Hybrid Log-MAP Algorithm for Turbo Decoding Over AWGN Channel

Li Li Lim
Faculty of Engineering
The University of Nottingham
Selangor, Malaysia
Email:keyx7lll@nottingham.edu.my

David Wee Gin Lim
Faculty of Engineering
The University of Nottingham
Selangor, Malaysia
Email:lim.wee-gin@nottingham.edu.my

***Abstract*—In Log-MAP turbo decoding, the complicated log exponential sum is often simplified with the Jacobian logarithm which consists of the max operation along with an exponential correction function. Although the Max-Log-MAP reduces the complexity of the Jacobian logarithm implementation by omitting the correction function, its performance is inferior to the exact Log-MAP algorithm. Hence, a simple approximation to the correction function is needed to complement the Max-Log-MAP algorithm. In this paper, a new suboptimal hybrid Log-MAP algorithm for decoding turbo code is proposed. It approximates the exact Log-MAP accurately and is simple for hardware implementation. The performance of the hybrid Log-MAP algorithm is shown to have the closest performance to the exact Log-MAP solution especially at low SNR. The result of other Log-MAP based algorithm are also presented.**

***Index Terms*—Log-MAP, turbo codes, correction function, Jacobian logarithm.**

I. Introduction

Turbo codes were first introduced by Berrou et al [1] and is among the most powerful error correcting codes. The MAP decoder is often operated in the log domain in order to reduce computational complexity [2]. However, the computation for state metrics and log likelihood ratio (LLR) is still burdened by the log exponential sum calculation. The Jacobian logarithm is used to simplify the log exponential sum by employing a correction function along with the maximum operator in the log domain [3]. Although simplified, the correction function remains a nonlinear exponential function. The manner in which the correction function is calculated is critical to the performance and complexity of the decoder. Several methods have been proposed to simplify its computation which gives a tradeoff between complexity and performance [4]–[6].

This paper presents a simplified hybrid algorithm for sub optimal Log-MAP. The algorithm employs linear fitting methods as well as minimal bitwise shift operations to compute the correction function that will be used in the calculations of state metrics and ultimately the LLR. In this algorithm, only linear multiplication, addition, comparator, and minimal number of bit shifts are required to obtain near Log-MAP performance.

The rest of the paper is organized as follows: in Section II, a brief review of the original MAP, Log-MAP as well as Max-Log-MAP algorithm are presented. Section III reviews existing methods for the approximation of the correction function. Next, we present the novel hybrid approximation in Section IV. In Section V, the simulation results of the hybrid Log-MAP in comparison with other approximation methods is presented. The paper is finally concluded in Section VI.

II. The MAP, Log-MAP, and Max-Log-MAP Algorithms

In this section, the derivation of the MAP algorithm will not be detailed but the results of the algorithm will be stated. For details on the derivation, see [1] and [2]. Following this, a review of the Log-MAP as well as the introduction to the max^* operator in the Log-MAP algorithm and the Max-Log-MAP algorithm will be presented.

A. The Encoder

The MAP algorithm operates in blocks of binary input (information) data sequence represented by $\overrightarrow{d} = (d_1, ..., d_N)$. The encoder consists of two identical recursive systematic convolutional (RSC) encoders with M memory elements. We will assume that each of the RSC encoder has a rate of $1/2$ with two outputs, which is the sequence of systematic bits $\overrightarrow{x^s} = (x_1^s, ..., x_N^s)$, and the parity bit sequence, $\overrightarrow{x^p} = (x_1^p, ..., x_N^p)$. The outputs are punctured and transmitted over the channel. The corresponding received sequences are $\overrightarrow{y^s}$ and $\overrightarrow{y^p}$ or for brevity, $\overrightarrow{y} = (\overrightarrow{y^s}, \overrightarrow{y^p})$ which refers to the pair of received systematic and parity bit received.

