STUDY OF SEQUENTIAL DECODING

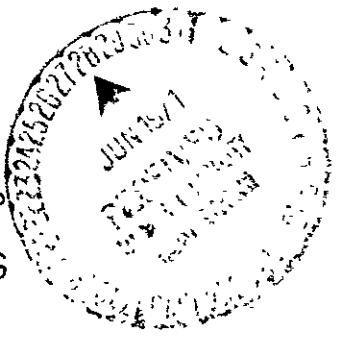
PROGRESS REPORT TO

NATIONAL AERONAUTICS

AND

SPACE ADMINISTRATION

CONTRACT NAS 2-5643



INFORMATION AND DECISION THEORY GROUP

SCHOOL OF ELECTRICAL ENGINEERING

CORNELL UNIVERSITY

Ithaca, New York

NASA CR 114277

STUDY OF SEQUENTIAL DECODING

By Dr. Frederick Jelinek

September 1970

Prepared under Contract No. NAS 2-5643 by

School of Electrical Engineering
Cornell University
Ithaca, New York

for

AMES RESEARCH CENTER

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
II.	REPORT OF PHASE I	2
	A. Work on Sequential Decoding	2
	B. Work on Bootstrap Hybrid Decoding	12
	C. Development of Good Convolutional Codes	35
	D. Application of Bootstrapping to Maximum Likelihood Decoding of Convolutional Codes	44
	References for Part II	46
	Figures for Part II	49 - 69
III.	REPORT OF PHASE II	70
	A. Tree Encoding of Sources with a Fidelity Criterion	70
	B. Permutation Codes as Source Codes	86
	References for Part III	102
	Figures for Part III	105 - 110
	APPENDIX 1 - Details of Stack and Map Maintenance and Purging	117
	APPENDIX 2 - Table Look-up for Multibranch Advance	126
	APPENDIX 3 - Description of the Rudimentary and Pull-up Decoding Algorithms	135
	APPENDIX 4 - New Upper Bounds on Certain Computational Parameters of Bootstrap Hybrid Sequential Decoding	145
	APPENDIX 5 - Estimation of EN_a and EN_b	161
	APPENDIX 6 - Proofs and Algorithms for Permutation Coding	166

I. INTRODUCTION

This final report on project NAS 2-5643, Research in Sequential Decoding consists of two main portions: results of Phase I and II of our work.

Phase I deals with problems of reliable transmission through noisy space channels and is subdivided into four areas: A. Work on sequential decoding in general and the Stack algorithm in particular. B. Work on the Bootstrap Hybrid Scheme. C. Development of good convolutional codes. D. Development of a new bootstrapping hybrid approach to the Viterbi decoding algorithm.

Phase II of the project deals with problems of encoding of space sources for the purpose of data compression. It is subdivided into two areas. A. Work on tree encoding with fidelity criterion. B. Work on Permutation encoding with a fidelity criterion.

This report is written according to the above outline. A substantial portion of it has already been presented in the three preceeding quarterly progress reports. We follow the precedent established there: The results are summarized and their implications are discussed in the body of the report, but details are left for Appendices.

II. REPORT ON PHASE I

II-A. Work on Sequential Decoding

II-A-1. Path Specifications in terms of Parity Digits

In this section we will describe how parity digits of binary convolutional codes should be used to speed up sequential decoding both by the Fano and the Stack algorithms. We will show what information ought to be saved so that the decoded message sequence can be recovered by the user. We confine ourselves to rate 1/2 codes, but generalization to rate 1/n codes is very simple.

Let $G(D)$ of degree $\upsilon-1$ be a binary convolutional generator, and let $S(D)$ be the input information sequence. The output sequence is then given by

$$X(D) = G(D)\ S(D) = \sum_{i=o}^{\infty} s_i\ D^i\ G(D) \qquad (1)$$

The digital circuit corresponding to (1) is given in Figure 1a. The contents of the shift register stages P_i are "0"'s at time 0. Let $P^n(D) = \sum_{i=o}^{\upsilon-2} p_i^n\ D^i$ be the shift register state sequence after s_{n-1} has been inserted. Then the output at time n + 1 is

$$x_n = s_n + p_o^n \qquad (2)$$

and in general, all the future outputs depend only on the initial state state sequence $P^n(D)$ and on the future inputs s_n, s_{n+1},...:

$$\sum_{i=0}^{\infty} x_{n+i}\ D^i = \sum_{j=o}^{\upsilon-2} p_j^n\ D^j + \sum_{j=o}^{\infty} s_{n+j}\ D^j\ G(D) \qquad (3)$$

The realization of Figure 1a is particularly convenient for digital computer implementation. Let $G^*(D)$ be defined by

$$G(D) = g_o + DG^*(D) \tag{4}$$

Then

$$P^n(D) = D^{-1} P^{n-1}(D) + p_o^{n-1} D^{-1} + s_{n-1} G^*(D) \tag{5}$$

with $P^o(D) = 0$. It follows from (2) and (5) that if the parity sequence $P^n(D)$ and the truncated generator sequence $G^*(D)$ are stored in index registers, then if $s_n = 1$, the output will be the complement $p_o^n + 1$ of the rightmost stage of the parity register, and the next parity register contents will be obtained by first a shift right of that register followed by an exclusive or into it of the contents of the generator register. Similarly, if $s_n = 0$ then $x_n = p_o^n$ and the next parity register contents are obtained by a shift right of the former contents. It follows that as long as $\nu-1$ does not exceed the size of the computer register, the number of operations necessary to generate $X(D)$ does not grow with ν.

In sequential decoding (this applies to both Fano and Stack algorithms), one must store as much information about a path being worked on as would be necessary for recovery of the message sequence corresponding to it. This is so because the path may become the decoded one in which case its message sequence must be supplied to the user. We will now show how $s_0, s_1, \ldots, s_{n-1}$ may be recovered from $P^n(D)$ and $p_o^{n-1}, p_o^{n-2}, \ldots, p_o^{\nu-1}$ provided $g_{\nu-1} = 1$ (which is so without loss of generality). In fact, since $D^{-1} P^n(D) + p_o^n D^{-1}$ is of degree $\nu-3$ then it follows from (5) that

$$s_{n-1} = p_{\nu-2}^n \tag{6}$$

Furthermore, using (6), for all $n = 1,2,\ldots$

$$P^{n-1}(D) = D\,P^{n}(D) + p_o^{n-1} + p_{\nu-2}^{n}\,DG^*(D) \tag{7}$$

Thus both s_{n-1} and $P^{n-1}(D)$ can be obtained from $P^n(D)$ and p_ν^{n-1}. By recursion therefore, $P^n(D)$, $p_o^{n-1},\ldots,p_0^1$ determine $s_{n-1},s_{n-2},\ldots,s_o$. However, it follows directly from Figure 1 that for $k = 1,2,\ldots$

$$P^{k}(D) = \left\lfloor D^{-(k-1)}\,G^*(D)\sum_{i=o}^{k-1} s_i\,D^i \right\rfloor \tag{8}$$

where $\lfloor\ \rfloor$ denotes the operation of dropping all negative degree terms. Since $g_{\nu-1} = 1$ then

$$s_{\nu-2} = p_{\nu-2}^{\nu-1} \tag{9}$$

Let

$$R^{\nu-1}(D) = P^{\nu-1}(D) \quad \text{and for} \quad k = 1,2,\ldots,\nu-1,$$

$$R^{k-1}(D) = D\left[R^{k}(D) + r_{\nu-2}^{k}\,G^*(D)\right] \mod D^{\nu-1} \tag{10}$$

then it follows from (8) and (10) that

$$s_i = r_{\nu-2}^{i+1} \qquad \text{for} \quad i = 0,1,\ldots,\nu-2 \tag{11}$$

Thus $s_o;s_1,\ldots,s_{\nu-2}$ may be recovered from $P^{\nu-1}(D)$ so that only $P^n(D)$, $p_o^{n-1},\ldots,p_o^{\nu-1}$ determine $s_{n-1},s_{n-2},\ldots,s_o$ as asserted. Figure 1b shows the digital circuit that does the job. It has the structure that performs according to (6) and (7). However if we feed into it the sequence

$$p_o^{n-1},p_o^{n-2},\ldots,p_o^{\nu-1},\underbrace{0\ ,\ 0\ ,\ \ldots\ ,\ 0}_{\nu-1\ \text{times}} \tag{12}$$

then after $n - \nu + 1$ shifts the state sequence will be $R^{\nu-1}(D)$, and after $n - \nu + i$ shifts it will be $R^{\nu-i}(D)$. The outputs will be $s_{n-1}, s_{n-2}, \ldots, s_o$ as indicated. The computer implementation of the process of Figure 1b is similar to that of Figure 1a. It shall be observed that it is possible to recover $s_n, s_{n-1}, \ldots, s_{n-k}$ from $P^n(D)$ if we feed the sequence $p_o^{n-1}, p_o^{n-2}, \ldots, p_o^{n-k+\nu-2}, 0, \ldots, 0$ into Figure 1b.

We shall now apply the above results first to Fano decoding and then to Stack decoding. In Fano decoding it is necessary to generate both $P^{n-1}(D)$ and $P^{n+1}(D)$ from $P^n(D)$ and when receding to find the likelihood of the preceding mode. Consider a rate 1/2 code with generators

$$G_1(D) = \sum_{i=o}^{\lambda-1} g_{1,i} D^i = 1 + DG^*_1(D)$$
$$G_2(D) = \sum_{i=o}^{\nu-1} g_{2,i} D^i = 1 + DG^*_2(D) \tag{13}$$

where $\lambda \leq \nu$ and $g_{1,o} = g_{2,o} = g_{1,\lambda-1} = g_{2,\nu-1} = 1$. For a systematic code, $\lambda = 1$ and $G^*_1(D) = 0$. The coder outputs are

$$X_1(D) = G_1(D)\, S(D) = \sum_{i=o}^{\infty} X_{1,i} D^i$$
$$X_2(D) = G_2(D)\, S(D) = \sum_{i=o}^{\infty} X_{1,i} D^i \tag{14}$$

If $Y_1(D)$ and $Y_2(D)$ are the corresponding received sequences (which need not be binary) then a likelihood of a branch at depth n is given by

$$\lambda(x_{1,n}, y_{1,n}) + \lambda(x_{2,n}, y_{2,n})$$

Therefore, for fast retreat, it would be useful if the decoder, located at depth n stored the unrelative likelihood L_n, the sequences $x_{1,o},\cdots,x_{1,n-1}$ and $x_{2,o},\cdots,x_{2,n-1}$ (as well as the received sequences $Y_1(D)$ and $Y_2(D)$) and the parity sequences

$$P_1^n(D) = \sum_{j=o}^{\lambda-2} p_{1,j}^n D^j \quad \text{and} \quad P_2^n(D) = \sum_{j=o}^{\nu-2} p_{2,j}^n D^j \tag{15}$$

When advancing, along a branch pertaining to s_n, the decoder generates

$$x_{i,n} = s_n + p_{i,o}^n$$

$$P_i^{n+1}(D) = D^{-1}\left[P_i^n(D) + p_{i,o}^n\right] + s_n G_i^*(D) \tag{16}$$

for $i = 1,2$.. This is accomplished by two circuits similar to that of Figure 1a. It stores $x_{1,n},x_{2,n}$ and replaces $P_i^n(D)$ by $P_i^{n+1}(D)$ for $i = 1,2$. Finally, it replaces L_n by

$$L_{n+1} = L_n + \lambda(x_{1,n},y_{1,n}) + \lambda(x_{2,n},y_{2,n}) \tag{17}$$

When retreating, the decoder replaces L_n by

$$L_{n-1} = L_n - \lambda(x_{1,n-1},y_{1,n-1}) - \lambda(x_{2,n-1},y_{2,n-1}) \tag{18}$$

and $P_i^n(D)$ by

$$P_i^{n-1}(D) = D\left[P_i^n(D) + P_{k_i}^n G^*{}_i(D)\right] + x_{i,n-1} + P_{k_i}^n \tag{19}$$

where $k_1 = \lambda-2$, $k_2 = \nu-2$. Finally, it erases $x_{1,n-1}$ and $x_{2,n-1}$ from its storage. The operation (19) is accomplished by the circuit of Figure 2a. If the code is systematic then $P_1^n(D) = 0$ for all n and

$x_{1,n} = s_n$. If the code is non-systematic then $s_0, s_1, \ldots, s_{p-1}$ must somehow be recovered for the user. There are two ways to do this. Either at the end of the block of feeding $x_{1,\Gamma-2}, \ldots, x_{1,o}$ through the circuit of Figure 2a for $i = 1$, or by forward generation using the circuit of Figure 2b that corresponds to $1/G_1(D)$. This latter method has the advantage that information may be released to the user before the block is entirely decoded.

In stack decoding one does not recede, so there is no sense in storing $x_{1,i}$ and $x_{2,i}$. However, it is essential to conserve storage as far as possible. Therefore, a stack entry corresponding to a path of depth n ought to contain the sequences $P_1^n(D)$ and $P_2^n(D)$ as well as pointers to its past $p_{1,o}^{n-1}, p_{1,o}^{n-2}, \ldots, p_{1,o}^{\lambda-1}$. $P_1^{n+1}(D)$ and $P_2^{n+1}(D)$ are obtained by use of circuits like Figure 1, and the decoded sequence $s_{\Gamma-1}, \ldots, s_o$ is obtained at the end from a circuit of Figure 1b. Of course, if the code is systematic, then $P_1^n(D) = 0$ and one saves $s_{n-1}, s_{n-2}, \ldots, s_o$ instead of $p_{1,o}^{n-1}, \ldots, p_{1,o}^{\lambda-1}$.

II-A-2 Maintenance and Purging of the Stack and the Associated Map for the Stack Decoding Algorithm

In the Stack algorithm, the Stack entries must contain information about the corresponding path necessary to extend the latter and to determine the corresponding message sequence (in case the path is closer to the decoded one). In the preceding section we have shown that it is advantageous if each Stack entry contains (if R =1/2) the two parity sequences $P_1^n(D)$ and $P_2^n(D)$ and either the past parity sequence $\underset{\sim}{p}_1^n = p_{1,o}^{n-1}, p_{1,o}^{n-2}, \ldots, p_{1,o}^{\lambda-1}$ or the past information sequence $\underset{\sim}{s}^n = s_{n-1}, s_{n-2}, \ldots, s_o$ (the two are identical for systematic codes). We will deal here with $\underset{\sim}{s}^n$. Remarks about $\underset{\sim}{p}_1^n$ would be similar and they are made wherever necessary in Appendix 1.

Since $\underset{\sim}{s}^n$ is only needed at the end of and not during the decoding process, access to it need not be a lost one. Thus, as described in reference [1], the various $\underset{\sim}{s}^n$ sequences are specified in a linked map, and the appropriate one is linked to the Stack entry by a pointer. The map specification itself takes advantage of the tree structure of the code.

The map must contain at all times the specification of all paths corresponding to "live" entries in the stack. Since the stack is finite, it is purged according to the principle "least likelihood first." The map may contain some paths no longer in the stack, but efficient storage use requires that there be as few dead paths as possible. Hence the need for map purging. A report [1] by the author describes how map purging can be carried out in a manner directly dependent on stack purging, but the method requires establishment of counters for every live map branch whose content indicates the number of live paths that have that

branch in common. New map management strategies were developed that do not require any counters.

The first two strategies are for a map that specifies $\underset{\sim}{s}^n$ by linking positions of 1-branches to preceding 1-branch positions., E.g., the path 100110100 is given by the linked position arrangement $\rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow -(\nu-1)$ (ν is the code constraint length and all paths are linked to position $-(\nu-1)$ by convention). The purging principle of the first strategy is as follows: a branch can be eliminated from the map if its depth is t less than the depth of the path on top of the stack and if that path leads through that branch. Of course, it is understood that if the furthest depth of advance in the tree is I_{MAX} then all information digits up to depth $I_{MAX} - t$ have been definitely decided. The value of t must be chosen so that the probability of erroneous premature decision is sufficiently low.

It may also be desirable to make final decoding decisions according to a different than $I_{MAX} - t$ depth rule. For instance, let $L_1, L_2, \ldots, L_k$ be the cumulative likelihood values at depths $1, 2, \ldots, k$ of a path of depth k that is on top of the stack. Then one might decide all information digits up to depth m where

$$m \triangleq \max \left\{ \ell : L_k - L_\ell \overset{\Gamma}{\gtrless} T \right\}$$

and T is some suitable fixed threshold. The second strategy purges all map positions of depth m or less where the value of m is determined by any arbitrary rule (m is, of course, a non-decreasing function of time). This strategy does require the establishment of additional arrays in storage.

Finally, the third maintenance and purging strategy applies to maps whose paths are specified by sequences of information digits. The stack

has locations M1 containing $(\nu-1)+k$ binary digits (ν is the constraint length of the code and k is arbitrary), the right-most being the most recent one. It has a counter indicating the depth of the path and a pointer P1 to the location in the map that contains the preceeding path sequence of length k. The map has locations M2 of k digits, pointers MPP indicating the location of the preceeding path sequence, and pointers MPL linking all M2 locations that correspond to the same path depth. There are also pointers to the first and last map locations of any given live depth (a fixed number j of depths are live at any time) and a pointer to the first free (or replaceable) map location. If I_{MAX} is the depth of deepest penetration in the tree, then the purging strategy assumes that the map will contain no locations referring to depths preceding $I_{MAX} - (\nu - 1) - jk$.

The details of the three strategies are described in Appendix 1.

II-A-3 Multibranch Advance through the Tree of the Decoding Algorithm

Rate 1/2 binary codes have 2^{ℓ} branches leaving every node, each branch containing 2ℓ digits. In practice codes with $\ell = 1$ are used only, since an advance by one node involves finding the branch whose likelihood is m^{th} largest. The straight-forward way of doing this is to evaluate each of the 2^{ℓ} likelihoods and then order them. This is too large an undertaking. However, if the m^{th} branch could be looked up directly in a moderate size table, making $\ell \gtrsim 1$ would speed-up both Fano and stack decoding appreciably. Furthermore, simulation has shown that the needed stack size could also be substantially reduced.

In Appendix 2 we show how such tables can be constructed for binary input symmetric channels.. The table size grows as $K2^{2\ell}$. The coefficient K is larger for non-systematic codes for the BSC than for systematic ones, and an extension is more cumbersome. For a channel with 2 inputs and $2j$ outputs the table sizes are also of size $K2^{2\ell}$, but exact likelihood ordering is not possible. However, the approximation seems sufficiently close as to make the procedure a worthwhile one.

II-B Work on Bootstrap Hybrid Decoding

II-B-1 Simulations of Bootstrap Hybrid Decoding over the BSC

Appendix 2 contains a detailed description of three (progressively more sophisticated) bootstrap hybrid decoding algorithms as used over the BSC. The first is the rudimentary algorithm in which the binary channel state stream is modified only if some received stream is completely decoded. The second is the pull-up algorithm where the state stream is modified even after partial decoding of some stream. Specifically, if the furthest advance along a stream is to depth I_{MAX} then all digits up to depth $I_{MAX} - J$ are considered definitely decoded and the state stream is therefore modified up to depth $I_{MAX}-J$. Finally, the two-way algorithm is the pull-up algorithm with the added feature that attempts at stream decoding are made in both forward and backward directions. It is based on the observation of Dr. Dale Lumb that it is possible to decode a convolutional code backward as well as forward, provided each string of Γ information symbols is terminated by $\nu-1$ dummy bits known to the decoder. The bootstrap algorithm starts by decoding forward in the pull-up mode and continues to do so until a full decoding round takes place without completing any of the streams. In that case decoding in the backward direction starts and continues until another unsuccessful full decoding round occurs, in which case forward decoding resumes, etc. A stack of 1000 entries is used and if succeeding forward and backward rounds end without an advance of more than 20 branches on any stream in either direction, the stack is increased to 8000 entries for the next two rounds.

Table I contains a summary of three randomly selected decoding runs that use the rudimentary bootstrap hybrid decoding scheme of convolutional rate $R = .5$ over a BSC with crossover probability $p = .07$ ($R_{comp} = .4$). Stack decoding is utilized. We use $m = 10$ streams so the net rate is $R_{NET} = .45$. Other parameters of interest are block length $\Gamma = 1000$, block termination $t = 25$, and number of decoding steps allowed on a stream $M = 5000$. The printout indicates which of the $m = 10$ streams was worked on (JNOW), how many decoding steps were taken (N3), how deeply the decoder penetrated (IMAX) into the tree within the N3 steps taken, and how many undecoded streams were left (KLEFT). Finally, the speed factor (SF) is given for the entire block. SF is defined as the ratio of the total number of decoding steps taken to the number of information lists decoded. The Table shows quite clearly how fast the remaining streams can be decoded once the first three or four are known.

Table 2 shows decoding progress in a typical run of the pull-up algorithm over a BSC with crossover probability $p = .08$. A convolutional code of rate $R = 1/2$ was used and $m = 10$ streams formed a block. The maximum allocation $M = 5000$ and the stack had 700 entries. The parameters JNOW, N3, IMAX, KLEFT, SF, and KTRY have the same meaning as in Table 1. The value of JSTART indicates the depth of the node at which the decoding of the particular stream began. The definitely decoded back-up limit was $J = 200$. If in a decoding round no stream advanced by more than 20 levels beyond its previous maximal depth, M was temporarily increased to 20000 and the stack size to 8000 until such an advance took place. This phenomenon can be observed in row 20 of the Table. It becomes apparent for the present example that without

bootstrapping it would be completely impossible to decode 9 of the 10 received streams of this block as the older Falconer scheme would require. In fact, we were not able to decode the fifth stream without 27000 steps even when using information from the decoded streams 2 and 7 and the almost decoded stream 8! It seems fair to say that the Falconer scheme could decode at most three of the ten received streams and no more. Bootstrapping is no "endgame"--it does not complicate the decoding search and ought to be used right from the start.

Table 3 shows an example of two-way decoding over a BSC with crossover probability $p = .09$. The parameters JNOW, N3, IMAX, JSTART, KLEFT, SF and KTRY have the meaning given them in Table 2, except that when decoding is backward, nodes are numbered in reverse order so that forward node 1000 is backward node 1, etc. (this affects IMAX and JSTART). The parameters IFORW is 1 when forward decoding took place and is 2 otherwise. The parameter KROUND indicates how many streams were attempted in a given direction since the last successful decoding. When its value reaches that of KLEFT, decoding direction is reversed.

We have run all of our simulations using the stack decoding algorithm applied to transmission of data over a binary symmetric channel with crossover probability p. The systematic code of constraint length $\nu = 72$ whose taps in octal notation are 651102104421022041101101 (obtained by Costello [1969]) was used, the number of streams was $m = 10$ (this value was picked arbitrarily without any attempt at optimization) and there were always 1000 true information bits per information stream. [i.e. 9000 bits per block].

Our simulation results are summarized in Table 4 which gives certain parameters of interest that we now explain. For different crossover

probabilities we have used different bootstrapping algorithms. The crossover probability $p = .056$ was chosen because the corresponding channel has $R_{comp} = .45$ which is equal to the net rate of our scheme. Hence the dB gain over straight sequential decoding is 0 . Figure 3 is based on 2000 blocks of data and shows the distribution of computation per decoded information bit [speed factor] when the rudimentary algorithm is used. As is usual, an extension of a node by the decoder serves as a unit of computation, and the speed factor was obtained by simply dividing by 9000 the total number of computations necessary for decoding of a block (the rudimentary algorithm is a block scheme and it is not clear how to assign particular decoding steps to particular information bits). The startling result of this simulation is that if tail behavior of the distribution could be extrapolated as a straight line on the log-log plot (which is certainly O.K. in sequential decoding) then the asymptotic computational distribution would be

$$P\{SF > x\} \approx 1380\ x^{-12.8}$$

This would mean that a speed factor 5.17 would be needed only once in 10^6 blocks, and a speed factor of 8.92 only once in 10^9 blocks! However, a glance at Table 4 shows that the largest limiting exponent (derived according to the analysis of reference [1]) can only be 2.74 and we are at this time at a loss to explain this discrepancy. The most likely reason is insufficient statistics - 2000 sample points is not enough.*

*It is difficult to extend the sample size substantially. 2000 blocks involves 18×10^6 bits and our Fortran algorithm took 80 minutes of IBM 360-91 computer running time. A similar discrepancy between an observed and theoretical Pareto exponent was reported by Forney [2] who did high-rate simulations of sequential decoding on the Gaussian channel. In his case it turned out that a theoretical exponent of 0.087 was observed to have an experimental value in the range 0.38-0.41.

Under this hypothesis the time distribution will assume its final slope somewhere below the probability 10^{-3}. The intriguing point is that should this take place at a small enough probability then the practical exponent might still be 12.8! Another cause for the anomaly might be the various computation truncations inherent in our algorithm. We shall investigate further and report more completely at a later date.

In any case, if the observed behavior can be extrapolated even approximately then the bootstrapping algorithm may be used to great advantage even at rates equal to R_{comp} in order to stabilize the decoding effort and prevent block erasures due to buffer overflow. It is particularly interesting that in the 2000 blocks decoded, only one required more than 12 attempts at stream decoding (the minimum is 9). The capacity of this channel is $C = .69$, so $R/C = .731$, and we have entered this point as a circle into the plot of Figure 5.

We feel that about the noisiest BSC over which it is practicable to run the rudimentary algorithm with stream length $\Gamma = 1000$ bits is one whose crossover probability is $p = .07$. Figure 4 displays the corresponding computational distribution. Again, the apparent Pareto exponent of 2.66 is larger than the theoretical maximum of 2.2. The R/C parameter of this experiment is entered as a triangle in Figure 5.

As a next experiment we ran the pull-up algorithm over the BSC with crossover probability $p = .08$. We used a stack with 1000 entries and stopped computation on a stream either if it was decoded or if a stack overflow took place. We considered permanently decoded all but the last 2000 bits of the path that was in the stack immediately before it overflowed. This caused no errors in the 1000 blocks that we ran and successfully decoded. If a round was completed without advancing the

decoding of any of the remaining streams by more than 20 branches then the stack size was increased to 8000 for the next round. We did not obtain an experimental distribution, but only the average and maximal speed factors. The R/C parameter of this experiment is entered as a square in Figure 5.

The final entry in Table 4 involves a BSC with crossover probability $p = .09$ over which we ran a <u>two-way</u> algorithm.

Since two-way decoding uses more information than the one-way kind, the bounds of reference [1] are not applicable to the former. Nevertheless, the entry in the lower bound to Pareto exponent column of Table 4 is derived according to the corresponding formula of reference [1]. Again, our simulation only determined the average and the maximal speed factors based on a run of 500 blocks. The R/C parameter of this experiment is entered as a star in Figure 5.

TABLE I

Simulation Examples of Rudimentary Bootstrap Hybrid Decoding of $R_{NET} = .45$, $m = 10$ over BSC with $p = 0.07$

Block 1

JNOW	N3	IMAX	KLEFT
1	5000	473	10
2	5000	1008	10
3	4864	249	10
4	1948	1025	9
5	1534	1025	8
6	3655	1025	7
7	1320	1025	6
8	1849	1025	5
9	1495	1025	4
10	1178	1025	3
1	1350	1025	2
2	1079	1025	1

SF = 3.36

Block 2

JNOW	N3	IMAX	KLEFT
1	5000	842	10
2	5000	749	10
3	2610	1025	9
4	5000	1010	9
5	5000	929	9
6	3735	1025	8
7	2132	1025	7
8	5000	948	7
9	2553	1025	6
10	5000	552	6
1	1739	1025	5
2	1863	1025	4
4	1297	1025	3
5	1160	1025	2
8	1066	1025	1

SF = 5.34

TABLE I CONT'D

Block 3

JNOW	N3	IMAX	KLEFT
1	2524	1025	9
2	4377	278	9
3	5000	239	9
4	4880	1025	8
5	2288	1025	7
6	3275	1025	6
7	1659	1025	5
8	1246	1025	4
9	1320	1025	3
10	1926	1025	2
2	1074	1025	1

SF = 3.28

TABLE 2

A Simulation Example of Pull-up Bootstrap Hybrid Decoding of R_{NET} = .45, m = 10 over BSC with p = 0.08

JNOW	N3	IMAX	JSTART	KLEFT
1	2744	215	0	10
2	2963	1025	0	9
3	2891	219	0	9
4	1858	93	0	9
5	2314	141	0	9
6	2207	192	0	9
7	3447	1025	0	8
8	5000	944	0	8
9	2958	294	0	8
10	2353	339	0	8
1	2729	235	15	8
3	2143	212	19	8
4	3052	212	0	8
5	2329	146	0	8
6	2767	166	0	8
8	3301	944	744	8
9	2037	293	94	8
10	2468	341	139	8
1	2834	235	35	8
5	27062	800	0	8
6	3030	287	0	8
8	3301	944	744	8
9	2283	287	94	8
10	2422	341	141	8
1	5000	762	35	8
3	2421	1025	19	7
4	1322	1025	12	6
5	2671	716	600	6
6	2913	292	87	6
8	2799	944	744	6
9	5000	852	94	6
10	2040	1025	141	5
1	839	1025	562	4
5	774	1025	600	3
6	979	1025	92	2
8	302	1025	744	1

SF = 13.05

KTRY = 36

TABLE 3

A Simulation Example of Two-way Bootstrap Hybrid Decoding of $R_{NET} = 0.45$, $m = 10$ over BSC with $p = 0.09$

JNOW	N3	IMAX	JSTART	KLEFT	KROUND	IFORM
1	5910	272	0	10	1	1
2	5356	139	0	10	2	1
3	5731	354	0	10	3	1
4	7514	640	0	10	4	1
5	4262	182	0	10	5	1
6	5537	164	0	10	6	1
7	3770	164	0	10	7	1
8	6002	200	0	10	8	1
9	5819	351	0	10	9	1
10	8401	734	0	10	10	1
1	5695	443	0	10	1	2
2	6542	589	0	10	2	2
3	8395	307	0	10	3	2
4	3740	166	0	10	4	2
5	4103	136	0	10	5	2
6	4671	114	0	10	6	2
7	4329	277	0	10	7	2
8	6909	733	0	10	8	2
9	5013	157	0	10	9	2
10	5373	332	0	10	10	2
1	3650	262	72	10	1	1
2	3306	1071	0	9	0	1
3	4589	388	154	9	1	1
4	4149	651	440	9	2	1
5	5443	228	0	9	3	1
6	5440	254	0	9	4	1
7	10265	950	0	9	5	1
8	4095	224	0	9	6	1
9	5738	351	151	9	7	1
10	4480	722	534	9	8	1
1	5751	244	72	9	9	1
3	6377	309	107	9	1	2
4	8493	278	0	9	2	2
5	7566	715	0	9	3	2
6	4788	373	0	9	4	2
7	975	1071	77	8	0	2
8	4283	874	533	8	1	2
9	5202	443	0	8	2	2
10	3805	1071	132	7	0	2
1	4685	465	243	7	1	2
3	3510	1071	109	6	0	2
4	5007	443	78	6	1	2
5	3260	1071	515	5	0	2
6	1445	1071	173	4	0	2
8	438	1071	674	3	0	2
9	1230	1071	243	2	0	2
1	779	1071	265	1	0	2

SF = 25.75 KTRY = 47

TABLE 4

Summary of Simulation Parameters for

BSC, $R_{NET} = 0.45$, m = 10

Crossover probability	dB gain over R_{comp} for sequential decoding	R/C	Sequential decoding exponent for R = .45	Theoretical upper bound to exponent	Theoretical lower bound to exponent	Type of hybrid decoding algorithm	Experimental exponent	Average speed factor	Maximal speed factor	Number of blocks decoded
0.056	0.00	0.731	1.0	2.74	2.25	rudim	12.8	1.535	3.93	2000
0.07	0.54	0.788	0.75	2.2	1.5	rudim	2.66	4.23	16.3	500
0.08	0.97	0.837	0.55	1.9	1.2	pull up	-----	7.00	24.5	1000
0.09	1.36	0.887	0.41	1.6	0.81	two way	-----	22.00	100.0	500

II-B-2. Bounds on Computing Effort for Bootstrap Hybrid Decoding on Binary Input Channels

Let us generalize the encoding and decoding methods of Appendix 3 to channels symmetrical from the input that have two input symbols and an arbitrary number $b(\geq 3)$ of output symbols. The encoding is one involving m-1 information streams and an additional parity check stream. Suppose we receive the m streams and wish to decode the last of them (this happens to make notation convenient and is without loss of generality), $y_1(m)$, $y_2(m)$,... Since for every time interval i the receiver has at its disposal the vector

$$\underset{\sim}{y}_i(m) \triangleq (y_i(1), y_i(2), \ldots, y_i(m)) \tag{1}$$

the sequential decoder ought to calculate the likelihood function $\lambda_m(i)$ at depth i by the formula (capitals denote random variables)

$$\lambda_m(i) = \log \frac{P\left\{\underset{\sim}{Y}_i(m) = \underset{\sim}{y}_i(m) \;/\; x_i(m)\right\}}{P\left\{\underset{\sim}{Y}_i(m) = \underset{\sim}{y}_i(m)\right\}} - R \tag{2}$$

where the algebraic constraint $\sum_{j=1}^{m} x_i(j) = 0$ is assumed to hold and must be used when calculating the probabilities in the argument of the logarithm.

It is shown in reference [3] that for the j^{th} received stream the expression (2) can be simplified to have the form

$$\lambda_m(i) = 1 - R - \log\left[1 + Q(\underset{\sim}{y}_i(m))\right] + \log\left\{q(x_i(j)/y_i(j)) \left[1 + \frac{Q(\underset{\sim}{y}_i(m))}{2q(x_i(j)/y_i(j))-1}\right]\right\} \tag{3}$$

where $x_i(j) \in (0,1)$,

$$q(x|y) = \frac{w(y|x)}{w(y|o) + w(y|1)} \tag{4}$$

and

$$Q(\underset{\sim}{y}_i(m)) \triangleq \prod_{j=1}^{m} [q(0/y_i(j)) - q(1/y_i(j))] \tag{5}$$

The above formula suggests an efficient instrumentation for hybrid decoding of the class of channels considered. The state of the channel at the various time instants is given by the sequence $Q(\underset{\sim}{y}_1(m))$, $Q(\underset{\sim}{y}_2(m))$, $Q(\underset{\sim}{y}_3(m)),\ldots$. In fact, except for $Q(\underset{\sim}{y}_i(j))$, the formula (3) is a function of events $x_i(j)$ and $y_i(j)$ that themselves pertain to the j^{th} stream.

Thus, upon receiving the symbols that correspond to the m transmitted streams, the decoder will compute the channel state stream whose i^{th} entry Q_i will be the <u>number</u> $Q(\underset{\sim}{y}_i(m))$* (i.e. not a binary digit signifying the parity of the i^{th} position as before). Decoding will then proceed as outlined in Appendix 3, based on the likelihood function (3), until one of the streams, say the j_1^{th}, is decoded. The necessary recomputation of the channel state stream will simply consist of replacing the i^{th} entry Q_i by its new value $Q_i' = Q_i/[2q(x_i(j_1)/y_i(j_1))-1]$ where $x_i(j_1)$ is the decoder's estimate of the i^{th} transmitted digit of the j_1^{th} stream. Decoding of the remaining $m-1$ streams will then start from the beginning and will continue to use the likelihood (3) based on the new state stream values Q_i'. When a stream, say the j_2^{th},

*Since the number of possible values of Q_i is rather limited, the state stream would in practice contain only the address $A(Q_i)$ of a table entry containing the number Q_i. Or, even better, there would be a likelihood table whose entries would be formed from the value of the triplet $[x_i(j), y_i(j), A(Q_i)]$. The problem of limiting the size of such a table is discussed at the end of this section.

is decoded, the i^{th} state stream entries Q_i' will be replaced by entries $Q_i'' = Q_i'/[2q(x_i(j_2)/y_i(j_2)) - 1]$, etc., until just one stream remains undecoded. The latter's identity will be determined from the parity constraint.

As mentioned in footnote * there might arise a problem of storing the state stream entries Q_i. Let us consider the case where the output alphabet size b is even. Since the channel is symmetric from the input, every digit y can be represented by a pair (u,v) where u is binary, $v \in \{0,1,\ldots,(b/2)-1\}$ and

$$w(u=0,v/x) = w(u=1,v/x \oplus 1) \tag{6}$$

for all $x \in (0,1)$ and v. It follows then that

$$g(v) \stackrel{\Delta}{=} q(0/u = 0,v) - q(1/u = 0,v) =$$

$$- [q(0/u = 1,v) - q(1/u = 1,v)] \tag{7}$$

and therefore, letting

$$z = u_1 \oplus u_2 \oplus \ldots \oplus u_m \tag{8}$$

we get that

$$Q(\underset{\sim}{y}(m)) = (-1)^z \prod_{j=1}^{m} g(v_j) \tag{9}$$

Since

$$2q(x/u,v) - 1 = q(0/x \oplus u,v) - q(1/x \oplus u, v)$$

$$= (-1)^{x \oplus u} \cdot g(v)$$

then if $x_i(j_1)$ is the decoder's final estimate of the i^{th} transmitted digit on the j_1 stream, Q_i is to be replaced by its new value

$$Q_i' = (-1)^{x_i(j_1) + u_i(j_1)} Q_i/g(v_i) \tag{10}$$

If $n(v)$ denotes the number of v_i's whose value is v, then after $m-k$ streams have been decoded, Q_i will have the form

$$Q_i = (-1)^z \prod_{v=0}^{b/2} g(v)^{n(v)} \tag{11}$$

where $g(b/2) = 1$ and $n(b/2) = m-k$. Since

$$\sum_{v=0}^{b/2} n(v) = m$$

it follows that Q_i must have one of at most

$$2 \binom{\frac{b}{2} + m}{m}$$

values. Hence a complete likelihood table would be of size

$$2b \binom{\frac{b}{2} + m}{m} \tag{12}$$

The values of (12) for a two bit and three bit output quantization with $m = 10$ are 528 and 16016 respectively. The latter figure certainly seems excessive and yet three bit quantization is used quite frequently. One possible remedy is not to use all available information at the receiver. The simplest would be to use in the state stream only the points z defined in (8) and use the likelihood

$$\lambda_m(i) = \log \frac{w_m(u_i(j),\ v_i(j),\ z_i/x_i(j))}{w_m(u_i(j),\ v_i(j),\ z_i)} - R \qquad (13)$$

where

$$\begin{aligned} w_m(0,v,0/0) &= w_m(1,v,0/0) = w(0,v/0)\ q_{m-1}(0) \\ w_m(0,v,1/0) &= w_m(1,v,1/1) = w(0,v/0)\ q_{m-1}(1) \\ w_m(1,v,0/0) &= w_m(0,v,0/1) = w(1,v/0)\ q_{m-1}(1) \\ w_m(1,v,1/0) &= w_m(0,v,1/1) = w(1,v/0)\ q_{m-1}(0) \end{aligned} \qquad (14)$$

is defined as in (1), and

$$p = \sum_{v=0}^{b/2-1} w(1,v/0) \qquad (15)$$

Obviously, less severe restrictions on the information used are also possible. E.G., for the purposes of Q_i - computation one may wish to partition the v-alphabet into subsets and represent each subset by some new letter v'. The likelihood table size is then obtained by formula (12) into which ℓ, the size of the v' alphabet, has been substituted for $b/2$.

