Trellis Decoding of 3-D Block Turbo Codes

Bo Yin

A Thesis
in
The Department
of
Electrical and Computer Engineering

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

May 2002

0-612-72917-6

ABSTRACT

Trellis Decoding of 3-D Block Turbo Codes

Forward Error Correction (FEC) technique provides a method to detect and correct errors in transmitted data. It is also a valuable technique to reduce the power requirement, thus have an important role in these systems. This reduction in power requirement is achieved at the expense of an increase in bandwidth requirement. The objective is usually to find error control techniques that give good tradeoff between power and bandwidth requirements. In this thesis, we present results for FEC technique using Turbo Block Codes and Turbo Product Codes. It is shown that these codes, not only in theory but also in hardware implementation, are capable of providing significant performance gains over other error-correction schemes.

This thesis investigates Trellis based iterative decoding techniques applied to concatenated coding schemes, Turbo Block Codes. We use RM(n, k) to construct 2-D and multi-dimensional Turbo Block Codes. Our objective is to get high code rates and long block sizes for more bandwidth efficiency and improving the performance of the optimised maximum *a posterior* decoding algorithm. The simulation results really show that multi-dimensional block turbo codes can achieve high coding gains at acceptable delays for real-time and data channels. Turbo Product Codes are investigated in this thesis to compare with Turbo Block Codes.

For reducing the complexity of hardware implementation, we introduce the suboptimal algorithm for the decoding of the proposed coding schemes. The algorithm MAX-Log-MAP is derived from optimal MAP decoding algorithm with the approximation to avoid a lot of complex computations. The suboptimal MAX-Log-MAP algorithm is combined with correction values in order to avoid performance degradation. This decoding strategy presents a good compromise between performance and complexity, making it very attractive for practical applications. For the easy hardware implementation, quantization would be discussed briefly. The Additive White Gaussian Noise channel model and different modulations like BPSK and QPSK are used in the simulations.

It is dedicated to my parents and my family

Thanks for all your love and support...

--B. Y.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr R. M. Soleymani for introducing me to the area of turbo codes for wireless and satellite communications, offerring the opportunity to me to attend the CITR conferrence, and for providing me with continuous feedback and encouragement during the period I worked in this thesis.

I would like to thank Miss Usa Vilaipornsawai and Miss Gao Yingzi for many discussions and suggestions pertinent to this work.

I would like to thank Dr Yousef R. Shayan for taking his time to review my presentation and give me some good suggestions in my thesis, and thank all professors for their time to serve on my defense committee.

Special thanks to my husband and all my friends for their sincere support and making my stay here an enjoyable one. I would like to thank the help and support from the faculty members of Electrical and Computer Engineering and all my colleagues working in Wireless and Satellite Communications LAB.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

ADSL	Asymmetric Digital Suberscriber Line
APP	A Posteriori Probability
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
ATM	Asynchronous Transfer Mode
BCH	Bose-Chaudhure-Hocquenghem
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit Error Rate
BPSK	Binary Phase Shift keying modulation
BTC	Block Turbo Code
CTC	Convolutional Turbo Code
CCSDS	Consultative Commitee for Space Date System
CITR	Canadian Institute for Telecommunications Rearch
CRSC	circular recursive systematic convolutional
DSP	Digital Signal Processing
DVB-RCS	Digital Video Broadcasting standard for Return Channel via Satellite
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FER	Frame Error Rate
GF	Galois Field
IP	Internet Protocal
LLR	Log-Likilihood Ratio
MAP	Maximum A Posteriori
MLD	Maximum-Likelihood Decoding

MPEG	Moving Picture Experts Group
MSGM	Minimal Span Generator Matrix
OFDM	Orthogonal Frequency-Division Multiplexing
PCCC	Parallel Concatenated Convolutional Code
PRBS	PseudoRandom Binary Sequences
QPSK	Quadrature Phase-Shift Keying
RCST	Return Channel Satellite Terminal
RM	Reed-Muller
RRE	Row-reduced Echelon
RS	Reed-Solomon
RSC	Recursiv Systematic Convolutional
SISO	Soft-Input Soft-Output
SOVA	Soft-Output Viterbi Algorithm
TC	Turbo Code
CTC	Convolutional Turbo Code
UMTS	Universal Mobile Telecommunication Service
VA	Viterbi algorithm

Chapter 1

Introduction

1.1 Coding

Let us start with a communication system, which connects a data source to a data user through a channel. Because the channel is subject to various types of noise, distortion, and interference, the channel output differs from the channel input [1]. The Figure 1.1 provides a rough idea of a general information transmission system.

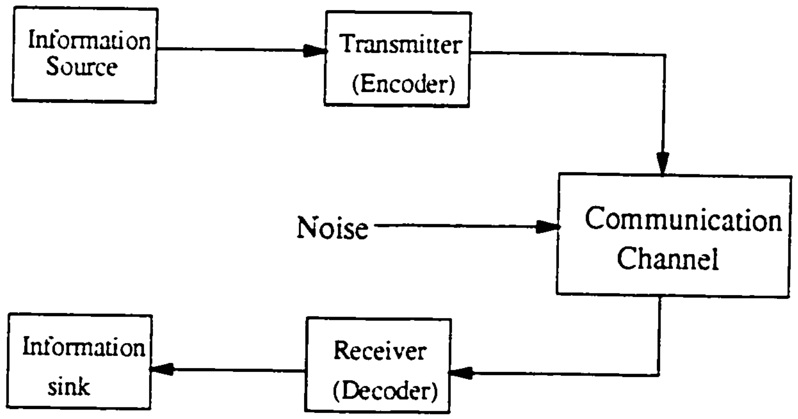


Figure 1.1: Simplified block diagram of a coded system

Coding theory, is the study of methods for efficient and accurate transfer of information from one place to another. It deals with the problem of detecting and correcting transmission errors caused by noise in the channel. The theory has been developed in order to minimize the effect of noise from source to the destination.

Recent advances in digital electronic devices have made the implementation of high speed and powerful encoders and decoders possible. We have to design the encoder/decoder pair such that, they

1. are capable of fast encoding and decoding,

2. can correct most of the errors due to the channel noise, i.e., have very low BER,

3. can ensure maximum transfer of information per unit of time, and

4. the cost of implementing the encoder and decoder falls within the acceptable limits.

There are two different types of codes in common use at present [2]: block codes and convolutional codes. The k bit length *message* u is encoded as an n bit long *code word* c, by predetermined rules, $n-k$ redundant bits are added to the k information bits to form the n coded bits., where $k \leq n$: therefore, corresponding to the 2^k different possible messages, there are 2^k different possible code words at the encoder output. This set of 2^k code words of length n is referred to as an (n, k) *block code.* The ratio $R = k/n$ is called the *code rate.* The encoder for a convolutional code also accepts k-bit blocks of the information sequence u and produces an encoded sequence c of n-symbol blocks. The difference from block codes is that the encoder for the convolutional codes has a *memory order* of m and the encoded bits depend not only on the current k input bits but also on past input bits because of the memory order of m. The set of encoded sequences produced by a k-input, n-output encoder of memory order m is called an (n, k, m) *convolutional code.* The definition of *code rate* is the same as the block codes.

In terms of digital communications systems, the maximum average mutual information which can be exchanged through a channel is called the capacity of the channel and is denoted by C. In 1948, Shannon demonstrated that given a suitable channel encoder and decoder we can transmit digital information through the channel at a rate up to the channel capacity with arbitrarily small probability of error. Shannon's famous capacity bound for a band-limited additive white Gaussian channel(AWGN) is

$$C = \log_2(1 + \frac{E_s}{N_0}), \tag{1.1}$$

bits per two-dimensional symbol (bit/sym), where E_s is the energy per two dimensional symbol and $N_0/2$ is called the double sided noise density. E_s/N_0 is called the signal to noise ratio (SNR) for two dimensional modulation schemes. For reliable communication, the above function can be written in terms of bandwidth efficiency as

$$\eta \leq \log_2(1 + \eta\frac{E_b}{N_0}), \tag{1.2}$$

where E_b is the energy per information bit. We can see the minimum E_b/N_0 that can be achieved is ln(2) or -1.6 dB when η tends to zero. This minimum SNR per bit (-1.6 dB) is called the Shannon limit [11] for an additive white Gaussian noise channel.

There are two protection schemes for minimization of the effect of noise when a transmitting message through a physical channel. They are: Forward Error Correction (FEC), that is, using error correcting codes that are able to correct errors at the receiving end, or Automatic Repeat reQuest (ARQ) systems. Recent development in FEC technology has yielded *turbo coding* hardware capable of providing significant performance gains over other error-correction schemes [12].

1.2 Turbo Codes

A newly invented forward error correcting code, Turbo Code (TC) [3] has been shown to be so powerful that the Shannon limit finally becomes practically meaningful. Impressive simulation results were presented in [3] achieving a bit error rate (BER) of 10^{-5} at a SNR of only 0.7 dB. The turbo code(or Convolutional Turbo Code (CTC)), which is parallel concatenation of two relatively simple convolutional encoders and an interleaver, can provide significant coding gains over the classically used cascading a block code (the outer code, typically a Reed-Solomon code) and a convolutional code (the inner code) in a serial structure [3]. Because of their powerful error correcting capability [see section2.1], being implemented with reasonable decoder complexity and also for their flexibility in terms of block size and code rate, the Turbo Codes have reached the top of interest in coding community.

Unlike the traditional encoder, the turbo code encoder introduced in [3] is built using a parallel concatenation of two or more Recursive Systematic Convolutional (RSC) codes with feedback. In turbo code encoder structure, an interleaver is used for getting two or more different information bit streams, then sending each to a different elementary encoder. All information bits with parity check bits are combined together by a multiplexer. Those combined date streams are then sent out to receiver side through the channel. At the decoder side, iterative decoding schemes with "soft-in/soft-out (SISO)" decoders were proposed and used in the turbo code decoder [4]. When decoded by an iterative process, turbo codes offer near optimum performance [3]. There are several decoding algorithms used for turbo code SISO decoder. They are various soft decision algorithms such as MAP [10], MAX-Log-MAP [10], Modified Chase [6] algorithm and so on, for detail information, please see Section2.4 in next Chapter.

Although turbo codes were introduced based on the convolutional code, recent research shows that turbo codes using block codes as elementary encoders can be more suitable for certain applications especial for high code rate [5].

1.3 Block Turbo Codes

The other part of application of 'turbo coding' concept is to the block codes. In 1994, Ramesh Pyndiah from *ENST de Bretagne*, proposed a turbo code based on block codes at the Globecom'94 conference in San Francisco. For this turbo code he used an iterative decoding of two Bose-Chaudhuri-Hocquenghem (BCH) codes concatenated in series through a uniform interleaver. For decoding of the component (elementary) codes he proposed a new SISO decoder for block codes. These codes are quite different from the original turbo codes since they use different component codes, different concatenation schemes and different SISO algorithms. The turbo codes based on convolutional codes are usually known as Convolutional Turbo Codes (CTCs) and those based on block codes are referred to as Block Turbo Codes (BTCs).

The performance of two dimensional block turbo codes can be enhanced to approach the theoretical limit. Compared to the CTCs , it is simple to generate and very efficient especially for high code rates ($R > 0.8$) [6]. Although most research so far has been focused on CTCs, BTCs have been shown to be a more attractive option for a wide range of applications since they can offer a very wide range of block sizes and code rates from below rate $\frac{1}{3}$ to as high as 0.98 without changing in coding strategy. Another advantage of BTCs is that the performance of simple row/column interleaving is as good as random interleaving in CTCs. This allows the use of simpler interleaving structures and hence a saving in terms of system complexity. In BTCs, according to the paper [8] serial concatenation is more advantageous over parallel concatenation and so for all practical purposes BTCs involve iterative SISO decoding of serially concatenated block codes separated by a simple row/column interleaver.Turbo block codes have powerful error-correction capabilities when implemented in multidimensional block turbo codes. They can also be decoded efficiently in hardware because the decoding complexity increases only linearly with the dimensions.

Turbo Product Codes (TPCs) are an important subclass of turbo block codes. They are also known as block turbo codes. TPCs do own all the characteristics of turbo block codes and they are very similar except that they have different code structure on the encoding side, please refer to the Figure 3.1 and Figure . The difference of encoder structure is due to the fact that with TPCs, the parity on parity check bits are also generated and transmitted. Turbo codes without parity on parity check bits would be called Block Turbo Codes. Product Codes were described in 1949 by Elias. These are two dimensional codes constructed from small component codes , and represent a special class of parallel concatenated block codes. We will give more description of turbo product codes in a later Chapter 3 .

1.4 Thesis Structure

This thesis investigates a bandwidth efficient coding scheme for wireless and satellite communication systems. Meanwhile, a reasonable compromise between implementation complexity and degradation of decoding performance is to be presented in this thesis. Our focus is mainly on the Reed-Muller (RM) block turbo codes.

Chapter 1 introduces the simplified communication system and coding theory. Followed by a brief description of turbo codes, its subclass block turbo codes and the thesis outline.

In Chapter 2 we present more detailed description of the turbo codes from its first introduction until the recent products in applications. After that, we will present to various decoding algorithms that are in use.

Chapter 3 begins with the definition of RM codes and how to construct 2-dimensional RM block turbo codes. At this place, we also present the code structure of turbo product codes. Then the optimal and suboptimal decoding algorithms for RM block turbo codes are presented. The simulation results are shown and conclusions added.

In Chapter 4, we continue to extend the RM block turbo codes to 3-dimension. In Chapter 4, the encoder and decoder model, the decoding algorithms are described in detail and the simulations show the more interesting results than 2-dimensional. We will also talk about the effects of quantization and show the simulation results for implementation issues. Finally, puncturing is used for the 3-dimensional code in order to obtain the same rate as the 2-dimensional code. This allows a fair comparison between the 2- and 3-dimensional cases.

Chapter 5 introduces some applications of the block turbo codes.

Conclusions will be drawn in Chapter 6. Also possible future works and a summary of this thesis will be included in this chapter.

Chapter 2

Turbo Codes

2.1 History of Turbo Codes

A good compromise between coding gain and complexity can be achieved by serial concatenated codes proposed by Forney [13]. A serial concatenated code is one that applies two levels of coding, an inner and an outer code linked by an interleaver. The approach has been used in some communication systems such as deep-space communication [14], using a convolutional code as the inner code and a low redundancy Reed-Solomon codes as the outer code. The primary reason for using a concatenated code is to achieve a low error rate with an overall decoding complexity lower than that required for a single code of the corresponding performance. The low complexity is attained by decoding each component code separately. As the inner decoder generates burst errors an interleaver is typically incorporated between the two codes to decorrelate the received symbols affected by burst errors. The serial concatenation model is illustrated in Figure 2.1:

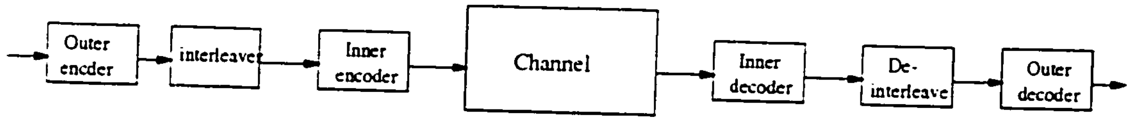


Figure 2.1: Serial Concatenation model

In 1993. a new encoding and decoding scheme. named Turbo Code was reported by: Berrou. Glavieux. and Thitimajshima [3]. that achieves near-capacity performance on the Additive White Gaussian Noise (AWGN) channel. In their paper. Berrou *et al.* used a parallel concatenation of two convolutional encoders that along with a new suboptimal decoding algorithm (modified Bahl *et al.* [3] algorithm) and achieved an E_b/N_0 of 0.7 dB although with huge interleaver of size 256×256 (65536 bits) matrix. 18 iterations at BER of 10^{-5} [3]. They used the parallel concatenation structure given in Figure 2.2.

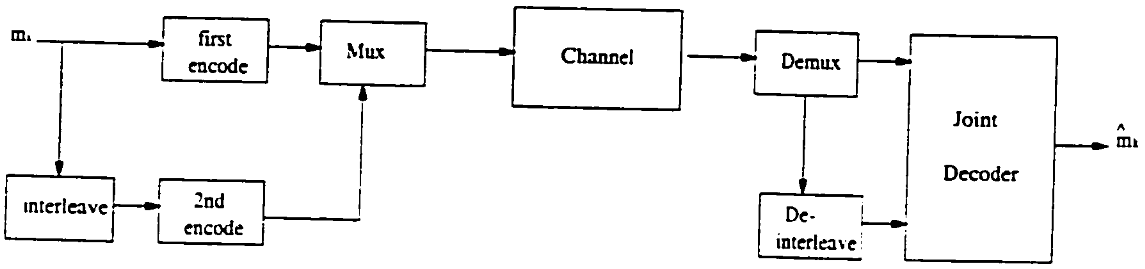


Figure 2.2: Parallel Concatenation model

These turbo codes take advantage of the idea of connecting two codes to get a longer code [3]. The difference is that in turbo codes two identical systematic component codes are connected in parallel and also. the Figure 2.1. the information bits for the second code are not transmitted: therefore. increasing the code rate relative to a corresponding serial concatenated code.

In 1994. Ramesh Pyndiah proposed a turbo code based on block codes in *Globecom*'94 [5]. For this turbo code he used an iterative decoding of two BCH codes concatenated in series through a uniform interleaver. For decoding the component codes he proposed a new SISO decoder for block code. After the first 1993 turbo code paper there has been a considerable number of publications on turbo codes [4]. [5]. [7], [8] and [14], then in1997 *ENST de Bretagne* organized the first International Symposium on Turbo Codes in Brest, France [7].

Due to turbo codes' excellent performance in fighting against the effects of fading and noise. they are widely used in the wireless communication systems. Because block codes can be used to construct of high code rate block turbo codes, which can have both power and bandwidth efficiency and an outstanding error correcting capability. They are used in data packet systems like *Asynchronous Transfer Mode* (ATM) for voice and data, *Motion Picture Expert Group* (MPEG) for video, and *Internet Protocal* (IP) for data.

In the following subsection, we will give more detailed information about turbo codes, their encoder, decoder, interleaver, etc.

2.2 Turbo Code Encoder

2.2.1 Parallel concatenation of codes

The turbo code encoder of [3] is formed by parallel concatenation of two *recursive systematic convolutional* encoders separated by an interleaver [3].

The encoder structure is called parallel concatenation because the two encoders operate on the same set of input bits, rather than one encoding the output of the other, thus turbo codes are also referred to as *parallel concatenated convolutional codes (PCCC)*. The key to the power of parallel concatenated convolutional codes lies in the recursive nature of the decoders and the impact of the interleaver on the coded information stream. The block diagram of a typical turbo code encoder is shown in Figure 2.3.

The output consists of three parts. The first is a straight copy of the input bits, with m (memory size) additional bits appended to ensure the final trellis state is the all zeros state. The second output is that of recursive systematic convolutional encoder #1. Prior to sending the input to the second RSC encoder, the data is interleaved. The interleaved data is then sent to the second encoder (which can be identical to the first). The three parallel streams of data are then multiplexed into one stream and transmitted over the channel. The naming of an "inner code" and an "outer code" is somewhat vague for parallel concatenation, so the codes are often

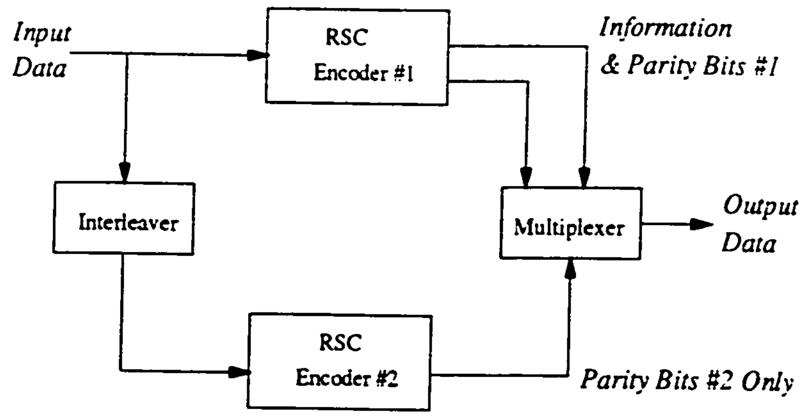


Figure 2.3: Turbo Code Encoder with *PCCC* structure

referred to as a first and second code instead. Quite often, puncturing is applied to the parity streams to get some higher code rate. The puncturing process increases the overall throughput of the system, but also increases the probability of bit error, as will be seen in Section 2.2.3.

Convolutional codes are used with a systematic feedback realization of the encoder. The generator polynomial of a rate $1/n$ encoder is given as

$$G(D) = (g_0(D),\ g_1(D),\ \ldots,\ g_{n-1}(D)), \tag{2.1}$$

the feedback encoder will be

$$G_{sys}(D) = (1,\ \frac{g_1(D)}{g_2(D)},\ \ldots,\ \frac{g_{n-1}(D)}{g_0(D)}). \tag{2.2}$$

The Figure 2.4 shows an example for the rate- 1/2 convolutional code with memory $m = 2$, the generator polynomials are $g_0(D) = 1 + D + D^2$ and $g_1(D) = 1 + D^2$.

2.2.2 Interleaver

Interleaver design is quite an important issue in a turbo-coded system. The interleaver must perform in a way that if the output of one coder produces a low weight codeword, the second encoder's output should have higher weight than the

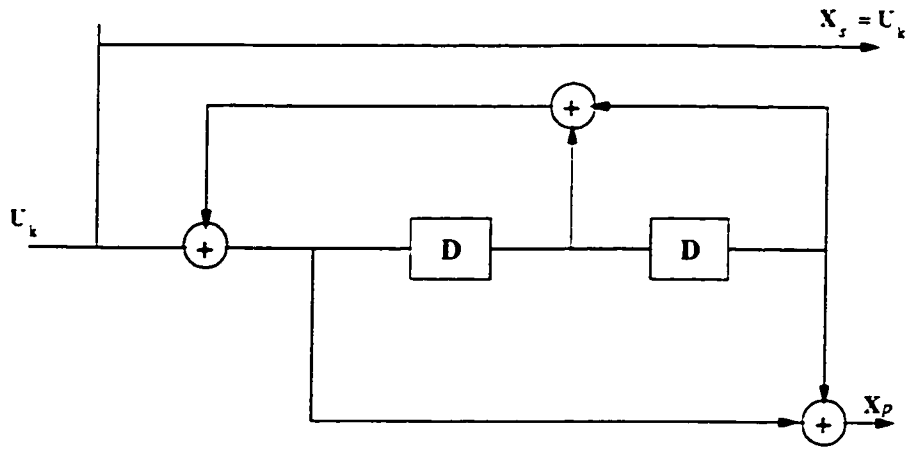


Figure 2.4: Realization of the systematic convolutional encoder with feedback

one coming from the first encoder. This is crucial since it lowers the number of codewords with small hamming weight, and allows the bit error curve to drop at a faster rate as SNR increases. Interleaving is a periodic reordering of blocks of L transmitted symbols. Symbols are correspondingly reordered by de-interleaving in the receiver.

