$$deg_{D_2}G(1,3) = 1. \tag{14}$$

Therefore, $del_{D_2}(G) = \sum deg_{D_2}(g) = 1 + 1 + 1 = 3$.

Now, few things should be observed from this example. Firstly, the presence of '1' in G is completely irrelevant to the number of delays. This is true since delays are D_1 and D_2, and not '1.' Furthermore, the method to find the delays in 1-D and 2-D are completely different. Lastly, the algorithm above is specific to a polynomial row vector.

2.4 Counting delays for a polynomial matrix

The algorithm will be presented in parallel with an example.

Example: Let

$$G = \begin{bmatrix} 1 + D_1 & D_1 D_2 & D_1^2 \\ D_2^2 & 1 & 1 + D_2 \end{bmatrix}. \tag{15}$$

Algorithm 2: Counting delays for a polynomial matrix

Step 1: Search all monomials in all entries of G. The number of delays D_1 needed is $\overset{max}{i,j}\ \{deg_{D_1}G(i,j)\}$. For the example, $G(1,3) = D_1^2$. Therefore, two D_1's are needed.

Step 2: Now, for D_2, find monomials without D_1. In the example, there are only two monomials without D_1, i.e. $G(2,1) := D_2^2$ and $G(2,3) := 1 + D_2$.

Step 3: Set $del_{D_2}(G) = max\{deg_{D_2}(g)\}$ out of all the monomials found in Step 2. In the example, $deg(D_2^2) > deg(D_2)$ therefore, $del_{D_2}(G) = 2$ so far.

Step 4: Find monomials with both D_1 and D_2. In the example, $G(1,2) := D_1 D_2$.

Step 5: Set $del_{D_2}(G) = max\{deg_{D_2}(g)\}$ out of all the monomials found in Step 4. In this case, $del_{D_2}(G) = 1$.

Step 6: Add up the number of degree found from Step 3 and 5. $del_{D_2}(G) = deg_{D_2,Step3}(g) + deg_{D_2,Step5}(g)$. For this example, the total number of D_2 is $2 + 1 = 3$. This is where the procedure for finding the numbers of delay D_2 ends.

The above algorithm as well as the algorithm for vector was simply derived from observation of the pattern through many worked out mapping to the diagram. Note that the second algorithm for matrix is quite similar to the preceding algorithm for vector.

Alternatively, the above algorithm may be reconsidered by using the notation of support.

2.5 Support

Support, as the name suggests, could be thought of as a foundation of a building's columns, i.e., if a building is to be removed, the remainder of the building would show its location and enable the important points of the support to be seen. For the purpose of illustration, (8) defines a map onto a support matrix in Fig. 3.

For the purpose of mathematical representation and computation, the support will be represented in a matrix form. The column of support represents $deg_{D_1}(g)$ and the row represents $deg_{D_2}(g)$. A '1' is used to represent the presence of the degree at that point. The rest of the matrix will be filled with 0's. To make things clear, the support of (8) is given as (16).

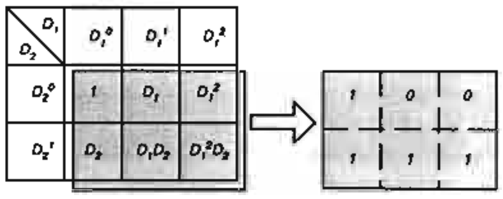

Figure 3. Mapping of a generator encoding matrix onto a support matrix using (8).

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{16}$$

In the following, the method to derive the support in matrix form is given.

Algorithm 3: Construction of a support

Again, (8) will be used to illustrate the method.

- For each monomial in $G(i,j)$, put a '1' in the support matrix, S, in the appropriate position. For example, $G(1,3) = 1 + D_1^2 D_2$. Therefore, $S(1,1) := 1$ because of the 1 in $G(1,3)$ and $S(2,1) := 1$ because of the D_2 in $G(1,1)$.
- If the same monomial is encountered more than once, then it is ignored. The presence of the second monomial does *not* cancel any entries in the support.
- Finally, the size of the matrix can be checked for verification. The number of columns is equal to the $\overset{max}{i,j}\ \{deg_{D_1}G(i,j)\} + 1$ and the number of rows is also $\overset{max}{i,j}\ \{deg_{D_2}G(i,j)\} + 1$.
- Note that for one G, there are as many S as the number of rows in G.

Algorithm 4: Counting delays for a polynomial matrix using support

Step 1: Each row of the generator matrix, G, is first mapped to the support, S_i, for $i = 1$ *to* k.

Step 2: Now, for each support, let the size of the support be $p \times q$.; Then,

$$del_{D_1}(G_i) = q - 1. \tag{17}$$

Step 4: Remove the first row of S_i, and thereby making it $p - 1 \times q$ matrix.

Step 5: This is the important step. Look at one column at a time and find '1' at the lowest row. Then count the number of rows including the final '1.' Add these numbers up for all columns. This is $del_{D_2}(G_i)$.

Step 6: Finally, to compute the total number of delays for G, add up each $del_{D_1}(G_i)$ for total number of D_1's and $del_{D_2}(G_i)$ to form total number of D_2's.

Example: Let

$$G = \begin{bmatrix} 1 + D_1 D_2 + D_1^3 & D_1 + D_2^2 \end{bmatrix}. \tag{18}$$

The support of the (18) is

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad (19)$$

3. MINIMAL DELAY GENERATOR ENCODER

3.1 Unimodular Matrix

Here, a unimodular matrix is defined as a square 2-D polynomial matrix whose determinant is a unit element in the coefficient field, i.e., one.

$$| U | = 1. \quad (20)$$

The following shows an example of unimodular matrix in of size 2×2:

$$U = \begin{bmatrix} D_1 D_2 + 1 & D_1 \\ D_2 & 1 \end{bmatrix}. \quad (21)$$

3.2 Equivalent Generator Encoder

In one-dimensional convolutional code, one way to define a generator matrix is by using an unimodular matrix [6]. Let G, G_{eq} be 2-D polynomial matrix of size $k \times n$ in the ring $R^{k \times n}$, then, G and G_{eq} are equivalent encoders if and only if there is unimodular matrix $U \in R^{k \times k}$ such that

$$G_{eq} = U \cdot G. \quad (22)$$

Since this is true for one-dimensional code by definition, therefore it is also true in two-dimensional convolutional code [7].

Given a generator matrix, to find all possible equivalent generator matrices, as many unimodular matrices must first be found. That is, given $G \in R^{k \times n}$, to find G_{eq}, $U \in R^{k \times k}$ must first exists. Instead of attempting to find all, only as many as possible equivalent generator matrices are attempted to be found.

One way to find as many as possible U for a particular $R^{k \times k}$ is to make a table and try every possible combination of polynomials.

3.3 Finding Unimodular using Support

Unimodular matrices may be generated by systematically arranging possible combinations of polynomial in the entries of the matrix. Let

$$U = \begin{bmatrix} P_1 & P_2 \\ P_3 & P_4 \end{bmatrix}, \quad (23)$$

where $P_i, i = 1,2,3,4$ is a 2-D polynomial. Now, the goal is to find a set of $\{P_i\}_{i=1}^4$ such that U becomes a unimodular matrix. For instance, P_1 may be in the form of

$$P_1 = a_1 + a_2 D_1 + a_3 D_1^2 + a_4 D_2 + a_5 D_1 D_2 + + a_6 D_1^2 D_2 + a_7 D_2^2 + a_8 D_1 D_2^2 + a_9 D_1^2 D_2^2 \quad (24)$$

where $a_i \in \{0,1\}$. Then, P_1 can be written in the form of a support:

$$S_{P_1} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}. \quad (25)$$

For P_2, a_i becomes b_i and so on.

3.4 Finding Minimal Delay Encoding Matrix

First, the given generator matrix is entered as an input. Then its required delay number is counted and stored. Next, using (22) and a stored unimodular matrix, an equivalent G is found. Then the number of delays for this new equivalent matrices is computed and compared with the original number. Whichever G that produces smaller number of delays will be stored. The process is repeated with new equivalent matrices until all unimodular matrices in storage are used. Finally, the stored G is the one that required the least number of delays found.

4. IMPLEMENTATION

An example will be given to illustrate how to generate minimal delay encoding matrix. The example was worked out with help of a computer software.

Example: Given,

$$G = \begin{bmatrix} D_1 D_2 & D_2^2 & 1 + D_1^2 \\ D_1 + D_1 D_2^2 & D_2 + D_2^3 & 1 + D_1 + D_1^2 D_2 \end{bmatrix}. \quad (26)$$

This requires four D_1's and nine D_2's. Let the input of the function be support of each entries of (26). After a search, the minimal delay generator matrix of (26) will be

$$G_{min} = \begin{bmatrix} D_1 & D_2 & 1 + D_1 \\ D_1 D_2 & D_2^2 & D_1^2 \end{bmatrix}, \quad (27)$$

which requires three D_1's and only four D_2's.

Unimodular used in the computation is

$$U_{min} = \begin{bmatrix} D_2 & 1 \\ 1 & 0 \end{bmatrix}. \quad (28)$$

5. CONCLUSION

An algorithm for finding an equivalent 2-D generator matrix which requires a smaller number of delay elements is presented. Along the derivation process, algorithms for counting the total number of 2-D delays are also derived and explained. The method proposed may sometimes not result in the minimal basic encoder but it does give a good practical result as illustrated by an example. The validity of the result depends directly on the amount of search time available and the given computation power of a search engine.

6. Acknowledgement

This research work is supported by The Thai Research Fund (TRG 4580063). The authors would like to thank all anonymous reviewers for their valuable and positive comments toward this paper.

References

[1] B. Buchberger, "Groebner Bases and System Theory," *Multidimensional Systems and Signal Processing*, vol. 12, 2001, Boston: Kluwer Academic Publishers, pp. 223-251.

[2] C. Charoenlarpnopparut, "Progress and Open Problems in Multidimensional Multirate System Design," Thailand: Sirindhorn International Institute of Technology.

[3] C. Charoenlarpnopparut and S. Tantaratana, "Multidimensional convolutional code: Progresses and bottlenecks," 2003 IEEE International Symposium on Circuits and Systems, Bangkok, Thailand, May 2003.

[4] D. Cox, J. Little and D. O'Shea, "IDEALS, VARIETIES, AND ALGORITHM: An Introduction to Computational Algebraic Geometry and Commutative Algebra," 2^{nd} ed., Springer-Verlag, New York, 1996.

[5] E. Fornasini and M. E. Valcher, "Algebraic Aspects of Two-Dimensional Convolutional Codes," *IEEE Trans. Info. Theory,* vol. 40, No. 4, July 1994, pp. 1068-1082.

[6] Johannesson, Rolf and Zigangirov, Kamil Sh., "Fundamentals of Convolutional Coding," New York: IEEE Press, 1999.

[7] Paul A. Weiner, "Multidimensional Convolutional Codes," Ph.D. Dissertation, University of Notre Dame, 1998. `http://www.nd.edu/ rosen/preprints.html`.

MULTIDIMENSIONAL CONVOLUTIONAL CODE: PROGRESSES AND BOTTLENECKS

C. Charoenlarpnopparut and S. Tantaratana

Telecommunications Engineering Program
School of Communications, Instrumentations & Control
Sirindhorn International Institute of Technology, Thammasat University
Klongluang, Pathumthani 12121 Thailand
Phone : (+662) 986-9009 Ext. 1808, FAX : (+662) 986-9009 Ext. 1801
E-mail : chalie@siit.tu.ac.th

ABSTRACT

Multidimensional (m-D, $m \geq 2$) finite support convolutional code has become a new area of research in the multidimensional system and signal processing. While the one-dimensional convolutional code and its variants have been thoroughly understood, the m-D counterpart still lacks unified notation and practical implementation. In this document, the issues related to equivalent encoder, syndrome decoder and the realization of m-D convolutional code are discussed. The application of Gröbner basis theory for finding a syndrome decoder matrix is given. In addition, the formulations of optimal realization problem and m-D convolutional code design problem is stated. Recent development and one open-problem are also reported.

