Correspondence

New Permutation Codes Using Hadamard Unscrambling

M. R. SCHROEDER, FELLOW, IEEE, AND N. J. A. SLOANE, FELLOW, IEEE

Abstract—**A new class of codes for data compression is described that combines permutations with the fast Hadamard transform (FHT). It was invented for digital speech compression based on linear predictive coding (LPC), but may be useful for other data compression applications. One particular code with rate $\frac{1}{2}$ is considered: a 16-bit code for a block length of 32 samples. All coding and decoding steps are fast, so that real-time applications with cheap hardware can be anticipated.**

I. Introduction

In linear predictive coding (LPC) of speech signals [1] both long-delay ("pitch") prediction and short-delay ("formant") prediction reduce a speech signal to a "prediction residual" of nearly independent, approximately Gaussian samples [10]. When a frequency-weighted error criterion based on human auditory perception [11] is used, the residual can be encoded by 1 bit/sample (8000 bits/s) while maintaining good speech quality [2]. By going to noninstantaneous (tree) coding, Atal and one of the authors (MRS) realized even lower bit rates, but at the cost of considerable computing complexity [10].

Finally, incorporating additional subjective criteria into the coding process, these authors compressed the bit rate for the prediction residual to 0.25 bit/sample (2000 bits/s) using an exhaustive code-book search of 1024 randomly generated Gaussian vectors of dimension 40 to match each residual frame of 40 samples [3]. While this coding gives near perfect speech quality, the simulation unfortunately runs much slower than real time even on a Cray-1 computer, and is therefore presently not usable in practice. But the computer runs did prove that bit rates of the order of 0.2 bits/sample for the prediction residual are sufficient.

This dilemma (between bit compression and computing complexity) led to a search for codes for quantizing at fractional bit rates that would permit fast coding and decoding while performing near the theoretical (rate-distortion) limit.

In 1969 the first author suggested "Hadamard scrambling" for picture coding (see the theoretical analysis by Berlekamp [5]). The idea then was to scramble, by a Hadamard transformation, the sparse nonzero amplitude array resulting from frame differential encoding of television pictures into an array of more nearly Gaussian amplitudes for easier coding. Around the same time the second author proposed using Hadamard transforms for encoding optical spectra, in order to reduce the noise in spectral measurements [13], [8].

Our proposal in the present correspondence is, in a sense, the converse of the scrambling idea. We start with a Gaussian frame (the prediction residual, possibly after a "critical frequency-band" transformation to better please the human ear) and transform it by a set of (modified) Hadamard transformations. Then we determine that transformation which best approximates a codeword with just two out of 32 (say) nonzero amplitudes. With two different nonzero amplitudes (± 1 say), there are $32 \cdot 31 = 992$ different permutations, and the best match can be selected very quickly. Ten bits suffice for the matching permutation, and if 64 different scrambling transformations are considered, the total bit rate is 16 bits/frame or 0.5 bits/sample for a frame size of 32.

The remainder of the correspondence elaborates this scrambling-cum-permutation idea and gives some early results.

II. Hadamard Permutation Codes

We describe here a 16-bit (rate $R = \frac{1}{2}$) code of block length 32 for data compression, or quantization, which (as shown in Section VI) performs near the rate-distortion limit for independent identically distributed (i.i.d.) Gaussian input. Generalizations to other rates and block lengths are obvious and we will not dwell on them here. In our proposal, the Hadamard transformation could be replaced by another type of "unscrambling matrix", such as a (discrete) Fourier transformation. Fast transformation algorithms would of course be preferred in most applications.

