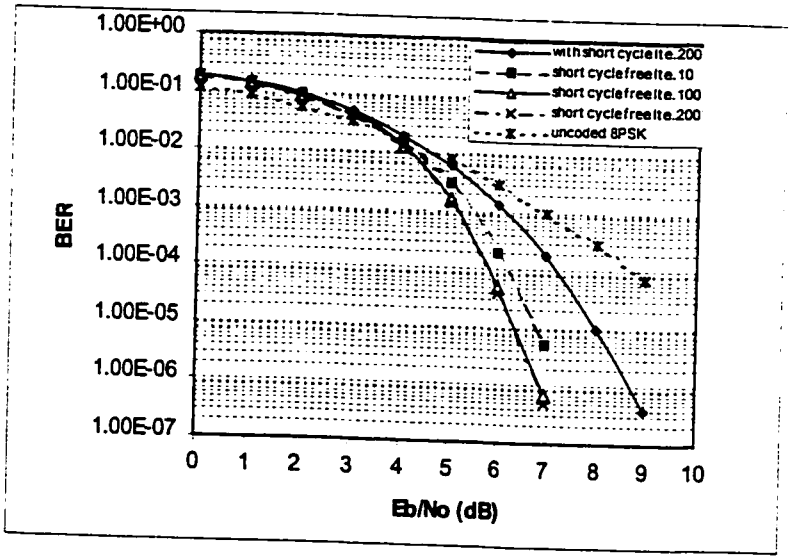


Fig.6.13 Performance comparison of LDPC code of 48*96 H matrix based on 8PSK in AWGN channel

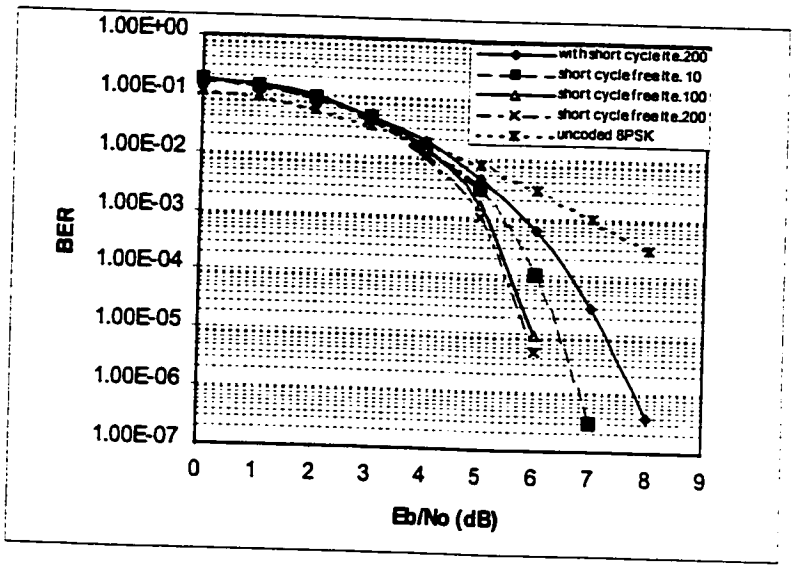


Fig.6.14 Performance comparison of LDPC code of 60*120 H matrix based on 8PSK in AWGN channel

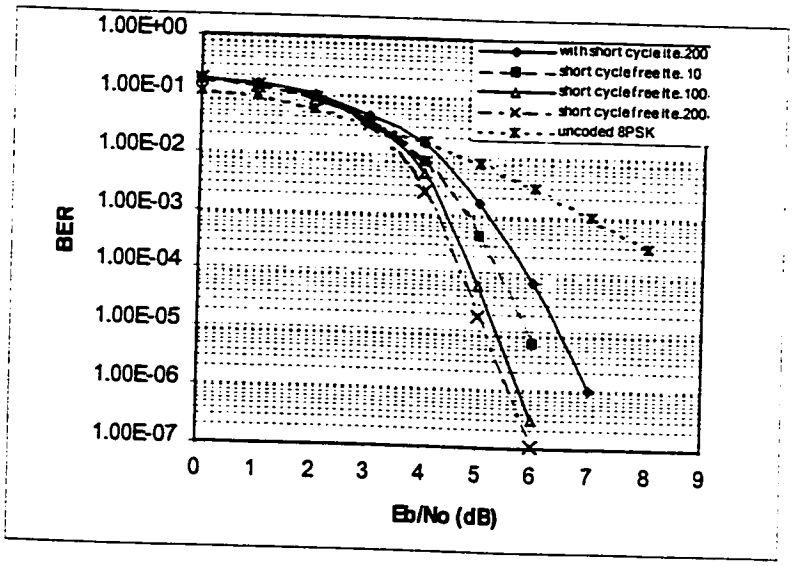


Fig.6.15 Performance comparison of LDPC code of 96*192 H matrix based on 8PSK in AWGN channel

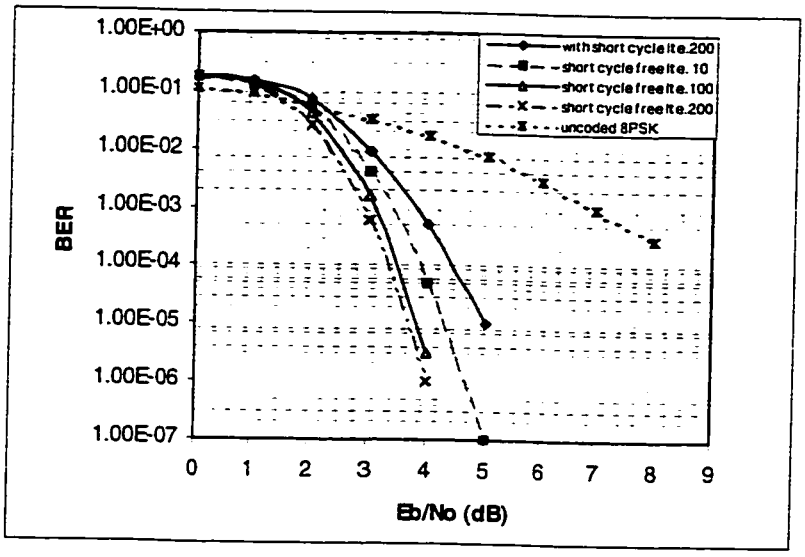


Fig.6.16 Performance comparison of LDPC code of 252*504 H matrix based on 8PSK in AWGN channel

Figs. 6.12, 6.13, 6.14, 6.15, and 6.16 have shown the performance comparison of LDPC code of five different sizes of H matrix for 8PSK in AWGN channel. According to the figures, we list some result in Table 6.5. From Table 6.5, we find that , with increasing the number of iterations, the bit errors will converge. Once the number of iterations reaches 200, the convergence basically is stopped . Furthermore, we also note that the size of the low density parity check **H** matrix affects the bit error probabilities. The bigger the size of **H** matrix is the better is the performance.

