Indian Streams Research Journal
Volume 2, Issue. 7, Aug 2012
ISSN:-2230-7850

Available online at www.isrj.net

ORIGINAL ARTICLE

Implementation of Turbo Codes in AWGN Channel

Rupesh Singh nd Nidhi Singh

Research Scholar, CMJ University, Shillong

Abstract:
The field of forward error correction was greatly influenced by the discovery of turbo codes. This invention led to a great improvement in terms of Bit-Error-Rate (BER). Various schemes have been proposed and are based either on parallel or serial designs of concatenated decoders. These decoders are iterative using SOVA (soft output viterbi Algorithm) or MAP (maximum a posteriori) algorithms. They introduce superior recovery functions of data which have been transmitted through noisy environments. Actually, these turbo schemes compared to convolutional codes achieve better data recovery with the increase of the constraint length. This paper shows the Turbo encoder and SOVA Decoder Implementation by Simulation carried out for different Bit Error Rates, iterations, constrained lengths and Frame sizes in Additive White Gaussian Noise Channels.

INTRODUCTION:

Theoretical bounds of Shannon's capacity can be reached in terms of performance for the case of Turbo codes. Iterative procedure, as a part of a Turbo decoder system, offers excellent detection of information streams. These streams are transmitted through Gaussian channels. Consequently, the invention of Turbo codes is considered to be a major breakthrough [1]. Their applications are numerous and include OFDM [2], Wireless LAN, 3rd Generation Partnership Project (3GPP), 3G mobile telephony standards and others. Also, Turbo codes have been adopted by European Space Agency and NASA missions such as SMART-1 and Mars Reconnaissance Orbiter respectively.

Turbo codes can be categorized into two schemes. These schemes are SCCC (Serially concatenated convolutional codes) and PCCC (Parallel concatenated convolutional codes). The previous topologies can be easily discriminated by the strategy of the block connections. These connections refer to the joining of convolutional encoders in a serial or in a parallel manner. In each case various interleavers must be present. The idea of iterative function is coming from the operation of a car's turbo charger. The fuel is enhanced through a feedback connection and the car's performance increases. In accordance to the previous idea, a Turbo decoder uses feedback loop in order to reevaluate data and finally to produce a better estimation. This estimation of received data is superior compared to other systems which are utilizing Viterbi decoders. The iterative function can be constituted of APP (A Posteriori Probability) decoders with two inputs and two outputs. These decoders provide a very efficient estimation of the value of the incoming information but they cannot conclude to a decision which bits are 1's or 0's. This decision is taken by the Hard Decision section. Also, two of the previous decoders can constitute a typical iterative decoding system. This system can be considered as a block with two inputs and two outputs.

The two inputs correspond to the incoming streams from the iterative connection and from the primary receiving route. The two outputs produce the data for reevaluation (for feedback loop) and the speculated sequences (for Hard Decision). A similar idea to the previous strategy of encoding and decoding has been adopted in order to design a new PCCC system. The main differences to standard architectures

Please cite this Article as : Rupesh Singh nd Nidhi Singh , Implementation of Turbo Codes in AWGN Channel : Indian Streams Research Journal (Aug. ; 2012)

include three convolutional encoders in the encoder's section, along with the appropriate number of interleavers. In the decoder's section, the incoming data stream is demultiplexed and then each produced data stream is fed to the appropriate APP decoders. These decoders are three instead of two but they still form an iterative function for estimating the received data. Moreover, in this estimation is taken into consideration the use of an additional parameter which is a scaling factor.

ranhetcleeitltaIuirmnbmidciaelteixschtkaiyemtlhciiVunnuemifltoaaterhtmrmipeboreniaVittoAr ioiriotlfciegnrtiponbhorafiifetothAhrVhmml(egiota,oeritrndriimobfetonhniirnmoAmPtim.leargtuoTtsihr om(hiien)te,hbb.dmdrietiaIp_snnw.eecnrihttedhhnimencaegefotpiorlsrlinocitowhhfroiaoinrwtignat,fthoh wterreemammnmaseteitoittroidrionicicnfytciocastalhdcpuueerso_leanddntoueibdodcn

