Rate 1/2 Low-Density Parity Check Codes for Moderate Block Lengths for a BI-AWGN channel

Pavan Kumar, EE12B010
Sourbh Bhadane, EE12B109

November 6, 2015

Abstract

Rate 1/2 Low-Density Parity Check (LDPC) codes for the BI-AWGN channel were designed, constructed and analyzed for block lengths equal to $n = 1000, 2000$ and 4000. The codes were designed using EXIT charts and the best code was obtained by evaluation using discretized density evolution. Parity check matrices for different lengths were obtained using the random constructions of Mackay. To analyze performance, a decoder was implemented using the log-likelihood version of the sum-product algorithm (SPA)

1 Introduction

LDPC codes are linear block codes built from sparse bipartite graphs called Tanner graphs. [1]The degrees of the nodes in the graphs are optimally chosen to give the best performance. The sparsity of the Tanner graph is exploited for formulating efficient algorithms for iterative message passing decoding of LDPC codes. Capacity approaching LDPC codes that beat turbo codes have been designed for large block lengths [2]. Discretized density evolution has been used to get to within $0.0045dB$ of the Shannon Limit at a bit error rate of 10^{-6} for large block lengths [3]. In this project, we obtain optimal degree distributions and construct sparse parity check matrices for moderate block length ($n = 1000, 2000, 4000$) LDPC codes for a BI-AWGN channel. We analyze the performance of these codes by simulations done on a BI-AWGN channel using the log-likelihood version of the Sum-Product algorithm. [4]

2 Methodology

2.1 Design and Density Evolution

2.1.1 EXIT Charts

Using EXIT charts we find an optimum $\lambda(x)$ by assuming a $\rho(x)$ for the rate $1/2$ code.

We assume and fix $\rho(x)$ and solve a linear programming problem to obtain an optimum $\lambda(x)$ which gives a rate $1/2$ code. As per the problem we assume only just two adjacent non-zero degrees in $\rho(x)$ and restrict degrees in $\lambda(x)$ to 25.

We have $\rho(x) = \rho_j x^{j-1} + (1-\rho_j)x^j$ and $\lambda(x) = \sum_{i=2}^{25} \lambda_i x^i$ where λ_i's are to be determined. Our objective function for LP is

$$\text{maximize} : \sum_{i=2}^{25} \lambda_i / i$$

Since λ_i's from degree distribution $\lambda(x)$,we have

$$\sum_{i}^{l_{max}} \lambda_i = 1$$

and for each λ_i,we have

$$0 \leq \lambda_i \leq 1, \forall i$$

The information exchange between check nodes and bit nodes is modelled and plotted and we obtain a curve as below:

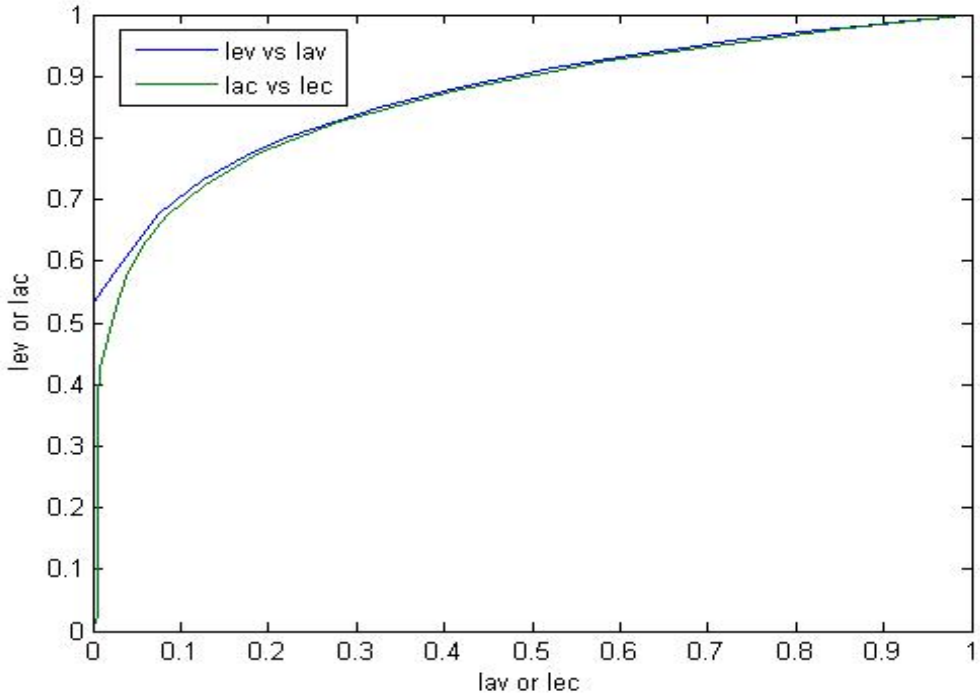

Figure 1: EXIT Chart

We ensure that convergence occurs so that finally perfect decoding happens. For that we see that the 2 graphs do not intersect with each other which is ensured by constraint:

$$I_{E,V}(I_A) \geq I_{E,C}^{-1}(I_A)$$

But this is continous which cannot be evaluated because of unavilability of closed form expression. So we use discrete values of $I_A$and form discrete set of equations which act as constraints for LP problem. Here 20 different values of I_A between 0 and 1 are used to get 20 constraints.

We need to compute $I_{E,V}(I_A)$and $I_{E,C}(I_A)$ for forming constraints at various I_A.We have,

$$I_{E,V}(I_A) = \sum_{i=2}^{25} \lambda_i I_{E,V}(i, I_A)$$

$$I_{E,C}(I_A) = \sum_{i=k}^{k+1} \rho_i I_{E,C}(i, I_A)$$

where k and k+1 are adjacent non-zero degrees in $\rho(x)$. The expressions for $I_{E,V}(i, I_A)$ and $I_{E,C}(i, I_A)$ in terms of $J(\sigma)$ and the approximations of $J(\sigma)$ used, are provided in the Appendix.

We compute $I_{E,C}(I_A)$ for 10,000 values of I_A and then iteratively search and find $I_{E,C}^{-1}(I_A)$.

With the above constriants and objective function which together form a linear programming problem,we can solve the LP obtained using GNU Linear Programming Kit (GLPK [5])

2.1.2 Density Evolution

Using density evolution we evaluate the threshold of LDPC codes whose degree profile is obtained from exit charts.

We start with the Shannon threshold for rate 1/2 and keep increasing the SNR/threshold till the probability of error falls below some threshold.

We proceed by doing discrete density evolution with 10 bit quantization. Assuming limits to be -20 and $+20$ and considering 1024 levels, we get step size as

$$\text{Step-Size } \Delta = \frac{Range}{Levels} = \frac{40}{1024} = 0.04$$

We use the following quantization function:

$$Q(m) = \begin{cases} \lceil m/\triangle + 0.5 \rceil \triangle & m \geq \triangle/2 \\ \lceil m/\triangle - 0.5 \rceil \triangle & m \leq -\triangle/2 \\ 0 & otherwise \end{cases}$$

where $\llcorner \lrcorner$ is the greatest integer function and $\ulcorner \urcorner$is the least integer function.

We have the bit node message probability update equation as:

$$P_l^{(v)} = P_0^{(c)} * \lambda_*(P_{l-1}^{(c)}))$$

where $P_l^{(v)}$ is probability out of bit node at the end of l^{th} iteration,$P_{l-1}^{(c)}$ is probability of message out of check node at the end of $(l-1)^{th}$iteration,$P_0^{(c)}$ is probability distribution received by a bit node from the channel and $*$indicates convolution.

As we send an all zero code word,$P_0^{(c)}$ is nothing but distribution of gaussian with mean as $\frac{2}{\sigma^2}$and variance as $\frac{4}{\sigma^2}$.

We have check node message probability update equation as:

$$P_l^{(c)} = \rho_{\boxplus}(P_l^{(v)})$$

where $P_l^{(c)}$is probability out of check node at the end of l^{th} iteration,$P_l^{(v)}$is probabiltiy of message out of bit node at the end of l^{th}iteration and $\boxplus$indicates box operation.