To understand the interleaver, a sequence of interleaver-input symbols can be represented by the s-dimensional vector sequence $X(D)$ where each element in the sequence spans one period of the interleaver

$$X_m = [x_{m+(s-1)}\ x_{m+(s-2)}\ \cdots\ x_m]. \tag{2.3}$$

where m can be considered as a time index corresponding to block of s successive interleaver-input symbols, and

$$X(D) = \sum_m X_m D^m. \tag{2.4}$$

Similarly $\tilde{X}(D)$ can be considered as an s-symbol-element sequence of interleaver outputs, Then, the interleavering can be modeled as a rate 1 convolutional or block code over the symbol alphabet with generator as

$$\tilde{X}(D) = X(D) \cdot G(D), \tag{2.5}$$

where $G(D)$ is an $s \times s$ nonsingular generator matrix as the equation 2.1.

The de-interleaver has generator $G^{-1}(D)$, so that

$$X(D) = \tilde{X}(D) \cdot G^{-1}(D). \tag{2.6}$$

Interleaver design is the key to achieve better performance for turbo codes. By increasing the interleaver size N times, the bit error probability is reduced by a factor N [15]. There are three popular classes of interleaving methods namely: block, convolutional and random interleaving. We will give a little bit more information about block and random interleaving.

Block interleaving or permutation interleaving is the simplest type of interleaving. The permutation of inputs to outputs is contained within one period in a block interleaver. The information data streams will be read row-wise and column-wise to their corresponding encoder through block interleaver [11]. Block turbo codes often use this interleaving method. Figure 2.5 illustrates block interleaver.

In random interleaving, the idea is to eliminate the regular patterns in $G(D)$ so that the period is very long. Random interleaving is often used in turbo coding and for very large frame sizes, random interleavers are near optimum. The main objective of random interleaving is to create a very long block length for the concatenated code when viewed as a single code. Codes selected randomly, as long as the block length is very long, often can achieve capacity. Random interleaving tries to install this element of randomness in the code design. The number of ways in which to make a specific type of error is essentially reduced by the large codeword or interleaver length. There are several types of random interleavers such as Berrou-Glavieux and JPL [9] interleaver. Among them, the one which makes use of PseudoRandom Binary Sequences (PRBS) is used very often by researchers. Such sequences are based on a theory of maximum-length polynomials.

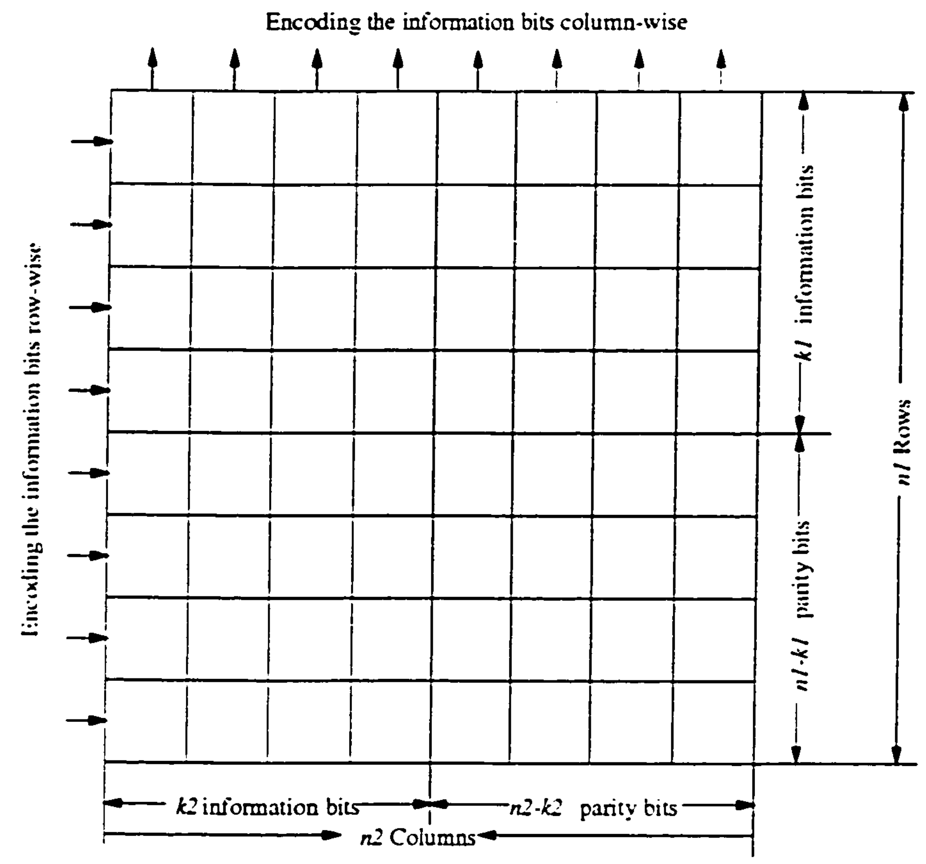


Figure 2.5: A block interleaver for turbo codes

2.2.3 Puncturing of Turbo Codes

The rate of a parallel concatenation of two systematic convolutional codes (where the information bits are sent only once, along with the parity bits of each code) will be

$$R = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} - 1}, \tag{2.7}$$

where the code rate of two component codes are denoted by R_1 and R_2, respectively [10]. In general, the two component codes and their rates are not necessarily the same: either $R_1 \neq R_2$ or $R_1 = R_2$.

Various overall code rates such as 1/2, 2/3, 3/4, 5/6 and so on can be obtained by puncturing the rate 1/3 turbo encoder. *Puncturing* is the regular or periodic deletion of parity bits from a convolutional code output to increase the rate of the code without changing any of its basic attributes. For example, you can increase the rate 1/2 of convolutional code to a rate 2/3 code by puncturing its output – dropping every other output bit of the parity stream. Figure 2.6 illustrates how

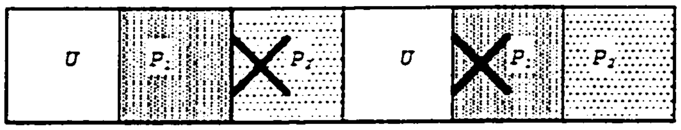


Figure 2.6: Puncturing of rate 1/3 turbo code to rate 1/2

the rate 1/3 turbo code that results from the parallel concatenation of two rate-1/2 codes by alternately deleting one of the two parity bits. CTCs often use puncturing, and Table 2.1 shows some examples of how puncturing affects their code rate. The table assumes Turbo Codes with two constituent encoders as in Figure 2.4, and that puncturing drops half the parity bits each encoder generates.

Table 2.1: Puncturing a Turbo Code increases its rate without changing any of its basic attributes

Rate of Encoder=1	Rate of Encoder=2	Resulting Turbo Code Rate (no puncturing)	Resulting Turbo Code Rate (with puncturing)
1/2	1/2	1/3	1/2
2/3	2/3	1/2	2/3
3/4	3/4	3/5	3/4
1/2	2/3	2/5	4/7
1/2	3/4	3/7	3/5
2/3	3/4	6/11	12/17

Puncturing is not used so often in block turbo codes or turbo product codes as convolutional turbo codes since for the former, it is easy to get any code rate from below rate 1/3 to as high as rate 0.98 and another reason is that performance will be lost when using puncturing techniques for increasing the data rate.

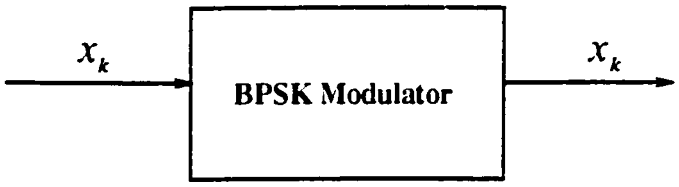
2.3 Iterative Decoding of Turbo Codes

The turbo code decoder uses iterative soft-in/soft-out decoding. The basic idea of iterative decoding can reduce the bit error rate by improving the decoding performance. Turbo code's iterative decoding scheme is based on *a-posteriori* probability (APP) decoding principle. Because *maximum a posteriori (MAP) decoding* [16] is optimal algorithm for turbo code decoding and plays an essential role in some iterative decoding algorithms including turbo decoding. The *MAP* decoding, [see detail in section 2.4.1] was originally defined as a decoding scheme to find a code vector $\vec{v} \in C$, which maximizes $Pr(\vec{v}|\vec{r})$, i.e., is the APP that, for the received sequence $\vec{r}$, the code vector $\vec{v}$ has been sent.

2.3.1 Log-Likelihood Ratio

To understand the iterative turbo decoding well, we must know some elementary concepts of the algebra used in decoding.

Let $U = (u_1, u_2, ..., u_N)$ be a binary random sequence representing the information bits in Figure 2.4. In the systematic encoders, one of the outputs $X_s = (x_1^s, x_2^s, ..., x_N^s)$ is identical to the information sequence U. The other is the parity information sequence output $X_p = (x_1^p, x_2^p, ..., x_N^p)$. We use Binary Phase Shift Keying (BPSK) modulation (Figure 2.7) and an AWGN channel with noise spectrum density $N_0/2$ (Figure 2.8). To avoid introducing another variable, we denote the



$$x_k = 2x_k - 1$$

Figure 2.7: BPSK modulator

modulator's input and output both by x_k since these are two different representations for the same random variable. The noisy versions of the output sequences are

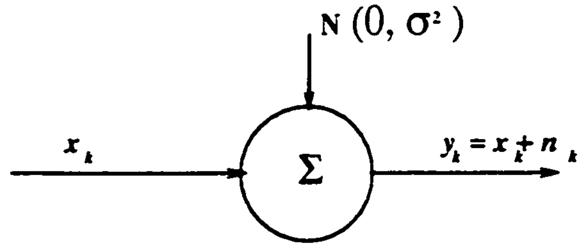


Figure 2.8: AWGN channel with distribution $\mathcal{N}(0,\ \sigma^2)$

$y_s = (y_1^s, y_2^s, ..., y_N^s)$, and $y_p = (y_1^p, y_2^p, ..., y_N^p)$,where, the kth component of the received sequence y_k is given by

$$y_k = x_k + n_k. \tag{2.8}$$

where n_k is a sample of a Gaussian noise with zero mean and variance σ^2.

The Log-Likelihood Ratio (LLR) of a binary random variable X, $L_X(x_k)$, is defined as

$$L_X(x_k) = \log \frac{P_X(x_k = +1)}{P_X(x_k = -1)}. \tag{2.9}$$

where $P_X(x)$ represents the probability that the random value X takes on the value x. The left side of definition equation 2.9 will be indicated as the "*soft*" value. In addition, further for the log-likelihood ratio of the information bit u_k[1] conditioned only by the received symbol y_k is

$$L_{U|Y}(u_k|y_k) = \log \frac{P_U(u_k = +1|y_k)}{P_U(u_k = -1|y_k)}. \tag{2.10}$$

Using Bayes' rule [11] we get

$$P_y(U|Y) = \frac{P_y(U) \cdot P_y(Y|U)}{P_y(Y)} \tag{2.11}$$

[1]As in BPSK modulator, information bit "0" and "1" will be mapped to modulated sequence "−1" and "1", respectively. We use modulated sequence for the information sequence for simplicity.

applying this to Equation 2.10 and since the probability $P_y(Y)$ can be canceled out. the Equation 2.10 can be expressed as

$$\begin{aligned} L_{U|Y}(u_k|y_k) &= \log\left(\frac{P_U(u_k=+1)}{P_U(u_k=-1)}\cdot\frac{P_{Y|U}(y_k|u_k=+1)}{P_{Y|U}(y_k|u_k=-1)}\right) \\ &= \log\frac{P_U(u_k=+1)}{P_U(u_k=-1)}+\log\frac{P_{Y|U}(y_k|u_k=+1)}{P_{Y|U}(y_k|u_k=-1)} \\ &= L_U(u_k)+L_{Y|U}(y_k|u_k). \end{aligned} \tag{2.12}$$

The first part in Equation 2.12 is called the *a priori* value for the information bit u_k corresponding to the a priori probability.

2.3.2 Soft Channel Outputs

For an AWGN channel, the Probability Density Function (pdf) is

$$P(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-m)^2\right). \tag{2.13}$$

where m and σ are the mean and the variance of the noise, respectively. The conditional pdf can be written as,

$$P(y_k|u_k=+1)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-1)^2\right). \tag{2.14}$$

$$P(y_k|u_k=-1)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x+1)^2\right). \tag{2.15}$$

We can now compute with Equations 2.14 and 2.15 applied

$$\log\frac{P_{Y|U}(y_k|u_k=+1)}{P_{Y|U}(y_k|u_k=-1)}=\frac{2}{\sigma^2}\cdot y_k=4\frac{E_s}{N_0}\cdot y_k. \tag{2.16}$$

Using the above result in 2.12 we obtain

$$L_{U|Y}(u_k|y_k)=4\frac{E_s}{N_0}\cdot y_k+\log\frac{P_U(u_k=+1)}{P_U(u_k=-1)}, \tag{2.17}$$

where $E_s$is the energy per symbol. We make the following definitions:

$$L_c=4\frac{E_s}{N_0}, \tag{2.18}$$

which is named the reliability value of the channel. There is a relationship between E_s energy per symbol and E_b energy per information bit

$$E_b = \frac{E_s}{R}, \tag{2.19}$$

since $1/R$ is the number of transmitted symbols per information bit. Hence,

$$L_c = 4\frac{E_b \cdot R}{N_0}. \tag{2.20}$$

Using the above notation we obtain

$$L_{U|Y}(u_k|y_k) = L_c \cdot y_k + L_U(u_k). \tag{2.21}$$

2.3.3 Principles of Iterative Decoding

As illustrated in the figure 2.4 and described above, the outputs of the encoder are the uncoded bit $X_s = U$, and the coded bit X_p. These outputs are modulated with a BPSK or QPSK modulator and sent through an AWGN channel. At the receiver side, we define the log-liklihood ratio, $L(u_k)$, as

$$L(U_k) = \log\frac{P_Y(u_k = +1|observation)}{P_Y(u_k = -1|observation)}, \tag{2.22}$$

where $P_Y(u_k = i|observation)$, $i = +1, -1$ is the APP of the information bit u_k. If we consider the Log-Likelihood Ratio of the information bit u_k conditioned on the whole observation sequence, we can rewrite it in a form similar to equation 2.21

$$L_{U|Y}(u_k|y_k) = L_c \cdot y_k + L_U(u_k) + L_e(u_k), \tag{2.23}$$

where $L_e(u_k)$ is the extrinsic information that is derived from all y_k and all u_i, $L_U(u_i)$, and i is different from k, $i \neq k$.

Turbo code decoder depicted in Figure 2.9, consists of two elementary decoders (Decoder #1 and Decoder #2) in a serial concatenation scheme. Both decoders in Turbo Codes are soft-in/soft-out decoders. For the first decoder, also called the

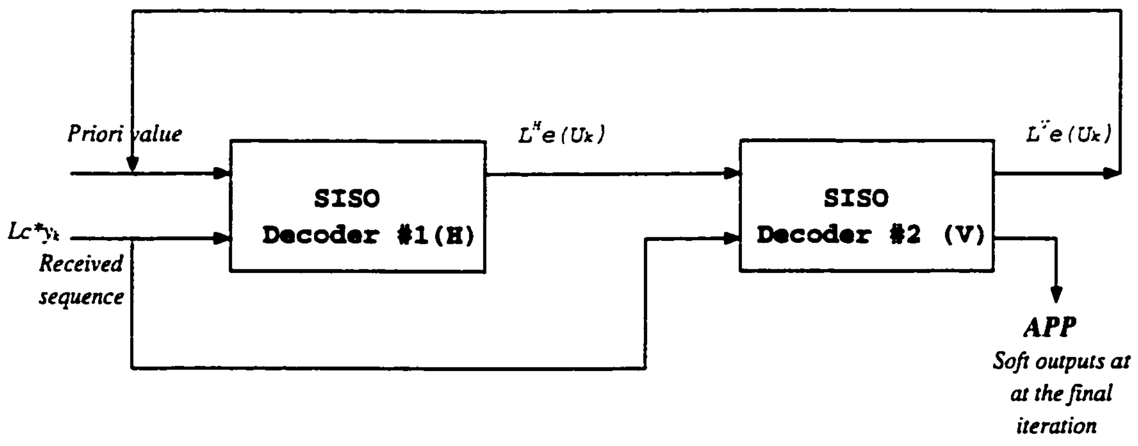


Figure 2.9: Iterative decoding procedure with two SISO decoders

horizontal decoder (detail in section 2 of Chapter 3): first half of an iteration, *a priori value (information)* in the initial iteration $L_U(u_k) = 0$, so we have

$$L_{U/y}(u_k/y_k) = L_c \cdot y_k + L_e^{-}(u_k). \tag{2.24}$$

where $L_e^{-}(u_k)$ is the extrinsic information output from the first (horizontal) decoder. For the decoder in the vertical dimension, which is the second half of an iteration, the *a priori value* is replaced by the extrinsic information from the decoder in the horizontal dimension and its output can be expressed as

$$L_{U/y}(u_k/y_k) = L_c \cdot y_k + L_e^{-}(u_k) + L_e^{|}(u_k), \tag{2.25}$$

where $L_e^{|}(u_k)$ is the extrinsic information output from the second (vertical) decoder, which will become the *a priori value* $L_U(u_k)$ for the next iteration and so on. At the beginning the extrinsic informations are statistically independent and the gain from first iteration to the second is significant. However, since the same information is used, the gain improvement after a few iterations becomes very small. After the last iteration, the soft output decision will be the sum of the last two extrinsic values and $L_c \cdot y_k$ as shown in equation 2.25. The decoder will make a hard decision using the following equation,

$$\widehat{u}_k = sgn[L(\widehat{u}_k)], \tag{2.26}$$

the decoded information bit $\widehat{u}_k$ is 1 when $L(\widehat{u}_k)$ is greater or equal than 0, otherwise is 0.

2.4 Decoding Algorithms for Turbo Codes

For the decoding of binary convolutional codes and block codes, many iterative soft-decision decoding algorithms have been derived. Most of these algorithms are devised to find the best (or the most likely) codeword in a series of candidates which are usually generated by means of simple decoders, such as where we presented in Section2.3, in successive iterative steps by using of information on the reliability measures L_c of the received sequence. There are several decoding algorithms for turbo codes so far, which are listed as below:

1. *MAP*: which is the optimal algorithm with a very high computation complexity for practical applications.

2. *MAX-Log-MAP*: which is the suboptimal algorithm and derived from MAP algorithm, its characteristic is that avoids a lots of computation though some performance sacrificed as compared to MAP algorithm.

3. *MAX*-Log-MAP*: which is the suboptimal algorithm, too. It is mainly investigated in this thesis report. The difference between MAX-Log-MAP and MAX*-Log-MAP is that the algorithm of BER performance with star is better than MAX-Log-MAP because the correction term is included in the computation; furthermore the coding gain loss is negligible as compared to MAP algorithm. This suboptimal algorithm is a good compromise between performance and complexity, thus it is very attractive for implementation.

4. *Soft-In/Soft-Out Viterbi (SOVA)*: which is the suboptimal algorithm with low complexity.

5. *Chase*: which is used for the Block Turbo Codes (BTCs) especially for Turbo Product Codes. It is a suboptimal algorithm for the performance and hardware implementation especially for big block codes [6] though the performance is not as good as MAP algorithm.

2.4.1 MAP Algorithm

The first MAP decoding algorithm, is the *symbol-by-symbol MAP algorithm* was proposed by L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv [16], which are known as the *BCJR algorithm*. It is an optimal algorithm based on *Trellis* for estimating the states or the outputs of a system observed in AWGN noise channel.

Trellis diagram [2] was first introduced by Forney in 1967 as a conceptual means to better explain the *Viterbi* algorithm. A trellis T is an edge-labeled directed graph with expanding the state diagram of the encoder in time (*i.e.*, to represent each time unit with a separate state diagram). The trellis diagram shows the time progression of the state sequences. For every state sequence S there is a unique path through the trellis diagram, and vice versa. We list some notations used in *BCJR algorithm*:

- s : state at level k in the trellis
- s' : state at level $k-1$ in the trellis
- S^{+1} : all (s', s) transition caused by $u_k = +1$
- S^{-1} : all (s', s) transitions caused by $u_k = -1$
- $\gamma_k(s', s)$: branch transitions probability, whenever a transition between s' and s exists

$$\gamma_k(s', s) = P(s|s') \cdot p(y_k|s', s) = p(y_k|u_k) \cdot P(u_k) \tag{2.27}$$

- $\alpha_k(s)$: forward recursion of the MAP algorithm yields

$$\alpha_k(s) = \sum_{s'} \gamma_k(s', s) \cdot \alpha_{k-1}(s'). \tag{2.28}$$

- $\beta_{k-1}(s')$: backward recursion of the MAP algorithm yields

$$\beta_{k-1}(s') = \sum_{s} \gamma_k(s', s) \cdot \beta_k(s). \tag{2.29}$$

The MAP algorithm can be applied only if the sequence length is finite and we assume that at the start and at the end of the observed sequence all paths converge at the zero state, then we have the following initial value for α and β

$$\alpha_{start}(0) = 1, \quad and\ \alpha_{start}(k) = 0 \qquad , \forall k \neq 0. \tag{2.30}$$

$$\beta_{end}(0) = 1, \quad and\ \beta_{end}(k) = 0 \qquad , \forall k \neq 0. \tag{2.31}$$

The output of the MAP decoder is defined as the a posteriori log-likelihood ratio,

$$\begin{aligned} L(\hat{u}) = L(u|y) &= \log \frac{P(u=+1|y)}{P(u=-1|y)} \\ &= \log \frac{\sum_{S^{+1}} p(s', s, y)}{\sum_{S^{-1}} p(s', s, y)}. \end{aligned} \tag{2.32}$$

The sums of the joint probabilities $p(s', s, y)$ in the numerator and in the denominator of 2.32 are taken over all branches in S^{+1} and S^{-1}, respectively. We expand the above equation assuming a memoryless transmission channel

$$\begin{aligned} p(s', s, y) &= p(s', y_{j<k}) \cdot p(s, y_k|s') \cdot p(y_{j>k}|s) \\ &= \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s) \end{aligned} \tag{2.33}$$

where $y_{j<k}$ expresses the sequence of received symbols until time $k-1$ and $y_{j>k}$ is the received sequence after time k until end of the trellis. The Equation 2.32 presents the fact that each bit decision is affected by received value of both *prior and future* symbols.