1) Principle of the General Soft-Output Viterbi Decoder

The Viterbi algorithm produces the ML output sequence for convolutional codes. This algorithm provides optimal sequence estimation for one stage convolutional codes. For concatenated (multistage) convolutional codes, there are two main drawbacks to conventional Viterbi decoders. First, the inner Viterbi decoder produces bursts of bit errors which degrades the performance of the outer Viterbi decoders [4]. Second, the inner Viterbi decoder produces hard decision outputs which prohibits the outer Viterbi decoders from deriving the benefits of soft decisions [4]. Both of these drawbacks can be reduced and the performance of the overall concatenated decoder can be significantly improved if the Viterbi decoders are able to produce reliability (soft-output) values [5]. The reliability values are passed on to subsequent Viterbi decoders as apriori information to improve decoding performance. This modified Viterbi decoder is referred to as the soft-output Viterbi algorithm (SOVA) decoder. Figure 1 shows a concatenated SOVA decoder.

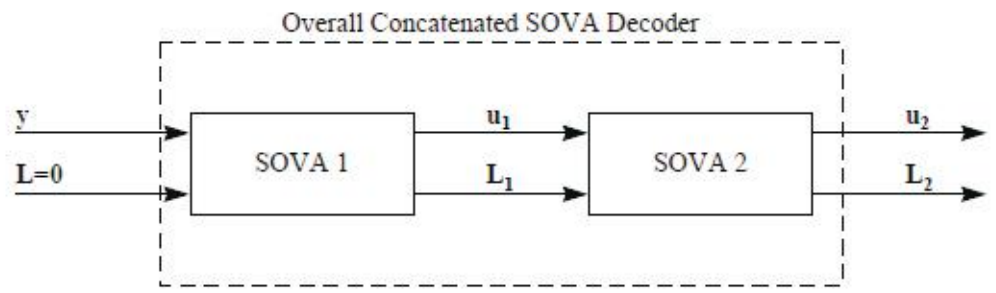


Figure 1: A concatenated SOVA decoder where y represents the received channel values, u represents the hard decision output values, and L represents the associated reliability values.

The reliability of the SOVA decoder is calculated from the trellis diagram as shown in Figure 2. In Figure 2, a 4-state trellis diagram is shown. The solid line indicates the survivor path (assumed here to be part of the final ML path) and the dashed line indicates the competing (concurrent) path at time t for state 1. For the sake of brevity, survivor and competing paths for other nodes are not shown. The label S1,t represents state 1 and time t. Also, the labels {0,1} shown on each path indicate the estimated binary decision for the paths. The survivor path for this node is assigned an accumulated metric Vs(S1,t) and the competing path for this node is assigned an accumulated metric Vc(S1,t). The fundamental information for assigning a reliability value L(t) to node S1,t's survivor path is the absolute difference between the two accumulated metrics, L(t)=| Vs(S1,t) - Vc(S1,t) | [5]. The greater this difference, the more reliable is the survivor path. For this

reliability calculation, it is assumed that the survivor accumulated metric is always “better” than the competing accumulated metric. Furthermore, to reduce complexity, the reliability values only need to be calculated for the ML survivor path (assume it is known for now) and are unnecessary for the other survivor paths since they will be discarded later.

