Electrical and Electronics Department
Faculty of Engineering and Technology
Near East University, Lefkosa KKTC

EE400

Parallel Concatenated Convolutional Coding: Turbo Codes

By

Ejiofor Chidera Macrowland

(20124639)

Supervised By

Assist. Prof. Dr. Ali Serener

Submitted in partial fulfillment of the requirements of B.SC Degree in Electrical and Electronics Engineering

May 2015

Near East University
Yakın Doğu Üniversitesi, Lefkoşa KKTC

Electrical and Electronics Department
Faculty of Engineering and Technology
Near East University, Lefkosa KKTC

Parallel Concatenated Convolutional Coding: Turbo Codes

By

Ejiofor Chidera Macro land

(20124639)
Supervised By

Assist. Prof. Dr. Ali Serener

Submitted in partial fulfillment of the requirements of B.SC Degree in Electrical and Electronics Engineering

May 2015

ACKNOWLEDGEMENTS

First of all i would love to appreciate my supervisor Dr. Ali Serener for his constant support during this project and having the wonderful time to explain a lot to me.

Secondly i would love to thank my family for the support they gave me in doing this project and also my friends who supported me throughout the project by helping with some ideas and the time and space they also gave me.

Finally i would love to appreciate the school for this wonderful opportunity to embark on this project.

ABSTRACT

In parallel concatenated convolution coding or also known as turbo codes encoder we have the first encoder operating strictly on input information bits to produce first parity bits and through the use of an interleaver to other convolutional code encoders for producing second parity bits. The output of the encoders comprises of at least some of the information bits contained in each of the first and subsequent encoder that has been used in the process after correcting the errors in them. Basically the use of the system is to reduce the amount of errors in each of the information bits sent into the encoders by interleaving the input bits over a block length. The interleaver then interleaves the information bits in groups of N bits, where N is an integer that is should be greater than one. A parity bit generator can produce additional parity bits which are then worked on by the convolutional code encoders and interleavers and also a turbo decoder is present. The parallel concatenated codes, decoded through an iterative decoding algorithm of relatively low complexity, has been shown of recent to provide good coding gains close enough to the theoretical limits and has been used well i a wide variety of applications some which include deep space communications, third and fourth generation wireless standards, digital video broadcasting and a lot other benefits in the technological world. In this paper we shall characterize the contributions of the encoders, interleaver length and the constituent codes give to the overall performance of the parallel concatenated convolutional code (turbo codes).

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

RSC- Recursive Systematic Convolutional

RPSC- Recursive Punctured Systematic Convolutional

BCJR- Bahl Cocke Jelinek Raviv

AWGN- Additive White Gaussian Noise

PCCC- Parallel Concatenated Convolutional Code

CC- Convolutional Code

BER - Bit Error Rate

BPSK - Binary Phase Shift Keying

LLR - Log-likelihood ratio

Log MAP - Logarithmic Maximum a Posteriori Probability

MAP - Maximum a Posteriori Probability

ML -Maximum Likelihood path

SDVA - Soft Decision Viterbi Algorithm

SOVA - Soft Output Viterbi Algorithm

TC - Turbo Code

RS-Reed Solomon

SNR- Signal To Noise Ratio

FEC-Forward Error Correction

ARQ- Automatic Repeat Request

TBCC- Tail Biting Convolutional Codes

SHT- Shannon-Hartley Theorem

LDPC-Low Density Parity Codes

DVB-Digital Video Broadcasting

DVB-T - Digital Video Broadcasting Terrestrial

DVB-H - Digital Video Broadcasting Handheld

HD-TV-High Definition Television

SD-TV- Standard Definition Television

OFDM- Orthogonal Frequency Division Multiplex

DAB- Digital Audio Broadcasting

IBOC- In Band on Channel

R-Transmission Block Rate

VA - Viterbi Algorithm

a - Fading amplitude

α - Interleaver (!)

αt- Probability of being at node ! While moving in forward direction in the: trellis (!)

βt- Probability of being at node! while moving in backward direction in ::the trellis

Bc- Coherence bandwidth

c -Coded data

C- Channel capacity

CAWGN - Channel capacity for AWGN channel

CNCSI -Channel capacity for independent Rayleigh fading channel with no::: channel state information ( )t

Λ x -Log-likelihood ratio

Λ1 -A priori information for decoder 2

Λ2 - A priori information for decoder 1

Λ1e - Extrinsic information of decoder 1

Λ2e - Extrinsic information of decoder 2 0

Eb / N - Bit energy to noise power spectral density

(!', !) i t γ - Transitional probability from node!' to node ! While moving in :::forward direction in the trellis at time t due to either 0 or 1 as a data bit

(!, !') i t γ -Transitional probability from node ! to node !' while moving in :::backward direction in the trellis at time t due to either 0 or 1 as a data :::bit (∈)

Hb - Binary entropy function

I(X;Y) - Mutual information

i(n) -Interleaver index

Lc -Channel reliability

μt -Path metric at a node at time t

LIST OF FIGURES

Figures

CHAPTER 1: INTRODUCTION

The history of turbo codes, "at first it was a great surprise to observe that bit error rate (BER) of these reconstructed symbols was lower than that of decode information. We were unable to find any explanation for this literature". Claude Berrou and Alain Glavieux (1998). This section gives readers a collection of important materials along the turbo codes discovery that begins with a group of scientists which their work was based on the contemporary scheme of convolutional encoding and Viterbi algorithm decoding. Referring to the “Reflection on the Prize Paper: Near optimum error-correcting coding and decoding: turbo codes” published on June 1998 in IEEE information theory society newsletter, Claude Berrou, Alain Glavieux and Patrick Adde were mentioned as key people prior to the time of turbo codes invention. At the Ecole Nationale Sup´erieure Des Telecommunications de Bretagne of France, these scientists started their work focusing on the Soft Output Viterbi Algorithm (SOVA). It was based on the literature of G. Battail in 1987 and of J. Hagenauer and P. Hoeher in 1989. Those were certainly referred to famous papers of A.J. Viterbi, “Convolutional codes and their performance in communication systems”, and of G.D. Forney, “The Viterbi algorithm”. Initially, their research was to transfer the SOVA into hardware platform on MOS transistors in the simplest possible way as the target. Consequently, they observed that SOVA can be considered as a signal-to-noise (SNR) amplifier. This could be mentioned as the beginning of “turbo codes" concept because it stimulated them to consider “feedback” techniques that are commonly used with electronic amplifier circuits. To explore that concept, they cascaded that signal-to-noise (SNR) amplifier or their SOVA version in order to obtain large asymptotic gains.

This connection bases on “concatenation” coding technique of the well-known concept in the literature. Their experiments were done on a serial concatenation of two ordinary convolutional codes at the early step. It was later concentrated on parallel concatenation. Because the idea of two component decoders working with the same clock signal matches with that the reason of hardware implementation (in parallel) for clock signal distribution. This parallel concatenation with amplifiers was now considered to be

meaningful only if the code is systematic, and it was straightforward to use recursive systematic convolutional (RSC) codes at the final. During this period of turbo codes foundation, a PHD student, Punya Thiti-Majshima, started joining this group to work on the distance properties analysis in the year 1989. His dissertation devotes to studying distance properties and of error probability of the Recursive Punctured Systematic Convolutional (RPSC) codes and their concatenation in serial and parallel styles. Certainly, it is combined with iterative decoding. This work is entitled “Les codes Covolutifs Recursifs Systmatiques et leur application la concatenation parallel”, as a dissertation at l’Universit de Bretagne Occidentale (UBO). Gradually, the construction of original turbo codes was formed with related technical blocks. In order to solve obstruction in those initial works which reported on weighing problems, the beginning of SOVA was then replaced by Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm at the end of the discovery. It was mentioned that the first experiment with this novel coding construction was run in 1991. With the founding of following well known technical terms of extrinsic information, iterative decoding, recursive systematic convolutional codes, parallel concatenation, and non-regular interleaving, turbo codes was born finally.

There are two other main publications regarding turbo codes which appeared to the public after its introduction. First, a part of above dissertation was published in “Recursive Systematic Convolutional codes and application to parallel concatenation”, which was presented at IEEE Globecom in the 1995 conference by Thiti-Majshima. Moreover, at a year later another well-known article was published as “Near optimum error correcting coding and decoding: turbo-codes” on the IEEE transactions on communications. That was issued on October 1996 and written by Claude Berrou and Alain Glavieux. Since 1993, the legacy of turbo codes has always opened new technical research areas continuously. It sparks new numerous ideas to improve its own performance. Moreover, its concept is combined with other communication techniques in order to improve the overall systems performance .The source emits a block of information bits which is encoded and transmitted over a channel with Additive White Gaussian Noise. The Turbo decoder calculates an estimation of the information bits based on the noisy received signal.

1.1 Convolutional Code

In telecommunication, a convolutional code is a type of error-correcting code that generates parity symbols via the sliding application of a Boolean polynomial function to a data stream. The sliding application represents the 'convolution' of the encoder over the data, which gives rise to the term 'convolutional coding'. The sliding nature of the convolutional codes facilities trellis decoding using a time invariant trellis. Time invariant trellis decoding allows convolutional codes to be maximum likelihood soft decision decoded with reasonable complexity. The ability to perform economical maximum likelihood soft decision decoding is one of the major benefits of convolutional codes. This is in contrast to classic block codes which are generally represented by a time variant trellis and therefore are typically hard decision decoded. Convolutional codes are often characterized by the base code rate and the depth (or memory) of the encoder [n, k, K]. The base code rate is typically given as n/k, where n is the input data rate and k is the output symbol rate. The depth is often called the "constraint length" 'K', where the output is a function of the previous K-1 inputs. The depth may also be given as the number of memory elements 'v' in the polynomial or the maximum possible number of states of the encoder (typically 2^v).Convolutional codes are often described as continuous. However, it may also be said that convolutional codes have arbitrary block length, rather than that they are continuous, since most real world convolutional encoding is performed on blocks of data. Convolution-ally encoded block codes typically employ termination .The arbitrary block length of convolutional codes can also be contrasted to classic block codes, which generally have fixed block lengths that are determined by algebraic properties. The code rate of a convolutional code is commonly modified via symbol puncturing. For example, a convolutional code with a 'mother' code rate n/k=1/2 may be punctured to a higher rate of, for example, 7/8 simply by not transmitting a portion of code symbols. The performance of a punctured convolutional code generally scales well with the amount of parity transmitted.

The ability to perform economical soft decision decoding on convolutional codes, as well as the block length and code rate flexibility of convolutional codes, makes them very popular for digital communications.

1.2 USES OF CONVOLUTIONAL CODES

Convolutional codes are used broadly in many applications in order to gain reliable data transfer, including Digital Video Broadcasting, Radio Broadcasting, Mobile Phones and Wireless Communication, Satellite and Deep Space Communications. These codes are often implemented in concatenation with a hard-decision code, particularly Reed Solomon. Prior to turbo codes, such constructions were the most effective, coming closest to the Shannon limit based on the Shannon Hartley theorem (which tells us basically the amount of information a channel can carry. In other words it specifies the capacity of the channel. The theorem can be stated in simple terms as follows given communication system has a maximum rate of information (C) known as the channel capacity. If the transmission information rate (R) is less than (C), then the data transmission in the presence of noise can be made to happen with arbitrarily small error probabilities by using intelligent coding techniques. To get lower error probabilities, the encoder has to work on longer blocks of signal data. This entails longer delays and higher computational requirements. The Shannon Hartley theorem indicates that with sufficiently advanced coding techniques, transmission that nears the maximum channel capacity is possible with arbitrarily small errors. One can intuitively reason that, for a given communication system, as the information rate increases, the number of errors per second will also increase) we would see the details in later chapters of this paper.

1.2.1 DIGITAL VIDEO BROADCASTING

In digital video broadcasting, the aspect of terrestrial broadcasting has brought entertainment and media information to mass audiences around the world for nearly a century. In the last two decades, the demand for consumption of multimedia services anywhere anytime has increased greatly. Because the spectrum resource is limited, a new spectrum efficient broadcasting technology is required to meet this demand. Digital broadcast technologies, such as the European Digital Video Broadcast -Terrestrial (DVB-T), moved terrestrial broadcast into the digital age. While analog broadcast can only deliver one program me per channel, digital broadcast namely DVB-T allows multi-programme broadcasting where 2 to 4 standard definition TV (SDTV) programme can be transmitted with time division multiplexing in a single 8 MHz channel. The bandwidth of DVB-T (from 12 to 24 Mbit/s) can be allocated to offer different TV qualities, such as enhanced definition television (EDTV, requiring about 10 to 12 Mbit/s per programme) or high definition TV (HDTV, requiring about 24 Mbit/s per programme). In a nutshell, DVB-T has brought a higher quality service to TV program and it helps by reducing the use of spectrum, thus allowing other promising wireless technologies to diversity its applications. This leads to the development of emergence standard, Digital Video Broadcast - Handheld (DVB-H), that takes DVB-T a step further by making mobile reception of digital broad- casting possible with small, handheld devices. The governments of different European countries have made their plans for analog to digital TV switchover and fully deploying DVB-T/H services. United Kingdom, in early 2007, switched off analog TV in one Welsh community as an experiment. It is expected that the analog TV is completely phased out by the end of 2012 nationwide. Meanwhile, analog and digital TV continue to co-exist in such a way to

enable a soft transition from analog to digital. This period which is known as the simulcast phase will entail increased spectrum needs and this whole idea was made possible in the first place by the use of convolutional coding. Digital video broadcasting has a its physical layer which includes outer coder/decoder, inner coder/decoder, QAM mapping, Frame adaptation and TPS insertion, OFDM and up/down converter and other various steps . This is seen in a basic block diagram of DVB system below.

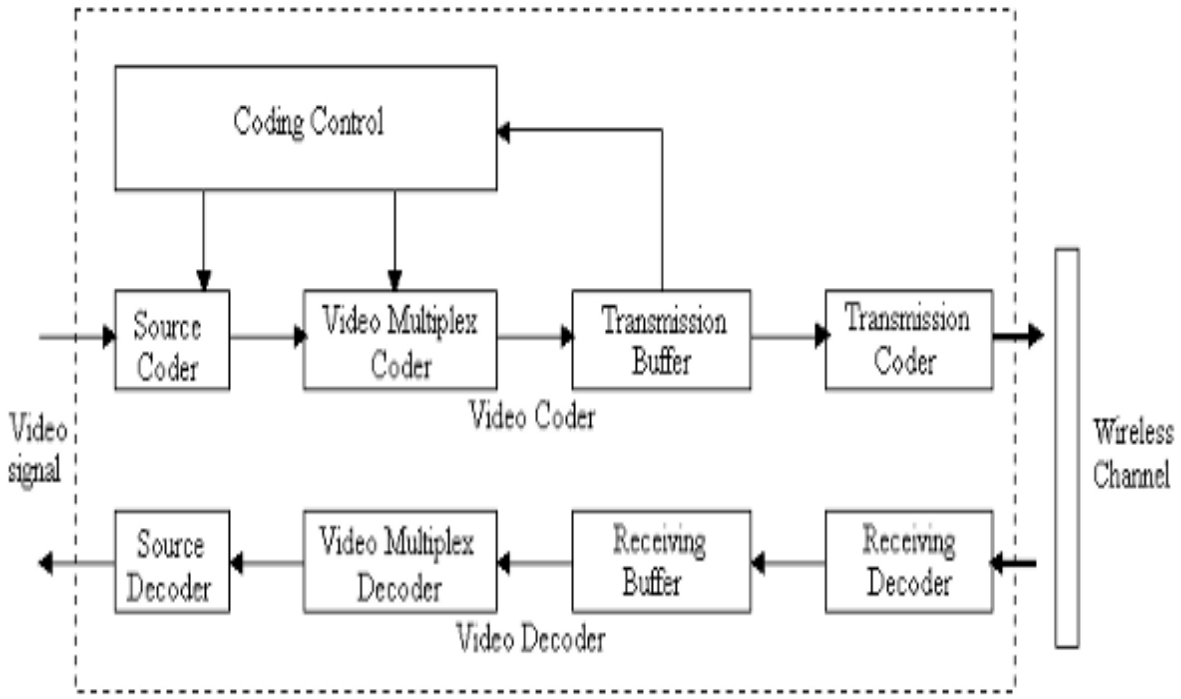


Figure 1: Block diagram of DVB system

1.2.2 RADIO BROADCASTING (DIGITAL AUDIO BROADCASTING)

In this section, this invention relates to radio broadcasting, and more particularly, to forward error correction in FM In-Band-On-Channel (IBOC) Digital Audio Broadcasting (DAB) and broadcasting systems utilizing such forward error correction. Digital Audio Broadcasting is a medium for providing digital quality audio, superior to existing analog broadcasting formats. Both AM and FM IBOC DAB can be transmitted in a hybrid format where the digitally modulated signal coexists with the currently broadcast analog signal. IBOC requires no new spectral allocations because each DAB signal is simultaneously transmitted within the same spectral mask of an existing channel allocation. IBOC promotes economy of spectrum while enabling broadcasters to supply digital quality audio to their present base of listeners. FM IBOC broadcasting systems uses a hybrid modulation format. An Orthogonal Frequency Division Multiplex (OFDM) technique has been described for IBOC DAB. OFDM signals consist of orthogonally spaced carriers all modulated at a common symbol rate. The frequency spacing for rectangular pulse symbols (e.g., BPSK, QPSK, 8PSK or QAM) is equal to the symbol rate. For IBOC transmission of FM/DAB signals, a redundant set of OFDM sub-carriers is placed within about 100 kHz to 200 kHz on either side of a coexisting analog FM carrier. The DAB power (upper or lower sideband) is set to about -25 dB relative to the FM signal. The level and spectral occupancy of the DAB signal is set to limit interference to its FM host while providing adequate signal-to-noise ratio (SNR) for the DAB sub-carriers. First adjacent signals spaced at +-200

