A Method for Lowering Turbo Code Error Flare using Correction Impulses and Repeated Decoding

Youssouf Ould-Cheikh-Mouhamedou[1], Stewart Crozier[2], Ken Gracie[2], Paul Guinand[2] and Peter Kabal[1]

[1] Electrical and Computer Engineering, McGill University, Montreal, Canada, {youssouf, kabal}@tsp.ece.McGill.ca
[2] Communications Research Centre, Ottawa, Canada, {stewart.crozier, ken.gracie, paul.guinand}@crc.ca

Abstract

Turbo codes exhibit an "error floor" or "flare" making it difficult to further improve the error performance without a significant increase in the signal-to-noise ratio (SNR). The error flare is mainly characterized by the distance properties of the code. The conventional way to lower the error flare is to increase the minimum distance, which is mainly determined by the interleaver. Unfortunately, the design of interleavers that yield high minimum distances is not a simple task. In fact, the design of such interleavers for applications requiring very low frame error rates (FERs) can be a real challenge. This paper introduces a new method that significantly lowers the error flare while leaving the interleaver unchanged. It also improves the error performance in the waterfall region. The key element of this method is the insertion of *correction impulses* in the received codeword and the use of *repeated decoding*. The effectiveness of this method is demonstrated by its ability to reduce the error rate by one order of magnitude in the waterfall region and more than three orders of magnitude in the flare region for 8-state single- and double-binary turbo codes of code rate 1/3 and 1/2 that use high-spread random (HSR) interleavers and packets of 1504 bits.

1 Introduction

Turbo codes [1] are one of the most important developments in coding theory in recent years. They offer performance approaching the Shannon capacity limit with reasonable decoding complexity. Their amazing error-correcting capabilities at low to moderate signal-to-noise ratios (SNR) make them suitable for communication systems where significant power savings are required such as deep-space and satellite communications.

A well known type of turbo code uses a parallel concatenation of two recursive systematic convolutional (RSC) encoders along with interleaving. The codewords of turbo codes consist of the information bits (possibly including trellis termination bits), the parity bits resulting from feeding the first encoder with information bits in original order, and the parity bits resulting from feeding the second encoder with the interleaved information bits. The decoding process is performed in an iterative manner by exchanging soft decoding information (called soft extrinsic information) between the constituent decoders. This way, each constituent decoder takes advantage of the extrinsic information produced by the other decoder at the previous step. The core of each constituent decoder is based on a soft-in/soft-out algorithm such as the Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [2] or its simplified logarithmic versions [3].

Typically, turbo code performance is divided into three regions. The first region is associated with very low SNR values. In this region, the error performance is very poor and can only be improved slightly by increasing the number of iterations; the overall error performance in this region is certainly not suitable for most communication systems. The second region, known as the waterfall region (also referred to as the "turbo cliff" region), is associated with low to moderate SNR values. In this region, the slope of the error-rate curves drops rapidly as the SNR increases. To achieve good error-rate performance in the waterfall region, many iterations are required. The third region, known as the error floor or flare region, is associated with moderate to high SNR values. In this region, the error-rate curve suffers from severe flattening making it difficult to further improve the error performance without a significant increase in the SNR. Only a few iterations are required and further iterations usually lead to only very small improvements in the error-rate performance.

It is desirable to have turbo codes that have very steep slopes in the waterfall region, which begins at very low SNRs, and very low error flares. Unfortunately, the design of such turbo codes is challenging. This is because the known design criteria for the waterfall and error flare regions are based on different approaches. In the waterfall region, Extrinsic Information Transfer (EXIT) chart analysis [4], [5] can be used to help predict the SNR at which the waterfall region begins as well as to estimate the bit error rate (BER) after an arbitrary number of iterations. In the error flare region, where the error performance is mainly determined by the distance properties of the code [6], the natural way to lower the error flare is to increase the minimum distance ($d_{\min}$). Increasing $d_{\min}$ by increasing the memory length of the RSC constituent encoders is undesirable due to corresponding increases in complexity and memory

requirements. A better approach is to use moderate memory RSC constituent encoders (i.e., memory 2, 3, or 4) with interleavers that yield high $d_{\min}$, such as dithered relative prime (DRP) interleavers [7].

