Iteration Reduction Using SNR Offset for Turbo and Serially Concatenated Convolutional Codes

Orhan Gazi

[1] Electronic and Communication Engineering Department
Cankaya University
E-mail: o.gazi@cankaya.edu.tr

Abstract

Although turbo-like codes show superior performance close to Shannon's limits, their high decoding complexity is a major discrepancy for their use in communication systems. One method to alleviate the decoding complexity is the reduction of iteration number. In this letter, we show that iteration number can be reduced by giving a moderate amount of positive offset to the estimated *SNR* at the receiver.

1. Introduction

Turbo codes have received considerable attention since their introduction in 1993 [1]. Turbo codes show very good performance at low *SNR* values. The use of *APP* algorithms is the key factor behind the astonishing performance of the turbo codes. The *APP* algorithms are complex algorithms and when used in iterative decoding, they face very high latencies. Maximum a-posteriori *(MAP)* algorithm is one of the decoding algorithms that is very widely used in *SISO* blocks. The high latency can be alleviated via two main methods. One is the application of parallel decoding algorithms. This requires the use of multiple processors and increases hardware complexity. In [2], a new turbo code structure suitable for parallel decoding operation is introduced by the author of this letter. Another way to reduce the high latency of iteratively decodable codes is to obtain reduced complexity decoding algorithms. A number of methods for reducing the complexity of *(MAP)* algorithm has been suggested so far. The computation amount is also related to the number of iterations which greatly affects decoding latency. By reducing the number of iterations, low latency can be achieved. It is possible to have less iterations by defining stopping criteria beyond which negligible performance improvement is observed. A number of stopping criteria have been suggested in [3, 4], and [5] to terminate the decoding procedure as soon as there is no more performance improvement observed. *MAP* needs the knowledge of the channel *SNR* for the computation of the a posteriori probabilities. A number of researches has been conducted on *SNR* and channel estimation errors and their effects on the performance of turbo codes. The sensitivity of the turbo decoder on *SNR* mismatches were inspected in [6, 7] where it is found that at very high *SNR* regions even large amount of positive offset has no effect on code performance. For low or moderate *SNR* regions, the degradation in code performance becomes larger as the amount of positive offset on estimated *SNR* is increased. On the other hand, decoder is less resistant to negative offsets on estimated *SNR*. At high *SNR* regions, decoder can resists only small negative offsets on estimated *SNR* whereas its performance is broken down for large amount of negative offset on estimated *SNR*. For low or moderate *SNR* regions, decoder performance is easily smashed even with small amount of negative offsets on estimated *SNR*.

Although, the effects of *SNR* mismatch on turbo and serially concatenated convolutional codes (*SCCCs*) has been intensively investigated in terms of BER in the previous works including [6, 7], the consequences in terms of the average number of iterations has not been considered so far. In this letter, we study the effects of *SNR* offsets on the number of iterations for both turbo and *SCCCs*. We show that giving positive offset on *SNR* reduces the number of iterations and thus, decoding complexity which is related to decoding latency. It is also found that negative offset always increases the number of iterations even though code performance in terms of BER is not affected for some E_b/N_0 values. The outline of the letter is as follows. In Section II, simulation parameters are presented. Results are declared in Section III. Finally, conclusions are drawn in Section IV.

2. The MAP Algorithm

The MAP algorithm as implemented through the BCJR algorithm is summarized here. Assume that there are P complex numbers used in the constellation scheme. The constellation alphabet consists of the symbols $C = \{c_1, c_2, \ldots, c_{P-1}, c_P\}$.

The received signal vector is denoted by y and has length L, i.e., L symbols are transmitted. The task is to determine the transmitted symbols. Given the received signal vector, the probability that a specific symbol value is sent

has to be determined. Given the received vector, the probability of the k^{th} data symbol being equal to each of the symbols should be evaluated as

$$p(u_k = c_i|y) = \sum_{s,s':u_k=c_i} p(s_{k-1} = s',\ s_k = s|\ y). \tag{1}$$

Since $p(y)$ does not depend on c_i, we will actually evaluate (1) using the states at time instants $k-1$ and k

$$p(s',\ s,\ y) = p(s_{k-1} = s',\ s_k = s,\ y), \tag{2}$$

where s_n is the state of the finite state machine at time instant n. For $y = [y_1,\ y_2,\ \ldots,\ y_L]$, the vector $y_1^k = [y_1, \ldots, y_k]$ is defined. Eqn. (2) can now be stated as

$$p(s', s'y) = \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s).$$

$$\alpha_k(s) = \sum_{s'} \gamma_k(s', s)\alpha(s')$$
$$\beta_{k-1}(s') = \sum_{s} \beta_k(s)\gamma_k(s', s).$$

