Lehigh University
Lehigh Preserve

Theses and Dissertations

1-1-1976

Sub-optimum sequential receivers for coded digital data and channels with intersymbol interference.

Clark D. Hafer

SUB-OPTIMUM SEQUENTIAL RECEIVERS FOR CODED DIGITAL DATA AND CHANNELS WITH INTERSYMBOL INTERFERENCE

by

Clark D. Hafer

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

1976

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Sept. 13, 1976
(date)

Professor in Charge

Chairman of Department

ACKNOWLEDGMENTS

In acknowledgment of his help and encouragement, I wish to thank my advisor, Professor Bruce D. Fritchman. I also extend my thanks to Professor Joseph C. Mixsell who provided additional insight and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

New simulation results presented herein indicate that certain sub-optimum forms of a nonlinear sequential receiver, which is used to jointly detect and decode high-speed digital data transmitted through noisy channels with intersymbol interference, will outperform an optimum linear receiver. Three methods of achieving near-optimum performance from a sequential receiver having only a fraction of the calculations of the optimum sequential receiver are discussed. The first eliminates marginal calculations based on a probability threshold criterion, the second based on a noise tolerance criterion, and the third ranks the decision statistics. The simulated performance of the sub-optimum receivers means a real software or hardware implementation is no longer impractical due to lengthy calculations or large data storage problems.

SUB-OPTIMUM SEQUENTIAL RECEIVERS FOR CODED DIGITAL DATA AND CHANNELS WITH INTERSYMBOL INTERFERENCE

1. INTRODUCTION.

When high-speed digital data is transmitted through noisy narrow-bandwidth channels, adjacent pulses begin to overlap. This phenomenon, called intersymbol interference, may severely affect the reliability of a communications system. There are several methods, however, of compensating for intersymbol interference. By designing a receiver with some knowledge of the transmitted symbol probabilities, as well as the channel characteristics, the probability of receiver error can be held to a minimum.

Several optimum receivers have been proposed recently, but all of them suffer from being too complex to implement economically for long codes or channels with severe interference. This study attempts to simplify the non-linear sequential receiver proposed by Abend and Fritchman [1], and the joint sequential receiver derived from it, which simultaneously detects and decodes convolutionally encoded data. The optimum performance of the joint receiver has previously been studied by Sattar [2], and his results are used as a yardstick for comparison of the sub-optimum results derived herein.

Chapter 2 briefly examines the history of optimum-receiver development, and explains why a sub-optimum receiver, rather than an optimum one, is generally desirable for practical application.

Chapter 3 develops the sequential receiver of Abend and Fritchman, beginning with the basic communications channel model. Chapter 4 adds convolutional coding to the transmitted source bits, which then requires an optimized decoder to be appended to the optimized detector discussed in Chapter 3.

Chapter 5 demonstrates how the separately-optimized detector-decoder can be greatly improved by a joint detector-decoder algorithm.

Chapter 6 contains the simulation results of three attempts at reducing the complexity of the joint receiver. The results indicate that even though performance is degraded below optimum for the joint receiver, the sub-optimum joint receiver still out-performs the separately-optimized receiver, with considerably less complexity and fewer calculations.

Chapter 7 summarizes the results of Chapter 6, attempts to choose the best sub-optimum scheme of the three examined, and concludes with suggestions for further study.

Details on the computer simulations appear in Appendix A.

2. TYPES OF RECEIVERS.

Intersymbol interference is the major hindrance to high data rates in typical wireline and radio data channels. Significant research has led to various schemes of minimizing the effects of the interference. These schemes can be broadly lumped into two classes, linear and nonlinear receivers.

The class of linear receivers is attractive from the standpoint that they can be described and evaluated analytically. Also, their implementation is straightforward, and hence they are frequently used in real applications.

The idea behind the linear receivers is to flatten out the amplitude and delay distortions which naturally occur in a real channel, so that the net affect of the channel and receiver approaches an ideal linear-amplitude-and-phase frequency response. This process, called equalization, is based on the fact that samples every T seconds from a receiving filter matched to the transmitting filter and channel characteristics constitute a sufficient set of statistics for estimating the input sequence [3].

A transversal equalizer is a tapped delay line that approximates the required matched filter. The process of adjusting the tap coefficients to a specific channel was a tedious manual process until algorithms introduced in 1965 [4],[5] provided automatic adjustment. Further improvements in 1966 [6] provided the ability to track time-varying channel coefficients.

A linear feedback equalizer is similar to the transversal equalizer except that intermediate outputs from the tapped delay line are fed backward as well as forward. The result is a small improvement in performance, but not a significant one.

Normally, the tap coefficients would be chosen to minimize P(E), the average probability of error [7]. But P(E) is such a nonlinear function of these coefficients that other criterions such as "peak distortion" [4],[6] are used instead.

The class of nonlinear receivers is based on efforts to use P(E) as a performance criterion. These receivers are characterized by excessive data manipulation and defy analytical prediction of their performance.

Fourney [8] has applied the Viterbi algorithm to processing samples from a whitened matched filter, and has obtained tight bounds on its performance. Ungerboeck and Mackechnie have developed a similar receiver [9], but have eliminated the need for a pre-whitening filter. Chang and Hancock [10] have proposed a receiver in which the received symbols are partitioned into overlapping sequences K symbols long. Then the sequences $A_k A_{k+1} A_{k+2} \cdots$ form a Markov chain from which maximum likelihood (ML) decisions are made.

A nonlinear ML receiver which minimizes P(E) on each symbol has been developed by Abend and Fritchman [1]. This receiver sequentially computes the a posteriori decision statistics for each received symbol, making symbol-by-symbol ML decisions after only a short delay D. Because the receiver is recursive, long sequences do not have to be stored, and the receiver remains optimum for any length sequence.

Unfortunately, the sequential receiver grows exponentially as m^D, where m is the size of the source symbol alphabet†. When the source data is convolutionally encoded, the receiver becomes a detector-decoder pair, increasing the complexity by that of the decoder. Because of the similarity between the optimum detector and the optimum decoder algorithms, however, a joint detector-decoder algorithm can be derived without much more complexity than either of the separate parts [2].

Simulation results indicate that the sequential

†Actually, the complexity increases as $m^L + (D-L)m$ for $D > L$, L is the effective duration of the interference.

detector is superior to the class of linear receivers [1], but lacks the simplicity of a linear receiver. Further results have shown that the optimum joint detector-decoder also does better than the separately optimized case [2]. This paper is motivated, then, by the possibility of reducing the complexity of the joint sequential receiver to a practical level, yet maintaining an edge in performance above what the separately optimized detector and decoder can achieve.

Linear equalizers, while mathematically tractable and practical to implement, are not optimum due to their tuning techniques; the "peak distortion" criterion is an example. The optimum nonlinear receivers are too complex to be practical. Hence a sub-optimum receiver results. The next several chapters provide the background needed to understand the reduced complexity sequential receivers of Chapter 6.

3. OPTIMUM SEQUENTIAL DETECTOR.

The basic model for a communications system with independent (non-coded) source symbols is shown in Figure 3.1.

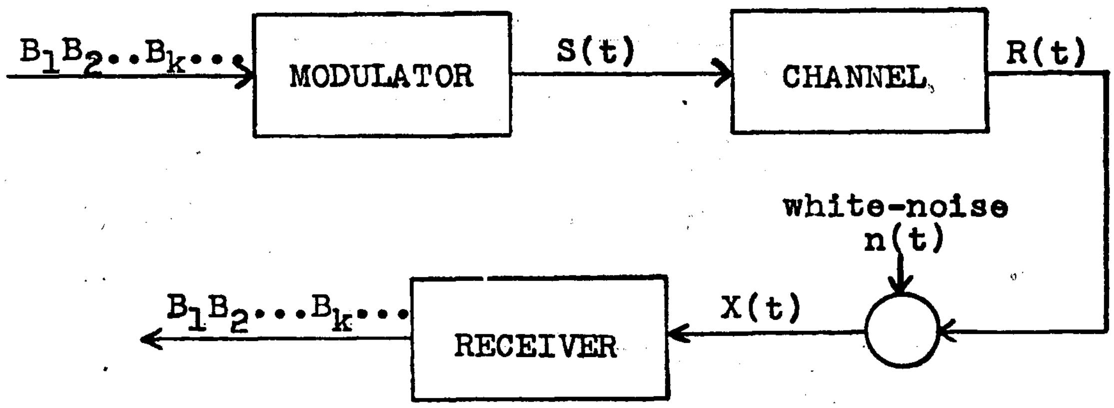


Fig. 3.1. Basic Communication System.

The source symbols are assumed to be binary for our purposes, although the m-ary case is easily derived. The ones and zeros from the data source are then passed through the digital data modulator. Here we will assume pulse-amplitude modulation (PAM), so the signal $S(t)$ becomes a train of pulses each of amplitude -1 or 1 and of T seconds duration. That is,

$$S(t) = \sum_k A_k \, g(t-kT) \tag{3.1}$$

where $A_k = 1$ if $B_k = 1$, $A_k = -1$ if $B_k = 0$, and $g(t)$ is a unit pulse T seconds long.

The finite bandwidth of the transmission channel causes adjacent pulses to overlap at the output. For a perfect Nyquist channel, this is no problem, because the channel is then sampl d such that all interfering terms are zero. But all real channels are subject to phase delays and other perturbations, causing intersymbol interference.

If the impulse response of the channel, for example, is as shown in Fig. 3.2, then the sampled value R_k is given by

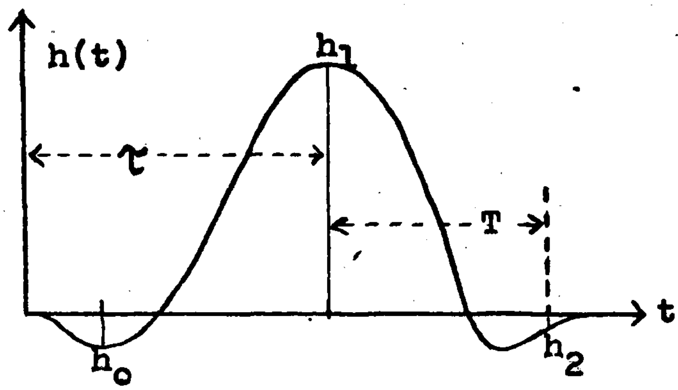


Fig. 3.2. Sample channel response.

$$R_k = B_k h_0 + B_{k-1} h_1 + B_{k-2} h_2$$

$$\text{or} \quad R_k = B_k h_0 + B_{k-1} h_1 + \ldots + B_{k-L+1} h_{L-1} \tag{3.2}$$

in general, for an impulse response L samples long. Intersymbol interference occurs when more than one of the h_i's are non-zero. The delay τ allows both future and past symbols to interfere.

The standard assumption of additive white Gaussian noise completes the channel model, so that the received signal becomes

$$X(t) = R(t) + N(t) \tag{3.3}$$

$$\text{or} \quad X_k = R_k + N_k \tag{3.4}$$

for statistically independent noise sarples.

Actually, "colored" noise can also be handled if a noise-whitening filter is added to the front end of the receiver in Fig. 3.1.

The basic problem this model presents is designing a receiver to produce an estimate $\hat{B}_k$ of B_k such that the average probability of error is a minimum. The sequential detector of Abend and Fritchman is an optimum receiver when $\hat{B}_k$ depends on no more than $X_1 X_2 \ldots X_{k+D}$, where D is the time delay before making a decision on B_k.

The decision, for our binary example, is to choose $B_k = b_i$ when

$$P(B_k = b_i \mid X_1 \ldots X_{k+D}) \geqslant P(B_k = b_j \mid X_1 \ldots X_{k+D})$$
$$b_i, b_j \in \left\{1, -1\right\}, \quad b_i \neq b_j \tag{3.5}$$

This is identical to calculating the probabilities $p(B_k, X_1 \ldots X_{k+D})$ because in

$$P(B_k \mid X_1 \ldots X_{k+D}) = p(B_k, X_1 \ldots X_{k+D})/p(X_1 \ldots X_{k+D}), \tag{3.6}$$

the term $p(X_1 \ldots X_{k+D})$ is a common proportionality constant. By noting that the input symbols are independent, and that X_k depends only on the L values $B_{k-L+1} \ldots B_k$, i.e.,

$$p(X_k \mid B_1 \ldots B_k, X_1 \ldots X_{k-1}) = p(X_k \mid B_{k-L+1} \ldots B_k), \tag{3.7}$$

then we can recursively calculate

$$p(B_1, X_1) = P(B_1)p(X_1 \mid B_1)$$

$$
\begin{aligned}
p(B_1B_2,X_1X_2) &= p(X_2|B_1B_2,X_1)p(B_1B_2,X_1)\\
& p(X_2|B_1B_2)P(B_2|B_1,X_1)p(B_1,X_1)\\
& P(B_2)p(X_2|B_1B_2)p(B_1,X_2)
\end{aligned}
$$

$$
\begin{aligned}
&p(B_1B_2B_3,X_1X_2X_3)\\
&\qquad = P(B_3)p(X_3|B_1B_2B_3)p(B_1B_2,X_1X_2)\\
&\qquad\qquad \vdots\\
&p(B_k\ldots B_{k+D},X_1\ldots X_{k+D})\\
&\qquad = P(B_{k+D})p(X_{k+D}|B_{k+D-L+1}\ldots B_{k+D})\\
&\qquad\quad \cdot \sum_{B_{k-1}} p(B_{k-1}B_k\ldots B_{k+D-1},X_1\ldots X_{k+D-1})
\end{aligned}
\tag{3.8}
$$

from which

$$
p(B_k,X_1\ldots X_{k+D}) = \sum_{B_{k+1}}\ldots\sum_{B_{k+D}} p(B_k\ldots B_{k+D},X_1\ldots X_{k+D}). \tag{3.9}
$$

For binary equally-likely source symbols, the term $P(B_{k+D})$ of (3.8) will always be 1/2. The third term, in the summation, is known from the calculations for the previous symbol. Finally, the second term is calculated for all 2^L sequences $B_{k+D-L+1}\ldots B_{k+D}$ by noting that

$$
p(X_k|B_{k-L+1}\ldots B_k) = f(X_k-R_k) \tag{3.10}
$$

and that $f(\cdot)$ is the noise probability density.

Equations (3.8) and (3.9) constitute the core of the sequential detection algorithm in [1], and also serve as a decoding algorithm for convolutional codes, with only slight modification, as the next chapter will show.

4. OPTIMUM DETECTOR PLUS OPTIMUM DECODER.

Shannon has shown that data sequences, when properly coded, can reduce the probability of transmission error to zero. Of course, an infinitely long code generator would be needed, not to mention the more difficult decoding problem. But even short coding techniques can be used to achieve higher reliability without too much additional cost.

A convolutional coder consists of ν shift registers and n modulo-two adders. Figure 4.1 shows such a coder with $\nu = 3$ and $n = 2$.

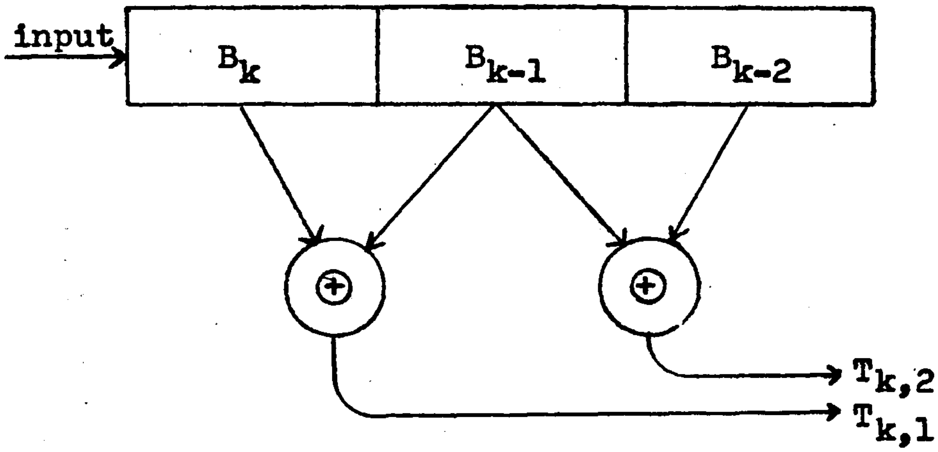


Fig. 4.1. Convolutional coding.

This coder can be represented by the code generator matrix

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} .$$

In general, if $g_{i,j} = 1$, there is a connection between the i^{th} shift register and the j^{th} modulo-two adder.

There are n outputs (rate 1/n) every T seconds when a new source symbol is shifted in. These can be computed as

$$
\begin{aligned}
T_{k,1} &= B_k g_{11} \oplus B_{k-1} g_{21} \oplus \cdots \oplus B_{k-\nu+1} g_{\nu 1} \\
&\;\;\vdots \\
T_{k,n} &= B_k g_{1n} \oplus \cdots \qquad \cdots \oplus B_{k-\nu+1} g_{\nu n} \;. \qquad (4.1)
\end{aligned}
$$

The nature of this coding technique makes decoding it very similar to detecting data in the presence of intersymbol interference, since the outputs $T_{k,1} \cdots T_{k,n}$ depend not only on B_k, but on ν-1 past symbols as well.

The decoder functions analogously to equation (3.8), only now the X_k's are replaced by the vectors

$$\underline{T}_k = (T_{k,1}, \ldots, T_{k,n}) \qquad (4.2)$$

and the necessary joint probabilities are calculated following a delay of d input symbols ($d \geqslant \nu$)

$$
\begin{aligned}
p(B_k \cdots B_{k+d}, \underline{T}_1 \cdots \underline{T}_{k+d}) \\
&= P(B_{k+d}) p(\underline{T}_{k+d} | B_{k+d-\nu+1} \cdots B_{k+d}) \\
&\cdot \sum_{B_{k-1}} p(B_{k-1} B_k \cdots B_{k+d-1}, \underline{T}_1 \cdots \underline{T}_{k+d-1}) .
\end{aligned} \qquad (4.3)
$$

In this case, the second term can be calculated as

$$
\begin{aligned}
P(\underline{T}_{k+d} | B_{k+d-\nu+1} \cdots B_{k+d}) &= P(\underline{T}_k | \underline{t}_i) \\
&= \prod_{j=1}^{n} P(T_{k,j} | t_{i,j}) ,
\end{aligned} \qquad (4.4)
$$

where $i = 1,2,\ldots,2^{\nu}$. That is, there are 2^{ν} possible sequences $\underline{t}_i = t_{i1}t_{i2}\ldots t_{i\nu}$ (some of which might be redundant) because there are 2^{ν} possible "states" of the shift registers. Each individual probability $P(T_{k,j}|t_{i,j})$ is either p, or 1-p, when we assume the channel to be binary symmetric with cross-over probability p. If $\underline{T}_k = \underline{t}_i$, then $P(T_k|t_i) = (1-p)^n$.

