An Iterative Algorithm for Decoding Block Codes Transmitted Over a Memoryless Channel

H. Greenberger
Communications Systems Research Section

An algorithm has been developed which optimally decodes a block code for minimum probability of symbol error in an iterative manner. The initial estimate is made by looking at each bit independently and is improved by considering bits related to it through the parity check equations. The dependent bits are considered in order of increasing probability of error. Since the computation proceeds in a systematic way with the bits having the greatest effect being used first, the algorithm approaches the optimum estimate after only a fraction of the parity check equations have been used. This decoding algorithm will be tested via simulations of the (128, 64, 22) BCH code over the deep space channel.

I. Introduction

The measure of performance most commonly used in comparing error-correcting codes for transmitting binary information over noisy channels is the information bit probability of error. When block codes are used, an alternative measure is the code word probability of error. An optimum decoding algorithm which minimizes the probability of error for one measure is not optimum for the other; however, they are very close and become asymptotically equal at very high signal-to-noise ratios.

The optimum algorithm, under either measure, requires an amount of computation proportional to the number of possible code words that can be transmitted, thus limiting the size of the code for which such algorithms can be used. This problem can be bypassed at high signal-to-noise ratios where one can use nonoptimum decoders having simple decoding algorithms and make up the loss by using larger codes with greater minimum distance. At low signal-to-noise, however, there is an absolute limit to the minimum probability of error, depending only upon the decoding algorithm but not upon the code used. In this region, if algebraic decoding is used, the minimum probability of error is greater than what is currently realizable using convolutional coding with an optimum decoding algorithm. The bound for optimum decoding, however, is considerably lower, and worthwhile improvement in performance can be achieved if this limit can be approached. For this reason, approximations to optimum algorithms are being sought whose complexities are very much less but whose performances are close to optimum.

The approximations that can be made are of two types. First, some of the information available at the receiver can be ignored. An example is to decode each bit of the received code word independently, disregarding the dependence between the bits of each code word. This approach is called hard decision

decoding and is equivalent to approximating the channel by a binary symmetric one. Second, an incomplete algorithm can be used. Optimum decoding of block codes is equivalent to calculating the distance, according to some metric, between the received vector and all possible code words and selecting as the best estimate of the transmitted code word that one which is "closest" to it. Examples of incomplete algorithms for block codes are those which search for code words only within a certain distance of the received vector or select, in some other way, a subset of all possible code words among which the best estimate of the full decoding algorithm has a high probability of being found (Refs. 1-3).

II. An Iterative Algorithm

Rather than selecting a set of candidate code words according to some scheme as those mentioned above, an alternate approach is to disregard some of the channel information and make an optimum decision on what remains. This method works well at low signal-to-noise ratios where many bits have a high probability of error and can be disregarded with only a small penalty in performance. Such an algorithm can be put into an iterative form. Starting with an independent estimate of each received bit, the estimate can be improved by successively looking at bits related to it through the parity check equations of the code. These redundant bits are examined in order of increasing probability error. Each bit, as it is examined, improves the estimate of the other bits, the ones with a higher probability of error having a smaller effect than those with a lower probability of error. As poorer and poorer bits are examined, they perturb the previous estimate less and less, and a point is reached where the algorithm may be stopped with a high probability of being close to the optimum estimate.

As each symbol is received, the likelihood ratio

$$\phi_m = \frac{Pr(r_m \mid C_m = 0)}{Pr(r_m \mid C_m = 1)}$$

is calculated and mapped into the region (-1, +1) by the transformation

$$P_m = \frac{1 - \phi_m}{1 + \phi_m}$$

When the entire code word has been received, the symbols are sorted according to increasing probability of error (or equivalently, magnitude of P_m) so that the least reliable bits are to the right. The columns of the parity check matrix of the code are then permuted to the same order as the symbols have been. By using row operations only, the permuted parity check matrix can be reduced to a form which has a triangle of zeros in the upper-right-hand corner. The first P rows of this matrix represent the dependency among the $k + p$ symbols with the least probability of error. If the remaining $n - (k + p)$ symbols are considered erased, then an "optimum" decision, in the sense of minimum probability of symbol error, can be made using only this portion of the matrix. This form of the matrix also leads to an iterative algorithm. Starting with $p = 0$ and increasing p by one each iteration, successively poorer received bits are considered in estimating the transmitted symbols, until for $p = n - k$ the "true" optimum estimate is reached.

