NASA TECHNICAL NOTE

NASA TN D-4402

c. 1

NASA TN D-4402

PERFORMANCE OF SEVERAL CONVOLUTIONAL AND BLOCK CODES WITH THRESHOLD DECODING

by Frank Neuman and Dale R. Lumb

Ames Research Center
Moffett Field, Calif.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • MARCH 1968

PERFORMANCE OF SEVERAL CONVOLUTIONAL AND BLOCK CODES WITH THRESHOLD DECODING

By Frank Neuman and Dale R. Lumb

Ames Research Center
Moffett Field, Calif.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

PERFORMANCE OF SEVERAL CONVOLUTIONAL AND BLOCK CODES WITH THRESHOLD DECODING

By Frank Neuman and Dale R. Lumb

Ames Research Center

SUMMARY

The performance of several codes applicable to space communications telemetry links was evaluated. The study was limited to high rate and relatively short constraint length codes. Explicitly, the (15,7) and (73,45) block codes and the (24,12) and (44,22) convolutional codes were investigated. Two types of threshold decoding schemes, derived by Massey, were applied, namely, majority decoding and the more powerful but complex a posteriori probability (APP) decoding.

The gaussian data channel and the decoders were simulated on a general purpose computer. The results show that, for the codes considered, APP decoding has approximately a 1.5 dB advantage over majority decoding. The most powerful code studied, (73,45) with APP decoding, gives a 2.1 dB improvement over a seventh bit parity check code. Also, new error deletion schemes were designed and tested for the codes studied. Because of their constant computation rate, these codes are candidates for high data rate channels. For the low data rates required for deep space missions such as Pioneer, however, the code performances reported here form a basis of comparison with the more powerful sequential decoding of convolutional codes, a variable computation rate decoding technique.

INTRODUCTION

The purpose of this investigation was to evaluate the performance of several cyclic block and convolutional codes with different decoding techniques introduced by Massey in 1963 (ref. 1). These techniques are called majority threshold decoding and a posteriori probability (APP) threshold decoding.

Although for the gaussian channel an improvement in performance was expected for APP decoding over majority decoding, no data were available on the amount of improvement possible with this more complex decoding scheme. Before this investigation, only the gross behavior of the codes was known for threshold decoding. The detail was insufficient to allow a deep space communication link designer to decide on the application of these codes. For this application of coding, it was found that the measures of performance given in the literature, namely, block and bit error probability estimates (for block codes) and probability to first error (for convolutional codes), were insufficient for comparing codes. A unifying measure is proposed in the section on measures of performance and has been applied in this report.

The codes were studied in somewhat greater detail than is required to calculate performance figures for several reasons: first, to present sufficiently detailed information to the spacecraft communication designer to allow him to choose a code suitable for his particular needs, or even to reject these coding schemes; second, to permit the study of additional processing steps that might improve the performance of the individual codes beyond that of the decoding procedures. Some improvements are indicated in this report, along with the results of trial experiments.

CODES AND DECODING TECHNIQUES INVESTIGATED

The codes investigated were from the class of block and convolutional codes which can be decoded by threshold decoding. In threshold decoding, the value of each bit is decided by comparison with a predetermined probability threshold. This type of decoding is relatively simple to implement compared to more optimal strategies. The convolutional codes that are threshold decodable have been constructed essentially through trial and error techniques, hence the name "trial and error codes" (ref. 1). The convolutional encoder is a binary shift register of length equal to the constraint length of the code. The shift register is tapped to calculate $m-1$ parity bits per information bit for a rate $1/m$ code.

A number of cyclic block codes have been found to be threshold decodable. The encoder, in this case, is a tapped cyclic shift register of length equal to the number of information bits in a block.

Of the codes that are threshold decodable, only relatively high rate codes were considered in this study. Compared to high rate codes, low rate codes have a lower signal-to-noise-power ratio per bit for the same information rate. This is not consistent with the requirement of maintaining coherent detection with state-of-the-art demodulation equipment. Also, the codes considered were of relatively short constraint length, since it is desirable for encoders for spacecraft application to be relatively simple.

Threshold decoding was introduced in 1963 by Massey (ref. 1), who invented two types of threshold decoding: majority decoding and a posteriori probability (APP) decoding of a set of orthogonal parity check equations. In the following section the procedure will be described briefly in terms of the algebraic manipulations required for the decoding process. The details of threshold decoding are shown by means of examples for the (24,12) convolutional code, and for the (15,7) block code in appendixes A and B.

Majority Decoding of a Convolutional Code

Majority decoding derives its name from the fact that a decision about whether to correct a received bit is based on the majority of a set of parity check equations. A functional block diagram is shown in figure 1. A part of the decoder duplicates the encoder which uses as input the received information bit sequence to calculate parity check bits. These are summed Mod 2 with the received parity bits. The resulting equation, called an S equation, is a function of error terms only. An S equation will be 0 when it contains an even number and 1 if it contains an odd number of error terms. There are as many S equations as the constraint length of the code. If one assumes that decoding will correct all errors, the error terms in the S equation, which were due to the received bit before decoding, are removed by complementing the S registers. With this assumption, the S equations are combined into a smaller set of orthogonal A equations, each of which consists of the error term for the bit to be decoded next, as well as other error terms; but none of the other error terms occur more than once in the set of A equations. For majority decision decoding, when the majority of A equations indicates an odd number of errors, the decision is made to correct the bit that is being decoded. In equation form, the rule is expressed as

$$e_1^i = 1$$

if

$$\sum_{i=1}^{J} A_i > J/2 \tag{1}$$

where e_1^i is the error term of the decoded bit. The decoder incorporates an alarm circuit (not shown) which resets the S register when the decoder has attempted too many corrections in a given interval. The intention is to reduce propagation of errors after the decoder has made an initial error.

APP Decoding of a Convolutional Code for the Gaussian Channel

Majority decoding, in general, is not optimum. The equations for the various A_k carry different weights of evidence that there may be an error in i_1', the bit being decoded. On the average, when $A_k = 1$, the more terms the equation has, the smaller is the probability that the error is in i_1', that is, $e_1^i = 1$. Therefore, optimum threshold decoding will be considered here for the case in which the received bits do not all have the same error probability, but the individual probabilities are known at the receiver. An example of such a channel is one employing coherent matched filter detection of a binary signaling alphabet. The matched filter output is a gaussian distributed noise voltage added to the binary signal. The receiver then uses the polarity of the received voltage V to assign to the received bit the more probable value of the binary number transmitted. In addition, the amplitude of the received voltage can be used to compute the probability that this assignment was wrong.

It is shown by Massey and is also proved in appendix C that for the time-varying channel, the following decoding theorem holds:

Choose

$$e_1^i = 1$$

if

$$\sum_{i=1}^{J} w_i A_i > T \tag{2}$$

Otherwise, choose $e_1^i = 0$

where w_i are weights and T is the threshold. Equation (2) is of the same form as the majority decoding theorem, where $T = J/2$ and $w_i = 1$ for all i. For APP decoding, the threshold and the weights are calculated for each bit separately from the individual bit error probabilities in the following manner: Define a new set of equations that corresponds to the A_i in equations (2) and (A16)

$$C_i = \sum_{j=1}^{n} c_j^\beta \tag{3}$$

where β denotes either an information bit i or a parity bit p, j indicates the jth information or parity bit, depending on the superscript, and the summation sign denotes ordinary addition. Note that the terms corresponding to e_1^i are missing from the new set of equations. The c_j are calculated from the bit error probabilities in the following manner:

$$c_j^\beta = -\log_e\left[1 - 2\Pr\left(e_j^\beta=1\right)\right] \tag{4}$$

and the weights are defined from the C_i as

$$w_i = 2\log_e\{\coth[(1/2)C_i]\} \tag{5}$$

$$w_o = 2\log_e\{\coth[(1/2)c_o]\} \tag{6}$$

where

$$c_o = -\log\left[1 - 2\Pr\left(e_1^i=1\right)\right] \tag{7}$$

Then the threshold is calculated for each particular bit as

$$T = (1/2)\sum_{i=o}^{J} w_i \tag{8}$$

Figures 2(a) to 2(c) are intended to clarify the above functional relationships. Figure 2(a) shows how, with increased energy level of the received signal plus noise, the individual bit error probability, $\Pr\left(e_j^\beta=1\right)$, decreases. It decreases faster for lower average bit error probability. However, c_j increases with decreasing level of the received bit (see fig. 2(b)). Consequently, the sum $(1/2)\sum c_j$ is large if many bits have below average levels. Figure 2(c) shows that a large $(1/2)\sum c_j$ results in a low weight, a result that agrees with intuition. The above generalization is illustrated by an example given in appendix A.

Majority Decoding of a Cyclic Block Code

The decoding process for majority decoding of a cyclic block code is very similar to that for the convolutional code. However, the A equations are generated differently. After each decoded bit in a block, the indices of the error terms are advanced by one (modulo the block length) to obtain the A equations for the next bit to be decoded. The circulation of the indices in the decoder is the circulation or feedback of the bits in a shift register. The majority decoding theorem is identical to that for the convolutional code:

$$\begin{aligned} &\text{Choose} && e_1^i = 1 \\ &\text{if} && \sum_{i=1}^{J} A_i > J/2 \qquad (9) \\ &\text{Otherwise, choose} && e_1^i = 0 \end{aligned}$$

In the decoding shift register the decoded bits can be fed back in one of two ways, either as they are received or after correction.

APP Decoding of Cyclic Block Codes for the Gaussian Channel

The APP decoding theorem for the cyclic block code is identical to that for the convolutional code, namely,

$$\begin{aligned} &\text{Choose} && e_1^i = 1 \\ &\text{if} && \sum_{i=1}^{J} w_i A_i > T \qquad (10) \\ &\text{Otherwise, choose} && e_1^i = 0 \end{aligned}$$

where the terms have the same meaning as before.

Compared with the binary feedback, the feedback of the c_j^β in the analog shift register is more involved. Here three types of feedback were explored.

(1) Soft feedback. The original c_j^β computed from the received bits are circulated in the register. This assumes that the error probability of the corresponding bit in the binary shift register has not changed. (This would have been a reasonable feedback connection if it had been used with the dotted feedback connection (Fig. 32 of appendix B) for the binary shift register. However, soft feedback was explored in conjunction with the corrected binary feedback, since this technique had been proposed in reference 1.)

(2) Hard decision feedback. After a bit is decoded its error probability is assumed to be zero, which means that the corresponding c_j^β is zero. This

causes a computing difficulty toward the end of each decoded block where the weights approach infinity. This difficulty was overcome by the rule of assigning to the decoded c_j^β a number corresponding to a low error probability.

(3) Full APP feedback. The name implies that after each bit is decoded, the bit error probability after decoding is fed back as its corresponding c_j^β. If one assumes bit for bit independence, the c_j^β can be calculated as follows: Let

$$x = \sum_{j=1}^{J} w_j A_j - T = -\log[p/(1 - p)] \tag{11}$$

where p is the bit error probability after decoding

$$e^{-x} = p/(1 - p) \tag{12}$$

$$c_j^\beta = -\log(1 - 2p) = -\log[(1 - e^{-x})/(1 + e^{-x})]$$

However, it should be noted that bit error probability after decoding each bit depends strongly on the bit error probability of many other bits and therefore the independence between bit error probabilities under which the coding theorem has been developed does not strictly hold. Nevertheless, this decoding method will be shown to perform somewhat better than the two others described previously.

MEASURES OF PERFORMANCE

In order to compare the effectiveness of different coding schemes, meaningful measures of performance must be found. The normalizing assumptions as well as the calculation of the performance measures and curves are shown in appendix D. Here the shortcomings and advantages of the various performance measures will be discussed, and the justification for the ones chosen will be given. One measure is the output bit error probability for a given signal energy per bit per noise spectral density, E_b/N_o. This measure accounts for the rate loss of codes with various ratios of information to parity bits. However, it does not account for the fact that various coding schemes will yield different output error streams with different statistical properties, the relative desirability of which will vary according to the processing strategy of the decoded data that follows.

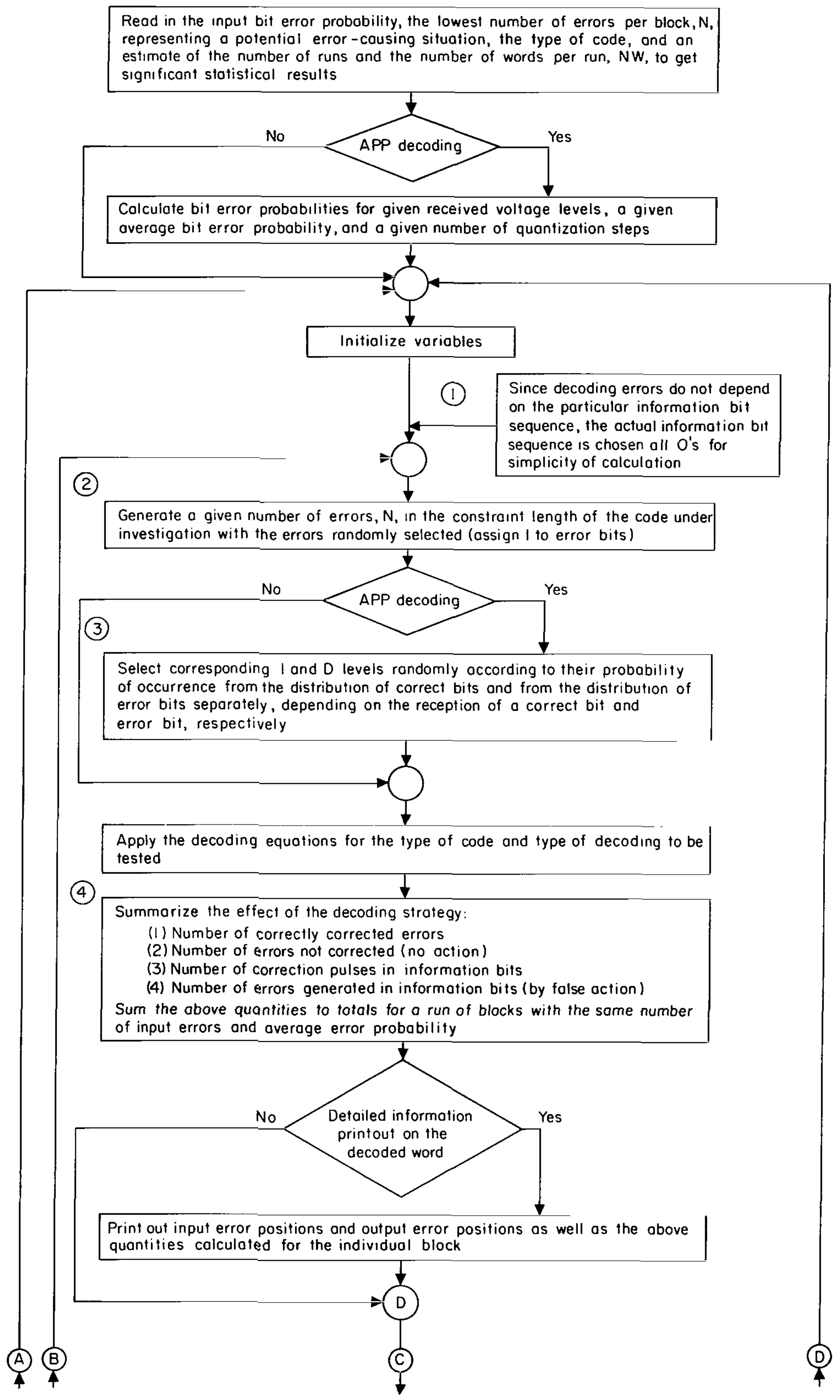
Scientific data from space probes are generally not sent in units of bits but words (groups of bits). For comparison of different coding schemes, word error probability will therefore be chosen. Some information about the performance of certain block codes has been published in terms of word error probability, with a word length equal to the block length of the code, but only because the bit error probability could not readily be calculated. The measure based on block length does not permit direct comparison of codes of various block lengths. For convolutional codes with threshold decoding usually the probability to the first error $p_1(e)$ has been computed. This measure is not satisfactory for the comparison between different kinds of codes. When the coding and decoding are completely simulated, one can choose any word length. The word length chosen for this study is six information bits, since this length has been used in the past on scientific deep space probes, such as Pioneer VI and VII. For both probes the simplest error detection codes are used, that is, one parity check bit is added to each six-bit data word. In such a code, words known to contain errors are deleted. For a space-to-ground telemetry channel, scientific data resulting from measurement of physical quantities are transmitted at specified intervals. As long as the word deletion rate is small (in the order of a few percent), the random deletions may have little more effect on the usefulness of the data than a coding rate loss. If one concedes this, the coding gain for the seven-bit parity check code is shown in figure 3. The coding gain is greater than 2.5 dB over a surprisingly large range of signal-to-noise ratios. One must keep in mind, however, that the deleted words in error occur at random. Random deletion of words may have a much more serious effect than the dB rate loss equation (10 $\log_{10}$(1-deletion rate)) would suggest, since for many experiments, groups of words are needed. (For an 8-percent deletion rate the rate loss is only 0.37 dB.) Thus, if another simple code with no deletions had the same word error probability, it might be preferred. Allowable deletion rates are somewhat subjective and depend on the experiment to which the data belong. The cost in deleted words (see fig. 3) was not treated as rate loss but instead was marked on the curves as a parameter.

DESCRIPTION OF DECODING SIMULATIONS

Most of the experiments were performed entirely on a general purpose computer for the following reasons: First, the gaussian channel is easily simulated on the computer. Second, decoding equipment for the APP decoding may best be simulated by a computer. Third, the input to the decoder and the output from it are available to the computer for immediate detailed comparison and analysis. Fourth, for low bit error probabilities, simulation of the data stream that includes errors allows a computing economy if only potential error-causing situations are generated, and the resulting output is decoded and analyzed. This is explained in detail in appendix E.

The overall experimental procedure for block code analysis is outlined in the accompanying flow chart. The flow chart has been somewhat idealized so that the arrangement would automatically generate sufficient data to produce the complete analysis for a given code at a specific input bit error probability. This would not have been difficult to program as an executive routine; however, the rather unpredictable computer time required to generate sufficient statistics made this method impractical.

Some of the individual blocks identified by reference numbers on the flow chart will now be discussed.



Read in the input bit error probability, the lowest number of errors per block, N, representing a potential error-causing situation, the type of code, and an estimate of the number of runs and the number of words per run, NW, to get significant statistical results

Calculate bit error probabilities for given received voltage levels, a given average bit error probability, and a given number of quantization steps

Initialize variables

Since decoding errors do not depend on the particular information bit sequence, the actual information bit sequence is chosen *all* O's for simplicity of calculation

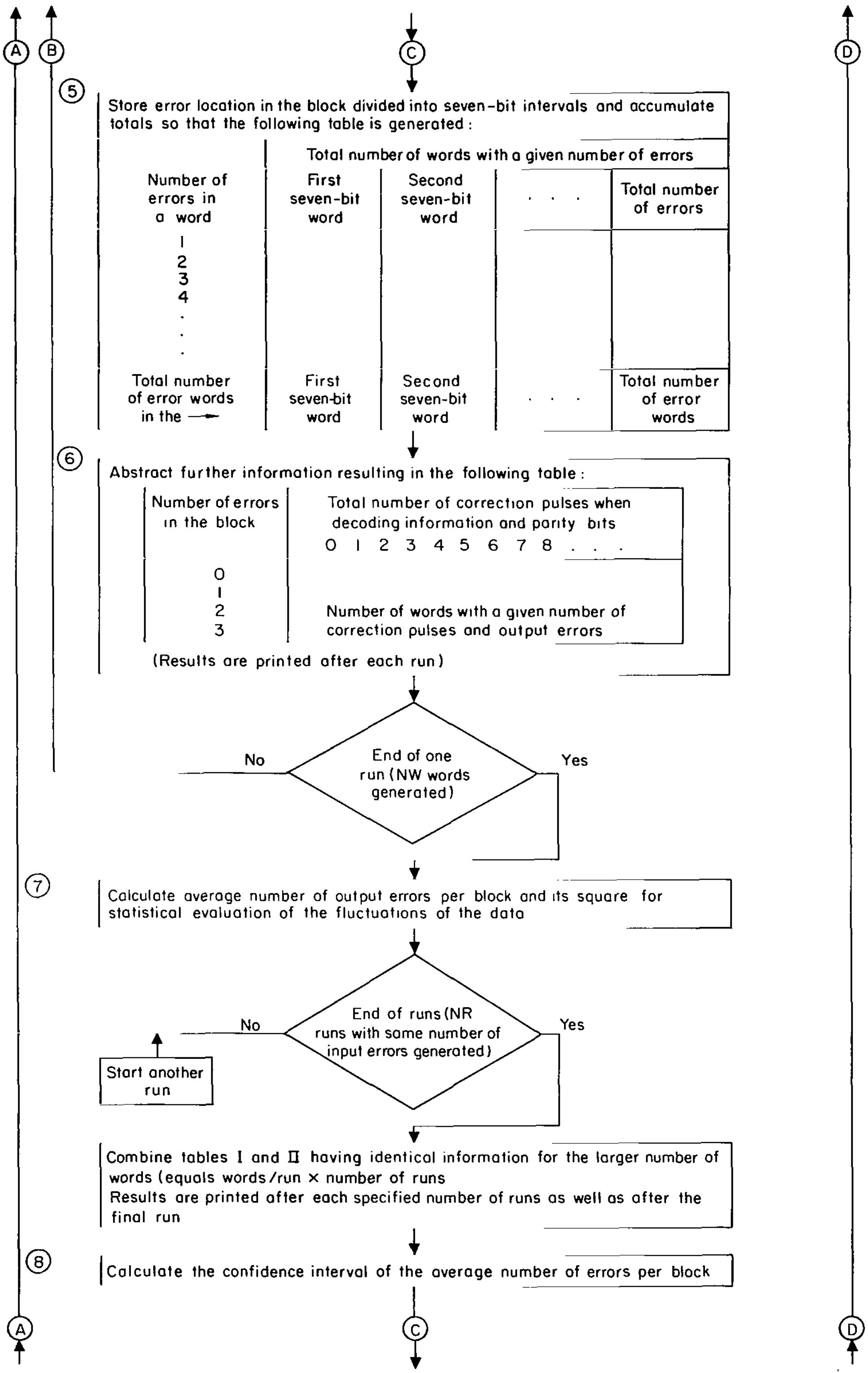
Generate a given number of errors, N, in the constraint length of the code under investigation with the errors randomly selected (assign I to error bits)

Select corresponding I and D levels randomly according to their probability of occurrence from the distribution of correct bits and from the distribution of error bits separately, depending on the reception of a correct bit and error bit, respectively

Apply the decoding equations for the type of code and type of decoding to be tested

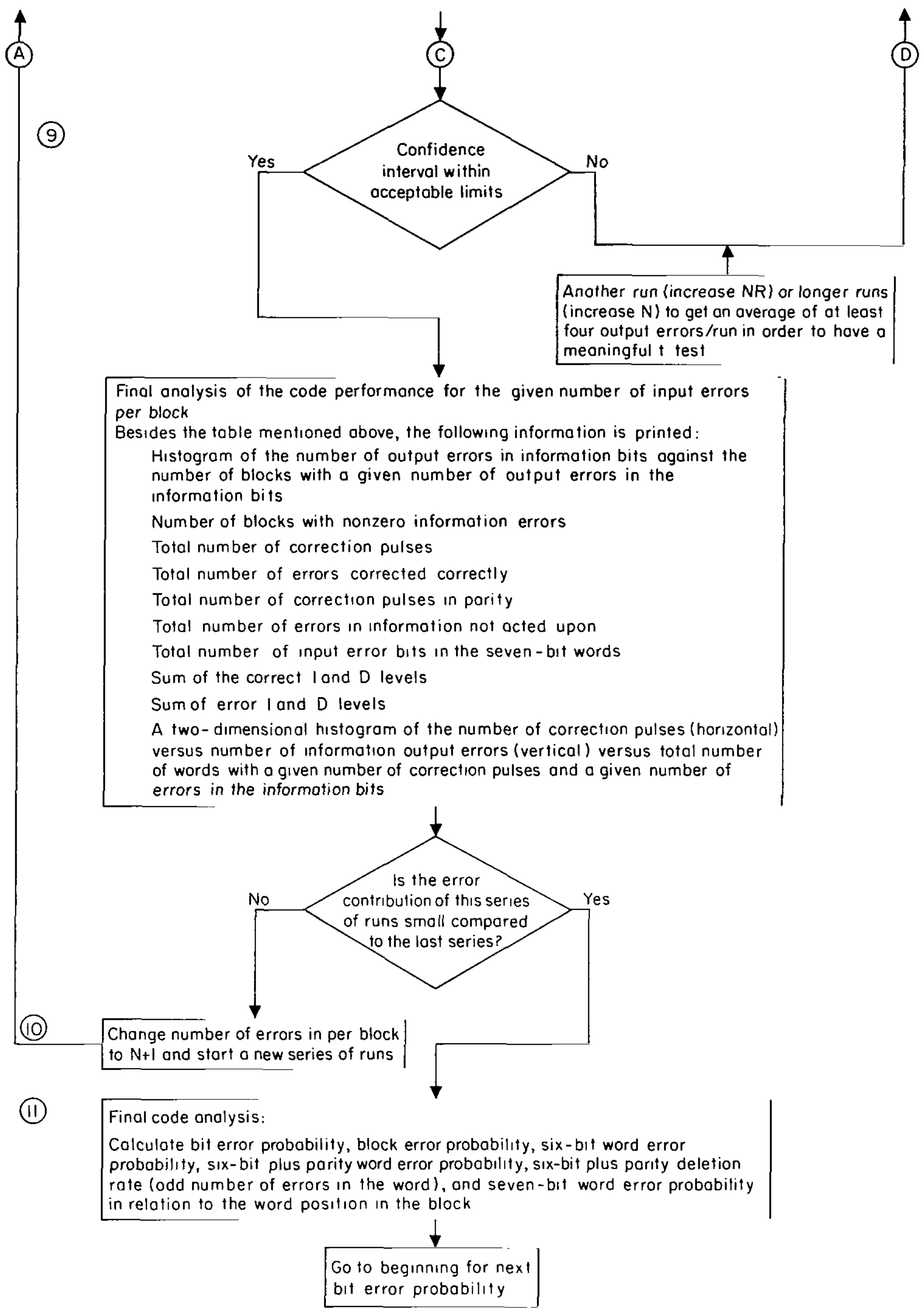
Summarize the effect of the decoding strategy:
(1) Number of correctly corrected errors
(2) Number of errors not corrected (no action)
(3) Number of correction pulses in information bits
(4) Number of errors generated in information bits (by false action)
Sum the above quantities to totals for a run of blocks with the same number of input errors and average error probability

Print out input error positions and output error positions as well as the above quantities calculated for the individual block


A
B
C
D
5
Store error location in the block divided into seven-bit intervals and accumulate totals so that the following table is generated:
6
Abstract further information resulting in the following table:
(Results are printed after each run)
End of one run (NW words generated)
No
Yes
7
Calculate average number of output errors per block and its square for statistical evaluation of the fluctuations of the data
End of runs (NR runs with same number of input errors generated)
No
Yes
Start another run
Combine tables I and II having identical information for the larger number of words (equals words/run × number of runs
Results are printed after each specified number of runs as well as after the final run
8
Calculate the confidence interval of the average number of errors per block
A
C
D

	Total number of words with a given number of errors			
Number of errors in a word	First seven-bit word	Second seven-bit word	. . .	Total number of errors
1				
2				
3				
4				
.				
.				
.				
Total number of error words in the →	First seven-bit word	Second seven-bit word	. . .	Total number of error words

Number of errors in the block	Total number of correction pulses when decoding information and parity bits 0 1 2 3 4 5 6 7 8 . . .
0	
1	
2	Number of words with a given number of correction pulses and output errors
3	



Confidence interval within acceptable limits

Another run (increase NR) or longer runs (increase N) to get an average of at least four output errors/run in order to have a meaningful t test

Final analysis of the code performance for the given number of input errors per block

Besides the table mentioned above, the following information is printed:

Histogram of the number of output errors in information bits against the number of blocks with a given number of output errors in the information bits

Number of blocks with nonzero information errors

Total number of correction pulses

Total number of errors corrected correctly

Total number of correction pulses in parity

Total number of errors in information not acted upon

Total number of input error bits in the seven-bit words

Sum of the correct I and D levels

Sum of error I and D levels

A two-dimensional histogram of the number of correction pulses (horizontal) versus number of information output errors (vertical) versus total number of words with a given number of correction pulses and a given number of *errors in the information* bits

Is the error contribution of this series of runs small compared to the last series?