The final decision depends on the sign of $L(\widehat{u})$ in Equation 2.32: the estimated information bit will be "1" when the sign of $L(\widehat{u})$ is positive and "0" otherwise.

Though turbo codes can achieve excellent performance by iteratively execution a MAP decoding algorithm many times, the complexity of the decoding algorithm is significant for realization of efficient turbo decoders because of the *logarithmic* functions, of mixed multiplications and additions of these values.

2.4.2 MAX-Log-MAP Algorithm

As we mentioned earlier that MAP algorithm is a very good decoding scheme for turbo codes, but is not very practical because of its complexity. Efforts have been made to reduce the decoding complexity of the MAP algorithm. These attempts include a suboptimal realization of the BCJR algorithm, *i.e.*, the MAX-log-MAP algorithm and the SOVA algorithm.

In MAX-Log-MAP algorithm, the same computations as the MAP algorithm is carried out in the log-likelihood domain. Hence a multiplication of probabilities $X \times Y$ is replaced by an addition $\log X + \log Y$, and then addition of probabilities is replaced by a max-operation given as,

$$\log(e^X + e^Y) \approx max(X, Y). \tag{2.34}$$

Using the above approximation equation, we do not calculate $\gamma_k(s', s)$, $\alpha_k(s)$, $\beta_{k-1}(s')$ and will work with $\log\gamma_k(s', s)$, $\log\alpha_k(s)$, $\log\beta_{k-1}(s')$ and then the forward recursion and the backward recursion are now represented in the additive forms

$$\log\gamma_k(s', s) = \log\left(p(y_k|u_k) \cdot P(u_k)\right) \tag{2.35}$$

$$\log\alpha_k(s) \approx \max_{s'}\left(\log\gamma_k(s', s) + \log\alpha_{k-1}(s')\right), \tag{2.36}$$

$$\log\beta_{k-1}(s') \approx \max_{s}\left(\log\gamma_k(s', s) + \log\beta_k(s)\right) \tag{2.37}$$

with the initial conditions

$$\log \alpha_{start}(0) = 0, \quad and \ \log \alpha_{start}(k) = -\infty, \tag{2.38}$$

$$\log \beta_{end}(0) = 0, \quad and \ \log \beta_{end}(k) = -\infty. \tag{2.39}$$

By substituting values for $\log \gamma_k(s', s)$, $\log \alpha_k(s)$, $\log \beta_{k-1}(s')$, the log-likelihood $L(\widehat{u})$ can be expressed

$$L(\widehat{u}) = L(u|y) = \log \frac{\sum_{S+1} e^{\log \alpha_{k-1} + \log \gamma_k(s', s) + \log \beta_k(s)}}{\sum_{S-1} e^{\log \alpha_{k-1} + \log \gamma_k(s', s) + \log \beta_k(s)}} \tag{2.40}$$

This expression can be simplified by using the approximation

$$\log(e^{\delta_1} + e^{\delta_2} + \ldots + e^{\delta_n}) \approx \max_i \delta_i \tag{2.41}$$

where $i \in \{1, 2, \ldots, n\}$ and $\max \delta_i$ can be computed by successively calculating $(n-1)$ maximum function over only two values. The Equation2.40 can be approximated by

$$L(\widehat{u}) = \max_s \left[\log \alpha_{k-1} + \gamma_k(s', s) + \beta_k(s)\right] - \max_{s'} \left[\log \alpha_{k-1} + \gamma_k(s', s) + \beta_k(s)\right]. \tag{2.42}$$

From the above equations, we can easily know that the decoding complexity is reduced a lot because by performing this algorithm in the logarithmic domain, the multiplication becomes addition and exponentials disappear. Therefore, these approximations indeed avoid computing the actual probabilities, and simplify some computations. At the same time, when we get a reasonable decoding complexity scheme for hardware implementation, we should expect that some coding gain are lost as compare to the MAP algorithm. The loss of coding gain will be shown through the simulation results in Section 2.4.3.

2.4.3 MAX*-Log-MAP Algorithm

This algorithm is also derived from MAP algorithm. As shown in Section 2.4.1, most computations are based on the logarithm of a sum of exponentials. Such an expression can be exactly computed, two terms at the time, using the *Jacobian* algorithm [17], as

$$\log(e^X + e^Y) = \max(X, Y) + \log(1 + e^{-|X-Y|}) \tag{2.43}$$

That is, the expression is computed as the maximum exponent and a correction term, which is a function of the absolute difference between exponents. With the correction term, the performance of this algorithm is better than the pure approximated algorithm. We denote the decoding algorithm with correction term using Jacobian algorithm, MAX*-Log-MAP algorithm.

If the correction term is omitted and only the max term is used to compute α, β, and the extrinsic value of log-likelihood $L(\hat{u})$, then we can obtain the MAX-Log-MAP approximation algorithm, which was described in previous Section. This algorithm does improve the performance as compared to the one without correction term. However, the correction term is a nonlinear function, which still brings the difficulty to practical implementation. We then find another way to reduce complexity and small BER performance degradation compared with MAP algorithm. Typically, this is implemented via a small lookup table, as given by table 2.2, which is addressed

Table 2.2: Look-up table for correction term used in MAX*-Log-MAP algorithm

$\lvert X-Y\rvert$	0.0625	0.5	1.0	1.5	1.75	2.25	≥ 2.25
$\delta_{X,Y}$(decimal)	0.6875	0.5625	0.375	0.25	0.1875	0.125	0.0
$\delta_{X,Y}$(binary)	0.1011	0.1001	0.0110	0.0100	0.0011	0.0010	0.0000

based on the difference of the two operands, A and B. Though it requires extra computation and more importantly, increasing the number of instructions in the loop as well as the number of registers used, more accurate computation yields a better performance. Where $\delta_{X,Y}$ in table 2.2 is equivalent to the term $\log(1 + e^{-|X-Y|})$ in equation 2.43 and we use $\delta_{X,Y}$ to represent the correction term for simplicity. We give two forms in decimal and binary, respectively. The four-bit-long binary form is constructed to represent fractional part for hardware implementation. From the

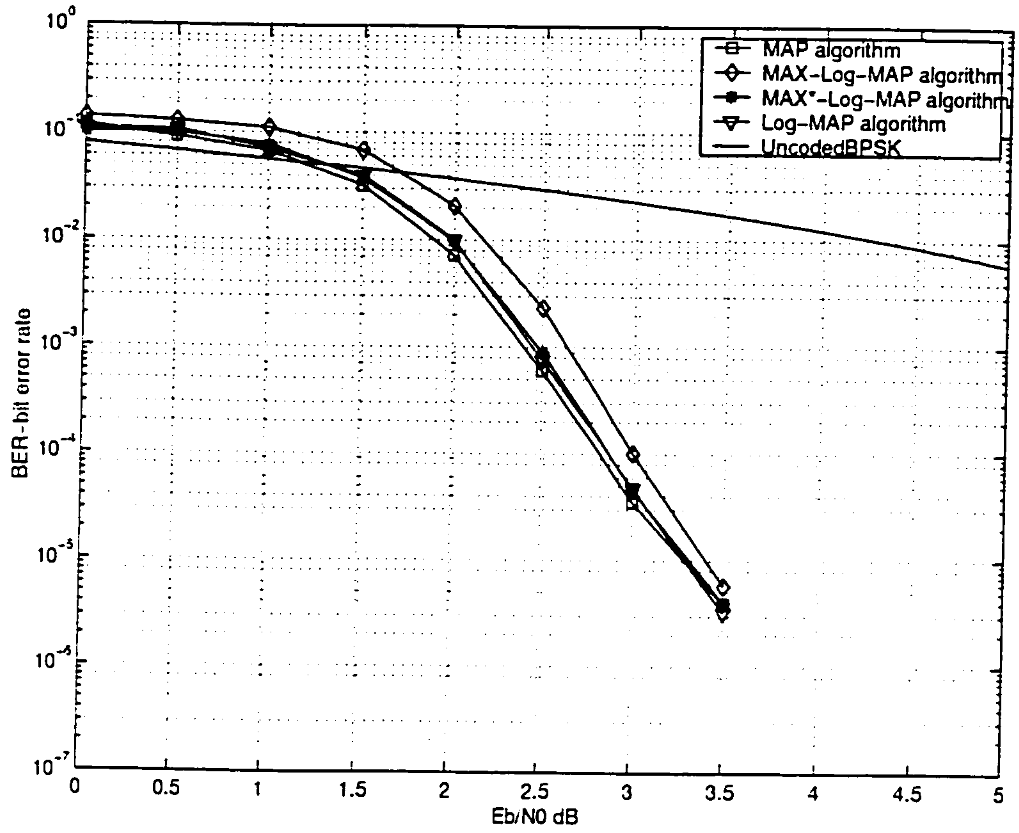


Figure 2.10: Comparison with MAP, MAX-Log-MAP, MAX*-Log-MAP and Log-MAP algorithm for $RM(32,26)^2$, iteration 5, R=0.68.

Figure 2.10 we can see the coding gain loss is negligible for MAX*-Log-MAP algorithm, and there is about 0.15 dB degradation between MAX-Log-MAP algorithm and MAP algorithm at BER of above 10^{-5}. For MAP algorithm, at BER of 10^{-5}, there is a coding gain of 6.3 dB from uncoded . We will present more detail in Chapter 3.

2.4.4 SOVA

This algorithm is another suboptimal decoding strategy for turbo codes, which was first introduced by J. Hagenauer and P. Hoeher [18]. It is a soft output Viterbi algorithm (VA), which is an extension of the Viterbi algorithm[19] and has an implementation advantage over MAP because it is much less complex than MAP algorithm.

We have to start with the VA because it is the basis of SOVA. The VA was originally proposed for decoding of convolutional codes by A. J. Viterbi in April of 1967 [19]. The VA is similar to BCJR algorithm, performs estimation of the input sequence of a discrete time finite-state Markov process observed in memoryless noise. However, BCJR algorithm needs forward recursion and backward recursion to calculate the estimated information sequences and it is applied only if the sequence length is finite, while the VA can be used for any sequence length, by truncating the survivors. So it is used as a solution to various communication estimation problems as long as the system can be represented by a time invariant or time varying trellis diagram. For each step and for each node of the trellis, there is always 2 path merging and in these 2 paths, only one is kept, the one with the highest cumulative metric (the most likely one) will be selected as the survivor. Maximum likelihood detector is used in the VA. In Equation 2.11 of Section 2.3.1, assuming that the signals are equally likely, it suffices for the receiver to maximize the likelihood function $P_y(Y|U)$, this results in the *maximum likelihood* decoder. The probability $P_y(Y|U)$ for the received sequence of length n can be expressed as

$$\begin{aligned} P_y(Y|U) = P_y(y_1^\tau|u_1^\tau) &= \prod_{t=1}^{\tau} P_y(y_t|u_t) \\ &= \prod_{t=1}^{\tau}\prod_{k=0}^{n-1} \frac{1}{\sqrt{2\pi}} e^{-\frac{(y_{t,k}-u_{t,k})^2}{2\sigma^2}}. \end{aligned} \tag{2.44}$$

We introduce the log function and the Equation 2.44, the above equation becomes

$$\begin{aligned}\log P_y(Y|U) &= \sum_{t=1}^{\tau} \log P_y(y_t|u_t) \\ &= -\frac{n\tau}{2}\log(2\pi) - n\tau\log\sigma - \sum_{t=1}^{\tau}\sum_{k=0}^{n-1}\frac{(y_{t,k}-u_{t,k})^2}{2\sigma^2}, \qquad (2.45)\end{aligned}$$

maximizing $P_y(Y|U)$ is equivalent to minimizing the *Euclidean* difference

$$\sum_{t=1}^{\tau}\sum_{k=0}^{n-1}(y_{t,k}-u_{t,k})^2 \qquad (2.46)$$

between the received sequence y_1^τ and the modulated sequence u_1^τ in the trellis diagram. We name the Euclidean distance as *branch metric*

$$\mu_t^{(i)} = \sum_{k=0}^{n-1}(y_{t,k}-u_{t,k})^2, \qquad (2.47)$$

then the *path metric* corresponding to the path i, denoted by M_t^i, is given by

$$M_t^{(i)} = \sum_{t'=1}^{t}\mu_t^{(i)} = M_{t-1}^{(i)} + \mu_t^{(i)}, \qquad (2.48)$$

the computation is based on keeping only one path per node with the minimum path metric at each time instant. The decision for the estimated information sequence is made at the final time instant τ and the maximum likelihood path is chosen as the survivor in the final node.

The SOVA estimates the soft output information for each transmitted binary symbol in the form of the log-likelihood function

$$L(\widehat{u}_t) = \log\frac{P(u_t=+1|y_1^\tau)}{P(u_t=-1|y_1^\tau)}, \qquad (2.49)$$

it makes a hard decision like the Equation 2.26 that are described in Section 2.3.3. the sign of $L(\widehat{u}_t)$ determines the hard estimate at time t and its absolute value represents the soft output information that can be used for decoding in the next stage.

2.4.5 Chase Algorithm

This algorithm was first discovered by D. Chase in 1972 [20] for block codes (n, k, d), where d is the minimum distance. It can be used under the condition when the encoding/decoding processing can not be represented by trellis diagram like non-binary block code: Reed-Solomon codes. R. M. Pyndiah [6] shows that BTC using the Chase algorithm had performances comparable to those of CTC.

The soft-input/soft-output concept is also used in Chase decoding algorithm for Turbo Codes except that the decoder is replaced by a Chase decoder comparing with another decoding algorithm. The soft output of the decoder is an estimation of the log-likelihood ratio of the binary decisions given by the Chase decoder. Let us assume the transmission of block coded binary sequences $\vec{X}$ using QPSK or BPSK over a AWGN channel, the received sequence $\vec{Y} = (y_1, \dots y_n)$ can be modeled by:

$$\vec{Y} = \vec{X} + \vec{N}, \tag{2.50}$$

where $\vec{N} = (n_1, \dots, n_n)$ are the samples of standard deviation σ. From the received symbols $\vec{Y}$, the decoded optimum sequence $\vec{S} = (s_1, s_2, \dots, s_n)$ corresponding to the $\vec{X}$ buy using maximum-likelihood decoding (MLD) is given by:

$$\vec{S} = \vec{C^i},\ if\ \left|\vec{Y} - \vec{C^i}\right|^2 \leq \left|\vec{Y} - \vec{C^j}\right|^2 \quad \forall j \neq i, \tag{2.51}$$

where $i, j \in [1, \dots, 2^k]$ and

$$\left|\vec{Y} - \vec{C^i}\right|^2 = \sum_{i=1}^{n}(y_i - c_i)^2 \tag{2.52}$$

is the squared Eucliden distance between $\vec{Y}$ and $\vec{C^i}$, where $\vec{C^i} = (c_1^i, \dots, c_n^i)$ is the ith codeword of $\vec{C}$.

In 1994, R. Pyndiah [5] supplemented this algorithm to compute the soft decisions associated with the maximum likelihood sequence $\vec{S}$, which gives a measure of the reliability of each component of $\vec{S}$. This reliability function is represented by the LLR of the decision S_j (jth element of $\vec{S}$). Let us consider the decoding of the BTC $\overrightarrow{C}$ in two-dimensional: $\overrightarrow{C} = \overrightarrow{C_1} \otimes \overrightarrow{C_2}$. It is transmitted through an AWGN channel using BPSK signaling. The iterative turbo decoding process can be achieved by cascading several elementary Chase decoders. The soft input for the decoding of the columns or rows at the second decoding of $\overrightarrow{C}$ is given by

$$\overrightarrow{Y'(m)} = \overrightarrow{Y} + \alpha(m) \cdot \overrightarrow{W(m)}, \tag{2.53}$$

where $\overrightarrow{Y}$ is defined as before, m is the mth elementary decoder, $\overrightarrow{W(m)}$ is the vector which contains the extrinsic information computed by the previous decoder (which is the difference between the output information and the input information) and $\alpha(k)$ is the scaling factor determined by simulation. Paper [6] and [21] tell us how to choose the scaling value of α.

2.5 Summary

In this Chapter, we started with the classical serial concatenation of convolutional codes and Reed-Solomon codes, then introduced turbo codes, which use a parallel concatenation of two codes. We briefly presented the history of turbo codes from its discovery, development and the applications. Turbo codes have been implemented in hardware for use in the broadband wireless communication system, ATM and MPEG even IP because of their impressive performance.

Some fundamental elements of Turbo Codes were described in Section 2.2 such as turbo code concatenation methods, the related concept of interleaving techniques with a few popular classes of interleaving methods, the procedure for designing codes with different rates using puncturing.

In this chapter, we introduced many basic equations for developing common turbo code iterative decoding algorithms and the principle of iterative decoding in Turbo codes. In future Chapters, we will show the simulation results concerning the impact of the number of iterations in turbo code decoding and give simple analysis of it. A few approaches based on bit-by-bit MAP decoding are discussed in the following chapters. Among them, the most important concepts are MAP decoder and the soft value for turbo codes decoding. Each algorithm has its own merits and disadvantages in terms of complexity and performance. The simulation results show the different BER performance for a given SNR. As a compromise between the complexity and the performance, we have chosen the MAX*-Log-MAP algorithm in our research.

Chapter 3

The Reed-Muller Block Turbo Codes

3.1 The Reed-Muller Codes

3.1.1 Introduction

The Reed-Muller Codes (RM codes) form a class of linear block codes over *Galois Fields*, $GF(2)$ (For the definition of GF(2), see the book of Shu Lin and Costello[2]). The Reed-Muller codes and their decoding algorithm were discovered by Reed and Muller, respectively in 1954 [22], [23]. These codes can be easily constructed for a wide range of rates and minimum distances. These codes can be decoded effectively with trellis-based decoding algorithm [30]. This allows us to use BCJR algorithm directly. RM Codes include the extended Hamming codes, which are the best known linear block codes, too.

3.1.2 Construction of the Reed-Muller codes

Linear block codes, which are specified by their code length n and information data length k. RM code [24], [25] can be specified in the same way. Moreover, it is defined very simply in terms of Boolean functions. For any m and r, let r satisfy $0 \leq r \leq m$, there is a RM code of blocklength 2^m called the $rth - order$ RM Code of blocklength 2^m. The RM code is usually denoted by $\Re(r, m)$, which is different from usual description of linear block codes in terms of n and k. There is a code in this class for which

$$\left.\begin{aligned} n &= 2^m, \\ k &= 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}, \\ n - k &= 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{m-r-1}, \\ d &= 2^{m-r} = \mathit{minimum\ distance\ (weight)} \end{aligned}\right\} \tag{3.1}$$

RM code will be defined by a procedure for constructing its generator matrix: we will construct a nonsystematic generator matrix that will prove convenient for decoding. First, define the product of two vectors f and g by a component-wise multiplication. That is, let

$$\begin{aligned} \vec{f} &= (f_0, f_1, \ldots, f_{n-1}), \\ \vec{g} &= (g_0, g_1, \ldots, g_{n-1}), \end{aligned} \tag{3.2}$$

then the product is the vector

$$\vec{fg} = (f_0 g_0, f_1 g_1, \ldots, f_{n-1} g_{n-1}). \tag{3.3}$$

We define the generator matrices for the $rth - order$ RM code of blocklength 2^m as an array of blocks:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_k \end{bmatrix}, \tag{3.4}$$

where G_0 is the vector of length $n = 2^m$ containing all ones; G_1, an m by 2^m matrix that has each binary m-tuple appearing once as a column; and G_l is constructured from G_1 by taking its rows to be all possible products of l rows of G_1. For definiteness, we take the leftmost column of G_1 to be the all zero vector, the rightmost to be the all one vector, and the others to be the binary m-tuples in increasing order, with the low-order bit in the bottom row. So in general, the generator matrix of the $rth-order$ RM code consists of all linear combinations of the vectors corresponding to the products

$$1,\ v_1,\ ...,\ v_{m,}\ v_1v_2,\ v_1v_3,...,\ v_{m-1,}\ v_m,\ ...(up\ to\ degree\ r)$$

which form a k basis vectors for the code. Since there are $\begin{pmatrix} m \\ l \end{pmatrix}$ ways to choose the l rows in a product, G_l is an $\begin{pmatrix} m \\ l \end{pmatrix}$ by 2^m matrix and it provides the rows of G, which are linearly independent. As an example for $r = 2$ and $m = 4$, the generator matrix for the $2nd - order$ RM code of blocklength 16 is the 11 by 16 matrix:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix}, \tag{3.5}$$

and each G_l is

$$G_0 = [1111111111111111] = [f_0]$$

$$G_1 = \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0000000000001111 \\ 0000000000110011 \\ 0000000001010101 \\ 0000001100000011 \\ 0000010100000101 \\ 0001000100010001 \end{bmatrix} = \begin{bmatrix} f_1 f_2 \\ f_1 f_3 \\ f_1 f_4 \\ f_2 f_3 \\ f_2 f_4 \\ f_3 f_4 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 0000000000000011 \\ 0000000000000101 \\ 0000000000010001 \\ 0000000100000001 \end{bmatrix} = \begin{bmatrix} f_1 f_2 f_3 \\ f_1 f_2 f_4 \\ f_1 f_3 f_4 \\ f_2 f_3 f_4 \end{bmatrix}$$

This generator matrix gives

$$k = 1 + \binom{4}{1} + \binom{4}{2} = 11,$$

$$n = 2^m = 16,$$

RM(16,11) over GF(2).

