CERTIFICATE

DELHI TECHNOLOGICAL UNIVERSITY,
(Formerly Delhi College of Engineering)
BAWANA ROAD, DELHI – 110042
Department of Electronics & Communication Engineering

This is to certify that the thesis titled **"Performance Analysis of Turbo Codes in Fading Environments"** submitted by Mudita Chandra, Roll. No. 2K12/SPD/11, in partial fulfillment for the award of degree of Master of Technology in Signal Processing & Digital Design at **Delhi Technological University, Delhi**, is a bonafide record of student's own work carried out by her under my supervision and guidance in the academic session 2013-14. The matter embodied in dissertation has not been submitted for the award of any other degree or certificate in this or any other university or institute.

(Dr. S. K. Soni)

ASSOCIATE PROFESSOR
Department of Electronics & Communication Engineering
Delhi Technological University,
(Formerly Delhi College of Engineering)
Delhi -110042

ACKNOWLEDGEMENT

I express my sincere gratitude to my guide **Dr. S.K. Soni**, Associate Professor, Dept. Of Electronics & Communication Engineering for giving valuable suggestion during the course of the investigation, for his ever encouraging and moral support. I truly admire his depth of knowledge and strong dedication to students and research . His mastery of any topic is amazing, but yet he is such a humble and down to earth person. I am glad that I was given opportunity to work with him. He surely brings out the best in his students.

I am greatly thankful to **Prof. Rajiv Kapoor, Professor and Head,** Department of Electronics & Communication Engineering, entire faculty and staff of Electronics & Communication Engineering for their, continuous support, encouragement and inspiration in the execution of this "**thesis"** work.

I would like to thank my parents Dr. Atul Chandra and Mrs Anju Chandra and my sister Ruchira Chandra who bestowed upon me their grace and were source of my inspiration and encouragement.

I am thankful to almighty God for his grace and blessings.I am also thankful to my classmates for their support and help.

Mudita Chandra

ABSTRACT

Turbo codes belong to the class of convolutional error correcting codes used in wireless data transmission applications like mobile communications, deep-sea communications, satellite communications and other power constrained fields. Turbo Decoder consists of concatenation of decoders based on Maximum a posteriori algorithm. Turbo decoding is an iterative principle where the constituent decoders pass extrinsic information to each other with an interleaver or deinterleaver in between. The performance of turbo codes approaches the Shannon limit making them most sought after in current wireless applications. Simulations of Turbo code performance under different parameters and different situations like number of iterations, different encoder structures, interleaver types, puncturing has been done. The performance of turbo codes was simulated in Rayleigh and Rician fading channels. The turbo codes have shown a considerably remarkable performance with variations in parameters and environments.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

The Shannon's channel coding theorem draws one of the predictions that a noisy channel can be converted into an error free channel by using a suitable coding technique and mathematically showed that the channel capacity[15], a theoretically maximum possible code rate could be achieved when transmitting information through a noisy channel. The modulation techniques that are traditionally used doesn't achieve this rate of transmission but however combining it with some suitable error control coding, the information can be transmitted and received which can achieve the performance close to the channel capacity. A class of forward error correcting codes which have been able to achieve this is Turbo Codes. Turbo decoders are based on soft decision in which decoding is done through iterative exchange of the extrinsic information from one decoder to another. The constituent encoders that are used in turbo encoding part are the recursive convolutional encoders. The decoders used in are Maximum A Posteriori probability (MAP) decoders. These MAP decoders are soft decision decoders which give improve performance than the earlier hard decision based viterbi decoders. Turbo codes [3],[4],[5] use concatenated MAP decoders. The concatenation can either be done in a serial or parallel fashion. The key to turbo decoder operation is the "message passing" between them. The message here is the extrinsic information which is basically the improved version of the apriori probability. Another important component of turbo decoders is the use of interleaver. This interleaver is a scrambler which permutes the information in such a way that errors spread out. The decoding operation is done by turbo decoder for several iterations and the probability of error decreases with each subsequent iteration. With higher number of iterations Turbo Codes are capable of achieving performance close to the Shannon's theorem. With such a remarkable performance turbo codes have been used in a variety of applications. Universal Mobile Telecommunication Systems (UMTS) standardized by the Third Generation Partnership Project (3GPP) and CDMA and CDMA-2000 standardized by International Telecommunication Union (ITU) both have adopted turbo codes to be used in high speed wireless data transmission. Turbo codes are being used in 3G

mobile communications, deep-sea communications, satellite communications and other power constrained fields. In this thesis performance of turbo codes close to Shannon limit is shown by simulations. Also the performance of turbo code has been analyzed for the encoder structures, puncturing, and for wireless channel fading and iterations.

1.2 Motivation for present work

In the present scenario, the field of wireless communications is progressing tremendously. However for any digital communication system getting the replica(if not exact then at least approximate of it) of whatever is transmitted is an important concern. Turbo decoders have shown remarkable BER performance close to Shannon limit. Also whenever a signal passes through a channel it experiences fading. So simulating turbo codes performance close to Shannon limit and analyzing its performance in fading environments [6], [7], [9] appeared quite interesting to me and motivated me to carry out this work.

1.3 Organization of thesis

Thesis is organized as follows: In chapter 2 convolutional coding which is a type of channel coding has been discussed. Analysis of Systematic convolutional and recursive convolutional codes has been done. The chapter ends with the explanation about Maximum Likelihood decoding by Viterbi decoders. The performance of Viterbi decoders have been shown through simulations.

Chapter 3 deals with the Maximum Posteriori Algorithm and soft decoding done by it. The proof of estimation of soft Log likelihood ratio values is done by computation of branch metrics, forward metrics and backward metrics. The performance of MAP decoder has been shown through simulations.

Chapter 4 deals with the concept of turbo codes. Turbo encoders, Turbo decoders and passing of extrinsic information between the concatenated decoders has been discussed and analyzed. The performance of Turbo codes has been analyzed and through simulations improved performance with increasing number of iterations has been shown. Performance approaching close to Shannon limit has also been shown.

Chapter 5 deals with an introduction of various channel fading. Their probability density functions plots have been developed with the help of simulations.

Chapter 6 deals with the performance analysis of turbo codes on various factors like encoder structures and puncturing. Performance of turbo codes has been analyzed in various fading environments like Rayleigh and Rician fading. The performance has been shown through simulations.

Chapter-2

CONVOLUTIONAL CODING

2.1 Coding Theory

In Information theory the most important application is coding. Coding can either be done at the source end or can be done at the transmitter end. The first one is the source coding and the latter is the channel coding. The basic aim of any coding is to minimize the errors and effects of distortion and noise. In the figure below is illustrated a typical digital communication system which shows where coding comes into picture.

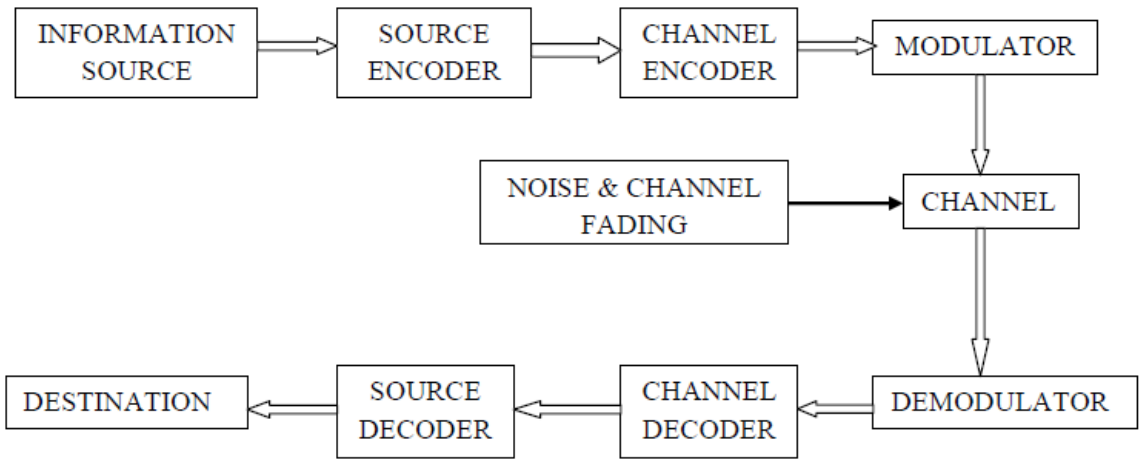

Fig 2.1: Block Diagram of a Digital Communication System

2.2 Source Coding

Source Coding is basically an analog to digital conversion. It codes the input message from the information source by removing any redundant information present in the message. This is accomplished by mapping. The mapping should be one to one i.e. if the given message is $[m_1,m_2 \ldots m_n]$ then the encoded codeword $[C_1,C_2,\ldots\ldots C_N]$ should be such that for each message there is only one encoded message.

2.3 Channel Coding

When the digital signals (the input message which has been suitably converted into digital form) is finally transmitted it passes through the communication channel. This channel introduces several impairments such as noise, fading, attenuation etc. These impairments corrupt the underlying message and at the receiver we don't get what is transmitted. So there is a great need to mitigate the effect of the errors caused by the channel before the message is passed through it.

A channel coded digital communication system is represented as

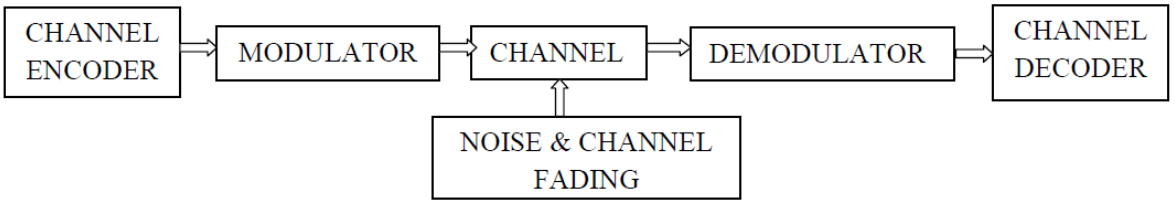

Fig 2.2: Block diagram of Channel coding

The major goal of channel encoding is to modify the binary stream before transmission in such a way that errors in the received signal can both be detected and corrected. For this to take place the channel encoding basically introduces some kind of redundancy or extra bits to the input bit stream. So at the output of channel encoders we get "code words" which are larger than the input data words.

There are two important classes of codes-block codes and the convolutional codes. In block codes [14] the bit stream is partitioned into blocks of code words. A message vector (or data word) which is k bit long is transformed into a longer code vector (or code word) of n bits. For a k bit message vector there can be total possible 2^k distinct data words which are referred as k-tuples (k bit long sequences). The encoding procedure assigns to each of the 2^k message one of the 2^n n-tuples. A block code represents a one to one assignment whereby 2^k data words are mapped to a new set of 2^k code words which are n bits long.

Hamming distance is an important aspect in the detection and correction of errors. It is defined as the distance between two equal length binary code words defined as the number of bit positions in which the two codes differ. For example, the distance between 000 and 001 is 1 and between 000 and 111 is 3. As per the coding theory the number of errors that can be detected and corrected is related as:

If $d_{\min}$ is the minimum hamming distance between the code words then up to $d_{\min} - 1$ errors can be detected. If $d_{\min}$ happens to be an even number then up to $d_{\min} - 1/2$ errors can be corrected. If $d_{\min}$ is odd then up to $\frac{d_{\min}}{2} - \frac{1}{2}$ errors can be corrected. Thus it can be inferred that it is desirable to have an increased minimum hamming distance. And this is what exactly repetitive codes do. They increase the hamming distance.

2.4 Convolutional Codes

In block encoding as described before, the input to the encoder is a data word of k bits long and the output of the encoder is a codeword of n bits long. Thus the ratio k/n is called the code rate which gives a measure of redundancy. Thus block codes [12], [14] are described by integers k an n. A convolutional code differs from the block codes in the way that the convolutional encoders have memory. The convolutional code are described by integers n,k and K. The integer K is a parameter known as constraint length which represents the number of stages (or flip flops) of the encoding shift register. Thus, in convolutional codes, the error correcting capability is achieved by introducing memory into the system. The name "convolutional" comes from the fact that the code word or the transmitted sequence is generated by convolving the source sequence or the data word with a fixed binary sequence.

As shown below in basic block of convolutional encoding of fig 2.3 the input message sequence is denoted by $m=m_1,m_2,\ldots,m_i,\ldots..$,where each m_i represents bit value at the i^{th} time index. Each m_i is equally likely to have value as 1 or 0. Thus each m_i is independent or we can say that knowledge of the bit value at any instant doesn't infer the knowledge at any other instant. The encoder transforms input bit sequence into a codeword sequence U=G(m).

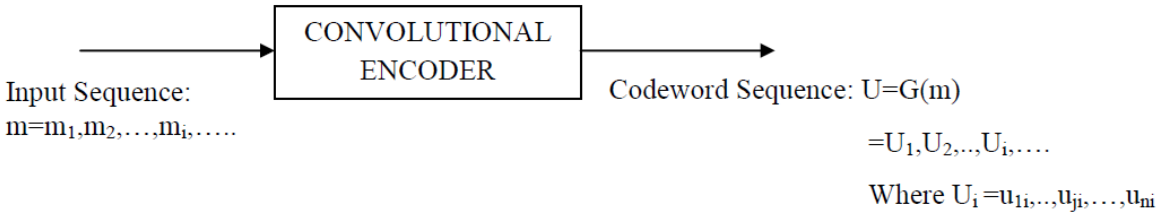

Fig 2.3: Convolutional Encoding

A key feature of the convolutional encoder is that given a k bit long data word in the input sequence m does not uniquely defines its associated n bit long codeword within U because the encoding of each k bit data word is not only a function of k bits but is also a function of K-1 input k bits that precede it. So the output codeword is not independent in contrary to the input bit sequence.

A general convolutional encoder can be structurally shown as

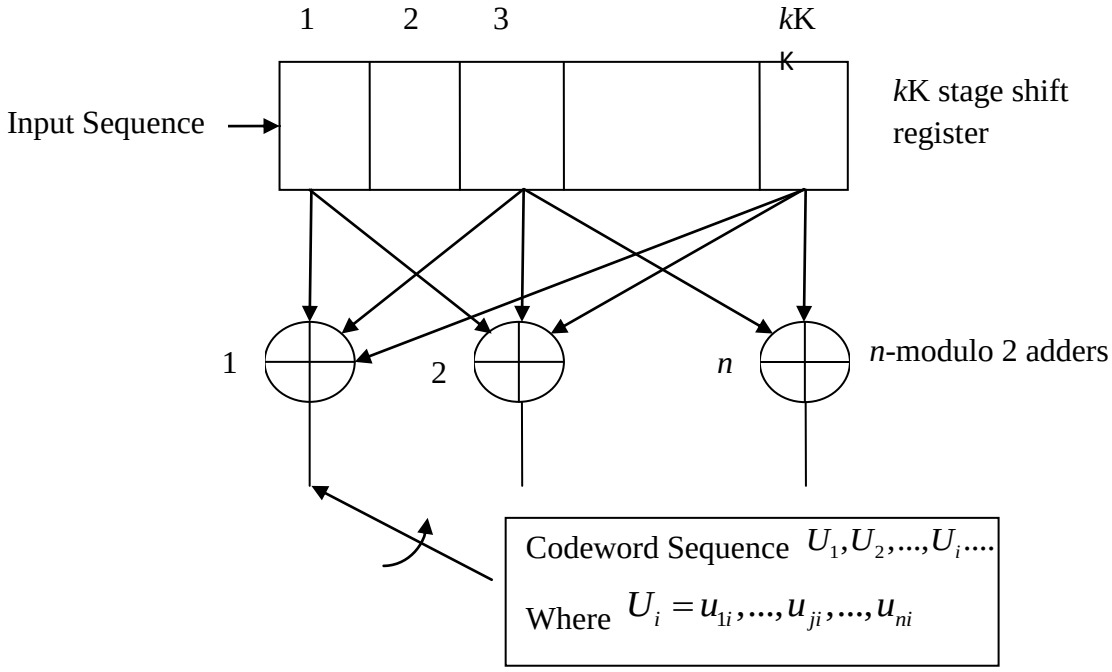

Fig 2.4: Structure of Convolutional Encoder

It consists of *k*K shift registers and n modulo-2 adders. The constraint length represents the number of k bits shifts over which a single input bit influences the encoder output. At each unit of time, k bits are shifted into the first k bits of the register; all the bits in the register are shifted k bits to the right, and the output of the n adders are sequentially sampled to yield the

binary codeword output. Clearly Code rate is k/n where k<n. The most commonly used binary convolutional encoders considers k=1. Clearly in these encoders the message bits are clocked in or shifted into the encoder one bit at a time. In generalized way we can say for k=1 encoder, at the i^{th} unit of time, input message bit m_i is clocked into the first stage of the shift register and rest all previous shift register bits are shifted one step to the right. The outputs of n adders are sequentially sampled and their modulo 2 addition is done.

2.5 Convolutional Encoder Representation

To describe a convolutional code, encoding function or the generator polynomial G(m) must be specified so that output can easily be determined for any given input. The generator polynomial basically specifies how the output bits of the encoder depend on the contents of the shift register. One way to represent it is by the way of connection vectors or tap connections one for each of the modulo-2 adders. A one in the i^{th} position of the vector shows that there exists a tap connection between the shift register and the modulo-2 adder, and a zero in a given position signifies an absence of any connection to the modulo-2 adder. The figure below is shown a rate ½ convolutional encoder

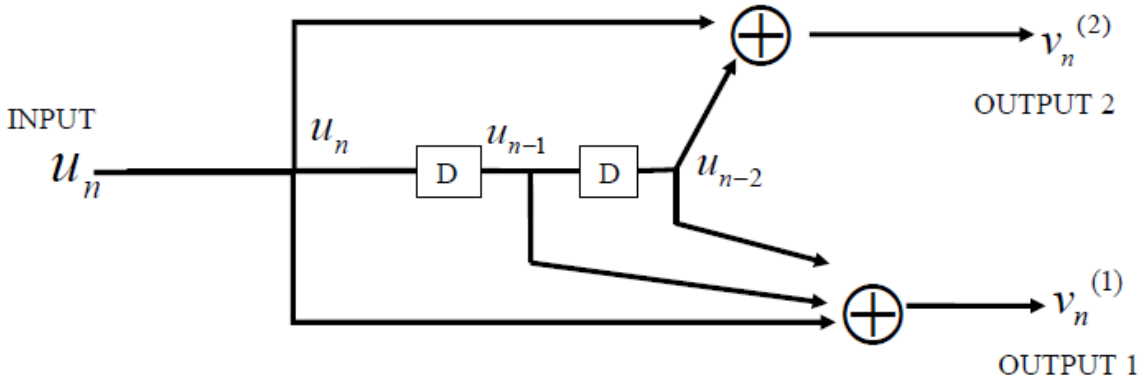

Fig 2.5: Rate ½ Convolutional Encoder

For the encoder example above the connection vector g_1 for output 1 and g_2 for output 2 is as follows:

$g_1 = 1\ 1\ 1$

$g_2 = 1\ 0\ 1$

Now consider a message vector m=101 has to be convolutionally encoded with the above encoder. The three message bits are clocked in at time instants t0,t1,t2. Initially both the flip flops are cleared to all zero state. Subsequently we will get the output as 11 10 00 10 11.

