DESIGNING TURBO CODES FOR LOW ERROR RATES
Claude Berrou, Catherine Douillard and Michel Jézéquel

Abstract. This paper describes the ways to construct turbo codes using a parallel concatenation of convolutional codes, in order to minimize the effects of flattening. Bit Error Rates as low as $10^{-10}$ may thus be reached by the associated decoder at signal to noise ratios not far from the theoretical limits, for different data block sizes and various coding rates.

I. Introduction

Many composite code structures, using convolutional or algebraic (especially BCH) component codes, have been imagined in the past few years, so as to make the most of the iterative ("turbo") decoding properties. Nevertheless, obtaining quasi-optimal performance (i.e. very low error rates close to the theoretical limits), whatever the block size and whatever the code rate, seemed to be a difficult task for a single coding scheme. In particular, the so-called "flattening" effect was a brake on the use of Parallel Concatenated Convolutional Codes (PCCCs) when very low Bit Error Rates (BER) or Frame Error Rates (FER) were required, especially at high code rates.

In this paper, we describe a single turbo coding structure able to satisfy most demands as regards channel coding with quasi-optimal performance, at the cost of reasonable decoding complexity. This new code was recently proposed for the DVB-RCS (Digital Video Broadcasting-Return Channel over Satellite) standard as an alternative to the classical concatenation of a Reed-Solomon and a convolutional code with constraint length 7.

In the next chapter, we discuss the schedule of conditions in the search for a PCCC having quasi-optimal performance, in a wide range of uses, thanks to iterative decoding and we give answers to all items. Chapter III deals with the particular problem of interleaving design for PCCCs, in order to get large minimum distances. Finally chapter IV gives some examples of simulated performance with the actual conditions of implementation in silicon and compares them with the classical concatenation performance.

II. Key to powerful turbo coding

What is needed for powerful turbo coding is a sophisticated combination (concatenation) of small codes. Small codes, that is codes with small minimum distances (i.e. with small constraint lengths for convolutional codes), are essential in order to ensure convergence at very low Signal to Noise Ratios (SNRs). Indeed, when iterative processing is worked out, one always has to begin with one of the component decoders, that has necessarily to process data "beyond capacity" (the code rate of the component code is higher than the code rate of the composite code). Therefore the first decoded code must not be a complicated one, because the more complex it is, the worse it behaves beyond capacity.

1 The authors are with the Ecole Nationale Supérieure des Télécommunications (ENST) Bretagne, Brest, France.
Supported in part by Eutelsat and France Télécom.

© 1999 The Institution of Electrical Engineers.
Printed and published by the IEE, Savoy Place, London WC2R 0BL, UK.
Figure 1. Erroneous paths in a two-dimensional convolutional code. Case (a) with a small constraint length code. Case (b) with a larger constraint length and longer erroneous paths, producing more frequent locked error patterns.

Another reason to use small codes with the aim of iterative decoding, is to minimize the correlation effects. This problem is illustrated in Fig. 1 as it was in [1]. It is supposed for sake of simplicity that the encoded block, containing $k$ bits, is seen as a two-dimensional $\sqrt{k} \times \sqrt{k}$, with first, horizontal encoding by any convolutional code $C_1$, and second, vertical encoding by $C_2$, identical to $C_1$. In all figures, the dashes in both dimensions symbolize the path error packets at the output of the two decoders, at a particular step in the iterative decoding process (these packets do not contain only erroneous binary decisions but they indicate where a wrong path has been selected either by the decoder of $C_1$ or by the decoder of $C_2$). Fig. 1(a) represents a two-dimensional code using a small component code, that is with a small constraint length; Fig. 1(b) represents a two-dimensional code using a component code with a larger constraint length. The latter case is more sensitive to correlation effects during the decoding, because the longer error packets may produce more locked error patterns, which are recognizable as rectangles in the drawing.

Figure 2. Reducing correlation effects in two-dimensional decoding thanks to double-binary coding.

