Journal of Advanced College of Engineering and Management, Vol.3, 2017

ITERATIVE DECODING OF TURBO CODES

Dhaneshwar Sah

Advanced College of Engineering and Management, T.U.

Email Address: dhaneshwar.sah@acem.edu.np

Abstract

This paper presents a Thesis which consists of a study of turbo codes as an error-control Code and the software implementation of two different decoders, namely the Maximum a Posteriori (MAP), and soft- Output Viterbi Algorithm (SOVA) decoders. Turbo codes were introduced in 1993 by berrouet at [2] and are perhaps the most exciting and potentially important development in coding theory in recent years. They achieve near- Shannon-Limit error correction performance with relatively simple component codes and large interleavers. They can be constructed by concatenating at least two component codes in a parallel fashion, separated by an interleaver. The convolutional codes can achieve very good results. In order of a concatenated scheme such as a turbo codes to work properly, the decoding algorithm must affect an exchange of soft information between component decoders. The concept behind turbo decoding is to pass soft information from the output of one decoder to the input of the succeeding one, and to iterate this process several times to produce better decisions. Turbo codes are still in the process of standardization but future applications will include mobile communication systems, deep space communications, telemetry and multimedia. Finally, we will compare these two algorithms which have less complexity and which can produce better performance.

Keywords: *SOVA, MAP, SISO, Turbo Codes, RSC, Channel Model, SNR, BER, LLR, VA*

1. Introduction

One of the aims of this paper will be to show that comprising and analysis for different decoding algorithm of turbo codes. There are various iterative decoding techniques. SISO: Soft information, or reliability, is crucial information type when turbo-like (iterative) processing of data is considered. With the advent of turbo codes in the area of information theory, a lot of attention is given to algorithm that can provide such soft reliability values while decoding the original information. There are two known soft-input soft-output. The thesis is proposed to work on these two SISO decoding Methods: Maximum a Posteriori (MAP) decoding algorithm and SOVA (Soft Output Viterbi Algorithm).

This algorithm is used to minimize the probability of word or sequence error.It works by rejecting the least likely path through the trellis at each node, and keeping the most likely one. The removal of unlikely paths leaves us, usually, with a single source path further back in the trellis. This path selection represents a ‘hard’ decision on the transmitted sequence. The Viterbi decoder estimates a maximum likelihood sequence.

2. Turbo Codes

Turbo codes were discovered in 1993 [1] before that, Shannon limit on code performance could only be approached with very long code word lengths. There was the problem of decoder complexity as well [9]. But we shall analyze in this chapter how decoder complexity can be reduced while implementing turbo codes.

2.1 Encoder

A parallel concatenated convolutional code is used for encoding turbo codes. In the Fig 1 [1] $d_i = (d_1, d_2, d_3 \ldots\ldots\ldots\ldots d_N)$ represent the binary input data sequence which is passed into the input of convolutional encoder [14][13] ENC$_1$, as denoted in the original paper. As a result, a coded bit stream x_{K1}^{P} is generated which is then interleaved, often in a pseudo-random pattern. The interleaved data sequence is passed to a second convolutional encoder ENC$_2$ and another coded bit stream x_{K2}^{P} is produced. Both of the code bit streams x_{K1}^{P} and x_{K2}^{P} are multiplexed (and possibly punctured) to form a systematic code bits x_k^5 and parity bits x_k^p

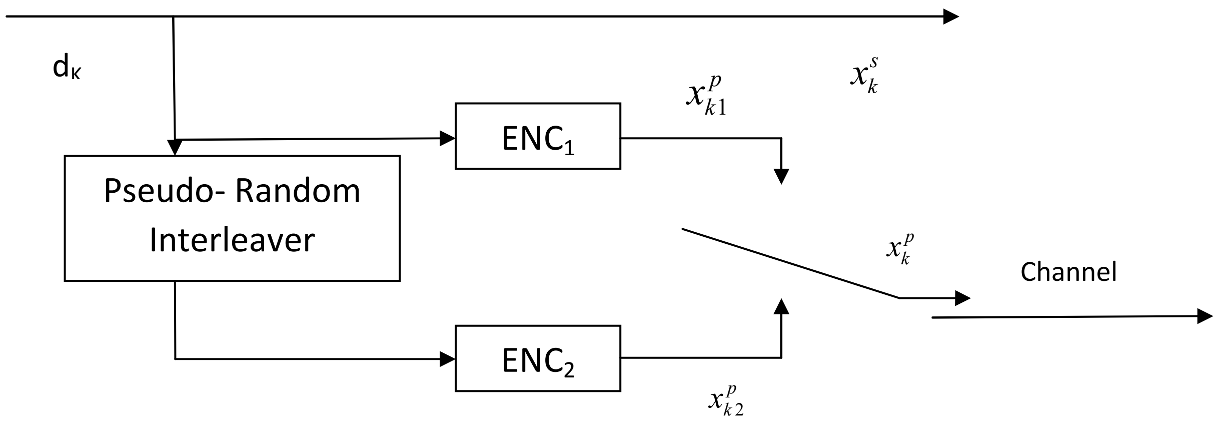


Fig 1 Turbo Encoder

2.2 Recursive Systematic Convolutional (RSC) codes

The convolution (RSC) coder ENC$_1$ and ENC$_2$ used turbo encoder are recursive systematic convolutional (RSC) codes. RSC codes are the convolutional codes that use feedback and the uncoded data bits are also present in the transmitted code bit sequence. Fig 2 shows a RSC encoder. The shown RSC encoder is of rate 1/2, with constraint length k = 3, and a generator polynomial G = {g1, g2} = {7, 5}, where g1 is the feedback connectivity and g2 is the output connectivity, in octal notation. An RSC component encoder has two output sequences: data: sequence $x_s^k = \left(x_1^s, x_2^s, \ldots\ldots, x_N^s\right)$ and parity sequence $x_k^p = \left(x_1^p, x_2^p, \ldots\ldots, x_N^p\right)$.

