theorems will be discovered to the mutual benefit of these and other areas.

ACKNOWLEDGMENT

The author would like to thank Peter Ney of the University of Wisconsin, Madison and John Sadowsky of Purdue University, West Lafayette, IN, for several conversations on the interrelationships between large deviation theory and convex analysis. The author would like to thank Toby Berger of Cornell University, Ithaca, NY, for pointing out the simple proof of Lemma 1 given in the text. An anonymous reviewer should also be acknowledged for numerous suggestions resulting in the present form of this paper.

APPENDIX
PROOF OF LARGE DEVIATION LEMMA

Let $Q_n(dD)$ denote the distribution of W_n/n on $\mathbf{R}$, and define the "twisted" measures

$$Q_{n,\theta}(dD) = \exp[n\theta D]\cdot Q_n(dD)/\exp[nc_n(\theta)], \quad n=1,2,\cdots.$$

As D ranges over a neighborhood $G_\epsilon = (D_0-\epsilon, \delta_0+\epsilon)$ of D_0, then $-\theta D \geq -\theta D_0 - |\theta|\epsilon$, and, consequently, if $G_\epsilon \subset G$ we have

$$Q_n(G) \geq Q_n(G_\epsilon) = \exp[nc_n(\theta)]\cdot\int_{G_\epsilon}\exp[-n\theta D]Q_{n,\theta}(dD)$$

$$\geq \exp\left[n\left(c_n(\theta)-\theta D_0-|\theta|\epsilon\right)\right]\cdot Q_{n,\theta}(G_\epsilon).$$

It follows (letting $\theta=\theta_0$) that

$$\liminf_n \frac{1}{n}\log Q_n(G) \geq c(\theta_0)-\theta_0 D_0-|\theta_0|\epsilon + \liminf_n \frac{1}{n}\log Q_{n,\theta_0}(G_\epsilon^n).$$

Note that $c(\theta_0)-\theta_0 D_0 = -c^*(D_0)$ since $c(\theta)$ is differentiable at θ_0 with finite derivative $c'(\theta_0)=D_0$. If $Q_{n,\theta_0}(G_\epsilon)\to 1$ as $n\to\infty$, the lemma follows by letting $\epsilon\to 0$.

It remains to show that $Q_{n,\theta_0}(G_\epsilon)\to 1$ as $n\to\infty$. Introduce random variables V_n such that V_n/n has distribution $Q_{n,\theta_0}(dD)$. Note also for any random variable Z and $t\geq 0$ we have $P\{Z\geq z\}\leq E\{\exp[t(Z-z)]\} = \exp[-tz]\cdot E\{\exp[tZ]\}$. Hence

$$P\{V_n/n\geq D_0+\epsilon\} \leq E\{\exp[nt(V_n/n-D_0-\epsilon)]\}$$

$$= \int \exp[nt(V_n/n-D_0-\epsilon)] \cdot\exp[n\theta_0 D]Q_n(dD)/\exp[nc_n(\theta)]$$

$$= \exp\left[n\left(c_n(\theta_0+t)-c_n(\theta_0)-t(D_0+\epsilon)\right)\right].$$

Hence

$$\limsup_n \frac{1}{n}\log P\{V_n/n\geq D_0+\epsilon\} \leq c(\theta_0+t)-c(\theta_0)-t(D_0+\epsilon).$$

The right side is strictly negative for sufficiently small t since $[c(\theta_0+t)-c(\theta_0)]/t\to D_0$ as $t\to 0$. Similarly,

$$\limsup_n \frac{1}{n}\log P\{V_n/n\leq D_0-\epsilon\} \leq c(\theta_0-t)-c(\theta_0)+t(D_0-\epsilon)$$

which, again, is strictly negative for small t. The two bounds imply that $1-Q_{n,\theta_0}(G_\epsilon)$ vanishes exponentially fast and hence $Q_{n,\theta_0}(G_\epsilon)\to 1$ as $n\to\infty$.

REFERENCES

[1] T. Berger, *Rate Distortion Theory, A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[2] A. Perez, "Extensions of Shannon-McMillan's limit theorem to more general stochastic processes," in *Trans. 3rd Prague Conf. Information Theory, Statistical Decision Functions, and Random Processes*. Prague, Czech.: Publ. House Czech. Acad. Sci., and New York: Academic, pp. 545-574.

[3] J. Dunham, "A note on the abstract-alphabet block source coding with a fidelity criterion theorem," *IEEE Trans. Inform. Theory*, vol. IT-24, no. 6, p. 760, Nov. 1978.

[4] J. Kieffer, "A counterexample to Perez's generalization of the Shannon-McMillan theorem," *Ann. Prob.*, vol. 1, no. 2, pp. 362-364, 1973.

[5] ——, "Correction to 'A counterexample of Perez's generalization of the Shannon-McMillan theorem,'" *Ann. Prob.*, vol. 4, no. 1, pp. 153-154, 1976.

[6] M. S. Pinsker, "Sources of messages," *Probl. Peredach. Inform.*, vol. 14, pp. 5-20, 1963 (in Russian).

[7] J. A. Bucklew, "The source coding theorem via Sanov's theorem," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 907-909, Nov. 1987.

[8] J. Gärtner, "On large deviations from the invariant measure," *Theory Probl. Appl.*, vol. 22, pp. 24-39.

[9] R. S. Ellis, "Large deviations for a general class of random vectors," *Ann. Prob.*, vol. 12, no. 1, pp. 1-12, 1984.

Finite-State Codes

FABRIZIO POLLARA, MEMBER, IEEE, ROBERT J. McELIECE, FELLOW, IEEE, AND KHALED ABDEL-GHAFFAR

***Abstract*—We define and investigate a class of codes called *finite-state* (FS) codes. These codes, which generalize both block and convolutional codes, are defined by their encoders, which are finite-state machines with parallel inputs and outputs. We derive a family of upper bounds on the free distance of a given FS code, based on known upper bounds on the minimum distance of block codes. We then give a general construction for FS codes, based on some recent ideas of Ungerboeck, and show that in many cases the FS codes constructed in this way have a d_{free} which is the largest possible. We also discuss the issue of catastrophic error propagation (CEP) for FS codes and discover that to avoid CEP we must solve an interesting problem in graph theory, that of finding a "uniquely decodable edge labeling" of the state diagram.**

I. INTRODUCTION

We begin with a description of what we shall call a *finite-state (FS) encoder*. An (n,k,m) FS encoder is a q^m-state (time-invariant) finite-state machine (FSM) with k parallel inputs and n parallel outputs taken from a q-letter alphabet (Fig. 1).