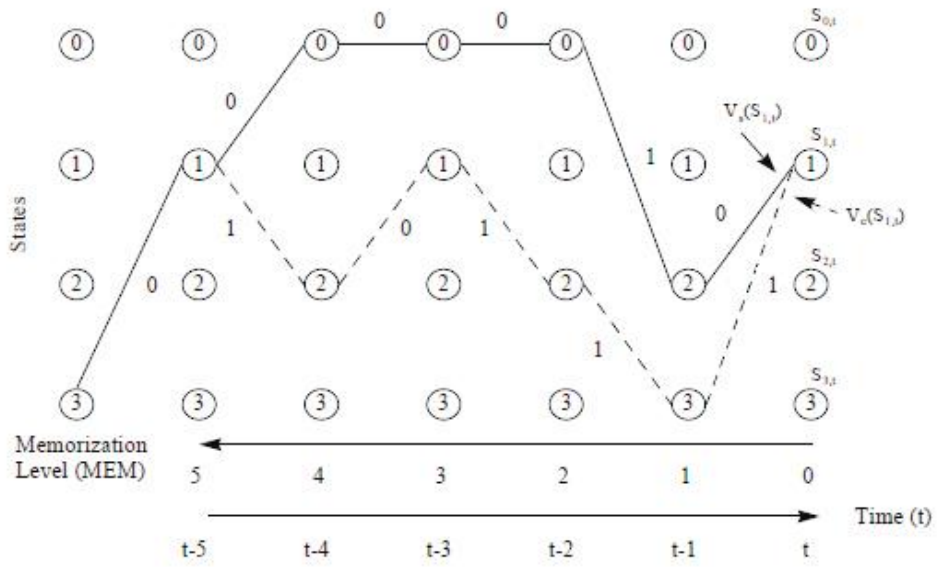


Figure 2: Example of survivor and competing paths for reliability estimation at time t

To illustrate the concept of reliability, two examples are given below. In these examples, the Viterbi algorithm selects the survivor path as the path with the smaller accumulated metric. In the first example, assume that at node S1,t the accumulated survivor metric Vs(S1,t)=50 and that the accumulated competing metric Vc(S1,t)=100. The reliability value associated with the selection of this survivor path is L(t)=|50-100|=50. In the second example, assume that the accumulated survivor metric does not change, Vs(S1,t)=50, and that the accumulated competing metric Vc(S1,t)=75. The resulting reliability value is L(t)=|50-75|=25 [6]. Although in both of these examples the survivor path has the same accumulated metric, the reliability value associated with the survivor path is different. The reliability value in the first example provides more confidence (twice as much confidence) in the selection of the survivor path than the value in the second example. Figure 3 illustrates a problem with the use of the absolute difference between accumulated survivor and competing metrics as a measure of the reliability of the decision.

In Figure 3, the survivor and competing paths at S1,t have diverged at time t-5. The survivor and competing paths produce opposite estimated binary decisions at times t, t-2, and t-4 as shown in bold labels. For the purpose of illustration, let us suppose that the survivor and competing accumulated metrics at S1,t are equal, Vs(S1,t) = Vc(S1,t) = 100. This means that both the survivor and competing paths have the same probability of being the ML path.

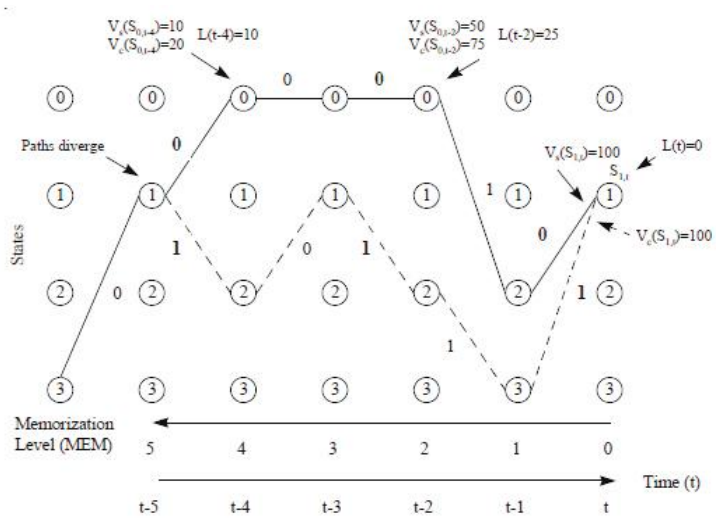


Figure 3: Example that shows the weakness of reliability assignment using metric values directly