kHz from the FM carrier can corrupt the DAB signal. However, at any particular location within a station's coverage area, it is unlikely that both first adjacents will significantly interfere with DAB. Therefore the upper and lower DAB sidebands carry the same redundant information such that only one sideband is needed to communicate the information. Inherent advantages of OFDM include robustness in the presence of multipath interference, and tolerance to non-Gaussian short term noise or notches due to selective fading. Forward error correction (FEC) and interleaving improve the reliability of the transmitted digital information over a corrupted channel. Complementary Pair Convolution (CPC) FEC code techniques were developed for Automatic Repeat Request (ARQ) schemes where retransmissions were coded using complementary codes instead of simply retransmitting the same coded sequence. CPC codes can be constructed according to previously published puncturing techniques like the "High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding ,"It is known that the periodic puncturing of bits from a convolutional code using Viterbi decoding is an effective means of creating higher rate convolutional codes. Rate compatible punctured convolutional (RCPC) codes have been conceived as a mechanism to adjust coding gain and bit energy as a function of channel capacity in a practical efficient manner. This is useful in a point-to-point (non-broadcast) automatic repeat request (ARQ) system where an the intended receiver assesses its signal to noise power ratio (Eb/No) and communicates its desire to the transmitter (via a return path) to increase or decrease energy per bit (Eb) and coding gain. The transmitter responds by adjusting its code rate R. This is accomplished with a punctured convolutional code where the transmission of all the bits typically employs an "industry standard" K=7, R=1/2 rate code, for example. It is assumed in this non punctured case that the maximum Eb and coding gain is achieved. To improve spectral and/or power efficiency, the transmitter may elect to eliminate (puncture at the receiver's request, for example) the transmission of some of the coded bits, resulting in a higher rate code. This puncturing has the effect of lowering the effective Eb and coding gain relative to the original unpunctured code; however, this punctured code may still be sufficient to successfully communicate the information over the channel in a more efficient manner .For best performance at a given code rate, a particular pattern of bits in the coded sequence is

punctured. Unfortunately, the puncture pattern for higher rate codes does not include all the bits punctured for lower rate codes. It shows that the puncture patterns for RCPC codes can include all punctures for lower rate codes with little loss compared to the optimal, but rate-incompatible, puncture patterns. Therefore the code rate can be increased from the original R=1/2 code simply by puncturing more of the puncturable bits of the same pattern. The higher rate codes are a subset of the bits of the lower rate codes.

The interference environment in VHF FM-band IBOC DAB channel is generally such that a DAB channel can be dichotomized into the following two subsets of sub channels: (a) a reliable part composed of regions of spectrum relatively free of interference from other stations' signals, characterized as being thermal or background noise limited, with multipath fading as an impairment; and (b) an unreliable part composed of regions of spectrum with intermittent intervals of heavy interference which corrupts the bits transmitted during those intervals, but is at other times (or for most geographical locations) similar to the reliable part described above. AM band IBOC DAB can be similarly characterized. The prior art utilizes one of two fundamental strategies to transmit data in this environment: (1) simply do not utilize the unreliable part of the channel, thus those times during which the unreliable part is clear and usable are essentially wasted; or (2) utilize a sufficiently low rate code (and appropriately increased coded bit rate) to guarantee the required bit error rate (BER), and spread the increased bandwidth across both the reliable and unreliable parts of the spectrum evenly. This is done by uniform allocation of bits to OFDM carriers in an OFDM system, or increasing the raw bit rate of a single carrier system. This utilizes the unreliable part of the channel, but also incurs a BER penalty (possibly catastrophic) when severe interference occurs in the unreliable part of the channel. Depending on the interference, the second alternative may or may not be better than the first. Below is an image of the basic block diagram of a DAB system

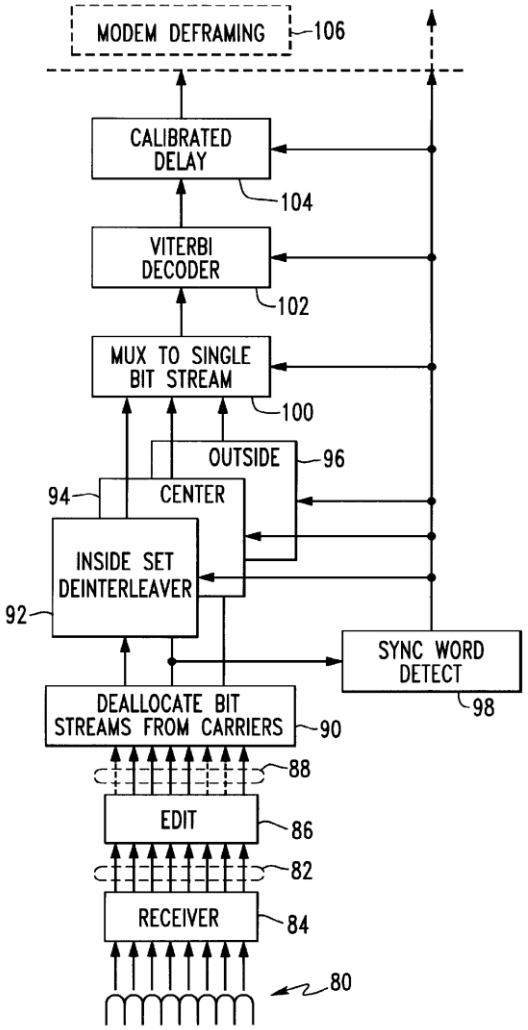


Figure 2: Simplified Functional Block Diagram of a Transmitter and Receiver Which Operate In Accordance With the Method of the Invention

1.2.3 SATELLITE AND WIRELESS COMMUNICATION (LTE)

In this section we shall see that convolutional codes are a major contribution in satellite and wireless communication used in mobile phones also known as Long Term Evolution (LTE) 4^{th} generation wireless communication and previous generations too(3rd generation.....). This involves the use of Tail Biting Convolutional Codes (TBCC) and turbo codes. TBCC which ensures that the starting state of the encoder is the same as it's ending state (and that this state value does not necessarily have to be the all-zero state). For a rate 1/n feed-forward encoder, this is achieved by initializing the m memory elements of the encoder with the last m information bits of a block of data of length L, and ignoring the output. All of the L bits are then input to the encoder and the resultant L*n output bits are used as the codeword.an example is Zero-Tailed Encoding, In comparison, the zero-tail termination method appends m zeros to a block of data to ensure the feed-forward encoder starts from and ends in the all-zero state for each block. This incurs a rate loss due to the extra tail bits (i.e. non-informational bits) that are transmitted. Tail-Biting Decoding is also involved in the convolutional coding done in LTE, The maximum likelihood tail-biting decoder involves determining the best path in the trellis under the constraint that it starts and ends in the same state. A way to implement this is to run M parallel Viterbi algorithms where M is the number of states in the trellis, and select the decoded bits based on the Viterbi algorithm that gives the best metric. However this makes the decoding M times more complex than that for zero-tailed encoding which is much simpler than the maximum likelihood approach and yet performs comparably. The scheme is based on the premise that the tail-biting trellis can be considered circular as it starts and ends in the same state. This allows the Viterbi algorithm to be continued past the end of a block by repeating the received code word circularly. As a result, the model repeats the received code word from the demodulator and runs this data set through the Viterbi decoder, performing the trace back from the best state at the end of the repeated data set. Only a portion of the decoded

bits from the middle are selected as the decoded message bits. The Operation mode parameter for the Viterbi Decoder block is set to be "Truncated" for the tail-biting case while it is set to "Terminated" for the zero-tailed case. While for the turbo codes used in LTE is based on two eight-state constituent encoders and one turbo code internal interleaver. The sequence of the original input bits (B) is encoded by the first encoder, the input to the second encoder is an interleaved version of the first sequence (B); therefore the resulting coded sequence is the result of combining the information sequence of bit (systematic bits, S) with two sequences of parity bits (p1 and p2).Hence the overall code rate is 1/3 even though the rate of each encoder is 1/2, because their outputs is combined with inputs data bits.

Finally, a trellis termination operation is performed to force the encoders back to a zero initial state after coding a transport block. Once all the information bits are encoded, the tail bits from the shift register feedback in each encoder are padded after the encoded information bits. The tail bits are used to terminate each other constituent encoder while the other constituent encoder is disabled (its correspondent switch in position 2).In other to reduce the complexity in both turbo encoder and decoder, a Quadratic Polynomial Permutation (QPP) interleaver is used instead of release-6 turbo interleaver (3GPP 36.212). This kind of interleaver (Takeshita 2006, 1249) holds the maximum contention free property. That is a great advantage from hard ware implementation perspective because contention in interleaver memory access is avoided and therefore it is possible to parallelize decoding with a single extrinsic memory. The reason for this interleaving process is that the performances of turbo and convolutional codes are improved when the errors introduced by the radio channel are statistically independent because they are random error correcting codes.

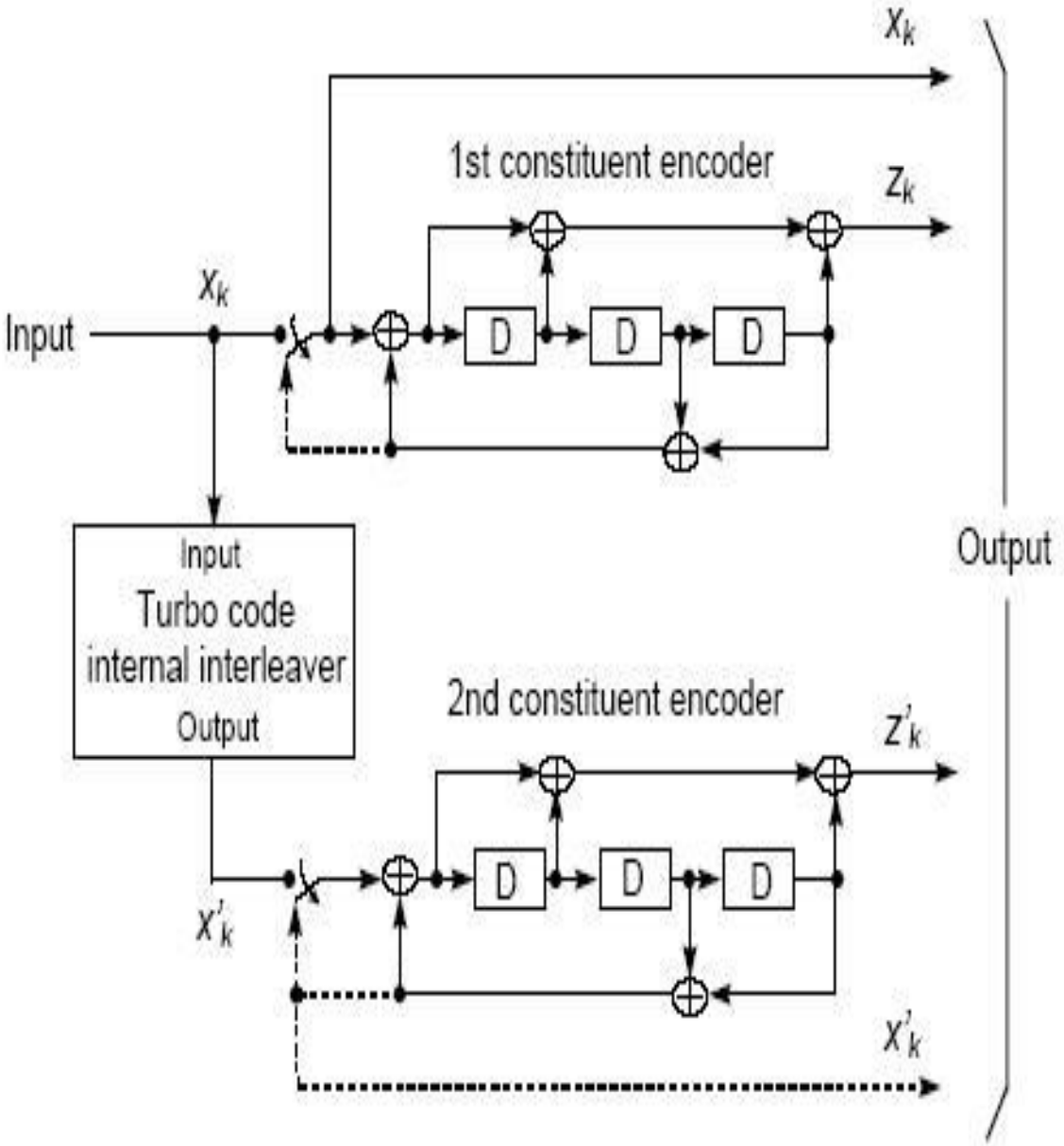


Figure 3: Turbo Code Coding Scheme for LTE

Tail-biting encoding in LTE As of 2014, many LTE supporting mobile phones and tablet phones are being released for sale to the public across the world. These shows below mentions of only some of the better known models: Apple; iPhone 5C, iPhone 5S, iPhone 6, Plus, iPad, iPad, iPad Air, iPad Air 2, iPad Mini (1st generation), iPad Mini 2, iPad Mini 3,Fujitsu; Arrows NX HTC; HTC Desire Eye, HTC One (2013), HTC One (M8), HTC One Mini, HTC One Mini 2 Huawei; Ascend D2 LTE. Ascend P7 Jolla; Jolla / MeeGo Sailfish OS LG; G Flex, G Pro, G2, G3, Gx, Isai VL, Nexus 5, Optimus LTE 2, Optimus LTE III. Optimus; Vu 2, Vu 3 Lenovo; Lenovo A6000 Motorola; Droid Turbo Moto G Nexus 6 Nokia; Lumia 830, Lumia 635, Lumia 640, Lumia 2520, OnePlus, One Pantech; Vega Iron, Vega N°6, Vega R3, Vega Secret Note Samsung; Galaxy A5, Galaxy Alpha, Galaxy Avant, Galaxy Core LTE, Galaxy Golden, Galaxy Grand, Galaxy Light, Galaxy Note 3, Galaxy Note 4, Galaxy Note 10.1 LTE, Galaxy Note Edge, Galaxy Note II LTE, Galaxy Pop, Galaxy Round, Galaxy S III LTE, Galaxy S4, Galaxy S4 Active, Galaxy S4 mini LTE, Galaxy S4 Zoom, Galaxy S5, Galaxy S5 Active, Galaxy S6, Galaxy S6 Edge, Galaxy Win Saygus; Saygus V2, Sharp; Aquos Pad, Aquos Zeta, Disney Mobile on docomo Sony; Xperia SP M35t, Xperia Z C6603, Xperia Z1, Xperia Z2, Xperia Z2 Tablet, Xperia Z3, Xperia Z3 Compact, Xperia Z4 Tablet LTE, Xperia ZL C6506, X-Systems, X-Tel 9500

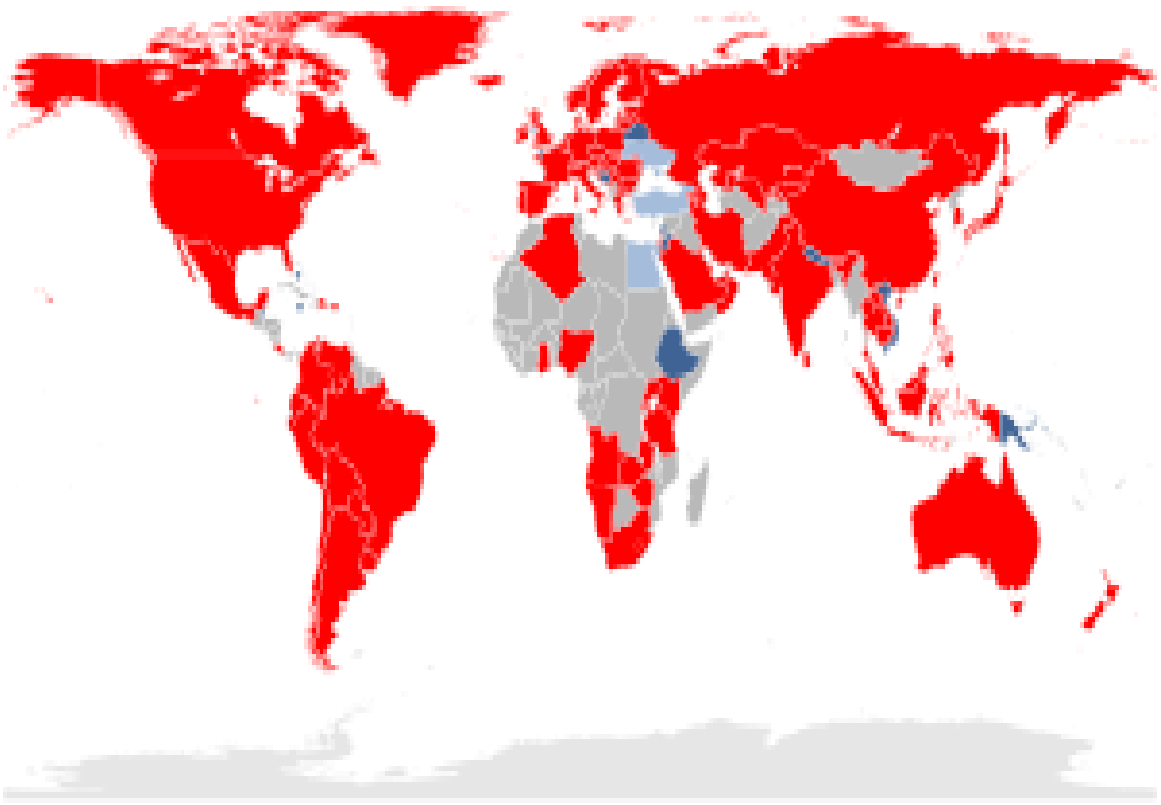


Figure 4: Adoption of LTE technology as of December 7, 2014

- Countries and regions with commercial LTE service
- Countries and regions with commercial LTE network deployment on-going or planned
- Countries and regions with LTE trial systems (pre-commitment)

CHAPTER 2: TURBO CODES ENCODING

In this section we shall take a deep and detailed look at turbo codes full and the various processes involved with it.From the encoding of the turbo codes to the iterative decoding of the codes and the interleaver used in separating them. Basically they are error-correcting codes with performance close to the Shannon theoretical limit [SHA]. These codes have been invented at ENST Bretagne (now TELECOM Bretagne) as explained in early chapters, in France, in the beginning of the 90's [BER]. The encoder is formed by the parallel concatenation of two convolutional codes separated by an interleaver or permuter. An iterative process through the two corresponding decoders is used to decode the data received from the channel. Each elementary decoder passes to the other soft (probabilistic) information about each bit of the sequence to decode. This soft information, called extrinsic information and is updated at each iteration. A basic block diagram of turbo codes process is shown below.

