MAP Decoding Algorithm in Turbo Code System

TAN CHAU WENG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electronic and Communication Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2012

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"MAP Decoding Algorithm in Turbo Code System"** was prepared by **TAN CHAU WENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electronic and Communication Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Mr.Balamuralithara

Date : _____

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ARQ	Automatic repeat request
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
ECC	Error Control Coding
FEC	Forward Error Correction
GSM	Global System for Mobile Communication
LLR	Log- Likehood Ratio
MAP	Maximum A Posteriori
ML	Maximum Likelihood
PCCC	Parallel Concatenated Convolutional Code
RSC	Recursive Systematic Convolutional
SNR	Signal to noise ratio
SOVA	Soft-Output Viterbi Algorithm
UMTS	Universal Mobile Telecommunication System
WCDMA	Wide Code Division Multiple Access

CHAPTER 1

INTRODUCTION

1.1 Background

In information theory, turbo codes are a class of high-performance forward error correction (FEC) codes developed in 1993, which were the first practical codes to closely approach the channel capacity, a theoretical maximum for the code rate at which reliable communication is still possible given a specific noise level. Turbo codes are finding use in (deep space) satellite communications and other applications where designers seek to achieve reliable information transfer over bandwidth or latency constrained communication links in the presence of data-corrupting noise. Turbo codes are nowadays competing with LDPC codes, which provide similar performance.

1.2 Aims and Objectives

The aim in this project is to make sure we gain the knowledge of turbo codes system. The decoder in turbo code system is very complex and involved with many algorithms. In order to get understand about the decoder algorithm, we will go through deeply in this project. This is to make sure that we can apply our knowledge of turbo code system in industrial area after we graduated.

1.3 Thesis Outline

The first two chapters in this thesis report are about the introduction for the communication system in your real world. I will go through the history of the communication system for the first generation until the fourth generation cellular network. After that, I will go through the Error Control Coding (ECC) and the type of the ECC. Moreover, the introduction for turbo code system will come afterward. The interleaving feature in turbo code system and puncturing system will be introduced in this chapter. The last part for chapter two is to talk about the decoding algorithm in the turbo code system.

The chapter three is all about the simulation on the Yufei Codes by using the Matlab software. I will go through the introduction for Matlab and Yufei Codes. In addition, I will capture image for the example during my simulation for the codes. The parameter for the simulation is frame size, generator polynomial, code rate, number of iteration and the final result for comparing the pros and cons of the Log-MAP algorithm and the SOVA algorithm.

I will come out with some result and discussion in the chapter four and the result is base on the simulation on the chapter three. The discussion for the performance of the decoding algorithm is based on the four parameters. The parameter is frame size, generator polynomial, code rate and number of iteration.

Last but not least, I will conclude with the future work that we can put more research on the few algorithms that has potential to have great advantage than turbo codes.

CHAPTER 2

LITERATURE REVIEW

2.1 History of Communication System

Wireless communications is the fastest growing segment of the communications industry. Indeed, it has captured the attention of the media and the imagination of the public. Cellular systems have experienced exponential growth over the last decade and there are currently around two billion users worldwide. As such, cellular phones become an important business tool and part of everyday life in most developed countries, such as US, UK, etc. (Seymour & Shaheen, 2011)

2.1.1 First Generation (1G)

In the late 1970s, AT&T Bell Laboratories developed the first US cellular telephone system called the Advanced Mobile Phone Service (AMPS) (Yong, 1979). AMPS was first deployed in late 1983 in the urban and suburban areas of Chicago by Ameritech. In 1983, a total of 40MHz of spectrum in the 800MHz band was allocated by the Federal Communications Commission for the Advanced Mobile Phone Service. In the 1989, as the demand for cellular telephone services increased, the Federal Communications Commission allocated an additional 10MHz for cellular telecommunications. The first AMPS cellular system used large cells and omnidirectional base station antenna to minimize initial equipment needs, and the

system was deployed in Chicago to cover approximately 2100 square miles (Yong, 1979).

The European Total Access Communication System (ETACS) was developed in the mid 1980s, and is virtually identical to AMPS, except it is scaled to fit in 25kHz channels used throughout Europe. Another difference between ETACS and AMPS is how the telephone number of each subscriber is formatted, due to the need to accommodate different country codes throughout Europe as opposed to area codes in the US (Yong, 1979).

2.1.2 Second Generation (2G)

Most of ubiquitous cellular network use since 1991 that is commonly called *second generation* or 2G technologies which conform to the second generation cellular standards. Unlike first generation cellular systems that relied exclusively on FDMA/FDD and analogue FM, second generation standards use digital modulation formats and TDMA/FDD and CDMA/FDD multiple access techniques.

The most popular second generation standards include three TDMA standards and one CDMA standard:

a) *Global System Mobile (GSM),* which supports eight time slotted users for each 200kHz radio channel and has been deployed widely in the cellular and PCS bands by service provider in Europe, Asia, Australia, South America and some parts of the US [2];

b) *Interim Standard 136* (IS-136), also known as North American Digital Cellular (NADC) or US Digital Cellular (USDC), which supports three time slotted users for each 30kHz radio channel and is a popular choice for carriers in North America, South America, and Australia;

c) *Pacific Digital Cellular (PDC),* a Japanese TDMA standard that is similar to IS-136 with more than 50 million users; and

d) The popular 2G CDMA standard *Interim Standard 95 Code Division Multiple Access (IS-95),* also known as cdmaOne, which supports up to 64 users that

are orthogonally coded and simultaneously transmitted on each 1.25MHz channel. CDMA is widely deployed by carriers in North America, as well as in Korea, Japan, China, South America, and Australia (Liberti & Rappaport, 1999), (Kim,2000), (Garg, 2000).

The 2G standards mentioned above represent the first set of wireless air interface standards to rely on digital modulation and sophisticated digital signal processing in the handset and the base station.

2.1.2.1 Evolution on 2G Cellular Network

Three different upgrade paths have been developed for GSM carriers, and two of these solutions also support IS-136. The three TDMA upgrade options include:

a) High Speed Circuit Switched Data (HSCSD)
b) General Packet Radio Service (GPRS)
c) Enhanced Data Rates for GSM Evolution (EDGE)

There options provide significant improvements in the Internet access speed and support the creation of new Internet-ready cell phone.

2.1.2.2 High Speed Circuit Switched Data (HSCSD)

As the name implies, High Speed Circuit Switched Data is a circuit switched technique that allow a single mobile subscriber to use consecutive user time slots in the GSM standard. That is, instead of limiting each user to only one specific time slot in the GSM TDMA standard, HSCSD allows individual data users to commandeer consecutive time slots in order to obtain higher speed data access on the GSM network. HSCSD relaxes the error control coding algorithms originally specified in the GSM standard for data transmissions and increase the available application data rate to 14,400 bps, as compared to the original 9,600 bps in the GSM specification. By using up to four consecutive time slots, HSCSD is able to provide a raw

transmission rate up to 57.6 kbps to individual users, and this enhanced data offering can be billed as a premium service by the carrier. HSCSD is ideal for dedicated streaming Internet access or real-time interactive web sessions and simply requires the service provider to implement a software change at existing GSM base stations.

2.1.2.3 General Packet Radio Service (GPRS)

General Packet Radio Service is a packet-based data network, which is well suited for non-real time Internet usage, including the retrieval of email, faxes, and asymmetric web browsing, where the user download much more data then it uploads on the Internet. Unlike HSCSD, which dedicates circuit switched channels to specific users, GPRS supports multi-user network sharing of individual radio channels and time slots. Similar to the Cellular Digital Packet Data (CDPD) standard developed for the North American AMPS systems in the early 1990s, the GPRS standard provides a packet network on dedicated GSM radio channels. GPRS retains the original modulation format specified in the original 2G TDMA standard, but uses a completely redefined air interface in order to better handle packet data access. GPRS subscriber units are automatically instructed to tune to dedicated GPRS radio channels and particular time slots for “always on” access to the network.

When all eight time slots of a GSM radio channel are dedicated to GPRS, an individual user is able to achieve as much as 171.2 kbps. Applications are required to provide their own error correction schemes as part of the carried data payload in GPRS. As is the case for any packet network, the data throughput experienced by an individual GPRS user decrease substantially as more users attempt to use the network or as propagation condition become poor for particular users. The implementation of GPRS merely requires the GSM operator to install new routers and Internet gateway at the base station, along with new software that redefines the base station air interface standard for GPRS channel and time slots.

2.1.2.4 Enhanced Data Rates for GSM Evolution (EDGE)

EDGE, which stands for Enhanced Data Rates for GSM Evolution is a more advanced upgrade to the GSM standard, and requires the addition of new hardware and software at existing base stations. Interestingly, EDGE was developed from the desire of both GSM and IS-136 operators to have common technology path for eventual 3G high speed data access, but the initial impetus came from the GSM user community.

EDGE introduces a new digital modulation format, 8-PSK which is used in addition to GSM's standard GMSK modulation. EDGE allows for nine different air interface formats, known as *multiple modulation and coding schemes* (MCS), with varying degrees of error control protection. Each MCS state may use either GMSK (low data rate) or 8-PSK (high data rate) modulation for network access, depending on the instantaneous demands of the network and the operating conditions. Because of the higher data rate and relaxed error control covering in many of the selectable air interface formats, the coverage range is smaller in EDGE than in HSDRC or GPRS. EDGE is sometimes referred to as Enhanced GPRS, or EGPRS (Frederic, 2001).

2.1.3 Third Generation (3G)

Mobile broadband networks are becoming increasingly faster and increasingly more pervasive. The Universal Mobile Telecommunications System (UMTS) is a visionary air interface standard that has evolved since late 1996 under the auspices of the European Telecommunications Standards Institute (ETSI). European carriers, manufacturers, and government regulators collectively developed the early version of UMTS as a competitive open air-interface standard for third generation wireless telecommunications.

The eventual 3G evolution for 2G CDMA system leads to cdma2000. Several variants of CDMA 2000 are currently being developed, but they all are based on the fundamentals of IS-95 and IS-95B technologies. The eventual 3G evolution for GSM,

IS-136, and PDC system leads to Wideband CDMA, also called Universal Mobile Telecommunications Service (UMTS). W-CDMA is based on the network fundamentals of GSM, as well as merged versions of GSM and IS-136 through EDGE. It is fair to say these two major 3G technology camps, cdma2000 and W-CDMA, will be remaining popular throughout the early part of the 21st century.

2.1.3.1 HSDPA (3.5G)

High-Speed Downlink Packet Access (HSDPA) is an enhanced 3G mobile telephony communication protocol in HSPA family, also dubbed 3.5G or turbo 3G which allows network on UMTS to have higher data transfer speed and capacity.

2.1.3.2 HSPA (3.75G)

The main difference for 3.7G standard compared with other standard is 3.75G based on High Speed Packet Access (HSPA). HSPA is an amalgamation of two mobile telephony protocols, High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA), that extends and improves the performance of existing WCDMA protocols (Seymour & Shaheen, 2011).

2.1.3.3 HSPA+ (3.9G)

3.9G or we called as Pre-4G is based on Evolved High-Speed Packet Access (HSPA+), is a technical standard for wireless, broadband telecommunication. HSPA+ was first defined in the technical standard 3GPP release 7. The pre-4G technology 3GPP Long Term Evolution (LTE) is often branded “4G”, but the first LTE release does not fully comply with the IMT-Advanced requirements. LTE has a theoretical net bit rate capacity of up to 100 Mbps in the downlink and 50 Mbps in the uplink if a 20MHz channel is used – and more if multiple-input multiple output (MIMO).

The world's first publicly available LTE service was opened in the two Scandinavian capitals Stockholm (Ericsson system) and Oslo (Huawei System) on 14 December 2009, and branded 4G. The user terminals were manufactured by Samsung. Currently, the two publicly available LTE service in the United Stated are provided by Metro PCS, and Verizon Wireless. AT&T also has an LTE service in the works (Seymour & Shaheen, 2011).

2.1.4 Fourth Generation (4G)

A 4G or we called as IMT-Advanced system is expected to provide a comprehensive and secure all-IP based mobile broadband solution to laptop computer wireless modems, Smartphone, and other mobile devices. Facilities such as ultra-broadband Internet access, IP telephony, gaming services, and streamed multimedia may be provided to users.

Currently, no industry group has required creating a formal definition of 4G. So, 4G can't be classified as an official technology in this moment. This has resulted in some near term technologies such as WiMAX or 3G Long Term Evolution (LTE) being classified as 4G. IMT-A is used to provide a seamless interworking at any network and terminal. IMT-A concept outlined data rates for 4G technology will achieve 100Mbps for high mobility, whereas 1Gbps of data rates for nomadic users. Highly mobile users are further defined as accessing the network at speeds up to 125 KMph, while maintaining network connectivity at speeds of up to 350 KMph. Data rates are not yet specified.

IMT-A are using Orthogonal frequency-division multiple access (OFDMA), multiple-input and multiple-output (MIMO) access technology (Tellabs,2008).

2.1.5 Summary of Generation Standards

In general, the most important feature among these generations is about the data rates and the multiple access techniques. We can see the huge differences started for the first generation cellular network until now we have the fourth generation cellular network with the data rate of 1Gbps at the maximum rates.

Tag	release	Important Feature additions	Data rates
1G		Voice	
2G		SMS and Circuit Switched data services	9.6kbps peak data rate
2.5G		General Packet Radio Services (GPRS)	
3G	R99	Circuit and Packet Switched MMS Services Location Services	
	R4	Enhancements	
3.5G	R5	HSDPA High-Speed Downlink Packet Access	14Mbps peak data rate
3.75G	R6	High-Speed Uplink Packet Access (HSUPA) HSPA = HSDPA (from R5) + HSUPA Multimedia Broadcast Multicast Services (MBMS) Enables multimedia broadcast services like Mobile TV	14 Mbps peak data rate (HSD A) + 5.74 Mbps peak data rate (HSUPA)
3.9G	R7	HSPA Evolution/HSPA+	28 Mbps Download peak data rate 11 Mbps Upload peak data rate
4G	R8, R9, R10	Long Term Evolution (LTE) System Architecture Evolution (SAE) Focused on Cell Edge User Through put, Average User Throughput, Coverage.	

Table 2.1: Summary of Generation Standards

(sayanthan, 2011)

2.2 Wireless Communications System

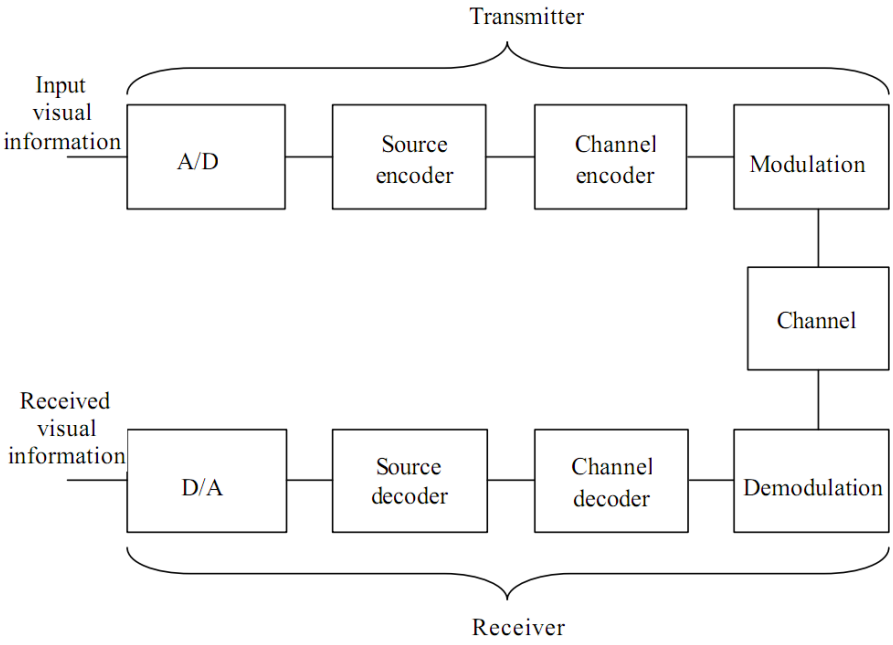

Figure 2.1: Block diagram of a wireless communication system

(Gersho, 1977)

Air is the medium of transmission in wireless communication system. This system allow multimedia communication from anywhere in the world using a small handheld device or laptop. In wireless communication system, the analogue sources must be converted into digital format. This is due to the reliability for digital format is better than analogue. Besides that, digital data is more easy and accurate to be detected during the error control coding process as compare to analogue data. It is easier to remove the unnecessary redundancy bits when the signal information presented in digital format. As we can see from figure 1, the transmitter consists of few of components such as, A/D converter, source encoder, channel encoder and modulator.