The communication model, with the addition of convolutional coding, appears in Fig. 4.2.

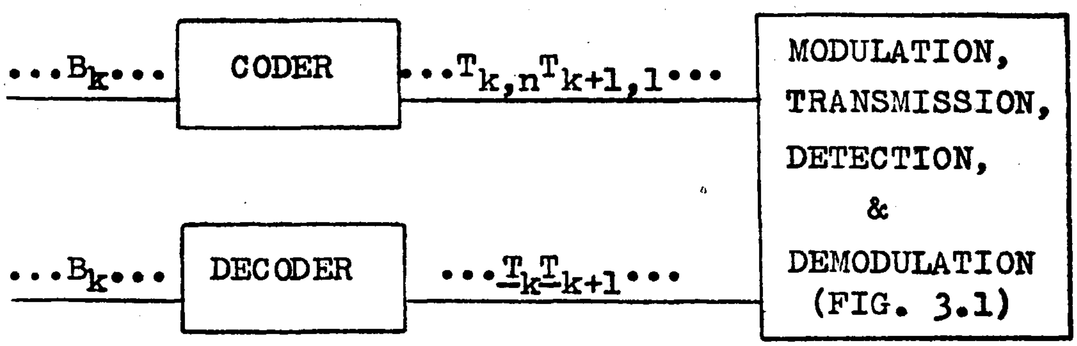


Fig. 4.2. Channel with coded symbols.

In this case, the model of Fig. 3.1 accepts the binary symbols $\ldots T_{k-1,n}T_{k,1}T_{k,2}\ldots$ as if they were independent, producing ML estimates $\ldots\hat{T}_{k-1,n}\hat{T}_{k,1}\hat{T}_{k,2}\ldots$ which are then processed by the decoder. The decoder produces one source-symbol estimate, $\hat{B}_k$, for every n detected symbols $\hat{T}_{k,j}$, or alternatively, for every vector $\hat{\underline{T}}_k$.

The detector of the previous chapter must delay its decision L-1 symbols $\hat{T}_{k,j}$, while the convolutional decoder must wait for $\nu \cdot n$ of these symbols. The result is an effective delay before estimating B_k of

$$D_{eff} = \nu + \left\lceil \frac{L-1}{n} \right\rceil \tag{4.5}$$

time intervals T, when the rate of the B_k's is $1/T$. The quantity $\left\lceil \frac{L-1}{n} \right\rceil$ is the least integer $\geqslant \frac{L-1}{n}$. An example makes this clearer. If $\nu = 3$, $n = 2$, and $L = 4$, then the source symbol B_k affects $\underline{T}_k$, $\underline{T}_{k+1}$, and $\underline{T}_{k+2}$, so the decoder must wait $\nu T = 3T$ seconds† until B_k is shifted out of the coder to compute $\hat{B}_k$. Note, however, that $X_{k+2,2}$ depends not only on $T_{k+2,2}$, but on $T_{k+3,1}$, $T_{k+3,2}$, and $T_{k+4,1}$ as well. This represents an additional lag on the system, hence the effective delay becomes

$$D_{eff} = 3 + \left\lceil \frac{4-1}{2} \right\rceil = 4.$$

†Note that it is possible to estimate B_k before its effects die out, for some delay d, $d < \nu$. Indeed, this example also assumes $D = L-1$, although some $D \leqslant L-1$ might perform nearly as well for negligible intersymbol interference. For the purposes of this paper, however, we generally allow $d \geqslant \nu$, $D \geqslant L-1$ to achieve the most favorable error rates.

5. OPTIMUM RECEIVER.

Intuitively, a detector which does not employ all of the information present in the coded symbols it receives will make more errors than one that does. Recall that the separate detector of Chapter 4 bases its decisions only on knowledge of the channel, and not of the code. This intermediate decision, prior to decoding, is a lossy process which can be eliminated by the jointly optimized receiver we shall now describe. The joint receiver estimates the original source symbols directly from the $\underline{X}_k$'s, rather than first making a bit-by-bit decision $\hat{T}_{k,1}, \hat{T}_{k,2} \ldots$ followed by a decoding process.

The procedure is the vector-extension of the scalar equations (3.8) and (3.9):

$$p(B_k, \underline{X}_1 \ldots \underline{X}_{k+\delta}) = \sum_{B_{k+1}} \cdots \sum_{B_{k+\delta}} p(B_k \ldots B_{k+\delta}, \underline{X}_1 \ldots \underline{X}_{k+\delta}) \qquad (5.1)$$

and

$$\begin{aligned} p(B_k \ldots B_{k+\delta}, \underline{X}_1 \ldots \underline{X}_{k+\delta}) &= P(B_{k+\delta})\, p(\underline{X}_{k+\delta} \mid B_{k+\delta-\ell+1} \ldots B_{k+\delta}) \\ &\quad \cdot \sum_{B_{k-1}} p(B_{k-1} B_k \ldots B_{k+\delta-1}, \underline{X}_1 \ldots \underline{X}_{k+\delta-1}). \end{aligned} \qquad (5.2)$$

The first term is again known to be 1/2 for our binary data. The third term is the stored value from the previous iteration, and the second term is now the product (assuming independent noise samples)

$$p(\underline{X}_{k+\delta} \mid B_{k+\delta-\ell+1} \ldots B_{k+\delta}) = \prod_{j=1}^{n} f(N_{k+\delta,j}). \qquad (5.3)$$

Again, there is a delay, δ , such that $B_{k+\delta}$ is transmitted before decision on B_k. The length ℓ is the effective overall constraint length, and is given by

$$\ell = \nu + \left\lceil \frac{L-1}{n} \right\rceil \qquad (5.4)$$

for the identical reasons stated for equation (4.5).

The joint algorithm, as expected, shows marked improvement over the separately optimized case. Fig. 5.1 illustrates an improvement of at least 3dB in the signal-to-noise ratio needed to achieve identical error rates, for the sample channel and convolutional code used.

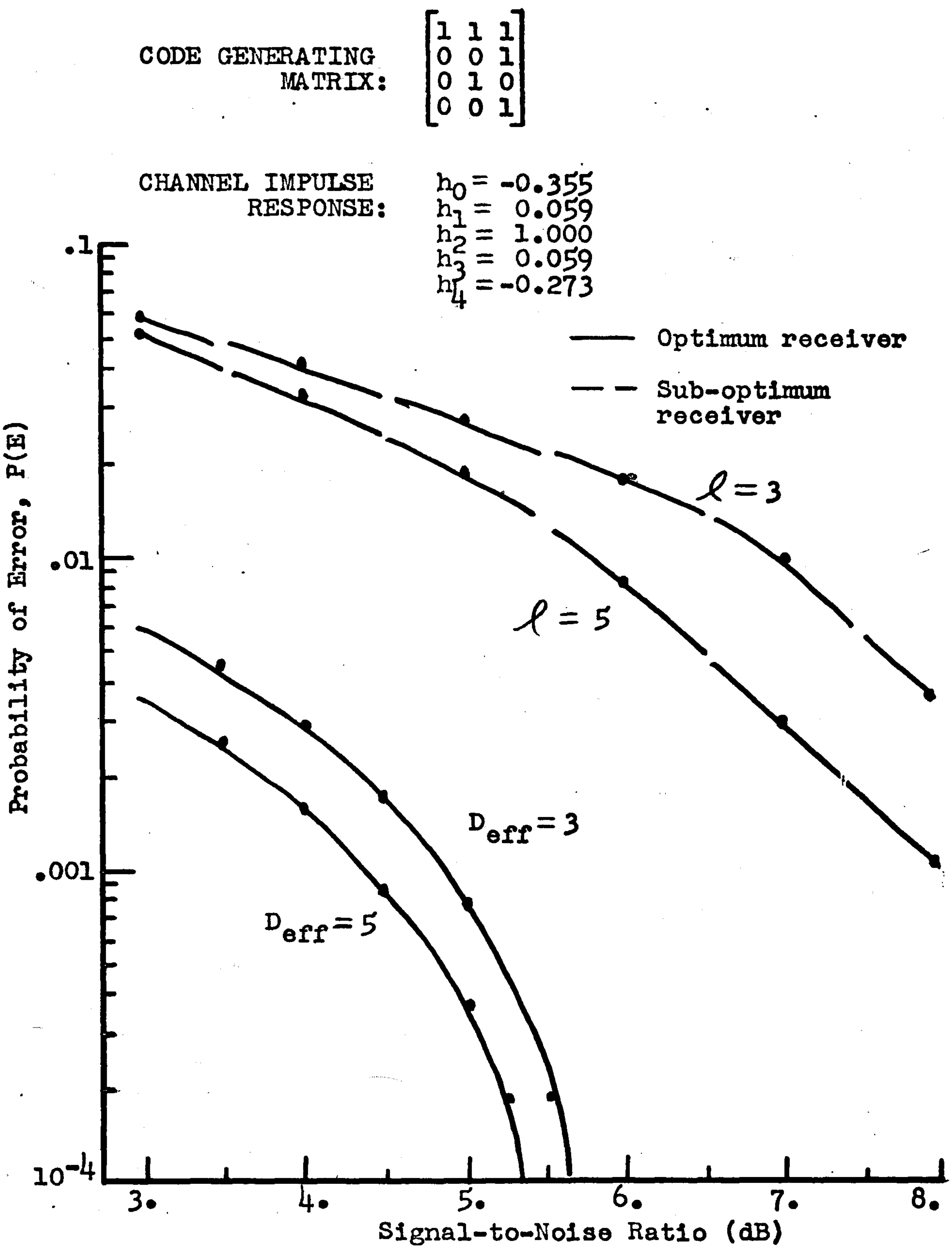


Fig. 5.1. Performance of sub-optimum and optimum receivers [2].

6. ALGORITHMS TO REDUCE THE COMPLEXITY OF THE JOINT SEQUENTIAL COMPOUND DETECTOR-DECODER.

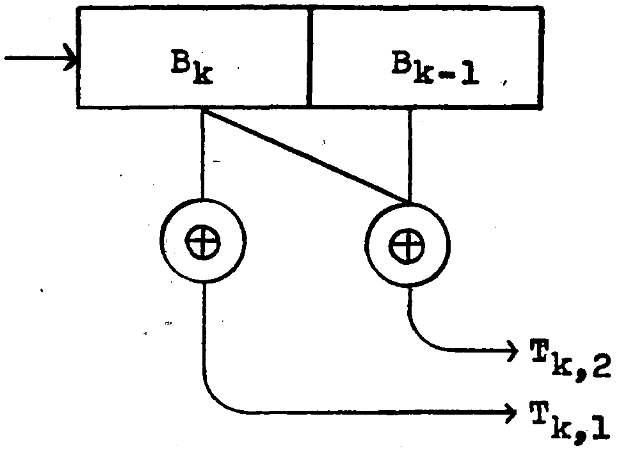
6.1. Motivation.

For binary data transmission, the size of the optimum sequential receiver grows exponentially as 2^{ℓ}, where ℓ is the effective length of the intersymbol interference when the effects of the code constraint length are combined with the channel pulse duration. It would be very desirable to trim the size of the receiver in a way which does not seriously degrade performance, while eliminating much of the required storage (in hardware or in software) and much of the data manipulation needed by the optimum algorithm. If the resulting sub-optimum sequential receiver performs better than the separately optimized detector-decoder pair, then the sub-optimum receiver is judged successful.

6.2. An Example.

To introduce the sub-optimum algorithms, a specific example of the functioning of the optimum joint algorithm will be helpful.

Consider the code generator in Fig. 6.2.1. The code used is rate 1/2 with a constraint length of 2, and is completely specified by the code generator matrix G. Fig. 6.2.2 is a tree which represents the pairs $t_{k,1}$, $t_{k,2}$ transmitted by the coder given any previous state. Moving up one level indicates a zero was shifted



$$G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Fig. 6.2.1. Code generator and generating matrix.

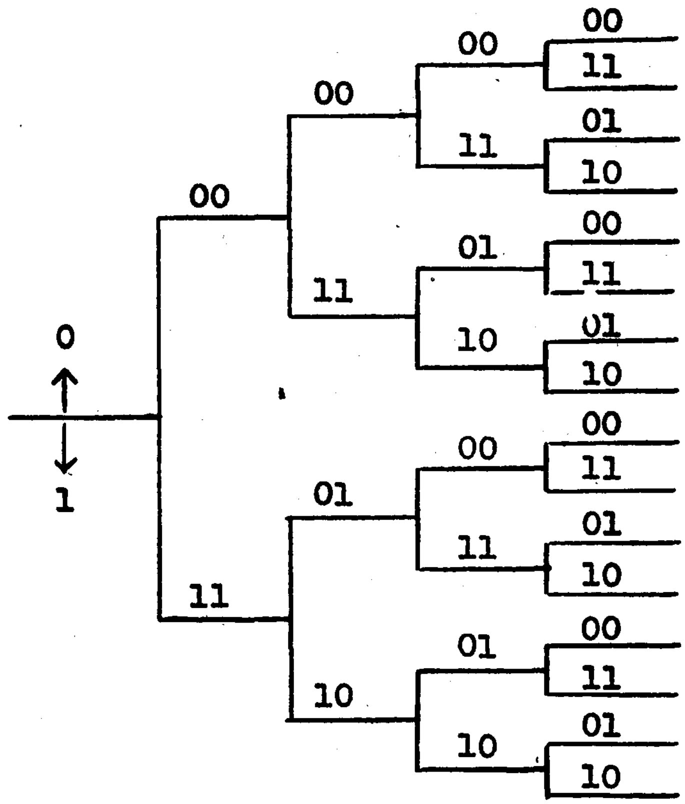


Fig. 6.2.2. Code tree of vectors $\underline{T}_k$.

into the coder, while moving down one level implies a 1 was shifted in. A source-symbol sequence of 0,1,1, for example, would transmit the coded pairs 00,11,01 (after modulation, these are really -1-1,11,-11). Note that the two source symbols in the convolutional coder uniquely determine which pair of symbols is transmitted.

Now assume the channel has an impulse response of $h_0 = 1$, $h_1 = .25$, causing interference between adjacent symbols. Then the possible <u>received</u> symbols $\underline{R}_k$ (see model of Fig. 3.1) appear in the tree of Fig. 6.2.3. The upshot of the intersymbol interference is an effective constraint length of three source-symbols. Each received vector $\underline{R}_k = \left\{\pm h_0 \pm h_1, \pm h_0 \pm h_1\right\}$ depends on the two source-symbols in the convolutional coder plus the symbol most recently shifted out. There are $2^{\ell} = 8$ such $\underline{R}_k$'s, and these are assumed known by the receiver.

Decisions on each B_k are made after a delay $d = \ell - 1 = 2$ to ensure that the effects of B_k have died away. The decision on B_2 (in the second column of Fig. 6.2.3) is delayed until the first information on B_4 is received, and made as follows:

Calculate the eight "incremental" probabilities

$$\underset{j=1,\dots,8}{\Delta^{(j)}_{k+d}} = P(B_{k+d})\,p(\underline{X}_{k+d} \mid B_k \dots B_{k+d})$$
$$= P(B_k)\,p(\underline{X}_4 \mid B_2 B_3 B_4)$$
$$= P(B_4)\prod_{i=1}^{2} f(X_{4,i} - R_{4,i})$$

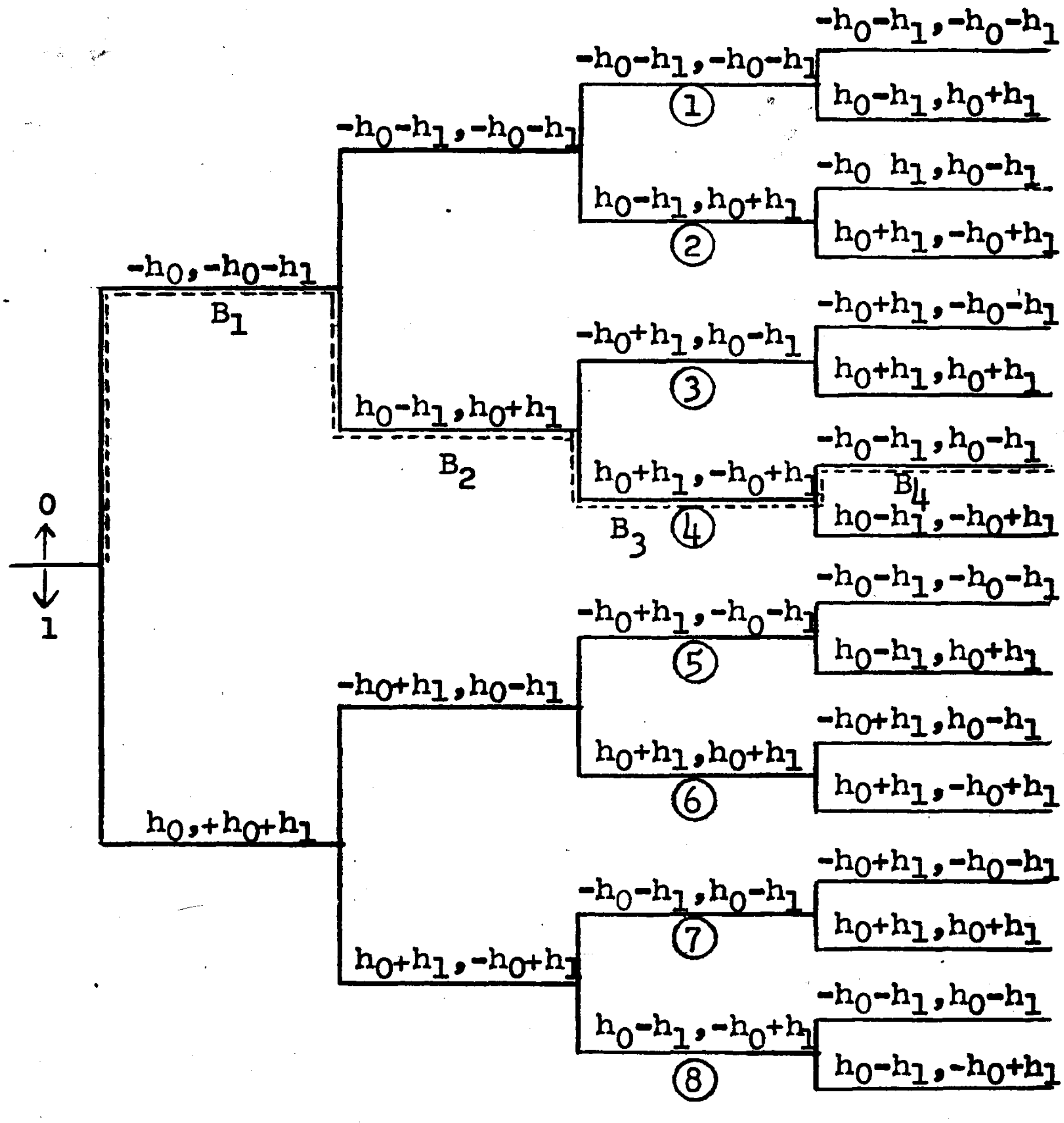


Fig. 6.2.3. Possible received vectors $\underline{R}_k$ for the code of Fig. 6.2.2 and a length-two impulse response.

$$= P(B_4)\prod_{i=1}^{2} f(N_{4,i}) \qquad (6.2.1)$$

where $\underline{X}_4 = (-h_0-h_1+N_{4,1}, h_0-h_1+N_{4,2})$.

then weight these by the "old" probabilities, or "OLDP's":

$$\underset{i=1,\dots,4}{\mathrm{OLDP}^{(i)}_{k+d}} = \sum_{B_{k-1}} p(B_{k-1}B_k\dots B_{k+d-1}, \underline{X}_1,\dots \underline{X}_{k+d-1})$$

$$= \sum_{B_1} p(B_1B_2B_3, \underline{X}_1\underline{X}_2\underline{X}_3) \qquad (6.2.2)$$

In this example, the four OLDP's are ①+⑤, ②+⑥, ③+⑦, and ④+⑧, representing the sums over B_1 of the eight statistics from the previous decision. Finally, we pick $\hat{B}_2 = 1$ if

$$\sum_{B_3}\sum_{B_4} P(B_4)p(\underline{X}_4|1\ B_3B_4)\sum_{B_1} p(B_1 1\ B_3, \underline{X}_1\underline{X}_2\underline{X}_3)$$

$$> \sum_{B_3}\sum_{B_4} P(B_4)p(\underline{X}_4|0\ B_3B_4)\sum_{B_1} p(B_1 0\ B_3, \underline{X}_1\underline{X}_2\underline{X}_3) \qquad (6.2.3)$$

An alternate expression would be to choose $\hat{B}_2 = 1$ if

$$\sum_{j+5}^{8} \triangle_4^{(j)}\mathrm{OLDP}_4^{(i)} > \sum_{j+1}^{4} \triangle_4^{(j)}\mathrm{OLDP}_4^{(i)},$$ where $i = \frac{j}{2}$ or $\frac{j+1}{2}$ whichever is even.

