CHAPTER 2

LITERATURE REVIEW, TURBO ENCODER, DECODER AND MODIFICATION OF TURBO DECODING ALGORITHMS

2.1 INTRODUCTION

This chapter gives detailed literature survey, various research groups working with Turbo code around the globe. Turbo encoder, Turbo decoder are explained with the aid of the Maximum A Posteriori Probability (MAP) algorithm and SOVA. Further, iterative Turbo decoding mathematical preliminaries are discussed. The results of the proposed modified SOVA and modified log MAP decoding algorithms are compared in the light of improved performance.

2.2 LITERATURE REVIEW

Turbo code is a parallel concatenation of two convolutional codes separated by a random interleaver. It is near channel capacity error correcting code. This error correcting code is able to transmit information across the channel with an arbitrary low bit error rate (Proakis 1995). It has been shown that a Turbo code can achieve performance within 1 dB of channel capacity (Berrou et al 1993). Random coding of long block lengths may also perform close to channel capacity, but this code is very hard to decode due to the lack of code structure. The performance of a Turbo code is partly due to the random interleaver used to give the Turbo code a “random” appearance.
However, one big advantage of a Turbo code is that there is enough code structure to decode it efficiently. There are two primary decoding strategies for Turbo codes. They are based on a maximum a posteriori probability algorithm and a soft output Viterbi algorithm. Regardless of which algorithm is implemented, the Turbo code decoder requires the use of two (same algorithm) component decoders that operate in an iterative manner.

Various research groups working with Turbo codes around the globe are Caltech Communications Group, California Institute of Technology, USA, Coding Research Group, University of Hawai at Monoa, USA, Coding Research Group, University of Notre Dame, USA, Communications Group, Politecnico di Torino, Italy, Communications Laboratory, Technion - Israel Institute of Technology, Israel, Communications Laboratory, Technische Universität Dresden, Germany, Communications Research Centre, Canada, Communications Research Group, University of York, United Kingdom, Communications Research in Signal Processing, Cornell University, USA, Communications Systems and Research Section, Jet Propulsion Laboratory, USA, Complex2Real, Turbo Code Tutorials, USA, Comtech AHA, USA, DataLab, University of California, Irvine, USA, David J.C. MacKay, Cambridge University, United Kingdom, DSP Derby, India, Efficient Channel Coding, USA, eritek, USA, Error Correcting Codes Home Page, Japan, Flarion Technologies, USA, iCODING Technology Incorporated, USA, Institute for Communications Engineering, Technische Universität München, Germany, Institute for Telecommunications Research, University of South Australia, International Symposium on Turbo Codes, ENST de Bretagne, France, Iterative Connections, Australia, Iterative Solutions, USA, Jakob Dahl Andersen, Technical University of Denmark, Lei Wei's, University of Central Florida, Orlando, USA, Mobile Multimedia Research, University of Southampton, United Kingdom. Patrick Robertson, Deutsches Zentrum für Luft-und Raumfahrt (DLR), Germany, Small World Communications,
Australia, Telecommunications Laboratory, University Erlangen-Nürnberg, Germany, Turbo Codes at West Virginia University, West Virginia University, USA, Turbo Codes in CCSR, University of Surrey, United Kingdom, Turbo Concept, France, VLSI Digital Signal Processing Laboratory, University of Minnesota, USA, What a wonderful Turbo world, Australia, Wireless Systems Laboratory, Georgia Institute of Technology, USA, Xenotran, USA, etc.

This Chapter gives a summary of Turbo codes and considers the related research efforts concerned to Turbo code. It also explains performance of modified SOVA and modified log MAP.

A thorough literature survey is listed below on Turbo codes, its modification and application by various research groups starting from 1948 to the recent times (2007).


2.3 SHANNON-HARTLEY CAPACITY THEOREM

Shannon (1948) showed that the system capacity $C$ of a channel perturbed by additive white Gaussian noise is function of the average received signal power $S$, the average noise power $N$, and the bandwidth $W$. The capacity can be stated as
\[ C = W \log_2 \left( 1 + \frac{S}{N} \right) \text{ Bits/second} \quad (2.1) \]

It is theoretically possible to transmit information over such a channel at any transmission rate \( R \), where \( R \leq C \), with an *arbitrarily small error probability* by using a sufficiently complicated coding scheme. For an information rate \( R > C \), it is not possible to find a code that can achieve an arbitrarily small error probability. Shannon’s work showed that the value of \( S, N \) and \( W \) set a limit on transmission rate, not on error probability. But noise power is proportional to bandwidth.

\[ N = N_0 W \quad (2.2) \]

where, \( N_0 \) is noise power spectral density.

Substituting equation (2.2) into equation (2.1) and rearranging terms yields

\[ \frac{C}{W} = \log_2 \left( 1 + \frac{S}{N_0 W} \right) \quad (2.3) \]

or the case where transmission bit rate is equal to channel capacity, \( R = C \),

\[ \frac{C}{W} = \log_2 \left[ 1 + \frac{E_b}{N_0} \left( \frac{C}{W} \right) \right] \quad (2.4) \]

There exist limiting values of \( \frac{E_b}{N_0} \) below which there can be no error free communication at any information rate. Using the
identity \lim_{x \to 0} (1 + x)^{\frac{1}{x}} = e$, it can be calculate the limiting value of $\frac{E_b}{N_0}$ as follows.

Let $x = \frac{E_b}{N_0} \left( \frac{C}{W} \right)$ then from equation (2.4) and simplifying, it yields

$$\frac{E_b}{N_0} = \frac{1}{\log_2 e} = 0.693 \quad (2.5)$$

or, in decibels $\frac{E_b}{N_0} = -1.6dB$.