Source information from analogue signal or video signal which has discrete time and finite number of outputs will converted into digital form so that is easy to recognized by the source encoder. Before the quantization begins, we should start with the sampling process. The purpose for sampling process is to make sure that we can exactly reconstruct the analogue signal from the samples at the receiver. In order to get the signal correctly, we must perform the sampling properly. After we done the

sampling process, the system will continued with next process which called as quantization. Quantization is the process of mapping a continuous range of amplitudes of a signal into a finite set of discrete amplitudes and also an irreversible process. Quantization process will drop some of the information from the original analogue source. Quantizers can be thought of as devices that remove the irrelevancies. Unlike sampling, quantization introduces distortion. Amplitude quantization is an important step in any speech coding process, and it determines to a great extent the overall distortion as well as the bit rate necessary to represent the speech waveform. There have four types of quantization techniques as shown in figures 2 which are the uniform quantization, non-uniform quantization, adaptive quantization and vector quantization.

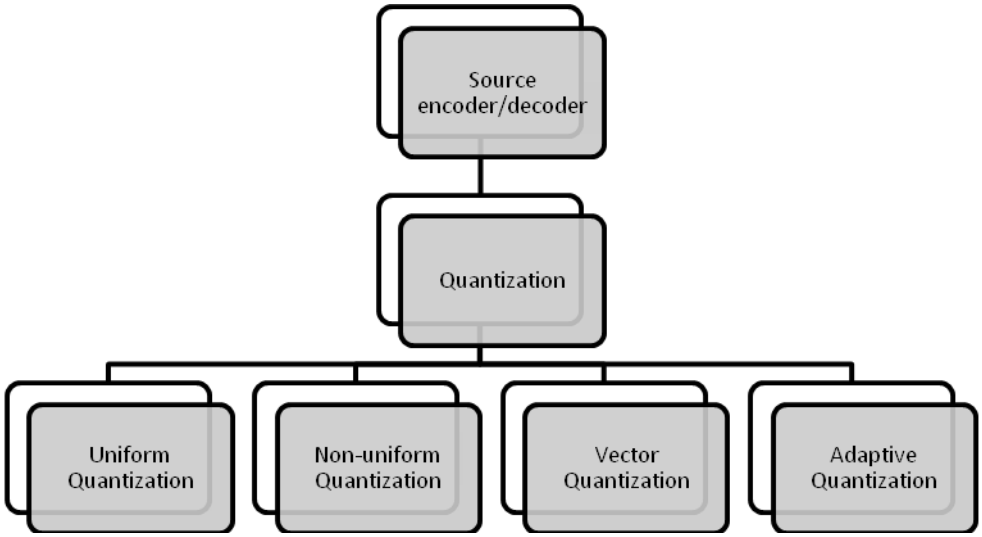

Figure 2.2: Block diagram for the Source encoder/decoder

After the quantization process, the signal will pass to the channel encoder for further process. Error control coding (ECC) is applied after the source information is converted into digital format by the source encoder. ECC is in principle a collection of digital signal processing techniques aiming to average the effects of channel noise over several transmitted signals. An important part of ECC is the incorporation of redundancy into the transmitted sequences. The number of bits transmitted as a result of the error correcting code is therefore greater than the needed to represent the information. Without this, the code would not even allow us to detect the presence of errors and therefore would not have any error controlling properties. Figures 3 showed some example for ECC in channel encoder.

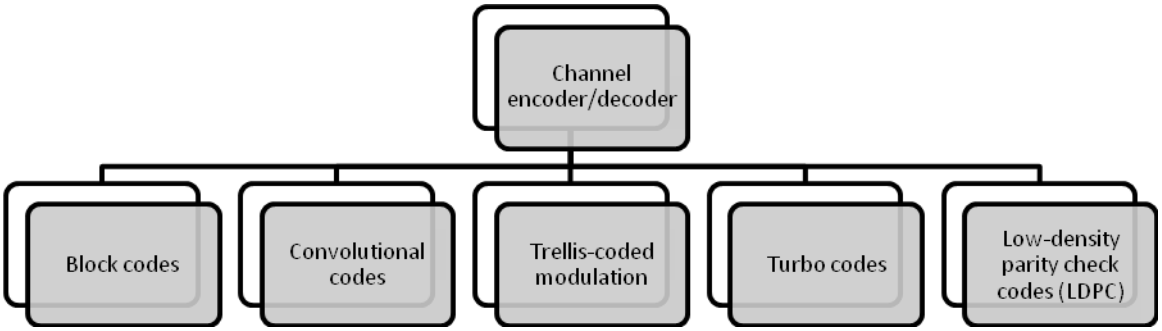

Figure 2.3: Block diagram for the Channel encoder/decoder

Before the signal is sending out to the air interface, the signal will pass through the modulator. Modulation is the process of encoding information from a message source in a manner suitable for transmission. There have two types of modulation which is analogue modulation and digital modulation. Amplitude Modulation (AM), Frequency Modulation (FM), and Phase Modulation is an analog modulation. For digital modulation which is Quadrature Phase Shift Keying (QPSK), Frequency Shift Keying (FSK), Minimum Shift Keying (MSK), Quadrature Amplitude Modulation (QAM), and so on. It generally involves translating a baseband message signal (called the source) to a bandpass signal at frequencies that are very high when compared to the baseband frequency. The bandpass signal is called the *modulated* signal and the baseband message signal is called the *modulating* signal. Modulation may be done by varying the amplitude, phase, or frequency of a high frequency carrier in accordance with the amplitude of the message signal. Demodulation is the process of extracting the baseband message from the carrier so that it may be processed and interpreted by the intended receiver (also called the *sink*). Figures 4 showed some example for some types of modulation.

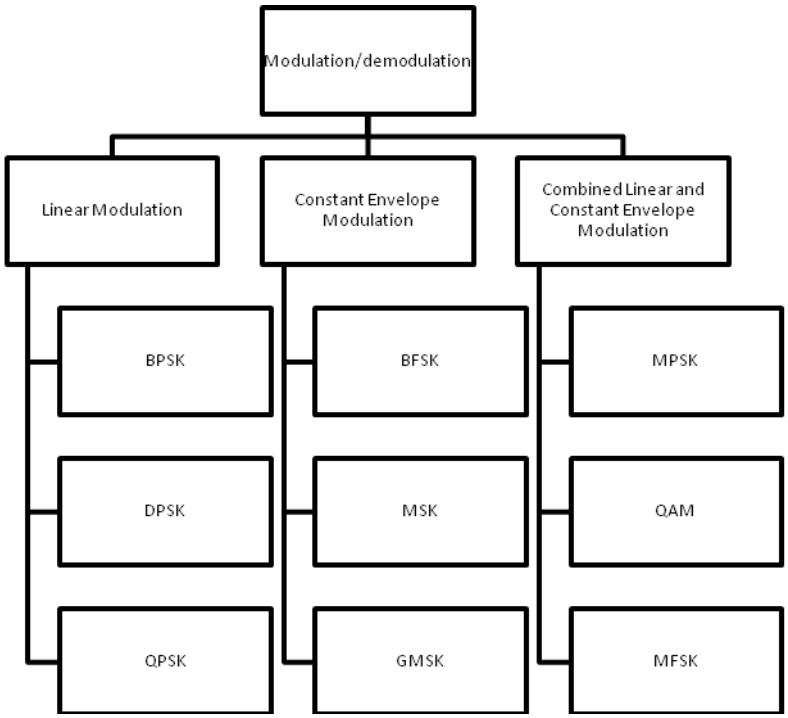

Figure 2.4: Block diagram for Modulation/demodulation

2.3 Error Control Coding (ECC)

In principle, error control coding is a collection of digital signal processing techniques aiming to average the effects of channel noise over several transmitted signals. The amount of noise suffered by a single transmitted symbol is much less predictable than that experienced over a longer interval of time, so the noise margins built into the code are proportionally smaller than those needed for uncoded symbols.

The incorporation of the redundancy into the transmitted sequences is the important part of the error control coding. The number of bits transmitted as a result of the error correcting code is therefore greater than that needed to represent the information. Without this, the code would not even allow us to detect the presence of errors and therefore would not have any error controlling properties. This mean that, in theory, any incomplete compression carried out by a source encoder could be regarded as having error control capabilities. In practice, however, it will be better to

compress the source information as completely as possible and then to re-introduce redundancy in a way that can be used to best effect by the error correcting decoder.

2.4 The Channel

The transmission medium introduces a number of effects such as attenuation, distortion, interference and noise, making it uncertain whether the information will be received correctly. Although it is easiest to think in term of the channel as introducing errors, it should be realized that it is the effects of the channel on the demodulator that produce the errors. On the other side, if the channel is without noise, there will be an unlimited amount of data to be transported to the receiver due to the infinite expansion also is an issue in the communications system.

2.4.1 Additive White Gaussian Noise (AWGN)

In communications, the additive white Gaussian noise (AWGN) channel model is one in which the information is given a single impairment which is a linear addition of wide band or white noise with a constant spectral density that is expressed as watts per hertz of bandwidth besides having a Gaussian distribution of amplitude. The model does not account for the phenomena of fading, frequency selectivity, interference, nonlinearity or dispersion. It does not s suffer from fading which means does not have to worry about distortion that a carrier-modulated telecommunication signal experiences over certain propagation media.

However, it produces simple and tractable mathematical models which are useful for gaining insight into the underlying behaviour of a system before these other phenomena are considered. Wideband Gaussian noise comes from many natural sources, such as the thermal vibrations of atoms in antennas or referred to as thermal noise, shot noise, black body radiation from the earth and other warm objects, and from celestial sources such as the Sun.

2.4.2 Rayleigh Fading

Rayleigh fading is a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices. Rayleigh fading models assume that the magnitude of a signal that has passed through such a transmission medium will vary randomly, or fade, according to a Rayleigh distribution - the radial component of the sum of two uncorrelated Gaussian random variables.

Rayleigh fading is viewed as a reasonable model for troposphere and ionosphere signal as well as the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver. If there is a dominant line of sight, Racing fading may be more applicable.

2.5 Types of Error Correction Codes

Error Correction Codes (ECC) is a very powerful technique that used in our wireless communication system. There have many codes algorithm in ECC such as linear block codes and convolutional code. The following are introduction for some example in ECC.

2.5.1 Liner Block Codes

Linear block codes are the basic error control coding algorithm. The basic property of linear block coder is called closure, and according to this property the sum of any two codeword is another codeword. The codeword of the linear block codes can be separated by two parts which is the part for parity bits and another part for message bits. Consider an (n,k) linear block diagram, the first portion of k bits is always identical to the message sequence to be transmitted. The second portion is n-k bits generalized parity check bits and is computed from the message bits in accordance

with a prescribed encoding rule that determines the mathematical structure of the code. Block codes in which message bits are transmitted in unaltered form are called systematic codes.

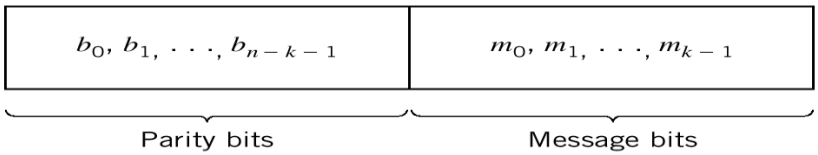

Figure 2.5: Structure of a systematic codeword

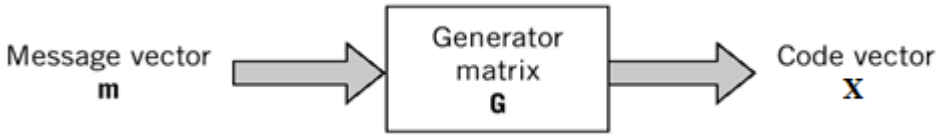

Figure 2.6: Structure of a generator matrix

Another way of expressing the relationship between the message bits and parity bit of linear block code is by parity check matrix. This matrix is used in decoding of the linear block codes.

$$\mathbf{H} = \left[\mathbf{I}_{n-k} \quad \vdots \quad \mathbf{P}^T\right]$$

$\mathbf{P}^T$ is an (n-k) x k matrix representing the transpose of $\mathbf{P}$

$\mathbf{I}_{n-k}$ is the (n-k) x (n-k) identity matrix.

2.5.2 Bose-Chaudhari-Hocquenqhem (BCH) Codes

BCH codes are one of the most important and powerful classes of cyclic codes. BCH codes are known for their multiple error correcting ability, and the ease of encoding and decoding. For any positive integers m (equal to or greater than 3) and t [< (2^m-1)/2] there exists a binary BCH code with the following parameters:

- **Block length:** $n = 2^m$**-1 symbols**
- **Number of message bites:** $\mathbf{k \geq n - mt}$
- **Minimum distance:** $d_{min} = 2t + 1$ **symbols**

Each BCH code is a *t*-error correcting code which means that it can detect and correct up to *t* random errors per code word. The Hamming single-error correcting codes can be described as BCH codes. Also, decoding algorithms for BCH codes can be implemented with a reasonable amount of equipment. The BCH codes provide a large selection of block lengths, code rates, alphabet sizes, and error correcting capability. (Bose, 2003)

2.5.3 Reed-Solomon Codes

Reed-Solomon codes are an important subclass of non-binary BCH codes. Reed-Solomon codes work on symbols rather than individual bits. The encoder of an (*n*,*k*) Reed-Solomon code receives input data stream in the form of blocks of *k* symbols each. The encoding algorithm expands each block of *k* symbols into a block of *n* symbols by adding *n*-*k* redundant (parity) symbols. A symbol is nothing but a combination of *m* bits. Thus an encoder converts blocks of *km* bits each into blocks of *nm* bits each. When *m* is an integer power of two, the *m*-bit symbols are called *bytes*. A popular value of m is 8; indeed, 8-bit Reed- Solomon codes are extremely powerful. (Bose, 2003)

A *t*-error correcting Reed Solomon code has the following parameters:

- **Block length:** $n = 2^m - 1$ **symbols**
- **Message size:** k **symbols**
- **Parity-check size:** $n-k = 2t$ **symbols**
- **Minimum distance:** $d_{min} = 2t + 1$ **symbols**
- **Number of correctable errors:** $t = \frac{1}{2}(d_{min} - 1)$ **symbols**

2.5.4 Convolutional Codes

An encoder for a binary block code takes a block of information bits and converts it into a block of transmitted bits (a codeword). A binary convolutional encoder takes a stream of information bits and converts it into a stream of transmitted bits, using a shift register bank. Redundancy for recovery from channel errors is provided by transmitting more bits per unit time than the number of information bits per unit time. Maximum likelihood decoding can be done using the Viterbi algorithm; other decoding algorithms such as SOVA (soft output Viterbi algorithm) and the BCJR algorithm are also commonly used. In practice the information stream is of finite duration and one typically appends a few termination bits to the input stream to bring the shift register bank back to the all zeros state, so that the convolutional code is in effect used as a very long block code. Often convolutional codes are used as inner codes with burst error correcting block codes as outer codes to form concatenated codes. Errors in Viterbi-like decoding algorithms for convolutional codes tend to occur in bursts because they result from taking a wrong path in a trellis. The burst error correcting capability of the outer code is used to recover from such burst error patterns in the decoding of the inner code (Ip & Tang, 2005).

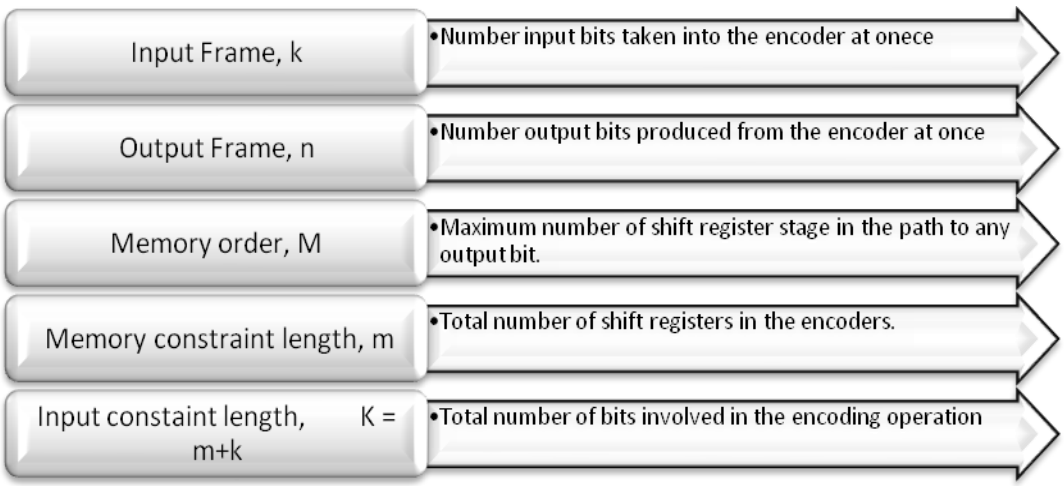

Table 2.2: Terminology for the Convolutional Code

2.5.5 Trellis Code Modulation (TCM)

TCM is which accomplishes this by the use of convolutional coding and increase the power efficiency. It conserves bandwidth by doubling the number of constellation points of the signal. This way the bit rate increase but the symbol rate stays the same.

