Improving turbocode performance by cross-entropy

Abstract—Turbocodes have been introduced as a new encoding technique for error correction over noisy channels. Their performance is close to theoretical limits. In this paper, we describe the use of the cross-entropy principle in the turbocode domain and we show that it is possible to use it to obtain an additional gain in terms of error correction performance.

I. INTRODUCTION

Turbocode technology [1] is an encoding technique used for communication over noisy channels. Their performance in terms of error correction is close to theoretical limits and outperforms the preceding communication standards. In spite of this fact, it is known that turbocode decoding is in fact suboptimal. For instance, it is easy to show that maximum likelihood decoding would give better performance if it could be performed. In this paper, we show how the principle of cross-entropy can be used with turbodecoding in order to improve error correction performance. In part II, we briefly recall the principle of turbocode encoding and decoding. In part III, we introduce the cross-entropy measure between probability density functions and we derive a simplified expression for it in the case of turbocodes. In part IV, we describe the use we made of cross-entropy minimization for turbocode decoding. The gains obtained by our technique are illustrated with some examples. Finally in part V, we draw conclusions from our work and emphasize some possible axes of research for further use of cross-entropy principle in turbocode technology.

II. ENCODING AND DECODING TURBOCODES

Figure 1 describes the encoding of turbocode. Let $(X_1, X_2, ..., X_N)$ be a frame of $N$ symbols to be encoded. These symbols are first driven into a convolutional encoder and produce the first set of redundancies $(Y_1, Y_2, ..., Y_N)$. In parallel, the data frame $(X_1, X_2, ..., X_N)$ is permuted by a fixed permutation $\Pi$. The permuted symbols are also driven into a convolutional encoder (generally of same type) and produce the second set of redundancies $(\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_N)$. The turbocode codeword is then formed by the concatenation of the data frame and the two redundancy frames, i.e. $(X_1, X_2, ..., X_N, Y_1, Y_2, ..., Y_N, \hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_N)$. In order to avoid turbodecoding side effects, terminated encoding is generally used. In our work, we considered tail-biting encoding [3]. In this scheme, the content of internal latches of convolutional encoders are initially set in such a way that, after the encoding of the data frame, internal latches return to the identical contents. The initial and final state of the encoder can be computed by solving a linear system depending on the encoding frame. This kind of termination was considered since it is more natural from an algebraic point of view. Figure 2 describes the decoding process of turbocode. The received codeword is denoted $(R_{X_1}, R_{X_2}, ..., R_{X_N}, R_{Y_1}, R_{Y_2}, ..., R_{Y_N}, R_{\hat{Y}_1}, R_{\hat{Y}_2}, ..., R_{\hat{Y}_N})$. The received samples $(R_{X_1}, R_{X_2}, ..., R_{X_N})$ (resp. $(R_{Y_1}, R_{Y_2}, ..., R_{Y_N})$) corresponding to data information symbols (resp. the first redundancy set) are driven into a soft-in soft-out (SISO for short) decoder and produce extrinsic a posteriori values $(E_{X_1}, E_{X_2}, ..., E_{X_N})$ on decision of the frame $(X_1, X_2, ..., X_N)$. These a posteriori values are then permuted by the permutation $\Pi$ and incorporated as additional a priori entries to the second SISO decoder together with the sets $(R_{X_{\Pi(1)}}, R_{X_{\Pi(2)}}, ..., R_{X_{\Pi(N)}})$ and $(R_{Y_{\Pi(1)}}, R_{Y_{\Pi(2)}}, ..., R_{Y_{\Pi(N)}})$. New extrinsic a posteriori values $(E_{X_{\Pi(1)}}, E_{X_{\Pi(2)}}, ..., E_{X_{\Pi(N)}})$ are then produced. These values are then permuted by the permutation $\Pi^{-1}$ and incorporated back as a priori entries to the first SISO decoder. Along iterations, extrinsic values are exchanged by both decoders and their values are refined. After a fixed number of iterations, received samples $(R_{X_1}, R_{X_2}, ..., R_{X_N})$ and extrinsic values are finally added and a decision on $(X_1, X_2, ..., X_N)$ is drawn.

Concerning convolutional codes, the first SISO decoder that was invented is the BCJR [4] also known as the Maximum A Posteriori algorithm (MAP for short). This algorithm has a high computational cost and involves multiplications. Thus it is not very suitable for implementation. In [5], simplifications of this algorithm were considered. The new derived algorithm called SUBMAP only involves additions and is now the most popular decoding algorithm for implementation. The downside
is a slight loss in terms of convergence.

III. CROSS-ENTROPY MINIMIZATION

Definition 3.1: Let \( p \) and \( q \) be two probability density functions defined on \( \Omega \). Then the relative cross entropy \([6]\) \( CE(p, q) \) between \( p \) and \( q \) is defined as:

\[
CE(p, q) = \int_{\Omega} p(x) \log \frac{p(x)}{q(x)} \, dx
\]

We have \( CE(p, q) \geq 0 \) and \( CE(p, q) = 0 \) if and only if \( p \) is equal to \( q \) almost everywhere.

