Simulation and Evaluation of a Rate $\frac{1}{3}$ Parallel Concatenated Convolutional Code for the BiAWGN channel

Pavan Kumar, EE12B010
Sourbh Bhadane, EE12B109

December 1, 2015

Abstract

A rate $\frac{1}{3}$ Parallel Concatenated Convolutional Code was simulated and evaluated for the BiAWGN channel for block lengths equal to $n = 128, 512$. The codes were evaluated and threshold was found using EXIT charts. The simulations were performed using a log BCJR decoder for each of the component codes.

1 Introduction

Turbo Codes result out of a concatenation of two convolutional encoders (typically recursive systematic encoders (RSC)). Turbo codes were the first practical codes to closely approach channel capacity[1]. In this project, we focus on a specific class of turbo codes, the Parallel Concatenated Convolutional Codes. The turbo decoder consists of two soft decoders that iteratively exchange extrinsic information about the input bits back and forth. The two soft decoders used in this case are the log BCJR decoders. Turbo codes are evaluated using EXIT (extrinsic-information transfer) charts that are a graphical tool for estimating the decoding thresholds of turbo code ensembles. EXIT charts plot the mutual information for the extrinsic information coming out of the decoder versus the mutual information for the apriori extrinsic information. For each component code, there is a different extrinsic information curve.

2 Methodology

2.1 Log BCJR Decoder

A bit wise log MAP BCJR decoder was used as the decoder for each of the component codes. The corresponding trellis was constructed from the generator matrix of the component code. Following implementation issues were faced in the decoder

- For the turbo decoder, as the iteration count increases, the apriori probabilities become smaller, the log values get close to $-\infty$. A lower cap of -10 for block length 128 and -15 for block length 512 is put so as to avoid blowing up of LLRs.

- To take care of unterminated transmission the initialization of $\widetilde{\beta}$ is such that all the states have equal probability ($\log(0.25)$)

2.2 EXIT Charts

The decoding threshold for PCCC was obtained using EXIT charts. For a particular value of $\frac{E_b}{N_0}$, the channel output is obtained by

$$y = 1 + n$$

where n is a zero mean Gaussian random variable with variance $\sigma^2 = (2R * \frac{E_b}{N_0})^{-1}$.

Linearly spaced apriori extrinsic information values, I_A are taken between 0 and 1. These values are converted to the appropriate σ_A values using the conversions given in the Appendix. These σ_A values are used to generate the apriori LLRs for the log BCJR decoder.

$$A = \mu_A x_i + z_i$$

where z_i is a zero mean Gaussian random variable with variance σ_A^2 and $\mu_A = \frac{\sigma_A^2}{2}$

For a vector length of 51200, the log BCJR decoder is executed with A as the apriori LLRs and the channel LLRs obtained as $\frac{2y}{\sigma^2}$.

The extrinsic information, E is obtained by

$$E = L - A - R^{(u)}$$

The conditional pdfs are approximated using a histogram over $[-\max|E|, \max|E|]$ with 10 bins. Small number of bins have been used so that the probability values don't become too low. Since, the all ones vector is used for the simulation, the conditional pdf $p(E \mid x = -1)$ are computed using the fact that the probability distribution of E is symmetric $p(E \mid x = 1) = p(-E \mid x = -1)$. Using the conditional pdfs, I_E values are computed using the equations given in the Appendix.

The threshold of the code is that value of $\frac{E_b}{N_0}$ for which the extrinsic information transfer curves of the component codes just touch each other. The threshold σ that we obtained for the given code is 1.188

