Journal of Theoretical and Applied Information Technology
31st December 2022. Vol.100. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A STUDY OF NON-BINARY PRODUCT CODES AND THEIR TURBO DECODING

ZAKARIA M'RABET*, FOUAD AYOUB+, MOSTAFA BELKASMI*

*ICES Team, ENSIAS, Mohammed V University in Rabat, Morocco
+ERIATC Team, LaREAMI Lab, CRMEF-K, Kenitra, Morocco
E-mail: zakaria.mrabet@um5s.net.ma

ABSTRACT

In this paper we study a family of non-binary product codes (nb-PC) constructed from the family of low density parity check (LDPC) codes known as Euclidean geometry (EG) codes. A coding/decoding scheme of these non-binary product codes (nb-PC) is presented where the turbo decoder is based on a soft input soft output (SISO) decoder suitable for non- binary one-step majority-logic decodable (OSMLD) codes, which is described in details in this work. We study the effect of iterations on the performance of our (PC) decoder, and exhibit the symbol and block error rates of some constructed codes transmitted over Additive Gaussian Noise (AWGN) channel. The obtained results are satisfying.

Keywords– *Non-Binary, Product Codes, Turbo Decoding, One-Step Majority-Logic Decodable (OSMLD) Codes, Low Density Parity Check (LDPC) Codes.*

1. INTRODUCTION

Shannon showed in [1] that random codes with long block length are more likely to approach the channel capacity. Unfortunately, increasing the code length results in an increase of the decoding complexity. One approach for designing long and powerful codes from short length component codes, is to use a coding technique called product coding. This coding technique was invented in 1954 by Elias in [2] and it consists on combining two or more codes to generate a code with longer block size.

This work is a study of the non-binary Product Codes, Their Construction and Decoding. This method is not well documented for the non-binary case [3], but it results in a variety of algebraically constructed large codes with known parameters, which is a strong point in code construction. Many constructed methods of LDPC codes focus on constructing the parity check matrix, and the code associated with it would be the null space of this matrix, which doesn't give a clue about the dimension or the minimum distance of the code, which make the coding process challenging. Therefore our approach of constructing large codes with known parameters is a strong asset.

Since the introduction of turbo codes by Berrou [4] for convolutionnal codes over two decades ago, there has been much interest in the conception of design of soft-input soft-output (SISO) decoding algorithms. Soon after, turbo decoding was applied to block codes by Pyndiah [5] and others. The decoding of a product code can be assimilated to a turbo decoder of serial concatenated block codes. However, for concatenated schemes with block component codes, the computational complexity of trellis-based SISO decoding algorithms is often high. This has led to the research for new SISO decoding algorithms with low complexity and high performance, e.g., [5–7]. These algorithms calculate extrinsic information using classical decoders such as the Chase algorithm in [5], ordered statistics decoding algorithm in [7] and the Hartmann/Rudolph algorithm in [6]. In this perspective Belkasmi et al. [10] presented a new iterative decoding algorithm based on a SISO extension form of the Massey algorithm [9]. Svirid [11] also used the Threshold algorithm in iterative decoding but for convolutional codes. Belkasmi's iterative decoding process follows that given by Pyndiah in [5]. However, instead of using an extension of the Chase algorithm on BCH codes, he applied an extension of Massey's algorithm on one step majority logic decodable (OSMLD) codes. Another known work is attributed to Lucas et al. [6], who introduced an iterative decoding algorithm for several families of codes including OSMLD codes but they used an approximation of Hartmann/Rudolph algorithm [12].

On the other hand, product codes can be constructed as binary codes or as non-binary ones.

Binary product codes can be constructed by using binary component codes, for example by using BoseChaudhuri-Hocquenghem (BCH) codes or Difference Set Cyclic (DSC) codes as their component codes. In the same logic, non-binary product codes can be constructed by combining non-binary codes with each other, for example Reed-Solomon (RS) codes or non-binary finite geometry codes like Euclidean and Projective geometry codes. Unfortunately non-binary turbo decoding is not well investigated.

Since, constructing a code without presenting its decoding is incomplete, and because a major issue of the decoding algorithms of non-binary LDPC codes is that they are very complex and are unsuitable for practical implementation, our contributions are to propose an iterative decoding scheme for non-binary product codes that involves non-binary Euclidean Geometry codes as component codes, and a non-binary version of the Iterative Threshold Decoder of Belkasmi et. al as elementary soft-input soft-output decoder which is based on Majority Logic Decoding (MLGD).

The rest of the paper is organized as follows; Section II is dedicated to the presentation of non-binary one-step majority-logic decodable (OSMLD) codes and the description of the majority-logic decoding (MLGD) algorithm. In Section III, Soft-input Softoutput (SISO) decoding algorithm is detailed along with its iterative version, which is the first time this decoder is presented. Section IV is for product code construction and iterative turbo decoder is presented. The simulation results are exhibited in Section V. Finally Section VI concludes the paper.

2. BINARY AND NON-BINARY ERROR CORRECTING CODES

Error correcting codes are used to reduce the errors in a received sequence of symbols that was transmitted through a noisy channel. In the non-binary codes, the transmitted symbols belong to a Galois Field. In this section we describe the Galois field elements, how we can obtain them, and why each symbol in a Galois field has a binary representation. Then we will present cyclic OSMLD codes, and how EG codes are a good choice.

