Anbar Journal of Engineering Science (AJES) **Vol. 16, No. 1 (2025), pp. 13 ~ 20**
P-ISSN: 1997-9428; E-ISSN: 2705-7440 DOI: http://doi.org/10.37649/aengs.2025.154788.1110

University of Anbar

Anbar Journal of Engineering Science
journal homepage: https://ajes.uoanbar.edu.iq/

Performance Measure of Turbo Code Using LLR Histogram

Mohammed AlMahamdy[a]
[a] *Department of Electrical Engineering, College of Engineering, University of Anbar, Ramadi, Iraq*
Email: mohammed.almahamdy@uoanbar.edu.iq ; ORCID: https://orcid.org/0000-0003-1691-0765

PAPER INFO

Paper history:
Received: 28/10/2024
Revised: 14/01/2025
Accepted: 10/02/2025

Keywords:
turbo code;
error correcting codes;
iterative decoding;
PCCC;

Copyright: ©2023 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY-4.0) license.
https://creativecommons.org/licenses/by/4.0/

ABSTRACT

Turbo codes have been deployed in many cutting-edge technologies because they can achieve very high coding gains. Turbo decoders deploy at least two Soft-Input-Soft-Out (SISO) decoders, which operate iteratively to incorporate their results to conclude the output. The soft outputs from the used constituent SISO decoders develop gradually along the iterations. This development is studied and analyzed in this work to understand the dynamics leading to the results. Histograms statistically group and visualize the soft results for further analysis and study. A method is proposed to evaluate the decoding performance based on the density of the values of the soft outputs within the histogram. Results show that the performance is inversely related to the ratio of the values of the soft outputs within the near-zero bins within the histogram. The proposed method can be deployed at the decoder to provide an early indication of the reception and whether it has the potential to be correctly decoded or not. This early decision can save the decoding resources.

1.Introduction

Turbo codes are error-correcting codes that have revolutionized digital communication systems. They offer exceptional performance in terms of error correction and data throughput. Turbo codes can achieve extremely low bit error rates, even in challenging communication environments, making them ideal for satellite communications, mobile phones, and digital video broadcasting applications.

The performance of turbo codes has been the subject of many works since its inception. The superb performance of turbo codes comes at the price of requiring high processing power and considerable latency [1]. For example, numerous efforts have been devoted to optimizing and speeding up the decoding [2-6]. In [7] Turbo codes have been studied from the digital signal processing perspective and show how the number of iterations affects the mean power of the noise in the decoded message. The performance of turbo codes has been studied for: OFDM [8], MIMO-based DVB-T2 [9], 5G [10], 6G [11, 12], and many other technologies like [13, 14]. Soft output Viterbi algorithm has been studied for turbo codes in [15]. The parity-check matrix of Turbo codes is improved in [16]. Turbo coded-spatial modulation scheme based on code-matched interleaved is studied for multiple input multiple output antenna systems in [17]. Different structures have been studied in [18] to measure the performance of turbo codes. Turbo-based encryption and coding scheme is proposed in [19] to improve reliability and security for the MIMO-OFDM wireless communication systems. In [20], the turbo code is studied under channels with inter-symbol interference.

2. System Structure

2.1. Encoding

Turbo codes are also called Parallel Concatenated Convolutional Codes (PCCC), as they are constructed by concatenating two (or more) convolutional codes separated by an interleaver in parallel. The interleaved permutes the data before encoding so that the two generated codes are statistically uncorrelated. This helps spread errors and improves the decoding process. The conventional configuration for turbo codes

[*] Corresponding author: Mohammed AlMahamdy; mohammed.almahamdy@uoanbar.edu.iq; 07802673166

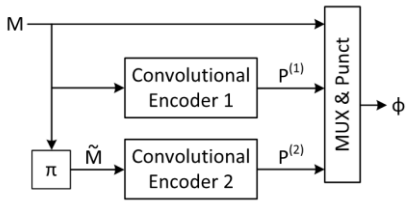

Figure 1. The Turbo Encoder.

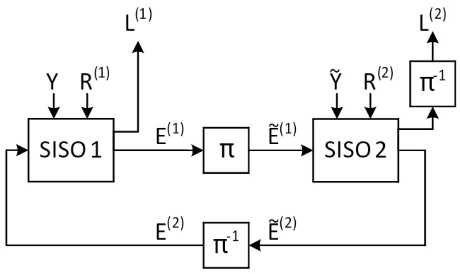

Figure 2. The Turbo Decoder.

includes two identical systematic convolutional encoders and an interleaved, as depicted in Figure 1. The input to one encoder is the original data vector $\mathbf{M}$, and the input to the other one is the interleaved data vector $\widetilde{\mathbf{M}}$. Accordingly, the i^{th} encoder produces the parity bits $\mathbf{P}^{(i)}$. The transmission is constructed by multiplexing the original data, the parity bits from both encoders, and the termination bits. The latter is optional to be included in the transmission. However, these bits improve the decoding [21, 22]. Data puncturing is possible to reduce the transmission rate.