3.1.3 Minimal Trellis for Linear Block Codes

In 1974, Bahl *et al* [16] introduced an optimal decoding algorithm with a method of representing the words in an arbitrary linear block code by the path labels in a trellis. In 1978, Wolf [26] introduced a trellis for block codes and showed that it could be used to implement the Viterbi algorithm for Maximum likelihood decoding of an arbitrary block code and a trellis diagram for a linear block code. The trellis diagram for block codes is time-variant [30], whereas the one for convolutional code is time-invariant. Because the Forney trellis minimized the number of vertexes and edges at each depth, Muder [27] called the Forney trellis the "*minimal*" trellis for the code. Later, Massey [28] made a further work on how to represent a block code by a trellis and gave an alternative solution. At the same time, a lot of research developed the properties of the important "*Trellis-oriented*" generator matrices. So far, the theory of "minimal trellises" has been applied successfully to reduce the decoding complexity of Turbo Codes. To use BCJR algorithm for a linear block code, we have to use the trellis to represent this block code and in order to reduce the complexity, we use minimal trellis with the method of Massey's construction. We will present the theory of *"minimal-span" generator matrices* (MSGM's) [29] in the next subsections.

3.1.3.1 Minimal-Span Generator Matrices (MSGM)

We begin with some definitions. If $\overrightarrow{x} = (x_1, x_2, ..., x_n)$ is a nonzero binary n-vector, then its *left index* $L(x)$ *and right index* $R(x)$ are defined as following:

- $L(x)$: the smallest index i such that $x_i \neq 0$.
- $R(x)$: the largest index i such that $x_i \neq 0$.
- $Span(x)$: the span of x, which is a discrete interval $[start, end] = (L(x), L(x)+1, ..., R(x))$.
- $spanlength(x)$: the span length of x, which is the number of elements in $Span(x)$.

Now, we can give the definition of the MSGM as following:

- *MSGM*: Let C be an (n, k) binary linear code over $GF(2)$. Among all generator matrices for C, the one for which the span length is as small as possible is called Minimal Span Generator Matrix (MSGM).

For our purpose to get the "best" trellis, we will use the Massey construction method, then we will get the corresponding generator matrix, which is called generator matrix in *row-reduced echelon* form (RRE).

- RRE form: Given a $k \times n$ matrix $G = [x_{i,j}]$ over $GF(2)$, if the rows x_1, x_2, ..., x_k of G are such that

$$L(x_1) < L(x_2) < ... < L(x_k) \tag{3.6}$$

and the k columns found at position $L(x_1)$, $L(x_2)$, ..., $L(x_k)$ in G are all of weight one: if $j = L(x_i)$, then $x_{i,j} \neq 0$ is the only nonzero entry in the $j - th$ column of G.

3.1.3.2 Construction of the Minimal Trellis From a Generator Matrix with RRE form

Every path in a trellis T represents a unique code codeword that can be easily determined by reading the branch labels along each path in T. Given a trellis, it is easy to find the code by listing the codewords by tracing the different path and reading the labels. We usually need to solve the converse problem: given a code C over $GF(2)$, there are several trellises representing the same code. Among them, we would generally like to construct the simplest and the most efficient trellis for a given code. The "best" trellis is the one or the ones with the smallest number of edges. Our purpose is to construct the minimal trellis that minimizes one ore more measures of trellis complexity for a fixed permutation of the code.

We have a linear block code C with a length n and dimension k, and we have a $k \times n$ generator matrix G with RRE form for the code C. In this matrix, we denote the left indices of its rows by γ_1, γ_2, ..., γ_k, and they have the following relationship with each other

$$\gamma_1 < \gamma_2 < ... < \gamma_k. \tag{3.7}$$

The Massey trellis T for a code C is constructed by first defining the vertices V_i with the parity check bits that are observed at time instant i while the information bits have been observed at time instant i and the rest of information bits are assumed to be 0. Let m be the largest integer such that $\gamma_m \leq i$, then

$$V_i \equiv \{(C_{i+1}, ..., C_n) : (C_1, ..., C_n) = (u_1, ..., u_m, 0, ..., 0) \cdot G\}, \qquad (3.8)$$

where $u_1, u_2, ..., u_m$ are information bits. We have

$$V_{start} = \left\{\overrightarrow{0}\right\}, \qquad V_{end} = \{\Phi\} \qquad (3.9)$$

The edges of T are defined as follows by considering two cases: one is the case of $i > \gamma_m$, the other is $i = \gamma_m$. In the first case, there is an edge e from a vertex $v' \in V_{i-1}$ to a vertex $v \in V_i$ iff there exists a codeword $(c_1, c_2, ..., c_n) \in C$, such that

$$(c_i, c_{i+1}, ..., c_n) = v' \qquad (3.10)$$

$$(c_{i+1}, ..., c_n) = v \qquad (3.11)$$

The label of this edge is c_i. For the second case of $i = \gamma_m$, there is an edge from a vertex $v' \in V_{i-1}$to a vertex $v \in V_i$ iff there exists a pair of codewords $c = (c_1, c_2, ..., c_n) \in C$ and $c' = (c_1', c_2', ..., c_n') \in C$, such that

$$(c_i, c_{i+1}, ..., c_n) = v' \qquad (3.12)$$

$$(c_{i+1}', ..., c_n') = v \qquad (3.13)$$

and in this case, there are two choices: either $c = c'$or $(c + c')$ equals the $m-th$ row of G and the label for this edge is c_i'.

3.2 The Reed-Muller Block Turbo Codes (2-dimensional)

RM block turbo codes are relatively large codes built from smaller code word blocks [33]. In this thesis, we first research two dimensional codes constructed from small component codes. Recent research and development on iterative decoding show that high rate product codes can achieve a performance close to Shannon capacity, which is similar to that of the convolutional turbo codes. In the paper [31], a rate 0.98 Hamming product code $(1023, 1013)^2$ was shown to achieve a BER of 10^{-5} within 0.27dB of the Shannon channel capacity, which was at the cost of high decoding complexity. Research has shown that block turbo codes are more efficient than convolutional turbo codes for high code rate [32]. We will show this with our simulation results in Section 3.3.

3.2.1 Construction of 2-dimensional RM Turbo Codes

A 2-dimensional RM block turbo code is constructed from smaller component code word blocks. BTCs involve serial concatenation of two or more block encoder separated by simple row and column interleaver.

Let us consider two linear block codes: $RM_1(n_1, k_1)$, $RM_2(n_2, k_2)$, where n_i and k_i represent the code word length, and the number of information bits, respectively. The structure of 2-dimensional RM block is illustrated in Figure 3.1, the $k_1 \times k_2$ information block $\overrightarrow{U}$ is ordered in a rectangular matrix. In order to encode the RM block turbo code, each information data bit is placed both into a *row* encoder and a *column* encoder. For instance, let us consider a $RM(16, 11)$ code. This code takes 11 information (input) bits, computes 5 parity bits, and appends these 5 parity bits to the information bits to create a 16 bit code word to be transmitted. We denote P_H to represent the parity check bits with each code word constructured *horizontally*, P_V are the parity check bits with each code word constructured *vertically* and $RM(n_1, k_1)(n_2, k_2)$ to represent the 2-dimensional RM block turbo code. In general, we can use different codes for both the horizontal and the vertical blocks as shown in the Figure 3.1.

The code rate for the 2-dimensional RM block turbo code is given as follows:

$$R = \frac{k * k}{n * n - (n - k) * (n - k)}. \tag{3.14}$$

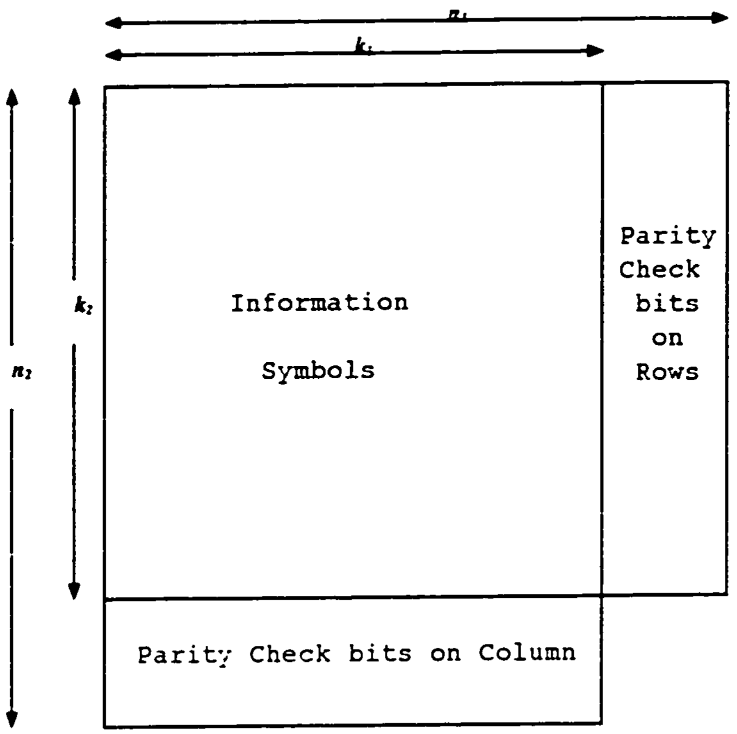


Figure 3.1: Structure of 2-dimensional RM block turbo code

also the code rate in general is given by:

$$R = \frac{k_1 * k_2}{n_1 * n_2 - (n_1 - k_1) * (n_2 - k_2)}. \tag{3.15}$$

We list the all RM codes used in this thesis in Table 3.1:

Table 3.1: Example RM block turbo codes with 2-dimensional using in this thesis

Code	Block size	Data size	Code rate
$RM(8, 4)^2$	48	16	0.33
$RM(16, 11)^2$	231	121	0.52
$RM(32, 26)^2$	988	676	0.68
$RM(64, 57)^2$	4047	3249	0.803

3.2.2 2-Dimensional RM BTC Encoders

Turbo code encoders are composed of two elementary encoders and an interleaver. Two-dimensional Turbo block encoders often employ recursive systematic convolutional encoding with two parallel-concatenated constituent encoders. In BTCs the systematic convolutional encoders are replaced by block code encoders. The encoding system model is shown in Figure 3.2. Both horizontal and vertical

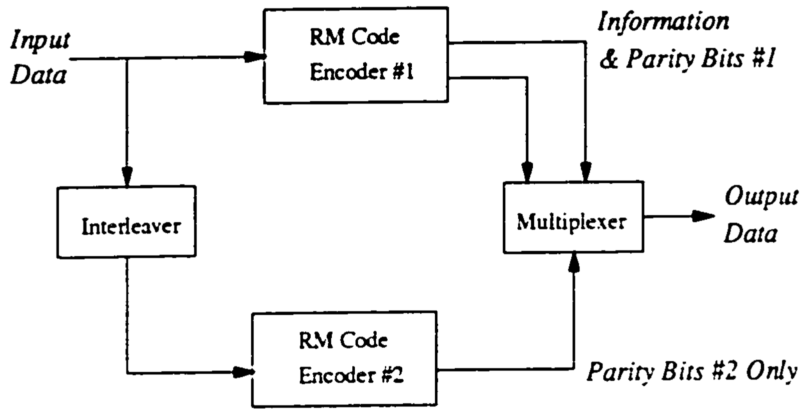


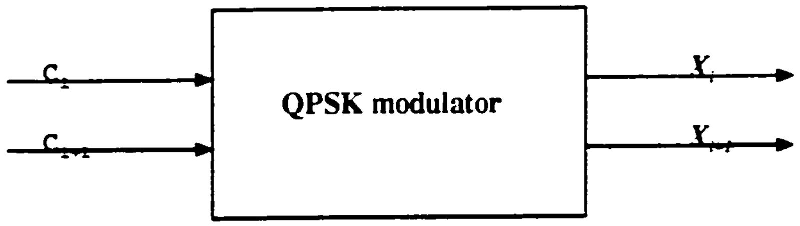
Figure 3.2: Two-dimensional RM block turbo code encoding model

block code words. $\vec{C}$, with the length, n, are generated by the information data bits. $\vec{U}$, with the length of k and generator matrix with the RRE form,

$$\vec{C} = \vec{U} \cdot G = (u_1, u_2, \ldots u_k) \cdot \begin{bmatrix} \vec{g_1} \\ \vec{g_2} \\ \vdots \\ \vec{g_k} \end{bmatrix} . \qquad (3.16)$$

The output codeword consists of three parts: information matrix of $k * k$, the parity bits, P_H and P_V. The information block is first encoded horizontally. The information data bits are then sent to the second encoder after block interleaving. The three parallel streams of data are then multiplexed into one stream and sent to the channel. It is not necessary to apply puncturing of the parity streams to get a higher code rate because it is easy to get any code rate by combining different RM block codes with different information data lengths or with multi-dimensional linear block codes. For example, we can combine $RM(8, 4)$ with $RM(16, 11)$ as 2-dimensional RM block turbo code $RM(8, 4) \times (16, 11)$ to get code rate $R = 0.407$ or we can combine $RM(16, 11)$ with $RM(32, 26)$ as a RM block turbo code $RM(32, 26) \times (16, 11)$, with code rate $R = 0.593$ and so on.

The output of the turbo encoder is the code word or code sequences, which can be BPSK (Figure 2.7) or Quadrature Phase Shift Keying (QPSK) (Figure 3.3) modulated . Then the modulated sequences are sent through the channel as shown



$$X_i = 2c_i - 1$$

$$X_{i+1} = 2c_{i+1} - 1$$

Figure 3.3: QPSK model used in 2-dimensional RM BTC

in Figure 3.4. These received sequences, which are affected by AWGN channel noise, will be decoded by BTC iterative decoding.

3.2.3 Iterative Decoding for 2-Dimensional RM BTCs

The decoding techniques used for convolutional turbo codes can also be used for BTCs. Let us consider the two dimensional case in which $k_1 \times k_2$ information data bits are ordered in a rectangular matrix. Let C_1 be a linear systematic code (n_1, k_1) in horizontal dimension and C_2 be a linear systematic code (n_2, k_2) in the vertical dimension; both $C_1$and C_2 are used to encode the same data. This two-dimensional code is a direct product of C_1 and C_2. A lot of studies investigate the iterative decoding method considering the complexity and performance. Therefore,

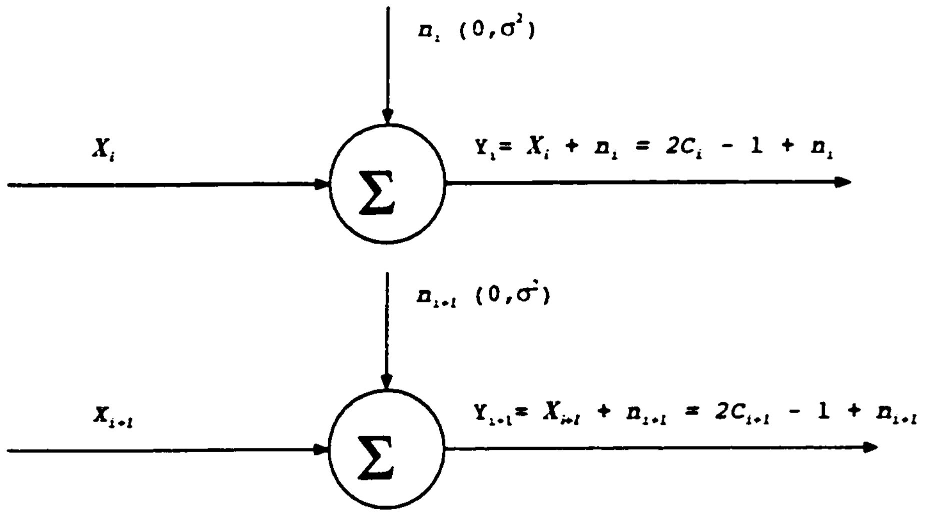

n_i $(0,\sigma^2)$
X_i
$Y_i = X_i + n_i = 2C_i - 1 + n_i$
n_{i+1} $(0,\sigma^2)$
X_{i+1}
$Y_{i+1} = X_{i+1} + n_{i+1} = 2C_{i+1} - 1 + n_{i+1}$
Σ
Σ

Figure 3.4: AWGN channel with distribution $N(0, \sigma^2)$

we first used the original MAP algorithm based on the minimal trellis to get good performance, and then used iteration a few times to make the performance better. Our idea of iterative decoding for 2-dimensional RM BTCs is to first decode the code in one dimension and then decode the code in the second dimension. This idea was introduced in the paper written by J. Hagenauer in 1996 [10]. All the ideas are illustrated in Figure 3.5. We have some notations listed as follows:

- $L_c * Y$: Information data bits with parity bits affected by AWGN channel noise.
- $L(\hat{U}) = 0$: Soft (logarithm) value of *a priori* probability of the information data bits.
- $L_e - H(\hat{U})$: Extrinsic values of the 1st dimension (horizontal) decoder
- $L_e - V(\hat{U})$: Extrinsic values of the 2nd dimension (vertical) decoder
- INT : interleaver
- $De - INT$: deinterleaver

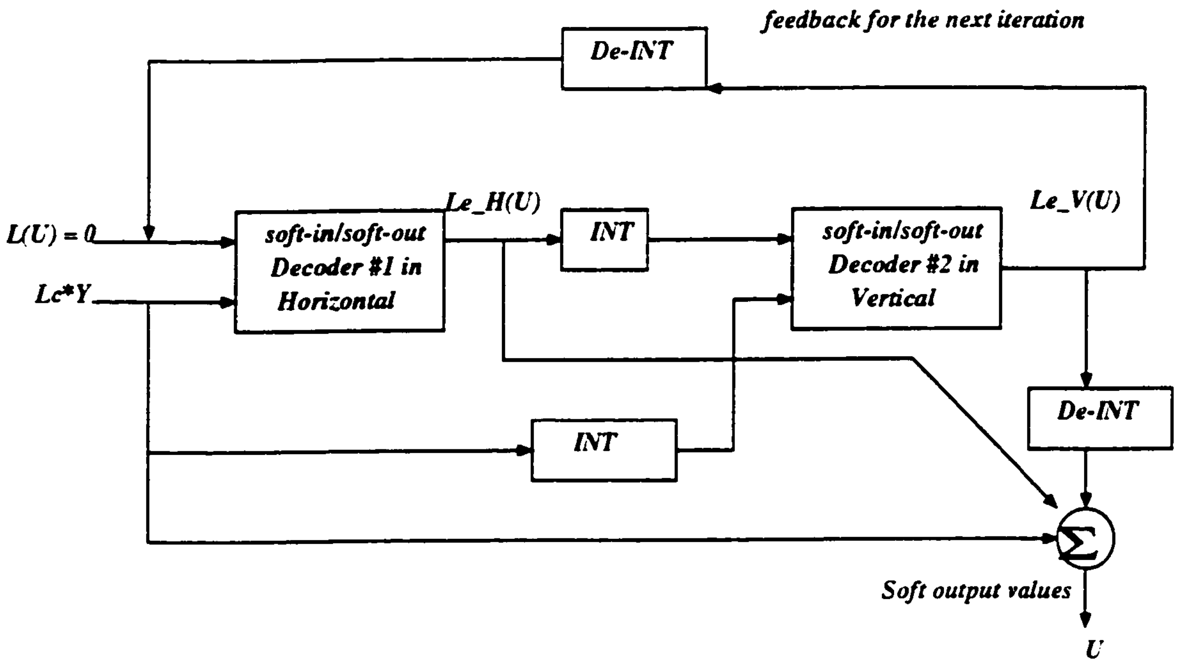

feedback for the next iteration
De-INT
L(U) = 0
Lc*Y
soft-in/soft-out Decoder #1 in Horizontal
Le_H(U)
INT
soft-in/soft-out Decoder #2 in Vertical
Le_V(U)
De-INT
INT
Σ
Soft output values
U

Figure 3.5: Iterative decoding procedure with two *SISO* decoders

The received sequence will be decoded by these two SISO decoders. The principle of iterative decoding in BTCs is basically follows the concept of MAP decoder, which was discussed in the previous chapter,

$$L_U(u_k) = L_c \cdot y_k + L_e^{-}(u_k) + L_e^{|}(u_k), \tag{3.17}$$

the equation explains the SISO decoding procedure. In the beginning, the received sequence affected by AWGN noise, $L_c \cdot y_k$, and the soft values of *a priori* probabilities for information data bits, $L(\vec{U})$, which is initialized by 0, through the first decoder produce the extrinsic values $L_e^{-}(u_k)$ for these information data bits. These extrinsic values of the horizontal decoder are interleaved to be used as a priori information $L(\vec{U})$ and sent to the second decoder with, also interleaved, $L_c * y_k$ to produce the extrinsic values $L_e^{|}(u_k)$, which are de-interleaved and sent to the first decoder to start the second iteration. The procedure will continue from the first iteration to the last iteration and the final result for the estimated information is obtained by hard decision

$$\hat{u}_k = sgn[L_U(u_k)]. \qquad (3.18)$$

3.2.3.1 Trellis Based MAP Decoding Algorithm

As we presented before, this algorithm is optimal but too complex for hardware implementation. It is still meaningful for the researchers to realize how close it is to the theoretical limitation with MAP algorithm applied to BTCs and even for BTCs with high code rate. The BTC decoder is made up of two MAP decoding modules that cooperate in an iterative scheme.