Box operation is defined as:

$$P_m[k] = \sum_{(i,j):k\triangle = B(i\triangle, j\triangle)} P_{m1}[i].P_{m2}[j]$$

where B is defined as:

$$B(m_1, m_2) = Q(2tanh^{-1}(tanh(m_1/2).tanh(m_2/2)))$$

We run this message passing for some maximum number of iterations(5000) and each time with a σ starting from Shannon's limit value of 0.9787 and keep decreasing till we get

$$\sum_{k<0} P_l^{(c)}[k] \leq c$$

where c is choosen as 10^{-5}as threshold for low probability whihc ensures that no more error is present.The value of σ where this happens becomes the threshold of our degree distribution used.

2.1.3 Optimal degree distribution

For obtaining an optimal degree distribution we run EXIT charts over a wide range of $\rho(x)$. and the obtained degree distributions are tested using density evolution for computing threshold.

The design rate of degree distribution is given by

$$\text{Rate} = 1 - \frac{\sum \frac{\rho_i}{i}}{\sum \frac{\lambda_i}{i}}$$

The degree distribution with rate close to 1/2 and threshold closest to shannon limit are choosen as output of design part.

For rate 1/2 code, Shannon's limit is $\frac{E_b}{N_0} = 0.187dB$or $\sigma = 0.9787$.

A $(3, 6)$ regular LDPC has threshold values as $\frac{E_b}{N_0} = 1.104dB$or $\sigma = 0.8806$.

But on simulating for various $\rho(x)$ we obtain the best irregular LDPC code as:

$$\lambda(x) = 0.230135x^1 + 0.189848x^2 + 0.126453x^5 + 0.18135x^6 + 0.272213x^{25}$$

$$\rho(x) = 0.532x^8 + 0.468x^9$$

with design rate as 0.4983 and threshold as $\frac{E_b}{N_0} = 0.21dB$or $\sigma = 0.976$ which is just 0.024dB away from Shannon's limit which is far better than $(3, 6)$ regular LDPC code.

2.2 Construction of Parity Check Matrix

Among the various existing methods for constructing parity check matrices for LDPC codes, we chose the random constructions of Mackay et al.
The optimal degree distributions obtained from the above method were converted to their corresponding bit node and check node degree sequences using the block length. Since, these don't turn out to be whole numbers, the degree sequences are rounded up so as to minimize deviation from the optimal degree distributions. These degree sequences are then checked for equality of edges. i.e

$$\sum iL_i = \sum jR_j$$

The row degree sequence is randomly fixed and each column of the parity check matrix is added and ones are filled so as to obey the row degree sequences and also ensuring that four cycles are not formed. However, using this method, for a few (approximately 5) columns, no ones can be filled without forming 4 cycles. In this case, instead of backtracking and starting again, a compromise is sought by filling in ones to invalidate the row degree sequence and as a result changing the row degree distributions. This is done so that convergence to a parity check matrix is sought. This effect is less prominent for larger block length ($n = 2000, 4000$).

2.3 SPA Decoder

The log-likelihood (LLR) version of the SPA decoder was used to analyze the performance of the LDPC code. LLRs are iteratively exchanged between the bit nodes and check nodes until either maximum number of iterations have occured or the decoder has converged to a codeword.
The input LLRs from the BI-AWGN channel output are computed as

$$L_i = \log \frac{p(y_i \mid 1)}{p(y_i \mid -1)}$$

Which for the BI-AWGN channel is

$$L_i = \frac{2y_i}{\sigma^2}$$

The Check Node and the Bit Node update equations have been given in the Appendix.
The maximum number of iterations was fixed to be 100. An analysis of the BER as number of iterations were varied is provided in the next section.
An implementation issue faced during simulations was of the blowing up of LLRs especially for channels with a high value of E_b/N_0. Under finite precision, the LLRs that blew up were taken to be a negative value and occassionally gave an output of all ones for an all zero codeword. To overcome this error a threshold of 15 was set on the value of LLRs.

