INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 4, PP. 527–533
Manuscript received April 11, 2018; revised October, 2018. DOI: 10.24425/123555

Trellis Coded 4-ary PAM using Distance-Preserving Mapping

Thokozani Shongwe

***Abstract*—A trellis coded 4-ary Pulse Amplitude Modulation (4-PAM) is presented, where the encoding algorithm is derived from Distance Preserving Mapping (DPM) algorithm. In this work, we modify the DPM algorithm for 4-PAM and obtain a new construction for mapping binary sequences to permutation sequences, where the permutation sequences are obtained by permuting symbols of a 4-PAM constellation. The resulting codebook of permutation sequences formed this way are termed *mappings*. We also present several metrics for assessing the performance of the *mappings* from our construction, and we show that a metric called the Sum of Product of Distances (SOPD) is the best metric to use when judging the performance of the mappings. Finally, performance results are presented, where the *mappings* from our construction are compared against each other and also against the conventional *mappings* in the literature.**

***Keywords*—Distance-preserving mappings, Hamming distance, Euclidean distance, Pulse Amplitude Modulation**

I. Introduction

DISTANCE-PRESERVING mapping (DPM) is not a new topic, it was investigated in 1989 by Ferreira *et al.* [1]. Ferreira *et al.* [1] mapped a binary convolutional code to a runlength constrained or balanced trellis code. The mapping was such that the free distance of the convolutional code was preserved or increased in the resulting runlength constrained or balanced trellis code. A similar idea of DPM was also presented by French [2], where he constructed distance-preserving runlength-limited (RLL) codes from binary convolutional codes.

In [3], [4] and [5] the authors considered convolutional codes mapped to permutation codes of length M. In the mapping, n-tuples taken from the output of a binary convolutional encoder were mapped to M-tuples of a permutation code such that the Hamming distance of the n-tuples was preserved in the M-tuple permutation codewords. The permutation codes constructed this way were termed permutation trellis codes. By "preserving the Hamming distance" of the n-tuples, the authors meant that the Hamming distance was either kept the same or increased in the M-tuple permutation codewords. Preserving the Hamming distances in the M-tuple permutation codewords meant that the error correcting capabilities of the permutation code was as good as (or better than) the corresponding binary codes (from which they were mapped).

The previous work on mapping binary sequences to permutation sequences dealt with creating *mappings* (sets of sequences) that preserve the Hamming distances of the corresponding binary sequences (see [1] – [8]). These types of mappings were termed Distance-Preserving Mappings (DPMs). By "a mapping" we mean the set of sequences (codebook) resulting from the DPM procedure, which maps binary n-tuples to non-binary M-tuples. In this article we make use of the DPM procedure in [8], of mapping binary n-tuples to permutation M-tuples, to develop a new way of mapping that can be used in M-ary PAM (Pulse Amplitude Modulation). We shall consider the case of $M = 4$ (4-PAM). Work that studied DPMs and PAM was presented in [9]. However, that work discussed the generation of PAM signals with spectral nulls and only considered the Hamming distance, which makes the mapping methods used in [9] different from the methods used in this article. The encoding process of the mappings generated from a PAM constellation can be the same as that of the permutation trellis codes in [4] shown by Fig. 1.

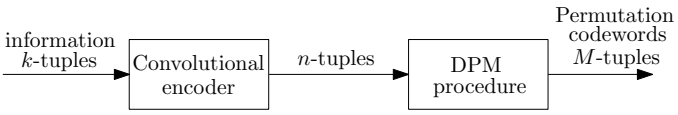

Fig. 1. Encoding process, converting n-tuples from the convolutional encoder into M-tuples, which are permutation sequences.

II. Distance-Preserving Mapping procedure

We first give a brief background discussion about DPMs and the algorithm used to generate DPMs in the next subsection. Then, in Section II-B, we give our new way of generating DPMs which considers the Euclidean distance.

A. Hamming distance to Hamming distance DPM

The existing mapping of binary n-tuples to permutation M-tuples in the literature considers permutations of the set $A = \{1, 2, \ldots, M\}$, where the elements of A are permuted according to unique binary n-tuples ([4], [5] and [8]). We use the following DPM algorithm found in [8], together with Example 1 to illustrate the DPM procedure.

Distance-Preserving Mapping algorithm: A binary sequence $(x_1, x_2, \ldots, x_n)$ is mapped to the permutation sequence $(y_1, y_2, \ldots, y_M)$. Let `swap` (y_i,y_j) denote the swapping of symbols y_i and y_j in a sequence.

Thokozani Shongwe is with Department of Electrical and Electronic Engineering Technology, University of Johannesburg, South Africa (e-mail: tshongwe@uj.ac.za).

