Blind Recovery of the Second Convolutional Encoder of a Turbo-Code when Its Systematic Outputs Are Punctured

Mélanie Marazin, Roland Gautier, Gilles Burel

MTA Review, Vol. XIX, No. 2, pp. 213-232, Jun. 2009 (ISSN 1843-3391)

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

Mélanie Marazin[1-2], Roland Gautier[1-2], Gilles Burel[1-2]

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 like in military interception or cognitive-radio applications. In this context, the intercepted data are the only known information. It also explains the residual indetermination due to "equivalent" coders and it gives the link between an NRNSC encoder and its RSC equivalent form. Furthermore, 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.**

Keywords – **Cognitive-Radio Applications, Turbo-Code, Convolutional Encoder, Interleaver, Blind Estimation, Non-Cooperative Communications, Interception.**

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. The information data are encoded by the first encoder and the interleaving version of this information data are encoded by the second one. In the most recent digital communication systems, such as the 3rd generation mobile system presented in [7] and [8], the systematic parts of the second convolutional encoder located after the interleaver are not transmitted to increase the Turbo-code rate.

Figure 2 presents the differences between cooperative and non-cooperative context. This figure shows that 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. But, in the case of military, spectrum surveillance or cognitive-radio applications, the parameters of the encoders and interleaver are unknown. Therefore, in

[1] Université Européenne de Bretagne, France.
[2] Université de Brest ; CNRS, UMR 3192 Lab-STICC, ISSTB, 6 avenue Victor Le Gorgeu, CS 93837, 29238 Brest cedex 3, France.
e-mail : melanie.marazin@univ-brest.fr, roland.gautier@univ-brest.fr, gilles.burel@univ-brest.fr

a non-cooperative context, the intercepted data are the only available information. In such situation, the encoders and interleaver parameters have to be blindly estimated, with no prior knowledge of the true ones used to encode the binary data stream on the transmitter side.

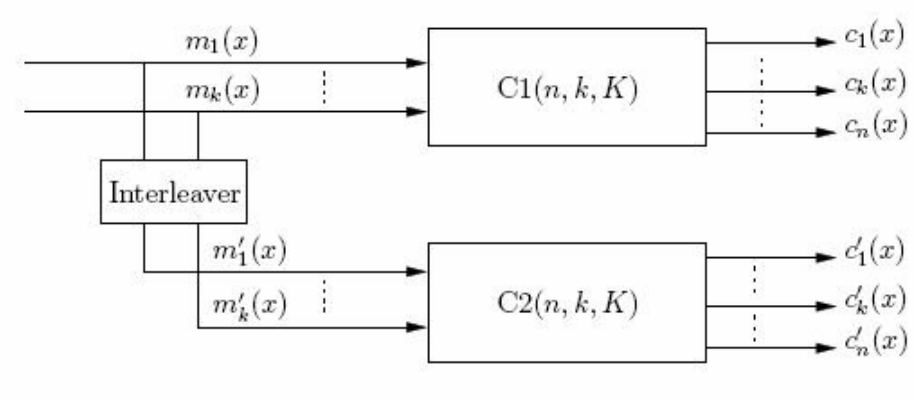

Figure 1: Turbo-code

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. At first, section 2 gives the principle of convolutional codes and introduces the concept of equivalent code, which is essential for encoder recovery. Furthermore, it presents the link between the NRNSC encoder and its equivalent form RSC, explained in [5]. Section 3 describes the approach we developed from the techniques proposed in [3] and [6] 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]. Section 4 reports the new approach on the blind estimation of the second encoder in the case of a classical implementation of the Turbo-code. Finally, section 5 shows an example of blind recovery of the Turbo-code convolutional encoders.

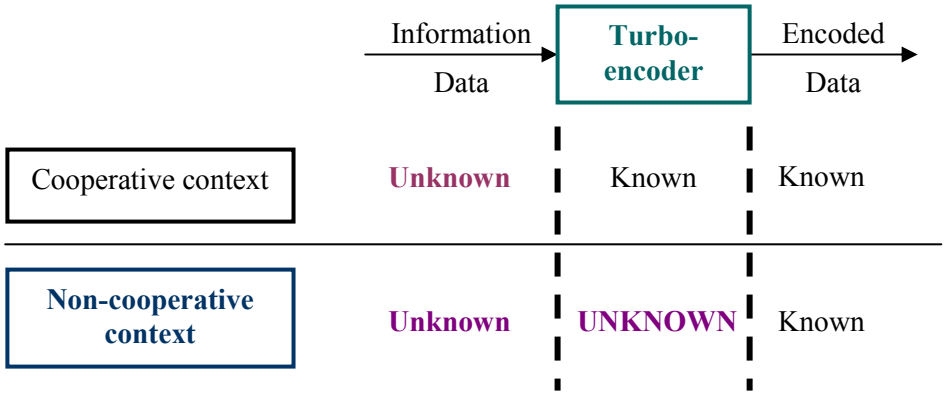

Figure 2: Cooperative versus non-cooperative context

2. Convolutional Encoder

2.1. Principle 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 (in the case without feedback) 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:

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

where the information data binary stream is represented by *k* binary sequences $m_i(x)$ (with $i = 1, \cdots, k$). Moreover, the encoded binary data are represented by *n* sequences $c_j(x)$ (with $j = 1, \cdots, n$), 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:

$$\mathbf{c}(x) = \mathbf{m}(x).\mathbf{G}(x) \tag{2}$$

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

$$\mathbf{G}(x) = \begin{bmatrix} g_{1,1}(x) & \cdots & g_{1,k}(x) & \cdots & g_{1,n}(x) \\ \vdots & \ddots & \vdots & & \vdots \\ g_{k,1}(x) & \cdots & g_{k,k}(x) & \cdots & g_{k,n}(x) \end{bmatrix} \tag{3}$$

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 the outputs are only direct linear combinations of the inputs, since there is no feedback part in the encoder. In this configuration, it is possible to rewrite equation (2) in matrix form. For that, consider the binary vectors, **m** and **c**, which are defined below:

- **m**: the vector of information data, with $\mathbf{m} = \left(\mathbf{m}^0, \mathbf{m}^1, \cdots\right) = \left(m_1^0, m_2^0, \cdots, m_k^0; m_1^1, m_2^1, \cdots, m_k^1; \cdots\right)$ (where $\mathbf{m}^t = \left(m_1^t, m_2^t, \cdots, m_k^t\right)$ represents the *k*-bit introduced at the time *t*);
- **c**: the vector of all encoded data, with $\mathbf{c} = \left(\mathbf{c}^0, \mathbf{c}^1, \cdots\right) = \left(c_1^0, c_2^0, \cdots, c_n^0; c_1^1, c_2^1, \cdots, c_n^1; \cdots\right)$ (where $\mathbf{c}^t = \left(c_1^t, c_2^t, \cdots, c_n^t\right)$ represents the *n*-bits encoded at the time *t*).

And, the encoding matrix, denoted **F**, is given below:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_K & & \\ & \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_K & \\ & & \ddots & \ddots & & \ddots \end{bmatrix} \tag{4}$$

where the sub-matrices $\mathbf{F}_j$ (with $j = 1, \cdots, K$) contains the j[th] binary coefficient of all generator polynomial:

$$\mathbf{F}_j = \begin{bmatrix} g_{1,1}(j) & g_{1,2}(j) & \cdots & g_{1,n}(j) \\ \vdots & \vdots & & \vdots \\ g_{k,1}(j) & g_{k,2}(j) & \cdots & g_{k,n}(j) \end{bmatrix} \tag{5}$$

Finally, the equation (2) rewritten under matrix form gives:

$$\mathbf{c} = \mathbf{m}.\mathbf{F} \tag{6}$$

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 $(c_1(x) \cdots c_k(x))$ are exact replicas of the input information sequences $(m_1(x) \cdots m_k(x))$; 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 $\mathbf{m}_1(x) \neq \mathbf{m}_2(x)$. Furthermore, if we consider the blind parameters identification problem, we can see that it may be a problem, because two equivalent encoders span the same subspace and if we have to identify the specific parameters of one encoder from encoded binary data, it will be mathematically impossible to distinguish two or more encoders in the same equivalence class excepted the case where the class contain only one element (which is not the case in practice). The encoded data obtained by encoding a binary data stream with C_1 will allow 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 information data.

It is possible to determine if two encoders belong to the same equivalence class. In fact, two encoders, C_1 and C_2, are equivalent if the cyclic versions of their coding matrix, defined

in equation (4), and their concatenation have the same rank. A cyclic version, denoted $\overline{\mathbf{F}}$, of period T (with $T>K$) given hereafter:

$$\overline{\mathbf{F}} = \begin{array}{c} \\ \mathrm{T} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 2.\mathrm{T}-1 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathrm{T} & \cdots & \cdots & \cdots & \cdots & \cdots & 2.\mathrm{T}-1 \end{array} \\ \begin{pmatrix} \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_K & & & \\ & \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_K & & \\ & & \ddots & & & \ddots & \\ & & & \ddots & & & \mathbf{F}_K \\ \mathbf{F}_K & & & & \ddots & & \vdots \\ \vdots & & & & & \ddots & \vdots \\ \mathbf{F}_2 & & & & & & \mathbf{F}_1 \end{pmatrix} \end{array} \tag{7}$$

For different value of T, two encoders, C_1 and C_2, are equivalent if:

$$rank\begin{bmatrix} \overline{\mathbf{F}}^{(1)} \\ \overline{\mathbf{F}}^{(2)} \end{bmatrix} = rank\left[\overline{\mathbf{F}}^{(1)}\right] = rank\left[\overline{\mathbf{F}}^{(2)}\right] \tag{8}$$

Let us illustrate this notion with a C(3,2,3) encoder. The generator matrix, noted $\mathbf{G}$, is composed by (2×3) generator polynomial represented in octal:

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{bmatrix}$$

The information data, denoted **m**, encoded by **G** gives the encoded data, noted **c**.

$$\mathbf{m} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{c} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

With the intercepted data, it is possible to obtain two generator matrices, denoted $\mathbf{G}_1$ and $\mathbf{G}_2$:

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{bmatrix} \text{ and } \mathbf{G}_2 = \begin{bmatrix} 6 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

