CHAPTER 5

COMPRESSED IMAGE TRANSMISSION USING LDPC CODES

5.1 INTRODUCTION

Computer communication via Internet is one of many examples of telecommunication. Communication can be seen as a transmission of information from one place (transmitter) to another (receiver). The problem in practice is that the received message may contain errors that occur due to noise in the channel or channel distortion. A way to protect the message from errors is called channel coding. The coding can help to retrieve the data even when noise is present.

Many error correction codes have been presented in the past but in recent years two classes of codes have proved themselves as the best candidates to solve the problem, namely Turbo Codes and Low Density Parity Check (LDPC) Codes. Turbo codes, invented by Berrou, Glavieux and Thitimajshim in 1993, are the first known capacity approaching error correction code that provides a powerful error correction capability when decoded by an iterative decoding algorithm.

A low-density parity check (LDPC) code is another type of error correcting code. LDPC is a method to transmit a message over a noisy transmission channel. Neither LDPC nor any other error correcting codes can guarantee perfect transmission but the probability of error can be made arbitrary small. Recently, it was shown that LDPC allows data transmission rates very close to the theoretical maximum (Shannon Limit).
Low-density parity check codes are attracting much attention to researchers. The codes are very promising and may become a standard error correction scheme in many sectors. A list of few applications that are currently being considered for are digital satellite television, ultra-high speed wireless local area networks, optical communications and hard disk drives.

The advantages of LDPC codes over other code classes, particularly Turbo codes which are their main rival are numerous. First of all, LDPC codes are not patented like Turbo codes, making them a more attractive candidate for use in commercial products. Secondly, very good algorithms for (suboptimal) decoding will be seen with linear time complexity exists. With certain structure, linear time encoding is also possible. Moreover, Turbo codes often exhibit error floors at relatively high error rates, whereas LDPC codes, when properly designed, can have significantly lower error floors. A more sophisticated and very important property of LDPC codes is that almost no undetected errors occur when decoding.

5.2 DIGITAL COMMUNICATION SYSTEM

Figure 5.1 shows a basic block diagram of a digital communication system (Proakis et al 1995). The Channel encoder codes the information sequence so that it can recover the correct information after passing through channel. Error correcting codes such as convolutional, turbo (Glavieux et al 1993) or LDPC codes are used as channel encoder.
Figure 5.1 Basic elements of a digital communication system

At the receiving end of the decoder reconstructs the original information by the knowledge of the code used by channel encoder and the redundancy contained in the received data. Channel decoders can be Viterbi (1967), turbo or LDPC decoder.

The probability of having error in the output sequence is a function of the code characteristics, the type of modulation, channel characteristics such as noise and interference level, etc. There is a trade-off between the power of transmission and the bit error rate. Researchers are trying to minimize the power consumption while maintaining a reliable communication. This arises a need for stronger codes with more error correction abilities.

In a digital wireless communication system, the purpose of the channel code is to add redundancy to the binary data stream to combat the effect of signal degradation in the channel. Ideally, channel codes should meet the following requirements:

1. Channel codes should be high rate to maximize data throughput.
2. Channel codes should have good Bit Error Rate (BER) performance at the desired Signal-to-Noise Ratio(s) (SNR) to minimize the energy needed for transmission.

3. Channel codes should have low encoder/decoder complexity to limit the size and cost of the transceivers.

4. Channel codes should introduce only minimal delays, especially in voice transmission, so that no degradation in signal quality is detectable.

These requirements are very difficult to obtain simultaneously; excellent performance in one requirement usually comes at the price of reduced performance in another.

5.3 ERROR CONTROL CODES (ERROR CORRECTING CODES OR CHANNEL CODES)

The channel encoder transforms the information message to allow it to be recovered without errors. The error correction capability is achieved by the addition of redundancy to the information message. The receiver uses the extra information added to the original message to reconstruct it. If the decoder is capable of retrieving the original message from the corrupted received message then there is no need to resend the information every time an error occurs. This saves time and bandwidth. The costs of such improvement are higher complexity of the receiver, longer time required before the reconstructed message is available and the need to send a higher quantity of data through the channel.

A coding scheme can be evaluated in many different ways from purely mathematical to more implementation oriented methods. Throughout this thesis, the codes were evaluated using a performance prospective. The
performance of the code is measured as the probability of the decoder to select for the wrong codeword at a given level of noise. The level of noise is quantified by the signal to noise ratio (SNR) often expressed in decibel (dB). The probability of having an error after the decision can be defined in two ways, the binary error ratio (BER) and the block error rate often called frame error rate (FER). The signal to noise ratio is the ratio of the energy per bit generated by the source $E_b$ over the noise spectral density $N_0$. The bit error rate is the ratio of the number of erroneous decoded bits to the total number of bits transmitted.

