DESIGN OF TURBO DECODER MODEL USING MAP AND SOFT-INPUT SOFT-OUTPUT VERTIBI ALGORITHM FOR AWGN AND RAYLEIGH CHANNELS

Charanjitkaur (ECE)
M.tech student, BBSBEC
Fatehgarh Sahib, Punjab, India

Supervisor's Name: HardeepSingh Rayat (HOD,ECE dept. BBSBEC Fatehgarh Sahib)

ABSTRACT-Turbo coding is an advanced error correction technique widely used in the communications industry. Its ease of implementation and its phenomenal performance, which in many cases pushes right up to Shannon's theoretical limits, lets one speculate that turbo coding is the right way to encode and decode digital data. Turbo encoders and decoders are key elements in today's communication systems to achieve the best possible data reception with the fewest possible errors. The basis of turbo coding is to introduce redundancy in the data to be transmitted through a channel. The redundant data helps to recover original data from the received data. This paper explores and analyzes techniques in convolution turbo decoding with the vertibi and MAP algorithm. LOG-MAP and SOVA Algorithms have been studied and their design considerations have been presented. Turbo coding is an advanced error correction technique widely used in the communications industry. Turbo encoders and decoders are key elements in today's communication systems to get the best possible data reception with the fewest possible errors. The basis of turbo coding is to introduce redundancy in the data to be transmitted through a channel.

***KEYWORDS*:Turbo decoder, Forward Error Correction, SOVA Algorithm, MAP Algorithm.**

1. INTRODUCTION

In digital systems, data is encoded into strings of zeros and ones, corresponding to the on and off states of semiconductor switches. This has brought about fundamental changes in how information is processed. Real-world data is typically in analog form; this is the only way we can perceive it with our senses [1]. This analog information needs to be encoded into a digital representation—for example, into a string of ones and zeros. The conversion from analog to digital and back are processes that have become ubiquitous, as, for example, in the digital encoding of speech. Digital information is treated differently in communications than analog information. Signal estimation becomes signal detection; that is, a communications receiver need not look for an analog signal

and make a "best" estimate, it only needs to make a decision between a finite number of discrete signals, say a one or a zero in the most basic case [2]. Digital signals are more reliable in a noisy communications environment. They can usually be detected perfectly, as long as the noise levels are below a certain threshold. This allows us to restore digital data, and, through error-correcting techniques, even correct errors made during transmission. Digital data can easily be encoded in such a way as to introduce dependency among a large number of symbols, thus enabling a receiver to make a more accurate detection of the symbols. This is called error control coding [1]. The digitization of data is convenient for a number of other reasons too. The design of signal processing algorithms for digital data seems much easier than designing analog signal processing algorithms. Turbo codes are finding use in (deep space) satellite communications and other applications where designers seek to achieve reliable information transfer over bandwidth- or latency-constrained communication links in the presence of data-corrupting noise [3].

2. BASIC COMMUNICATION SYSTEM

Figure 1 shows the basic configuration of a point-to-point digital communications link. The data to be transmitted over this link can come from some analog source, in which case it must first be converted into digital format (digitized), or it can be a digital information source. If this data is a speech signal, for example, the digitizer is a speech codec [9]. Usually the digital data is source encoded to remove unnecessary redundancy from it; that is, the source data is compressed [10]. This source encoding has the effect that the digital data which enter the encoder has statistics which resemble that of a random symbol source with maximum entropy; that is, all the different digital symbols occur with equal likelihood and are statistically independent [5]. The channel encoder operates on this compressed data, and introduces controlled redundancy for transmission over thechannel. The modulator converts the discrete channel symbols into waveforms which are transmitted through the waveform channel. The demodulator reconverts the waveforms back into a discrete sequence of received symbols, and the decoder reproduces an estimate of the compressed input data sequence, which is subsequently reconverted into the original signal or data sequence. We usually need to acquire carrier frequency and phase synchronization, as well as symbol timing synchronization, in order for the receiver to be able to operate Another important feature in some communication systems is automatic repeat request (ARQ). In ARQ the receiver also performs error detection and, through a return channel, requests retransmission of erroneous data blocks, or data blocks that cannot be reconstructed with sufficient confidence [7]. ARQ can usually improve the data transmission quality substantially, but the return channel needed for ARQ is not always available, or may be impractical.