1. INTRODUCTION

Examples of 1-D information are voice signal, audio signal, node voltage, line current. By sampling these signals and quantizing the samples, one obtain discrete sequences of binary representation. On the other hand, multidimensional (m-D, $m \geq 2$) information may be taken from 2-D images ($m = 2$), e.g. MRI film, photos, 3-D images ($m = 3$), e.g. holograms, animation of 2-D images, e.g. video, movies ($m = 3$) and animated 3-D images ($m = 4$). To encode multidimensional information, it is conventional to first transform m-D data into 1-D sequence by means of scanning, and then apply the 1-D encoding technique. This procedure is, in fact, unnatural and ignores the correlation in all other direction except in the scanning direction.

To motivate the importance of multidimensional convolutional code, note that 1-D convolutional code is a generalization of a block code, i.e. block code is a 1-D convolutional code with the delay variable D replaced by zero. Equivalently, 0-D convolutional code is essentially a block code. As a result a multidimensional convolutional code can be viewed as a generalization of all codes.

While the one-dimensional convolutional code and its variants have been thoroughly understood [7, 8, 10, 11, 12, 13], the m-D counterpart still lacks unified notation and efficient implementation. In this document, the aim is to emphasize the recent development [6, 15, 16] of m-D convolutional code theory as well as to point out the bottleneck as well as an open problem to the multidimensional community.

This research work is supported by The Thai Research Fund (TRG4580063)

2. REPRESENTATION OF MULTIDIMENSIONAL SIGNALS

In general, there are two approaches to represent multidimensional signals:

- Geometrical Approach: an m-D signal is represented by a m-D finite matrix, for example a 3×3 pixel image I with 16 gray-scale levels (4 bits) can be represented as

$$I = \begin{bmatrix} 1100 & 0101 & 1001 \\ 1000 & 1001 & 0000 \\ 1011 & 0100 & 0010 \end{bmatrix} \quad (1)$$

- Algebraical Approach: an m-D signal is represented by a m-variate polynomial vector of size $1 \times b$, where b is the number of symbols used for representing each sample, for example the 2-D image in the previous example ($b = 4, m = 2$) may be represented by

$$u = \begin{bmatrix} 1 + D_2 + D_1^2 + D_1D_2 + D_2^2 \\ 1 + D_1 + D_1D_2^2 \\ D_2^2 + D_1^2D_2^2 \\ D_1 + D_1^2 + D_1D_2 + D_2^2 \end{bmatrix}^T \quad (2)$$

Note that in the above example, the i^{th}-row j^{th}-column element of the matrix I in Eq.(1) is associated with the multiplying factor $D_1^{j-1}D_2^{i-1}$ and the p^{th} element of the vector u in Eq.(2) is taken from the summation of all p^{th} symbols, of all elements of the matrix I, multiplying by their associated monomials.

3. CONVOLUTIONAL ENCODER

3.1. Notation and definition

Let $\mathbf{F} = \mathbf{F}_q$ be the finite field with q elements and $R = \mathbf{F}[D_1, D_2, \ldots, D_m]$ be the ring of m-variate polynomials whose coefficients belong to the field $\mathbf{F}$. Let $k \geq 1$ be a positive integer, then the free

R-module [1, p.114], R^k, is the m-variate polynomial row vector space of length k over $\mathbf{F}$, i.e.

$$R^k = \{[\ v_1 \quad v_2 \quad \cdots \quad v_k\ ] \,|\, v_i \in R, i = 1, 2, \ldots, k\}$$

Definition 1: An $m-$dimensional convolutional code of length n over $\mathbf{F}$ is an R-submodule $C \subseteq R^n$. An element $w \in C$ is called a *codeword.*

Definition 2[15]: A code is called a *modular code* if it is closed under vector addition and under m-variate polynomial multiplication.

Fact[16]: Every m-dimensional convolutional code is finitely generated as an R-module, i.e. there exists a finite set of row vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in R^n$ such that

$$C = \left\{ \Sigma_{i=1}^{k} u_i \mathbf{g}_i | u_i \in R \right\}$$

3.2. Generator matrix

Similar to 1-D case, a generator matrix of a m-D convolutional code can be viewed as a transfer function matrix of the encoder and it is a center of focus in the encoder design and realization stage.

Definition 3: Let G be an $k \times n$ m-variate polynomial matrix constructed row-wise from a set of row vectors $\{\mathbf{g}_i\}_{i=1}^{k}$, i.e.

$$G = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix}.$$

If an m-D convolutional code C = rowspace(G), the polynomial matrix, $G \in R^{k \times n}$, is called a *generator matrix* of C. The existence of G is guaranteed since any m-D convolutional code is a finitely generated R-module.

Furthermore if G is of full row rank (i.e. rank$(G) = k$), then the rate of C is $\frac{k}{n}$. Note that it is desirable for a generator matrix to have full row rank in order to avoid having different input sequences mapped to the same codeword (undistinguishable by the decoder which message is being sent).

Definition 4: If C is free of rate $\frac{k}{n}$, then C = rowspace(G) for some $G \in R^{k \times n}$ and G is called an *encoder* of the code C

3.3. Encoding procedure

In this section, a 2-D convolutional encoding process is illustrated by a given example.

Let us consider a 2-D convolutional code, of rate $\frac{2}{3}$ whose generator matrix is given by

$$G = \begin{bmatrix} 1 & D_1 & D_1 D_2 \\ 0 & D_2 & D_1 + 1 \end{bmatrix},$$

and let I be the 2-D binary input taken from a 4-color image with the size of 3×3 pixels, i.e.

$$I = \begin{bmatrix} 01 & 10 & 10 \\ 11 & 00 & 00 \\ 01 & 10 & 01 \end{bmatrix},$$

whose algebraic representation is given as,

$$w = [\ D_1 + D_2 + D_1^2 + D_1 D_2^2 \quad 1 + D_2 + D_2^2 + D_1^2 D_2^2\ ].$$

The output y of an encoder G can be obtained by vector matrix multiplication,

$$\begin{aligned} y &= w \cdot G \\ &= \begin{bmatrix} D_1 + D_2 + D_1^2 + D_1 D_2^2 \\ \begin{pmatrix} D_2 + D_1^2 + D_1 D_2 + D_2^2 + \\ D_1^3 + D_2^3 + D_1^2 D_2^2 + D_1^2 D_2^3 \end{pmatrix} \\ \begin{pmatrix} 1 + D_1 + D_2 + D_1 D_2 + D_2^2 + D_1^2 D_2 + \\ D_1^2 D_2^2 + D_1^3 D_2 + D_1^3 D_2^2 + D_1^2 D_2^3 \end{pmatrix} \end{bmatrix}^T. \end{aligned}$$

The polynomial output vector y can be represented in the geometrical form as

$$I_{out} = \begin{bmatrix} 001 & 101 & 110 & 010 \\ 111 & 011 & 001 & 001 \\ 011 & 100 & 011 & 001 \\ 010 & 000 & 011 & 000 \end{bmatrix}.$$

Observation 1: If k is the number of symbols(bits) of each input pixel, and the generator matrix is of size $k \times n$, the number of symbols(bits) of each encoded pixel is n and independent of the number of input pixels. On the other hand, the size of the encoded image (2-D array) depends directly on the row-wise maximal total degree of the generator matrix.

Observation 2: In the case when the input is of large size (e.g. 256×256 pixels), the total degree of the input vector would grow significantly. However, this is not a problem in the implementation stage since the realization of the encoder is done by using an array shift register, not polynomial multiplication.

Observation 3: One assumption used here is that the number of rows of the generator matrix G must be the same as the number of symbols (bits) representing each pixel. When this is not the case, one needs to regroup the m-D data symbols to suit the encoder matrix.

3.4. Equivalence of encoders

It is worth noting that there may be more than one generator matrix G that can generate a given m-D convolutional code C. Those generator matrices that generate the same set of codewords are said to be *equivalent.*

Definition 5: A m-variate polynomial matrix $U \in R^{k \times k}$ is *unimodular* if U is a unit in $R^{k \times k}$. In other words, U is unimodular if and only if det(U) is a unit in R (i.e. a nonzero element of $\mathbf{F}$).

Proposition 1[16]: Let G, $G_1 \in R^{n \times k}$ be of full row rank. G and G_1 are equivalent encoders if and only if there exists a unimodular matrix $U \in R^{k \times k}$ such that $U \cdot G = G_1$.

From Proposition 1, given two generator matrices G and $G_1 \in R^{n \times k}$, the following procedure can be used to determine whether they are equivalent.

- *Step 1.* Construct a module M generated by the columns of matrix G.
- *Step 2.* Compute the Gröbner basis module of M with respect to any ordering by using Buchberger's algorithm for module[1, p.149]

- *Step 3.* Use the division algorithm [1, p.145] to identify if all columns of G_1 are members of the Gröbner basis module. If not, conclude that G and G_1 are not equivalent. If so, G and G_1 are equivalent, and the unimodular matrix U can be computed by retracing the steps of Buchberger's algorithm (see example 3.6.1 in [1, pp.153-154]).

3.5. Syndrome decoder and dual code

The decoding of m-D convolutional code can be performed in a straightforward manner by means of syndrome decoding. However, unlike the soft-decision scheme, e.g. Viterbi decoding in 1-D, this hard-decision decoding method may not be optimal in the maximum-likelihood (ML) sense nor the maximum apriori (MAP) sense [13]. In this section, the basic definition and existence criteria of syndrome decoder are given along with the algorithm for computing the syndrome decoder matrix.

Definition 6: Let C be an R-submodule of the free R-module, R^n. An *image representation* of C is a matrix $G \in R^{k\times n}$ such that $C = \text{rowspace}(G)$. A *kernel representation* of C is a matrix $H \in R^{n\times p}$, where p is some positive integer, such that $C = \ker(H) = \{w \in R^n | w \cdot H = 0\}$.

If a sequence $w \in C$, then $w = u \cdot G$ for some $u \in R^k$, and so $w \cdot H = u \cdot G \cdot H = 0, \ \forall u$. This implies $GH = 0$.

Given an m-D convolutional code, C with the associated generator matrix G, the kernel representation (i.e. the matrix H), if exists, can be constructed by computing the syzygy [1, pp.161-168] of the module generated by all columns of the generator matrix G.

Example 1: Let G be the minor left prime matrix

$$G = \begin{bmatrix} 1 & D_1 & D_1D_2 \\ 0 & D_2 & D_1+1 \end{bmatrix}.$$

The kernel representation of G is computed by using Singular program [9] command lines:

```
>ring r=2,(x,y),(c,dp);
>module M=[1,0],[x,y],[xy,x+1];
>module H=syz(M);
>H;
H[1]=[x2+xy2+x,x+1,y]
```

to obtain

$$H = \begin{bmatrix} D_1 + D_1^2 + D_1D_2^2 & D_1+1 & D_2 \end{bmatrix}^T.$$

By the properties of the matrix H and syzygy, all codewords in the set $C = \text{rowspace}(G)$ satisfy the condition:

$$w \cdot H = 0, \quad \forall\, w \in C.$$

This condition provides a *parity check*, which can be applied to a received sequence for testing whether it is a codeword. The matrix H is often called the *parity check matrix*, or the *syndrome decoder* of C and the corresponding codes are called *finite convolutional code*. Note that not every convolutional code admit a syndrome decoder (i.e. the parity check matrices of some m-D convolutional codes do not exist). The following proposition characterizes the existence of a syndrome decoder.

