A Turbo Codes Research Tool 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

Richardson et al. [3,4] used similar probability density evolution to compute iterative decoding thresholds for LDPC codes over a binary input AWGN channel. Similarly, S. ten Brink [5] developed a method for analysing the convergence of the decoder based on the evolution of mutual information. D.Divsalar et al.[6] apply these analyses to gain new insights into designing new turbo like code structures. The method in this article is similar to all of these approaches. Our main contribution is a tool which is capable to easily distinguish the qualitative values of compared turbo code structures. Since the method is based on the extrinsic's evolution, the extension to all iteratively decoded binary codes is straightforward.

In the next section we shortly explore the components of a turbo code and than develop a model to track the density evolution of the extrinsic's probability density function (pdf). Finally we apply the models in a simulation process to confirm some results known from the analytical constituent encoder design process. In our simulations we use modular "C", "C^{++}" and "MATLAB" programs which are modified to support statistical evaluation of the extrinsics. Being independent of the program in use , the results prove our feelings that for any given turbo code, the speed and manner of its extrinsics probability density evolution somehow represents a unique comparable "fingerprint" for the code.

II. Turbo Codes Background

The turbo codes (TC) are a class of FEC (forward error control) codes, known as parallel concatenation of two or more recursive systematic convolutional (RSC) codes (RSCC) produced by a turbo encoder (TE) composed of two or more component RSC encoders (CEs) with input to each but one CE permuted by an interleaver of length N. For further considerations we will use TCs with only two concatenated CEs, CE1 and CE2, with overall code rate $R = 1/3$, as shown on the Fig. 1. It is proved that these codes can perform very close to the Shannon limit.

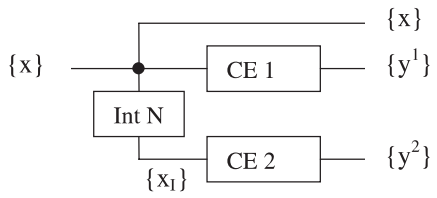

Fig. 1. Turbo encoder structure

Due to the interleaver, both the coding and decoding procedures are carried out on blocks of information sequences of length N plus some tail bits used to drive the constituent encoder trellises to all zero path, for decoder to know the beginning and the end of the codewords. Unfortunately, the length of N information bits can, and usually contains sequences, termed as self terminating (ST) sequences, which prematurely drives the CEs to the all zero path thus producing error events. Note that a codeword is also an "error event".

Basically, the performance of a turbo-code composition is determined by five factors: constituent encoders design, interleaver design, decoding algorithm, interleaver size and number of iterations. To achieve better coding gain, the latter three factors require either increases in delay or in complexity. The former two factors, however, can be thought of as a matter of "a good choice".

The most important impact of the interleaver on the overall turbo code performance is its information weight distribution from one to the other component encoder input.

As it can be seen from the Fig. 1, the parity vector $\boldsymbol{y}^1$ depends on $\boldsymbol{x}$ and CE1, while the $\boldsymbol{y}^2$ is determined by $\boldsymbol{x}$, CE2 and the interleaver. An explanation of the interleaver's role is almost obvious: If the sum of Hamming weights of $\boldsymbol{x}$ and $\boldsymbol{y}^1$, $W(\boldsymbol{x})$ and $W(\boldsymbol{y}^1)$, are "low", then, in order to brake up the ST sequence for the second encoder, the interleaver should provide such a permutation $\boldsymbol{x}_I$ which will yield a "high" weight contribution by the second parity vector $\boldsymbol{y}^2$.

A satisfactory overall weight of the TC naturally depends on length of N. Yet, there is no mathematical framework to

[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

solve this problem for all ST sequences and the value of N is often heuristically determined. In other words, for a given structure of the constituent RSCCs and the size of N, the goal mapping for each input sequence can not be determined. One may find some efforts in the literature but the problem of a perfect interleaver is still away from an exact science and, for given conditions, a "good choice" should be worked out.

The design of turbo code component encoders should also help solving the problems caused by ST sequences.