TABLE 6.5

Comparison of E_b/N_0 (dB) at BER 10^{-4} about LDPC Codes based on 8PSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	7.6	7.2	6.6	5.9	4.4
Free cycle Ite. 10	7.2	6.2	6	5.4	3.9
Free cycle Ite. 100	6.6	5.8	5.6	4.8	3.4
Free cycle Ite. 200	6.3	5.7	5.5	4.6	3.3

TABLE 6.6

Comparison of the short cycle affecting E_b/N_0 (dB) at BER 10^{-4} about LDPC Codes based on 8PSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	0	0	0	0	0
Free cycle Ite. 10	0.4	1	0.6	0.5	0.5
Free cycle Ite. 100	1	1.4	1	1.1	1
Free cycle Ite. 200	1.3	1.5	1.1	1.3	1.1

note* the value of table 6.6 is equal to that a E_b/N_0 (dB) - E_b/N_0 (dB) of cycle Ite. 200 in same H matrix

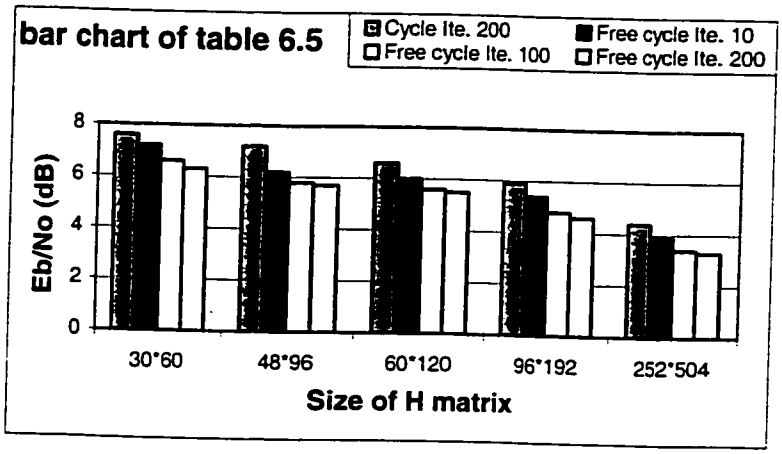


Fig.6.17 bar chart of LDPC code of different H matrix based on 8PSK in AWGN channel at bit errors probabilities 10^{-4}

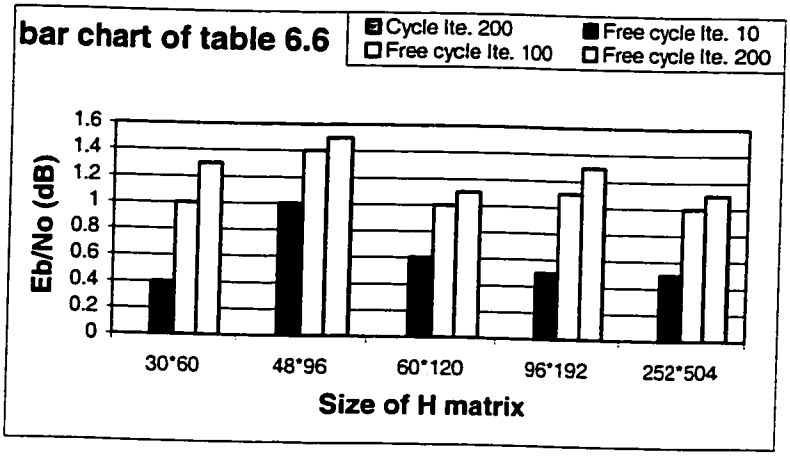


Fig.6.18 Short cycle affecting E_b/N_0 bar chart of LDPC code of different H matrixes based on 8PSK in AWGN channel at bit errors probabilities 10^{-4}

TABLE 6.7

Comparison of E_b/N_0 (dB) at BER 10^{-5} about LDPC Codes based on 8PSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	8.5	8	7.2	6.5	5
Free cycle Ite. 10	8	6.9	6.4	5.9	4.3
Free cycle Ite. 100	7.3	6.4	6	5.3	3.9
Free cycle Ite. 200	7	6.3	5.9	5.1	3.6

TABLE 6.8

Comparison of the short cycle affecting E_b/N_0 (dB) at BER 10^{-5} about LDPC Codes based on 8PSK and AWGN channel

Number of Ite.\ H matrix	30*60	48*96	60*120	96*192	252*504
Cycle Ite. 200	0	0	0	0	0
Free cycle Ite. 10	0.5	1.1	0.8	0.6	0.7
Free cycle Ite. 100	1.2	1.6	1.2	1.2	1.1
Free cycle Ite. 200	1.5	1.7	1.3	1.4	1.4

note* the value of table 6.8 is equal to that a E_b/N_0 (dB) - E_b/N_0 (dB) of cycle Ite. 200 in same H matrix

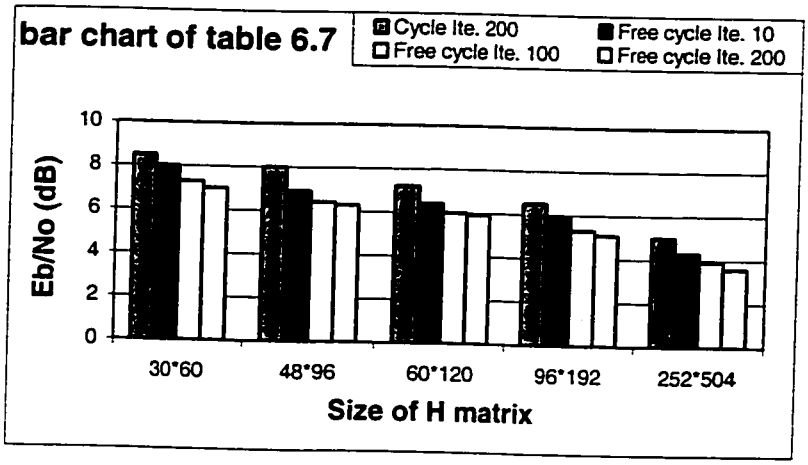


Fig.6.19 E_b/N_0 bar chart of LDPC code of different H matrixes based on 8PSK in AWGN channel at bit errors probabilities 10^{-5}