Proposition 2 [15, 16]: A free modular code C admits a syndrome decoder if and only if C has a **minor prime** generator matrix G.

(An m-variate polynomial matrix G is minor prime[3], if all maximum-size minors (major determinants) are devoid of common factor other than units.) For a Laurent polynomial ring, the existence condition[15] is relaxed to the condition that the generator matrix must be left factor prime.

Example 2: Consider a 3-D convolutional code C with a corresponding generator matrix

$$G = \begin{bmatrix} D_1 & 0 & D_2 \\ 0 & D_1 & D_3 \end{bmatrix}.$$

The major determinants of G contain a common factor of D_1. As a result, G is not minor prime. In [16, p.16], it has been shown that G is in fact left factor prime. By using Singular program, $H = \begin{bmatrix} D_2 & D_3 & D_1 \end{bmatrix}^T$ is the syzygy of the module generated by all columns of G. However, this matrix H is not a syndrome decoder of C since $v = \begin{bmatrix} D_3 & D_2 & 0 \end{bmatrix}$ satisfies $v \cdot H = 0$, but it does not belong to the set C (i.e. there does not exist a row vector $u \in R^2$ that produces $u \cdot G = \begin{bmatrix} D_3 & D_2 & 0 \end{bmatrix}$.)

Definition 7: The *orthogonal module* of M is denoted by $M^\perp$ and is given as

$$M^\perp \triangleq \{w \in R^n | wv^t = 0, \forall v \in M\}.$$

For an m-D convolutional code C, the *orthogonal code* of C is denoted by the module $C^\perp$. In the case where the C admits a syndrome decoder (i.e. has a kernel representation), the orthogonal code $C^\perp$ is called the *dual code* of C.

4. REALIZATION OF CONVOLUTIONAL ENCODER

The implementation of multidimensional convolutional code requires the usage of various dimensional delays (shift registers), $D_1, D_2, \ldots$ where an example is shown in Figure 1.

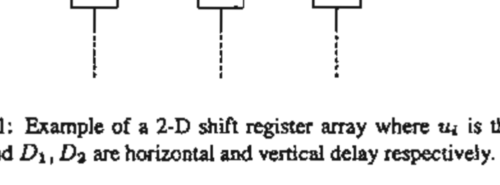

Figure 1: Example of a 2-D shift register array where u_i is the input and D_1, D_2 are horizontal and vertical delay respectively.

To realize the encoding process efficiently in the form of hardware, it is necessary to minimize the number of delays needed. In the 1-D case, one objective is to single out the generator matrix, from all equivalent ones, that has the minimum overall constraint length. However, a meaningful definition of constraint length, in a higher dimension case, has not been given anywhere. A good candidate for this parameter should directly indicate the total number of delays in the implementation of a desired encoder. A practical procedure towards the solution of this problem is only possible by direct searches via computer programming which is tedious and inefficient.

Therefore here is an open problem. Given a generator matrix $G \in R^{k \times n}$, identify, among all equivalent generator matrices, the one whose realization uses the minimum number of delay elements.

5. CONCLUSION

It has been shown that there exists a strong relationship between the theories of multidimensional system and those of m-D convolutional code. In the system point of view, the focus is on the input and output relationship and how the system responds to a different input. On the other hand, the coding side only concentrates on the set of output (codewords) and its properties. The set of all m-D convolutional codewords generated by a generator matrix G can be modelled as a R-submodule, where many mathematical tools such as Gröbner basis, module theory and algebraic geometry can be applied.

There are still many areas of m-D convolutional which are rather unexplore, e.g. the design of m-D convolutional code, performance measurement, minimal realization. There has been a study [16] on the weight and distance bound of m-D convolutional code which can be further investigated and may lead to the completion of optimal m-D convolutional code design procedure. Some progress has been made for special class of m-D convolutional code design, namely the unit memory codes (the ones whose generator matrix contains only first or zero degree polynomials.

6. ACKNOWLEDGEMENT

This research work is supported by The Thai Research Fund (TRG 4580063). The authors would like to thank Associate Professor Zhiping Lin for his valuable comment and discussion.

7. REFERENCES

[1] W.W. Adams and P. Loustaunau, *"An Introduction to Gröbner Bases,"*, Graduate Studies in Mathematics, vol.3, American Mathematical Society, 1994.

[2] B. Buchberger, "Gröbner bases: An algorithmic method in polynomial ideal theory," *in Multidimensional System Theory (N.K. Bose, ed.)*, Reidel, Dordrecht, 1985, pp. 184-232.

[3] C. Charoenlarpnopparut and N.K. Bose, "Multidimensional filter bank design using Groebner bases," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 12, December 1999, pp. 1475-1486.

[4] C. Charoenlarpnopparut and N.K. Bose, "Groebner Bases For Problem Solving in Multidimensional Systems," *Multidimensional Systems and Signal Processing*, vol. 12, no. 3-4, 2001, pp. 365-376.

[5] D. Cox, J. Little and D. O'Shea, *"IDEALS, VARIETIES, AND ALGORITHM: An Introduction to Computational Algebraic Geometry and Commutative Algebra,"* 2^{nd} edition, Springer-Verlag, New York, 1996.

[6] E. Fornasini and M.E. Valcher, "Algebraic aspect of 2D convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-40, No. 4, 1994, pp.1068-1082.

[7] G.D. Forney, Jr, "Convolutional codes I: Algebraic structure," *IEEE Trans. Info. Theory*, vol. IT-16, Nov. 1970, pp. 720-738.

[8] G.D. Forney, Jr, "The Viterbi algorithm," *Proc. of the IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[9] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 2.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2001). `http://www.singular.uni-kl.de`.

[10] R. Johannesson and Z. Wan, "A linear algebra appoach to minimal convolutional encoders," *IEEE Trans. Information Theory*, vol. 39, no. 4, Jul. 1993, pp. 1219-1233.

[11] R. Johannesson and Z. Wan, "Some structural properties of convolutional codes over rings," *IEEE Trans. Information Theory*, vol. 44, no. 2, Mar. 1998, pp. 839-845.

[12] S. Mahapakulchai and R. E. Van Dyck, "Design of ring convolutional trellis codes for MAP decoding of MPEG-4 imagery," *Proc. IEEE Inter. Conf. Commun. (ICC 2001)*, June 2001.

[13] S. Mahapakulchai, "MAP source-controlled channel decoding for image transmission using CPFSK and ring convolutional codes," *Ph.D. Dissertation*, Pennsylvania State University, PA, USA, 2002.

[14] J.L. Massey and T. Mittelholzer, "Convolutional codes over rings," in *Proc. 4th Joint Swedish-Soviet Int. Workshop Information Theory* Gotland, Sweden, Aug. 27-Sept. 1, 1989, pp. 14-18.

[15] M.E. Valcher and E. Fornasini,"On 2-D finite support convolutional codes: An algebraic approach," *Multidimensional Systems and Signal Processing,"* Vol. 5, No. 3, July 1994, pp. 231-243.

[16] P.A. Wiener,"Multidimensional Convolutional Codes," *Ph.D. Dissertation*, University of Notre Dame, IN, USA, 1998.

Algebraic Approach to Reduce the Number of Delay Elements in the Realization of Multidimensional Convolutional Code

C. Charoenlarpnopparut and S. Tantaratana
Telecommunications Engineering Program
Sirindhorn International Institute of Technology, Thammasat University, Thailand
E-mail : chalie@siit.tu.ac.th

***Abstract*—Given an $m-$D convolutional code with a generator matrix G, the goal is to find an equivalent generator matrix G' (which will generate the same code) that requires the fewest number of delay elements to implement in the canonical form. The technique proposed earlier is based on the sequential search which is not suitable for large generator matrices. It has been inspected that the number of delay elements required for implementation depends greatly on the maximum total degree of each row vector of the generator matrix. In this paper, the algebraic approach based on the usage of Gröbner basis theory and the row reduction technique is proposed in the form of an iterative algorithm for row-wise reducing the maximum total degree and thus decreasing the total number of delay elements.**

Keywords: $m-$D convolutional code, canonical realization, Gröbner basis, channel coding, generator matrix, unimodular

I. Introduction

The heart of digital communication relies on the generating, transmitting and receiving of binary digits (bits). When the digital information is transmitted over a noisy communication channel, it is often corrupted by noise and interference. In his celebrated channel coding theorem, Claude E. Shannon showed that every communication channel has a certain level of channel capacity C, which is a function of channel noise and the signal power, where the data transmission rate R cannot exceeded. To achieve the transmission rate close to the capacity of the channel, some level of coding must be implemented on the information sequence (channel coding).

In multidimensional signal processing community, over the last decade, significantly larger part of the research interest has been put on the post-processing and the source coding of the multidimensional information e.g. the m-D wavelet decomposition, the filter bank design[4] and the multi-sensor, multi-resolution signal processing[2]. To complete the communication system for $m-$D information, the compatible and efficient channel coding/decoding scheme has to be implemented. The focus of this work is on the minimal realization of the modular $m-$D convolutional encoder. Given an $m-$D convolutional code with a generator matrix G, the goal is to find an equivalent generator matrix G' (which will generate the same code) that requires the fewest number of delay elements to implement in the canonical form[6], [8], [14].

The organization of this paper is as follows. First, the general description and notation on $m-$D convolution code is explained and the algorithm for counting the nu ber of delay elements for a given generator matrix is pr posed. The main result of the paper is Algorithm 2 whe the equivalent generator matrix requiring fewer number delay elements for realization is algebraically compute The algorithm is based on the row reduction techniq and the usage of Gröbner basis theory for module[1], [Finally, the example and conclusion are included to co plete this paper.

A. *Multidimensional signals and codes*

When transmitting an m-D signal, it is conventional convert the multidimensional array to a 1-D sequence mean of scanning and encode the 1-D sequence by usi 1-D coding scheme [11], [13], [15]. However, it is mo natural and systematic to directly encode the m-D arr by using multidimensional coding scheme. The correlati among elements in the array are generally well preserv and can potentially improve the performance of the err correction/recovery algorithm in the decoding stage.

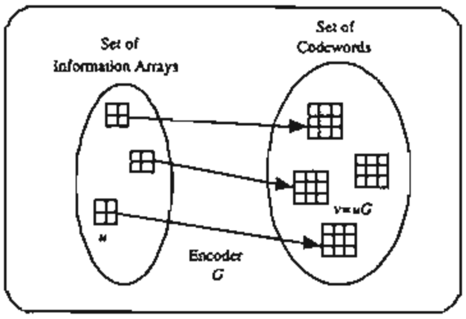

Fig. 1. Graphical explanation of the $m-$D encoding process

In general, an encoding process is an one-to-one mappi between the input data array and the output codewor shown in Figure 1. To protect the transmitted inform tion from channel noise and interference, the known r dundancy is incorporated into the transmitted array. a result, the codeword set is generally a larger array. code is define as a set of all possible codewords generat by mapping from all possible input data array. Wh some conditions such as linearity and shift-invariance a imposed, the set of codes is called modular[16].