Let $n = 3$ and $M = 4$. Then the $(n = 3) \rightarrow (M = 4)$ DPM algorithm is defined as follows:

```
Input: (x1, x2, x3)
Output: (y1, y2, y3, y4)
begin
    (y1, y2, y3, y4) ← (1, 2, 3, 4)
    if x3 = 1 then swap(y2, y4)
    if x2 = 1 then swap(y1, y2)
    if x1 = 1 then swap(y3, y4)
end.
```

Example 1 By applying the $(n = 3) \rightarrow (M = 4)$ DPM algorithm we obtain:

$$\{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow \{1234, 1432, 2134, 4132, 1243, 1342, 2143, 2341\}. \quad (1)$$

As was done in [5], we set up the Hamming distance matrices for the binary sequences and the permutation sequences which are $D = [d_{ij}]$ and $E = [e_{ij}]$, respectively, for the mapping in (1) as follows:

$$\mathbf{D} = \begin{array}{c} \\ 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \begin{array}{c} \begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 \\ 1 & 0 & 2 & 1 & 2 & 1 & 3 & 2 \\ 1 & 2 & 0 & 1 & 2 & 3 & 1 & 2 \\ 2 & 1 & 1 & 0 & 3 & 2 & 2 & 1 \\ 1 & 2 & 2 & 3 & 0 & 1 & 1 & 2 \\ 2 & 1 & 3 & 2 & 1 & 0 & 2 & 1 \\ 2 & 3 & 1 & 2 & 1 & 2 & 0 & 1 \\ 3 & 2 & 2 & 1 & 2 & 1 & 1 & 0 \end{array}\right] \end{array},$$

$$\mathbf{E} = \begin{array}{c} \\ 1234 \\ 1432 \\ 2134 \\ 4132 \\ 1243 \\ 1342 \\ 2143 \\ 2341 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1234 & 1432 & 2134 & 4132 & 1243 & 1342 & 2143 & 2341 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 2 & 2 & 3 & 2 & 3 & 4 & 4 \\ 2 & 0 & 3 & 2 & 3 & 2 & 4 & 4 \\ 2 & 3 & 0 & 2 & 4 & 4 & 2 & 3 \\ 3 & 2 & 2 & 0 & 4 & 3 & 3 & 4 \\ 2 & 3 & 4 & 4 & 0 & 2 & 2 & 3 \\ 3 & 2 & 4 & 3 & 2 & 0 & 3 & 2 \\ 4 & 4 & 2 & 3 & 2 & 3 & 0 & 2 \\ 4 & 4 & 3 & 4 & 3 & 2 & 2 & 0 \end{array}\right] \end{array}.$$

Note that d_{ij} and e_{ij} are the entries of D and E, respectively.□

In general, the matrices D and E can be used to verify the distance-preserving property of DPMs, that is, the corresponding distance entries in E are at least as large as the ones in D ($e_{ij} \geq d_{ij}$) for $i, j = 1, 2, \ldots, 2^n$ as can be seen in Example 1.

In [8], the sum of Hamming distances of E, which we denote by X_{SHD} in this paper for reference, was used to determine the distance optimality of distance-preserving mappings. To calculate X_{SHD}, all the entries in E are summed up, resulting in

$$X_{\text{SHD}} = \sum_{j=1}^{2^n} \sum_{i=1}^{2^n} e_{ij}. \quad (2)$$

The higher the value of the sum of Hamming distances of E (X_{SHD}), the better the mapping in terms of error correcting capabilities [8].

B. Hamming distance to Euclidean distance DPM

Our new mapping procedure maps the 2^n binary n-tuples to permutation M-tuples taken from a set, $A = \{-M + 1, -M + 3, \ldots, -1, +1, \ldots, M - 3, M - 1\}$, for M even; $A = \{(-M + 1)/2, (-M + 3)/2, \ldots, -1, 0, +1, \ldots, (M - 3)/2, (M - 1)/2\}$, for M odd. In this article, we will focus the application of our new mapping procedure to the case of $M = 4$, that is $A = \{-3, -1, +1, +3\}$. Now, instead of the Hamming distance metric we will use the Euclidean distance metric for our E matrix. However, we still consider the Hamming distance for our D matrix.

Our goals are to: (a) "preserve" the D distances in E, that is $e_{ij} \geq d_{ij}$, and (b) make sure that the maximum Euclidean distances are on the minor diagonal of E. Goals (a) and (b) are a systematic way to guarantee that the mappings are distance-preserving, and optimal (or sub-optimal, if not optimal).

We use the following observation and knowledge to achieve goals (a) and (b): to achieve goal (a), note that the existing mapping mentioned in (1) builds/increases the Hamming distance between the permutation sequences by swapping the positions of the permutation sequence according to the corresponding binary sequence. In our new set A, *building/increasing the Hamming distance implies building/increasing the Euclidean distance*. To achieve goal (b), we observe that for a given permutation sequence, P_k ($k = 1, 2, \ldots, M!$) from the set of $M!$ permutation sequences, there can only be one permutation sequence with which it can give the maximum Euclidean distance. That permutation sequence is the one that whose symbols are the complements of the symbols of P_k, which we denote by $\bar{P}_k$. For example, for $M = 4$, $A = \{-3, -1, +1, +3\}$, if $P_k = +1\ -1\ +3\ -3$, then $\bar{P}_k = -1\ +1\ -3\ +3$.

