Turk J Elec Eng & Comp Sci, Vol.18, No.6, 2010, © TÜBİTAK
doi:10.3906/elk-0811-14

A second order approximation to reduce the complexity of LDPC decoders based on Gallager's approach

Aykut KALAYCIOĞLU[1], Oktay ÜRETEN[2], H. Gökhan İLK[1]
[1]*Department of Electronics Engineering, Faculty of Engineering, Ankara University, Ankara, TURKEY*
e-mail: aykut.kalaycioglu@eng.ankara.edu.tr, h.gokhan.Ilk@eng.ankara.edu.tr
[2]*Communications Research Centre Canada, Ottawa-CANADA*
e-mail: oktay.ureten@crc.ca

Abstract

A piece-wise second order approximation to the $f(x) = -\log[\tanh(x/2)]$ *function is proposed to reduce the computational complexity of LDPC decoder's utilizing Log-Likelihood Ratio Belief Propagation (LLR-BP) algorithm based on Gallager's approach. Simulation results show that the proposed low complexity approximation doesn't cause BER performance degradation.*

1. Introduction

Reduced complexity decoding of low-density parity-check (LDPC) codes [1] has been receiving much attention recently as these powerful codes are employed in various communications standards, such as IEEE 802.16e, DVB-S2 and IEEE 802.3an. Even though effective decoding of LDPC codes can be achieved using the sum-product (SP) or belief-propagation (BP) algorithms [2], these algorithms demand heavy computational resources. The computational complexity of the BP algorithm can be reduced using the min-sum approach at the expense of a small performance loss [3, 4]. Further computational reductions can be achieved if log-likelihood ratios (LLRs) are incorporated instead of likelihood ratios or probabilities, as the multiplications are replaced by additions in this approach.

LLR-BP based on the tanh rule and LLR-BP based on Gallager's approach are the two numerically accurate representations of the BP algorithm employed in decoding LDPC codes [3]. These algorithms can be summarized in four steps: initialization, variable and check-node updates and decision. All steps are identical for the two algorithms except the check-node update step, which is the computationally most complex part of these algorithms. Following the same notation in [5], the initialization step assigns prior values to the LLR values $\lambda_{n\to m}(u_n) = L(u_n)$ and $\Lambda_{m\to n}(u_n) = 0$ to every entry $(m,\ n)$ of the parity-check matrix which has a non-zero element. For every received codeword bit y_n corresponding to the transmitted codeword bit u_n, LLRs derived from the channel outputs are $L(u_n) = 2y_n/\sigma^2$, where σ^2 is the variance of the additive white Gaussian

channel noise. Following the initial assignments, the LLR values of the variable-nodes are updated according to $\lambda_{n\to m}(u_n) = L(u_n) + \sum_{m'\in M(n)\setminus m} \lambda_{m'\to n}(u_n)$ and $\lambda_n(u_n) = L(u_n) + \sum_{m\in M(n)} \Lambda_{m\to n}(u_n)$, where $M(n)$ denotes the set of check nodes connected to the variable node n. The check-nodes are then updated according to the following equations for the *tanh* rule and Gallager's approach, respectively:

$$\Lambda_{m\to n}(u_n) = 2 \prod_{n'\in N(m)\setminus n} sign\left[\lambda_{n'\to m}(u_{n'})\right] \tanh\left(\prod_{n'\in N(m)\setminus n} \tanh\left[|\lambda_{n'\to m}(u_{n'})/2|\right]\right) \quad (1)$$

$$\Lambda_{m\to n}(u_n) = \prod_{n'\in N(m)\setminus n} sign\left[\lambda_{n'\to m}(u_{n'})\right] f\left(\sum_{n'\in N(m)\setminus n} f\left(|\lambda_{n'\to m}(u_{n'})|\right)\right) \quad (2)$$

where $f(x) = -\log[\tanh(x/2)]$, for $x > 0$ and $N(m)$ denotes the set of symbol nodes that participate in the m^{th} parity-check equation. Finally, a bit is decided to be zero if $\lambda_n(u_n) \geq 0$, or one otherwise.