Convolutional coding constrains allowed symbol transitions, creating sequence coding. Unlike a true convolutional coding, not all incoming bits are coded. Increasing the constellation size reduces Euclidean distances between the constellation points but sequence coding offers a coding gain that overcomes the power disadvantage of going to the higher constellation. Performance is measured by coding gain over an uncoded signal. The decoding metric is the Euclidean distance and not Hamming distance. Ungerboeck originally proposed TCM which used set-partitioning and small number of states with code rates that varied with the input signal type. TCM is a general concept and by varying k, we can create a QPSK, 8PSK, or higher level signals. (Bose, 2003)

2.5.6 Viterbi algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources, and more generally, hidden Markov models. The forward algorithm is a closely related algorithm for computing the probability of a sequence of observed events. These algorithms belong to the realm of information theory.

The Viterbi algorithm was conceived by Andrew Viterbi in 1967 as a decoding algorithm for convolutional codes over noisy digital communication links. The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech

recognition, keyword spotting, computational linguistics, and bioinformatics. For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal (Viterbi, 1967) .

2.6 Hard and Soft Decision

Kouraichi et al. (2004) found that hard-decision and soft-decision decoding refer to the type of quantization used on the received bits. The channel used in our study is the AWGN, In this channel, noise with uniform power spectral density (hence the term white) is assumed to be added to the information signal. It is a simple and common channel model in a communication system. Hard-decision decoding use 1-bit quantization on the received channel values. As shown in figure 3, we add a conformer that makes decision on the signal issued from the demodulator: if the signal voltage exceeds a certain value, we have logic "one", otherwise its output is equal to logic "zero".

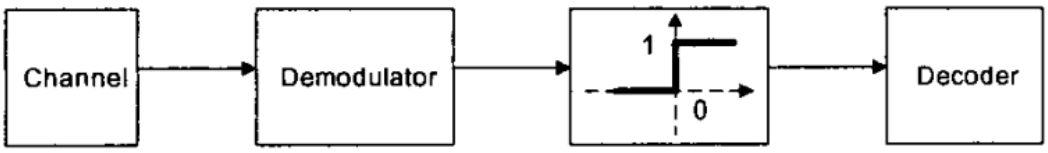

Figure 2.7: Hard decision decoding

(Kouraichi M, 2004)

Soft-decision decoding uses multi-bit quantization on the received channel values. For the ideal soft- decision decoding (infinite-bit quantization), the received channel values are directly used in the channel decoder. Figure 3 shows decision decoding.

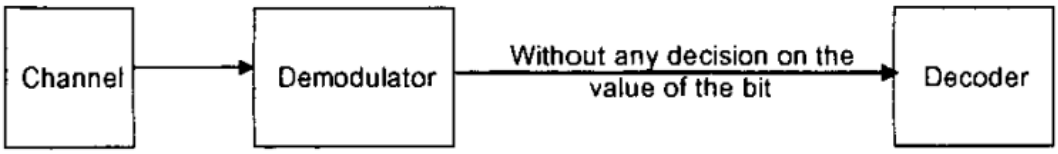

Figure 2.8: Soft decision decoding

(Kouraichi M, 2004)

In the soft decision decoding, we do not use the conformer that makes decision on the value of the bit but we inject it directly in the decoder. Thus all the perturbation caused by the imperfection of the channel will be integrated in estimation process. The estimation of the bit value will be done after the decoding process.

2.7 History of Turbo Codes

Claude Shannon published a classic paper in 1949 (Shannon, 1949) that established a mathematical basic for the consideration of the noisy communications channel. In his analysis, he came out with Shannon limit that quantified the maximum theoretical capacity for a communications channel and indicated the error-correcting channel codes must exist that allowed this maximum capacity to be achieved. The intervening years have been seen many well-considered channel codes inch toward the Shannon limit, but all contenders have required large block lengths to perform close to the limit. The consequent complexity, cost, and signal latency of these codes have made them impractical within 3 to 5 dB of the limit, but they provide useful coding gain at higher values of Eb/No and bit error rate.

In 1993 Berrou, Glavieux and thitimajshima proposed “a new class of convolution codes called turbo codes whose performance in terms of Bit Error Rate (BER) are close to the Shannon limit”. It has been claimed these codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required Eb=N0 of 0.7 dB was reported for a bit error rate (BER) of 10-5, using a rate 1/2 turbo code (Berrou et al, 1993).

It is theoretically possible to approach the Shannon limit by using a block code with large block length or a convolutional code with a large constraint length. The processing power required to decode such long codes makes this approach impractical. Turbo codes overcome this limitation by using recursive coders and iterative soft decoders. The recursive coder makes convolutional codes with short constraint length appear to be block codes with a large block length, and the iterative soft decoder progressively improves the estimate of the received message.

2.8 Turbo Code Encoder

A specific type of convolutional coder is used to generate turbo codes. The convolutional coder shown in Figure 1a has a single input, x, outputs p0 and p1, and a constraint length K=3. Multiplexing the outputs generates a code of rate R=1/2.

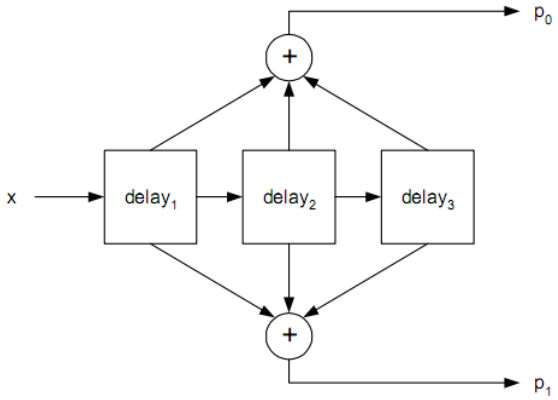

Figure 2.9: Convolutional coder

(Gomilko, 2008)

The convolutional coder shown in Figure 1b differs in that one of the outputs, p0, has been “folded back” and is presenting one of its output sequences at the coder input, making it recursive. This has the effect of increasing the apparent block length without affecting the constraint length of the coder. The input is also presented as one outputs of the coder, making it systematic. Such coders are thus called recursive systematic convolutional (RSC) coders.

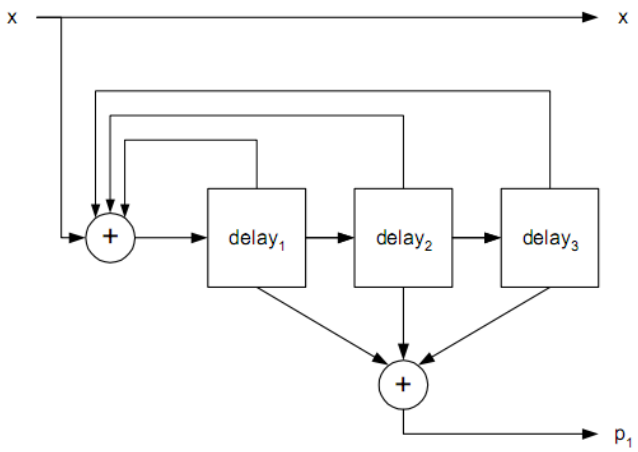

Figure 2.10: Recursive systematic convolutional coder

(Gomilko, 2008)

In non-recursive convolutional codes it is common practice to flush the coder with zeros to bring the decoder to an end state. Flushing with zeros does not readily work with recursive coders, however relatively simple binary arithmetic can establish the input sequence that will generate a zero state. RSC codes can thus be made to appear like linear block codes.

A turbo code is the parallel concatenation of a number of RSC codes. Usually the number of codes is kept low, typically two, as the added performance of more codes is not justified by the added complexity and increased overhead. The input to the second decoder is an interleaved version of the systematic x, thus the outputs of coder 1 and coder 2 are time displaced codes generated from the same input sequence. The input sequence is only presented once at the output. The outputs of the two coders may be multiplexed into the stream giving a rate R=1/3 code, or they may be punctured to give a rate R=1/2 code. This is illustrated in Figure 2.11.

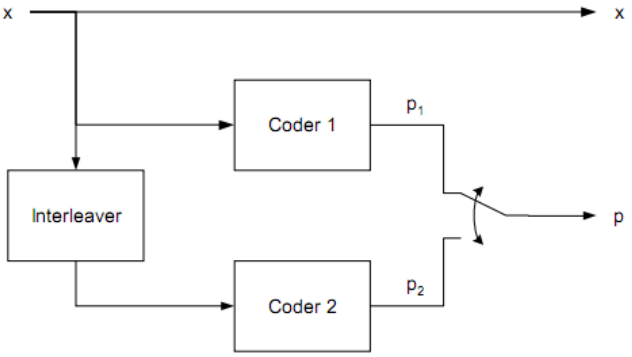

Figure 2.11: Punctured rate R=1/2 turbo coder (Gomilko, 2008)

The interleaver design has a significant effect on code performance. A low weight code can produce poor error performance, so it is important that one or both of the coders produce codes with good weight. If an input sequence x produces a low weight output from coder 1, then the interleaved version of x needs to produce a code of good weight from coder 2. Block interleavers give adequate performance, but pseudo random interleavers have been shown to give superior performance.

2.8.1 Interleaver

For turbo codes, an interleaver is used between the two component encoders. Interleaver plays an important role in determining the performance of a particular turbo code. The interleaver is used to provide ran domness to the input sequences.

For short interleavers, the performance of the Turbo code with a random interleaver degrades substantially up to a point where its BER performance is worse than the BER performance of convolutional codes with similar computational complexity. For short block length interleavers, selection of the interleaver has a significant effect on the performance of the Turbo code. In many applications, such as voice, delay is an important issue in choosing the block size.

2.8.2 Puncturing

Puncturing is a way of adding or removing parity bits based on the channel properties. Turbo coding introduces puncturing for two reasons. First is rate matching where bits are punctured so the number of coded bits fit the available bits in the physical channel. Another reason is to make different redundancy versions, adding more parity bits when the decoder fails to decode the transmitted bits. Puncturing can be used to prioritize between systematic bits and parity bits. An example of prioritizing could be seen between the first transmission and a retransmission. For the first transmission systematic bits should be prioritized so the decoder receives a minimum of redundancy. An error in the decoding of the first

transmission implies a need for redundancy and therefore priority bits should be prioritized.

Original Message:
"Prediction is very difficult, especially if it's about the future"
-Niels Bohr, Danish physicist (1885 - 1962)

Received message with burst error:
PredictionIsVeryDifficult,XXXXXXXXXXIfIt'sAboutTheFuture

Original message interleaved:
PioVDitpal'ohtrcneic,elfsueuetIrfuEclIAtFrdisyflsiytbTue

Interleaved message with burst error:
PioVDitpal'ohtrcneic,elfsuXXXXXXXXXXAtFrdisyflsiytbTue

De-interleaved message with burst error:
PrXdicXionXsVeXyDiXficXlt,XspeXiallyIfIt'sAboutThXFutXre

Figure 2.12: An example for interleaver.

(Gomilko, 2008)

2.9 Turbo Code Decoder

Although the encoder determines the capability for error correction, it is the decoder that determines the actual performance. The performance, however, depends upon which algorithm is used. Since turbo decoding is an iterative process, it requires a soft output algorithm like the maximum a-posteriori algorithm (MAP) or the Soft Output Viterbi Algorithm (SOVA) for decoding. Soft output algorithm out-perform hard decision algorithms because they have available a better estimate of what the sent data actually was. This is because soft output yields a gradient of information about the computed information bit rather than just choosing a 1 or 0 like hard output a typical turbo is shown in Figure 2.13.

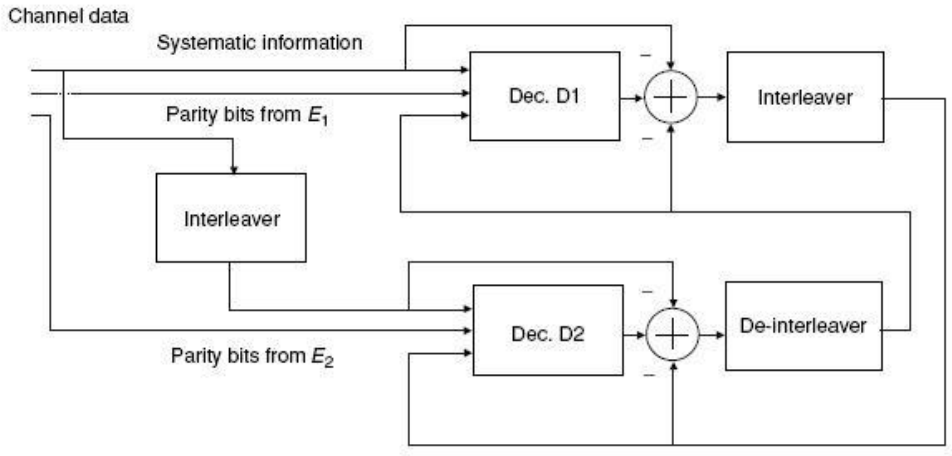

Figure 2.13: Turbo Code Decoder

(Harrison, 2001)

The MAP algorithm is often used to estimate the most likely information bit to have been transmitted in a coded sequence. The MAP algorithm is favoured because it outperforms other algorithms, such as the SOVA, under low SNR conditions. The major drawback, however, is that it is more complex than most algorithms because of its focus on each individual bit of information. In the past decade years, the research has resulted in great simplification of the MAP algorithm.

A turbo decoder generally uses the MAP algorithm in least one of its component decoders. The decoding process begins by receiving partial information from the channel and passing it to the first decoder. The rest of the information, parity 2, goes to the second decoder and waits for the rest of the information to catch up. While the second decoder is waiting, the first decoder makes an estimate of the transmitted information, interleavers it to match the format of parity 2, and sends it to the second decoder. The second decoder takes information from both the first decoder and the channel and re-estimates the information. This second estimation is looped back to the first encoder where the process starts again.

This cycle will continue until certain conditions are met, such as a certain number of iterations being performed. It is from this iterative process that turbo coding gets its name. The decoder circulates estimates of the sent data like a turbo engine circulates air. When the decoder is ready, the estimated information is finally

kicked out of the cycle and hard decisions are made in the threshold component. The result is the decoded

2.9.1 Parameters of Turbo Decoder

There are two main parameters in turbo decoder which is

1. Soft Input Soft Output (SISO)
2. Iterations

2.9.1.1 Soft Input Soft Ouput (SISO)

The important part of the turbo decoder is the general soft-input-soft-output a posteriori probability (APP) module that can be constructed codes. Several different algorithms can be used to implement the SISO decoder, such as MAP algorithm. According to (Benedetto, 1995), who proposed the concept of applying SISO APP general modules to decode parallel and serial turbo codes in 1996, the proposed universal SISO APP module is recognized as critical breakthrough in bridging the gap between diversified turbo codes besides having the advantage of being further applied into the design of more advance iterative decoding schemes.

These algorithms can be split into two groups. First group represents the algorithms derived from the Viterbi algorithm and the second group includes algorithms based on the Maximum a Posteriori (MAP) algorithm. The Maximum Likelihood Algorithms finds the most probable information sequence that was transmitted like SOVA. The MAP algorithm finds the most probable information bit to have been transmitted given (whole) coded sequence like BCJR (Bahl, Cocke, Jelinek, Raviv) (Gomilko, 2008). The SISO APP is normally equipped with the MAP algorithm in basic form and can be further extended to additive MAP (Log-MAP) or Max-Log MAP or even SOVA algorithms.

2.9.1.2 Iterations

Iteration is a computation of repeating a process. The more number of iterations during the decoding process, the better the performance because it will improve the BER for that frame. But the number of iterations can be increased up to a certain level where it will reach saturation stage and will produce optimum values. In this stage, even if the number of iterations is increased, the BER for that frame will not increase since it has already reached its optimum value. An example of the performance of iteration in turbo decoder is shown below, where as we can see the more number of iterations involved in the decoding process the better the BER performance.