This paper is concerned with the problem of how to improve the error performance of turbo codes, especially in the error flare region, while leaving the RSC encoders and the interleaver unchanged.

The remainder of this paper is organized as follows. The second section provides a review of past work aimed at lowering the error flare. The third section provides the details of the proposed new method for improving the error performance of turbo codes. The fourth section presents example simulation results. The fifth section summarizes the paper and presents some concluding remarks.

2 Past Work

Motivated by the observation that at high SNR values the number of information bit errors per frame error is usually small, the serial concatenation of a high-rate algebraic outer code and a turbo code has been proposed to improve the bit error-rate performance in the error flare region [8], [9], [10], [11], [12]. The main purpose of the algebraic outer code is to correct the information bit errors that remain after turbo decoding, but it can also be used to detect whether the turbo decoding was successful or not. Note that the use of an outer code results in a small reduction in code rate as well as a small increase in the overall decoding complexity. This is because the complexity of the outer decoder is typically low compared to the complexity of the turbo decoder.

Costello *et al.* proposed the use of a Reed-Solomon (RS) outer code to improve the flare performance [8]. In the flare region, the information bit errors appear in bursts, but their densities in the bursts are low. Since RS codes are designed to correct error bursts with high bit error densities, the use of a RS code as an outer code may not be the best choice. Andersen proposed the use of a Bose, Chaudhuri, and Hocquenghem (BCH) outer code that also protects the entire frame of information bits [9]. The simulation results reported in [9] show that the use of a BCH outer code can result in a significant improvement in BER flare.

Takeshita *et al.* showed that Andersen's method can not only lower the BER flare, but it can also significantly improve the frame error rate (FER) in the flare region [10]. Takeshita also used the BCH code as an early stopping tool for turbo iterative decoding. That is, after each iteration, the BCH outer decoder can be used to check if turbo decoding was successful. If it was, the turbo decoder is stopped; otherwise, the turbo decoder continues until the maximum number of allowed iterations is reached.

Narayanan *et al.* also proposed the use of a BCH outer code [11]. In contrast to Andersen's method

where the BCH outer code protects the entire frame of information bits, Narayanan's method employs a BCH outer code that protects only a few positions in the frame of information bits. These positions are the ones associated with the lowest distance codewords. Narayanan also proposed the use of a cyclic redundancy check (CRC) code between the BCH outer code and the turbo inner code. The CRC code serves only as an early stopping tool for iterative decoding. That is, after each full-iteration, the CRC is used to check for errors. If no errors are detected by the CRC code, the iterative turbo decoding is stopped. Note that the block length of the BCH outer code increases, and so does the complexity, as the number of positions to be protected increases. Thus, the complexity and overhead of this method is usually lower for random interleavers than for well-designed high-distance interleavers such as the highly structured DRP interleavers [7] and almost regular permutation (ARP) interleavers [13]. This is because the lowest distances of random interleavers usually have low multiplicities caused by low-weight input sequences, whereas the lowest distances of well-designed interleavers typically have higher multiplicities and higher input weights. This is not to say that the method works best for random interleavers. On the contrary, the improvement with random interleavers is only significant because they tend to have very poor distance properties. The better the interleaver design, the less effective this method becomes, and the more it must approach that of Andersen's method where the entire frame is protected by the BCH outer code.

Öberg *et al.* proposed a method for lowering the BER flare based on distance spectrum analysis [12]. The first step of this method is the identification of the bit positions in the information frame associated with the lowest distances. Then, a modified turbo-code encoder inserts dummy bits in these positions. Consequently, the positions and values of these dummy bits are known to the turbo decoder. The decoding can be done easily by associating the received values corresponding to the dummy bits with strong indications that reflect their original values (for example, strong indications that all dummy bits were zeros, assuming that the modified encoder set all dummy bits to zero). After decoding is complete, these dummy bits in the decoded frame are discarded. Note that Öberg's method removes the contribution of the lowest distances to the error performance by inserting dummy bits at information bit positions associated with the lowest distances. Thus, the improvement in the error performance at high SNR values is expected to be the same as the improvement in the union bound asymptote caused by ignoring these lowest distances. Note that the loss in code rate caused by the insertion of dummy bits increases as the multiplicities of the lowest distance codewords increase. Thus, the loss in code rate tends to be smaller for random interleavers than for well-designed interleavers.