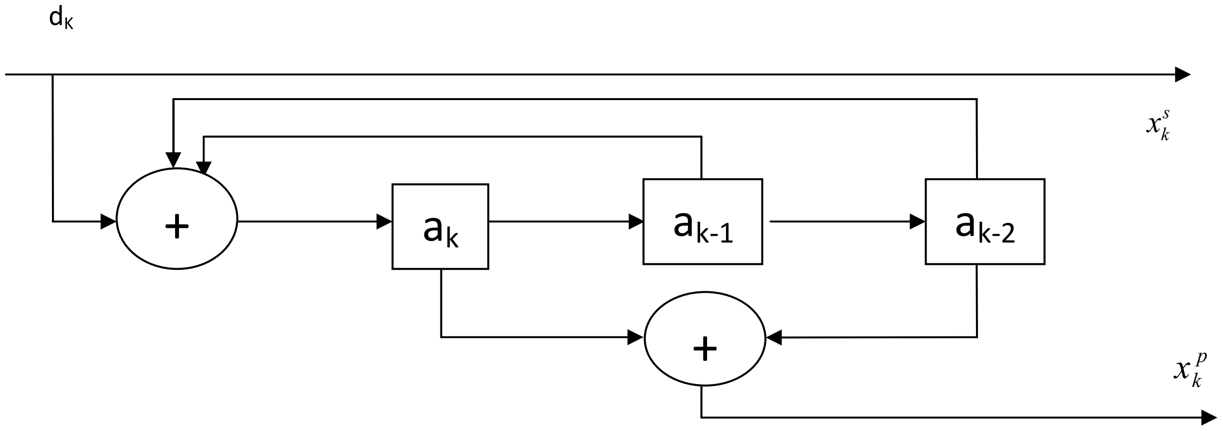


Fig 2 Recursive Systematic Convolution Code

The linear nature of turbo codes (at least, those using BPSK/QPSK modulation) means that the minimum Hamming distance of the code can be determined by comparing each Possible code wordwith the all–zeroes codeword .This process simplifies analysis of the code somewhat and the minimum hamming distance is then equal to the minimum codeweight (number of '1's) which occurs in any codeword.. This is the relationship between the codeweight and the number of codeword with that codeweight. An NSC code, however, will return to the all zeroes state after k-1 input zeroes, where K is the constraint length of the encoder. The infinite impulse response property of RSC codes is complemented in turbo codes by the interleaver between component encoders. The result is a composite codeword which will often have a high codeweight.. The pseudo-random nature of most turbo code interleavers tends to results in a mapping such that a few combinations of input bit positions which cause low codeweight sequences in one RSC component code are permuted into combinations of positions which generate low codeweight sequence in the second RSC code. The results in such a case are a low composite codeweight. such pseudo-random mapping often lead to turbo codes having a low minimum codeweight compared to say , NSC-based convolutional codes, resulting in a marked error floor at high SNR. . The distance spectrum of the code as a whole becomes significant in determining BER performance, and that the combination of RSC code and pseudo-random interleaving produces codeword with higher code weights most of the time. The low multiplicity of low codeweiught sequence associate with turbo codes sometimes referred to as spectral thinning, leads to their good BER performance at low SNR.

2.3 Interleaver

An interleaver does the work of re-arranging a sequence of symbols. One use of interleavers in communications is that of the symbols interleaver which is used after error control coding and signal mapping to ensure that fading bursts affecting blocks of symbols transmitted over the channel are broken up at the receiver by a de-interleaver, prior to decoding. Most error control codes work much better when errors in the received sequence are spread far apart. Another use is to place an interleaver between component codes in a serially concatenated code scheme for example, between a Reed Solomon outer code and a convolutional inner code. In both cases, the interleaver is typically implemented as a block interleaver.

The original data sequence is represented by the sequence of white squares, and the interleaved data sequence is represented by the grey squares. Berrou and Glavieux's original paper [1] featured results using a 256*256 interleaver. Turbo code BER performance improves with interleaver length-the so called interleaver gain- but the loading and unloading of the interleaver adds a considerable delay to the decoding process. This would make a 256*256 interleaver unsuitable for say real time speech applications which are delay sensitive.

2.6 Termination

Convolutional coding is a continuous process and code words do not have a fixed block length. The process can span the whole message rather than a small group of bits. But the turbo codes do have the fixed block length which is a determined by the length of the interleaver. Usual procedure is to append tail bits to each block of data bits entering one or other of the component encoders, to return it to the all zero state at the end of the trellis. This process is called termination

3. Turbo Decoding

The turbo decoder consists of two component decoders: DEC_1 to decode the sequence from ENC_1 and DEC_2 decode the sequence from ENC_2. Both DEC_2 are Maximum a posteriori (MAP) decoder. DEC_1 takes the received sequence of systematic values y_k^s and the received sequence of parity values y_k^s belonging to the first encoder ENC_1. The output of the decoder DEC_1 is sequence of soft estimates EXT_1 of the transmitted data bits dk. The EXT_1 is called the extrinsic data, in that it does not contain any information which was given to DEC_1 to DEC_2. This information is interleaved, and then passed to the second decoder DEC_2. The encoder is identical to the used in the encoder. DEC_2 takes as its input the interleaved systematic received values y_k^s and the sequence of received parity values y_{k2}^s from the second encoder ENC$_2$ along with the interleaved from of the extrinsic information EXT$_1$, provided by the first decoder. The second decoder DEC$_2$ produces as its outputs a set of values which when de-interleaved using the inverse form of interleaver, constitutes soft estimates EXT$_2$ of the transmitted data sequence d$_{k2}$. This extrinsic data, formed without the aid of parity bits from the first code, is feedback to DEC$_1$. This procedure is repeated in an iterative manner. The iterative decoding process adds greatly to the BER performance of turbo codes for example, Berrou and Glavieux achieved $BER = 10^{-5}\, at \frac{E_b}{N_o}$ within 0.7 dB of the Shannon limit, using a rate 1/2 turbo code and 18 decoding iterations. However, after several iterations, the two decoder's estimates of d$_k$ will tend to coverage. At this point, DEC$_2$ outputs a value $\Lambda(d_k)$ a log – likelihood representation of the estimate of d$_k$.This log- likelihood value takes into account the probability of a transmitted 0 or 1 based on the systematic information and parity information from both component codes. More negative values of $\Lambda(d_k)$ represent a strong likelihood that the transmitted bit was a 0 and more positive values represent a strong likelihood that it was transmitted bit. $\Lambda(d_k)$ is de- interleaved so that its sequence coincides with that of the systematic and first parity streams. Then a simple threshold operation is performed on the result, to produce hard decision estimates, dk for the transmitted bits. The decoding estimates EXT$_1$ and EXT$_2$, do not necessarily converge to a correct bit decision. If a set of corrupted code bits form a pair of error sequences that neither of the decoders is able to correct, then EXT$_1$ and EXT$_2$ may either diverge, or converge to incorrect soft value.

