$$Y_H\left(k_1+\frac{N}{2},k_2+\frac{N}{2}\right)=S_{00}(k_1,k_2)-\left(S_{01}(k_1,k_2)\cos\frac{2\pi k_2}{N}+S_{01}\left(\frac{N}{2}-k_1,\frac{N}{2}-k_2\right)\sin\frac{2\pi k_2}{N}\right)$$
$$-\left(S_{10}(k_1,k_2)\cos\frac{2\pi k_1}{N}+S_{10}\left(\frac{N}{2}-k_1,\frac{N}{2}-k_2\right)\sin\frac{2\pi k_1}{N}\right)$$
$$+\left(S_{11}(k_1,k_2)\cos\frac{2\pi}{N}(k_1+k_2)+S_{11}\left(\frac{N}{2}-k_1,\frac{N}{2}-k_2\right)\sin\frac{2\pi}{N}(k_1+k_2)\right). \quad (11)$$

The computation implied by (8)–(11) is given in Fig. 1 which is called a 2-D DHART butterfly. Each butterfly involves 6 real multiplications and 8 real additons. The basic 2×2 DHART needs no multiplications. The computation of 2-D DHART for an $N\times N$ real sequence is given in Fig. 2. After calculating the 2-D DHART, the DFT is computed by simple additions as in (7).

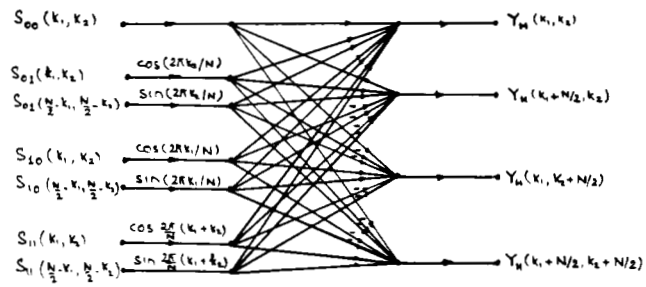

Fig. 1. Radix (2×2) butterfly.

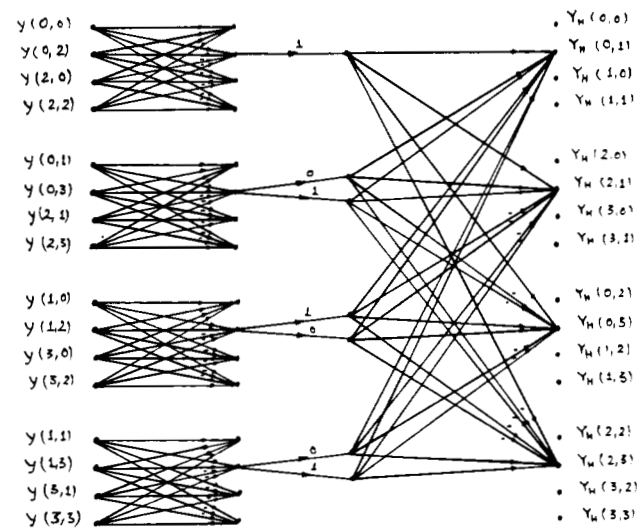

Fig. 2. Complete (4×4) point radix (2×2) DHART. Only one butterfly shown.

III. Complexity

We will assume that our input data are real valued. For the row–column decomposition approach as in (4) the number of real multiplications required is $N^2\log_2 N$, using the fast algorithm in [5]. The number of real additions is also $N^2\log_2 N$. To obtain the real and imaginary parts of the DFT an additional $N^2/2$ additions are needed.

For the vector-radix algorithm the DHART butterfly needs 6 real multiplication and 8 real additons. Since the whole DHART computation needs $N^2/4$ stages, $(3/2)N^2\log_2 N$ real multiplications (which is half the complexity of the vector-radix DFT algorithm) and $2N^2\log_2 N$ real additions are needed. Again $N^2/2$ extra additions are needed to obtain the DFT.

The real and imaginary part of the 2-D DFT need not always be computed from the 2-D DHART. The power spectrum and convolution of two multidimensional sequences can be computed directly from the DHART defined in (6).

