University of Southampton
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

A project report submitted for the award of
BEng Electronic Engineering

PROJECT SUPERVISOR: Prof. Lajos Hanzo
SECOND EXAMINAR: Dr. David Nunn

Turbo Block Coding

by

Soon Xin Ng
May 1999

Abstract

This project is concerned with the theoretical evaluation of five block codes, i.e. Bose-Chaudhuri-Hocquenghem (BCH) codes, Turbo BCH (TBCH) codes, Reed-Solomon (RS) codes, Redundant Residue Number System (RRNS) codes and Turbo RRNS (TRRNS) codes. The performance is evaluated in terms of the bit error rate (BER) curves of the codes using Binary Phase Shift Keying (BPSK) modulation scheme over Additive White Gaussian Noise (AWGN) channel for a variety of parameters.

Trellis decoding (TD) was applied to the binary BCH and TBCH codes. Algebraic decoding (AD) was applied to the non-binary RS and RRNS codes. A bit-oriented TD was applied to the non-binary RRNS and TRRNS codes. The TD for block codes was derived from that of convolutional codes and the AD for RRNS codes was derived from that of RS codes.

The coding gain tables of these codes are shown in Appendix B while the source codes to evaluate the performance of these codes are in the diskette attached at the back of this report. Appendix C shows the program listing of these source codes while the abbreviations used in this report are defined in the Glossary section.

Acknowledgements

I would like to thank my supervisor Prof. Lajos Hanzo for his encouragement and guidance. I am also grateful to Dr. David Nunn, Mr. T. H. Liew and Dr. L. L. Yang for all the support and help throughout this project. Finally, I would like to thank MARA for the financial sponsorship and my parents for their continual support during my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of coding is to combat the effect of channel impairments, and thereby aids the receiver in its decision making process. The history of channel coding or forward error correction (FEC) coding dates back to Shannon's pioneer work [3] in 1948, in which he predicted that there exist channel coding techniques achieving error-free transmission if the data rate is less than the channel capacity.

The invention of turbo codes by Berrou, Glavieux and Thitimajshima [4] in 1993 is a history breakthrough. The convolutional codes were used as the component codes and the results proved that a performance close to the Shannon limit can be achieved in practice with binary codes. Their achievement has attracted a lot of researches in this area. One of these is the implementation of turbo codes using block codes.

Several approaches can be used to estimate the performance of the coding scheme. One is to develop a computer simulation model, which closely reflects the actual transmission channel. Another approach is to develop a theoretical analysis based on the distance properties of the coding scheme.

In the first part of this project, we dealt with the theoretical evaluation of the performances of BCH codes, which is a subclass of block codes, and Turbo codes with BCH codes as the component codes.

In the second part, we dealt with the theoretical evaluation of the RS codes, which is a non-binary subclass of BCH codes. Then the theoretical performance of the RRNS codes which exhibit very similar distance properties to RS codes was evaluated. Finally, the performance of the Turbo codes with RRNS codes as the component codes was evaluated.

Chapter 2

BCH Block Codes

2.1 Introduction

Bose-Chaudhuri-Hocquenghem codes were discovered by Hocquenghem [5] in 1959 and independently by Bose and Chaudhuri [6] in 1960. These codes constitute a prominent class of cyclic block codes that have multiple-error detection and correction capabilities.

A BCH block code accepts k information databits as the information-word and produces n coded bits. A codeword is the representation of these n coded bits. If a codeword is designed to correct t random errors, the code is called a t-error-correcting code. However, it is possible to correct more than t errors if we apply soft decision Viterbi decoding algorithm. There exists a minimum Hamming distance d_{min} among the codewords and is given by [7]:

$$d_{min} = 2t + 1 \tag{2.1}$$

Hence, we denote BCH codes as BCH(n, k, d_{min}).

Table A.1 at Appendix A lists some commonly used code generators $g(x)$ for the construction of BCH codes [2]. The coefficients of $g(x)$ were presented as octal numbers arranged so that when they are converted to binary digits, the rightmost digit corresponds to the zero-degree coefficient of $g(x)$.

2.2 BCH Encoder

Since the BCH codes are cyclic codes, their encoders and decoders can be implemented using shift-register circuits [8, 7, 9]. The codes can be encoded either non-systematically or systematically. Systematic BCH codes are found to perform better than the non-systematic BCH codes. This chapter concerns about the systematic binary BCH codes. The codeword of a systematic code contains a copy of the information-word.

The systematic codes have the generator polynomial, $g(x)$ of the form:

$$g(x) = g_0 + g_1x + g_2x^2 + ... + g_{n-k-1}x^{n-k-1} + g_{n-k}x^{n-k}. \tag{2.2}$$

The generator polynomial, $g(x)$ appends $(n - k)$ parity check bits to information-word of k bits to produce a codeword of n bits. Hence, the coding rate for BCH codes is:

$$R = \frac{k}{n}. \tag{2.3}$$

The encoder employs a shift register having $(n-k)$ stages as depicted in Figure 2.1, where $\otimes$ represents multiplication and $\oplus$ represent modulo two additions.

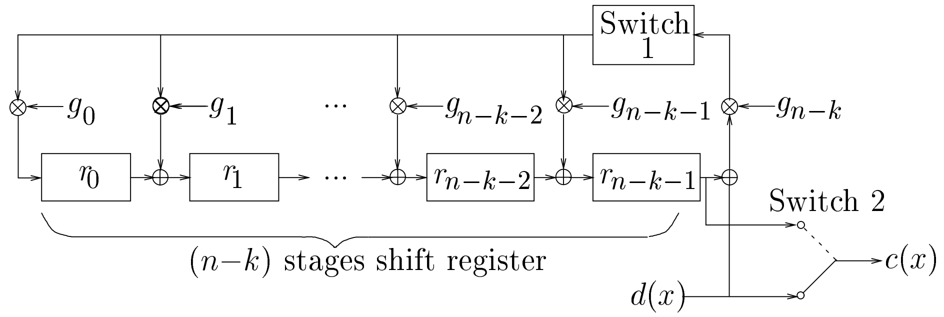

Figure 2.1: Systematic encoder for BCH codes having $(n-k)$ shift register stages ©[1] Liew 1999.

The encoding procedures are as described bellow:

1. Switch 1 is closed during the first k shifts, to allow for the information-word, $d(x)$ to be shifted into the $n-k$ stages of the shift register.
2. At the same time, Switch 2 is in the down position to allow for the information-word, $d(x)$ to be copied directly to the codeword, $c(x)$.
3. After the k^{th} shift, Switch 1 is opened and Switch 2 is moved to the upper position. Now the shift register contains the $(n-k)$ parity bits.
4. The remaining $n-k$ shifts clear the shift register by transferring its content to the codeword, $c(x)$.
5. Hence $(n-k)$ parity bits were appended to the k bits information-word to produce an n bits codeword.

2.3 State and Trellis Diagrams

Let us consider BCH(7,4,3) with generator $g = 13_{octal}$ as an example to illustrate the state and trellis diagram representations of BCH codes. There are a few points worth noting:

- The encoding process always starts at the all zero state and ends at the all zero state.
- There is always one output bit produced following a clock pulse.
- For the first k (which is 4 for this example) shifts, the output bits are the same as the input bits.
- After the k^{th} shift, the parity bits of the shift register are shifted to the output bit.
- The number of states is equal to 2^{n-k}, which increases exponentially when $n-k$ increases.

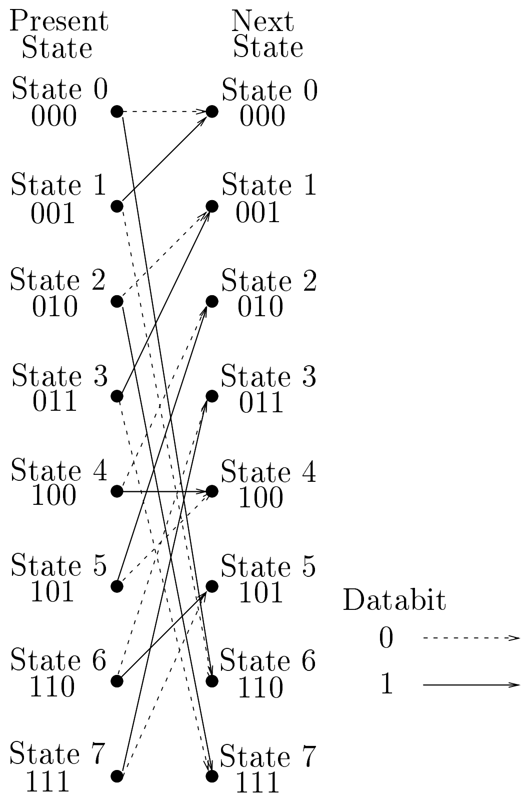

Figure 2.2: State Transition Diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states ©[1] Liew 1999.

The state transition of the encoding process for BCH(7,4,3) is shown in Figure 2.2. It shows all possible state transitions at any states when the register is in a particular state. The branch emanating from the present state to the next state indicates the state transition. The solid line is the transition initiated by a databit of logical 1, whereas the broken line is due to the databit of logical 0. The number of branches emanating form the present state is 2 for our binary BCH codes, which corresponding to the number of possible input pattern. The output bits are the same as the input bits for the first k state transitions. As explained earlier, $n - k$ parity bits will be appended to the output bits after k transitions to form the codeword.

The state diagram corresponding to the state transition diagram is portrayed in Figure 2.3. It consists of a total of $2^{n-k} = 8$ states connected by all the possible transitions depicted by Figure 2.2. By using the state diagram in Figure 2.3, we can encode the databits, $d = 1\ 0\ 0\ 1$ by the following procedure:

The first databit is a logical 1, hence the state changes from 000 to 110, as illustrated by the solid branch emanating from state 000 in Figure 2.3. The encoder output is the same as the input databit, which is a logical 1. At the next instant, the present state becomes 110 and the new databit is logical 0. This causes the state transition from 110 to 011. The encoding cycle is repeated for subsequent databits, which changes the states. By following the change of states throughout the first k cycles of the encoding process, a particular path associated with states $000 \xrightarrow{1} 110 \xrightarrow{0} 011 \xrightarrow{0} 111 \xrightarrow{1} 011$, can be observed. After the k^{th} cycle, the state changes correspond to shifting out the parity bits from the shift register. In our example, the parity bits are 011 which are the content of the shift register at k^{th} cycle. In the following cycle, the parity bits are shifted to the right. The rightmost bit of the parity

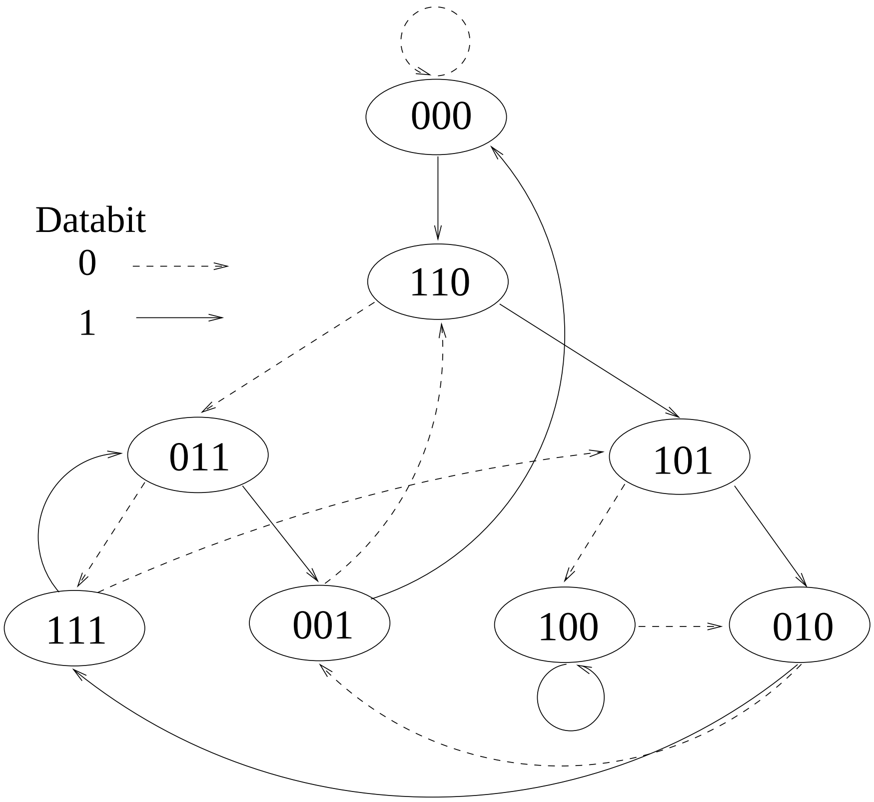

Figure 2.3: State Diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states ©[1] Liew 1999.

bits is shifted out to become the output bit and the leftmost bit is filled with logical 0.

