Performance of Bit-Level Decoding Algorithms for Binary LTE Turbo Codes with Early Stopping

Yogesh BEEHARRY[1], Tulsi Pawan FOWDUR [1], Krishnaraj M. S. SOYJAUDAH [1]

[1]Department of Electrical and Electronic Engineering, University of Mauritius, Réduit, Mauritius
y.beeharry@uom.ac.mu, p.fowdur@uom.ac.mu, ssoyjaudah@uom.ac.mu

Abstract: *Turbo codes apply an iterative message passing mechanism between two concatenated decoders. Their astounding performance has led to their widespread adoption in several communication standards such as Long Term Evolution, and Code Division for Multiple Access 2000. This paper gives an overview of three bit-level decoding Max-Log-MAP algorithms with Sign Difference Ratio-based early stopping that can be used with Long Term Evolution Turbo Codes. A detailed computational complexity analysis is given for the three methods. It is observed that the complexity of the decoding methods decrease significantly as the E_b/N_0 increases when Sign Difference Ratio is used. Moreover, the Bit Error Rate performance of the methods was also assessed and compared for different modulation schemes. Results show that the different methods perform differently in the waterfall and error-floor regions with different modulation schemes. Moreover, the computational complexity analysis show that Method 2 requires fewer overall number of computations compared to Methods 1 and 3.*

Keywords: *LTE Turbo Codes, Max-Log-MAP Turbo Decoding, Computational Complexity Analysis.*

1. Introduction

Turbo codes are a class of error correcting codes in coding theory and consist of two concatenated decoders passing messages iteratively between each other. They are used in several communication standards such as Long Term Evolution (LTE), and Code Division for Multiple Access (CDMA) 2000. The Turbo decoding algorithms are the Maximum A-Posteriori Probability (MAP), Logarithmic MAP (Log-MAP), and the Maximum Log-MAP (Max Log-MAP) algorithms. The MAP algorithm was the initially presented Turbo decoding algorithm in [1]. However, due to its numerical instability and high computational complexity, the Log-MAP algorithm was proposed by researchers. The Max Log-MAP algorithm was brought forward with the aim of having additional reduction in computational complexity. The trade-off accompanied by this reduction is a slight degradation in error performance. There exists this computational complexity issue of Turbo decoding algorithms and several papers have proposed variants to address this problem. In [2], a complexity analysis with three symbol-level decoding algorithms for duo-binary Turbo codes was performed. The work in [3] focussed on the decoder complexity for LTE downlink Turbo code and proposed the application of a decoding framework with adaptive complexity. The author of [4] has proposed a new substitute to the conventional Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm for Turbo decoding. It has been shown that the proposed algorithm which is compatible with all Turbo codes, can provide throughputs and latencies of up to 6.86 times better than the conventional algorithms when employed with Turbo codes of the LTE and WiMAX standards with a trade-off of slightly increased resource requirement and computational complexity. The authors of [5], investigated the computational complexity of Turbo decoding with multiple-input multiple-output (MIMO) detection. Results demonstrated the suitability of the proposed decoder for parallel and pipeline architectures which can meet the requirements for high throughput.

Schemes like iterative detection or early stopping are employed in the decoding process to prevent unnecessary additional iterations with the aim to reduce the decoding complexity of Turbo codes. For example, in [6], the authors proposed early stopping mechanisms using a predicted decoding threshold. These techniques have been categorised into two groups. One is soft-bit decision based, such as absolute LLR measurements [7] and Cross Entropy (CE) [8]. The second class is hard-bit decision based, such as Sign Change Ratio (SCR) [9] and Sign-Difference Ratio (SDR) [10]. These mechanisms help reducing the computational complexity by not performing additional unnecessary iterations. The report of [11] presents a stopping criterion used with the Turbo decoder of the LTE standard with the objective of reducing the power and energy consumption. A configuration of the decoder to use 1, 2, 4, 8, or 16 MAP decoders in parallel can be performed. This has shown to

Received on: 16.01.2017
Accepted on: 23.06.2017

help achieve a throughput of 110Mb/s for 7 iterations when operating at a clock frequency of 200 MHz. The work of [12] proposed two early stopping mechanisms based on the regression angle computed at every half-iteration and the Pearson's correlation coefficient for the low and high E_b/N_0 range respectively.

This paper gives an overview of bit-level decoding Max-Log-MAP algorithms with the different equations used for Binary LTE Turbo codes together with SDR-based early stopping mechanism. The computations involved in the three decoding techniques are explicitly detailed and an analysis of the complexity has been provided. Simulation results with different modulation schemes reveal that the different decoding techniques perform differently in the waterfall and error floor regions with different modulation schemes.

The organisation of the paper is as the following. The different decoding methods and results for LTE Turbo codes with Max Log-MAP decoding algorithm are presented in Section 2. The simulation results are presented in Section 3 and Section 4 concludes the paper.

2. Binary LTE Turbo Codes

Figure 1 depicts the complete encoding structure for Binary LTE Turbo codes [13], [14].

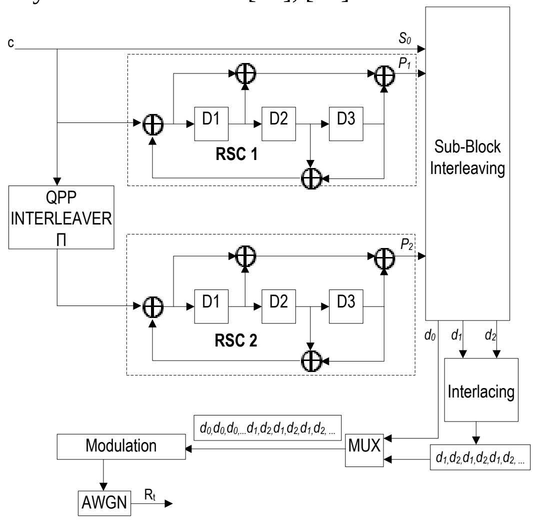

Figure 1. Binary LTE Turbo Codes Transmitter System [13]

The information bits are sent to an LTE Turbo encoder consisting of a parallel concatenation of two Recursive Systematic Convolutional (RSC) encoders, RSC1 and RSC2, separated by an interleaver (Π). A Quadratic Polynomial Permutation (QPP) interleaver is employed in the design of the LTE Turbo encoding system. The interleaved permutation with a QPP interleaver is obtained using a quadratic polynomial. The relationship between the output index z and input index $\pi(z)$ is expressed by the following equation [14]:

$$\pi(z) = (f_1.z + f_2.z^2)\ mod\ K \qquad (1)$$

Where,
f_1 and f_2 are the permutation coefficients,
K is the size of the information block.

After the encoding process, each of the three output streams S_0, P_1, and P_2 is interleaved with its own sub-block interleaver resulting in d_0, d_1, and d_2 respectively. Sub-block interleaving is performed as follows:

a. The input block is written row-wise in a rectangular matrix to form a R_s by C_s matrix.
b. A permutation is performed on the columns of this matrix.
c. A column-wise read is performed on the permuted matrix to obtain the output.

The number of columns, C_s, is fixed to 32 and the number of rows, R_s, is (K / 32). The inter-column permutation is achieved using the Bit-Reversal-Order (BRO) pattern given below:
$Q(j)$= [0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30, 1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31]
$Q(j)$ is the original column position of the j^{th} permuted column.

LTE Turbo coded standard performs multiplexing such that the rearranged systematic bits, d_0, are placed in the beginning followed by bit-by-bit interlacing of the two rearranged parity streams, d_1 and d_2, in order to form a single output buffer [14]. The resulting bit-stream is then sent to the modulation block to be mapped to the corresponding constellation.

The modulated signal is then transmitted over an AWGN channel. The received noisy symbols are demodulated, sub-block de-interleaved and de-interlaced.

Figure 2 shows a typical Turbo decoding framework with an early stopping mechanism. The Turbo decoder is made up of two decoders concatenated with an interleaver similar to the one in the encoder. Decoder 1 accepts $r0$ and $r1$ which are noisy versions of S_0 and P_1, obtained after modulation and transmission over the channel. Decoder 2 accepts $\overline{r0}$ which is the interleaved version of $r0$ and $r2$ which is the noisy version of P_2. $\Lambda_{1e}(t)$and $\Lambda_{2e}(t)$ are the extrinsic information generated by the decoders. $\Lambda_1(t)$ and $\Lambda_2(t)$ are the Log Likelihood Ratio (LLR) output from Decoders 1 and 2 respectively.

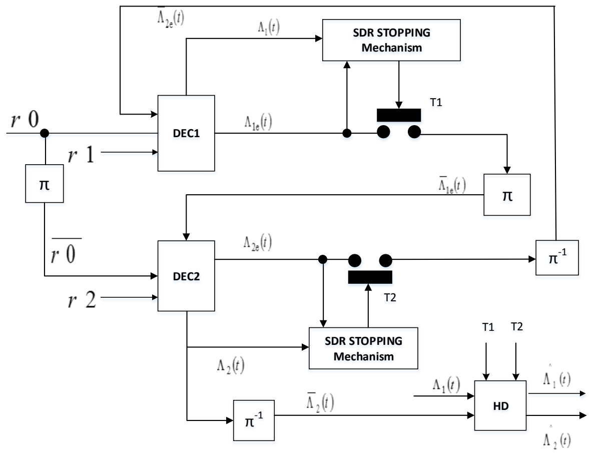

Figure 2. Generic Turbo decoding structure. Source: [1]

The extrinsic and a-posteriori LLRs from each of Decoders 1 and 2 are fed to the SDR stopping mechanism block. A signal T1 is generated to stop the decoding at Decoder 1 and a signal T2 is generated to stop the decoding at Decoder 2. The scale factor for each decoder d at iteration n is computed as follows:

$$S_{dn} = \frac{1}{\mathrm{N}}\sum_{t=1}^{N} f\left(\wedge_{de}^{(n)}, \wedge_{d}^{(n)}\right) \tag{2}$$

Where,
$f\left(\wedge_{de}^{(n)}, \wedge_{d}^{(n)}\right) = 1$ if $\wedge_{de}^{(n)}$, and $\wedge_{d}^{(n)}$have the same sign, otherwise $f\left(\wedge_{de}^{(n)}, \wedge_{d}^{(n)}\right) = 0$,
N is the frame size in bits.
The complexity breakdown for the SDR based scale factor at each half-iteration is shown in Table 2

Table 1. Complexity breakdown for one SDR-based scale factor

	Comp()	Add()	Div()	Total
SDR Scale Factor	N	$N-1$	1	$2N$

Where,
Comp() refers to the Comparison operations,
Add() refers to the Addition operations, and
Div() refers to the Division operations.
Depending on which signal has been generated, the hard decision is performed on the correponding a-posteriori LLR to estimate the transmitted information. The decoding equations for the max Log-MAP algorithm are presented in the sub-sections 2.1, 2.2 and 2.3.