2.5.1 State Diagram

A convolutional encoder is a finite state machine since it has a memory. The term finite here refers to the fact that there is only finite number of unique states that the machine can have. The convolutional encoder can be regarded as mealy machine because the knowledge of the present input and the state of the system can completely determine the output of the system. The state provides some knowledge of the past signaling events and the restricted set of possible outputs in the future. A future state is restricted by the past state. The state diagram of the encoder shown in the figure can be represented as

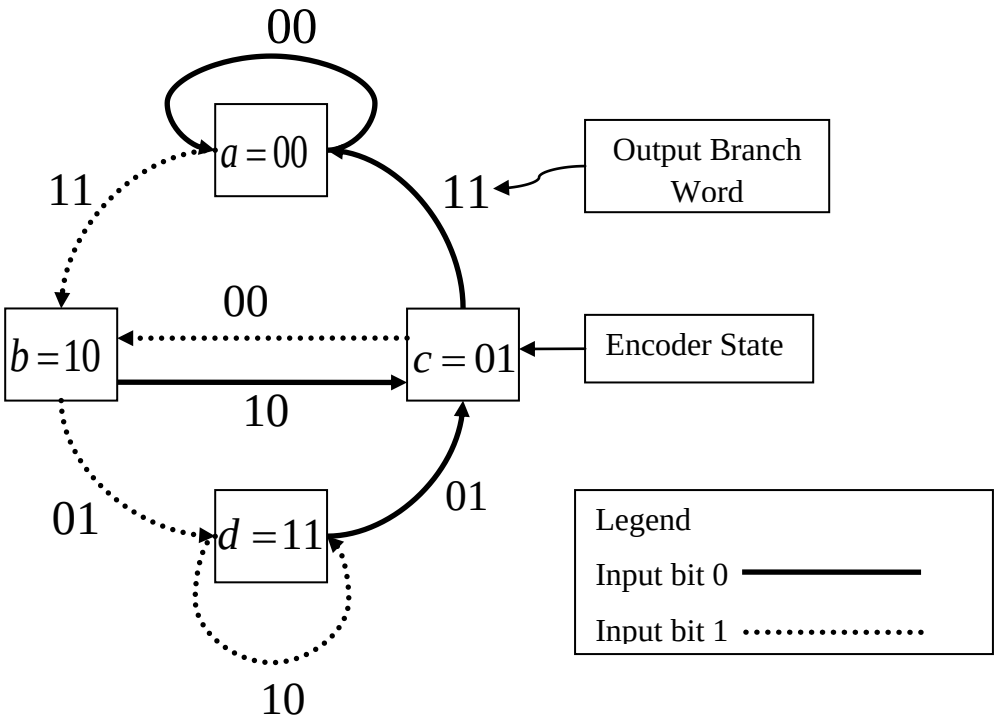

Fig 2.6: State Diagram

As seen in the state diagram above, on each line both the input causing the state transition and the corresponding output are specified.

2.5.2 Trellis Diagram

The state diagram does not have a dimension of time so it cannot be used for tracking the transitions of the encoder with time. The trellis diagram is a manageable representation of the encoder which has a time axis. The convolutional encoders can be specified by a trellis diagram that can be obtained by specifying all states on a vertical axis and repeating this vertical axis along the time axis. Then each transition from one state to another state is denoted by a line connecting the two states on two adjacent vertical axes i.e a trellis diagram is nothing but a repetition of the state diagram along the time axis. There are 2^k branches of the trellis leaving each state and 2^k branches merging at each state. The trellis diagram for the rate 1/2 convolutional encoder of the fig 2.5 can be shown below as

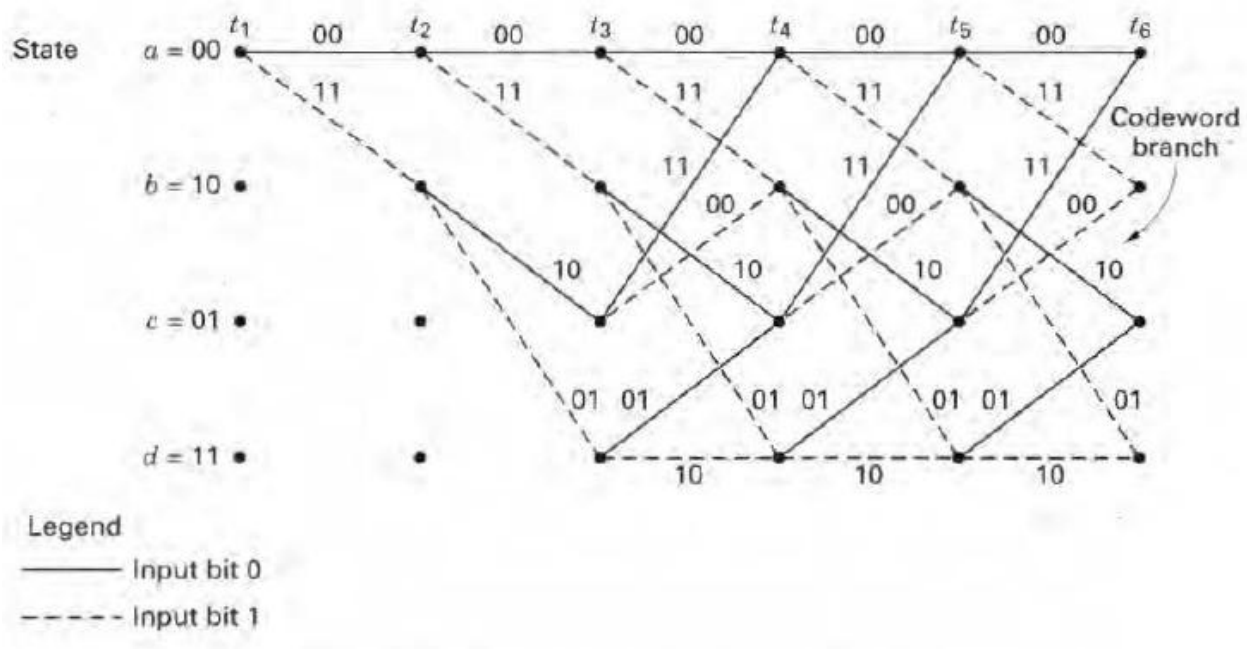

Fig 2.7: Trellis diagram

Here as we see the encoder may not terminate to the state with which it started. So for the initial state as $[00]$, the final state should also be $[00]$. So appending or padding the input with some bits is necessary. The necessary bits to append the input can be determined from the trellis diagram as shown in fig 2.8

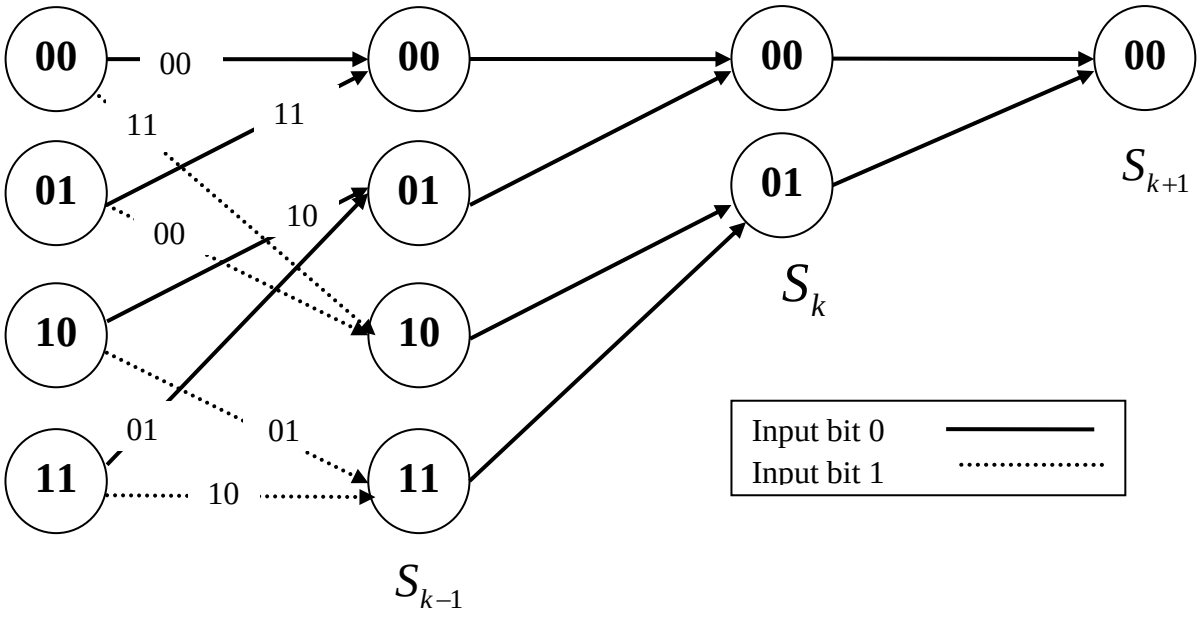

Fig 2.8: Tail Biting

When k bits are clocked into the convolutional encoder the then for time t=0,1…..k-1 the final state is S_{k-1}. This may be any of the four states the encoder has. So to bring it back to all zero state two extra zeros are sufficient as can be seen in the figure above.

2.5.3 Code word Output

For the ½ Rate convolutional encoder of fig 2.5 the following analysis is done to express the codeword and the rate. Here as we can see the input bits which are clocked in the encoder. The clock is not shown explicitly but is assumed to be present implicitly. The nth input bit is u_n i.e the input at nth clock is u_n.

Then we can easily see

$$v_n^{(2)} = u_n \oplus u_{n-2} \quad \text{….(2.1)}$$

$$v_n^{(1)} = u_n \oplus u_{n-1} \oplus u_{n-2} \quad \text{…..(2.2)}$$

State of encoder: $\lfloor s_1 s_2 \rfloor$ => This corresponds to 2 bits

Let k bits of input are clocked in then input $\underline{u}$ is

$$\underline{u} = \lfloor u_0 u_1 .... u_{k-1} \rfloor$$

Let the initial state of encoder be $\lceil 0\,0 \rceil$

Then state after k bits of input have been clocked in is

$s_1 = u_{k-1}$ and $s_2 = u_{k-2}$

So $\lceil s_1\, s_2 \rceil = \lceil u_{k-1}\, u_{k-2} \rceil$

This final state may not be the state with which we started. So for the initial state as $\lceil 0\,0 \rceil$ the final state should also be $\lceil 0\,0 \rceil$. So Tail biting is necessary. So the actual input for all zero state termination is $\lceil u_0 u_1 .... u_{k-1} 0\,0 \rceil = \lceil \underline{u}\, 0\,0 \rceil$ then

Output1: $\underline{v}^{(1)} = \left[v_0^{(1)} v_1^{(1)} .... v_{k-1}^{(1)} v_k^{(1)} v_{k+1}^{(1)} \right]$

Output2: $\underline{v}^{(2)} = \left[v_0^{(2)} v_1^{(2)} .... v_{k-1}^{(2)} v_k^{(2)} v_{k+1}^{(2)} \right]$

Then codeword output $\left[\underbrace{v_0^{(1)} v_0^{(2)}}_{time\,0} \underbrace{v_1^{(1)} v_1^{(2)}}_{time\,1} .....v_{k-1}^{(1)} v_{k-1}^{(2)} v_k^{(1)} v_k^{(2)} \underbrace{v_{k+1}^{(1)} v_{k+1}^{(2)}}_{time\,k+1} \right]$

Rate (for zero terminated case) $= \dfrac{k}{2k+4}$

But the actual designed rate $= \dfrac{1}{2}$

So there is some loss in rate in zero terminated case for small packet inputs

In general encoder might have μ flip flops and at any time instant we have only one input stream. If correspondingly we have m outputs then the designed rate is $= \dfrac{1}{m}$

Zero Terminated rate(for k bit input) $= \dfrac{k}{mk+\mu}$ …..(2.3)

2.5.4 D-Transform

The flip flops or the memory elements that are shown in the encoder diagram are basically delay elements. Thus the generator polynomials can be written in the form of polynomials of these delay elements. Each polynomial describes the connection of the encoding shift register to that of the modulo 2 adder much the same way the connection vector does. The coefficient of each term of the polynomial is either 1 or 0, depending on whether a connection exists

between the shift register and the modulo 2 adder in question. The generator polynomial also represents the impulse response that is the response of the encoder to a "single" bit that moves through it. Hence each term of the generator polynomial represents unit delay transform of the impulse response. Using this D-Transform the output codeword can be represented by the analysis as shown below

$$u(D) = \sum_{i=1}^{K} u_i D^i$$

$$v^{(1)}(D) = u(D)G^{(1)}(D)$$

$$g^{(1)} = 101$$

$$G^{(1)}(D) = 1 + D^2 \qquad \text{.....(2.4)}$$

$$g^{(2)} = 111$$

$$G^{(2)}(D) = 1 + D + D^2 \qquad \text{.....(2.5)}$$

$$v^{(2)}(D) = u(D)G^{(2)}(D)$$

Generator matrix

$$\left[G^{(1)}(D) G^{(2)}(D) \right]$$

Output: $\left[v^{(1)}(D) v^{(2)}(D) \right] = u(D) \left[G^{(1)}(D) G^{(2)}(D) \right]$

2.6 Recursive Systematic Convolutional Encoders

A systematic convolutional code is one in which the input k tuple appears as a part of the output branch word n tuple associated with that k tuple. A recursive systematic convolutional encoder is the one in which the previously encoded information bits are continually fed back to the encoder's input. They are like IIR filters and are basic building blocks for a turbo code. Considering the rate ½ convolutional encoder of fig 2.5 its generator polynomial and outputs as mentioned was

$G(D) = \left[G^{(1)}(D) G^{(2)}(D) \right]$ or

$$G(D) = [1 + D^2 \; 1 + D + D^2]$$

Outputs are $v_n^{(1)} = u_n \oplus u_{n-2}$ and

$$v_n^{(2)} = u_n \oplus u_{n-1} \oplus u_{n-2}$$

The encoder considered here is a Non systematic encoder. Now for systematic encoder we need a 1 in generator polynomial $G(D)$ i.e.

$$G^{(s)}(D) = [\frac{1 + D^2}{1 + D^2 + D} \; 1]$$

Then $v_n^{(2)} = u_n$

And $v_n^{(1)}$ represents a recursive or feedback form similar to an IIR filter. This $v_n^{(1)}$ can be represented in a recursive equation of the form

$$v_n^{(1)} + v_{n-1}^{(1)} + v_{n-2}^{(1)} = u_n + u_{n-2} \qquad \ldots(2.6)$$

To implement this systematic generator polynomial $G^{(s)}(D)$ the encoder structure should be like that given below

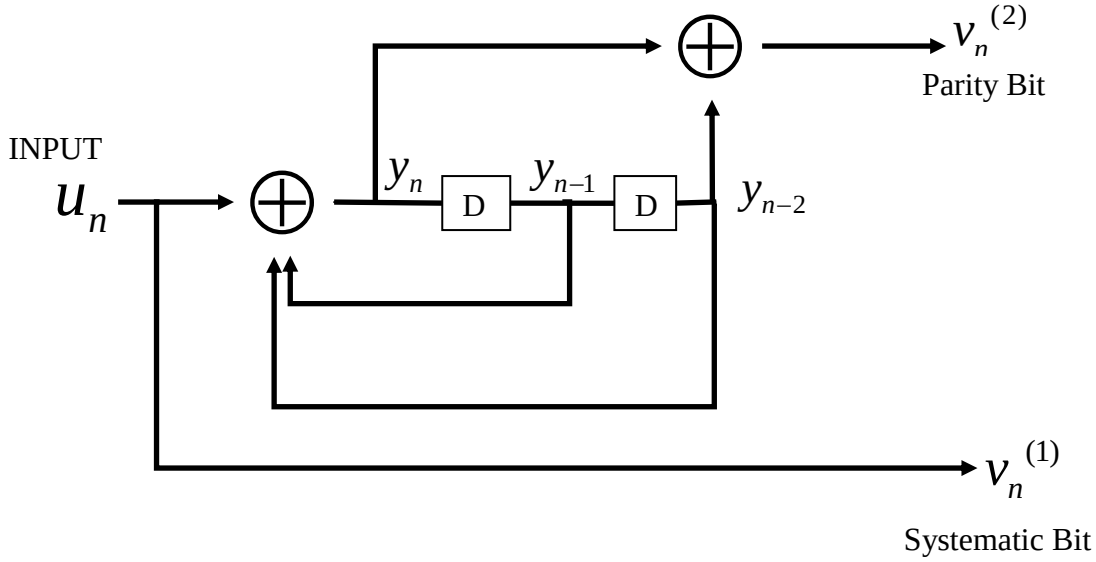

Fig 2.9: Recursive Systematic Convolutional Encoder

The above is the most commonly used RSC encoder. Its trellis diagram is as shown in the following figure

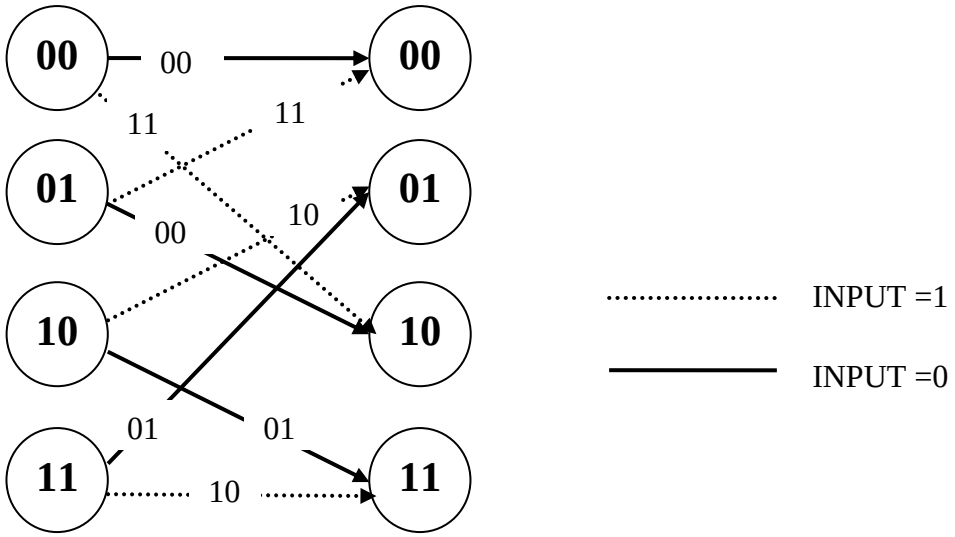

Fig 2.10 Trellis Diagram of Recursive Systematic Convolutional Encoder

Again as we can see the final stage of the encoder may not be the same as the one with which it started. So to terminate in the initial all zero state tail biting [13] or appending the input with some extra bits becomes necessary. The tail biting is demonstrated as shown below