### 5.4 APPLICATIONS OF ERROR CORRECTING CODES

Since the focus of this thesis work is on error correcting codes, some of the applications of these codes are mentioned in Table 5.1.

**Table 5.1 Applications of error correcting codes**

| Application                  | Code                        | Comment                 |
|------------------------------|                            |                         |
| Wireless Communications       | Convolutional, Turbo, LDPC  | Random Noise            |
| Satellite downlink           |                             |                         |
| CD player Tape storage       | Reed-Solomon + cross-interleaving | Bursty channel     |
| Computer memory              | Hamming code                | -                       |
| Magnetic discs               | Fire codes                  | -                       |
| Computer networks            | CRC                         | -                       |

Satellite downlinks are generally characterized as power-limited channels. On-board batteries and solar cells are heavy and thus contribute significantly to launch costs. A communication-channel bit error rate of $10^{-5}$ is desired for many applications. There is thus a need for strong error control codes that operate efficiently at extremely low signal to noise ratios.
Convolutional codes have been particularly successful in these applications. Turbo codes and LDPC codes are other choices for these channels. Similar principles apply to the wireless communications for cell phones, lap-tops and PDAs. In order to increase the battery life, powerful codes like LDPC, turbo or convolutional codes are needed.

The channel in a CD playback system consists of a transmitting laser, a recorded disc and a photo-detector. The primary contributors to errors in this channel are fingerprints and scratches of the surface. As the surface contamination affects an area that is usually quite large compared to the surface used to record a single bit, channel errors occur in bursts when the disc is played. The CD error control system handles the bursts through cross-interleaving and through the burst error-correcting capability of Reed-Solomon codes.

Various applications exist for the error control codes in computer systems, such as memory (random access and read-only memory), disk storage, tape storage and inter-processor communication. Each of these has its unique characteristics that indicate the use of certain type of codes. Hamming codes are used for the computer memories, Fire codes for magnetic discs and Reed Solomon based system is used for the tape mass storage system. Computer networks and internet use Cyclic Redundancy Code (CRC) to detect packet errors.

5.5 TURBO CODES

Turbo codes, invented by Berrou et al (1993), are the first known capacity approach error correction code that provides a powerful error correction capability when decoded by an iterative decoding algorithm. Nowadays, the present standards of communications are incorporating turbo codes for the development of satellite links in the space communication
systems, the systems of third-generation telephony and also to increase the speed of data transmission in some versions of Wi-Fi networks.

5.5.1 Turbo Encoder

The general structure used in turbo encoders is shown in Figure 5.2. Two component codes are used to code the same input bits but an interleaver is placed between the encoders. Generally Recursive Systematic Convolutional (RSC) codes are used as the component codes but it is possible to achieve good performance using a structure like that seen in Figure 5.2 with the aid of other component codes, such as block codes (Hagenauer et al (1996) and Pyndiah et al (1997)).

Furthermore, it is also possible to employ more than two component codes. The outputs from the two component codes are then punctured and multiplexed. Usually both component RSC codes are half rate, giving one parity bit and one systematic bit output for every input bit. Then to give an overall coding rate of one half, half the output bits from the two encoders must be punctured. The arrangement that is often favoured and that is used in thesis work is to transmit all the systematic bits from the first RSC encoder and half the parity bits from each encoder. Note that the systematic
bits are rarely punctured, since this degrades the performance of the code more drastically than puncturing the parity bits.

### 5.5.2 Turbo Decoder

The general structure of an iterative turbo decoder is shown in Figure 5.3. Two component decoders are linked by interleavers in a structure similar to that of the encoder. As seen in the figure, each decoder takes three inputs: the systematically encoded channel output bits, the parity bits transmitted from the associated component encoder, and the information from the other component decoder about the likely values of the bits concerned. This information from the other decoder is referred to as a-priori information. The component decoders have to exploit both the inputs from the channel and this a-priori information.

![Figure 5.3 Turbo Decoder (Schematic)](image)

They must also provide what are known as soft outputs for the decoded bits. This means that as well as providing the decoded output bit sequence, the component decoders must also give the associated probabilities for each bit that it has been correctly decoded.
The decoder of Figure 5.3 operates iteratively and in the first iteration the first component decoder takes channel output values only and produces a soft output as its estimate of the data bits. The soft output from the first encoder is then used as additional information for the second decoder which uses this information along with the channel outputs to calculate its estimate of the data bits. Now the second iteration can begin and the first decoder decodes the channel outputs again but now with additional information about the value of the input bits provided by the output of the second decoder in the first iteration. This additional information allows the first decoder to obtain a more accurate set of soft outputs which are then used by the second decoder as a-priori information. This cycle is repeated and with every iteration, the Bit Error Rate (BER) of the decoded bits tends to fall. However the improvement in performance obtained with increasing numbers of iterations decreases as the number of iterations increases. Hence, for complexity reasons, usually only about 8 iterations are used.