Again looking at the tree of Fig. 6.2.3, we see that the upper four paths in the rightmost column represent paths for which $B_2 = 0$. The next four paths are from $B_2 = 1$. Had we let $d = 3$, then all 16 paths would have been retained, but with no gain in information because the top half of the tree is identical to the lower half.

6.3. Sub-optimum Receiver by Threshold Techniques.

Clearly, to reduce the complexity of the optimum joint sequential receiver, we must calculate only a subset each time of the incremental probabilities $\Delta^{(j)}_{k+d}$, $j = 1,\ldots,2^{\ell}$. Each of these probabilities can be thought of as a branch on a tree (Fig 6.2.3), weighted by terms from earlier branches. A logical criterion for deciding which paths to retain, therefore, would be some quality possessed by the weights.

If most of the energy due to the source symbol B_k has been received prior to receipt of $\underline{X}_{k+d}$, then it is reasonable to expect that much of the information for the decision on B_k is contained in the weighting terms

$$\mathrm{OLDP}^{(i)}_{k+d} = \sum_{B_{k-1}} p(B_{k-1}, B_k \ldots B_{k+d-1}, \underline{X}_1 \ldots \underline{X}_{k+d}) \qquad i = 1,2,\ldots,2^{\ell-1} ,$$

summarizing the history of the received sequence. Many of these terms, the "old" probabilities, are very small compared to the ones which are "closest" to the true sequence. That is,

$$\sum_{i=1}^{2^{\ell-1}} \mathrm{OLDP}^{(i)}_{k+d} = 1 \qquad (6.3.1)$$

for the optimum receive , and if we discard all those OLDP's satisfying $\mathrm{OLDP}^{(i)}_{k+d} <$ THRESHOLD, then

$$\sum_{i=1}^{2^{\ell-1}} \mathrm{OLDP}^{(i)}_{k+d} = 1-\epsilon . \qquad (6.3.2)$$

The smaller ϵ is, the more closely the sub-optimum approximates the optimum receiver. But the larger ϵ (and the larger THRESHOLD), the less the required calculations by the receiver. In practice, all OLDP's are normalized with respect to the largest OLDP. Every time an OLDP falls below the threshold, it is not necessary to calculate the two incremental probabilities associated with it, and in this manner the receiver size is reduced.

Fig. 6.3.1 shows the effect of arbitrarily picking a fixed threshold to trim marginal paths from the received-symbol tree. The two convolutional codes used are each constraint length two and code rate two, and the channel is similar to the wireline channel used in [1]. Whenever the noise gets large (the noise samples are shown in Fig. 6.3.2), the receiver responds by retaining more paths. Likewise, few paths are retained when the additive noise is relatively quiet. Fig. 6.3.3 is the probability of error (P(E)) for these two codes as a function of the signal-to-noise ratio, with THRESHOLD as a parameter, and Fig. 6.3.4 is the probability of error as a function of the threshold.

These two codes, though very simple, point out several interesting facts. First, P(E) is affected hardly at all by eliminating the lowest probability

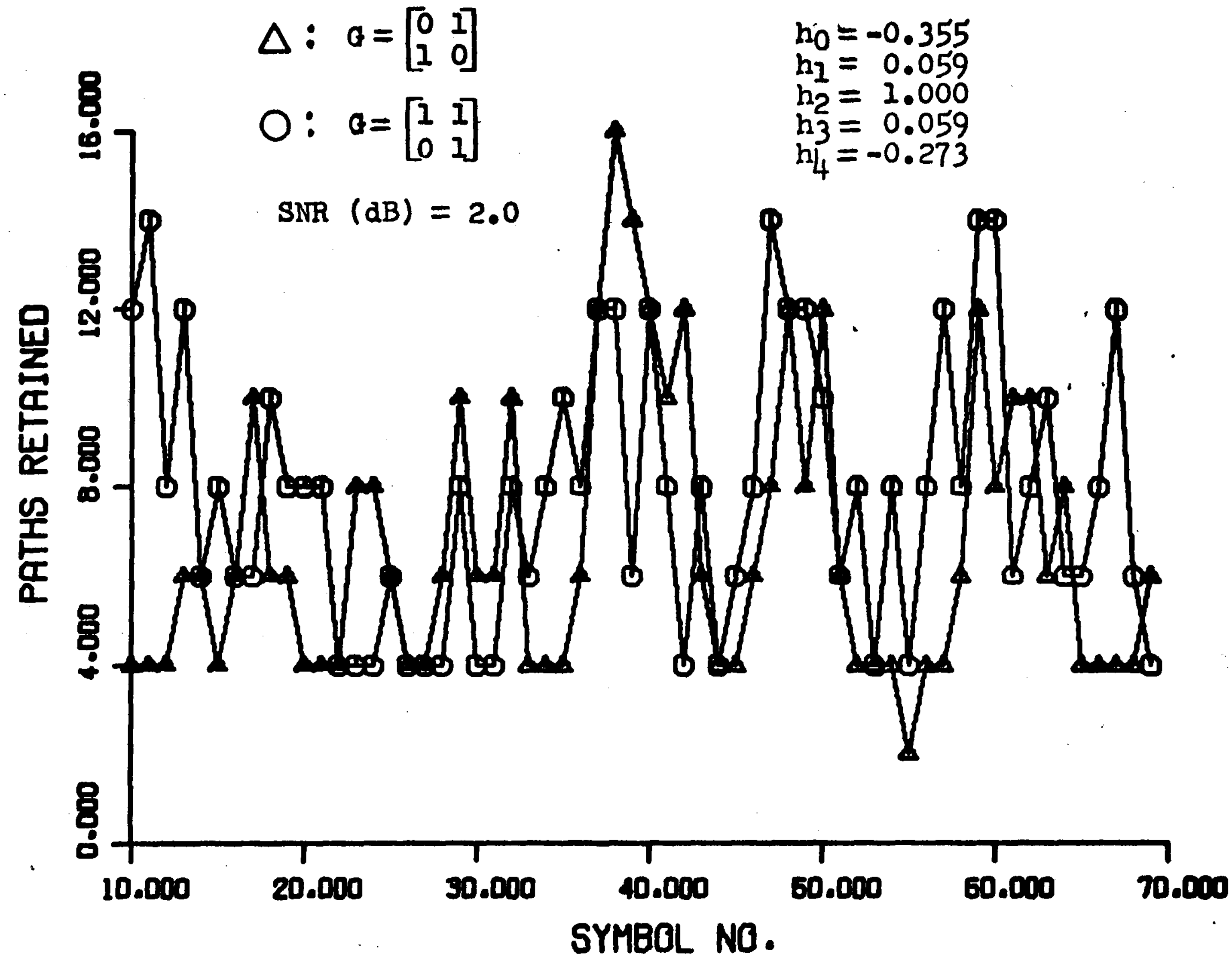


Fig. 6.3.1. Paths retained by two different length-two codes and a threshold of .01.

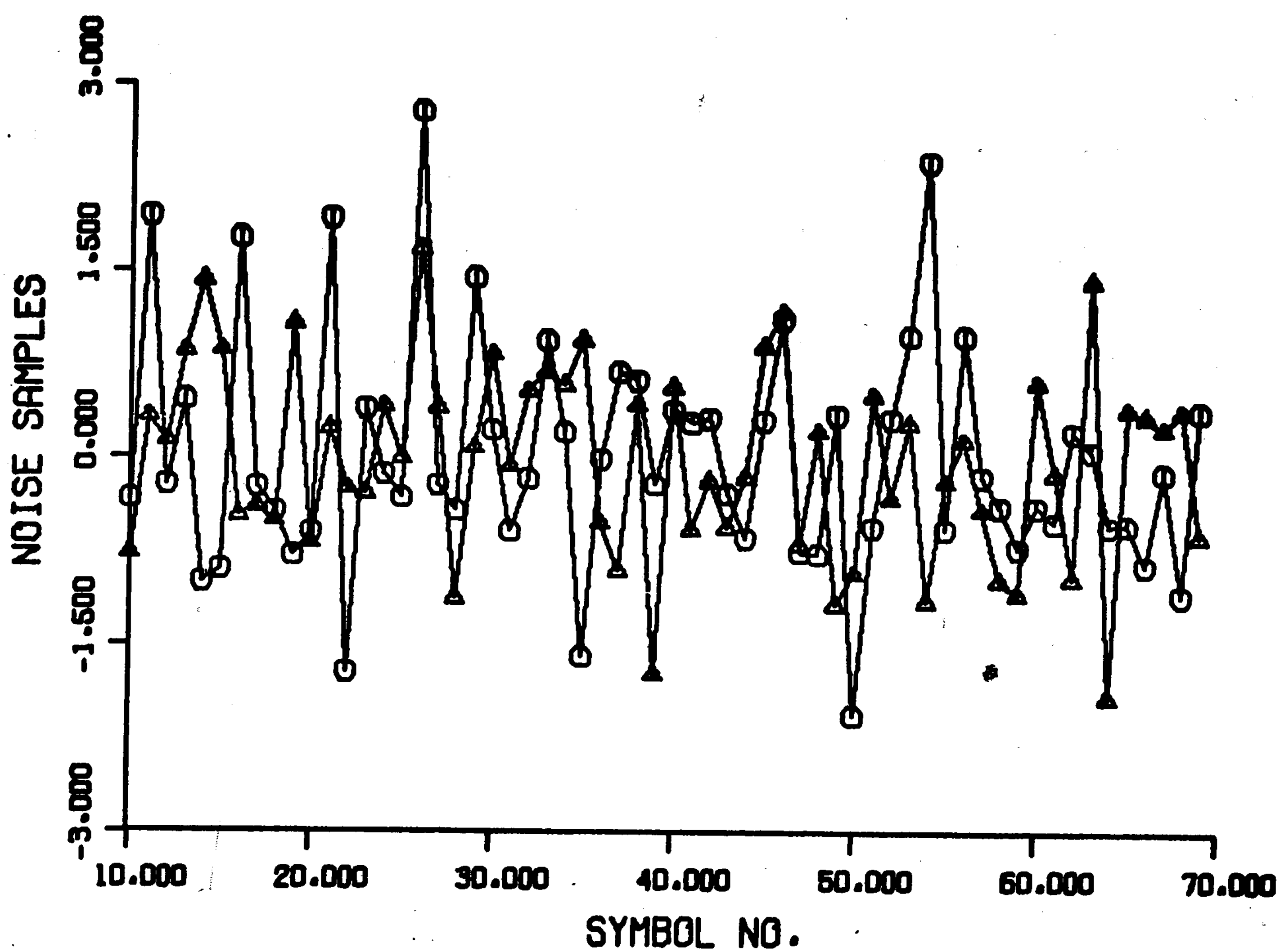


Fig. 6.3.2. Two-dimensional noise samples affecting the graphs of Figs. 6.3.1, .3, and .4.

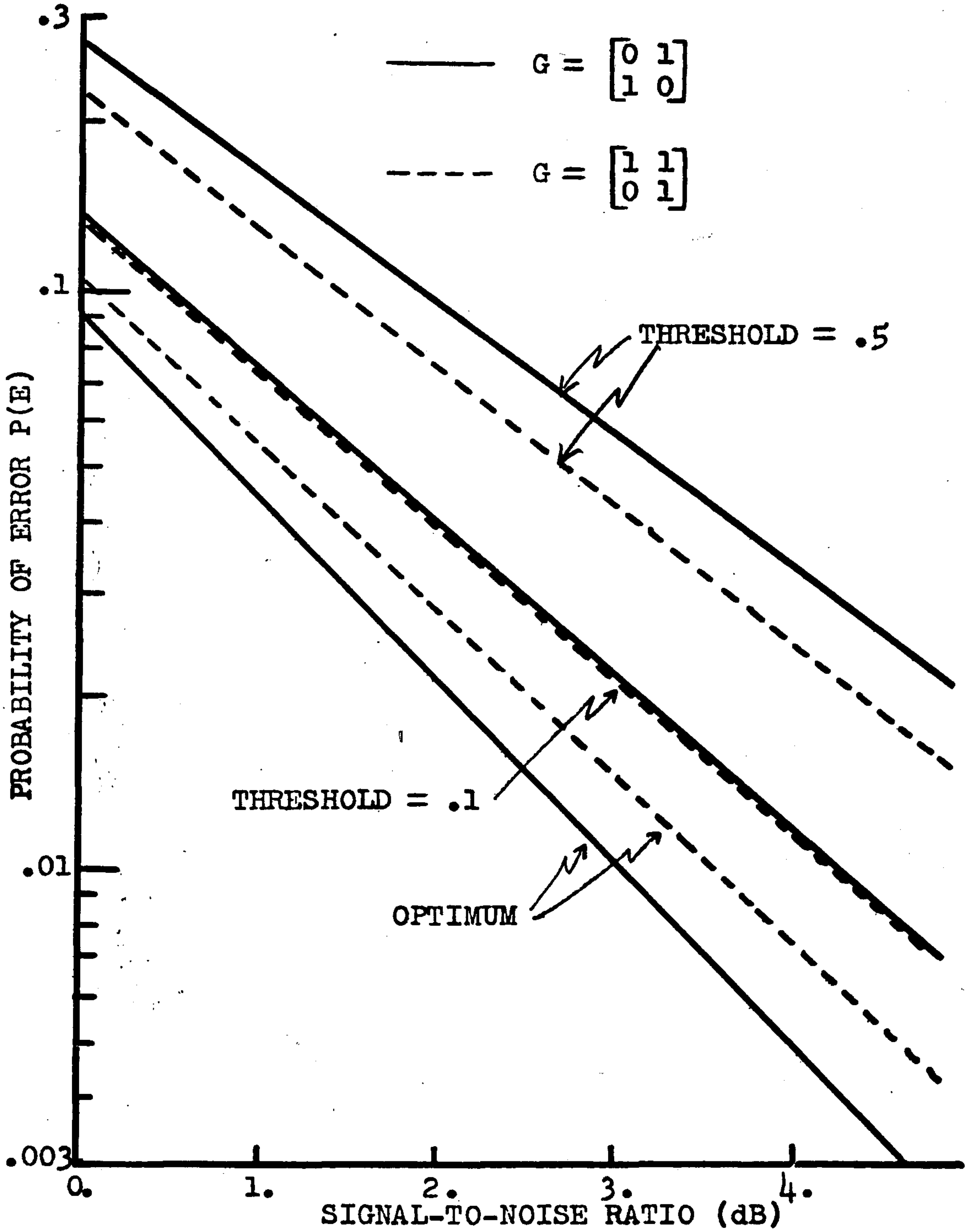


Fig. 6.3.3. Performance of two length-two codes.

