A Turbo Codes Delay Reduction Based on Probability Density Evolution

Zafir Popovski[1], Tatjana Ulcar-Stavrova[2]

Abstract – **The turbo codes [1] are decoded in a iterative decoding scheme [2] performing a predefined number of iterations before the final decision comes up. Since the method is time consuming, stopping rules can be apply to prematurely quit the process, for the decoder has already done its job. This technique requires a reasonable trade-off which should results in a average decoding speed increase while not sacrificing the decoder performance.**

Keywords – **Stopping rules, number of iterations**

I. Introduction

The turbo codes (TC) are a class of FEC (forward error control) codes, known as parallel concatenation of two or more recursive systematic convolutional codes (RSCC), produced by a turbo encoder composed of two or more constituent RSC encoders (CEs) with input to each but one constituent encoder permuted by an interleaver of length N.

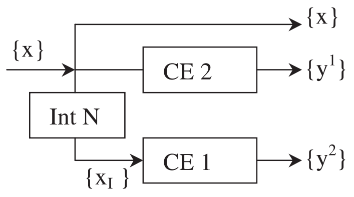

Fig. 1. Turbo encoder structure

Such a composition allows for replacement of the optimal but rather complex maximum likelihood decoding algorithm by two or more relatively simple constituent decoders to decode corresponding constituent codes one by one.

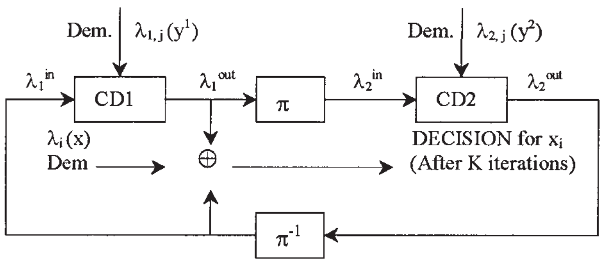

Fig. 2. Iterative TD with two MAP CDs

[1]Zafir Popovski is with the Faculty of Electrical Engineering, "St.Cyril and Methodius" University – Skopje, E-mail: zpopovski@yahoo.com

[2]Tatjana Ulcar-Stavrova is with the Faculty of Electrical Engineering, "St.Cyril and Methodius" University – Skopje, E-mail: tanjaus@etf.ukim.edu.mk

A simple SISO (Soft-In Soft-Out) maximum a posteriori (MAP) decoder [3] which minimise the probability of bit error appears to be the best solution for component decoders (CDs).

We consider only TCs with two CEs for which the coding and decoding processes are shown on Figs. 1 and 2, respectively.

The MAP algorithm provide as an output a real number which is a measure of the probability of error in decoding a particular bit. Since both CDs decode the same information bit xi, coded twice but in different order, it becomes possible this extra information, named extrinsic information, li, to be passed as input to the second CD allowing it to improve its own output extrinsic which will be passed to the first CD in the next iteration. The process is than iterated until reaching a satisfactory degree of confidence regarding the received noisy examples contained in length N sequence. This "satisfactory degree of confidence" does not depend on the pre-set number of iterations and can be reached in any of them. So, the idea is to follow the extrinsic's density evolution until it reaches a "confidential threshold" and then cease the iterative decoding process.

In the follow-up we first briefly present some published rules and a "magic genie rule" [4] used as a benchmark for other rules to be compared with. Later we adopt a model for probability density evolution of the extrinsic information suitable to impose a confidential threshold and then explore few threshold values and compare them with known results through a "C^{++}" turbo code simulation programme. The criteria for comparison are decoding speed, BER and FER, and computational complexity. At the end a conclusion is given.

II. Stopping Rules

Basically, all the stopping rules, by checking the stopping condition at the end of each or each half iteration, attempt to determine the moment when a frame can be reliably decoded. If the condition is true the iterative process is terminated. Otherwise, it continues with the next iteration and, if the stopping condition is never met, stops after a pre-set maximum number of iterations to prevent an endless loop.