If logarithms of the forward ($\alpha_k(s)$) and backward ($\beta_{k-1}(s')$) state probabilities are used, the recursions for forward and backward state probabilities become as:

$$\tilde{\alpha}_k(s) = ln\sum_{s'}(e^{\tilde{\alpha}_{k-1}(s')} + e^{\tilde{\gamma}_k(s',s)})$$
$$\tilde{\beta}_{k-1}(s') = ln\sum_{s'}(e^{\tilde{\gamma}_k(s',s)} + e^{\tilde{\beta}_k(s)}).$$

The log-branch probabilities used in recursion operations are evaluated by

$$\tilde{\gamma}_k(s', s) = \tilde{p}(u_k) + \tilde{p}(y_k|c_k), \tag{3}$$

where the dataword and codeword pair u_k/c_k correspond to the transition from s' to s, and $\tilde{p}(u_k)$ is the a-priori probability for the dataword u_k. Its value is nonzero if there is a valid transition otherwise it is zero. Turbo encoder consists of two convolutional codes connected in parallel and an interleaver is placed in front of the second convolutional encoder. During the turbo decoding operation, one decoder computes the extrinsic information probability for information bits, and passed to the second decoder. Second decoder uses the extrinsic information probability supplied by first decoder as the a-priori probability of the information word for the second decoder and so on. This scheme iteratively runs for a sufficiently number of times until all the dataword probabilities converged to their ultimate values. For a turbo code with rate $1/2$ component convolutional codes and employing log-MAP at decoders, the extrinsic probability supplied from one decoder to another is given as:

$$L_e(u_k) = ln\sum_{u_k}(e^{\tilde{\alpha}_{k-1}(s')} + e^{\tilde{\gamma}_{ke}(s',s)} + e^{\tilde{\beta}_k(s)})$$

where $\tilde{\gamma}_{ke}(s', s) = \frac{p_k y_k^p}{\sigma^2}$, in which p_k are parity bits, and y_k^p is the received parity bit. For BPSK modulated binary data, absolute value of $\tilde{\gamma}_{ke}(s', s)$ is $|\tilde{\gamma}_{ke}(s', s)| = \frac{|y_k^p|}{\sigma^2}$. Use of negative SNR offset at the receiver means accepting noise power more than its actual value. In other words, using larger variance of noise than its actual value, i.e., $|\tilde{\gamma}_{ke}(s', s)| = \frac{|y_k^p|}{\sigma'^2}$ where $\sigma'^2 > \sigma^2$.

3. Simulation

Turbo codes and serially concatenated convolutional codes are used for testing the effects of *SNR* offset on code performance. We used $(1, 5/7)_{octal}$ recursive systematic convolutional code for all the constituent codes used in turbo and serially concatenated convolutional codes. Trellis termination bits are added by the constituent codes. The code rate for turbo code is $\sim 1/3$ and it is $\sim 1/4$ for serially concatenated convolutional code. The frame lengths are chosen as 256, and 1024. The *S-random* interleavers [1] are used for simulations. The *S* parameters are taken as 10, and 20 for turbo and serially concatenated convolutional codes respectively. The encoded bits are *BPSK* modulated and passed through an additive white Gaussian noise (AWGN) channel with double-sided noise power spectral density $\frac{N_0}{2}$, i.e., noise variance is $\sigma^2 = \frac{N_0}{2}$. *Log-MAP* soft decoding algorithm is used to iteratively decode the concatenated codes. We used the stopping rule in [5]. At each iteration the decoded frame is checked for being a codeword or not, if it is a codeword, decoding stops. Otherwise, decoding continues as long as the decoded frame is not a codeword up to a definite number of iterations. The maximum number of iterations is limited to 12. Simulations are executed until 100 erroneously decoded frames are received. The average number of iterations are considered. Our concern in this letter is mainly on the effects of positive *SNR* offset on average iteration number and code performance.

4. Results

We simulate both turbo codes and serially concatenated convolutional codes. Turbo code simulation results are depicted in Figs. 1 and 2 for an interleaver size of 256, and in Figs. 3 and 4 for an interleaver size of 1024.

It is obvious from Fig. 1 that at low E_b/N_0 values, positive offset increases the BER performance slightly.