2.1. Max Log-MAP Turbo Decoding: Method 1

The decoding equations based on the Max-Log MAP algorithm used for the first decoder are [14], [13], [15]:

$$\gamma_t^{1(i)}(l', l) = log\left[p_t^1(i).exp\left(-\frac{[r0_t - S_{0_t}]^2 + [r1_t - P_{1_t}]^2}{2\sigma^2}\right)\right] = log[p_t^1(i)] - \left(\frac{[r0_t - S_{0_t}]^2 + [r1_t - P_{1_t}]^2}{2\sigma^2}\right) \tag{3}$$

Where,
$\gamma_t^{1(i)}(l', l)$ is the branch transition probability for decoder 1 from state l' to state l of bit i (i = 0 or 1) at time instant t.
$p_t^1(i)$ is the a-priori probability of bit i resulting from the channel extrinsic information and input to the first decoder.
$r0_t$ *and* $r1_t$ are the soft bits which have been de-mapped and correspond to the bipolar equivalent of the transmitted systematic bits, S_{0_t} and first parity bits, P_{1_t} respectively.
σ^2 is the noise variance.
The complexity breakdown for the branch transition probability is shown in Table 2. The number of computations is for one branch transition.

Table 2. Complexity breakdown for one branch transition metric of Decoder 1 with Method 1

	Log()	Add()	Sub()	Mult()	Total
Branch Transition Metric	1	1	3	2	6

Where,
Log() refers to the Logarithmic operations,
Sub() refers to the Subtraction operations, and
Mult() refers to the Multiplication operations.

The forward recursive variable, $\alpha_t^1(l)$, for a decoder with $\mathrm{M_S}$ states at state l and time t is computed as follows [16] [17]:

$$\alpha_t^1(l) = \max\left(\alpha_{t-1}^1(l) + \gamma_t^{1(i)}(l', l)\right) \;\; for \; 0 \le l' \le \mathrm{M_S} - 1 \tag{4}$$

The max() operator outputs the maximum of the two forward metrics pertaining to each bit i for each of the state l. The complexity breakdown for the forward recursive variable at each state l is shown in Table 3.

Table 3. Complexity breakdown for one Forward recursive variable of Decoder 1 with Method 1

	Max()	Add()	Total
Forward Recursive Variable	1	2	3

The backward recursive variable, $\beta_t^1(l)$, is computed as follows [16] [17]:

$$\beta_t^1(l) = \max\left(\beta_{t-1}^1(l') + \gamma_t^{1(i)}(l, l')\right), for \; 0 \le l' \le \mathrm{M_S} - 1 \tag{5}$$

The complexity breakdown for the backward recursive variable for each state l' is shown in Table 4.

Table 4. Complexity breakdown for one Backward recursive vatiable of Decoder 1 with Method 1

	Max()	Add()	Total
Backward Recursive Variable	1	2	3

The Log-Likelihood Ratio (LLR), $\Lambda_1^{(r)}(t)$ for the first decoder at time t and iteration r is computed as [16] [17]:

$$\Lambda_1^{(r)}(t) = \max\left(\alpha_{t-1}^1(l') + \gamma_t^{1(1)}(l, l') + \beta_t^1(l)\right) - \max\left(\alpha_{t-1}^1(l') + \gamma_t^{1(0)}(l, l') + \beta_t^1(l)\right) \;\; for \; 0 \le l' \le \mathrm{M_S} - 1 \tag{6}$$

The complexity breakdown for the LLR variable is shown in Table 5.

Table 5. Complexity breakdown for the Log-Likelihood Ratio of Decoder 1 with Method 1

	Max()	Add()	Sub()	Total
Log Likelihood Ratio	2	32	1	35

The extrinsic information $\Lambda_{1e}^{(r)}(t)$ for the first decoder at time *t* and iteration *r* is computed as [16] [17]:

$$\Lambda_{1e}^{(r)}(t) = \Lambda_{1}^{(r)}(t) - \frac{2}{\sigma^2} r0_t - \bar{\Lambda}_{2e}^{(r-1)}(t) \quad (7)$$

Where,
$\bar{\Lambda}_{2e}^{(r-1)}(t)$ is the de-interleaved version of the extrinsic information for the second decoder at time *t* and iteration (*r*-1). For the first iteration, $\bar{\Lambda}_{2e}^{(r-1)}(t)$, is initialized to zero. The complexity breakdown for an extrinsic LLR variable is shown in Table 6.

Table 6. Complexity breakdown for an extrinsic LLR of Decoder 1 with Method 1

	Sub()	Total
Extrinsic Log Likelihood Ratio	2	2

The a-priori probability for the second decoder, $p_t^2(i)$ is computed as follows:

$$p_t^2(i) = \begin{cases} \frac{\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)}{1+\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)} \; for\ i = 1 \\ \frac{1}{1+\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)} \; for\ i = 0 \end{cases} \quad (8)$$

Where, $\bar{\Lambda}_{1e}^{(r)}(t)$ is the interleaved version of $\Lambda_{1e}^{(r)}(t)$.

The complexity breakdown required for the a-priori probability is shown in Table 7.

Table 7. Complexity breakdown for the a-priori probability of Decoder 1 with Method 1

	Exp()	Add()	Div()	Total
Log Likelihood Ratio	3	2	2	7

Where,
Exp() refers to the exponential operations, and
Div() refers to the division operations.

The computations for the second decoder can now be started. The decoding equations based on the Max-Log MAP algorithm used for the second decoder are as presented in [16] [17]. The branch metric [15] is given as:

$$\gamma_t^{2(i)}(l',l) = log\left[p_t^1(i).exp\left(-\frac{[\overline{r0}_t - \overline{S_0}_t]^2 + [r2_t - P_{2_t}]^2}{2\sigma^2}\right)\right] = log[p_t^2(i)] - \left(\frac{[\overline{r0}_t - \overline{S_0}_t]^2 + [r2_t - P_{2_t}]^2}{2\sigma^2}\right) \quad (9)$$

Where,
$\gamma_t^{2(i)}(l',l)$ is the branch transition probability for decoder 2 from state *l*' to state *l* of bit *i* (*i* = 0 or 1) at time instant *t*.
$p_t^2(i)$ is the a-priori probability of bit *i* derived from the channel extrinsic information and input to the second decoder.
$\overline{r0}_t$ *and* $r2_t$ are the de-mapped soft bits corresponding to the bipolar equivalent of the interleaved systematic bits, $\overline{S_{0_t}}$ and second parity bits, P_{2_t} respectively.
σ^2 is the noise variance.

The complexity breakdown for the branch transition probability is similar to that for the first decoder as shown in Table 2. The forward recursive variable, $\alpha_t^2(l)$, for a decoder with M_S states at state *l* and time *t* is computed as the following [16] [17]:

$$\alpha_t^2(l) = \max\left(\alpha_{t-1}^2(l) + \gamma_t^{2(i)}(l',l)\right), \; for\ 0 \le l' \le M_S - 1 \quad (10)$$

The complexity breakdown for the forward recursive variable is similar to that for the first decoder as shown in Table 3. The backward recursive variable, $\beta_t^2(l)$, is computed as follows [16] [17]:

$$\beta_t^2(l) = \max\left(\beta_{t-1}^2(l') + \gamma_t^{2(i)}(l,l')\right) \; for\ 0 \le l' \le M_S - 1 \quad (11)$$

The complexity breakdown for the backward recursive variable is similar to that for the first decoder as shown in Table 4. The Log-Likelihood Ratio (LLR), $\Lambda_2^{(r)}(t)$ for the second decoder at time *t* and iteration *r* is computed as [16] [17]:

$$\Lambda_2^{(r)}(t) = \max\left(\alpha_{t-1}^2(l') + \gamma_t^{2(1)}(l,l') + \beta_t^2(l)\right) - \max\left(\alpha_{t-1}^2(l') + \gamma_t^{2(0)}(l,l') + \beta_t^2(l)\right) \; for\ 0 \le l' \le M_S - 1 \quad (12)$$

The complexity breakdown for a LLR variable is similar to that for the first decoder as shown in Table 4. The extrinsic information, $\Lambda_{2e}^{(r)}(t)$, for the second decoder at time *t* and iteration *r* is computed as [16] [17]:

$$\Lambda_{2e}^{(r)}(t) = \Lambda_{2}^{(r)}(t) - \frac{2}{\sigma^2} \overline{r0}_t - \bar{\Lambda}_{1e}^{(r-1)}(t) \quad (13)$$

Where, $\bar{\Lambda}_{1e}^{(r-1)}(t)$ is the interleaved extrinsic information for the first decoder at time *t* and iteration *r*.

The complexity breakdown for an extrinsic LLR variable is similar to that for the first decoder as shown in Table 6. The a-priori probability for the first decoder, $p_t^1(i)$ is computed as follows:

$$p_t^1(i) = \begin{cases} \frac{\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)}{1+\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)} \; for\ i = 1 \\ \frac{1}{1+\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)} \; for\ i = 0 \end{cases} \quad (14)$$

Where,
$\bar{\Lambda}_{2e}^{(r)}(t)$ is the de-interleaved version of $\Lambda_{2e}^{(r)}(t)$.

The number of computations required for the a-priori probability is similar to that for the first decoder as shown in Table 7. For the second iteration, the same computations are repeated by the first and second decoders with the updated values of the a-priori probabilities.