Mainly, the stopping rules that cause the decoder to use a variable number of iterations are divided in two main types, hard and soft decision rules, both of which provide for a more or less easy computation based on the data available during the decoding process.

Hard decision (HD) rules evaluate the tentative decoded bits (hard bit decisions) at the end of each or half iteration and the decision is taken after detecting a consistency in two, three or more successive full or half iterations.

Soft decision (SD) rules are based on comparing a metric on bit reliabilities (soft bit decisions) with a threshold. Suitable metrics used in [4] are the average and minimum reliabilities of the information bits.At each iteration, the turbo coder computes the relevant metric and compares it with preset threshold value.

Our rule belongs to the SD type rules but the turbo decoder has to perform less computations than in any previously published SD rule because it uses the actual density evolution of exchanged extrinsics that contribute to the information bit reliabilities. Since the extrinsics density evolves from iteration to iteration, the turbo decoder's CPU only needs to compare this evolution with a pre-set threshold. The only problem we experienced is to conduct a comparison after the first iteration, for some skewness in the density evolution appears on the first constituent decoder's noisy output which might result in falsely decoded bits at the end of the first iteration. This mainly due to the lack of a extrinsic information for the first constituent decoder at the first iteration.

The "magic genie" rule is an unrealisable (in reality) rule useful for establishing an unbeatable performance benchmark against which the other rules are measured. For this rule, the correct decoded codeword is immediately recognised due to the foreknowledge of the transmitted bit sequence and it stops the iterative process in exactly the minimum number of iterations required to produce the correct codeword.

III. Density Evolution Model

The iterative decoder can be considered as a non-linear dynamical feedback system as shown on Fig. 2.

Extrinsic information messages λ_i are passed from one to the other constituent decoder. The message λ_i measures the log-likelihood ratio for the i-th bit based on input massages λ_j from all other bits but the i-th. So, if we assume that the all-zero codeword is transmitted (with BPSK modulation it corresponds to transmission of "+1"s on the channel) then a positive value of the extrinsic information, $\lambda_i > 0$, for each i, will represent a favourable evidence toward determining the true value of the i-th bit.

When the interleaver is large and random, the extrinsics $lambda_i$ are independent and identically distributed with probability density function $f(\lambda)$. As shown in [5], this pdf is consistent ($\lambda = \log[f(\lambda)/f(-\lambda)]$), its mean, $\mu = E(\lambda)$, is discrimination between the two densities $f(\lambda)$ and $f(-\lambda)$, and the error probability, $e = \Pr\{\lambda < 0\}$, can be evaluated as $e = \mathrm{E}\{1/(1 + e^{|\lambda|})\}$. Computed histograms of the λ_{in} and λ_{out} extrinsics at the input and output of a SISO MAP decoding module for a 4 states, rate 1/3, $[1,5/7]_8$ turbo code are plotted on Fig. 3 for a number of iterations. As it can be seen, the empirical probability densities $f(\lambda_{\text{in}})$ and $f(\lambda_{\text{out}})$ evolve with successive decoder iterations from narrow densities concentrated nearby $\lambda = 0$, to broader Gaussian-shaped densities with increasing means as the iterations continue. Ignoring some irregularities at the beginning of the process, this probability density function can be approximated by a Gaussian density in which case its statistics depend on two parameters: its mean $\mu = \mathrm{E}(\lambda)$ and its variance $\sigma^2 = \mathrm{Var}(\lambda)$.