Change number of errors in per block to N+I and start a new series of runs

Final code analysis:

Calculate bit error probability, block error probability, six-bit word error probability, six-bit plus parity word error probability, six-bit plus parity deletion rate (odd number of errors in the word), and seven-bit word error probability in relation to the word position in the block

Go to beginning for next bit error probability

(1) As mentioned in the Theory of Operation section the code performances depend only on the error locations and not on the sequence of zeros and ones representing the information. For this reason an all 0 information bit sequence was used. Errors are then easily identified by ones.

(2), (3) The method of generating errors has been described elsewhere (ref. 2).

(4) This information is collected to give clues as to how codes fail.

(5) This is an examination of the error distribution throughout the block.

(6) This is a test of the correlation between number of output errors and number of correction pulses. If a large number of correction pulses occur, this implies there are errors in the block and this block could be deleted.

(7), (8), (9) These were hand simulated. The activity described in these blocks serves the purpose of obtaining sufficiently long data runs. The total run is partitioned so that a meaningful t-test can be applied, and so that the 95-percent confidence interval is not greater than ±1/2 of the measured parameter for potential error-causing situations of significant probability of occurrence. This interval might seem large, but when all experiments are combined (with different numbers of errors per block and different input error probabilities), statistical errors tend to cancel. Also, knowing the average error rates within a factor of 2 is quite adequate when one considers how small a change of signal-to-noise ratio changes the error rate by a factor of 2.

(10) For each type of potential error-causing situation, a summary of the performance is printed. In addition, the input bit stream is analyzed to assure that the overall distribution of levels is truly gaussian.

(11) Significant information is added together, weighted by the probability of occurrence of the different output error-causing events.

The flow chart for the decoding simulation of the convolutional code is similar except that it also contains other tabulations of interest which will be discussed in the section on test results. For high error probabilities it is more efficient and accurate to simulate a gaussian channel directly rather than use only potential error-causing events. Data points obtained in this manner have been shown as solid symbols on the appropriate figures.

TEST RESULTS FROM HARDWARE ENCODING/DECODING SYSTEMS

A hardware model of the (24,12) majority equation decoder was used to determine the performance of this type of coding scheme on the binary symmetric channel. The test simulated this channel. Data consisted of a pseudo random sequence generator, and errors were introduced randomly by synchronously gating a threshold detector from a gaussian noise source.

Experimental data have been obtained on this code and plotted in figure 4. Channel bit error probability, P_e, is plotted against the output error probability from the decoder. Each point represents a sample size of 500,000 bits. For the (24,12) double error-correcting code, for example, a channel error probability, $P_e = 5\times10^{-3}$, gives a decoded output bit error rate of $P_{bit} = 10^{-5}$. Error correction performance from a mechanized (73,45) block code has been obtained and plotted in figure 5. The curves in figures 4 and 5, however, are drawn from the computer simulation data to show the excellent agreement.

TEST RESULTS FROM THE COMPUTER EXPERIMENT

The test results are presented and discussed in two stages. First, the results for the individual codes are shown and second, the codes are compared to each other and to no coding. For ease of comparison, the second step necessitates superimposing some of the performance curves already given.

Performance of the (15,7) Block Code

Figures 6 and 7 represent the raw data for the code. These figures show on a bit and on a word error probability basis that APP decoding has a large error reduction capability beyond that of unweighted (majority decision) decoding, the reason being that APP decoding corrects many of the otherwise error-causing situations. On the other hand, while majority decoding always corrects two errors in the block, APP decoding will sometimes make output errors in such cases, which accounts for as many as 20 percent of the errors made at an input error probability of 3 percent. For higher error probabilities, even more of the double error words will be incorrectly decoded, which means that eventually the curves for APP and majority decoding intersect. However, this happens at an error rate at which neither decoding scheme is of any use for this code. The curves have been translated into plots of error rates versus E_b/N_o in dB. Figure 8 shows that majority decoding has a very limited gain of only 1 dB at a bit error probability of 10^{-5}, and the coding gain reduces rapidly as E_b/N_o decreases. APP decoding tends to keep the coding gain constant over the region of interest, with a gain of about 2.6 dB over no coding. Note that there is little difference in performance between full APP and hard decision feedback decoding.

Since the simple seventh bit parity check code seems to work so well when a small deletion rate is permitted, its effect when it is concatenated with the different codes was investigated. As figure 9 shows, the performance, which includes the rate loss of 0.67 dB, is slightly worse by 0.2 dB. It will be seen that this condition prevails for all the codes tested. The data are included since it might be useful to employ the seventh bit parity check code before further encoding in a spacecraft, so that if the encoder should fail it could be switched out, while the parity check would continue to make the received data useful. Deletion rate with coding is negligible compared to that of the seventh bit parity check code, and therefore is not shown. While hard decision decoding performs slightly better than full APP when the parity check code is not included, the situation is reversed when it is included. This is explained by the fact that compared to hard decision feedback, full APP generates significantly more single error words while producing fewer words with more than one error.

It will be remembered that in the cyclic block codes, it is possible to decode the parity bits first, and then the information bits. This would be a useful technique for the full APP decoding, since the word error probability for the second seven-bit word is about half that of the first word. The hard decision feedback decoding has the same word error probability for the two seven-bit words, and it would be wasteful of computer time to decode the parity check bits also.

Performance of the (73,45) Block Code

The raw data and the data converted to the performance curves are shown in figures 10 through 13. Most of the remarks made for the (15,7) apply also to the (73,45) code, except that the performance of the latter is considerably better. There is also a clear advantage compared to the seventh bit parity check code, at least as far as APP decoding is concerned. For the 3-percent error probability data, the full APP decoding is clearly superior to hard decision APP decoding. It was, therefore, the only method used for the 1-percent runs since they must be longer than the 3-percent runs to get statistically significant data.

Figure 14 shows the word error probability normalized to the first word as a function of the word position in the block for various decoding systems. The figure illustrates the effect of the different types of feedback. Data points for majority decision decoding in which the original bit decision is fed back (dotted connection in figs. 31(b) and 31(c) in appendix A) are not shown since they were not measured. However, it is clear that the corresponding curve (fig. 14) should be the horizontal line I, since the word position in the block will not affect any coding decisions. Curve II shows the beneficial effect of hard decision feedback on majority decoding. This effect becomes even more pronounced for APP decoding with hard decision feedback (curve III). But the improvement levels off rapidly. Curve IV shows full APP decoding for identical data. As expected, it begins in a manner similar to the one for hard decision APP feedback, but the improvement continues all through the block. Consequently, it is worthwhile to decode the parity check bits first and then the information bits. Figures 12 and 13 give the performance for decoding all 73 bits. The additional improvement from the extra effort of decoding the parity bits first can be estimated from figure 14.

Performance of the (24,12) Convolutional Code

The raw data and the data converted to performance curves are shown in figures 15 through 18. In spite of its higher error correction capability the performance of the (24,12) convolutional code is not much better than that of the (15,7) block code for majority logic decoding. Figure 19 shows some of the details of the error dispersion. The decoder slides into a bit stream containing a potential error-causing situation within a constraint length of 12 information and 12 parity bits. Figure 19(a) shows that some output error patterns are 35 bits long. A comparison of figures 19(a) and 19(b) shows that the alarm indeed reduces the average length of the error patterns. Especially, it removes the curious peak at a distance of 14 bits. Even though the alarm reduces the average number of errors per potential error-causing situation, the actual number of output error packets is increased. For APP decoding the alarm is not useful. The dispersion for APP decoding, as shown in figure 19(c), is much larger than for majority decision decoding, but is compensated for by the much smaller number of output error packets per error-causing situation. Another fact worth noting is that the shape of the dispersion curves seems to be independent of the number of input errors per potential error-causing situation.

Performance of the (44,22) Convolutional Code

The raw data and the data converted to performance curves are shown in figures 20 to 23. A comparison of figures 13 and 23 shows that for the convolutional code there is a 0.4 dB difference in performance between concatenating the seventh bit parity check code and not concatenating, while for the (73,45) block code there is only a very small difference. This is explained by the fact that errors occur more in bunches in the (44,22) convolutional code than in the (73,45) block code; consequently, the ratio of double error words to single error words is relatively high. As far as error dispersion is concerned, the remarks made for the (24,12) code hold, except that the average dispersion has also increased with the increased constraint length (see fig. 24).

Summary of Test Results

The performance of the codes is summarized in figure 25. To illustrate the type of results obtained, assume the following condition: The six-bit word error rate is not to exceed 10^{-5}. Then, for no coding the E_b/N_0 required is about 10.4 dB, while the undetected word error rate for the simple parity check code requires only 7.8 dB, which, of course, includes a 4.8×10^{-3} word deletion rate (due to parity tagging). If one considers this deletion rate negligible, then, for majority decoding, only the (73,45) code shows a moderate improvement of 0.8 dB over the seventh bit parity check code. For APP decoding all codes show an improvement: 0.2 dB for the (15,7), 1.0 dB for the (24,12), 1.5 dB for the (44,22), and 2.1 dB for the (73,45) code. For all codes investigated, APP decoding shows about a 1.5 dB gain over majority decision decoding.

Figure 26 helps to visualize the error bunching that occurs after decoding. The figure is essentially a plot of double error seven-bit words versus single error seven-bit words. The steepest curve is that of the binomial channel (no coding). The remaining curves show that there are many mory double error words after decoding than a binomial distribution of errors would suggest. The least error clustering occurs in the (73,45) code, when it is full APP decoded. Consequently, except for the (73,45) code, performance is generally degraded when the seventh bit parity check code is superimposed on the other codes.

ERROR DETECTION FOR BLOCK AND CONVOLUTIONAL CODES

Error detection methods discussed in this section have been tested only sufficiently to prove that they will be of value when an extremely low error rate is required with a constant computation rate.

Suggestions for Further Error Detection for Block Codes

The (73,45) code was the most powerful investigated. It gave an advantage of more than 1 dB over the simple parity check code, with the additional advantage of no deletion. However, it is legitimate to ask if a still lower error probability can be achieved at a cost of deleting a moderate number of words likely to be in error. Some preliminary work regarding this question will be presented here.

The most obvious answer is to try to concatenate two codes. The simple seven-bit parity check code was concatenated with the four codes investigated in this report, and, in general, overall performance decreased slightly. Forney (ref. 3), however, proved that concatenating two powerful codes can improve overall coding gain but at the cost of increasing complexity at the encoder as well as at the decoder. To hold spacecraft complexity at a minimum, it would be desirable to achieve a coding gain at the cost of increasing earth-based decoder complexity alone. A simple example was given in reference 2, where the trade offs between error rate and deletion rate were emphasized rather than the coding gain, which was small. To repeat the theme, space scientists often choose to disregard information with a higher error probability in order to increase the reliability of their data. The same method seems applicable to full APP decoding, which leaves, in the c_j register, a sequence of bit error estimates after decoding. For several data runs the estimated block error probabilities were calculated for each block from c_j. A typical result is shown here. In 800 blocks with a potential error-causing situation of 7 errors per block, and a 1.5 percent average bit error probability, 83 error blocks occurred. If a block error probability threshold of 10^{-3} were chosen, all the error blocks would have been deleted at the cost of deleting 50 percent of the blocks with 7 input errors which were correctly decoded. It must be remembered that at the 1.5-percent input error probability, only 1 percent of the blocks have potential error-causing situations. To get an idea of the small deletion rate, 1,000 blocks containing 42,000 data words were decoded with one input error each. Each of the calculated output block error probabilities was smaller than 10^{-5}, indicating that virtually no words are removed from blocks with few input errors.

The deletion rate for any chosen threshold can be estimated by determining the distributions of the word error probabilities, given a certain number of output errors. For high input bit error probabilities these distributions can be simply obtained by generating the error stream in the more natural way, where the number of errors in the block are not predetermined. This was done for a data run with a 7-percent bit error probability, which strains the performance of the nominally four error correcting code. The cumulative distributions are shown in figure 27. Unfortunately, the distributions are not sufficiently well approximated by normal distributions, or one could now calculate deletion versus error rate for all possible thresholds. However, from the tabulated output, for a deletion threshold with probability of no error equal to 0.4, 99.5 percent of the blocks in error after decoding were deleted at a deletion rate of 29.4 percent. If one is willing to accept this deletion rate as a simple rate loss, 10 log(1/(1-deletion rate)) = 1.5 dB, the performance of the code would have improved by 3.3 dB. This might well be a useful technique of gaining data from space probes when switching to a lower transmission rate is not possible, and when the error rate would have otherwise been considered unacceptable.

Another attempt to detect error blocks was somewhat disappointing but provided valuable insight into the performance of the decoder. It was noticed that blocks in error seemed to fall into two groups, either they had very few errors or a large number of errors. It was thought that by applying majority decoding after the APP decoding of the complete block, which, of course, would eliminate errors in all the blocks with less than four errors, one would easily detect blocks containing many errors by the behavior of the sum of the A equations for each decoded bit. But the majority decoder treated error blocks, which had 10 errors after APP decoding, exactly as if no errors were present; that is, each A equation is equal to zero. This means that the decoder would accept the error stream exactly as it was. It has therefore been demonstrated experimentally, that when the decoder fails, it tends to decode to the nearest correct neighbor of the actually transmitted code word. This conclusion is further strengthened when one observes that 9 and 11 error sequences from the APP decoder are treated as a 1 error sequence by the following majority decoder. Most of the time the majority decoder will therefore add another error to the others in the above-mentioned cases. While the above method is not satisfactory for error detection and correction, it would tend to reduce the block error probability to one-third of what it was before, and it would reduce the bit error probability by about half if it were simply used for error correction.

Combining the two error detection schemes may result in virtually error-free decoding. If, from the error probabilities of all bits in a block, one calculates the block error probability after decoding and then removes all blocks that have a higher error probability than a given threshold, one eliminates virtually all blocks with large numbers of output errors. The remaining blocks are then decoded by the majority decoder, and all remaining error blocks with less than four errors are corrected.

An Error Detection/Deletion Strategy for Convolutional Codes

The performance of majority decoding of convolutional codes was shown to be somewhat disappointing. Because of error clustering and rate loss, concatenation of the single parity check code did not improve the overall performance. Hence, a detection/deletion scheme that will take advantage of the error clustering is desirable.

The scheme tested in our simulation is based on the fact that the S equations can be combined in more than one way to result in a set of orthogonal equations. For instance, Massey's A equations for the (24,12) code are as follows:

$$\left.\begin{aligned} A_1 &= S_1 \\ A_2 &= S_7 \\ A_3 &= S_8 \\ A_4 &= S_9 \\ A_5 &= S_2 + S_4 + S_{11} \\ A_6 &= S_5 + S_9 + S_{12} \end{aligned}\right\} \qquad (13)$$

This set is to be compared with the set of equations used in the hardware decoder (previously discussed) and also in all our computer experiments (see eq. (A15)). The important difference between these two sets of equations is that the error terms of previously decoded bits (terms with negative subscripts in equations (A17) to (A20)) are distributed quite differently among the A equations. The decoding/deletion scheme is now as follows:

Run two decoders with different sets of A equations in parallel. Delete all seven-bit words when the outputs from the two sets of decoding equations differ by at least one position per word. Delete also the two words preceding the first detection of an error. The reason for the last part of the strategem is that there is a tendency for decoding errors to occur at the same positions at the beginning of an error packet. This is understandable, since fewer error terms are involved at the beginning of an error packet. These initial errors then cause random generation of further errors. The effectiveness of the last part of the strategem is shown by the following examples.

Since APP decoding is of more interest, the effectiveness of the above strategy was tested with the more powerful decoding method. Runs with high error probabilities were chosen since these data would be of very limited use without deletion. For the (24,12) code at a bit error probability of 1.5 percent, 12 error words remained after decoding. The deletion strategy caught all errors at the cost of deleting 1.5 percent of all words. The cost is high because Massey's orthogonalization by itself results in a higher error rate than that of equations (A15), and a word is deleted, of course, when either decoder makes an error. For a 7-percent input error probability there were 349 error words in 12,000 seven-bit words. With the deletion scheme, only 25 error words remained at a cost of deleting 16.6 percent of all words. Without the last part of the strategem 116 error words would have remained.

For the (44,22) code only Massey's orthogonalization was available. Therefore, a second set of A equations was developed (appendix A). For an input error probability of 5 percent, there were 145 error words in 12,000 data words. With the strategem only 3 error words were left after 446 data words were removed (22 error words would have been left without the second part of the strategem).

Depending on the type of decoder, the above scheme doubles either decoder equipment or computer time. A simplified error detection circuit is shown in figure 28. Although the circuit has not been evaluated, its performance is believed to be similar to or better than that of the parallel decoders. The circuit operates best by connecting to threshold element 1 the set of A equations, which if used alone would result in the better decoder. This threshold element alone has control of complementing the S register when it detects an error. Threshold element 2, with its different set of A equations, only serves to detect error packets for comparison. Thus, only error packets caused by the better decoder will be deleted. The decoder will simply mark bits not checked by the comparator circuit and leave the deletion up to the user.

CONCLUDING REMARKS

The work reported forms a basis for comparing the coding techniques evaluated here with other coding and decoding techniques. The decoding methods described are characterized by a constant computation rate per decoded bit, independent of the channel noise. The optimum decoding strategy for these codes, maximum likelihood decoding, has not been investigated, since computation time per bit is too large to be practical.[1] For convolutional codes, an algorithm for sequential decoding[2] is available, which closely approaches the performance of maximum likelihood decoding. However, sequential decoding has a variable computation rate, which might make it ineligible for high bit rate coding. Sequential decoding of convolutional codes is presently under investigation with the same restrictions imposed on the codes as on those in the present report, namely, short constraint length and high rate. The most powerful of these codes investigated (ref. 6) is a (50,25) code which has a gain over the (73,45) APP decoded code of 1.9 dB over a range of word error probabilities from 10^{-3} to 10^{-5}.

Ames Research Center
National Aeronautics and Space Administration
Moffett Field, Calif., 94035, Sept. 29, 1967
125-23-02-00-00-21

[1]In maximum likelihood decoding one calculates for each possible message the probability that a given received message is indeed that message. Then the message with the highest probability of having been sent is chosen. For example, the (73,45) code has 2^{45} possible messages. For convolutional coding one would have to truncate the bit stream, and the number of possible messages is 2^N, where N is the number of information bits in the truncated bit stream. Again the number of possible messages and associated computing time is enormous.

[2]Sequential decoding algorithm is a tree search procedure for seeking the highest probability message in an efficient manner by pruning less likely branches.

APPENDIX A

MAJORITY DECODING AND APP DECODING OF A (24,12) THREE ERROR CORRECTING CONVOLUTIONAL CODE

Figure 29 shows the structure of the triple error correcting encoder and decoder. The encoding operation is the following: Information bits are fed into a 12-stage shift register sequentially, as they come from the source. Taps on the shift register feed the most recent bit, i_{12}, and the five older bits in the register, i_1, i_2, i_3, i_5, and i_6, to a modulo 2 adder (parity generator). The output of this adder is zero if there are an even number of ones in its input; its output is one if there are an odd number of ones in its input.

The output of the adder is the parity bit p_{12}, associated with information bit i_{12}, and these are both ready for transmission over the channel. (In the actual equipment they are transmitted in sequence.) It should be noted that it is the most recent information bit that is transmitted, along with its parity, so there is negligible delay in the encoder. As soon as i_{12} and p_{12} are transmitted, i_{13} is read into the encoder, all bits shifting one place to the right, and $p_{13} = i_2 \oplus i_3 \oplus i_4 \oplus i_6 \oplus i_7 \oplus i_{13}$ and i_{13} are ready for transmission. Encoding continues in this fashion.

At the decoder, received information is fed into a duplicate of the encoder, but received parity is also fed into the mod 2 adder. The output of the adder at this moment is denoted by S_{12}.

By definition, at the encoder, with modulo 2 addition always understood,

$$p_{12} = i_1 \oplus i_2 \oplus i_3 \oplus i_5 \oplus i_6 \oplus i_{12} \tag{A1}$$

Now at the decoder the received forms of these information bits are added to the parity bit p_{12}; therefore, if the bits were correct, p_{12} would be added to itself, thus getting zero. If one of the bits were wrong, S_{12} would equal one; if two were wrong, S_{12} would be zero, and so on. Therefore S_{12} is independent of the actual transmitted values of the seven bits of which it is composed and depends only on the number of errors in the received versions of those bits, being one if the number of errors is odd, and zero if the number of errors is even.

In order to analyze this precisely, denote the received bits by $i'_k = i_k \oplus e^i_k$ and $p'_k = p_k \oplus e^p_k$ (the error term, e^i_k or e^p_k, is zero if the corresponding bit was received correctly, and one if it was received incorrectly); then

$$S_{12} = e^i_1 \oplus e^i_2 \oplus e^i_3 \oplus e^i_5 \oplus e^i_6 \oplus e^i_{12} \oplus e^p_{12} \tag{A2}$$

Now, at the moment shown in figure 29, i'_1 is at the right end of the decoder chain, and the function of the decoder at this instant is to decide whether i'_1 is correct or not, and change it if it is wrong. Another pair, i'_{13} and p'_{13}, will then enter the decoder, i'_2 will be at the right end, and the same process will decode i'_2. Decoding will continue in this fashion. In order to show how i'_1 is decoded, make the preliminary assumption that there have been *no* errors before i'_1 and p'_1.

With this assumption it can be seen that when S_1 was formed 12 bits ago, the only bits involved that might have been wrong were i_1 and p_1, so that

$$S_1 = e^i_1 \oplus e^p_1 \tag{A3}$$

Similarly, up to S_6

$$S_2 = e^i_2 \oplus e^p_2 \tag{A4}$$

$$S_3 = e^i_3 \oplus e^p_3 \tag{A5}$$

$$S_4 = e^i_4 \oplus e^p_4 \tag{A6}$$

$$S_5 = e^i_5 \oplus e^p_5 \tag{A7}$$

$$S_6 = e^i_6 \oplus e^p_6 \tag{A8}$$

When the next bit enters the shift register, i_1 is in position 7 of the register and will be added modulo 2 to the seventh parity and information bits. Thus,

$$S_7 = e^i_7 \oplus e^p_7 \oplus e^i_1 \tag{A9}$$

Again all bits shift by one and the modulo 2 addition gives

$$S_8 = e^i_8 \oplus e^p_8 \oplus e^i_1 \oplus e^i_2 \tag{A10}$$

Similarly,

$$S_9 = e^i_9 \oplus e^p_9 \oplus e^i_2 \oplus e^i_3 \tag{A11}$$

$$S_{10} = e^i_{10} \oplus e^p_{10} \oplus e^i_1 \oplus e^i_3 \oplus e^i_4 \tag{A12}$$

$$S_{11} = e^i_{11} \oplus e^p_{11} \oplus e^i_1 \oplus e^i_2 \oplus e^i_4 \oplus e^i_5 \tag{A13}$$

and as in equation (A2)

$$S_{12} = e^i_{12} \oplus e^p_{12} \oplus e^i_1 \oplus e^i_2 \oplus e^i_3 \oplus e^i_5 \oplus e^i_6 \tag{A14}$$

Another way of seeing how the S equations are developed is to assume that i'_1 has been correctly decoded into i^o_1, which means that the value of e^i_1 is known

$$e^i_1 = 0 \text{ if } i'_1 = i_1 \quad \text{correctly received}$$

$$e^i_1 = 1 \text{ if } i'_1 \neq i_1 \quad \text{error received}$$

Therefore, S_{12} can be complemented if $e_1^i = 1$ to remove the effect of e_1^i. In any case,

$$S_{12} = e_1^i \oplus e_2^i \oplus e_3^i \oplus e_5^i \oplus e_6^i \oplus e_{12}^i \oplus e_{12}^p$$

Now the S equation register is advanced one step to the right, thus lowering the indices by 1

$$S_{11} = e_1^i \oplus e_2^i \oplus e_4^i \oplus e_5^i \oplus e_{11}^i \oplus e_{11}^p$$

Consecutively removing the term e_1^i (if it is present in the S equation) and shifting down by 1 will generate the remaining S equations.