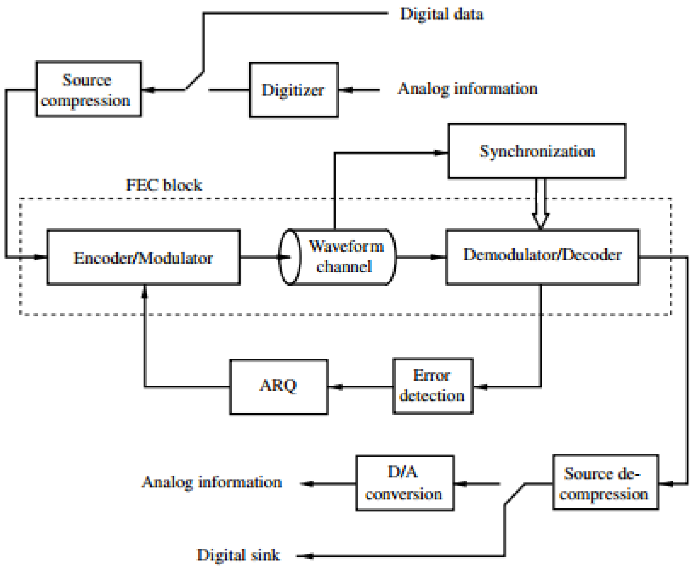

Figure 1: Block Diagram of Basic Communication System using encoder and decoder

Error control coding without ARQ is termed forward error correction or control (FEC). FEC is more difficult to perform than simple error detection and ARQ, but dispenses with the return channel. So, basically in digital communication there are two kinds of error coding techniques, ARQ and FEC explained in detail below.

3. TYPES OF ERROR CORRECTION TECHNIQUES

Two broad categories of Error Correction techniques are studied here in this paper, ARQ (Automatic Repeat Request) and FEC (Forward Error Correction).

I) Automatic Repeat Request (ARQ)

Automatic Repeat Request (ARQ) is an error control method for data transmission that makes use of error-detection codes, acknowledgment and/or negative acknowledgment messages, and timeouts to achieve reliable data transmission. An *acknowledgment* is a message sent by the receiver to indicate that it has correctly received a data frame [7]. Usually, when the transmitter does not receive the acknowledgment before the timeout occurs (i.e., within a reasonable amount of time after sending the data frame), it retransmits the frame until it is either correctly received or the error persists beyond a predetermined number of retransmissions.Three types of ARQ protocols are Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ.ARQ is appropriate if the communication channel has varying or unknown capacity, such as is the case on the Internet. However, ARQ requires the availability of a back channel, results in possibly increased latency due to retransmissions, and requires the maintenance of buffers and timers for retransmissions, which in the case of network congestion can put a strain on the server and overall network capacity.

II) Forward error correction

An error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or *parity data*, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.Error-correcting codes are usually distinguished between convolutional codes and block codes:

- *Convolutional codes* are processed on a bit-by-bit basis. They are particularly suitable for implementation in hardware, and the Viterbi decoder allows optimal decoding.
- *Block codes* are processed on a block-by-block basis. Early examples of block codes are repetition codes, Hamming codes and multidimensional parity-check codes. They were followed by a number of efficient codes, Reed–Solomon codes being the most notable due to their current widespread use.
- Turbo codes and low-density parity-check codes (LDPC) are relatively new constructions that can provide almost optimal efficiency.