2.1 Non-Binary Cyclic EG OSMLD LDPC Codes

The elements of a non-binary code can either belong to a finite ring of integers, or to a finite Galois field, in this study we investigate only the latter family. Galois Fields are just regular fields that contain a finite number of elements, denoted q which is a power of a prime. Such Galois Fields are denoted $GF(q)$. Then elements of $GF(q)$ can be derived from its primitive element α, which is the root of the primitive polynomial characterizing the field $GF(q)$. Let α be the primitive element of a $GF(q)$, where $q = 2^m$ and m is a positive integer. Then $GF(q) = \{\alpha^{-\infty} = 0, \alpha^0 = 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$, and each element α_s has a binary representation on m bits, with $s = 0,1,\ldots,q-1$. Therefore $\alpha_s \in GF(q)$ means $\alpha_s = (a_{s,0}, a_{s,1}, \ldots, a_{s,m-1})$ is its binary representation.

2.2 1st Contribution : Non-Binary Extrinsic MLGD

Consider a non-binary (n,k) cyclic code C with parity check matrix H, and symbols belonging to $GF(q)$. The row space of H is an $(n-k,n)$ cyclic code, denoted by $C^{\perp}$ or C_d, which is the dual code of C, or also called the null space of C. For any vector $\underline{u} \in C$ in and $\underline{w} \in C^{\perp}$, the inner product of $\underline{u}$ and $\underline{w}$ is $\alpha^{-\infty}$ $GF(q)$, that is,

$$\underline{u}.\underline{w} = u_1w_1 + u_2w_2 + \cdots + u_nw_n = 0 \quad (1)$$

In fact, an n-tuple $\underline{u}$ is a code-word in C means that $\underline{w}$ for any vector $\underline{w}$ of the dual code $C^{\perp}$, $\underline{u} \cdot \underline{w} = 0$. The equality (1) is called a *parity check* equation. It is clear that there are 2^{n-k} such parity check equations.

Suppose that $\underline{z}$ is a hard received vector. For any vector $\underline{w}$ in the dual code $C^{\perp}$, we can form the following linear sum of the received symbols:

$$A = (\underline{z}.\underline{w}).y_n^{-1} = (z_1w_1 + z_2w_2 + \cdots + z_nw_n).w_n^{-1} \quad (2)$$

which is called the *parity check sum* or simply *check sum*. This parity check A must be equal to zero if the received vector $\underline{z}$ is a code-word in C, however, if $\underline{z}$ is not a code-word, then A may not be equal to zero.

A received symbol z_j is said to be *checked* by the check sum A if the coefficient w_j is non-zero.

We can also estimate the received symbol z_n checked by a check sum B, by calculating the following sum:

$$B = (z_1w_1 + z_2w_2 + \cdots + z_{n-1}w_{n-1}).w_n^{-1} \quad (3)$$

There are some codes that can be decoded by taking a majority logic vote from a set of check equations orthogonal on each data symbol. When the code is cyclic, we need just one set of J equations orthogonal on the last symbol, then we cycle the data

symbols so that the orthogonality becomes on the $(n-1)^{th}$ position, and so on until estimating all the n data symbols. The first Majority Logic Decoding algorithm was devised by Reed in 1954 [4] for Reed-Muller codes.

Table 1 shows some examples of non-binary cyclic OSMLD codes. In this table we use the abbreviations EG for Euclidean Geometry codes, PG for Projective Geometry codes and RS for Reed-Solomon codes. These codes can serve to construct a product code as we will discuss in later sections.

Table 1: Set Of Non-Binary Cyclic OSMLD Codes.

code	*GF* (*q*)	*n*	*k*	R	*J*	*dmin*
PG	*GF* (8)	7	3	0.42	3	4
EG	*GF* (16)	15	7	0.46	4	5
EG	*GF* (4)	15	7	0.46	4	5
PG	*GF* (64)	21	11	0.52	5	6
EG	*GF* (8)	63	37	0.58	8	9
RS	*GF* (256)	255	239	0.93		17
EG	*GF* (16)	255	175	0.68	16	17
RS	*GF* (256)	255	153	0.6		103
EG	*GF* (256)	255	175	0.68	16	17
RS	*GF* (256)	255	223	0.87		33

In the following, we show that certain properly formed check sums can be used for estimating the received symbols in z_j.

Suppose that there exist J vectors in the dual code $C^{\perp}$,

$$w_1 = (w_1,1,w_1,2,\ldots,w_{1,n})$$
$$w_2 = (w_2,1,w_2,2,\ldots,w_{2,n})$$
$$\ldots$$
$$w_J = (w_J,1,w_J,2,\ldots,w_{J,n})$$

which have the following properties:

1. The n^{th} component of each vector is " 6= $\alpha^{-\infty}$, 0", that is , $w_{1,n} = w_{2,n} = \ldots = w_{J,n}$ 6= 0.

2. For i 6= n, there is *at most* one vector whose i^{th}component is " 6= 0"; for example, if $w_{1,i}$ 6= 0, then $w2,i = w3,i = \cdots = wJ,i = 0$.

These J vectors are said to be orthogonal on the n^{th} symbol position. We call them *orthogonal vectors* or *orthogonal equations*.