The decoding rule, for minimum probability of symbol error, using the method of decoding in the dual space of the code, is (see appendix)

$$\Lambda_m = \sum_j \prod_l P_l^{C'_{jl} \oplus \delta_{ml}} \underset{H_0}{\overset{H_i}{\gtrless}} 0$$

C'_{jl} is the lth bit of the jth code word in the dual code, that is, one formed by a linear combination of the rows of the parity check matrix; $\delta_{ml} = 1$ if $m = l$ and 0 if $m \neq l$. Thus when estimating the mth bit, the term P_l is included in the product if the lth bit of $C'_j = 1$ and $m \neq l$, or the lth bit of $C'_j = 0$ and $m = l$.

As an initial estimate of the transmitted symbols ($p = 0$), let $\hat{C}_m = 1$ if $P_m > 0$ and $\hat{C}_m = 0$ if $P_m \leqslant 0$. The first iteration uses only a single parity check equation so that there are only two words in the dual code, the all-zero code word and the one equal to the first row of the parity check matrix. The all-zero vector contributes to Λ_m the term P_m, which is the initial estimate, and the single parity check equation contributes a single product term of the P_l's.

The algorithm is then iterated, each time adding another parity check equation and taking into consideration the "best" of the remaining received bits. At each iteration the number of terms in the sum is doubled. Because of the reduced form of the parity check matrix, with zeros in the upper-right-hand corner, the terms Λ_m from previous iterations are not changed when a new row of the matrix is added and the new terms can be added directly to the previous stages' estimate. At each iteration the estimate of each bit is improved and the bit probability of error decreases.

A detailed example of this algorithm will be given for the (23,11) Golay code. The block length of this code is long enough to see the convergence of the algorithm to the optimum solution as the number of iterations increases and yet short enough so that a complete decoding can be done in a reasonable time. In order to estimate the performance of the algorithm at a given SNR, a large number of received vectors was generated and fully decoded. Only the 12 most reliable bits are estimated by the algorithm; the remainder are calculated through the parity check matrix. This forces the estimate of the transmitted vector to be a code word. The estimates of these bits were stored after each iteration, and the code word was considered to be correctly decoded when the 12 bits indicated as most reliable were estimated correctly.

Since the code is linear, the code space looks the same when viewed from any code word. Therefore, in an example, it can be assumed without loss of generality that the all-zero code word is transmitted. The received word can be represented by a vector of dimension 23, $y = (y, y_2, \ldots, y_{23})$. Each element of the vector is of the form $y_i = 1 + n_i$, where n_i is a sample of a zero mean gaussian process of variance $1/2\beta$, where

$$\beta = \sqrt{2 \frac{E_b}{N_0} rate}$$

As a numerical example, consider a received code word which contains four hard errors and therefore cannot be decoded correctly using algebraic decoding. In addition, one of the errors is among the 12 best bits so that the initial estimate of this algorithm would also be in error. A received vector (SNR = 1.0 dB) and the corresponding likelihood ratios are tabulated below

m	1	2	3	4	5	6	7	8	9	10
A_m	.99	1.25	.44	-.63	.52	.25	.01	.97	.36	-.90
Λ_m	.074	.041	.26	3.04	.22	.41	.70	.080	.32	5.62
P_m	.86	.92	.59	-.50	.64	.42	.17	.86	.52	-.70

m	11	12	13	14	15	16	17	18	19
A_m	1.13	.41	.11	.06	1.12	.33	-.74	-.24	.94
Λ_m	.054	.28	.56	.62	.054	.33	3.90	1.24	.083
P_m	.90	.56	.28	.24	.90	.50	-.59	-.11	.85

m	20	21	22	23
A_m	1.86	.71	2.87	1.73
Λ_m	.010	.14	.001	.013
P_m	.98	.75	.99	.97

m = bit number.