As a result, the state changes are $011 \xrightarrow{1} 001 \xrightarrow{1} 000 \xrightarrow{0} 000$. The whole encoding process can be associated with state transitions of $000 \xrightarrow{1} 110 \xrightarrow{0} 011 \xrightarrow{0} 111 \xrightarrow{1} 011 \xrightarrow{1} 001 \xrightarrow{1} 000 \xrightarrow{0} 000$. Therefore the codeword is 1001110_{bin} encoded from information-word 1001_{bin} using generator $g = 13_{octal}$.

Another representation of the encoding process is the trellis diagram shown in Figure 2.4. This is formed by concatenating the consecutive instants of the state transition diagram of Figure 2.2 starting from the all zero state. The diagram illustrates all the possible 2^{n-k} paths for BCH(7,4,3) code. The state transitions between adjacent columns are drawn by a solid line if the databit is a logical 1, or drawn by a broken line if the databit is a logical 0.

Initially, there is only one state that is the all zero state (state a). The symbol signaling instants are indexed by the integer T. The number of possible state increases at each arrival of a databit, i.e. equal to 2^T. However, the maximum number of possible state is limited by 2^{n-k}, which is reached at the instant $T = n - k$. From instant $T = n - k$ until $T = k$, the number of possible states is constant, which is $2^{n-k} = 8$ states for BCH(7,4,3). This is because the state transition begins to converge back to the all zero state after $T = n - k$ and the state transition stop diverging from the all zero state at instant $T = k$ onwards. Hence the number of possible states is reduced by a factor of 2 at every instant after $T = k$. At the end, i.e. at instant $T = n$, there is only one state left that is the all zero state.

Therefore, BCH codes have limited trellis paths that are always converge back to the all zero state at or before instant $T = n$. This is a significant difference from the convolutional codes [10], where

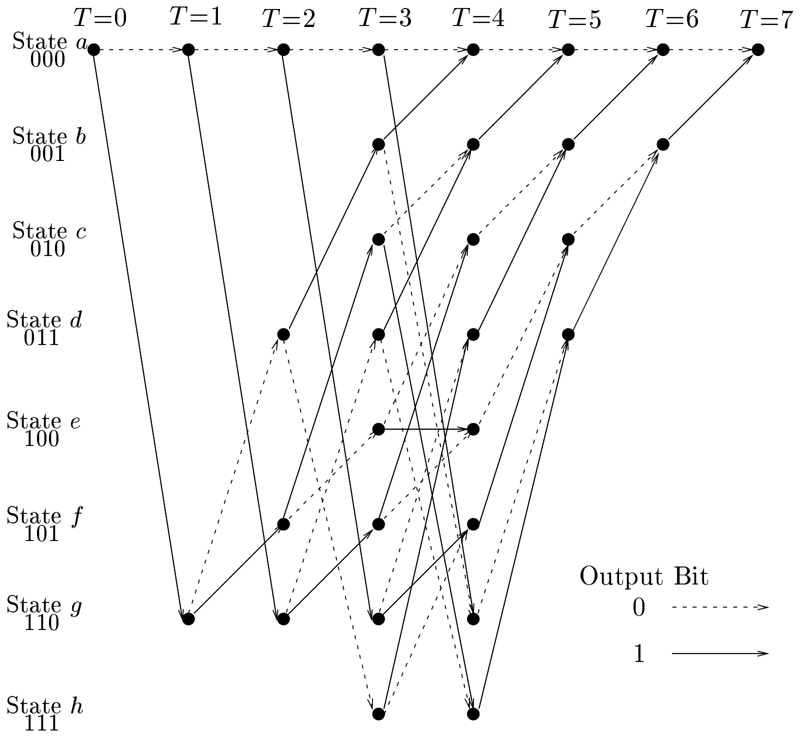

Figure 2.4: Trellis Diagram for the BCH(7,4,3) code having $2^{n-k} = 8$ states and $n+1 = 8$ consecutive stages ©[1] Liew 1999.

the trellis paths of convolutional codes are unlimited and there is never an instant that has only one state, i.e. all zero state, after instant $T = 0$.

We will investigate the performance of Trellis decoding [11, 12] of BCH block codes using the Viterbi Algorithm.

2.4 Viterbi Algorithm

The Viterbi Algorithm (VA) was proposed by Viterbi in 1967 [10]. It searches all the possible paths in the trellis and the Hamming or Euclidean distances from the received sequence at the decoder input are compared. Hamming distance is equal to the number of bits that are different between the possible transmitted sequence and the received sequence. VA selects the path with the smallest Hamming distance and the associated information-word are regenerated. The most likely path from all the paths in the trellis is selected, hence it is known as a maximum likelihood sequence estimation.

There are two ways of using the Viterbi Algorithm depending on the demodulator type. The first one is called the hard decision (HD) decoding and is used with demodulators whose output is either 0 or 1. The second one is soft decision (SD) decoding and is used with demodulators that provide an estimation of confidence in the result. Details on Viterbi decoding can be found in [7, 1].

2.5 Distance Properties of BCH Codes

The resistance of a code to errors is closely linked to the Hamming distance between the encoded sequences. If the Hamming distance of two codewords are small, i.e. the codewords are quite similar, the corruption of a few bits may lead to an incorrect decoding. On the other hand, if the encoded sequences have very big Hamming distance, more bits will have to be corrupted to cause decoding errors. Thus the BER performance of a code depends on its distance properties.

Since the BCH encoding involves modulo-2 linear operations on the information sequence, BCH codes are linear and therefore the distance separations of all the paths from a code sequence in trellis is independent of which particular code sequence is considered. Hence, for the sake of simplicity, we assumed that the all zero sequence is transmitted. If there is no channel error, the received sequence is 0 and the next state transition is again at the all zero state, which is state a as shown in Figure 2.4. When channel noise corrupted the original sequence, the path will diverge from the all zero sequence. It will take some instances before the path converges back to the all zero state again.

Since all the paths of the trellis for BCH code will eventually converge back to the all zero state at instant $T = n$ as shown in the Figure 2.4, the maximum instant that the incorrect paths will leave the all zero state is therefore equals to n. There is only one correct path, which is the all zero path, and $2^k - 1$ incorrect paths in the trellis of BCH codes. Our aim is to list down all the incorrect paths and to calculate their corresponding Hamming distances. Then these information will be used to compute the probability of error decoding.

Let us denote the number of instant the incorrect path leaving the all zero path as the exponent of the factor J. The factor H is introduced such that the exponent of H represents the number of incorrect bit of the information-word. A logical 1 is considered as an incorrect bit since we considered the all zero sequence as the transmitted sequence. Since systematic BCH codes append $n - k$ parity bits to the k databits, the exponent of H is simply the number of 1's of an incorrect path in the trellis before instant $T = k$. The exponent of the factor D denotes the Hamming distance of the received codeword, which is the number of 1's of the codeword. Since the length of the codeword of BCH block codes is n bits, the maximum Hamming distance is therefore equals to n.

2.5.1 Transfer Function

Transfer function of a code is the representation of all the incorrect paths with their respective J, H and D information. Let us consider BCH(7,4,3) as an example to illustrate the transfer function.

From Table 2.1 or from the trellis diagram portrayed in Figure 2.4, we can represent the transfer function as below:

$$\begin{aligned} T_{BCH(7,4,3)}(J,H,D) = {} & 2J^4H^1D^3 + J^7H^1D^3 + J^4H^2D^3 + 2J^6H^2D^3 + J^4H^3D^3 \\ & + J^6H^1D^4 + J^5H^2D^4 + J^6H^2D^4 + J^7H^2D^4 + 2J^5H^3D^4 \\ & + J^7H^3D^4 + J^7H^4D^7 \end{aligned} \tag{2.4}$$

where the first term of the transfer function $T_{BCH(7,4,3)}(J,H,D)$ indicates that there are two incorrect paths having Hamming distance of 3 bits (exponent of D) from the all zero sequence that merges back to state a after 4 instants (exponent of J), and there is one erroneous information databit (exponent

Index	*databits*	*codeword*	J	H	D
1	0001	0001011	4	1	3
2	0010	0010110	4	1	3
3	0011	0011101	5	2	4
4	0100	0100111	6	1	4
5	0101	0101100	4	2	3
6	0110	0110001	6	2	3
7	0111	0111010	5	3	4
8	1000	1000101	7	1	3
9	1001	1001110	6	2	4
10	1010	1010011	7	2	4
11	1011	1011000	4	3	3
12	1100	1100010	6	2	3
13	1101	1101001	7	3	4
14	1110	1110100	5	3	4
15	1111	1111111	7	4	7

Table 2.1: Transfer function of BCH(7,4,3).

of H). The minimum distance between two paths of the code is called the minimum free distance denoted by d_{free}.The d_{free} of BCH(7,4,3) is 3, which is the exponent of D in the first term of Equation 2.4.

The factor J is to determine the number of branches spanned by the paths. This factor can be suppressed by setting $J = 1$ in Equation 2.4 to yield:

$$\begin{aligned} T_{BCH(7,4,3)}(1, H, D) = {} & 3H^1D^3 + 2H^2D^3 + H^3D^3 + H^1D^4 + 3H^2D^4 \\ & + 3H^3D^4 + H^4D^7 \end{aligned} \tag{2.5}$$

which does not depend on the path length J. The first and second terms of Equation 2.4 are combined together to form the first term of Equation 2.5 because they are having the same H and D information. Furthermore, H is set to 1 to yield Equation 2.6 in order to ignore the number of erroneous information databits associated with the path.

$$T_{BCH(7,4,3)}(1, 1, D) = 7D^3 + 7D^4 + D^7 \tag{2.6}$$

The Hamming distance between an incorrect and correct path is the weight, i.e. number of 1, of the incorrect path. If d denotes the Hamming distance of a weight-d path, Equation 2.6 can be express as:

$$T_{BCH(7,4,3)}(D) = \sum_{d=min}^{n} A_d D^d \tag{2.7}$$

where the coefficient A_d is the number of incorrect paths of weight-d regardless of the information databits on the path.