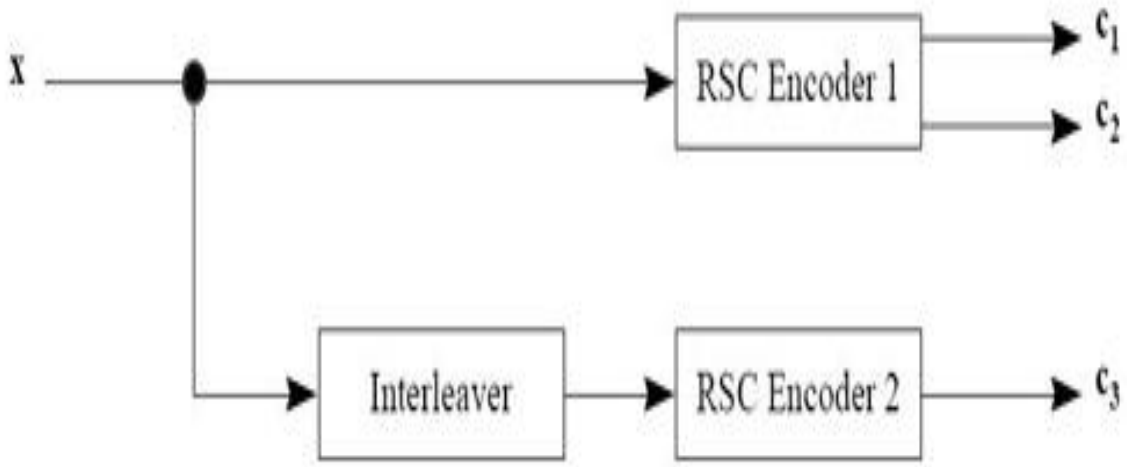


Figure 5: A Block Diagram of a Typical Turbo Code System

2.1 TURBO ENCODING

The basic idea of turbo codes is to use two convolutional codes in parallel with some kind of interleaving in between. Convolutional codes can be used to encode a continuous stream of data, but in this case we assume that data is configured in finite blocks corresponding to the interleaver size. The frames can be terminated - i.e. the encoders are forced to a known state after the information block. The termination tail is then appended to the encoded information and used in the decoder. The system is illustrated in Figure below.

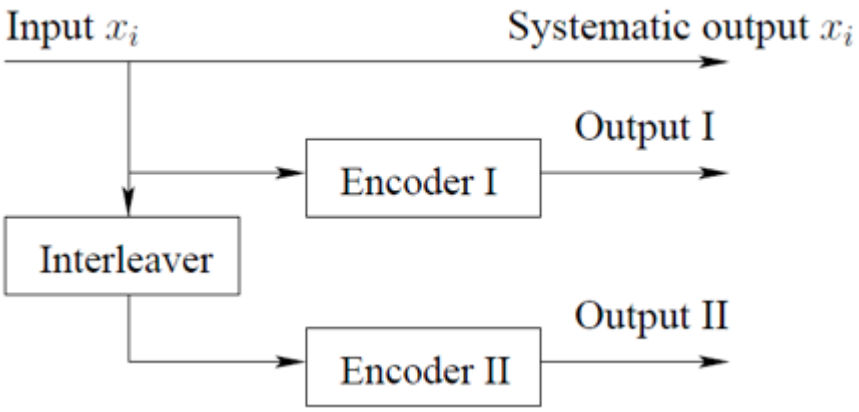


Figure 6: Turbo Encoder

We can regard the turbo code as a large block code. The performance depends on the weight distribution not only the minimum distance but the number of words with low weight. Therefore, we want input patterns giving low weight words from the first encoder to be interleaved to patterns giving words with high weight for the second encoder. Convolutional codes have usually been encoded in their feed-forward form, like (G1, G2) = (1+D^2, 1+D+D^2). However, for these codes a single 1, i.e. the sequence ...0001000..., will give a code word which is exactly the generator vectors and the weight of this code

word will in general be very low. It is clear that a single 1 will propagate through any interleaver as a single 1, so the conclusion is that if we use the codes in the feed-forward form in the turbo scheme the resulting code will have a large number of code words with very low weight.

The trick is to use the codes in their recursive systematic form where we divide with one of the generator vectors. Our example gives (1, G2/G1) = (1, (1+D+D^2)/(1+D^2)). This operation does not change the set of encoded sequences, but the mapping of input sequences to output sequences is different. We say that the code is the same, meaning that the distance properties are unchanged, but the encoding is different.

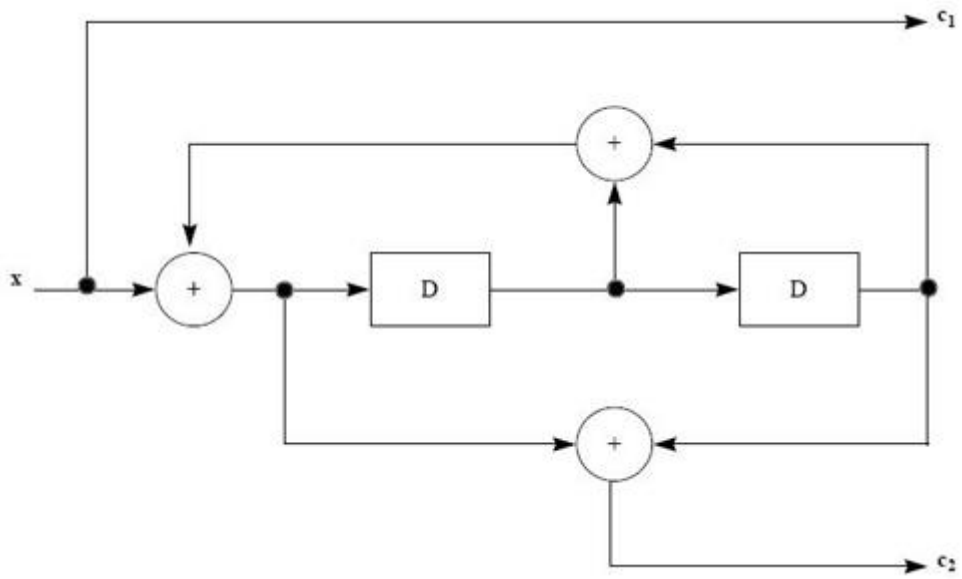


Figure 7: Recursive systematic encoder

In Figure 7 we have shown an encoder on the recursive systematic form. The output sequence we got from the feed-forward encoder with a single 1 is now obtained with the input 1+D^2=G1. More important is the fact that a single 1 gives a code word of semi-infinite weight, so with the recursive systematic encoders we may have a chance to find an interleaver where information patterns giving low weight words from the first encoder are interleaved to patterns giving words with high weight from the second encoder.

The most critical input patterns are now patterns of weight 2. For the example code the information sequence ...01010...will give an output of weight 5. Notice that the fact that the codes are systematic is just a coincidence, although it turns out to be very convenient for several reasons. One of these is that the bit error rate (BER) after decoding of a systematic code cannot exceed the BER on the channel. Imagine that the received parity symbols were completely random, then the decoder would of course stick to the received version of the information. If the parity symbols at least make some sense we would gain information on the average and the BER after decoding will be below the BER on the channel. One thing is important concerning the systematic property, though. If we transmit the systematic part from both encoders, this would just be a repetition, and we know that we can construct better codes than repetition codes. The information part should only be transmitted from one of the constituent codes, so if we use constituent codes with rate 1/2 the final rate of the turbo code becomes 1/3. If more redundancy is needed, we must select constituent codes with lower rates. Likewise we can use puncturing after the constituent encoders to increase the rate of the turbo codes.

Puncturing is used to delete one or more coded parity bits from a code word in a code. This well-known technique helps to obtain high-rate codes without modifying the structure of the encoder and the decoder circuit for an existing encoder/decoder circuit. Although punctured codes makes a system flexible by allowing the change in the code rate they usually suffer from performance degradation as compared to the unpunctured codes. Now comes the question of the interleaving. A first choice would be a simple block interleaver, i.e. to write by row and read by column. However, two input words of low weight would give some very unfortunate patterns in this interleaver. The pattern is shown in Figure 8 for our example code. We see that this is exactly two times the critical two-input word for the horizontal encoder and two times the critical two-input pattern for the vertical

. . . . . .

. . . 0 0 0 0 0 . . .

. . . 0 1 0 1 0 . . .

. . . 0 0 0 0 0 . . .

. . . 0 1 0 1 0 . . .

. . . 0 0 0 0 0 . . .

. . . . . .

Figure 8: Critical pattern in block interleaver

encoder as well. The result is a code word of low weight (16 for the example code) not the lowest possible, but since the pattern appears at every position in the interleaver we would have a large number of these words. This time the trick is to use a pseudo-random interleaver, i.e. to read the information bits to the second encoder in a random (but fixed) order. The pattern from Figure 8 may still appear, but not nearly as often. On the other hand we now have the possibility that a critical two-input pattern is interleaved to another critical two-input pattern. The probability that a specific two-input pattern is interleaved to another (or the same) specific two-input pattern is 2/N, where N is the size of the interleaver. Since the first pattern could appear at any of the N positions in the block, we must expect this unfortunate match to appear 2 times in a pseudo-random interleaver of any length. Still the pseudo random interleaver is superior to the block interleaver, and the pseudo-random interleaving is standard for the turbo codes. It is possible to find interleavers that are slightly better than the pseudo-random ones, some papers on this topic are included in the literature list. We will end this section by showing a more detailed drawing of a turbo encoder, Figure 9. Here we see the two recursive systematic encoders, this time for the code (1, (1+D4)/ (1+D+D^2+D^3+D^4)). Notice that the systematic bit is removed from one of them. At the input of the constituent encoders we see a switch. This is used to force the encoders to the all-zero state i.e. to terminate the trellis. The complete incoming frame is kept in a buffer from where it is read out with two different sets of addresses - one for the original sequence and one for the interleaved one. This way output 1 and output 2 correspond to the same frame and can be merged before transmission.

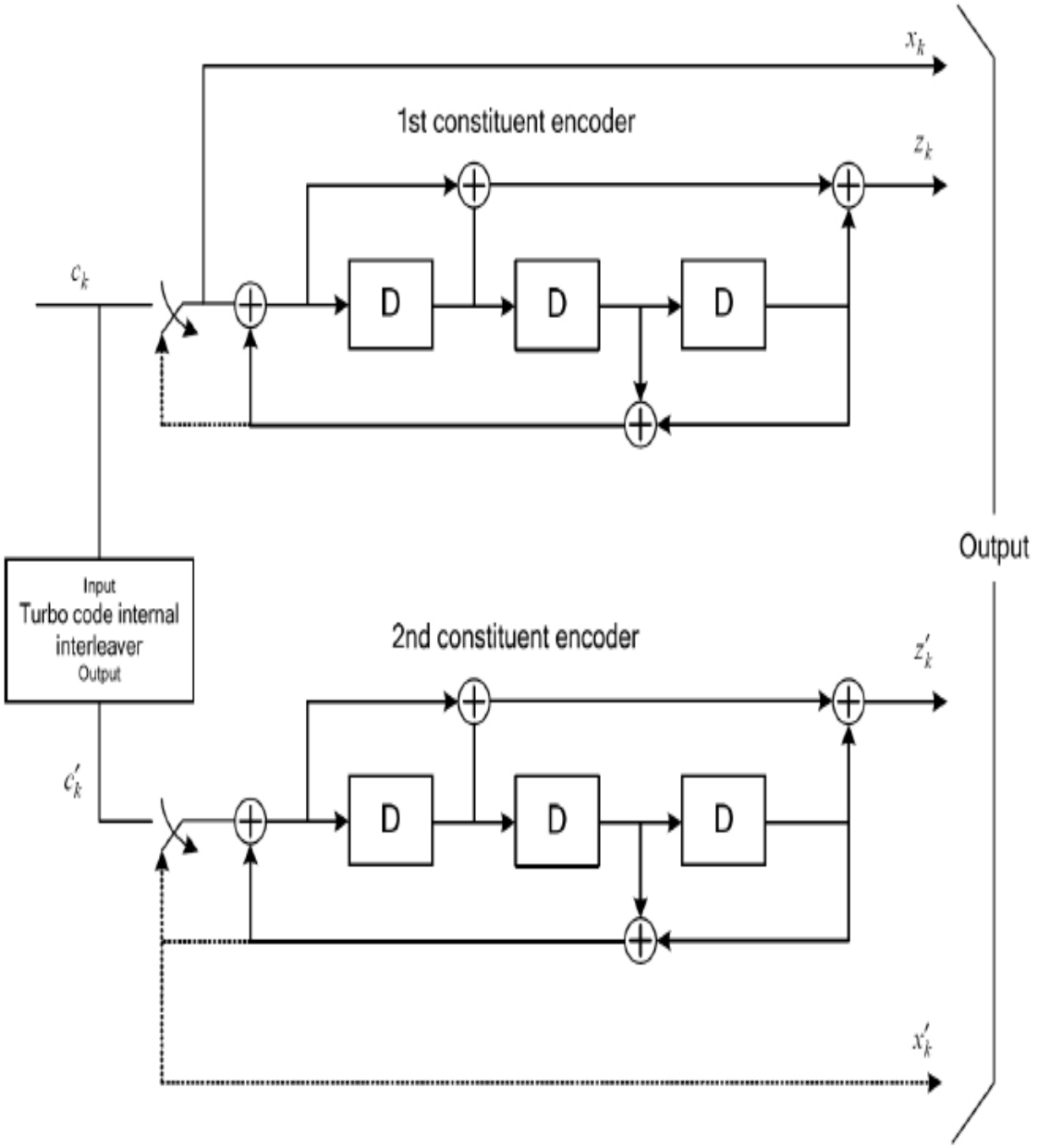


Figure 9: Turbo Encoder Example

2.2 TRELLIS TERMINATION

For turbo codes, appending five additional zero bits at the end of each N bit input Sequence does terminate the trellis in the all zero state due to the presence of feedback in The encoder structure. The insertion of extra bits for the recursive encoder to terminate in the null state depend on the current state of the encoder which is not predictable. In order to overcome this problem a simple technique as shown in Figure 10 for a (1, 5/7, 5/7) encoder is employed to accomplish trellis termination. Keeping the switch to position A(output 1) encodes the input sequence, in order to terminate the trellis in the zero memory state the switch position is simply moved to second location or position B(output 2)

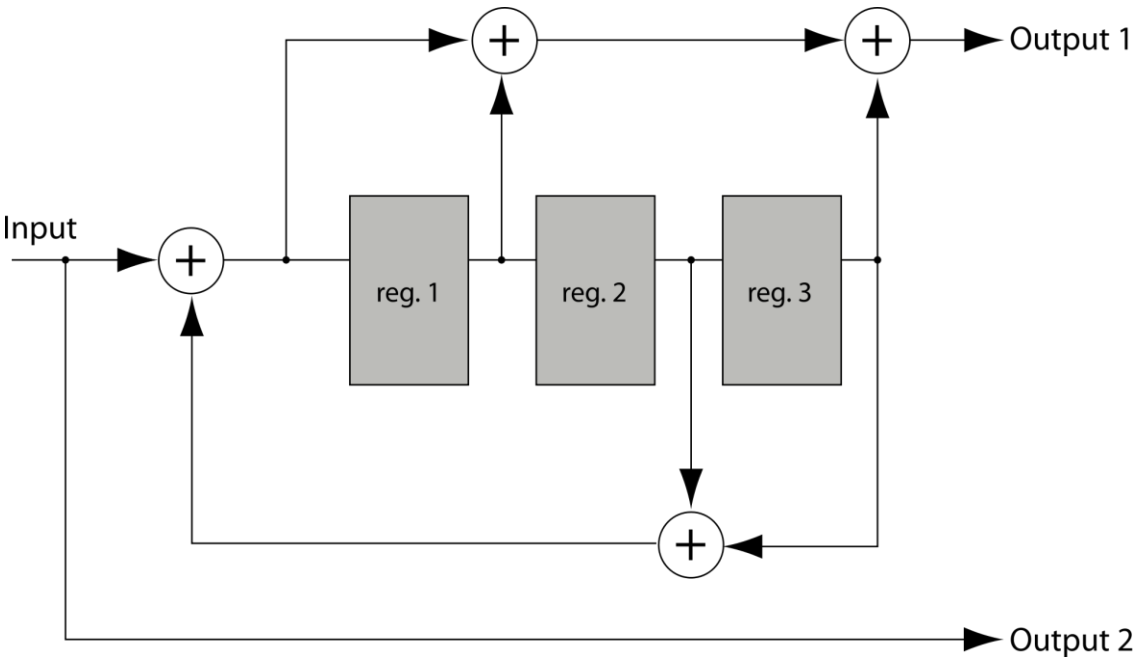


Figure 10: Trellis Termination Strategy for A RSC Encoder

2.3 STATE DIAGRAM

The state diagram completely describes the encoding of turbo codes for a particular generator polynomial. The memory contents of the encoder define the state of the encoder at a particular time instant. If (V) denotes the overall encoder memory, the total number of states is 2^V. The current state and the output of the encoder are uniquely determined by the previous state and current input. The encoder undergoes a state transition when the bits in the message block are shifted into the encoder. The state diagram is a graph that consists of nodes, representing the encoder states with the arrow direction representing the state transitions. Each directed line is labeled with the input/output pair. Given a current encoder state, the information sequence at the input determines the path through the state diagram and the output sequence.