Let us assume that we wish to convolve (circular convolution) two 2-D sequences $x(n_1,n_2)$ and $y(n_1,n_2)$ to obtain $z(n_1,n_2)$. Then we can compute the 2-D DHART $X_H(k_1,k_2)$ and $Y_H(k_1,k_2)$ as defined in (6) using our vector-radix algorithm. It is easy to show that $Z_H(k_1,k_2)=1/2((Y_H(k_1,k_2)+Y_H(N-k_1,N-k_2))\cdot X_H(k_1,k_2)+Y_H(k_1,k_2)-Y_H(N-k_1,N-k_2))\cdot X_H(N-k_1,N-k_2))$. Thus we can obtain $Z_H(k_1,k_2)$ using the above formula and inverse transform it to get $z(n_1,n_2)$. In the special case where one of the sequences $y(n_1,n_2)$ obeys a zero-phase symmetry [1], that is, $y(n_1,n_2)=y(N-n_1,N-n_2)$, the odd part, $Y_H(k_1,k_2)-Y_H(N-k_1,N-k_2)$ is zero, and the computation of $Z_H(k_1,k_2)$ is further simplified. These ideas are analogous to those in [4], but our alternative DHART and the corresponding vector-radix algorithm make them more attractive.

References

[1] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[2] G. K. Rivard, "Direct fast Fourier transform of bivariate functions," *IEEE Trns. Acoust., Speech, Signal Processing*, vol. ASSP-25, no. 3, pp. 250–252, June 1977.

[3] D. B. Harris *et al.*, "Vector-radix fast Fourier transform," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing* (May 1977), pp. 548–551.

[4] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Amer.*, vol. 73, no. 12, pp. 1832–1835, Dec. 1983.

[5] ____, "Fast Hartley transform," *Proc. IEEE*, vol. 72, no. 8, pp. 1010–1018, Aug. 1984.

[6] R. Kumaresan and P. K. Gupta, "A prime factor algorithm using discrete Hartley transform," under preparation.

[7] T. W. Parson, "A Winogrd-Fourier transform algorithm for real-valued data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 4, Aug. 1979.

An Eight-Dimensional Trellis Code

A. R. CALDERBANK AND N. J. A. SLOANE

An 8-state trellis code is described that uses a signal constellation from the 8-dimensional Gosset lattice E_8. It can be used for example to transmit data at 9.6, 14.4, and 19.2 kbits/s with a nominal coding gain of close to 6 dB

I. Introduction

A *trellis code* [1], [2] is a technique for encoding a data stream into a sequence of real vectors that are transmitted over a noisy channel. The set of all possible vectors forms the *signal constellation*. This letter describes a trellis code for which the signal constellation consists of vectors in the eight-dimensional Gosset lattice E_8. These vectors are divided equally among the sixteen cosets of a certain sublattice $M(E_8)$. The input data stream is partioned into blocks of $k+3$ bits. Three bits in each block drive a generalized convolutional code whose output symbols are names of cosets of the sublattice, and the remaining k bits select vectors from these

Manuscript received October 10, 1985; revised December 13, 1985.

The authors are with the Mathematical Sciences Research Center, AT & T Bell Laboratories, Murray Hill, NJ 07974, USA.

0018-9219/86/0500-0757$01.00 ©1986 IEEE

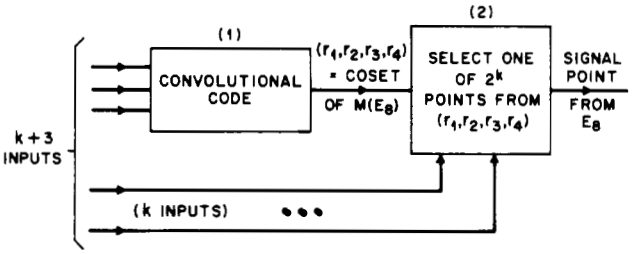

Fig. 1. Overall trellis code.

cosets, as shown in Fig. 1. (This technique for constructing trellis codes using cosets of a lattice was introduced in [3].) By taking k to be 13, 21, or 29 we obtain codes with rate 2, 3, or 4 bits/dimension, which may be used to transmit data at 9.6, 14.4, or 19.2 kbits/s, respectively, for a modulation rate of 2400 complex signals per second. The nominal coding gain is close to 6 dB.

Other trellis codes have been constructed in [1]–[7]. However, none of these codes appear to achieve a 6-dB coding gain at the same rates as our code while using only an 8-state encoder.