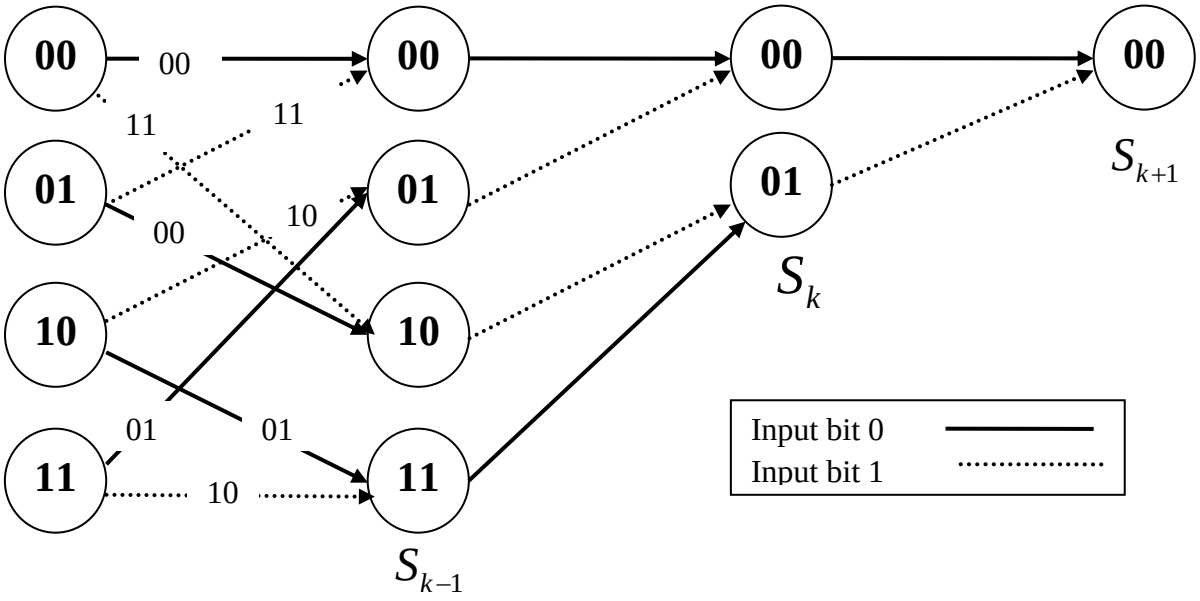

Fig 2.11: Tailbiting of Recursive Systematic Convolutional Encoder

As can be seen from above padded bits are not 00 as was with the case of the trellis diagram of fig 2.8. Rather the appended or tail bits here depend on the state at S_{k-1}

2.7 Decoders

Convolutional decoding is done after the demodulator as was shown in the block diagram of the digital communication system. Basically the aim of the decoding is to get the original information sequence back. For this decoding can either be hard decision decoding or soft decision decoding.

Hard Decision decoding: We will compare this received sequence with all permissible sequences and pick the one with the smallest Hamming distance (or Bit disagreement)

Soft Decision decoding: Here we do correlations and the sequences which give best correlations are picked up.

2.7.1 Maximal Likelihood Decoding

The maximal likelihood concept[4] is a common sense way of making decisions when statistical knowledge of possibilities is given. In simple binary modulation there are only two possible signals say $s1$ and $s2$. So given the received sequence as Z the binary maximum likelihood decision that s1 was transmitted meant

$p(Z \mid s1) > p(Z \mid s2)$ Otherwise $s2$ was transmitted.

In convolutional code there is memory. So k branch words in trellis diagram corresponding to message/codeword are a member of 2^k paths. There in this context of maximum likelihood the decoder chooses a particular branch word if its likelihood is greater than all other possible transmitted sequences. Such a decoder which minimizes the error probability when all the transmitted sequences are equally likely is known as Maximum Likelihood decoder.

2.7.2 Viterbi Decoder

The Viterbi Convolutional Decoding Algorithm:

The viterbi algorithm [16] is based on maximum likelihood decoding. At each time t_i, the algorithm calculates a measure of similarity or distance between the received signal and all the trellis paths which enter the state at time t_i. At each stage there is a removal of that trellis path which could not possibly be the candidates for maximum likelihood [5] or minimum distance metric choice. When two paths enter the same state, the one having the best metric i.e. the

least metric is chosen. This path is called surviving path and for all the states the selection of surviving path is done. The decoder continues in this way and advance deeper into the trellis making decisions by eliminating the least likely paths.

Viterbi decoding Example

Let us consider a case where the transmitted code word was $\bar{c} = 00\ \ 00\ \ 00\ \ 00\ \ 00$. After passing through the channel it gets corrupted and finally the received code word is $\bar{r} = 01\ \ 00\ \ 01\ \ 00\ \ 00$. Now we have to decode $\bar{r}$ using this viterbi decoder

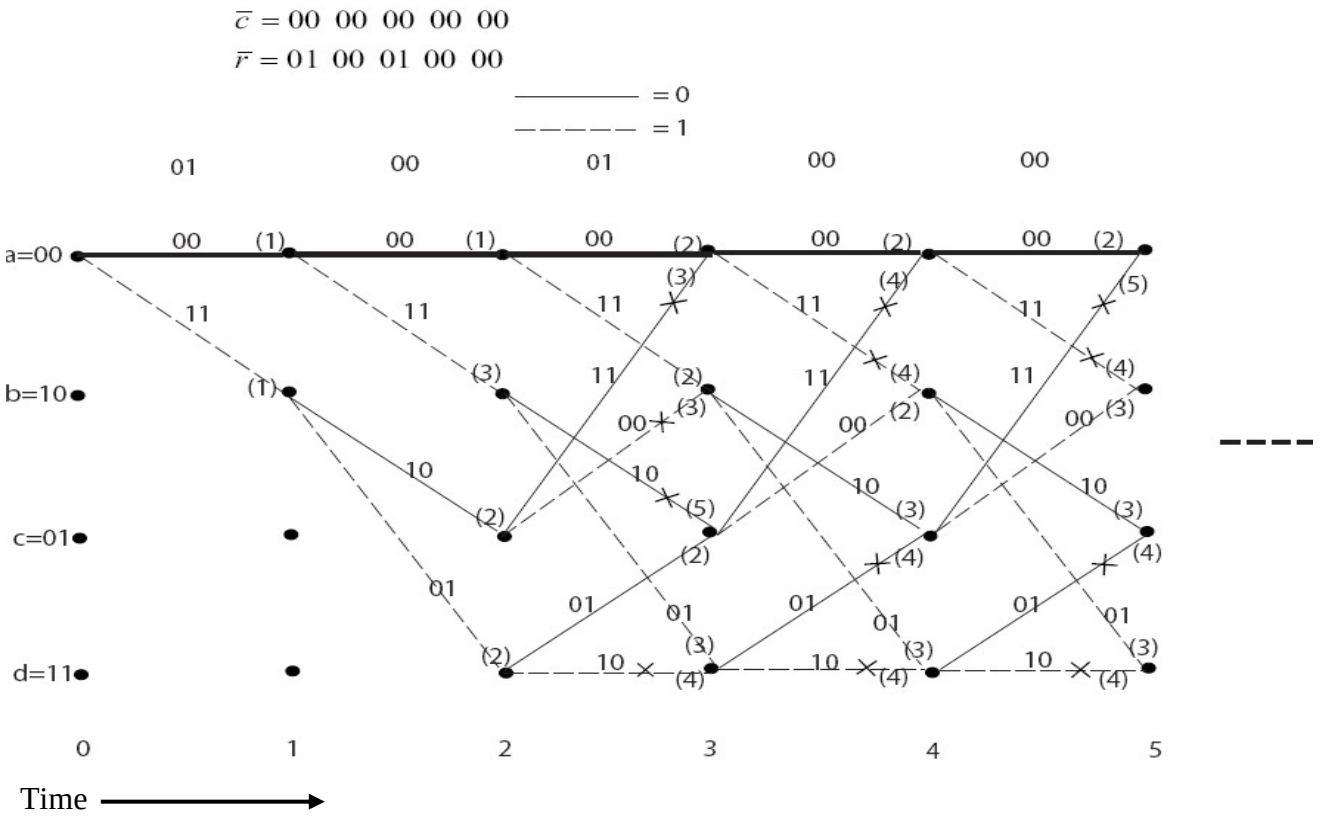

Fig 2.12: Viterbi Decoding

We start at time t_0 in the 00 state.With input 1 or zero and starting state as 00 there are only two branches leaving state 00 so only two branches are shown at time t_0. The full trellis evolves after time t_3. To begin with the decoding, at each tiime interval each branch is labelled with the hamming distance between the received code symbols and the branch word corresponding to the same branch from the encoder trellis. With the received sequence as

$\bar{r} = 01\ \ 00\ \ 01\ \ 00\ \ 00$ the hamming distance between the branch words and the received noisy code word is evaluated for each branch of the trellis. For example a state 00 transition yields an output codeword as 00. But we have received 01. The hammimg distance is 1 so label this branch after t_0 (before the beginning of t_1) as 1. Similarly a state 00 to 10 transition at t_0 with input 1 results in branch word of 11. But we have received 01 so the hamming distance is 1. So after t_0 (before t_1) the label 1 is put at state 10. Similar exercise is done for all the branches in the trellis. In summary the metric entered on the decoder trellis branch represents the difference between what was received and what "should have been" received had the branch word associated with that branch been transmitted. We continue labelling the decoder trellis branches in this way as the symbols are recived at each time t_i. We then find cummulative Hamming path metric of a given path at time t_i as the sum of the branch hamming distance metrics along that path up to time t_i. At each suceeding stage of the decodeer there will be two possible paths entering each state and the one with higher cummulative hamming path metric is eliminated.

Add Compare Select Computation

The decoding is basically an "Add Compare Select" computation. The decoding operation begins from the last stage i.e it progresses from backwards At the last stage of trellis(at t_5 here) the decoder selects the state having lowest distance metric. Here in this example the metric corresponding to state 00 is 2 ; state 01 is 3 ; state 10 is 3 ; and state 11 is 3. Out of all these the lowest metric is 2 so the decoder starts from state 00. So the decoded codeword is 00. Now at t_4 and at state 00 there are two branches entering with metrics 2 and 4. So the decoder selects the minimum i.e 2 out of them and travels through the branch corresponding it. So the decoded codeword is 00. Similar exercise is done till t_1. The decoded word is then 00 00 00 00 00. The trellis diagram is basically a pictorial representation of all the possible paths from the input to the the output. So basically the decoder starting from the last stage advances deeper and deeper into the trellis and make decisions about the input bit by eliminating all paths but one. The final decode path has been shown in bold in fig 2.12

The following figure shows BER error performance of Viterbi decoder in the presence of AWGN noise. The simulation was done for 50,000 bits in MATLAB. As can be seen from the plot, the error reduces with increasing energy per bit.

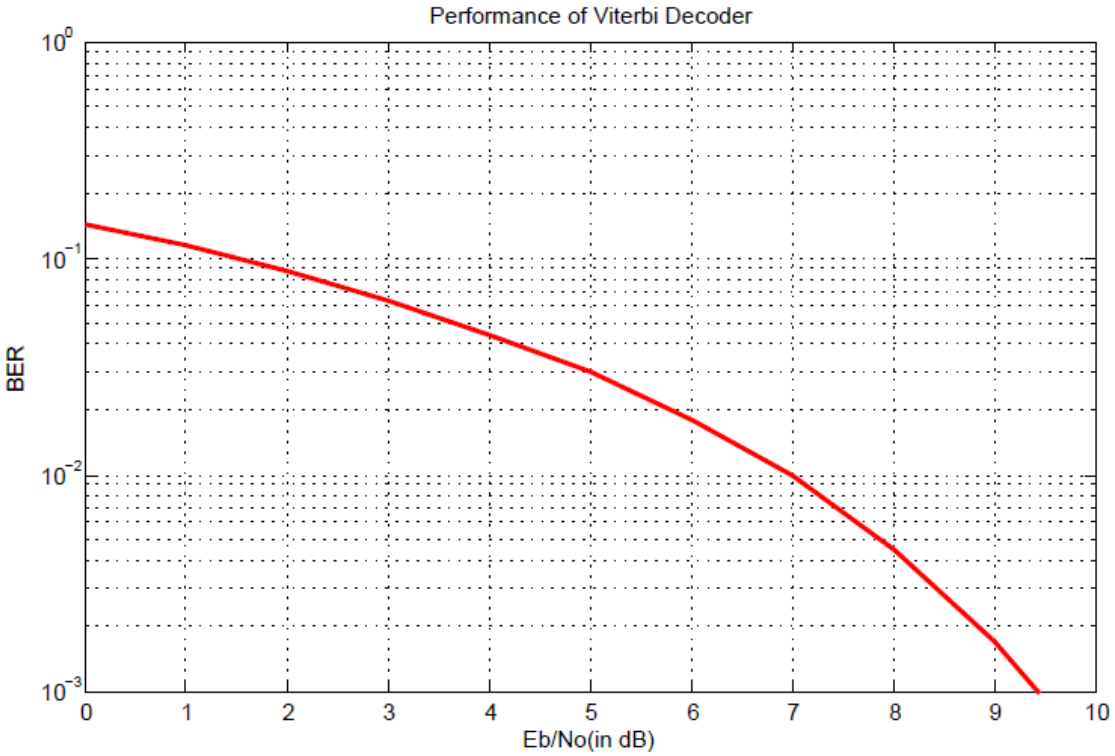

Fig 2.13: Simulated BER v/s Eb/No plot of Viterbi Decoder

Chapter-3

THE MAP ALGORITHM

3.1 MAP or BCJR decoder

In 1974 Bahl, Cocke, Jelinek and Raviv [2] published the decoding algorithm based on a posteriori probabilities later on known as the BCJR, Maximum a Posteriori (MAP) algorithm. The procedure was first applied to block and convolutional codes but, as it is more complex than the Viterbi algorithm and also didn't provide any advantage, for about 20 years it was not used in practical implementations. But this algorithm uses soft input and provides soft-output decisions which make it suitable for its use in the iterative decoding of turbo codes. Berrou, Glavieux and Thithimajshima [1], [8] who introduced turbo codes in 1993, used a modified version of the BCJR algorithm.

The MAP algorithm[4] computes the soft estimate LLR of a given bit based on the reception of sampled values of a given received sequence. The MAP algorithm[2],[3] is based on taking a decision based on a posteriori probability of each data bit. Given the noisy received sequence, a posteriori probabilities of each data bit is calculated and the data bit value having the maximum a posteriori values is chosen. In viterbi there is no concept of a posteriori probability. Instead it involves finding the most likely sequence that may have been transmitted. In the contrary in MAP we find the most likely value of the data bit at each bit time.

Consider a ½ rate zero terminated convolutional encoder. Let the input be $\underline{u}$ which can be represented as

Input $\underline{u} = \left[u_0 u_1 .... u_{k-1} \right]$

This input is then passed through a rate ½ zero terminated convolutional encoder,then we get output as:

Codeword $\underline{v} = \left[v_0^{(1)} v_0^{(2)} v_1^{(1)} v_1^{(2)} .....v_{k-1}^{(1)} v_{k-1}^{(2)} v_k^{(1)} v_k^{(2)} v_{k+1}^{(1)} v_{k+1}^{(2)} \right]$

This codeword $\underline{v}$ is then BPSK modulated which converts 0's to -1 and 1's to +1. The BPSK modulated version is represented as

$$\underline{y} = \left[y_0^{(1)} y_0^{(2)}\ y_1^{(1)} y_1^{(2)} \ldots\ldots\ldots y_{k-1}^{(1)} y_{k-1}^{(2)} y_k^{(1)} y_k^{(2)}\ y_{k+1}^{(1)} y_{k+1}^{(2)} \right]$$

After modulation the signal is finally transmitted through the channel. As the signal passes through the channel it suffers attenuation and fading. As a result many errors creep in. The most common form of impairment that can be considered is due to Additive white Gaussian noise (AWGN). This has an additive effect and thus adds up to the signal $\underline{y}$. So the noisy version of $\underline{y}$ can be represented as

$$\underline{r} = \left[r_0^{(1)} r_0^{(2)}\ r_1^{(1)} r_1^{(2)} \ldots\ldots\ldots r_{k-1}^{(1)} r_{k-1}^{(2)} r_k^{(1)} r_k^{(2)}\ r_{k+1}^{(1)} r_{k+1}^{(2)} \right]$$

This $\underline{r}$ is the signal which is finally received at the receiver side and which has to be decoded to recover the original input signal.

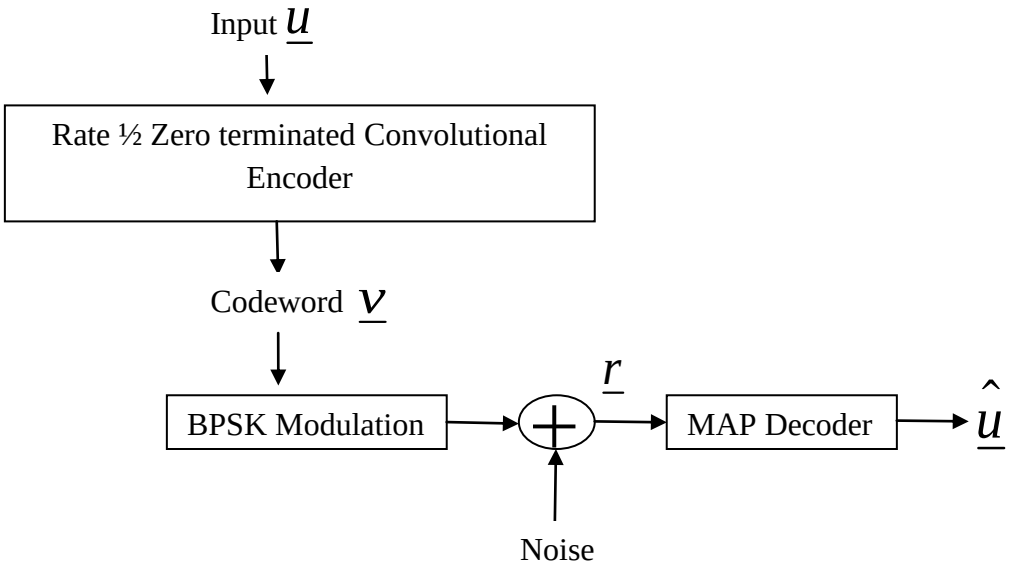

Fig 3.1: MAP decoder as Channel decoder

So at any l^{th} bit time or stage we have the received or output values as $r_l = \left[r_l^{(1)}\ r_l^{(2)} \right]$

Knowing this received value at each l^{th} stage i.e r_l the aim is to find the corresponding u_l i.e the input value. So decoder only knows r_l . u_l is a random variable at the input of the receiver. So goal in bitwise MAP decoder is to compute

$p(u_l \mid r_l)$ **:goal of bit wise MAP**

We don't compute this probability directly. We rather compute log likelihood ratios or LLR's Log Likelihood ratio is defined as

$$\text{LLR}(u_l) = L(u_l = 1 | \underline{r}) = \ln \frac{p(u_l = 1 | \underline{r})}{p(u_l = 0 | \underline{r})} \quad \ldots\ldots\ldots(3.1)$$

Here the numerator corresponds to the following probability

$$p(u_l = 1 | \underline{r}) = \sum_{s' \xrightarrow[u_l=1]{} s} p(s_l = s', s_{l+1} = s, \underline{r})$$

Here where the transition to next state occurs with input bit value equal to 1 and so has been labelled as $s' \xrightarrow[u_l=1]{} s$

Similarly the denominator corresponds to the following probability

$$p(u_l = 1 | \underline{r}) = \sum_{s' \xrightarrow[u_l=0]{} s} p(s_l = s', s_{l+1} = s, \underline{r})$$

For a four state trellis,there will be in total 8 branches. Out of these 4 will correspond to ul=0 in trellis. The other four correspond to $u_l = 1$. So in the computation of the denominator of the LLR we will only consider those branches of the trellis where $u_l = 0$ happen and then look at their starting and ending states and then let s' and s vary over those starting states and ending states.