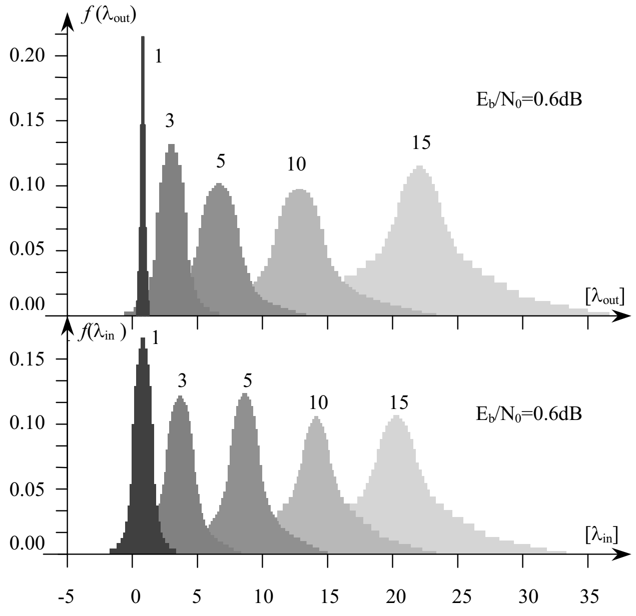

Fig. 3. Evolution of the input and output extrinsics

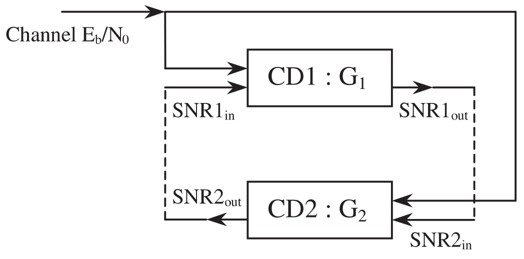

Fig. 4. Analytical density evolution model

A signal-to-noise ratio for such random variable can be defined as $\mathrm{SNR} = \mu^2/\sigma^2$ but since it is both Gaussian and consistent, then $\sigma^2 = 2\mu$ and, consequently, $\mathrm{SNR} = \mu/2$. This evaluation gives the best approximation of the empirically measured variance.

Now we can observe the input and output SNRs for each decoder denoted as $\mathrm{SNR1}_{\text{in}}$, $\mathrm{SNR1}_{\text{out}}$, $\mathrm{SNR2}_{\text{in}}$, and $\mathrm{SNR2}_{\text{out}}$, at each iteration as shown on Fig. 4. A non zero E_b/N_0 from the channel helps CD1 to produce a non zero $\mathrm{SNR1}_{\text{out}}$ for the extrinsic information despite starting with $\mathrm{SNR1}_{\text{in}} = 0$. So, for given value of E_b/N_0 the output SNR of each CD is a non-linear function of its input, denoted as G1 for CD1 and G2 for CD2. Thus we have:

$$\mathrm{SNR1}_{\text{out}} = \mathrm{G1}(\mathrm{SNR1}_{\text{in}}, E_b/N_0) \tag{1}$$

$$\mathrm{SNR2}_{\text{out}} = \mathrm{G2}(\mathrm{SNR2}_{\text{in}}, E_b/N_0) \tag{2}$$

From the Fig. 5 it follows that $\mathrm{SNR2}_{\text{in}} = \mathrm{SNR1}_{\text{out}}$, so we have

$$\mathrm{SNR2}_{\text{out}} = \mathrm{G2}(\mathrm{G1}(\mathrm{SNR1}_{\text{in}}, E_b/N_0), E_b/N_0) \tag{3}$$

The G1 and G2 functions can be evaluated either directly from the histogram of output λ's from the previous decoder

