#3

THRESHOLD DECODING

JAMES L. MASSEY

LOAN COPY

only

TECHNICAL REPORT 410

APRIL 5, 1963

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
RESEARCH LABORATORY OF ELECTRONICS
CAMBRIDGE, MASSACHUSETTS

The Research Laboratory of Electronics is an interdepartmental laboratory in which faculty members and graduate students from numerous academic departments conduct research.

The research reported in this document was made possible in part by support extended the Massachusetts Institute of Technology, Research Laboratory of Electronics, jointly by the U.S. Army, the U.S. Navy (Office of Naval Research), and the U.S. Air Force (Office of Scientific Research) under Contract DA36-039-sc-78108, Department of the Army Task 3-99-25-001-08; and in part by Contract DA-SIG-36-039-61-G14.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Technical Report 410 April 5, 1963

THRESHOLD DECODING

James L. Massey

Submitted to the Department of Electrical Engineering,
M. I. T., August 20, 1962, in partial fulfillment of the
requirements for the degree of Doctor of Philosophy.

(Manuscript received August 31, 1962)

Abstract

Two procedures for decoding linear systematic codes, majority decoding and a posteriori probability decoding, are formulated. The essential feature of both methods is a linear transformation of the parity-check equations of the code into "orthogonal parity checks." The decoding decisions are then made on the basis of the values assumed by these orthogonal parity checks. For binary codes, the principal component required in the circuitry for instrumenting these decoding rules is an ordinary threshold logical element. For this reason, we refer to these. decoding rules as forms of "threshold decoding."

It is shown that threshold decoding can be applied effectively to convolutional codes up to approximately 100 transmitted bits in length over an interesting range of rates. Very simple decoding circuits are presented for such codes. However, it is also shown that the probability of error at the receiver cannot be made as small as desired by increasing the length of the code that is used with threshold decoding, rather this probability approaches a nonzero limit as the code length is increased indefinitely. A large number of specific convolutional codes, suitable for threshold decoding, are tabulated. Some of these codes are obtained by hand construction and others by analytical techniques.

It is shown that threshold decoding is applicable to certain low-rate block codes, and that a generalization of the method is applicable to several other classes of block codes. It is shown that simple decoding circuits can be used for such codes. The theoretical limits of threshold decoding with block codes are still not clear, but the results presented here are promising.

TABLE OF CONTENTS

CONTENTS

CONTENTS

GLOSSARY

Symbol	Definition
A_i	An orthogonal parity check on some noise digit, say e_m
B_i	Sum of the received digits, r_m excluded, checked by A_i
D	Delay operator
$G^{(j)}(D)$	Code-generating polynomial in a convolutional code
J	Number of orthogonal parity checks
k	Number of information symbols in a code word
k_o	Number of information symbols per time unit in a convolutional code
L	Number of steps required to orthogonalize a block code
m	Degree of code-generating polynomials
n	Number of symbols in a code word
n_o	Number of symbols per time unit in a convolutional code
n_A	Actual constraint length of a convolutional code
n_E	Effective constraint length of a convolutional code
n_i	Number of symbols checked by B_i
p_o	$\Pr[e_m=1]$, where e_m is the bit to be determined by decoding
p_o	Transition probability of a binary symmetric channel
p_i	Probability of an odd number of errors in the bits checked by B_i
p	Erasure probability of a binary erasure channel
$P_1(e)$	Error probability in determining set of first information symbols of a convolutional code

GLOSSARY

Symbol	Definition
q	Number of elements in a finite field or GF(q)
T	Threshold
w_i	Weighting factor
$\binom{a}{b}$	Binomial coefficient
$\lfloor I \rfloor$	Greatest integer equal to or less than I
$\lceil I \rceil$	Least integer equal to or greater than I
$\{X_i\}$	Set of all elements X_i, where the index i runs over all elements in the set
$\{f(D)\}$	Polynomial obtained from f(D) by dropping all terms with power of D greater than m.

I. THE CONCEPT OF THRESHOLD DECODING

In 1948, Shannon first demonstrated that errors in the data transmitted over a noisy channel can be reduced to any desired level without sacrificing the data rate.[1] His "Noisy Coding Theorem" stated that such a channel is characterized by a quantity C, called the channel capacity, so that the error probability at the receiver can be made arbitrarily small by proper encoding and decoding of the data when, and only when, the rate R of information transmission is less than C. With the goals and limitations of their art thus clearly delineated, communication engineers have since focused considerable, and often ingenious, effort on the dual problems that stand in the way of full exploitation of Shannon's theorem, along the following lines:

(i) the construction of good codes that are readily instrumented, and

(ii) the development of simple and efficient decoding apparatus for such codes.

The first of these problems constitutes, by itself, no real obstacle. Let us for convenience confine our discussion at the moment to binary codes and transmission through a memoryless binary symmetric channel. A mathematical model of such a channel is shown in Fig. 1. When either a "one" or a "zero" is transmitted over this channel, it is received correctly with probability q_o and incorrectly with probability $p_o = 1 - q_o$. The channel capacity can be shown to be $1 - H(p_o)$ bits per transmitted symbol, where $H(x) = -x \log_2 x - (1-x) \log_2 (1-x)$ is the entropy function.[2] Each message to be transmitted over this channel will be encoded into a block of n binary digits. The number of allowable messages is 2^{nR} where R is the rate of information transmission in bits per transmitted symbol when the input messages are all equiprobable.

In proving his "noisy coding theorem" for the binary symmetric channel, Shannon showed that when R is less than C, the average error probability vanishes with increasing n for the ensemble of binary block codes in which each of the 2^{nR} message sequences is chosen at random with equal probability from the set of 2^n binary sequences of length n. Since a code taken at random from such an ensemble has probability less than $1/N$ of having error probability more than N times the ensemble average, one can quite reasonably select a good code at random. The encoding apparatus would, however, be prohibitively complex since each of the 2^{nR} code words, of n bits each, would have to be stored.

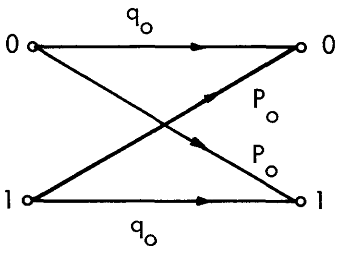


Fig. 1. Binary symmetric channel.

Elias[3] has shown that the much smaller ensemble of sliding parity-check codes has the same average probability of error as the ensemble when all the code words are selected at random and, furthermore, that this error probability decreases exponentially with the block length n with the optimum exponent for rates near C. Fano[4] has shown that a linear sequential network containing only n - 1 stages of shift register can serve as an encoder for a sliding parity-check code. Recently, Wozencraft[5] has found an even smaller ensemble of codes, which we shall call the randomly shifted codes, that again have the same ensemble average probability of error as the other random codes. Wozencraft's codes can be encoded by a linear sequential network containing the maximum of Rn or (1-R)n stages of shift register (cf. section 2.5). From either the ensemble of sliding parity-check codes or the ensemble of randomly shifted codes, it is thus quite feasible to select a good, and readily instrumented, code by a random choice. However, at present, no way is known to solve the second coding problem for these ensembles, that is, the construction of a simple and efficient decoder. (The complications of the general decoding problem will be discussed below.)

Remarkably, efficient decoding procedures have been found for certain ensembles of randomly selected codes. The sequential decoding technique of Wozencraft was devised for the ensemble of convolutional codes.[6] Wozencraft and Reiffen have demonstrated that for this method an exponential decrease in error probability can be attained by a decoder whose complexity and average number of computations increase by only a small power of the code length.[7] Gallager has found a similar result to apply to his iterative procedure for decoding the ensemble of parity-check codes that are constrained to have a low density of "ones" in the parity-check matrix.[8] For both of these schemes, the decoding effort is small only when the information transmission rate is less than some "computational rate" which is less than channel capacity. These two decoding procedures (and the modified forms of sequential decoding recently proposed by Ziv[9] and Fano[10]) are the only known general solutions to the dual coding problems described above. Both of these methods have the disadvantages that the amount of computation per decoded digit is a random variable, and, for short codes, the decoders are more complex than those that can be constructed for certain specific codes by using other techniques.

A second approach to the coding problems stands in sharp contrast to the probabilistic approach taken by Wozencraft and Gallager. Based on the pioneering work of Hamming[11] and Slepian,[12] the algebraists take as their starting point a formal mathematical structure that is demonstrated to have metric properties that are desirable for coding purposes. Often the decoding methods are found after the codes have been known for some time, as is illustrated most strikingly by the Reed algorithm[13] for the Muller codes,[14] and the Peterson[15] algorithm for the Bose-Chaudhuri codes.[16] Because of the systematic structure of the codes obtained by the algebraic approach, the decoding procedures usually require a fixed amount of computation. Very likely, for the same reason, the codes of any one known type always become poor as the code length is increased with the rate held constant. (The single exception to this statement is the iterated coding of Elias[17] which does succeed in making the error probability vanish, but more slowly than the optimum exponential rate with block length, and only for rates that are substantially less than channel capacity.)

1.1 THRESHOLD DECODING OF LINEAR CODES

Against this background, we can describe how the concept of threshold decoding which will be introduced in this report is related to the previous approaches to the coding problems. As in the probabilistic approach, we shall begin with a decoding algorithm rather than with specific codes. However, our algorithm will be primarily algebraic and will require a fixed amount of computation per decoded symbol. Because of the special algebraic code structure which it requires, the algorithm will not be applicable to whole ensembles of codes; rather, specific codes to which it applies will have to be constructed. Finally, and most important, we choose as our algorithm a procedure that lends itself to simple machine implementation. Before outlining the decoding procedure, we shall present some preliminary remarks on linear, or group, coding.

a. Linear Codes and the Decoding Problem[18]

We shall be concerned primarily with linear codes in systematic form. The set of code words for such a code is a subset of the set of n-tuples of the form

$$(t_1, t_2, \dots, t_n), \qquad (1)$$

where each t_i is an element of GF(q), the finite field of q elements (cf. Appendix A). The symbols $t_1, t_2, \dots, t_k$ are chosen to be the information symbols. The remaining n-k symbols are called the parity symbols and are determined from the information symbols by a set of linear equations

$$t_j = \sum_{i=1}^{k} c_{ji} t_i \qquad j = k+1, k+2, \dots, n \qquad (2)$$

where the set of coefficients, c_{ji}, are elements of GF(q) specified by a particular code. (All operations on these symbols are to be performed in GF(q) unless expressly stated otherwise.)

We assume now that after transmission through some channel, a received n-tuple $(r_1, r_2, \dots, r_n)$ is obtained which differs from the transmitted n-tuple $(t_1, t_2, \dots, t_n)$ by a noise sequence $(e_1, e_2, \dots, e_n)$, that is

$$r_i = t_i + e_i \qquad i = 1, 2, \dots, n, \qquad (3)$$

where again r_i and e_i are elements of GF(q), and all arithmetic operations are carried out in this field. It then follows from Eq. 3 that knowledge of the received sequence and the noise sequence suffices to determine the transmitted sequence.

It can be readily verified that the set of transmitted code words form an additive

Abelian group of q^k members (cf. Appendix A for the defining axioms of such a group). There are q^k code words, since each of the k information symbols can have any one of q values. Let $(t_1, t_2, \ldots, t_n)$ and $\left(t_1^*, t_2^*, \ldots, t_n^*\right)$ be any two code words. Then, by Eq. 2, their sum is the n-tuple $\left(t_1+t_1^*, t_2+t_2^*, \ldots, t_n+t_n^*\right)$, where

$$t_j + t_j^* = \sum_{i=1}^{k} c_{ji}\left(t_i+t_i^*\right) \qquad j = k+1, k+2, \ldots, n \tag{4}$$

and this is the same n-tuple as is obtained by encoding $t_i+t_i^*$, $i = 1, 2, \ldots, k$ as information symbols. Thus the first group axiom is satisfied, and the other axioms are satisfied trivially.

We shall always use the notation, an (n, k) code, to mean a linear systematic code as defined by Eq. 2.

Equation 2 may be rewritten as

$$\sum_{i=1}^{k} c_{ji}t_i - t_j = 0 \qquad j = k+1, k+2, \ldots, n \tag{5}$$

and we say that each of these n-k equations defines a parity set for the code, that is, some weighted sum of code symbols which is zero for all code words.

We shall define a parity check to be the sum in Eq. 4 formed at the receiver, that is,

$$s_j = \sum_{i=1}^{k} c_{ji}r_i - r_j \qquad j = k+1, k+2, \ldots, n. \tag{6}$$

Using Eqs. 3 and 5, we may rewrite the parity checks as

$$s_j = \sum_{i=1}^{k} c_{ji}e_i - e_j \qquad j = k+1, k+2, \ldots, n \tag{7}$$

from which we see that the $\{s_j\}$ constitute a set of n-k linear equations in the n unknowns $\{e_i\}$. (We shall use the notation $\{a_i\}$ to mean the set of objects a_i, where i ranges over all indices for which the objects are defined.) The general solution of Eq. 7 can be written immediately as

$$e_j = \sum_{i=1}^{k} c_{ji}e_i - s_j \qquad j = k+1, k+2, \ldots, n. \tag{8}$$

This general solution has k arbitrary constants, namely the values of $e_1, e_2, \ldots, e_k$. Each of these arbitrary constants can be assigned any of q values, and thus there are q^k distinct solutions of Eq. 7.

We have not yet considered the mechanism by which the noise symbols $\{e_i\}$ are generated. The $\{e_i\}$ can be considered as sample points from the random process described by the communication channel. For instance, it is readily seen that the binary symmetric channel in Fig. 1 is fully equivalent to the model in Fig. 2 for which the noise source is an independent letter source that has probability p_o of giving a "one" output, and probability q_o of giving a "zero" output. In this case, a noise sequence of n bits that contains w "ones" has a probability of occurrence of $p_o^w q_o^{n-w}$ and this is a monotonically decreasing function of w, the number of errors in the received set of n bits $\left(\text{we assume } p_o \leq \frac{1}{2}\right)$.

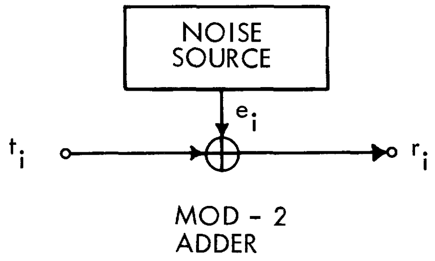


Fig. 2. Noise-source model of the Binary Symmetric Channel.

The general decoding problem for linear codes is to find that solution of Eq. 7 that is most probable from consideration of the channel. For example, when binary data are transmitted over a binary symmetric channel, the problem is to find that solution of Eq. 7 that contains the smallest number of "ones." In practice, it is generally not feasible to find the most probable solution of Eq. 7 for an arbitrary parity-check pattern $\{s_i\}$, simply because of the enormous number of possibilities. An efficient solution of the decoding problem depends on finding a simple method for determining the most probable solution of Eq. 7 for a high probability subset of the set of all possible parity check patterns.

b. Orthogonal Parity Checks

We shall now consider a procedure by which the parity checks $\{s_j\}$ of Eq. 6 can be transformed into another set of quantities that will be found more convenient for decoding purposes. We begin by considering linear combinations of the $\{s_j\}$. We define a composite parity check, A_i, to be a linear combination of the $\{s_j\}$. That is, each A_i is given by an equation

$$A_i = \sum_{j=k+1}^{n} b_{ij} s_j, \tag{9}$$

where the coefficients, b_{ij}, are again elements of GF(q).

From Eqs. 7 and 9, we have

$$A_i = \sum_{j=k+1}^{n} b_{ij} \left[\sum_{h=1}^{k} c_{jh} e_h - e_j \right] \tag{10}$$

which may be written

$$A_i = \sum_{j=1}^{n} a_{ij} e_j, \tag{11}$$

where

$$a_{ij} = \begin{cases} \displaystyle\sum_{h=k+1}^{n} b_{ih} c_{hj} & j = 1, 2, \dots, k \\ \\ b_{ij} & j = k+1, k+2, \dots, n. \end{cases} \tag{12}$$

It is now convenient to make the following definition.

DEFINITION: A set of J composite parity checks, $\{A_i\}$, is said to be *orthogonal on* e_m if in Eq. 11

$$a_{im} = 1 \qquad i = 1, 2, \dots, J \tag{13}$$

and

$$a_{ij} = 0 \qquad \text{for all, but at most one, index } j \text{ different from } m \text{ for any fixed } i. \tag{14}$$

In other words, a set of J composite parity checks is called orthogonal on e_m if e_m is checked by each member of the set, but no other noise digit is checked by more than one member of the set. Thus e_m is able to affect all of the equations in the set, but no other noise bit can affect more than a single equation in the set.

c. Majority Decoding

We shall now give the first of two algorithms that can be used to determine e_m from a set of J parity checks $\{A_i\}$ orthogonal on e_m. We shall use the notation $\lfloor I \rfloor$ and $\lceil I \rceil$ to mean the greatest integer equal to or less than I and the least integer equal to or greater than I respectively, and we shall say that a noise bit is checked by some parity check if and only if it appears with a nonzero coefficient in the equation for that parity check.

THEOREM 1: Provided that $\lfloor J/2 \rfloor$ or fewer of the $\{e_j\}$ that are checked by a set of J parity checks $\{A_i\}$ orthogonal on e_m are nonzero (that is, there are $\lfloor J/2 \rfloor$ or fewer errors in the corresponding received symbols), then e_m is given correctly as that value of GF(q) which is assumed by the greatest fraction of the $\{A_i\}$. (Assume $e_m = 0$ in the case for which no value is assumed by a strict plurality of the $\{A_i\}$, and 0 is one of the several values with most occurrences.)

PROOF 1: Suppose $e_m = V$ and assume initially that all other e_j that are checked have value 0. Then from Eqs. 11 and 13 it follows that

$$A_i = V \qquad i = 1, 2, \ldots, J. \tag{15}$$

Now suppose $V = 0$ and the conditions of the theorem are satisfied. The $\lfloor J/2 \rfloor$ nonzero noise digits can change at most the same number of the equations in Eq. 15 and hence at least $\lceil J/2 \rceil$ of the $\{A_i\}$ are still zero. Thus zero is either the value with most occurrences in the set $\{A_i\}$ or one of the two values with the same largest number of occurrences; in both cases the decoding rule of the theorem gives the correct value of e_m.

Conversely, suppose that $V \neq 0$. Then since $e_m = V \neq 0$, it follows that fewer than $\lfloor J/2 \rfloor$ of the other noise digits checked by the $\{A_i\}$ are nonzero, and hence that more than $\lceil J/2 \rceil$ of the equations in Eq. 15 are still correct. Hence the decoding rule of the theorem is again correct, and the theorem is proved.

If $J = 2T$ is an even number, it follows from Theorem 1 that e_m can be correctly determined whenever T or fewer of the received symbols are in error. Similarly if $J = 2T + 1$ is an odd number, then again e_m can be found whenever T or fewer errors occur, and in addition $T + 1$ errors can be detected by saying that a detectable error has occurred when the value assumed by the greatest fraction of the $\{A_i\}$ is nonzero and has exactly $T + 1$ occurrences. These considerations imply that any two code words with different values of t_m must be at least distance $J + 1$ apart. (We shall use distance to mean always Hamming distance, that is, the number of symbols in which the code words differ.) Hence, Theorem 1 has the following corollary.

COROLLARY: Given a linear code for which it is possible to form a set of J parity checks orthogonal on e_m, then any two code words for which t_m differs are separated by a Hamming distance of at least $J + 1$.

We shall refer to decoding performed according to the algorithm given in Theorem 1 as majority decoding of orthogonal parity checks. It should be clear from the preceding discussion that majority decoding is a form of minimum distance decoding; that is, when decoding e_m, majority decoding assigns the value to e_m that it takes on in the noise sequence of minimum weight that satisfies Eqs. 11. It is important to note that this noise sequence does not in general coincide with the noise sequence of minimum weight that satisfies Eqs. 7, since the mapping from the $\{s_j\}$ to the $\{A_i\}$ need not be one-to-one.

d. A Posteriori Probability Decoding

Majority decoding is inefficient in the sense that it does not take into account the details of the channel statistics, that is, of the probability distributions of the noise bits. We shall now give a decoding algorithm that makes the best possible use of the information contained in a set of J parity checks orthogonal on e_m in arriving at a decision on the value of e_m.

We assume the noise sequence is additive and is independent from digit-to-digit, but is not otherwise restricted. This means that the channel must be able to be put in the form of the model in Fig. 2 in which the noise source is an independent letter source and the adder operates in GF(q).

Taking average probability of error as the criterion of goodness, we seek a decoding algorithm that will assign to e_m that value V for which the conditional probability

$$\Pr(e_m=V\,|\,\{A_i\}) \tag{16}$$

is a maximum. Using Bayes' rule, we have

$$\Pr(e_m=V\,|\,\{A_i\}) = \frac{\Pr(\{A_i\}\,|\,e_m=V)\,\Pr(e_m=V)}{\Pr(\{A_i\})}. \tag{17}$$

Because of the orthogonality on e_m of the $\{A_i\}$ and the digit-to-digit independence of the noise sequence, it follows that

$$\Pr(\{A_i\}\,|\,e_m=V) = \prod_{i=1}^{J} \Pr(A_i\,|\,e_m=V). \tag{18}$$

Substituting Eqs. 17 and 18 in Eq. 16 and taking logarithms, we can phrase the decoding rule as follows: Choose e_m to be that value V for which

$$\log\,[\Pr(e_m=V)] + \sum_{i=1}^{J} \log\,[\Pr(A_i\,|\,e_m=V)] \tag{19}$$

is a maximum. For emphasis, we state this result as a theorem.

THEOREM 2: Given a set $\{A_i\}$ of J parity checks orthogonal on e_m, and that the noise sequence is additive with digit-to-digit independence, then the decoding rule based on $\{A_i\}$ which determines e_m with the least average probability of error is: Choose e_m to be that value V of GF(q) for which

$$\log\,[\Pr(e_m=V)] + \sum_{i=1}^{J} \log\,[\Pr(A_i\,|\,e_m=V)]$$

is a maximum.

We shall refer to decoding performed according to the algorithm of Theorem 2 as a posteriori probability decoding of orthogonal parity checks, or, more simply, as APP decoding.

e. Threshold Decoding

Let us now consider the specialization of the majority decoding and APP decoding algorithms to the binary case. Let $p_o = 1 - q_o$ be the error probability for bit e_m, that is,

$$\Pr(e_m=1) = p_o. \tag{20}$$

If $p_i = 1 - q_i$ denotes the probability of an odd number of "ones" among the noise bits exclusive of e_m that are checked by A_i, the i^{th} parity check orthogonal on e_m, then it follows that

$$\Pr(A_i=0 \mid e_m=1) = \Pr(A_i=1 \mid e_m=0) = p_i \tag{21}$$

and

$$\Pr(A_i=1 \mid e_m=1) = \Pr(A_i=0 \mid e_m=0) = q_i. \tag{22}$$

Since e_m can have only two possible values, 0 or 1, it follows that the APP decoding rule is simply: Choose $e_m = 1$ if, and only if,

$$\log (p_o) + \sum_{i=1}^{J} \log [\Pr(A_i \mid e_m=1)] > \log (q_o) + \sum_{i=1}^{J} \log [\Pr(A_i \mid e_m=0)]. \tag{23}$$

Using Eqs. 21 and 22, we can reduce (23) to

$$\sum_{i=1}^{J} (2A_i-1) \log (q_i/p_i) > \log (q_o/p_o) \tag{24}$$

or

$$\sum_{i=1}^{J} A_i[2 \log (q_i/p_i)] > \sum_{i=0}^{J} \log (q_i/p_i), \tag{25}$$

where the $\{A_i\}$ are treated as real numbers in (24) and (25). We summarize these results in the next theorem.

THEOREM 3: For a binary memoryless channel with additive noise, the APP decoding rule becomes: Choose $e_m = 1$ if, and only if, the sum of the members of the set $\{A_i\}$ of J parity checks orthogonal on e_m, treated as real numbers and weighted by the factor $2 \log (q_i/p_i)$, exceeds the threshold value

$$\sum_{i=0}^{J} \log (q_i/p_i),$$

where $p_o = 1 - q_o = \Pr(e_m=1)$ and $p_i = 1 - q_i$ is the probability of an odd number of errors in the symbols, exclusive of e_m, that are checked by the i^{th} orthogonal parity check.

In a similar way, Theorem 1 for the binary case reduces to the following theorem.

THEOREM 4: Given a set $\{A_i\}$ of J parity checks orthogonal on e_m, then the majority decoding rule is: Choose $e_m = 1$ if, and only if, the sum of the A_i (as real numbers) exceeds the threshold value $\lceil 1/2\ J \rceil$.

Because of the similarity of the decoding rules of Theorems 3 and 4, and because these decoding rules can be instrumented by means of a simple threshold logical element, we use the generic term, threshold decoding, to describe either majority decoding or APP decoding of orthogonal parity checks. For convenience, we shall use the same nomenclature for nonbinary decoding.

1.2 SUMMARY

By considering only orthogonal parity checks rather than the entire set of ordinary parity checks, the general decoding problem described in section 1.1a can be reduced to the simple forms given by Theorems 1-4. The reasons for this simplification are: In general, there is a very complex relationship between the values of the s_j in Eq. 6 and the corresponding most probable noise sequence e_i, $i = 1, 2, \dots, n$. However, if the s_j are linearly transformed into a set of parity checks orthogonal on e_m, there is a very simple relationship between the set $\{A_i\}$ of orthogonal parity checks and e_m – the A_i conditioned on e_m are a set of independent random variables. Thus the factorization of Eq. 18 can be carried out, and this permits each of the orthogonal parity checks to be treated separately in the process of computing the most probable value of e_m.

It remains to be shown that there exist codes for which the mapping from the ordinary parity checks to the set of orthogonal parity checks can be carried out in an efficient manner, that is, that the most probable value of e_m obtained by one of the threshold decoding algorithms will coincide with the most probable value of e_m with respect to the entire set of ordinary parity checks for a high probability subset of possible values of the ordinary parity checks. The rest of this report will be devoted to this task.

In formulating the concept of threshold decoding, we have not restricted the choice of a finite field for the coded symbols. We shall hereafter restrict ourselves almost exclusively to the binary field, for two reasons: because this is a case of practical interest, and because, as we shall point out, there are difficulties in applying the method of threshold decoding to nonbinary codes in an efficient manner.

II. CONVOLUTIONAL ENCODING AND PARITY CHECKING

We shall now give a brief discussion of convolutional encoding that will include algebraic properties of such codes, bounds on code quality, and circuits for encoding and parity checking. Although this section is intended primarily to serve as a background for the following ones, some of the material presented is new. Sections 2.2, 2.4, and 2.7 represent original work. The random-coding bound of section 2.5 was first derived by Elias[3] (who introduced convolutional coding at the same time), and we present merely a different method of proof. The Gilbert bound in section 2.3 was shown to hold for binary convolutional codes with rate $1/n_o$ by Wozencraft[19]; we give a proof that applies to nonbinary codes and to codes with arbitrary rate. The encoding circuit of section 2.6a was first described by Wozencraft and Reiffen,[20] but the circuit of section 2.6b appears to be new.

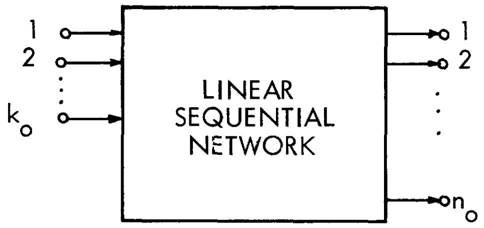


Fig. 3. General convolutional encoder.

2.1 CONVOLUTIONAL ENCODING

A general convolutional encoder is shown in Fig. 3. Each unit of time, an information symbol enters each of the k_o input lines, and an encoded symbol leaves each of the n_o ($n_o \geq k_o$) output lines. The symbols can be elements in any finite field, GF(q), and the rate $R = k_o/n_o$ then has the units of $\log_2 q$ bits per output symbol. All operations on the symbols are assumed hereafter to be carried out in GF(q) unless it is expressly stated to the contrary.

Using the delay operator, or D-notation, introduced by Huffman,[21] we can represent the k_o input sequences by the set of polynomials

$$I^{(j)}(D) = i_o^{(j)} + i_1^{(j)}D + i_2^{(j)}D^2 + \dots \qquad j = 1, 2, \dots, k_o, \tag{26}$$

where $i_u^{(j)}$ is the information symbol that enters the j^{th} input line of the encoder at time u. Similarly, the n_o output sequences are denoted

$$T^{(j)}(D) = t_o^{(j)} + t_1^{(j)}D + t_2^{(j)}D^2 + \dots \qquad j = 1, 2, \dots, n_o, \tag{27}$$

where $t_u^{(j)}$ is the symbol that leaves the j^{th} output line at time u.

We assume, without loss of generality, that the code is in systematic form, that is, that the first k_o output sequences are identical to the input sequences. In the D-notation, this property can be expressed as

$$T^{(j)}(D) = I^{(j)}(D) \qquad j = 1, 2, \ldots, k_o. \tag{28}$$

The defining property of a convolutional code is that the remaining $n_o - k_o$ output sequences, or parity sequences, be linear combinations of the input sequences. That is,

$$T^{(j)}(D) = G^{(j)}(D)\, I^{(1)}(D) + H^{(j)}(D)\, I^{(2)}(D) + \ldots + Z^{(j)}(D)\, I^{(k_o)}(D)$$

$$j = k_o + 1, k_o + 2, \ldots, n_o. \tag{29}$$

Each transmitted parity digit is thus a linear combination of preceding information digits. The set of $k_o(n_o-k_o)$ polynomials

$$G^{(j)}(D),\ H^{(j)}(D),\ \ldots\ Z^{(j)}(D), \qquad j = k_o + 1, k_o + 2, \ldots, n_o$$

comprises the code-generating polynomials, and their choice specifies the code.

Let m be the largest of the degrees of the code-generating polynomials, that is, the largest power of D which multiplies any of the input sequences in Eq. 29. Then any particular information symbol can affect the output sequences over a span of, at most, m + 1 time units. During this time span, a total of $(m+1)n_o$ symbols leaves the encoder, and hence the code is said to have a constraint length, n_A, of $(m+1)n_o$ symbols.

In the D-notation, the code-generating polynomials will be represented as

$$G^{(j)}(D) = g_o^{(j)} + g_1^{(j)}D + \ldots + g_m^{(j)}D^m, \tag{30}$$

where the coefficients are again elements of GF(q).

We define an initial code word of a convolutional code to be the first set of n_A symbols output from the encoder. We shall use the notation

$$T_m^{(j)}(D) = t_o^{(j)} + t_1^{(j)}D + \ldots + t_m^{(j)}D^m \qquad j = 1, 2, \ldots, n_o \tag{31}$$

to represent the symbols in a first code word. Similarly, we shall use the notation

$$I_m^{(j)}(D) = i_o^{(j)} + i_1^{(j)}D + \ldots + i_m^{(j)}D^m \qquad j = 1, 2, \ldots, k_o \tag{32}$$

to represent the transmitted symbols over the same time span.

With this definition, the set of initial code words of a convolutional code forms a systematic linear code, or (n, k) code, as defined in section 1.1. In making the correspondence with the notation of that section, n_A is identified with n, and $Rn_A = (m+1)k_o$ is identified with k. In other words, there are n_A symbols in an initial code word and, of these symbols, Rn_A are information symbols.

These concepts are best clarified by giving an example. Consider the R = 1/2 binary convolutional code for which the code-generating polynomial is

$$G^{(2)}(D) = 1 + D + D^3. \tag{33}$$

For an arbitrary information sequence, $I^{(1)}(D)$, the encoded parity sequence, from Eq. 29, is given by

$$T^{(2)}(D) = (1+D+D^3)\, I^{(1)}(D) \tag{34}$$

from which we see that any transmitted parity digit is the sum of the information symbols occurring at the same time and at one and three time units earlier. Hence if the input sequence were

$$I^{(1)}(D) = 1 + D^2 + D^6 + \dots, \tag{35}$$

the output parity sequence is

$$T^{(2)}(D) = 1 + D + D^2 + D^5 + D^6 + \dots . \tag{36}$$

The manner in which the parity bits are formed can be seen from Fig. 4. As the information bits move through the four-stage shift register, the fixed connections to the adder insure that each formed parity bit is the sum of the current information bit and the information bits one and three time units earlier which are stored in the second and fourth stages of the shift register.

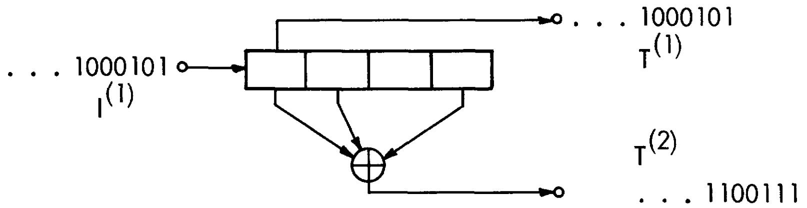


Fig. 4. Encoder for the example.

Since the code-generating polynomial has degree m = 3 in this example, the initial code word is

$$T_3^{(1)} = I_3^{(1)} = 1 + D^2,$$

and (37)

$$T_3^{(2)} = 1 + D + D^2.$$

From the preceding example, it should be clear that the initial code word defined by Eq. 31 can be obtained from (28) and (29) simply by dropping all terms in D with exponents greater than m. Hence, we can write

$$T_m^{(j)}(D) = I_m^{(j)}(D) \qquad j = 1, 2, \dots, k_o \tag{38}$$

and

$$T_m^{(j)}(D) = \left\{ G^{(j)}(D)\, I_m^{(1)}(D) + \dots + Z^{(j)}(D)\, I_m^{(k_o)}(D) \right\}$$

$$j = k_o + 1, k_o + 2, \dots, n_o. \tag{39}$$

Here and hereafter we use the brace notation to enclose polynomials that are to be expanded under the constraint that all terms in D with exponents greater than m are to be dropped from the resulting expressions. (The operation of Eq. 39 is exactly the same as specifying that the polynomial on the right be taken to be the minimum degree polynomial in the residue class containing the polynomial within braces modulo the ideal generated by $F(D) = D^{m+1}$. The terms in this statement are defined in Appendix A.)

2.2 ALGEBRAIC STRUCTURE

We shall now make a closer investigation of the set of initial code words of a convolutional code as given by Eqs. 38 and 39. Our purpose is to derive certain properties of these code words which can be used later as a basis for proving the Gilbert bound and the random-coding bound for the class of convolutional codes. We begin by proving a lemma which seems quite abstract at this point, but whose utility will later become apparent.

LEMMA 1: Given an arbitrary polynomial $I_m^{(j)}(D)$ of degree m or less such that $i_o^{(j)}$ is nonzero, for all q^{m+1} choices of G(D) as a polynomial of degree m or less, the polynomial

$$\left\{ G(D)\, I_m^{(j)}(D) \right\}$$

is distinct, and hence takes on once, and once only, the identity of each of the q^{m+1} polynomials of degree m or less.

PROOF: Suppose there exists some polynomial $G^*(D)$ of degree m or less which is such that

$$\left\{ G^*(D)\, I_m^{(j)}(D) \right\} = \left\{ G(D)\, I_m^{(j)}(D) \right\}. \tag{40}$$

Then, it follows that

$$\left\{ [G^*(D) - G(D)]\, I_m^{(j)}(D) \right\} = 0. \tag{41}$$

Equation 41 states that the polynomial on the left can have no terms of degree m or less. Since $i_o^{(j)}$, the zero-degree term of $I_m^{(j)}(D)$, is nonzero, there will be a term of degree m or less if $G^*(D) - G(D)$ has such a term. Since both $G^*(D)$ and G(D) are of degree m or less, it follows from (41) that $G^*(D) = G(D)$. Thus each choice of G(D) yields a distinct $\left\{ G(D) I_m^{(j)}(D) \right\}$ and this proves the lemma.

We have seen that a convolutional code is specified by the choice of $(n_o - k_o)n_o$ code-generating polynomials. Since a polynomial of degree m or less can be chosen in any of q^{m+1} ways, there are exactly $q^{(m+1)(n_o-k_o)k_o}$ distinct convolutional codes with rate

$R = k_o/n_o$ and constraint length $n_A = (m+1)n_o$. (We allow any generating polynomials of degree m or less, and do not require that any of the generating polynomials have a nonzero term in D^m.) The utility of Lemma 1 is that it permits one to determine in exactly how many convolutional codes a specified initial code word can appear, and this, in turn, can be used as a basis for proving that the class of convolutional codes meets the Gilbert and random-coding bounds.

It is convenient to use the following nomenclature: The set of k_o symbols input to the encoder at time zero, namely $i_o^{(1)}, i_o^{(2)}, \ldots, i_o^{(k_o)}$, will be called the set of first information symbols. Since, according to Eq. 28, $t_o^{(j)} = i_o^{(j)}$ for $j = 1, 2, \ldots, k_o$, the following theorem gives the number of distinct codes in which some initial code word, with a specified set of first information symbols, appears.

THEOREM 5: Given an initial code word $T_m^{(j)}(D)$, $j = 1, 2, \ldots, n_o$, for which at least one of the symbols $t_o^{(j)}$ $(j=1, 2, \ldots, k_o)$ is nonzero, this first code word appears in exactly $q^{(m+1)(n_o-k_o)(k_o-1)}$ distinct convolutional codes with rate $R = k_o/n_o$ and constraint length $n_A = (m+1)n_o$.

PROOF 5: The assumption that at least one of the $t_o^{(j)}$, $j = 1, 2, \ldots, k_o$, is nonzero implies that at least one of the set of first information symbols is nonzero. Without loss of generality, we can assume that $i_o^{(1)} \neq 0$.

From (38), it follows that prescribing an initial code word fixes the polynomials $I_m^{(j)}(D)$, $j = 1, 2, \ldots, k_o$. Consider the parity sequences that are given by (39). Equation 39 can be written in the form

$$T_m^{(j)}(D) - \left\{H^{(j)}(D)\, I_m^{(2)}(D) + \ldots + Z^{(j)}(D)\, I_m^{(k_o)}(D)\right\} = \left\{G^{(j)}(D)\, I^{(1)}(D)\right\} \tag{42}$$

for $j = k_o + 1, k_o + 2, \ldots, n_o$. Now for any fixed choice of the polynomials $H^{(j)}(D)$, $\ldots$, $Z^{(j)}(D)$, the left-hand side of this equation is a fixed polynomial. It then follows from Lemma 1 that this equation will be satisfied for one and only one choice of $G^{(j)}(D)$.

Since each of the $(k_o-1)(n_o-k_o)$ polynomials $H^{(j)}(D), \ldots, Z^{(j)}(D)$, $j = k_o + 1, k_o + 2, \ldots, n_o$, can be chosen in any of q^{m+1} ways as a polynomial of degree m or less, it follows that the specified code word appears in exactly $q^{(m+1)(k_o-1)(n_o-k_o)}$ distinct codes and this proves the theorem.

We shall now show how Theorem 5 can be used to prove the Gilbert bound for convolutional codes.

