

# ITERATIVE FREQUENCY DOMAIN EQUALIZATION AND MAX-LOG-MAP DECODING

## A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

MASTER OF TECHNOLOGY

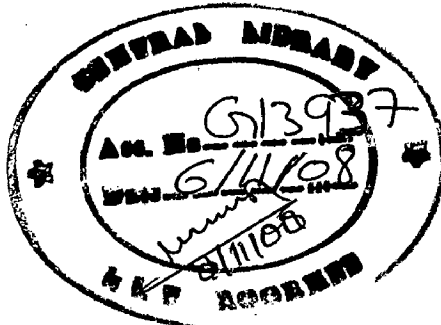
*in*

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Communication Systems)

*By*

**SRINIVAS D.J.**



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE - 247 667 (INDIA)

JUNE, 2008

## CANDIDATE'S DECLARATION

---

I hereby declare that the work, which is presented in this dissertation report, entitled “**ITERATIVE FREQUENCY DOMAIN EQUALIZATION AND MAX-LOG-MAP DECODING**”, being submitted in partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** with specialization in **COMMUNICATION SYSTEMS**, in the Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee is an authentic record of my own work carried out from June 2007 to June 2008, under guidance and supervision of **Dr. S. K. VARMA**, professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee.

The results embodied in this dissertation have not submitted for the award of any other Degree or Diploma.

Date 27<sup>th</sup> Jun 08

Place: Roorkee

  
(SRINIVAS D.J.)


## CERTIFICATE

---

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

Date: 27<sup>th</sup> Jun 08

Place: Roorkee

  
**Dr. S. K. VARMA**  
Professor, E&CE Department  
Indian Institute of technology, Roorkee  
Roorkee – 247667, (INDIA)

## ACKNOWLEDGEMENT

---

I would like to extend my heartfelt gratitude to my guide, **Dr. S.K.VARMA** for his able guidance, valuable suggestions and constant attention. It is his constant encouragement that inspired me throughout my dissertation work. I consider myself fortunate to have my dissertation done under him.

I am indebted to all my teachers who shaped and molded me. I would like to express my deep sense of gratitude to all the authorities of Department of Electronics and Computer Engineering, IIT Roorkee for providing me with the valuable opportunity to carry out this work and also providing me with the best of facilities for the completion of this work.

I am very thankful to my HOD, **Dr. D.K.MEHRA** and the staff of Signal Processing Lab for their constant cooperation and assistance.

I am indebted to my parents **SHIVAKANTH D.J** and **SHYAMALA D.J**, my sister **SUBHASHINI** and three important persons **NIHARIKA JADHAV**, **D. RAKESH KUMAR** and **K. PRASHANTH KUMAR**, for everything that they have done for me. They always supported me with their love, care and valuable advices. Thanks are also due to all my classmates who have helped me directly or indirectly in my work.

Last, but not the least my deepest gratitude to the Almighty God whose Divine grace provided me guidance, strength and support always.

**SRINIVAS D.J.**

## ABSTRACT

---

Multipath fading channels introduces intersymbol interference (ISI), which is not known a priori, as such we face problem of data transmission over such channels. To protect the integrity of the data a controlled amount of redundancy is added (encoding) using an error correction code (ECC). For coded data transmission over such channels Douillard et al. proposed the “turbo equalization” approach, which is an iterative equalization and decoding algorithm for receiver. The decoding strategy used to decode convolutional codes is the basic of ‘turbo principle’, which suggests the iterated exchange of soft information between different blocks of a communication receiver. The potential importance and applicability of this principle has been found to extend to a wide range of problems in communication. In iterative adaptive equalization and decoding, channel equalization and MAP decoding are jointly optimized in an iterative process. One drawback of this process is the exponentially increasing complexity of the equalization step. To overcome this, several equalizers applicable to “turbo equalization” requiring reduced complexity have been proposed. Furthermore, we present an alternate approach to rederive some of these approaches and equalize the received data in frequency domain.

In this dissertation, simulation of iterative frequency domain equalization and Max-Log-MAP decoding has been carried out in MATLAB environment. The performance has been evaluated using approximate MMSE Equalizer, approximate MMSE Equalizer with average variance and Max-Log-MAP algorithm for decoding.

# CONTENTS

---

<b>CANDIDATE'S DECLARATION</b> .....	i
<b>ACKNOWLEDGEMENTS</b> .....	ii
<b>ABSTRACT</b> .....	iii
<b>CONTENTS</b> .....	iv
<b>Chapter 1: INTRODUCTION</b> .....	1
1.1 Broadband Wireless Access .....	2
1.2 Statement of Problem .....	6
1.3 Organization of the Dissertation .....	6
<b>Chapter 2: TURBO DECODING SCHEME AND TURBO EQUALIZATION</b> .....	7
2.1 Transmission Scheme.....	7
2.2 Turbo Decoder.....	8
2.2.1 The Maximum a posteriori Algorithm .....	13
2.2.2 Calculation of the $\gamma_k(s', s)$ Values.....	16
2.3 Iterative Turbo Decoding Principles .....	17
2.4 Modifications of the MAP algorithm .....	21
2.5 MAP-Equalization Algorithm .....	22
2.6 Iterative MAP-Equalization and MAP-Decoding.....	25
<b>Chapter 3: TURBO EQUALIZATION USING MMSE EQUALIZER</b> .....	27
3.1 THE SISO EQUALIZER.....	27
3.2 Estimator Using Linear Equalizer .....	28
3.3 Turbo Equalization Using Frequency Domain Equalizer .....	32
3.4 Improved Max-Log-MAP Algorithm: The Log-MAP Extension.....	34
3.5 Interleaver Design .....	36

<b>Chapter 4: SIMULATION DETAILS</b> .....	39
4.1 Channel Encoder .....	39
4.2 Simulation of Decoder .....	40
<b>Chapter 5: SIMULATION RESULTS</b> .....	50
<b>Chapter 6: CONCLUSIONS AND FUTURE SCOPE</b> .....	60
<b>REFERENCES</b> .....	62

# Chapter 1

## INTRODUCTION

---

Today we can observe a transition in the way we are living. The so-called information society replaces the established industrial society. Accelerated by genius inventions as the Internet or the mobile phone, people demand to communicate more often, with more parties, at increasing data rates, and over longer distances. Where we mainly interacted with our voice in the past, today a broad range of data such as voice, images, video, or software is exchanged. Besides this, noncommercial applications such as science or the military call for new approaches in communication technologies, e.g., to transmit with less energy or more securely. But in order to use any of the communications services available today, several complex data transmission problems have to be solved. These include the transmission of digital data over broad ranges of analog channels, such as wireline (e.g., telephone line, coaxial cable), fibre optic, or wireless (radio, underwater acoustics) channels.

Wireless communications is, by any measure, the fastest growing segment of the communications industry. As such, it has captured the attention of the media and the imagination of the public. Cellular systems have experienced exponential growth over the last decade and there are currently around two billion users worldwide. Indeed, cellular phones have become a critical business tool and part of everyday life in most developed countries, and are rapidly replacing antiquated wireline systems in many developing countries. In addition, wireless local area networks currently supplement or replace wired networks in many homes, businesses, and campuses. Many new applications, including wireless sensor networks, automated highways and factories, smart homes and appliances, and remote telemedicine, are emerging from research ideas to concrete systems. The explosive growth of wireless systems coupled with the development of laptop and palmtop computers indicate a bright future for wireless networks, both as stand-alone systems and as part of the larger networking infrastructure. However, many technical challenges remain in designing robust wireless networks that deliver the performance necessary to support emerging applications.

The vision of wireless communications supporting information exchange between people or devices is the communications frontier of the next few decades, and much of it already exists in some form. This vision will allow multimedia communication from

anywhere in the world using a small handheld device or laptop. Wireless networks will connect palmtop, laptop, and desktop computers anywhere within an office building or campus, as well as from the corner cafe. In the home these networks will enable a new class of intelligent electronic devices that can interact with each other and with the Internet in addition to providing connectivity between computers, phones, and security/monitoring systems. Many technical challenges must be addressed to enable the wireless applications of the future. These challenges extend across all aspects of the system design. As wireless terminals add more features, these small devices must incorporate multiple modes of operation to support the different applications and media.

### **1.1 Broadband Wireless Access**

Broadband wireless access provides high-rate wireless communications between a fixed access point and multiple terminals. These systems were initially proposed to support interactive video service to the home, but the application emphasis then shifted to providing high speed data access (tens of Mbps) to the Internet, the WWW, and to high speed data networks for both homes and businesses.

WiMAX is an emerging broadband wireless technology based on the IEEE 802.16 standard. The core 802.16 specification is a standard for broadband wireless access systems operating at radio frequencies between 10 GHz and 66 GHz. Data rates of around 40 Mbps will be available for fixed users and 15 Mbps for mobile users, with a range of several kilometers. Many laptop and PDA manufacturers are planning to incorporate WiMAX once it becomes available to satisfy demand for constant Internet access and email exchange from any location. WiMAX will compete with wireless LANs, 3G cellular services, and possibly wireline services like cable and DSL. The ability of WiMAX to challenge or supplant these systems will depend on its relative performance and cost, which remain to be seen.

The wireless radio channel poses a severe challenge as a medium for reliable high-speed communication. It is not only susceptible to noise, interference, and other channel obstructions, but these change over time in unpredictable ways due to user movement. Variation due to multipath occurs over very short distances, on the order of the signal wavelength, so these variations are sometimes referred to as *small-scale propagation effects*.



These Broadband systems are likely to face unfriendly radio propagation environments, with multipath delay spread extending over tens or hundreds of bit intervals; interfering with the subsequently transmitted bits (This effect is called ISI) [1]. Choosing a suitable air interface technology is essential, in order to meet the required quality of service (QoS), and under various constraints such as bandwidth and equipment costs. The channel distortion resulting in ISI is not known a priori, which if left uncompensated, causes high error rates. The solution to ISI is to design a receiver that employs a means for compensating or reducing ISI in the receiver; this compensator for ISI is called an *equalizer*.

❖ *Classical Techniques for Equalization of Uncoded Data:*

Uncoded modulation does not use a coding step and simply feeds the data into a signal mapper and use a coding step and transmits the obtained, and in general complex, symbols over the channel. The ISI is removed through equalization. The data estimate is obtained from a signal mapper converting the hard decided equalized channel symbols to the input data alphabet. No channel coding (encoder/decoder) is used here.

Standard equalization techniques can be applied to this scenario such as a linear equalizer (LE) or a decision feedback equalizer (DFE) [1], [2]. The basic idea is to obtain a symbol estimate by filtering the received data (LE) or by also filtering the past symbol decisions (DFE). Both the LE and DFE contain linear filters as basic functional elements. The DFE also contains a nonlinearity, namely a hard decision element, to provide estimate of past symbols for feedback. To implement the linear filters, several structures such as transversal, cascade, parallel, or lattice filter implementations are available. The associated parameters to set up the filters are obtained using the channel response and cost criteria such as the zero forcing (ZF) or minimum mean squared error (MMSE) criterion [1].

Receivers using LE and DFE approaches are inherently suboptimal in terms of error probability since they are designed using different cost criteria. Also, some inherent weaknesses such as constraint filter lengths, noise enhancement in ZF-based solutions, or error propagation in DFE solutions limit the capabilities of pure LE or DFE approaches. Nonetheless, they are widely used in practice since a broad knowledge about them is available and the computational complexity is significantly smaller compared to optimal techniques.

For digital communication, a common cost criterion is the bit error rate (BER) of the system, which does not necessarily coincide with equalization criteria. The optimum receiver with respect to minimization of the BER is a maximum a-posteriori probability (MAP) or maximum likelihood (ML) detector. Such detectors does not remove ISI, but tries to find the most likely channel input given the output symbols disrupted by noise, which is equal to minimizing the BER. Whereas a ML detector assumes the symbols to be equally likely to occur on any value of the symbol alphabet, a MAP detector employs knowledge about the occurrence probability as well. An efficient implementation of the ML sequence estimation (MLSE) is the Viterbi algorithm (VA) [1], [3], which is described in standard literature about coding. A MAP symbol estimation (MAPSE) equalizer is often based on the forward-backward algorithm from Bahl et al. (BCJR) [4].

Unfortunately, all known MAP/ML-based methods suffer from high computational complexity with increasing channel length  $M$  and alphabet size  $q$  due to the exponential complexity  $O(q^M)$ . Furthermore, they can be applied directly only blockwise, since a time reversed backward step is involved.

❖ *Classical Techniques for Equalization of Coded Data:*

Significant improvements of the BER performance are possible using coding [3]. However, there is now a trade-off between achievable performance improvement and decreased data rate using a particular coding scheme.

Still an equalizer is required to accommodate ISI. The range of possible combinations of equalizers (LE, DFE, or ML/MAP) and coding schemes (e.g., block codes, convolutional codes) is virtually unlimited. The decoder is fed either with hard (decided) information (e.g., for algebraic block code decoding algorithms) or soft information (e.g., for ML/MAP detectors). Later decoding algorithms are standard for convolutional codes where the convolutional encoder is again analyzed as a finite state machine. An important result of the analysis of coded data transmission over ISI channels is that communicating soft information between the equalizer and the following decoder improves the receiver performance and surpasses similar receivers communicating hard information [5].

Some applications implement a coding step to overcome deficiencies of the chosen equalizer, e.g., the use of a DFE and low-redundancy coding to deal with the error propagation of the DFE [6].

A variety of modern techniques employs a convolutional code and a ML/MAP equalizer. A significant amount of research has been done on these schemes. One of the major challenges is to decrease the computational complexity of both the ML/MAP equalizer and the convolutional decoder, which is typically also applying a ML/MAP detector, using suboptimal algorithms, and efficient use of the equalizer soft output [7]. State-of-the-art systems include an interleaver after the encoder and a deinterleaver before the decoder to further improve the BER [7]. Interleaving shuffles symbols within a given time frame or block of data and thus decorrelates error events introduced by the equalizer between neighboring symbols. The interleaver will play a central role later for iterative solutions.

Finally, Ariyavisitakul and Li [8] proposed a joint coding-equalization approach working with convolutional coding and a DFE. Here, within the DFE, soft information from the DFE forward filter output and tentative (hard) decisions from the following decoder using the VA are feedback. This system can be thought of as a link between separate receiver approaches, where equalization and decoding are distinct tasks performed only once, and joint equalization-decoding schemes, where the latter use extensive feedback of information from the decoder.

❖ *Iterative (Turbo) Techniques:*

In previous two sections all ingredients to build an iterative system have been considered. These include the use of coding in addition to equalization, soft information for communication between the receiver functional blocks, and interleaving. An iterative receiver algorithm processes the received data by at least two distinct processing blocks, e.g., an equalizer and a decoder, interacting with each other in both directions. Turbo Equalization is an Iterative receiver algorithm for coded data transmission applying the “Turbo Principle”, where equalization and decoding tasks are repeated on the same set of received data while feedback information from the decoder is incorporated into the equalization process until a termination or convergence criterion is matched. The “Turbo Principle” was originally developed for concatenated convolutional codes (“Turbo Coding” [9]) and is now adapted to various communication problems such as trellis coded modulation (TCM) and Code Division Multiple Access (CDMA).

## **1.2 Statement of Problem**

A pure MAP-based solution for equalization and decoding is fastest in the bit error rate (BER) improvement over successive iterations and achieves these improvements at the lowest signal-to-noise ratios (SNRs) compared to all other techniques. But these ML/MAP-based solutions suffer from the high computational load for long channels (expensive equalizer) or convolutional codes with high memory order (expensive decoder). This situation is worsened since the equalization and decoding steps are performed several times for each block of data. Even with sophisticated algorithms, or suboptimal solutions, the inherent exponential complexity of the ML/MAP techniques limits their range of possible applications.

In this dissertation, the performance of Approximate Linear Equalization working in the frequency domain, applied to Turbo Equalization has been evaluated for ISI channels using simulation techniques. Max-Log-Map algorithm is used for decoding process.

## **1.3 Organization of the Dissertation**

Chapter 2 discusses the principle of turbo decoding scheme and Turbo-equalization. The transmission scheme, turbo decoder, MAP-algorithm, Iterative Turbo Decoding Principle, Improved MAP algorithm, and Iterative MAP-Equalization and MAP-Decoding algorithm are described in detail. In chapter 3, we describe about SISO Linear MMSE Estimator, Linear MMSE equalization is based on this estimator and also two different schemes of iterative equalization and MAP-decoding are discussed:

- Approximate Linear MMSE Equalization and
- Modified Approximate Linear MMSE Equalization with Average Variance.

Turbo equalization using Frequency domain equalizer is also described which is followed by improved Max-Log-MAP algorithm and interleaver design.

The simulation approach adopted for implementation of above mentioned algorithms is presented in chapter 4. The simulation results are presented in chapter 5. Finally we conclude our work in Conclusion and Future Scope in chapter 6.

## Chapter 2

# TURBO DECODING SCHEME AND TURBO EQUALIZATION

---

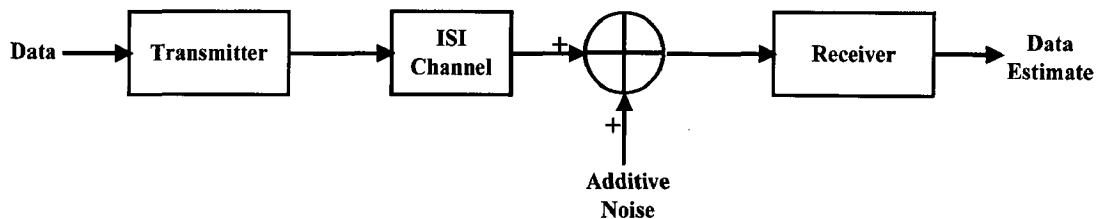
In this chapter, we first present the transmission scheme for turbo equalization and then introduce the turbo decoding scheme. Turbo decoding and turbo equalization algorithm are then presented in details.

### 2.1 Transmission Scheme

We Assume a coherent, symbol-spaced receiver front-end and precise knowledge of the signal phase and symbol timing, such that the channel can be approximated by an equivalent, discrete-time, baseband model, as shown in Fig: 2.1, where the transmit filter, the channel, and the receive filter are represented by a discrete-time *linear* filter, with the finite-length impulse response(FIR) [10]

$$h[n] = \sum_{k=0}^{M-1} h_k \delta[n-k] \quad (2.1)$$

of length  $M$ . The coefficients  $h_k$  are assumed to be time-invariant and known to the receiver.



**Fig: 2.1** Representation of a Data Transmission System

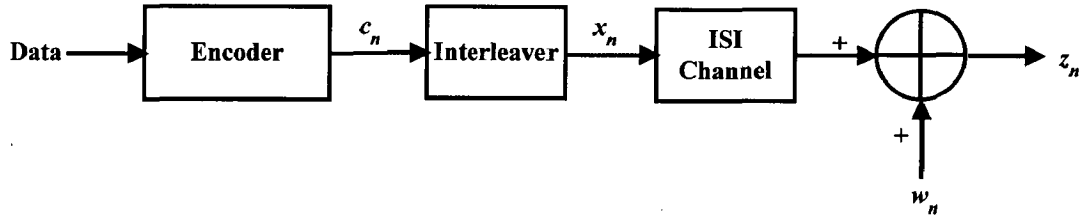
To simplify the derivations, all the systems to be investigated contain the same transmitter shown together with the ISI channel in Fig: 2.2. The binary data is encoded with a binary convolutional encoder yielding the code symbols  $c_n$ , which are mapped to the alphabet  $\mathcal{E}$  of the signal constellation. For simplicity we assume binary phase shift keying (BPSK), i.e.  $\mathcal{E} = \{+1, -1\}$ , and that the channel impulse response coefficients  $h_k$  and noise samples  $\omega_n$  are real valued.

The transmission and receiving tasks are applied to blocks of data bits  $b_i \in \{0, 1\}$  of length  $K_d$ . They are encoded to  $K_c = K_d/R + K_o$  code symbols  $c_n, n = 1, 2, \dots, K_c, c_n \in \mathcal{E}$ , where  $R \in [0,1]$  is the code rate and  $K_o \geq 0$  is any overhead introduced by the encoder, for e.g. termination sequence. The interleaver permutes the  $c_n$  and outputs  $K_c$  symbols  $x_n, n = 1, 2, \dots, K_c, x_n \in \mathcal{E}$ , to be transmitted over the ISI channel. This operation is denoted  $x_n = \Pi(c_n)$ , where  $\Pi(\cdot)$  is a fixed random permutation on  $K_c$  elements. The permutation  $\Pi^{-1}(\cdot)$ , the deinterleaver, reverses the  $\Pi(\cdot)$  operation. The noise is modeled as additive white Gaussian noise (AWGN), i.e., the noise samples  $\omega_n$  are independent and identically distributed (i.i.d) with normal probability density function (pdf)

$$f_\omega(\omega) = \phi(\omega / \sigma_\omega) / \sigma_\omega \quad (2.2)$$

and independent of data, where  $\phi(x) = e^{-x^2/2} / \sqrt{2\pi}$ . Given (2.1), the receiver input  $z_n$  is given by,

$$z_n = \left( \sum_{k=0}^{M-1} h_k x_{n-k} \right) + \omega_n \quad (2.3)$$



**Fig: 2.2** Transmitter section of the data transmission scheme

## 2.2 Turbo Decoder

Turbo decoding is accomplished by more than one component (often two) decoders. Special decoding algorithms must be used which accept soft inputs and give soft outputs for the decoded sequence [11]. These soft inputs and outputs provide not only an indication of whether a particular bit was a 0 or a 1, but also a likelihood ratio which gives the probability that the bit has been correctly decoded. The turbo decoder operates iteratively. In the first iteration the first component decoder provides a soft output giving an estimation of the original data sequence based on the soft channel inputs alone. It also provides an *extrinsic* output. The extrinsic output for a given bit is based not on the channel input for that bit, but on the information for surrounding bits and the constraints imposed by the code being used. This extrinsic output from the first decoder is used by the second component decoder as *a-priori information*, and this information together

with the channel inputs are used by the second component decoder to give its soft output and extrinsic information. In the second iteration the extrinsic information from the second decoder in the first iteration is used as the a-priori information for the first decoder, and using this a-priori information the decoder can hopefully decode more bits correctly than it did in the first iteration. This cycle continues, with at each iteration both component decoders producing a soft output and extrinsic information based on the channel inputs and a-priori information obtained from the extrinsic information provided by the previous decoder. After each iteration the Bit Error Rate (BER) in the decoded sequence drops, but the improvements obtained with each iteration fall as the number of iterations increases so that for complexity reasons usually only between 4 and 12 iterations are used.