A constituent RSCC is commonly denoted in matrix D form as $[1, d(D)/g(D)]$ where the first term provides for the systematic outputs when multiplied by input sequence $x(D)$, and the second term, the quotient of the feed forward $d(D)$ and the feed back polynomial $g(D)$, also multiplied by $x(D)$, provides for the parity outputs. More conveniently, the polynomials are usually given in octal form.

It is known that the asymptotic performance of a convolutional code can be improved by maximising the free distance of its distance spectrum. However, due to the presence of the interleaver it is extremely difficult to find the distance spectrum for the TC. This is overcome by averaging the weight spectrum over all interleavers using the notion of an "uniform interleaver"[8]. The basic idea is to combine the distance spectra of the two CEs to create a super spectra which contains all possible combinations of all the paths. This technique shows that there is a need to alter the usually desired characteristics of the CEs in order to suit the characteristics of the TC.

By choosing recursive CE, any weight one information sequence can not be a ST sequence due to the infinite impulse response of the recursive structure, and we are to consider higher input weights. Due to the interleaver embedded into the TC structure, most of the low weight ST sequences will have a chance of being broken up to produce a high weight at the second CE. Also, heavier input weights, after permuting, will have a much higher chance of being broken up than those of lower weight. It is well known that the minimum information weight in the error events of a RSCC is $w_{\min} = 2$, and this particular input weight is most contributory one for the bit error probabilities ranging from 10^{-3} down to 10^{-10}.

Hence the problem of finding good codes for TCs lies in finding RSCCs that have maximum output weight for weight two input sequences which define a figure of merit for TCs named as effective free distance:

$$d_{\text{free,eff}} = 2 + 2z_{\min} \tag{1}$$

where $z_{\min}$ is the weight of minimum weight parity sequence generated by RSC CE with a weight 2 input. So, a TC_1 composed by two equal $\text{CE}_1 = [1, 5/7]_8$ will have $d_{\text{free,eff}} = 10$, while TC_2 composed by two $\text{CE}_2 = [1, 7/5]_8$ will have $d_{\text{free,eff}} = 8$ and the "good choice" is quite obvious. The higher order polynomials, however, have much more different polynomials of the same order among which many of them have the same value of respective $d_{\text{free,eff}}$. For example a 32-state CE has 15 different polynomials yielding the same value of $z_{\min}$. The "good choice" then should search for maximising $d_{\text{free,eff}}$ for minimum weight parity sequence generated by a weight 3 input, and so on. It appears that the number of nearest neighbours, defined as the number of paths having the same effective distance, is also an important parameter which should be minimised. It is also noticed that the order of importance of different parameters varies...

Having also in mind the above mentioned unrealisable uniform interleaver, the simulation of TC's BER and FER performances remains as a unique model to evaluate both the CEs and the interleaver "good choices" to suit one another. The method in this article offers another possibility.

III. Density Evolution Model

The optimum decoding of TCs is the maximum likelihood (ML) decoding algorithm applied to the TC trellis structure. However, due to the interleaver embedded into the TE's structure, the trellis will have an extremely large number of states thus making the whole ML decoding process almost unrealisable in practice. Since the TC is a concatenation of component codes, a more practical solution is to sequentially decode the component codes in a iterative fashion using one decoder at a time for each code. A simple SISO (Soft-In Soft-Out) maximum a posteriori (MAP) decoder which minimise the probability of bit error appears to be the best solution for component decoders (CDs). The MAP algorithm provide as an output a real number which is a measure of the probability of error in decoding a particular bit. This extra information termed as extrinsic information, λ_i, can be passed as input to the second CDs allowing it to create its own extrinsic to 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 sequence of length N, as shown on Fig. 2.

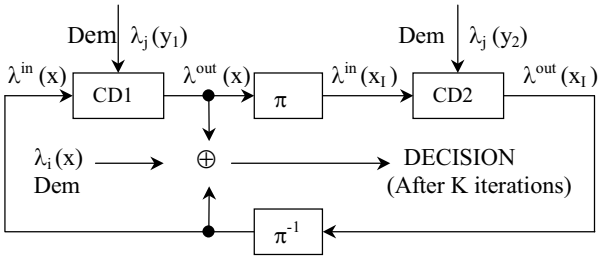

Fig. 2. Iterative TD with two MAP CDs

Such iterative decoder can be considered as a non-linear dynamical feedback system. Extrinsic information messages $\{\lambda_i\}$ 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 $\{\lambda_j\}$ from all other bits but the i-th. So, if we assume that the all-zero codeword is transmitted (with BPSK modulation 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 on Figs. 1 and 2 is large and random, the extrinsics λ_i are independent and identically distributed with probability density function $f(\lambda)$. As shown in [4], this pdf is consistent ($\lambda = \log[f(\lambda)/f(-\lambda)]$), its mean,