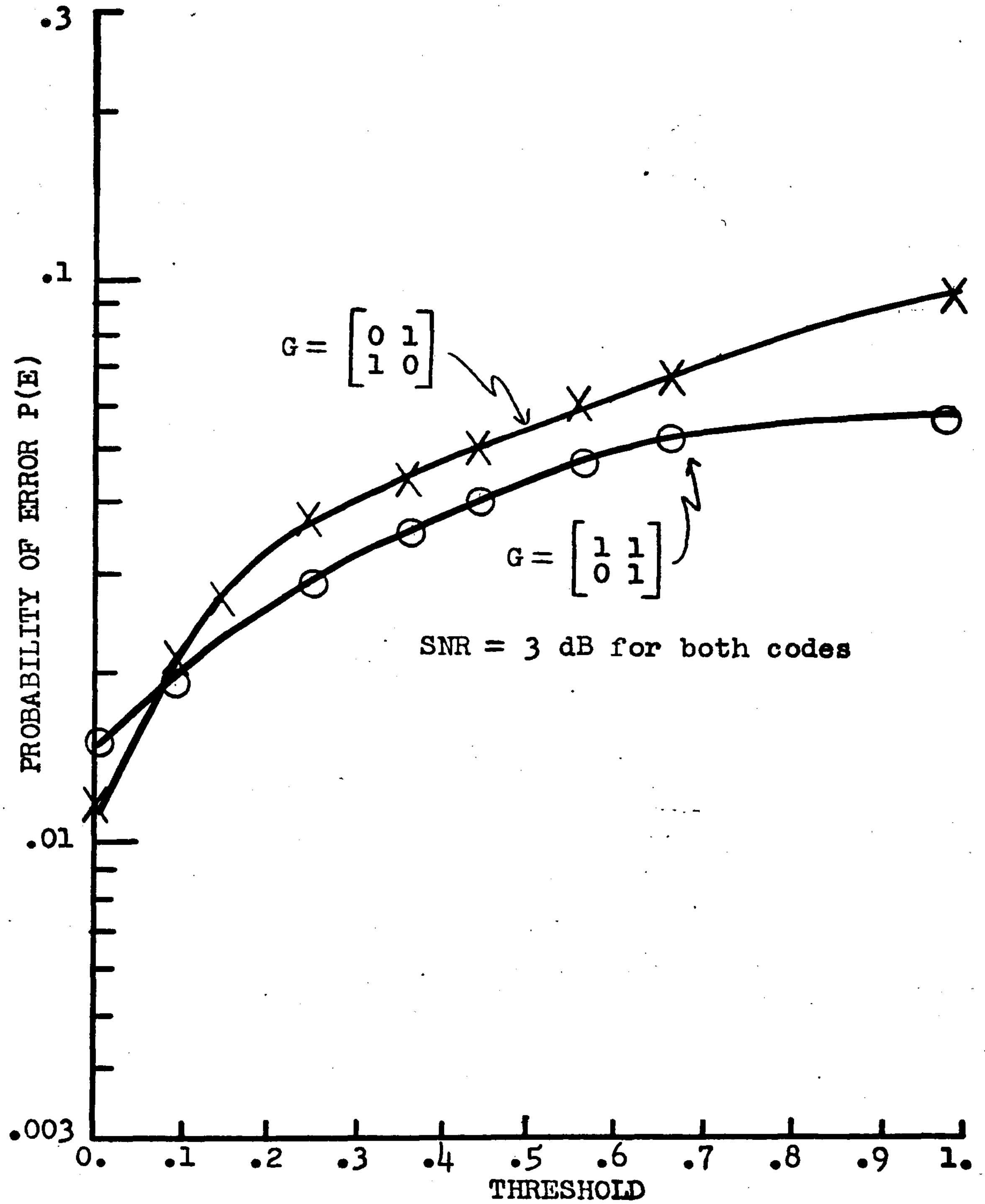


Fig. 6.3.4. Similar codes perform differently.

paths . Second, even though most paths are rejected by setting THRESHOLD high, P(E) does not blow up to 1/2. Indeed, for a very high threshold (say, .999 for the normalized OLDP's), the algorithm becomes "decision-directed," allowing only two paths to be considered following retention of only one OLDP from the previous decision. One might believe that a decision-directed process like this would continue to make errors after a burst of noise causes a deviation from the correct path. That the threshold algorithm always (as far as we can tell) returns to the correct path, without a long string of errors, is a remarkable fact. Last, we observe that although one code may out-perform another in the optimum case, it may be worse for a given threshold.

In order to more reliably predict the effects of the THRESHOLD algorithm, simulation on a more complicated code was performed. Fig. 6.3.5 shows P(E) for several thresholds and the code and channel used in [2]. As a result of the small number of errors and hence the need for excessive computer time, simulation was not done for signal-to-noise ratios above 5dB. But the pattern is clear: only a small subset of the paths used by the optimum algorithm can out-perform the separately-optimized detector-decoder. Fig. 6.3.5 is better understood with the aid of Table 6.3.1, which

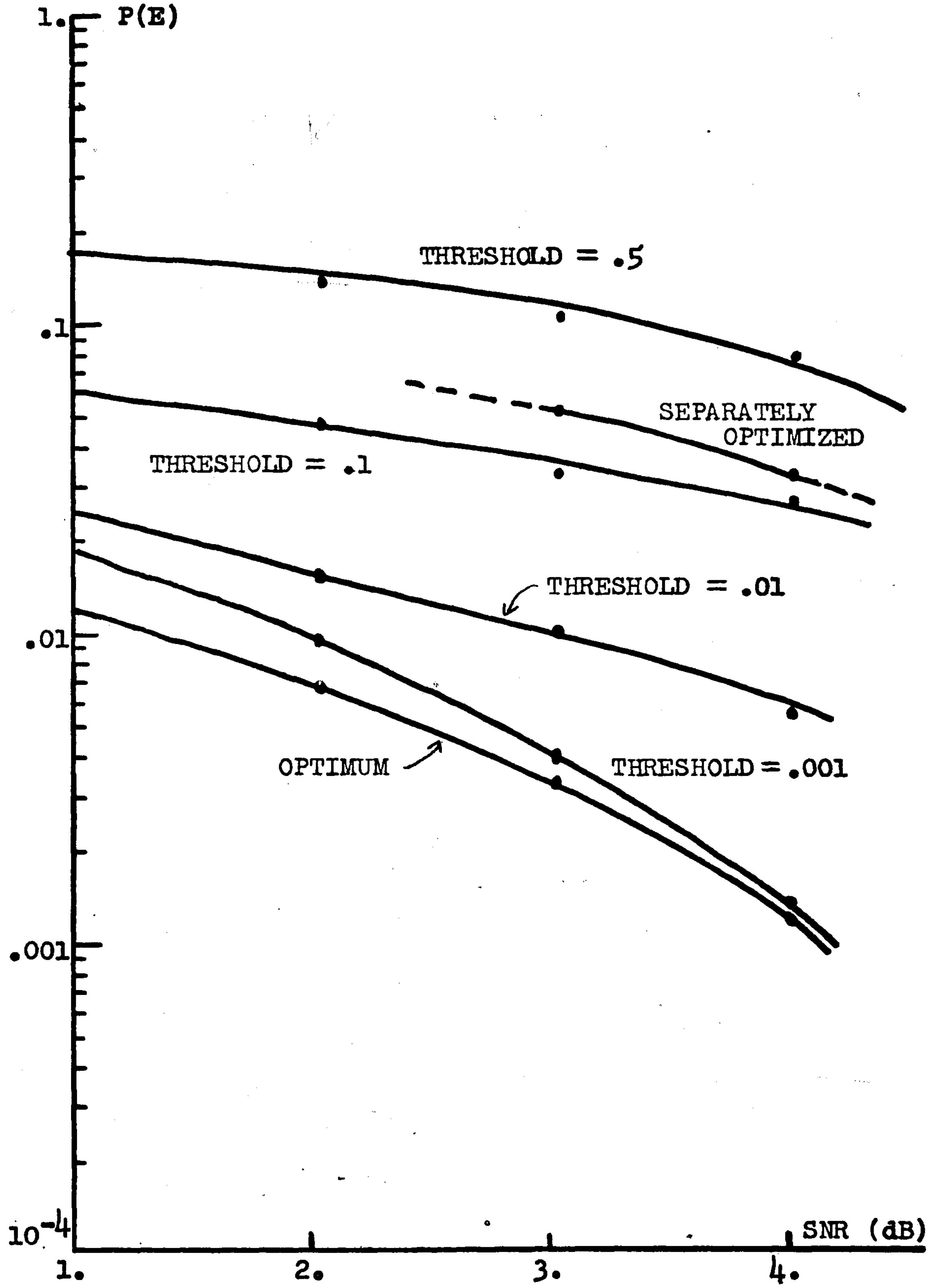


Fig. 6.3.5. P(E) vs. SNR for THRESHOLD algorithm.

THRESH-OLD	SIGNAL-TO-NOISE RATIO 1.		2.		3.		4.	
	AVE	DEV	AVE	DEV	AVE	DEV	AVE	DEV
0.5	2.3	.79	2.3	.70	2.2	.60	2.2	.50
0.1	3.5	1.7	3.1	1.4	2.8	1.1	2.5	1.0
.01	6.2	4.0	4.8	2.8	3.8	1.8	3.3	1.4
.001	10.	7.0	7.1	4.7	5.2	3.0	4.1	2.0

Table 6.3.1. Few paths retained for high thresholds.

lists the average number of paths retained (out of 64) and the associated standard deviation for each point on the sub-optimum curves.

Fig. 6.3.6 illustrates how widely changing the number of paths retained by this code can be. As in Fig. 6.3.1, the number increases as the noise does, and drops during more quiet periods. The four curves have roughly the same shape, indicating that a noisy interval causes most of the marginal (smallest) OLDP's to increase in likelihood.

6.4. Sub-optimum Receiver by Noise Tolerance Criterion.

The vectors $\underline{X}_k$ can be thought of as points in n-space (if the code rate is 1/n), and the noise $\underline{N}_k$ as a distance vector from the true point $\underline{R}_k$ in that space:

$$\underline{X}_k = \underline{R}_k + \underline{N}_k ,$$

$$\underline{N}_k = \underline{X}_k - \underline{R}_k . \qquad (6.4.1)$$

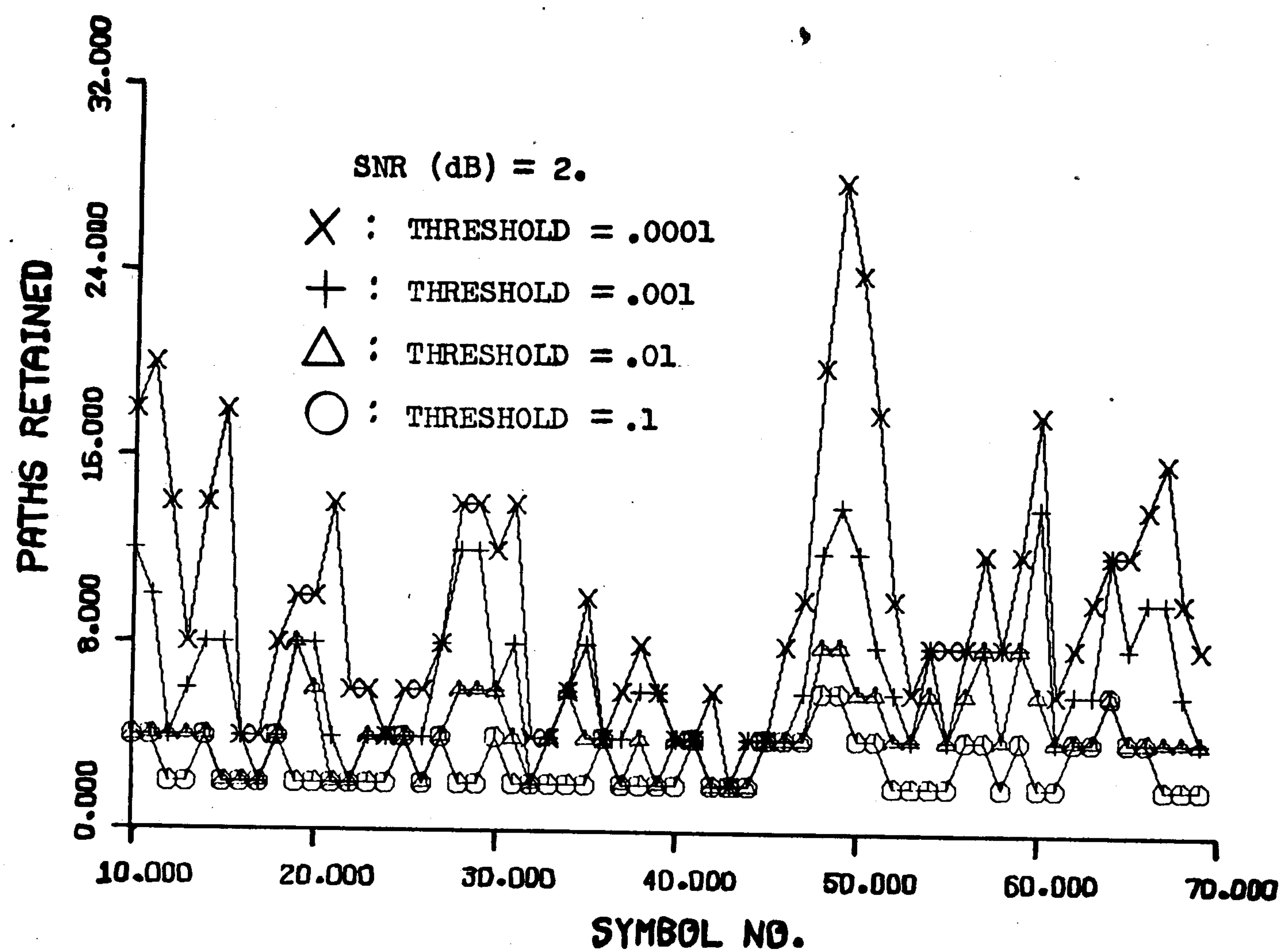


Fig. 6.3.6. Paths retained depend on noise and threshold.

This suggests another method for limiting the optimum receiver complexity. Calculate only those incremental probabilities Δ_k falling inside an n-sphere of radius $C\sigma$ from $\underline{R}_k$ where σ is the standard deviation of the noise. The effect is the same as the THRESHOLD algorithm, but not nearly as stable. The number of paths retained is allowed to vary, depending mostly on the noise, but also on the location of the points $\underline{R}_k$ in n-space. Certain codes result in better separation of the $\underline{R}_k$'s, and it is possible for the intersymbol interference to improve separation even more.

Fig. 6.4.1 shows curves of P(E) for various tolerances $C\sigma$, compared with the optimum results for the code and channel in [2]. As was the case for the THRESHOLD algorithm, a select subset of paths yields nearly optimum performance. Only 39.2 out of 64 paths were retained on the average for TOLERANCE = 5 (and SNR(dB) = 3.0), yet the simulated error rate was the same as the optimum P(E) (noting, of course, that only a finite number of symbols can be economically simulated, hence small differences in P(E) are obscured).

Unlike the THRESHOLD algorithm, the TOLERANCE algorithm falls apart when the tolerance is set to exclude too many paths. The culprit causing this problem is the low energy of h_0 and h_1, compared to

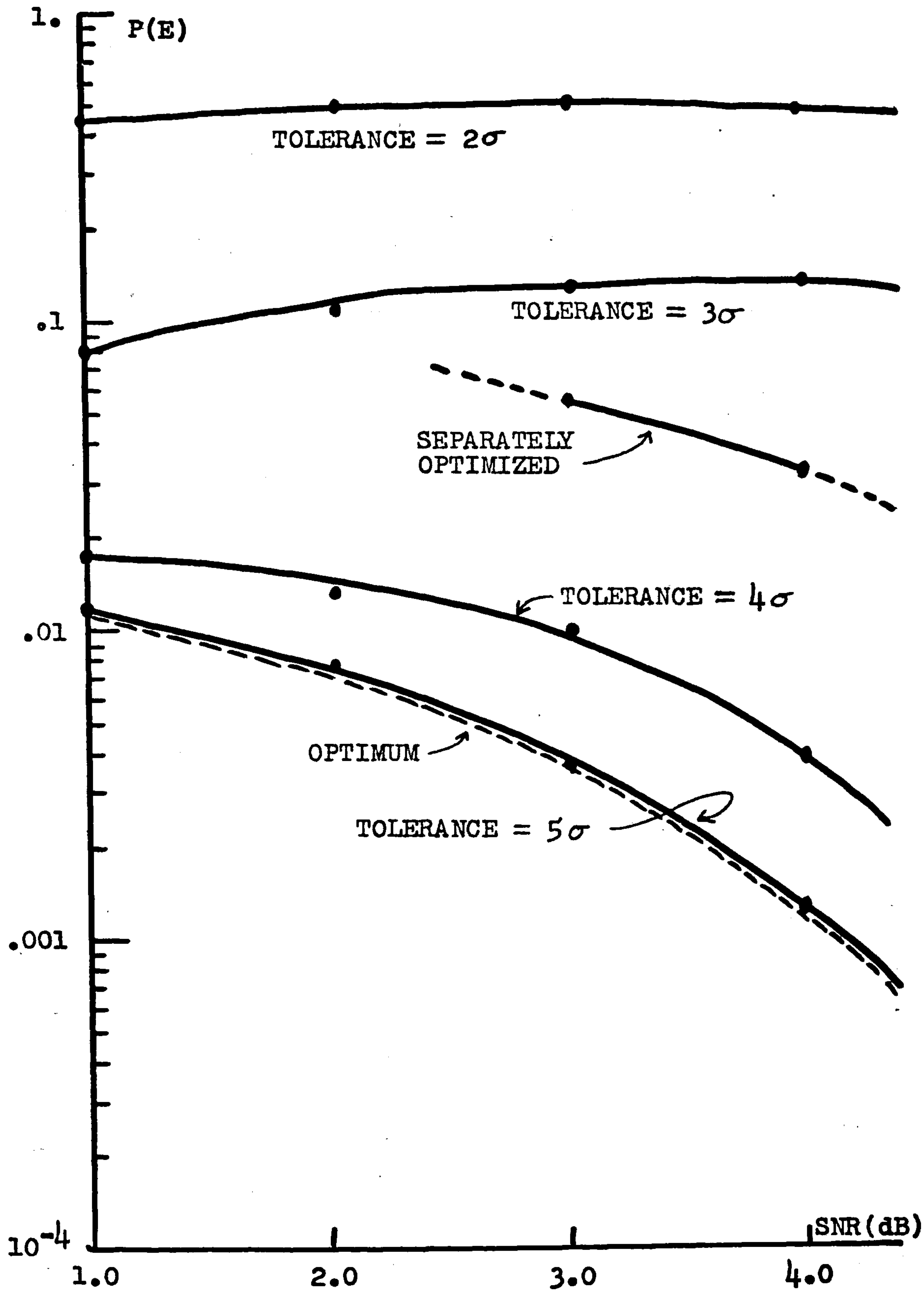


Fig. 6.4.1. P(E) vs. SNR for TOLERANCE algorithm.

h_2, the main pulse of the channel response used in the simulations. The low energy tail of h(t) places several of the possible $\underline{R}_k$'s close together, and when a noise sample brings the received value $\underline{X}_k$ too close to the wrong $\underline{R}_k$ and the tolerance is small, only the one wrong path is retained. Errors seem to propagate using the TOLERANCE algorithm, thus there would be a sharp knee in a graph of P(E) vs. $C\sigma$, where the algorithm suddenly begins to work well.

Overall, the TOLERANCE algorithm is less reliable and predictable than the THRESHOLD algorithm. There is a third algorithm, however, which is more promising than either TOLERANCE or THRESHOLD, because it limits the potential size of the receiver. This is the RANKING algorithm.

6.5. Sub-optimum Receiver by Ranking.

The RANKING algorithm is based on the same logic as the THRESHOLD algorithm -- limit the number of paths kept in the received symbol tree; only the approach is a little more involved. Whereas a simple comparison was all that was needed for each OLDP in THRESHOLD, RANKING requires each new set of OLDP's to be ranked by value, choosing a fixed number, N_R, to keep each time. Because N_R is fixed, there is no need for the "spare" room that THRESHOLD and TOLERANCE retain for expansion during noisy sequences.