Consider a column data vector $\boldsymbol{x}$ of length (or dimension) 32 or its transpose

$$\boldsymbol{x}^T = (x_1, x_2, \cdots, x_{32}),$$

where the components x_i are i.i.d. random variables with zero mean and unit variance, and a set of 64 (or some other number in the range 8 to 512, say) "quasi-identity" matrices:

$$I^{(i)} = \begin{pmatrix} \pm 1 & & & \\ & \pm 1 & & 0 \\ & & \ddots & \\ & 0 & & \pm 1 \end{pmatrix}, \qquad i = 1, 2, \cdots, 64, \quad (1)$$

where the diagonal entries have 64 different sign combinations. For example, six of the diagonal elements could be independently $+1$ or -1. Many other strategies for choosing the signs are conceivable and possible, including random and pseudorandom sequences (such as shift-register sequences)—as long as they are compatible with the unscrambling matrix (see Section V).

Now "multiply" $\boldsymbol{x}$ by the 64 matrices $I^{(i)}$ to form 64 vectors

$$\boldsymbol{y}^{(i)} = I^{(i)}\boldsymbol{x}, \qquad i = 1, 2, \cdots, 64. \quad (2)$$

(The "multiplication" amounts to nothing more than changing the signs of some of the components of $\boldsymbol{x}$.)

Next consider the 32×32 Hadamard matrix of Sylvester type [8]

$$H_{32} = H_2^{[5]},$$

where H_2 is the 2×2 Hadamard matrix

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and the superscript in brackets indicates Kronecker or tensor product. Thus

$$H_4 = H_2^{[2]} = H_2 \otimes H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Manuscript received January 7, 1985; revised December 16, 1985.

M. R. Schroeder is with Bell Laboratories, Murray Hill, NJ 07974. He is also with the University of Goettingen, D-3400 Goettingen, Federal Republic of Germany.

N. J. A. Sloane is with Bell Laboratories, Murray Hill, NJ 07974.

IEEE Log Number 8610196.

Similarly

$$H_8 = H_2^{[3]} = H_4 \otimes H_2,$$

and in general

$$H_{2^m} = H_2^{[m]}. \tag{3}$$

As a consequence of the factorization (3), fast algorithms for multiplication by H_{2^m} exist [8]. Specifically, the 2^m multiplications and $2^m - 1$ additions required to calculate a single vector component in a 2^m-dimensional matrix multiplication are replaced by a mere $m - 1$ additions. Note that Hadamard matrices are orthogonal [8]:

$$H_n H_n^T = nI_n, \tag{4}$$

where I_n is the $n \times n$ unit matrix. The Sylvester type matrices are also symmetric: $H_n^T = H_n$. Thus Hadamard "unscrambling" of the $y^{(i)}$, giving

$$z^{(i)} = H_{32}\, y^{(i)} = H_{32} I^{(i)} x, \tag{5}$$

is a fast operation, and so is its inverse.

The norm of x, $N(x) = x^T x = \Sigma x_i^2$, has mean 32, so the mean value of $N(z^{(i)})$ is 1024. The purpose of the unscrambling is to see which of the 64 vectors $z^{(i)}$ $(i = 1, 2, \cdots, 64)$ best matches one of the $32 \cdot 31 = 992$ codewords that result from the permutations of the *basic codeword*

$${}^1c^T = (16\sqrt{2}, -16\sqrt{2}, 0^{30}), \tag{6}$$

where 0^{30} stands for 30 zeros. The nonzero entries have magnitude $16\sqrt{2}$, so that the norm of ${}^1c$ equals 1024 and thus matches the norm of the vectors $z^{(i)}$.

We shall designate the 992 permutations of ${}^1c$ by

$${}^k c, \qquad k = 1, 2, \cdots, 992. \tag{7}$$

This set of vectors is an example of a "variant I permutation code" as introduced by Slepian [12], [4].

Besides having a rate that is ideal for our purpose, this code is also a very natural one to use when considered geometrically. More generally, let $\mathscr{C}_n$ consist of the $n(n-1)$ vectors of the form $(a, -a, 0^{n-2})$, where a is a constant and all permutations of the coordinates occur. Then $\mathscr{C}_3$ represents the six vertices of a regular hexagon, $\mathscr{C}_4$ the 12 vertices of a regular cuboctahedron [7], and in general $\mathscr{C}_n$ is the set of minimal vectors of the $(n-1)$-dimensional lattice A_{n-1} [6]. The points of $\mathscr{C}_n$ lie in an $(n-1)$-dimensional subspace (since the coordinates add to zero), but in that subspace they are symmetrically placed around the origin.