Due to the interleaving used at the encoder, care must be taken to properly interleave and de-interleave the registers which are used to represent the soft values of the bits as seen in Figure 5.3. Furthermore, because of the iterative nature of the decoding, care must be taken not to re-use the same information more than once at each decoding step. For this reason, the concept of so-called extrinsic and intrinsic information was used in the original paper by Berou et al (1993) describing iterative decoding of turbo codes. Other, non-iterative, decoders have been proposed by Breiling et al (2000) which give optimal decoding of turbo codes. However the improvement in performance over iterative decoders was found to be only about 0.35 dB and they are hugely complex. Therefore the iterative scheme shown in Figure 5.3 is commonly used.
5.6 OVERVIEW OF LOW DENSITY PARITY CHECK (LDPC) CODE

Shannon et al (1948) has defined the capacity of a communication channel, which anticipates the rates at which the information consistently communicates over a noisy channel. He has recommended that the capacity is attainable with beneficial channel codes.

The foremost idea of error-correcting codes is to put in redundancy that is correlated to the information to be transmitted. Accordingly, the receiver can feat the correlation between the information bits and the redundancy bits and then correct or detect errors. The codes are classified into two major categories, explicitly block codes and convolutional codes. Hamming codes, Bose-Chaudhuri-Hocquenghem (BCH) codes; Reed-Solomon (RS) codes (Lin & Costello, 2004 and Wicker & Kim 2003) and newly rediscovered LDPC codes are the examples of block codes.

Block codes like Hamming, BCH and RS codes have structures but with limited code length. A bounded-distance decoding algorithm is usually employed in decoding block codes except LDPC codes. In general it is hard to use soft decision decoding for block codes.

The achievement of Turbo codes directed to the rediscovery of LDPC codes by Mackay (1999). They have recognized seemingly independently of the work of Gallager (1962), the advantages of linear block codes which own sparse (low-density) parity check matrices. LDPC codes were neglected for a long time since their computational complexity for the hardware technology was high at the time.

LDPC codes are a category of linear codes which caters near capacity performance on a large collection of data transmission and storage channels whilst concurrently accommodating executable decoders. LDPC
codes are rendered with probabilistic encoding and decoding algorithms. In earlier, sparse random parity check matrices were being used which established promising distance properties (Gallager 1962).

LDPC Codes are designated by a parity check $H$ matrix comprising largely 0’s and has a low density of 1’s. More precisely it can be articulated that LDPC codes have very few 1’s in each row and column with large minimum distance. In specific, a $(n, j, k)$ low-density code is a code of block length ‘$n$’ and source block length ‘$k$’. The number of parity checks is delimited as $m = n - k$. The parity check matrix weight (number of ones in each column or row) for LDPC codes can be either regular or irregular. LDPC can be regular if the number of 1’s is constant in each column or row and gets irregular with a variable number of ones in each column or row.

A regular LDPC code is a linear block code whose parity-check matrix $H$ constitutes exactly $J$ 1’s in each column and exactly $k = j (nm)$ 1’s in each row, with the code rate $R = 1 - j/k$. Low density codes are not that much optimal in the fairly contrived sense of the probability of decoding error for a known block length and it can be illustrated that the minimum rate being employed by them is bounded below channel capacity. Nevertheless, the simple decoding scheme innovation more than remunerates for these disadvantages.

LDPC codes have congenital error detection capability for instance one can merely check if the decoded sequence is an applicable code word by multiplying by the parity check matrix.

5.7 DESIGNING THE PARITY CHECK MATRIX

The Parity check matrix plays a major role in the performance of LDPC encoding/decoding. As mentioned by Gallager, this matrix should be
very sparse. It also determines the complexity of the encoder/decoder. Depending on the platform that is going to do the encoding/decoding process, this matrix can be random or structured. Random matrices are suitable for the decoders running on general purpose processors, but for dedicated hardware like FPGAs or ASICs, it is better to have a structured matrix. Structure in parity check matrix leads to a more efficient hardware representation. It also requires less memory to keep the matrix. Here, the ways to generate a sparse matrix $H$ is listed. Some of these ways are more complex than the others but they do not necessarily lead to a better code. They are as follows:

1. Start from all zero matrix of the size $(N - K) \times N$ and randomly invert some elements in the matrix to reach the resulting degrees for different nodes.

2. Generate $H$ by randomly creating weight $W_c$ columns.

3. Generate $H$ with weight $W_c$ columns and uniform row weights of $W_r$.

4. Generate $H$ with weight $W_c$ columns and uniform row weights of $W_r$ with no two columns with overlap of more than one. This condition removes all the length-four cycles which results in better performance.

5. Generating $H$ like (4) and avoiding other short cycles.

6. Generate the parity check matrix in a structured manner. For example a structure that is used in hardware design is to generate this matrix using a combination of the shifted blocks of identity matrices.