The advantage of a fixed-size receiver outweighs the disadvantage of the additional calculations needed to rank the OLDP's (as detailed in the next chapter). It also outweighs the simulation results, showing that the RANKING algorithm does worse for a given N_R than the THRESHOLD receiver and an equivalent average path retention. Fig. 6.5.1, for example, indicates that 6.2 paths (THRESHOLD = .01) has P(E) = .024, while $N_R = 8$ (RANKING) has P(E) = .026. This result can be expected, because the THRESHOLD algorithm is allowed to "open up," or expand, when it needs to.

Fig. 6.5.2 more vividly demonstrates how only a small set of paths need be retained to achieve a nearly optimal error rate. Out of 64 possible paths, going from two to four yields the most substantial improvement. After about ten paths are retained, no further improvement is noticed. Changing the signal-to-noise ratio changes the vertical position, but not the shape, of the curves P(E) vs. paths retained.

A more detailed explanation of the method of simulating RANKING, as well as the THRESHOLD, TOLERANCE, and optimum algorithms appears in Appendix A. But the next chapter tries to sort out the complexity of the simulations to see if anything was really gained, and speculates on the complexity of a hardware realization.

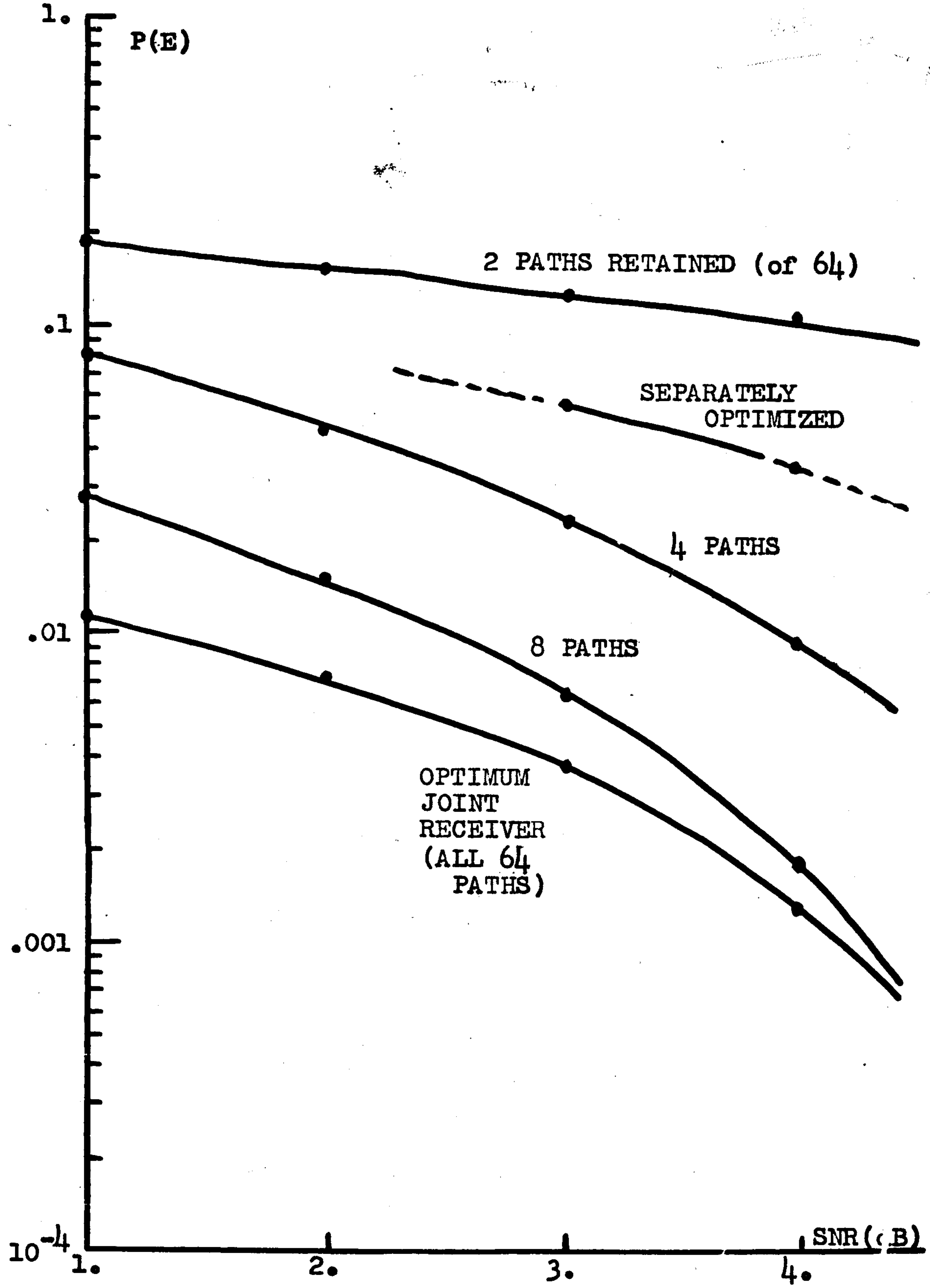


Fig. 6.5.1. P(E) vs. SNR for the RANKING algorithm.

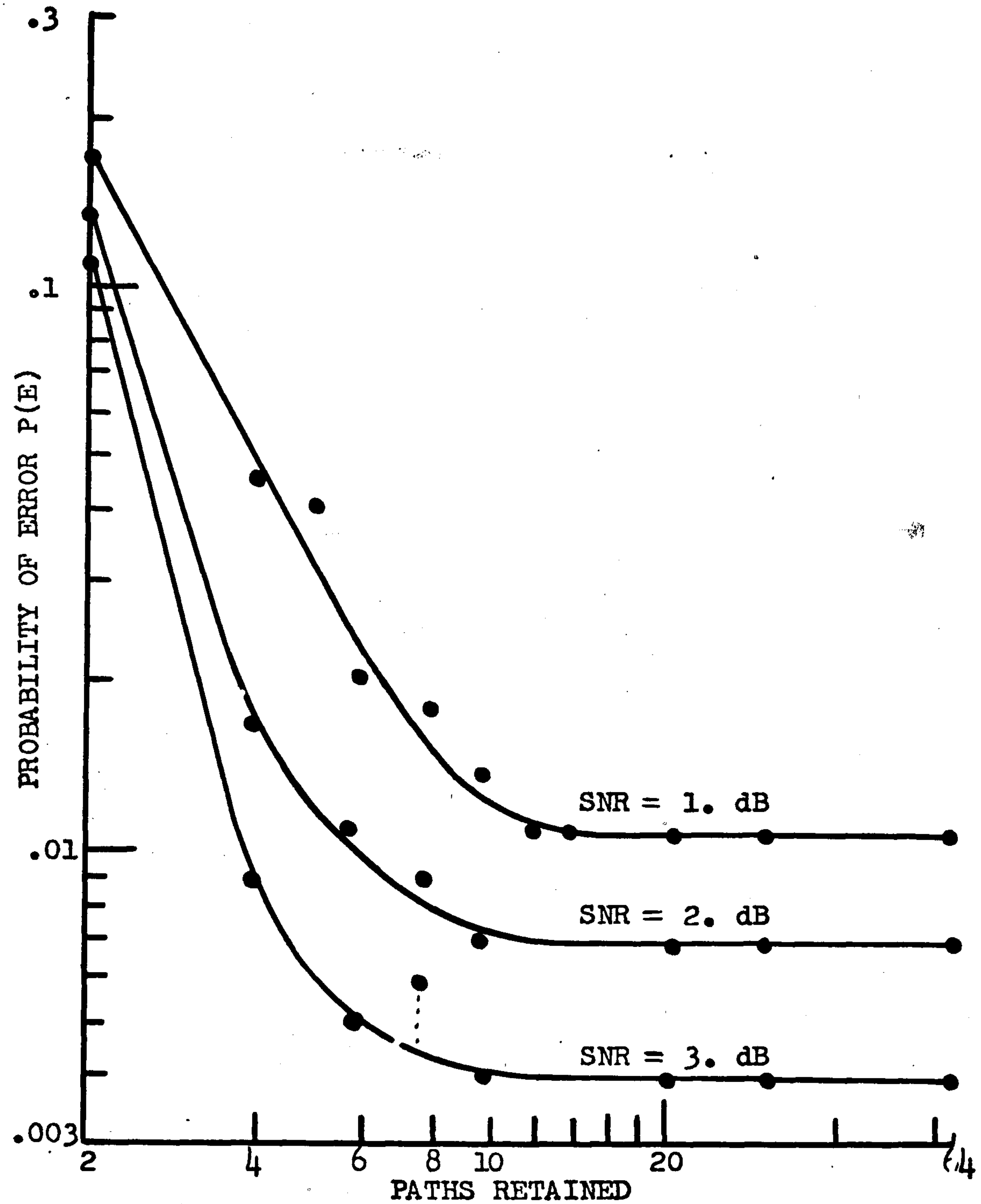


Fig. 6.5.2. Few paths yield near-optimal results.

7. COMPLEXITY AND REALIZATION OF THE SUB-OPTIMUM ALGORITHMS.

The simulation results of Chapter 6 indicate that by using only a small subset of the possible paths as a basis for an ML decision on the source-symbols, an error rate is achieved below the rate of the separately-optimized detector-decoder. This conclusion, however, is only useful if the sub-optimum joint receiver can be implemented for less cost than the optimum case.

One reasonable criterion for judging a software approach to realizing the sub-optimum receiver is the amount of CP time consumed by processing one symbol. Fig. 7.1 represents the CP time/symbol for the code and channel used extensively for error rate comparisons in Chapter 6.

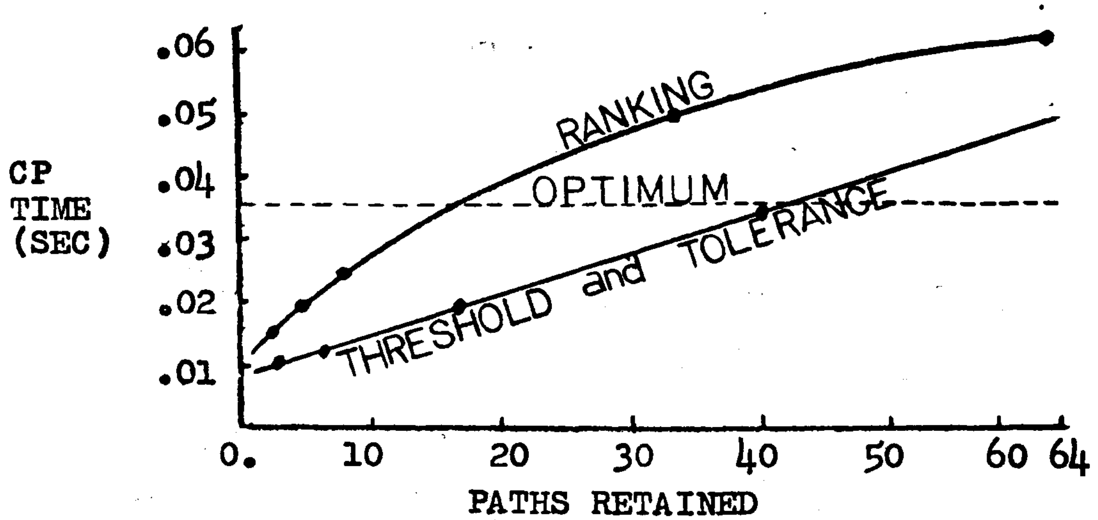


Fig. 7.1. CP time in FORTRAN simulations.

The THRESHOLD and TOLERANCE algorithms linearly consume less CP time for each path dropped, since dropping one path is equal to skipping that part of the code which computes the associated incremental probability. While the data for Fig. 7.1 comes from the FORTRAN simulation outlined in Appendix A, the general shape and relative position of each curve is probably similar to a dedicated software approach which pays more attention to code optimization.

On the basis of time consumed, the RANKING algorithm performs least satisfactorily. The reason for this is due to the particular manner that the incremental probabilties were ranked. If two paths were required, all 32 OLDP's were interchange-sorted, requiring 31 comparison of mostly zero data. Similarly, for 62 paths, $31+30+29+ \ldots +1 = 465$ comparisons must be made for each symbol. By ranking only non-zero data, the sorting algorithm is simplified, but this advantage is lost in additional memory references needed to keep track of which incremental probability is associated with which "old" probability.

To get a rough idea of the computations saved by trimming the potential paths, consider that the CDC 6400 can do a floating point multiply in $5.7 \mu s$, and an integer addition in 600 ns. That means that a subset of less than ten paths out of 64 saving .02 CP seconds/symbol off the optimum algorithm saves 3500 multiplies, or 33,000 additions, or a combination thereof.

Ideally, a sub-optimum algorithm could be incorporated into a piece of hardware, such as a MODEM for voice-grade channels. For this application, the RANKING algorithm is the only practical one because it requires a fixed size receiver. The THRESHOLD saves little or nothing in hardware since it can, in theory, expand to the size of the optimum receiver when all OLDP's exceed the threshold. The RANKING algorithm hardware could be serial, with minimum hardware and minimum speed, or it could have a register and arithmetic unit for each path, a "pipeline" effect with maximum speed. Only the ranking itself would require serial processing. The various possible R_k's could be maintained in a ROM and looked up as in the FORTRAN simulation.

Thus we have progressed from the sequential detector algorithm through the addition of a separate convolutional encoder to the joint detector-decoder. For a single symbol, the matched filter receiver provides a lower bound on the error rate P(E). But for long strings, the optimum joint sequential receiver outperforms the matched filter/transversal equalizer, which cannot be practically optimized. The complexity of the sequential receiver, however, invites the study of a simplified sub-optimum form, hence the simulation results presented herein. Indications are that a sub-optimum algorithm like THRESHOLD or RANKING is especially attractive for long codes, or severe symbol overlap, because good performance is obtained even with small path subsets.

Further study of this receiver structure should include a search for an algorithmic estimate of P(E), and finding out why the THRESHOLD and RANKING algorithms return to the correct path following an error. An ambitious project would be the construction of a hardware realization of the RANKING algorithm.

REFERENCES

1. K. Abend and B. D. Fritchman, "Statistical Detection for Communication Channels with Intersymbol Interference," Proc. IEEE, Vol. 58, pp. 779-785, 1970.

2. M. A. Sattar, "Joint Detection and Decoding of Convolutional Codes for Channels with Intersymbol Interference," Master's Thesis, Lehigh University, 1974.

3. R. W. Lucky, J. Salz, and E. J. Weldon, Jr., Principles of Data Communication, New York, McGraw-Hill, 1968.

4. R. W. Lucky, "Automatic Equalization for Digital Communication," BSTJ, Vol. 44, p. 547, 1965.

5. F. K. Becker, L. N. Holzman, R. W. Lucky, and E. Port, "Automatic Equalization for Digital Communication," Proc. IEEE, Vol. 53, 1965.

6. R. W. Lucky, "Adaptive Equalization of Digital Communication Systems," BSTJ, Vol. 45, 1966.

7. M. R. Aaron and D. W. Tufts, "Intersymbol Interference and Error Probability," IEEE Trans. on Inform. Theory, Vol. IT-12, p. 26, 1966.

8. C. D. Forney, Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference," IEEE Trans. on Inform. Theory, Vol. IT-18, 1972.

9. J. G. Proakis, "Advances in Equalization for Intersymbol Interference," Advances in Communication Systems, Vol. 4, A. J. Viterbi, editor, Academic Press, New York, 1975.

10. R. W. Chang and J. C. Hancock, "On Receiver Structures for Channels Having Memory," IEEE Trans on Inform. Theory, Vol. IT-12, 1966.

APPENDIX A

COMPUTER SIMULATION OF THE OPTIMUM AND SUB-OPTIMUM RECEIVER ALGORITHMS

A computer simulation of the optimum and sub-optimum algorithms described in Chaps. 5-6 was performed on a Control Data 6400 computer, and the programs were written in the FORTRAN IV language. The 6400 can do a floating point multiply in 5.7μs and an integer addition in 600ns, but when one considers that parts of the decision segment of the optimum program may be evaluated thousands of times, it is clear why long codes were not tested nor were high SNR's used. Every attempt to optimize oft-used code was made, hence subroutine calls were mostly eliminated and several FORTRAN conventions were adapted to fit special needs.

The optimum receiver algorithm follows the logic of the flow-charts in Fig. A1-A11. The code rate is 1/N, the code constraint length is L. Other important variables are described in Table A1.

Rather than computing the code symbols $\underline{T}_k$ as each B_k is shifted into the coder, prior to "transmission," and then calculating the intersymbol interference due to previous $\underline{T}_k$'s, we note that each sequence $B_{k-\ell+1}$

$\ldots B_k$ can be used directly to find $\underline{R}_k$. First, a code table is constructed (flow-chart of Fig. A4) in which the 2^{ν} possible shift-register combinations map into a set of coded symbols $\underline{T}_k$, whose cardinality is less than or equal to 2^n. Second, the 2^L possible channel symbols $R_{k,i}$ (the HK's in Fig. A5) are found as $-h_0 -h_1 \ldots -h_{L-1}, \ldots, +h_0 + h_1 + \ldots + h_{L-1}$. Last, by using this information, the intermediate step of finding the $\underline{T}_k$'s is eliminated (Fig. A6), reducing the simulation of the coder and the channel to a table look-up for each sequence $B_{k-\ell+1} \ldots B_k$.

Using the example of section 6.2, a source-symbol sequence $B_{k-2}, B_{k-1}, B_k = 0,1,1$ generates $\underline{T}_{k-2}$, $\underline{T}_{k-1}, \underline{T}_k = 00,11,01$. From this we find $\underline{R}_k = (-1+.25, 1-.25) = (-.75,+.75)$. But the sequence 0,1,1 is an effective-length sequence, and will always yield the same $\underline{R}_k$, so we write

$$\underline{R}_k(0,1,1) = \underline{R}_k(4) = (-.75,+.75), \qquad \text{(A1)}$$

using the fact that 0,1,1 looks like the binary form of three, and noting that one must be added to correct for the lack of zero subcripting in FORTRAN.

Whenever modulo-n and logical AND functions appear, they are used to obtain special bits within a data word. For example, MOD(7,4) yields the rightmost bits 1,1 out of the sequence 1,1,1. Integer multiplies and divides are used as left and right shifts. 7/4 corresponds to shifting 1,1,1 two places to the right, leaving 0,0,1. In this manner a long binary sequence can be stored in one word of memory. The variables NUSEQ, HSEQ, TKSEQ, BKSEQ, and IZ all represent symbol sequences, not integer numbers.

Random input symbols and white Gaussian noise are generated by the subroutines RANDU and GAUSS, respectively, which are part of the IBM Scientific Subroutine Package.