B. Maximum A Posteriori (MAP) Algorithm

Let the state of the encoder at time k be S_k, where it can take on values between 0 and $2^M - 1$. The bit d_k represents the transition from step $k-1$ to k. The goal of the MAP algorithm is to provide the LLR, $\Lambda(d_k)$ of the a posteriori probability (APP) of $d_k = 1$ to the APP of $d_k = 0$ as follows:

$$\Lambda(d_k) = \ln \frac{\sum_{S_k} \sum_{S_k - 1} \gamma_1(y_k, S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{S_k} \sum_{S_k - 1} \gamma_0(y_k, S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}. \quad (1)$$

 ISBN: 978-1-61208-140-3

The forward state metric α_k can be expressed as:

$$\alpha_k(S_k) = \frac{\sum\limits_{S_{k-1}} \sum\limits_{i=0}^{1} \gamma_i(y_k, S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1})}{\sum\limits_{S_k} \sum\limits_{S_k-1} \sum\limits_{i=0}^{1} \gamma_i(y_k, S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1})} \quad (2)$$

and the backward state metric β_k is given as:

$$\beta_k(S_k) = \frac{\sum\limits_{S_{k+1}} \sum\limits_{i=0}^{1} \gamma_i(y_{k+1}, S_k, S_{k+1}) \cdot \beta_{k+1}(S_{k+1})}{\sum\limits_{S_k} \sum\limits_{S_k+1} \sum\limits_{i=0}^{1} \gamma_i(y_{k+1}, S_k, S_{k+1}) \cdot \alpha_k(S_k)}. \quad (3)$$

The branch transition probabilities, γ_i are given as:

$$\gamma_i((y_k^s, y_k^p), S_{k-1}, S_k) = \exp[\frac{1}{2} x_k^s (L_e(x_k^s) + L_c y_k^s) + L_c y_k^p x_k^p]; \quad (4)$$

where the channel reliability value, $L_c = \frac{2}{\sigma^2}$ with σ^2 being the noise variance and L_e is the *extrinsic* information which serves as the *a priori* information.

C. Log-MAP Algorithm

The MAP algorithm is too complex for practical implementation in a real system. To avoid complicated operations, the entire MAP algorithm can be computed in the log domain. By taking the logarithm of $\alpha_k(S_k), \beta_k(S_k)$ and $\gamma_k(S_{k-1}, S_k)$, the MAP algorithm reduces to addition and multiplication operations. However, the computation of forward state metric, $\tilde{\alpha_k}(S_k) = \ln\alpha_k(S_k)$ involves the log exponential sum which is complicated to implement in hardware:

$$\tilde{\alpha_k}(S_k) = \ln(\sum_{S_{k-1}} \sum_{i=0}^{1} e^{\ln\gamma_i((y_k^s, y_k^p), S_{k-1}, S_k) + \ln\alpha_{k-1}(S_{k-1})}) - \ln(\sum_{S_k} \sum_{S_{k-1}} \sum_{i=0}^{1} e^{\ln\gamma_i((y_k^s, y_k^p), S_{k-1}, S_k) + \ln\alpha_{k-1}(S_{k-1})}); \quad (5)$$

Simplifying (5) gives the general form:

$$F(x_1, x_2, ..., x_n) = \ln(\sum_{i=1}^{n} e^{x_i}); \quad (6)$$

where n is a function of the encoder states. Consider the Jacobian logarithm for two variables given as:

$$\begin{aligned} \max^*(x_1, x_2) &= \ln(e^{x_1} + e^{x_2}) \\ &= \max(x_1, x_2) + \ln(1 + e^{-|x_1 - x_2|}) \\ &= \max(x_1, x_2) + f_c(x) \end{aligned} \quad (7)$$

where $f_c(x)$ is the correction function, and $\max(x_1, x_2)$ is the maximum of the function's two arguments. The computation of the exponential term of $f_c(x)$ in $\max^*(x_1, x_2)$ greatly increases the complexity of the Log-MAP algorithm. A simple way to obtain (6) with $n > 2$ is to recursively perform the $\max^*$ operator as follows [6]:

$$F(x_1, x_2, ..., x_n) = max^*(x_n, max^*(x_{n-1}, ..., max^*(x_3, max^*(x_2, x_1)))...). \quad (8)$$

Similar forms of the log-exponential-sum occurs for the calculation of the backward state metric, $\beta_k(S_k)$ and the LLR, $\Lambda(d_k)$. It is clear that in order to improve the performance of the Log-MAP algorithm, a simpler implementation to the correction function must be found. One of the earliest significant finding is to present the correction function as a single dimension look up table [3]. However, table sequences require storage, and additional memory units will have to be added to the Log-MAP decoder rendering the increase in area, power, and a drop in the overall speed [5]. Hence, another sub-optimal solution to this would be to implement the correction function with a simpler approximate as will be detailed in Section III.

