Low-Density Parity-Check Codes*

R. G. GALLAGER†

Summary—**A low-density parity-check code is a code specified by a parity-check matrix with the following properties: each column contains a small fixed number $j \geq 3$ of 1's and each row contains a small fixed number $k > j$ of 1's. The typical minimum distance of these codes increases linearly with block length for a fixed rate and fixed j. When used with maximum likelihood decoding on a sufficiently quiet binary-input symmetric channel, the typical probability of decoding error decreases exponentially with block length for a fixed rate and fixed j.**

A simple but nonoptimum decoding scheme operating directly from the channel a posteriori probabilities is described. Both the equipment complexity and the data-handling capacity in bits per second of this decoder increase approximately linearly with block length.

For $j > 3$ and a sufficiently low rate, the probability of error using this decoder on a binary symmetric channel is shown to decrease at least exponentially with a root of the block length. Some experimental results show that the actual probability of decoding error is much smaller than this theoretical bound.

Coding for Digital Data Transmission

CODING for error correction is one of the many tools available for achieving reliable data transmission in communication systems. For a wide variety of channels, the Noisy Channel Coding Theorem [1, 6] of Information Theory proves that if properly coded information is transmitted at a rate below channel capacity, then the probability of decoding error can be made to approach zero exponentially with the code length. The theorem does not, however, relate the code length to the computation time or the equipment costs necessary to achieve this low error probability. This paper describes a class of coding and decoding schemes that can utilize the long block lengths necessary for low error probability without requiring excessive equipment or computation.

The codes to be discussed here are special examples of parity-check codes.[1] The code words of a parity-check code are formed by combining a block of binary information digits with a block of check digits. Each check digit is the modulo 2 sum[2] of a prespecified set of information digits. These formation rules for the check digits can be conveniently represented by a parity-check matrix, as in Fig. 1. This matrix represents a set of linear homogeneous modulo 2 equations called parity-check equations, and the set of code words is the set of solutions of these equations. We call the set of digits contained in a parity-check equation a parity-check set. For example, the first parity-check set in Fig. 1 is the set of digits (1, 2, 3, 5).

The use of parity-check codes makes coding (as distinguished from decoding) relatively simple to implement. Also, as Elias [3] has shown, if a typical parity-check code of long block length is used on a binary symmetric channel, and if the code rate is between *critical rate* and channel capacity, then the probability of decoding error will be almost as small as that for the best possible code of that rate and block length.

Unfortunately, the decoding of parity-check codes is not inherently simple to implement, and thus we must look for special classes of parity-check codes, such as described below, for which reasonable decoding procedures exist.

Low-Density Parity-Check Codes

Low-density parity-check codes are codes specified by a matrix containing mostly O's and only a small number of 1's. In particular, an (n, j, k) low-density code is a code of block length n with a matrix like that of Fig. 2 where each column contains a small fixed number, j, of 1's and each row contains a small fixed number, k, of 1's. Note that this type of matrix does not have the check digits appearing in diagonal form as in Fig. 1. However, for

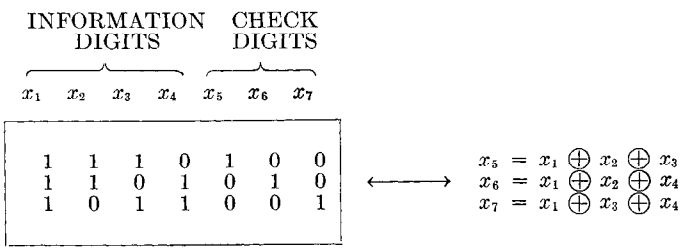

Fig. 1—Example of parity-check matrix.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Fig. 2—Example of a low-density code matrix; $N = 20, j = 3, k = 4$.

* Received by the PGIT, March 15, 1961. Supported in part by the U. S. Army Signal Corps, the AF Office of Scientific Research, the Office of Naval Research, the Mass. Inst. Tech. Computation Center, and the Applied Science Div., Melpar, Inc.

† Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, Mass.

[1] For a more detailed discussion of parity-check codes, see Slepian [2].

[2] The modulo 2 sum is 1 if the ordinary sum is odd and 0 if the ordinary sum is even.

coding purposes, the equations represented by these matrices can always be solved to give the check digits as explicit sums of information digits.

These codes are not optimum in the somewhat artificial sense of minimizing probability of decoding error for a given block length, and it can be shown that the maximum rate at which these codes can be used is bounded below channel capacity. However, a very simple decoding scheme exists for low-density codes, and this compensates for their lack of optimality.

The analysis of a low-density code of long block length is difficult because of the immense number of code words involved. It is simpler to analyze a whole ensemble of such codes because the statistics of an ensemble permit one to average over quantities that are not tractable in individual codes. From the ensemble behavior, one can make statistical statements about the properties of the member codes. Furthermore, one can with high probability find a code with these properties by random selection from the ensemble.

