



## **The Hadamard transform - a tool for index assignment**

Downloaded from: <https://research.chalmers.se>, 2026-04-18 13:43 UTC

Citation for the original published paper (version of record):

Knagenhjelm, P., Agrell, E. (1996). The Hadamard transform - a tool for index assignment. IEEE Transactions on Information Theory, 42(4): 1139-1151. <http://dx.doi.org/10.1109/18.508837>

N.B. When citing this work, cite the original published paper.

© 1996 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

# The Hadamard Transform - A Tool for Index Assignment

Petter Knagenhjelm, *Member, IEEE*, and Erik Agrell

## ABSTRACT

We show that the channel distortion due to noise on a binary symmetric channel is minimized for maximum entropy encoders, if the vector quantizer can be expressed as a linear transform of a hypercube. The index assignment problem is regarded as a problem of linearizing the vector quantizer. We define classes of index assignments with related properties, within which the best index assignment is found by sorting, not searching. Two powerful algorithms for assigning indices to the codevectors of non-redundant coding systems are presented. One algorithm finds the optimal solution in terms of linearity, whereas the other finds a very good, but suboptimal, solution in a very short time.

*Index terms:* Vector Quantization, Noisy Channel, Index Assignment, Robust Coding, Hadamard transform

Submitted to

IEEE Transactions on Information Theory

December 1993

First revision May 1995

Second revision January 1996

## I. INTRODUCTION

In this report we are concerned with non-redundant coding systems, i.e. coding without explicit error precaution. In digital communication over a noisy channel, other measures must be incorporated in the design to ensure robust transmission for such systems. The effect of channel errors on a zero-redundancy quantizer, can be severe if the codeword indices are not carefully selected. Codewords that interchange frequently may be far apart in signal space and cause large contributions to the overall distortion when transmission errors do occur. Robustification of non-redundant coding systems can be divided into two categories. Firstly, there are encoders where careful assignment of the codevector indices is the only precaution taken to mitigate the effects of channel errors. The labeling of the codevectors is often referred to as index assignment (IA) and the design of an encoder in this category can be separated into two steps: *i*) the source coding problem and *ii*) the index assignment problem. A coder designed in this manner is called a Robust Vector Quantizer (RVQ).

The other category consists of coders that are affected by the channel errors in their design and, hence, the final positioning of the reconstruction vectors depends on the channel error. The encoder-decoder system is trained, or optimized, to the channel error assumed at the time of training. Such a system will be called a Channel Optimized Vector Quantizer (COVQ). The usefulness of this approach may be restricted, firstly due to the increased training time needed for the design of the COVQ compared to an equally large encoder designed for a noiseless channel. Secondly, the channel characteristics must be well specified (i.e., the transition probabilities must be known) which can be hard, as the channel attributes probably will vary in time. However, the correctly designed COVQ will always outperform the RVQ. In this paper we address the index assignment problem which is, as mentioned above, an important part of the proper VQ design of the first category (RVQs). As there are  $M!$  ways to order  $M$  numbers, exhaustive search for the optimal ordering is not feasible and practical IA algorithms are (to this day) suboptimal. However, we identify that a robust quantizer needs structured index assignment and use the Hadamard transform to search for an assignment that is more robust than the initial. We define a fidelity criterion, highly correlated to the channel distortion, and present an algorithm which finds the optimal index assignment given an arbitrary vector quantizer. We then modify this algorithm to a practical and powerful tool for finding a suboptimal solution.

In Section II, we present the nomenclature employed in the report and introduce the concept of using the Hadamard matrix for codebook description. The channel distortion of a maxi-

mum entropy encoder transmitting over a binary symmetric channel is stated in Section III and in Section IV, it is shown that the channel distortion is minimized if the codebook can be expressed as a linear transform of a hypercube. The index assignment problem is then regarded as a problem of linearizing the codebook. In Section V, we decompose the index assignments into categories with similar features, and in Section VI we state the properties of random assignment and full search assignment. We show in Section VII that the set of all index assignments can be divided into classes, and that starting with an arbitrary assignment, the best index assignment within the class is easily found by a sorting procedure. An algorithm to successively generate one member in each class is provided in Section VIII. A practical approach called LISA, not visiting all Hadamard classes, is described in Section IX. To explore the performance of the algorithm, experimental results, including comparisons to established methods, are given in Section X. Section XI, finally, summarizes some conclusions.

#### *Related work*

Totty and Clark [1] showed that the total squared error distortion in scalar quantizing can be divided into a source distortion term and a channel distortion term provided the quantization is optimal. This separation lead to an awareness that the channel distortion term can be coped with by a proper index assignment. The problem of assigning  $N$  elements in one set to  $M$  elements in another set in general, and the classical Quadratic Assignment Problem in particular, are predecessors to the Index Assignment problem and hence, much background literature can be found. Potter [2] parallels the index assignment to the Quadratic Assignment Problem stated by Koopmans and Beckmann in 1957. An early reference on the importance of having a good IA is Rydbeck and Sundberg [3] addressing the case of scalar quantizers. Extensions of index assignment to vector quantizers were introduced in [4-8]. Cheng and Kingsbury [9] point out that the LBG-split algorithm [10] gives a better IA than average, measured in terms of SNR performance. Zeger and Gersho [8] compute a cost function which governs a binary index-swap strategy while Farvardin employs in [6] a simulated annealing algorithm to the problem. Knagenhjelm [11] uses the Hadamard transform to derive an objective measure on the success of the index assignment. Hagen and Hedelin [12] employ the Hadamard transform in the VQ design to get a suboptimal, but robust, VQ. McLaughlin et al. [13] show that the Natural Binary Code is optimal for uniform scalar quantizers and uniform source distribution and the work is extended to vector quantizers in [14]. Recently Chiang and Potter [15] presented a paper where a minimax criterion is used instead of the usual MSE criterion in the channel coding. This improves the worst case performance and the minimax criterion is

shown to be a useful objective for the perception of images. Wu and Barba exploit the unequal apriori probabilities in a fast, greedy algorithm in [16]. An in depth discussion on transmitting VQ data over a noisy channel is found in [17, 18].

## II. PRELIMINARIES

We here state the problem of index assignment, introduce the transformed hypercube description of the codebook and recall some matrix algebra.

### A. Problem statement

We will address the problem of assigning indices to the codevectors of a vector quantizer to reduce the effects of channel errors when the codewords are transmitted on a noisy, memoryless, Binary Symmetric Channel (BSC). A  $d$ -dimensional vector  $\mathbf{X}$  is fed to the quantizer producing a  $k$ -bit binary *codeword* for transmission. The codeword is said to be the index of the vector used for signal reproduction at the receiver. For convenience, the index is regarded as an integer where the corresponding base 2 representation of the integer is the codeword. The set of  $M = 2^k$  vectors defines the *codebook*. The encoder is said to be a *Maximum Entropy Encoder* (MEE), if the entropy is full, i.e.  $H(\hat{\mathbf{X}}) = k$ , where  $\hat{\mathbf{X}}$  denotes the random output from the vector quantizer. Of course, full entropy is only reached when all codewords are used with equal probability. In this paper we will only address such encoders, which facilitates the distortion analysis, but represents a difficult task for the Index Assignment, as we can not focus on finding a good index assignment for just a subset of frequently used codewords. The *decoder* receives a codeword  $j$ , and produces a  $d$ -dimensional reconstruction vector  $\mathbf{y}_j$  as output. Here we adopt the name *VQ point* for this vector, and thereby regard the codebook as a set of points in the  $d$ -dimensional signal space. A codebook with  $M$  reconstruction vectors is said to be an  $M$ -point codebook.

The total distortion related to the quantization and transmission of information can be divided into three terms as

$$D = D_{\text{Source}} + D_{\text{Channel}} + D_{\text{Mixed}}, \quad (1)$$

where  $D_{\text{Source}}$  is the distortion due to quantizing effects only and  $D_{\text{Channel}}$  is the distortion caused by misinterpretation of the received codeword, and hence,  $D_{\text{Channel}}$  is dependent on the index assignment. The mixed term of (1) is zero for optimum codebooks designed for noiseless channels [1, 6] since then the VQ points are the centroids of the reconstruction cells. The term is also zero in the limit when the number  $M$  approaches infinity even when the VQ

points are not positioned in the centroids [19]. In this paper we restrict ourselves to study the effects the index assignment on the channel distortion,  $D_C$  for short.