Now doing step by step analysis:

$p(s_l = s', s_{l+1} = s, \underline{r})$ is basically a probability that in the lth stage, the previous state was s', the next state was s and the vector r was received.In MAP algorithm this probability is decomposed into the following form

$$p(s_l = s', s_{l+1} = s, \underline{r}) = \alpha_{l-1}(s')\gamma_l(s', s)\beta_l(s) \quad \ldots(3.2)$$

Where

$$\alpha_{l-1}(s') = p(s_l = s', \underline{r}_0^{l-1})$$
$$\gamma_l(s', s) = p(s_{l+1} = s, r_l^{(1)}, r_l^{(2)} | s_l = s')$$
$$\beta_l(s) = p(\underline{r}_{l+1}^{k+1} | s_{l+1} = s)$$
$$p(s_l = s', s_{l+1} = s, \underline{r}) = p(s', s, \underline{r})$$

So the LLR's calculation can then be written as

$$LLR(u_l) = \log \frac{\sum_{(s',s):input=1} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}{\sum_{(s',s):input=0} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)} \quad \text{..... (3.3)}$$

3.2 Proof of MAP algorithm

The received vector is $\underline{r}$ which has both the systematic bits and the parity bits. So the received vector at any stage l i.e. $\underline{r}_l$ can be written as $\underline{r}_l = \left[r_l^{(1)} r_l^{(2)} \right]$

Then $p(s_l = s', s_{l+1} = s, \underline{r})$ can be written in the simplified form as $p(s', s, \underline{r})$

So $p(s_l = s', s_{l+1} = s, \underline{r}) = p(s', s, \underline{r})$ …….. (3.4)

The sequence $\underline{r}$ above represents the corrupted bit sequence after it has travelled through the decoder and presented to the decoder Let the received sequence $\underline{r}$ be written in the form of sequence representing the sequence in the past, at the present and in the future i.e. $\underline{r} = [\underline{r}_0^{l-1}, \underline{r}_l, \underline{r}_{l+1}^{k+1}]$

Then $p(s', s, \underline{r}) = p(s', s, \underline{r}_0^{l-1}, \underline{r}_l, \underline{r}_{l+1}^{k+1})$ … (3.5)

Now from the Bayes rule $p(A, B, C, D, E) = p(E \mid A, B, C, D)\, p(A, B, C, D)$

Using this $p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid s', s, \underline{r}_0^{l-1}, \underline{r}_l)\, p(s', s, \underline{r}_0^{l-1}, \underline{r}_l)$

Now the term $p(\underline{r}_{l+1}^{k+1} \mid s', s, \underline{r}_0^{l-1}, \underline{r}_l)$ means the probability of received sequence at $(l+1)$ th stage to be $\underline{r}_{l+1}^{k+1}$ when the state at l th stage be s' ; the state at $(l+1)$ th stage to be s and the past received sequences being $\underline{r}_0^{l-1}$ and $\underline{r}_l$. Now the terms s', $\underline{r}_l$ and $\underline{r}_0^{l-1}$ represent past information and are irrelevant. These terms do not matter and don't exert any influence. Once we have state at $(l+1)$ th stage to be s the states and received sequences of past irrelevant. Hence the above equation becomes and can be stroke out as shown below

$$p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid \cancel{s'}, s, \cancel{\underline{r}_0^{l-1}}, \cancel{\underline{r}_l})\, p(s', s, \underline{r}_0^{l-1}, \underline{r}_l)$$

$$= p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s', s, \underline{r}_0^{l-1}, \underline{r}_l)$$

Again using the Bayes rule the above equation is further decomposed as

$$= p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s', \underline{r}_0^{l-1})\, p(s', \underline{r}_0^{l-1})$$

Again neglecting the past information the above equation reduces to

$$= p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s', \cancel{\underline{r}_0^{l-1}})\, p(s', \underline{r}_0^{l-1})$$
$$= p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s')\, p(s', \underline{r}_0^{l-1})$$

So we get

$$p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s')\, p(s', \underline{r}_0^{l-1})$$
$$p(s', s, \underline{r}) = p(s_l = s', \underline{r}_0^{l-1})\, p(s_{l+1} = s, \underline{r}_l \mid s_l = s')\, p(\underline{r}_{l+1}^{k+1} \mid s_{l+1} = s)$$
$$p(s', s, \underline{r}) = \alpha_{l-1}(s')\gamma_l(s', s)\beta_l(s)$$
$$\alpha_l(s) = p(s_{l+1} = s, \underline{r}_0^l)$$
$$\alpha_{l-1}(s') = p(s_l = s', \underline{r}_0^{l-1})$$
$$\gamma_l(s', s) = p(s_{l+1} = s, \underline{r}_l \mid s_l = s')$$
$$\beta_l(s) = p(\underline{r}_{l+1}^{k+1} \mid s_{l+1} = s)$$

The equation $p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s', \underline{r}_0^{l-1})\, p(s', \underline{r}_0^{l-1})$ using the Bayes rule the above equation is further decomposed as follows

$$p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s', \underline{r}_0^{l-1})\, p(s', \underline{r}_0^{l-1})$$

Again the term $\underline{r}_0^{l-1}$ in the second factor is past information which doesn't exert influence and hence can be ignored and hence is stroke out as shown below

$$= p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s', \cancel{\underline{r}_0^{l-1}})\, p(s', \underline{r}_0^{l-1})$$

$$p(s', s, \underline{r}) = p(\underline{r}_{l+1}^{k+1} \mid s)\, p(s, \underline{r}_l \mid s')\, p(s', \underline{r}_0^{l-1})$$

Or $p(s', s, \underline{r}) = p(s_l = s', \underline{r}_0^{l-1})\, p(s_{l+1} = s, \underline{r}_l \mid s_l = s')\, p(\underline{r}_{l+1}^{k+1} \mid s_{l+1} = s)$ ….(3.6)

In MAP algorithm [3], [4] the above expression is decomposed into the following form

$p(s', s, \underline{r}) = \alpha_{l-1}(s')\gamma_l(s', s)\beta_l(s)$

Where

$\alpha_l(s)$ is the forward state metric which is defined as

$\alpha_l(s) = p(s_{l+1} = s, \underline{r}_0^l)$

So clearly $\alpha_{l-1}(s') = p(s_l = s', \underline{r}_0^{l-1})$

$\gamma_l(s',s)$ is the branch metric defined as

$\gamma_l(s',s) = p(s_{l+1} = s, \underline{r}_l \mid s_l = s')$

$\beta_l(s)$ is the backward state metric which is defined as $\beta_l(s) = p(\underline{r}_{l+1}^{k+1} \mid s_{l+1} = s)$

3.3 Computation of Metrics

3.3.1 Calculation of branch metrics

$\gamma_l(s',s) = p(s_{l+1} = s, r_l \mid s_l = s')$

$$= p(s_{l+1} = s \mid s_l = s')\, p(r_l \mid s_l = s', s_{l+1} = s) \quad \ldots. (3.7)$$

Now the probability term $p(s_{l+1} = s \mid s_l = s')$ is the probability of the input which causes a transition from state s' to s. Thus it is basically the probability of input. In the trellis diagram we have 8 branches for two states. Four branches correspond to transitions with 0 as input and the other four correspond to input 1 as input. Eventually this probability term can be regarded as the a priori probability. Since both 1's and 0's are equal likely this is equal to ½.

Thus $p(s_{l+1} = s \mid s_l = s') = p(u_l = u(s' \to s))$

$$= 1/2$$

Now $r_l = \left[r_l^{(1)}\, r_l^{(2)} \right]$

$p(r_l \mid s_l = s', s_{l+1} = s) = p(r_l^{(1)}, r_l^{(2)} \mid y^{(1)}(s' \to s), y^{(2)}(s' \to s))$

This $y^{(1)}(s' \to s)$ and $y^{(2)}(s' \to s)$ are BPSK modulated versions of the encoder outputs of $v^{(1)}(s' \to s)$ and $v^{(2)}(s' \to s)$ respectively i.e

$y^{(1)}(s' \to s) = 2v^{(1)}(s' \to s) - 1$

$y^{(2)}(s' \to s) = 2v^{(2)}(s' \to s) - 1$

Now $r_l^{(1)}$ is Gaussian distributed with mean $y^{(1)}(s' \to s)$ and $r_l^{(2)}$ is Gaussian distributed with mean $y^{(2)}(s' \to s)$ therefore

Equation $p(r_l \mid s_l = s', s_{l+1} = s) = p(r_l^{(1)}, r_l^{(2)} \mid y^{(1)}(s' \to s), y^{(2)}(s' \to s))$ becomes

$$= \frac{1}{2\pi\sigma^2} e^{\frac{-\left[\left(r_l^{(1)} - y^{(1)}\right)^2 + \left(r_l^{(2)} - y^{(2)}\right)^2\right]}{2\sigma^2}} \quad \text{….. (3.8)}$$

Where σ^2 is the variance of the additive white Gaussian noise.

3.3.2 Calculation of Forward Metrics

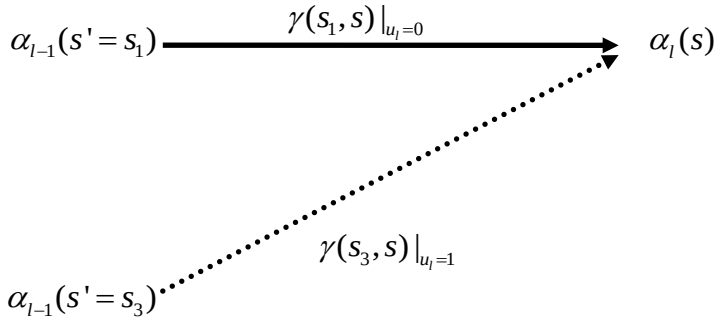

Fig 3.2: Forward Metric computation

$$\alpha_l(s) = p(s_{l+1} = s, \underline{r}_0^l)$$

$$= \sum_{s'} p(s_{l+1} = s, s_l = s', \underline{r}_0^{l-1}, r_l)$$

Using the Bayes rule this can be written as

$$= \sum_{s'} p(s_{l+1} = s, r_l \mid s_l = s', \underline{r}_0^{l-1}) p(s_l = s', \underline{r}_0^{l-1})$$

$$= \sum_{s'} p(s_{l+1} = s, r_l \mid s_l = s', \underline{r}_0^{l-1}) \alpha_{l-1}(s')$$

$$= \sum_{s'} \gamma_l(s', s) \alpha_{l-1}(s') \quad \text{…(3.9)}$$

Here the summations over s' observing the trellis there are eight such branches, four of them with input zero and the other four with input equal to 1.

3.3.3 Calculation of Backward Metrics

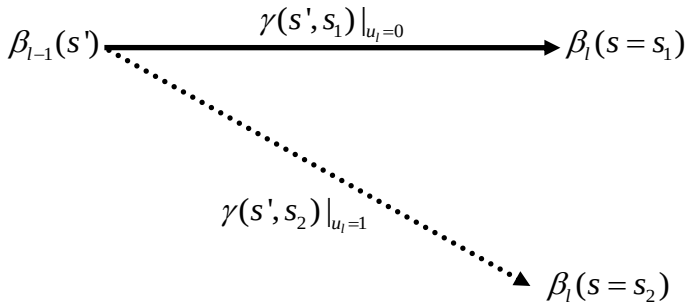

Fig 3.3: Backward Metric computation

$$\beta_l(s) = p(r_{l+1}^{k+1} \mid s_{l+1} = s)$$

Then $\beta_{l-1}(s') = p(r_l^{k+1} \mid s_l = s')$

$$= \sum_s p(r_l, r_{l+1}^{k+1}, s_{l+1} = s \mid s_l = s')$$

Using Bayes Rule

$$= \sum_s p(s_{l+1} = s, r_l \mid s_l = s')\, p(r_{l+1}^{k+1} \mid s_{l+1}, r_l, s_l)$$

Since past information can be neglected the above expression reduces to

$$= \sum_s p(s_{l+1} = s, r_l \mid s_l = s')\, p(r_{l+1}^{k+1} \mid s_{l+1})$$

$$= \sum_s \gamma(s', s)\beta_l(s) \qquad \ldots. (3.10)$$

While computing these forward branch metrics and backward branch metrics recursively their values may increase to a very large numbers for large number of bits. This may cause problems of overflow and increases complexity. To solve this, in every iteration the forward and backward branch metrics are normalized to their average of that iteration.

The following figure shows BER error performance of MAP decoder in the presence of AWGN noise environment. The simulation was done for 6 lakhs bits in MATLAB. The frame size was taken to be 6000. As can be seen from the plot, the error reduces with increasing energy per bit. Also the performance is found to better than that of the performance of the hard decision viterbi decoder simulated in fig 2.13.

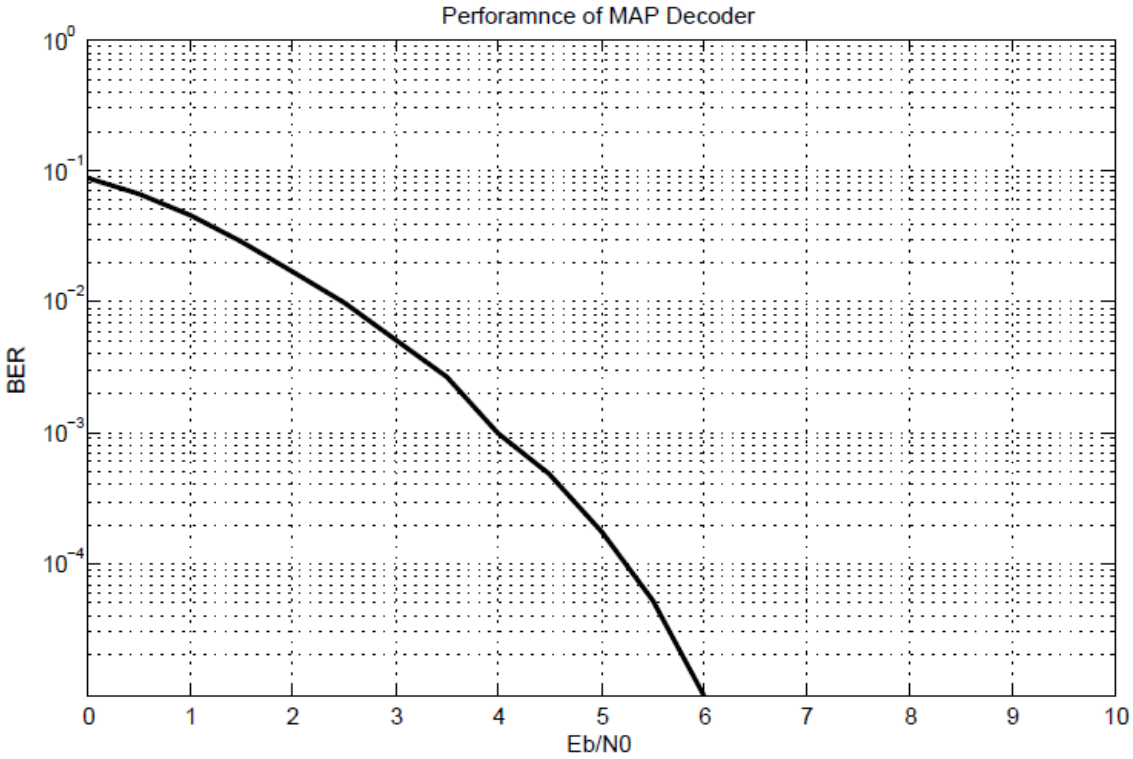

Fig 3.4: Simulated BER v/s Eb/No plot of MAP Decoder

Chapter-4

TURBO CODES AND ITERATIVE DECODING

4.1 Fundamental of Turbo Code

Shannon's random coding theoram states that codes achieving channel capacity must have random distribution. However the decoding complexity follows an exponential growth for large length codes generated randomly. Hence Maximum Likelihood decoding becomes impractical. Turbo codes is a method to combine two simple codes connected by a pseudorandom interleaver of large length to generate code close to random structure. They give better performance at lower SNR's. However because the resulting code is based on combining simple codes, its decoding is possible by an iterative scheme based on the decoding of constituent codes.

Turbo principle is actually a concept that can be used for anything other than this channel codes also. It can be applied whereever there is a concatenation of two trellises or two structured entities. Turbo decoder is basically a message type decoder where we can go back and forth between the two decoders and that is the turbo principle idea. The name Turbo comes from the reason of its iterative cyclic feedback mechanism employed for decoding similar to the turbo machines. These individual cycles which produce a small change add up to a big improvement in the turbo principle idea.

With the introduction of turbo codes[10],it has been adopted in a variety of communication systems. The turbo decoding algorithm [3] has shown remarkable receiver performance improvements over hard decision decoding or separate equalization and decoding, by exchanging soft information between two SISO decoding blocks. This performance enhancement has led turbo codes to be accepted as the coding technique in several standards such as Wideband CDMA (WCDMA), the 3rd Generation Partnership Project (3GPP) for IMT- 2000, Consultative Committee for Space Applications (CCSDS) telemetry channel coding, UMTS, DVB-RCS, and IEEE 802.16ab.

4.2 Turbo Encoder

A basic turbo encoder is an encoder employs two recursive systematic convolutional encoders in parallel [1], [3], [4]. The input is passed through an interleaver which scrambles the input message and this interleaved message is fed to the second encoder. On the other hand original non interleaved version of the input message is simultaneously fed to the first encoder. The recursive convolutional encoders employed may be either identical or different.