Equation (7) and (8) are used to determine if these encoders belong to the same equivalence class. The cyclic version and the rank of these matrices are computed for different period (T):

$$T = 15, rank\begin{bmatrix} \overline{\mathbf{F}}^{(1)} \\ \overline{\mathbf{F}}^{(2)} \end{bmatrix} = rank\left[\overline{\mathbf{F}}^{(1)}\right] = rank\left[\overline{\mathbf{F}}^{(2)}\right] = 30$$

$$T = 35, rank\begin{bmatrix} \overline{\mathbf{F}}^{(1)} \\ \overline{\mathbf{F}}^{(2)} \end{bmatrix} = rank\left[\overline{\mathbf{F}}^{(1)}\right] = rank\left[\overline{\mathbf{F}}^{(2)}\right] = 70$$

$$T = 40, rank\begin{bmatrix} \overline{\mathbf{F}}^{(1)} \\ \overline{\mathbf{F}}^{(2)} \end{bmatrix} = rank\left[\overline{\mathbf{F}}^{(1)}\right] = rank\left[\overline{\mathbf{F}}^{(2)}\right] = 80$$

Thus, it is possible to deduce that C_1 and C_2 are equivalent but the decoding of the intercepted data, **c**, with the second encoder $(\mathbf{G}_2)$ gives the data $\mathbf{m}'$:

$$\mathbf{m}' = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$$

And, this data, $\mathbf{m}'$, are not the true information data:

$$\mathbf{m} \neq \mathbf{m}'$$
$$[1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0] \neq [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$$

Therefore, this method allows to determine if two encoders belong to the same equivalence class but it is impossible to distinguish the encoder used at the transmitter side. 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 information data are not the true ones.

2.3. Generator matrix of NRNSC encoder to RSC equivalent encoder

In practice, an equivalence class of convolutional encoders will always contain, at least, one NRNSC form and its equivalent encoder in RSC form. Given that each NRNSC encoder has its equivalent RSC form, 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 $(g_{1,1}(x))$. The equivalent RSC encoder is expressed as follows:

$$\mathbf{G}_{RSC}(x) = \frac{1}{g_{1,1}(x)} . \mathbf{G}_{NRNSC}(x) \tag{9}$$

Let us illustrate this transformation with the example of a C(2,1,3) encoder:

$$\mathbf{G}_{NRNSC}(x) = \left[1 + x^2 + x^3 \quad 1 + x + x^3\right]$$

and the equivalent RSC encoder given by equation (9):

$$\mathbf{G}_{RSC}(x) = \frac{\mathbf{G}_{NRNSC}(x)}{1 + x^2 + x^3} = \left[1 \quad \frac{1 + x + x^3}{1 + x^2 + x^3}\right]$$

- 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 (3) and that the matrix composed of the *k* first groups of the generator polynomials given hereafter is denoted $\mathbf{Q}(x)$:

$$\mathbf{Q}(x)=\begin{bmatrix} g_{1,1}(x) & \cdots & g_{1,k}(x) \\ \vdots & \ddots & \vdots \\ g_{k,1}(x) & \cdots & g_{k,k}(x) \end{bmatrix} \tag{10}$$

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

$$\mathbf{Q}^{-1}(x)=\frac{adj\ \mathbf{Q}(x)}{\det(\mathbf{Q}(x))} \tag{11}$$

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

$$\mathbf{G}_{RSC}(x)=\mathbf{Q}^{-1}(x).\mathbf{G}_{NRNSC}(x) \tag{12}$$

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

$$\mathbf{G}_{RSC}(x)=\begin{bmatrix} 1 & & 0 & \frac{f_{1,k+1}(x)}{f_{1,1}(x)} & \cdots & \frac{f_{1,n}(x)}{f_{1,1}(x)} \\ & \ddots & & \vdots & & \vdots \\ 0 & & 1 & \frac{f_{k,k+1}(x)}{f_{1,1}(x)} & \cdots & \frac{f_{k,n}(x)}{f_{1,1}(x)} \end{bmatrix} \tag{13}$$

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

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

So, the matrix $\mathbf{Q}$(x) is $\mathbf{Q}(x)=\begin{bmatrix} x & 1 \\ 1 & x^2 \end{bmatrix}$, and $\mathbf{Q}^{-1}(x)=\frac{1}{1+x^3}\begin{bmatrix} x^2 & 1 \\ 1 & x \end{bmatrix}$.

Using equation (12) to calculate $\mathbf{G}_{RSC}(x)$ leads to:

$$\mathbf{G}_{RSC}(x)=\begin{bmatrix} 1 & 0 & \frac{1+x+x^3}{1+x^3} \\ 0 & 1 & \frac{1+x^2+x^3}{1+x^3} \end{bmatrix}$$, where the feedback polynomial $f_{1,1}(x)=1+x^3$.

3. Blind recovery of 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 benefit from the redundancy introduced by the error-correcting code. A similar approach was used in [6] 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. This rank has two different performances:

- If i is not a multiple of the number of outputs ($i \neq \alpha.n$ with $\alpha \in \mathbb{N}$):

The columns of $\mathbf{H}_i$ are all independent therefore $\mathbf{H}_i$ is a full rank matrix.