Now let us form J check sums from this orthogonal vector using (3). We can form the relationship between these J vectors and the received symbols in the following manner:

$$\begin{cases} B_1 = (z_1 w_{1,1} + z_2 w_{1,2} + \cdots + z_{n-1} w_{1,n-1}) . w_{1,n}^{-1} \\ B_2 = (z_1 w_{2,1} + z_2 w_{2,2} + \cdots + z_{n-1} w_{2,n-1}) . w_{2,n}^{-1} \\ \vdots \\ B_J = (z_1 w_{J,1} + z_2 w_{J,2} + \cdots + z_{n-1} w_{J,n-1}) . w_{J,n}^{-1} \end{cases}$$

Therefore, we see that the symbol z_n is checked by all the check sums above. Because of the second property of the J orthogonal vectors, any received symbol other than z_n is checked by at most one check sum. These J check sums are said to be *orthogonal on the symbol* z_n and they are crucial for the Majority Logic (MLG) based decoding algorithms. In fact, this structural property is crucial to deliver extrinsic information about the symbol z_n.

Let d_{min} be the minimum distance of the code. Clearly, the threshold decoding is more effective for codes when it is equal to or close to the error correcting capability $t = (d_{min} - 1)/2$ of the code; in other words, J should be equal to or close to $d_{min} - 1$. For the cyclic Euclidean geometry codes $d_{min} = J + 1$, thus this family of codes are best suited for MLGD algorithms, and the bigger the set of equations is, the more extrinsic information it will gather, therefore the more powerful the decoding performance will be.

Algorithm 1 describes the non-binary Majority Logic Decoding Algorithm nb-MLGD.

The performance of this decoder is displayed in the Figure 1 for the nb-EG(255,175,17) over GF(16).

Algorithm 1 nb-MLGD

Input: Received signal y
Output: Non-binary Decoded sequence z
1: take the hard decision vector z from y
2: for $j = (1 : n)$ do
3: for $i = (0 : J)$ do
4: B_i
5: end for
6: z_n is the Majority of the B_i
7: operate a cyclic shift to the vector z
8: end for

Journal of Theoretical and Applied Information Technology
31st December 2022. Vol.100. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

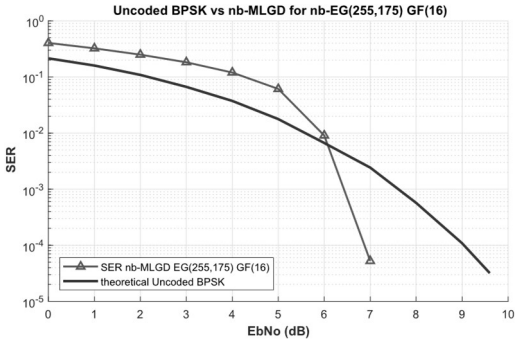

Figure 1: Uncoded BPSK vs nb-MLGD for nb-EG(255,175,17) over GF(16).

We observe from Figure 1 that by using the nb MLGD, in late SNR we have nearly 2dB of coding gain compared to the uncoded BPSK modulation. In the next section, we present the straightforward amelioration of this decoder, which involves the information delivered by the channel.

3. FROM SIHO TO SISO DECODER FOR NONBINARY CODES

Reed's algorithm was improved by Massay, who put the first unified formulation of the MLGD named Threshold Decoding (TD) [9]. In his original work he considered two different variations of the decoding algorithm. Consider here the method which involves the computations of the B_i equations instead of the A_i equations. Since the B_i's are obtained from the A_i's by removing the symbol on which the equations are orthogonal, the information delivered by the B_i's will be extrinsic to that symbol.

3.1 Non-Binary SIHO Threshold Decoder

In this section, we will describe the steps of the Threshold Decoder for non-binary codes.

Now suppose that a code-word $\underline{x}$ in C is transmitted. Let $y = (y_1,y_2,...,y_n)$ be the real received vector, we can compute the reliability of the received symbols being symbols from $GF(q)$ as follows:

We can compute llr's with (4) to work only with positive values.

$$llr_j(\alpha_s) = ln\frac{P(x_j=\alpha_s|y_j)}{\min\limits_{\theta\in GF(2^m)}\{P(x_j=\theta|y_j)\}} \quad (4)$$

Therefore we compute 2^s values of $llr_j(\alpha_s)$ for each received symbol y_j being $\alpha_s \in GF(q)$.

Let $\underline{z} = (z_1,z_2,...,z_n)$ be the hard decision vector, computed from the channel reliability:

$$z_j = \arg\max_{\alpha s\in GF(q)0\le s\le q-1} llr_{s,j}(5)$$

Let the magnitude of the reliability of the hard decision symbol be the maximum value :

$$r_j = \max_{0\le s\le q-1} llr_j(\alpha_s) \quad (6)$$

Let $N(i) = \{j/w_{i,j}\neq 0\}$, be the set of indices of non-zero coefficients of w_i involved in the parity check sum B_i. Therefore, the weighting coefficient of the i^{th} equation can be computed in (7):

$$W_i = \log\left(\frac{1+\prod\limits_{p\in N(i)\setminus n}\tanh\left(\frac{r_p}{2}\right)}{1-\prod\limits_{p\in N(i)\setminus n}\tanh\left(\frac{r_p}{2}\right)}\right) \quad (7)$$

Let W_0 be:

$$W_0 = r_j \quad (8)$$

Let $X_{s,j}$ be the extrinsic reliability of the j^{th} received symbol being α_s. X is initialized to zero, then it is computed as follows:

$$XBi,j = XBi,j + Wi \quad (9)$$

Then we add the reliability of the symbol z_j on which the orthogonality holds as follows:

$$Xzj,j = Xzj,j + rj \quad (10)$$

Then the final reliability of the decoder is calculated from the channel reliability and the extrinsic information computed throuth the orthogonal weighted equations as in (10)

$$LLR = llr + X \quad (11)$$

$$z_j = \arg\max_{\alpha s\in GF(q)0\le s\le q-1} LLR_{s,j} \quad (12)$$

The hard vector $\underline{z}$ is then cycled n times to compute LLR for every position j.