For a given convolutional code, there exists a simple means to reduce the correlation effects, as explained in [1], by substituting a double-binary code for the classical code, for each component. Figure 2 gives an example of substitution and shows the incidence on the erroneous paths. The length of these wrong paths is halved when double-binary coding is used (the same amount of information is used by the two decoders associated with binary and double-binary codes, therefore with half as
many transitions in the latter case and so finally with halved path error lengths), while each dimension of the block is divided by \( \sqrt{2} \) only. This leads to a lowered path error density and reduces correlation effects in the decoding (the rectangles have disappeared in Fig. 2(b)).

Moreover, combining double-binary convolutional coding with the tail-biting principle, extended to recursive encoders [2-4] yields a powerful component for turbo coding. The tail-biting, or circular principle, enables convolutional coding to be applied to any finite data sequence, without a termination problem. As far as turbo coding is concerned, where the termination problem falls on several component codes, Circular Recursive Systematic Convolutional (CRSC) coding is quite appropriate for block error correction.

Up to now, we have given some arguments for choosing a proper component code, that is a double-binary CRSC code, having good convergence and termination properties (note that the complexity of the double-binary decoder is roughly twice that of the binary decoder, but with a doubled data rate, giving about the same ratio complexity/data rate). But we also want to achieve low error rate protection, that is to construct a turbo code with a large minimum distance or asymptotic gain. As shown in the next chapter, double-binary coding also offers nice characteristics to take advantage of, as regards minimum distances.

III. Interleaving with local disorder

The performance of a PCCC, at low error rates, is essentially governed by the permutation that links the component codes. The simplest way to achieve permutation (or interleaving) in a block, is to organize it, if possible, as a rectangular matrix with \( M \) rows and \( N \) columns (the block containing \( k = M \cdot N \) bits). Then data are written linewise and read columnwise to achieve what is called regular (or uniform) interleaving. Regular interleaver behaves very well towards the error patterns with weights 2 or 3. This is because two or three neighbours before permutation will be scattered far from each other after permutation, and the associated distances will be large. But this kind of permutation is very sensitive to square or rectangle error patterns, like the one depicted in Fig. 3(a), in which we chose \( M = N = \sqrt{k} \) again. This error pattern corresponds, for each dimension, to the superposition of two weight 2 minimal error patterns, that is, for the code considered: 10000001 (the period of the recursive generator used -15 in octal - being 7). A more detailed explanation is given in [5]. If the sought for rate is 1/2 for instance, redundancy \( Y \) is punctured (one \( Y \) symbol out of two is transmitted), and the resulting distance of this particular codeword is 16, which is a rather low distance for a composite code.
Figure 3. Square or rectangle error patterns in binary and double-binary turbo codes.

Now, if we adopt the double-binary code of Fig. 3(b), other periodicities than that of the binary code have to be considered. Let us denote the four possible values of the input couple $(A, B)$ as:

$(0,0) : 0 ; (0,1) : 1 ; (1,0) : 2 ; (1,1) : 3.$

The periodicities of the double-binary input recursive encoder are summarized in the diagram of Fig. 4. It gives all the combinations of pairs of couples, different from 0, that make the encoder leave state 0 and return to it again. For instance, if the encoder, initialized at state 0, is fed by successive couples $(0,1) : 1$ and $(1,1) : 3$, it will recover state 0 again.

![Diagram](image)

Figure 4. Periodicities of the double-binary encoder of Fig. 3(b). Input couples $(A,B)$ : $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ are denoted, 0, 1, 2 and 3, respectively.

Fig. 3(b) gives two examples of rectangle error patterns, corresponding to the minimum distance of 18. To our knowledge, there is no other such pattern with a lower distance. So, there is a (slight) improvement in replacing binary by double-binary coding (this is due to not puncturing in the latter case), but not still sufficient to ensure very good performance at low error rates.