This value of $\frac{E_b}{N_0}$ is called Shannon limit. Shannon’s work provided a theoretical proof for the existence of codes that could improve probability of bit error ($P_b$) performance or reduce $\frac{E_b}{N_0}$-required.

2.4 ADDITIVE WHITE GAUSSIAN NOISE CHANNEL

In communications, the Additive White Gaussian Noise (AWGN) channel model (Bernard Sklar 2005) is one in which the only impairment is the linear addition of wideband or white noise with a constant spectral density (expressed as watts per hertz of bandwidth) and a Gaussian distribution of amplitude. The model does not account for the phenomena of fading, frequency selectivity, interference, nonlinearity or dispersion. However, it produces simple, tractable mathematical models which are useful for gaining insight into the underlying behaviour of a system before these other phenomena are considered.
2.5 **FADING CHANNEL**

For most channels, where signal propagate in the atmosphere and near the ground, the free-space propagation model is inadequate to describe the channel behaviors and predict system performance. In wireless system, signal can travel from transmitter to receiver over multiple reflective paths. This phenomenon, called multipath fading, can cause fluctuations in the received signal’s amplitude, phase, and angle of arrival, giving rise to the terminology multipath fading. The received signal may thus be represented in the complex base band form. Raleigh fading (Bernard Sklar 2005) is a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices. It assumes that the magnitude of a signal that has passed through such a transmission medium (also called a communications channel) will vary randomly, or fade, according to a Raleigh distribution. Fading is due to multi path propagation. Fading phenomenon is multiplication of the signal waveform with a time-dependent coefficient which is often modelled as a random variable, making the received Signal to Noise Ratio (SNR) a random quantity.

2.6 **TURBO ENCODER**

A generic Turbo encoder (Barbulescu et al 1999) has been shown in Figure 1.2. The input sequence of the information bits is organized in blocks of length $N$. The first block of data will be encoded by the Recursive Systematic Convolutional codes RSC ENCODER1 block, which is recursive systematic encoder. The same block of information bits is interleaved by the interleaver, and encoded by RSC ENCODER2, which is also systematic recursive encoder. The code word is framed by concatenating output code words $X_k$, $Y_{1k}$, and $Y_{2k}$. 
Due to similarities with product codes (Orhan Gazi and Ali Ozgur Yilmaz 2006), it can be called the RSC ENCODER1 block as the encoder in the horizontal dimension and the RSC ENCODER2 block as the encoder in the vertical dimension. The interleaver block, rearranges the order of the information bits of input to the second encoder. The main purpose of the Interleaver (Sadjadpour et al 2000) is to increase the minimum distance of the Turbo code such that after correction in one dimension the remaining errors should become correctable error patterns in the second dimension. Ignoring for the moment the delay for each block, we assume both encoders output data simultaneously. This is rate 1/3 Turbo code, the output of the Turbo encoder being the triplet \( (X_k, Y_{1k}, \text{and } Y_{2k}) \). This triplet is then modulated for transmission across the communication channel, which is Additive White Gaussian Noise channel. Since the code is systematic, \( X_k \) is the input data at time \( k \), \( Y_{1k} \), and \( Y_{2k} \) are the two parity bits at time \( k \). The two encoders do not have to be identical. In figure 1.2 the two encoders are rate 1/2 systematic encoders in only one parity bit shown. The parity bits can be “punctured” as in Figure 1.2. The process of removing some of the parity bits after encoding in an error correction code is called puncturing. Where puncturing is implemented by a multiplexing switches in order to obtain higher coding rates, a rate 1/2 Turbo codes can be implemented by alternatively selecting the outputs of the two encoders in order to produce the following output sequence:

\[
\text{Output} = (X_1Y_{11}, X_2Y_{22}, X_3Y_{13}, X_4Y_{24}, \ldots) \tag{2.6}
\]

### 2.7 INTERLEAVER

The interleaver design (Khandani 1998) is a key factor, which determines the good performance of a Turbo code. Shannon (1948, 1949) showed that large block-length random codes achieve channel capacity. The
pseudo-random interleaver makes the code appear random. In this work the pseudo Random Interleaver has been used.

2.8 TURBO DECODER

Block diagram of Turbo decoder is shown in Figure 2.1. Turbo decoder (Gerard Battail 1998) performs iteratively. In iterative decoder (Hagenauer et al 1996, Szeto and Pasupathy 1999) structure, two component decoders are linked by interleaver in a structure similar to that of the encoder. Each decoder takes three inputs (Woodard and Hanzo 2000): 1) The systematically encoded channel output bits $X_k$; 2) the parity bits transmitted from the associated component encoder; $Y_{1k}$ or $Y_{2k}$ and 3) the information from the associated component decoder about the likely values of the bits concerned. This information from the other component decoder is referred to as apriori information. The component decoders have to exploit both the inputs from the channel and this apriori information.

![Block diagram of Turbo decoder](image)

**Figure 2.1 Block diagram of Turbo decoder**

The decoders must also provide what are known as soft outputs for the decoded bits. This means that as well as providing the decoded output bit
sequence, the component decoders must also give the associated probabilities for each bit it has been correctly decoded. Two suitable decoders are the so called SOVA proposed by Hagenauer and Hoeher (1989) and the MAP algorithm of Bahl (1974). The soft outputs from the component decoders are typically represented in terms of the so called Log Likelihood Ratios (LLRs), the magnitude of which gives the sign of the bit, and 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 Log likelihood Ratio $L(u_k)$ is the value of decoded bit $u_k$, it is given by

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

(2.7)

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 $u_k$ are taken to be +1 and -1, rather than 1 and 0, as this simplifies the derivations that follow.

The decoder operates iteratively (Bernard Sklar 1997), and in the first iteration the decoder1 takes channel output values only, and produces a soft output as its estimate of the data bits. The soft output from the decoder1 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 output 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 apriori information. This cycle is repeated and with every iteration the Bit
Error Rate of the decoded bits tends to fall. However, the error decrease as the number of iterations increases. Hence, for complexity reasons, usually only about eight iterations are used.