2.2. Max Log-MAP Turbo Decoding: Method 2

In [18], an approximate Log BCJR algorithm has been presented. The decoding equations are presented next.

$$y_t^r = log\left(\frac{P(r0_t=1)}{P(r0_t=0)}\right) = r0_t + \bar{\Lambda}_{2e}^{(r-1)}(t) \qquad (15)$$

Where,
y_t^r is the soft decisions of the received bit $r0_t$.
$\bar{\Lambda}_{2e}^{(r-1)}(t)$ is the de-interleaved extrinsic LLR from decoder 2 in the previous iteration.

The transition state branch metrics are computed as below:

$$\gamma_{s,t}^{1(i)}(l',l) = y_t^r \qquad (16)$$

$$\gamma_{p,t}^{1(i)}(l',l) = r1_t \qquad (17)$$

Where,
$\gamma_{s,t}^{1(i)}(l',l)$ is the branch transition metric for the systematic information,
$\gamma_{p,t}^{1(i)}(l',l)$ is the branch transition metric for the parity information, and
$r1_t$ is the received noisy parity information from the first encoder.

The complexity breakdown for one branch transition metric is shown in Table 8.

Table 8. Complexity breakdown for a branch transition metric of Decoder 1 with Method 2

	Add()	Total
Branch Transition Metric	1	1

The forward recursive variable, $\alpha_t^1(l)$, for a decoder with M_S states at state *l* and *time t* is computed as follows [18]:

$$\alpha_t^1(l) = \max\left(\alpha_{t-1}^1(l') + \gamma_{s,t}^{1(i)}(l',l) + \gamma_{p,t}^{1(i)}(l',l)\right) \; for\ 0 \le l' \le M_S - 1 \qquad (18)$$

The number of computations required for one forward recursive variable is shown in Table 9.

Table 9. Complexity breakdown for a forward recursive variable of Decoder 1 with Method 2

	Max()	Add()	Total
Forward Recursive Variable	1	4	5

The backward recursive variable, $\beta_t^1(l)$, is computed as follows [18]:

$$\beta_t^1(l) = \max\left(\beta_{t-1}^1(l') + \gamma_{s,t}^{1(i)}(l,l') + \gamma_{p,t}^{1(i)}(l',l)\right) \; for\ 0 \le l' \le M_S - 1 \qquad (19)$$

The complexity breakdown required for one backward recursive variable is shown in Table 10.

Table 10. Complexity breakdown for a backward recursive variable of Decoder 1 with Method 2

	Max()	Add()	Total
Backward Recursive Variable	1	4	5

The un-coded extrinsic log confidences are computed as the following:

$$\delta_t^{1(i)}(l,l') = \alpha_t^1(l) + \gamma_{p,t}^{1(i)}(l',l) + \beta_t^1(l) \qquad (20)$$

The complexity breakdown for the un-coded extrinsic log confidences is shown in Table 11.

Table 11. Complexity breakdown for the un-coded extrinsic log confidences of Decoder 1 with Method 2

	Add()	Total
Un-coded extrinsic Log confidences	2	2

The un-coded Extrinsic Log-Likelihood Ratio (LLR), $\Lambda_{1e}^{(r)}(t)$ for the first decoder at *time t* and iteration *r* is computed as [18]:

$$\Lambda_{1e}^{(r)}(t) = \max\left(\delta_t^{1(1)}(l,l')\right) - \max\left(\delta_t^{1(0)}(l,l')\right) \; for\ 0 \le l' \le M_S - 1 \qquad (21)$$

The complexity breakdown for the un-coded extrinsic LLRs is shown in Table 12.

Table 12. Complexity breakdown for the un-coded extrinsic LLRs of Decoder 1 with Method 2

	Max()	Sub()	Total
Un-coded extrinsic LLRs	2	1	3

The systematic and extrinsic information are interleaved to be fed to the second decoder:

$$\overline{\Lambda_{1e}^{(r)}}(t) = \pi\left(\Lambda_{1e}^{(r)}(t)\right) \qquad (22)$$

$$\overline{r0_t} = \pi(r0_t) \quad (23)$$

The un-coded a-priori input information for the second decoder is computed as the following:

$$z_t^r = \overline{r0_t} + \overline{\Lambda_{1e}^{(r)}}(t) \quad (24)$$

The second decoder can now be started. The transition state branch metric is computed as below:

$$\gamma_{s,t}^{2(i)}(l', l) = z_t^r \quad (25)$$

$$\gamma_{p,t}^{2(i)}(l', l) = r2_t \quad (26)$$

$\gamma_{s,t}^{2(i)}(l', l)$ is the branch transition metric for the systematic information and second decoder
$\gamma_{p,t}^{2(i)}(l', l)$ is the branch transition metric for the parity information and second decoder

The complexity breakdown for the branch transition probability is similar to that for the first decoder as shown in Table 7. The forward recursive variable, $\alpha_t^2(l)$, for a decoder with M_S states at state *l* and *time t* is computed as follows [18]:

$$\alpha_t^2(l) = \max\left(\alpha_{t-1}^2(l') + \gamma_{s,t}^{2(i)}(l', l) + \gamma_{p,t}^{2(i)}(l', l)\right) \; for\; 0 \le l' \le M_S - 1 \quad (27)$$

The complexity breakdown for the forward recursive variable is similar to that for the first decoder as shown in Table 8. The backward recursive variable, $\beta_t^1(l)$, is computed as follows [18]:

$$\beta_t^2(l) = \max\left(\beta_{t-1}^2(l') + \gamma_{s,t}^{2(i)}(l, l') + \gamma_{p,t}^{2(i)}(l', l)\right) \; for\; 0 \le l' \le M_S - 1 \quad (28)$$

The complexity breakdown for the backward recursive variable is similar to that for the first decoder as shown in Table 10 The un-coded extrinsic log confidences are computed as the following:

$$\delta_t^{2(i)}(l, l') = \alpha_t^2(l) + \gamma_{p,t}^{2(i)}(l', l) + \beta_t^2(l) \quad (29)$$

The complexity breakdown for the un-coded extrinsic log confidences is similar to that for the first decoder as shown in Table 11. The un-coded Extrinsic Log-Likelihood Ratio (LLR), $\Lambda_{2e}^{(r)}(t)$ for the second decoder at *time t* and iteration *r* is computed as [18]:

$$\Lambda_{2e}^{(r)}(t) = \max\left(\delta_t^{2(1)}(l, l')\right) - \max\left(\delta_t^{2(0)}(l, l')\right) \; for\; 0 \le l' \le M_S - 1 \quad (30)$$

The complexity breakdown for the un-coded extrinsic LLRs is similar to that for the first decoder as shown in Table 12. The extrinsic information is de-interleaved to be fed to the first decoder:

$$\overline{\Lambda_{2e}^{(r)}}(t) = \pi^{-1}\left(\Lambda_{2e}^{(r)}(t)\right) \quad (31)$$

The a-priori LLR to be fed to the first decoder is computed as:

$$y_t^{r+1} = r0_t + \overline{\Lambda_{2e}^{(r)}}(t) \quad (32)$$

The a-posteriori LLR computation to be used in the hard-decision process is as the following:

$$\Lambda_2^{(r)}(t) = \overline{\Lambda_{2e}^{(r)}}(t) + r0_t + \Lambda_{1e}^{(r-1)}(t) \quad (33)$$

For the second iteration, the same computations are repeated by both decoders with the updated values of the a-priori LLRs.

2.3. Max Log-MAP Turbo Decoding: Method 3

This decoding method is depicted in [19] and [20].

$$\gamma_t^{1(i)}(l', l) = \left(\frac{S_{0_t}}{2}\right).\left(\overline{\Lambda_{2e}^{(r-1)}}(t)\right) + \frac{L_c}{2}\left[(r0_t).\left(S_{0_t}\right) + (r1_t).\left(P_{1_t}\right)\right] \quad (34)$$

Where,
$\gamma_t^{1(i)}(l', l)$ is the branch transition probability for decoder 1 from state *l'* to state *l* of bit *t* (*t* = 0 or 1) at time instant *t*.
S_{0_t} is the systematic information bit at time instant *t*.
$r0_t$ *and* $r1_t$ are the soft bits which have been de-mapped and correspond to the bipolar equivalent of the transmitted systematic bits, S_{0_t} and first parity bits, P_{1_t} respectively.
$L_c = \frac{2}{\sigma^2}$ is the channel reliability estimate and σ^2 is the noise variance.

The complexity breakdown for one branch transition probability is shown in Table 13.

Table 13. Complexity breakdown for a branch transition metric of Decoder 1 with Method 3

	Add()	Mult()	Div()	Total
Branch Transition Metric	2	3	1	6

The forward recursive variable, $\alpha_t^1(l)$, for a decoder with M_S states at state *l* and *time t* is computed as follows [16] [17]:

$$\alpha_t^1(l) = \max\left(\alpha_{t-1}^1(l) + \gamma_t^{1(i)}(l', l)\right), \; for\; 0 \le l' \le M_S - 1 \quad (35)$$

The complexity breakdown for the forward recursive variable is similar to that for the first decoder with Method 1 as shown in Table 3. The backward recursive variable, $\beta_t^1(l)$, is computed as follows [16] [17]:

$$\beta_t^1(l) = \max\left(\beta_{t-1}^1(l') + \gamma_t^{1(i)}(l, l')\right) \; for\; 0 \le l' \le M_S - 1 \quad (36)$$

The complexity breakdown for one backward recursive variable is similar to that for the first decoder of Method 1 as shown in Table 4. The Log-Likelihood Ratio (LLR), $\Lambda_1^{(r)}(t)$ for the first decoder at *time t* and iteration *r* is computed as [16] [17]:

$$\Lambda_1^{(r)}(t) = \max\left(\alpha_{t-1}^1(l') + \gamma_t^{1(1)}(l,l') + \beta_t^1(l)\right) - \max\left(\alpha_{t-1}^1(l') + \gamma_t^{1(0)}(l,l') + \beta_t^1(l)\right) \; for\ 0 \le l' \le M_S - 1 \quad (37)$$

The complexity breakdown for the LLRs is similar to that for the first decoder of Method 1 as shown in Table 5. The extrinsic information $\Lambda_{1e}^{(r)}(t)$ for the first decoder at *time t* and iteration *r* is computed as [16] [17]:

$$\Lambda_{1e}^{(r)}(t) = \Lambda_1^{(r)}(t) - \frac{2}{\sigma^2} r0_t - \bar{\Lambda}_{2e}^{(r-1)}(t) \quad (38)$$

Where,
$\bar{\Lambda}_{2e}^{(r-1)}(t)$ is the de-interleaved version of the extrinsic information for the second decoder at *time t* and iteration (*r*-1).