7. Generate the parity check matrix using a polynomial.
Each of the above ways has their own pros and cons, depending on the application one of them can be chosen. In this research, parity check matrix was generated in a structural way. It is more suitable for the hardware design. After designing the parity check matrix $H$, the generator matrix $G$ can be derived by solving $GH^T = 0$. Performing Gaussian elimination on the resulting matrix $G$, will put it in systematic form $G = [I | P]$. This results in the easy recovery of the information bits after decoding.

5.8 REPRESENTATIONS OF LDPC CODES

Basically there are two different possibilities to represent LDPC codes. Like all linear block codes they can be described via matrices. The second possibility is a graphical representation.

5.8.1 Matrix Representation

Consider a low-density parity-check matrix first. The matrix defined in equation (5.1) is a parity check matrix with dimension $n \times m$ for a $(8, 4)$ code.

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (5.1)$$

Two numbers describing these matrices are now defined. $w_r$ for the number of 1’s in each row and $w_c$ for the columns. For a matrix to be called low-density the two conditions $w_c << n$ and $w_r << m$ must be satisfied. In order to do this, the parity check matrix should usually be very large, so the example matrix can’t be really called low-density.
5.8.2 Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. This graph provides a complete representation of the code used and also helps to describe the decoding algorithm.

![Tanner graph](image)

**Figure 5.4** Tanner graph corresponding to the above parity check matrix H

Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes in a Tanner graph are called variable nodes (v_nodes) and check nodes (c_nodes).

Figure 5.4 is an example for such a Tanner graph and represents the same code as the above matrix H. The creation of such a graph is rather straightforward. It consists of ‘m’ check nodes (the number of parity bits) and ‘n’ variable nodes (the number of bits in a codeword). Check node $f_i$ is connected to variable node $c_j$ if the element $h_{ij}$ of H is a 1.

5.8.3 Regular and Irregular LDPC codes

A LDPC code is called regular if $w_c$ is constant for every column and $w_r = w_c \cdot (n/m)$ is also constant for every row. The example matrix from equation (5.1) is regular with $w_c = 2$ and $w_r = 4$. It’s also possible to see the
regularity of this code while looking at the graphical representation. There is the same number of incoming edges for every v-node and also for all the c-nodes. If H is low density but the number of 1’s in each row or column is not constant, the code is called an irregular LDPC code.

5.9 ENCODING OF LDPC CODES

As with any linear code, an LDPC code can be encoded by simply multiplying the vector corresponding to the information bits that is to be encoded with the generator matrix, i.e.

\[ C = uG, \quad u \in \{0,1\}^k \] (5.2)

Since LDPC codes are usually defined through their parity-check matrix, rather than their generator matrix, introducing a way to encode making sole use of the parity-check matrix would be useful. Recall that all code words satisfy

\[ cH^T = 0 \] (5.3)

Now find a decomposition of H using column permutations, given by

\[ H = [ H_p \quad H_s ] \] (5.4)

such that \( H_p \) is \( m \times m \) and invertible. The vector \( c \) can be split into a systematic and a parity part

\[ c = [c_p \quad c_s] \] (5.5)

Then combine the parity check equation with the above decomposition to get

\[ c_p H_p^T = c_s H_s^T \] (5.6)
From equation 5.6, the parity bits can be solved as

\[ c_p = c_s H^T_s (H^T_p)^{-1} \]  

(5.7)

A straightforward construction of such an encoder is to use Gaussian elimination and column permutations to bring \( H \) into an equivalent upper triangular form. Then, it can be solved for the ‘m’ parity bits using back substitution. More precisely

\[ c_{p,s} = \sum_{j=s+1}^m h_{i,j} c_{p,j} + \sum_{j=1}^k h_{i,j+m} c_{s,j} \]  

(5.8)

Unfortunately, matrix sparsity is generally not preserved by Gaussian elimination. Hence, in general the resulting equivalent matrix will be dense.

**5.10 ADDITIVE WHITE GAUSSIAN NOISE CHANNEL**

In the design of a system information is transferred through a communication medium. It is always convenient to construct a mathematical model to analyze the communication medium. This mathematical model should reflect all the characteristics of the transmission medium (Proakis et al 2002). Bearing this in mind, the fundamental AWGN physical channel will be addressed first. This channel is an additive statistical model and only the first and second moment of the process is required. The AWGN channel model is applied as

\[ r(t) = x(t) + \eta(t) \]  