### 2.9 THE MAXIMUM APOSTERIORI (MAP) ALGORITHM

The MAP algorithm was proposed by Bahl (1974) in order to estimate the aposteriori probabilities of the states and the transitions of a Markov source observed in memory less noise, because RSC introduces Markov property into probability structure. Bahl (1974) showed how the algorithm could be used to decode both block and convolutional codes. When used to decode convolutional codes, the algorithm is optimal in terms of minimizing the decoded BER, unlike the Viterbi algorithm which minimizes the probability of an incorrect path through the Trellis being selected by the decoder.

Thus the Viterbi algorithm (Viterbi 1967) can be thought of as minimizing the number of groups of bits associated with the Trellis paths, rather than the actual number of bits, which are decoded incorrectly. Nevertheless as stated by Bahl et al (1974) in most applications the performance of log MAP and SOVA algorithms will be almost identical. However, the log MAP algorithm examines every possible path through the convolutional decoder Trellis and therefore initially seemed to be unfeasibly complex for application in most systems. Hence it was widely used before the discovery of Turbo codes. However the log MAP algorithm provides not only the estimated bit sequence, but also the probabilities for each bit that it has been decoded correctly. This is essential for the iterative decoding of Turbo codes proposed by Berrou et al (1993). Since then research efforts have been invested in reducing the complexity of the MAP algorithm of a reasonable level. This section describes the theory behind the MAP algorithm used for the soft output decoding of the component convolutional codes of the Turbo
codes. It is assumed that binary codes are used. The MAP algorithm gives, for each decoded bit $u_k$, the probability that this bit was +1 and -1, given the received symbol sequence $y$. This is equivalent to finding the a posteriori probability (APP) log likelihood ratio $L(u_k \mid y)$,

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

(2.8)

If the previous state $S_{k-1} = s$ and the present state $S_k = s$ are known in a Trellis then the input bit $u_k$ which caused the transition between these states will be known. This, along with Bayes’ rule and the fact that the transitions between the previous $S_{k-1}$ the present state $S_k$ in a Trellis are mutually exclusive allow to rewrite as,

$$L(u_k \mid y) = \ln \left( \frac{\sum_{u_k = 1} P(S_{k-1} = s \land S_k = s \land y)}{\sum_{u_k = -1} P(S_{k-1} = s \land S_k = s \land y)} \right)$$

(2.9)

where $(s,s) \Rightarrow u_k = +1$ is the set of transitions from the previous state $S_{k-1} = s$ to the present state $S_k = s$ that can occur if the input bit $u_k = +1$, and similarly for $(s,s) \Rightarrow u_k = -1$. For brevity it is written as $P(S_{k-1} = s \land S_k = s \land y)$ as $P(s \land s \land y)$.

Let consider the individual probabilities $P(s \land s \land y)$ from the numerator and denominator. The received sequence $y$ can be split up into three sections: the received codeword associated with the present transition $y_k$, the received sequence prior to the present transition $y_{j<k}$ and
the received sequence after the present transition \( y_{j>k} \). We can thus write for the individual probabilities \( P(s \land s \land y) \).

\[
P(s \land s \land y) = P(s \land s \land \sum_{j<k} y_j \land \sum_{j>k} y_j)
\]

(2.10)

Figure 2.2 which shows a section of a four state Trellis for a constrain length \( K=3 \) RSC code, and the split of the received channel sequence. In the Figure 2.2 solid lines represent transitions as a -1 input bit, and dashed lines represent transition resulting from a +1 input bit. The \( \alpha_{k-1}(s), \gamma_{k(x,s)} \) and \( \beta_k(s) \) symbols shown represent values calculated by the MAP algorithm. Using a derivation from Bayes’ rule that \( P(a \land b) = P(a \mid b)P(b) \) and assume that the channel is memory less, then the future received sequence \( y_{j>k} \) will depend only on the present state \( s \) and not on the previous state \( s' \) or the present and previous received channel sequences \( y_{<k} \) and \( y_{j<k} \).

It can be written as

\[
P(s \land s \land y) = \beta_k(s) \gamma_{k(x,s)} \alpha_{k-1}('s)
\]

(2.11)

where

\[
\alpha_{k-1}('s) = P(S_{k-1} = 's \land y_{j<k})
\]

(2.12)

is the probability that the Trellis is in state \('s\) at time \( k-1 \) and the received channel sequence up to point is \( y_{j<k} \), as visualized in Figure 2.3.

\[
\beta_k(s) = P(y_{j<k} \mid S_k = s)
\]

(2.13)

is the probability that given the Trellis is in state \( s \) at time \( k \) and the future received channel sequence will be \( y_{j>k} \), and lastly

\[
\gamma_{k(x,s)} = P(\{y_k \land S_k = s\} \mid S_{k-1} = 's)
\]

(2.14)
Figure 2.2 MAP decoder Trellis for K=3 RSC code

Equation (2.14) shows that the probability \(P(s_{k-1} = s)\) that the encoder Trellis took the transition from state \(S_{k-1} = s\) to state \(S_k = s\) and the received sequence is \(y\). It can be split into the product of three terms \(\alpha_{k-1}(s), \gamma_{k(s,s)}\), and \(\beta_k(s)\). The meaning of these three probability terms is shown in Figure 2.3, for the transition \(S_{k-1} = s\) to \(S_k = s\) shown by the bold line in Figure 2.2. From the equations (2.12) and (2.13) it can be written for the conditional log likelihood ratio of \(u_k\), given the received sequence \(y\).