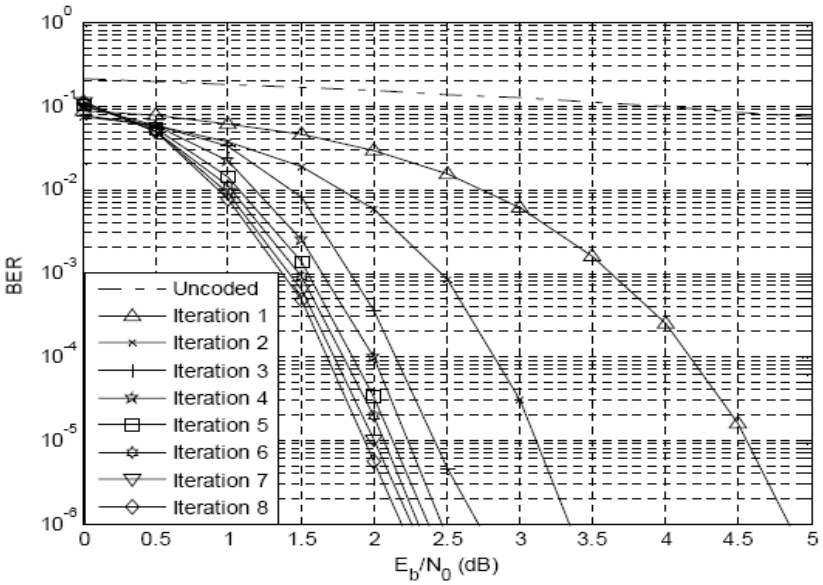

Figure 2.14: BER performance of different number of iterations

(Hsian, May 2005)

2.9.1.3 Branch metric computation – γ unit

In the algorithm for turbo decoding the first computational block is the branch metric computation. The branch metrics is computed based on the knowledge of input and output associated with the branch during the transition from one state to another. There are four states and each state has two branches, which gives a total of

eight branch metrics. The computation of branch metric is done using by taking the systematic bits of information with frame-length, the information that is fed back from one decoder to the other decoder, the channel estimate which corresponds to the maximum signal to distortion ratio, the encoded parity bits of the encoder, the noisy observed values of the encoded parity bits and the observed values of the encoded systematic bits. The γ unit takes the noisy systematic bit stream, the parity bits from first and second encoder to first and second decoder respectively and the a priori information to compute the branch metrics. The branch metrics for all branches in the trellis are computed and stored.

2.9.1.4 Forward metric computation – α unit

The forward metric is the next computation in the algorithm. This represents the probability of a state at time, given the probabilities of states at previous time instance where the summation is over all the state transitions to be computed at each node at a time instance *k* in the forward direction traversing through the trellis.

This metric is termed as forward metric because the computation is in an increasing order is from and the value for index time instance, *k* is initialized to zero. The unit recursively computes the metric using values computed in the above step. A forward recursion on the trellis is performed by computing α for each node in the trellis, α is the sum of the previous alpha times the branch metric along each branch from the two previous nodes to the current node.

2.9.1.5 Backward metric unit – β unit

The backward state probability being in each state gives the knowledge of all the future received symbols, which is recursively calculated and stored. The backward metric is computed in the backward direction going from the end to the beginning of the trellis. Backward metric computation can start only after the completion of the computation by the γ unit. Observing the trellis diagram, there are

four β values to be computed, but now in backward order, from [*N*-1 down to 0] .The four β values and the initialized values for index *k* equal to *N*-1. A backward recursion on the trellis is performed by computing β[*k*] for each node in the trellis. The computation is the same as for α, but starting at the end of the trellis and going in the reverse direction.

2.10 Decoding Algorithm

Some of the major decoding approaches, developed for turbo decoding are:

1. Maximum Aposteriori Probability (MAP)
2. Log-MAP,
3. Max-Log-MAP and
4. Soft Output Viterbi Algorithm (SOVA)

2.10.1 Maximum Aposteriori Probability (MAP)

The MAP decoding algorithm for convolutional codes was proposed over two decades ago, but initially received very little attention because of its increased complexity over alterative convolutional decoders for a minimal advantage in BER performance. Recently, the MAP decoder has enjoyed renewed and greatly increased attention as an iterative soft-output decoder for the class of turbo codes, which were discovered in 1993 and are planned to be used in 3rd generation standards.

MAP algorithm is a trellis-based decoding algorithm which is also known as the BJCR algorithm. The MAP algorithm is most favourable for convolutional codes and it minimizes the number of bits decoded wrongly. The input bits x_k and parity bits y_k may include additive white Gaussian noise at time *k*. The MAP decoding algorithm is devised to minimize the bit error probability. The MAP algorithm is computationally complex and sensitive to SNR mismatch and inaccurate estimation of the noise variance. This algorithm requires non-linear functions for computation

of the probabilities and both multiplication and addition are also required to compute the variables of this algorithm.

The MAP algorithm is a forward-backward recursion algorithm, which minimizes the probability bit error but has computational complexity and numerical instability. The forward state probability of being in each state of the trellis at each time *k* given the knowledge of all the previous received symbols is recursively calculated and stored. The backward state probability of being in each state of the trellis at each time *k* given the knowledge of all the future received symbols is recursively calculated and stored. The solution to these problems is to operate in the log-domain.

MAP algorithm finds the marginal probability that the received bit was 1 or 0. Since the bit 1 (or 0) could occur in many different code words, we have to sum over the probabilities of all these code words. The decision is made by using the likelihood ratio of these marginal distributions from 1 and 0. The calculation can be structured by using trellis diagram. For every state sequence there is a unique path through the trellis and vice versa. The objective of the decoder is to examine states and compute a posteriori probability (APP) associated with the state transitions. There are two major problems with MAP decoding algorithm. First, MAP requires accurate estimation of the noise variance and its performance is very sensitive to SNR mismatch. The fixed-point representation of the MAP decoding variables usually require between 16 to 24 bits for a QPSK signal group

To construct the turbo decoder two MAP decoders with the same function as well as an interleaver and a deinterleaver are needed. These components are arranged as shown in the figure corresponding to the turbo encoder.

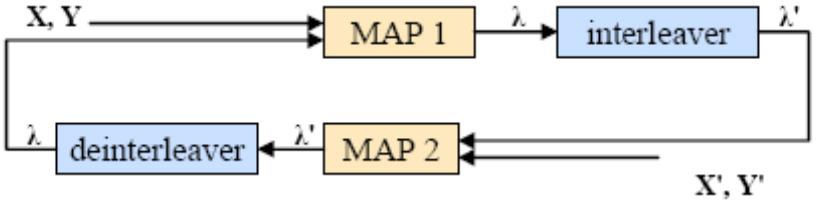

Figure 2.15: Turbo decoder using MAP

(Kharagpur, 2004)

In the structure above, X and Y are input parameters of the first MAP decoder. The X input refers to the received systematic symbols while Y refers to the received coded symbols. The interleaved sequences of the received bits X' and Y' are input parameters of the second MAP decoder. Each MAP decoder computes the extrinsic information known as lambda or λ. The second MAP produces lambda values with an interleaved sequence λ'. The hard decision, whether a bit is 0 or 1, is a function of the lambda values.

The initial lambda value which is the first set of λ supplied to MAP 1 is set to zeros. The absolute values of lambda are expected to be larger after iteration. One turbo decoding iteration consists of two MAP decoder iterations. The number of iterations to run depends on the power capability, speed requirement, and decoding performance requirement. The decoding performance increases greatly after the first few turbo decoding iterations, but starts to show insignificant improvements after several decoding iterations. The hard decision from MAP 1 is taken directly from the lambda produced by MAP 1, but the hard decision from MAP 2 must be taken from after the de-interleaving process of the lambda output from MAP 2. There are three main types of internal signals which are calculated and used within the Map algorithm. They are named alpha α, beta β, and lambda λ. The calculation of alpha and beta are independent of each other but the calculation of lambda for a symbol requires alpha of that symbol and beta of the next symbol.

2.10.2 Log-MAP

MAP algorithm was ignored by designers during a long time because of the hardware complexity required by multipliers and exponential units. In instead of a complex architecture of the MAP algorithm in the time domain, we develop architecture in the log-domain. This converts all multiplications to additions, divisions to subtractions, and eliminates exponentials entirely, without affecting BER performance. There are other reasons use the log domain like logs and exponentials can be eliminated and the numbers do not grow as rapidly. Because the forward and

backward recursions require successive multiplications by numbers less than one, even 32-bit floating-point numbers will underflow unless they are scaled.

Scaling requires additional operations that will slow down the turbo decoder. By going to the log domain no scaling is needed for 32-bit fixed-point numbers, and minimal scaling can be employed to utilize 16-bit or even 8-bit numbers. Also, avoiding the use of multiplications will reduce the power consumption of the processor.

Another advantage of going to the log domain is that the desired output of the algorithm, the log likelihood ratio or LLR, is in the log domain so it is automatically produced without having to actually take a logarithm. The LLR is the ratio of the logs of the probabilities that the particular data bit is a 1 or a 0.

The major drawback of the Log-MAP decoder is its huge requirement of memory storage for storing the state metrics before finally obtaining the likelihood decision especially in a decoder not employing the sliding window technique. To solve this problem a low power implementation of the MAP decoder through the forward recursive calculation of reverse state metrics was presented. This is based on the algorithmic optimization of the MAP algorithm.

To avoid the performance degradation of the decoders due to the modification of the conventional MAP algorithm to compute the reverse state metrics in a forward recursive manner, the complete frame or the block of data to be decoded is divided into several smaller blocks.

2.10.3 Max-Log-MAP

Similar to Log-MAP but replaces the Max-Log operation with taking maximum. Because at each state in forward and backward calculations only the path with maximum value is considered the probabilities are not calculated over all the codeword. In recursion calculation of α and β also only the best transition is

considered. The algorithm gives the logarithm of the probability that only the most likely path reaches the state. In calculations of log likelihood ratio only two codeword are considered. The best transition that would give +1 and the best transition that would give -1. Max-Log-MAP performs worse than MAP and Log-MAP

2.10.4 Soft Output Viterbi Algorithm (SOVA)

The Soft-Output Viterbi Algorithm (SOVA) is a variation of the Viterbi algorithm. This algorithm has two modifications (Viterbi, 1998) over the classical Viterbi algorithm. First, the path metrics used to select the maximum likelihood path through the trellis are modified to take account of a-priori information. Second, the algorithm is modified to provide a soft output for each decoded bit.

Consider the operation of a Viterbi algorithm. At some time t , each surviving path in the trellis denotes a series of add/compare/select operations, each resulting in the selection of a value for an information bit or symbol. Hagenauer and Hoeher noted that the probability that a given value is correct is proportional to how close the algorithm came to selecting the other value (or values).

CHAPTER 3

METHODOLOGY

3.1 Introduction

To achieve the simulations for Turbo Code System, MATLAB software is used. The simulations results are displayed in graphs where the Bit Error Rate (BER) is plotted versus the Eb/N0. All graphs are plotted by using Microsoft Excel.

3.2 Simulation Model

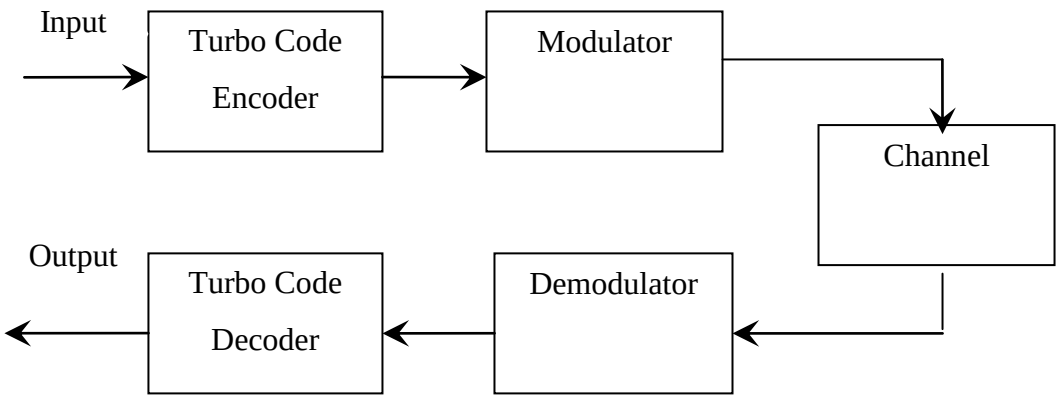

Figure 3.1: Block diagram of simulation model

The simulation in the communication system is represented by a standard description which is the block diagram, where each block contains the algorithm and equations that represents the signal processing operation. The sequence of binary

digits from which is known as the information sequence is passed on to the channel encoder redundancy will be introduced. Error control codes will calculate extra control bits from the information that we wish to transmit and then transmit those control bits together with the information sequence for the decoder to detect or correct the most possible error patterns.

In the turbo encoder, the fundamental turbo code encoder is built using two identical recursive systematic convolutional (RSC) codes with parallel concatenation that provides a very straightforward means of achieving a long, complex code out of much shorter component codes, which can be decoded much more easily. An interleaver is used between the encoders to improve burst error correction capacity or to increase the randomness of the code. Puncturing takes place after the encoder to change the overall coding rate. The binary symbol outputs are used to modulate where Binary Phase Shift Key (BPSK) modulation is used. At the receiver end the signal is faded and added to Additive White Gaussian Noise.

Before decoding the signal is demodulated before passing to the presenting to turbo decoder. The turbo decoder is implemented using two constituent decoders. Each decoder uses either Log-MAP or SOVA decoding algorithm. The entire procedure, from generation of the information bits until the decoding process where decoder makes decision, is repeated frame-by- frame according to the number of frames needed to be simulated.

3.3 Yufei Codes

Yufei codes were developed by Wu, Yufei in Nov 1998 (Wu). Yufei Wu is a physician from Virginia Tech University. This script simulates the classical turbo encoding-decoding system where the encoder architecture consists of parallel concatenation through random interleaver of two RSC component encoders while the decoder architecture consists of iterative cooperation between soft-input-soft-output decoders for the constituent codes. Random information bits are modulated into +1/-1, and transmitted through an Additive White Gaussian Noise (AWGN) channel.

Log-MAP algorithm without quantization or approximation is used. Using ln (e^x+e^y) = max(x,y) + ln(1+e^(-abs(x-y))),the Log-MAP can be simplified with a look-up table for the correction function. If use approximation ln (e^x+e^y) = max(x,y), it becomes MAX-Log-MAP. SOVA decoding algorithm is also one optional.

3.4 Introduction to MATLAB

This chapter introduces MATLAB 7.3 software, which is used to construct the turbo code system. The design of turbo code system with Additive White Gaussian Noise (AWGN) as its main channel condition is done using this software.

3.4.1 Description of MATLAB

MATLAB is a sophisticated language for technical computing. It integrates wide range of applications, algorithm development, computation, visualization, and programming in a comprehensible environment where problems and solutions are expressed in recognized mathematical notation. It also has interactive tools for iterative exploration, design and problem solving. The software MATLAB had been known as the well-organized user friendly software besides providing the most efficient services for the design simulation of this project.

3.5 Design the Model

3.5.1 Create and Launch Program

First start MATLAB to design the software. Then select File which is placed on top of the MATLAB command window. Secondly to creating a new model click File >

(Wu)New > M-file which is placed at the top of the MATLAB editor. Next step is done by clicking Open >Turbo Sys Demo > Run to launch and run the simulation codes. In order to meet the purpose of this research, simulation is done by modifying the Matlab script developed by Yufei (Wu) files.