The rest of the program is the straightforward application of the recursive rule given by (5.1) and (5.2). For each new input symbol B_k, an output vector $\underline{X}_k$ is calculated, and the 2^ℓ terms of (6.2.1) are found from

$$\underset{j=1,\ldots,2^\ell}{\Delta_k^{(j)}} = P(B_k)\prod_{i=1}^{n} f(N_{k,i})$$

$$= \frac{.5}{\sqrt{2\pi\sigma}}\exp\left(-\frac{\prod_{i=1}^{n}(X_{k,i}-R_{k,i})^2}{\sigma_N}\right) \qquad \text{(A2)}$$

for each possible $\underline{R}_k$. Each term is weighted by the correct "OLDP," and the terms are summed to obtain B_{k-d+1}. The weighted Δ_k's are then summed over $B_{k-\ell}$ to become the next OLDP's, and the cycle is repeated. Note that the OLDP'S must be normalized each time to compensate for rounding errors, and to allow common factors such as $.5/(\sqrt{2\pi}\sigma_N)$ to be dropped.

An explanation of modifications to the optimum program to simulate various sub-optimum cases follows the flow-charts of Fig.'s A1-A11.

N - Inverse of the code rate

NU - Code constraint length

L - Channel constraint length

H - Channel response samples

G - Code matrix

NCOUNT - No. of symbols simulated in each run

SNRDB - Signal-to-noise ratio (dB)

D - Delay (no. of intervals of T sec.)

LEF - Effective channel length

AM - noise mean

SUMH - Sum of channel samples squared

TK, NUSEQ, HSEQ, SYMSEQ, TKSEQ - Used as binary sequences for mapping input sequences into channel responses

HRK - Channel responses

VRNC - Noise variance

ERCNT - Error counter

RANDU - Random number generator, uniform distribution

GAUSS - Random number generator, normal distribution

BK - A generated symbol

BKK - Generated symbol sequence

XK - Channel response plus noise terms

NWPRB - New probabilities computed

OLDP - Old probabilities, formed from the NWPRB's

Table A1. Flow-chart nomenclature.

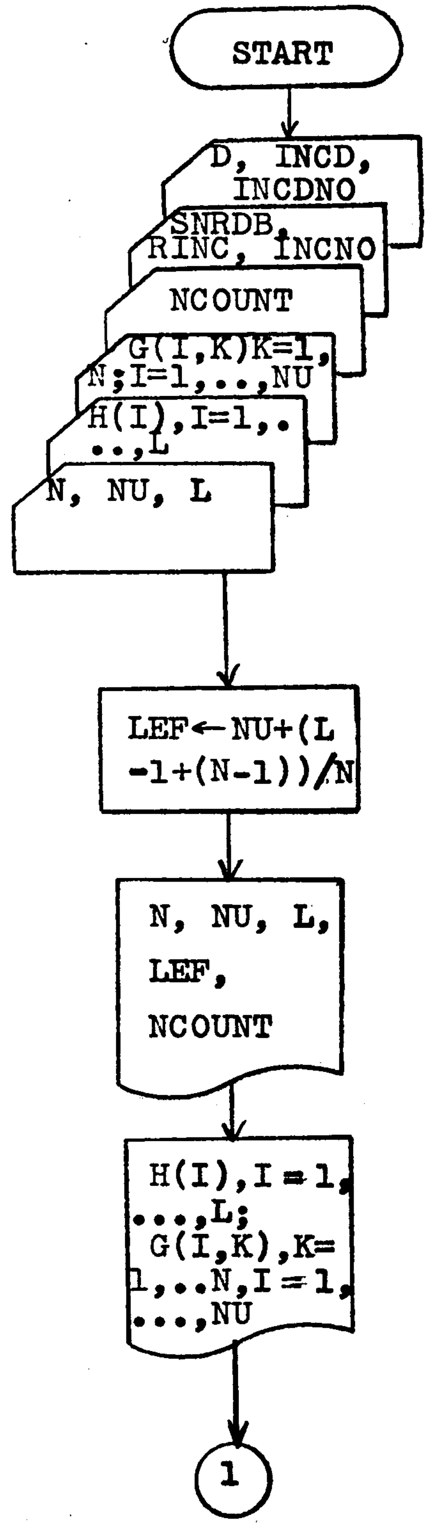


Fig. A1. Data Input / Output.

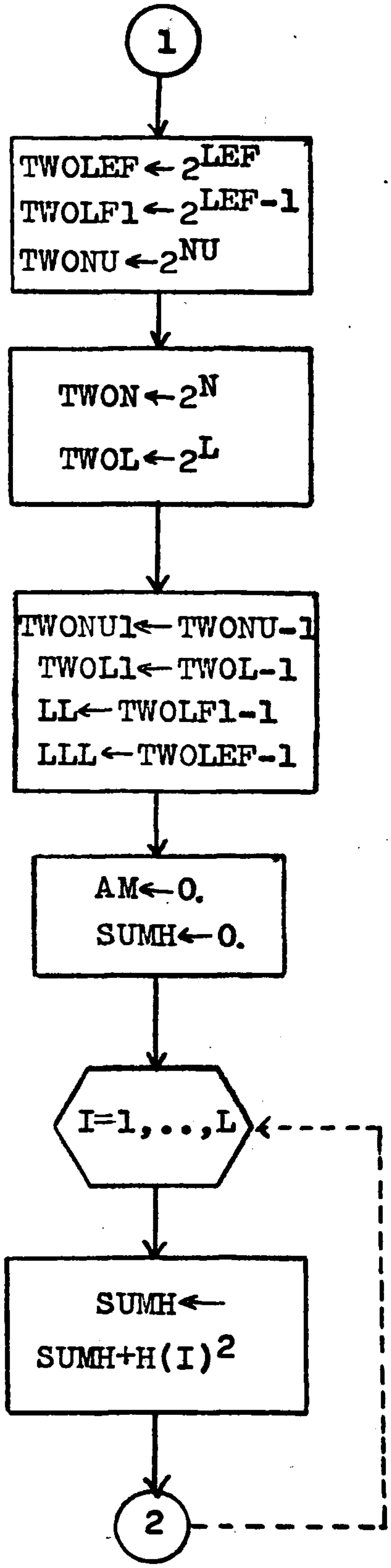


Fig. A2. Initialization.

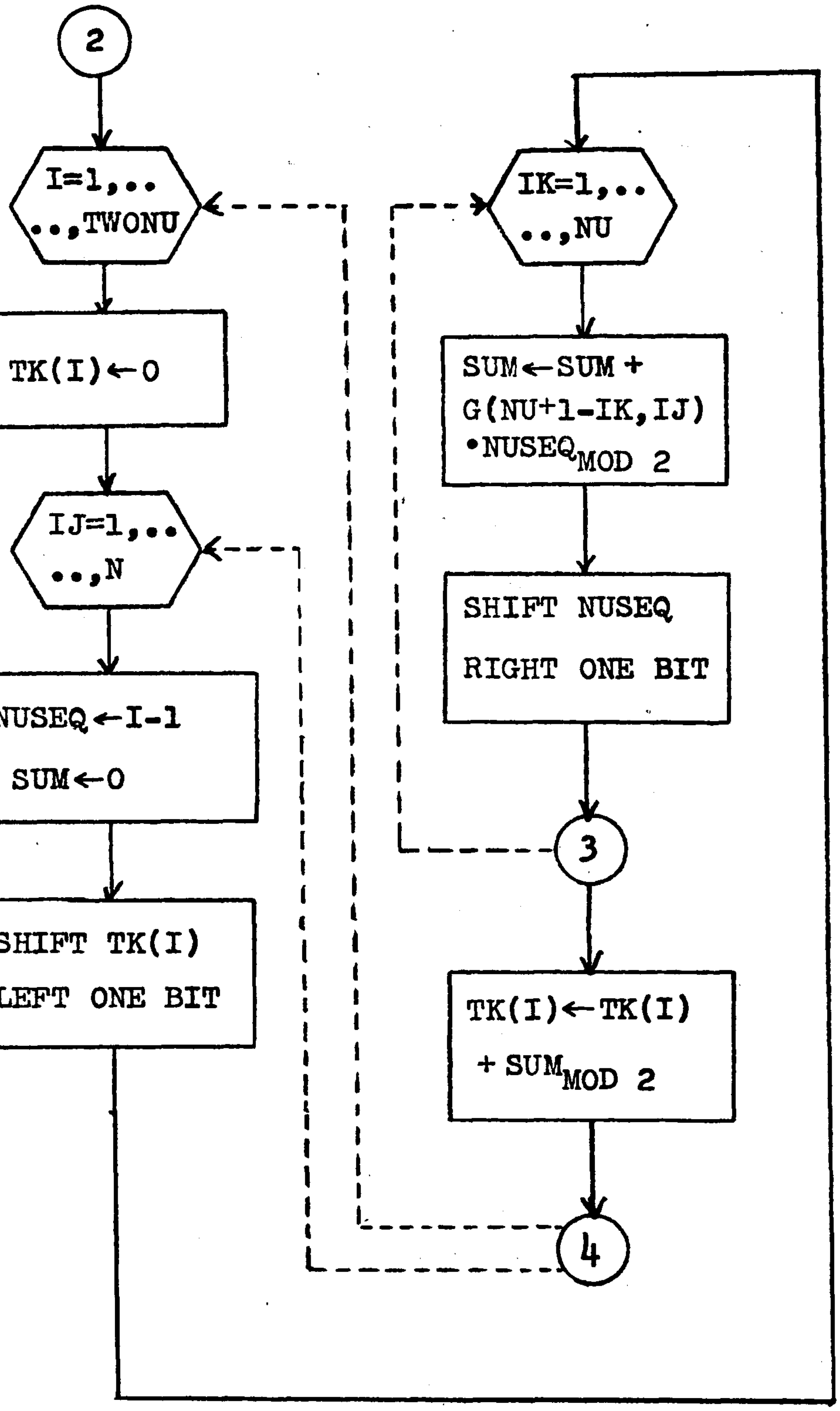


Fig. A3. Code table.

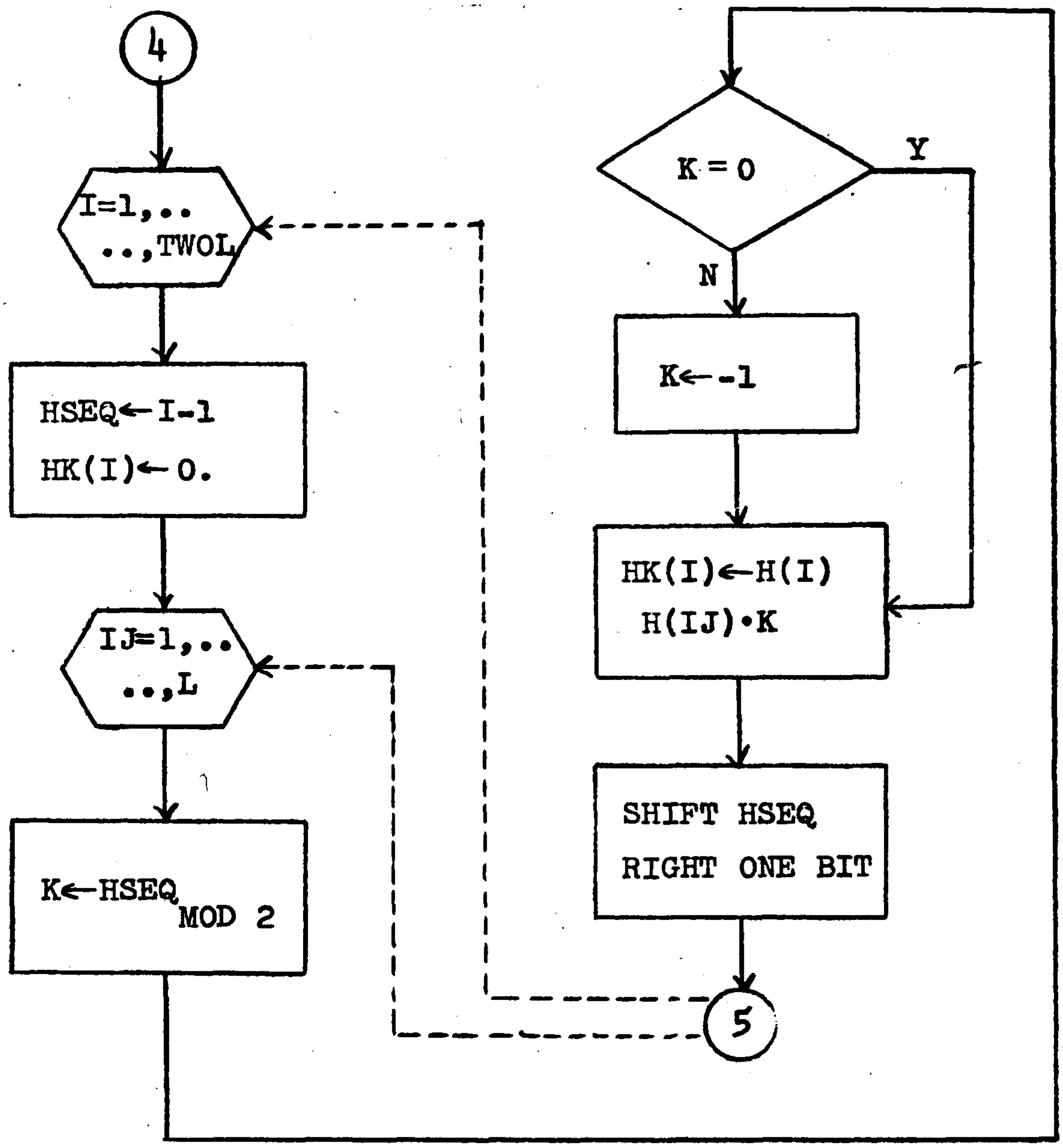


Fig. A4. Channel symbols.

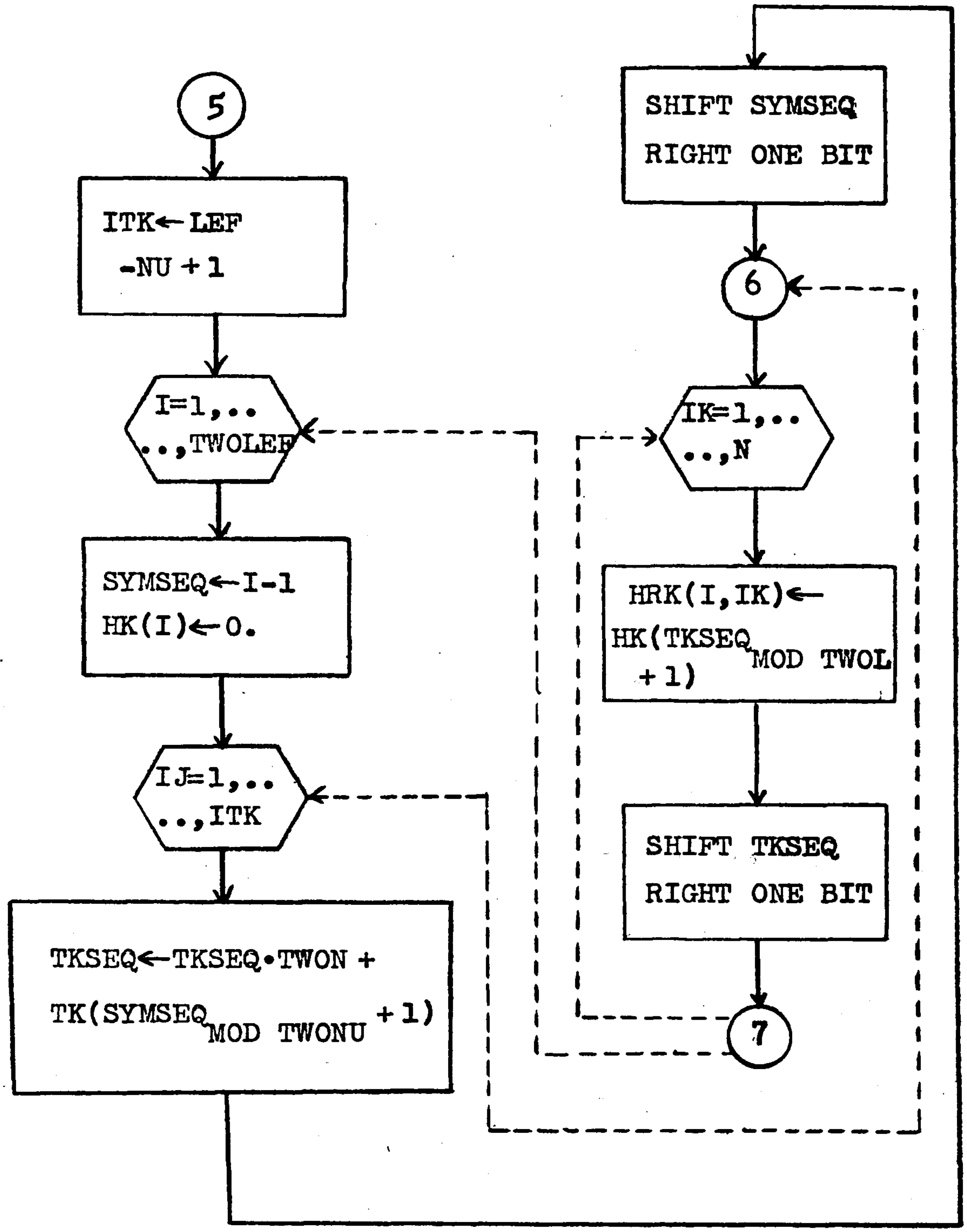


Fig. A5. Input sequences → output symbols.