2.3 THE GILBERT BOUND

The minimum distance, d, of a convolutional code is defined to be the smallest number of symbols for which two initial code words differ that do not have identical sets of first information symbols. Since the set of initial code words forms a linear code and hence forms a group, it follows that the minimum distance is also equal to the minimum weight (that is, the number of nonzero symbols) of an initial code word that has at least one nonzero first information symbol. (Minimum distance d does not imply that there exists two infinitely long transmitted sequences, with different sets of first information symbols, that are distance d apart; but rather there are two such sequences whose first n_A symbols are distance d apart.) We now prove a lower bound on minimum distance for convolutional codes.

THEOREM 6: Given a rate $R = k_o/n_o$ and a constraint length $n_A = (m+1)n_o$, then there exists at least one convolutional code with minimum distance d, where d is the largest integer for which

$$\sum_{j=1}^{d-1} \binom{n_A}{j} (q-1)^j < q^{n_A(1-R)}. \tag{43}$$

PROOF 6: A convolutional code has minimum distance at least d if it has no initial code word of weight d - 1 or less for which some first information symbol is nonzero. Thus, if in the set of all codes of length n_A, we count the total number of initial code words with some nonzero first information symbol which have weight d - 1 or less, and if this total is less than the number of codes, then there must be at least one code with minimum distance d or greater.

Since there are n_A symbols in an initial code word, it follows that when

$$\begin{pmatrix}\text{number of nonzero}\\ n_A\text{-tuples with}\\ \text{weight } d-1 \text{ or less}\end{pmatrix}\begin{pmatrix}\text{maximum number of distinct}\\ \text{codes in which each initial}\\ \text{code word with at least one}\\ \text{nonzero first information}\\ \text{symbol can appear}\end{pmatrix} < \begin{pmatrix}\text{total}\\ \text{number}\\ \text{of}\\ \text{distinct}\\ \text{codes}\end{pmatrix} \tag{44}$$

then there must exist at least one code that has minimum distance d or greater. From the fact that the $(n_o-k_o)k_o$ polynomials that specify a distinct convolutional code can each be chosen in q^{m+1} ways as polynomials of degree m or less, it follows there are exactly $q^{(m+1)(n_o-k_o)k_o}$ convolutional codes. Thus with the aid of Theorem 5, (44) can be written

$$\left(\sum_{j=1}^{d-1} \binom{n_A}{j} (q-1)^j\right)\left(q^{(m+1)(n_o-k_o)(k_o-1)}\right) < q^{(n_o-k_o)k_o(m+1)} \tag{45}$$

or

$$\sum_{j=1}^{d-1} \binom{n_A}{j} (q-1)^j < q^{(m+1)(n_o-k_o)} = q^{n_A(1-R)} \tag{46}$$

and this is the result stated in the theorem.

The bound of Theorem 6 is equivalent to the Gilbert bound on minimum distance for block codes with the same rate and constraint length.[22]

COROLLARY: There exists at least one binary convolutional code with rate R and constraint length n_A with minimum distance at least d, where d is the largest integer for which

$$H(d/n_A) \leqq 1 - R, \tag{47}$$

where $H(x) = -x \log_2 x - (1-x) \log_2 (1-x)$ is the entropy function.

PROOF: For q = 2, (43) reduces to

$$\sum_{j=1}^{d-1} \binom{n_A}{j} < 2^{n_A(1-R)}. \tag{48}$$

The summation on the left can be overbounded[23] as

$$\sum_{j=1}^{d-1} \binom{n_A}{j} < 2^{n_A H(d/n_A)}. \tag{49}$$

Hence if

$$2^{n_A H(d/n_A)} < 2^{n_A(1-R)} \tag{50}$$

or

$$H(d/n_A) < 1 - R, \tag{51}$$

then (48) is satisfied, and there exists at least one convolutional code with minimum distance d or greater as was to be shown.

Inequality (47) is the usual assymptotic form of the Gilbert bound and will be used in Section III as a measure of quality for specific convolutional codes.

2.4 AN UPPER BOUND ON MINIMUM DISTANCE

In a manner similar to that used by Plotkin[24] for block codes, we can compute the average weight of an initial code word for which at least one of the first set of information symbols is nonzero, and then use the average weight to bound the minimum distance. We proceed as follows.

Since the set of initial code words of a convolutional code forms a linear code, and hence forms a group (cf. sec. 1.1), it follows that each element of GF(q) appears in a given code position in exactly 1/q of the code words. (We assume that for each j, at least one of the symbols $g_o^{(j)}$, $h_o^{(j)}$, ..., $z_o^{(j)}$ is nonzero. This is a necessary and sufficient condition that none of the positions in an initial code word will contain only "zeros" for all code words, that is, there is no "idle" position.) This fact can be verified as follows: Adding any code word that has the element $-\beta$ in the given position to all of the code words must reproduce the original set of code words in some order, because of the closure property of a group. There will then be as many code words with zero in the given position as there were originally with β in that position. Since β is arbitrary, it follows that all elements of GF(q) must appear in the given position the same number of times in the set of code words. Thus, since there are q^{Rn_A} code words in the set of initial code words and each has n_A positions, it follows that the total number of nonzero positions in the entire set of initial code words is $[(q-1)/q]n_A q^{Rn_A}$.

The set of all initial code words for which all of the first information symbols are zero form a subgroup because the sum of any two such words is another such code word. From Eq. 39, it follows that $t_o^{(j)} = 0$, $j = 1, 2, \ldots, n_o$, for these code words. By the same argument as that given above, each element of GF(q) will appear in any given one of the remaining $n_A - n_o$ code positions in exactly $1/q$ of this set of initial code words. Since this subgroup contains $q^{R(n_A-n_o)}$ members, it follows that there is a total of $[(q-1)/q(n_A-n_o)]\, q^{R(n_A-n_o)}$ nonzero entries in the set of all such initial code words.

The average weight of a code word for which at least one of the set of first information symbols is nonzero is given by

$$d_{avg} = \frac{\begin{pmatrix}\text{number of nonzero positions}\\ \text{in the set of all initial code}\\ \text{words}\end{pmatrix} - \begin{pmatrix}\text{number of nonzero positions}\\ \text{in the set of all code words}\\ \text{for which all first informa-}\\ \text{tion symbols are "zeros"}\end{pmatrix}}{(\text{number of initial code words}) - \begin{pmatrix}\text{number of initial code words}\\ \text{for which all first informa-}\\ \text{tion symbols are zero}\end{pmatrix}}.$$

Using the preceding results, we can write this equation

$$d_{avg} = \frac{\frac{q}{q-1}\, n_A q^{Rn_A} - \frac{q}{q-1}\,(n_A-n_o)\, q^{R(n_A-n_o)}}{q^{Rn_A} - q^{R(n_A-n_o)}}. \tag{52}$$

Equation 52 can be reduced to

$$d_{avg} = \frac{q-1}{q}\left(n_A + \frac{n_o}{q^{Rn_o} - 1}\right). \tag{53}$$

Finally, since the minimum weight code word with some nonzero first information symbol must have integer weight, which in turn must be no greater that the average weight, we have the following result.

THEOREM 7: The minimum distance, d, of a convolutional code satisfies the inequality

$$d \leqq \left\lfloor \frac{q-1}{q}\left(n_A + \frac{n_o}{q^{Rn_o} - 1}\right)\right\rfloor. \tag{54}$$

COROLLARY: The minimum distance, d, of a binary convolutional code with $R = 1/n_o$ satisfies

$$d \leqq \left\lfloor \frac{1}{2}(n_A+n_o) \right\rfloor . \tag{55}$$

The corollary is obtained by substituting $q = 2$ and $Rn_o = 1$ in (54). Inequality (55) is the form of the upper bound on minimum distance which will be useful in Section III.

Inequalities (54) and (55) do not give good upper bounds on minimum distance for arbitrary n_A because, for large n_A, these bounds are roughly independent of the rate R. We are interested in these bounds for the following reason. In section 3.6 we shall demonstrate the existence of a class of codes with prescribed values of n_A and n_o for which (55) is satisfied with the equality sign.

2.5 RANDOM-CODING BOUND

We turn our attention now to the task of computing what sort of error probabilities can be attained at the receiver when data are transmitted over a binary symmetric channel after convolutional encoding. Rather than handle the problem directly, we shall prove a more general random-coding bound that can be applied to the ensemble of convolutional codes, as well as to several other ensembles of linear codes.

Let us focus our attention on binary (n, k) codes. For any such code the set of 2^k possible information sequences forms a group, the group of all binary k-tuples. We define a group partition to be a mapping of this group of k-tuples into any subgroup and its proper cosets (cf. Appendix A). As an example, take the case $k = 2$. One possible group partition is then

$$H = [00, 01]$$

$$C_1 = [10, 11].$$

Here, we have taken the subgroup H to be the 2-tuples whose first bit is a zero, in which case there is only a single proper coset, C_1, which is the set of 2-tuples whose first bit is a one.

Next we observe that when the parity sequences are added to the information places, the resulting n-tuples may still be assigned to a subgroup of the group of code words and its proper cosets, with the information sequences being mapped in the same manner as was done in the original group partition. If we continue the example above by setting $n = 4$ and specifying that the first parity bit be equal to the first information bit and that the second parity bit be equal to the sum of the two information bits, then we obtain

$$H' = [0000, 0101]$$

$$C_1' = [1011, 1110].$$

The group partition of the information sequences is nothing more than the "standard array" of a subgroup and its cosets, which was first introduced into coding theory by

Slepian.[12] We use a different nomenclature for the following reason. In Slepian's standard array, the subgroup is the group of code words, and the parent group is the group of all n-tuples. When the parity sequences are attached to the information sequences in a group partition, the subgroup is the set of code words whose information sequences formed the subgroup in the original group partition, and the parent group is the group of all code words.

We are interested in group partitions for the following reasons. First, the set of distances between members of a proper coset is the same as the set of weights of the members of the subgroup. This follows trivially from the definition of a coset given in Appendix A. Second, the set of distances from any member of one coset to the members of a second coset is the same as the set of weights of the members in some fixed proper coset, and does not depend on the particular choice of the member from the first coset. Again, this follows from the fact that the members in any coset differ from each other by the members in the subgroup. For example, in the code used in the previous example, the members of a coset are distance 2 apart, but any member of C_1' is distance 3 from any member of H', since all code words in the only proper coset have weight 3.

In Appendix B, we prove the following lower bound on the error probability that can be achieved in deciding to which coset of a group partition the transmitted information sequence belongs when the code words are transmitted through a binary symmetric channel.

THEOREM 8: Given an ensemble of equiprobable binary (n, k) codes and a group partition of the information sequences with the property that an information sequence in any proper coset has equal probability of being assigned any of the 2^{n-k} possible parity sequences, after transmission through a binary symmetric channel at any rate $R = k/n$ less than the channel capacity C, the coset to which the transmitted information sequence belongs can be determined with an average error probability, $P(e)$, that satisfies

$$P(e) < K\, 2^{-na}, \tag{56}$$

where K and a are the coefficient and exponent, respectively, in the usual random-coding bound. (See Appendix B for precise values.)

The first error probability of a convolutional code, $P_1(e)$, is defined to be the average error probability in decoding the set of first information symbols $i_o^{(j)}$, $i = 1, 2, \ldots, k_o$, given the first set of n_A received bits. With this definition we have the following corollary as an immediate consequence of Theorem 8.

COROLLARY 1: The average error probability, $P_1(e)$, for the ensemble of binary convolutional codes satisfies (56), with n_A replacing n.

PROOF: We have seen (Section II) that the set of initial code words of a convolutional code form an (n, k) code, where $n = n_A$ and $k = (m+1)k_o$. Consider now the following group partition: Take as the subgroup the set of all information sequences for which the set of all first information symbols are zero, that is, $i_o^{(j)} = 0$, $j = 1, 2, \ldots, k_o$.

This subgroup has $2^{k_o} - 1$ proper cosets in which each proper coset contains all of the information sequences that have the same set of first information symbols, not all of which are zeros. Given this group partition, Theorem 5 implies that over the ensemble of all convolutional codes, any information sequence in a proper coset has equal probability of being assigned any possible parity sequence. Thus all the conditions of Theorem 8 are satisfied, and (56) applies to the ensemble of binary convolutional codes.

We have thus established the fact that for the ensemble of convolutional codes, there must be at least one code for which the set of first information symbols can be determined with an error probability that satisfies (56). It will be shown in section 4.1 that decoding for convolutional codes can be done sequentially by using the same decoding algorithm to determine the subsequent sets of information symbols as was used to determine the set of first information symbols. Thus if all previous sets have been correctly decoded, the next set of information symbols can also be determined with an error probability that satisfies (56).

Although this report is not primarily concerned with other ensembles of codes, we shall apply Theorem 8 to Wozencraft's ensemble of randomly shifted codes (cf. Section I) to illustrate its general usefulness in proving the random-coding bound for ensembles of linear codes.

COROLLARY 2: The block error probability, P(e), for the ensemble of randomly shifted codes satisfies (56).

PROOF: The randomly shifted codes can be described in the following manner. Consider any maximal-length shift register of N stages, where N is the maximum of k and $n - k$. If any nonzero initial conditions are placed in this shift register and it is then shifted some number of times, say M, a distinct nonzero N-tuple will remain in the shift register for each choice of $M = 1, 2, \ldots, 2^N - 1$. Coding for the randomly shifted codes is done as follows: The k information bits are placed in the k lowest order positions of the shift register which is then shifted some fixed number of times, M. The contents of the $n - k$ lowest order positions of the shift register are then attached as the parity sequence. Since the encoding circuit is linear, the code is a true (n, k) code. There are 2^N codes in the ensemble, one for each choice of $M = 0, 1, \ldots, 2^N - 1$; here we define the case $M = 0$ to be the case for which the shift register contains only zeros. Now consider the following group partition of the information sequences: The all-zero sequence is the subgroup and each proper coset contains a single distinct nonzero information sequence. Clearly any nonzero information sequence has equal probability of being assigned any parity sequence if all choices of M are equiprobable. Thus the conditions of Theorem 8 are satisfied, and hence the transmitted information sequence can be determined with an error probability that satisfies (56). (This ensemble of codes is interesting in that it contains the fewest codes of any ensemble to which the random-coding bound has yet been shown to apply.)

It is interesting that the same basic theorem can be used to prove the random-coding bound for the ensemble of convolutional codes and an ensemble of block codes. Theorem 8 can also be used to prove that the ensemble of sliding parity-check codes and the ensemble of all systematic codes satisfy the random-coding bound. The proofs for these last two ensembles is very similar to the proof of Corollary 2 and hence will not be given here.

2.6 ENCODING CIRCUITS

We stated in Section I that the first coding problem is the construction of good codes that are readily instrumented. The class of convolutional codes satisfy both the Gilbert and random-coding bounds, and hence can certainly be classified as a "good" class of codes. We shall show also that they satisfy the requirement of being readily instrumented.

One of the principal advantages for the polynomial approach to convolutional encoding, which we have adopted in this section, is that it leads naturally to the specification of encoding circuits. The encoding equations, (28) and (29), are already in convenient delay operator form.

a. Rn_A-Stage Encoder

A circuit for carrying out the operations of Eqs. 28 and 29 can be synthesized as shown in Fig. 5. In this circuit, each of the k_o information sequences, $I^{(j)}(D)$, $j = 1, 2, \ldots, k_o$, is fed into a separate shift-register chain of m stages, as well as being fed directly to one of the output terminals. The $n_o - k_o$ parity sequences, $T^{(j)}(D)$,

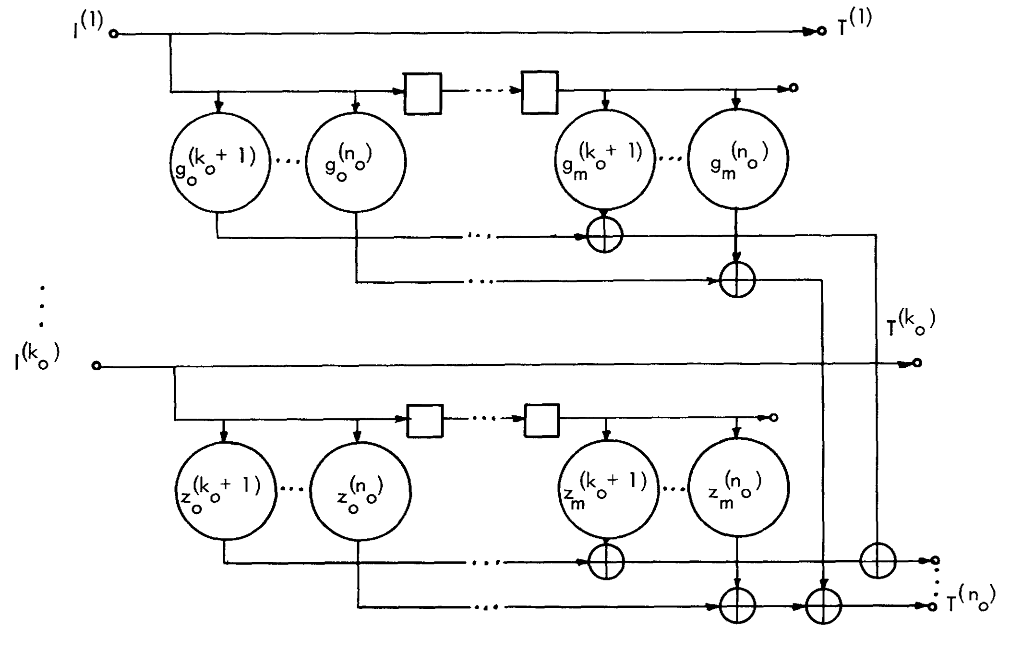


Fig. 5. Rn_A-Stage encoder.

$j = k_o + 1, k_o + 2, \ldots, n_o$, are formed by the adders outside the feedback chain. Since there are m stages in each of the k_o shift-register chains, there is a total of $mk_o = R(n_A - n_o)$ stages, or approximately Rn_A stages, of shift register in this encoder. We shall refer to this type of encoder as an Rn_A-stage encoder.

The output of the $n_A R$-stage encoder can be verified to conform to Eqs. 28 and 29 in the following manner: Assume that $i_o^{(1)} = 1$ and that all other input symbols are zeros. Then the output on line j is readily checked to be $G^{(j)}(D)$ for $j = k_o + 1, k_o + 2, \ldots, n_o$ and these are the polynomials that multiply the first information sequence in (29). Similarly, a single "one" input on any other input line gives as outputs the polynomials associated with that input sequence in (29). The linearity of the circuit then guarantees that the output will be correct for arbitrary input sequences.

b. $(1-R)n_A$-Stage Encoder

A second circuit that performs the operations of Eqs. 28 and 29 is shown in Fig. 6. This circuit has a shift-register chain of m stages for each of the $n_o - k_o$ parity sequences, and thus contains a total of $m(n_o - k_o) = (1-R)(n_A - n_o)$, or approximately $(1-R)n_A$, stages of shift register. We shall refer to this circuit as the $(1-R)n_A$-stage encoder. The adders in this circuit are placed between stages of the shift-register chains, and hence no adder has more than $k_o + 1$ inputs. This can be an important

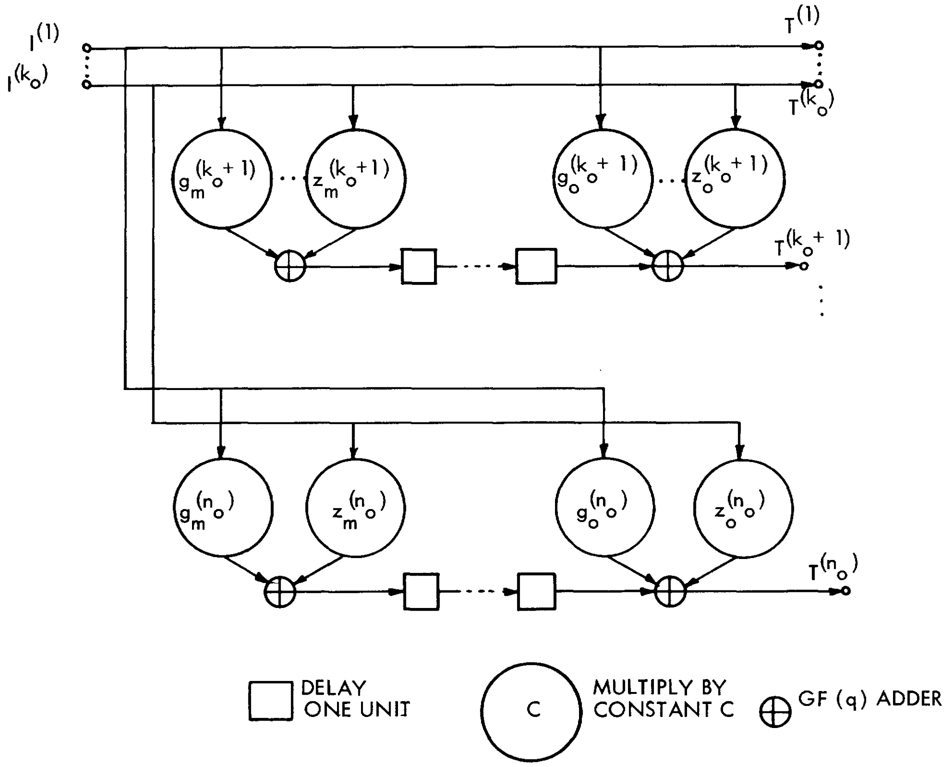


Fig. 6. $(1-R)n_A$-Stage encoder.

feature in high-speed digital circuits in which delay in the logical elements becomes important.

The output of the $(1-R)n_A$-stage encoder can be verified to conform to Eqs. 28 and 29 in exactly the same manner as was described for the Rn_A-stage encoder.

c. Serializing of Symbols

The output symbols of both the Rn_A-stage and the $(1-R)n_A$-stage encoders occur in sets of n_o symbols each time unit, one on each of the n_o output lines. When it is desired to transmit these symbols over a single communication channel, they must be serialized to occur at the rate of one symbol every $1/n_o$ time units. This symbol interlacing can be accomplished by a commutating circuit such as the one shown in Fig. 7, which consists principally of a sampling circuit and a shift register of $n_o - 1$ stages.

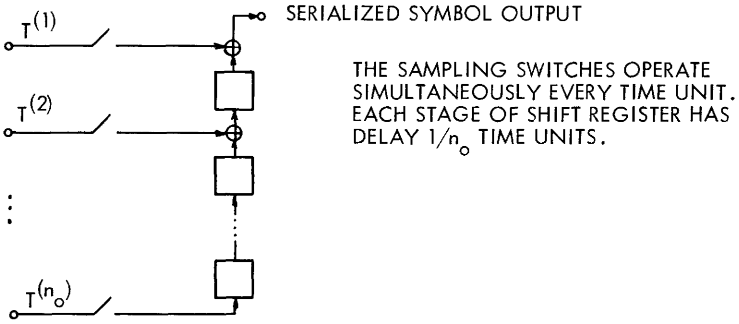

$T^{(1)}$
$T^{(2)}$
$T^{(n_o)}$
SERIALIZED SYMBOL OUTPUT
THE SAMPLING SWITCHES OPERATE SIMULTANEOUSLY EVERY TIME UNIT. EACH STAGE OF SHIFT REGISTER HAS DELAY $1/n_o$ TIME UNITS.

Fig. 7. Commutating circuit.

d. Remarks on Encoding Circuits

The Rn_A-stage encoder described above is essentially the same as the canonic-form encoder developed by Wozencraft and Reiffen.[20] The $(1-R)n_A$-stage encoder for convolutional codes is not believed to have been described previously.

Since Rn_A is the number of information symbols within the constraint length of the convolutional code and $(1-R)n_A$ is the number of parity symbols within the constraint length, a convolutional code can be encoded by a linear sequential network containing a number of storage devices (that is, stages of shift register) equal to either, and hence to the minimum, of these two quantities. Peterson[25] has proved this same result for block cyclic codes. Interest in this parallelism stems from the fact that, as shown in sections 2.3 and 2.5, convolutional codes satisfy both the Gilbert and random-coding bounds, whereas little is known about the behavior of cyclic codes of arbitrary length.

2.7 PARITY CHECKING

In section 1.1a we defined the parity checks of a linear code to be

$$s_j = \sum_{i=1}^{k} c_{ji} r_i - r_j \qquad i = k+1, k+2, \ldots, n$$

and we saw that the parity checks furnished a set of n - k equations in the n noise variables $e_1, e_2, \ldots, e_n$. We call the process of forming the parity checks, given by Eq. 5, parity checking.

Before illustrating the manner in which parity checking can be performed for a convolutional code, we shall first prove a more general statement.

THEOREM 9: The parity checks for an (n, k) code may be formed by subtracting the received parity symbols, r_j, $j = k+1, k+2, \ldots, n$, from the parity symbols obtained by encoding the received information symbols, r_j, $j = 1, 2, \ldots, k$.

PROOF 9: According to Eq. 2, encoding of the received information symbols will give the set of parity symbols, t_j', where

$$t_j' = \sum_{i=1}^{k} c_{ji} r_i \qquad j = k+1, k+2, \ldots, n. \tag{57}$$

Subtraction of the received parity symbols then yields

$$t_j' - r_j = \sum_{i=1}^{k} c_{ji} r_i - r_j \qquad j = k+1, k+2, \ldots, n \tag{58}$$

and this coincides with the definition of the parity checks given by Eq. 4, which proves the theorem.

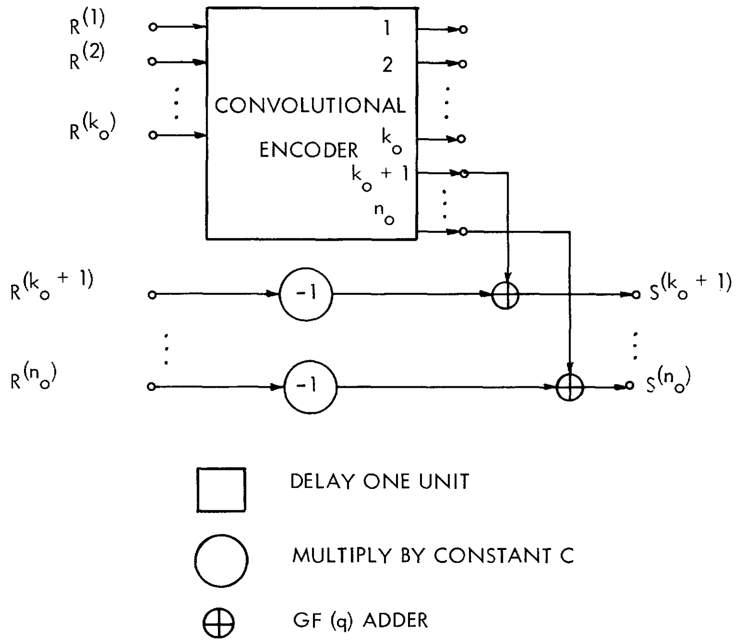


Fig. 8. Parity checker for convolutional codes.

We have already seen that the set of initial code words of a convolutional code forms an (n, k) code and hence Theorem 9 may be applied to such a code. It follows from this theorem that the circuit of Fig. 8 can be used to compute the parity checks of a convolutional code. The encoder in this circuit may be any convolutional encoder such as the $n_A R$-stage or the $n_A(1-R)$-stage encoders which have been described.

The received sequences, $R^{(j)}(D)$, $j = 1, 2, \ldots, n_o$, may be written in the D-notation as

$$R^{(j)}(D) = r_o^{(j)} + r_1^{(j)}D + r_2^{(j)}D^2 + \ldots \qquad j = 1, 2, \ldots, n_o, \tag{59}$$

where $r_j^{(j)}$ is the symbol received on the j^{th} input line at time u. With this notation, the noise sequences, $E^{(j)}(D)$, $j = 1, 2, \ldots, n_o$, are given by

$$E^{(j)}(D) = e_o^{(j)} + e_1^{(j)}D + e_2^{(j)}D^2 + \ldots \qquad j = 1, 2, \ldots, n_o, \tag{60}$$

where $e_u^{(j)}$ is the noise in the symbol received at time u in the j^{th} received sequence, that is,

$$R^{(j)}(D) = T^{(j)}(D) + E^{(j)}(D) \qquad j = 1, 2, \ldots, n_o. \tag{61}$$

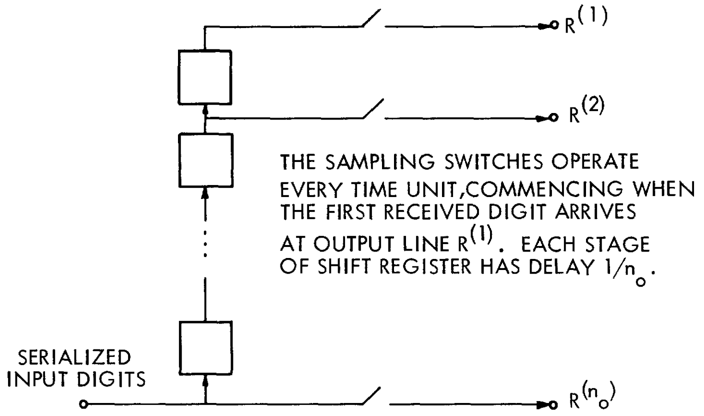
Then it follows from Theorem 9 that the parity-check sequences, $S^{(j)}(D)$, $j = k_o + 1$, $k_o + 2, \ldots, n_o$, are given by

$$S^{(j)}(D) = \left[G^{(j)}(D)R^{(1)}(D) + \ldots + Z^{(j)}(D)R^{(k_o)}(D)\right] - R^{(j)}(D)$$
$$j = k_o + 1, k_o + 2, \ldots, n_o. \tag{62}$$

Upon substitution of Eq. 29, Eq. 62 reduces to

$$S^{(j)}(D) = \left[G^{(j)}(D)E^{(1)}(D) + \ldots + Z^{(j)}(D)E^{(k_o)}(D)\right] - E^{(j)}(D)$$
$$j = k_o + 1, k_o + 2, \ldots, n_o. \tag{63}$$

In the parity-checking circuit of Fig. 8, it is assumed that the received symbols enter in sets of n_o symbols each time unit. When the transmitted symbols have been serialized for transmission over a single channel (cf. section 2.6c), it is necessary to restore the original parallel timing by a de-commutating circuit such as the one shown in Fig. 9. This circuit operates by storing the received symbols until n_o such symbols are received; at this time, the stored symbols are sampled to form the n_o synchronous received sequences. As was the case for the commutating circuit of Fig. 7, this circuit for restoring the original timing consists principally of a sampling



$R^{(1)}$
$R^{(2)}$
THE SAMPLING SWITCHES OPERATE EVERY TIME UNIT,COMMENCING WHEN THE FIRST RECEIVED DIGIT ARRIVES AT OUTPUT LINE $R^{(1)}$. EACH STAGE OF SHIFT REGISTER HAS DELAY $1/n_o$.
SERIALIZED INPUT DIGITS
$R^{(n_o)}$

Fig. 9. De-commutating circuit for serialized symbols.

circuit and $n_o - 1$ stages of shift register. It is of interest to note that the shift registers in the encoder and decoder are shifted once per time unit, only the shift registers in the commutating and de-commutating circuits are required to shift at the channel-symbol rate of once per $1/n_o$ time units.

III. CONVOLUTIONAL CODES FOR THRESHOLD DECODING

3.1 INTRODUCTION

The concept of threshold decoding which was formulated in Section I will now be applied to convolutional codes and this will be continued in Sections IV and V. We shall construct codes for which the threshold decoding algorithms can be efficiently applied and present decoding circuits for these codes and data on code performance over several communication channels.

3.2 DECODING PROBLEM FOR CONVOLUTIONAL CODES

We shall present those features of the decoding problem for convolutional codes that are necessary for understanding the principles of code construction for threshold decoding.

In section 2.2, it was shown that the initial code words of a convolutional code (i.e., all possible sets of the n_A symbols transmitted from time 0 through time m) form a linear or (n, k) code. Thus the remarks of section 1.2 concerning the decoding problem for linear codes apply to convolutional codes. In addition, there is an important special feature of convolutional decoding, namely the fact that only the set of first information symbols ($i_o^{(j)}$, $j = 1, 2, \ldots, k_o$) need be determined by the decoding algorithm from the n_A symbols in the received initial code word. One could, of course, use fewer, or more, than n_A received symbols in decoding the set of first information symbols. If fewer are used, a shorter code would suffice. If more are used, the error probability, $P_1(e)$, in decoding the set of first information symbols can be reduced; but, as pointed out by R. Gallager (private communication), the reduced error probability can be obtained more simply by using a code with greater n_A to begin with.

Let us assume that we have an algorithm that permits us to decode the set of first information symbols from the n_A symbols in the received initial code word. We shall denote the decoded estimates of the set of first information symbols as $i_o^{(j)*}$ to emphasize that these quantities may differ from the true values. Now consider the altered received sequences $R^{(j)*}(D)$ given by

$$R^{(j)*}(D) = R^{(j)}(D) - i_o^{(j)*} \qquad j = 1, 2, \ldots, k_o \tag{64}$$

and

$$R^{(j)*}(D) = R^{(j)}(D) - \left[i_o^{(1)*}G^{(j)}(D) + \ldots + i_o^{(k_o)}Z^{(j)}(D) \right] \qquad j = k_o + 1, k_o + 2, \ldots, n_o. \tag{65}$$

From Eqs. 28 and 29, it follows that if the decoding were correct, that is, if $i_o^{(j)*} = i_o^{(j)}$ for $j = 1, 2, \ldots, k_o$, then the effect of the set of first information symbols has been removed from the transmitted sequences and hence from the received sequences. Thus the decoding of $i_1^{(j)}$, for $j = 1, 2, \ldots, k_o$, can be performed by using the same algorithm on the n_A symbols from time 1 through time $m+1$ of the altered received sequences, $R^{(j)*}(D)$, as was used to decode $i_o^{(j)}$, for $j = 1, 2, \ldots, k_o$, from the n_A symbols from time 0 through time m of the original received sequences, $R^{(j)}(D)$. By continuing this procedure, the decoding can proceed sequentially with a new set of k_o information symbols being decoded each time unit.

An obvious difficulty arises when a decoding error is made. For, suppose that the set of first information symbols is incorrectly decoded, then the operation of Eq. 65 introduces a spurious pattern of symbols into the received parity sequences, namely

$$\left[i_o^{(1)}-i_o^{(1)*}\right]G^{(j)}(D) + \ldots + \left[i_o^{(k_o)}-i_o^{(k_o)*}\right]Z^{(j)}(D) \qquad j = k_o+1, k_o+2, \ldots, n_o. \tag{66}$$

These spurious symbols affect the decoding algorithm in much the same manner as a burst of channel noise of length n_A, and hence it can be expected that successive sets of information symbols will have a high probability of being incorrectly decoded. This is the familiar "propagation effect" of errors made in decoding convolutional codes. The two basic methods by which this effect can be circumvented, namely "resynchronization" and "error-counting," will now be explained.

In the "resynchronization" method, arbitrary information symbols are encoded for only a fixed number, N, of time units. "Zeros" are then encoded for the next m time units. Since no digits more than m time units in the past are used in the decoding process, the decoding can be carried out independently on the symbols received in the corresponding $N+m$ time units, after which the decoder is cleared and the decoding process is started anew on the next received symbols. Thus any propagation of error is confined within $N+m$ time units or $(N+m)n_o$ received symbols. The resynchronization method reduces the information rate to R' given by

$$R' = \frac{N}{N+m} R, \tag{67}$$

where $R = k_o/n_o$ is the nominal rate of the convolutional code. If $N \gg m$, the rate decrease is not substantial.

The "error-counting" method is a more refined manner of combating propagation error. This method was proposed by Wozencraft and Reiffen (in a slightly different form than that given here) to turn error propagation to useful advantage in a two-way communication system.[26] Briefly, the method operates as follows: When the decoding algorithm is "correcting more errors" over some span of received bits than the code itself can reliably correct, it is assumed that a decoding error has been made, or that the channel has temporarily become too noisy for reliable operation. In this event, the receiver asks that the data be re-transmitted from some point before that at which the high incidence of errors began. This method will be considered in more detail in section 4.2c.

The main point of the preceding discussion is that the threshold-decoding algorithm need be tailored only to give the decoded estimates of $e_o^{(j)}$, $j = 1, 2, \dots, k_o$, the error in the set of first information symbols. The remainder of the decoding process, namely the altering of the received sequences to remove the effect of the first symbols and the controlling of the error propagation effect, can be handled in routine fashion. Thus the problem of convolutional code construction for threshold decoding reduces to finding codes for which the mapping from the ordinary parity checks to the sets of parity checks orthogonal on $e_o^{(j)}$, $j = 1, 2, \dots, k_o$, can be carried out in an efficient manner, since these are the only noise digits that must be determined by the decoding process.

3.3 PARITY-CHECK STRUCTURE

From the foregoing remarks, it is clear that a careful investigation of the parity-check structure of convolutional codes will be needed. In Eq. 58, it was seen that the parity-check sequences are given by

$$S^{(j)}(D) = \left[G^{(j)}(D)E^{(1)}(D)+\dots+Z^{(j)}(D)E^{(k_o)}(D)\right]-E^{(j)}(D)$$

$$j = k_o + 1, k_o + 2, \dots, n_o. \tag{68}$$

In our usual manner, we represent these sequences by the notation

$$S^{(j)}(D) = s_o^{(j)} + s_1^{(j)}D + s_2^{(j)}D^2 + \dots , \tag{69}$$

where $s_u^{(j)}$ is then the parity check in sequence $S^{(j)}(D)$ that is formed at time u by the circuit of Fig. 8. From Eq. 68, we have

$$s_u^{(j)} = \left[g_u^{(j)}e_o^{(1)}+g_{u-1}^{(j)}e_1^{(1)}+\dots+g_o^{(j)}e_u^{(1)}\right]+\dots$$

$$+\left[z_u^{(j)}e_o^{(k_o)}+z_{u-1}^{(j)}e_1^{(k_o)}+\dots+z_o^{(j)}e_u^{(k_o)}\right]-e_u^{(j)}. \tag{70}$$

Since $g_u^{(j)} = 0$ for $u > m$, no parity check will check noise digits more than m time units in the past.