In fact, this method is not suitable for most well-designed interleavers because the loss in code rate is not tolerable.

Another reason why Narayanan's and Öberg's methods provide significant improvements for low-distance random interleavers is because the performance of iterative decoding at high SNR values is close to that of maximum-likelihood (ML) decoding. That is, the distance between the estimated codeword and the transmitted codeword is usually close to $d_{\min}$ for most packet errors. Consequently, protecting the positions in the information frame associated with the lowest distances improves the error performance in the error flare region. Unfortunately, the higher the value of $d_{\min}$ the less likely it is that the iterative decoding performs close to ML decoding. That is, the distance between the estimated codeword and the transmitted codeword is close to $d_{\min}$ only for a few packet errors. Consequently, less improvement in error performance is to be expected with high $d_{\min}$ interleavers compared to low $d_{\min}$ interleavers. Another drawback of these methods is the fact that any changes in the constituent encoders, interleaver, or interleaver length requires repeating the tedious off-line task of determining the bit positions in the information frame that need to be protected.

3 The Correction Impulse Method (CIM) - A New Approach

This section introduces a new method for improving the error performance of turbo codes, especially in the error flare region, while leaving the RSC encoders and the interleaver unchanged. This method requires the use of an error detection method such as a CRC. Thus, only a small reduction in the code rate is required. For example, the use of a 16-bit CRC in an MPEG-sized packet (i.e., 1504 information bits) reduces the code rate by a factor of 0.9894. The corresponding penalty in SNR is 0.046 dB. Generally, the longer the frame, the smaller the impact on code rate and SNR.

This method involves the following steps. First, decode the received codeword in the normal manner. Then, in the case of decoding failure, determine a few positions in the decoded information frame that are most likely to have bit errors. One-by-one, each position that is determined to possibly have a bit error is forced to have a value that corresponds to the opposite of the decision from the first decoding, and the received codeword is decoded again. The repeated decoding continues until a successful decoding is declared or all candidate bit positions have been tested. The strength of this method lies in allowing the iterative decoding to exploit the new information (i.e., the positions of the least reliable bits) extracted from the first decoding process. By exploiting this additional information, along with error detection, the reliability of the soft output values is improved and therefore the error performance is improved.

Let u and y be the information frame and the received codeword, respectively. For simplicity, assume the use of antipodal signaling that maps the bit b to $(-1)^b$. Let E be a real number greater than $d_{\min}$. To force the turbo decoder to decide for the bit b_i at position i in u, it is sufficient to insert the impulse $I_i = (-1)^{b_i} E$ at position i in y. Let Ω be the set of positions in the decoded frame, $\hat{u}$, that are most likely to have bit errors. Let L be the number of elements in Ω (the value of L is usually small and cannot exceed the length of the information frame). The positions of interest are recorded in Ω according to their reliabilities (i.e., position $\Omega(i)$ is more likely to have an error than position $\Omega(i+1)$). A short description of this method follows, where j is the position of the bit forced to change.

$\|$ set the noise variance σ^2,
  if it is required for iterative decoding;
$\|$ choose E;
$\|$ iterative decoding of $y \Rightarrow \hat{u}$;
$\|$ if(error detected)
  $\mid$ determine Ω;
  $\mid$ set $i = 0$;
  $\mid$ while((error detected) and $(i < L)$)
    - set $j = \Omega(i)$;
    - set $z = y(j)$;
    - set $y(j) = -(-1)^{\hat{u}(j)} E$;
    - iterative decoding of y;
    - set $y(j) = z$;
    - set $i = i + 1$;
    end while
  end if