2.4 INTERLEAVING

An interleaver is a device which scrambles the symbols from several code words so that the symbols from any given code word give a random effect. When the deinterleaver reconstructs the code word by arranging the received sequence to its original order, error bursts introduced by the channel are broken up and spread across several code words. Before the discovery of turbo codes interleavers were mostly used in serially concatenated codes and in multipath fading channels to enhance the overall error correcting capability of the coding scheme. They were used differently for the first time in parallel concatenated turbo codes and proved to reduce the number of code words with small distances in the code distance spectrum (that is it generates less code words with minimum hamming distance). Hence, understanding the behavior of interleavers is essential to obtain improved performance in the error floor region of turbo codes. A comparative analysis of turbo codes using different interleavers was carried out and described below are the different types of interleavers.

2.4.1 RANDOM INTERLEAVERS

A random interleaver scheme uses a fixed randomly generated permutation scheme to map the information sequence to the permutation order. A random interleaver scheme having a size of 12 is shown below in figure 11

Data index ⟶	1	2	3	4	5	6	7	8	9	10	11	12
Interleaved data index	9	5	11	2	1	7	3	10	8	12	4	6

Original data index Interleaved data index

Figure 11: A Random Interleaver.

2.4.2 BLOCK INTERLEAVERS

The block interleaver is one of the most basic and commonly used types of interleaver in communication systems. It scrambles the information sequence by writing it row wise and reading it column wise. In general a block interleaver can be described in terms of a (N × M) matrix. There exist four variations for this scheme. The schemes vary according to the order in which columns are read (LR: left to right or RL: right to left) and the order in which rows are read (TB: top to bottom or BT: bottom to top). A (4×4) block interleaver is shown in Figure 12, where the matrix elements represent the index of the information sequence. All of the four possible schemes of a block interleaver are shown in Figure 12

□□0 1 2 3 4 5 6 7 8 9 10 11 13 12 14 15 □□
□□0 4 8 12 1 5 9 13 2 6 10 14 3 7 11 15 □□

(a) A (4x4) LR/TB block interleaver.

□□0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 □□
□□12 8 4 0 13 9 5 1 14 10 6 2 15 11 7 3 □□

(b) A (4x4) LR/BT block interleaver.

□□0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 □□
□□3 7 11 15 2 6 10 14 1 5 9 13 0 4 8 12 □□

(c) A (4x4) RL/TB block interleaver.

□□0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 □□

□□15 11 7 3 14 10 6 2 13 9 5 1 12 8 4 0 □□

(d) A (4x4) RL/BT block interleaver.

Figure 12: Four Different Strategies For Block Interleaving.

	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15

Table 1: For the Block Interleaver Above

2.4.3 ODD EVEN INTERLEAVER

This interleaver design is specially used with a punctured rate 1/2 turbo code, which is obtained by applying the puncturing operation on the two parity output sequences of a rate 1/3 turbo code. This interleaving strategy is useful in uniformly distributing the error correcting capability of the code. This interleaver picks the odd positioned coded parity bits from the sequence c2 as shown in Table 2, here the subscript denotes the position of the bit sequence

X0	X1	X2	X3	X4	X5	X6	X7	X8
C20	-	C22	-	C24	-	C26	-	C28

Table 2: Odd Coded Parity Digits of the First Encoder

The data for the second encoder is assumed to be interleaved (using as 3x3 LR/TB block interleaver as shown in Table 3)

0	1	2
3	4	5
6	7	8

Table 3: A (3×3) LR/TB block interleaver.

The second component encoder is used to generate the even positioned coded parity bits. Table 4 shows the block-interleaved data in the first row, whereas the even parity bits are in the second row.

X0	X1	X2	X3	X4	X5	X6	X7	X8
C20	C33	C26	C31	C24	C37	C22	C35	C28

Table 4: Multiplexed Coded Sequence

2.4.4 INTERLEAVER TYPES BASED ON GOLDEN SELECTION

The golden section has application in many mathematical problems. The golden section principle states that for a given line segment of unit length, the aim is to divide it into a long segment of length g, and a shorter segment of length 1-g in such a way that the relation of the longer segment to the entire segment is equal to the relation of the shorter segment to the longer segment. Figure 13 below illustrates this principle

Using this principle the golden section value is calculated to be g = 0.618. This value of g will be used in the definition of two subsequent interleaver definitions which are Golden Relative Prime Interleaver and Golden Interleaver and we see them in details in subsequent sub chapters.

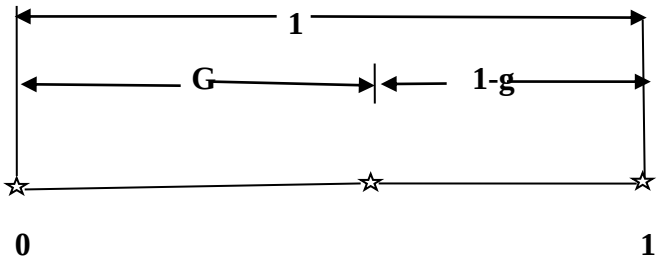


Figure 13: Illustration of Golden Section Principle

2.4.4. a. GOLDEN RELATIVE PRIME INTERLEAVER

The golden relative prime interleaver computes the scrambler indexes using the following relation (i (n) =s + np mod N, n=0....N-1) where, s is an integer starting index, p is an integer index increment, and N is the interleaver length. The values of N and p must be relative primes in order to ensure the uniqueness of each index element. The starting

index s is usually set to 0, but other integer values can also be selected. The relative prime increment p is chosen close to one of the non-integer values of c, which is computed as shown below

c=n (g^m + j) /r ……eqn1

In the expression of c, g is the golden section value, m is any integer greater than zero, r is the index spacing, and j is any integer modulo r. The preferred values of j, m and r are 0, 1 and 1 respectively.

2.4.4. b GOLDEN INTERLEAVER

This does not depend on the usage of relative primes and integer modulo arithmetic but is rather based on the principle of sorting real- valued numbers derived from the golden section value. After computing the value of c as given by equation (1) the real-valued golden vector E is calculated as follows:

E (n) = s+ nc mod N, n=0.....N-1.....eqn 2

Where, s is any real starting value. After finding the vector e it is possible to find the index vector z. If the sorted version of the vector e is denoted by a, then the index vector z and the sorted vector a will be relates as shown below:

A (n) = E [z (n)].....eqn 3

Finally, the golden interleaver indexes are then computed by the following expression:

I (z (n)) =n, n=0.....N-1.....eqn 4

The interleaver design for turbo codes is not an exact science, so the design methods can only be validated with the simulation results, we will also analyze the performance of the described interleaver through the simulation results in the following chapter.

CHAPTER 3 CHANNEL CODING (ADDITIVE WHITE GAUSSIAN NOISE)

3.1 DEFINITION OF ADDITIVE WHITE GAUSSIAN NOISE

It's a very simple model of the imperfections contained in a communications channel. When you transmit a specific signal into deep space or atmosphere or copper line to be received at the next end(receiver end), there are disturbances (also called noise) present in the channel(space/atmosphere/copper line) due to various reasons and interferences. One such reason is the thermal noise by the virtue of electrons' movement in the electronic circuit being used for transmission and receiving of the signals. This so called disturbances or noise is modeled here as an Additive White Gaussian Noise. Mostly it is also assumed that the channel is Linear and Time Invariant. The most basic results further assume that it is also frequency non-selective.

Let' us look at the time-domain behavior of this noise:

Additive: Because the noise will get added to your transmitted signal not multiplied. So, the received signal y (t) = x (t) + n (t), where x (t) was the original clean transmitted signal, and n (t) is the noise or disturbance in the channel and y (t) is the new distorted signal.

Gaussian: This thermal noise is random in nature, of course noise can't be deterministic otherwise you would subtract the deterministic noise from y (t) as soon as you receive y (t). So, this random thermal noise has Gaussian distribution with 0 mean and variance as the Noise power. Leaving out the variance part to avoid further complications in this paper ,but do remember that if variance of Gaussian is high then it’s dangerous as you may need to increase the power of x(t) or be satisfied with higher chances of error in the signal. 0 mean

means that the expected value n (t) during any time interval T is 0. But simply put, it also means that on an average n (t) will take 0 value. And probability of n (t) = 0 is the highest and the probability rapidly decreases as you increase the magnitude of n (t) which is a very good thing.

Looking at the frequency domain behavior of this noise:

White: meaning same amount of all the colors or same power for all the frequencies. Which means that this noise is equally present with the same power at all the frequencies. So, in frequency domain, Noise level is flat and the same throughout at every frequency this white noise is also knows as wideband noise.

Wideband noise comes from many natural sources, such as the thermal vibrations of atoms in conductors (referred to as thermal noise or Johnson-NY Quist noise), noise, black from the earth and other warm objects, and from celestial sources such as the Sun. The central limit theorem of probability theory indicates that the summation of many random processes will tend to have distribution called Gaussian or Normal.

Additive White Gaussian Noise (AWGN) is often used as a channel model in which the only impairment to communication is a linear addition of wideband or white noise with a constant spectral density(expressed as watts per hertz of bandwidth) and a Gaussian distribution of amplitude. The model does not account for fading, frequency selectivity, interference, nonlinearity or dispersion. However, it produces simple and tractable mathematical models which are useful for gaining insight into the behavior of a system before these other phenomena are considered.

The Additive White Gaussian Noise (AWGN) channel is a very suitable model for many satellite and deep space communication links. It is not a good model for most terrestrial and earthly links because of multipath, terrain blocking such as mountains, interference from other signals, etc. However, for terrestrial path modeling, AWGN is commonly used to simulate background noise of the channel under study, in addition to multipath, terrain blocking, interference, ground clutter and self-interference that modern radio systems encounter in terrestrial communication operations.

3.2 AWGN CHANNEL CAPACITY

In this section, we define the information capacity of the AWGN channel as the maximum of the mutual information between the input and the output over all distributions of the input that satisfy the power constraint defined below. Information capacity of an AWGN channel, the information capacity of the AWGN channel with power constraint P is defined as;

$C = \max I(X; Y)$

$F(x): E[X/2] \leqslant P$

By expanding the common information, we get

$I(X; Y) = h(Y) - h(Y|X)$ ……..eqn 5

$= h(Y) - h(X + Z|X)$ ……..eqn 6

$= h(Y) - h(Z)$ ……eqn 7

Where eqn. 7 follows from the result in eqn. (5) ($h(Y|X) = h(X + Z|X) = h(Z|X) = h(Z)$,). Recall that if $Z \sim N(0, \sigma 2)$, then its differential entropy is: $h(Z) = 1/2 \log(2\pi e\sigma 2)$. We also remark that since X and Z are independent, and using the fact that $Var(X) \leqslant P$, then

$Var(Y) = Var(X) + Var(Z)$…….eqn 8

$\leq P + \sigma 2$ …….eqn 9

Moreover, we use the fact that the Gaussian distribution maximizes the entropy for a given variance. Applying this fact to the received signal Y, whose variance is upper-bounded by $P + \sigma 2$, we get that $h(Y) \leqslant 1/2 \log[2\pi e(P + \sigma 2)]$. This says that the input which maximizes this entropy is $X \sim N(0, P)$. We are now ready to upper-bound the mutual information

$I(X; Y) = h(Y) - h(Z)$ …….eqn 10

$\leq 1/2 \log[2\pi e(P + \sigma 2)] + 1/2 \log(2\pi e\sigma 2)$…….eqn 11

= 1/2 log (1+ P/σ2)……eqn 12

By using Eq. 26, we finally got that the information capacity of the AWGN channel is

C = max I(X; Y) = 1/2log (1+ P /σ^2)……..eqn 13

F(x): E[X^2] ⩽P……..eqn 14

And this maximum is achieved when X ∼ N (0, P), i.e. f(x) = 1/√2πP e^−(x^2/2). In communication theory, the ratio P/σ2 is often called signal-to-noise ratio (SNR).The AWGN channel is depicted by a series of outputs Y*i* at discrete time event index represented as *i* which is the addition of the input X*i* to the noise Z*i*, where Z*i* is independent and identically distributed and is gotten from a zero-mean normal distribution with variance N (the noise). The Z*i* are further assumed to not be correlated with the X*i*. The capacity of the channel is said to be infinite unless the noise n is at a nonzero state, and the X*i* are sufficiently constrained. The most common constraint on the input is the so-called "power" constraint, requiring that for a code word (X1, X2,..., Xn) transmitted through the channel, we have:

$$\frac{1}{k}\sum_{i=1}^{k} x_i^2 \leq P,$$ …….eqn 15

Where P represents the maximum channel power. Therefore, the channel capacity for the power-constrained channel is given by:

$$C = \max_{f(x)\ \text{s.t.}\ E(X^2)\leq P} I(X;Y)$$ …….eqn 16

Thus the channel capacity C for the AWGN channel is given by:

$$C = \frac{1}{2}\log\left(1+\frac{P}{N}\right)$$ …….eqn 17

Also the Shannon's power efficiency limit has to be considered strongly and effectively and this has no dependency on BER. Shannon's limit tells us the minimum possible Eb/N0 required for achieving an arbitrarily small probability of error as M→∞. (M is the number of signaling levels for the modulation technique, for BPSK M=2, QPSK M=4

and so on…).It gives the minimum possible Eb/N0 that satisfies the Shannon's-Hartley law. In other words, it gives the minimum possible Eb/N0 required to achieve maximum transmission capacity (R=C, where, R is the rate of transmission and C is the channel capacity). It will not specify at what BER you will get at that limit. It also will not specify which coding technique to use to achieve the limit. As the capacity is approached, the system complexity will increase drastically. So the aim of any system design is to achieve that limit. As an example, a class of codes called Low Density Parity Codes (LDPC) near the Shannon's limit but it cannot achieve it. The Shannon limit derived above is called absolute Shannon power efficiency Limit. It is the limit of a band-limited system irrespective of modulation or coding scheme. This is also called unconstrained Shannon power efficiency Limit. If we select a particular modulation scheme or an encoding scheme, we can calculate the constrained Shannon limit for that scheme.

3.3 AWGN CHANNEL RELIABILITY

With the emergence of wireless and mobile communication systems and technology, new channel models and impairment phenomena have appeared. Fading and multipath effects are the most severe among these phenomena. The techniques developed for communications over a band limited AWGN channels have been applied to fading channels. However, performance of these techniques in fading is significantly inferior to additive noise channel results. So far, some great improvements have been made, but data rates achievable in fading are still quite modest in comparison to those for wire-line channels. Due to the extreme difference in reliability between fading and noise channels, some fundamental questions arise which causes the creation of a new code which may result in an improvement over all other known codes, to check the improvements the limits to communications over multipath fading channels should be determined. The channel

capacity is a crucial asset that can be found by the use of information theory to determine the achievable transmission rates for reliable communication. That is to say, if the transmission rate is higher than the channel capacity, reliable communication and arbitrarily small decoding error probability cannot be achieved. However, if the transmission rate is to be kept below the capacity, we know that reliable communication is possible. The reliability of the communication process can be measured by defining the reliability. This measure, in fact, exists and is known to information theorists as the "Random Coding Reliability Function". Also known as the "Random Coding Error Exponent", due to Gallager [1965]. Gallager has shown that the probability of error of the best block codes of length N and rate R decreases exponentially with block length. Gallager's bound determines the behavior of the probability of error in terms of the transmission rate as well as the code length N, which reflects coding complexity, thus, making the bound an attractive measure of reliability.