In matrix form, Eq. 70 may be written for the parity checks from time 0 through time m as

$$\begin{bmatrix} s_o^{(k_o + 1)} \\ s_1^{(k_o + 1)} \\ \vdots \\ s_m^{(k_o + 1)} \\ \cdot \\ \cdot \\ \cdot \\ s_o^{(n_o)} \\ s_1^{(n_o)} \\ \vdots \\ s_m^{(n_o)} \end{bmatrix} = [H_\Delta : -I_{m(n_o - k_o)}] \begin{bmatrix} e_o^{(1)} \\ e_1^{(1)} \\ \vdots \\ e_m^{(1)} \\ \cdot \\ \cdot \\ \cdot \\ e_o^{(n_o)} \\ e_1^{(n_o)} \\ \vdots \\ e_o^{(n_o)} \end{bmatrix} \tag{71}$$

where $I_{m(n_o - k_o)}$ is the unit matrix of dimension $m(n_o - k_o)$, and H_Δ is the matrix

$$H_\Delta = \begin{bmatrix} g_o^{(k_o+1)} & & & & & z_o^{(k_o+1)} & & & \\ g_1^{(k_o+1)} & g_o^{(k_o+1)} & 0 & & \cdots & z_1^{(k_o+1)} & z_o^{(k_o+1)} & 0 & \\ \vdots & & \ddots & & & \vdots & & \ddots & \\ g_m^{(k_o+1)} & g_{m-1}^{(k_o+1)} & \cdots & g_o^{(k_o+1)} & & z_m^{(k_o+1)} & z_{m-1}^{(k_o+1)} & \cdots & z_o^{(k_o+1)} \\ \vdots & & & & & \vdots & & & \\ g_o^{(n_o)} & & & & & z_o^{(n_o)} & & & \\ g_1^{(n_o)} & g_o^{(n_o)} & 0 & & \cdots & z_1^{(n_o)} & z_o^{(n_o)} & 0 & \\ \vdots & & \ddots & & & \vdots & & \ddots & \\ g_m^{(n_o)} & g_{m-1}^{(n_o)} & \cdots & g_o^{(n_o)} & & z_m^{(n_o)} & z_{m-1}^{(n_o)} & \cdots & z_o^{(n_o)} \end{bmatrix} \tag{72}$$

The nonzero entries of H_Δ are seen to lie entirely within the set of $(n_o - k_o)k_o$ triangular submatrices to which we shall refer as the parity triangles. There are $n_o - k_o$ rows of parity triangles in H_Δ, and k_o columns of parity triangles.

The structure of the parity triangles will be of considerable utility in constructing codes for threshold decoding. Each parity triangle has the following general form

$$\begin{array}{llllll}
g_o \\
g_1 & g_o \\
g_2 & g_1 & g_o \\
g_3 & g_2 & g_1 & g_o \\
\cdot & & & \ddots \\
\cdot \\
\cdot \\
g_m & g_{m-1} & \cdots & & g_1 & g_o
\end{array}$$

In other words, each parity triangle has a structure that is determined by one of the code-generating polynomials. The entire parity triangle is determined uniquely by the first column, or the last row.

An example at this point will be helpful in clarifying the parity-check structure. Consider a binary convolutional code with rate $R = k_o/n_o = 2/3$ and $n_A = (m+1)n_o = 9$. We choose the code-generating polynomials to be $G^{(3)}(D) = 1 + D$ and $H^{(3)}(D) = 1 + D^2$. Then Eq. 71 becomes ($-I = +I$ in the binary number field, since $1 + 1 = 0$ implies $1 = -1$)

$$\begin{bmatrix} s_o^{(3)} \\ s_1^{(3)} \\ s_2^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & & & 1 & & & 1 & & \\ 1 & 1 & & 0 & 1 & & & 1 & \\ 0 & 1 & 1 & 1 & 0 & 1 & & & 1 \end{bmatrix} \begin{bmatrix} e_o^{(1)} \\ e_1^{(1)} \\ e_2^{(1)} \\ e_o^{(2)} \\ e_1^{(2)} \\ e_2^{(2)} \\ e_o^{(3)} \\ e_1^{(3)} \\ e_2^{(3)} \end{bmatrix} \qquad (73)$$

which is the matrix representation of the following equations:

$$\begin{aligned} s_o^{(3)} &= e_o^{(1)} + e_o^{(2)} + e_o^{(3)} \\ s_1^{(3)} &= e_o^{(1)} + e_1^{(1)} + e_1^{(2)} + e_1^{(3)} \\ s_2^{(3)} &= e_1^{(1)} + e_2^{(1)} + e_o^{(2)} + e_2^{(2)} + e_2^{(3)} . \end{aligned} \tag{74}$$

From (74) and (73), it is easily seen, by comparison, that the manner in which the noise bits enter into the parity checks can be read directly from the matrix $[H_\Delta : -I]$. Each row of $[H_\Delta : -I]$ gives the coefficients of the noise bits in one parity check.

It should be noticed that $s_o^{(3)}$ and $s_1^{(3)}$ form a set of two parity checks orthogonal on $e_o^{(1)}$, since both check $e_o^{(1)}$, but no other noise bit is checked by both. Similarly, $s_o^{(3)}$ and $s_2^{(3)}$ can be seen to form a set of two parity checks orthogonal on $e_o^{(2)}$. It now follows from Theorem 1 that the noise bits in the first set of information symbols, namely $e_o^{(1)}$ and $e_o^{(2)}$, can be correctly determined by the majority decoding algorithm provided that there are one or fewer errors in the first $n_A = 9$ received symbols. Thus, as we have explained, this same majority decoding algorithm can be used to correct all errors in the received sequences, provided that the 9 symbols received in any 3 consecutive time units never contain more than a single error.

We shall now show how the concepts in this example can be generalized to yield an interesting and useful set of codes. In the rest of this section, we shall consider binary codes only.

3.4 BOUNDS ON CODES

It is convenient to introduce the following definitions at this point. Let $\{A_i\}$ be a set of J parity checks orthogonal on $e_o^{(1)}$ for a binary convolutional code with rate $R = 1/n_o$.

The number of noise bits, exclusive of $e_o^{(1)}$, which are checked by some A_i, will be called the size of A_i and will be denoted n_i.

The total number of distinct noise bits checked by the $\{A_i\}$ will be called the effective constraint length of the code and will be denoted n_E.

From this definition, it follows immediately that

$$n_E = 1 + \sum_{i=1}^{J} n_i \tag{75}$$

and, of course,

$$n_E \leqq n_A , \tag{76}$$

since n_A is the total number of distinct bits checked by the parity checks, $s_u^{(j)}$, which are combined to form the $\{A_i\}$.

THEOREM 10: For any rate $R = 1/n_o$ and any integer J, there exists a convolutional code for which a set $\{A_i\}$ of J parity checks orthogonal on $e_o^{(1)}$ can be formed in such a manner that the effective constraint length satisfies

$$n_E \leq \frac{1}{2}\frac{R}{1-R}J^2 + \frac{1}{2}J + 1 + \frac{1}{2}r\left[1 - r\frac{R}{1-R}\right], \tag{77}$$

where $r \equiv J \bmod(n_o-1)$.

The proof of Theorem 10 is given in Appendix C. This theorem is proved by showing that it is possible to construct a convolutional code so that there are J parity checks, $s_u^{(j)}$, which check $e_o^{(1)}$ and are such that these parity checks themselves form a set of parity checks orthogonal on $e_o^{(1)}$. In this construction, the sizes, n_i, of these parity checks are as stated in the following corollary:

COROLLARY: For any rate $R = 1/n_o$ and any integer J, there exists a binary convolutional code for which J parity checks orthogonal on $e_o^{(1)}$ can be formed so that $n_o - 1$ of these parity checks have size j for each j from 1 through Q, and are such that r have size Q+1, with

$$J = Q(n_o-1) + r, \quad 0 \leqq r < n_o - 1. \tag{78}$$

For rates $R = k_o/n_o$, where $k_o \neq 1$, Theorem 10 can be generalized in the following manner: The effective constraint length, n_E, is defined as the maximum over j of the number of distinct noise bits that appear in the set of J parity checks orthogonal on $e_o^{(j)}$ for j = 1, 2, ..., k_o. With this definition we have the following theorem, the proof of which is also given in Appendix C.

THEOREM 11: For any rate $R = k_o/n_o$ and any integer J, there exists a convolutional code for which J parity checks orthogonal on $e_o^{(j)}$ for j = 1, 2, ..., k_o can be formed so that the effective constraint length, n_E, satisfies

$$n_E \leq \frac{1}{2}\frac{R}{1-R}J^2 + \frac{1}{2}k_oJ + 1 + \frac{1}{2}r\left[k_o - r\frac{R}{1-R}\right], \tag{79}$$

where $r \equiv J \bmod(n_o-k_o)$.

For $k_o = 1$, Theorem 11 reduces to Theorem 10 as expected.

An estimate of the quality of the codes which satisfy (79) or (77) can be obtained as follows. The Gilbert bound, Eq. 47, states that the ratio, d/n_A, of minimum distance to constraint length can be made to satisfy $H(d/n_A) = 1 - R$. Minimum distance d implies that the first set of information symbols can always be correctly decoded whenever $\lfloor (d-1)/2 \rfloor$ or fewer errors occur among the n_A symbols within the constraint length. On the other hand, we have seen that $e_o^{(1)}$ can be correctly decoded by majority decoding whenever $\lfloor J/2 \rfloor$ or fewer errors occur among the n_E symbols within the effective constraint length. Thus the ratio $J/2n_E$ ought to be at least as great as the ratio $(d-1)/2n_A$ as guaranteed by the Gilbert bound if threshold decoding is to be efficient.

In Fig. 10, we compare the ratio $J/2n_E$ guaranteed by Theorems 10 and 11 to the ratio $(d-1)/2n_A$ guaranteed by the asymptotic Gilbert bound of Eq. 47. (Comparison

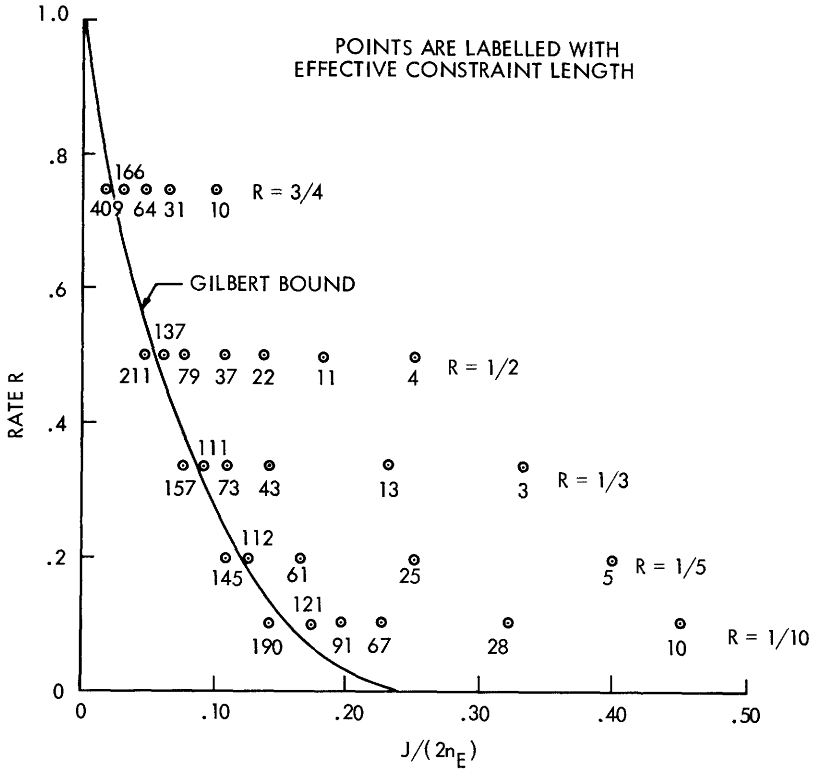


Fig. 10. Ratio of number of orthogonal parity checks, J, to twice the effective constraint length, n_E, as given by Theorems 10 and 12 and by comparison with the asymptotic Gilbert bound.

with the nonasymptotic bound of (43) is the proper choice but is difficult to present graphically. However, the comparison is hardly affected since for $n_A \approx 100$, $d/2n_A$ is nearly the same for the asymptotic and nonasymptotic bounds. For example, at $n_A = 100$ and $R = 1/2$, the former gives .055 and the latter gives .060.) For a wide range of rates, the ratio $J/2n_E$ does not fall below the Gilbert bound until the effective constraint length becomes on the order of 100 to 150 bits. It is for codes up to these effective constraint lengths that threshold decoding can be expected to be efficient. Clearly, for codes that satisfy Theorem 10 with the equality sign, the J/n_E ratio must eventually become poor, since J increases only as the square root of n_E, whereas the Gilbert bound shows that it is possible to make d grow linearly with n_A.

We have not been successful in finding codes for which the ratio $J/2n_E$ is substantially smaller than the bounds implied by Theorems 10 and 11 for rates 1/10 and greater. On the other hand, we have been unable to prove that it is impossible to find such codes except for the particular case $R = 1/2$. In this case we have been able to show that the bound of Theorem 10 is also a lower bound on n_E.

THEOREM 12: For rate $R = 1/2$ and any integer J, it is possible to find a convolutional code which is such that J parity checks orthogonal on $e_o^{(1)}$ can be constructed and for which

$$n_E = \frac{1}{2}J^2 + \frac{1}{2}J + 1 . \qquad (80)$$

Conversely, it is impossible to find such a code for which n_E is smaller than this number.

PROOF 12: The proof that it is possible to make n_E at least as small as the number in Eq. 80 follows by substituting R = 1/2 in (77). The proof that it is impossible to do better than this bound proceeds as follows.

The first step in the proof is to place a lower bound on the number of "ones" that must be contained in certain sums of columns of H_Δ. In the same manner that Eq. 71 was derived from Eq. 68, it can be readily shown that for an R = 1/2 code, the parity sequence $T_m^{(2)}(D)$ is given in terms of the information sequence $I_m^{(1)}(D)$ by the matrix product

$$\begin{bmatrix} t_o^{(2)} \\ t_1^{(2)} \\ \vdots \\ t_m^{(2)} \end{bmatrix} = [H_\Delta] \begin{bmatrix} i_o^{(1)} \\ i_1^{(1)} \\ \vdots \\ i_m^{(1)} \end{bmatrix} \qquad (81)$$

The parity sequence as a vector is then the sum of the columns of H_Δ corresponding to the information symbols that are "ones." Suppose there is some sum of C columns of H_Δ, including the first column which contains N "ones." Equation 81 then implies that there is an initial code word, for which $i_o^{(1)} = 1$, which has C nonzero information symbols and N nonzero parity symbols. The weight of this initial code word is then C + N and must be at least d, the minimum distance of the code. Hence we must have $N \geqq d - C$, that is, any sum of C columns of H_Δ, including the first column, must have at least d - C ones.

For R = 1/2, the matrix H_Δ consists of a single parity triangle, namely

$$\begin{array}{lllll} g_o^{(2)} & & & & \\ g_1^{(2)} & g_o^{(2)} & & & \\ g_2^{(2)} & g_1^{(2)} & g_o^{(2)} & & \\ g_3^{(2)} & g_2^{(2)} & g_1^{(2)} & g_o^{(2)} & \\ \vdots & & & & \\ g_m^{(2)} & \cdots & g_3^{(2)} & g_2^{(2)} & g_1^{(2)} \quad g_o^{(2)} \end{array} \qquad (82)$$

The second step in the proof is to notice the row-column equivalence of H_Δ. Specifically, we observe that if only the first four rows of H_Δ are retained, then the sum of the second and fourth rows contains exactly the same number of "ones" as the sum of the first and third columns since the same terms are added together. In general, there is a one-to-one correspondence between sums of rows that include the last row and sums of columns that include the first column. Thus we can conclude, from the previous paragraph, that any sum of C rows of H_Δ, including the last row, must have at least d - C ones.

Assume now that we have an R = 1/2 code for which a set A_1, A_2, ..., A_J of parity checks orthogonal on $e_o^{(1)}$ has been formed. The coefficients in each A_i correspond, as we have seen, to the sum of selected rows of the matrix $[H_\Delta:I]$. We assume that the parity checks have been ordered so that if $i < j$, then the lowermost row added in forming A_j is beneath the lowermost row added to form A_i. Now consider the R = 1/2 convolutional code for which the last row in its parity triangle is the lowermost row added to form A_i. The minimum distance, d_i, of this code must be at least i + 1 according to the corollary of Theorem 1.1 since it is possible to construct i parity checks orthogonal on $e_o^{(1)}$, namely A_1, A_2, ..., A_i. Finally, suppose that C_i rows of $[H_\Delta:I]$ were added to form A_i. Then the size, n_i, of this parity check must satisfy

$$n_i \geqq [(d_i - C_i) - 1] + C_i \tag{83}$$

as we can see in the following way. The number of information noise bits checked by A_i exclusive of $e_o^{(1)}$, is one less than the number of "ones" in the sum of the C_i rows of H_Δ. Since this sum includes the last row, it must have at least $d_i - C_i$ "ones." The last term on the right of (83) is accounted for by the fact that the sum of C_i rows of the identity matrix, I, always contains C_i "ones," and this is the number of parity noise bits checked by A_i.

Since d_i is at least i + 1, (83) becomes simply

$$n_i \geqq i . \tag{84}$$

Substituting (84) in (75), we obtain

$$n_E \geqq \sum_{i=1}^{J} i + 1 = \frac{1}{2}J^2 + \frac{1}{2}J + 1 \tag{85}$$

and this is the statement of the theorem.

The fact that, for R = 1/2, Theorem 10 gives the smallest possible value of n_E tempts one to conjecture that the same result might hold for other rates. This conjecture is not valid. In the following sections, many codes will be constructed for which n_E is somewhat smaller than the upper bound of Theorem 10 when $R \neq 1/2$.

3.5 SELF-ORTHOGONAL CODES

We now consider in more detail those convolutional codes for which the set of parity checks orthogonal on $e_o^{(i)}$ is of the simplest kind. We define a self-orthogonal convolutional code to be a code in which the set of all parity checks, $s_u^{(j)}$, that check $e_o^{(i)}$ for i = 1, 2, ..., k_o, is itself a set of parity checks orthogonal on $e_o^{(i)}$.

With this definition, the codes used in the constructive proof of Theorems 10 and 11 are self-orthogonal codes. Let us restrict our attention to these codes and to rate R = 1/2. In this case there is only a single parity triangle, Eq. 82, corresponding to the single code-generating polynomial $G^{(2)}(D)$. The construction used to prove Theorem 10 gives the code for which the nonzero terms in $G^{(2)}(D)$ are

$$g_{2^{i-1}-1}^{(2)} = 1 \qquad i = 1, 2, \ldots, J . \tag{86}$$

The degree of $G^{(2)}(D)$ then is $m = 2^{J-1} - 1$. The set of J parity checks that check on $e_o^{(1)}$, namely

$$s_{2^{i-1}-1}^{(2)} \qquad i = 1, 2, \ldots, J , \tag{87}$$

is a set of J parity checks orthogonal on $e_o^{(1)}$, having an effective constraint length

$$n_E = \frac{1}{2} J^2 + \frac{1}{2} J + 1 . \tag{88}$$

On the other hand, the actual constraint length, n_A, is

$$n_A = n_o(m+1) = 2^J \tag{89}$$

from which fact we see that, for even moderate values of J, n_A is much greater than n_E.

A large ratio of n_A to n_E is generally undesirable for two reasons. First, both encoder complexity (cf. sec. 2.7) and decoder complexity (cf. sec. 4.2c) are directly proportional to n_A rather than to n_E. Second, the resynchronization period may be unacceptably long (cf. sec. 3.2) and thus may not provide an effective safeguard against the propagation of a decoding error. This latter fact can be seen best from an example. It follows from Eq. 67 that N = 9m is a reasonable choice if the actual information rate is not to be substantially less than the nominal rate R = 1/2. The resynchronization period, N + m, is then 10m time units. For J = 10, we have $m = 2^{J-1} - 1 = 511$ and thus the resynchronization period is 5110 time units. On the other hand, $m_E = 56$ for this code, and hence if n_A and n_E were equal (which would imply m = 27) the resynchronization period would be only 270 time units.

It is an interesting fact that self-orthogonal codes can be constructed with n_A much smaller than for the codes used to prove Theorem 10. We shall illustrate the technique for R = 1/2. The method is, for increasing j, to choose $g_j^{(2)} = 1$ when, and only when, the parity check $s_j^{(2)}$ contains no noise variable except $e_o^{(1)}$ in common with the preceding parity checks on $e_o^{(1)}$. Thus, for example, the J = 5 code formed in this manner has the parity triangle

$$\begin{array}{rcl}
1 & \rightarrow & 1 \\
2 & \rightarrow & 1\ \boxed{1} \\
 & & 0\ 1\ 1 \\
3 & \rightarrow & 1\ 0\ \boxed{1}\ \boxed{1} \\
 & & 0\ 1\ 0\ 1\ 1 \\
 & & 0\ 0\ 1\ 0\ 1\ 1 \\
 & & 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
4 & \rightarrow & 1\ 0\ 0\ 0\ \boxed{1}\ 0\ \boxed{1}\ \boxed{1} \\
 & & 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
 & & 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
 & & 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
 & & 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \\
5 & \rightarrow & 1\ 0\ 0\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ \boxed{1}\ 0\ \boxed{1}\ \boxed{1}
\end{array} \tag{90}$$

Here, we have adopted the convention of using a numbered arrow to indicate an orthogonal parity check and its size, n_i, and of placing a box about each nonzero coefficient (except the coefficient of $e_o^{(1)}$) of an information noise bit in an orthogonal parity check. This convention renders the orthogonality of the set of parity checks readily visible.

In Table I, we list the set of codes for even values of J up to J = 14 that are obtained by the preceding construction. The constraint length 2^J of Eq. 89 is listed for comparison. From Table I, we see that a great improvement in the n_A/n_E ratio is obtained for this second class of self-orthogonal codes, but the ratio is still large enough to require many additional storage devices in the encoder and decoder beyond those that would be required when the n_A/n_E ratio is almost unity.

Table I. Self-orthogonal codes for rate $\frac{1}{2}$.

R	J	n_E	n_A	2^J	code
$\frac{1}{2}$	2	4	4	4	$g_o^{(2)} = g_1^{(2)} = 1$
	4	11	16	16	add $g_3^{(2)} = g_7^{(2)} = 1$
	6	22	42	64	add $g_{12}^{(2)} = g_{20}^{(2)} = 1$
	8	37	90	256	add $g_{30}^{(2)} = g_{44}^{(2)} = 1$
	10	56	162	1024	add $g_{65}^{(2)} = g_{80}^{(2)} = 1$
	12	79	238	4096	add $g_{96}^{(2)} = g_{118}^{(2)} = 1$
	14	106	356	16384	add $g_{143}^{(2)} = g_{177}^{(2)} = 1$

("add" means that the code-generating polynomial is the same as for the previous code with the additions indicated.)

3.6 TRIAL-AND-ERROR CODES

The fact that the n_A/n_E ratio is much greater than unity for most self-orthogonal codes, fortunately, does not imply that this must always be the case for other convolutional codes. By trial-and-error techniques, it has been found possible to construct codes for which n_E meets or is smaller than the upper bound of Eq. 77 and for which n_A/n_E is almost unity. An extensive list of such codes is given in Table II for rates 1/2, 1/3, 1/5, and 1/10.

An example will serve both to indicate how Table II is to be read and to give some idea of how the codes were constructed. Consider the R = 1/2, J = 6 code in Table II. The code is listed as $(0, 6, 7, 9, 10, 11)^2$ which is our short notation for indicating that the code-generating polynomial $G^{(2)}(D)$ has as its nonzero coefficients

$$g_o^{(2)} = g_6^{(2)} = g_7^{(2)} = g_9^{(2)} = g_{10}^{(2)} = g_{11}^{(2)} = 1. \qquad (91)$$

The parity triangle (Eq. 82) then becomes

$$\begin{array}{rcllllllllllll}
1 & \rightarrow & 1 \\
 & & 0 & 1 \\
 & & 0 & 0 & 1 \\
 & & 0 & 0 & 0 & 1 \\
 & & 0 & 0 & 0 & 0 & 1 \\
 & & 0 & 0 & 0 & 0 & 0 & 1 \\
2 & \rightarrow & 1 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\
3 & \rightarrow & 1 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\
 & & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\
4 & \rightarrow & 1 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\
5 & \rightarrow & 1 & 1 & 0 & 1 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & \boxed{1} \\
6 & \rightarrow & 1 & 1 & 1 & 0 & 1 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & \boxed{1}
\end{array} \qquad (92)$$

Here, in addition to the convention used in Eq. 90, we have indicated with the lines to the right of the triangle which rows are added together to form the orthogonal parity checks. The rules for forming the orthogonal parity checks are listed in the table as 0^2, 6^2, 7^2, 9^2, $1^2 3^2 10^2$, $4^2 8^2 11^2$ which is short notation for

$$\begin{aligned}
A_1 &= s_o^{(2)} \\
A_2 &= s_6^{(2)} \\
A_3 &= s_7^{(2)} \\
A_4 &= s_9^{(2)} \\
A_5 &= s_1^{(2)} + s_3^{(2)} + s_{10}^{(2)} \\
A_6 &= s_4^{(2)} + s_8^{(2)} + s_{11}^{(2)}
\end{aligned} \qquad (93)$$

and this set of 6 parity checks orthogonal on $e_o^{(1)}$ corresponds to the parity checks constructed in (92). The sizes (n_i) of these parity checks are given in the table as 1, 2, 3, 4, 5, 6 which indicates that A_1 has $n_1 = 1$, A_2 has $n_2 = 2$, etc. The effective constraint length is given in the table as $n_E = 22$. For comparison, the bound of Eq. 77 is listed, and is also $n_E = 22$. The actual constraint length, $n_A = 24$, is also listed.

The manner in which this particular code was found is as follows: By beginning the second through the sixth rows of the parity triangle with "zeros," noise bits $e_6^{(1)}$ through $e_{11}^{(1)}$ can have a nonzero coefficient in only one row of the parity triangle of (92). Moreover, the second through the sixth rows can then be used to eliminate one nonzero coefficient for each of the variables $e_1^{(1)}$ through $e_5^{(1)}$, as is evident from the parity triangle. The first row gives a parity check on $e_o^{(1)}$ which checks no other information noise bit. Thus the problem reduces to choosing the last six rows of the parity triangle so that they can be combined to give a set of five parity checks that check $e_o^{(1)}$ but have the property that none of the variables $e_1^{(1)}$ through $e_5^{(1)}$ are checked more than twice in the set. The choice made in Eq. 92 fulfills these requirements.

The rest of the codes in Table II were hand-constructed by using this and numerous other techniques to exploit the parity-triangle structure. The simplicity of the concept of orthogonal parity checks makes it possible to construct codes of substantial length by hand.

A few more remarks on the trial-and-error codes are in order. Consider the code word in a rate $R = 1/n_o$ code for which $i_o^{(1)}$ is the only nonzero information bit. From Eqs. 28 and 29, it follows that the weight of this code word is one plus the number of nonzero terms in the code-generating polynomials $G^{(j)}(D)$, $j = 2, 3, \dots, n_o$. Thus there must be at least d-1 such nonzero terms, where d is the minimum distance of the code. On the other hand, the existence of J parity checks orthogonal on $e_o^{(1)}$ implies that d is at least J + 1, and hence that there must be at least J nonzero terms in the code-generating polynomials.

The trial-and-error codes in Table II (with the few exceptions marked by an asterisk) have the property that the number of nonzero terms in the code-generating polynomials is exactly J, which is the minimum number possible. This has the desirable result, as can be seen from Figs. 6 and 5, of reducing the number of inputs to the adders in the encoding circuits to the minimum number possible. Moreover, the minimum distance of these codes is exactly J + 1, since there is a code word with $i_o^{(1)} = 1$ which has this weight.

We shall say that a convolutional code with minimum distance d and rate $R = 1/n_o$ can be completely orthogonalized if d-1 parity checks orthogonal on $e_o^{(1)}$ can be formed. In this case, $e_o^{(1)}$ will be correctly decoded by majority decoding for any error pattern that has weight $\lfloor (d-1)/2 \rfloor$ or less. In other words, any error pattern that is guaranteed to be correctable by the minimum distance of the code is also correctable by majority decoding when the code can be completely orthogonalized. With this definition, the

Table II. Trial-and-Error Codes.

RATE	J	n_E	n_E (Eq.3-14)	n_A	CODE	RULES FOR ORTHOGONALIZATION	SIZES
1/2	2	4	4	4	$(0,1)^2$	$0^2,1^2$	1,2
	4	11	11	12	$(0,3,4,5)^2$	$0^2,3^2,4^2,1^25^2$	1,2,3,4
	6	22	22	24	$(0,6,7,9,$ $10,11)^2$	$0^2,6^2,7^2,9^2,$ $1^23^210^2,4^28^211^2$	1,2,3,4, 5,6
	8	37	37	44	$(0,11,13,16,$ $17,19,20,21)^2$	$0^2,11^2,13^2,16^2,$ $17^2,2^23^26^219^2,$ $4^214^220^2,$ $1^25^28^215^221^2$	1,2,3,4, 5,6, 7, 8
	10	56	56	72	$(0,18,19,27,$ $28,29,30,32,$ $33,35)^2$	$0^2,18^2,19^2,27^2,$ $1^29^228^2,10^220^229^2,$ $11^230^231^2,$ $13^221^223^232^2,$ $14^233^234^2,$ $2^23^216^224^226^235^2$	1,2,3,4, 5,6, 7, 8, 9, 10
	12	79	79	104	$(0,26,27,39,$ $40,41,42,44,$ $45,47,48,51)^2$	$0^2,26^2,27^2,39^2,$ $1^213^240^2,14^228^241^2$ $15^242^243^2,$ $17^229^231^244^2,$ $18^245^246^2,$ $2^23^220^232^234^247^2,$ $21^235^248^249^250^2,$ $24^230^233^236^238^251^2$	1,2,3,4, 5,6, 7, 8, 9, 10, 11, 12
1/3	2	3	3	3	$(0)^2(0)^3$	$0^2,0^3$	1,1
	4	7	7	9	$(0,1)^2(0,2)^3$	$0^2,0^3,1^2,2^3$	1,1,2,2
	6	13	13	15	$(0,1)^2$ $(0,2,3,4)^2$	$0^2,0^3,1^2,2^3,$ $1^33^3,2^24^3$	1,1,2,2, 3,3
	8	20	21	24	$(0,1,7)^2$ $(0,2,3,4,6)^3$	$0^2,0^3,1^2,2^3,$ $1^33^3,2^24^3,7^2,$ $3^25^26^26^3$	1,1,2,2, 3,3,3, 4

Table II. (continued).

RATE	J	n_E	n_E (Eq.3-14)	n_A	CODE	RULES FOR ORTHOGONALIZATION	SIZES
1/3	10	30	31	33	$(0,1,9)^2$ $(0,1,2,3,$ $5,8,9)^3$	$0^2,0^3,1^2,2^22^3,$ $9^2,3^34^3,3^25^25^3,$ $1^34^26^26^3,$ $8^28^3,7^39^310^3$	1,1,2,2, 3,3,3, 4, 5,5
	12*	40	43	54	$(0,4,5,6,7,$ $9,12,13,16)^2$ $(0,1,14,15,16)^3$	$0^2,0^3,1^21^3,4^2,$ $5^2,2^36^2,14^3,$ $7^210^211^211^3,$ $3^35^39^2,6^38^312^2,$ $3^316^317^3,$ $4^310^312^316^2$	1,1,2,2, 3,3,3, 4, 4,5, 5, 6
	14*	55	57	69	$(0,4,5,6,7,9,$ $12,13,16,19,$ $20,21)^2$ $(0,1,20,22)^3$	$0^2,0^3,1^21^3,4^2,$ $5^2,2^36^2,$ $7^210^211^211^3,$ $3^35^39^2,19^320^3,$ $22^3,6^38^312^2,$ $4^310^312^316^2,$ $3^27^313^315^319^2,$ $9^313^214^318^2$ $20^221^221^3$	1,1,2,2, 3,3, 4, 4,4, 4,5, 6, 7, 8
	16	68	73	108	$(0,4,5,6,7,$ $9,12,16,17,$ $30,31)^2$ $(0,1,22,$ $25,35)^3$	$0^2,0^3,1^21^3,4^2,$ $5^2,2^36^2,22^3,$ $7^210^211^211^3,$ $3^225^3,3^35^39^2,$ $6^38^312^2,$ $7^314^217^218^218^3,$ $9^316^219^220^220^3,$ $14^315^335^3,$ $12^321^328^231^232^2,$ $10^313^319^326^329^330^2$	1,1,2,2, 3,3,3, 4, 4,4, 5, 5, 6, 7, 8, 9

Table II. (continued).

RATE	J	n_E	n_E (Eq. 3-14)	n_A	CODE	RULES FOR ORTHOGONALIZATION	SIZES
1/5	4	5	5	5	$(0)^{2}(0)^{3}(0)^{4}(0)^{5}$	$0^{2},0^{3},0^{4},0^{5}$	1,1,1,1
	6	9	9	10	add $(1)^{2}(1)^{3}$	$1^{2}1^{4},1^{3}1^{5}$	2,2
	8	13	13	15	add $(2)^{2}(2)^{4}$	$2^{2}2^{3},2^{4}2^{5}$	2,2
	10	18	19	20	add $(3)^{2}(3)^{5}$	$3^{5},3^{2}3^{3}$	2,3
	13	27	29	30	add $(4)^{2}(5)^{4}(5)^{5}$	$3^{4}4^{2}4^{4},5^{5},4^{3}5^{3}5^{4}$	3,3,3
	14	30	33	35	add $(6)^{4}$	$4^{5}6^{4}$	3
	16	37	41	45	add $(8)^{3}(7)^{4}$	$8^{3},5^{2}6^{3}7^{2}7^{4}$	3,4
	18	47	51	55	$(0,1,2,3,5,6,$ $8,10)^{2}(0,3,5,$ $6,8)^{3}(0,1)^{4}$ $(0,2,10)^{5}$	$0^{2},0^{3},0^{4},0^{5},$ $1^{2},2^{2}2^{4},3^{3},1^{4}1^{5},$ $2^{3}2^{5},3^{2}4^{2},3^{4}5^{2}5^{4},$ $9^{4}10^{2}10^{3},5^{3},3^{5}6^{3},$ $10^{5},1^{3}4^{4}6^{2}6^{4},$ $7^{2}7^{4}8^{2}9^{2},$ $4^{3}5^{5}7^{3}8^{3}8^{4}$	1,1,1,1, 2,2,2,2, 2,3,3, 3,3,3, 3,4, 5, 5
	20	55	61	65	add $(10)^{4}(12)^{3}$	$6^{5}9^{5}12^{3},$ $10^{4}11^{5}12^{4}12^{5}$	4, 4
	22*	65	73	80	add $(11,13,14)^{2}$ $(15)^{5}$	$4^{5}13^{4}14^{2}14^{3},$ $13^{5}14^{4}15^{4}15^{5}$	4, 6
1/10	9	10	10	10	$(0)^{2}(0)^{3}(0)^{4}$ $(0)^{5}(0)^{6}(0)^{7}$ $(0)^{8}(0)^{9}(0)^{10}$	$0^{2},0^{3},0^{4},$ $0^{5},0^{6},0^{7},$ $0^{8},0^{9},0^{10}$	1,1,1, 1,1,1, 1,1,1
	14	20	20	20	add $(1)^{6}(1)^{7}$ $(1)^{8}(1)^{9}(1)^{10}$	$1^{2}1^{6},1^{3}1^{7},$ $1^{4}1^{8},1^{6}1^{9},1^{10}$	2,2, 2,2,2
	18	28	28	30	add $(2)^{2}(2)^{3}$ $(2)^{6}(2)^{9}$	$2^{2},2^{3}2^{4},$ $2^{6}2^{7},2^{8}2^{9}$	2,2, 2,2

Table II. (continued).

RATE	J	$n_{E_{(Eq.}}$	$n_{E_{3-14)}}$	n_A	CODE	RULES FOR ORTHOGONALIZATION	SIZES
1/10	23	39	43	40	add $(3)^2(3)^4$ $(3)^8(3)^9(3)^{10}$	$3^2 3^3, 3^4, 3^7 3^8,$ $3^6 3^9, 2^5 3^5 3^{10}$	2,2,2, 2,3
	26	48	52	50	add $(4)^2(4)^8(4)^9$	$4^8 4^{10}, 2^{10} 4^2,$ $4^3 4^4 4^7 4^9$	2,3, 4
	33	69	79	70	add $(6)^2(5)^5$ $(5,6)^6(5,6)^9$ $(6)^{10}$	$5^5, 4^6 5^4 5^6,$ $4^5 5^2 5^9, 5^7 6^5 6^6,$ $5^8 6^4 6^9, 5^3 6^{10},$ $6^2 6^3 6^7 6^8$	2,3, 3,3, 3,3, 4
	38	88	101	90	add $(7)^6(7,8)^9$ $(8)^7(8)^8$	$5^{10} 7^6 7^{10}, 8^7,$ $7^2 7^5 7^7 7^9,$ $7^3 7^4 8^5 8^9,$ $7^8 8^3 8^4 8^5 8^8$	3,3, 4, 4, 5

. . . .

Codes marked with an asterisk have J + 2 nonzero terms in the code-generating polynomials; all other codes have the minimum possible number of nonzero terms, namely J.

The symbols used in Table II have the following meanings:

J = number of parity checks orthogonal on $e_o^{(1)}$.

n_E = effective constraint length of the code.

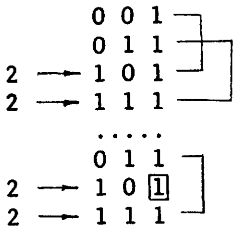
n_A = actual constraint length of the code.

codes of Table II (with the possible exceptions of those codes marked with an asterisk) can be completely orthogonalized.

3.7 UNIFORM CONVOLUTIONAL CODES

In contrast to the codes of the previous section (which were constructed by cut-and-try procedures) we shall next study some classes of convolutional codes for which a systematic formulation can be given and which can be completely orthogonalized. In this section, we show the existence of a class of convolutional codes that satisfy the distance bound of Theorem 7 with the equality sign, and which can be completely orthogonalized.

For any integers L and M, we consider the set of $L2^M-1$ binary (M+1)-tuples formed as shown below for L = 2 and M = 2.



(94)

That is, there are L sets of rows formed, and every row has a "one" in the last, or $M+1^{th}$, position. The first L-1 sets of rows each contain the set of all 2^M M-tuples in the first M positions. The last set contains all 2^M-1 nonzero M-tuples in the first M positions.

Now suppose that the rows in (94) are the <u>last rows</u> in a set of $L2^M-1$ parity triangles. The complete parity triangles then must be

$$\begin{array}{lll}
1 & & \\
0 & 1 & \\
0 & 0 & 1 \\
1 & & \\
1 & 1 & \\
0 & 1 & 1 \\
1 & & \\
0 & 1 & \\
1 & 0 & 1 \\
1 & & \\
1 & 1 & \\
1 & 1 & 1 \\
1 & & \\
1 & 1 & \\
0 & 1 & 1 \\
1 & & \\
0 & 1 & \\
1 & 0 & 1 \\
1 & & \\
1 & 1 & \\
1 & 1 & 1
\end{array}$$

and these are the parity triangles corresponding to the matrix H_{Δ} of a code with rate $R = 1/n_o$, where $n_o = L2^M$. We now determine the number of parity checks orthogonal on $e_o^{(1)}$ which can be formed for this code.