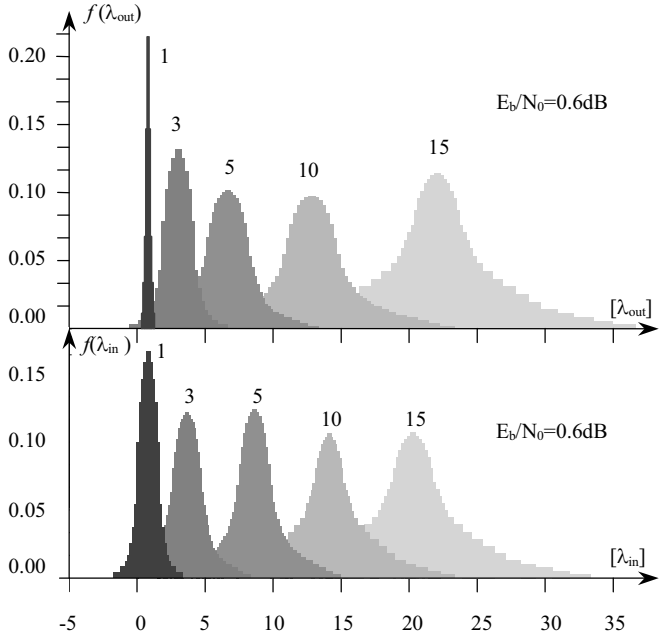

Fig. 3. Evolution of the input and output extrinsics

$\mu = \mathrm{E}(\lambda)$, is discrimination between the two densities $f(\lambda)$ and $f(-\lambda)$, and the error probability, $e = \mathrm{Prob.}\{\lambda < 0\}$, 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 $f(\lambda_{\mathrm{in}})$ and $f(\lambda_{\mathrm{out}})$ 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 $\sigma^2 = \mathrm{Var}(\lambda)$.

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 $\mathrm{SNR1_{in}}$, $\mathrm{SNR1_{out}}$, $\mathrm{SNR2_{in}}$, and $\mathrm{SNR2_{out}}$, 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 $\mathrm{SNR1_{out}}$ for the extrinsic information despite starting with $\mathrm{SNR1_{in}} = 0$.

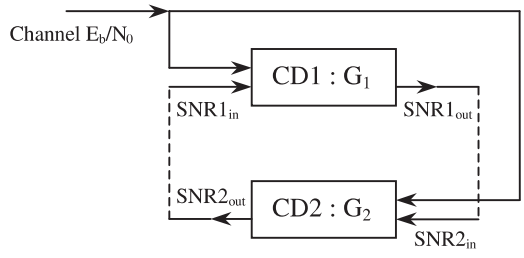

Fig. 4. Analytical density evolution model

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:

$$\mathrm{SNR1_{out}} = \mathrm{G1}(\mathrm{SNR1_{in}}, E_b/N_0) \quad (2)$$

$$\mathrm{SNR2_{out}} = \mathrm{G2}(\mathrm{SNR2_{in}}, E_b/N_0) \quad (3)$$

From the Fig. 4 it follows that $\mathrm{SNR2_{in}} = \mathrm{SNR1_{out}}$, so we have

$$\mathrm{SNR2_{out}} = \mathrm{G2}(\mathrm{G1}(\mathrm{SNR1_{in}}, E_b/N_0), E_b/N_0) \quad (4)$$

The G1 and G2 functions can be evaluated either directly from the histogram of output λ's from the previous decoder or to generate input λ's from the consistent Gaussian density with mean μ and variance 2μ. In our simulations we use the former model and SNRs are computed from the actual λhistograms as $\mathrm{E}\{\lambda\}/2$.