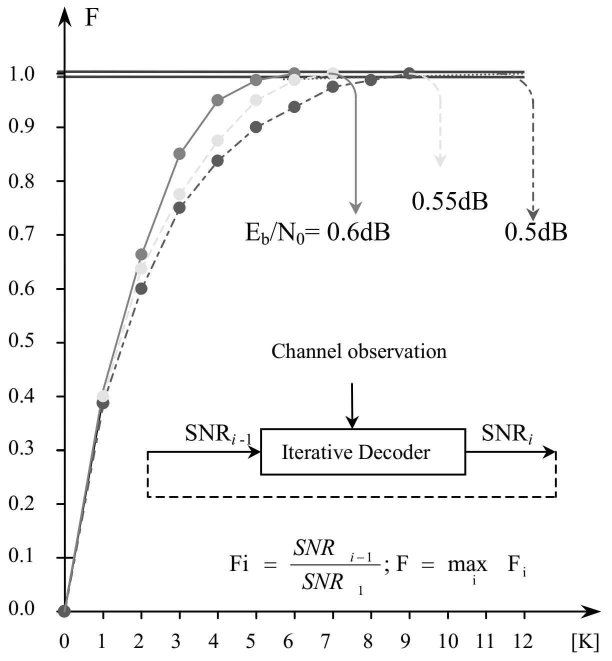

Fig. 5. Noise figure versa the number of iterations

or to generate input λ's from a consistent Gaussian density with mean μ and variance 2μ. In our simulations we use the former model to suit our intention and SNRs are computed from the actual λ-histograms as $E\{\lambda\}/2$.

The decoder convergence is assessed by measuring the change in the extrinsic information's SNRs from one to the next iteration which allows for defining a noise figure F as a ratio of the input SNR of CD1 at the beginning of the iteration to the output SNR of CD2 at the end of the iteration,

$$F_i = \text{SNR}_i 1_{\text{in}} / \text{SNR}_i 2_{\text{out}} = \text{SNR}_{i-1} / \text{SNR}_i \qquad (4)$$

So, the noise figure is bounded by 0 dB and, if its value at a given iteration is less than 1, this indicates an improvement in the SNR of the extrinsic information from the beginning to the end of a iteration. If that is the case for the entire range of CD1's input SNRs, then, according to [6], the turbo decoder will converge to the correct codeword. The increase of F is plotted on Fig. 5 for the rate 1/3, $N = 128$, [1,(5/7,5/7)] turbo code, for some values of E_b/N_0.

Let now observe a typical variation of codeword error rate, or frame error rate (FER), with the number of iterations, for the same code at $E_b/N_0 = 0.6$ dB, given on Fig. 6.

A careful observation allows for drawing an analogy between Figs. 5 and 6. We can see on Fig. 6 that a FER of about 3×10^{-3} is achieved with 10 iterations and a further increase of the number of iterations do not significantly reduce this error rate, reaching a value of about 1×10^{-3} at 20 iterations. However, below 10 iterations, the difference is great – one order of magnitude between 6 and 4 iterations. The conclusion is that after the 10th iteration, the decoder is wasting effort (and time) by continuing to iterate on about 95% of frames that are already decoded by then.

On Fig. 5, there is a dramatic increase of the F value in approximately first $6 \div 8$ iterations, reaching $F > 0.9$. The following next iterations allow only for a negligible increase. So, this $F \geq 0.9$ might be appropriate values for a threshold to cut off further iterations because the decoder has already gained enough reliability to come to a correct decision.

IV. Simulation Results

The performance results for this new stopping rule are presented in this section. We simulated rate 1/3 turbo codes with very short block size, $N = 128$ bits, with $K = 10$ iterations, for some threshold values chosen to cover a reasonable range of undetected frame error rates (FER). The performance for each threshold is then compared with the performance of the magic genie rule and the performance of the turbo decoder operating with fixed 10 iterations.

Fig. 7 shows the results concerning the average number of iterations per decoded frame as a function of the bit signal to noise ratio, E_b/N_0, for the chosen threshold values.

From these plots we can conclude that the average number of iterations is roughly between 4 and 6 for E_b/N_0 near the so called "waterfall" region where the bit error rate (BER) changes most abruptly. The number of iterations for each threshold matches the corresponding number of iterations on Fig. 5. Es expected, the average decoding speed increases