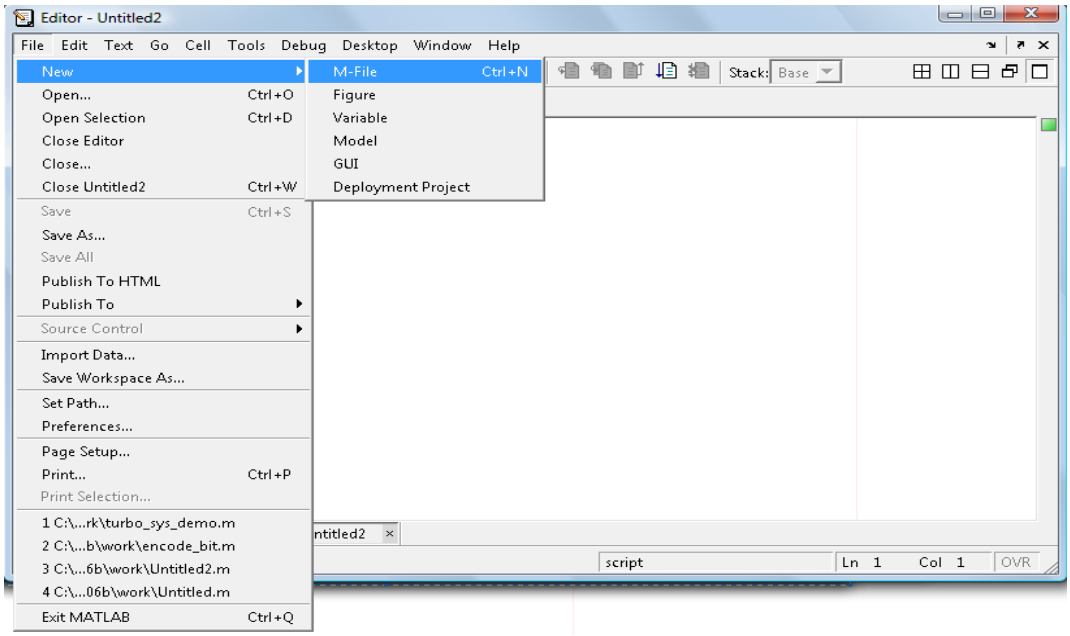

Figure 3.2: Creating and launching the program

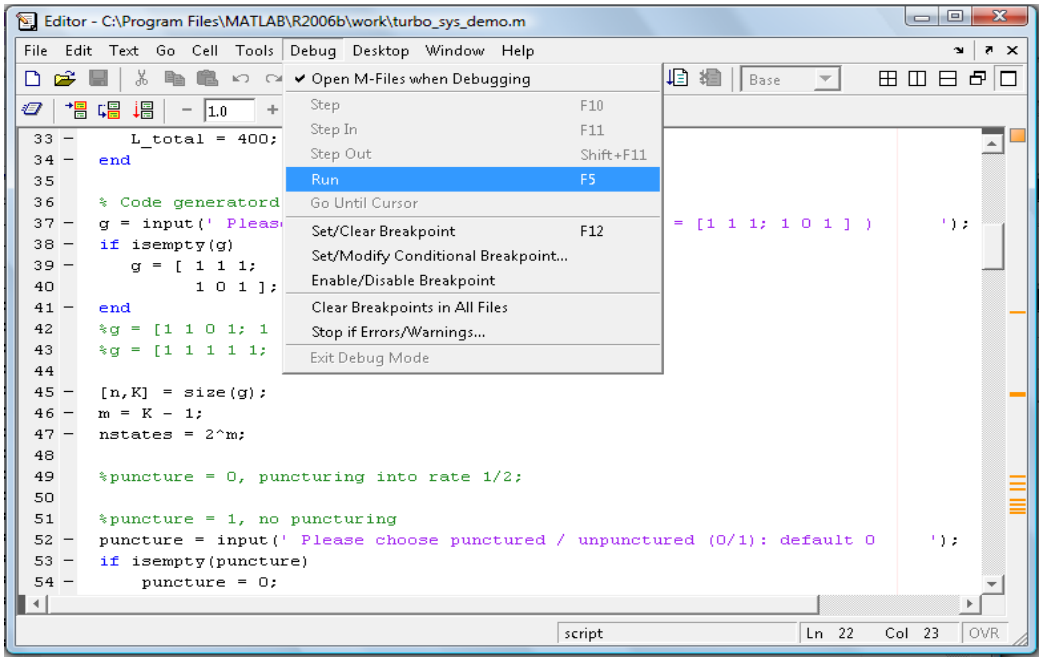

Figure 3.3: Running Yu Fei Codes

3.5.2 Simulation

After running the codes, the simulation is then conducted. This is done by inserting and varying the values of the parameters such as type of decoding algorithm, frame size, generator polynomial, puncturing, iterations, and frame errors to terminate at different Signal to Noise ratio. The simulation results will be then obtained.

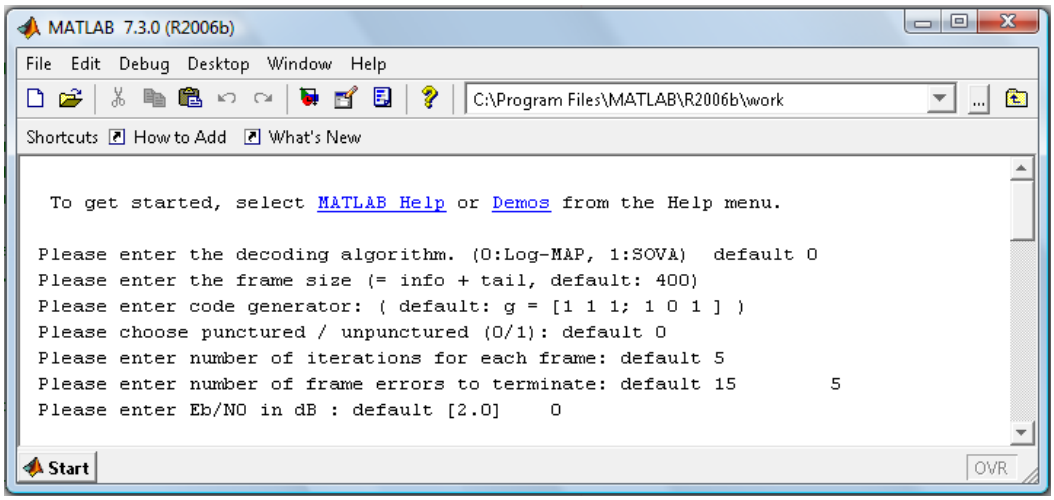

Figure 3.4: Running the simulation

3.5.2.1 Simulation for frame size

Firstly the RSC Encoder is terminated with tails bits which is the combination of information bits and tail bits and then scrambled and passed on to the second encoder, while second encoder is left open without tail bits of itself. For this parameter frame size of N=40,400 and 1000 is used to compare the differences of frame size in Log MAP and SOVA decoding algorithm.

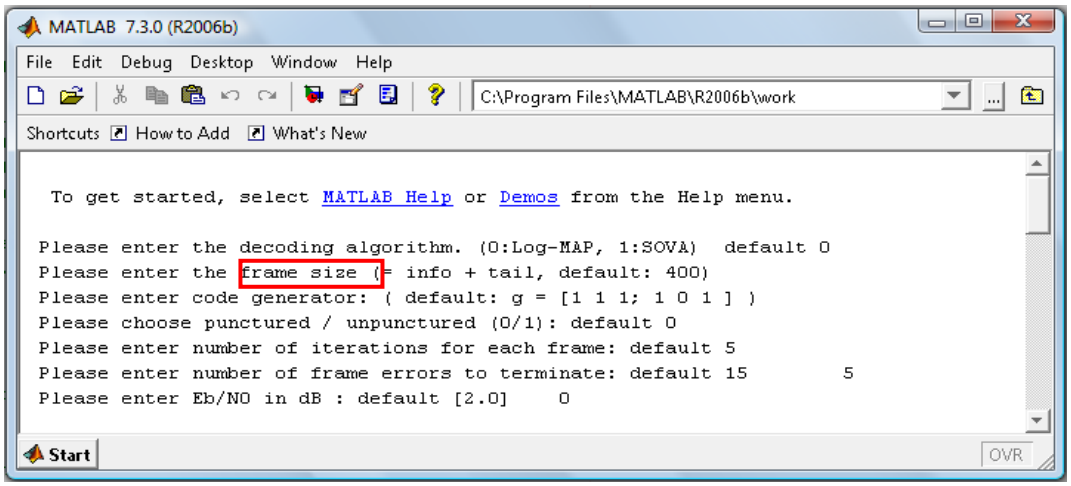

Figure 3.5: Simulation window for frame size

3.5.2.2 Simulation for generator polynomial

For this parameter, generator sequences is represented by values in an octal format where [feedback,feedfoward]. The values used are [111,101] and [1111,1011] which is used in both encoders simultaneously to adds randomness to the turbo code.

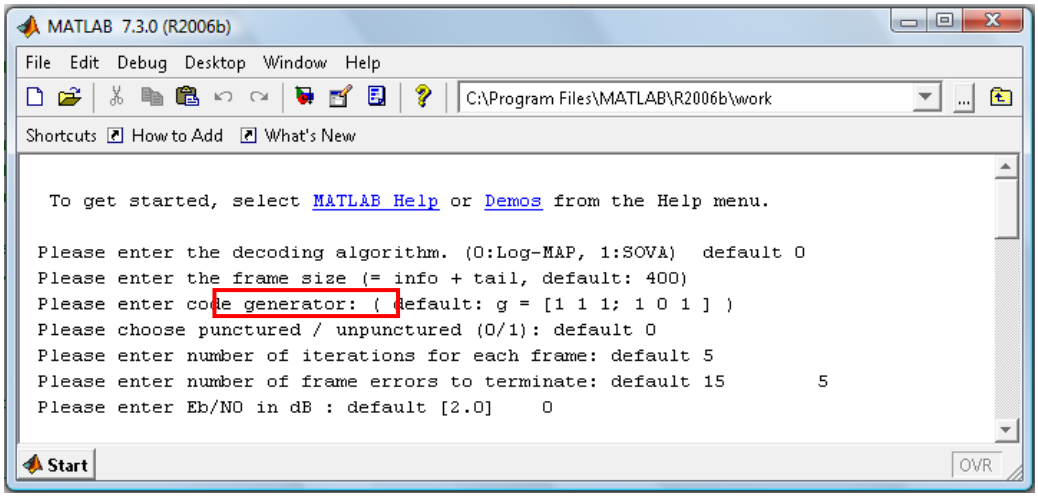

Figure 3.6: Simulation window for polynomial generator

3.5.2.3 Simulation for code rate

The outputs of both encoders are punctured. The puncturing is an optional where it changes according to required code rate needed. Puncturing gives lower code rate value (1/2) or no puncturing which gives higher code rate value (1/3).

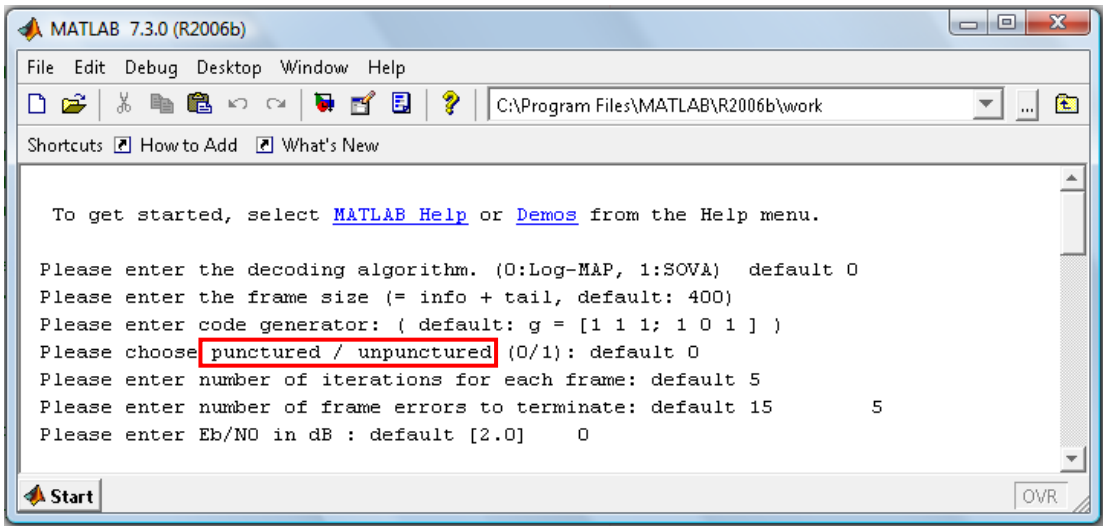

Figure 3.7: Simulation window for code rate

3.5.2.4 Simulation for number of Iterations

Iterations are used to reduce the number of errors in different channel condition such as condition at low Eb/No or high Eb/No. For this simulation iteration 5, 6 and 7 is tested in the turbo decoder.

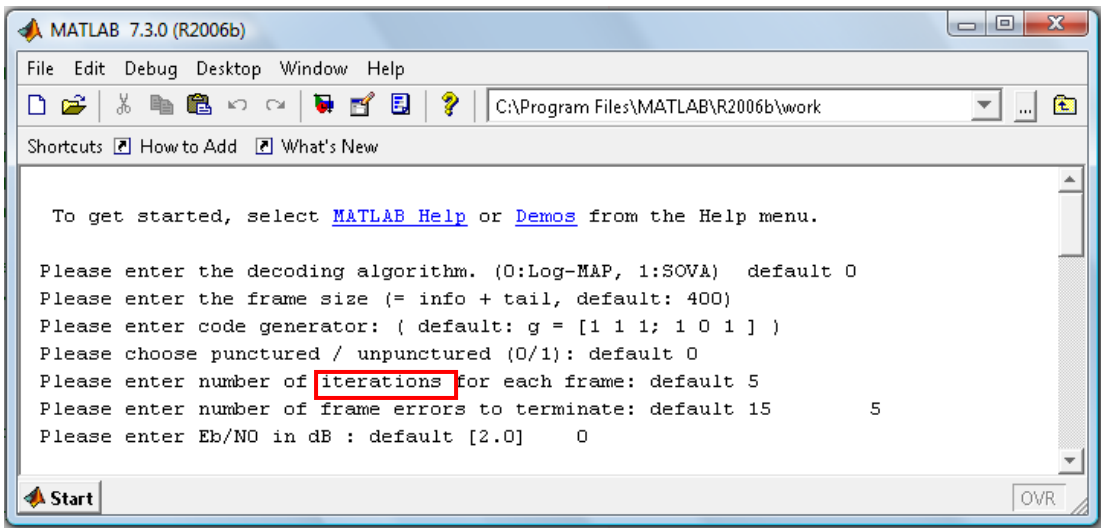

Figure 3.8: Simulation window for number of iterations

3.5.2.5 Simulation Results

Once all the simulations for all the parameters tested are completed, the results are obtained synchronized . The last value of each simulation will be value for that particular SNR value. And the whole process is repeated until the desired amount of SNR values are obtained. The results are then displayed in a graph form using Microsoft Excel for further analyses and discussion.

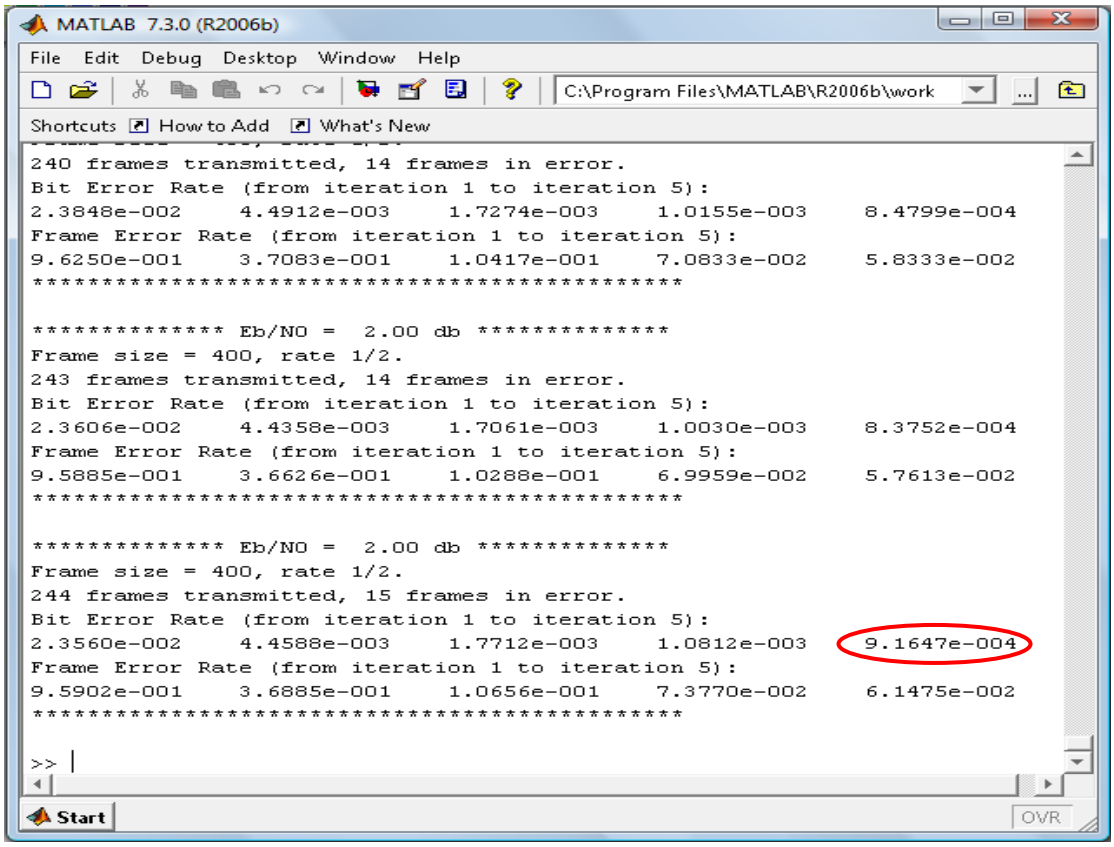

Figure 3.9: Example of simulation results when N=400, r=1/2, I=5, g (P) = [7, 5] at Eb/N0 of 2.00dB.