2.2. Decoding

Turbo codes employ algorithms for iterative decoding, where the decoder repeatedly processes the received data using updated information from previous iterations to improve its decoding performance. These algorithms and the parallel concatenation structure have enabled turbo codes to achieve remarkable results in various applications.

The received signal is first demultiplexed to its principal components: the received data bits. $\mathbf{Y}$, the received parity bits for the i^{th} decoder $\mathbf{R}^{(i)}$, and possibly the termination bits. These vectors are used to construct the input to each decoder. The conventional turbo receiver, depicted in Figure 2, involves two SISO decoders. They run the BCJR algorithm [23], which is a maximum-likelihood-decoder. The decoding process of the convolutional code is performed with the aid of the a priori information passed from other decoders. Since the original data is assumed to be independent and identically distributed, the initial values for this a priori information are set to zeros. After each decoding process, the deployed SISO decoders generate an LLR (Log-likelihood-Ratio). It is a numeric value per the j^{th} bit that is considered the soft output L_j. The absolute value this measure reflects the level of 'confidence' the decoder has about decoding the j^{th} bit. The higher the absolute value the higher the reliability of the decoding of that bit.

The extrinsic information from the i^{th} SISO decoder per the j^{th} bit is denoted here as $E_j^{(i)}$. This information is passed to the other SISO decoder to be considered a priori for its next decoding phase. For turbo decoders comprising two SISO decoders, a decoding iteration is completed after both decoders generate their soft outputs. We denote the vector. $\mathbf{E}_k^{(i)}$ as the vector of individual extrinsic information values generated by the i^{th} SISO decoder at the iteration index k. It is extracted from the vector of LLRs at the k^{th} iteration $\mathbf{L}_k^{(i)}$ after subtracting the received values for the bits and the used extrinsic information from the previous iteration. For example, after completing the decoding process performed by the 1st SISO decoder at the k^{th} iteration, the extrinsic information is computed as in equation (1). The vector $\mathbf{E}_k^{(1)}$ must be interleaved (denoted as $\tilde{\mathbf{E}}_k^{(1)}$) before it can be used as a priori by the second SISO decoder. $\tilde{\mathbf{E}}_k^{(1)}$ is used to decode the inter-leaved data set $\widetilde{\mathbf{Y}}$ after is combined with $\mathbf{P}^{(2)}$. The generated LLRs $\mathbf{L}_k^{(2)}$ are used to extract $\mathbf{E}_k^{(2)}$ as in equation (2). The resulted $\mathbf{E}_k^{(2)}$ is passed back to the 1st decoder after it has been deinterleaved.

$$\mathbf{E}_k^{(1)} = \mathbf{L}_k^{(1)} - \mathbf{Y} - \mathbf{E}_{k-1}^{(2)} \tag{1}$$

$$\mathbf{E}_\text{k}^{(2)} = \mathbf{L}_\text{k}^{(2)} - \widetilde{\mathbf{Y}} - \tilde{\mathbf{E}}_\text{k}^{(1)} \tag{2}$$

The process is repeated iteratively, with the decoders exchanging extrinsic information until a "satisfactory" decoding result is achieved. One way of marking the end of this iterative decoding is reaching the "convergence". This point is defined as the situation when the turbo decoder generates the same hard output at two consecutive iterations. At this point, it is assumed that no further improvement is expected where the decoder keeps iterating. Terminating the iterative loop at this point is beneficial for preserving resources. Such a decision is determined by what is called an Early Stopping rule. In some cases, especially at low SNRs, the turbo decoder is unlikely to reach convergence, as the reception is mostly severely corrupted by noise. In such cases, the decoding process keeps iterating up to the maximum number of iterations allowed for the selected configuration.

3. Simulation

For the simulation deployed for this work, we consider the configuration of LTE turbo codes, consisting of two rate-half-RSC encoders and a quadratic polynomial interleaved. The block size is set to 512 bits. The encoders have feedforward and feedback polynomials of $(15)_8$ and $(13)_8$ [24]. We use the Max* Log-MAP algorithm [25] for the

SISO decoders. The decoder iterates up to 12 times. The early stopping rule [26] is used to facilitate the simulation. The simulation of this work was implemented in MATLAB.