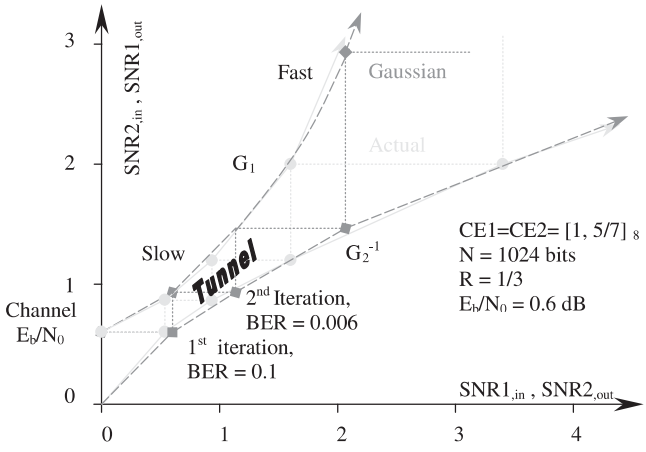

Fig. 5. Iterations and convergence of $[1, 5/7\text{-}5/7]_8$ TC

The decoder convergence is assessed by tracking the evolution of the extrinsic information's SNRs in each half iteration. As it is shown on Fig. 5, the analytical model is to plot the output SNR of CD1 versus its input, and the input SNR of CD2 versus its output SNR. For this case we followed the extrinsics evolution of the memory 2, rate 1/3, [1,5/7] turbo code, at $E_b/N_0 = 0.6$ dB.

Both curves for G_1 and G_2^{-1} obtained from actual density evolution are just sequences of discrete points joined by linear interpolation to give estimates at intermediate points. Actually, the Fig. 5 shows the progress of the decoder's iterations. The improvement in the SNR of the extrinsics and corresponding BER follows a staircase path reflecting at right angles between the G_1 and G_2^{-1} curves.

The steps are large when the bounding curves are far apart, and small when they are close together in which case the improvement in BER slows down , for many iterations are required to get through the narrow iterative decoding tunnel between the curves. If the iterative decoding process successfully passes through the tunnel, the convergence becomes rapid as the curves part more at the higher SNRs.

Since the curves are a reflection of the density evolution of the extrinsics we believe that the manner in which the curves

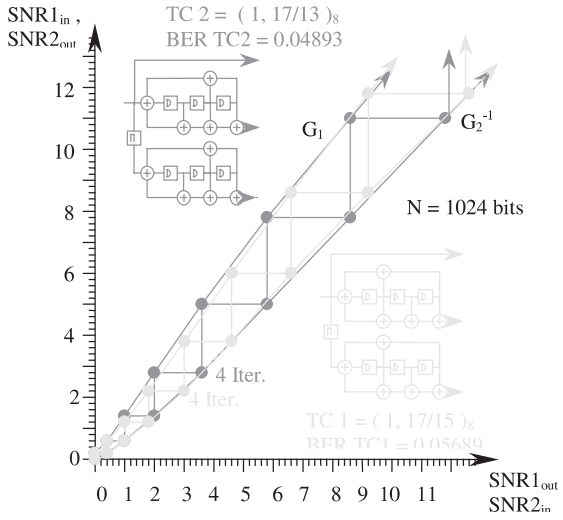

Fig. 6. Comparison at $E_b/N_0 = 0.4$ dB

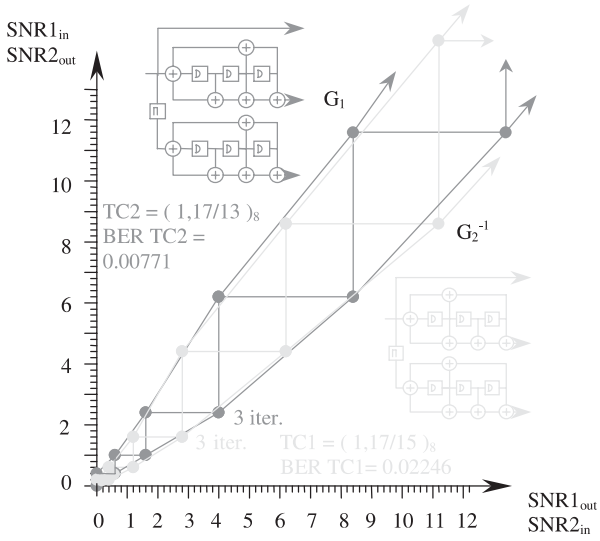

Fig. 7. Comparison at $E_b/N_0 = 0.5$ dB

part might be interpreted as a response of the component encoder's quality. So, through a simulation process we tested a number of CEs known as a "good choice" and compared the results for several E_b/N_0 values with the results of the CEs which are not classified as a "good choice" with respect to their distance properties.