The last rows in the parity triangles can be added as shown in (94) to produce $L2^{M-1}$ parity checks of size two orthogonal on $e_o^{(1)}$. That the last rows can always be so combined is proved as follows: There are L-1 sets of last rows each having 2^M rows. These rows all have "ones" in the last position and have the set of all 2^M M-tuples in the first M positions. In each such set of rows, for every row beginning with a "one" there is a unique row beginning with a "zero" that is otherwise identical. The sum (modulo-two) of each such pair of rows corresponds to a parity check that checks $e_o^{(1)}$ and no other information noise bits. The size of the parity check is two because two parity checks are added and this implies that two parity noise bits are checked. Thus 2^{M-1} parity checks of size two orthogonal on $e_o^{(1)}$ can be formed from each of these L-1 sets of last rows. Finally, there is one set of last rows all of which have "ones" in the last position and have the set of all 2^M-1 nonzero M-tuples in the first M positions. The parity checks in this set can be combined as before, except that because the row 0 0 ... 0 1 is missing the 1 0 ... 0 1 must be used alone. This row corresponds to a parity check on $e_o^{(1)}$ and $e_M^{(1)}$. Thus an additional set of 2^{M-1} parity checks orthogonal on $e_o^{(1)}$ can be formed from this set of last rows. In all, $L2^{M-1}$ parity checks orthogonal on $e_o^{(1)}$ can be formed by using only the last rows of the parity triangles, and $e_M^{(1)}$ is the only information noise bit, exclusive of $e_o^{(1)}$, that is checked.

The same process can then be repeated for the next-to-last rows of the parity triangles. If the last rows are as given in (94), then the next-to-last rows are

$$\begin{array}{cc}
0 & 1 \\
1 & 1 \\
\multicolumn{2}{c}{\cdots} \\
0 & 1 \\
1 & 1 \\
\multicolumn{2}{c}{\cdots} \\
0 & 1 \\
1 & 1 \\
\multicolumn{2}{c}{\cdots} \\
1 & 1
\end{array} \tag{95}$$

and this is another set of rows of the same type as (94) with L' = 2L and M' = M - 1. Thus another set of $L'2^{M'-1} = L2^{M-1}$ parity checks of size two orthogonal on $e_o^{(1)}$ can be formed from the next-to-last rows of the parity triangle. The union of this set with the set of parity checks formed from the last rows is a set of $2L2^{M-1}$ parity checks orthogonal on $e_o^{(1)}$, since the only information noise bit, $e_{M'}^{(1)}$, checked by the former set, is distinct from the only information noise bit, $e_M^{(1)}$, checked by the latter set.

This same process can be repeated for the third-to-last, the fourth-to-last, etc., rows of the parity triangles, until the first rows are reached. The process is thus performed a total of M times, giving $ML2^{M-1}$ parity checks of size two in all. The first rows each correspond to a parity check that checks $e_o^{(1)}$ and no other information noise bit. Thus an additional $L2^M-1$ checks of size one orthogonal on $e_o^{(1)}$ can be formed from these rows.

The total number of parity checks orthogonal on $e_o^{(1)}$ that can be formed from the parity triangles then is

$$J = (M)L2^{M-1} + (L2^M-1) = L(M+2)2^{M-1} - 1. \tag{96}$$

Using Eq. 75, we find the effective constraint length to be

$$n_E = 1 + 2(M)L2^{M-1} + (L2^M-1) = L(M+1)2^M. \tag{97}$$

For this code, we have $m = M$ and $n_o = L2^M$, hence

$$n_A = (m+1)n_o = L(M+1)2^M = n_E. \tag{98}$$

By the corollary to Theorem 1, the minimum distance, d, must be at least J + 1 or $L(M+2)2^{M-1}$. On the other hand, from Eq. 53, we find that

$$d_{avg} = \frac{1}{2}(n_A+n_o) = L(M+2)2^{M-1}. \tag{99}$$

Thus the minimum distance must be exactly $L(M+2)2^{M-1}$, since it cannot exceed the average distance.

We summarize these results in Theorem 13.

THEOREM 13: For any integers L and M, there exists a convolutional code with $R = 1/n_o$ and $n_o = L2^M$ and with constraint length $n_A = n_E = L(M+1)2^M$ which is such that the minimum distance d satisfies

$$d = d_{avg} = L(M+2)2^{M-1},$$

and so that the code can be completely orthogonalized.

We call a convolutional code with $d = d_{avg}$ a uniform code. It seems probable that there are no uniform binary convolutional codes with $R = 1/n_o$ other than those given by Theorem 13. In Table III we list the set of codes with L = 1 for M up to 6.

Table III. Some uniform binary convolutional codes.

M	R	n_A	d
1	1/2	4	3
2	1/4	12	8
3	1/8	32	20
4	1/16	80	48
5	1/32	192	112
6	1/64	448	256

3.8 "REED-MULLER-LIKE" CODES

Another class of convolutional codes that can be completely orthogonalized will now be presented. The method of construction is best understood through an example.

Consider the array

$$\begin{array}{cccccccc} & v_2v_3 & v_1v_3 & v_1v_2 & v_3 & v_2 & v_1 & v_o \\ 4 \rightarrow & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 4 \rightarrow & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ & 0 & 0 & 0 & 1 & 0 & 1 & \boxed{1} \end{array} \qquad (100)$$

which is formed by taking v_1, v_2, and v_3 in such a way that the set of 3-tuples in their rows is the set of all nonzero 3-tuples, and then adding the all "one" column vector to the list and the column vectors formed by taking all vector inner products of v_1, v_2, and v_3 taken two at a time. Again suppose these are the last rows in a set of n_o-1 parity triangles. Then, as shown above, we can add these rows to form a set of $2^{3-2}=2$ parity checks orthogonal on $e_o^{(1)}$, each of size $n_i=2^2=4$. Except for $e_o^{(1)}$, the only information noise bit checked is $e_6^{(1)}$. The same process can be repeated in the second-to-last and third-to-last rows of the parity triangles. The remaining rows of the parity triangle are the same as those for the L = 1, M = 3 uniform code in section 3.7. Thus, in all, for this code: $\binom{3}{2}2^{3-2}$ parity checks of size 2^2, plus $\binom{3}{1}2^{3-1}$ parity checks of size 2^1, plus $\binom{3}{0}2^3-1$ parity checks of size 2^0, all orthogonal on $e_o^{(1)}$, can be formed.

When this example is generalized so that the array contains all vector inner products of $v_1, v_2, \ldots, v_M$ taken K or fewer at a time, we obtain Theorem 14.

THEOREM 14: For any integers M and K with the property that K ≦ M, there exists a convolutional code with $R = 1/2^M$ and $n_A = n_E = \left[\sum_{i=0}^{K} \binom{M}{j} 2^M\right]$ so that

$$d = \sum_{j=0}^{K} \binom{M}{j} 2^{M-j} \qquad (101)$$

and so that the code can be completely orthogonalized.

PROOF 14: The proof involves two steps. First, we must show that the number, J, of parity checks orthogonal on $e_o^{(1)}$ is one less than the sum in (101). Second, we must show that the minimum distance of the code is exactly equal to this sum.

We begin by noting that if a lowermost row of 0 0 0 0 0 0 1 is added to the array in (100), it becomes the transpose of the generating matrix for a second-order (K = 2) Reed-Muller code. (This is the reason for which we have referred to the codes in this section as "Reed-Muller-like codes.") Reed has shown[2] that, for the generating matrix of the K^{th}-order code, the columns can be grouped into 2^{M-K} sets of 2^K columns each, in such a manner that the sum of the columns in each set has a "one" only in the first row. (Reed's proof is quite lengthy and will not be reproduced here.) Thus the rows in the transposed matrix can be grouped into 2^{M-K} sets, of 2^K rows each, in such a manner that the sum of the rows in each set has a "one" only in the first column. If the row 0 0 . . . 0 1 is omitted from the transposed matrix, one set of rows will contain 2^K-1 rows, and the sum of these rows will be 1 0 . . . 0 1. Thus, if the rows in the transposed matrix, with the row 0 0 . . . 0 1 omitted, are the last rows in a set of parity triangles, it is possible to form 2^{M-K} parity checks orthogonal on $e_o^{(1)}$, each of size $n_i = 2^K$, and such that there is a single information noise bit, excluding $e_o^{(1)}$, that is checked by these 2^{M-K} parity checks.

The same process can be carried out on the second-to-last, the third-to-last, etc. rows of the parity triangles. After $\binom{M}{K}$ rows have been processed, the remaining rows reduce to the K-1 order code. The union of the sets of orthogonal parity checks, formed from each set of rows of the parity triangles, itself is orthogonal on $e_o^{(1)}$, since there is only one information noise bit checked by the set formed from each set of rows, and these bits are all distinct.

The total number, J, of parity checks orthogonal on $e_o^{(1)}$ that are formed in this manner is

$$J = \sum_{j=0}^{K} \binom{M}{j} 2^{M-j} - 1 \tag{102}$$

which is one less than the sum in Eq. 101, as claimed.

It remains to show that the minimum distance is given exactly by (101). The minimum distance is at least J + 1, and hence must be at least as large as the sum in (101). It suffices, then, to show that there is some code word with $i_o^{(1)} = 1$ which has exactly this weight. We recall that there is an initial code word, with $i_o^{(1)} = 1$, whose weight is one plus the number of "ones" in the code-generating polynomials. This last number is just the number of "ones" in the first columns of the parity triangles and, by the symmetry of each triangle, this is equal to the number of "ones" in the last rows of the parity triangles. Since the Reed-Muller generating matrix has one more "one" than the set of last rows of the parity triangles (it includes an extra column 0 0 . . . 0 1), it follows that there is a code word with $i_o^{(1)} = 1$ that has weight equal to the number of "ones" in the generating matrix of a K^{th}-order Reed-Muller code. But any vector inner product of v_1, v_2, . . . v_M taken j at a time contains exactly 2^{M-j} "ones." The number of "ones" in the generating matrix then is

$$\sum_{j=0}^{K} \binom{M}{j} 2^{M-j}, \tag{103}$$

and thus there is an initial code word, with $i_o^{(1)} = 1$, having this weight. This equals the sum in Eq. 101, which was to be shown.

In Table IV, we list the Reed-Muller-like convolutional codes for M up to 5. The M, K code, where K = 1, is the same as the M, L uniform code (in section 3.7), where L = 1.

There are important differences between the codes in this section and the ordinary Reed-Muller codes besides the obvious one that the former are convolutional codes and the latter are block codes. With fixed M, the block length of the Reed-Muller codes is 2^M, and adding the higher order vector products to the generating matrix increases the rate but reduces the minimum distance. On the other hand, for the Reed-Muller-like convolutional codes, it is the rate $R = 1/2^M$ that is fixed, and adding the higher order vector inner products increases both the constraint length and the minimum distance.

Table IV. Reed-Muller-like convolutional codes.

M	K	R	$n_A = n_E$	d
1	1	1/2	4	3
2	1	1/4	12	8
	2		16	9
3	1	1/8	32	20
	2		56	26
	3		64	27
4	1	1/16	80	48
	2		176	72
	3		240	80
	4		256	81
5	1	1/32	192	112
	2		512	192
	3		832	232
	4		992	242
	5		1024	243

3.9 SUMMARY

We have constructed a fairly extensive set of codes, suitable for threshold decoding, by both cut-and-try and analytical procedures. For the case R = 1/2, we have been able to show that the effective constraint length must grow as the square of the number of constructed orthogonal parity checks, and to form codes that achieved this bound. For other rates, we have not been able to derive a good lower bound on effective constraint length, but we conjecture that a similar bound applies as the number of orthogonal parity checks increases indefinitely.

The upper bound of Theorem 10 was found to give the smallest possible value of n_E for R = 1/2, but not for lower rates. For example, in the limit as $M \rightarrow \infty$, the uniform codes with L = 1 have an effective constraint length that is approximately 2/M times the bound of Theorem 10, but these are very low-rate codes. The effective constraint length of the Reed-Muller-like codes is also much smaller than the bound of Theorem 10, for the very low-rate codes.

The construction used to prove Theorem 10 gave codes for which the effective constraint length, n_E, was much smaller than the actual constraint length, n_A. On the other hand, the trial-and-error codes were formed so that n_E and n_A were approximately equal. For both the uniform codes, and the Reed-Muller-like codes, n_E and n_A were identical. We have already emphasized the importance of having n_A as small as possible.

A final remark on the trial-and-error codes of Table II seems to be in order. These codes were all hand-constructed, by using techniques of the nature described in section 3.6. For several reasons, it does not seem feasible to make a computer search for such codes. The special technique by which each code was constructed was developed after a careful study of the appropriate parity triangles, and no systematic set of rules was found which would enable the construction of all codes in Table II, or even of all of the codes of a fixed rate. Many of the codes required somewhat "ingenious" tricks in their construction which would not be readily programmable. At best, it seems that a computer could be used for making searches for some of the longer codes with a high degree of operator control of the program.

In Section IV we shall give threshold-decoding circuits that can be used with any of the codes in Section III. Following that, we shall give numerical and analytical results for the error probabilities that can be obtained when the codes in Section III are threshold-decoded.

IV. THRESHOLD-DECODING CIRCUITS FOR BINARY CONVOLUTIONAL CODES

We shall present digital circuits that can be used to instrument the threshold-decoding algorithms for binary convolutional codes.

It is convenient at this point to state and prove a lemma that will be used frequently in the following sections.

LEMMA 2: Let e_1, e_2, ... e_{n_i} be a set of n_i statistically independent, binary-valued random variables for which $\Pr[e_j=1] = 1 - \Pr[e_j=0] = \gamma_j$. Then the probability, p_i, that an odd number of these random variables are "ones" is

$$p_i = \frac{1}{2}\left[1 + \prod_{j=1}^{n_i} (1-2\gamma_j)\right]. \tag{104}$$

PROOF: The technique of "enumerating, or generating, functions"[27] is useful here. The enumerating function, $g_j(s)$, for the j^{th} random variable is $\gamma_j s + (1-\gamma_j)$, that is, it is the polynomial in s for which the coefficient of s^v is the probability that the random variable takes on value v. Since the random variables are statistically independent, the enumerating function, $g(s)$, of their sum as real numbers is

$$g(s) = \prod_{j=1}^{n_i} \quad g_j(s) = \prod_{j=1}^{n_i} [p\gamma_j s+(1-\gamma_j)]. \tag{105}$$

The desired probability, p_i, is then the sum of the coefficients of odd powers of s in $g(s)$, hence

$$p_i = \frac{1}{2}[g(1)-g(-1)]. \tag{106}$$

Substituting (105) in (106), and noting that $g(1) = 1$, we obtain Eq. 104.

Before proceeding to a specification of actual decoding circuits, we shall first restate the threshold-decoding algorithms of Theorems 3 and 4 in their specific form for binary convolutional codes with rate $R = 1/n_o$. For this case, given a set $\{A_i\}$ of J parity checks orthogonal on $e_o^{(1)}$, the threshold-decoding algorithms can be stated thus: Choose $e_o^{(1)} = 1$ if, and only if,

$$\sum_{i=1}^{J} w_i A_i > T. \tag{107}$$

Here,

(a) $\{A_i\}$ are treated as real numbers in this sum;

(b) w_i are the <u>weighting factors</u> and are given by

$$w_i = 1 \qquad \text{for majority decoding, and} \tag{108}$$

$$w_i = 2 \log \frac{q_i}{p_i} \qquad \text{for APP decoding;} \tag{109}$$

$p_i = 1 - q_i$ is the probability of an odd number of "ones" in the n_i noise bits, exclusive of $e_o^{(1)}$, which are checked by A_i; and

(c) T is the threshold and is given by

$$T = \left\lceil \frac{1}{2} J \right\rceil \qquad \text{for majority decoding, and} \tag{110}$$

$$T = \frac{1}{2} \sum_{i=0}^{J} w_i \qquad \text{for APP decoding,} \tag{111}$$

where $p_o = 1 - q_o = \Pr[e_o^{(1)}=1]$, and we set $w_o = 2 \log \frac{q_o}{p_o}$.

This algorithm is just a rewording of the decoding rules of Theorems 3 and 4, and we shall now consider specific circuits for its implementation. For convenience, we shall restrict the discussion to rates $R = 1/n_o$, and indicate afterwards how the general case, $R = k_o/n_o$, is handled.

4.1 DECODING FOR THE BINARY SYMMETRIC CHANNEL

We begin with the simplest case for which the noise probability is the same for all received bits, that is, $\Pr(e_u^{(j)}=1) = p_o$ for all u and j. The only channel that meets this specification is the binary symmetric channel discussed in Section I. The decoding circuits for APP decoding are especially simple in this case because the weighting factors $\{w_i\}$ and the threshold T are all constants. (They are, of course, always constants for majority decoding, and hence the circuits presented in this section can be used for majority decoding with any binary channel.) For example, consider a particular parity check, A_i, of size n_i. Applying Lemma 2, we have

$$p_i = 1 - q_i = \frac{1}{2}\left[1-(1-2p_o)^{n_i}\right] \tag{112}$$

and this depends only on n_i, since p_o is a constant. Thus the $\{w_i\}$ of (109) and the T of (111) are all constants.

a. Type I Decoder

The first circuit that we shall present for implementing (107) is shown in Fig. 11. That this circuit is a proper threshold decoder can be seen in the following manner.

The parity-check sequences, $S^{(j)}(D)$ $j = 2, 3, \ldots n_o$, are first formed by encoding the received information sequence, $R^{(1)}(D)$, and adding the parity sequences so formed to the received parity sequences, $R^{(j)}(D)$ $j = 2, 3, \ldots n_o$, as explained in section 2.7. (Binary addition and subtraction are identical, since 1 + 1 = 0 implies that 1 = -1.) The $n_o - 1$ parity-check sequences are then stored in as many shift registers. The outputs of the storage devices at time m, when the decision on $e_o^{(1)}$ is to be made, are as shown in Fig. 11.

The $\{A_i\}$ are then formed by the set of J modulo-two adders below the shift registers in Fig. 11. The inputs to the ith adder are just the set of parity checks $s_u^{(j)}$ that

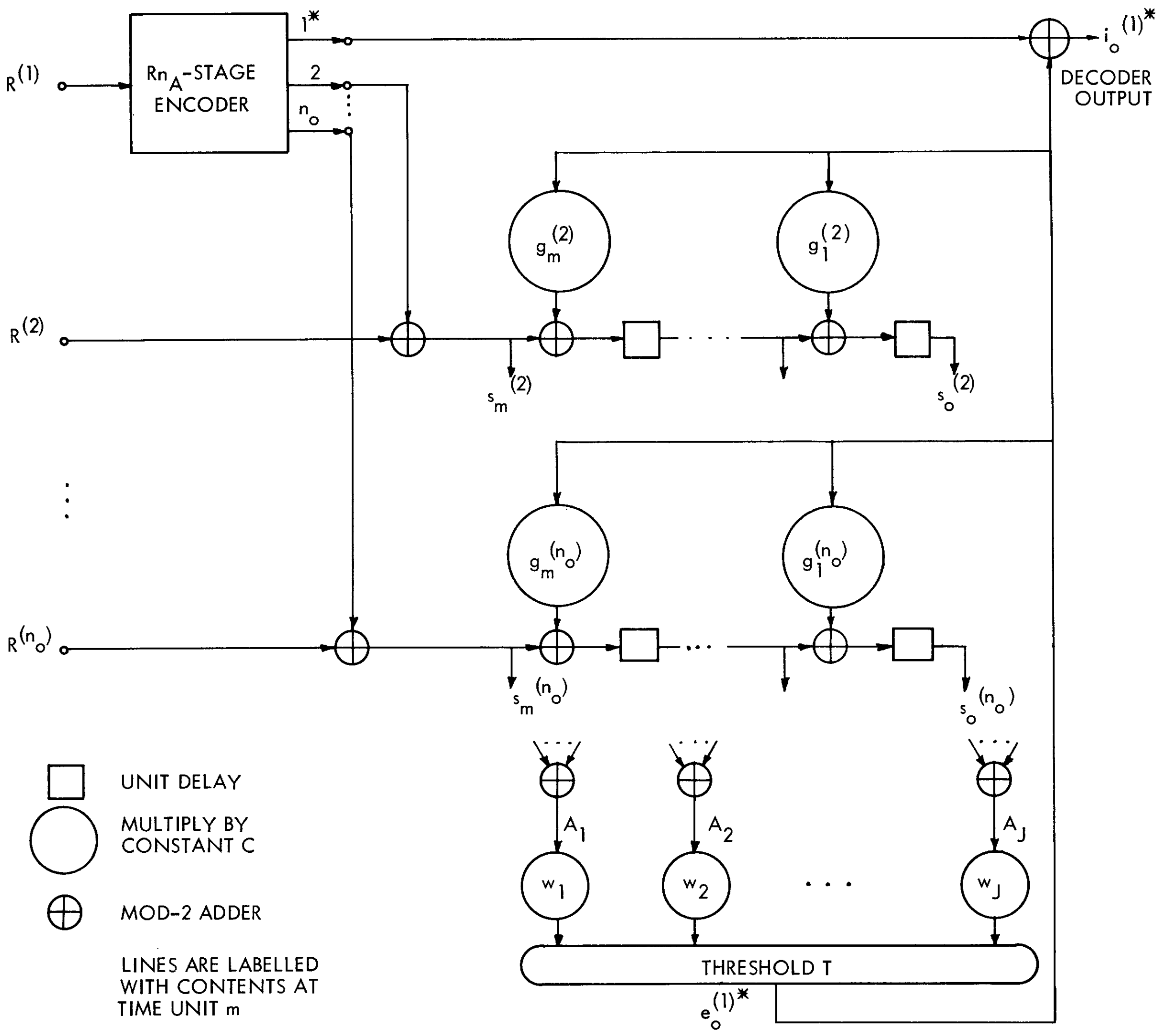


Fig. 11. Type I decoder.

are added to form A_i. The outputs of these J adders are then treated as real numbers and weighted by the factors w_i of (108) or (109). This weighted sum is then compared with the constant T given by (110) or (111). Thus the output of the threshold device is $e_o^{(1)*}$, the decoded estimate of $e_o^{(1)}$. This output is then added modulo-two to $r_o^{(1)}$, which at time m is just emerging from the delayed input terminal of the encoder, to produce $i_o^{(1)*}$, the decoded estimate of $i^{(1)}$.

Finally, consider the altered parity-check sequences given by

$$S^{(j)*}(D) = S^{(j)}(D) - \left[e_o^{(1)*}\right] G^{(j)}(D). \tag{113}$$

From Eq. 63, it follows that if decoding has been correct, that is, if $e_o^{(1)} = e_o^{(1)*}$, then the effect of $e_o^{(1)}$ has been removed from the altered parity-check sequences.

Moreover, terms from time 1 through time m + 1 have exactly the same structure with respect to $e_1^{(1)}$ as the original terms from time 0 through time m had with respect to $e_o^{(1)}$. The decoding for $e_1^{(1)}$ can then be performed by using exactly the same algorithm on the altered parity-check sequence from time 1 through time m + 1 as was used in decoding $e_o^{(1)}$ from the original parity-check sequences from time 0 through time m. Thus, barring a decoding error, the Type I decoder will continue to operate correctly at successive time instants when the parity-check sequences are modified according to Eq. 113. This is seen to be accomplished in Fig. 11 by feeding back the output, $e_o^{(1)*}$, of the threshold element to the modulo-two adders between stages of the shift registers that store the parity-check sequences. The connections to the adders correspond to the code-generating polynomials, $G^{(j)}(D)$.

A few more remarks concerning the Type I decoder are in order. First, the shift registers used to store the parity-check sequence contain $(n_o-k_o)m = (n_A-n_o)(1-R)$ stages in all. The encoder section contains an additional $(n_A-n_o)R$ stages. Thus there is a total of $n_A - n_o$ stages of shift register in the Type I decoder. Second, since the all-zero sequences form a legitimate initial code word in any convolutional code, it follows that the received sequences can be fed into the decoder, beginning at time zero, without any need to disable the threshold decision element until time m when the decision on $e_o^{(1)}$ is to be made. On the other hand, the information symbols output from the decoder up to time m must all be zeros if the decoding is correct. The decoder output should then be monitored over this time span, and any "one" output taken as indication of an error pattern that is likely to cause $e_o^{(1)}$ to be decoded incorrectly.

b. Type II Decoder

A second circuit that implements the threshold-decoding algorithms is shown in Fig. 12. Since this circuit applies the algorithms in a form that is quite different from that given above, it will be necessary at this point to present the theory behind the circuit in Fig. 12.

Each one of a set $\{A_i\}$ of parity checks orthogonal on $e_o^{(1)}$ can be written as a sum of noise bits, one of which is $e_o^{(1)}$, that is,

$$A_i = e_o^{(1)} + \sum_{j=1}^{n_i} e_{\alpha_j}^{(\beta_j)}, \quad i = 1, 2, \ldots J. \tag{114}$$

Since A_i is a parity check, (114) can also be written

$$A_i = r_o^{(1)} + \sum_{j=1}^{n_i} r_{\alpha_j}^{(\beta_j)}, \quad i = 1, 2, \ldots J. \tag{115}$$

We now define the quantity B_i to be the sum of the received bits, excluding $r_o^{(1)}$, which appear in the expression for A_i, that is,

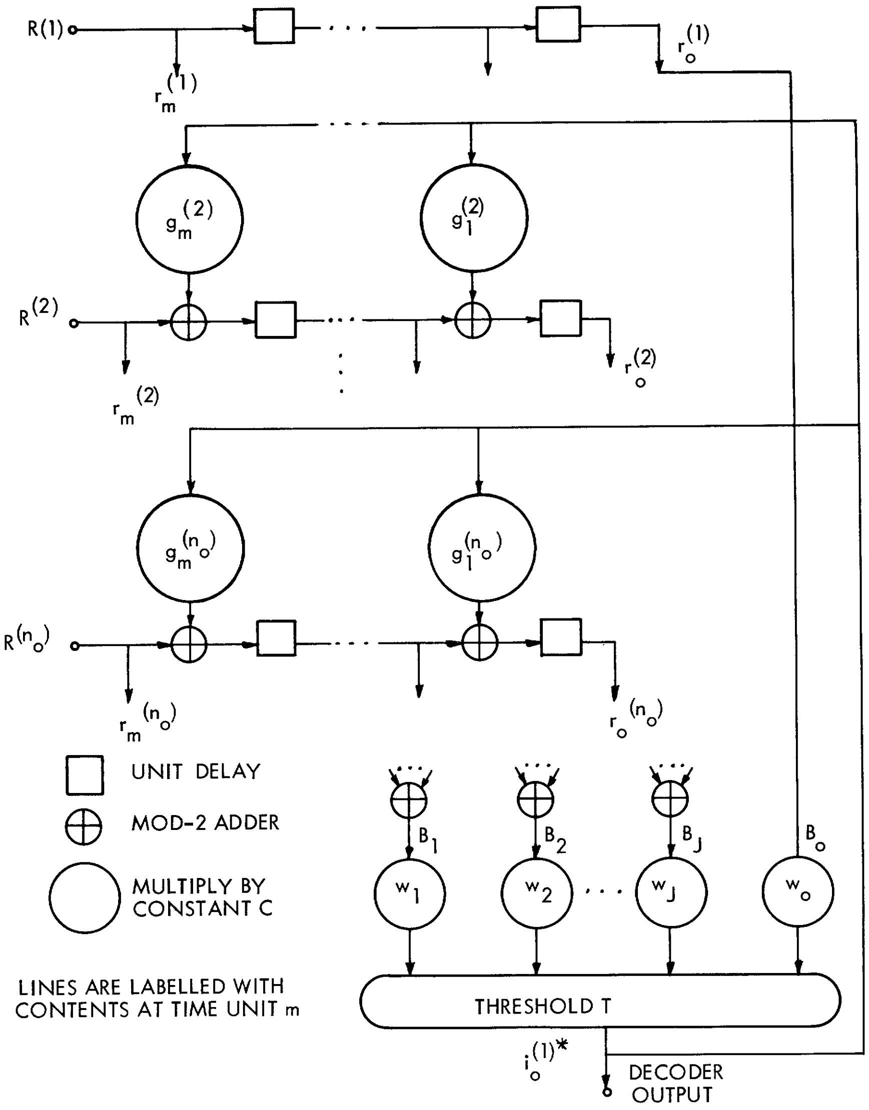


Fig. 12. Type II decoder.

$$B_i = \sum_{j=1}^{n_i} r_{\alpha_j}^{(\beta_j)}, \quad i = 1, 2, \ldots J. \tag{116}$$

Recalling that $r_o^{(1)} = i_o^{(1)} + e_o^{(1)}$ and substituting (114) and (115) in (116), we obtain

$$B_i = i_o^{(1)} + \sum_{j=1}^{n_i} e_{\alpha_j}^{(\beta_j)}, \quad i = 1, 2, \ldots J. \tag{117}$$

For convenience, we define B_o as

$$B_o = r_o^{(1)} = i_o^{(1)} + e_o^{(1)}. \tag{118}$$

From Eqs. 118 and 117, it can be seen that the $\{B_i\}$ form a set of $j+1$ equations (but not parity checks) each of which is the sum of $i_o^{(1)}$ and a set of noise bits that are such that no noise bit enters into more than one equation. By a development exactly parallel to that of section 1.1, we find that the threshold decoding rules become:

Choose $i_o^{(1)} = 1$ if, and only if,

$$\sum_{i=0}^{J} w_i B_i > T, \tag{119}$$

where the $\{B_i\}$ are treated as real numbers in this sum, and the $\{w_i\}$ and T have the same meanings as in Eqs. 108-111. When J is odd, the majority decoding rule given here is not exactly the same as that of (107). For a full equivalence in this instance, the rules of (119) should be stated: Choose

$$i_o^{(1)} = r_o^{(1)} \quad \text{when} \quad \sum_{i=0}^{J} B_i = \frac{J+1}{2}.$$

This algorithm performs the same decoding operation as the algorithm stated in (107), the only difference being that the former gives $i_o^{(1)*}$ (the decoded estimate of $i_o^{(1)}$) directly, whereas the latter gives $e_o^{(1)*}$ from which the decoded estimate of $i_o^{(1)}$ can be found as $i_o^{(1)*} = r_o^{(1)} - e_o^{(1)*}$.

The operation of the Type II decoder shown in Fig. 12 can now be readily explained. The received sequences are stored in n_o shift registers so that the received bits are available for the formation of the $\{B_i\}$. The $\{B_i\}$ are formed by the set of adders beneath the shift registers, the inputs to the i^{th} such adder being the set of n_i received bits in Eq. 116 whose sum is B_i. The adder outputs are then the $\{B_i\}$, and these are weighted and compared with the threshold in accordance with (119). The output of the threshold element is then $i_1^{(1)*}$. Finally, the circuit is prepared to operate correctly at successive time instants by altering the received sequences according to Eq. 65. This is accomplished by feeding back the output, $i_o^{(1)*}$, of the threshold device to the modulo-two adders between stages of the shift register, with the connections to the adders corresponding to the code-generating polynomials $G^{(j)}(D)$.

The Type II decoder contains n_o shift registers of m stages each for a total of $mn_o = n_A - n_o$ stages of shift register. This is the same total number as for the Type I decoder. Moreover, as for the Type I decoder, the fact that the all-zero sequences are valid code words means that the threshold element need not be disabled from time zero when the inputs are first applied to time m when the decision on $i_o^{(1)}$ is made. However, unlike the Type I decoder, the Type II decoder cannot also serve as an encoder without modification.

c. Summary

The material of the preceding discussions can be summarized in a theorem.

THEOREM 15: A binary convolutional code with rate $R = 1/n_o$ and constraint length n_A can be majority-decoded for any binary output channel, or APP-decoded for the binary symmetric channel, by a sequential network containing $n_A - n_o$ stages of shift register and one threshold logical element.

It is interesting to note that the components in both Type I and Type II decoders work at the rate of one operation per time unit, whereas the received bit rate is n_o bits per time unit. This fact permits the use of lower speed components in construction of the decoders.

Also, it seems plausible that these decoders contain the minimum storage possible for a convolutional decoder. This can be shown in the following manner. Since n_A received bits are required for making the decision on $e_o^{(1)}$ and n_o bits are received at any time instant, at least $n_A - n_o$ received bits (or their equivalents) must be stored in the decoder.

Finally, it should now be clear that for the case $R = k_o/n_o$ the Types I and II decoders would be modified to include a total of k_o threshold devices, each with its own set of adders to form the set of parity checks orthogonal on $e_o^{(j)}$ for $j = 1, 2, \dots k_o$. The outputs of the threshold elements would then be $e_o^{(j)*}$ in the Type I decoder, or $i_o^{(j)*}$ in the Type II decoder. Thus for the case $R = k_o/n_o$, Theorem 15 would be unchanged, except that the last phrase would read "k_o threshold logical elements."

4.2 DECODING FOR TIME-VARIANT CHANNELS

We now consider the case in which the received bits do not all have the same error probability, but these probabilities are known at the receiver. In other words, the quantities $\Pr[e_u^{(j)}=1]$ are known, but may vary with u and j.

An example of such a channel would be one that adds a voltage pulse whose amplitude is Gaussianly distributed to the transmitted waveform, which is assumed to be either a pulse of $+\sqrt{S}$ volts for a "one" or a pulse of $-\sqrt{S}$ volts for a "zero." The receiver then uses the polarity of the received pulse to assign to the received bit the more probable value of the binary number transmitted. The amplitude of the received pulse can be used to compute the probability that this assignment was wrong. (See sec. 5.3 for more details concerning this channel.)

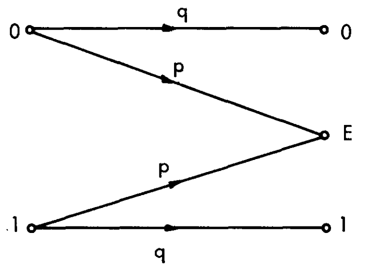


Fig. 13. Binary Erasure Channel.

Another example is the binary erasure channel shown in Fig. 13. When a "one" or "zero" is transmitted, it is received correctly with probability q and is erased with probability p. Then, at the receiver, any received "one" or "zero" has zero error probability, whereas any erased symbol can be assigned as a "zero" and has error probability one-half. (See sec. 5.2 for more details concerning this channel.)

We shall now show how APP decoding can be instrumented for the class of time-variant binary-output channels.

a. Weighting Factors

The application of APP decoding to time-variant channels requires the calculation of the weighting factors $\{w_i\}$ and the threshold T of Eqs. 109 and 111, and these quantities are now functions of time.

Consider the set $\{A_i\}$ of parity checks orthogonal on $e_o^{(1)}$. Let γ_j be the error probability of $e_{\alpha_j}^{(\beta_j)}$, the j^{th} noise bit, exclusive of $e_o^{(1)}$, which is checked by a particular A_i. Then, from Lemma 2, we have

$$p_i = \frac{1}{2}\left[1 - \prod_{j=1}^{n_i} (1-2\gamma_j)\right] = 1 - q_i . \tag{120}$$

Hence, it follows that

$$\frac{q_i}{p_i} = \frac{1 + \prod_{j=1}^{n_i} (1-2\gamma_j)}{1 - \prod_{j=1}^{n_i} (1-2\gamma_j)} . \tag{121}$$

It is more convenient to write (121) as follows. Let

$$c_{\alpha_j}^{(\beta_j)} = - \log_e(1-2\gamma_j) . \tag{122}$$

Then (122) may be written

$$\frac{q_i}{p_i} = \coth\left(\frac{1}{2} \sum_{j=1}^{n_i} c_{\alpha_j}^{(\beta_j)}\right), \tag{123}$$

where coth (x) = $(e^x+e^{-x})/(e^x-e^{-x})$ is the ordinary hyperbolic cotangent function. Then, with natural logarithms used, the weighting factor, w_i, of Eq. 109 becomes

$$w_i = 2 \log_e \left[\coth\left(\frac{1}{2}\sum_{j=1}^{n_i} c_{a_j}^{(\beta_j)}\right)\right]; \tag{124}$$

and also, since we always have $p_o = \Pr\left(e_o^{(1)}=1\right)$, we obtain

$$w_o = 2 \log_e (q_o/p_o) = 2 \log_e \left[\coth\left(\frac{1}{2} c_o^{(1)}\right)\right]. \tag{125}$$

b. Analog Circuit for Computing Weighting Factors

A circuit that calculates the $\{w_i\}$ and the threshold, based on (124) and (125), can

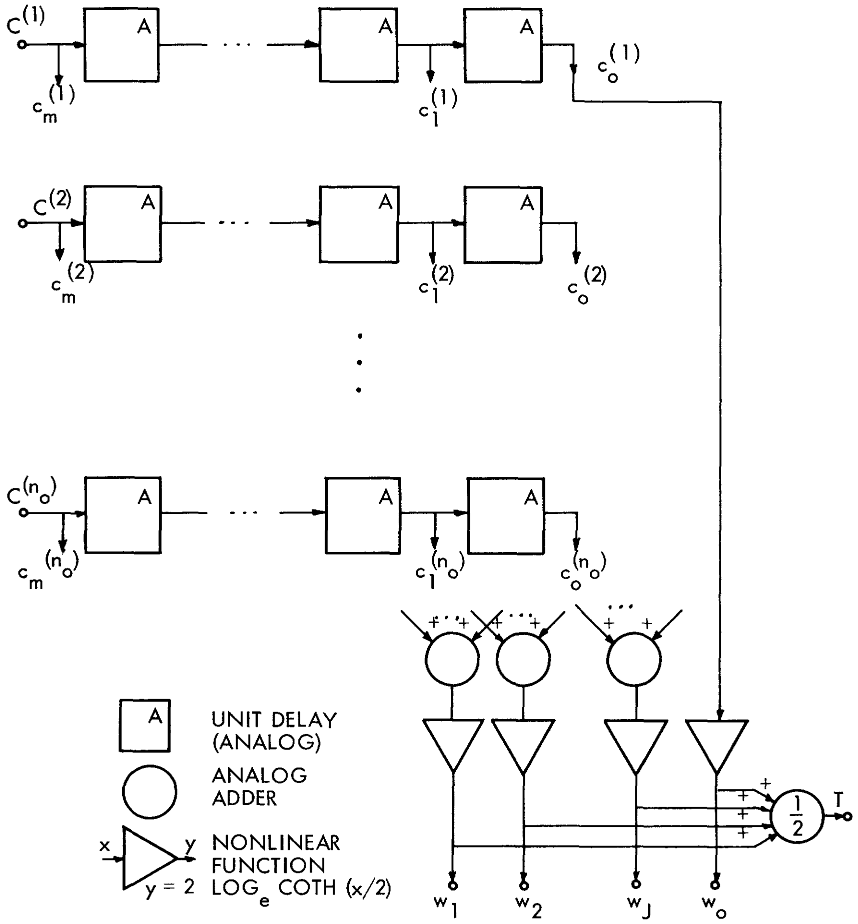


Fig. 14. Analog circuit for computing time-variant weighting factors and threshold.

now be obtained. Such a circuit is shown in Fig. 14.