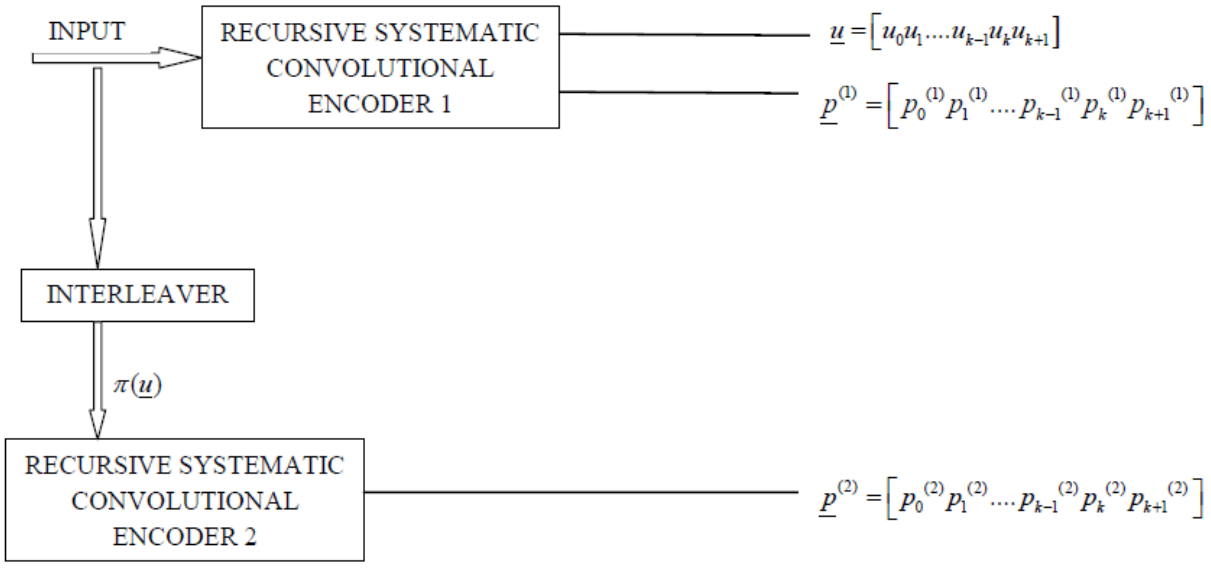

Fig 4.1: Block Diagram of Turbo Encoder

4.2.1 Turbo Encoder using Recursive Systematic Encoders

The weight distribution of the code words produced by the turbo encoder above depends on the method of combining the code words from the two encoders. Intuitively low weight code words from one encoder should not be paired with low weight code words from the other encoder. This can be achieved by the proper design of the interleaver [3], [11].

The use of two recursive convolutional encoders in conjunction with the interleaver produces a code that contains very few code words of low weight having relatively few nearest neighbors. That is the code words are relatively sparse. Hence the coding gain achieved by a turbo code is due in part to this feature. A turbo encoder making use of Recursive Convolutional encoders as its constituent encoders is shown in the following figure 4.2.

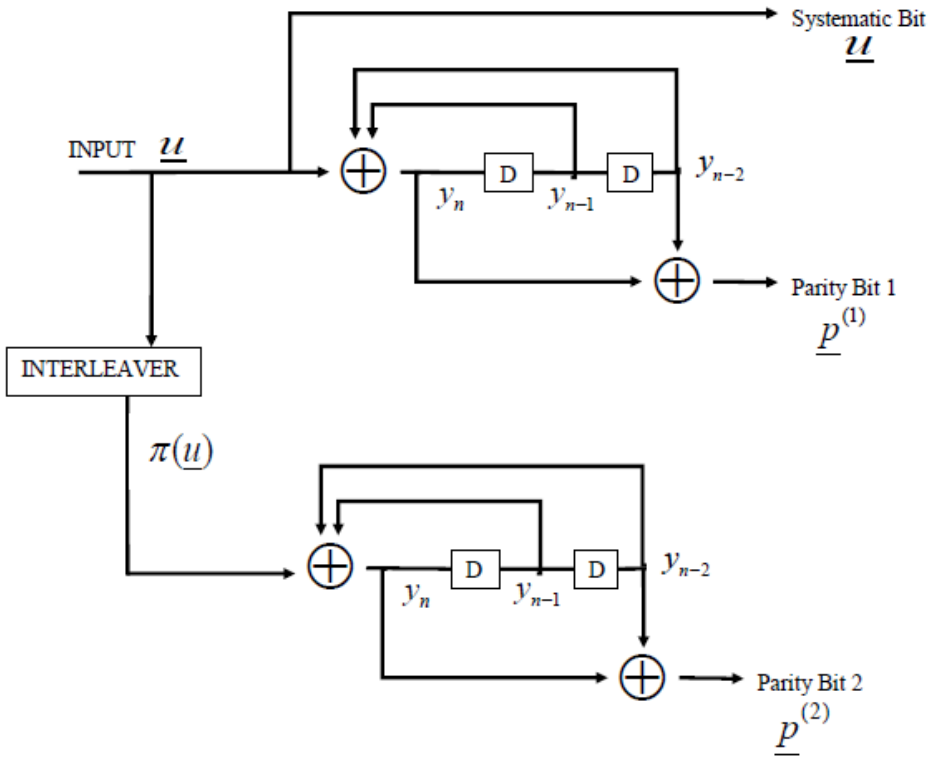

Fig 4.2: Recursive Systematic Convolutional Encoders as building blocks of Turbo Encoder

The recursive systematic convolutional encoder is like an IIR filter. Thus it is possible that an input sequence may not yield a minimum weight codeword out of the encoder. The encoder output weight is thus kept finite by trellis termination. Through this process the encoder returns to the all zero state. This is done by tail biting in which the input is padded with few bits so that the encoder may return to the all zero state. The usual practice is to zero terminate only the first encoder and not to zero terminate the second encoder. For an input $\underline{u}$ having k bits i.e. $\underline{u} = [u_0\, u_1 \ldots\ldots\ldots u_{k-1}]$ with tail biting two bits are appended to the input to terminate the first encoder with all zero state. So effectively the input is $\underline{u} = [u_0\, u_1 \ldots\ldots\ldots u_{k-1} u_k u_{k+1}]$ and its length will be $k+2$. The interleaver thus employed is of length $k+2$.

4.3 Turbo Decoder

Optimal decoding of turbo codes is impossible due to the large number of states in the code trellis. A suboptimal iterative decoding algorithm [2],[3] known as the turbo decoding algorithm achieves excellent performance close very close to the theoretical bound predicted

by Shannon. The iterative turbo decoding consists of two component decoders serially concatenated via an interleaver identical to the one used in the encoder.

The first MAP decoder takes as input the received systematic information sequence $r^{(s)}$ and the received parity sequence $r^{(1)}$ generated by the first recursive systematic encoder of turbo encoder. The decoder then produces a soft output (called extrinsic information) which is interleaved (in the same fashion as the interleaver of turbo encoder) and is fed to the second decoder. This soft output act as a priori probabilities of the information sequence fed to the second decoder after interleaving (in the same fashion as the interleaver of turbo encoder). The second decoder is fed with interleaved version of received systematic information $\pi(r^{(s)})$ and the received parity sequence generated by the second recursive systematic encoder of the turbo encoder $r^{(2)}$ .The second MAP decoder also produces soft output (extrinsic information) which is deinterleaved and then fed to the first decoder to improve the estimate of a priori probabilities for the information sequence at the input of the first MAP decoder. This completes one iteration. After a defined number of iterations hard decision is done on the log likelihood ratio's values (i.e. LLR's) for the estimate of the input message. The operation is carried out for multiple iterations and with each further iteration the error reduces and the estimation of the input message sequence becomes better.

4.4 Iterative MAP decoding Method of Turbo Decoding

Let us say we are using a rate 1/3 turbo code. For an input $\underline{u} = \left[u_0\, u_1 \ldots\ldots\ldots u_{k-1} u_k u_{k+1}\right]$ codeword output of the turbo encoder is $\underline{y} = \left[u_0 p_0^{(1)} p_0^{(2)}, u_1 p_1^{(1)} p_1^{(2)} \ldots\ldots u_{k-1} p_{k-1}^{(1)} p_{k-1}^{(2)} \ldots.\right]$. After BPSK modulation this becomes $\underline{v} = \left[y_0 y_0^{(1)} y_0^{(2)}, y_1 y_1^{(1)} y_1^{(2)} \ldots\ldots y_{k-1} y_{k-1}^{(1)} y_{k-1}^{(2)} \ldots.\right]$. After passing it through Gaussian channel the output sequence is $\underline{r} = \left[r_0 r_0^{(1)} r_0^{(2)}, r_1 r_1^{(1)} r_1^{(2)} \ldots\ldots r_{k-1} r_{k-1}^{(1)} r_{k-1}^{(2)} \ldots.\right]$. This can be written after rearranging as $\underline{r} = \left[r_0 r_1 r_{k-1} \ldots\ldots., r_0^{(1)} r_1^{(1)} r_{k-1}^{(1)} \ldots\ldots., r_0^{(2)} r_1^{(2)} r_{k-1}^{(2)} \ldots.\right]$ or

$$\left[\underbrace{r_0 r_1 \ldots r_{k-1} \ldots\ldots.}_{r^{(s)}}, \underbrace{r_0^{(1)} r_1^{(1)} \ldots r_{k-1}^{(1)} \ldots\ldots.}_{r^{(1)}}, \underbrace{r_0^{(2)} r_1^{(2)} \ldots r_{k-1}^{(2)} \ldots..}_{r^{(2)}}\right].$$ Now this $r(s)$ and $r^{(1)}$ are fed to the

decoder 1 while an interleaved version of $r(s)$ i.e $\pi(r^{(s)})$ and $r^{(2)}$ are fed to the decoder 2. This is illustrated with the turbo decoder [3] block diagram of fig 4.3

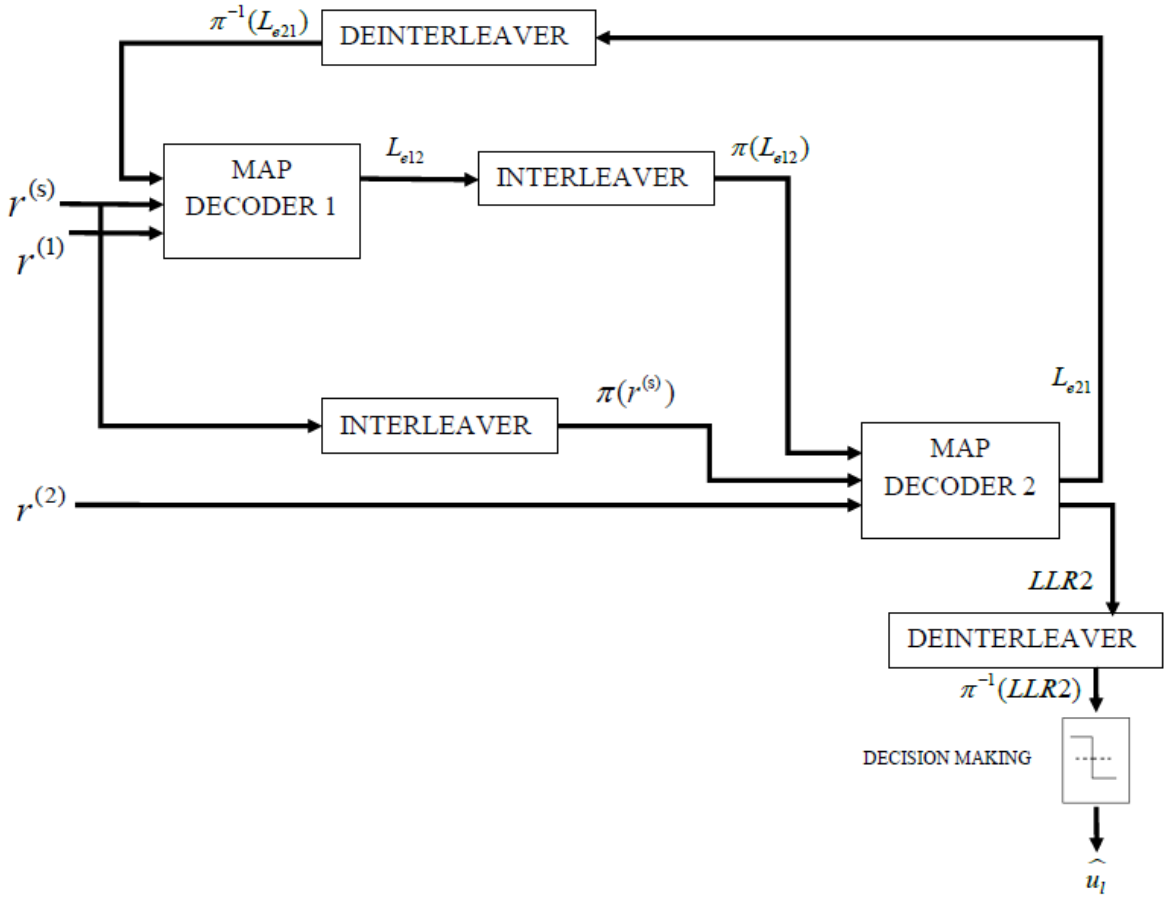

Fig 4.3: Turbo Decoder

Now the method of decoding is as follows

- The extrinsic information is basically the logarithmic ratio of probability of getting 1 with the probability of getting 0. i.e $L_e^{\,t}(u_l) = [\delta_0\, \delta_1 ..... \delta_l ... \delta_{k-1} ....]$ where t denotes the iteration number and $\delta_l = \log \dfrac{p(u_l = 1)}{p(u_l = 0)}$ .Then the individual probabilities of 1's or 0's

can be calculated from the given extrinsic information as $p(u_l = 1) = \frac{e^{\delta_l}}{1+e^{\delta_l}}$ and $p(u_l = 0) = \frac{1}{1+e^{\delta_l}}$

- In the first iteration the decoder assumes all the bits are equiprobable i.e. the probability of getting either 1 or 0 is ½. This implies $\delta_l = 0$ for each bit. Thus for the first iteration for the first decoder initialize $L_{e21}{}^{(0)}(u_l) = 0$
- For iterations t=1,2…….,I where I is the total number of iterations

  For the **First MAP decoder**

  Compute Output Likelihood ratios (LLR) of decoder 1

$$LLR1^{(t)}(u_l) = \log \frac{\sum_{s',s:u(s'\to s)=1} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}{\sum_{s',s:u(s'\to s)=0} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)} \quad \text{….. (4.1)}$$

  Where $\gamma_l(s',s) = p(u_l \mid (s' \to \mathrm{s}))\, p(r_l^{(\mathrm{s})}, r_l^{(1)} \mid y_l^{(\mathrm{s})}, y_l^{(1)})$

  Also $p(u_l \mid (s' \to \mathrm{s}))$ is the a priori probability which is updated with deinterleaved version of the extrinsic information i.e. $\pi^{-1}(L_{e21}{}^{(t-1)})$ or $\pi^{-1}(\delta_l)$ from the other decoder with each iteration as $p(u_l = 1) = \frac{e^{\pi^{-1}(\delta_l)}}{1+e^{\pi^{-1}(\delta_l)}}$ and $p(u_l = 0) = \frac{1}{1+e^{\pi^{-1}(\delta_l)}}$

  Since white noise has been added it is Gaussian distributed

  Therefore $LLR1^{(t)}(u_l) = \log \frac{\sum_{s',s:u(s'\to s)=1} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}{\sum_{s',s:u(s'\to s)=0} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}$ ….(4.2)

  Now

$$LLR1^{(t)}(u_l) = \pi^{-1}(L_{e21}{}^{(t-1)}) + \frac{2}{\sigma^2} r_l^{(\mathrm{s})} + L_{e12}{}^{(t)}(u_l) \quad \text{…. (4.3)}$$

  The term $L_{e12}{}^{(t)}$ corresponds to the extrinsic information output from decoder 1 for this iteration while the term $\frac{2}{\sigma^2} r_l^{(\mathrm{s})}$ is the intrinsic component. As can be seen from the

above equation the extrinsic information can be calculated by subtracting other terms from the Log Likelihood ratio values (LLR1) i.e.

$$L_{e12}{}^{(t)}(u_l) = LLR1^{(t)}(u_l) - \pi^{-1}(L_{e21}{}^{(t-1)}) + \frac{2}{\sigma^2} r_l^{(s)} \quad \text{….. (4.4)}$$

- The extrinsic information [3] output from the decoder 1 i.e. $L_{e12}{}^{(t)}$ is passed through an interleaver and fed to decoder 2. So decoder 2 gets $\pi(L_{e12}{}^{(t)})$ along with the inputs $\pi(r^{(s)})$ and $r^{(2)}$

Now for **MAP Decoder 2**

Compute Output Likelihood ratios(LLR) of decoder 2

Therefore $LLR2^{(t)}(u_l) = \log \dfrac{\sum_{s',s:u(s'\to s)=1} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}{\sum_{s',s:u(s'\to s)=0} \alpha_{l-1}(s')\gamma_l(s',s)\beta_l(s)}$ ….(4.5)

Where $\gamma_l(s',s) = p(u_l \mid (s' \to s))\, p(\pi(r_l^{(s)}), r_l^{(2)} \mid \pi(y_l^{(s)}), y_l^{(2)})$

Also $p(u_l \mid (s' \to s))$ is the a priori probability which is updated with interleaved version of the extrinsic information i.e. $\pi(L_{e12}{}^{(t)})$ or $\pi(\delta_l)$ from the other decoder with each iteration as $p(u_l = 1) = \dfrac{e^{\pi(\delta_l)}}{1+e^{\pi(\delta_l)}}$ and $p(u_l = 0) = \dfrac{1}{1+e^{\pi(\delta_l)}}$

Since white noise has been added it is Gaussian distributed

Therefore $p(\pi(r_l^{(s)}), r_l^{(2)} \mid \pi(y_l^{(s)}), y_l^{(2)}) = \dfrac{1}{2\pi\sigma^2} e^{\frac{-[(\pi(r_l^{(s)}) - \pi(y_l^{(s)}))^2 + (r_l^{(2)} - y_l^{(2)})^2]}{2\sigma^2}}$ ….(4.6)

Now

$$LLR2^{(t)}(u_l) = \pi(L_{e12}{}^{(t)}) + \frac{2}{\sigma^2}\pi(r_l^{(s)}) + L_{e21}{}^{(t)}(u_l)$$

$$L_{e21}{}^{(t)}(u_l) = LLR2^{(t)}(u_l) - \pi(L_{e12}{}^{(t)}) + \frac{2}{\sigma^2}\pi(r_l^{(s)}) \quad \text{…. (4.7)}$$

- The term $L_{e21}{}^{(t)}$ corresponds to the extrinsic information output from decoder 2 for this iteration while the term $\dfrac{2}{\sigma^2}\pi(r_l^{(s)})$ is the intrinsic component. As can be seen from the equation (4.7) the extrinsic information can be calculated by subtracting other terms from the Log Likelihood ratio values (LLR2) i.e.

$$L_{e21}{}^{(t)}(u_l) = LLR2^{(t)}(u_l) - \pi(L_{e12}{}^{(t)}) + \frac{2}{\sigma^2}\pi(r_l^{(s)})$$

- After I number of iterations make a hard decision on the deinterleaved values of the log likelihood ratios of the second decoder to estimate the input $\hat{u}$

  i.e $\widehat{u_l} = 1$ for $\pi^{-1}(LLR2(u_l)) > 0$

  and $\widehat{u_l} = 0$ for $\pi^{-1}(LLR2(u_l)) < 0$

The following figure shows BER error performance of Turbo Decoder in the presence of AWGN noise environment. The turbo decoder here comprises of parallel concatenated MAP decoders. A pseudo random interleaver has been used here. The simulation was done for 6 lakhs bits in MATLAB. The frame size was taken to be 6000. The result given in Fig 4.4 has been generated using Simulink model while that in Fig 4.5 has been generated in MATLAB.