In order to define an ensemble of (n, j, k) low-density codes, consider Fig. 2 again. Note that the matrix is divided into j submatrices, each containing a single 1 in each column. The first of these submatrices contains all its 1's in descending order; *i.e.*, the i'th row contains 1's in columns $(i - 1)k + 1$ to ik. The other submatrices are merely column permutations of the first. We define an ensemble of (n, j, k) codes as the ensemble resulting from random permutation of the columns of each of the bottom $j - 1$ submatrices of a matrix such as Fig. 2, with equal probability assigned to each permutation.[3] There are two interesting results that can be proven using this ensemble, the first concerning the minimum distance of the member codes, and the second concerning the probability of decoding error.

The minimum distance of a code is the number of positions in which the two nearest code words differ. Over the ensemble, the minimum distance of a member code is a random variable, and it can be shown [4] that the distribution function of this random variable can be overbounded by a function such as sketched in Fig. 3. As the block length increases, for fixed $j \geq 3$ and $k > j$, this function approaches a unit step at a fixed fraction δ_{jk} of the block length. Thus, for large n, practically all the codes in the ensemble have a minimum distance of at least $n\delta_{jk}$. In Fig. 4 this ratio of typical minimum distance to block length is compared to that for a parity-check code chosen at random, *i.e.*, with a matrix filled in with equiprobable independent binary digits. It should be noted that for all the specific nonrandom procedures known for constructing codes, the ratio of minimum distance to block length appears to approach 0 with increasing block length.

[3] There is no guarantee that all the rows in such matrices will be linearly independent, and, in fact, all the matrices to be discussed here contain at least j-1 dependent rows. This simply means that the codes have a slightly higher information rate than the matrix indicates.

The probability of error using maximum likelihood decoding for low-density codes clearly depends upon the particular channel on which the code is being used. The results are particularly simple for the case of the BSC, or binary symmetric channel, which is a binary-input, binary-output, memoryless channel with a fixed probability of transition from either input to the opposite output. Here it can be shown [4] that over a reasonable range of channel transition probabilities, the low-density code has a probability of decoding error that decreases exponentially with block length and that the exponent is the same as that for the optimum code of slightly higher rate as given in Fig. 5.

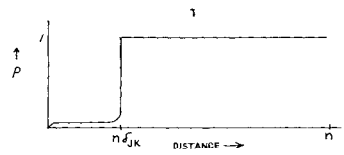

Fig. 3—Sketch of bound to minimum distance distribution function.

j	k	Rate	δ_{JK}	δ
5	6	0.167	0.255	0.263
4	5	0.2	0.210	0.241
3	4	0.25	0.122	0.214
4	6	0.333	0.129	0.173
3	5	0.4	0.044	0.145
3	6	0.5	0.023	0.11

Fig. 4—Comparison of δjk, the ratio of typical minimum distance to block length for an (n, j, k) code, to δ, the same ratio for an ordinary parity-check code of the same rate.

j	k	Rate	RATE FOR EQUIVALENT OPTIMUM CODE
3	6	0.5	0.555
3	5	0.4	0.43
4	6	0.333	0.343
3	4	0.25	0.266

Fig. 5—Loss of rate associated with low-density codes.

Although this result for the BSC shows how closely low-density codes approach the optimum, the codes are not designed primarily for use on this channel. The BSC is an approximation to physical channels only when there is a receiver that makes decisions on the incoming signal on a bit-to-bit basis. Since the decoding procedure to be described later can actually use the channel *a posteriori* probabilities, and since a bit-by-bit decision throws away available information, we are actually interested in the probability of decoding error of a binary-input, continuous-output channel. If the noise affects the input symbols symmetrically, then this probability can again be bounded by an exponentially decreasing function of the block length, but the exponent is a rather complicated function of the channel and code. It is expected that the same type of result holds for a wide class of channels with memory, but no analytical results

have yet been derived. For channels with memory, it is clearly advisable, however, to modify the ensemble somewhat, particularly by permuting the first submatrix and possibly by changing the probability measure on the permutations.

Decoding

Introduction

Two decoding schemes will be described here that appear to achieve a reasonable balance between complexity and probability of decoding error. The first is particularly simple but is applicable only to the BSC at rates far below channel capacity. The second scheme, which decodes directly from the *a posteriori* probabilities at the channel output, is more promising but can be understood more easily after the first scheme is described.

In the first decoding scheme, the decoder computes all the parity checks and then changes any digit that is contained in more than some fixed number of unsatisfied parity-check equations. Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.