Cross-entropy measure has been early identified as a useful criterion in error decoding. For instance, Battié [7], [8] considered the minimization of cross-entropy as a method of decoding in the domain of concatenated codes. On the other hand, in [9], Moher and Gulliver studied the relations between the turbodecoding principle and cross-entropy minimization. In fact, they showed that although cross-entropy minimization is computationally feasible in the domain of turbocodes, the turbodecoding principle can be considered as a practical attempt of cross-entropy minimization.

In the context of decoding, cross-entropy minimization is used the following way: the distribution \( p(x) \), for a random variable \( x \) with an a priori distribution \( q(x) \), is determined in such a way that it minimizes \( CE(p, q) \), amongst all possible distributions acceptable for \( x \). This latter point means that samples of \( p(x) \) are supposed to respect the constraints on the variable \( x \) (parity constraints for instance).

In the case of turbocodes, a priori and a posteriori distributions are explicitly computed by the turbodecoding procedure. More precisely, at a given iteration, the decoder receives data samples \( R_{X_i} \) and incoming extrinsic values \( E_{X_i}^{in} \). As an output, the new extrinsic values \( E_{X_i}^{out} \) are produced. Following Hagenauer’s notations [10] and with respect to the previous notations \( p(x) \) (resp. \( q(x) \)) is the probability distribution function of samples \( L_{i}^{out} = L_c R_{X_i} + E_{X_i}^{out} \) (resp. \( L_{i}^{in} = L_c R_{X_i} + E_{X_i}^{in} \)), where \( L_c \) is the channel reliability. From [10], we have in fact:

\[
p(x_i = \pm 1) = \frac{\exp(\pm L_{i}^{out})}{1 + \exp(\pm L_{i}^{out})}
\]

\[
q(x_i = \pm 1) = \frac{\exp(\pm L_{i}^{in})}{1 + \exp(\pm L_{i}^{in})}
\]

Moreover, since the probability space \( \Omega \) is a discrete space, the cross-entropy expression becomes:

\[
CE(p, q) = \sum_{i=1}^{N} \left( p(x_i = +1) \log \frac{p(x_i = +1)}{q(x_i = +1)} + p(x_i = -1) \log \frac{p(x_i = -1)}{q(x_i = -1)} \right)
\]

Given the two previous sets of expressions, it is then possible to compute explicitly \( CE(p, q) \). However, the resulting expression is much more complicated than computations involved in the SUBMAP algorithm. Since we wish to investigate the benefits of cross-entropy in this latter case, we need to simplify the expression defining \( CE(p, q) \). To this end, we make two hypotheses. First, when a sample \( L_{i}^{in} \) has a low value, then it is also the case for \( L_{i}^{out} \). Second, when the value \( L_{i}^{in} \) is high, then \( L_{i}^{out} \) is also high and has the same sign as \( L_{i}^{in} \). The first hypothesis states that uncertain incoming samples cannot produce samples with high confidence. The second hypothesis states that whenever we obtain a high confidence decision for \( x_i \), this decision is not changed any more with the next iterations of turbodecoding. These two hypotheses are quite relevant with what is observed with turbodecoding. The cross-entropy is now defined by:

\[
CE(p, q) = \sum_{i=1}^{N} \left( \frac{\exp(-L_{i}^{in})}{1 + \exp(-L_{i}^{in})} \log \frac{\exp(-L_{i}^{in})}{\exp(-L_{i}^{out})} \right)
\]

Suppose first that \( L_{i}^{in} \) has a large positive value. So, by virtue of the second hypothesis, this is also the case for \( L_{i}^{out} \). We have then:

\[
\log \left( \frac{1 + \exp(L_{i}^{in})}{\exp(L_{i}^{in})} \right) \approx \exp(-L_{i}^{in}) \frac{1}{2}
\]

and:

\[
\log \left( \frac{1 + \exp(-L_{i}^{in})}{\exp(-L_{i}^{in})} \right) \approx -L_{i}^{in}
\]

If we substitute those terms for \( L_{i}^{in} \) and \( L_{i}^{out} \), then the summand becomes:

\[
\frac{\exp(-L_{i}^{out})}{1 + \exp(-L_{i}^{out})} \left( \frac{\exp(-L_{i}^{out})}{\exp(-L_{i}^{in})} \right) (L_{i}^{out} - L_{i}^{in})
\]

Now if we suppose that \( L_{i}^{in} \) and \( L_{i}^{out} \) are large, it is clear that the first term of this sum is near 0. The second term, is also near 0, since \( \exp(L_{i}^{out}) \) is much larger than \( L_{i}^{out} \) and \( L_{i}^{in} \). The same reasoning gives the same conclusion if \( L_{i}^{in} \) has a large negative value.