Algorithm 2 Describes The Non-Binary Threshold Decoding.

Algorithm 2 nb-TD

Input: Received signal y
Output: Non-binary Decoded sequence z for $j = (1 : n)$ do
2: for $s = (0 : q-1)$ do
Calculate $llr_j(\alpha_s)$ from (4)
4: r_j from (5) z_j from (6)
6: end for end for
8: for $j = (n : 1)$ do for $i = (0 : J)$ do
10: compute B_i
compute W_i from (7) and (8)
12: end for
compute X_j from (9) and (10) 14: compute LLR_j from (11)
Calculate the hard decision z_j of LLR_j from (12)
16: end for

Journal of Theoretical and Applied Information Technology
31st December 2022. Vol.100. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To summarize, the non-binary Threshold Decoding algorithm is the logical extension of the Majority Logic Decoding algorithm. It enhances the performance of this latter by taking into consideration the channel output before computing the orthogonal equations, which assigns weights to each one of these equations, and this leads to more accurate results. In the next section we will present for the iterative version of this threshold decoder, which will represent the non-binary version of the binary ITD devised by Belkassmi et-al in [10]. In Figure 2 we notice the coding gain obtained by the nb-TD vs the nb-MLGD.

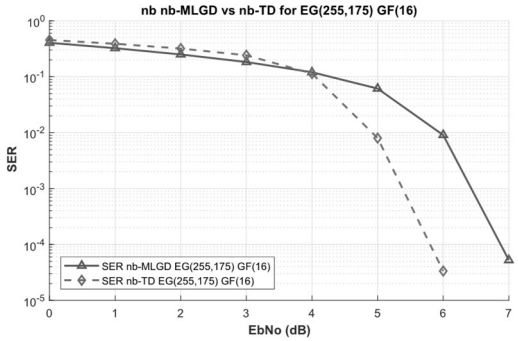

Figure 2: nb-TD vs nb-MLGD for EG codes over GF(16).

3.2 Non-Binary SISO Iterative Threshold Decoder

In [6], Lucas et al. proposed the first iterative decoding algorithm for OSMLD codes. His algorithm computes the extrinsic information using a weighted version of the Hartmann/Rudolph algorithm [12]. In [6], Lucas new algorithm showed that OSMLD codes can perform better than their LDPC counterparts, for moderate code lengths (between 200-1000). In [5], Pyndiah proposed an iterative decoding scheme that updates only the extrinsic information over the iterations, while keeping the same channel reliabilities of the received symbols during each iteration. This principle was also used by Belkasmi et al. in [10], where they have modified Pyndiah's algorithm by associating with each decision a reliability that is weighted by a fixed parameter λ_2, and which will be exploited during the iterations.

$$LLR(t+1) = \lambda_1 llr(t) + \lambda_2 X(t) \quad (13)$$ then

compute the most likely hard decision by this equation (14):

$$z_j^{(t+1)} = \arg \max_{\alpha_s \in GF(q)} {}_{0 \le s \le q-1} LLR_{s,j}^{(t+1)} \quad (14)$$

To jump from soft output to iterative, we just need to reinject the extrinsic information generated by the decoder in an iteration as *a-priori* information for the decoder in the next iteration.

Algorithm 3 describes our proposition of the nonbinary Iterative Threshold Decoding.

Performance of the non-binary Iterative Threshold decoding algorithm is displayed in the Figure 3 in terms of symbol error rate, for the code $EG(255,175,17)$ over $GF\ 2^4)$. Clearly, the nb-ITD outperforms the other algorithms, where a coding gain of 1.5dB is achieved compared to the non-binary Threshold Decoding algorithm.

Algorithm 3 nb-iTD

Input: Received signal y
Output: Binary Decoded sequence z for j = $(1 : n)$ do
for $s = (0 : q-1)$ do
3: Calculate $llr_j(\alpha_s)$ from (4) r_j from (5) z_j from (6)
6: end for end for
for $j = (n : 1)$ do
9: for $i = (0 : J)$ do compute B_i compute W_i from (7) and (8)
12: end for
compute X_i from (10) and (11) compute the overall reliability LLR_j from (13)
15: Calculate the hard decision z_j of LLR_j from (14) operate a cyclic shift to the vectors z and r
end for

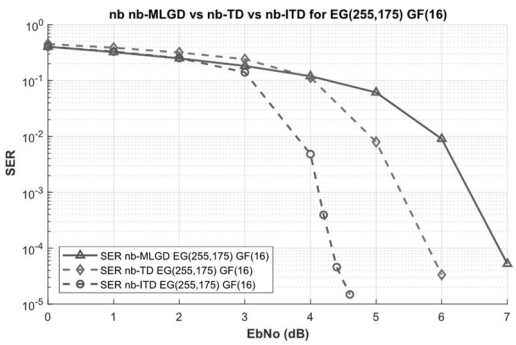

Figure 3: nb-SISO vs nb-SIHO vs nb-HIHO.