The inputs to the circuit shown in Fig. 14 are assumed to be the n_o sequences

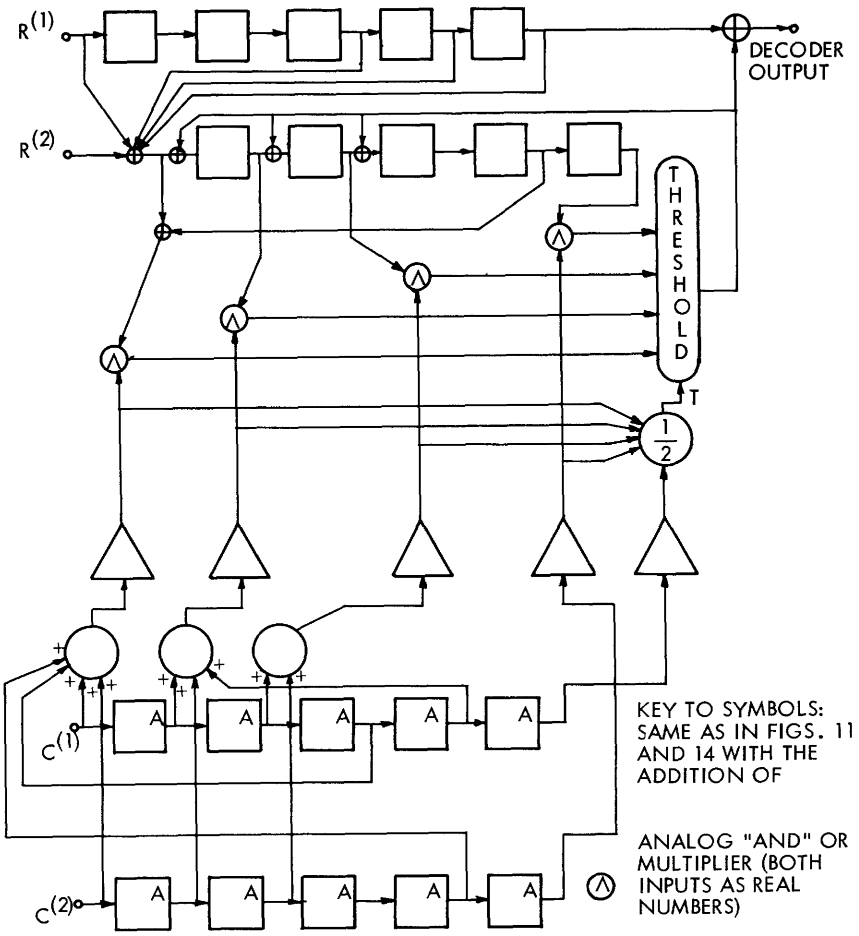
$$C^{(j)}(D) = c_o^{(j)} + c_1^{(j)}D + c_2^{(j)}D^2 + \dots \quad j = 1, 2, \dots n_o, \tag{126}$$

where $c_u^{(j)}$ is the input on line j at time u and has the value

$$c_u^{(j)} = -\log_e\left[1-2\Pr\left(e_u^{(j)}=1\right)\right]. \tag{127}$$

The weighting factors $\{w_i\}$ are computed (Fig. 14) as follows: The i^{th} analog adder, beneath the analog shift registers that store the $c_u^{(j)}$, has as inputs the set of $c_u^{(j)}$ corresponding to the noise bits $e_u^{(j)}$, exclusive of $e_o^{(1)}$, which are checked by A_i. The output of this analog adder is fed to a nonlinear device that has an output of $2\log_e\left[\coth\left(\frac{1}{2}x\right)\right]$ for an input of x. From (124), it follows that this is the correct weighting factor for parity check A_i. The threshold T is formed by taking half the sum of all the weighting factors as called for by Eq. 111.

Since the analog circuit of Fig. 14 computes the correct set of weights $\{w_i\}$ and the threshold T at every time instant, it can be combined with either the Type I or the Type II decoder to give a complete APP decoding circuit for a time-variant channel. In Fig. 15 we show the complete decoding circuit for the R = 1/2, J = 4,



$R^{(1)}$
DECODER OUTPUT
$R^{(2)}$
THRESHOLD
T
$\frac{1}{2}$
+
A
$C^{(1)}$
$C^{(2)}$
KEY TO SYMBOLS: SAME AS IN FIGS. 11 AND 14 WITH THE ADDITION OF
ANALOG "AND" OR MULTIPLIER (BOTH INPUTS AS REAL NUMBERS)

Fig. 15. Complete decoding circuit for the R = 1/2, m_E = 11 trial-and-error code.

trial-and-error code of Table I, in which we have coupled a Type I decoder to the analog circuit of Fig. 14.

4.3 SUMMARY

We have presented circuits that may be used to threshold-decode any binary convolutional code. The decoding circuits are quite simple, containing what is apparently the minimum possible number of storage devices and using a simple threshold logical element as the decision component. The decoding circuits make only one operation per time unit (during which n_o bits are received), and it is certainly feasible to construct such circuits for real-time decoding on channels for which the input bit rate is in the megacycle region. The fact that there is no variance in the decoding time per received bit is also desirable, in that it eliminates any queueing problems for the received bits.

All of these features of the decoding circuits are quite attractive to the communication engineer. The central question, however, remains unanswered. What error probabilities can be obtained at the receiver? Answers to this question will be given in Section V.

V. PERFORMANCE DATA FOR THRESHOLD DECODING OF CONVOLUTIONAL CODES

We shall conclude our treatment of convolutional codes with a presentation of the error probabilities that can be attained by using threshold decoding. We shall consider several communication channels and our interest will always be in the quantity $P_1(e)$. $P_1(e)$ was defined in section 2.5 as the average probability of incorrectly decoding the set of first information symbols. Since nearly all of the codes constructed in Section III had rates of the form $R = 1/n_o$, we shall restrict ourselves to such codes. In this case $P_1(e)$ becomes simply the average probability of incorrectly decoding $i_o^{(1)}$.

There are two main points concerning threshold decoding of convolutional codes that will be made here: on the one hand, its lack of generality; on the other hand, its excellent performance in particular cases. As a specific example, recall the binary symmetric channel of Fig. 1. The random coding bound of section 2.6 shows that when $R < C$, the average of $P_1(e)$ over the ensemble of convolutional codes approaches zero exponentially with the constraint length, n_A. In sharp contrast to this result, we shall show that when threshold decoding is used, $P_1(e)$ always exceeds some positive constant no matter how large n_A becomes (at least when the codes satisfy the corollary of Theorem 10). On the other hand, we shall see that, for codes of moderate length, the performance of threshold decoding compares favorably with the random-coding bound and to the error probabilities that can be obtained by using other available decoding systems.

5.1 THE BINARY SYMMETRIC CHANNEL

The major portion of this section will be devoted to the study of $P_1(e)$ for the binary symmetric channel of Fig. 1. We place our emphasis on this channel for two reasons. First, it is the channel that has been studied most thoroughly by communication theorists, and is now familiar to all communication engineers. Second, it is reasonable to infer that the performance of threshold decoding on this channel should be typical of its performance on a broader class of binary output channels.

a. A Bound on Error Probability

We wish to show that $P_1(e)$ cannot be made to vanish by using threshold decoding. The demonstration will be facilitated by making use of the log-likelihood-ratio, L, which we define as

$$L = \log_e \frac{\Pr\left[e_o^{(1)}=1 \,\middle|\, \{A_i\}\right]}{\Pr\left[e_o^{(1)}=0 \,\middle|\, \{A_i\}\right]}. \tag{128}$$

Using Eqs. 14 and 15, we can write Eq. 128 as

$$L = \sum_{i=1}^{J} \log_e \frac{\Pr\left[A_i \mid e_o^{(1)}=1\right]}{\Pr\left[A_i \mid e_o^{(1)}=0\right]} + \log_e \frac{\Pr\left[e_o^{(1)}=1\right]}{\Pr\left[e_o^{(1)}=0\right]}. \tag{129}$$

Finally, using Eqs. 18 and 19, we can reduce Eq. 129 to

$$L = \sum_{i=1}^{J} w_i A_i - T, \tag{130}$$

where the $\{A_i\}$ are treated as real numbers in this sum, and the weighting factors $\{w_i\}$ and the threshold T have the values that were assigned in Eqs. 109 and 111. Equation 130 expresses the interesting fact that the log-likelihood-ratio is just the difference between the weighted sum of the orthogonal parity checks and the threshold when APP decoding is used.

The log-likelihood-ratio is a useful quantity; when L is known, the error probability in decoding $i_o^{(1)}$, $P_1(e|L)$, can be determined from

$$P_1(e|L) = \frac{e^{-|L|}}{1 + e^{-|L|}}. \tag{131}$$

Equation 131 follows directly from (128) when it is noted that for $L > 0$ we have $P_1(e|L) = \Pr\left[e_o^{(1)}=0 \mid \{A_i\}\right]$, while for $L \leqq 0$ we have $P_1(e|L) = \Pr\left[e_o^{(1)}=1 \mid \{A_i\}\right]$.

We are now in a position to prove that when the codes are constructed to satisfy the corollary of Theorem 10, the probability of error cannot be made arbitrarily small if threshold decoding is used.

THEOREM 16: Given a binary convolutional code with $R = 1/n_o$ for which a set $\{A_i\}$ of $J = I(n_o-1)$ parity checks orthogonal on $e_o^{(1)}$ are formed. Suppose that $n_o - 1$ of these parity checks have $n_i = j$ for $j = 1, 2, \ldots I$; then for any J, $P_1(e)$ obtained by threshold decoding satisfies

$$P_1(e) > \frac{1}{2}\left(\frac{p_o}{q_o}\right)\frac{[2p_o+n_o-1]}{2p_o} \tag{132}$$

when the code is used on a binary symmetric channel with transition probability $p_o = 1 - q_o$.

PROOF 16: For threshold decoding, $P_1(e)$ is a minimum when APP decoding is used. Moreover, $P_1(e)$ must monotonically decrease as J increases, since APP decoding always makes best use of the information in the set $\{A_i\}$ and the set $\{A_i\}$, for a longer code constructed in accordance with the theorem always includes the set $\{A_i\}$ of each smaller code as a subset. Thus we need to show that as $J \to \infty$, $P_1(e)$ satisfies (132) when APP decoding is used. We begin by showing that $|L|$ is bounded.

Using Eq. 111, we may write Eq. 132 as

$$L = \sum_{i=1}^{J} w_i A_i - \frac{1}{2} \sum_{i=0}^{J} w_i \tag{133}$$

from which it follows that $|L|_{\max}$ is obtained when all of the $\{A_i\}$ are "zeros," that is, when all of the orthogonal parity checks are satisfied. Thus

$$|L|_{\max} = \frac{1}{2} \sum_{i=0}^{J} w_i = \sum_{i=0}^{J} \log_e \frac{q_i}{p_i}. \tag{134}$$

Substituting Eq. 112 in (134), we obtain

$$L_{\max} = (n_o - 1) \sum_{i=1}^{I} \log_e \frac{1 + (1-2p_o)^i}{1 - (1-2p_o)^i} + \log_e \frac{q_o}{p_o}. \tag{135}$$

Using the series expansion for the logarithm,

$$\log_e \frac{1+x}{1-x} = 2\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 +, \dots\right) \qquad 0 \leqq x < 1, \tag{136}$$

we can write (135) as

$$\lim_{J\to\infty} |L|_{\max} = \log_e \frac{q_o}{p_o} + (n_o - 1)(2) \sum_{i=1}^{\infty} \left[(1-2p_o)^i + \frac{1}{3}(1-2p_o)^{3i} + \dots \right]. \tag{137}$$

Since the series in brackets converges absolutely, the summation on each term may be carried out separately as a geometric series and gives

$$\lim_{J\to\infty} |L|_{\max} = \log_e \frac{q_o}{p_o} + (n_o - 1)(2) \left[\frac{(1-2p_o)}{1 - (1-2p_o)} + \frac{1}{3} \frac{(1-2p_o)^3}{1 - (1-2p_o)^3} + \dots \right]. \tag{138}$$

We can overbound the series on the right to give

$$\lim_{J\to\infty} L_{\max} < \log_e \frac{q_o}{p_o} + \frac{n_o - 1}{1 - (1-2p_o)} (2) \left[(1-2p_o) + \frac{1}{3}(1-2p_o)^3 + \dots \right]. \tag{139}$$

We recognize the series on the right as the expansion of

$$\log_e \frac{1 + (1-2p_o)}{1 - (1-2p_o)} = \log_e \frac{q_o}{p_o}, \tag{140}$$

and hence Eq. 139 becomes

$$\lim_{J\to\infty} |L|_{\max} < \frac{2p_o + n_o - 1}{2p_o} \log_e \frac{q_o}{p_o}. \tag{141}$$

From Eq. 131, it follows that $P_1(e|L)$ is a minimum when $|L|$ is a maximum and that this probability is greater than $\frac{1}{2}e^{-|L|_{\max}}$. Substituting (141), we find that $P_1(e|L)$ must satisfy

$$P_1(e|L) > \frac{1}{2} e^{-\left[\frac{2p_o+n_o-1}{2p_o}\log_e\frac{q_o}{p_o}\right]}. \qquad (142)$$

But since $P_1(e)$ is the average over all values of L of $P_1(e|L)$, $P_1(e)$ must certainly be greater than the minimum value of $P_1(e|L)$, and hence greater than the right-hand side of Eq. 142. This is the result stated in the theorem.

From Theorem 16 we see that $P_1(e)$ cannot be made to vanish by using threshold decoding with the class of codes that satisfy the corollary of Theorem 10. We saw in Theorem 12 that for $R = 1/2$ this class of codes is the best that can be constructed for threshold decoding. Thus for $R = 1/2$ it is impossible to make $P_1(e)$ arbitrarily small by threshold decoding. We can conjecture that it is also true that for other rates, $P_1(e)$ cannot be made arbitrarily small when threshold decoding is used, but we have not been able to prove this.

We have actually proved more than Theorem 16 states. Inequality (142) shows that for the class of codes considered in Theorem 16, there is never a decoding decision made which has error probability smaller than the right-hand side of (142) or (132). Even in the most favorable case (when the entire set of orthogonal parity checks is satisfied) the probability of a decoding error exceeds the bound on the right-hand side of (132). For this reason, (132) does not give a good lower bound on $P_1(e)$, but it does give a good bound on the minimum value of $P_1(e|L)$.

b. Data for the Trial-and-Error Codes

Having established the lack of generality of threshold decoding, we turn next to the task of computing $P_1(e)$ in particular cases. For this purpose, we choose the trial-and-error codes of Table III. These codes have the properties that n_A and n_E are nearly equal, and that n_E is never greater than the bound in Theorem 10.

Before expressing $P_1(e)$ in the form best suited for the binary symmetric channel, we shall give the general expression that applies to any binary output channel, whether time-variant or not. As usual, let $p_o = 1 - q_o = \Pr\left(e_o^{(1)}=1\right)$, p_o is then a random variable in the case of a time-variant channel. The general expression for $P_1(e)$ is

$$P_1(e) = \overline{q_o \Pr\left[\sum_{i=1}^{J} w_iA_i > T \,\middle|\, e_o^{(1)} = 0\right] + p_o \Pr\left[\sum_{i=1}^{J} w_iA_i \leqq T \,\middle|\, e_o^{(1)} = 1\right]} \qquad (143)$$

in which the bar indicates that the average is to be taken over all values of the weighting factors, $w_i = 2 \log (q_i/p_i)$, $i = 0, 1, 2, \ldots J$. Equation 143 states the obvious fact that an error is committed when either of the mutually exclusive events — that the threshold is exceeded when $e_o^{(1)} = 0$, or is not exceeded when $e_o^{(1)} = 1$ — occurs. But since the

$\{A_i\}$ conditioned on $e_o^{(1)}$ are the complements of the $\{A_i\}$ conditioned on $e_o^{(1)}=0$, we have

$$\Pr\left[\sum_{i=1}^{J} w_iA_i \leqq T \,\middle|\, e_o^{(1)} = 1\right] = \Pr\left[\sum_{i=1}^{J} w_iA_i \geqq \sum_{i=1}^{J} w_i - T \,\middle|\, e_o^{(1)} = 0\right]. \tag{144}$$

Hence, (143) may be rewritten as

$$\overline{P_1(e) = q_o\Pr\left[\sum_{i=1}^{J} w_iA_i > T \,\middle|\, e_o^{(1)} = 0\right] + p_o\Pr\left[\sum_{i=1}^{J} w_iA_i \geqq \sum_{i=1}^{J} w_i - T \,\middle|\, e_o^{(1)} = 0\right]} \tag{145}$$

and we see that only the probability distributions of the $\{A_i\}$ conditioned on $e_o^{(1)} = 0$ need be considered. For the binary symmetric channel, the bar in (145) can be ignored because the weighting factors are constants.

We have seen (section 1.1d) that the $\{A_i\}$ conditioned on $e_o^{(1)} = 0$ are a set of J statistically independent random variables for which $\Pr(A_i=1) = p_i = 1 - \Pr(A_i=0)$. We define the random variable λ_i as

$$\lambda_i = w_iA_i \qquad i = 1, 2, \ldots J, \tag{146}$$

where the A_i are treated as real numbers. Equation 145 may then be written

$$P_1(e) = q_o\Pr\left[\sum_{i=1}^{J} \lambda_i > T\right] + p_o\Pr\left[\sum_{i=1}^{J} \lambda_i \geqq \sum_{i=1}^{J} w_i - T\right], \tag{147}$$

where the λ_i are a set of J statistically independent random variables for which $\Pr(\lambda_i=w_i) = p_i$ and $\Pr(\lambda_i=0) = q_i$.

The determination of $P_1(e)$ from Eq. 147 reduces to the classical problem of calculating the probability that a sum of statistically independent random variables exceeds some fixed number. However, we have been unable to obtain a closed-form expression for $P_1(e)$, or even a good lower bound, for the following reasons. Since, by Eq. 112, for the binary symmetric channel

$$p_i = 1 - q_i = \frac{1}{2}\left[1 - (1-2p_o)^{n_i}\right], \tag{148}$$

the λ_i are not equidistributed in the general case. Moreover, since n_i grows in direct proportion to J (at least for codes that satisfy the corollary of Theorem 10), p_i approaches 1/2 as J increases; this means that the weighting factors approach zero in APP decoding. For large J, the distribution of the sum of the λ_i is determined almost entirely by the first several λ_i with small values of n_i. For these reasons, the standard procedures for handling sums of independent random variables, such as the Chernov bound, or the Central Limit Theorem, cannot be applied.

A numerical evaluation of Eq. 147 is facilitated by use of enumerating functions that were mentioned in section 4.1. The enumerating function, $g_i(s)$, for λ_i for the binary symmetric channel is

$$g_i(s) = p_i s^{w_i} + q_i. \tag{149}$$

Since the λ_i are statistically independent, the enumerating function, $g(s)$, of their sum is just the product of the $g_i(s)$, or

$$g(s) = \sum_{i=1}^{J} \left(p_i s^{w_i} + q_i\right). \tag{150}$$

Then, since each coefficient in an enumerating function is the probability that the random variable is equal to the exponent of s in that term, (147) can be stated

$$P_1(e) = q_o[\text{sum of coefficients in } g(s) \text{ of terms with exponents} > T] +$$
$$p_o\left[\text{sum of coefficients in } g(s) \text{ of terms with exponents} \geqq \sum_{i=1}^{J} w_i - T\right]. \tag{151}$$

Equation 151 was used as the basis for a machine calculation of $P_1(e)$ for the trial-and-error codes of Table III. For majority decoding, since $w_i = 1$ for all i, g(s) in (150) is an ordinary polynomial with J + 1 terms. The calculation of $P_1(e)$ is quite simple in this case. For APP decoding, the w_i can all be different, and g(s) then contains as many as 2^J terms, making machine calculation mandatory in all but the simplest cases.

In Figs. 16, 17, 18, and 19, we have plotted $P_1(e)$ versus n_E for the trial-and-error codes with rates 1/2, 1/3, 1/5, and 1/10, respectively. (These data were all obtained by machine calculation, and each figure required approximately three minutes of time on the IBM 7090 computer in the Computation Center, M.I.T.) Five different channels were used with the codes at each rate and were chosen so as to give a wide range of $P_1(e)$.

Several features of the data in Figs. 16-19 are immediately evident. First, $P_1(e)$ does not decrease exponentially with n_E for either majority decoding or APP decoding. $P_1(e)$ does decrease monotonically with n_E for APP decoding, for the reason that was stated at the beginning of the proof of Theorem 16. However, for majority decoding, $P_1(e)$ has a minimum for some value of n_E and increases thereafter, for the channels with large p_o. The reason for this is that p_i is almost 1/2 for the parity checks with large n_i in the longer codes, but these "bad" parity checks are being given the same weight in the decision on $e_o^{(1)}$ as the "good" parity checks with small n_i. Ultimately $P_1(e)$ would approach 1/2 as n_E increased indefinitely for majority decoding with any value of p_o.

As a first basis for evaluating the performance of the trial-and-error codes with

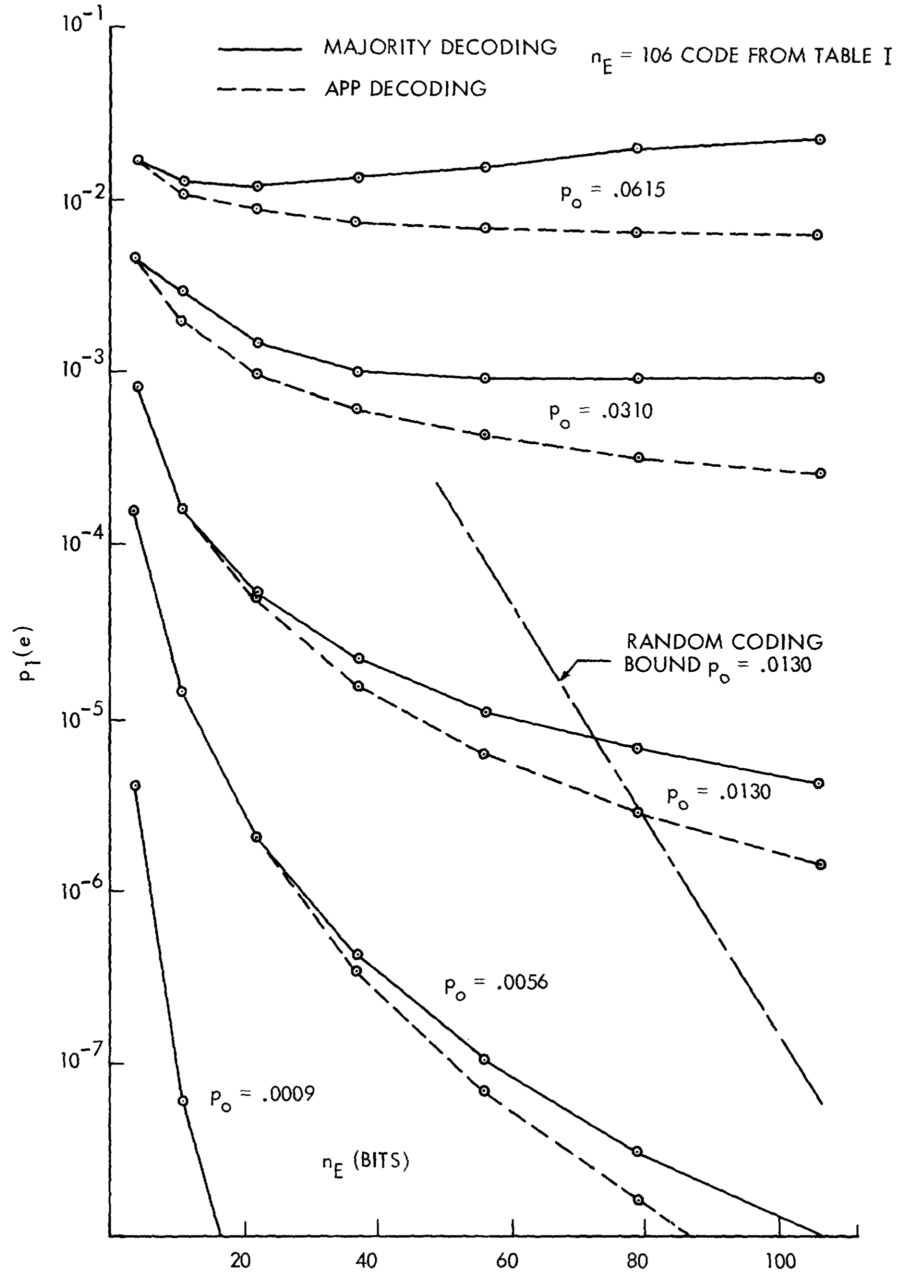


Fig. 16. Performance of R = 1/2 trial-and-error codes on the Binary Symmetric Channel.

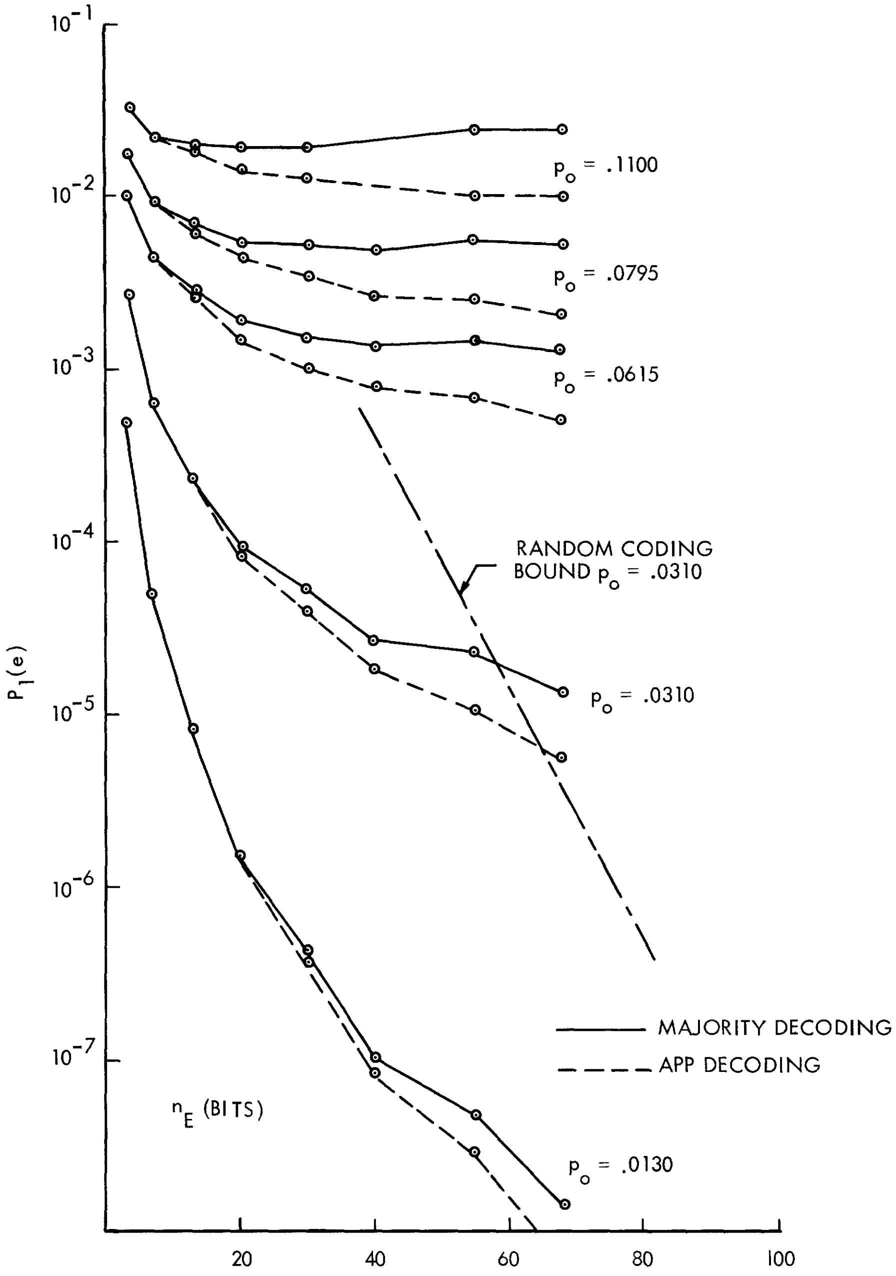


Fig. 17. Performance of R = 1/3 trial-and-error codes on the Binary Symmetric Channel.

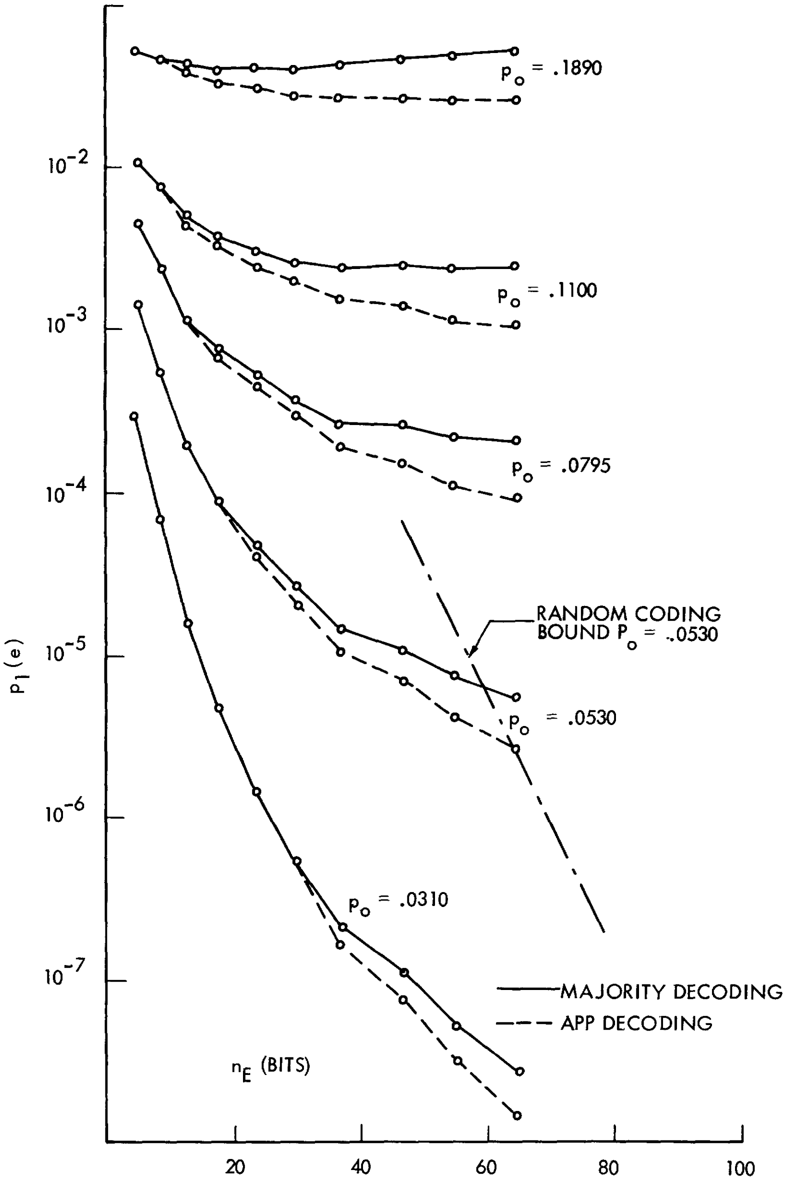


Fig. 18. Performance of R = 1/5 trial-and-error codes on the Binary Symmetric Channel.

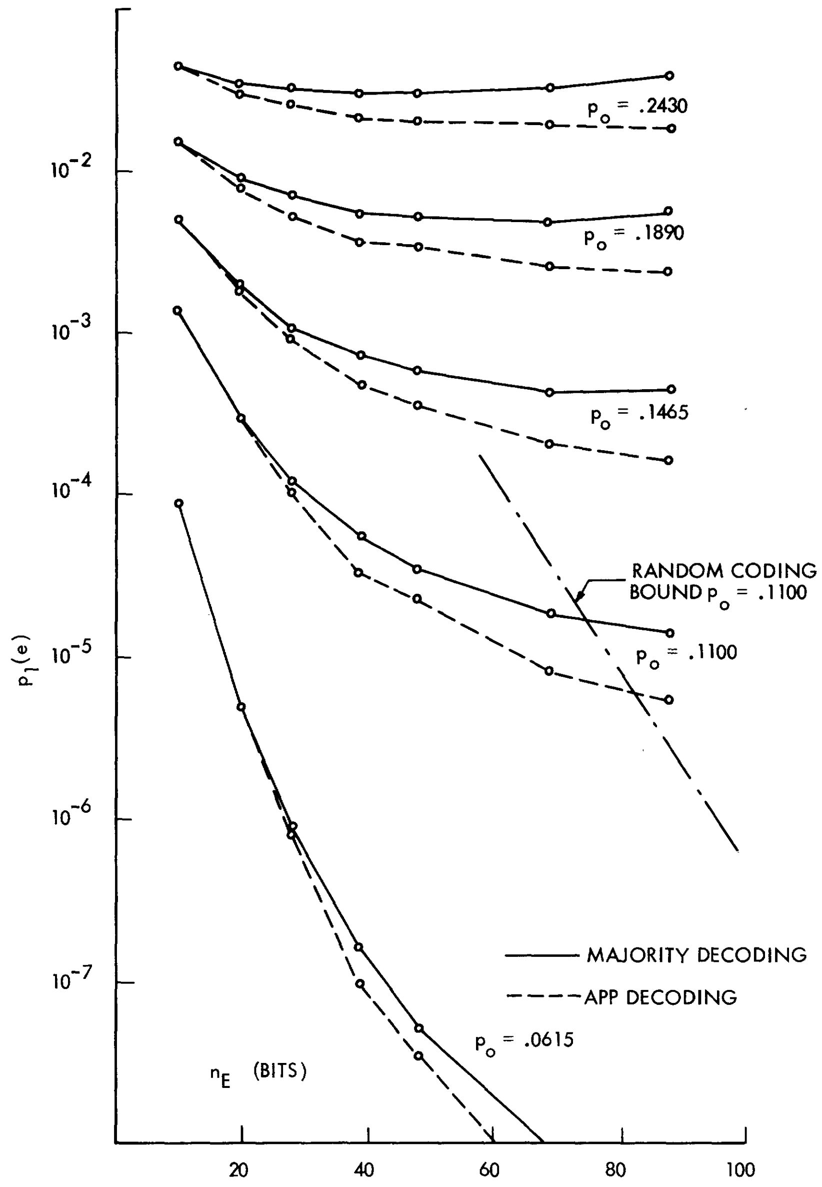


Fig. 19. Performance of R = 1/10 trial-and-error codes on the Binary Symmetric Channel.

threshold decoding used, we have plotted the upper bound on $P_1(e)$ as given by the random coding bound of Eq. B-10 for one channel in each of Figs. 16-19. (The rate in each case is slightly less than R_{crit} for the channel chosen.) We have used n_E rather than n_A when plotting the random coding bound, for the reason that n_E is the number of bits on which the decision on $e_o^{(1)}$ is made with threshold decoding used, whereas all n_A bits are used in maximum-likelihood decoding as assumed in deriving $P_1(e)$ in the random-coding bound. For all four rates, $P_1(e)$ for the trial-and-error codes equals $P_1(e)$ for the random-coding bound for $n_E \approx 70$. For the longer codes, $P_1(e)$ obtained by threshold decoding of the trial-and-error codes exceeds this upper bound to the average error probability of the ensemble of convolutional codes, with maximum-likelihood decoding assumed.

Since the practical considerations pointed out in section 1.1a generally make maximum-likelihood decoding infeasible, a more realistic criterion for evaluating the performance of the trial-and-error codes is the error probability that can be attained by using other practical encoding and decoding procedures. For this comparison, at rates 1/2, 1/3, and 1/5 we have chosen the nearest Bose-Chaudhuri codes,[16] for which the Peterson decoding algorithm[28] can be used to correct any error pattern of weight (d-1)/2 or less, where d is the minimum distance. At rate 1/10, we have assumed the existence of block codes for which d is equal to the average weight of a nonzero code word. In computing P(e), the block probability of error, we assumed that in every case an error pattern was correctable if and only if it had weight (d-1)/2, or less. Finally, since at n = 63 there were no Bose-Chaudhuri codes close to the desired rates, we shortened the code with rate just exceeding the desired rate by dropping information symbols until the shortened code had rate equal to or less than the desired rate. In Table V, we list the complete set of block codes that were used for comparison with the trial-and-error codes.

P(e), the block probability of error, and P(e)/k, where k is the number of information symbols, is given in Table VI for the codes of Table V and the trial-and-error codes of Table II. The same channel was used at each rate as was used for the

Table V. Block codes used for comparison with trial-and-error codes.

n	k	d	R	Type of Code
31	16	7	.517	Bose-Chaudhuri
54	27	11	.500	"shortened" Bose-Chaudhuri (63, 36) code
31	11	11	.355	Bose-Chaudhuri
58	19	15	.328	"shortened" Bose-Chaudhuri (63, 24) code
31	6	15	.194	Bose-Chaudhuri
58	11	23	.190	"shortened" Bose-Chaudhuri (63, 16) code
30	3	17	.100	average distance code
60	6	31	.100	average distance code

random-coding bound in Figs. 16-19, and n was equated with n_E.

Although no method of comparing error probabilities for block decoding and convolutional decoding is entirely satisfactory, the best basis seems to be comparison between $P_1(e)$ and $P(e)/k$. The reasoning behind this choice is the following: Let $P_k(e)$ be the probability of making any error in decoding the first k received information bits. Then if $P_k(e) = P(e)$, the average number of information bits decoded correctly before the first error is made will be approximately the same for convolutional decoding and block decoding. On the other hand, $P_k(e)$ is conservatively bounded as

$$P_k(e) \leqq k\, P_1(e) \tag{152}$$

in which we overbound the probability of a union of events by the sum of the individual probabilities. Thus a comparison between P(e) and k $P_1(e)$, or (that which is the same) between P(e)/k and $P_1(e)$, seems a reasonable choice for comparing block decoding and convolutional decoding.

From Table VI it can be seen that threshold decoding of the trial-and-error codes compares quite favorably with the block decoding used for comparison. There is little difference in the performances at the higher rates. The marked superiority of threshold decoding at R = 1/10 stems from the fact that at such a low rate it is important to correct a sizeable percentage of error patterns with weight greater than (d-1)/2. Even majority decoding will correctly decode $e_o^{(1)}$ for many such error patterns. For example, in the R = 1/1 0, J = 26, d = 27 trial-and-error code, a decoding error is made when $e_o^{(1)} = 0$ only when 14 or more of the orthogonal parity checks are "ones." Thus with a pattern of 14 errors among the 48 bits in the orthogonal parity checks, an error is made only in the unlikely event that each bit in error is checked by a different one of the 26 orthogonal parity checks $\left(\text{with } e_o^{(1)}=0 \text{ assumed}\right)$; and if two of the bits in error are in the same parity check, no decoding error will be made unless the error pattern has weight at least 16.

c. Tolerances for the Weighting Factors and Threshold

The performance of threshold decoding with the trial-and-error codes of Table II used is quite creditable, and the simple decoding circuits for implementing threshold decoding may render it of considerable practical value for codes of these lengths. But, whereas for the decoding circuits and the calculations for $P_1(e)$ it is assumed that it is possible to set the weighting factors and the threshold at precisely their correct values, it is of considerable engineering import to know what effect inaccuracies in these quantities will have on decoding performance.

It is easy to see that tolerance requirements are not strict. From Eq. 130, it follows that in the critical instance in which the weighted sum of the orthogonal parity checks is approximately equal to the threshold, L is approximately zero and the error probability of the decision is approximately 1/2 according to Eq. 131. Thus it makes little difference whether the decoder assigns $e_o^{(1)}$ equal to "one" or "zero." In Fig. 20a

Table VI. Comparison of performance of the trial-and-error convolutional codes with the block codes of Table V on the binary symmetric channel.