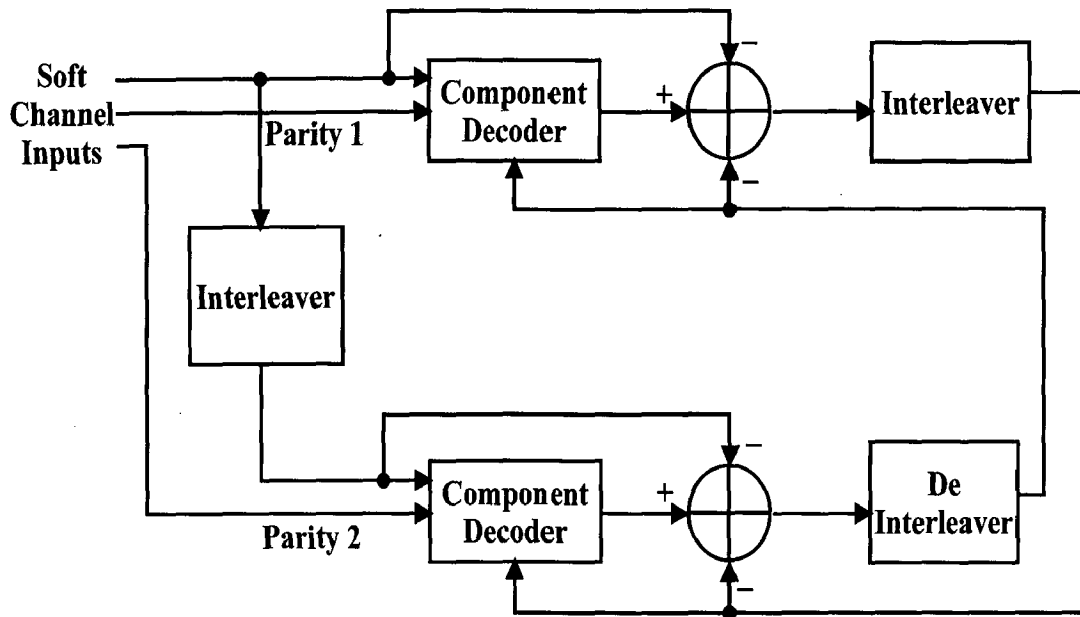


Fig: 2.3 Turbo Decoder Schematic

The general structure of an iterative turbo decoder is shown in Fig: 2.3. Two component decoders are linked by interweavers in a structure similar to that of the encoder. As seen in the figure, each decoder takes three inputs, the systematically encoded channel output bits, the parity bits transmitted from the associated component encoder, and the information from the other component decoder about the likely values of the bits concerned. This information from the other decoder is referred to as a-priori information. The component decoders have to exploit both the inputs from the channel and this a-priori information. They must also provide what are known as soft outputs for

the decoded bits. Two suitable decoders are the so-called SOVA proposed by Hagenauer and Hoher and the MAP algorithm of Bhal.

The decoder of Fig: 2.3 operates iteratively, and in the first iteration the first component decoder takes channel output values only, and produces a soft output as its estimate of the data bits. The soft output from the first encoder is then used as additional information for the second decoder, which uses this information along with the channel outputs to calculate its estimate of the data bits. Now the second iteration can begin, and the first decoder decodes the channel outputs again, but now with additional information about the value of the input bits provided by the output of the second decoder in the first iteration. This additional information allows the first decoder to obtain a more accurate set of soft outputs, which are then used by the second decoder as *a-priori* information. This cycle is repeated, and with every iteration the BER of the decoded bits tends to fall. However, the improvement in performance obtained with increasing numbers of iterations decreases as the number of iterations increases. Hence, for complexity reasons, usually only about eight iterations are used.

The soft outputs from the component decoder are typically represented in terms of the so-called Log Likelihood Ratios (LLRs). The polarity of the LLR determines the sign of the bit, while its amplitude quantifies the probability of a correct decision. The LLRs are simply, as their name implies, the logarithm of the ratio of two probabilities. For example, the LLR  $L(u_k)$  for the value of a decoded bit  $u_k$  is given by

$$L(u_k) = \ln \left( \frac{P(u_k = +1)}{P(u_k = -1)} \right) \quad (2.4)$$

where  $P(u_k) = +1$  is the probability that the bit  $u_k = +1$ , and similarly for  $P(u_k) = -1$ . Notice that the two possible values of the bit are taken to be  $+1$  and  $-1$ , rather than  $1$  and  $0$ , as this simplifies the derivations that follow. Fig: 2.4 shows how the LLR  $L(u_k)$  of a bit  $u_k = +1$  varies. It can be seen from this figure that the sign of the LLR  $L(u_k)$  of a bit  $u_k$  will indicate whether the bit is more likely to be  $+1$  or  $-1$ , and the magnitude of the LLR gives an indication of how likely it is that the sign of the LLR gives the correct value of  $u_k$ . When the LLR  $L(u_k) \approx 0$ , we have  $P(u_k = +1) \approx P(u_k = -1) \approx 0.5$ , and we cannot be certain about the value of  $u_k$ . Conversely, when  $L(u_k) \gg 0$ , we have  $P(u_k = +1) \gg P(u_k = -1)$  and we can be almost certain that  $u_k = +1$ .



Given the LLR  $L(u_k)$ , it is possible to calculate the probability that  $u_k = +1$  or  $u_k = -1$  as follows. Remembering that  $P(u_k = -1) = 1 - P(u_k = +1)$ , and taking the exponent of both sides in Eq: 2.4, we can write:

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)} \quad (2.5)$$

So,

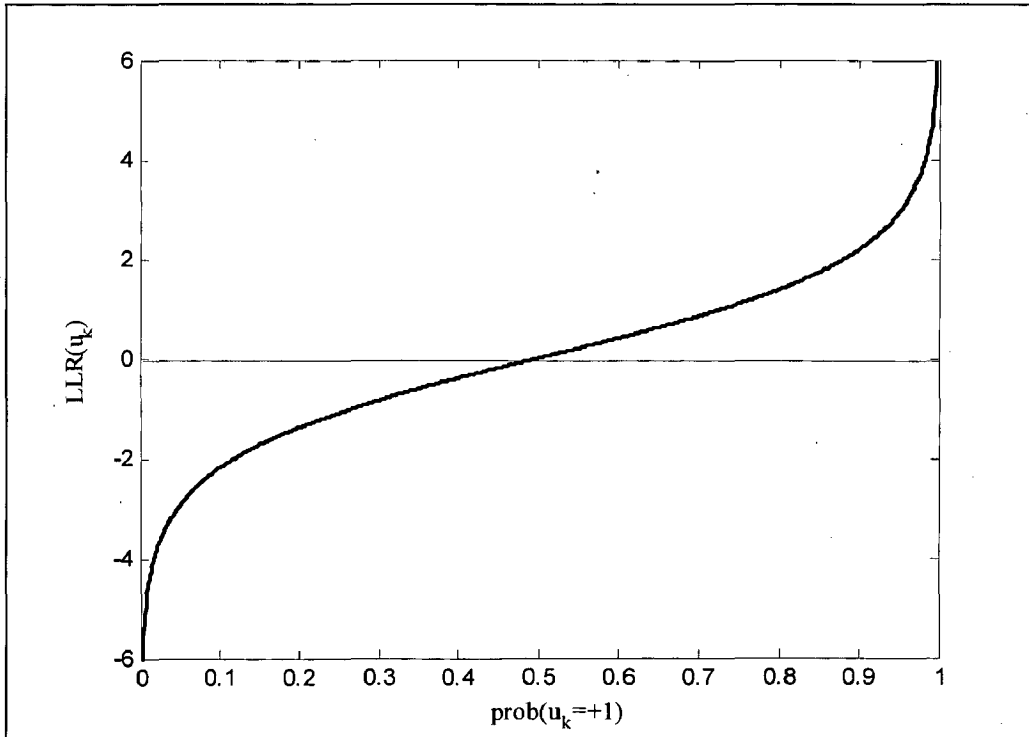
$$\begin{aligned} P(u_k = +1) &= \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} \\ &= \frac{1}{1 + e^{-L(u_k)}} \end{aligned} \quad (2.6)$$

Similarly:

$$\begin{aligned} P(u_k = -1) &= \frac{1}{1 + e^{L(u_k)}} \\ &= \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}}, \end{aligned} \quad (2.7)$$

and hence we can write:

$$P(u_k = \pm 1) = \left( \frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)/2}} \right) e^{\pm L(u_k)/2}. \quad (2.8)$$



**Fig: 2.4** LLR  $L(u_k)$  versus the probability of  $u_k = +1$

Apart from LLRs based on unconditional probabilities we are also interested in LLRs based on conditional probabilities. For example, in channel coding theory we are

interested in the probability that  $u_k = \pm 1$  based, or conditioned, on some received sequence  $\underline{z}$ , and hence we may use the conditional LLR  $L(u_k | \underline{z})$ , which is defined as,

$$L(u_k | \underline{z}) \triangleq \ln \left( \frac{P(u_k = +1 | \underline{z})}{P(u_k = -1 | \underline{z})} \right) \quad (2.9)$$

The conditional probabilities are known as the *a-posteriori* probabilities of the decoded bit  $u_k$ , and it is these a-posteriori probabilities that the component soft-in soft-out decoders attempt to find.

Apart from the conditional LLR based on the a-posteriori probabilities, we will also use conditional LLRs based on the probability that the receiver's matched filter output would be  $z_k$  given that the corresponding transmitted bit  $x_k$  was either +1 or -1. This conditional LLR is written as  $L(z_k | x_k)$  and is defined as:

$$L(z_k | x_k) \triangleq \ln \left( \frac{P(z_k | x_k = +1)}{P(z_k | x_k = -1)} \right) \quad (2.10)$$

If we assume that the transmitted bit  $x_k = \pm 1$  has been sent over a Gaussian or fading channel using BPSK modulation, then we can write for the probability of the matched filter output  $z_k$  that:

$$P(z_k | x_k = +1) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left( -\frac{E_b}{2\sigma^2} (z_k - a)^2 \right) \cdot dz_k, \quad (2.11)$$

where  $E_b$  is the transmitted energy per bit,  $\sigma^2$  is the noise variance and  $a$  is the fading amplitude ( $a=1$  for non-fading AWGN channels). Similarly, we have:

$$P(z_k | x_k = -1) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left( -\frac{E_b}{2\sigma^2} (z_k + a)^2 \right) \cdot dz_k. \quad (2.12)$$

Therefore, when we use BPSK over a (possibly fading) Gaussian channel, we can rewrite Eq: 2.10 as:

$$\begin{aligned} L(z_k | x_k) &= \ln \left( \frac{P(z_k | x_k = +1)}{P(z_k | x_k = -1)} \right) \\ &= \ln \left( \frac{\exp \left( -\frac{E_b}{2\sigma^2} (z_k - a)^2 \right)}{\exp \left( -\frac{E_b}{2\sigma^2} (z_k + a)^2 \right)} \right) \end{aligned}$$

$$\begin{aligned}
&= \left( -\frac{E_b}{2\sigma^2}(z_k - a)^2 \right) - \left( -\frac{E_b}{2\sigma^2}(z_k + a)^2 \right) \\
&= \frac{E_b}{2\sigma^2} 4a \cdot z_k \\
&= L_c \cdot z_k
\end{aligned} \tag{2.13}$$

where,  $L_c$  is defined as channel reliability value, and depends only on the signal-to-noise ratio (SNR) and fading amplitude of the channel. Hence, for BPSK over a (possibly fading) Gaussian channel, the conditional LLR  $L(z_k | x_k)$ , which is referred as the soft output of the channel, is simply the matched filter output  $z_k$  multiplied by the channel reliability value  $L_c$ .

Having introduced to LLRs, we now proceed to describe the operation of the MAP algorithm, which is one of the possible SISO component decoders that can be used in an iterative turbo decoder.

### 2.2.1 The Maximum a posteriori Algorithm

In 1974 Bahl, Cocke, Jelinek, and Raviv [12] introduced a MAP decoder, called the *BCJR algorithm*, that can be applied to any linear code, block or convolutional. The computational complexity of the BCJR algorithm is greater than that of the Viterbi algorithm, and thus Viterbi decoding is preferred in the case of equally likely information bits. When the information bits are not equally likely, however, better performance is achieved with MAP decoding. Also, when iterative decoding is employed, and the a priori probabilities of the information bits change from iteration to iteration, a MAP decoder gives the best performance.

The MAP algorithm gives, for each decoded bit  $u_k$ , the probability that this bit was +1 or -1, given the received symbol sequence  $\underline{z}$ . As explained in previous section this is equivalent to finding the a-posteriori LLR  $L(u_k | \underline{z})$ , where:

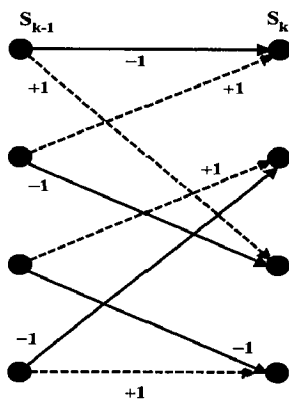
$$L(u_k | \underline{z}) = \ln \left( \frac{P(u_k = +1 | \underline{z})}{P(u_k = -1 | \underline{z})} \right) \tag{2.14}$$

Baye's rule allows us to rewrite this equation as:

$$L(u_k | \underline{z}) = \ln \left( \frac{P(u_k = +1 \wedge \underline{z})}{P(u_k = -1 \wedge \underline{z})} \right) \tag{2.15}$$

where by definition  $P(a \wedge b) = P(a | b) \cdot P(b)$  is the *joint probability* of a and b.

Let us now consider Fig: 2.5 showing the transitions possible for a constraint length two encoder, which has four encoder states, and since we consider a binary code, in each encoder state two state transitions are possible. One of these transitions is associated with the input bit of -1 shown as continuous line, while other transition corresponds to the input bit of +1 shown as a broken line. It can be seen from Fig: 2.5 that if the previous state  $S_{k-1}$  and the present state  $S_k$  are known, then the value of the input bit  $u_k$ , which caused the transition between these two states, will be known. For each  $u_k = \pm 1$ , there are



**Fig: 2.5** Possible Transitions for an Encoder with Constraint Length Two

four possible transitions that can occur and this set of transitions is mutually exclusive. Hence we can rewrite Eq: 2.15 as:

$$L(u_k | \underline{z}) = \ln \left( \frac{\sum_{\substack{(s',s) \Rightarrow \\ u_k = +1}} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{z})}{\sum_{\substack{(s',s) \Rightarrow \\ u_k = -1}} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{z})} \right) \quad (2.16)$$

$$= \ln \left( \frac{\sum_{\substack{(s',s) \Rightarrow \\ u_k = +1}} P(s' \wedge s \wedge \underline{z})}{\sum_{\substack{(s',s) \Rightarrow \\ u_k = -1}} P(s' \wedge s \wedge \underline{z})} \right) \quad (2.17)$$

The received sequence  $\underline{z}$  can be split up into three sections: the received codeword associated with the present transition  $\underline{z}_k$ , the received sequence prior to the present transition  $\underline{z}_{j < k}$  and the received sequence after the present transition  $\underline{z}_{j > k}$ . We can thus write for the individual probabilities  $P(s' \wedge s \wedge \underline{z})$ :

$$P(s' \wedge s \wedge \underline{z}) = P(s' \wedge s \wedge \underline{z}_{j < k} \wedge \underline{z}_k \wedge \underline{z}_{j > k}) \quad (2.18)$$

Using Baye's rule and the fact that if we assume that the channel is memoryless, then the future received sequence  $\underline{z}_{j>k}$  will depend only on the present state  $s$  and not on the previous state  $s'$  or the present and previous received channel sequences  $\underline{z}_k$  and  $\underline{z}_{j<k}$ , we can write:

$$\begin{aligned}
P(s' \wedge s \wedge \underline{z}) &= P(\underline{z}_{j>k} | \{s' \wedge s \wedge \underline{z}_{j<k} \wedge \underline{z}_k\}) \cdot P(s' \wedge s \wedge \underline{z}_{j<k} \wedge \underline{z}_k) \\
&= P(\underline{z}_{j>k} | s) \cdot P(s' \wedge s \wedge \underline{z}_{j<k} \wedge \underline{z}_k) \cdot \\
&= P(\underline{z}_{j>k} | s) \cdot P(\{\underline{z}_k \wedge s\} | \{s' \wedge \underline{z}_{j<k}\}) \cdot P(s' \wedge \underline{z}_{j<k}) \\
&= P(\underline{z}_{j>k} | s) \cdot P(\{\underline{z}_k \wedge s\} | s') \cdot P(s' \wedge \underline{z}_{j<k}) \\
&= \beta_k(s) \cdot \gamma_k(s', s) \cdot \alpha_{k-1}(s')
\end{aligned} \tag{2.19}$$

where  $\alpha_{k-1}(s')$  is the probability that the trellis is in state  $s'$  at time  $k-1$  and the received channel sequence up to this point is  $\underline{z}_{j<k}$ ,  $\beta_k(s)$  is the probability that given the trellis is in state  $s$  at time  $k$  the future received channel sequence will be  $\underline{z}_{j>k}$ , and lastly  $\gamma_k(s', s)$  is the probability that given the trellis was in state  $s'$  at time  $k-1$ , it moves to state  $s$  and the received channel sequence for this transition is  $\underline{z}_k$ .

We can now rewrite the expression for the probability  $\alpha_k(s)$  as,

$$\begin{aligned}
\alpha_k(s) &= P(S_k = s \wedge \underline{z}_{j<k+1}) \\
&= P(s \wedge \underline{z}_{j<k} \wedge \underline{z}_k) \\
&= \sum_{all s'} P(s \wedge s' \wedge \underline{z}_{j<k} \wedge \underline{z}_k) \\
&= \sum_{all s'} P(\{s \wedge \underline{z}_k\} | \{s' \wedge \underline{z}_{j<k}\}) \cdot P(s' \wedge \underline{z}_{j<k}) \\
&= \sum_{all s'} P(\{s \wedge \underline{z}_k\} | s') \cdot P(s' \wedge \underline{z}_{j<k}) \\
&= \sum_{all s'} \alpha_{k-1}(s') \cdot \gamma_k(s', s)
\end{aligned} \tag{2.20}$$

Thus, we can compute a *forward metric*  $\alpha_k(s)$  for each state  $s$  at time  $k$  using *forward recursion* (2.20). Similarly, we can compute a *backward metric*  $\beta_{k-1}(s')$  for each state  $s'$  at time  $k$  using *backward recursion* as,

$$\begin{aligned}
\beta_{k-1}(s') &= P(\underline{z}_{j>k-1} | s') \\
&= \sum_{all s} \beta_k(s) \cdot \gamma_k(s', s).
\end{aligned} \tag{2.21}$$

The forward recursion begins at time  $k=0$  with initial condition

$$\alpha_0(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$

since the encoder starts at all-zero state  $S_0 = \mathbf{0}$ .

Similarly, the backward recursion begins at time  $k=K$  ( $K$  is the length of the input sequence) with initial condition

$$\beta_K(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases}$$

since the encoder ends in the all-zero state  $S_0 = \mathbf{0}$ .

### 2.2.2 Calculation of the $\gamma_k(s', s)$ Values

Using the definition of  $\gamma_k(s', s)$  from Eq: 2.19 and Baye's rule we have:

$$\begin{aligned} \gamma_k(s', s) &= P(\{\underline{z}_k \wedge s\} | s') \\ &= P(\underline{z}_k | \{s' \wedge s\}) \cdot P(s | s') \\ &= P(\underline{z}_k | \{s' \wedge s\}) \cdot P(u_k), \end{aligned} \quad (2.22)$$

From Eq: 2.8 we have

$$\begin{aligned} P(u_k) &= \left. \begin{aligned} &\left( \frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)/2}} \right) \cdot e^{(u_k L(u_k)/2)} \\ &= C_{L(u_k)}^{(1)} \cdot e^{(u_k L(u_k)/2)} \end{aligned} \right\} \end{aligned} \quad (2.23)$$

where, as stated before,  $C_{L(u_k)}^{(1)}$  depends only on the LLR  $L(u_k)$  and not on whether  $u_k$  is  $\pm 1$ .

Again assuming the channel is memoryless we can write:

$$P(\underline{z}_k | \{s' \wedge s\}) \equiv P(\underline{z}_k | \underline{x}_k) = \prod_{l=1}^n P(z_{kl} | x_{kl}), \quad (2.24)$$

where  $\underline{x}_k$  is the transmitted codeword associated with the transition from state  $S_{k-1} = s'$  to state  $S_k = s$ ,  $x_{kl}$  and  $z_{kl}$  are the individual bits within the transmitted and received codewords. Assuming that the transmitted codeword  $\underline{x}_k$  have been transmitted over a Gaussian channel using BPSK we have:

$$\begin{aligned}
P(\underline{z}_k | \{s' \wedge s\}) &= \prod_{l=1}^n \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2} (z_{kl} - ax_{kl})^2\right) \cdot dz_{kl} \\
&= \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^n (z_{kl} - ax_{kl})^2\right) \prod_{l=1}^n dz_{kl} \\
&= \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^n (z_{kl}^2 + a^2 x_{kl}^2 - 2ax_{kl}z_{kl})\right) \prod_{l=1}^n dz_{kl} \\
&= C_{\underline{z}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \cdot \exp\left(\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^n z_{kl} x_{kl}\right) \cdot K,
\end{aligned} \tag{2.25}$$

where,  $C_{\underline{z}_k}^{(2)}$  depends only on the channel SNR and on the magnitude of the received sequence  $\underline{z}_k$ , while,  $C_{\underline{x}_k}^{(3)}$  depends only on the channel SNR and on the fading amplitude.

Hence we can write for  $\gamma_k(s', s)$ :

$$\begin{aligned}
\gamma_k(s', s) &= P(u_k) \cdot P(\underline{z}_k | \{s' \wedge s\}) \\
&= C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} \sum_{l=1}^n z_{kl} x_{kl}\right) \cdot K,
\end{aligned} \tag{2.26}$$

where  $C = C_{L(u_k)}^{(1)} \cdot C_{\underline{z}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)}$  does not depend on the sign of the bit  $u_k$  or the transmitted codeword  $\underline{x}_k$  and so is constant over the summation in the numerator and denominator in Eq: 2.17 and cancels out, similar argument is valid for constant  $K$ .

Form Eq: 2.17 and Eq: 2.19 we can write for conditional LLR of  $u_k$ , given the received sequence  $\underline{z}_k$  as:

$$L(u_k | \underline{z}) = \ln \left( \frac{\sum_{\substack{(s', s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s)}{\sum_{\substack{(s', s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s)} \right) \tag{2.27}$$

It is this LLR  $L(u_k | \underline{z})$  that the MAP decoder delivers. Summary of operations in the MAP algorithm is shown in Fig: 2.6.

### 2.3 Iterative Turbo Decoding Principles

In this section, we explain the concepts of extrinsic and intrinsic information, and highlight how the MAP algorithm described above, and other soft-in soft-out decoders, can be used in the iterative decoding of turbo codes.