To achieve goal (a), we apply the DPM algorithm presented earlier to obtain the first half of the permutation sequences with symbols taken from $A = \{-3, -1, +1, +3\}$. To illustrate this we use the mapping in Example 1 and obtain:

$$\{000, 001, 010, 011\} \rightarrow \{-3\ -1\ +1\ +3, -3\ +3\ +1\ -1, -1\ -3\ +1\ +3, +3\ -3\ +1\ -1\},$$

or

$$\{000, 001, 010, 011\} \rightarrow \{P_1, P_2, P_3, P_4\}$$

in short notation. Then the other half of the permutation sequences will be the mirrored opposites of the first half of the permutation sequences, hence achieving goal (b). This results in the complete DPM, which we call DPM_1 as:

$$\text{DPM}_1 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow \{-3\ -1\ +1\ +3, -3\ +3\ +1\ -1, -1\ -3\ +1\ +3, +3\ -3\ +1\ -1, -3\ +3\ -1\ +1, +1\ +3\ -1\ -3, +3\ -3\ -1\ +1, +3\ +1\ -1\ -3\},$$

or

$$\{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow \{P_1, P_2, P_3, P_4, \bar{P}_4, \bar{P}_3, \bar{P}_2, \bar{P}_1\}.$$

Note that the third symbol in the first half of the permutation sequences is the same, due to the DPM algorithm. This is used here to guarantee that P_k and $\bar{P}_k$ do not appear together in the first half of the permutation sequences. The E matrix of Euclidean distances for DPM_1 is

$$\mathbf{E}_1 = \begin{bmatrix} 0 & 5.7 & 2.8 & 7.5 & 4.9 & 8.5 & 6.9 & 8.9 \\ 5.7 & 0 & 7.5 & 8.5 & 2.8 & 4.9 & 8.9 & 6.9 \\ 2.8 & 7.5 & 0 & 5.7 & 6.9 & 8.9 & 4.9 & 8.5 \\ 7.5 & 8.5 & 5.7 & 0 & 8.9 & 6.9 & 2.8 & 4.9 \\ 4.9 & 2.8 & 6.9 & 8.9 & 0 & 5.7 & 8.5 & 7.5 \\ 8.5 & 4.9 & 8.9 & 6.9 & 5.7 & 0 & 7.5 & 2.8 \\ 6.9 & 8.9 & 4.9 & 2.8 & 8.5 & 7.5 & 0 & 5.7 \\ 8.9 & 6.9 & 8.5 & 4.9 & 7.5 & 2.8 & 5.7 & 0 \end{bmatrix}. \tag{3}$$

In general, the mappings we are creating are of the form $\{P_1, P_2, \ldots, P_{2^n/2}, \bar{P}_{2^n/2}, \ldots, \bar{P}_2, \bar{P}_1\}$, where the first half of this set, $\{P_1, P_2, \ldots, P_{2^n/2}\}$ is obtained using a DPM algorithm that guarantees that P_k and $\bar{P}_k$ do not appear in that set. Throught the article we shall be denoting the E matrix of a corresponding mapping, DPM_x by $\mathbf{E}_x$. Next, we give performance measure metrics which can be used to compare the performance of the different mappings we create with our algorithm.

III. Performance Measure Metrics

Since we are now dealing with different distance metrics in D and E to assess the mappings, we look at different ways of assessing the performance of mappings, which are an adaptation of the one in (2) which was proposed [8]. In the metrics introduced in this article, we seek to capture the distance improvement (increase) from D to E matrices. We begin by explaining two closely related metrics as follows. Given a fixed, positive number, say N, the conventional way to measure the improvement from N to another number, say N_i (where $N_i \geq N$ and i is a positive integer for all possible numbers) is to find the difference between the two numbers, $N_i - N$. The larger the difference, the larger the improvement. Another way of finding the improvement from N to N_i would be to find the product of the two numbers, $N \times N_i$. The larger the product, the larger the improvement.

The measure of the performance of the mappings is captured in the relationship between the corresponding elements of the D and E matrices. As mentioned in the definition of a DPM, the distances (or elements) in the E matrix should be at least as large as the corresponding distances in the D matrix. To compare the improvement from one distance (D distances) to the other distance (E distances), we can either subtract the two corresponding distances and pay attention to the value of their difference or multiply the two corresponding distances and pay attention to the value of their product. We need to remember that the D matrix will be the same for different E matrices. For example, $D \times E_1$ and $D \times E_2$ which are element-wise multiplication, can be compared because in each case we are multiplying the elements of an E matrix with fixed elements of a D matrix. After performing either a subtraction or multiplication of the D and E matrices, a new matrix is obtained which is the difference of distances (DD) or product of distances (PD), respectively. Summing up the entries of the DD or PD matrix we obtain a value which we call the sum of difference of distance (SODD) or sum of product of distances (SOPD), respectively. The SOPD is maximised when a large value is multiplied by another large value in the D and E matrices. Whereas the SODD is maximised when the two entries being subtracted are very different in value, that is a large value minus a small value.

Recall that d_{ij} and e_{ij} denote the entries of D and E, respectively, for $i, j = 1, 2, \ldots, 2^n$. Then the SODD and SOPD can be mathematically described as follows:

(a) the *sum of the difference of distances* (SODD) of D and E. Let the difference of two corresponding entries of D and E, be denoted by $\mathcal{D}_{ij}$ as $\mathcal{D}_{ij} = e_{ij} - d_{ij}$, resulting in a new matrix $\mathcal{D} = E - D$ (element-wise subtraction). The value of the SODD is then given as

$$\begin{aligned} X_{\text{SODD}} &= \sum_{j=1}^{2^n} \sum_{i=1}^{2^n} \mathcal{D}_{ij} \\ &= \sum_{j=1}^{2^n} \sum_{i=1}^{2^n} e_{ij} - d_{ij}. \end{aligned} \tag{4}$$

(b) the *sum of the product of distances* (SOPD) of D and E. Let the product of two corresponding entries of D and E, be denoted by P_{ij} as $P_{ij} = d_{ij} \times e_{ij}$, resulting in a new matrix $P = D \times E$ (element-wise multiplication). The value of the SOPD is then given as

$$\begin{aligned} X_{\text{SOPD}} &= \sum_{j=1}^{2^n} \sum_{i=1}^{2^n} P_{ij} \\ &= \sum_{j=1}^{2^n} \sum_{i=1}^{2^n} d_{ij} \times e_{ij}. \end{aligned} \tag{5}$$