The 12 S equations are now combined to give the following set of equations:

$$\left.\begin{aligned}
A_1 &= S_1 = e_1^i \oplus e_1^p \\
A_2 &= S_2 + S_8 = e_1^i \oplus e_8^i \oplus e_2^p \oplus e_8^p \\
A_3 &= S_7 = e_1^i \oplus e_7^i \oplus e_7^p \\
A_4 &= S_9 + S_{12} = e_1^i \oplus e_5^i \oplus e_6^i \oplus e_9^i \oplus e_{12}^i \oplus e_9^p \oplus e_{12}^p \\
A_5 &= S_4 + S_{10} = e_1^i \oplus e_3^i \oplus e_{10}^i \oplus e_4^p \oplus e_{10}^p \\
A_6 &= S_5 + S_{11} = e_1^i \oplus e_2^i \oplus e_4^i \oplus e_{11}^i \oplus e_5^p \oplus e_{11}^p
\end{aligned}\right\} \qquad \text{(A15)}$$

To simplify the notation for APP decoding, the above set of equations is written in abbreviated form as

$$A_k = e_1^i \oplus \sum_{j=1}^{n_i} e_{\alpha_j}^{(\beta_j)} \qquad \text{(A16)}$$

where n_i is the number of error terms in the given equation minus one. All addition is modulo 2. The important thing to note about this set of equations is that it is "orthogonal in i_1"; that is, e_1^i occurs in each equation, and no other term occurs more than once in the whole array. This orthogonality permits use of the following decoding rule to determine e_1^i; that is, to determine if i_1' is right or wrong. *If more than three of the six equations are equal to 1, then* i_1' *is wrong (*$e_1^i = 1$*) and must be changed.*

It will be shown that this rule will decode i_1' correctly if no more than 3 of the 24 bits $i_1 \ldots p_{12}$ are wrong. (In fact, only 22 bits are concerned, since p_3 and p_6 are not in the equations.) To verify that the rule works is simple. If i_1' is right and one, two or three other bits are wrong, there can be at most 3 ones among the equations, since each of the other bits occurs in only one place. But 3 or less ones is not enough to change i_1'. If i_1' is wrong, it makes all six equations one, and if i_1' and another bit or two are wrong, the other bits restore one or two of the equations to zero but leave at least 4 ones. Each of these cases gives more than 3 ones and thus leads to correcting i_1'.

Thus i_1' will be decoded correctly if not more than 3 of the 22 bits are wrong. Actually *some* combinations of more than 3 errors will also be decoded correctly. Now, when it is decided that i_1' is wrong, complement it and the values of S_7, S_8, S_9, S_{10}, S_{11}, and S_{12}. This removes the e_1^i term. Thus, before a new pair of bits is entered, the effect of the i_1' error has been eliminated, and for decoding i_2, the decoder is in the same state it would have been in if i_1' had been correct. As far as the decoder is concerned, there have been no previous errors.

The S register resetting operation is called hard decision feedback. Assume that the decoded bit is always decoded correctly. If e_1^o is called the possible error in the output bit, then S_{12} after decoding should be rewritten as

$$S_{12} = e_1^o \oplus e_2^i \oplus e_3^i \oplus e_5^i \oplus e_6^i \oplus e_{12}^i \oplus e_{12}^p \tag{A17}$$

and shifting by one space, S_{11} actually becomes

$$S_{11} = e_0^o \oplus e_1^i \oplus e_2^i \oplus e_4^i \oplus e_5^i \oplus e_{11}^i \oplus e_{11}^p \tag{A18}$$

likewise,

$$S_{10} = e_{-1}^o \oplus e_0^o \oplus e_1^i \oplus e_3^i \oplus e_4^i \oplus e_{10}^i \oplus e_{10}^p \tag{A19}$$

similarly down to S_1:

$$S_1 = e_{-10}^o \oplus e_{-9}^o \oplus e_{-8}^o \oplus e_{-6}^o \oplus e_{-5}^o \oplus e_1^i \oplus e_1^p \tag{A20}$$

In other words, the S equations actually depend on previously decoded bits. This explains why once a decoding error is made, more errors tend to be made even though only correct bits may subsequently enter the decoder.

Figure 30 shows the APP decoder which includes an analog circuit for computing the weighting factors. The weighting factors for the gaussian channel are computed as follows: The incoming voltage levels V for each bit are passed through a nonlinear amplifier that has an output $-\log_e\{\coth[(\overline{V}/\sigma^2) \cdot |V|]\}$ where $\overline{V}$ and σ^2 are the average received voltage and its variance. The top analog shift register has as inputs the set of c^i and the bottom one has as inputs the set of c^p. The kth analog adder above the analog shift registers has as inputs the set of c's to form C_k. The output of this adder is fed to a nonlinear device that has an output of $2\log_e[\coth(x/2)]$ for an input of x. This is the weighting factor for the parity check A_k. The threshold T is formed by taking half the sum of the weighting factors as called for by equation (8). Since the analog circuit computes the correct set of weights and the threshold T at any instant, it is combined with the decoder of figure 29 to give a complete APP decoding circuit.

APPENDIX B

MAJORITY DECODING AND APP THRESHOLD DECODING OF THE (15,7) BOSE-CHAUDHURI CODE

Decoding the (15,7) Bose-Chaudhuri code will be described as an example of threshold decoding of a block code. This is a type of cyclical block code that corrects two errors in a block. Figure 31 shows the structures of the encoder and the decoder. The encoding operation is as follows. Information bits are loaded in parallel into a seven-stage shift register. Taps on the shift register feed the most recent bit i_1, along with i_5 and i_7 into a modulo 2 adder, and the output is transmitted as well as fed back into position 7 of the shift register. The shifting and transmission continue through information bit i_7. The process is continued until eight parity bits are generated, namely,

$$i_k = i_{k-1} \oplus i_{k-3} \oplus i_{k-7}, \qquad k = 8, 9, \ldots, 15 \tag{B1}$$

After all 15 code digits have been shifted out, the 7-stage register once again contains the original information, i_1 to i_7.

At the decoder, received information is fed in parallel into a 15-stage shift register. Taps lead to four mod 2 adders to calculate the parity equations. Majority decoding, shown in figure 31(b), is ordinarily used. The equation

$$A_1' = i_5' \oplus i_7' \oplus i_8'$$

can be reduced to the transmitted information plus error terms

$$A_1' = i_5 \oplus e_5 \oplus i_7 \oplus e_7 \oplus i_8 \oplus e_8$$

but from equation (B1)

$$i_1 = i_5 \oplus i_7 \oplus i_8$$

thus

$$A_1' = i_1 \oplus e_5 \oplus e_7 \oplus e_8$$

Writing all parity equations in the same manner results in

$$\left.\begin{aligned} A_0' &= i_1 \oplus e_1 \\ A_1' &= i_1 \oplus e_5 \oplus e_7 \oplus e_8 \\ A_2' &= i_1 \oplus e_3 \oplus e_4 \oplus e_{12} \\ A_3' &= i_1 \oplus e_2 \oplus e_{10} \oplus e_{14} \\ A_4' &= i_1 \oplus e_9 \oplus e_{13} \oplus e_{15} \end{aligned}\right\} \tag{B2}$$

The majority decision rule is $i_1^o = 1$ if

$$\sum_{0}^{4} A_i' \geq 3 \tag{B3}$$

To produce a threshold decoder which performs exactly like the majority decoder (see fig. 31(c)), connections from i_1 are added. The first parity equation for this decoder becomes

$$\begin{aligned} A_1 &= A_1' \oplus i_1 \oplus e_1 \\ &= i_1 \oplus e_5 \oplus e_7 \oplus e_8 \oplus i_1 \oplus e_1 \\ &= e_1 \oplus e_5 \oplus e_7 \oplus e_8 \end{aligned}$$

Similarly,

$$\left.\begin{aligned} A_1 &= e_1 \oplus e_5 \oplus e_7 \oplus e_8 \\ A_2 &= e_1 \oplus e_3 \oplus e_4 \oplus e_{12} \\ A_3 &= e_1 \oplus e_2 \oplus e_{10} \oplus e_{14} \\ A_4 &= e_1 \oplus e_9 \oplus e_{13} \oplus e_{15} \end{aligned}\right\} \tag{B4}$$

The A_i are orthogonal in e_1 and are independent of the value of the transmitted bit. If there are an odd number of errors in the equation $A_i = 1$; and if there are an even number, $A_i = 0$. The decision rule with these parity check equations is:

If

$$\sum_{i=1}^{4} A_i \geq 3 \tag{B5}$$

then i_1' is assumed to be in error; hence, change it to produce i_1^o, the output bit, which still may contain an error $(i_1^o \neq i_1)$.

This decision rule is of exactly the same form as that for the convolutional code discussed earlier. If all the received bits were correct, each equation for A_i would add to 0. If one or two of these bits were wrong, at most two of the A_i would equal 1, the threshold would not be exceeded, and the bit would be correctly decoded.

After decoding the first bit i_1', the bit is corrected and circulated to position 15. Identical decoding rules are applied to decode i_2'. The process continues until all information bits are decoded. The formal proof for the above is given by W. H. Peterson (ref. 4). However, it is simple to advance the indices of the parity equations in steps of 1 (mod 16) and to show by substitution that the equations always form orthogonal parity checks for the bit which is presently in position "one" in the decoder. Even the parity bits can be decoded in this manner by shifting a full cycle, a fact which is shown to be useful in the discussion of the test results.

Figures 31(b) and 31(c) show dashed and solid feedback lines. It is not at all clear which connection should result in the better performance. Clearly, when the number of errors in the block does not exceed the error correction capability, the decoder will eliminate all errors with both types of feedback. For potential error-causing situations, it was found that the feedback of the decoded value will result in a lower average error probability. This is the only feedback connection of the binary shift register explored in connection with APP decoding and will be called binary hard decision feedback. The APP threshold decoder is shown in figure 32. The operation of this circuit is so similar to that in figure 30 that no detailed explanation should be necessary.

For completeness, the encoding and decoding equations for the (73,45) code are given. Encoding:

$$i_k = i_{k-45} \oplus i_{k-43} \oplus i_{k-35} \oplus i_{k-21} \oplus i_{k-20} \oplus i_{k-16} \oplus i_{k-9} \oplus i_{k-3} \qquad \text{(B6)}$$

where

$$i = 46, 47, \ldots, 73$$

and decoding:

$$\left.\begin{aligned}
A'_1 &= i'_1 \\
A'_2 &= i'_3 \oplus i'_{11} \oplus i'_{25} \oplus i'_{26} \oplus i'_{30} \oplus i'_{37} \oplus i'_{43} \oplus i'_{46} \\
A'_3 &= i'_9 \oplus i'_{23} \oplus i'_{24} \oplus i'_{28} \oplus i'_{35} \oplus i'_{41} \oplus i'_{44} \oplus i'_{72} \\
A'_4 &= i'_{15} \oplus i'_{16} \oplus i'_{20} \oplus i'_{27} \oplus i'_{33} \oplus i'_{36} \oplus i'_{64} \oplus i'_{66} \\
A'_5 &= i'_2 \oplus i'_6 \oplus i'_{13} \oplus i'_{19} \oplus i'_{22} \oplus i'_{50} \oplus i'_{52} \oplus i'_{60} \\
A'_6 &= i'_5 \oplus i'_{12} \oplus i'_{18} \oplus i'_{21} \oplus i'_{49} \oplus i'_{51} \oplus i'_{59} \oplus i'_{73} \\
A'_7 &= i'_8 \oplus i'_{14} \oplus i'_{17} \oplus i'_{45} \oplus i'_{47} \oplus i'_{55} \oplus i'_{69} \oplus i'_{70} \\
A'_8 &= i'_7 \oplus i'_{10} \oplus i'_{38} \oplus i'_{40} \oplus i'_{48} \oplus i'_{62} \oplus i'_{63} \oplus i'_{67} \\
A'_9 &= i'_4 \oplus i'_{32} \oplus i'_{34} \oplus i'_{42} \oplus i'_{56} \oplus i'_{57} \oplus i'_{61} \oplus i'_{68} \\
A'_{10} &= i'_{29} \oplus i'_{31} \oplus i'_{39} \oplus i'_{53} \oplus i'_{54} \oplus i'_{58} \oplus i'_{65} \oplus i'_{71}
\end{aligned}\right\} \qquad \text{(B7)}$$

The decision rule with these parity check equations is:

If

$$\sum_{i=1}^{9} A_i \geq 5$$

i'_1 is assumed to be in error; hence change it to produce i^o_1.

APPENDIX C

DERIVATION OF THE APP DECODING ALGORITHM

For optimum threshold decoding the rule is chosen to make the conditional probability that e_m is either 1 or 0 (i'_m is in error or not) a maximum given the set of J parity checks.

$$\Pr(e_m=V/\{A_i\}) \text{ maximum} \tag{C1a}$$

In other words, choose $e_m = 1$. Then

$$\Pr(e_m=1/\{A_i\}) > \Pr(e_m=0/\{A_i\}) \tag{C2}$$

From Baye's rule, equation (C1a) is rewritten

$$\Pr(e_m=V/\{A_i\}) = [\Pr(\{A_i\}/e_m=V) \cdot \Pr(e_m=V)]/\Pr(\{A_i\}) \tag{C3}$$

From the orthogonality on e_m of the A_i and the digit to digit independence

$$\Pr(\{A_i\}/e_m=V) = \prod_{i=1}^{J} \Pr(A_i/e_m=V) \tag{C4}$$

When equations (C3) and (C4) are substituted into (C2) and the common term $\Pr(\{A_i\})$ is cancelled:

$$\Pr(e_m=1) \prod_{i=1}^{J} \Pr(A_i/e_m=1) > \Pr(e_m=0) \prod_{i=1}^{J} \Pr(A_i/e_m=0)$$

or

$$\prod_{i=1}^{J} \Pr(A_i/e_m=1)/\Pr(A_i/e_m=0) > \Pr(e_m=0)/\Pr(e_m=1) \tag{C1b}$$

Define

$\Pr(e_m=1) = p_o = 1 - q_o$ = error probability of bit e_m (decoded bit)

$p_i = 1 - q_i$ = probability (odd number of 1's among the noise bits that are checked by A_i, exclusive of e_m)

then

$$\Pr(A_i=0/e_m=1) = \Pr(A_i=1/e_m=0) = p_i \tag{C5}$$

$$\Pr(A_i=1/e_m=1) = \Pr(A_i=0/e_m=0) = q_i \tag{C6}$$

Since A_i is either 1 or 0 each factor in equation (C1b) is in one of two forms:

If $A_i = 1$

$$Pr(A_i=1/e_m=1)/Pr(A_i=1/e_m=0) = q_i/p_i = (q_i/p_i)^{2A_i-1} \qquad (C7)$$

If $A_1 = 0$

$$Pr(A_i=0/e_m=1)/Pr(A_i=0/e_m=0) = p_i/q_i = (q_i/p_i)^{2A_i-1} \qquad (C8)$$

Equations (C7) and (C8) are used to rewrite (C1b) as

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

or

$$\prod_{i=1}^{J} (q_i/p_i)^{2A_i} > \prod_{i=o}^{J} q_i/p_i \qquad (C1c)$$

Since the e_j^β are independent random variables with

$$Pr\left(e_j^\beta=1\right) = 1 - Pr\left(e_j^\beta=0\right) = \gamma_j^\beta$$

it can be shown that (see ref. 1)

$$p_i = \frac{1}{2}\left[1 - \prod_{j=1}^{n_i} \left(1 - 2\gamma_j^\beta\right)\right] \qquad (C9)$$

where n_i is the number of error terms in A_i exclusive of e_m. Note that the product does not include a term for the encoded bit e_m; also note that equation (C9) is also true for $p_o = (1/2)\left[1 - \left(1 - 2\gamma_1^i\right)\right] = \gamma_1^i$. Now

$$\frac{q_i}{p_i} = \frac{1 + \prod_{j=1}^{n_i} \left(1 - 2\gamma_j^\beta\right)}{1 - \prod_{j=1}^{n_i} \left(1 - 2\gamma_j^\beta\right)} \qquad (C10)$$

Since summation is preferable to multiplication, define:

$$C_j^\beta = -\log_e\left(1 - 2\gamma_j^\beta\right) \qquad (C11)$$

Then

$$\frac{q_i}{p_i} = \coth\left(\frac{1}{2}\sum_{j=1}^{n_i} C_j^\beta\right) \qquad (C12)$$

and

$$\frac{q_o}{p_o} = \coth\left[(1/2)C_1^i\right] \tag{C13}$$

Taking logarithms on both sides of equation (C1c)

$$\sum_{i=1}^{J} 2A_i \log_e \frac{q_i}{p_i} > \sum_{i=0}^{J} \log_e \frac{q_i}{p_i} \tag{C1d}$$

and using equations (C12) and (C13) yields

$$\sum_{j=1}^{J} 2A_i \log\left[\coth\left(\frac{1}{2} \sum_{j=1}^{n_i} C_j^{\beta}\right)\right] > \frac{1}{2}\left\{2 \log\left[\coth\left(\frac{1}{2} C_1^i\right)\right] + \sum_{j=1}^{J} 2 \log\left[\coth\left(\frac{1}{2} \sum_{j=1}^{n_i} C_j^{\beta}\right)\right]\right\}$$

If the weights are defined, the threshold equation becomes

$$\sum_{i=1}^{J} w_i A_i > \frac{1}{2} \sum_{i=o}^{J} w_i \tag{C1e}$$

Further, if

$$T = \frac{1}{2} \sum_{i=o}^{J} w_i$$

the final form of the APP threshold decoding equation becomes

$$\sum w_i A_i > T \tag{C1f}$$

APPENDIX D

CALCULATION OF PERFORMANCE MEASURES FROM AN ENERGY EFFICIENCY STANDPOINT

To compare different coding schemes consistent measures of performance must be found, and finding them requires first a definition of the channel under consideration. The following normalizing assumptions are thought to be reasonable for interplanetary channels in the S-band, where the dominant source of noise is the receiver, and where provision for bit rate changes must be made because of the changing distance between the spacecraft and earth.

(1) An identical modulator, transmitter, and receiver system will be assumed for all coding schemes considered. (Thus rate 1/2 codes will not be penalized for their greater bandwidth requirement for the same information bit rate.)

(2) The system delivers a stream of bit log likelihood ratios to the decoder (matched filter reception); that is, the individual bit error probability can be calculated from the received voltage at the bit decision time.

(3) The transmission rate of information is assumed constant.

(4) The noise is additive white gaussian and changes in average power are very slow.

The received signal power varies very slowly with time. Its magnitude at any particular time will be denoted as S. The time required to send a single bit is T. The energy per transmitted bit is then ST joules. The noise power has a spectral density of N_o W/Hz. The total noise power in a band of f Hz is proportional to $N_o f$. Since the bandwidth is inversely proportional to the bit duration, the noise power in the transmission spectrum is also proportional to N_o/T. The noise energy per bit is therefore N_o, independent of the bit duration. A convenient normalized variable for the calculations to follow is the signal-to-noise energy ratio per transmitted bit

$$E/N_o = ST/N_o \qquad \text{(D1)}$$

It is well known (ref. 5) that under these conditions, the output of the matched filter receiver is a gaussian random variable y whose mean is positive or negative according to whether a 0 or 1 was sent in the corresponding time period.

Under these conditions the average bit error probability is

$$p_e = \frac{1}{\sqrt{2\pi}} \int_{\sqrt{\frac{2E}{N_o}}}^{\infty} e^{-\frac{x^2}{2}}\, dx \qquad \text{(D2)}$$

where E/N_0 is the linear signal-to-noise power ratio. For reference for large E/N_0 this is approximated within 1 percent by

$$p_e = \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{E}{N_0}}}{\sqrt{\frac{2E}{N_0}}} \quad \text{for } \frac{E}{N_0} > 5 \tag{D3}$$

The probability that a received bit i_j' is in error, given its received voltage level, is, from figure 33,

$$p_e = \frac{e^{-\frac{(|V|-\bar{V})^2}{2\sigma^2}}}{e^{-\frac{(|V|-\bar{V})^2}{2\sigma^2}} + e^{-\frac{(-|V|-\bar{V})^2}{2\sigma^2}}} \tag{D4}$$

Having characterized the channel, one can now proceed to describe the most often used performance criterion for codes, that of coding gain. In short, coding gain is the increase of the received power that would be required to achieve the same bit error probability for no decoding as for coding.

With the relationship of equation (D4), a graph can be drawn of bit error probability versus E/N_0 (fig. 34) corresponding to the case of no coding. In figure 34, the performance of a hypothetical code is also shown. It can be seen that the coding gain is reduced as E/N_0 becomes smaller, and eventually, the code performs worse than no coding. This is an unfortunate characteristic particularly of simple codes, where otherwise they would be most useful, they are least effective.

For the above comparison, the code performance curve is drawn in the following manner. It is assumed that the decoded bit versus input bit error probabilities have somehow been determined (1).[1] Then by means of figure 34 for a given input bit error probability (1) E/N_0 is read (2).

Since codes characteristically send more than one bit per information bit, a rate loss is added (2-3)

$$(E/N_0)_R = (E/N_0) + 10 \log(1/R) \tag{D5}$$

where R is the ratio of information bits to information plus parity bits transmitted. This is the E/N_0 which would be received by the identical communication system and information rate if no coding were used. For this increased signal-to-noise ratio the input bit error probability (4) is lower than the input bit error probability to the decoder (1). Only if the decoder reduces the output bit error probability (5) below this value can one speak of

[1]Numbers in parentheses refer to encircled numbers on the figure.

a coding gain. Above $(E/N_0)_R$ the output bit error probability is entered (5). And the horizontal distance between the point just drawn and the curve for no coding is the coding gain (3-6).

While the bit error probability criterion is sufficient for selecting a small number of interesting codes for a given application, detailed simulation is required for the final choice. As explained in the text, word error probability for six-bit words has been chosen as the measure of performance. Figure 35 shows the steps of the calculations for the specific example of the (7,6) parity check code. The word error probability curve for no coding is drawn by calculating the probability of errors occurring in a six-bit word (1-P(OE)) for given bit error probabilities (1). The calculated point (3) is then drawn vertically above the E/N_0 (2) for the selected bit error probability. The coding performance is calculated in the following steps. Select a bit error probability (1). Find the corresponding E/N_0 (2) and add the rate loss of 10 log(7/6) = 0.67 dB (4). The new increased bit error probability is found (5) at which the bits will be received from the same transmitter at the 7/6 higher bit rate. Calculating the probability of an even number of errors in a word received at that average bit error probability results in the word error probability (6). However, parity tagged words are discarded. Therefore, one must calculate the word error probability on the remaining words, which increases the word error probability by a factor 1/(1-P(odd number of errors)) (7). Also, the information rate has decreased by the additional rate loss of 10 log(1(1-P(oddE))) (8). However, this small additional loss is not shown on the above curve, since E/N_0 is the received signal-to-noise energy ratio per information bit before decoding; instead, the deletion rate is shown as a parameter in the text. From the two curves thus generated, one can determine the coding gain for any desired word error probability.

APPENDIX E

BIT ERROR CALCULATIONS FROM ERROR PACKET SIMULATIONS

For a low bit error probability, block codes and convolutional codes correct most of the errors. To obtain a sufficient statistical sample a very long bit stream would have to be examined. It would take hours of IBM 7094 computer time to evaluate even one code at one bit error probability. Therefore, a scheme was developed to calculate code performances by examining only potential error-causing situations.

For block codes the method is very simple. The bit stream is divided into blocks of a given constraint length (e.g., 73 for the (73,45) code). A given number of input errors are introduced randomly into blocks of data and the average number of output errors are calculated simply as the weighted average

$$P_{bit} = \sum_{i=i_{min}}^{i_c} \bar{e}_i [P(i)/N_{bl}] \qquad \text{(E1)}$$

where

P_{bit}	bit error probability
i_c	constraint length
i_{min}	minimum number of errors which can give output errors
$P(i)$	probability of i errors in a block
N_{bl}	block length
$\bar{e}_i$	average number of output errors in i_c given i input errors

In practice only a few terms are needed since $P(i)$ decreases rapidly with increasing i for the error probabilities under consideration.

For convolutional codes, the calculations are not so direct. Intuitively, it is clear that a convolutional code compared to a block code with identical constraint length and identical error-correcting capability will encounter error-causing situations more frequently. A model will be developed which is sufficiently accurate to predict the average number of error-causing situations which are encountered when N information plus parity bits are received.

Consider a sequence of N zeros and ones where the zeros represent correctly received and the ones represent error bits. If one counts the number of errors contained for each constraint length, a second sequence results.

This is shown in an example for a constraint length of 24, where it is assumed that all zeros border the sequence which is shown.

. . . 000000001010000010000001 . . .

0000000011222222333333$\underline{4}$4444444433222222211111111000000 . . .