2.4 Simulation Setup

For the simulation, 8 values of ${}^{E_b}/_{N_0}$ were considered from 0 to $6dB$. For each value of ${}^{E_b}/_{N_0}$, we sent the all zero codeword repeatedly, which becomes all one codeword under the BPSK mapping. A number of bits ranging from 10^5 to 10^7 were sent, where less number of bits were sent for lower values of ${}^{E_b}/_{N_0}$. These bits were divided into blocks of the corresponding block length and passed to the SPA decoder which runs for atmost 100 iterations.
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 ${}^{E_b}/_{N_0}$
To analyze effect of number of iterations, the maximum number of iterations was taken to be $20, 50, 80, 100$.
To analyze coding gain for the lowest BER simulated, the $n = 1000$ code was compared with the uncoded case.

3 Results

3.1 Effects of number of iterations

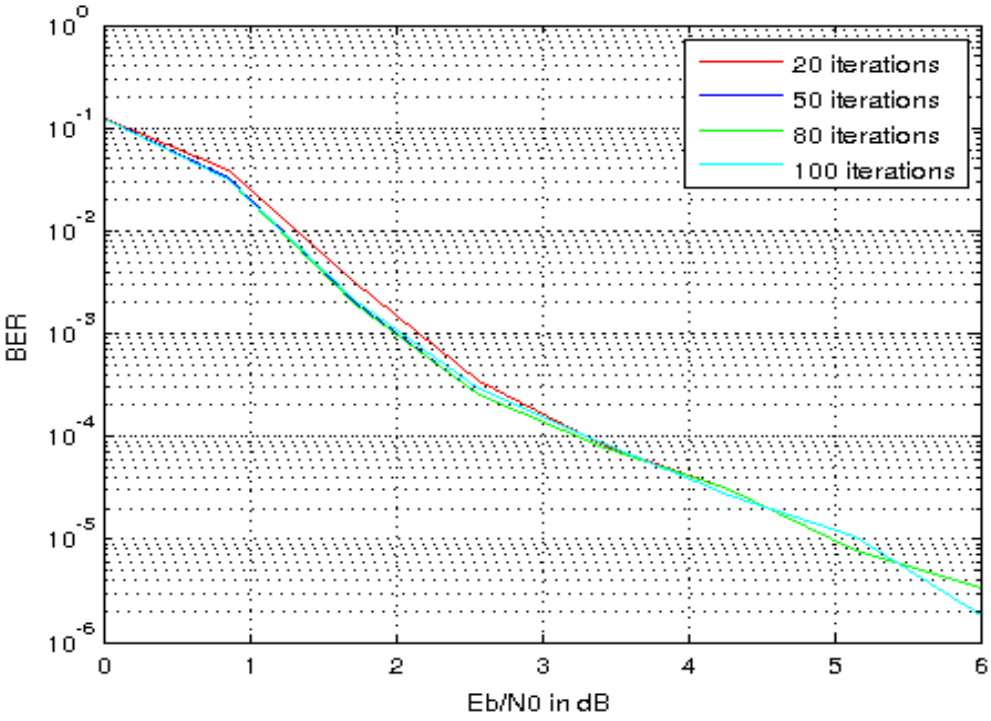

Figure 2: BER vs $^{E_b}/_{N_0}$ for different number of iterations of SPA

As can be seen in the above figure, for lower $^{E_b}/_{N_0}$, the error rate is high for 20 iterations and nearly the same for the other cases. For a higher value of $^{E_b}/_{N_0}$, we can see that the performance of the cases other than 100 iterations have an error floor, whereas the 100 iterations case continues to dip