2.5.2 Weight Distribution Function

Weight distribution determines the number of information bit errors at the decoder output if the weight-d path is incorrectly selected. The weight distribution $W_{BCH(7,4,3)}(d)$ of BCH(7,4,3) code is characterized by the total number of erroneous information databits of all weight-d trellis paths and is obtained by differentiating $T_{BCH(7,4,3)}(1, H, D)$ with respect to H and setting $H = 1$:

$$W_{BCH(7,4,3)}(d) \quad = \quad \frac{d(T_{BCH(7,4,3)}(1, H, D))}{dH}|_{H=1} \tag{2.8}$$

Differentiating $T_{BCH(7,4,3)}(1, H, D)$ effectively yields the multiplication of the total number of incoming paths with the number of incorrect decoded bits in all weight-d paths. Hence, $W_{BCH(7,4,3)}(d)$ could be obtained by substituting Equation 2.5 into 2.8 to yield:

$$W_{BCH(7,4,3)}(d) = 12D^3 + 16D^4 + 4D^7. \tag{2.9}$$

Generally, the weight distribution function of a BCH(n, k, d_{min}) code can be described by:

$$W_{BCH(n,k,d_{min})}(d) = \sum_{d=d_{min}}^{n} W_d D^d \tag{2.10}$$

where the coefficient W_d is defined as the total number of erroneous information databits for all weight-d paths. The minimum distance among all these paths is d_{free} and hence $W_d = 0$ for $d < d_{min}$.

2.6 Error Decoding Probability

The bit error rate (BER) can be computed for both HD and SD decoding using the weight distribution function by knowing the Signal to Noise Ratio (SNR). SD decoding will be discussed for BCH codes since its performance is better than the HD BCH codes. We will consider Binary Phase Shift Keying (BPSK) modulation scheme and Additive white Gaussian noise (AWGN) channel to evaluate the performance of the codes.

AWGN [13] is the noise generated in the receiver when the transmission path is ideal. The noise is assumed to have a constant power spectral density over the channel bandwidth, and a Gaussian amplitude probability density function (PDF). AWGN channel alters the transmitted signals only to the extent that it introduces an attenuation factor and a phase shift, in addition to the AWGN.

The modulator performs the function of mapping the digital sequence into waveforms. In BPSK, the phase of the transmitted radio carrier is 0^o when the data is a binary 1, and it changes to 180^o when the data is a logical 0.

2.6.1 Soft Decision Decoding Theory

The demodulated signal of the SD decoding is quantised and conveyed to the decoder. We assume that the voltage levels at the demodulator output are statistically independent from each other. To simplify the analysis we transmit the all zero sequence, knowing that our results are applicable to other transmitted sequences.

The transmitted signal is BPSK modulated and the coherent demodulation is used at the receiver. The sampled demodulated signal is applied directly to the decoder. When the Viterbi decoding method is applied, the probability of an incorrect decoding $P_{ICD}(d)$ is given by [7]:

$$P_{ICD}(d) \quad = \quad P\left(\sum_{p=1}^{d} r_p > 0\right) \tag{2.11}$$

where the index p is the position of those d bits in which the two paths differ, and r_p represents the demodulator output corresponding to one of these d bits. For an AWGN channel, the signal r_p is a Gaussian random variable.

Another representation of the incorrect decoding probability is derived by using the complimentary error function $erfc(x)$:

$$P_{ICD}(d) \quad = \quad \frac{1}{2} erfc\left(\sqrt{\frac{E_b R d}{N_o}}\right) \tag{2.12}$$

where the symbol E_b/N_o is the energy per information bit over noise ratio or the SNR of the channel per information databit:

$$\frac{E_b}{N_o}(dB) \quad = \quad SNR(dB) - 10log_{10}R. \tag{2.13}$$

R is the coding rate and d is the Hamming distance of a path. Equation 2.12 gives the probability of incorrect decoding from the pairwise comparisons into a path of distance d from the all zero path. We can bound the error probability by summing the incorrect decoding probabilities of the pairwise comparisons over all possible paths that merge with the all zero path at a given node. This is called the union bound of the first-event probabilities P_{FE}:

$$\begin{aligned} P_{FE} \quad &< \quad \sum_{d=d_{min}}^{n} A_d P_{ICD}(d) \\ &= \quad \frac{1}{2} \sum_{d=d_{min}}^{n} A_d erfc\left(\sqrt{\frac{E_b R d}{N_o}}\right) \end{aligned} \tag{2.14}$$

where A_d is the coefficient of the transfer function $T(H, D)$, representing the number of non-zero paths with weight-d. The probability of post-decoding bit errors p_{bp} is obtained by weighting the first-event error probability term with the weight of the incorrect paths, i.e. number of 1 in the non-zero information sequence. Hence, the union bound of the post-decoding bit error probability is given by:

$$\begin{aligned} p_{bp} \quad &< \quad \frac{1}{k} \sum_{d=d_{min}}^{n} W_d P_{ICD}(d) \\ &< \quad \frac{1}{2k} \sum_{d=d_{min}}^{n} W_d erfc\left(\sqrt{\frac{E_b R d}{N_o}}\right) \end{aligned} \tag{2.15}$$

2.7 Simulation Results

A program called 'BCH performance evaluator' was written in C to generate the distance properties of the BCH codes using BPSK modulation scheme over AWGN channel. The union bounds of the codes for SD Trellis decoding were generated using the distance properties.

The notations used in the graphs are:

1. D terms, which is the Hamming distance or number of error bits of a codeword.
2. Weight of D terms, which is the number of trellis path having Hamming distance of D.
3. BER, which is the Bit Error Rate of the code at a certain SNR.
4. R, which is the coding rate.
5. E_b/N_o, which is the bit energy per noise ratio.

The simplest BCH code is the BCH(7,4,3) code generated by $g(x) = 138_{octal}$ with d_{min} of 3, which was used as an example in Section 2.5. The corresponding weight function and the union bound are shown in Figure 2.5. There are only three D terms with Hamming distance of 3, 4 and 7. The uncoded BER is plotted as a reference to measure the coding gain of the code. The coding gain of BCH(7,4,3) code at BER of 10^{-3} is $6.78 - 5.39 = 1.39$ dB.

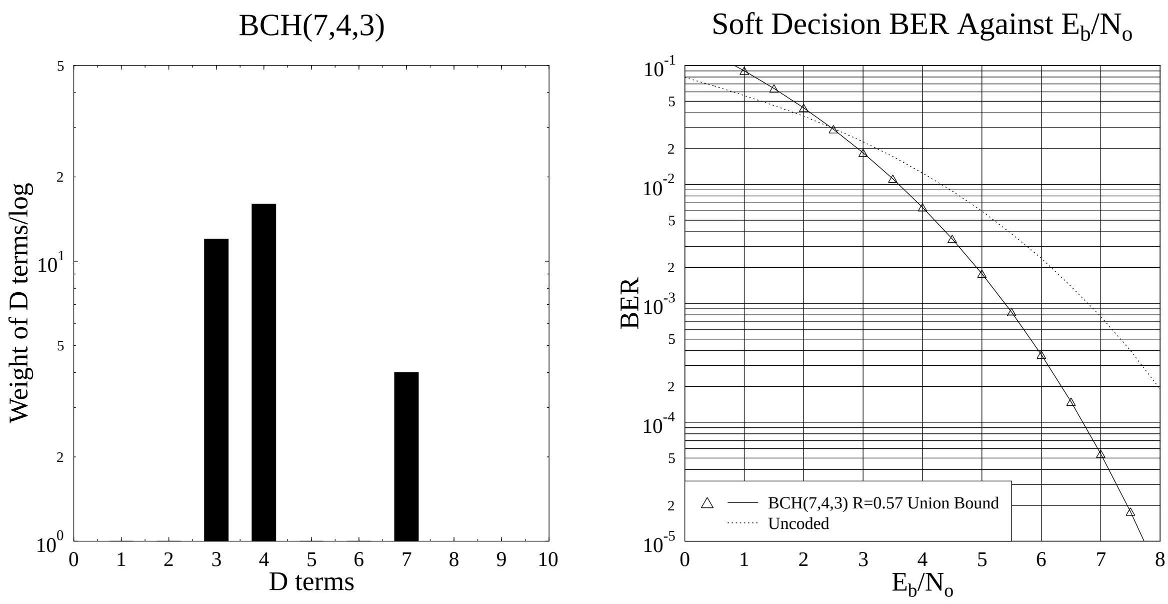

Figure 2.5: Weight function and union bound for systematic BCH(7,4,3).

2.7.1 Parameters Affecting the Code performance

There are some parameters that affect the BER performance of BCH codes such as:

- Generator Polynomials, $g(x)$
- Length of Information-word, k
- Length of Codeword, n

- Code rates, R

Some of these parameters are interlinked. For example the BCH code rate, R is correlated with the length of information-word, k and the length of codeword, n by the Formula 2.3.

2.7.1.1 Effects of Generator Polynomials

A family of BCH codes has a fix codeword length, n but different information-word length, k depending on the generator polynomials, $g(x)$ applied. However, for a particular BCH(n, k, d_{min}) code, the optimum generator polynomials were already defined in [2]. Varying the k of a family code will change the optimum generator polynomial. Hence the effect of $g(x)$ can be observed in a code family of different k as described in the following section.

2.7.1.2 Effects of Information-word Length

The shift register length required to generate the BCH codes decreases with the increase of the information-word length, k as can be noticed from Table A.1.

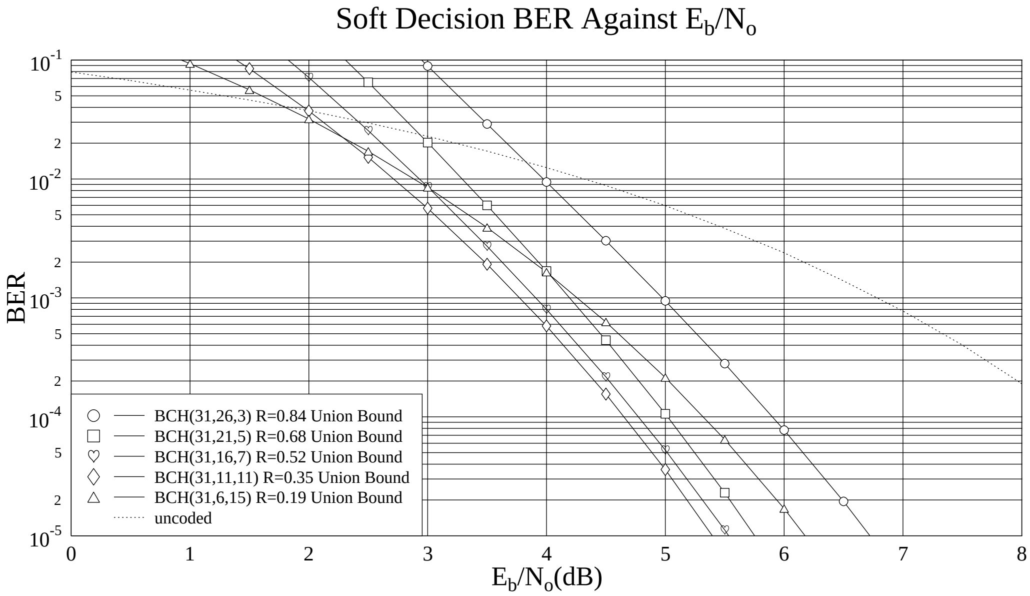

Figure 2.6: Union bounds of systematic SD decoding for BCH(31,26,3) $R = 0.84$, BCH(31,21,5) $R = 0.68$, BCH(31,16,7) $R = 0.52$, BCH(31,11,11) $R = 0.35$ and BCH(31,6,15) $R = 0.19$. This shows the effect of increasing information-word length, k while decreasing the coding rate, R.