4. Results and Discussion

The speed of LLR development along the iterations is mainly affected by the level of corruption the received data endures and the location of these corrupted bits. At high SNRs, the decoder is likely to reach the convergence fast and the reception to be correctly decoded. The opposite is true: severely corrupted blocks are unlikely to reach convergence and will likely be decoded with errors. In this work, we perform the simulation of running the decoding process at different SNRs, ranging from $E_b/N_0 = 0$dB up to 2dB, to cover the range from a very- low to a high SNR. To compare the turbo code's performance at the selected SNRs, we choose to simulate 1,000 blocks. These blocks are generated randomly as binary, coded/decoded into turbo code, and applied to randomly generated noise at the given SNR. After each iteration, the results of these blocks and the generated LLRs are studied here for analysis and discussion.

The generated LLRs by the second SISO per bit per iteration are stored in a 3-dimensional matrix. We can present these LLRs using a histogram, as in **Error! Reference source not found.** to Figure 7. This illustration helps visualize these random-like values into meaningful forms. For consistency of the comparison, we divide these histograms into 24 bins. The edges of these bins start from -60 to top 60 with step size 5. It is also important to draw these plots at the same scale. We draw the histogram of all the generated LLRs in each figure at the indicated iteration index per SNR.

For successful decoding, the absolute value of an individual LLR grows along the iteration. Convergence is considered when that value does not change the sign and/or value for further iterations. On the other hand, at the small SNRs (in this paper, they are 0 and 0.5 dB), most LLRs do not grow in value significantly as the iteration progresses. This is caused by the fact that almost all the decoded blocks contain many errors. Hence, the turbo decoder does not reach convergence for them. This is apparent **Error! Reference source not found.**, where the values of the LLRs are virtually unchanged along the iterations. The reason for this is that the reception is so severely corrupted with noise that the decoder cannot develop constructive soft outputs. This case also appears in **Error! Reference source not found.**. However, the decoder here can develop a few LLRs that are less corrupted by noise than the rest of the block. Although all the LLRs start from zero as initial values, the values of almost all these LLRs remain small and around zero along the iterations, which reflects unreliable decoding for that bit. This explains the low probability of error at small SNRs.

As the SNR increases, the number of errors at the decoding decreases. This can be shown by **Error! Reference source not found.** to Figure 7. The SNR increase increases the number of received blocks with manageable noise corruption. The growth of the LLRs within these blocks per iteration becomes recognizable. Ultimately, they reach convergence. Decoding these blocks results in a small number of error bits per block. In these figures, the values of the LLRs drastically increase as the iterative decoding progresses. This is evident by the fact that the key feature of the turbo decoder is that it regenerates new outputs after each iteration based on values gathered from the previous iteration. These figures show that most LLRs increase in value as the SNR increases and move away from zero. In the late iterations, most of the LLRs have converged. As the plots in **Error! Reference source not found.** to Figure 7 The generated LLRs for converging blocks grow in value along the iterations. At the start, the histogram of these LLRs is concentrated at and around zero. For successful decoding, the histogram of these values emerges into two groups (one at each sign) as the iteration index increases. At convergence, these groups are virtually isolated from each other. This is especially true for large SNRs, where the decoding will likely converge within a few iterations. On the other hand, the histogram of the results at low SNRs remains clustered around zero. Even when they start to group away from zero, they are not separate as many LLRs have absolute values less than ~ 10.

Most blocks reach convergence when the SNR is high for successful decoding. This means the number of blocks within the histogram bins around zero is small, or even zero. As **Error! Reference source not found.** to Figure 7 show, this is achieved early in terms of the number of performed iterations. At mid-range SNRs, this is also achieved but mostly after more iterations are completed [27, 28]. In the last case, where the SNR is low, most LLRs end with small values around zero. Successful decoding is associated with large values for the generated LLRs, and small to null density of LLRs around zero. In this work, we present the method for measuring the performance by counting the number of LLRs per the decoded block with values within the histogram bins around the zero. The edges of these two bins are -5 to 0 and 0 to 5. To visualize these cases, we plot the ratio of the number of LLRs at the bins around zero (-5 to 0 and 0 to 5) to the total number of LLRs per block (in this work it is 512) per the whole simulated blocks (here 1000 blocks) after the last iteration is completed (here it is after the 12th iteration). These ratios are plotted in Figure 8.