The encoder starts from a fixed initial state. At each clock pulse, k symbols (the information symbols) are input to the encoder, and in response the encoder changes state and outputs n symbols (the code symbols). Thus if $(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots)$ is a sequence of k-symbol information blocks, then the encoder's output will be a sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2\cdots)$ of n-symbol code blocks, which we call a *code sequence*. The set of all such code sequences is called the code generated by the FS encoder. A code generated by a (n,k,m) FS encoder will be called an (n,k,m) finite-state code. We note that if only one state exists in the encoder, the resulting $(n,k,0)$ FS code is in fact an ordinary block code. Similarly, a linear convolutional code is just an FS code in which the finite-

Manuscript received April 9, 1987; revised December 21, 1987. This work was supported in part by the National Aeronautics and Space Administration, and by Grants from IBM and Pacific Bell.

F. Pollara is with the Jet Propulsion Laboratory, 238-420, 4800 Oak Grove Drive, Pasadena, CA 91109.

R. J. McEliece is with the Department of Electrical Engineering, 116-81, California Institute of Technology, Pasadena, CA 91125.

K. Abdel-Ghaffar was with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA. He is now with the IBM Almaden Research Center, San Jose, CA 95120.

IEEE Log Number 8824057.

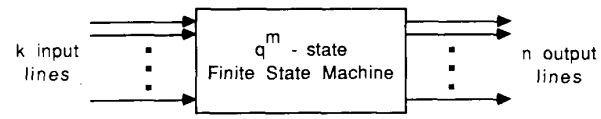

Fig. 1. Finite-state encoder.

state machine is a bank of k parallel shift registers, where each output symbol is a linear combination of the k input symbols and the symbols stored in the shift registers. Thus FS codes include both block and convolutional codes as special cases.

The *free distance* (d_{free}) of an FS code is defined to be the minimum Hamming distance between all pairs of distinct (infinite) code sequences. For given values of n, k, and m the FS code with the largest free distance will usually give the best performance. In this correspondence our concern is to find bounds on d_{free} in terms of the parameters n, k, and m to produce a family of FS codes meeting these bounds in certain cases. In Section II we derive our bounds. In Section III we describe a very general construction for (n, k, m) FS codes using ideas similar to those which are current in the trellis-coding literature [2], [6], [11]–[13]. In Section IV we discuss the issue of catastrophic error propagation and, using techniques from graph theory, describe an optimal noncatastrophic edge labeling of the complete 2^m state diagram. In Section V we combine the results of Sections III and IV to construct a class of Reed–Solomon-like FS codes which meet the bounds of Section II whenever $n \le q$ and $m \le \min(k-1, n-k-1)$. Finally, in Section VI we attempt to draw conclusions and compare some of our new codes with codes which have been previously studied.

II. Bounds on d_{free}

We derive a family of upper bounds on d_{free} in terms of the parameters n, k, and m. The basic idea is to find "subcodes" of a given FS code which are block codes and use the fact that any upper bound on the minimum distance of the block code is also an upper bound on the free distance of the parent code.

Here is some needed notation: let $\Delta(n, k)$ denote the largest possible minimum distance for a block code over a q-letter alphabet with length n and q^k codewords. (We note for future reference the fact that $\Delta(n, k)$ is meaningless for $k \le 0$.) The following theorem gives a bound on the free distance of a FS code in terms of Δ.

Theorem 1: For any FS code with parameters n, k, and m, the free distance is bounded as follows:

$$d_{\text{free}} \le \min_{L:\ Lk > m} \Delta(Ln, Lk - m).$$

Proof: We consider all possible input sequences consisting of L k-symbol input blocks $(\boldsymbol{u}, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_L)$. There are q^{Lk} such input sequences. For each of these sequences the encoder starts in the initial state and terminates in one of q^m states. It follows from the pigeon-hole principle that there must be at least q^{Lk-m} of these length-L input sequences which have the same final state. The code sequences corresponding to these input sequences can be thought of as a block code with length Ln with at least q^{Lk-m} codewords. The minimum distance of this block code is by definition at most $\Delta(Ln, Lk-m)$. On the other hand, by the definition given in Section I, the minimum distance of this block code is an upper bound on the d_{free} of the original FS code. Since this is true for all L, we apparently have

$$d_{\text{free}} \le \min_{L \ge 1} \Delta(Ln, Lk - m).$$

However, as we noted earlier, $\Delta(n, k)$ is meaningless if $k \le 0$, and so the minimization can only be taken over those values of L for which $Lk - m > 0$.

Corollary 1: If $k > m$, then $d_{\text{free}} \le \Delta(n, k-m)$.

Proof: If $k > m$, then it is permissible to take $L = 1$ in the minimization in Theorem 1.

Corollary 2: The free distance of an (n, k, m) FS code over a q-letter alphabet satisfies

$$\begin{aligned} d_{\text{free}} &\le \min_{L:\ Lk > m} (Ln - Lk + m + 1) \\ &= (n-k)\left\lfloor \frac{m}{k} + 1 \right\rfloor + m + 1 \\ &= n - k + 1 + m \qquad \text{if } k > m. \end{aligned}$$

Proof: This follows from Theorem 1 and the Singleton bound [8, theorem 1.11] which says that

$$\Delta(n, k) \le n - k + 1.$$

Corollary 3: The free distance of an FS code also satisfies

$$d_{\text{free}} \le \min_{L:\ Lk > m} \left(Ln \frac{q-1}{q} \frac{q^{kL-m}}{q^{kL-m} - 1} \right).$$