\[
L(u_k | y) = \ln \left( \frac{\sum_{(s,s) \in k = 0,1, \ldots, l-1} \alpha_{k-1}(s), \gamma_{k(s,s)}, \beta_k(s)}{\sum_{u_k = 1}^{(s,s) \in k = 0,1, \ldots, l-1} \alpha_{k-1}(s), \gamma_{k(s,s)}, \beta_k(s)} \right) \tag{2.15}
\]

The MAP algorithm finds \(\alpha_k(s)\) and \(\beta_k(s)\) for all states \(s\) throughout the Trellis i.e., for \(k=0,1, \ldots, l-1\), and \(\gamma_{k(s,s)}\) for all possible transitions from state \(S_{k-1} = s\) to state \(S_k = s\), again for \(k=0,1, \ldots, l-1\). These values are then used to give the conditional LLRs \(L(u_k | y)\) that the MAP
decoder delivers. Let now describe how the values $\alpha_k(s), \beta_k(s)$ and $\gamma(s,s)$ can be calculated.

### 2.9.1 Forward Recursive Values ($\alpha_k(s)$) Calculation

Consider first $\alpha_k(s)$. From the definition of $\alpha_{k-1}(s)$, it can be written as

$$\alpha_k(s) = \sum_s P(s \wedge \tilde{y} k_s \wedge y_k)$$  \hspace{1cm} (2.16)

Split the probability $P(s \wedge \tilde{y} k_s \wedge y_k)$ into the sum of joint probabilities $P(s \wedge \tilde{y} k_s \wedge y_{k+1})$ over all possible previous states $s$. Using Bayes’ rule and the assumption that the channel is memory less again, one can proceed as follows:

$$\alpha_k(s) = \sum_s \alpha_{k-1}(s) \gamma_k(s,s)$$ \hspace{1cm} (2.17)

Thus once the $\gamma(s,s)$ values are known, the $\alpha_k(s)$ values can be calculated recursively. Assuming that the Trellis has initial state $s_0 = 0$, the initial conditions for this recursion are

$$\alpha_0(S_0 = 0) = 1$$

$$\alpha_0(S_0 = s) = 0 \text{ for all } s \neq 0.$$ \hspace{1cm} (2.18)

Figure 2.3 shows an example of how one $\alpha_k(s)$ value, for $s=0$, is calculated recursively using values $\alpha_{k-1}(s)$ and $\gamma(s,s)$ for the example $K=3$ RSC code.
Figure 2.3 Recursive values of $\alpha_k(0)$ and $\beta_k(0)$ calculation

Since considered a binary Trellis are considered, only two previous states, $S_{k-1} = 0$ and $S_{k-1} = 1$, have paths to the state $S_k = 0$. Therefore, the summation is over only two terms.

### 2.9.2 The Backward Recursive Values ($\beta_k(s)$) Calculation

The values of $\beta_k(s)$ can similarly be calculated recursively. Using a similar derivation to that it can be shown that,

$$\beta_k(s) = \sum_{s'} \beta_k(s') \gamma_{k(s,s')}$$  \hspace{1cm} (2.19)

Thus, once the values $\gamma_{k(s,s')}$ are known, a backward recursion can be used to calculate the values of $\beta_k(s)$ from the values of $\beta_k(s)$. Figure 2.3
shows the example of how the $\beta_k(0)$ value is calculated recursively using values $\beta_{k-1}(s)$ and $\gamma_{k-1}(0,s)$ for the example $K=3$ RSC code.

2.9.3 Branch Metric Values ($\gamma_k(s,s)$) Calculation

Now consider how the transition probability values $\gamma_k(s,s)$ can be calculated from the received channel sequence and any apriori information that is available. Using the definition of $\gamma_k(s,s)$ and the derivation from Bayes' rule we have

$$\gamma_k(s,s) = p(y_k / x_k).p(u_k)$$

(2.20)

where

$u_k$ input bit necessary to cause the transition from state $S_{k-1} = s$ to state $S_k = s'$;

$P(u_k)$ apriori probability of this bit.

$x_k$ is transmitted codeword associated with this transition. Hence, the transition probability $\gamma_k(s,s)$ is given by the product of the apriori probability of the input bit $u_k$ necessary for the transition, and probability that given the codeword $x_k$ associated with the transition was transmitted. It received the channel sequence $y_k$. The apriori probability $P(u_k)$ is derived in an iterative decoder from the output of the previous component decoder, and the conditional received sequence probability $p(y_k / x_k)$ is given, assuming a memory less Gaussian channel with Binary Phase Shift Keying (BPSK) modulation, as
\[ p(y_k / x_k) = \prod_{i=1}^{n} p(y_{ki} / x_i) \]  
(2.21)

\[ p(y_k / x_k) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_{ki} - a_{ki})^2}{2\sigma}} \]  
(2.17)

where \( x_{kl} \) and \( y_{kl} \) individual bits within the transmitted and received code words \( y_k \) and \( x_k \):

- \( n \) number of these bits in each codeword;
- \( E_b \) transmitted energy per bit;
- \( \sigma^2 \) noise variance;
- \( a \) fading amplitude.