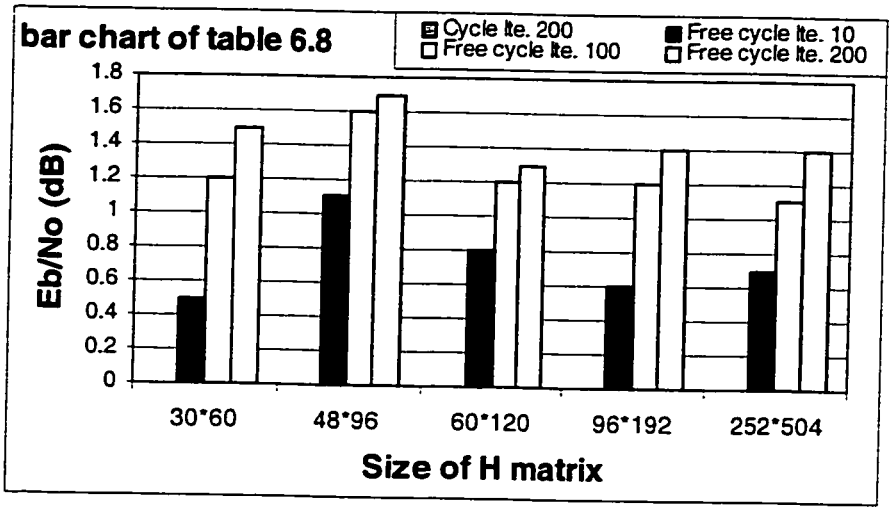


Fig.6.20 Short cycle affecting E_b/N_0 bar chart of LDPC code of different H matrixes based on 8PSK in AWGN channel at bit errors probabilities 10^{-5}

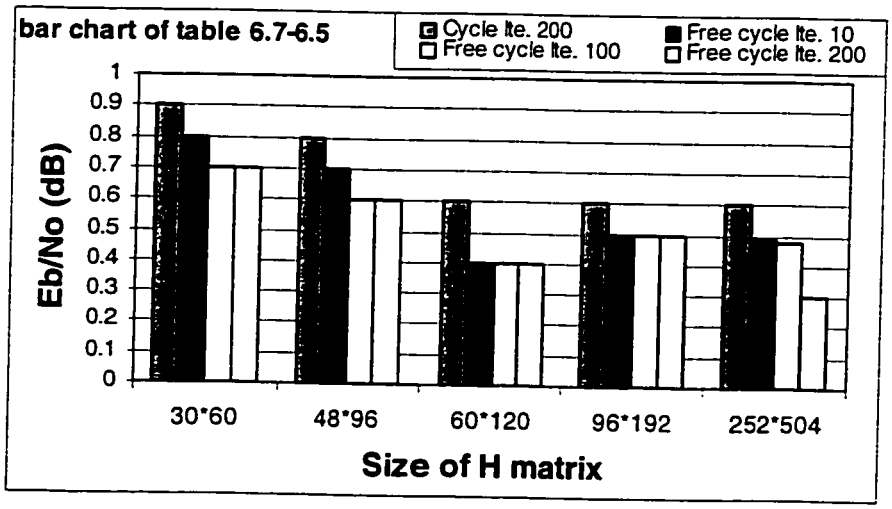


Fig.6.21 E_b/N_0 bar chart difference of LDPC code of five sizes of H matrix based on 8PSK in AWGN channel between bit errors probabilities 10^{-5} and 10^{-4}

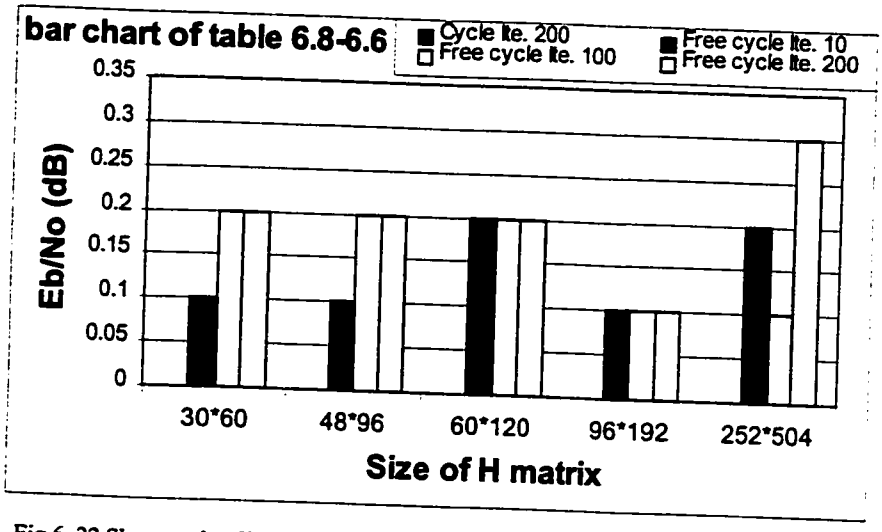


Fig.6. 22 Short cycle affecting E_b/N_0 bar chart difference of LDPC code of five sizes of H matrix based on 8PSK in AWGN channel between bit errors probabilities 10^{-5} and 10^{-4}

After analyzing Table 6.6 and comparing to the bar chart of Fig. 6.18, we note that the short cycles in the bipartite graph affects the performance of LDPC codes with 8PSK over AWGN channel. Firstly, the effect of short cycles gradually becomes obvious with the increase in the number of iterations for codes without short cycles. The effect means that, if there is some short cycle in the bipartite graph of an **H** matrix of a given size, the same bit error probabilities will arise for a different values of E_b/N_0.

If the number of iterations is set to 200, the results show that for an **H** matrix of size 252*504 constructed by W3, W4, or W5 we can save 1.4 dB energy over an **H** matrix of size 252*504 produced by W1, or W2 . The bar chart Fig. 6.18 shows the relationship between the various values of energy saving in dB and the size of the H matrix as well as the number of iterations. For a given **H** matrix, a large number of iterations will also increase the energy saving for E_b/N_0. However, the increase is not linear.

Similar to Section 6.2, we also draw bar charts of Fig.17, Fig.18, Fig.19, Fig.20, Fig.21, and Fig.22. After analyzing , the figures lead to similar results and interpretation as Fig.6.6, Fig.6.7, Fig.6.8, Fig.6.9, Fig.6.10, and Fig.6.11.