- If i is a multiple of the number of outputs ($i = \alpha.n$ with $\alpha \in \mathbb{N}$):

$\mathbf{H}_i$ is not a full rank matrix because some among of the columns are linear combinations of the others induced by the redundancy introduced by the code. In this case, the rank of $\mathbf{H}_i$ is given by equation (14).

$$rank(\mathbf{H}_i) = i.\frac{k}{n} + k.(K-1),\ \forall i = \alpha.n,\ \alpha \in \mathbb{N} \tag{14}$$

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

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 3 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.
- The line joining all the points for $i = \alpha.n$ has a slope equal to $2/3$. Equation (14) allows to deduce that $k = 2$.
- Finally with the same line and equation (14), it is easy to estimate that $K = 3$.

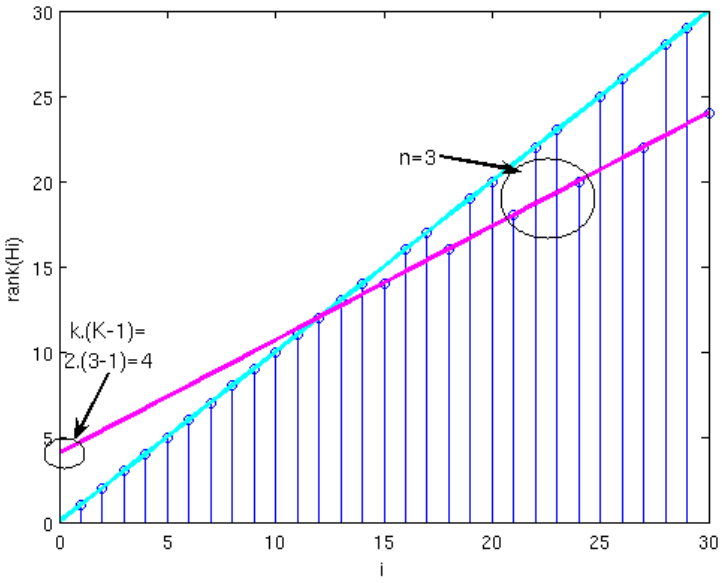

Figure 3: *rank*($\mathbf{H}_i$)=$f_{k,n,K}$ (i) =>
Estimation of the encoder parameters

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, presented in algo.1, 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.

In our method, presented in algo.2, the parameters n, k and K are obtained with the rank method and a part of the methods described in [1] and [4] is used to estimate the generator matrix. The fact that the rank computation of 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. In fact, once n, k and K have been estimated, there are only $(n-k)$ systems of $(k+1)$ equations to be solved to estimate $\mathbf{G}$. Moreover, to estimate the parameters of the encoder, two matrices $\mathbf{H}_i$ with rank deficiency are sufficient. In example presented in subsection 3.1, when i is equal to 18, it is possible to estimate the parameters (n, k and K) and to solve the system of 3 equations to estimate the generator matrix.

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, $\mathbf{G}_{RSC}$.

1- **for** $n = 2$ to $n_{\max}$
2- **for** $k = 1$ to $n-1$
3- **for** $K = 2$ to $K_{\max}$
4- $(n-k)$ systems of $(k+1)$ equations have to be solved $\Rightarrow$ $n, k, K, \mathbf{G}_{NRNSC}$
5- **end for**
6- **end for**
7- **end for**

Algo. 1: Method described in [1] and [4]

1- **for** $i = 2$ to $i_{\max}$
2- $rank(\mathbf{H}_i)$
3- **end for**
4- Estimation of n, k and K
5- $(n-k)$ systems of $(k+1)$ equations have to be solved $\Rightarrow$ $\mathbf{G}_{NRNSC}$

Algo. 2: New method

4. Method of blind recovery of Turbo-coder

4.1. Hypotheses

This section deals with the problem of the blind recovering of convolutional encoders used in Turbo-code. The previous section described a method to recover the convolutional parameters and the generator matrix. If the encoders of Turbo-code are both in the NRNSC form, it is possible to apply this method to estimate the parameters and the generator matrix of the convolutional encoders. On the contrary, if the encoders are both in the RSC form and the encoder rate is $k/(k+1)$, which means that $n = k+1$; its direct application to Turbo-coders is non-trivial because of the particular configuration of the convolutional encoders. In fact, in this case, the systematic part of the second encoder is punctured, which means that the k first outputs are not transmitted. The goal of this puncturing is to increase the Turbo-code rate. The encoder rate of C_1 and C_2 are denoted R_1 and R_2 and the Turbo-code rate given as follows:

$$R_{\text{turbo}} = \frac{R_1.R_2}{R_1 + R_2} \tag{15}$$

If the k-systematic outputs of the second encoder are punctured, the rate becomes:

$$R_{\text{turbo}} = \frac{R_1.R_2}{R_1 + R_2 - R_1.R_2} \tag{16}$$

The classical implementation of Turbo-code, (i.e. when each encoder is RSC form and systematic outputs of the second encoder are punctured) is represented in figure 4.

