Distance-Preserving Mapping with Euclidean Distance for 4-ary PAM

Thokozani Shongwe
Department of Electrical and Electronic Engineering Technology, University of Johannesburg, P.O. Box 17011, Doornfontein, 2028, Johannesburg, South Africa
Email: tshongwe@uj.ac.za

Theo G. Swart
Department of Electrical and Electronic Engineering Science, University of Johannesburg, P.O. Box 524, Auckland Park, 2006, Johannesburg, South Africa
Email: tgswart@uj.ac.za

Hendrik C. Ferreira
Department of Electrical and Electronic Engineering Science, University of Johannesburg, P.O. Box 524, Auckland Park, 2006, Johannesburg, South Africa
Email: hcferreira@uj.ac.za

***Abstract*—In this article we give: (a) a new construction for mapping binary sequences to permutation sequences formed from a 4-PAM constellation, and call the resulting codebooks *mappings*; (b) a metric for assessing the performance of *mappings* from our construction; (c) performance results comparing *mappings* from our construction against the conventional *mappings* in the literature. The results show that our mappings outperform the conventional mappings. Against conventional soft-decision decoded 4-PAM, our mapping showed 2.4 dB and 3.2 dB improvement over ($R = 1/3$, $K = 4$, $d_{\text{free}} = 10$) and ($R = 1/3$, $K = 3$, $d_{\text{free}} = 6$) convolutional codes, respectively.**

***Index Terms*—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.

In [2], [3] and [4] 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 [3] shown by Fig. 1.

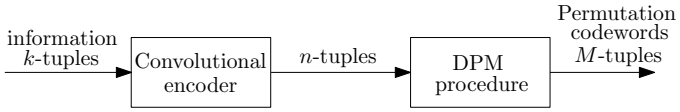

Fig. 1. Encoding process, converting n-tuples from the convolutional encoder into M-tuples, which are permutation sequences.

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 ([3], [4] 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.

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 [4], 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, where all the entries in E are summed up ($\sum_{j=1}^{2^n} \sum_{i=1}^{2^n} e_{ij}$) was used to determine the distance optimality of distance-preserving mappings. The higher the sum of Hamming distances of E, the better the mapping in terms of error correcting capabilities.

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). Since we are now dealing with different distance metrics in D and E to assess the mappings, we propose the use of a *sum of the product of distances* (SOPD) of D and E, which is $\sum_{j=1}^{2^n} \sum_{i=1}^{2^n} d_{ij} \times e_{ij}$, where d_{ij} and e_{ij} denote the entries of D and E, respectively. The higher the SOPD, the better the mapping. We will later give justification for the use of SOPD as measure of the performance for our mappings, in Section IV.

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

$$\mathbf{E} = \begin{bmatrix} 0 & 5.7 & 2.8 & 7.5 & 4.9 & 8.5 & 6.9 & 8.9 \\ 5.7 & 0 & 7.5 & 8.5 & 2.8 & 4.9 & 8.9 & 6.9 \\ 2.8 & 7.5 & 0 & 5.7 & 6.9 & 8.9 & 4.9 & 8.5 \\ 7.5 & 8.5 & 5.7 & 0 & 8.9 & 6.9 & 2.8 & 4.9 \\ 4.9 & 2.8 & 6.9 & 8.9 & 0 & 5.7 & 8.5 & 7.5 \\ 8.5 & 4.9 & 8.9 & 6.9 & 5.7 & 0 & 7.5 & 2.8 \\ 6.9 & 8.9 & 4.9 & 2.8 & 8.5 & 7.5 & 0 & 5.7 \\ 8.9 & 6.9 & 8.5 & 4.9 & 7.5 & 2.8 & 5.7 & 0 \end{bmatrix}. \quad (2)$$

Using the D matrix in Example 1 we obtain the SOPD for DPM_1, which is 665.4580. By computer search, we found the highest SOPD to be 665.4580 and DPM_1 attains this SOPD.

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.

For comparison purposes, 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

$$\mathbf{E} = \begin{bmatrix} 0 & 5.7 & 2.8 & 7.5 & 2.8 & 4.9 & 4.0 & 6.9 \\ 5.7 & 0 & 7.5 & 8.5 & 4.9 & 2.8 & 6.9 & 4.0 \\ 2.8 & 7.5 & 0 & 5.7 & 4.0 & 6.3 & 2.8 & 7.5 \\ 7.5 & 8.5 & 5.7 & 0 & 6.9 & 7.5 & 4.9 & 6.3 \\ 2.8 & 4.9 & 4.0 & 6.9 & 0 & 2.8 & 2.8 & 4.9 \\ 4.9 & 2.8 & 6.3 & 7.5 & 2.8 & 0 & 4.9 & 2.8 \\ 4.0 & 6.9 & 2.8 & 4.9 & 2.8 & 4.9 & 0 & 5.7 \\ 6.9 & 4.0 & 7.5 & 6.3 & 4.9 & 2.8 & 5.7 & 0 \end{bmatrix}. \quad (3)$$

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 IV.