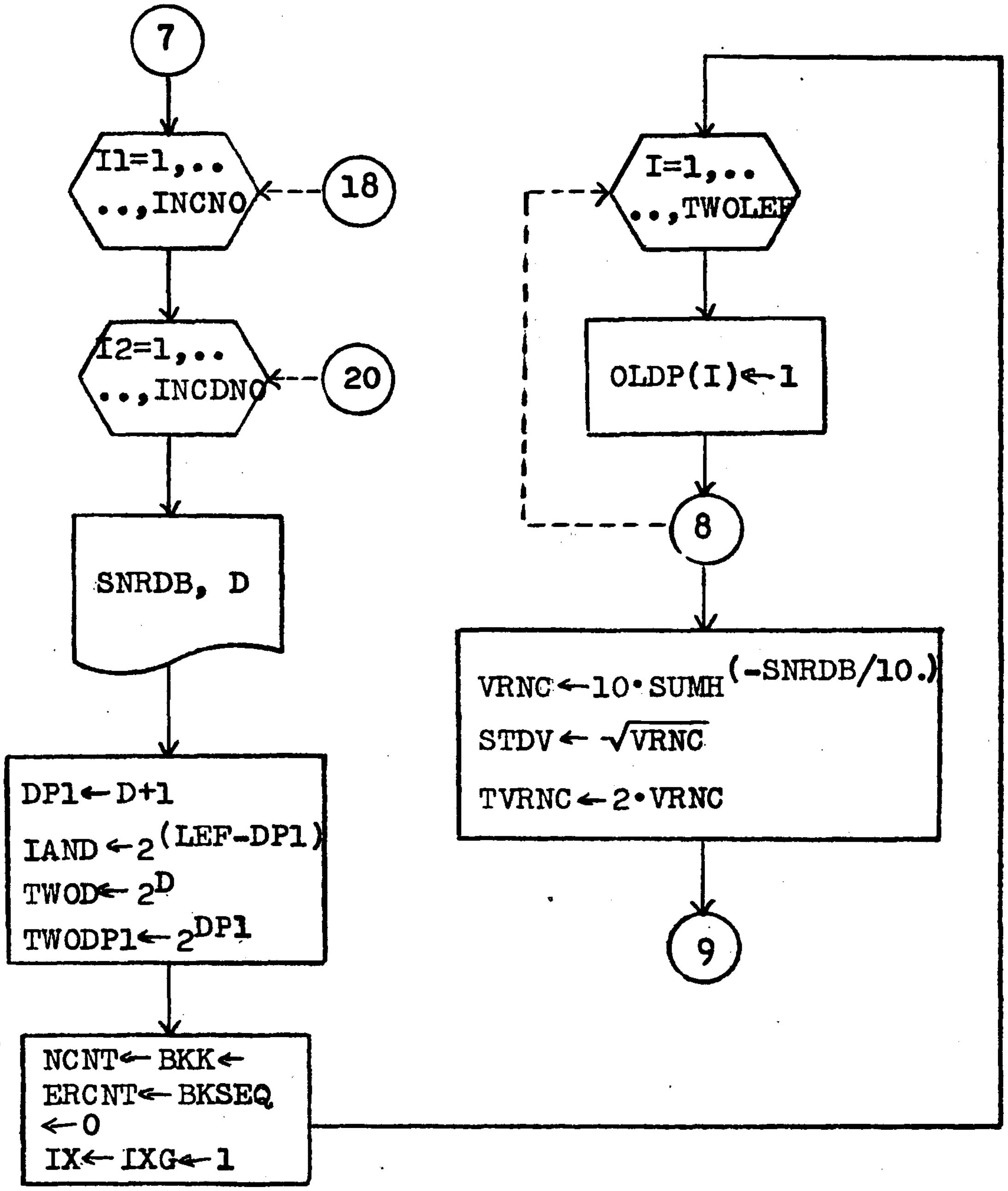


Fig. A6. Main Loop Initialization.

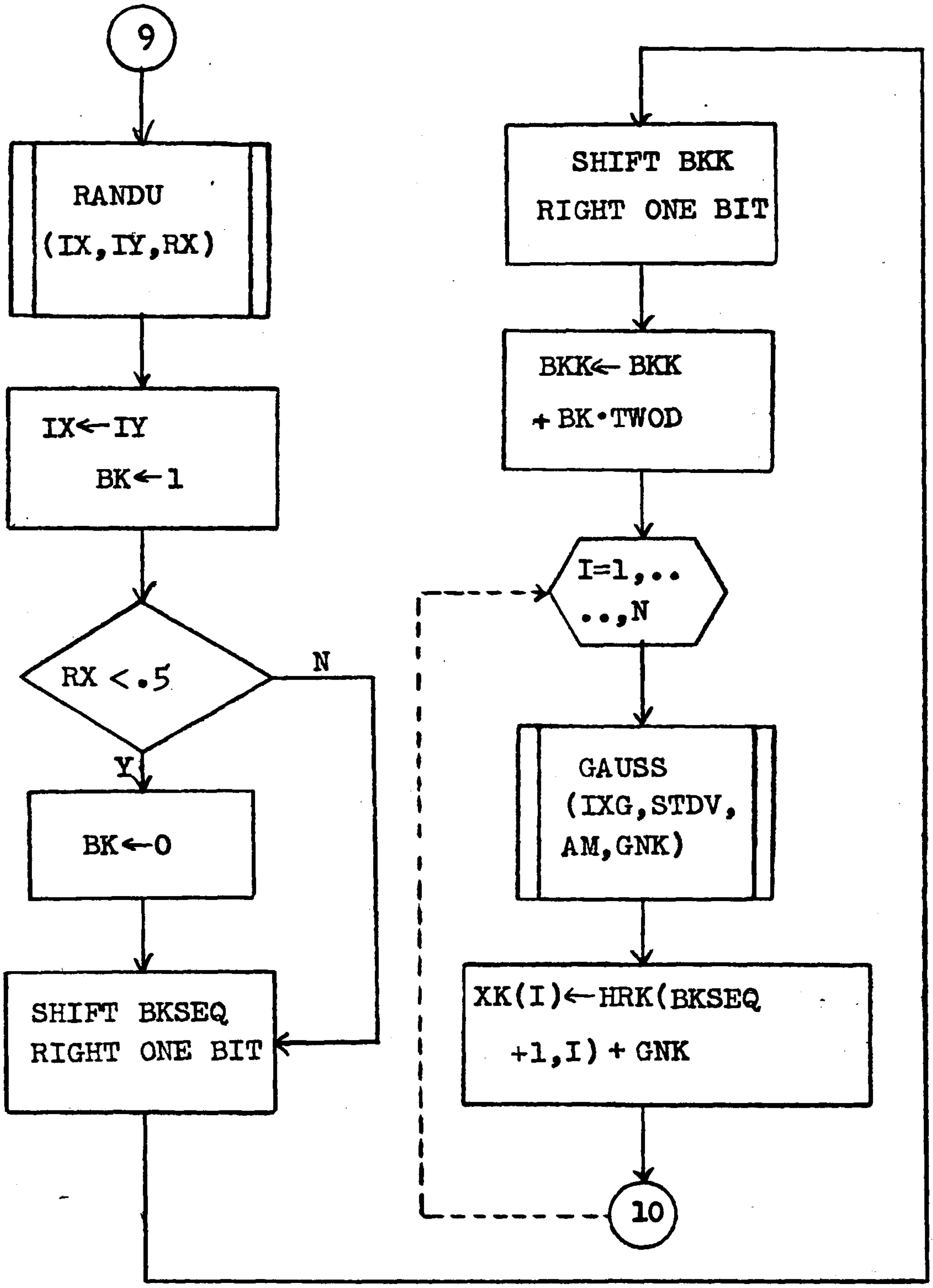


Fig. A7. "Transmitter."

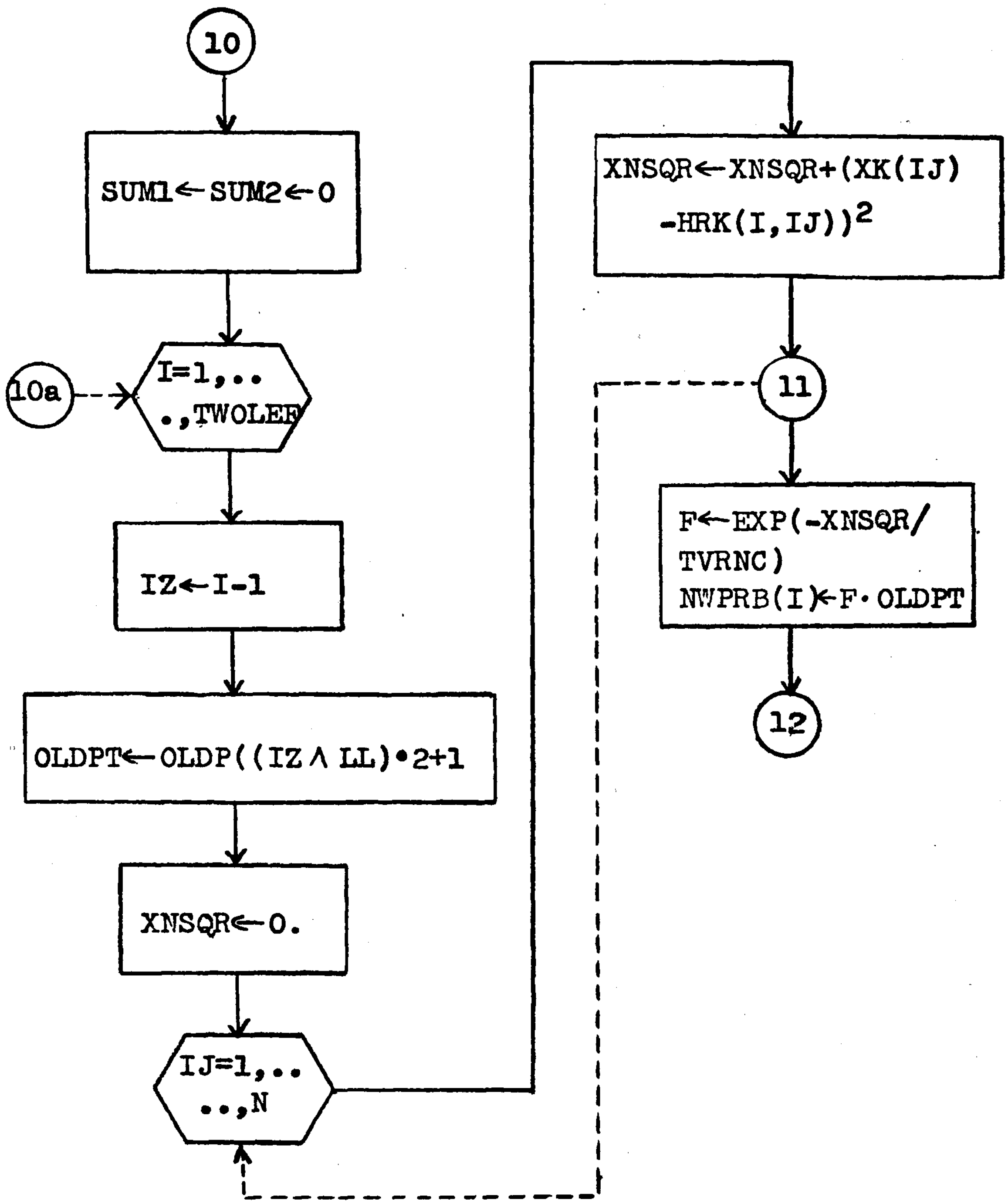


Fig. A8. Calculation of the incremental probabilities.

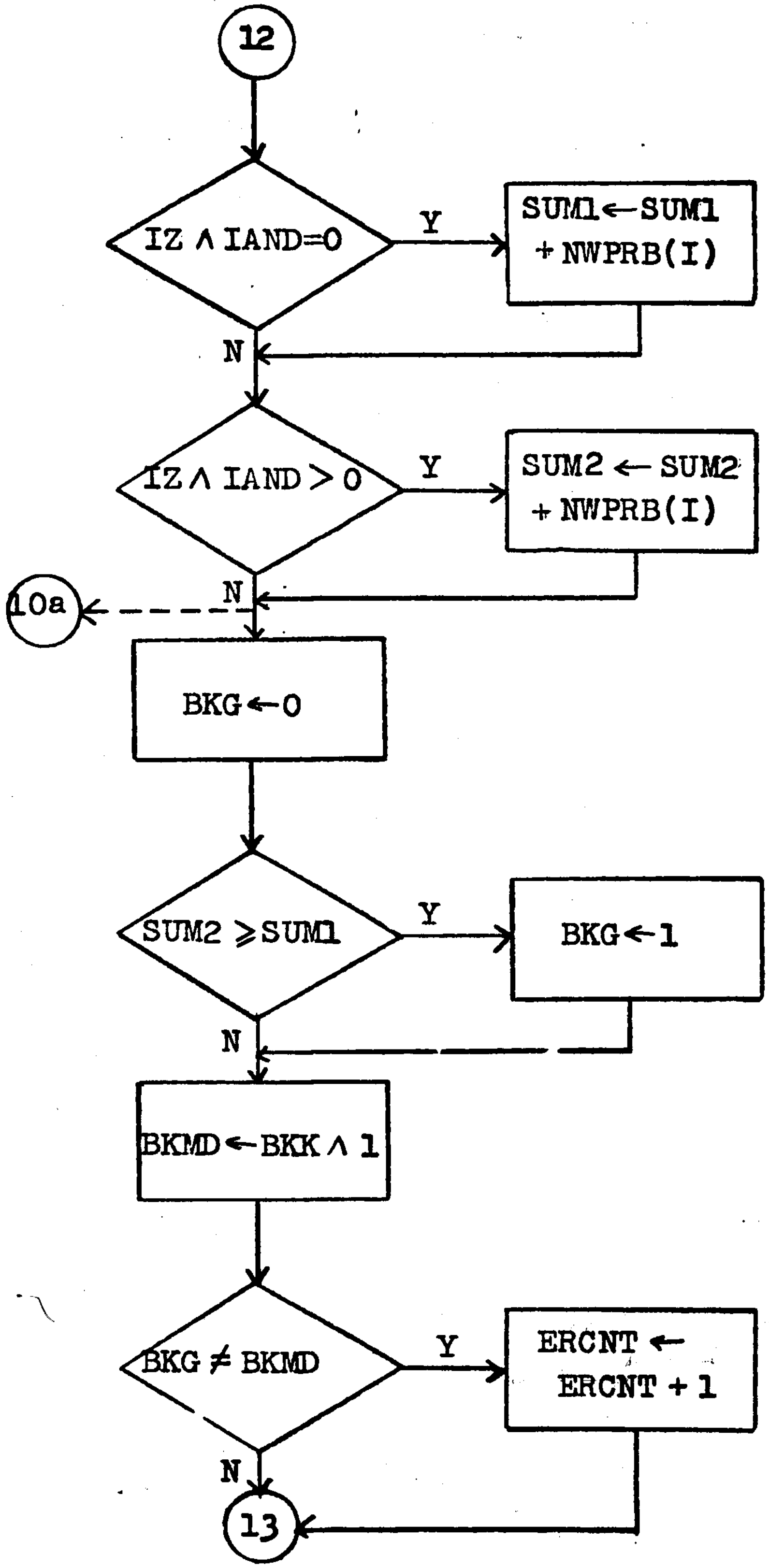


Fig. A9. Decision calculation.

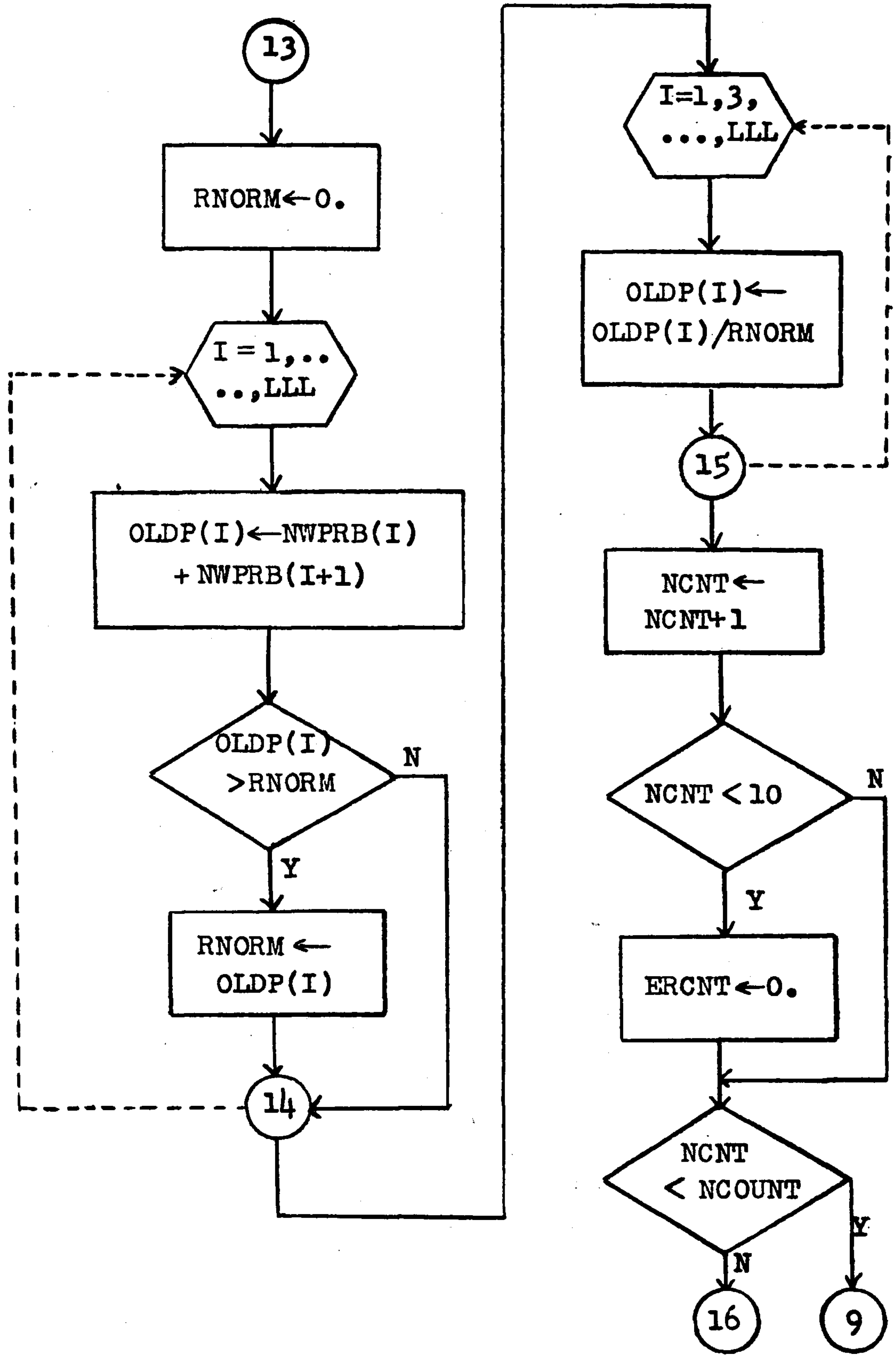


Fig. A10. Normalization of OLDP's and error summary.