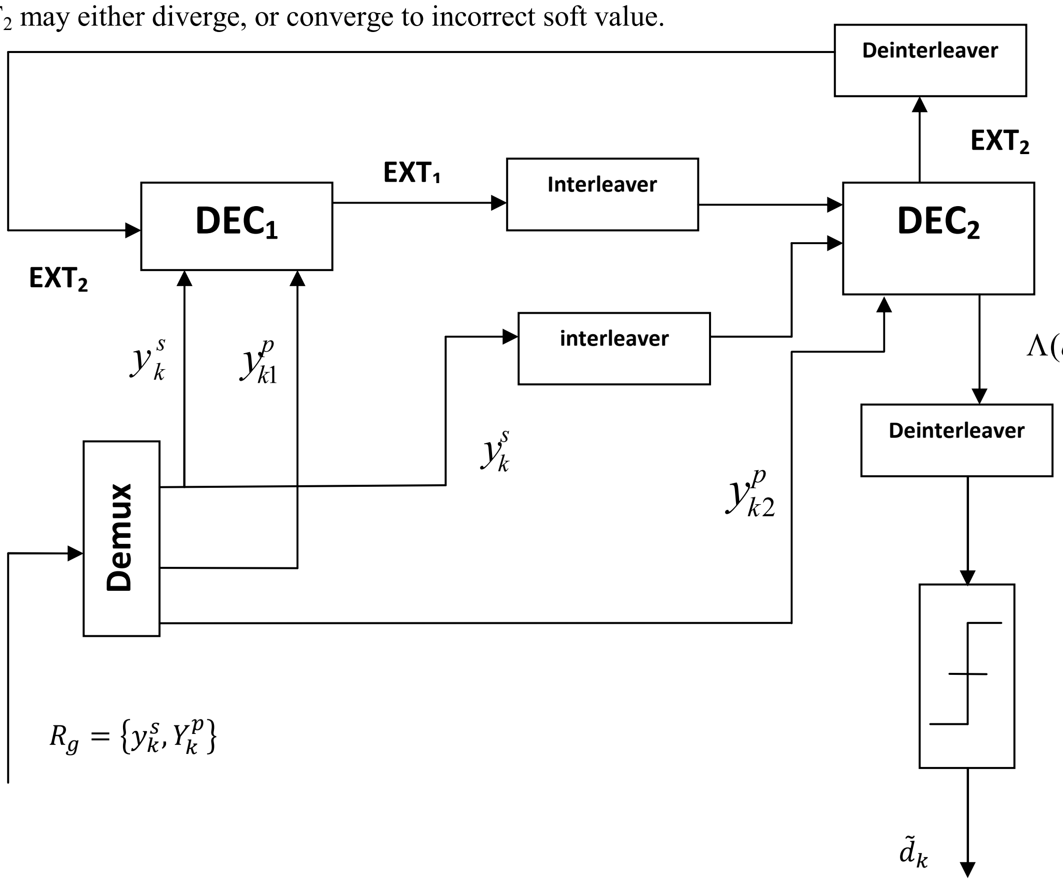


Fig 3 Turbo Decoder Structure

3.1 The MAP Algorithm

The decoding algorithm implemented in DEC_1 and DEC_2 for iterative decoding need to be analyzed. The first one under discussion is the Maximum A posteriori (MAP) algorithm presented in Berrouet. al's original paper [1]. We describe here a derivation of the MAP decoding algorithm for systematic convolutional assuming an AWGN channel model, as presented by pietrobo [16].

4. Channel models

In additive white Gaussian noise channel, the received signal is the sum of the transmitted (attenuated in some way) and noise with a Gaussian probability density function (pdf) given by:

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{n^2}{2\sigma^2}\right\} \tag{3.26}$$

The effect of AWGN is to hinder a detector in the estimate of the transmitted signal based on a possibly very weak received signal. Because AWGN affects all electronic circuitry, it almost always added to a simulation channel model.

4.1 Performance of Turbo Codes for Log-MAP

A simulation program has been written and it gives the performance for and length data. However, it takes much time depending on data length, punctured or unpunctured pattern, the E_b/N_0 ratios provided and the channel models used.An analysis was done on an AWGN channel for rate 1/2 , length 1024-bit and log MAP Turbo decoder.

We observe the gain achieved by the turbo code relative to convolutional code of relative complexity. We can clearly see the iterative power of turbo code.

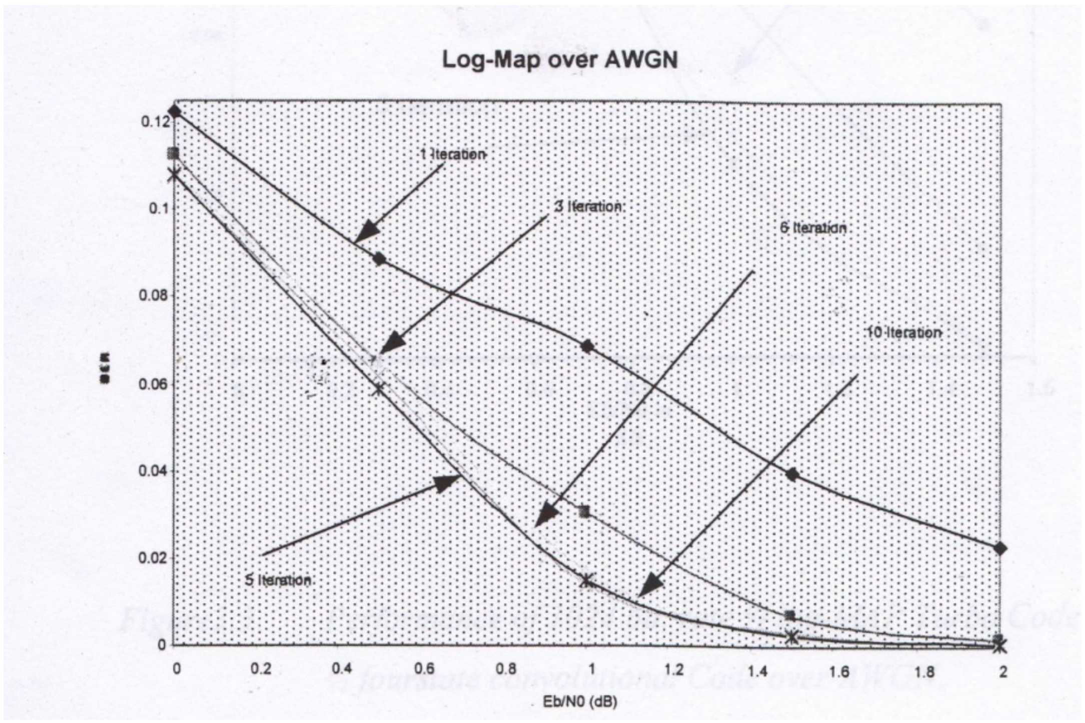


Fig 4 Performance of 400 bit, rate 1/2, log-MAP Turbo Code versus ratefour State Convolutional Code over AWGN

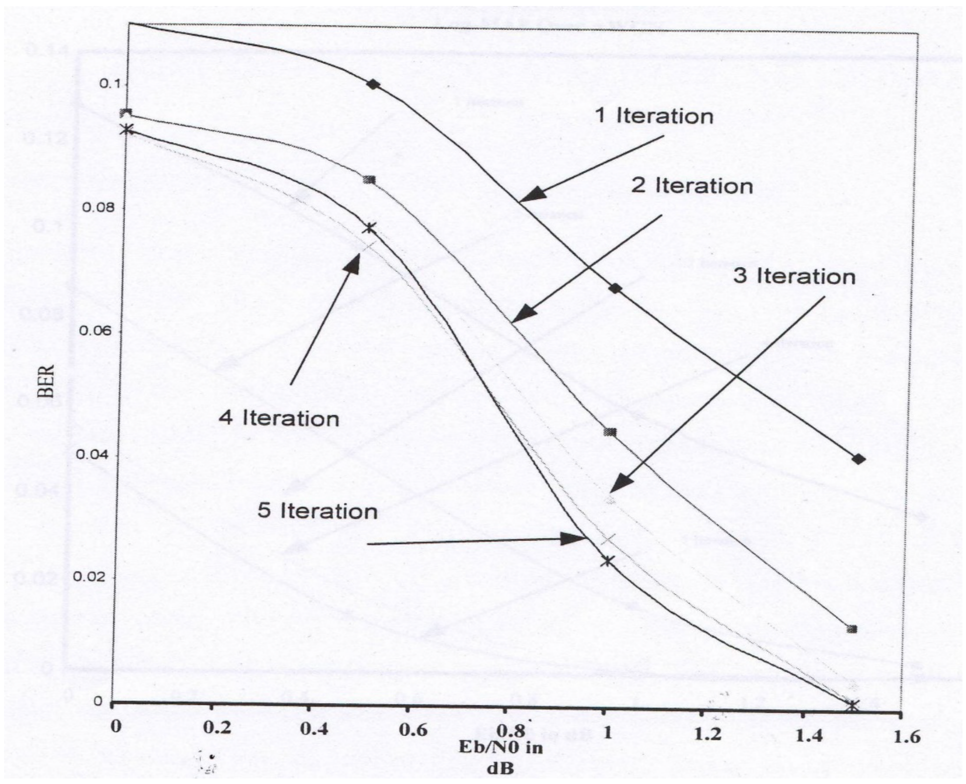


Fig 5 Performance of 1024 bit, rate 1/2 log-MAP Turbo Code versus rate 1/2 fourstate convolutional Code over AWGN.