III. The Coding Process

In coding a Gaussian vector x, we are looking for that pair of vectors $z^{(i)}$ and ${}^k c$ that minimizes the mean squared difference between them over all $64 \cdot 992$ possibilities. Equivalently, we are looking for a pair of indices (i, k) that maximizes the inner product

$${}^k c^T z^{(i)}, \qquad i = 1, \cdots, 64;\ k = 1, \cdots, 992. \tag{8}$$

In view of (6), the maximal inner product can be found by finding that index i for which $z^{(i)}$ has the largest *range*, i.e., the greatest difference between its largest and smallest components. The positions of the largest and smallest components of $z^{(i)}$ then determine k. (Thus i and k can be found by scanning each vector $z^{(i)}$ once.) There are 64 possible values for the index i, requiring 6 bits, and $32 \cdot 31 = 992$ possible values for k, requiring $\log_2 32 + \log_2 31 = 9.95 \cdots < 10$ bits. Thus the data vector x can be encoded by $6 + 10 = 16$ bits, which corresponds to 0.5 bits/sample (a 2 : 1 data compression if the data were binary to begin with).

IV. The Decoding Process

The decoding process at the receiving end is nearly trivial: ten of the 16 received bits specify the position of the two nonzero samples in the transformed output vector. In addition, some of the algebraic signs of the components of the vector have to be changed. The components to be changed are determined by the 6 bits specifying the index i (and by the form chosen for the $I^{(i)}$). Because of the symmetry of H_n and (4), the inverse Hadamard matrix is given by $H_n^{-1} = (1/n) H_n$. In other words, apart from the constant factor n, the required operation is identical to the original transformation, and is therefore also fast. This transformation yields the output vector

$$\tilde{x} = \frac{1}{32} I^{(i)} H_{32}{}^k c. \tag{9}$$

(Note that the $I^{(i)}$ are their own inverses.)

V. More on the Quasi-Identity Transformation

It remains to choose the sign patterns of the 64 quasi-identity matrices $I^{(i)}$ to be used in (5). Certain choices are clearly undesirable. For example, it would be pointless to use an $I^{(i)}$ and $I^{(j)}$ whose diagonal entries are the negatives of each other. Sign patterns with the same symmetry as the Sylvester-type Hadamard matrix H_{32} are also undesirable. A reasonable solution, therefore, is to choose the signs according to some pseudo-random process that is unconnected to H_{32}. The solution adopted was to choose the 64 sign patterns to be the first 32 columns of a 64×64 Hadamard matrix of *shift-register type* (see for example [8, §A.2.3]).

VI. Simulation Results for Gaussian Data

For good performance of the code the mean squared difference between the unquantized input x and the quantized output $\tilde{x}$ should be small. Of course, it cannot be smaller than the bound given by rate distortion theory [9]:

$$D = \sigma^2 2^{-2R} \tag{10}$$

or, for $R = 1/2$, $D = \sigma^2/2$. Thus the signal-to-noise ratio (SNR) is at most $10 \log_{10} (\sigma^2/D) = 3.01$ dB. (If $R = 1/2$ was realized by replacing every other data sample by zero and preserving only the sign of the remaining samples, the SNR would be only 1.66 dB.)

Applying the above [32, 16] code to a fixed-energy Gaussian data frame, we obtained, by a series of computer simulations, an average SNR of 2.57 dB.

Acknowledgment

We thank H. Alrutz for assisting us with the computer simulations.

References

[1] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell Syst. Tech. J.*, vol. 49, pp. 1973–1986, 1970.