Proof: This follows from Theorem 1 and the Plotkin bound [1, theorem 13.49] which says that

$$\Delta(n, k) \le n \cdot \frac{q-1}{q} \cdot \frac{q^k}{q^k - 1}.$$

Comments: Most of the ideas used in the proof of Theorem 1 are already in the literature. The first such occurrence is in [10], where McEliece and Rumsey proved Corollary 3 for the special case of systematic convolutional codes with $k = 1$. Soon afterwards, Heller [7] generalized the result of McEliece and Rumsey to the case of nonsystematic convolutional codes still with $k = 1$. Then McEliece and Layland [9] further generalized the result to arbitrary time-varying linear convolutional codes and showed that block code bounds other than the Plotkin bound (in particular, the Griesmer bound) could sometimes be used to obtain useful bounds. Thus Theorem 1 adds little to what is contained in [9] except that it applies to nonlinear codes and explicitly points out that any upper bound on the minimum distance of a block code bound can be used to obtain an upper bound on the free distance of a FS code. Although in this correspondence we usually only need to take $L = 1$ in Theorem 1 and its corollaries, for many sets of parameter values, in particular, for the important case $n = 2$, $k = 1$, $q = 2$ (i.e., binary rate $1/2$ convolutional codes), the optimal value of L is typically much larger than 1.

III. Code Constructions

In the last section we derived bounds on d_{free} which apply to arbitrary FS codes. In this section, we describe a very general construction for a class of FS codes which is based on an idea that originated in Ungerboeck's landmark paper [11]. We find that many of the codes constructed by this technique meet the bounds of Section II.

Our basic idea is to start with an explicit state-transition diagram and build a code around this diagram. For definiteness, we will consider only *completely connected* q^m-state transition diagrams, in which every ordered pair of states is connected by a directed edge (illustrated in Fig. 2 for $q = 2$, $m = 2$); however, most of our ideas can be generalized to other diagrams.

The FS code is to have parameters n and k. This means that at every clock cycle, k symbols go into the encoder and n symbols come out. Of the k input symbols it is reasonable to suppose that m are used to determine the next state of the encoder (recall that there are q^m states altogether) and $k - m$ to determine which n symbols are to be output. Thus it is natural to think of the possible n-symbol output blocks associated with a fixed state transition as the words in an $(n, k-m)$ block code. Our basic idea is to assign an $(n, k-m)$ block code to each possible state transition.

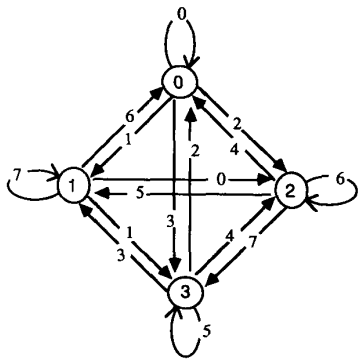

Fig. 2. Completely connected four-state diagram (edge labels are as described in Theorem 4).

Here then is our general construction for an (n, k, m) FS code. We begin with a linear (n, k_2) block code C_2 with minimum distance d_2. If we form the union of $q^{k_1-k_2}$ cosets of C_2, we will obtain an (n, k_1) block code C_1 (not necessarily linear) whose minimum distance we denote by d_1. We assign one of the cosets of C_2 to each of the q^{2m} state transitions in the state transition diagram. This assignment cannot be arbitrary. For example, we require that all q^m transitions originating at a given state or terminating at a given state be assigned a different subcode. This forces us to use at least q^m different cosets; however, to avoid catastrophic error propagation, we will need more than q^m subcodes, as we shall see in Section IV. If the big code C_1 has q^{k_1} codewords, where k_1 is an integer, this implies

$$k_1 \geq k_2 + m + 1.$$

The encoder now works as follows (see Fig. 3). The FSM is started in the initial state. At every clock pulse the encoder accepts k symbols. The first m of these symbols are input to the FSM. These m symbols determine the next state of the FSM and produce $k_1 - k_2$ output symbols which are used to select one of the $q^{k_1-k_2}$ cosets of C_2. The remaining $k - m = k_2$ input symbols are used to determine which of the q^{k_2} codewords from the selected coset is to be output from the encoder. Note that if $m = 0$, the encoder in Fig. 3 is simply an encoder for an (n, k) block code, and if $k_2 = 0$ and the FSM is linear, the encoder becomes an ordinary encoder for a convolutional code. We will use this fact in Section VI when we compare block, convolutional, and FS codes.

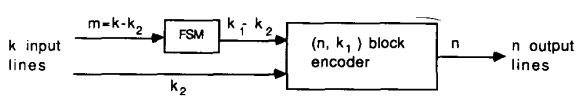

Fig. 3. Conceptual view of encoder.

This design is similar to a construction of Ungerboeck [12, part II, fig. 1] and the Class V codes of Forney [6, part I]. The key difference is that while Ungerboeck and Forney seek to construct codes over the reals with large Euclidean distance via lattice partitions, we seek to maximize Hamming distance via partitions of block codes. Still, as one of the referees has observed, no essential difference exists between constructing trellis codes to maximize Euclidean distance using sublattices of Z^n that lie above $2Z^n$ and constructing trellis codes to maximize Hamming distance.

In the next theorem we estimate the d_{free} of the code constructed in this way. The essential idea of this theorem is implicit in Ungerboeck [11].

Theorem 2: The free distance of the $(n, m + k_2, m)$ FS code constructed as described earlier satisfies

$$\min(d_2, 2d_1) \leq d_{\text{free}} \leq d_2.$$

Proof: We need to estimate the Hamming distance between pairs of code sequences corresponding to paths in the state diagram which begin and end in the same state. Let us say that these paths both begin in state s_1 and end in state s_K. There are two cases to consider: 1) when the *second* states in the two paths are the same and 2) when they are different.

In case 1) we look at only the first n-symbol block of each path. These two blocks are distinct codewords in the same (n, k_2) subcode and so must differ in at least d_2 positions. Thus if case 1) holds, the Hamming distance between the two code sequences is at least d_2. Furthermore, since d_2 is the minimum distance of each of the subcodes, we know that the Hamming distance between some pair of code sequences is exactly d_2. Thus $d_{\text{free}} \leq d_2$.