RATE	n_E or n	BLOCK CODES		CONVOLUTIONAL CODES
		P(e)	P(e)/k	$P_1(e)$
1/2	31	$6.8\text{x}10^{-4}$	$4.3\text{x}10^{-5}$	$2.4\text{x}10^{-5}$
	54	$7.3\text{x}10^{-5}$	$2.7\text{x}10^{-6}$	$6.6\text{x}10^{-6}$
1/3	31	$3.4\text{x}10^{-4}$	$3.0\text{x}10^{-5}$	$3.8\text{x}10^{-5}$
	58	$1.3\text{x}10^{-5}$	$7.0\text{x}10^{-7}$	$9.3\text{x}10^{-6}$
1/5	31	$1.6\text{x}10^{-4}$	$2.7\text{x}10^{-5}$	$1.9\text{x}10^{-5}$
	58	$4.1\text{x}10^{-5}$	$3.7\text{x}10^{-6}$	$3.4\text{x}10^{-6}$
1/10	30	$3.7\text{x}10^{-3}$	$1.2\text{x}10^{-3}$	$7.7\text{x}10^{-5}$
	60	$6.0\text{x}10^{-4}$	$1.0\text{x}10^{-4}$	$1.3\text{x}10^{-5}$

$P_1(e)$ for the convolutional codes was found by extrapolating the data given in Figs. 16-19 for $P_1(e)$, with the use of APP decoding assumed.

The transition probabilities that were used are:

$p_o = .0130$ for $R = 1/2$

$p_o =$ 0310 for $R = 1/3$

$p_o = .0530$ for $R = 1/5$

$p_o = .1100$ for $R = 1/10$.

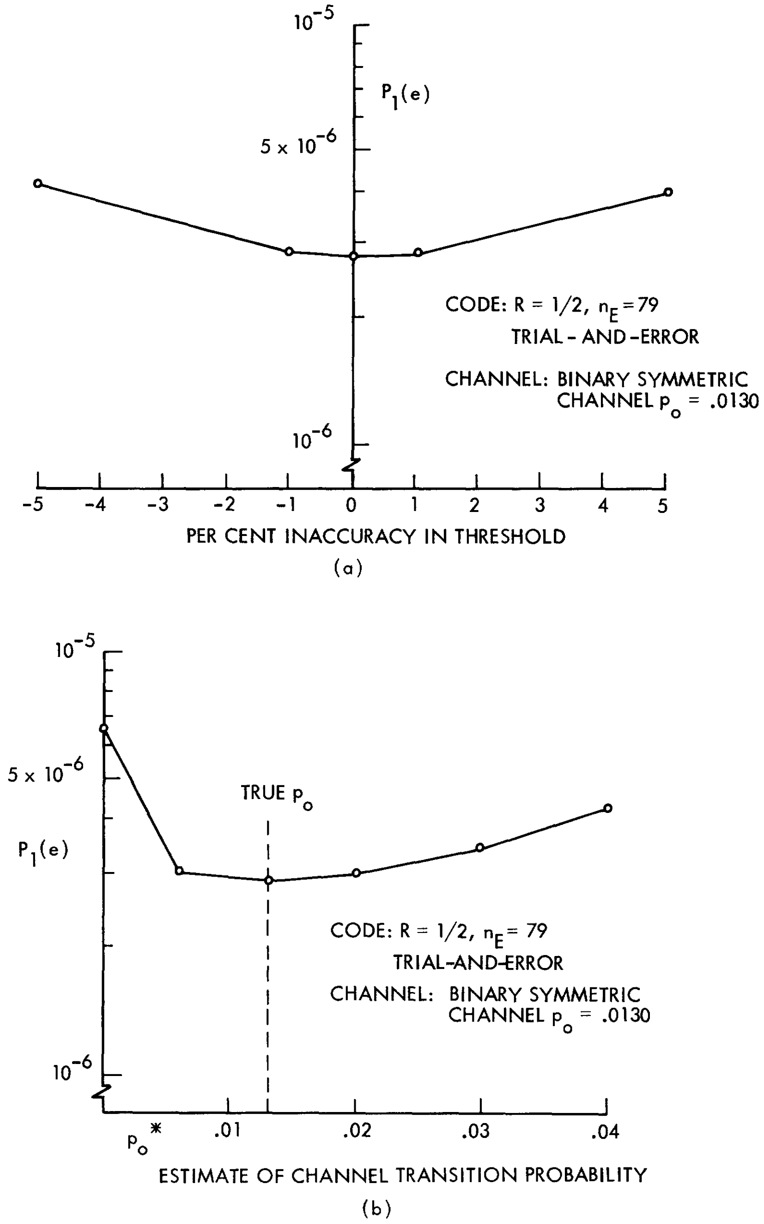


Fig. 20. (a) Effect of inaccuracies in the threshold setting on the probability of error.
(b) Effect of inaccuracies in estimation of the channel on the probability of error.

we give numerical results, for a typical case, which illustrate the effect of inaccuracies in the threshold setting. Even a 5 per cent maladjustment does not severely degrade performance. Minor inaccuracies in the weighting factors have the same effect.

A more interesting question concerns the effect of errors in estimation of the channel. According to Eq. 112, the weighting factors, and the threshold also, are set according to the estimate of p_o, which we shall denote p_o^*. Fortunately, a very accurate estimate is not required. This could be inferred from Figs. 15-19, in which it is clear that even the equal weighting of all parity checks prescribed by majority decoding does not severely degrade performance in most instances. In Fig. 20b, we give numerical results for a typical case which show the minor effect on $P_1(e)$ of moderate inaccuracies in the channel estimation.

Finally, we observe that majority decoding can be thought of as the limiting case of APP decoding when $p_o^* \to 0$. For, consider two parity checks, one of size $n_i = a$, the other of size $n_i = b$. The ratio of the weighting factors w_a and w_b is, according to Eq. 112,

$$\frac{w_a}{w_b} = \frac{\log \dfrac{1+\left(1-2p_o^*\right)^a}{1-\left(1-2p_o^*\right)^a}}{\log \dfrac{1+\left(1-2p_o^*\right)^b}{1-\left(1-2p_o^*\right)^b}}. \tag{153}$$

As $p_o^* \to 0$, both weighting factors approach infinity, but their ratio approaches a limit that can be evaluated by two successive applications of l'Hopital's rule and is found to be

$$\lim_{p_o^* \to 0} \frac{w_a}{w_b} = 1 \tag{154}$$

independently of a and b. In this case all parity checks are weighted the same, which is the distinguishing feature of majority decoding. Thus one expects the performance of APP and majority decoding to be nearly the same on channels with small p_o. From Figs. 16-19, this is indeed seen to be the case.

d. Modifications of Threshold Decoding

There are two basic ways in which the threshold decoding algorithms can be modified to improve decoding reliability on the binary symmetric channel (or any other channel) equipped with a feedback provision. (The basic concept of this section — improving reliability by decoding only the most likely error patterns and requesting retransmission in all other cases — was first suggested by Wozencraft and Horstein.[29])

The first method is to equip the threshold device with an alarm zone of width 2Δ centered about the threshold. That is, whenever $\left|\sum_{i=1}^{J} w_iA_i - T\right| < \Delta$, an alarm is sounded and a repeat of the data is requested. From Eq. 131, it follows that for APP decoding the error-alarm method is equivalent to making a decoding decision when and only when the error probability is less than $e^{-\Delta}/(1+e^{-\Delta})$.

The alarm zone is useful with majority decoding also. Specifically, let M be any integer that is such that J - M is an even number. Then $e_o^{(1)}$ will be correctly decoded whenever there are $\frac{J-M}{2}$, or fewer, errors among the n_E symbols checked by the $\{A_i\}$, and no incorrect decision on $e_o^{(1)}$ will be made when no more than $\frac{J+M}{2}$ errors occur among these symbols, when the decoding rules are: Choose $e_o^{(1)} = 1$ if $\frac{J+M}{2}$ or more of the $\{A_i\}$ have value "one"; choose $e_o^{(1)} = 0$ when $\frac{J-M}{2}$ or less of the $\{A_i\}$ have value "one"; and otherwise signal the presence of a detected error. The proof of this statement is a trivial modification of the proof of Theorem 1.

The second method of improving reliability when a feedback channel is available will be called random supplementation of the convolutional code. This method is designed for use in conjunction with the error-count technique described in section 3.2. The idea behind this method is to cause a decoding error to propagate so that it may subsequently be detected by the error-count procedure, and a repeat then requested. This can be accomplished by adding terms with degree greater than m to the code-generating polynomials of Eq. 65, the coefficients in the extra terms being chosen independently as "one" or "zero" with probability one-half. (Since n_A is increased in this process, the encoder and decoder complexities are increased in the same proportion.) Suppose that L additional terms are added to each code-generating polynomial. From Fig. 11 it can be seen that when a decoding error is made, the effect of the random supplementation is to add a burst of $L(n_o-1)$ random bits into the parity-check shift registers. As these bits advance into that section of the parity-check shift register which is being used for decoding, that is, into the last m stages of shift register in each chain, the probability is increased that the error will propagate. (A similar analysis applies to the Type II decoder shown in Fig. 12.) By choosing L sufficiently large, the probability of an undetected error can be made as small as desired. However, since the number of bits that must be repeated grows as L, the amount of time that is spent in requesting repeats increases. (We shall not consider here such important questions as how the data for repeats is stored, how many bits are repeated, etc. The reader is referred to the paper of Wozencraft and Horstein[29] for a full treatment of these matters. Our purpose here is merely to point out those features for implementation of feedback strategies which are peculiar to threshold decoding.)

Finally, in order to implement the error-count procedure, it is necessary to have a means for counting all of the errors corrected in the received sequences. The output of the threshold device in the Type I decoder is a "one" each time an error is corrected in the information sequence. The number of errors corrected in the parity sequences can be obtained by modifying the Type I decoder as shown in Fig. 21. The code-generating

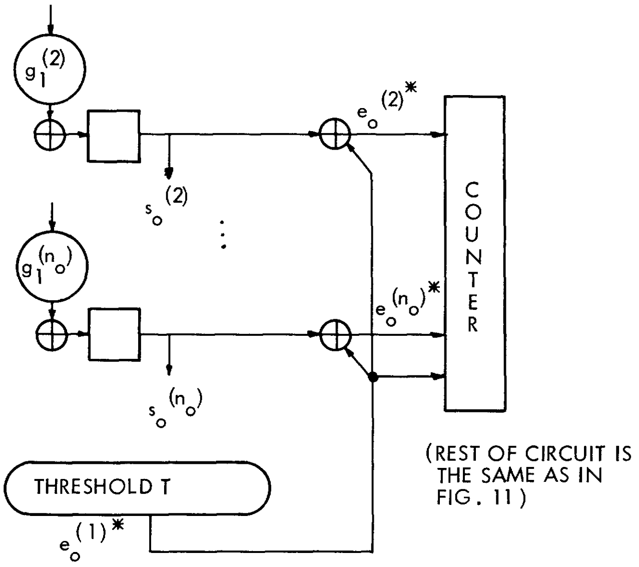


Fig. 21. Circuit for counting all errors in received sequences.

polynomials are always selected with $g_o^{(j)} = 1$, $j = 2, 3, \ldots n_o$; otherwise, there is an idle position in the first code word, that is, a position that contains only zeros for all first code words. Thus from Eq. 70, it follows that

$$s_o^{(j)} = e_o^{(1)} + e_o^{(j)} \qquad j = 2, 3, \ldots n_o. \tag{155}$$

Hence, $e_o^{(j)*}$, the decoded estimate of $e_o^{(j)}$, can be obtained by adding $e_o^{(1)*}$ modulo-two to $s_o^{(j)}$. This is done in Fig. 21; the adders in this circuit then have a "one" output each time an error is corrected in the corresponding parity sequences.

5.2 THE BINARY ERASURE CHANNEL

A very powerful decoding method for convolutional codes has been developed by Epstein for this channel.[30] Epstein's method results in an exponential decrease in error probability at any rate less than channel capacity and with a finite average number of computations per decoded bit independent of code length.

In this section we shall very briefly treat APP decoding for the binary erasure channel of Fig. 13. This channel has capacity C = q and is characterized by the fact that a received bit may be ambiguous, but can never be in error.[31] In section 4.2, we saw that this channel may be reduced to a binary output channel by assigning the value "zero" to all erased bits and giving these bits error probability one-half. The expression for $P_1(e)$ may be written directly as

$$P_1(e) = \frac{1}{2}p \Pr\left[\begin{array}{l}\text{Each } A_i \text{ checks at least one bit that has been}\\ \text{erased, not including } e_o^{(1)}\end{array}\right]. \tag{156}$$

Equation 156 can be seen to hold as follows: $e_o^{(1)}$ can be determined immediately from any parity check in which no other erased bit appears; on the other hand, if there is another erased bit, such a parity check gives no information about $e_o^{(1)}$. Thus an error will be made with probability one-half when, and only when, $e_o^{(1)}$ corresponds to an erased bit and every one of the parity checks orthogonal on $e_o^{(1)}$ checks one or more erased bits in addition to $e_o^{(1)}$. Equation 156 may be written in the form

$$P_1(e) = \frac{1}{2} p \prod_{i=1}^{J} \left(1-q^{n_i}\right) \tag{157}$$

in which we use the fact that no bit, except $e_o^{(1)}$, appears in more than one of the orthogonal parity checks.

Using (157), we have computed $P_1(e)$ for two binary erasure channels in Fig. 22. Since the general features are the same as those for the binary symmetric channel, we

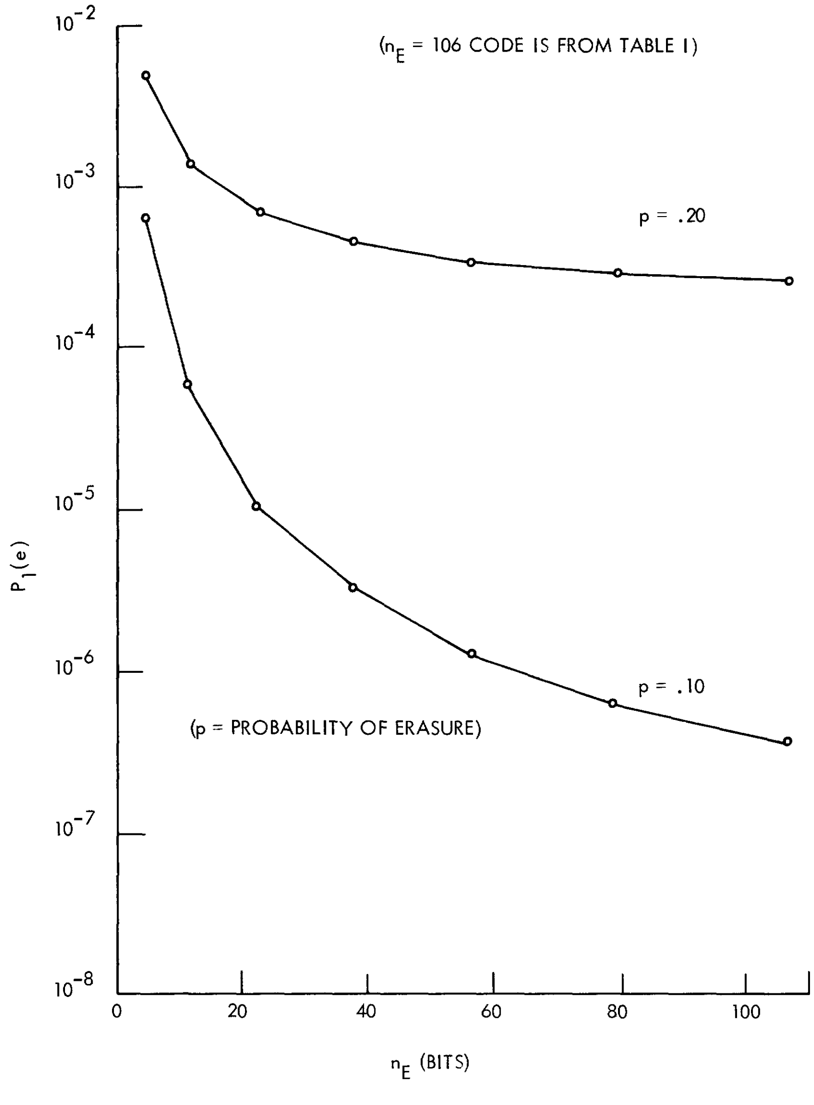


Fig. 22. Performance of R = 1/2 trial-and-error codes on the Binary Erasure Channel.

shall not discuss them further.

As was the case for the binary symmetric channel, $P_1(e)$ cannot be made arbitrarily small when the codes satisfy the corollary of Theorem 10. Specifically, we have the following theorem.

THEOREM 17: Given a binary convolutional code with $R = 1/n_o$ for which a set $\{A_i\}$ of $J = I(n_o-1)$ parity checks orthogonal on $e_o^{(1)}$ are formed in such a manner that $n_o - 1$ of these checks have $n_i = j$ for $j = 1, 2, \ldots I$, for any J, $P_1(e)$ obtained by threshold decoding satisfies

$$P_1(e) \geq \frac{1}{4} p \prod_{i=1}^{M} \left(1-q^i\right)^{n_o-1}, \tag{158}$$

where

$$M = \left\lceil [\log_2 p - \log_2 (n_o-1) - 1]/\log_2 q \right\rceil \tag{159}$$

when the code is used on a binary erasure channel with erasure probability $p = 1 - q$.

PROOF 17: For the codes in Theorem 17, Eq. 157 becomes

$$P_1(e) = \frac{1}{2} p \prod_{j=1}^{I} \left(1-q^i\right)^{n_o-1}, \tag{160}$$

and this clearly decreases monotonically with I. We also have

$$\prod_{i=M}^{\infty} \left(1-q^i\right)^{n_o-1} \geq 1 - (n_o-1) \sum_{i=M}^{\infty} q^i = 1 - (n_o-1)(q^M)/(1-q). \tag{161}$$

When M is chosen to satisfy (159), $(n_o-1)(q^M)/(1-q) \leq \frac{1}{2}$, and (161) becomes

$$\prod_{i=M}^{\infty} \left(1-q^i\right)^{n_o-1} \geq \frac{1}{2}. \tag{162}$$

From (162), it follows that the product of all of the terms in (160) cannot be smaller than one-half the product taken up to $i = M$, and this is the result stated in the theorem.

5.3 THE GAUSSIAN CHANNEL

The last channel that we shall study here is the Gaussian channel that was described briefly in section 4.2. The transmitted signal is a pulse of $+\sqrt{S}$ volts for a "one," and a pulse of $-\sqrt{S}$ volts for a "zero." The received pulse differs from the transmitted pulse by the addition of a noise pulse that is gaussianly distributed with zero mean and variance N. Now we shall show how the probability inputs, $c_u^{(j)}$, to the analog circuit of Fig. 14 can be obtained for this channel. We shall also give some indication of the improvement in decoding performance that can be realized by use of the time-variant weighting factors, as opposed to constant weighting factors for this channel.

From section 4.1a, we recall that the probability inputs to the circuit for calculating the weighting factors were defined as

$$c_u^{(j)} = -\log_e \left[1-2\Pr(e_u^{(j)} = 1)\right] \tag{163}$$

for all u and j. These probability inputs can be obtained as follows. When a pulse of v volts is received, the corresponding received bit is assigned as a "one" or "zero" according as v is positive or negative. The log-likelihood-ratio, L_i, for this decision is

$$L_i = \log_e \frac{\Pr\,[e_u^{(j)}=0]}{\Pr\,[e_u^{(j)}=1]} = \log_e \frac{\frac{1}{\sqrt{2\pi N}}\, e^{-\frac{(\sqrt{S}-|v|)^2}{2N}}}{\frac{1}{\sqrt{2\pi N}}\, e^{-\frac{(\sqrt{S}+|v|)^2}{2N}}}. \tag{164}$$

and this reduces to

$$L_i = 2\frac{\sqrt{S}}{N}\,|v|. \tag{165}$$

Equation 165 states that L_i is directly proportional to the magnitude of the received voltage. Using Eq. 131, we obtain

$$\Pr\,[e_u^{(j)}=1] = \frac{e^{-L_i}}{1+e^{-L_i}}. \tag{166}$$

Using (165) and (166), we can rewrite (163) as

$$c_u^{(j)} = -\log_e \frac{1-e^{-L_i}}{1+e^{-L_i}} = \log_e \coth\left[\frac{\sqrt{S}}{N}|v|\right]. \tag{167}$$

From Eq. 167, it follows that the $c_u^{(j)}$ can be obtained by passing the received pulses through the circuit of Fig. 23. The nonlinear device in this circuit is exactly the same type as those in the analog circuit of Fig. 14.

In principle, the computation of $P_1(e)$ for the Gaussian channel can be carried out by

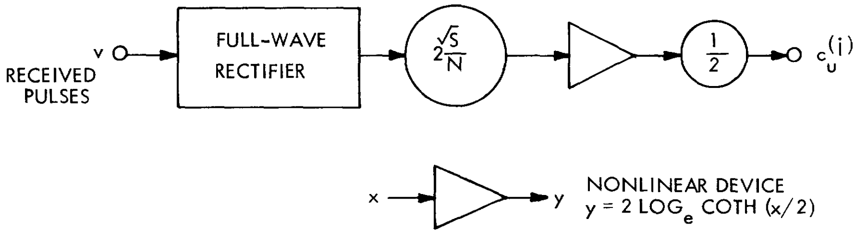


Fig. 23. Circuit for computing $c_u^{(j)}$ for a Gaussian channel.

using Eq. 145. However, since the average must be taken over the distribution of the weighting factors, and since the weighting factors are nonlinear functions of the gaussianly distributed received voltages, the expression is not analytically tractable in the general case. There is one simple case, however, for which $P_1(e)$ can be found directly, and which will serve to illustrate how much advantage is gained by using the time-variant weighting factors. This simple case is the code for which each of the $n_o - 1$ parity symbols is a repeat of the single information symbol in the constraint span; such a block code can be considered as a degenerate case of a convolutional code. (A thorough treatment of this channel including computations similar to Eqs. 172-174 is available in a paper by Bloom et al.[32])

In this case, it is convenient to use the $\{B_i\}$ as defined in section 4.1b, for we have

$$B_i = r_o^{(i)} \qquad i = 1,2,\ldots n_o \tag{168}$$

and

$$P_i = \Pr\,[e_o^{(i)}=1] \qquad i = 1,2,\ldots n_o. \tag{169}$$

Using Eqs. 164, we find that the weighting factors are

$$w_i = 2 \log_e \frac{q_i}{p_i} = 4 \frac{\sqrt{S}}{N} |v_i| \qquad i = 1,2,\ldots n_o, \tag{170}$$

where v_i is the received pulse for bit $r_o^{(i)}$. Then since $r_o^{(i)}$ is assigned as a "one" when v_i is positive, or as a "zero" when v_i is negative, the APP decoding rule of section 4.1b becomes: Choose $i_o^{(1)} = 1$ if, and only if,

$$\sum_{i=1}^{n_o} v_i > 0. \tag{171}$$

Without loss of generality, we may assume that a "one" is transmitted. Then the sum on the left side of (171) is a Gaussian random variable with mean $n_o\sqrt{S}$ and variance n_oN, since it is the sum of n_o statistically independent Gaussian random variables with mean $\sqrt{S}$ and variance N. Since an error is made whenever this sum is negative, we have

$$P_1(e) = \frac{1}{\sqrt{2\pi n_o N}} \int_{-\infty}^{0} \exp\left(-\frac{(X-n_oS)^2}{2n_oN}\right) dX \tag{172}$$

which reduces to

$$P_1(e) = \frac{1}{2\pi} \int_{-\infty}^{\sqrt{(n_oS)/N}} \exp\left(-\frac{X^2}{2}\right) dX. \tag{173}$$

For example, if n = 10, and if $P_1(e) = 1.38 \times 10^{-3}$, then Eq. 173 gives $\sqrt{S/N} = 0.98$. From Fig. 19, we find that the same code ($n_E=10$) gives the same $P_1(e)$ when used on a binary symmetric channel with transition probability $p_o = 0.110$. This binary symmetric channel can be considered as derived from a Gaussian channel in a manner such that only the polarity of the output pulse can be observed, and such that

$$0.110 = \frac{1}{\sqrt{2\pi N}} \int_{-\infty}^{0} \exp\left(-\frac{(X-\sqrt{S})^2}{2N}\right) dX. \tag{174}$$

The solution of Eq. 174 gives $\sqrt{S/N} = 1.24$. Thus, we can obtain a signal-to-noise advantage of $20 \log_{10} (1.24/0.98) = 2.3$ db as compared with the use of constant weighting factors in the example considered.

5.4 SUMMARY

The most striking result in this section is a negative one, namely that $P_1(e)$ is bounded away from zero when threshold decoding is employed with convolutional codes on the binary symmetric channel or the binary erasure channel, at least when the codes satisfy Theorem 10 and its corollary. Since it is impossible to construct better $R = \frac{1}{2}$ codes for threshold decoding than those that satisfy Theorem 10 and its corollary, it follows that this result is rigorously true for any code at rate $R = \frac{1}{2}$, and it probably applies also to all other rates. Not withstanding this failure of threshold decoding to exploit the full potential of convolutional codes, it is clear from Table VI (and Figs. 16-19) that the error probability attainable by threshold decoding of convolutional codes compares favorably with that of other known decoding systems for codes of moderate length. The simplicity of instrumentation for threshold decoding may render it of considerable practical value for codes of such length.

It was also shown here how modification, such as an alarm zone around the threshold, can be made to improve the reliability of threshold decoding. Moreover, from an engineering point of view, the important fact that tolerance requirements in the threshold element are lenient, was demonstrated. This result applies to the accuracy of setting of the threshold and the weighting factors, and to the accuracy of prior estimation of the channel.

VI. THRESHOLD DECODING OF BLOCK CODES

6.1 INTRODUCTION

In the preceding sections, we have seen how the concept of threshold decoding can be applied to the class of convolutional codes. We shall now study its applicability to the more familiar class of block linear codes. This class of codes is distinguished from the class of convolutional codes by the fact that each set of k information symbols is independently encoded into n symbols for transmission. Hereafter, we use the notation, an (n, k) code, to mean a linear block code in systematic form (systematic form implies, as explained in section 1.1, that the first k transmitted symbols are identical to the information symbols).

According to Eq. 2, the parity symbols, $t_{k+1}, t_{k+2}, \dots t_n$, are determined from the information symbols, $t_1, t_2, \dots t_k$, by the following set of linear equations

$$t_j = \sum_{i=1}^{k} c_{ji} t_i \qquad j = k+1, k+2, \dots n \tag{175}$$

which may be represented in matrix form as

$$\begin{bmatrix} t_{k+1} \\ t_{k+2} \\ \vdots \\ t_n \end{bmatrix} = \begin{bmatrix} c_{k+1,1} & \cdots & c_{k+1,k} \\ \vdots & & \vdots \\ c_{n,1} & \cdots & c_{n,k} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_k \end{bmatrix} = [P] \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_k \end{bmatrix} \tag{176}$$

where [P] is the (n−k)xk matrix of coefficients $[c_{ji}]$. It now follows from Eq. 5 that the parity checks are given by the matrix equation

$$\begin{bmatrix} s_{k+1} \\ s_{k+2} \\ \vdots \\ s_n \end{bmatrix} = [P:-I] \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \tag{177}$$

where [I] is the identity matrix of dimension n−k.

The matrix

$$H = [P:-I] \tag{178}$$

is called the parity-check matrix of the code. From (177), it can be seen that each row of H gives the coefficients of the noise digits that appear in one of the parity checks. We call $e_1, e_2, \ldots e_k$ the information noise digits, and we call $e_{k+1}, e_{k+2}, \ldots e_n$ the parity noise digits. Then from (177) it can be seen that the rows of $[P]$ give the coefficients of the information noise digits appearing in the parity checks; moreover, each parity noise digit is checked by one and only one parity check. It should be clear, by reference to the material in section 3.2, that the task of forming a set of parity checks orthogonal on e_j reduces to finding disjoint sets of rows of $[P]$ for which some linear combination of the rows in each such set has a "one" in the j^{th} position, but no other position has a nonzero entry in more than one of these linear combinations of twos.

For example, consider the binary (6,3) code for which the parity check matrix is

$$H = [P:I] = \begin{bmatrix} 0 & 1 & 1 & \vdots & 1 & 0 & 0 \\ 1 & 0 & 1 & \vdots & 0 & 1 & 0 \\ 1 & 1 & 0 & \vdots & 0 & 0 & 1 \end{bmatrix}. \tag{179}$$

The first row of H gives the coefficients of the noise bits in parity check $s_{k+1} = s_4$. Thus s_4 checks e_2, e_3, and e_4. Similarly, s_5 checks e_1, e_3, and e_5; and s_6 checks e_1, e_2, and e_6. We observe that s_5 and s_6 are a set of two parity checks orthogonal on e_1. Similarly, s_4 and s_6 are orthogonal on e_2; and s_4 and s_5 are orthogonal on e_3. From Theorem 1, it now follows that e_1, e_2, and e_3 can all be correctly determined by majority decoding, provided that there is no more than a single error in the six received bits in a block. The transmitted information bits can then be found by adding e_1, e_2, and e_3 modulo-two to the received information bits.

Before making a systematic investigation of orthogonal parity checks for block codes, we shall first generalize some of the definitions given in Section III to the case of block codes.

The minimum distance, d, of a block (n,k) code is customarily defined as the smallest number of positions in which two code words differ. Since the set of code words form a group, d is also the weight of the nonzero code word with the fewest nonzero positions.

We say that a block (n,k) code can be completely orthogonalized if d-1 parity checks orthogonal on each e_j can be formed, for $j = 1, 2, \ldots k$, that is, on each of the information noise digits.

Only the information noise digits need to be found by the decoding process, since the entire set of transmitted information digits can be found from them by using the relation $t_j = r_j - e_j$, $j = 1, 2, \ldots k$. (The transmitted parity digits can, of course, be reconstructed from the information digits by using Eq. 175 if these parity digits are required at the receiver.) From Theorem 1, it follows that each of the k information noise digits will be correctly decoded, provided that there are no more than $\left\lfloor\frac{d-1}{2}\right\rfloor$ or fewer errors in a received block, if majority decoding is used with a code that can be completely orthogonalized. Thus the entire set of information noise digits will be correctly decoded in this case. In other words, any error pattern that is guaranteed correctable by the minimum distance of the code is correctable by majority decoding when the code can be completely orthogonalized. Many error patterns of greater weight may also be corrected, as will be shown by a specific example in section 6.2b.

Given a set $\{A_i\}$ of J parity checks orthogonal on some e_j, we say that the size, n_i, of each such parity check is the number of noise digits, exclusive of e_j, checked by that parity check.

6.2 MAXIMAL-LENGTH CODES

In section 3.7, it was shown that the class of uniform binary convolutional codes could be completely orthogonalized. These codes had the property that their minimum distance coincided with their average distance. It seems natural then to begin our investigation of block codes with that class of codes for which the minimum distance between code words coincides with the average distance, namely the maximal-length codes. These codes derive their name from the fact that the code word can be considered as the first q^k-1 output symbols of a maximal-length k-stage shift register having the k information symbols as initial conditions,[33] where all the symbols are elements of GF(q).

One may consider a block (n,k) code to be the special case of a convolutional code for which $n_A = n_o = n$ and $k_o = k$. It then follows from Eq. 53 that the average distance between code words in a block (n,k) code is

$$d_{avg} = \frac{q^{k-1}}{q^k - 1}(q-1)\, n. \tag{180}$$

For the maximal-length codes, since $n = q^k - 1$ and $d = d_{avg}$, we have, from Eq. 180, $d = q^{k-1}(q-1)$.

The parity-check matrix, $[P:-I]$, for the maximal-length codes is such that P has as rows the set of all $q^k - k - 1$ nonzero k-tuples, excluding the k that are all zero except for a single "one" in some position. This fact can be seen in the following way: For any nonzero initial conditions, the first q^k-1 output digits of a maximal-length shift register contain each nonzero k-tuple in some set of k successive positions (with cycling from the last digit to the first allowed). Each shift of such an output is also a possible output and hence also a code word in a maximal-length code. The set of code words can then be considered the row space of the matrix $[G]$, where the first row of G is any nonzero output sequence, and the remaining k - 1 rows are each the cyclic shift of the preceding row. Every set of k successive positions in the first row then is the same as some column of G, hence the q^k-1 columns of G must be the set of all nonzero k-tuples. This is still true after elementary row operations are performed on G to put G in the form $[I:P^*]$ (this can be seen by noting that the set of k-tuples, formed from the set of all nonzero k-tuples by leaving all positions unchanged, except the first which is the sum of the first and second positions in the previous set, is again the set of all k-tuples). Thus P^* must have as columns the set of all nonzero k-tuples, excluding the k unit vectors. But if the code is the row space of $[I:P^*]$, then the parity-check matrix is $[P^{*t}:-I]$, where t indicates the transposed matrix.[34] Thus P^{*t} must have as rows the set of all nonzero k-tuples excluding the unit vectors, and this is the property that was to be shown.

For example, the binary (7,3) maximal-length code has the parity-check matrix

$$H = \left[\begin{array}{ccc:c} 1 & 1 & 0 & \\ 0 & 1 & 1 & \\ 1 & 1 & 1 & I \\ 1 & 0 & 1 & \end{array}\right] \qquad (181)$$

and this is of the form $H = [P:I]$, where the rows of P are all the nonzero 3-tuples excluding the unit vectors.

a. Number of Orthogonal Parity Checks

We shall now determine the maximum number of parity checks orthogonal on e_1 that can be formed from the parity-check matrix of a maximal-length code. Let the $q^k - k - 1$ rows of $[P]$ be decomposed into two sets of rows, S_1 and S_2, such that

(i) S_1 contains all q-2 rows of weight one with the first position nonzero, and all k-1 rows of weight two with a "one" in the first position; and otherwise all zero except for a single position that contains a "minus one."

(ii) S_2 contains the remaining $q^k - q - 2k + 2$ rows of $[P]$.

The set of rows in S_1 correspond to a set of parity checks orthogonal on e_1 because, except for e_1, no information noise digit is checked by more than one of the parity checks. Moreover, for each row in S_2 that has first digit β, there is a unique row in S_2 that has first digit $(1-\beta)$ and for which the digits in the remaining positions are the negative of those in the former row. The sum of these rows then corresponds to a parity check that checks e_1 and no other information noise digit. All of the parity checks formed in this way can be joined to those corresponding to the rows in S_1 and the entire set is orthogonal on e_1. The number, J, of parity checks orthogonal on e_1 that are formed in this way is

$$J = \#(S_1) + \frac{1}{2}\#(S_2), \qquad (182)$$

where #(S) is the number of elements in the set S. Equation 182 gives

$$J = \frac{1}{2} q(q^{k-1}+1) - 2. \tag{183}$$

It can be seen that this is the maximum number of parity checks orthogonal on e_1 which can be formed because

(i) no row of [P] can be used more than once in forming parity checks orthogonal on e_1,

(ii) the rows in S_1 are the largest number of single rows of [P] that can be used in a set of parity checks orthogonal on e_1,

(iii) the remaining rows of [P] must be combined at least in pairs to produce additional parity checks orthogonal on e_1.

From the symmetry of P, it follows that the entire process can be iterated for $j = 2, 3, \ldots k$ in order to obtain the same number of parity checks orthogonal on each e_j.

b. Complete Orthogonalization

Since the maximum number, J, of parity checks orthogonal on any information symbol in a maximal-length code is given by (183), it follows that the code can be completely orthogonalized if and only if $J = d - 1$, where $d = (q-1)\,q^{k-1}$. Using (183), the difference between d-1 and J is found to be

$$(d-1) - J = \frac{1}{2}(q-2)(q^{k-1}-1). \tag{184}$$

From Eq. 184 we are able to make the conclusions stated below.

THEOREM 18: The binary maximal-length codes can be completely orthogonalized.

PROOF 18: Substitution of $q = 2$ in Eq. 184 gives $d - 1 = J$.

THEOREM 19: The nonbinary maximal-length codes can be completely orthogonalized when, and only when, $k = 1$, that is, when there is a single information symbol.

PROOF 19: For $q \neq 2$, the right-hand side of (184) will be a positive number except when the last factor vanishes. This factor vanishes if and only if $k = 1$. Thus only in this case does $d - 1 = J$.

Theorem 18 establishes the important result that the class of binary maximal-length codes can be completely orthogonalized. These are $(2^k-1,k)$ codes with minimum distance 2^{k-1}. For large k, the rate, $k/n = k/(2^k-1)$, is very small.

The fact that threshold decoding is not limited to correcting only the error patterns of weight $\left\lfloor\frac{d-1}{2}\right\rfloor$, or less, is especially important for such low-rate codes. As a specific example, consider the (1023,10) maximal-length code with $d = 512$. Suppose that this code is used on a binary symmetric channel with transition probability $p_0 = 0.25$. The average number of errors in a received block is then $(1023)p_0$, or approximately 256. On the other hand, $\left\lfloor\frac{d-1}{2}\right\rfloor = 255$. Thus, the probability of error would be approximately 1/2 for a decoding algorithm that was capable of correcting only errors of weight $\left\lfloor\frac{d-1}{2}\right\rfloor$, or less. Suppose now that majority decoding is used to determine each of the 10 information noise bits from the set of $d - 1 = 511$ parity checks orthogonal on each such bit.

The total probability of error is approximately 5×10^{-7}. The reason for this drastic reduction can be seen. Suppose that $e_1 = 0$, then e_1 will be incorrectly decoded only when more than 256 of the 511 parity checks orthogonal on e_1 are "ones." Since each of these parity checks includes two noise bits exclusive of e_1, the probability that such a check is "one" can be found from Lemma 3 to be 0.375. The probability that more than 256 of these 511 parity checks are "ones" can then be calculated to be less than 3.1×10^{-8}. In a similar manner, the probability that e_1 is incorrectly decoded when $e_1 = 1$ is found to be less than 1.0×10^{-7}. The average error probability in decoding e_1 is thus less than $(.750)(3.1\times10^{-8}) + (.250)(10^{-7}) = 4.9 \times 10^{-8}$. The probability of any decoding error among all 10 information noise bits is certainly less than 10 times the probability of incorrectly decoding e_1, or less than 4.9×10^{-7}.

Theorem 19 emphasizes the remark made in section 1.2 to the effect that there are difficulties in applying threshold decoding to nonbinary codes in an efficient manner. For large k, it follows from (183) that $J \approx \frac{1}{2} n$, independent of q, whereas from Eq. 180 we have $d \approx \frac{q-1}{q} n$. This indicates the fundamental difficulty that we have encountered in trying to orthogonalize nonbinary codes, namely that the number of orthogonal parity checks that can be formed is about the same as can be formed for a binary code with the same n and k. This means that full advantage is not being taken of the higher order alphabet of the nonbinary codes. On the other hand, although complete orthogonalization is not possible for the nonbinary maximal-length codes, the simplicity of the threshold-decoding algorithms might make them reasonable choices for decoding these codes and, perhaps, other nonbinary codes also; but we shall not pursue the matter further. Hereafter, we shall consider only binary codes.