As portrayed in Figure 2.6, the code of a family with smaller k performs better. For example, BCH(31,21,5) performs better than BCH(31,26,3) at BER of 10^{-5} for about one dB. However, there is a limit for the smallest k that produces good BER performance. For the code of $k = 6$ in the BCH$(31, k, d_{min})$ family, the performance is worse than the codes with $k = 11, 16$ and 21 at BER of

10^{-5} as evidenced by Figure 2.6. Hence the performance of a BCH family codes improves when k decreases but only up to a certain limit.

2.7.1.3 Effects of Codeword Length

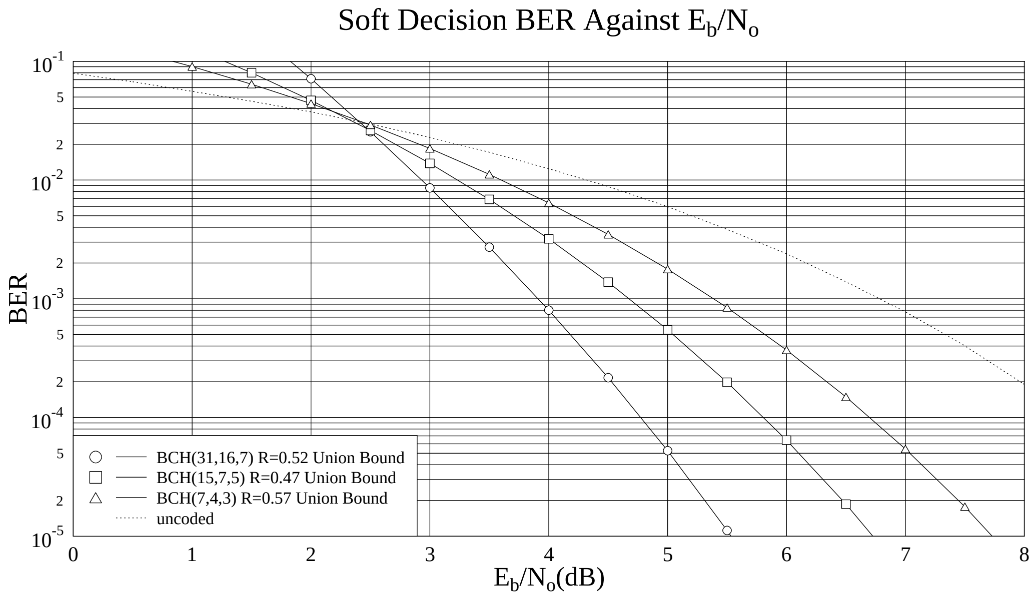

Figure 2.7: Union bounds of systematic SD decoding for BCH(31,16,7) $R = 0.52$, BCH(15,7,5) $R = 0.47$ and BCH(7,4,3) $R = 0.57$. This shows the effect of increasing the codeword length, n.

From Figure 2.7 we can learn that the performance of BCH codes improved when the codeword length, n increases although the coding rates are very close. At the coding rate of about 0.5, there is an improvement of about 2.2 dB at BER of 10^{-5} when n increases from 7 to 31. However, the shift register of BCH(31,16,7) is much longer than the one of BCH(7,4,3) since there are more parity bits to be appended to the information-words of BCH(31,16,7).

Therefore, by using a bigger n code the performance could be further improved by the cost of a longer encoder structure.

2.7.1.4 Effects of Coding Rate

Codes with high coding rate, R is more efficient in transmission. The coding rate of a BCH codes family could be modified by varying the length of the information-word, k. High coding rate can be achieved by increasing the k.

From Figure 2.6 we notice that when the coding rate is decreased, the BER performance is improved, i.e. the BER curve stretches further to the left. As shown in the figure, the BER performance improves as the coding rate decreases from $R = 0.84$ to $R = 0.35$. However, at $R = 0.19$ the BER performance is worsened.

Let us examine this property from the perspective of smaller coding rates that are less than 0.5. From Figure 2.8 we found that the BER performance of the BCH$(63, k, d_{min})$ codes improves with the increase of the coding rate from $R = 0.11$ to $R = 0.38$.

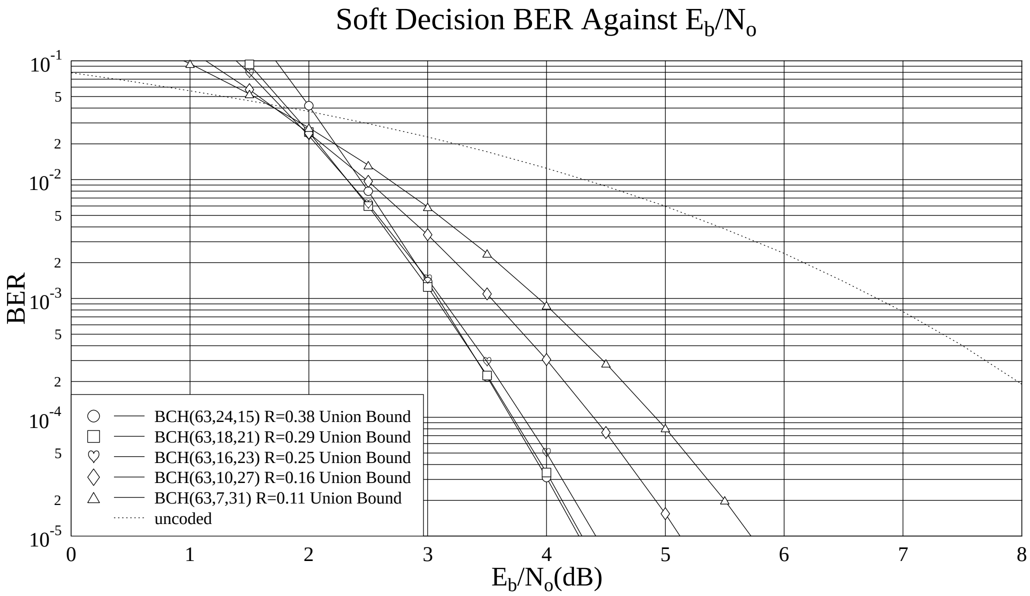

Figure 2.8: Union bounds of systematic SD decoding for BCH(63,24,15) $R = 0.38$, BCH(63,18,21) $R = 0.29$, BCH(63,16,23) $R = 0.25$ and BCH(63,7,31) $R = 0.11$. This shows the effect of increasing the coding rate, R to half rate.

Hence we could conclude that the BER performance is not optimum at coding rates near 1 and 0. The optimum BER performance is at coding rate around 0.5.

2.7.2 Comparisons of the Results from Computer Modelling Simulation

In this section, we compare some of the theoretical results with the computer modelling simulations in order to confirm the validity of the theoretical results.

The BER computed is not an exact value. It is the union bound, which means that the computation is based on all the errors without subtracting cases that overlap. Therefore, the results are higher than the real value.

From Figure 2.9 we notice that the theoretical bounds are worse than the modelling simulations at low SNR. This is because at low SNR all deviating paths can occur quite easily in theory even for the ones with high Hamming distance. Therefore, at low SNR the overlapping is important and degrades dramatically the theoretical results as compare to the simulation results. However, at high SNR the theoretical results are accurate. From Figure 2.9 the union bound and its corresponding simulation are matched at high SNR.

From Figure 2.10 we notice that there are slight deviations at high E_b/N_o between the union

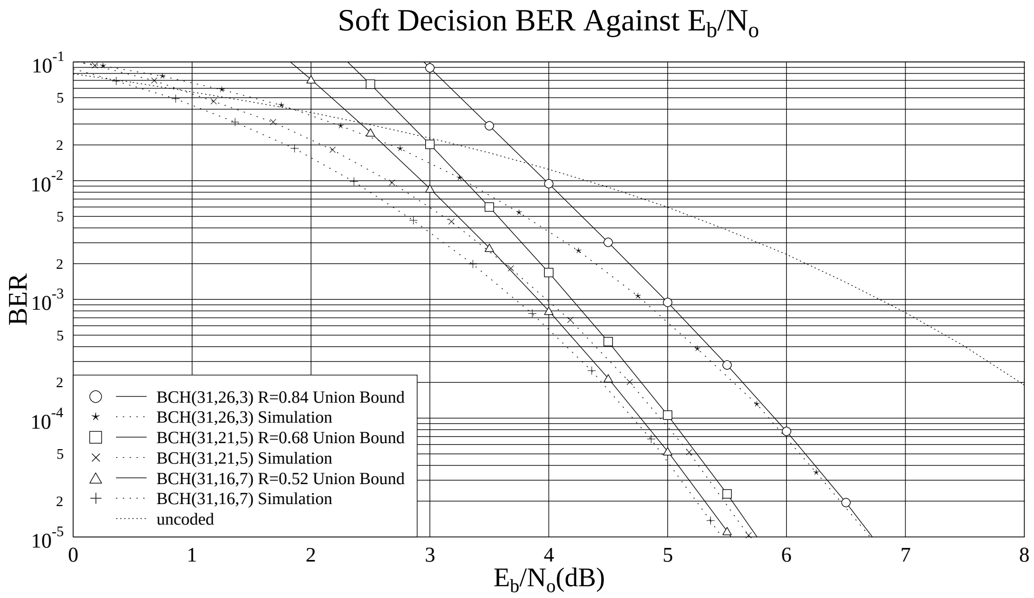

Figure 2.9: Comparison between Union bounds and modelling simulations for BCH(31,26,3), BCH(31,21,5) and BCH(31,16,7). This shows the matching of the theoretical and the simulation results at high SNR.

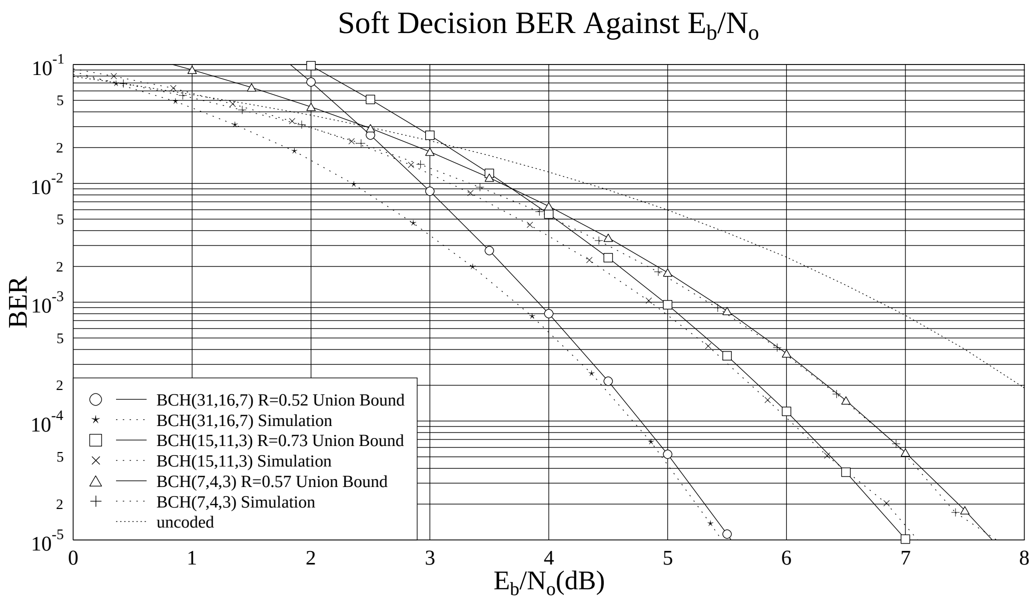

Figure 2.10: Comparison between Union bounds and modelling simulations for systematic SD decoding of BCH(7,4,3), BCH(15,11,3) and BCH(31,16,7). This shows the deviations between the theoretical and the simulation results at high SNR for small sample size.

bound and the modelling simulation for BCH(15,11,3) code such that the simulation result is slightly greater than the union bound. This is because the sample size of the transmitted symbol in the modelling simulation might not be enough at high E_b/N_o to keep the fractional root mean square, rms measurement below a specific value [14]. We can compute the modelling simulations using bigger sample size to get a more accurate result, but this would require greater computational time. This is unnecessary, because if we compute several modelling simulations using this moderate sample size, the accuracy of the average simulation result will be improved.