If the parity-check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors. Thus when most of the parity-check equations checking on a digit are unsatisfied, there is a strong indication that that digit is in error. For example, suppose a transmission error occurred in the first digit of the code in Fig. 2. Then parity checks 1, 6, and 11 would be violated, and all three parity-check equations checking digit 1 would be violated. On the other hand, at most, one of the three equations checking on any other digit in the block would be violated.

To see how an arbitrary digit d can be corrected even if its parity-check sets contain more than one transmission error, consider the tree structure in Fig. 6. Digit d is represented by the node at the base of the tree, and each line rising from this node represents one of the parity-check sets containing digit d. The other digits in these parity-check sets are represented by the nodes on the first tier of the tree. The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the digits on tier 1, and the nodes on tier 2 represent the other digits in those parity-check sets. Notice that if such a tree is extended to many tiers, the same digit will appear in more than one place, but this will be discussed.

Assume now that both digit d and several of the digits in the first tier are transmission errors. Then on the first decoding attempt, the error-free digits in the second tier and their parity-check constraints will allow correction of the errors in the first tier. This in turn will allow correction of digit d on the second decoding attempt. Thus digits and parity-check equations can aid in decoding a digit seemingly unconnected with them. The probabilistic decoding scheme to be described next utilizes these extra digits and extra parity-check equations more systematically.

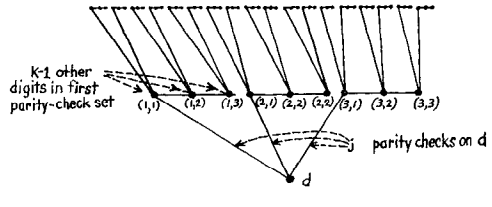

Fig. 6—Parity-check set tree.

Probabilistic Decoding

Assume that the code words from an (n, i, k) code are used with equal probability on an arbitrary binary-input channel. For any digit d, using the notation of Fig. 6, an iteration process will be derived that on the m'th iteration computes the probability that the transmitted digit in position d is a 1 conditional on the received symbols out to and including the m'th tier. For the first iteration, we can consider digit d and the digits in the first tier to form a subcode in which all sets of these digits that satisfy the j parity-check equations in the tree have equal probability of transmission.[4]

Consider the ensemble of events in which the transmitted digits in the positions of d and the first tier are independent equiprobable binary digits, and the probabilities of the received symbols in these positions are determined by the channel transition probabilities $P_x(y)$. In this ensemble the probability of any event conditional on the event that the transmitted digits satisfy the j parity-check equations is the same as the probability of an event in the subcode described above. Thus, *within this ensemble* we want to find the probability that the transmitted digit in position d is a 1 conditional on the set of received symbols $\{y\}$ and on the event S that the transmitted digits satisfy the j parity-check equations on digit d. We write this as

$$Pr\,[x_d = 1 \mid \{y\}, S].$$

Using this ensemble and notation, we can prove the following theorem:

Theorem 1: Let P_d be the probability that the transmitted digit in position d is a 1 conditional on the received digit in position d, and let P_{il} be the same probability for the l'th digit in the i'th parity-check set of the first tier in Fig. 6. Let the digits be statistically independent of each other, and let S be the event that the transmitted digits satisfy the j parity-check constraints on digit d. Then

$$\frac{Pr\,[x_d = 0 \mid \{y\}, S]}{Pr\,[x_d = 1 \mid \{y\}, S]} = \frac{1 - P_d}{P_d} \prod_{i=1}^{j} \left[\frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{il})}{1 - \prod_{l=1}^{k-1} (1 - 2P_{il})} \right].$$

[4] An exception to this statement occurs if some linear combination of those parity-check equations not containing d produces a parity-check set containing only digits in the first tier. This will be discussed later but is not a serious restriction.

In order to prove this theorem, we need the following lemma:

Lemma 1: Consider a sequence of m independent binary digits in which the l'th digit is 1 with probability P_l. Then the probability that an even number of digits are 1 is

$$\frac{1 + \prod_{l=1}^{m} (1 - 2P_l)}{2}.$$

Proof of Lemma: Consider the function

$$\prod_{l=1}^{m} (1 - P_l + P_l t).$$

Observe that if this is expanded into a polynomial in t, the coefficient of t^i is the probability of i1's. The function $\prod_{l=1}^{m} (1 - P_l - P_l t)$ is identical except that all the odd powers of t are negative. Adding these two functions, all the even powers of t are doubled, and the odd terms cancel out. Finally, letting $t = 1$ and dividing by 2, the result is the probability of an even number of ones. But

$$\frac{\prod_{l=1}^{m} (1 - P_l + P_l) + \prod_{l=1}^{m} (1 - P_l - P_l)}{2} = \frac{1 + \prod_{l=1}^{m} (1 - 2P_l)}{2},$$

thus proving the lemma.