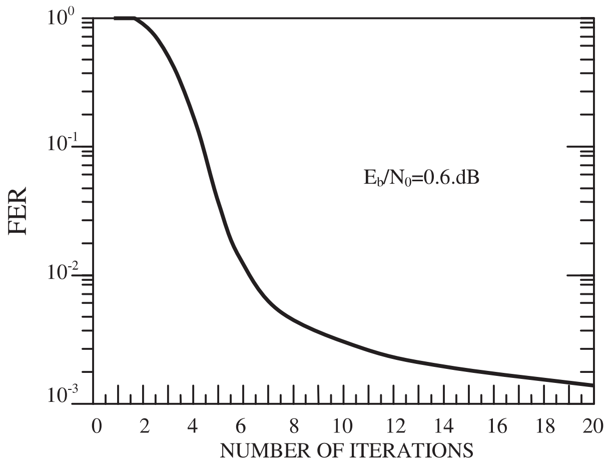

Fig. 6. Variation of FER with the number of iterations

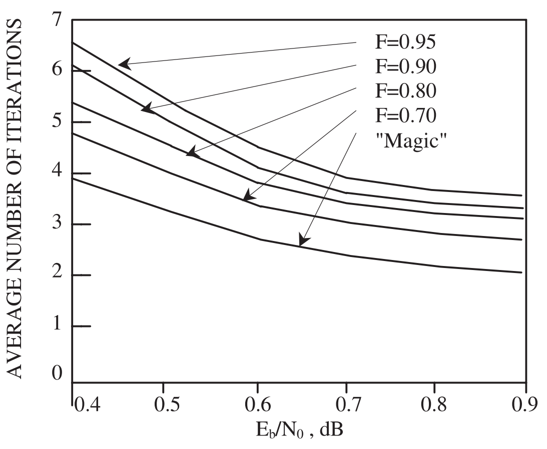

Fig. 7. Average number of iterations

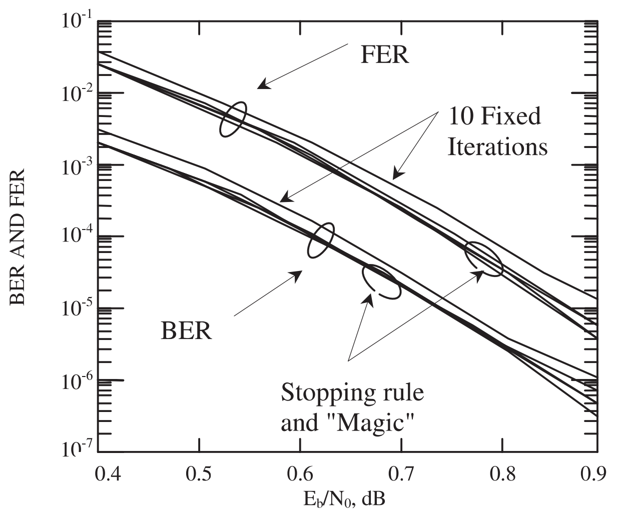

Fig. 8. FER and BER for the stopping rule

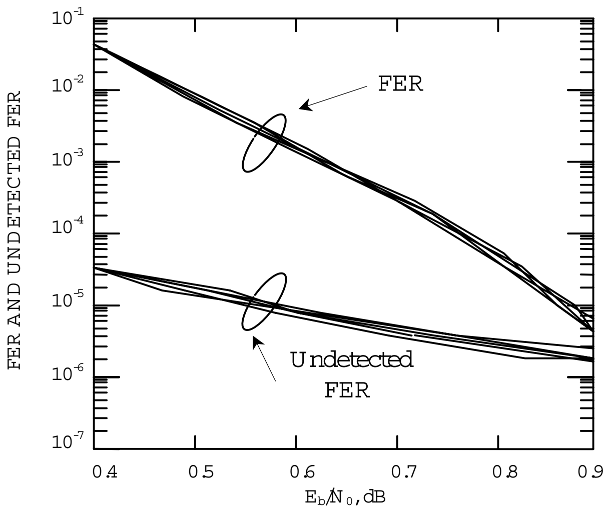

Fig. 9. FER and undetected FER for the stopping rule

with E_b/N_0, for the decoded sequences converge to the correct codeword in fewer iterations. Also, as it is common for all soft rules, there is a consistent offset from the average number of iteration required by the magic rule.

Both, the FER and BER performance of this rule are compared with the reference FER and BER performance curves for $K = 10$ fixed iterations per decoded frame and for the magic stopping rule with 20 iterations. The results for the chosen thresholds are shown on Fig. 8.