The complexity breakdown for the extrinsic LLRs is similar to that for the first decoder of Method 1 as shown in Table 6. The a-priori probability for the second decoder, $p_t^2(i)$ is computed as follows:

$$p_t^2(i) = \begin{cases} \dfrac{\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)}{1+\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)} \; for\ i = 1 \\ \dfrac{1}{1+\exp\left(\bar{\Lambda}_{1e}^{(r)}(t)\right)} \; for\ i = 0 \end{cases} \quad (39)$$

Where,
$\bar{\Lambda}_{1e}^{(r)}(t)$ is the interleaved version of $\Lambda_{1e}^{(r)}(t)$.

The complexity breakdown for the a-priori probability is similar to that for the first decoder of Method 1 as shown in Table 7. The decoding operation for the second decoder can now be started. The decoding equations based on the Max-Log MAP algorithm used for the second decoder are [19], [20]:

$$\gamma_t^{2(i)}(l',l) = \left(\frac{\overline{S_{0_t}}}{2}\right).\left(\overline{\Lambda_{1e}^{(r-1)}}(t)\right) + \frac{L_c}{2}\left[(\overline{r0}_t).\left(\overline{S_{0_t}}\right) + (r2_t).\left(P_{2_t}\right)\right] \quad (40)$$

Where,
$\gamma_t^{2(i)}(l',l)$ is the branch transition probability for decoder 2 from state *l'* to state *l* of bit *t* (*t* = 0 or 1) at time instant *t*.
$\overline{S_{0_t}}$ is the interleaved systematic information bit at time instant *t*.
$\overline{r0}_t\ and\ r2_t$ are the de-mapped soft bits corresponding to the bipolar equivalent of the interleaved systematic bits, $\overline{S_{0_t}}$ and second parity bits, P_{2_t} respectively.
$L_c = \frac{2}{\sigma^2}$ is the channel reliability estimate and σ^2 is the noise variance.

The complexity breakdown for the branch transition probability is similar to that for the first decoder as shown in Table 13. The forward recursive variable, $\alpha_t^2(l)$, for a decoder with $\mathrm{M_S}$ states at state *l* and *time t* is computed as the following [16] [17]:

$$\alpha_t^2(l) = \max\left(\alpha_{t-1}^2(l) + \gamma_t^{2(i)}(l',l)\right) \; for\ 0 \le l' \le M_S - 1 \quad (41)$$

The complexity breakdown for the forward recursive variable is shown in Table 3. The backward recursive variable, $\beta_t^2(l)$, is computed as follows [16] [17]:

$$\beta_t^2(l) = \max\left(\beta_{t-1}^2(l') + \gamma_t^{2(i)}(l,l')\right) \; for\ 0 \le l' \le M_S - 1 \quad (42)$$

The complexity breakdown for the backward recursive variable is shown in Table 4. The Log-Likelihood Ratio (LLR), $\Lambda_2^{(r)}(t)$ for the second decoder at *time t* and iteration *r* is computed as [16] [17]:

$$\Lambda_2^{(r)}(t) = \max\left(\alpha_{t-1}^2(l') + \gamma_t^{2(1)}(l,l') + \beta_t^2(l)\right) - \max\left(\alpha_{t-1}^2(l') + \gamma_t^{2(0)}(l,l') + \beta_t^2(l)\right) \; for\ 0 \le l' \le M_S - 1 \quad (43)$$

The complexity breakdown for the LLRs is shown in Table 5. The extrinsic information, $\Lambda_{2e}^{(r)}(t)$, for the second decoder at *time t* and iteration *r* is computed as [19], [20]:

$$\Lambda_{2e}^{(r)}(t) = \Lambda_2^{(r)}(t) - \frac{2}{\sigma^2}\overline{r0}_t - \bar{\Lambda}_{1e}^{(r-1)}(t) \quad (44)$$

Where,
$\bar{\Lambda}_{1e}^{(r-1)}(t)$ is the interleaved extrinsic information for the first decoder at *time t* and iteration *r*-1.

The complexity breakdown for the extrinsic LLRs is shown in Table 6. The a-priori probability for the first decoder, $p_t^1(i)$ is computed as follows:

$$p_t^1(i) = \begin{cases} \dfrac{\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)}{1+\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)} \; for\ i = 1 \\ \dfrac{1}{1+\exp\left(\bar{\Lambda}_{2e}^{(r)}(t)\right)} \; for\ i = 0 \end{cases} \quad (45)$$

The complexity breakdown for the a-priori probability is shown in Table 7. For the second iteration, the same computations are repeated by the first and second decoders with the updated values of the a-priori probabilities.

2.4. Analysis of Computational Complexity

In this section, a comparison of the computational complexities for the three decoding methods of binary LTE Turbo codes has been performed. The complexity breakdowns at each half-iteration for Methods 1, 2 and 3 with respect to the number of computations are shown in Table 14,

Table 15 and

Table 16 respectively. An explanation on the values obtained for the metrics of Method 1 computed over one half-iteration is given next. Table 2 shows the number of computations for a single branch transition metric of equation (3). There are N transitions in all in the trellis and each transition in the trellis consists of 16 branch transition metrics. Hence, the branch transition metrics are scaled by a factor of 16 and N. The number of computations for the forward recursive variable of equation (4) and backward recursive variable of equation (5) as shown in Table 3 and Table 4 respectively are multiplied by 8 and N since there are 8 states for each transition over a total packet length of N. Table 5 shows the computations for the a-posteriori LLRs of equation (6) which have to be multiplied by N for the whole packet length. The computations of the extrinsic LLRs of equation (7) as shown in Table 6 are also multiplied by N. Similarly, the computations of the a-posteriori probabilities of equations (8) as shown in Table 7 are multiplied by the packet length, N.

Table 14. Computational complexity breakdown for Method 1 at one half-iteration

	Comp()	Log()	Exp()	Max()	Add()	Sub()	Mult()	Div()	Total
Branch Transition Metric	0	1 x 16 x N	0	0	1 x 16 x N	3 x 16 x N	2 x 16 x N	0	112 N
Forward Metric	0	0	0	1 x 8 x N	2 x 8 x N	0	0	0	24 N
Backward Metric	0	0	0	1 x 8 x N	2 x 8 x N	0	0	0	24 N
A-Posteriori LLR	0	0	0	2 x N	32 x N	1 x N	0	0	35 N
Extrinsic LLR	0	0	0	0	0	2 x N	0	0	2 N
A-Posteriori Probabilities	0	0	3 x N	0	2 x N	0	0	2 x N	7 N
SDR Scale Factor	N	0	0	0	$N-1$	0	0	1	$2N$
TOTAL	N	16 N	3 N	18N	83 N -1	51N	32N	2 N +1	206N

The values obtained for the metrics computed over one half-iteration for Method 2 are explained next. The number of computations for the branch transition metric of equations (16–17) shown in Table 8 are multiplied by 8 and N to compute the transitions pertaining to the systematic information bits equal to 1. The number of mathematical computations for the forward recursive variable, backward recursive variable and un-coded extrinsic log confidences are multiplied by the number of states, 8, and the packet length, N. The computations of the extrinsic LLRs of equation (21) as shown in Table 11 are multiplied by the packet length, N.

Table 15. Computational complexity breakdown for Method 2 at one half-iteration

	Comp()	Max()	Add()	Sub()	Div()	Total
Branch Transition Metric	0	0	1 x 8 x N	0	0	8N
Forward Metric	0	1 x 8 x N	4 x 8 x N	0	0	40 N
Backward Metric	0	1 x 8 x N	4 x 8 x N	0	0	40 N
Un-coded Extrinsic Log Confidences	0	0	2 x 8 x N	0	0	16 N
Extrinsic LLR	0	2 x N	0	1 x N	0	3 N
SDR Scale Factor	N	0	$N-1$	0	1	2N
TOTAL	N	18N	89 N -1	1N	1	109 N

The values obtained for the metrics computed for Method 3 over one half-iteration are explained next. The number of mathematical computations for the branch transition metric of equation (34) shown in Table 13 are multiplied by 16 and N as explained for Method 1. The computations for the forward recursive variable, backward recursive variable, a-posteriori LLRs, extrinsic LLRs and a-posteriori probabilities are exactly as explained for Method 1.

Table 16. Computational complexity breakdown for Method 3 at one half-iteration

	Comp()	Exp()	Max()	Add()	Sub()	Mult()	Total
Branch Transition Metric	0	0	0	2 x 16 x N	0	3 x 16 x N	80 N
Forward Metric	0	0	1 x 8 x N	2 x 8 x N	0	0	$24N$
Backward Metric	0	0	1 x 8 x N	2 x 8 x N	0	0	$24N$
A-Posteriori LLR	0	0	2 x N	32 x N	1 x N	0	$35N$
Extrinsic LLR	0	0	0	0	2 x N	0	$2N$
A-Posteriori Probabilities	0	3 x N	0	7 x N	0	0	$10N$
SDR Scale Factor	N	0	0	$N-1$	0	1	$2N$
TOTAL	N	3 N	$18N$	$104N$-1	$3N$	$48N$+1	$177N$

The total number of different computations required for each of the decoding method can be compared to analyse the number of computations as shown in Table 17.