The computationally most complex parts of these algorithms are the check-node update steps given by equations 1 and 2 for the *tanh* rule and Gallager's approach, respectively. From the implementation point of view, the *tanh* function is rather complex as it involves operations such as additions, exponentiations and divisions. An accurate representation of the $tanh(x)$ is essential for reliable decoding of the LDPC codes by the LLR-BP algorithm based on the *tanh* rule. Recently, an approximation method is proposed to modify $tanh(x)$ and $tanh^{-1}(x)$ functions to reduce the computational complexity without sacrificing the performance [5]. In this approach, a piecewise linear function with seven regions is employed for the approximation. First-order approximations in each region are determined such that the root mean square (rms) value of the approximation error is 0.02. Simulation results in [5] show that the proposed approximation to $tanh(x)$ reduces the computational complexity significantly without noteworthy performance degradation.

To the best knowledge of the authors, no such approximation exists for the $-\log[\tanh(x/2)]$ function used in Gallager's approach, which is another numerically accurate version of the LLR-BP decoding algorithm. In this paper, we propose a piece-wise second order approximation to the $-\log[\tanh(x/2)]$ function to reduce the computational complexity of the LLR-BP based on Gallager's approach. Simulation results show that our proposed approximation reduces the complexity without degrading the decoding performance.

2. Proposed algorithm

A first-order piecewise approximation, which is shown to be a good choice for $tanh(x)$ in [5], is not suitable for the $-\log[\tanh(x/2)]$ function used in Gallager's approach. This is mostly due to the fact that the Taylor series expansion of the logarithm contains higher-order terms with both even and odd powers whereas the hyperbolic tangent function contains only odd-power terms in its Taylor expansion. The magnitudes of the higher order terms in the *tanh* expansion are rapidly approaching to zero, making the linear approximation a suitable choice. On the other hand, the logarithm function includes a second-order term with relatively high magnitude in its Taylor-series expansion. Based on this fact, the second-order approximation is viewed as an appropriate choice for approximating the $-\log[\tanh(x/2)]$ function.

In order to determine the number of piecewise regions, a heuristic approach is employed. Even small inaccuracies in the approximation may result in large errors, especially for small input values, due to the nature

of the $-\log[\tanh(x/2)]$ function. The input space is divided into twelve regions in order to keep the rms value of the approximation error below the level of 10^{-3}. As x approaches to zero, $-\log[\tanh(x/2)]$ becomes very large, therefore a fixed value of 10^4 is used for $-\log[\tanh(x/2)]$ when x is less than 10^{-6}. A similar approximation is also used in the exact calculation of the function, which is given in equation (2), in Gallager's approach in order to avoid numerical instabilities due to overflows. The obtained second order approximation values are tabulated in Table 1.

Table 1. Second order piecewise approximation for $-\log(\tanh(x/2))$.

$[0.0, 10^{-6})$	10^4
$[10^{-6}, 0.1)$	no approximation
$[0.1, 0.3)$	$14.15x^2 - 10.91x + 3.92$
$[0.3, 1.0)$	$1.43x^2 - 3.40x + 2.76$
$[1.0, 2.0)$	$0.271x^2 - 1.301x + 1.794$
$[2.0, 3.0)$	$0.0856x^2 - 0.5976x + 1.124$
$[3.0, 4.0)$	$0.0309x^2 - 0.278x + 0.6553$
$[4.0, 5.0)$	$0.0113x^2 - 0.1247x + 0.3539$
$[5.0, 6.0)$	$0.004163x^2 - 0.0542x + 0.1802$
$[6.0, 7.0)$	$0.001531x^2 - 0.02299x + 0.08774$
$[7.0, 9.0)$	$0.0003603x^2 - 0.006505x + 0.02965$
$[9.0, \infty)$	0.00010672

In summary, a piece-wise second order approximation to the $f(x) = -\log[\tanh(x/2)]$ function is proposed to reduce the computational complexity of LDPC decoder's based on Gallager's approach. The approximation method presented here requires additions and multiplications in nine regions while the exact implementation of the f function is needed in one region. No calculations are required in the other two regions, in which fixed values are employed.