3.2 Effects of block length

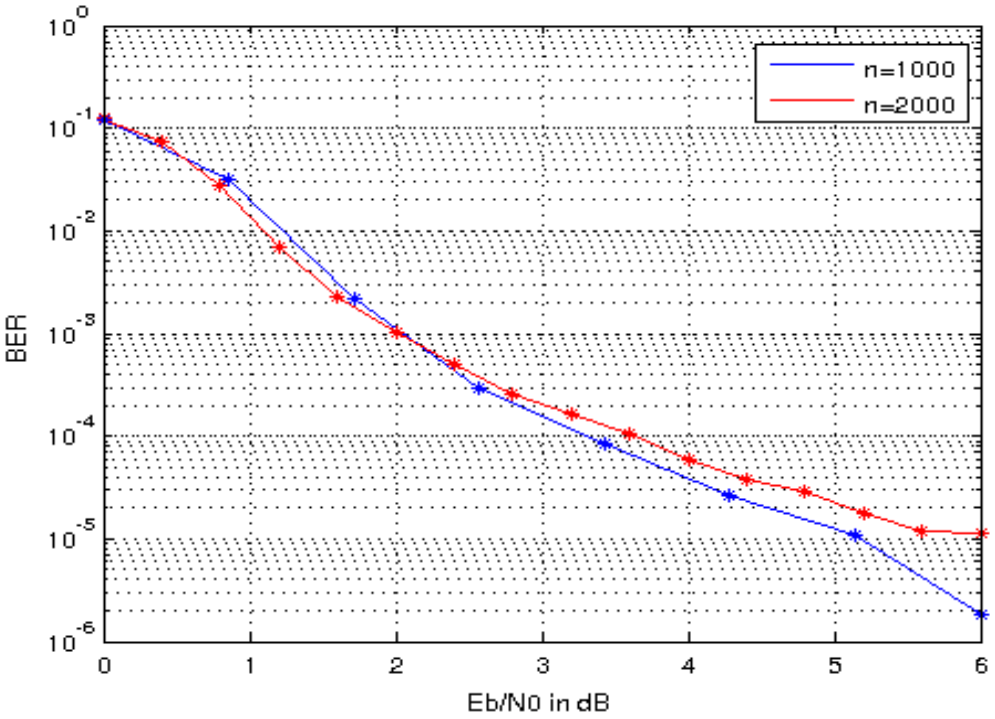

Figure 3: BER vs $^{E_b}/_{N_0}$ for different block lengths

As can be seen in the above figure, in the waterfall region, the $n = 2000$ is seen to perform better compared to the $n = 1000$ code. However, it attains the error floor quicker than the $n = 1000$ code. The reason is that for $n = 2000$ code, there are more number of weight two codewords compared to the $n = 1000$ codeword and hence possibly more number of trapping sets.