In case 2) the paths must differ in at least two edges: the edges leaving s_1 and the edges next entering a common state (there must be such a common state since the paths both terminate at s_K—see Fig. 4). The n-symbol blocks corresponding to these pairs of edges are distinct codewords in the (n, k_1) parent code since we have assumed that different subcodes are assigned to all state transitions beginning (or ending) in the same state, and so each pair must differ in at least d_1 positions. This means that the two code sequences must differ in at least $2d_1$ positions. Thus if case 2) holds, the Hamming distance between the two code sequences is at least $2d_1$. Combining cases 1) and 2) we obtain the statement of the theorem.

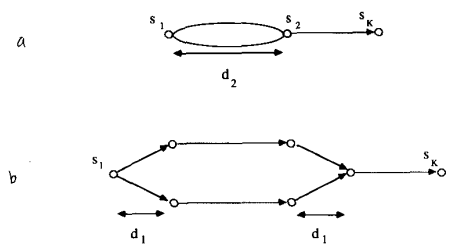

Fig. 4. Proof of Theorem 2. (a) Case 1. (b) Case 2.

Once the free distance of a FS code is known, the next important number to know is the *error coefficient*, which is the number of code sequences at distance d_{free} from a given code sequence. (If the code is linear, this number is independent of the code sequence chosen.) This is a difficult quantity to compute in general, but the following corollary is often useful.

Corollary 4: Let A_{d_2} denote the number of codewords of weight d_2 in the code C_2. If $d_2 \leq 2d_1$, then the error coefficient of the FS code constructed by Theorem 2 is $\geq A_{d_2}$. If $d_2 < 2d_1$, then the error coefficient is exactly equal to A_{d_2}.

Proof: According to Theorem 2, if $d_2 \leq 2d_1$, then $d_{\text{free}} = d_2$. According to the proof of Theorem 2, there are potentially two classes of code sequences at distance d_{free} from a given code sequence, viz. the "Case 1" and the "Case 2" code sequences. In Case 1 the only way to get distance d_2 is if the two length n code sequences are codewords at distance d_2 in the code C_2, and the number of codewords at distance d_2 from a given codeword is the number of words of weight d_2 since C_2 is assumed to be linear. This accounts for at least A_{d_2} code sequences at distance d_{free} from a given code sequence. However, if $d_2 < 2d_1$, Case 2 does not contribute to the error coefficient.

Example 1: Let $q = 2$, and consider the four-state state diagram of Fig. 2. We choose as the parent code C_1 a (16,5)

first-order Reed–Muller code with $d_1 = 8$. This code contains a (16,1) linear subcode C_2 (the repetition code) with $d_2 = 16$. There are 16 cosets of C_2 in C_1, and so it is possible to assign a different coset to each of the 16 state transitions in the state diagram. The result is a (16,3,2) code with (according to Theorem 3.1) $d_{\text{free}} = 16$. (A calculation which we omit shows that the error coefficient is 13. For each code sequence there is one "Case 1" code sequence, and 12 "Case 2" code sequences at distance 16.) On the other hand, by Corollary 1, we find that the free distance of a (16,3,2) code with $q = 2$ is at most 16. Therefore, the code constructed this way has the largest possible d_{free} for its given n, k, and m. (This example will be generalized in Example 4 in the next section.)

Example 2: We again take $q = 2$, and C_2 to be the (16,5), $d_{\min} = 8$ first-order Reed–Muller code. C_2 is a subcode of the (16,11) $d_{\min} = 4$ second-order Reed–Muller code which we will call C^*. If we take C_1 to be the union of eight cosets of C_2 in C^*, then C_1 will be a (16,8), $d_{\min} \geq 4$ code. Thus if we use the edge-labeling described in Fig. 2 to assign these eight cosets to the 16 state transitions, Theorem 2 tells us that we get a (16,7,2) FS code with $d_{\text{free}} = 8$. On the other hand, the bound in Corollary 3 (take $L = 1$) shows that any (16,7,2) FS code must have $d_{\text{free}} \leq 8$, and so any code constructed in this manner is optimum. By Corollary 4, the error coefficient of this code must be ≥ 30, which is the number of words of weight 8 in C_2. However, we note that if we choose the eight cosets properly, C_1 will be the (16,8) nonlinear Nordstrom–Robinson code, with $d_1 = 6$ [8, ch. 15]. In this case, we have $d_2 < 2d_1$, and so by Corollary 4, the error coefficient is exactly 30. This example will be generalized in Example 5 to follow.

The codes constructed by the techniques of this section are often, as we have seen, quite good if d_{free} is used as the figure of merit. However, they are not yet guaranteed to be noncatastrophic. In the next section we will address the problem of how to assign subcodes to state transitions to avoid catastrophy.

IV. Uniquely Decodable Edge Labelings

In Section III we showed how to construct an (n, k, m) FS code by assigning cosets of a subcode of an (n, k) block code to the edges of a state diagram. This coset-to-edge assignment must be done carefully. For example, for the construction of Theorem 2 to work, all edges leaving a given state and all edges entering a state must be assigned distinct cosets. A coset assignment that has this necessary property is called *nonsingular*. However, there is a subtler issue. If the coset-to-edge assignments are not done carefully, the resulting encoder could be catastrophic.

Here we study the problem of assigning cosets in a nonsingular manner so to avoid catastrophy. We will see that for the completely connected state diagram with q^m states at least $2q^m$ cosets are needed; we will also see one way to make a nonsingular noncatastrophic coset-edge assignment if q^{m+1} cosets are available. The key concept is that of a *uniquely decodable* edge-labeling of a state diagram, which we now define.

If s and t are two states, we denote by $L(s,t)$ the label on the directed edge from s to t. (In our application, the "labels" are cosets.) Let $(s_1, s_2, \cdots, s_K)$ and $(t_1, t_2, \cdots, t_K)$ be two sequences of K states. If the two corresponding sequences of labels

$$L(s_1, s_2), L(s_2, s_3), \cdots, L(s_{K-1}, s_K)$$

and

$$L(t_1, t_2), L(t_2, t_3), \cdots, L(t_{K-1}, t_K)$$

are identical, we call two such state sequences *label indistinguishable*.

Definition 1: An edge labeling of a state diagram is said to be uniquely decodable (UD) if and only if an integer K_0 exists such that any two label-indistinguishable state sequences of length $K \geq K_0$ are identical. (This says that any sufficiently long state sequence can be recovered uniquely from its label sequence.)