III. 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.

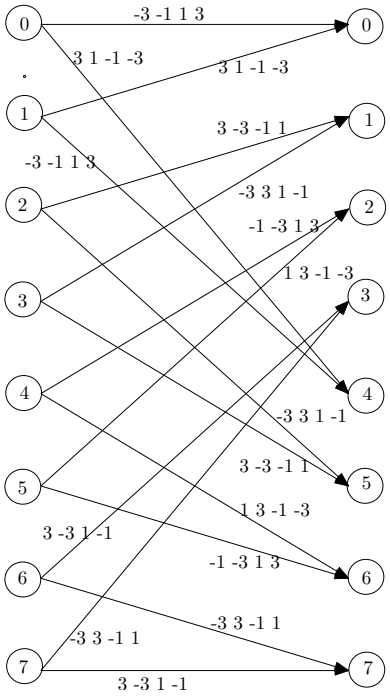

Fig. 2. Trellis diagram for the $R = 1/3$, $K = 4$, $d_{\text{free}} = 10$ convolutional code with codewords from DPM_1 on the branches.

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.

IV. 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:

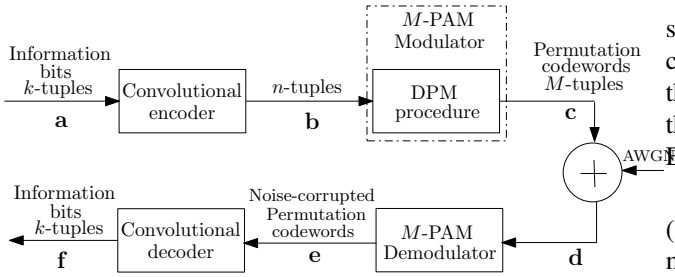

Fig. 3. System model used for the simulations.

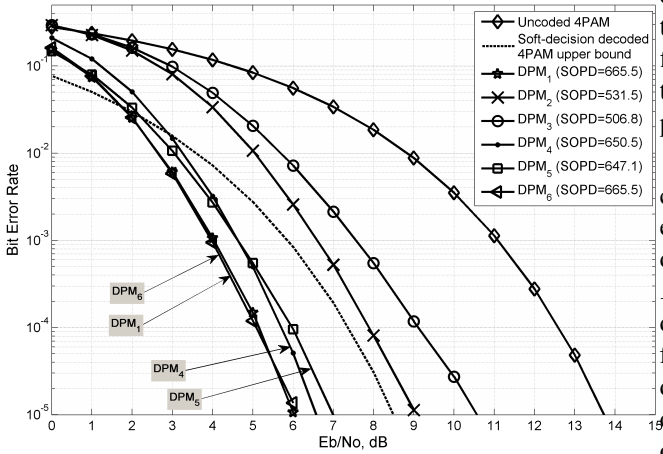

Fig. 4. Performance comparison results of six different mappings: DPM_1, DPM_2, DPM_3, DPM_4, DPM_5 and DPM_6. Uncoded 4-PAM and convolutional encoded soft-decision decoded 4-PAM results are also displayed. An ($R = 1/3$, $K = 4$, $d_{\text{free}} = 10$) convolutional code was used.

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

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

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

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

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.

Before we give simulation results for all the mappings (DPM_1–DPM_6), we discuss the use of SOPD as a performance measure. Given a fixed, positive number, say N, the conventional way to measure the improvement from N to another number, say N_i (where $N_i \geq N$) 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. 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 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 metric.

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 in Table I and the corresponding performance results in Fig. 4 of the mappings, it is evident that the SODD cannot be used to judge the performance of the mappings.

Looking at the E matrices for DPM_1 and DPM_2 in (2) and (3), 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 Hamming Distances	Sum of the Euclidean Distances	Sum of the difference of distances (SODD)	Sum of the product of distances (SOPD)
DPM_1	168	361.8	265.8	665.5
DPM_2	162	290.2	194.2	531.5
DPM_3	156	283.8	187.3	506.8
DPM_4	160	347.9	251.9	650.5
DPM_5	168	361.8	265.8	647.1
DPM_6	168	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 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 a 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.

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.

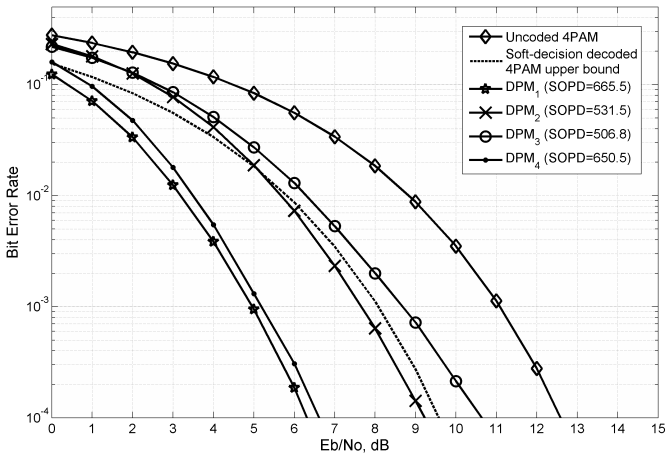

Fig. 5. Performance comparison results of four different mappings: DPM_1, DPM_2, DPM_3 and DPM_4. Uncoded 4-PAM and convolutional encoded soft-decision decoded 4-PAM results are also displayed. An ($R = 1/3$, $K = 3$, $d_{\text{free}} = 6$) convolutional code was used.

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.

V. 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, D. A. Wright, and A. L. Nel, "Hamming distance preserving mappings and trellis codes with constrained binary symbols," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1098–1103, Sept. 1989.

[2] 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. 24–28, 2000, pp. 2401–2407.

[3] 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.

[4] 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.

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

[6] J.-C. Chang, R.-J. Chen, T. Kløve, and S.-C. Tsai, "Distance-preserving 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 multilevel 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. 2, pp. 87–108, 2011.