The channel distortion, using the squared error distortion measure, associated with choosing the index  $i$  as a representative for a sample is

$$D_{C_i} = \sum_{j=0}^{M-1} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \cdot p_{j|i}, \quad (2)$$

where  $p_{j|i}$  is the probability of receiving the index  $j$  given that index  $i$  was sent, i.e.  $p_{j|i} = P[J = j|I = i]$ . As the encoder is assumed to have maximum entropy, the final expression to minimize is

$$D_C = \frac{1}{M} \cdot \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \cdot p_{j|i}. \quad (3)$$

The minimization of (3) is the problem of index assignment. Suppose that a codebook,  $\mathbf{Y}^0 = [\mathbf{y}_0^0 \ \mathbf{y}_1^0 \ \cdots \ \mathbf{y}_{M-1}^0]$ , is given. There exists a class of related codebooks having the same VQ points, but in another order. To describe the ordering of the VQ points, we define a row vector  $\mathbf{g} = [g_0 \ g_1 \ \cdots \ g_{M-1}]$ , called the *index assignment*. This vector, which is any permutation of  $\mathbf{g}^0 = [0 \ 1 \ \cdots \ M-1]$ , contains the codewords (integers) to be associated with the corresponding VQ points in  $\mathbf{Y}^0$ . The reordered codebook is called  $\mathbf{Y}$ , and its vectors are thus  $\mathbf{y}_{g_i} = \mathbf{y}_i^0$ ;  $i = 0, \dots, M-1$ . Sometimes it is more appropriate to regard the codewords as bit sequences. We therefore define the *index assignment matrix*

$$\mathbf{G} = [\mathbf{b}(g_0) \ \cdots \ \mathbf{b}(g_{M-1})], \quad (4)$$

where  $\mathbf{b}(i)$  is the  $k$ -dimensional binary column vector describing the base 2 representation of the integer  $i$ . Its elements will be called  $b_j(i)$ . The inverse of the function  $\mathbf{b}(\cdot)$  is a scalar product with the binary vector  $\mathbf{c}^T = [2^{k-1} \ \cdots \ 2^0]$ , so that  $\mathbf{c}^T \mathbf{b}(i) = i$ .

### B. Transformed hypercubes

The distribution of the VQ points in signal space depends, of course, on the source distribution, but in the case of COVQs, also on how likely indices are to be confused with other indices when transmitted on a noisy channel. The probability of confusing codewords is again dependent on the indices given to the VQ points and is best expressed by the Hamming distance between the indices. A way of combining the logical description (i.e. indices) and the physical description (i.e. position in the signal space) is to move the logical hypercube, spanned by the codebook's binary codewords, to the signal space with preserved topology. For convenience, we replace the logical zeros and ones with +1 and -1 respectively, and de-

note by  $b'_j(i) = 1 - 2b_j(i)$  the  $j$ th bit of the positive integer  $i$ . The  $M$  VQ points can then be described by a transform, or a mapping, of the  $M$  vertices of the hypercube (see Fig. 1). As soon will become clear, the Hadamard transform is especially suitable for this task.

In general, any scalar, nonlinear function  $y(b_0, \dots, b_{k-1})$  can be expressed as a discrete Volterra series [20]

$$y = t + \sum_{i_1=0}^{k-1} t_{i_1} b_{i_1} + \sum_{i_1=0}^{k-1} \sum_{i_2=0}^{k-1} t_{i_1, i_2} b_{i_1} b_{i_2} + \dots + \sum_{i_1=0}^{k-1} \sum_{i_2=0}^{k-1} \dots \sum_{i_L=0}^{k-1} t_{i_1, i_2, \dots, i_L} b_{i_1} b_{i_2} \dots b_{i_L} + \dots, \quad (5)$$

where  $t$  is the *offset*, the first sum is the *linear* part, and the remaining sums constitute the *nonlinear* part of the function. Let  $y$  be the scalar describing one dimension of an arbitrary VQ point  $\mathbf{y}_i$ , and  $b_i$  the bits in the binary representation of the index associated with the VQ point. In this particular case we have that  $b_j = b'_j(i) \in \{\pm 1\}$ ,  $\forall i, j$ , thus the series expansion folds into itself, and only  $2^k$  unique elements remain for a  $k$ -bit codebook [21]. For example, with two nonlinear functions,  $y_i^{(1)}$  and  $y_i^{(2)}$ , we can express one of the eight 2-dimensional VQ points of a 3-bit codebook as

$$\mathbf{y}_i = \begin{bmatrix} y_i^{(1)} \\ y_i^{(2)} \end{bmatrix} = \begin{bmatrix} t_0^{(1)} & t_1^{(1)} & t_2^{(1)} & t_3^{(1)} & t_4^{(1)} & t_5^{(1)} & t_6^{(1)} & t_7^{(1)} \\ t_0^{(2)} & t_1^{(2)} & t_2^{(2)} & t_3^{(2)} & t_4^{(2)} & t_5^{(2)} & t_6^{(2)} & t_7^{(2)} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ b'_0(i) \\ b'_1(i) \\ b'_0(i) \cdot b'_1(i) \\ b'_2(i) \\ b'_0(i) \cdot b'_2(i) \\ b'_1(i) \cdot b'_2(i) \\ b'_0(i) \cdot b'_1(i) \cdot b'_2(i) \end{bmatrix} = \mathbf{T} \cdot \mathbf{h}_i. \quad (6)$$

We have rearranged the order in which the linear and nonlinear coefficients appear in (5) so that the full codebook,  $\mathbf{Y} = [\mathbf{y}_0 \ \mathbf{y}_1 \ \dots \ \mathbf{y}_{M-1}]$ , can be represented as

$$\mathbf{Y} = \mathbf{T} \cdot \mathbf{H} \quad (7)$$

where the  $M \times M$  matrix  $\mathbf{H} = [\mathbf{h}_0 \ \mathbf{h}_1 \ \dots \ \mathbf{h}_{M-1}]$  is a normalized (Sylvester style) Hadamard matrix. The  $d \times M$  matrix  $\mathbf{T}$  is the Hadamard *transform* of  $\mathbf{Y}$ . Sometimes we refer to  $\mathbf{T} = [\mathbf{t}_0 \ \mathbf{t}_1 \ \dots \ \mathbf{t}_{M-1}]$  as the *transform matrix* and the vectors,  $\mathbf{t}_i$ , as *transform vectors*. In agreement with (5), we call  $\mathbf{t}_0$  the codebook offset,  $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_4, \dots, \mathbf{t}_{M/2}$  the linear part of the transform and the remaining  $\mathbf{t}$ -vector the non-linear part. A codebook for which the non-linear part is zero is a *linear* codebook.

A Sylvester style Hadamard matrix [22] is generated by the simple recursion

$$\mathbf{H}_{2n} = \begin{bmatrix} \mathbf{H}_n & \mathbf{H}_n \\ \mathbf{H}_n & -\mathbf{H}_n \end{bmatrix}$$

with  $\mathbf{H}_1 = \mathbf{1}$ . Its elements  $h_{i,j}$  are

$$h_{i,j} = (-1)^{\mathbf{b}^T(i)\mathbf{b}(j)} = \prod_l (-1)^{b_l(i)b_l(j)} \quad (8)$$

and the elements of the columns satisfy

$$\mathbf{h}_i \circ \mathbf{h}_j = \mathbf{h}_{i \oplus j} \quad (9)$$

where  $\circ$  denotes the Schur product<sup>1</sup>, and  $\oplus$  the bitwise modulo-2 addition. A Hadamard matrix is a symmetrical matrix with orthogonal rows (and columns), so  $\mathbf{H}$  is its own inverse (except for a scaling factor  $1/M$ ). The inverse Hadamard transform is thus

$$\mathbf{T} = \frac{1}{M} \mathbf{Y} \cdot \mathbf{H}, \quad (10)$$

which shows that there exists a  $\mathbf{T}$  matrix for an *arbitrary* codebook  $\mathbf{Y}$ .

In Section IV, we will demonstrate that for maximum entropy encoders it is advantageous, from a channel-robustness viewpoint, if the transform is linear. However, a linear transform imposes a too restrictive constraint on the positioning of the VQ points to achieve the optimal *source coder*. In order to abridge the concept of linear and nonlinear transforms, we introduce in Section IV a measure called the linearity index, which equals 1 for linear transforms, and is less than 1 for nonlinear transforms. By decomposing the transform into linear and nonlinear parts as above, we strive, by changing the index assignment of the codebook, to find transforms where the linear part is dominant.

Although the use of the Hadamard transform is extensive in various applications like generating nonlinear error correcting codes (see for example [22]), speech coding [23], image coding [24], and echo cancellation [21], the use of the Hadamard matrix for general codebook description, was, to the best of our knowledge, introduced quite recently [11, 12].

### C. Matrix algebra

We will review some ideas from matrix algebra. The following terminology and theory is extracted from Birkhoff and Mac Lane [25, p. 177] For a more substantial presentation of matrix algebra, the reader is referred to that source.

An *elementary row operation* belongs to one of the three following types:

---

<sup>1</sup> The Schur product is defined as  $[\mathbf{A} \circ \mathbf{B}]_{i,j} = a_{i,j} \cdot b_{i,j}$

- (1) The interchange of any two rows,
- (2) multiplication of a row by any non-zero constant  $c$ ,
- (3) the addition of any multiple of one row to any other row.

The  $m \times n$  matrix  $\mathbf{B}$  is called *row-equivalent* to the  $m \times n$  matrix  $\mathbf{A}$  if  $\mathbf{B}$  can be obtained from  $\mathbf{A}$  by a finite succession of elementary row operations. If  $\mathbf{B}$  is row-equivalent to  $\mathbf{A}$ , then  $\mathbf{A}$  is row-equivalent to  $\mathbf{B}$ .