Note that this method does not require the determination of all bit errors in the decoded information frame. It is difficult, if not impossible, to determine all of them. In fact, if there was a way to accurately determine all bit errors, then repeated decoding would not be necessary. This method requires only the knowledge of a few possible error positions in the decoded frame and these positions are tested successively according to their reliabilities. This method is referred to as the *correction impulse method* (CIM). In the worst case, the complexity of this method increases by a factor of $1+L$ for each frame that is in error. On average, the complexity increases by at most a factor of $(1+L \cdot \text{FER})$, where FER is the frame error rate before applying the CIM. The actual average complexity is lower still because the number of extra decodings rarely reaches L when L is much greater than 1. The extra processing resulting from repeated decoding is guaranteed to be low on average if the original FER is low compared to $1/L$.

The remainder of this section describes how to determine the reliability of each information symbol in the decoded frame based on the soft log-likelihood ratios (LLRs) produced by the turbo decoder. Each symbol consists of 1 bit for single-binary turbo codes and 2 bits for double-binary turbo codes.

For single-binary turbo codes, each bit index in the information frame has two LLRs corresponding to the bit values 0 and 1, where it is understood that one of the bit values (e.g., 0) always has an LLR value of zero. An LLR with a small absolute value indicates a low level of confidence in the decision made by the turbo decoder, whereas a large absolute value indicates a high level of confidence. Thus, bits whose LLRs have small magnitudes are more likely to be in error than bits whose LLRs have large magnitudes.

For double-binary turbo codes, each symbol index in the information frame has four LLRs corresponding to the symbols 00, 01, 10 and 11, where it is understood that one of the symbols (e.g., 00) always has an LLR value of zero. One method of determining the symbol reliabilities is as follows. For each symbol index, compute the difference between the largest and second largest LLRs. The larger the difference, the more likely that the turbo decoder made a correct symbol decision.

For a given symbol index assumed to be in error, the next step is to guess the value of the transmitted symbol. Unlike single-binary turbo codes where the guess of the transmitted bit is obvious (i.e., if a '1' is assumed to be in error then the transmitted bit must be a '0' and vice versa), double-binary turbo codes have three alternate symbol candidates for the transmitted symbol. All three are considered.

4   Simulation Results

The CIM is applied to single-binary turbo codes that use the Universal Mobile Telecommunications System (UMTS) 8-state polynomial generators [14] as well as double-binary turbo codes that use the Digital Video Broadcast with Return Channel via Satellite (DVB-RCS) 8-state polynomial generators [15]. Enhanced max-log *a posteriori* probability (EML-APP) decoding with maximum of 16 full iterations and early stopping with $B=2$ was used, where B is the number of consecutive sets of hard decisions that must agree before stopping [16], [17]. Quadrature phase shift keying (QPSK) modulation and transmission over an additive white Gaussian noise (AWGN) channel were assumed. MPEG packets of length 1504 bits were simulated. Various high-spread random (HSR) interleavers [18] were tested for code rates of 1/3 and 1/2. The code rate of 1/2 for double-binary turbo codes was obtained using DVB-RCS standard puncturing.

At high SNR values, the FER of turbo codes can be approximated using the union bound, given by

$$\text{FER} \leq \frac{1}{2} \sum_{d \geq d_{\min}} A_{\text{d}} \, \text{erfc}\left(\sqrt{dR_c \frac{E_{\text{b}}}{N_0}}\right). \quad (1)$$

Here, the multiplicity A_{d} is the number of codewords with Hamming weight d, R_c is the code rate, $\text{erfc}(x)$ is the complementary error function, E_{b} is the energy per information bit and N_0 is the one-sided noise power spectral density. The truncated union bound (TUB) curves shown in Figs. 1 through 4 used only the first three terms of the original distance spectrum, determined for the single-binary turbo codes using Garello's distance measurement method [19] and for the double-binary turbo codes using a modified version of Garello's method [20]. The TUB results correspond to the original turbo code, without the CRC for error detection.