Shannon's theorem is an important theorem in forward error correction, and describes the maximum information rate at which reliable communication is possible over a channel that has a certain error probability or signal-to-noise ratio (SNR). This strict upper limit is expressed in terms of the channel capacity. More specifically, the theorem says that there exist codes such that with increasing encoding length the probability of error on a discrete memory less channel can be made arbitrarily small, provided that the code rate is smaller than the channel capacity. The code rate is defined as the fraction *k/n* of *k* source symbols and *n* encoded symbols [8].The actual maximum code rate allowed depends on the error-correcting code used, and may be lower. This is because Shannon's proof was only of existential nature, and did not show how to construct codes which are both optimal and have efficient encoding and decoding algorithms.

4. TURBO CODES

Turbo codes are nowadays competing with LDPC codes, which provide similar performance. According to Shannon, the ultimate code would be one where a message is sent infinite times, each time shuffled randomly. The receiver has infinite versions of the message albeit corrupted randomly. From these copies, the decoder would be able to decode with near error-free probability the message sent. This is the theory of an ultimate code, the one that can correct all errors for a virtually signal. Turbo code is a step in that direction [3]. But it turns out that for an acceptable performance we do not really need to send the information infinite number of times, just two or three times provides pretty decent results for our earthly channels. In a SISO, it firstly computes branch metrics (or *γ* metrics), which represent the probability of a transition occurring between two trellis states. Then a SISO computes forward and backward recursions. Forward recursion (or *α* recursion) computes a trellis section (i.e., the probability of all states of the trellis) using the previous trellis section and branch metrics between these two sections, while backward recursion (or *β* recursion) computes a trellis section using the future trellis section and branch metrics between these two sections. Turbo decoding [1] is increasingly proposed in emerging and future digital communication systems, for example, fiber-optic communication, wireless communication, and storage applications. Practical turbo decoder designs, as the one used for IEEE 802.16e, Long-term Evolution (LTE) or IEEE 802.11 standards, require high data throughput (several hundred of Mbps) and low latency (ten ms or so). To cope with these requirements, turbo decoder implementations have to be massively parallel. Turbo codes are nowadays competing with LDPC codes, which provide similar performance. Turbo codes involve the deisgn of an turbo encoder and a parallel concatenated turbo decoder as explained below:

I) Turbo encoder

This encoder implementation sends three sub-blocks of bits. The first sub-block is the m-bit block of payload data. The second sub-block is n/2 parity bits for the payload data, computed using a recursive systematic convolutional code (RSC code). The third sub-block is n/2 parity bits for a known permutation of the payload data, again computed using an RSC convolutional code. Thus, two redundant but different sub-blocks of parity bits are sent with the payload. The complete block has m + n bits of data with a code rate of m/(m + n). The permutation of the payload data is carried out by a device called an Interleaver.Hardware-wise, this turbo-code encoder consists of two identical RSC coders, C_1 and C_2, as depicted in the figure, which are connected to each other using a concatenation scheme, called parallel concatenation:

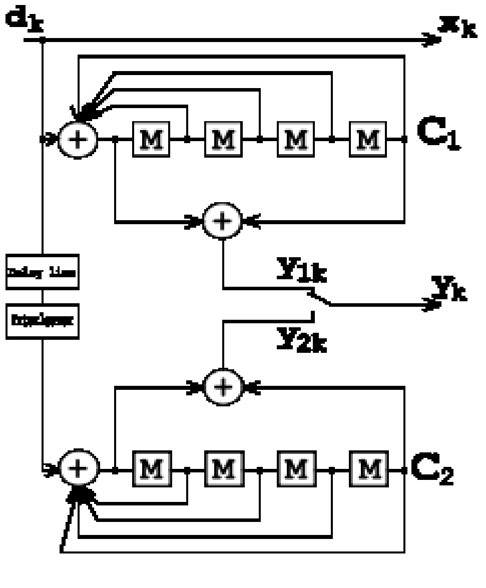

Figure 2: Turbo encoder

In the figure 2, M is a memory register. The delay line and interleaver force input bits d_k to appear in different sequences. At first iteration, the input sequence d_k appears at both outputs of the encoder, x_k and y_{1k} or y_{2k} due to the encoder's systematic nature. If the encoders C_1 and C_2 are used respectively in n_1 and n_2 iterations, their rates are respectively equal to