The first non-zero entry in a row is called a *leading entry* of that row. A matrix  $\mathbf{R}$  which satisfies the following four conditions is called a *reduced echelon matrix*:

- (1) Every leading entry (of a non-zero row) is 1,
- (2) Every *column* containing such a leading entry 1 has all its other entries zero,
- (3) Each zero row of  $\mathbf{R}$  comes below all non-zero rows of  $\mathbf{R}$ ,
- (4) Suppose that there are  $r$  non-zero rows, and that the leading entry of row  $i$  appears in column  $s_i$ , for  $i = 1, \dots, r$ . These numbers are ordered such that  $s_1 < s_2 < \dots < s_r$ .

We will apply this theory to the index assignment matrices. Such a matrix has always the size  $k \times M$ , and all its columns are different. The appropriate field is GF(2), which means that the only numbers are 0 and 1, and addition is carried out modulo 2. In this special case, the theory can be simplified somewhat, which will be done in Section VIII.

### III. THE CHANNEL DISTORTION OF A BINARY SYMMETRIC CHANNEL

It is sometimes convenient to relate the channel distortion to something that is invariant under all index assignments as, for instance, the sum of the norms of the reconstruction vectors,  $\mathbf{y}_i$ . Using Parseval's formula, we define a "variance" of the codebook,  $\sigma_{\text{VQ}}^2$ , and express it in terms of the transform vectors, as

$$\sigma_{\text{VQ}}^2 \triangleq \frac{1}{M} \cdot \sum_{i=0}^{M-1} \|\mathbf{y}_i\|^2 - \left\| \frac{1}{M} \sum_{i=0}^{M-1} \mathbf{y}_i \right\|^2 = \sum_{i=0}^{M-1} \|\mathbf{t}_i\|^2 - \|\mathbf{t}_0\|^2 = \sum_{i=1}^{M-1} \|\mathbf{t}_i\|^2. \quad (11)$$

As  $\mathbf{h}_0$  contains ones only, we have, using (10), that  $\mathbf{t}_0 = \frac{1}{M} \sum \mathbf{y}_i$ . In other words,  $\mathbf{t}_0$  is the offset vector and typically equals zero for codebooks trained on symmetric, zero-mean sources. The channel distortion (3) can conveniently be expressed by means of the norms of the  $\mathbf{t}$ -vectors as stated in the following.

**Theorem 1:** The channel distortion for an  $M$  point maximum entropy encoder due to noise on a binary symmetric channel with bit error probability  $q$ , is

$$D_C = 2\sigma_{\text{VQ}}^2 - 2 \sum_{i=1}^{M-1} (1-2q)^{w_h(i)} \cdot \|\mathbf{t}_i\|^2, \quad (12)$$

where  $w_h(i)$  is the Hamming weight of the base 2 representation of  $i$ .

**Proof of Theorem 1:** Let the random integer  $Q$  represent the bit error pattern for a  $k$ -bit word transmitted over the BSC. The channel distortion is

$$\begin{aligned} D_C &= \frac{1}{M} \cdot \sum_{i=0}^{M-1} E \left[ \|\mathbf{y}_i - \mathbf{y}_{i \oplus Q}\|^2 \right] \\ &= \frac{1}{M} \cdot \sum_{i=0}^{M-1} E \left[ \sum_{j=0}^{M-1} \mathbf{t}_j^T (h_{i,j} - h_{i \oplus Q,j}) \cdot (\mathbf{y}_i - \mathbf{y}_{i \oplus Q}) \right]. \end{aligned}$$

Expanding the product yields, using (9)

$$\begin{aligned} D_C &= \frac{1}{M} \cdot \sum_{j=0}^{M-1} \sum_{i=0}^{M-1} \mathbf{t}_j^T E \left[ \mathbf{y}_i h_{i,j} - \mathbf{y}_i h_{i,j} h_{Q,j} - \mathbf{y}_{i \oplus Q} h_{i \oplus Q,j} h_{Q,j} + \mathbf{y}_{i \oplus Q} h_{i \oplus Q,j} \right] \\ &= \sum_{j=0}^{M-1} \mathbf{t}_j^T E \left[ \mathbf{t}_j - \mathbf{t}_j h_{Q,j} - \mathbf{t}_j h_{Q,j} + \mathbf{t}_j \right] \\ &= 2 \cdot \sum_{j=1}^{M-1} \|\mathbf{t}_j\|^2 E \left[ 1 - h_{Q,j} \right]. \\ &= 2\sigma_{\text{VQ}}^2 - 2 \cdot \sum_{j=1}^{M-1} \|\mathbf{t}_j\|^2 E \left[ h_{Q,j} \right] \end{aligned} \quad (13)$$

The expectation is (8)

$$E \left[ h_{Q,j} \right] = E \left[ \prod_{l=0}^{k-1} (-1)^{b_l(Q) b_l(j)} \right],$$

where both the product and the exponent,  $b_l(j)$ , can be moved outside the expectation; the product because the bit errors occur independently, and the exponent because it is either 0 or 1. Thus,

$$E \left[ h_{Q,j} \right] = \prod_{l=0}^{k-1} E \left[ (-1)^{b_l(Q)} \right]^{b_l(j)} = \prod_{l=0}^{k-1} \left( (1-q) \cdot (+1) + q \cdot (-1) \right)^{b_l(j)} = (1-2q)^{w_h(j)},$$

which inserted into (13) completes the proof. □

Equation (12) is, in spite of its simplicity, a powerful equation, as it expresses the performance of a fully trained codebook, with maximum entropy and a certain IA, for an arbitrary channel bit error rate. The theory is generalized to non-MEEs in [17].

#### IV. THE RELATION BETWEEN DISTORTION AND LINEARITY

In this section we show that for a Maximum Entropy Encoder, MEE, a very good index assignment is the one yielding the most linear transform of the hypercube.

As discussed in Section IIB, the  $\mathbf{t}$ -vectors associated with indices with Hamming weight one are of special interest, as they represent the linear part of the transform. In the sequel we find it convenient to employ a measure on how dominant the linear part of the general nonlinear transform in (7) is. We define the *linearity index*,  $\lambda$ , as

$$\lambda \triangleq \frac{1}{\sigma_{\text{VQ}}^2} \cdot \sum_{l=0}^{k-1} \|\mathbf{t}_{2^l}\|^2 \quad (14)$$

where  $\sigma_{\text{VQ}}^2$  is the sum (11) of the norms of  $\mathbf{t}_i$ ,  $i = 1, \dots, M-1$ . The range of  $\lambda$  is readily seen to be  $0 \leq \lambda \leq 1$  where  $\lambda = 1$  denotes a purely linear transform.

We now give bounds on the channel distortion followed by an important corollary.

**Theorem 2:** The distortion due to noise on a binary symmetric channel with bit error probability  $q$  is, for a codebook with linearity index  $\lambda$ , bounded by

$$2\sigma_{\text{VQ}}^2 \left[ 2q\lambda + (1 - (1 - 2q)^2) \cdot (1 - \lambda) \right] \leq D_C \leq 2\sigma_{\text{VQ}}^2 \left[ 2q\lambda + (1 - (1 - 2q)^k) \cdot (1 - \lambda) \right]$$

**Proof of Theorem 2:** Separating the sum in (12) into one when  $i$  is a power of 2, and one for the remaining values of  $i$ , the latter sum can be bounded by employing the inequality  $2 \leq w_h(i) \leq k$ . Insertion of (14), using (11), gives the theorem. □

**Corollary:** If there exists an index assignment giving  $\lambda = 1$ , then it is optimal and the channel distortion is

$$D_{C\text{min}} = 4 \cdot q \cdot \sum_{i=1}^{M-1} \|\mathbf{t}_i\|^2 = 4 \cdot q \cdot \sigma_{\text{VQ}}^2 \quad (15)$$

The parameter  $\lambda$  is therefore an indicator of how good an index assignment is, but it gives no information on how to find the optimal index assignment.

Equation (14) is attractive as it can be understood intuitively and may give a valuable insight into the structure of robust VQs. It is also useful in the index assignment algorithms presented in subsequent sections. Figure 2 relates the measurement of the linearity index to the ‘‘amount of linearity’’ in the mapping. The more ‘‘tidy structure’’ of Fig. 2 (b) compared to Fig. 2 (a) is a consequence of a more linear mapping of the hypercube. Unfortunately, the distortion is not a monotonic function of the linearity index. Figure 3 (a) shows a scatter plot of the channel

distortion as a function of  $\lambda$  together with the bounds of theorem 2. The corresponding histogram for random index assignments is depicted in Fig. 3 (b). The figure in (a) shows that a range of values of the channel distortion exists for a single value of the linearity index  $\lambda$ , and hence, maximizing  $\lambda$  is not equivalent to minimizing  $D_C$ . Still the correlation between the channel distortion and the linearity index is very strong, and fortunately strongest for the IAs of most interest, that is, for low  $D_C$ . This means that the linearity index is a potent parameter in the search for robustness. The histogram of the measure shows that very few index assignments result in a high linearity index.

It is worth noting that the linearity index (14) is independent of the channel bit error probability,  $q$ . This feature is attractive since it eliminates the need to design the codebook for a certain fixed value of a parameter that normally varies with time. Another approach to attain this advantage is through the assumption that only single bit-errors occur [6, 8].

We have seen that the channel distortion of an arbitrary MEE working over a BSC, is minimized if the codebook can be expressed as a linear transform of the codeword hypercube. Furthermore, it is conjectured, and confirmed by an example, that the correlation between the distortion and linearity is strong and, hence, the search for a good index assignment may be transformed to a search for a high linearity index. We now turn to the index assignment problem and specifically present the theory behind an algorithm to achieve a good index assignment under the constraint that we seek as linear transforms as possible.