A_m = amplitude of mth bit.

Λ_m = likelihood ratio of mth bit.

P_m = transformed likelihood ratio of mth bit.

The first step of the algorithm is to sort the bits according to the absolute values of the transformed likelihood ratios P_m. The sorted order is

Sorted bit order	1	2	3	4	5	6	7	8	9
Original bit order	22	20	23	2	11	15	8	1	19
Sorted P_m	.99	.98	.97	.92	.90	.90	.86	.86	.85

Sorted bit order	10	11	12	13	14	15	16	17
Original bit order	21	10	5	17	3	12	9	16
Sorted P_m	.75	-.70	.64	-.59	.59	.56	.52	.50

Sorted bit order	18	19	20	21	22	23
Original bit order	4	6	13	14	7	18
Sorted P_m	-.50	.42	.28	.24	.17	-.11

The original parity check matrix for this code is

$$\begin{bmatrix}
10100100111110000000000 \\
01010010011111000000000 \\
00101001001111100000000 \\
00010100100111110000000 \\
00001010010011111000000 \\
00000101001001111100000 \\
00000010100100111110000 \\
00000001010010011111000 \\
00000000101001001111100 \\
00000000010100100111110 \\
00000000001010010011111
\end{bmatrix}$$

The columns are permuted to the same order as the received symbols to yield

$$\begin{bmatrix}
000010100010011100 11000 \\
000110000010001010 01110 \\
000011010001011000 01100 \\
000001000000001111 11100 \\
000001000011100001 01110 \\
000011010000100001 10101 \\
000001001000101101 00011 \\
010000011010100001 01001 \\
010010001100100100 00101 \\
110001001110001000 00001 \\
111010001100000001 01000
\end{bmatrix}$$

Reducing this matrix by row operations to form a triangle of zeros in the upper-right-hand corner yields

$$\begin{bmatrix}
001110101101100000 00000 \\
111100000111010000 00000 \\
011011011000101000 00000 \\
110010010100011100 00000 \\
100001011001001110 00000 \\
110011010011000001 00000 \\
010001011100000101 10000 \\
100001010100101001 01000 \\
100011000010101100 00100 \\
110000000110100101 00010 \\
110001001110001000 00001
\end{bmatrix}$$

The estimator $\Lambda_m^{(i)} = \sum_j \prod_l P_l^{C'_{jl} \oplus \delta_{ml}}$ (i = iteration number) will be used with this matrix and the sorted P_m's. For the ith iteration the code words C'_j are formed by all linear combinations of the first i rows of the matrix.

The initial estimate can be thought of as using the estimator with a dual code consisting of the all-zero vector alone. The index j, which indicates the jth code word in the dual code, has only the value 1, and C'_{jl} is zero for all values of l. In this case $\Lambda_m^{(0)} = \prod_l P_l^{\delta_{ml}}$ since $\delta_{ml} = 0$ for $m \neq l$ and 1 for $m = l$, $\Lambda_m^{(0)} = P_m$. When the only code word in the dual code is the all-zero one, the code itself contains all possible binary n-tuples. All the bits are independent, and the best estimate of any bit depends only upon the likelihood ratio of that bit itself.

The first iteration begins to use the dependence between the bits as expressed by the first parity check equation (the first row) of the H matrix

$$h_1 = 00111\ 01011\ 01100\ 00000\ 000$$

There are only two code words in the dual code, the all-zero code word and h_1, so that

$$\Lambda_m^{(1)} = P_m + \frac{P_3 P_4 P_5 P_7 P_9 P_{10} P_{12} P_{13}}{P_m}$$

for $m = 3,4,5,7,9,10,12,13$

$$= P_m + P_m P_3 P_4 P_5 P_7 P_9 P_{10} P_{12} P_{13}$$

otherwise.

Note that each term corresponds to a code word in the dual code. Each term contains the product of the transformed likelihood ratios of all the bits in the code word which are equal to one multiplied or divided by the ratio of the mth bit.