A third method of assessing the mappings would be to treat the entries of D and E matrices as vectors in the dimension (or unit) D and dimension E, respectively. This results a metric we call,

(c) the *sum of the resultant magnitude* (SRM) of the corresponding entries of D and E, d_{ij} and e_{ij}, respectively. This criteria of assessing the mappings treats the entries d_{ij} and e_{ij} as vectors, where d_{ij} is the magnitude of a vector in the Hamming distance (x-axis) direction and e_{ij} is the magnitude of a vector in the Euclidean distance (y-axis) direction. These two entries result in a resultant vector $\mathcal{R}_{ij} = d_{ij} + \mathcal{J}e_{ij}$, where $\mathcal{J}$ denotes the unit vector on the y-axis, and $\mathcal{R}_{ij}$ are entries of matrix $\mathcal{R} = D + \mathcal{J}E$ (element-wise addition). The magnitude of the resultant vector $|\mathcal{R}_{ij}| = \sqrt{(d_{ij})^2 + (e_{ij})^2}$ captures the improvement from d_{ij} to e_{ij}. Therefore, to assess the mappings using SRM, we obtain the sum of the resultant vectors, which is

$$X_{\text{SRM}} = \sum_{j=1}^{2^n}\sum_{i=1}^{2^n}|\mathcal{R}_{ij}| = \sum_{j=1}^{2^n}\sum_{i=1}^{2^n}\sqrt{(d_{ij})^2+(e_{ij})^2}. \quad (6)$$

Unlike the sum of Hamming distances of E used in [8], the SODD, SOPD and SRM involve both the D and E matrix entries in the calculation. Including both D and E matrices gives the improvement (distance increase) in E relative to the distances in D. It can be seen that setting $D = 1$ in (5) reduces it to (2); setting $D = 0$ in (4) or (6) reduces it to (2). This eliminates the reference to the D matrix as was done in (2). The conclusion drawn from this observation is that the sum of Hamming distances of E can be a specific case of either the SODD or the SOPD or the SRM. We will present results later to show that the sum of the distances of E as well as the SODD cannot be used as performance measure metrics in the mappings in this article because the D and E distances are of different units/dimensions. However, the SOPD and SRM can be used for both cases, when the units of the D and E distances are the same and when they are different. This means that the SOPD and the SRM proposed in this article can be used in place of the sum of Hamming distances of E.

In the next subsections we illustrate the use of matrices D and E in calculating the performance measure metrics discussed above, but before we do that we need to have two mappings to compare in terms of the metrics. We have already created DPM_1, then we need to create a second mapping for comparison purposes. To create the second mapping, we use the DPM algorithm applied in Example 1 without modification, to obtain a mapping using the set $A = \{-3, -1, +1, +3\}$. This results in a mapping, which we call DPM_2 as:

$$\text{DPM}_2 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow \{-3\ -1\ +1\ +3, -3\ +3\ +1\ -1, -1\ -3\ +1\ +3, +3\ -3\ +1\ -1, -3\ -1\ +3\ +1, -3\ +1\ +3\ -1, -1\ -3\ +3\ +1, -1\ +1\ +3\ -3\},$$

which has the E matrix of Euclidean distances

$$\mathbf{E}_2 = \begin{bmatrix} 0 & 5.7 & 2.8 & 7.5 & 2.8 & 4.9 & 4.0 & 6.9 \\ 5.7 & 0 & 7.5 & 8.5 & 4.9 & 2.8 & 6.9 & 4.0 \\ 2.8 & 7.5 & 0 & 5.7 & 4.0 & 6.3 & 2.8 & 7.5 \\ 7.5 & 8.5 & 5.7 & 0 & 6.9 & 7.5 & 4.9 & 6.3 \\ 2.8 & 4.9 & 4.0 & 6.9 & 0 & 2.8 & 2.8 & 4.9 \\ 4.9 & 2.8 & 6.3 & 7.5 & 2.8 & 0 & 4.9 & 2.8 \\ 4.0 & 6.9 & 2.8 & 4.9 & 2.8 & 4.9 & 0 & 5.7 \\ 6.9 & 4.0 & 7.5 & 6.3 & 4.9 & 2.8 & 5.7 & 0 \end{bmatrix}. \quad (7)$$

DPM_1 metrics calculation

- SODD: $\mathcal{D} = E_1 - D$, then

$$\mathcal{D} = \begin{bmatrix} 0 & 4.7 & 1.8 & 5.5 & 3.9 & 6.5 & 4.9 & 5.9 \\ 4.7 & 0 & 5.5 & 7.5 & 0.8 & 3.9 & 5.9 & 4.9 \\ 1.8 & 5.5 & 0 & 4.7 & 4.9 & 5.9 & 3.9 & 6.5 \\ 5.5 & 7.5 & 4.7 & 0 & 5.9 & 4.9 & 0.8 & 3.9 \\ 3.9 & 0.8 & 4.9 & 5.9 & 0 & 4.7 & 7.5 & 5.5 \\ 6.5 & 3.9 & 5.9 & 4.9 & 4.7 & 0 & 5.5 & 1.8 \\ 4.9 & 5.9 & 3.9 & 0.8 & 7.5 & 5.5 & 0 & 4.7 \\ 5.9 & 4.9 & 6.5 & 3.9 & 5.5 & 1.8 & 4.7 & 0 \end{bmatrix}.$$