Approximations to the binary-input sphere-packing bound (SPB) are also shown in the figures for each code rate and block size. Approximations to the continuous-input SPB were first generated based on an approach outlined in [21]. The bounds for binary signalling were obtained by shifting the continuous-input bounds to the right by 0.06 dB for $R_c=1/3$ and 0.19 dB for $R_c=1/2$. These offsets are the differences between the capacity limits of the continuous- and binary-input cases for an infinite block length and are thought to be reasonably accurate for $R_c \leq 1/2$ [21]. A "genie" CRC was used for error detection, i.e., the decisions from the decoder were compared to the transmitted codeword. A 16-bit CRC should provide essentially the same performance as the genie for improvements of up to four orders of magnitude. A 32-bit CRC would be more than sufficient for most applications.

CIM(L) denotes the error performance obtained by testing the least reliable L symbols in the decoded frame. For an MPEG-sized packet, the maximum values that are allowed for L are 1504 and 752 for single- and double-binary turbo codes, respectively. The testing of these symbols is done in a consecutive manner starting with the least reliable. Note that CIM(L) requires a maximum of L and $3L$ extra turbo decoding operations per packet error for single- and double-binary turbo codes, respectively, because three alternate symbol candidates are considered for double-binary turbo codes. Consequently, the peak complexity for double-binary turbo codes is higher than that of single-binary turbo codes.

4.1 Single-Binary Results

Fig. 1 shows FER performance with the CIM for an 8-state, single-binary turbo code at a nominal code rate of 1/3. Dual trellis termination [22] is used with an MPEG block length (1498 information bits + 6 termination bits = 1504 total bits), yielding an actual code rate of $R_c=0.3320$. The code has a $d_{\min}$ of 28, a multiplicity of 5, and an information bit multiplicity of 20. In the waterfall region, the CIM yields improvements of up to 0.1 dB at FER=10^{-3}, and the slope of the curves more closely matches that of the binary-input SPB. The improvements in the flare are more dramatic: roughly one, two, and three orders of magnitude for CIM(1), CIM(4), and CIM(16), respectively, at $E_b/N_0=1.50$ dB. Testing 256 or more indices corrected all of the observed error events at 1.25 and 1.50 dB.

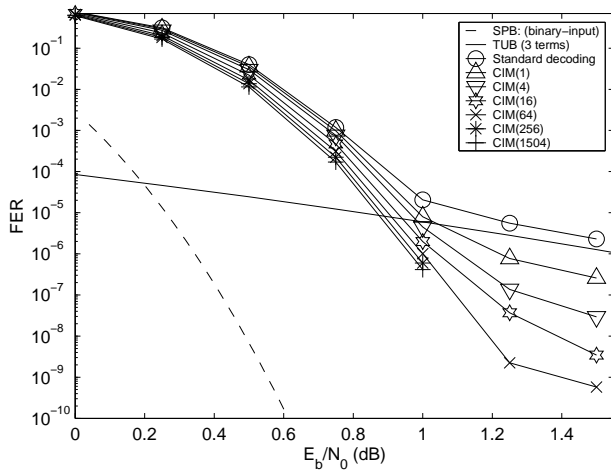Fig. 2 shows FER performance with the CIM when

Fig. 1. FER performance for an 8-state, single-binary turbo code for MPEG packets and R_c=0.3320 using EML-APP with an extrinsic scale factor of 0.75. Before applying the CIM, 4×10^3 packet errors were counted at 1.5 dB and 5×10^3 packet errors at all other SNR values.
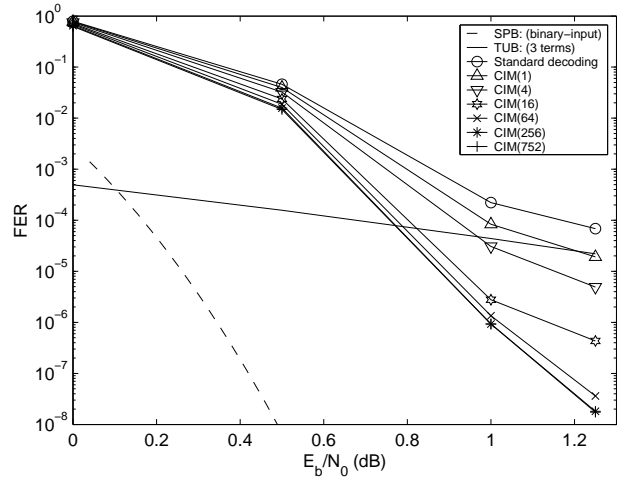