Chapter 3

Turbo Block Coding

3.1 Introduction

Turbo coding [4, 15, 16, 17, 18] is a parallel-concatenated coding scheme consists of two simple constituent systematic encoders linked by an interleaver. It is a new form of channel coding that is proved to perform near the Shannon [3] limit. Generally, Recursive Systematic Convolutional (RSC) codes are used as the component codes. However, turbo codes using block codes as the component codes perform better at high coding rate. We will evaluate the performance of the standard turbo codes structure using BCH block codes as the component codes.

3.2 Turbo Encoder

The structure of the turbo encoder is shown in Figure 3.1. Two BCH systematic encoders are used as the component codes with an interleaver place before the second BCH encoder.

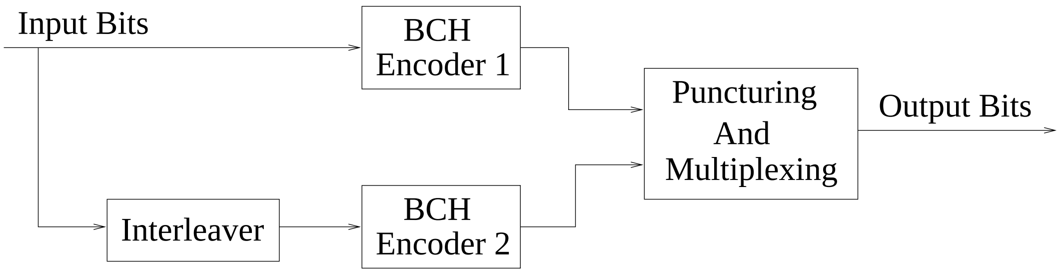

Figure 3.1: Turbo Encoder Schematic ©[1] Liew 1999.

The interleaver is used to permute the information databits such that the two encoders are operating on the same set of input bits, but different input sequences. Therefore the information of the databits is spread into more codewords. Hence turbo codes may be more resistant to corruption. There are four interleaving techniques, which are block interleaving, diagonal interleaving, inter block interleaving and convolutional interleaving. This project concerns about the block interleaving technique. The interleaver size, N is the product of the number of column with the number of row of the interleaver:

$$N = col \times row \ . \tag{3.1}$$

N must be dividable by the information-word length, k so that it could be fully filled. Hence, k of the component codes determines the choice of the interleaver.

The codeword produced consists of the information-word followed by the parity check bits generated by both encoders. The parity bits are then punctured and multiplexed. Puncturing allows us to obtain a high-rate code by periodically deleting some of the coded bits from a low-rate encoder output. Hence applying puncturing method can vary the coding rate and the BER performance of the turbo codes.

3.3 Distance Properties of Turbo Codes

Since we are using BCH codes as the component codes of the Turbo BCH (TBCH) codes, the distance properties of the TBCH codes are very similar to the one for BCH codes.

A TBCH(7,4,3) code with interleaver of size $N = 4 \times 2$ has factor $N/k = 2$, means that the interleaver can contain 2 information-words. The length of the turbo information-word, K_L is the size of the interleaver, N. However, since there are 2 BCH encoders each produces $n - k = 3$ parity bits, the length of the total parity bits, P_L of the codeword is given by:

$$P_L = \frac{N}{k} \times 2(n-k) \tag{3.2}$$

which is $2 \times 2 \times 3 = 12$ bits in this example. The length of the turbo codeword, L is given by:

$$\begin{aligned} L &= K_L + P_L \\ &= N + \frac{N}{k} \times 2(n-k). \end{aligned} \tag{3.3}$$

which is $8 + 12 = 20$ bits. The coding rate for non punctured turbo codes is given by:

$$R = \frac{k}{2n-k} \tag{3.4}$$

which is independent of the interleaver size N.

We define the parity puncturing weight, pw as the number of desirable parity bits to be appended to the information-word to form the codeword. These are the number of parity bits not to be deleted out of the total P_L parity bits generated. The coding rate for punctured turbo code is given by:

$$R = \frac{N}{N + pw.} \tag{3.5}$$

The factors J, H and D were used with the same definitions as in Section 2.5 to evaluate the distance properties of turbo codes. Hence the transfer function and the weight distribution function of turbo codes were evaluated exactly the same as in Section 2.5.1 and 2.5.2.

3.4 Error Decoding Probability

The union bounds of turbo codes were computed using BPSK modulation scheme over AWGN channel. The union bound formula of the post-decoding BER for soft decision (SD) Viterbi decoding is given by:

$$p_{bp} \quad < \quad \frac{1}{2N} \sum_{d=d_{min}}^{L} W_d erfc\left(\sqrt{\frac{E_b R d}{N_o}}\right) \tag{3.6}$$

where N is the interleaver size and L is the codeword length of turbo codes. These are the only factors that are different from the Formula 2.15, the SD decoding p_{bp} formula of BCH codes.

3.5 Simulation Results

A program called 'TURBO performance evaluator' was written in C to obtain the union bounds of the TBCH. There are many parameters that effect the performance of the turbo codes. Some of them are interlinked. The same graph notations as describe in Section 2.7 are used.

3.5.1 Effects of Distance Spectra

A distance spectrum is the p_{bp} for a path of distance d, given by:

$$p_{bp}(d) \quad < \quad \frac{1}{2N} W_d erfc\left(\sqrt{\frac{E_b R d}{N_o}}\right) \tag{3.7}$$

The union bound of a code is the summation of all the distance spectra of different Hamming distance, d which is denoted as D in the plots. These spectra have different degree of influence to the Union bound.

3.5.1.1 Minimum Hamming Distance

In turbo codes, the free distance, d_{free} is defined as the minimum Hamming distance, d_{min} of all possible codewords. The weight distribution for the free distance, $W_{d_{free}}$ is the number or multiplicity of the free distance codewords [19]. The formula for the free distance spectrum is given by:

$$p_{bp}(d_{free}) \quad < \quad \frac{1}{2N} W_{d_{free}} erfc\left(\sqrt{\frac{E_b R d_{free}}{N_o}}\right) \tag{3.8}$$

From Figure 3.2 we notice that the free distance is the nearest asymptote to the union bound. The 2^{nd} spectrum corresponds to the trellis paths of Hamming distance $D = d_{free} + 1$ and 3rd spectrum corresponds to the paths of Hamming distance $D = d_{free} + 2$. The free distance spectrum significantly influences the code performance and dominates the bound.

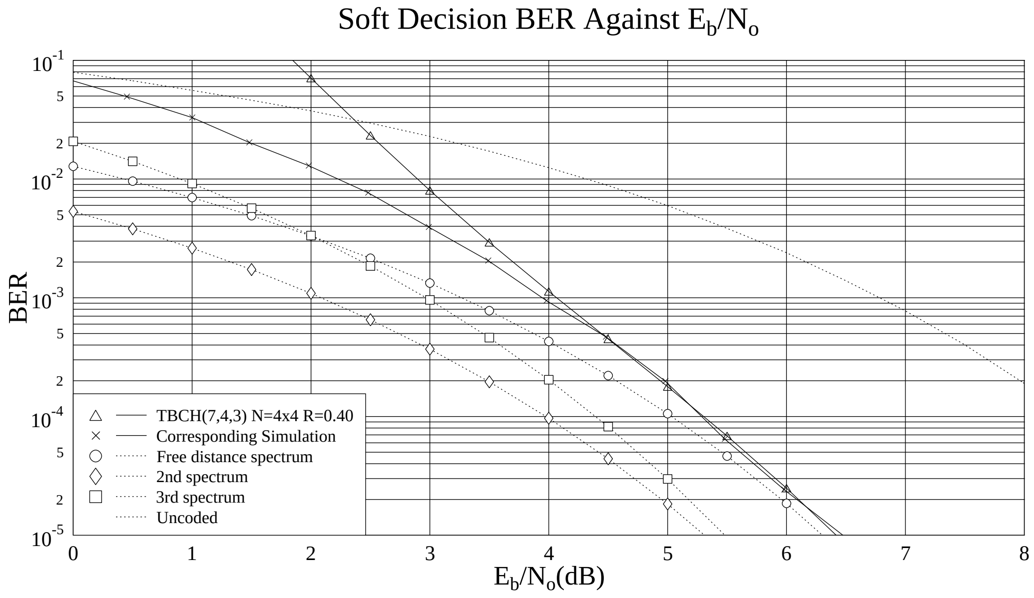

Figure 3.2: Union bound of TBCH(7,4,3) $N = 4 \times 4$ with its modelling simulation and the first three spectra. This shows the influence of the free distance spectrum to the union bound.

3.5.1.2 Higher Order Hamming Distance

The number of error bits in a codeword is defined as the Hamming distance of the codes, D. The codeword length increases when the N is increases. The maximum number of error bits that might be introduced during the channel transmission is the length of the codeword, L. Therefore the number of D term increases as the N increases, up to the value of the codeword length. We have limited D terms due to the block code structure.

From Figure 3.3 we can see the significant change of the weight distribution function as the N increases, although the coding rate remains the same, i.e. $R = 0.40$ for TBCH(7,4,3). The d_{free} for the code with $N = 4 \times 2$ and $N = 4 \times 4$ is 4 and 5 respectively. Their corresponding $W(d_{free})$ is 2 and 9 respectively.

From Figure 3.4 it is clear that the spectra corresponding to higher D terms move further away to the left of the union bound. Therefore, spectra of greater D values, i.e. the last few spectra, contribute less significantly to the union bound.

However, from Figure 3.4 for TBCH(7,4,3) $N = 4 \times 4$ code, the third spectrum is nearer to the bound than the second spectrum. The corresponding weight distribution function is shown on the right hand side of Figure 3.3, where the weight of the second spectrum is much smaller than the weight of the third spectrum. A spectrum of $D = k + 1$ with a very high weight is nearer to the union bound compared to the spectrum of $D = k$ with very low weight. In other word, a spectrum of small weight, $W(D)$ would stretch to the left.