3. Complexity

The proposed approximation method requires computation of the hyperbolic tangent and logarithm functions only for the input values in the range $[10^{-6}, 0.1)$ whereas the exact implementation of the $-\log[\tanh(x/2)]$ function needs these calculations for the entire input space. Computation savings with the proposed approximation depend on the percentage of computations that the LLR values are outside the range where the exact implementation of the function is used. This is difficult to predict precisely as the LLR distribution depends on noise variance, number of decoding iterations and the code structure.

In order to predict the typical computational saving, we have run simulations at different SNR levels and code lengths. A regular $(252, 504)$ and $(504, 1008)$ code with no cycles of length 4 obtained from MacKay's online database [6] are used in simulations. A histogram was updated during the simulation when the input values fell into the range requiring an exact calculation. The number of maximum decoding iterations is fixed at 16 and the decoder was allowed to stop earlier if all of the parity equations were satisfied. The results are shown in Figure 1. As shown in the figure, only up to 7% of all computations of the proposed approximation method take place in the no approximation region for both $(252, 504)$ and $(504, 1008)$ codes. Thus, we can expect that at least 93% of the time, the approximation method presented here does not require the implementation of the

exact function, which will achieve a computational saving over the exact calculation, which includes logarithm and hyperbolic tangent functions.

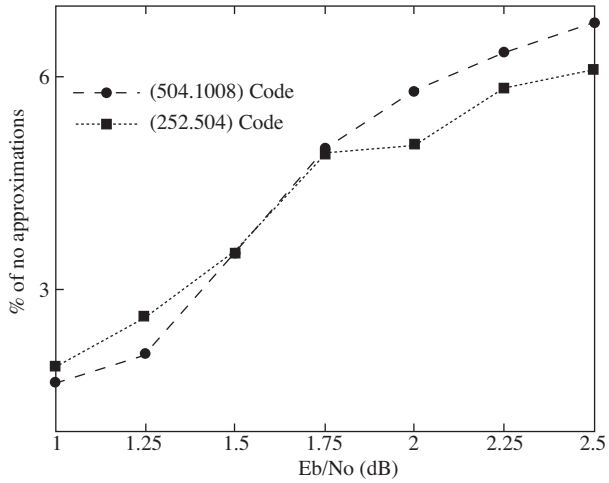

Figure 1. Percentage of computations of the proposed method that takes place in the no approximation region.

As stated in section II, it is clear that computation complexity of a logarithm and a hyperbolic tangent function is higher than that of additions and multiplications in a second order polynomial. Since Taylor series expansion of the logarithm function includes higher-order terms, a second order polynomial in the proposed approximation method requires less additions and multiplications than the exact function demands in a series expansion.

4. Simulation examples

The (252, 504), (504, 1008) and (2000, 4000) regular LDPC codes with parity-check matrices containing no length-4 cycles in their corresponding graphs [7] obtained from MacKay's online database [6] were used in the simulations. Additionally, equal-size parity matrices containing 30 length-4 cycles [8] have been generated and used to evaluate the performance when length-4 cycles exist in the parity matrices. The number of fixed ones in each column and row is three and six, respectively. Encoded bits are binary-phase-shift-keying (BPSK) modulated and transmitted over the simulated AWGN channels. The number of maximum decoding iterations is set to 16 for all three regular LDPC codes at each E_b/N_0 value. Blocks have been transmitted until 50 block errors are counted for each E_b/N_0 value.

Simulation results for codes with no length-4 cycles are shown in Figure 2. As illustrated in this figure, there is no performance degradation over the exact implementation when the approximation with reduced complexity is used. The performance of the approximated version is slightly better at higher SNR values for shorter codes. This anomaly is the result of the non-optimal behavior of the BP algorithm for short codes [3, 5] and it diminishes as the code length increases, as seen from the BER curve of (2000, 4000) code in the same figure.

