Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured

Roland Gautier, Mélanie Marazin, Gilles Burel

BLIND RECOVERY OF THE SECOND CONVOLUTIONAL ENCODER OF A TURBO-CODE WHEN ITS SYSTEMATIC OUTPUTS ARE PUNCTURED

R. Gautier, M. Marazin and G. Burel

LEST – UMR CNRS 6165, University of Brest
CS 93837, 29238 BREST cedex 3, FRANCE
phone: + 33.(0)2.98.01.82.40, fax: + 33.(0)2.98.01.63.95, email: Roland.Gautier@.univ-brest.fr
http://www.univ-brest.fr/lest/tst

ABSTRACT

Turbo-codes are error-correcting codes used in powerful digital transmission systems to ensure a low binary error rate. This paper presents different approaches aimed at recovering a convolutional encoder in a non-cooperative context; it also explains the residual indetermination due to "equivalent" coders. Moreover, it reports on a new approach developed to recover the interleaved version of the second encoder of a Turbo-code when its systematic outputs are punctured.

1. INTRODUCTION

Error-correcting codes enhance the quality of communications by enabling the binary data stream to better withstand channel impairments such as noisy transmission channel, interference or channel fading. For this purpose, error-correcting codes introduce some redundancy in the informative binary data stream. Turbo-codes, first introduced in [2], belong to a family of error-correcting codes designed to reach the best correction capacity through an iterative decoding scheme. Figure 1 shows the generic form of a Turbo-code consisting of a parallel concatenation of two convolutional encoders and one interleaver. In a cooperative context, every parameter of the two encoders and of the interleaver are all known; the signal obtained on the receiver side can be demodulated and decoded to correct the errors due to the propagation channel and to the imperfections of digital transmission systems. In the case of military or spectrum surveillance applications, the parameters of the encoders are unknown, and the received data are the only available information. Under such conditions, the encoder parameters have to be blindly estimated.

This paper introduces a method dedicated to the blind recovery of the second convolutional encoder of a Turbo-code located after the interleaver in the case where its systematic parts are punctured. It explains why the previous methods developed in [1] and [4] are inefficient in this specific configuration used in the most recent digital communication systems. At first, section 2 gives the principle of convolutional codes and introduces the concept of equivalent code, which is essential for encoder recovery. Section 3 describes the approach we developed from the techniques proposed in [3] and [5] in order to blindly estimate the convolutional encoder parameters in a well-conditioned case. Moreover, it also explains briefly how the generator matrix of a convolutional code can be recovered by application of the methods developed in [1] and [4]. Finally, Section 4 reports on the blind estimation of the second encoder in the case of a classical implementation of the Turbo-code.

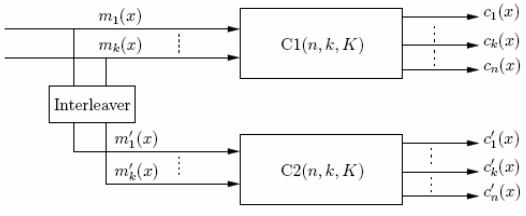

Figure 1: Turbo-code

2. CONVOLUTIONAL ENCODER

2.1 Principe and mathematical model

A convolutional code is an error-correcting code defined by three parameters (*n,k,K*), where *n* is the number of outputs, *k* is the number of inputs, and *K* is the constraint length of the code. Convolutional encoding can be modeled and implemented by shift-registers. Each block of *n* encoded bits at the output depends on *K* blocks of *k* information bits, which means that each of the *n* encoded bits at the output is a linear combination of the content of $k \times K$ shift-register cells. The encoder can be easily described by using its generator polynomial form given hereafter:

$$\begin{cases} c_1(x) = m_1(x).g_{1,1}(x) + \cdots + m_k(x).g_{k,1}(x) \\ \quad \vdots \qquad\qquad\qquad \vdots \\ c_n(x) = m_1(x).g_{1,n}(x) + \cdots + m_k(x).g_{k,n}(x) \end{cases} \quad (1)$$

where the information data binary stream is represented by k binary sequences $m_i(x)$ (with $i = 1,\ldots,k$). Moreover, the encoded binary data are represented by n sequences $c_j(x)$ (with $j = 1,\ldots,k$), and $g_{i,j}(x)$, in the general case, stand for the generator polynomial rational fraction associated to the input and output, i and j, respectively.