We will need uniquely decodable edge labelings to ensure that a bad burst of channel noise will never cause the decoder to make an infinite number of decoder errors, i.e., will not cause *catastrophic error propagation*. If the edge labeling is not uniquely decodable, catastrophic error propagation might happen. We can see this as follows. Let $(s_1, s_2, \cdots)$ and $(t_1, t_2, \cdots)$ be two arbitrarily long label-indistinguishable state sequences. If the encoder follows a state sequence that finishes with the sequence $(s_1, s_2, \cdots)$, it is possible for the channel noise to be such that the decoder will correctly determine the state sequence up to (but not including) s_1, but then choose t_1 instead of s_1. If this happens, the decoder will almost surely never recover since its metric calculations based on hypothesizing the incorrect state sequence $(t_1, t_2, \cdots)$ will be identical to those based on the true state sequence $(s_1, s_2, \cdots)$.

Notice that if all the edge labels are distinct, any state sequence can be uniquely recovered from its label sequence, and so the labeling is uniquely decodable. (If the state diagram is the complete N-state diagram, this requires N^2 labels.) On the other hand, if all edges in the state diagram have the same label, all pairs of state sequences are label indistinguishable, and so the labeling cannot be UD. The basic problem is to find the minimum number of different labels that are needed for a UD labeling. We do not know what this number is for an arbitrary state diagram. However, if we assume that the state diagram is *D-regular*, i.e., that there are exactly D edges coming into (and going out of) each state, the following theorem places a nontrivial lower bound on the required number of distinct labels.

Theorem 3: For a D-regular state diagram with at least two states, at least $2D$ distinct labels are required for a UD edge labeling.

Proof: As a first step, note that if we have labeling L such that $L(s,t) = L(s',t)$, where s and s' are distinct states, the labeling cannot be UD, since then $(s, s_2, s_3, s_4, \cdots, s_K)$ and $(s', s_2, s_3, s_4, \cdots, s_K)$ are label indistinguishable but not identical state sequences for any value of K. Similarly, if $L(s,t) = L(s,t')$, where t and t' are distinct states, the labeling is also non-UD since $(s_1, s_2, \cdots, s_{K-1}, t)$ and $(s_1, s_2, \cdots, s_{K-1}, t')$ are label indistinguishable but not identical state sequences for any value of K. Thus we have for any UD labeling that

$$L(s,t) \neq L(s',t), \qquad \text{if } s \neq s'$$
$$L(s,t) \neq L(s,t'), \qquad \text{if } t \neq t'.$$

In other words, if we are given x and $L(x, y)$, we can recover y; if we are given y and $L(x, y)$, we can recover x. If the labeling has this property we say that it is *nonsingular*. Thus all UD labelings are nonsingular, but the converse may not hold.

Next we assume that we are given a UD labeling of a D-regular state diagram, but that the labeling uses fewer than $2D$ labels. We will show by induction that this assumption leads to a contradiction by constructing a pair of arbitrarily long nonidentical but label-indistinguishable state sequences. If s_1 and t_1 are distinct states, then (s_1) and (t_1) are two label-indistinguishable but distinct state sequences of length 1. Now assume that we have already constructed a pair of nonidentical but label-indistinguishable state sequences of length K, say $(s_1, s_2, \cdots, s_K)$ and $(t_1, t_2, \cdots, t_K)$. (We have just seen that we can do this for $K = 1$.) Since the labeling is nonsingular, we know that $s_K \neq t_K$. Now consider the $2D$ labels of the form $L(s_K, s)$ and $L(t_K, s)$. Since there are fewer than $2D$ labels available, two of these labels must be identical. These identical labels cannot be of the form $L(s_K, s)$ and $L(s_K, s')$, or $L(t_K, t)$ and $L(t_K, t')$ since the labeling is nonsingular. Hence we must have $L(s_K, s) = L(t_K, t)$ for some s and t. Thus if we set $s_{K+1} = s$ and $t_{K+1} = t$, then $(s_1, s_2, \cdots, s_{K+1})$ and $(t_1, t_2, \cdots, t_{K+1})$ are nonidentical but label-indistinguishable state sequences of length $K+1$. This completes the proof that $2D$ labels are necessary in any UD edge labeling.

Theorem 3 says that $2D$ labels are necessary for a UD labeling of an D-regular state diagram. However, $2D$ labels are not always sufficient as can be seen by considering a three-state state diagram with each state connected only to itself with a loop. Here $D=1$, but plainly three labels are required since the loops must all have different labels. However, the next theorem implies that $2D$ labels are sufficient if D is a power of two and if the state diagram is completely connected.

Theorem 4: The completely connected q^m-state diagram has a UD labeling that uses only q^{m+1} labels, so that every sequence of m edge labels uniquely identifies the state sequence.

Proof: Let us number the q^m states with the integers in the set $\{0,1,\cdots,q^{m-1}\}$. We will use the integers in the set $\{0,1,\cdots,q^{m+1}-1\}$ as edge labels. Indeed, if x and y are two states, we label the directed edge from x to y with the integer $L(x,y)=y-qx \bmod q^{m+1}$. We claim that this labeling is UD. As a first step in this direction we note that this labeling is nonsingular. To see this, note that if we are given x and $L=y-qx \bmod q^{m+1}$, then $y=L+qx \bmod q^{m+1}$; if we are given y and L, then $x=(y-L)/q \bmod q^m$.

To see why the labeling is UD, let $x_0,x_1,\cdots,x_m$ and $y_0,y_1,\cdots,y_m$ be a pair of label-indistinguishable state sequences of length $m+1$, and let L_i be the common label on the edges $x_{i-1}\rightarrow x_i$ and $y_{i-1}\rightarrow y_i$. Then (all arithmetic is interpreted mod q^{m+1}) we have

$$L_i = x_i - qx_{i-1} = y_i - qy_{i-1}, \qquad \text{for } i=1,2,\cdots,m.$$

From this we conclude that

$$\begin{aligned} x_1 &= L_1 + qx_0 \\ x_2 &= L_2 + qL_1 + q^2x_0 \\ &\vdots \\ x_m &= L_m + \cdots + q^{m-1}L_1 + q^m x_0. \end{aligned}$$

It follows from this that

$$x_m = L_m + \cdots + q^{m-1}L_1 \bmod q^m.$$

Thus x_m depends only on the m labels $L_1,\cdots,L_m$ and not on the initial state x_0. Since y_m can be computed in exactly the same way, it follows that $x_m=y_m$. Since the labeling is nonsingular, it now follows that $x_{m-1}=y_{m-1},\cdots,x_1=y_1$, and so the two state sequences are identical. This proves that the given labeling is UD and indeed that any state sequence can be identified after at most m labels.