In summary, the results show that short cycles of length 4 in the bipartite graph also affect the bit error probabilities for a given E_b/N_0 . In general, the performance of codes without short cycles in their bipartite graph will save 0.5 ~ 1.7 E_b/N_0 (dB) energy relative to the performance of codes with short cycles in their bipartite graph at a BER of 10^{-5}.

6.4 Comparison of performance between Gray mapping of bits to 8PSK and mapping of bits to BPSK on AWGN

In Sections 6.2 and 6.3, we have respectively analyzed the performance of mapping of bits to BPSK and performance of Gray mapping of bits to 8PSK on AWGN channel. So far, we have known the fact that the short cycles affect the performance. But, we do not know the difference between Gray mapping of bits to 8PSK and mapping of bits to BPSK on AWGN channel. Therefore, we will use four bar charts to show the difference. The four bar charts are as follows:

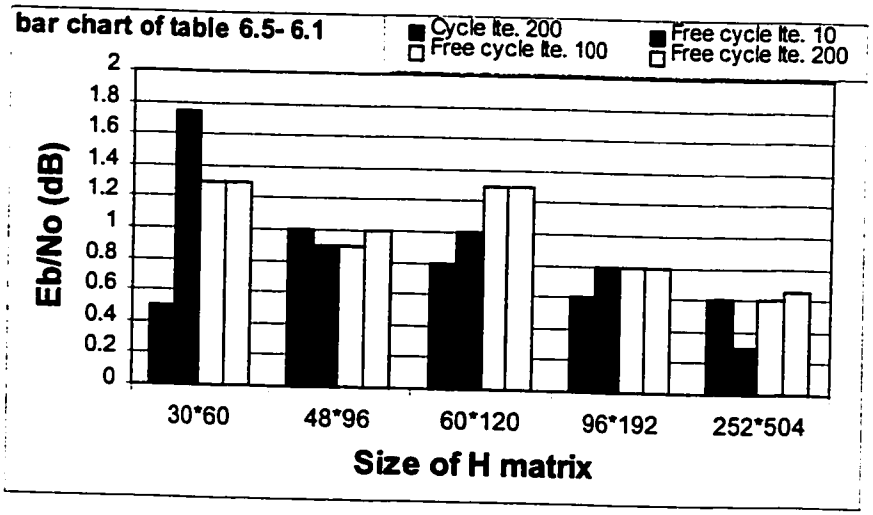


Fig.6.23 E_b/N_0 bar chart difference between LDPC codes of five sizes of H matrices based on BPSK and 8PSK in AWGN channel at bit errors probabilities 10^{-4}

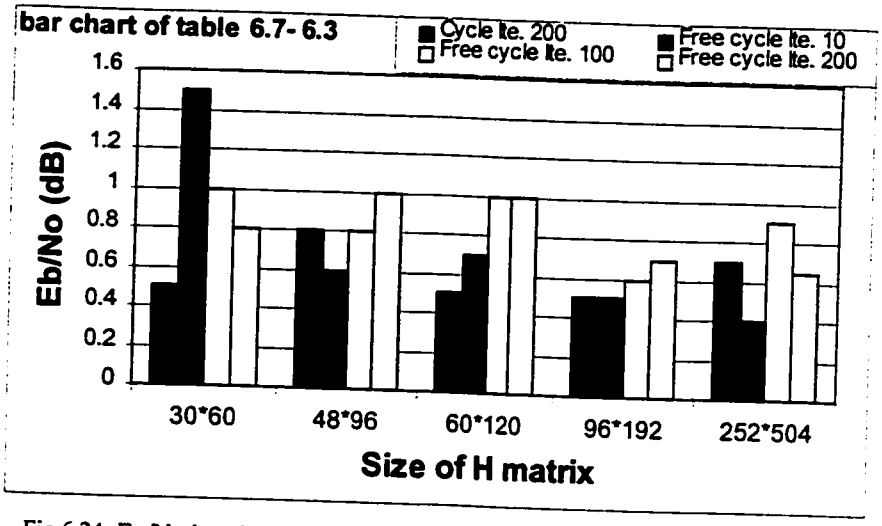


Fig.6.24 E_b/N_0 bar chart difference between LDPC codes of five sizes of H matrices based on BPSK and 8PSK in AWGN channel at bit errors probabilities 10^{-5}

In Fig. 6.23, the bar chart shows the saved energy E_b/N_0 dB difference between LDPC codes of five sizes of H matrices based on BPSK and 8PSK in AWGN channel at bit errors probabilities 10^{-4}. The minimum difference is 0.3 dB, and maximum difference is 1.75 dB. The result means that, in order to arrive at the same performance , the used energy based on BPSK will be 0.3 ~ 1.75 dB less than when 8PSK is used. Furthermore, in Fig.6.24, there are similar conclusions at bit errors probabilities 10^{-5}. The minimum difference is 0.4 dB, and maximum difference is 1.5 dB. But, the differences are not linear.

Using Fig.6.23 and Fig. 6.24, we can further analyze the convergence behavior related to the differences. Firstly, when we draw Fig. 6.25 and Fig. 6.26 , the bars labeled Cycle Ite. 200 represent the asymptotic performance for each **H** matrix and can be regarded as a reference for consideration of convergence behavior.

Both bar chart of Fig. 6.25 and Fig 6.26 show that the effect of short cycles based on reference bar of Cycle Ite. 200 is not simply increasing or decreasing in every **H** matrix. In other words, the bars change with increasing and decreasing of iteration

number. But, the variable bars are either positive or negative. The results express that there is difference of convergence between Gray mapping of bits to 8PSK and mapping of bits to BPSK on AWGN . The phenomenon belongs to the character of irregularly separated quality in different **H** matrices.