B. *Notation and definition*

Let $\mathbb{F} = \mathbb{F}_q$ be the finite field with q elements an $R = \mathbb{F}[D_1, D_2, \ldots, D_m]$ be the ring of m-variate polynom als whose coefficients belong to the field $\mathbb{F}$. The variabl $D_1, D_2, \ldots, D_m$ are the representations of multidime sional delay elements in Z-domain and thus by means

Z-transform (some literature uses the term *D-transform*) and portioning, an input data array can be represented by an m-variate polynomial vector of length k.

Let $n \geq 1$ be a positive integer, then the free R-module [1, p.114], R^n, is the m-variate polynomial row vector space of length n over $\mathbb{F}$, i.e.

$$R^n = \{[\begin{array}{cccc} v_1 & v_2 & \cdots & v_n \end{array}] | v_i \in R, i = 1, 2, \ldots, n\}$$

Definition 1: An $m-$dimensional convolutional code of length n over $\mathbb{F}$ is an R-submodule $C \subseteq R^n$. An element $v \in C$ is called a *codeword.*

Definition 2[16]: A code is called a *modular code* if it is closed under vector addition and under m-variate polynomial multiplication. All convolutional codes are modular.

Fact 1[17]: Every m-dimensional convolutional code is finitely generated as an R-module, i.e. there exists a finite set of row vectors $g_1, g_2, \ldots, g_k \in R^n$ such that

$$C = \{\Sigma_{i=1}^{k} u_i g_i | u_i \in R\}.$$

C. Generator matrix

Similar to 1-D case, a generator matrix of a m-D convolutional code can be viewed as a MIMO transfer function matrix of the encoder and it often is a center of focus in the encoder design and realization stage.

Definition 3: Let G be an $k \times n (k \leq n)$ m-variate polynomial matrix constructed row-wise from a set of row vectors $\{g_i\}_{i=1}^{k}$, i.e.

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix}.$$

If an m-D convolutional code $C = \text{rowspace}(G)$, the polynomial matrix, $G \in R^{k \times n}$, is called a *generator matrix* of C. The existence of G is guaranteed since any m-D convolutional code is a finitely generated R-module.

Furthermore if G is of full row rank (i.e. $\text{rank}(G) = k$), then the rate of C is $\frac{k}{n}$. Note that it is desirable for a generator matrix to have full row rank in order to avoid having different input sequences mapped to the same codeword (undistinguishable by the decoder which message is being sent).

Definition 4: If a code C is free of rate $\frac{k}{n}$, then $C = \text{rowspace}(G)$ for some $G \in R^{k \times n}$ and G is called an *encoder* of the code C

It is well-known that an R-module can be generated by many different sets of module generators. Analogously, a given m-D convolutional code C can be generated by more than one encoders. This class of generators that generate the same set of code is defined as a class of *equivalent generators*

Definition 5: An m-variate polynomial matrix $U \in R^{k \times k}$ is *unimodular* if U is a unit in $R^{k \times k}$. In other words, U is unimodular if and only if $\det(U)$ is a unit in R (i.e. a nonzero element of $\mathbb{F}$).

II. Main Result

In the 1-D case, there exists a well-known algebraic algorithm for computing the minimal convolutional encoder [13], [14]. The 1-D algorithm is based on the Smith-form description. The direct generalization of this algebraic approach is not possible due to the lack of generalization of Smith-form description in $m-$D. Furthermore, the number of delay elements in the realization of $m-$D convolutional encoder cannot be computed by adding the row-wise constrain length (constrain length is undefined in $m-$D convolutional code). The attempt here is to determine an alternative $m-$D approach similar to the algorithm given in [13], [14]. In the next subsection, the algorithm for counting delay elements is given and followed by the algorithm for finding the minimal encoder.

A. Determination of total number of delay elements

Using the direct realization[6], given a 2-D generator matrix G, the algorithm for counting the total number of delay elements can be implemented by first finding the support of each row of the generator matrix. Then, apply Algorithm 1 [6], [8].

Algorithm 1: Counting delays for a generator matrix using support

Step 1. Each row of the $k \times n$ generator matrix, G, is first represented by the corresponding support matrix, S_i, for $i = 1$ *to* k.

Step 2. For each support S_i, let the size of the support be $p \times q$, where D_1 is denoted row-wise and D_2 is denoted column-wise. Then, the number of delays D_1 required to implement the i^{th} row is given by $del_{D_1}(S_i) = q - 1$.

Step 3. Remove the first row of S_i, and thereby making it a $(p-1) \times q$ matrix.

Step 4. Consider one column at a time and identify the row number (the top row is counted as row number 1) of the last row (of that column) that contains '1'. The number, $del_{D_2}(S_i)$ of delays D_2 required is obtained by adding all of these row numbers together.

Step 5. Finally, to compute the total number of delays for G, add up each $del_{D_1}(S_i)$ for number of D_1's and $del_{D_2}(G_i)$ for that of D_1's to form total number of delays i.e.

$$N = \sum_{i=1}^{k} del_{D_1}(S_i) + del_{D_2}(S_i).$$

By inspection, it can be inferred that the total number of delay elements required for realizing a given convolutional code depends greatly on the maximum total degree of each row vector of its generator matrix G. Using this fact, one can remove the unnecessary delays during the implementation by first finding a lower row-wise maximum total degree equivalent generator matrix $G' = UG$, where U is the unimodular matrix and use it for the realization instead.

B. Minimal Delay Encoder

There are two approaches proposed for finding the minimal delay encoder. The first approach is based on the sequential search over the finitely many set of sample space, whose members are the equivalent generator matrices obtained by multiplying the given generator matrix by unimodular matrices. The result of this approach directly depend on the number of unimodular matrices used for forming the searching space. The second approach here is based on the usage of Gröbner basis to perform the algebraic row-operation of the generator matrix to reduce row-wise the maximum total degree. This approach has a significant advantage over the previous method in that it provides fast and systematic way to the suboptimal solution. The term *"suboptimal"* is used here due to the fact that there may exist a generator matrix with larger total degree but requiring a smaller number of delay elements than the one with minimum total degree. When the encoder is realized in the direct form[6], the total degree is directly (but not proportionally) related to the number of delays. The algebraic approach to find the minimal realization where the optimal solution is guaranteed may not xist due to the nonlinear relationship between the wellrdering degree and the total number of delays.

Consider a generator matrix G, whose rows are $_1, g_2, \ldots, g_k \in R^n$. Multiplying G by a unimodular matrix $_1$ to get another equivalent generator matrix $G_1 = U_1 G$ s essentially performing a row operation on G. The muliplying factor U_1 is constructed, by using Gröbner basis vith respect to a proper choice of degree ordering, such hat the maximum total degree of a selected row in G' is ss than that of G.

Algorithm 2: Minimization of the row-wise maximum egree (w.r.t. degree reverse lexicographical ordering)

Give an $k \times n$ generator matrix G whose rows are denoted y $g_1, g_2, \ldots, g_k \in R^n$,

- *Step 0:* Initialize $i = 1$, where i is the row number to be reduced.
- *Step 1:* Define a module M generated by all row vectors except the i^{th} row.
- *Step 2:* Compute the *reduced* Gröbner basis S of the module M w.r.t. degree lexicographical ordering and *term over position* (TOP) ordering [1, p.142].
- *Step 3:* Reduce the i^{th} row w.r.t. the Gröbner basis S. The leading monomial of the reduced i^{th} row will have a smaller or equal degree.
- *Step 4:* Form the updated generator matrix G' by replacing the i^{th} row of G by the reduced i^{th} row obtained in Step 3. This generator matrix is equivalent to the given generator matrix.
- *Step 5:* If $i < k$, let $i = i + 1$ and go to Step 1. Otherwise, the algorithm is completed.

; should be noted that

. In Step 2 of Algorithm 2, the degree lexicographical ordering is used because the aim here is to reduce the maximum total degree. In the degree lexicographical ordering, the monomial with the maximum total degree is the leading monomial and will be reduced first in the reduction process.

2. The algorithm can be reapplied to the resulting generator matrix to further reduced the maximum degree (w.r.t. degree lexicographical and TOP orderings).

3. Furthermore, by computing the reduced Gröbner basis, each row of the updated generator matrix is reduced with respect to the module generated by the remaining rows. As a result, the final generator matrix after repeatedly applying Algorithm 2 will converge to the minimum total degree among all possible equivalent generator matrices.

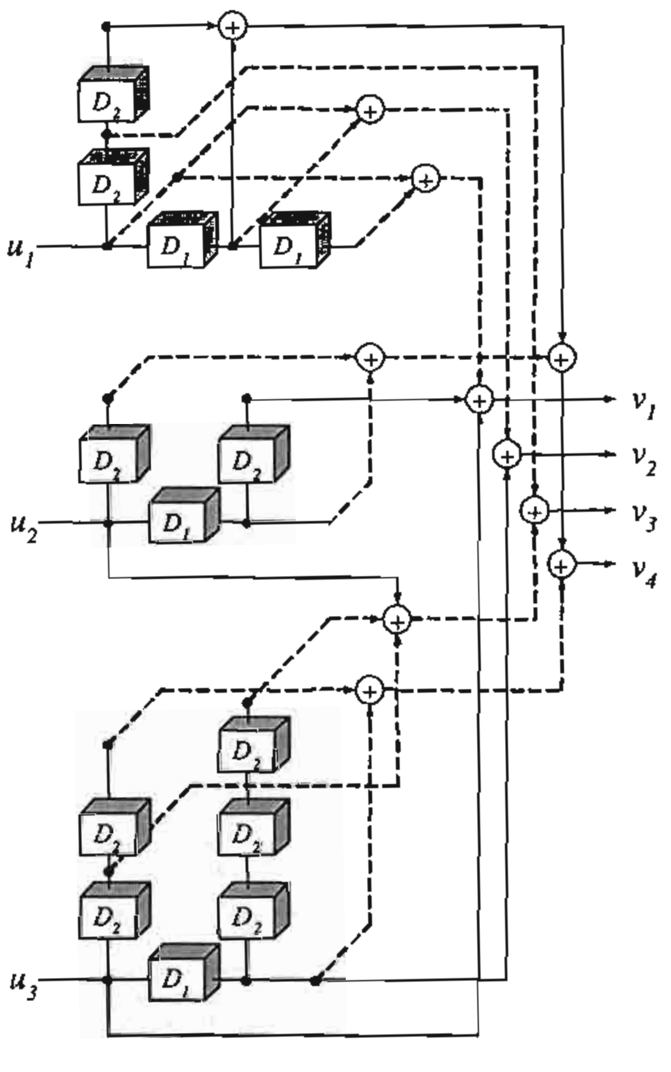

Fig. 2. Realization of the convolutional code with the generator matrix G'.

C. Examples

Consider a binary 2-D 3×4 generator matrix $G \in \mathbb{F}_2[D_1, D_2]$,

$$G = \begin{bmatrix} G_{11} & 1 + D_1 + D_1^3 D_2 & G_{13} & G_{14} \\ G_{21} & 1 + D_1^2 + D_1 D_2 & G_{23} & G_{24} \\ 1 & D_1 & D_2 + D_1 D_2^3 & D_1 + D_2^2 \end{bmatrix},$$

where

$$\begin{aligned}
G_{11} &= 1 + D_1^2 + D_1^2 D_2, \\
G_{13} &= D_2 + D_1^2 D_2^2 + D_1^3 D_2^4, \\
G_{14} &= D_1 + D_2^2 + D_1^3 D_2 + D_1^2 D_2^3, \\
G_{21} &= 1 + D_1 + D_2 + D_1 D_2 + D_1^2 + D_1^3, \\
G_{23} &= 1 + D_2 + D_1 D_2 + D_2^2 + D_1 D_2^4, \\
G_{24} &= D_2 + D_1 D_2 + D_1^2 + D_2^2 + D_1 D_2^2 + D_2^3.
\end{aligned}$$

This generator matrix G requires 28 delay elements, detailed in Table I.