Furthermore, let us assume that the survivor accumulated metric is "better" than the competing accumulated metric at time t-2 and t-4 as shown in Figure 3. To reduce the figure complexity, these competing paths for times t-2 and t-4 are not shown. From this argument, it can be seen that the reliability value assigned to the survivor path at time t is L(t)=0, which means that there is no reliability associated with the selection of the survivor path. At times t-2 and t-4, the reliability values assigned to the survivor path were greater than zero (L(t-2)=25 and L(t-4)=10) as a result of the "better" accumulated metrics from the survivor path. However, at time t, the competing path could also have been the survivor path because they have the same metric. Thus, there could have been opposite estimated binary decisions at times t, t-2, and t-4 without reducing the associated reliability values along the survivor path. To improve the reliability values of the survivor path, a trace back operation to update the reliability values has been suggested [4], [5]. This updating procedure is integrated into the Viterbi algorithm as follows [4]:

For node Sk,t in the trellis diagram (corresponding to state k at time t),

1. Store L(t) = | Vs(Sk,t) - Vc(Sk,t) |. (This is also denoted as D in other papers.) If there is more than one competing path, then multiple reliability values must be calculated and the smallest reliability value is then set to L(t).
2. Initialize the reliability value of Sk,t to +¥ (most reliable).
3. Compare the survivor and competing paths at Sk,t and store the memorization levels (MEMs) where the estimated binary decisions of the two paths differ.
4. Update the reliability values at these MEMs with the following procedure:

a. Find the lowest MEM>0, denoted as MEMlow, whose reliability value has not been updated.

b. Update MEMlow's reliability value L(t-MEMlow) by assigning the lowest reliability value between MEM = 0 and MEM = MEMlow. Continuing from the example, the opposite bit estimations between the survivor and competing bit paths for S1,t are located and stored as MEM={0, 2, 4}. With this MEM information, the reliability updating process is accomplished. In Figure 4, the first reliability update is shown.

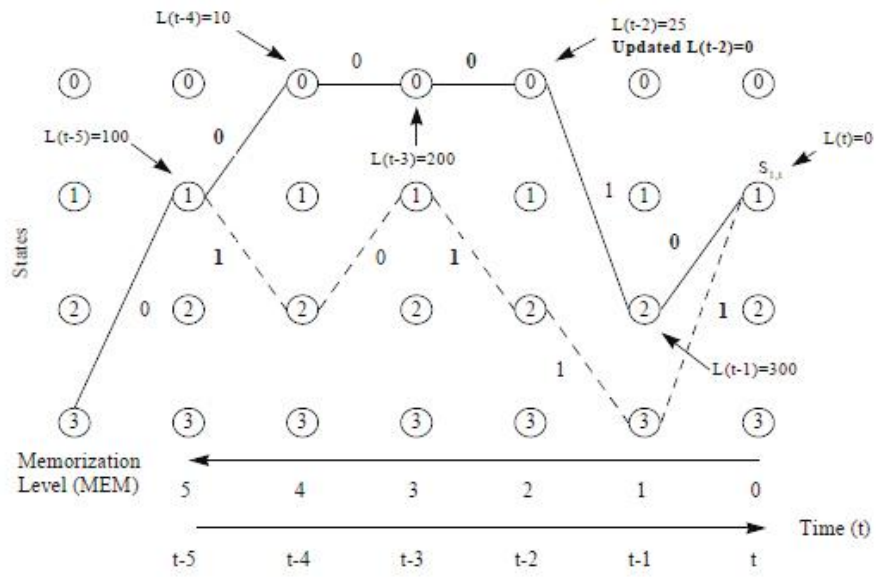


Figure 4: Updating process for time t-2 (MEM_{low}=2)

The lowest MEM>0, whose reliability value has not been updated, is determined to be MEMlow=2. The lowest reliability value between MEM=0 and MEM=MEMlow=2 is found to be L(t)=0. Thus, the associated reliability value is updated from L(t-2)=25 to L(t-2)=L(t)=0. The next lowest MEM>0, whose reliability value has not been updated, is determined to be MEMlow=4. The lowest reliability value between MEM=0 and MEM=MEMlow=4 is found to be L(t)=L(t-2)=0. Thus, the associated reliability value is updated from L(t-4)=10 to L(t- 4)=L(t)=L(t-2)=0.