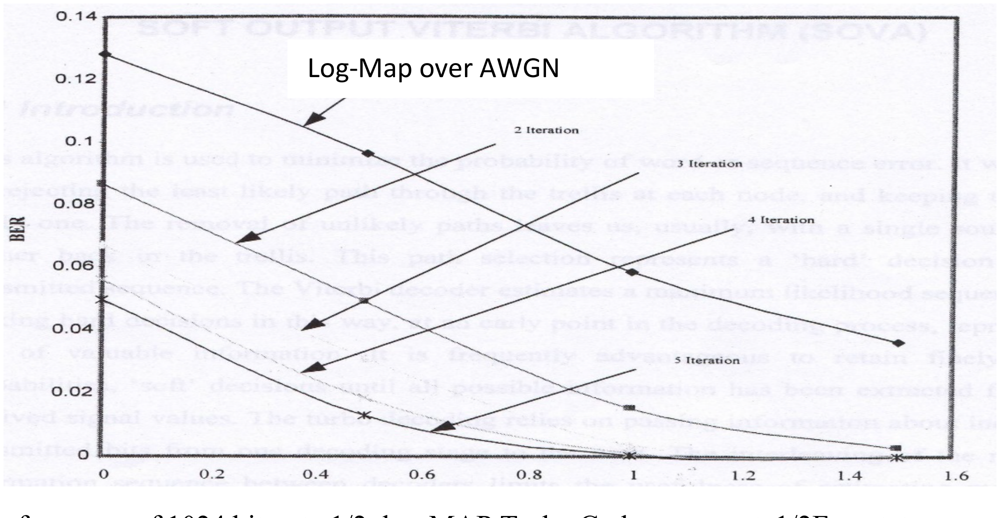


Fig 6 Performance of 1024 bit, rate 1/2, log-MAP Turbo Code versus rate1/2Fourstate convolutional Code over AWGN.

4.2 Soft Output Virtebri Algorithm (SOVA)

This algorithm is used to minimize the probability or word or sequence error. It will work by rejecting the least likely path through the trellis at each node, and keeping the most likely one. The removal of unlikely paths leaves us, usually, with a single source path further back in the trellis. This path selection represents a 'hard' decision on the transmitted sequence. The Viterbi decoder estimates a maximum likelihood sequence.

It is described here a derivation of MAP decoding algorithm for systematic convolutional codes assuming an AWGN channel model, as presented by pietrobon [16]. We start with the ratio of the APPs, known as the likelihood Ratio $\Lambda(\hat{d}_k)$, or its logarithm, called the LLR, as shown below.

$$\Lambda(\hat{d}_k) = \frac{\sum_m \lambda_k^{i,m}}{\sum_m \lambda_k^{0,m}} \tag{4.1}$$

$$L(\hat{d}_k) = \log\left[\frac{\sum_m \lambda_k^{i,m}}{\sum_m \lambda_k^{0,m}}\right] \tag{4.2}$$

Where $\lambda_k^{i,m}$ the joint probability that data $d_k = I$ and state $S_k = m$ conditioned on the received binary sequence R_1^N observed from time k = 1 through some time N, is described as

$$\lambda_k^{i,m} = p(d_k = i, S_k = m \mid R_1^N) \tag{4.3}$$

R_1^N Represents a corrupted code bit sequence after it has been transmitted through the channel, demodulated, and presented to the decode in soft decision form. In effect, the MAP algorithm requires that the output sequence from the demodulator be presented to the decoder as a block of N bits at a time. Let R_1^N be written as follows,

$$R_1^N = \left\{R_1^{k-1}, R_k, R_{k+1}^N\right\} \tag{4.4}$$

To facilitate the use of Bayes' theorem, Equation (4.2) is partitioned using the letters A, B, C, D and Equation (4.3). Equation (4.2) can be written in this form:

$$\lambda_k^{i,m} = p(d_k = i, S_k = m \mid R_1^{k-1}, R_k, R_{k-1}^N) \tag{4.5}$$

$$\text{A} = p(d_k = i, S_k = m \mid)$$

$$\text{B} = R_1^{k-1}$$

$$C = R_k \quad \text{and } D = R_{k+1}^N$$

From Bayes theorem

$$p(A \mid B, C, D) = \frac{p(A, B, C, D)}{p(B, C, D)} = \frac{p(B \mid A, C, D) P(A, C, D)}{p(B, C, D)}$$

$$= \frac{P(B \mid A, C, D) P(D \mid A, C) P(A, C)}{P(B, C, D)} \tag{4.6}$$

Hence, application of this rule to Equation (4.5) yields

$$y_k^{i,m} = p(R_1^{k-1} \mid d_k = i, S_k = m, R_k^N) p(R_{k+1}^N \mid d_k = i, S_k = m, R_k)$$

$$\text{x}\, p(d_k = i, S_k = m, R_k) \div p(R_1^N) \tag{4.7}$$

Where $R_K^N = \{R_k, R_{K+1}^N\}$ Equation (4.7) can be expressed in a way that gives greater meaning to the probability terms contributing to $\lambda_k^{i,m}$. The three numerator factors on the right side of Equation (4.7) will be defined and developed as the forward state metric, the reverse state metric, and the branch metric.

4.3 State metrics and the Branch Metric

We define the first numerator factor on the right side of Equation (4.7) as the forward state metric at time K and state m, and denote it as a_k^m Thus for i = 1,0.

$$p(R_1^{k-1}|d_k = i, s_k = m, R_k^N) = p(R_1^{k-1}|S_k = m)\underline{\Delta} a_{\underline{k}}^m \tag{4.8}$$

Notice that d_k = i and R_k^N are designated as irrelevant, since the assumption the S_k = m implies that events before time k are not influenced by observation after time K. In other words, the past is not affected by the future, hence $P(R_1^{k-1})$ is independent of the fact that d_k = i and sequence R_k^N . However, since the encoder has memory, the encoder state S_k=m is based on the pair, so this term is relevant and must be left in the expression. The form of Equation (4.8) is intuitively satisfying, since it presents the forward state metric a_k^m at time k as being a probability of the past sequence; that is, dependent only on the current state induced by this sequence, and nothing more. This should be familiar from the second numerator factor on the right side of Equation (4.7) represents a reverse state metric, β_k^m at time k and state m, described below

$$P(R_{k+1}^N|d_k = i, S_k = m, R_k) = \mathrm{P}(\mathrm{R}_{k+1}^{\mathrm{N}}|\mathrm{S}_{k+1} = \textstyle\int(\mathrm{i,m}))\underline{\Delta}\beta_{k+1}^{\int(\mathrm{i,m})} \tag{4.9}$$