D. Max-Log-MAP Algorithm

With the Max-Log-MAP algorithm the max^* operation is loosely approximated using

$$max^*(x_1, x_2) \approx max(x_1, x_2). \quad (9)$$

The Max-Log-MAP simplifies the Log-MAP algorithm by simply omitting the correction function $f_c(x)$ altogether. The performance for the Max-Log-MAP algorithm gives up to a 10% performance drop [7] when compared to the Log-MAP. The Max-Log-MAP algorithm is the least complex of all the existing methods but offers the worst BER performance. This creates a need to complement the Max-Log-MAP algorithm with a simple implementation of the correction function in order to improve performance.

III. Sub Optimal Log-MAP Algorithms: Approximation to the Correction Function.

This section gives a brief review of existing algorithms which approximates the correction function in order to achieve a simple implementation yet improved performance as compared to Max-Log-MAP.

A. Constant Log-MAP Algorithm.

In this algorithm proposed by [4] , the correction function $f_c(x)$ is approximated with the following rule:

$$f_c(x) = \begin{cases} \frac{3}{8}, & -2 \leq x < 2 \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

The constant Log-MAP algorithm offers a simple implementation in hardware but with trade off in performance.

B. Linear Log-MAP Algorithm.

In [5], the author suggests a linear approximation to the correction function by employing the MacLaurin Series expansion. It is observed that the correction function is effective when $f_c(x)$ is around zero. Therefore, the Maclaurin series can be exploited to approximate the correction function about zero. By neglecting Maclaurin's series order two and above, the approximation for the correction term is given as:

$$f_c(x) \approx max(0, \ln 2 - \frac{1}{2}x). \quad (11)$$

 ISBN: 978-1-61208-140-3

This approximation offers better performance than the Constant Log-MAP algorithm and requires only a simple linear implementation.

C. Multistep Log-MAP Algorithm.

A more accurate and elegant solution to approximate the correction function is given by [6]. The approximation to the correction term as suggested by [6] is given as:

$$f_c(x) \approx \frac{\ln 2}{2^{\lfloor x+0.5 \rfloor}}. \quad (12)$$

Where $\lfloor x+0.5 \rfloor$ denotes the largest integer that is smaller or equal to $x+0.5$. The correction term given here is a more accurate yet simple approximation to the correction function. Note that division by 2 can be easily done in digital systems by implementing $\lfloor x+0.5 \rfloor$ number of binary shifts. The algorithm employs shift registers storing the constant $\ln(2)$ to perform the division in [8]. However, in order to facilitate fast computation, a high speed shift register is needed for this algorithm.

IV. Novel Hybrid Log-MAP Approximation

In our novel development, our approximation has the advantage over the linear and multistep Log-MAP algorithm in [5] and [6] respectively in terms of accuracy as well as hardware simplicity. The hybrid approximation is proposed as:

$$f_c(x) \approx \begin{cases} 0.6512 - 0.3251x & \text{for } x < 1.5 \\ \frac{0.1635}{2^{\lfloor 0.5x \rfloor}} & \text{otherwise} \end{cases} \quad (13)$$

When plotted against the exact correction function in Fig. 1, the hybrid approximation proves to be a better fit to the correction term than [5] and [6]. The hybrid approximation is divided into two regions i.e., $|x| < 1.5$ and $|x| \geq 1.5$.