$$R1 = \frac{n1 + n2}{2n1 + n2}$$
$$R2 = \frac{n1 + n2}{n1 + 2n2}$$

II) Turbo decoder

Two elementary decoders are interconnected to each other, but in serial way, not in parallel. The decoder operates on lower speed (i.e., R_1), thus, it is intended for the C_1 encoder, and $DEC_2$is for C_2 correspondingly. DEC_1 yields a soft decision which causes L_1 delay. The same delay is caused by the delay line in the encoder. The DEC_2's operation causes L_2 delay.

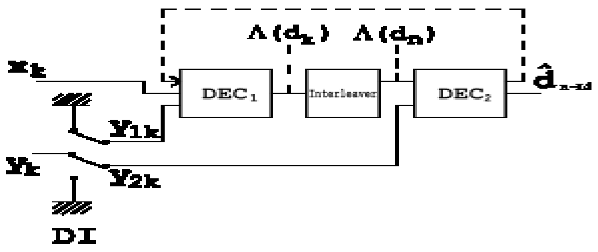

Figure 3: Turbo decoder

An Interleaver installed between the two decoders is used here to scatter error bursts coming from DEC_1 output. DI block is a demultiplexing and insertion module. It works as a switch, redirecting input bits to DEC1 at one moment and to DEC_2 at another. In OFF state, it feeds both y_{1k} and y_{2k} inputs with padding bits (zeros). Consider a memory less AWGN channel, and assume that at *k*-th iteration, the decoder receives a pair of random variables:

$x_k = (2d_k - 1) + a_k$

$y_k = (2y_k - 1) + b_k$

Where a_k and b_k are independent noise components having the same variance σ^2. y_k is a *k*-th bit from y_k encoder output. Redundant information is demultiplexedand sent through *DI* to DEC_1 (when yk = y1k) and to DEC_2 (when yk = y2k).

5. TURBO DECODING ALGORITHMS

There are various decoding algorithms available for decoding of turbo codes. All the algorithms are based upon the trellis-based estimation. The trellis based estimation algorithms are classified into two types. They are sequence estimation algorithms and symbol-by-symbol estimation algorithms. The Viterbi algorithm, SOVA (soft output Viterbi algorithm) and improved SOVA are classified as sequence estimation algorithms [10].Whereas the MAP algorithm, Max-Log-Map and the Log-Map algorithm are classified as symbol-by-symbol estimation algorithms. In general the symbol-by-symbol estimation algorithms are more complex than the sequence estimation algorithms but their BER performance is much better than the sequence estimation algorithms.

I) MAP Algorithm (HARD DECISION DECODING)

Maximum APosteriori algorithm, now commonly named BCJR algorithm, presents an optimal decoding method for linear codeswhich minimizes the symbol error probability. This is in contrast to the traditionally usedViterbi algorithm, which is based on the principle of maximum length sequenceestimation (MLSE) [10]. In essence the Viterbi algorithm minimizes the probability ofsequence (or word) error, which does not translate to minimizing the probability ofindividual bit (symbol) errors.

Let $u = (u_1, u_2 \dots u_N)$ be the binary random variables representing information bits. In the systematic encoders, one of the outputs $x_s = (x^s_1, x^s_2 \dots x^s_N)$ is identical to the information sequence u. The other is the parity information sequence output $x_p = (x^p_1, x^p_2 \dots x^p_N)$. We assume BPSK modulation and different channel with noise spectrum density N_o. The noisy versions of the outputs is$y_s = (y^s_1, y^s_2 \dots y^s_N)$ and $y_p = (y^p_1, y^p_2 \dots y^p_N)$, and $y = (y_s, y_p)$ is used for simplicity. In the MAP decoder, the decoder decides whether $u_k = +1$ or $u_k = -1$ depending on the sign of the following log-likelihood ratio (LLR)

$$L_R(u_k) = \log \frac{P(u_k = +1 \mid y)}{P(u_k = -1 \mid y)}$$

---------------------- (1)