The Figs. 6, 7, 8 and 9 represent test-comparison between TC1 with two CE1 = $[1,17/15]_8$ and TC2 with a pair of CE2 = $[1,17/13]_8$, the later being the best among the all possible, rate 1/2, memory 3, constituent RSCCs. It can be noticed on Fig. 6 that both TC1 and TC2 have the same properties nearby the waterfall region because the TC's performances at low E_b/N_0 values are mainly governed by the interleaver gain. By increasing the E_b/N_0 value the curves go apart and it becomes quite clear that TC2 is the better choice which is also confirmed by corresponding BER values.

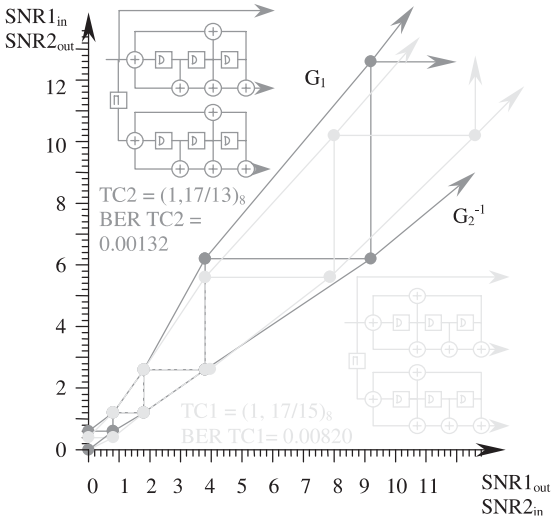

Fig. 8. Comparison at $E_b/N_0 = 0.6$ dB

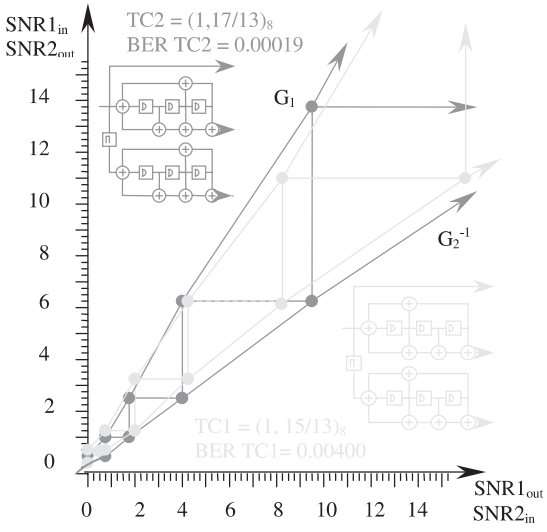

Fig. 9. Comparison at $E_b/N_0 = 0.7$ dB

IV. Conclusion

Basically, we introduced the main problems encountered in turbo codes design process. Then we adopted a model to trace the density evolution of the extrinsics and explored the possibility of its application in the evaluation process of a turbo code constituent encoders for several E_b/N_0 values. The simulations carried out for a "good" and a "not so good" CEs confirmed our expectations. During the simulations we used the same type and the length of the interleaver. It is quite obvious that we might have kept the CEs as constant and to vary the type or the length of interleaver. Also, by using this density evolution model of a tested turbo code as a benchmark one can evaluate any turbo or turbo-like structure.

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] T.Richardson, R.Urbanke, “The Capacity of Low Density Parity Check Codes Under Message Passing Decoding”, *IEEE Transac. on Information Theory*.

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

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

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

[7] S. Benedetto and G. Montorsi, “Average Performance of Parallel Concatenated Block Codes”, *El. Letters*, 31(3): 156-158, Feb 1995.