TABLE I

Numbers of delays required for implementation of G and G'

Row No.	G			G'		
	D_1	D_2	Sum	D_1	D_2	Sum
1	3	9	12	2	2	4
2	3	7	10	1	2	3
3	1	5	6	1	5	6
Total	7	21	28	4	9	13

After a passing through the first iteration of Algorithm 2, by using SINGULAR software package[12], one obtains an equivalent generator matrix G', shown in Eq.(1) which requires only 13 delay elements for realization using canonical form. The subsequent iteration results in the same generator matrix G' and so, the algorithm converges to G'.

$$\begin{aligned}
G' &= UG \\
&= \begin{bmatrix} 1+D_1^2 & 1+D_1 & D_2 & D_1 + D_2^2 \\ D_1 D_2 & 0 & 1 & D_1+D_2 \\ 1 & D_1 & D_2+D_1 D_2^3 & D_1 + D_2^2 \end{bmatrix}, \quad (1)
\end{aligned}$$

where

$$U = \begin{bmatrix} 1 & 0 & D_1^2 D_2 \\ 1 + D_1 & 1 & D_2 \\ 0 & 0 & 1 \end{bmatrix},$$

is a unimodular matrix i.e. det $U = 1$. The realization of the 2-D convolutional encoder with the generator matrix G' is shown in Figure 2.

III. Conclusion

It has been shown that given a generator matrix of a $m-$D convolutional code, one can effectively reduces the row-wise maximum total degree to yield an equivalent generator matrix whose realization requires fewer number of delay elements. The proposed algorithm (Algorithm 2) is based on the application of Gröbner basis theory and row-reduction by unimodular matrix technique.

The algorithm proposed here is applicable with large generator matrices of any number of dimensions (with straightforward generalization). The algorithm will generally yield an optimal solution, however, since the relationship between the number of delay elements and the row-wise maximum total degree is not linear, there could rarely exist a generator matrix with a larger row-wise maximum total degree that requires fewer number of delay elements for implementation.

The applicability of Algorithm 2 is not only limited to the $m-$convolutional code area. It could potentially become, by minor modification, a base-line algorithm to reduce the complexity of the realization of a multi-input/multi-output multidimensional linear system in general. Current study along this direction is undertaken.

IV. Acknowledgement

This research work is supported by The Thai Research Fund (TRG 4580063). The authors thanks the reviewers for their time and constructive comments.

References

[1] W.W. Adams and P. Loustaunau, *"An Introduction to Gröbner Bases,"*, Graduate Studies in Mathematics, vol.3, American Mathematical Society, 1994.

[2] K.J. Boo and N.K. Bose, "Multispectral image restoration with multisensors," IEEE Trans on Geoscience and Remote Sensing, vol. 335B, 1997, pp. 1367-1409.

[3] B. Buchberger, "Gröbner Bases: An algorithmic method in polynomial ideal theory," *in Multidimensional System Theory (N.K. Bose, ed.)*, Reidel, Dordrecht, 1985, pp. 184-232.

[4] C. Charoenlarpnopparut and N.K. Bose, "Multidimensional filter bank design using Groebner bases," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 46, no. 12, December 1999, pp. 1475-1486.

[5] C. Charoenlarpnopparut and N.K. Bose, "Groebner Bases For Problem Solving in Multidimensional Systems," *Multidimensional Systems and Signal Processing,* vol. 12, no. 3-4, 2001, pp. 365-376.

[6] C. Charoenlarpnopparut, S. Wongsura, and A. Davies, "Direct realization of a 2-D convolutional encoder with fewer number of delay elements." Proceedings of the Third International Symposium on Communications and Information Technology (ISCIT 2003), 3-5 September 2003, Songkhla, Thailand, Vol. I, pp. 258-262.

[7] C. Charoenlarpnopparut and S. Tantaratana, "Multidimensional convolutional code: Progresses and bottlenecks," Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003), 25-28 May 2003, Bangkok, Thailand, pp. III-686 to III-689.

[8] C. Charoenlarpnopparut and S. Tantaratana, "Application of Gröbner Bases on the Algebraic Structure of Multidimensional Convolutional Code," submitted for publication.

[9] D. Cox, J. Little and D. O'Shea, *"IDEALS, VARIETIES, AND ALGORITHM: An Introduction to Computational Algebraic Geometry and Commutative Algebra,"* 2nd edition, Springer-Verlag, New York, 1996.

[10] E. Fornasini and M.E. Valcher, "Algebraic aspect of 2D convolutional codes," *IEEE Trans. Inform. Theory,* vol. IT-40, No. 4, 1994, pp.1068-1082.

[11] G.D. Forney, Jr, "Convolutional codes I: Algebraic structure," *IEEE Trans. Info. Theory,* vol. IT-16, Nov. 1970, pp. 720-738.

[12] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 2.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2001). http://www.singular.uni-kl.de.

[13] R. Johannesson and Z. Wan, "A linear algebra appoach to minimal convolutional encoders," *IEEE Trans. Information Theory,* vol. 39, no. 4, Jul. 1993, pp. 1219-1233.

[14] R. Johannesson and K.S. Zigangirov,"Fundamentals of Convolutional Coding,", IEEE Press, New York, USA, 1998. ISBN:0-7803-3483-3.

[15] S. Mahapakulchai, "MAP source-controlled channel decoding for image transmission using CPFSK and ring convolutional codes," *Ph.D. Dissertation,* Pennsylvania State University, PA, USA, 2004.

[16] M.E. Valcher and E. Fornasini,"On 2-D finite support convolutional codes: An algebraic approach," *Multidimensional Systems and Signal Processing,"* Vol. 5, No. 3, July 1994, pp. 231-243.

[17] P.A. Wiener,"Multidimensional Convolutional Codes," *Ph.D. Dissertation,* University of Notre Dame, IN, USA, 1998.

Application of Gröbner Bases on the Algebraic Structure of Multidimensional Convolutional Code [1]

C. Charoenlarpnopparut * S. Tantaratana

P.O. Box 22, Thammasat University Rangsit Post Office, Telecommunications Program, Sirindhorn International Institute of Technology, Thammasat University, Klongluang, Pathumthani, 12121, THAILAND. Phone:(+662)501-3501 Ext. 1808 Fax:(+662)501-3501 Ext. 1801

Abstract

Over the past few years, the multidimensional convolutional code has become a new area of research in signal processing community. While the one-dimensional convolutional code and its variants have been thoroughly understood, the m-D counterpart still lacks unified notation and efficient encoding/decoding implementation. Here, the strong link between the theory of Gröbner basis and m-D convolutional code are explored. Several applications of Gröbner bases to the characterization of m-D convolutional encoders are proposed. Furthermore, the more practical problem of minimal encoder realization is discussed and partial solution to the problem is provided. From the implementation point of view, the syndrome decoder is currently the only mean for decoding the m-D convolutional code. Based on the usage of the theory of syzygy module, the constructive method for computing the syndrome decoding matrix is proposed.

Key words: Multidimensional convolutional code, Gröbner basis, minimal encoder, syndrome decoding, syzygy, unimodular matrix

* Corresponding Author.
Email addresses: `chalie@siit.tu.ac.th` (C. Charoenlarpnopparut), `sawasd@siit.tu.ac.th` (S. Tantaratana).

[1] This research work is supported by The Thai Research Fund (Contract number TRG 4580063)

1 Introduction

Examples of 1-D information are voice signal, audio signal, node voltage, line current. By sampling these signals and quantizing the samples, one obtain discrete sequences of binary representation. On the other hand, multidimensional (m-D, $m \geq 2$) information may be taken from 2-D images ($m = 2$), e.g. MRI film, photos, 3-D images ($m = 3$), e.g. holograms, animation of 2-D images, e.g. video, movies ($m = 3$) and animated 3-D images ($m = 4$). To encode multidimensional information, it is conventional to first transform m-D data into 1-D sequence by means of scanning, and then apply the 1-D encoding technique. This procedure is, in fact, unnatural and ignores the correlation in all other direction except in the scanning direction.

To motivate the importance of multidimensional convolutional code, note that 1-D convolutional code is a generalization of a block code, i.e. block code is a 1-D convolutional code with the delay variable D replaced by zero. Equivalently, 0-D convolutional code is essentially a block code. As a result a multidimensional convolutional code can be viewed as a generalization of all linear codes.

While the one-dimensional convolutional code and its variants have been thoroughly understood [11,12,15–18], the m-D counterpart still lacks unified notation and efficient implementation. In this document, the aim is to emphasize the recent development [10,20,21] of m-D convolutional code theory as well as to point out the bottleneck as well as an open problem to the multidimensional community.

One of the goals here is to demonstrate the usefulness of the theory of Gröbner basis in the field of convolutional code which has not been done elsewhere. The theory of Gröbner basis was first introduced in 1967 by Bruno Buchberger in his doctoral dissertation and the name Gröbner was used to honor his thesis advisor. Since the first introduction, the theory of Gröbner basis has been widely applied in a wide range of engineering problems e.g. in the field of computer algebra for solving multivariate polynomial equation, integer programming problem, in the system and control area for solving controller design problem, and in the signal processing area for solving $m-$D filter bank design and ladder decomposition problems.

This paper is organized in the following fashion. First the basic definition and convention in m-D system theory are given and briefly explained. In the followed section, the properties of $m-$D convolutional code and its encoder are defined. Then, by using the theory of Gröbner basis, the algorithms for testing the existence and characterizing those properties are presented. It will be evident that the application of Gröbner bases and its variants will ultimately define a strong link between the $m-$D convolutional code and system theory.

Since the encoder matrix of a 1-D convolutional code can be represented by a rectangular univariate polynomial matrix, in the $m-$D case, the representation of encoder matrix becomes the multivariate polynomial matrix, where the theory of Gröbner basis for module[1] can be readily applicable. Each row of this generator matrix can be realized (implemented in the form of hardware) separately and thus it has the row independent property.

In the later part of the paper, the problem of minimal realization [7,8] is considered and the partial solution is provided. Various algorithms related to counting the number of delay elements are given. In the last section, the decoding process of multidimensional code by means of syndrome decoding is discussed and with the help of Gröbner basis the construction procedure of the decoding matrix is explained.

1.1 Representation of multidimensional signals

In general, there are two approaches to represent multidimensional signals:

- Geometrical Approach: an m-D signal is represented by a m-D finite matrix, for example a 3×3 pixel image I with 16 gray-scale levels (4 bits) can be represented as

$$I = \begin{bmatrix} 1100\ 0101\ 1001 \\ 1000\ 1001\ 0000 \\ 1011\ 0100\ 0010 \end{bmatrix} \tag{1}$$

- Algebraical Approach: an m-D signal is represented by a m-variate polynomial vector of size $1 \times b$, where b is the number of symbols used for representing each sample, for example the 2-D image in the previous example $(b = 4, m = 2)$ may be represented by

$$u = \begin{bmatrix} 1 + D_2 + D_1^2 + D_1D_2 + D_2^2 \\ 1 + D_1 + D_1D_2^2 \\ D_2^2 + D_1^2D_2^2 \\ D_1 + D_1^2 + D_1D_2 + D_2^2 \end{bmatrix}^T \tag{2}$$

Note that in the above example, the i^{th}-row j^{th}-column element of the matrix I in Eq.(1) is associated with the multiplying factor $D_1^{j-1}D_2^{i-1}$ and the p^{th} element of the vector u in Eq.(2) is taken from the summation of all p^{th} symbols, of all elements of the matrix I, multiplying by their associated monomials.