CHAPTER 4

RESULT AND DISCUSSIONS

4.1 Performance of Decoding Algorithm

4.1.1 Frame Size

The simulations above have been carried out by varying the frame size. Frame size means the number of information bits including tail bits. Using larger frame size means having more number of bits. The total number of bits will be shuffled by the interleaver in the encoder to reduce the correlation between adjacent bits therefore give better performance efficiency.Hence the decoder gives better performance. The simulation results verified this conclusion. However, since Turbo code is a block code, it causes a time delay before getting the complete decoding output. Increasing the frame size also increases the delay time. We fixed some parameter such as generator polynomial g (D) = [7, 5], a punctured turbo code at rate R = 1/2, 5 iteration for this simulations in order to explore the performance of turbo code in term of frame size.

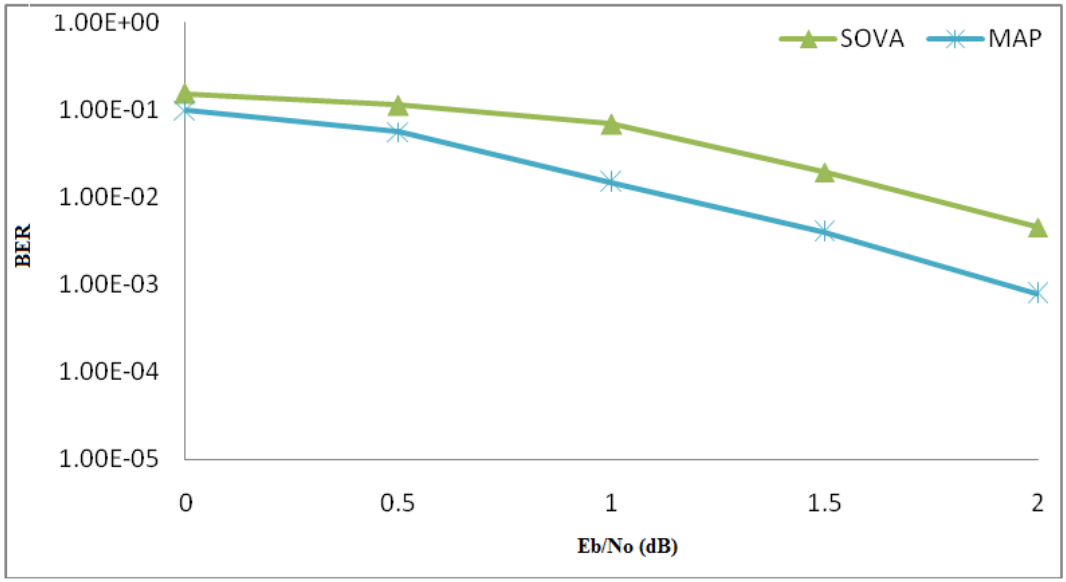

Figure 4.12: Simulation results for frame size N=500 bits for Log-MAP and SOVA decoding algorithm in AWGN channel

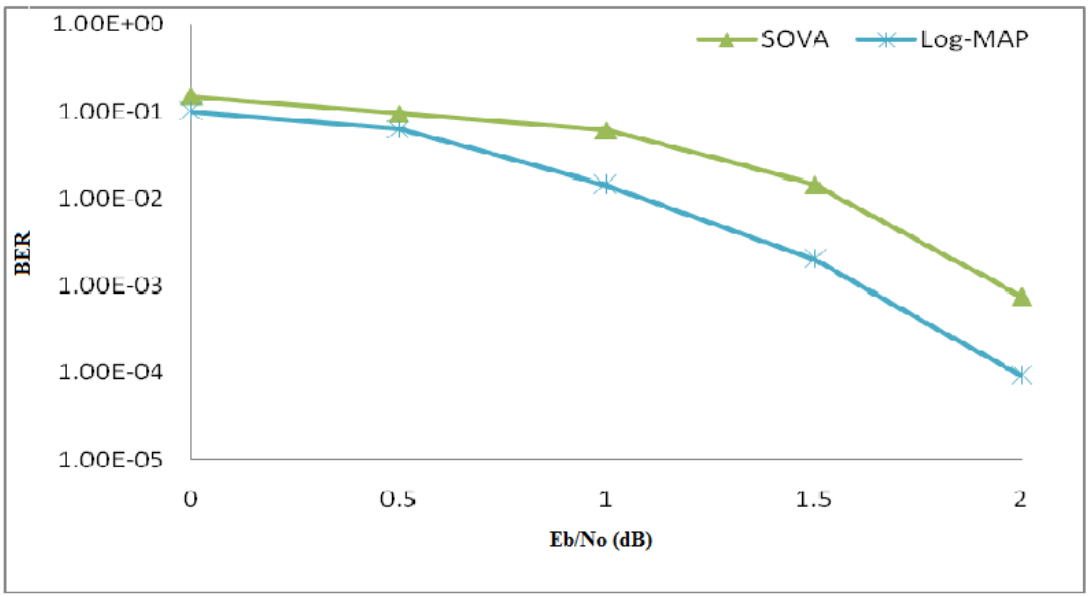

Figure 4.2: Simulation results for frame size N=1000 bits for Log-MAP and SOVA decoding algorithm in AWGN channel

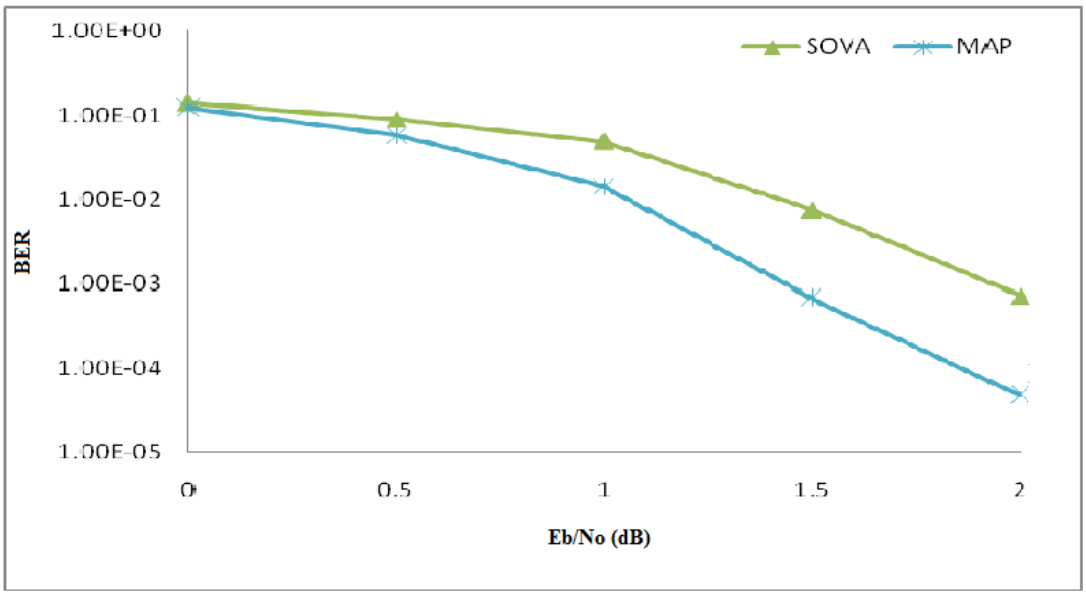

Figure 4.3: Simulation results for frame size N=1500 bits for Log-MAP and SOVA decoding algorithm in AWGN channel

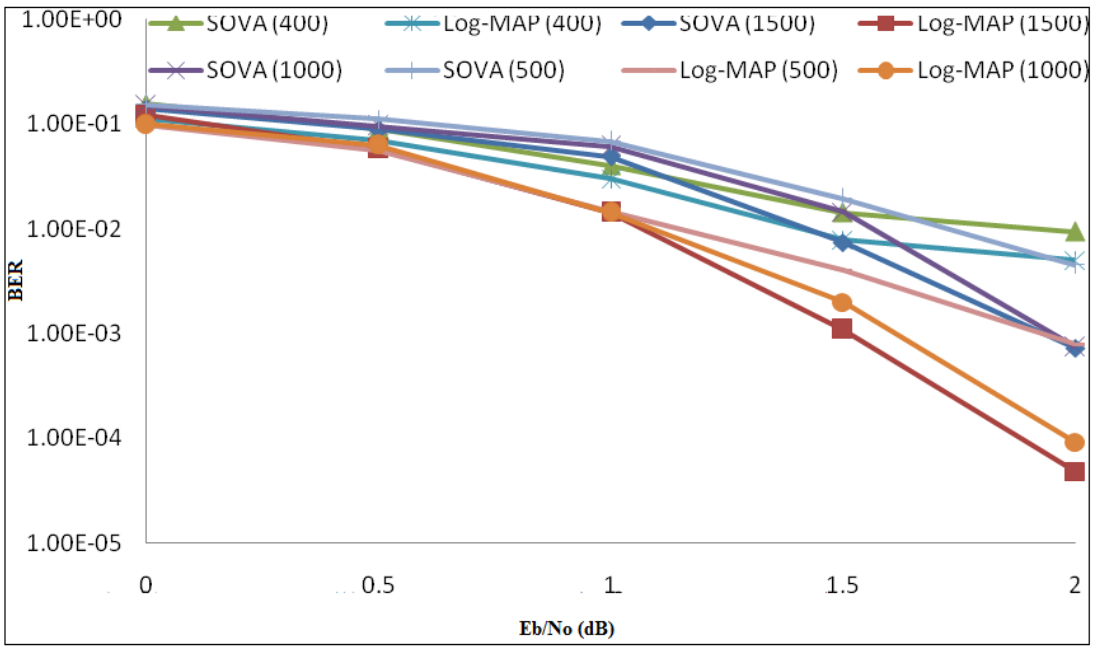

Figure 4.4: Simulation results of various frame size for Log-MAP and SOVA decoding algorithm in AWGN channel

As we observe from the graph above, we can clearly see there are three different frame size simulations which is N=500, 1000 and 1500 simulated with a standard value of generator polynomial g (D) = [7, 5] in octal representation with code rate, r =1/2 and a maximum iteration of 5 decoded with Log MAP and SOVA

algorithm each. At increased frame size, the performance of a turbo code improves substantially with the largest frame size with N=1500 which has the most number of bits has the lowest BER value compared to the other two frame size. From figure above, we can see that the Turbo code with larger frame size has better performance but decoding processing time and complexity increase.

In these simulations, Log-MAP algorithm for all the three frame size N= 500,1000 and 1500 performs better compared to SOVA where their coding gain is approximately around 0.6 dB to 0.4dB at BER value of 10^{-2}. Frame size affects the size of the interleaver hence increases the complexity and even though Log-MAP has higher computational complexity compared to SOVA, it still produces optimum performance despite longer decoding time during larger frame size.

4.1.2 Code Rate

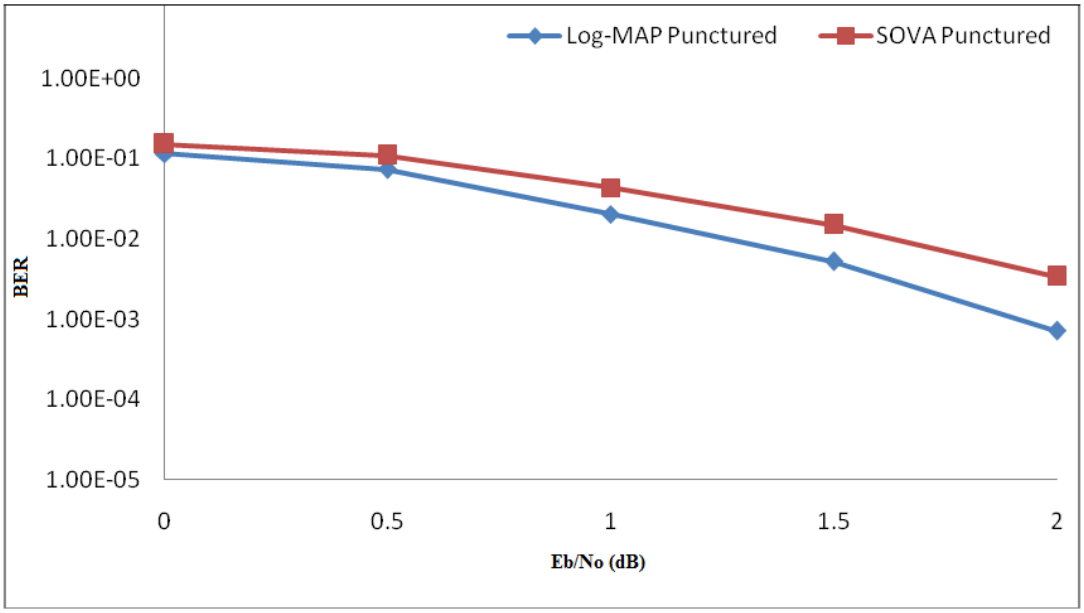

Figure 4.5: Simulation results for code rate, *r* = 1/2for Log-MAP and SOVA decoding algorithm in AWGN channel

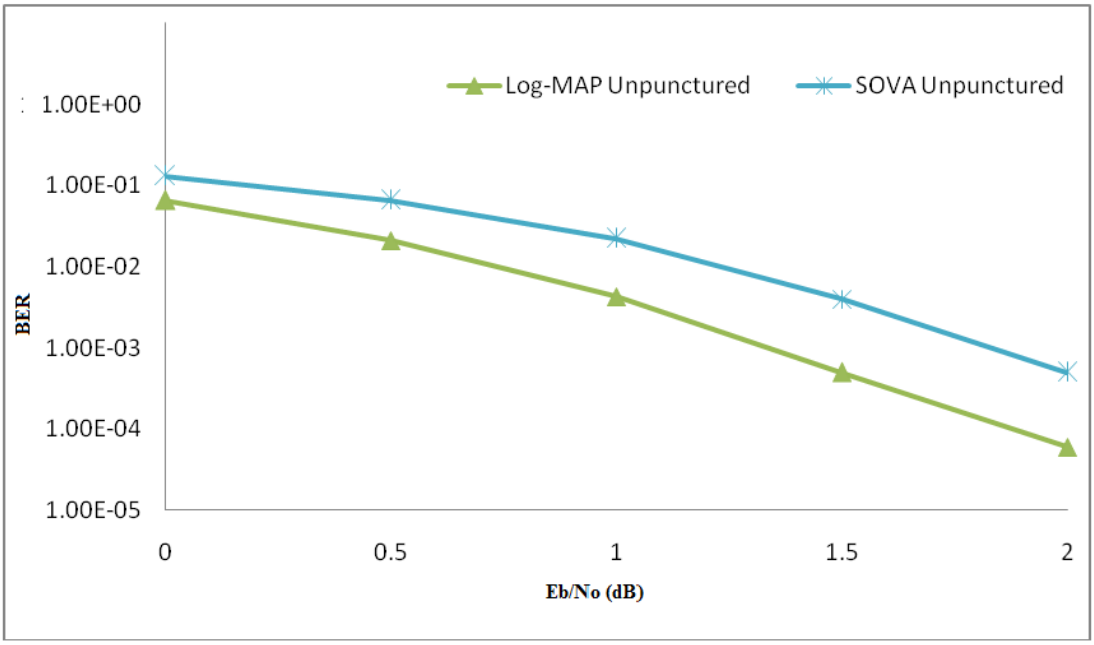

Figure 4.6: Simulation results for code rate, *r* = 1/3for Log-MAP and SOVA decoding algorithm in AWGN channel

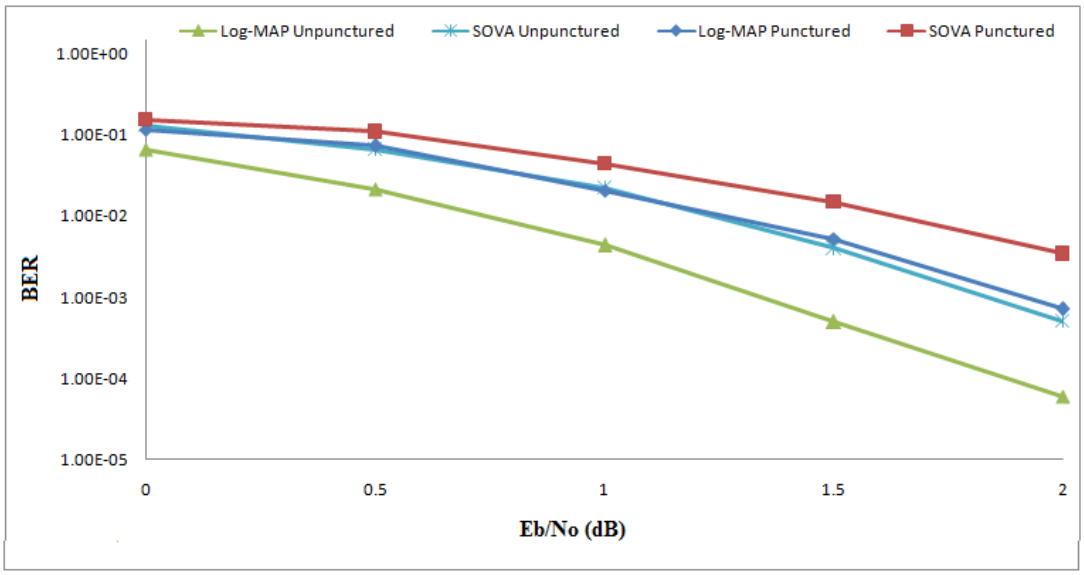

Figure 4.73: Simulation results of various code rate for Log-MAP and SOVA decoding algorithm in AWGN channel

As we observe from the figure above, two different simulations on the effects puncturing is carried out by varying its code rate value over different decoding algorithm. Puncturing adjusts the number of bits so that the width of the encoded bit sequence is constant and easy to handle. The code rate r =1/2, also known as half rate codes, is a punctured code where bits in the encoder is deleted thus decreases the

code rate. Third rate codes or r =1/3, is the un-punctured codes where puncturing does not take place in the encoder therefore no bits are deleted and code rate is increased.