The generator polynomial rational fractions are all components of the code generator matrix, $\mathbf{G}(x)$, used to

rewrite equation (1) as follows:

$$\mathbf{c}(x) = \mathbf{m}(x).\mathbf{G}(x)$$

where $\mathbf{c}(x) = (c_1(x)\ldots c_n(x))$, $\mathbf{m}(x) = (m_1(x)\ldots m_k(x))$ and $\mathbf{G}(x)$ given as follow:

$$\mathbf{G}(x) = \begin{bmatrix} g_{1,1}(x) & \cdots & g_{1,k}(x) & \cdots & g_{1,n}(x) \\ \vdots & \ddots & \vdots & & \vdots \\ g_{k,1}(x) & \cdots & g_{k,k}(x) & \cdots & g_{k,n}(x) \end{bmatrix} \quad (2)$$

In the general case, $g_{i,j}(x)$ are polynomial rational fractions. But, as found in the NRNSC (No Recursive and No Systematic Code) version of a convolutional code, they can be only simple polynomials with no denominator polynomial, which means that there is no feedback part in the encoder. The other well known convolutional code configuration or convolutional code type is RSC (Recursive Systematic Code). The term "systematic" means that the k first output sequences ($c_1(x)$, … ,$c_k(x)$) are exact replicas of the input information sequences ($m_1(x)$, … ,$m_k(x)$); on the other hand, the term, "recursive", indicates that the output can be re-injected at the input side as the feedback part. It is worth noting that, according to only mathematical considerations, any convolutional code can be described by either an NRNSC form or an RSC one. This point will be detailed hereafter.

2.2 Equivalent encoder

"Equivalent" encoder or "equivalent" code means that the codewords, or the encoded sequences, generated by two equivalent encoders belong to the same codeword set. Let C_1 and C_2 be equivalent encoders, then the sequences $\mathbf{c}_1(x)$ and $\mathbf{c}_2(x)$ generated by the two encoders span the same subspace Θ, but usually they are different. In practice, the decoding of an encoded sequence, $\mathbf{c}(x)$ by two equivalent encoders, C_1 and C_2, in order to recover information data, usually leads to two decoded sequences, $\mathbf{m}_1(x)$ and $\mathbf{m}_2(x)$, where $\mathbf{m}_1(x) \neq \mathbf{m}_2(x)$. The encoded data obtained by encoding a binary data stream with C1 will allows one to identify one version of possible encoders belonging to the same equivalence class as C_1, but liable to be different from C_1. Even more, it is impossible to recover the true informative data.

This notion of equivalent encoder is of key-importance because it makes the problem of blind encoder identification non-trivial. Even when an equivalent encoder is identified, the obtained informative data are not the true ones. In practice, an equivalence class of convolutional encoders will always contain, at least, one NRNSC form and its equivalent encoder in RSC form.

2.3 Generator matrix of NRNSC encoder to RSC equivalent encoder

Given that each NRNSC encoder has its equivalent form RSC, and that it is essential to get the right encoder (used at the transmitter side), but not its equivalent, the issue is to transform the generator matrix, in NRNSC form, of a convolutional encoder into its RSC equivalent form

- NRNSC encoder of rate $1/n$ (i.e $k = 1$):

For such an NRNSC encoder, the RSC equivalent encoder generator matrix is easily obtained by dividing each generator polynomial of NRNSC encoder by the first one. Let us illustrate this transformation with the example of a C(2,1,3) encoder:

$$\mathbf{G}_{NRNSC}(x) = \begin{bmatrix} 1+x+x^2 & 1+x^2 \end{bmatrix}$$

The equivalent RSC encoder is expressed as follows:

$$\mathbf{G}_{RSC}(x) = \frac{\mathbf{G}_{NRNSC}(x)}{1+x+x^2} = \begin{bmatrix} 1 & \dfrac{1+x^2}{1+x+x^2} \end{bmatrix}$$

- NRNSC encoder of rate k/n:

In this case, where $k > 1$, it is more difficult to get the equivalent generator matrix. Let us assume that the generator matrix, $\mathbf{G}_{NRNSC}(x)$, is given by equation (2) and that the matrix composed of the k first groups of the generator polynomials given hereafter is denoted $\mathbf{Q}(x)$:

$$\mathbf{Q}(x) = \begin{bmatrix} g_{1,1}(x) & \cdots & g_{1,k}(x) \\ \vdots & \ddots & \vdots \\ g_{k,1}(x) & \cdots & g_{k,k}(x) \end{bmatrix}$$