II. The Gosset Lattice E_8

The lattice E_8 consists of the vectors $\boldsymbol{x} = (x_1, \cdots, x_8)$, where the x_i are all integers, or all halves of odd integers, and $x_1 + \cdots + x_8$ is even [8]. The map $\boldsymbol{x} \rightarrow \boldsymbol{x}M$, where

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

doubles the *norm* (or energy) $\boldsymbol{x} \cdot \boldsymbol{x}$, and we define the sublattice $M(E_8)$ to consist of the vectors $\boldsymbol{x}M$ for $\boldsymbol{x} \in E_8$. Then E_8 is partitioned into 16 cosets of $M(E_8)$. The quotient group $E_8/M(E_8)$ is elementary abelian, and is described by the set of binary 4-tuples. Every vector in E_8 has 240 nearest neighbors, at squared distance 2 from it, which are divided equally among the 15 nonzero cosets (with 16 neighbors in each coset). The minimal norm of any nonzero coset of $M(E_8)$ is[1] 2.

In order to minimize the average energy of the constellation we work, not with the lattice E_8 itself, but with the *shifted* E_8, with the origin moved to $(1,0,0,0,0,0,0,0,)$. The shifted E_8 consists of the vectors $(x_1, \cdots, x_8)$ where the x_i are all integers or all halves of odd integers, and $x_1 + \cdots + x_8$ is odd [9]. The vectors in the shifted E_8 are divided into shells, the nth shell consisting of the vectors of norm n. For example, the first shell contains 16 points $(\pm 1, 0^7)$ of norm 1 and the second shell contains 128 points $(\pm \frac{1}{2}^8)$ of norm 2 (with an odd number of minus signs). The nth shell contains $16(\sigma_3(n) - \sigma_3(n/2))$ points, where $\sigma_3(n)$ is the sum of the cubes of the divisors of n if n is an integer, and is 0 otherwise [9].

The signal constellation consists of 2^{k+4} vectors from the shifted E_8, with 2^k vectors in each coset of $M(E_8)$. It is formed by taking a union of successive shells, omitting enough vectors from the outermost shell to obtain exactly 2^{k+4} vectors. For $k = 13$ the $2^{17} = 131072$ vectors comprise shells 1 through 12 plus 31936 vectors from shell 13, and the average norm P is $10.722\cdots$. For $k = 21$ we use shells 1 through 53 plus 346016 vectors form shell 54, and $P = 42.899\cdots$. For $k = 29$ we use shells 1 through 213 plus 155410896 vectors from shell 214, and $P = 171.589\cdots$.

III. The Convolutional Code

The convolutional code is described by the generator matrix

$$G = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \qquad (1)$$

where the columns specify cosets of $M(E_8)$. The output symbols are sums of columns of G and are also cosets of $M(E_8)$. Equation (1) means that the current output (r_1, r_2, r_3, r_4) depends on the current input (u_1, u_2, u_3) and on the previous input (v_1, v_2, v_3), and is given by

$$x_1 = v_1 + v_3 + u_2 \qquad x_2 = v_1 + u_1 + u_3$$
$$x_3 = v_1 + v_2 + u_1 + u_2 \qquad x_4 = v_2 + v_3 + u_3.$$

This code is noncatastrophic, has memory 3, and there are eight states.

IV. The Trellis Code

The overall trellis code is shown in Fig. 1. The minimal squared Euclidean distance between output sequences corresponding to distinct inputs streams is $d = 4$. We use d/P as a figure of merit for the code. We compare this code with uncoded transmission at the same rate, in the same dimension, and with a constellation in the shape of a cube. It follows from [3, eq. (58)] that the figure of merit for uncoded transmission is $(d/P)_u = 3/(2^{(k+7)/4} - 2)$. The nominal coding gain is

$$10 \log_{10} \left(\frac{d}{P} \Big/ \left(\frac{d}{P} \right)_u \right) \text{dB}.$$

For $k = 13, 21, 29$ this is 5.718, 5.929, and 5.980 dB, respectively.

The number of paths through the trellis at minimal distance from a given path is 3056, or 382 per dimension. This can be reduced by increasing the memory of the code. Let G' be the matrix obtained form G by changing column 2 from $(0011)^T$ to $(0110)^T$. To obtain a 16-state code with path multiplicity 190 per dimension simply add $(1001)^T$ to G' (on the left). Let G'' be the matrix obtained from G by changing column 2 from $(0011)^T$ to $(1010)^T$. Adding $(1001)^T$, $(0110)^T$ to G'' gives a 32-state code with path multiplicity 62 per dimension whilst adding $(1001)^T, (0110)^T, (0011)^T$ to G'' gives a 64-state code with path multiplicity 30 (cf. [7, Table 11]).