Consider first the expression for  $\gamma_k(s', s)$  in Eq: 2.26, which is restated here for convenience:

$$\gamma_k(s', s) = C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} \sum_{l=1}^n z_{kl} x_{kl}\right) \quad (2.28)$$

Considering systematic code, where one of the  $n$  transmitted bits will be the systematic bit  $u_k$ , and assuming this systematic bit is the first of the  $n$  transmitted bits, then we will have  $x_{k1} = u_k$ , and we can rewrite Eq: 2.28 as:

$$\begin{aligned} \gamma_k(s', s) &= C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} z_{ks} u_k\right) \cdot \exp\left(\frac{L_c}{2} \sum_{l=2}^n z_{kl} x_{kl}\right) \\ &= C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\frac{L_c}{2} z_{ks} u_k\right) \cdot \chi_k(s', s), \end{aligned} \quad (2.29)$$

where,  $z_{ks}$  is the received version of the transmitted systematic bit  $x_{k1} = u_k$  and:

$$\chi_k(s', s) = \exp\left(\frac{L_c}{2} \sum_{l=2}^n z_{kl} x_{kl}\right). \quad (2.30)$$

Using Eq: 2.29 and remembering that in the numerator we have  $u_k = +1$  for all terms in the summation, whereas in the denominator we have  $u_k = -1$ , we can rewrite Eq: 2.27 as:

$$\begin{aligned} L(u_k | \underline{z}) &= \ln \left( \frac{\sum_{\substack{(s', s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s)}{\sum_{\substack{(s', s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s)} \right) \\ &= \ln \left( \frac{\sum_{\substack{(s', s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot e^{+L(u_k)/2} \cdot e^{+L_c z_{ks}/2} \cdot \chi_k(s', s) \cdot \beta_k(s)}{\sum_{\substack{(s', s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot e^{+L(u_k)/2} \cdot e^{+L_c z_{ks}/2} \cdot \chi_k(s', s) \cdot \beta_k(s)} \right) \\ &= L(u_k) + L_c z_{ks} + \ln \left( \frac{\sum_{\substack{(s', s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot \chi_k(s', s) \cdot \beta_k(s)}{\sum_{\substack{(s', s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot \chi_k(s', s) \cdot \beta_k(s)} \right) \\ &= L(u_k) + L_c z_{ks} + L_e^E(u_k) \end{aligned} \quad (2.31)$$

where:

$$L_e^E(u_k) = \ln \left( \frac{\sum_{\substack{(s', s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot \chi_k(s', s) \cdot \beta_k(s)}{\sum_{\substack{(s', s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot \chi_k(s', s) \cdot \beta_k(s)} \right). \quad (2.32)$$



Thus we can see that the a-posteriori LLR  $L(u_k | \underline{z})$  calculated with the aid of the MAP algorithm can be viewed as comprising three additive soft-metric terms:-  $L(u_k)$ ,  $L_c z_{ks}$  and  $L_e^E(u_k)$ .

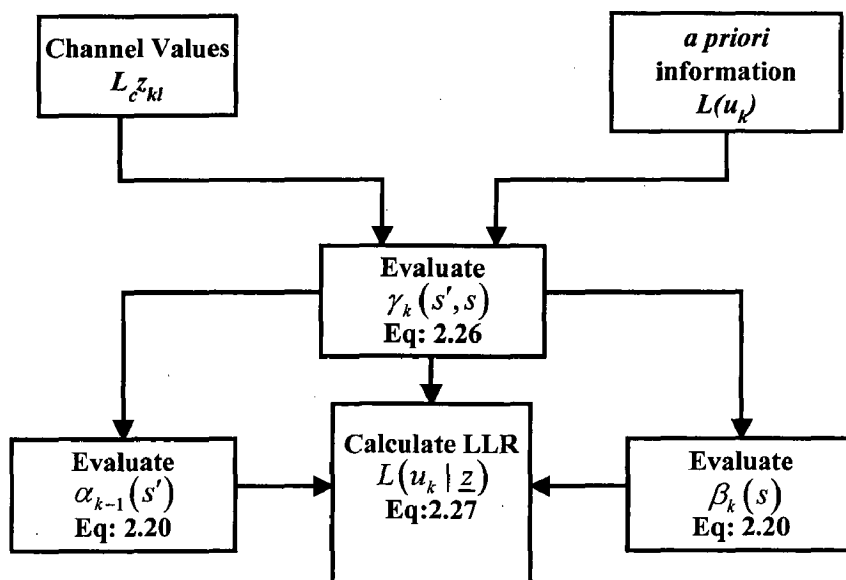
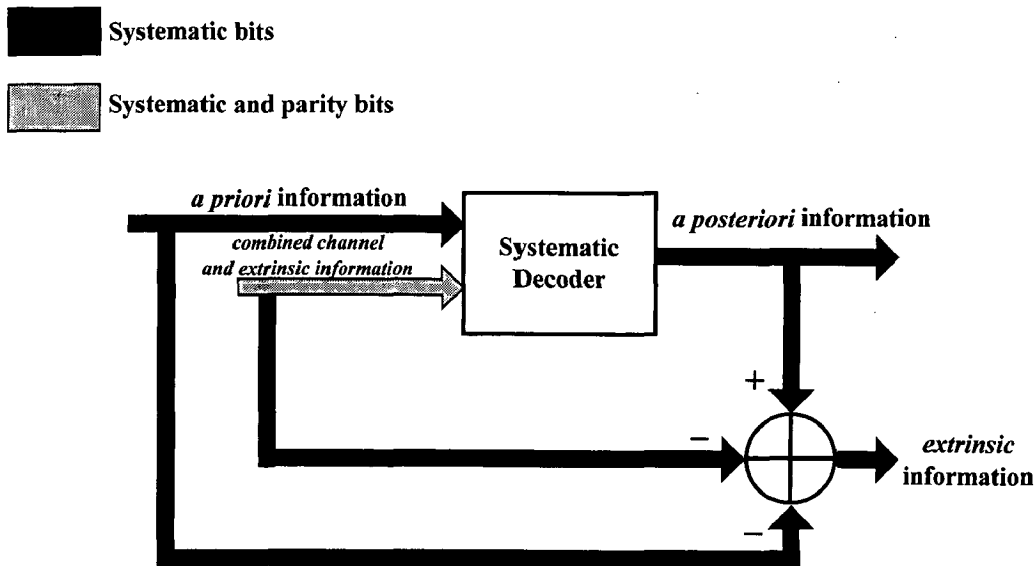


Fig: 2.6 Summary of the Key Operations in the MAP Algorithm

The *a-priori* LLR term  $L(u_k)$  comes from  $P(u_k)$  in the expression for the branch transition probability  $\gamma_k(s', s)$ . This probability should be generated by an independent source and is referred to as the a-priori probability of the  $k$ th information or systematic bit represented as +1 or -1, as illustrated in Fig: 2.7. Initially this value will be zero. The second term corresponds to the systematic bits conveyed by the channel and to the extrinsic LLR values. The final term is derived using the constraints imposed by the code used, from the a-priori information sequence  $L(u_n)$  and the received channel information sequence  $\underline{z}$ , *excluding* the received systematic bit  $z_{ks}$  and the a-priori information  $L(u_k)$  for the bit  $u_k$ . Hence it is referred to as the *extrinsic* LLR for the bit  $u_k$ .

We summarize below what is meant by the terms a-priori, extrinsic and a-posteriori information, which we use throughout this treatise.

**a priori:** The a-priori information related to a bit is information known before decoding commences, from a source other than the received sequence or the code constraints. It is also sometimes referred to as intrinsic information for contrasting it with the extrinsic information to be described next.



**Fig: 2.7** Schematic of a Component Decoder Employed in a Turbo Decoder, Showing the Input Information Received and Output Information Corresponding to the Systematic and Parity Bits

**extrinsic:** The extrinsic information related to a bit  $u_k$  is the information provided by a decoder based on the received sequence and on the a-priori information, but *excluding* the received systematic bit  $z_{ks}$  and the a-priori information  $L(u_k)$  related to the bit  $u_k$ . Typically the component decoder provides this information using the constraints imposed on the transmitted sequence by the code used. It processes the received bits and the a-priori information surrounding the systematic bit  $u_k$ , and uses this information and the code constraints for providing information about the value of the bit  $u_k$ .

**a posteriori:** The a-posteriori information related to a bit is the information that the decoder generates by taking into account *all* available sources of information concerning  $u_k$ . It is the a-posteriori LLR, i.e.  $L(u_k | \underline{z})$ , that the MAP algorithm generates as its output.

When the series of iterations halts, after either a fixed number of iterations or when a termination criterion is satisfied, the output from the turbo decoder is given by the de-interleaved a-posteriori LLRs of the second component decoder. The sign of these a posteriori LLRs gives the hard-decision output, i.e. whether the decoder believes that the transmitted data bit  $u_k$  was +1 or -1, and in some applications the magnitude of these LLRs, which gives the confidence the decoder has in its decision, may also be useful.

## 2.4 Modifications of the MAP algorithm

The MAP algorithm described above is much complex, as we see that the forward and backward metrics are the sums of exponential terms, one corresponding to each valid state transition in the trellis. This can be simplified by making use of the identity:

$$\max^*(x, y) \equiv \ln(e^x + e^y) = \max(x, y) + \ln(1 + \exp(-|x - y|)) \quad (2.33)$$

which will replace the computationally more difficult operation  $\ln(e^x + e^y)$  with a max function plus a lookup table for evaluating  $\ln(1 + e^{-|x-y|})$ . We first define the following log-domain metrics:

$$\left. \begin{aligned} \Gamma_k(s', s) &\triangleq \ln(\gamma_k(s', s)) \\ &= \ln\left(C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left[\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^n z_{kl} x_{kl}\right]\right) \\ &= \ln\left(C \cdot e^{(u_k L(u_k)/2)} \cdot \exp\left[\frac{L_c}{2} \sum_{l=1}^n z_{kl} x_{kl}\right]\right) \\ &= \hat{C} + \frac{1}{2} u_k L(u_k) + \frac{L_c}{2} \sum_{l=1}^n z_{kl} x_{kl}, \end{aligned} \right\} \quad (2.34)$$

where  $\hat{C} = \ln C$  does not depend on  $u_k$  or on the transmitted codeword  $\underline{x}_k$  and so can be considered a constant and omitted.

$$\left. \begin{aligned} A_k(s) &\triangleq \ln(\alpha_k(s)) \\ &= \ln\left(\sum_{all s'} \alpha_{k-1}(s') \cdot \gamma_k(s', s)\right) \\ &= \ln\left(\sum_{all s'} \exp[A_{k-1}(s') + \Gamma_k(s', s)]\right) \\ &= \max_{s'}^*(A_{k-1}(s') + \Gamma_k(s', s)). \end{aligned} \right\} \quad (2.35)$$

Similarly,

$$\left. \begin{aligned} B_{k-1}(s') &\triangleq \ln(\beta_{k-1}(s')) \\ &= \ln\left(\sum_{all s} \beta_k(s) \cdot \gamma_k(s', s)\right) \\ &= \ln\left(\sum_{all s} \exp[B_k(s) + \Gamma_k(s', s)]\right) \\ &= \max_s^*(B_k(s) + \Gamma_k(s', s)) \end{aligned} \right\} \quad (2.36)$$

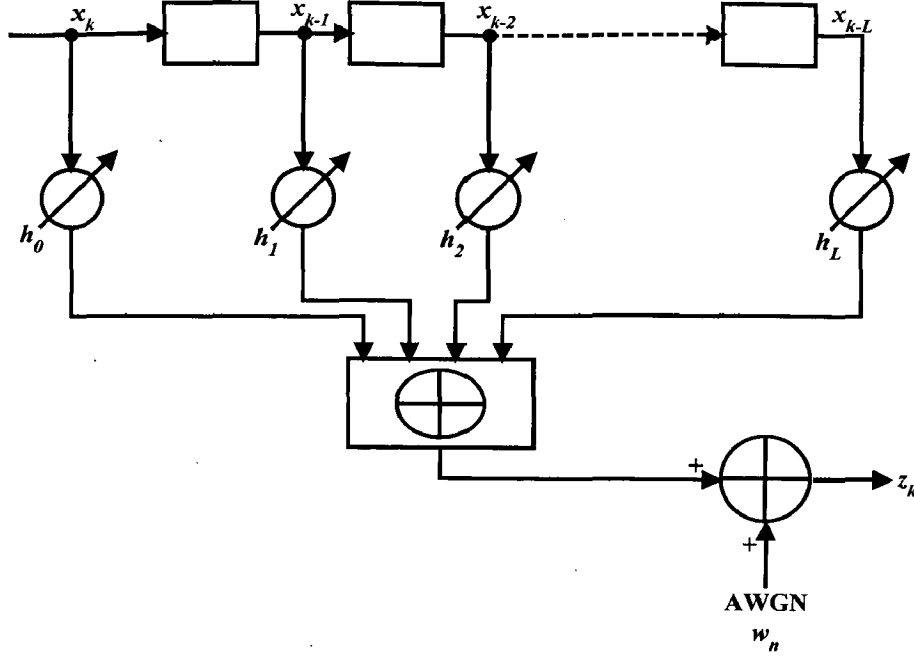


Fig: 2.8 Discrete Time Channel Model

It may be emphasized here that the output of MAP-equalizer is the LLRs corresponding to each input channel symbol whereas in MAP-decoder there are two output LLRs, the first LLR is corresponding to information bits and the second LLR is corresponding to encoded symbol. Hence, using Eq: 2.27 the LLR for the MAP-equalizer is:

$$L(\hat{x}_k | \underline{z}) = \ln \left( \frac{\sum_{(s',s) \in B_k^j = +1} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)}{\sum_{(s',s) \in B_k^j = -1} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)} \right) \quad (2.43)$$

where,  $B_k^j = +1$  is the set of transitions,  $S_{k-1} = s' \rightarrow S_k = s$  such that the  $j^{\text{th}}$  output digit of the encoded symbol  $x_k$  is equal to +1. Similarly,  $B_k^j = -1$  is the set of transitions,  $S_{k-1} = s' \rightarrow S_k = s$  such that the  $j^{\text{th}}$  output digit of the encoded symbol  $x_k$  is equal to -1.  $\alpha_k$  and  $\beta_k$  denotes the forward and backward metric of the MAP-equalizer, which can be calculated by the same formula as given in Eq: 2.20 and Eq: 2.21 respectively.

Similar to the derivation of Eq: 2.31, we can prove that the *a posteriori* LLR of the MAP-equalizer can also be written as:

$$L(\hat{x}_k | \underline{z}) = L(x_k) + L_c(x_k) + L_e^E(x_k) \quad (2.44)$$

Where,  $L(x_k)$  is the *a priori* LLR,  $L_c(x_k)$  is the channel LLR, and  $L_e^E(x_k)$  is the extrinsic LLR of the MAP-equalizer. But unlike MAP-decoder, in MAP-equalizer, we cannot separate channel information and extrinsic information in the output of the equalizer as the ISI channel is equivalent to a non-systematic non-binary convolutional coder. At the starting iteration, the value of *a priori* LLR  $L(x_k)$  is equal to zero and in later iterations this quantity is obtained from MAP-decoder.

## 2.6 Iterative MAP-Equalization and MAP-Decoding

Fig: 2.9 show the Iterative MAP equalization and MAP decoding. The equalizer computes the *a posteriori* probabilities,  $P(\hat{x}_n = x | \underline{z}) \approx P(\hat{x}_n = x | z_1, \dots, z_{K_c})$ ,  $x \in \mathcal{B}$ , given  $K_c$  received symbols  $z_n$ ,  $n=1, 2, \dots, K_c$ , and delivers *a posteriori* LLR  $L(\hat{x}_k | \underline{z})$  about channel input symbols/bits and feeds the MAP-decoder the difference of *a posteriori* LLR  $L(\hat{x}_k | \underline{z})$  and *a priori* LLR  $L(\hat{x}_k)$ :

$$\begin{aligned} L_E(\hat{x}_n) &\triangleq \ln \frac{P(\hat{x}_n = +1 | \underline{z})}{P(\hat{x}_n = -1 | \underline{z})} - \ln \frac{P(\hat{x}_n = +1)}{P(\hat{x}_n = -1)} \\ &= L(\hat{x}_n | \underline{z}) - L(\hat{x}_n) \end{aligned} \quad (2.45)$$

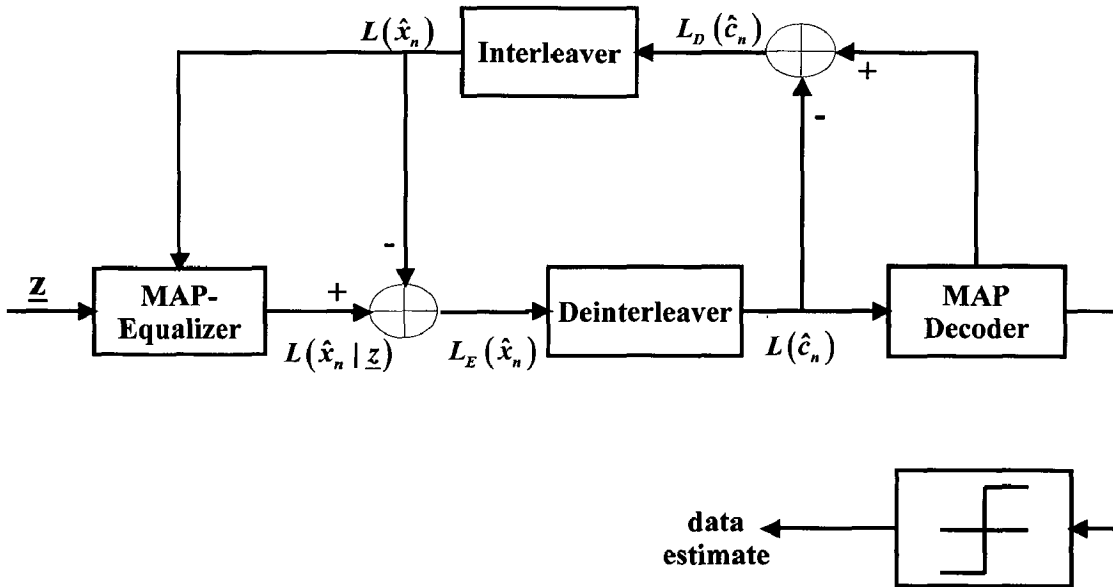


Fig: 2.9 Iterative MAP Equalization and MAP Decoding

The *a priori* LLR, which is  $L(\hat{x}_n)$ , represents priori information on the occurrence probability of  $x_n$  and is provided by the decoder. For initial equalization step, no *a priori* information is available and hence we have  $L(\hat{x}_n) = 0, \forall n$ . We emphasize that  $L_E(\hat{x}_n)$  is independent of  $L(\hat{x}_n)$ . This and the concept of treating feedback as a priori information are the two essential features of any system applying the turbo principle and turbo equalization in particular. The MAP decoder computes the APPs  $P(\hat{c}_n = x | L(\hat{c}_1), \dots, L(\hat{c}_{K_c}))$ ,  $x \in \mathcal{B}$ , given  $K_c$  code bit LLRs  $L(\hat{c}_n)$ ,  $n=1, 2, \dots, K_c$ , and delivers *a posteriori* LLR about coded symbols/bits, and feeds the MAP-equalizer the difference of *a posteriori* LLR and *a priori* LLR:

$$L_D(\hat{c}_n) \triangleq \ln \frac{P(\hat{c}_n = +1 | L(\hat{c}_1), \dots, L(\hat{c}_{K_c}))}{P(\hat{c}_n = -1 | L(\hat{c}_1), \dots, L(\hat{c}_{K_c}))} - \ln \frac{P(\hat{c}_n = +1)}{P(\hat{c}_n = -1)} \quad (2.46)$$

where,  $L_E(\hat{x}_n)$  is considered to be a priori LLR  $L(\hat{c}_n)$  for the decoder. The interleaver  $\Pi(\bullet)$  and the deinterleaver  $\Pi^{-1}(\bullet)$  provide the correct ordering of the LLRs  $L(\hat{c}_n) = \Pi^{-1}(L_E(\hat{x}_n))$  and  $L(\hat{x}_n) = \Pi(L_D(\hat{c}_n))$ , which are input to the equalizer and decoder, respectively. The MAP decoder also computes the data bit estimates

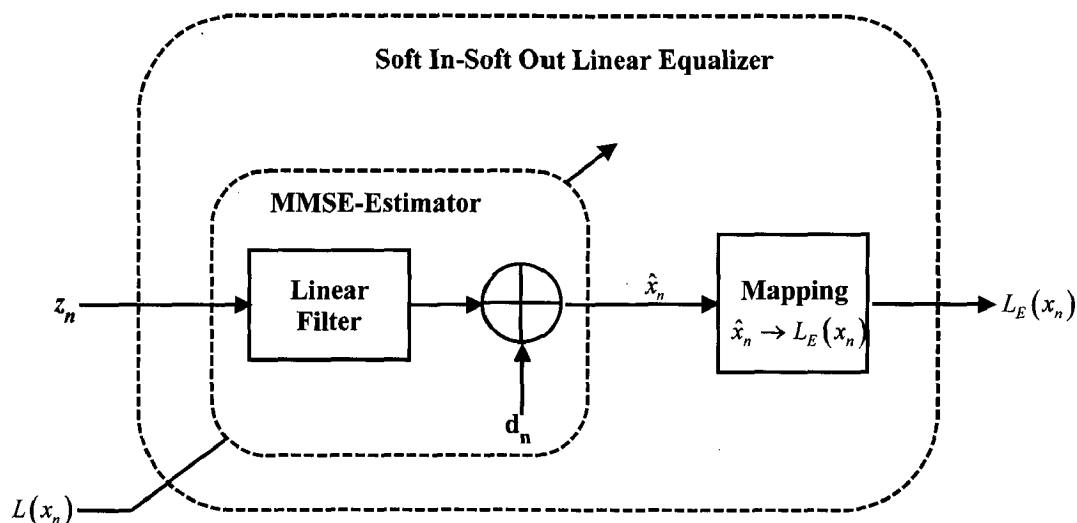
$$\hat{b}_i \triangleq \arg \max_{b \in \{0,1\}} P(b_i = b | L(c_1), \dots, L(c_{K_c})). \quad (2.47)$$

## Chapter 3

### TURBO EQUALIZATION USING MMSE EQUALIZER

#### 3.1 THE SISO EQUALIZER

The SISO Equalizer derivation is developed in two steps. First the estimation step produces an estimate  $\hat{x}_n$  of the channel input  $x_n$  by processing the receiver input  $z_n$  with help of priors  $L(x_n)$ . The next step is the formation of the soft output  $L_E(x_n)$ . Fig: 3.1 shows the two functional parts (estimator, mapper) of the SISO equalizer for MMSE-LE estimator implementation including the structure of the estimator. Given the channel co-



**Fig: 3.1** Components of the SISO equalizer

efficients  $h_k$  defined in Eq: 2.1 and the noise samples  $w_n$  the estimator input symbols  $z_n$  are obtained as

$$z_n = \sum_{k=0}^{M-1} h_k x_{n-k} + w_n \quad (3.1)$$

The symbols  $x_n$  are assumed to be independent, at least in a neighborhood around each symbol. This assumption will be essential for the derivation of the estimator and can be justified by de-correlating properties of the interleaver since the code symbols  $c_n$  are clearly independent, in particular neighboring symbols.