### 2.10 ITERATIVE TURBO DECODING MATHEMATICAL PRELIMINARIES

It can is shown by Berrou et al (1993), for a systematic code such as a RSC code, the output from the MAP decoder, given by

\[
L(u_k | y) = \ln \left( \frac{\sum_{(s,s)\Rightarrow u_k, u_k = 1} \alpha_{k-1}(s) \gamma_k(s,s) \beta_k(s)}{\sum_{(s,s)\Rightarrow u_k, u_k = -1} \alpha_{k-1}(s) \gamma_k(s,s) \beta_k(s)} \right)  
\]
(2.23)

\[ = L(u) + L(y_k) + L_e(u) \]  
(2.24)

where

\[
L_e(u_k) = \ln \left( \frac{\sum_{(s,s)\Rightarrow u_k, u_k = 1} \alpha_{k-1}(s) \gamma_k(s,s) \beta_k(s)}{\sum_{(s,s)\Rightarrow u_k, u_k = -1} \alpha_{k-1}(s) \gamma_k(s,s) \beta_k(s)} \right)  
\]  
(2.25)
Here, \( L(u) \) is the apriori LLR and \( L_c \) is called the channel reliability measure and is given by \( L_c = \frac{4a}{2\sigma^2} \). \( y_{ks} \) is the received version of the transmitted systematic bit \( x_{ks} = u_k \) and \( \chi_{k(s,s)} = \exp (\frac{L_c}{2} \sum_{t=2}^{\infty} y_{ts}s_{st}) \) thus it can be seen that aposteriori LLR \( L(u_k \mid y) \) calculated with the MAP algorithm can be thought of comprising of three terms \( L(u), L_{y_k} \) and \( L_c(u) \). The apriori LLR term \( L(u) \) comes from \( P(u) \) in the expression for the branch transition probability \( \gamma_k(s,s) \). This probability should come from an independent source. In most cases it will not have independent or apriori knowledge of the likely value of the bit \( u_k \), and so the apriori LLR \( L(u) \) will be zero, corresponding to an apriori probability \( P(u_k) = 0.5 \). In case of an iterative Turbo decoder each component decoder can provide the other decoder with an estimate of LLR \( L(u) \).

The term \( L_{y_k} \) is the soft output of the channel for the systematic bit \( u_k \), which was directly transmitted across the channel and received as \( y_{ks} \). When the signal to nose ratio (SNR) is high, the channel reliability value \( L_c \) will be high and this systematic bit will have a large influence on the aposteriori LLR \( L(u_k \mid y) \). Conversely, when the channel is poor and \( L_c \) is low, the soft out of the channel for the received systematic bit \( y_{ks} \), will have less impact on the aposteriori LLR delivered by the MAP algorithm.

The term \( L_c(u) \) is derived, using the constraint imposed by the code used, from the apriori information sequence \( L(u_k) \) and the received channel information sequence \( y \), excluding the received systematic bit \( y_{ks} \) and apriori information \( L(u_k) \) for the bit \( u_k \). Hence, it is called the extrinsic LLR for the bit \( u_k \). The extrinsic information from a MAP decoder can be obtained by
subtracting the apriori information $L(u_k)$ and received systematic channel input $L_{y_k}$, from the soft output $L(u_k \mid y)$ of the decoder.

The terms apriori, aposteriori and extrinsic information are central concepts behind the iterative decoding of Turbo codes.

**apriori**: The apriori information about a bit is information known before decoding starts, from a source other than the received sequence or the code constraints. It also sometimes (Woodard and Hanzo 2000) referred to as intrinsic information.

**extrinsic**: The extrinsic information about a bit $u_k$ is the information provided by decoder based on the received sequence and on apriori information excluding the received systematic bit $y_k$, and the apriori information $L(u_k)$ for 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 apriori information surrounding the systematic bit $u_k$, and uses this information and the code constraints to provide information about the value of $u_k$.

**aposteriori**: The aposteriori information about a bit is the information that the decoder gives taking into all available sources of information about $u_k$. It is the aposteriori LLR, that is $L(u_k \mid y)$, that the MAP algorithm gives as its output.
The MAP algorithm is much more complex than the Viterbi algorithm and with hard decision outputs performs almost identically to it. It was realized that its complexity can be reduced without affecting its performance. Initially the Max-log-MAP algorithm was proposed by Koch and Baier (1990), Erfanian et al (1990). This technique simplified the MAP algorithm by transferring the recursion into log domain and reduces the complexity. Robertson et al in 1995 proposed the log MAP algorithm, which corrected the approximation used in the Max-log-MAP algorithm (Vogt and Finger 2000) and hence gave a performance identical to that of the MAP algorithm. In this thesis only log MAP algorithm has been considered.

2.11 THE SOFT OUTPUT VITERBI ALGORITHM

A variation of the Viterbi algorithm, referred to as the SOVA (Hagenauer et al 1989, Berrou et al 1997). SOVA has two modifications over the classical Viterbi algorithms which allow it to be used as a component decoder for Turbo codes. Firstly the path metrics used are modified to take account of apriori information when selecting the maximum likely (ML) path through the Trellis. Secondly, the algorithm is modified so that it provides a soft output in the form of the aperiodic LLR $L(u_k | y)$ for each decoded bit.

The first modification is easily accomplished. Consider the state sequence $s^s_k$ which gives the states along the surviving path at state $S_k = s$ at stage in the Trellis. The probability that this is the correct path through the Trellis is given by

$$p(s^s_k | y_{\neg j \leq k}) = \frac{p(s^s_k \wedge y_{\neg j \leq k})}{p(y_{\neg j \leq k})} \quad (2.26)$$

As the probability of the received sequence $y_{\neg j \leq k}$ for transitions up to and including the $k^{th}$ transition is constant for all paths $s_k$ through the
Trellis to stage, the probability that the path \( s^s_k \) is the correct one is proportional to \( p(s^s_k \wedge y_{j \leq k}) \). Therefore, the metric should be defined so that maximizing the metric will maximize \( p(s^s_k \wedge y_{j \leq k}) \). The metric should also be easily computable in a recursive manner as we go from \((k-1)\)th stage in the Trellis to the \( k \)th stage. If the path \( s^s_k \) at the \( k \)th stage has the path \( s^s_{k-1} \) for its first \((k-1)\) transitions then, assuming a memory less channel and using the definition of \( \gamma_k(s,s) \), we will have

\[
P(s^s_k \wedge y_{j \leq k}) = p(s^s_{k-1} \wedge y_{j \leq k-1}) \gamma_k(s,s) \tag{2.27}
\]