6.3 THRESHOLD-DECODING CIRCUITS FOR CYCLIC CODES

The maximal-length codes are cyclic codes, that is, a code for which a cyclic shift ($t_j \to t_{j-1}$ with $t_1 \to t_n$) of any cyclic code is again a code word. Peterson has shown that any (n,k) cyclic code can be encoded by a linear sequential network containing either k or n-k stages of shift register.[25] The cyclic structure of these codes also makes possible simple threshold decoding circuits as we shall now show.

A cyclic code can be completely orthogonalized if, and only if, d-1 parity checks orthogonal on e_1 can be formed. This follows from the fact that the parity checks on e_2 must be able to be put into the same form as the parity checks on e_1, but with all indices increased by one cyclically. If J parity checks orthogonal on e_1 can be formed, then J parity checks orthogonal on e_2 can be formed, and conversely. A similar argument applies for e_3, e_4, ... e_k. This feature of cyclic codes makes possible the use of threshold decoders very similar to the Types I and II decoders for convolutional codes. We call these circuits the cyclic Types I and II decoders and we shall illustrate their construction by using, as an example, the (7,3) maximal-length code having the parity check matrix of Eq. 181.

The cyclic-Type I decoder for this code is given in Fig. 24. The received bits are

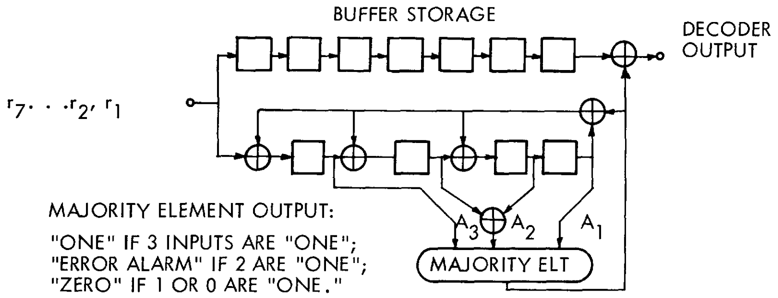

BUFFER STORAGE
DECODER OUTPUT
r7 · · ·r2, r1
A3
A2
A1
MAJORITY ELT
MAJORITY ELEMENT OUTPUT:
"ONE" IF 3 INPUTS ARE "ONE";
"ERROR ALARM" IF 2 ARE "ONE";
"ZERO" IF 1 OR 0 ARE "ONE."

Fig. 24. Cyclic Type I decoder for (7,3) maximal-length code.

fed simultaneously into n stages of buffer storage and into an (n-k)-stage encoding circuit. After n shifts, the (n-k)-stage register contains the modulo-two sum of the received parity bits and the encoded received information bits. By Theorem 9, this sum is just the set of parity checks. The parity checks are then combined to produce the set of parity checks orthogonal on e_1, after which the set of orthogonal parity checks is weighted and compared with the threshold. The output of the threshold element is e_1^*, the decoded estimate of e_1. This is added to r_1 which is just emerging from the buffer to give t_1^*, the decoded estimate of t_1. Since the code is cyclic, the circuit as shown (without the dotted connection) will continue to operate correctly at successive time instants, the output after the next shift being t_2^*, and so forth.

The cyclic Type I circuit is of the type originally proposed by Meggitt[35] for decoding an arbitrary cyclic code. Meggitt specified the main decision element in the circuit only as a combinatorial element that has a "one" output for all parity-check patterns in the (n-k)-stage register corresponding to error patterns with a "one" in the first position. In general, there seems to be no way to construct a simple combinatorial element for this circuit, the difficulty may be seen as follows. There is a single error pattern of weight one with a "one" in first position, but there are n-1 of weight two, (n-1)(n-2) of weight three, etc., with "one" in the first position. Each such error pattern must give a distinct parity-check pattern if it is correctable. Thus, if the decoder is to be able to correct any combination of T or fewer errors, the combinatorial element must be able to recognize approximately n^{T-1} parity-check patterns. It is not practical to attempt a synthesis of such a combinatorial element, by the standard minimization techniques used in logical design, when T is greater than approximately 2. The only hope is to find some specific structural properties of the code that will suggest a practical form for the combinatorial element.

When the cyclic code can be completely orthogonalized, then threshold decoding suggests the form of the combinatorial element, namely a threshold logical element with the necessary modulo-two adders needed to form the orthogonal parity checks. In such a case, Meggitt's theoretically general circuit becomes a practical decoding circuit.

Meggitt included the connections shown dotted in Fig. 24 with his general cyclic decoder. This causes the contents of the (n-k)-stage shift register to contain the parity checks corresponding to the received symbols as altered by the decoding process. After decoding is complete, this register will contain only "zeros" if the output is a valid code word. The presence of any "ones" indicates that an uncorrectable error has been detected. Finally, it should be clear from Fig. 24 that the buffer storage can be reduced to k stages when, as is usual, the decoded parity symbols are not of interest. When this is done, the cyclic Type I decoder contains a total of n stages of shift register.

The cyclic Type II decoder for the same (7,3) maximal-length code is shown in Fig. 25. This circuit uses the threshold-decoding algorithms in the form specified in section 4.1b. In this case the set $\{B_i\}$ of equations used in the decoding decision are given by

$$B_o = r_1 \tag{185}$$

and

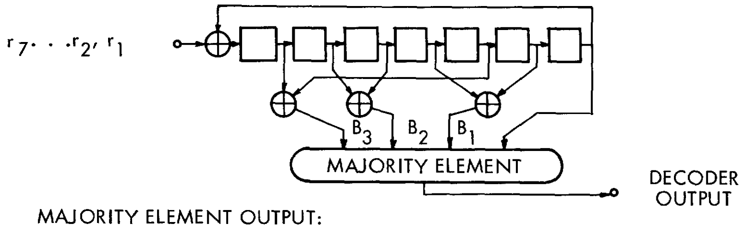
$$B_i = (\text{sum of the received bits, exclusive of } r_1\text{, whose noise bits are checked by the } i^{th} \text{ parity check orthogonal on } e_1). \tag{186}$$

The manner in which these equations are weighted and compared with the threshold is exactly the same as described in section 4.1b.

The cyclic Type II decoder is the essence of simplicity. The received bits are first stored in an n-stage shift register. The $\{B_i\}$ are formed by adding the appropriate received bits. The output of the threshold element is i_1^*, the decoded estimate of i_1. Since the code is cyclic, the same circuit gives i_2^* as its output after the shift register is cycled once, and so on.

The manner in which the weighting factors and the threshold are calculated is exactly the same for the cyclic Types I and II decoders as for the convolutional Types I and II decoders, and no further explanation will be given here. The set of weights and the threshold are constants for the binary symmetric channel when APP decoding is used, and are always constants for majority decoding.

The main features of the cyclic Types I and II decoders can be summarized in the following theorem.



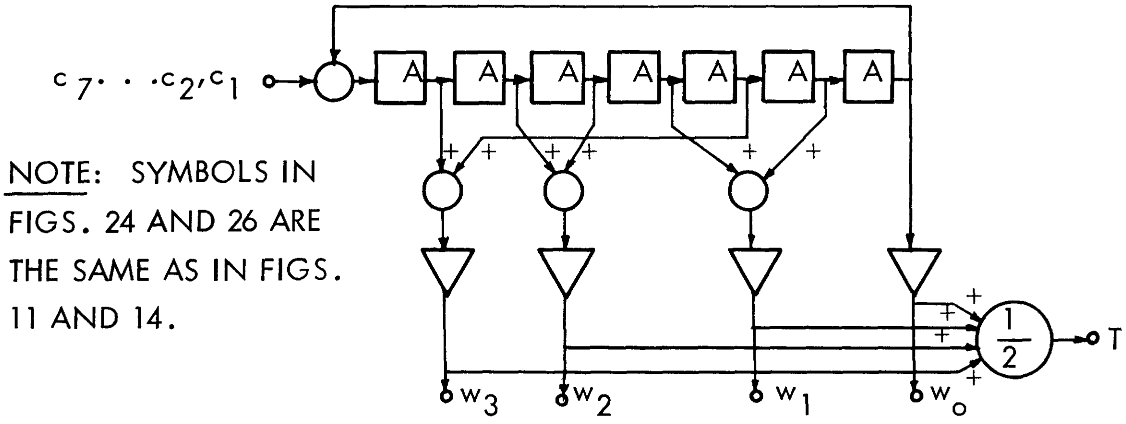
MAJORITY ELEMENT OUTPUT:
"ONE" IF 3 OR 4 INPUTS ARE "ONE";
"ERROR ALARM" IF 2 ARE "ONE";
"ZERO" IF 1 OR 0 ARE "ONE."

Fig. 25. Cyclic Type II decoder for (7,3) maximal-length code.

THEOREM 20: Given a cyclic (n,k) binary code that can be completely orthogonalized, any combination of $\left\lfloor\frac{d-1}{2}\right\rfloor$ errors, or fewer, in a block can be corrected by a decoding circuit containing n stages of shift register and one threshold logical element.

In Theorem 20, we assume that majority decoding is used on the set of orthogonal parity checks that can be formed on e_1. We have underlined part of the statement of this theorem to emphasize the fact that the code must have a structure that permits the formation of d-1 parity checks on e_1 if the cyclic Types I and II decoders are to be efficient and practical decoding circuits. There is no reason to suspect that such a structure is a general property of cyclic codes.

When APP decoding is used to determine e_1 from the set of parity checks orthogonal on e_1, it becomes necessary to use weighting factors on each of the input lines to the threshold element of the cyclic Types I and II decoders. For the binary symmetric channel, these weighting factors are constants as explained in section 4.1. For a time-variant binary output channel, the weighting factors and the threshold can be computed by an analog circuit similar to that of Fig. 14. Such a circuit for the (7,3) maximal-length code is shown in Fig. 26. The inputs to this circuit are $c_1, c_2, \ldots c_7$ where $c_i = -\log_e[1-2\Pr(e_i=1)]$. The operation of this circuit is so similar to that of the circuit in Fig. 14 that no further explanation is required.



c7· · ·c2,c1
A
A
A
A
A
A
A
+
+
+
+
+
+
+
+
+
+
1/2
T
w3
w2
w1
wo
NOTE: SYMBOLS IN FIGS. 24 AND 26 ARE THE SAME AS IN FIGS. 11 AND 14.

Fig. 26. Analog circuit for computing weighting factors and threshold.

6.4 BOSE-CHAUDHURI (15,7) CODE

Except for the maximal-length codes, we have been unable to find classes of good block (n,k) codes that can be completely orthogonalized. However, in certain isolated cases we have found good codes that can be completely orthogonalized. One such code is the (6,3) code of Eq. 179. A more interesting case is the (15,7), d = 5, binary Bose-Chaudhuri code.[16] This code has the parity-check matrix H = [P:I] for which the rows of P are

$$\begin{array}{lccccccc l}
3\rightarrow & 1 & \boxed{1} & 0 & \boxed{1} & 0 & 0 & 0 & \\
 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & \rceil \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & \dashv \\
 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \vert \\
3\rightarrow & 1 & 1 & 0 & 1 & 1 & 1 & 0 & \rfloor \\
 & 0 & 1 & 1 & 0 & \boxed{1} & 1 & 1 & \rceil \\
3\rightarrow & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \rfloor \\
3\rightarrow & 1 & 0 & \boxed{1} & 0 & 0 & 0 & \boxed{1} &
\end{array} \qquad (187)$$

and d - 1 = 4 parity checks orthogonal on e_1 can be formed as shown. (The same convention for indicating the formation of the orthogonal parity checks is used here as was described in sections 3.5 and 3.6.) Since this is a cyclic code, it follows that the code can be completely orthogonalized and can be decoded by either the cyclic Type I or cyclic Type II decoders.

Aside from the fact that the (15,7) Bose-Chaudhuri code is a useful code, this example is interesting in that this code is one for which Peterson[36] attempted a computer minimization of the logical expression, defining the truth table for the combinatorial element in a Meggitt decoder, and was unable to find a simple logical circuit from this approach. The orthogonal parity-check approach leads naturally to a simple logical circuit for the combinatorial element in the Meggitt decoder for this code, namely a majority logic element and two modulo-two adders.

6.5 A SUFFICIENT CONDITION FOR COMPLETE ORTHOGONALIZATION

We have not been able to formulate a good set of conditions sufficient to establish whether or not a binary (n,k) code can be completely orthogonalized. Our only result in this direction is the following theorem whose proof is given in Appendix D.

THEOREM 21: Any binary block (n,k) code with $k \leqq 3$ can be completely orthogonalized.

Theorem 21 establishes the maximum value of k for which a block (n,k) code can always be completely orthogonalized. The simplest nontrivial k = 4 code is the Hamming (7,4) d = 3 code for which the parity-check matrix is

$$H = \left[\begin{array}{cccc c c}
1 & 0 & 1 & 1 & \vdots & \\
1 & 1 & 1 & 0 & \vdots & I \\
0 & 1 & 1 & 1 & \vdots &
\end{array}\right] \qquad (188)$$

and it is readily verified that there is no way to form d-1 parity checks orthogonal on e_1. Since this is a cyclic code, it follows also that there is no way to form d-1 parity checks orthogonal on any of the information noise bits.

6.6 SUMMARY

We have seen how threshold decoding can be applied to block (n,k)-codes. For cyclic codes, we have shown that the threshold-decoding circuits can be made quite simple. However, for the most part, it appears that efficient use of threshold decoding for block (n,k) codes is confined to binary low-rate codes such as the maximal-length codes. This consideration is reinforced by Theorem 21 which stated that complete orthogonalization of a binary code is always possible only when the number of information symbols is 3 or less. In order to by-pass this limitation on the class of codes to which threshold decoding can be applied in an efficient manner, we shall, in Section VII, generalize the manner of application of the threshold-decoding algorithms in such a way that they can be applied to an enlarged class of block codes in an efficient manner.

VII. GENERALIZED THRESHOLD DECODING FOR BLOCK CODES

In Section VI we were able to establish that the threshold-decoding algorithms could be applied efficiently to the binary maximal-length codes, a class of low-rate codes. The effectiveness of the basic algorithms appears to be limited to such low-rate codes, in the sense that complete orthogonalization of a code is usually not possible for high-rate codes. We now extend the procedure for forming orthogonal parity checks to enlarge the class of codes for which threshold decoding can be efficient. We limit our treatment to binary codes. (It is not yet clear to what extent the generalized procedure is applicable to nonbinary codes.)

Before describing the generalized procedure, we shall illustrate its use by an example. Consider the Hamming (7, 4), d=3, code for which the parity-check matrix, H = [P:I] is given in Eq. 188. We have seen that this code could not be completely orthogonalized. For this code, the rows of P correspond to parity checks s_5, s_6, and s_7, and are

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ 2 \rightarrow & \boxed{1} & 1 & 1 & 0 \\ 2 \rightarrow & 0 & 1 & 1 & \boxed{1} \end{array}.$$

(As in Section III, we use a numbered arrow to indicate an orthogonal parity check and its size, and we place a box about each nonzero coefficient of an information noise bit, other than those in the sum, appearing in the parity checks orthogonal on that sum.) As shown, s_6 and s_7 form a set of $d-1=2$ parity checks orthogonal on the sum, e_2+e_3. Provided that no more than a single error is in the received block, majority decoding can be used to determine this sum correctly. Let us call this decoded estimate $(e_2+e_3)^*$. Similarly, s_5 and s_7 form $d-1=2$ parity checks orthogonal on e_3+e_4, and we can find $(e_3+e_4)^*$ by majority decoding.

Now consider modifying the original parity checks to form a set s_5' and s_6' given by

$$s_5' = s_5 + (e_3+e_4)^*$$

and

$$s_6' = s_6 + (e_2+e_3)^* . \tag{189}$$

From Eq. 189, it can be seen that if the sums were correctly decoded, then s_5' and s_6' are parity checks corresponding to the parity-check matrix H' = [P':I] where the rows of P' are

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix} \tag{190}$$

and $d-1=2$ parity checks orthogonal on e_1 can now be formed. The entire decoding process will be correct provided that no more than a single error is present in the received block. Since the code is cyclic, e_2, e_3, and e_4 can all be determined in a similar manner.

7.1 L-STEP ORTHOGONALIZATION

The generalized orthogonalization procedure can now be described. Starting with the original set of parity checks corresponding to the parity-check matrix H, suppose that we can form sets of at least d-1 parity checks orthogonal on selected sums of information noise bits. (If the code cannot be L-step orthogonalized, one might wish to use the decoding algorithm by forming as many orthogonal parity checks at each stage as possible.) These sums are then assumed to be known (threshold decoding is used to estimate each of these sums) and are treated as additional parity checks, that is, as known sums of noise bits. These additional parity checks are then combined with the original parity checks to produce a set of parity checks corresponding to a parity-check matrix H'. Provided that all the sums were correctly decoded, H' will be a true parity-check matrix. H' is then transformed to H" by a similar process. Ultimately, some $H^{(L)}$ is produced from which d-1 parity checks orthogonal on e_1 can be produced. If this procedure can be carried out for all e_j, j = 1, 2, ... k, then we say that the code can be L-step orthogonalized.

According to this definition, one-step orthogonalization is the same as complete orthogonalization as defined in section 6.1. We see that the Hamming (7,4) code can be 2-step orthogonalized.

If majority decoding is used as the decision rule at every stage of the orthogonalization procedure, then any combination of $\left\lfloor \frac{d-1}{2} \right\rfloor$, or fewer, errors in the received block will be corrected when the code can be L-step orthogonalized. This follows from the fact that the set of sums at the first stage will then be correctly decoded and H' will be a true parity-check matrix. By the same reasoning, H", H"', ... $H^{(L)}$ will be true parity-check matrices, and hence the majority decision on e_j will be correct for j = 1, 2, ... k.

It is plausible that better results could be obtained by using APP decoding at each stage. This must be true at the first stage because the APP decoding rule must give at least as small an error probability as majority decoding in estimating the sums from the sets of orthogonal parity checks. However, after these sums are combined with the original parity checks to form parity checks corresponding to the modified matrix H, the set of orthogonal parity checks formed from H' no longer have the strict statistical independence which they would have if H' were the original parity-check matrix. It seems reasonable, however, to treat these orthogonal parity checks as though they enjoyed the proper independence needed for application of the APP decoding rule because if H' were incorrectly formed and hence were not a true parity check matrix, then the decoding process would probably fail anyway.

7.2 FIRST-ORDER REED-MULLER CODES

As a first example to illustrate how the generalized orthogonalization procedure can be applied, we prove the following theorem.

THEOREM 22: For any M, there exists a binary $(2^M, M+1)$ block code with $d = 2^{M-1}$

that can be 2-step orthogonalized.

PROOF 22: Consider the parity-check matrix [P:I], where the rows of P contain all (M+1)-tuples of odd weight three and greater. There are exactly $2^M - M - 1$ such rows and hence $n - k = 2^M - M - 1$. This can be seen from the fact that half of the set of all 2^{M+1} (M+1)-tuples have odd weight, and P has as rows all these (M+1)-tuples except the M+1 unit vectors. Since $k = M + 1$, we have $n = 2^M$ as stated in the theorem. We now show that such a code has minimum distance $d = 2^{M-1}$ and can be two-step orthogonalized.

Consider the number of parity checks orthogonal on $e_1 + e_2$ which can be formed. For each row of P beginning with "1 0", there is a unique row beginning with "0 1" that is otherwise the same. The sum of these rows gives a parity check on $e_1 + e_2$ and no other information noise bits. Also, for each row of P beginning with "1 1" and having weight 5 or more, there is a unique row beginning with "0 0" that is otherwise the same. The sum of these rows again gives a check on $e_1 + e_2$ and no other information noise bits. Finally, the rows of P beginning with "1 1" and having weight three all have the third "one" in disjoint positions. Each of these gives a check on $e_1 + e_2$ and one other distinct information noise bit. Using this procedure, we can form as many parity checks orthogonal on $e_1 + e_2$ as there are rows of P with a "one" in the first column, and this number is $2^{M-1} - 1$.

From the symmetry of P, which has as rows all vectors of odd weight three and greater, it follows that $2^{M-1} - 1$ parity checks orthogonal on any sum of two information noise bits can be formed. We can then form a modified parity-check matrix [P:I] by assuming that $e_2 + e_3$, $e_3 + e_4$, ..., $e_{k-1} + e_k$ are now known and can be used to eliminate variables from the original parity-check equations. We observe that any sum of an even number of the variables e_2, e_3, ..., e_k can be formed from the assumed known sums (for example, $e_2 + e_4 = (e_2+e_3) + (e_3+e_4)$). But since all the rows of P have odd weight, all of the parity checks on e_1 check an even number of the variables e_2, e_3, ... e_k and these can all be eliminated by using the assumed known sums. Thus, using the rows of P beginning with a "one," we can form a P' with $2^{M-1} - 1$ rows of the form 1 0 0 ... 0, and hence $2^{M-1} - 1$ parity checks orthogonal on e_1 can be formed from the modified parity-check matrix. Again, from the symmetry of P, it follows that a similar process can be applied to e_2, e_3, ... e_k.

It remains to show that $d = J + 1 = 2^{M-1}$. The minimum distance must be at least this great. On the other hand, d can be at most one greater than the number of "ones" in any column of P (cf. Lemma D.1) and there are J "ones" in each column. Hence d must be exactly J + 1 which proves the theorem.

The codes described in the proof of Theorem 18 are equivalent to the first-order Reed-Muller codes,[37] that is, they differ from the latter codes only by a permutation of the code positions. However, the codes as we have used them are in systematic form which means that the first k encoded symbols are identical to the information symbols. This simplifies encoding, and is also usually desirable for other reasons (for example, some receiving stations may not have decoding circuits but may still wish to extract some data from the received messages). The Reed-Muller codes are not usually given in systematic form because the Reed decoding algorithm[13] does not apply directly, that is, without a prior linear transformation.

Finally, it should be observed that the threshold-decoding procedure can be streamlined in the following way. After all k-1 sums, $(e_1+e_2)^*$, $(e_2+e_3)^*$, ... $(e_{k-1}+e_k)^*$ have been determined and e_1^* has been determined from the modified parity-check matrix, the remaining variables e_2^*, e_3^*, ... e_k^* can all be found directly by combining these known quantities. For example, $e_3^* = e_1^* + (e_1+e_2)^* + (e_2+e_3)^*$. Thus a total of only k threshold-decoding decisions need to be made in the entire decoding process.

In general, when threshold decoding with L-step orthogonalization of a code is possible, it is never necessary to make more than k threshold-decoding decisions in the entire decoding process. This follows from the fact that each decision gives the decoded estimate of some sum of the variables e_1, e_2, ... e_k. Since there are only k of these variables, there can be at most k linearly independent sums formed from them. If k + m estimates of sums are formed by threshold decoding in the decoding process, then at least m of these estimates could have been found by taking linear combinations of the other estimates. Conversely, fewer than k threshold-decoding decisions can never suffice for the entire decoding process, since there are k independent quantities to be estimated.

7.3 HAMMING CODES

We have shown that the (7, 4), d=3, Hamming code can be 2-step orthogonalized. This code is one in a class of d=3 codes, having $n = 2^M - 1$ and $k = 2^M - M - 1$, which were discovered by Hamming.[11] These codes are cyclic codes. They are also "sphere-packed" codes, which means that all 2^n possible received n-tuples are distance $\frac{d-1}{2} = 1$ or less from some code word. Any decoding algorithm that corrects a single error in a block is then a maximum-likelihood decoding algorithm for these codes on the binary symmetric channel (see Appendix B). Majority decoding with 2-step orthogonalization is thus a maximum-likelihood decoding algorithm for the (7, 4) code.

We shall now show that all of the Hamming codes can be L-step orthogonalized for some L, and hence that majority decoding with L-step orthogonalization is a maximum-likelihood decoding procedure for these codes on a binary symmetric channel. We do not suggest that threshold decoding with L-step orthogonalization is a desirable way to decode this class of codes, since very simple decoding procedures may be employed such as the original procedure suggested by Hamming,[11] or the procedure advanced by Huffman.[38] Our purpose is only to show that it is possible to apply the generalized threshold decoding procedure to at least one class of high-rate codes. We conjecture that the same result will apply to other classes of high-rate codes.

THEOREM 23: For M = 2, 3, 4, ..., the M^{th}-order Hamming $(2^M - 1, 2^M - M - 1)$, d = 3, codes can be L-step orthogonalized for L no greater than M - 1.

PROOF 23: The M^{th}-order Hamming code has the parity-check matrix [P:I], where the columns of P contain all distinct nonzero M-tuples of weight two and greater. The M=2, or (3, 1), code can be 1-step orthogonalized according to Theorem 21. The M=3, or (7, 4) code can be 2-step orthogonalized as we have shown. We now show by induction that the M^{th}-order code can be (M-1)-step orthogonalized.

Suppose, first, that e_1 appears in an odd number of the original M parity checks, that is, that the first column of P has odd weight. Then let r_i be the i^{th} row of P, and γ_i be the sum of the remaining M-1 rows of P. γ_i and r_i both have "ones" in those positions, and only those, corresponding to columns of P with even weight and a "one" in the i^{th} row. Thus γ_i and r_i form a set of $d-1 = 2$ parity checks orthogonal on the sum of the information noise bits in these positions. M such pairs of orthogonal parity checks can be formed. The assumed known sums can then be used to eliminate all "ones" from columns of even weight in P. The only nonzero columns are the original set of $2^{M-1} - (M-1) - 1$ columns of P with odd weight. Then by omitting the last row and the all-zero columns, P is transformed into P' corresponding to the parity-check matrix of the $M-1^{th}$-order Hamming code. e_1 is still checked by the modified parity-check matrix, since its position corresponded to a column of P with odd weight.

Conversely, suppose that e_1 appears in an even number of the original M parity checks, that is, the first column of P has even weight. Consider now the set of M assumed-known sums described in the preceding paragraph. These sums contain a total of $2^{M-1} - 1$ distinct noise bits, namely those information noise bits in positions corresponding to columns of P with even weight. These sums, by themselves, correspond to a modified parity-check matrix made up of all of the columns of even weight in P. Omit any row of this matrix (unless there are only two modified parity checks on e_1, in which case omit a row that does not check on e_1), that is, discard one of the sums. The remaining M-1 sums then correspond to a modified parity-check matrix which is that of an M-1 order Hamming code. The modified parity bits are the M-1 information noise bits that were checked in the omitted row and only one other row.

After performing this process a total of M-3 times, we reach the parity matrix of the third-order Hamming code, and e_1 is still checked. This code can be 2-step orthogonalized. Thus $d-1 = 2$ parity checks orthogonal on e_1 can be formed after an (M-1)-step orthogonalization procedure. (We have not proved that this is the minimum number of steps required.) Clearly, the same argument applies as well to $e_2, e_3, \dots e_k$ and this proves the theorem.

7.4 BOSE-CHAUDHURI CODES

The Hamming codes can be considered as a special case of the more general class of cyclic codes known as the Bose-Chaudhuri codes.[16] (These codes were discovered independently by Hocquenghem.[39]) Although we have been unable to prove that this entire class of codes can be L-step orthogonalized, we have verified the fact that the codes of length 15 or less can be so orthogonalized. There are four such codes. The

(7, 4) and (15, 11) codes are Hamming codes and can be 2-step and 3-step orthogonalized, respectively, according to Theorem 23. The (15, 7) code can be 1-step orthogonalized as shown in section 6. 3. The remaining code, the (15, 5), d=7, code can be 2-step orthogonalized, as we now show.

The (15,5), d=7, Bose-Chaudhuri code has the parity-check matrix H = [P:I], where the rows of P are

$$\begin{array}{cccccc}
 & 1 & 0 & 1 & 0 & 1 \\
\rightarrow & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 0 & 1 & 0 \\
\rightarrow & 0 & 1 & 1 & 0 & \boxed{1} \\
 & 1 & 0 & 0 & 1 & 1 \\
\rightarrow & \boxed{1} & 1 & 1 & 0 & 0 \\
\rightarrow & 0 & 1 & 1 & \boxed{1} & 0 \\
 & 0 & 0 & 1 & 1 & 1 \\
\rightarrow & 1 & 0 & 1 & 1 & 0 \\
\rightarrow & 0 & 1 & 0 & 1 & 1
\end{array} \tag{191}$$

corresponding to the information noise bits that are checked by parity checks s_6 through s_{15}, respectively. For this code, $d - 1 = 6$ parity checks orthogonal on $e_2 + e_3$ can be formed as shown in (191). Since the code is cyclic, $d - 1 = 6$ parity checks orthogonal on $e_3 + e_4$ and $e_4 + e_5$ can also be formed, and this is easily verified directly. Thus we may assume that these sums are known and use them to eliminate variables in the original parity-check equations. From these sums, any sum of an even number of the variables e_2, e_3, e_4 and e_5 can be formed. This permits all other information bits to be eliminated from the $d - 1 = 6$ parity checks in Eq. 191 which check on e_1. This process transforms H into an H' from which six parity checks orthogonal on e_1 can be formed. Since the code is cyclic, the same procedure can be carried out for e_2, e_3, e_4, and e_5. Hence, the code can be 2-step orthogonalized.

To illustrate how L-step orthogonalization can be instrumented, we show in Fig. 27 the combinatorial element that would be used in the cyclic Type I decoder for the Bose-Chaudhuri (15,5) code. The remainder of the decoding circuit is the same as for the cyclic Type I decoder described in section 6. 3 for cyclic codes that can be one-step orthogonalized.

The upper three majority elements in Fig. 27 are used to form the decoded estimates of $e_2 + e_3$, $e_3 + e_4$, and $e_4 + e_5$. These quantities are then treated as additional parity checks and are combined with the original parity checks to form a set of $d - 1 = 6$ parity checks orthogonal on e_1. These latter checks are then operated on by the fourth majority element to produce the decoded estimate of e_1. It should be observed that two levels of majority logic are required in this combinatorial element. In general, with L-step orthogonalization, L levels of threshold logic will be required. However, as explained in section 7.2, no more than k majority elements are ever required in the combinatorial element. Thus, a number of majority elements growing exponentially with

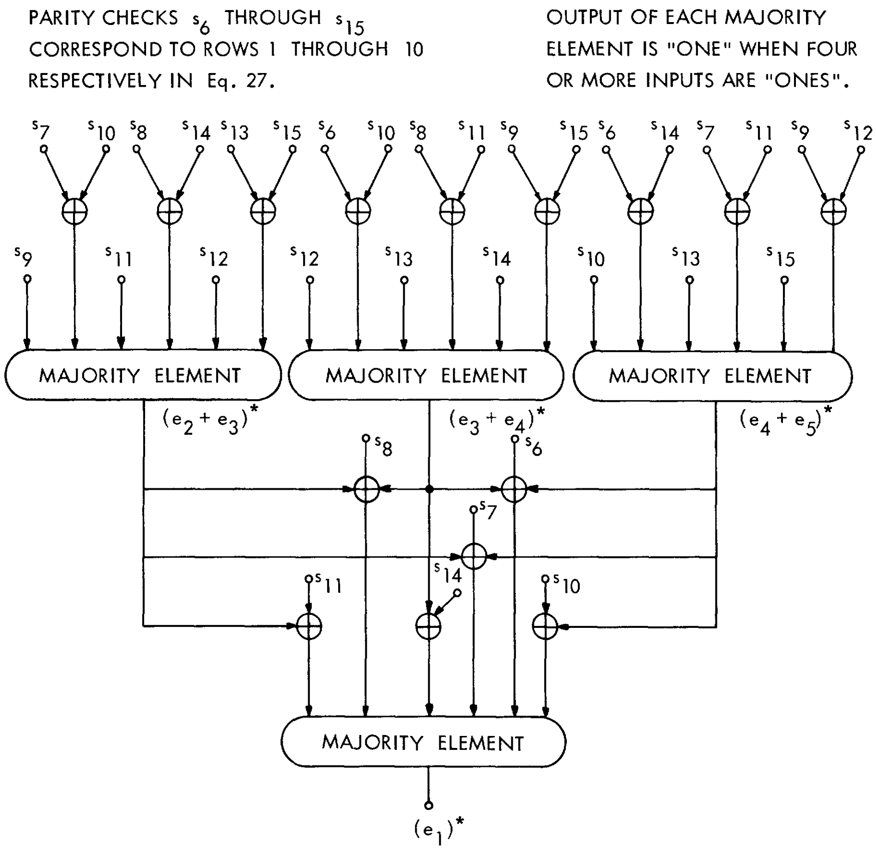

PARITY CHECKS s_6 THROUGH s_{15} CORRESPOND TO ROWS 1 THROUGH 10 RESPECTIVELY IN Eq. 27.
OUTPUT OF EACH MAJORITY ELEMENT IS "ONE" WHEN FOUR OR MORE INPUTS ARE "ONES".
s_7 s_{10} s_8 s_{14} s_{13} s_{15} s_6 s_{10} s_8 s_{11} s_9 s_{15} s_6 s_{14} s_7 s_{11} s_9 s_{12}
s_9 s_{11} s_{12} s_{12} s_{13} s_{14} s_{10} s_{13} s_{15}
MAJORITY ELEMENT
MAJORITY ELEMENT
MAJORITY ELEMENT
$(e_2 + e_3)^*$
$(e_3 + e_4)^*$
$(e_4 + e_5)^*$
s_8
s_6
s_7
s_{11}
s_{14}
s_{10}
MAJORITY ELEMENT
$(e_1)^*$

Fig. 27. Combinatorial element of a Cyclic Type I decoder for the (15, 5) Bose-Chaudhuri code.

L is not required, as would be the case when the L levels of majority elements had the structure of a full tree.

7.5 NONAPPLICABILITY TO CONVOLUTIONAL CODES

At first glance, it might appear that the L-step orthogonalization procedure could be used to enlarge the class of convolutional codes that can be threshold-decoded efficiently. Such is not the case as we now show, at least for the important cases for which the rate is one over an integer, that is, $R = 1/n_o$.

The parity-check matrix, $[H_\Delta:I]$ for an $R = 1/n_o$ convolutional code (cf. section 3.3) is such that the rows of H_Δ are given by

$$\begin{array}{lllll}
g_o^{(2)} & & & & \\
g_1^{(2)} & g_o^{(2)} & & & \\
g_2^{(2)} & g_1^{(2)} & g_o^{(2)} & & \\
\vdots & & & & \\
g_m^{(2)} & \cdots & g_2^{(2)} & g_1^{(2)} & g_o^{(2)} \\
 & & \vdots & & \\
g_o^{(n_o)} & & & & \\
g_1^{(n_o)} & g_o^{(n_o)} & & & \\
g_2^{(n_o)} & g_1^{(n_o)} & g_o^{(n_o)} & & \\
\vdots & & \ddots & & \\
g_m^{(n_o)} & \cdots & g_2^{(n_o)} & g_1^{(n_o)} & g_o^{(n_o)}
\end{array} \qquad (192)$$

Here, the g's are the coefficients in the (n_o-1) code-generating polynomials. Now suppose that it is possible to construct a set of J parity checks orthogonal on some sum of information noise bits, say on $e_{\alpha_1}^{(1)} + e_{\alpha_2}^{(1)} + \ldots + e_{\alpha_j}^{(1)}$. This means that it is possible to find J subsets of the rows of H such that the sum of the rows in each subset has a "one" in each of the j positions corresponding to the information noise bits $e_{\alpha_1}^{(1)}, e_{\alpha_2}^{(1)}, \ldots e_{\alpha_j}^{(1)}$, but any other position has a "one" in at most one of the J sums of rows. Assume that $\alpha_1 < \alpha_2 < \ldots < \alpha_j$. Now consider discarding all of the leftmost columns of H_Δ up to the column corresponding to $e_{\alpha_j}^{(1)}$. The rows of this modified matrix could then be combined to produce a set of J parity checks orthogonal on $e_{\alpha_j}^{(1)}$. But, because of the symmetry of each parity triangle, the array of coefficients in this modified matrix is exactly the same as for the matrix formed from H_Δ by discarding the same number of rows at the bottom of each parity triangle as columns that were discarded in the previous construction. It must then be possible to construct J parity checks orthogonal on $e_o^{(1)}$ directly from these rows of H_Δ.

In other words, it is always possible to form directly as many parity checks orthogonal on $e_o^{(1)}$ as can be formed on any sum of information noise bits. Thus, if L-step orthogonalization is possible, one-step, or complete, orthogonalization is also possible. Hence there is no advantage in the more general orthogonalization procedure for convolutional codes with rate $R = 1/n_o$.

7.6 SUMMARY

By extending the manner in which orthogonalization of parity checks is performed, we have seen that it is possible to apply threshold decoding in an efficient manner to at least one high-rate class of codes, as well as to low-rate codes. We have called this generalized procedure L-step orthogonalization.

The ultimate capability of L-step orthogonalization has not yet been determined, but the results obtained thus far have been most encouraging. We have not yet investigated a block (n, k) code for which it could be shown that L-step orthogonalization is impossible. Unfortunately, this may be due to the fact that the codes that we have studied were necessarily of short length, rather than to the generality of the method.

Although the circuitry required to implement threshold decoding with L-step orthogonalization is more complex than that required when one-step orthogonalization is possible, the circuits are still simple enough to be of practical interest (especially in the case of cyclic codes). The set of n-k parity checks can always be formed by a replica of the encoding circuit (cf. section 2.6). These parity checks can then be considered as the inputs to a combinatorial network having k outputs, namely e_1^*, e_2^*, ... e_k^*, the decoded estimates of the information noise bits. These quantities can then be added (modulo-two) to the received information bits to form the decoder output.

If the code can be L-step orthogonalized, the complete combinatorial network need contain no more than k threshold elements and at most (d)(k) modulo-two adders, as can be seen in the following way. From section 7.2, we conclude that no more than k threshold elements are needed. Enough adders are needed to form the set of orthogonal parity checks for each threshold element, and to combine the output of these devices to form the decoded estimates of the k information noise bits. Since no more than d-1 inputs are required for each of the threshold elements in order to be able to correct any combination of $\left\lfloor \frac{d-1}{2} \right\rfloor$ or fewer errors, no more than a total of k(d-1) adders will be required to form these inputs. Since the decoded estimate of each of the k information noise bits is the sum of the outputs of some set of the threshold elements, one adder suffices for the formation of each of these quantities. Thus a total of (d-1)k + k = dk adders always suffices in the complete combinatorial network.

VIII. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH

8.1 SUMMARY OF RESEARCH

In Section II we reviewed the properties of convolutional codes and proved some generalizations of known results for this class of codes. With this exception, all of this report has been devoted to the investigation of a decoding procedure for linear codes, which we have called " threshold decoding."

In Section I, two forms of threshold decoding were formulated: majority decoding, and APP decoding. These are both methods by which some noise bit, say e_m, can be estimated from a set of parity checks orthogonal on that noise bit. The majority decoding rule is based entirely on minimum distance, that is, it assigns to e_m the value that it has in the noise pattern of minimum weight that satisfies the set of orthogonal parity checks. The APP decoding rule, on the other hand, is based on a probability metric, that is, it assigns to e_m the value that is most probable, given the particular values of the orthogonal parity checks. Although these decoding rules are ordinarily distinct, we have seen (Sec. 5.1c) that majority decoding can be considered as the limiting case of APP decoding when the channel is estimated to have vanishingly small error probability.