We can clearly see the difference on the performance between half rate codes, r =1/2 and third rate codes, r = 1/3 simulated based on a standard parameter value with a maximum iteration of 5, together with frame size of N=400 and generator polynomial g (D) = [7, 5]. The simulations were conducted for both Log-MAP and SOVA algorithm in AWGN channel condition.

The third rate codes, r =1/3 performs better than the half rate codes, r =1/2, where third rate codes reaches a lower BER value compared to the half rate codes. The half rate codes might have lost some information during the puncturing process and decreases the bandwidth requirements which eventually results in performance degradation compared to the third rate codes where full parity information is sent exclusive of any puncturing.

For both code rate r = 1/2, 1/3 it is shown that Log-Map has a lower BER compared to SOVA with a coding gain between 0.3 to 0.35dB at BER value 10^{-2}. As a result, from here we can see that the higher the code rate, the lower the BER, hence the better the performance and in both variation of code rate, Log-MAP algorithm performs better compared to SOVA algorithm, although might have slight delay in time due to computational complexity in Log-MAP algorithm.

4.1.3 Generator Polynomial

The generator sequence and can be equivalently represented in a more compact form as g (D) = [g1, g2]. Where g2 denotes the feed forward output and g1 is the feedback to the input of the RSC encoder. The feedback from the RSC encoder output would results in major performance difference even if the same code generator was applied to both component codes. The rate of generator polynomials affects the puncturing pattern.

The simulations that has been carried out involves frame size of N=400 and a maximum iteration of 5, together with code rate of r = 1/3. Here two different generator polynomial value in octal representation which is g (D) = [7, 5] or [111, 101] and [15, 13] or [1111, 1101] in its respective octal representation is simulated using different type of decoding algorithms.

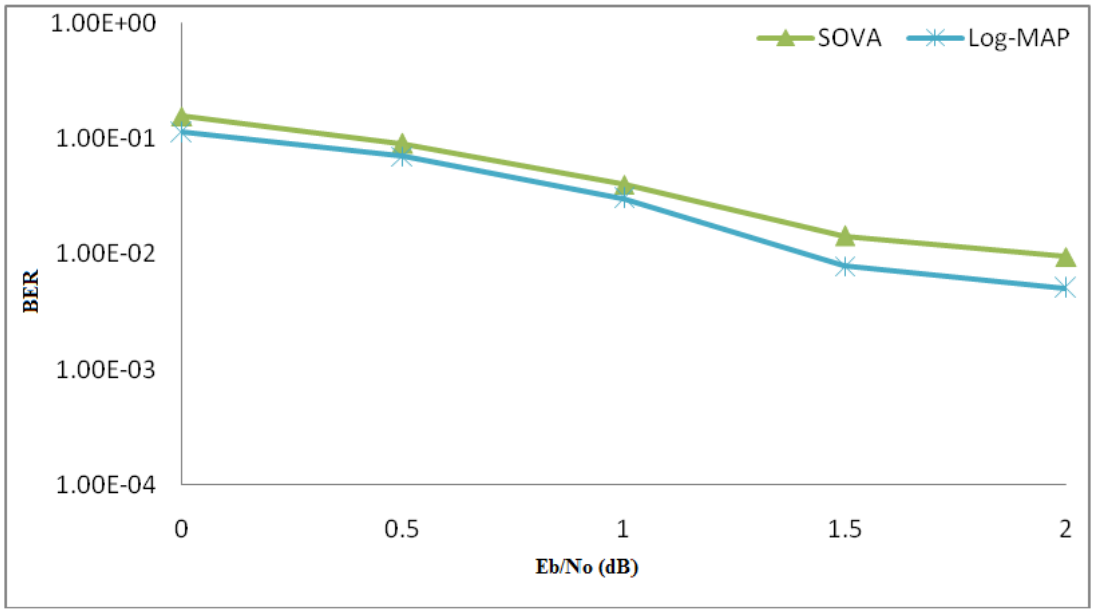

Figure 4.8: Simulation results for generator polynomial [7, 5] octal for Log-MAP and SOVA decoding algorithm in AWGN channel

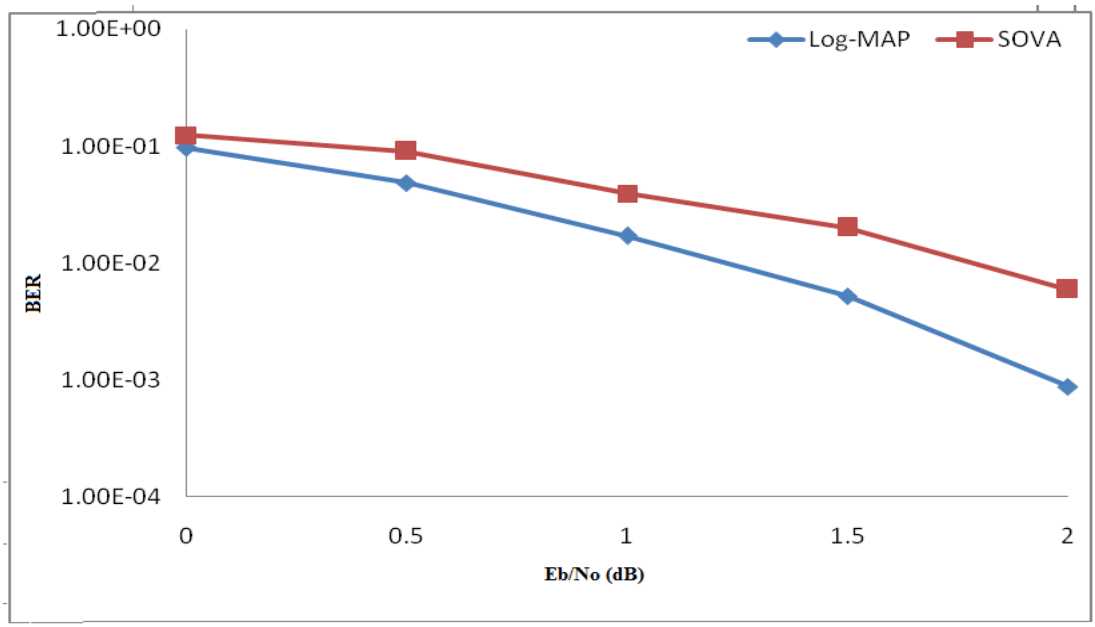

Figure 4.4: Simulation results for generator polynomial [15, 13] *octal* for Log-MAP and SOVA decoding algorithm in AWGN channel

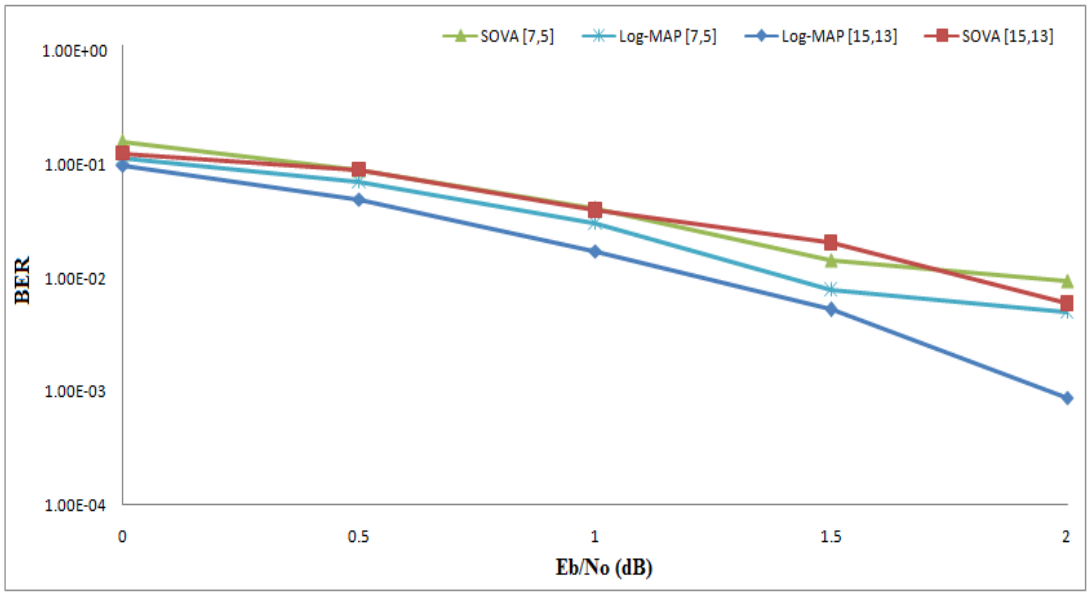

Figure 4.5: Simulation results of various generator polynomial for Log-MAP and SOVA decoding algorithm in AWGN channel

The combination with higher feedback and feed forward value g (D) = [15, 13] gives a better performance when compared to other combination using the same number of iterations besides giving optimum weight spectrum compared to the other. Log-MAP algorithm performs better compared to SOVA algorithm with a coding gain between 0.2dB to 0.25dBat BER value of 10^{-2}. This could be due to the forward backward recursion that takes place in Log-MAP compared with only forward recursion in SOVA algorithm.

4.1.4 Iterations

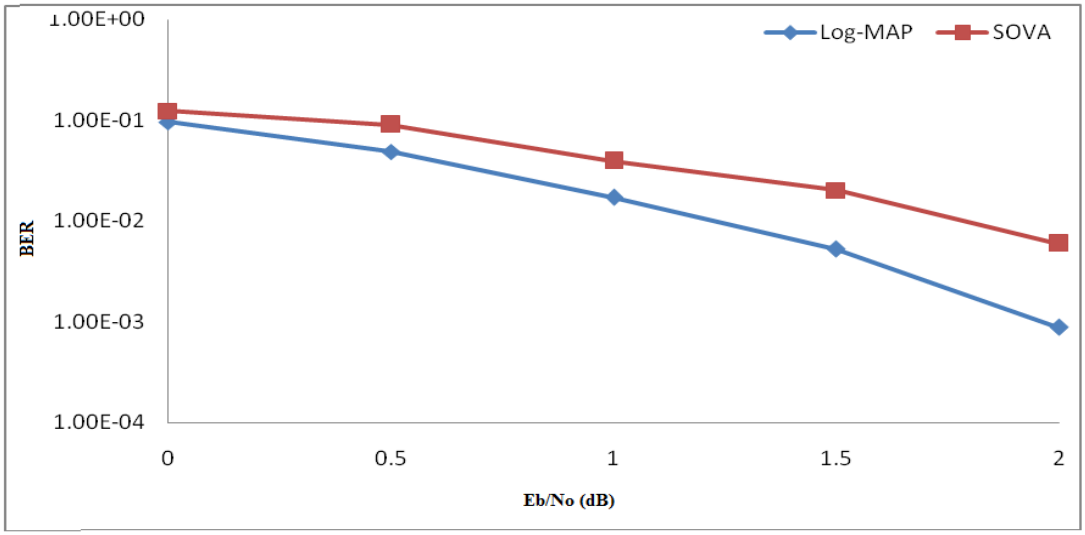

Figure 4.11: Simulation results for 5 iterations for Log-MAP and SOVA decoding algorithm in AWGN channel

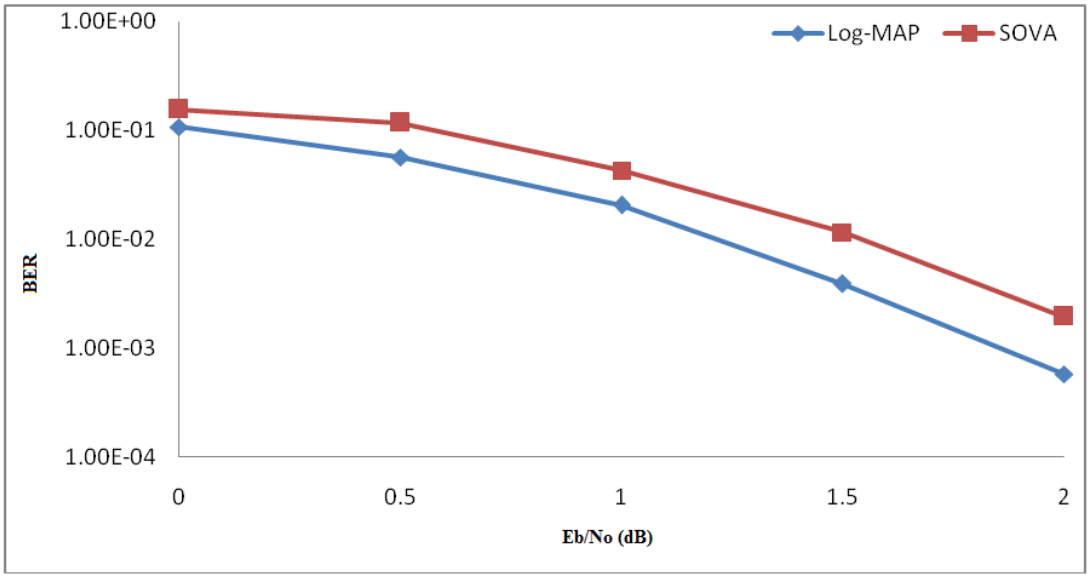

Figure 4.12: Simulation results for 7 iterations for Log-MAP and SOVA decoding algorithm in AWGN channel

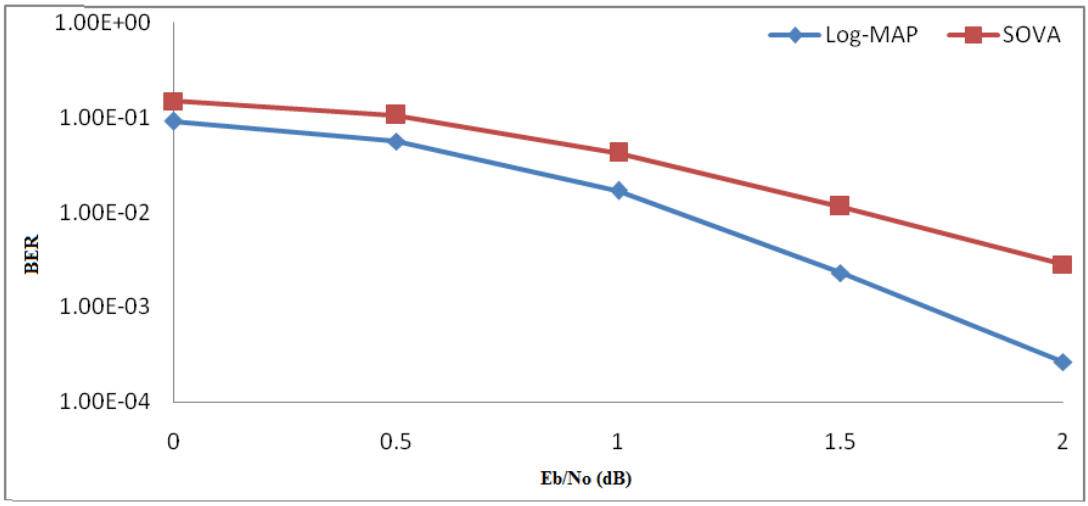

Figure 4.13: Simulation results for 9 iterations for Log-MAP and SOVA decoding algorithm in AWGN channel