Therefore,

$$X_{\text{SODD}} = \sum_{j=1}^{8}\sum_{i=1}^{8}\mathcal{D}_{ij} = 265.80.$$

- SOPD: $P = E_1 \times D$, then

$$P = \begin{bmatrix} 0 & 5.7 & 2.8 & 15.0 & 4.9 & 17.0 & 13.9 & 26.8 \\ 5.7 & 0 & 15.0 & 8.5 & 5.7 & 4.9 & 26.8 & 13.9 \\ 2.8 & 15.0 & 0 & 5.7 & 13.9 & 26.8 & 4.9 & 17.0 \\ 15.0 & 8.5 & 5.7 & 0 & 26.8 & 13.9 & 5.7 & 4.9 \\ 4.9 & 5.7 & 13.9 & 26.8 & 0 & 5.7 & 8.5 & 15.0 \\ 17.0 & 4.9 & 26.8 & 13.9 & 5.7 & 0 & 15.0 & 2.8 \\ 13.9 & 26.8 & 4.9 & 5.7 & 8.5 & 15.0 & 0 & 5.7 \\ 26.8 & 13.9 & 17.0 & 4.9 & 15.0 & 2.8 & 5.7 & 0 \end{bmatrix}.$$

Therefore,

$$X_{\text{SOPD}} = \sum_{j=1}^{8}\sum_{i=1}^{8}P_{ij} = 665.46.$$

- SRM: $\mathcal{R} = D + \mathcal{J}E_1$, then

$$\mathcal{R} = \begin{bmatrix} 0 & 1+\mathcal{J}5.7 & 1+\mathcal{J}2.8 & 2+\mathcal{J}7.5 & 1+\mathcal{J}4.9 & 2+\mathcal{J}8.5 & 2+\mathcal{J}6.9 & 3+\mathcal{J}8.9 \\ 1+\mathcal{J}5.7 & 0 & 2+\mathcal{J}7.5 & 1+\mathcal{J}8.5 & 2+\mathcal{J}2.8 & 1+\mathcal{J}4.9 & 3+\mathcal{J}8.9 & 2+\mathcal{J}6.9 \\ 1+\mathcal{J}2.8 & 2+\mathcal{J}7.5 & 0 & 1+\mathcal{J}5.7 & 2+\mathcal{J}6.9 & 3+\mathcal{J}8.9 & 1+\mathcal{J}4.9 & 2+\mathcal{J}8.5 \\ 2+\mathcal{J}7.5 & 1+\mathcal{J}8.5 & 1+\mathcal{J}5.7 & 0 & 3+\mathcal{J}8.9 & 2+\mathcal{J}6.9 & 2+\mathcal{J}2.8 & 1+\mathcal{J}4.9 \\ 1+\mathcal{J}4.9 & 2+\mathcal{J}2.8 & 2+\mathcal{J}6.9 & 3+\mathcal{J}8.9 & 0 & 1+\mathcal{J}5.7 & 1+\mathcal{J}8.5 & 2+\mathcal{J}7.5 \\ 2+\mathcal{J}8.5 & 1+\mathcal{J}4.9 & 3+\mathcal{J}8.9 & 2+\mathcal{J}6.9 & 1+\mathcal{J}5.7 & 0 & 2+\mathcal{J}7.5 & 1+\mathcal{J}2.8 \\ 2+\mathcal{J}6.9 & 3+\mathcal{J}8.9 & 1+\mathcal{J}4.9 & 2+\mathcal{J}2.8 & 1+\mathcal{J}8.5 & 2+\mathcal{J}7.5 & 0 & 1+\mathcal{J}5.7 \\ 3+\mathcal{J}8.9 & 2+\mathcal{J}6.9 & 2+\mathcal{J}8.5 & 1+\mathcal{J}4.9 & 2+\mathcal{J}7.5 & 1+\mathcal{J}2.8 & 1+\mathcal{J}5.7 & 0 \end{bmatrix}.$$

Therefore,

$$X_{\text{SRM}} = \sum_{j=1}^{8}\sum_{i=1}^{8}|\mathcal{R}_{ij}| = 375.99.$$

DPM_2 metrics calculation

In this subsection we avoid showing the entire procedure of the calculation of the metrics for DPM_2 because it is similar to that of DPM_1. We show only the results as follows.

- SODD: $\mathcal{D} = E_2 - D$, then

$$X_{\text{SODD}} = \sum_{j=1}^{8}\sum_{i=1}^{8}\mathcal{D}_{ij} = 194.23.$$

- SOPD: $P = E_2 \times D$, then

$$X_{\text{SOPD}} = \sum_{j=1}^{8}\sum_{i=1}^{8}P_{ij} = 531.53.$$

- SRM: $\mathcal{R} = D + \mathcal{J}E_2$, then

$$X_{\text{SRM}} = \sum_{j=1}^{8}\sum_{i=1}^{8}|\mathcal{R}_{ij}| = 307.38.$$

Using the D matrix in Example 1 we obtain the SOPD for DPM_2, which is 531.5264. It can be seen that the SOPD for DPM_2 is lower than that for DPM_1 which is 665.4580. We will show the significance of this difference in SOPD in Section V.