4. TURBO DECODING OF NON-BINARY PRODUCT CODES

The decoding of product codes consists of successively decoding lines and columns of a

received matrix, using an elementary soft decoder. Among the first authors that used this principle of

Table 2: Set of non-binary Product Code based on OSMLD codes

code	*GF* (*q*)	(*n,k,d*$_{min}$)	R	(*n,k,dmin*)2	Binary expantion	R'
PG	*GF* (8)	(7,3,4)	0.42	(49,9,16)	(147,27,48)	0.18
EG	*GF* (16)	(15,7,5)	0.46	(255,49,25)	(1020,196,100)	0.19
EG	*GF* (4)	(15,7,5)	0.46	(255,49,25)	(510,98,50)	0.19
PG	*GF* (64)	(21,11,6)	0.52	(441,121,36)	(2646,726,216)	0.27
EG	*GF* (8)	(63,37,9)	0.58	(3969,1369,81)	(11907,4107,243)	0.34
RS	*GF* (256)	(255,239,17)	0.93	(65025,57121,289)	(520200,456968,2312)	0.87
EG	*GF* (16)	(255,175,17)	0.68	(65025,30625,289)	(260100,122500,1156)	0.47
RS	*GF* (256)	(255,153,103)	0.6	(65025,23409,10609)	(520200,187272,84872)	0.36
EG	*GF* (256)	(255,175,17)	0.68	(65025,30625,289)	(520200,245000,2312)	0.47
RS	*GF* (256)	(255,223,33)	0.87	(65025,49729,1089)	(520200,397832,8712)	0.76

decoding lines and columns of a matrix, we find Berrou [4] and Robertson [8] for convolutionnal codes. Also Pyndiah [5] and Lucas [6] used the same principal to decode block codes. Hagenaur [13] presented in his work a log likelihood algebra that allows to determine the extrinsic information passing between the elementary decoders. Fouad et. al proposed a decoding scheme related to this family of binary product codes.

4.1 Non-Binary Product Codes

Binary product codes were invented (in 1954 [2]) long before turbo product codes, but the qualifier "turbo" refers to the iterative decoder which comprises two SISO constituent decoders. Such a turbo decoder is easy to derive if we assimilate a product code to a serial concatenation of block codes. The challenge then would be to design a good constituent SISO block decoders necessary for the iterative decoding scheme. In the following, we will use the SISO non-binary ITD as constituent block decoder due to its low complexity compared with other known decoders. But first, we describe the construction and the coding process of product codes.

4.1.1 Product codes construction and systematic encoding

Let C_1 and C_2 be linear block codes over the Galois field *GF* (*q*), with parameters (n_1,k_1,d_1) and (n_2,k_2,d_2), respectively. The product code $PC = C_1 \otimes C_2$ consists of all $n_1 \times n_2$ matrices such that each column is a codeword in C_1 and each row is a codeword in C_2, or inversely. It is well known that *PC* is a ($n_1 \times n_2$,$k_1 \times k_2$) linear block code with minimum distance $d_1 \times d_2$ over *GF* (*q*) [14]. The direct product construction thus offers a simple way to build long block codes with relatively large minimum distance using simple, short component codes with small minimum distance. When C_1 and C_2 are two *EG* codes over *GF* (*q*), we obtain an *EG* product code over *GF* (*q*).

Table 2 shows some examples of non-binary block product cyclic OSMLD codes. In this table we use the abbreviations as in Table 1 by adding a power of 2 in the code parameter $(n,k,d_{min})^2$, to indicate that is a symmetric product code (i.e. formed by the same component code horizontally and vertically) .

Starting from a $k_1 \times k_2$ information matrix, systematic encoding of *PC* is easily accomplished by first encoding the k_1 information rows using a systematic encoder for C_1. Then, the n_1 columns are encoded using a systematic encoder for C_2, thus resulting in the $n_1 \times n_2$ coded matrix shown in Figure 4.

In the case of a symmetrical product code, the code used horizontally and vertically is the same, thus to send a $k \times k$ matrix of information symbols, we have to send an $n \times n$ matrix. Therefore the code rate of 2 the resulting product code is $R = \frac{k \times k}{n \times n} = \frac{k}{n^2}$.

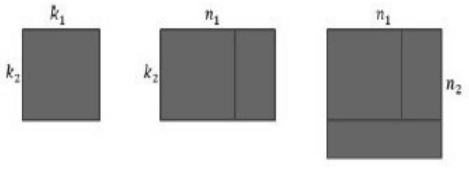

Figure 4: Code Word In The Product Code PC=$C_1 \times C_2$.

4.1.2 Binary image of EG product codes

Communication systems usually employ some form of binary signaling. Binary expansion of the EG PC

code is then required for transmission. The element of the Galois field $GF(2^m)$ have a binary representation of m bits.