As we showed in Section 2.4.1, the three most important values are $\gamma_i(s', s)$, $\alpha_i(s)$ and $\beta_{i-1}(s')$. We assume that we have a linear systematic (n, k) block code $\vec{C}$, which can be represented by trellis T with the depth of n [34]-[36], [25], [37], [10], [29]. The trellis is a time-indexed version of the state diagram. Each node corresponds to a state at a given time index i, and each branch corresponds to a state transition, which is shown in Figure 3.6. Before we present the MAP decoding

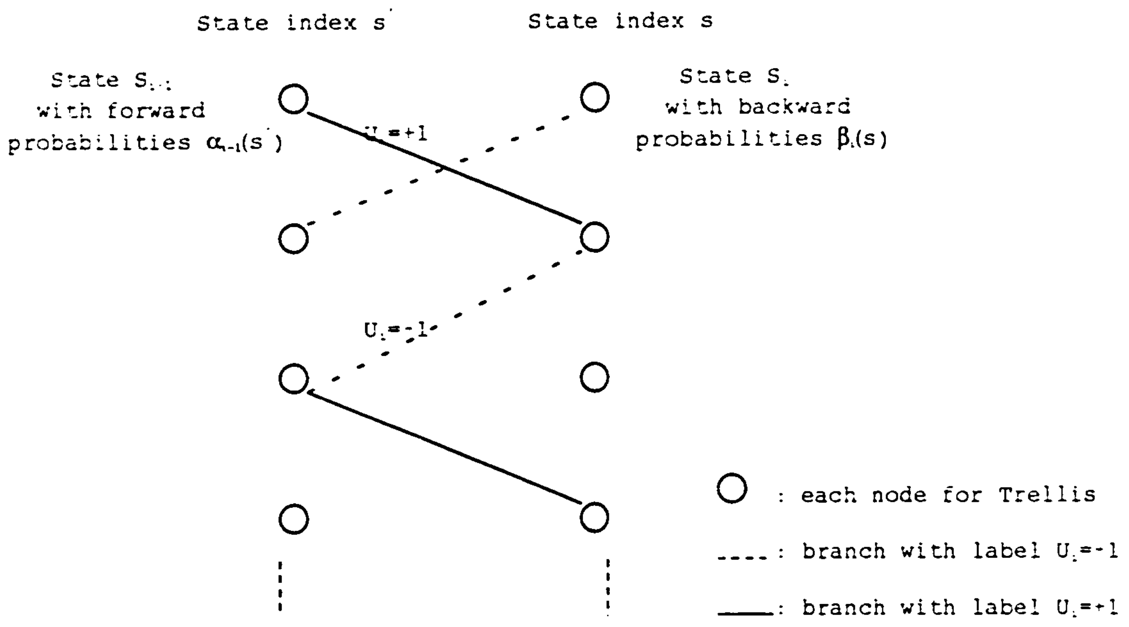

State index s'
State index s
State S_{i-1} with forward probabilities $\alpha_{i-1}(s')$
State S_i with backward probabilities $\beta_i(s)$
$U_i=+1$
$U_i=-1$
: each node for Trellis
----: branch with label $U_i=-1$
____: branch with label $U_i=+1$

Figure 3.6: Simple trellis structure

rule, we list some denotations:

- $\vec{C}$: a linear systematic (n, k) block code.
- $\vec{U}$: the information data bits with length k.
- $\vec{X}$: the modulated code sequence with the values of "+1" or "−1" after encoding for BPSK modulation.
- $\vec{Y}$: the received sequence through the AWGN channel.

Each transition between any two states is labeled with the corresponding codeword symbol X_i, where the first k symbols are equal to the information bit u_k and the following $n-k$ symbols represent parity bits. The branch transition probability in equation 2.27 can be written as the following equation:

$$\gamma_i(s', s) = p(y_i|u_i) \cdot P(u_i) = p(x_i; y_i), \tag{3.19}$$

where $p(x_k; y_k)$ is defined for two parts, the information bits part and parity check bits part as follows

$$p(x_i; y_i) \equiv \begin{cases} p(y_i|x_i) \cdot P(u_i), & 1 \le i \le k \\ p(y_i|x_i), & k+1 \le i \le n. \end{cases} \tag{3.20}$$

From Trellis T, we can easily understand that there are two branches for the first depth of k because first k bits are information bits. However, there is only one branch from the time point $i-1$ to i for the remaining depths of $n-k$. By taking the log-likelihood ratio associated with definition 3.20, we get

$$\begin{aligned} L(x_i; y_i) &= \log\left(\frac{p(x_i=+1; y_i)}{p(x_i=-1; y_i)}\right) \\ &\equiv \begin{cases} L_c \cdot y_i + L(u_i) & 1 \le i \le k \\ L_c \cdot y_i & k+1 \le i \le n. \end{cases} \end{aligned} \tag{3.21}$$

The forward recursion, α, and backward recursion, β, of the "symbol-by-symbol" MAP algorithm are performed using equations 2.28 and 2.29 as given in Section 2.4.1. Then, each MAP decoder for BTCs can be written as

$$L(\widehat{u}) = L_c \cdot y_i + L(u_i) + \log \frac{\sum_{S^{+1}} \alpha_{i-1}(s') \cdot \beta_i(s)}{\sum_{S^{-1}} \alpha_{i-1}(s') \cdot \beta_i(s)}. \tag{3.22}$$

3.2.3.2 MAX-Log-MAP Algorithm with Correction Value

Although MAP algorithm offers very good performance for BTCs, its complexity makes the hardware implementation very difficult. For the purpose of reducing the complexity, we introduce the sub-optimal algorithm: MAX-Log-MAP. It is an approximation of MAP using equation 2.41 applied in MAP decoding algorithm for BTCs and gives the following approximation equation

$$\begin{aligned} L_{MAX-log-MAP}(\widehat{u}_i) \quad = \quad & L_c \cdot y_i + L(u_i) \\ & + max_{S^{+1}}(\log \alpha_{i-1}(s') + \log \beta_i(s)) \\ & - max_{S^{-1}}(\log \alpha_{i-1}(s') + \log \beta_i(s)), \end{aligned} \tag{3.23}$$

where

$$\log \alpha_i(s) = max_{s'}(\log \alpha_{i-1}(s') + \frac{1}{2}L(x_i; y_i) \cdot x_i \tag{3.24}$$

and

$$\log \beta_{i-1}(s') = max_s(\log \beta_i(s) + \frac{1}{2}L(x_i; y_i) \cdot x_i. \tag{3.25}$$

From the simulation results in Figure 2.10, we can see that coding gain loss is around 0.15 dB at BER of 10^{-4} for MAX-Log-MAP algorithm as compared to MAP algorithm. To obtain a better performance and at the same time not add considerable complexity to the hardware implementation, we introduce the MAX*-Log-MAP algorithm as a compromise between the complexity and performance. This algorithm was presented in Section 2.4.3. The max* operation is the basic operation in the Log-MAP algorithm. This operation can be implemented efficiently with a maximum operation and an optional lookup table. We use the dynamic values to replace the correction part: $\log(1 + e^{-|A-B|})$ in equation 2.43, which is shown in the small lookup table 2.2 in Section 2.4.3. The results show that the performance of MAX-Log-MAP algorithm with the correction value of $\delta_{A,B}$ is very similar to the decoding with MAP algorithm without degradation of the bit-error performance. The MAX*-Log-MAP algorithm is a very good decoding scheme providing excellent compromise between the complexity and performance because it does reduce the complexity and avoid costly shift operations [38]. The coding gain lost is almost negligible (only 0.04 dB for the $RM(32,26)^2$, $R = 0.68$, iteration 5 at the BER of above 10^{-4}) as compared to MAP algorithm applied under the same conditions.

3.2.3.3 The Iterative Decoding

The iterative decoding algorithm, proposed by Berrou in 1993, works very well in practice. In Figure 3.7, a half iteration is defined as the decoding of the sequence through one of the decoders of the turbo decoding system, and a full it-

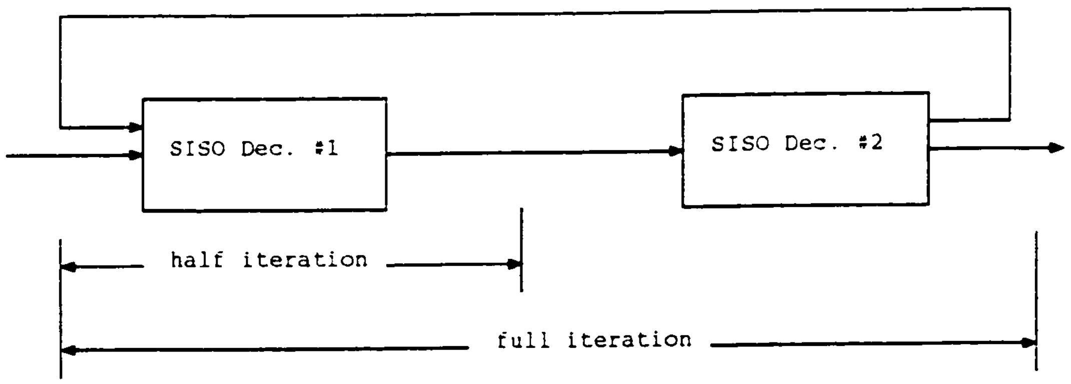


Figure 3.7: Iterative decoding

eration includes decoding by two decoders of turbo decoding system. Each encode n-bit sequence in a BTC is decoded independently. First, all the horizontal blocks are decoded then all the vertical received blocks are decoded or vice versa. The decoding procedure is generally iterated several times to maximize the decoder performance. To achieve the optimal performance, the block by block decoding must be done utilizing soft information, This soft decision decoder must also output decision matrices corresponding to the likelihood that the decoder output is correct. This is required so that the next decoding will have soft input information as well. In this way, each decoding iteration builds on the previous decoding performance. At the beginning the extrinsic values are statistically independent and the coding gain from one iteration to another is high. However, since the same information is used, the improvement after a few iterations becomes very small. After the last iteration, the soft output decision will be the sum of the last two extrinsic values and the received sequence affected by the AWGN noise $L_c * y_k$. The decoder will make a hard decision by comparing the soft output to a threshold equal to zero. In the equation 2.32, if $L(\widehat{u}) \geq 0$, the decoded bit is 1, if $L(\widehat{u}) < 0$, the decoded bit is 0. We can see the impact of iteration to the BTC decoding from the simulation results in Section 3.3.

3.3 Results

3.3.1 Simulation Results for RM BTC Using MAP Algorithm

We give some simulation results in Figures 3.8 , 3.9, 2.10 on the performance in terms of BER versus E_b/N_0 for RM block turbo codes with MAP and MAX*-Log-MAP algorithm for different code rates: $RM(8, 4)^2$, $RM(16, 11)^2$, $RM(32, 26)^2$. The results show that RM block turbo codes with high code rate performs better or even much better than the ones with low code rates, especially at higher E_b/N_0. This is due to their longer block length. Also we investigate the RM block turbo codes with different modulations such as BPSK and QPSK and in AWGN channel, the simulation results show that the performance is almost same for these two modulations.

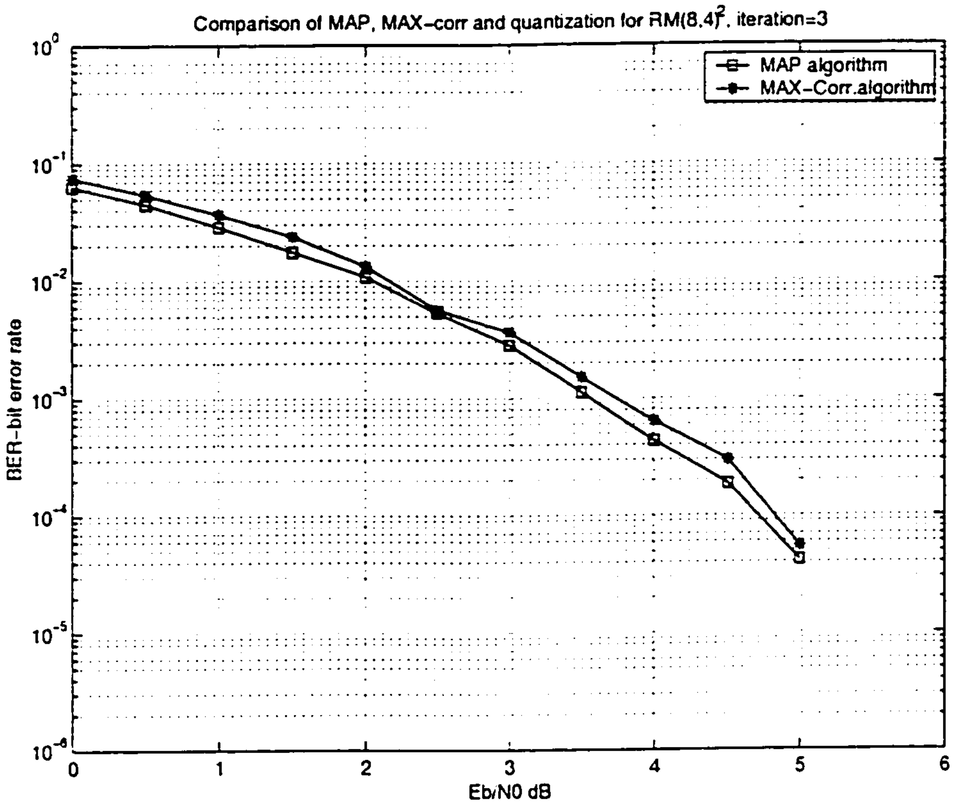


Figure 3.8: Comparison of various decoding algorithms for $RM(8, 4)^2$, R= 0.33, iteration 3

3.3.2 The Impact of Iteration

Figure 3.10 (Page 52) illustrates that at the first iteration, the performance is improved with increasing the number of iterations. The gain from the first iteration to the second one is high as shown in these figures. However, after a few iterations, the gap of performance becomes very narrow. That is because in the beginning the extrinsic values are statistically independent and become more and more correlated. So after a few iterations, the decoders practically use the same information in different iterations. since the decoding always uses the same information sequence. Moreover, we can see that the number of iterations for high code rates is more than the low rate ones. For a low code rate such as for $RM(8, 4)^2$, the coding gain would be saturated after 3 iterations, whereas for $RM(32, 26)^2$ the iteration number is 5,

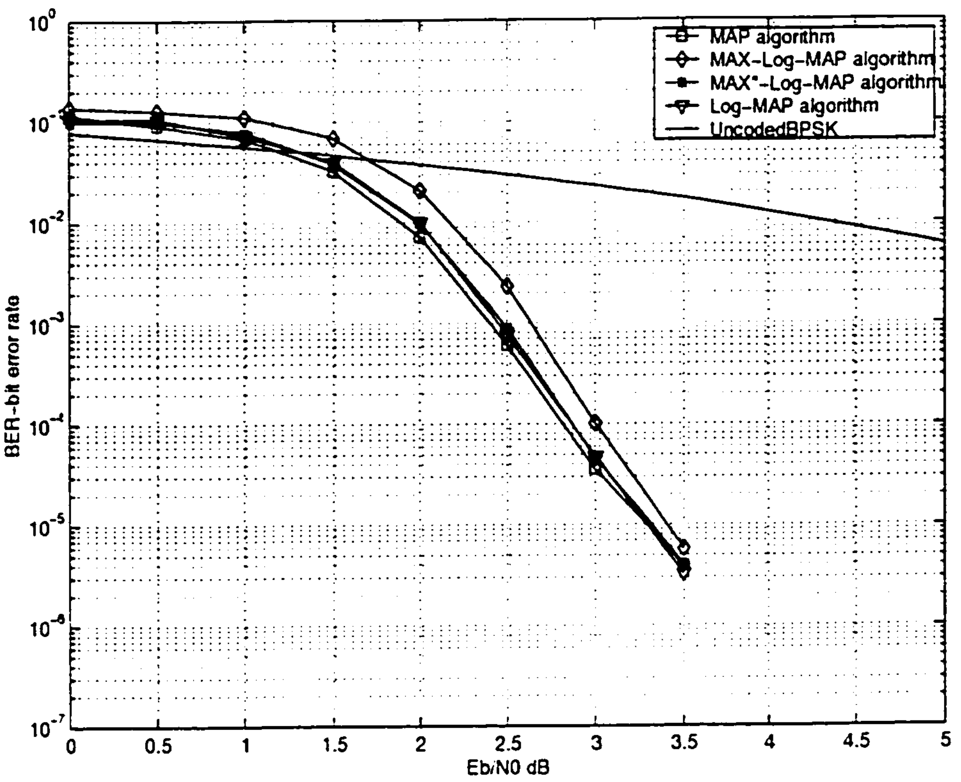


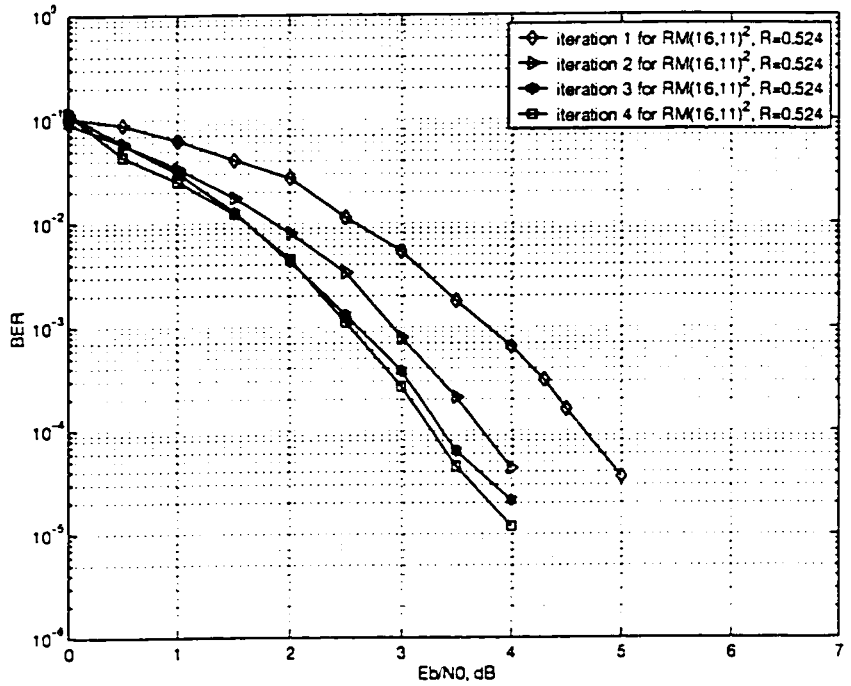
Figure 3.9: Comparison of various decoding algorithms for $RM(16, 11)^2$, R= 0.52, iteration 4

which is because different code lengths causes the different sizes of interleaver.

3.4 Conclusion

This chapter discussed 2-dimensional RM BTC. We discussed the RM block code construction. The RM block turbo code encoding system model employs the same principle as the conventional turbo code encoding system model, except that the two convolutional encoders are replaced by block encoders. Then we presented the optimal decoding algorithm: MAP, based on the trellis. The decoding process is performed block by block employing a forward and a backward recursion. To reduce the complexity of the decoding, we introduced the minimal trellis and described the Massey trellis used in this thesis. We also described the decoding algorithm:

MAX*-Log-MAP, which is an approximation of the MAP algorithm with a correction term to compensate the difference. The MAX*-Log-MAP, while a suboptimal algorithm, does provide a good trade-off between the hardware and performance. The simulation results showed that this suboptimal algorithm meets the practical implementation and performance goals.



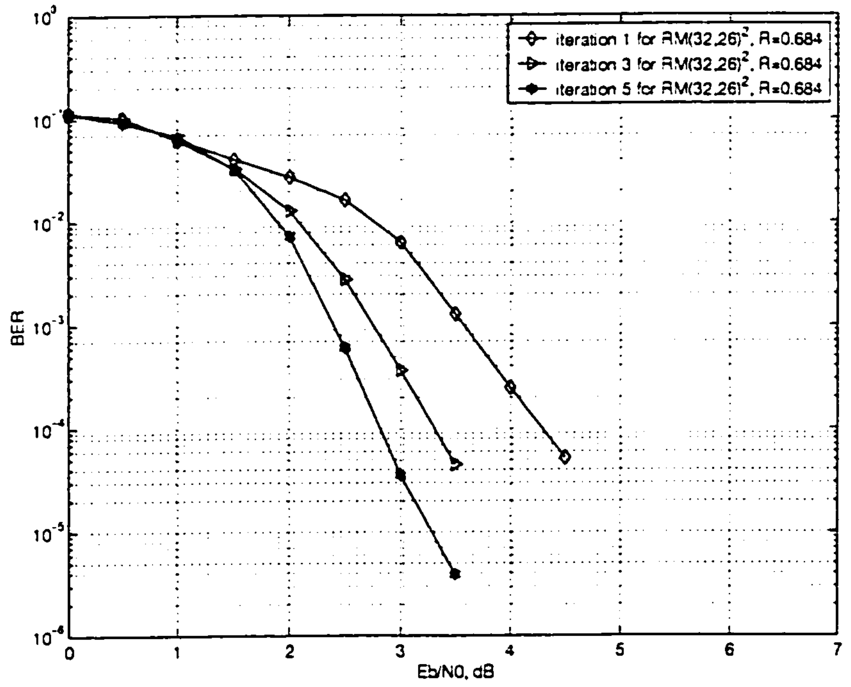


Figure 3.10: The effect of iteration to $RM(16, 11)^2$, $RM(32, 26)^2$

Chapter 4

RM Block Turbo Codes (3-dimensional) Implementation Design and TPC

Block turbo codes are composed of simple block codes applied not only in two (orthogonal) dimensions, but also in multiple dimensions. To obtain better performance for block turbo codes and to ensure that, the complexity does not grow exponentially, which is easier for hardware implementation, we extended the 2-dimensional block turbo code to 3-dimensional case. Three dimensional block turbo codes can provide flexibility in choice of code rates and coding gain advantages. In this chapter, we will investigate 3-dimensional RM block turbo codes with MAP decoding algorithm. Then, we will describe some of the implementation issues: how to quantize the received sequence and how to choose the best bit-width for quantization through simulation results.

4.1 Three-dimensional RM Block Turbo Codes

Multi-dimensional BTCs are higher dimensional codes where each bit is also encoded by three or more encoders. Therefore, third or more codes are produced. With multi-dimensional block turbo codes, we can get a large number of block sizes and information sizes. In this way, we also can get a large range of code rates. This flexibility allows BTCs to be used in a wide range of applications. The following table shows the example codes for 3-dimensional block codes. From this table, we

Table 4.1: Example of RM block turbo codes in 3-dimension as used in this thesis

	Block size	Information size	R (Code rate)
$RM(8, 4)^3$	256	64	0.25
$RM(16, 11)^3$	3146	1331	0.423
$RM(32, 26)^3$	29744	17576	0.59
$RM(64, 57)^3$	253422	185193	0.73
$RM(16, 11)^2(8, 4)$	1408	484	0.344
$RM(16, 11)^2(32, 26)$	6732	3146	0.467

can see that 3-dimensional block turbo codes have a greater possibility to get various code rates than convolutional turbo codes and even 2-dimensional block turbo codes and, thus, meet the practical applications.

4.1.1 Construction of 3-dimensional RM Turbo Codes

For a three-dimensional block turbo code, the element ordering for input/output of both encoding and decoding is usually in the following order: X-axis (rows), Y-axis (column) and then the Z-axis. For a three-dimensional information block of size $(i * j * k)$ and total block size $((i * j * k)$+parity check bits), we can further understand that a product code is a relatively large code just built from smaller codes. For example, let us consider $RM(8, 4)$, $R = 0.5$. This code has 4 information bits, computes another 4 parity checking bits via the encoding system, and then transmits the 4 parity check bits together with the 4 information bits. In Chapter 3, we investigated the two-dimensional RM block turbo codes, which result in a (48, 16) code, $R = 0.33$. In this chapter, we consider 3-dimensional RM turbo block codes, which would result in a (256, 64) code with code rate $R = 0.25$. Consequently, we can see that by introducing the multi-dimensional block turbo code, we get a lower code rate but a larger block size code.In addition, performance is getting better, which will be shown from the future simulation results.