Proof of Theorem: By the definition of conditional probabilities,

$$\frac{Pr\,[x_d = 0 \mid \{y\}, S]}{Pr\,[x_d = 1 \mid \{y\}, S]} = \left(\frac{1 - P_d}{P_d}\right)\left(\frac{Pr\,(S \mid x_d = 0, \{y\})}{Pr\,(S \mid x_d = 1, \{y\})}\right).$$

Given that $x_d = 0$, a parity check on d is satisfied if the other $(k - 1)$ positions in the parity-check set contain an even number of 1's. Since all digits in the ensemble are statistically independent, the probability that all j parity checks are satisfied is the product of the probabilities of the individual checks being satisfied. Using Lemma 1 this is

$$Pr\,(S \mid x_d = 0, \{y\}) = \prod_{i=1}^{j} \left[\frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{il})}{2}\right]. \tag{3}$$

Similarly,

$$Pr\,(S \mid x_d = 1, \{y\}) = \prod_{l=1}^{j} \left[\frac{1 - \prod_{l=1}^{k-1} (1 - 2P_{il})}{2}\right]. \tag{4}$$

Substituting (3) and (4) into (2) we get the statement of the theorem; Q.E.D.

Judging from the complexity of this result, it would appear difficult to compute the probability that the transmitted digit in position d is a 1 conditional on the received digits in two or more tiers of the tree in Fig. 6. Fortunately, however, the many-tier case can be solved from the 1-tier case by a simple iterative technique.

Consider first the 2-tier case. We can use Theorem 1 to find the probability that each of the transmitted digits in the first tier of the tree is a 1 conditional on the received digits in the second tier. The only modification of the theorem is that the first product is taken over only $j - 1$ terms, since the parity-check set containing digit d is not included. Now these probabilities can be used in (1) to find the probability that the transmitted digit in position d is 1. The validity of the procedure follows immediately from the independence of the new values of P_{il} in the ensemble used in Theorem 1. By induction, this iteration process can be used to find the probability that the transmitted digit d is 1 given any number of tiers of distinct digits in the tree.

The general decoding procedure for the entire code may now be stated. For each digit and each combination of $j - 1$ parity-check sets containing that digit, use (1) to calculate the probability of a transmitted l conditional on the received symbols in the $j - 1$ parity-check sets. Thus there are j different probabilities associated with each digit, each one omitting 1 parity-check set. Next, these probabilities are used in (1) to compute a second-order set of probabilities. The probability to be associated with one digit in the computation of another digit d is the probability found in the first iteration, omitting the parity-check set containing digit d. If the decoding is successful, then the probabilities associated with each digit approach 0 or 1 (depending on the transmitted digit) as the number of iterations is increased. This procedure is only valid for as many iterations as meet the independence assumption in Theorem 1. This assumption breaks down when the tree closes upon itself. Since each tier of the tree contains $(j - 1)(k - 1)$ more nodes than the previous tier, the independence assumption must break down while m is quite small for any code of reasonable block length. This lack of independence can be ignored, however, on the reasonable assumption that the dependencies have a relatively minor effect and tend to cancel each other out somewhat. Also, even if dependencies occur in the m'th iteration, the first $m - 1$ iterations have reduced the equivocation in each digit. Then we can consider the probabilities after the $m - 1$ iterations to be a new received sequence that should be easier to decode than the original received sequence.

The most significant feature of this decoding scheme is that the computation per digit per iteration is independent of block length. Furthermore it can be shown that the average number of iterations required to decode is bounded by a quantity proportional to the log of the log of the block length.

For the actual computation of the probabilities in Theorem 1, it appears to be more convenient to use (1) in terms of log-likelihood ratios. Let

$$\ln \frac{1 - P_d}{P_d} = \alpha_d \beta_d, \qquad \ln \frac{1 - P_{il}}{P_{il}} = \alpha_{il} \beta_{il}, \tag{5}$$

$$\ln \left[\frac{Pr\,[x_d = 0 \mid \{y\}, S]}{Pr\,[x_d = 1 \mid \{y\}, S]}\right] = \alpha'_d \beta'_d,$$

where α is the sign and β the magnitude of the log-likelihood ratio. After some manipulation, (1) becomes

$$\alpha'_d\beta'_d = \alpha_d\beta_d + \sum_{i=1}^{j}\left\{\left(\prod_{l=1}^{k-1}\alpha_{il}\right)f\left[\sum_{l=1}^{k-1}f(\beta_{il})\right]\right\}, \qquad (6)$$

where

$$f(\beta) = \ln\frac{e^\beta + 1}{e^\beta - 1}.$$

The calculation of the log-likelihood ratios in (6) for each digit can be performed either serially in time or by parallel computations. The serial computation can be programmed for a general-purpose computer, and the experimental data at the end of this paper was obtained in this manner. For fast decoding, parallel computing is more promising, and Fig. 7 sketches a simplified block diagram showing how this can be done.