## V. DECOMPOSITION OF AN INDEX ASSIGNMENT

In order to find a good index assignment, a time-saving trick is to identify assignments having related properties. Using such knowledge, we can evaluate whole classes of assignments simultaneously. Our first observation is that any  $k \times M$  matrix,  $\mathbf{G}$ , can be uniquely factorized as

$$\mathbf{G} = \mathbf{L}\mathbf{G}_e ,$$

where  $\mathbf{L}$  is a  $k \times k$  matrix and  $\mathbf{G}_e$  is a reduced echelon matrix [25]. If the rows of  $\mathbf{G}$  are linearly independent, as they are for any index assignment matrix (4), then so are the rows of  $\mathbf{L}$ . Second, because an IA matrix,  $\mathbf{G}$ , consists of the columns  $\{\mathbf{b}(0), \dots, \mathbf{b}(M-1)\}$ , though not necessarily in that order, the matrix  $\mathbf{G} \oplus \mathbf{z}\mathbf{1}^T$ , where  $\mathbf{1}$  is the column vector of  $M$  ones, will contain the same vectors, for any  $k$ -dimensional vector  $\mathbf{z}$ . (Remember that the field is GF(2)). Specifically, we may set  $\mathbf{z} = \mathbf{b}(g_0)$ , which yields

$$\mathbf{G} = \mathbf{G}_z \oplus \mathbf{z}\mathbf{1}^T$$

where  $\mathbf{G}_z$  has the zero vector in the first column. This is equivalent to assigning the zero codeword to an arbitrary VQ point and relate all distances to this VQ point. Combining the two decompositions of  $\mathbf{G}$  into one, we have proved the following theorem.

**Theorem 3:** Any IA matrix,  $\mathbf{G}$ , can be decomposed as

$$\mathbf{G} = \mathbf{L}\mathbf{G}_{ez} \oplus \mathbf{z}\mathbf{1}^T \quad (16)$$

in one and only one way, where  $\mathbf{G}_{ez}$  is a reduced echelon IA matrix with an initial zero vector,  $\mathbf{L}$  is an invertible  $k \times k$  matrix, and  $\mathbf{z}$  is a  $k$ -dimensional vector.

Any IA can be uniquely specified through its triplet  $(\mathbf{G}_{ez}, \mathbf{L}, \mathbf{z})$ . There are

$$M_\lambda = \frac{M!}{M \cdot \prod_{l=0}^{k-1} (M - 2^l)} \quad (17)$$

possible values for  $\mathbf{G}_{ez}$  (see Section VIII),

$$M_l = \prod_{l=0}^{k-1} (M - 2^l) \quad (18)$$

values for  $\mathbf{L}$ , and  $M$  values for  $\mathbf{z}$ . Table I gives numerical values of  $M_\lambda$  and  $M_l$  for some  $k$ . It turns out that it is not necessary to examine all combinations of  $\mathbf{G}_{ez}$ ,  $\mathbf{L}$ , and  $\mathbf{z}$ . In the following sections, we will present an algorithm to find one of the best  $\mathbf{L}$  matrices *without* generating all of them, and another algorithm to generate all  $\mathbf{G}_{ez}$  matrices. We will not generate more than one value of  $\mathbf{z}$ ; this vector does not affect the distortion for a binary symmetric channel, since the probability of confusing codewords  $\mathbf{b}(i)$  and  $\mathbf{b}(j)$  is the same as that of confusing  $\mathbf{b}(i) \oplus \mathbf{z}$  and  $\mathbf{b}(j) \oplus \mathbf{z}$ .

## VI. UNCONSTRAINED INDEX ASSIGNMENT

In this section we state some properties of random index assignment and full search index assignment to get a perspective on the constrained assignment presented in Section VIII. The methods are trivial and extreme cases of index assignment; the first being fast but bad, and the second being slow but good. The goal in index assignment research, is to find a compromise between the two. Note that the positioning of the VQ points is not affected by any IA procedure.

### A. Assigning Indices Randomly

For a random assignment of the indices, the expected norms of the transform vectors in the numerator and denominator of (14) are equal. Hence, the expected value of  $\lambda$  over all index assignments becomes

$$E[\lambda] = \frac{k}{M-1}. \quad (19)$$

The expected value decreases rapidly with increasing number of bits in the codebook. The mean of the histogram in Fig. 3 (b) for the 4-bit codebook is accurately predicted by  $4/15 \approx 0.267$ .

Another interpretation of (19) is that the probability of finding a good IA by random search strategies approaches zero for increasing  $k$ . The expected channel distortion for an MEE at low bit error probability and a randomly chosen index assignment can be shown to be [6]

$$\bar{D}_C = 2 \cdot q \cdot k \cdot \frac{M}{M-1} \cdot \sigma_{vQ}^2, \quad (20)$$

which is roughly  $k/2$  times the distortion for a codebook with a good IA.

### B. Full Search Index Assignment

As there are  $M!$  ways to order  $M$  elements, exhaustive search belongs to a class of problems called NP-complete, i.e., no solution is reached in reasonable (polynomial) time. The  $M!$  possibilities needed for an unconstrained search are reduced by identifying several solutions yielding identical distortion. We may add any vector  $\mathbf{b}(i); i = 0, \dots, M-1$  to the vector representation of the indices, and we may permute the bits. This corresponds to selecting an arbitrary  $\mathbf{z}$  and permuting the rows of  $\mathbf{L}$ . To perform a full search, i.e. just as good as exhaustive, we thus have to examine

$$M_F = \frac{M!}{M \cdot k!} \quad (21)$$

possibilities. Table I shows that this is only feasible for codebook sizes up to 3 or 4 bits.

## VII. HADAMARD CLASSES

In this section, we employ the decomposition of Section V to demonstrate that index assignments of a codebook can be categorized into groups or classes, which we call *Hadamard classes*.

**Definition:** A Hadamard class is the set of index assignments having the same reduced echelon matrix  $\mathbf{G}_{ez}$ .

In a Hadamard class, all elements (IAs) can be generated from an arbitrary element within the class by elementary matrix manipulations. Below we will show how to generate the elements in a Hadamard class and how to find the element with maximum linearity by a simple sorting procedure.

It is readily understood from (7) that for a fixed Hadamard matrix  $\mathbf{H}$ , a reordering of the columns in  $\mathbf{Y}$  must alter the transform matrix  $\mathbf{T}$ , and that some of the alterations may result in a higher linearity index  $\lambda$ . Conversely, at least some reorderings of the  $\mathbf{t}$ -vectors should affect just the index assignment, whereas other reorderings of  $\mathbf{t}$ -vectors are not allowed, since the codevectors  $\{\mathbf{y}_i\}$  would be repositioned. Knowing that a high linearity index is favorable in terms of robustness, we may try to increase the linearity by reordering the indices of the  $\mathbf{t}$ -vectors so that the  $\mathbf{t}$ -vectors with large norms correspond to the linear part. First we show that some reorderings of the  $\mathbf{t}$ -vectors indeed are reorderings of the  $\mathbf{y}$ -vectors, which may not be immediately obvious. We regard the equivalent problem of reordering the indices instead of the vectors, and especially study reorderings that may be expressed by multiplying (in GF(2)) each binary representation of the index by a binary  $k \times k$  matrix  $\mathbf{D}$ . If we reorder  $\mathbf{t}_j$  to become  $\mathbf{t}'_j = \mathbf{t}_{\mathbf{c}^T \mathbf{D} \mathbf{b}(j)}$ , the effect on  $\mathbf{y}_i$  is, according to (8),

$$\mathbf{y}'_i = \sum_{j=0}^{M-1} \mathbf{t}_{\mathbf{c}^T \mathbf{D} \mathbf{b}(j)} \cdot (-1)^{\mathbf{b}^T(j) \cdot \mathbf{b}(i)}. \quad (22)$$

Substituting  $\mathbf{D}^{-1} \mathbf{b}(l)$  for  $\mathbf{b}(j)$ ,

$$\begin{aligned} \mathbf{y}'_i &= \sum_{l=0}^{M-1} \mathbf{t}_{\mathbf{c}^T \mathbf{D} \mathbf{D}^{-1} \mathbf{b}(l)} \cdot (-1)^{\mathbf{b}^T(l) \cdot (\mathbf{D}^{-1})^T \mathbf{b}(i)} \\ &= \sum_{l=0}^{M-1} \mathbf{t}_l \cdot (-1)^{\mathbf{b}^T(l) \cdot \mathbf{b}(\mathbf{c}^T (\mathbf{D}^{-1})^T \mathbf{b}(i))} \\ &= \mathbf{y}_{\mathbf{c}^T (\mathbf{D}^{-1})^T \mathbf{b}(i)}. \end{aligned} \quad (23)$$

This means that if we reorder the indices of  $\mathbf{T}$  using the matrix  $\mathbf{D}$ , we reorder the indices of  $\mathbf{Y}$  according to  $(\mathbf{D}^{-1})^T$  (and vice versa), provided that  $\mathbf{D}$  is invertible. For  $\mathbf{D}$  to be invertible, all rows must be linearly independent.