(5.9)

where \( x(t) \) denotes the transmitted symbol and \( r(t) \) the received symbol. The received symbol is corrupted by the communication channel and the AWGN
process $\eta(t)$ has a probability density function (PDF) that is Gaussian distributed as

$$P(\eta(t)) = \frac{1}{\sqrt{2\Pi} \sigma_{\eta(t)}} \exp\left(-\frac{(\eta(t) - m_{\eta(t)})^2}{2\sigma_{\eta(t)}^2}\right)$$  \hspace{1cm} (5.10)$$

with mean $m_{\eta(t)}$ and variance $\sigma_{\eta(t)}^2$. The power spectral density (PSD) of the process $\eta(t)$ is a constant double-sided flat spectrum for the entire frequency spectrum and is a memoryless process. The induced entropy for the corresponding AWGN process that is given by Proakis et al (2001) as

$$H(\eta(t)) = \frac{1}{2} \log_2(2\Pi e \sigma_{\eta(t)}^2)$$  \hspace{1cm} (5.11)$$

with the natural number $e = 2.7183$. All of the above mentioned properties of the AWGN process is important in this analysis and will become more apparent.

## 5.11 DECODING LDPC CODES

The algorithm used to decode LDPC codes was discovered independently several times and as a matter of fact comes under different names. The most common ones are the belief propagation algorithm, the message passing algorithm and the sum-product algorithm.

In order to explain this algorithm, a very simple variant which works with hard decision, will be introduced first. Later on, the algorithm will be extended to work with soft decision which generally leads to better decoding results. Only binary symmetric channels will be considered.
5.11.1 Hard-decision Decoding

The algorithm will be explained on the basis of the example code already introduced in parity check matrix $H$ given in section 5.8.1. An error free received codeword would be e.g. $c = [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1]$. Let’s suppose that we have a BSC channel and the received the codeword with one error – bit $c_1$ flipped to 1.

1. In the first step all v-nodes $c_i$ send a “message” to their c-nodes $f_j$ containing the bit they believe to be the correct one for them. At this stage the only information a v-node $c_i$ has, is the corresponding received i-th bit of $c$, $y_i$. That means for example, that $c_0$ sends a message containing 1 to $f_1$ and $f_3$, node $c_1$ sends messages containing $y_1$ (1) to $f_0$ and $f_1$, and so on.

2. In the second step, every check nodes $f_j$ calculate a response to every connected variable node. The response message contains the bit that $f_j$ believes to be the correct one for this v-node $c_i$ assuming that the other v-nodes connected to $f_j$ are correct.

In other words, every c-node $f_j$ is connected to 4 v-nodes. So a c-node $f_j$ looks at the message received from three v-nodes and calculates the bit that the fourth v-node should have in order to fulfill the parity check equation. Table 5.2 gives an overview about this step.

It is important to note that this might also be the point at which the decoding algorithm terminates. This will be the case if all check equations are fulfilled. The whole algorithm contains a loop, so
another possibility to stop would be a threshold for the amount of loops.

Table 5.2 Overview over messages received and sent by the c-nodes in step 2 of the message passing algorithm

<table>
<thead>
<tr>
<th>c-node</th>
<th>received/sent</th>
</tr>
</thead>
</table>
| $f_0$  | received: $c_1 \rightarrow 1$, $c_3 \rightarrow 1$, $c_4 \rightarrow 0$, $c_7 \rightarrow 1$  
|        | sent: $0 \rightarrow c_1$, $0 \rightarrow c_3$, $1 \rightarrow c_4$, $0 \rightarrow c_7$ |
| $f_1$  | received: $c_0 \rightarrow 1$, $c_1 \rightarrow 1$, $c_2 \rightarrow 0$, $c_5 \rightarrow 1$  
|        | sent: $0 \rightarrow c_2$, $0 \rightarrow c_1$, $1 \rightarrow c_2$, $0 \rightarrow c_5$ |
| $f_2$  | received: $c_2 \rightarrow 0$, $c_5 \rightarrow 1$, $c_6 \rightarrow 0$, $c_7 \rightarrow 1$  
|        | sent: $0 \rightarrow c_2$, $1 \rightarrow c_5$, $0 \rightarrow c_6$, $1 \rightarrow c_7$ |
| $f_3$  | received: $c_0 \rightarrow 1$, $c_3 \rightarrow 1$, $c_4 \rightarrow 0$, $c_6 \rightarrow 0$  
|        | sent: $1 \rightarrow c_0$, $1 \rightarrow c_3$, $0 \rightarrow c_4$, $0 \rightarrow c_6$ |

3. Next phase: the v-nodes receive the messages from the check nodes and use this additional information to decide if the original received bit is OK. A simple way to do this is a majority vote. When coming back to the said example, each v-node has three sources of information concerning its bit. The original bit received and two suggestions from the check nodes.