2 SOVA IMPLEMENTATION

The SOVA decoder can be implemented in various ways. The straightforward implementation of the SOVA decoder [7] may become computationally intensive for large constraint length K codes and long frame

sizes because of the need to update all of the survivor paths. Because the update procedure is meaningful only for the ML path, an implementation of the SOVA decoder that only performs the update procedure for the ML path is shown in Fig.. 5.

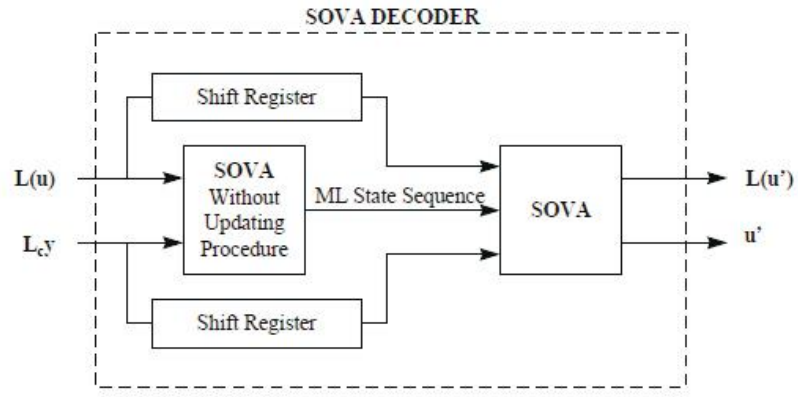


Figure 5: SOVA decoder implementation

The SOVA decoder inputs L(u) and Lcy, the a-priori values and the weighted received values respectively and outputs u' and L(u'), the estimated bit decisions and its associated "soft" or L-values respectively. This implementation of the SOVA decoder is composed of two separate SOVA decoders. The first SOVA decoder computes the metrics for the ML path only and does not compute (suppresses) the reliability values. The shift registers are used to buffer the inputs while the first SOVA decoder is processing the ML path. The second SOVA decoder (with the knowledge of the ML path) recomputes the ML path and also calculates and updates the reliability values. As it can be seen, this implementation method reduces the complexity in the updating process. Instead of keeping track and updating 2m survivor paths, only the ML path needs to be processed.

3 SOVA Iterative Turbo Code Decoder

The iterative turbo code decoder is composed of two concatenated SOVA component decoders. Figure 6 shows the turbo code decoder structure.

The turbo code decoder processes the received channel bits on a frame basis. As shown in Figure 6, the received channel bits are demultiplexed into the systematic stream y1 and two parity check streams y2 and y3 from component encoders 1 and 2 respectively [8]. These bits are weighted by the channel reliability value and loaded on to the CS registers. The registers shown in the figure are used as buffers to store sequences until they are needed. The switches are placed in the open position to prevent the bits from the next frame from being processed until the present frame has been processed.

Figure 6 shows that the turbo code decoder is a closed loop serial concatenation of SOVA component decoders. In this closed loop decoding scheme, each of the SOVA component decoders estimates the information sequence using a different weighted parity check stream. The turbo code decoder further implements iterative decoding to provide more dependable reliability/a-priori estimations from the two different weighted parity check streams, hoping to achieve better decoding performance.