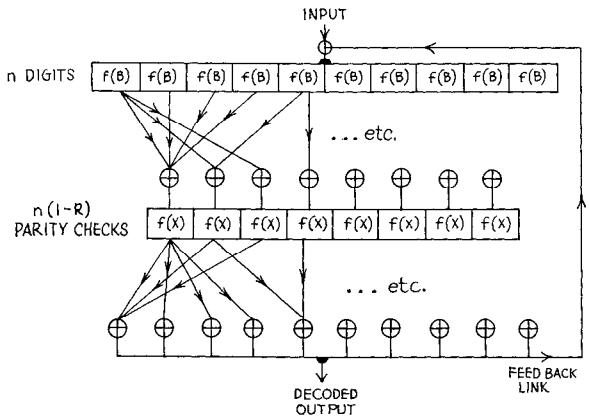

Fig. 7—Decoding apparatus.

If the input to the decoder is in the form of a log-likelihood ratio, the first row of boxes in Fig. 7 computes $f(\beta)$ for each digit, corresponding to the right-most operation in (6). The output from the adders on the next row is $\sum_{l=1}^{k-1} f(\beta_{il})$, corresponding to the two right-most operations in (6). Likewise, successive rows in Fig. 7 correspond to operations in (6) working to the left. Clearly, Fig. 7 omits some details, such as operations on the signs of the log-likelihood ratios and the association of j different log-likelihood ratios with each digit, but these create no essential difficulty.

We see from Fig. 7 that a parallel computer can be simply instrumented requiring principally a number proportional to n of analogue adders, modulo 2 adders, amplifiers, and nonlinear circuits to approximate the function $f(\beta)$. How closely this function must be approximated is a subject for further study, but there are indications that it is not critical.[5]

[5] Some recent experimental work indicates that if computation is strictly digital, 6 significant bits are sufficient to represent $f(\beta)$ without appreciable effect on the probability of decoding error.

Probability of Error Using Probabilistic Decoding

A mathematical analysis of probabilistic decoding is difficult, but a very weak bound on the probability of error can be derived easily.

Assume a BSC with crossover probability p_0 and assume first an (n, j, k) code with $j = 3$ parity-check sets containing each digit. Consider a parity-check set tree, as in Fig. 6, containing m independent tiers, but let the tiers be numbered from top to bottom so that the uppermost tier is the 0 tier and the digit to be decoded is tier m.

Modify the decoding procedure as follows: if both parity checks corresponding to the branches rising from a digit in the first tier are unsatisfied, change the digit; using these changed digits in the first tier, perform the same operation on the second tier, and continue this procedure down to digit d.

The probability of decoding error for digit d after this procedure is an upper bound to that resulting from making a decision after the m'th iteration of the probabilistic decoding scheme. Both procedures base their decision only on the received symbols in the m-tier tree, but the probabilistic scheme always makes the most likely decision from this information.

We now determine the probability that a digit in the first tier is in error after applying the modified decoding procedure described above. If the digit is received in error (an event of probability p_0) then a parity check constraining that digit will be unsatisfied if, and only if, an even number (including zero) of errors occur among the other $k - 1$ digits in the parity-check set. From Lemma 1, the probability of an even number of errors among $k - 1$ digits is

$$\frac{1 + (1 - 2p_0)^{k-1}}{2}. \qquad (7)$$

Since an error will be corrected only if both parity checks rising from the digit are unsatisfied, the following expression gives the probability that a digit in the first tier is received in error and then corrected.

$$p_0\left[\frac{1 + (1 - 2p_0)^{k-1}}{2}\right]^2. \qquad (8)$$

By the same reasoning, (9) gives the probability that a digit in the first tier is received correctly but then changed because of unsatisfied parity checks.

$$(1 - p_0)\left[\frac{1 - (1 - 2p_0)^{k-1}}{2}\right]^2. \qquad (9)$$

Combining (8) and (9), the probability of error of a digit in the first tier after applying this decoding process is

$$p_1 = p_0 - p_0\left[\frac{1 + (1 - 2p_0)^{k-1}}{2}\right]^2 + (1 - p_0)\left[\frac{1 - (1 - 2p_0)^{k-1}}{2}\right]^2. \qquad (10)$$

By induction it easily follows that if p_i is the probability of error after processing of a digit in the ith tier, then

$$p_{i+1} = p_0 - p_0\left[\frac{1 + (1 - 2p_i)^{k-1}}{2}\right]^2 + (1 - p_0)\left[\frac{1 - (1 - 2p_i)^{k-1}}{2}\right]^2. \quad (11)$$