Table 5.3 illustrates this step. Now the v-nodes can send another message with their (hard) decision for the correct value to the check nodes.
Table 5.3 Step 3 of the described decoding algorithm

<table>
<thead>
<tr>
<th>v-node</th>
<th>y_i received</th>
<th>messages from check nodes</th>
<th>decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_0</td>
<td>1</td>
<td>f_1 \rightarrow 0, f_3 \rightarrow 1</td>
<td>1</td>
</tr>
<tr>
<td>c_1</td>
<td>1</td>
<td>f_0 \rightarrow 0, f_1 \rightarrow 0</td>
<td>0</td>
</tr>
<tr>
<td>c_2</td>
<td>0</td>
<td>f_1 \rightarrow 1, f_2 \rightarrow 0</td>
<td>0</td>
</tr>
<tr>
<td>c_3</td>
<td>1</td>
<td>f_0 \rightarrow 0, f_3 \rightarrow 1</td>
<td>1</td>
</tr>
<tr>
<td>c_4</td>
<td>0</td>
<td>f_0 \rightarrow 1, f_3 \rightarrow 0</td>
<td>0</td>
</tr>
<tr>
<td>c_5</td>
<td>1</td>
<td>f_1 \rightarrow 0, f_2 \rightarrow 1</td>
<td>1</td>
</tr>
<tr>
<td>c_6</td>
<td>0</td>
<td>f_2 \rightarrow 0, f_3 \rightarrow 0</td>
<td>0</td>
</tr>
<tr>
<td>c_7</td>
<td>1</td>
<td>f_0 \rightarrow 1, f_2 \rightarrow 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The v-nodes use the answer messages from the c-nodes to perform a majority vote on the bit value.

4. Go to step 2.

In this example, the second execution of step 2 would terminate the decoding process since c_1 has voted for ‘0’ in the last step. This corrects the transmission error and all check equations are now satisfied.

5.11.2 Soft-decision Decoding

The above description of hard-decision decoding was mainly for educational purpose to get an overview about the idea. Soft-decision decoding of LDPC codes which is based on the concept of belief propagation, yields in a better decoding performance and is therefore the preferred method. The underlying idea is exactly the same as in hard decision decoding. Before presenting the algorithm lets introduce some notations:

- \( P_i = \Pr(c_i = 1 | y_i) \)
- \( q_{ij} \) is a message sent by the variable node \( c_i \) to the check node \( f_j \). Every message contains always the pair \( q_{ij}(0) \) and \( q_{ij}(1) \) which stands for the amount of belief that \( y_i \) is a ”0” or a ”1”.

\[ \]
- $r_{ji}$ is a message sent by the check node $f_j$ to the variable node $c_i$. Again there is a $r_{ji}(0)$ and $r_{ji}(1)$ that indicates the (current) amount of believe in that $y_i$ is a "0" or a "1".

The step numbers in the following description correspond to the hard decision case.

1. All variable nodes send their $q_{ij}$ messages. Since no other information is available at this step, $q_{ij}(1) = P_i$ and $q_{ij}(0) = 1 - P_i$.

   ![Diagram](image)

   (a) Calculation of $r_{ji}(b)$

   (b) Calculation of $q_{ij}(b)$

   **Figure 5.5 The calculation of $r_{ji}(b)$ and $q_{ij}(b)$**

2. The check nodes calculate their response messages $r_{ij}$

   $$r_{ij}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i \neq j} (1 - 2q_{ij}(1))$$

   (5.12)

   and $r_{ij}(0) = 1 - r_{ji}(0)$

   (5.13)

   So they calculate the probability that there is an even number of 1’s among the variable nodes except $c_i$ (this is exactly what $Vj\bar{u}$
means). This probability is equal to the probability $r_{ji}(0)$ that $c_i$ is a 0. This step and the information used to calculate the responses are illustrated in Figure 5.5.

3. The variable nodes update their response messages to the check $c_i/f_j$ nodes. This is done according to the following equations:

$$q_{ij}(0) = k_{ij} (1 - P_i) \prod_{j \in C_{ij}} r_{ji}(0)$$  \hspace{1cm} (5.14)$$

$$q_{ij}(1) = k_{ij} P_i \prod_{j \in C_{ij}} r_{ji}(1)$$  \hspace{1cm} (5.15)$$

whereby the Constants $K_{ij}$ are chosen in a way to ensure that $q_{ij}(0) + q_{ij}(1) = 1$. $C_{ij}$ now means all check nodes except $f_j$. Again Figure 5.5 illustrates the calculation in this step.

At this point the v-nodes also update their current estimation $\hat{c}_i$ of their variable $c_i$. This is done by calculating the probabilities for 0 and 1 and voting for the bigger one. The used equations are quite similar to the ones to compute $q_{ij}(b)$ but now the information from every c-node is used.

$$Q_{ij}(0) = K_{ij}(1 - P_i) \prod_{j \in C \setminus j} r_{ji}(0)$$  \hspace{1cm} (5.16)$$

$$Q_{ij}(1) = K_{ij} P_i \prod_{j \in C \setminus j} r_{ji}(1)$$  \hspace{1cm} (5.17)$$

If the current estimated codeword fulfills now the parity check equations the algorithm terminates. Otherwise termination is ensured through a maximum number of iterations.