So therefore the random coding reliability function has been derived for various fading channels that model current wireless and mobile communication channels. The effects of amplitude fading, non-ideal channel state information, space diversity and fading time correlation on the reliability function have been considered. Channels with average and peak-power-constrained inputs as well as channels with discrete and continuous inputs have also been studied. Comparison with the additive white Gaussian noise (AWGN) channel has been made in order to determine the amount of degradation due to fading. In addition, estimates of the required code lengths for a certain probability of error have often been calculated in order to aid in the assessment of the required coding complexity over fading channels

The performance of the turbo decoding principle depends on the sharing of information between the constituent decoders. The more information we will have the better the performance of the decoding scheme will become. In order to obtain the channel information a variable called the channel reliability introduced in the beginning and is defined as follows

Lc =4(Eb/No) *a* ……eqn 18

The value *a* is defined as the fading amplitude of the channel. It is defined to be *a* =1 for the additive white Gaussian channel as it is a non-fading channel. For fading channel the fading amplitude must be obtained by using channel state estimation and since this thesis assume that there is no channel state information available the value of a is chosen to be a = 0.8 for slow independent Rayleigh fading channel. The value was chosen to be less than one as multipath channels have a reliability value less than one.

CHAPTER 4 TURBO DECODING

Iterative coding is the major type used here and it has been seen to be used in the decoding of the class of parallel concatenated codes first introduced in 1993 by G.Berrou. An iterative decoder for concatenated codes consist of two soft output decoders as mentioned in earlier chapters for the component codes separated by an interleaver
The component decoders are based on either a maximum a posteriori (MAP) algorithm or a Soft Output Virtebri Algorithm (SOVA). The iterative algorithms performs the information exchange between the two components decoders.

The iterative decoding of turbo codes is based on the soft output maximum a posteriori probability (MAP) algorithm .The MAP algorithm was first brought to light by Chang and Hancock in 1966 and was proposed for signal detection in channels with memory. The same algorithm was also used by Bahl et al in 1974 for decoding of linear codes. The MAP algorithm minimizes the symbol (or bit) error probability, while SOVA reduces the sequence error probability. The performance of MAP and SOVA algorithms are high and identical at signal-noise-ratio (SNR). The performance gain of MAP algorithm over the SOVA at low SNR leads to a performance advantage in iterative decoding.

The MAP is algorithm is computationally more complex than the SOVA. MAP algorithm involves multiplication and exponentiations while SOVA involves adding, compare and select operation. However, due to the push for strikingly low bit error rates, the MAP or the Log-MAP has been most commonly used in turbo codes since they are based on the optimal decoding rule. In contrast, the SOVA is an approximation to the MAP sequence decoder and will have a slightly worse bit error performance. Though SOVA suffers from performance degradation as opposed to the Log-MAP decoding rules, it has much reduced complexity.

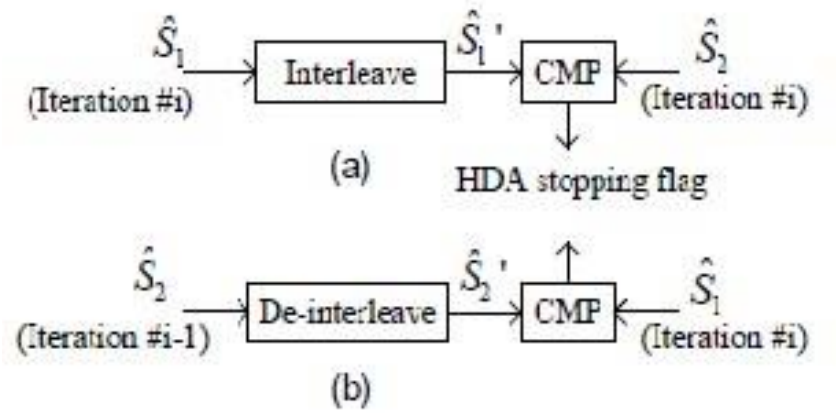


Figure 14: The iterative structure of a turbo decoding scheme

The turbo decoder uses a posteriori probability (APP) aslo known as a MAP probability decoder as the constituent decoder component. Each System object corresponds to a constituent encoder which provides an updated sequence of log-likelihood values for the uncoded bits from the received sequence of log-likelihoods for the channel (coded) bits. For each set of receiver channel sequences, the decoder iteratively updates the log-likelihood channel sequences, the decoder iteratively updates the log-likelihood for the uncoded bits until a stopping criterion is met. This simulation in this paper uses a fixed number of decoding iterations, as specified by the (Number of decoding iterations) parameter in the model's (Model Parameters) block The default number of iterations is set automatically at six The (Termination Method) property for the APP Decoder System object is set to be "Terminated" to match the encoders. The decoder does not assume knowledge of the tail bits and as a result, these are excluded from the

multiple iteration. The internal interleaver of the decoder is identical to the one the encoder uses. It reorders the sequences so that they are properly aligned at the two decoders

4.1 MAP ALGORITHM

To first understand the decoding of turbo codes, a preliminary understanding of the MAP (BCJR) algorithm is necessary known as Bahl, Cocke, Jelinek and Raviv. In the originally the idea was set out to calculate and estimate the a posteriori probabilities of the states and transitions of a Markov sequence which is transmitted through a discrete memory-less channel. This work resulted in an algorithm that minimizes symbol error rates while trying to decode the block and trellis codes. The purpose of the MAP algorithm is to minimize the symbol error rate for the decoding of trellis and block codes. Therefore, after receiving the information bits through the channel, the job of the decoder is to determine the most likely input bits (original/uncoded information sequence), based on the received signals. Since the input is over the binary alphabet, it is conventional to form a log-likelihood ratio (LLR) and base the bit estimates on comparisons based on magnitude of the likelihood ratio to a threshold. The log-likelihood ratio for the input signal indexed at time t is defined as:

$\Lambda (Xt) = In\ P (Xt= 1|r)/P (Xt=0|r)$……..eqn 19

In this expression, P (Xt =i|r) is the a posteriori probability of the information bit, Xt = i, where i ∈ {0, 1}, when we have the knowledge of the received data r. The decoder produces estimates of the information bits based on the values of the log-likelihood ratio. The magnitude of the log-likelihood ratio is defined as the soft output or soft value which can be passed after processing to the other decoder as a priority information. Also, the sign of the LLR determines the hard estimate of the original information sequence. The estimator obeys the following rule:

$Xt = *1$ *if* $\Lambda (Xt) \geq 0$…….eqn 20

** 0 otherwise*

In order to perform the decoding when the information is received through the channel, the log-likelihood ratio must be computed and present.

4.2 LOG MAP DECODING

The LLR computed by the MAP algorithm is the soft output value which must be converted into the a priori information before it can be utilized by the next component decoder. The a priori information for the first and second constituent decoder is obtained by the following relations below;

$\Lambda 2 = \Lambda 2e - Lc\,(y) - \Lambda 1$…….eqn 21

$\Lambda 1 = \Lambda 1e - Lc\,(y) - \Lambda 2$……eqn 22

Where $\Lambda 1e$ and $\Lambda 2e$ represent the extrinsic information from the first and second decoder Respectively. The turbo decoder generally performs the decoding of the noisy, received Information sequence through the communication channel in an iterative manner. This Iterative decoding structure of the error correcting coding schemes has proved to perform Amazingly well in providing low bit error rates and has thus reduced the gap between the Channel capacity limit defined by Shannon and the attained bit error rates. In order to Decode the data received from the channel, the first decoder should have a priori information and since, for the first iteration there is none available, it is assumed as zero. This is also called initialization. In section 4.1 the MAP algorithm was discussed in details in order to have the understanding of the decoding algorithm. In this work, the Log-MAP Decoding algorithm is used which in fact is based on the same idea as the MAP decoding Algorithm with the benefit that it simplifies the computation by eliminating the multiplicative operations and the need to store small values for the probabilities by including the logarithm operator in the computation. The multiplicative operation is computationally more expensive than the addition operator in terms of processing speed of a microprocessor. Also the requirement for large amounts of memory to store the probabilities in the computation of the log-likelihood ratio makes the implementation of this algorithm complex. After defining all the entities required in the decoding process the complete decoder structure for Log-MAP algorithm is shown below The Log-MAP algorithm is implemented as a two part algorithm. These two parts are the forward and

backward recursions. Details of each part are explained in what follows

Basically, max-log-MAP algorithm can be divided into four computation tasks, which are branch metrics generation, forward metrics generation, backward metrics generation, and generation of a soft or hard bit estimate together with new extrinsic information

Forward recursion:

Initialize α0 (k) for the states k = 0, MS −1, with α0 (0) = 0 and α0 (k) = −∞for k not = 0.

For t = 1, 2, τ and the states k = 0, 1... MS − 1 calculate for all the branches in the trellis

$$\alpha_k(s) = \log\left[\sum_{s',m} \exp\left(\alpha_{k-1}(s') + \gamma_{k,m}(s',s)\right)\right]$$ .....eqn 23

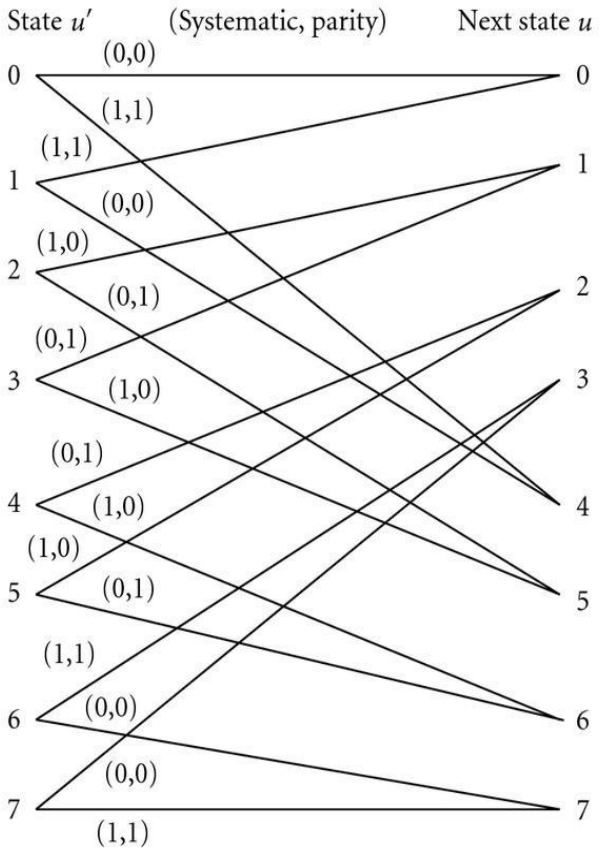


Figure 15: Trellis of eight state 3GPP constituent code. Transmitted systematic and parity bit pairs and corresponds with state changes of the component encoder.

Backward Recursion

Initialize βτ (k) for the states k = 0... MS −1, with βτ(0) = 0 and βτ(k) = −∞for k not = 0. - For t = τ −1... 1,0 and the states k = 0,1,...,MS calculate βt(k) as

$$\alpha_k^{(i)}(1) = \pi_k b_k(u_{i,1})$$
$$1 \le k \le M,\ 1 \le i \le N\ .$$

$$\alpha_k^{(i)}(t) = b_k(u_{i,t}) \sum_{l=1}^{M} \alpha_l^{(i)}(t-1) a_{l,k}\ ,$$
$$1 < t \le T,\ 1 \le k \le M,\ 1 \le i \le N\ .$$

$$\beta_k^{(i)}(T) = 1\ , 1 \le k \le M,\ 1 \le i \le N$$

$$\beta_k^{(i)}(t) = \sum_{l=1}^{M} a_{k,l} b_l(u_{i,t+1}) \beta_l^{(i)}(t+1)$$
$$1 \le t < T,\ 1 \le k \le M,\ 1 \le i \le N\ .$$

Figure 16: Shows the Backward Recursion Formula

Naturally, turbo decoders applying more accurate algorithms like log-MAP instead of max-log-MAP require more area and the longer critical path lowers clock frequency.

4.3 SOFT OUTPUT VITEBRI ALGORITHM (SOVA)

It is basically a modified vitebri decoder, the turbo code decoder is based on a modified Viterbi algorithm that incorporates soft-input and soft-output values along with the channel reliability values to improve decoding performance. For multistage (concatenated) convolutional codes, there are two main drawbacks to the conventional Viterbi decoders. First, the inner Viterbi decoder produces bursts of bit errors, which degrades the performance of the outer Viterbi decoders. Second, the inner Viterbi decoder

produces hard decision outputs, which prohibit the outer Viterbi decoders from deriving the benefits of soft decisions. Both of these drawbacks can be reduced and the performance of the overall concatenated decoder can be significantly improved if the Viterbi decoders are able to produce soft-output values. The soft values are passed on to subsequent Viterbi decoders as a priority information to improve decoding performance. This modified Viterbi decoder is referred to as the soft-output Viterbi algorithm (SOVA) decoder as mentioned in the earlier part of this paper. Inside the decoder, Soft-Output Viterbi Algorithm (SOVA) is used to determine the result with maximum likelihood. The process SOVA is similar to that Viterbi algorithm. A trellis is formed first.

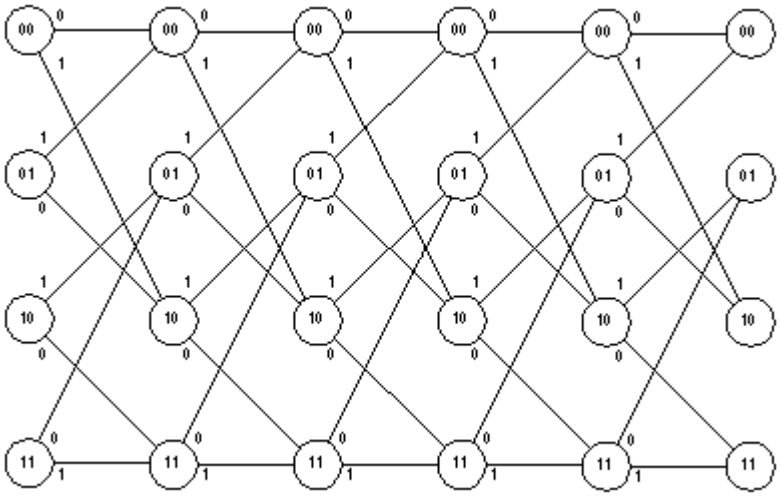


Figure 17: A Typical Trellis

The trellis is to show how the codes are outputted by encoder. The bits in each node represents the states of the encoder. Each line represents a transition on receipt of codes from encoder the encoder output (decoder input) for all combination of states and input of the encoders are summarized as follows.

State 1	State 2	Input	Output	Next State 1	Next State 2	Decoder Input
0	0	0	0	0	0	00
0	0	1	1	1	0	11
0	1	0	0	1	0	00
0	1	1	1	0	0	11
1	0	0	1	1	1	01
1	0	1	0	0	1	10
1	1	0	1	0	1	01
1	1	1	0	1	1	10

We the determine the Accumulated Maximum Likelihood for Each State from the above table, each state has its own input bit pattern. When bit streams are inputted to decoder, they can be compared to the input (x_0 x_1) to see whether they are matched. The result is represent by likelihood of input:

L_i = - if no bits are matched
1 if 1 bit are matched
0 if all 2 bits are matched
1

The overall likelihood of a transition is sum of likelihood of input and a-prior likelihood information (L_p).

$L = L_i + L_p$ ..... eqn 25

L_p = 0.5 x a-prior information L_{an} if x_0 are 1
-0.5 x a-prior information L_{an} if x_0 are 0

The algorithm is as follows: Start from state 00, the overall likelihood of each transition are evaluated. The overall likelihood of each node is obtained by the maximum accumulated likelihood. With this algorithm, the following will be obtained.

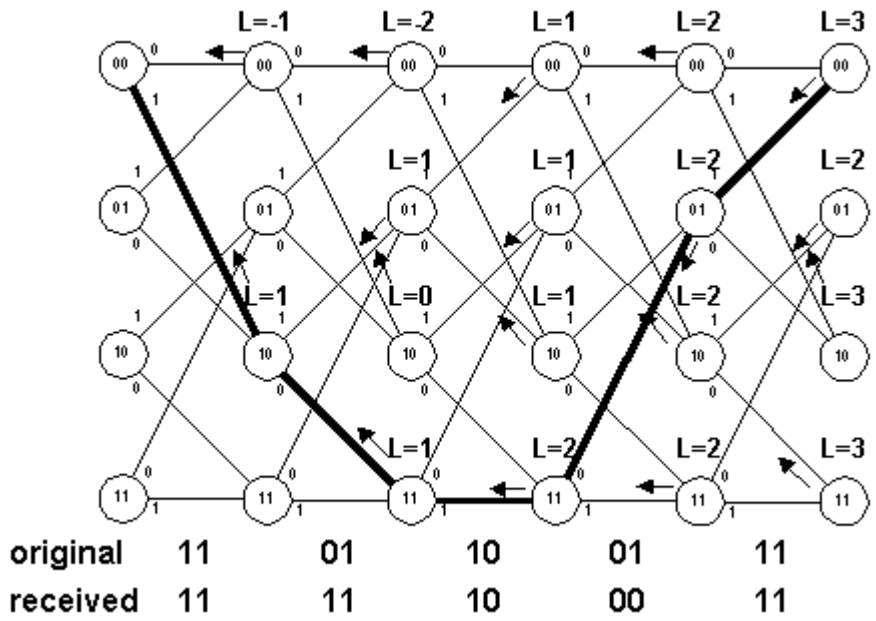


Figure 17: Another Trellis Diagram

Then you find the Surviving Path; The surviving path is thus derived by tracing back from last stage (stage 5) to the first one (stage 0) and is the results of hard-output Viterbi Algorithm.