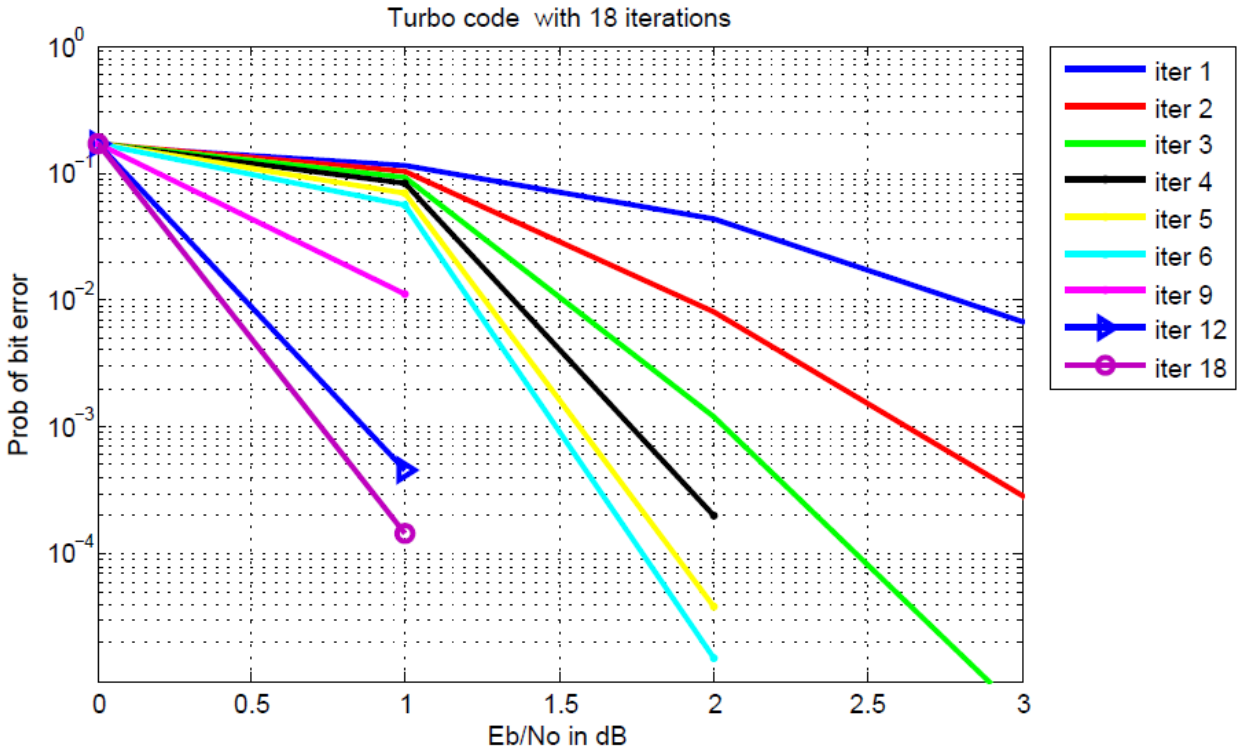

Fig 4.4: BER v/s Eb/No plot of Turbo Decoder generated using Simulink

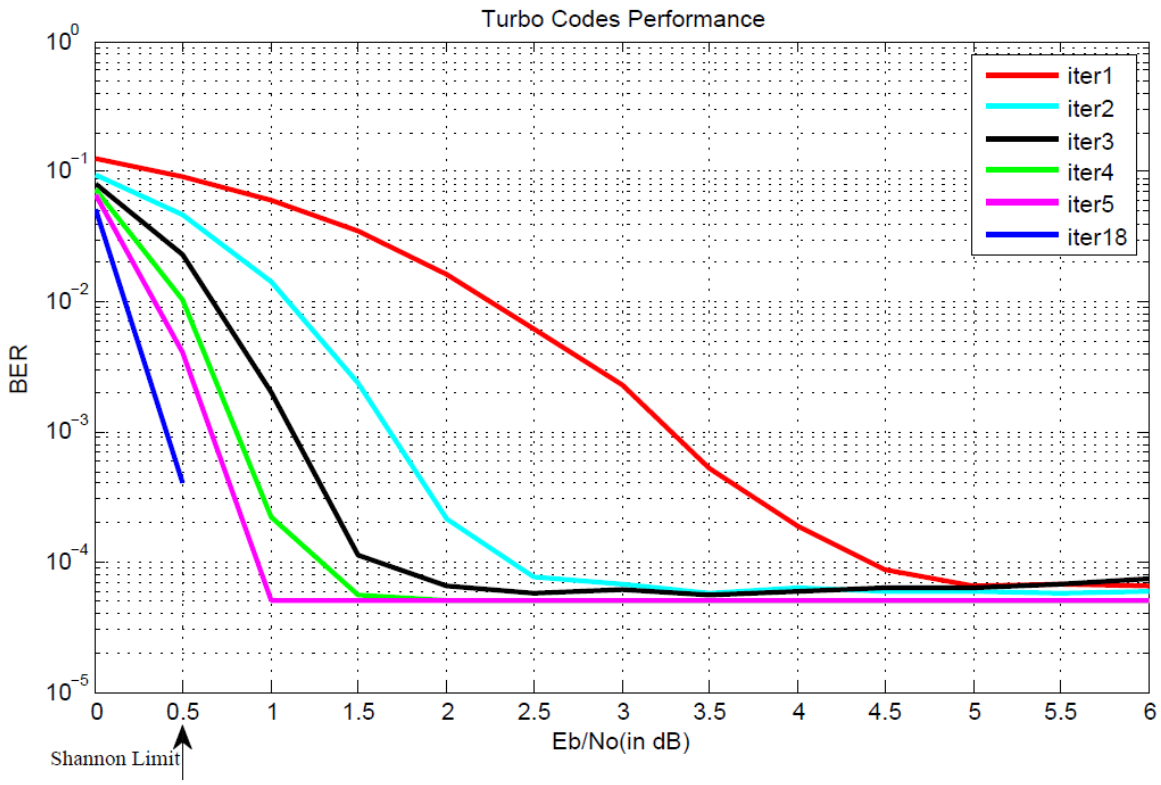

Fig 4.5: Simulated BER v/s Eb/No plot of Turbo Decoder in MATLAB

As can be seen from the plots of both the figures above the error reduces with each iteration.So performance improves with each iteration. Also performance is quite good even in noisy SNR's. A BER of the order of 1E-4 has been achieved which is sufficient in wireless applications. Also as can be seen from the plot, as the iterations continue to increase, the performance ultimately approaches the shannon limit.

Chapter 5

WIRELESS FADING

5.1 Fading

In wireless communications, fading is deviation of the attenuation affecting a signal over certain propagation media. The fading may vary with time, geographical position or radio frequency, and is often modeled as a random process. A fading channel is a communication channel comprising fading. In wireless systems, fading may either be due to multipath propagation, referred to as multipath induced fading, or due to shadowing from obstacles affecting the wave propagation, sometimes referred to as shadow fading.

5.2 Types of Fading

5.2.1 Gaussian Noise

The PDF of the Gaussian distribution (also called as Normal Distribution) denoted by $N(\mu,\sigma^2)$ is completely characterized by its mean μ and variance σ^2.

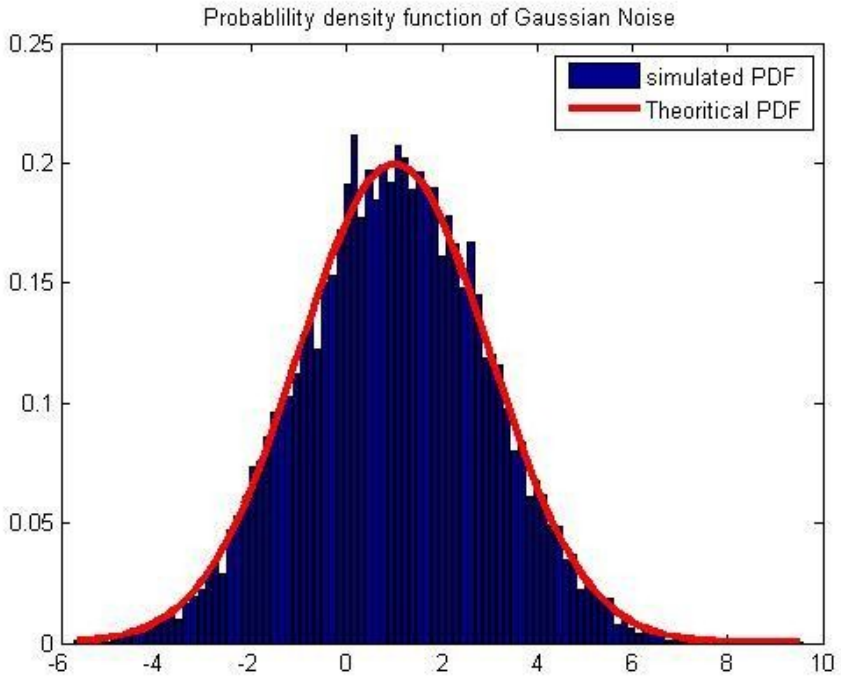

Fig 5.1: Simulated Gaussian distribution plot

The pdf of Gaussian noise is given as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{[x-\mu]^2}{2\sigma^2} \right)$$

Fig 5.1 shows a plot of simulation of Gaussian probability density function in MATLAB. The theoretical plot of pdf has been compared with the probability density plot of explicitly generated Gaussian distributed Random Variables.

The Gaussian distribution is used most widely in modeling the communication channel. For example, all channels are assumed to be Additive White Gaussian Noise channel. The reason behind it is that Gaussian noise gives the smallest channel capacity with fixed noise power. This means that it results in the worst channel impairment. So the coding designs done under this most adverse environment will give superior and satisfactory performance in real environments.

5.2.2 Rayleigh Fading

In communication systems the signal amplitude values of a randomly received signal usually can be modeled as Rayleigh distribution. In multipath propagation scenarios like urban environments, the transmitted signal arrives at the receiver following different paths and through different mechanisms of Refraction, Scattering and reflection. So the resultant signal at the receiver can be modeled as a random variable since it is a summation of different multipath components characterized by time varying attenuations, delays and phase shifts. When there are a large number of paths, the central limit theorem can be applied to model the time-variant impulse response of the channel as a complex-valued Gaussian random process. When the impulse response is modeled as a zero-mean complex-valued Gaussian process, the channel is said to be a Rayleigh fading channel. Here the Rayleigh Fading model is assumed to have only two multipath components X(t) and Y(t). Rayleigh Fading can be obtained from zero-mean complex Gaussian processes (X(t) and Y(t)). Simply adding the two Gaussian Random variables and taking the square root (envelope) gives a single tap Rayleigh distributed process. The phase of such random variable follows uniform distribution.

Consider two Gaussian random variables with zero mean and same variance $X \sim N(0,\sigma^2)$ and $Y \sim N(0,\sigma^2)$. Let's define a complex Gaussian random variable mimicking an IQ channel as $Z = X + jY$. Now, the envelope of the complex random variable is given by

$$R = \sqrt{X^2 + Y^2}$$

And the phase is given by

$$\varphi = \tan^{-1}\left(\frac{Y}{X}\right)$$

The envelope follows Rayleigh distribution and the phase will be uniformly distributed .The probability density function (Rayleigh distribution) of the above mentioned amplitude response is given by

$$f(r) = \frac{r}{\sigma^2}\exp(-r^2/2\sigma^2)$$

Rayleigh fading model [6] is generally used model to describe rich scattering environments. Fig below shows a plot of simulation of Rayleigh probability density function in MATLAB. The theoretical plot of pdf has been compared with the probability density plot of explicitly generated Rayleigh distributed Random Variables.

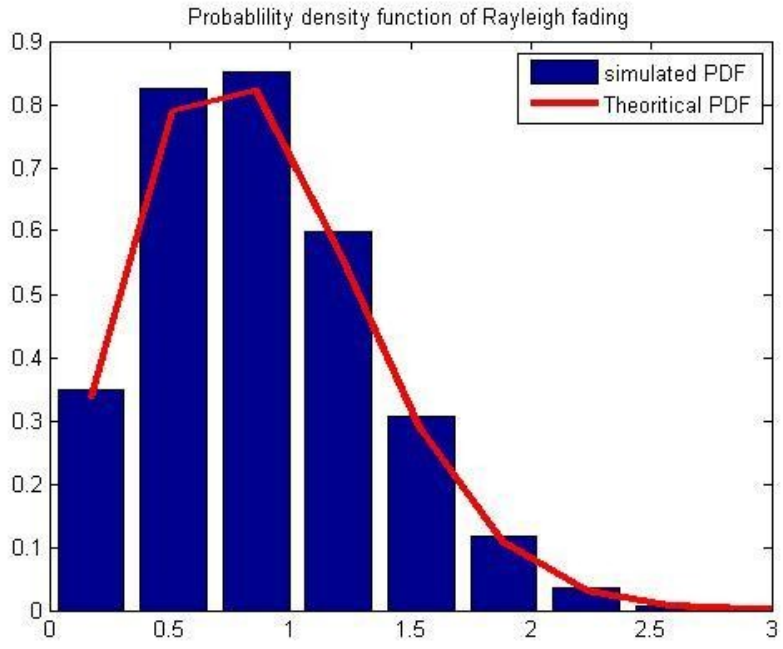

Fig 5.2: Simulated Rayleigh Distribution Plot

5.2.3 Rician fading

When strong LOS path exists, the received signal vector can be considered to be the sum of two vectors: a scattered Rayleigh vector with random amplitude and phase and a vector which is deterministic in amplitude and phase, representing the fixed path. In propagation scenarios when a LOS component is present between transmitter and receiver, signal arriving at receiver is expressed as sum of one dominant vector and large number of independently fading uncorrelated multipath components with amplitudes of the order of magnitudes and phases uniformly distributed in interval (0,2π). The received signal is characterized by Rice distribution and is given as follows:-

$$R_{Rice} = C + \sum_{i=1}^{n} a_i(t)\cos(w_c t + \theta_i(t))$$

Where C: magnitude of LOS signal between Transmitter and Receiver PDF of envelope of received signal is illustrated in following mathematical form:-

$$P_{rice}(r) = \frac{r}{\sigma^2}\exp[-\frac{(r^2+c^2)}{2\sigma^2}]I_0(\frac{rc}{\sigma^2})$$

Where I_0: is the zeroth modified Bessel function of the first kind and zero order

$c^2/2$: mean power of LOS components

For c=0, above equation reduces to Rayleigh distribution. Ratio of average specular power (direct path) to average fading power (multipath) over specular paths is known as Rician factor [7] and is expressed in dBs.

5.2.4 Log Normal Distribution

In addition to signal power loss due to hindrance of buildings, hills etc; vegetation and foliage is another important factor that cause scattering and absorption of radio waves by trees with irregular pattern of branches and leaves with different densities. As a result power of received signal varies about the mean power predicted by path loss. This type of variation in received signal power is called shadowing and is usually formulated as log normally distribution over the ensemble of typical locals. Shadowing creates holes in coverage areas and results in poor coverage. The pdf of received signals envelope affected by shadowing follows lognormal distribution

$$P_{\log normal}(r) = \frac{1}{\sqrt{2\pi}\sigma_r} \exp[-\frac{1}{2}(\frac{(\ln r - \mu)^2}{2\sigma^2})]$$

Where μ and σ are mean and standard deviation of shadowed component of received signal. With this distribution, log r has a Gaussian(normal) distribution. Due to multiple reflections in multipath, fading phenomenon can be characterized as multipative process. Multiplication of signal amplitude gives rise to lognormal distribution, in the same manner that can additive process results in normal distribution(with reference to Central limit theorem).