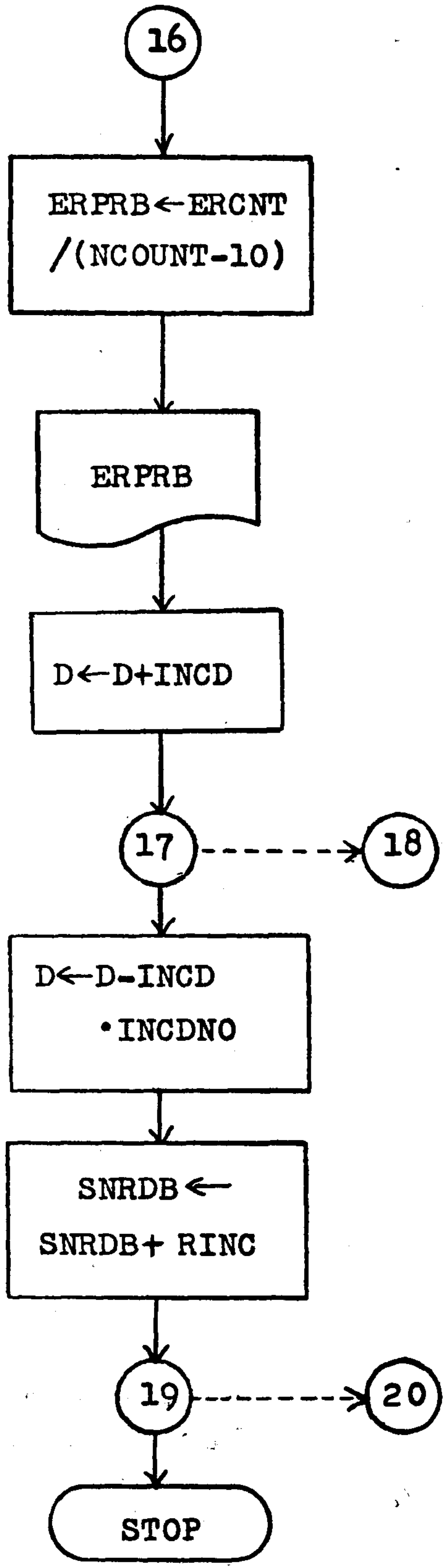


Fig. A11. Output and wrapup.

The modifications to the optimum nonlinear joint sequential detector-decoder appear in the flow-charts of Figures A12-A15.

Basically, the THRESHOLD (sub-optimum) program functions identically to the optimum program (see Fig. A12), except that only a fraction of the data manipulation is done, particularly in the segment where significant amounts of squaring and exponentiation are performed. This segment is bypassed whenever the variable OLDPT falls below the prescribed threshold. Fewer calculations result in a shorter program running time, or alternatively, less hardware, when parallel processing is performed.

The TOLERANCE algorithm, illustrated by Fig. A13, is similar to the THRESHOLD algorithm in that it bypasses many calculations, but the approach is different. Rather than examining OLDPT, which represents all the old information available on a symbol, this algorithm allows the noise estimate, XNSQR, to be computed for each allowable $\underline{R}_k$. All vectors not within the preset tolerance are eliminated.

The RANKING algorithm (Figs. A14-A15) is implemented in two segments. The first is the decision segment, similar to the TOLERANCE and THRESHOLD decisions. The second is the actual ranking segment which ranks the OLDP's and maintains the correlation between the

OLDP's and the NWPRB's affected by them. An interchange sort is used, and all OLDP's not within the group are set to zero. This particular program is quite inefficient, but generality, not efficiency, was stressed.

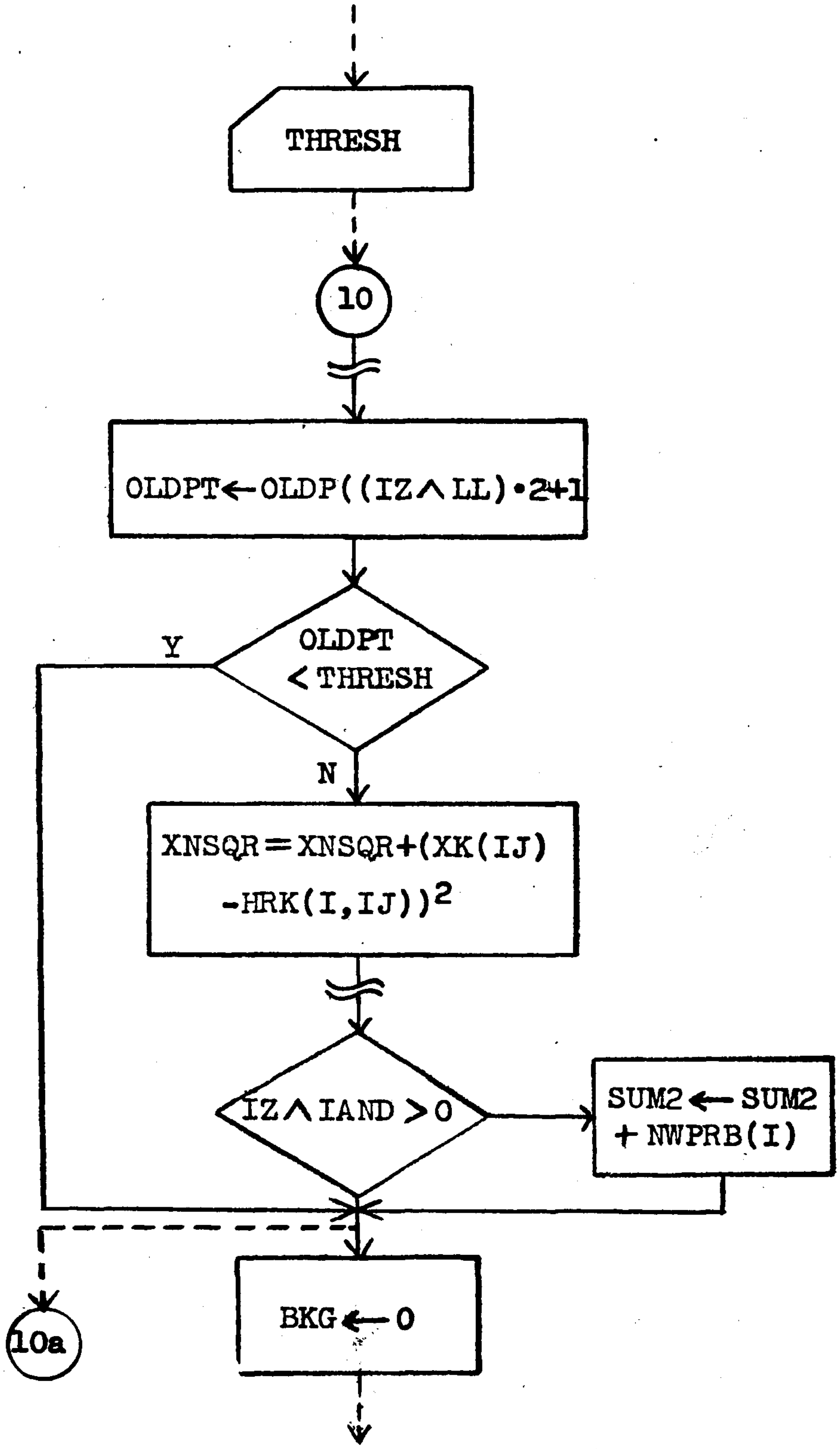


Fig. A12. Changes to optimum program to simulate THRESHOLD algorithm.

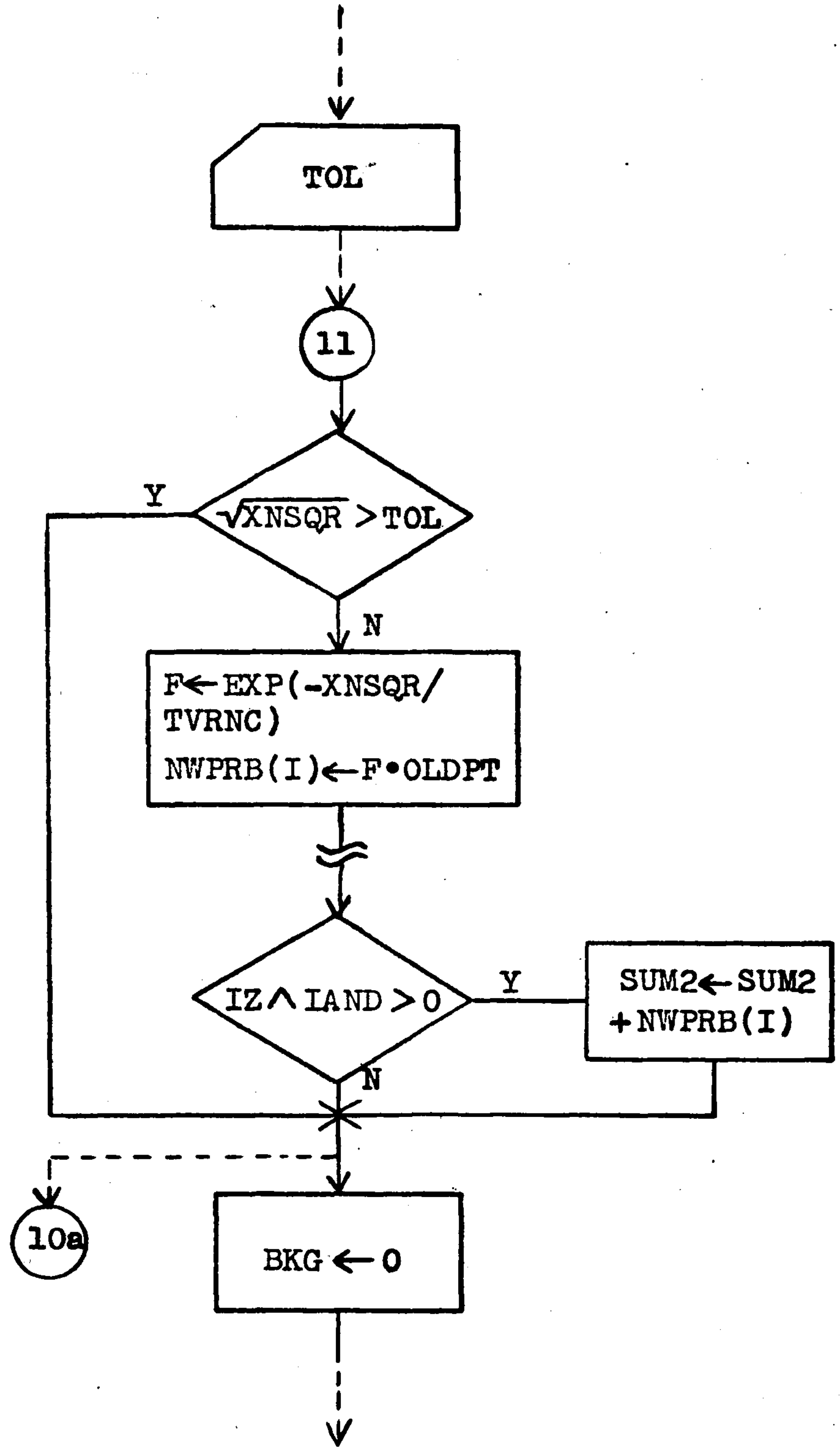


Fig. A13. Program flow for TOLERANCE algorithm.

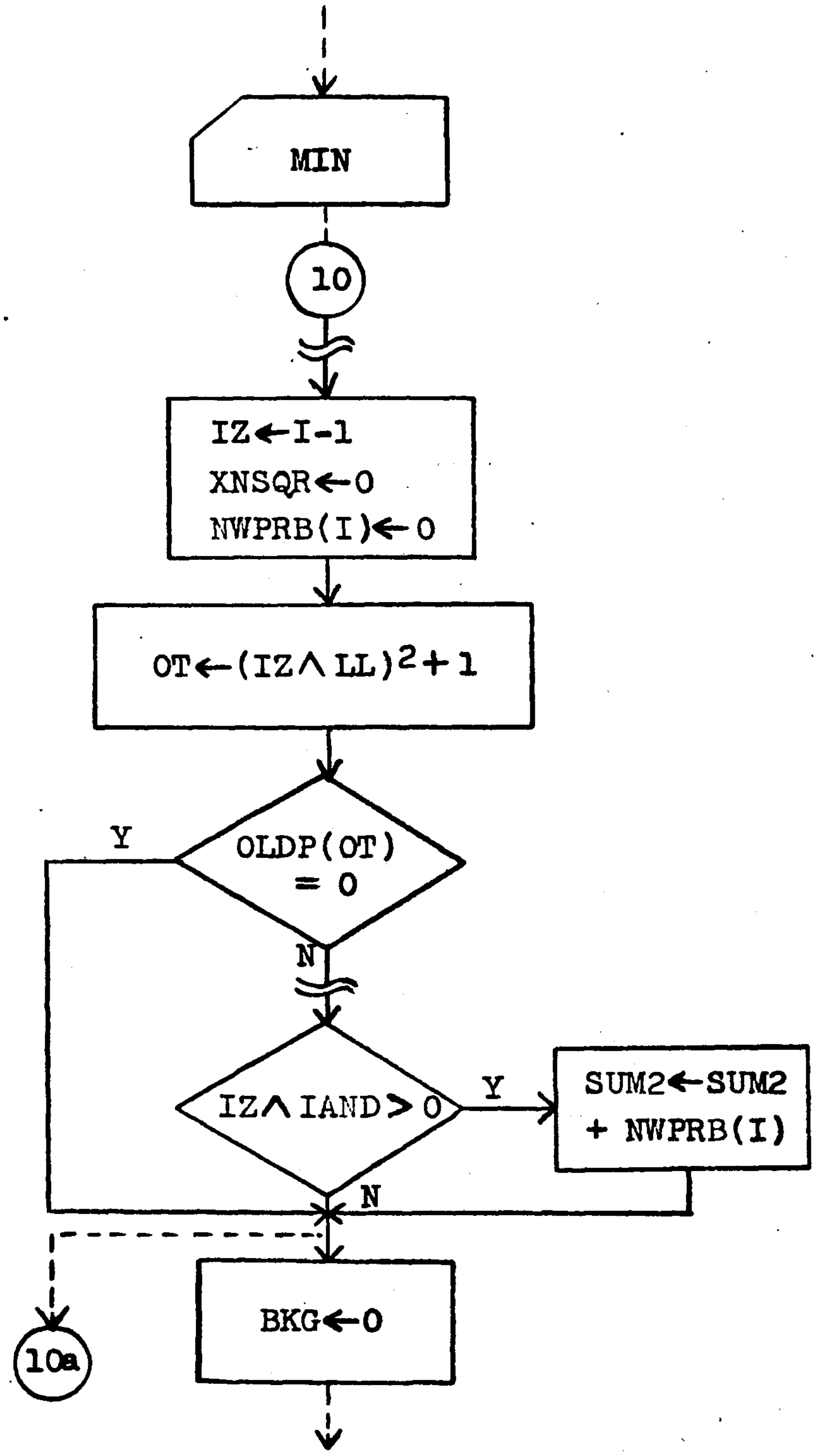


Fig. A14. Program flow for decision segment of RANKING algorithm.

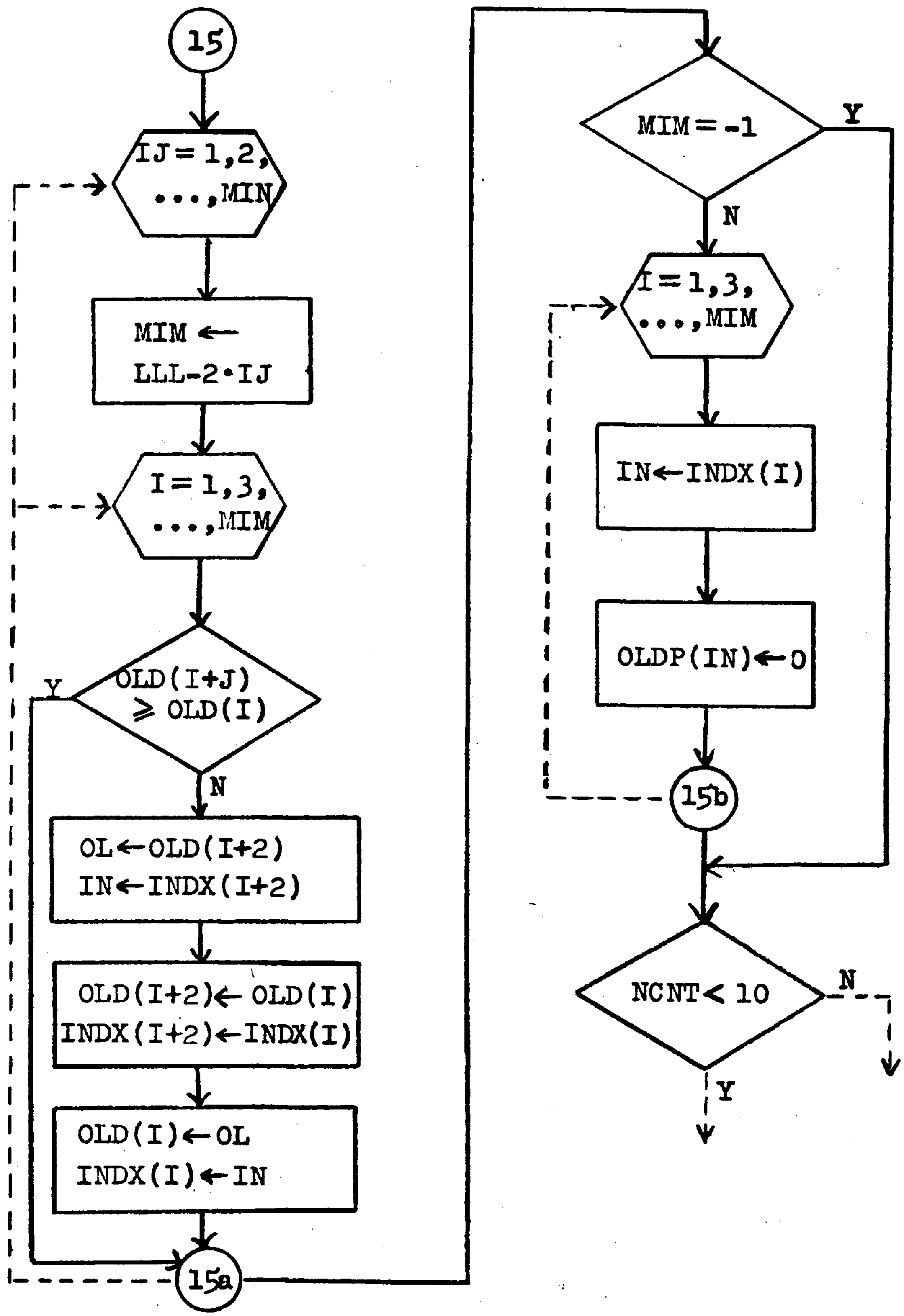


Fig. A15. Ranking segment of RANKING algorithm.

VITA

Clark Deturck Hafer was born in Philadelphia, Pa., on October 27th, 1953, the second son of Edith Hafer (nee Deturck) and the late Curtis S. Hafer, Sr.

Moving to Hatfield, Pa, in 1964, he later attended North Penn High School, where he was valedictorian of the class of 1971.

Hafer received his Bachelor of Science degree at Lehigh University, Bethlehem, Pa., in December, 1974, graduating with highest honors in the Department of Electrical Engineering. While an undergraduate at Lehigh he became a member of Eta Kappa Nu, Tau Beta Pi, and served as treasurer of the IEEE student chapter.

He is currently employed by the Space Division of General Electric Co. in Valley Forge, Pa.