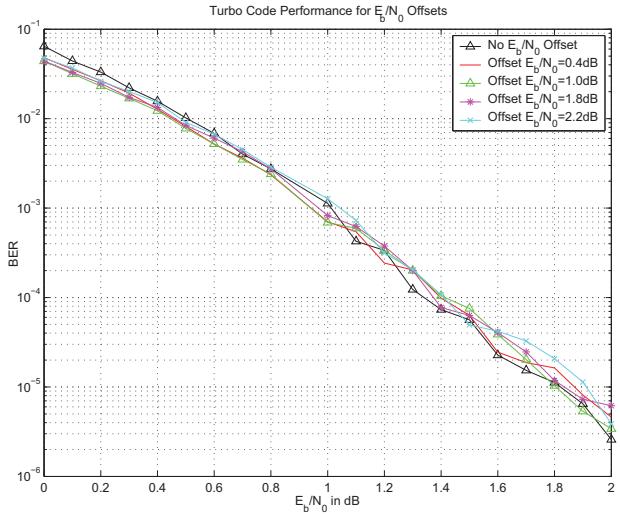

Figure 1: Turbo Code Performance for Positive E_b/N_0 OffSets. Interleaver Size=256.

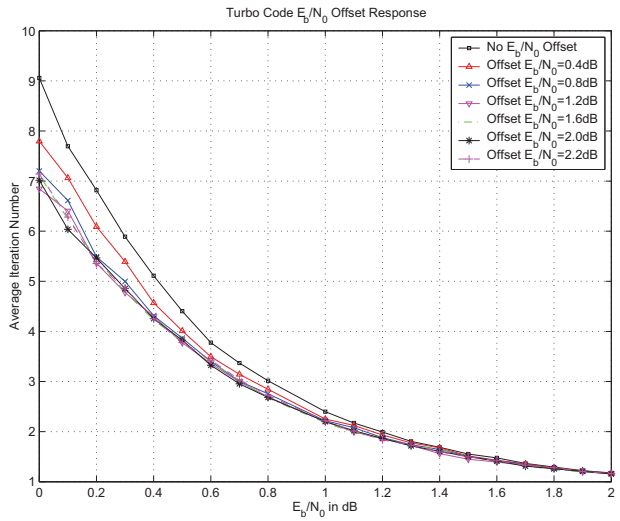

Figure 2: Turbo Code Average Iteration Number for Positive E_b/N_0 OffSets. Interleaver Size=256.

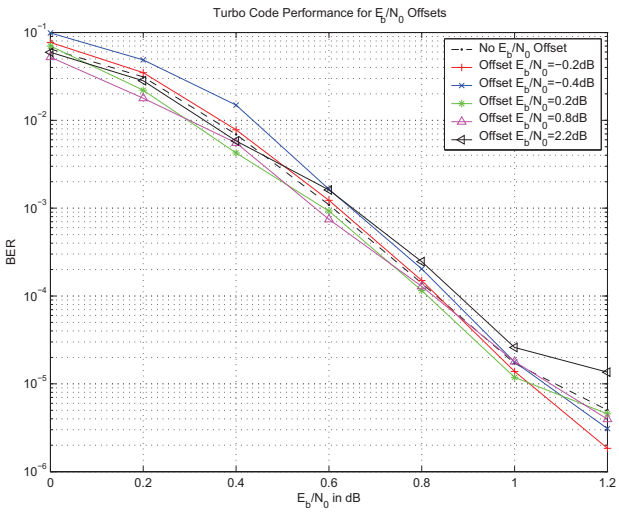

Figure 3: Turbo Code Performance for E_b/N_0 OffSets. Interleaver Size=1024.

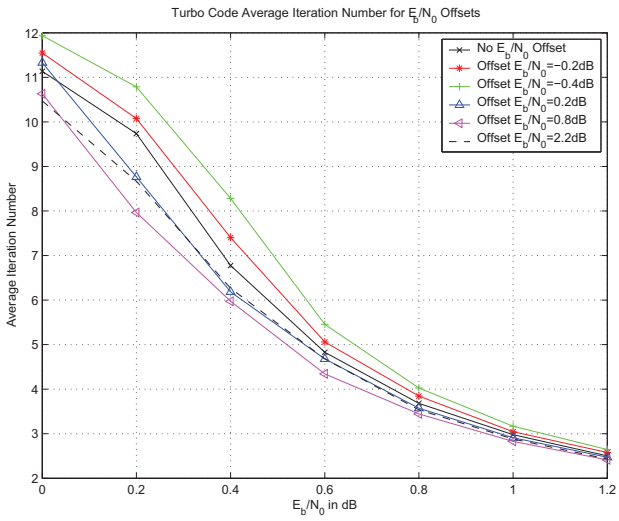

Figure 4: Turbo Code Average Iteration Number for E_b/N_0 OffSets. Interleaver Size=1024.