IV. System Model

The system model to be used for the simulation of mappings is described in Fig 3. The system employs a convolutional code of rate $R = k/n$, constraint length K and free-distance d_{free}, and an M-PAM modulator as follows: at point $\mathbf{a}$ the system takes in information bits in k-tuples, and produces n bits for every k information bit, at point $\mathbf{b}$. Every n-tuple of bits is mapped to an M-tuple permutation codeword by the DPM procedure. The permutation codewords at point $\mathbf{c}$ are of length M taken from an M-PAM constellation. The receiver accepts symbols corrupted by AWGN (additive white Gaussian noise) at point $\mathbf{d}$. Then the symbols are demodulated, resulting in a stream of noise-corrupted permutation codewords (taken from an M-PAM constellation) at point $\mathbf{e}$. The decoder takes in the received symbols at point $\mathbf{e}$ in M-tuples, and performs soft-decision decoding using the Viterbi algorithm to produce an estimate of the information bits at point $\mathbf{f}$. For the soft-decision decoding, every received M-tuple is compared with M-tuples on the branches of the trellis using the squared Euclidean distance. The M-tuples on the branches of the trellis are the codewords of the mapping. In Figure 2 we used the codewords (4-tuples) of DPM_1 and a convolutional code of $R = 1/3$, $K = 4$, $d_{\text{free}} = 10$ to illustrate how the 4-tuples appear on the branches of the trellis. The circled numbers (0 to 7) are the states of the decoder.

The 4-tuples on the branches of the trellis are the codewords of the mapping, hence this step of the decoding process is equivalent to comparing the received 4-tuple with each of the codewords of the mapping to find the codeword that was transmitted.

We observed that for the viterbi decoder to give good performance, the larger distance in the D matrix must correspond to a larger distance value in the E matrix; the smaller distance in the D matrix must correspond to a smaller distance value in the

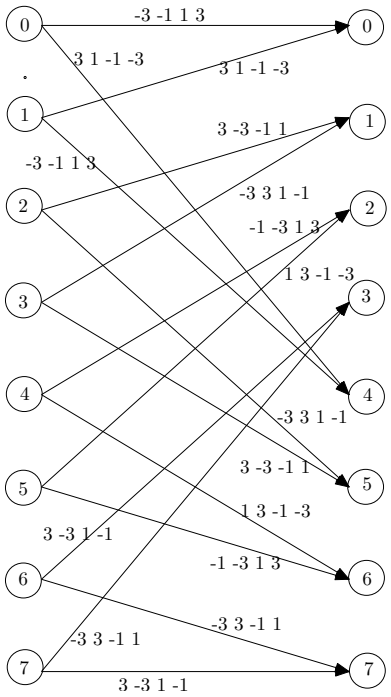

Fig. 2. Trellis diagram for the $R = 1/3$, $K = 4$, $d_{\text{free}} = 10$ convolutional code with codewords from DPM_1 on the branches.

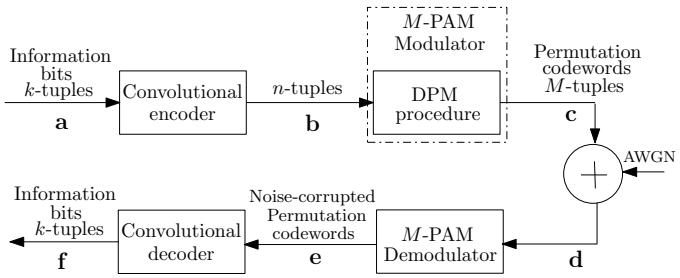

Fig. 3. System model used for the simulations.

E matrix. This is equivalent to keeping the original distances arrangement of the trellis unchanged and only scaling the values of the distances for better performance. This is best captured by a SOPD metric instead of a SODD or SRM metric.

In this article we use the SOPD metric, described above, as a performance measure because it is easier to compute compared to the conventional free distance. We therefore use the free distance calculation of convolutional codes to validate the SOPD used in this article. The following example shows that both the free distance and SOPD give the same results for our mappings.

It needs to be stressed that distance preserving mappings, by definition are concerned with the improvement in distance from the D matrix to the E matrix ($e_{ij} \geq d_{ij}$ for i, j). The larger the corresponding distance entries in E matrix from the D matrix, the better the mapping. Simply summing the E matrix entries will not accurately capture the performance of the mappings when the D and E matrices have different distance types, like in the case of this article.

V. Performance Results

In addition to DPM_1 and DPM_2, we introduce DPM_3, DPM_4, DPM_5 and DPM_6. DPM_3, DPM_4, DPM_5 and DPM_6 are as follows:

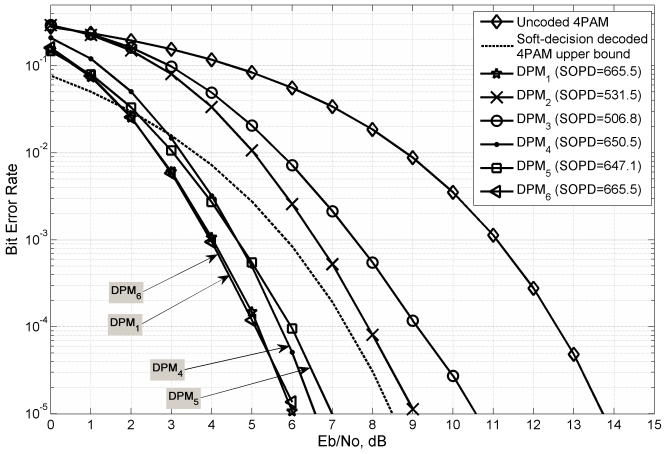

Fig. 4. Performance comparison results of six different mappings: DPM_1, DPM_2, DPM_3, DPM_4, DPM_5 and DPM_6. Uncoded 4-PAM and convolutional encoded soft-decision decoded 4-PAM results are also displayed. An ($R = 1/3$, $K = 4$, $d_{\text{free}} = 10$) convolutional code was used.