The 'a priori' probability of information bits generated by the other MAP decoder must be considered in iterative decoders. MAP decoding, as indicated in the equations above, is a multiplication-intensive operation as it stands. The algorithm is likely to be considered too complex for implementation in many communication systems. However, the algorithm can be rearranged in the log-domain to reduce the computational. Therefore, the log-domain computations of the BCJR algorithm can be separated into three main categories for radix-2 trellises:

1. Branch metric computation
2. Forward/ Backward metric Computation
3. Combination of forward and backward state metrics

Let $S_k$denote the state of the encoder at time k. It can take values from 0 to 2M-1 where M is the number of memory elements in the encoder. LLR can be rewritten as

$$L_R(u_k) = \log \frac{\sum s_k \sum s_{k-1} \gamma_1(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1}).\beta_k(S_k)}{\sum s_k \sum s_{k-1} \gamma_0(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1}).\beta_k(S_k)}$$

-------------------- (2)

Where the forward recursion metric, β_k isis the backward recursion metric and $\gamma_i$is the branch metric. They are defined as

$$\alpha_k(S_k) = \sum_{S_{k-1}} \sum_{i=0} \gamma_i(y_k, S_{k-1}, S_k).\alpha_{k-1}(S_{k-1})$$

------------------------- (3)

$$\beta_k(S_k) = \sum_{S_{k+1}} \sum_{i=0} \gamma_i(y_{k+1}, S_k, S_{k+1}).\beta_{k+1}(S_{k+1})$$

----------------------------- (4)

$$\gamma_i((y^s_k, y^p_k), S_{k-1}, S_k) = q(u_k = i \mid S_k, S_{k-1}).p(y^s_k \mid u_k = i).p(y^p_k \mid u_k = i, S_k, S_{k-1}).P_r(S_k \mid S_{k-1})$$

----------------------------(5)

The parameter q (u_k = i/S_k, S_{k-1}) is either one or zero depending on whether u_k = i is possible for the transition from state S_{k-1} to S_k or not. Calculating p (y^s_k | u_k = i) and p (y^p_k| u_k = i, S_k, S_{k-1}) is trivial if the channel is AWGN. The last component P_r ($S_k|S_{k-1}$) usually has a fixed value for all k. However, this is not the case in the iterative decoding.

II) SOVA Algorithm (SOFT DECISION DECODING)

However, in practice the MAP turbo decoder is too complex to be implemented due to the large number of multiplications and the need of non-linear functions. For thatreason, two simplified versions of it were proposed in the past, namely Log-MAP and Max-Log-MAP *[3]*. The latter algorithm is sub-optimum in terms of bit error rate (BER) performance but easier to be implemented, as it requires only additions and the max operator.Another sub-optimum algorithm that is suitable for turbo decoding is the soft output Viterbi algorithm (SOVA). It is amodified Viterbi algorithm (VA) that produces, in addition to the most likely path sequence, a reliability value of each estimated bit **[4].** It was found that the iterative SOVA is 0.7 dB worse than the MAP algorithm at BER of **IO4** *[3]*. This is largely because the SOVA considers only two path sequences to update its **soft** output, namely the survivor and the concurrent path sequences. A first attempt to improve the SOVA was reported in ***[5]*** with two proposed modifications, *so* as to comect its soft output and to follow a Gaussian distribution. In the first modification, the extrinsic information is normalized by multiplying with a correcting factor c that depends on the variance of the decoder output, while in the second one (that is less effective) the correlation in the decoder input is eliminating by inserting two more correcting coefficients. Another attempt to improve the SOVA was described in [6] where the reliability of the soft output is limited to a small range of values. In [6] was also described the SOVA updating soft output rule by *Butfail*and it was later shown that this is equivalent to the Max-Log-MAP algorithm **[7].** Finally, the Max-Log-MAP turbo decoder was improved in **[SI** by following a normalization method similar to ***[5]*** hut keeping constant the correcting factor c. It is known that the performance of a SOVA (soft output Viterbi algorithm) turbo decoder can be improved, as the extrinsic information that is produced at its output is over optimistic. A new parameter associated with the branch metrics calculation in the standard Viterbi algorithm is introduced that affects the turbo code performance. Different parameter values show a simulation improvement in the AWGN channel as well as in an uncorrelated Rayleigh fading channel [10]. There are different efficient approaches proposed to improve the performance of soft-output Viterbi algorithm (SOVA)-based turbo decoders. In the first approach, an easily obtainable variable and a simple mapping function are used to compute a target scaling factor to normalize the extrinsic information output from turbo decoders. The scaling factor can be a variable scaling factor or a fixed scaling factor. In Variable scaling factor method, a scaling factor „c‟ of should be employed to normalize the soft output of SOVA decoders. In practice, to compute the mean and variance of the soft output from SOVA decoders, multiplication and addition operations must be performed at each symbol-processing cycle within each iterative decoding. Also, to compute the final scaling factor, a division operation must be performed before the next iteration begins. All of these imply that a practical SOVA-based turbo decoder with the normalization process embedded may work either with a larger clock cycle period or with a considerable extra latency when pipeline techniques are employed [10].The SOVA is a modified Viterbi algorithm which produces soft outputs associated with the decoded bit sequence. These modifications include a modified metric computation and a reliability value update along the maximum likelihood (ML) path. Let m denote a trellis branch into a node and $M_k^{(m)}$ denote the accumulated metric at time k for branch m.