The extension field $GF(2^m)$ forms a vector space of dimension m over $GF(2)$. A binary image PC_b of PC is thus obtained by expanding each code symbol, either horizontally or vertically, into m bits using the polynomial basis $\{1, \alpha, \ldots, \alpha^{m-1}\}$ where α is a primitive element of $GF(2^m)$. In this study, the product code PC is divided into one binary image, either
PC_b^- or $PC_b^|$ to symbolize the horizontal or vertical expansions respectively.

4.2 Turbo Decoding Scheme Using SISO nb-ITD As Elementary Decoder

In this section we describe the proposed turbo decoding algorithm for non-binary product codes. The decoding process of turbo codes is a suboptimal iterative processing in which each component decoder takes advantage of the extrinsic information produced by the other component decoder at the previous step.The original work in this context is due to Berrou [4] and Robertson [8] for convolutional codes, Pyndiah [5] and Lucas [6] for block codes. Hagenauer [13] gave an extrapolation of Robertson's scheme for block codes by using a trellis decoder. Belkasmi [11] used connexion scheme of Pyndiah and Threshold decoding of Massey [9] to derive a turbo decoding scheme for binary block product codes.

The iterative decoding process presented in this section follows that given by Belkasmi in [10] where we use a non binary version of the soft-input softoutput iterative threshold decoding algorithm with Pyndiah's connexion scheme, and it is designed to decode product codes based on two systematic OSMLD codes. The iterative decoding process is shown in Figure 5, and it is described in the following.

In this section we describe in details the proposed decoding process of non binary product codes. As shown in Figure 4, starting from a $I = [k_1 \times k_2]$ information matrix, systematic encoding of PC is easily accomplished by first encoding the k_1 information rows using a systematic encoder for C_1. Then, the n_1 columns are encoded using a systematic encoder for C_2, thus resulting in the $U = [n_1 \times n_2]$ coded matrix. This matrix has elements in $GF(2^m)$, therefore we can expand each of its symbols either horizontally, resulting in $V^- = [m \times n_1, n_2]$ binary matrix, or vertically resulting in $V^| = [n_1, m \times n_2]$ binary matrix. After that, we obtain the channel reliabilities llr^- and $llr^|$. Then the iteration process starts.

The block diagram of the turbo decoder at the k^{th} half-iteration is shown in Figure 6. A half-iteration stands for a row or column decoding step, and one iteration comprises two half-iterations.

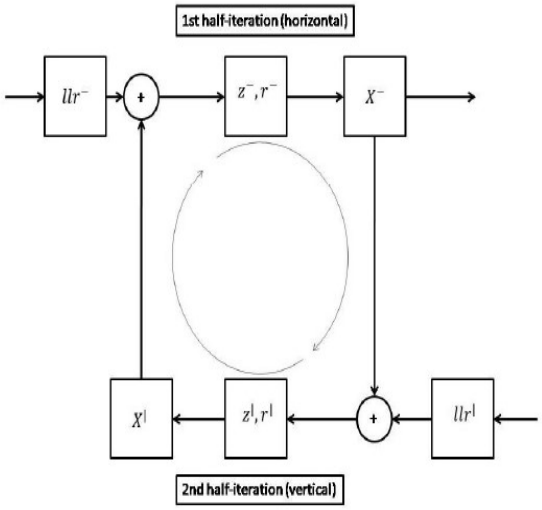

Figure 5: Block Diagram Of Our Turbo-Decoder At The K^{th}iteration Of The Prodect Decoder.

The output of the SISO decoder at horizontal halfiteration k is given by:

$$LLR^-)^{(k)} = \lambda_1 llr^- + \lambda_2 \left(X^|\right)^{(k-1)} \quad (15)$$

where λ_1 is a scaling factor used to amplify the channel reliability, and where λ_2 is a scaling factor used to attenuate the influence of extrinsic information during the iterations. The term $X^|)^{(k-1)}$ is the extrinsic information matrix delivered by the SISO decoder at the previous half-iteration and it is initially equal to 0 (i.e. .

The output of the SISO decoder at horizontal halfiteration k is given by:

$$\left(LLR^|\right)^{(k)} = \lambda_1 llr^| + \lambda_2 \left(X^-\right)^{(k-1)} \quad (16)$$

The decoder outputs an updated extrinsic information matrix $X^|)^{(k)}$, and possibly a code array $z^{(k)}$ of hard-decisions. Decoding stops when a given maximum number of iterations has been performed.

Algorithm 4 describes our proposition of the turbo block decoder based on the elementary decoder nonbinary Iterative Threshold Decoding described in the previous section.

5. RESULTS AND PERFORMANCE ANALYSIS

In this section we present the error performance of the proposed decoder for various non-binary EG product codes in terms of Symbol Error Rates (SER) and Block Error Rates (BLER). We investigate the effect of the iterations on the performance of the decoder, in order to find the optimal maximum number of iterations for each code used.