Table 17. Total of the number of computations for the different operation at one half-iteration for the 3 Methods

	Comp()	Log()	Exp()	Max()	Add()	Sub()	Mult()	Div()	TOTAL
Method 1	N	$16N$	3 N	$18N$	$83N$-1	$51N$	$32N$	2 N +1	206 N
Method 2	N	0	0	$18N$	$89N$-1	$1N$	0	1	109 N
Method 3	N	0	3 N	$18N$	$104N$-1	$3N$	$48N$	1	177 N

Table 16 shows that the number of mathematical operations required by Methods 1, 2 and 3 are 7, 3, and 5 respectively. Clearly, Method 2 requires fewer mathematical operations compared to Methods 1 and 3. Method 1 requires $16N$ Logarithm operations while Methods 2 and 3 require none. $3N$ exponential operations are required by bot h Methods 1 and 3 while Method 2 requires none. All the 3 methods require the same number of Max operations, i.e 18 N. Method 3 requires the maximum number of addition operations which is 103 N followed by Method 2 which requires 88N and finally Method 1 which needs $82N$. The number of subtraction operations is highest for Method 1, i.e 51 N followed by Methods 3 and 2 which require 3 N and 1 N subtraction operations respectively. Only Methods 1 and 3 require multiplication operations of 32 N and 48 N respectively. 2 N division operations are required by Method 1 only.

3. Simulation Results

In this section, the performances of the different decoding methods are compared. Three modulation schemes have been used, namely: Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), and 16-Quadrature Amplitude Modulation (QAM). An interleaver size of 6144 bits has been used in all the simulations. The parameters for the LTE Turbo code used are as follows [14]:

Generator: G = [1, g1/g2], where g1 = 15 and g2 = 13 in Octal.

Code-rate = 1/3 and channel model: Complex AWGN.

QPP Interleaver parameters: f_1= 283 and f_2 = 480

Interleaver size, N = 6144 bits.

Maximum number of iterations, T = 12.

The BER performance of binary LTE Turbo codes with BPSK modulation with early stopping is shown in Figure 3. The overall number of computations at each E_b/N_0 for each decoding method is shown in Figure 5. A more detailed breakdown of the total number of computations per mathematical operation is shown in Figure 6.

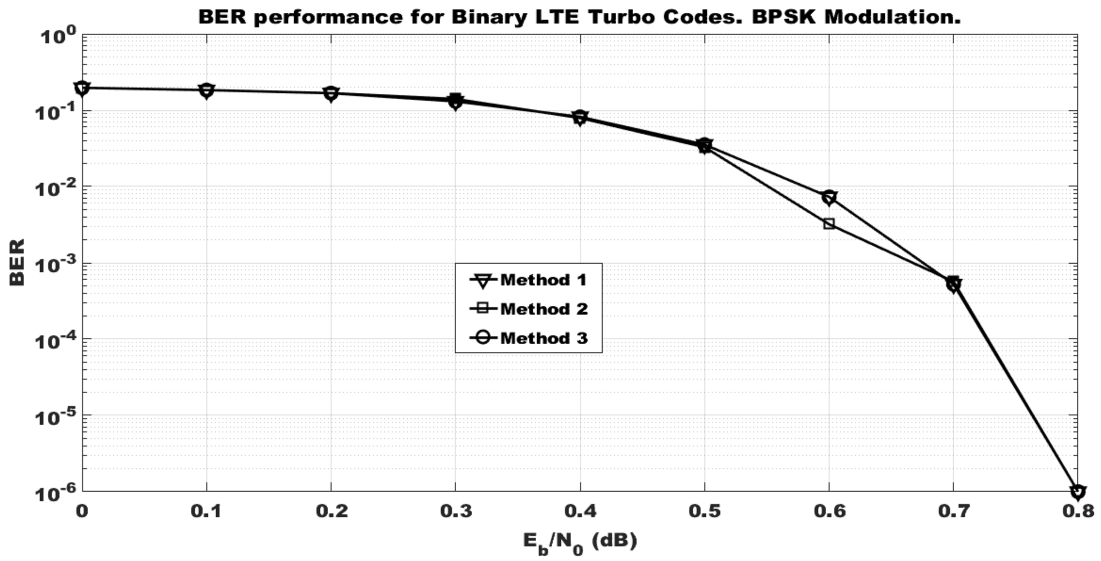

Figure 3. BER performance for Binary LTE Turbo codes with BPSK modulation and SDR stopping

It can be observed from Figure 3 that all 3 decoding methods have similar error performances for almost the whole E_b/N_0 range despite the differences in total number of different mathematical operations required. The performance of binary LTE Turbo codes with SDR based early stopping in terms of the average number of iterations required is shown in Figure 4.

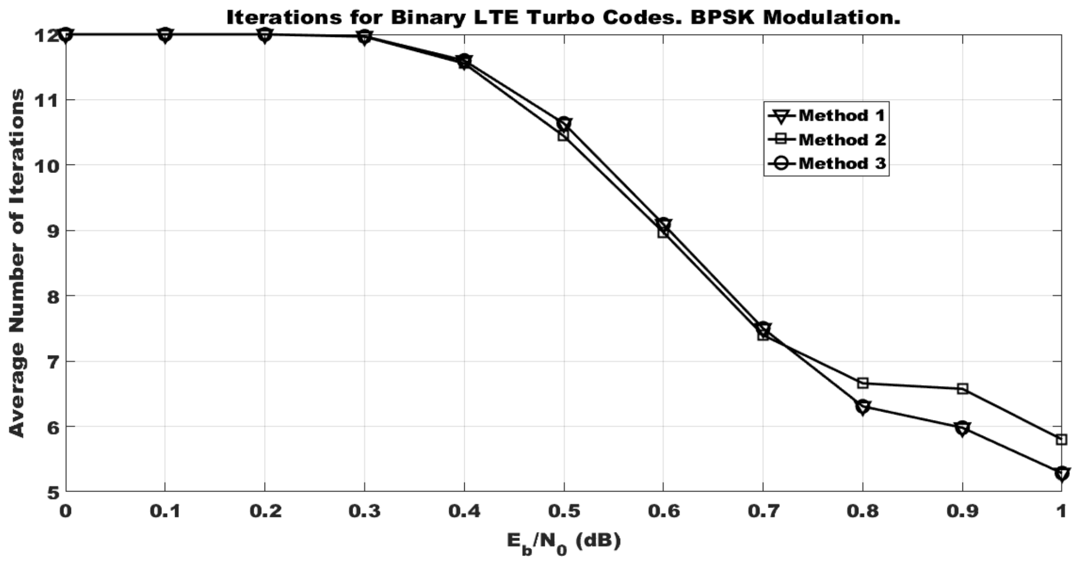

Figure 4. Iterations performance for Binary LTE Turbo codes with BPSK modulation and SDR stopping

It can be observed from Figure 4 that all 3 decoding methods require almost the same number of average decoding iterations for the range $E_b/N_0 \leq 0.7$ dB despite the differences in total number of different mathematical operations required. For the range $E_b/N_0 > 0.7$ dB Methods 1 and 3 require fewer average decoding iterations as compared to Method 2. The overall number of computations for binary LTE Turbo codes with B-PSK modulation and SDR stopping is shown in Figure 5.

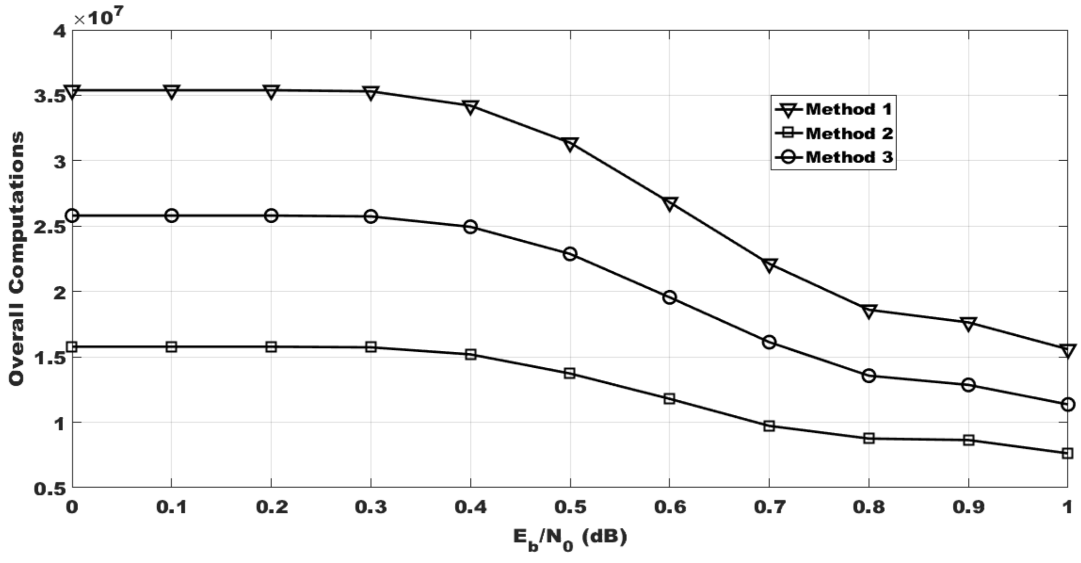

Figure 5. Overall computations for Binary LTE Turbo codes with BPSK modulation and SDR stopping

It can be observed from Figure 5 that all 3 decoding methods have different overall number of computations. Method 2 has the least number of overall computations throughout the whole E_b/N_0 range. Method 3 gives an average gain of 3.3812x10^6 computations over Method 1. Method 2 gives an average gain of 1.1189x10^7 computations over Method 1. Method 2 gives an average gain of 7.8075x10^6 computations over Method 3. The breakdown of the overall number of computations per mathematical operation is shown in Figure 6.