Let us note that the switch from likelihood (2) to (13) simply involves a switch between equivalent channels used by the receiver for decoding. The maximum information channel (using the Q-state stream) is based on transmission probability $P\{Y_i(m) = y_i(m)/x_i(m)\}$ while the binary state stream channel is based on $w_m(u_i(j), v_i(j), z_i/x_i(j))$. In general, let $w_k^*(\underset{\sim}{y}/x)$ denote the transmission probability of the equivalent channel used when decoding one of k undecoded streams, and define the function

$$E_k(\sigma) = (1+\sigma) - \log \sum_{\underset{\sim}{y}} \left[\sum_{x=0}^{1} w_k^*(\underset{\sim}{y}/x)^{\frac{1}{1+\sigma}} \right]^{1+\sigma} \qquad (16)$$

Thus, for the BSC $E_k(\sigma)$ is given by

$$E_k(\sigma) = \sigma - \log \left(\left\{ \left[(1-p) q_{k-1}(0) \right]^{\frac{1}{1+\sigma}} + \left[p\, q_{k-1}(1) \right]^{\frac{1}{1+\sigma}} \right\}^{1+\sigma} + \left\{ \left[(1-p) q_{k-1}(1) \right]^{\frac{1}{1+\sigma}} + \left[p\, q_{k-1}(0) \right]^{\frac{1}{1+\sigma}} \right\}^{1+\sigma} \right) \tag{17}$$

For the binary input, b-nary output symmetrical channel with a Q-type state stream it is

$$E_k(\sigma) = -\log \left[\sum_{y(k)} \left(\left\{ w(y(k)/o) \left[f_+(\underset{\sim}{y}(k-1)) + f_-(\underset{\sim}{y}(k-1)) \right] \right\}^{\frac{1}{1+\sigma}} + \left\{ w(y(k)/1) \left[f_+(\underset{\sim}{y}(k-1)) - f_-(\underset{\sim}{y}(k-1)) \right] \right\}^{\frac{1}{1+\sigma}} \right)^{1+\sigma} \right] + k+\sigma \tag{18}$$

where

$$f_+(y(k-1)) \triangleq \prod_{i=1}^{k-1} \left[w(y(i)/0) + w(y(i)/1) \right]$$

$$f_-(y(k-1)) \triangleq \prod_{i=1}^{k-1} \left[w(y(i)/0) - w(y(i)/1) \right] . \tag{19}$$

Finally, for the same channel, when only the parity is used in the state stream,

$$E_k(\sigma) = \sigma - \log \sum_{v} \left(\left\{ \left[w(0,v/0) q_{k-1}(0) \right]^{\frac{1}{1+\sigma}} + \left[w(1,v/0)\, q_{k-1}(1) \right]^{\frac{1}{1+\sigma}} \right\}^{1+\sigma} + \left\{ \left[w(0,v/0) q_{k-1}(1) \right]^{\frac{1}{1+\sigma}} + \left[w(1,v/0)\, q_{k-1}(1) \right]^{\frac{1}{1+\sigma}} \right\}^{1+\sigma} \right) \tag{20}$$

In reference [3] the following two theorems are proven:

Theorem 1

Let R be the convolutional coding rate used in each of m streams of a decoding block. The bootstrap hybrid decoding procedure based on $w_k(\underset{\sim}{y}/x)$ leads to a finite γ^{th} moment of computation per decoded digit provided

$$\min\ (\sigma(k(R)),\ \ (k(R)+1)\ \ \sigma(\infty))\ >\ \gamma\ . \tag{21}$$

is satisfied, where $k(R)$ is the unique integer such that

$$k(R)\ \ \sigma(\infty)\ <\ \sigma(k(R)) \tag{22}$$

$$\sigma(k(R)+1)\ \leq\ (k(R)+1)\ \ \sigma(\infty)\ .$$

The function $\sigma(k)$ is the unique solution of

$$R\ =\ \frac{E_k(\sigma)}{\sigma}\qquad \text{for}\qquad \frac{E_k(2)}{2}\ \leq\ R\ <\ C \tag{23a}$$

and

$$\sigma(k)\ =\ \frac{E_k(2)}{R}\qquad \text{for}\qquad 0\ <\ R\ <\ \frac{E_k(2)}{2}\ . \tag{23b}$$

The function $E_k(\sigma)$ is the concave, positive, increasing function of $\sigma > 0$ defined in (16).

Theorem 2

$\underset{\sim}{E}[N^{\gamma}]$ grows exponentially with block length Γ whenever

$$\min\ \{\sigma(2),\ k\sigma(k)\}\ <\ \gamma \tag{24}$$

where $\sigma(k)$ is the solution of

$$R\ =\ \frac{E_k(\sigma)}{\sigma}$$

In this theorem it is assumed that $R < C$, and that the convolutional code used is a good one in the sense that its associated probability of error is exponentially optimal.

Obviously, the net transmission rate (taking into account the loss due to the extra parity stream) is

$$R_{NET} = \frac{m-1}{m} R \quad . \tag{25}$$

Ordinarily, one would wish to transmit at a rate R_{NET} exceeding R_{comp} of the underlying channel so that $\sigma(\infty) < 1$. Define $R^L_{BOOT}(\gamma)$ to be the supremum of rates for which (21) is satisfied. Then we can say that

<u>the γ^{th} computational moment will be bounded for the bootstrap hybrid scheme using m streams provided the net rate satisfies</u>

$$R_{NET} < \frac{m-1}{m} R^L_{BOOT}(\gamma) \quad . \tag{26}$$

Define $R^U_{BOOT}(\gamma)$ as the greatest lower bound on rates for which (24) is satisfied. Then

<u>the γ^{th} computational moment will grow exponentially with block length Γ if</u>

$$R_{NET} > \frac{m-1}{m} R^U_{BOOT}(\gamma) \tag{27}$$

In reference [3] we show that $R^L_{BOOT}(\gamma)$ and $R^U_{BOOT}(\gamma)$ can be computed by the formulas

$$R^L_{BOOT}(\gamma) = \min_{k \geq 2} \left\{ \max \left[\frac{1}{\gamma} E_k(\gamma), \frac{k}{\gamma} E_\infty\left(\frac{\gamma}{k}\right) \right] \right\} \tag{28}$$

and

$$R^U_{BOOT}(\gamma) = \min \left\{ \frac{1}{\gamma} E_2(\gamma), \min_{k \geq 3} \left[\frac{k}{\gamma} E_k\left(\frac{\gamma}{k}\right) \right] \right\} \tag{29}$$

When evaluating $R^{L}_{BOOT}(\gamma)$ one computes the differences $\frac{1}{\gamma} E_k(\gamma) - \frac{k}{\gamma} E_\infty(\frac{\gamma}{k})$ for $k = 2,3,\ldots$ until their value becomes negative. If this takes place for k^+ then

$$R^{L}_{BOOT}(\gamma) = \min[\frac{1}{\gamma} E_{k^+-1}(\gamma) , \frac{k^+}{\gamma} E_\infty(\frac{\gamma}{k^+})] \tag{30}$$

It can be shown that the function $\frac{k}{\gamma} E_k(\frac{\gamma}{k})$, $k = 3,4,\ldots$ has at most one local minimum and no local maxima. Therefore when trying to evaluate $R^{U}_{BOOT}(\gamma)$ one computes the differences

$$\frac{k}{\gamma} E_k(\frac{\gamma}{k}) - \frac{k+1}{\gamma} E_{k+1}(\frac{\gamma}{k+1})$$

for $k = 3,4,\ldots$ until their value becomes negative. If this takes place for k^+, then

$$R^{U}_{BOOT}(\gamma) = \min [\frac{1}{\gamma} E_2(\gamma), \frac{k^+}{\gamma} E_{k^+}(\frac{\gamma}{k^+})] \tag{31}$$

The qualitative improvement achieved by bootstrap hybrid decoding over straight sequential decoding for the BSC can be estimated from a comparison of the curve R_{comp}/C vs. p (C is the capacity) with the curve $R^{L}_{BOOT}(1)/C$ vs. p. Figure 5 shows the corresponding plots together with those of $R_{FAL}(1)/C$ vs. p and $R^{U}_{BOOT}(1)/C$ vs. p. The quantity $R_{FAL}(1)$ is the rate above which the Falconer [4] scheme has an unbounded first computation moment. None of the latter three curves takes account of the algebraic degradation factor $\frac{m-1}{m}$ (see (26)) which must be used when any particular hybrid set-up is compared with straight sequential decoding.

Figure 6 shows the curves R_{comp}, $R^{L}_{BOOT}(1)$, $R^{U}_{BOOT}(1)$, R_{FAL}, and C plotted against the signal-to-noise ratio (in dB) per bit transmitted through a hard-quantized gaussian channel with binary inputs. It can be

seen that using convolutional codes of rate 1/2 a hybrid scheme with $m = 10$ streams will perform satisfactorily with an SNR per information bit that is at least 1.47 db smaller than the SNR needed for straight sequential decoding. Figure 7 shows the first four curves normalized by the fifth (capacity C). Finally, in Figure 8 we plot the values of γ vs. SNR per transmitted list that are solutions to equations $R^{L}_{BOOT}(\gamma) = 1/2$ (γ_{lower}) and $R^{U}_{BOOT}(\gamma) = 1/2$ (γ_{upper}) for the BSC obtained from a Gaussian additive noise channel. For comparison we also plot the Pareto exponent σ that corresponds to straight sequential decoding. In this connection the reader should recall the simulation results of the preceding section that seemed to indicate that the "practical" Pareto exponent is higher than the limiting theoretical one of Figure 8.

We have also evaluated theoretical performance curves for the binary input Gaussian channel with octal and quarternary quantization. To make comparison easy, Figures 10 through 15 are all drawn to the same scale. The quantization levels used throughout are the ones maximizing R_{comp}, as obtained by Lumb.[5] Therefore slight improvements might be possible in the C, $R^{L}_{BOOT}(1)$, and $R^{U}_{BOOT}(1)$ curves, if the optimization were to be carried out with respect to those parameters. Figure 10 shows the relationship between capacities and R_{comp}'s for binary, quarternary, and octal quantizations. Figure 9 gives the ratios R_{comp}/C for these quantizations which show the margin of possible improvement attainable through more sophisticated methods ofwhich bootstrap hybrid decoding is an example. We see that the margin decreases as the number of quantization levels increases.

Figure 11 contains plots of $R^{L}_{BOOT}(1)$ vs. SNR per transmitted digits for the three kinds of quantization when maximal information is used to form the state stream. Figure 12 provides the same curves when

the state stream is binary instead (i.e. when the likelihood is given by (13)). The next two figures show clearly that the degradation in performance is only a very slight one and it might well be worth that price to obtain the attendant reduction in decoder complexity. Figure 13 compares C and R_{comp} for the quarternary channel with $R^{L}_{BOOT}(1)$ curves for the binary and full channel state. Figure 14 does the same things for the octal channel. However, it turns out that in this case the difference between full channel state performance and a "quarternary" one is in the third significant digit and thus the latter curve cannot be entered separately into the graph. A "quarternary" state is one that would result if the 8 channel outputs were optimally partitioned into 4 classes and membership in the latter was used to determine the Q-type channel state. Finally, Figure 15 is a plot of $R^{U}_{BOOT}(1)$ vs. SNR per transmitted bit for binary, quarternary and octal channel quantizations when a binary channel state is used. The quarternary and octal curves are so close to capacity that it would be impossible to draw the $R^{U}_{BOOT}(1)$ curves for full information channel states.

As the last comment we would like to caution once more that none of the $R^{L}_{BOOT}(\gamma)$ or $R^{U}_{BOOT}(\gamma)$ curves involve the algebraic loss factor $\frac{m-1}{m}$ that must be used for fair comparison with non-hybrid schemes.

II-B-3. A Bound on a Computational Parameter of Bootstrap Hybrid Decoding

Let a bootstrap hybrid scheme involve transmission of m streams, m-1 carrying information. Let the decoding be of the rudimentary kind: one either succeeds in decoding a stream entirely, in which case the state information is adjusted and decoding of the next stream is attempted, or one does not succeed in decoding a stream in which case one passes to the next stream without having made any state adjustment. Decoding of any undecoded stream always starts from the first digit, regardless of whether previous decoding attempts at that stream have been made. Let us next define $N_i(K)$ to be the number of decoding steps in the first incorrect subset of the i^{th} among the K streams that have been left undecoded (i.e. $M \geq K$ streams were received, M-K were decoded by the hybrid method, and K streams--probably the most difficult ones--are still to be decoded). We suggest that a very good measure of computational complexity is the parameter

$$\underset{\sim}{E}\left[\max_{2 \leq K \leq M} \min_{1 \leq i \leq K} N_i(K) \right]$$

which may be interpreted as the expected maximum number of decoding steps that <u>need</u> be done in the course of decoding of the entire hybrid block in any first incorrect subset.

In Appendix 4 we find the rate below which the above quantity is bounded by a constant. The derivation is applicable to <u>all</u> channels symmetrical from the input (included in this class are all discrete channels derived through quantization of Gaussian additive noise channels). In the next reporting period we will evaluate these limiting rates for some channels of interest.

II-C. Development of Good Convolutional Codes

A binary, rate $R = 1/n$ convolutional code of constraint length ν can be specified by n generators

$$G^{(j)}(D) = g_0^{(j)} + g_1^{(j)}D + g_2^{(j)}D^2 + \ldots + g_{\nu-1}^{(j)} D^{\nu-1}, \quad j = 1,2,\ldots n \tag{1}$$

It is assumed that for at least one value of j_1 and j_2, $g_0^{(j_1)} = 1$ and $g_{\nu-1}^{(j_2)} = 1$, $j_1, j_2 \in \{1,2,\ldots,n\}$.

Every input sequence $i_0, i_1, \ldots, i_k$ can be represented by its D-transform polynomial

$$I(D) = i_0 + i_1 D \ldots + i_k D^k \quad . \tag{2}$$

If by convention, $i_t = 0$ for $t \geq K$, then the encoder outputs for such an input are the sequences

$$X^{(j)}(D) = G^{(j)}(D) \cdot I(D) = x_0^{(j)} + x_1^{(j)}D + \ldots x_{\nu+k-1}^{(j)} D^{\nu+k-1}, \quad j = 1,2,\ldots,n \tag{3}$$

where

$$x_t^{(j)} = i_t \cdot g_0^{(j)} \oplus i_{t-1}g_1^{(j)} \oplus \ldots \oplus i_{t-\nu+1}g_{\nu-1}^{(j)} = \bigotimes_{m=0}^{\nu-1} i_{t-m}g_m^{(j)} \quad [1] \tag{4}$$

A convolutional code is called <u>systematic</u> if $G^{(1)}(D) = 1$. In that case $X^{(1)}(D) = I(D)$ which is desirable for some applications.

---

[1] $\bigotimes$ denotes summation over GF(2)

$\sum$ denotes summation over the integers.

The output sequences produced by the encoder are conveniently represented by the state space 'trellis' diagram of the code given in Figure 16. The state $S(t)$ of the encoder at time t is determined by the $(\nu-1)$ preceding information digits;

$$S(t) = (i_{t-1}, i_{t-2}, \ldots, i_{t-\nu+1}) \tag{5}$$

where $i_t \triangleq 0$ for $t < 0$ and $t > k$. There are $2^{\nu-1}$ different states for an encoder of constraint length ν. For each state $S(t)$ there are two possible values of $S(t+1)$, depending on whether $i_t = 0$ or 1. The state space diagram shows the possible transitions for $t = 0,1,2,\ldots$. The branches in the diagram are labelled with the outputs corresponding to the transitions. The trellis of Figure 16 corresponds to the rate 1/2 code $G^{(1)}(D) = 1+D^2$, $G^{(2)}(D) = 1 + D + D^2$. Since $g_0^{(j_1)} = 1$ for at least one $j_1 \in 1,2,\ldots,n$, the two branches diverging from a state cannot be identical. Similarly, since $g_{\nu-1}^{(j_2)} = 1$ for at least one $j_2 \in 1,2,\ldots,n$, the two branches converging into a state cannot be identical.

The coder is initially started in state $S(0) = \underline{0} = (0,0,\ldots,0)$. For every input polynomial $I(D)$, there is a series of state transitions $\underline{0} = S(0) \longrightarrow S(1) \longrightarrow S(2) \longrightarrow \ldots \; S(\nu+k-1) \longrightarrow S(\nu+k) = \underline{0}$. Tracing the path corresponding to this series of transitions through the trellis diagram determines the output sequence corresponding to $I(D)$.

Let $\underline{0}^*$ denote the path $\underline{0} \longrightarrow \underline{0} \longrightarrow \underline{0} \; \ldots \longrightarrow \underline{0} \longrightarrow$ i.e. the path corresponding to an input of all zeros. Massey and Sain [6] have called a code <u>catastrophic</u> if there is an infinite path through the trellis that has no branch in common with the $\underline{0}^*$ path and whose Hamming weight is finite. The reason for this nomenclature is that in such a code a finite number of transmission errors may cause an infinite number of

errors in the decoded information sequence $I(D)$. Massey and Sain [6] have shown that a rate $1/n$ code is catastrophic if and only if

$$g.c.d.\left\{G^{1}(D),\ G^{2}(D),\ldots,G^{n}(D)\right\} \neq D^{r} \tag{6}$$

for some non-negative integer, r.

Let $X_t = [x_t^{(1)},\ x_t^{(2)},\ldots,x_t^{(n)}]$ denote the block of n output symbols at time t . The minimum distance of the code generated by $G^{(j)}(D)$, $j = 1,2,\ldots,n$ is

$$d_m(G^{(1)},\ldots,G^{(n)}) = \min_{\substack{I(D)\\ i_0=1}} \sum_{t=0}^{\nu-1} \omega_M(x_t) = \min_{\substack{I(D)\\ i_0=1}} \sum_{j=1}^{n} \omega_M(X^{(j)}(D) \bmod D^{\nu}), \tag{7}$$

where ω_M is the usual Hamming weight operator. In the trellis diagram this corresponds to the weight of the minimum weight path of exactly ν branches which diverges from the state $\underline{0}$ at $t = 0$. Bussgang [7] , Lin and Lync [8] , and Costello [9] have explored methods for constructing codes with large minimum distance.

The free distance of the code is

$$d_t(G^{(1)},\ldots,G^{(n)}) = \min_{\substack{I(D)\\ i_0=1}} \sum_{t=0}^{\infty} \omega_H(x_t) = \min_{\substack{I(D)\\ i_0=1}} \sum_{j=1}^{n} \omega_H(X^{(j)}(D)) \tag{8}$$

In the trellis diagram this corresponds to the weight of the minimum weight path of arbitrary length that diverges from the state $\underline{0}$ at $t = 0$ and reconverges to the state $\underline{0}$ at some later time. For the binary symmetric channel, maximum likelihood decoding corresponds to a search for that trellis path whose Hamming distance from the received sequence is minimal. Since convolutional codes are linear, free

distance is a good indication of maximum likelihood decoding strength of the code at least for low crossover probabilities. Minimum distance is in the same way important for feedback decoding of convolutional codes. Moreover, the computational effort in sequential decoding seems strongly influenced by d_m .

We have derived the following upper bound on d_t[10]

Theorem 1

For all rate $1/n$ convolutional codes of constraint length ν , the free distance is upper bounded by

$$d_f \leq \frac{n}{2} (\nu + [\log_2 \nu] + 1)$$

The evaluation of d_f of an arbitrary code is quite complicated, because one may have to search very deep into the coding tree to determine what d_f is . Although it is conjectured that the degree of the information sequence $I(D)$ that achieves d_f is only of the order of $\nu \log \nu$, the best general bound on that degree for rate $1/2$ codes is [10] $(\nu + \log \nu)(\nu-1) + 1$.

For the class of complementary codes that bound can be lowered to[10] $\nu(\nu-3)$, but what is more important, a very efficient search procedure determining d_f exists that allows early identification of $I(D)$ sequences that cannot possibly achieve d_f . Moreover, the d_f values of the best complementary codes are excellent and seen to grow as ν .

A rate $1/2$ code is a complementary code if and only if

$$g_0^{(1)} = g_0^{(2)} = g_{\nu-1}^{(1)} = g_{\nu-1}^{(1)} = 1 \qquad (9a)$$

and

$$g_m^{(2)} = g_m^{(1)} \oplus 1 \quad \text{for} \quad 1 \leq m \leq \nu-2 \qquad (9b)$$

The generators may therefore be written as

$$G^{(1)}(D) = 1 + g_1 D + g_2 D^2 + \ldots g_{\nu-2} D^{\nu-2} + D^{\nu-1}$$

$$G^{(2)}(D) = 1 + \bar{g}_1 D + \bar{g}_2 D^2 + \ldots \bar{g}_{\nu-2} D^{\nu-2} + D^{\nu-2}$$

where $\bar{g}_i$ is the binary complement of g_i i.e. $\bar{g}_i = g_i \oplus 1$

$$G^{(2)}(D) = G^{(1)}(D) + D + D^2 + \ldots + D^{\nu-2}$$

Following Massey, we can use this relation between $G^{(1)}(D)$ and $G^{2}(D)$ to reduce the number of adders needed to implement the encoder. If the indices i of $G^{(i)}(D)$ are selected so that

$$\sum_{i=1}^{\nu-2} w_H(g_i) \leq \sum_{i=1}^{\nu-2} w_H(\bar{g}_i)$$

then the encoding circuit is that of Figure 17.

As mentioned the structure of complementary codes allows construction of an efficient algorithm based on the stack decoding principle that determines d_f.

The stack is arranged according to the values of a lower bound $W(t)$ on the weights of all possible codewords corresponding to extensions of some given input sequence $I(D)$ of length t. The top of the stack is allocated to the codeword of lowest weight. Since it turns out that only sequences $I(D)$ of even weight can achieve d_f, the search considers inputs

$$P(D) = \frac{I(D)}{1+D} \tag{10}$$

to the convolutional code

$$G^{(1)}(D) = (1+D)\, G^{(1)}(D)$$

$$G^{(2)}(D) = (1+D)\, G^{(2)}(D) \qquad (11)$$

Each entry in the stack contains the following information:

a) $U(t) = (p_t, p_{t-;}, \ldots, p_{t-\nu+1})$, the current contents of the encoder

b) $W(t)$, the lower bound on the weight of codewords corresponding to extensions of $P(D)$ considered.

c) $C^P(t)$, a count of the length of the last run of zeros in $P(D) = I(D)/(1+D)$ ($P(D)$ can be discarded if $C^P(t) \geq \nu-2$)

d) $C^V(t)$, a count of the length of the last run of zeros in $v_o = v_1 D + \ldots + v_t D^t$ where $V(D) = 1+D^{\nu-2}/1+D\ I(D)$ ($P(D)$ can be discarded if $C^V(t) \geq \nu$) .

The following is then the algorithm

I. Initialization:

The stack contains one entry

$$U(0) = (1,0,\ldots,0),\ W(0) = 6\ ,\ C^P(0) = 0,\ C^V(0) = 0$$

II. Regular operation

1) If stack is empty, go to 17, else continue

2) Eliminate top entry of stack, $u(t)$, $w(t)$, $C^P(t)$, $C_v(t)$. If $C^P(t) = \nu-2$ go to 16.

3) $U(t+1) \longleftarrow (0, p_t, p_{t-1}, \ldots, p_{t-\nu+2})$, [Zero extn.]

4) If $p_{t-\nu+3} = 0$, $C^V(t+1) \longleftarrow C^V(t) + 1$, else $C^V(t+1) = 0$. If $C^V(t+1) = \nu$ go to 9 , else continue.

5) $C^P(t+1) \longleftarrow C^P(t) + 1$

6) $W(t+1) = W(t) + \omega_H(X^{(1)}_{t+1}, X^{(2)}_{t+1}) - \omega_H(p_t \oplus p_{t-\nu+2})$ where $(X^{(1)}_{t+1}, X^{(2)}_{t+1})$ is the output of the encoder.

7) If $W(t+1) \geq \nu + 2$, go to 9, else continue.

8) Insert entry $U(t+1)$, $W(t+1)$, $C^P(t+1)$, $C^V(t+1)$ in stack according to value of $W(t+1)$

9) $U(t+1) \longleftarrow (1, p_t, p_{t-1}, \ldots, p_{t-\nu+2})$ [one's extn.]

10) If $p_{t-\nu+3} = 1$, $C^V(t+1) \longleftarrow C^V(t) + 1$, else $C^V(t+1) = 0$
If $C^V(t+1) = \nu$ go to 1, else continue.

11) $C^P(t+1) = 0$

12) $W(t+1) \longleftarrow W(t) + \omega_H(\bar{X}^{(1)}_{t+1}, \bar{X}^{(2)}_{t+1}) + 2\,\omega_H(1 \oplus p_{t-\nu+2}) - \omega_H(p_t \oplus p_{t-\nu+2})$

13) If $\omega(t+1) \geq \nu+2$, go to 1, else continue

14) Insert $U(t+1)$, $\omega(t+1)$, $C^P(t+1)$, $C^V(t+1)$ in stack according to value of $\omega(t+1)$

15) Go to 1

16) $d_f = W(t)$ Stop.

17) $d_f = \nu+2$ Stop.

The free distance achieved by the complementary codes given in Table I is far in excess of any other known rate 1/2 codes. Figure 18 shows a comparison of the free distance of complementary codes with various bounds. It is seen that the codes come quite close to achieving the upper bound of Theorem 1. Neumann [11] has obtained a lower bound for free distance, but his bound is weaker than the usual Gilbert Bound for short constraint lengths. It is seen that the complementary codes are far better than the Gilbert bound, which is of course a lower bound on d_f as well as d_m. Figure 16 also contains the Costello lower bound for time-varying codes. It should be pointed out that the Costello bound is asymptotic and does not necessarily apply at short constraint lengths.

Figure 19 shows a comparison of the free distance of complementary codes with some other known codes. Costello [9], has devised two algorithms A6 (systematic) and A9 (nonsystematic) to construct codes with large free distance. It is seen that the complementary codes do far better than either of these codes. Also included in the comparison is the Lin-type code [8]. Figure 20 gives a comparison of the number of steps taken by the usual stack algorithm (i.e. one that would examine inputs $I(D)$ to $G^{(1)}(D)$ and $G^{(2)}(D)$ and would have only the structure properties of general convolutional codes) with the steps taken by the special algorithm for complementary codes. The comparison is made for the codes in Table II. It is evident that the special algorithm provides a tremendous advantage in computing the free distance of these codes.

Figure 21 shows that the minimum distance of the complementary codes always exceeds the Gilbert bound. At most constraint lengths the minimum distance equals the minimum distance of the Lin-Lyne code.

Some complementary codes were used in simulation studies for sequential and maximum likelihood decoding on a binary symmetric channel. The performance of these codes was consistently better than all other known codes [12].

The motivation for this work was to look for methods of constructing convolutional codes with large free distance. The results are partially successful since we found a good class of rate 1/2 codes whose free distance exceeds the free distance of any other known codes. However such codes were found only for $\nu \leq 24$ and there is no evidence to show whether good codes do or do not exist for longer constraint lengths. The major problem in searching for long codes is that the amount of computation needed to calculate the free distance grows at least exponentially.

We were able to utilize the special properties of complementary codes to cut down on the amount of computation.

Unfortunately there does not appear to be any simple way to generalize these codes to rates other than 1/2 .

ν	Gen. (octal)	d_{free}	d_{min}	wt.
3	5	5	3	2
4	13	6	3	3
5	31	7	4	3
6	61	8	4	3
7	121	9	5	3
8	211	10	5	4
9	503	11	6	4
10	1065	12	6	5
11	2415	13	7	5
12	5121	14	7	5
13	12043	15	7	5
14	24421	16	8	5
15	51303	17	7	7
16	120643	18	8	7
17	352411	18	9	8
18	425551	20	8	9
19	1411041	20	9	6
20	2734605	20	10	11
21	5011303	22	9	8
22	11047441	22	10	9
23	22517023	24	10	11
24	51202215	24	10	9

Table I. R = 1/2 Complementary Codes

II-D. Application of Bootstrapping to Maximum Likelihood Decoding of Convolutional Codes

We are trying to see if the basic idea of bootstrap hybrid sequential decoding can also be helpful to the Viterbi decoder. It will hopefully reduce the decoding complexity (that grows as $2^{\nu-1}$ in the Viterbi algorithm) for a given probability of error and transmission rate D.

We have completed a Fortran program whose basic idea is as follows: There are m-1 convolutionally encoded information streams and their exclusive-or sum forms the m^{th} parity stream (the BSC is implied). After reception the channel state stream is found in the usual way. Viterbi decoding of the first stream is undertaken whose likelihood values are based on the state information. The likelihood function of the decoded path is then examined and with its help reliable subintervals of the path are determined (e.g. a subsequence of the decoded sequence is considered reliable if it corresponds to a consistently rising likelihood). These are substituted for the corresponding portions of the received sequence and the state sequence is accordingly recomputed. The second stream is then decoded and its reliable subintervals determined. The transmitted digits falling within these subintervals replace the received digits and the state sequence is again adjusted. This work continues in a round robin fashion as long as re-decoding of received streams results in an enlargement of the reliable subintervals. When no such enlargement occurs for any of the m streams computation stops and the paths decoded last are supplied to the user.

The main problem in running this algorithm is the finding of criteria that could be used to determine the reliable subintervals. We

have written a program that collects statistics on the behavior of the likelihood function of decoded sequences when it corresponds to correct and incorrect information supplied to the user. The criteria will of course be more stringent the smaller the code constraint length and the larger the channel error probability. We hope to report some initial results soon.

References for Part II

1. F. Jelinek and J. Cocke: "Adaptive Hybrid Sequential Decoding" IBM Research Report, RC 2593, 1969.

2. G.D. Forney: A Study of Hybrid Sequential and Algebraic Coding Techniques, Final Report to NASA Ames Research Center by Codex Corporation, 1969.

3. F. Jelinek and J. Cocke: "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels," Information and Control, to be published.

4. D.D. Falconer: "A Hybrid Coding Scheme for Discrete Memoryless Channels," BSTJ, 48, No. 3, 1969.

5. D.R. Lumb: Notes for a Lecture on Coding for the Deep Space Telemetry Channel, UCLA Short Course in Recent Advances in Space Communications, July 1968.

6. J.L. Massey and M.K. Sain, "Inverses of Linear Sequential Circuits," IEEE Trans. on Computers, CT-17, No. 2, April 1968.

7. J.J. Bussgang, "Some Properties of Binary Convolutional Code Generators," IEEE, IT-11, Pg. 90-100, January 1965.

8. S. Lin and H. Lyne, Some Results on Binary Convolutional Code Generators," IEEE, IT-13, Pgs. 134-139, January 1967.

9. D.J. Costello, "A Construction Technique for Random Error Correcting Convolutional Codes," IEEE, IT-15, Pg. 631 ff. September 1969.

10. L.R. Bahl and F. Jelinek: "Free Distance Structure of Convolutional Codes and Complementary Rate 1/2 Codes," IEEE Transactions on Information Theory, submitted.

11. Bernd Neumann: Distance Properties of Convolutional Codes, M.S. Thesis, Massachusetts Institute of Technology, August 1968.

12. F. Jelinek and L.R. Bahl: "Maximum Likelihood and Sequential Decoding of Short Constraint Length Convolutional Codes," Proc. 7th Allerton Conference, pp. 130-139, 1969.

Captions for Figures of PartIII

Fig. 1a: Encoding circuit of a single convolutional generator.

Fig. 1b: Circuit that recovers information digits $s_{n-1}, s_{n-2}, \ldots, s_o$ from parity polynomial $P^n(D)$ and parity digits p_o^{n-1}, $p_o^{n-2}, \ldots, p_o^{\nu-1}$

Fig. 2a: Circuit that obtains $P_i^{n-1}(D)$ from $P_i^n(D)$ and $x_{i,n-1}$. It can also be used to recover the information digits $s_{n-1}, \ldots, s_o$ from $P_i^n(D)$ and $x_{i,n-1}, x_{i,n-2}, \ldots, x_{i,0}$.

Fig. 2b: Feedback circuit that obtains $s_0, \ldots, s_{n-1}$, from $x_{1,0}, \ldots, x_{1,\Gamma-1}$. Initial contents of the shift register are 0's.

Fig. 3: Empirical distribution of the speed factor necessary for rudimentary bootstrap hybrid decoding of $R_{NET} = 0.45$, $m = 10$ over a BSC with $p = 0.056$.

Fig. 4: Empirical distribution of the speed factor necessary for rudimentary bootstrap hybrid decoding of $R_{NET} = 0.045$, $m = 10$ over a BSC with $p = 0.07$.

Fig. 5: Comparison of performance characteristics of sequential decoding, Falconer's hybrid decoding, and bootstrap hybrid decoding over the BSC. The experimental points denote simulations at R = 0.5 referred to in Table 4.

Fig. 6: Comparison of performance characteristics of sequential decoding, Falconer's hybrid decoding, and bootstrap hybrid decoding with the capacity of a Gaussian channel with binary inputs and outputs.

Fig. 7: Plots of R_{comp}/C, R_{FAL}/C, $R_{BOOT}^L(1)/C$, and $R_{BOOT}^U(1)/C$ as a function of SNR per transmitted digit in dB's for the binary quantized gaussian channel with binary inputs.

Fig. 8: Upper and lower bounds to the Pareto exponent γ for hybrid decoding as a function of SNR per transmitted digit (dB) when the convolutional rate $R = 1/2$. The sequential decoding Pareto exponent σ is provided for comparison.

Fig. 9: R_{comp}/C vs. SNR per transmitted digit (dB) for binary, quarternary and octal optimal quantization of the Gaussian channel with binary inputs.

Fig.10: C and R_{comp} vs. SNR per transmitted digit (dB) for binary, quarternary and octal optimal quantization of the Gaussian channel with binary inputs.

Fig.11: $R^{L}_{BOOT}(1)$ vs. SNR per transmitted digit (dB) for binary, quarternary and octal quantization when state stream contains maximal information.

Fig. 12: The parameter of Fig. 11 when state stream is binary.

Fig.13: C, R_{comp}, and $R^{L}_{BOOT}(i)$ for binary and full information state stream for the quarternary output channel as a function of SNR per transmitted digit (dB).

Fig.14: C, R_{comp}, and $R^{L}_{BOOT}(1)$ for binary and full information state stream for the octal output channel as a function of SNR per transmitted digit (dB).

Fig.15: $R^{U}_{BOOT}(1)$ vs. SNR per transmitted digit (dB) curves for binary, quarternary, and octal output channel with binary inputs when the channel state stream is binary.

Fig.16: The trellis state diagram for the code $G^{(1)}(D) = 1+D+D^2$, $G^{2}(D) = 1+D^2$.

Fig.17: A simplified encoding circuit for complementary rate 1/2 codes.

Fig.18: Free distance of best complementary codes compared to the best available bounds.

Fig.19: Free distance of complementary and other best codes.

Fig.20: Computational effort necessary to determine free distance of an ordinary stack algorithm and of the special algorithm utilizing the structure of complementary codes.

Fig.21: Minimum distance of the highest free distance complementary codes compared to the Gilbert bound and to the minimum distance of the best available codes.

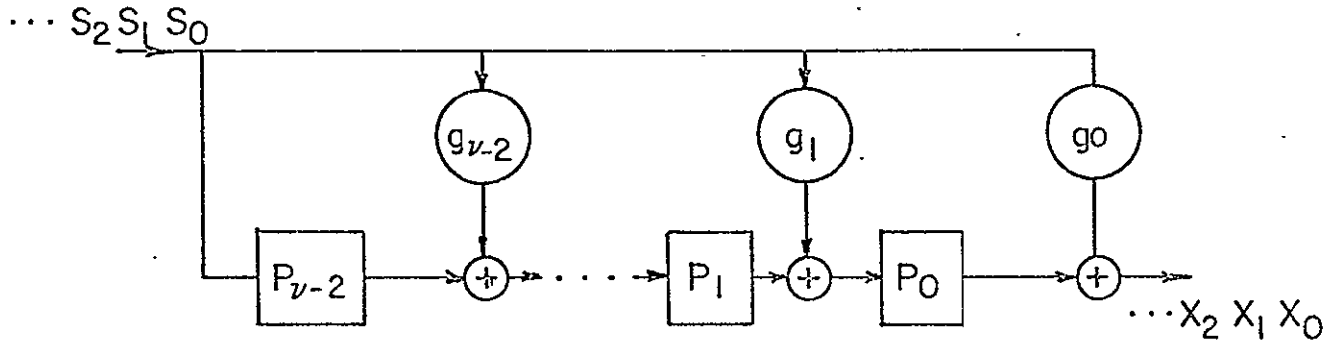


Figure 1a: Encoding circuit of a single convolutional generator.

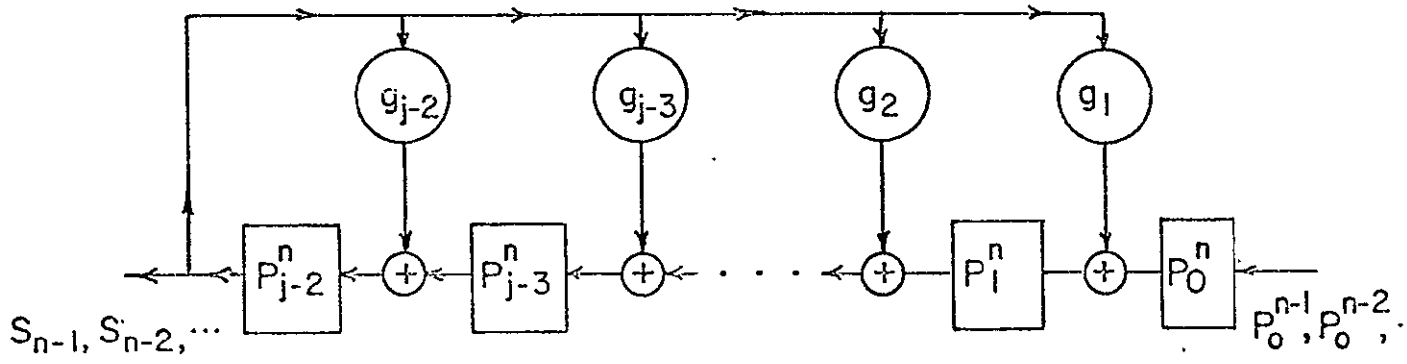


Figure 1b: Circuit that recovers information digits $s_{n-1}, s_{n-2}, \ldots, s_o$ from parity polynomial $P^n(D)$ and parity digits $p_o^{n-1}, p_o^{n-2}, \ldots, p_o^{\nu-1}$

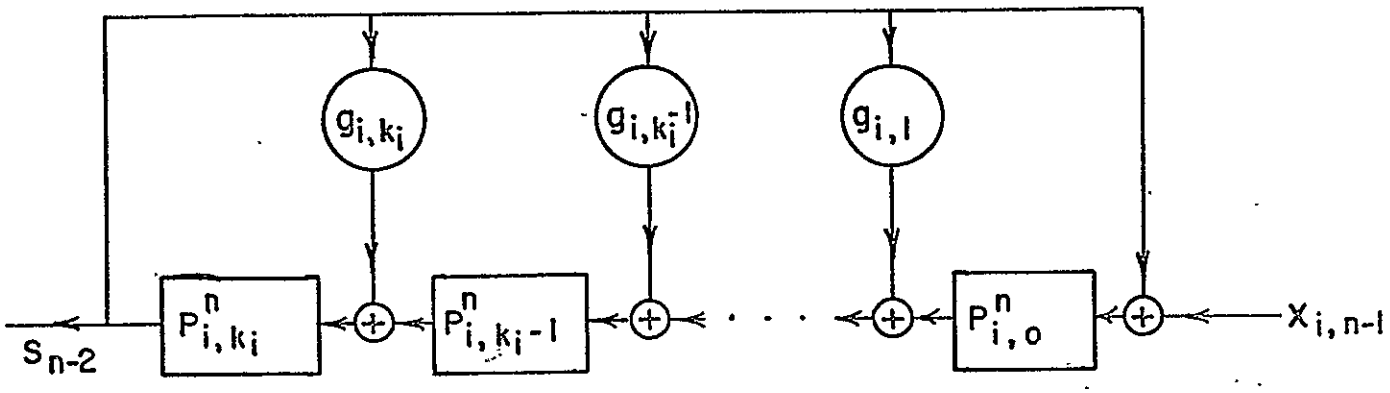


Figure 2a: Circuit that obtains $P_i^{n-1}(D)$ from $P_i^n(D)$ and $x_{i,n-1}$. It can also be used to recover the information digits $s_{n-1},\ldots,s_o$ from $P_i^n(D)$ and $x_{i,n-1},x_{i,n-2},\ldots,x_{i,0}$.

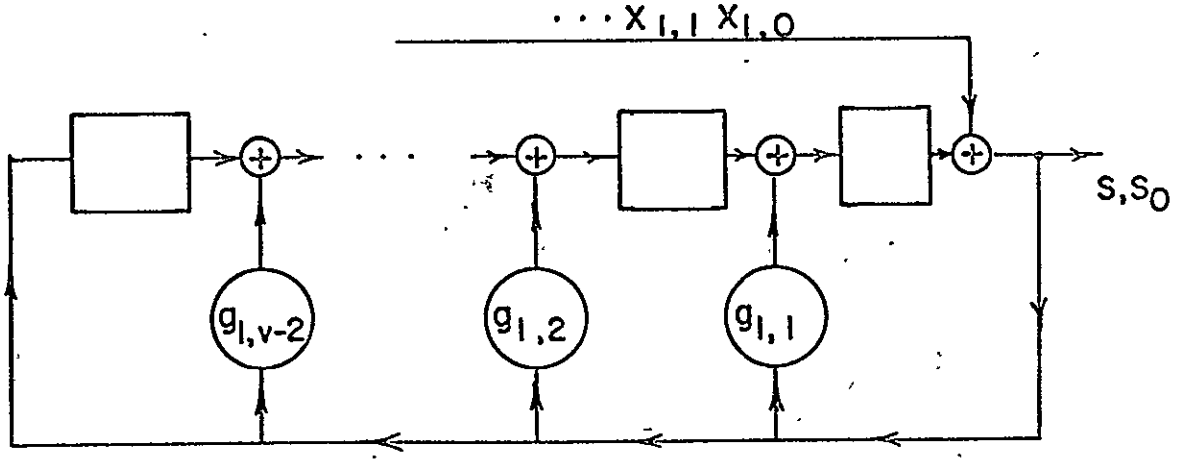


Figure 2b: Feedback circuit that obtains $s_0,\ldots,s_{n-1}$, from $x_{1,0},\ldots,x_{1,\Gamma-1}$. Initial contents of the shift register are 0's.