4. Go to step 2.
The explained soft decision decoding algorithm is a very simple variant, suited for BSC channels and could be modified for performance improvements. Beside performance issues, there are numerical stability problems due to the many multiplications of probabilities. The results will come very close to zero for large block lengths. To prevent this, it is possible to change into the log-domain and doing additions instead of multiplications. The result is a more stable algorithm that even has performance advantages since additions are less costly.

5.12 PERFORMANCE OF LDPC CODE

The performance of the standardized LDPC code was evaluated in AWGN channel by way of BER analysis. All simulations were performed assuming codewords are transmitted over an additive white Gaussian noise (AWGN) channel with zero mean and variance $N_0 / 2$ via binary phase shift keying (BPSK) signaling. This is equivalent to transmitting the codewords over a binary symmetric channel (BSC) with $p_e = Q(\sqrt{2N_0})$.

Figures 5.6 to 5.9 show the error performance of the LDPC code with the message passing decoding as a function of the iteration number for boat, Cameraman, CT chest and CT abdomen images respectively.
Figure 5.6 Performance of LDPC coding for boat image

Table 5.4 SNR(dB) Vs BER values for boat image

<table>
<thead>
<tr>
<th>SNR(dB)</th>
<th>Uncoded</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>4 Iteration</th>
<th>10 Iteration</th>
<th>20 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.0712</td>
<td>0.0710</td>
<td>0.0709</td>
<td>0.0707</td>
<td>0.0706</td>
<td>0.0706</td>
</tr>
<tr>
<td>4.5</td>
<td>0.0631</td>
<td>0.0624</td>
<td>0.0621</td>
<td>0.0618</td>
<td>0.0512</td>
<td>0.0509</td>
</tr>
<tr>
<td>5</td>
<td>0.0523</td>
<td>0.0512</td>
<td>0.0510</td>
<td>0.0502</td>
<td>0.0482</td>
<td>0.062</td>
</tr>
<tr>
<td>5.5</td>
<td>0.0421</td>
<td>0.0418</td>
<td>0.0384</td>
<td>0.0311</td>
<td>0.0298</td>
<td>0.0254</td>
</tr>
<tr>
<td>6</td>
<td>0.0324</td>
<td>0.0289</td>
<td>0.0267</td>
<td>0.0243</td>
<td>0.0205</td>
<td>0.0124</td>
</tr>
<tr>
<td>6.5</td>
<td>0.031</td>
<td>0.025</td>
<td>0.017</td>
<td>0.011</td>
<td>0.0055</td>
<td>0.0052</td>
</tr>
<tr>
<td>7</td>
<td>0.0298</td>
<td>0.015</td>
<td>0.010</td>
<td>0.0042</td>
<td>0.0021</td>
<td>0.0019</td>
</tr>
</tbody>
</table>
Figure 5.7 Performance of LDPC coding for Lena image

Table 5.5 SNR(dB) Vs BER values for Lena image

<table>
<thead>
<tr>
<th>SNR(dB)</th>
<th>Uncoded</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>4 Iteration</th>
<th>10 Iteration</th>
<th>20 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.0978</td>
<td>0.0965</td>
<td>0.0962</td>
<td>0.0935</td>
<td>0.0927</td>
<td>0.024</td>
</tr>
<tr>
<td>4.5</td>
<td>0.0784</td>
<td>0.0773</td>
<td>0.0768</td>
<td>0.0757</td>
<td>0.0735</td>
<td>0.0732</td>
</tr>
<tr>
<td>5</td>
<td>0.0675</td>
<td>0.0592</td>
<td>0.0583</td>
<td>0.0578</td>
<td>0.0569</td>
<td>0.0558</td>
</tr>
<tr>
<td>5.5</td>
<td>0.0584</td>
<td>0.0574</td>
<td>0.065</td>
<td>0.047</td>
<td>0.0439</td>
<td>0.0492</td>
</tr>
<tr>
<td>6</td>
<td>0.0467</td>
<td>0.0467</td>
<td>0.065</td>
<td>0.062</td>
<td>0.0386</td>
<td>0.0315</td>
</tr>
<tr>
<td>6.5</td>
<td>0.0331</td>
<td>0.0291</td>
<td>0.0292</td>
<td>0.0285</td>
<td>0.015</td>
<td>0.012</td>
</tr>
<tr>
<td>7</td>
<td>0.0289</td>
<td>0.0195</td>
<td>0.0183</td>
<td>0.0082</td>
<td>0.0056</td>
<td>0.0041</td>
</tr>
</tbody>
</table>
Figure 5.8 Performance of LDPC coding for CT chest image