V. Implementation Considerations

Decoding may be accomplished by combining one of the decoding methods for E_8 [10], [11] with the Viterbi algorithm [1], [2]. In encoding, the difficult step is in Box (2) of Fig. 1, when it is necessary to associate lattice vectors with the 2^k inputs (and conversely, after the decoding step). One solution to this problem uses the scheme proposed in [12], which involves a signal constellation slightly different from the one described above. Consider the case $k = 29$. The constellation described above uses 2^{33} points from shells 1 through 214, a roughly spherical configuration. Instead, let Q denote the Voronoi polytope of the E_8 lattice [10] (a certain highly symmetric eight-dimensional polytope bounded by 240 seven-dimensional faces). In each coset of $M(E_8)$ we take 2^{29} vectors arranged in a constellation of shape Q. The whole constellation is then the union of 16 overlapping copies of Q. Since this is no longer spherical the signal energy is increased slightly.[2] On the other hand, if this constellation is used, encoding can be carried out by an E_8 decoder, i.e., is very fast.

Since every coset of $M(E_8)$ is invariant under rotation through 90° it is easy to modify the encoding scheme to provide transparency to 90° phase shifts.

VI. Conclusion

A new family of eight-state, eight-dimensional trellis codes based on the E_8 lattice has been described, which can be used at 9.6, 14.4, or 19.2 kbits/s. The nominal coding gain is close to 6 dB, which compares favorably with codes given in [1]–[7]. The relatively high path multiplicity of 382 per dimension can be reduced by increasing the memory of the code.

[1]Thus the metric structure of $E_8/M(E_8)$ is the same as that of the Galois field $GF(2^4)$ with respect to the Hamming metric. This suggests that other efficient codes may be found by replacing the convolutional code in Fig. 1 with a Reed–Solomon code over $GF(2^4)$.

[2]An upper bound on the signal energy in this constellation may be obtained as follows. If the constellation consisted of a single copy of Q, then the average energy is increased by approximately

$$\int_Q x \cdot x\, dx \Big/ \int_S x \cdot x\, dx$$

where S is an eight-dimensional sphere of the same volume as Q. From [10] this ratio is $1.017\cdots$. For $k = 29$ this increases the average energy to $174.581\cdots$, and reduces the gain to 5.905 dB. Since the actual constellation is the "average" of 16 copies of Q, the true increase in energy will be less than this.

References

[1] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, 1982.

[2] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Selected Areas Commun.*, vol. SAC-2, pp. 632–647, 1984.

[3] A. R. Calderbank and N. J. A. Sloane, "New trellis codes," *IEEE Trans. Inform. Theory*, in press.

[4] A. R. Calderbank and J. E. Mazo, "A new description of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 784–791, 1984.

[5] S. G. Wilson, H. A. Sleeper, and N. K. Srinath, "Four-dimensional modulation and coding: An alternative to frequency-reuse," in *Science, Systems and Services for Communications*, P. Dewilde and C. A. May, Eds. New York: IEEE/Elsevier-North Holland, 1984, pp. 919–923.

[6] A. R. Calderbank and N. J. A. Sloane, "Four-dimensional modulation with an eight-state trellis code," *AT&T Tech. J.*, vol. 64, pp. 1005–1018, 1985.

[7] L.-F. Wei, "Trellis coded modulation with multi-dimensional constellations," *IEEE Trans. Inform. Theory*, to appear.

[8] J. Leech and N. J. A. Sloane, "Sphere packing and error-correcting codes," *Canad. J. Math.*, vol. 23, pp. 718–745, 1971.

[9] J. H. Conway and N. J. A. Sloane, *Sphere Packing and Applications*. New York: Springer-Verlag, 1987, to appear.

[10] ——, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 211–226, 1982.

[11] ——, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 41–50, Jan. 1986.

[12] ——, "A fast encoding method for lattice codes and quantizers," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 820–824, 1983.

Identification of Stable Nonstationary Lattice Predictors by Linear Programming

G. MARTINELLI, G. ORLANDI, L. PRINA RICOTTI, AND S. RAGAZZINI

Time-varying models can be identified by lattice predictors with time-dependent reflection coefficients. The main defect of these predictors is that the magnitude of their coefficients cannot be controlled. In the present work a method is presented for removing such inconvenience based on linear programming.

I. Introduction

The lattice predictor with time-dependent reflection coefficients represents a very effective tool for modeling nonstationary situations [1]. However, the methods for determining the coefficients do not guarantee the stability of the resulting schemes. Since stability is a physical requirement, this inconvenience invalidates the resulting models. A method for eliminating this limitation is suggested in [2].