Now we know that this type of  $\mathbf{t}$ -vector reordering indeed corresponds to another index assignment for the *same* VQ points. To prove that the two IAs always belong to the same Hadamard class, we examine the IA matrices in terms of the triplet  $(\mathbf{G}_{ez}, \mathbf{L}, \mathbf{z})$ . The new IA matrix is

$$\begin{aligned}\mathbf{G}' &= [\mathbf{b}(g'_0) \cdots \mathbf{b}(g'_{M-1})] = [(\mathbf{D}^{-1})^T \mathbf{b}(g_0) \cdots (\mathbf{D}^{-1})^T \mathbf{b}(g_{M-1})] = (\mathbf{D}^{-1})^T \mathbf{G} \\ &= (\mathbf{D}^{-1})^T \mathbf{L} \mathbf{G}_{ez} + (\mathbf{D}^{-1})^T \mathbf{z} \mathbf{1}^T\end{aligned}\quad (24)$$

and thus the new triplet  $(\mathbf{G}'_{ez}, \mathbf{L}', \mathbf{z}') = (\mathbf{G}_{ez}, (\mathbf{D}^{-1})^T \mathbf{L}, (\mathbf{D}^{-1})^T \mathbf{z})$ . Since  $\mathbf{G}_{ez}$  is not affected, the new IA is, by definition, within the same Hadamard class.

The Hadamard transform of one codebook (i.e.  $\mathbf{T}$ ) has the same columns, but in a different order, as the Hadamard transform of other IAs within the same Hadamard class. If the objective is to minimize the distortion directly, this would not help us much. However, in order to maximize the linearity index, all we have to do is to locate the  $k$   $\mathbf{t}$ -vectors with largest norms. Then the whole class,  $M_l \cdot M$  index assignments, has been handled simultaneously. The only restriction is that the indices of the  $k$   $\mathbf{t}$ -vectors must be linearly independent. The following sorting algorithm generates the matrix  $\mathbf{D}$  which can be used to reorder  $\mathbf{T}$  to achieve higher linearity index.

**Input:** A codebook, specified by its Hadamard transform  $\mathbf{T}$ .

**Output:** The best IA within the class, specified by the reordering matrix  $\mathbf{D} = [\mathbf{d}_{k-1} \cdots \mathbf{d}_0]^T$ .

**Sorting Algorithm:**

$$\mathcal{A} := \{\mathbf{b}(1), \dots, \mathbf{b}(M-1)\}$$

For  $l := 0$  to  $k-1$ :

$$\text{Find } \mathbf{d}_l := \arg \max_{\mathbf{d} \in \mathcal{A}} \|\mathbf{t}_{\mathbf{c}^T \mathbf{d}}\|^2$$

For  $i := 2^l$  to  $2^{l+1} - 1$ :

Remove  $\mathbf{D}\mathbf{b}(i)$  from  $\mathcal{A}$

The reordered version of the matrix  $\mathbf{T}$  is the matrix  $[\mathbf{t}_{\mathbf{c}^T \mathbf{D}\mathbf{b}(0)} \cdots \mathbf{t}_{\mathbf{c}^T \mathbf{D}\mathbf{b}(M-1)}]$ , which, after taking the inverse Hadamard transform, gives a codebook with a higher linearity index than the original (if possible). The importance of the algorithm is due to the following theorem.

**Theorem 4:** The reordering matrix  $\mathbf{D}$  generated by the sorting algorithm produces the highest linearity index within a Hadamard class.

**Proof of Theorem 4:** Sort the  $\mathbf{t}$ -vectors according to decreasing norms by introducing  $s_1, s_2, \dots, s_{M-1}$  so that  $\|\mathbf{t}_{s_1}\|^2 \geq \|\mathbf{t}_{s_2}\|^2 \geq \dots \geq \|\mathbf{t}_{s_{M-1}}\|^2$ . Let  $q_i$  be the smallest integer such that  $\mathbf{b}(s_1), \dots, \mathbf{b}(s_{q_i})$  span an  $(i+1)$ -dimensional space, for  $i = 0, \dots, k-1$ . The problem is to pick  $k$  integers  $d_0, \dots, d_{k-1}$  to maximize  $\sum_l \|\mathbf{t}_{d_l}\|^2$  under the restriction that  $\mathbf{b}(d_0), \dots, \mathbf{b}(d_{k-1})$  are linearly independent. Without loss of generality we may let  $d_0 < \dots < d_{k-1}$ . At most  $l$  elements of  $\{\mathbf{b}(s_1), \dots, \mathbf{b}(s_{q_{l-1}})\}$  are linearly independent, so  $d_l \in \{\mathbf{b}(s_{q_l}), \dots, \mathbf{b}(s_{M-1})\}; \forall l$ . Thus,  $\sum_l \|\mathbf{t}_{d_l}\|^2$  is maximized when  $d_l = s_{q_l}; \forall l$ . □

We have shown that the index assignments can be divided into  $M_\lambda$  disjunct classes and that the most linear element within each class is found by a simple sorting procedure. Table I compares the full search complexity,  $M_F$ , to the number of Hadamard classes,  $M_\lambda$ , for a few codebook sizes. In the next section, the search method with complexity  $M_\lambda$  is completed.

## VIII. CONSTRAINED FULL SEARCH INDEX ASSIGNMENT

With a constrained full search IA strategy, we refer to a search strategy sufficient to find the best possible IA with respect to some criterion, e.g. the linearity. The strategy of maximizing the linearity is straightforward in view of the preceding sections; visit each Hadamard class once (successively generate all possible  $\mathbf{G}_{ez}$  matrices) and use the sorting algorithm to find the maximum linearity *within* each Hadamard class. Consequently, an algorithm to maximize linearity over *all* index assignments for a given set of VQ points is formed.

The algorithm is called the *Full Linear Search Algorithm* (FLSA), because it performs a full search among the Hadamard classes, which gives the same result as if an exhaustive search were performed over all index assignments.

**Input:** A codebook  $\mathbf{Y}^0$ .

**Output:** A better index assignment  $\hat{\mathbf{G}}$  for  $\mathbf{Y}^0$ .

**The FLSA:**

$\mathbf{G} := \mathbf{G}^0$

$\hat{\lambda} := 0$

Repeat:

    Compute the Hadamard transform  $\mathbf{T}$  of  $\mathbf{Y} = [\mathbf{y}_{g_0}^0 \cdots \mathbf{y}_{g_{M-1}}^0]$

    Use the sorting algorithm to find the best IA  $\mathbf{G}'$  within the current Hadamard class

    Compute the linearity  $\lambda'$  of  $\mathbf{G}'$

    If  $\lambda' > \hat{\lambda}$ : Let  $\hat{\lambda} := \lambda'$  and  $\hat{\mathbf{G}} := \mathbf{G}'$

$index := 3$

    Repeat while  $g_{M-1} = index$ :

        Circularly shift  $[g_{index}, \cdots, g_{M-1}]$  one step to the right

$index := index + 1$

        If  $index$  is a power of 2:  $index := index + 1$

        If  $index := M - 1$ : **Terminate** the procedure and return  $\hat{\mathbf{G}}$ .

    Find  $p$  such that  $g_p = index^2$

    Swap  $g_p$  and  $g_{p+1}$

---

<sup>2</sup> This should be done by maintaining a table of  $p$  for every  $index$  value, not by search.

**Example:** An 8-point VQ has  $M_\lambda = 8!/(8 \cdot 7 \cdot 6 \cdot 4) = 30$  Hadamard classes. The following 30 index assignments,  $\mathbf{g}$ , are generated by the FLSA, reading columnwise from left to right.

```

01234567  01234657  01234675  01234576  01234756  01234765
01243567  01243657
01245367  01246357      :      :      :      :
01245637  01246537
01245673  01246573  01246753  01245763  01247563  01247653

```

To show that this algorithm indeed visits each Hadamard class, that is, that  $\mathbf{G}$  goes through all  $\mathbf{G}_{ez}$  matrices, we take another look at reduced echelon matrices.

**Theorem 5:** An index assignment matrix  $\mathbf{G}$  is a reduced echelon matrix if and only if no element of the index assignment  $\mathbf{g}$  that is a power of two is preceded by a larger value.

**Proof of Theorem 5:** In an IA matrix, there is no zero row, and all non-zero elements are equal to one. This makes properties 1 and 3 of a reduced echelon matrix (Section IIC) lose significance. Thus, an IA matrix  $\mathbf{G}$  is a reduced echelon matrix if and only if

- (1) Every column containing a leading entry has all its other entries zero,
- (2) Suppose that the leading entry of row  $i$  appears in column  $t_i$ , for  $i = 1, \dots, k$ . These numbers are ordered such that  $t_1 < t_2 < \dots < t_k$ .

The two conditions can be translated into tests on  $\mathbf{g}$ :

- (1) The first element with bit  $i$  set is  $2^i$ , for  $i = 0, \dots, k-1$ ,
- (2) The first element with bit  $i$  set precedes all elements with bit  $i+1$  set, for  $i = 1, \dots, k-1$ .