This increase is almost negligible. For moderate-to-high E_b/N_0 values, positive E_b/N_0 offset increases has little negative effect. If we sum up, it can be said that moderate positive E_b/N_0 offset does not effect the performance of the turbo code. The effect of the positive E_b/N_0 offset on iteration number is depicted in Fig. 2. It is clear from the Fig. 2 that positive offset reduces the average iteration number. The reduction in average iteration number at low E_b/N_0 ranges is enormous. However, as E_b/N_0 increases the amount of reduction in average iteration number decreases. In addition, as the offset increases more reduction is achieved, however too much offset increases the iteration number although it is still less than the original one. For an interleaver size of 1024 similar results are obtained for positive *SNR* offsets in Figs. 3 and 4. In addition, the effects of negative offset on turbo code are also investigated in Figs. 3 and 4 where it is seen that small amount of negative offset has negligible negative effect on BER performance of the turbo code at low *SNR* regions. It has slight positive effect on BER at high *SNR* regions. However, the average iteration number always increases even for little amount of negative *SNR* offsets in every *SNR* regions.

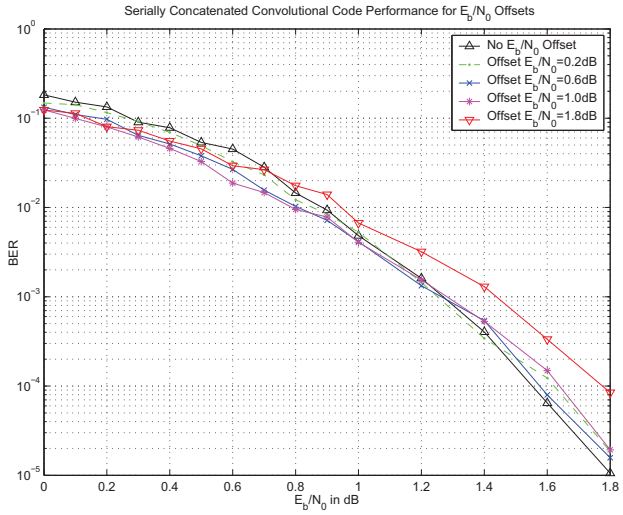

Figure 5: Serially Concatenated Convolutional Code Performance for Positive E_b/N_0 OffSets. Interleaver Size=256.

Simulations results for SCCC for an interleaver size of 256 are depicted in Figs. 5, and 6. It is obvious from the Figs. 5, and 6 that iteration number is reduced for all offset values for all *SNR* regions. However, code performance degrades slightly for comparatively large offset values at high *SNR* regions.

The consequences of the negative offsets for large *SNR* values are shown in Figs. 7, and 8. It is seen that for moderate negative offsets code performance in terms of BER stays almost the same whereas average iteration number always increases.

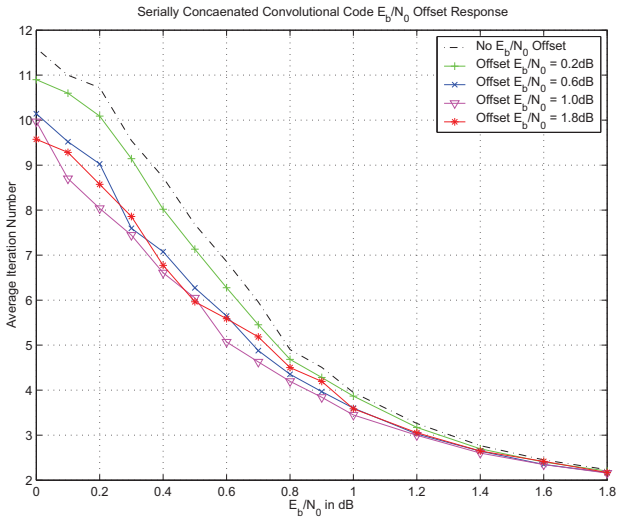

Figure 6: Serially Concatenated Convolutional Code Average Iteration Number for Positive E_b/N_0 OffSets. Interleaver Size=256.