The second iteration uses two parity check equations h_1 and h_2.

$$h_2 = 11110\ 00001\ 11010\ 00000\ 000$$

The four code words in the dual code are

$$C'_1 = 0h_2 + 0h_1 = 00000\ 00000\ 00000\ 00000\ 000$$

$$C'_2 = 0h_2 + 1h_1 = 00111\ 01011\ 01100\ 00000\ 000$$

$$C'_3 = 1h_2 + 0h_1 = 11110\ 00001\ 11010\ 00000\ 000$$

$$C'_4 = 1h_2 + 1h_1 = 11001\ 01010\ 10110\ 00000\ 000$$

At this step the purpose of permuting the columns of the H matrix and reducing it to one with all zeros in a triangle in the upper-right-hand corner becomes clear. First, the code words in the dual code at the lth iteration contain all the code words of the $l-1$th iteration so that $\Lambda_m^{(l)} = \Lambda_m^{(l-1)} + 2^{l-1}$ new products. These new products are formed from the code words generated by adding modulo 2, the new parity

check equation h_l, to all the code words of the previous iteration. Second, all the new products include the transformed likelihood ratio of the $(k+l)$th bit but not of bits less reliable than this. Thus the lth iteration uses the previous estimate and the parity check equation containing the best bit not yet used to obtain an improved estimate.

At each iteration the estimate as to which bits are best may change. This is seen in Fig. 1, where the $\Lambda_m^{(l)}$'s are plotted as a function of iteration number. As more parity check equations are used, the absolute value of Λ (which is a measure of goodness) of the bits whose initial estimate was wrong decreases relatively rapidly. At the seventh iteration the best 13 bits are correct and the code word would be decoded correctly if the algorithm would be stopped at this point. At the final iteration, using the full decoding algorithm (equivalent to maximum likelihood), the best 17 bits are correct. Of the four bits whose initial estimate was wrong, only the one with the poorest initial estimate was corrected. In cases where there are few errors or the likelihood ratios of the correct bits are initially higher, the wrong bits are also corrected, but this is not necessary for correct decoding using this algorithm.

The performance of this code as a function of number of iterations is shown in Fig. 2. The zero iteration curve illustrates the performance possible by assuming there are no errors in the best 12 bits, while the 11th iteration curve represents the performance using the full decoding algorithm. Note that at the sixth iteration using $2^6 = 64$ terms in the sum of the estimator, the performance is almost equivalent to the full decoding algorithm which requires $2^{11} = 2048$ terms in the sum.

III. Conclusion

This algorithm is unique among those which approximate maximum likelihood decoders since it is not based upon generating a set of candidate code words. Its performance, in terms of probability of error for a given amount of computation, is comparable to these techniques and may be the basis of more efficient algorithms to be developed in the future.

Appendix

Minimum Probability of Symbol Error Decoding in the Dual Space of the Code (Ref. 4)

The decision rule which minimizes the error probability of the ith symbol is

$$Pr(C_i = 0 \mid \mathbf{y}) \underset{H_1}{\overset{H_0}{\gtrless}} Pr(C_i = 1 \mid \mathbf{y})$$

where the hypotheses are

H_0: The ith symbol of the transmitted code word $C_i = 0$

H_1: The ith symbol of the transmitted code word $C_i = 1$

In terms of the set of possible transmitted code words $\mathbf{C}_m \in C$

$$\sum_{\substack{\mathbf{C}_m \in C \\ C_{mi}=0}} Pr(\mathbf{C}_m \mid \mathbf{y}) \underset{H_1}{\overset{H_0}{\gtrless}} \sum_{\substack{\mathbf{C}_m \in C \\ C_{mi}=1}} Pr(\mathbf{C}_m \mid \mathbf{y})$$

If the code words are equally likely, using Bayes rule yields

$$\sum_{\substack{\mathbf{C}_m \in C \\ C_{mi}=0}} Pr(\mathbf{y} \mid \mathbf{C}_m) \underset{H_1}{\overset{H_0}{\gtrless}} \sum_{\substack{\mathbf{C}_m \in C \\ C_{mi}=1}} Pr(\mathbf{y} \mid \mathbf{C}_m)$$

Both sides of the inequality can be combined so that the sum is over the entire set of code words

$$\sum_{\mathbf{C}_m \in C} Pr(\mathbf{y} \mid \mathbf{C}_m)(-1)^{C_{mi}} \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