[1 0 1 0 1]

The next step is that you determine the Soft Output, to find the soft-output, non-surviving paths are considered. We consider the non-surviving path the one that is product by making different decision in one of the stage in tracing back. For example, when tracing back from (last stage) stage 5, one of the non-surviving path is found by tracing back to state 00 instead of state 01 from stage 5 to stage 4. Following the arrows, bit 1 is also 1. We found that bit 1 will have the following results when decision is changed in different stages:

Stag 1 2 3 4 5

e						
Bit 1	X	X	0	1	1	where X means cannot trace back to state 00 in stage 0
Delta	-	-	3	1	2	

Here we define a function delta to describe the tendency to have non-surviving path. It is the difference in overall likelihood when a different decision in a particular the stage.
The soft-output of bit 1 is evaluated by the following formula:
Bit_value x *min* (*delta* which make bit 1 change to value other than *bit_value*)

bit_value = 1 for bit output = 1
-1 for bit output = 0

In bit one, the decision only changes when a state changes in stage 3. The minimum delta is 3. Thus the soft-output of the bit one is 3.
Repeat it for bit two (originally bit two = 0),

Stage	2	3	4	5	
Bit 1	X	1	1	1	where X means cannot trace back to state 00 in stage 0
Delta	-	3	1	2	

From the above results, in bit two, the decision changes when a state changes in stage 3, 4 and 5. The minimum delta is 1. Thus the soft-output of the bit two is -1.
Using this algorithm, the soft-output will become
[3 -1 1 -1 2]

4.3a FEEDING DATA TO ANOTHER DECODER

After the soft-output is evaluated by SOVA decoder, the data will be passed to the second decoder for further decoding. Before passing to data to the second decoder, two processes are performed to the decoder output:

the a-prior information (La_2) and the systematic data (x_0) are subtracted

d_1	3	-1	1	-1	2
La_2	0	0	0	0	0
x_0	1	1	1	-1	1
Le_1	2	-2	0	0	1

The result is multiplied by the scaling factor called channel reliability L_c. The reason of the factor is because the SOVA algorithm suffers a major distortion which is caused by over-optimistic soft outputs. The factor is used to compensate this distortion.

L_c = mean (Le_1) x 2 / var (Le_1)…….eqn 26

4.3. b ITERATIVE DECODING IN SOVA

The data from first decoder L_cL_{e1}, placed together with the systematic data *a* (x_k) and code information from of second encoder (y_2), are then fed into the second decoder for the decoding process, the decoding algorithm is the same as the first one. After decoding, the output of second decoder is processed in the same way and fed back to the first decoder. The process continues. The number of iterations depends on the designer that which I set. Usually, the larger the iteration, the more accurate the data but the longer the time its takes for decoding. Decision of Output: After iterations of decoding, the decoding results is the sign of the soft-output of the last decoder. Take the example, for the results of first decoder, the output become:

decoder output	3	-1	1	-1	2
Result	1	0	1	0	1

Which is the same as the input bit stream *u*. i.e., the error can be recovered.

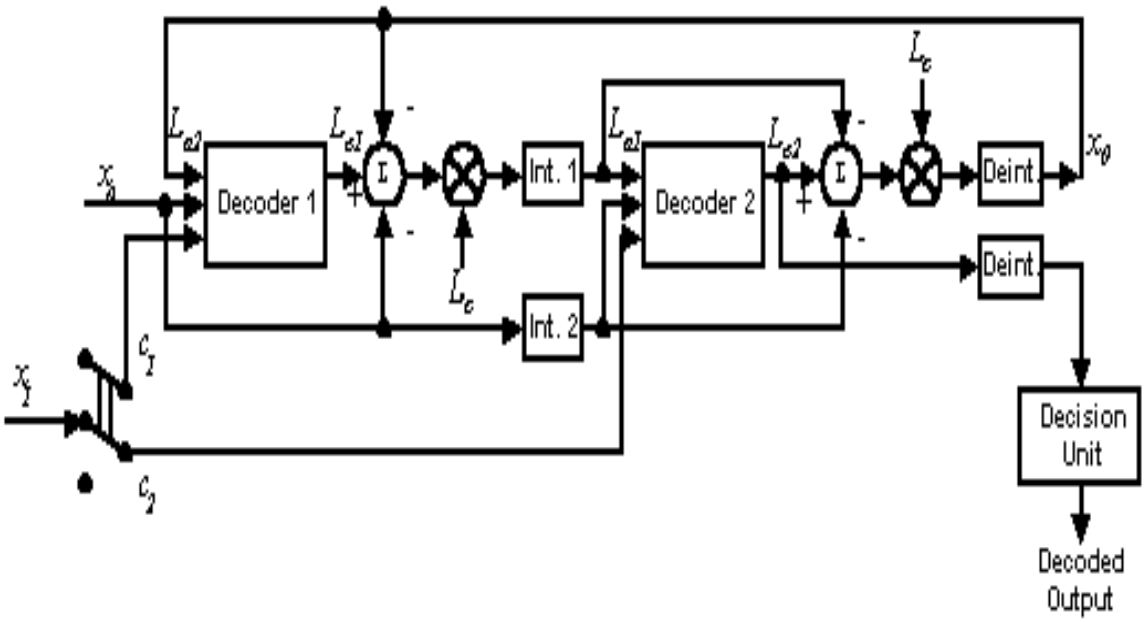


Figure 18: Turbo code decoding scheme for SOVA

CHAPTER 5 SIMULATIONS RESULTS AND ANALYSIS OF PERFOMANCE

In this chapter we shall see different examples of simulations done by myself on the Mat lab working template by running a giving command which carries out the simulation in it and displays the result as an information bit after calculating a specific error by comparing the information sent into the system and the information gotten from the decoder in the presence of an Additive White Gaussian Noise(AWGN).Then a graph is plotted where by it is the bit error rate (BER) against bit energy to noise power spectral density measured in decibels. As i started up with the simulations i took note of the time to ensure that it was factor and I noticed how long the simulations takes even as the various model parameters are being changed. Below we shall see a block diagram of the parallel concatenated convolution as I used it

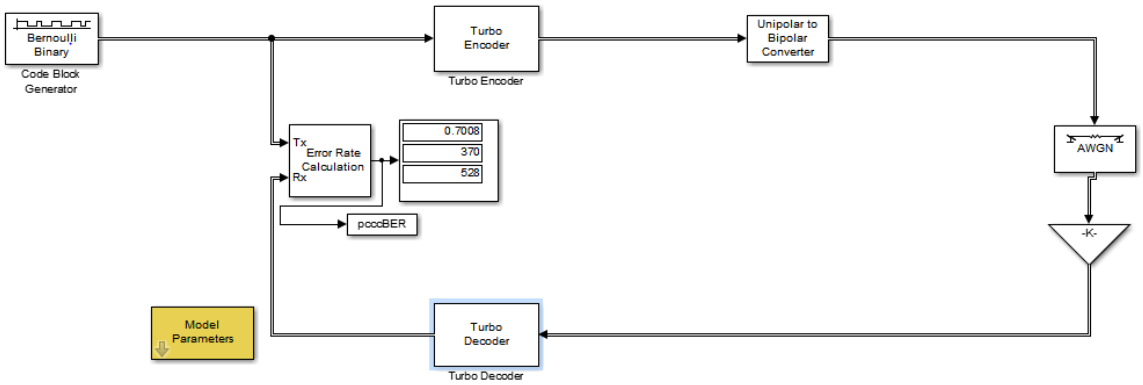


Figure 19: The Block Diagram Of My Parallel Concatenated Convolutional Code Model

For the all simulations below I used the code COMMPCCC in the MATLAB software or surrounding to run the model of simulation and the idea is to repeat the process till it cannot Correct any more errors depending on the number of iterations put in the model parameter. To set the model parameter I had to double click on the yellow box which says model

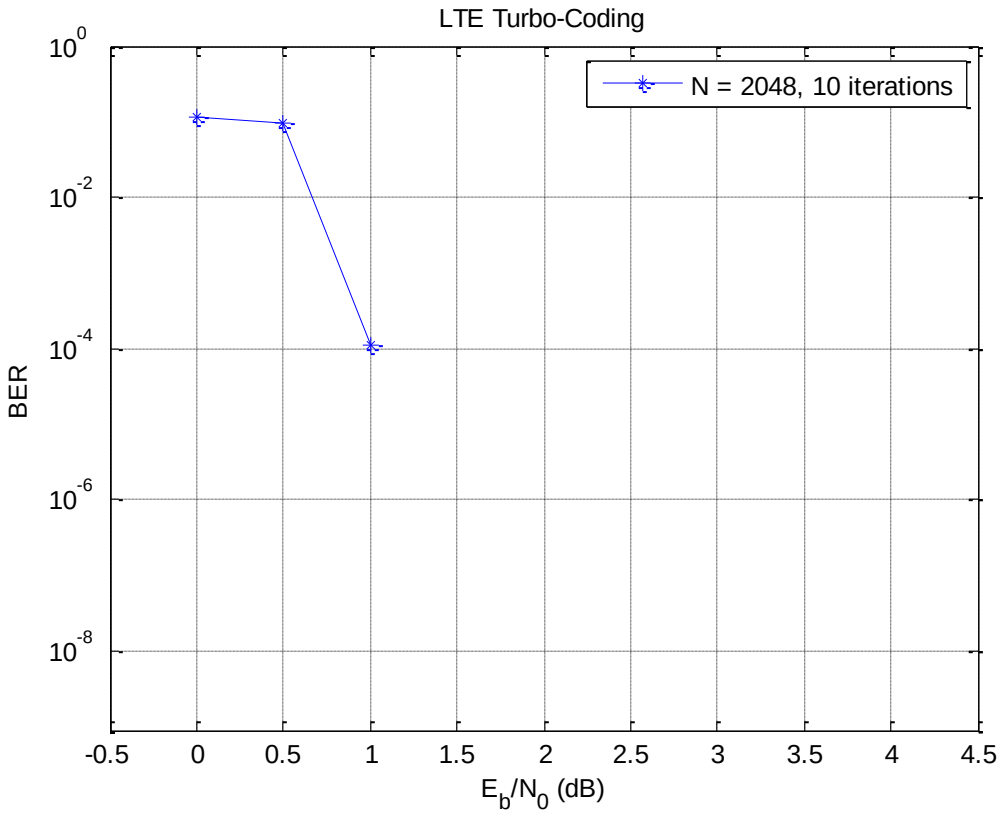
parameter then set the number of code block legnths also known as the bits to a suitable amount for that simulation and I would give each code block legnth below the result of the simulation after plotting it.An error rate is calculated by the comparison of the initail information and the processed and final information which has been influenced by the noise put into the system(AWGN).For the graph to be plotted I had to use the help of the bit error rate analysis tool to analyse the BER against the bit energy to noise power spectral density.
In the bit error rate analysis tool you have to use the Monte Carlo first of to run the simulation and mark out the points on the graph or you could use this code that I wrote below inserting it into matlab and running it. Here you change the parameters by changing the values of the parameters manually in the code and running the simulations simultaneously, after you change the parameters you want for each simulation you would get your different results. Below you would see the code necessary for this operation and the parameters highlighted in red.

```
%% Parallel Concatenated Convolutional Coding: Turbo Codes
%
% MATLAB(R) version of the Turbo coding example.
% Refer to <html/commpccc.html> for system details.

% Notice: Supply of this software does not convey a license nor imply any
% right to use any Turbo codes patents owned by France Telecom,
% Telediffusion de France and/or Groupe des Ecoles des Telecommunications
% except in connection with use of the software for the purposes of design,
% simulation and analysis. Code generated from Turbo codes technology in
% this software is not intended and/or suitable for implementation or
% incorporation in any commercial products.
%
% Please contact France Telecom for information about Turbo Codes Licensing
% program at the following address: France Telecom R&D - PIV/TurboCodes
% 38-40, rue du General Leclerc 92794 Issy-les-Moulineaux Cedex 9, France.

%     Copyright 2010-2012 The MathWorks, Inc.
%     $Revision: 1.1.6.5.4.1 $    $Date: 2012/12/15 20:24:59 $

%% System configuration includes Communications System Object constructions
numIter     = 3;                    % Number of decoding iterations
blkLength   = 6144;                 % Block length
EbNo          = [0 0.5 1 1.5 2];            % Eb/No values to loop over
```

```
maxNumErrs = 100;                       % maximum number of errors per Eb/No value
maxNumBlks = 2000;                      % maximum number of blocks per Eb/No value

% Turbo Encoder
%      Internal Interleaver indices
intrlvrIndices = commExamplePrivate('lteIntrlvrIndices', blkLength);
hTEnc = comm.TurboEncoder('TrellisStructure',...
            poly2trellis(4, [13 15], 13), 'InterleaverIndices', intrlvrIndices);

% AWG Noise - with variance to be reset in the processing loop
hAWGN = comm.AWGNChannel('NoiseMethod', 'Variance', 'Variance', 1);

% Turbo Decoder
hTDec    = comm.TurboDecoder('TrellisStructure', poly2trellis(4, [13 15], 13),...
             'InterleaverIndices', intrlvrIndices, 'NumIterations', numIter);

% BER measurement
hBER = comm.ErrorRate('ResetInputPort',true);

%% Processing loop
ber = zeros(length(EbNo),1); numBlks = ber; numErrs = ber;
R      = blkLength/(3*blkLength + 4*3);
for ebNoIdx = 1:length(EbNo)

      disp(['Processing EbNo = ' num2str(EbNo(ebNoIdx))]);
      noiseVar = 1./(2*R*10.^(EbNo(ebNoIdx)/10));
      hAWGN.Variance = noiseVar;

      while (numErrs(ebNoIdx) < maxNumErrs && numBlks(ebNoIdx) < maxNumBlks)
            data = randi([0 1], blkLength, 1);
            % Encode random data bits
            yEnc = step(hTEnc, data);

            % Add noise to real bipolar data
            rData = step(hAWGN, 1-2*yEnc);

            % Convert to log-likelihood ratios for decoding
            llrData = (-2/noiseVar).*rData;

            % Turbo Decode
            decData = step(hTDec, llrData);

            % Calculate errors - for the final iteration only
            berTemp = step(hBER, data, decData, 1);
            numErrs(ebNoIdx) = numErrs(ebNoIdx) + berTemp(2);
```

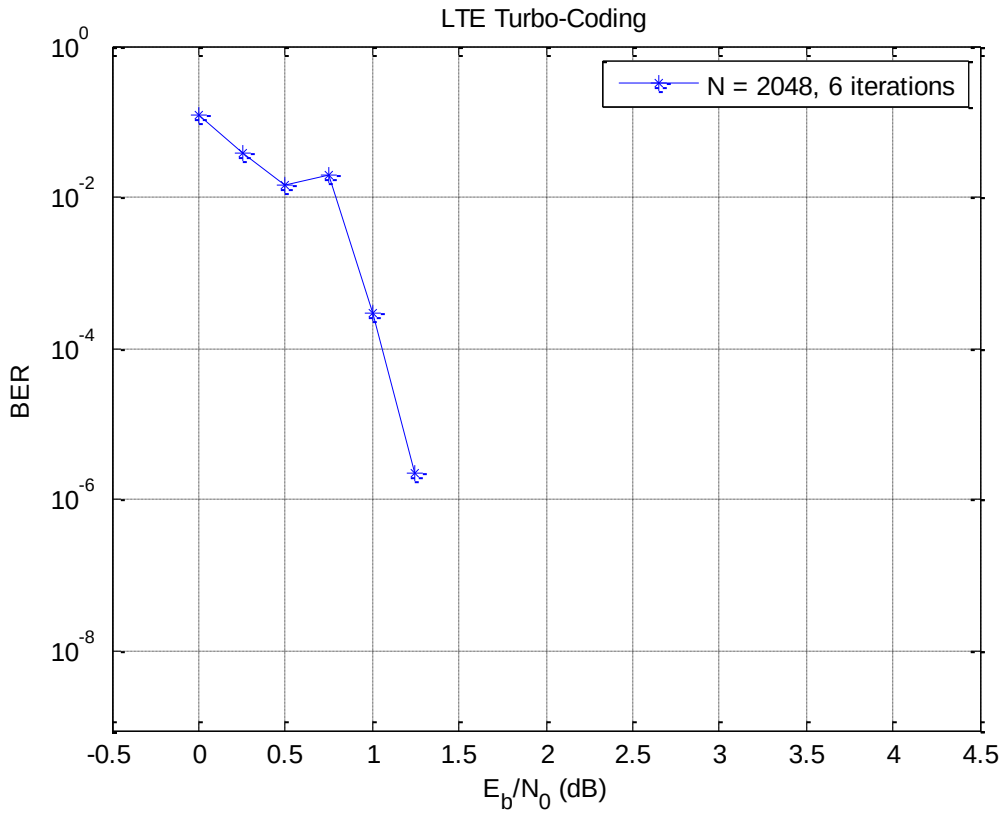
```
        numBlks(ebNoIdx) = numBlks(ebNoIdx) + 1;
    end

    ber(ebNoIdx) = numErrs(ebNoIdx)/(blkLength*numBlks(ebNoIdx));

end

%% Display results
figure; semilogy(EbNo, ber, '*-');
grid on; xlabel('E_b/N_0 (dB)'); ylabel('BER'); title('LTE Turbo-Coding');
axis([-0.5 4.5 9e-10 1]);
legend(['N = ' num2str(blkLength) ', ' num2str(numIter) ' iterations']);

% [EOF]
```

After I ran the simulations with BER Tool, I used the BER Figure window to plot individual BER data points. For me to fit a curve to a data set that contains at least four points, I had to select the box in the Fit column of the data viewer. The plot in the BER Figure window responds immediately to your choice. After that I have graphs of such below with their different parameters specified under them.