Where $\int(i,m)$ is the next state, given an input I and state m, and $\beta_{k+1}^{\int(i,m)}$ is the inverse state metric at time k+1 and state $\int(i,m)$. The form of Equation (4.9) is intuitively satisfying since it presents the reverse state metric, $\beta_{k+1}^{\int(i,m)}$ at future time k+1, as being a probability of the future sequence, which depends on the state (at future time k). This should be familiar because it creates the basic definition of a finite-state machine [17]. We define the third numerator factor on the right side of Equation (4.7) as the branch metric at time k and state m. denoted $\delta_k^{i,m}$ Thus we write

$$P(d_k = i, S_k = m, R_k)\underline{\Delta}\delta_k^{i,m} \tag{4.10}$$

Substituting Equation (4.8) through (4.10) into Equation (4.7) yields the following more compact expression for the joint probability:

$$\lambda_k^{i,m} = \frac{a_k^m \delta_k^{i,m} \beta_{l+1}^{\int(i,m)}}{p(R_1^N)} \tag{4.11}$$

Equation (4.11) can be used to express Equation (4.1) and (4.2) as follows:

$$\Lambda\left(\hat{d}_k\right) = \frac{\sum_m a_k^m \delta_k^{1,m} \beta_{k+1}^{\int(1,m)}}{\sum_m a_k^m \delta_k^{0,m} \beta_{k+1}^{\int(0,m)}} \tag{4.12}$$

$$L\left(\hat{d}_k\right) = \log\left[\frac{\sum_m a_k^m \delta_k^{1,m} \beta_{k+1}^{\int(1,m)}}{\sum a_k^m \delta_k^{0,m} \beta_{k+1}^{\int(1,m)}}\right] \tag{4.13}$$

Where $\Lambda(\hat{d}_k)$ is the likelihood ratio of the kth data bit and $L(\hat{d}_k)$, the logarithm of $\Lambda(\hat{d}_k)$,is the LLR of the kth data bit, where the logarithm is generally taken to the base e.

4.4 Calculating the Forward State Metric

Starting from Equation (4.8), a_k^m can be expressed as the summation of all possible transition probabilities from time k-1, as follows

$$a_k^m = \sum_m \sum_{j=0}^{1} p(d_{k-1} = j, S_{k-1} = m', R_1^{k-1} | S_k = m) \tag{4.14}$$

We can rewrite $R_1^{k-1} as \left\{R_1^{k-2}, R_{k-1}\right\},$ and from Bayes theorem,

$$a_k^m = \sum_m \sum_{j=0}^{1} p(R_1^{k-2} | S_k = m, d_{k-1} = j, S_{k-1} = m', R_{k-1})$$

$$\text{x}\, p(d_{k-1} = j, S_{k-1} = m', R_{k-1} | S_k = m) \tag{4.15a}$$

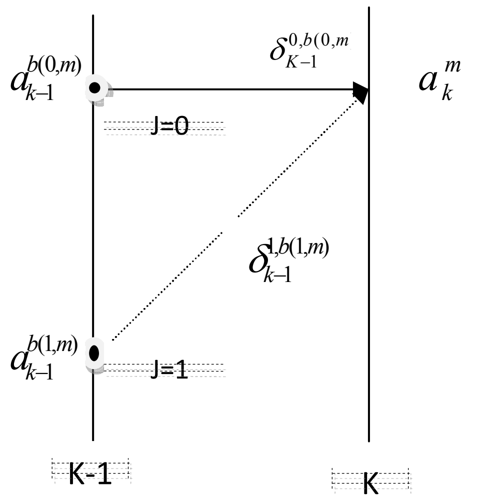
$$= \sum_{j=0}^{1} p\left((R_1^{k-2} | S_{k-1} = b(j,m)\right) p(d_{k-1} = j, S_{k-1} = b(j,m), R_{k-1}) \tag{4.15b}$$

Where b(j,m) is the state going backward in time from state m, via the previous branch corresponding to input j. Equation (4.15b) can replace Equation (4.15a) since knowledge about the state m' and the input j, at time k-1 , completely, defines the path resulting in 42.

State S_k = m. Using Equation (4.8) and Equation (4.10) to simplify the notation of Equation (4.15) yields the following:

$$a_k^m = \sum_{j=0}^{1} a_{k-1}^{b(j,m)}\, \delta_{k-1}^{j,b(j,m)} \tag{4.16}$$

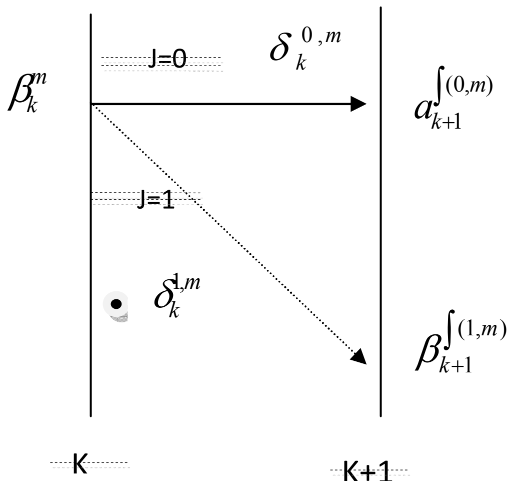
Equation (4.16) indicates that a new forward state metric at time k and state m is obtained by summing two weighted state metrics from time k-1. The weighting consists of the branch metrics associated with the transitions corresponding to data bit 0 and 1. Figure 3.1 illustrates the use of two different types of notations for the parameter alpha. We use $a_{k-1}^{b(j,m)}$ for the forward state metric at time k-1, when there two possible underlying states (depending upon whether j=0 or 1). And we use a_k^m for the forward state metric at time k, when the two possible transitions from the previous time terminate on the same state m time k.



(a) Forward state metric

$$a_k^m = a_{k-1}^{b(0,m)} \delta_{k-1}^{0,b(0,m)} + a_{k-1}^{b(1,m)} \delta_{k-1}^{1,b(1,m)}$$

Where b (j,m) is the state going backward in time corresponding to an input j.



(b) Reverse state metric

$$\beta_k^m = \beta_{k+1}^{\int(0,m)} \delta_k^{0,m} + \beta_{k+1}^{\int(1,m)} \delta_k^{1,m}$$

where $\int(i,m)$ is the next state going an input j and state m.

4.5 Branch Metric:

$$\delta_k^{i,m} = \pi_k^i \exp\left(x_k u_k^i + y_k v_k^{i,m}\right)$$

Graphical representation for calculating a_k^m and β_k^m [2].

4.6 Calculating the Reverse State Metric

Starting from Equation (4.9) where $\beta_{k+1}^{\int(i,m)} = \left\lfloor R_{k+1}^N \middle| S_{k+1} = \int(i,m) \right\rfloor,$ we show β_k^m as follows:

$$\beta_k^m = p(R_k^N | S_k = m) = p(R_k, R_{k+1}^N | S_k = m) \tag{4.17}$$

We can express β_k^m as the summation of all possible transition probabilities to time k+1, as follows:

$$\beta_k^m = \sum_{m'} \sum_{j=0}^{1} p(d_k = j, S_{k+1} = m', R_k, R_{k+1}^N | S_k = m) \tag{4.18}$$

Using Bayes' theorem,

$$\beta_k^m = \sum_{m'} \sum_{j=0}^{1} p(R_{k+1}^N | S_k = m, d_k = j, S_{k+1} = m', R_k) \text{ x } p(d_k = j, S_{k+1} = m', R_k | S_k = m) \tag{4.19}$$