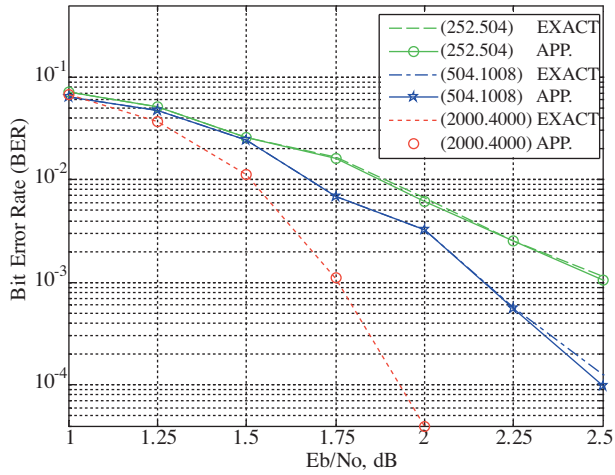

Figure 2. BER curves for $(252,\ 504)$, $(504,\ 1008)$ and $(2000,\ 4000)$ codes (Codes don't contain length-4 cycles).

Further simulations have been run using codes that contain length-4 cycles and the results are presented in Figure 3. As seen from the figure, the performance of the approximated version with reduced complexity is closer to the exact implementation as the code length increases. For short codes, it performs better than the exact version due to the non-optimal characteristics of the BP algorithm.

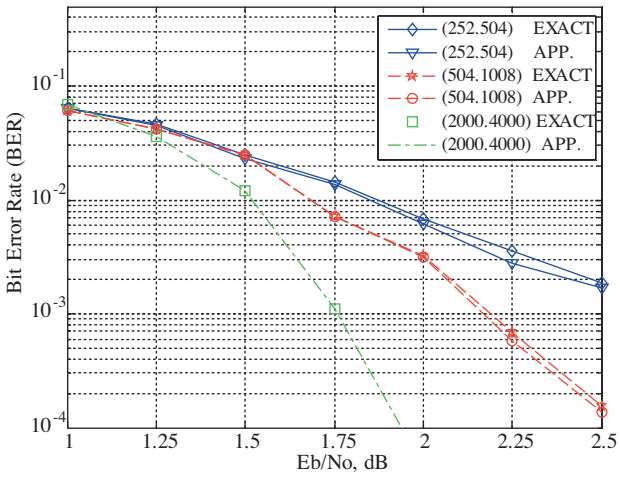

Figure 3. BER curves for $(252,\ 504)$, $(504,\ 1008)$ and $(2000,\ 4000)$ codes (Codes contain 30 length-4 cycles).

5. Conclusions

In this paper, a second order approximation to the $-\log\left[\tanh\left(x/2\right)\right]$ function is proposed for reduced complexity decoding of LDPC codes based on Gallager's approach. The approximation method reduces the computational complexity by avoiding computations of the logarithm and hyperbolic tangent functions presented in the exact implementation of the f function. Although the computational savings tend to decrease with longer code lengths and higher SNR values, the approximation method is efficient for shorter codes as shown in the simulations.

Simulation results show that some anomalies can be observed in testing the approximated function in short codes and codes containing length-4 cycles. These are due to the non-optimum character of the belief propagation algorithm, which is only optimal for cycle-free graphs.

Acknowledgment

The authors would like to thank Dr. Ron Kerr (CRC) for his valuable comments. Aykut Kalaycıoğlu has been supported by a TÜBİTAK study grant during his Ph.D. study.

References

[1] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Transactions on Information Theory*, vol. 7, pp. 21-28, 1962.

[2] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399-431, 1999.

[3] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, X.Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," *IEEE Transactions on Communications*, vol. 53, no. 8 pp. 1288-1299, 2005.

[4] M. P. C. Fossorier, M. Mihaljevic, H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity-Check Codes Based on Belief Propagation", *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673-680, 1999.

[5] S. Papaharalabos, P. Sweeney, B.G. Evans, P.T. Mathiopoulos, G. Albertazzi, A. Vanelli-Coralli, G.E. Corazza, "Modified sum-product algorithms for decoding low-density parity-check codes," *IET Communications*, vol. 1, no. 3, pp. 294-300, 2007.

[6] D.J.C. MacKay, "Online database of low-density parity-check codes," *http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html.*

[7] N. Wiberg, "Codes and Decoding on General Graphs", Ph.D. Dissertation, Linköping University, Linköping, Sweden, 1996.

[8] J. Fan, Y. Xiao, "A Method of Counting the Number Cycles in LDPC Codes", *The 8^{th} International Conference on Signal Processing*, vol. 3, pp. 16-20, 2006.