This decision rule requires calculating the probability of a given received vector for all words in the code. The estimate of the ith bit is found by summing separately all the probabilities for which the ith bit is equal to 1 and those for which the bit equals 0. The estimate is that value for which the sum is greater. The amount of computation required is proportional to 2^k, the number of words in the code. This decision rule can be transformed to a sum over the code words in the dual codes where the number of calculations required is proportional to 2^{n-k}. This in itself is not of much help when 2^{n-k} is also very large; however, this form of the decision rule can be converted to an approximate decoding algorithm where the sum is over only a small number of code words in the dual code. Such a decoding algorithm is useful if, for a large reduction in the amount of computation, only a small increase in the bit probability of error results, which indeed is the case.

To convert the decision rule to a sum over the dual code, the finite Fourier transform is used. Defining the input points as the elements of V_N, where V_N is the vector space of the 2^N possible binary vectors of dimension N, the Fourier transform of the function $f(\mu)$ defined on these points is

$$F(\mu) = \sum_{\mathbf{v} \in V_N} f(\mathbf{v})(-1)^{\mu \cdot \mathbf{v}}$$

Summing $F(\mu)$ over all $\mu \in C$

$$\sum_{\mu \in C} F(\mu) = \sum_{\mu \in C} \sum_{\mathbf{v} \in V_N} f(\mathbf{v})(-1)^{\mu \cdot \mathbf{v}}$$

Interchanging the order of summation

$$\sum_{\mu \in C} F(\mu) = \sum_{\mathbf{v} \in V_N} f(\mathbf{v}) \sum_{\mu \in C} (-1)^{\mu \cdot \mathbf{v}}$$

Representing the code by a set of basis vectors $\mathbf{b}_i$, which may be the rows of the generator matrix,

$$\mu = a_1 \mathbf{b}_1 + a_2 b_2 + \cdots + a_k \mathbf{b}_k \text{ (where } a \in \{0, 1\})$$

All possible code words are generated as the a_i take on the 2^k possible values so that

$$\sum_{\mu \in C} F(\mu) = \sum_{\mathbf{v} \in V_N} f(\mathbf{v}) \sum_{a_1, a_2 \cdots a_k} (-1)^{(a_1 \mathbf{b}_1 + a_2 \mathbf{b}_2 + \cdots + a_k \mathbf{b}_k) \cdot \mathbf{v}}$$

$$= \sum_{\mathbf{v}\in V_N} f(\mathbf{v}) \left(\sum_{a_1=0}^{1} (-1)^{a_1 \mathbf{b}_1 \cdot \mathbf{v}} \right)$$

$$\cdot \left(\sum_{a_2=0}^{1} (-1)^{a_2 \mathbf{b}_2 \cdot \mathbf{v}} \right) \cdots \left(\sum_{a_k=0}^{1} (-1)^{a_k \mathbf{b}_k \cdot \mathbf{v}} \right)$$

Any term, and therefore the entire expression equals zero if $\mathbf{v}\mathbf{b}_i \neq 0$. The only nonzero case is when $\mathbf{v} \cdot \mathbf{b}_i = 0$ for all i, that is for those $\mathbf{v}$ which are in the null space of the code C. These vectors are the elements of the dual code C' and the sum over μ for these vectors is $|C|$ = the number of vectors in C. The sum over C therefore reduces to

$$\sum_{\mu\in C} F(\mu) = |C| \sum_{\mathbf{v}\in C'} f(\mathbf{v})$$

For convenience the names of the code and its dual may be interchanged so that

$$\sum_{\mathbf{v}\in C} f(\mathbf{v}) = \frac{1}{|C'|} \sum_{\mu\in C'} F(\mu)$$

This expression can now be used to simplify the decision rule. Letting

$$f(\mathbf{v}) = Pr(\mathbf{y}|\mathbf{v})(-1)^{v_i}$$

the decision rule can be written as

$$\sum_{\mathbf{v}\in C} f(\mathbf{v}) \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

or

$$\sum_{\mu\in C'} F(\mu) \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

$F(\mu)$ may be calculated from the definition of the Fourier transform

$$\sum_{\mu\in C'} \sum_{\mathbf{v}\in V_N} Pr(\mathbf{y}|\mathbf{v})(-1)^{v_i}(-1)^{\mu\cdot\mathbf{v}} \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