### 3.2 Estimator Using Linear Equalizer

As shown in Fig: 3.1 the estimator in the MMSE-LE implementation is a single linear filter, realized as a tapped delay line with time-varying parameters  $c_{n,k}$ . The filter has the impulse response

$$c[n] = \sum_{k=-N_1}^{+N_2} c_{n,k} \delta[n-k] \quad (3.2)$$

of length  $N$  where  $N=N_1+N_2+1$ . The output of the estimator is  $\hat{x}_n$ , the MMSE linear estimate of  $x_n$  given  $z_{\langle n-N_2, n+N_1 \rangle}$ , i.e.,

$$\hat{x}_n = \sum_{k=-N_1}^{+N_2} c_{n,k} z_{n-k} + d_n \quad (3.3)$$

where,  $d_n$  is a time varying offset compensating for a possible nonzero mean of random variable  $x_n$  given the prior  $L(x_n)$ . With the vectors

$$\begin{aligned} \mathbf{x}_n &\triangleq [x_{n-M-N_2+1} \ x_{n-M-N_2+2} \ \cdots \ x_{n+N_1}]^T, \\ \mathbf{w}_n &\triangleq [w_{n-N_2} \ w_{n-N_2+1} \ \cdots \ w_{n+N_1}]^T, \\ \mathbf{z}_n &\triangleq [z_{n-N_2} \ z_{n-N_2+1} \ \cdots \ z_{n+N_1}]^T, \\ \mathbf{c}_n &\triangleq [c_{n,N_2}^* \ c_{n,N_2-1}^* \ \cdots \ c_{n,-N_1}^*]^T, \end{aligned} \quad (3.4)$$

and the  $N \times (N+M-1)$  matrix

$$\mathbf{H} \triangleq \begin{bmatrix} h_{M-1} & h_{M-2} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_{M-1} & h_{M-2} & \cdots & h_0 & \cdots & 0 \\ & & & \ddots & & & \\ 0 & \cdots & 0 & h_{M-1} & h_{M-2} & \cdots & h_0 \end{bmatrix} \quad (3.5)$$

the vector  $\mathbf{z}_n$  can be expressed using Eq: 3.1 as,

$$\mathbf{z}_n = \mathbf{H}\mathbf{x}_n + \mathbf{w}_n. \quad (3.6)$$

The equation to obtain  $\hat{x}_n$  is now written in matrix form using Eq: 3.3,

$$\hat{x}_n = \mathbf{c}_n^H \mathbf{z}_n + d_n. \quad (3.7)$$

To perform MMSE estimation, the statistics  $\bar{x}_n \triangleq E(x_n)$  and  $\nu_n \triangleq \text{Cov}(x_n, x_n)$  of the symbols  $x_n$  are required. Usually, the  $x_n$  are assumed to be equiprobable and i.i.d., which corresponds to  $L(x_n) = 0, \forall n$ , and yields  $\bar{x}_n = 0$  and  $\nu_n = 1$ . For general  $L(x_n) \in \mathfrak{R}$  (the  $x_n$  are not equiprobable)  $\bar{x}_n$  and  $\nu_n$  are obtained as



$$\begin{aligned}
\bar{x}_n &= \sum_{x \in \{-1, +1\}} x \cdot P(x_n = x) = P(x_n = +1) - P(x_n = -1) \\
&= \frac{e^{L(x_n)}}{1 + e^{L(x_n)}} - \frac{1}{1 + e^{L(x_n)}} = \tanh(L(x_n) / 2) \\
v_n &= \sum_{x \in \{-1, +1\}} |x - E(x_n)|^2 \cdot P(x_n = x) = 1 - |\bar{x}_n|^2
\end{aligned}$$

The MMSE-LE parameter set  $(\mathbf{c}_n, d_n)$  is obtained using the MMSE cost criterion

$$(\mathbf{c}_n, d_n) = \arg \min_{\mathbf{c} \in \mathbb{C}^N, d \in \mathbb{C}} E\{|x_n - \hat{x}_n|^2\}. \quad (3.8)$$

This vector optimization problem with  $N+1$  parameters is easily solved using partial differentiation since there are no constraints imposed on the range of the parameter set:

$$\begin{aligned}
\frac{\partial E\{|x_n - \hat{x}_n|^2\}}{\partial \mathbf{c}_n} &= -2 \cdot E\{(x_n - \mathbf{c}_n^H \mathbf{z}_n - d_n) \mathbf{z}_n^H\} = \mathbf{0}_N, \\
\frac{\partial E\{|x_n - \hat{x}_n|^2\}}{\partial d_n} &= -2 \cdot E\{x_n - \mathbf{c}_n^H \mathbf{z}_n - d_n\} = 0.
\end{aligned} \quad (3.9)$$

The solution to Eq: 2.9 is unique and the global minimum for Eq: 3.8 since the Jacobian matrix of  $\hat{x}_n(\mathbf{c}_n, d_n)$  is positive definite unless all  $x_n$  are zero. Expanding using Eq: 3.7 yields

$$\begin{aligned}
E\{x_n \mathbf{w}_n^H\} + E\{x_n \mathbf{x}_n^H \mathbf{H}^H\} &= E\{\mathbf{c}_n^H \mathbf{w}_n \mathbf{w}_n^H\} + E\{\mathbf{c}_n^H \mathbf{H} \mathbf{x}_n \mathbf{w}_n^H\} \\
&\quad + E\{d_n \mathbf{w}_n^H\} + E\{\mathbf{c}_n^H \mathbf{w}_n \mathbf{x}_n^H \mathbf{H}^H\} \\
&\quad + E\{\mathbf{c}_n^H \mathbf{H} \mathbf{x}_n \mathbf{x}_n^H \mathbf{H}^H\} + E\{d_n \mathbf{x}_n^H \mathbf{H}^H\}, \\
\bar{x}_n &= E\{\mathbf{c}_n^H \mathbf{w}_n\} + E\{\mathbf{c}_n^H \mathbf{H} \mathbf{x}_n\} + E\{d_n\}.
\end{aligned} \quad (3.10)$$

From the noise specifications and the independence between  $w_n$  and  $x_n$ , it follows that  $E\{\mathbf{w}_n\} = \mathbf{0}_N$  and  $E\{\mathbf{w}_n \mathbf{w}_n^H\} = \sigma_w^2 \mathbf{I}_N$ . Finally, the following equation system has to be solved for the parameters  $(\mathbf{c}_n, d_n)$ :

$$\begin{aligned}
E\{x_n \mathbf{x}_n^H\} \mathbf{H}^H &= \mathbf{c}_n^H (\sigma_w^2 \mathbf{I}_N + \mathbf{H} E\{\mathbf{x}_n \mathbf{x}_n^H\} \mathbf{H}^H) + d_n \bar{x}_n^H \mathbf{H}^H, \\
\bar{x}_n &= \mathbf{c}_n^H \mathbf{H} \cdot \bar{\mathbf{x}}_n + d_n.
\end{aligned} \quad (3.11)$$

Introducing the quantities

$$\begin{aligned}
\mathbf{V}_n &= \text{Cov}(\mathbf{x}_n, \mathbf{x}_n) \triangleq E\{\mathbf{x}_n \mathbf{x}_n^H\} - E\{\mathbf{x}_n\} E\{\mathbf{x}_n^H\}, \\
\text{Cov}(x_n, \mathbf{z}_n) &\triangleq \mathbf{H} \cdot (E\{x_n \mathbf{x}_n^H\} - E\{x_n\} E\{\mathbf{x}_n^H\}), \\
\mathbf{u} &\triangleq \begin{bmatrix} \mathbf{0}_{1 \times (N_2 + M - 1)} & 1 & \mathbf{0}_{1 \times M_1} \end{bmatrix}^T \\
\mathbf{s} &\triangleq \mathbf{H} \mathbf{u}
\end{aligned}$$

yields the solution

$$\begin{aligned} \mathbf{c}_n &= (\sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H)^{-1} \cdot \nu_n \mathbf{s} = \mathbf{Cov}(\mathbf{z}_n, \mathbf{z}_n)^{-1} \mathbf{Cov}(\mathbf{z}_n, x_n), \\ d_n &= \bar{x}_n - \mathbf{c}_n^H \mathbf{H} \cdot \bar{\mathbf{x}}_n = \bar{x}_n - \mathbf{c}_n^H E\{\mathbf{z}_n\}. \end{aligned} \quad (3.12)$$

As such, the linear MMSE estimate  $\hat{x}_n$  of  $x_n$  can be written as,

$$\begin{aligned} \hat{x}_n &= \mathbf{c}_n^H \mathbf{z}_n + d_n \\ &= \bar{x}_n + \nu_n \mathbf{s}^H (\sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H)^{-1} \cdot (\mathbf{z}_n - E\{\mathbf{z}_n\}) \\ &= \bar{x}_n + \mathbf{Cov}(x_n, \mathbf{z}_n) \mathbf{Cov}(\mathbf{z}_n, \mathbf{z}_n)^{-1} \cdot (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}}_n) \end{aligned} \quad (3.13)$$

However,  $\hat{x}_n$  depends on  $L(x_n)$  via  $\bar{x}_n$  and  $\nu_n$ . In order that  $\hat{x}_n$  be independent of  $L(x_n)$ , we set  $L(x_n)$  to 0 while computing  $\hat{x}_n$ , yielding  $\bar{x}_n = 0$  and  $\nu_n = 1$ . This changes Eq: 3.13 to

$$\begin{aligned} \mathbf{Cov}(\mathbf{z}_n, \mathbf{z}_n) &= (\sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H + (1 - \nu_n) \mathbf{s} \cdot \mathbf{s}^H), \\ \mathbf{c}_n &= (\sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H + (1 - \nu_n) \mathbf{s} \cdot \mathbf{s}^H)^{-1} \mathbf{s}, \\ \hat{x}_n &= \mathbf{s}^H \mathbf{Cov}(\mathbf{z}_n, \mathbf{z}_n)^{-1} (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}}_n + (\bar{x}_n - 0) \mathbf{s}) \\ &= \mathbf{c}_n^H (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}}_n + \bar{x}_n \mathbf{s}) \end{aligned} \quad (3.14)$$

After MMSE equalization, for simplification the pdfs  $p(\hat{x}_n | x_n = x)$ ,  $x \in \{-1, +1\}$ , are approximated by Gaussian distribution with the parameters  $\mu_{n,x} \triangleq E(\hat{x}_n | x_n = x)$  and  $\sigma_{n,x}^2 \triangleq \mathbf{Cov}(\hat{x}_n, \hat{x}_n | x_n = x)$ .

$$p(\hat{x}_n | x_n = x) \approx \phi((\hat{x}_n - \mu_{n,x}) / \sigma_{n,x}) / \sigma_{n,x}. \quad (3.15)$$

This assumption tremendously simplifies the computations of the SISO equalizer output LLR  $L_E(x_n)$ . It is important that  $L_E(x_n)$  should not depend on the particular *a priori* LLR  $L(x_n)$  and therefore it is required that  $\hat{x}_n$  does not depend on  $L(x_n)$ , else it would affect the derivation of MMSE equalization algorithms. The statistics  $\mu_{n,x}$  and  $\sigma_{n,x}^2$  of  $\hat{x}_n$  are computed as,

$$\begin{aligned} \mu_{n,x} &= \mathbf{c}_n^H (E(\mathbf{z}_n | x_n = x) - \mathbf{H} \cdot \bar{\mathbf{x}}_n + \bar{x}_n \mathbf{s}) = x \cdot \mathbf{c}_n^H \mathbf{s} \\ \sigma_{n,x}^2 &= \mathbf{c}_n^H \mathbf{Cov}(\mathbf{z}_n, \mathbf{z}_n | x_n = x) \mathbf{c}_n \\ &= \mathbf{c}_n^H (\sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H - \nu_n \mathbf{s} \cdot \mathbf{s}^H) \mathbf{c}_n \\ &= \mathbf{c}_n^H \mathbf{s} (1 - \mathbf{s}^H \cdot \mathbf{c}_n). \end{aligned}$$

The output LLR  $L_E(x_n)$  follows as,

$$\begin{aligned}
L_E(x_n) &= \ln \frac{\phi\left(\frac{\hat{x}_n - \mu_{n,+1}}{\sigma_{n,+1}}\right) / \sigma_{n,+1}}{\phi\left(\frac{\hat{x}_n - \mu_{n,-1}}{\sigma_{n,-1}}\right) / \sigma_{n,-1}} = \frac{2\hat{x}_n \mu_{n,+1}}{\sigma_{n,+1}^2} \\
&= 2\mathbf{c}_n^H (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}} + \bar{\mathbf{x}}\mathbf{s}) / (1 - \mathbf{s}^H \mathbf{c}_n)
\end{aligned} \tag{3.16}$$

When  $L(x_n) = 0, \forall n$ , e.g., for the initial equalization step, we have  $\bar{x}_n = 0$  and  $\nu_n = 1, \forall n$ , yielding a time-invariant coefficient vector  $\mathbf{c}_n = \mathbf{c}_{NA}$  (NA stands for no *a priori* information), the usual MMSE LE solution,

$$\begin{aligned}
\mathbf{c}_{NA} &\triangleq \left( \sigma_w^2 \mathbf{I}_N + \mathbf{H} \mathbf{V}_n \mathbf{H}^H + (1 - \nu_n) \mathbf{s} \cdot \mathbf{s}^H \right)^{-1} \cdot \mathbf{s} \Big|_{L(x_n)=0} \\
&= \left( \sigma_w^2 \mathbf{I}_N + \mathbf{H} \cdot \mathbf{H}^H \right)^{-1} \cdot \mathbf{s}
\end{aligned}$$

*Approximate Implementation I:* For computing  $\mathbf{c}_n$  and  $L_E(x_n)$  for each time step  $n$ , we need to compute the inverse of  $N \times N$  matrix inverse, i.e.,  $\text{Cov}(\mathbf{z}_n, \mathbf{z}_n)^{-1}$ , which causes a high computational load. There are methods to reduce the computational load for e.g., a recursive algorithm to compute  $\mathbf{c}_n$  from  $\mathbf{c}_{n-1}$  [13]. In an alternate approach the use of time-invariant coefficient vector  $\mathbf{c}_{NA}$  to compute  $\hat{x}_n$  reduces the computational burden. Given a general  $L(x_n) \in \mathfrak{R}$ , we have:

$$\hat{x}_n = \mathbf{c}_{NA}^H (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}}_n + \bar{\mathbf{x}}_n \mathbf{s}).$$

The use of this approximation was justified by the excellent complexity / performance tradeoff.

*Approximate Implementation II:* In this method instead of assuming all variances are one, we assume that all output bits have the same variance which is equal to their average variance. This variance changes during iterations based on the performance of the decoder. With this assumption,

$$\nu = \frac{1}{N} \sum_{n=0}^{N-1} \nu_n = 1 - \frac{1}{N} \sum_{n=0}^{N-1} |\bar{x}_n|^2, V_n = \nu I_{N \times N} \tag{3.17}$$

If we use the previous formula, the filter coefficients in the time domain would be,

$$\mathbf{c}_n = \left( \sigma^2 I_{N \times N} + \nu \mathbf{H} \cdot \mathbf{H}^H + (1 - \nu) \mathbf{s} \cdot \mathbf{s}^H \right)^{-1} \cdot \mathbf{s} \tag{3.18}$$

Let  $A = \sigma^2 I_{N \times N} + \nu \mathbf{H} \cdot \mathbf{H}^H$ . When matrix inversion lemma is applied to Eq: 3.18, the result is:

$$\begin{aligned}
\mathbf{c}_n &= \left( A + (1-\nu)\mathbf{s} \cdot \mathbf{s}^H \right)^{-1} \cdot \mathbf{s} \\
&= \left( A^{-1} - A^{-1}\mathbf{s} \left( (1-\nu)^{-1} + \mathbf{s}^H A^{-1}\mathbf{s} \right)^{-1} \mathbf{s}^H A^{-1} \right) \cdot \mathbf{s} \\
&= \left( A^{-1} - K A^{-1}\mathbf{s} \cdot \mathbf{s}^H A^{-1} \right) \cdot \mathbf{s} \\
&= A^{-1} \cdot \mathbf{s} - \left( 1 - \frac{K}{1-\nu} \right) A^{-1} \cdot \mathbf{s} \\
&= \frac{K}{1-\nu} A^{-1} \cdot \mathbf{s}
\end{aligned} \tag{3.19}$$

where,  $K = \frac{1}{(1-\nu)^{-1} + \mathbf{s}^H A^{-1}\mathbf{s}}$ .

### 3.3 Turbo Equalization Using Frequency Domain Equalizer

In the approach derived in turbo equalization, a linear equalizer consists of a FIR filter coefficients  $c_{k,n}$ ,  $k = -N_1, -N_1+1, \dots, N_2$ , of length  $N = N_1 + N_2 + 1$  is used as MMSE estimator [14].

We have the estimates  $\hat{x}_n$  computed by filtering  $z_k - E\{z_k\}$ ,  $k = n, n+1, \dots, n+L-1$ , with  $\mathbf{c}$ ,

$$\begin{aligned}
\bar{\mathbf{x}}_n &\triangleq \left[ E\{x_n\} E\{x_{n+1}\} \dots E\{x_{n+L-1}\} \right]^T = \left[ \bar{x}_n \bar{x}_{n+1} \dots \bar{x}_{n+L-1} \right]^T, \\
\hat{x}_n &= \mathbf{c}^H (\mathbf{z}_n - \mathbf{H} \cdot \bar{\mathbf{x}}_n + \bar{x}_n \mathbf{H} \cdot \mathbf{u}).
\end{aligned}$$

With this interpretation the equalization step can be denoted as follows:

$$\begin{aligned}
\mathbf{p} &\triangleq \mathbf{H}^H \cdot \mathbf{c} \\
\hat{\mathbf{x}} &= (\text{Circ}_L[\mathbf{c}])^H \mathbf{z}_0 - (\text{Circ}_L[\mathbf{p}])^H \mathbf{x}_0 + \mathbf{p}^H \mathbf{u} \mathbf{x}_0
\end{aligned} \tag{3.20}$$

The advantage of this structure becomes obvious if we compute Eq: 3.17 in the frequency domain by applying DFT, which is done by multiplying a length  $L$  column vector containing elements in the time domain with the  $L \times L$  matrix:

$$\mathbf{F} \triangleq (W_{n,k}), n, k = 0, 1, \dots, L-1, W_{n,k} = \exp\left(-\frac{2\pi nki}{L}\right),$$

where,  $i \triangleq \sqrt{-1}$ . Rewriting Eq: 3.17 using the property  $L \cdot \mathbf{F}^{-1}\mathbf{F} = \mathbf{F}^H \mathbf{F} = L \cdot \mathbf{I}_L$  of the DFT to yield the frequency domain vector  $\hat{\mathbf{X}}$  of the estimate  $\hat{x}_n$ :

$$\begin{aligned}
\mathbf{C} &\triangleq \mathbf{F}\mathbf{c} = [C_0 \ \cdots \ C_{L-1}]^T, \\
\tilde{\mathbf{H}} &\triangleq \mathbf{F} \begin{bmatrix} h_0 & h_1 & \cdots & h_{M-1} & \mathbf{0}_{1 \times (L-M)} \end{bmatrix}^T = [\tilde{H}_0 \ \cdots \ \tilde{H}_{L-1}]^T, \\
\mathbf{P} &\triangleq \mathbf{F}\mathbf{p} = \mathbf{F}\mathbf{F}^{-1} \mathbf{Diag}[\tilde{\mathbf{H}}]^H \mathbf{F}\mathbf{c} = \mathbf{Diag}[\tilde{\mathbf{H}}]^H \mathbf{C} = \left[ (C_0 \cdot \tilde{H}_0^*) \ \cdots \ (C_{L-1} \cdot \tilde{H}_{L-1}^*) \right]^T, \\
\hat{\mathbf{X}} &= \mathbf{F}\hat{\mathbf{x}} = \mathbf{F}(\text{Circ}_L[\mathbf{c}])^H \mathbf{F}^{-1}\mathbf{F} \cdot \mathbf{z}_0 - \mathbf{F}(\text{Circ}_L[\mathbf{p}])^H \mathbf{F}^{-1}\mathbf{F} \cdot \bar{\mathbf{x}}_0 \\
&= \mathbf{Diag}[\mathbf{C}]^H \mathbf{F}\mathbf{z}_0 - \mathbf{Diag}[\mathbf{P}]^H \mathbf{F} \cdot \bar{\mathbf{x}}_0 + \frac{1}{L} \cdot \mathbf{F}\mathbf{P}^H \mathbf{F}\mathbf{u}\bar{\mathbf{x}}_0 \\
&= \mathbf{Diag}[\mathbf{C}]^H \mathbf{F}\mathbf{z}_0 - \mathbf{Diag}[\mathbf{P}]^H \mathbf{F} \cdot \bar{\mathbf{x}}_0 + \frac{1}{L} \cdot \sum_{k=0}^{L-1} (C_k^* \cdot H_k) \cdot \mathbf{F}\bar{\mathbf{x}}_0
\end{aligned} \tag{3.21}$$

The vector  $\mathbf{C}$  is written in terms of  $\mathbf{H}$  as,

$$\begin{aligned}
\mathbf{C} &= \mathbf{F}\mathbf{c} = \mathbf{F}(\sigma_w^2 \mathbf{I}_L + \mathbf{H}\mathbf{H}^H)^{-1} \mathbf{H}\mathbf{u} \\
&= \mathbf{F} \left( \sigma_w^2 \mathbf{I}_L + \mathbf{F}^{-1} \mathbf{Diag}[\tilde{\mathbf{H}}] \mathbf{Diag}[\tilde{\mathbf{H}}]^{-H} \mathbf{F} \right)^{-1} \mathbf{F}^{-1} \mathbf{Diag}[\tilde{\mathbf{H}}] \mathbf{F}\mathbf{u} \\
&= \left( \sigma_w^2 \mathbf{I}_L + \mathbf{Diag}[\tilde{\mathbf{H}}] \mathbf{Diag}[\tilde{\mathbf{H}}]^{-H} \right)^{-1} \mathbf{Diag}[\tilde{\mathbf{H}}] \cdot \mathbf{1}_{L \times 1}, \\
C_k &= \frac{H_k}{\sigma_w^2 + H_k H_k^*}, \quad k = 0, 1, \dots, L-1.
\end{aligned}$$

Given the estimates  $\hat{\mathbf{x}} = \mathbf{F}^{-1}\hat{\mathbf{X}}$ , the equalizer output  $L_e^E(x_n)$  is computed as [14]

$$\begin{aligned}
\mu &= \mathbf{c}^H \mathbf{H}\mathbf{u} = \mathbf{p}^H \mathbf{u} = \frac{1}{L} \cdot \sum_{k=0}^{L-1} (C_k^* \cdot H_k), \\
\hat{\sigma}^2 &= \frac{1}{L} \sum_{n=0}^{L-1} |\text{sign}(\hat{x}_n) \cdot \mu - \hat{x}_n|^2, \\
L_e^E(x_n) &= \frac{2\hat{x}_n \mu}{\hat{\sigma}^2}, \quad n = 0, 1, \dots, L-1.
\end{aligned} \tag{3.22}$$

For the first equalization step, no a-priori information is available from decoder. The equalizer assumes in this case that  $L_e^D(x_n) = 0, \forall n$ , and uses

$$L_e^E(x_n) = \frac{2\hat{x}_n}{1 - \mathbf{u}^H \mathbf{H}^H \mathbf{c}} = \frac{2\hat{x}_n}{1 - \frac{1}{L} \cdot \sum_{k=0}^{L-1} (C_k \cdot H_k^*)}, \quad n = 0, 1, \dots, L-1, \tag{3.23}$$

to compute  $L_e^E(x_n)$  [15]. We summarize the algorithm as below

**Input:**

- Received symbols  $[z_0 \ \cdots \ z_{L-1}]^T$ , a-priori information  $[L_e^D(x_0) \ \cdots \ L_e^D(x_{L-1})]^T$ ,
- channel and receiver characteristics  $h[n]$  and  $\sigma_w^2$ ,