Algorithm 4 nb-block-turbo-ITD

Input: horizontal channel reliabilities llr^{-}
Input: vertical channel reliabilities $llr^{|}$
Output: non-binary Decoded array z
  while $k < I_{max}$) **do**
    compute LLR^{-} from (15)
    for $i = (1:n)$ **do**
      r_i from LLR^{-}
      z_i from LLR^{-}
      compute X_i^{-} from z_i and r_i
    end for
    compute $LLR^{|}$ from (16)
    for $j = (1:n)$ **do**
      r_j from $LLR^{|}$
      z_j from $LLR^{|}$
      compute $X_i^{|}$
    end for
    $k = k + 1$
  end while

5.1 Symbol and Block Error Rate

In the following, we will use Euclidean Geometry codes to illustrate the decoding performance of the non-binary block turbo decoder described above. The non-binary EG product codes used here have

symbols belonging to $GF(2^m)$, therefore each code symbol can be represented by m bits. Each codeword is decoded with the nb-ITD, and we compute its symbol error performance assuming BPSK signaling for transmission over the AWGN channel with a singlesided power spectral density N_0, are used in our numerical simulations. Our results are obtained by the Monte-Carlo simulation, with stopping criteria of at least 1000 decoded blocks and 200 residual symbol errors. In Table 3, we summarize the simulation settings.

Table 3: Simulation settings.

Model codes	nb-EG-$PC(n,k)^2$	Alphabet
	nb-EG-$PC(15,7)^2$	$GF(2^2)$
	nb-EG-$PC(63,37)^2$	$GF(2^3)$
	nb-EG-$PC(255,7175)^2$	$GF\ ^{24)}$
Modulation	BPSK	
Channel	AWGN	
Simulation method	Monte Carlo	
Minimum number of transmitted blocks	1000	
Minimum number of residual symbol errors	200	

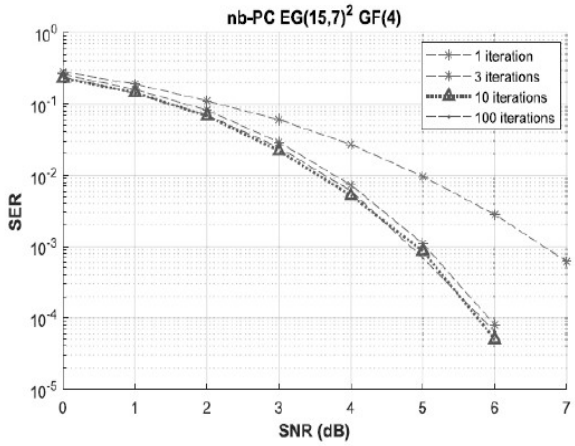

Figure 6: Symbol Error Rate Performances Of 4-Ary Nb-EG(15;7)2 Decoded With A Different Maximum Number Of Iterations.

Figure 6 shows the convergence rate of the iterative decoding of the 4-*ary* nb-$EG(15,7)^2$ product code decoded with the proposed decoder based on two nb-ITD elementary decoders. We see that convergence is very fast, about 3 iterations to reach the maximum performance, i.e. after 3 iterations there is no more noticeable amelioration. This code is based on the Euclidean Geometry $EG^{2,2^2)}$, and has symbols belonging to $GF^{2^2)}$. Thus each symbol has a 2-tuple binary representation. Therefore the resulting product code is equivalent to a (255,49,25) codes over $GF^{2^2)}$ which is itself equivalent to a binary (510,98,50).

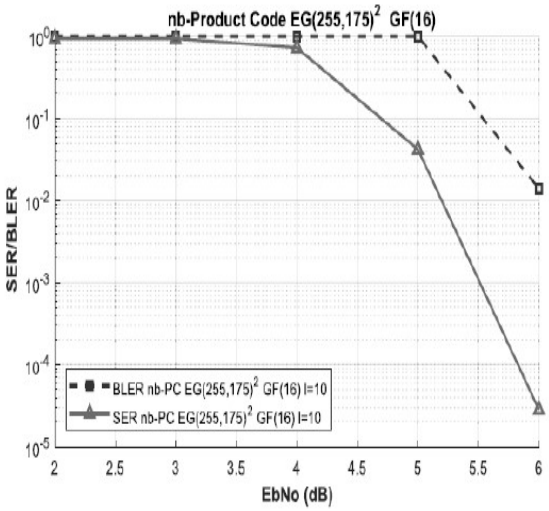

Figure 7: Symbol And Block Error Rate Performance Of Nbeg(255,175)^2 Over GF(2^4) Decoded With Nb-ITD Product Decoding Algorithm.

Example: Consider the non-binary (255,175) cyclic EG code over $GF(2^4)$ with column weight 16 and minimum distance at least 17, whose generator polynomial is $g(X) = \alpha^5 + \alpha^3X^2 + \alpha X^4 + \alpha 14X6 + \alpha 11X9 + \alpha 9X11 + \alpha 8X12 + \alpha 7X13 + {}_{\alpha}5X15 + {}_{\alpha}4X16 + {}_{\alpha}2X18 + X20 + {}_{\alpha}14X21 + {}_{\alpha}11X24 + {}_{\alpha}10X25 + {}_{\alpha}7X28 + {}_{\alpha}5X30 + {}_{\alpha}3X32 + {}_{\alpha}X34 + {}_{\alpha}13X37 + {}_{\alpha}11X39 + {}_{\alpha}10X40 + {}_{\alpha}9X41 + {}_{\alpha}8X42 + {}_{\alpha}7X43 + {}_{\alpha}13X52 + {}_{\alpha}12X53 + {}_{\alpha}11X54 + \alpha 9X56 + \alpha 8X57 + \alpha 7X58 + \alpha 5X60 + \alpha 4X61 + {}_{\alpha}3X62 + {}_{\alpha}2X63 + {}_{\alpha}X64 + X65 + {}_{\alpha}14X66 + {}_{\alpha}12X68 + {}_{\alpha}11X69 + {}_{\alpha}10X70 + {}_{\alpha}8X72 + {}_{\alpha}7X73 + {}_{\alpha}4X76 + X80$. Each symbol in GF can be expended into 4 bits. The symbol error performance of the 16-ary product code $EG(255,175)^2$ decoded with the proposed decoder (10 iterations) is shown in Figure 7.