On the other hand, a spectrum of big d_{free} term will influence the union bound to move to the

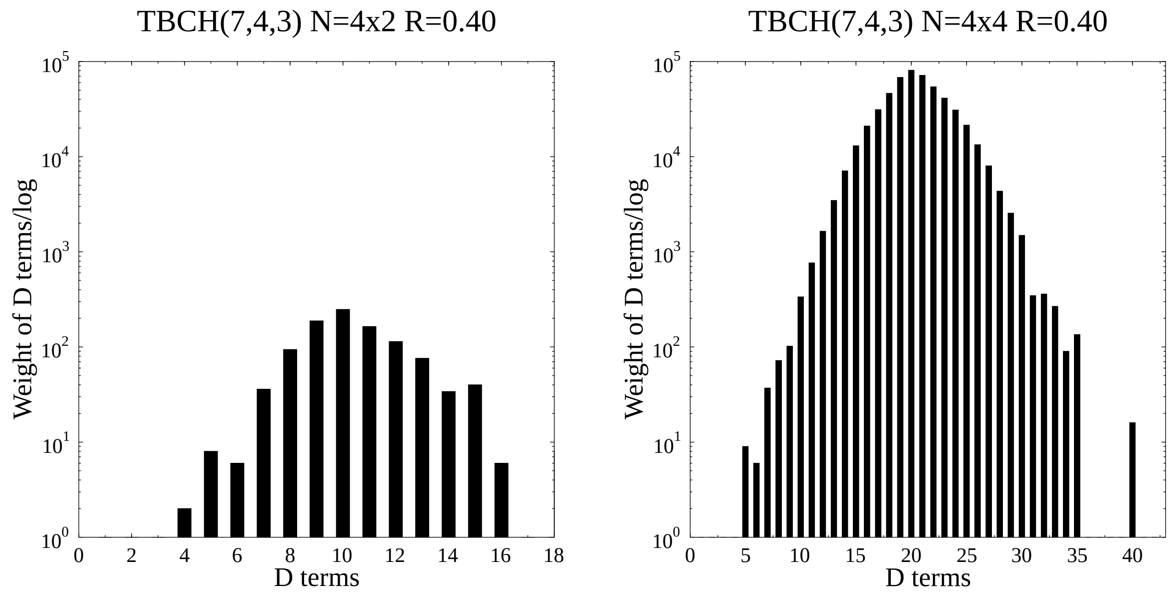

Figure 3.3: Weight distributions of TBCH(7,4,3) for N=[4x4, 4x2]. This shows the effect of increasing the interleaver size, N of the TBCH codes with a constant coding rate, R.

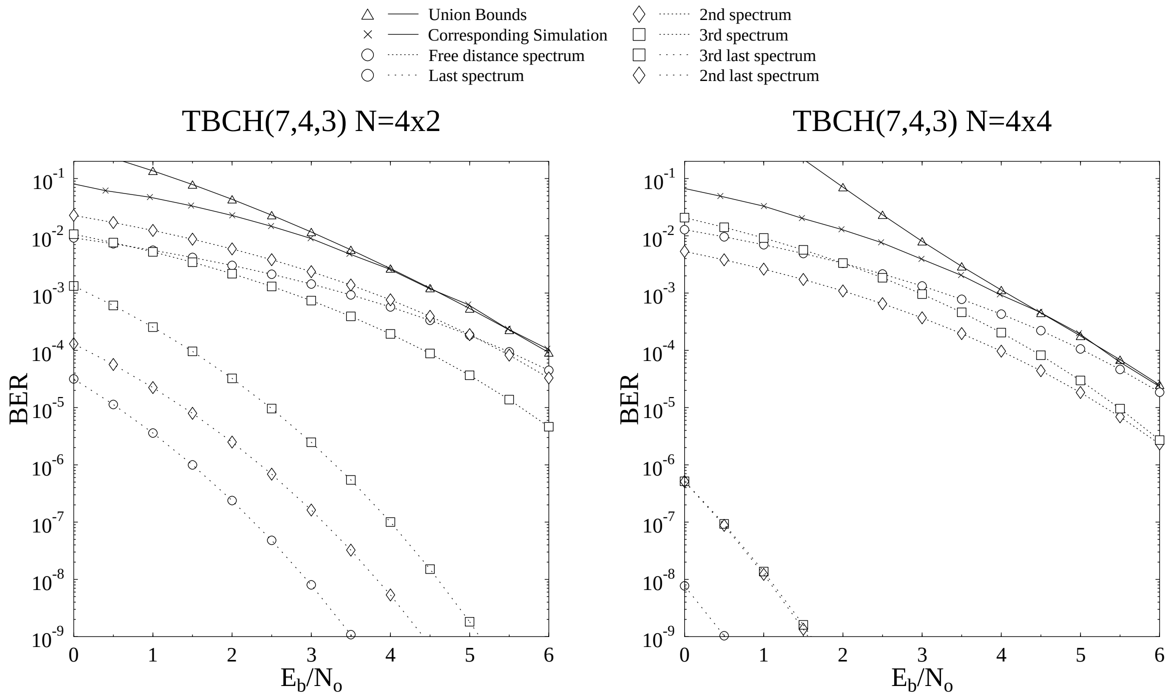

Figure 3.4: Union bounds and some spectra of TBCH(7,4,3) for N=[4x2, 4x4]. This shows the effect of the spectra on the union bound.

left and the performance of a code is better if the union bound draws to the left. Therefore, a good code has:

1. Big free distance, i.e. the first D term is big, and
2. Small free distance weight, $W(d_{free})$.

3.5.2 Effects of Block Interleaving

The information databits are read into the content of the block interleaver in a row by row basis and then the content is written to the second encoder in a column by column basis. This interleaver must be fully filled by the databits before its content is read out.

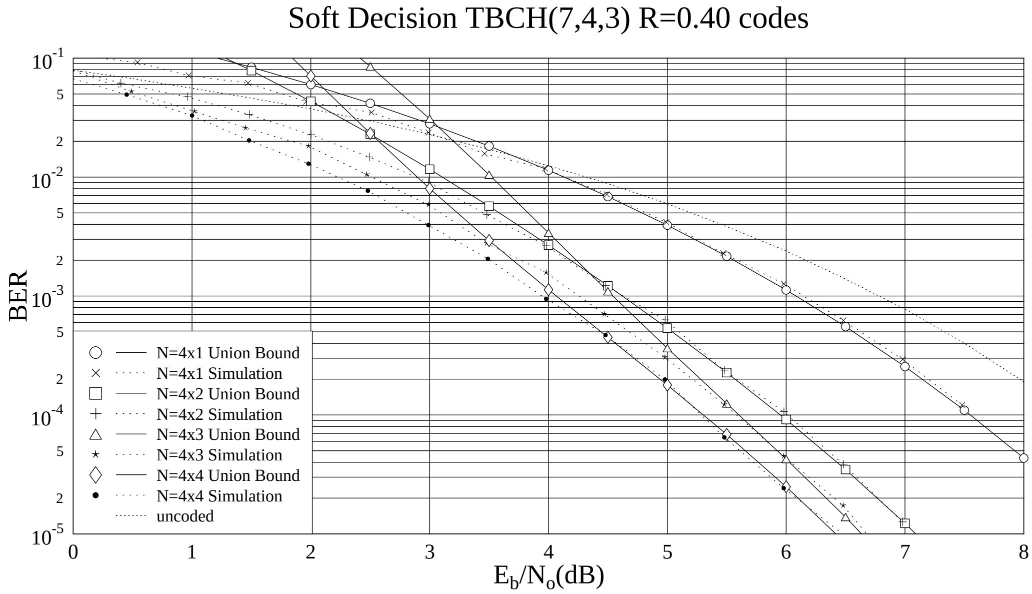

Figure 3.5: Comparison of Union bounds and modelling simulations for TBCH(7,4,3) for N=[4x1,4x2,4x3,4x4]. This shows the effect of increasing interleaver size, N.

Figure 3.5 shows the union bounds and the modelling simulations for TBCH(7,4,3) code with R=0.40. It is clear that the code performs better with larger N.

3.5.3 Effects of Puncturing

We can apply puncturing technique to obtain a code with desired coding rate. When more parity bits in the codeword of the systematic code is deleted by puncturing, the coding rate will be increased. A code with higher coding rate has better transmission efficiency but in return the BER performance degrades. This phenomenon is explicitly portrayed in Figure 3.6, where the BER performance of the code decreases as the coding rate is increased by reducing the pw value.

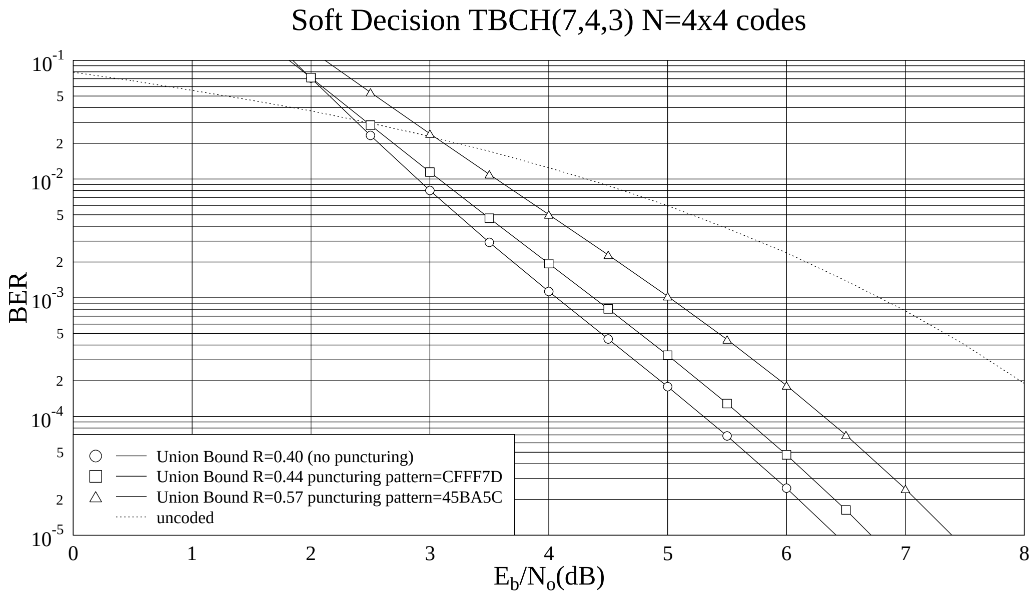

Figure 3.6: Union bounds of TBCH(7,4,3) $N = 4x4$ for R=[0.41,0.44,0.57]. This shows the effect of puncturing by increasing the coding rate, R.

3.5.3.1 Puncturing Patterns

As can be noticed from Figure 3.7 the performance of the codes may differ significantly by applying different puncturing patterns. A program called 'optimum puncturing pattern search' was written in C to find the optimum puncturing pattern for turbo codes using the distance spectra effect of the codes, as describe in Section 3.5.1, as the test parameters. Puncturing patterns showed in Figure 3.6 and 3.7 are in hexadecimal form. When the pattern is converted to binary form, a '1' means the parity bit at that position is to be appended to form the codeword, whereas a '0' means the parity bit at that position is to be deleted.