**Initialization:**

$$\begin{aligned} [Z_0 \ \cdots \ Z_{L-1}]^T &\leftarrow \text{DFT}[z_0 \ \cdots \ z_{L-1}]^T, \\ [\tilde{H}_0 \ \cdots \ \tilde{H}_{L-1}]^T &\leftarrow \text{DFT}[h_0 \ h_1 \ \cdots \ h_{M-1} \ \mathbf{0}_{1 \times (L-M)}]^T, \\ \mu &\leftarrow \frac{1}{L} \sum_{k=0}^{L-1} \frac{H_k H_k^*}{\sigma_w^2 + H_k H_k^*}, \end{aligned}$$

**Equalization:**

$$\begin{aligned} \bar{x}_n &\leftarrow \tanh\left(\frac{1}{2} L_e^D(x_n)\right), n=0,1,\dots,L-1, \\ [\bar{X}_0 \ \cdots \ \bar{X}_{L-1}]^T &\leftarrow \text{DFT}[\bar{x}_0 \ \cdots \ \bar{x}_{L-1}]^T, \\ \hat{X}_k &\leftarrow \frac{\tilde{H}_k^*}{\sigma_w^2 + \tilde{H}_k \tilde{H}_k^*} Z_k + \left(\mu - \frac{\tilde{H}_k^* \tilde{H}_k}{\sigma_w^2 + \tilde{H}_k \tilde{H}_k^*}\right) \bar{X}_k, k=0,1,\dots,L-1, \\ [\hat{x}_0 \ \cdots \ \hat{x}_{L-1}] &\leftarrow \text{DFT}^{-1}[\hat{X}_0 \ \cdots \ \hat{X}_{L-1}]^T, \\ \hat{\sigma}^2 &\leftarrow \frac{1}{L} \sum_{n=0}^{L-1} |\text{sign}(\hat{x}_n) \cdot \mu - \hat{x}_n|^2, \\ L_e^E(x_n) &\leftarrow \frac{2\hat{x}_n \mu}{\hat{\sigma}^2}, n=0,1,\dots,L-1. \end{aligned} \tag{3.24}$$

Now converting the approximate implementation II formula to the frequency domain, the filter coefficient would then be:

$$C_k = \frac{K}{1-\nu} \times \frac{\tilde{H}_k^*}{\sigma^2 + \nu \tilde{H}_k \tilde{H}_k^*} \tag{3.25}$$

The estimated symbol in the frequency domain at the output of the FDE would be,

$$\hat{X}_k = \frac{K}{1-\nu} \times \frac{\tilde{H}_k^*}{\sigma^2 + \nu |\tilde{H}_k|^2} Z_k - \frac{K}{1-\nu} \times \frac{|\tilde{H}_k|^2}{\sigma^2 + \nu |\tilde{H}_k|^2} \bar{X}_k + \mu \bar{X}_k \tag{3.26}$$

where,

$$\mu = \frac{1}{N} \sum_{k=0}^{N-1} H_k C_k = \frac{1}{N} \times \frac{K}{1-\nu} \times \sum_{k=0}^{N-1} \frac{|\tilde{H}_k|^2}{\sigma^2 + \nu |\tilde{H}_k|^2} \tag{3.27}$$

### 3.4 Improved Max-Log-MAP Algorithm: The Log-MAP Extension

The Log-MAP algorithm introduced in [16] evaluates  $\alpha_k(s')$ ,  $\beta_k(s)$ , and  $\gamma_k(s',s)$  in logarithmic terms. Using this approach, the forward and backward recursion is calculated as in Eq: 2.35 and Eq: 2.36 respectively, and using the identity in Eq: 2.33. On one hand, evaluation of the Log-MAP algorithm by exploiting the Jacobian logarithmic function,

Eq: 2.33, results in high complexity. On the other hand, saving the results of  $\log(1 + \exp(-|x - y|))$  in a lookup table would introduce a quantization error caused by truncation of the input of the lookup table (this truncation is necessary because otherwise, the size of the look up table becomes extremely large, and the implementation is infeasible). Another problem with the Log-MAP algorithm is that multiple lookup tables are required for a wide range of operating signal-to-noise ratios (SNRs), thus increasing hardware cost. Moreover, in order to evaluate each MAP L-value at the output of the equalizer or the decoder, correction terms should be added to the intermediate results from the lookup tables recursively. Reading data from these tables is time consuming process, and thus, it is not desirable; however, high-speed adders and comparators can be implemented effectively, thereby reducing power and area; therefore, increasing the overall speed that is desperately needed by communication systems. It should be noted that the logarithmic term of Eq: 2.33 is effective when  $|x - y|$  is around zero; otherwise, the effect of this term is negligible. Therefore, when MacLaurin Series expansion is employed to describe the logarithmic term around zero and neglecting orders greater than one [17], it comes up with

$$\log(1 + \exp(-x)) \approx \log 2 - \frac{1}{2}x \quad (3.28)$$

and since the logarithmic term of Eq: 2.33 is always greater than zero, this equation using Eq: 3.28 can be rewritten as follows,

$$\max^*(x, y) \approx \max(x, y) + \max\left(0, \log 2 - \frac{1}{2}|x - y|\right) \quad (3.29)$$

Eq: 3.29 can be easily implemented using comparators, adders and a shift register.

The block diagram in Fig: 3.2 show the node metric calculation units. In this figure,  $s_j$  refers to the state  $j$  at time  $k$ , while  $s'_j$  and  $s''_j$  refer to those previous states at time  $k-1$  which enter state  $s_j$  at time  $k$ . the delay element shown by "D" in this figure are employed in order to provide the node metric values at time  $k-1$ . The values of  $\alpha_k(s)$  in this figure are computed for a turbo decoder (or a turbo equalizer) with  $n$  different states using Eq: 2.35. It should be noted that  $\beta_k(s)$  is determined using the same structure in backward recursion. Fig: 3.3 show the detailed architecture of each block in Fig: 3.2. It is evident from this figure that the implementation of the additional part in Eq: 3.29 in this method is simple and needs a smaller ship area size compared to the additional part in

the Log-MAP algorithm of Eq: 2.33 that needs multiple lookup tables for a wide range of SNRs, which increases the chip area and, consequently, the implementation cost.

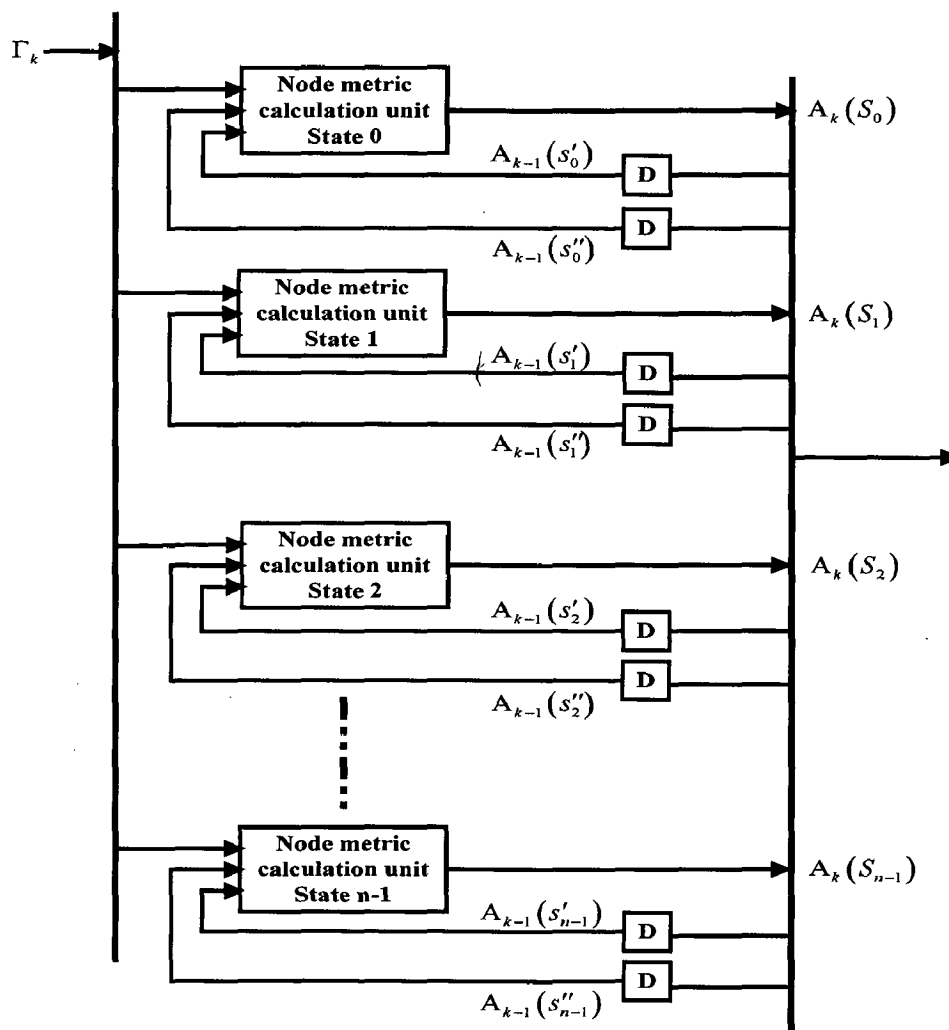


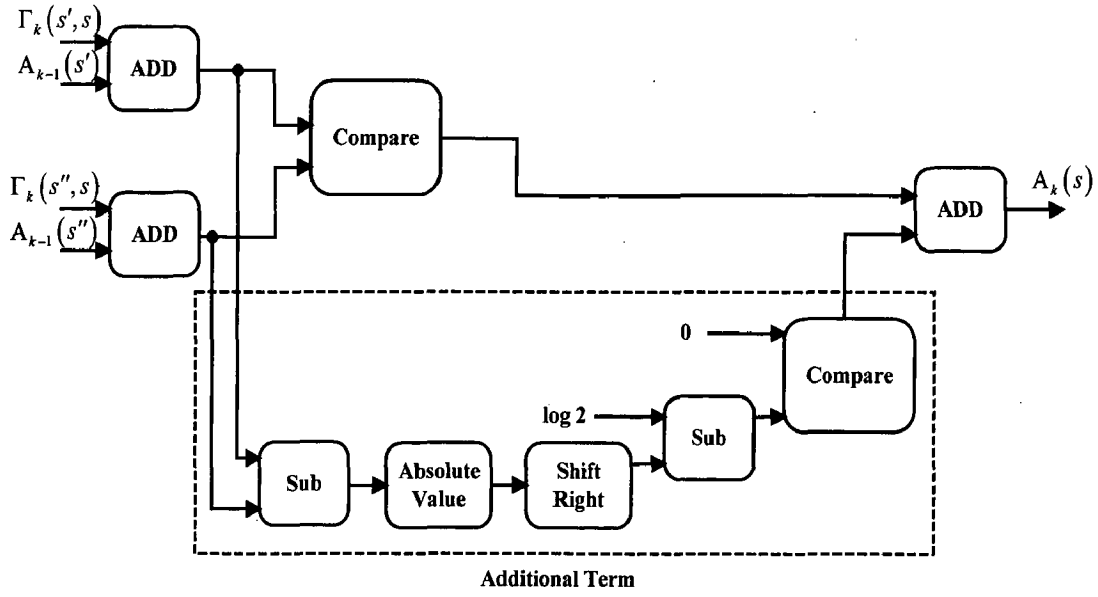
Fig: 3.2 Node Metric Calculation Unit for n Different States

### 3.5 Interleaver Design

The interleaver is an integral part of the overall turbo encoder, to achieve performance close to Shannon limit, the information block length (interleaver size)  $K$  is chosen to be very large, usually at least several thousand bits. The best interleaver reorders the bits in a pseudorandom manner. Conventional block (row-column) interleavers do not perform well in turbo codes, except at relatively short block lengths.

Pseudorandom interleaving patterns can be generated in many ways, for example, by using a primitive polynomial to generate a maximum-length shift-register sequence whose



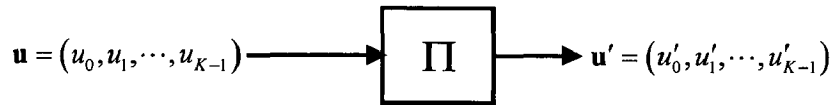


**Fig: 3.3** Detailed Architecture of New Method

cycle structure determines the permutation. Another method uses a computationally simple algorithm based on quadratic congruence [3],

$$c_m \equiv \frac{km(m+1)}{2} \pmod{K}, 0 \leq m \leq K, \quad (3.30)$$

to generate an index mapping function  $c_m \rightarrow c_{m+1} \pmod{K}$ , where  $K$  is the interleaver size,



**Fig: 3.4** Interleaver

and  $k$  is an odd integer. For example, for  $K=16$  and  $k=1$ , we obtain,

$$(c_0, c_1, c_2, \dots, c_{15}) = (0, 1, 3, 6, 10, 15, 5, 12, 4, 13, 7, 2, 14, 11, 9, 8) \quad (3.31)$$

which implies that index 0 (input bit  $u'_0$ ) in the interleaved sequence  $\mathbf{u}'$  is mapped into index 1 in the original sequence  $\mathbf{u}$  (i.e.,  $u'_0 = u_1$ ), index 1 in  $\mathbf{u}'$  is mapped into index 3 in  $\mathbf{u}$  ( $u'_1 = u_3$ ), and so on, resulting in the permutation

$$\prod_{16} = [1, 3, 14, 6, 13, 12, 10, 2, 0, 8, 15, 9, 4, 7, 11, 5] \quad (3.32)$$

If this interleaving pattern is shifted cyclically to the right  $r = 8$  times, we obtain the interleaving pattern of

$$\prod_{16} = [0, 8, 15, 9, 4, 7, 11, 5, 1, 3, 14, 6, 13, 12, 10, 2] \quad (3.33)$$

For  $K$  a power of 2, it has proved that these quadratic interleavers have statistical properties similar to those of randomly chosen interleavers, and thus they give good performance when used in turbo coding [18]. Other good interleaving patterns can be generated by varying  $k$  and  $r$ , and the special case  $r = K/2$  (used to obtain Eq: 3.33) results in an interleaver that simply interchanges pairs of indices. This special case is particularly interesting in terms of implementation, since the interleaving and deinterleaving functions (both used in decoding) are identical.

## Chapter 4

### SIMULATION DETAILS

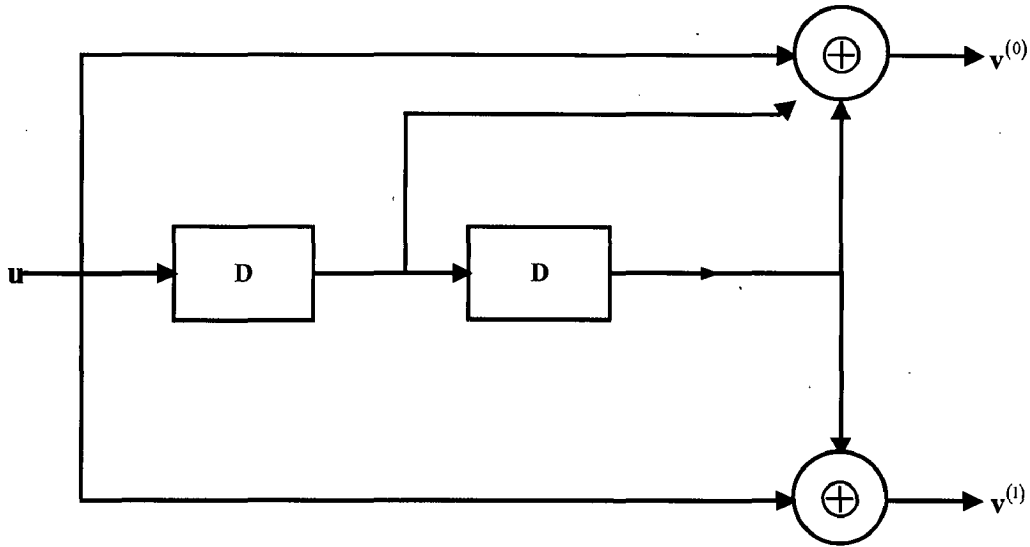
---

In this chapter, the steps involved in the simulation of Turbo equalization and decoding are described in detail. The simulation is done in MATLAB environment.

#### 4.1 Channel Encoder

The channel encoder used in the simulation is rate  $\frac{1}{2}$  constraint length  $\gamma = 2$  non-systematic convolutional encoder (Fig: 4.1). Steps involved in channel encoder are given below. The D represents delay element

Flow chart for simulating the encoder is shown in Fig: 4.2, the data frame is first zero padded by adding zeros equal to constraint length of encoder. Next, the contents of shift register are assumed to be zero initially. Later the contents are updated as can be seen from Fig: 4.1.



**Fig: 4.1** A (2,1,2) Nonsystematic Feedforward Encoder

As the code rate we have considered is rate  $\frac{1}{2}$  the number of code bits obtained are equal to,

$$n = \frac{(k + \gamma)}{\text{coderate}} \quad (4.1)$$

where,  $k$  is the frame length of message bit frame. For every message bit 2 code bits are generated as is evident Fig: 4.1, the first component of code bit is obtained by modulo 2 addition of three inputs; present data bit and previous two data bits, whereas the second

component of code bit is obtained by modulo 2 addition of two inputs; present data bit and 2<sup>nd</sup> previous data bit. Once the code bits for present time instant the contents of shift register are updated, and the process of obtaining code bits for every time instant is continued till the data frame end.

After obtaining the entire code bits for a given data frame, the new formed code bits frame is passed through pseudorandom interleaver. The interleaved bits are then transmitted through ISI channel. We have generated independent and identically distributed AWGN noise samples and added these to the received signal samples. This set of data is first passed through the Equalizer block, for initial equalization step no *a priori* information is so we assume the transmitted data bits are equiprobable, with this assumption we have  $\bar{x}_n = 0$  and  $\nu_n = 1$ , for all n Fig: 4.3 shows the flow chart corresponding to equalization step. The output of equalizer is the soft extrinsic values which are passed to decoder.

## 4.2 Simulation of Decoder

For decoder, we first calculate the value of log domain transition metric,  $\Gamma_k(s', s) \triangleq \ln(\gamma_k(s', s))$  as given in Eq: 2.34. Fig: 4.4 show the decoder trellis for convolutional encoder used in Fig: 4.1. We consider the following,

$$S_0 \rightarrow [0 \ 0]$$

$$S_1 \rightarrow [1 \ 0]$$

$$S_2 \rightarrow [0 \ 1]$$

$$S_3 \rightarrow [1 \ 1]$$

$$v = \begin{bmatrix} -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 \\ -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 \end{bmatrix}$$

$$u = [-1 \ +1 \ -1 \ +1 \ -1 \ +1 \ -1 \ +1]$$

where, each column in  $v$  corresponds to output code bits for following set of transitions,

$$[S_0 \rightarrow S_0, \ S_0 \rightarrow S_1, \ S_1 \rightarrow S_2, \ S_1 \rightarrow S_3, \ S_2 \rightarrow S_0, \ S_2 \rightarrow S_1, \ S_3 \rightarrow S_2, \ S_3 \rightarrow S_3]$$

and each element in  $u$  corresponds to input bit due to which the above corresponding set of transitions take place.

As there are four different states we have 16 different combinations of state transition at every instant of input but not all of these 16 different transitions are valid transitions at all time instances. Following shows the list of valid state transitions.

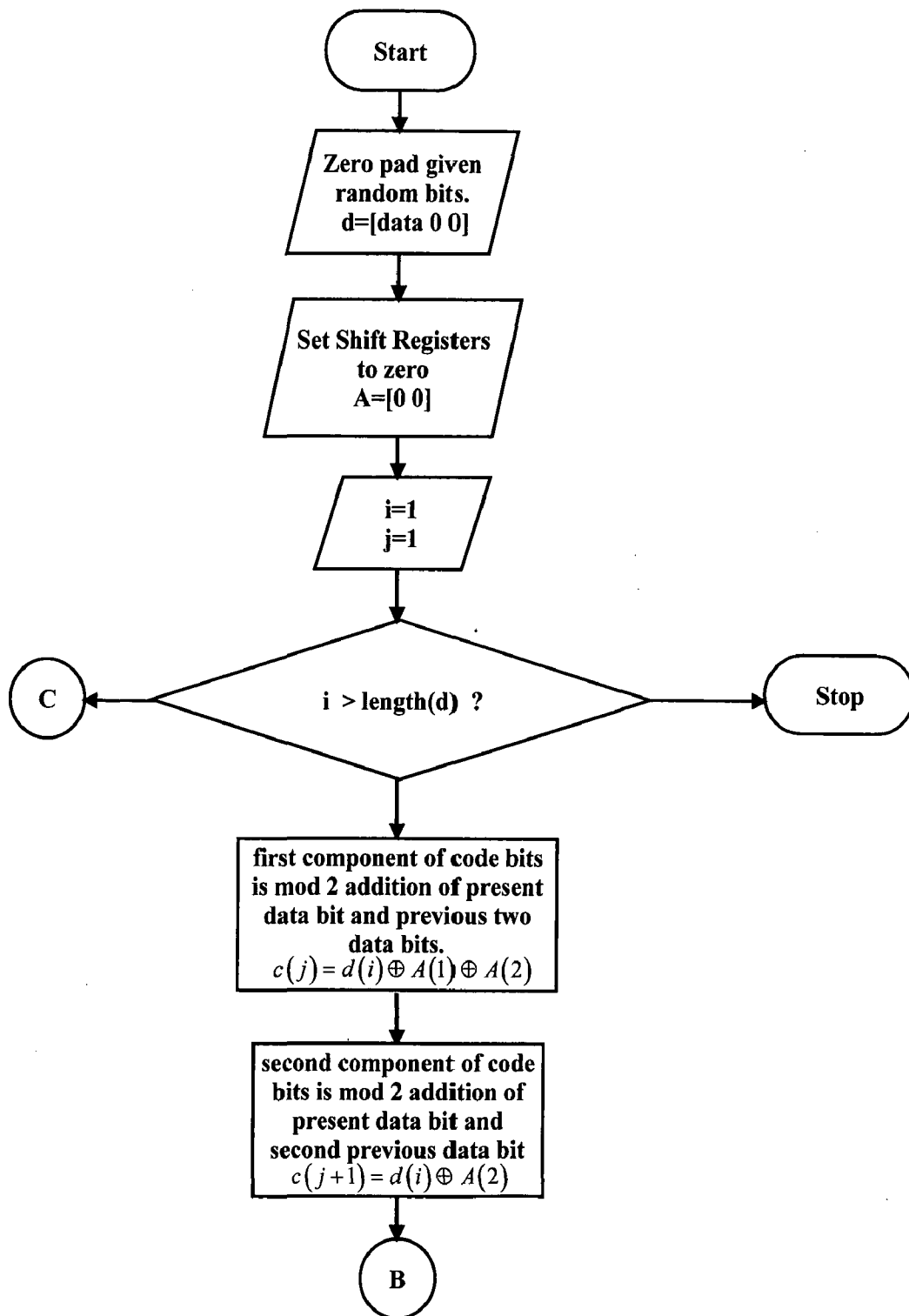
Time Instant (i)	Valid Transitions
1	$(S_0 \rightarrow S_0)$ and $(S_0 \rightarrow S_1)$
2	$(S_0 \rightarrow S_0), (S_0 \rightarrow S_1), (S_1 \rightarrow S_2)$ and $(S_1 \rightarrow S_3)$
3 to ( <i>message frame length</i> - 2)	$(S_0 \rightarrow S_0), (S_0 \rightarrow S_1), (S_1 \rightarrow S_2), (S_1 \rightarrow S_3)$ $(S_2 \rightarrow S_0), (S_2 \rightarrow S_1), (S_3 \rightarrow S_2)$ and $(S_3 \rightarrow S_3)$
( <i>message frame length</i> - 1)	$(S_0 \rightarrow S_0), (S_1 \rightarrow S_2), (S_2 \rightarrow S_0)$ and $(S_3 \rightarrow S_2)$
( <i>message frame length</i> )	$(S_0 \rightarrow S_0)$ and $(S_2 \rightarrow S_0)$

While calculating  $\Gamma_k(s', s)$  all other invalid transition at all time instants is considered as  $-\infty$ . Fig: 4.5 show the flow chart that corresponds to calculating of transition metric. Since there are 8 valid transitions for almost entire message frame duration (except at beginning and ending) we form a matrix of dimension  $8 \times \text{msg frame length}$ .