We now show that for sufficiently small p_0, the sequence $[p_i]$ converges to 0. Consider Fig. 8 which is a sketch of p_{i+1} as a function of p_i. Since the ordinate for one value of i is the abscissa for the next, the dotted zig-zag line illustrates a convenient graphical method of finding p_i for successive values of i. It can be seen from the figure that if

$$\begin{aligned} 0 < p_{i+1} < p_i \quad &(\text{for} \quad 0 < p_i \leq p_0) \\ p_{i+1} = p_i \quad &(\text{for} \quad p_i = 0), \end{aligned} \quad (12)$$

then the sequence $[p_i] \to 0$. It can be seen from (11) that for p_0 sufficiently small, inequality (12) is satisfied. Fig. 9 gives the maximum p_0 for several values of k.

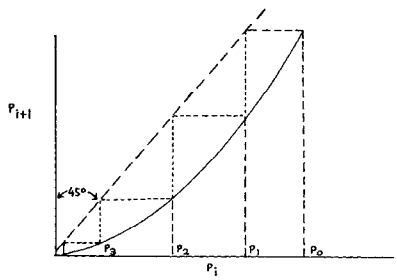

Fig. 8.

j	K	Rate	p_0
3	6	0.5	0.04
3	5	0.4	0.061
4	6	0.333	0.075
3	4	0.25	0.106

Fig. 9—Maximum p_0 for weak bound decoding convergence.

The rate at which $[p_i] \to 0$ may be determined by noting from (11) that for small p_i

$$p_{i+1} \approx p_i 2(k - 1)p_0.$$

From this it is easy to show that for sufficiently large i,

$$p_i \approx c[2(k - 1)p_0]^i,$$

where C is a constant independent of i. Since the number of independent tiers in the tree increases logarithmically with block length, this bound to the probability of decoding error approaches 0 with some small negative power of block length. This slow approach to 0 appears to be a consequence of the modification of the decoding scheme and of the strict independence requirement, rather than of probabilistic decoding as a whole.

This same argument can be applied to codes with more than 3 parity-check sets per digit. Stronger results will be achieved if for some integer b, to be determined later, a digit is changed whenever b or more of the parity-check constraints rising from the digit are violated. Using this criterion and following the reasoning leading to (11) we obtain

$$\begin{aligned} p_{i+1} = p_0 &- p_0 \sum_{l=b}^{j-1} \binom{j-1}{l} \cdot \left[\frac{1 + (1 - 2p_i)^{k-1}}{2}\right]^l \left[\frac{1 - (1 - 2p_i)^{k-1}}{2}\right]^{j-1-l} \\ &+ (1 - p_0) \sum_{l=b}^{j-1} \binom{j-1}{l} \cdot \left[\frac{1 - (1 - 2p_i)^{k-1}}{2}\right]^l \left[\frac{1 + (1 - 2p_i)^{k-1}}{2}\right]^{j-1-l}. \end{aligned} \quad (13)$$

The integer b can now be chosen to minimize p_{i+1}. The solution to this minimization is the smallest integer b for which

$$\frac{1 - p_0}{p_0} \leq \left[\frac{1 + (1 - 2p_i)^{k-1}}{1 - (1 - 2p_i)^{k-1}}\right]^{2b-j+1}. \quad (14)$$

From this equation, it is seen that as p_i decreases, b also decreases. Fig. 10 sketches p_{i+1} as a function of p_i when b is changed according to (14). The break points in the figure represent changes in b.

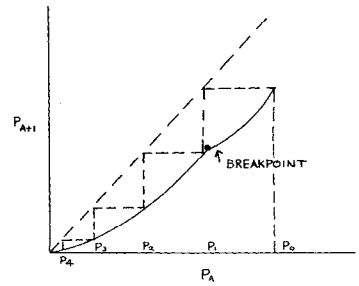

Fig. 10—Behavior of decoding iterations for $j > 3$.