Combining these conditions yields Theorem 5. □

**Corollary:** The FLSA finds the IA with maximum linearity for any codebook.

Essentially, the algorithm creates an index assignment by insertion of one codeword at a time into a list. We consider the codewords in decreasing order. In the initially empty list, we first insert the codewords  $2^k - 1, 2^k - 2, \dots, 2^{k-1} + 1$ . They can be placed in any order without violating Theorem 5. Then comes  $2^{k-1}$ , which must be inserted first in the list. The insertion continues in the same fashion: the  $2^i - 1$  codewords  $2^{i+1} - 1, \dots, 2^i + 1$  can be inserted anywhere in the list and  $2^i$  must be before all the others. This is repeated for  $i = k-1, k-2, \dots, 0$ . Finally, the zero codeword is, according to Theorem 3, inserted first.

The FLSA generates the  $\mathbf{G}_{ez}$ 's in a faster way than by straightforward insertion. To move from one  $\mathbf{G}_{ez}$  to the next one, it is not necessary to empty the list and start over again. It is sufficient to redo the last insertion or insertions, which implies one or more swaps in  $\mathbf{g}$ . Most

often, the codeword 3 is swapped with another codeword, because 3 is the last codeword that can be inserted in different positions.

The FLSA is of more theoretical than practical interest, because of the rapid increase of Hadamard classes for increased number of bits in the codebook. Before the codewords  $2^{i+1} - 1, \dots, 2^i + 1$  are inserted, the list consists of  $M - 2^{i+1}$  elements, so these codewords can be inserted in  $(M - 2^{i+1} + 1) \cdot \dots \cdot (M - 2^i - 1)$  ways. The total number of Hadamard classes is thus

$$\begin{aligned} M_\lambda &= \prod_{i=1}^{k-1} (M - 2^{i+1} + 1) \cdot \dots \cdot (M - 2^i - 1) \\ &= \frac{(M-3)!}{\prod_{i=2}^{k-1} (M-2^i)}, \end{aligned}$$

which is the number used in (17) and in Table I. Although 4-bit codebooks can be assigned with the best indices found by exhaustive search, optimal index assignment of 5-bit codebooks is still infeasible. Therefore we proceed to suboptimal, but much faster, search strategies.

## IX. THE LINEARITY INCREASING SWAP ALGORITHM

The obvious modification of the FLSA for  $k > 4$  is to examine just a subset of the  $M_\lambda$  classes. A straightforward algorithm would be to systematically do pairwise swaps of the codewords and, for each swap, sort for the best index assignment within the current Hadamard class. Although correct in theory, this strategy is not efficient, as the linearity index varies very little within almost all Hadamard classes when the number of bits is more than, say, 7-8. Exceptions are within the Hadamard classes which contain the really good index assignments, because then there are  $k$   $\mathbf{t}$ -vectors with large norms and  $M - 1 - k$   $\mathbf{t}$ -vectors with small norms, which implies a large range of  $\lambda$  values. Unfortunately, as discussed earlier, the probability of finding such a Hadamard class is close to zero, making this strategy not more efficient than swapping codewords alone.

A more constructive approach is to focus first on *what* can be done fast, and second *where* applied effort is most effective. Below we address these foci, and present a suboptimal algorithm that rapidly finds a good index assignment.

In the first category, we identify that it is unnecessary to compute the complete Hadamard transform in order to calculate the linearity index. From (10), using the notation of (8), it is

easy to express what effect a swap of two VQ points has on the  $\mathbf{t}$ -vectors. Suppose we swap the VQ points  $\mathbf{y}_K$  and  $\mathbf{y}_L$ . The new  $\mathbf{t}$ -vectors,  $\mathbf{t}'$ , become

$$\begin{aligned}\mathbf{t}'_i &= \frac{1}{M} \left[ \mathbf{y}_0 + \dots + \mathbf{y}_L (-1)^{\mathbf{b}^T(K)\mathbf{b}(i)} + \dots + \mathbf{y}_K (-1)^{\mathbf{b}^T(L)\mathbf{b}(i)} + \dots \right] \\ &= \mathbf{t}_i - \frac{1}{M} \left[ (\mathbf{y}_L - \mathbf{y}_K) \cdot \left( (-1)^{\mathbf{b}^T(L)\mathbf{b}(i)} - (-1)^{\mathbf{b}^T(K)\mathbf{b}(i)} \right) \right].\end{aligned}\quad (25)$$

This is how an arbitrary  $\mathbf{t}$ -vector is affected by an arbitrary codeword swap. Now we focus on a subset of the  $\mathbf{t}$ -vectors and a special type of swaps.

As the denominator of (14) is invariant under all index assignments, the linearity only depends on the  $\mathbf{t}$ -vectors for which  $i = 2^l$ . For such indices, the norm of  $\mathbf{t}'_{2^l}$  is

$$\|\mathbf{t}'_{2^l}\|^2 = \|\mathbf{t}_{2^l}\|^2 + \frac{(\Delta b'_l)^2}{M^2} \|\Delta \mathbf{y}\|^2 - 2 \cdot \frac{\Delta b'_l}{M} \mathbf{t}_{2^l}^T \Delta \mathbf{y}, \quad (26)$$

where  $\Delta \mathbf{y} = \mathbf{y}_L - \mathbf{y}_K$ ,  $\Delta b'_l = b'_l(L) - b'_l(K)$ , and  $b'_l(\cdot)$  is defined in Section II B. The new linearity becomes

$$\lambda' = \lambda + \frac{1}{\sigma_{\text{VQ}}^2 M^2} \sum_{l=0}^{k-1} \left[ (\Delta b'_l)^2 \|\Delta \mathbf{y}\|^2 - 2 M \Delta b'_l \mathbf{t}_{2^l}^T \Delta \mathbf{y} \right]. \quad (27)$$

To find out whether a swap will increase the linearity or not, all we have to do is to evaluate the sum above. If it is positive, the swap is favorable. The existence of this fast test is one strong argument for using linearity as the performance measure.

We now concentrate on swaps such that  $\Delta b'_l = 0$  for all but one  $l$  value. This occurs for swaps of indices with a Hamming distance of one. Thus, if  $2n \cdot 2^l \leq K \leq (2n+1) \cdot 2^l - 1$ , for any integer  $n$ , and  $L = K + 2^l$ , then  $K$  and  $L$  differ only in bit  $l$ . Hence  $\Delta b'_l = -2$  and  $\Delta b'_j = 0$  for  $j \neq l$ . All possible swaps of this type are depicted as a butterfly structure in Fig. 4, well-known from FFT and similar algorithms. For  $L$  and  $K$  matching this pattern, (25) and (27) yield the following simple expressions.

$$\begin{aligned}\text{Test:} \quad & \frac{1}{M} \cdot \|\Delta \mathbf{y}\|^2 + \mathbf{t}_{2^l}^T \Delta \mathbf{y} > 0 \\ \text{Update } \mathbf{t}_{2^l}: \quad & \mathbf{t}'_{2^l} = \mathbf{t}_{2^l} + \frac{2}{M} \cdot \Delta \mathbf{y} \\ \text{Update } \lambda: \quad & \lambda' = \lambda + \frac{4}{\sigma_{\text{VQ}}^2 \cdot M^2} \cdot \left[ \|\Delta \mathbf{y}\|^2 + M \cdot \mathbf{t}_{2^l}^T \Delta \mathbf{y} \right]\end{aligned}\quad (28)$$

This swap strategy is just a special case of the general pairwise swap. It picks out only a subset (the  $k \cdot 2^{k-1}$  Hamming 1 neighbors) of all pairwise swaps, but the computational burden is so low that it is well worth treating separately. The  $\mathbf{t}$ -vectors that are important for the linear-

ity are successively and independently increased, which results in a high linearity almost instantly. We glimpse into the experimental section by mentioning that we typically rise the linearity of a 12 bit codebook with randomized indices from 0.0 to 0.9 in a tenth of a second on a modern computer. We name this swap strategy “Hamming 1 Butterflies” and provide a detailed description in the Appendix.

The remaining  $2^{k-1}(2^k - k - 1)$  pairwise swaps involve more than one  $\mathbf{t}$ -vector for each swap, and the somewhat more complicated test ( implicitly implied by (27) ) must be invoked. Still the additional increase of linearity may motivate these swaps. A procedure named “Remaining Butterflies” is also relegated to the Appendix.

Below we present a routine performing all pairwise swaps by first swapping the Hamming 1 neighbors and then all the others. This split is especially favorable when the total time consumption is of concern and the procedure has to be terminated before a full cycle is completed.

**Input:** An initial codebook  $\mathbf{Y}$ .

**Output:** A permutation of the initial codebook, also named  $\mathbf{Y}$ .

**Linearity Increasing Swap Algorithm (LISA):**

Compute the Hadamard transform  $\mathbf{T}$  of  $\mathbf{Y}$

Repeat:

**Hamming 1 Butterflies**  
**Remaining Butterflies**

Until convergence

Most index assignment procedures are indeed computationally oppressive, and the following list of computational aspects is worth taking into consideration when implementing the LISA.

- The Hadamard transform should be computed using an FFT-type structure.
- Only  $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_4, \dots, \mathbf{t}_{M/2}$  are updated by the algorithm, so if the  $\mathbf{T}$  matrix is needed afterwards, it has to be computed anew from  $\mathbf{Y}$ .
- Store and update  $M \cdot \mathbf{t}_i$  instead of  $\mathbf{t}_i$ .
- If desired, the progress of  $\lambda$  can be monitored using (27) and (28).
- Powers of two should not be implemented as such; use bit operations instead.
- Do not compute  $\Delta b'_j$  as a variable, but use *if* statements to take care of its three possible values.