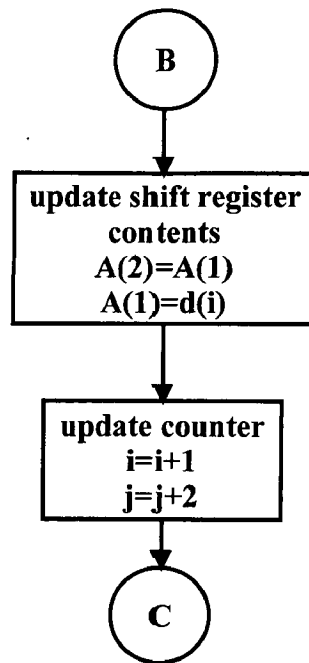
After calculating the value of  $\Gamma_k(s', s)$ , we calculate the value of  $A_k(s)$  and  $B_k(s')$  as given in Eq: 2.35 and Eq: 2.36 respectively. For the convolutional encoding we considered has four different states, as such we calculate at each time instant four different values of  $A_k(s)$  and  $B_k(s')$ .

For calculating forward recursion algorithm, we first initialize  $A_k(s)$  at time instant  $i=1$ , as follows,

$$A_1(s) \equiv \ln \alpha_1(s) = \begin{cases} 0 & s = S_0 \\ -\infty & s \neq S_0 \end{cases} \quad (4.2)$$



**Fig: 4.2** Flow Chart for Rate  $\frac{1}{2}$  Constraint Length 2 Nonsystematic Encoder (*Contd...*)



**Fig: 4.2(Contd...)** Flow Chart for Rate  $\frac{1}{2}$  Constraint Length 2 Nonsystematic Encoder

The following tables explains the possible transition that are needed for calculating forward recursion algorithm

Present State (s)	Possible Transitions to Present State	Respective Previous States Corresponding to Transitions (s')
S <sub>0</sub>	$S_0 \rightarrow S_0$ and $S_2 \rightarrow S_0$	S <sub>0</sub> and S <sub>2</sub>
S <sub>1</sub>	$S_0 \rightarrow S_1$ and $S_2 \rightarrow S_1$	S <sub>0</sub> and S <sub>2</sub>
S <sub>2</sub>	$S_1 \rightarrow S_2$ and $S_3 \rightarrow S_2$	S <sub>1</sub> and S <sub>3</sub>
S <sub>3</sub>	$S_1 \rightarrow S_3$ and $S_3 \rightarrow S_3$	S <sub>1</sub> and S <sub>3</sub>

Flow chart in Fig: 4.6 give the details for calculating  $A_k(s)$  using forward recursive algorithm.

Finally, we can write for the a-posteriori LLRs:

$$L(u_k | \underline{z}) = \ln \left( \frac{\sum_{\substack{(s',s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)}{\sum_{\substack{(s',s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(s') \cdot \gamma_k(s',s) \cdot \beta_k(s)} \right) \quad (2.37)$$

$$= \ln \left( \frac{\sum_{\substack{(s',s) \Rightarrow \\ u_k = +1}} \exp[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)]}{\sum_{\substack{(s',s) \Rightarrow \\ u_k = -1}} \exp[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)]} \right)$$

Now we note that we can apply the  $\max^*$  function to sums of more than two exponential terms by making use of the result,

$$\max^*(x, y, z) \equiv \ln(e^x + e^y + e^z) = \max^*[\max^*(x, y), z], \quad (2.38)$$

Finally, we can express the LLRs as:

$$L(u_k | \underline{z}) = \max^*_{\substack{(s',s) \Rightarrow \\ u_k = +1}} (A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)) - \max^*_{\substack{(s',s) \Rightarrow \\ u_k = -1}} (A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)) \quad (2.39)$$

The above algorithm which computes the LLRs using the log-domain metrics defined in Eq: 2.32, Eq: 2.33 and Eq: 2.34, and uses the  $\max^*$  function defined in Eq: 2.31 is called *log-MAP algorithm*, or the *log-domain BCJR algorithm*. The log-MAP algorithm because it uses just  $\max(\bullet)$  function and a lookup table, is considerably simpler to implement and provides greater stability than the MAP algorithm.

An even simpler algorithm results if we ignore the correction term  $\ln(1 + e^{-|x-y|})$  in the  $\max^*$  function and simply use the approximation,

$$\max^*(x, y) \approx \max(x, y) \quad (2.40)$$

instead. Because the correction term is bounded by  $0 < \ln(1 + e^{-|x-y|}) \leq \ln(2) = 0.693$ , the approximation is reasonably good whenever  $|\max(x, y)| \geq 7$ . The algorithm that computes LLRs using the  $\max$  function instead of  $\max^*$  is called the *MAX-log-MAP algorithm*.

## 2.5 MAP-Equalization Algorithm

Considering coded transmission in mobile system, a mobile radio multi-path channel including transmit and receive filters can be modeled as tapped delay line digital



system. This channel with intersymbol interference (ISI) can be regarded as a convolutional encoder (Fig: 2.8). We view channel encoder and ISI-channel as a serially concatenated coding scheme with channel as an inner encoder. If we know the coefficients  $h_i$  of the discrete time channel model, we can decode the channel by means of MAP symbol estimation or sequence estimation. Output of the ISI-channel is given by:

$$z_k = \sum_{i=0}^L h_i x_{k-i} + w_k \quad (2.41)$$

From Fig: 2.8 it is clear that ISI channel is a rate 1/1 nonsystematic convolutional code whose outputs are in non-binary form. Hence the only difference in the metrics of the equalizer and of the channel decoder appears in the calculation of the branch transition probabilities  $\gamma_k(s', s)$ . We use ‘‘Symbol-by-Symbol’’ MAP-algorithm for both equalization and decoding. For MAP-equalizer, the branch transition probabilities can be calculated as mentioned in Eq: 2.22, since ISI channel is equivalent to a rate 1/1 non-systematic convolutional code, Eq: 2.22 can be rewritten as

$$P(z_k | x_k) = p(z_k | x_k) \cdot dz_k \quad (2.40)$$

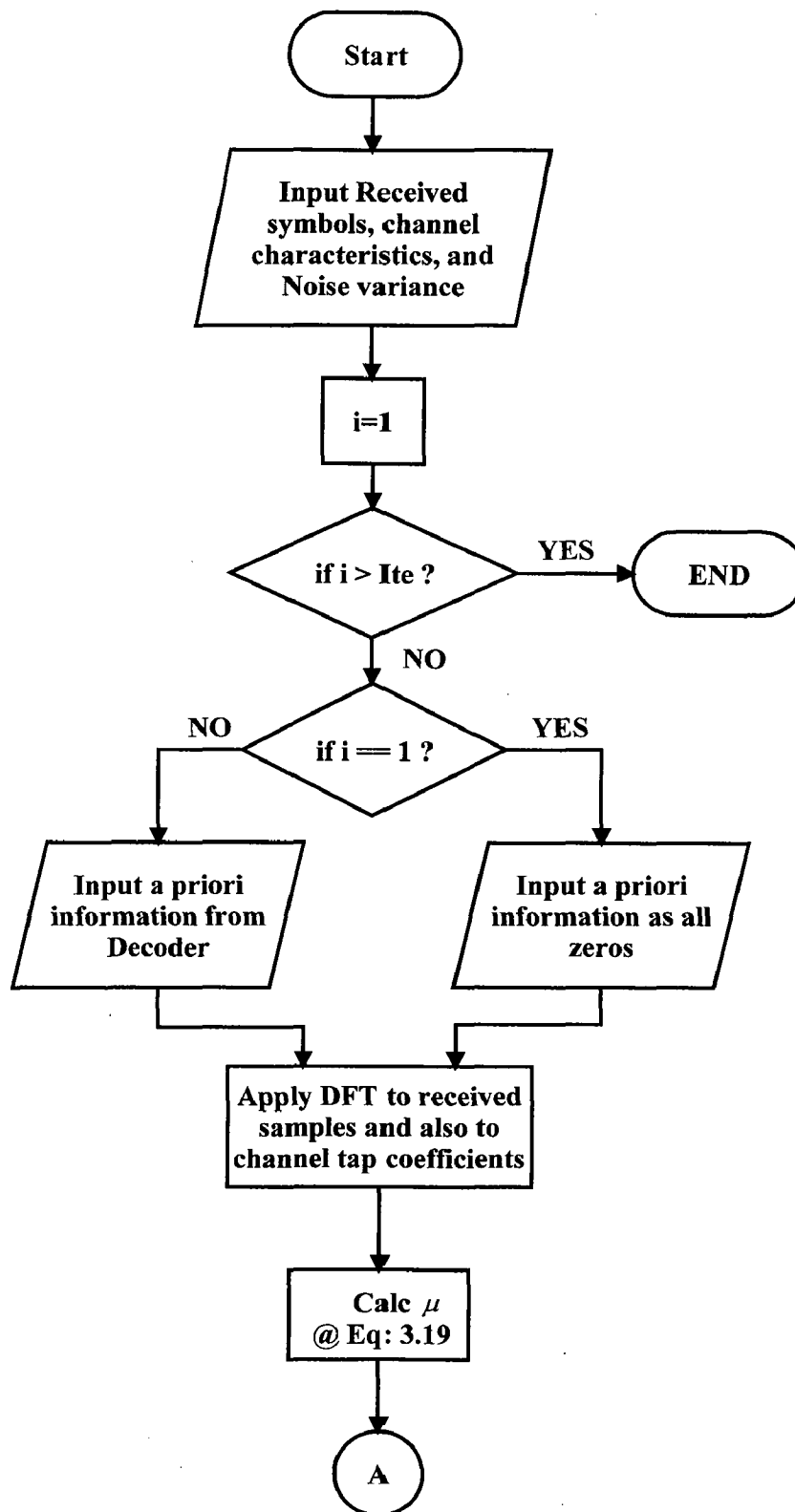
For received signal  $z_k$  and AWGN channel with zero mean and variance  $\sigma^2$

$$p(z_k | x_k) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot \exp \left[ -\frac{\left( z_k - \sum_{i=0}^L h_i x_{k-i} \right)^2}{2 \cdot \sigma^2} \right] \quad (2.41)$$

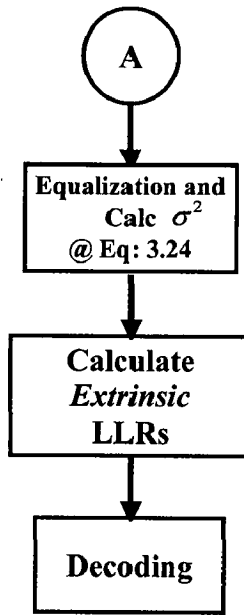
Using the equation  $P(x_k = \pm 1) = A_k \cdot \exp \left( \frac{L(x_k) \cdot x_k}{2} \right)$ , the MAP-equalizer transition can

be calculated as:

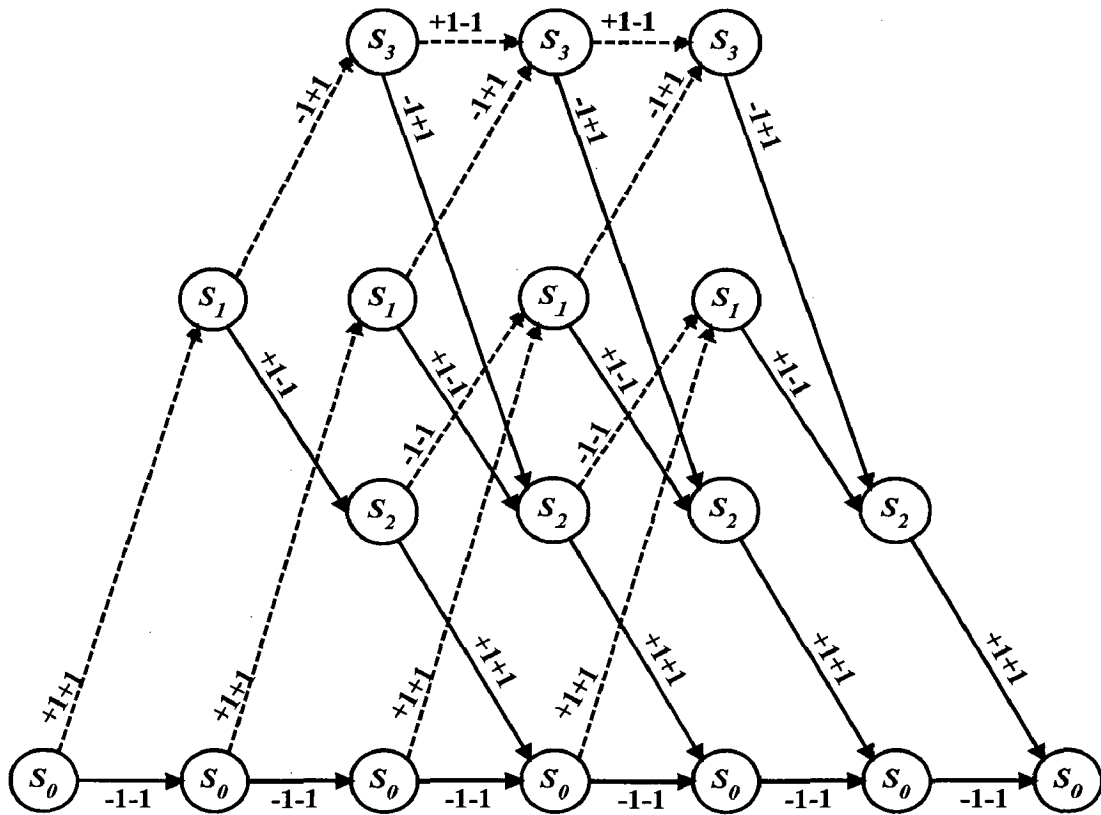
$$\gamma_k(s', s) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot \exp \left[ -\frac{\left( z_k - \sum_{i=0}^L h_i \cdot x_{k-i} \right)^2}{2 \cdot \sigma^2} \right] \cdot A_k \cdot \exp \left( \frac{L(x_k) \cdot x_k}{2} \right) \quad (2.42)$$



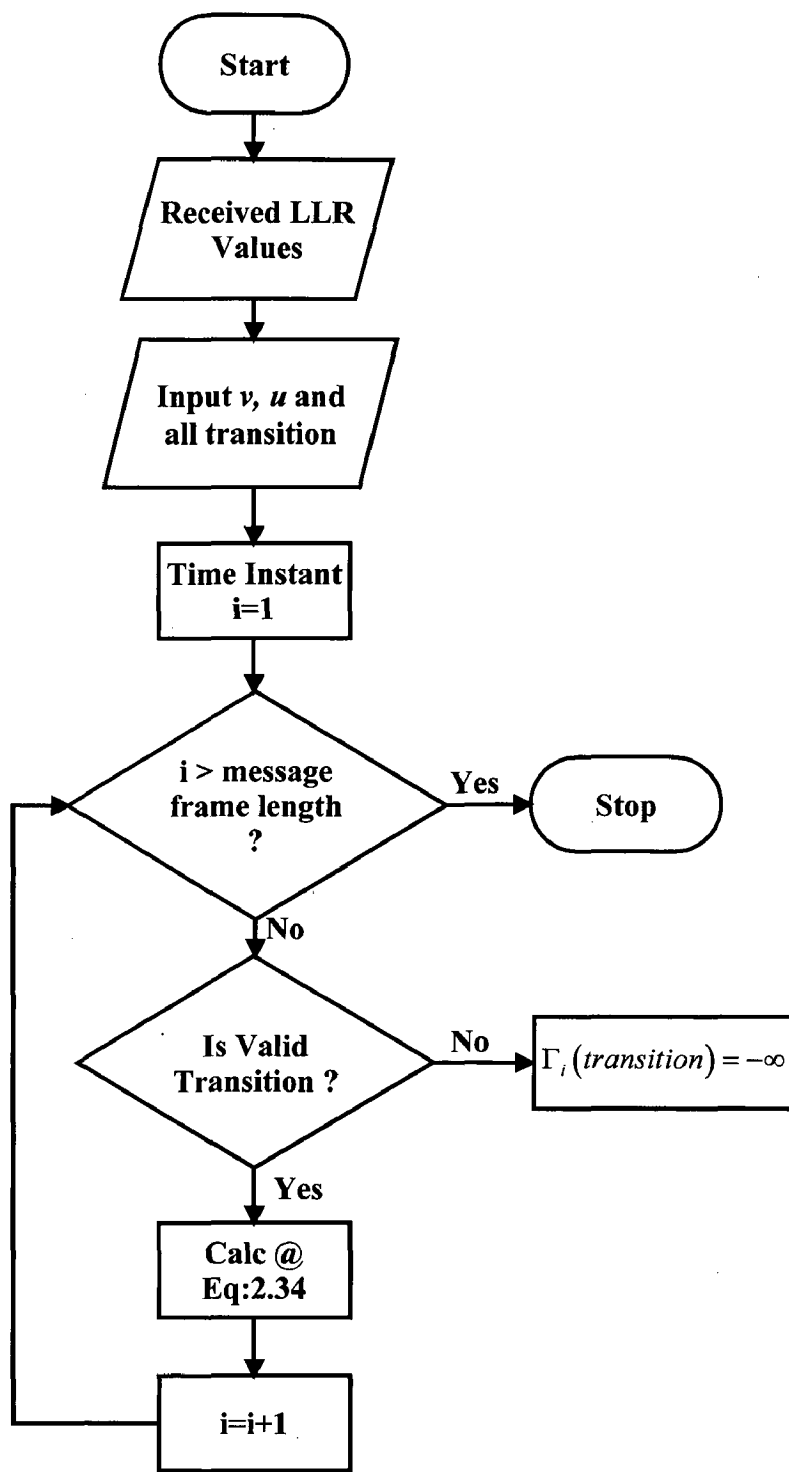
**Fig: 4.3** Flowchart of Iterative Frequency Domain Equalization (*Contd....*)



**Fig: 4.3(Contd....)** Flowchart of Iterative Frequency Domain Equalization



**Fig: 4.4** Decoding Trellis for the (2,1,2) code



**Fig: 4.5** Flow Chart for Transition Metric Calculation of the Log-MAP-Decoder

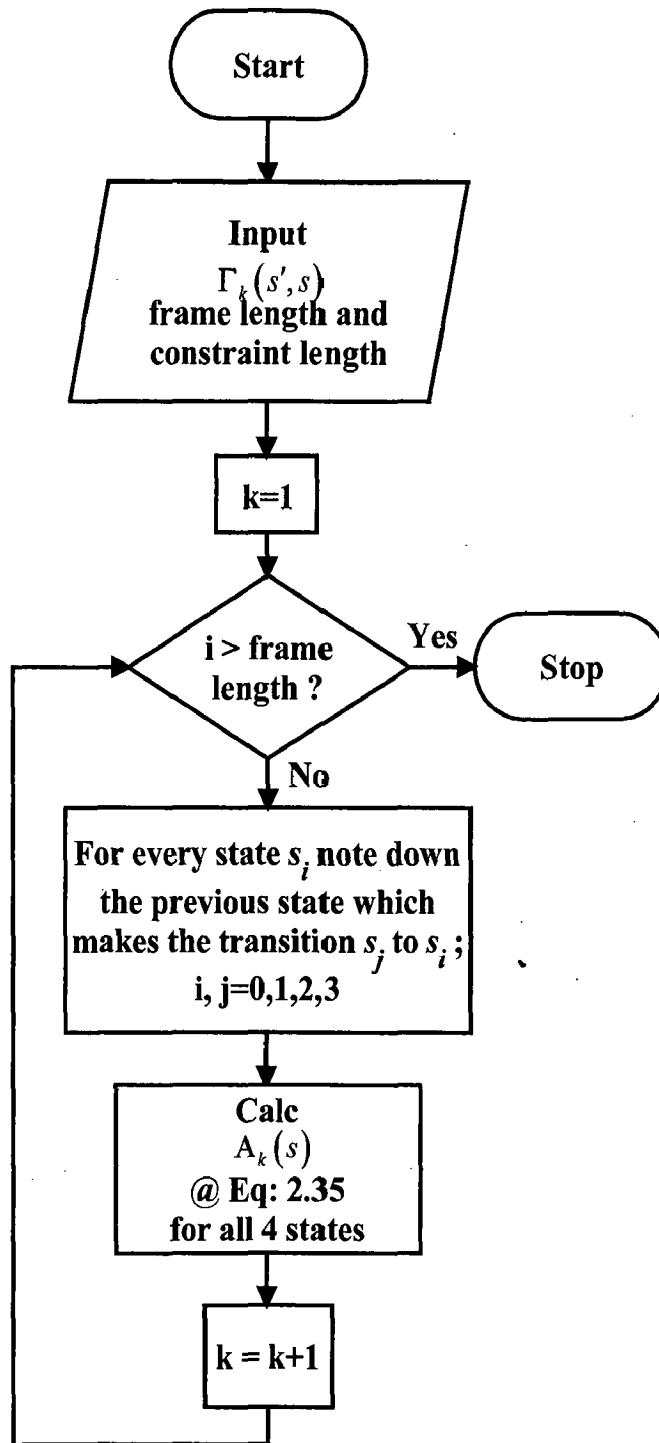


Fig: 4.6 Flow Chart for Calculating  $A_k(s)$  using forward recursive algorithm

For calculating backward recursion algorithm, we first initialize  $B_k(s)$  at time instant  $i = \text{frame length}$ , as follows,

$$B_{\text{framelength}+1}(s') \equiv \ln \beta_{\text{framelength}+1}(s') = \begin{cases} 0 & s' = S_0 \\ -\infty & s' \neq S_0 \end{cases} \quad (4.3)$$

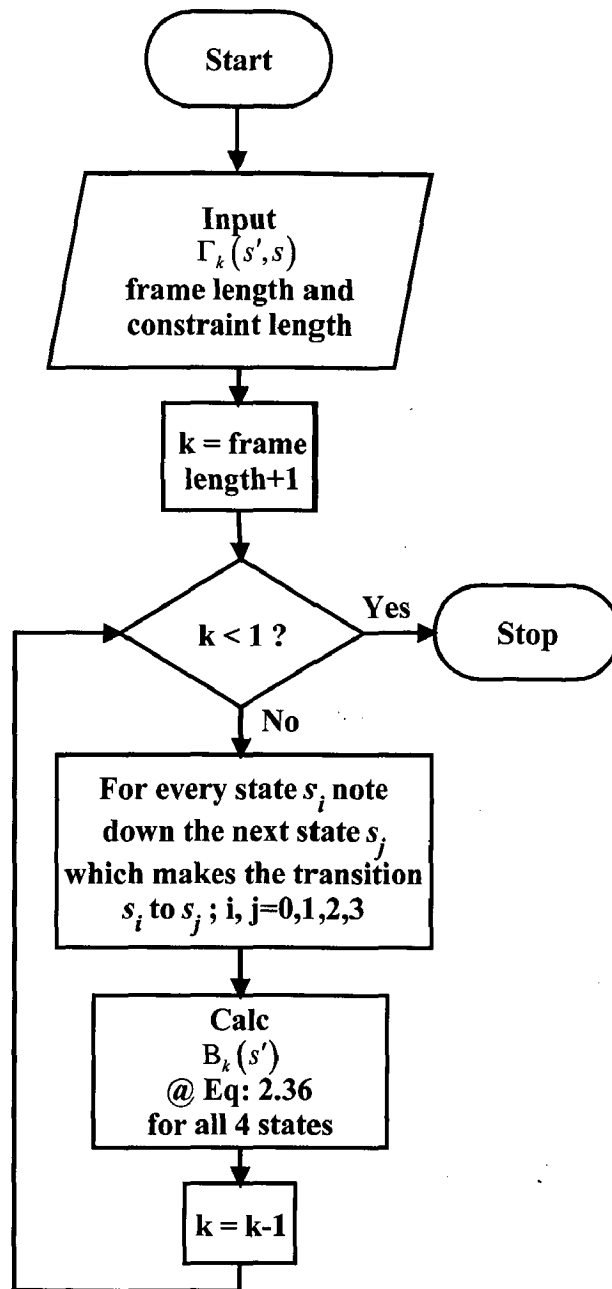
The following tables explains the possible transition that are needed for calculating backward recursion algorithm

Present State ( $s'$ )	Possible Transitions from Present State	Respective Next States Corresponding to Transitions ( $s$ )
$S_0$	$S_0 \rightarrow S_0$ and $S_0 \rightarrow S_1$	$S_0$ and $S_1$
$S_1$	$S_1 \rightarrow S_2$ and $S_1 \rightarrow S_3$	$S_2$ and $S_3$
$S_2$	$S_2 \rightarrow S_0$ and $S_2 \rightarrow S_1$	$S_0$ and $S_1$
$S_3$	$S_3 \rightarrow S_2$ and $S_3 \rightarrow S_3$	$S_2$ and $S_3$

Flow chart in Fig: 4.7 give the details for calculating  $B_k(s')$  using forward recursive algorithm.