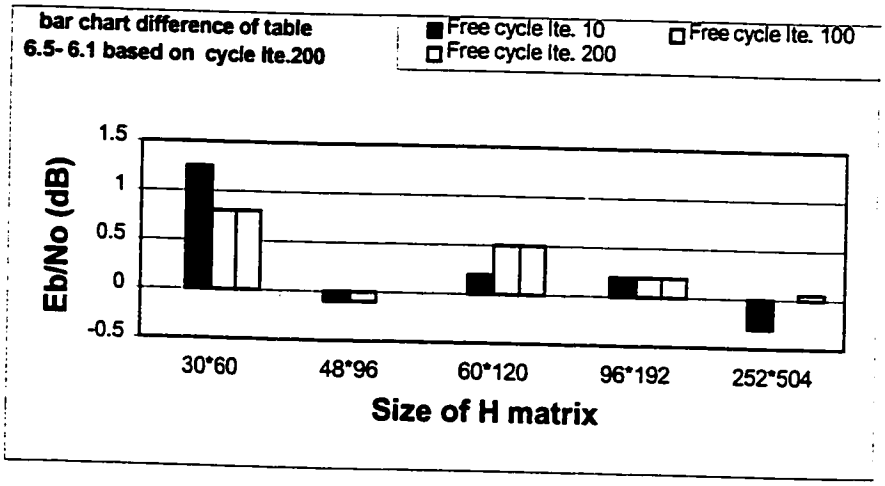


Fig.6.25 E_b/N_0 bar chart difference based on cycle Ite.200 between BPSK and 8PSK in AWGN channel at bit errors probabilities 10^{-4}

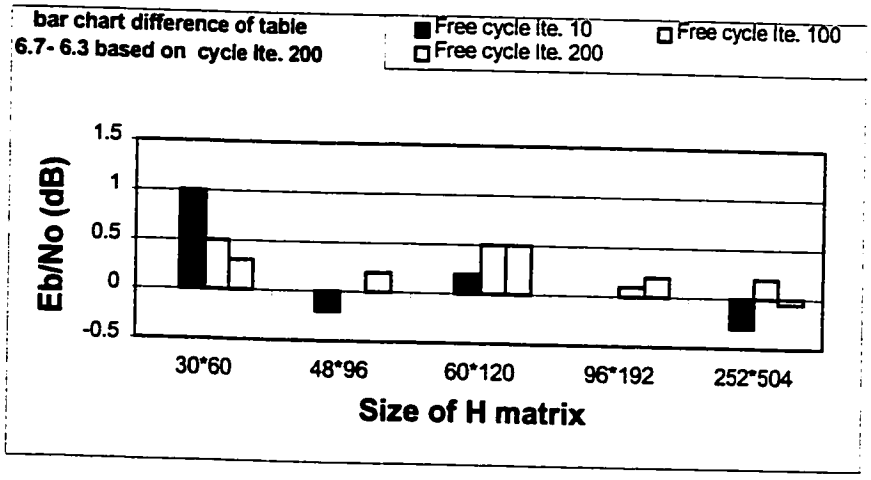


Fig.6.26 E_b/N_0 bar chart difference based on cycle Ite.200 between BPSK and 8PSK in AWGN channel at bit errors probabilities 10^{-5}

6.5 Performance comparison with an anti-Gray mapping of bits to 8PSK on AWGN

We have done some simulation similar to the previous schemes, with encoding, channel simulation, anti-Gray mapping of bits to 8PSK , and decoding based on the sum-product algorithm. The simulation results have been obtained. Unfortunately, these results show that performance is so bad that the curves of the bit errors probabilities basically limit with the curves of uncoded performance. As a result, it is not necessary for us to list the associated simulation results. Finally, we can conclude that the performances of the anti-Gray mapping of bits to 8PSK for AWGN is not suitable for the schemes in this thesis.

6.6 Comparison between LDPC code and Turbo code

In general , both LDPC codes and Turbo codes are excellent codes. They have four differences.

Firstly, Turbo code [2][21] has become very popular in recent years. In terms of E_b/N_0, Turbo code is better than LDPC code. However, there are some similarities between the codes. The sum product algorithm may be used in turbo decoding algorithm [12][13][17].

Secondly, using the sum product algorithm, we can state that all the bit errors produced by LDPC codes that we have viewed are detected errors; in contrast , the bit errors made by turbo codes are undetected errors. Likely, turbo code errors are caused by some low weight codewords. Furthermore, the complexity of LDPC codes is less than that of turbo codes.

Finally, the meaning of decoding iteration between LDPC codes and Turbo codes is different. The former means the number of iteration is a limit , and it does not

mean every codeword must pass that number of iterations when decoding. For example, the number of iteration is 500, but some codewords run less than 10 times. The correct codewords have been solved. However, the number of iteration in Turbo code is completely used for decoding every codeword.

Chapter 7

Conclusions and Some Suggestions For Further Work

7.1 Conclusions.

Low density parity check codes were hardly paid any attention to in the communication field and other scientific literature from 1962 to 1995. But, since MacKay rediscovered LDPC codes 1995[5], research into LDPC codes has become more and more popular. Indeed, when one employs huge **H** matrices, some literature has shown that the performance is substantially better than that of standard convolutional turbo codes or of block turbo codes.[14][24] . Recent work has shown they can arrive at the capacity under optimal decoding.[9].

The main objective of the thesis was to develop some more efficient **H** matrices which are easier operated by computer program and to analyze the effect of short cycles of length 4 in the bipartite graph. In order to do this, we first studied the sum product algorithm which can be used as a decoding algorithm for LDPC codes. Then, we did some simulations of the LDPC codes. Our objective has been to find the best **H** matrix because we know a better **H** matrix can improve the performance of LDPC codes. Therefore , five schemes to construct the **H** matrix were considered. The **H**

matrices produced by schemes W1 and W2 include short cycles of length 4 in their bipartite graph. Meanwhile, the **H** matrices constructed by schemes W3 , W4, and W5 do not possess short cycles of length 4 in their corresponding bipartite graph. When the **H** matrices are constructed, not only do the weight of every column stay the same, but also the weight of every row is the same. This means that the LDPC codes encoded by the **H** matrices are completely regular. These LDPC codes are completely consistent with the definition by Gallager in 1963. [3].

For the encoding, at first, we mainly studied the efficient encoding based on approximate lower triangulations.[10]. This scheme arises from the sparseness of the parity-check matrix **H** and the algorithm can be used for any sparse matrix **H** . If the rows are linearly dependent, then the algorithm which is used for the decoding can find the dependency. Therefore, one can either change to another matrix **H** or eliminate the redundant rows from the matrix **H** in the process of encoding. Thus reasonable LDPC codes can still be encoded. Then, we also studied the systematic encoders based on a generator matrix. In fact, the generator is derived from a sparse matrix **H.** Similarly, we require that the rows of matrix **H** are linearly independent. If the rows of matrix **H** are not linearly independent , all dependent rows must be removed and only those rows which are linearly independent be kept. As a result, the matrix **H** will have the same number of columns and fewer rows. One obtains another matrix **H** whose rows are linearly independent. An equivalent generator matrix will be derived after using Gaussian elimination and reordering of columns . Finally, we combined the two encoding schemes to derive systematic encoders only based on an H matrix. The major idea is that , when a low density parity check **H** matrix has been obtained, we use Gaussian elimination and reorder columns to get a new **H** matrix of the form ($\mathbf{H}' = [\,\mathbf{P} \mid \mathbf{I}_M\,]$) . Then, we apply the condition $\mathbf{H}*\mathbf{t}^T = 0^T$ to encode LDPC codes. The encoding processing is clearer than the systematic encoders based on a generator matrix, because the codewords obtained by the method have direct relationship with the **H** matrix and do not need to be transformed into some other matrix like a generator matrix. However, when we use the scheme

to encode LDPC codes, we still want to consider the problem whether the rows of matrix **H** are linearly independent. If the rows of the matrix **H** are not linearly independent , all requirements are the same as the second encoding scheme. In this thesis, the final encoding scheme was adopted.