Each number on the last line represents the number of errors within the sliding constraint length. Neglecting end effects for large N, the second sequence has the same number of members as the first, namely, N. The average number of times each member of the sequence occurs is

$$N_i = P(i)N \tag{E2}$$

One is interested only in how many times a maximum occurs, representing an error-causing situation. One must therefore calculate the average number of shifts that one stays inside an error cluster of a given number of errors. (In the above example, the 4 appears nine times in sequence.)

$$\bar{d}_i = \frac{\sum\limits_{j=1}^{j=i_c-i+1} j\binom{i_c-1-j}{i-2}}{\binom{i_c-1}{i-1}} \tag{E3}$$

where d_i is the average number of shifts within a cluster of i errors. Equation (E3) can best be understood when it is derived for an example, $(i = 4,\ i_c = 24)$. It is assumed that when the error condition is first reached the other $i-1$ error bits are randomly distributed over the constraint length. When the first and last bit are in error $(d = 1)$ the remaining $i-2$ errors may be located in any order in the i_c-2 bits. There are $\binom{i_c-2}{2}$ ways to distribute the errors. When the last bit in error is in the second position $(d = 2)$, there are $\binom{i_c-3}{2}$ ways to distribute the remaining two error bits, etc. Hence, the average distance is

$$\bar{d}_4 = \frac{1\binom{22}{2} + 2\binom{21}{2} + 3\binom{20}{2} + \ldots + 20\binom{3}{2} + 21\binom{2}{2}}{\binom{23}{3}} = 6.0$$

where $\binom{23}{3}$ is the total number of ways the $i-1 = 3$ errors can be distributed over the 23 positions. It can be shown by mathematical induction that equation (E3) can be reduced to

$$\bar{d}_i = i_c/i \tag{E4}$$

which is the average distance between errors, provided there are i errors in i_c. When clustering of more than i errors in i_c has negligible probability of occurrence compared to i errors in i_c, the average total number of error-causing situations is from equations (E2) and (E3):

$$N_i/\bar{d}_i = P(i)N/\bar{d}_i \tag{E5}$$

When $\bar{e}_i$ is the average number of output errors in $N \cdot R$ information bits, and R is the signaling rate (equal to 1/2 for the codes considered in this report), then

$$\text{Average total number of errors in } N \text{ information bits caused by error bursts of length } i = NP(i)\bar{e}_i/R\bar{d}_i \tag{E6}$$

As the example shows, a peak is reached in steps of one. Hence, when the peak is of magnitude i on the average, it is flanked by $2d_i$ $(i-1)$. These must be subtracted to count peaks of magnitude $i-1$.

$$N_{i-1} = [P(i-1)N - 2NP(i)]/\bar{d}_{i-1} \tag{E7}$$

and

$$\text{Average number of errors in } N \text{ information bits caused by error bursts of length } i-1 = \{N[P(i-1) - 2P(i)]/R\bar{d}_{i-1}\}\bar{e}_i \tag{E8}$$

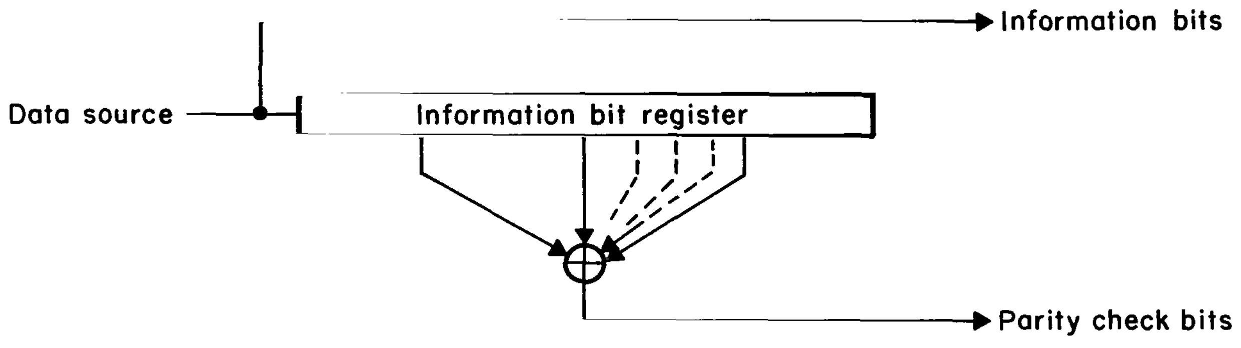
For $i-2$ error bursts one would have to subtract terms containing $P(i-1)$ and $P(i)$. However, equation (E8) is sufficiently accurate, since $P(i)$ decreases rapidly as i increases for the error probabilities considered. It is also valid for the highest clustering that may be considered, since $P(i+1) = 0$ for $i + 1 > i_c$; thus, the equation for convolutional codes for bit error probability is:

$$P_{bit} = \frac{1}{R} \sum_{i=i_{min}}^{i_c} \bar{e}_i \left[\frac{P(i) - 2P(i+1)}{\bar{d}_i}\right] \tag{E9}$$

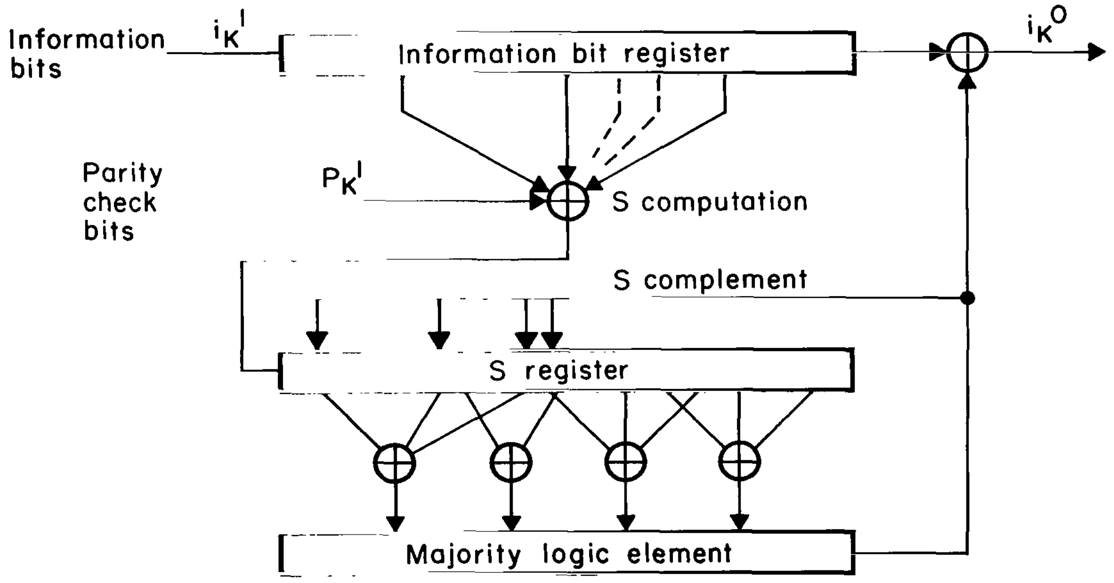
As in equation (E1) only the first few terms need to be considered.

REFERENCES

1. Massey, James L.: Threshold Decoding. M.I.T. Press, Cambridge, Mass., 1963.
2. Lumb, Dale R.; and Neuman, Frank: Error Rate Reduction of Parity Checked Telemetry Data by a Likelihood Deletion Strategy. NASA TN D-3576, 1966.
3. Forney, G. David, Jr.: Concatenated Codes. Tech. Rep. 440, Res. Lab. of Electronics, M.I.T., Cambridge, Mass., Dec. 1, 1965.
4. Peterson, William Wesley: Error-Correcting Codes. M.I.T. Press and John Wiley and Sons, Inc., 1961.
5. Baghdady, Elie J., ed.: Lectures on Communication System Theory. McGraw-Hill Book Co., Inc., 1961.
6. Lumb, Dale R.; and Hofman, Larry B.: An Efficient Coding System for Deep Space Probes With Specific Application to Pioneer Missions. NASA TN D-4105, 1967.

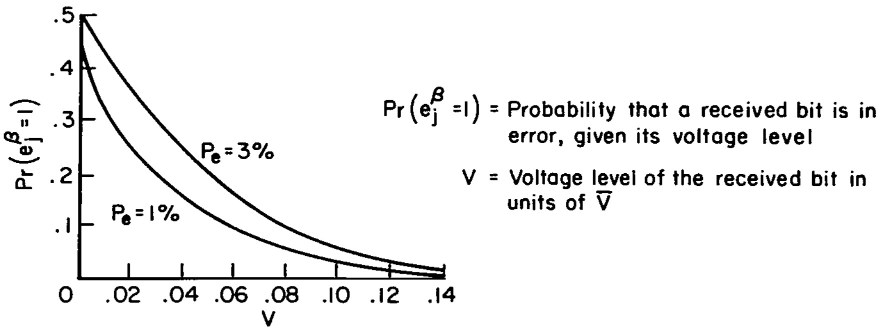


(a) Encoder.

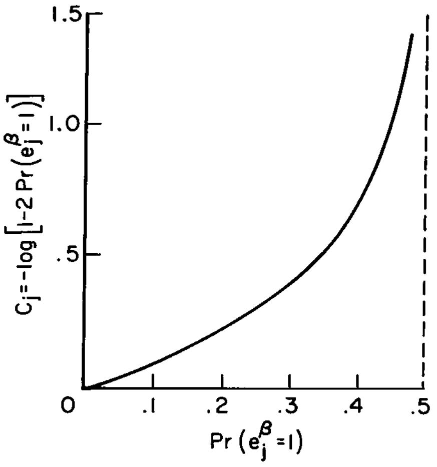


(b) Decoder.

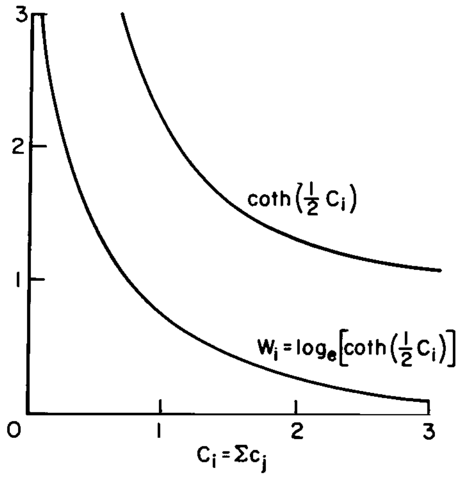
Figure 1.- Block diagrams of rate one-half convolutional encoder and threshold decoder.



(a) Error probability of a given bit versus voltage received.



(b) Calculated c_j from the error probability of the bits.



(c) Calculation of the weights.

Figure 2.- Illustration of the calculation of weights.

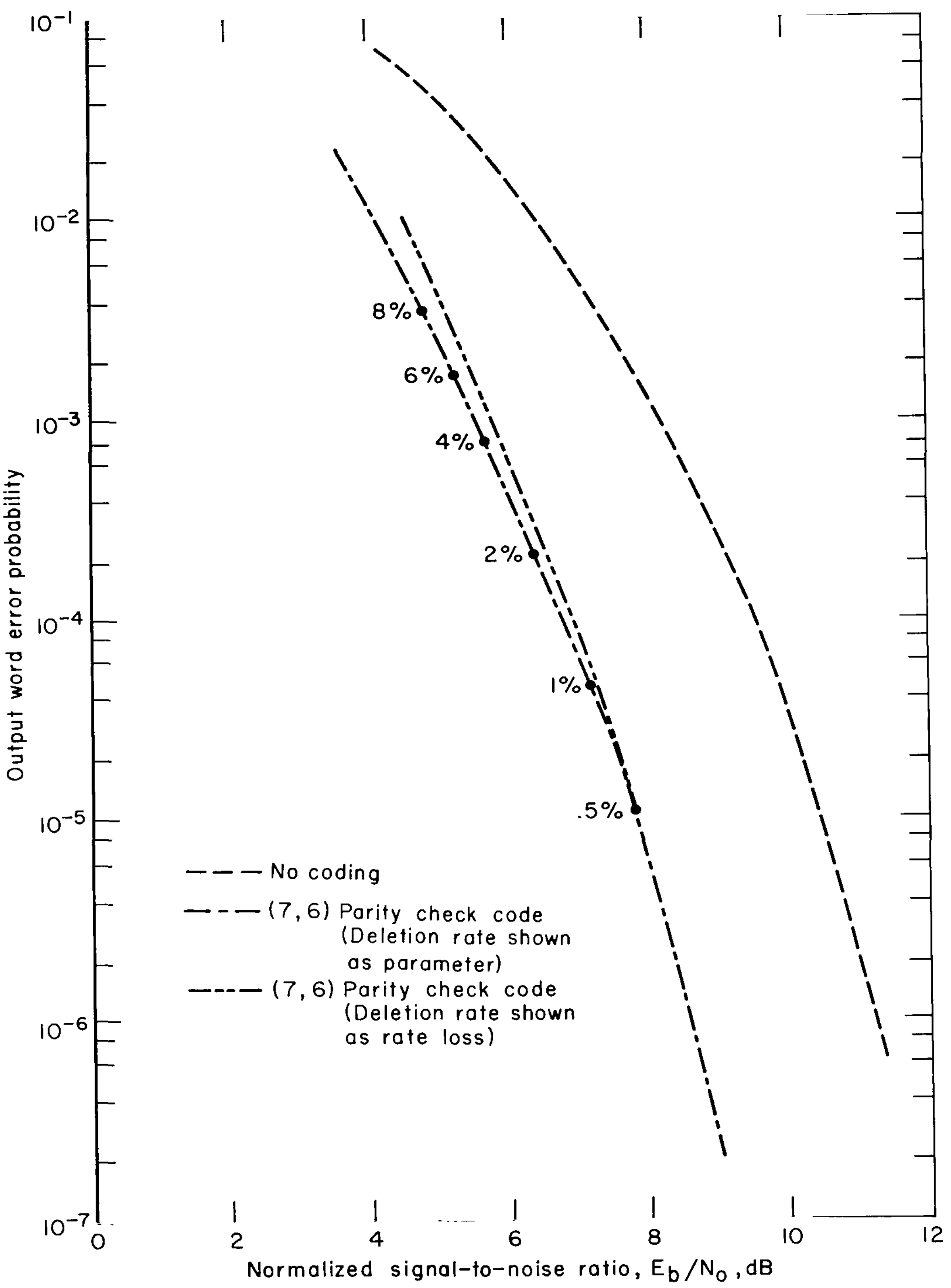


Figure 3.- Performance of parity error detection compared to no coding.

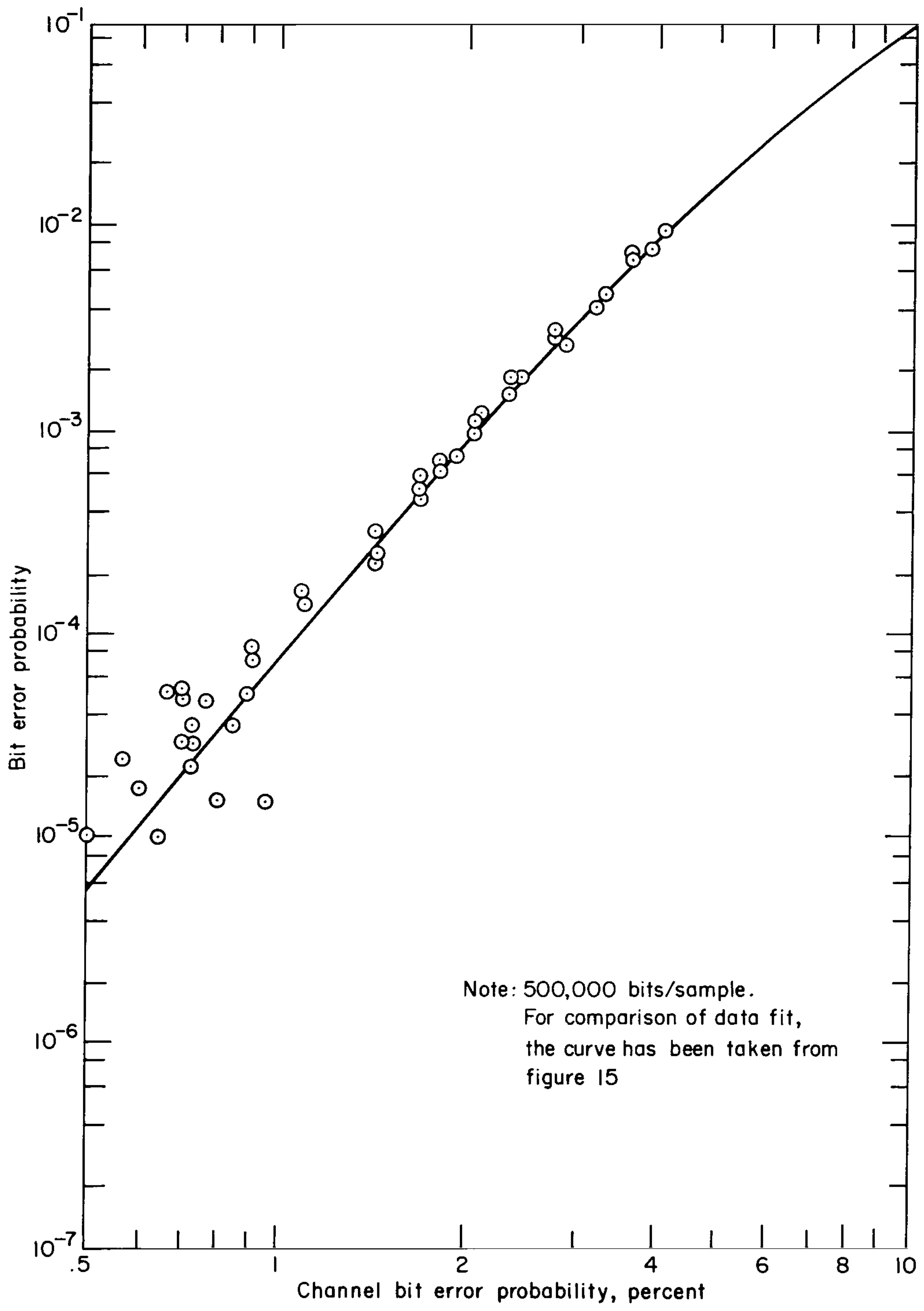


Figure 4.- Performance of majority decision (24,12) hardware decoder.

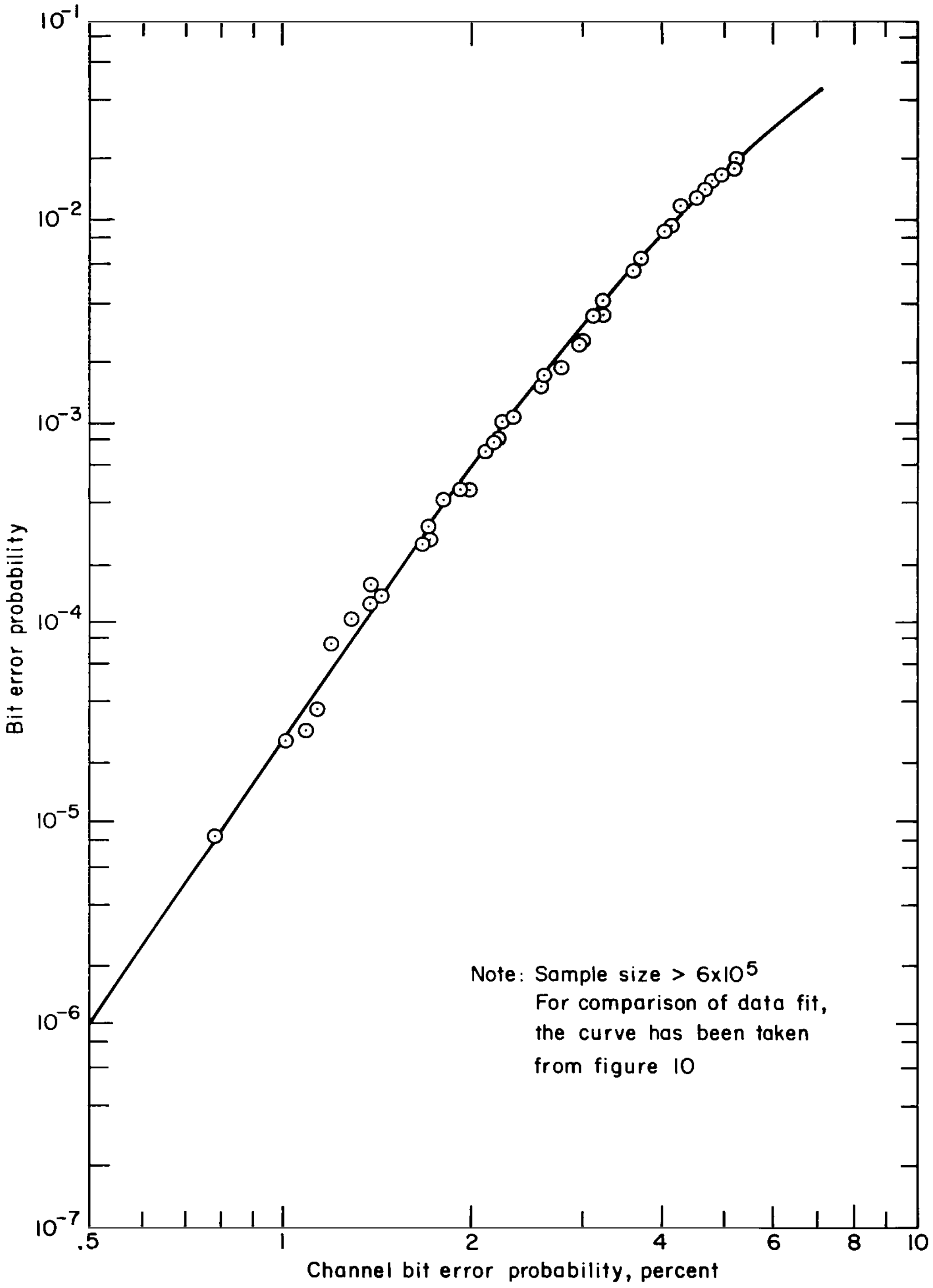


Figure 5.- Performance of the majority decision (73,45) hardware decoder.

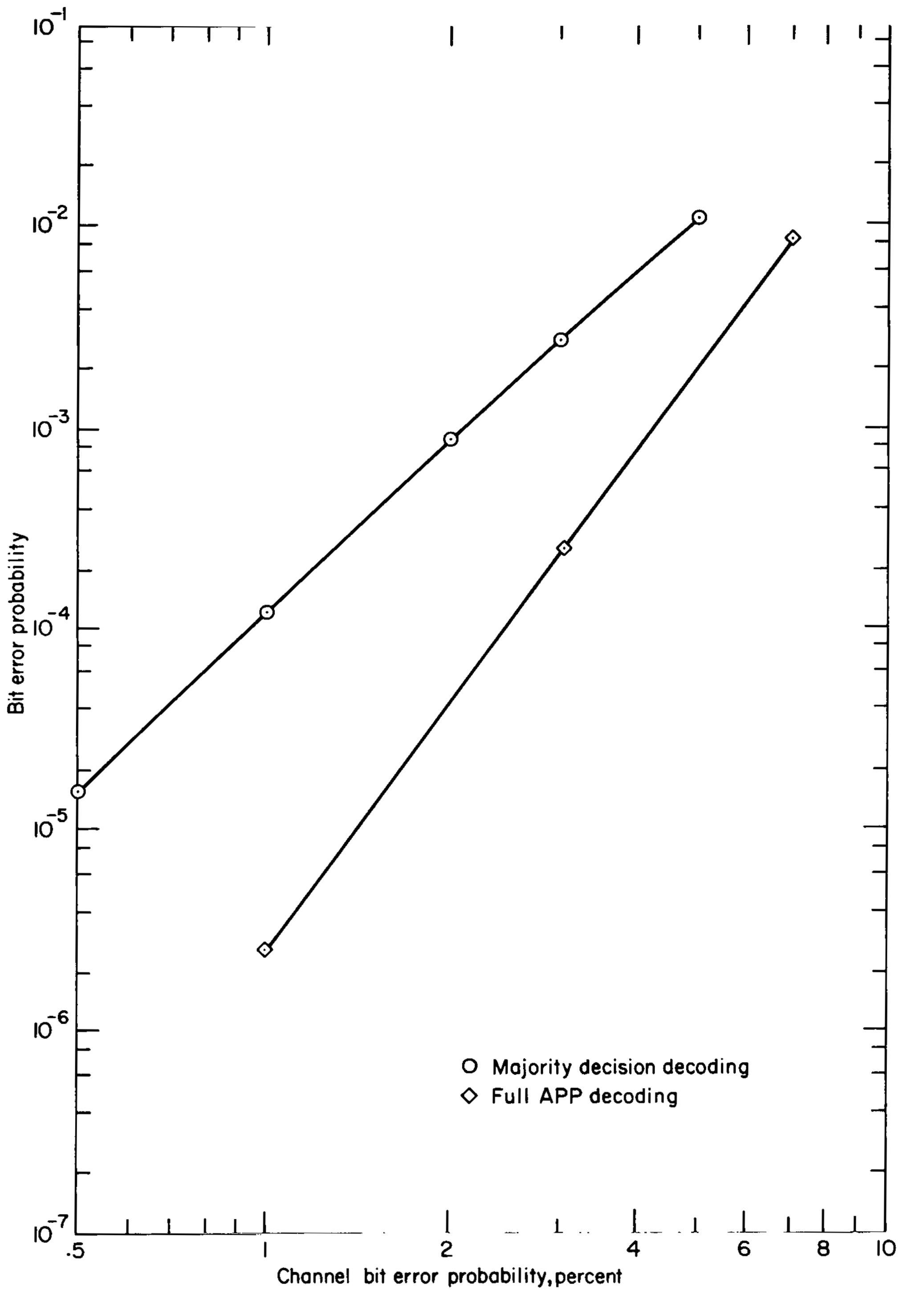


Figure 6.- (15,7) block code - bit error rates.