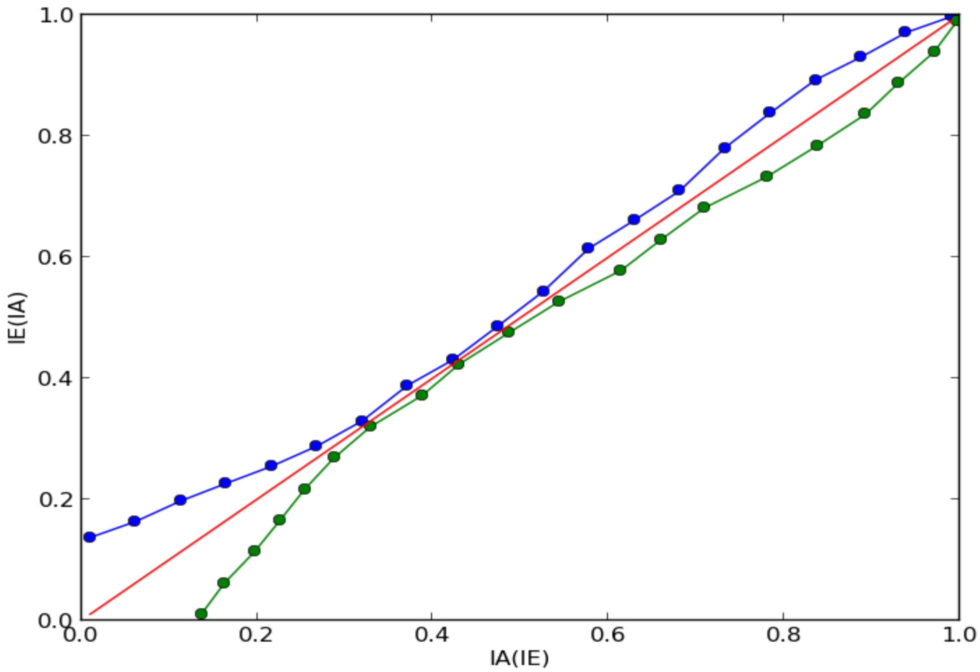

Figure 1: EXIT chart for $\sigma = 1.188$

2.3 Simulation Setup for Turbo Decoder

For the simulation, 7 values of $\frac{E_b}{N_0}$ from 0 to $6dB$ are considered. For each value of $\frac{E_b}{N_0}$, the all ones codeword is sent (all zeros under the BPSK mapping). The bits are divided into blocks of 128 or 512 The number of errors correspond to the total number of ones in the resulting output bits. This number divided by the total number of bits sent gives us the BER for that value of Eb/N0 To analyze eect of number of iterations, the maximum number of iterations was taken to be 5,10,15,20. To analyze coding gain for the lowest BER simulated, the n = 128 code was compared with the uncoded case.