For the channel, we focus on the simple case of a channel with inputs of $t \in \{0,1\}$ and real valued outputs. For independent noise included in the arbitrary received vector **r**, the bits being 1 is determined by the likelihood ratio for the received signal. In the thesis, we derived the mathematic equations for mappings of the bits to BPSK, Gray mapping of bits to 8PSK , and anti-Gray mapping of bits to 8PSK for AWGN. Then, the results of the simulation channel were fed into the sum product algorithm.

For the decoding, we implemented syndrome decoding combined with the sum product algorithm. Using the bipartite graph corresponding to the **H** matrix, we simulated five different sizes of LDPC codes with code rate 0.5. These LDPC codes employed different sizes of **H** matrix and different mappings of bits.

For the analysis of simulation result, we carefully compared the performances between 200 iterations of codes with length 4 cycles and 10, 100 and 200 iterations of codes without length 4 cycles at the point of bit errors probabilities of 10^{-4} and 10^{-5}. The simulation results have shown that the short cycles of length 4 in the bipartite graph affect the performance of LDPC codes. For the mapping of bits to BPSK in the thesis, the maximum difference is 2.1 dB at the bit errors probability of 10^{-4},and 1.9 dB at 10^{-5}. For Gray mapping of bits to 8PSK, the maximum difference is 1.5 dB at bit error probability of 10^{-4},and 1.7 dB at 10^{-5}. Obviously, the performance of the former is better than the latter.

7.2 Some suggestions for further work .

The performance of LDPC codes , which is affected by short cycles in the regular bipartite graph based on the **H** matrixes, has received attention in the thesis. However, three suggestions can be made for further research in the future.

♣ Block length

In this thesis, we concentrated on short block LDPC codes. After doing the simulation, we know that the short cycles of bipartite graph affect the performance of LDPC codes. However, with the increase of the block length, will the effect be eliminated or decreased? We guess the effect should decrease because the **H** matrix will become more and more sparse with the increase of the matrix size. If the guess is true, what size of **H** matrix will be enough for this to become obvious? How long should the Block length be?

♣ The effect of short cycle in the irregular LDPC codes

When we researched the effect of short cycles in the regular bipartite graph, the weight of every row and every column in the **H** matrix was , constrained to be the same. Therefore, it is not easy for us to find the suitable and good **H** matrix although some methods of **H** matrix construction were presented. If the constraint is relaxed, the **H** matrix should be easy to obtain. But, the **H** matrix is not regular, rather it is irregular. For the irregular **H** matrix, we must first apply the encoding scheme, then do some further simulation. This would let us analyze the effect of short cycles in the irregular bipartite graph corresponding to the **H** matrix.

- An application for LDPC codes

It is well known that the LDPC codes are single block , low complexity , and have good performance. Therefore, when we research their application in the future, not only will we consider them for use in the general communication field, but also we should develop new fields , for example : electrical power systems. In these system, there is a lightning line above a set of electrical power transmission lines. The lightning line can be used to deliver information. If the technology of combining the LDPC codes and communication using the lightning line will be developed, the two aspects will benefit. One aspect is that the electrical power system will obtain greater communication quality and also decrease the cost ; another aspect is that the application will enrich the research of LDPC codes.

Appendix A

The statistical schemes used in the simulation

A.1 Binomial distribution

It is well known that the a binomial experiment possesses the following four properties:

(1) The experiment consists of a fixed number N_{size} of trials.

(2) The result of each trial can be classified into one of two categories: success or failure.

(3) The probability p of a success remains constant in each trial.

(4) Each trial of the experiment is independent of the other trials.

In the simulation of LDPC codes , we assume that the bit error probability of a received bit is p (error) or a correct bit (correct) in a given E_b/N_0 dB. Obviously, the condition is satisfied by the binomial experiment.

A.2 Normal approximation of the Binomial Distribution

The estimator of a population proportion of errors is the sample proportion. That is, we define the number of errors in a sample and calculate in the formula (A.1)

$$\hat{p} = \frac{X_{num}}{N_{size}} \tag{A.1}$$

Where X_{num} is the number of errors and N_{size} is the sample size. The normal distribution can be used to approximate the probability distribution of the binomial. The normal approximation of the binomial distribution works best when the number of experiments (sample size) is large, for example, $N_{size}p>5$ and $N_{size}(1-p)>5$. Thus, the number of errors in N_{size} identical independent tests X_{num} is approximately a normal distribution, whose mean is $N_{size}p$ and and standard deviation $\sqrt{N_{size}p(1-p)}$. We can derive (A.2) [26].

$$Z_{sn} = \frac{\hat{p} - p}{\sqrt{p(1-p)/N_{size}}} \tag{A.2}$$

Where Z_{sn} is approximately standard normally distributed [26].

A.3 Confidence level

First, let us define that the sample mean $\bar{X}$ as the arithmetic average of all samples, and the population standard deviation is σ_d. We can derive confidence interval according to confidence level of Table A.1.

Table A.1 Tail probabilities for the normal distribution

1-α	α	α/2	$Z_{\alpha}/2$
0.9	0.1	0.05	$Z_{0.05}$=1.645
0.95	0.05	0.025	$Z_{0.025}$=1.96
0.98	0.02	0.01	$Z_{0.01}$=2.33
0.99	0.01	0.005	$Z_{0.005}$=2.575

The lower confidence limit is $\bar{X} - Z_{\alpha/2}\sigma_d / \sqrt{N_{size}}$, and the upper confidence limit is $\bar{X} + Z_{\alpha/2}\sigma_d / \sqrt{N_{size}}$.