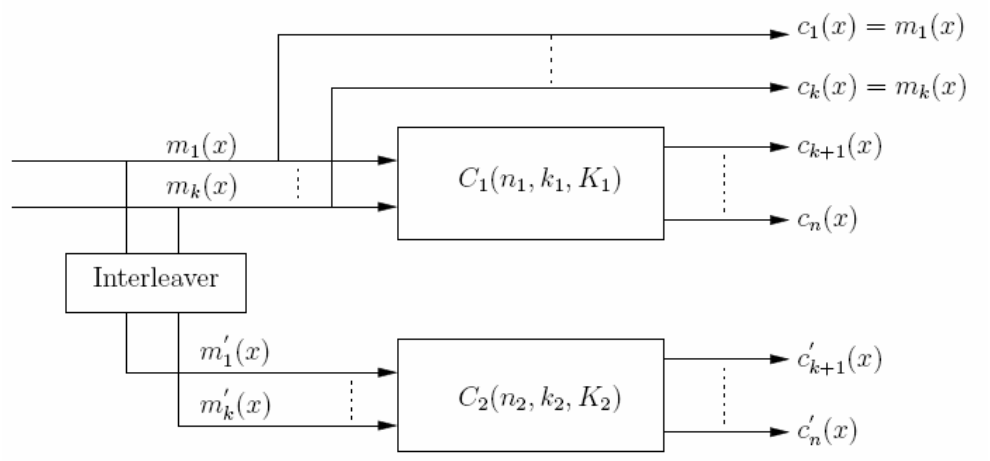

Figure 4: Turbo-code puncturing

In this specific configuration, for the first encoder, C_1, the blind estimation of parameters and generator matrix is achievable with the method described in section 3. Moreover, if the encoder form is known, 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.

For the second encoder, C_2, there are two configurations:

- The number of outputs n is greater than $k+1$:

In this case, the rate of the second encoder is k/n with $n > k+1$ and after puncturing of k-systematic part, the outputs are, at least, $(n-k) \geq 2$. Since this method used the redundancy introduce by the code, with only two outputs this redundancy is always present therefore it is possible to estimate the parameters and the generator matrix of this encoder.

- The number of outputs n is equal to $k+1$:

In this case where the rate of the second encoder is $k/(k+1)$ (i.e. $n = k+1$) with punctured systematic part, it remains only one output. In this configuration, the redundancy introduced by the code is not present. Consequently, it is impossible to apply directly the method described in section 3 to recover the second encoder.

Let us now focus on the case where the encoder rate is $1/n$, the extension of our method to the general $k/(k+1)$ case is still under study. The first idea to extend the method in this particular configuration must be to consider the systematic outputs of the first encoder and the outputs of the second encoder as being all of the outputs of this encoder. But, the outputs of the second $(c'_2(x) \cdots c'_n(x))$ and the first $(c_1(x))$ 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 5 deals with the most critical case, when the encoder rate is equal to $1/2$ $(k = 1, n = 2)$.

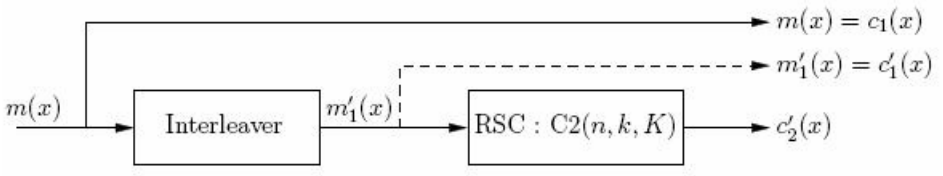

Figure 5: Special equivalent scheme of Turbo-code (k=1, n=2)

4.2. Blind estimation of the second encoder

Application of the rank method described in subsection 3.1 to this case gives the number of output (n), the constraint length (K) and the size of the interleaver (l_e). It ensues that there are $(n-1)$ non systematic outputs, denoted $\mathbf{c}'$ (with $\mathbf{c}' = \left(c_2' \cdots c_n'\right)$) in vector form, as well as the vector of information data, **m**, for C2 recovery.

The block interleaver can be modelled by a permutation matrix $\mathbf{E}_0$ of size $l_e \times l_e$. This permutation matrix operates on l_e-sized blocks, and the global interleaver can be represented by a square matrix, **E**:

$$\mathbf{E} = \begin{bmatrix} \mathbf{E_0} & & & \\ & \mathbf{E_0} & \ddots & \\ & & & \mathbf{E_0} \end{bmatrix} \tag{17}$$

The vector of all the interleaved data denoted $\mathbf{m}'$ is equal to $\mathbf{m}' = \mathbf{m}.\mathbf{E}$, and with an RSC-type encoder the total output stream and the input stream for all $j = 2, \cdots, n$ are linked by equation (18):

$$\mathbf{c}_j'.\mathbf{G}_1 = \mathbf{m}.\mathbf{E}.\mathbf{G}_j \Rightarrow \mathbf{c}_j'.\mathbf{G}_1 + \mathbf{m}.\mathbf{E}.\mathbf{G}_j = 0 \tag{18}$$

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

$$\mathbf{G}_i = \begin{bmatrix} g_i(K) & & & \\ g_i(K-1) & g_i(K) & & \\ \vdots & g_i(K-1) & \ddots & \\ g_i(1) & \vdots & \ddots & g_i(K) \\ & g_i(1) & & g_i(K-1) \\ & & \ddots & \vdots \\ & & & g_i(1) \end{bmatrix} \tag{19}$$