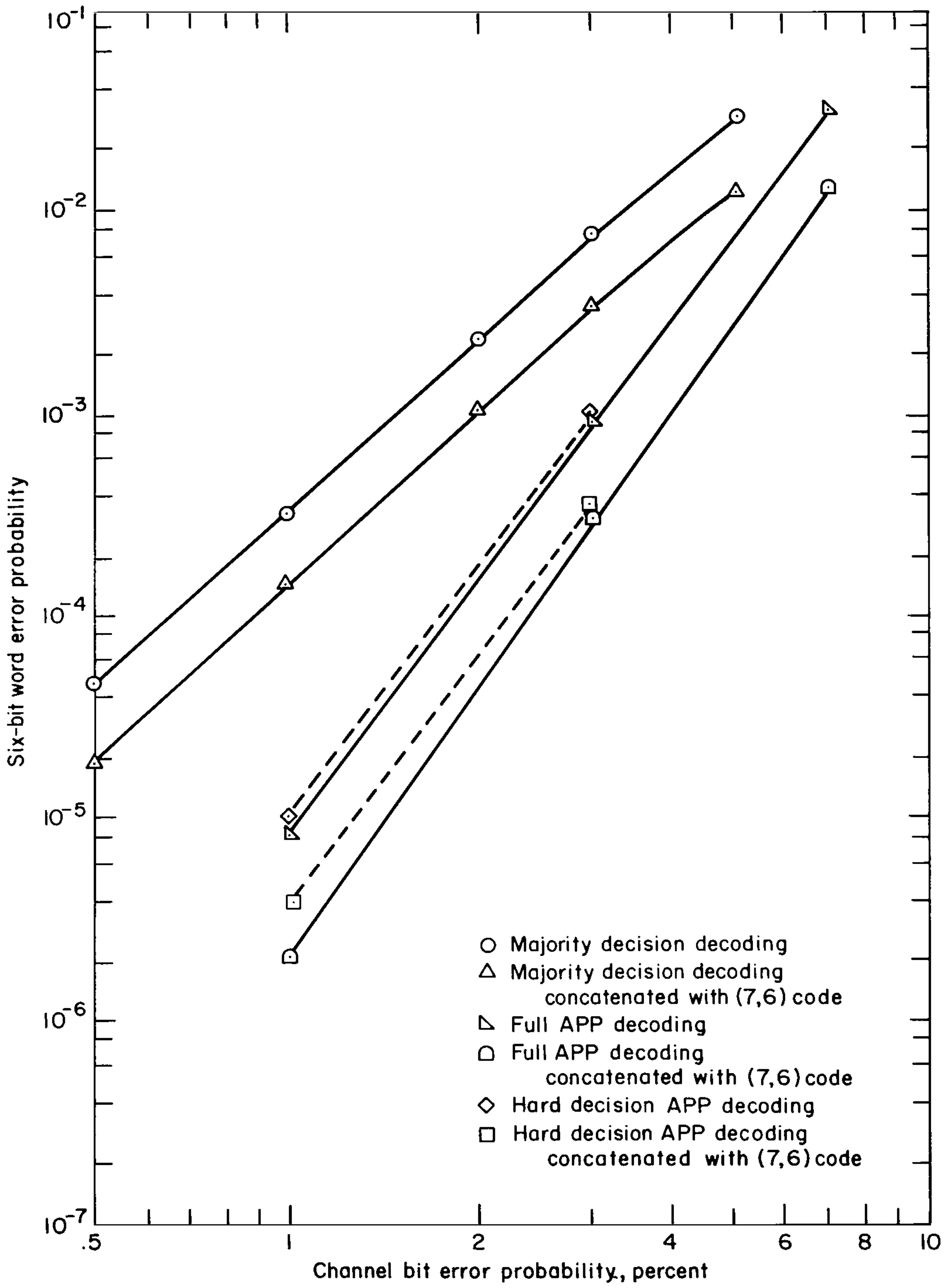


Figure 7.- (15,7) block code - six-bit word error rates.

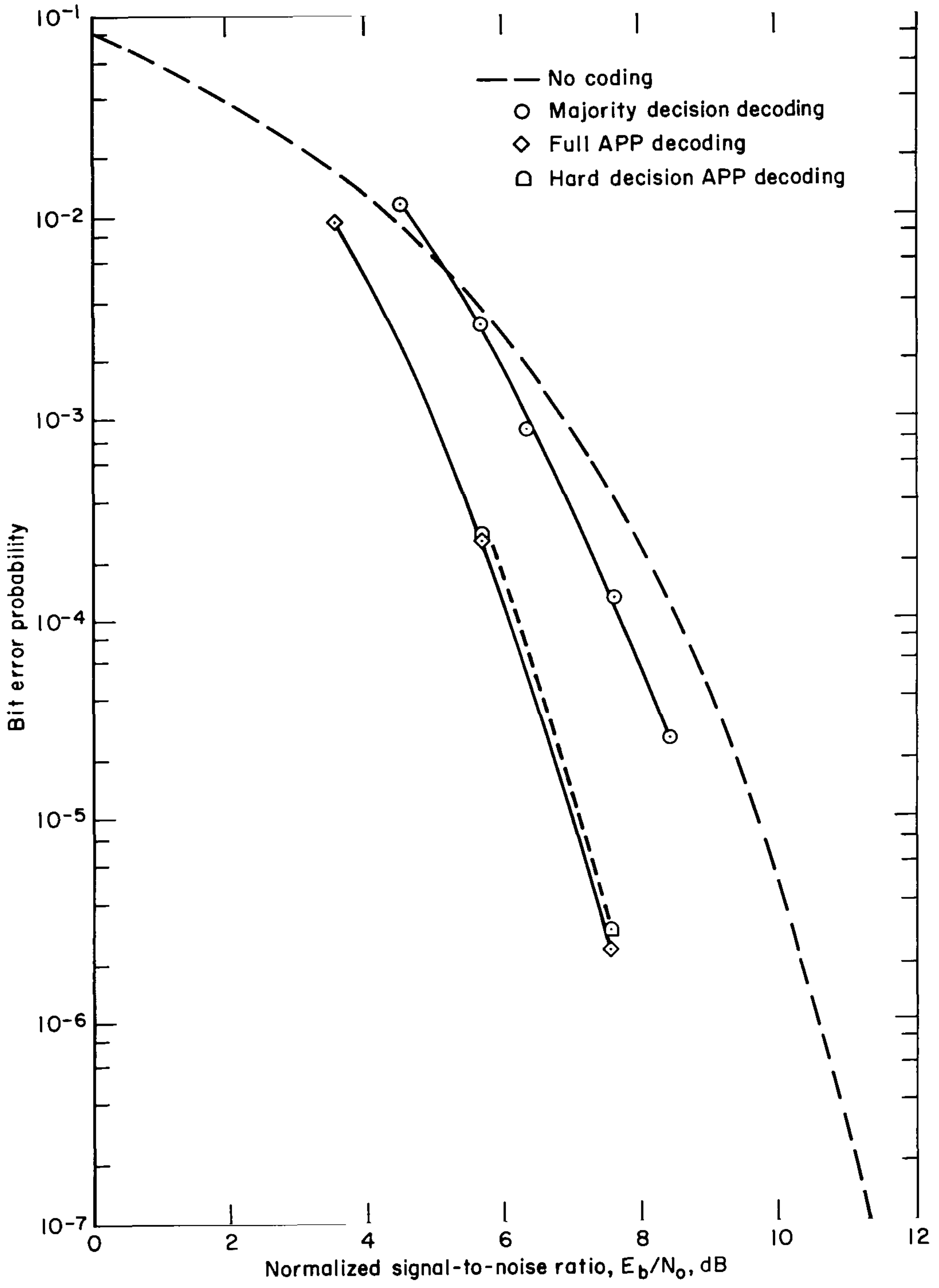


Figure 8.- (15,7) block code - bit error rate performance.

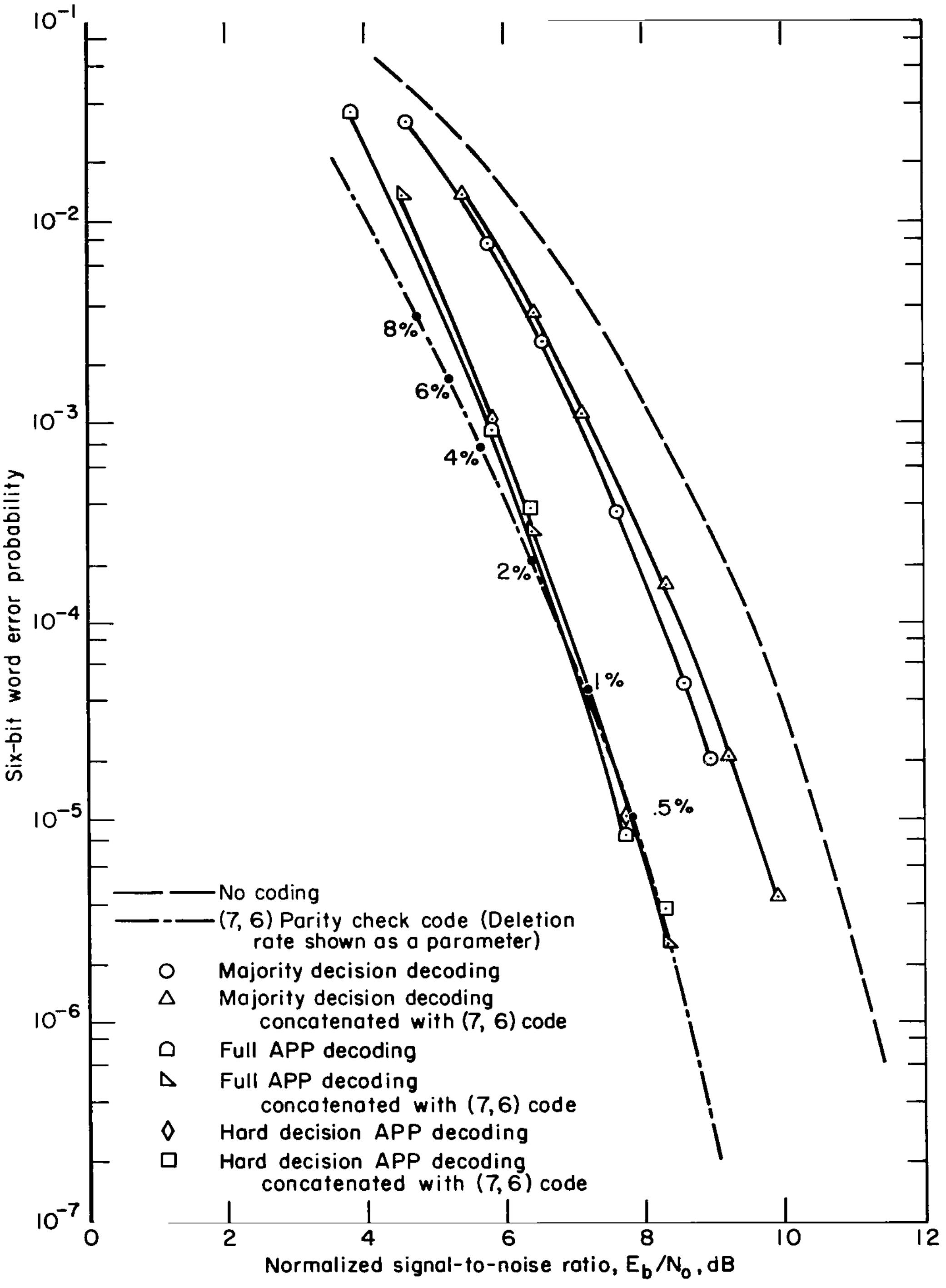


Figure 9.- (15,7) block code - six-bit word error performance.

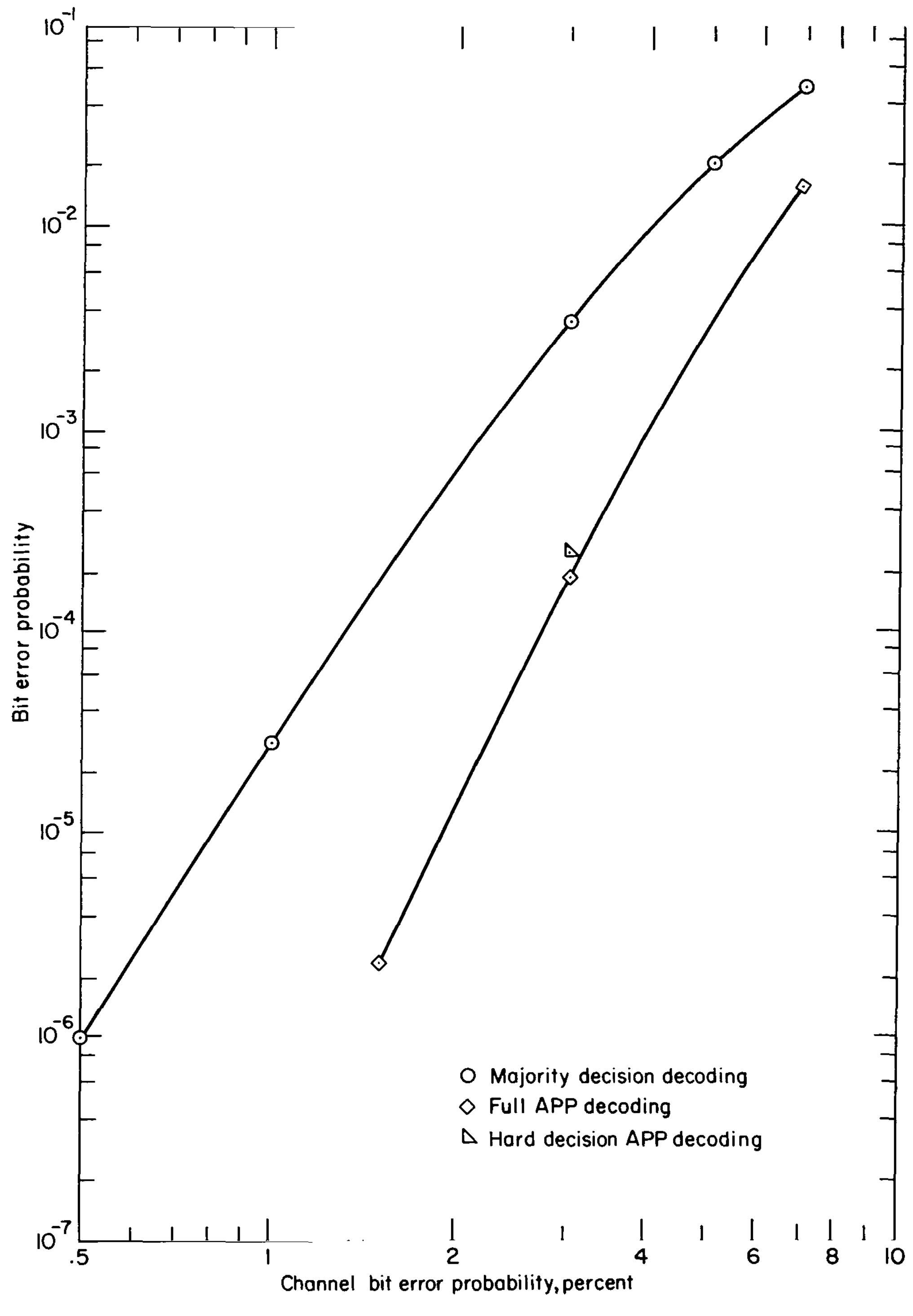


Figure 10.- (73,45) block code - bit error rates.

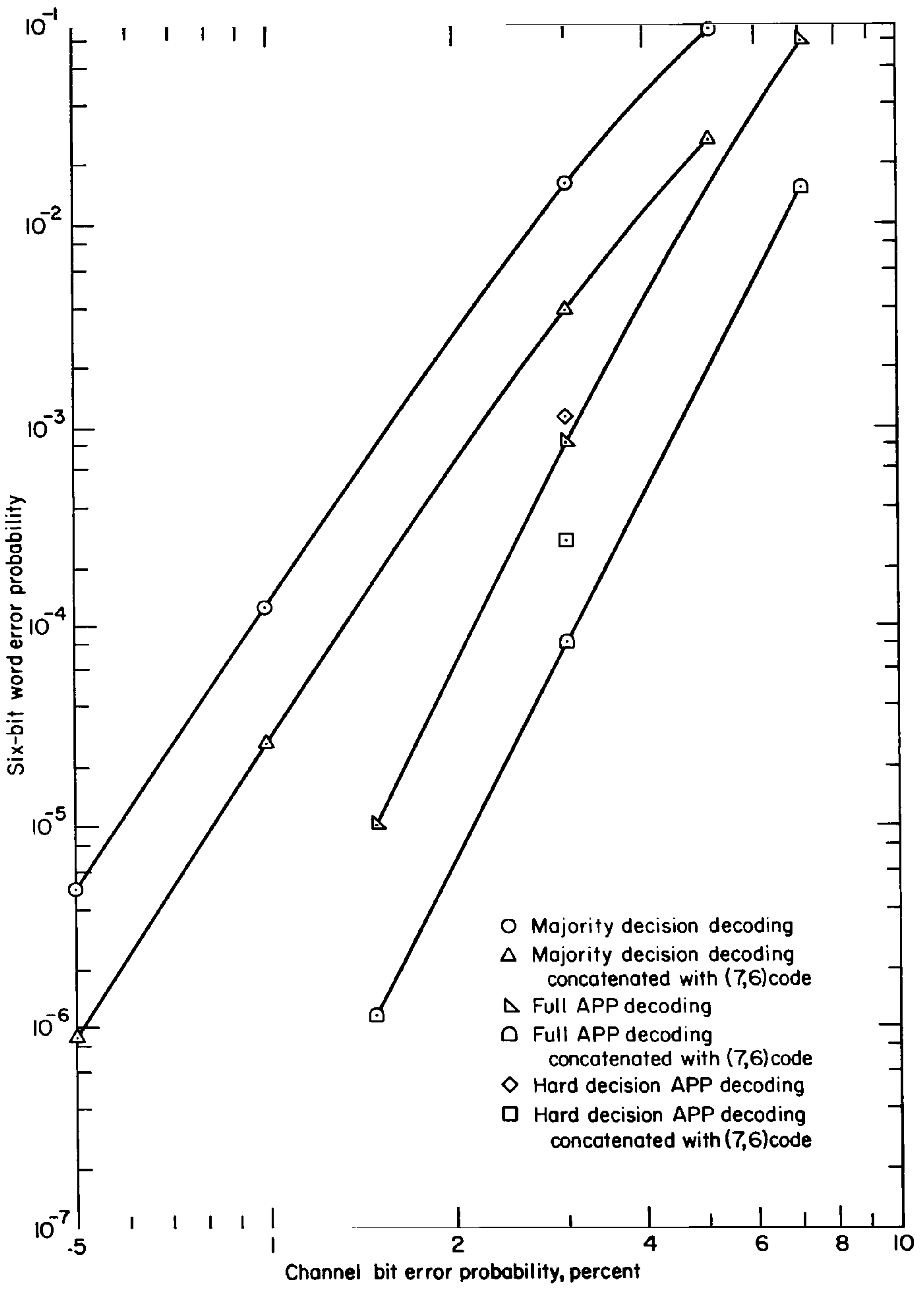


Figure 11.- (73,45) block code - six-bit word error rates.

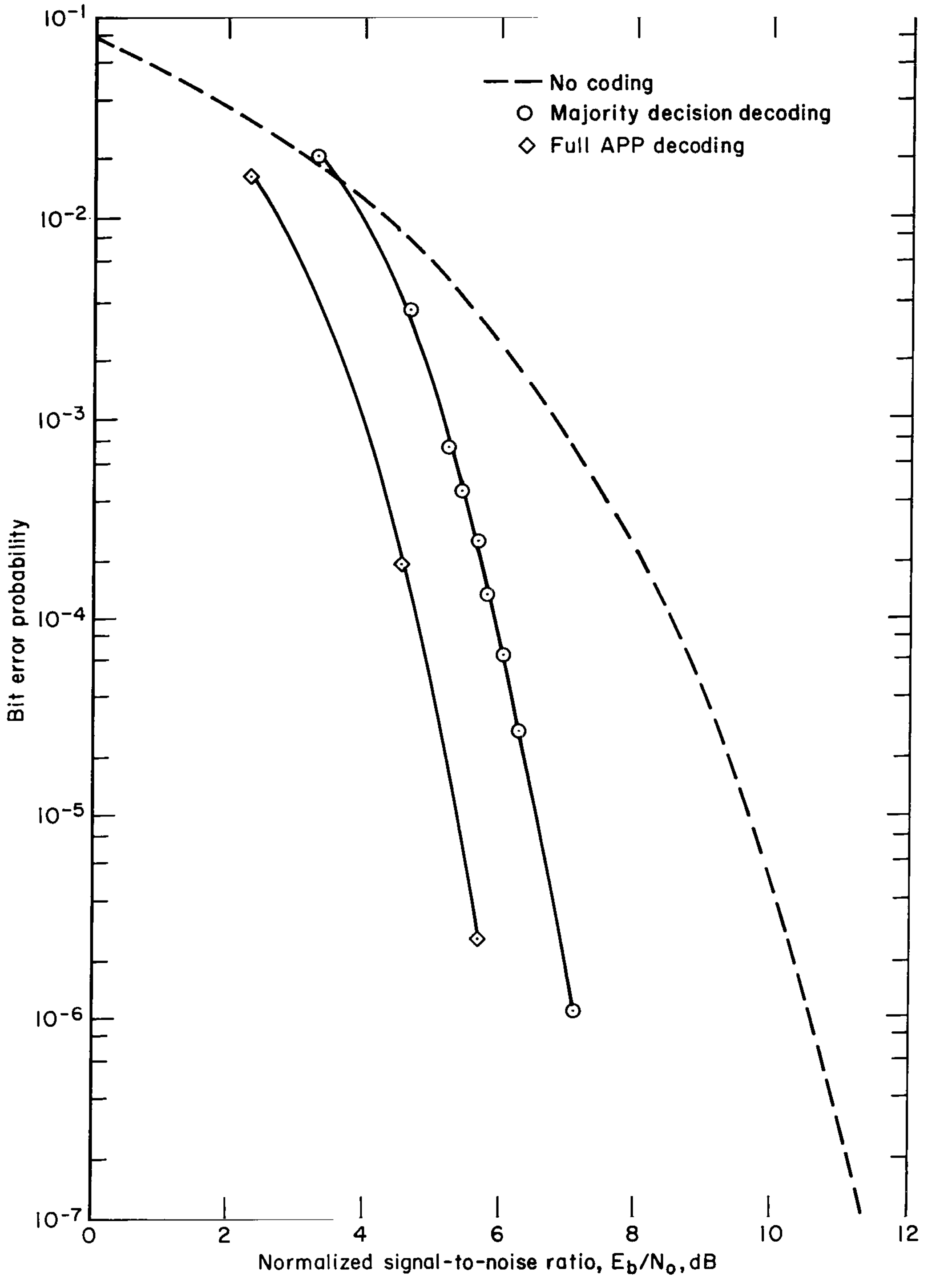


Figure 12.- (73,45) block code - bit error rate performance.

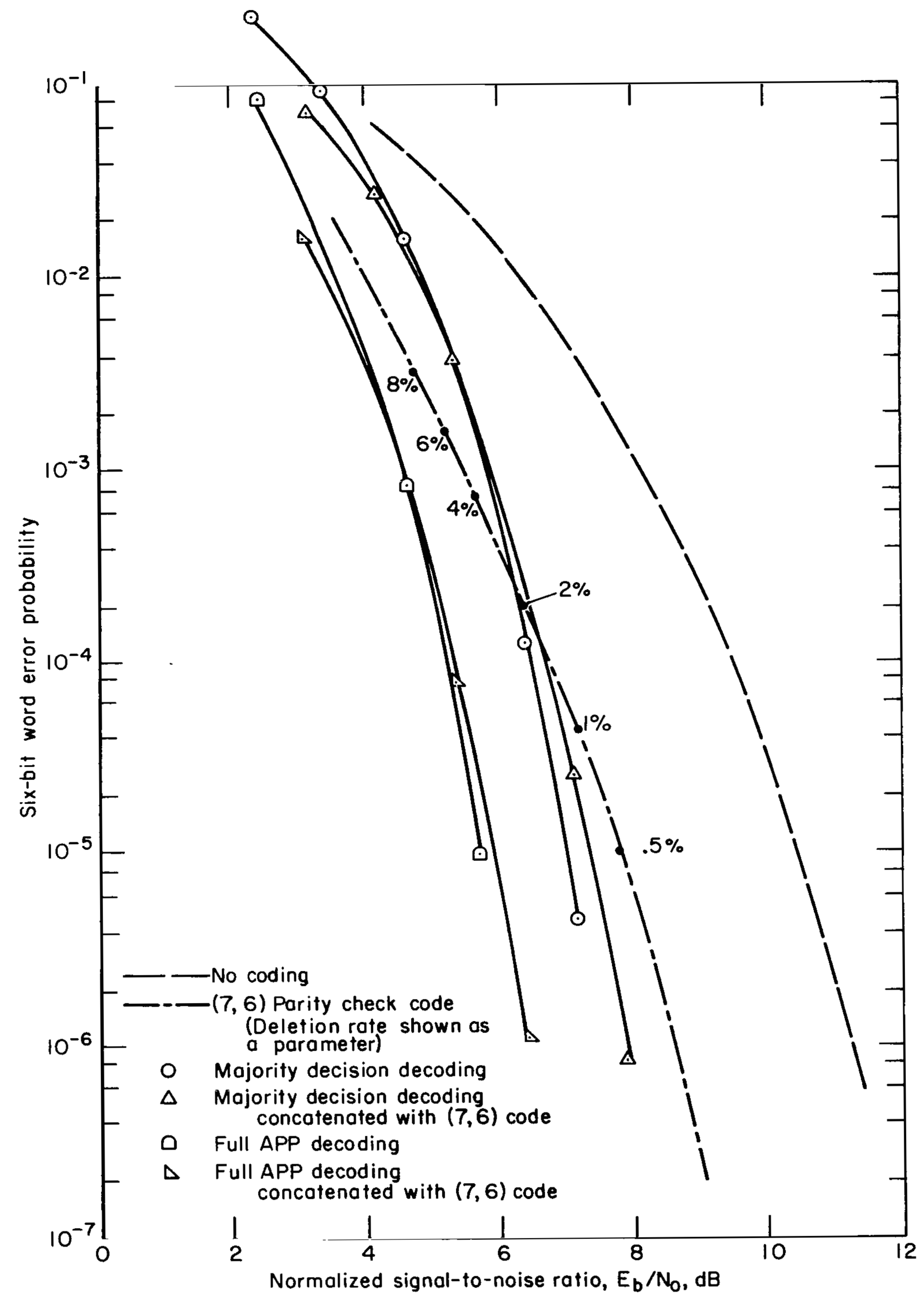


Figure 13.- (73,45) block code - six-bit word error rate performance.