A.4 Selecting the sample size

From the Section A.3, we know the formula for the interval estimate population mean is $\bar{X} - Z_{\alpha/2}\sigma_d / \sqrt{N_{size}}$ and $\bar{X} + Z_{\alpha/2}\sigma_d / \sqrt{N_{size}}$. Now, let us derive a general formula for the sample size needed to estimate a population mean. Let W_{wide} represent the width of the interval. Therefore, when an interval estimator $\bar{X} \pm W_{wide}$ has 1-α confidence, the required sample size is

$$N_{size} = \left(\frac{Z_{\alpha/2}\sigma_d}{W_{wide}} \right)^2 \qquad \text{(A.3)}$$

However, we do not know σ_d , and only know $\hat{p}$. If the $N_{size}\hat{p}$ and $N_{size}(1-\hat{p})$ are greater than 5, the sampling distribution of $\hat{p}$ is also approximately normal with mean p and standard deviation $\sqrt{p(1-p)/N_{size}}$. We can also derive an interval estimator $\hat{p} \pm W_{wide}$ of p .

$$\hat{p} \pm W_{wide} = \hat{p} \pm Z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N_{size}}} \qquad \text{(A.4)}$$

Clearly, if $\hat{p}$, N_{size} , and confidence level are given , we can get the confidence interval . For example, determine the confidence interval of $N_{size}=10^6$, $\hat{p}=10^{-4}$, with 95% confidence. Then, 1-α=0.95, α=0.05, α/2=0.025, $Z_{\alpha/2}=Z_{0.025}=1.96$.

$$\hat{p} \pm W_{wide} = 10^{-4} \pm 1.96\sqrt{\frac{10^{-4}(1-10^{-4})}{10^{6}}}$$
$$= 10^{-4} \pm 0.196 \times 10^{-4}$$

Where the (10^{-4} - 0.196×10^{-4} , 10^{-4} + 0.196×10^{-4}) expresses the confidence interval. The 10^{-4} - 0.196×10^{-4} is called the lower confidence limit (LCL), and the10^{-4}+ 0.196×10^{-4} is called the upper confidence limit (UCL).

However, what we are interested in is the N_{wide} because we want to know how large the sample size can satisfy the 95% confidence of bit error probabilities p in the simulation of LDPC or other codes. First, let us see a simple fact. Suppose that as a result of statistical study we estimate with 95% confidence that the average bit error probabilities lie between 10^{-1} and 10^{-5}. This interval is so wide that very little information was derived from the data. Suppose, however, that the interval estimate was 0.91×10^{-5} to 1.09×10^{-5}. This interval is much smaller, and thus provides the bit error probabilities with more precise information. The W_{wide} of the interval estimate is a function of the sample proportion, sample size, and confidence interval. So, we can derive the sample size N_{size} according to the formula (A.4), if the W_{wide} is given. In fact, W_{wide} is one of the key parameters for us to derive the sample size N_{size} in order to ensure the confidence .

The size of sample will be

$$N_{size} = \left(\frac{Z_{\alpha/2}\sqrt{\hat{p}(1-\hat{p})}}{W_{wide}} \right)^2 \qquad \text{(A.5)}$$

We find that decreasing the width of the interval will increase the sample size from the formula (A.5). In other words, a larger sample size provides more potential information. The increased amount of information is reflected in a narrower interval, however increasing the sample size increases the sampling cost.

In the practical simulation of LDPC and other codes, it is impossible for us to run an infinite number of bits to find a bit error probability p. Therefore, only the thing we can do is to use enough sample size to limit p. In general,

if	$p=10^{-e'}$,
we take	$W_{wide}= 0.19\times10^{-e'}$ or less
or	$N_{size}=10^{e'+2}$ or more

Where e' is a positive integer. For example $p=10^{-4}$, what sample size will make $\hat{p}$ limit $p=10^{-4}$ with 95% confidence of $W_{wide} = 0.18\times10^{-4}$? Then, according to the formula (A.5) the sample size will , at least, be

$$N_{size} = \left(\frac{1.96\sqrt{10^{-4}(1-10^{-4})}}{0.18\times10^{-4}} \right)^2 = 1185560 \approx 1.18\times10^{6}(bits)$$

The above equation means that , when we simulate LDPC codes and require $p=10^{-4}$ performance of bit errors probabilities with 95% confidence in the interval between 10^{-4} –0.18×10^{-4} and 10^{-4} +0.18×10^{-4}, we must simulate at least 1180000 bits . Otherwise, the performance does not possess sufficient confidence. Similarly, we can get that sample size for 10^{-5} performance of bit errors probabilities with 95% confidence is 11856790 bits.

In this thesis, all the simulations satisfy this condition. One can check the results one by one . For example, when we check the simulation result of Fig. 6.5 , the sample size is 11894400 bits at 3 dB and $\hat{p}=10^{-5}$ for the cycle free bipartite graph using 200 iterations. Does the size satisfy the above confidence level ? Clearly, because the size 11894400 is greater than 11856790, the simulation result at bit error probability 10^{-5} has 95% confidence for the interval between 10^{-5}- 0.18×10^{-5} and 10^{-5} +0.18×10^{-5}.

Bibliography

[1] C.E. Shannon, "A mathematical theory of communication", Bell Sys. Tech. J., 27:379- 423, 623-656, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes", In Proc. 1993 IEEE International Conference on Communications, Geneva, Switzerland, pages 1064-1070, 1993.

[3] R.G. Gallager, "Low density parity check codes, in Research Monograph series", Cambridge, MIT Press (1963).

[4] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices", IEEE Trans. Inform. Theory, IT-45, pp.399-431 , 1999.

[5] D.J.C.MacKay and R.M.Neal, "Good codes based on very sparese matrices", in Colin Boyd, editor, Cryptography and Coding. 5th IMA Conference , number 1025 in Lecture Notes in Computer Science, pages 100-111, Springer, Berlin, 1995.

[6] David.J.C MacKay, Simon T. Wilson, and Matthew C. Davey, "Comparison of Constructions of Irregular Gallager Codes", IEEE Trans. On Communications, Vol.47 No. 10 October 1999.

[7] Michael G. Luby, Michael Mitzenmacher, M Amin Shokrollahi, and Daniel A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs," IEEE Trans. On Inform. Theory, Vol.47 No. 2 Feb. 2001.

[8] M.G. Luby, M. Mitzenmacher, M.A Shokrollahi, and D.A .Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation", in Proceedings of the IEEE International Symposium on Information Theory(ISIT), page 117, 1998.

[9] T. Richardson, A Shokrollahi, and R.Urbanke, "Design of provably good low-density parity-check codes," IEEE International Symposium on Information Theory, p.199, June 2000 .