A suitable metric for the path \( s^s_k \) is therefore \( M(s^s_k) \), where,

\[
M(s^s_k) = M(s^s_{k-1}) + \ln(\gamma_k(s,s)) \tag{2.28}
\]

Omitting the constant term we then have

\[
M(s^s_k) = M(s^s_{k-1}) + \frac{1}{2} u_k L(u_k) + \frac{1}{2} \sum_{i=1}^{k} y_{ki} x_{li} \tag{2.29}
\]

Hence, the metric in the SOVA algorithm is updated as in the Viterbi algorithm, with the additional \( u_k L(u_k) \) term included so that the apriori information available is taken into account. Let us now discuss the second modification of the algorithm required, i.e., to give soft outputs. In a binary Trellis there will be two paths reaching state \( S_k = s \) at stage \( k \) in the Trellis. The modified Viterbi algorithm, which takes account of the apriori information \( u_k L(u_k) \), calculates the metric for both merging paths, and discards the path with the lower metric. If the two paths \( s^s_k \) and \( \hat{s}^s_k \)
reaching state $S_k = s$ have metrics $M(s_{k}^s)$ and $M(\hat{s}_{k}^s)$, and the path $s_{k}^s$ is selected as the survivor because its metric is higher, and then we can define the metric difference $\Delta'_k$ as

$$\Delta'_k = M(s_{k}^s) - M(\hat{s}_{k}^s) \geq 0$$ (2.30)

The probability that we have made the correct decision when we selected path $s_{k}^s$ as the survivor and discarded path $\hat{s}_{k}^s$, is then

$$P(\text{correct decision at } S_k = s)$$

$$= \frac{P(s_{k}^s)}{P(s_{k}^s) + P(\hat{s}_{k}^s)}$$ (2.31)

Correct decision taking into account the metric definition in equation (2.26) is

$$P(\text{correct decision at } S_k = s)$$

$$= \frac{e^{M(s_{k}^s)}}{e^{M(s_{k}^s)} + e^{M(\hat{s}_{k}^s)}} = \frac{e^{\Delta'_k}}{1 + e^{\Delta'_k}}$$ (2.32)

Correct decision at and the LLR that this is the correct decision is simply given by metric difference $\Delta'_k$. Figure 2.4 shows a simplified section of the Trellis of the with constraint length K=3 RSC code, with the metric differences $\Delta'_k$ marked at various points in the Trellis.
Figure 2.4  Simplified section of the Trellis for the K=4 RSC code with SOVA decoding algorithm

When it reach the end of the Trellis and have identified the ML path through the Trellis, need to find the log likelihood ratios giving the reliability of the bit decisions along the ML path. Observations of the Viterbi algorithm have shown that all the surviving paths at a stage no steps back from t (l) in the Trellis will normally have come from the same path at some point before l in the Trellis. This point is taken to be at most decoding delay (δ) transitions before l, where usually δ is set to be five times the constraint length of the convolution code. Therefore, the value of the bit $u_k$ associated with the transition from state $S_{k-1} = s_1$ to state $S_k = s$ on the ML path may have been different if, instead of the ML path, the Viterbi algorithm had selected one of the paths which merged with the ML path up to δ transitions later, i.e., up to the Trellis stage $K + \sigma$. By the arguments above if the algorithm had selected any of the paths which merged with the ML path after this point the value of $u_k$ would not be affected, because such paths will have diverged from the ML path after the transition from $S_{k-1} = s_1$ to $S_k = s$. Thus, when calculating the LLR of the bit $u_k$, the Soft Output Viterbi Algorithm (SOVA) must take account of the probability that the paths merging with the
ML path from stage $k$ to stage $k+\delta$ in the Trellis were incorrectly discarded. This is done by considering the values of the metric difference $\Delta_{i}^{s_{i}}$ for all states $s_{i}$ along the ML path from Trellis stage $i=k$ to $i=K+\delta$. It is shown by Hagenauer (1995). The LLR can be approximated by

$$L(u_{k} \mid y) \approx u_{k} \min_{i=k,...,k+\delta} \Delta_{i}^{s_{i}}$$

where $u_{k}$ is the value of the bit given by the ML path, and $u_{k}^{i}$ is the value of this bit for the path which merged with the ML path and was discarded at Trellis stage $i$.

Thus the minimization in (2.32) is carried out only for those paths merging with the ML path which would have given a different value for the bit $u_{k}$ if they had been selected as the survivor. The paths which merge with the ML path, but would have given the same value for $u_{k}$ as the ML path, obviously do not affect the reliability of the decision of $u_{k}$.

The SOVA algorithm is implemented as follows. For each state in the Trellis the metric $M(s_{k}^{a})$ is calculated for both of the two paths merging into the state using (2.29). The path with highest metric is selected as the survivor, for this state at this stage in the Trellis pointer to the previous state along the surviving path is stored. In order to allow the reliability of the decoded bit to be calculated the information used in (2.33) to give $L(u_{k} \mid y)$ is also stored. The difference $\Delta_{i}^{s_{i}}$ between the metric of the surviving and discarded paths is stored together with the binary vector containing $\delta+1$ bits, which indicates whether or not the discarded paths would have given the same
series of bits $u_l$ for $l = k$ back $l = k - \delta$ as the surviving path does. This series of bits is called the update sequence as noted by Hagenauer (1989); it is given by the result of modulo two additions between $\delta + 1$ decoded bits along the surviving and discarded paths. When SOVA has identified the ML path, the stored updates sequence and metric difference along this path are used (2.33) to calculate $L(u_k | y)$. The SOVA algorithm described is the least complex of all the soft in soft out decoders. It is shown by Robertson et al (1995) that the SOVA algorithm is about half as complex as the Max-log-MAP algorithm.