Since the interleaver operated on l_e-sized blocks, equation (18) can be rewritten as a vector form of size l_e on condition to not take into account the overlap-related problem. One should note that the interleaver is missing in the first part $\left(\mathbf{c}_j'.\mathbf{G}_1\right)$ of equation (18). It ensues that the number of bits required for the determination of the first generator polynomial is only K instead of l_e; moreover, these bits have to be taken from every l_e-bit block. So, for each block of size l_e, this system can be rewritten in vector form as follows:

$$\underline{c}'_j.f_1 + \underline{m}.\mathbf{E_0}.f_j = 0 \tag{20}$$

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

$$f_1 = \left[g_1(K) \cdots g_1(1) \right]^{\mathrm{T}}, \; f_j = \left[g_j(K) \cdots g_j(1) \;\; 0 \cdots 0 \right]^{\mathrm{T}} \tag{21}$$

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

$$\mathbf{C}_j . f = \underline{0} \quad \forall j = 2, ..., n \tag{22}$$

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

$$\mathbf{C}_j = \begin{bmatrix} c_j^{'}(1) & \cdots & c_j^{'}(K) & m(1) & \cdots & m(l_e) \\ c_j^{'}(l_e+1) & \cdots & c_j^{'}(l_e+K) & m(l_e+1) & \cdots & m(2.l_e) \\ \vdots & & \vdots & \vdots & & \vdots \end{bmatrix} \tag{23}$$

and the vector column f of size $(K + l_e)$:

$$f = \begin{bmatrix} f_1 \\ E_0 . f_j \end{bmatrix} \tag{24}$$

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 deinterleaving.

5. Example of blind recovery of Turbo-coder

5.1. Description of Turbo-code used

In this section, we can take an example of blind estimation of Turbo-code when the systematic output of the second encoder is punctured. Turbo-code presented in figure 6, is composed by two identical convolutional encoders, C(2,1,4), and one block interleaver of size l_e (with $l_e = 10$). These convolutional encoders are the ones used in the UMTS standard [7].

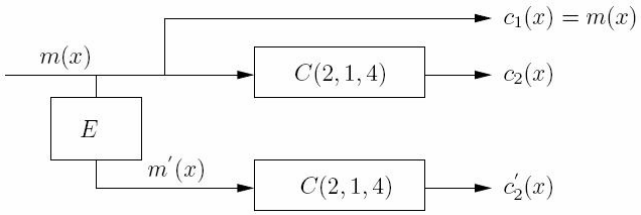

Figure 6: Turbo-code

The generator matrix is: $\mathbf{G}(x) = \begin{bmatrix} 1 & \dfrac{g_2(x)}{g_1(x)} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{1+x+x^3}{1+x^2+x^3} \end{bmatrix}$

The feedback polynomial: $g_1 = [1\,0\,1\,1]$ and the second generator polynomial: $g_2 = [1\,1\,0\,1]$.

The permutation matrix, $\mathbf{E}_0$, of size (10×10) representing one block of the interleaver is given graphically on figure 7 (where the element 0 is represented in white and the element 1 in black) and the global interleaver matrix **E** on figure 8. With equation (19) we can construct the matrix $\mathbf{G}_2$ shown on figure 9 and we can see on figure 10 the effect of the interleaver on the generator polynomial g_2, where the product $(\mathbf{E}.\mathbf{G}_2)$ is plotted.