Classically, in order to increase distances given by rectangle error patterns such as those of Fig. 3, non uniformity is introduced in the permutation relations. Many proposals have been made in this direction, especially for the UMTS application. The CCSDS turbo code standard [6] may also be cited as an example of non uniform permutation. Nevertheless, the disorder that is introduced in the simple "linewise writing-columnwise reading" rule may affect the scattering properties about weight 2 or 3 error patterns. Indeed, what is gained, thanks to non uniformity, for higher weights (typically rectangle) patterns, may thus be wasted by poor distances associated with low weights. With double-binary codes, non uniformity can be introduced without any repercussion on the good scattering properties of regular interleaving, as explained in the following.

![Examples](image)
Let us suppose that couples are inverted (i.e. \((A,B)\) becomes \((B,A)\)) every other time before second (vertical) encoding, as depicted in Fig. 5. Then, many of the error patterns that were possible with the code of Fig. 3(b), do not remain error patterns with the local disorder that this periodic inversion produces. Fig. 5(a) gives such examples, while Fig. 5(b) shows possible remaining error patterns, whether the couples are inverted or not. But in these cases, the associated distances are large enough to provide sufficient asymptotic gains. The gain over minimum distances that the local disorder brings is not at all obtained to the detriment of the good scattering properties of the regular interleaving. This is simply because this remains regular interleaving, falling not on bits but on couples. Therefore, this appears to be a net gain in the search for large minimum distances. Note that there are various possibilities for achieving local disorder, for instance \((A, B)\) becoming periodically \((B, A+B)\) or others, before second encoding. There are still some investigations to make on local disorder in permutations, not only about the kind of modification to use but also about its periodicity.

IV. Performance and conclusions

Figure 6 shows two examples of simulated performance, for 57-byte blocks (typical of ATM applications) with coding rates 2/5 and 2/3. The turbo code component is that of Fig. 2(b) (with additional redundancy symbol \(11\) in octal notation – for rate 2/5). Circular termination is used and the permutation with local disorder is detailed in Appendix A. The component decoding algorithm is the Max-Log-MAP, with samples quantized on 4 bits. Good convergence, close to the theoretical limits, can be observed, thanks to the double-binary component code. Slight flattening is noticed for the highest code rate, giving a debasement of about 0.2 dB for a BER of \(10^{-9}\).

Gaussian channel. Respective theoretical limits: 1.2 and 2.3 dB (after [7,8]).

Figure 7. Comparison of the turbo code (TC with the parameters specified in Fig. 6) and
the classical concatenation (RS-C) of a convolutional code (constraint length 7) and a Reed-Solomon code (either (73,57) or (204,188)) for \( k = 456 \) or \( k = 1504 \). Gaussian channel. The ordinate indicates the \( E_b/N_0 \) required for a FER of \( 10^{-5} \). Given also are the corresponding theoretical limits.

Finally, Fig. 7 compares the turbo code presented in this paper and the classical concatenated scheme. The gain improvement goes from 2 to 3.1 dB for \( k = 456 \), depending on the code rate, and from 1.3 to 2.2 dB for \( k = 1504 \). These differences could be increased, especially for the latter case, by using the full MAP algorithm or even the Max-Log-MAP with metric correction \([9]\), but with the cost of more decoding complexity. For studies to come, we would rather investigate further the properties of local disorder and try to still increase the minimum distances.

Appendix A. Permutation for double-binary turbo code and \( k = 456 \)

The permutation is done on two levels, the first one inside the couples (level 1), the second one between couples (level 2):

\[ N = k/2 = 228 \]
\[ i = 0, \ldots, N-1 \] (address in natural order), \[ j = 0, \ldots, N-1 \] (incremental address for second encoding)

**level 1**
- if \( j \equiv 0 \mod 2 \), let \((A,B) = (B,A)\) (invert the couple)

**level 2**
- if \( j \equiv 0 \mod 4 \), then \( P = 0 \)
- if \( j \equiv 1 \mod 4 \), then \( P = N/2 + 4 \)
- if \( j \equiv 2 \mod 4 \), then \( P = 8 \)
- if \( j \equiv 3 \mod 4 \), then \( P = N/2 \)

Finally:
\[ i = 11j + P + 1 \mod N \]

(as explained in the main text, the first level is to increase the minimum distance. The second level is to combat the correlation during decoding)

References