[10] Thomas J. Richardson and Rudiger L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes", IEEE Trans. On Inform. Theory, VOL, 47, NO. 2 FEBRUARY 638-656 2001.

[11] R.M Tanner, "A recursive approach to low complexity codes", IEEE Trans Inform. Theory vol 17-27 pp 533-547 Sept 1981.

[12] N Wiberg H-A loeliger ,and R, kotter, "Codes and iterative decoding on general graphs", Eur, Trans Telecomm. Vol. 6. pp 513-525 Sept./Oct. 1995

[13] N Wiberg, "Codes and decoding on general graphs", ph.D dissertations univ. Linkoping , Linkoping Sweden, 1996.

[14] D.J.C Mackay and R.M Neal, "Near Shannon limit performance of low-density parity-check codes", Elect. Let., vol. 32, pp. 1645-1646, Aug. 1996.

[15] M. Sipser and D.A. spielman, "Expander codes", IEEE trans inform, theory vol 42 pp 1660-1686 Nov 1996.

[16] F.R. Kschischang and B.J Frey, "Iterative decoding of compound codes by probability propagation in graphical models", IEEE, J , select Areas Commmun, vol 16, pp 219-230 Feb 1998.

[17] R.J. McEliece, D. J. C. Mackay, and J-F Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation'algorithm", IEEE , J Select Areas Commun, vol 16.pp. 140-152 Feb 1998.

[18] M.C.Davey and D.J.C. MacKay, "Low density parity check codes over GF(q)", IEEE Communications Letter, 2(6): 165-167, june 1998.

[19] Stephan ten Brink, "Designing Iterative Decoding Schemes with the Extrinsic Information Transfer Chart", AEU Int. J. Electron Commun. 54(2000) No. 6, 389-398.

[20] Shu Lin & Daniel J . Costello, Jr, "Error control coding fundamentals and applications", ISBN 0-13-283796-X 1983 by Prentice-Hall , Inc, England Cliffs, N.J 07632

[21] C.Berrou, A.Glavieux, "Near optimumerror error-correcting coding and

decoding: Turbo codes",IEEE Trans. Commun., vol.44, pp.1261-1271,Oct.1996

[22] L.R. Bahl, J Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", IEEE Trans. Inform. Theory, vol. IT-20, pp. 284-287 , 1974.

[23] L.E. Baum and T.Pettrie, "Statistical inference for probabilistic function of finite-state Markov Chains", Ann. Math.Stat., vol.37,pp.1559-1563, 1966.

[24] Qi Wang, and Lei Wei, "Graph-based Iterative Decoding Algorithm for parity-Concatenated trellis Codes", IEEE Trans. Inform. Theory, vol. 47, NO. 3, MARCH 2001.

[25] Sae-Young Chung, Thomas J, Richardson, and Rudiger L. Urbanke, "Ananlysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation", IEEE Trans. Inform. Theory, vol. 47, NO. 2, FERBRUARY 2001.

[26] Gerald Keller, Brian Warrack , "Statistics for Management and Economics", Belmont, CA: Wadsworth/Thompson Learning, c2001 Publisher, 5th ed. ISBN 0534475590(pbk/CD_ROM). pp.160-298.

[27] M. Luby, M. Mitzenmacher, A Shokrollahi, D.Spielman , and V.Stemann, "Practical loss-resilient codes", in Proc. 29th Annual ACM Symp. Theory of Computing, 1997, pp. 150-159.

[28] Branka Vucetic & Jinhong Yuan, "Turbo Codes Principles and Applications", Kluwer Academic Publishers, ISBN 0-7923-7868-7.

[29] Peter Hoeher, and John Lodge, "Turbo DPSK: Iterative Differential PSK Demodulation and Channel Decoding", IEEE Transactions on communications, VOL. 47, NO. 6, JUNE 1999.

[30] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger, "Factor Graphs and the Sum-Product Algorithm", IEEE Transactions on information theory, VOL. 47, NO. 2 Feb. 2001.

[31] Marc P.C. Fossorier, "Iterative Reliability-Based Decoding of Low-Density Parity Check Codes", IEEE, Journal on Selected Areas in Communications, vol. 19, No. 5 May 2001.

[32] David Burshtein, and Gadi Miller, "Expander Graph Arguments for Message-Passing", IEEE Transactions on information theory, Vol. 47, NO. 2, February 2001.

[33] Joachim Hagenauer, Elke Offer, and Lutz Papke, "Iterative Decoding of Binary Block and Convolutional Codes", IEEE Transactions on information theory, Vol. 42 , No. 2 March 1996.

[34] G. David Forney, Jr., "Codes on Graphs : Normal Realizations", IEEE Transactions on information theory, Vol. 47 , No. 2 February 2001.

[35] Ramesh Mahendra Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes", IEEE Transactions on communications , Vol. 46 NO. 8. AUGUST 1998.

[36] Thomas J. Richardson, M. Amin Shokrollahi, and Rudiger L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes", IEEE Transactions on information theory, Vol. 47 , NO. 2 Februry 2001.

[37] Matthew C.Davey and David MacKay, "Low-Density Parity Check Codes over GF(q)", IEEE Communication letters , Vol. 2, NO. 6, JUNE 1998.

[38] S.Y. Le Goff, "Channel capacity of bit-interleaved coded modulation schemes using 8-ary signal constellations", Electronic Letters, 14th February 2002 Vol. 38 No. 4

[39] Hongxin Song, Richard M., Richard M. Todd, and J. R. Cruz, "Applications of Low-Density Parity-Check Codes to magnetic Recording Channels", IEEE Journal on Selected Areas in communications, vol. 19, No. 5, May 2001.

[40] Daniel A. Spielman, "Linear-Time Encodable and Decodable Error-Correcting Codes", IEEE Transactions on information theory, Vol. 42 NO. 6. November 1996.

[41] Yu Kou, Shu Lin, and Marc P.C. Fossorier, "Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and new results", IEEE transactions on information theory, Vol, 47, NO. 7, Novermber 2001.

[42] Robert J, McEliece, David J. C. MacKay, and Jung-Fu Cheng, "Turbo Decoding as an Instance of Pearl's Belief propagation Algorithm", IEEE Journal on Selected Arears in Communications, VOL. 16, NO. 2 February 1998.

[43] Jinghu Chen, and Marc P.C. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of Low-Density parity Check Codes", IEEE Transactions on communications, Vol. 50, NO. 3 March 2002.