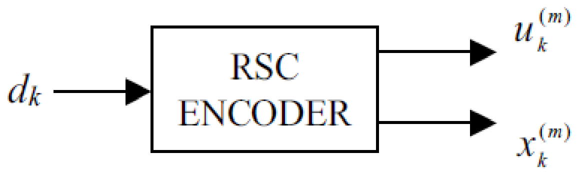

Fig.4. An RSC Encoder

Also, let $u_k^{(m)}$ be the systematic encoder output (in bipolar voltage form) for path m and $x_k^{(m)}$be the corresponding parity output (Fig. 4). If $y_{k,1}$and $y_{k,2}$ are the channel outputs corresponding to the systematic and parity outputs of the RSC encoder, then the metric used for the SOVA algorithm becomes

$$M_k^{(m)} = M_{k-1}^{(m)} + u_k^{(m)} L_c y_{k,1} + x_k^{(m)} L_c y_{k,2}$$

6. RESULTS:

1. USING SOVA DECODER WITH PUNCTURED CODES FOR RAYLEIGH CHANNEL

INPUT PARAMETERS

Decoding Algorithm: Soft OutputVertibiAlgorithm
Frame Size: 400
Code Generator Matrix: [1 1 0; 1 0 1]
Punctured / Unpunctured: punctured codes, code rate: 1/2
Number of iterations for each frame: 5
Number of frame errors to terminate: 10

OUTPUT

```
Eb/N0 in dB: 2.0
=== SOVA decoder ===
 Frame size =    400
code generator:
    1    1    1    1    1
    1    0    0    0    1
 Punctured, code rate = 1/2
Iteration number =       5
Terminate frame errors =     10
Eb / N0 (dB) =       2.00
```

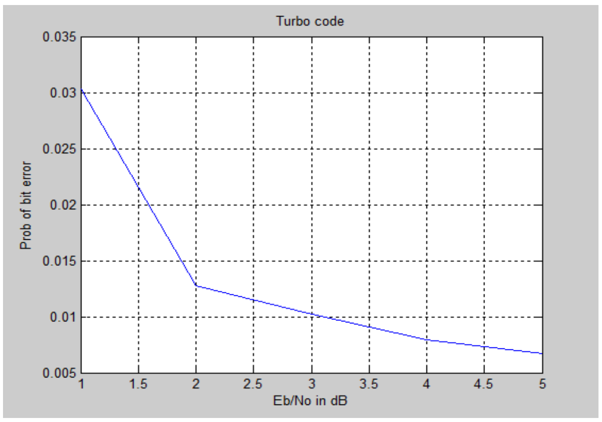

Figure 5. PROBABILITY OF BIT ERROR vs. Eb/No (Signal bit energy / Noise Spectral Density) for SOVA decoder with punctured codes for Rayleigh channel