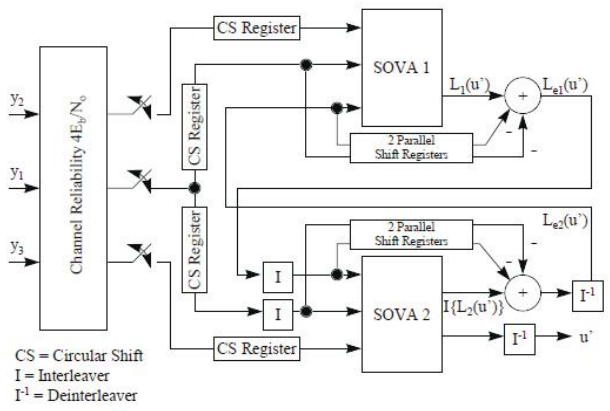
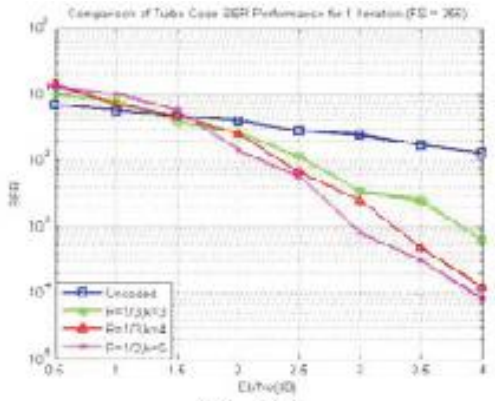


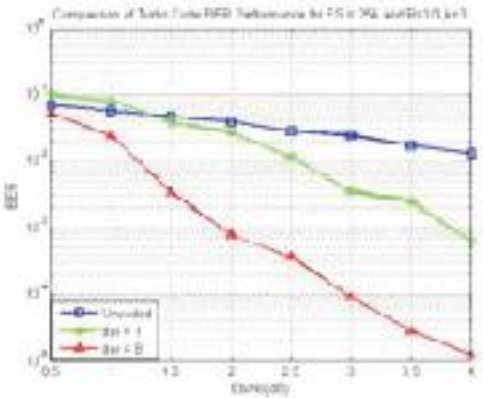
Figure 6: SOVA iterative turbo code decoder

RESULTS

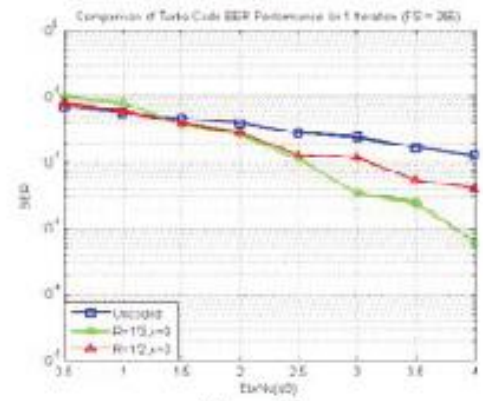
The performance of the rate 1/3 turbo code in soft decision Viterbi decoding for different constraint lengths is shown in Result 1. In this figure, it can be seen that as the constraint length increases, the performance of the code also increases, resulting in lower BER. This is the typical characteristic of any convolutional code.



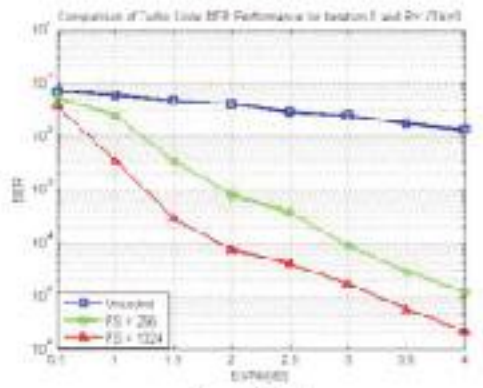
Result 1



Result 3



Result 2



Result 4

The performance of the rate 1/3 turbo code in soft decision Viterbi decoding for different constraint lengths is shown in Result 1. In this figure, it can be seen that as the constraint length increases, the performance of the code also increases, resulting in lower BER. This is the typical characteristic of any convolutional code.