3.3 Coding Gain at lowest BER simulated

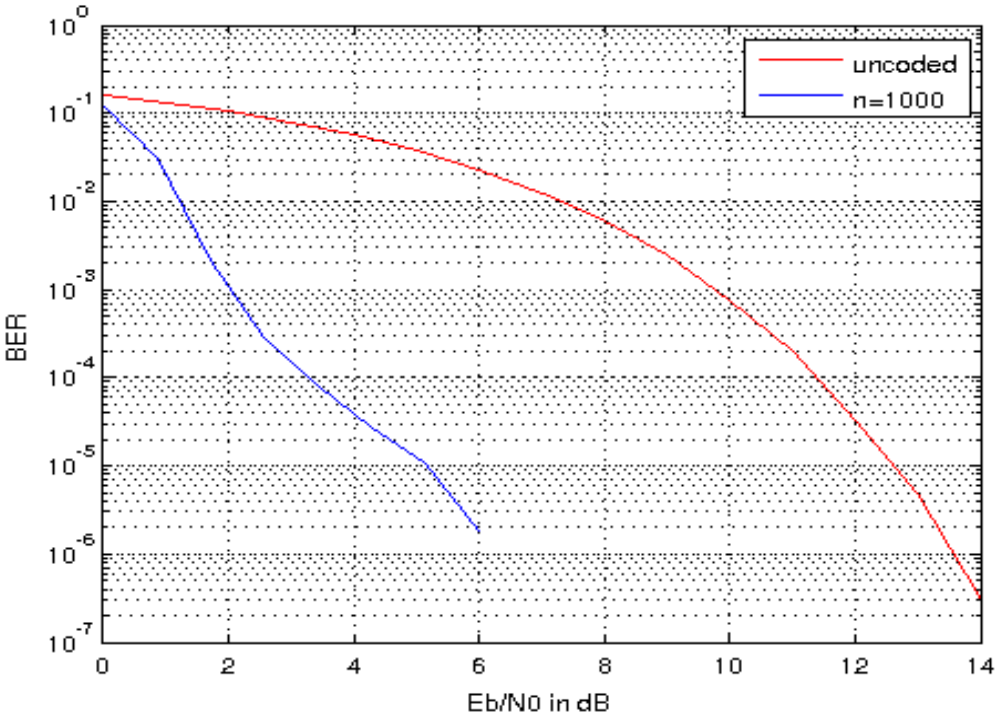

Figure 4: Coding Gain compared to uncoded scenario for BI-AWGN

As can be seen in the above figure, there is a coding gain of approximately $7.5dB$ obtained by using a LDPC code at BER just above 10^{-6}