Computing the inverse of matrix $\mathbf{Q}(x)$ leads to:

$$\mathbf{Q}^{-1}(x) = \frac{adj\ \mathbf{Q}(x)}{\det(\mathbf{Q}(x))}$$

and the generator matrix of the equivalent RSC form is given by:

$$\mathbf{G}_{RSC}(x) = \mathbf{Q}^{-1}(x).\mathbf{G}_{NRNSC}(x) \quad (3)$$

This matrix is composed of $\mathbf{I}_k$ the $k \times k$ identity matrix and of $k \times (n\text{-}k)$ polynomial rational fractions, such as:

$$\mathbf{G}_{RSC}(x) = \begin{bmatrix} 1 & & 0 & \dfrac{f_{1,k+1}(x)}{f_{1,1}(x)} & \cdots & \dfrac{f_{1,n}(x)}{f_{1,1}(x)} \\ & \ddots & & \vdots & & \vdots \\ 0 & & 1 & \dfrac{f_{k,k+1}(x)}{f_{1,1}(x)} & \cdots & \dfrac{f_{k,n}(x)}{f_{1,1}(x)} \end{bmatrix}$$

where $f_{1,1}(x) = \det(\mathbf{Q}(x))$ is called the feedback polynomial part, and the polynomials $f_{i,j}(x)$, $\forall\, i \in \{1,\ldots k\}$ and $\forall\, j \in \{k+1,\ldots n\}$ are the numerator polynomials of the non systematic outputs of the RSC encoder. Let us exemplify this transformation by detailing these calculations for an NRNSC C(2,3,3) encoder whose generator matrix is given by:

$$\mathbf{G}_{NRNSC}(x) = \begin{bmatrix} 1+x+x^2 & 1 & x^2 \\ x & 1+x^2 & 1+x+x^2 \end{bmatrix}$$

So, the matrix $\mathbf{Q}$(x), is $\mathbf{Q}(x) = \begin{bmatrix} 1+x+x^2 & 1 \\ x & 1+x^2 \end{bmatrix}$,

$$\text{and } \mathbf{Q}^{-1}(x) = \frac{1}{1+x^3+x^4}\begin{bmatrix} 1+x^2 & 1 \\ x & 1+x+x^2 \end{bmatrix}.$$

Using equation (3) to calculate $\mathbf{G}_{RSC}(x)$ leads to:

$$\mathbf{G}_{RSC}(x) = \begin{bmatrix} 1 & 0 & \dfrac{1+x+x^4}{1+x^3+x^4} \\ 0 & 1 & \dfrac{1+x^2+x^3+x^4}{1+x^3+x^4} \end{bmatrix}$$

3. CONVOLUTIONAL ENCODER

3.1 Blind estimation of the convolutional encoder parameters

It is now worth focusing on the blind estimation of convolutional encoder parameters in the non-cooperative context. The method proposed in [3] permits estimations of the interleaver period and of the code rate for a block code by taking profit from the redundancy introduced by the error-correcting code. A similar approach was used in [5] when the intercepted sequence was severely corrupted. From both methods, we developed an approach to be applied to the convolutional code in the case of a perfect transmission (no transmission error).

Its principle is to reshape columnwise the intercepted data bit stream under matrix form. This matrix, denoted $\mathbf{H}_i$, is computed for different values of i where i is the number of rows. For each matrix, the rank in the Galois Field GF(2) is computed. For $i \neq \alpha \cdot n$ with $\alpha \in \mathbb{N}$, $\mathbf{H}_i$ is a full rank matrix. But, for $l = \alpha \cdot n$, $\mathbf{H}_l$ is not a full rank matrix because some among the columns are linear combinations of the others induced by the redundancy introduced by the code. In this case, the rank of $\mathbf{H}_l$ is given by equation (4):

$$rank(\mathrm{H}_l) = l.\frac{k}{n} + k.(K-1),\ \forall l = \alpha.n,\ \alpha \in \mathbb{N} \qquad (4)$$

So, the encoder parameters, n, k and K, are identified by application of a linear regression method.