5.2.5 Nakagami Distributed fading

The random fluctuations in radio signal propagating through communication channels can be categorized into 2 types of fading: multipath fading and shadowing. The composite shadow fading (line of sight and multiplicative shadowing) can be modeled by lognormal distribution. The application of lognormal distribution to categorize shadowing effects results in complicated expressions for the first and second order statistics and also the performance evaluation of composite systems such as interference analysis and bit error rates become difficult. An alternative to lognormal is Nakagami which can produce simple statistical models with same performance. PDF of received signal envelope following Nakagami [9] is given by:-

$$P_r(r) = \frac{2}{\Gamma(m)}(\frac{m}{2\sigma^2})^m r^{2m-1} \exp(-\frac{mr^2}{2\sigma^2}) \ , \ r \geq 0$$

Where $\Gamma(.)$: Gamma function
$2\sigma^2 = E(r^2)$ is average power of LOS component and m is nakagami m parameter

m=1 corresponds to Rayleigh distribution. When m increases, the number of gaussian random variables contributions increases and corresponds to faster decay. Small and large values of m correspond to urban and open areas repectively. The intermediate values of m correspond to rural and suburban areas. Thus Nakagami distribution is perfect to model any fading scenario.
Fig 5.1 shows a plot of simulation of Nakagami probability density function in MATLAB. The theoretical plot of pdf has been compared with the probability density plot of explicitly

generated Nakagami distributed Random Variables. Plots have been developed for m=0.5,m=1,m=4

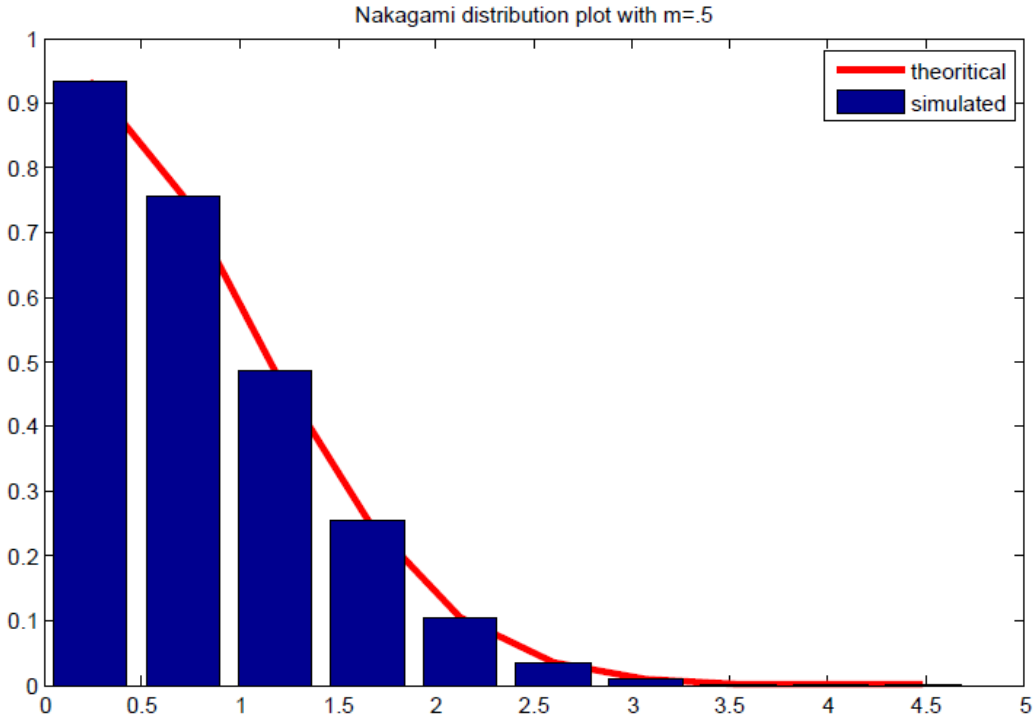

Fig 5.3: Simulated Nakagami Distribution Plot for m=0.5(One Sided Gaussian)

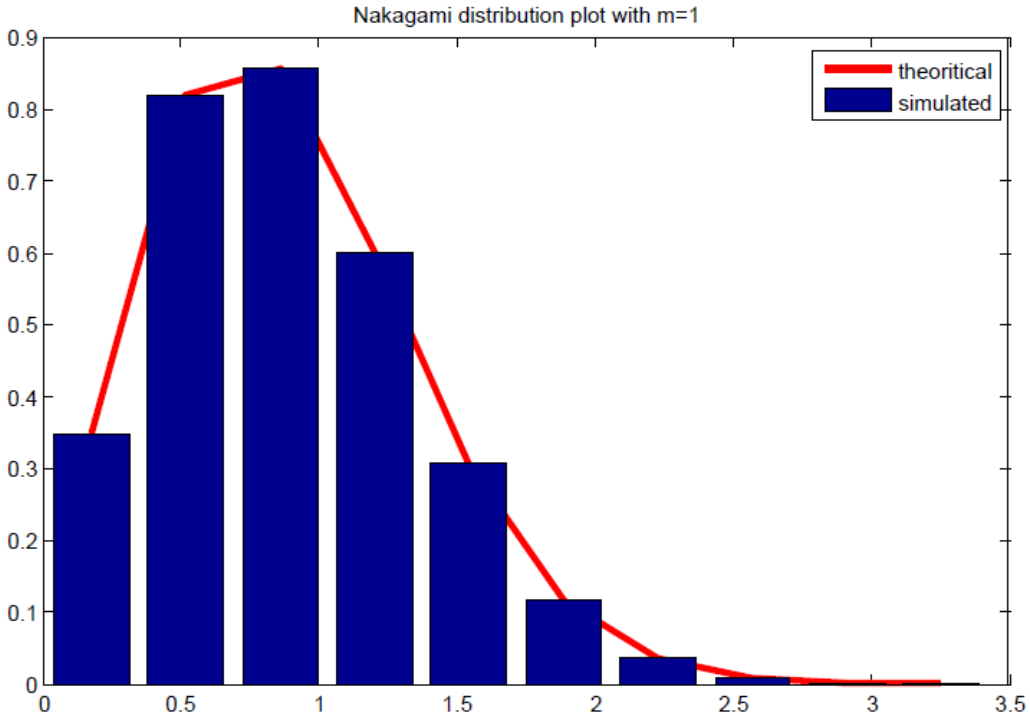

Fig 5.4: Simulated Nakagami Distribution Plot for m=1(Rayleigh)

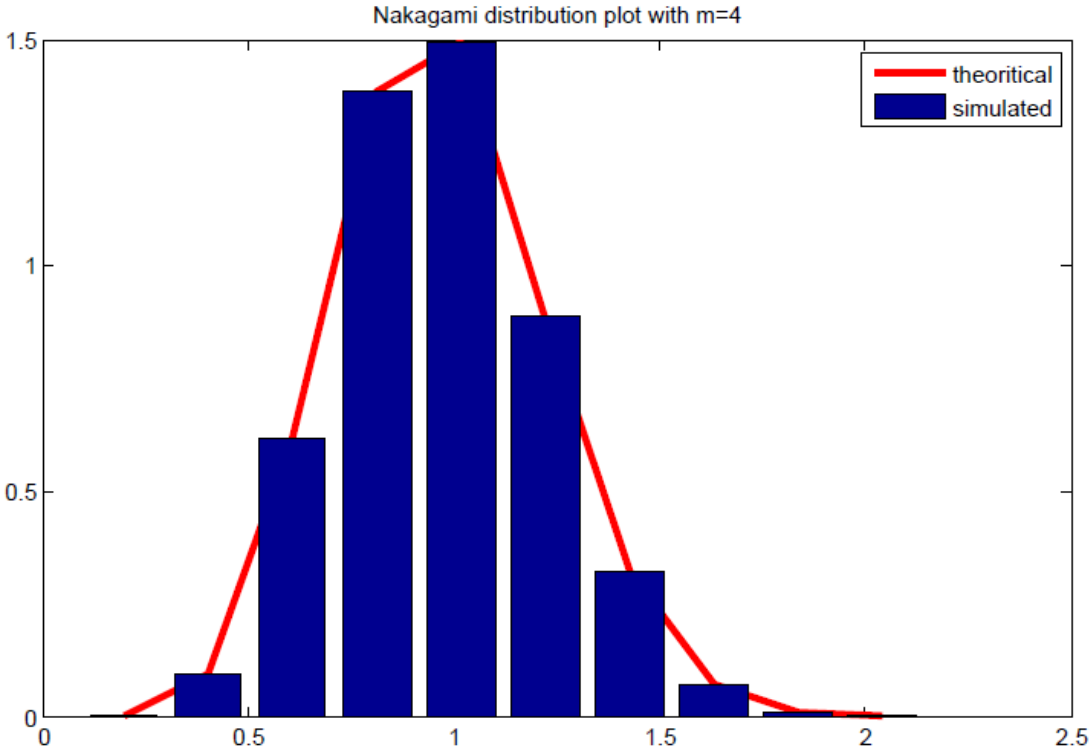

Fig 5.5: Simulated Nakagami Distribution Plot for m=4

Chapter-6

TURBO CODES PERFORMANCE

6.1 Encoding rate of Turbo code performance

One way of increasing the rate of turbo code is by puncturing the encoder outputs of the basic 1/3 rate turbo code. The method of puncturing of this rate 1/3 mother code to get higher rate codes is an important issue. To find the optimal puncturing pattern the technique is to test the decoder with different puncturing patterns.The puncturing pattern which gives the best BER performance is then selected.

The puncturing method most commonly done to obtain ½ code rate from 1/3 code rate decoder is to drop odd numbered parity bits from the first recursive systematic convolutional encoder and to drop even numbered parity bits of the second recursive systematic convolutional encoder.

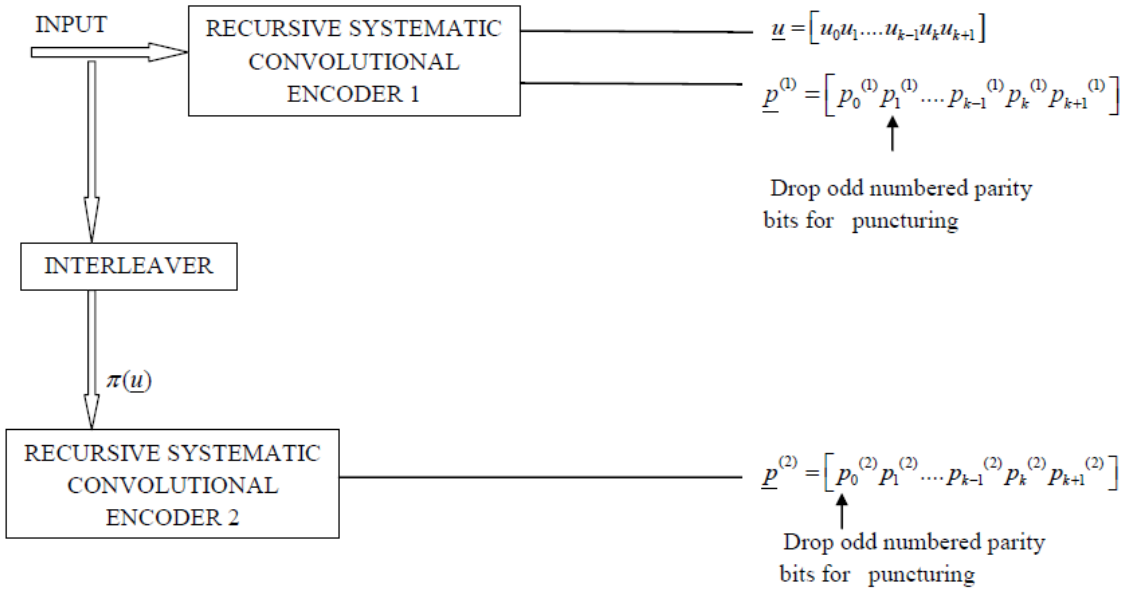

Fig 6.1 Puncturing Turbo Codes

The decoder remains the same and the punctured or missing bits are plugged in as zero by the decoder. Following plot shows the effect of puncturing. The simulation was done in

MATLAB in AWGN environment for 6 lakhs bits with a frame size of 6000 bits. Random interleaving was used.

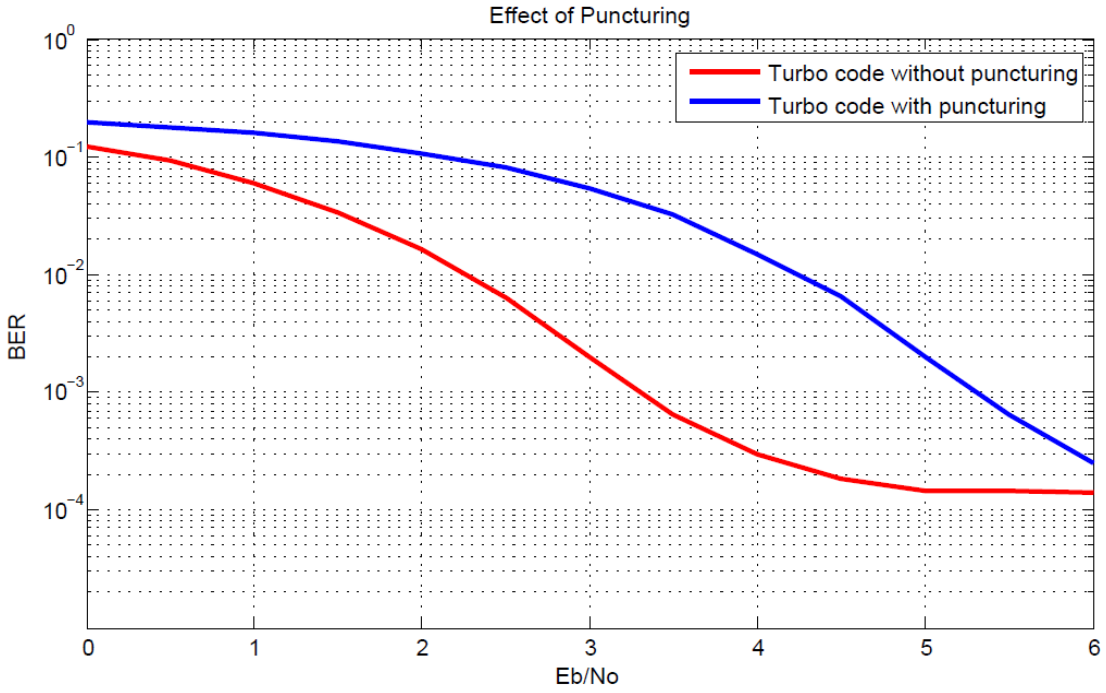

Fig 6.2: Simulated BER v/s Eb/No plot for Puntured Turbo Codes

6.2 Performance with Types of Interleaver

The operation of turbo codes is incomplete without interleavers. The coded information is reordered by the interleaver[11] and then transmitted over the channel. At the receiver, the deinterleaver performs the reverse operation to restore the original information back. In turbo codes the use of interleaver accomplishes mainly two functions. The interleaver constructs a long length code from small memory convolutional codes so that the codes can approach the Shannon capacity limit. Firstly it spreads out the burst errors so that code words appear independent. Thus the burst error channel is transformed into random error channel at the input of the decoder and the code designed for independent error channels could be used for burst error channels. The interleaver scrambles the data in such a way that the inputs to the two decoders are uncorrelated. So after corrections of some errors in the first component decoder, some of the remaining errors are spread by the interleaver in such a way that they get

corrected by the second decoder. On increasing the number of iterations in the decoding process the bit error probability approaches the channel capacity.

Generally the interleavers that are used of the pseudo random type or block interleaving type.

A block interleaver formats the input sequence of N bits in a matrix of m rows and n columns such that N=m*n. The input sequence is written into the matrix row wise and read out column wise as shown below

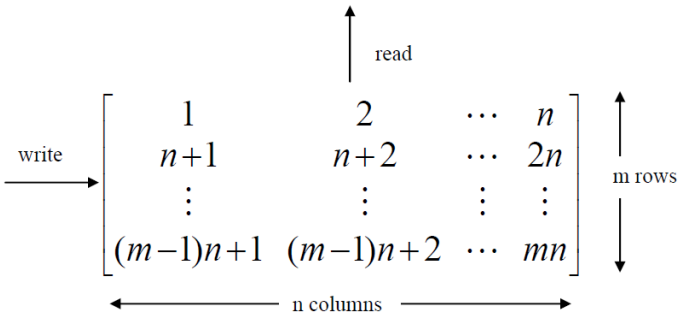

Fig 6.3: A Block interleaver

The deinterleaver stores the data into a matrix with the same format as interleaver. The data are read out and delivered to the decoder row wise. The performance of turbo decoder with random interleaving as compared to block interleaving is shown in the following figure.

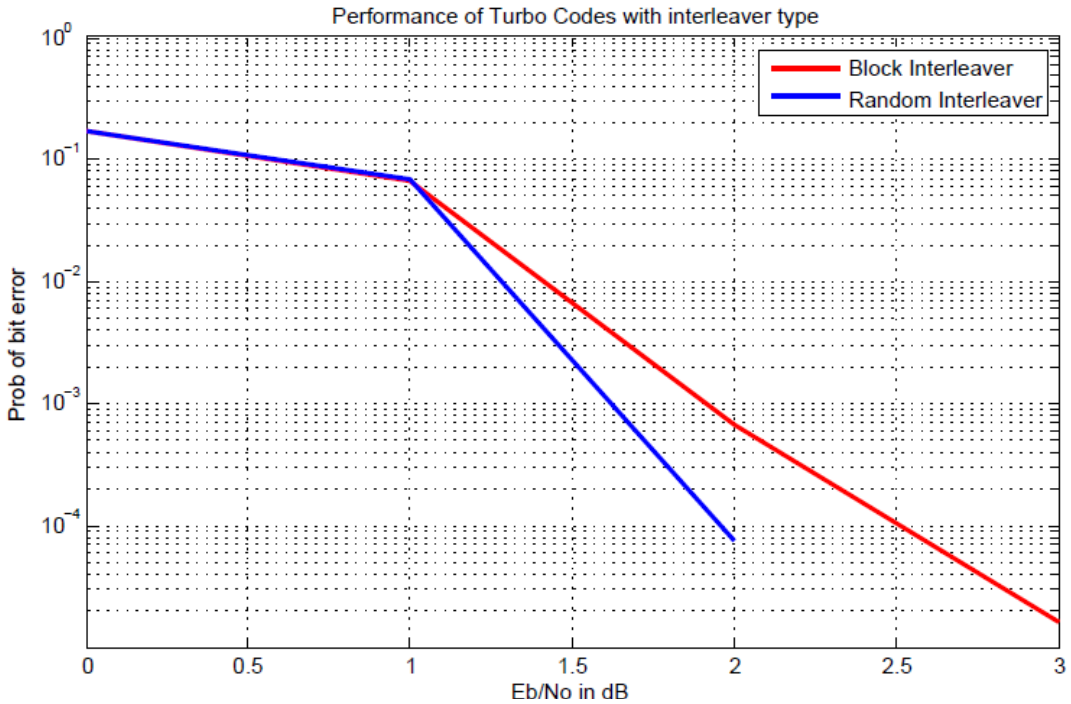

Fig 6.4: Simulated BER v/s Eb/No plot for turbo codes for interleaver types

The above simulation was done in Simulink model for 1048576 bits.A block interleaver of 1024*1024 was used.

6.3 Component codes of Turbo code performance

Under the same length of information sequence and the code rate, the performance of the turbo code varies with the constraint lengths of the encoder structure. This is displayed in the following simulation results. The simulation was done in Simulink model for 1048576 bits. Random interleaving was used.