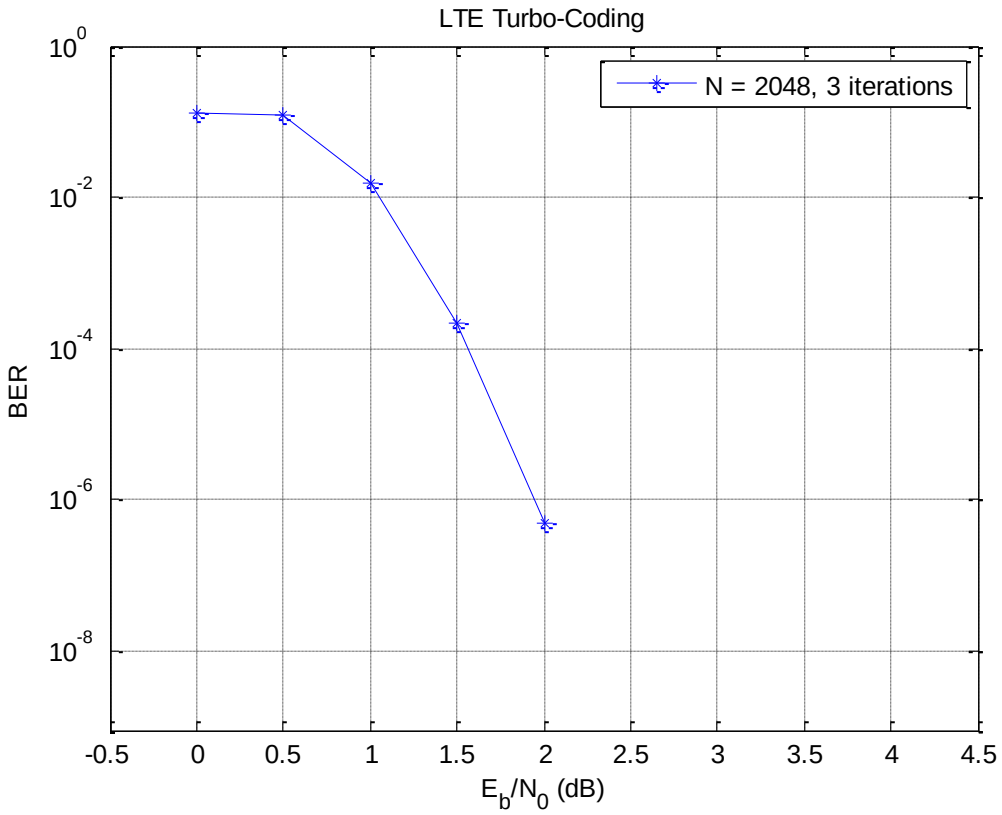
Here we start with a graph of iteration 10 and range of Eb/No to be (0:0.5:2)

And code block of 2048 bits

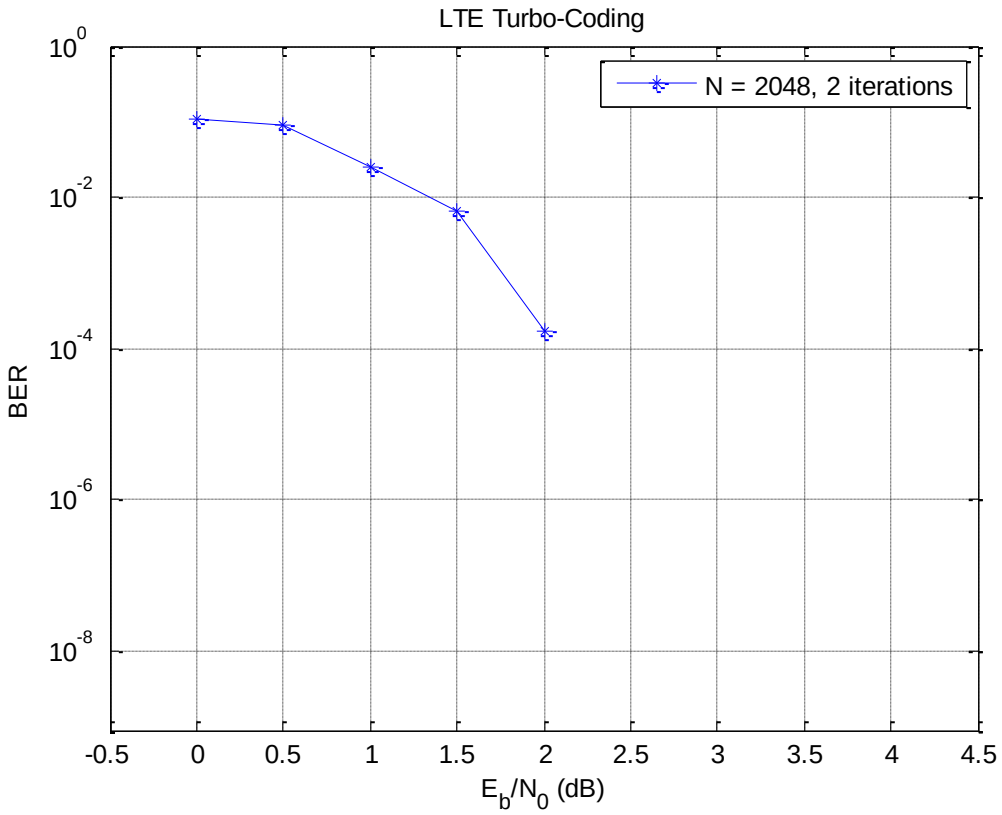


This is for a 2048 code block length and for 6 iterations and a range of (0:0.25:2) for (Eb/No) signal to noise ratio. Here we see another graph as we reduce the iterations leaving the code block length same and we would have such a graph;

For

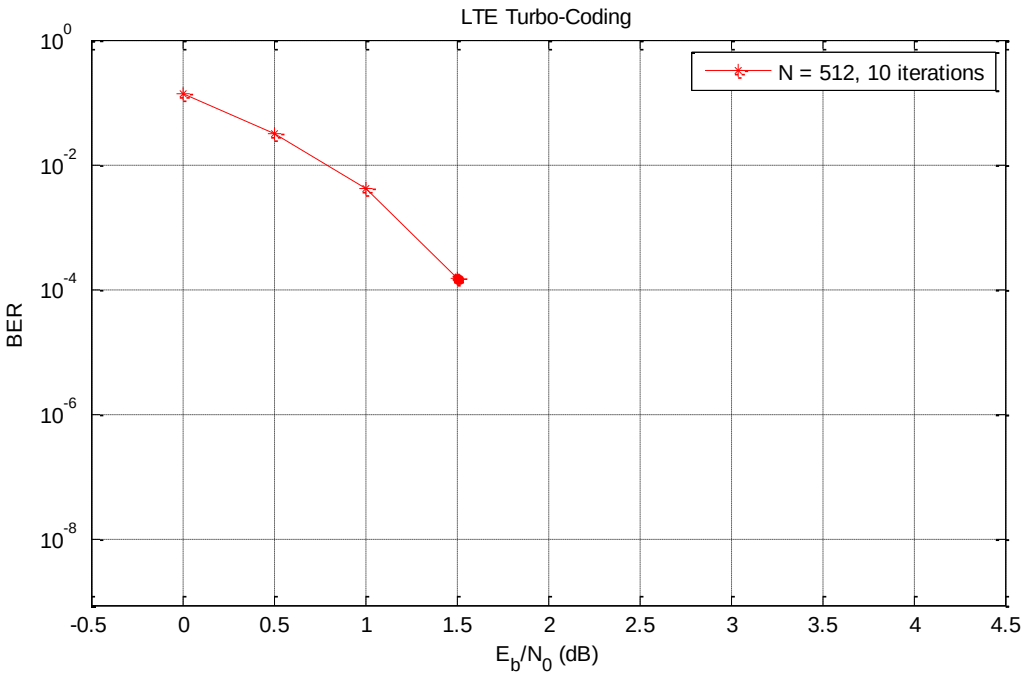


For code block length of 2048 with a reduced iteration of 3 this one took less time for the simulation. Below we have another simulation graph with same axis parameters and different iteration of 2 but same code block length of 2048 and this was a quick simulation

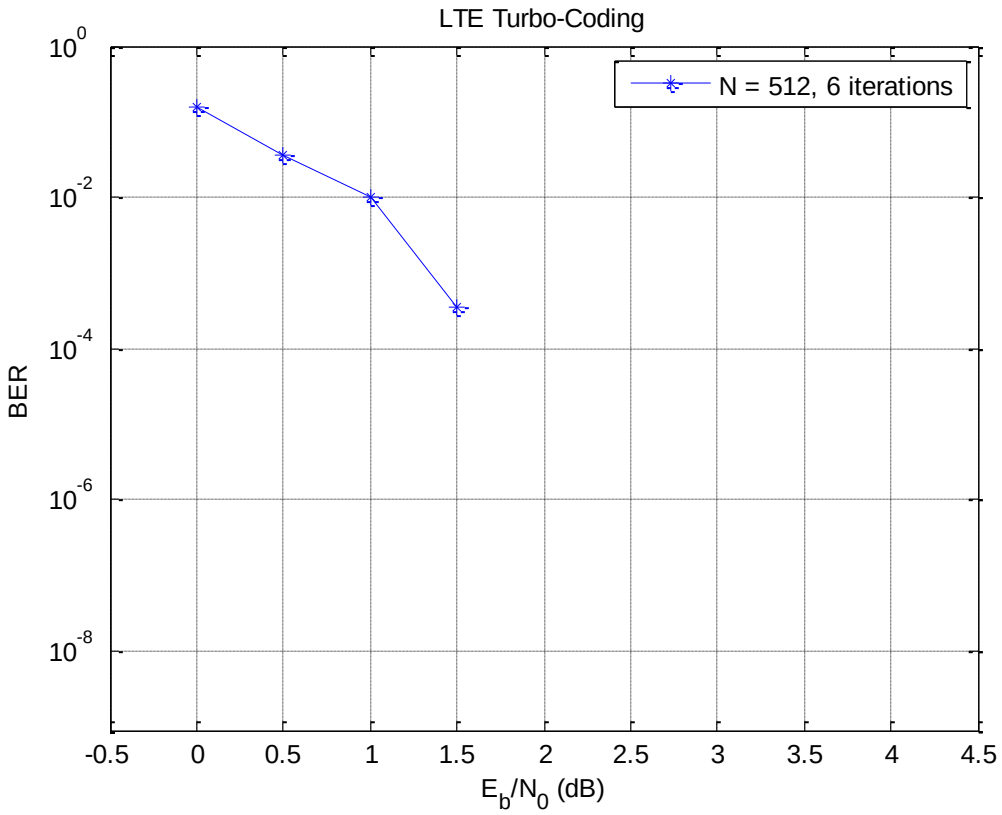


The number of iterations being 2 but same block code length. And noticing this we can see that the graph moves from left to right at the base meaning an increase in error as the iterations are being reduced.

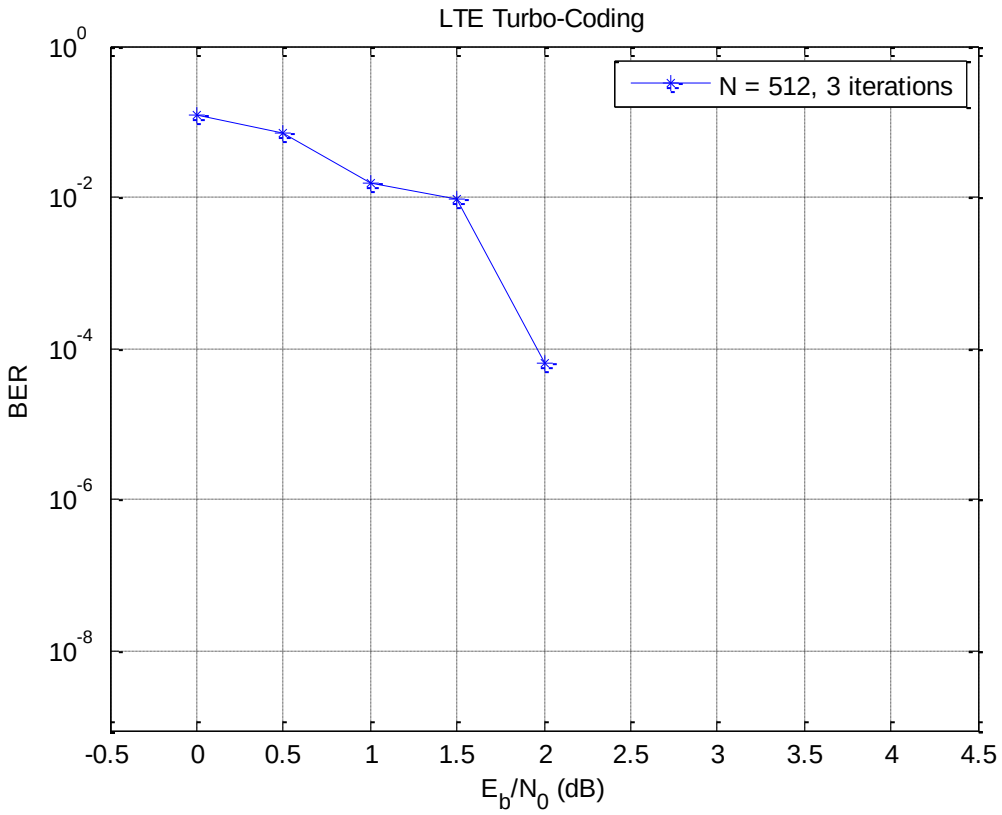
Now we are going to run other simulations using lower block code lengths such as 512 and 48 but using the same iteration values of the above graphs being (10, 6, 3, and 2) and compare the graphs to notice and comment on the effects of the change in parameters



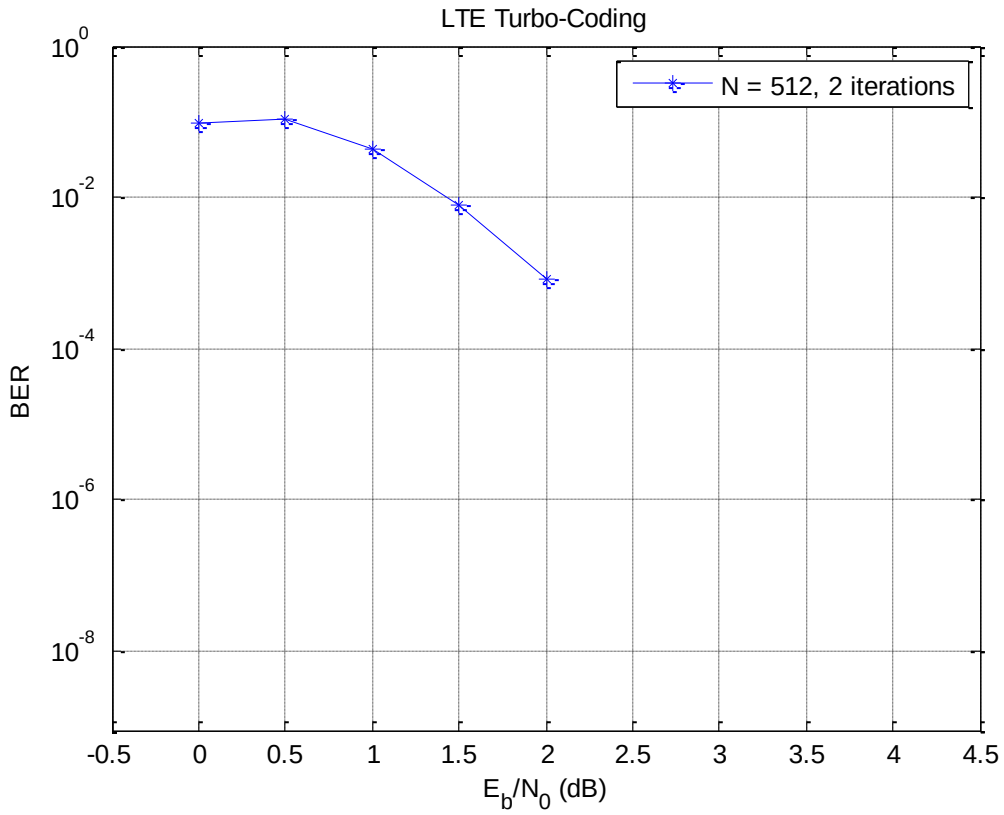
Here we see that the iteration is 10 but for a code block length of 512 bits and this simulation was short based on time description. The next graph is below but for the same code block length but different and lower iteration of 6 and we have