The total parity size for TBCH(7,4,3) with interleaver size $N = 4 \times 4$ could be calculated using Formula 3.2 and it is 24 bits in this case. When applying parity weight of $pw = 12$ we only transmit $pw = 12$ parity bits out of the total $P_L = 24$ parity bits. There are $N/k = 4$ information-words contained in the interleaver. Hence, there are $N/k = 4$ codewords produced by each encoder before new information databits are read into the interleaver in the new cycle. Each codeword has $n - k = 3$ parity bits and they are punctured according to the puncturing pattern. Hence, each puncturing pattern for a codeword has $n - k = 3$ bits. If the 3 parity bits of a 7-bit codeword of the TBCH(7,4,3) code is punctured by the pattern 011_{bin}, then the codeword becomes 'dddd0pp' where 'd' is the position of the unpunctured information databit, 'p' is the position of the unpunctured parity bit and '0' is the position of the punctured parity bit. Each encoder produces $N/k = 4$ codewords at each cycle, hence there are 8 patterns for the two encoders. We can represent the whole puncturing pattern as $[P_1P_1P_1\ P_1P_1P_1\ P_1P_1P_1\ P_1P_1P_1\ P_2P_2P_2\ P_2P_2P_2\ P_2P_2P_2\ P_2P_2P_2]_{bin}$ to the whole $8 \times 3 = 24$ parity bits, where P_1 is for the first encoder and P_2 is for the second encoder. This binary puncturing pattern

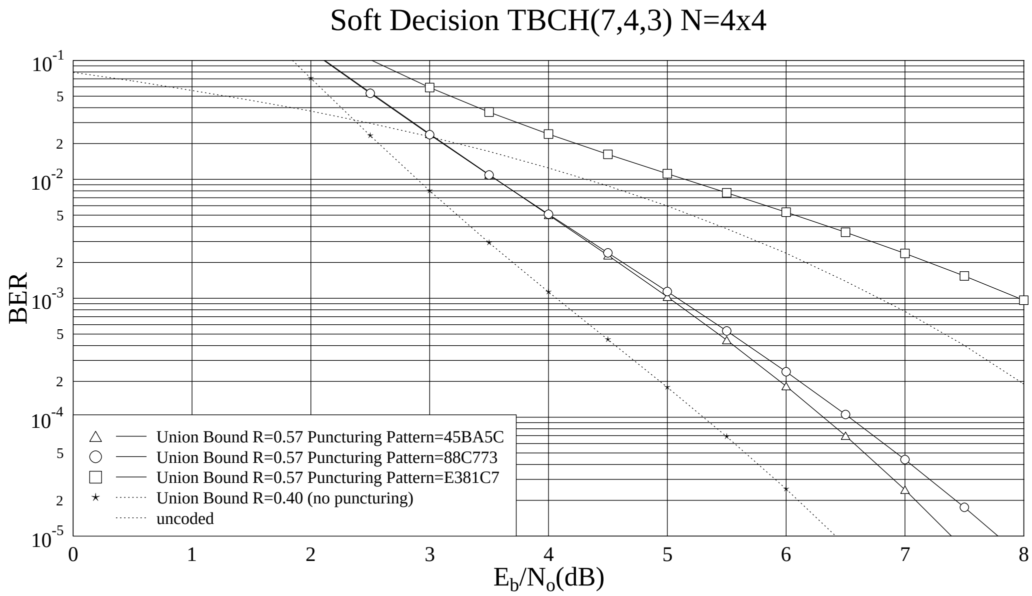

Figure 3.7: Union bounds of TBCH(7,4,3) $N = 4x4$ for some puncturing patterns with $R = 4/7$. This shows the effect of puncturing pattern using a constant coding rate, R.

is then converted into hexadecimal form.

The optimum puncturing pattern for TBCH(7,4,3) with $pw = 12$ is found to be 45BA5C$_{hex}$ as shown in Figure 3.7. 45BA5C$_{hex}$ corresponds to [010 001 011 011 101 001 011 100]$_{bin}$, therefore the whole set of data in one cycle can be represented by Table 3.1. This pattern gives d_{min} of 3, which is the biggest d_{min} could be found in all puncturing patterns with $pw = 12$. The corresponding weight is $W(d_{min}) = 20$, which is the smallest $W(d_{min})$ could be found in all available $d_{min} = 3$ patterns.

Encoder	*databits*	*paritybits*
$1st$	dddd	0p0
	dddd	00p
	dddd	0pp
	dddd	0pp
$2nd$	xxxx	p0p
	xxxx	00p
	xxxx	0pp
	xxxx	p00

Table 3.1: The whole data after being punctured by puncturing pattern 45BA5C$_{hex}$. 'd' represents the position of the original information databit, 'x' represents the position of the interleaved information databit, 'p' represents the position of the unpunctured parity bit and '0' represents the position of the punctured parity bit.

3.5.4 Effects of Various Component Codes

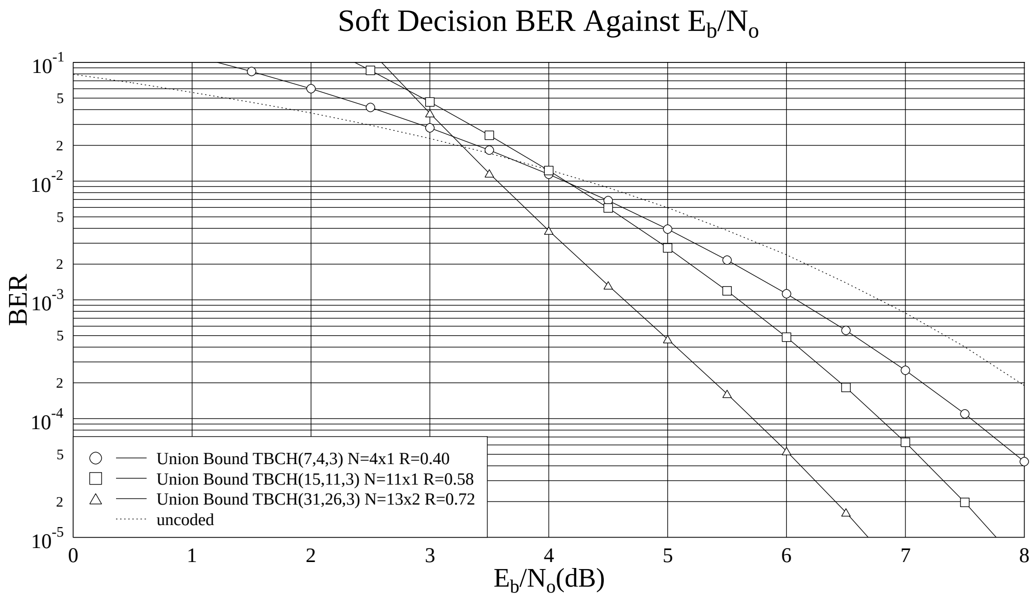

Figure 3.8: Union bounds of TBCH(7,4,3) $N = 4x1$, TBCH(15,11,5) $N = 11x1$ and TBCH(31,26,3) $N = 13x2$. This shows the effect of different component code.

There are a lot of BCH codes that could be used as the component codes. A wide variety of turbo block codes families could be produced with different combinations of these BCH codes.

Figure 3.8 shows some of the families of TBCH codes using the smallest interleaver size, N. We can notice that the performance is better for BCH component codes with large n.

Chapter 5

Conclusion

In Trellis decoding, the BCH codes have optimum performance at half rate as shown in Section 2.7.1.4, using large codeword length as shown in Section 2.7.1.3 and small information-word length as shown in Section 2.7.1.2. The Turbo BCH codes have optimum performance with large interleaver size as shown in Section 3.5.2, using the optimum BCH codes as the component codes as shown in Section 3.5.4. Puncturing increases the coding rate but degrades the performance as shown in Section 3.5.3. A code of large free distance with small free distance weight is a good code as shown in Section 3.5.1.

In Algebraic decoding, the RS and RRNS codes have very similar BER performance as shown in Section 4.5.1. The RRNS codes have optimum performance at large coding rate as shown in Figure 4.6, large codeword length as shown in Section 4.5.4 and large information-word length as shown in Section 4.5.3, with small symbol length as shown in Figure 4.7. An optimum moduli set gives symbols of equal length with largest dynamic range as shown in Section 4.5.2.

In bit-oriented Trellis decoding, a small rate RRNS code can be made to perform better at low SNR by increasing the ratio of parity bits to databits within a fixed codeword length as shown in Figure 4.10 and 4.11. The turbo RRNS codes performance is optimum with large interleaver size as shown in Figure 4.12.

The main obstacle in this project is in simulating the transfer function of the codes for Trellis decoding method. The computational complexity increases exponentially with the information-word length and the interleaver size.

Future Works

Although the optimum parameters for the codes are found, we can't simulate the theoretical performance for codes of huge k and N due to the computational complexity. Hence, other efficient and accurate techniques should be sought in order to obtain a huge number of performance bounds.

The theoretical performance program of Algebraic decoding for the RRNS codes is very fast compared to the modelling simulation program. Hence, the optimum moduli sets can be generated much more quickly by adapting this theoretical performance program.

The performance evaluation of these codes using other modulation schemes and channel types is another good investigation.

Appendix A

Generators for BCH codes

n	k	t	d_{min}	$g(x)$
7	4	1	3	13
15	11	1	3	23
	7	2	5	721
	5	3	7	2467
31	26	1	3	45
	21	2	5	3551
	16	3	7	107657
	11	5	11	5423325
	6	7	15	313365047
63	57	1	3	103
	51	2	5	12471
	45	3	7	1701317
	39	4	9	166623567
	36	5	11	1033500423
	30	6	13	157464165547
	24	7	15	17323260404441
	18	10	21	1363026512351725
	16	11	23	6331141367235453
	10	13	27	472622305527250155
	7	15	31	5231045543503271737
127	120	1	3	211
	113	2	5	41567
	106	3	7	11554743
	99	4	9	3447023271
	92	5	11	624730022327
	85	6	13	130704476322273
	78	7	15	26230002166130115
	71	9	19	6255010713253127753
	64	10	21	1206534025570773100045
	57	11	23	335265252505705053517721
	50	13	27	54446512523314012421501421
	43	15	31	17721772213651227521220574343
	36	15	31	3146074666522075044764574721735
	29	21	43	403114461367670603667530141176155
	22	23	47	123376070404722522435445626637647043
	15	26	53	22057042445604554770523013762217604353
	8	31	63	7047264052751030651476224271567733130217
255	247	1	3	435
	239	2	5	267543
	231	3	7	156720665
	223	4	9	75626641375
	215	5	11	23157564726421
	207	6	13	16176560567636227
	199	7	15	7633031270420722341
	191	8	17	2663470176115333714567
	187	9	19	52755313540001322236351
	179	10	21	22624710717340432416300455
	171	11	23	15416214212342356077061630637
	163	12	25	7500415510075602551574724514601
	155	13	27	3757513005407665015722506464677633
	147	14	29	1642130173537165525304165305441011711
	139	15	31	461401732060175561570722730247453567445

Table A.1: Table of Generators for BCH codes [2] ©IEEE, Stenbit

Appendix B

Coding Gain Table

The performance of the block codes are evaluated using BPSK modulation scheme over AWGN channel. The coding gains at BER of 10^{-3} and 10^{-6} are tabulated.

B.1 BCH Codes

The coding gains of theoretical bound for some BCH codes using soft decision Viterbi Algorithm are tabulated in Table B.1 and B.2.