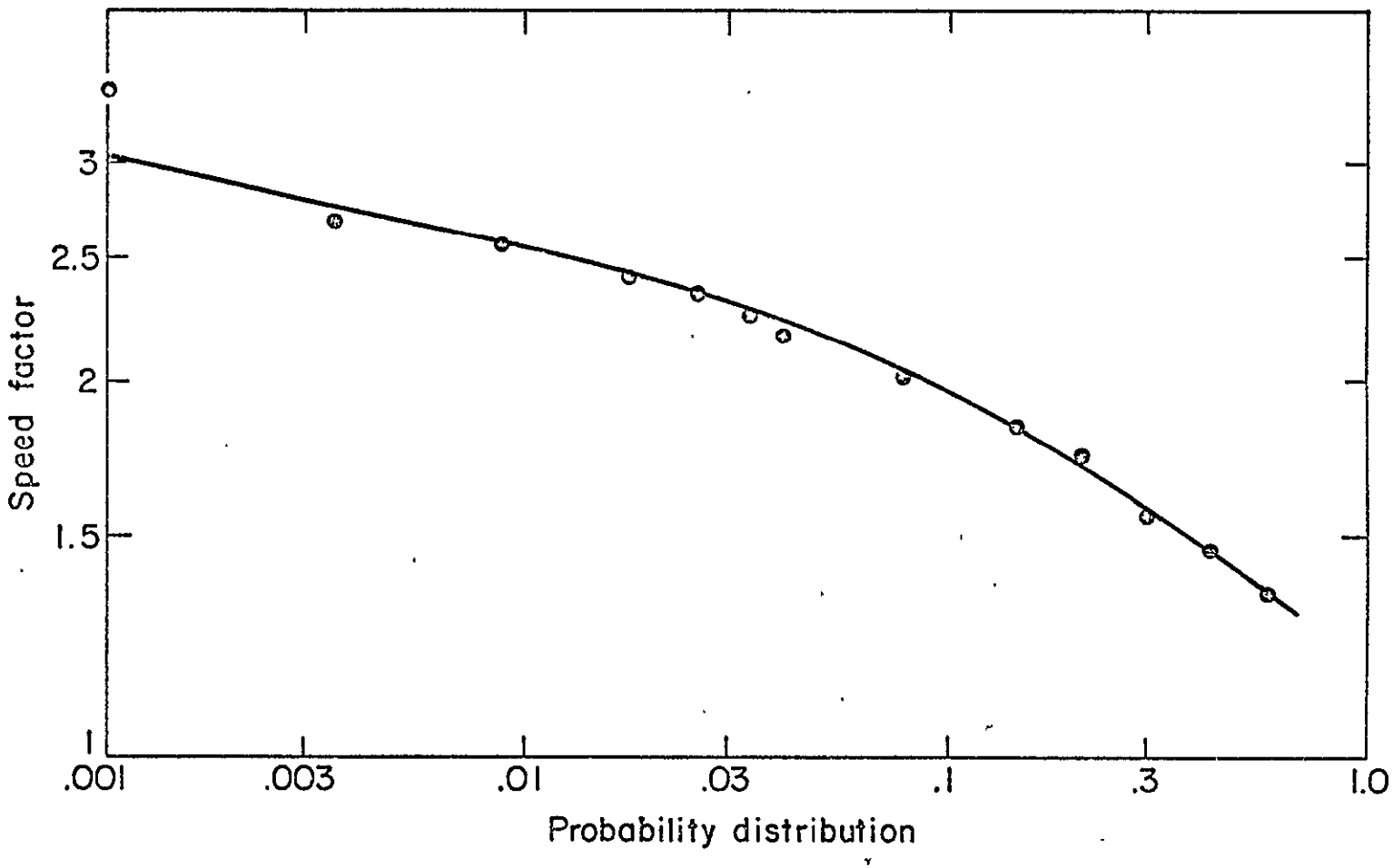


Figure 3: Empirical distribution of the speed factor necessary for rudimentary bootstrap hybrid decoding of $R_{NET} = 0.45$, $m = 10$ over a BSC with $p \approx 0.056$.

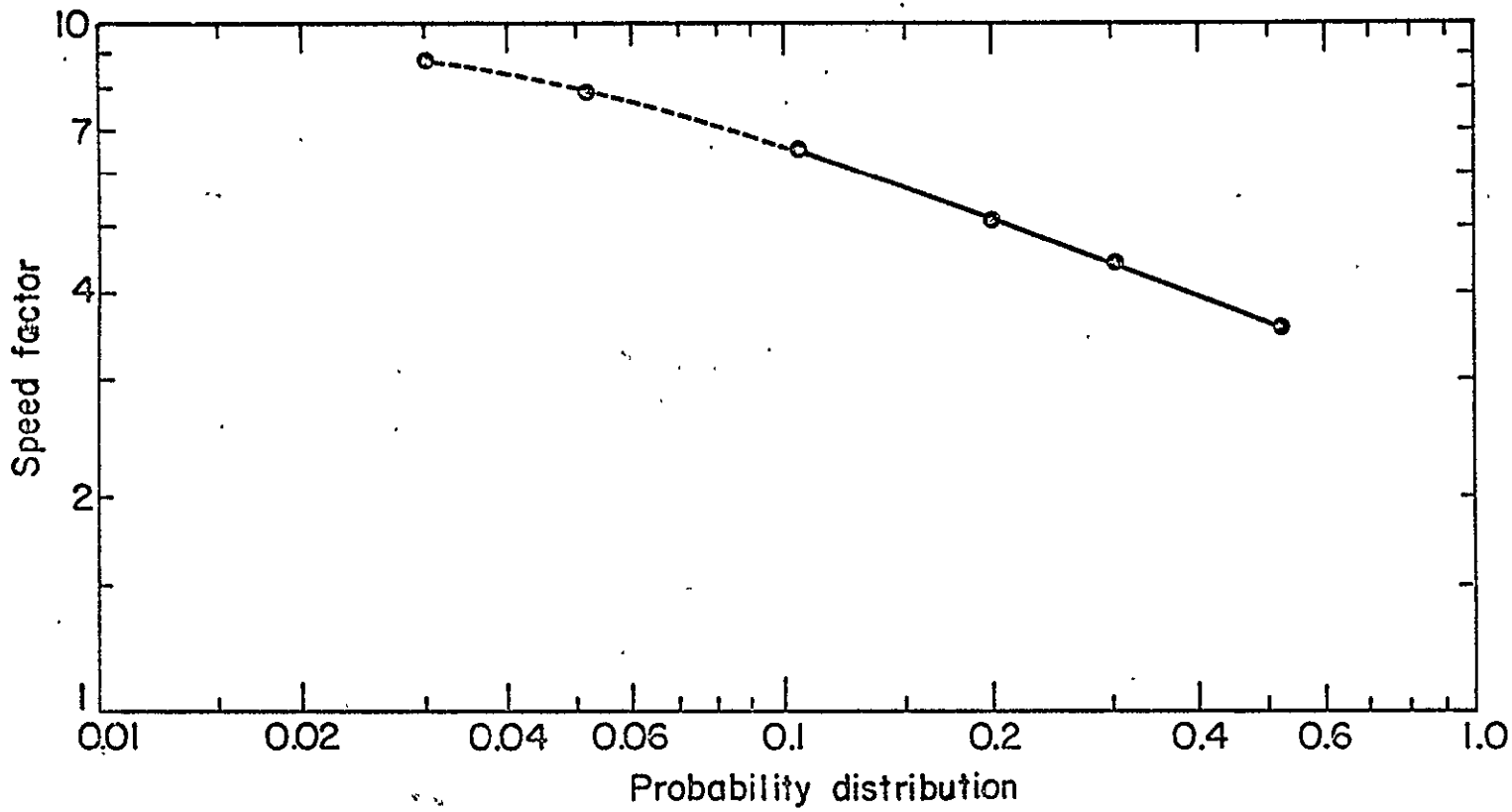


Figure 4: Empirical distribution of the speed factor necessary for rudimentary bootstrap hybrid decoding of $R_{NET} = 0.045$, $m = 10$ over a BSC with $p = 0.07$.

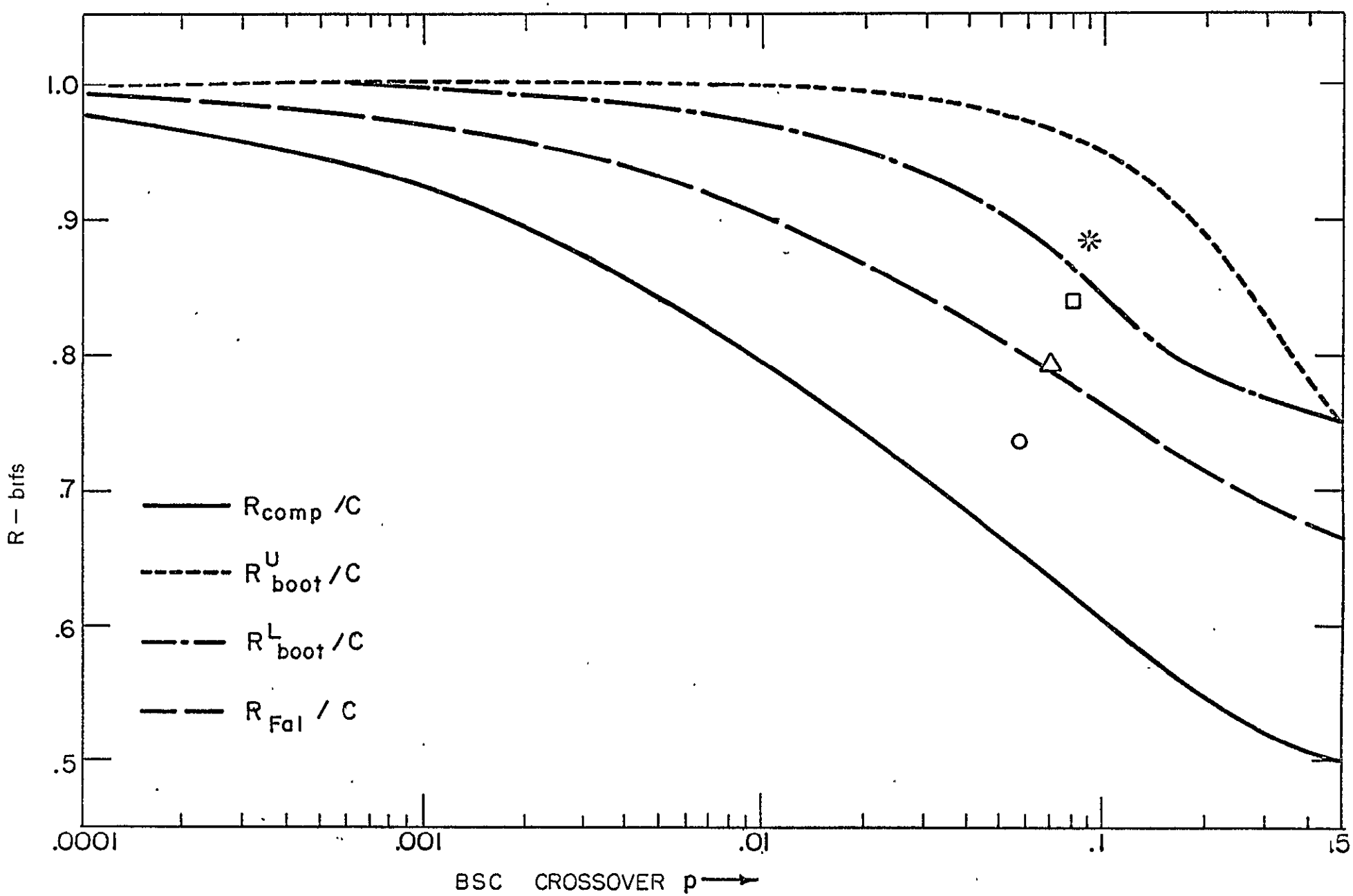


Figure 5: Comparison of performance characteristics of sequential decoding, Falconer's hybrid decoding, and bootstrap hybrid decoding over the BSC. The experimental points denote simulations at R=0.5 referred to in Table 4.

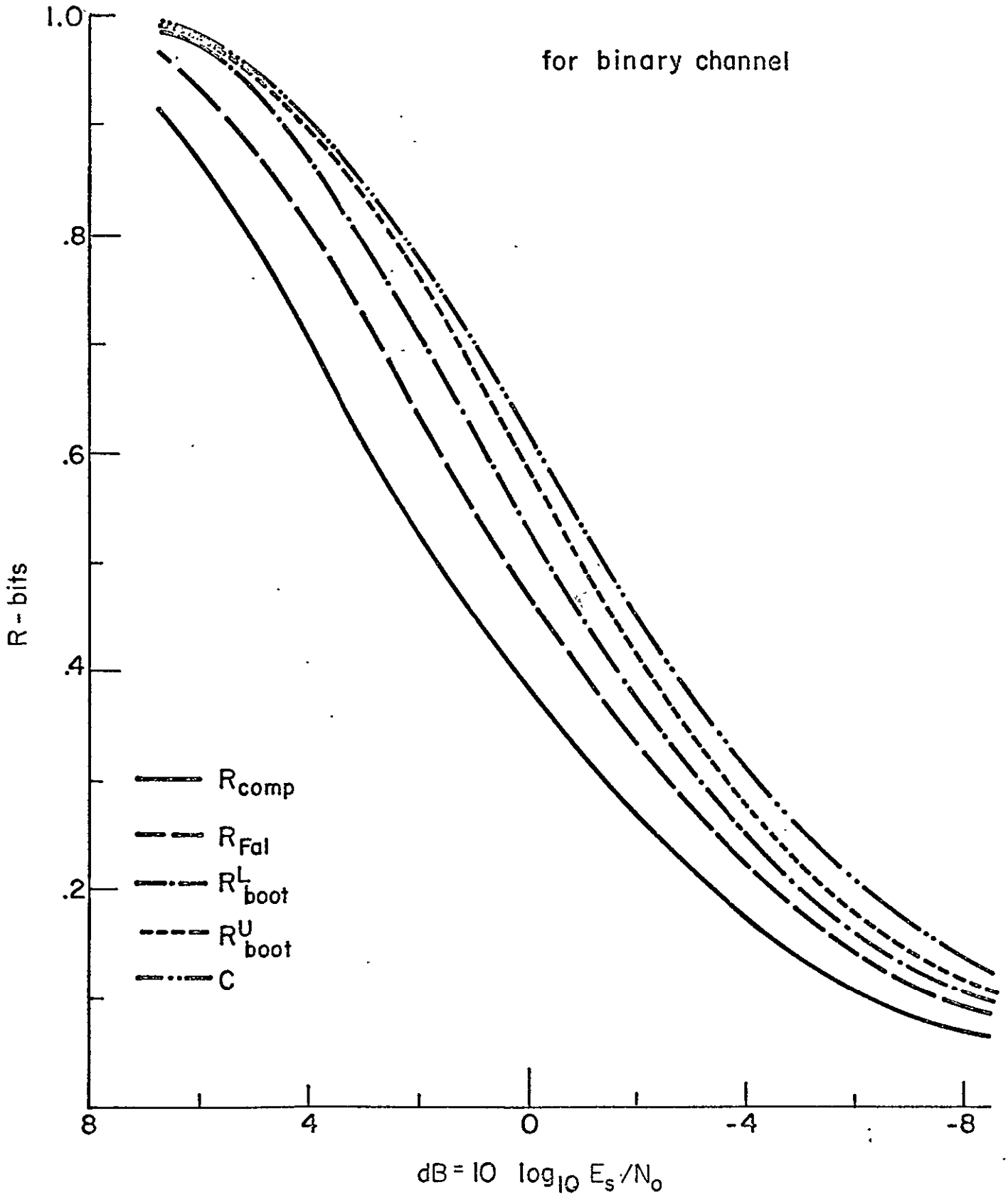


Figure 6: Comparison of performance characteristics of sequential decoding, Falconer's hybrid decoding, and bootstrap hybrid decoding with the capacity of a Gaussian channel with binary inputs and outputs.

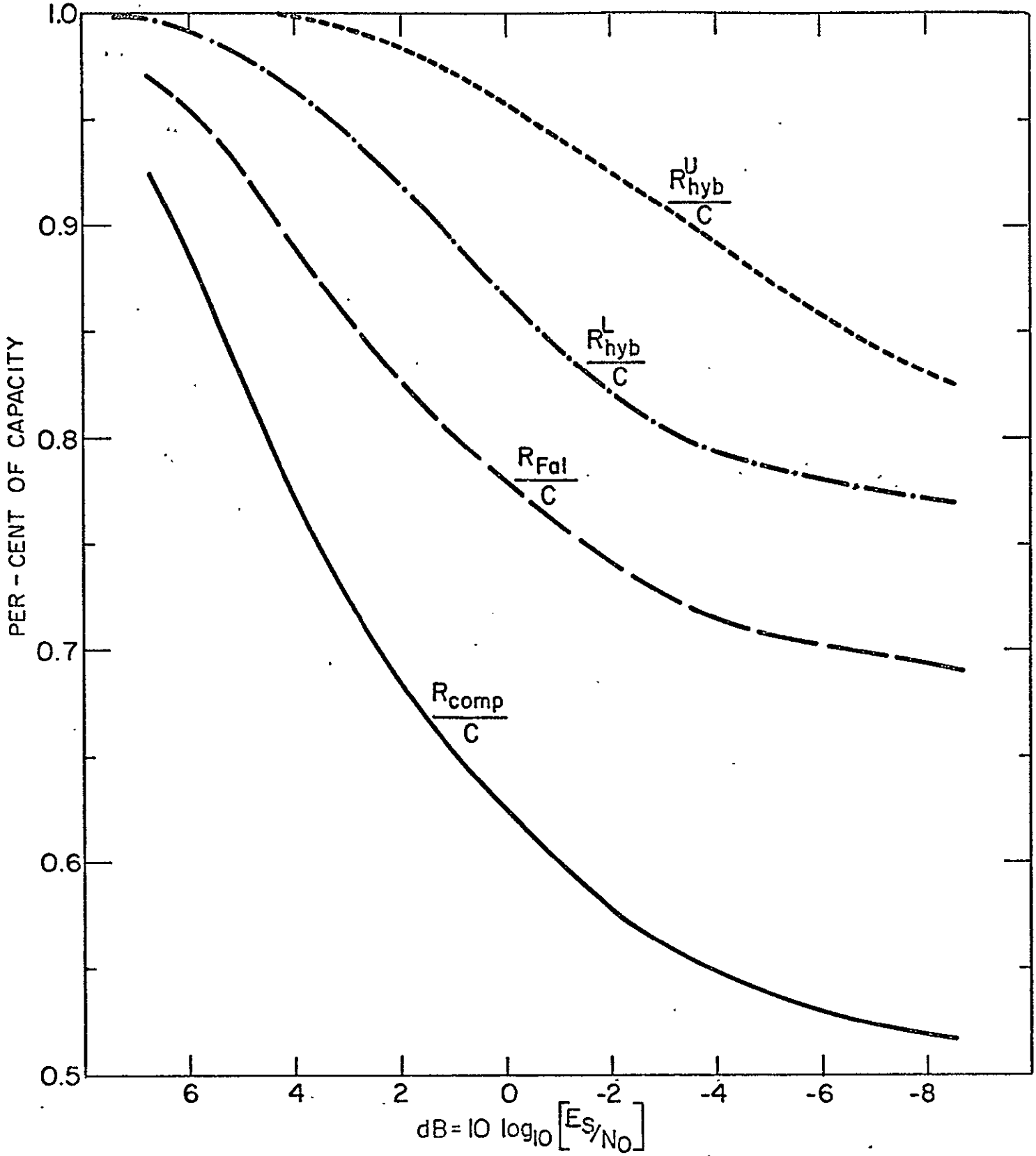


Figure 7: Plots of R_{comp}/C, R_{FAL}/C, $R^L_{BOOT}(1)/C$, and $R^U_{BOOT}(1)/C$ as a function of SNR per transmitted digit in dB's for the binary quantized gaussian channel with binary inputs.

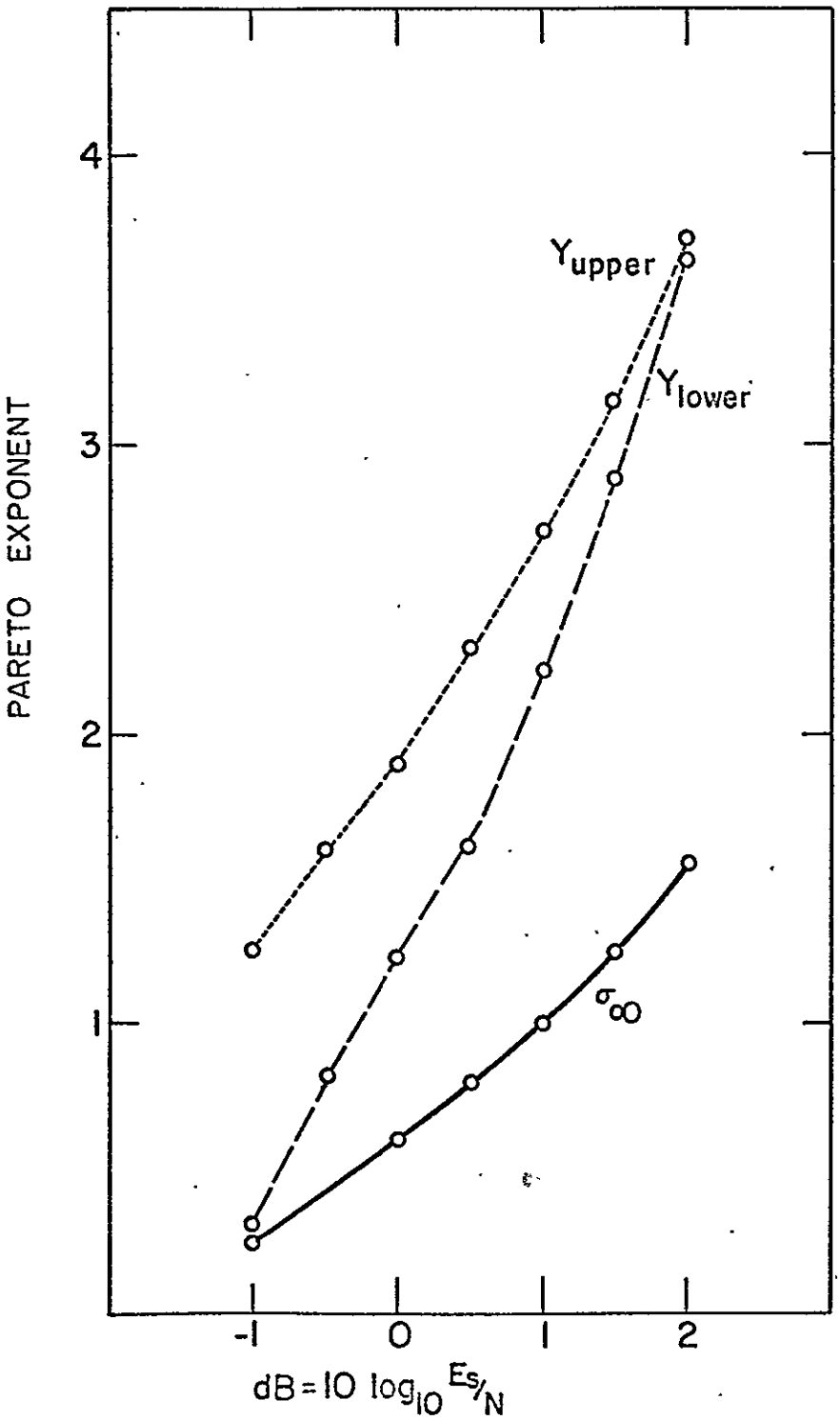


Figure 8: Upper and lower bounds to the Pareto exponent γ for hybrid decoding as a function of SNR per transmitted digit (dB) when the convolutional rate R = 1/2. The sequential decoding Pareto exponent σ is provided for comparison.

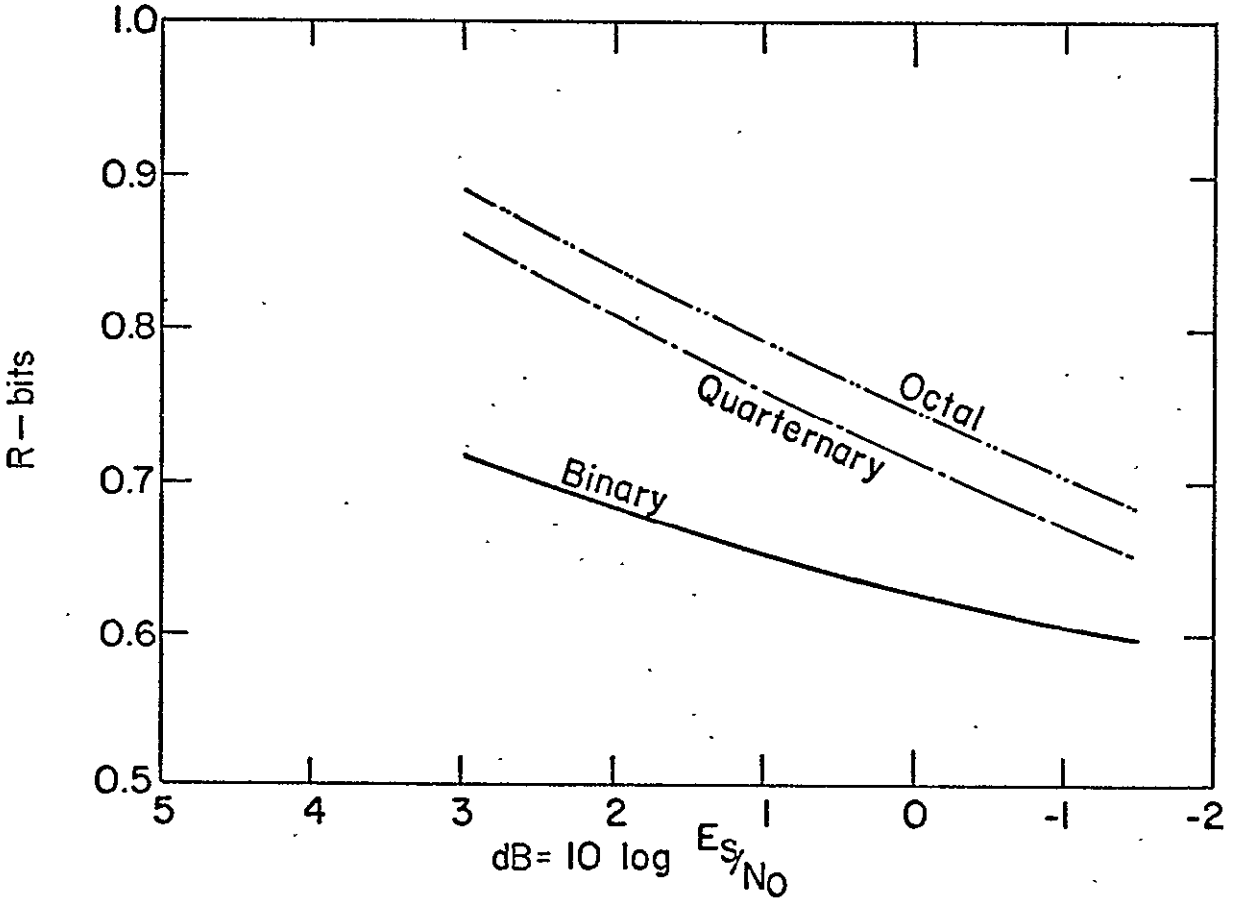


Figure 9: R_{comp}/C vs. SNR per transmitted digit (dB) for binary, quarternary and octal optimal quantization of the Gaussian channel with binary inputs.

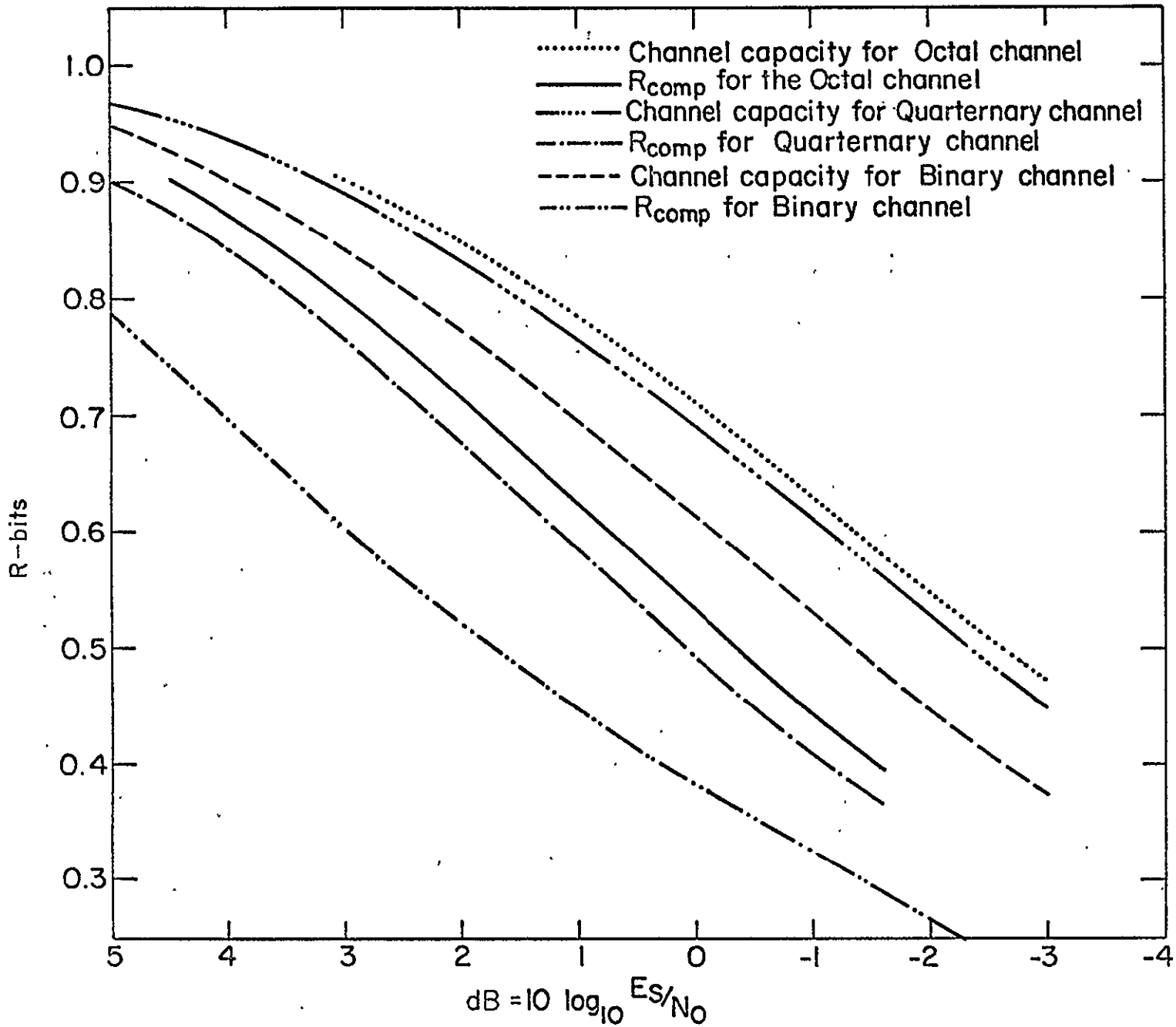


Figure 10: C and R_{comp} vs. SNR per transmitted digit (dB) for binary, quarternary and octal optimal quantization of the Gaussian channel with binary inputs.

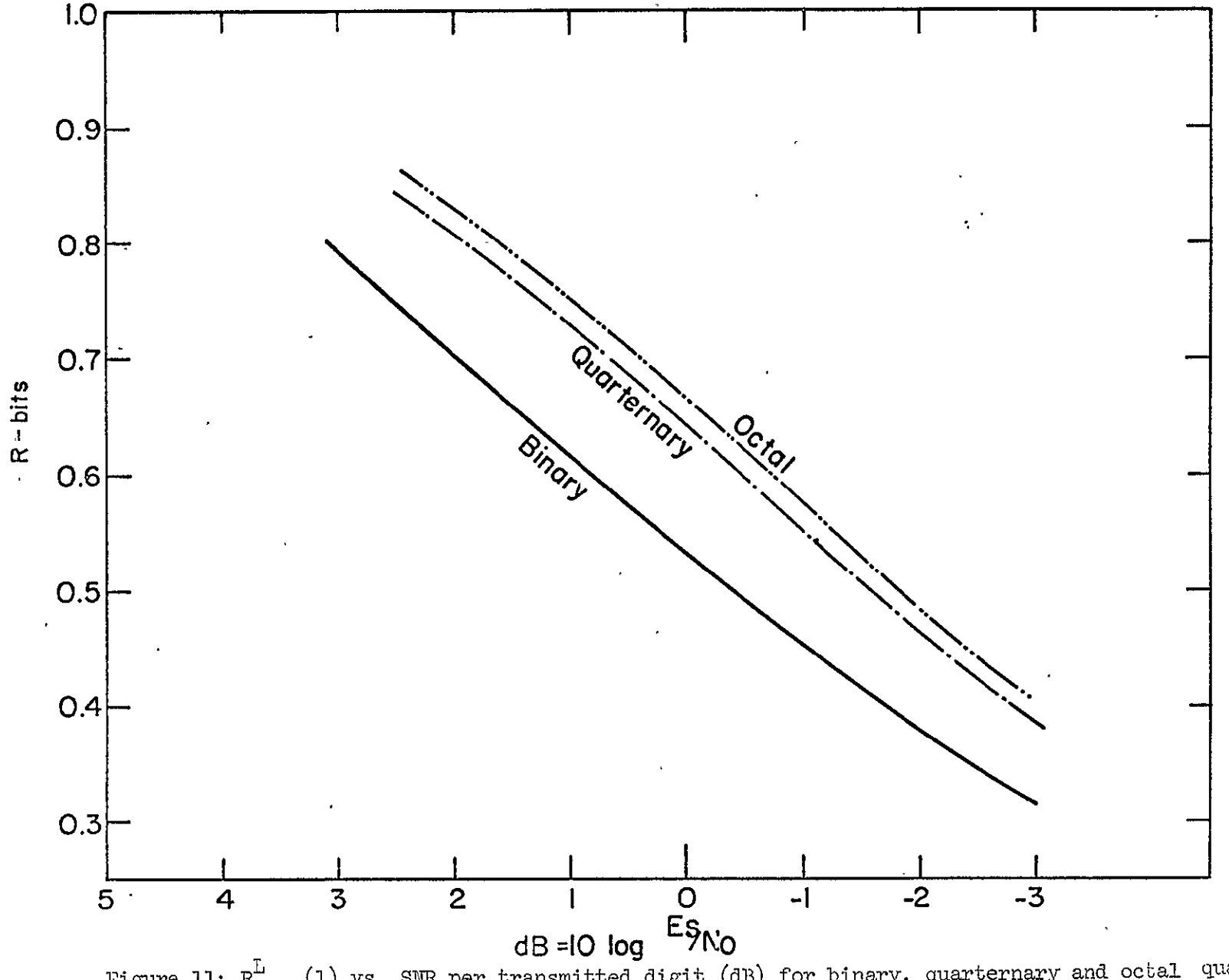


Figure 11: $R^{L}_{BOOT}(1)$ vs. SNR per transmitted digit (dB) for binary, quarternary and octal quantization when state stream contains maximal information.

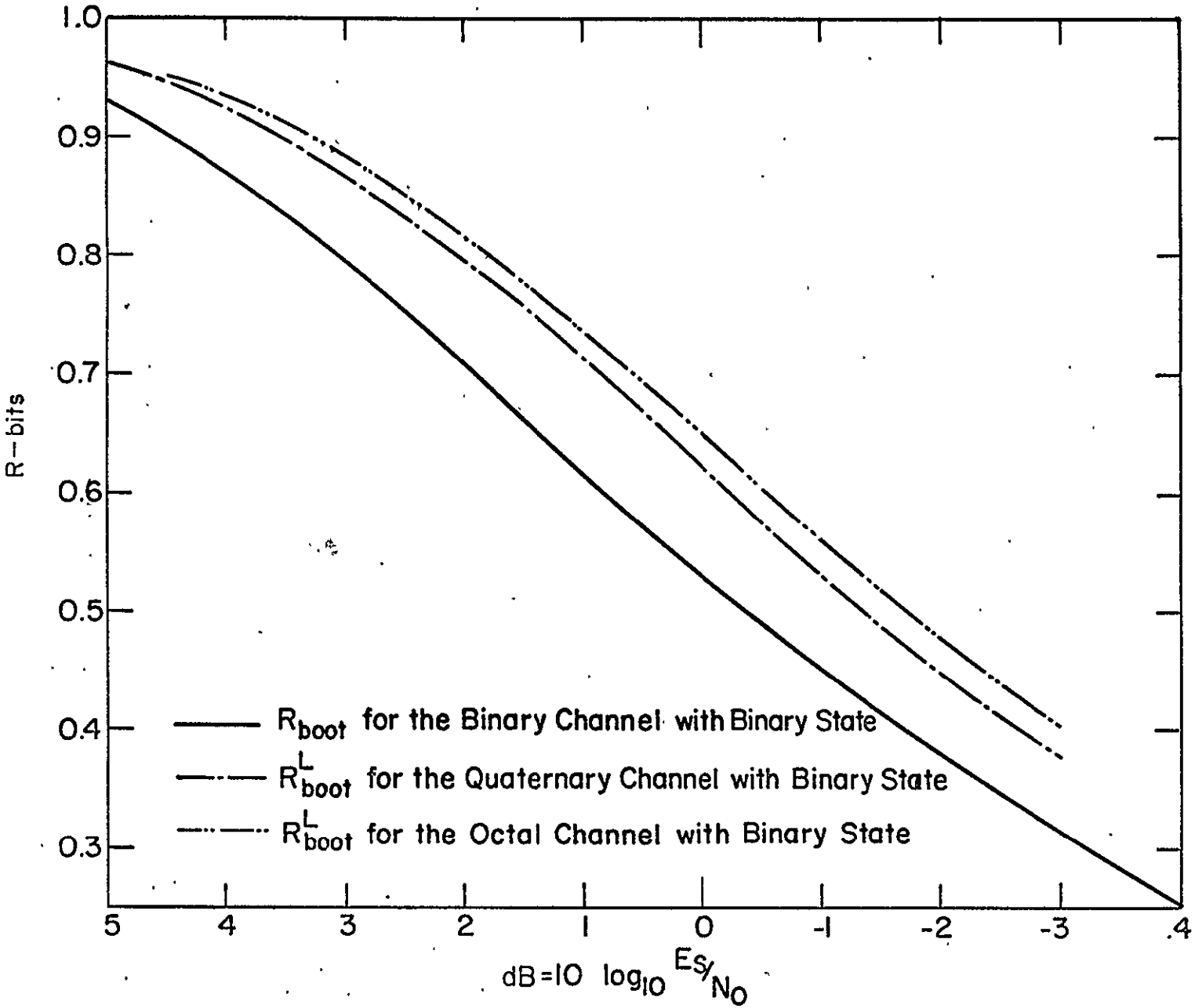


Figure 12: The parameter of Figure 11 when state stream is binary.