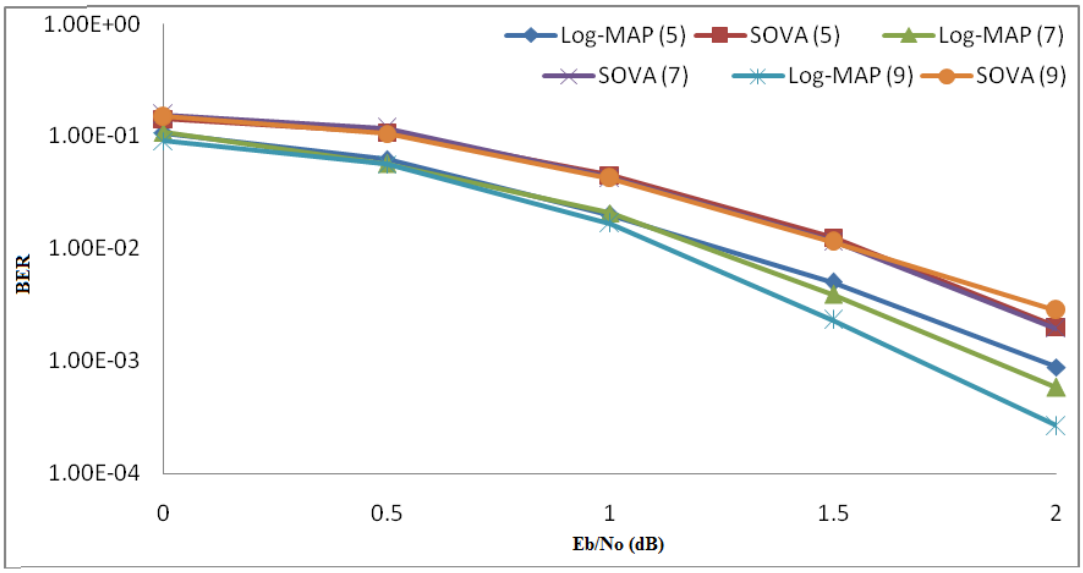

Figure 4.14: Simulation results of various iterations for Log-MAP and SOVA decoding algorithm in AWGN channel

As we observe from the graph above, the simulations varying the number of iterations has been conducted with a code rate =1/2, N=400, g(D)= [7,5] by varying the iterations I=5,7 and 9 for both Log MAP and SOVA algorithm.

From here we can see that the more number of iterations involved the lower the BER value. The performance of the iterations improves significantly from 5th iteration to 9th iteration. For each iteration, Log MAP has a better BER performance compared to SOVA for all the iterations with a coding gain between 0.26 dB to 0.30

dB. As we increase the number of iterations, the decoding complexity and latency also increases so for this simulation 9^{th} iteration gives satisfactory performance without excessive time delay. As we can see, if the simulation is continued most probably the 7^{th} and 9^{th} iteration will meet at a certain SNR value and will be saturated, improvement is not significant. As can be seen, the performance improves with the increased number of iterations. However, this leads to additional complexity and delay, which may cause glitches in applications involving real time data

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

From the error performance results, it is evident that turbo codes are quite suitable for the wireless communications applications under consideration with ahead mentioned requirements. In the system the encoder architecture is based on parallel concatenation through random interleaver of Recursive Systematic Convolutional encoders the decoder architecture is based on iterative cooperation between soft-input-soft-output decoders for the constituent codes. As we know now that the frame size, generator polynomial, code rate and also number of iterations affects the performance of turbo encoder and turbo decoder as a whole system and any changes in these parameters will affect the performance of the turbo codes.

After completing the simulations, it can be see that the frame size plays an important role in the turbo code system. When larger frame size is used, it means having more number of bits (information bits + tail bits). From the simulations it can be see that turbo code with frame size of 1500 bits gives better BER performance compared to the frame size 500 and 1000 bits, hence we can say that the larger the frame size better the performance.

Based on the simulations of different combination of generator polynomial, it can be clearly see that value of generator polynomial affects the performance in the turbo encoder. The combination with the higher feedback and feed forward value [15, 13] gives much better performance when compared to other combination using (Ioannis Chatzigeorgiou) the same number of iterations besides giving optimum weight spectrum compared to the other.

Besides that, different puncturing pattern also influence the performance of the generator polynomial. We emphasize that the puncturing pattern depends on the rate of generator polynomials rate hence different polynomials yield different puncturing patterns (Ioannis Chatzigeorgiou). (Hsian, May 2005)

Based on the simulations we can see that third rate codes are the output code which is fully transmitted from the encoder without puncturing and half rate codes are the codes in which their half parity bits from each of the component encoder are deleted to increase the system rate. The third rate codes can achieve coding gain of 0.30 dB to 0.35 dB at 10^{-2} BER. The third rate codes has a lower BER value compare to the half rate codes, and according to (Hsian, May 2005), this can be the due to the fact that full parity information is sent in third rate codes therefore the iterative decoder can best estimate the original message sequence from the received channel sequence. Thus we can conclude that third rate codes without puncturing is indeed better than half rate codes.

Iterations also play an important role in the effects of turbo decoding since one of the two main characteristic of turbo code decoder is iterative decoding besides SISO algorithm. Effect of varying the number of iterations during the decoding process is an interesting observation in system level studies. From the simulations conducted in AWGN channel, the number of iterations has been fixed. It can be see that the more number of iterations involved, the lower the BER value thus better performance in the system.

It can be clearly conclude now that Log-MAP is a much better decoding algorithm compared to SOVA but depends on the choice of complexity of the

decoding process. If a fast simulation need to be conducted but neglecting the BER performance, then SOVA is the right one for it but if a better BER performance needed to have more precise error correcting process then Log-MAP is the best choice despite the additional complexity and latency it has to deal with. That will be the major disadvantage that has to be dealt with in order to achieve the desired objectives. An improvement in the present decoding algorithms, utilizing hardware equipment with enhanced capabilities or the use of serial concatenated turbo codes are some of the options worth considering.

5.2 Future Work

In future we plan to investigate this new algorithmic technique that has been derived from an existing SOVA decoder and is known as Adaptive Soft Output Viterbi Architecture (ASOVA) turbo decoder. It is known to provide reduced computational complexity and a competitive bit error rate (BER) for decoders with the same operating parameters. This new adaptive SOVA (ASOVA) approach attempts to eliminate intermediate trellis paths during processing that are least likely to lead to the decoded output bit sequence.

Besides that, Low-density parity-check (LDPC) codes are a class of linear block LDPC codes. LDPC codes are also known as Gallager codes, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960 (Leiner, 2005). LDPC codes are already equipped with very fast (probabilistic) encoding and decoding algorithms. This makes LDPC codes not only attractive from a theoretical point of view, but also perfect for practical applications.

Reference

ADABI, I. (2011, august 23). *AFENDA DAILY INSIGHT NEWS*. Retrieved august 23, 2011, from AFENDA DAILY INSIGHT NEWS : http://www.agendadaily.com/Analisa/cabaran-konsep-1malaysia-mengikis-jurang-perkauman.html

Berrou, C. (2010). *Code and Turbo code.* France: Springer.

Berrou, C., Glavieux, A., & Thitimajshima, P. (1993). Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes. *IEEE International Conference on Communications (ICC)* , 1064-1070.

D.J.C. Mackay, R. (1997, march 6). Near Shannon limit performance of low density parity check codes.

Du, K.-L., & M.N.S.SWAMY. (2010). *Wireless Communication Systems: From RF Subsystems to 4G Enabling Technologies.* UK: CAmbridge University.

Glavieux, A. (2007). *Channel Coding in Communication Networks.* United States: ISTE.

Hsian, P. C. (May 2005). Iterative decoding of parallel and serial concatenated convolutional codes.

Ioannis Chatzigeorgiou, M. R. (n.d.). Pseudo- random Puncturing: A Technique to Lower the Error Floor of Turbo Codes.

Korhonrn, J. (2003). *Introduction to 3G Mobile Communication.* United States: Artech House.

Langton, C. (2006). Turbo Coding and MAP Decoding.

Leiner, B. M. (2005). LDPC Codes – a brief Tutorial.

Malarić, K. (2010). *EMI protection for Communication Systems.* MA canton street Norwood , MA: Artech House.

Proakis, J. (1995). *Digital Communications.* New york: McGraw-Hill.

Rekh, S., rani, D. S., & A.Shanmugam, D. (2000). Optimal Choice of Interleaver for Turbo codes. *IEEE* .

Scott, A. (2008). *RF Measurements for Cellular Phones and Wireless Data Systems.* canada: John Wiley &Sons.

Singal, T. (2010). *Wireless Communications.* New Delhi: Tata McGraw Hill.

Wu, Y. (n.d.). *google*. Retrieved from https://sites.google.com/site/bsnugroho/turbo

Yoon, S. (2009, march). *Turbo Code Distance Spectrum Calculator Version 1.2*. Retrieved from www,eccpage.com/tcds_readme.txt

Reference

APPENDIX A: TURBO CODE DEMO PROGRAM

```
% This script simulates the classical turbo encoding-decoding system.
% It simulates parallel concatenated convolutional codes.
% Two component rate 1/2 RSC (Recursive Systematic Convolutional) component encoders are assumed.
% First encoder is terminated with tails bits. (Info + tail) bits are scrambled and passed to
% the second encoder, while second encoder is left open without tail bits of itself.
%
% Random information bits are modulated into +1/-1, and transmitted through a AWGN channel.
% Interleavers are randomly generated for each frame.
%
% Log-MAP algorithm without quantization or approximation is used.
% By making use of ln(e^x+e^y) = max(x,y) + ln(1+e^(-abs(x-y))),
% the Log-MAP can be simplified with a look-up table for the correction function.
% If use approximation ln(e^x+e^y) = max(x,y), it becomes MAX-Log-MAP.
%
% Copyright Nov 1998, Yufei Wu
% MPRG lab, Virginia Tech.
% for academic use only

clear all

% Write display messages to a text file
diary turbo_logmap.txt

% Choose decoding algorithm
```

```
dec_alg = input(' Please enter the decoding algorithm. (0:Log-MAP, 1:SOVA) default 0    ');
if isempty(dec_alg)
  dec_alg = 0;
end

% Frame size
L_total = input(' Please enter the frame size (= info + tail, default: 500)   ');
if isempty(L_total)
  L_total = 500;          % infomation bits plus tail bits
end

% Code generator
g = input(' Please enter code generator: ( default: g = [1 1 1;1 0 1]      ');
if isempty(g)
  g = [1 1 1;1 0 1];
end
%g = [1 1 0 1; 1 1 1 1];
%g = [1 1 1 1 1; 1 0 0 0 1];

[n,K] = size(g);
m = K - 1;
nstates = 2^m;

%puncture = 0, puncturing into rate 1/2;
%puncture = 1, no puncturing
puncture = input(' Please choose punctured / unpunctured (0/1): default 1     ');
if isempty(puncture)
   puncture = 1;
end

% Code rate
rate = 1/(2+puncture);
```

```
% Fading amplitude; a=1 in AWGN channel
a = 1;

% Number of iterations
niter = input(' Please enter number of iterations for each frame: default 5       ');
if isempty(niter)
  niter = 5;
end
% Number of frame errors to count as a stop criterior
ferrlim = input(' Please enter number of frame errors to terminate: default 7       ');
if isempty(ferrlim)
  ferrlim = 7;
end

EbN0db = input(' Please enter Eb/N0 in dB : default [2.0]    ');
if isempty(EbN0db)
  EbN0db = [2.0];
end

fprintf('\n\n-----------------------------------------------------\n');
if dec_alg == 0
  fprintf(' === Log-MAP decoder === \n');
else
  fprintf(' === SOVA decoder === \n');
end
fprintf(' Frame size = %6d\n',L_total);
fprintf(' code generator: \n');
for i = 1:n
   for j = 1:K
      fprintf( '%6d', g(i,j));
   end
   fprintf('\n');
end
if puncture==0
```

```
   fprintf(' Punctured, code rate = 1/2 \n');
else
   fprintf(' Unpunctured, code rate = 1/3 \n');
end
fprintf(' iteration number =  %6d\n', niter);
fprintf(' terminate frame errors = %6d\n', ferrlim);
fprintf(' Eb / N0 (dB) = ');
for i = 1:length(EbN0db)
    fprintf('%10.2f',EbN0db(i));
end
fprintf('\n------------------------------------------------\n\n');

fprintf('+ + + + Please be patient. Wait a while to get the result. + + + +\n');

for nEN = 1:length(EbN0db)
   en = 10^(EbN0db(nEN)/10);      % convert Eb/N0 from unit db to normal numbers
   L_c = 4*a*en*rate;  % reliability value of the channel
   sigma = 1/sqrt(2*rate*en);   % standard deviation of AWGN noise

% Clear bit error counter and frame error counter
   errs(nEN,1:niter) = zeros(1,niter);
   nferr(nEN,1:niter) = zeros(1,niter);

   nframe = 0;    % clear counter of transmitted frames
   while nferr(nEN, niter)<ferrlim
      nframe = nframe + 1;
      x = round(rand(1, L_total-m));    % info. bits
      [temp, alpha] = sort(rand(1,L_total));        % random interleaver mapping
      en_output = encoderm( x, g, alpha, puncture ) ; % encoder output (+1/-1)

      r = en_output+sigma*randn(1,L_total*(2+puncture)); % received bits
      yk = demultiplex(r,alpha,puncture); % demultiplex to get input for decoder 1 and
2
```

```
% Scale the received bits
    rec_s = 0.5*L_c*yk;

% Initialize extrinsic information
    L_e(1:L_total) = zeros(1,L_total);

    for iter = 1:niter
% Decoder one
      L_a(alpha) = L_e;  % a priori info.
      if dec_alg == 0
        L_all = logmapo(rec_s(1,:), g, L_a, 1);  % complete info.
      else
        L_all = sova0(rec_s(1,:), g, L_a, 1);  % complete info.
      end
      L_e = L_all - 2*rec_s(1,1:2:2*L_total) - L_a;  % extrinsic info.

% Decoder two
      L_a = L_e(alpha);  % a priori info.
      if dec_alg == 0
        L_all = logmapo(rec_s(2,:), g, L_a, 2);  % complete info.
      else
        L_all = sova0(rec_s(2,:), g, L_a, 2);  % complete info.
      end
      L_e = L_all - 2*rec_s(2,1:2:2*L_total) - L_a;  % extrinsic info.

% Estimate the info. bits
      xhat(alpha) = (sign(L_all)+1)/2;

% Number of bit errors in current iteration
      err(iter) = length(find(xhat(1:L_total-m)~=x));
% Count frame errors for the current iteration
      if err(iter)>0
        nferr(nEN,iter) = nferr(nEN,iter)+1;
      end
```

```
    end%iter

% Total number of bit errors for all iterations
    errs(nEN,1:niter) = errs(nEN,1:niter) + err(1:niter);

    if rem(nframe,3)==0 | nferr(nEN, niter)==ferrlim
% Bit error rate
      ber(nEN,1:niter) = errs(nEN,1:niter)/nframe/(L_total-m);
% Frame error rate
      fer(nEN,1:niter) = nferr(nEN,1:niter)/nframe;

% Display intermediate results in process
      fprintf('**************  Eb/N0  =  %5.2f  db  **************\n',
EbN0db(nEN));
      fprintf('Frame size = %d, rate 1/%d. \n', L_total, 2+puncture);
      fprintf('%d frames transmitted, %d frames in error.\n', nframe, nferr(nEN,
niter));
      fprintf('Bit Error Rate (from iteration 1 to iteration %d):\n', niter);
      for i=1:niter
        fprintf('%8.4e    ', ber(nEN,i));
      end
      fprintf('\n');
      fprintf('Frame Error Rate (from iteration 1 to iteration %d):\n', niter);
      for i=1:niter
        fprintf('%8.4e    ', fer(nEN,i));
      end
      fprintf('\n');
      fprintf('************************************************\n\n');

% Save intermediate results
      save turbo_sys_demo EbN0db ber fer
    end

  end             %while
```

```
end                %nEN

diary off
```