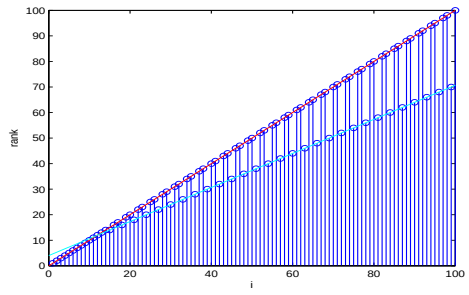

Figure 2: Estimation of the encoder parameters

Let us illustrate this approach with a C(3,2,3) code. The matrices, $\mathbf{H}_i$, are built, and then the rank is computed for increasing values of i. Figure 2 presents the obtained values of rank as a function of i. It is worth noting that: the rank is equal to i, except when i is a multiple of n and the method permits to estimate that n is equal to 3. Furthermore, when i is a multiple of n, the slope is equal to $2/3$, and it is easy to find that $k = 2$. Finally, $K = 3$ is estimated by using equation (4).

3.2 Blind estimation of the generator matrix

Now, it is possible to estimate the generator matrix of the code by applying a part of the methods developed in [1] and [4]. The method described in [1] is a general implementation of [4]. Both permit a global estimation of n, k, K and $\mathbf{G}$ in an iterative way. In these algorithms, the values of n, k and K are incremented at each iteration and, for each iteration, the $(n-k)$ systems of $(k+1)$ equations have to be solved. The fact that the rank computation of a matrix is easier than an iterative solving of linear-equation systems for every combination of n, k and K drove us to improve these methods by reducing the total number of equations to be solved. Once n, k and K have been estimated, there are only $(n-k)$ systems of $(k+1)$ equations to be solved to estimate $\mathbf{G}$. In fact, the part of the methods described in [1] and [4] and used, here, to estimate the generator matrix allows one to find the NRNSC equivalent form, $\mathbf{G}_{NRNSC}$, of the encoder really used. Thus, if the encoder is known to be in the RSC form, the algorithm proposed in subsection 2.3 can be used to get the generator matrix, $\mathbf{G}_{RSC}$.

4. BLIND RECOVERY OF TURBO-CODER

This section deals with the problem of the blind recovering of convolutional encoders used in Turbo-code. The past section described a method to recover the convolutional parameters and the generator matrix. On the other hand, its direct application to Turbo-coders is non-trivial because of the particular configuration of the convolutional encoders. In practice, the encoders are both in the RSC form and the encoder rate is $k/(k+1)$, which means that $n = k+1$; moreover, the systematic part of the second encoder is punctured. In such a specific configuration, the method described in section 3 is inapplicable to the second encoder because it requires data from, at least, two outputs for each encoder. For the first encoder, the blind estimation is achievable; if this encoder is in RSC form the encoded data, $\mathbf{c}(x)$, are decoded to get the information data, $\mathbf{m}(x)$. One should note that the blind determination of the encoder form is out of the scope of this paper. When the rate of the second encoder is k/n with $n > k+1$, its parameters as well as the generator matrix are both given by this method even with punctured k-systematic outputs, because after puncturing the outputs are, at least, $(n - k) \geq 2$. But, in the case where the rate of the second encoder is $k/(k+1)$ together with a punctured systematic part, it remains only one output and, thus, the above method is not applicable.

Let us now focus on the case where *1/n* the extension of our method to the general *k/(k+1)* case is still under study. It sounded us worth, at first, to consider all of the outputs of the Turbo-code. But, the outputs of the second and first encoder are not direct linear combinations of the same input since the output of the second one are obtained after interleaving of the information data. Figure 3 deals with the most critical case: 1/2 (*k = 1, n = 2*). Application of the rank method described in section 3.1 to this case gives the parameters, *n, K* and *le*. It ensues that there are (*n-1*) non systematic outputs, denoted **c**' (with **c**' = **c**'$_2$…**c**'$_n$) in vector form, as well as the vector of information data, **m,** for C2 recovery.

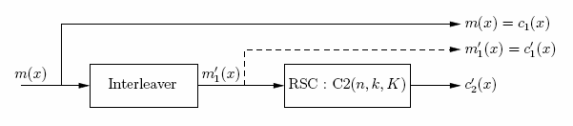

Figure 3: Special equivalent scheme of Turbo-code

The block interleaver can be modelled by a permutation matrix $\mathbf{E_0}$ of size, *le* × *le*. This permutation matrix operates on *le*-sized blocks, and the global interleaver can be represented by a square matrix, **E**:

$$\mathbf{E} = \begin{bmatrix} \mathbf{E_0} & & & \\ & \mathbf{E_0} & & \\ & & \ddots & \\ & & & \mathbf{E_0} \end{bmatrix}$$