2.12 MODIFICATION INCORPORATED IN SOFT OUTPUT VITERBI ERROR CORRECTING ALGORITHM

Turbo codes use soft in soft out (SISO) decoders and employ decoding algorithms such as SOVA or log MAP to perform decoding operation in an iterative fashion. The soft output of the decoder consists of three terms and can be described mathematically as

$$L(\hat{d}) = L_{c}(x) + L(d) + L_{e}(\hat{d})$$

(2.34)

where

$L_{c}(x)$ is the channel reliability factor.

$L(d)$ is the apriori information of the data bit.

$L_{e}(\hat{d})$ is the extrinsic information gleaned from the decoder.

Several modifications have been done by various researchers (Jordan 1998) with respective to the channel reliability factor $L_{c}(x)$ with an aim to improve the performance of decoding algorithm. In the present work, SOVA decoder has been modified by compensating extrinsic information
with a common scaling factor which is constant and independent of the bit energy to noise ratio \((E_b/N_0)\).

### 2.12.1 Modified SOVA Algorithm

SOVA is modified by multiplying extrinsic information \(L_e(\hat{d})\) with the chosen scaling factor before it is being fed back to the input. The details of the modifications and corresponding output results are discussed below. The scaling factor must be chosen in such a way that it gives better improvement in the reliability of output from the decoder and decreases the number of iterations involved in attaining the pragmatic Shannon’s capacity limit of error performance. From the present work it is inferred that, better improvement of the results is obtained by introducing the scaling factor \((S_1)\) at the outer decoder as shown below in Figure 2.5.

![Figure 2.5 Turbo decoding with scaling factor](image)

#### 2.12.2 Discussion on Simulation Results

This section presents simulation results obtained from the modified SOVA along with SOVA and log MAP. Without loss of generality let the
interleaver length equal to 2048. The simulation parameters are as given below:

Channel : AWGN

Modulation : Binary Phase shift Keying (BPSK)

Component encoder : Recursive convolution codes (RSC)

Interleaver : 2048 bit random interleaver

Iteration : 8

Rate : 1/2

Figure 2.6 shows performance of modified SOVA and SOVA in AWGN Channel.

Figure 2.6 Performance of modified SOVA and SOVA in AWGN channel
The results of the various iterations on the performance of SOVA yielded that there are two scaling factor namely 0.9 and 1 which produces the best optimum result (Figure 2.6). From Figure 2.6, it is observed that the bit error rate (BER) improves for scaling factor at higher bit energy to noise power ($E_b/N_o$). SOVA performance deteriorates due to over estimation of extrinsic information.

Figure 2.7 depicts performance of modified SOVA and log MAP. It can be observed that modified SOVA gives better performance at $E_b/N_o > 3.5$ dB.

![Figure 2.7 Performance of modified SOVA and log MAP in AWGN channel](image)

Figure 2.8 compares the performance of SOVA and modified SOVA in fading channel.
The scaling factor (0.9, 1) improves the performance of SOVA decoder. The performance of modified SOVA is found to be better at $E_b/N_0 > 3.5$ dB.

2.12.3 Conclusion

1. SOVA, log MAP, modified SOVA has been simulated in both AWGN channel and fading channel.

2. From the results, it is observed that the BER improves in both the cases (AWGN and fading channel) beyond a threshold value of $E_b/N_0 = 3.5$ dB.

3. It is found that probability of decoding error decreases when an optimum scaling factor (0.9, 1) which is independent of $E_b/N_0$ is used.
4. The limitations of the present study are:
   a) The modification is made only in the outer code of SOVA
   b) The scaling factor is an empirical value and it is not optimized.
   c) The scaling factor is used for the outer code only and not for the inner code.
   d) The modifications pertaining to log MAP are not performed.

   These limitations are addressed and the results are discussed in the succeeding chapters.

2.13 MODIFICATION OF LOG MAP ALGORITHM USING SCALING FACTOR IN SIMULINK PLATFORM

This section explains modified log MAP and shows the improvement using the scaling factor in the extrinsic information at decoder. The Symbol Error Rate performance of Turbo codes is improved even in fading environment by using modified log MAP algorithm. In SIMULINK environment, evaluation of BER values for Convolutional and Turbo coding are done and the results are compared with Convolutional, log MAP and modified log MAP decoders. Further more for fixed values of iterations $E_s/N_o$ or $E_b/N_o$ are calculated for all the above decoding algorithms with respect to AWGN and Rayleigh fading channel.

2.13.1 SIMULINK

SIMULINK is a tool for modeling, simulating, and analyzing dynamic systems. SIMULINK provides an environment where modelling the
physical system and controller as a block diagram can be undertaken. SIMULINK uses the following communication block set.

- Source: Bernoulli Random Binary Generator
- Random interleaver and de interleaver
- Encoder: Convolution encoder
- Interleaver and deinterleaver: random interleaver
- Modulator and demodulator: Quadrature Phase shift Keying (QPSK)
- Channel: AWGN and Rayleigh fading
- Decoder: APP Decoder
- Output: Error Rate calculator

2.13.2 SIMULINK Models

The following blocks have been developed for simulation in SIMULINK environment.

- Convolutional decoder model with QPSK Modulation for AWGN and Rayleigh fading.
- Turbo decoder (log MAP) model with QPSK Modulation for AWGN and Rayleigh fading.
- Turbo decoder (Modified log MAP) model for AWGN and Rayleigh fading.
2.13.3 Simulation Profile

Without loss of generality let the interleaver length equal to 1024. The profile of the simulation, for present work, is detailed below:

Block size : 1024
Iterations : 6
Eb/No      : 1.5 dB
Rate       : 1/3

Figure 2.9 illustrates the various blocks created for convolutional encoder and decoder using SIMULINK both for AWGN and fading channel. At encoder part the blocks used are Bernoulli Random Binary Generator as a source, Convolution encoder, random interleaver, and QPSK modulator. At receiving side QPSK demodulator, convolutional decoder using Viterbi decoding algorithm, random de interleaver and error Rate calculator are used. The type channel is AWGN channel. In Figure 2.9, the dashed portion is used for convolutional code for Raleigh fading channel.