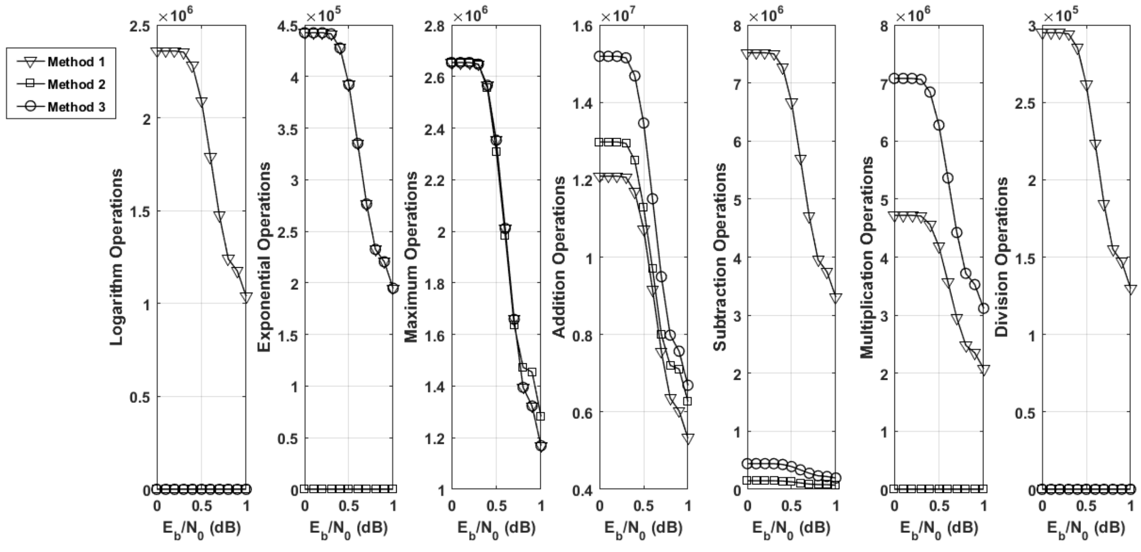

Figure 6. Breakdown of overall number of computation per mathematical operation for Binary LTE Turbo codes with BPSK modulation and SDR stopping

It can be observed from Figure 6 that all 3 decoding methods use approximately the same number of maximum operations throughout the whole E_b/N_0 range. Another observation is that eventhough Method 1 uses fewer addition operations than Methods 2 and 3 over the whole E_b/N_0 range, there are the large numbers of logarithm and division operations which are also used.

The BER performance of binary LTE Turbo codes with Q-PSK modulation with SDR stopping is shown in Figure 7.

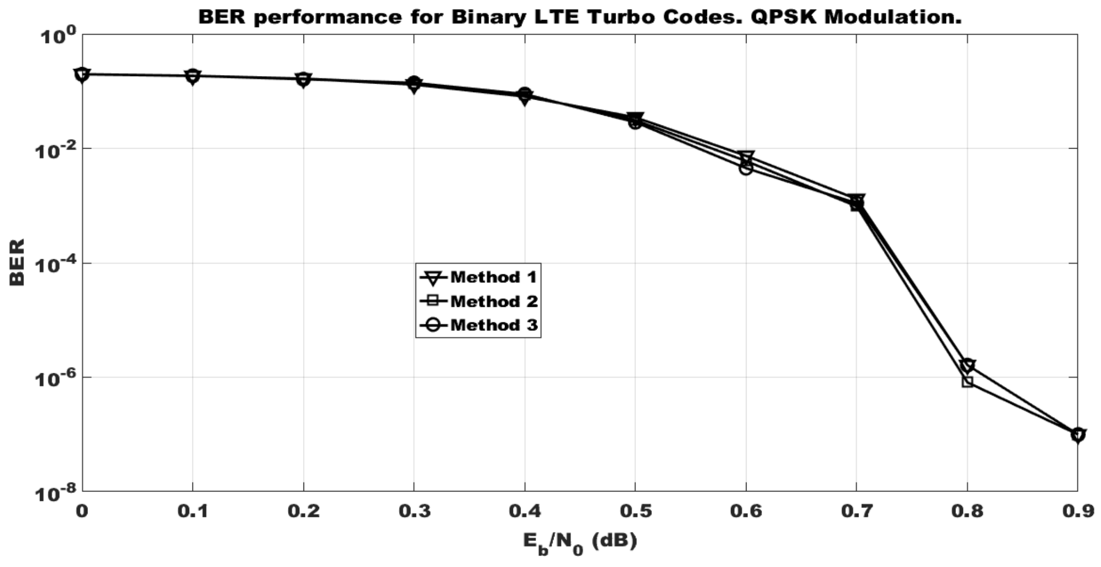

Figure 7. BER performance for Binary LTE Turbo codes with QPSK modulation with SDR stopping

It can be observed from Figure 7 that all 3 decoding methods with SDR stopping have almost similar error performances for the whole E_b/N_0 range despite the differences in total number of different mathematical operations required. The average number of iterations for binary LTE Turbo codes with QPSK modulation and SDR stopping is shown in Figure 8.

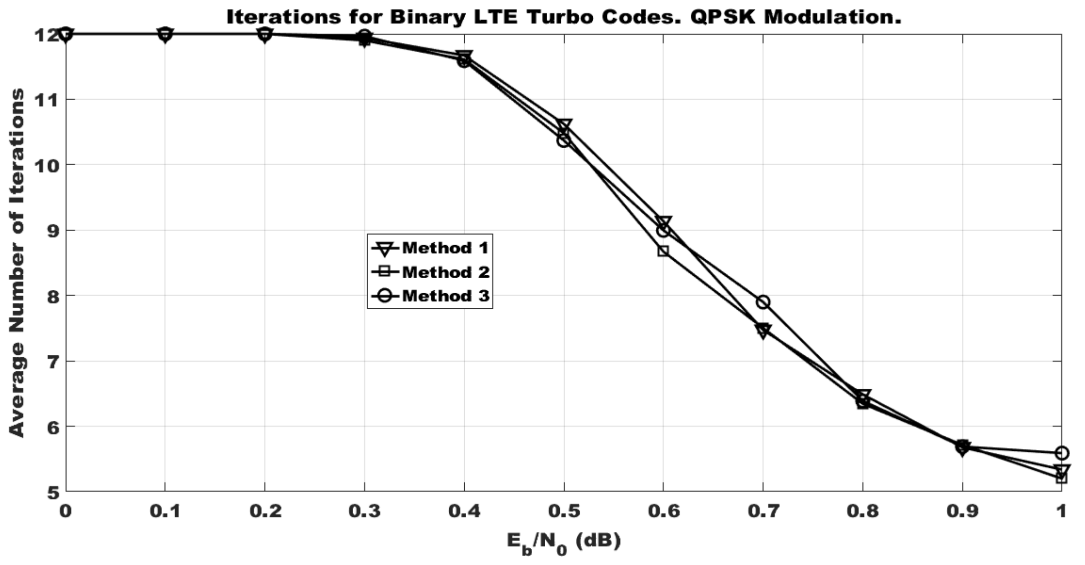

Figure 8. Average iterations for Binary LTE Turbo codes with QPSK modulation and with SDR stopping

It can be observed from Figure 8 that all 3 decoding methods have almost similar average number of decoding iterations throught the whole E_b/N_0 range. The overall number of computations for binary LTE Turbo codes with QPSK modulation and SDR stopping is shown in Figure 9.

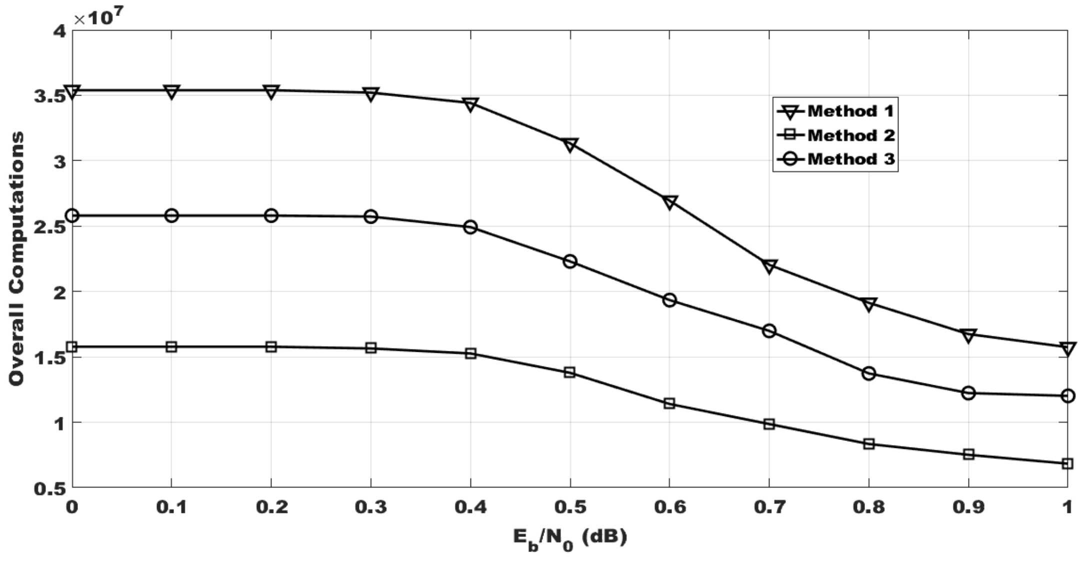

Figure 9. Breakdown of overall number of computation per mathematical operation for Binary LTE Turbo codes with QPSK modulation and SDR stopping

It can be observed from Figure 9 that all 3 decoding methods have different overall number of computations. Method 2 has the least number of overall computations throughout the whole E_b/N_0 range. Method 3 gives an average gain of 3.3510x10^6 computations over Method 1. Method 2 gives an average gain of 1.1417x10^7 computations over Method 1. Method 2 gives an average gain of 8.0665x10^6 computations over Method 3.The breakdown of the overall number of computations per mathematical operation is shown in Figure 10.