S$_k$ =m and d$_k$=j in the first term on the right side of Equation (4.19) completely defines the path resulting in $S_{K+1} = \int(j,m))$the next state given an input j and state m. thus, these conditions allow replacing $S_{k+1} = m'$ with $S_k = m$ in the second term of Equation (4.19), yields the following:

$$\beta_k^m = \sum_{j=0}^{1} p(R_{k+1}^N \Big| S_{k+1}) = \int(j,m) p(d_k = j, S_k = m, R_k) = \sum_{j=0}^{1} \delta_k^{j,m} \beta_{k+1}^{\int(j,m)} \tag{4.20}$$

Equation (4.20) indicates that a new reverse state metric at time k and state m is obtained by summing two weighted state metrics from time k+1The weighting consists of the branch metrics associated with the transitions corresponding to data bits 0 and 1 Figure 3.1.b illustrates the use of two different types of notation for the parameter beta. We use $\beta_{k+1}^{\int(i,m)}$ for the reverse state metric at time k+1 when there are two possible underlying states (depending on whether j=0 or 1). And we use β_k^m for the reverse state metric at time k, where the two possible transitions arriving at time k+1 stem from the same state m at time k. Fig 3 represents a graphical illustration for calculating the forward and reverse state metrics. Implementing the MAP decoding algorithm has some similarities to implementing the Viterbi decoding algorithm [3]. In the Viterbi algorithm, we add branch metrics to state metrics. Then we compare and select the minimum distance (maximum likelihood) in order to form the next state metric. The process is called add-compare-select (ACS). In the MAP algorithm, we multiply (add, in the logarithmic domain) state metrics by branch metrics. Then, instead of comparing them, we sum them to form the next forward (or reverse) state metric, as seen in figure 1. The differences should make intuitive sense. With the Viterbi algorithm, the most likely sequence is being the best path. With the MAP algorithm, soft number (likelihood or Log-likelihood) is being sought; hence the process uses all the metrics from all the possible transitions within a time interval, in order to come up with the best overall statistic regarding the data bit associated with that time interval.

4.7 Calculating the Branch Metric

We start with Equation (4.10), which is rewritten below:

$$\delta_k^{i,m} = p(d_k = i, S_k = m, R_k) = p(R_k | d_k = i, S_k = m) p(S_k = m | d_k = i) p(d_k = i) \qquad (4.21)$$

Where $R_k = x_k, y_k, x_k$ is the received data bit, and y_k is the corresponding noisy received parity bit. Since the noise affecting the data and the parity are independent, the current state is independent of the current input, and can therefore be any one of the 2^0states, where V is the number of memory elements in the convolution code system. That is, the constraint length, k, of the code is equal to V+1. Hence,

$$p(S_k = m | d_k = i) = \frac{1}{2^v}$$

and

$$\delta_k^m = p(x_k | d_k = i, S_k = m) p(y_k | d_k = i, S_k = m) \frac{\pi^i}{2^v} \qquad (4.22)$$

Where π_k^i is defined as p(d_k=i), the a priori probability of d_k.

The probability p(X_k=x_k) of a random variable. X_k taking on the value x_k is related to the probability density function (pdf) p_x,(x_k) as follows [17].

P(X_k=x_k)=px_k,(x_k)d_k (4.23)

For notational convenience, the random variableX_k, which takes on value x_k, is often termed "the random variable x_k" ,which represents the meanings of X_k and Y_k in Equation (4.22). Thus, for an AWGN channel where the noise has zero mean and variance σ^2 , we use Equation (4.23) in order to replace the probability terms in Equation(4.22) with their pdf equivalents, and we write

$$\delta_k^{i,m} = \frac{\pi_k^i}{2^0\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left\{\frac{x_k - u_k^i}{\sigma}\right\}^2\right] dx_k \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left\{\frac{y_k - v_k^{i,m}}{}\right\}^2\right] dy_k \tag{4.24}$$

Where u_k and v_k represent the transmitted data bits and parity bits, respectively (in bipolar form), dx_k and dy_k are the differentials of $x_{k,}y_k$ and get absorbed into the constant A_k below. Note that the parameter u_k^i represents data that has no dependence on the state m. However, the parameter $v_k^{i,m}$ represents data, which does depend on the state m, since the code has memory.

$$\delta_k^{i,m} = A_k \pi_k^i \exp\left[\frac{1}{\sigma^2}\left(x_k u_k^i + y_k v_k^{i,m}\right)\right] \tag{4.25}$$

If we substitute Equation (4.24) into Equation (1), we obtain

$$\Lambda(\hat{d}_k) = \pi_k \exp\left\{\frac{2x_k}{\sigma^2}\right\} \frac{\sum_m a_k^m \exp\left\{\frac{y_k v_k^{1,m}}{\sigma^2}\right\} \beta_{k+1}^{\int(1,m)}}{\sum_m a_k^m \exp\left\{\frac{y_k v_k^{0,m}}{\sigma^2}\right\} \beta_{k+1}^{\int(0,m)}} \tag{4.26a}$$

$$\pi_k \exp\left(\frac{2x_k}{\sigma^2}\right) \pi_k^e \tag{4.26b}$$

and

$$L\left(\hat{d}_k\right) = L\left(d_k\right) + L_c\left(x_k\right) + L_e\left(\hat{d}_k\right) \tag{4.26c}$$

Where, $\pi_k = \pi_k^1 / \pi_k^0$ is the input priori probability ratio (prior likelihood) and π_k^e is the output extrinsic likelihood each at time k. In equation (4.26b), one can think of π_k^e as a correction term (due to the coding) that changes the input prior knowledge about a data bit. In a turbo code, such correction terms are passed from one decoder to the next, in order to improve the likelihood ratio for each data bit, and thus minimize the probability of decoding error. Thus the decoding process entails the use of Equation (4.26b) to Compute $\Lambda(\hat{d}_k)$ for several iterations. The extrinsic likelihood π_k^e, resulting from a particular iteration replaces the a priori likelihood ratio π_{k+1} for the next iteration. Taking the logarithm of $l(\hat{d}_k)$ in Equation (4.26b) yields Equation (4.26c) which shows that the final soft number $L(\hat{d}_k)$is made up of three LLR terms: the priori LLR, the channel measurement LLR, and the extrinsic LLR [17]. The MAP algorithm can be implemented in terms of likelihood ratio $L(\hat{d}_k)$ as shown in Equation (4.26a) or (4.26b). However, implementation using likelihood ratios is very complex because of the multiplication operations that are required. By operating the MAP algorithm in the logarithmic domain [16, 18] as described by the LLR in Equation (4.26b) or (4.26c), the complexity can b greatly reduced by eliminating the multiplication operations.

4.8 Performance of Turbo Codes for SOVA

A simulation program has been written and it gives the performance for any length data. However, it takes much time depending on data length, punctured or unpunctured pattern, The E_b/N_0 ratios provided and the channel models used. At first, analysis was done on an AWGN channel for rate 1/2 lengths 400-bit and SOVA turbo decoder. We observe the gain achieved by the turbo code relative to convolutional code of relatively complexity. We can clearly see the iterative power of turbo code figure.