2 Convolutional Encoder

2.1 Notation and definition

Let $\mathbb{F} = \mathbb{F}_q$ be the finite field with q elements and $R = v[D_1, D_2, \ldots, D_m]$ be the ring of m-variate polynomials whose coefficients belong to the field $\mathbb{F}$. Let $n \geq 1$ be a positive integer, then the free R-module [1, p.114], R^n, is the m-variate polynomial row vector space of length n over $\mathbb{F}$, i.e.

$$R^n = \left\{ \begin{bmatrix} v_1 \; v_2 \cdots \; v_n \end{bmatrix} | v_i \in R, i = 1, 2, \ldots, n \right\}$$

Definition 1: An m−dimensional convolutional code of length n over $\mathbb{F}$ is an R-submodule $C \subseteq R^n$. An element $w \in C$ is called a *codeword.*

Definition 2[20]: A code is called a *modular code* if it is closed under vector addition and under m-variate polynomial multiplication.

Fact 1[21]: Every m-dimensional convolutional code is finitely generated as an R-module, i.e. there exists a finite set of row vectors $g_1, g_2, \ldots, g_k \in R^n$ such that

$$C = \left\{ \Sigma_{i=1}^{k} u_i g_i | u_i \in R \right\}.$$

2.2 Generator matrix

Similar to 1-D case, a generator matrix of a m-D convolutional code can be viewed as a transfer function matrix of the encoder and it often is a center of focus in the encoder design and realization stage.

Definition 3: Let G be an $k \times n (k \leq n)$ m-variate polynomial matrix constructed row-wise from a set of row vectors $\{g_i\}_{i=1}^{k}$, i.e.

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix}.$$

If an m-D convolutional code $C = \text{rowspace}(G)$, the polynomial matrix, $G \in R^{k \times n}$, is called a *generator matrix* of C. The existence of G is guaranteed since any m-D convolutional code is a finitely generated R-module.

Furthermore if G is of full row rank (i.e. $\text{rank}(G) = k$), then the rate of C is $\frac{k}{n}$. Note that it is desirable for a generator matrix to have full row rank in order to avoid having different input sequences mapped to the same codeword (undistinguishable by the decoder which message is being sent).

Definition 4: If a code C is free of rate $\frac{k}{n}$, then $C = \text{rowspace}(G)$ for some $G \in R^{k\times n}$ and G is called an *encoder* of the code C

It is well-known that an R-module can be generated by many different sets of module generators. Analogously, a given m-D convolutional code C can be generated by more than one encode. This class of generators that generate the same set of code is defined as a class of *equivalent* generators

Definition 5: An m-variate polynomial matrix $U \in R^{k\times k}$ is *unimodular* if U is a unit in $R^{k\times k}$. In other words, U is unimodular if and only if $\det(U)$ is a unit in R (i.e. a nonzero element of $\mathbb{F}$).

Proposition 1[21]: Let G, $G_1 \in R^{n\times k}$ be of full row rank. G and G_1 are equivalent encoders if and only if there exists a unimodular matrix $U \in R^{k\times k}$ such that $U \cdot G = G_1$.

From Proposition 1, given two generator matrices G and $G_1 \in R^{n\times k}$, the following algorithm can be used to determine whether they are equivalent. Furthermore, it can be used to compute the unimodular matrix $U \in R^{k\times k}$ such that $U \cdot G = G_1$.

Algorithm 1: Equivalence test

- *Step 1.* Construct a module M generated by the columns of matrix G.
- *Step 2.* Compute the Gröbner basis module of M with respect to any ordering by using Buchberger's algorithm for module[1, p.149]
- *Step 3.* Use the division algorithm [1, p.145] to identify if all columns of G_1 are members of the Gröbner basis module. If not, conclude that G and G_1 are not equivalent. If so, G and G_1 are equivalent, and the unimodular matrix U can be computed by retracing the steps of Buchberger's algorithm (see example 3.6.1 in [1, pp.153-154]).

Definition 6: A convolutional generator matrix G of size $k \times n, k \leq n$ is called *basic* if it is polynomial and it has a polynomial right inverse.

It is equivalent to say that G is basic if G is minor left prime (MLP) i.e. all $k \times k$ minors of G are relatively prime. The theory of Gröbner basis can be applied here to compute a right inverse of a polynomial generator matrix G. A more general algorithm and examples to compute the whole class of right inverse can be found in [5]

Algorithm 2: Computing the right inverse of a given generator matrix

- *Step 1.* Let M be a module generated by the columns of matrix G.
- *Step 2.* Compute the Gröbner basis module of M with respect to any ordering by using Buchberger's algorithm for module[1, p.149]
- *Step 3.* Use the division algorithm [1, p.145] to determine if all column vectors of the identity matrix I_k are the membership of the module M. If so, the right inverse G^{-1} can be computed by retracing the steps of Buchberger's algorithm.

3 Realization of Convolutional Encoder

The implementation of multidimensional convolutional code requires the usage of various dimensional delays (shift registers), $D_1, D_2, \ldots$ where an example of the canonical form realization is shown in Figure 1. In this section, the focus is on the minimal realization of a 2-D convolutional code. The higher dimension cases can be tackled in the same fashion.

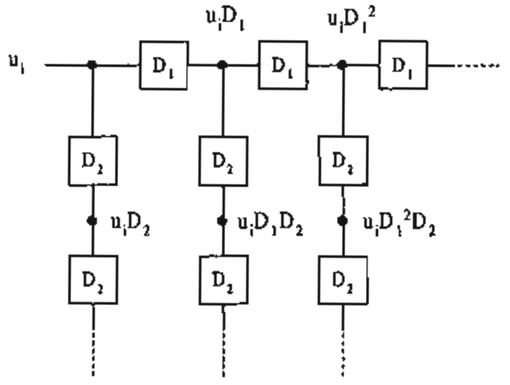

Fig. 1. Example of a 2-D shift register array where u_i is the input and D_1, D_2 are horizontal and vertical delay respectively.

To realize the encoding process efficiently in the form of hardware, it is necessary to minimize the number of delays needed. In the 1-D case, one objective is to single out the generator matrix, from all equivalent ones, that has the minimum overall constraint length. However, a meaningful definition of constraint length, in a higher ($m \geq 2$) dimension case, has not been given anywhere. A good candidate for this parameter should directly indicate the total number of delays in the implementation of a desired encoder. Given a generator matrix $G \in R^{k \times n}$, identify, among all equivalent generator matrices, the one whose realization uses the minimum number of delay elements. A practical procedure towards the solution of this problem is only possible by direct searches via computer programming.

One approach to solve the problem stated is to perform a search over the space of all possible equivalent generator matrices. Since an equivalent generator matrix can be generated by multiplying a unimodular matrix to the given generator matrix, there are infinite number of elements to search from. As a result, a finite scope for searching must be set in such a way that the minimal basic encoder should be found most of the time. Here, we limit the searching space by limiting the total degree of each element in the unimodular matrix to be less than a certain integer, chosen according to the available search time.

3.1 Number of delay elements

Let u be the 2-D uncoded input sequence expressed in the polynomial vector form [7,21], G be the generator matrix. In the polynomial vector form, the encoded codeword v, then, can be expressed as

$$v = u \cdot G. \tag{3}$$

Since there is only one kind of delay in one-dimensional convolutional encoder realization, it is straight forward to count the total delays [14]. However, in 2-D, the counting procedure is nontrivial as illustrated by the following example.

Example 1: Let

$$G_1 = [1 + D_2^2 \;\; D_1D_2^2 \;\; D_1^2]. \tag{4}$$

It is not so obvious that two D_1's and four D_2's are required. A realization diagram in Fig. 2 can clearly indicate how many delays are needed. Given

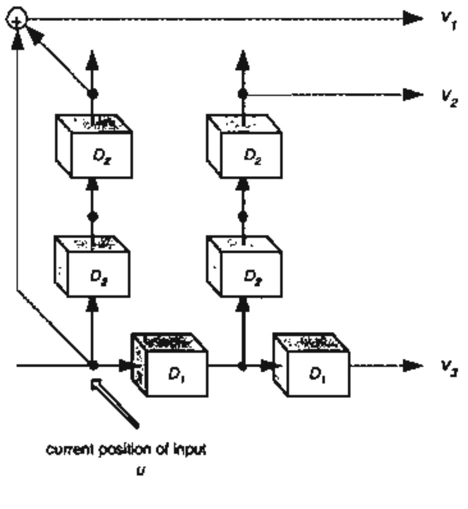

Fig. 2. Direct realization for G_1 (4)

a realization diagram, the total number of the delays required can be easily counted. However, one of the goals here is to find an algorithm for quickly counting the number of delays needed without drawing the diagram.

3.2 Counting delays for a monomial G

For generator matrices of size 1×1 whose element is also a monomial, the total number of delays can be easily counted from the generator matrix itself.

Example 2: Take a look at the following generating matrix:

$$G_2 = [D_1^2 D_2^2]. \tag{5}$$

To realize G_2, two D_1's and two D_2's are required. In other words, the total number of delays required are the same as the total degree of the monomial.

3.3 Counting delays for a polynomial row vector G

As the generator matrix becomes larger and more complex, the number of delays required are not so obvious.

Example 3: Let G_3 be a generator matrix of size 1×2.

$$G_3 = [D_1^2 D_2 + 1 \quad D_1 D_2^2]. \tag{6}$$

The realization diagram of G in Eq.(6) is given as

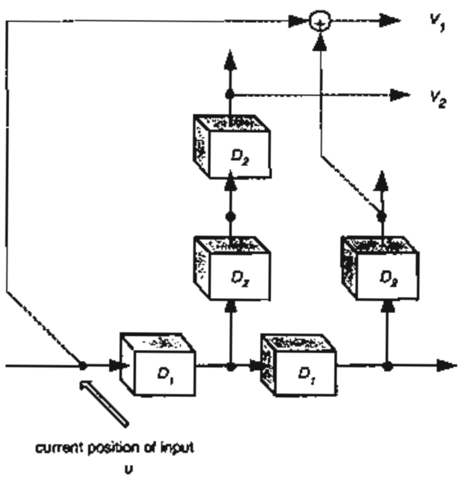

Fig. 3. Direct realization for G_3

From the diagram in Fig. 3, it is clear that to realize G_3, two D_1 and three D_2 are required.

The following algorithm can be used to find the total number of delays for a polynomial row vector G without drawing a realization diagram.

Algorithm 3: Counting delays for a polynomial row vector G

- *Step 1.* Let $G(1,j)$ be the polynomial in j^{th} column. First, to find out the number of D_1, compare all polynomials in G and find D_1 with the highest degree (treating D_2 as constant). The number of D_1 required is equal to the maximum degree of all polynomials, i.e.

$$del_{D_1}(G) = \overset{max}{j} \; \{deg_{D_1} G(1,j)\} \tag{7}$$

- *Step 2.* Next, the number of delays, D_2, is the sum over r_1 of all the maximum degrees r_2 of monomials in the form $D_1^{r_1} D_2^{r_2}$ of all $G(1,j), r_1 = 0, 1, 2, \ldots$.
- *Step 3.* Finally, the total number of delays needed can be found by simply summing up all the delays, i.e.

$$del(G) = del_{D_1}(G) + del_{D_2}(G). \tag{8}$$

Example 4: Let

$$G = [D_2 \;\; D_1D_2 \;\; 1 + D_1^2 D_2]. \tag{9}$$

Step 1.

$$\begin{aligned} deg_{D_1} G(1,1) &= 0, \\ deg_{D_1} G(1,2) &= 1, \\ deg_{D_1} G(1,3) &= 2. \end{aligned}$$

Therefore, $del_{D_1}(G) = \overset{max}{j} \; \{deg_{D_1} G(1,j)\} = 2$.
Step 2. For $r_1 = 0, G(1,1) = D_1^0 D_2^1$, so one D_2 is needed. For $r_1 = 1, G(2,1) = D_1^1 D_2^1$, so another D_2 is needed. Finally, $r_1 = 2, G(1,2) = 1 + D_1^2 D_2$, so another D_2 is needed. Therefore, $del_{D_2}(G) = 1 + 1 + 1 = 3$ and $del(G) = 5$.