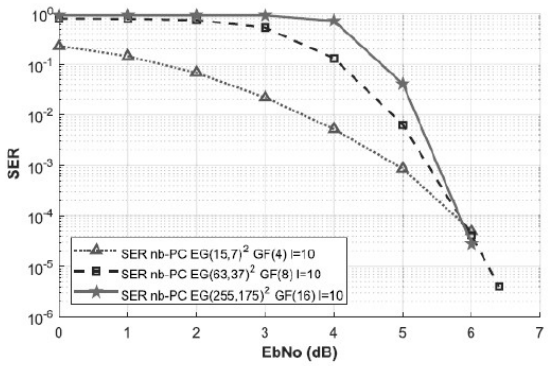

Figure 8: Symbol Error Rate Performances Of Different Nb EG Product Codes.

Finally, Figure 7 shows the symbol error rate of the product codes $(15,7,5)^2$, $(63,37,9)^2$ and $(255,175,17)^2$. These codes respectively equivalent to binary codes (510,98,50) of

0.19, (11907,4107,243) of rate 0.34 and (260100,122500,1156) of rate 0.47, as shown in Table II. We notice that the obtained product codes that have higher rates have better performance.

CONCLUSION

In this paper we presented a comprehensive study about the construction and the turbo decoding of Product Codes based on a family of cyclic EG codes, that are LDPC and OSMLD at the same time, due to their structural properties. This method of Product Codes construction results in a wide range of possible codes lengths and rates with different possible alphabets, which adds variety to the pool on nonbinary error correcting codes. The non-binary Iterative Threshold Decoding algorithm is an upgrade of the binary ITD algorithm proposed by Belkasmi *et al.* to make it decode non-binary codes. The nb-ITD updates only the extrinsic information over the iterations, while keeping the same channel reliabilities of the received symbols during each iteration. By taking into consideration the channel output, we assign weights to each one of these orthogonal equations, which associate with each decision a reliability that will be exploited during the iterations. The most pertinent point is that this decoding algorithm is of low complexity compared to most well-known decoders for non-binary codes, and since the decoding complexity is a key challenge, our decoder seems to be a good proposition for turbo decoding of non-binary product codes based on nonbinary OSMLD EG codes. We carried out simulations on an AWGN channel using BPSK modulation, and the results showed that codes with higher code rates achieve better performance. We also showed the effect of the iterations on the proposed decoder, and noticed that our decoder reaches its maximum performance just after 10 iterations, therefore no need to go all the way to a 100 iterations. The performance can be improved by changing the factors that affect channel reliability and extrinsic information during the iteration. One perspective of our work is to design a decoding scheme for parallel concatenation of non-binary OSMLD codes using SISO nb-ITD as component decoder, and study the performance in the Rayleigh fading channels using high order modulations such as QAM modulations.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, Vol. 27, pp. 379-423, 1948.

[2] P. Elias, "Error free coding,". *IRE Trans. Inform. Theory*, vol. 4, pp.29-37, Sept. 1954.

[3] Mukhtar, H., Al-Dweik, A., and Shami, A. (2016). "Turbo product codes: Applications, challenges, and future directions. "*IEEE Communications Surveys and Tutorials*, 18(4), 3052-3069.

[4] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," *IEEE Int. Conf. on Comm. ICC'93*, Geneva, May 1993, pp. 10641071.

[5] R. Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Trans. Commun.,* Aug. 1998, Vol. 46, N° 8, pp. 10031010.

[6] R. Lucas, M. Bossert and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes, " *IEEE Journal on selected areas in communications,* February 1998, Vol. 16, No 2, pp. 276-296.

[7] M. P. C. Fossorier and S. Lin. "Soft-Input SoftOutput Decoding of Linear Block Codes Based on Ordered Statistics," *Proc. 1998 IEEE Global Telecomm. Conf. (GLOBECOM'98)*, Sydney, Australia Nov. 1998. pp. 2828-2833,

[8] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. IEEE Global Commun. Conf. (GLOBCOM'94)*, San Francisco, CA, Dec. 1994, pp. 1298-1303.

[9] J.L Massey, "Threshold Decoding," Cambridge, Ma, M.I.T. Press, 1963.

[10] M. Belkasmi, M. Lahmer, and M. Benchrifa , "Iterative Threshold Decoding of Parallel Concatenated Block Codes," *Turbo Coding 2006 Conf.*, 4-7 April 2006, Munich.

[11] Yuri V. Svirid and Sven Riedel, "Threshold Decoding of Turbo-Codes," *IEEE Int. Symposium on Information Theory*, 1995, pp. 39.

[12] C. R. P. Hartman and L. D. Rudolph, "An optimum symbol by symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, Sept. 1973, Vol. IT-22, pp. 514-517.

[13] J. Hagenauer, E. Offer, and L. Papke, "iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, Mar. 1996, Vol. 42, pp. 429-446.