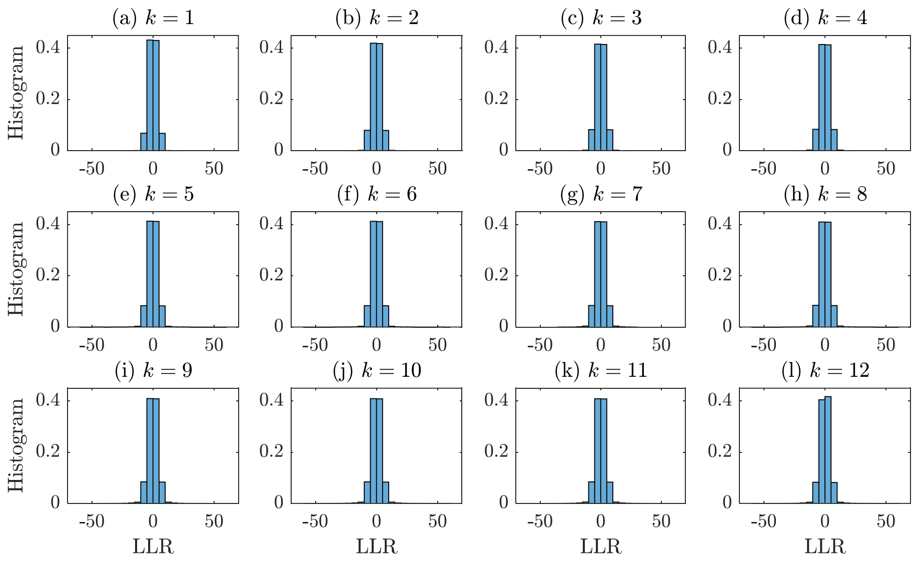

Figure 3. Histogram of all the generated L_j per iteration at $E_b/N_0 = 0$dB.

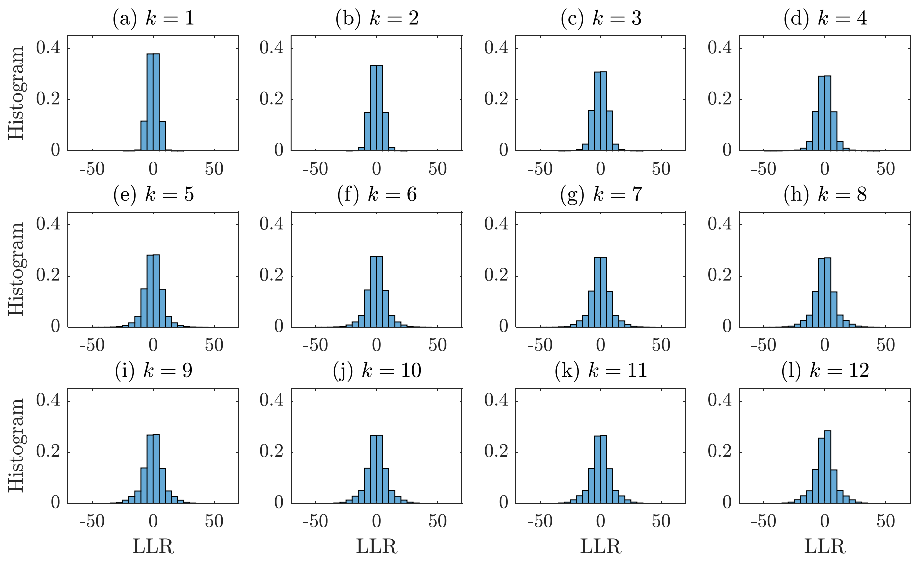

Figure 4. Histogram of all the generated L_j per iteration at $E_b/N_0 = 0.5$dB.