Note that the presence of '1' in the elements of G is completely irrelevant to the total number of delays.

3.4 Counting delays for a polynomial matrix

To facilitate the understanding of the reader, the algorithm will be presented in parallel with an example.

Example 5: Let

$$G = \begin{bmatrix} 1 + D_1 & D_1 D_2 & D_1^2 \\ D_2^2 & 1 & 1 + D_2 \end{bmatrix}. \tag{10}$$

Algorithm 4: Counting total number of delays for a given generator matrix G

Step 1. Count one row at a time using Step 1 of Algorithm 3. The total number of delays D_1 needed is

$$deg_{D_1} G = \sum_i \overset{max}{j} \{deg_{D_1} G(i,j)\}.$$

In this example, $G(1,3) = D_1^2$ and there is no D_1 in the second row. Therefore, only two D_1's are needed.

Step 2. Count the number of D_2's row-wise and add them up, by using Step 2 in Algorithm 3.

In the example, in the first row the number of D_2 delays required is only one due to the term D_1D_2. In the second row, the number of D_2 delays required is two due to the monomials D_2^2. So, three delays D_2 are required for implementing G.

Step 3. Compute the total number of delays required by adding results from two previous steps.

For the generator matrix G in Eq(10), five (two D_1's and three D_2's) delays are needed.

The above algorithm as well as the algorithm for vector are simply derived from observation of the pattern through many worked out mapping to the diagram. Note that the second algorithm for matrix is quite similar to the preceding algorithm for vector.

Alternatively, the above algorithm may be reconsidered by using the notation of support which can directly used for programming in a computer.

3.5 Support of a generator matrix

Support, as the name suggests, could be thought of as a foundation of a building's columns, i.e., if a building is to be removed, the remainder of the building would show its supporting foundation. For the purpose of illustration, Eq.(9) defines a map onto a support matrix in Fig. 4.

For the purpose of mathematical representation and computation, the support will be represented in a matrix form. Each row of G can be represented by one support matrix. The i^{th}-row and j^{th}-column of support matrix corresponds to the presence of the monomial $D_1^i D_2^j, i, j = 0, 1, 2, \ldots$ in the row of generator matrix G. To avoid confusion, elements of G are generally polynomials, each

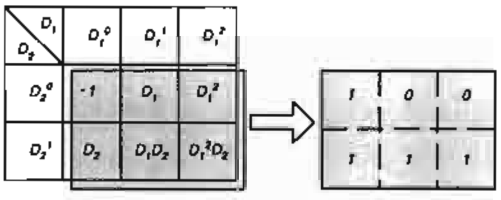

Fig. 4. Mapping of a generator encoding matrix onto a support matrix using Eq.(9).

of which consists of the sum of one or more monomials.

The rest of the matrix will be filled with 0's. For example, the support of Eq.(9) is given as Eq.(11).

$$S = \begin{bmatrix} 1\,0\,0 \\ 1\,1\,1 \end{bmatrix} \tag{11}$$

The next algorithm summarize the method to generate the support of a generator matrix.

Algorithm 5: Construction of a support

Again, the generator matrix in Eq.(9) will be used to illustrate the method.

- For each monomial in the i^{th} row G, put a '1' in the support matrix, S_i, in the associated position. For example, $G(1,3) = 1 + D_1^2 D_2$. Therefore, $S(0,0) := 1$ because of the monomial $1 = D_1^0 D_2^0$ in $G(1,3)$ and $S(1,0) := 1$ because of the D_2 in $G(1,1)$.
- If the same monomial is encountered more than once, then the later found monomials are ignored. The presence of the second monomial does *not* cancel any entries in the support.
- Finally, the size of the matrix can be checked for verification. The number of columns is equal to the $\overset{max}{i,j}\ \{deg_{D_1} G(i,j)\} + 1$ and the number of rows is $\overset{max}{i,j}\ \{deg_{D_2} G(i,j)\} + 1$.
- Note that for a given generator matrix G, there are as many support matrices S as the number of rows in G.

Example 6: Let

$$G = \begin{bmatrix} 1 + D_1 D_2 + D_1^3 & D_1 + D_2^2 \\ D_1^3 + D_2 + D_1 D_2^3 & 1 + D_1^2 + D_1^2 D2^2 \end{bmatrix}. \tag{12}$$

The supports of the (12) are

$$S_1 = \begin{bmatrix} 1\,1\,0\,1 \\ 0\,1\,0\,0 \\ 1\,0\,0\,0 \end{bmatrix} \text{ and } S_2 = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \end{bmatrix}.$$

Algorithm 6: Counting delays for a generator matrix using support

Step 1. Every row of the $n \times k$ generator matrix, G, is first represented by the corresponding support matrix, S_i, for $i = 1$ *to* n.
Step 2. Now, for each support, let the size of the support be $p \times q$.; Then,

$$del_{D_1}(g_i) = q - 1, \tag{13}$$

where g_i is the i^{th}-row of the generator matrix G.
Step 3. Remove the first row of S_i, and thereby making it a $(p-1) \times q$ matrix.
Step 4. This is the important step. Consider one column of S_i at a time and identify the row number (the top row is counted as row number 1) of the last row (of that column) that contains '1'. Add these numbers up for all columns. The result is the number, $del_{D_2}(g_i)$ of delays D_2 required.
Step 5. Finally, to compute the total number of delays for G, add up each $del_{D_1}(G_i)$ for number of D_1's and $del_{D_2}(G_i)$ for that of D_1's to form total number of delays i.e.

$$N = \sum_{i=1}^{n} del_{D_1}(g_i) + del_{D_2}(g_i).$$

3.6 Minimal Encoder

In this subsection, let first define the term **minimal** and propose two approaches to produce the minimal encoder. It is well know that a given m-D convolutional code can be generated by many equivalent encoders, each of which is characterized by a generator matrix. The class of equivalent generator matrices can be parameterized by unimodular multiplication i.e.

$$\mathcal{S}(G_0) = \left\{ G \in R^{k\times n} \middle| \; G = UG_0, U \text{ is a unimodular matrix and } G_0 \in R^{k\times n} \right\}.$$

Here, a unimodular matrix is defined as a square multivariate polynomial matrix whose determinant is a unit element in the coefficient field $\mathbb{F}$, i.e., one.

$$U \text{ is a unimodular matrix} \longleftrightarrow |U| = \det(U) = 1. \tag{14}$$

Examples of unimodular matrices are:

$$U_1 = \begin{bmatrix} D_1D_2+1 & D_1 \\ D_2 & 1 \end{bmatrix}, U_2 = \begin{bmatrix} D_1 & 1 \\ 1+D_1^2+D_1D_2 & D_1+D_2 \end{bmatrix}. \tag{15}$$

Given a generator matrix, to find all possible equivalent generator matrices, as many unimodular matrices must first be found. That is, given $G_0 \in R^{k\times n}$, to find G which is equivalent to G_0, one needs to construct $U \in R^{k\times k}$. One way to find as many as possible U for a particular $R^{k\times k}$ is to consider a *sequential* search through as many polynomial matrices as possible. Unimodular matrices may be generated by systematically arranging possible combinations of polynomial entries of the matrix.

First, the given generator matrix is entered as an input. Then its number of delays required is counted and stored. Next, multiply the given generator matrix by a stored unimodular matrix to produce a different equivalent generator matrix$G$. Then the number of delays for this new equivalent matrices is computed and compared with the original number. Whichever G that produces smaller number of delays will be stored. The process is repeated with new equivalent matrices until all unimodular matrices in storage are used. Finally, the stored generator matrix G is the one that required the smallest number of delays found. It is quite obvious that the search result depends greatly on the searching space. Since theoretically the searching space is infinitely uncountable, the minimal encoder is not guaranteed to be found. However, from the practical point of view, one can start the search by generating a large storage of unimodular matrices. From our experimental testing, we found that this approach does provide a quick and suboptimal result, "most of the time". Next, the detail implementation is explained.

Consider the two-dimensional case. Let

$$U = \begin{bmatrix} P_1 & P_2 \\ P_3 & P_4 \end{bmatrix}, \tag{16}$$

where $P_i, i = 1,2,3,4$ is a 2-D polynomial. Now, the goal is to find a set of $\{P_i\}_{i=1}^4$ such that U becomes a unimodular matrix. For instance, P_1 may be in the form of

$$\begin{aligned} P_1 = & a_1 + a_2D_1 + a_3D_1^2 + a_4D_2 + a_5D_1D_2 + a_6D_1^2D_2 + a_7D_2^2 + a_8D_1D_2^2 \\ & + a_9D_1^2D_2^2, \end{aligned}$$

where $a_i \in \{0,1\}$. The concept of support is particularly helpful to visualize this polynomial P_1. P_1 can be written in the form of a support as

$$S_{P_1} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}. \tag{17}$$

Similar treatment can be applied for other $P_k, k = 2, 3, 4$ with a_i replaced by b_i, c_i, and d_i respectively.

An example is given to illustrate how to generate minimal delay encoding matrix. The example was worked out with help of a computer software to generate all possible 2×2 unimodular matrices with maximum degree of each element less than or equal to $(2, 2)$.

Example 7: Given,

$$G = \begin{bmatrix} D_1D_2 & D_2^2 & 1 + D_1^2 \\ D_1 + D_1D_2^2 & D_2 + D_2^3 & 1 + D_1 + D_1^2D_2 \end{bmatrix}. \tag{18}$$

This requires four D_1's and nine D_2's. Let the input of the function be support of each row of (18). After the search, the minimal delay generator matrix of (18) will be

$$G_{min} = \begin{bmatrix} D_1 & D_2 & 1 + D_1 \\ D_1D_2 & D_2^2 & D_1^2 \end{bmatrix}, \tag{19}$$

which requires three D_1's and only four D_2's and the unimodular matrix U_{min} used to produce G_{min} is

$$U_{min} = \begin{bmatrix} D_2 & 1 \\ 1 & 0 \end{bmatrix}.$$

In this sequential search approach, the performance of the search is not optimal and largely depends on the size of searching space(i.e. the number of unimodular matrices used). Furthermore, for a large generator matrix, the sequential approach will suffer from the higher complexity. The general problem to find the optimal solution is still open. The author is working on the algebraic approach to solve this problem and the result will be reported once it is completed.

4 Encoding procedure

In this section, a 2-D convolutional encoding process is illustrated by a given example.