The inner sum is over all possible vectors in the space V_N. In the general case this requires a summation over 2^N terms. However, if the channel is memoryless, this can be reduced to only $2N$ terms. In this case,

$$Pr(\mathbf{y}|\mathbf{v}) = \prod_{j=1}^{N} Pr(y_j|v_j)$$

So that the decision rule becomes

$$\sum_{\mu\in C'} \prod_{j=1}^{N} \sum_{v_j=0}^{1} Pr(y_i|v_j)(-1)^{\mu_j v_j}(-1)^{v_j \delta_{ij}} \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

Note that the term $(-1)^{v_i}$ is included only in the product for $i = j$ by representing it as $(-1)^{v_j \delta_{ij}}$. Expanding the inner sum

$$\sum_{\mu\in C'} \prod_{j=1}^{N} \left[Pr(y_j|v_j = 0) + Pr(y_j|v_j = 1)(-1)^{\mu_j + \delta_{ij}} \right] \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

This expression can be written in terms of the likelihood ratio

$$\phi_j = \frac{Pr(y_j|v_j = 0)}{Pr(y_j|v_j = 1)}$$

by dividing by

$$\prod_{j=1}^{N} Pr(y_j|v_j = 0)$$

The decision rule is then

$$\sum_{\mu\in C'} \prod_{j=1}^{N} \left[1 + \phi_j (-1)^{\mu_j \delta_{ij}} \right] \underset{H_1}{\overset{H_0}{\gtrless}} 0$$

Simplifying even further by dividing by $\prod_{j=1}^{N}(1+\phi_j)$

$$\sum_{\mu\in C'}\prod_{j=1}^{N}\left(\frac{1+\phi_j(-1)^{\mu_j\delta_{ij}}}{1+\phi_j}\right)\underset{H_1}{\overset{H_0}{\gtrless}}0$$

or in its simplest form

$$\sum_{\mu\in C'}\prod_{j=1}^{N}\left(\frac{1-\phi_j}{1+\phi_j}\right)^{\mu_j\oplus\delta_{ij}}\underset{H_1}{\overset{H_0}{\gtrless}}0$$

References

1. Chase, David, "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Trans. Inform. Th.*, Vol. 17-18, Jan. 72, pp. 170-180.

2. Dorsh, B. G., "A Decoding Algorithm for Binary Block Codes and J-ary Output Channels," *IEEE Trans. Inform. Th.*, Vol. 17-20, May 1974, pp. 391-394.

3. Baumert, L. D., and McEliece, R. J., "Performance of Some Codes on a Gaussian Channel," Proc. 1975 International Telemetry Conference, Washington D.C., pp. 189-195.

4. Hartmann, C. P., and Rudolph, L. D., "An Optimum Symbol-By-Symbol Decoding Rule for Linear Codes," *IEEE Trans. Inform. Th.*, Vol. 17-22, Sept. 1976, pp. 514-517.