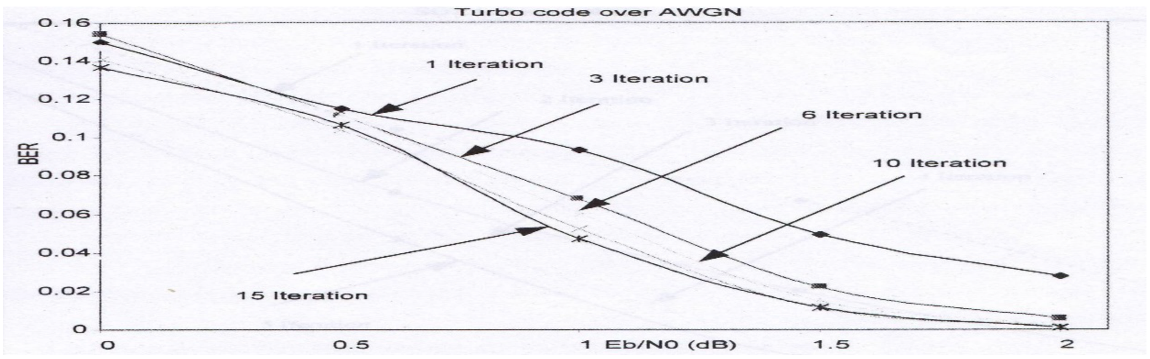


Fig 7 Performance by Simulation of Length 400 bit rate 1/2, and GeneratorPolynomial G = {7,5},Turbo code over AWGN Channel.

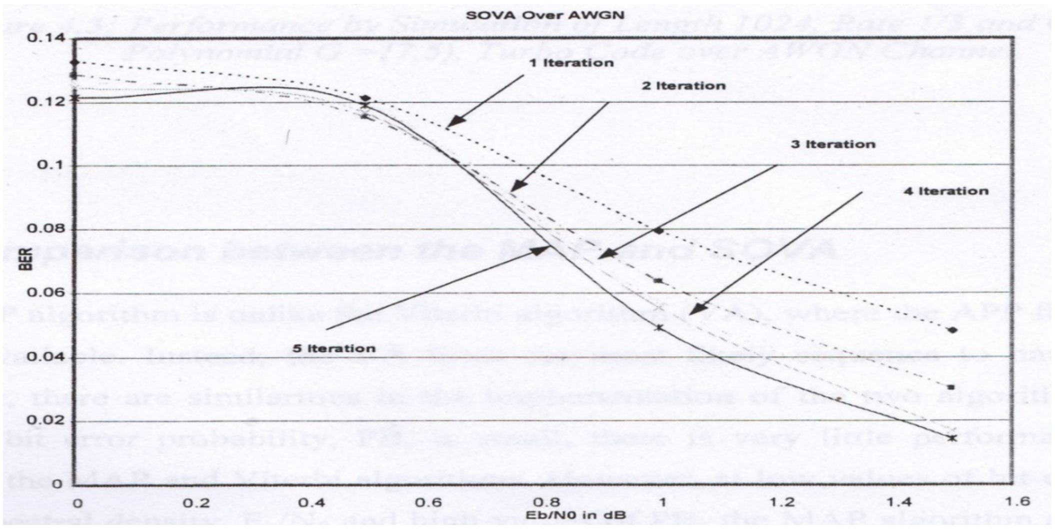


Fig 8 Performance by simulation ofLength 1024, Rate 1/2 and GeneratorPolynomial G = {7,5}, Turbo Code over AWGN Channel.

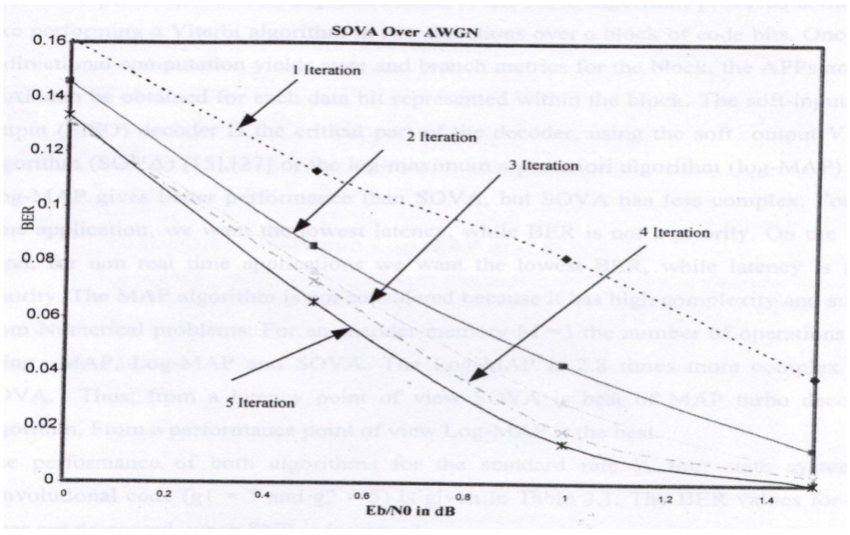


Fig 9 Performance by Simulation of Length 1024, Rate 1/3 and GeneratorPolynomial G = {7,5} Turbo Code over AWGN Channel.

4.9 Comparison between the MAP and SOVA

The MAP algorithm is unlike the Viterbi algorithm (VA), where the APP for each data bit is not available. Instead, the VA finds the most likely sequence to have transmitted. However, there are similarities in the implementation of the two algorithms. When the decoder bit error probability, PB, is small, there is very little performance difference between the MAP and Viterbi algorithms. However, at low values of bit-energy to noise power spectral density, E_b/N_o and high values of PB, the MAP algorithm can outperform decoding with a soft-output Viterbi algorithm called SOVA [15] by 0.5 dB or more [16]. For turbo codes, this can be very important, since the first decoding iterations can yield 49.

Poor error performance: The implementation of the MAP algorithm proceeds somewhat like performing a Viterbi algorithm in two directions over a block of code bits. Once this bidirectional computation yields state and branch metrics for the block, the APPs and the MAP can be obtained for each data bit represented within the block. The soft-input/soft-output (SISO) decoder is the critical part of the decoder, using the soft output Viterbi algorithm (SOVA) [15], [27] or the log-maximum a posteriori algorithm (log-MAP) [27]. Log-MAP gives better performance than SOVA, but SOVA has lesser complexes. For real time application, we want the lowest BER, while latency is not a priority. The MAP algorithm is not considered because it has high complexity and suffers from Numerical problem. For an encoder memory M=3 the number of operations [18] using MAP, Log-MAP and SOVA. The Log-MAP is 2.8times more complex than SOVA. Thus, from a latency point of view SOVA is best of MAP turbo decoding algorithm, form a performance of view Log-MAP is the best.The performance of both algorithms for the standard rate 1/2 four state systematic convolutional code (g1 = 7 and g2 = 5) is given in Table 1. The BER values for both cases are decreased, when SNR is increased.