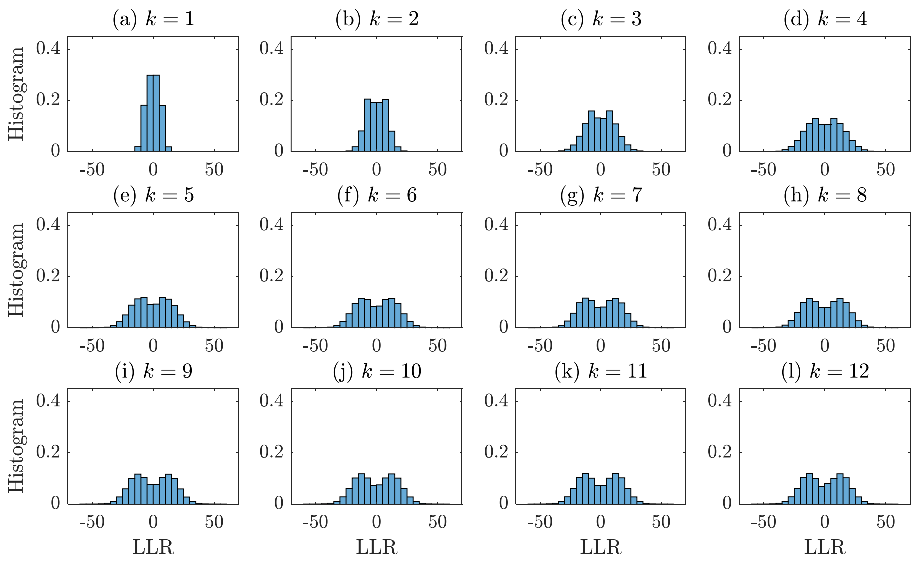

Figure 5. Histogram of all the generated L_j per iteration at $E_b/N_0 = 1.0$dB.

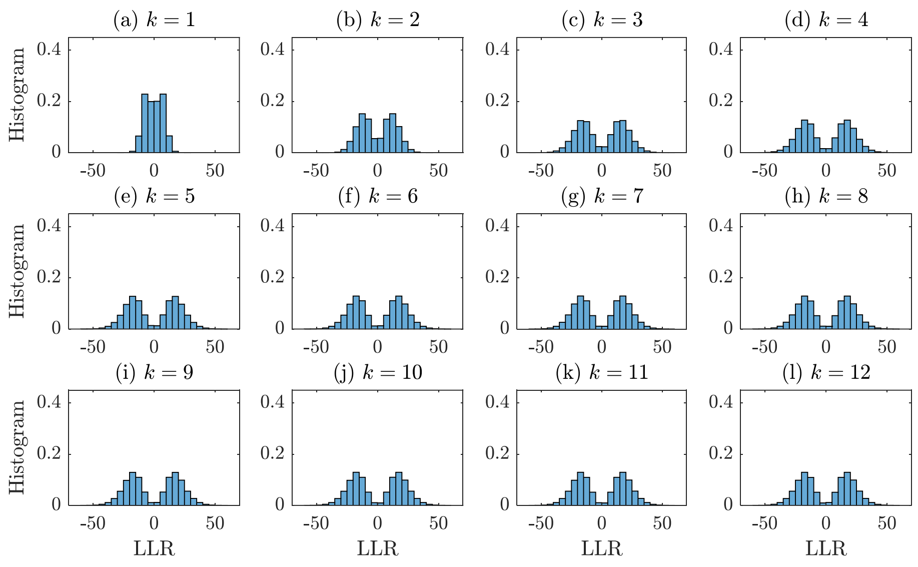

Figure 6. Histogram of all the generated L_j per iteration at $E_b/N_0 = 1.5$dB.

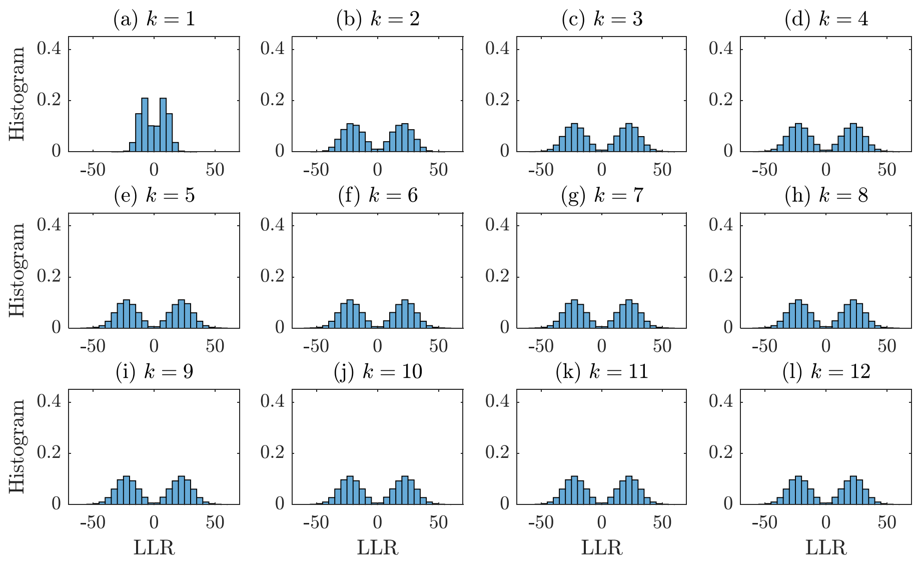

Figure 7. Histogram of all the generated L_j per iteration at $E_b/N_0 = 2.0$dB.