After calculating all the above metrics we now calculate the a posteriori value the output of decoder for all the coded bits and in final iteration we also calculate the data estimate. The LLR for coded bits is given by similar expression as in Eq: 2.39, which is modified as below.

$$L(c_{k,i} | \underline{z}) = \max_{\substack{(s',s) \Rightarrow \\ c_{k,i}=+1}}^* (A_k(s') + \Gamma_k(s',s) + B_{k+1}(s)) \\ - \max_{\substack{(s',s) \Rightarrow \\ c_{k,i}=-1}}^* (A_k(s') + \Gamma_k(s',s) + B_{k+1}(s)) \\ i = 1, 2$$



**Fig: 4.7** Flow Chart for Calculating  $B_k(s)$  using backward recursive algorithm

## Chapter 5

### SIMULATION RESULTS

---

In simulation, the performance of iterative frequency domain equalization and decoding is evaluated for three different ISI channels, which are given below.

- i) Channel Proakis C: It is the worst frequency selective channel due to its spectral nulls. The channel parameters for this channel are:  $h = [0.227 \ 0.46 \ 0.688 \ 0.46 \ 0.227]$ .
- ii) Channel Proakis B: This ISI channel also provides poor performance for the receiver using MMSE criterion for equalization. The channel parameters are:  $h = [0.407 \ 0.815 \ 0.407]$ .
- iii) Channel A:  $h = [0.336 \ 0.858 \ 0.336]$ .

For simulation purpose we have transmitted code bits frame of length  $L=512$ , the choice for this value was based on pseudorandom interleaver design, because the quadratic interleaver we considered have statistical properties similar to those of randomly chosen interleaver and gives good performance if the interleaver size is a power of two. We have considered 2500 such frames for simulation purpose. For bit error rate calculations we have considered 0-12 db SNR range, at each SNR value independent and identically distributed Gaussian random variable samples are generated. Noise variance is obtained by following equation,

$$N \text{ var} = 0.5 \times 10^{(-SNR / 10)}$$

and noise samples are generated using the following formula,

for  $i = 1$  to transmitted frame length

$$n(i) = \sqrt{2 \times N \text{ var}} \times \text{erfinv}(2 \times \text{rand} - 1)$$

end

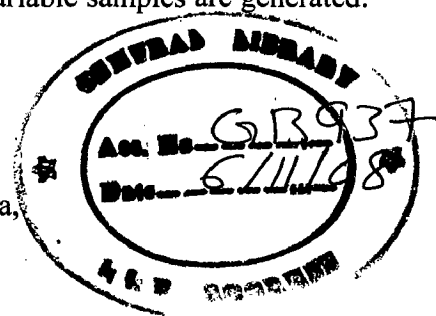
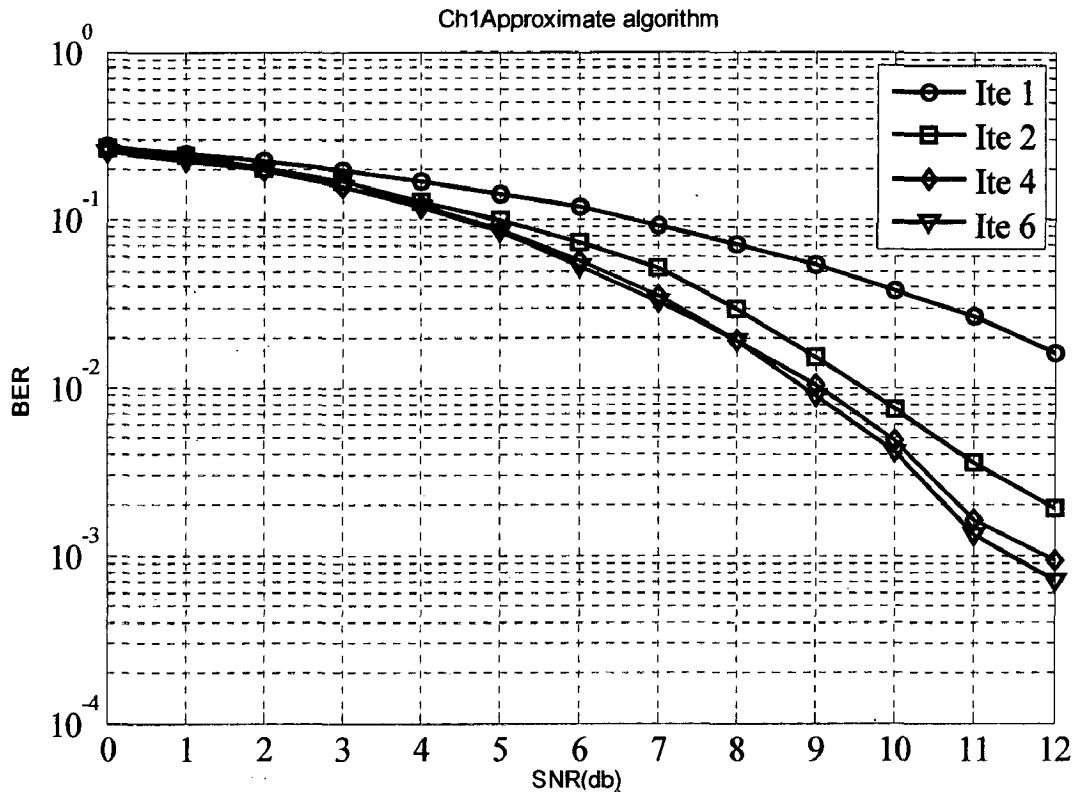




Fig: 5.1 show the simulation details of approximate algorithm for channel Proakis C. The simulation is carried for four different iterations; with increase in iteration value there is improvement in BER performance. Notice from figure that at iteration 1 and SNR = 12 db we obtained bit error rate of order  $10^{-2}$ . With iteration 2 this order increases to  $10^{-3}$ , finally at iteration 4 and 6 BER attains a saturated value of  $7-9 \times 10^{-4}$ .

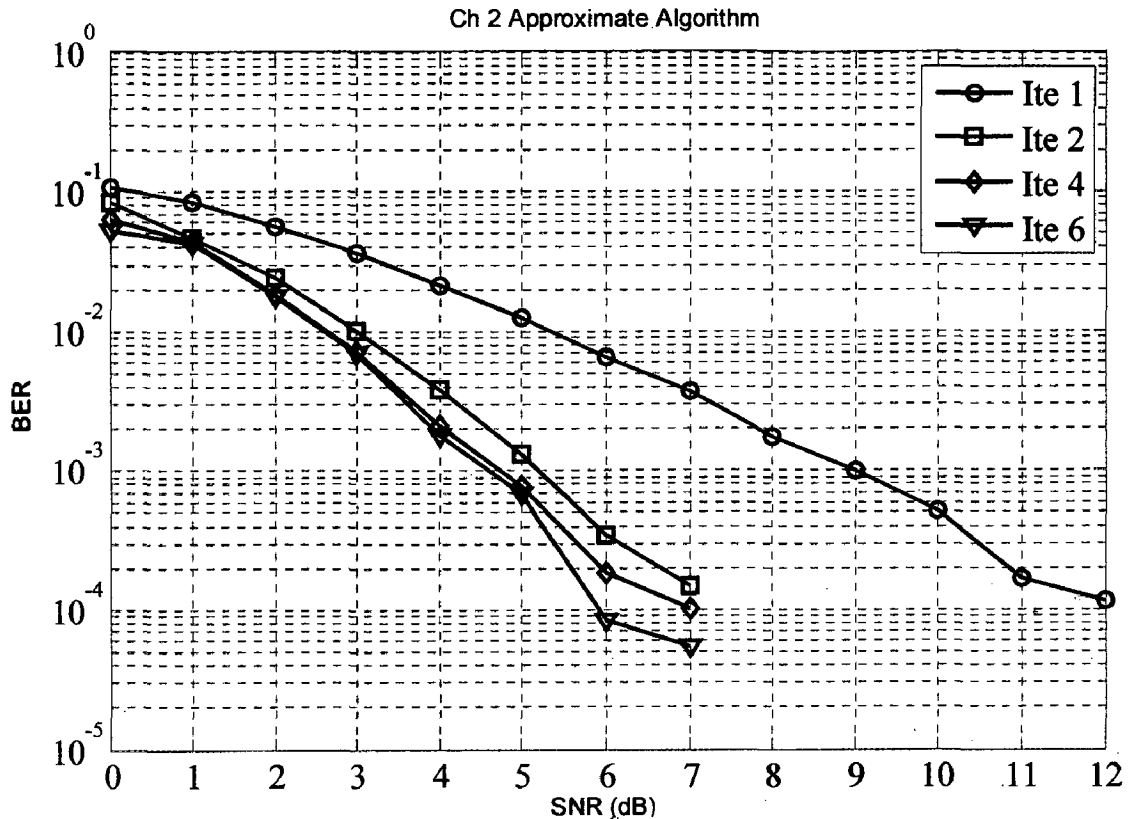


**Fig: 5.1** Simulation Result of Ch1 Approximate Algorithm

At Iteration 1, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (12 – 12.5 db). At iteration 2, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (9 - 9.5 db) and BER of order  $10^{-3}$  is attained at SNR  $\sim$  (12 – 12.5 db). At iteration 4, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (8.5 – 9 db) and BER of order  $10^{-3}$  is attained at (11 - 11.5 db).

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-2}$  there is a gain of  $\sim$  3 db, whereas for iteration 4 compared to iteration 2 this gain is only 0.5 db. Similarly for iteration 4 compared to iteration 2, to attain BER of order  $10^{-3}$  there is a gain of  $\sim$  1 db.

Fig: 5.2 show the simulation details of approximate algorithm for channel Proakis B. The simulation is carried for four different iterations; with increase in iteration value there is improvement in BER performance. Notice from figure that at iteration 1 and SNR = 7db, SNR = 12 db we obtained bit error rate of order  $10^{-3}$  and  $10^{-4}$  respectively. With iteration 2, and SNR = 7db the order increases to  $10^{-4}$ , finally at iteration 4 and 6 BER attains a saturated value of  $7-9 \times 10^{-5}$  at SNR = 7 db.

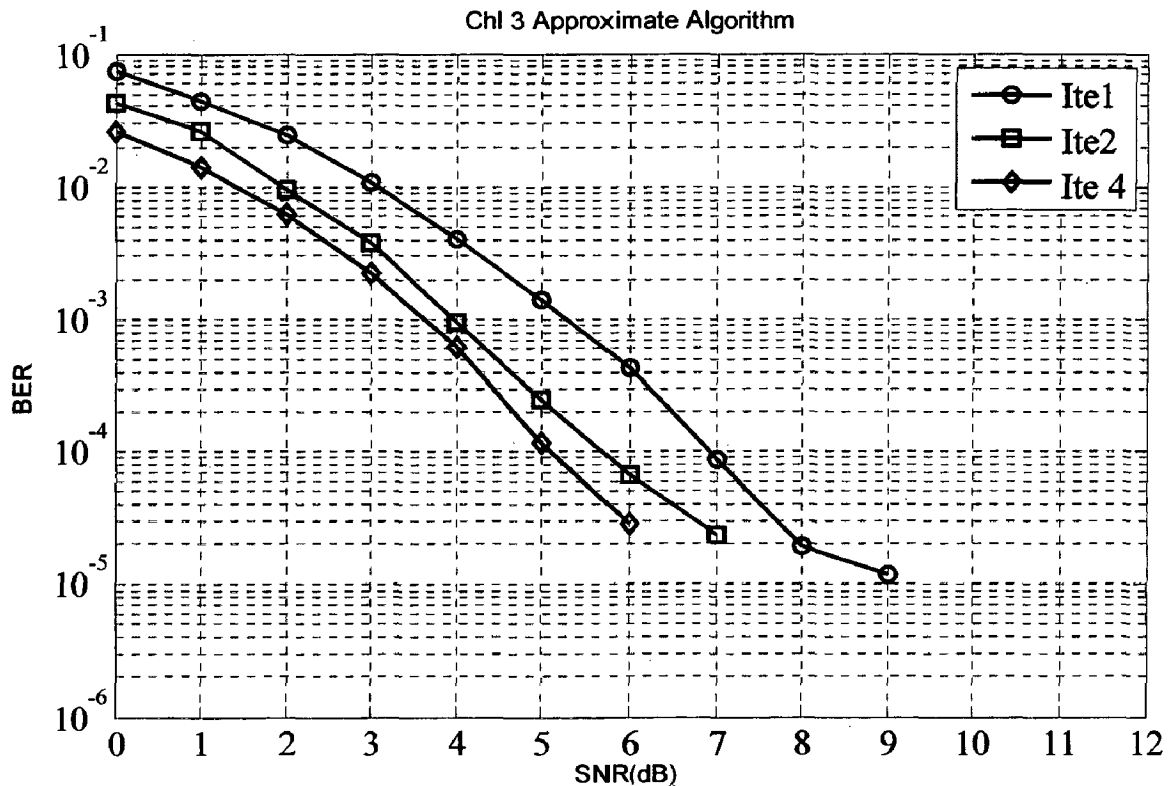


**Fig: 5.2** Simulation Result for Ch2 Approximate Algorithm

At Iteration 1, BER of order  $10^{-3}$  is attained at SNR  $\sim$  (8 – 8.5 db). At iteration 2, BER of order  $10^{-3}$  is attained at SNR  $\sim$  5 db and BER of order  $10^{-4}$  is attained at SNR  $\sim$  7.5 db. At iteration 4, BER of order  $10^{-3}$  is attained at SNR  $\sim$  (4 – 4.5 db) and BER of order  $10^{-4}$  is attained at (6 – 7 db).

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-3}$  there is a gain of  $\sim$  (3 – 3.5 db), whereas for iteration 4 compared to iteration 2 this gain is only (0.5 – 1 db). Similarly for iteration 4 compared to iteration 2, to attain BER of order  $10^{-4}$  there is a gain of  $\sim$  (1 – 1.5 db).

Fig: 5.3 show the simulation details of approximate algorithm for channel A. The simulation is carried for three different iterations; with increase in iteration value there is improvement in BER performance. Notice from figure that at iteration 1 for SNR = 6db and SNR = 9 db we obtained bit error rate of order  $10^{-4}$  and  $10^{-5}$  respectively. With iteration 2, and SNR = 6db the order increases to  $6-7 \times 10^{-5}$ , finally at iteration 4 BER attains a value of  $3 \times 10^{-5}$ .



**Fig: 5.3 Simulation Result for Ch3 Approximate Algorithm**

At Iteration 1, BER of order  $10^{-3}$  is attained at SNR  $\sim$  (5 – 5.5 db). At iteration 2, BER of order  $10^{-3}$  is attained at SNR  $\sim$  4 db and BER of order  $10^{-4}$  is attained at SNR  $\sim$  5.5 db. At iteration 4, BER of order  $10^{-3}$  is attained at SNR  $\sim$  3.5 db and BER of order  $10^{-4}$  is attained at 5 db.

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-3}$  there is a gain of  $\sim$  (1 – 1.5 db), whereas for iteration 4 compared to iteration 2 this gain is 1.5 db. Similarly for iteration 4 compared to iteration 2, to attain BER of order  $10^{-4}$  there is a gain of  $\sim$ 0.5 db.

Fig: 5.4 show the simulation details of approximate algorithm with average mean for channel Proakis C. The simulation is carried for three different iterations; with increase in iteration value there is no improvement in BER performance. Notice from figure that at iterations 1, 2, and 4 the BER performance obtained is similar to that of the BER performance obtained in iteration 1 for approximate algorithm.

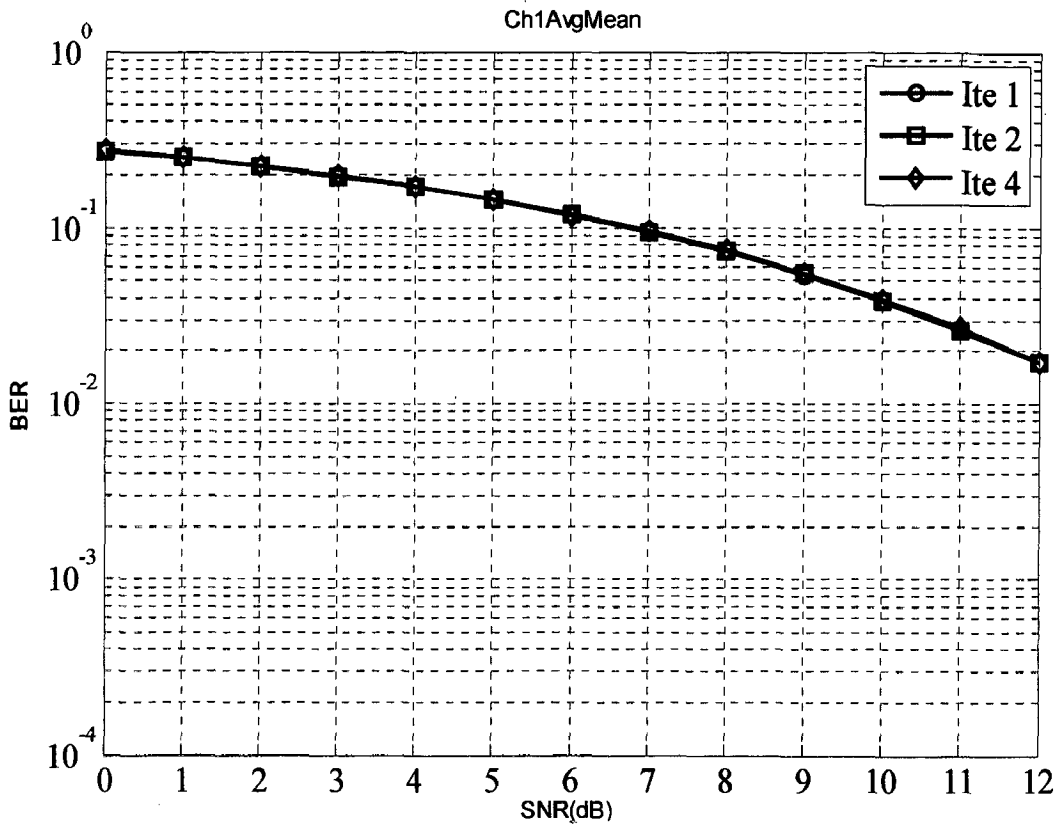


Fig: 5.4 Simulation Result for Ch1 Average Mean Algorithm

Fig: 5.5 show the simulation details of approximate algorithm with average mean for channel Proakis B. The simulation is carried for three different iterations; with increase in iteration value there is no improvement in BER performance. Notice from figure that at iterations 1, 2, and 4 the BER performance obtained is similar to that of the BER performance obtained in iteration 1 for approximate algorithm.