Code	Code Rate R	E_b/N_o (dB)		Gain (dB)	
		BER			
		10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded	1.00	6.78	10.53	0.00	0.00
BCH(7,4,3)	0.57	5.39	8.58	1.39	1.95
BCH(15,11,3)	0.73	4.97	7.78	1.81	2.75
BCH(15,7,5)	0.47	4.68	7.51	2.10	3.02
BCH(15,5,7)	0.33	4.70	7.56	2.08	2.97
BCH(31,26,3)	0.84	4.97	7.45	1.81	3.08
BCH(31,21,5)	0.68	4.20	6.42	2.58	4.11
BCH(31,16,7)	0.52	3.91	6.18	2.87	4.35
BCH(31,11,11)	0.35	3.77	6.04	3.01	4.49
BCH(31,6,15)	0.19	4.26	6.90	2.52	3.63

Table B.1: Coding gain of systematic BCH codes with $n = 7, n = 15$ and $n = 31$ using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Code	Code Rate R	E_b/N_o (dB)		Gain (dB)	
		BER			
		10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded	1.00	6.78	10.53	0.00	0.00
BCH(63,24,15)	0.38	3.09	4.80	3.69	5.73
BCH(63,18,21)	0.29	3.07	4.82	3.71	5.71
BCH(63,16,23)	0.25	3.12	4.96	3.66	5.57
BCH(63,10,27)	0.16	3.54	5.75	3.24	4.78
BCH(63,7,31)	0.11	3.93	7.00	2.85	3.53
BCH(127,22,47)	0.17	2.63	4.10	4.15	6.43
BCH(127,15,53)	0.12	2.91	4.69	3.87	5.84
BCH(127,8,63)	0.06	3.67	6.02	3.11	4.51

Table B.2: Coding gain of systematic BCH codes with $n = 63$ and $n = 127$ using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

B.2 TBCH Codes

The coding gains of theoretical bound for some Turbo BCH codes using soft decision Viterbi Algorithm are tabulated in Table B.3, B.4, B.5, B.6 and B.7.

Code	Interleaver Size N col x row	Code Rate R	E_b/N_o (dB)		Gain (dB)	
			BER			
			10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded		1.00	6.78	10.53	0.00	0.00
TBCH(7,4,3)	4 x 1	0.40	6.08	9.64	0.70	0.89
TBCH(7,4,3)	4 x 2	0.40	4.62	8.05	2.16	2.48
TBCH(7,4,3)	4 x 3	0.40	4.54	7.53	2.24	3.00
TBCH(7,4,3)	4 x 4	0.40	4.07	7.36	2.71	3.17
TBCH(7,4,3)	4 x 5	0.40	4.07	7.36	2.71	3.17
TBCH(7,4,3)	4 x 6	0.40	4.07	7.33	2.71	3.17

Table B.3: Coding gain of systematic TBCH(7,4,3) codes with different interleaver size N using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Code	Interleaver Size N col x row	Code Rate R	E_b/N_o (dB)		Gain (dB)	
			BER			
			10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded		1.00	6.78	10.53	0.00	0.00
TBCH(15,5,7)	5 x 1	0.20	4.93	7.87	1.85	2.66
TBCH(15,5,7)	5 x 2	0.20	3.92	6.89	2.86	3.64
TBCH(15,5,7)	5 x 3	0.20	3.54	6.56	3.24	3.97
TBCH(15,5,7)	5 x 4	0.20	3.33	6.33	3.45	4.20
TBCH(15,5,7)	5 x 5	0.20	3.15	6.19	3.63	4.34
TBCH(15,7,5)	7 x 1	0.30	5.14	8.18	1.64	2.35
TBCH(15,7,5)	7 x 2	0.30	3.87	6.94	2.91	3.59
TBCH(15,7,5)	7 x 3	0.30	3.55	6.74	3.23	3.79
TBCH(15,11,3)	11 x 1	0.58	5.60	8.59	1.18	1.94
TBCH(15,11,3)	11 x 2	0.58	4.54	8.06	2.24	2.47

Table B.4: Coding gain of systematic TBCH$(15, k, d_{min})$ codes with different interleaver size N using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Code	Interleaver Size N col x row	Code Rate R	E_b/N_o (dB)		Gain (dB)	
			BER			
			10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded		1.00	6.78	10.53	0.00	0.00
TBCH(31,6,15)	6 x 1	0.11	4.35	7.02	2.43	3.51
TBCH(31,6,15)	6 x 2	0.11	3.41	6.15	3.37	4.38
TBCH(31,6,15)	6 x 3	0.11	3.03	5.84	3.75	4.69
TBCH(31,6,15)	6 x 4	0.11	2.89	5.78	3.89	4.75
TBCH(31,11,11)	11 x 1	0.22	3.98	6.31	2.80	4.22
TBCH(31,11,11)	11 x 2	0.22	2.81	4.97	3.97	5.56
TBCH(31,16,3)	4 x 4	0.35	3.44	5.77	3.34	4.76
TBCH(31,21,5)	7 x 3	0.51	3.64	5.87	3.14	4.66
TBCH(31,26,3)	13 x 2	0.72	4.63	7.52	2.15	3.01

Table B.5: Coding gain of systematic TBCH$(31, k, d_{min})$ codes with different interleaver size N using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Code	Interleaver Size N col x row	Code Rate R	E_b/N_o (dB)		Gain (dB)	
			BER			
			10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded		1.00	6.78	10.53	0.00	0.00
TBCH(63,7,31)	7 x 1	0.06	3.98	6.46	2.80	4.07
TBCH(63,7,31)	7 x 2	0.06	3.10	5.66	3.68	4.87
TBCH(63,7,31)	7 x 3	0.06	2.73	5.34	4.05	5.19
TBCH(63,10,27)	10 x 1	0.09	3.59	5.82	3.19	4.71
TBCH(63,10,27)	10 x 2	0.09	2.64	4.73	4.14	5.80
TBCH(63,16,23)	4 x 4	0.15	2.91	4.70	3.87	5.83
TBCH(63,18,21)	6 x 3	0.17	2.82	4.52	3.96	6.01
TBCH(63,24,15)	6 x 4	0.24	2.69	4.17	4.09	6.36

Table B.6: Coding gain of systematic TBCH$(63, k, d_{min})$ codes with different interleaver size N using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Code	Interleaver Size N col x row	Code Rate R	E_b/N_o (dB)		Gain (dB)	
			BER			
			10^{-3}	10^{-6}	10^{-3}	10^{-6}
Uncoded		1.00	6.78	10.53	0.00	0.00
TBCH(127,8,63)	4 x 2	0.03	3.69	6.03	3.09	4.50
TBCH(127,8,63)	4 x 4	0.03	2.83	5.18	3.95	5.35
TBCH(127,8,63)	8 x 3	0.03	2.48	4.89	4.30	5.64
TBCH(127,15,53)	5 x 3	0.06	2.82	4.58	3.96	5.95
TBCH(127,22,27)	11 x 2	0.09	2.48	3.88	4.30	6.65

Table B.7: Coding gain of systematic TBCH$(127, k, d_{min})$ codes with different interleaver size N using SD Viterbi Algorithm and BPSK modulation scheme over AWGN channel

Glossary

AD	Algebraic Decoding
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem. A class of forward error correcting codes (FEC)
BER	Bit error rate, the number of the bits received incorrectly
BEX	Base Extension
BPSK	Binary Phase Shift Keying
FEC	Forward Error Correction
HD	Hard Decision
PDF	Probability Density Function
RNS	Residue Number System
RRNS	Redundant Residue Number System
RS	Reed Solomon Codes
RSC	Recursive Systematic Convolutional
SD	Soft Decision
SNR	Signal to Noise Ratio, noise energy compared to the signal energy
TBCH	Turbo Bose-Chaudhuri-Hockuenghem codes
TD	Trellis Decoding
TRRNS	Turbo Redundant Residue Number System
VA	Viterbi Algorithm

Bibliography

[1] T. H. Liew, "Turbo Block Coding," tech. rep., University of Southampton, January 1999.

[2] John P. Stenbit, "Table of Generators for Bose-Chaudhuri Codes," *IEEE Transactions on Information Theory*, vol. 10, pp. 390–391, October 1964.

[3] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell system Technical Journal*, vol. 27, pp. 379–656, July 1948.

[4] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding : Turbo Codes," in *Proceedings, IEEE International Confrence on Communications*, pp. 1064–1070, 1993.

[5] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, September 1959.

[6] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error-correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, March 1960.

[7] K.H.H. Wong and L. Hanzo, *Mobile Radio Communications*, ch. 4.4, pp. 399–476. IEEE Press, 445 Hoes Lane, Piscataway, NJ, 08855, USA: IEEE Press and Pentech Press, 1992.

[8] Bernard Sklar, *Digital Communications Fundamentals and Applications*, ch. 5.6.5, pp. 294–296. Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[9] Richard E. Blahut, *Theory And Practice of Error Control Codes*, ch. 6, pp. 130–160. IBM Corporation, Owego, NY 13827, USA: Addison-Wesley Publishing Company, 1983.

[10] Andrew J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[11] Bahram Honary and Garik Markarian, *Trellis Decoding Of Block Codes.* Kluwer Academic Publishers Group, Distribution Centre, Post Office Box 322, 3300 AH Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997.

[12] Christian Schlegel, "Performance Bounds," in *Trellis Coding*, (New York), pp. 121–151, September 1997.

[13] Bernard Sklar, *Digital Communications Fundamentals and Applications*, ch. 1.5.5, pp. 27–30. Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[14] Phillip A. Bello, "Sample Size Required in Error-Rate Measurement on Fading Channel," *IEEE Proceedings*, vol. 86, pp. 1435–1441, July 1998.

[15] Sergio Benedetto and Guido Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, March 1996.

[16] Sergio Benedetto and Guido Montorsi, "Generalized Concatenated Codes with Interleavers," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 32–39, September 1997.

[17] Christian Schlegel, "Turbo Codes," in *Trellis Coding*, (New York), pp. 233–263, September 1997.

[18] O. Aitsab, R. Pyndiah and B. Solaiman, "Iterative Source/Channel Decoding using Block Turbo Code and VQ for image transmission," in *International Symposium on Turbo Codes and related topics*, (Brest, France), pp. 170–173, September 1997.

[19] Lance C. Perez, Jan Seghers and Daniel J. Costello, "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1698–1709, November 1996.

[20] Fred J. Taylor, "Residue Arithmetic: A Tutorial with Examples.," *IEEE Computer Magazine*, May 1984.

[21] R. W. Watson and C. W. Hastings, "Self-Checked Computation Using Residue Arithmetic," *Proceedings of the IEEE*, vol. 54, pp. 1920–1931, December 1966.

[22] Nicholas S.Szabo and Richard I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. McGraw-Hill Book Company, 1967.

[23] Hari Krishna, Kuo-Yu Lin and Jenn-Dong Sun, "A Coding Theory Approach to Error Control in Residue Number Systems - Part I: Theory and Single Error Correction," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 8–17, January 1992.

[24] Jenn-Dong Sun and Hari Krishna, "A Coding Theory Approach to Error Control in Residue Number Systems - Part II: Multiple Error Detection and Correction," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 18–34, January 1992.

[25] T. H. Liew, L. L. Yang, L. Hanzo, "Soft-Decision Redundant Residue Number System Based Error Correction coding," *Submitted to IEEE Communications Letters*, February 1999.

[26] Elwyn R. Berlekamp, *Algebraic Coding Theory*. McGraw-Hill Book Company, 1968.

[27] Toshimitsu Kaneko, Toshihisa Nishijima, Hiroshige Inazumi and Shigeichi Hirasawa, "An Efficient Maximum-Likelihood-Decoding Algorithm for Linear Block Codes with Algebraic Decoder," *IEEE Transactions on Information Theory*, vol. 40, pp. 320–327, March 1994.

[28] John G. Proakis, "Block And Convolutional Channel Codes," in *Digital Communication*, (New York), pp. 464–465, September 1995.

[29] John G. Proakis, "Optimum Receivers for the Additive White Gaussian Noise Channel," in *Digital Communication*, (New York), pp. 260–274, September 1995.