All these curves show that the number of near-zero-LLRs decreases as the iteration index increases. However, the rate of this decrease is directly related to SNR. The reduction is slightly at low SNRs and becomes faster as the SNR increases. This ratio is high at the start of decoding (the 1st iteration) since the decoder starts from zero a priori values, which are updated after each iteration. At low SNRs, the number of LLRs whose values are very small (around zero) virtually do not change, even as the iteration proceeds. This is because the noise levels of the received blocks exceed the capability of correction the turbo decoder can provide. As the SNR increases, the values of the LLRs increase along with the iterations. This figure shows the reduction of the ratio of the LLRs with near-zero values as the iteration index increases. The speed of this reduction is directly proportional to the SNR. This ratio drastically decreases at high SNRs at early stages of the iterative decoding.

We can incorporate the above results to examine the overall turbo code performance for the maximum number of iterations allowed per decoding. For the selected iteration indices, Figure 9 and Figure 10 plot the probability of frame error (Frame Error Rate FER) and the probability of bit error (Bit Error Rate BER) respectively. FER is the measure of decoding fidelity, the number of blocks correctly decoded without any error at the given SNR. The BER reflects the total correctly decoded bits among the whole transmission (which is here 1000 blocks times 512 bits per block). We compute FER and BER at different numbers of iterations. It is apparent in these figures that the more iterations the decoder performs, the better the performance can be achieved.

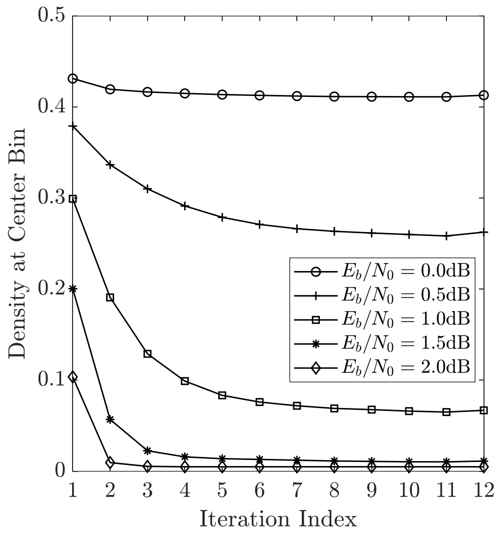

Figure 8. Density of LLRs located at histogram bins around the zero $(|L_j| \leq 5)$ per E_b/N_0 for the indicated iteration index.

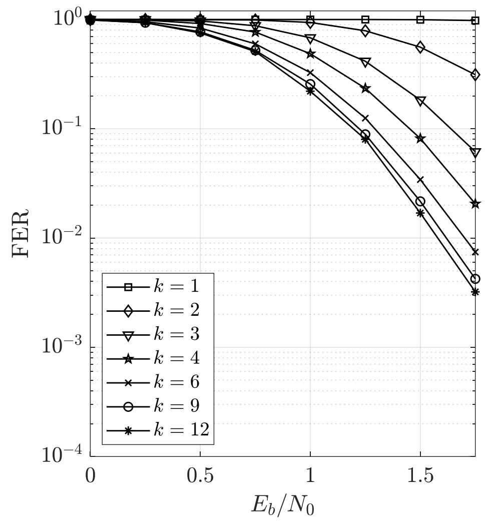

Figure 9. FER at the indicated iteration index.

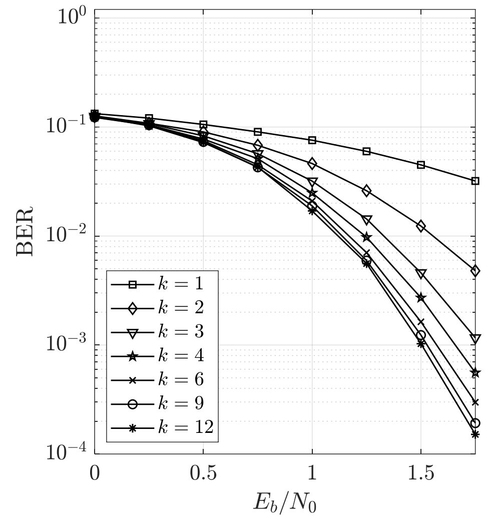

Figure 10. BER at the indicated iteration index.