Comment: Another way to produce a UD labeling of the completely connected q^m state diagram has been suggested by Forney [6]. Take a noncatastrophic encoder for a $(m+1,m,m)$ convolutional code whose state diagram is the completely connected q^m state diagram, and label each state transition with the encoder's output corresponding to this state transition. This will work provided the encoder is also "nonsingular" (which is not hard to ensure). This idea has recently been expanded upon by Cheung and Popović [3].

Example 3: The edge labels prescribed by the construction of Theorem 4 in the case $q=2$, $m=2$ are given in Fig. 2; in the case $q=2$, $m=3$ they are given in the following 8×8 matrix:

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	14	15	0	1	2	3	4	5
2	12	13	14	15	0	1	2	3
3	10	11	12	13	14	15	0	1
4	8	9	10	11	12	13	14	15
5	6	7	8	9	10	11	12	13
6	4	5	6	7	8	9	10	11
7	2	3	4	5	6	7	8	9

where the (x,y) entry of the matrix gives the label on the $x\rightarrow y$ state transition. Notice, for example, that the label sequence (7,4) is ambiguous (the state sequences (0,7,5) and (7,5,1) both yield the label sequence (7,4)), but the label sequence (7,4,11) uniquely specifies the state sequence (0,7,5,5). We can combine Theorems 2 and 4 to give the following general construction for linear FS codes.

Theorem 5: Suppose that C_1 is an (n,k_1), $d_{\min}=d_1$ block code over GF(q) and is the union of $q^{k_1-k_2}$ cosets of C_2, an (n,k_2), $d_{\min}=d_2$ block code. Then for every m in the range $0\le m\le k_1-k_2-1$, there exists a noncatastrophic $(n,m+k_2,m)$ FS code with $\min(d_2,2d_1)\le d_{\text{free}}\le d_2$.

Proof: Using the construction of Section III, we begin with a completely connected q^m-state diagram, where $0\le m\le k_1-k_2-1$. Since there are $q^{k_1-k_2}$ cosets of C_2 in C_1, by Theorem 4 it is possible to use these cosets to make a UD labeling of the state diagram. By Theorem 2 the result is an $(n,m+k_2,m)$ FS code whose free distance satisfies the bounds given in the statement of the theorem.

Example 4 (cf. Example 1): Let $q=2$; let C_1 be the $(2^j,j+1)$, $d_1=2^{j-1}$ first-order Reed–Muller code, and let C_2 be the $(2^j,1)$, $d_2=2^j$ repetition code with is a subcode of C_1. Using Theorem 5, for each $0\le m\le j-1$ we can construct a $(2^j,m+1,m)$, $d_{\text{free}}=2^j$ FS code which by Corollary 1 is optimal. A calculation (which we omit) shows that the error coefficient for this code when $j=m+1$ is $2^{m+3}-4m-7$. (For $m=2$ this gives 17, as compared to the error coefficient 13 given in Example 1. This discrepancy is due to the edge labeling; in Example 1, 16 labels were used, but the construction given here only requires eight labels. The price paid for the more efficient labeling is a larger error coefficient.)

Example 5 (cf. Example 2): Let j be even, let C_2 be the $(2^j,j+1)$ first-order Reed–Muller code with $d_2=2^{j-1}$, and C_1 be the $(2^j,2j)$, $d_1=2^{j-1}-2^{(j-2)/2}$ Kerdock code which is known to be the union of 2^{j-1} cosets of C_2. Then, using Theorem 5, for each $0\le m\le j-2$ we can construct a $(2^j,m+j+1,m)$ FS code with $d_{\text{free}}=2^{j-1}$, which by Corollary 3 (again take $L=1$) is the largest possible. Also, since $2d_1>d_2$, it follows from Corollary 4 that the error coefficient is equal to the number of words of weight d_2 in C_2, viz. $2^{j+1}-2$.

The FS codes constructed in Example 4 can be thought of as generalizing the $(2^j,1)$ repetition codes, and those in Example 5 as generalizing the $(2^j,j+1)$ first-order Reed–Muller codes. We could continue this series of examples, using the fact that the rth order Reed–Muller code is a subcode of the $(r+1)$st order Reed–Muller code and construct higher order "Reed–Muller-like" FS codes. However, the FS codes constructed this way would not have the largest possible d_{free} since the higher order RM codes themselves are not particularly good. A much more interesting possibility is to generalize the family of *Reed–Solomon* codes. We consider this in the next section.

V. Reed–Solomon FS Codes

Now we use the techniques of Sections II–IV to construct a class of FS codes which are "Reed–Solomon-like" and which meet the bounds of Section II in many cases.

The ancestors of the FS codes to be constructed are Reed–Solomon codes. Reed–Solomon codes are (n,k) block codes with minimum distance $d=n-k+1$ (thus achieving the Singleton bound). These codes exist for all n and k satisfying $1\le k\le n\le q$, where q is the alphabet size. (See [8, ch. 10].)

Our goal is to use a Reed–Solomon code to construct a (n,k,m) FS code with $d_{\text{free}}=n-k+1+m$, which by Corollary 2 is the largest possible value. Following the prescription in Section III, we let C_1 be an $(n,k+1)$ Reed–Solomon code with $d_1=n-k$. The subcode C_2 is taken to be an $(n,k-m)$ Reed–Solomon code, with $d_2=n-k+1+m$ (this requires $m<$

k). Thus by Theorem 5 the resulting (n, k, m) code will have

$$\min(2(n-k), n-k+1+m) \le d_{\text{free}} \le n-k+1+m.$$

If $2(n-k) \ge n-k+1+m$, i.e., $m \le n-k-1$, this gives $d_{\text{free}} = n-k+1+m$ and by Corollary 1 the free distance cannot be larger than this. Therefore, we have proved the following.