3 Results

3.1 Effects of number of iterations

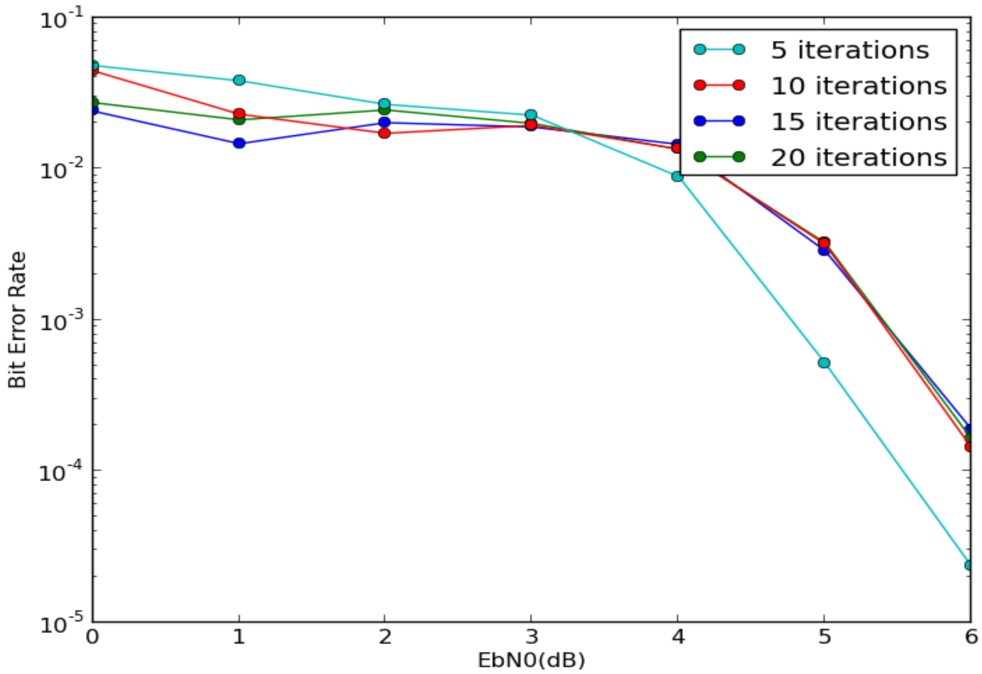

Figure 2: BER vs $\frac{E_b}{N_0}$ for number of iterations equal to 5, 10, 15, 20

As can be seen from the above plot, the more the number of iterations, faster is the error floor achieved. Some discrepancies in the plots have been addressed in the Conclusions and Scope for improvement section.

3.2 Effects of block length

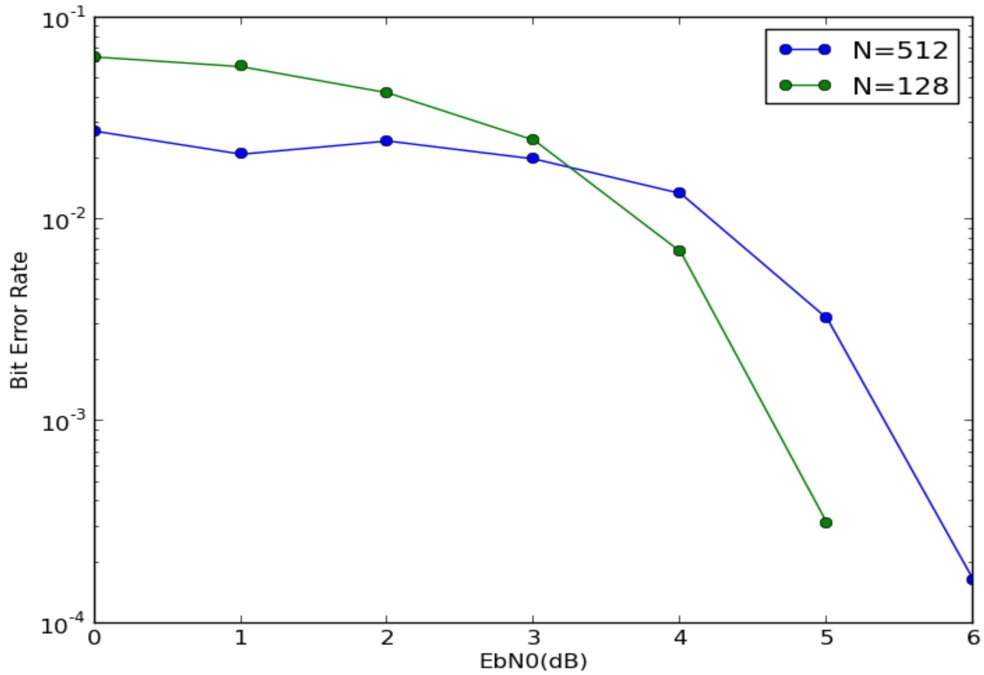

Figure 3: BER vs $\frac{E_b}{N_0}$ for block lengths equal to 128, 512

As can be seen from the plots, the $N = 512$ case attains the error floor quicker than $N = 128$, this is primarily because of the increase in number of low weight codewords for the larger block length