The proof that the probability of decoding error approaches 0 with an increasing number of iterations for sufficiently small cross over probabilities is the same as before. The asymptotic approach of the sequence $[p_i]$ to 0 is different, however. From (14), if p_i is sufficiently small, b takes the value $j/2$ for j even and $j + 1/2$ for j odd. Using these values of b and expanding (13) in a power series in p_i,

$$p_{i+1} = p_0\binom{j-1}{\frac{j-1}{2}}(k - 1)^{(j-1)/2}p_i^{(j-1)/2} + \text{higher order terms} \quad (j \text{ odd}) \quad (15)$$

$$p_{i+1} = \binom{j-1}{j/2}(k - 1)^{j/2}p_i^{j/2} + \text{higher order terms} \quad (j \text{ even}).$$

Using this, it can be shown that for a suitably chosen positive constant C_{jk} and sufficiently large i

$$p_i \leq \exp\left[-C_{jk}\left(\frac{j-1}{2}\right)^i\right] \quad (j \text{ odd})$$
$$p_i \leq \exp\left[-C_{jk}\left(\frac{j}{2}\right)^i\right] \quad (j \text{ even}). \tag{16}$$

It is interesting to relate this result to the block length of the code. Since there are $(j-1)^m(k-1)^m$ digits in the m'th tier of a tree, n must be at least this big, giving the left side of (17). On the other hand, a specific procedure can be described [4] for constructing codes satisfying the right side of (17).

$$\frac{\ln (n)}{\ln (j-1)(k-1)} \geq m \geq \frac{\ln\left(\frac{n}{2k} - \frac{n}{2j(k-1)}\right)}{2 \ln (k-1)(j-1)}. \tag{17}$$

Combining (16) and (17), the probability of decoding error for a code satisfying (17) is bounded by

$$P_m \leq \exp -C_{jk}\left[\frac{n}{2k} - \frac{n}{2j(k-1)}\right]$$
$$\ln [(j-1)/2]/[2 \ln (j-1)(k-1)] \quad (j \text{ odd})$$

$$P_m \leq \exp -C_{jk}\left[\frac{n}{2k} - \frac{n}{2j(k-1)}\right]$$
$$\ln [(j/2)]/[2 \ln (j-1)(k-1)] \quad (j \text{ even}).$$

For $j > 3$, this probability of decoding error bound decreases exponentially with a root of n. Observe that if the number of iterations m which can be made without dependencies were $(2 \ln(j-1)(k-1))/(\ln j/2)$ times larger, then the probability of decoding error would decrease exponentially with n. It is hypothesized that using the probabilistic decoding scheme and continuing to iterate after dependencies occur will produce this exponential dependence.

A second way to evaluate the probabilistic decoding scheme is to calculate the probability distributions of the log-likelihood ratios in (6) for a number of iterations. This approach makes it possible to find whether a code of given j and k is capable of achieving arbitrarily small error probability on any given channel. With the aid of the IBM 709 computer, it was found that a code with $j = 3$, $k = 6$ is capable of handling transition probabilities up to 0.07 and with $j = 3$, $k = 4$, transition probabilities up to 0.144 can be handled. These figures are particularly interesting since they disprove the common conjecture that the computational cutoff rate of sequential decoding [7] bounds the rate at which any simple decoding scheme can operate.

Experimental Results

The probability of decoding an error $P(e)$ associated with a coding and decoding scheme can be directly measured by simulating both the scheme and the channel of interest on a computer. Unfortunately, the experiment must be repeated until there are many decoding failures if $P(e)$ is to be evaluated with any accuracy, and thus many times $1/P(e)$ trials are necessary. For block lengths of about 500, an IBM 7090 computer requires about 0.1 seconds per iteration to decode by the probabilistic decoding scheme. Consequently, many hours of computation time are necessary to evaluate even a $P(e)$ of the order of 10^{-4}.

Because of limitations on available computer time, all of the results presented will be for situations in which $P(e)$ is large. Certainly it would be more interesting to have results for small $P(e)$. However, the data presented are at least sufficiently convincing to justify further experimental work.

The first two codes to be discussed were used on the BSC and the last code on a Gaussian noise channel. The BSC was unduly emphasized for the following reasons: first, the effect of channel variations on the BSC can be eliminated by controlling the number of crossovers rather than the crossover probability; next, the BSC is convenient for comparison with other coding and decoding schemes; and finally, it is likely that the operation of the decoding scheme on one channel is typical of its operation on other channels.

A (504, 3, 6) Code on Binary Symmetric Channel

A code of block length 504 with each digit contained in three parity-check sets and each parity-check set containing 6 digits was selected by the IBM 704 computer using a pseudo-random number routine. The only restriction on the code was that no two parity-check sets should contain more than one digit in common. That restriction guaranteed the validity of the first-order iteration in the decoding process and also excluded the remote possibility of choosing a code with minimum distance of 2.

Fig. 11 plots the fraction of times the decoder was unable to decode correctly as a function of the number of crossovers. The number in parentheses beside each point is the number of trials performed with that number of crossovers. In all the trials on this code, the decoder never decoded to the wrong code word; it just failed to find a code word. If a feedback channel is available, this inability to decode troublesome noise patterns is not a serious limitation, since retransmission is possible.