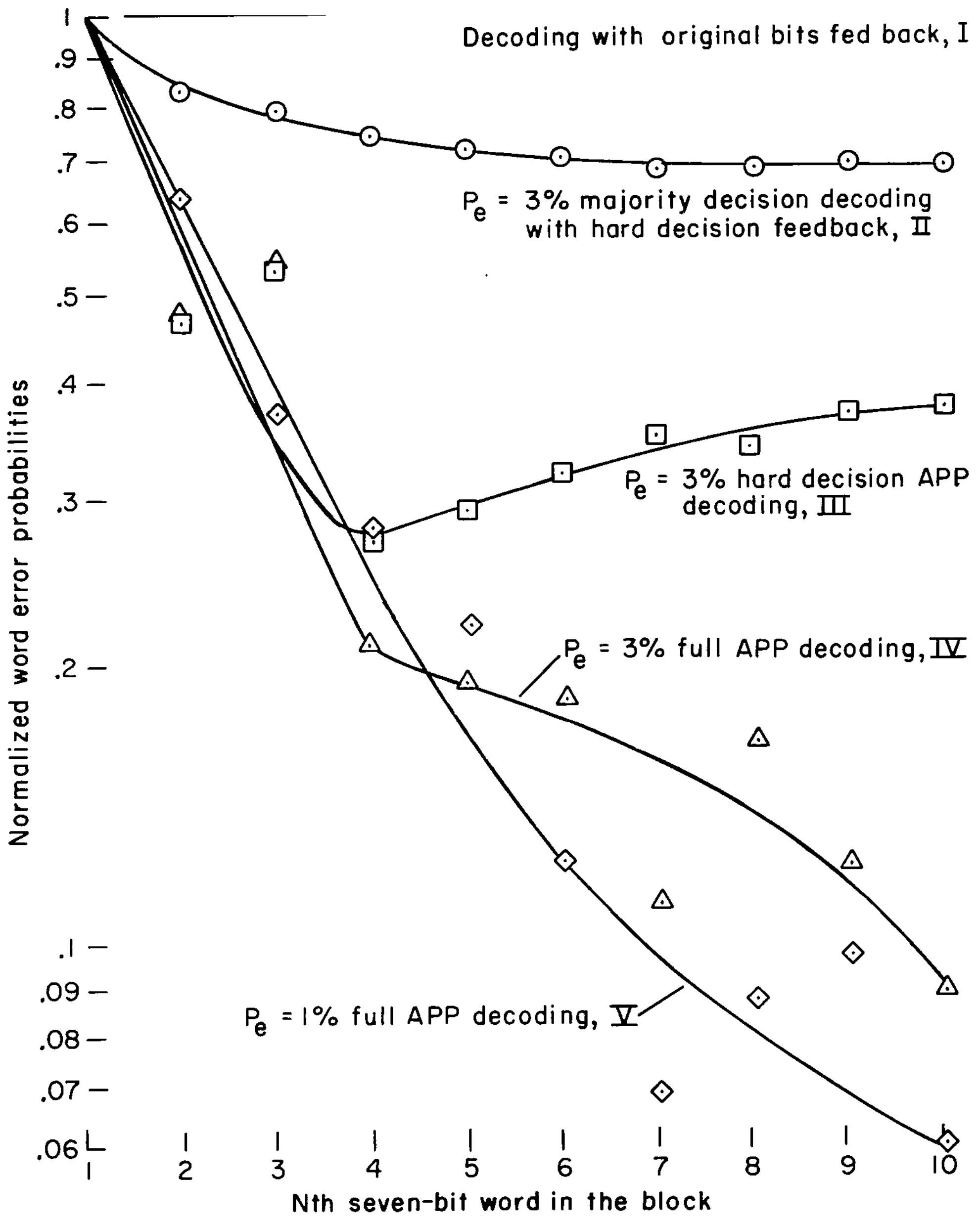


Figure 14.- Normalized seven-bit word error probability as a function of the decoding sequence.

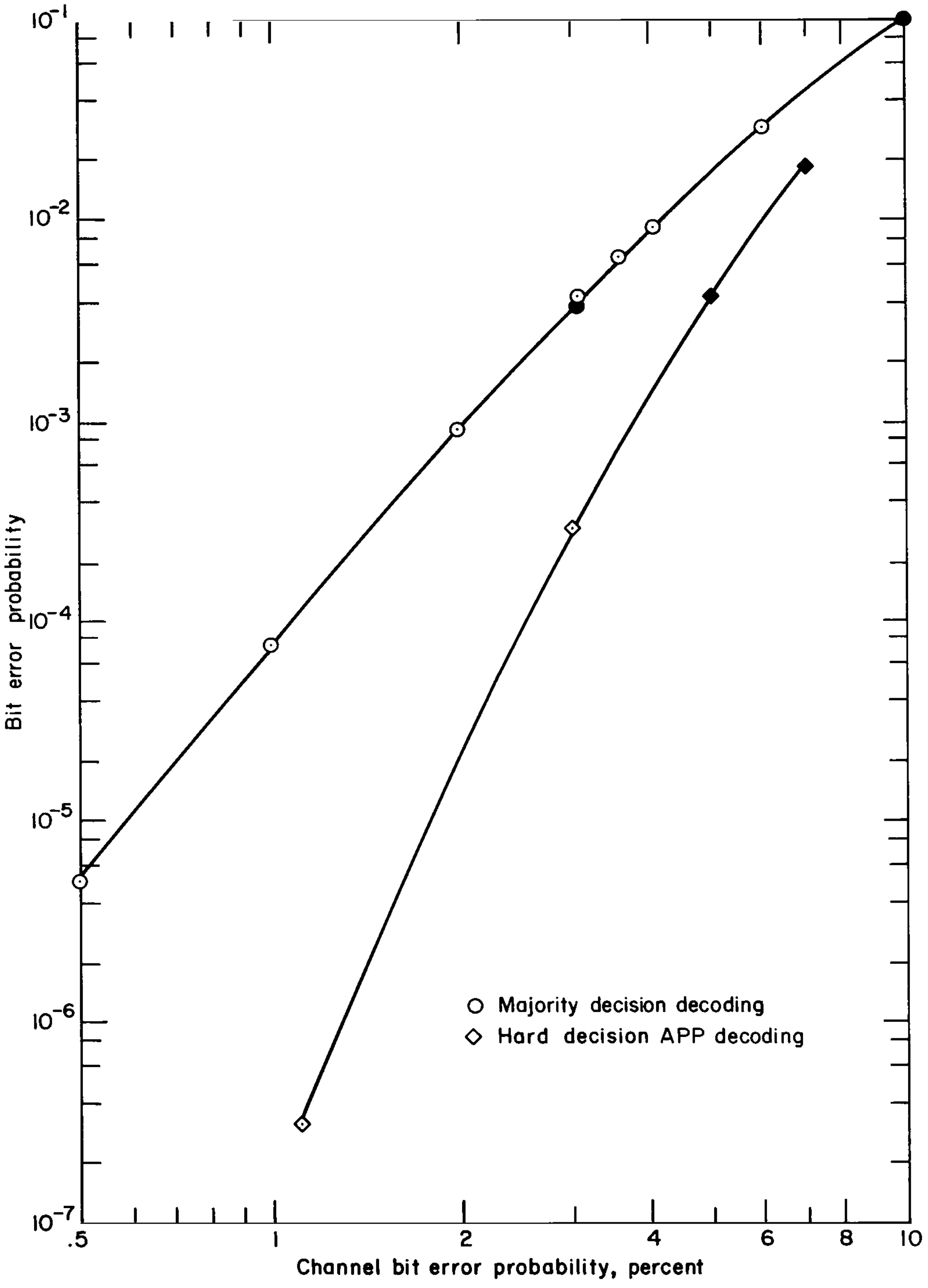


Figure 15.- (24,12) convolutional code - bit error rates.

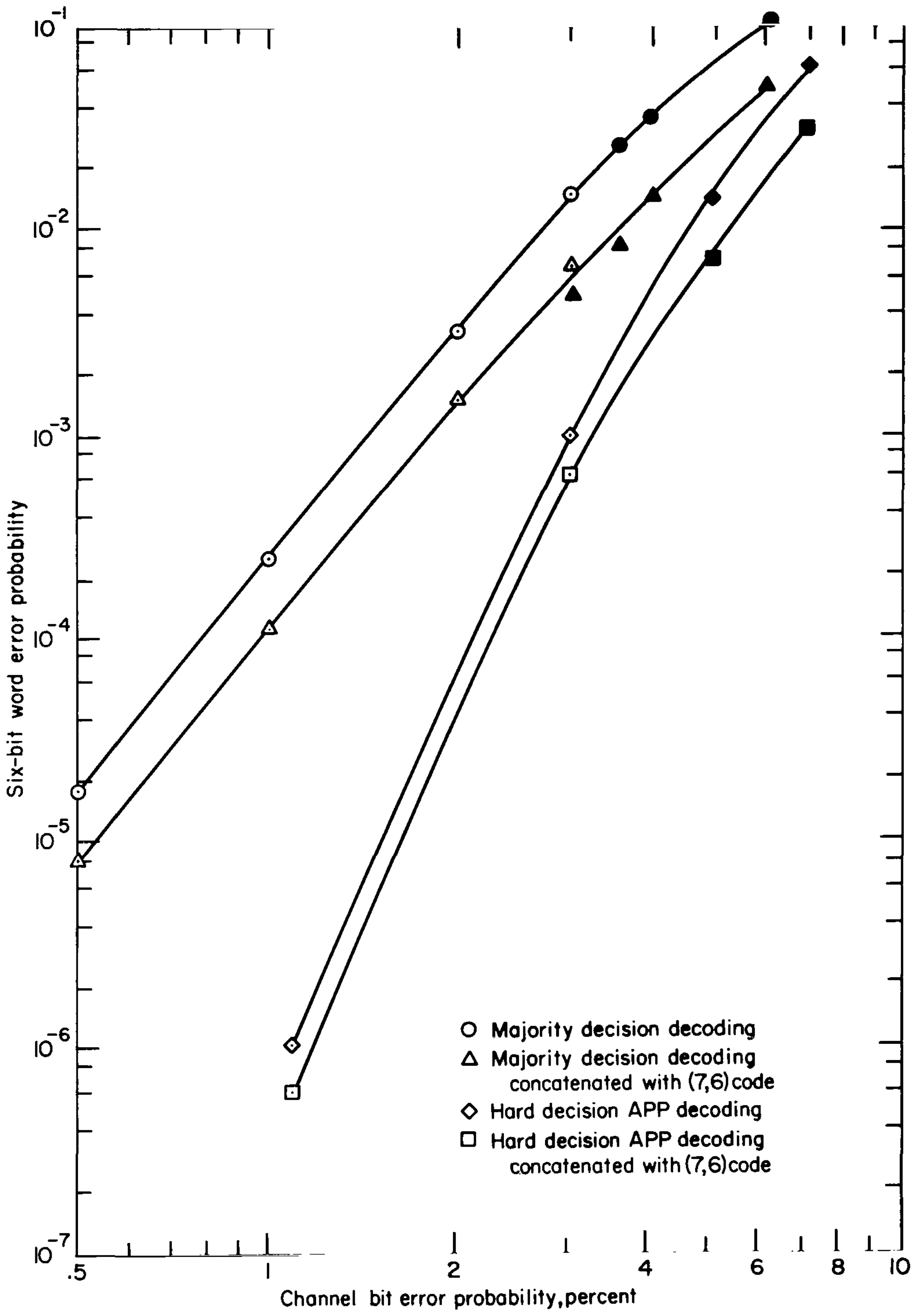


Figure 16.- (24,12) convolutional code - six-bit word error rates.

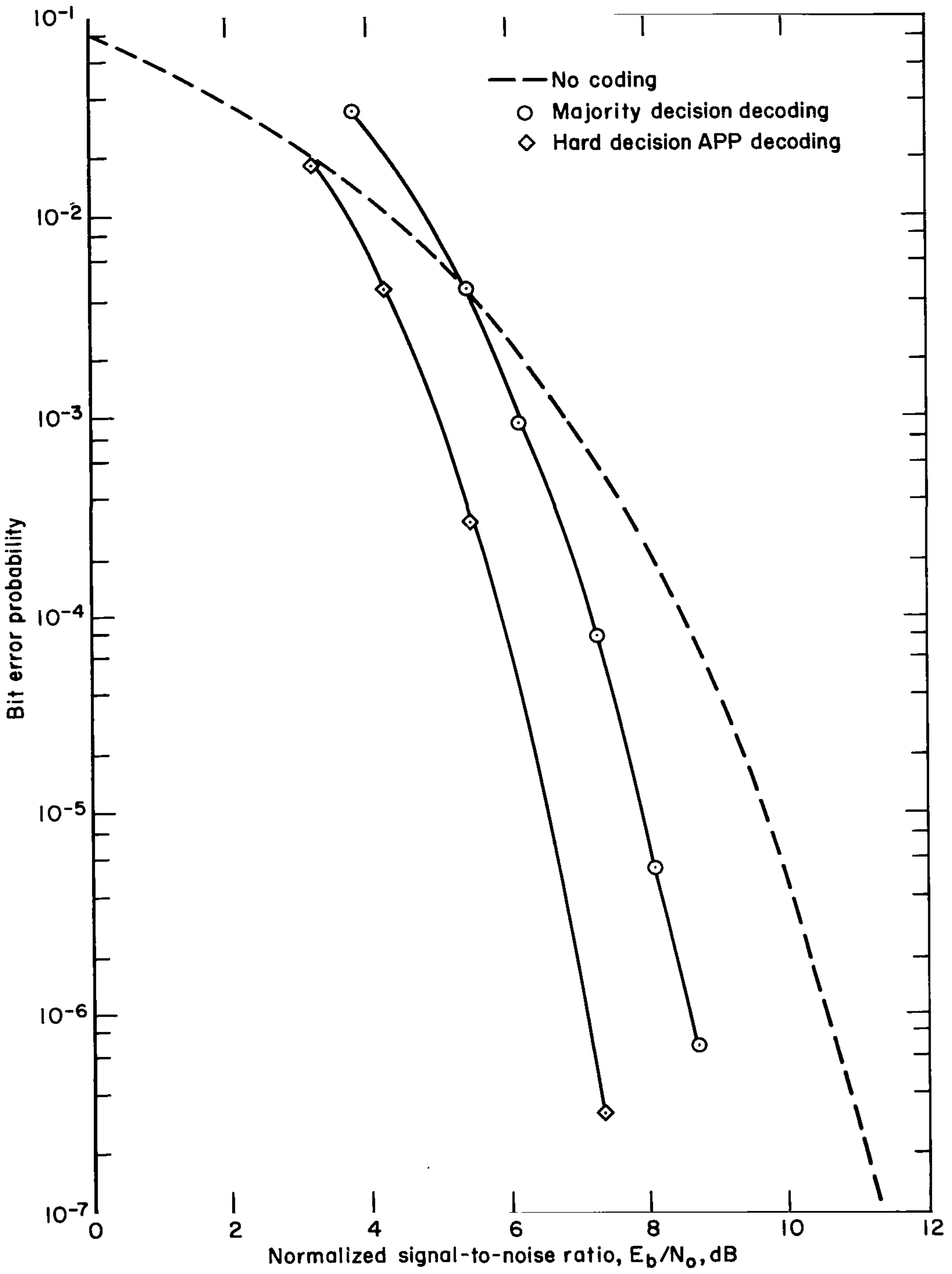


Figure 17.- (24,12) convolutional code - bit error rate performance.

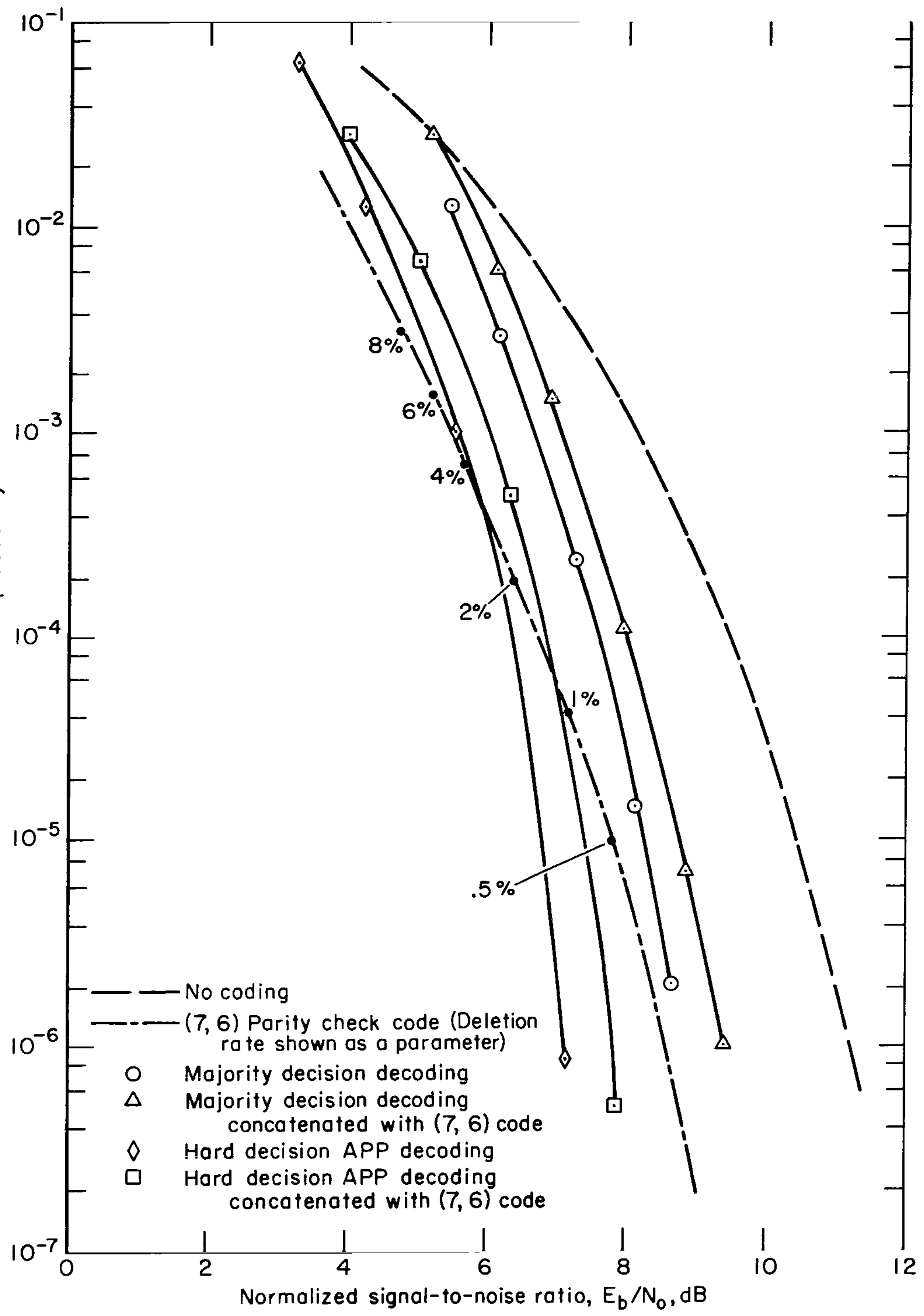
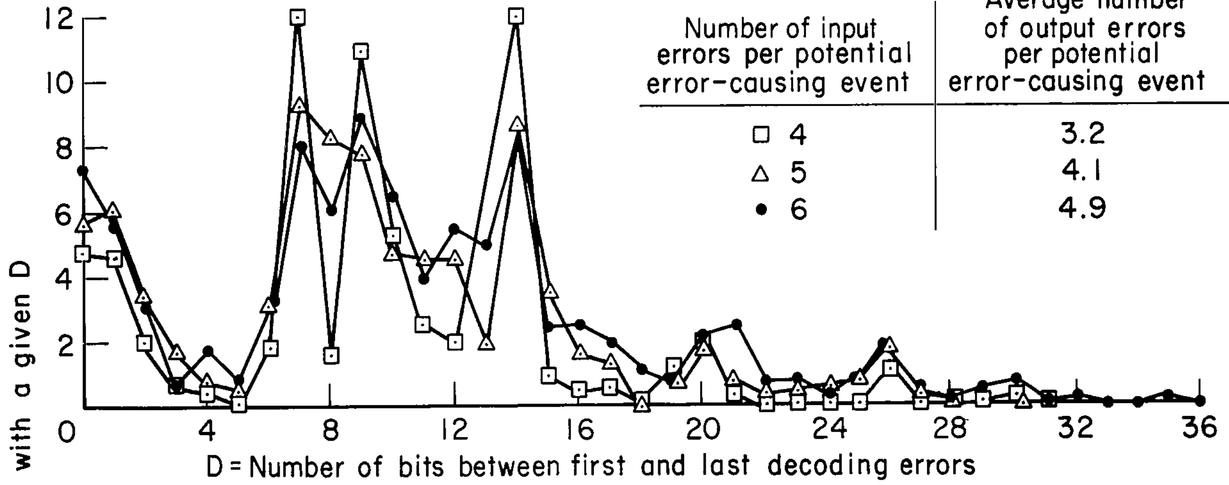
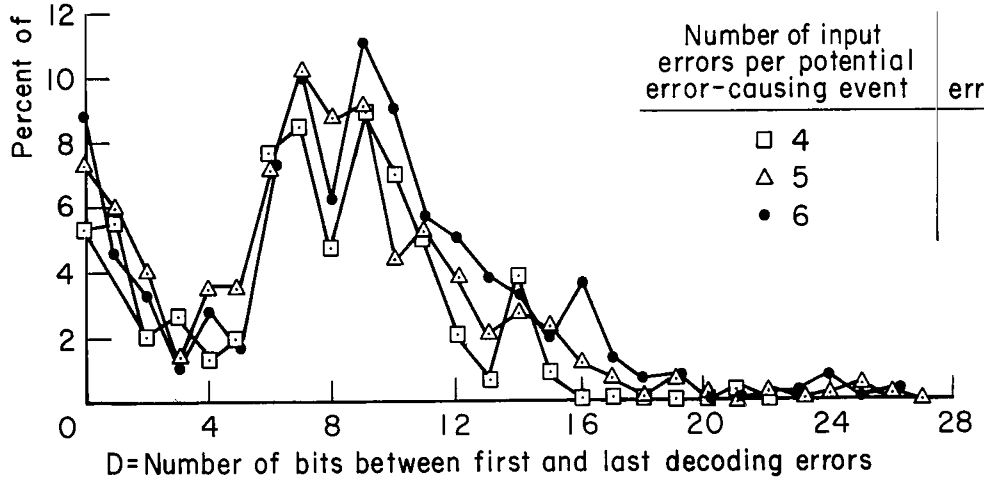


Figure 18.- (24,12) convolutional code - six-bit word error rate performance.



Number of input errors per potential error-causing event	Average number of output errors per potential error-causing event	Percent of decoded potential error-causing events with no errors out
□ 4	3.2	33.3
△ 5	4.1	16.7
● 6	4.9	5.5

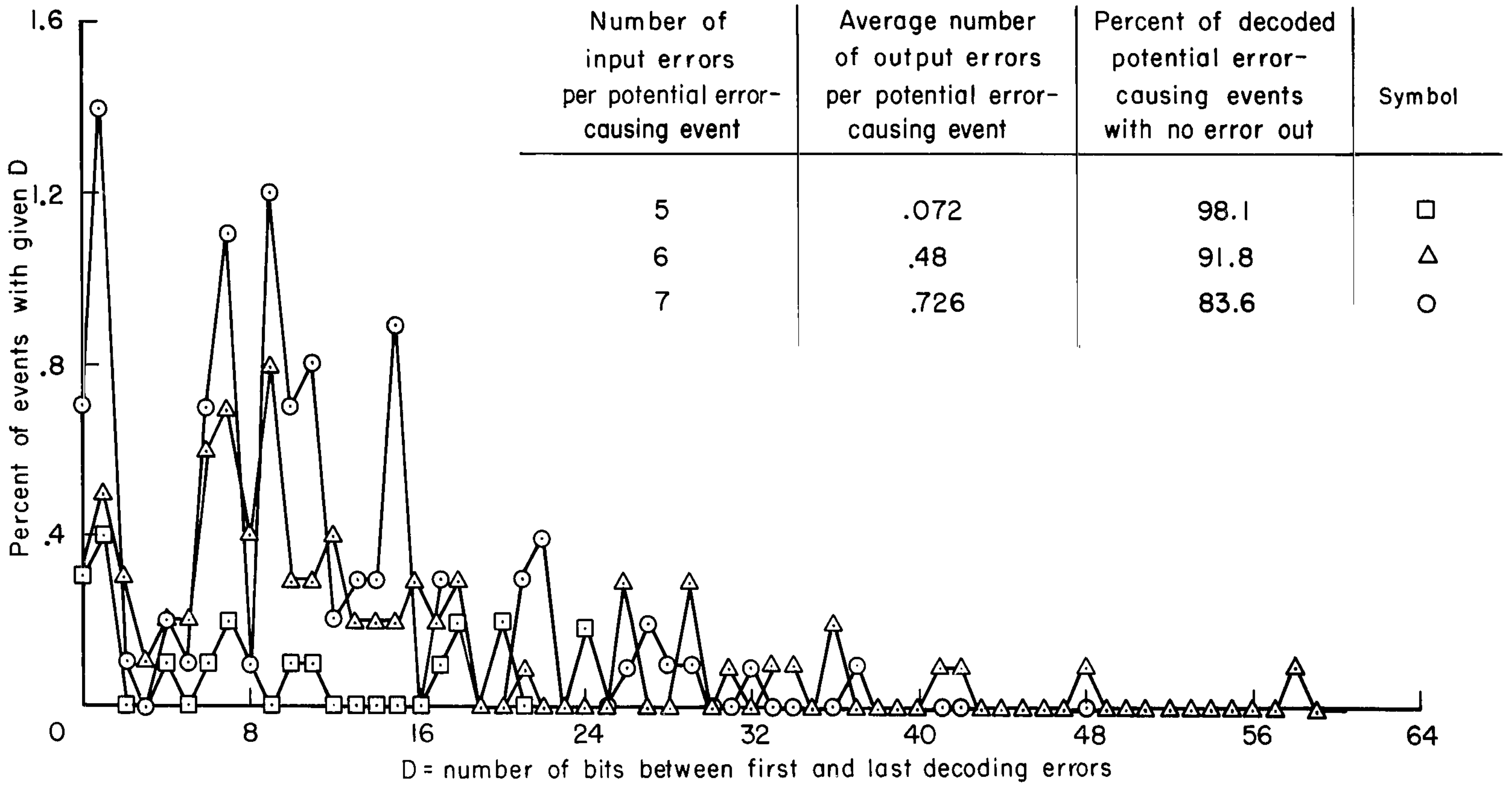
(a) (24,12) code; majority decision decoded without alarm.



Number of input errors per potential error-causing event	Average number of output errors per potential error-causing event	Percent of decoded potential error-causing events with no errors out
□ 4	2.5	33.3
△ 5	3.3	14.7
● 6	4.2	3.1

(b) (24,12) code; majority decision decoded with alarm.

Figure 19.- Error dispersion for the (24,12) convolutional code.