The three-dimensional block turbo code structure can be seen in the following

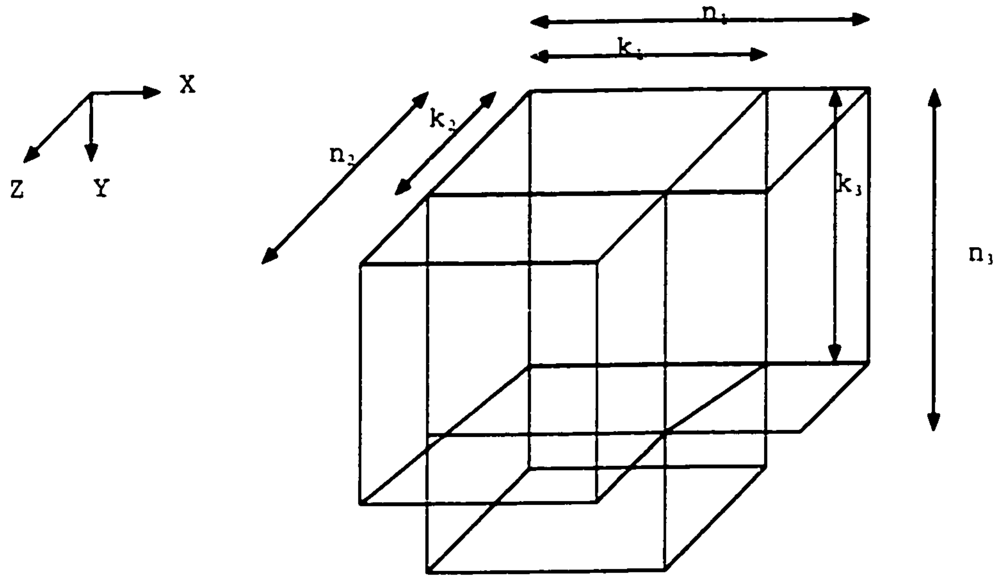


Figure 4.1: The (n, k) 3-Dimensional block turbo code constructured from $(n_1, k_1) * (n_2, k_2) * (n_3, k_3)$ RM Codes

Figure 4.1. Each code symbol is a cube, or a 3-D vector. This model can be extended to an arbitrary of dimensions. In this 3-dimensional block code, we do not think about "parity on parity" bits so the block length, n, and information length, k, are given for the new code (n, k), respectively as

$$n = [n_1 * n_2 - (n_1 - k_1) * (n_2 - k_2)] * k_3 + k_1 * k_2 * (n_3 - k_3) \tag{4.1}$$

$$k = k_1 * k_2 * k_3. \tag{4.2}$$

Therefore, the code rate for this new code is

$$R = \frac{k}{n} = \frac{k_1 * k_2 * k_3}{[n_1 * n_2 - (n_1 - k_1) * (n_2 - k_2)] * k_3 + k_1 * k_2 * (n_3 - k_3)} \tag{4.3}$$

4.1.2 Three-Dimensional RM BTCs Encoders

Two-dimensional RM BTCs encoding systems are based on two parallel-concatenated block codes; the interleaver is simply row-wise and column-wise. Three-dimensional BTCs encode the same information bits three times. There is a similar encoding system mode for 3-dimensional RM BTCs except there are three elementary encoders for each dimension and three interleavers . The encoding system model is shown in Figure 4.2. There are three block interleavers: X-axis, Y-axis and Z-axis

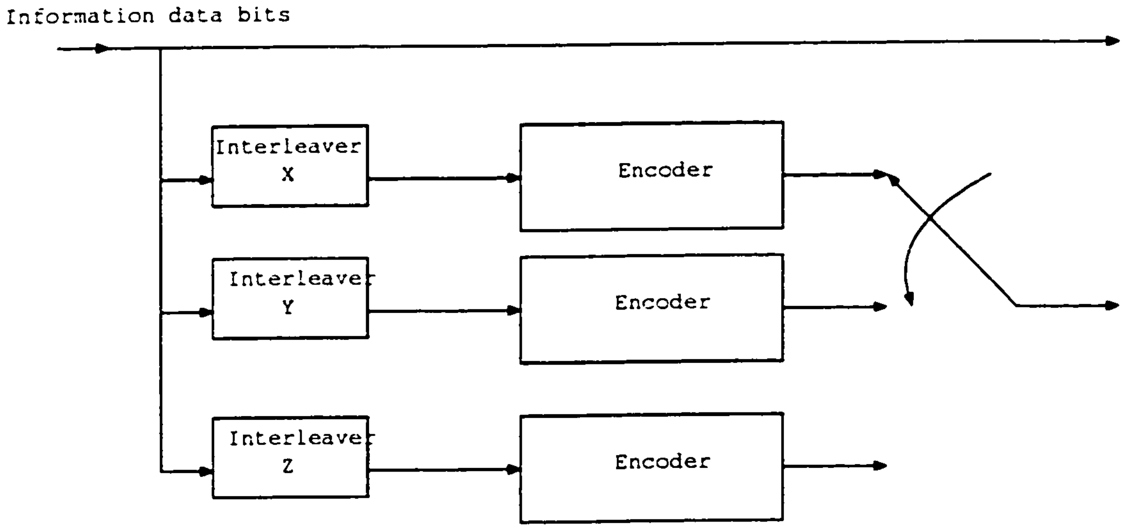


Figure 4.2: 3-dimensional RM block turbo codes encoding model

in this model. Those block interleavers are used to separate each dimension so that the information bits can be effectively exchanged in the iterative decoding process. The output codes are combined by four parts: information matrix of $k_1 * k_2 * k_3$, the coded bits P_x for X-axis (the first RM block code encoder), P_y for Y-axis (the second RM block code encoder), and P_z for Z-axis (the third RM block code encoder) by generator matrix and information data bits. The information data bits are sent to the second and the third encoder after block interleaver. The four parallel streams of data are then multiplexed into one stream and sent to the channel. We notice that the interleaver for block turbo codes is very simple and each elementary encoder is similar and not too complicated for hardware implementation. For these three elementary encoders, we can use different RM block codes on each dimension. For example, in the code of $RM(8, 4)(16, 11)(32, 26)$, the first encoder encodes 4 information bits to code length of 8 for X-axis, the second one computes 5 parity check bits to create a $16-bit$ code word for Y-axis, and so on. Different code combinations for different dimensions will easily create a large range of codes with different code rates and different block sizes, which can avoid the use of puncturing technique to increase the code rate but result in a worse performance if compared with turbo codes of the same rate. Later, when we talk about multi-dimensional block turbo codes decoding, we will see that multi-dimensional block turbo codes provide excellent performance while the complexity of encoding and decoding increases only linearly with the dimension.

The output codes are modulated as modulation sequences by BPSK or QPSK and then sent out through AWGN channel. The process is identical to that of 2-dimensional block turbo codes. These modulation sequence with affected noise component value are transmitted to the receiver side waiting to be decoded.

4.1.3 Decoding Scheme for 3-dimensional Block Turbo Codes

The received sequences of 3-dimensional block turbo codes can be decoded by several algorithms as in the case for 2-dimensional block codes and convolutional turbo codes: BCJR algorithm, SOVA algorithm, chase algorithm, and so on. In this thesis, we used the suboptimal decoding algorithm MAX*-Log-MAP algorithm. This decoding scheme, was presented in Chapter 2 and Chapter 3, and we now apply it to 3-dimensional block turbo codes for the reason that it is a good compromise between performance and hardware cost.

We still use the suboptimal algorithm based on the minimal trellis, which is no more than 2^{n-k} states in the trellis depth to reduce the decoding complexity. The trellis starts and ends in the same zero state. This characteristic allows the use of the MAP or MAX*-Log-MAP decoding algorithm without the need of appending any tail to the information sequence in order to drive the encoder to the zero state like convolutional turbo codes do. To use the suboptimal algorithm for 3-dimensional block codes, we have to present it beginning with MAP algorithm. In Chapter 3 (2-dimensional block turbo decoding), we showed how the iterative decoding with MAP algorithm works. It is a serial structure for a block turbo decoder [10]. Each decoder uses the *a priori* value that actually is the *extrinsic* value from the other decoder after de-interleaving and at initial state, *a priori* value is 0. As illustrated in Figure 4.3, the input arrows to each block represent the *a priori* information. The output arrows represent the extrinsic information generated by the particular block. Each estimated information bit is decided by the soft value of equation 2.25, which is for 2-dimensional BTCs:

$$L_U(u_k) = L_c \cdot y_k + L_e^-(u_k) + L_e^|(u_k).$$

For the lower block size and code rate, we use BCJR algorithm, based on the trellis diagram MAP decoding algorithm because it is more precise for decoding. Then, equation 2.25 can be further expanded in to equation 3.22 as follows:

$$L(\widehat{u}) = L_c \cdot y_i + L(u_i) + \log \frac{\sum_{S^{+1}} \alpha_{i-1}(s') \cdot \beta_i(s)}{\sum_{S^{-1}} \alpha_{i-1}(s') \cdot \beta_i(s)}.$$

For the 3-dimensional block turbo codes decoding, we use Figure 4.3 to illustrate the decoding process. The three-dimensional RM block turbo decoder depicted in Figure 4.3, consists of three elementary decoders (Decoder in $X - axis$, Decoder in $Y - axis$, and Decoder in $Z - axis$) in series. We have presented how the information from this iteration to next iteration is passed. The process also works for 3-dimensional BTCs. Three decoders in this diagram are $soft - in/soft - out$ decoders. For each decoder, there are two input values: *a priori* value, received

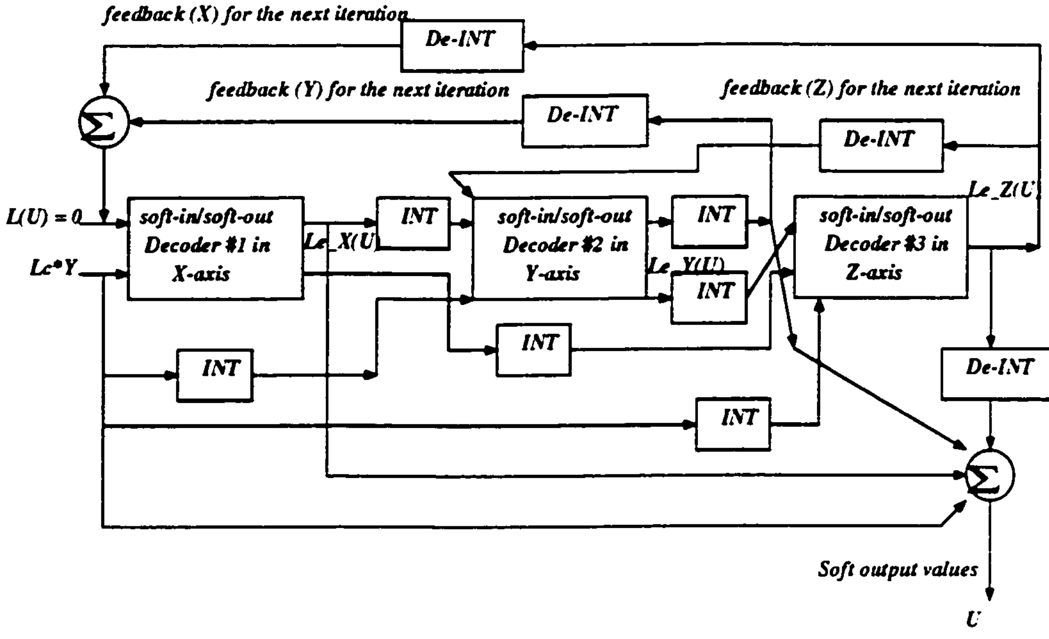

feedback (X) for the next iteration
De-INT
feedback (Y) for the next iteration
De-INT
feedback (Z) for the next iteration
De-INT
L(U) = 0
Lc*Y
soft-in/soft-out Decoder #1 in X-axis
Le_X(U)
INT
soft-in/soft-out Decoder #2 in Y-axis
Le_Y(U)
INT
INT
soft-in/soft-out Decoder #3 in Z-axis
Le_Z(U)
INT
INT
INT
De-INT
Soft output values
U

Figure 4.3: Iterative decoding process for 3-dimensional block turbo code

sequences in the beginning, then after decoding, the extrinsic value is generated to replace *a priori* value. The extrinsic value is just a soft output value. This value will be passed to the next two decoders for decoding. Each decoder uses the *a priori* value that is actually the extrinsic values from the other two decoders. At the receiver end, the decoding algorithm is as follows:

- the $X - axis$ decoder produces the extrinsic information for the received sequence
- the received sequence is interleaved and input to the $Y - axis$ decoder together with the interleaved extrinsic information summation from $X - axis$, and $Z - axis$ which is used as the *a priori* value to the $Y - axis$ decoder.
- similarly with the $Y - axis$ decoder, the interleaved received sequence with the interleaved summation of extrinsic value used as a *priori* value input to the $Z - axis$ decoder.

- the $Z - axis$ decoder produces the new extrinsic information which will be added to the new extrinsic value produced by $Y - axis$, deinterleaved and used as *a priori* input in the $X - axis$ decoder in the next iteration, and so on.

The soft-outputs produced by the decoder for $X - axis$ is given by

$$L_U^x(u_k) = L_c \cdot y_k + L_e^y(u_k) + L_e^z(u_k), \tag{4.4}$$

where $L_c \cdot y_k$ is the received sequence affected by AWGN channel noise, $L_e^x(u_k)$, $L_e^y(u_k)$, $L_e^z(u_k)$ are respectively the extrinsic values from corresponding decoder, . Similar equations can be written for $L_U^y(u_k)$ and $L_U^z(u_k)$

$$L_U^y(u_k) = L_c \cdot y_k + L_e^x(u_k) + L_e^z(u_k), \tag{4.5}$$

$$L_U^z(u_k) = L_c \cdot y_k + L_e^x(u_k) + L_e^y(u_k). \tag{4.6}$$

At the final iteration, the final soft value will be obtained by the sum of channel values $L_c \cdot y_k$ and the three extrinsic values of the three decoders in different axes. Three-dimensional turbo codes require 3 sub-iterations per iteration and, therefore, require twice as much extrinsic information as two-dimensional BTCs.

$$L_U(u_k) = L_c \cdot y_k + L_e^x(u_k) + L_e^y(u_k) + L_e^z(u_k). \tag{4.7}$$

Note that the *a priori* information is provided only for the information bits in the block part. The decoding of the parity bits does not use any *a priori* information. In addition, the rule for making decisions regarding the information bits is exactly the same as the one in 2-dimensional BTCs, which we presented in Chapter 2. Similarly, with the increasing of the iteration number, the BER performance will be improved.

From the above equations and combined with the approximation equation, which is described in Chapter 2, we obtain the suboptimal decoding scheme for 3-dimensional BTCs. This decoding scheme does not lose much coding gain and is much easier for hardware design as compared to MAP algorithm. We will see how much coding gain loss with the suboptimal algorithm comparing to the optimum MAP algorithm.

4.2 Quantization for Decoders of BTCs

Software simulation for BTCs in any applied algorithms, such as MAX*-Log-MAP algorithm used in a BTC-decoder, are usually specified in the floating-point domain. It is necessary for an actual hardware implementation to transform from floating to the fixed-point because fixed-point number representation is mandatory for most target architectures [38]. The basic goal for a software simulation for implementation is to find a fixed-point model that corresponds to the given bit-width of the decoder chips. Quantization of the received sequence is very important because a large word length not only takes a lot of hardware for the buffers but also causes a large amount of hardware for the computation [39]. The simulation for software implementation with fixed-point number offers a suitable bit-width as small as possible and, at the same time, the performance degradation is of an acceptable value. We investigate the dynamic range of received sequence and the influence of quantization and fixed-point arithmetic for the implementation of the turbo decoder.

4.2.1 The System Model

A block diagram of the communication system considered in quantization is illustrated in Figure 4.4. The coded bits are BPSK modulated into $(-1, +1)$ signal and transmitted through the channel. AWGN noise $n_k \in \mathcal{N}(0, \sigma^2)$ is added to the transmitted signal and the received bits are scaled by a factor g before they are digitized by the quantizer. The gain control adjusts the amplitude of the signal to minimize the distortion with the optimal scaling factor or to achieve the best signal-to-distortion ratio at the quantizer.

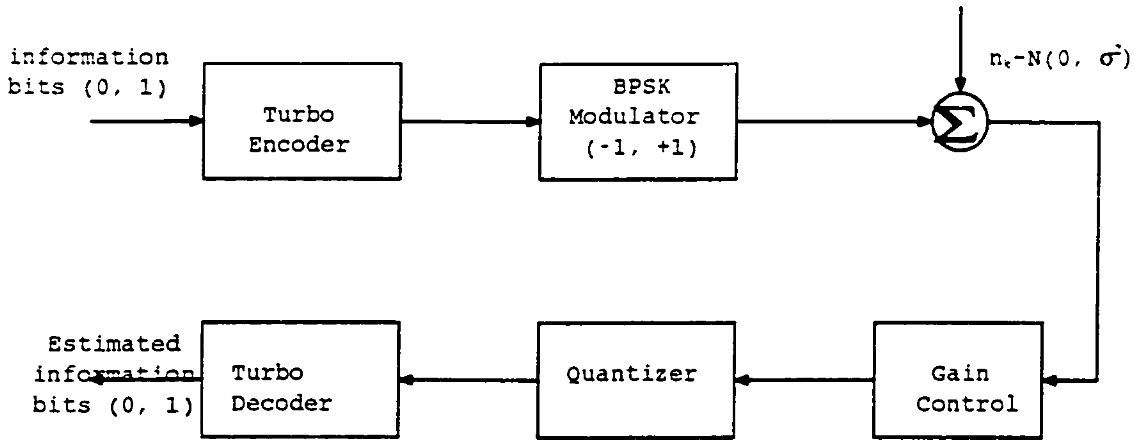


Figure 4.4: System model including quantization [40]

4.2.2 Channel Input Quantization

In [41], an optimized quantization scheme of the internal values does not degrade the bit-error performance, so in this thesis, we only investigate the quantization for the input received sequence and the effect of the quantization on the performance. Because the system model we have studied is only AWGN channel, the input received sequence is distributed with a Gaussian distribution around the transmitted symbols $(-1, +1)$. Assuming we have a data stream with a mean equal to 0 and a variance equal to σ,more than 99% of the data sequences are covered in the range of $(-3\sigma, 3\sigma)$.

$$\begin{aligned} \int_{-3\sigma}^{3\sigma} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(x-0)}{2\sigma^2}\right] &= G(\tfrac{3\sigma-0}{\sigma}) - G(\tfrac{-3\sigma-0}{\sigma}) \\ &= G(\tfrac{3\sigma}{\sigma}) - G(\tfrac{-3\sigma}{\sigma}) \\ &= G(3) - [1 - G(3\sigma)] \\ &= 0.997 \end{aligned} \qquad (4.8)$$

Also more than 99% of the received sequences are covered by limiting the dynamic-range of the received channel values $(-1 - 3\sigma, +1 + 3\sigma)$, where σ is the variance of the AWGN distribution because the transmitted sequence is either -1 or $+1$ as shown in Figure 4.5. There are several quantizations: uniform, non-uniform and logarithmic quantizers [42]. We use uniform quantizer to quantize the input received

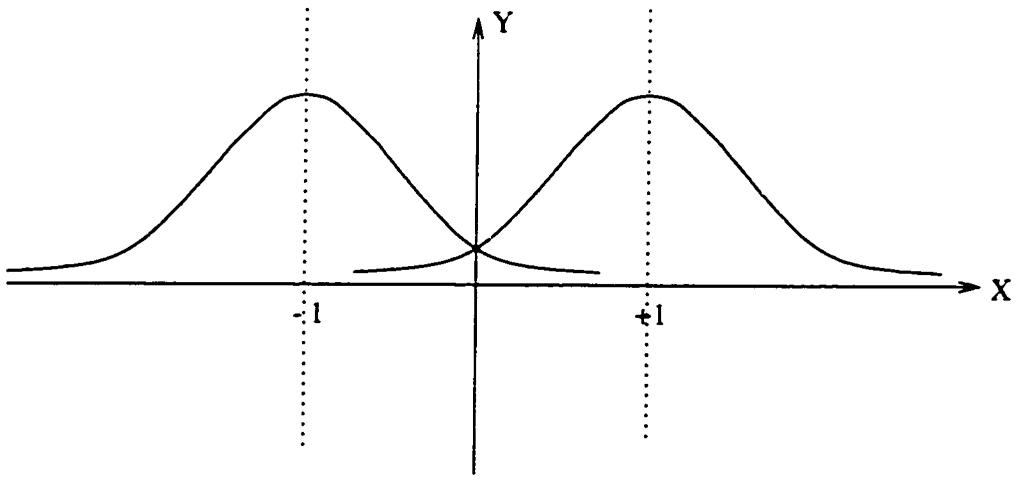


Figure 4.5: The distribution of received sequence in AWGN channel

sequence. There is a simplified method that considers that the received sequences are in the range of (-4, 4) because most received sequences are limited to this area as shown in the following table. For the reason of precision, in this thesis, uniform

Table 4.2: The received sequence corresponding to the various $\frac{E_b}{N_0}$ for $RM\ (32,\ 26)^2$

$\frac{E_b}{N_0}$	$0\,dB$	$0.5\,dB$	$1\,dB$	$2\,dB$	$3\,dB$	$4\,dB$	$4.5\,dB$	$5.0\,dB$
σ	0.855	0.807	0.762	0.68	0.605	0.54	0.51	0.481
$1+3\sigma$	3.565	3.421	3.286	3.04	2.815	2.62	2.53	2.443

quantizer deals with the range of received sequence at $(-1-3\sigma,\ +1+3\sigma)$. We suppose $D = 1 + 3\sigma$ and that all received sequences in the range of $(-D,\ D)$ are divided by 2^m evenly spaced bins, where m is the number of bits of the quantizer [43]. The quantization step size d is

$$d = 2D/2^m, \tag{4.9}$$

each step size boundaries are at

$$\left\{(-\infty,\ -\frac{7}{8}D);\ (-\frac{7}{8}D,\ (-\frac{7}{8}D+d));\cdots(-\frac{1}{8}D,\ 0);\ (0,\ \frac{1}{8}D)\cdots(\frac{7}{8}D,\ \infty)\right\} \tag{4.10}$$

and each received sample is mapped into the center of the corresponding bin. For example, when $\frac{E_b}{N_0}$is 0 dB, then D is 3.565 and a sample with the value 0.246 is mapped into $9th$ step. The sample falls into the bin boundary $(0, \frac{1}{8}D)$, it will be mapped into the

$$(\frac{1}{8}D - 0)/2 = 0.223, \tag{4.11}$$

and then it will be represented as a binary number.