2. USING SOVA DECODER WITH PUNCTURED CODES FOR AWGN CHANNEL

INPUT PARAMETERS

Decoding Algorithm: Soft OutputVertibi Algorithm
Frame Size: 400
Code Generator Matrix: [1 1 0; 1 0 1]
Punctured / Unpunctured: Punctured codes, code rate: 1/2
Number of iterations for each frame: 5
Number of frame errors to terminate: 10

OUTPUT

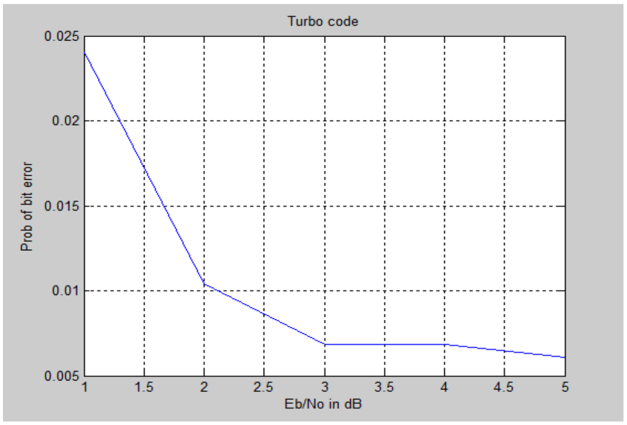

Figure 6.PROBABILITY OF BIT ERROR vs. Eb/No (Signal bit energy / Noise Spectral Density) for SOVA decoder with punctured codes for AWGN channel

3. USING MAP DECODER WITH PUNCTURED CODES FOR RAYLEIGH CHANNEL

INPUT PARAMETERS

Decoding Algorithm: Maximum - a - posterior
Frame Size: 400
Code Generator Matrix: [1 1 0; 1 0 1]
Punctured / Unpunctured: Punctured codes, code rate: 1/2
Number of iterations for each frame: 5
Number of frame errors to terminate: 10

OUTPUT

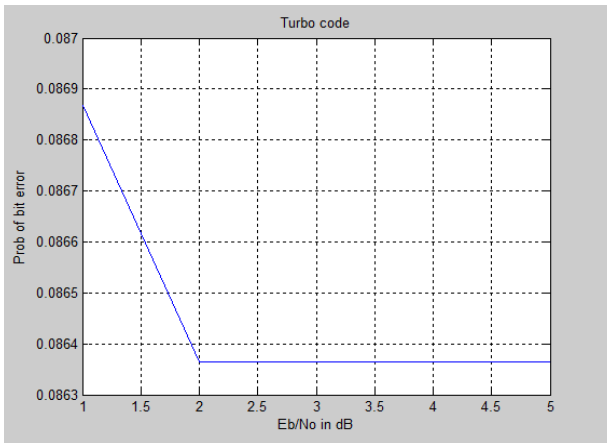

Figure 7.PROBABILITY OF BIT ERROR vs. Eb/No (Signal bit energy / Noise Spectral Density) for MAP decoder with punctured codes for Rayleigh channel

4. USING MAP DECODER WITH PUNCTURED CODES FOR AWGN CHANNEL

INPUT PARAMETERS

Decoding Algorithm: Maximum - a - posteori
Frame Size: 400
Code Generator Matrix: [1 1 0; 1 0 1]
Punctured / Unpunctured: Punctured codes, code rate: 1/3
Number of iterations for each frame: 5
Number of frame errors to terminate: 5