In the region of $|x| < 1.5$ the hybrid algorithm employs a linear polynomial fit. The accuracy of the approximation is verified through the goodness of fit test against the exact Log-MAP curve with parameters of SSE: 0.05058, R-square: 0.9836, Adjusted R-square: 0.9835, and RMSE: 0.01842. The polynomial fit is acceptably accurate as indicated with a confidence interval of 95%.

Inspired by [6], a good approximation can be achieved by adjusting Equ. (12). The constant ln2 is replaced with the constant 0.1635 where it is the corresponding value for $|x| = 1.5$ for the linear region. The replacement gives a double advantage. The replacement with the constant 0.1635 requires lesser number of bitwise shifts for larger values of $|x|$ and this reduces the number of shift operations needed to perform computation for $f_c(x \geq 1.5)$. It is observed that $f_c(x)$ can be approximated to zero for values of $|x| > 4$. With this adjustment, only a maximum number of 3 shifts (i.e., 3 extra shift operations) would suffice to give a good approximation and thus reducing the number of shift operations as compared to [8]. A good approximation using the multistep approximation can continue to be achieved by adjusting the number of integer shifts to $\lfloor 0.5x \rfloor$ to accommodate the change in the replacement of the constant 0.1635 in the multistep region. Note that the

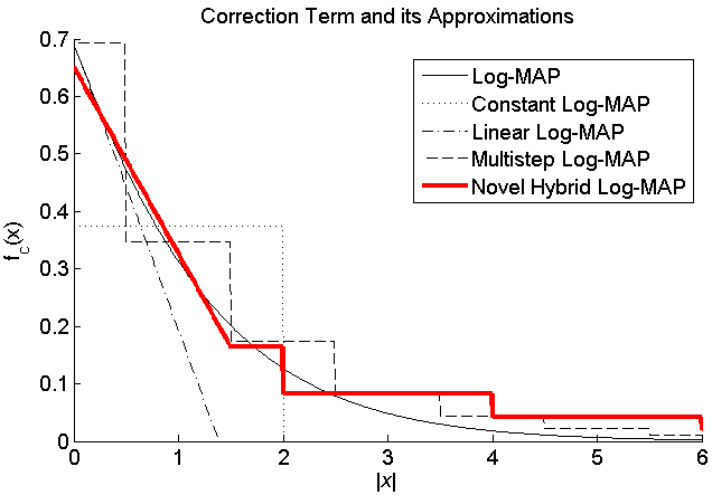

Fig. 1. Approximations to the correction function in Equ. (1).

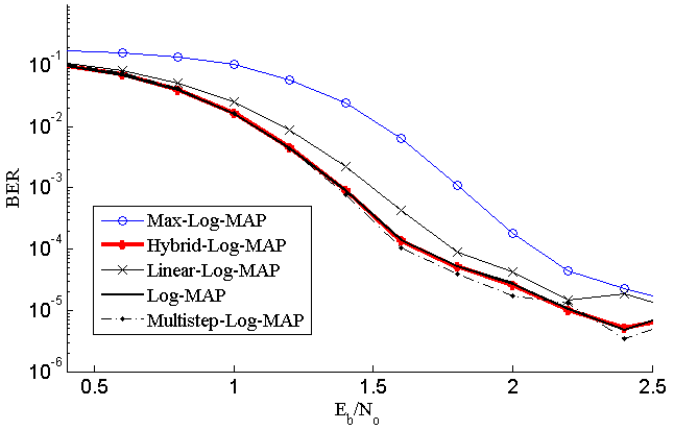

Fig. 2. BER performances for 16 states Turbo decoder with different correction functions.

calculation for $\lfloor 0.5x \rfloor$ can be easily achieved by bit shifting as well. To facilitate fast computation, the value of $0.5x$ can be performed by implementing a dedicated logic circuit to readout values of $0.5x$ which provides a fast combinational logic solution as compared to the shift register method where it is sequential.

V. Simulation Parameters and Results

Two 16 state parallel concatenated Log-MAP turbo decoder with generator polynomial $g[23, 33]$, implemented in 5 iterations, with overall rate of $1/2$, and a random interleaver of size 1000 bits was used in the simulation over the AWGN channel for 10,000 bits modulated using BPSK.