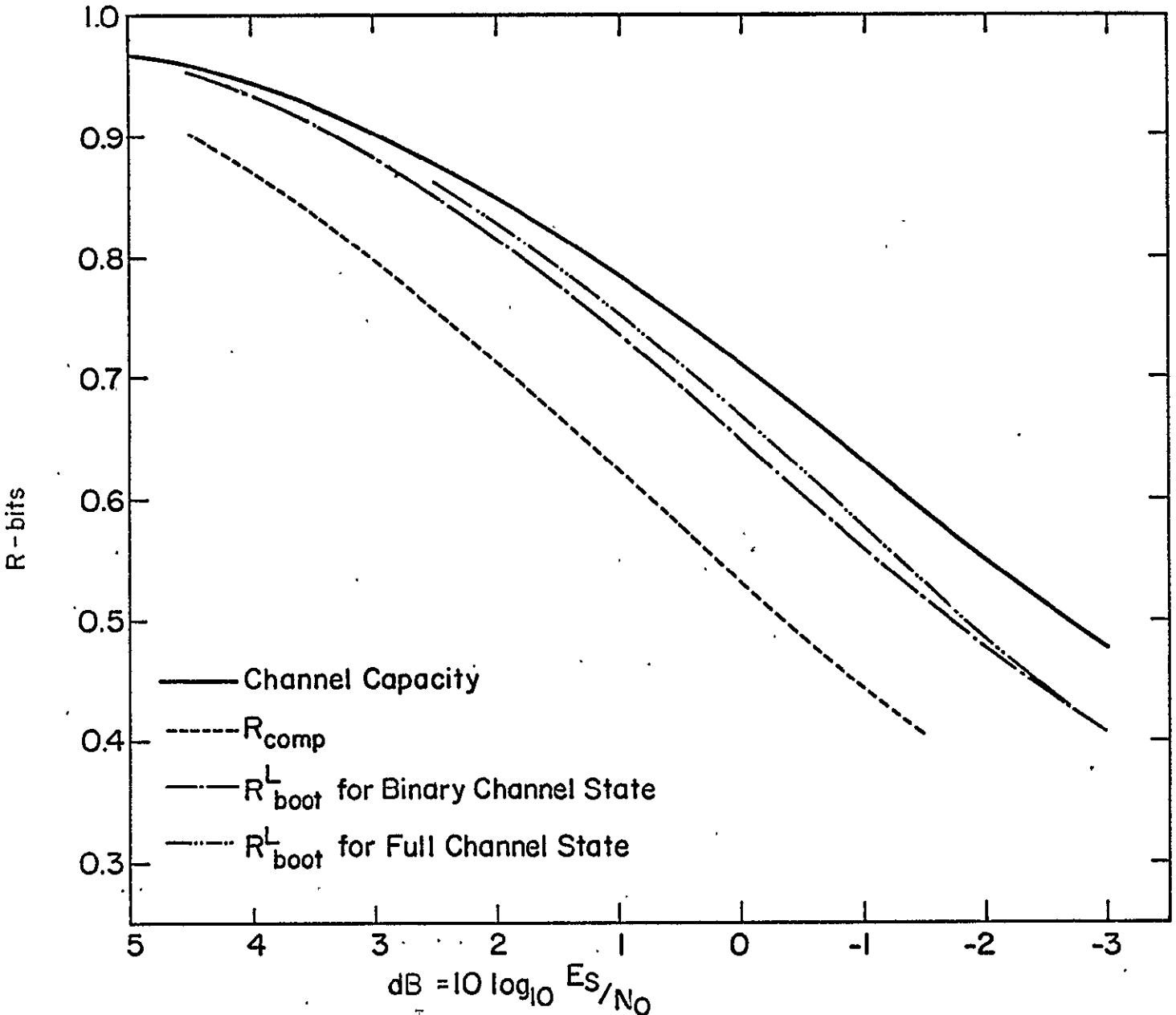


Figure 13: C, R_{comp}, and $R^{L}_{BOOT}(1)$ for binary and full information state stream for the quarternary output channel as a function of SNR per transmitted digit (dB).

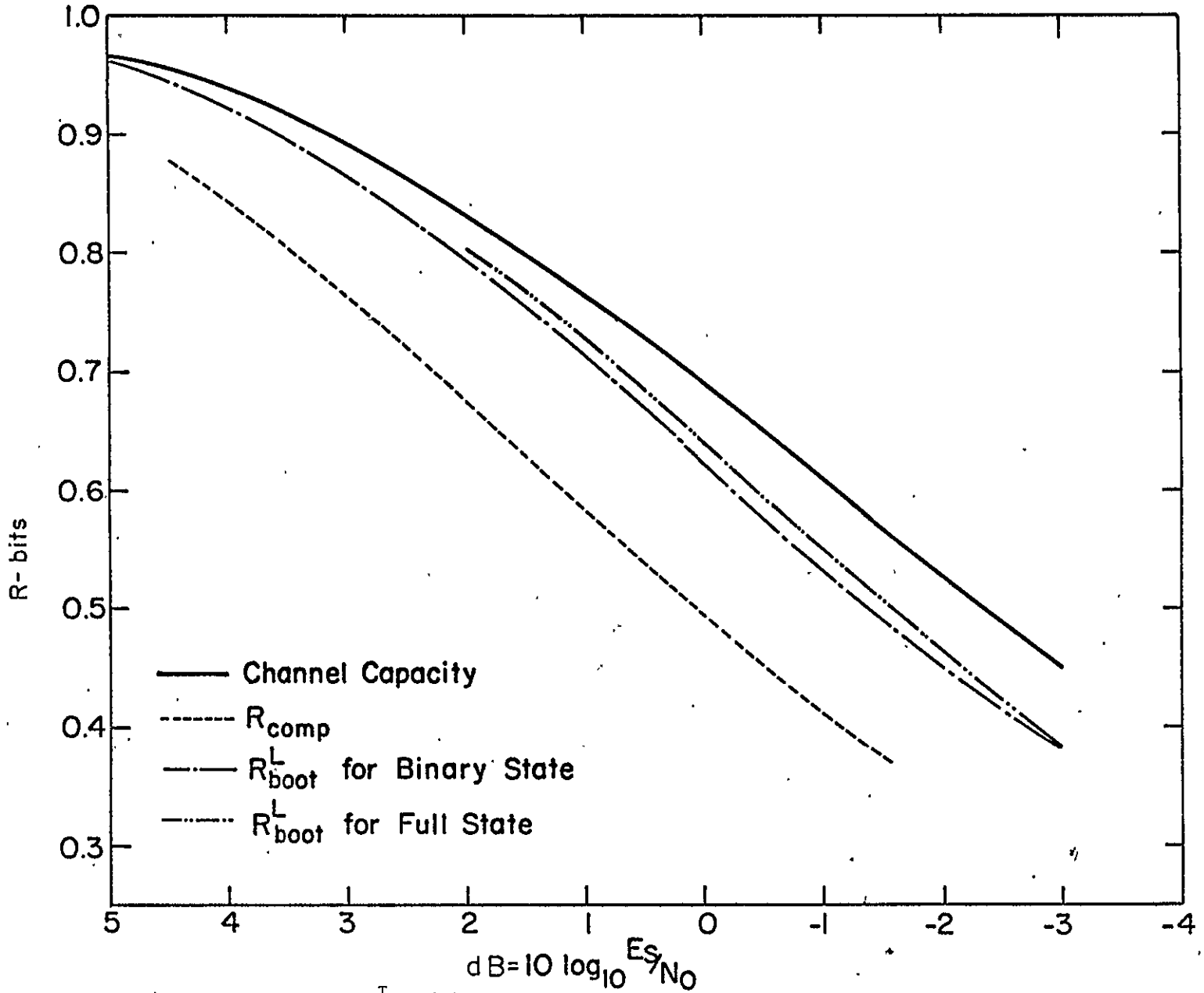


Figure 14: C, R_{comp} and $R^{L}_{BOOT}(1)$ for binary and full information state stream for the octal output channel as a function of SNR per transmitted digit (dB).

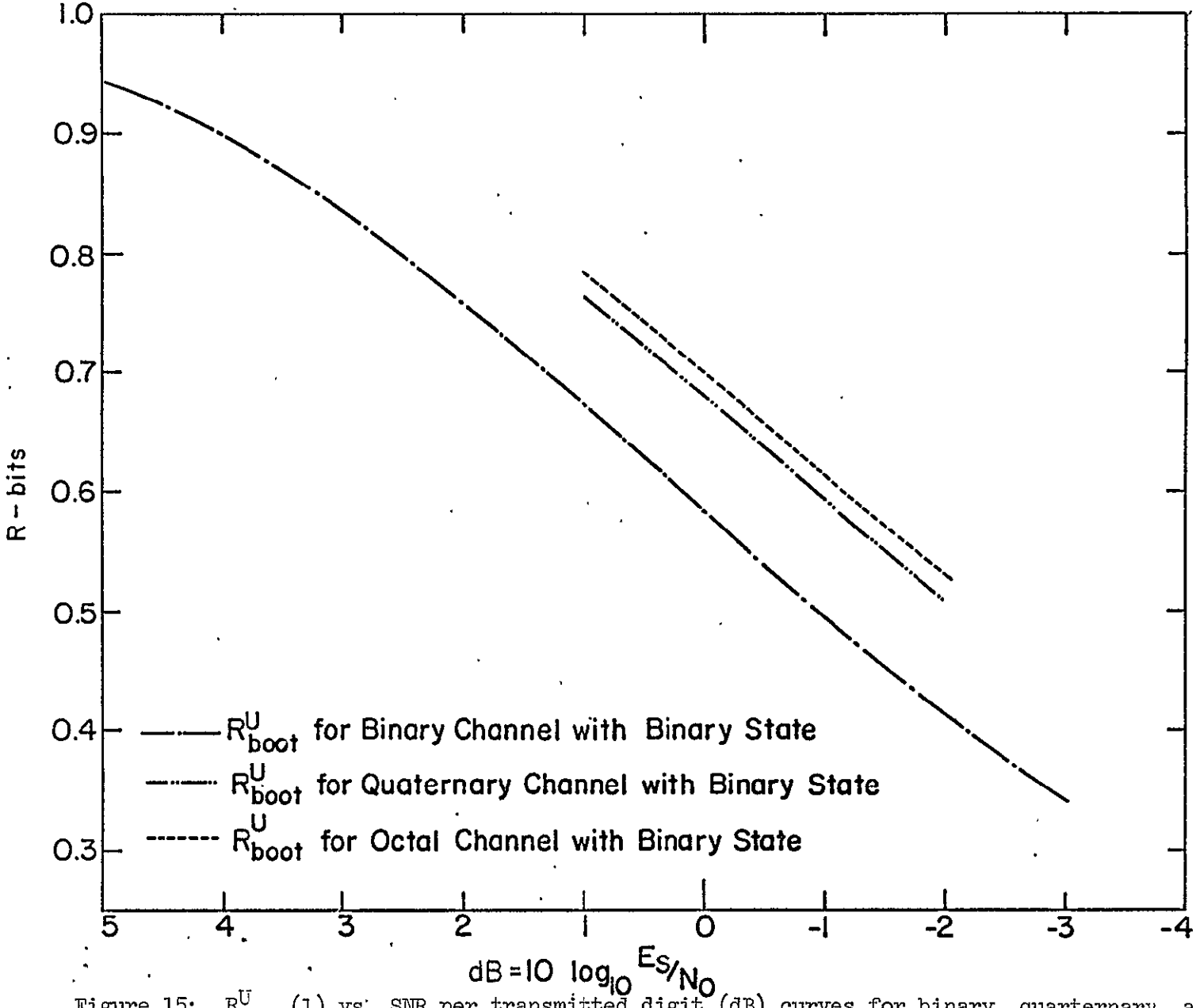


Figure 15: $R^U_{BOOT}(1)$ vs. SNR per transmitted digit (dB) curves for binary, quarternary, and octal output channel with binary inputs when the channel state stream is binary.

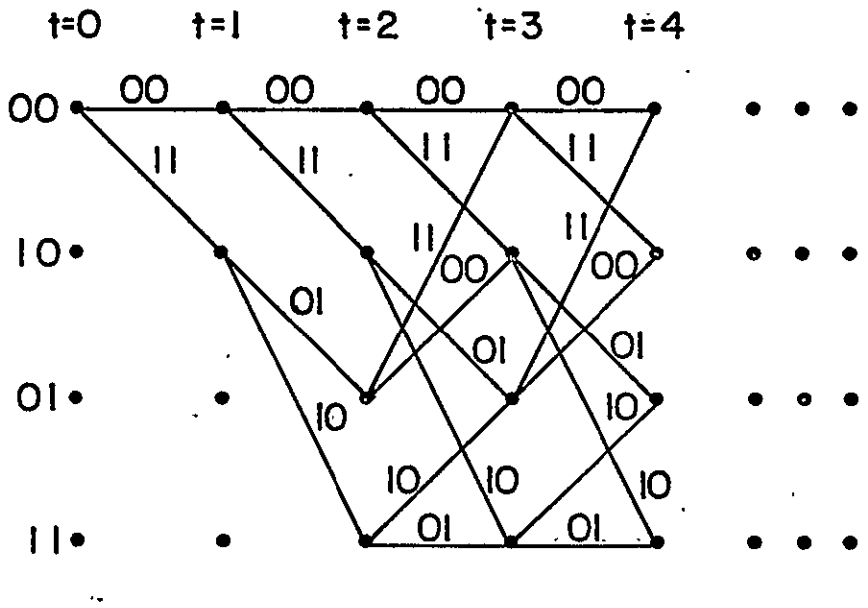


Figure 16: The trellis state diagram for the code $G^{(1)}(D) = 1+D+D^2$, $G^2(D) = 1+D^2$.

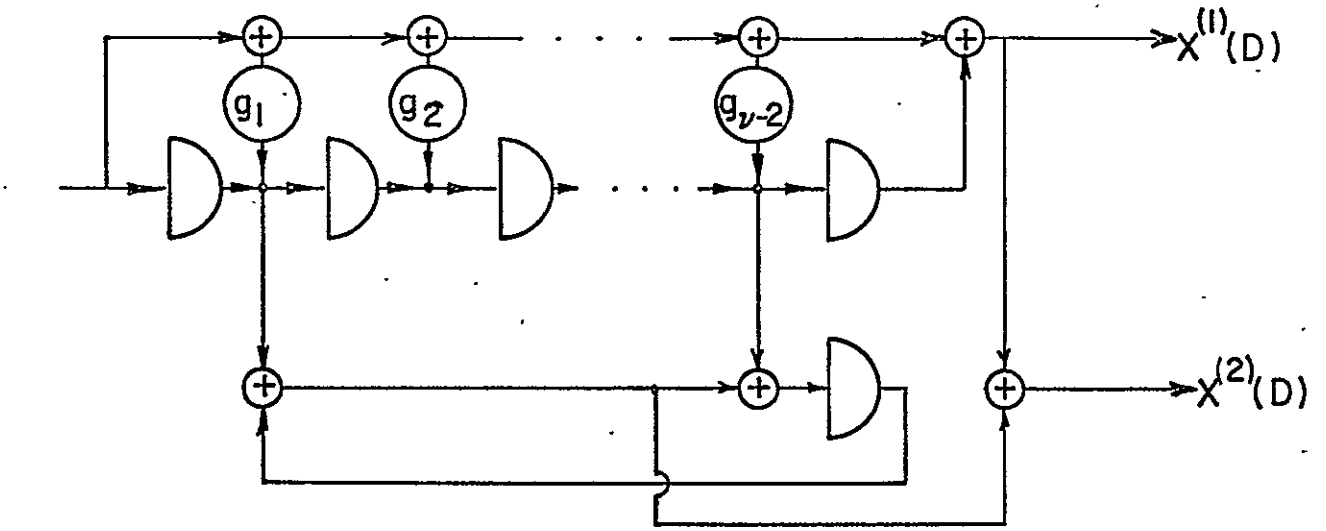


Figure 17: A simplified encoding circuit for complementary rate 1/2 codes.

Figure 18: Free distance of best complementary codes compared to the best available bounds.

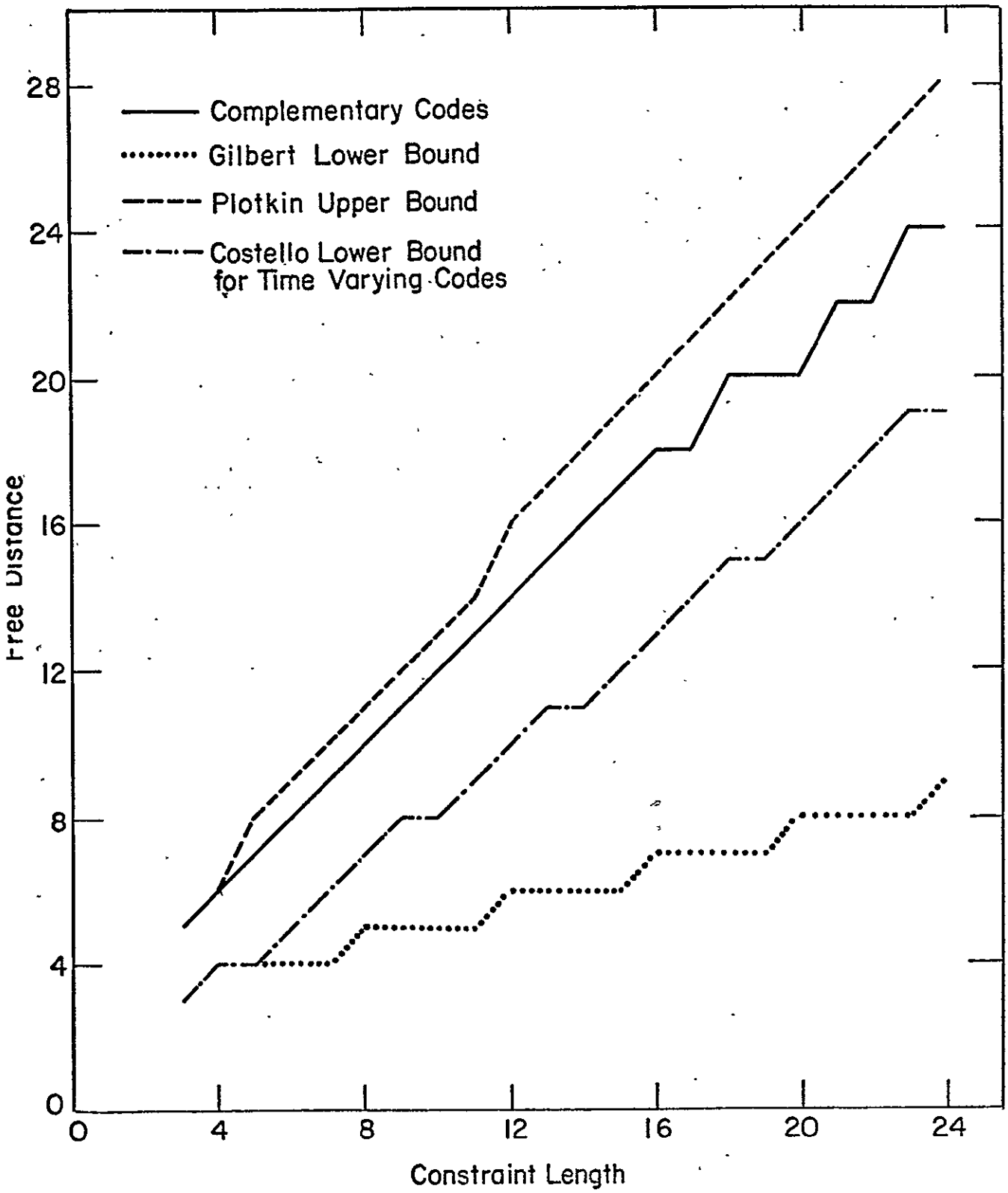


Figure 19: Free distance of complementary and other best codes.

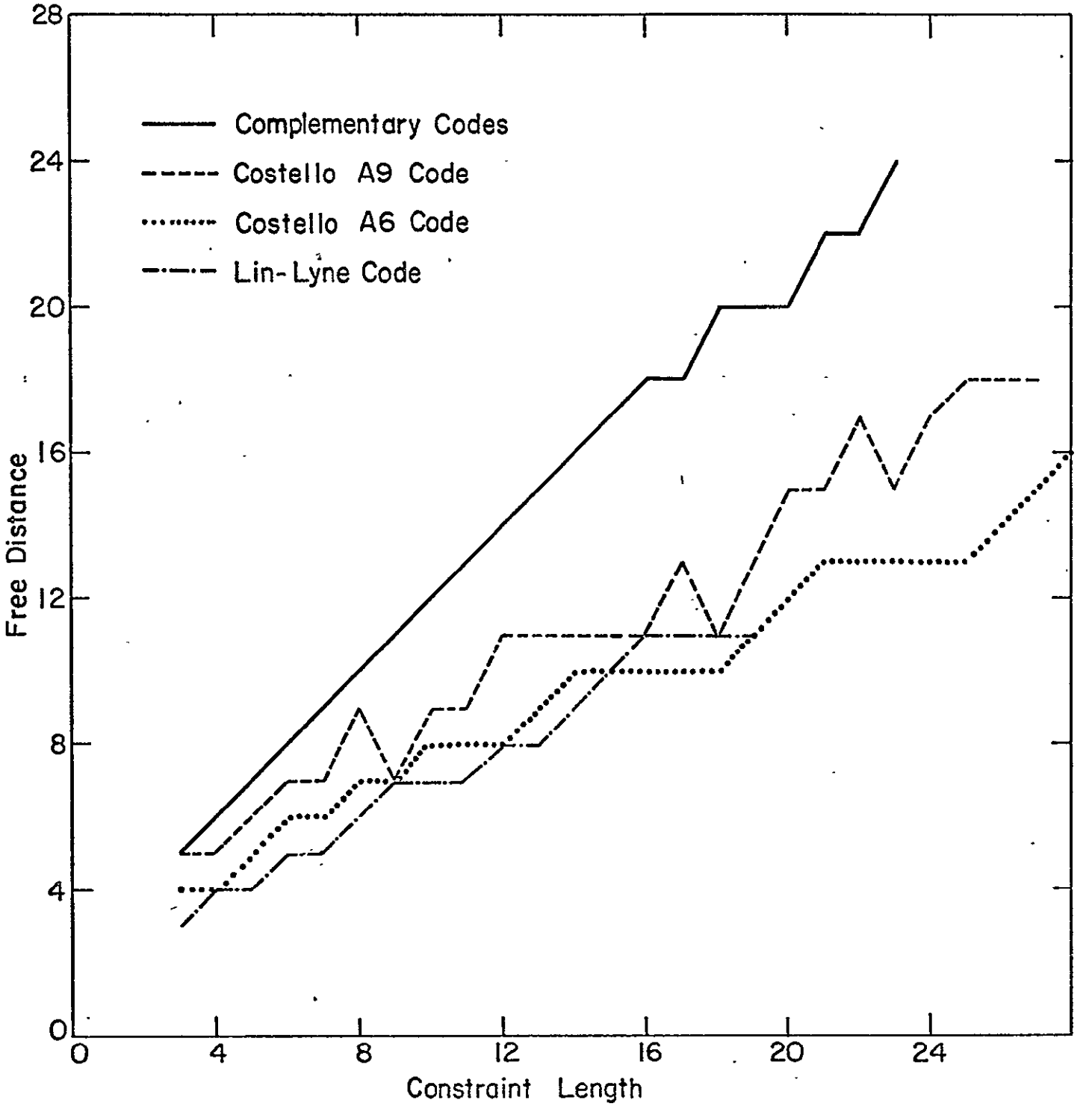
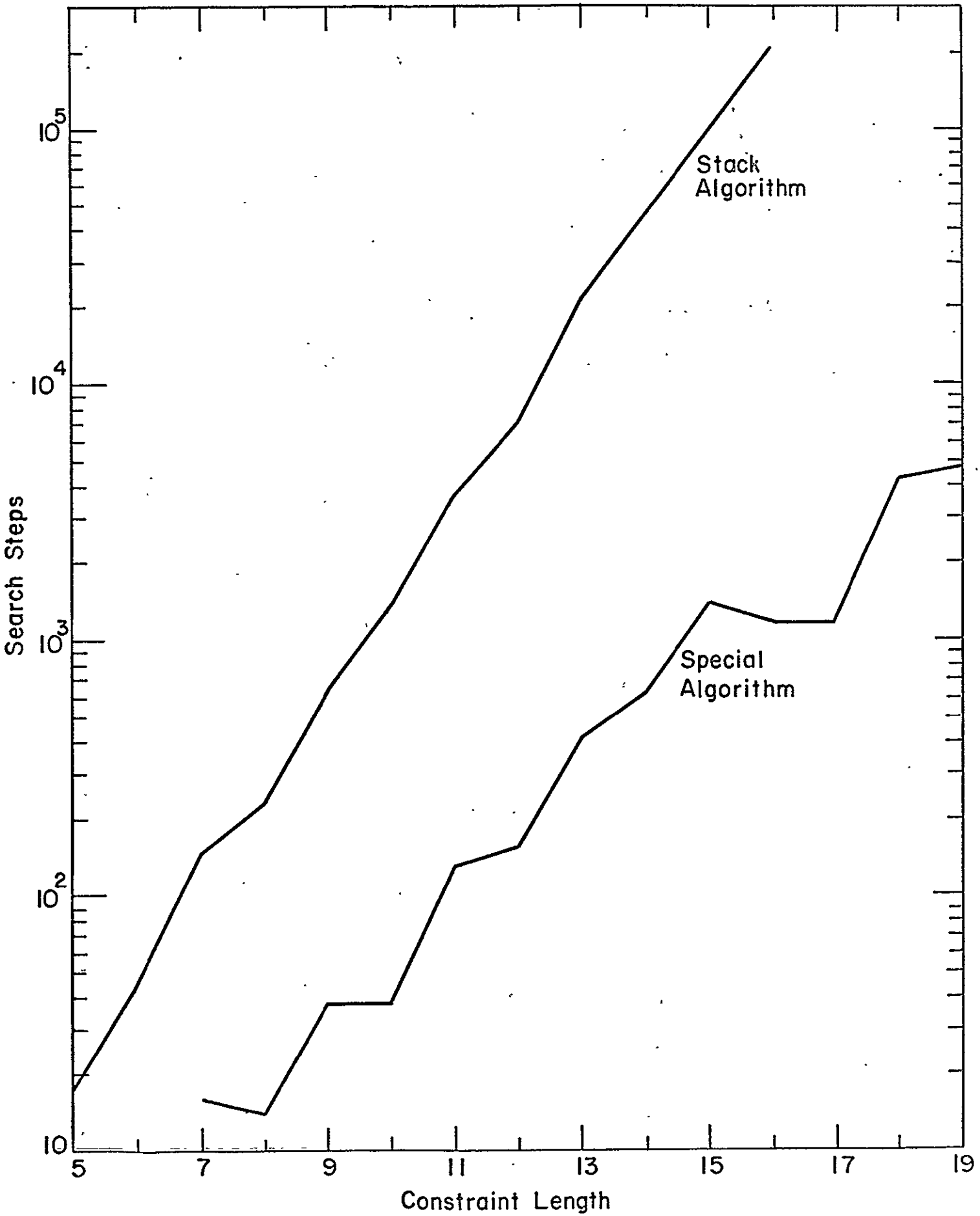


Figure 20: Computational effort necessary to determine free distance of an ordinary stack algorithm and of the special algorithm utilizing the structure of complementary codes.



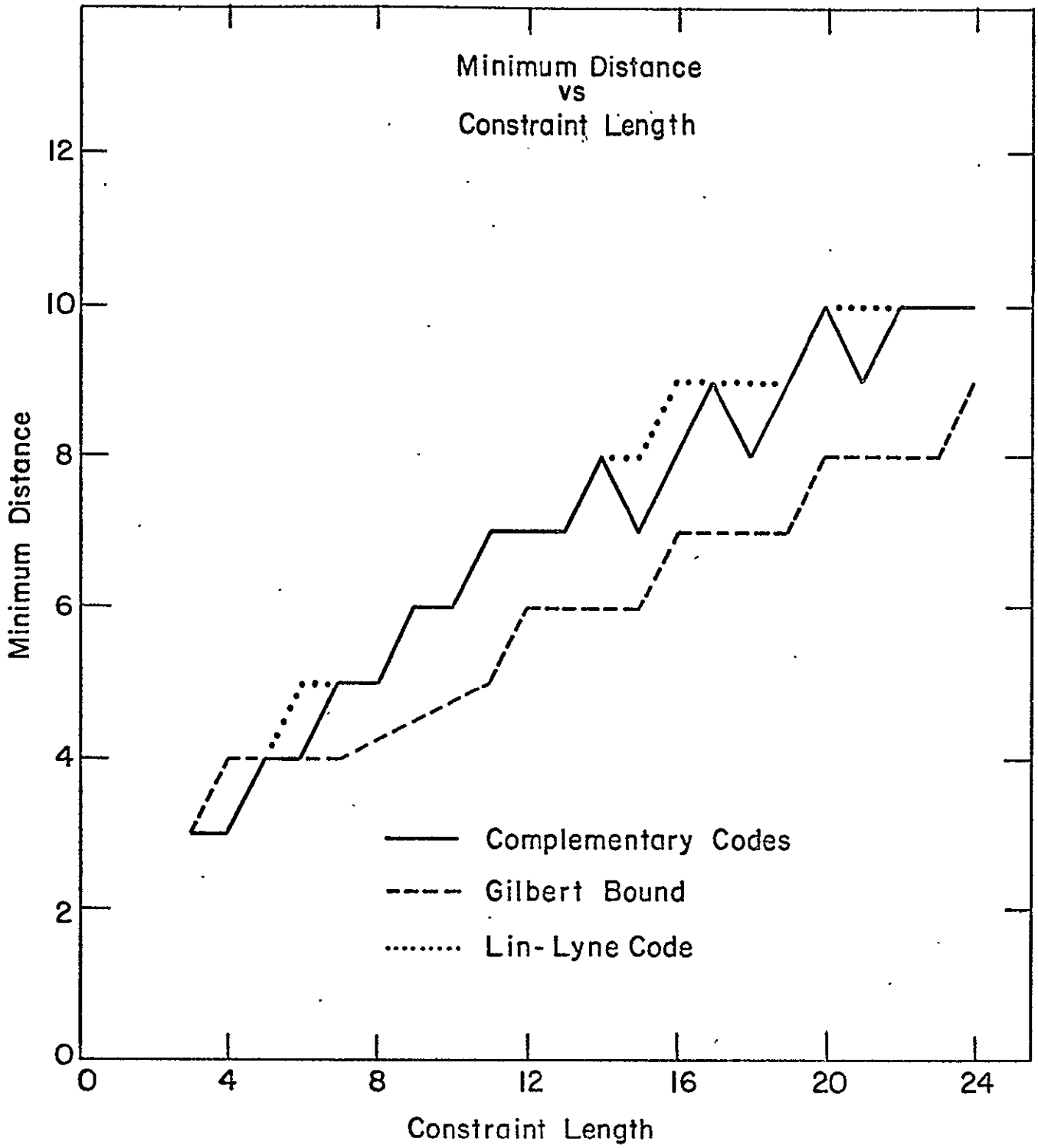


Figure 21: Minimum distance of the highest free distance complementary codes compared to the Gilbert bound and to the minimum distance of the best available codes.

III. REPORT ON PHASE II

III-A. Tree Encoding of Sources with a Fidelity Criterion

III-A-1. Experimental Comparison of Two Encoding Algorithms

The most general theoretical formulation of the data compression problem was provided by Shannon in 1959 in his paper "Coding Theorems for a Discrete Source with a Fidelity Criterion."[1] He enlarged there on his 1949 source coding ideas[2] referred to in the literature as variable length source coding and block source coding. Concisely stated, Shannon's results are as follows: Let a memoryless source of alphabet $A = (0,1,\ldots,a-1)$ governed by the probability distribution $Q(z)$, $z \in A$ be given. Let an approximation of the source outputs in the reproducer alphabet $B = (0,1,\ldots,b-1)$ be desired (in practice $b \leq a$) with an attached additive per letter distortion criterion $d(z,\hat{z})$ defined for all pairs $z \in A, \hat{z} \in B$. (i.e. the distortion between sequences $\underset{\sim}{z}^n = z_1,\ldots,z_n$ and $\underset{\sim}{\hat{z}}^n = \hat{z}_1,\ldots,\hat{z}_n$ is defined to be $d(\underset{\sim}{z}^n,\underset{\sim}{\hat{z}}^n) = \sum_{i=1}^{n} d(z_i,\hat{z}_i)$). Let $\Psi_n(z^n)$ be an encoding function that assigns some reproducer sequence $\underset{\sim}{\hat{z}}^n$ to each possible source sequence $\underset{\sim}{z}^n$. The rate of the resultant code is defined to be $R = \log\left|\left|\Psi_n\right|\right| /n$ where $\left|\left|\Psi_n\right|\right|$ denotes the number of sequences in the range of $\Psi_n(\ )$. Shannon shows the existence of a rate distortion function $R(D)$ [whose shape depends on $Q(\ )$ and $d(\ ,\ )$ only] that has the following properties:

a) for all n and all codes Ψ_n, if $R < R(D)$ then the expected distortion $\underset{\sim}{E}\left[\frac{1}{n} d(\underset{\sim}{z}^n;\Psi_n(\underset{\sim}{z}^n))\right] > D$.

b) for $R \geq R(D)$ there exists a sequence of codes Ψ_n^* of rate $\log\left|\left|\Psi_n\right|\right| /n \leq R(D)$ such that $E[\frac{1}{n} d(\underset{\sim}{z}^n;\Psi_n^*(\underset{\sim}{z}^n))] \longrightarrow D$.

In recent years a lot of work has been done generalizing the above results to a broader class of sources, evaluating the performance of existing systems relative to the achievable optimum, and developing methods for evaluation of the R(D) function. The first consideration of the actual coding problem was undertaken by Jelinek[3] who showed that the sequence of coding functions $\{\psi_n^*\}$ can possess the above desirable properties even if it is restricted to generate tree codes (instead of block codes to which Shannon's theorem applies). It was hoped that a tree code structure would facilitate the development of a computationally feasible encoding algorithm.

Our work concerns the performance of such algorithms as applied to the restricted class of binary symmetrical sources $[Q(0) = Q(1) = 1/2$, $a = b = 2$, $d(0,0) = d(1,1) = 0$, $d(0,1) = d(1,0) = 1]$. The algorithms themselves are, however, completely general. An example of a tree code is given in Figure 1. The various codewords are the sequences associated with the $2^5 = 32$ different paths of the tree. A path is specified by a binary <u>map sequence</u> $\underset{\sim}{s}^5$ which determines at each node level if the upper (0) or the lower (1) branch was taken. Thus the map sequence $\underset{\sim}{s}^5 = 01101$ corresponds to the codeword $\hat{\underset{\sim}{z}}^{10} = 0011101100$. The rate of the code of Figure 1 is $R = \frac{\log 32}{10} = 1/2$ so that the theoretically optimal achievable average distortion is $D = .11$. Figure 2 shows an experimentally derived, ultimate capability of specific codes (believed to be near optimal) of constraint lengths 5,7,10, and 14. The curve does seem to indicate that the ultimate performance of $D = .11$ will be achievable with codes of sufficiently long constraint length. The simulation was carried out with the help of a straightforward modification of the Viterbi algorithm that necessitates $2^{\nu-1}$ steps per encoded source digit pair. The top curve in Figure 3 then gives the corresponding distortion performance as a function of the number ff encoding steps. The algorithm compares the

beginning subsequence of length $\geq \nu$ of the source outputs with the difference sequences corresponding to the 2^{ν} initial paths of the code trellis (see Fig. 14 of Section V). Each state at depth ν of the trellis has two such paths entering it. For each state, the one of these two paths whose distortion from the source subsequence is least is retained and the other is eliminated. Extensions of length $\nu+1$ of the retained paths are then compared with the initial course subsequence of length $2(\nu+1)$ and the elimination process is repeated at each trellis state of depth ν. This continues until a preassigned depth Γ in the trellis has been reached. Then the best of the 2^{ν} "live" paths is selected to represent the source output sequence of length 2^{Γ}.

The next algorithm evaluated is based on the stack principle.[4] Let D^* be the per letter distortion desired by the user. To be realistic (see the previously quoted results) we must have $R > R(D^*)$. Define a metric distortion function $d^*(z,\hat{z}) = d(z,\hat{z}) - D^*$. Then $\hat{\underset{\sim}{z}}^i$ will be a acceptable approximation of a source sequence $\underset{\sim}{z}^i$ if and only if $\sum_{j=1}^{i} d^*(z_j,\hat{z}_j) \leq 0$ (we assume that the code is indefinitely extensible, i.e. that the number of levels in the tree is practically infinite). Suppose the sequence $\underset{\sim}{z}^n$ (n large) was generated by the source, let $d^*(\underset{\sim}{s}^j)$ denote the metric relative to $\underset{\sim}{z}^n$ corresponding to the last branch of the path $\underset{\sim}{s}^j$ [e.g., $d^*(101) = d^*(z_5,1) + d^*(z_6,0)$ and $d^*(100) = d^*(z_5,0) + d^*(z_6,1)$], and let $D(\underset{\sim}{s}^j)$ be the cumulative metric along the path $\underset{\sim}{s}^j$, $D(\underset{\sim}{s}^j) = \sum_{i=1}^{j} d^*(\underset{\sim}{s}^i)$ where $\underset{\sim}{s}^i$ are the initial subsequences of length i of $\underset{\sim}{s}^i (i \leq j)$. The <u>stack</u> will contain different paths $\underset{\sim}{s}^j$ and their cumulative metrics $D(\underset{\sim}{s}^j)$, and will be arranged in ascending order of the latter (i.e. at the top of the stack there will be that path $\underset{\sim}{s}^j$ whose $D(\underset{\sim}{s}^j)$ is least).

1. At the beginning of the decoding process, the paths $\underset{\sim}{s}^1=0$ and $\underset{\sim}{s}^2 = 1$ are arranged in the stack according to the values of $D(0)$ and $D(1)$.

2. The encoder checks whether the path $\underset{\sim}{s}^j$ on top of the stack is such that $D^*(\underset{\sim}{s}^j) \leq 0$. If so, go to step 4, if not, go to step 3.

3. The top entry $[\underset{\sim}{s}^j, D(\underset{\sim}{s}^j)]$ is eliminated from the stack, the branch metrics $d^*(\underset{\sim}{s}^j 0)$ and $d^*(\underset{\sim}{s}^j 1)$ are computed, and two new entries $[s^j 0,\ D(\underset{\sim}{s}^j 0) = D(\underset{\sim}{s}^j) + d^*(s^j 0)]$ and $[s^j 1, D(s^j 1) = D(s^j) + d^*(s^j 1)]$ are inserted in the proper location into the stack. Go to 2.

4. The subsequence $\underset{\sim}{z}^{2j}$ is encoded into the codeword $\underset{\sim}{z}^{2j}$ that corresponds to the path $\underset{\sim}{s}^j$. The stack is cleared of all its entries and encoding of the sequence $z_{2j+1}, z_{2j+2}, \ldots$ starts with the insertion of two new entries $[\underset{\sim}{s}^j 0, D(\underset{\sim}{s}^j 0) = d^*(\underset{\sim}{s}^j 0)]$ and $[\underset{\sim}{s}^j 1, D(s^j 1) = d^*(\underset{\sim}{s}^j 1)]$ in their proper order into the stack. Go to 2.

The bottom curve in Figure 3 is a plot of average distortion achieved as a function of the average number of steps necessary to encode a source digit pair when the code of constraint length $\nu = 14$ whose ultimate performance is $D = .116$ was used (see Figure 2). The performance curve for the stack algorithm dominates that corresponding to the modified Viterbi algorithm.

The stack algorithm is readily generalizable to tree codes of rate $R = \frac{k}{n}$ with 2^k branches leaving each node and n digits per branch. Its suitability is determined by the average number of steps necessary to encode a source digit.

III-A-2. Theoretical Analysis of the Stack Encoding Algorithm

Our analytical work with the stack algorithm has divided into two efforts, finding equations in relevant variables and approximating solutions to these equations. Presented here is the result of the first effort.

To facilitate analysis, consider several component processes, all running on the copies of the same tree and source. These will combine to form a stack encoder. Let $a > 0$ and $b < 0$. As usual, let a node extension include scrutiny of the d branches extending from a common parent node.

Process 1 Suppose an entire tree is explored by the stack algorithm until either the top metric in the stack exceeds a or falls below b, whichever comes first. Define $N(a,b)$ to be the number of extensions in the first of the d subtrees stemming from the tree's root node.

Process 2 In this process only the first subtree is explored in the stack, again until the stack top exceeds a or falls below b. Let $N^*(a,b)$ be the number of extensions.

Process 3 Here let subtrees $2,\ldots,d$ be explored, until the stack top exteeds a. b is effectively $-\infty$. If $0 > b_1 > b_2 > \ldots$ and the possible top stack minima in this process, let

$$\phi(b_i) = \begin{cases} 1 & \text{if the stack top falls to } b_i \text{ and no further} \\ 0 & \text{otherwise} \end{cases}$$

Concerning N^* and N, certainly

$$N(a,b) \leq N^*(a,b) \tag{1}$$

since in Process 1 searching in the first subtree may be terminated by events in the other $d-1$ subtrees. Process 1 nearly constitutes the stack algorithm, and $N(a,b)$ is closely related to its computation. In fact, defining $N_T(a,b)$ to be the computations in a stack encoder which "gives up" when its stack top falls below b,

$$EN_T(a,b) = 1 + d\, EN(a,b) \tag{2}$$

In (2), the unit term on the right represents the initial computation needed to reach the d subtree structures. The d-factor follows from the statistically IID behavior of the d subtrees. To reflect exactly the four step stack algorithm of the previous section, a is set arbitrarily close to zero and b is reduced to $-\infty$, so that the algorithm stops only when its top path metric exceeds zero.

To pursue this further and arrive at an equation in EN_T, we prove the following lemma about N and N^*:

Lemma

$$N(a,b^*) = \sum_{b_i > b^*} N^*(a,b_i)\, \phi(b_i) + N^*(a,b^*) \sum_{b_i \leq b^*} \phi(b_i) \tag{3}$$

Proof: Case I. $\phi(b_i) = 1$ for $b_i > b^*$. In Process 1, no part of subtree 1 can be examined whose path metrics fall below b_i. On the other hand, if Process 2 with $b = b_i$ can terminate by its stack top rising above a, Stack 1 must never have fallen to b_i. Overall, then, Process 1 examines in subtree 1 precisely what Process 2 does.