[2] ——, "Improved quantizer for adaptive predictive coding of speech signals of low bit rates," in *Proc. 1980 Int. Conf. Acoust., Speech, Signal Processing*, 1980, pp. 535–538.

[3] ——, "Stochastic coding of speech signals at very low bit rates," in *Proc. Int. Conf. Communications—ICC '84*, 1984, pp. 1610–1613.

[4] T. Berger, F. Jelinek, and J. K. Wolf, "Permutation codes for sources," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 160–169, 1972.

[5] E. R. Berlekamp, "Some mathematic properties of a scheme for reducing the bandwidths of motion pictures by Hadamard smearing," *Bell Syst. Tech. J.*, vol. 49, pp. 969–986, 1970.

[6] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 211–226, 1982.

[7] H. M. Cundy and A. P. Rollett, *Mathematical Models*. Oxford, England: Oxford Univ. Press, 1957, §3.7.2.

[8] M. Harwit and N. J. A. Sloane, *Hadamard Transform Optics*. New York: Academic, 1979.

[9] R. J. McEliece, *The Theory of Information and Coding*. Reading, MA: Addison-Wesley, 1977.
[10] M. R. Schroeder and B. S. Atal, "Rate distortion theory and predictive coding," in *Proc. 1981 Conf. Acoust., Speech, Signal Processing*, pp. 201–204.
[11] M. R. Schroeder, B. S. Atal, and J. L. Hall, "Optimizing digital speech codes by exploiting the masking properties of the human ear," *J. Acoust. Soc. Amer.*, vol. 65, pp. 1647–1652, 1979.
[12] D. Slepian, "Permutation modulation," *Proc. IEEE*, vol. 53, pp. 228–236, 1965.
[13] N. J. A. Sloane, T. Fine, P. G. Phillips, and M. Harwit, "Codes for multislit spectrometry," *Appl. Opt.*, vol. 8, pp. 2103–2106, 1969.

A Simple Derivation of the Berlekamp–Massey Algorithm and Some Applications

KYOKI IMAMURA AND WATARU YOSHIDA

Abstract—**Another viewpoint is presented on the derivation of the Berlekamp–Massey algorithm. Our approach differs from previous ones in the following manner. The properties of the shortest linear feedback shift register that generates a given sequence are first derived without reference to the Berlekamp–Massey algorithm. The Berlekamp–Massey algorithm is then derived using these properties. Our approach has the advantage of being easier to understand.**

I. Introduction

The algorithm discovered by Berlekamp [1] for decoding BCH codes is very elegant. Massey [2] showed that Berlekamp's algorithm is best interpreted as an efficient recursive method for finding the shortest linear feedback shift register (LFSR) that generates a given sequence. Since Massey's interpretation is very useful, the algorithm is often called the Berlekamp–Massey algorithm.

The derivation of the Berlekamp–Massey algorithm, however, seems to be rather difficult, and why it works is not so easy to understand [3], [4]. Many of recent coding theory books [3]–[5] prefer another algorithm based on the Euclidean algorithm discovered by Sugiyama *et al.* [6] to the Berlekamp–Massey algorithm because the former is easier to understand.

The main purpose of this correspondence is to present a new method for deriving the Berlekamp–Massey algorithm more heuristically to make this important algorithm more easily understandable. Our method is new in the sense that the problem of finding the shortest LFSR is separated in the following two steps, and step 1 is first solved without using any result of step 2, and next the results of step 1 are used successfully in solving step 2. 1) Find a general rule how the length of the LFSR grows with the sequence length and find a necessary and sufficient condition on the LFSR to be unique. 2) Find a recursive algorithm for updating the LFSR.

II. Dynamic Behavior of Recursive Algorithms for Finding the Shortest LFSR

The Berlekamp–Massey algorithm [1], [2] may be best interpreted as an efficient recursive method for finding the shortest linear feedback shift register (LFSR) that generates a given sequence.