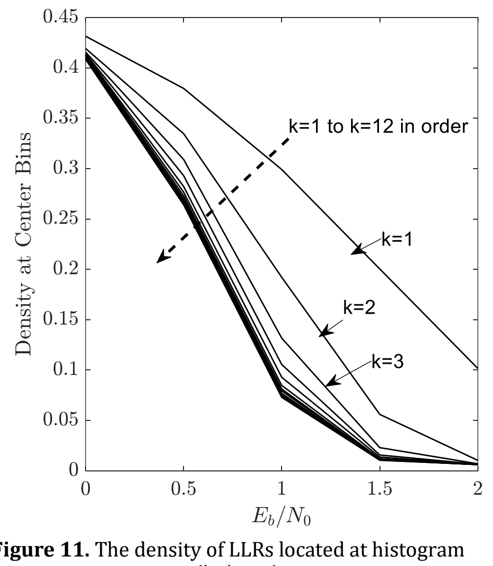

Figure 11. The density of LLRs located at histogram bins around the zero $(|L_j| \leq 5)$ per iteration index for the indicated SNRs (E_b/N_0).

First, we can review the histograms (a) from the figures. **Error! Reference source not found.** to Figure 7, which are the decoding results after only one iteration, and relate them to the curve 1 in Figure 9 and Figure 10. At this iteration, the turbo decoder has not achieved any respected results. This is especially obvious at low to mid SNRs. The progress of the iterative decoding improves the results, as it is clear from later cases in **Error! Reference source not found.** to Figure 7. For example, if we compare the results of plots (c) to (a) in these figures, we can see the reflection clear in the curve 3 compared to 1 in Figure 9 and Figure 10. This improvement continues as the number of iterations increases. The 'best' results the decoder can reach are plotted in histogram (l) in the figures **Error! Reference source not found.** to Figure 7. Their corresponding performance curves is labeled 12 in Figure 9 and Figure 10.

All the plots in **Error! Reference source not found.** to Figure 7 The results show that the decoder reaches convergence at high SNRs after few iterations. This means that without an early stopping rule, the decoder performs unnecessary processing that does not lead to significant improvement.

Another way to present the gathered results is the Figure 11. The curves in this chart, we plot the number of LLRs within the near-the-zero bins as a function of E_b/N_0 at each iteration index. These curves show the direct inverse relation between the number of LLRs near zero and the SNR and the iteration index. This chart shows that the number of LLRs within the near-zero-bins can be a reliable performance measure.

The proposed method can indicate the development of the soft output. As the discussed figures show, the decoder has a better chance of correct decoding when the number of LLRs within the bin around 0 decreases along the iterations. If this number does not change considerably, the decoder is unlikely to be able to decode the received message correctly. So, saving the decoding resources and initiating the retransmission request can be beneficial.

5. Conclusion

Turbo codes are highly effective for data transmission because they can achieve significant error correction. They use multiple decoders that interact with each other repeatedly to refine the decoded data. This study investigates how the quality of the decoded data improves over the iterations. By examining the distribution of decoded soft values using histograms, we can better visualize and understand this progress. We propose a method to evaluate and predict the decoding performance based on the distribution of generated soft values within the histogram. Results show that the decoder's effectiveness is inversely related to the number of soft-generated outputs near zero. The proposed method can be implemented within the decoder to provide an early indication of whether the reception has the potential to be accurately decoded. This early decision can help conserve decoding resources.

Funding

None.

Acknowledgements

None.

Conflicts of Interest

The author declares no conflict of interest.

References

[1] M. Rowshan, M. Qiu, Y. Xie, X. Gu, and J. Yuan, "Channel Coding Toward 6G: Technical Overview and Outlook," IEEE Open Journal of the Communications Society, vol. 5, pp. 2585-2685, 2024.

[2] Y. Ding, L. Zhao, and J. Feng, "Turbo code optimization based on terahertz communication," Eighth International Conference on Electronic Technology and Information Science (ICETIS). SPIE, 2023.

[3] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo decoders on multi-core architectures," (in en), Ann. Telecommun., vol. 75, no. 1, pp. 27-42, 2020/02/01/2020.

[4] S. Weithoffer, G. Aousaji, J. Nadal, and C. A. Nour, "Iteration Overlap for Low-Latency Turbo Decoding," in 12th International Symposium on Topics in Coding (ISTC), 4-8 Sept. 2023, pp. 1-5.

[5] Z. Liu, R. Liu, H. Zhang, N. Wang, L. Sun, and J. Wang, "Parallel implementation of the CCSDS turbo decoder on GPU," China Communications, pp. 1-8, 2023.

[6] S. Saito, K. Fujimoto, and A. Shiraga, "Low-latency remote-offloading system for accelerator," Ann. Telecommun., vol. 79, no. 3, pp. 179-196, 2024/04/01 2024.

[7] X.-G. Xia, "Understanding turbo codes: A signal processing study," Journal of Information and Intelligence, vol. 2, no. 1, pp. 1-13, 01/2024.