Case II. $\phi(b_i) = 1$ for $b_i \leq b^*$

The all subset 1 paths with metrics between a and b_i are examined in both Processes 1 and 2. QED

An expectation operation on (3) now yields,

$$EN(a,b) = \sum_{b_i > b^*} EN^*(a,b_i)\, P_3^a(b_i) + EN^*(a,b^*) \sum_{b_i \leq b^*} P_3^a(b_i) \qquad (4)$$

where $P_3^a(b) = \Pr\left\{\begin{array}{c}\text{Top Stack Minimum in Process 3} \\ \text{is } b\end{array}\right\}$

By a few more maneuvers, we can change (4) into a function of one expectation only, EN_T. We can write immediately,

$$EN^*(a,b) = \sum_{\lambda} EN_T(a-\lambda, b-\lambda)\, P(\lambda) \qquad (5)$$

where

$$P(\lambda) = \Pr\left\{\text{a given branch had incremental metric } \lambda\right\}$$

and $N_T(a,b) = 0$ if $a \leq 0$ or $b \geq 0$. Now combining (2), (4), and (5), we get

$$EN_T(a,b^*) = 1 + d\left\{\sum_{b_i > b^*} P_3^a(b_i) \sum_{\lambda} P(\lambda)\, EN_T(a-\lambda, b_i-\lambda)\right.$$

$$\left. + \left[\sum_{b_i \leq b^*} P_3^a(b_i)\right]\left[\sum_{\lambda} P(\lambda) EN_T(a-\lambda, b^*-\lambda)\right]\right\} \qquad (6)$$

If $P_3^a(b_i)$ were known, (6) would constitute a linear difference equation in the unknown $EN_T(a,b)$. Standard solution methods could then

be used to obtain a tight bound on $EN_T(a,-\infty)$, the amount of computation necessary for stack encoding. Unfortunately, $P_3^a(b_i)$ is itself a solution to the non-linear difference equation. In fact, let

$$G(a,x) = \text{Prob}\left(\begin{array}{l}\text{In Process 2 with } b = -\infty, \text{ the}\\ \text{top of the stack falls below}\\ \text{the value } x\end{array}\right) \tag{7}$$

Then

$$G(a,x) = \sum_{a>\lambda\geq b} P(\lambda)\, G^d(a-\lambda, b-\lambda) + \sum_{\lambda<b} P(\lambda) \quad . \tag{8}$$

and $P_3^a(b_i)$ is related to $G(a,x)$ by

$$G(a,x) = \sum_{b_i \leq x} P_3^a(b_i) \tag{9}$$

We do not know how to solve (8), except numerically, In the near future, we will do just that, and we shall apply the result to (6) so as to gain a better feeling about the behavior of $EN_T(a,-\infty)$.

III-A-3. Another Tree Encoding Algorithm and a New Source Coding Theorem

In this section, we describe a source encoding algorithm for use with tree codes. Tree searching does not proceed in a stack manner as in the preceding section, but instead uses two lists of temporary path hypotheses.

Assume code words for encoding a binary digit IID source have been arranged in a tree structure. The tree has rate $R = \log_2 d/n$, with d branches stemming from each node and n source approximating binary digits on each branch. The object of the encoder is to find a path of branches through the tree, the digits of which approximate the source sufficiently closely. To measure distance between the source output and various paths, we use the Hamming measure

$$d(\underset{\sim}{z}^{\ell}, \hat{\underset{\sim}{z}}^{\ell}) = \sum_{i=1}^{\ell} \left[1 - \delta(z_i, \hat{z}_i)\right] \tag{1}$$

z^{ℓ} is a source sequence, $\hat{z}^{\ell}$ is an hypothesized path, and δ is the Kronecker delta function.

The encoder operates with two lists of tree path hypotheses in arriving at one path for release to the user. The main list functions as a temporary "scratch pad," and the auxiliary list is a repository for "good" paths. Goodness of paths in these lists is judged by a path metric that depends on path length as well as distortion,

$$\mu(\hat{z}^{\ell}) = \ell D^* - d(z^{\ell}, \hat{z}^{\ell}) \tag{2}$$

Here ℓ is the length of z^{ℓ} (note that ℓ must be a multiple of n). D^* is the distortion per encoded source digit desired at the end of encoding, and $D^* > \Delta(R)$, the inverse rate distortion function relative to (1) and the source. Eqn. (2) is justified in earlier reports on the Jelinek stack encoder.

With this path metric in mind, we define two freezing barriers, one at metric $\underline{a} \gtrsim 0$, the other at $\underline{b} < 0$. Further extension of paths whose metrics rise above $\underline{a}$ will be frozen temporarily and the paths removed to the auxiliary list. Paths falling below $\underline{b}$ normally will be dropped forever -- "permanently" frozen.

Specifically, the algorithm works as follows:

Step (1) Starting at the tree root node (which is assigned the metric zero), paths are extended in the main list until all root node descendants crash a freezing barrier and are frozen. Paths which rise above the $\underline{a}$ barrier are placed in the auxiliary list in order of their length, the longest being on top. The longest of paths frozen at the b-barrier is also saved.

Step (2) When no paths remain in the main list, attention turns to the auxiliary list. In this "good" list, the final node of the longest path (which is on top of the list) now becomes a new root node (metric value 0 is assigned to it) for the main list, and the encoder executes again Step (1). The rest of the auxiliary list is retained and a-barrier crashing paths keep being added to it in the proper order.

Step (3) If there are no paths in the auxiliary list by the end of some execution of Step (1), the saved longest path frozen at the b-barrier is chosen to supply the new root node and again metric value 0 is assigned to it. The encoder then executes Step (1) again.

Definite encoding of the source sequence takes place whenever step (3) is involved, since only one path is then left. Some stopping rule must also be specified that will go into effect if the time elapsed since the last invocation of Step (3) is large (as hopefully happens often).

The analysis of this algorithm is an interesting one. However, the scheme has two practical advantages: if b is not too negative (which it need not be if D* is not too close to A(R)) then the main list can be allowed to be quite small. Also, the stack algorithm described in the preceding two sections has a start-up problem which is mostly avoided here: when encoding takes place there, only a single root node is provided and the paths that emerge from it might all approximate the source sequence quite badly.

To analyze and understand the two-list encoder better, we can view Steps (1) and (2) in terms of a branching process. In the language of Feller,[5] Pg. 293, let the paths that are frozen at a during Step (1) be the particles of the branching process, so that the auxiliary list is actually a list of untried progeny. With each particle associate also the main list computation to follow. Corresponding to the tree root node and the first execution of Step (1) is the branching process's initial particle, and paths which now crash the a-barrier become the first generation of particles. The first generation gives rise to the second according to some probability distribution independent from particle to particle and determined by the statistics of the main list. We can think of the succeeding progeny as occurring in generations, even though the encoder does not necessarily exhaust all auxiliary list "particles" on one generation before going to the next. The branching process either terminates by extinction of progeny, or goes on forever. In the former instance, Step (3) is invoked to start a new process.

Our analysis begins by finding the average computation necessary in the main list. We assume both lists are of infinite length, so that the parameters of interest are the freezing barriers (a,b) and the hoped for distortion D*. It concludes by using the branching process analogy

to prove the encoder can achieve any distortion $D^* > \Delta(R)$, so long as $\underline{b}$ is less than some b^* which depends on D^* and $\underline{a}$.

Main List Computation Related to (a,b), D*, and R

In the main list, let

N_a = Number of paths frozen at a-barrier

N_b = Number of paths frozen at b-barrier

N_∞ = Number of paths remaining forever unfrozen

We state immediately, but without proof, that the expected value of N_∞ is zero under proper conditions:

Theorem 1 For a tree of rate $R = \log_2 d/n$ used to encode a binary IID source with respect to the Hamming distortion measure.

$$|b - a| < \pi/\omega \quad \text{implies} \quad EN_\infty = 0 \ ,$$

where $\omega = \omega(R,D^*)$ and $\omega > 0$ for all $D^* \in (\Delta(R), 1/2)$

(The proof follows from difference equation methods explored first by Zigangirov[6] in a sequential decoding context. The function $\omega(R,D^*)$ is made specific in Appendix 5).

Assuming $EN_\infty = 0$, the expected number of main list paths frozen at the end of Step (1), EN, is

$$EN = E[N_a + N_b] \tag{3}$$

A short derivation shows that the expected number of extended branches present in a tree containing EN paths is related by

$$E[\text{branches}] = \left(\frac{EN - 1}{d - 1}\right) d \tag{4}$$

Customarily, a "Computation" is meant to include the scrutiny of d branches from their common parent node, so that

$$E[\text{Comps}] = \frac{EN - 1}{d - 1} \tag{5}$$

Eqns. (3), (4), and (5) measure in various ways the work done in the main list.

It remains to estimate EN_a and EN_b. Between these, EN_a is of crucial importance to a coding theorem because it corresponds to the expected number of descendants of each particle in the analogous branching process. Parts of the following proof are inspired by ideas used by Gallager,[7] again in sequential decoding analyses. The proof appears in the Appendix.

Theorem 2 Under the hypotheses of Theorem 1, $|b - a| < \pi/\omega$ implies

$$EN_a \doteq \frac{r^{-a}}{\sin \omega_a} \Bigg/ \left(\frac{\cos \omega_a}{\sin \omega_a} - \frac{\cos \omega_b}{\sin \omega_b} \right) \tag{6a}$$

$$EN_b \doteq \frac{r^{-b}}{\sin \omega_b} \Bigg/ \left(\frac{\cos \omega_a}{\sin \omega_a} - \frac{\cos \omega_b}{\sin \omega_b} \right) \tag{6b}$$

ω and r are functions of D^* and R, and are found as shown in Appendix 5. $\omega \searrow 0$ as $D^* \searrow \Delta(R)$, and r is typically near $(1-D^*)/D^*$. A careful look at (6) reveals that as $|b - a|$ tends to π/ω, both EN_a and EN_b tend to infinity. In fact, given an $\underline{a}$ one may choose b to make the right hand side of (6a) precisely unity. In this way, R; D^*, and $\underline{a}$, with the aid of Theorem 3, specify a minimal b necessary for the encoder to achieve D^*. In preparation for Theorem 3, we restate this as a

Corollary For any given $a < \pi/\omega$, there exists b^* such that if $|b-a| < \pi/\omega$ and $b < b^*$, then $EN_a > 1$.

We feel confident that further information about N is available from these methods. For instance, higher moments of N may be found in a way similar to the proof of Theorem 1.

Sample calculations have been made for $R = 1/2$ and D^* either 0.125 or 0.111. $\Delta(1/2)$ is 0.110.

	$D^* = 0.125$	$D^* = 0.110$
ω	0.789	0.206
r	6.46	7.98
π/ω	3.98	15.25

Table 1
Sample Values of r & ω

		$D^* = 0.111$		$D^* = 0.125$	
a	b	EN_a	EN_b	EN_a	EN_b
0.5	- 2	0.288	13.4	0.409	17.1
0.5	- 3	0.310	80.4	0.746	280
0.5	-14.5	1.06	2.4×10^{13}	∞	∞
0.5	-15.25	∞	∞	∞	∞
0.25	- 3.0			0.805	96.5
0.25	- 3.5	.....		1.28	737
0.25	- 3.73			∞	∞
0.17	- 3	0.669	29.9		
0.17	-14.5	0.921	3.5×10^{12}	.....	
0.17	-15.08	∞	∞		

Table 2

EN_a and EN_b vs. a, b, and D^*

Coding Theorem Proof Using the Two-List Encoder

We now prove the source coding theorem for our present source and distortion measure using the Two-List Encoder -- that is, we show the encoder can achieve any distortion greater than $\Delta(R)$. The proof uses the fundamental theorem of branching processes (see Feller[5], Pg. 297), with the branching process analogous to the encoder.

Theorem 3 Under the hypotheses of Theorem 1, whenever $EN_a > 1$ the expected per source digit distortion produced by the Two-List Encoder is at worst D^*, for any $D^* > \Delta(R)$.

Proof: Let an encoder cycle run from the extension of a root node to the invocation of Step (3). If the longest b-crashing path is of length L then the total distortion for this cycle is $LD^* - b$. Let $N(M)$ be the number of cycles it takes to encode a source sequence of length M [the last of these cycles may be completed at some sequence length that exceeds M]. The distortion per source digit D_M is then upper bounded by

$$D_M \leq D^* - \frac{N(M)}{M} b$$

so that the expected distortion is

$$E[D_M] \leq D^* - \frac{b}{M} E[N(M)] \leq D^* - \frac{b}{M} E[N(\infty)] .$$

But by the fundamental theorem of branching processes, $EN_a > 1$ implies that a cycle ends with infinite progeny (i.e. never ends) with probability $\eta > 0$.

Hence

$$E[N(\infty)] = \sum_{k=1}^{\infty} k(1-\eta)^{k-1} \eta = \frac{1}{\eta}$$

and

$$E[D_M] \leq D^* - \frac{b}{\sqrt{M}}$$

The theorem is thus proven by taking the limit as $M \rightarrow \infty$. QED

Theorem 3 contains no necessity for a large $\underline{a}$, so that a sensible encoder would place $\underline{a}$ as close to zero as possible. With this in mind, sample calculations were carried out for $R = 1/2$ and $D^* = 0.125$. For a code chosen at random from the usual random ensemble $EN \approx 40$, and b is required to be about -3.1. If D^* is lowered to 0.111 (very near $\Delta(R)$!), $EN \approx 10^{12}$ and $b \approx -14.7$.

The large literature on branching processes suggests more results can be obtained by these methods. We hope in the near future to obtain results concerning the auxiliary list size and the computation per encoded source digit needed in the main list.

The theorem is readily generalizable to other finite distortion discrete memoryless sources.

III-B. Permutation Codes as Source Codes

Permutation codes are a class of codes originally described by Slepian[8] for use as a method of achieving reliable transmission of digital data over an additive Gaussian noise channel. One variation of these codes was considered by Dunn[9-10] for the vector quantization of data from a time discrete, Gaussian, memoryless source. In this study, we have extended the work of Dunn in several ways: namely, (1) optimizing the parameters of the codes, (2) considering a second variation of the codes, (3) developing an efficient encoding algorithm for the codes, and (4) deriving some special properties associated with the codes.

The basic idea is that of block coding (or block quantizing) for a time discrete source. The source is thought to emit a sequence of statistically independent, identically distributed, random variables $x_1 x_2 \ldots$ each of zero mean and variance σ^2. We will be concerned with encoding the first N symbols, $\underline{x}^{(N)} = (X_1, X_2, \ldots, X_N)$. A set of M N-vectors, $\underline{C}_i^{(N)}$, $i = 1,2,\ldots,M$, are chosen as code words and the source output vector is represented by the closest (in accordance with some distortion measure) codeword. The rate of the code is defined to be

$$R = \frac{\log_2 M}{N} \tag{1}$$

and the resulting average distortion D is defined as

$$D = \frac{1}{N} E \left\{ \min_i d(\underline{X}^{(N)}, \underline{C}_i^{(N)}) \right\} \tag{2}$$

where $d(\underline{X}^{(N)}, \underline{C}_i^{(N)})$ is the distortion between the source vector $\underline{X}^{(N)}$ and the i^{th} codeword $\underline{C}_i^{(N)}$.

Permutation codes are codes for which the M codewords are chosen in a particular manner. Two different types of codes are considered and

are termed Variant I and Variant II codes as in Slepian. Their descriptions follow:

Variant I Codes: Let the first codeword $\underline{c}_1^{(N)}$ be chosen as the N-vector

$$\underline{c}_1^{(N)} = \underbrace{\mu_1,\ldots,\mu_1}_{n_1},\underbrace{\mu_2,\ldots,\mu_2}_{n_2},\ldots \quad , \quad \underbrace{\mu_k,\ldots,\mu_k}_{n_k} \tag{3}$$

where $\mu_1,\mu_2,\ldots,\mu_k$ are k real numbers such that

$$\mu_1 > \mu_2 > \quad > \mu_k$$

and

$$n_1 + n_2 + \ldots + n_k = N \quad , \tag{4}$$

where the n_i are positive integers. The other words of the code are chosen as all distinct permutations of the elements of the first codeword. There are a total of

$$M = N! \Big/ \prod_{i=1}^{K} n_i ! \tag{5}$$

codewords.

Variant II Codes: The first codeword $\underline{c}_1^{(N)}$ is again given as the N-vector

$$\underline{c}_1^{(N)} = \underbrace{\mu_1,\ldots,\mu_1}_{n_1},\underbrace{\mu_2,\ldots,\mu_2}_{n_2},\ldots \quad , \quad \underbrace{\mu_k,\ldots,\mu_k}_{n_k} \tag{6}$$

where now the μ_i are k nonnegative numbers such that

$$\mu_1 > \mu_2 \ldots\ > \mu_k \geq 0$$

The other words of the code are chosen by assigning a sign (positive or negative) to each component of the first codeword and by permuting these signed components in all possible ways. The number of codewords in the

code is now:

$$M = 2^{h} N! \Big/ \prod_{i=1}^{K} n_i ! \tag{7}$$

where

$$h = \begin{cases} N & \mu_k > 0 \\ N-n_k & \mu_k = 0 \end{cases} \tag{8}$$

The encoding procedure for block codes is in general a very complex procedure. In its worst form, each source output vector $\underline{X}^{(N)}$ must be compared with each of the M codewords $\underline{C}_i^{(N)}$, $i = 1,2,\ldots,M$ and is then represented by that codeword which attains the minimum distortion. For very large M this is a horrendous task. Permutation codes are of particular interest in that they lead to a relatively easy encoding algorithm for distortion measures of the form

$$d(\underline{X}^{(N)},\underline{Y}^{(N)}) = \sum_{i=1}^{N} F(|X_i - Y_i|) \tag{9}$$

where $f(|\alpha|)$ is a nonnegative, monotonic, nondecreasing, convex upward function of $|\alpha|$ and Y_i is the i^{th} component of the vector chosen to represent $\underline{X}^{(N)}$. The encoding algorithm which encodes $\underline{X}^{(N)}$ into the code vector which minimizes the distortion is described below for Variant I and Variant II codes. The proof that this encoding algorithm minimizes the resultant distortion is given in Appendix 6, part A.

Encoding Variant I Codes:

1. Replace the n_1 largest components of $\underline{X}^{(N)}$ with μ_1

2. Replace the next n_2 largest components of $\underline{X}^{(N)}$ with μ_2

...

K. Replace the smallest n_k components of $\underline{X}^{(N)}$ with μ_k. The result is a permutation of the codeword given in Equation (3) and is indeed a codeword in the code.

Encoding Variant II Codes:

1. Replace the n_1 largest, in absolute value, components of $\underline{X}^{(N)}$ with either $+\mu_1$ or $-\mu_1$, the sign chosen to agree with the sign of the component it replaced

2. Replace the next n_2 largest, in absolute value, components of $\underline{X}^{(N)}$ with either $+\mu_2$ or $-\mu_2$, again the sign chosen to agree with the sign of the component it replaced.

...

K. Replace the n_k smallest in absolute value components of $\underline{X}^{(N)}$ with either $+\mu_k$ or $-\mu_k$, again the sign chosen to agree with the sign of the component it replaced.

It should be noted that for identically distributed, statistically independent source outputs, all codewords for the Variant I codes are equally probable. If the source distribution is symmetric about zero the same is true for Variant II codes.

A commonly used distortion measure is "mean-square error" distribution whereby $f(|\alpha|)$ of Equation (9) is given by

$$f(|\alpha|) = \alpha^2 \tag{10}$$

It is shown in Appendix 6, part B that for a given choice of $n_1, n_2, \ldots, n_k$, the best choice of the parameters $\mu_1, \mu_2, \ldots, \mu_k$ in the sense of minimizing the mean-square error is given by the following equations:

Variant I

$$\mu_j = \frac{1}{n_j} \sum_{i=n_1+n_2+\ldots+n_{j-1}+1}^{n_1+n_2+\ldots+n_j} E\left\{X^{(i)}\right\} \qquad j = 1,2,\ldots,K \quad (11)$$

where $E\left\{X^{(i)}\right\}$ is the expected value of the i^{th} largest of N independent random variables: i.e., $X^{(1)} \geq X^{(2)} \geq \ldots \geq X^{(N)}$.

Variant II

$$\mu_j = \frac{1}{n_j} \sum_{i=n_1+n_2+\ldots+n_{j-1}+1}^{n_1+n_2+\ldots+n_j} E\left\{|X|^{(i)}\right\} \quad (12)$$

where $E\left\{|X|^{(i)}\right\}$ is the expected value of the absolute value of the i^{th} largest of N random variables. That is, the absolute value of N random variables are ordered in terms of their magnitudes and $E\left\{|X|^{(i)}\right\}$ is the expectation of the i^{th} largest. For a mean-square error distortion measure, and for the choice of μ_j given by Equations (11) and (12) the resulting average distortion is given as

$$D = \sigma^2 - \frac{1}{N} \sum_{j=1}^{K} n_j \mu_j^2 . \quad (13)$$

The rate of a permutation code for a given N is a function of the choice of the groupings $n_1, n_2, \ldots, n_k$. The highest rate codes occur for $n_i = 1$, for $i = 1,2,\ldots, K = N$. (For Variant II codes, in order

to achieve the largest rate we have the added restriction that $\mu_k > 0$.)
The maximum rate is

$$R_{MAX} = \begin{cases} \dfrac{\log_2 N!}{N} & \text{Variant I} \\ 1 + \dfrac{\log_2 N!}{N} & \text{Variant II} \end{cases} \tag{14}$$

and the corresponding mean-square error distortion is

$$D = \begin{cases} \sigma^2 - \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} E\left\{X^{(i)}\right\}^2 \\ \sigma^2 - \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} E\left\{|X|^{(i)}\right\}^2 \end{cases} \tag{15}$$

For a Gaussian source with unit variance, the summation

$$\sum_{i=1}^{N} E\left\{X^{(i)}\right\}^2$$

is tabulated by David et al[11] for values of N up to 400 . The resulting distortion for maximum rate Variant I codes is found to be much greater than the corresponding distortion given by Rate-Distortion theory;[12] namely

$$D = \sigma^2 \, 2^{-2R} \tag{16}$$

In fact, the resulting performance is inferior to that of an ordinary scalar quantizer with 2^R equally spaced quantization levels. Thus we see that if such codes are to be of value, we must have a method for the judicious choice of the groupings $n_1, n_2, \ldots, n_k$.

Many different choices of groupinsg $n_1,\ldots,n_k$ exist which result in the same rate $R < R_{MAX}$. (In fact any permutation of the values of $n_1,\ldots,n_k$ yields the same rate.) The optimum choice of $n_1,n_2,\ldots,n_k$ and k for a given rate will be that set of parameters which yields the minimum distortion. For a given K, it is shown in Appendix C that a necessary condition on the choice of $n_1,n_2,\ldots,n_k$ to yield the minimum distortion is that $n_1 \leq n_2 \leq n_3 \leq \ldots \leq n_k$ for Variant II codes and that $n_1 \leq n_2 \leq n_3 \leq \ldots \leq n_i$ where $E\left\{X^{(n_1+\ldots+n_i)}\right\} \geq 0$ for Variant I codes.

The following approach was used in a computer optimization procedure to find the best values of $n_1,n_2,\ldots,n_k$ and k for a Gaussian source. Several approximations were used in this algorithm so the resulting parameters may not be truly optimum. However, there is reason to expect that the performance of the codes obtained from thus algorithm is essentially that of the very best codes. The procedure is based upon the following observations. Define

$$p_i = n_i/N \quad , \quad i = 1,2,\ldots,k \tag{17}$$

Then, for large n_i and N, the rate R can be written approximately as:

$$R \approx \begin{cases} -\sum_{i=1}^{k} p_i \log_2 p_i & \text{Variant I} \\ 1 - \sum_{i=1}^{k} p_i \log_2 p_i & \text{Variant II} \end{cases} \tag{18}$$

Furthermore the distortion (mean-squared error) is given exactly as

$$D = 1 - \sum_{i=1}^{k} p_i \mu_i^2 \tag{19}$$

Treating (18) as an equality we can minimize D with respect to $p_1, p_2, \ldots, p_k$ subject to the rate constraint. The optimum p_i are given as

$$p_i = \frac{2^{-\beta\mu_i^2}}{\sum_{j=1}^{k} 2^{-\beta\mu_j^2}} \tag{20}$$

where β is chosen so that (18) is satisfied. Note that in actuality we do not have an analytic solution for the best n_i for two reasons. First, $n_i = p_i N$ may not be an integer and second, p_i is given in terms of the μ_j which are, in turn, functions of the groupings n_i. Furthermore the above solution assumes that k is known while we would also want to find the best k.

The flow diagram for the computer algorithm used in finding good codes is shown in Figure 4. A rough outline of this algorithm can be found in Appendix 6, part D.

As an example, for $N = 400$, $R \doteq 1.5$, K odd, the groupings obtained is

$$n_1=1 \quad n_2=4 \quad n_3=74 \quad n_4=242 \quad n_5=74 \quad n_6=4 \quad n_7=1$$

The resulting rate is $R = 1.47514$, and distortion is $D = 0.18595$.

The Gaussian order statistics required for Variant I codes were taken from the table of David et al.[11] The results of this computer optimization for Variant I codes with $N = 400$ are plotted in Figure 5. (A smooth curve has been drawn through the resultant R-D points,) Also plotted on this graph are

1) The rate-distortion curve for the Gaussian independent source: as given by Equation 16

2) Several points corresponding to optimum scalar quantizers. The quantization regions and representation points have been optimized to yield the smallest mean-squared error for that number of representation points. The rate of the uncoded quantizer is $\log_2$ (number of quantization points). The coded quantizer's rate is the entropy of the representation points. See Lloyd[13] or Max[14].

3) The performance of a uniform quantizer whose spacing is optimized and whose outputs are then Huffman coded.[15]

4) The performance of some Variant I, N = 400 found by Dunn.

In conjunction with this figure we see that:

a) The N = 400, Variant I codes are superior to Lloyd-Max uncoded quantizers for $R < 3.7$ and are superior to Lloyd-Max coded quantizers for $R < 3.2$. Their performance is approximately equal to that of the uniform quantizer (coded and optimized) over the range $1 \lesssim R < 2.7$.

b) For small rates ($R < 1$), the performance of the Variant I codes approach that of the rate-distortion curve. The highest rate code plotted in this figure corresponds to the grouping $n_1 = 1$, $n_2 = N-1$. This code is a simplex code and its rate and corresponding distortion is given as

$$R = \frac{\log_2 N}{N} \tag{21}$$

$$\frac{D}{\sigma^2} = 1 - \left(E\left\{X^{(1)}\right\}\right)^2 \Big/ N-1 \tag{22}$$

where, here, $E\left\{X^{(1)}\right\}$ is the expectation of the largest of N Gaussian

random variables of zero mean and unit variance. For very large N, Gumbel[16] has shown that

$$E\left\{X^{(1)}\right\} \cong \sqrt{2 \ln N} + C \tag{23}$$

Combining Equations (21), (22) and (23) we have, for large N

$$\frac{D}{\sigma^2} \approx 1 - 2R \ln 2 \tag{24}$$

Comparing Equations (24) and (16) we see that the two agree for small values of R. Thus, the simplex Variant I codes are asymptotically optimum for large N. Furthermore it is easily shown that the best quantizer which has two representation points for N outputs from a Gaussian, independent source behaves as

$$\frac{D}{\sigma^2} = 1 - \frac{2}{\pi} R \ln 2 \tag{25}$$

Thus, this type of quantizer is not asymptotically optimum.

c) The codes obtained by Dunn are not quite as good as the codes found by the computer optimization procedure. In particular, the following two codes are easily compared:

Dunn: $\bar{n} = (5, 5, 35, 40, 65, 100, 65, 40, 35, 5, 5)$

$R = 2.86367$

$D = 0.03389$

Computer: $\bar{n} = (1, 2, 7, 20, 46, 77, 94, 77, 46, 20, 7, 2, 1)$

$R = 2.79184$

$D = 0.03362$

The computer generated code achieves essentially the same distortion as the code of Dunn but at a reduced rate.

The evaluation of Variant II codes was hampered by the unavailability of tables for the expected value of absolute Gaussian order statistics, $E\left\{|X|^{(i)}\right\}$. The only tables found were those of Klatz[17] which gives these statistics only for $N \leq 10$. It was reasonably simple to evaluate the performance of all groupings for small N . The results for $N = 10$ are plotted in Figure 6.

It is difficult to draw conclusions on the efficacy of Variant II codes from the present data since we need to evaluate the performance of these codes for large values of N . A computer program is presently being written to obtain the expected value of absolute Gaussian order statistics for large values of N .

Two interesting properties of Variant II codes follow.

1. For any N , if we choose only one grouping (i.e., $n_1 = N$), then the representation points are located on the vertices of a hypercube with coordinates $\left(\pm\sqrt{\frac{2}{\pi}}, \pm\sqrt{\frac{2}{\pi}}, \ldots, \pm\sqrt{\frac{2}{\pi}}\right)$. The representation points and performance of this code are identical to those of an optimum 1 bit single sample quantizer.

2. For $N = 2$ and $n_1 = n_2 = 1$, the eight representation points are uniformly spaced on a circle of radius

$$4\sqrt{\frac{2}{\pi}} \sin\frac{\pi}{8}$$

Although appealing from the standpoint of symmetry, this configuration is a relatively poor quantizer with rate

$$R = 1.5 \text{ bits/sample}$$

and mean-square distortion

$$D = \frac{1}{4} - \frac{4}{\pi}\sin^2\frac{\pi}{8} \approx .23$$

These values are plotted as an asterisk on Figure 6.

The method of encoding described earlier for Variant I and Variant II codes assumed that the encoding process consisted of replacing the output of the source by its closest codeword. In actuality, for transmission over a communications channel (or for storage in a memory) one would order the codewords and then transmit (or store) the rank order of the appropriate codeword. We now give a method for achieving this. This method is similar to Jelinek's[18] version of the Elias variable length noiseless coding scheme.

The idea of this scheme is to map each of the $M = N! \Big/ \prod_{i=1}^{k} n_i!$ permutations of the vector

$$\mu_1,\ldots,\mu_1,\ \overset{n_1}{\mu_2,\ldots,\mu_2},\ldots\ ,\ \overset{n_2}{\mu_k,\ldots,\mu_k}$$

into a point on the real line in the interval (0,1). These points will be equally spaced and the mapping will be one-one. Then, various methods can be used to enumerate these points. In the method described later, each point is represented by its binary fraction expansion.

We now give the method to map the sequence

$$\underline{U}^{(N)} = \mu_{j_1}, \mu_{j_2},\ldots, \mu_{j_N}$$

onto the unit interval. Define the set of integers $I_\ell(i)$, $i = 1,2,\ldots,k$, $\ell = 1,2,\ldots,N$ as follows:

$$I_1(i) = n_i \qquad i = 1,2,\ldots,k \tag{26}$$

$$I_\ell(i) = \begin{cases} I_{\ell-1}(i) - 1 & j_{\ell-1} = i \\ I_{\ell-1}(i) & j_{\ell-1} \neq i \end{cases} \qquad \ell = 2,3,\ldots,N \tag{27}$$

and

$$I_{\ell}(0) = 0 \qquad \ell = 1,2,\ldots,N \quad . \tag{28}$$

The mapping of $\underline{U}^{(N)}$ onto the unit interval, $\pi(\underline{U}^{(N)})$ is then given as

$$\pi(\underline{U}^{(N)}) = \frac{1}{N}\sum_{i=0}^{j_1-1} I_1(i) + \frac{I_1(j_1)}{N(N-1)}\sum_{i=0}^{j_2-1} I_2(i) + \frac{I_1(j_1)I_2(j_2)}{N(N-1)(N-2)}\sum_{i=0}^{j_3-1} I_3(i)+\ldots$$

$$\ldots+\frac{1}{N}\prod_{\ell=1}^{N-2}\frac{I_{\ell}(j_{\ell})}{(N-\ell)}\sum_{i=0}^{j_{N-1}-1} I_{N-1}(i) + \frac{\prod_{i=1}^{k} n_i!}{N!} \tag{29}$$

With this procedure, the sequence

$$\mu_1,\mu_1,\ldots,\mu_1,\ \mu_2,\mu_2,\ldots,\ \mu_k,\mu_k,\ldots,\mu_k$$

maps to the point $\left(\prod_{i=1}^{k} n_i!\right)\Big/ N!$, the sequence

$$\mu_1,\mu_1,\ldots,\mu_1,\ \ldots\ ,\ \underbrace{\mu_{k-1},\mu_{k-1},\ldots,\mu_{k-1}}_{n_{k-1}-1},\ \mu_k,\mu_{k-1},\ \underbrace{\mu_k,\ldots,\mu_k}_{n_k-1}$$

maps to the point $2\left(\prod_{i=1}^{k} n_i!\right)\Big/ N!$, etc., the last sequence

$$\mu_k,\ \mu_k,\ldots,\mu_k,\ \mu_{k-1},\mu_{k-1},\ldots,\mu_{k-1},\ldots\ ,\ \mu_1,\mu_1,\ldots,\mu_1$$

mapping to the point 1 .

The binary codeword corresponding to $\underline{U}^{(N)}$ is derived as follows: Expand $\pi(\underline{U}^{(N)})$ into an infinite, unique, binary fraction

$$\pi(U^{(N)}) = s_1(U^{(N)})2^{-1} + S_2(U^{(N)})2^{-2} + \ldots + S_\ell(U^{(N)})2^{-\ell} + \ldots \quad (30)$$

where $S_j(U^{(N)}) \in (0,1)$ are chosen so that

$$0 \leq \left(\pi(U^{(N)}) - \sum_{j=1}^{i} s_j(\underline{U}^{(N)}).2^{-j}\right) < 2^{-i} \quad (31)$$

Let Q be the smallest integer greater than $\log_2 M$. The binary codeword $\Phi(\underline{U}^{(N)})$ corresponding to $\underline{U}^{(N)}$ is the sequence

$$\Phi(\underline{U}^{(N)}) = S_1(U^{(N)}), S_2(\underline{U}^{(N)}),\ldots,S_Q(U^{(N)}) \quad (32)$$

The codeword $\Phi(U^{(N)})$ specifies $\underline{U}^{(N)}$ uniquely since $\hat{\pi}(\underline{U}^{(N)})$ defined as

$$\hat{\pi}(U^{(N)}) = \sum_{j=1}^{Q} S_j(U^{(N)})\, 2^{-j} \quad (33)$$

falls in the half open interval $\left(\pi(U^{(N)}) - \frac{N!}{n_1! \ldots n_k!}, \pi(\underline{U}^{(N)})\right]$

An efficient decoding method is as follows. Rewrite (29) as

$$\pi(\underline{U}^{(N)}) = \frac{1}{N}\sum_{i=1}^{j_1} I_1(i) - \frac{1}{N} I_1(j_1)\left[1 - \frac{1}{N-1}\sum_{i=1}^{j_2} I_2(i)\right]$$

$$- \frac{1}{N(N-1)} I_1(j_1)\, I_2(j_2)\left[1 - \frac{1}{N-2}\sum_{i=1}^{j_3} I_3(i)\right] - \ldots$$

$$\ldots - \prod^{N-2} \frac{I_\alpha(j_\alpha)}{(N-\alpha+1)}\left[1 - \frac{1}{2}\sum^{j_{N-1}} I_{N-1}(i)\right] \quad (34)$$

In order to recover $\underline{U}^{(N)}$ from $\pi(\underline{U}^{(N)})$, the decoder follows the following recursive procedure:

$$j_1 = \min\left\{j : \frac{1}{N}\sum_{i=1}^{j} I_1(i) \geq \pi(\underline{U}^{(N)})\right\} \tag{35}$$

Knowledge of j_1 allows the decoder to compute $I_2(i)$ for $i = 1,2,\ldots,k$. Then

$$j_2 = \min\left\{j : \frac{1}{N}\sum_{i=1}^{j_1-1} I_1(i) + \frac{1}{N(N-1)} I_1(j_1)\sum_{i=1}^{j} I_2(i) \geq \hat{\pi}\,(\underline{U}^{(N)})\right\} \tag{36}$$

This continues until the next to last step where

$$j_{N-1} = \min\left\{j : \frac{1}{N}\sum_{i=1}^{j_1-1} I_1(i) + \frac{1}{N(N-1)}\, I_1(j_1)\sum_{i=1}^{j_2-1} I_2(i) + \ldots\right.$$

$$\left. + \frac{1}{N!}\, I_1(j_1)\, I_2(j_2)\cdots I_{N-2}(j_{N-2})\sum_{i=1}^{j} I_{N-1}(i) \geq \hat{\pi}\,(\underline{U}^{(N)})\right\} \tag{37}$$

The final step determines j_N as

$$j_N = \left\{j : I_N(j) = 1\right\} \tag{38}$$

APL-type encoding and decoding algorithms are given in Appendix 6, part E.

The following is a summary of the various steps required in a quantization scheme based upon permutation codes. For convenience, only Variant I codes are discussed.

1. The outputs of a source are subdivided into blocks of N symbols.

2. The positions of the n_1 largest samples, n_2 next largest samples,...,n_k smallest samples are noted.

3. This position vector is coded into binary digits by Equations (29),(30) and (32).

4. The binary digits are decoded into a position vector by the method described by Equations (35)-(38).

5. The representation vector is then obtained by placing in the largest n_1 positions, the real number μ_1, in the next largest n_2 positions, the real number μ_2, etc., and in the smallest n_k positions, the real number μ_k. If available, the values to be used for μ_i are those given by Equation (11). Alternatively, the encoder could collect the sample order statistics and transmit these numbers to the receiver at the end of the transmission. The receiver would then use these sample statistics as if they were the actual order statistics.

References to Part III

1. C.E. Shannon: "Coding Theorems for a Discrete Source with a Fidelity Criterion," IRE National Convention Record, Part 4, pp. 142-163, 1959.
2. C.E. Shannon: "A Mathematical Theory of Communication," Bell System Tel.J., 27, p. 379 and p. 623, 1948.
3. F. Jelinek: "Tree Encoding of Memoryless Time-Discrete Sources with a Fidelity Criterion," IEEE Transactions on Information Theory, IT-15, No. 5, Sept. 1969.
4. F. Jelinek: "A Fast Sequential Decoding Algorithm Utilizing a Stack," IBM Journal of Research and Development, 13, No. 6, Nov.1969.
5. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. I, 3rd Edition, Wiley, N.Y., 1968.
6. K. Sh. Zigangirov, "Some Sequential Decoding Procedures," Problemy Peredachi Informatsii, Vol.2, No. 4,pp. 13-25,1966.
7. R.G. Gallager, "Random Trees and Tree Codes," Unpublished memorandum, June 1970.
8. D. Slepian, "Permutation Modulation," Proceedings of the IEEE, vol. 53, pp. 228-236, 1965.
9. J.G. Dunn, "Coding for Continuous Sources and Channels," Ph.D. Dissertation, Columbia University, Department of Electrical Engineering, 1965.
10. J.G. Dunn, "The Performance of a Class of n-Dimensional Quantizers for a Gaussian Source," Symposium on Signal Transmission and Processing, Columbia University, May 13, 14, 1965, pp. 76-81.
11. F.N. David, D.E. Barton, S. Ganeshalingam, H.L. Harter, P.J. Kim, M. Merrington, and D. Walley, Normal Centroids, Medians and Scores for Ordinal Data, Cambridge University Press, Cambridge, 1968.

12. C.E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, pp. 379-423,623-656,1945.

13. S. Lloyd, "Least Square Quantization PCM," Internal Memorandum, Bell Telephone Laboratories, 1959.

14. J. Max, "Quantizing for Minimum Distortion," IEEE. Transactions on Information Theory, vol. IT-6, pp. 7-12, 1960.

15. T.J. Goblick,Jr. and J.L. Holsinger, "Analog Source Digitization: A Comparison of Theory and Practice," IEEE Trans. on Information Theory, Vol.IT-13, pp. 323-326,1967.

16. E.J. Gumbel, Statistics of Extremes, Columbia University Press, New York, 1958.

17. J. Klatz, "Small Sample Power and Efficiency for the One Sample Wilcoxon and Normal Scores Test," Annals of Mathematical Statistics, vol. 34, pp. 624-632,1963.

18. F. Jelinek, Probabilistic Information Theory, McGraw-Hill Book Co.,Inc., New York, 1968, pp.479-489.

Captions for Figures of Part III

Figure 1. An Example of a tree code.

Figure 2. Best average distortion achievable of near-optimal convolutional codes of constraint length 5, 7, 10, and 14.

Figure 3. Average distortion vs. number of encoding steps of the Viterbi encoding algorithm when used with the codes of Figure 1. Also plotted is the average distortion vs. average number of encoding steps of the Stack Algorithm when used with the code of constraint length 14 whose ultimate performance is given in Figure 2.

Figure 4. Flow chart for determining optimal groupings $\{n_1,\ldots,n_k\}$ for permutation codes.