4.2.3 Simulation Results for Quantization

Figure 4.6 shows the performance with MAX*-LOG-MAP decoding algorithm for three-dimensional $RM(32, 26)$ at iteration 5 when the received signal is quantized. At the BER $= 10^{-5}$, we find 0.3 dB coding degradations for an AWGN channel under the condition of 4-bit quantization.

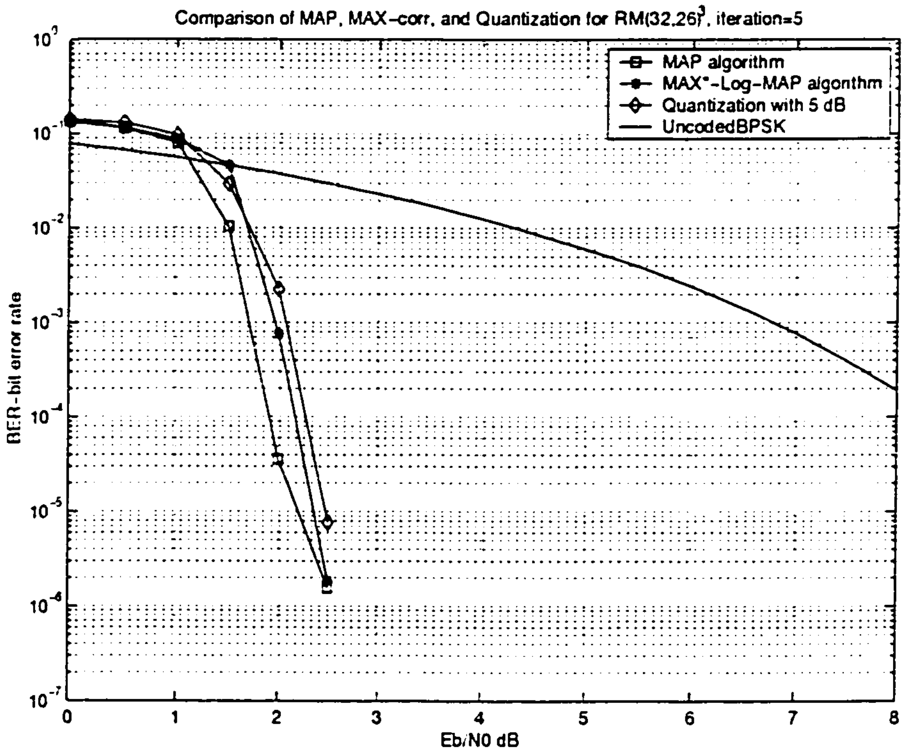


Figure 4.6: The effect of quantization received sequence for 3-dimensional RM(32,26)

The amount of bits needed for quantization is also a problem for the effect of performance and also for reducing the complexity of hardware design. Figure 4.7

shows the performance of the code for different numbers of quantization bits. From

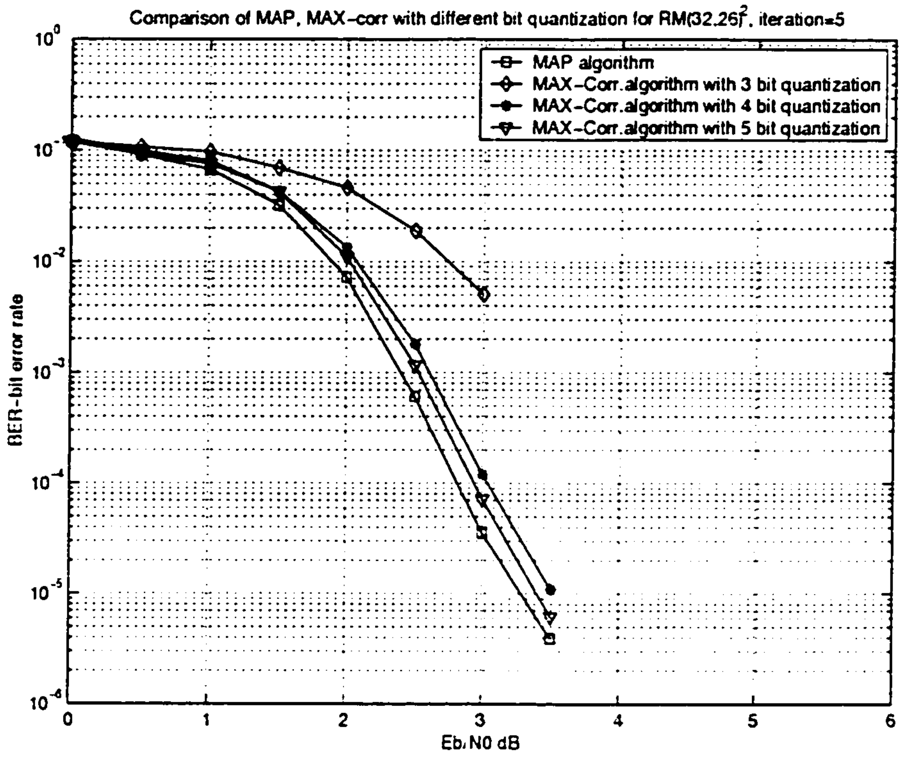


Figure 4.7: Simulation for different bit quantization schemes for two-dimensional block turbo code

the simulation results of Figure 4.7, which shows that the performance degradation from the infinite precision is negligible if 4-bit quantization is used for received sequence, we can see that 4-bit quantization is the best choice because the performance for 4-bit quantization is much better than that of 3-bit quantization. It can be seen that 4-bit quantization provides a good tradeoff between hardware complexity and BER performance. Hence, we choose 4-bit quantization and σ is fixed for $\frac{E_b}{N_0} = 5dB$.

4.3 Comparison the BTCs in 3D and 2D

To compare the three-dimensional and two-dimensional BTCs in terms of BER performance, we have to use puncturing skill to get the same code rate for these two different dimensional BTCs. As we have presented in Chapter 2, puncturing is the technique used to obtain any higher code rate by deleting the parity check bits periodically or regularly from a encoder output without changing any of its basic attributes. Since all information bits should be retained to obtain good results from iterative decoding, only the parity bits are punctured.

4.3.1 Puncturing Rule

We compare 2D and 3D BTCs in terms of the BER performance under the same conditions such as all the code words with BPSK are modulated and affected by an AWGN channel noise before reaching the received end, and optimum MAP algorithm is used. We choose the component code $RM(32, 26)$ and $RM(8, 4)$ as our experimental sample. For the component code $RM(32, 26)$, there are different code rates for 2-dimensional $RM(32, 26)$, $R_1 = 0.68$ and 3-dimensional $RM(32, 26)$, $R_2 = 0.59$. This is why we have to use the puncturing method to increase the BTC $RM(32, 26)^3$ and to get the new code rate $R_2' = 0.68$. We list some information in the below table for these two component codes with puncturing and non-puncturing.

Table 4.3: Comparing the basic information between puncturing and non-puncturing for $RM(32, 26)^3$ and $RM(8, 4)^3$

	No puncturing		Puncturing	
	Block Size	R(Code Rate)	Block Size	R(Code Rate)
$RM(32, 26)^3$	29744	0.59	25688	0.68
$RM(8, 4)^3$	256	0.25	192	0.33

The criteria to select puncturing pattern should also be based on the theory of output weight distribution [44]. There are several ways to delete these redundancy parity check bits: $29744 - 25688 = 4056$ for the $RM(32, 26)^3$. One is we can leave with (29, 26) component codes in 2 of the dimensions, then we can delete:

$$(32 - 29) * 26 * 26 * 2 = 2028 * 2 = 4056. \tag{4.12}$$

We use this way to delete the 4056 parity check bits from 2 dimensions: x-axis and y-axis, for each one we delete 2028 parity check bits, then we obtain $R =$

0.68 for $RM(32, 26)^3$. We can also increase the code rate from 0.59 to 0.68 for $RM(32, 26)^3$ by doing another puncturing: keep the 13 information columns and 13 information rows as the complete encoded code bits (32,26), the remaining 13 information columns and 13 information rows only keep information bits, all parity check bits for the rest columns and rows will be deleted. By that we get the code after puncturing with block size 25688 and the code rate is $R = 0.68$ for $RM(32, 26)^3$.

We could use the puncturing technique for $RM(8, 4)^3$, which is the same as the one used for $RM(32, 26)^3$. We keep (6, 4) component codes in any 2 of the dimensions by deleting the 2 parity check bits in each code word instead of sending 4 parity check bits in each code words. In total, we delete 64 parity check bits in the two dimensions, then we obtain code rate $R = 0.333$, which is same as that of $RM(8, 4)^2$. The simulations to compare 2D and 3D BTCs will be described in the next section.

4.3.2 Simulation Results

Fig. 4.8 illustrates the performance on BER of $RM(8, 4)^2$, $RM(8, 4)^3$ and $RM(8, 4)^3$ after puncturing, the three BTC codes BPSK modulated and, through an AWGN channel, decoded by MAP algorithm with 3 iterations. The code rates are: 0.33, 0.25, 0.33, respectively.We can see that the BER performance of $RM(8, 4)^3$ with code rate $R = 0.25$ is better than $RM(8, 4)^2$ with code rate $R = 0.333$. However, it is different story for the puncturing $RM(8, 4)^3$ with the same code rate as $RM(8, 4)^2$. At low $\frac{E_b}{N_0}$, such puncturing 3D code performs slightly better than the full 2D code (in this case of $RM(8, 4)$, until $\frac{E_b}{N_0}$ =4.5 dB), after 4.5 dB, the puncturing code performs worse than the 2D code.

For the case of component $RM(32, 26)$, until $\frac{E_b}{N_0}$ =2.8 dB, the puncturing one performs slightly worse than the $RM(32, 26)^2$ though it performs better than 2D before 2.5 dB.

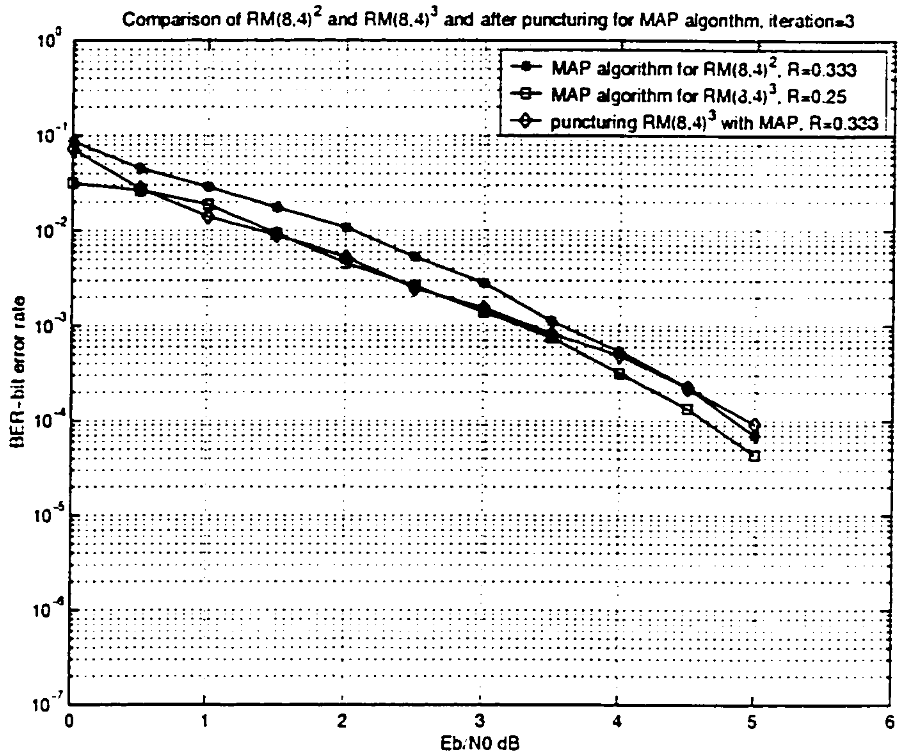


Figure 4.8: Comparing the 3D and 2D on BER performance, example 1 of $RM((8,4)^2$

4.4 Turbo Product Code

4.4.1 TPCs Code Construction

TPCs are a class of block turbo codes. The turbo product code structure is similar to that of block turbo codes. Both of them encode twice or more times the same information bits. The difference for turbo product code is that the parity bits are also encoded. This part is called parity check on check bits. However, this part does not exist in block turbo codes. For the TPC construction, we can refer to Fig. 4.9.The turbo product code $C = C_1 * C_2$ is obtained by:

1. placing $(k_1 \times k_2)$ information bits in a matrix of k_1 rows and k_2 columns,

2. coding the k_1 rows using code C_1,

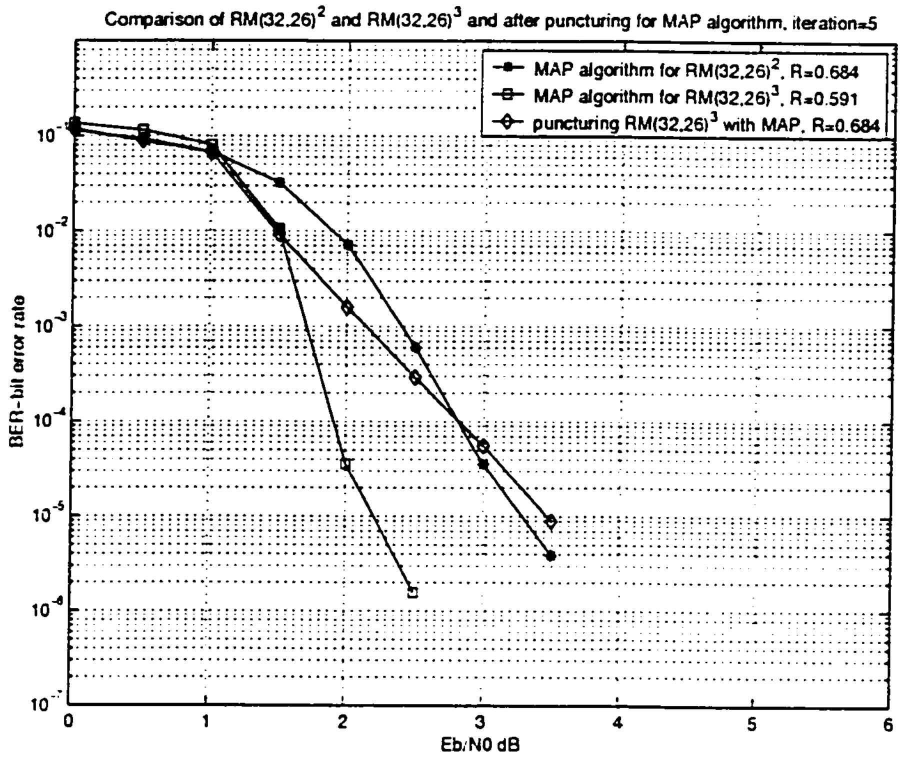


Figure 4.9: Comparing the 3D and 2D on BER performance, example 2 of $RM(32, 26)^2$

3. coding the n_2 columns using code C_2.

The matrix of parity on parity check bits can also be obtained by coding n_1 rows. This gives the same result as the result of the listed processing. Block codes C_1 and C_2 have parameters $(n_1,\ k_1,\ d_1)$ and $(n_2,\ k_2,\ d_2)$, where d_i $(i = 1,\ 2)$ is the code word weight, respectively.

From this structure, we can get the parameters $(n,\ k,\ d)$for the turbo product code C, $(n,\ k,\ d)$.

$$n = n_1 \times n_2, \qquad k = k_1 \times k_2 \qquad and \qquad d = d_1 \times d_2. \tag{4.13}$$

and the code rate is $R = R_1 \times R_2$, where $R_i(i = 1,\ 2)$ is the code rate $C_i(i = 1,\ 2)$. Because of the parity on parity check bits, the code rate for BTC is smaller than that of TPC when the same component codes are used.

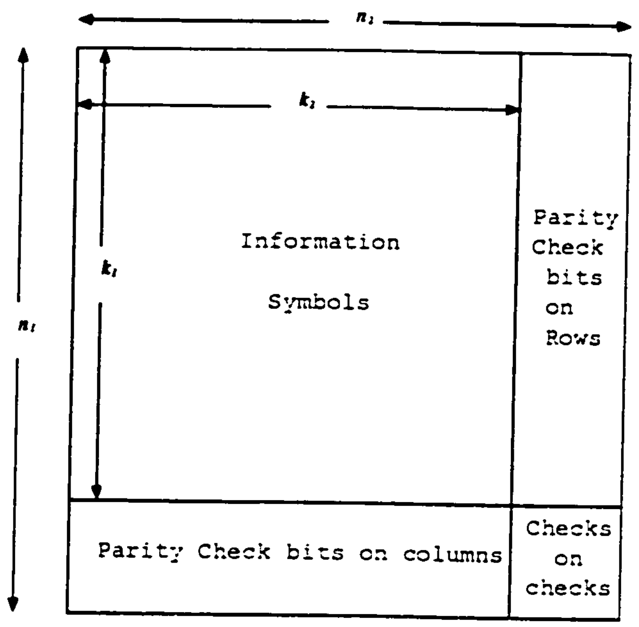


Figure 4.10: Construction of Turbo Product Code C for 2-dimensional

Table 4.4: Comparing the code rate between TPC and BTC with same component code used

	R (code rate for BTC)	R(code rate for TPC)
$RM(8,4)^2$	0.33	0.25
$RM(16,11)^2$	0.52	0.473
$RM(32,26)^2$	0.68	0.66
$RM(64,57)^2$	0.803	0.793

4.4.2 TPC Coding Scheme

The TPC encoding model is the same as that of BTC. For more details, please refer to Section 3.2.2.

There are several decoding schemes investigated for TPC. Reddy [45] used iterated method to decode product code, however, the performance is not so good as it is based on hard decoding which is sub-optimal on an AWGN channel. In 1992, in [46], the decoding algorithm was proposed using a separable MAP filter, which allows soft decisions to be passed from one iteration to the next. A one-dimensional MAP filter is used sequentially in each dimension. The probabilities of error obtained at the first step are further refined by another MAP filter used for the next dimension, which completes a single filtering cycle. After the first cycle, the resulting word may not be a valid codeword, but , by iterating the decoding operation a small number of times, the MAP algorithm is able to decode valid codewords.

Following the famous results obtained from Turbo Code in 1993, research started to focus on the iterative algorithm which yields near optimum decoding of products in terms of BER based on soft decoding and soft decision output. In 1994, Pyndiah [5], [6] and some other people [8] modified chase algorithm to achieve excellent BER performance comparable to the those of CTCs especial for high code rate. Although modified chase algorithm can achieve a very good compromise in terms of BER performance and hardware cost for TPCs, it is still a suboptimal algorithm. We use BCJR and also suboptimal MAX*-Log-MAP algorithm based on minimal trellis in this thesis. The BER performance is better than the modified chase algorithm. The decoding procedure for TPCs is similar to the decoding of BTCs. Decoding can be performed either row-wise first and then column-wise or vice versa. For iterative decoding, the decoding process at each step should be SISO. For more details regarding iterative decoding, please refer to Section 3.2.3 and Section 4.1.3. During the decoding processing, the *a priori* information is provided only for the information bits in the block part, which is the same for BTCs. The decoding of the parity bits and parity on parity check bits do not use any *a priori* information. The rule for making decision of the information bits is exactly same as the one in 2-, 3-dimensional BTCs, which we presented in Chapter 2. Also, with more iteration used, the BER performance will be improved. Figure 4.11 shows us the simulation for TPC with component $RM(32, 26)^2$, the block size is $32 \times 32 = 1024$, data size is $26 \times 26 = 676$, and code rate is $R = R_1 * R_2 = 0.66$. At BER of 10^{-5}, there is 0.12 dB coding gain loss for TPC $RM(32, 26)^2$ as compared to BTC $RM(32, 26)^2$.

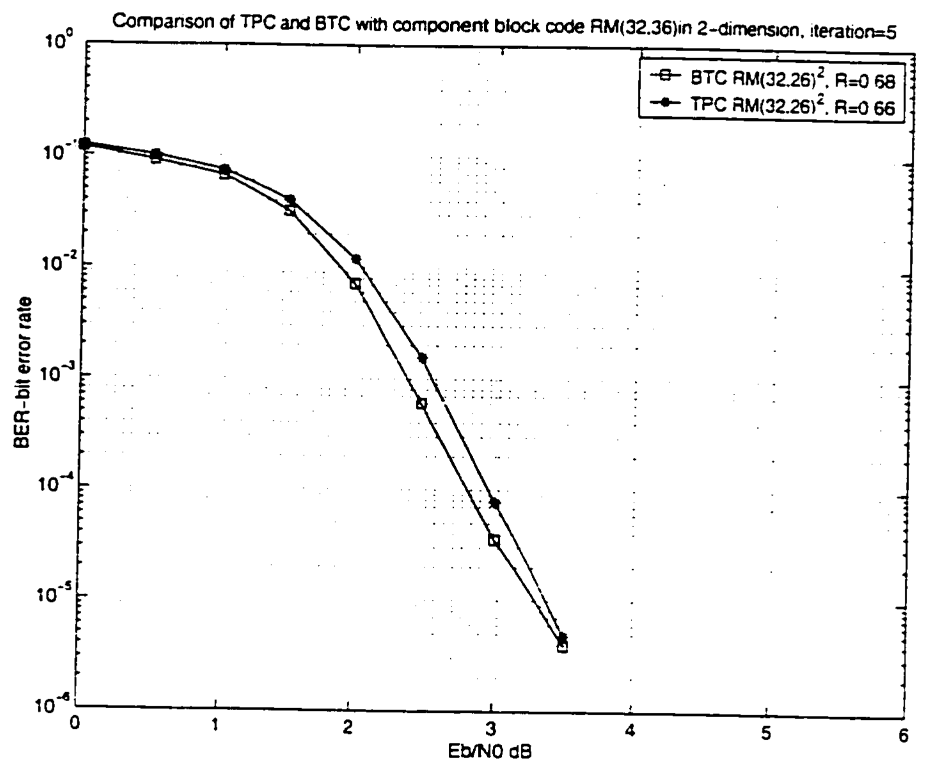


Figure 4.11: Comparing of BTC and TPC with component $RM(32, 26)$ in 2-dimension, iteration 5

4.5 Conclusion

This chapter presented the multi-dimensional BTCs as an example of three-dimensional $RM(32, 26)$ and $RM(8, 4)$. Since multi-dimensional BTCs can offer more choices for code rate and block size and avoid the using of puncturing that can deteriorate the performance and the decoding complexity increases only linearly with the dimension, the study and investigation of multi-dimensional BTCs is very meaningful. For 3D BTCs, we gave the structure of this kind of code and presented the encoder model consisting of three encoders for three different axis: x, y and z. At the receiver end, we explained how the decoder works with BCJR algorithm based on the Massey trellis. From the simulation results, multi-dimensional BTCs do provide more excellent performance than 2D BTCs do. To achieve a higher code rate, some of the bits in the code word must be punctured. Using puncturing method, we get we get the same code rate as the 2D code such that they can be compared in terms of BER performance. The simulations show that at low SNR, the puncturing one performs better than the 2D code, that is due to a better overall weight spectrum. When increasing the SNR, the puncturing one starts to perform worse than the 2D code due to a worse minimum distance. Due to a larger minimum distance, block sizes and lower code rate, multi-dimensional BTCs perform better than 2-dimensional BTCs though they use the same component codes.