Number of input errors per potential error-causing event	Average number of output errors per potential error-causing event	Percent of decoded potential error-causing events with no error out	Symbol
5	.072	98.1	□
6	.48	91.8	△
7	.726	83.6	○

(c) (24,12) code; APP decoded without alarm.

Figure 19.- Concluded.

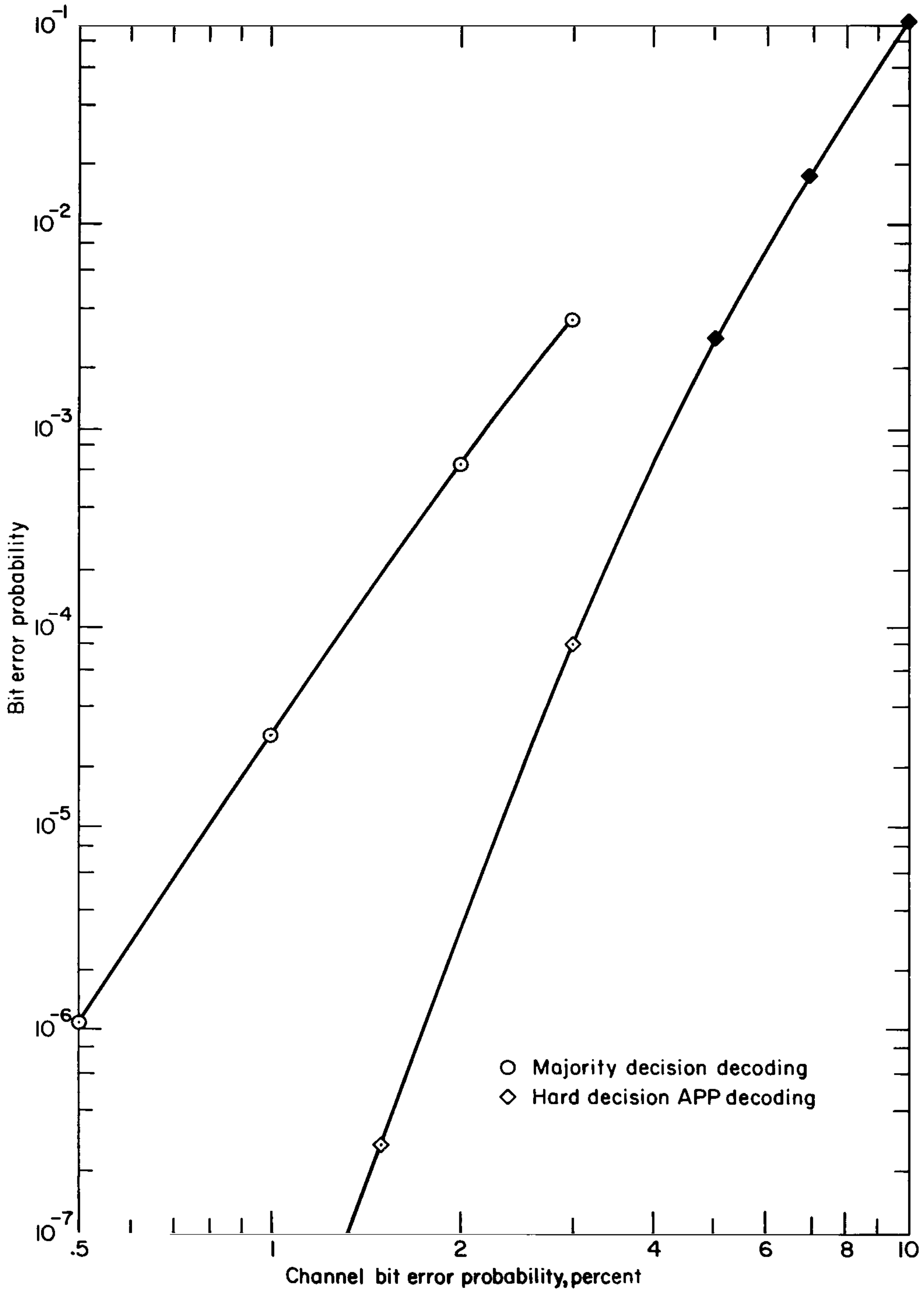


Figure 20.- (44,22) convolutional code - bit error rates.

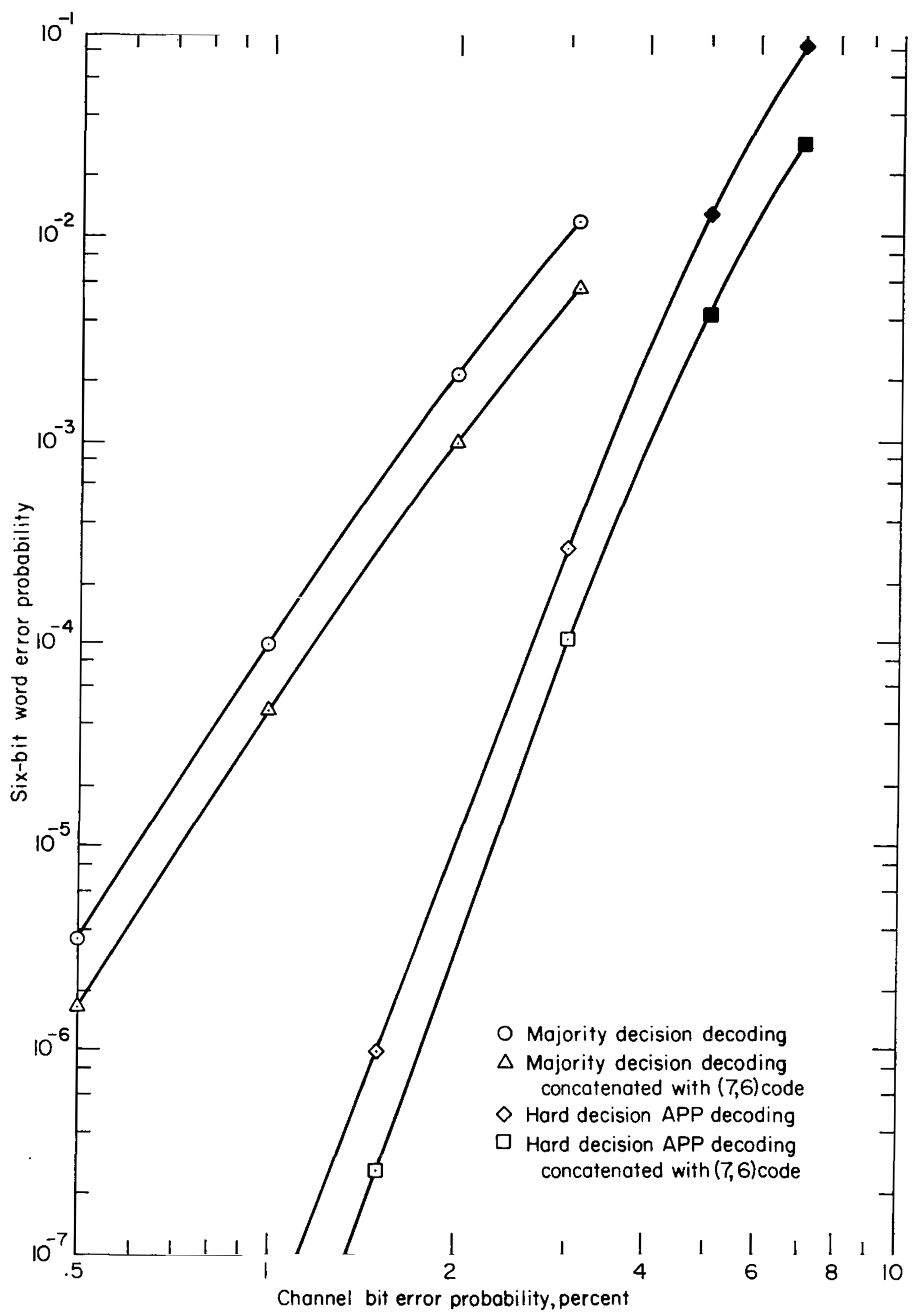


Figure 21.- (44,22) convolutional code - six-bit word error rates.

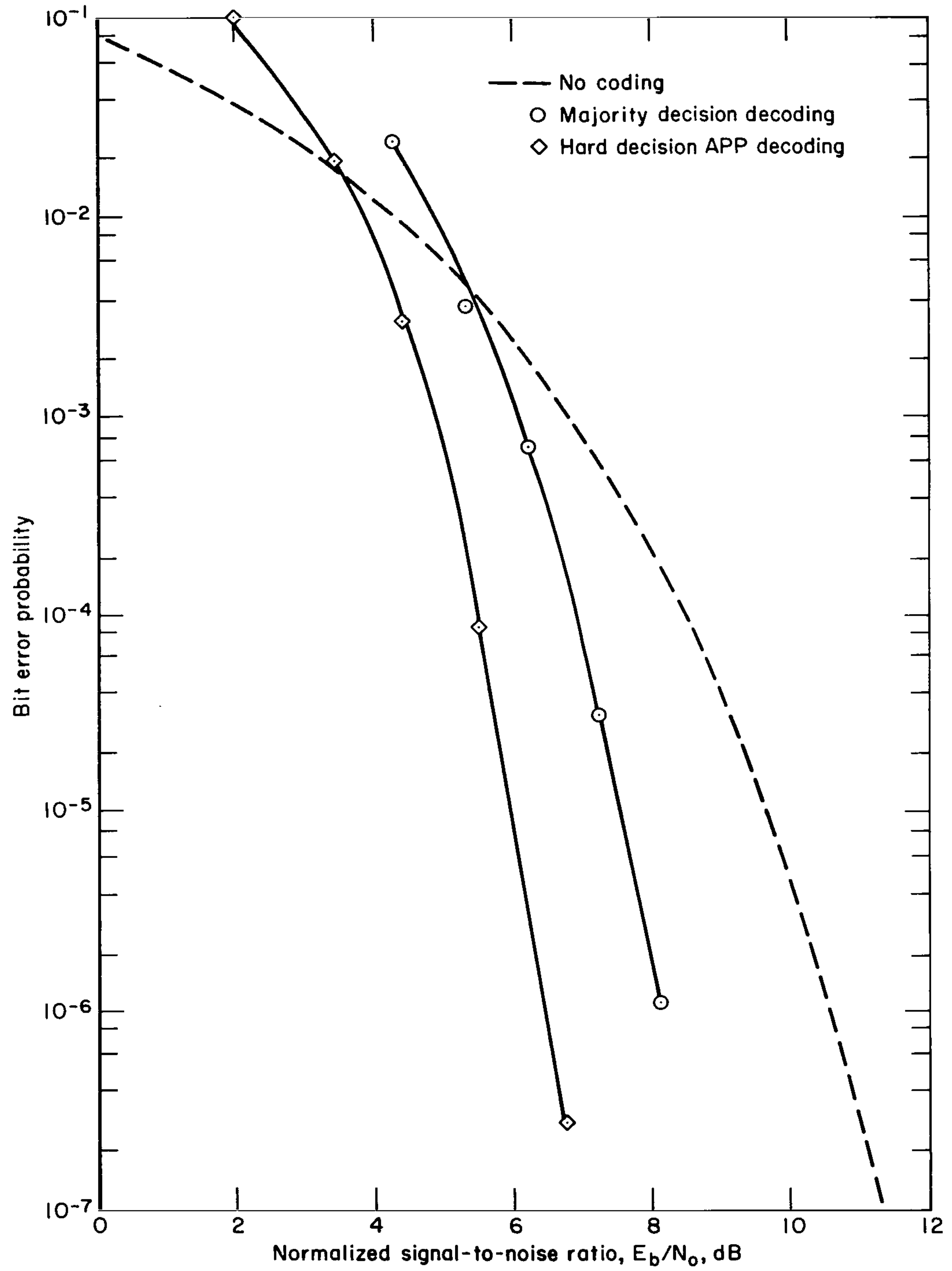


Figure 22.- (44,22) convolutional code - bit error rate performance.

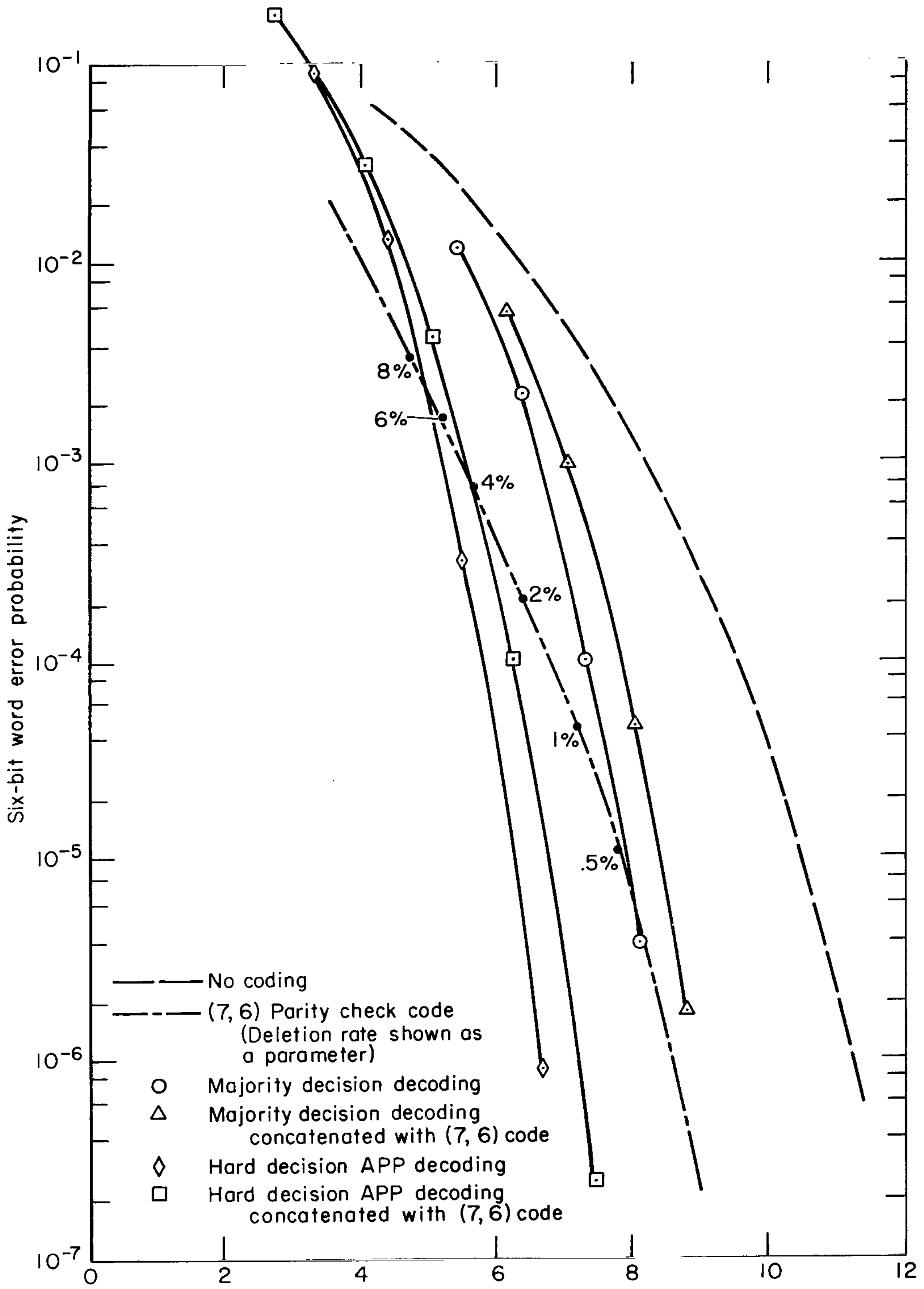


Figure 23.- (44,22) convolutional code - six-bit word error rate performance.

Number of input errors per potential error-causing event	Average number of output errors per potential error-causing event	Percent of decoded potential error-causing events with no error out	Symbol
5	1.74	58.5	□
6	3.3	29.2	○
7	4.6	11.0	△

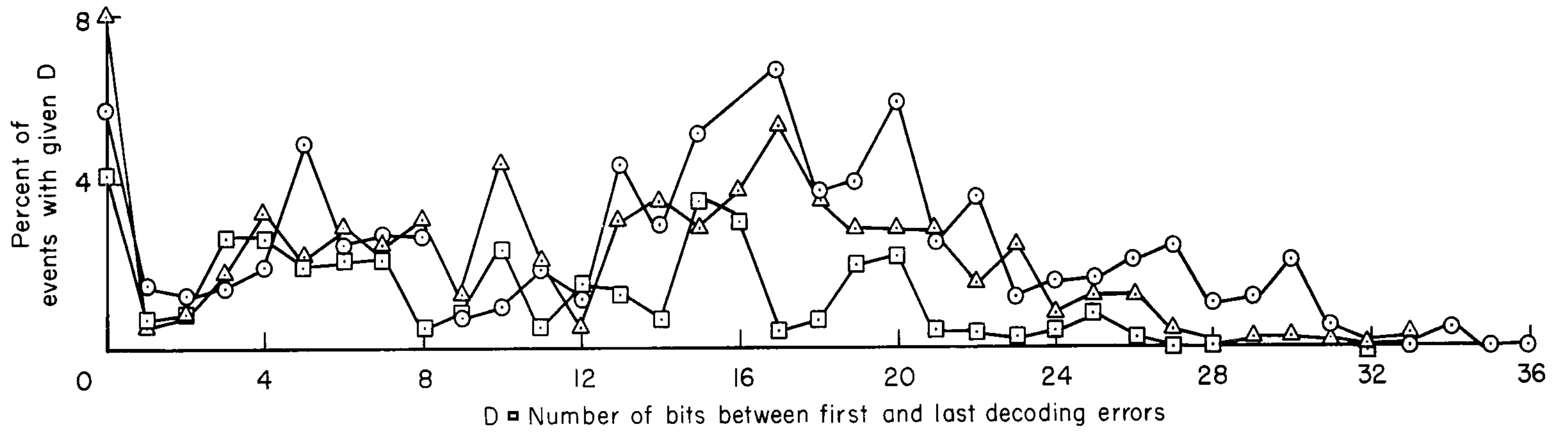


Figure 24.- Error dispersion for the (44,22) convolutional code majority decision decoded with alarm.

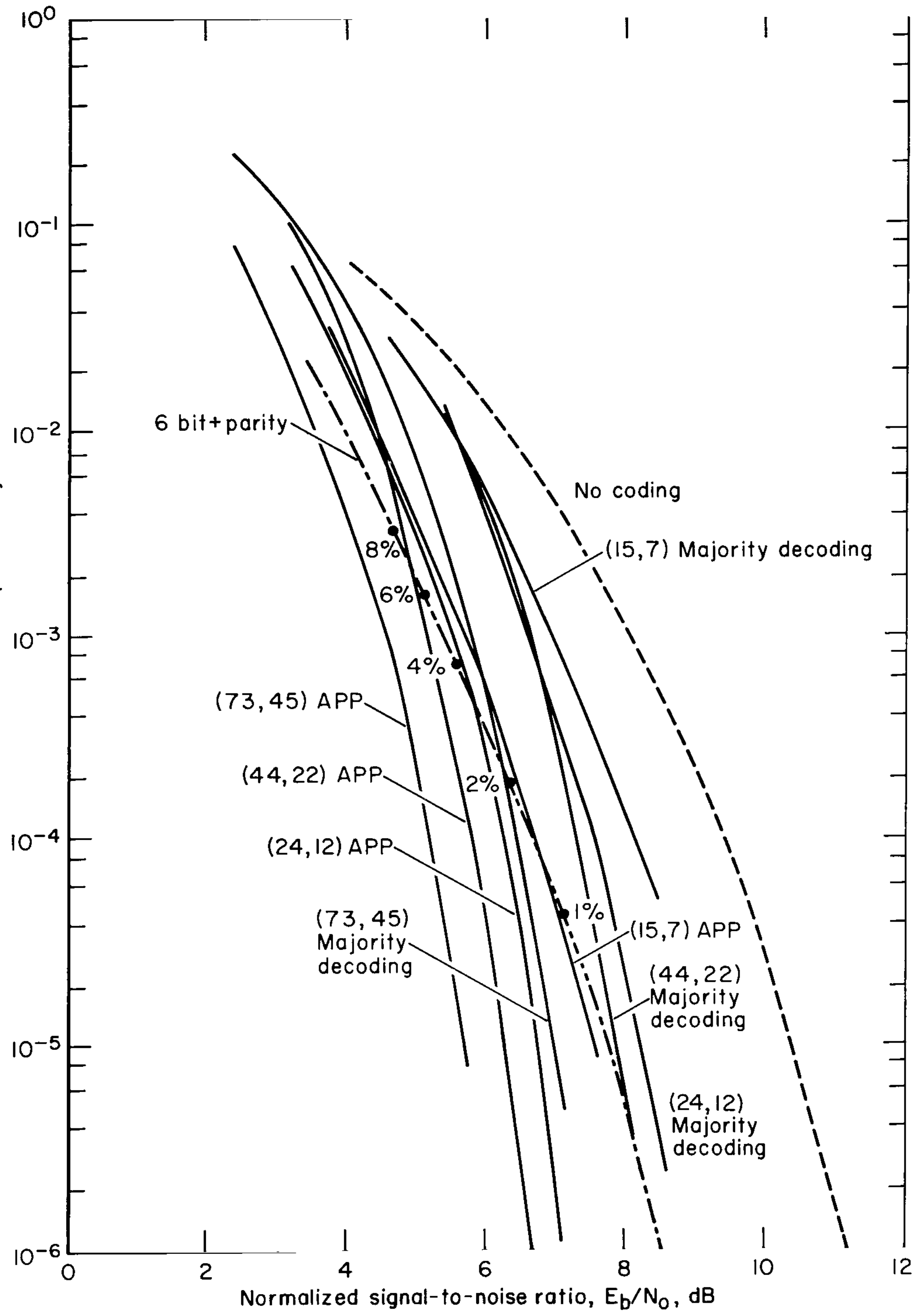


Figure 25.- Performance of four error correction codes with two different decoding methods.

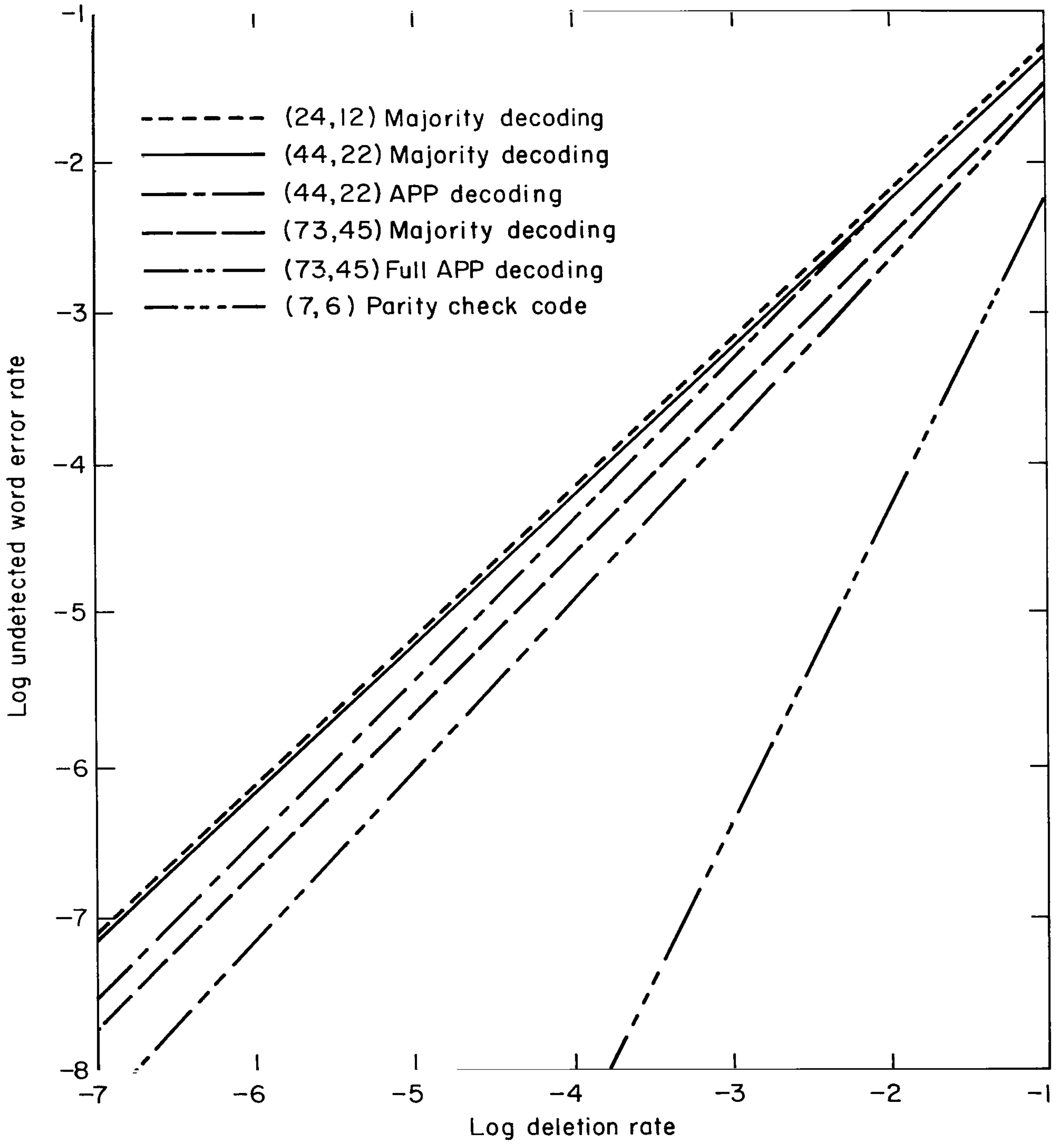


Figure 26.- Detected versus undetected word errors for several codes and decoding methods.