Figure 5. Comparison of Variant I-type code performance with that achievable by quantizers and with the rate-distortion function for Gaussian sources.

Figure 6. Short Variant II-type code performance compared to that achievable by quantizers and to the rate-distortion function for Gaussian sources.

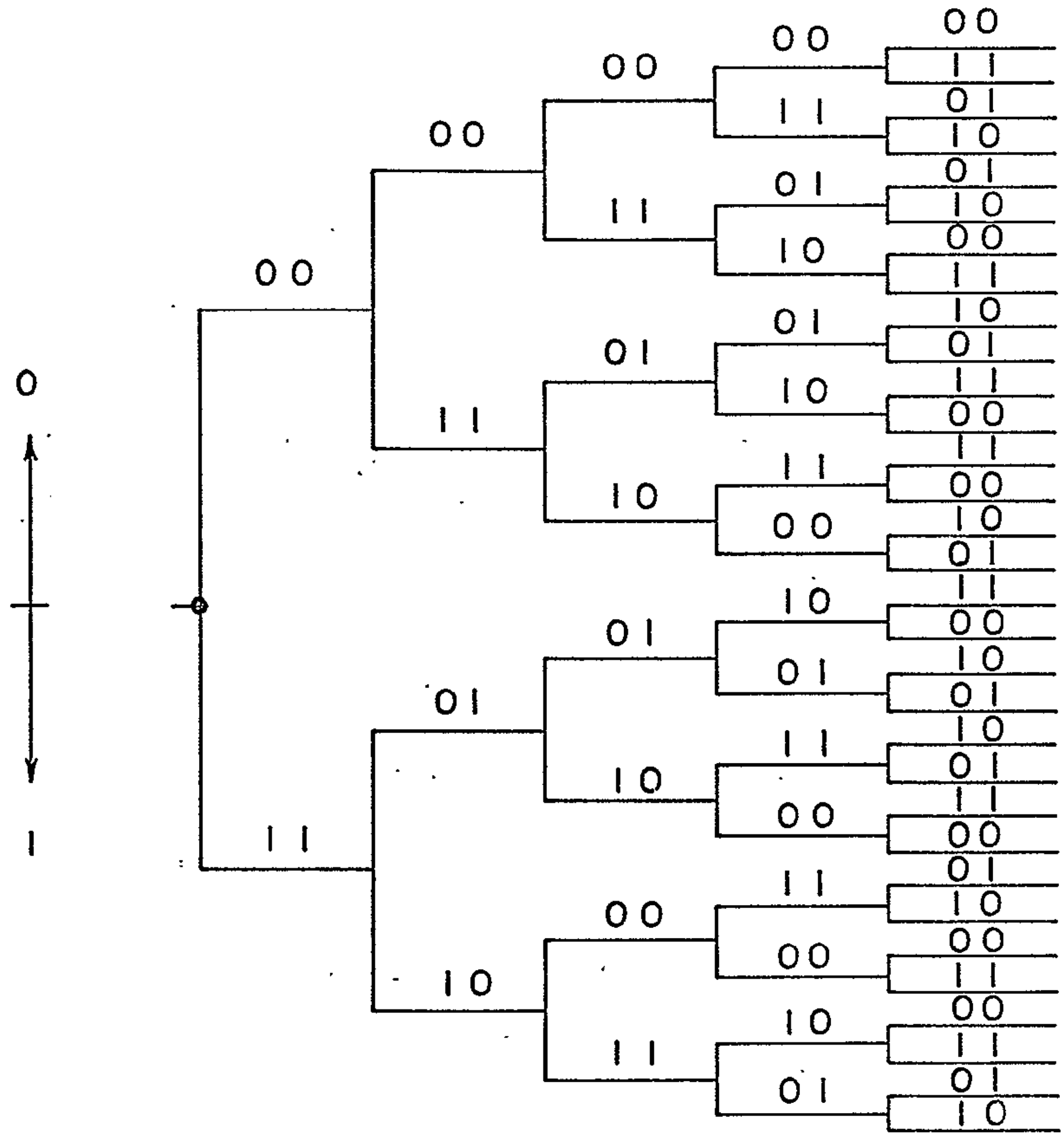


Figure 1: An Example of a tree code.

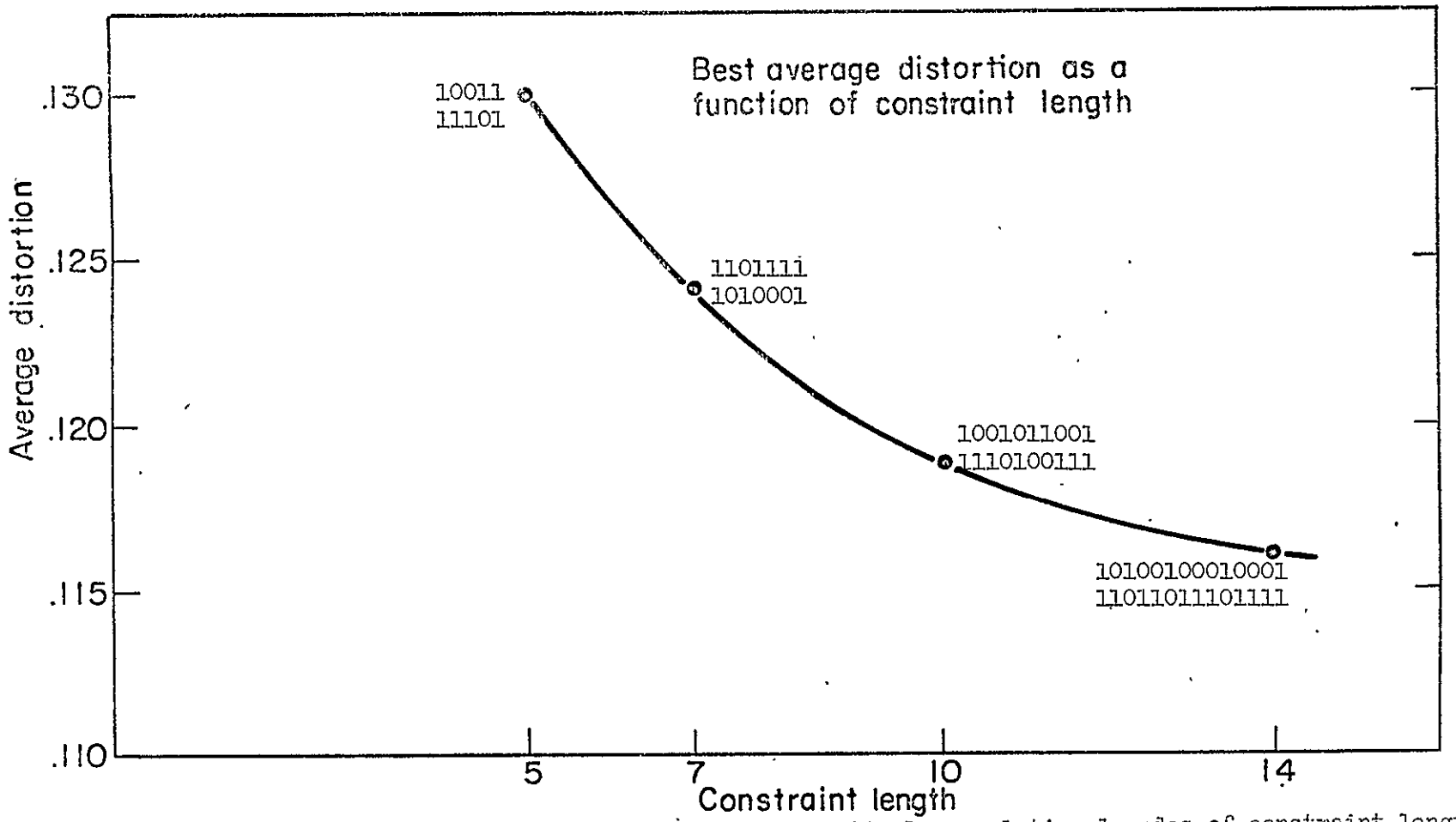


Figure 2: Best average distortion achievable of near-optimal convolutional codes of constraint length 5,7,10, and 14.

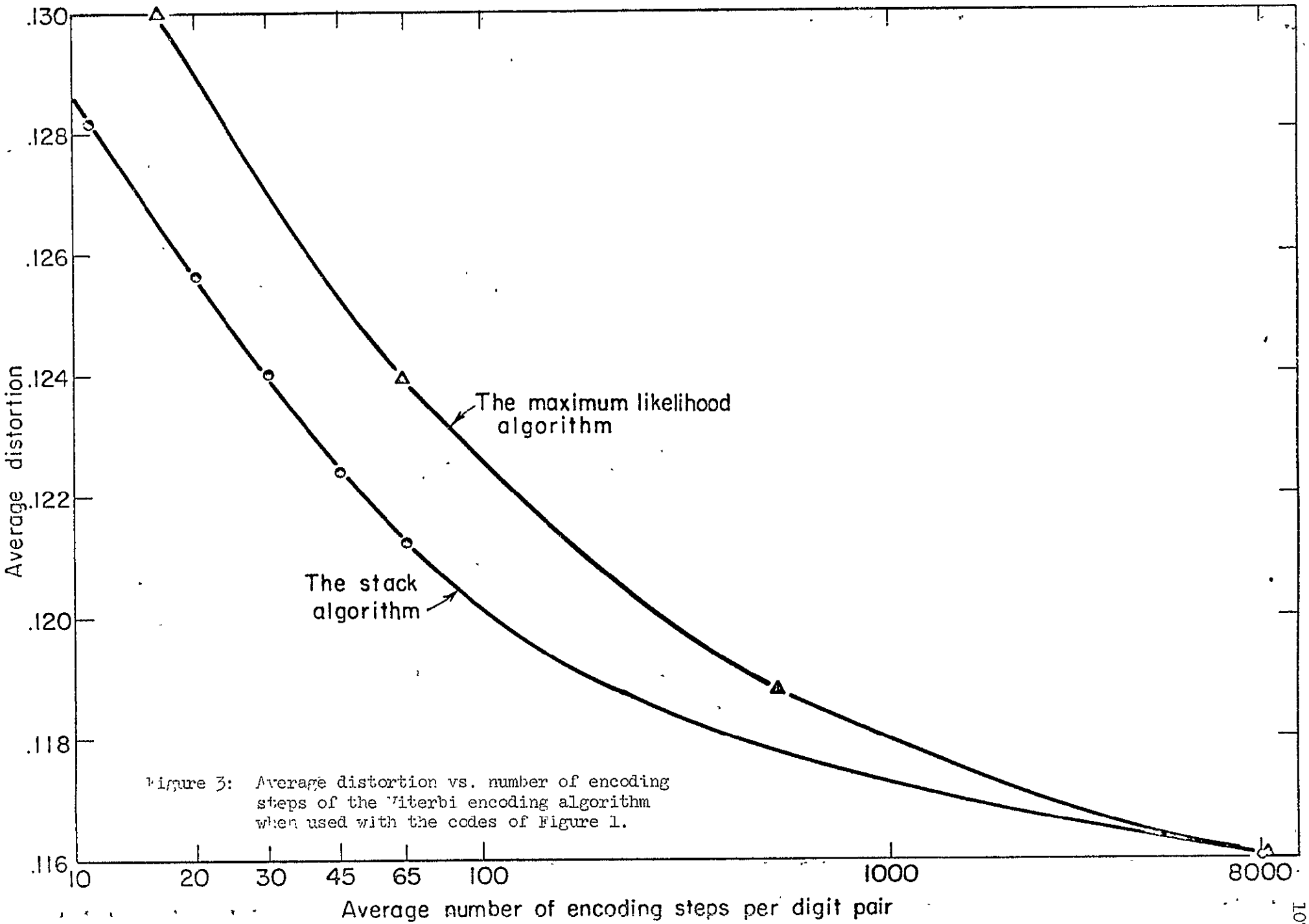
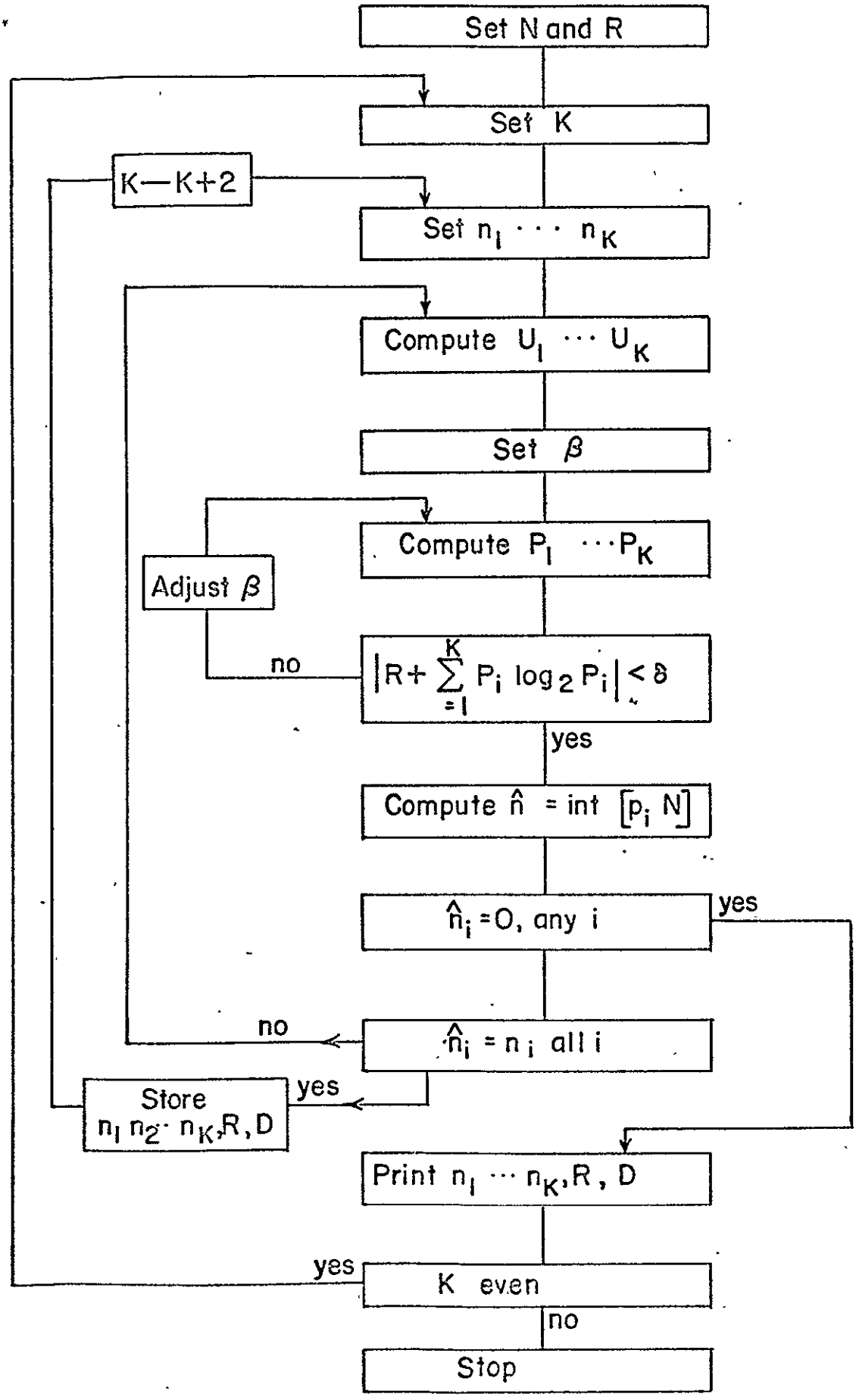


Figure 3: Average distortion vs. number of encoding steps of the Viterbi encoding algorithm when used with the codes of Figure 1.

Figure 3.

Figure 4: Flow chart for determining optimal groupings $\{n_1, \ldots, n_k\}$ for permutation codes.



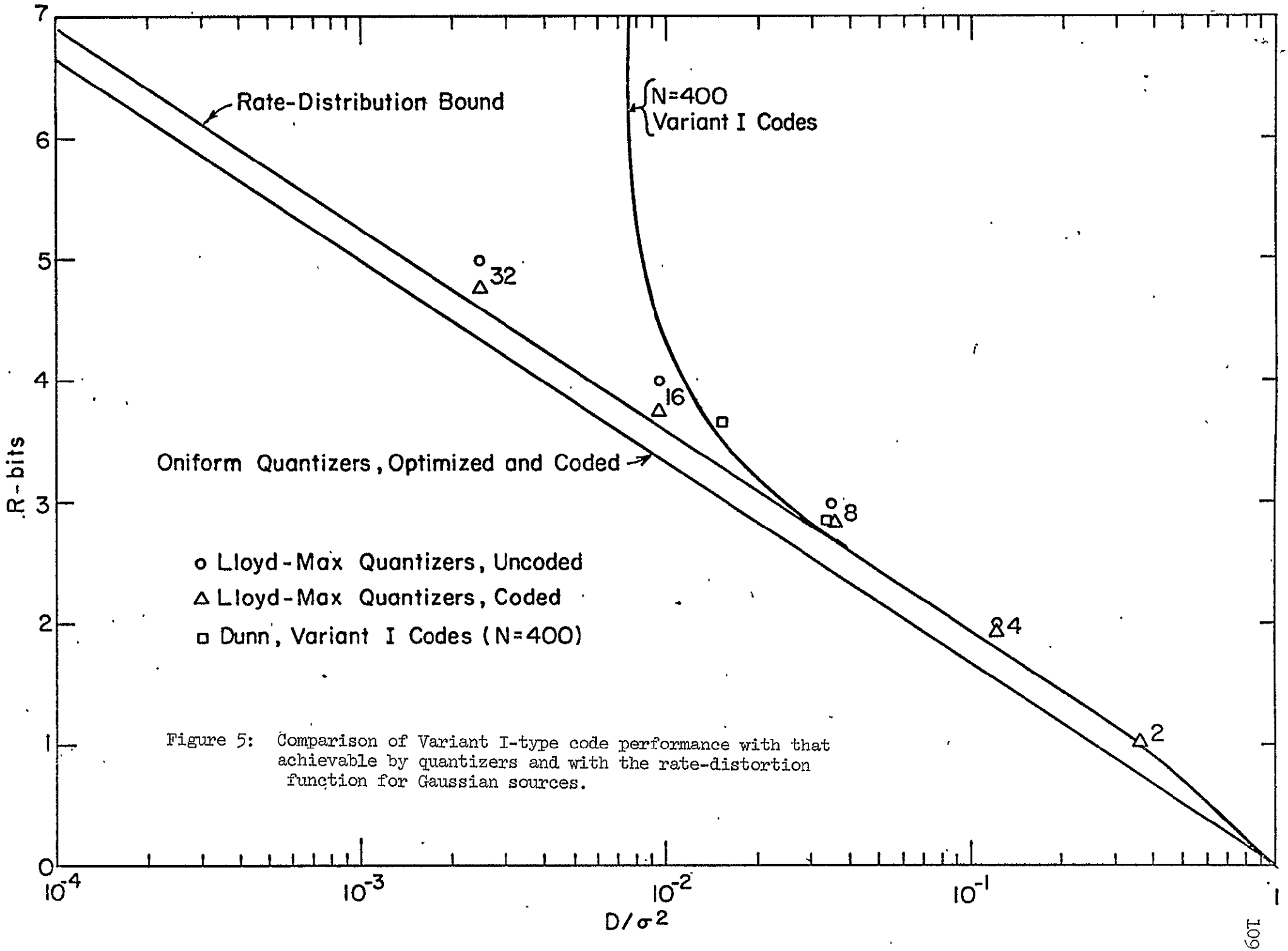


Figure 5: Comparison of Variant I-type code performance with that achievable by quantizers and with the rate-distortion function for Gaussian sources.

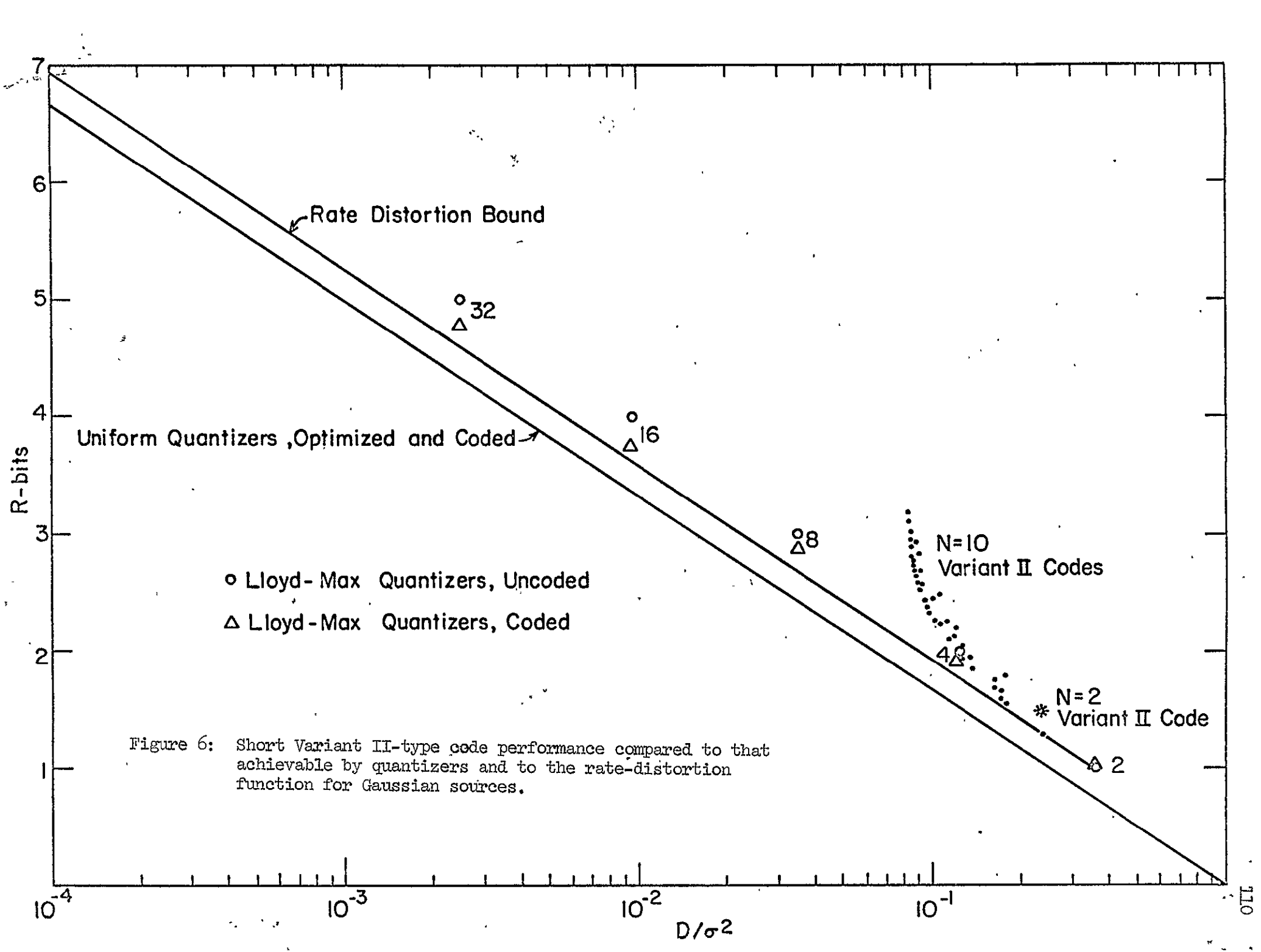


Figure 6: Short Variant II-type code performance compared to that achievable by quantizers and to the rate-distortion function for Gaussian sources.

APPENDIX 1

DETAILS OF STACK AND MAP MAINTENANCE AND PURGING

MANAGEMENT OF THE MAP OF VISITED NODES AND ITS PURGING

The following purging strategy will be based on the requirement that if there is any entry of depth I in the stack, the decoder has made a final decision about information digits of depth at least I-t, where t is some conveniently large integer exceeding the constraint length ν of the code (good rule of thumb seems to be $t \approx 3\nu$).

All variable names used in this description are those used in the FORTRAN stack decoding program. The operations outlined below are in addition to those already in that program.

1. Before a node is extended the decoder checks whether $0 \leq$ I-M1(NPOINT(N1)) $\leq t$. If not, the decoder sets NPOINT(N1) = 1 (the location of the root node is M1(1) = -L1MASK). Also, should I $\leq$ IMAX-t, the node is dropped from the stack and no extension is made.

2. At the beginning of the decoding process, we set M1(J) = -t, J = 2,3,...,IMAP, where IMAP is the number of locations in the map. There will be a pointer LOCPUR whose initial value is 2. When a new map entry is to be made corresponding to depth I, the decoder checks whether

$$I - M1(LOCPUR) \geq t + 1$$

If so, the entry is made into location LOCPUR. If not, then we increment LOCPUR by 1 and try again (when LOCPUR = IMAP, instead of incrementing, LOCPUR is set equal to 2). If the search has been unsuccessful for IMAP-1 tries, a map overflow is declared. One may stop at that point or take a risk and replace that entry whose M1(J) value is smallest. LOCPUR would then be set to J.

3. Decision Making

When a node is to be extended such that I = IMAX, we set IMAX = IMAX + 1 and make a decision on the node at depth IMAX-t = I+1-t. This is done as follows:

Set M1I = I, NPO1 = NPOINT(N1)

CASE I: M1(NPO1) = I+1-t

In this case the decision is a 1.

CASE II: NPO1 = 1 or ML(NPO1) > M1I. In this case the decision is a 0.

CASE III: Neither of the above. In this case set M1I = M1(NPO1) and NPO1 = MP(NPO1) and repeat above.

Argument why strategy works:

Because of 1, when the entry at location NPO1 was made then either MP(NPO1) = 1 or M1(NPO1) - M1(MP(NPO1)) $\leq$ t . In the first case, the value MP(NPO1) = 1 is either natural, or results from application of rule 1. In either case, at depths M1(NPO1)-1, M1(NPO1)-2,...,M1(NPO1)-t, the path has 0 branches only. Suppose the latter case is true. Then the entry at MP(NPO1) may be replaced only (see 2) by an entry whose value M1' satisfies M1' $\geq$ M1(MP(NPO1)) + t+1, i.e. such that M1' > M1(NPO1). The new "unnatural" entry will then be recognized by the decision procedure (as the instance M1(NPO1) > M1I of CASE II). Note from 2 and 3 that the replacement takes place when the decision about depth M1(MP(NPO1)) has already been taken. Thus when the stop of CASE II occurs, a decision is to be taken about a branch inside the depth interval (M1(MP(NPO1)), M1(NPO1)), and all such branches are 0's by definition.

Finally, if the stop of CASE I occurs, M1(NPO1) is the original entry at NPO1, and does correspond to a 1-branch.

The fact that either CASE I or II will eventually occur need not be labored.

Purging of Map when the depth of definitely decoded digits is not given by formula J-t where t is a fixed constant and J is the depth of furthest advance in the tree.

Note: A good decision method may be: whenever the likelihood on some path $\underset{\sim}{s}^J = s_1,\ldots,s_J$ exceeds a maximal threshold T, all digits $s_1,\ldots,s_k$ $(K < J)$ are decided such that the likelihoods $L(\underset{\sim}{s}^i) \leq T\text{-}a$, for $i = 1,2,\ldots,K$, where the value of a is chosen in some convenient manner.

Assume that in accordance with some decision strategy, all message digits at levels $0,1,2,\ldots,t$ are definitely decoded for some $t \geq 0$. We will create 2 new arrays: M2[IMAP] and NPTM1[KSTACK] (in addition to those arrays that are utilized in the original Fortran Stack Decoding Program). Their values will be

a) Initially: M1(1) = -L1MASK, NPOINT (1) = 1, NPTM1(1) = -L1MASK, M1(J) = -L1MASK for J = 1,...,IMAP.

b) Suppose a node at location N1 of depth I1 is being extended, the 1-extension goes into stack location N2, and the newly created map location will be J. Then we leave NPOINT(N1) and NPTM1(N1) as before. We set M1(J) = NPTM1[N2] = I1+1, MP(J) = NPOINT(N1), M2(J) = NPTM1[N1], NPOINT(N2) = J. As a result of the above strategy, as long as no map location is purged, we will always have

$$M2[J] = M1(MP(J)) \tag{1}$$

and

$$NPTM1(K) = M1(NPOINT(K)) \tag{2}$$

The relations (1) and (2) will then provide a check on pointer validity:

MAP PURGING:

When levels $0,1,\ldots,t$ have been definitely decoded, no node of depth $I < t$ will ever be extended, and all map locations J such that $M1(J) \leq t$ will be available for re-assignment. This can be done by a pointer LOCPUR that is initially set to 1 . LOCPUR is incremented by 1 until it has a value such that $M1(LOCPUR) \leq t$. In that case the new map entry will go into location LOCPUR.

DECODING DECISIONS

Suppose decisions at levels $0,1,\ldots,t$ have been made and a decision at levels $t+1,\ldots,t+j$ is to be made next ($j > 1$ for instance when decisions are likelihood-oriented) with node at location N1 determining the choice. Set NPO1 = NPOINT(N1), NPTM = NPTM1(N1) (we assume that $I \geq t+j$)

1. If NPTM > t go to 2. Otherwise stop.

The digits at levels $t+1,\ldots,t+j$ are those revealed by the map so far (i.e. those found by the usual following of back pointers in the map).

2. If M1(NPO1) = NPTM go to 3. Otherwise stop.

Then for the purposes of the decision all digits at levels lower than NPTM are zeros. The digit at level NPTM is a 1 and digits at levels higher than NPTM are those revealed by the map so far.

3. Set NPTM = M2(NPO1), NPO1 = MP(NPO1) and go to 1. The map digits revealed so far are valid.

Note: This procedure is successful because any new entry has a value of M1 that exceeds the old value of M1. Therefore if an old pointer NPO1 is involved, we will surely get M1(NPO1) > NPTM.

STACK MAINTENANCE AND PURGING STRATEGY WHEN PATHS ARE SPECIFIED BY ACTUAL SEQUENCES OF INFORMATION DIGITS OR PARITY DIGITS

Stack: Has locations called M1 of ν-1+k digits, the rightmost being the most recent, some of the leftmost possibly dummies. It has an I counter indicating the depth of the path and a pointer P1 to the preceding location in the map. In the forthcoming discussion it is assumed that M1 contains an info. sequence. If parity sequences are involved, only step 11 used need be changed in accordance with Note I. below.

Map: Has locations called M2 of k digits (no dummies here), pointer MPP indicating the preceding M2-location in the map, and pointers MPL indicating the next M2 location of the same depth.

Table: Entries A(1),...,A(j) indicate the values of the various "live" depths that exist in the map. Entries B(1),...,B(j), B(j+1), C(1),...,C(j) are pointers. B(i) points to the first location and C(i) to the last location in the map of depth A(i), i = 1,...,j. B(i) is chained to C(i) by means of the pointers MPL. In fact, $C(i) = MPL^{t}(B(i))$ where t is such that $MPL^{t+1}(B(i)) = 0$ and $MPL^{r}(B(i)) \neq 0$ for r = 0,1,...,t . B(j+1) points to the first available location of the map.

INITIALIZATION

$A(1) = -(j-1),\ldots,A(j) = 0$; $B(1) = \ldots = B(j) = 0$

$B(j+1) = 1$, $MPL(i) = i+1$ $i = 1,2,\ldots, \|M2\| - 1$

$MPL(\|M2\|) = 0$.

Rest is initialized to 0.

OPERATION

1. Upon obtaining for extension some stack location β, the decoder checks whether $I(\beta) = (\nu-1) + \ell k$ for some $\ell = 1,2,\ldots$

If $I(\beta) \neq (\nu-1) + \ell k$ go to 11, else continue.

2. If $I(\beta) \leq I_{max}$ go to 6, else $I_{max} \leftarrow I(\beta)$.

3. If $I_{max} < (\nu-1) + (j+1)k$ go to 5, else continue.

4. Go through the chained list $P1(\beta) \sim MPP(P1(\beta)) \sim \ldots \sim MPP^{j-1}(P1(\beta))$ and release the digits in location $M2(MPP^{j-1}(P1(\beta)))$ to the user.

5. Find the value t^* such that $A(t^*) = \ell - j$ and make available to the map those locations that are linked to $B(t^*)$. This is done by setting $MPL(C(t^*)) \leftarrow B(j+1)$ and $B(j+1) \leftarrow B(t^*)$.

If $B(j+1) = 0$, map overflow takes place and stop. Otherwise, set $A(t^*) \leftarrow \ell$, $B(t^*) \leftarrow C(t^*) \leftarrow B(j+1)$, $B(j+1) \leftarrow MPL(B(t^*))$, $MPL(B(t^*)) \leftarrow 0$, $MPP(B(t^*)) \leftarrow P1(\beta)$, and $P1(\beta) \leftarrow B(t^*)$. Copy the last k digits of $M1(\beta)$ into $M2(B(t^*))$. Go to 11 .

6. If $I_{max} - I(\beta) < \nu-1+jk$ go to 8, else continue.

7. In this case all of the decisions about digits contained in $M1(\beta)$ have already been made and this entry should therefore be purged from the stack. Go to 12.

8. If there is no t such that $A(t) = \ell$ go to 14, else continue.

9. Find t^+ such that $A(t^+) = \ell$. See if there exists a location α^+ linked to $B(t^+)$ such that $M1(\alpha^+) = P1(\beta)$ (this requirement is ignored if $\ell = \min_t A(t)$ and the contents of $M2(\alpha^+)$ are identical to the last k digits of $M1(\beta)$. If α^+ exists, set $P1(\beta) \leftarrow \alpha^+$ and go to 11, else continue (Note: if $B(t^+) = 0$ then no α^+ exists by definition).

10. If $B(j+1) = 0$, map overflow takes place and stop. Otherwise set $B^+ \leftarrow B(t^+)$, $B(t^+) \leftarrow B(j+1)$, $B(j+1) \leftarrow MPL(B(t^+))$, $MPL(B(t^+) \leftarrow B^+$ (this puts old $B(j+1)$ to the top of the t^+ set, and establishes a new top for the set $j+1$). If $B^+ = 0$ then set $C(t^+) \leftarrow B(t^+)$. Also, set $MPP(B(t^+)) \leftarrow P1(\beta)$ and $P1(\beta) \leftarrow B(t^+)$. Finally, copy the last k digits of $M1(\beta)$ into $M2(B(t^+))$.

11. The rightmost $(\nu-1)$ digits of $M1(\beta)$ are used to find the likelihoods λ_o and λ_1 of the two extensions of the path on top of the stack. If the 0-extension stays at stack location β and the 1-extension goes into location β_1, then $I(\beta) \leftarrow I(\beta_1) \leftarrow I(\beta) + 1$, $P1(M_1) \leftarrow P1(\beta)$, $M1(\beta)$ is shifted left by 1 stage, a 0 being entered into the rightmost stage, $M1(M)$ is copied into $M1(\beta_1)$ and a 1 is entered into the rightmost stage of the latter, $CUM(\beta_1) \leftarrow CUM(\beta) + \lambda_1$, and $CUM(\beta) \leftarrow CUM(\beta) + \lambda_o$. Appropriate pointers to locations β and β_1 are set in the auxiliary stack as usual.

12. Find the top of the stack.

13. Go to 1.

14. This is the case then $I_{max} - (j-1)k > I(\beta) > I_{max} - [(\nu-1)+jk]$ so there exist some digits in $M1(\beta)$ that have not been decided yet, but the pointer $P1(\beta)$ does not point to any valid entry, and furthermore $\min_t A(t) > \Omega$. Go to 11.

NOTE I.

If stack is not to contain message digits but rather the digits of the parity vector, step 11 of the procedure must be modified. In this case, what is to be saved are the parity digits.

We have two parity sequences $P_1^n(D)$ and $P_2^n(D)$ (see (15) of II-A-1) that must be saved. Furthermore,

$$x_{\cdot,n} = s_n + p_{i,0}^n$$

$$i = 1,2$$

$$P_i^{n+1}(D) = D^{-1}\left[P_i^n(D) + p_{i,0}^n\right] + s_n G_i^*(D)$$

We assume as previously, that $g_{1,0} = g_{2,0} = g_{1,\lambda-1} = g_{1,\nu-1} = 1$, $G_1(D)$ and $G_2(D)$ being of degrees $\lambda-1$ and $\nu-1$, respectively. Therefore, M1(β) must contain $k + (\lambda-1) + (\nu-1)$ stages. One possibility is that its contents are given by

$$\left[s_{n-k},\ldots,s_{n-1};p_{1,0}^n,\ldots,p_{1,\lambda-2}^n,p_{2,0}^n,\ldots,p_{2,\nu-2}^n\right]$$

The other possibility is to save $\nu-1$ positions in the map by taking advantage of the fact that $p_1^{(n)}(D)$ and $p_{1,0}^{n-1},\ldots,p_{1,0}^{\lambda-1}$ determine $s_{n-1},\ldots,s_0$ uniquely by use of the circuit of Fig. 1b of Part II. In this case M1(β) would contain

$$\left[p_{1,0}^{n-k},\ldots,p_{1,0}^{n-1},p_{1,0}^n,p_{1,1}^n,\ldots,p_{1,\lambda-2}^n,p_{2,0}^n,\ldots,p_{2,\nu-2}^n\right]$$

We will denote the coefficient vector of $P_i^n(D)$ by $\underset{\sim}{p}^i$.

Therefore we get the following two possible alternatives to step 11.

<u>11a</u>. (M1(β) contains $[\underset{\sim}{s},\underset{\sim}{p}^1,\underset{\sim}{p}^2]$, map contains $\underset{\sim}{s}$). $\underset{\sim}{s},\underset{\sim}{p}^1$ and $\underset{\sim}{p}^2$ are shifted separately to the left, the leftmost digits $p_{1,0}^n$ and $p_{2,0}^n$ being used to compute the likelihoods λ_o and λ_1 of the two extensions of this path. After the shift the leftmost digits are dumped and a 0 is supplied to all three rightmost positions of $\underset{\sim}{s}$, $\underset{\sim}{p}^1$ and $\underset{\sim}{p}^2$.

If the 1-extension goes into location β_1 then $I(\beta) \longleftarrow I(\beta_1) \longleftarrow I(\beta)+1$, $P1(\beta_1) \longleftarrow P1(\beta)$ and $M1(\beta_1) \longleftarrow M1(\beta) + [0,\ldots,1, \underset{\sim}{g}^1, \underset{\sim}{g}^2]$ where $\underset{\sim}{g}^1 = g_{1,1},\ldots,g_{1,\lambda-1}$ and $\underset{\sim}{g}^2 = g_{2,1},\ldots,g_{2,\nu-1}$. Finally, $CUM(\beta_1) \longleftarrow CUM(\beta) + \lambda_1$ and $CUM(\beta) \longleftarrow CUM(\beta) + \lambda_o$. Appropriate pointers to locations β and β_1 are set in the auxiliary stack as usual.

11b. ($M1(\beta)$ contains $[\underset{\sim}{p}^*, \underset{\sim}{p}^1, \underset{\sim}{p}^2]$, map contains $\underset{\sim}{p}^*$, where $\underset{\sim}{p}^* = p_{1,0}^{n-k},\ldots,p_{1,0}^{n-1}$. $(\underset{\sim}{p}^*, \underset{\sim}{p}^1)$ and $(\underset{\sim}{p}^2)$ are shifted separately left, the digits $p_{1,0}^n$ and $p_{2,0}^n$ being used to compute the likelihoods λ_o and λ_1 after two extensions of this path. After the shift the leftmost digits of $(\underset{\sim}{p}^*, \underset{\sim}{p}^1)$ and $(\underset{\sim}{p}^2)$ are dumped and a 0 is supplied to both rightmost positions. If the 1-extension goes into location β_1 then

$$I(\beta) \longleftarrow I(\beta_1) \longleftarrow I(\beta)+1, P(\beta_1) \longleftarrow P(\beta), \text{ and } M1(\beta_1) \longleftarrow M1(\beta) + [0,\ldots,0, \underset{\sim}{g}^1, \underset{\sim}{g}^2]$$

Finally, $CUM(\beta_1) \longleftarrow CUM(\beta)+\lambda_1$ and $CUM(\beta) \longleftarrow CUM(\beta)+\lambda_o$. Appropriate pointers to locations β and β_1 are set in the auxiliary stack as usual.

APPENDIX 2

TABLE LOOK-UP FOR MULTIBRANCH ADVANCE

I. Binary Symmetric Channel-Systematic Code

Suppose we wish to advance ℓ message bits at a time, and let us assume a rate 1/2, systematic codes.