$$\text{DPM}_3 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow$$
$$\{-3\ -1\ +1\ +3, -3\ -1\ +3\ +1, -3\ +1\ -1\ +3, -3\ +1\ +3\ -1,$$
$$-1\ -3\ +3\ +1, -1\ -3\ +1\ +3, -1\ +1\ -3\ +3, -1\ +1\ +3\ -3\},$$

$$\text{DPM}_4 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow$$
$$\{-3\ -1\ +1\ +3, +1\ -1\ -3\ +3, -3\ +3\ +1\ -1, +1\ +3\ -3\ -1,$$
$$-1\ -3\ +1\ +3, +1\ -3\ -1\ +3, -1\ +3\ +1\ -3, +1\ +3\ -1\ -3\},$$

$$\text{DPM}_5 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow$$
$$\{-3\ -1\ +1\ +3, -3\ +3\ +1\ -1, -1\ -3\ +1\ +3, +3\ -3\ +1\ -1,$$
$$-3\ +3\ -1\ +1, +1\ +3\ -1\ -3, +3\ +1\ -1\ -3, +3\ -3\ -1\ +1\},$$

$$\text{DPM}_6 : \{000, 001, 010, 011, 100, 101, 110, 111\} \rightarrow$$
$$\{-3\ +3\ +1\ -1, -3\ -1\ +1\ +3, +3\ -3\ +1\ -1, -1\ -3\ +1\ +3,$$
$$+1\ +3\ -1\ -3, -3\ +3\ -1\ +1, +3\ +1\ -1\ -3, +3\ -3\ -1\ +1\}.$$

These new mappings, DPM_3, DPM_4, DPM_5 and DPM_6 were constructed as follows: DPM_3 and DPM_4 were constructed using the type of DPM algorithm used for DPM_2, but these algorithms are different from the one specifically used to generate DPM_2. DPM_5 was constructed by simply interchanging the positions of the last two permutation sequences in DPM_1. DPM_6 was constructed using our new construction of DPMs, similary to DPM_1. To verify that the mappings are distance-preserving, the reader can create the D and E matrices for each mapping as was done in Example 1.

Fig. 4 shows the simulation results comparing the following: the six different mappings (DPM_1–DPM_6); uncoded 4-PAM; the theoretical bound on conventional soft-decision decoding of 4-PAM where the ($R = 1/3$, $K = 4$, $d_{\text{free}} = 10$) convolutional code is employed. The mappings (DPM_1 and DPM_6) obtained from our new construction have equal performance and their performance is the best against all other mappings. Our mappings, DPM_1 and DPM_6 also show a 5.5 dB coding gain when compared with the conventional soft-decision decoding of the ($R = 1/3$, $K = 4$, $d_{\text{free}} = 10$) convolutional coded data.

From Fig. 4 it can be seen that the higher the SOPD, the better the mapping's performance. This shows that the SOPD is a good metric for judging the performance of the mappings. However, when considering the SODD and SRM in Table I and the corresponding performance results in Fig. 4 of the mappings, it is evident that the SODD and SRM cannot be used to judge the performance of the mappings, especially for high signal-to-noise ratio (SNR). However, a close observation of SODD in Table I and the corresponding performance results in Fig. 4 of the mappings reveal that actually the SODD is good metric for low SNR: for example, DPM_4 and DPM_5 in Table I have SOPDs of 650.5 and 647.1, respectively. While DPM_4 has a higher SOPD than that of DPM_5, it has a lower SODD of 251.9 (versus an SODD of DPM_5 of 265.8). Fig. 4 shows that DPM_5 performs better than DPM_4 at SNR below 4.5dB; It can also be observed in Table I that DPM_5 and DPM_6 which have the same SODD of 265.8, have the same performance at SNR below 1dB in Fig. 4. Having said this about the SODD metric, the SOPD is a metric of choice because the bit-error rate (BER) vs SNR performance curves in communications become asymptotically accurate as the BER approaches zero (which corresponds to increasing SNR). Therefore in terms of asymptotic BER, the SOPD is a metric of choice over all the other metrics.

Looking at the E matrices for DPM_1 and DPM_2 in (3) and (7), respectively we can see that the minimum Euclidean distance is the same, 2.8. It is interesting that even though DPM_1 and DPM_2 have the same minimum Euclidean distance, their performance in Fig. 4 is different. This difference in performance is due to their different SOPD.

TABLE I
Mappings and their sum of distances: sum of the Hamming distances of E, sum of the Euclidean distances of E and sum of the product of distances from D and E (SOPD).

Mapping	Sum of the Resultant Magnitudes (SRM)	Sum of the Euclidean Distances	Sum of the difference of distances (SODD)	Sum of the product of distances (SOPD)
DPM_1	376.0	361.8	265.8	665.5
DPM_2	194.2	290.2	194.2	531.5
DPM_3	302.2	283.8	187.3	506.8
DPM_4	362.1	347.9	251.9	650.5
DPM_5	376.7	361.8	265.8	647.1
DPM_6	376.0	361.8	265.8	665.5

The sum of the Hamming distances of the E matrices was considered as a metric for assessing the performance of the mappings in [8], and was found to be a good metric. The same idea can be applied to the mappings in this article to obtain the sum of the Euclidean distances of the E matrices.