Fig. 3. FER performance for an 8-state, double-binary turbo code for MPEG packets and R_c=1/3 using EML-APP with an extrinsic scale factor of 0.75. 10^3, 10^3, 10^4 and 10^5 packet errors were counted at 0 dB, 0.5 dB, 1.0 dB and 1.25 dB, respectively, before applying the CIM.
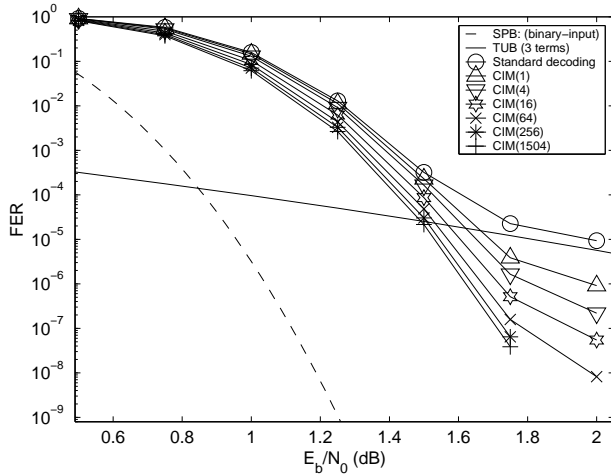

Fig. 2. FER performance for an 8-state, single-binary turbo code for MPEG packets and R_c=0.4980 using EML-APP with an extrinsic scale factor of 0.8. Before applying the CIM, 3.4×10^3 packet errors were counted at 2.00 dB, 7×10^3 were counted at 1.75 dB, and 10^4 were counted at all other SNR values.
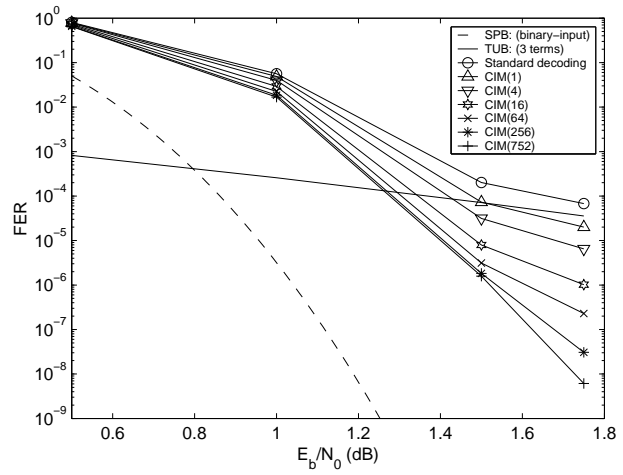

Fig. 4. FER performance for an 8-state, double-binary turbo code for MPEG packets and R_c=1/2 using EML-APP with an extrinsic scale factor of 0.8. 10^3, 10^3, 10^4 and 10^5 packet errors were counted at 0.5 dB, 1.0 dB, 1.5 dB and 1.75 dB, respectively, before applying the CIM.

the code of Fig. 1 is punctured to R_c=0.4980 (i.e., nominally rate 1/2). This code has a $d_{\min}$ of 16, a multiplicity of 10, and an information bit multiplicity of 30. In the waterfall region, the CIM yields improvements of roughly 0.1 dB at FER=10^{-3}. Flare performance is again improved substantially, by approximately one, two, and three orders of magnitude for CIM(1), CIM(16), and CIM(64), respectively, at 2.00 dB. Testing 256 or more indices corrected all of the observed error events at 2.00 dB.