## X. EXPERIMENTS

In this section we present some numerical results to get a perspective on the complexity of index assignment. For comparison, we solve the same problems with three different algorithms: the proposed LISA, and two well-known index assignment algorithms, namely a Simulated Annealing Algorithm (SAA) as described by Farvardin [6], and the Binary Switching Algorithm (BSA) by Zeger and Gersho [8]. All three algorithms are based on pairwise swaps to improve a given index assignment, but they employ different swap criteria and test order. We have tried to adhere as closely as possible to the descriptions of these algorithms, using the same program structures, parameter values, approximations, etc., as in the papers describing them, except in Fig. 7 below, where slightly modified versions were used.

We provide results for experiments on rate  $k/d = 1$  ( $k = 6, d = 6$ ), and rate 3 ( $k = 9, d = 3$ ) codebooks, designed using a sample iterative training algorithm [26]. The codebooks were optimized for sequences of samples from first-order Gauss-Markov sources, with a sample-to-sample correlation of  $\rho = 0.0$  (i.i.d. Gaussian variables) and  $\rho = 0.9$ , with the assumption of an error-free channel. As all algorithms employ a local descent concept, the results are dependent on the initial codebook. All results presented here (except Fig. 6) are therefore averaged over 20 runs of each algorithm, each time beginning from a new randomized index assignment<sup>3</sup>.

There was no need to reoptimize the index assignments for different bit error probability values. This constant does not enter any of the IA algorithms used, since the BSA and the SAA use the single-error approximation of  $D_C$ , and the LISA uses  $\lambda$ .

In Fig. 5, the signal-to-noise ratio (SNR) is given as a function of the channel bit error probability,  $q$ . It is clear that all three algorithms improve the SNR considerably for noisy channels, compared to a random IA. For the small codebook in Figs. 5 (a) and (b), the results are almost indistinguishable, their difference being less than 0.13 dB.

Index assignment for 9-bit codebooks is a harder problem. In Figs. 5 (c) and (d), the differences between the algorithms are apparent. Here, the SAA terminates far from the optimal IA, mainly because of the fixed cooling schedule [6]. A large codebook requires more pairwise swaps than a small one, which in terms of simulated annealing means that an extended cooling schedule should be applied. We will do that in the following.

---

<sup>3</sup>It is known that e.g. the LBG-split algorithm yield a more robust IA than a random assignment [6]. Structured VQ initializations and/or temporarily training for a noisy channel have also proved to give robust results [26], and consequently, such VQs could serve as good initial IAs. In this report however, the initial IA is randomized in order to remove the dependency of the VQ design procedure, which also makes it possible to measure statistics over a large number of runs of each IA algorithm. We have not observed a systematic decrease in the final robustness due to random initialization.

The LISA does not reach the good performance of the BSA for codebooks (c) and (d) in Fig. 5. The difference is about 0.6 dB, which shows the influence of the MEE assumption. When the rate is high, a Gaussian codebook will generally have an entropy significantly less than  $k$  (for the 9-bit codebooks, 8.80 and 8.83), which increases the discrepancy between linearity and channel distortion. A method to abridge the gap between the LISA and the BSA is suggested in the next section.

Obviously, it is important how fast an algorithm reaches good performance. Table II shows the computational complexity for each of the curves in Fig. 5. Three different measures related to the speed of the algorithms are presented. First, the number of pairs that are tested as candidates for a swap. Second, the number of these swaps that are actually retained. And third, the consumed CPU time<sup>4</sup>. While being the most important of the three features in applications, the CPU time is also the one most difficult to measure, in the sense that it is dependent on exterior circumstances such as hardware, software and programmer. Still, some general tendencies can clearly be seen, e.g., that the time needed by the SAA does not vary much with the codebook size. This is again due to the fixed cooling schedule employed. The BSA requires several times more CPU time than the LISA, but common for both, is that time increases rapidly with  $k$ .

Figure 6 shows the search procedures at work. The curves represent one sample run of each algorithm, using the same codebook and the same initial IA. The “noisy” behavior during the first phase of the SAA, when the algorithm sometimes allows the cost function to increase in order to avoid shallow local minima, is typical for simulated annealing. On a microscale, the two other curves are not monotonically increasing either, because none of them uses true channel distortion as the fidelity measure.

When a robust VQ is designed, the time allowed for the index assignment is normally not unlimited. A realistic problem may be to obtain as good results as possible in a pre-specified period of time. This is the setting for Fig. 7, where we have modified the algorithms to increase the SNR as much as possible in at most 1 minute each. The BSA was simply interrupted when the allowed period of time had elapsed whereas for the SAA, we adjusted the cooling schedule. The parameters ( $\alpha = 0.97$ ,  $T_f/\sigma_x^2 = 2 \cdot 10^{-5}$ ;  $T_0/\sigma_x^2 = 0.004$  for  $\rho = 0.0$  and  $T_0/\sigma_x^2 = 0.05$  for  $\rho = 0.9$ ) were empirically found to yield good performance under this time constraint. There was no need to interrupt the LISA in this experiment, since it had converged well within the time limit.

---

<sup>4</sup> The CPU time was measured on a DEC Alpha AXP 400 Workstation. Its clock speed is 133 MHz, and its SPECfp92 measure is 112.5.

In applications, it does not matter much whether the index assignment takes seconds or hours. However, since the time required to obtain a good result increases fast with the codebook size, a fast algorithm makes it possible to use larger codebooks. For instance, the LISA completes an index assignment for 12-bit codebooks in 10–20 minutes. The improvement is still greatest in the beginning, as in Fig. 6, so very little is lost if the algorithm is interrupted after, say, one minute.

## XI. DISCUSSION AND CONCLUSIONS

In the experiments of the previous section, we have intentionally avoided creating any combination of the three algorithms. Doing so may however prove fruitful, since the algorithms exhibit different advantages. The possibilities to create such hybrid algorithms are almost endless. For instance, the fast method to compute  $D_C$  that the BSA employs can be used in other algorithms using the same fidelity measure. The idea of simulated annealing can be incorporated into other swap-based algorithms, such as the LISA. Two or more algorithms with different fidelity measures can be used after each other, so that one improves the result from another. Especially, Figs. 5 (c)–(d) and 6 suggest that a superior algorithm can be created by succeeding the LISA with the BSA or the SAA.

We have introduced an objective measure called the linearity index, which was shown to be decisive for robustness in maximum entropy coding systems without explicit error protection. The Hadamard matrix proved effective in describing the codebook and the Hadamard transform facilitated the search for the index assignments yielding a high linearity index. One algorithm, the FLSA, finds the optimal index assignment and is of more theoretical than practical interest. Another algorithm, the LISA, proves to be significantly faster than the other algorithms tested, reaching a good, but not the best IA in the test. For large codebooks, say  $k > 10$ , the speed makes the LISA an attractive choice, if not the only practical alternative among the algorithms tested, for finding a good IA.

Experimental results verifies that good performance is reached also for encoding systems without full entropy.

## ACKNOWLEDGMENTS

The authors wish to thank professor Per Hedelin for the idea of using the Hadamard transform for description of codebooks. Fruitful discussions with Hedelin has also helped improving the LISA. We also appreciate the helpful suggestions given by the reviewers.

## APPENDIX

The following procedures are used in the LISA algorithm presented in Section IX.

### Hamming 1 Butterflies:

For  $l := 0$  to  $k - 1$ :

For  $i := 0$  to  $M - 2^{l+1}$ , step  $2^{l+1}$ :

For  $K := i$  to  $i + 2^l - 1$ :

$\Delta \mathbf{y} := \mathbf{y}_{K+2^l} - \mathbf{y}_K$   
 If  $\|\Delta \mathbf{y}\|^2 + M \cdot \mathbf{t}_{2^l}^T \Delta \mathbf{y} > 0$ :  
 Swap  $\mathbf{y}_K$  and  $\mathbf{y}_{K+2^l}$   
 $\mathbf{t}_{2^l} := \mathbf{t}_{2^l} + \frac{2}{M} \cdot \Delta \mathbf{y}$

### Remaining Butterflies:

For  $l := 1$  to  $k - 1$ :

For  $w := 2^l + 1$  to  $2^{l+1} - 1$ :

For  $i := 0$  to  $M - 2^{l+1}$ , step  $2^{l+1}$ :

For  $K := i$  to  $i + 2^l - 1$ :

$L := K \oplus w$

$\Delta \mathbf{y} := \mathbf{y}_K - \mathbf{y}_L$

$\Delta b'_j := b'_j(L) - b'_j(K); j = 0, \dots, l$

If  $\sum_{j=0}^l [(\Delta b'_j)^2 \|\Delta \mathbf{y}\|^2 - 2M \Delta b'_j \mathbf{t}_{2^j}^T \Delta \mathbf{y}] > 0$ :

Swap  $\mathbf{y}_K$  and  $\mathbf{y}_L$

$\mathbf{t}_{2^j} := \mathbf{t}_{2^j} - \frac{\Delta b'_j}{M} \cdot \Delta \mathbf{y}; j = 0, \dots, l$