[8] E. Yoon, S. Kwon, and S. Y. Kim, "An Efficient Application of Turbo Coding for OFDM In-Phase/Quadrature Index Modulation," *IEEE Access,* vol. 11, pp. 37031-37040, 2023.

[9] R. D. Agustin, I. G. P. Astawa, and A. Pratiarso, "Performance Analysis of Turbo Coding Implementation in MIMO Based DVB-T2 System," 2020 International Electronics Symposium (IES), 29-30 Sept. 2020, pp. 179-183.

[10] A. Alashqar, J. Alkasassbeh, R. Mesleh, and A. Al-Qaisi, "SDR implementation and real-time performance evaluation of 5G channel coding techniques," AEU - International Journal of Electronics and Communications, vol. 170, p. 154852, 2023/10/01/ 2023.

[11] M. Wei, D. Ren, and Q. Huang, "Design and Implementation of Low-Rate Turbo Coding and Decoding," 2024 IEEE/CIC International Conference on Communications in China (ICCC), 7-9 Aug. 2024, pp. 450-454.

[12] S. Miao *et al.*, "Trends in Channel Coding for 6G," Proceedings of the IEEE, pp. 1-23, 2024.

[13] S. Vijayalakshmi, A. Paramasivam, S. Sakthivel, S. Kudiyarasan, E. Sankaran, and V. Nagarajan, "Design of high-speed data transfer turbo code for body channel communication transceivers," International Journal of Communication Systems, vol. 36, no. 7, p. e5447, 2023.

[14] C. Zhang, Y. Lin, D. Wang, and J. Hu, "Design of Low-Power Turbo Encoder and Decoder for NB-IoT," Chinese Journal of Electronics, vol. 33, no. 2, pp. 403-414, 2024.

[15] G. Zhao, "Dissecting and Implementing SOYA Algorithm Variations in Convolutional and Turbo Code Decoding: An Analytical Approach," 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), 24-26 Feb. 2024, pp. 780-785.

[16] Y. Shen, Y. Ren, A. T. Kristensen, X. You, C. Zhang, and A. Burg, "Improved Belief Propagation Decoding of Turbo Codes," ICASSP - IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-10 June 2023, pp. 1-5.

[17] R. Umar, F. Yang, H. Xu, and S. Mughal, "Distributed turbo coded spatial modulation based on code matched interleaver for MIMO system," Wireless Networks, vol. 29, no. 5, pp. 1995-2013, 2023/07/01 2023.

[18] J. Wang and Z. Wang, "Research on Parallel Turbo Encoding and Decoding Technology," IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 24-26 May 2024, vol. 6, pp. 1378-1381.

[19] D. V. Linh and V. V. Yem, "A turbo-based encryption and coding scheme for multiple-input multiple-output orthogonal frequency division multiplexing wireless communication systems affected by Doppler frequency offset," IET Communications, vol. 17, no. 5, pp. 632-640, 2023.

[20] A.-M. Cuc, F. L. Morgos, and C. Grava, "Perfor-mance Analysis of Turbo Codes, LDPC Codes, and Polar Codes over an AWGN Channel in the Presence of Inter Symbol Interference," Sensors, vol. 23, no. 4, p. 1942, 2023.

[21] L. C. Perez, J. Seghers, and D. J. Costello, "A Distance Spectrum Interpretation of Turbo Codes," IEEE Transactions on Information Theory, vol. 42, no. 6, pp. 1698-1709, 1996.

[22] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in IEEE GLOBECOM. Communications: The Global Bridge, 1994/11, vol. 3, pp. 1298-1303.

[23] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," (in English), IEEE Transactions on Information Theory, vol. 20, no. 2, pp. 284-287, 1974/03// 1974.

[24] I. European Telecommunications Standards, "Technical Specification; LTE; Evolved Universal Terrestrial Radio Access (E-UTRA), Multiplexing and channel coding (3GPP TS 36.212 version 16.2.0 Release 16)," 2020 2020.

[25] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," IEEE International Conference on Communications, 1995/06, vol. 2, pp. 1009-1013 vol.2.

[26] M. AlMahamdy and J. Dill, "Half-Iteration Early Termination of Turbo Decoding," IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017/01/09/11, Las Vegas, NV USA: IEEE, pp. 591-595.

[27] M. AlMahamdy and J. Dill, "Early Termin-ation of Turbo Decoding by Identification of Undecodable Blocks," EEE Wireless Communications and Networ-king Confere-nce (WCNC), 2017/03/19/22 2017, San Francisco, CA USA: IEEE.

[28] M. A. H. AlMahamdy, "New Methods to Reduce Turbo Decoding Latency and the Complexity of Bit Insertion Techniques," Ohio University, 2017.