We will describe how the move forward is carried out at some time i at which the parity state polynomial is

$$P_i(D) = p_1(i) + p_2(i)D + \ldots + p_{\nu-1}(i)\, D^{\nu-2} \tag{1}$$

where ν is the constraint length of the code. We will assume that the generator polynomial is

$$G(D) = 1 + g_1 D + \ldots + g_{\nu-1}\, D^{\nu-1} \tag{2}$$

Suppose the next ℓ-digit information polynomial is

$$S_i(D) = s_{i+1} + s_{i+2}D + \ldots + s_{i+\ell}\, D^{\ell-1} \tag{3}$$

Then the first-position transmitted polynomial is

$$X_1(D) = S_i(D) \tag{4}$$

and the second-position transmitted polynomial is

$$X_2(D) = [P_i(D) + S_i(D)\, G(D)]^{\ell} \tag{5}$$

where $[\ ]^{\ell}$ denotes truncation of ℓth and higher powers.

Next, suppose first and second position polynomials $Y_1(D)$ $Y_2(D)$ are received, respectively, and it is desired to find the k^{th} most likely sequence $s_{i+1},\ldots,s_{i+\ell}$ that could have caused it $(k=1,2,\ldots,2^{\ell})$.

If the channel is BSC, then the answer depends strictly on the weight of the difference polynomials

$$Z_1(D) = X_1(D) + Y_1(D)$$

$$Z_2(D) = X_2(D) + Y_2(D) \quad . \tag{6}$$

But, note from (5) and (4) that

$$Z_2(D) = Y_2(D) + [P_i(D) + X_1(D)\ G(D)]^{\ell}$$

$$= Y_2(D) + [P_i(D) + Z_1(D)\ G(D) + Y_1(D)\ G(D)]^{\ell}$$

$$= \left\{[P_i(D) + Y_1(D)\ G(D)]^{\ell} + Y_2(D)\right\} + [Z_1(D)\ G(D)]^{\ell}$$

$$= B(D) + [Z_1(D)\ G(D)]^{\ell} \tag{7}$$

where $B(D)$ is independent of $Z_1(D)$.

It thus follows from (7) that we can arrange tables that will be useful in evaluating likelihoods and identities of k^{th} most likely branches leaving a node.

The first table, called LTABM, lists for each of 2^{ℓ} possible different values of $B(D)$ the weight-ordered sequence of the 2^{ℓ} different outgoing branches $Z_1(D)$ (the weights are simply $wt(Z_1(D)) + wt(Z_2(D))$ both of which depend on $Z_1(D)$ and $B(D)$ only).

The second table, called LTABW, lists for each $B(D)$ the weights corresponding to the outgoing branches of LTABM.

A third table, LIK, gives the correspondence between weights and likelihood values.

Finally, it might be useful to have a fourth table, called CODEY that would supply the correspondence between $Y_1(D)$ and $[Y_1(D)\ G(D)]^{\ell}$

Suppose the k^{th} most likely outgoing branch was wanted, one would proceed as follows:

1) Look-up $[Y_1(D)\ G(D)]^{\ell}$ in CODEY and form

$$B(D) = [P_i(D)]^{\ell} + [Y_1(D)\ G(D)]^{\ell} + Y_2(D)$$

2) Find the k^{th} entry $Z_1(D)$ and the $B(D)$-row of LTABM and form the corresponding message sequence

$$S_i(D) = Z_1(D) + Y_1(D)$$

3) Form the next parity state polynomial $P_{i+\ell}(D)$ recursively as follows

$$\begin{aligned} P_{i+1}(D) &= \left\{D^{-1}[P_i(D) + s_{i+1}G(D)]\right\}^* \\ &\vdots \\ P_{i+\ell}(D) &= \left\{D^{-1}[P_{i+\ell-1}(D) + s_{i+\ell}G(D)]\right\}^* \end{aligned}$$

where $\{\}^*$ denotes the dropping of the D^{-1} term. Before generation of $P_{i+j}(D)$, the coefficient $p_1(i+j-1)$ is stored in the map.

4) Find the weight w_k of the k^{th} entry in the $B(D)$-row of LTABW and look up in LIK the likelihood of the corresponding branch. The latter is then used in forming the cumulative likelihood of the new path.

NOTE: The value of $B(D)$ should really be computed only when the node is extended for the first time, i.e. along the most likely branch. Then it should be stored for later use if the k^{th} $(k=2,3,\ldots,2^{\ell})$ most likely branch is needed.

NOTE: Obviously a straight-forward modification of this method will apply to any rate code. Regardless of the rate, the LTABM and LTABW

tables will have 2^n entries, where n is the number of received bits that correspond to a path extension (In the preceding discussion, $n = 2\ell$.).

II. BINARY INPUT-M-ary OUTPUT SYMMETRICAL CHANNEL-SYSTEMATIC CODE

We will consider the situation where for simplicity the received symbols can be written as pairs (Y,V), where Y is binary and V has alphabet of size $m = M/2$. Furthermore, the channel structure is such that

$$\begin{aligned} w(0,V/0) &= w(1,V/1) \\ w(o,V/1) &= w(1,V/0) \end{aligned} \tag{8}$$

for all $V \in \{0,1,\ldots,M-1\}$. In this formulation the Ames channel has $M = 4$. Note from (8) that the likelihood

$$\log \frac{w(Y,V/X)}{w(Y,V)} - R = \log \frac{f_1(Y \oplus X,V)}{f_2(V)} - R \tag{9}$$

is a function of the pair $(Y \oplus X,V)$ only. Assuming that

$$\min_V \frac{f_1(0,V)}{f_2(V)} > \max_V \frac{f_1(1,V)}{f_2(V)} \tag{10}$$

(which is true on the Ames Channel), the following strategy is very reasonable.

Create a table LTAB which for each of the 2^ℓ possible different values of $B(D)$ (they are based on the Y-components of the received symbols only!) lists the weight-ordered sequence of 2^ℓ different outgoing branches $(Z_1(D),Z_2(D))$ (note that it will be handy to list $Z_2(D)$ also). Ties in weights are resolved in some arbitrary manner.

Create a table LIK giving the correspondence between the pairs $(Z,V) = (Y \oplus X,V)$ and the likelihood values

$$\log \frac{f_1(Z,V)}{f_2(V)} - R$$

Finally, construct the table CODEY that will have the correspondence between $Y_1(D)$ and $[Y_1(D)\ G(D)]^{\ell}$.

The path extension procedure below will not be able to pick every time the k^{th} most likely outgoing branch, because, e.g., in case of ties although the distance between a received sequence

$$y_1, y_2, \ldots, y_n$$

and two possible branch sequences $x_1, \ldots, x_n$ and $x_1^1, \ldots, x_n^1$ might be the same, the distances between the latter and the actual symbol sequence

$$(y_1, v_1)\ ,\ (y_2, v_2), \ldots, (y_n, v_n)$$

may turn out to be very different. However, it is believed that most of the time the errors in ordering will not damage the algorithm's performance too much. Furthermore, experiments will no doubt bear out the simplicity and speed advantage of the suggested extension procedure:

1) Look-up $[Y_1(D)\ G(D)]^{\ell}$ in CODEY and form

$$B(D) = [P_i(D)]^{\ell} + [Y_1(D)\ G(D)]^{\ell} + Y_2(D)$$

2) Find the k^{th} entry $[Z_1(D), Z_2(D)]$ in the $B(D)$ row of LTABM and form the corresponding message sequence

$$S_i(D) = Z_1(D) + Y_1(D)$$

3) Form recursively the next parity state polynomial $P_{i+\ell}(D)$:

$$P_{i+1}(D) = \left\{D^{-1}\ [P_i(D) + s_{i+1}\ G(D)]\right\} *$$

$$\vdots$$

$$P_{i+\ell}(D) = \left\{D^{-1}\ [P_{i+\ell-1}(D) + s_{i+\ell}\ G(D)]\right\} *$$

4) Using the results of (2), look-up in LIK the likelihoods $\log (f_1(z_j,v_j)/f_2(v_j) - R$ and form the likelihood increment λ corresponding to the branch $S_i(D)$:

$$\lambda = \sum_{j=1}^{2\ell} \left[\log \frac{f_1(z_j,v_j)}{f_2(v_j)} - R\right]$$

III. BINARY SYMMETRIC CHANNEL - NON-SYSTEMATIC CODES

We will conclude by treating non-systematic codes of rate 1/2 for the BSC. The treatment of such codes for the symmetric channels of Sec. II. is similar and is left as an exercise.

Let the two generator polynomials be $G_1(D)$, and $G_2(D)$, and denote teh two parity state polynomials by $P_i^1(D)$, $P_i^2(D)$.

CASE I: One of $G_1(D)$, $G_2(D)$., say $G_1(D)$ is such that

$$g_{1,1} = g_{1,2} = \cdots = g_{1,\ell-1} = 0 \tag{11}$$

In this case the first position transmitted polynomial is

$$X_1(D) = [P_i^1(D)]^{\ell} + S_i(D) \tag{12}$$

(it is assumed that $g_{1,0} = 1$), and the second position polynomial is

$$X_2(D) = [P_i^2(D)]^{\ell} + [S_i(D)\ G_2(D)]^{\ell} \tag{13}$$

Therefore, the difference polynomials $Z_1(D)$ and $Z_2(D)$ are

$$\begin{aligned} Z_1(D) &= Y_1(D) + [P_i^1(D)]^{\ell} + S_i(D) \\ Z_2(D) &= Y_2(D) + [P_i^2(D)]^{\ell} + [S_i(D)\ G_2(D)]^{\ell} = \end{aligned} \tag{14}$$

$$= Y_2(D) + [P_i^2(D)]^{\ell} + [(Z_1(D) + Y_1(D) + [P_i^1(D)]^{\ell})\, G_2(D)]^{\ell}$$

$$= B(D) + [Z_1(D)\, G_2(D)]^{\ell} \qquad (15)$$

where

$$B(D) = Y_2(D) + [P_i^2(D)]^{\ell} + [(Y_1(D) + [P_i^1(D)]^{\ell})\, G_2(D)]^{\ell} \qquad (16)$$

is not a function of the branch being extended.

CASE II: Neither $G_1(D)$ nor $G_2(D)$ has leading coefficients that satisfy (11). In this case

$$Z_1(D) = [P_i^1(D)]^{\ell} + S_i(D)[G_1(D)]^{\ell} + D^{\ell}F(D) + Y_1(D) \qquad (17)$$

where $D^{\ell} F(D)$ is identical with the polynomial consisting of higher than $(\ell-1)$ degree terms of $S_i(D)[G_1(D)]^{\ell}$. Also, there exists a polynomial $H(D)$ of degree at most $\ell - 1$ such that

$$[G_1(D)]^{\ell}\, H(D) = 1 + D^{\ell} E(D) \qquad (18)$$

where $E(D)$ is some polynomial of degree at most $\ell - 2$. Post-multiplying both sides of (17) by $H(D)$ we get

$$(Z_1(D) + [P_i^1(D)]^{\ell} + Y_1(D))H(D) = S_i(D) + D^{\ell}E(D)\, S_i(D) + D^{\ell}F(D)H(D) \qquad (19)$$

Since

$$[D^{\ell}F(D)H(D)]^{\ell} = 0 \quad \text{and} \quad [D^{\ell}E(D)S_i(D)]^{\ell} = 0$$

we get that

$$S_i(D) = [(Z_1(D) + [P_i^1(D)]^{\ell} + Y_1(D))\, H(D)]^{\ell} \qquad (20)$$

Therefore,

$$Z_2(D) = Y_2(D) + [P_i^2(D)]^{\ell} + [S_i(D)G_2(D)]^{\ell} =$$

$$Y_2(D) + [P_i^2(D)]^{\ell} + [([P_i^1(D)]^{\ell} + Y_1(D)H(D)G_2(D)]^{\ell}$$

$$+ [Z_1(D)H(D)G_2(D)]^{\ell}$$

111-116

INTENTIONALLY LEFT BLANK

Hence

$$Z_2(D) = B(D) + [Z_1(D)\ H(D)\ G_2(D)]^{\ell} \tag{21}$$

where

$$B(D)=Y_2(D)+[P_i^2(D)]^{\ell} + [([P_i^1(D)]^{\ell} +Y_1(D))H(D)G_2(D)]^{\ell} \tag{22}$$

is not a function of the branch being extended.

It is clear that for the non-systematic codes, relations (14), (15), and (16) (CASE I) or (20, (21), and (22) (CASE II) will be the basis for our table construction and for our path extension strategy.

We suggest the formation of the following tables:

I. LTABM, listing for each of the 2^{ℓ} different values of $B(D)$ the weight-ordered sequence of the 2^{ℓ} different outgoing branches $Z_1(D)$ (formula (15) is used for CASE I, and (21) for CASE II).

II. Table LTABW, listing for each $B(D)$ the weights corresponding to the outgoing branches of LTABM.

III. Table LIK listing the correspondence between weights and likelihoods.

IV. Code I listing the correspondence between $Y(D)$ and $[Y(D).G_2(D)]^{\ell}$ for CASE I and between $Y(D)$ and $[Y(D)H(D)G_2(D)]^{\ell}$ for CASE III.

V. CODE II listing for correspondence between $Y(D)$ and $[Y(D)H(D)]^{\ell}$ for CASE II.

We will now describe the method of finding the k^{th} most likely outgoing branch for CASE II. The treatment of CASE I is similar.

1) Look up $W_1(D) = ([(P_i^1(D))^{\ell} + Y_1(D)]\ H(D)\ G_2(D))^{\ell}$ in CODE I and form

$$B(D) = Y_2(D) + [P_i^2(D)]^{\ell} + W_1(D)$$

2) Find the k^{th} entry $Z_1(D)$ in the $B(D)$-row of LTABM, and form

$$W_2(D) = Z_1(D) + Y_1(D) + [P_i^1(D)]^{\ell}$$

3) Look up $S_i(D) = [W_2(D)\ H(D)]^{\ell}$ in CODE II and form recursively the parity state polynomials $P_{i+\ell}^1(D)$ and $P_{i+\ell}^2(D)$. Store the coefficients $p_1^1(i+j+1)$ in the map.

4) Find the weight w_k of the k^{th} entry in the $B(D)$-row of LTABW and look-up in LIK the likelihood of the extended branch.

NOTE: Extension of paths in non-systematic codes is clearly more cumbersome than that for systematic codes. It is therefore the latter that should be used wherever possible.

APPENDIX 3

DESCRIPTION OF THE RUDIMENTARY AND PULL-UP DECODING ALGORITHMS

It has been shown in Jelinek and Cocke[1] that boot-strap hybrid decoding is applicable to all channels symmetrical from the input that have input alphabets in a finite galois field. It is easiest to describe the method first as it applies to binary symmetric channels (BSC). The generalization to symmetrical channels with binary inputs and arbitrary output alphabets is described in section II-B-2.

As usual, we will encode blocks of Γ binary information symbols into codewords of length $(\Gamma + t)/R$ where R is the sequential coding rate and t is the length of the dummy information sequence (known to the decoder) that is used to make the sequential decoding of the last information symbols reliable. Let us encode $m-1$ blocks of information using the same convolutional code. We will refer to the resulting codewords as information streams. Let us arrange these streams underneath each other, obtaining the solid line array of Figure 1 . Let us then generate the m^{th} parity check stream (interrupted line in Figure 1) whose i^{th} digit will be the parity of the i^{th} digits of the $m-1$ information streams. Stated in another way, the parity stream is a modulo 2 position-by-position sum of the information streams. Because of the linearity of convolutional encoding, the parity check stream corresponds to a path in the coding tree whose information digits are the mod 2 sums of the information digits underlying the information streams. Hence, all m of the streams are in principle sequentially decodable. Moreover, if any subset of $m-1$ of these streams is correctly decoded, the remaining m^{th} stream can be determined by use of the parity relationship (in fact, Falconer's [Ref. (4), Part II] strategy is based solely on this observation). We now describe the rudimentary bootstrap hybrid decoding

scheme. Suppose that the m streams are sent through the binary symmetric channel, and that the corresponding received digits are again arranged by the decoder into an m by $(\Gamma + t)/R$ array (see the solid lines of Figure 2). If the j^{th} received stream is to be decoded, the received digits of all other streams should also be taken into account, since these contain information about the transmitted digits of the j^{th} stream (the transmitted digits are related by the parity constraint). However, it is easy to show that all the pertinent information of the i^{th} received digits $y_i(1), y_i(2), \ldots, y_i(m)$ about the i^{th} transmitted digit $x_i(j)$ in the j^{th} stream is contained in the pair $y_i(j)$, $z_i = y_i(1) \oplus y_i(2) \oplus \ldots \oplus y_i(m)$. Therefore, let the decoder generate a $(m+1)^{th}$ channel state stream (see interrupted line of Figure 2) whose i^{th} digit will be the parity of the i^{th} digits of the m received streams. Before specifying exactly how the state stream is to be used in the decoding, let us note that if it has a 1 in its j^{th} position, an odd number of received streams have an error in the j^{th} position, and if the state stream has a 0 in the j^{th} position, an even number of received streams have an error there.

Let $q_k(0)$ [$q_k(1)$] denote the probability that of k digits independently transmitted through a binary symmetric channel, an even [odd] number was incorrectly received. By a well known formula (see Gallager (1963), p. 40),

$$q_k(0) = \frac{1+(1-2p)^k}{2} \qquad q_k(1) = \frac{1-(1-2p)^k}{2} \tag{1}$$

where p is the crossover probability of the binary symmetric channel.

Let z_i denote the i^{th} state stream digit, and let $y_i(j)$ and $x_i(j)$ denote the i^{th} received and transmitted digits of the j^{th}

stream. For the purpose of decoding of the j^{th} stream we can view the transmission process as having taken place over an augmented channel with inputs $x_i(j)$, and outputs the pairs $(y_i(j),z_i)$. This channel is governed by the transmission probability matrix $w_m(y,z/x)$ that is specified by

$$w_m(0,0/0) = w_m(1,0/1) = (1-p)\, q_{m-1}(0)$$

$$w_m(0,1/0) = w_m(1,1/1) = (1-p)\, q_{m-1}(1)$$

$$w_m(1,0/0) = w_m(0,0/1) = p q_{m-1}(1)$$

$$w_m(1,1/0) = w_m(0,1/1) = p\, q_{m-1}(0) \qquad (2)$$

When sequentially decoding the j^{th} stream, the receiver should use in the usual way (Jelinek [1968] Sec. 105) the likelihood function

$$\lambda_m(i) \triangleq \log \frac{w_m(y_i(j),\, z_i/x_i(j))}{w_m(y_i(j),z_i)} - R \qquad (3)$$

$$w_m(y,z) \triangleq \frac{1}{2}\,[w_m(y,z/0) + w_m(y,z/1)] = \frac{1}{2}\, q_m(z) \qquad (4)$$

We are now ready to describe precisely the rudimentary bootstrap hybrid decoding algorithm. Let a step in the decoding process consist of a change of the decoder's node location in the coding tree. Let M be some convenient positive integer. Let the decoder start out by decoding the first stream (using the likelihood function (3) with j =1). If it does not complete the decoding job within M steps, it stores the parameters necessary for resumption of decoding at the node at which it was last located, and starts decoding the second stream from its origin). Again, if within M steps it does not successsfully decode the second stream, it stores the necessary parameters and switches its attentions to the the third stream, etc. If it turns out that the

decoding was not completed on any of the m received streams within the allotted M steps, the decoder returns to the first stream and resumes its decoding from the point at which it left off (the parameters stored previously for this purpose will enable it to do so). Again in this second round a maximum of M additional steps is alloted to each stream and if this does not suffice a next round is started beginning with the first stream, etc. After continuing in this manner the decoder will finally succeed in decoding one stream, say the j_1^{th}. This means that the decoder has found a path in the coding tree corresponding to message digits whose symbols it believes to have been those of the j_1^{th} transmitted stream. The receiver will then replace the j_1^{th} received stream in the array of Figure 2 by the estimated j_1^{th} transmitted stream and will recompute the symbols of the channel state stream. Assuming the decoding to be errorless, a 1 in the i^{th} position of the new state stream will indicate that an odd number of the $m-1$ undecoded streams has an error in the i^{th} position, and a 0 will indicate that an even number of transmission errors occurred. To decode any of the remaining $m-1$ received streams the decoder will take advantage of the newly computed channel state stream. Thus it will use the likelihood function λ_{m-1} based on the probabilities $w_{m-1}(y,z/x)$ that are defined by (2) if m is replaced everywhere by $m-1$. Decoding will start from the beginning of the first stream (assuming that $j_1 \neq 1$) and continue in a round robin fashion (with the j_1^{th} stream excluded), each stream being allocated M steps per try, until an additional stream is decoded, say the j_2^{th}. As before, the j_2^{th} received stream is replaced by the j_2^{th} estimated transmitted stream and the channel state stream is accordingly recomputed. The decoding of the $m-2$ remaining received streams then starts from the beginning node of the first undecoded stream

again, the likelihood λ_{m-2} used being based on the probabilities $w_{m-2}(y,z/x)$ defined in (2). The pattern is now clear, it only remains to note that when m-1 streams have been decoded, the remaining stream is determined from the parity constraint by taking mod 2 sums of the corresponding digits of the m-1 decoded streams.

Our method is seen to be a bootstrapping operation, with each additional decoded stream being helpful in the decoding of the remaining streams. Just how helpful the state stream is can be seen from the extreme use when all but two streams have been decoded. Then, when $z_i = 0$ the error probability in the i^{th} position on either of the streams is $p^2/[p^2 + (1-p)^2]$, and when $z_i = 1$, the error probability is 1/2 [the original crossover probability of the BSC is assumed to be p]. We therefore place great reliance on the correctness of those received digits corresponding to a 0 in the state stream, and no reliance on those corresponding to a 1. This speeds up decoding immensely.

We describe next the <u>pull-up</u> decoding algorithm as it applies to a Fano sequential decoder. The modifications necessary for stack decoding are easy and can be found in Jelinek and Cocke.[3] The pull-up scheme will do away with the excessively frequent (one every M steps) changes in the identity of the stream being decoded which involve a large overhead cost. In fact, there is no need to discontinue work on one stream as long as the decoder has not run into computational trouble such as takes place when the value of the running threshold T_0 drops by a predetermined amount U below the maximal value T_{MAX} ever achieved. We will say that a <u>U-drop</u> takes place at a node of depth i whose cumulative likelihood value is greater than or equal to $T_{MAX} + \tau - U$, and whose immediate predecessor has likelihood value less than or equal to $T_{MAX} - U$, where τ is the threshold increment of the Fano Algorithm.

The following suggested procedure will apply directly to the BSC, but its generalization to the various categories of channels symmetrical from the input are obvious. To describe the scheme simply, we will need to equip the channel state stream with an additional component k_i, $i = 1, 2, \ldots, (n+t)/R$ whose purpose will be to indicate how many streams have undecoded digits at position i. Thus at the start of the process, $k_i = m$ for all i. The function $\lambda_{k_i}(\ell)$ (see (3)) will be used in computing the likelihood of a branch of depth i belonging to the ℓ^{th} stream.

(1) Using the likelihood $\lambda_{k_i}(1) = \lambda_m(1)$ the receiver continues to decode the first stream until either a U-drop takes place or the decoding of the block is completed. If the latter event happens, the received first stream is replaced by the estimated transmitted one, the channel state stream is recomputed, and k_i is decremented by 1 for all i.

(2) If a U-drop takes place at a node of depth i_1, then all branches on the path to that node up to depth i_1-J will be considered definitely decoded, where J is a suitably large integer. Accordingly, the corresponding received digits will be replaced by the estimated transmitted ones, and the corresponding segment of the channel state stream will be recomputed. All the parameters necessary for eventual resumption of the decoding from the node at which the U-drop took place will be saved. Also, the value of a new parameter $k^*(1)$ will be set equal to the current value of k_{i_1-J+r} where r is a convenient integer. Finally, the values k_j will be decremented by 1 for $j = 1, 2, \ldots, i_1-J$, and a parameter $I(1)$ will be set to i_1-J.

(3) Decoding of the second stream will now begin based on the functions $\lambda_{k_i}(\ell)$, and continue until either a U-drop or stream decoding

completion takes place. In the second eventuality, the values k_j will be decremented by 1 for all j . In the first eventuality, k*(2) and I(2) are set equal to k_{i_2} and i_2-J , and then all k_j , j ϵ (1,...,i_2-J), are decremented by 1, where i_2 is the depth of the node at which the U-drop occurred. Decoding continues in the indicated manner until all m of the streams have been worked on.

(4) If there exist integers $\ell_1 > \ell_2 > 0$ such that $k_i = 0$, $i = 1,\ldots, \ell_2$, $k_i = 1$, $i = \ell_2+1,\ldots, \ell_1$, then we find the unique stream j* whose digits on levels $\ell_2+1,\ldots, \ell_1$ remain undecoded. These digits are then decoded from the algebraic constraint, the parameter I(j*) is set to ℓ_1, k_i is set to 0 for $i = \ell_2+1,\ldots, \ell_1$, and the parameters necessary to start decoding of the j* stream at the appropriate node of level ℓ_1 are stored.

(5) Undecoded streams are next divided into two categories. Category $\mathcal{S}_1$ includes streams $j_1, j_2, \ldots, j_\ell$ ($\ell \leq$ m-1) such that $k^*(j_t) > k_{I(j_t)+r}$; t = 1,...,ℓ (note that $I(j_t)$ is the depth of the furthest node of stream j_t that has been definitely decoded). Category $\mathcal{S}_2$ includes all the remaining undecoded streams. Decoding of the j_1^{th} stream will now start in the forward mode by placing the decoder at the node at which the U-drop took place and setting the threshold and cumulative likelihood values to 0 . The established pattern repeats until all of the streams $j_1, j_2, \ldots, j_\ell$ of $\mathcal{S}_1$ have been worked on, except yhat k_i will be decremented only for values $i \geq I(j)$ when work on the j^{th} stream is terminated. If any segment of any stream can be definitely decoded from the algebraic constraint, this is done and new parameters for that stream are determined as described in the preceding step. The undecoded streams are again partitioned into the categories $\mathcal{S}_1$ and $\mathcal{S}_2$. Note that the <u>new</u> $\mathcal{S}_1$ may now include some streams that belonged to the <u>old</u> $\mathcal{S}_2$.

If $\mathcal{S}_1$ is not empty and more than one undecoded stream remains, decoding of the streams of $\mathcal{S}_1$ continues as before. If only one undecoded stream remains, its identity is determined from the parity information and the task is completed.

(6) If $\mathcal{S}_1$ is found empty while $\mathcal{S}_2$ contains more than one stream, only one of two actions is possible. Either the decoding effort is abandoned or the size of U is increased and all of the undecoded streams are put into $\mathcal{S}_1$. After all the latter have been worked on, a new $\mathcal{S}_1$ is again formed in the regular manner. If the new $\mathcal{S}_1$ is empty, U must be increased further; if not, then work on streams of $\mathcal{S}_1$ resumes with U equal to its original value.

As pointed out earlier, analysis of a slight modification of this pull-up algorithm reveals (see the Appendix of Jelinek and Cocke [1]) that upper and lower bounds on $\underset{\sim}{E}[N^{\gamma}]$ can be obtained that are essentially independent of the block length Γ.

FIGURE CAPTIONS

Figure 1: The structure of the encoding block.

Figure 2: The structure of the decoding block.

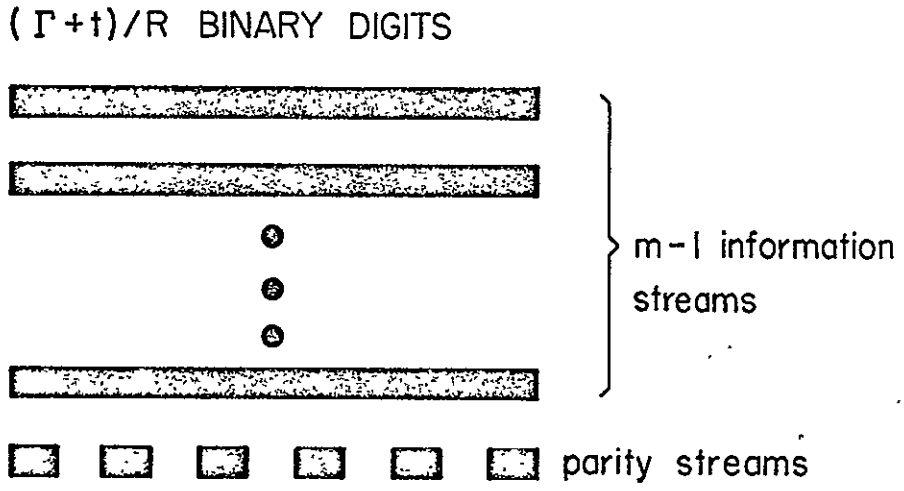


Figure 1: The structure of the encoding block.

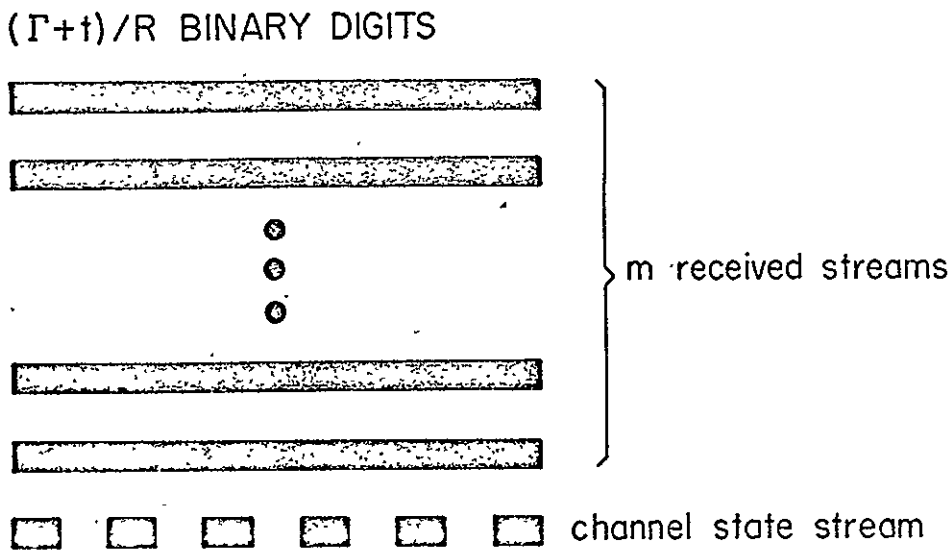


Figure 2: The structure of the decoding block.

APPENDIX 4

NEW UPPER BOUNDS ON CERTAIN COMPUTATIONAL PARAMETERS OF BOOTSTRAP HYBRID SEQUENTIAL DECODING

I. Introduction

We will be considering binary input discrete memoryless channels that are symmetrical from the input. However, the results are completely generalizeable to all channels symmetrical from the input. We impose the restriction to simplify our proofs.

A binary input channel of that class can be described as follows: Let any input $x \in \{0,1\}$ produce at the output a pair of digits (y,u) $y \in \{0,1\}$, $u \in \{0,1,\ldots,b-1\}$ and let the underlying channel transmission probability distribution have the following characteristic:

$$\begin{aligned} w(0,u/0) &= w(1,u/1) \\ w(1,u/0) &= w(0,u/1) \end{aligned} \tag{1}$$

for all $u \in \{0,\ldots,b-1\}$. Except for (1), the transmission function $w(y,u/x)$ will be considered arbitrary. Note that for the BSC, $b = 1$, so the u-portion of the pair may be omitted. In the sequel we will be considering the bootstrapping hybrid coding scheme that transmits M streams, M-1 of which are convolutionally encoded binary information digits, and the M^{th} stream is a modulo 2, position-by-position sum of the first M-1 streams. The convolutional code used is the same for each stream and as a consequence the M^{th} stream is also a codeword and can thus be decoded. The code will generate a tree with 2^k branches leaving each node, m digits to a branch (thus the rate $R = k/m$), and it will simplify our reasoning if the code constraint length will be infinite.

The decoding at the receiver will be done in the way described in Section VI of reference 2. Suppose K streams are left undecoded ($K \leq M$), and let

$$[\underline{y}_i, \underline{u}_i] = [(y_i(1), u_i(1)), (y_i(2), u_i(2)), \ldots, (y_i(K), u_i(K))]$$

be the vector pair of received digit pairs of the K undecoded streams in the i^{th} position. Then the decoding of the J^{th} stream will be based on likelihoods

$$\lambda_i(J) = \log \frac{P_K^J \left\{ \underline{y}_i, \underline{u}_i / x_i(J), t_i \right\}}{P_K \left\{ \underline{y}_i, \underline{u}_i / t_i \right\}} - R \qquad (2)$$

where the subscript K indicates the number of undecoded streams, and t_i is the parity of the i^{th} position digits that the decoder determined to have been transmitted in the $M-K$ decoded streams. Section IV of reference 2 shows how the righthand side of (2) can be simplified and easily computed. The probability $P_K^J \{ \}$ is, of course, given by

$$P_K^J \left\{ \underline{y}_i, \underline{u}_i / x_i(J), t_i \right\} + w(y_i(J), u_i(J) / x_i(J)) \times \sum_{\substack{x_K \\ \sum_{j=1} \\ x_i(j) = t_i}} \prod_{j \neq J} w(y_i(j), u_i(j) / x_i(j)) \quad . \qquad (3)$$

and

$$P_K \left\{ \underline{y}_i, \underline{u}_i / t_i \right\} = \frac{1}{2} \left[P_K^J \left\{ \underline{y}_i, \underline{u}_i / 0, t_i \right\} + P_K^J \left\{ \underline{y}_i, \underline{u}_i / 1, t_i \right\} \right] \qquad (4)$$

We conclude this section by proving

Lemma 1

Let a channel satisfying (1) and a convolutional code be given. The distribution of the number of decoding steps for any stream as well as the probability of error are invariant with respect to the actual information sequences encoded.

Proof

Let $\underset{\sim}{s}_1, \underset{\sim}{s}_2, \ldots, \underset{\sim}{s}_{M-1}$ be the information sequences of the first M-1 streams. Then by linearity of the convolutional code, the M^{th} stream corresponds to the sequence $\underset{\sim}{s}_M = \underset{\sim}{s}_1 \dot{+} \underset{\sim}{s}_2 \dot{+} \ldots \dot{+} \underset{\sim}{s}_{M-1}$ where mod 2, position-by-position sum is understood. Let the corresponding codewords be denoted by $\underset{\sim}{x}(\underset{\sim}{s}_1), \ldots, \underset{\sim}{x}(\underset{\sim}{s}_M)$, where, of course,

$$\underset{\sim}{x}(\underset{\sim}{s}_M) = \underset{\sim}{x}(\underset{\sim}{s}_1) \dot{+} \ldots \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_{M-1}) \tag{5}$$

Suppose the received sequence pairs are $(\underset{\sim}{y}_1, \underset{\sim}{u}_1), \ldots, (\underset{\sim}{y}_M, \underset{\sim}{u}_M) = (\underset{\sim}{y}, \underset{\sim}{u})$. Consider the J^{th} stream ($J \in \{1, \ldots, M\}$) and let $\underset{\sim}{x}(\underset{\sim}{s}, J)$ be its codeword corresponding to some arbitrary information sequence $\underset{\sim}{s}$. Then the likelihood associated with this codeword depends on the probabilities

$$P_M^J \left\{ \underset{\sim}{y}, \underset{\sim}{u} \,/ \underset{\sim}{x}(\underset{\sim}{s}, J), \underset{\sim}{0} \right\} \tag{6}$$

and

$$P_M \left\{ \underset{\sim}{y}, \underset{\sim}{u} \,/ \underset{\sim}{0} \right\} \tag{7}$$

where $\underset{\sim}{0}$ denotes an all zero sequence.

Now because of (1), the probability of receiving $(\underset{\sim}{y}, \underset{\sim}{u})$ when $\underset{\sim}{x}(\underset{\sim}{s}_1), \ldots, \underset{\sim}{x}(\underset{\sim}{s}_M)$ was transmitted is the same as the probability of receiving $(\underset{\sim}{y}_1 \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_1), \underset{\sim}{u}_1), \ldots, (\underset{\sim}{y}_M \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_M), \underset{\sim}{u}_M)$ when $\underset{\sim}{x}(\underset{\sim}{0}), \ldots, \underset{\sim}{x}(\underset{\sim}{0}) = \underset{\sim}{0}, \ldots, \underset{\sim}{0}$ was transmitted. Furthermore, it follows from (3) that for any $\underset{\sim}{s}$ and J,

$$P_M^J \left\{ (\underset{\sim}{y}_1, \underset{\sim}{u}_1), \ldots, (\underset{\sim}{y}_M, \underset{\sim}{u}_M) / \underset{\sim}{x}(\underset{\sim}{s}, J), \, 0 \right\} =$$

$$P_M^J \left\{ (\underset{\sim}{y}_1 \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_1), \underset{\sim}{u}_1), \ldots, (\underset{\sim}{y}_M \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_M), \underset{\sim}{u}_M) / \underset{\sim}{x}(\underset{\sim}{s}, J) + \underset{\sim}{x}(\underset{\sim}{s}_J, \underset{\sim}{0}) \right\} =$$

$$= P_M^J \left\{ (\underset{\sim}{y}_1 \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_1), \underset{\sim}{u}_1), \ldots, (\underset{\sim}{y}_M \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_M), \underset{\sim}{u}_M) / \underset{\sim}{x}(\underset{\sim}{s} + \underset{\sim}{s}_J, J), \underset{\sim}{0} \right\} \tag{8}$$

where the last equality is a consequence of the linear character of convolutional codes. It follows directly from (8) and (4) that also

$$P_M\left\{ \underset{\sim}{y},\underset{\sim}{u}/\underset{\sim}{0} \right\} = P_M \left\{ (\underset{\sim}{y}_1 \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_1),\underset{\sim}{u}_1),\ldots,(\underset{\sim}{y}_M \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_M),\underset{\sim}{u}_M)/\underset{\sim}{0} \right\} \qquad (9)$$

Since both whether or not an error was committed and the number of decoding operations depend on the likelihoods associated with the various paths in the tree and on their relation to each other, we see from (8) and (9) that these parameters will have the same value when $\underset{\sim}{s}_1,\ldots,\underset{\sim}{s}_M$ are transmitted and $(\underset{\sim}{y}_1,\underset{\sim}{u}_1),\ldots,(\underset{\sim}{y}_M,\underset{\sim}{u}_M)$ are received (event A) as when $\underset{\sim}{0},\ldots,\underset{\sim}{0}$ are transmitted and $(\underset{\sim}{y}_1 \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_1),\underset{\sim}{u}_1),\ldots,(\underset{\sim}{y}_M \dot{+} \underset{\sim}{x}(\underset{\sim}{s}_M),u_M)$ are received (event B). The conclusion of the Lemma then follows from the observation that both events A and B have equal probabilities for any $\underset{\sim}{s}_1,\ldots,\underset{\sim}{s}_M$, and any $(\underset{\sim}{y}_1,\underset{\sim}{u}_1),\ldots,(\underset{\sim}{y}_M,\underset{\sim}{u}_M)$.

QED

Corollary

When evaluating the probabality of error or the distribution of the number of decoding steps in the bootstrapping hybrid decoding scheme used with a binary input symmetrical channel, it may always be assumed that all-zero information sequences have been sent.