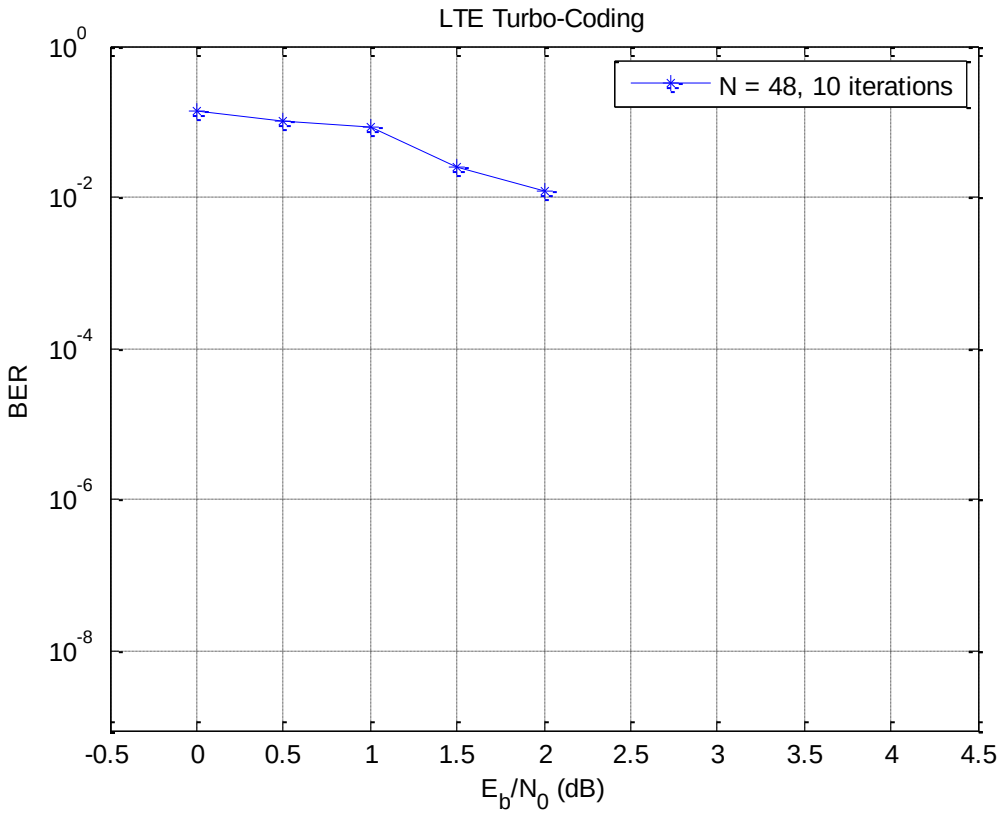
And as we see the end of the graph starts slowly drifting to the right once again showing the increase in error rate as the iteration begins to drop



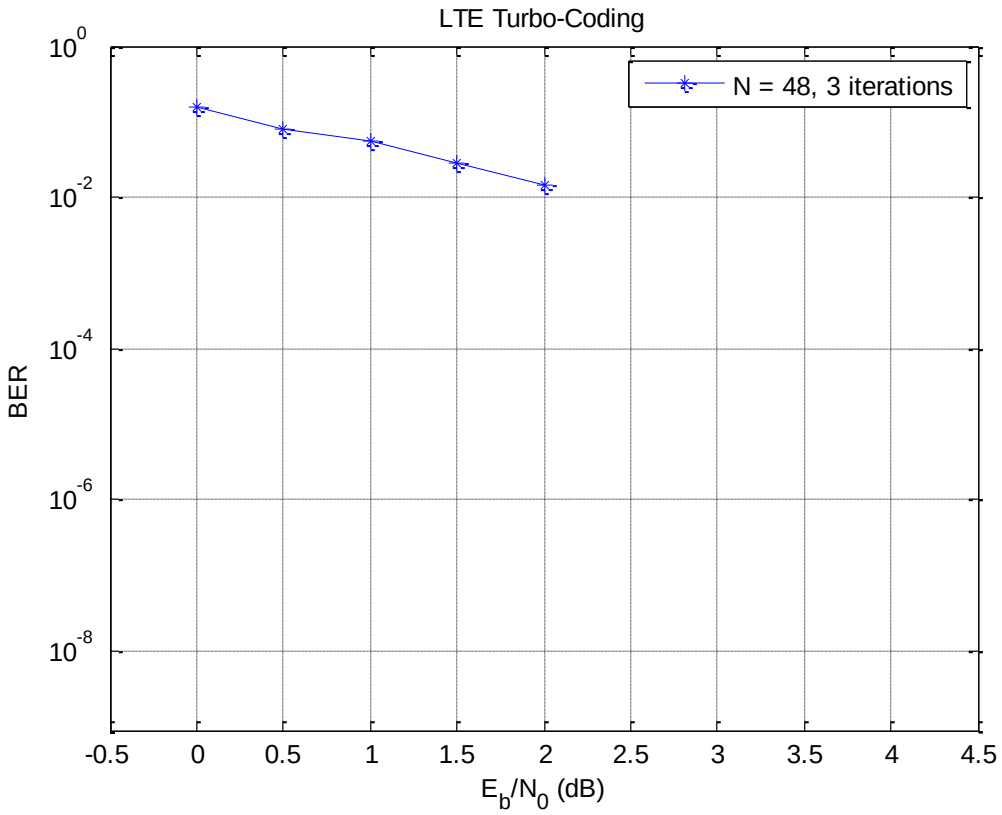
Next we see the iteration of 3 for a code block length of 512 bits and the end point of the graph still shifts to the right therefore showing an increase in error rate as the iteration drops



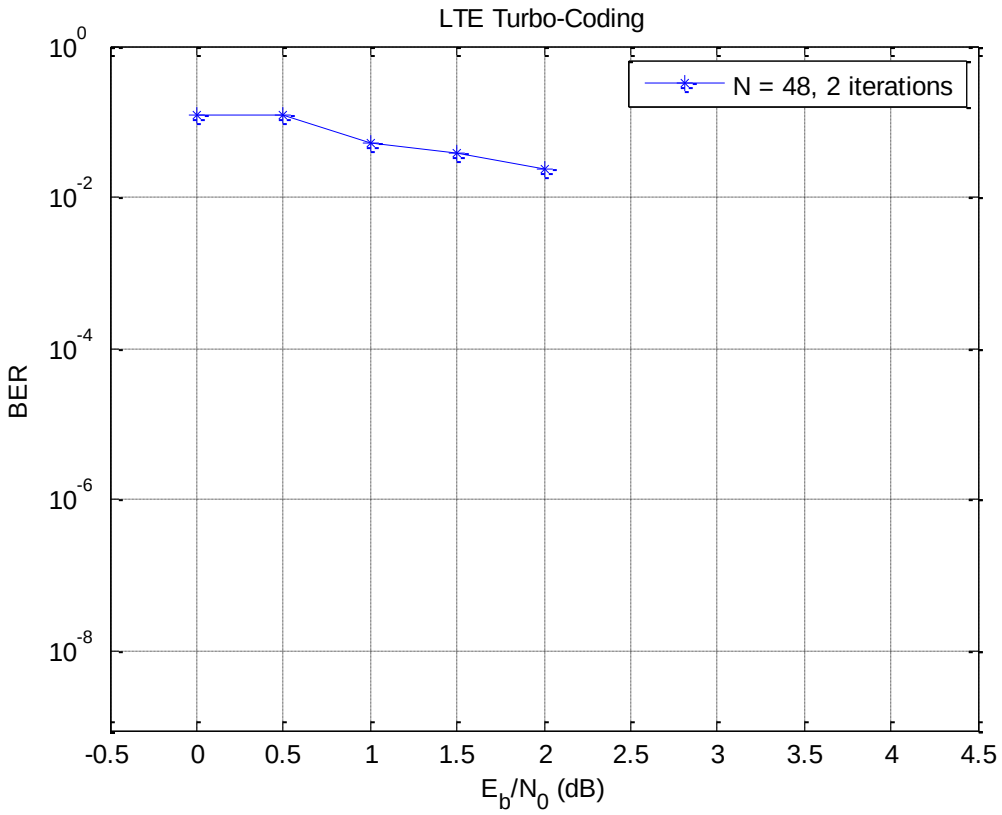
Here we have a low iteration of 2 and number of code block length stays the same as 512 the same result as above occurs causing it to shift to the right more



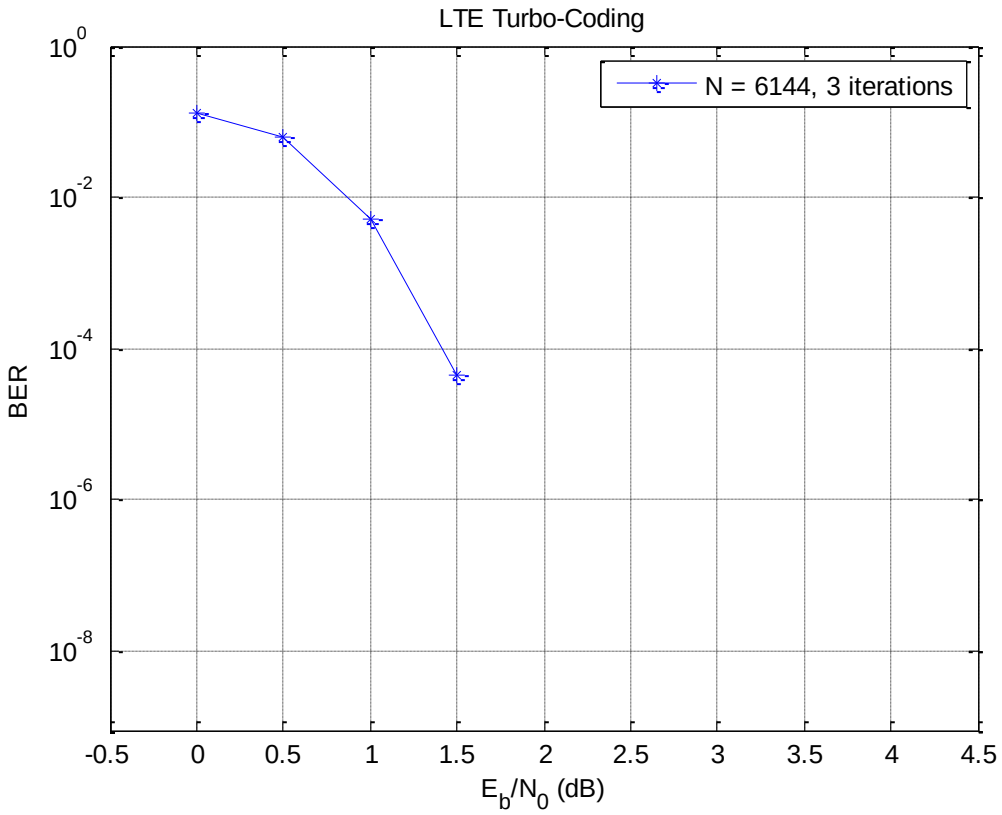
Here we have a code block length size of 48 bits which is very low and number of 10 iterations



Here the iteration is as low as 3 and the code block length too is low 48



Here the iteration is 2 and the code block length size is 48. Therefore we can generally say that the for the same size of code block, we notice from the graph that as the iteration is increased the bit error rate performance to increases. To show that as the block lengths size is increased the bit error rate performance increases too ill show a graph of size of code length 6144 and iteration was 5 it is seen below



And we can see that this bit error rate performance is really heightened as it is more to the left than simulations from lower block sizes and this took a long time. Therefore also increase in block length an iteration size increases the amount of time the simulation takes

CHAPTER 6 CONCLUSION

In this paper, turbo code, a very powerful error correcting coding scheme, which is formed by the parallel concatenation of two recursive non-systematic convolutional codes, is presented. The conclusions drawn from the work carried out: The simulation results clearly depicts that the code has the capability of reaching very low bit error rates at even small signal to noise ratios with an increase in the iterations. The main objective of increasing the iterative process or number of iterations is to further reduce bit errors. However, the evaluation of the number of iterations necessary for optimal results has proven to be a difficult task. In general, the increase in iterations improve performance, however, it is based on the channel conditions and the size of the input frame or block lengths. From the simulation results we observed that for a fixed turbo encoder the performance of the decoder improves as the data frame size is increased. This implies that the data frame size or size of block length is also an important factor in the performance of parallel concatenated convolutional codes: turbo codes and as the size of block length is increased the code gives better performance, which on the other hand is not the case with the conventional Viterbi decoder. Note also that the increase in block lengths gives a better performance at the expense of a quicker simulation and from the plots of simulation viewed we see that the error reduces as a single graph tends to go to the left. Although interleaver design is crucial in the performance of the turbo code, unfortunately there is no analytical closed form expression that describe which interleaver design will perform better than the other one and minimum distance and its multiplicity of turbo codes are used to estimate the error performance in high SNR region. The factor of iterations also become irrelevant at a point such that the error calculation tends to zero, meaning the error in the information bits Is corrected to a point that there is no more error so other iterations after this point produce no effect or result what so ever. Also note that there are specifications for the block length size when inputting your desired parameters and this data must be inserted accurately for good error correction or good simulation to be induced.

The turbo codes in this paper is used largely in the world now and is speedily on the rise we can see it in wireless communication (mobile phones), satellite and deep space communication, LTE (4^{th} generation wireless communication) WI-max and a lot of other advanced technological areas and it is has been seen as very important technology for advanced technologies and future works.

REFERENCES

[1]T. Cover and J. Thomas, Elements of Information Theory. New York: Wiley Interscience, 1991.

[2]C. E. Shannon, “A Mathematical Theory of Communication”, Bell System Technical Journal, Vol. 27, pp. 379-423 (Part One), pp. 623-656 (Part Two), October 1948.

[3]S. Lin and D. J. Costello, Jr., “Error Control Coding: Fundamentals and Applications”, Prentice-Hall, 1983.

[4]S. B. Wicker “Error Control Systems for Digital Communications and Storage” Englewood Cliffs: Prentice-Hall, 1995.

[5]J. G. Proakis, “Digital Communications”, 3rd Edition McGraw Hill, New York, 1995. [6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv “Optimal Decoding of Linear Codes for Minimizing Symbol error Rate”, IEEE Transactions on Information Theory IT-20, pp. 284-287, March 1974.

[7]J. G. Proakis and M. Salehi “Communication Systems Engineering”, Prentice Hall, 1994.

[8]C. Berrou, R. Glavieux, and P. Thitimajshima, “Near Shannon limit error correcting coding and decoding”, in Proc. ICC 1993, pp. 1064-1070.

[9]C. Berrou, and A. Glavieux “Near Optimum Error Correcting Coding and Decoding: Turbo-Codes”, IEEE Transactions on Communications, Vol. 44, No.10, pp. 1261-1271 October 1996.

[10]J. Hagenauer and P. Robertson “Iterative Turbo Decoding of Systematic Convolutional Codes with MAP and SOVA algorithms”, in Proc. ITG Conference on Source and Channel Coding, Munich, October 1994.

[11]J. Hagenauer, E. Offer and L. Papke “Iterative Decoding of Binary Block and Convolutional Codes”, IEEE Transactions on Information Theory, Vol. 42, No.2, March 1996.

[12]B. Vucetic, and J. Yuan “Turbo Codes Principles and Applications”, Kluwer Academic Publishers, 2000. 54

[13]J. Hagenauer and P. Hoeher “A Viterbi Algorithm with Soft-Decision Outputs and its Applications”, Proc. Globecom pp. 1680-1686 November 1989.

[14]C. Heegard and S. B. Wicker “Turbo Coding”, Kluwer Academic Publishers, 1999.

[15]J. Banks, J. S. Carson, II, B. L. Nelson “ Discrete-Event System Simulation”, Second Edition, Prentice Hall, 1996.

[16]T. S. Rappaport “Wireless Communications Principles & Practice”, Prentice Hall, 1996.

[17]S. Crozier, J. Lodge, P.Gunand and A. Hunt “Performance of Turbo-Codes with Relative Prime and Golden Interleaving Strategies” Sixth International Mobile Satellite Conference (IMSC’99), Ottowa, Canada, pp. 268-275, June, 1999.

[18]ITU Document SG15/Q4 BO-087: “Comparison of simulation results for different coding techniques (Uncoded, Reed-Solomon plus Trellis and Reed-Solomon plus Parallel Concatenated Convolutional Codes) for G.992.1.bis and G.992.2.bis”. Boston, MA 10-14 May 1999

[19] Fundamentals of turbo codes B Skylar - Prentice Hall, Mar, 2002 - itglitz.in

[20]

http://www.mathworks.com/help/comm/examples/parallel-concatenated-convolutional-coding-turbo-codes.html

[21] http://www.scholarpedia.org/article/Turbo_code