Theorem 6: For any parameters n, k, m, and q satisfying $k \le n-1 \le q-1$ and

$$m \le \min(k-1, n-k-1)$$

a noncatastrophic (n, k, m) FS code exists whose free distance meets the Singleton bound, viz.

$$d_{\text{free}} = n-k+1+m.$$

Example 6: If we start with a (15,11) RS code over GF(16), a code with $d_{\min} = 5$, (thus $k = 10$ in the construction described earlier), we can construct (15,10, m) FS codes for $0 \le m \le 10$. For $m = 0$, 1, 2, 3, and 4, the codes have $d_{\text{free}} = m+6$ which agrees with the Singleton bound of Corollary 2, and so these codes are all optimal with respect to free distance. For $m = 0$, 1, 2, and 3, the error coefficients of these codes can be calculated by Corollary 4 together with the known result [8, ch. 11] that the number of words of weight $d_{\min}$ in a Reed–Solomon code is $(q-1)\binom{n}{d_{\min}}$. For $m = 4$, however, we do not know the error coefficient. (For $5 \le m \le 10$ the codes constructed all have $d_{\text{free}} = 10$, independent of m, and do not meet the Singleton bound; we conjecture that they do not have the largest possible d_{free} for their values of n, k, and m.)

Even though the RS-like codes constructed by Theorem 6 are optimal with respect to free distance, they may be prohibitively complex to decode. For example, one possible decoder for the $m = 1$ code of Example 6 would require an array of 16 parallel decoders for a (15,9) RS code over GF(16). As a comparison, the $m = 0$ code would require only a single decoder for a (15,10) RS code. This increase in decoder complexity would seem to outweigh the advantages of increasing the free distance by one. The design of practical decoders for these codes is a challenging research problem.

VI. Conclusion

To compare some of the new codes we have presented to existing codes, both in terms of coding gain and decoding complexity, we define, for each FS code, two figures of merit, the *asymptotic coding gain* (ACG) and the *Viterbi decoding complexity* (VDC).

We define the asymptotic coding gain of an FS code as

$$\text{ACG} = \frac{k}{n} d_{\text{free}}.$$

Although this definition has been rigorously justified only for binary codes used with binary phase-shift keying (BPSK) modulation on a Gaussian channel with high signal-to-noise ratio (see [4, sec. 1.3], for example), it nevertheless gives a very handy way of making quick comparisons of the performance of codes competing in other arenas as well.

Similarly, we define the *Viterbi decoding complexity* of an FS code as

$$\text{VDC} = \frac{n}{k} q^{m+k}.$$

This quantity is a measure of the number of symbol operations required to decode one q-ary information symbol in an FS code if a Viterbi-like algorithm is used to decode. It is assumed that such a decoder uses a q^m state diagram in which each state is connected to q^{k-k_2} other states and that each state transition corresponds to q^{k_2} possible code blocks (see Fig. 3). Thus to completely update all q^m metrics requires $q_m \cdot q^{k-k_2} \cdot q^{k_2} = q^{m+k}$ block comparisons and n times this many symbol comparisons. These nq^{m+k} symbol comparisons yield decoder estimates of k information symbols, resulting in the given formula for VDC.

The parameters ACG and VDC make it easy to make rough comparisons of block, convolutional, and FS codes with differing values of n, k, m, and d_{free}. For example, in the following table we compare four representative codes: the (24,12) Golay block code, the (16,8) Nordstrom–Robinson (NR) code, the (2,1,6) convolutional code which has been adopted by NASA for deep-space communication, and the (16,7,2) FS code we introduced in Example 2. In each case we have listed the parameters of the code in the format $(n, k, m; d_{\text{free}})$:

Code	ACG	VDC
(24,12,0; 8) (Golay)	4.00	8192
(16,8,0; 6) (NR)	3.00	512
(2,1,6; 10) (NASA)	5.00	256
(16,7,2; 8) (Example 2)	3.50	1170

According to this table the NASA code ranks first, though not dramatically so, both in terms of coding gain and in terms of decoder complexity. The Golay code is second in terms of ACG but last in terms of VDC, while the NR code is second in terms of VDC but last in terms of coding gain. Finally, the code of Example 2 ranks third in both ACG and VDC. Of course, the table does not tell the whole story. For example, the ACG provides a reliable comparison only at high signal-to-noise ratios, and we have performed computer simulations which indicate that the code of Example 2 performs essentially better than the Golay code and as well or better than the NASA code for decoded bit error probabilities greater than about 10^{-3}. (For applications to concatenated systems such large error probabilities are desirable). Also, the VDC measure of complexity assumes a "brute-force" decoder for the cosets of the code C_2, whereas in practice there will often be much less complex algorithms available. For example, there are several known ways to decode the Golay code which require fewer than $2^{13} = 8192$ operations per decoded bit, and the decoder for the code of Example 2 can be simplified using the "Green Machine" [8, ch. 14] decoder for the (16,5) code.

We find it gratifying that the simple construction of Example 2 produces a code which competes so favorably with these three powerful and well-known codes. This leads us to hope that further research will produce FS codes that will find important practical applications.

References

[1] E. R. Berlekamp, *Algebraic Coding Theory*, revised 1984 ed. Laguna Hills, CA: Aegean Park Press, 1984.

[2] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177–196, 1987.

[3] K.-M. Cheung and L. Popović, "A new labeling procedure for trellises," in *Proc. IEEE Int. Symp. Inform. Theory*, p. 147, June 1988.

[4] G. C. Clark and J. B. Cain, *Error-Correcting Coding for Digital Communications*. New York: Plenum, 1981.

[5] G. D. Forney *et al.*, "Efficient modulation on band-limited channels," *IEEE J. Selected Areas Commun.*, vol. SAC-2, pp. 632–697, 1984.

[6] G. D. Forney, "Coset Codes—Part I," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1123–1151, Sept. 1988, Part II.