However, the sum of the Euclidean distances of E fails to capture the accurate performance of the mappings (that are using the Euclidean distances in their E matrices). This failure of the sum of the Euclidean distances of E as an accurate metric for judging the performance of the mappings is evident when looking at the sum of the Euclidean distances of the E matrices of DPM_1 and DPM_5 in Table I, together with the performances of DPM_1 and DPM_5 in Fig. 4. DPM_1 and DPM_5 have the same sum of Euclidean distances, 361.8, but Fig. 4 shows that DPM_1 performs better than DPM_5. Having said that, however, the sum of the Euclidean distances of the E matrix still closely tracks the performance of the mappings, for mappings that are too different in terms of bit error rate performance.

To further show the consistency of the SOPD as a metric for assessing the performance of DPMs, we present other performance results in Fig. 5, where we use a different convolutional code ($R = 1/3$, $K = 3$, $d_{\text{free}} = 6$) than the one used for the results in Fig. 4. The system set-up in Fig. 5 is the same as the one used to get the results in Fig. 4; the only thing that was changed was the convolutional code.

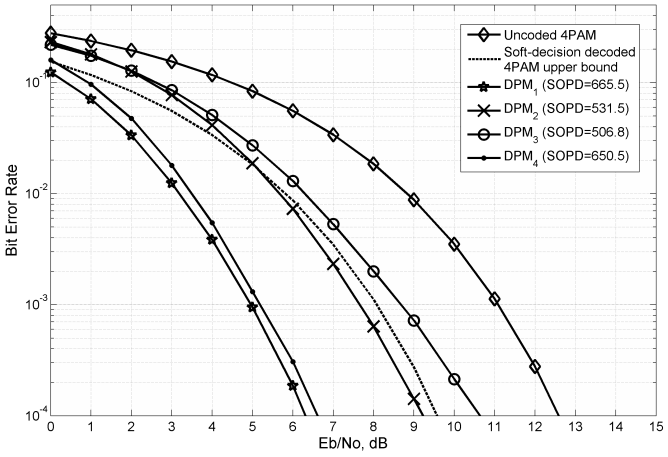

Fig. 5. Performance comparison results of four different mappings: DPM_1, DPM_2, DPM_3 and DPM_4. Uncoded 4-PAM and convolutional encoded soft-decision decoded 4-PAM results are also displayed. An ($R = 1/3$, $K = 3$, $d_{\text{free}} = 6$) convolutional code was used.

In Fig. 5 we have used a subset of the mappings used for Fig. 4, and the same trend of performance as in Fig. 4 is observed. We show results for only DPM_1, DPM_2, DPM_3 and DPM_4 because they suffice to show the relationship between the performance of the mappings and SOPD.

VI. Conclusion

A distance-preserving mapping construction for mapping binary sequences to permutation sequences for 4-PAM constellation was presented. A mapping from our construction was presented and resulted in the best performance compared to mappings from the conventional DPM procedure. We have shown that the *sum of the product of distances* (SOPD) is a good metric for assessing the performance of the mappings. We have also shown that both the sum of the Euclidean distances and the minimum Euclidean distance are not appropriate metrics for assessing the performance of mappings. Having found mappings that outperform conventional mappings, we still need to find a general construction for M-ary PAM and proof that mappings from our construction are optimal.

References

[1] H. C. Ferreira and D. A. Wright and A. L. Nel, "Hamming distance preserving mappings and trellis codes with constrained binary symbols", *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1098–1103, sept. 1989.

[2] C. A. French, "Distance preserving run-length limited codes", *IEEE Transactions on Magnetics*, vol. 25, no. 5, pp. 4093–4095, Sept. 1989.

[3] H. C. Ferreira and A. J. H. Vinck, "Interference cancellation with permutation trellis codes", in *Proceedings of the 2000 IEEE Vehicular Technology Conference, Boston*, MA, USA, Sept. 2428, 2000, pp. 2401–2407.

[4] A. J. H. Vinck and H. C. Ferreira, "Permutation trellis codes", in *Proceedings of the 2001 IEEE International Symposium on Information Theory*, Washington, DC, USA, June 24–29, 2001, p. 279.

[5] H. C. Ferreira, A. J. H. Vinck, T. G. Swart and I. de Beer, "Permutation trellis codes", *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1782–1789, Nov. 2005.

[6] J.-C. Chang, R.-J. Chen, T. Klve, and S.-C. Tsai, "Distancepreserving mappings from binary vectors to permutations", *IEEE Transactions on Information Theory*, vol. 49, no. 4, pp. 1054–1059, Apr. 2003.

[7] K. Lee, "New distance-preserving mappings of odd length", *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2539–2543, Oct. 2004.

[8] T. G. Swart and H. C. Ferreira, "A generalized upper bound and a multi-level construction for distance-preserving mappings", *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3685–3695, Aug. 2006.

[9] K. Ouahada, T. G. Swart and H. C. Ferreira, "Permutation sequences and coded PAM signals with spectral nulls at rational submultiples of the symbol frequency", *Cryptography and Communications*, vol. 3, no.1, pp. 87–108, 2011.