The simulated performance results of turbo codes with fixed frame sizes but different rates are shown in Result 2. From these figures, it can be seen that for a fixed constraint length, a decrease in code rate increases the turbo code performance.

The simulated performance results of turbo codes with fixed frame sizes, fixed constraint length and fixed rate are shown in Result 3. From figure an increase in number of iteration improves the turbo code performance. The overall iterative (8 iterations) decoding gain for a turbo code with the same constraint length and rates but different frame sizes are shown in Result 4. As shown in these figures, the overall iterative decoding gain increases as the frame size increases.

CONCLUSION

To validate the turbo code simulation, comparisons were made between the simulated and published bit error rate (BER) results. There were some differences between the BER results for high Eb/No (>2). These differences are believed to be caused by two independent factors, namely, the numerical inaccuracies introduced by the workstations and the lack of "critical" details about the SOVA decoding algorithm. The BER performance for turbo codes is investigated for many different cases.

These different cases are summarized under the following three main categories:

1. Turbo code BER performance of 10 decoding iterations for fixed code rates and constraint lengths but different frame sizes.
2. Turbo code BER performance of 10 decoding iterations for fixed frame sizes but different code rates and constraint lengths.

3. Turbo code BER performance improvement between 1 decoding iteration and 10 decoding iterations for fixed code rates and constraint lengths but different frame sizes.

The simulation results showed many interesting properties about turbo codes that are in the same direction with current published research work. Some of these important results are listed below:

· For a fixed turbo code encoder, its performance improves as the frame size increases.

· For a fixed frame size, the turbo code performance increases under two different conditions. First, for a fixed constraint length, a decrease in code rate improves the performance. Second, for a fixed code rate, an increase in constraint length improves the performance.

· Substantial decoding gain is observed if more than one decoding iteration is used. As it is known by now, turbo code decoding can become computationally intensive. As a result, most of the simulated performance results are for high code rates, short constraint lengths, and small frame sizes.

REFERENCES

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in Proceedings of the IEEE International Conference of Communications ICC 93, IEEE, Geneva, 23-26 May 1993, pp. 1064–1070.

[2] M. K. Gupta, Vishwas Sharma, 2009, "To improve bit error rate of turbo coded OFDM transmission over noisy channel", Journal of Theoretical and Applied Information Technology, Vol 8.

[3] Claude Berrou, "A Low Complexity Soft-Output Viterbi Decoder Architecture", IEEE Trans. Commun , pp. 737 745, 1996

[4] Hagenauer, J. and Hoeher, P., "A Viterbi Algorithm with Soft-Decision Outputs and Its Applications," GLOBECOM 1989, Dallas, Texas,pp.1680-1686, Nov. 1989.

[5] Berrou, C., Adde, P. Angui, E., and Faudeil, S., "A Low Complexity Soft-Output Viterbi Decoder Architecture," Proceedings of ICC 1993, Geneva, Switzerland, pp. 737-740, May 1993.

[6] Divsalar, D. and Pollara, F., "Turbo Codes for PCS Applications," Proceedings of ICC 1995, Seattle, WA., pp. 54-59, June 1995.

[7] Benedetto, S., and Montorsi, G., "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," IEEE Transactions on Information Theory, Vol. 42, No. 2, pp. 409-428, March 1996.

[8]Hagenauer, J., Robertson, P., and Papke, L., "Iterative ("Turbo") Decoding of Systematic Convolutional Codes with the MAP and SOVA Algorithms," Proceeding of ITG, pp. 21-29, Oct. 1994

[9] P. Robertson, "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.

[10] Proakis, J. G., Digital Communications, 3rd ed., New York, McGraw-Hill, 1995.

[11] Rappaport, T.S, Wireless Communications Principles and Practice, New Jersey, Prentice-Hall, 1996.