[7] J. A. Heller, "Short constraint length convolutional codes," *Jet Propulsion Lab. Space Prog. Summary*, vol. 37-54, pp. 171–176, 1969.

[8] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[9] R. J. McEliece and J. W. Layland, "An upper bound on the free distance of a tree code," *Jet Propulsion Lab. Space Programs Summary*, vol. 37-62, pp. 63–64, 1970.

[10] R. J. McEliece and H. C. Rumsey, Jr., "Capabilities of convolutional codes," *Jet Propulsion Lab. Space Programs Summary*, vol. 37-50, pp. 248–252, 1968.

[11] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.
[12] ____, "Trellis-coded modulation with redundant signal sets, Parts I and II," *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 5–21, Feb. 1987.
[13] L.-F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 483–501, 1987.

A Generalized Convergence Theorem for Neural Networks

JEHOSHUA BRUCK AND JOSEPH W. GOODMAN, FELLOW, IEEE

Abstract—**A neural network model is presented in which each neuron performs a threshold logic function. An important property of the model is that it always converges to a stable state when operating in a serial mode and to a cycle of length at most 2 when operating in a fully parallel mode. This property is the basis for the potential applications of the model, such as associative memory devices and combinatorial optimization. The two known convergence theorems (for serial and fully parallel modes of operation) are reviewed, and a general convergence theorem is presented which unifies the two known cases. Some new applications of the model for combinatorial optimization are also presented, in particular, new relations between the neural network model and the problem of finding a minimum cut in a graph.**

I. Introduction

The neural network model is a discrete-time system that can be represented by a weighted and undirected graph. A weight is attached to each edge of the graph and a threshold value attached to each node (neuron) of the graph. The *order* of the network is the number of nodes in the corresponding graph. Let N be a neural network of order n; then N is uniquely defined by (W, T) where

- W is an $n \times n$ symmetric matrix, where W_{ij} is equal to the weight attached to edge (i, j);
- T is a vector of dimension n, where T_i denotes the threshold attached to node i.

Every node (neuron) can be in one of two possible states, either 1 or -1. The state of node i at time t is denoted by $V_i(t)$. The *state* of the neural network at time t is the vector $V(t)$.

The next state of a node is computed by

$$V_i(t+1) = \operatorname{sgn}(H_i(t)) = \begin{cases} 1, & \text{if } H_i(t) \geq 0; \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

where

$$H_i(t) = \sum_{j=1}^{n} W_{j,i} V_j(t) - T_i.$$

The next state of the network, i.e., $V(t+1)$, is computed from the current state by performing the evaluation (1) at a set S of the nodes of the network. The modes of operation are determined by the method by which the set S is selected in each time interval. If the computation is performed at a single node in any time interval, i.e., $|S| = 1$, then we say that the network is operating in a *serial* mode; if $|S| = n$, then we say that the network is operating in a *fully parallel* mode. All the other cases, i.e., $1 < |S| < n$, will be called *parallel* modes of operation. The set S can be chosen at random or according to some deterministic rule.

A state $V(t)$ is called *stable* if and only if $V(t) = \operatorname{sgn}(WV(t) - T)$, i.e., no change occurs in the state of the network regardless of the mode of operation.

An important property of the model is that it always converges to a stable state when operating in a serial mode and to a cycle of length at most 2 when operating in a fully parallel mode [3], [5]. Section II contains a description of these convergence properties and a general convergence theorem which unifies the two known cases. New relations between the energy functions which correspond to the serial and fully parallel modes are presented as well.

The convergence properties are the basis for the application of the model in combinatorial optimization. In section III we describe the potential applications of a neural network model as a local search device for the two modes of operation, that is, serial mode and fully parallel mode. In particular, we show that an equivalence exists between finding a maximal value of the energy function and finding a minimum cut in an undirected graph, and also that a neural network model can be designed to perform a local search for a minimum cut in a directed graph.

II. Convergence Theorems

An important property of the model is that it always converges, as summarized by the following theorem.

Theorem 1: Let $N = (W, T)$ be a neural network, with W being a symmetric matrix; then the following hold.

1) Hopfield [5]: If N is operating in a serial mode and the elements of the diagonal of W are nonnegative, the network will always converge to a stable state (i.e., there are no cycles in the state space).

2) Goles [3]: If N is operating in a fully parallel mode, the network will always converge to a stable state or to a cycle of length 2 (i.e., the cycles in the state space are of length ≤ 2).

The main idea in the proof of the two parts of the theorem is to define a so-called *energy function* and to show that this energy function is nondecreasing when the state of the network changes. Since the energy function is bounded from above, the energy will converge to some value. Note that, originally, the energy function was defined so that it is nonincreasing [3], [5]; we changed it to be nondecreasing in accordance with some known graph problems (see, e.g., min cut in the next section).

The second step in the proof is to show that constant energy implies in the first case a stable state and in the second a cycle of length ≤ 2. The energy functions defined for each part of the proof are different:

$$E_1(t) = V^T(t) WV(t) - (V(t) + V(t))^T T$$
$$E_2(t) = V^T(t) WV(t-1) - (V(t) + V(t-1))^T T \quad (2)$$

where $E_1(t)$ and $E_2(t)$ denote the energy functions related to the first and second part of the proof.

An interesting question is whether two different energy functions are needed to prove the two parts of Theorem 1. A new result is that convergence in the fully parallel mode can be proven using the result on convergence for the serial mode of operation. For the sake of completeness, the proof for the case of a serial mode of operation follows.

Proof of the First Part of Theorem 1: Using the definitions in (1) and (2), let $\Delta E = E_1(t+1) - E_1(t)$ be the difference in the energy associated with two consecutive states, and let ΔV_k denote the difference between the next state and the current state of

Manuscript received June 1987; revised November 1987. This work was supported in part by the Rothschild Foundation and by the U.S. Air Force Office of Scientific Research. This correspondence was presented in part at the IEEE First International Conference on Neural Networks, San Diego, CA, June 1987.

The authors are with the Information Systems Laboratory, Department of Electrical Engineering, Durand Building, Stanford University, Stanford, CA 94305.

IEEE Log Number 8824061.

0018-9448/88/0900-1089$01.00 ©1988 IEEE