3.3 Coding Gain at lowest BER simulated

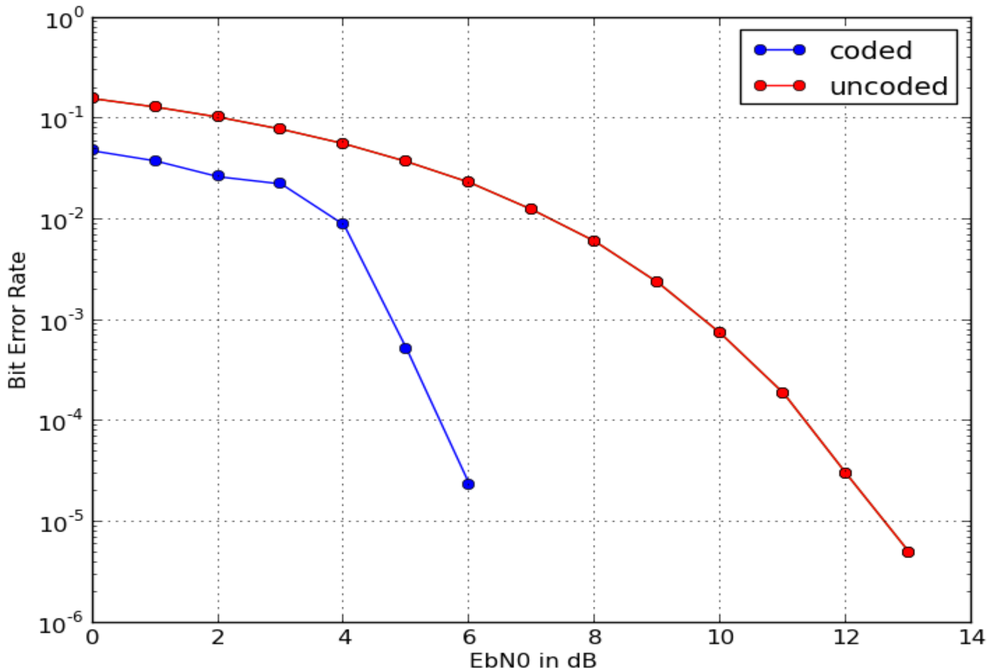

Figure 4: BER vs $\frac{E_b}{N_0}$ comparing coded versus uncoded

As can be seen in the above figure, a 6dB coding gain is obtained at the lowest BER.

4 Conclusions and Scope for Improvement

A rate $\frac{1}{3}$ PCCC for the BiAWGN channel has been simulated and evaluated using EXIT charts. As can be seen from the plots, the simulated codes do not come very close to Shannon's limit. Following are a few reasons

- The lower threshold for the log of the apriori probabilities for different values of $\frac{E_b}{N_0}$ were not changed. The threshold was not changed even when the number of iterations were varied. For more number of iterations, the threshold should be decreased as the LLRs keep on increasing as number of iterations increase.
- The initialization of the alpha values can be modified so as to give all states equal probability for iterations other than the first.

5 Appendix

5.1 EXIT Charts

The sigma values are computed from the I_A values using the expressions below [2]

$$J^{-1}(I) = \begin{cases} 1.09542I^2 + 0.214217I + 2.33727\sqrt{I} & 0 \le I < 0.3646 \\ -0.70692\ln[0.386013(1-I)] + 1.75017I & 0.3646 \le I \le 1 \end{cases}$$

Given the conditional pdfs, I_E is computed using the following expression [3] [4]

$$I_E = \frac{1}{2}\int_{e \in E} \sum_{x \in \pm 1} p(e \mid x) \log_2 \frac{2p(e \mid x)}{p(e \mid x=1) + p(e \mid x=-1)} de$$

In practice, however the above expression is implemented by summing

$$\sum_{x \in \pm 1} p(e \mid x) \log_2 \frac{2p(e \mid x)}{p(e \mid x=1) + p(e \mid x=-1)}$$

over all bins

References

[1] Claude Berrou, Alain Glavieux, and Punya Thitimajshima. Near shannon limit error-correcting coding and decoding. pages 1064–1070, 1993.

[2] Stephan Ten Brink, Gerhard Kramer, and Alexei Ashikhmin. Design of low-density parity-check codes for modulation and detection. *Communications, IEEE Transactions on*, 52(4):670–678, 2004.

[3] William E. Ryan and Shu Lin. *Channel codes. Classical and modern.* Cambridge: Cambridge University Press, 2009.

[4] Sarah J Johnson. *Iterative error correction: turbo, low-density parity-check and repeat-accumulate codes.* Cambridge University Press, 2010.