OUTPUT

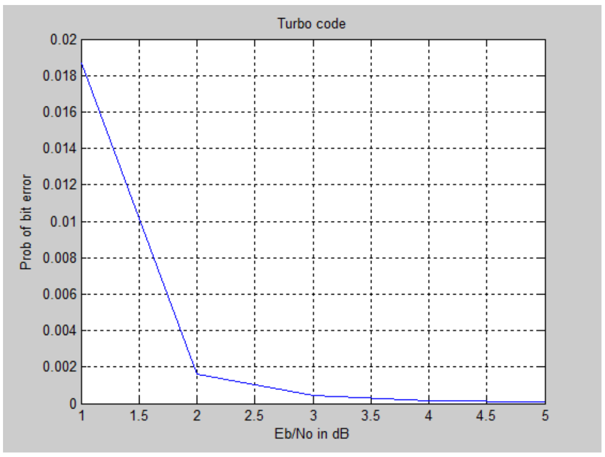

Figure 8.PROBABILITY OF BIT ERROR vs. Eb/No (Signal bit energy / Noise Spectral Density) for MAP decoder with punctured codes for AWGN channel

7. CONCLUSIONS FROM THE RESULT

It has been seen that probability of bit error has reduced significantly in SOVA Decoder as compared to LOG-MAP Decoder for Rayleigh channel. However, time for transmission and BER calculation in SOVA has increased significantly.

1. Maximum probability of bit error in SOVA decoder (punctured codes): 0.03 and in MAP .0865 for Rayleigh channel and .025 and .0868 for AWGN channel.
2. It has also been seen that bit error rate and frame error rate reduces significantly with respect to SNR of the channel

for each iteration at every frame transmission.

3. Also, the number of frames transmitted without error has been significantly increased in SOVA decoder for particular frame error termination.

REFERENCES

[1] Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE ChristophStuder, Student Member, IEEE, Christian Benkeser, Member, IEEE, SandroBelfanti, and Quiting Huang, Fellow, IEEE, January 2011

[2] On the Design of Turbo Codes, D. Divsalar and F. Pollara, Communications Systems and Research Section, November 1995

[3] A Lowpower Design Methodology For Turbo Encoder And Decoder, Rajeshwari. M. Banakar, Department Of Electrical Engineering, Indian Institute Of Technology, Delhi, July 2004

[4] Sae-Young Chung, G. David Forney, Thomas J. Richardson, and RüdigerUrbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit" IEEE Communications Letters, Vol. 5, No. 2, page(s):58-60, February 2001.

[5]Sadjadpour, H.R. Sloane, N.J.A. Salehi, M. Nebe, G. "Interleaver design for turbo codes", *IEEEJournal on Selected Areas in Communications*, Volume: 19, Issue: 5, page(s): 831-837, May 2001.

[6]T.P. Fowdur, K.M.S. Soyjaudah, "Joint source channel decoding and iterative symbol combining with turbo trellis-coded modulation", *Signal Processing,* Volume 89, Issue 4, page(s):570-582, April 2009.

[7]Erl-Huei Lu, Yi-Nan Lin, Wei-Wen Hung ,"Improvement of turbo decoding using cross-entropy", *Computer Communications*, Volume 32, Issue 6, page(s):1034-1038,27 April 2009.

[8]Fan Zhang and Henry D. Pfister, "On the Iterative Decoding of High-Rate LDPC Codes with Applications in Compressed Sensing", arXiv: 0903.2232v2 [cs.IT], 17 June, 2009.

[9]David Haley, Vincent Gaudet, Chris Winstead, Alex Grant, Christian Schlegel, "A dual-function mixed-signal circuit for LDPC encoding/decoding", *Integration, the VLSI Journal*, Volume 42, Issue 3,page(s): 332-339 ,June 2009.

[10] "Performance Analysis of Log-map, SOVA and Modified SOVA Algorithm for Turbo Decoder", Mohammad Salim, R.P. Yadav, S.Ravikanth, Dept. of Electronics & Communication Engineering, MNIT, Jaipur, Rajasthan (India), International Journal of Computer Applications (0975 – 8887) Volume 9– No.11, November 2010.