The stability of an actual model is not the only requirement related to the physical background. In all the applications, in fact, the reflection coefficients are directly related to physical quantities. On the basis of the expression of the former in function of the latter, it is easy to deduce that the magnitudes of the reflection coefficients should be less than a value ρ dependent on the application. Consequently, the only requirement of stability, i.e., $\rho < 1$, is usually not sufficient to match the predictor to the physical nature of the model.

In the present letter, we propose a simple method for constraining the lattice predictor to have coefficients less than a suitable ρ in magnitude. The method is based on a linear programming extension of the usual LMS (least mean square) criterion, in order to incorporate the said constraints. The efficiency of the resulting algorithm will be illustrated by an example.

Manuscript received September 30, 1985.

G. Martinelli is with the INFOCOM Department, University of Rome, Rome, Italy.

G. Orlandi is with the Department of Electronics and Automatics, University of Ancona, Ancona, Italy.

L. Prina Ricotti and S. Ragazzini are with the Fondazione Ugo Bordoni, 00153 Rome, Italy.

II. The Algorithm

The lattice predictor is shown in Fig. 1, where $x(n)$ is the signal to be processed and $f_i(n)$, $b_i(n)$ are the forward and backward residuals at the output of the ith section. The nonstationarity of the

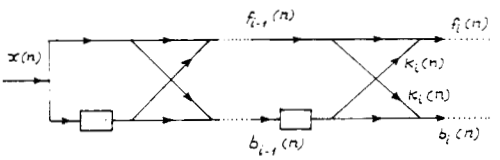

Fig. 1. Lattice predictor.

reflection coefficient K_i is modeled by expanding it into a linear combination of functions $g_j(n)$, $j = 1, \cdots, M$ of a suitable base, i.e.,

$$K_i(n) = \sum_{j=1}^{M} K_{ij} g_j(n). \tag{1}$$

The determination of the parameters K_{ij} of the predictor is usually carried out by solving the system of equations obtained by minimizing with respect to K_{ij} the sum ϵ of the variances of the forward and backward residuals at the output of the ith section. The resulting system is

$$\sum_{j=1}^{M} K_{ij} \langle A(n) g_j(n) g_r(n) \rangle = -2 \langle B(n) g_r(n) \rangle, \qquad r = 1, \cdots, M$$

$$A(n) = \left(f_{i-1}(n)\right)^2 + \left(b_{i-1}(n-1)\right)^2$$

$$B(n) = f_{i-1}(n) b_{i-1}(n-1) \tag{2}$$

where $\langle \cdot \rangle$ stands for mean value.

The previous system (2) yields the values K_{ij} which minimizes ϵ without any control on the resulting magnitude of $K_i(n)$. In order to introduce this control, we take advantage of the property of a function to be flat around its minimum. This property suggests to shift the solution from the minimum of ϵ under the conditions of satisfying the physical constraint $|K_i(n)| < \rho$ and of using the minimum possible shift. The resulting algorithm can be directly set under a form suitable to the application of linear programming techniques

$$\min \sum_{r=1}^{M} (\epsilon_{1r} + \epsilon_{2r})$$

$$\sum_{j=1}^{M} (K_{1ij} - K_{2ij}) \langle A(n) g_j(n) g_r(n) \rangle = -2 \langle B(n) g_r(n) \rangle (1 + \epsilon_{1r} - \epsilon_{2r}), \qquad r = 1, \cdots, M \tag{3}$$

$$K_{1ij}, K_{2ij}, \epsilon_{1r}, \epsilon_{2r} > 0$$

$$\begin{cases} \sum_{j=1}^{M} (K_{1ij} - K_{2ij}) g_j(n) + h_1(n) = \rho \\ \sum_{j=1}^{M} (K_{2ij} - K_{1ij}) g_j(n) + h_2(n) = \rho, \qquad n = 1, \cdots, N \end{cases} \tag{4}$$

$$h_1(n), h_2(n) > 0$$

where N is the number of samples of $x(n)$ and $K_{ij} = K_{1ij} - K_{2ij}$.

In the actual application of the previous algorithm it is convenient to reduce its computational cost at the expense of a less tight control on the magnitude of $K_i(n)$, since N is usually very large in the nonstationary case. A very simple way for accomplishing this goal is suggested by the remark that the maxima and the minima of $K_i(n)$ lie very close to the maxima and the minima of the basis

0018-9219/86/0500-0759$01.00 ©1986 IEEE