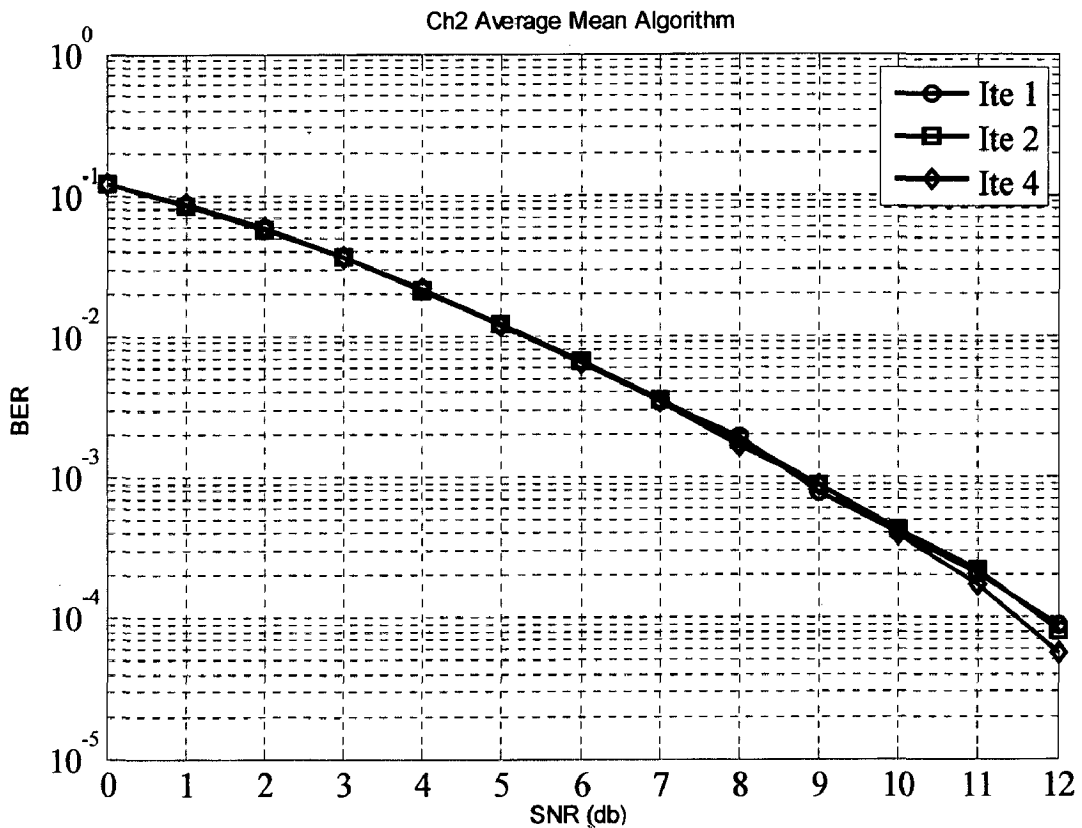
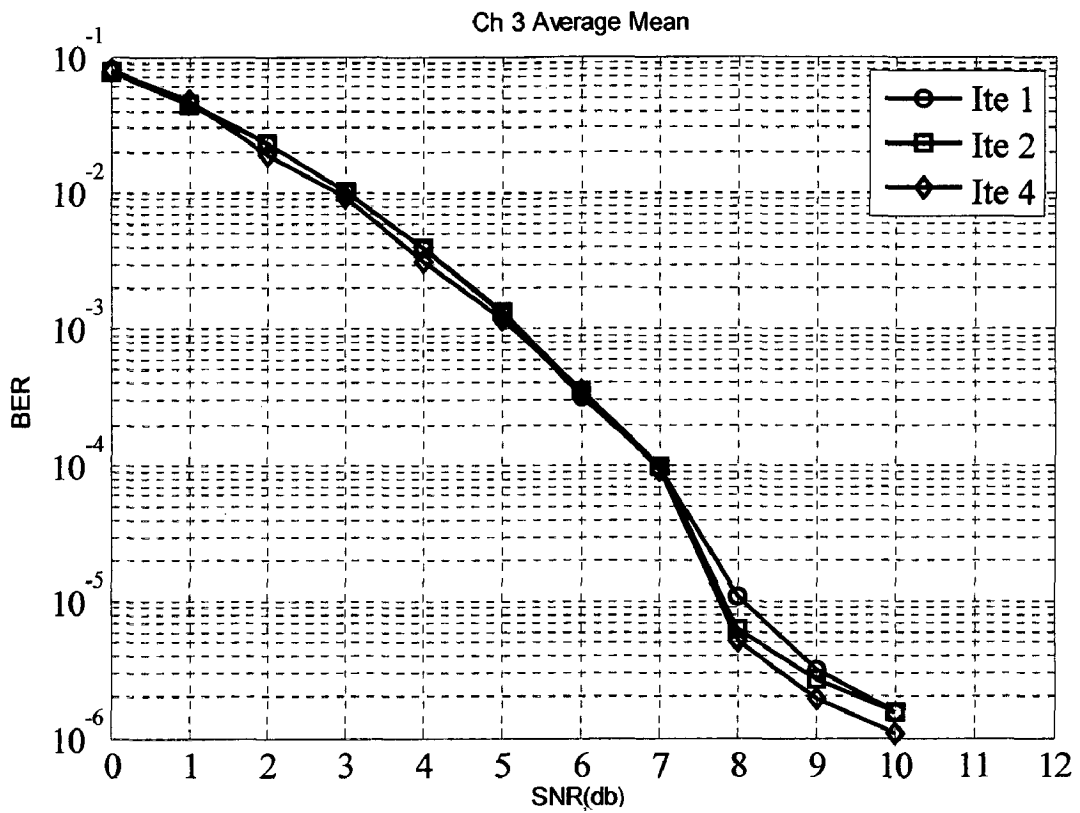


Fig: 5.5 Simulation Result for Ch2 Average Mean Algorithm

Fig: 5.6 show the simulation details of approximate algorithm with average mean for channel A. The simulation is carried for three different iterations; with increase in iteration value there is no improvement in BER performance. Notice from figure that at iterations 1, 2, and 4 the BER performance obtained is better to that of the BER performance obtained in iteration 1 for approximate algorithm at SNRs greater than 7 db.



**Fig: 5.6 Simulation Result for Ch3 Average Mean Algorithm**

Fig: 5.7 show the simulation details of approximate algorithm with average variance for channel Proakis C. The simulation is carried for four different iterations; with increase in iteration value there is good improvement in BER performance. Notice from figure that at iterations 1, SNR = 12db BER is of  $2 \times 10^{-2}$  whereas at iteration 2 for same SNR value BER is of order  $1.5 \times 10^{-3}$  and for iterations 4 and 8, the BER attains a value around  $1-2 \times 10^{-4}$  which is good performance compared to approximate algorithm.

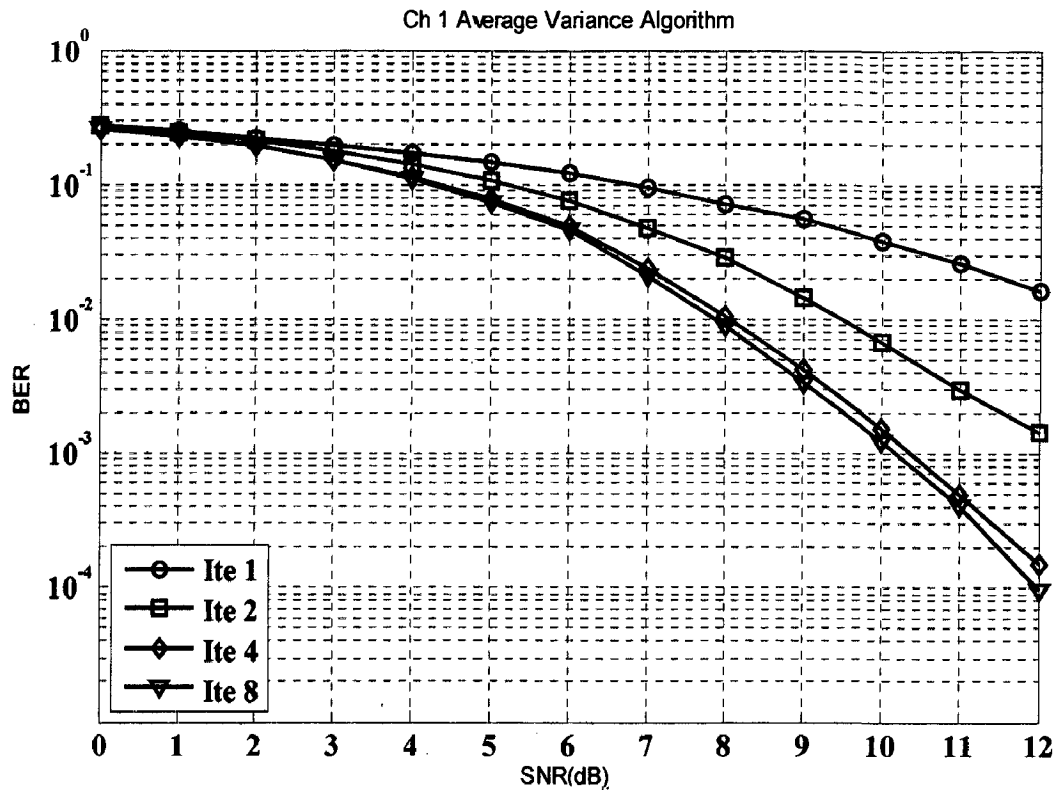
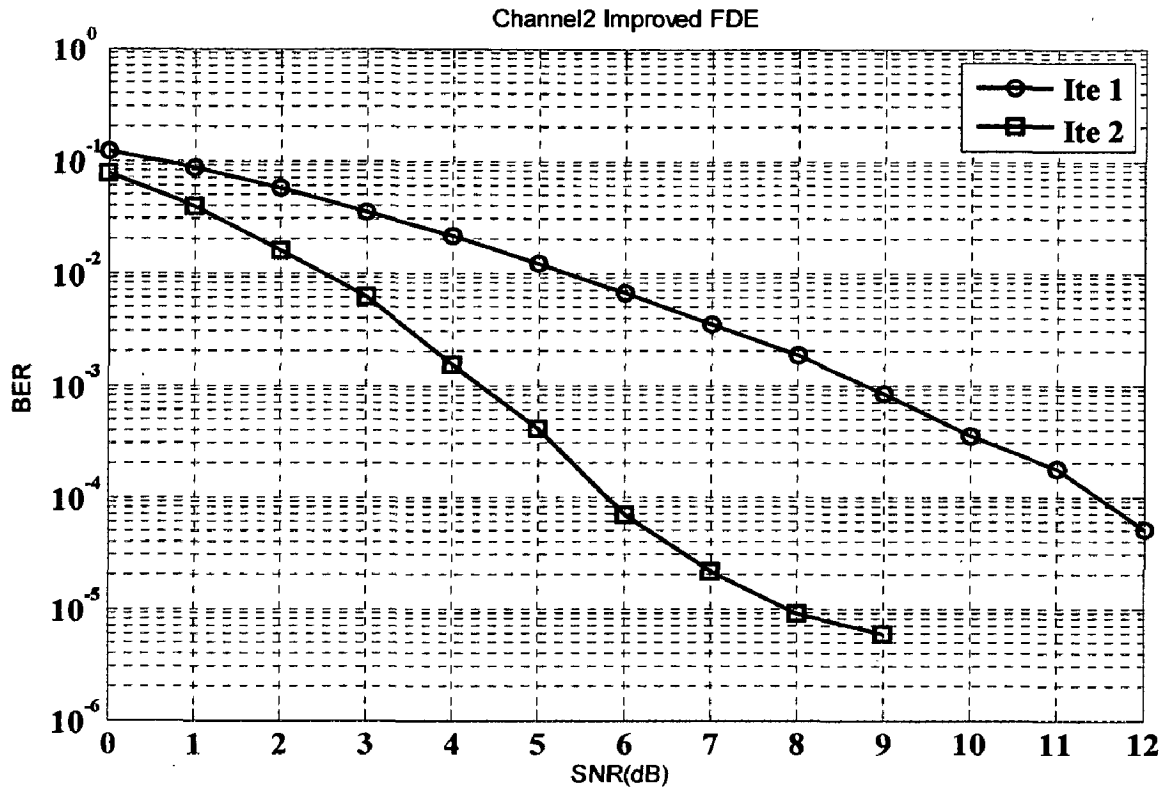


Fig: 5.7 Simulation Result for Ch1 Average Variance Algorithm

At Iteration 1, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (12 – 12.5 db). At iteration 2, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (9 - 9.5 db) and BER of order  $10^{-3}$  is attained at SNR  $\sim$  (12 – 12.5 db). At iteration 4, BER of order  $10^{-2}$  is attained at SNR  $\sim$  (7.5 – 8 db) and BER of order  $10^{-3}$  is attained at  $\sim$ 10 db.

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-2}$  there is a gain of  $\sim$  3 db, whereas for iteration 4 compared to iteration 2 this gain is 1.5 db. Similarly for iteration 4 compared to iteration 2, to attain BER of order  $10^{-3}$  there is a gain of  $\sim$  (2 – 2.5 db).

Fig: 5.8 show the simulation details of approximate algorithm with average variance for channel Proakis B. The simulation is carried for two different iterations; with increase in iteration value there is good improvement in BER performance. Notice from figure that at iterations 1, SNR = 8 db and SNR = 12db BER is  $2 \times 10^{-3}$  and  $5 \times 10^{-5}$  respectively, whereas at iteration 2 for SNR = 8 db value BER is of order  $9 \times 10^{-6}$ .



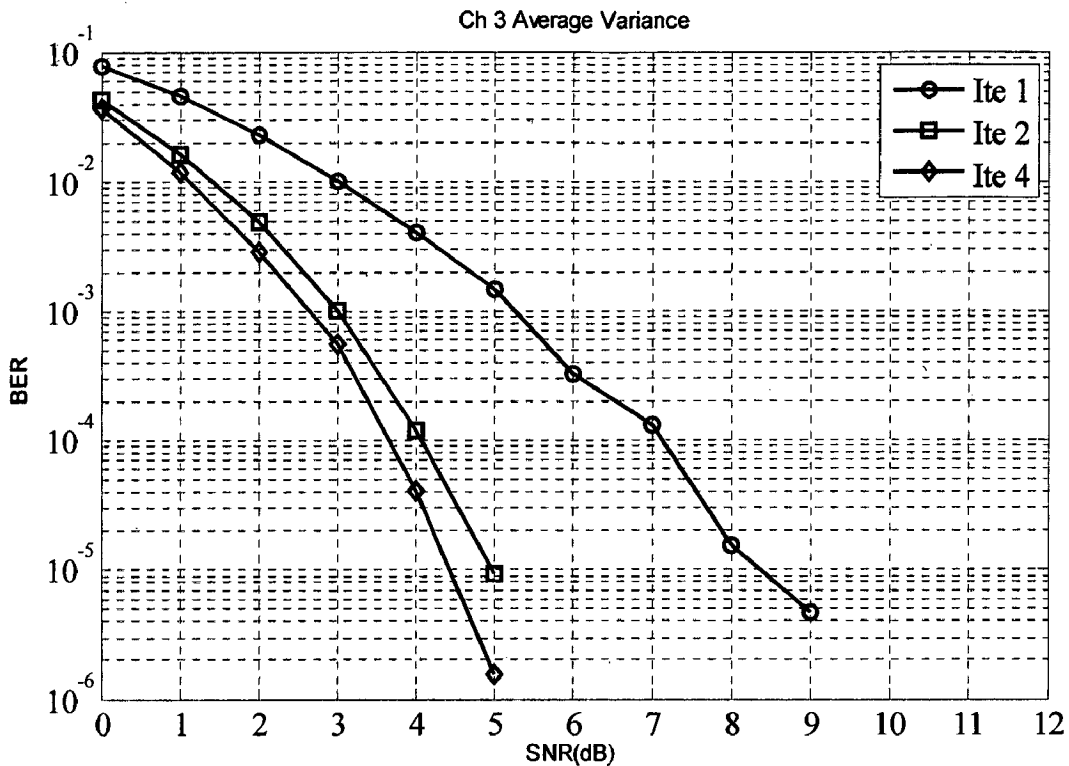
**Fig: 5.8 Simulation Results for Ch2 Average Variance Algorithm**

At Iteration 1, BER of order  $10^{-3}$  is attained at SNR  $\sim$  (8.5 – 9 db) and BER of order  $10^{-4}$  is attained at SNR  $\sim$  (11 - 11.5 db). At iteration 2, BER of order  $10^{-3}$  is attained at SNR  $\sim$  (4 - 4.5 db) and BER of order  $10^{-4}$  is attained at SNR  $\sim$  (5.5 – 6 db).

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-3}$  there is a gain of  $\sim$  4.5 db. Similarly to attain BER of order  $10^{-4}$  there is a gain of  $\sim$  5.5 db.



Fig: 5.9 show the simulation details of approximate algorithm with average variance for channel A. The simulation is carried for three different iterations; with increase in iteration value there is good improvement in BER performance. Notice from figure that at iterations 1, SNR = 5 db and SNR = 9 db BER is  $\sim 10^{-3}$  and  $\sim 5 \times 10^{-6}$  respectively, whereas at iteration 2 for SNR = 5 db value BER is of order  $\sim 10^{-5}$ . For iteration 4 at SNR = 5 db BER is of order  $\sim 10^{-6}$ .



**Fig: 5.9 Simulation Results for Ch3 Average Variance Algorithm**

At Iteration 1, BER of order  $10^{-3}$  and  $10^{-4}$  are attained at SNR  $\sim (5 - 5.5 \text{ db})$  and  $\sim 7 \text{ db}$  respectively. At iteration 2, BER of order  $10^{-3}$ ,  $10^{-4}$ , and  $10^{-5}$  are attained at SNR  $\sim 3 \text{ db}$ ,  $4 \text{ db}$ , and  $5 \text{ db}$  respectively. At iteration 4, BER of order  $10^{-4}$  and  $10^{-5}$  are attained at SNR  $\sim 3.5 \text{ db}$  and  $4.5 \text{ db}$ .

Notice that for iteration 2 compared to iteration 1, to attain a BER of order  $10^{-3}$  and  $10^{-4}$  there is a gain of  $\sim (2 - 2.5 \text{ db})$  and  $3 \text{ db}$  respectively, whereas for iteration 4 compared to iteration 2, to attain BER of order  $10^{-4}$  and  $10^{-5}$  the gain is only  $0.5 \text{ db}$  for both.

## Chapter 6

### CONCLUSIONS AND FUTURE SCOPE

---

In this dissertation, iterative frequency domain equalization and improved MAX-LOG-MAP decoding technique has been considered in detail. In this approach MMSE linear equalizer has been used to equalize the ISI-channel, this Equalizer has replaced the original MAP equalizer. We have used the approximate version of linear MMSE equalization, in addition few more approximations were also considered. We have evaluated the performance of these techniques for rate  $\frac{1}{2}$ , constraint length 2, non-systematic convolutional encoder as such we need 4 states MAP decoder.

For MAP decoding we have used an improved MAX-LOG-MAP algorithm as this algorithm produces results almost equivalent to the optimum LOG-MAP algorithm and has obvious advantage over implementation.

The following are the assumptions made for MMSE Linear equalization,

- ❖ No *a priori* information about transmitted bits as such the transmitted bit statistics are zero mean and unit variance, for all the transmitted bits in a frame. This assumption is taken for initial iteration, and for further iteration the soft extrinsic output from the decoder is taken as *a priori* information and we calculate the statics according to  $\bar{x}_n = \tanh\left(\frac{1}{2}L_e^D(x_n)\right)$  and  $v_n = 1 - |x_n|^2$ .
- ❖ For first iteration, we consider no a priori information as such the statics are same as those with the first iteration assumptions considered in previous assumption. For further iterations, we take the block average of mean, as given by  $\bar{x} \equiv \sum_{v_n} \bar{x}_n = \sum_{v_n} \tanh\left(\frac{1}{2}L_e^D(x_n)\right)$  for entire frame and assign this average value for all  $\bar{x}_n$ . Similar to previous assumption we consider  $v_n = 1 - |x_n|^2$ .
- ❖ In this assumption, for first iteration, similar to previous assumption we consider no *a priori* information. For further iterations we calculate the

statistics according to  $\bar{x}_n = \tanh\left(\frac{1}{2}L_e^D(x_n)\right)$  and consider block average of

variance, as given by  $\nu \equiv \sum_{\forall n} \nu_n = \sum_{\forall n} 1 - |x_n|^2$ .

From simulation results it may be observed that the BER performance characteristics increases with successive iterations and the improvement in performance of iterative Equalization and decoding is dependent on channel characteristics. We have considered three different channel characteristics for simulation purpose.

Approximate algorithm with average variance has better performance characteristics. From Fig: 5.7, Fig: 5.8 and Fig: 5.9 observe that at low SNR values we are obtaining good order of BER, but this gain in performance is obtained at a cost of complexity as this algorithm requires calculation of matrix inversion whose order will be dependent on transmitted bit frame length.

Even for highly dispersive channel Proakis C we have obtained a BER of Order  $10^{-4}$  at SNR = 12 db with only four iterations. Hence it may be concluded that the iterative equalization and decoding is an efficient method for equalization and decoding in case of channels with significant amount of ISI.

Obviously there is a future scope for this work, instead of calculating the time varying filter coefficients in every iteration using matrix inversion, there are recursive algorithms to compute them which reduces load tremendously. Other powerful coding methods like Low Density Parity Check (LDPC) codes can also be used for encoding and decoding for obtaining still better performance.

## REFERENCES

---

- [1]. J. Proakis, *Digital Communications*, 3<sup>rd</sup> Ed. New York: McGraw-Hill, 1995.
- [2]. S. Haykin, *Communication Systems*, 3<sup>rd</sup> Ed. New York: Wiley, 1994.
- [3]. S. Lin and J. J. Costello, *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [4]. L. R. Bahl *et al.*, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transaction on Information Theory*, vol. IT-20, pp. 284-287, March 1974.
- [5]. W. Koch and A. Baier, "Optimal and sub-optimum detection of coded data distributed by time varying Intersymbol interference", in *IEEE Proceedings on the Global Telecommunications Conference '90*, pp. 1679-1684, December 1990.
- [6]. D. Yellin, A. Vardy, and O. Amrani, "Joint equalization and coding for Intersymbol interference channels," *IEEE Transactions on Information Theory*, vol. 43, no. 2, pp. 409-425, March 1997.
- [7]. Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with Intersymbol interference," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 704-713, May 1995.
- [8]. S. L. Ariyavisitakul and Y. Li, "Joint coding and decision feedback equalization for broadband wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 9, pp. 1670-1678, December 1998.
- [9]. C. Berrou, A. Glavieux, "Near Optimum error correcting coding and decoding: Turbo codes," *IEEE Transactions on Communication*, vol. 44, pp. 1261-1271, October 1996.
- [10]. M. Tuchler, R. Koetter and A. C. Singer, "Turbo equalization: principles and new results," *IEEE Transaction on Communication*, vol. 50, no. 5, pp. 754-767, May 2002.
- [11]. L. Hanzo, T. H. Liew, B. L. Yeap, *Turbo Coding, Turbo Equalization and Space-Time Coding for Transmission over Wireless Channels*.
- [12]. L. R. Bhal, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284-87, March 1974.

- [13]. M. Tuchler, A. Singer, and R. Koetter, "Minimum mean square error equalization using a priori information," *IEEE Transactions on Signal Processing*, vol. 50, pp. 673-683, March 2002
- [14]. M. Tuchler and J. Hagenauer, "Turbo equalization using frequency domain equalizers," *Proc. Allerton Conf.*, Monticello, IL, USA, Oct 2000.
- [15]. M. Tuchler and J. Hagenauer, "Linear time and frequency domain Turbo equalization," *Proc. 53rd Vehicular Technology Conf. (Spring)*, Rhodes, Greece, pp. 1449-1453, May 2001.
- [16]. P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119-125, Apr 1997.
- [17]. S. Talakoub, L. Sabeti, B. Shahrava, M. Ahmadi, "An Improved Max-Log-MAP Algorithm for Turbo Decoding and Turbo Equalization," *IEEE Transactions on Instrumentation and measurement*, vol. 56, no. 3, pp. 1058-1063, Jun 2007.
- [18]. O. Y. Takeshita and D. J. Costello, Jr., "New Deterministic Interleaver Designs for Turbo Codes," *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 1988-2006, September 2000.