Table 5.6 SNR(dB) Vs BER values for CT chest image

<table>
<thead>
<tr>
<th>SNR(dB)</th>
<th>Uncoded</th>
<th>1 Iteration</th>
<th>2 Iteration</th>
<th>4 Iteration</th>
<th>10 Iteration</th>
<th>20 Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.0725</td>
<td>0.0721</td>
<td>0.0719</td>
<td>0.0705</td>
<td>0.0624</td>
<td>0.0615</td>
</tr>
<tr>
<td>4.5</td>
<td>0.0694</td>
<td>0.0624</td>
<td>0.0611</td>
<td>0.0611</td>
<td>0.0598</td>
<td>0.0587</td>
</tr>
<tr>
<td>5</td>
<td>0.0612</td>
<td>0.0612</td>
<td>0.0597</td>
<td>0.0574</td>
<td>0.0421</td>
<td>0.0454</td>
</tr>
<tr>
<td>5.5</td>
<td>0.0512</td>
<td>0.0411</td>
<td>0.0410</td>
<td>0.0410</td>
<td>0.0321</td>
<td>0.0345</td>
</tr>
<tr>
<td>6</td>
<td>0.0395</td>
<td>0.0354</td>
<td>0.0337</td>
<td>0.0201</td>
<td>0.0201</td>
<td>0.0198</td>
</tr>
<tr>
<td>6.5</td>
<td>0.0312</td>
<td>0.0245</td>
<td>0.0125</td>
<td>0.0064</td>
<td>0.0051</td>
<td>0.0053</td>
</tr>
<tr>
<td>7</td>
<td>0.0247</td>
<td>0.0167</td>
<td>0.0198</td>
<td>0.0421</td>
<td>0.0024</td>
<td>0.0012</td>
</tr>
</tbody>
</table>
It can be seen from Figures 5.6 to 5.9 and tables 5.4 to 5.7 that the performance of LDPC coding is improved with the increase of iteration number of LDPC decoder but not significantly when the iteration number is bigger than 10.
5.13 EVALUATION OF CHANNEL CODING TECHNIQUE

The performance of the standardized LDPC code was evaluated in AWGN channel by way of BER analysis. A brief comparison of LDPC codes and Turbo codes in terms of performance will be given in this section. All simulations are performed assuming codewords are transmitted over an additive white Gaussian noise (AWGN) channel with zero mean and variance $N_0/2$ via binary phase shift keying (BPSK) signaling. This is equivalent to transmitting the codewords over a binary symmetric channel (BSC) with $p_o = Q(\sqrt{2N_0})$.

Figures 5.10 to 5.14 show the bit error rates (BERs) of LDPC code under the message passing decoding with various receive diversity counts for boat, Lena, Cameraman, CT chest and CT abdomen images, respectively. Also shown are those of the Turbo code under the iterative decoding which has the comparable codeword length and code rate. All the message passing decoding schemes are assigned 20 iterations.

![BER for BPSK modulation in AWGN with receive diversity](image)

**Figure 5.10** Performance comparisons between LDPC codes and Turbo codes for boat image
Figure 5.11 Performance comparisons between LDPC codes and Turbo codes for Lena image

Figure 5.12 Performance comparisons between LDPC codes and Turbo codes for Cameraman image
Figure 5.13 Performance comparisons between LDPC codes and Turbo codes for CT Chest Image

Figure 5.14 Performance comparisons between LDPC codes and Turbo codes for CT Abdomen Image
Figure 5.10 shows the bit error rate plot of LDPC and Turbo codes for boat image. Here the error rate is compared for different receive diversity count (nRx). It can be noted that, for example, to maintain a bit error rate of $10^{-5}$, the LDPC code with nRx = 4 require 1.2 dB less $E_b/N_0$ value when compared to LDPC code with nRx = 3. Thus the transmitted power can be decreased by increasing the receive diversity count to maintain a particular bit error rate.

To compare the performance of LDPC code with turbo code, the same Bit Error Rate (BER) value of $10^{-5}$ is considered in Figure 5.10. To maintain this BER, LDPC require 0.05 dB less $E_b/N_0$ value than turbo code when receive diversity count nRx is 4. Similarly, for any bit error rate value, LDPC code requires less $E_b/N_0$ value (less transmitting power) than Turbo code. In same way, it can be noted from Figures 5.11 to 5.14 that the performance LDPC code better than the Turbo code.

5.14 CONCLUSION

In this chapter, the error performance of LDPC codes with message passing decoding and Turbo codes for various receive diversity are compared. Also, the error performance of LDPC code with message passing decoding as a function of the iteration number were analyzed. All simulations were performed assuming codewords are transmitted over an AWGN channel via binary phase shift keying signaling. It can be seen from Figures 5.10 to 5.14 that the LDPC code outperforms Turbo codes for any number of receive diversity. Also it was clear that the performance of LDPC and Turbo codes improve with increasing receive diversity count.