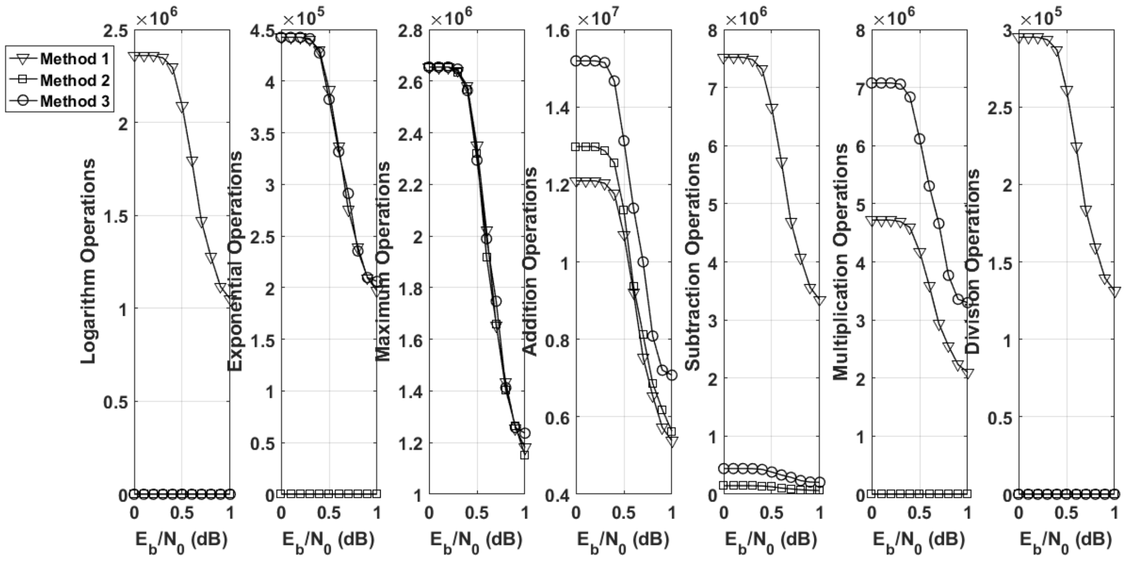

Figure 10. BER performance for Binary LTE Turbo codes with QPSK modulation and SDR stopping

It can be observed from Figure 10 that all 3 decoding methods use approximately the same number of maximum operations throughout the whole E_b/N_0 range. Another observation is that eventhough Method 1 uses fewer addition operations than Methods 2 and 3 over the whole E_b/N_0 range, there are the large numbers of logarithm and division operations which are also used.

The BER performance of binary LTE Turbo codes with 16-QAM and SDR stopping is shown in Figure 11.

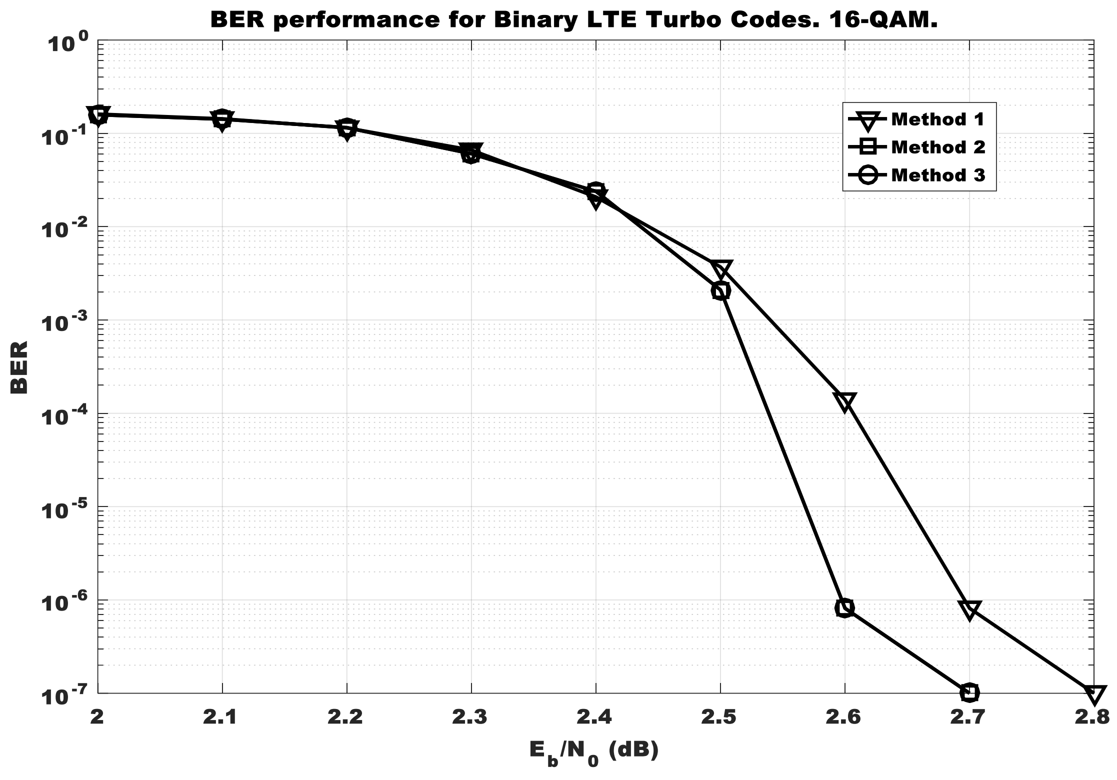

Figure 11. BER performance for Binary LTE Turbo codes with 16-QAM and SDR stopping

It can be observed from Figure 11 that all 3 decoding methods with SDR stopping have almost similar error performances for the range $E_b/N_0 \leq 2.4$ dB despite the differences in total number of different mathematical operations required. For the range $E_b/N_0 > 2.4$ dB, Methods 2 and 3 outperform Method 1 with a gain of 0.1dB on average. The average number of iterations for binary LTE Turbo codes with 16-QAM and SDR stopping is shown in Figure 12.

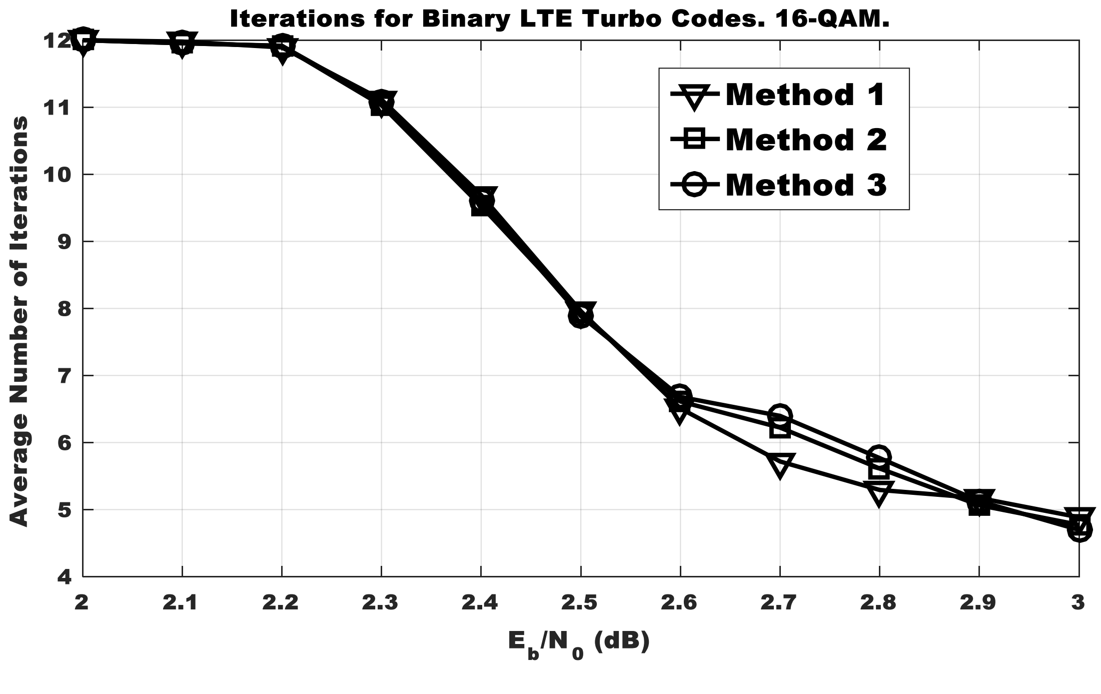

Figure 12. Average iterations for Binary LTE Turbo codes with 16-QAM and SDR stopping

It can be observed from Figure 12 that all 3 decoding methods have almost similar average number of decoding iterations throught the whole E_b/N_0 range. The overall number of computations for binary LTE Turbo codes with 16-QAM and SDR stopping is shown in Figure 13.

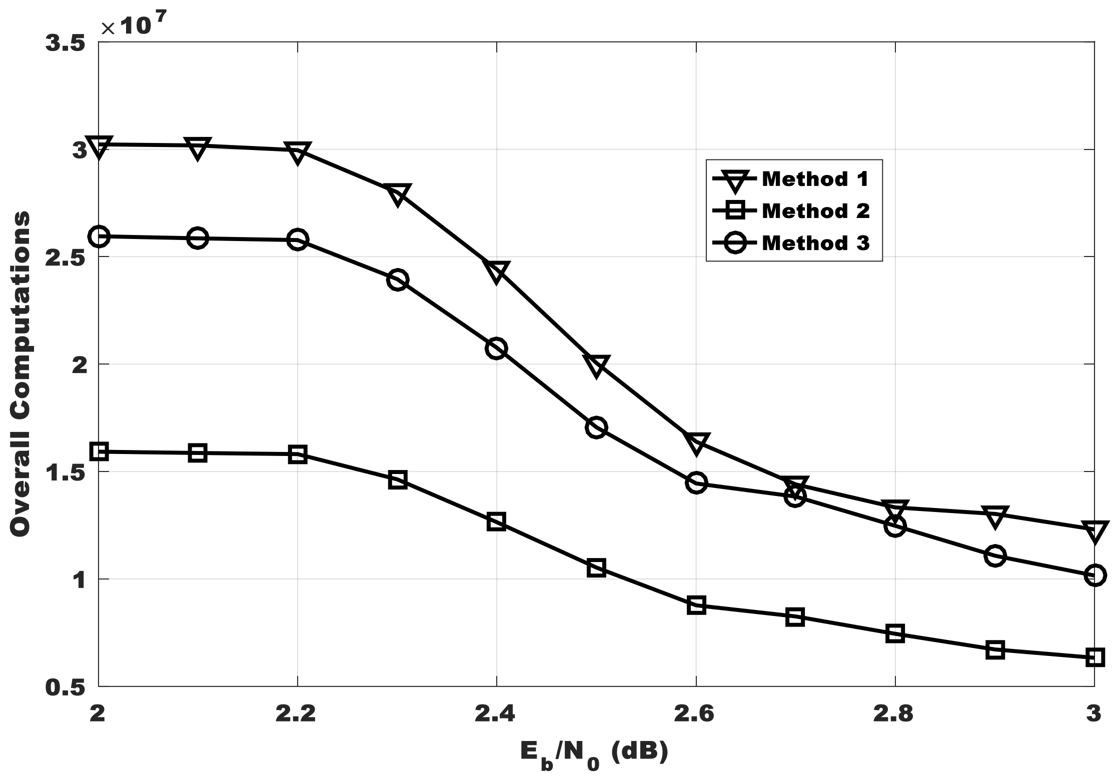

Figure 13. Overall computations for Binary LTE Turbo codes with 16-QAM and SDR stopping

It can be observed from Figure 13 that all 3 decoding methods have different overall number of computations. Method 2 has the least number of overall computations throughout the whole E_b/N_0 range. Method 3 gives an average gain of $2.8108\text{x}10^6$ computations over Method 1. Method 2 gives an average gain of $9.9390\text{x}10^6$ computations over Method 1. Method 2 gives an average gain of $7.1282\text{x}10^6$ computations over Method 3. The breakdown of the overall number of computations per mathematical operation is shown in Figure 14.