Having in mind the skewness of the statistics in the first iteration we didn't allow the decoder to take decisions after the first iteration, for there is a small number of first step decisions and they usually result in falsely decoded bits. By applying this measure, we noticed that the improvement in BER and FER is much greater than loss in speed, so it has a positive influence on overall performance.

The Fig. 8 indicates that the overall error rates of the turbo code employing this stopping rule are noticeable better than those for 10 fixed iterations as in any other soft stopping rule case. In fact they are nearly equal to the error rates achieved by a decoder using 20 iterations. Furthermore, it can also be seen that this rule has no characteristic error floor of its own (a matter of the length of the simulations), but the floor constrain is set by the inherent turbo code floor.

To get a deeper insight into the FER composition one should consider four different conditions that can occur when stopping rules are used by turbo decoder, depending whether the decoded sequence is detected to be reliable or unreliable and whether it is actually correct or in error. First is the case of correct decoding when stopping rule is satisfied at some iteration, $k < K_{\max}$, and the decoded sequence is correct. Second case is when the rule is satisfied but the sequence is actually in error which produces an undetected error. Next case is when the rule fails to stop the decoder in $k = K_{\max}$ iterations and the decoded sequence is indeed incorrect which corresponds to a detected error. The last case is when the sequence is declared as unreliable but is actually correct which means a falsely detected error. A good stopping rule has a small undetected error probability and a small probability of falsely detected errors.

The Fig. 9 plots the overall FER and undetected FER for this rule. We can observe that the undetected error rates decrease slowly over the entire range of tested E_b/N_0 values. Naturally, there is a small difference in FER values due to the level of the thresholds. At low E_b/N_0 the detected errors dominate the overall FER, whereas at high E_b/N_0 values the undetected errors are more contributory.

If the performance of this rule is compared with the other mentioned soft rules, one can see neither significant advantage in favor of this rule nor significant disadvantage. However, if we consider the soft rules from computational effort and implementation complexity aspects, then this rule certainly has a comparative advantage.

V. Conclusion

In this article we propose a new type of stopping rules that can be used to reduce the average number of iterations to decode a turbo code. This type of stopping rules belongs to the class of soft stopping rules but, for difference, it is based on the evolution of the probability density function (pdf) of the extrinsic information. A noise figure is defined through the pdf parameters as a measure of quality of the extrinsic and its values are used to set up appropriate stopping threshold.

We assessed our stopping rule in accordance with the evaluation process applied to evaluate other soft rules [4].

Though the performance simulation doesn't show any significant performance improvement for the range of tested thresholds, this rule has an advantage of low computational burden and low complexity requirements which make it more suitable for implementation.

References

[1] C.Berrou, A.Glavieux and P.Thitimajshima, “Near Shannon Limit Error Correcting Coding: Turbo Codes”, *Proceedings 1993 IEEE Int.Conf. on Comm.*, pp.1064-1070, Geneva, May 1993.

[2] L.Bahl, J. Cocke, F. Jelinek and J. Raviv, “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate”, *IEEE Transactions on Information Theory*, vol.IT-20, 1974.

[3] S.Benedetto, D.Divsalar, G.Montorsi and F.Pollara, “Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes”, *The Telecomm. and Data Acquisition Prog. Rep.*, Jet Propulsion Laboratory, Pasadena, Sep’96

[4] A.Matache, S.Dolinar and F.Polara, “Stopping Rules for Turbo Decoders”, *TMO Progress Report* 42-142, August 15, 2000.

[5] T.Richardson, A.Shokrollahi and R.Urbanke, “Design of probably Good LDPC Codes”, *IEEE Transactions on Information Theory*.

[6] S. ten Brink, “Convergence of Iterative Decoding”, *Electronics Letters*, vol.35, May 24, 1999.

[7] D. Divsalar, S. Dolinar, and F. Pollara, “Iterative Turbo Decoder Analysis Based on Density Evolution”, *TMO Progress Report*, pp.42-144, February 15, 2001.