The BER performance for the HLM algorithm including Log-MAP, Max-Log-MAP, Linear Log-MAP, and Multistep Log-MAP are presented in Fig. 2. The HLM algorithm is shown to have the closest performance to the exact Log-MAP solution. The HLM algorithm outperforms other algorithm especially in lower SNRs shown in Fig. 3. This is because the Log-MAP algorithm is sensitive to the SNR [8]. At high SNR,

 ISBN: 978-1-61208-140-3

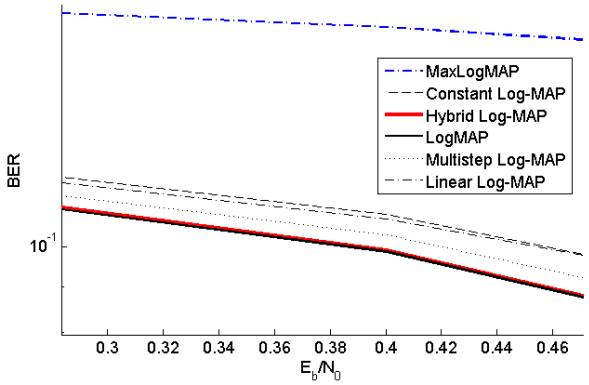

Fig. 3. BER performances at low E_b/N_0.

the performance of the turbo code approaches the Max-Log-MAP performance and does not rely heavily on the correction function. However, if the SNR is low, the decoder considers the *a priori* or *extrinsic* information from the previous decoder more. The *a priori* is Gaussian distributed with increased number of iteration [9] which causes an increased distribution of argument $|x|$ in the correction function to regions close to zero where it is most effective. Due to the good fit of the HLM for smaller arguments of $|x|$, the HLM algorithm performs better than other approximations in low SNR. The sensitivity of the Log-MAP algorithm to SNR is also more pronounced in encoders with more memory elements [8].

VI. Conclusion

The exact Log-MAP algorithm involves computationally intensive operations in order to attain the ideal performance. Neglecting the correction function, the Log-MAP algorithm reduces to the simple Max-Log-MAP algorithm. However, this rough approximation gives a capacity loss, and hence the correction term will have to be included or approximated which gives us the suboptimal Log-MAP algorithm. The novel HLM algorithm is a suboptimal Log-MAP solution that achieves nearly identical performance to the Log-MAP algorithm. The hybrid approximation offers a simple implementation on hardware involving shift registers, multiplications, comparators, and addition operations. In addition, we also show that the hybrid algorithm outperforms existing Log-MAP based algorithm.

Acknowledgment

The authors would like to thank Tunku Abdul Rahman College and its staff for useful discussion and feedback.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," *IEEE International Conference on Communications*, May 1993.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Information Theory*, vol. 20, pp. 284 – 287, Mar 1974.

[3] 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*, vol. 2, pp. 1009–1013, Jun 1995.

[4] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electronic Letters, IET Journal*, vol. 34, pp. 1577 – 1578, Aug 1998.

[5] S. Talakoub, L. Sabeti, B. Shahrrava, and M. Ahmadi, "A linear log-MAP algorithm for turbo decoding and turbo equalization," *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, vol. 1, pp. 182 – 186, Aug 2005.

[6] Hao Wang, Hongwen Yang, and Dacheng Yang, "Improved log-MAP decoding algorithm for turbo-like codes," *IEEE Communications Letters*, vol. 10, pp. 186 – 188, Mar 2006.

[7] Lin Zhang and Shun-Zheng Yu, "A simplified log-MAP turbo decoder by fitting method," *IEEE International Conference on Advanced Communication Technology*, vol. 2, pp. 854–857, Jul 2005.

[8] M. A. Khalighi, "Effect of mismatched SNR on the performance of log-MAP turbo detector," *IEEE Trans. Vehicular Technology*, vol. 52, pp. 1386 – 1397, Sept 2003.

[9] M. Fu, "Stochastic analysis of turbo decoding," *IEEE Trans. Information Theory*, vol. 51, pp. 81 – 100, Jan 2005.

 ISBN: 978-1-61208-140-3