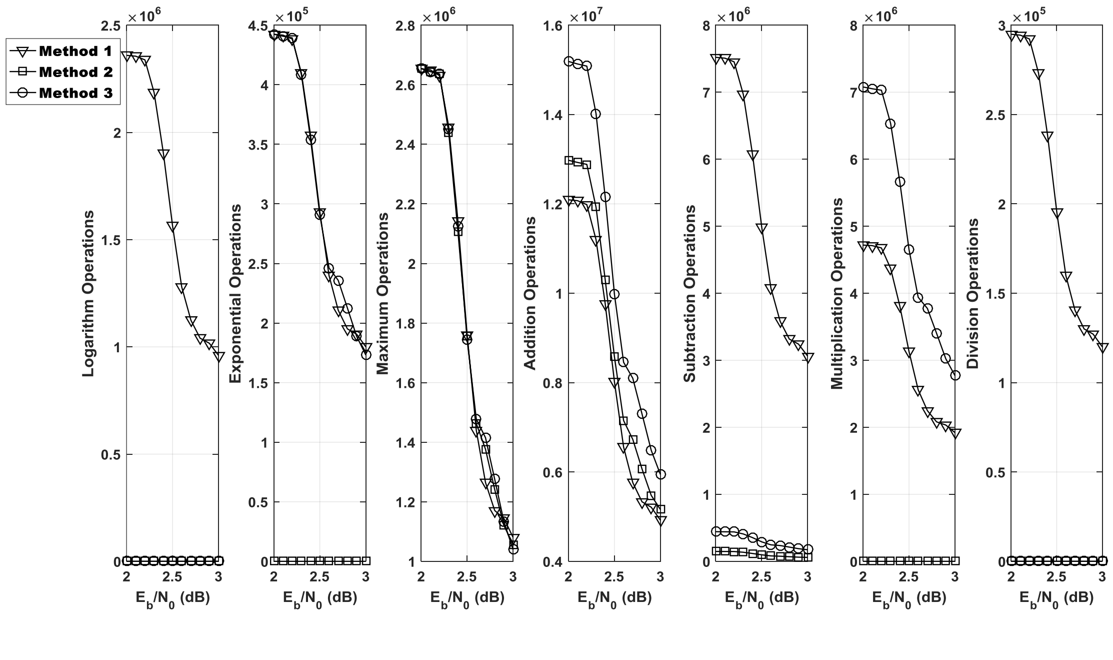

Figure 14. Breakdown of overall number of computation per mathematical operation for Binary LTE Turbo codes with 16-QAM and SDR stopping

It can be observed from Figure 14 that all 3 decoding methods use approximately the same number of maximum operations throughout the whole E_b/N_0 range. Another observation is that eventhough Method 1 uses fewer addition operations than Methods 2 and 3 over the whole E_b/N_0 range, there are the large numbers of logarithm and division operations which are also used.

4. Conclusion

In this paper, a performance analysis of three different iterative decoding techniques for the Max-Log MAP algorithm with SDR-based early stopping has been presented for binary LTE Turbo codes. Three different decoding methods have been shown for Binary LTE Turbo codes with BPSK, Q-PSK and 16-QAM. It can be observed in Figure 3, that with BPSK modulation, the three decoding methods have almost similar performance over the whole E_b/N_0 range. From Figure 7, it can be observed that with QPSK modulation, the three decoding methods have almost similar performance over the whole E_b/N_0 range. Finally, in Figure 11, it can be observed that with 16-QAM, Methods 2 and 3 outperform Method 1 with a gain of 0.1dB on average for the range $E_b/N_0 > 2.4$ dB. The different schemes perform differently in the waterfall and error floor regions for different modulation schemes. From the breackdown of the overall computations, it is observed that eventhough Method 1 uses fewer addition operations than Methods 2 and 3 over the whole E_b/N_0 range, there are the large numbers of logarithm and division operations which it uses. A possible future work which can be envisaged from this work would be to perform more in-depth analysis of the decoding mechanisms for even higher order modulation schemes and different code-rates. Another interesting future work would be to provide an analytical proof of the BER performances of these three Turbo decoding techniques using EXtrinsic Information Transfer (EXIT) charts.

5. Acknowledgements

The technical support of the University of Mauritius and the financial support of the Tertiary Education Commission is duly acknowledged.

6. References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *IEEE Trans.*, May 1993.

[2] Y. Beeharry, T. P. Fowdur and K. M. S. Soyjaudah, "Symbol Level Decoding Algorithms for Duo-Binary Turbo codes," *IIUM Journal (Article in Press)*, 2017.

[3] J. Geldmacher, K. Hueske, M. Kosakowski and J. Gotze, "Application of syndrome based Turbo decoding with adaptive computational complexity in LTE downlink," *Advances in Radio Science*, vol. 2012, no. 10, pp. 159 - 165, 2012.

[4] R. G. Maunder, "A Fully-Parallel Turbo Decoding Algorithm," *IEEE transactions on Communications*, vol. 63, no. 8, pp. 2762 - 2775, 2015.

[5] R. El Chall, F. Nouvel, M. Hélard and M. Liu, "Iterative receivers combining MIMO detection with turbo decoding: performance-complexity trade-offs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 69, pp. 1 - 19, 2015.

[6] F. Li and A. Wu, "On the New Stopping Criteria of Iterative Turbo Decoding by Using Decoding Threshold," *IEEE Trans. on Signal Processing*, vol. 55, no. 11, pp. 5506 - 5516, 2007.

[7] Z. Wang, H. Suzuki and K. K. Parhi, "VLSI implementation issues of turbo decoder design for wireless applications," *Proc. of IEEE Workshop Signal Process. Syst.*, pp. 503 - 512, 1999.

[8] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429 - 445, 1996.

[9] R. Y. Shao, S. Lin and M. P. C. Fossorier, "Two Simple Stopping Criteria for Turbo Decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117 - 1120, 1999.

[10] Y. Wu, B. Woerner and J. Ebel, "A Simple Stopping Criterion for Turbo Decoding," *IEEE Commun. Lett.*, vol. 4, no. 8, pp. 258 - 260, 2000.

[11] L. Henrik, "Turbo Decoder with early stopping criteria," Lund University Libraries, 2016.

[12] T. P. Fowdur, Y. Beeharry and K. M. S. Soyjaudah, "A novel scaling and early stopping mechanism for LTE turbo code based on regression analysis," *Annals of Telecommunications*, pp. 1-20, 2016.

[13] "3GPP: Technical Specifications Rel. 8," 3GPP, 2009.

[14] F. Khan, LTE for 4G Mobile Broadband Air Interface Technologies and Performance, Cambridge University Press, 2009.

[15] B. Vucetic and J. S. Yuan, Turbo Codes: Principles and Applications, Kluwer Academic Publications, 2000, pp. 58-75.

[16] T. P. Fowdur, Y. Beeharry and K. M. S. Soyjaudah, "Performance of Turbo coded 64-QAM with Joint Source Channel Decoding, Adaptive Scaling and Prioritised Constellation Mapping," in *CTRQ, 6th International Conference on Communication Theory, Reliability and Quality of Service*, Venice, Italy, 2013.

[17] T. P. Fowdur, Y. Beeharry and K. M. S. Soyjaudah, "Performance of LTE Turbo Codes with Joint Source Channel Decoding, Adaptive Scalig and Prioritised QAM Constellation Mapping," *International Journal on Advances in Telecommunications*, vol. 6, no. 3 & 4, pp. 143 - 152, 2013.

[18] L. Li, "University of Southampton," 2015. [Online]. Available: http://users.ecs.soton.ac.uk/rm/wp-content/liang_li_nine_month_report.pdf. [Accessed 18 April 2015].

[19] V. Tursenia, "Performance Comparison of Turbo Code in WiMAX Sysetem with Various Detection techniques," *International Journal of Engineering Research*, vol. 2, no. 3, pp. 232 - 236, 2013.

[20] S. A. Abrates, April 2004. [Online]. Available: http://paginas.fe.up.pt/~sam/textos/From%20BCJR%20to%20turbo.pdf. [Accessed 01 May 2015].

Yogesh Beeharry is currently a PhD student in the field of error control coding at the University of Mauritius. He holds a BEng (Hons) Electronics and Communications engineering with First class honours from the University of Mauritius. He was also the recipient of the Mrs. L. F. Lim Fat Engineering Gold medal. His main research interests are: error control coding, Turbo codes and source and channel coding.

Dr. T. P. Fowdur received his BEng (Hons) degree in Electronic and Communication Engineering with first class honours from the University of Mauritius in 2004. He was also the recipient of a Gold medal for having produced the best degree project at the Faculty of Engineering in 2004. In 2005 he obtained a full-time PhD scholarship from the Tertiary Education Commission of Mauritius and was awarded his PhD degree in Electrical and Electronic Engineering in 2010 by the University of Mauritius. He is presently a Senior Lecturer at the Department of Electrical and Electronic Engineering at the University of Mauritius. His research interests include Coding Theory, Multimedia and Wireless Communications, Networking and Security.

Professor K. M. S. Soyjaudah is a Professor of Communications Engineering at the University of Mauritius. Professor Soyjaudah has a number of publications in the fields of Communications engineering, error control coding, information theory among others.