4.2 Double-Binary Results

Fig. 3 shows performance results for an 8-state, double-binary turbo code for a code rate of R_c=1/3 and an encoder using circular encoding (i.e., tail-biting) to terminate the trellises. This code has a $d_{\min}$ of 24, a

multiplicity of 4, and an information bit multiplicity of 22. As before, a significant improvement in error performance is achieved by increasing the maximum number of tested symbols L. In the waterfall region, testing all possible symbols (i.e., CIM(752)) resulted in SNR savings of 0.2 dB at FER=10^{-3}. In the flare region, applying CIM(4), CIM(16), and CIM(64) resulted in FER improvements of more than one, two, and three orders of magnitude, respectively, at 1.25 dB.

Fig. 4 shows performance results for an 8-state, double-binary turbo code for R_c=1/2 and an encoder using circular encoding. This code has a $d_{\min}$ of 14, multiplicity of 5, and an information bit multiplicity of 28. In the waterfall region, applying CIM(752) resulted in SNR savings of 0.2 dB at FER=10^{-3}. In the flare region, the use of CIM(16) resulted in FER improvements

of about 1.5 and 2 orders of magnitude at 1.5 dB and 1.75 dB, respectively. Testing all possible symbols (i.e., CIM(752)) reduced the number of packet errors from 10^5 to only 7 at 1.75 dB (i.e., a FER improvement of more than four orders of magnitude). As in the single-binary case, using the CIM noticeably reduces the gap to the binary-input SPB and yields performance roughly parallel to the bound in the waterfall region.

Note that for both the single- and double-binary cases, these impressive improvements in error-rate performance appear as soon as performance is no longer dominated by convergence in the waterfall region. That is, the gains become substantial as soon as the flare region is encountered. The flare region is also the region where the average increase in complexity is very small.

It is expected that the correction capability of the CIM would be improved further if two or more symbols are simultaneously forced to take certain values. This is particularly desirable when significant power savings are required, such as is the case in deep-space and satellite communications. Note that the average complexity is still low if L is limited. For example, forcing up to two symbols to take certain values requires at most $L + L(L-1)/2$ and $3L + 9L(L-1)/2$ extra turbo decoding operations per packet error for single- and double-binary turbo codes, respectively. To keep the peak complexity reasonable, the multiple symbols that should be forced to certain values should be chosen from a small set (e.g., $L < 30$).

5   Conclusion

A new method was introduced that significantly improves error-rate performance in both the waterfall and error flare region given fixed RSC encoders and interleaver. The method inserts correction impulses into the received codeword and decodes multiple times in an effort to correct residual errors. Allowing the iterative decoding process to exploit the new information extracted from the first decoding, along with error detection, improves the reliability of the decoder's soft outputs and therefore the error-rate performance is improved. The effectiveness of this method is demonstrated by its ability to improve the error performance curve by several orders of magnitude in the error flare region for rate 1/3 and rate 1/2 8-state single- and double-binary turbo codes using MPEG-sized packets. This significant improvement is achieved with a very small increase in the average decoding complexity in the flare region.

The method is expected to work very well with any coding scheme that employs soft iterative decoding. Future work includes applying this method to low-density parity-check (LDPC) codes [23].

6   Acknowledgments

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. *Near Shannon limit error-correcting coding and decoding: Turbo-codes*. Proc. IEEE Int. Conf. Commun. (ICC'93). pp. 1064–1070, May 1993.

[2] L. Bahl, J. Cocke, F. Jelinek and J. Raviv. *Optimal decoding of linear codes for minimizing symbol error rate*. IEEE Trans. Inform. Theory. Vol. 7, pp. 284–287, Mar. 1974.

[3] P. Robertson, E. Villebrun, and P. Hoeher. *A Comparison of Optimal and Suboptimal MAP Decoding Algorithms Operating in the Log Domain*. Proc. IEEE Int. Conf. Commun. (ICC'95)", pp. 1009–1013, Jun. 1995, (Seattle, WA).

[4] S. ten Brink. *Convergence of iterative decoding.* Electron. Lett. Vol. 35, pp. 806–808, May 1999.