The vector of all the interleaved data denoted **m**' is equal to **m**' = **m·E,** and with an RSC-type encoder the total output stream and the input stream for all $j = 2,\ldots,n$ are linked by equation (5):

$$\mathbf{c}'_j.\mathbf{G}_1 = \mathbf{m}.\mathbf{E}.\mathbf{G}_j \Rightarrow \mathbf{c}'_j.\mathbf{G}_1 + \mathbf{m}.\mathbf{E}.\mathbf{G}_j = 0 \qquad (5)$$

where $\mathbf{G}_i$ contains the binary coefficients of the ith generator polynomial (for all $i = 1,\ldots,n$) :

$$\mathbf{G}_i = \begin{bmatrix} g_i(K) & & & \\ g_i(K-1) & g_i(K) & & \\ \vdots & g_i(K-1) & \ddots & \\ g_i(1) & \vdots & \ddots & g_i(K) \\ & g_i(1) & & g_i(K-1) \\ & & \ddots & \vdots \\ & & & g_i(1) \end{bmatrix}$$

Since the interleaver operated by *le*-sized blocks, equation (5) can be rewritten as a vector form of size *le* on condition to not take into account the overlap-related problem. One should note that the interleaver is missing in the first part ($\mathbf{c'}_j.\mathbf{G}_1$) of equation (5). It ensues that the number of bits required for the determination of the first generator polynomial is only *K* instead of *le;* moreover, these bits have to be taken from every *le*-bit block. So, for each block of size *le,* this system can be rewritten in vector form as follows:

$$\underline{c}'_j.f_1 + \underline{m}.\mathbf{E_0}.f_j = 0$$

where the vector, $\underline{c'_j}$, is composed of the *K* first bits from each *le*-sized block of $\mathbf{c'}_j$, and $\underline{m}$ is composed of *le* bits of **m**. The vectors, f_1 and f_j, correspond to the *K* binary coefficients of the feedback generator polynomial and to the *le* bits of $\mathbf{G}_j$, respectively, so that:

$$f_1 = \left[g_1(K)\cdots g_1(1)\right]^T,\ f_j = \left[g_j(K)\cdots g_j(1)\ \ 0\cdots 0\right]^T$$

Now, it is easy to construct and to solve the (*n*-1) systems given hereafter to recover the generator polynomials

$$.\mathbf{C}_j.f = \underline{0} \quad \forall j = 2,\ldots,n$$

with the matrix $\mathbf{C}_j$ given by:

$$\mathbf{C}_j = \begin{bmatrix} c'_j(1) & \cdots & c'_j(K) & m(1) & \cdots & m(le) \\ c'_j(le+1) & \cdots & c'_j(le+K) & m(le+1) & \cdots & m(2.le) \\ \vdots & & \vdots & \vdots & & \vdots \end{bmatrix}$$

and the vector column f of size $(K+le)$: $f = \begin{bmatrix} f_1 \\ E_0.f_j \end{bmatrix}$

This method gives the first generator polynomial (f_1) and the interleaving version of the other generator polynomials. The complete blind estimation of the Turbo-code with recovery of the interleaver as well as the determination of encoder type are both out of the scope of this paper and will be described in the future. Once the interleaver has been estimated, the generator matrix of the encoder is easily obtained after de-interleaving.

5. CONCLUSION

This paper presented an approach developed to estimate convolutional codes in a non-cooperative context. This method allows a reduction of the complexity of previous algebraic approach introduced in [1] and [4]. Moreover, it leads to the generator matrix of an RSC encoder from its equivalent NRNSC. We also described a new approach aimed at estimating the second encoder of Turbo-code in the case of puncturing on its systematic part; it allowed us to get the interleaved version of these generator polynomials.

REFERENCES

[1] J. Barbier, "Reconstruction of turbo-code encoders", in Proc. SPIE Security and Defence Symposium, Vol. 5819, pp. 463-473, Mar. 2005.

[2] C. Berrou, A. Glavieux and T. Thitimajshima , " Near shannon limit error-correcting coding and decoding: turbo-codes", in Poc. ICC, pp 1064-1070, Geneva, Switzerland, May 1993.

[3] G. Burel and R. Gautier, "Blind estimation of encoder interleaver characteristics in a non-cooperative context", in Poc. IASTED-CIIT, Scottsdale, AZ, USA, Nov. 2003.

[4] E. Filiol, "Reconstruction of punctured convolutional encoder", in Proc. SITA, pp. 4-7, Hawaii, USA, Nov. 2000.

[5] G.. Sicot and S. Houcke, "Blind detection of interleaver parameters", in Proc. ICASSP, vol. 3, pp 829-832, Philadelphia, USA, Mar. 2005.