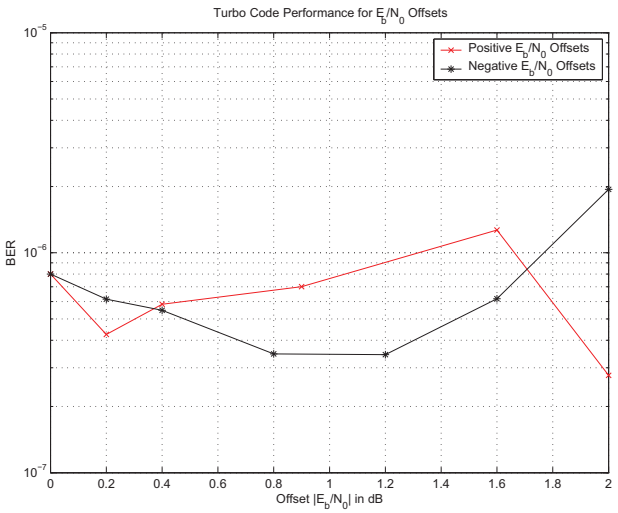

Figure 7: Turbo Code Performance at E_b/N_0=1.5dB for $|E_b/N_0|$ OffSets. Interleaver Size=1024.

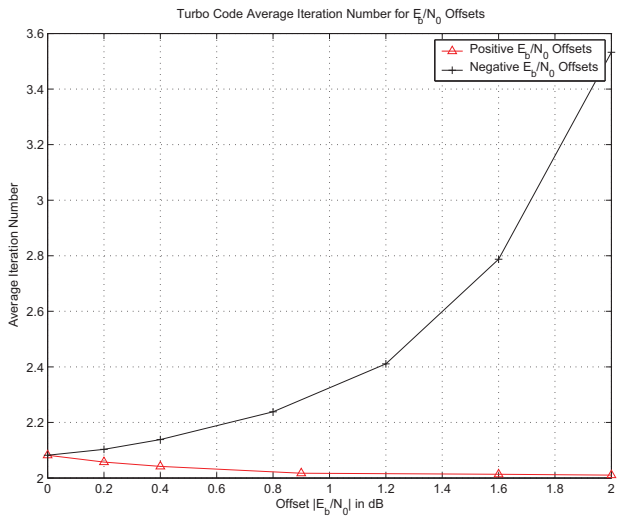

Figure 8: Turbo Code Average Iteration Number at E_b/N_0=1.5dB for $|E_b/N_0|$ OffSets. Interleaver Size=1024.

5. Conclusion

The effects of *SNR* offsets on turbo and serially concatenated convolutional code performance and average iteration number are investigated. It is found that, at low *SNR* regions giving a small amount of positive offset reduces the average iteration number enormously. At high *SNR* regions giving large amount of positive offset has negative effect on the code performance, however average iteration number is still less compared to the scenario where no offset is applied. At very high *SNR* regions, the code is robust to both negative and positive offsets, i.e., code performance stays almost the same with moderate offsets on *SNR*. However, for negative offsets it is seen that although code performance stays the same, average iteration number increases. Hence it is concluded that, at the receiver side a moderate amount of positive offset to the estimated *SNR* should be added in order to both reduce the average iteration number and decrease the possibility of underestimated *SNR*.

6. References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbocodes," in Proc. ICC93, Geneva, Switzerland, May 1993, pp. 10641070.

[2] O. Gazi and A. O. Yilmaz, Fast decodable turbo codes," IEEE Commun. Lett., vol. 11, no. 2, pp. 173175, 2007.

[3] D. Lee and I. Park, "A low complexity stopping criterion for iterative turbo decoding," IEICE Trans. Commun., vol. 88, no. 1, pp. 399401, 2005.

[4] F. Zhai and I. Fair, "Techniques for early stopping and error detection in turbo decoding," IEEE Trans. Commun., vol. 51, no. 10, pp. 16171623, 2003.

[5] A. O. Yilmaz and W. E. Stark, "Application of gaussian approximation to iterative decoding with finite blocklenghts," in IEEE Vehicular Tech. Conf., vol. 3, 2002, pp. 17631767.

[6] T. A. Summers and S. G. Wilson, "Snr mismatch and online estimation in turbo decoding," IEEE Trans. Commun., vol. 46, no. 4, pp. 421423, 1998.

[7] M. A. Khaligni, "Effect of mismatched snr on the performance of log-map turbo detector," IEEE Trans. Veh. Technol., vol. 52, no. 5, pp. 13861397, 2003.