[5] S. ten Brink. *Convergence behavior of iteratively decoded parallel concatenated codes.* IEEE Trans. Commun. Vol. 49, pp. 1727–1737, Oct. 2001.

[6] L. C. Perez, J. Seghers, and D. J. Costello Jr. *A distance spectrum interpretation of turbo codes.* IEEE Trans. Inform. Theory. Vol. 42, pp. 1698–1709, Nov. 1996.

[7] S. Crozier and P. Guinand. *Distance Upper Bounds and True Minimum Distance Results for Turbo-Codes Designed with DRP Interleavers.* Proc. 3rd Int. Symp. turbo codes. pp. 169–172, Sep. 2003, (Brest, France).

[8] D. J. Costello Jr. and G. Meyerhans. *Concatenated turbo codes.* IEEE Int. Symp. Inform. Theory and Its Applications (ISITA96). pp. 571–574, Sept. 1996, (Victoria, BC, Canada).

[9] J. D. Andersen. *Turbo codes extended with outer bch code.* Electron. Lett. Vol. 32, pp. 2059–2060, Oct. 1996.

[10] O. Y. Takeshita, O. M. Collins, P. C Massey, and D. J. Costello Jr. *On the frame-error rate of concatenated turbo codes.* IEEE Trans. Commun. Vol. 49, pp. 602–608, Apr. 2001

[11] K. R. Narayanan and G. L. Stüber. *Selective serial concatenation of turbo codes.* IEEE Commun. Lett. Vol. 1, pp. 136–139, Sept. 1997.

[12] M. Öberg and P. H. Siegel. *Application of distance spectrum analysis to turbo code performance improvement.* Proc. 35th Annu. Allerton Conf. Commun., Control, and Computing. pp. 701–710, Sept. 1997.

[13] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel. *Designing good permutations for turbo codes: towards a single model.* Proc. IEEE Int. Conf. Commun. (ICC'04). pp. 341–345, Jun. 2004.

[14] 3rd *Generation Partnership Project (3GPP) Technical Specification Group: Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (FDD).* TS 25.212 V3.4.0. Sep. 2000.

[15] European Telecommunications Standards Institute", *Interaction channel for satellite distribution systems.* ETSI EN 301 790, V1.3.1. Mar. 2003.

[16] Y. Ould-Cheikh-Mouhamedou, P. Guinand, and P. Kabal. *Enhanced Max-Log-APP and enhanced Log-APP decoding for DVB-RCS.* Proc. 3rd Int. Symp. turbo codes. pp. 259–262, Sep. 2003, (Brest, France).

[17] K. Gracie, S. Crozier, and P. Guinand. *Performance of an MLSE-based Early Stopping Technique for Turbo Codes.* 60th IEEE Vehicular Technology Conference 2004 - Fall (VTC 2004 - Fall), Sep. 2004 (Los Angeles, California, USA).

[18] S. Crozier. *New high-spread high-distance interleavers for turbo codes.* Proc. 20th Biennial Symp. Commun., pp. 3–7, May. 2000., (Queen's University, Kingston, Canada).

[19] R. Garello, P. Pierleoni, and S. Benedetto. *Computing the Free Distance of Turbo Codes and Serially Concatenated Codes with Interleavers: Algorithms and Applications. 2000.* IEEE Journal on Selected Areas Commun., pp. 800–812, May. 2001.

[20] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal. *Distance Measurement Method For Double Binary Turbo Codes and A New Interleaver Design For DVB-RCS.* Proc. IEEE Globecom. Nov./Dec. 2004, (Dallas, Texas).

[21] S. Dolinar, D. Divsalar, and F. Pollara. *Code Performance as a Function of Block Size.* JPL, The TMO Progress Report: Technical Report 42-133, May. 1998.

[22] P. Guinand and J. Lodge. *Trellis Termination for Turbo Encoders.* Proc. 17th Biennial Symp. Commun., pp. 389–392, May-Jun. 1994., (Queen's University, Kingston, Canada).

[23] R. G. Gallager. *Low-density parity-check codes.* IEEE Trans. Inform. Theory. Vol. 8, pp. 21–28, Jan. 1962.