Manuscript received September 10, 1985; revised March 17, 1986. This paper was presented at the IEEE International Symposium on Information Theory, Brighton, England, June 24–28, 1985.

The authors are with the Department of Electrical Engineering, Saga University, Saga 840, Japan.

IEEE Log Number 8610199.

Let

$$a_1, \cdots, a_N \tag{1}$$

be a given sequence of length N. Find an LFSR of length L as shown in Fig. 1 such that the first N outputs of the LFSR starting from the initial state $(a_L, \cdots, a_1)$ are exactly (1) and L is minimal.

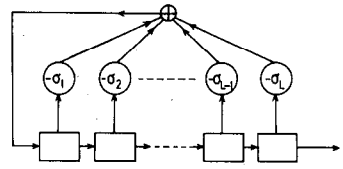

Fig. 1. LFSR of length L that generates sequence (1).

Since the LFSR in Fig. 1 is completely determined by its length L and its connection polynomial

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_L x^L, \tag{2}$$

it will be denoted as the LFSR $(L, \sigma(x))$.

In this correspondence we will consider algorithms which are recursive with respect to the sequence length. Let

$$a_1, \cdots, a_n, \qquad n \le N \tag{3}$$

be the first n terms of (1) and the LFSR $(L(n), \sigma^{(n)}(x))$ be the shortest LFSR that generates (3). We want to find an updating algorithm for obtaining the LFSR $(L(n+1), \sigma^{(n+1)}(x))$ from the knowledge of LFSR's $(L(i), \sigma^{(i)}(x))$ for $i \le n$.

In this section we will find a general rule which every recursive algorithm must follow. We will find a general rule how $L(n)$ grows with n and find a necessary and sufficient condition on the LFSR $(L(n), \sigma^{(n)}(x))$ to be unique. The rule must be found without using any property special to the particular algorithm.

Let us write $\sigma^{(n)}(x)$ as

$$\sigma^{(n)}(x) = 1 + \sigma_1^{(n)} x + \cdots + \sigma_l^{(n)} x^l, \qquad l = L(n). \tag{4}$$

Then unknown parameters $l = L(n)$, $\sigma_1^{(n)}, \cdots, \sigma_l^{(n)}$ must be determined such that the equation

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_{l+1} \\ a_2 & a_3 & \cdots & a_{l+2} \\ \vdots & \vdots & & \vdots \\ a_{n-l} & a_{n-l+1} & \cdots & a_n \end{pmatrix} \cdot \begin{pmatrix} \sigma_l^{(n)} \\ \vdots \\ \sigma_1^{(n)} \\ 1 \end{pmatrix} = 0 \tag{5}$$

holds with as small $l = L(n)$ as possible. The difficulty of the problem lies in the fact that l is not known beforehand and must be chosen as small as possible. The coefficient matrix of (5) is an $(n-l) \times (l+1)$ Hankel matrix and will be denoted as $A(n-l, l+1)$. First we will prove the following lemma.

Lemma 1: $L(n) = n/2$ if and only if $A(n/2, n/2)$ is nonsingular.

Proof: Let $n = 2l$. First we prove that $A(l, l)$ is nonsingular if $L(2l) = l$. If the $(m+1)$th row of $A(l, l)$ is a linear combination of the first m $(m < l)$ rows of $A(l, l)$, then, using (5) with $n = 2l$, we can show that $L(2l) \le m < l$, which is a contradiction with the assumption $L(2l) = l$.

Next we prove that $L(2l) = l$ if $A(l, l)$ is nonsingular. It is obvious that $L(2l) \le l$ because (5) can be uniquely solved for $\sigma_1^{(n)}, \cdots, \sigma_l^{(n)}$ if $n = 2l$ and $A(l, l)$ is nonsingular. We have $L(2l) = l$ because $L(2l) = m < l$ means that the $(m+1)$th row of $A(l, l)$ is a linear combination of the first m rows of $A(l, l)$.