Out of the error patterns correctly decoded, 86 per cent were decoded in between 9 and 19 iterations. The rest were spread out between 20 and 40 iterations. There appeared to be a slight increase in the number of iterations necessary to decode as the number of crossovers was increased from 37 to 41, but not enough to be statistically significant. The other curve drawn in Fig. 11 is the theoretical bound using maximum likelihood decoding.

In a later test made on an IBM 7090 computer, a (504,3,6) code was generated and 1000 sequences of 32 errors each were decoded. The process failed to decode in 26 cases and decoded the other 974 sequences correctly.

These results appear encouraging when we observe that no other known coding and decoding scheme of this rate is able to decode this many errors with a reasonable amount of computation. How well the decoding scheme works with smaller numbers of errors is of greater interest, though. The rate at which the experimental probability of error decreases as the number of crossovers decreases is discouraging, but there is no justification for extrapolating this curve to much smaller numbers of crossovers. Either a great deal of additional experimental data or a new theoretical approach will be necessary for evaluation of smaller numbers of cross overs.

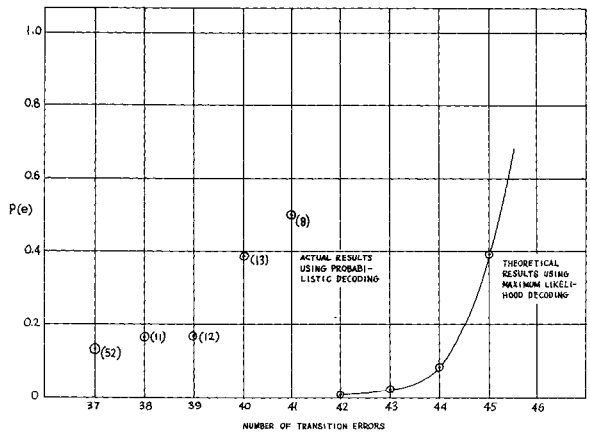

Fig. 11—Experimental results for (504, 3, 6) code as function of number of transition errors.

A (500, 3, 4) Code on the Binary Symmetric Channel

A (500, 3, 4) code, which has a rate of $\frac{1}{4}$, was chosen by the IBM 704 computer in the same way as the (504, 3, 6) code of the last section. Sequences containing from 20 to 77 crossovers were put in to be decoded. There were two sequences for each number of crossovers from 65 to 69 and from 72 to 77 and one sequence for all the other numbers. The decoding was successful for all sequences except one 73-crossover case, one 75-crossover case, and both 77-crossover cases. The theoretical error-correcting breakpoint for the (500, 3, 4) ensemble is 103 errors, and the error-correcting breakpoint for the ensemble of all codes of rate $\frac{1}{4}$ is 108 errors.

A (500, 3, 5) Code on White Gaussian Noise Channel

Assume a channel that accepts inputs of plus or minus 1 and adds a Gaussian random variable of mean 0 and variance 1 to the input to form the output. The log-likelihood ratio of the input conditional on the output is simply twice the received signal. The channel capacity of this channel can be calculated [5] to be 0.5 bits per symbol. However, if the receiver converts the channel into a BSC by making a decision on each symbol and throwing away the probabilities, the probability of crossover becomes 0.16, and the channel capacity is reduced to 0.37 bits per symbol.

In this experiment a (500, 3, 5) code, which has a rate of 0.4 bits per symbol, was simulated on the IBM 704 computer along with the channel just described. Probabilistic decoding was performed using the log-likelihood ratios at the output of the channel. Out of 13 trials, the decoding scheme decoded correctly on 11 trials and failed to decode twice.

This experiment is interesting since it suggests that the loss of rate necessitated by the nonoptimum coding and decoding techniques proposed here is more than compensated for by the opportunity of using the *a posteriori* probabilities at the channel output.

Bibliography

[1] C. E. Shannon, "Certain results in coding theory for noisy channels," *Information and Control*, vol. 1, pp. 6–25; September, 1957.
[2] D. Slepian, "A class of binary signalling alphabets," *Bell Sys. Tech. J.*, vol. 35, pp. 203–234; January, 1956.
[3] P. Elias, "Coding for two noisy channels," in "Information Theory," C. Cherry, Ed., 3rd London Symp., September, 1955; Butterworths Scientific Publications, London, Eng., 1956.
[4] R. G. Gallager, "Low Density Parity Check Codes," Sc.D. thesis, Mass. Inst. Tech., Cambridge; September, 1960.
[5] F. J. Bloom, *et al.*, "Improvement of binary transmission by null-zone reception," Proc. IRE, vol. 45, pp. 963–975; July, 1957.
[6] R. M. Fano, "The Transmission of Information," The Technology Press, Cambridge, Mass.; 1961.
[7] J. M. Wozencraft and B. Reiffen, "Sequential Decoding," The Technology Press, Cambridge, Mass.; 1961.