In Sections III, IV, and V threshold decoding was applied to the specific task of decoding convolutional codes. First, it was necessary to construct codes to which the threshold decoding rules could be efficiently applied. Bounds on code quality were formulated to guide this search. Actual codes were then constructed both by trial-and-error and analytical techniques. This research is reported in Section III. In Section IV, simple decoding circuits were developed for the implementation of threshold decoding with convolutional codes. Finally, in Section V, data on error probability for threshold decoding of convolutional codes were presented for the binary symmetric channel, the binary erasure channel, and the Gaussian channel. This concluded the treatment of convolutional codes.

The application of threshold decoding to block linear codes was studied in Sections VI and VII. In Section VI we saw that the basic threshold decoding algorithms could be applied efficiently only to a small number of interesting block codes. To obviate this difficulty, a more general procedure for forming sets of orthogonal parity checks was introduced in Section VII. This generalization permitted efficient decoding of a somewhat larger class of block codes by the iterating of threshold-decoding decisions.

8.2 CONCLUSIONS

Based on the research reported in this report, the following conclusions with regard to threshold decoding can be stated:

a. Convolutional Codes

(i) Threshold decoding is easily instrumented (when applicable)

The decoding circuit need contain only $n_A - n_o$ stages of shift register, which appears to be the minimum one can hope to achieve in a reasonable decoding network. Moreover, the only nonlinear elements in the decoding circuit are of a very simple type, namely threshold logical elements. Even the necessary circuitry to compute the time-variant weighting factors (when applicable) for the threshold elements is not prohibitively complex.

(ii) Decoding performance is satisfactory for codes up to 100 bits in length

For very low rate codes (cf. secs. 3.6 and 3.7) good error-correcting performance can be obtained for long codes. However, over the range of rates of most practical interest at present, good error-correcting performance is limited to codes of approximately 100 transmitted bits in length. For such codes, the error probabilities that can be attained by threshold decoding are competitive with those obtained by other known error-correcting means.

(iii) The error probability cannot be made arbitrarily small at the receiver for a fixed rate of data transmission

This negative result establishes the lack of generality of threshold decoding for convolutional codes. We were able to show rigorously that this negative result obtains for the case $R = \frac{1}{2}$, but it seems plausible that it also applies to all other rates. In particular, for both the binary symmetric channel and the binary erasure channel, we have demonstrated the impossibility of making the decoding error probability arbitrarily small when $R = \frac{1}{2}$.

b. Block Codes

(i) Threshold decoding is easily instrumented (when applicable)

No more than k threshold logical elements are required in the complete decoding circuit for a block (n, k) code. The remainder of the decoding circuit contains a modest amount of linear components, namely a replica of the encoding circuit and no more than (k)(d) modulo-two adders, where d is the minimum distance of the code. In the important special case for cyclic codes, even simpler decoding circuits are possible.

(ii) Several interesting classes of codes can be efficiently decoded

The maximal-length codes, the first-order Reed-Muller codes, and the Hamming codes can all be L-step orthogonalized. This means that for these codes any error pattern of weight $\left\lfloor \frac{d-1}{2} \right\rfloor$ or less is correctable by majority decoding. It was shown in Sections VI and VII that several isolated, but interesting, codes could also be L-step orthogonalized.

c. General Remark

(i) Threshold decoding is limited primarily to binary codes

In section 6.2 we saw that the manner in which orthogonal parity checks are formed from the original parity checks of the code is such that full advantage cannot be taken of the higher-order, or nonbinary, alphabets.

(ii) Error-correction is not completely restricted by the minimum distance of the code

In general, the threshold-decoding algorithms permit the correction of many error patterns of weight greater than $\left\lfloor \frac{d-1}{2} \right\rfloor$. This is a very important feature in low-rate codes.

(iii) Error-correction can be improved when the a posteriori error probabilities of the received bits are known

In contrast to purely algebraic decoding procedures, the APP decoding rule makes use of the a posteriori probabilities of the received bits to decrease the probability of a decoding error. This feature of threshold decoding can be expected to be of special importance when decoding for a fading channel. This feature of threshold decoding is similar to that for Gallager' s low-density parity-check decoding, which is also a composite of algebraic and probabilistic techniques.[8]

8.3 RECOMMENDATIONS FOR FURTHER RESEARCH

Further research in continuation of that reported here is especially recommended in the following areas:

(i) Construction of good systematic classes of convolutional codes for random-error correction

Aside from certain classes of codes constructed for burst-error correction,[40] the only systematic classes of convolutional codes known are the uniform codes and the Reed-Muller-like codes that were found in the research for this report. Unfortunately, these are both classes of low-rate codes. It would be desirable to have several classes of good codes that might be used with threshold decoding (or other decoding algorithms).

(ii) Investigation of error-propagation with convolutional codes

It does not seem worth while to expend additional effort on the study of the general properties of threshold decoding with convolutional codes. The essential features are quite clear from Theorems 10-12. Similarly, it does not seem likely that simpler decoding circuits can be found. On the other hand, certain other features warrant additional research, and chief among these is the question of error propagation.

Two methods of controlling error propagation, resynchronization and error counting, were discussed in section 3.1. There is also a third possibility, automatic recovery by the decoder after a short burst of erroneously decoded symbols. This automatic recovery seems to be possible with sequential decoding, at least at high rates.[41, 42] However, the large increase in decoding computations which accompanies a decoding error with sequential decoding limits the practical use of automatic recovery to combat error propagation. However, threshold decoding does not share this limitation, since the computation effort is always fixed. It seems advisable to make a simulation study by using a fairly long code (e.g., the $n_A = 104$, $R = \frac{1}{2}$ code of Table II) to determine whether automatic recovery from an error is possible. If so, then an analytic effort would be in order to prove that the burst of erroneous symbols has some small average length.

(iii) Study of other nonlinear functions for use in decoding convolutional codes

Any combinatorial element which can form the decoded estimate of $e_o^{(1)}$ can be used as the decision element in the Type I decoder of Fig. 11, by replacing the threshold element and its associated adders. It might be possible to find other simple nonlinear functions that could be used to decode certain classes of convolutional codes efficiently, but this will doubtless be a very difficult area for research.

(iv) Investigation of the generality of L-step orthogonalization for block linear codes

This is the most important area for additional research. Theorems 21 and 23 suggest the possibility of a general theorem of the nature that " any binary code of length 2^M-1 or less can be (M-1)-step orthogonalized." Such a result, if it were true, would be extremely important. It would mean that any block code could be decoded at least up to its minimum distance, that is, correction of any error pattern of weight $\left\lfloor \frac{d-1}{2} \right\rfloor$ or less, in one operation by a combinatorial circuit consisting of no more than k threshold elements, an encoder, and no more than (k)(d) modulo-two adders. Research to prove this conjecture, or to provide a counterexample, could be of considerable value.

APPENDIX A
BASIC DEFINITIONS AND PROPERTIES OF MODERN ALGEBRA

An Abelian group is a set of elements and a rule of composition (which we write here as addition) such that for any three elements a, b, and c in the set the following axioms are satisfied: (1) $a + b$ is in the set. (2) $(a+b) + c = a + (b+c)$. (3) There is a unique element, 0, in the set which is such that $a + 0 = a$. (4) There is a unique element, $-a$, in the set which is such that $a + (-a) = 0$. (5) $a + b = b + a$. We shall use the term "group" to mean always an Abelian group.

A subgroup is a subset of the elements of a group which itself forms a group with respect to the same rule of composition.

If H is a subgroup of a group G, and if a is any element of G, then the coset containing a modulo H is the set of all elements b of G so that $a - b$ is in the subgroup H. The coset containing 0 is the subgroup itself. Any other coset will be called a proper coset.

A ring is an additive Abelian group for which a second rule of composition (which we write as multiplication) is defined in such a way that for any elements a, b, and c of the group, the following axioms are satisfied: (1) ab is in the group. (2) $a(bc) = (ab)c$. (3) $a(b+c) = ab + ac$. (4) $(b+c)a = ba + ca$. The ring is commutative if $ab = ba$.

A field is a ring in which the nonzero elements form an Abelian group with respect to multiplication. For $q = p^k$, where p is any prime number, it can be shown that there exists a field containing q elements. This field is called the Galois field of q elements and is denoted by GF(q).

The set of all polynomials in a single indeterminant, D, and with coefficients in GF(q) forms a commutative ring called the ring of polynomials with coefficients in GF(q).

The ideal generated by a polynomial f(D) is the set of all polynomials of the form h(D)f(D), where h(D) is any polynomial.

The residue class containing g(D) modulo the ideal generated by f(D) is the set of all polynomials h(D) such that $g(D) - h(D)$ is in the ideal.

The set of residue classes modulo the ideal generated by f(D) form a ring called the residue-class ring modulo the ideal. (This property implies that the sum, or product, of polynomials from two residue classes is always in the same third residue class regardless of the particular choice of those polynomials.

(The definitions and properties given here can be found in any text on modern algebra such as Garrett Birkhoff and Saunders MacLane, A Survey of Modern Algebra, Macmillan Company, New York, 1941.)

APPENDIX B
PROOF OF THEOREM 8 (RANDOM-CODING BOUND)

A group partition of an (n,k) code was defined in section 2.5 as a mapping of the code words into a subgroup H' and its proper cosets C_1', C_2', ... C_N' corresponding to a mapping of the set of 2^k information sequences into a subgroup H and its proper cosets C_1, C_2, ... C_N. We assume the existence of an ensemble of (n, k) codes in which each information sequence in a proper coset has probability $2^{-(n-k)}$ of being assigned any of the 2^{n-k} possible parity sequences.

Let I_j be any information sequence. A code word with this information sequence will be denoted I_j'. Suppose that a particular code is used over a binary symmetric channel with transition probability $p_o < 1/2$. Then, given a received sequence R', the probability $P(I_j' | R')$ that I_j' was transmitted is

$$P(I_j' | R') = p_o^w \, q_o^{n-w}, \tag{B-1}$$

where w is the number of positions in which I_j' and R' differ. The maximum likelihood rule is to choose that I_j as the transmitted information sequence for which the probability $P(I_j' | R')$ is greatest. Since the probability in Eq. B-1 decreases monotonically with w, the maximum likelihood rule reduces to choosing that I_j as the transmitted sequence for which I_j' and R' differ in the fewest positions.

We shall now calculate P(e), the average probability of error in deciding to which coset the transmitted information sequence belongs, with maximum likelihood decoding assumed. Because the set of distances from any code word to the code words in all other cosets is just equal to the set of weights of the code words in the proper cosets, without loss of generality we may assume that the all-zero sequence, I_o', was transmitted.

Let I_j be any information sequence in a proper coset, and assume that a noise pattern of weight W' has occurred on the channel. Let W_j be the number of information positions in the noise pattern that differ from I_j, and let W_j' be the total number of positions in which I_j' differs from the noise pattern. Since the all-zero sequence was transmitted, the received sequence differs from the transmitted sequence in W' positions. Over the ensemble of codes, the probability $P(W_j' \leq W')$ that I_j' is closer to the received sequence than is I_o' is given by

$$P(W_j' \leqq W') = 2^{-(n-k)} \sum_{i=0}^{W'-W_j} \binom{n-k}{i} \tag{B-2}$$

as can be seen from the fact that the summation gives the total number of parity sequences that, when attached to I_j, form code words differing in W' or fewer positions from the noise pattern, and each of these parity sequences has probability $2^{-(n-k)}$ by assumption.

The probability of error $P(e | W')$, given that the noise pattern has weight W', is the probability that some code word in a proper coset has W_j' less than or equal to W', and this probability is overbounded by the sum of the probabilities that each W_j' is less than or equal to W'.

$$P(e|W') \leqq \sum_{\substack{\text{all } W_j \text{ in the}\\ \text{proper cosets}}} 2^{-(n-k)} \sum_{i=0}^{W'-W_j} \binom{n-k}{i}. \tag{B-3}$$

The right-hand side is overbounded by summing over all possible W_j,

$$P(e|W') \leqq \sum_{W_j=0}^{k} \binom{k}{W_j} 2^{-(n-k)} \sum_{i=0}^{W'-W_j} \binom{n-k}{i} = 2^{-(n-k)} \sum_{i=0}^{W'} \binom{n}{i}. \tag{B-4}$$

Also, $P(e|W')$ is always overbounded by unity, since it is a probability. Thus, the average probability of error P(e), which is given by

$$P(e) = \sum_{\text{all } W'} P(W')P(e|W') = \sum_{W'=0}^{n} \binom{n}{W'} p_o^{W'} q_o^{n-W'} P(e|W'), \tag{B-5}$$

can be overbounded as

$$P(e) \leqq \sum_{W'=0}^{n} \binom{n}{W'} p_o^{W'} q_o^{n-W'} \left[\text{min.} \left[1,\ 2^{-(n-k)} \sum_{i=0}^{W'} \binom{n}{i}\right]\right]. \tag{B-6}$$

Inequality (B-6) is the usual form of the random-coding bound.

Several authors[1, 3, 4] have derived asymptotic bounds on (B-6). The tightest such bounds are those given by Wozencraft,[6] and we shall now state his results without proof. The bounds are stated in terms of the parameters p_t, p_{crit}, and R_{crit}, where

$$R = 1 - H(p_t), \tag{B-7}$$

$$\frac{p_{crit}}{1 - p_{crit}} = \sqrt{\frac{p_o}{1 - p_o}}, \tag{B-8}$$

$$R_{crit} = 1 - H(p_{crit}). \tag{B-9}$$

With this notation, (B-6) can be expressed in asymptotic form as

$$P(e) < A_{crit} 2^{-nE_{crit}} + A_t 2^{-nE_t} \qquad R < R_{crit}, \tag{B-10}$$

$$P(e) < (A_r + A_t) 2^{-nE_t} \qquad R_{crit} \leqq R < C = 1 - H(p_o), \tag{B-11}$$

where

$$E_{crit} = H(p_o) + H(p_t) - 2H(p_{crit}) + (p_{crit}-p_o) \log_2 (q_o/p_o), \tag{B-12}$$

$$E_t = H(p_o) - H(p_t) + (p_t-p_o) \log_2 (q_o/p_o), \tag{B-13}$$

$$A_t = \frac{1}{\sqrt{2\pi np_tq_t}} \frac{p_oq_t}{p_t - p_o}, \tag{B-14}$$

$$A_{crit} = \frac{1}{1 - (p_t/q_t)} \frac{p_t}{2\pi p_{crit}q_{crit}}, \tag{B-15}$$

and

$$A_r = \frac{1}{1 - (p_t/q_t)} \frac{1}{1 - (q_o/p_o)(p_t/q_t)^2} \frac{1}{2\pi np_tq_t} . \tag{B-16}$$

The first derivation of asymptotic forms for (B-6) was given by Elias[3] who showed that the exponent in (B-11) was the same as the exponent in the lower bound on P(e) obtained by "sphere-packing" arguments. This exponent can be obtained geometrically by the construction shown in Fig. B-1. In this figure, p_{crit} is that value for which the tangent to the entropy curve has a slope that is one-half that at p_o. Elias[3] credits Shannon with this interesting geometric interpretation of the exponent in the expression for P(e).

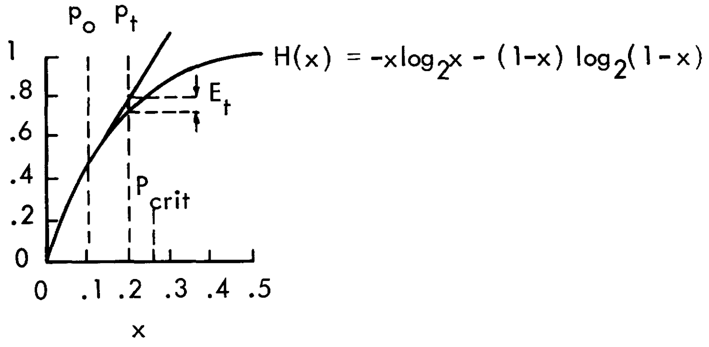


Fig. B-1. Geometric interpretation of exponent in expression for P(e).

APPENDIX C

PROOF OF THEOREMS 10 AND 11

1. Proof of Theorem 10

Consider the parity-check matrix $[H_\Delta:I]$ of a rate $R = 1/n_o$ convolutional code. H_Δ consists of a column of $n_o - 1$ parity triangles each of which corresponds to one of the code-generating polynomials. We have

$$H_\Delta = \left[\begin{array}{llll} g_o^{(2)} & & & \\ g_1^{(2)} & g_o^{(2)} & & 0 \\ \vdots & & & \\ g_m^{(2)} & & g_1^{(2)} & g_o^{(2)} \\ \vdots & & & \\ g_o^{(n_o)} & & & \\ g_1^{(n_o)} & g_o^{(n_o)} & & 0 \\ \vdots & & & \\ g_m^{(n_o)} & & g_1^{(n_o)} & g_o^{(n_o)} \end{array}\right] \tag{C-1}$$

Suppose, for example, that $g_{\alpha_1}^{(2)}, g_{\alpha_2}^{(2)}, \ldots g_{\alpha_N}^{(2)}$ are all of the nonzero coefficients of $G^{(2)}(D)$ in increasing order. Then from Eq. C-1 it can be seen that the parity checks in the first parity triangle which check on $e_o^{(1)}$ are $s_{\alpha_1}^{(2)}, s_{\alpha_2}^{(2)}, \ldots s_{\alpha_N}^{(2)}$ and these parity checks check on the following noise bits:

$$\begin{array}{lll} s_{\alpha_1}^{(2)} & \text{checks on} & e_o^{(1)}, e_{\alpha_1}^{(2)} \\ s_{\alpha_2}^{(2)} & \text{checks on} & e_o^{(1)}, e_{\alpha_2-\alpha_1}^{(1)}, e_{\alpha_2}^{(2)} \\ \vdots & & \\ s_{\alpha_N}^{(2)} & \text{checks on} & e_o^{(1)}, e_{\alpha_N-\alpha_{N-1}}^{(1)}, \ldots e_{\alpha_N-\alpha_1}^{(1)}, e_{\alpha_N}^{(2)}. \end{array} \tag{C-2}$$

The manner in which the information noise bits enter into these parity checks can be interpreted geometrically as shown in Fig. C-1. In Fig. C-1 α_1, α_2, and α_3 represent the first three nonzero terms in $G^{(2)}(D)$. X and Y correspond to the information positions, excluding the first position which corresponds to $e_o^{(1)}$, that are checked by

$s_{a_3}^{(2)}$. We observe that by choosing a_3 sufficiently large the noise bit corresponding to X can be made to have as large a subscript as desired, and all of the other noise bits checked will have larger subscripts.

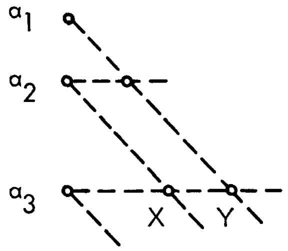


Fig. C-1. Geometric interpretation of parity-check structure.

From (C-2), we see that the i^{th} parity check on $e_o^{(1)}$ in each parity triangle checks a total of i noise bits with $e_o^{(1)}$ excluded; thus a total of

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2} \qquad \text{(C-3)}$$

noise bits are checked by the parity checks on $e_o^{(1)}$ in a parity triangle corresponding to a code-generating polynomial with N nonzero coefficients.

The construction of J parity checks orthogonal on $e_o^{(1)}$ can proceed in the following manner. One nonzero term at a time is placed into $G^{(2)}(D)$, $G^{(3)}(D)$, ... $G^{(n_o)}(D)$, in that order, until a total of J nonzero terms have been so placed. The j^{th} term is placed into $G^{(i)}(D)$ in such a way that the lowest order information noise bit checked by $s_{a_j}^{(i)}$, namely $e_{a_j-a_{j-1}}^{(1)}$, has subscript one greater than any information noise bit that is checked by any of the parity checks already formed that check $e_o^{(1)}$. This can always be done as explained above. In this manner, the set of J parity checks formed that check on $e_o^{(1)}$ consititute a set of J parity checks orthogonal on $e_o^{(1)}$.

The effective constraint length, n_E, is one plus the number of noise bits with $e_o^{(1)}$ excluded that are checked by these parity checks on $e_o^{(1)}$. Let

$$J = Q(n_o-1) + r \qquad 0 \leq r < n_o - 1. \qquad \text{(C-4)}$$

Then there are Q + 1 parity checks on $e_o^{(1)}$ in each of the first r parity triangles, after our construction, and Q in each of the last $n_o - 1 - r$ parity triangles. Thus, using (C-3), we find that the effective constraint length is given by

$$n_E - 1 = (r)\frac{(Q+1)(Q+2)}{2} + (n_o-1-r)\frac{(Q)(Q+1)}{2}. \qquad \text{(C-5)}$$

Finally, using (C-4) and the fact that $R = 1/n_o$, we can reduce (C-5) to

$$n_E = \frac{1}{2} J^2 \frac{R}{1-R} + \frac{1}{2} J + 1 + \frac{1}{2} r \left[1 - r \frac{R}{1-R} \right], \tag{C-6}$$

which is the statement of the theorem.

2. Proof of Theorem 11

For the case $R = k_o/n_o$, we seek a construction that will produce J parity checks orthogonal on $e_o^{(j)}$ for $j = 1, 2, \ldots k_o$. For this case, the parity-check matrix $[H_\Delta : I]$ is such that H_Δ consists of an $(n_o - k_o) x k_o$ array of parity triangles, one for each of the code-generating polynomials, as shown in Fig. C-2. The construction technique is very similar to that described above. Again, we shall place one nonzero term at a time into each of the code-generating polynomials, following the order shown in Fig. C-2, until J parity checks on $e_o^{(j)}$ have been formed for $j = 1, 2, \ldots k_o$. The manner of choosing terms will be such that each of these sets of J parity checks is orthogonal on $e_o^{(j)}$ for $j = 1, 2, \ldots k_o$.

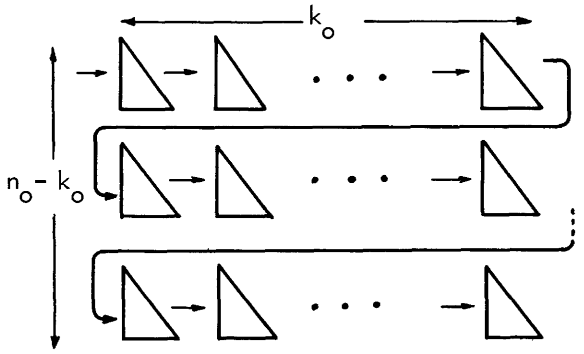


Fig. C-2. Arrangement of parity triangles in H_Δ.

The manner in which the information noise bits enter into a parity check on $e_o^{(j)}$ can be shown geometrically as in Fig. C-3. To be specific, we have drawn Fig. C-3 to correspond to the case in which the second nonzero term is placed in the third parity triangle in some row of Fig. C-2 for a code with $k_o = 4$. This forms a parity check on

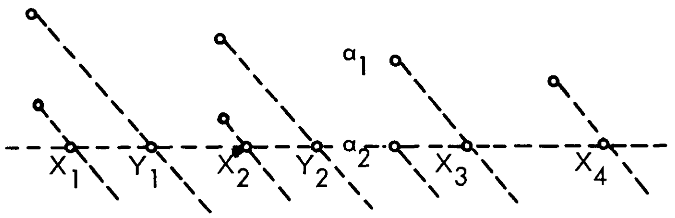


Fig. C-3. Geometric interpretation of parity-check structure.

$e_o^{(3)}$ and the information noise bits corresponding to positions X_1, Y_1, X_2, Y_2, X_3 and X_4. In addition, a single parity noise bit is also checked, namely $s_{a_2}^{(j)}$. Clearly, by choosing a_2 large enough, all of the information noise bits checked by this parity check, with $e_o^{(3)}$ excluded, can be made to have subscripts greater than any that appears in the parity checks already formed on $e_o^{(3)}$. Thus, by proceeding in this way, the J parity checks formed on $e_o^{(j)}$ for j = 1, 2, ... k_o are orthogonal on $e_o^{(j)}$. Clearly, from Fig. C-3, the set of parity checks on $e_o^{(k_o)}$ will check the greatest number of other noise bits. By definition, n_E is one plus this number.

If we let

$$J = Q(n_o-k_o) + r \qquad 0 \leq r < n_o - k_o, \tag{C-7}$$

then our construction places Q + 1 parity checks on $e_o^{(k_o)}$ in each of the first r rows of parity triangles, and Q parity checks on $e_o^{(k_o)}$ in each of the last $n_o - k_o - r$ rows. From Fig. C-3, we see that the i^{th} parity check on $e_o^{(k_o)}$ in any parity triangle will check exactly $k_o i$ noise bits, exclusive of $e_o^{(k_o)}$. Thus, using Eq. C-3, we find that

$$n_E - 1 = k_o\left[r\,\frac{(Q+1)(Q+2)}{2} + (n_o-k_o-r)\,\frac{Q(Q+1)}{2}\right]. \tag{C-8}$$

Finally, using the fact that $R = k_o/n_o$ and using Eq. C-7, we can reduce Eq. C-8 to

$$n_E = \frac{1}{2}\, J^2\, \frac{R}{1-R} + k_o\frac{J}{2} + 1 + \frac{1}{2}\, r\left[k_o - r\,\frac{R}{1-R}\right] \tag{C-9}$$

which is the statement of the theorem.

APPENDIX D

PROOF OF THEOREM 21

1. Preliminary Considerations

It will be convenient to represent the parity-check matrix, $H = [P:I]$, of a block (n, k) code in the following form:

$$P = [h_1 h_2 \dots h_k], \tag{D-1}$$

where $h_1, h_2, \dots h_k$ are column vectors of dimension $n - k$. We use the notation $\|h_j\|$ to denote the weight of h_j, and we write $h_j \oplus h_i$ to denote the vector formed by adding the components of h_j and h_i mod-two.

The two following lemmas will be needed in the proof of the main theorem.

LEMMA D. 1: Given a block (n, k) code with minimum distance d, then

$$\|h_{a_1} \oplus h_{a_2} \oplus \dots h_{a_N}\| \geq d - N \tag{D-2}$$

for any distinct indices $a_1, a_2, \dots a_N$ in the set $1, 2, \dots k$.

This is a well-known result and states simply that the sum of any N columns of P must have weight at least $d - N$. The proof is very similar to that used for the corresponding result with convolutional codes (cf. proof of Theorem 11) and will be omitted here.

We shall write $h_1 h_2 \dots h_i$ to mean the vector inner product of $h_1, h_2, \dots h_i$, that is, the vector whose j^{th} component is the product of the j^{th} components of $h_1, h_2, \dots h_i$. With this notation, we have the following lemma.

LEMMA D. 2:

$$\|h_1 \oplus h_2 \oplus \dots h_N\| = \sum_{j=1}^{N} (-2)^{j-1} \sum_{\substack{\text{all distinct} \\ \text{sets of indices} \\ a_1 < a_2 < \dots a_j}} \|h_{a_1} h_{a_2} \dots h_{a_j}\| . \tag{D-3}$$

PROOF D. 2: Since this result does not appear to be generally known, we shall give a proof. Clearly, it suffices to prove that (D-3) holds for the mod-two sum of the binary scalars $b_1, b_2, \dots b_N$.

For N=2, Eq. D-3 gives

$$b_1 \oplus b_2 = b_1 + b_2 - 2b_1 b_2. \tag{D-4}$$

Here, we use $\oplus$ to indicate mod-two addition and "+" (and Σ) to indicate ordinary real number addition. Equation D-4 is readily verified by testing all four possible values of b_1, b_2.

We now assume that the lemma holds for $N = M$, and we wish to show that it holds also for $N = M + 1$. Using the associative law, we have

$$b_1 \oplus b_2 \oplus \dots \oplus b_{M+1} = (b_1 \oplus \dots \oplus b_M) \oplus b_{M+1}. \tag{D-5}$$

Applying (D-4) to the two terms on the right of (D-5), we obtain

$$b_1 \oplus \dots \oplus b_{M+1} = (b_1 \oplus \dots \oplus b_M) + b_{M+1} - 2b_{M+1}\,(b_1 \oplus \dots \oplus b_M). \tag{D-6}$$

Using the inductive assumption, we can write Eq. D-6 as

$$\begin{aligned} b_1 \oplus \dots \oplus b_{M+1} = {} & \sum_{j=1}^{M} (-2)^{j-1} \sum_{\substack{\text{distinct}\\ \text{indices}}} b_{a_1} b_{a_2} \dots b_{a_j} \\ & + b_{M+1} + (-2) b_{M+1} \sum_{j=1}^{M} (-2)^{j-1} \sum_{\substack{\text{distinct}\\ \text{indices}}} b_{a_1} b_{a_2} \dots b_{a_j}. \end{aligned} \tag{D-7}$$

But Eq. D-7 is the same as

$$b_1 \oplus \dots \oplus b_{M+1} = \sum_{j=1}^{M+1} (-2)^{j-1} \sum_{\substack{\text{distinct}\\ \text{indices}}} b_{a_1} b_{a_2} \dots b_{a_j} \tag{D-8}$$

which is the statement of the lemma. By induction, the lemma is true for all M.

2. Proof of Theorem 21

We wish to prove that when k is 3 or less, any block (n, k) code can be completely orthogonalized, that is, $d - 1$ parity checks orthogonal on e_j can be formed for $j = 1, 2, \dots k$. We shall prove this result by showing that when k is three or less the necessary conditions for minimum distance d coincide with the sufficient conditions for complete orthogonalization. (At this point, we observe from (D-1) that $\|h_1\|$ is the number of parity checks that check e_1, $\|h_1\| - \|h_1h_2\|$ is the number of parity checks that check on e_1 but not on e_2, etc.)

Case 1: $k = 1$, $H = [h_1 \vdots I]$.

A sufficient condition for $d - 1$ parity checks orthogonal on e_1 is

$$\|h_1\| \geq d - 1. \tag{D-9}$$

Whereas, from Lemma D. 1, a necessary condition for minimum distance d is

$$\|h_1\| \geq d - 1 \tag{D-10}$$

and this coincides with the sufficient condition for complete orthogonalization.

Case 2: $k = 2$, $H = [h_1h_2 \vdots I]$.

If e_2 appears alone in enough parity checks to eliminate e_2 from all but at most one of the parity checks in which it appears with e_1, then the case reduces to the previous case. Otherwise, at least as many parity checks orthogonal on e_1 can be formed as there are checks on e_1 alone (that is, no other information noise bits) plus checks on e_2 alone plus one. Thus a sufficient condition for $d - 1$ checks orthogonal on e_1 is

$$(\|h_1\| - \|h_1h_2\|) + (\|h_2\| - \|h_1h_2\| - 1) \geqslant d - 1 \tag{D-11}$$

or

$$\|h_1\| + \|h_2\| - 2\|h_1h_2\| \geqslant d - 2. \tag{D-12}$$

By symmetry, (D-12) is also a sufficient condition for $d - 1$ parity checks orthogonal on e_2, and hence is a sufficient condition for complete orthogonalization. Applying Lemma D. 1 to the first two columns of H and with the aid of Lemma D. 2, we find that (D-12) is also a necessary condition for minimum distance d.

Case 3: $k = 3$, $H = [h_1h_2h_3:I]$.

If the number of parity checks on e_2 and e_3 alone is no greater than the number on e_1 and e_2 and e_3, then this case reduces to the previous case. Otherwise, at least as many parity checks orthogonal on e_1 can be formed as there are checks on e_1 alone plus checks on e_2 alone plus one plus checks on e_3 alone plus one plus checks on e_1 and e_2 and e_3. This follows from the fact that when the latter condition is satisfied, there are enough checks on e_2 and e_3 to eliminate these variables from all of the parity checks on e_1 and e_2 and e_3, and from the fact that e_2, as well as e_3, can appear in one of the parity checks on e_1 after orthogonalization. Thus, a sufficient condition for $d - 1$ parity checks orthogonal on e_1 is

$$\begin{aligned}(\|h_1\| - \|h_1h_2\| - \|h_1h_3\| + \|h_1h_2h_3\|) + (\|h_2\| - \|h_1h_2\| - \|h_2h_3\| + \|h_1h_2h_3\| + 1)\\ + (\|h_3\| - \|h_1h_3\| - \|h_2h_3\| + \|h_1h_2h_3\| + 1) + \|h_1h_2h_3\| \geqslant d - 1\end{aligned} \tag{D-13}$$

or

$$\|h_1\| + \|h_2\| + \|h_3\| - 2(\|h_1h_2\| + \|h_1h_3\| + \|h_2h_3\|) + 4\|h_1h_2h_3\| \geqslant d - 3. \tag{D-14}$$

Again, by symmetry, (D-14) is also a sufficient condition for $d - 1$ parity checks orthogonal on e_2 or e_3, and hence is a sufficient condition for complete orthogonalization. Applying Lemmas D. 1 and D. 2 to the first three columns of H, we find that (D-14) is also a necessary condition for minimum distance d.

This proves the theorem.

Acknowledgement

I owe a very great debt of gratitude to Professor John M. Wozencraft for his patient and generous guidance of this research. To my thesis readers, Professor Peter Elias, Professor Robert G. Gallager, and Professor Claude E. Shannon, I wish to express my thanks for their suggestions and encouragement. The opportunity to associate with these four distinguished scholars has been one of the most rewarding aspects of this year of research.

I gratefully acknowledge contributions to this research which resulted from many stimulating conversations with my fellow-students in the Processing and Transmission of Information Group of the Research Laboratory of Electronics of the Massachusetts Institute of Technology. I am indebted to Dr. J. J. Bussgang of Signatron, Inc., for the support gained from his enthusiasm for this research.

The numerical work in Section V was done in part at the Computation Center of the Massachusetts Institute of Technology, and I am grateful for the use of this facility and for the resources of the Research Laboratory of Electronics. I wish also to thank the National Science Foundation whose fellowship grants made this research possible.

References

1. C. E. Shannon, Mathematical theory of communication, Bell System Tech. J. 27, 379; 623 (July and October 1948).
2. Ibid., loc. cit.
3. P. Elias, Coding for noisy channels, IRE Convention Record, Part IV, 1955, pp. 37-44.
4. R. M. Fano, Transmission of Information (The M.I.T. Press, Cambridge, Mass., and John Wiley and Sons, Inc., New York, 1961), pp. 244-245.
5. J. M. Wozencraft, Private communication (cf. Sec. 2.5).
6. J. M. Wozencraft, Sequential Decoding for Reliable Communications, Sc.D. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, June 1957.
7. J. M. Wozencraft and B. Reiffen, Sequential Decoding (Technology Press of Massachusetts Institute of Technology, Cambridge, Mass., and John Wiley and Sons, Inc., New York, 1961).
8. R. G. Gallager, Low Density Parity Check Codes, Sc.D. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, 1960.
9. J. Ziv, Coding and Decoding for Time-Discrete Amplitude-Continuous Memoryless Channels, Sc.D. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, February 1962, pp. 69-76.
10. R. M. Fano, Unpublished material presented at two Seminars given during the Spring 1962 semester at Massachusetts Institute of Technology.
11. R. W. Hamming, Error detecting and error correcting codes, Bell System Tech. J. 29, 147-160 (1950).
12. D. Slepian, A class of binary signalling alphabets, Bell System Tech. J. 35, 203-234 (1956).
13. I. S. Reed, A class of multiple-error-correcting codes and the decoding scheme, IRE Trans., Vol. IT-4, pp. 38-49, 1954.
14. D. E. Muller, Application of Boolean algebra to swtiching circuit design and to error detecting, IRE Trans.,Vol. EC-3, pp. 6-12, 1954.
15. W. W. Peterson, Encoding and error-correction procedures for the Bose-Chaudhuri codes, IRE Trans., Vol. IT-6, pp. 459-470, 1960.
16. R. C. Bose and D. K. Ray-Chaudhuri, On a class of error correcting binary group codes, Information and Control 3, 279-290 (1960).
17. P. Elias, Error-free coding, IRE Trans.,Vol. IT-4, pp. 29-37, 1954.
18. C. Campopiano, Construction of relatively maximal systematic codes of specified minimum distance from linear recurring sequences of maximal period, IRE Trans., Vol. IT-6, pp. 523-528, 1960. (This appears to be the first treatment of linear codes in an arbitrary Galois field from the same point of view as the material is presented in Sec. I of this report.)
19. J. M. Wozencraft and B. Reiffen, op. cit., pp. 51-53.
20. Ibid., p. 54.
21. D. A. Huffman, The synthesis of linear sequential coding networks, Information Theory, edited by C. Cherry (Academic Press, New York,1956), pp. 77-95.
22. W. W. Peterson, Error-Correcting Codes (Technology Press of Massachusetts Institute of Technology, Cambridge, Mass., and John Wiley and Sons, Inc., New York, 1961), pp. 51-52.
23. J. M. Wozencraft and B. Reiffen, op. cit., p. 71.

24. M. Plotkin, Binary codes with specified minimum distance, IRE Trans., Vol. IT-6, pp. 445-450, 1960.

25. W. W. Peterson, Error-Correcting Codes, op. cit., pp. 148-150.

26. J. M. Wozencraft and B. Reiffen, op. cit., p.67.

27. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. I (John Wiley and Sons, Inc., New York, 1957).

28. W. W. Peterson, Error-Correcting Codes, op. cit., pp. 175-180.

29. J. M. Wozencraft and M. Horstein, Digitalised Communication over Two-Way Channels,a paper presented at the Fourth London Symposium on Information Theory, London, August 29-September 3, 1960; see also Coding for Two-Way Channels, Technical Report 383, Research Labortory of Electronics, M.I.T., August 30, 1961.

30. M. A. Epstein, Algebraic Decoding for a Binary Erasure Channel, Technical Report 340, Research Laboratory of Electronics, M.I.T., March 14, 1958.

31. R. M. Fano, Transmission of Information, op. cit., pp. 129-130.

32. F. J. Bloom, S. S. L. Chang, B. Harris, A. Hauptschein, and K. C. Morgan, Improvement of binary transmission by null-zone reception, Proc. IRE 45,963-975 (1957).

33. W. W. Peterson, Error-Correcting Codes, op. cit., p. 147.

34. Ibid., p. 34.

35. J. E. Meggitt, Error correcting codes and their implementation for data transmission systems, IRE Trans., Vol. IT-7, No. 4, pp. 234-244, 1961.

36. W. W. Peterson, Error-Correcting Codes, op. cit., p. 204.

37. Ibid., p. 73.

38. D. A. Huffman, A linear circuit viewpoint on error-correcting codes, IRE Trans., Vol. IT-2, pp. 20-28, 1956.

39. A. Hocquenghem, Codes Correcteurs d'Erreurs, Chissres 2, 147-156 (1959).

40. D. W. Hagelbarger, Recurrent codes: Easily mechanized burst-correcting,binary codes, Bell System Tech. J. 38, 969-984 (1959).

41. J. M. Wozencraft, Private communication, Massachusetts Institute of Technology, 1962.

42. J. J. Bussgang (Signatron, Inc.), Private communication, 1962.