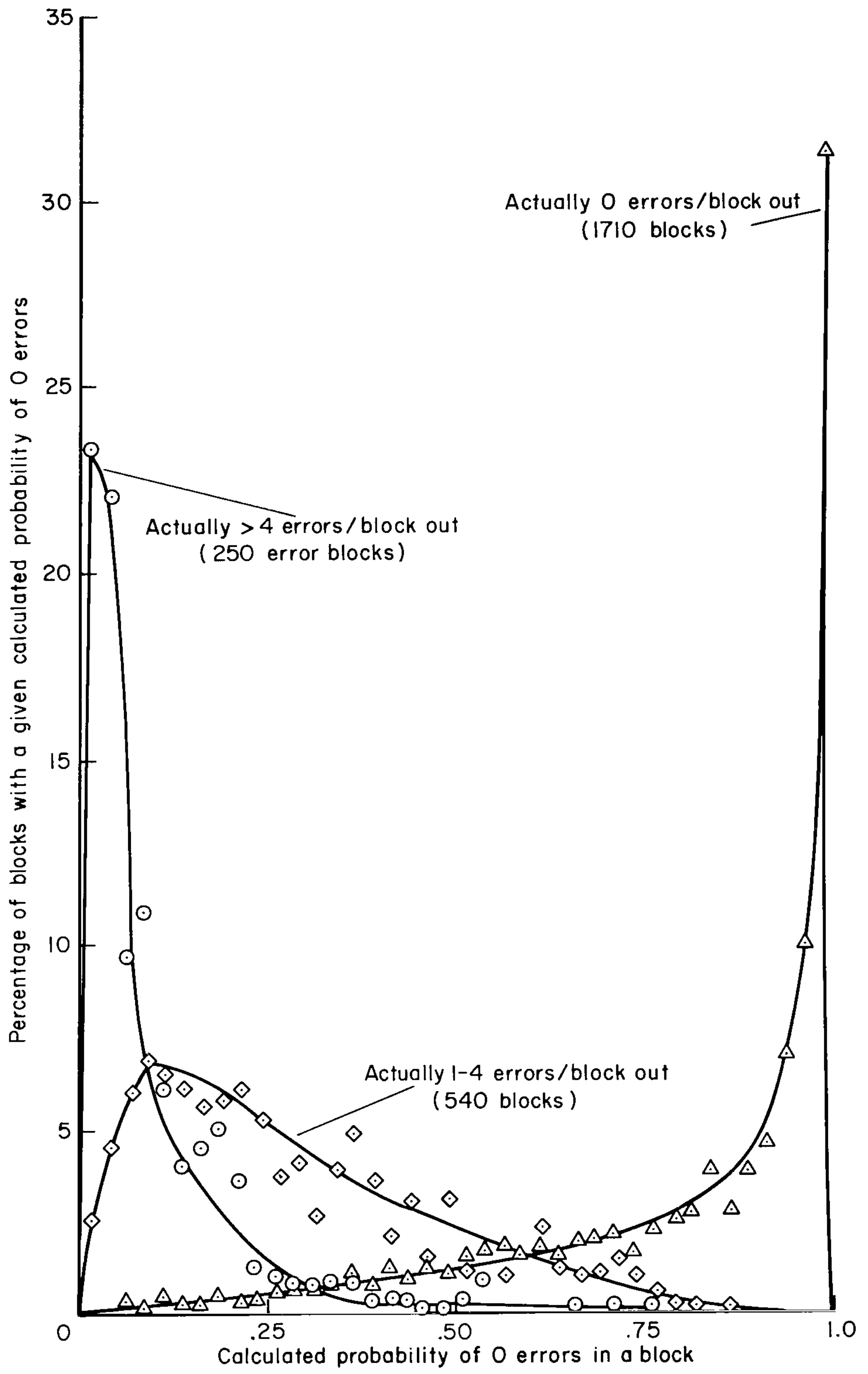
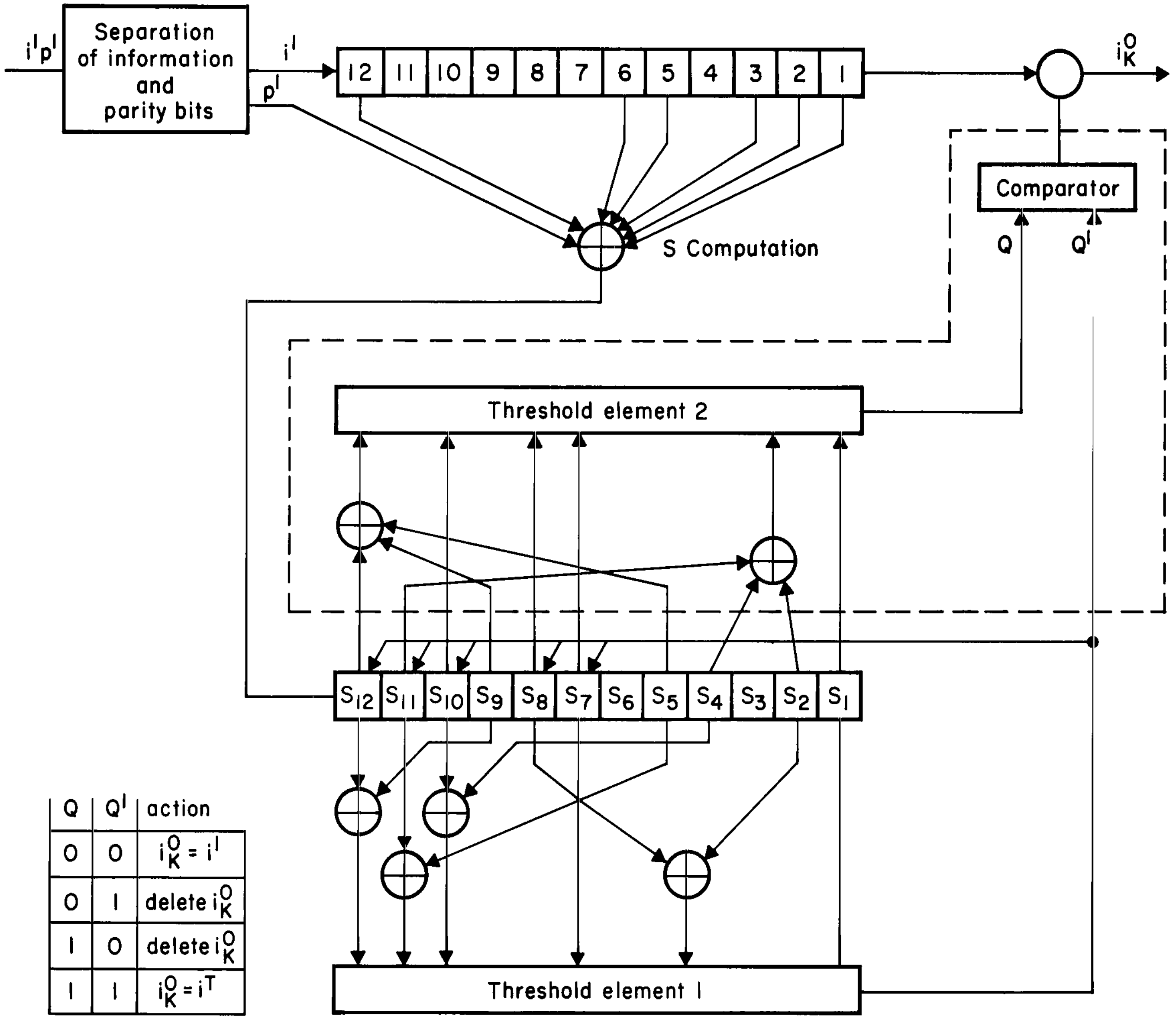
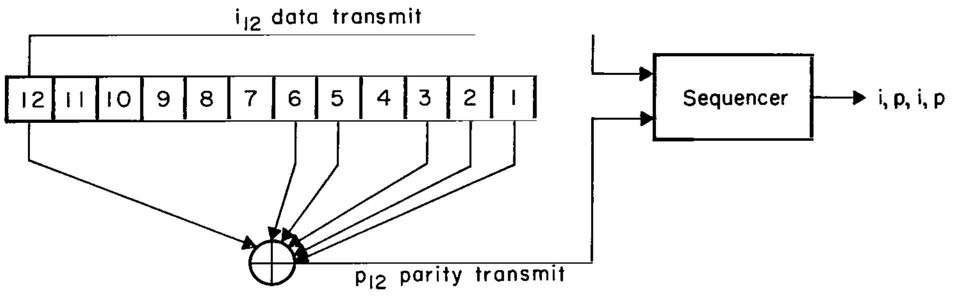


Figure 27.- Distributions of calculated P(OE) for different counted numbers of output errors per block (continuously generated error stream at p_e = 7 percent).

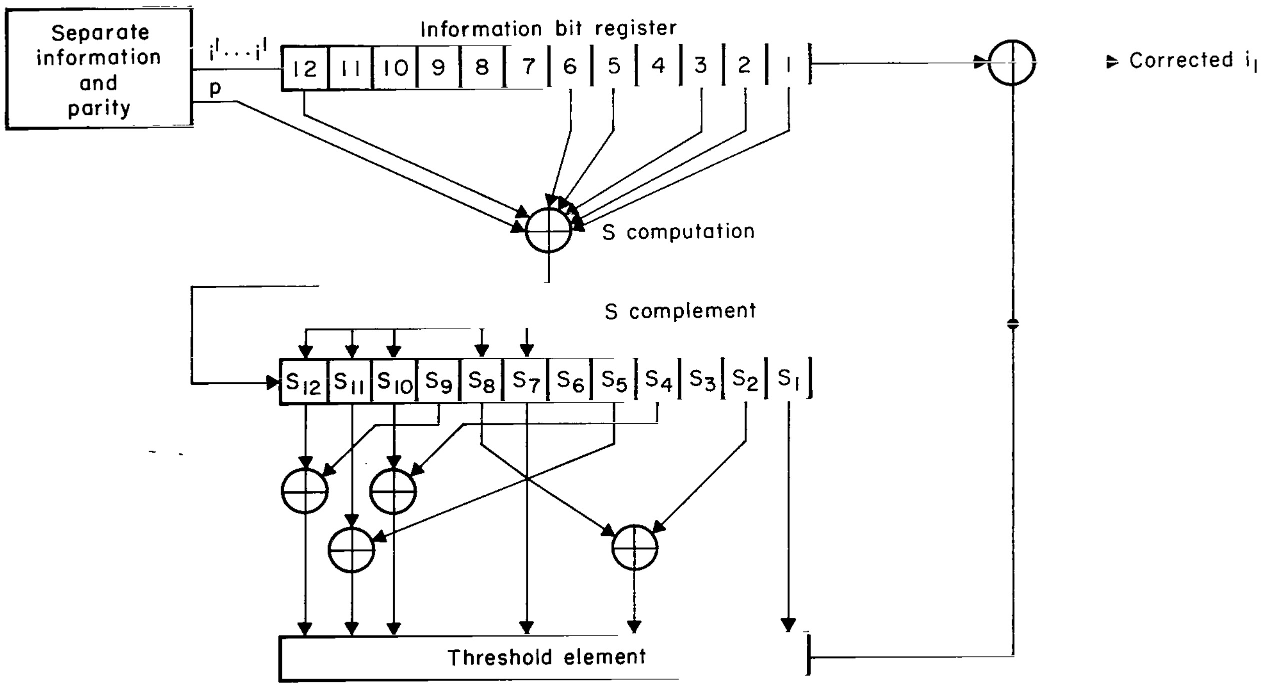

$i^I p^I$
Separation of information and parity bits
i^I
p^I
12 11 10 9 8 7 6 5 4 3 2 1
i_K^O
Comparator
Q
Q^I
S Computation
Threshold element 2
S_{12} S_{11} S_{10} S_9 S_8 S_7 S_6 S_5 S_4 S_3 S_2 S_1
Threshold element 1

Q	Q^I	action
0	0	$i_K^O = i^I$
0	1	delete i_K^O
1	0	delete i_K^O
1	1	$i_K^O = i^T$

Figure 28.- Simplified output error detection circuit for the (24,12) code.



(a) Encoder.



(b) Majority decision decoder.

Figure 29.- Block diagram of the (24,12) encoder and majority decision decoder.

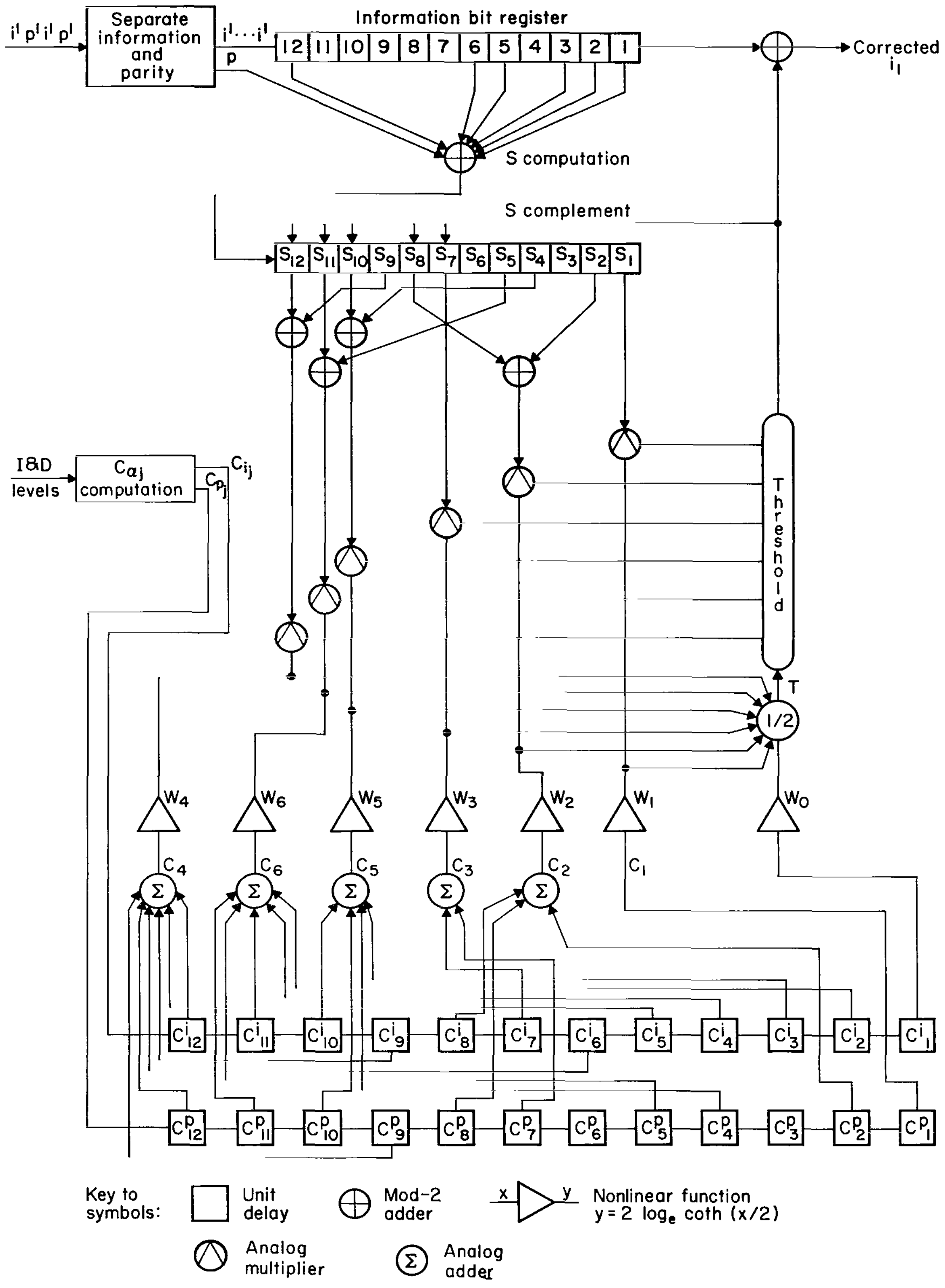
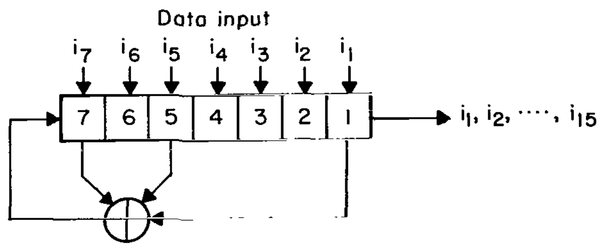
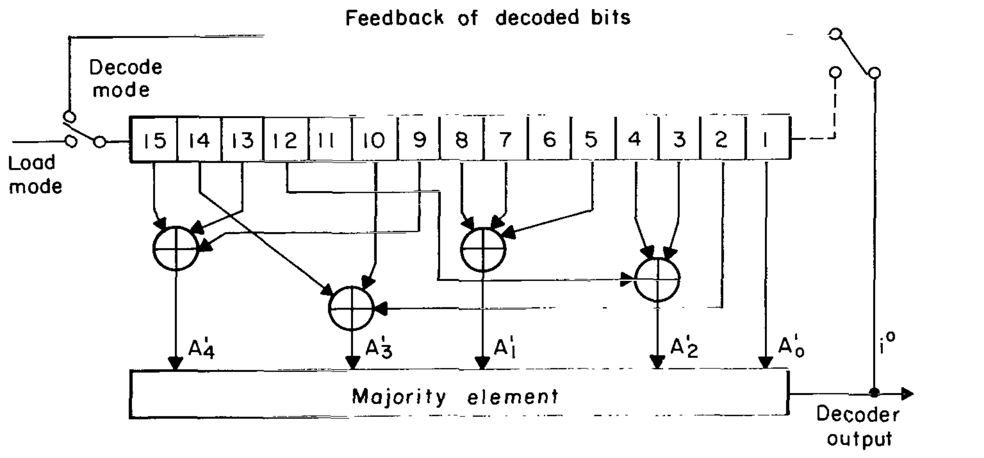


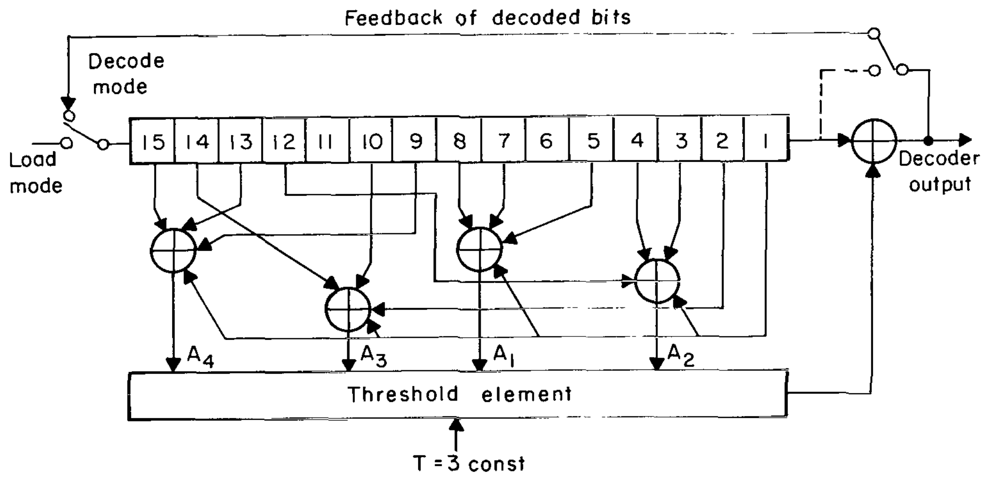
Figure 30.- Complete hard decision feedback APP decoder for the (24,12) convolutional code.



(a) Encoder.



(b) Majority decision decoder.



(c) Majority decision decoder employing a threshold element.

Figure 31.- Encoder and majority decoder for the (15,7) Bose-Chaudhuri code.

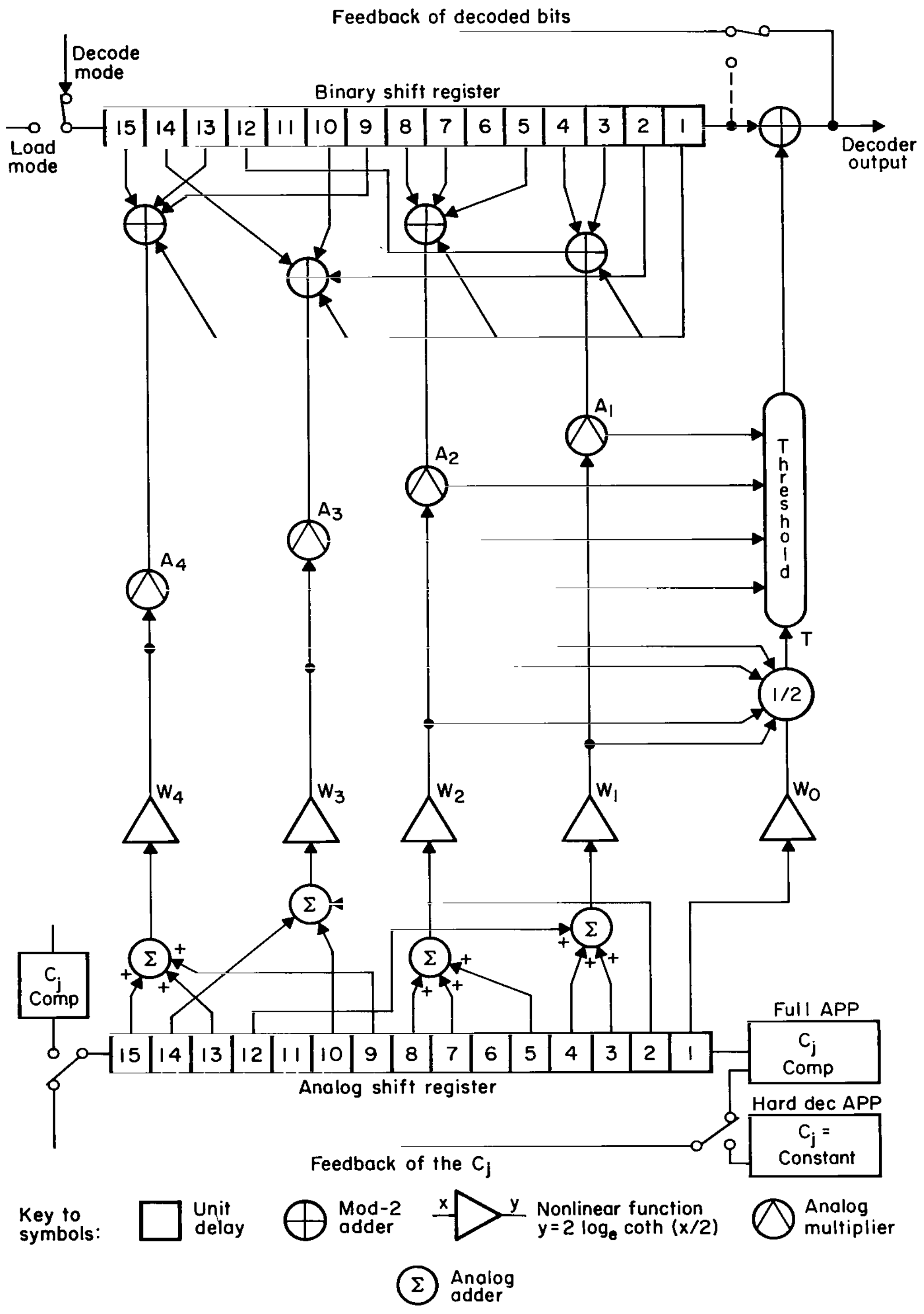


Figure 32.- Complete APP decoding circuit for the (15,7) Bose-Chaudhuri code.

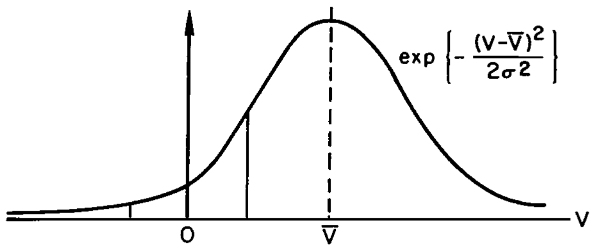


Figure 33.- Bit error probability calculations.

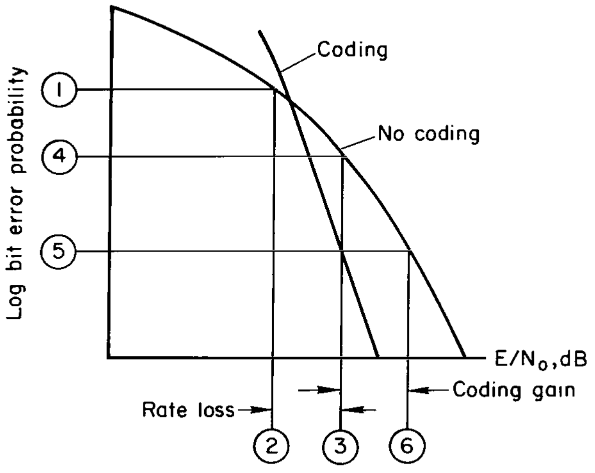
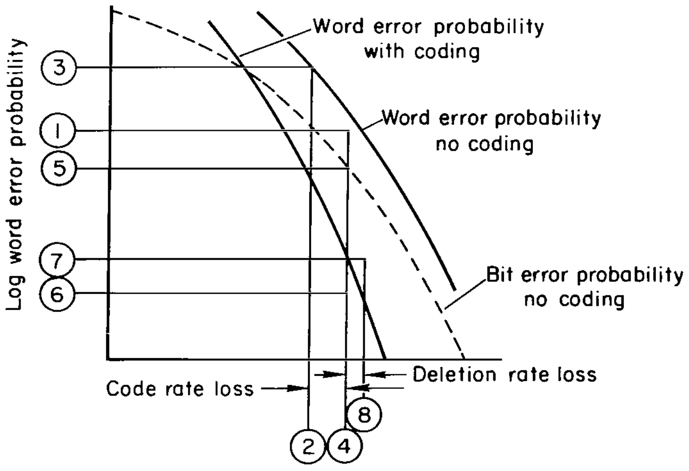


Figure 34.- Coding performance calculations on a bit error probability basis.



Log word error probability
Word error probability with coding
Word error probability no coding
Bit error probability no coding
Code rate loss
Deletion rate loss
3
1
5
7
6
2
4
8

Figure 35.- Coding performance on a word error probability basis.

NASA-Langley, 1968 —— 7 A-2752

```
04U 001 32  51 30S          68059  00903
AIR FORCE WEAPONS LABORATORY/AFWL/
KIRTLAND AIR FORCE BASE, NEW MEXICO 87117


ATT MISS [illegible] F. CANOVA, CHIEF TECHNI
LIBRARY /WLIL/
```

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—National Aeronautics and Space Act of 1958