## REFERENCES

- [1] R. E. Totty and G. C. Clark, "Reconstruction error in waveform transmission," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 336-338, Apr. 1967.
- [2] L. C. Potter, "Minimax nonredundant channel coding," *Submitted to IEEE Trans. Commun.*, May 1993.
- [3] N. Rydbeck and C.-E. Sundberg, "Analysis of digital errors in nonlinear PCM systems," *IEEE Trans. Commun.*, vol. COM-24, no. 1, pp. 59-65, Jan. 1976.
- [4] K. Zeger and A. Gersho, "Zero redundancy channel coding in vector quantization," *Electron. Lett.*, vol. 23, pp. 654-655, June 1987.
- [5] J. R. B. De Marca and N. S. Jayant, "An algorithm for assigning binary indices to the codevectors of a multi-dimensional quantizer," in *Proc. IEEE Int. Conf. Commun.*, pp. 1128-1132, Seattle, WA, June 1987.
- [6] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Trans. Inform. Theory*, vol. 36, no. 4, pp. 799-809, July 1990.
- [7] K. Zeger, E. Paksoy, and A. Gersho, "Source/channel coding for vector quantizers by index assignment permutations," in *Proc. IEEE Int. Symp. Information Theory*, pp. 78-79, San Diego, CA, Jan. 1990.
- [8] K. Zeger and A. Gersho, "Pseudo-Gray coding," *IEEE Trans. on Communications*, vol. 38, no. 12, pp. 2147-2158, Dec. 1990.
- [9] N.-T. Cheng and N. K. Kingsbury, "Robust zero-redundancy vector quantization for noisy channels," in *Proc. IEEE Int. Conf. Commun.*, pp. 1338-1342, Boston, MA, June 1989.
- [10] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84-95, Jan. 1980.
- [11] P. Knagenhjelm, "How good is your index assignment?," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. II, pp. 423-426, Minneapolis, MN, Apr. 1993.
- [12] R. Hagen and P. Hedelin, "Robust vector quantization in speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. II, pp. 13-16, Minneapolis, MN, Apr. 1993.
- [13] S.W. McLaughlin, J. Ashley, and D.L. Neuhoff, "The optimality of the natural binary code," *Coding and Quantization*, vol. 14, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1993.

- [14] S. W. McLaughlin, D. L. Neuhoff, and J. J. Ashley, "Optimal binary index assignment for a class of equiprobable scalar and vector quantizers," *Submitted to IEEE Trans. Inform. Theory*, 1993.
- [15] D.-M. Chiang and L. C. Potter, "Minimax non-redundant channel coding for vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. V, pp. 617-620, Minneapolis, MN, Apr. 1993.
- [16] H.-S. Wu and J. Barba, "Index allocation in vector quantization for noisy channels," *Electron. Lett.*, vol. 29, no. 15, pp. 1317-1319, July 1993.
- [17] P. Hedelin, P. Knagenhjelm, and M. Skoglund, "Vector quantization for speech transmission," in *Speech Coding and Synthesis*, W. B. Kleijn, K. K. Paliwal, eds., Elsevier, 1995.
- [18] P. Hedelin, P. Knagenhjelm, and M. Skoglund, "Theory for transmission of vector quantization data," in *Speech Coding and Synthesis*, W. B. Kleijn, K. K. Paliwal, eds., Elsevier, 1995.
- [19] K. Zeger and V. Manzella, "Asymptotic bounds on optimal noisy channel quantization via random coding," *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1926-1938, Nov. 1994.
- [20] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [21] O. Agazzi, D. G. Messerschmitt, and D. A. Hodges, "Nonlinear echo cancellation of data signals," *IEEE Trans. Commun.*, vol. COM-30, no. 11, pp. 2421-2433, Nov. 1982.
- [22] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, Holland: North-Holland Publishing Company, 1977.
- [23] W. R. Crowther and C. M. Rader, "Efficient coding of vocoder channel signals using linear transformation," *Proc. IEEE*, vol. 54, no. 11, pp. 1594-1595, Nov. 1966.
- [24] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, no. 1, pp. 58-68, Jan. 1969.
- [25] G. Birkhoff and S. Mac Lane, *A Survey of Modern Algebra*, revised ed. New York, NY: MacMillan, 1953.
- [26] P. Knagenhjelm, "A recursive design method for robust vector quantization," in *Proc. International Conference on Signal Processing, Applications and Technology (ICSPAT)*, vol. II, pp. 423-426, Boston, MA, Nov. 1992.

## LIST OF FIGURES

- Fig. 1** The codebook regarded as a transform of the hypercube, in this case 3-dimensional, spanned by the codewords. Codewords with a Hamming distance of 1 are connected with a line.
- Fig. 2** Two codebooks with identical codevectors. In (a) the linearity index  $\lambda = 0.12$  and in (b)  $\lambda = 0.91$ . Codewords with a Hamming distance of 1 are connected with a line.
- Fig. 3** Fig. (a) shows a scatter plot of the distortion versus  $\lambda$  for various index assignments, for a 4-bit codebook and  $q = 0.01$ . The lines visualize the bounds of Theorem 2. The channel is presumed to be BSC. The histogram (based on approximately 100000 samples) of the parameter is shown in (b). The mean value is 0.267 and the correlation between the distortion and  $\lambda$  is  $\rho_{D_C, \lambda} = -0.846$ .
- Fig. 4** VQ points involved in a butterfly swap for the updating of the vectors  $\mathbf{t}_1$ ,  $\mathbf{t}_2$ , and  $\mathbf{t}_4$  in the case of a 3-bit codebook.
- Fig. 5** SNR in dB as a function of bit error probability. Index assignments by three algorithms, compared to random IAs. (a)  $k = 6$ ,  $d = 6$ , and  $\rho = 0$ . (b)  $k = 6$ ,  $d = 6$ , and  $\rho = 0.9$ . Here the results of using the BSA and the SAA coincide. (c)  $k = 9$ ,  $d = 3$ , and  $\rho = 0.0$ . (d)  $k = 9$ ,  $d = 3$ , and  $\rho = 0.9$ .
- Fig. 6** Achieved SNR performance as a function of time for one trial of each algorithm. SNR was measured at  $q = 0.01$ . The codebook is  $k = 9$ ,  $d = 3$ , and  $\rho = 0.0$ .
- Fig. 7** SNR as a function of bit error probability. Each of the three algorithms was modified to terminate in 1 minute.  $k = 9$ ,  $d = 3$ . (a)  $\rho = 0.0$ . (b)  $\rho = 0.9$ .

## LIST OF TABLES

TABLE I: SEARCH COMPLEXITY FOR FULL SEARCH ( $M_F$ ) AND FULL LINEAR SEARCH ( $M_\lambda$ ).

TABLE II: THE COMPLEXITY OF THE THREE ALGORITHMS, MEASURED FOR FOUR CODEBOOKS.

**TABLE I**  
SEARCH COMPLEXITY FOR FULL SEARCH ( $M_F$ ) AND FULL LINEAR SEARCH ( $M_\lambda$ ).

$k$	$M!$	$k!$	$M_l$	$M_F$	$M_\lambda$
2	24	2	6	3	1
3	$4.0 \cdot 10^4$	6	168	840	30
4	$2.1 \cdot 10^{13}$	24	$2.0 \cdot 10^4$	$5.4 \cdot 10^{10}$	$6.5 \cdot 10^7$
5	$2.6 \cdot 10^{35}$	120	$1.0 \cdot 10^7$	$6.9 \cdot 10^{31}$	$8.2 \cdot 10^{26}$

**TABLE II**  
THE COMPLEXITY OF THE THREE ALGORITHMS, MEASURED FOR FOUR CODEBOOKS.

$k$	$d$	$\rho$	Number of tests			Number of swaps			CPU time		
			BSA	SAA	LISA	BSA	SAA	LISA	BSA	SAA	LISA
6	6	0.0	14000	30000	12000	62	2200	130	1 s	30 s	0 s
6	6	0.9	71000	31000	11000	120	2400	160	7 s	20 s	0 s
9	3	0.0	$2.9 \cdot 10^7$	10000	$1.3 \cdot 10^6$	1800	2900	2200	50 min	50 s	9 s
9	3	0.9	$7.4 \cdot 10^7$	7000	$1.3 \cdot 10^6$	1800	2900	2400	2 h	30 s	6 s

## FOOTNOTES

<sup>1</sup> The Schur product is defined as  $[\mathbf{A} \circ \mathbf{B}]_{i,j} = a_{i,j} \cdot b_{i,j}$

<sup>2</sup> This should be done by maintaining a table of  $p$  for every *index* value, not by search.

<sup>3</sup> It is known that e.g. the LBG-split algorithm yield a more robust IA than a random assignment [6]. Structured VQ initializations and/or temporarily training for a noisy channel have also proved to give robust results [26], and consequently, such VQs could serve as good initial IAs. In this report however, the initial IA is randomized in order to remove the dependency of the VQ design procedure, which also makes it possible to measure statistics over a large number of runs of each IA algorithm. We have not observed a systematic decrease in the final robustness due to random initialization.

<sup>4</sup> The CPU time was measured on a DEC Alpha AXP 400 Workstation. Its clock speed is 133 MHz, and its SPECfp92 measure is 112.5.