Table1 BER for a four state code using MAP decoder & SOVA decoder

Eb/N0(dB)	0	0.1	0.5	1.5	2
Log-MAP	0.19714	0.06446	0.02183	0.006272	0.00225
SOVA	0.145829	0.1066	0.059479	0.017459	0.004567

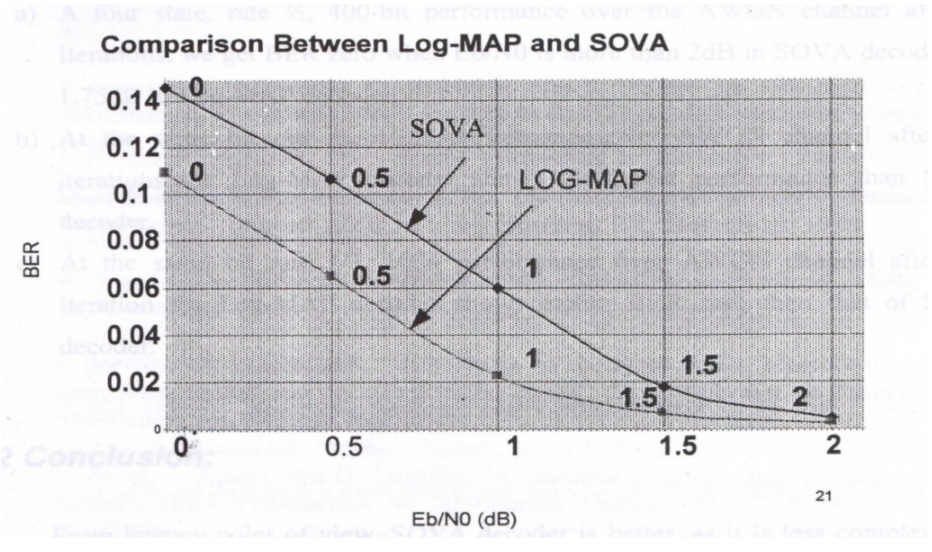


Fig 10 Performance by Simulation of Length 1024, Rate 1/2 and GeneratorPolynomial G = {7,5}, Turbo Code over AWGN Channel.

5. Results

A four state, rate1/2, 400- bit performance over the AWGN channel after 15 Iterations, we get BER zero whenm E_b/N_0 is more than 2dB in SOVA decoder and 1.75dB in Log-MAP decoder.At the same bit rate 1/2, 1024 performance over AWGN channel after five iterations the Log-MAP decoder shows the better performance than SOVA decoder.At the same bit rate 1/3, 1024 performance over AWGN channel after five iteration the Log-MAP decoder shows earlier BER zero than that of SOVA decoder.

6. Conclusion

From latency point of view, SOVA decoder is better, as it is less complex, than Log-MAP decoder. On the other hand, the performance of Log-MAP decoder is more sound than SOVA decoder. But it is more complex than SOVA.

7. Recommendation

Research on the optimum decoding strategies of the MAP decoder and SOVA decoder over the Raleigh Channel is recommended.

References

1. Berrou G., Glavieuc A., and Thitmajshima P., *"Near Shannon limit error-coding: Turbo codes",* in proc.1993, int. conf. com., Geneva, Seitzerland, May 1993,pp.1064-1070.
2. Bahl L R., Cocke J., Jelinek F., and Racic J.,*"Optimal decoding of linear codes for minimizing symbol error rate,"* IEEE Trans, Inform. Theory, Vol, IT-20, pp, 284-287, 1974
3. Benedetto S., Divsalar D., Montorsi G., and Pollara F., *"A soft-input soft-output Maximum A posteriori (MAP) module to decode parallel and serial concatenated codes"*,TDA progress report 42-127, November 15,1996.
4. Eroz Mustafa, Roger Hammons A. Jr.,*"On the design of prunableinterleavers for Turbo codes,"* in proc. IEEE VTC'99, Houston, TX, May 15-19,1999.
5. Ungerboeck G., *"Channel Coding with Multilevel/phase Systems"*, IEEE Trans, on information Theory, vol, 28no,1, pp, 55-67, January 1982.
6. Berrou C. and Glavieux A., *"Reflections on the prize paper: Near Optimum error correctin coding and decoding turbo codes"*, IEEE Information Theory Society Newsletter, vol. 48 no.2, june 1998.
7. Perez L., Sghers J., and Costello D., *"A Distance Spectrum Interpretation of turbo Codes"*, IEEE Transaction on Information Theory, vol, 42, 6, pp, 1698-1709, Nov, 1996.
8. Barbulescu A. S. and Pietrobon S.S., *"On terminating the trellis of Turbo –Codes in the Same State"*, IEE Electronics Letters, vol.31, no. I, Jan 1995.
9. Blackert W. J., Hall E. K., and Wilson S. G., *"Turbo Code Termination and InterleaverConditons"*, IEE Electronics Letters, vol, 31, no. 24, pp, 2082-2084, Nov 1995.
10. Sklar B., *"Digital Communications: Fundamentals and Applications"*, Chapter 6, Prentice-Hall International, Inc.1988.
11. Viterbi A.J, *"Convolutional Codes and their performance in Communication Systems"*. IEEE Trans Communication Technology, vol.Com-19, no.15, pp 751-772, October 1971.
12. Hagenauer J., Hoeher P.,*"A Viterbi Algorithm with Soft-Decision Outputs and its Applications,"* proc. GLOBECOM' 89, Dallas, Texas, November 1989, pp. 1680-1686.
13. Pietrobon S.S., *"Implementation and performance of a Turbo/MAP decoder,"*Int't, J Satellite Communications vol. 15, Jan-Feb 1998, pp. 23-46.

14. Sklar B., "*Digital Communications Fundamentals and Applications"*, Second Edition (Upper Saddle River, NJ: Prentice-Hall, 2001)

15. Robertson P., Villebrum E., and Hoeher P., *"A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain,"* proc, of ICC '98, Seattle, Washington, June 1995, pp, 1009-1013

16. Berrou C., Glavieux A., *"Near Optimum Error Correcting Coding and Decoding Turbo Codes,"* IEEE Trans. On Communication, vol. 44, no. 10, October 199, pp 1261-1271.

17. Robertson P., Hoeher P., and Villbrun E., *"Optimal and Sub-Optimal MAP Algorithms Suitable for Turbo Decoding",*Europeon Transactions on Telecommunications. Vol.8, no. 2, pp, 119-125, March-April 1997, paper used in writing York's turbo coders.

18. Jung P., *" Comparison of Turbo Code Decoders Applied to Short Frame Transmission"*, IEEE Journal on selected Areas in Communications, vol. 14, no.3, pp, 530-537, April 1996.

19. Robertson P., *"Improving Decoder and Code Structure of Parallel Concatenated Recursive Systematic (Turbo) Codes"*, in IEE Trans of International Conference on Universal Personal Communications, San Diego, Sept. 1994, pp, 183-187.

20. Hagenauer J., Robertson P., and Papke L., *"Iterative (Turbo) Decoding of Systematic Convolutional Codes with the MAP and SOVA Algorithms"*, in ITG-Fachbericht 130, Oct 1994, pp, 21-29.

21. Erfanin J., Pasupathy S., and Gulak G., *"Reduced Complexity symbol Detectors with Parallel Structures for ISI Channels"*, IEEE Trans Communications, vol.42, pp. 1661-1671, Feb, Mar. Apr. 1994.

22. Koch W. and Baier A., *"Optimum and Sub=Optimum Detection of Coded Data Disturbed by Time Varying ISI"*, in proceedings GLOBECOM '90, San Diego, Dec.1990, pp. 1679-1684.

23. Robertson P., *"Illuminating the Structures of Code and Decoder for parallel Concatenated Recursive Systematic (Turbo) Codes"*, in proceeding of GLOBECOM 94, San Francisco, December 1994, pp. 1298-1303.

24. Woodard J.P. and Hanzo L., *"Comparative Study of Turbo Decoding Techniques: An overview,"* IEEE Trans, on vehicular Technology, Nov. 2000, vol. 49, No. 6, pp 2208-2233.