Figure 2.9 Convolutional decoder (Viterbi) model for AWGN channel and Raleigh fading channel
Figure 2.10 shows the blocks created for Turbo encoder and decoder using SIMULINK.

**Figure 2.10 Turbo decoder (APP) model for AWGN channel and Rayleigh fading channel**

At encoder part the blocks used are Bernoulli Random Binary Generator as a source, Convolution encoder, random interleaver, and QPSK modulator. At receiving side QPSK demodulator, Turbo decoder using MAP decoding algorithm and error Rate calculator are used. The type channel is AWGN channel. In Figure 2.10, the dashed portion is used for Rayleigh fading channel.

Figure 2.11 shows the blocks created for Turbo encoder with modified decoder using SIMULINK.

At encoder part the blocks used are Bernoulli Random Binary Generator as a binary source, Convolution encoders and QPSK modulator. At receiving side QPSK demodulator, Turbo decoder using log MAP as a decoding algorithm, and error rate calculator are used. Figure 2.11 the dashed portion is used for Rayleigh fading channel.
2.13.4 Simulation Results and Discussion

Figure 2.12 shows the results of convolution decoder. From Figure 2.12, it is observed that convolutional decoder produce better result in AWGN channel than fading channel in terms of symbol error rate. Convolutional decoder uses Viterbi algorithm as decoding algorithm.
Figure 2.13 shows the performance of Turbo decoders in AWGN and fading channels respectively. From Figure 2.13, it is observed that Turbo decoder using log MAP algorithm produces better performance than the convolutional decoder in terms of symbol error rate. It is also observed that Turbo decoder also produces better results in AWGN channel than the fading channel.

Figure 2.14 shows the effects of various scaling factor for Turbo codes in AWGN channel. It can be clearly observed Figure 2.14 that scaling factor of 0.5 gives better performance than other pre selected values. Hence this optimal value of 0.5 is taken as a scaling factor and is applied to outer decoder of Turbo decoder using log MAP decoding algorithm.

The results obtained from log MAP and modified log MAP in AWGN and Fading channel are shown in Figure 2.15. The modification is in terms of scaling the extrinsic information. The modified log MAP decoder produces better performance than the conventional log MAP decoder both in AWGN channel and fading channel respectively.
Figure 2.14 Results of scaling factors in modified MAP decoder for AWGN channel

Figure 2.15 Results of modified log MAP decoder in AWGN and fading channels

Figure 2.16 shows the comparisons of results obtained from log MAP and Modified log MAP decoders in AWGN channel.
Figure 2.16 Results of log MAP and modified log MAP decoders in AWGN channel

Figure 2.17 shows comparison of results obtained from log MAP and modified log MAP decoders in Raleigh fading Channel.

Figure 2.17 Results of log MAP and modified log MAP decoders in Raleigh fading channel

Figures 2.16 and 2.17 shows the comparison of results obtained from log MAP and modified log MAP in AWGN and fading channel.
respectively. It is observed that modified log MAP decoding algorithm produces one fold better performance in AWGN and fading channel.

Performance of Convolutional code, log MAP and modified log MAP are tabulated in Table 2.1 for two different values of $E_s/N_0 = 2$ dB and $E_s/N_0 = 5$ dB with various types of channels and decoders and the values of iterations, frame size and code rate are 6, 1024 and 1/3 respectively.

**Table 2.1 Results of convolutional, log MAP and modified log MAP**

<table>
<thead>
<tr>
<th>Decoder</th>
<th>Convolutional</th>
<th>log MAP</th>
<th>Modified MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>$E_s/N_0=2$ dB</td>
<td>$E_s/N_0=5$ dB</td>
<td>$E_s/N_0=2$ dB</td>
</tr>
<tr>
<td>AWGN</td>
<td>4.971x10$^{-2}$</td>
<td>4.922x10$^{-2}$</td>
<td>4.985x10$^{-4}$</td>
</tr>
<tr>
<td>Fading</td>
<td>5.001x10$^{-2}$</td>
<td>4.962x10$^{-2}$</td>
<td>5.005x10$^{-4}$</td>
</tr>
</tbody>
</table>

From the results complied in Table 2.1 it is found that the probability of decoding error is less in Turbo decoder (log MAP algorithm) compared to convolutional decoder with two signal strengths namely 2dB and 5dB. Thus the performance of 2dB and less than 2dB are important. At 2dB, log MAP and modified log MAP produce better performance than convolutional code. Comparing log MAP and modified log MAP, the later produces slightly better performance than the former.

**2.13.5 Conclusion**

1. Convolutional code, log MAP and Modified log MAP decoder have been simulated and analyzed in SIMULINK platform for both AWGN channel and Rayleigh Fading channel respectively.
2. It is observed that the log MAP produces better performance than convolutional code both in AWGN and fading channel. Thus Turbo decoding algorithms are superior to convolutional code, which is mandatory channel code for most of wireless communication systems.

3. From the above study, it is concluded that comparing log MAP and modified log MAP algorithms, the later produces less significant improvement in symbol error rate compared to the former.

4. The limitations of the present work are:
   
a) In the SIMULINK platform, there is no provision to measure $E_b/N_0$ and different types of code generator which generate the polynomials.
   
b) Interfacing the various blocks in SIMULINK is tedious.
   
c) The performance of modified log MAP needs optimization.

These limitations are addressed in the forthcoming chapters.