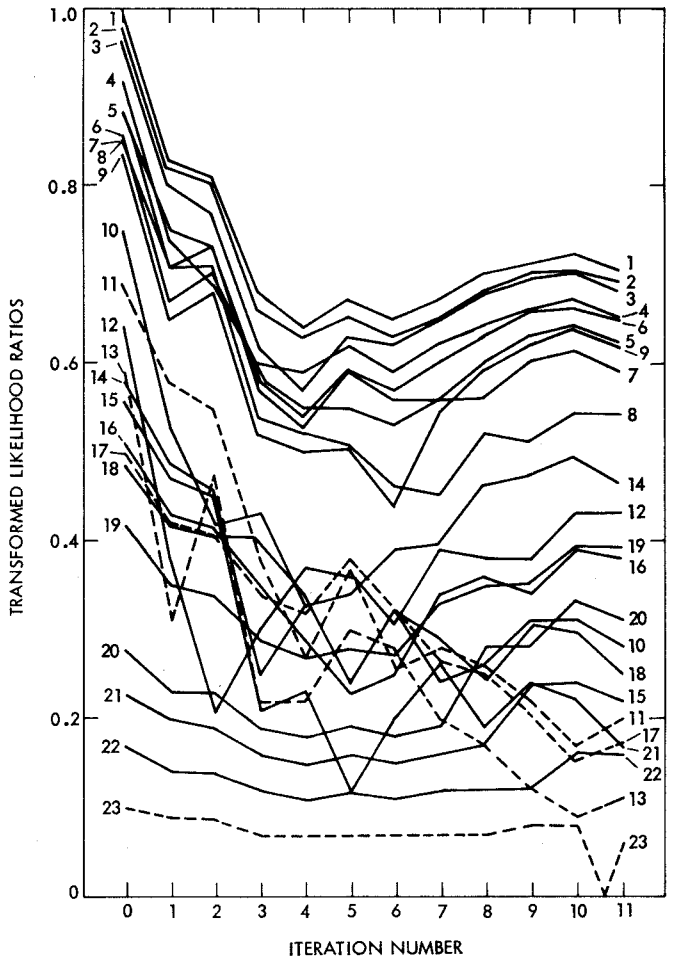

Fig. 1. Improvement of bit estimates as a function of iteration number

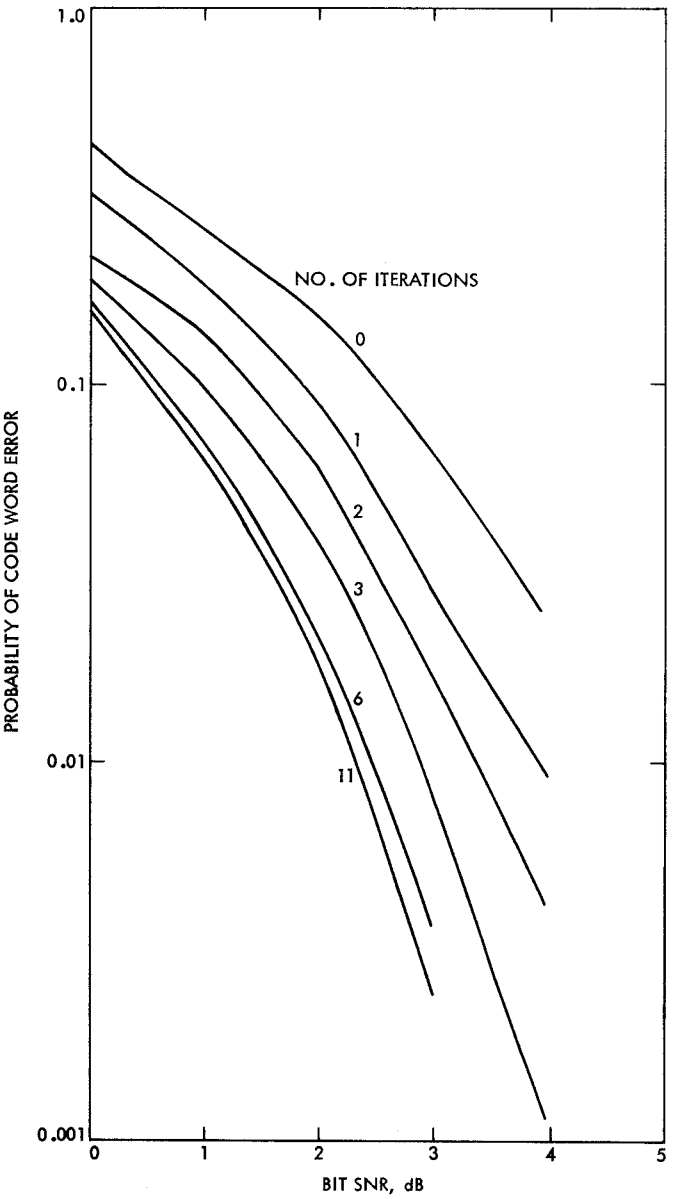

Fig. 2. Performance of the algorithm for the Golay (23.12) code