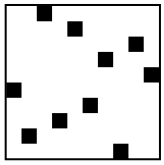

Figure 7: Graphical representation of E_0

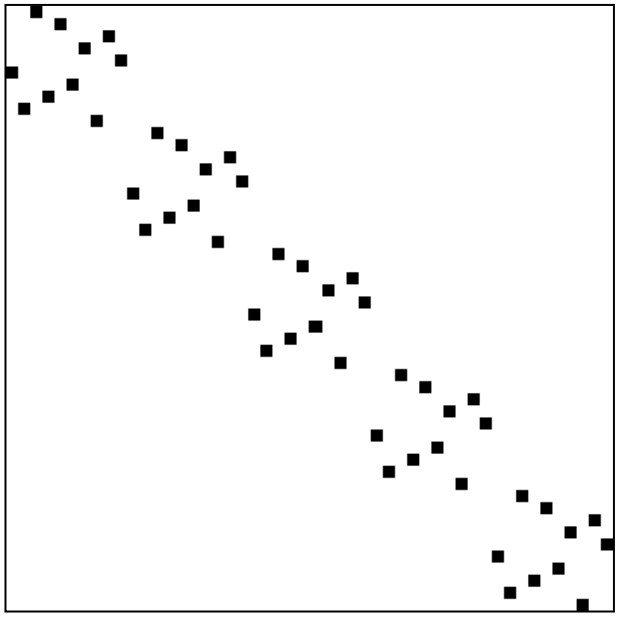

Figure 8: Graphical representation of E

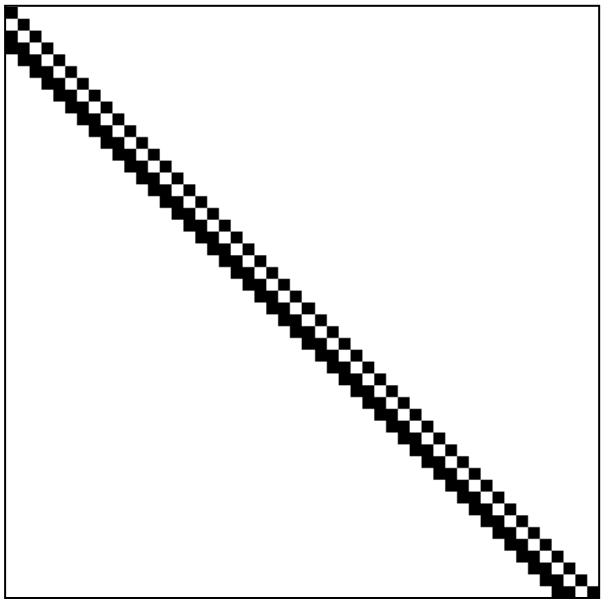

Figure 9: Graphical representation of G_2

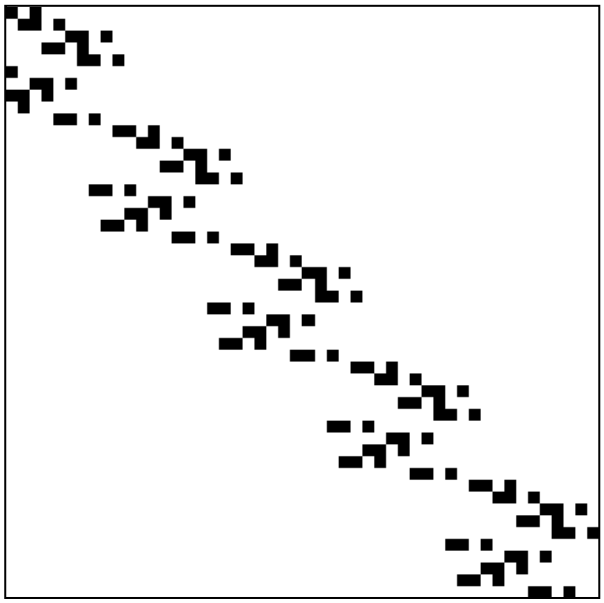

Figure 10: Graphical representation of the product $E.G_2$

5.2. Blind recovery of Turbo-code

For the first encoder C_1, the method explained in section 3 is used to estimate the parameters and the generator matrix of this code. The systematic output, $c_1(x) = m_1(x)$, of this encoder will be used to recover the second encoder.

In the first step, the method described in subsection 3.1 is used to blindly recover parameters of the encoder and the size of the interleaver. Figure 11 represents the values of $\mathbf{H}_i$ rank as a function of *i*. We can see on this figure, that the rank is equal to *i*, except when *i* is a multiple of $(n.l_e)$. It is possible to estimate that $(n.l_e)$ is equal to 20. Furthermore, according to the equation (14), when *i* is a multiple of $(n.l_e)$, the slope is equal to k/n. It is easy to determine that $k/n = 1/2$ and to deduce that $l_e = 10$. The same equation allows to estimate that $K = 4$.

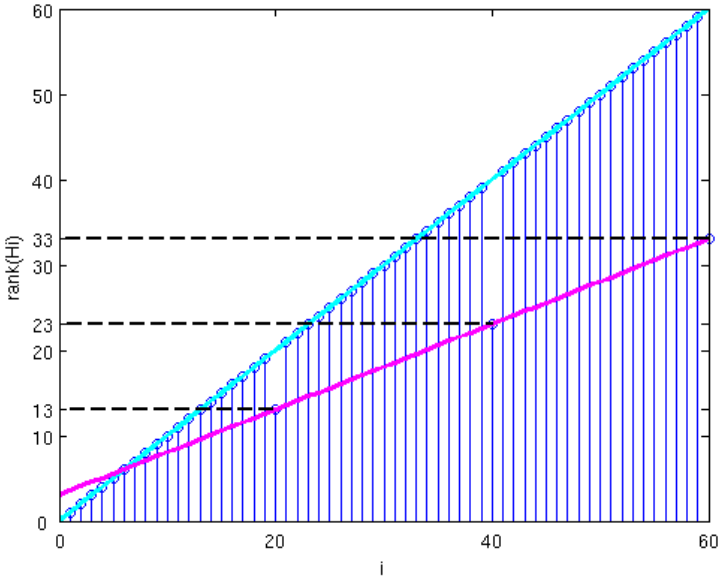

Figure 11: *rank*($\mathbf{H_i}$)=$\mathbf{f}_{k,n,K}$(i) => C(2,1,4)

In the second step, with the parameters of the encoder and the interleaver, the method describe in subsection 4.2 can be used to estimate the first generator polynomial and the interleaved version of the second generator polynomial. For that, the system presented in (22) has been constructed and resolved. With this system, the vector *f* obtained is:

$$f = [1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]^T$$

The *K*-first bits of this vector corresponds to the binary coefficient of g_1 and the other bits to the interleaving version of g_2, noted $f_2^{'}$.

$$f_1 = [1 \quad 1 \quad 0 \quad 1]^T \text{ and } f_2^{'} = \mathbf{E}_0 . f_2 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]^T$$

It is worth noting that the first column of matrix $\mathbf{G}_2^{'}$, represented on figure 10, corresponds to the vector $f_2^{'}$. Therefore, this method allows to obtain the good interleaved version of the second generator polynomial.

Estimation of the permutation matrix $\mathbf{E}_0$ is not described in this paper but once this matrix is obtained, it is easy to deinterleave the vector $f_2^{'}$. For that, the inverse of the permutation matrix is computed and is given graphically on figure 13. The global deinterleaver, matrix $\mathbf{E}^{-1}$, is shown on figure 14.