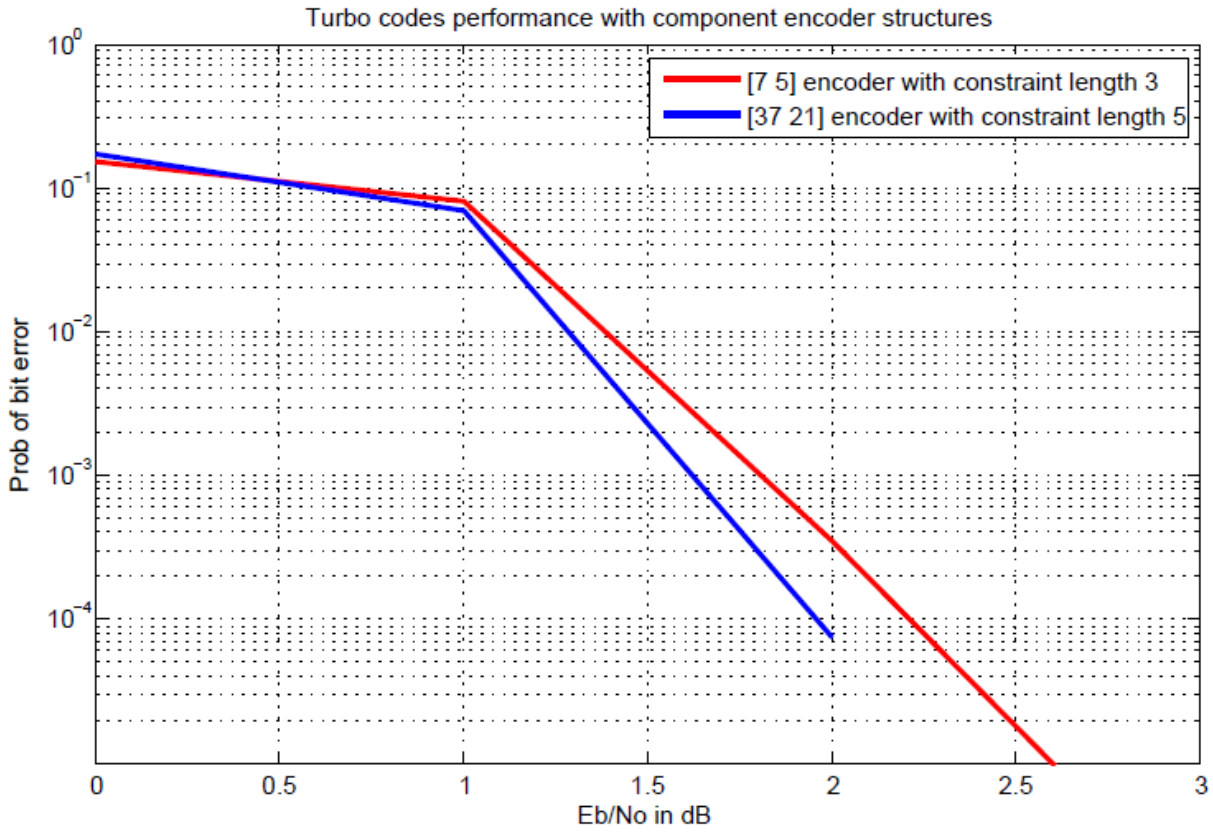

Fig 6.5: Simulated BER v/s Eb/No plot of Turbo Decoder with different encoder structures

The above result is quite evident as with the increase in constraint length of the encoder, the number of soft decision levels also increase and thus the soft decision decoding is also better. This improves the performance.

6.4 Performance on Fading channels

Because of multiplicity of factors involved in propagation in a cellular mobile environment, it is necessary to apply statistical techniques to describe signal variations. The fading analysis here has been done on Rayleigh and rician fading models.

6.4.1 Rayleigh fading analysis

In a common land mobile channel, the direct wave is obstructed and the mobile unit receives only reflected waves. When the number of reflected waves is large, according to the central limit theorem, two quadrature components of the channel impulse response will be uncorrelated Gaussian random processes with a zero mean and variance σ_s^2 . As a result, the envelope of the channel impulse response at any time instant undergoes a Rayleigh probability distribution and phase of the channel response obeys a uniform distribution between $-\pi$ and π . The probability density function (pdf) of the Rayleigh distribution [6] is given by

$$p(a) = \begin{cases} \frac{a}{\sigma_s^2}.e^{-a^2/\sigma_s^2} \\ 0 \end{cases}$$

The mean value denoted by m_a and variance denoted by σ_a^2 of the Rayleigh distributed random variable [3] is given by

$$m_a = \sqrt{\frac{\pi}{2}}.\sigma_s = 1.2533\sigma_s$$

$$\sigma_a^2 = \left(2 - \frac{\pi}{2}\right)\sigma_s^2 = 0.4292\sigma_s^2$$

If the probability density function is normalized so that the average signal power is unity, then the Rayleigh distribution will become

$$p(a) = \begin{cases} 2ae^{-a^2} \\ 0 \end{cases}$$

The mean and variance will be

$m_a = 0.8862$

$\sigma_a^{\,2} = 0.2146$

For analysis the Rayleigh fading of the above mean and variance is generated. The Rayleigh fading channel so generated is a kind of slow fading channel. Slow means that the fading bandwidth is small compared to the signal bandwidth so that the receiver will be able to track the phase variations enabling coherent detection at the receiver. A slow fading process can be modeled as a constant random variable during each symbol interval. This slow fading has a sort of multiplicative effect in contrast to additive effect of the Additive white Gaussian noise. Then the received signal can be modeled as

$r_t = a_t y_t + n_t$

Where a_t is a random variable which represents the channel fading variation and n_t is Gaussian noise with single sided power spectral density N_0 .The effect of the above equation is shown in fig 6.6.In this model, the envelope amplitude of the fading attenuation a_t is a Rayleigh random variable.

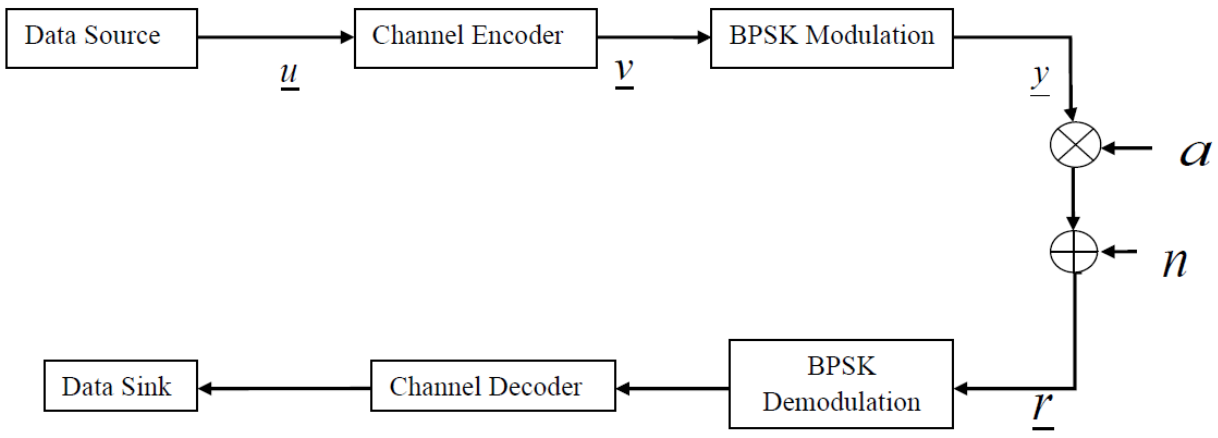

Fig 6.6: Channel Fading Model

The plot below shows the performance of Turbo decoder in Rayleigh fading environment. The result has been generated for 6 lakh bits with frame size equal to 6000 bits in MATLAB. Random interleaving was used.

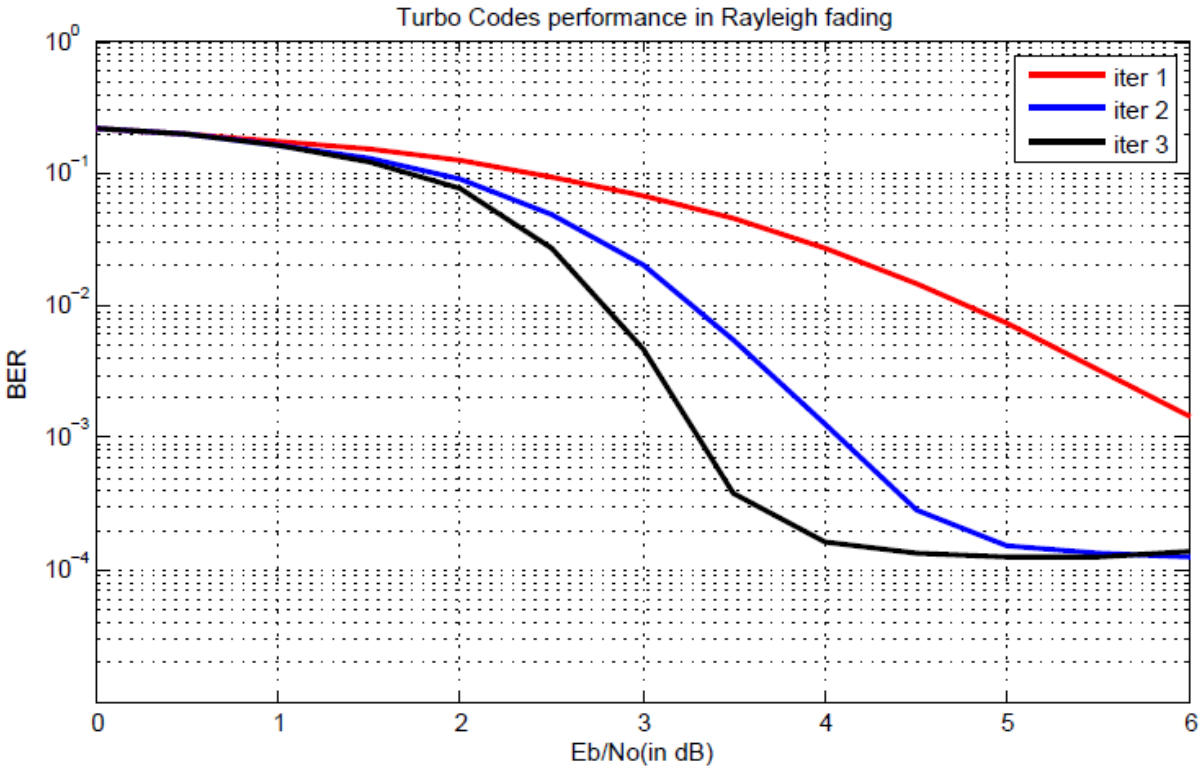

Fig 6.7: Simulated BER v/s Eb/No plot of Turbo Decoder in Rayleigh Fading

6.4.2 Rician Fading Analysis

As mentioned before, the sum of a constant direct signal and a Rayleigh distributed scattered signal results in a signal with rician envelope distribution. The pdf of Rician distribution is given by

$$p(a) = \begin{cases} \frac{a}{\sigma_s^2} e^{-\frac{(a^2+D^2)}{2\sigma_s^2}} I_0\left(\frac{aD}{\sigma_s^2}\right) \\ 0 \end{cases}$$

Where D^2 is the direct signal power and $I_0(.)$ is the modified Bessel function of the first kind and zero order. By assuming that the total average signal power is normalized to unity, the above pdf becomes

$$p(a) = \begin{cases} 2a(1+K)e^{-K-(1+K)a^2} I_0\left(2a\sqrt{K(K+1)}\right) \\ 0 \end{cases}$$

where K is the Rician factor [7] denoting the power ratio of the direct and the scattered signal components. The Rician factor is given by

$$K = \frac{D^2}{2\sigma_s^{\,2}}$$

The mean and variance of the Rician distributed random variable are given by

$$m_a = \frac{1}{2}\sqrt{\frac{\pi}{1+K}}\, e^{-\frac{K}{2}\left[(1+K)I_0\left(\frac{K}{2}\right)+KI_1\left(\frac{K}{2}\right)\right]}$$

$$\sigma_a^{\,2} = 1 - m_r^{\,2}$$

Where $I_0(.)$ is the first order modified Bessel function of the first kind. Small values of K indicate a severely faded channel. For small K around 0, there is no direct signal component and the rician pdf becomes a Rayleigh pdf. On the other hand a large value of K indicates a slightly faded channel and approximates an AWGN channel. The plot below shows the performance of Turbo decoder in Rician fading environment. The result has been generated in MATLAB for 6 lakh bits with frame size of 6000 bits. Random interleaving was used. The plot below is for K=100 which approximates the performance in AWGN environment.

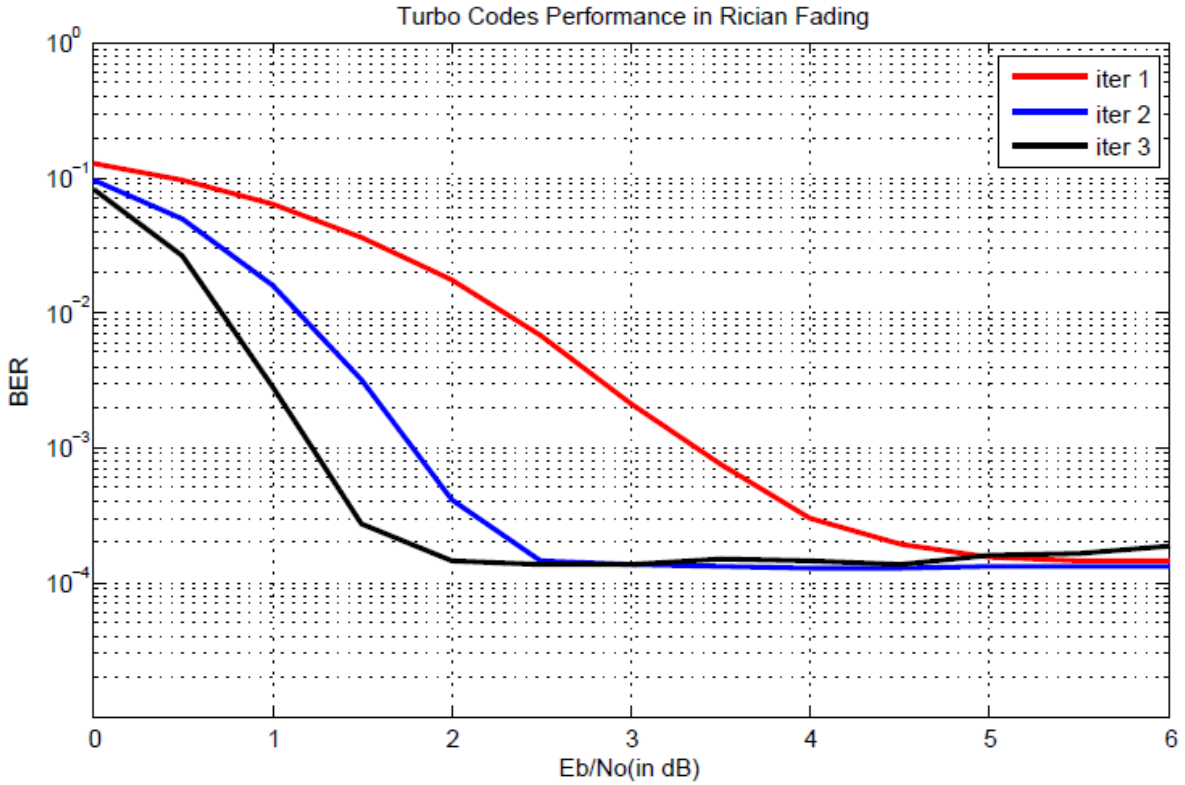

Fig 6.8: Simulated BER v/s Eb/No plot of Turbo Decoder in Rician Fading

Chapter-7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

A detailed description of convolutional codes, with a reference to viterbi decoding has been presented. BER performance of viterbi decoder was simulated. Detailed analysis of MAP decoding method was done along with the simulation of the performance of soft decoding done by it. The technique of turbo decoding was analyzed and its Encoder and decoder structure were presented. Through simulations it was shown that the BER performance of the turbo decoder improves with the number of iterations. Almost after 18 iterations the performance approached the Shannon limit and this was validated by simulations. Various fading that are most commonly encountered in urban scenarios were analyzed. Their theoretical probability density functions were simulated. They were compared against the probability density functions of the randomly generated discrete variables (of the given distribution) through simulations. The performance of the turbo decoders was then judged against various factors and situations. Analysis of the Performance of turbo codes were carried out for different encoder structures and interleaver types. The technique of puncturing was also studied and was simulated. The performance of turbo codes was simulated in Rayleigh and Rician fading channels. The turbo codes have shown a considerably remarkable performance with variations in parameters and environments.

7.2 Future Work

Some extensions related to the work of the thesis that needs to be done are

(1) Turbo decoding in fading environments needs to be carried out in the presence of fading environments with channel side information (i.e. CSI). The suitable modifications that need to be done in the constituent MAP decoders of the turbo decoder for decoding in fading environments with CSI needs to be studied and analyzed.

(2) Nakagami Distribution model is a general and universal model that can describe a wireless fading scenario in the best way. Through its m parameter it can span a range

of fading channels. Thus Nakagami distribution often gives the best fit to land-mobile and indoor-mobile multipath propagation. The Nakagami distribution model needs to be incorporated with Turbo decoder performance in fading environments such that Nakagami distribution appears the best fit model to describe any channel fading.

REFERENCES

[1] C. Berm, A. Galvieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes”, Proceedings of the ICC'93, pp.1064-1070, Geneva, Switzerland, May 1993.

[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate”, IEEE Transaction on Information Theory, vol.-20, pp. 284-287, Mar. 1974

[3]Branka Vucetic, Jinhong Yuan, “Turbo Codes Principles and Applications”, Kluwer Academic Publishers, 2000

[4]Bernard Sklar, “Digital Communications-Fundamentals and Applications”, Pearson Education, Second edition, 2003

[5]John G.Proakis, “Digital Communications”, McGraw Hill, Fourth edition, 2000

[6] Eric K. Hall and Stephen G. Wilson, Design and Analysis of Turbo Codes on Rayleigh Fading Channels, IEEE Journal on selected areas in communications, Vol. 16, No. 2, February 1998

[7]Xiangming Yu, Yanmin Kang, and Dongfeng Yuan, Senior Member, “Performance Analysis of Turbo Codes in Wireless Rician Fading Channel with Low Rician Factor”, 2010 IEEE

[8] Claude Berrou, Alain Glavieux, “Near Optimum Error Correcting coding and decoding: Turbo Codes”, IEEE Transactions on Communications, Vol. 44, No.10, October 1996

[9] Syed Amjad Ali, Erhan AliRiza ˙Ince, “Performance Analysis Of Turbo Codes Over Nakagami-m fading channels with impulsive noise”, 2007 IEEE

[10] Dhouha Kbaier Ben Ismail, Catherine Douillard, Sylvie Kerouedan, “A survey of three dimensional Turbo Codes and recent performance enhancements”, EURASIP Journal on Wireless Communications and Networking 2013

[11] Ajit Nimbalker, Yufei Blankenship, Brian Classon, T. Keith Blankenship, “ARP and QPP Interleavers for LTE Turbo Coding”, Wireless Communications and Networking Conference, 2008 IEEE

[12] Ramesh Pyndiah, Alain Glavieux, Annie Picart and Sylvie Jacq, “Near Optimum Decoding of product codes”, 1994 IEEE

[13] J.B. Anderson, S.M. Hladik, “MAP Tailbiting Decoders”, 1997 IEEE

[14] Ramesh Mahendra Pyndiah, “Near-Optimum Decoding of Product Codes: Block Turbo Codes”, IEEE Transactions On Communications, Vol. 46, No. 8, August 1998

[15]C.E Shannon, “Communication in the presence of Noise,” Proc. IRE, Vol. 37, no.1,pp. 10-21,Jan 1949

[16] Nambirajan Seshadri, Carl-Erik W. Sundberg,"List Viterbi Decoding Algorithms with Applications", IEEE Transactions On Communications,Vol. 42,1994

[17] www.nptel.iitm.ac.in