3.4 Comparison with MacKay's codes

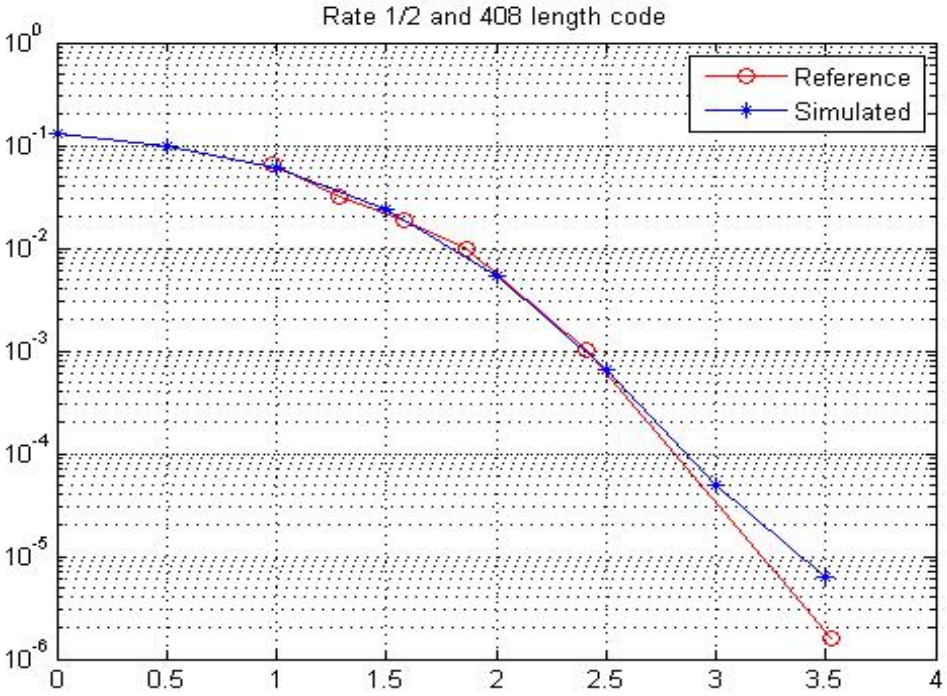

Figure 5: $n = 408$

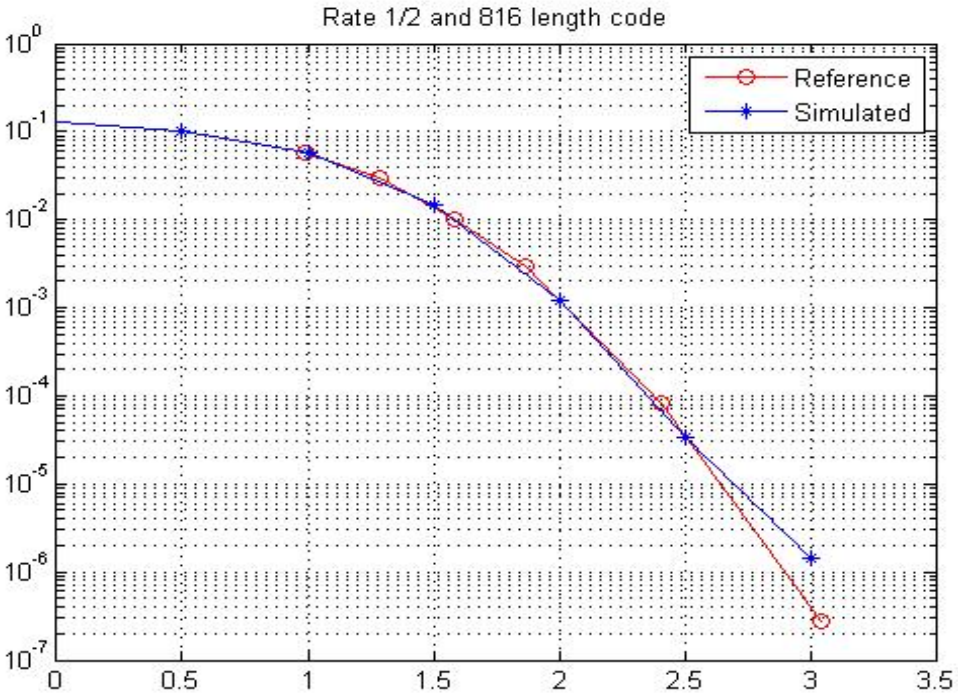

Figure 6: $n = 816$

4 Conclusions and Scope for Improvement

A rate $1/2$ LDPC code for the BI-AWGN channel has been designed using EXIT charts and the resulting degree distributions have been evaluated using discretized density evolution. A sparse parity check matrix without four cycles has been constructed using Mackay's constructions. For block lengths 1000,2000, the BER vs E_b/N_0 performance has been analyzed. As can be seen from the plots above, the codes designed do not come very close to the Shannon Limit. Following are a few reasons

- The design can be such that number of two-weight codewords is minimized further.
- The construction of the parity check matrix alters the resulting optimal degree distribution by some amount. A different construction method like PEG or bit filling must be tried out.
- A more efficient form of a decoder can be implemented instead of the LLR version of the SPA.

5 Appendix

5.1 SPA Decoder

Check Node update equations

$$L_{j\to i} = 2\,\text{arctanh}\,[\prod_{k\in N(j)\setminus i} \tanh\frac{L_{k\to j}}{2}]$$

Bit Node Update Equations

$$L_{i\to j} = \sum_{k\in N(i)\setminus j} L_{k\to i}$$

$$L_i = \sum_{k\in N(i)} L_{k\to i}$$

Stopping criterion For $j = 0, 1, 2...n-1$

$$\hat{v_j} = \begin{cases} 0, & \text{if } L_i < 0 \\ 1, & \text{otherwise} \end{cases}$$

These $\hat{v_j}$ are used to determine whether SPA converged to a codeword.

5.2 Density Evolution

We have,

$$I_{E,V}(i, I_A) = J(\sqrt{(i-1)(J^{-1}(I_A))^2 + \sigma_{ch}^2})$$

$$I_{E,C}(i, I_A) = 1 - J(\sqrt{(i-1)(J^{-1}(1-I_A))^2})$$

For these computations we need functions of J and J^{-1}whose approximate form has been given by [6] :

$$J(\sigma) \quad = \quad \begin{cases} -0.0421061\sigma^3 + 0.209252\sigma^2 - 0.00640081\sigma & 0 \le \sigma < 1.6363 \\ 0 \le \sigma < 1.63631 - \exp(0.00121491\sigma^3 - 0.142675\sigma^2 - 0.082054\sigma + 0.0549608) & 1.6363 \le \sigma < 10 \\ 1 & \sigma \ge 10 \end{cases}$$

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

References

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

[2] Thomas J Richardson, M Amin Shokrollahi, and Rüdiger L Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):619–637, 2001.

[3] Sae-Young Chung, G David Forney, Thomas J Richardson, and Rüdiger Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *Communications Letters, IEEE*, 5(2):58–60, 2001.

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

[5] GLPK (GNU linear programming kit), 2006.

[6] 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.