In this chapter, we have talked about the quantization, this is the first and most important step of hardware design. Choosing the good quantizer will not affect the BER much. The simulations show us that the 4-bit quantization is the good tradeoff in terms of hardware design and BER performance.

As TPCs are the important part of block turbo codes, we briefly describe TPC in terms of construction product code, encoding and decoding algorithm. The product code encoder is the systematic block code encoder, which is the same as BTCs'. As they belong to the same class, their decoding procedure and decoding algorithms are similar.

Chapter 5

Applications

Block turbo codes and turbo product codes are a class of codes with a wide range of flexibility in terms of performance, complexity and code rate. This flexibility allows BTCs and TPCs to be used in a wide range of applications. For this reason. we suggest the use of BTCs and TPCs as alternatives for the forward and return channels of *DVB-RCS* (Digital Video Broadcasting - Return Channel via Satellite). The present DVB-RCS standard suggests the use of non-binary convolutional turbo codes. In this chapter, we will present the DVB-RCS standard and discuss the application of BTCs in this standard.

5.1 DVB-RCS Standard

DVB-RCS is a standard for Return Channel via satellite recently approved by the DVB-RCS Committee or EN 301 790 in ETSI (European Telecommunications Standard Institute `http://www.etsi.org/getastandard/home.htm`) to compete with ADSL (Asymmetric Digital Subscriber Line) and Cable modem to provide broadband access [47]. DVB-RCS system provides two-way, full-IP, asymmetric communications via satellite.

The standard illustrates the system model, which is to be used within DVB for interactive services. Two channels are established between the service provider and the user: the broadcast channel and the interaction channel. For the latter, a bi-directional Interaction Channel is established between the service provider and the user for interaction purposes. It is formed by:

- Return Interaction Path (Return Channel): it is from the user to the service provider. It is used to make requests to the service provider, to answer questions or to transfer data.
- Forward Interaction Path: from the service provider to the user. It is used to provide information from the service provider to the user and any other required communication for the interactive service provision.

We will present two types of traffic bursts: one is the ATM cells used in the return channel and the other is the MPEG2-TS packet used in the forward channel.

5.1.1 ATM cells

Standardization of ATM over Satellite is currently under progress and several organizations are working in parallel. Recently, there have been some important developments that have had a big impact on the use of ATM over Satellite. Most important is the trend using DVB-S to the end-user. This enables bundling of services such as video broadcast and IP traffic. The use of k ATM cells is recommended in the return channel. On the return link (from the end-user to gateway direction) the DVB Interaction channel for satellite distribution system may be the key to affordable transmit/ receive user terminals. In order to compare the performance of TPC with traditional coding methods, the ATM packet size of 53 bytes will be used. Figure 5.1 shows the ATM cell with $53 + 4$ bytes, in where 4 bytes is added to the packet for physical level administration. In order to code this packet size with block turbo code, a component code of $RM(32, 26)$ is used. This structure has a

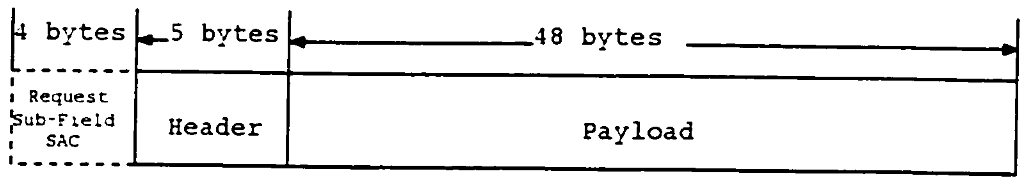


Figure 5.1: Turbo codes for ATM cell-based transmission

data size of 26^2 bits, or 84.5 bytes. In order to optimally fit the 57 byte packet, the code is shortened by 220 bits for block turbo code: $26^2 - 57 \times 8 = 220$. In order to optimally fit the 57 bytes packet, the code is shortened by 5 rows and 4 columns or vice versa, and the first row shortened by an additional 6 bits. The block turbo code is shortened as (714, 456) with code rate of 0.64. Note that other code rates are achievable with different shortening patterns [48]. The BER performance of this shortening pattern is better than the results in[49], which shows the performance of Reed-Solomon/convolutional concatenated codes for ATM cell transmission, by about 0.4 dB at BER of 10^{-5}.

5.1.2 MPEG2 Packet

Another application of the developed turbo codes can be the use of MPEG-2 systems digital audio and video transmission in satellite. MPEG is the family name of standard of coding audio-visual information (e.g., movies, video, music) in a digital compressed format. It is an ISO/IEC (International Organization for Standardization/ International Electrotechnical Commission) standard for medium quality and medium bit rate video and audio compression. It allows video to be compressed by the ratios in a particular range.

The MPEG2 transport stream has a frame composed of 188-byte packets containing program-specific information. The 4 additional bytes will be added in the beginning of the packet. According to the standard for DVB-RCS, it is recommended to use two coding schemes: turbo code and concatenated coding. As a kind of turbo code, BTC or TPC is alternative for the forward link to carry MPEG-2 Transport

Streams. For example, use the product code of $RM(64, 57)$ by shortening certain bits such as: $57 \times 57 - (188+4) \times 8 = 1713$ bits. The shortening pattern could be the following: shorten 25 rows of information bits and then shorten another 9 columns of information bits. The block turbo code is shortened as (2096, 1536) and the code rate is 0.733. Figure 5.2 shows the result of shortened $RM(64, 57)^2$ with the shortening pattern we presented. We compare the simulation result with Pyndiah's

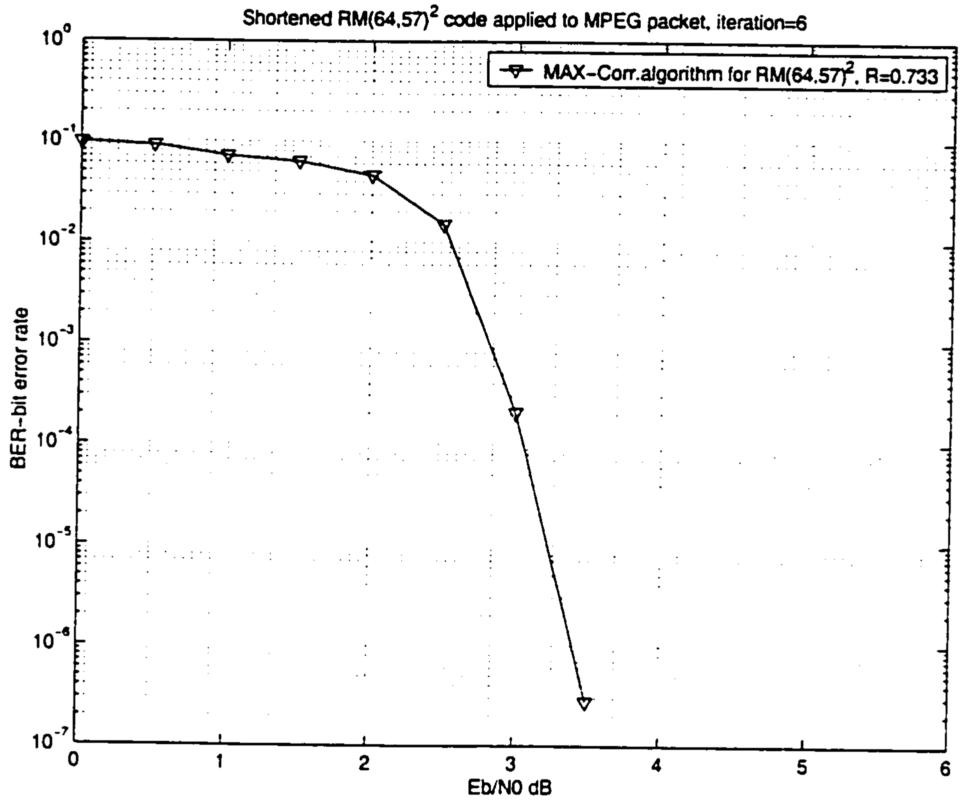


Figure 5.2: Shortened $RM(64, 57)^2$ with code rate $R = 0.733$ applied for MPEG packet

paper in 1998 [6] for $RM(64, 57)$, we can get a 0.2 dB coding gain. Moreover, the simulation result obtained from Pyndiah is based on the modified chase algorithm and without any performance loss in shortening. By comparing this to DVB-RCS, [47], which uses double-binary CTC, we can see that they are very close to each other by using the empirical formula: for 188-byte packet: BER $<\approx$ PER/300 for PER $< 10^{-7}$.

Chapter 6

Conclusion and Future Works

6.1 Conclusion for the Thesis

The purpose of this research is to study the characteristics of block turbo codes, turbo product codes, and to investigate ways to practical application as alternatives for the forward and return channels from DVB-RCS system. Turbo codes offer a performance superior to all other coding techniques. The main factors that make turbo codes so efficient include parallel concatenation structure of the encoding system, recursive convolutional encoder, interleaver, puncturing techniques and iterative decoding.

As another type of code, we found that the iterative decoding also performs very well when we use the linear block code as the component code for turbo code. With the linear block codes being the component codes and through different concatenation, we can easily get a larger number of block turbo codes with different code rates from low to high (up to 0.98) without puncturing. Also, compared to CTCs, BTCs use simple block interleaving (row/column), which is as good as random interleaving in CTCs. Some papers have already mentioned that BTCs are more efficient than CTCs in terms of high code rates [8], [7].

We briefly presented a few popular decoding algorithms used in Turbo Block

Codes in Chapter 2 and also presented in later Chapters how the construction of the encoding model for multi-dimensional BTCs. For the purpose of obtaining good performance, we take advantage of optimal MAP algorithm based on the trellis as the basic decoding scheme. Furthermore, for the purpose of reducing the implementation complexity, we employ an approximation method to avoid a lot of computations and we also introduced minimal trellis for the same purpose. Simulation results show that the coding gain degradation for BER performance can be negligible using the approximation method with a correction term. Through simulation, also showed how the number of iterations affect the turbo decoding performance.

For practical implementation, it is necessary to translate the floating point to fixed-point. For software implementation, a fixed-point model that corresponds to the given bit-width of quantization is to be found in order to avoid the much loss in BER performance. This thesis research also helps hardware implementation to find a reasonable bit-width quantizer to avoid much performance loss. Easier for hardware implementation leads to lower costs. So, this thesis also presented a little implementation design with the sample of 3-dimensional BTCs in Chapter 4.

A better approach to obtain higher coding gain is to use multi-dimensional turbo codes. The performance of multi-dimensional block turbo codes shows us that BTCs and TPCs are the more attractive option for a wide range of applications. Furthermore, we can get different code rates of block turbo codes by combining different linear block codes and with different dimensions in BTCs rather than using puncturing techniques to avoid the loss in BER performance. BTCs provide excellent performance at high code rates and can offer a very wide range of block sizes and code rates without change in coding strategy. BTCs offer a low complexity and higher efficiency than CTCs for higher rates. The simple row/column interleaving and linear block encoding in the BTC encoding system are a good example. All simulations of 2-, 3-dimensional BTCs in Chapter 3 and 4 show that the BTCs decoder is nearly optimal as compared to the convolutional turbo codes, thus, making the BTCs very attractive for practical application such as DVB-RCS system.

6.2 Future Works

In this thesis, we have investigated the BTCs from the encoding and decoding aspects. From the research, we know that the multi-dimensional BTCs have more potential and deserve further consideration. All the simulations were performed for these kinds of codes under ideal channel conditions for AWGN channel with BPSK or QPSK modulation. We would like to suggest the following future work:

1. Using minimal trellis reduces the complexity, but as the code length increases, the number of trellis state would increase exponentially, which can cause decoding delay and also much more complexity. So, we suggest that in the future work to use a more economical representation of the codes such as the sectionalized trellises.

2. To study the performance of BTCs and TPCs on other channel models, such as the fading channels; also use different modulation techniques such as OFDM.

3. Comparing to another decoding scheme usually applied for BTCs or TPCs: modified Chase algorithm. the chase algorithm is much less complex than suboptimal MAX*-Log-MAP algorithm based on the trellis, however, the performance is not as good as the latter. So, we have to further find a good decoding scheme to satisfy a better performance and less complexity. That is, to find an implementation having a complexity similar to the Chase algorithm and a performance close to MAP algorithm.

Bibliography

[1] FroRichard E. Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley Publishing Company, 1983

[2] Shu Lin, Daniel J.Costello, JR. *Error Control Coding,* Prentice-Hall, Englewood Cliffs, 1983

[3] C. Berrou, A. Glavieux, and P. Thitimajshuma, "Near Shannon limit error-correcting coding and decoding: turbo codes," *Proc. 1993 IEEE International Conference on Communications, Geneva, Switzerland,* pp. 1064-1070, May 1993.

[4] J. Lodge, R. Young, P. Hoeher, "Separable MAP filters for the decoding of product and concatenated codes," in *Proc.. IEEE Int. Conf. on Communication* (Geneva, Switzerland, May. 1993), pp. 1740-1745.

[5] R. M. Pyndiah, A. Glavieux, A. Picart and S. Jacq, "Near Optimum Decoding of Product Codes,: in *GLOBECM'94*, San Francisco, CA, Dec. 94, pp. 339-343.

[6] R. Mahendra Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Trans. Commun.*, vol. 46, pp.1003-1010, Aug. 1998.

[7] –,"Iterative decoding of product codes: Block turbo codes," in *Proc. Int. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp.71-79.

[8] S. Dave, J. H. Kim, S. C. Kwatra, "An Efficient Decoding Algorithm for Block Turbo Codes," *IEEE Trans. Commun.*, vol. 49, pp. 41-46, Jan. 2001.

[9] Divsalar, D. and Pollara F., "On the Design of Turbo Code", *The JPL TDA Progress Report 42-123*, Nov. 15, 1995

[10] J. Hagenauer, E. Offer, L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445, Mar. 1996.

[11] J. G. Proak, *Digital Communications*, McGraw Hill.

[12] B. Thomson, "Advanced Error Correction Enables Broadband Wireless," *Wireless System Design*, Jun. 2000.

[13] G. D. Forney, Jr., *Concatenated Codes. Cambridge*, MA: M.I.T. Press, 1996.

[14] S. Benedetto, G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409-428, Mar. 1996.

[15] B. Vucetic, J. H. Yuan, *Turbo Codes Principles and Applications*, Kluwer Academic Publishers, 2000.

[16] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbols Error Rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, 1974.

[17] J. A. Erfanian, S. Pasupathy and G. Gulot, "Reduced Complexity Symbol Detectors with Parallel Strucutres for ISI Channels," *IEEE Trans. Commun.*, vol. *42*, pp.1661-1671, Feb./Mar./Apr. 1994.

[18] J. Hagenauer,P. Hoeher, "A Viterbi Algorithm with Soft Decision Outputs and its Applications," *Proceedings of IEEE GLOBECOM*, pp. 1680-1686, Nov. 1989.

[19] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory,* vol. IT-13, No.2, pp. 260-269, Apr. 1967.

[20] D. Chase, "A Class of a Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 170-179, Jan., 1978.

[21] Kerouedian, S.; Adde, P. "Block Turbo Codes: towards implementation," *Electronics, Circuits and Systems,* 2001. ICECS 2001. *The 8th IEEE International Conference on,* Vol:3, pp. 1219-1222, 2001

[22] I. S. Reed, " A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IEEE Trans. Inform. Theory,* vol. IT-4, pp. 38-49, 1954.

[23] D. E. Muller, "Application of Boolean Algebra to Switching Circuit Design and to Error Detection," *IEEE Trans. Computers,* vol. 3, pp. 6-12, 1954.

[24] F. J. MacWilliams, N. J. A. Sloane, *The Theory of Error-Correcting Code,* North-Holland Publishing, 1981.

[25] V. S. Pless, W. C. Huffman, *Handbook of Coding Theory,* vol I & II, Elsevier Science B.V., 1998.

[26] J. K. Wolf, " Efficient Maximum Likelihood Decoding of Linear Block Codes," *IEEE Trans. Inform. Theory,* vol. IT-24, pp. 76-80, 1978.

[27] D. J. Muder, "Minimal Trellises for Block Codes," *IEEE Trans. Inform. Theory,* vol. 34, pp. 1049-1053, Sept., 1988.

[28] J. L. Massey, "Foundations and methods of Channel Coding," *NTG-Fachberichte (Proc. Int. Conf. on Information Theory and Systems),* vol. 65, pp. 148-157, 1978.

[29] R. J. McEliece, "On the BCJR Trellis for Linear Block Codes," *IEEE Trans. Inform. Theory.* vol. 42, No. 4, pp. 1072-1092, 1996.

[30] S. Lin, T, Kasami, T, Fujiwara, and M. Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*, Kluwer Academic Publishers, 1998.

[31] H. Nickl Nickl, J. Hagenauer, F. Burkert, "Approaching Shannon's Capacity Limit by 0.27 dB Using Simple Hamming Codes," *Communications Letters*, vol. 9, no. 5, pp.130-132, Sept. 1997.

[32] S. Dave, J. H. Kim, S. C. Kwatra, "An Efficient Decoding Algorithm for Block Turbo Codes," *IEEE Trans. Commun.*. vol. 49, no. 1, pp. 41-46, Jan. 2001.

[33] W. Peterson, E. Weldon, *Error Correcting Codes*, The MIT Press, Cambridge Mass., 1972.

[34] A. B. Kiely, S. Donlinar, R. J. McEliece, L. Ekroot, and W. Lin, "Trellis Complexity Bounds for Decoding Linear Block Codes," *The Telecommunications and Data Acquisition Progress Report 42-121, January-March 1995*, Jet Propulision Laboratory, Passdena, California, pp. 159-172, May 15, 1995.

[35] A. B. Kiely, S. Donlinar, R. J. McEliece, L. Ekroot, and W. Lin, "Trellis Complexity Bounds for Decoding Linear Block Codes," *The Telecommunications and Data Acquisition Progress Report 42-121, January-March 1995*, Jet Propulision Laboratory, Passdena, California, pp. 148-157, May 15, 1995.

[36] G. D. Forney, "Dimension/ Length Profiles and Trellis Complexity of Linear Block Codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1741-1752, 1994.

[37] C. Schlegel, *Trellis Coding*, IEEE PRESS, 1997.

[38] H. Michel, N. Wehn, "Turbo-Decoder Quantization for UMTS," *IEEE Communications Letters*, vol. XX, No. Y. Month 2000.

[39] Z. F. Wang, H. Suzuki, K. K. Parhi, "VLSI Implementation Issues of Turbo Decoder Design for Wireless Applications," *IEEE Workshop on Signal Processing Systems Design and Implementation*, Taibei, Taiwan, Oct 20-22, 1999.

[40] Y. Wu and B. D. Woerner, " The Influence of Quantization and Fixed Point Arithmetic upon the BER Performance of Turbo Codes," in *Proc. IEEE International Conference on Vehicular Technology (VTC Spring '99)*, May 1999, vol. 2 pp. 1683-1687.

[41] H. Michel, A.Worm, and N. Wehn, "Influence of Quantization in the Bit-Error Performance of Turbo-Decoders," *in Proc. VTC '00 Spring* Tokyo, Japan, May 2000.

[42] G. B. J, D. H, "Optimal Quantization for Soft-Decision Turbo Decoder," *in Proc. IEEE International Conference on Vehicular Technology VTC '99* Amsterdam, the Netherlands, Sept 19-22, 1999.

[43] T. K. Blankenship, "Design and Implementation of a Pilot Signal Scanning Receiver for CDMA Personal Communication Services Systems," *Master's thesis*, Virginia Tech, Apr. 1998.

[44] F. Mo, S. C. Kwatra, J. Kim, "Analysis of Puncturing Pattern for High Rate Turbo Codes," Dept. of Electrical Engineering and Computer Science, the University of Toledo, Toledo, OH 43606.

[45] S. M. Reddy, "On Decoding Iterated Codes," *IEEE Trans. Inform. Theory*, vol IT-16, Sept 1970, pp.624-627.

[46] J. Lodge, P. Hoeher and J. Hagenauer, "The Decoding of Multidimensional Codes Using Separable MAP Filters," *16th Biennial Symposium on Communications*, Kingston, Canada, pp. 343-346, May 1992.

[47] C. Doucillard, M. Jezequel, C. Berrou, N. Brengarth, J. Touch and N.Pham, "The Turbo Code Standard for DVB-RCS," *2nd International Symposium on Turbo Codes & Related Topics*, Brest, France, 2000, pp. 535-538.

[48] U. Vilaipornsawai, M. R. Soleymani, "Turbo Codes for Satellite and Wireless ATM," *in Proc. ITCC'01 International Conference on Information Technology: Coding and Computing*, Las Vegas, Nevada., 2-4 Apr, 2001

[49] M. Vanderaar, R. T. Gedney and E. Hewitt, "Comparative performance of turbo product codes and Reed-Solomon/convolutional concatenated codes for ATM cell transmission," *Fifth Ka Band Utilization Conf.*, Toarmina, Italy, October 1999.