2. Some Preliminary Results

Let M streams be received and let $N_i^M (i \in \{1,\ldots,M\})$ be the number of decoding steps in the first incorrect subset of the i^{th} stream when the stack sequential decoding algorithm is used. In this section we will derive an upper bound on

$$E[\min_{1\le i\le M} N_i^M] \qquad (10)$$

We will follow a modification of an approach developed by Zigangirov.[3]

Consider the operation of the stack algorithm in the incorrect subset that starts with a particular branch emanating from some node whose path likelihood value is 2^{z}. Let the stack algorithm continue its operation until for the first time the likelihood value on the top of the stack falls below $\delta (\delta \leq z)$. Let $n_{\delta}(z)$ denote the number of operations until the stopping rule is invoked, and let (the expectation is over the ensemble of convolutional codes and over the transmission process)

$$N_{\delta}(z) = E[n_{\delta}(z)] \tag{11}$$

Let 2^{k} be the number of branches leaving each node, and let $\vec{\underline{y}}, \vec{\underline{u}}, \vec{\underline{x}}$ be the sequences of length m of $\underline{y}, \underline{u}, \underline{x}$ corresponding to a branch. Define the branch likelihood function

$$\lambda^{J}(\vec{\underline{y}}, \vec{\underline{u}}, \vec{\underline{x}}) = \log \frac{P_{M}^{J}(\vec{\underline{y}}, \vec{\underline{u}} / \vec{\underline{x}}, 0)}{P_{M}(\vec{\underline{y}}, \vec{\underline{u}} / \vec{0})} - mR \tag{12}$$

Then, since in the code ensemble the branches in the incorrect subset are selected independently from the all-zero transmitted branches (see Lemma 1), $N_{\delta}(z)$ satisfies the difference equation

$$N_{\delta}(z) = 2^{k} \sum_{\vec{\underline{y}}, \vec{\underline{u}}, \vec{\underline{x}}} N_{\delta}(z + \lambda^{1}(\vec{\underline{y}}, \vec{\underline{u}}, \vec{\underline{x}}) \, 2^{-m} \prod_{i=1}^{M} w(\vec{y}(i), \vec{u}(i)/0) + 1$$

$$z \geq \delta \tag{13}$$

$$N_{\delta}(z) = 0 \qquad z < \delta$$

Lemma 2

For $\delta < 0$,

$$N_{\delta}(z) \leq \frac{1}{2^{k}-1} \left[2^{\rho(z-\delta-\alpha)} - 1 \right] \tag{14}$$

where

$$\alpha = \min_{\vec{y},\,\vec{u},\,\vec{x}} \lambda(\vec{y},\vec{u},\vec{x}) \tag{15}$$

and $\rho \in (0,1)$ satisfies

$$\sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} w(y(i),u(i)/0) \left(\sum_{x} \left[\frac{P_M^1(\underline{y},\underline{u}/x,0)}{P_M(\underline{y},\underline{u}/0)} \right]^{\rho} \right)^{1/\rho} \leq 2^{\frac{1}{\rho}[1-(1-\rho)R]} \tag{16}$$

Proof

By the well-known maximum principle[4], $N_\delta^*(z)$ will be an upper bound on $N_\delta(z)$, provided that

$$N_\delta^*(z) \geq 0 \quad \text{for} \quad z \in (\delta+\alpha, \delta) \tag{17}$$

and that the lefthand side of (13) is not smaller than the righthand side for $z \geq \delta$ when $N_\delta^*(z)$ is substituted for $N_\delta(z)$. Substituting

$$N_\delta^*(z) = \frac{1}{2^k-1}\left[2^{\rho(z-\delta-\alpha)} - 1\right] \tag{18}$$

into the righthand side of (13), we get

$$1 - \frac{2^k}{2^k-1} + \frac{1}{2^k-1} 2^{\rho(z-\delta-\alpha)} 2^{k-m} \sum_{\vec{y},\,\vec{u},\,\vec{x}} 2^{\rho\lambda(\vec{y},\vec{u},\vec{x})} \prod_{i=1}^{M} w(\vec{y}(i),\vec{u}(i)/\vec{0})$$

$$= \frac{1}{2^k-1}\left[2^{\rho(z-\delta-\alpha)} \left\{ 2^{k-m-m\rho R} \sum_{\vec{y},\,\vec{u},\,\vec{x}} \left[\frac{P_M^1(\vec{y},\vec{u}/\vec{x},0)}{P_M(\vec{y},\vec{u}/\vec{0})} \right] \prod_{i=1}^{M} w(\vec{y}(i),\vec{u}(i)/0) \right\} - 1 \right] \tag{19}$$

Since $N^*_\delta(z)$ does satisfy (17), then the bound (14) will be valid provided the braced expression in (19) is smaller than or equal to 1. Using Hölder's inequality and the relation (16) (we are making use of the independence of digits along branches),

$$\sum_{\vec{y},\,\underline{u}} \prod_{i=1}^{M} w(\vec{y}(i),\vec{u}(i)/\vec{0}) \sum_{\vec{x}} \left[\frac{P_M^1(\vec{y},\underline{u}/\vec{x},\vec{0})}{P_M(\vec{y},\underline{u}/\vec{0})}\right]^{\rho} \leq$$

$$\left[\sum_{\vec{y},\underline{u}} \prod_{i=1}^{M} w(\vec{y}(i),\vec{u}(i)/\vec{0}) \left(\sum_{\vec{x}} \left[\frac{P_M^1(\vec{y},\underline{u}/\vec{x},\vec{0})}{P_M(\vec{y},\underline{u}/\vec{0})}\right]^{\rho}\right)^{\frac{1}{\rho}}\right]^{\rho} \leq 2^{m[1-(1-\rho)R]} =$$

$$= 2^{m-k+m\rho R} \tag{20}$$

Therefore, the righthand side of (19) is less than or equal to $N^*_\delta(z)$ and the Theorem is proven. QED

Next, let $z_1, z_2, \ldots, z_M$ be the likelihood values on the i^{th} $(i \geq 0)$ nodes of the true paths of the M received streams (by Lemma 1 these are the all-zero paths). Let $\nu < 0$ be arbitrary and define the indicator function $\phi_\nu(z_1, z_2, \ldots, z_M)$ to be equal to 1 if the likelihood on all of M of the true paths leaving the i^{th} node falls below the value ν. Otherwise let $\phi_\nu(z_1, \ldots, z_M)$ be equal to 0. Furthermore, define

$$\Phi_\nu(z_1, z_2, \ldots, z_M) = E[\phi_\nu(z_1, z_2, \ldots, z_M)] \tag{21}$$

Since the all-zero information path corresponds to the all-zero transmitted sequence, Φ_ν satisfies the following recurrence:

$$\Phi_\nu(z_1, \ldots, z_M) = \sum_{\vec{y},\,\underline{u}} \Phi_\nu(z_1 + \lambda^1(\vec{y},\underline{u},\vec{0}), \ldots, z_M + \lambda^M(\vec{y},\underline{u},\vec{0})) \cdot$$

$$\cdot\, w(\vec{y}(1),\vec{u}(1)/\vec{0}) \ldots w(\vec{y}(M),\vec{u}(M)/\vec{0}) \tag{22}$$

if $z_i > \nu$ for some i,

$$\Phi_\nu(z_1,\ldots,z_m) = 1 \qquad \text{if} \quad z_i \le \nu \quad \text{for all} \quad i$$

Lemma 3

$$\Phi_\nu(z_1,\ldots,z_M) \le 2^{-\mu(z_1+\ldots+z_M-M\nu)} \quad \text{if} \quad \max_i (z_i,\ldots,z_M) > \nu \tag{23}$$

where $\nu < 0$, $\mu > 0$ satisfies

$$\sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} w(y(i),u(i)/0) \left[\frac{P_M^i(\underline{y},\underline{u}/0,0)}{P_M(\underline{y},\underline{u}/0)}\right]^{-\frac{\mu}{1-\rho}} \le 2^{-\frac{\mu}{1-\rho}MR} \tag{24}$$

and $\rho \in (0,1)$ is the parameter defined in (16).

Proof

By the maximum principle[4], $\Phi^*_\nu(z_1,\ldots,z_M)$ will be an upper bound on $\Phi_\nu(z_1,\ldots,z_M)$, provided that

$$\Phi_\nu(z_1,\ldots,z_M) \ge 1 \quad \text{if} \quad z_i < \nu \quad \text{for all} \quad i \tag{25}$$

and that the lefthand side of (22) is not smaller than the righthand side when Φ^*_ν is substituted for Φ_ν. The function

$$\Phi^*(z_1,\ldots,z_M) = 2^{-\mu(z_1+\ldots+z_M-M\nu)} \tag{26}$$

surely satisfies (25). Substituting it for Φ_ν into the righthand side of (22) we get

$$2^{-\mu(z_1+\ldots+z_M-M\nu)} \sum_{\vec{y},\vec{u}} \prod_{i=1}^{M} \left[2^{-\mu\lambda^i (\vec{y},\vec{u},\vec{0})} \; w(\vec{y}(i), \vec{u}(i)/\vec{0},\vec{0}) \right] =$$

$$= 2^{-\mu(z_i+\ldots+z_M-M\nu)} \left\{ 2^{\mu MmR} \sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} \left[\frac{P_M^i(\vec{y},\vec{u}/\vec{0},\vec{0})}{P_M(\vec{y},\vec{u}/0)} \right]^{-\mu} w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0}) \right\} \tag{27}$$

Thus Φ_ν^* will be an upper bound on Φ_ν provided the value of the braced expression in (27) does not exceed 1 . However for $\rho \in (0,1)$ that value is by Hölder's inequality dominated by

$$2^{\mu MmR} \left(\sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} \left[\frac{P_M^i(\underline{y},\underline{u}/0,0)}{P_M(\underline{y},\underline{u}/0)} \right]^{-\frac{\mu}{1-\rho}} w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0}) \right)^{1-\rho} \leq$$

$$2^{\mu MmR} \; 2^{-\mu MmR} = 1$$

where the inequality is due to (24) and the fact that digits along branches are independent. QED

Finally, let us define the function

$$\Psi_{\nu,\delta}(z_1,\ldots,z_M,z) = E[\phi_\nu(z_1,\ldots,z_M)\; n_\delta(z)] \tag{28}$$

where it is understood that

a) $n_\delta(z)$ refers to the incorrect subset of some arbitrary but fixed stream $J \in \{1,2,\ldots,M\}$

b) the likelihoods $z_1,\ldots,z_M,z$ occur on the <u>same</u> i^{th} depth level in all streams.

$\Psi_{\nu,\delta}(z_1,\ldots,z_M,z)$ may thus be interpreted as the expected number (over

the set of events for which $\phi_\nu(z_1,\ldots,z_M) = 1$) of decoding steps in the J^{th} incorrect subset stemming from some branch that leaves a node on depth i whose likelihood value is z, if decoding terminates when the likelihood value of the top of the stack falls below δ. $\Psi_{\nu,\delta}$ then satisfies the recurrence

$$\Psi_{\nu,\delta}(z_1,\ldots,z_M,z) =$$

$$2^k \sum_{\vec{y},\vec{u},\vec{x}} \Big[\Psi_{\nu,\delta}(z_1+\lambda^1(\underline{y},\underline{u},0),\ldots,z_M+\lambda^M(\vec{y},\vec{u},\vec{0}),z+\lambda^J(\vec{y},\vec{u},\vec{x})) \cdot 2^{-m} \prod_{i=1}^{M} w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0}) \Big] + \Phi_\nu(z_1,\ldots,z_M) \tag{29}$$

if $z > \delta$ and $\max_i (z_1,\ldots,z_M) > \nu$ where

$$\Psi_{\nu,\delta}(z_1,\ldots,z_M;z) = N_\delta(z) \quad \text{if} \quad \max_i (z_1,\ldots,z_M) \le \nu$$

$$\Psi_{\nu,\delta}(z_1,\ldots,z_M,z) = 0 \quad \text{if} \quad z \le \delta \tag{30}$$

Lemma 4

$$\Psi_{\nu,\delta}(z_1,\ldots,z_M,z) \le \frac{1}{2^k-1} \left[2^{-\mu(z_1+\ldots+z_M-M\nu)+\rho(z-\delta-\alpha)} - \Phi_\nu(z_1,\ldots,z_M) \right] \tag{31}$$

where ρ satisfies (16), μ satisfies (24), and α is defined in (15).

Proof

Let $\Psi^*_{\nu,\delta}(z_1,\ldots,z_M;z)$ be the righthand side of (31). Then by (23) and (14),

$$\Psi^*_{\nu,\delta}(z_1,\ldots,z_M,z) \geq N_\delta(z) \quad \text{if} \quad \max_i (z_1,\ldots,z_M) \leq \nu$$

Furthermore, if $z \geq \delta + \alpha$ then since $\rho \in (0,1)$, we get

$$\Psi^*_{\nu,\delta}(z_1,\ldots,z_M,z) \geq \frac{1}{2^k-1}\left[2^{-\mu(z_1+\ldots+z_M-M\nu)} - \Phi_\nu(z_1,\ldots,z_M)\right] \geq 0$$

where the last inequality follows from Lemma 3. Thus by the maximum principle[4] $\Psi^*_{\nu,\delta}$ will be an upper bound on $\Psi_{\nu,\delta}$, provided the righthand side of (29) exceeds the lefthand side when $\Psi^*_{\nu,\delta}$ is substituted for $\Psi_{\nu,\delta}$ into it. The righthand side of (29) is then equal to

$$\frac{1}{2^k-1}\,2^{-\mu(z_1+\ldots+z_M-M\nu)+\rho(z-\delta-\alpha)}\left\{2^{k-m}\sum_{\vec{y},\vec{u},\vec{x}}2^{\lambda^J(\vec{y},\vec{u},\vec{x})}\prod_{i=1}^{M}2^{-\mu\lambda^i(\vec{y},\vec{u},\vec{0})}w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0})\right\}$$

$$-\frac{2^k}{2^k-1}\;\Phi_\nu(z_1,\ldots,z_M)+\Phi_\nu(z_1,\ldots,z_M)$$

Thus all we need to show is that the expression in braces does not exceed 1 . In fact, it is equal to

$$2^{k-m-\rho mR+\mu MmR}\sum_{\vec{y},\vec{u},\vec{x}}\left[\frac{P_M^J(\vec{y},\vec{u}/\vec{x},\vec{0})}{P_M(\vec{y},\vec{u}/\vec{x})}\right]^{\rho}\prod_{i=1}^{M}\left[\frac{P_M^i(\vec{y},\vec{u}/\vec{0},\vec{0})}{P_M(\vec{y},\vec{u}/\vec{0})}\right]^{-\mu}w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0})$$

$$\le 2^{k-m-\rho mR+\mu MmR}\left\{\sum_{\vec{y},\vec{u}}\prod_{i=1}^{M}w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0})\left(\sum_{\vec{x}}\left[\frac{P_M^J(\vec{y},\vec{u}/\vec{x},\vec{0})}{P_M(\vec{y},\vec{u}/\vec{0})}\right]^{\rho}\right)^{\frac{1}{\rho}}\right\}^{\rho}$$

$$\cdot\left\{\sum_{y,u}\prod_{i=1}^{M}\left[\frac{P_M^i(\vec{y},\vec{u}/0,0)}{P_M(\vec{y},\vec{u}/0)}\right]^{-\frac{\mu}{1-\rho}}w(\vec{y}(i),\vec{u}(i)/\vec{0},\vec{0})\right\}^{1-\rho}$$

$$= 2^{k-m-\rho mR+\mu MmR}\;2^{m(1-(1-\rho)R)}\;2^{-\mu MmR} = 1 \qquad (32)$$

The inequality in (32) follows from Hölder's inequality, the . from the fact that $k = mR$, and the next-to-last equality from the fact that μ and ρ satisfy relations (16) and (24) (by definition of the probability measure P_M^J, the first braced expression on the lefthand side of (32) is independent of J). QED

3. An Upper Bound on the Expected Minimum Number of Decoding Steps in the First Incorrect Subset

We will now use the conclusion of Lemma 4 to obtain an upper bound on the quantity $E[\min_{1\le i\le M} N_i^M]$ described at the beginning of Section 2. Note first that the upper bound (31) is independent of the index J of the stream whose incorrect subset is being decoded (see (28) and following). Let δ be the maximum of the likelihood minima pertaining to the correct paths of the M different streams. If this maximum is attained on the J^{th} stream, then the number of steps in the first incorrect subset of the J^{th} stream will be exactly $n_\delta(0)$. Since the first node of each stream has likelihood 0 , it follows that

$$E\left[\min_{1\le i\le M} N_i^M\right] \le 2^k-1 \int_{-\infty}^{0} \left[\frac{\partial}{\partial\nu}\Psi_{\nu,\delta}\ (0,\ldots,0,0)\right]\Bigg|_{\nu=\delta} d\delta \qquad (33)$$

where the coefficient 2^k-1 is necessary because there are that many incorrect branches leaving the first node. Now using (31),

$$\frac{\partial}{\partial\nu}\Psi(0,\ldots,0,0) \le \frac{1}{2^k-1}\ M\mu\ 2^{\mu M\nu - \rho(\delta+\alpha)}$$

since Φ_ν is an increasing function of ν. Hence

$$E\left[\min_{1\le i\le M} N_i^M\right] \le \frac{2^{-\rho\alpha}}{1-(\rho/M\mu)} \quad \text{if} \quad M\mu > \rho \qquad (34)$$

Now in (24) we have a relationship of the form

$$\left[E\, X^{\mu}\right]^{\frac{1}{\mu}} \leq 2^{-\frac{MR}{1-\rho}}$$

Since the lefthand side is a monotone increasing function of μ, the inequality is easier to satisfy if μ is as small as possible. But (34) says that $\mu > \rho/M$. So in order to find the rate R below which the lefthand side of (34) is finite, we will set $\mu = \rho/M$. Inequality (24) then becomes equivalent to

$$F_M(\rho) = \left(\sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} w(y(i),u(i)/0) \left[\frac{P_M^i(\underline{y},\underline{u}/0,0)}{P_M(\underline{y},\underline{u}/0)} \right]^{-\frac{\rho}{M(1-\rho)}} \right)^{\frac{1-\rho}{\rho}} \leq 2^{-R} \tag{35}$$

and (16) can be rewritten as

$$G_M(\rho) = \left(\sum_{\underline{y},\underline{u}} \prod_{i=1}^{M} w(y(i),u(i)/0) \left[\sum_{x} \left(\frac{P_M^1(\underline{y},\underline{u}/x,0)}{P_M(\underline{y},\underline{u}/0)} \right)^{\rho} \frac{1}{2} \right]^{\frac{1}{\rho}} \right)^{\frac{\rho}{1-\rho}} \leq 2^{-R} \tag{36}$$

and it is understood that $\rho \in (0,1)$. It can be shown that the lefthand side of (35), $F_M(\rho)$, is monotonically increasing with $\rho \in (0,1)$, and the lefthand side of (36), $G_M(\rho)$ is monotonically decreasing. Therefore, if $F_M(0) \leq G_M(0)$ and $F_M(1) \geq G_M(1)$, then there is a unique $\rho_M \in (0,1)$ such that $F_M(\rho_M) = G_M(\rho_M)$ and for all

$$R < -\log F_M(\rho_M) = -\log G_M(\rho_M) \tag{37}$$

the expected minimal amount of computation in the first incorrect subset is bounded by a constant. Since it can be shown that $F_M(0) \leq G_M(0)$ and $F_M(1) \geq G_M(1)$ is true always, then we have

Theorem 1

Let $\rho_M \in (0,1)$ be the unique value for which $F_M(\rho_M) = G_M(\rho_M)$. Then $E\left[\min_{1\le i\le M} N_i^M\right]$ is upper bounded by a constant for all rates

$$R < -\log F_M(\rho_M) \quad .$$

N_i^M is the number of decoding steps in the first incorrect subset of the i^{th} stream when M streams have been transmitted.

Let us next define $N_i(K)$ to be the number of decoding steps in the first incorrect subset of the i^{th} among the K streams that have been left undecoded (i.e. $M \ge K$ streams were received, M-K were decoded by the hybrid method, and K streams--probably the most difficult ones--are still to be decoded). We suggest that a very good measure of computational complexity is the parameter

$$E\left[\max_{2\le K\le M} \min_{1\le i\le K}\right] N_i(K) < \sum_{k=2}^{M} E\left[\min_{1\le i\le K} N_i(K)\right] \qquad (38)$$

which may be interpreted as the expected maximum number of decoding steps that need be done in the course of decoding of the entire hybrid block in any first incorrect subset.

Let $i_1, i_2, \ldots, i_k$ $(i_j \in (1,2,\ldots,M))$ be the indexes of those K streams that remain undecoded. Now by definition,

$$E\left[\min_{1\le i\le K} N_i(K)\right] = \sum_{\ell=0}^{\infty} P\left\{N_{i_1}^K > \ell, N_{i_2}^K > \ell, \ldots, N_{i_K}^K > \ell\right\} \qquad (39)$$

But $P\left\{N_{i_1}^K > \ell, \ldots, N_{i_K}^K > \ell\right\}$ is less than or equal to the probability that there is a subset of K streams from among the M which when considered together are such that the first incorrect subset of each

stream requires more than ℓ steps for its decoding. Hence by the union bound,

$$P\left\{N_{i_1}^K > \ell, \ldots, N_{i_K}^K > \ell\right\} \leq \binom{M}{K} P\left\{N_1^K > \ell, \ldots, N_K^K > \ell\right\}$$

Therefore by (38),

$$E\left[\max_{2\leq K\leq M} \min_{1\leq i\leq K} N_i(K)\right] \leq \sum_{K=2}^{M}\binom{M}{K}\sum_{\ell=0}^{\infty} P\left\{N_1^K > \ell, \ldots, N_K^K > \ell\right\} =$$

$$= \sum_{K=2}^{M}\binom{M}{K} E\left[\min_{1\leq i\leq K} N_i^K\right] \qquad (40)$$

From (39) and Theorem 1 we can then come to the following conclusion.

Theorem 2

Let $\rho_K \in (0,1)$, $K = 2,3,\ldots,M$ be the unique values for which $F_K(\rho_K) = G_K(\rho_K)$. Then $E\left[\max_{2\leq K\leq M} \min_{1\leq i\leq K} N_i(K)\right]$ is upper bounded by a constant for all rates

$$R < \min_{2\leq K\leq M} [-\log F_K(\rho_K)] \qquad (41)$$

APPENDIX 5

ESTIMATION OF EN_a AND EN_b

Lemma 1: Equation

$$2^{1-R} = e^{s(D^2-1)} + e^{sD^*} \tag{1}$$

has a (possibly complex) solution s for all $D^* \in (0,\infty)$

Proof

Suppose first $D^* = P/q$, p,q integers. Then (1) becomes

$$2^{1-R} = e^{s(\frac{p-q}{q})} + e^{s\ p/q} \tag{2}$$

Making the variable change $(e^s)^{1/q} = z$ and multiplying by z^q , (2) becomes

$$z^q\ 2^{1-R} = z^P + z^{P+q} \tag{3}$$

Now (3), is a polynomial in z , and as such, has at least one root by the theorems of algebra. If z_o is one of these roots, then clearly $e^s = (z_o)^q$ is a root of (1). Observing that rational numbers are everywhere dense on the real line, the Lemma follows. QED

Theorem 2

Under the hypotheses of Theorem 1 $|b-a| < \pi/\omega$ implies

$$EN_a \simeq \frac{r^{-a}}{\sin \omega a} \Big/ \left(\frac{\cos \omega a}{\sin \omega a} - \frac{\cos \omega b}{\sin \omega b}\right) \tag{5a}$$

$$EN_b \simeq \frac{r^{-b}}{-\sin \omega b} \Big/ \left(\frac{\cos \omega a}{\sin \omega a} - \frac{\cos \omega b}{\sin \omega b}\right) \tag{5b}$$

where r and ω are solutions of

$$2^{1-R} = r^{D^*-1} \cos \omega(D^*-1) + r^{D^*} \cos \omega D^*$$

$$0 = r^{D^*-1} \sin \omega(D^*-1) + r^{D^*} \sin \omega D^* \qquad (4)$$

which exist for all $D^* \in (\quad (R) \ , 1/2)$.

Proof:

For paths of length $\ell > 1$, let the j^{th} path cumulative metric be denoted $\underset{\sim}{\mu}_j$. Denote the metric of the j^{th} single branch μ_j . Observe the μ_j : all tree branches are I.I.D .

For some complex s , define

$$T_o(s) = \sum_{j=1}^{d} e^{s\mu_j} \qquad (6)$$

and define

$$T_\ell(s) = \sum_{j} e^{s\mu_j} f_j = \left[\sum_{j} e^{s\mu_j} \right]^{+1} \qquad (7)$$

where the $\sum$ is meant to run over all paths frozen and unfrozen at level ℓ .

f_j is defined in one of two ways:

1) If node j at level ℓ is frozen, we arbitrarily define there to be one extension to level $\ell+1$ with zero additional metric. Thus

$$f_j = 1 \cdot e^{s \cdot 0} = 1 \qquad (8)$$

2) If node j at level ℓ is not frozen, we define f_j to reflect d extensions with each branch having an I.I.D. μ_i . So

$$f_j \triangleq \sum_{i=1}^{d} e^{s\mu_i} \qquad (9)$$

Suppose s can be chosen such that $ET_o(s) = 1$ it will be seen that in our case s will exist and will be complex). By the I.I.D. property on the branch incremental metrics μ_i,

$$ET_o(s) = d\, E\left[e^{s\mu_i}\right] = 1 \tag{10}$$

It follows immediately that for all nodes frozen or not,

$$Ef_j = 1 \tag{11}$$

We now show $ET_\ell(s) \triangleq ET_{\ell-1}(s)$, thus proving by induction that

$$\lim_{\ell \to \infty} ET_\ell(s) = 1 \quad . \tag{12}$$

$$\text{Write } ET_\ell(s) = \underset{\text{(over } j)}{E}\left[E\left[\sum_j e^{s\underset{\sim}{\mu}_j} f_j \,\middle|\, j \text{ fixed}\right]\right]$$

$$= E\left[\sum_j e^{s\underset{\sim}{\mu}_j^{\ell}}\; E\; f_j\right]$$

$$= E\left[\sum_j e^{s\underset{\sim}{\mu}_j^{\ell}}\right] \qquad \text{by (11)}$$

$$= E\, T_{\ell-1}(s)$$

We can now rewrite (12), breaking up the sum into sums of paths frozen at $\underline{a}$, paths frozen at $\underline{b}$, and paths remaining active over an infinite length:

$$1 = \underset{\substack{j \\ \text{frozen} \\ \text{at a}}}{E\left[\sum e^{s\underset{\sim}{\mu}_j}\right]} + \underset{\substack{j \\ \text{frozen} \\ \text{at b}}}{E\left[\sum e^{s\underset{\sim}{\mu}_j}\right]} + \underset{\substack{j \\ \infty\text{ly} \\ \text{active}}}{E\left[\sum e^{s\underset{\sim}{\mu}_j}\right]} \tag{13}$$

Theorem 1 implies that the third term in (13) is zero so long as $|b-a| < \pi/\omega$. The first two terms are approximately

$$E\left[\sum_j e^{sa}\right] \quad \text{and} \quad E\left[\sum_j e^{sb}\right] \quad \text{respectively}$$

In actuality, frozen paths do not have precisely metrics $\underline{a}$ or $\underline{b}$, since paths may "overshoot" the barriers below freezing. The ambiguity in (14) may be resolved but only with tedious calculations, which will not appear here.

Thus (13) may be rewritten

$$1 \simeq EN_a e^{sa} + EN_b e^{sb} \tag{14}$$

If the value of s which satisfies (10) is expressed as

$$e^{s} = r[\cos\omega + i \sin \omega] \tag{15}$$

we can write (12) in real and imaginary parts,

$$1 \simeq EN_a r^a \cos\omega a + EN_b r^b \cos \omega b$$

$$0 \simeq EN_a r^a \sin \omega a + EN_b r^b \sin \omega b \tag{16}$$

(16) are simultaneous equations in two unknowns EN_a and EN_b . When solved, (16) yields the claimed result.

It remains to show that s exists satisfying (10), Now,

$$E[T_o(s)] = 2^{-n} d \sum_{k=0}^{n} \binom{n}{k} e^{s(nD^*-k)} = 1 \tag{17}$$

when the source and distortion measure are used to evaluate the expectation. (17) in turn reduces to

$$2^{1-R} = e^{s(D^*-1)} + e^{sD^*} \tag{1}$$

whose solution exists by Lemma 1. QED

APPENDIX 6

PROOFS AND ALGORITHMS FOR PERMUTATION CODING

A. Proof of Optimality of Encoding Procedure

Theorem: Let $f(|\alpha|)$ be any nonnegative, monotone nondecreasing, convex upward function of $|\alpha|$. Let the distance between the vector $\underline{X}^{(N)}=(X_1,X_2,\ldots,X_N)$ and $\underline{Y}^{(N)}=(Y_1,Y_2,\ldots,Y_N)$ be measured by

$$d(\underline{X}^{(N)},\underline{Y}^{(N)}) = \sum_{i=1}^{N} f(|X_i-Y_i|) \qquad \text{(A-1)}$$

Let $\underline{V}^{(N)} = (V_1,V_2,\ldots,V_N)$ be any vector for which $V_1 \geq V_2 \geq \ldots \geq V_N$ and let B be a block code whose codewords $\underline{Y}^{(N)}$ are all distinct permutations of $\underline{V}^N$. Then if X_{i_k} denotes the kth largest component of $\underline{X}^{(N)}$, the $\underline{Y}^{(N)} \in B$ that minimizes $d(\underline{X}^{(N)},\underline{Y}^{(N)})$ has $Y_{i_k} = V_k$ for $k = 1,2,\ldots,N$.

Proof: From the additive nature of Equation (A-1), it suffices to show that if $X_1 \geq X_2 \geq \ldots \geq X_N$, then $\underline{Y}^{(N)}= (V_1,V_2,\ldots,V_N) = \underline{V}^{(N)}$ is the $\underline{Y}^{(N)} \in B$ that minimizes $d(\underline{X}^{(N)},\underline{Y}^{(N)})$. Furthermore, once this has been established for $N = 2$, it is easily established for $N > 2$ by induction.

When $N = 2$, there are six cases to consider: namely

Case 1	$x_1 \geq v_1 \geq v_2 \geq x_2$	Case 4	$v_1 \geq x_1 \geq x_2 \geq v_2$
Case 2	$x_1 \geq v_1 \geq x_2 \geq v_2$	Case 5	$v_1 \geq x_1 \geq v_2 \geq x_2$
Case 3	$x_1 \geq x_2 \geq v_1 \geq v_2$	Case 6	$v_1 \geq v_2 \geq x_1 \geq x_2$

In each case we must establish that

$$f(|x_1-v_1|) + f(|x_2-v_2|) \leq f(|x_1-v_2|) + f(|x_2-v_1|) \qquad \text{(A-2)}$$

Since $f(\cdot)$ is a function of the absolute value of its argument, Cases 4,5, and 6 will follow immediately from the establishment of Equation (A-2) for Case (1), (2), and (3).

Case 1: We have $V_1-X_2 \geq V_2-X_2 \geq 0$ and $X_1-V_2 \geq X_1-V_1 \geq 0$.

Hence, (A-2) follows from the monotonicity of $f(\ )$.

Before treating Cases (2), and (3), we note that if we can establish Equation (A-2) for $\tilde{f}(|x-v|) = f(|x-v|) - f(0)$ then it will clearly hold for $f(\cdot)$ as well. Hence we lose no generality be assuming $f(0) = 0$.

Lemma For $a \geq 0$, $b \geq 0$, $f(a) + f(b) \leq f(a+b)$

Proof See Figure A-1. A straight line is drawn through the points $(a,f(a))$ and $(b,f(b))$. Since $f(\cdot)$ is convex upward and $f(0) = 0$, the line intersects the abscissa at a nonnegative value. Triangles T_1 and T_2 are similar. The base of T_2 is larger than the base of T_1 so the altitude of T_2 is larger than $f(a)$, the altitude of T_1. Thus the straight line intersects the point $(a+b,h)$ where

$$f(a) + f(b) \leq h \leq f(a+b) \quad .$$ QED

Case 2: We have

$$f(|x_1-v_1|) + f(|x_2-v_2|) \leq f(|x_1-x_2|) + f(|x_2-v_2|)$$

$$\leq f(|x_1-v_2|) \leq f(|x_1-v_2|) + f(|x_2-v_1|)$$

where the first inequality follows from $x_2 \leq v_1$ and monotonicity, the second from the lemma and the third from nonnegativity.

Case 3: We have

$$f(|x_2-v_2|) \leq \min\{f(|x_1-v_1|), f(|x_2-v_2|)\} \leq \max\{f(|x_1-v_1|), f(|x_2-v_2|)\}$$

$$\leq f(|x_1-v_2|)$$

Let $f(|\alpha|) = f(|\alpha + x_2 - v_1|) - f(|x_2 - v_1|)$. Applying the lemma to $\tilde{f}(\cdot)$ yields

$$f(|x_1 - v_1|) + f(|x_2 - v_2|) = \tilde{f}(|x_1 - x_2|) + \tilde{f}(|v_1 - v_2|) + 2f(|x_2 - v_1|)$$

$$\leq \tilde{f}(|x_1 - x_2 + v_1 - v_2|) + 2f(|x_2 - v_1|)$$

$$= f(|x_1 - v_2|) + f(|x_2 - v_1|) \qquad \text{QED}$$

B. Best Choice of μ_i for Mean-Square Error Criterion

Let $X^{(i)}$ denote the ith largest of N random variables, each with mean zero and variance σ^2. Let $|X|^{(i)}$ denote the ith largest of the absolute value of these random variables. Then the mean-squared error for Variant I and Variant II codes are:

$$\text{Variant I} \quad D = E\left\{\sum_{j=1}^{K}\left(\sum_{i=n_1+\ldots n_{j-1}+1}^{n_1+\ldots+n_j} (X^{(i)} - \mu_j)^2\right)\right\} \qquad \text{(B-1)}$$

$$\text{Variant II} \quad D = E\left\{\sum_{j=1}^{K}\left(\sum_{i=n_1+\ldots+n_{j-1}+1}^{n_1+\ldots+n_j} (|X|^{(i)} - \mu_j)^2\right)\right\} \qquad \text{(B-2)}$$

Noting that

$$\sum_{i=1}^{N} E\left\{\left(X^{(i)}\right)^2\right\} = \sum_{i=1}^{N} E\left\{\left(|X|^{(i)}\right)^2\right\} = \sigma^2$$

these equations can be rewritten as

$$\text{Variant I} \quad D = \sigma^2 - 2\sum_{j=1}^{K} \mu_j \left(\sum_{i=n_1+\ldots+n_{j-1}+1}^{n_1+\ldots+n_j} X^{(i)}\right) = \sum_{j=1}^{K} n_j \mu_j^2 \qquad \text{(B-3)}$$

$$\text{Variant II} \quad D = \sigma^2 - 2 \sum_{j=1}^{K} \mu_j \left(\sum_{i=n_1+\ldots+n_{j-1}+1}^{n_1+\ldots+n_j} |X|^{(i)} \right) + \sum_{j=1}^{K} n_j \mu_j^2 \qquad \text{(B-4)}$$

Differentiating Equations (A-3) and (A-4) with respect to μ_j, setting the result equal to zero and solving for μ_j results in the expressions given in Equations (11) and (12).

C. Monotonicity of n_i for Minimum Distortion

Let α_i be the appropriate ith coder statistic for Variant I or Variant II codes. That is,

$$\alpha_i = \begin{cases} E\left\{X^{(i)}\right\} & \text{Variant I} \\ E\left\{|X|^{(i)}\right\} & \text{Variant II} \end{cases} \qquad \text{(C-1)}$$

Then from Equations (11) and (12), the optimum values of the μ_j which minimize the mean-square error are

$$\mu_j = \frac{1}{n_j} \sum_{i=n_1+\ldots+n_{j-1}+1}^{n_1+n_2+\ldots+n_j} \alpha_i \qquad j = 1,2,\ldots,K \qquad \text{(C-2)}$$

and the resulting mean-square error distortion (from Equation (13)) is

$$D = \sigma^2 - \frac{1}{N} \sum_{j=1}^{K} n_j \mu_j^2 \qquad \text{(C-3)}$$

Choose an ℓ such that $\alpha_i > 0$, $i = 1,2,\ldots,n_1+n_2+\ldots+n_\ell$, and let

$$a = n_{\ell-1} > n_\ell = b \qquad \text{(C-4)}$$

It will now be shown that if all the other n_i ($i \neq \ell-1$ or ℓ) remain fixed, the distortion given by (C-3) can be made smaller by reversing the

roles of $n_{\ell-1}$ and n_{ℓ}. That is, define a new set of groupings n_i', given as

$$\begin{aligned} n_i' &= n_i \qquad i \neq \ell-1 \text{ or } \ell \\ n_\ell' &= n_{\ell-1} \\ n_{\ell-1}' &= n_\ell \end{aligned} \tag{C-5}$$

Then

$$D' = \sigma^2 - \frac{1}{N}\sum_{j=1}^{K} n_j'(\mu_j')^2 < \sigma^2 - \frac{1}{N}\sum_{j=1}^{K} n_j \mu_j^2 = D \tag{C-6}$$

Proof

Let $L = n_1+n_2+\ldots+n_{\ell-1}$. Then $D-D'$ can be written as

$$D-D' = \frac{1}{a}(\alpha_{L+1}+\ldots+\alpha_{L+a})^2 + \frac{1}{b}(\alpha_{L+1+a}+\ldots+\alpha_{L+a+b})^2$$

$$= \frac{1}{b}(\alpha_{L+1}+\ldots+\alpha_{L+b})^2 + \frac{1}{a}(\alpha_{L+b+1}+\ldots+\alpha_{L+a+b})^2 \tag{C-7}$$

After some manipulation, this can be written as

$$D-D' = \frac{1}{ab}\left[(\alpha_L+\ldots+\alpha_{L+a})-(\alpha_{L+b+1}+\ldots+\alpha_{L+a+b})\right] \times$$

$$\left[(b-a)(\alpha_{L+1}+\ldots+\alpha_{L+a})+(b-a)(\alpha_{L+b+1}+\ldots+\alpha_{L+a+b})-a(\alpha_{L+a+1}+\ldots+\alpha_{L+b})\right] \tag{C-8}$$

Now

$$(\alpha_L+\ldots+\alpha_{L+a}) \geq (\alpha_{L+b+1}+\ldots+\alpha_{L+a+b}) \tag{C-9}$$

so the first bracket is nonnegative. The following inequalities establish that the second bracket is nonnegative:

$$(b-a)(\alpha_{L+1}+\ldots+\alpha_{L+a}) \geq (b-a)a\,\alpha_{L+a} \tag{C-10}$$

$$(b-a)(\alpha_{L+b+1}+\ldots+\alpha_{L+a+b}) \geq (b-a)a\ \alpha_{L+a+b} \tag{C-11}$$

$$a(\alpha_{L+a+1}+\ldots+\alpha_{L+b}) \leq (b-a)a\ \alpha_{L+a+1} \tag{C-12}$$

The second bracket in Equation (C-8) is then bounded from below by

$$\left[\quad\right] \geq (b-a)a\left[\alpha_{L+a}+\alpha_{L+a+b}-\alpha_{L+a+1}\right] \geq 0 \tag{C-13}$$

Thus $D - D' \geq 0$, as was to be proved.

D. Algorithm that Determines Almost Optimal Grouping $\{n_1,n_2,\ldots,n_k\}$ for Permutation Codes

1. Choose N and R.
2. Initially set K as the smallest even integer such that $\log_2 K \geq R$
3. Initially set the groupings to be approximately equal. (If K divides N set $n_i = N/K$ for all i.)
4. Compute $u_1,u_2,\ldots,u_K$
5. Set $\beta = 1$. Solve Equation (20) for p_i. Adjust β until Equation (18)is satisfied for the desired rate.
6. Compute n_i as the closest integers to p_iN such that $\sum_{i=1}^{K} n_i = N$
7. Test if any $n_i = 0$. If yes, proceed to step 11. If not, proceed to step 8.
8. Test if new set of n_i agree with old set of n_i. If yes, proceed to step 9. If no, go back to step 4.
9. Store $n_1,n_2,\ldots,n_K$, and the exact values of R and D corresponding to this partitioning.
10. Let $K \longrightarrow K+2$ and start with new grouping closely approximating grouping stored in step 9. (For Variant I codes, let $n_1,=n_K=1$ and $n_2,n_3,\ldots,n_{K-1}$ be the same as the grouping stored in 9 except that the largest n_i has been reduced by 2.) Return to step 4.

11. Print $n_1, n_2, \ldots, n_k$; R and D stored in step 9. If K is odd go to step 14.

12. Set K as smallest odd integer such that $\log_2 K \geq R$

13. Return to step 3.

14. Stop.

E. Binary Coding of Permutations Encoding Algorithms

1. $\pi \leftarrow \frac{n_1!\, n_2! \ldots n_K!}{N!}$

$P \leftarrow \frac{1}{N}$

$I(i) \leftarrow n_i \quad i = 1,2,\ldots,K$

$I(0) \leftarrow 0$

$\ell \leftarrow 0$

2. $\ell \leftarrow \ell+1$

3. $\pi \leftarrow \pi + P \sum_{i=0}^{j_\ell - 1} I(i)$

4. If $\ell = N-1$, go to (8) . Otherwise continue

5. $P \leftarrow P \frac{I(j_\ell)}{N-\ell}$

6. $I(j_\ell) \leftarrow I(j_\ell) - 1$

7. Go to 2

8. $\ell \leftarrow 0$

9. $\ell \leftarrow \ell+1$

10. If $\pi < 2^{-\ell}$, $s_\ell \leftarrow 0$, otherwise ($s_\ell \leftarrow 0$ and $\pi \leftarrow \pi - 2^{-\ell}$)

11. If $\ell < Q$ go to (8). Otherwise stop.

Decoding Algorithm

1. $P \leftarrow N \sum_{i=1}^{Q} s_i 2^{-i}$

   $I(i) = n_i \quad i = 1,2,\ldots, K$

   $\ell \leftarrow 0$

2. $\ell \leftarrow \ell + 1$

3. $R \leftarrow 0$

   $i \leftarrow 0$

4. $i \leftarrow i+1$

5. $R \leftarrow R + I(i)$

6. If $R \leq P$, go to (4), otherwise continue..

7. $j_\ell \leftarrow i$

8. If $\ell < N-1$ continue, otherwise go to (12).

9. $P \leftarrow (P-R + I(j_\ell))\ (N-\ell)/I(j_\ell)$

10. $I(j_\ell) \leftarrow I(j_\ell)-1$

11. Go to (2)

12. $I(j_\ell) \leftarrow I(j_\ell)-1$

13. $i \leftarrow 0$

14. $i \leftarrow i + 1$

15. If $I(i) = 0$, go to (14), otherwise continue

16. $j_N \leftarrow i$

17. Stop.