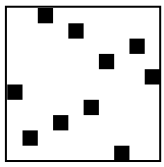

Figure 12: Graphical representation of $\mathbf{E_0}$

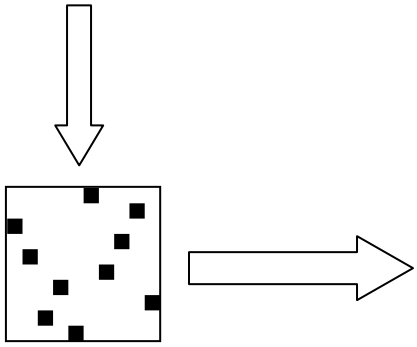

Figure 13: Graphical representation of $\mathbf{E_0^{-1}}$

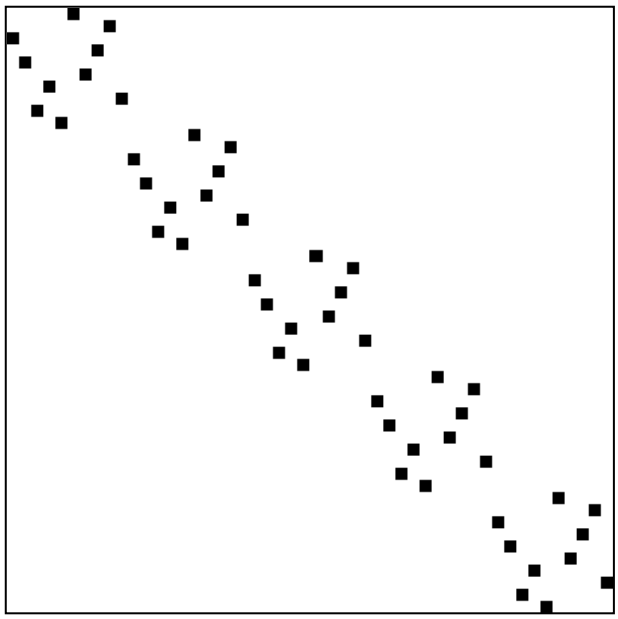

Figure 14: Graphical representation of $\mathbf{E^{-1}}$

The link between the interleaved version of f_2 and this deinterleaved version is given by:

$$f_2 = \mathbf{E}_0^{-1}.f_2^{'} \tag{25}$$

This equation is used to deinterleaving vector $f_2^{'}$:

$$\begin{aligned}\mathbf{E}_0^{-1}.f_2^{'} &= \mathbf{E}_0^{-1}.\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0\end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix}1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix}f_2 & zeros(1, le-K)\end{bmatrix}^{\mathrm{T}}\end{aligned}$$

and the second generator polynomial f_2 is obtained. It is also possible to deinterleave the global matrix $\mathbf{G}_2$:

$$\mathbf{G}_2 = \mathbf{E}^{-1}.\mathbf{G}_2^{'} \tag{26}$$

Finally, if the permutation matrix is recovered, this method permits to estimate all parameters of a Turbo-code when the systematic output of the second encoder is punctured.

Conclusion

This paper presented an approach developed to estimate convolutional codes in a non-cooperative context which allows a reduction of the computation cost compared to previous algebraic approaches introduced in [1] and [4]. Moreover, it introduced the notion of the equivalence class and the necessity to estimate the true encoder and not its equivalent version. In this objective, it gives the link between the generator matrix of an NRNSC encoder and its RSC equivalent form.

Finally, by using these methods, it described a new approach aimed at estimating the second encoder of a Turbo-code when its systematic outputs are punctured. In this specific configuration, used in the most recent digital communication systems, it allowed us to get the interleaved version of these generator polynomials.

These different approaches provide essential parameters in the context of spectrum surveillance and cognitive-radio applications as well as in the purpose of building a self-recovering receiver.

Acknowledgment

This work was supported by the Brittany Region (France).

References

[1] J. Barbier, "*Reconstruction of turbo-code encoders*", in Proc. SPIE Security and Defence, Space Communication Technologies Symposium, vol. 5819, pp. 463-473, March 2005.

[2] C. Berrou, A. Glavieux and T. Thitimajshima , " *Near shannon limit error-correcting coding and decoding: turbo-codes*", in Poc. IEEE International Conference on Communications, 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 Proc. IASTED International Conference on Communications, Internet and Information Technology, pp 275-280, Scottsdale, AZ, USA, November 2003.

[4] E. Filiol, "*Reconstruction of punctured convolutional encoder*", in Proc. 2000 International Symposium on Information Theory and Applications, pp. 4-7, Hawaii, USA, Nov. 2000.

[5] R. Johannesson and K. Sh. Zigangirov, "*Fundamentals of convolutional coding*", IEEE Series on Digital and Mobile Communication, IEEE Press, 1999.

[6] G. Sicot and S. Houcke, "*Blind detection of interleaver parameters*", in Proc. IEEE International Conference on Accoustics, Speech, and Signal Processing, vol. 3, pp. 829-832, Philadelphia, USA, Mar. 2005.

[7] 3GPP TS 25.212 V6.5.0: "*3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD) (Release 6)*", June 2005.

[8] TIA/EIA/IS – 2000.2-A-2; "*Physical layer Standard for CDMA2000 Spread Spectrum Systems*" (Revision), April 2002.