With the hypothesis we made, the only remaining case of study is when \( L_{i}^{in} \) and \( L_{i}^{out} \) have low values. In this case, it is possible to use Taylor series, in order to obtain an equivalent for the summand. By hand, computations are rather lengthy but symbolic computation softwares can be used here. Using Maple™ [11], we obtained the following development:

\[
\left( \frac{\exp(-L_{i}^{out})}{1 + \exp(-L_{i}^{out})} \left( \frac{\exp(-L_{i}^{out})}{\exp(-L_{i}^{in})} \right) (L_{i}^{out} - L_{i}^{in}) \right) \approx \left( L_{i}^{out} + L_{i}^{in} \right)^2
\]

Using the latter expression, it is then possible to compute an approximation of the relative cross-entropy of incoming and outgoing likelihood ratios, using quite elementary arithmetical operations.
IV. PERFORMANCES

In turbodecoding iterations, it is well known that an appropriate scaling of extrinsic values is necessary in order to obtain good error correcting performance. More precisely, at the iteration $m$ of the process, if $E_{X_i}^{out,m}$ is an output extrinsic value of the decoder, then at step $m + 1$, the corresponding incoming extrinsic value $E_{X_i}^{in,m+1}$ will be set to $\alpha_mE_{X_i}^{out,m}$ with $0 \leq \alpha_m \leq 1$. Generally the $\alpha_m$ values are chosen quite low in the early iterations and grow with $m$. This choice is justified by the fact that extrinsic values are supposed to become more and more reliable with iterations. Another point to note is that the $\alpha_m$ values are also generally statically chosen, i.e. their values do not depend on the decoded frame.

The computation of the cross-entropy gives a reliability criterion. Indeed, as stated before, a correct decoding should minimize the relative cross-entropy between incoming and outgoing likelihood distributions. Thus, at iteration $m$, the quality of the set of extrinsic values $E_{X_i}^{out,m}$ can be inferred from cross-entropy: the lower the cross-entropy, the more reliable extrinsic values. As a consequence in our experiment, the scaling factor $\alpha_m$ was set in order to advantage good distributions and to penalize bad ones. More precisely we use the following defining equation:

$$\alpha_m = 1 - \frac{CE(L_i^{out,m}, L_i^{in,m})}{N}$$

In order to lighten the computation of cross-entropy, we replace the true definition by the following one:

$$CE(L_i^{out,m}, L_i^{in,m}) = \sum_{i=1}^{N} \frac{(\tilde{L}_i^{out,m} - \tilde{L}_i^{in,m})^2}{8}$$

where values $\tilde{L}_i^{out,m}$ and $\tilde{L}_i^{in,m}$ are truncated in the interval $[-T, T]$ where $T$ is a threshold value, determined by simulation. Note that, with this latter definition, if $L_i^{out,m}$ and $L_i^{in,m}$ have large values of the same sign, then we have $\tilde{L}_i^{out,m} - \tilde{L}_i^{in,m} = 0$. Thus the corresponding term in the summation does not contribute to the cross-entropy and so our definition is compatible with the asymptotic derivation we made near infinity.

The following simulations were done with a binary turbocode with a frame length equal to $N = 2000$. The convolutional code used is the classical 8-state 13/15 code. The permutation $\Pi$ used is an ARP permutation [12]. More precisely, we have $\Pi(i) = (P \cdot i + S[i \mod 8]) \mod N$ with $P = 49$ and $S[0,7] = \{0, 129, 1621, 1593, 575, 1684, 160, 1806\}$. This set of values was obtained by performing a search to maximize the correlation girth [2] of the turbocode permutation. Encoding of this turbocode uses tail-biting termination. Figure 3 compares performance of fixed scaling strategy and cross-entropy scaling strategy for eight iterations of SUBMAP decoding. In the case of fixed scaling, it turns out in fact that a common value $\alpha_m = 1$ for all iterations gives the best results. This comes from the fact that this turbocode permutation was optimized against auto-correlation effects. Figure 3 exhibits a gain of 0.25 dB for the cross-entropy scaling strategy.

Figure 4 depicts comparisons for the same turbocode case but with a different number of iterations. In this figure, performance is reported for fixed scaling with 8 iterations and for cross-entropy scaling with 6 iterations. Performance in both cases is quite similar. This latter point is interesting for the prospect of circuit implementation. Indeed, in real communication applications, the number of turbodecoding iterations is generally reduced. The figure 4 shows that, for a given level of output performance, cross-entropy scaling allows the same performance as fixed scaling with fewer iterations. In implementations, this feature could be used to obtain a decoding architecture with a larger data throughput or with reduced energy consumption.

V. CONCLUSION

In this article, we have considered cross-entropy minimization in the area of turbocode decoding. An approximation formula was derived for cross-entropy and simulations show that cross-entropy minimization could be used in order to improve performance of turbodecoding system. This optimization technique seems attractive from an implementation point of view for high data decoding rate and for low energy consumption. In this latter case, since the overhead of computation of scaling
factor is quite low, with respect to the decoding algorithm, this kind of solution could be easily implemented in systems built over DSP. In the case of high rate transmission, further research will be necessary to deal with implementation aspects.

REFERENCES