Let us consider a 2-D convolutional code, of rate $\frac{2}{3}$ whose generator matrix is given by

$$G = \begin{bmatrix} 1 & D_1 & D_1D_2 \\ 0 & D_2 & D_1+1 \end{bmatrix},$$

and let I be the 2-D binary input taken from a 4-color image with the size of 3×3 pixels, i.e.

$$I = \begin{bmatrix} 01 & 10 & 10 \\ 11 & 00 & 00 \\ 01 & 10 & 01 \end{bmatrix},$$

whose algebraic representation is given as,

$$u = \begin{bmatrix} D_1+D_2+D_1^2+D_1D_2^2 & 1+D_2+D_2^2+D_1^2D_2^2 \end{bmatrix}.$$

The output v of an encoder G can be obtained by vector matrix multiplication,

$$\begin{aligned} v &= u \cdot G \\ &= \begin{bmatrix} D_1+D_2+D_1^2+D_1D_2^2 \\ \begin{pmatrix} D_2+D_1^2+D_1D_2+D_2^2+ \\ D_1^3+D_2^3+D_1^2D_2^2+D_1^2D_2^3 \end{pmatrix} \\ \begin{pmatrix} 1+D_1+D_2+D_1D_2+D_2^2+D_1^2D_2+ \\ D_1^2D_2^2+D_1^3D_2+D_1^3D_2^2+D_1^2D_2^3 \end{pmatrix} \end{bmatrix}^T. \end{aligned}$$

The polynomial output vector y can be represented in the geometrical form as

$$I_{out} = \begin{bmatrix} 001 & 101 & 110 & 010 \\ 111 & 011 & 001 & 001 \\ 011 & 100 & 011 & 001 \\ 010 & 000 & 011 & 000 \end{bmatrix}.$$

Observation 1: If k is the number of symbols(bits) of each input pixel, and the generator matrix is of size $k \times n$, the number of symbols(bits) of each encoded pixel is n and independent of the number of input pixels. On the other hand, the size of the encoded image (2-D array) depends directly on the row-wise maximal total degree of the generator matrix.

Observation 2: In the case when the input is of large size (e.g. 256×256 pixels), the total degree of the input vector would grow significantly. However, this is not a problem in the implementation stage since the realization of the encoder is done by using an array shift register, not polynomial multiplication.

Observation 3: One assumption used here is that the number of rows of the generator matrix G must be the same as the number of symbols (bits) representing each pixel. When this is not the case, one needs to regroup the m-D data symbols to suit the encoder matrix.

5 Syndrome decoder and dual code

The decoding of m-D convolutional code can be performed in a straightforward manner by means of syndrome decoding. However, unlike the soft-decision scheme, e.g. Viterbi decoding in 1-D, this hard-decision decoding method may not be optimal in the maximum-likelihood (ML) sense nor the maximum apriori (MAP) sense [18]. In this section, the basic definition and existence criteria of syndrome decoder are given along with the algorithm for computing the syndrome decoder matrix.

Definition 7: Let C be an R-submodule of the free R-module, R^n. An *image representation* of C is a matrix $G \in R^{l \times n}$ such that $C = \text{rowspace}(G)$. A *kernel representation* of C is a matrix $H \in R^{n \times p}$, where p is some positive integer, such that $C = \ker(H) = \{w \in R^n | w \cdot H = 0\}$.

If a sequence $w \in C$, then $w = u \cdot G$ for some $u \in R^k$, and so $w \cdot H = u \cdot G \cdot H = 0, \ \forall\, u$. This implies $GH = \mathbf{0}$.

Given an m-D convolutional code, C with the associated generator matrix G, the kernel representation (i.e. the matrix H), if exists, can be constructed by computing the syzygy [1, pp.161-168] of the module generated by all columns of the generator matrix G.

Example 8: Let G be the minor left prime matrix

$$G = \begin{bmatrix} 1 & D_1 & D_1 D_2 \\ 0 & D_2 & D_1 + 1 \end{bmatrix}.$$

The kernel representation of G is computed by using Singular program [13] command lines:

```
>ring r=2,(x,y),(c,dp);
>module M=[1,0],[x,y],[xy,x+1];
>module H=syz(M);
>H;
H[1]=[x2+xy2+x,x+1,y]
```

to obtain

$$H = \begin{bmatrix} D_1 + D_1^2 + D_1 D_2^2 & D_1 + 1 & D_2 \end{bmatrix}^T .$$

By the properties of the matrix H and syzygy, all codewords in the set $C =$ rowspace(G) satisfy the condition:

$$w \cdot H = 0, \quad \forall\, w \in C.$$

This condition provides a *parity check,* which can be applied to a received sequence for testing whether it is a codeword. The matrix H is often called the *parity check matrix*, or the *syndrome decoder* of C and the corresponding codes are called *finite convolutional code.* Note that not every convolutional code admit a syndrome decoder (i.e. the parity check matrices of some m-D convolutional codes do not exist). The following proposition characterizes the existence of a syndrome decoder.

Proposition 3 [20,21]: A free modular code C admits a syndrome decoder if and only if C has a **minor prime** generator matrix G.

(An m-variate polynomial matrix G is minor prime[4], if all maximum-size minors (major determinants) are devoid of common factor other than units.) For a Laurent polynomial ring, the existence condition[20] is relaxed to the condition that the generator matrix must be left factor prime.

Example 9: Consider a 3-D convolutional code C with a corresponding generator matrix

$$G = \begin{bmatrix} D_1 & 0 & D_2 \\ 0 & D_1 & D_3 \end{bmatrix} .$$

The major determinants of G contain a common factor of D_1. As a result, G is not minor prime. In [21, p.16], it has been shown that G is in fact left factor prime. By using Singular program, $H = \begin{bmatrix} D_2 & D_3 & D_1 \end{bmatrix}^T$ is the syzygy of the module generated by all columns of G. However, this matrix H is not a syndrome decoder of C since $v = \begin{bmatrix} D_3 & D_2 & 0 \end{bmatrix}$ satisfies $v \cdot H = 0$, but it does not belong to the set C (i.e. there does not exist a row vector $u \in R^2$ that produces $u \cdot G = \begin{bmatrix} D_3 & D_2 & 0 \end{bmatrix}$.)

Definition 8: The *orthogonal module* of M is denoted by $M^\perp$ and is given as

$$M^\perp \triangleq \left\{ w \in R^n | wv^t = 0, \; \forall v \in M \right\}.$$

For an m-D convolutional code C, the *orthogonal code* of C is denoted by the module $C^\perp$. In the case where the C admits a syndrome decoder (i.e. has a kernel representation), the orthogonal code $C^\perp$ is called the *dual code* of C.

6 Conclusion

It has been shown that there exists a strong relationship between the theories of multidimensional system and those of m-D convolutional code. In the system point of view, the focus is on the input and output relationship and how the system responds to a different input. On the other hand, the coding side only concentrates on the set of output (codewords) and its properties. The set of all m-D convolutional codewords generated by a generator matrix G can be modelled as a R-submodule, where many mathematical tools such as Gröbner basis, module theory and algebraic geometry can be applied.

The minimal encoder realization is explored and the algorithm for finding a suboptimal solution is given. Although the solution obtained from the sequential search may not be minimal in some cases, with the current computing technology, it is possible to extent the searching space so that the optimal solution is warranted every time. Another algorithm for computing the minimal encoder based on some algebraical theories is in the developing process and will be reported in a near future.

There are still many areas of m-D convolutional which are rather unexplore, e.g. the design of m-D convolutional code, performance measurement, soft decoding. There has been a study [21] on the weight and distance bound of m-D convolutional code which can be further investigated and may lead to the completion of optimal m-D convolutional code design procedure. Some progress has been made for special class of m-D convolutional code design, namely the unit memory codes (the ones whose generator matrix contains only first or zero degree polynomials.

7 Acknowledgement

This research work is supported by The Thai Research Fund (TRG 4580063). The authors would like to thank Associate Professor Zhiping Lin and Associate Professor H. A. Park for their valuable comments and discussion.

References

[1] W.W. Adams and P. Loustaunau, "An Introduction to Gröbner Bases,", Graduate Studies in Mathematics, Vol.3, American Mathematical Society, 1994

[2] B. Buchberger, "Gröbner bases: An algorithmic method in polynomial ideal theory," in Multidimensional System Theory (N.K. Bose, ed.), Reidel, Dordrecht, 1985, pp. 184-232

[3] B. Buchberger, "Gröbner Bases and System Theory," Multidimensional Systems and Signal Processing, Vol. 12, 2001, Boston: Kluwer Academic Publishers, pp. 223-251

[4] C. Charoenlarpnopparut and N.K. Bose, "Multidimensional filter bank design using Gröbner bases," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 46, No 12, December 1999, pp. 1475-1486

[5] C. Charoenlarpnopparut, "Gröbner Bases in Multidimensional Systems and Signal Processing," *Ph.D. Dissertation*, Pennsylvania State University, PA, USA, 2000

[6] C. Charoenlarpnopparut and N.K. Bose, "Gröbner Bases For Problem Solving in Multidimensional Systems," Multidimensional Systems and Signal Processing, Vol. 12, No. 3-4, 2001, pp. 365-376

[7] C. Charoenlarpnopparut and S. Tantaratana, "Multidimensional convolutional code: Progresses and bottlenecks," 2003 IEEE International Symposium on Circuits and Systems, Bangkok, Thailand, May 2003 (CD-ROM)

[8] C. Chaoenlarpnopparut, S. Wongsura, and A. Davies, "Direct realization of a 2-D convolutional encoder with lesser number of delay elements," Proc. 3rd International Symposium on Communications and Information Technologies, Thailand, 3-5 September 2003, Vol. 1, pp.258-262

[9] D. Cox, J. Little and D. O'Shea, IDEALS, VARIETIES, AND ALGORITHM: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 2^{nd} edition, Springer-Verlag, New York, 1996

[10] E. Fornasini and M.E. Valcher, "Algebraic aspect of 2D convolutional codes," *IEEE Trans. Inform. Theory,* Vol. IT-40, No 4, 1994, pp.1068-1082

[11] G.D. Forney, Jr, "Convolutional codes I: Algebraic structure," IEEE Trans. Info. Theory, Vol. IT-16, November 1970, pp. 720-738

[12] G.D. Forney, Jr, "The Viterbi algorithm," *Proc. of the IEEE*, Vol. 61, March 1973, pp. 268-278

[13] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 2.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2001). `http://www.singular.uni-kl.de`.

[14] Johannesson, Rolf and Zigangirov, Kamil Sh., Fundamentals of Convolutional Coding, New York: IEEE Press, 1999

[15] R. Johannesson and Z. Wan, "A linear algebra appoach to minimal convolutional encoders," IEEE Trans. Information Theory, Vol. 39, No 4, July 1993, pp. 1219-1233

[16] R. Johannesson and Z. Wan, "Some structural properties of convolutional codes over rings," IEEE Trans. Information Theory, Vol. 44, No 2, March 1998, pp. 839-845

[17] S. Mahapakulchai and R. E. Van Dyck, "Design of ring convolutional trellis codes for MAP decoding of MPEG-4 imagery," Proc. IEEE Inter. Conf. Commun. (ICC 2001), June 2001

[18] S. Mahapakulchai, "MAP source-controlled channel decoding for image transmission using CPFSK and ring convolutional codes," *Ph.D. Dissertation*, Pennsylvania State University, PA, USA, 2002

[19] J.L. Massey and T. Mittelholzer, "Convolutional codes over rings," in Proc. 4th Joint Swedish-Soviet Int. Workshop Information Theory Gotland, Sweden, 27 August - 1 September 1989, pp. 14-18

[20] M.E. Valcher and E. Fornasini,"On 2-D finite support convolutional codes: An algebraic approach," Multidimensional Systems and Signal Processing," Vol. 5, No 3, July 1994, pp. 231-243

[21] P.A. Wiener,"Multidimensional Convolutional Codes," *Ph.D. Dissertation*, University of Notre Dame, IN, USA, 1998